THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITÉ DE MONTPELLIER

En informatique

École doctorale I2S

Unité de recherche LIRMM

# Génération procédurale d'environnements 3D sous-marins
# Procedural generation of 3D underwater environments


Présentée par Marc HARTLEY
le 15 décembre 2025

Sous la direction de Christophe FIORIO, Noura FARAJ
et Karen GODARY-DEJEAN


Devant le jury composé de

| | | | |
|---|---|---|---|
| Mme. | CANI Marie-Paule | Professeure des universités | Rapporteuse |
| M. | GUÉRIN Éric | Professeur des universités | Rapporteur |
| M. | BOUDON Frédéric | Maître de conférences | Examinateur |
| M. | MARI Jean-Luc | Professeur des universités | Examinateur |
| M. | BARTHE Loïc | Professeur des universités | Invité |
| Mme. | GODARY Karen | Maître de conférences | Invitée |
| M. | FIORIO Christophe | Professeur des universités | Directeur de thèse |
| Mme. | FARAJ Noura | Maître de conférences | Encadrante de thèse |

UNIVERSITÉ
DE MONTPELLIER

## Lay summary (EN)

This thesis contributes to the development of underwater robots for studying marine ecosystems. Since field missions are costly and difficult to organize, it proposes the creation of realistic virtual environments to test and validate these robots before deployment. The work explores the procedural generation of underwater worlds inspired by coral reefs, combining digital sketching, machine learning, and biological knowledge. It introduces methods to build large-scale coral landscapes, simulate the living organisms that inhabit them, and reproduce erosion and aging processes of marine terrains. These interactive and modular tools keep the user at the center of creation while integrating geological, biological, and oceanographic constraints. By bridging robotics, ecology, and computer graphics, this research opens new perspectives for virtual underwater exploration and robotic validation.

## Résumé vulgarisé (FR)

Cette thèse s'inscrit dans le contexte des processus de validation de robots sous-marins pour l'étude des écosystèmes marins. Les campagnes de test en mer sont coûteuses et difficiles à organiser, et l'environnement marin est difficilement controlable, ce qui rend compliqué de mettre en oeuvre des scénarios de tests spécifiques. Cette thèse propose donc de créer des environnements virtuels réalistes pour tester et valider ces robots avant leur déploiement. Le travail explore la génération procédurale d'univers sous-marins inspirés des récifs coralliens, en combinant dessin numérique, apprentissage automatique et connaissances biologiques. Il introduit des méthodes pour construire de vastes paysages coralliens, simuler la vie qui les peuple et reproduire l'érosion et le vieillissement des fonds marins. Ces outils interactifs et modulaires placent l'utilisateur au centre du processus tout en intégrant les contraintes géologiques, biologiques et océanographiques. En réunissant robotique, écologie et informatique graphique, cette recherche ouvre de nouvelles perspectives pour l'exploration sous-marine virtuelle et la validation robotique.

# Abstract (EN)

This thesis explores the procedural generation of 3D underwater environments to provide realistic, controllable, and reproducible testbeds for developing autonomous robots that observe marine fauna and flora. Because sea missions are rare, costly, and hazardous, creating credible virtual seascapes becomes essential for verifying and validating robot behavior, but also for observing and understanding the real world. Such environments must respect geological, biological, and oceanographic constraints while remaining sufficiently controllable to keep the user in charge.

This work presents a methodological framework for the design and modeling of underwater environments, with a focus on coral-reef islands. The manuscript is divided into three parts: large-scale generation of islands and their reefs via digital sketching; ecosystem population with biotic and abiotic elements through multi-scale simulation; and erosion simulation of virtual worlds to enhance realism.

The first contribution introduces a sketch-guided method for generating coral-reef islands. The user draws the plan view and elevation profile to build an initial height map of the island; a vector field representing winds and currents then deforms the landscape to simulate their long-term effects. An analytical model of coral-reef morphology is proposed to produce coherent height maps. We use this generation method to train a conditional Generative Adversarial Network (cGAN), which significantly reduces the constraints placed on the user during landscape design.

The second contribution proposes a multi-level semantic representation for ecosystem population. We describe the landscape using biotic and abiotic environmental objects endowed with attributes and generative rules. Our pipeline places and adapts objects while preserving global coherence and locally modifying the environment, thereby capturing ecological feedback loops between environment and objects without heavy simulation. The output is independent of the terrain representation as well as its resolution.

The third contribution introduces particle-based erosion for aging marine and terrestrial landscapes, applicable to all terrain representations (surface- and volume-based) and able to be coupled with an external fluid simulation. Thanks to their generality, particles can simulate many natural phenomena such as wind, rain, currents, waves, etc. Parallelization and controllability enable interactive use of this method.

Taken together, these three components form a coherent pipeline that links large-scale island design, ecological population, and terrain evolution down to robot-scale details, while keeping the user at the center of the process. The outcomes impact the fields of computer graphics, underwater robotics, and ecology.

# Résumé (FR)

Cette thèse explore la génération procédurale d'environnements sous-marins 3D afin d'offrir des cas de test réalistes, contrôlables et reproductibles pour la validation de robots autonomes d'observation des écosystemes marins. Les missions en mer étant difficiles, coûteuses et dangereuses, la création de paysages virtuels crédibles devient essentielle pour valider le fonctionnement d'un robot, mais aussi pour observer et comprendre le monde réel. Il faut des environnements à la fois fidèles aux contraintes géologiques, biologiques et océanographiques, et suffisamment contrôlables pour que l'utilisateur en garde la maîtrise.

Ce travail présente un cadre méthodologique pour la conception et la modélisation d'environnements sous-marins, en particulier des îles à récifs coralliens. Le manuscrit est divisé en trois parties : la génération à grande échelle d'îles et de leurs récifs par dessin numérique, le peuplement d'éléments biotiques et abiotiques dans le paysage par simulation d'écosystèmes multi-échelles, ainsi que la simulation d'érosion des mondes virtuels afin d'ajouter du réalisme.

La première contribution présente une méthode de génération d'îles coralliennes guidée par l'esquisse. L'utilisateur dessine la forme en plan et le profil altimétrique pour construire une première carte d'altitude de l'île ; un champ vectoriel représentant vents et courants déforme le paysage pour simuler leur effet à long terme. Un modèle analytique de forme de récif corallien est proposé pour produire des cartes d'altitude cohérentes. Nous utilisons cette méthode de génération pour entraîner un réseau génératif antagoniste conditionnel (cGAN), ce qui permet de réduire significativement les contraintes imposées à l'utilisateur lors de la conception de son paysage.

La deuxième contribution propose une représentation sémantique multi-niveaux pour le peuplement d'écosystèmes. Nous décrivons le paysage au moyen d'objets environnementaux biotiques et abiotiques dotés d'attributs et de règles de génération. Notre pipeline place et adapte les objets en respectant la cohérence d'ensemble et modifie localement l'environnement, ce qui permet de capturer des boucles de rétroaction entre l'environnement et les objets sans simulation lourde. La sortie de cette méthode est indépendante de la représentation du terrain, ainsi que de sa résolution.

La troisième contribution introduit une érosion par particules pour le vieillissement de paysages marins et terrestres, applicable à toutes les représentations de terrain (surfaciques et volumiques) et pouvant être couplée à une simulation de fluides externe. Grâce à leur généralité, les particules permettent de simuler de nombreux phénomènes naturels : vent, pluie, courants, vagues, etc. La parallélisation et le contrôle permettent d'utiliser cette méthode de manière interactive.

Ce travail en trois parties compose une chaîne cohérente qui relie la conception à grande échelle des îles, le peuplement écologique et l'évolution des terrains jusqu'aux détails perceptibles à l'échelle du robot, tout en conservant l'utilisateur au cœur du processus. Les retombées touchent les domaines de l'informatique graphique, la robotique sous-marine, et l'écologie.

# Résumé long (FR)

Cette thèse porte sur la génération procédurale d'environnements sous-marins tridimensionnels, avec une attention particulière portée aux récifs coralliens.
Elle s'inscrit plus largement dans le cadre d'une collaboration entre roboticiens, écologistes marins et informaticiens.
Le projet vise à développer des outils robotiques et informatiques pour mieux comprendre l'évolution de la biodiversité marine et faciliter l'observation écologique de long terme.

Les missions de robotique sous-marine présentent des défis majeurs : conditions environnementales imprévisibles, logistique coûteuse et risques matériels et humains élevés.
Dans ce contexte, la simulation virtuelle d'environnements réalistes offre une solution complémentaire permettant de vérifier et valider les comportements des robots autonomes dans des milieux complexes avant toute campagne en mer.
La génération procédurale d'environnements 3D vise précisément à produire de tels mondes virtuels crédibles, contrôlables et multi-échelles, tout en conservant l'humain au centre du processus créatif.

## Contexte scientifique et enjeux

Les océans recouvrent plus de 70 % de la surface terrestre et jouent un rôle essentiel dans la régulation du climat et la préservation de la biodiversité.
Les récifs coralliens, bien que couvrant moins de 0,1 % du plancher océanique, abritent près du quart des espèces marines connues. Leur observation régulière est indispensable pour détecter les changements écologiques et orienter les politiques de conservation.
Cependant, les méthodes classiques de suivi (plongeurs, comptages visuels, capteurs embarqués) demeurent coûteuses et limitées spatialement.

Les robots autonomes tels que REMI, développés au LIRMM, permettent de pallier ces limites.
Néanmoins, leur validation nécessite des environnements d'essai diversifiés et reproductibles, difficiles à obtenir dans le monde réel.
Les tests unitaires ou en bassin ne couvrent que des comportements simples, tandis que les campagnes en mer introduisent une grande variabilité (remous, turbidité, faune,

courants, etc.).
L'usage de simulateurs robotiques permet alors de tester des comportements complexes, mais la création de ces environnements requiert expertise logicielle et temps.
La conception automatisée de paysages sous-marins virtuels constitue donc un enjeu central, tant pour la vérification robotique que pour la modélisation écologique et l'informatique graphique.

# Objectifs de la thèse

Cette recherche explore de nouveaux moyens de concevoir et générer des environnements sous-marins procéduraux, plausibles et multi-échelles, afin de :

- Fournir des terrains de simulation réalistes pour la robotique et la modélisation écologique;
- Offrir des outils de création interactifs à la fois rapides, contrôlables et scientifiquement cohérents;
- Intégrer dans la génération des contraintes géologiques, biologiques et océanographiques.

La question centrale posée est la suivante : Comment guider efficacement l'utilisateur dans la création de mondes virtuels tout en maintenant un contrôle maximal sur le résultat final ?

# Contributions principales

La thèse s'articule autour de trois contributions complémentaires formant un pipeline complet de génération, de la conception sémantique d'un paysage jusqu'à sa simulation physique finale.

## Génération d'îles coralliennes par esquisse et apprentissage automatique

La première contribution introduit une méthode originale de génération d'îles volcaniques coralliennes à partir d'esquisses interactives en deux projections (plan et profil) couramment utilisées en géologie et en télédétection.
L'utilisateur définit la forme générale de l'île (vue de dessus) et son profil altimétrique (vue latérale) pour obtenir un premier modèle 3D.
Un champ vectoriel de vent, dessiné par l'utilisateur, déforme cette esquisse pour simuler les effets morphologiques à long terme du vent et des vagues.
Une fonction de résistance à la déformation contrôle la sensibilité des différentes zones (roche volcanique, plage, récif, fond marin) à ces forçages, permettant de reproduire des reliefs cohérents tout en restant interactif et intuitif.

Nous intégrons ensuite un modèle analytique de croissance récifale inspiré de la théorie de Darwin sur la formation des récifs coralliens.

Celui-ci simule l'évolution conjointe du volcan (subsidence) et du récif (croissance verticale) afin de reproduire les principales morphologies observées : récifs frangeants, récifs barrières et atolls.
Les résultats de ce modèle procédural servent à générer un ensemble de données synthétiques, composé de paires (cartes sémantiques, champs de hauteur), qui alimentent l'apprentissage d'un réseau antagoniste génératif conditionnel (cGAN pix2pix).
Ce dernier associe un générateur U-Net et un discriminateur PatchGAN, entraînés avec une perte combinée L1 + adversarielle, via l'optimiseur Adam, sur 10 000-20 000 échantillons issus du modèle procédural enrichis par augmentations géométriques et fusion douce smoothmax multi-îles.
Le réseau apprend ainsi la distribution statistique des structures coralliennes tout en relâchant les contraintes initiales (symétrie radiale, îlot central).
L'approche combine la rigueur géologique du modèle procédural et la flexibilité créative offerte par l'apprentissage automatique, ouvrant la voie à une génération d'îles coralliennes contrôlable, réaliste et diversifiée.

#### Aspects techniques développés

- **Interface à deux projections et esquisses paramétriques** Définition géométrique de l'île à partir de deux vues complémentaires (plan et profil) fixant structure et échelle.
- **Déformation contrôlée par champ vectoriel** Les tracés de vent produisent un champ anisotrope simulant les effets morphogéniques du vent et des vagues.
- **Fonction de résistance paramétrée** Régule localement l'amplitude des déformations selon profondeur et matériau.
- **Modèle analytique de croissance récifale** Couplage subsidence volcanique / croissance corallienne pour générer récifs frangeants, barrières et atolls.
- **Génération et augmentation de données synthétiques** Corpus structuré (cartes sémantiques-hauteur) enrichi par translations, rotations, mises à l'échelle et smoothmax multi-îles.
- **Apprentissage adversarial conditionnel** Entraînement du cGAN pix2pix (U-Net + PatchGAN) assurant cohérence locale et diversité morphologique.
- **Correction des biais procéduraux** Le réseau surmonte la symétrie radiale et le centrage imposés par le générateur procédural.
- **Interopérabilité et export** Sorties (cartes de hauteur et cartes sémantiques) directement exploitables dans des simulateurs robotiques ou moteurs 3D.

## Représentation sémantique des environnements

La seconde contribution introduit un cadre de représentation sémantique multi-niveau pour la description et la génération d'environnements sous-marins.
Alors que les approches classiques reposent principalement sur la géométrie des objets, notre modèle propose une structuration hiérarchique d'entités symboliques (coraux, roches, sédiments, algues, îles, etc.) reliées par des relations spatiales et écologiques.

Chaque entité est décrite par ses attributs géométriques (forme, orientation, échelle) et sémantiques (type biologique, conditions préférentielles, interactions locales).

Cette organisation sémantique assure la cohérence écologique et morphologique d'un paysage à différentes échelles, depuis la topographie globale jusqu'à la distribution locale des colonies.
Le modèle fonctionne comme une boucle de génération itérative : chaque placement est évalué selon des critères de viabilité, les champs environnementaux sont mis à jour (diffusion, advection, dépôt), puis de nouvelles entités apparaissent dans les zones restées favorables.
Les interactions reposent sur un schéma advection-réaction-diffusion appliqué à des champs scalaires et vectoriels (énergie des vagues, luminosité, composition du sol, humidité, sédiments).
Ces échanges produisent des effets d'auto-organisation comparables à ceux observés dans les récifs réels : zonation, densité différentielle, colonisation préférentielle.
La représentation, indépendante du support géométrique (carte de hauteur, voxels, maillages implicites), s'intègre aisément dans des pipelines de simulation ou de création et relie modélisation géométrique et écologique dans un cadre unifié.

#### Aspects techniques développés

- **Boucle itérative de génération et rétroaction** Cycle instanciation-évaluation-mise à jour couplant entités et champs environnementaux, garantissant convergence et stabilité.
- **Fonctions de fitness et fitting squelettique** Placement guidé par compatibilité environnementale et ajustement des courbes par minimisation d'énergie.
- **Modificateurs et propagation des effets** Opérateurs locaux (matériaux, flux, reliefs) simulant dépôts et perturbations hydrodynamiques.
- **Matériaux environnementaux paramétrés** Diffusion multi-couches (sable, calcaire, matière organique) sous advection contrôlée, sans solveur fluide explicite.
- **Événements géomorphologiques et interactivité** Injection d'événements globaux (tempêtes, dépôts, hausse du niveau d'eau) modifiant les champs écologiques tout en préservant la cohérence sémantique.
- **Compatibilité multi-représentations et export** Instanciation possible en cartes de hauteur, voxels ou surfaces implicites pour édition ou simulation.

## Simulation d'érosion physique

La troisième contribution présente un modèle générique d'érosion par particules, destiné à reproduire les processus de vieillissement et de remodelage des paysages terrestres et sous-marins.
L'approche repose sur une formulation lagrangienne : des particules indépendantes suivent la dynamique d'un fluide (air ou eau) et interagissent localement avec le terrain. À chaque collision, elles détachent, transportent puis déposent de la matière selon des lois physiques simplifiées dérivées de modèles de cisaillement pseudoplastiques et de

sédimentation.
Cette séparation explicite entre transport du fluide et modification du matériau permet d'appliquer le modèle à différents supports (cartes de hauteur, voxels, terrains à couches ou surfaces implicites) sans ajustement structurel majeur.

Chaque particule constite une unité indépendante, permettant une implémentation parallèle.
Le simulateur atteint ainsi des performances interactives tout en reproduisant des effets de long terme tels que abrasion, dépôt sédimentaire ou formation de pentes d'équilibre.
Ce cadre unifié couvre aussi bien des phénomènes atmosphériques que sous-marins (pluie, vent, houle, courants) et peut modéliser l'érosion chimique ou le scouring autour d'obstacles.
Il s'intègre dans des pipelines de rendu, de génération procédurale ou de validation robotique.

#### Aspects techniques développés

- **Schéma lagrangien indépendant** Chaque particule suit un cycle détachement-transport-dépôt, sans interaction inter-particules, garantissant parallélisation et stabilité.
- **Découplage fluide / matériau** Le champ de vitesse est soit prescrit, soit fourni par un solveur externe (SPH, FLIP), permettant un contrôle flexible.
- **Modèle d'abrasion et de dépôt** Les quantités érodées et déposées dépendent de lois pseudoplastiques et de vitesses de sédimentation régies par densité, viscosité et capacité de charge.
- **Agnosticité représentationnelle** Schéma commun pour cartes de hauteur, terrains à couches, voxels et surfaces implicites.

## Apports et retombées

Réunies, ces contributions forment un pipeline cohérent plaçant l'utilisateur au centre : du grand paysage (définition d'îles et plateformes) jusqu'au détail robotique (micro-habitats, obstacles, turbidité), avec traçabilité des choix et contrôles explicites à chaque étape.
L'approche combine modèles informés par les processus, règles écologiques simplifiées et expertise humaine, conciliant exploration créative et exigence scientifique.

Sur le plan applicatif, ces outils ouvrent plusieurs perspectives :

- Robotique sous-marine : création d'environnements de test pour la navigation autonome, la vision par ordinateur et la planification de trajectoires;
- Informatique graphique et divertissement : génération de mondes sous-marins crédibles pour le cinéma, les jeux vidéo ou la réalité virtuelle;
- Écologie et géosciences : exploration virtuelle d'hypothèses sur la formation, l'évolution et l'érosion des récifs coralliens.

## Perspectives

Plusieurs prolongements sont envisagés :

- (i) procédurale inverse pour inférer des règles à partir de scènes observées;
- (ii) guidage par données utilisateur pour capturer le style créatif et adapter la génération;
- (iii) modèles substitutifs appris pour accélérer le vieillissement sous contrainte de fidélité;
- (iv) couplage plus étroit avec des solveurs d'écoulement lorsque la précision hydrodynamique est requise.

## Conclusion

La thèse démontre qu'il est possible de concevoir des environnements sous-marins virtuels à la fois réalistes, contrôlables et scientifiquement crédibles, en combinant modélisation procédurale, apprentissage automatique, représentation sémantique et simulation physique.
En plaçant l'utilisateur au centre de la boucle de création, elle dépasse les approches purement automatiques pour proposer des outils collaboratifs entre humain et machine.

# Contents

Chapter 1

# Introduction

The planet is currently experiencing an ecological crisis marked by global warming, the accelerated loss of biodiversity, and the degradation of ecosystems. The oceans, which cover more than 70% of the Earth's surface, play a crucial role in climate regulation and in sustaining life on Earth [PEV20, Vis18]. Among them, coral reefs occupy a unique place: although they cover only a small portion of the ocean floor, they are home to about 25% of known marine species. Additionally, these living structures play an essential role in protecting coasts from erosion, support fisheries that are vital for millions of people, and underpin tourist activities that are crucial for most island economies [FBS*14, SBW*17, PCBK11].

In this context, biodiversity monitoring has become a strategic priority. Tracking the dynamics of marine communities makes it possible to identify threats rapidly, assess the effectiveness of marine protected areas, and adapt conservation policies [HPSD17, YLPY24]. Yet underwater monitoring remains challenging: traditional methods based on divers or visual counts are costly, limited in time and space, and subject to observation bias [EBM04]. In coral reefs, where conditions are complex (low light, irregular currents, fragile habitats), reliable and frequent surveys are indispensable for ecologists to understanding and anticipating ecological change [MLD*21].

It this context the BUBOT project (Better Understanding Biodiversity changes thanks to new Observation Tools) was launched. It is an interdisciplinary project led by the LIRMM (University of Montpellier, CNRS) in collaboration with MARBEC (marine ecology), Espace-Dev (geography and anthropology), CUFR Mayotte, Universidade Lúrio in Mozambique, and the University of California, Davis. BUBOT combines robotics, computer vision, ecology and the social sciences to improve the understanding of biodiversity change and to provide robust tools for the long-term monitoring of the reefs of Mayotte and the Sparse Islands, in the Mozambique Channel. Among its major achievements is the development of the REMI underwater robot, designed to automate ecological surveys [HGG*20].

Using a robot offers several decisive advantages. Unlike divers, REMI can explore great depths or hazardous areas, collect standardised data over long periods, and operate in conditions that would be impossible or risky for a human [GGC*20]. Its video recordings make it possible to identify species using artificial intelligence algorithms [VMC*20, LBZ*24] and to analyse reef structure with increased precision through photogrametry [NMG*20].

REMI is an underwater robotic platform developed at LIRMM. Equipped with autonomous navigation systems, and acoustic and optical sensors, it has been tested in a range of environments such as Mayotte and the Mediterranean Sea [HGG*20, MLD*21]. These campaigns demonstrated its potential but also highlighted the complexity of missions in natural environments: rugged topography, unpredictable currents, variable turbidity, and technical limitations such as power autonomy or underwater communications.

Testing in robotics generally follows a multi-stage process [GGC*20]. First, unit tests verify the

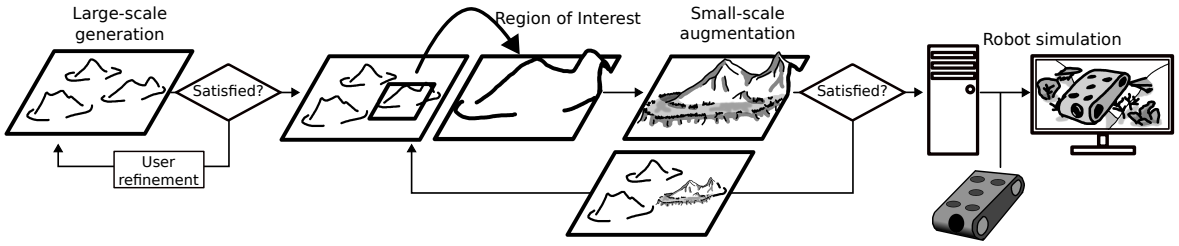



**Figure 1.1:** *The global objective of this project is to propose a pipeline from large-scale landscape generation to robot-scale environment generation. The large-scale modelling allows to make sure the environment will be consistent with the target context. Being able to navigate in different scales permits to focus details in certain regions for the final result, or to iterate front-and-back with different area of the landscape until satisfaction.*

proper functioning of each component (motors, cameras, sealing systems). Next come pool tests, easier to organise but far from real conditions: fresh water does not test resistance to salt, artificial waves often reduce to simple sinusoidal shapes, there are neither complex ocean currents nor realistic turbidity, sensors perform unrealistically well due to the absence of noise, and there is no fauna or flora. Navigation in natural currents and natural seabed cannot be properly evaluated in a swimming pool. Finally, field tests provide validation under real conditions but pose serious challenges: high financial and logistical costs, mobilisation of large teams, dependence on a particular site (finding a fault, an arch or a given type of fauna is not guaranteed), and risk of loss or damage to the robot. Recovering a failed robot may require risky human intervention and prematurely end a mission if spare parts are not available. Moreover, each field mission allows the testing of only a single combination of conditions, leaving many potential situations unexplored.

REMI's navigation must also contend with specific obstacles. A canyon should be followed down to the bottom, but a perpendicular crack must be avoided to prevent collision with the opposite wall. Arches and overhangs require avoiding unwarranted ascents to the surface in the presence of a rock ceiling. Caves present an even more complex case: although the walls may appear distant, the robot must recognise that it is trapped and turn back. The presence of mobile animals further complicates decision-making: a school of fish can appear as a moving wall that the robot must interpret without endangering itself or the fauna. Coral reefs themselves pose problems: their porosity can mislead range sensors; algae such as *Posidonia*, which are difficult to detect, can damage the propellers; and fine sand can be lifted into suspension by the thrusters, completely obscuring optical sensors.

These challenges illustrate the difficulty of reproducing in real conditions all the situations that REMI may encounter. Hence the growing interest in realistic virtual environments instead of a real one in robot control loop (Figure 1.2). 3D modelling of underwater environments make it possible to simulate the presence of canyons, arches, fauna, turbulence, or turbidity conditions that are almost impossible to bring together in a single mission. It reduces costs, avoids the risk of equipment loss, and accelerates the development of decision-making algorithms.

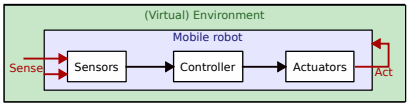



**Figure 1.2:** *A generic control loop structure for a mobile robot [Her22]: the robot inside an environment, natural or virtual, uses sensors to sense its surrounding, process it with its controllers, and activate actuators to react. Well defined virtual environments should be indistinguishable to a real one for the robot.*

These constraints highlight the need for controllable and realistic virtual testing environments. This work therefore focuses on the modelling of virtual terrains designed to host a digital underwater observation robot, offering a flexible testbed for validating its behaviour under well-defined conditions. Procedural generation is the main method for avoiding the tedious task of creating large-scale and small-scale 3D scenes by hand. The global pipeline of our approach (Figure 1.1) keeps the user involved throughout the environment creation process, enabling progressive refinement from large-

scale landscapes down to small-scale, robot-level details.

Over the past 50 years, terrain generation has emerged as an increasingly active domain within the field of computer graphics [FFC82, MKM89, Mil86, GGP*19]. As the demand for realistic and automated processes has grown to support the creation of ever-larger and more detailed landscapes effortlessly, terrain generation techniques have evolved accordingly. As these goals are progressively achieved, a new trend shifts the focus towards greater user control. This work is focused on a specific branch of terrain generation: modelling and authoring of landscapes.

Landscape generation finds applications in diverse fields such as biology, geology, and robotics [THR23, CWL*23, Ger25, RHRH22]. Additionally, it has become a central tool in the entertainment industry, particularly in video games and cinema, where creating realistic and dynamic environments is key for immersive experiences. The ability to generate landscapes that help in understanding natural rules and testing hypotheses makes this the ideal tool for both researchers and industries.

However, this exploration of the domain comes with significant challenges. In the case of underwater environments, finding a balance between automation and user expectations is particularly complex: the visual and physical rules governing seascapes (such as coral growth patterns, sediment deposition, erosion by currents, or the interaction of light with water) differ markedly from those of terrestrial landscapes. Managing scaling is also more challenging in this context, as underwater landscapes often involve vast bathymetric structures while simultaneously requiring fine-scale details (e.g., coral colonies, rocks, or vegetation) to remain scientifically relevant or visually convincing.

The central question guiding this research is: "How can we efficiently guide the user in the creation of virtual content along the production process line to maintain as much control as possible over the final product?". This concept of "guiding" rather than "replacing" the user is, from my point of view, fundamental, as no machine can truly know better than a human what the final product should look like. The goal is to present algorithms that can be used flexibly within a production pipeline, adapting to many terrain representations, fluid solvers, landscape types, users' hardware, or objectives, whether the final use is real-time rendering, realism, or animation.

Maintaining control over the creative process is essential. Each generated result should meet the user's expectations, be explainable, and be easily correctable without requiring a complete regeneration. This approach ensures that the user remains at the centre of the creative process, with the tools and algorithms serving to enhance their objectives, rather than replacing the users themselves.

What sets this work apart is its emphasis on underwater landscapes or "seascapes", an area only slightly touched upon in Computer Graphics, but important in domains such as marine biology, oceanography, and underwater robotics. Furthermore, the extension of these techniques to new areas for the entertainment industry, such as video games and cinema, opens the door to novel visual experiences and storytelling techniques that take full advantage of the unique aesthetic and physical characteristics of underwater environments.

## 1.1 Underwater landscape modelling challenges

Seascapes encompass the complex and dynamic terrains found beneath the ocean surface. Similarly as landscapes above water level (which we will call "aerial landscapes"), underwater environments are shaped by geological forces. However, biological activity, hydrodynamic processes, and chemical interactions represent significantly more important factors of their formation and evolution than for aerial environments. Coral reef islands represent a particularly intricate subset of these environments, where the terrain is actively constructed and modified by living organisms, primarily reef-building corals. The spatial and structural complexity of such systems, which span multiple scales from individual coral polyps to entire reef platforms, poses unique challenges for their representation and simulation.

In the context of procedural terrain generation, the modelling of underwater landscapes demands

fundamentally different assumptions and techniques from those used for land-based terrains. The submerged setting alters the influence of gravity, light, and fluid dynamics, while also introducing significant observational constraints. Furthermore, the scarcity of high-resolution and volumetric data capturing the structural complexity and biological processes of coral reef systems poses a major challenge for data-driven or example-based modelling approaches. In this context, procedural generation offers an alternative by synthesising underwater landscapes from first principles, guided by interdisciplinary understanding of geological, biological, and hydrodynamic processes. This section reviews the key environmental, observational, and modelling challenges associated with seascapes, with a focus on those that motivate and constrain procedural terrain generation for coral reef island systems.

### 1.1.1 Physical environment differences

The underwater environment differs fundamentally from terrestrial settings in the physical forces that shape landscape formation, the conditions for biological growth, and the constraints imposed on sensing and modelling. These differences directly impact the assumptions underlying terrain generation and limit the applicability of methods developed for aerial landscapes.

First, although gravity remains a dominant force in geomorphological evolution, provoking sediment transport and slope stability, buoyancy significantly alters the behaviour of materials and organisms underwater. The net effect is a reduction in apparent weight and in gravity-driven surface processes, particularly at smaller scales, where water movement becomes a dominant shaping force. Hydrodynamics plays a central role in shaping the morphology of coral reef systems by redistributing sediments, constraining reef growth zones, and sculpting reef edges and channels [LF15].

Second, optical properties of seawater impose strict environmental constraints. Light attenuation limits the available area for photosynthesis, which in turn defines vertical growth boundaries for reef-building corals [Hus85]. It also constrains the applicability of remote sensing and optical measurement techniques, limiting their effectiveness to shallow, clear-water environments.

Third, underwater terrains are shaped not only by physical processes but also by biological construction and erosion. Reef-building organisms contribute directly to topographic complexity through accretion, while bioeroder species like parrotfish, sponges, and urchins actively degrade and remodel the substrate [PMK*13].

Finally, these coupled physical-biological interactions result in terrain features that are not only highly complex, but also scale-dependent and dynamic. Overhangs, cavities, and porous structures restrict standard heightmap-based representations. Furthermore, many features emerge from feedback loops between hydrodynamics, sedimentation, and biological growth, further complicating attempts to generalise aerial terrain modelling techniques to underwater environments.

### 1.1.2 Observation and 3D data acquisition challenges

Despite growing interest in seafloor mapping, high-resolution, volumetric datasets of coral reef environments remain scarce. Most available data are limited to bathymetric or altimetric surface representations, typically acquired through sonar or satellite-derived methods, which offer only 2.5D elevation maps and lack information on vertical and sub-surface complexity. Features such as overhangs, internal cavities, and fine-scale biological structures are not captured, limiting the fidelity of directly observed data.

Moreover optical or active remote sensing techniques, such as underwater photogrammetry or bathymetric LIDAR, are constrained to shallow and optically clear waters, and environments that are safe for the diver or robot carrying the sensor. These methods are ineffective in turbid or deep environments, which characterise large portions of reef systems, particularly mesophotic and fore-reef zones. As a result, consistent 3D coverage across depth gradients is complex and time consuming with current

technology.

In situ surveys are limited by logistical, financial, and environmental factors. Diver-based surveys, effective at fine scales, are labour-intensive, spatially limited, and can disturb local fauna, thereby introducing bias in biological assessments. Remotely Operated Vehicles (ROVs) offer greater depth access but require tethering and constant supervision. Autonomous Underwater Vehicles (AUVs), in contrast, have shown promise in collecting low-disturbance imagery and acoustic data in a more scalable manner [GBR*16, MRYR18], and represent a potential pathway for systematic 3D surveying in the future.

### 1.1.3 Environmental dynamics and interdisciplinary integration

Coral reef terrains develop in relation to underlying geological structures and sedimentary processes. Substrate stability and morphology are influenced by sediment supply which controls the distribution of fine sediments in lagoons versus coarser rubble zones on reef fronts [Mon05]. Tectonic uplift or subsidence modulates relative sea level, determining exposure intervals that drive the formation of reef terraces and platform architecture [Hop14]. Geomorphologic events, such as submarine landslides or turbidity currents on reef slopes, and wave-driven abrasion, further reshape bathymetry at meso- to macroscale.

Hydrodynamic regimes strongly influence sediment transport, mechanical stress, and ecological viability in reef environments. Persistent currents and wave action shaping erosion and deposition patterns on slopes and flats, shape reef crests, and drive overtopping and flushing in lagoonal areas [LFMA09]. Episodic storms or large swell events can produce rapid morphological change via scour, sediment redistribution, and coral breakage, punctuating longer-term growth trajectories. The evolving reef morphology modifies local flow fields which in turn influence subsequent sediment dynamics and biological processes.

In parallel to erosion processes driven by geological, climatic and hydrological events, the organic nature of coral reef continuously remodels and regenerates the substrate. However, abiotic factors such as light attenuation, temperature regimes, and water chemistry constrain both biological growth and sedimentary behaviour. Light attenuation due to turbidity and depth limits photosynthesis, setting depth boundaries for coral accretion [Kir94]. Nutrient concentrations from terrestrial runoff can shift community composition towards macroalgae or favour coral health. Turbidity governs sediment deposition on reef surfaces.

Effectively integrating geological, ecological, hydrodynamic, and chemical knowledge requires the reconciling of diverse data types and spatiotemporal resolutions. Geological data often spans over millennial time scales and kilometre space scale, whereas ecological observations like colony growth rates are studied in years and counted in metres; hydrodynamic models and sensor time series each have their own spatiotemporal resolutions.

The complex, dynamic nature of reef environments motivates the use of process-informed procedural generation rather than example-based interpolation. Algorithms should embody process-based constraints such as depth-dependent accretion limits, susceptibility to storm-induced degradation, and form-flow feedbacks while allowing staged synthesis to yield realistic structural patterns. Simplified feedback loops, such as adjusting local growth likelihood based on simulated flow attenuation, can capture essential interactions without full physical simulation. Given data scarcity, procedural outputs should support exploration across parameter spaces or alternative scenarios, making assumptions transparent.

### 1.1.4 Procedural modelling challenges

Paris classifies geological structures in four spatial scales [Par23]: the *megascale* describes landforms spanning over more than 100km such as continents or mountain ranges, the *macroscale* between

1km and 100km (e.g., islands, rivers or cave networks), the *mesoscale* ranging from 10m to 1km contains most complex structures (e.g., arches, ravines, cliffs, ...) and finally the *microscale* for features between 10cm and 10m, (e.g., sand ripples, ventifacts, rocks). Generally, we consider as *large-scale* features that can be seen from far away while *small-scale* elements requires to be within arm's length to be observed.

Coral reef island landscapes exhibit structural complexity across scales ranging from millimetre- to kilometre-scale features, and procedural generation must support seamless transitions among these scales. At the microscale, individual coral morphologies, crevice networks, and sediment textures determine fine habitat details, whereas at the mesoscale, colony aggregations, reef crests, and lagoon basins define intermediate morphology. Macroscale features include the overall platform shape, island emergence, and large geomorphic structures such as reef rims or terrace outlines.

Unlike aerial landscapes' height fields, reef environments contain overhangs, cavities, and porous frameworks created by biological accretion and bioerosion. Procedural methods must therefore synthesise plausible volumetric geometry rather than relying solely on 2.5D surfaces. This could be achieved through combinations of multiple surface and volume representations, the introduction of rule-based solid modelling such as the use of L-systems for branching coral forms [PL92, ALY15], and the stochastic placement of "voids" to mimic bioerosion. A procedural pipeline may first establish a coarse reef framework from depth-dependent envelope shapes and then instantiate volumetric modules representing coral colonies and cavities according to ecological rules or bioerosion simulations. The main challenge for such method is to integrate ecological rules and maintain computational tractability.

Physical processes such as hydrodynamic forces and sediment transport impose constraints on feasible reef morphologies. While full Computational Fluid Dynamics (CFD) coupling for procedural generation is typically impractical at large scales, hybrid approaches could incorporate simplified physics-based rules. For example, a priori knowledge of relationships between reef shape and prevailing wave energy informs the geometry of grooves-and-spurs. Sediment deposition rules from in situ observation can guide the formation of lagoon floors or backreef accumulations. Such hybrid modelling ensures that generated forms remain within plausible physical bounds.

Process-informed generation relies on encoding ecological constraints and interactions. Coral growth algorithms may simulate accretion under light-dependent rules according to environmental parameters. Agent-based models can represent competition for space, where faster-growing but fragile forms may dominate in sheltered zones, whereas robust morphologies persist in high-energy areas [AF00]. Simulated bioerosion events introduce heterogeneity and temporal variability. Although full temporal simulation may be computationally expensive, stochastic modelling or staged synthesis (e.g., iterative growth phases punctuated by disturbance events) may yield realistic geometry.

An effective procedural modelling framework for coral reef islands begins by defining a macroscopic envelope (i.e., reef and island outline) based on generic geomorphic templates and parameters such as island size and shape, sea level, or tectonic parameters (Chapter 3). Subsequent stages instantiate mesoscale reef structures by populating the base environment with ecologically informed semantic entities (Chapter 4). Microscale details are then added by applying procedural textures or controlled erosion simulations (Chapter 5).

## 1.2 Contributions and outlines

This thesis explores the procedural generation of underwater environments, with a particular focus on coral reef islands. Our contributions are organised into three complementary parts, covering the creation, structuring, and physical evolution of underwater terrains.

In Chapter 2 we first introduce the fundamentals of procedural terrain generation. A description of the different terrain representations is provided, as well as an overview of coral biology and coral reef formation.

In Chapter 3, we propose a user-guided method for the procedural creation of coral reef islands as illustrated in Figure 1.3. Users sketch the island shape and elevation from two projections and define a wind field that simulates long-term environmental deformation. The system models coral growth and outputs heightmaps that are further used to train a conditional Generative Adversarial Network (cGAN) for diversified generation. This method enables fast and controllable generation of varied reef island configurations.

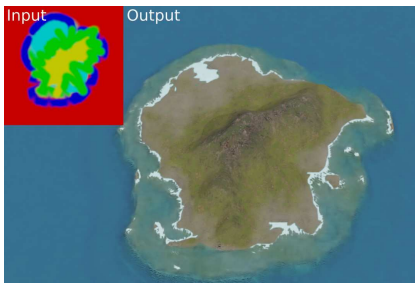



**Figure 1.3:** *Example of island generated in Chapter 3.*

The semantic representation we present in Chapter 4 aims to design features of a terrain with an abstraction of its geometry. We introduce environmental objects and their simplified representation from the real world, which we used to obtain a symbolic representation of the terrain features, biotic and abiotic, that are present in the scene (such as the canyon, rocks and corals in Figure 1.4). Using symbolism allows us to focus on the interactions between the different elements to generate a plausible ecosystem without the high computational needs of running an accurate multiphysics simulation. Moreover, the simplified representation used allows the user to manipulate the layout of the final terrain without having to choose a specific terrain representation.

**Figure 1.4:** *Example of ecosystem generated in Chapter 4.*

To increase the realism and the visual impact of the generated synthetic landscapes, the use of terrain enhancement techniques is often required. In Chapter 5, we tackle the specific challenge of running erosion simulations, a type of enhancement that mimics the effects of water, wind, and erosive forces on a virtual terrain to improve the believability of the final landscape. A particle-based erosion method is proposed, designed to be generalisable for flexibility on multiple scales and representations, with its implementation oriented towards speed and parallelisation. The main flexibility of our method is to be applicable to multiple terrain representations, agnostic to the fluid solver used, and generalised for both landscapes and seascapes. Figure 1.5 illustrates our method simulating coastal erosion, operating at the air-water interface on a voxel grid.

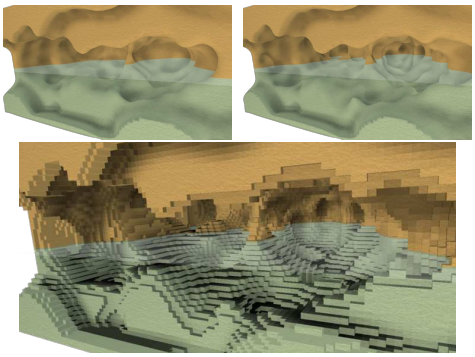


**Figure 1.5:** *Example of sea caves generated in Chapter 5.*

Chapter 2

# Background

## Contents



The purpose of this chapter is to establish the common ground required for the remainder of this thesis. Because the three contributions presented in this thesis each come with their own specialised state of the art, the role of this chapter is not to provide an exhaustive survey of algorithms, but rather to:

- introduce key concepts and terminology shared across all contributions;
- present the main terrain representations and interaction paradigms relevant to procedural modelling;
- highlight biological and ecological principles of coral reefs that directly constrain modelling choices.

Procedural generation of environments is at the intersection of realism, speed, and user control. Achieving all three simultaneously remains an open challenge, particularly in the context of coral reef islands where terrain is co-constructed by geological processes, hydrodynamics, and living organisms. Understanding how existing approaches position themselves along this speed-realism-control triangle will provide a framework for situating our own contributions.

A second cross-cutting issue is the choice of representation. From heightmaps to voxels and layered models, each structure affords specific strengths (efficiency, volumetric detail, analytic flexibility) but also imposes limitations (no overhangs, high memory cost, restricted editing). Because each contribution in this thesis relies on a different representation, a concise overview is needed upfront to

avoid duplication later.

Finally, modelling coral reef seascapes demands knowledge beyond computer graphics. The morphology and ecology of corals impose environmental constraints (light dependence, depth ranges, growth morphologies, competition with algae) that must be abstracted into procedural rules at small scales (individual corals) to large scales (complete reefs).

Together, these elements form the conceptual scaffolding for the rest of the document. The chapter first reviews the foundations of procedural terrain generation, then examines terrain representations and interaction paradigms, before introducing coral colonies and coral reef biology.

## 2.1 Procedural terrain generation

Procedural generation consists of algorithmically creating content based on predefined rules, mathematical models, or data-driven methods, rather than authoring it manually. The key advantage include reducing storage needs (since content is generated on-the-fly), enabling rapid creation of large or varied content, and supporting both reproducibility and controlled variability. These benefits are especially valuable in terrain generation where large landscapes or seascape can be created from compact parameters sets where artists, researchers, or domain experts can iterate quickly towards a desired output through as few trials and errors as possible.

The conceptual roots of procedural techniques in graphics trace back to fractal geometry (see Figure 2.1) and noise functions (see Figure 2.2). Mandelbrot's work on fractals demonstrated how complex, self-similar structures can arise from simple rules and randomness, influencing landscape modelling approaches that mimic natural roughness and multi-scale detail [Man83]. Perlin's introduction of gradient noise provided a practical mechanism to generate coherent pseudo-random patterns for textures and terrain features [Per85]. These foundational ideas underpin various terrain algorithms, where noise at multiple scales produces hills, valleys, and finer details.

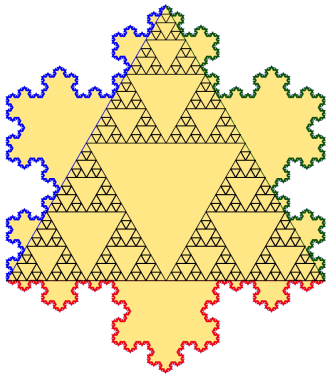


**Figure 2.1:** *Fractal geometry: Sierpinski Triangle in Koch Snowflake, from Larry Riddle.*

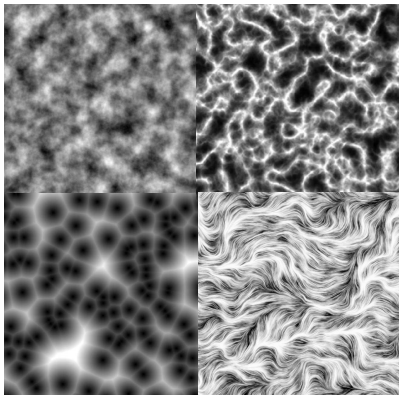


**Figure 2.2:** *Example of procedural noises.*

Procedural paradigms are often categorised as deterministic (repeatable given fixed inputs), stochastic (introducing randomness for variation), or hybrid (structure defined by deterministic rules, with randomness filling details). Rule-based systems, where content emerges via iterative application of predefined rules or grammars, play a central role in procedural generation, particularly in generating structured features (e.g., river networks, vegetation patterns). In terrain generation, deterministic components ensure global coherence (e.g., major landmass shapes), while stochastic elements add natural variability (e.g., subtle elevation perturbations).

Methods proposed in later chapters leverage these paradigms: for instance, the sketch-based island method, combined with noise and data-driven augmentation from Chapter 3 yields varied yet controlled outputs, and our ecosystem generator presented Chapter 4 is governed by rule-based systems with

stochastic sampling.

Recent advances integrate machine learning to learn distributions of terrain patches or entire landscapes from data, enabling synthesis of realistic terrain patterns beyond hand-tuned noise parameters, using especially generative models such as GANs or autoencoders. In Chapter 3 we use a conditional GAN to diversify coral reef island heightmaps. Hence, a detailed description of learning-based terrain methods appears in that chapter's State-of-the-Art.

In interactive and production pipelines, procedural systems must balance realism, speed, and user control. Fast evaluation (often parallelised and/or using GPU) supports real-time or near-interactive feedback for artist-driven editing. Parameter design and UI tools (e.g., sketching, semantic objects, masking) are critical to let users influence generation intuitively. The semantic environmental objects framework proposed in Chapter 4 exemplifies user-centric design, enabling multi-scale editing and expert knowledge integration.

Procedural terrain generation also encompasses physical simulation techniques such as erosion and sediment transport models, that refine initial shapes to enhance plausibility. These simulations are typically more expensive but can run offline or via parallel algorithms. Chapter 5 presents a particle-based erosion method applicable across representations (height fields, voxels, etc.), reflecting this integration of procedural and physical approaches.

Terrain refers to the physical features and configuration of a specific area of land. It includes the elevation, slope, and the overall topography (e.g., mountains, valleys, plains, ...). Terrain is often used to describe the surface characteristics of the land, focusing on the natural contours and the geographical aspects that define a region's physical form.

While the term "terrain" describes the physical characteristics of land, it does not include the natural elements that shape an area's identity. Elements such as vegetation, water bodies, and climatic conditions are essential our perception and understanding of landscapes. Therefore, when discussing procedural generation in virtual environments, "landscape generation" is a more fitting term, as it integrates these natural elements along with the topographical features.

In addition to "terrain generation", other terms such as "landscape generation", "world generation", and "environment generation" can be used to describe the creation of virtual landscapes. These terms are interchangeable and all refer to the process of generating physical terrain along with natural and artificial elements. However, by convention and for simplicity, the term "terrain generation" is most commonly used in the Computer Graphics field. Despite its original focus on the physical features of the land, "terrain generation" has evolved to encompass a broader range of environmental elements, making it a convenient and widely accepted term for describing the comprehensive process of creating virtual environments.

Next section presents different terrain representations used for virtual environments with their main advantages and inconvenients.

## 2.2 Terrain representations

A terrain can be represented in many different ways, each suited to particular algorithms and interaction paradigms. Figure 2.3 illustrates four main representations: height fields, layered models, density- and binary-voxel grids. Because converting from one representation to another generally incurs information loss, choosing an appropriate underlying model is crucial for the quality and controllability of procedural results.

For completeness, we note that several families do not appear in our later chapters, including irregular meshes and TINs (Triangulated Irregular Networks), and point-based splats or surfels, or neural representations [YSL05, ZZL08, AFD*21, ACG22, CLY*24, CSB*25, DGG24, LYZ*22].

This chapter therefore introduces only the models that will be used in the remainder of the thesis; the reader is referred to [GGP*19] for a comprehensive survey.

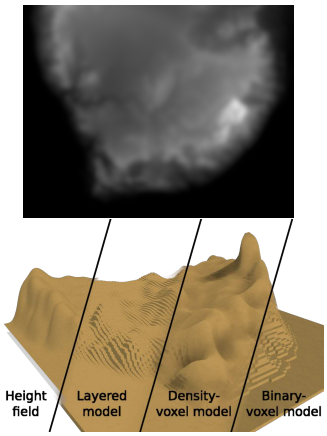



**Figure 2.3:** *An initial height map (top) can be converted into multiple terrain representations (bottom, from left to right): discrete height field, layered model, density-voxel model (rendered with Marching Cubes), binary-voxel model.*

### 2.2.1 2.5D representations (Elevation models)

Elevation models describe the terrain by a height field $h : \mathbb{R}^2 \mapsto \mathbb{R}$, assigning a single elevation value to every planimetric location. Because each $(x, y)$ has only one height, such models cannot express overhangs or caves; we therefore refer to them as "2.5D". Mathematically, height fields are analogous to images, making them amenable to decades of image-processing techniques. Their favourable memory footprint and analytic simplicity explain their preponderance in open-world games, GIS, and remote sensing.

#### Discrete height fields

Discrete height fields, often called "heightmaps", store sampled elevations on a regular grid (Figure 2.3, top). Formally, they are a function $h_\mathrm{d} : \mathbb{Z}^2 \mapsto \mathbb{R}$ defined only on integer lattice points. To render or analyse the terrain at arbitrary positions we reconstruct a continuous surface $\tilde{h}_\mathrm{d} : \mathbb{R}^2 \mapsto \mathbb{R}$ using interpolation (linear, bilinear, bicubic, ...) (Figure 2.4, right). Their raster nature aligns with GPUs and with convolutional neural architectures such as those used in Chapter 3. Discrete grids scale to large terrains with modest memory footprint, but fixes the dimensions and resolution of the domain.

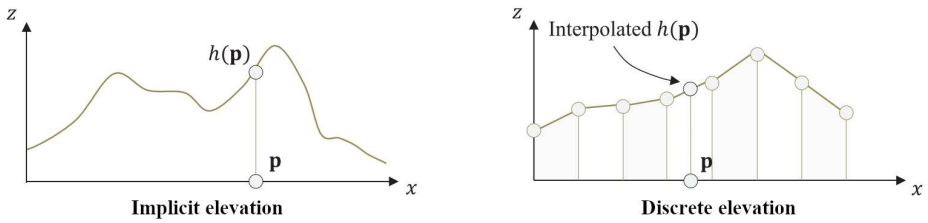



**Figure 2.4:** *Elevation models can be (left) implicit functions evaluated at any point, or (right) discrete samples augmented with interpolation to recover a continuous surface [GGP*19].*

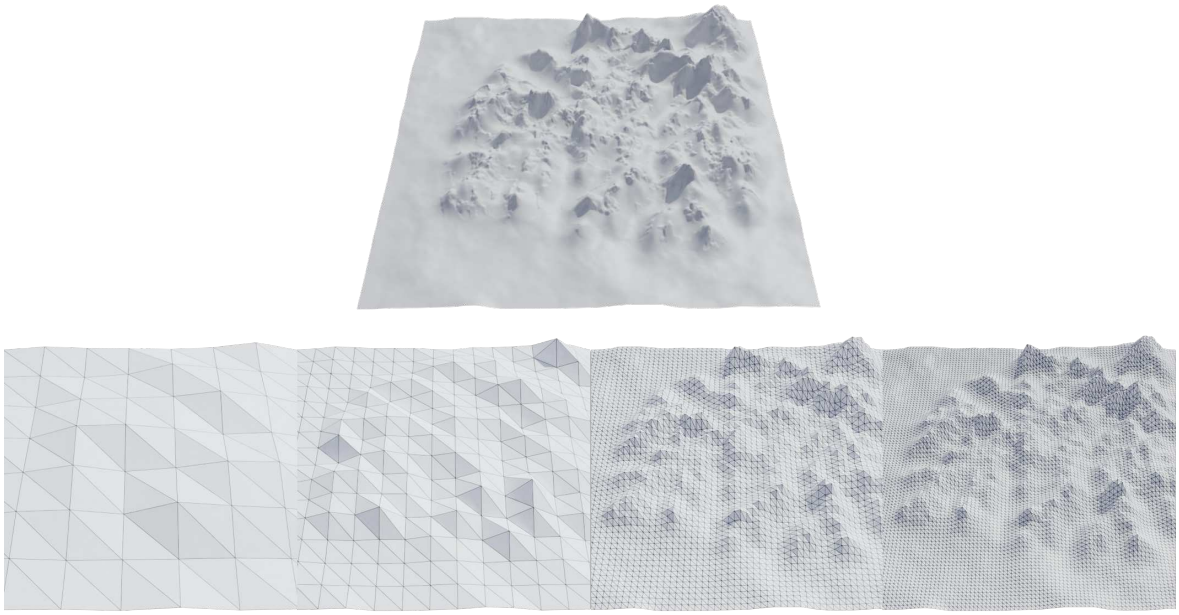


**Figure 2.5:** *Top: an implicit height field can be ray-traced at any resolution. Bottom: discrete height fields (samples of the top function) have fixed resolution: finer meshes approach the implicit surface, whereas down-sampling inevitably loses detail.*

### Implicit height fields

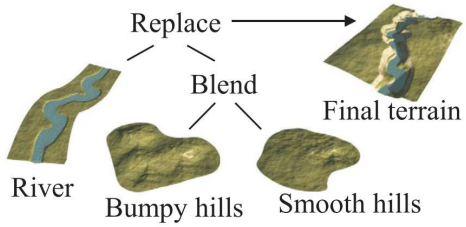



**Figure 2.6:** *Primitives composition tree, blending two hills together, then replacing a part of the surface with a river path [GGP*15].*

An implicit height field represents the terrain by a closed-form or procedural function $h_i : \mathbb{R}^2 \mapsto \mathbb{R}$ that can be evaluated at any $(x, y)$ (Figure 2.4, left). Because no samples are stored, the representation is compact and inherently multi-scale (in opposition of discrete height fields, as visible in Figure 2.5), and it supports analytic operations. These properties make it the backbone of the semantic terrain synthesis developed in Chapter 4. The drawback is runtime cost: densely evaluating a complex function over a large domain is computationally expensive, but allow "on-demand" evaluation.

Beyond the construction-tree model of [GGP*15] (Figure 2.6), several other implicit 2.5D terrain representations are commonly employed: gradient-noise fractals (e.g., Perlin, Simplex or FBM) provide fast, tileable, and stochastic detail; spectral or analytic formulas (e.g., ridge-noise, domain warps or erosion kernels) enable semantically tuned landforms; feature-curve diffusion techniques allow users to define ridges and rivers via polylines which are then diffused into the height field; vector-primitive models blend geometric shapes like ellipsoids or plates directly into the terrain surface; and radial-basis-function (RBF) implicits fit a sparse set of control points with smooth Hermite interpolants for interactive editing .

These alternatives are surveyed in depth in Chapter 3' State-of-the-Art.

## 2.2.2 3D representations (Volumetric models)

Volumetric models encode not only the terrain surface but also its inside and outside, making it possible to represent overhangs, caves, voids, and material heterogeneity that 2.5D elevation models

cannot capture. We organise them into three families: implicit fields, discrete voxel grids, and layered models.

### Implicit volumetric models

An implicit model defines a signed or unsigned scalar field $f_{\mathrm{i}} : \mathbb{R}^3 \mapsto \mathbb{R}$; the terrain surface is the isosurface $f_{\mathrm{i}}(\mathbf{x}) = 0$. Positive values typically denote solid matter, negative values air. Because the field is continuous, implicit models enable smooth blends, Constructive Solid Geometry (CSG) operations (Figure 2.7), and multi-scale detail synthesis [PGP*19]. Their main drawbacks are the cost of dense evaluation and the need to polygonise (e.g. Marching Cubes) for rasterisation [dALJ*15].

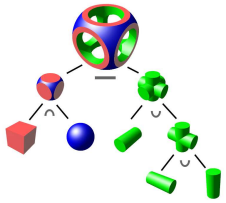


**Figure 2.7:** *CSG tree combining implicit primitives with intersection (left), unions (right) and difference (top) operations.*

Signed-Distance Fields (SDFs) store the shortest distance to the surface and are widely used for real-time deformation and global illumination. Adaptively-Sampled Distance Fields (ADF) refine the grid where detail is needed, while sparse voxel SDFs, such as OpenVDB, scale to billions of voxels [FPRJ00, Mus13]. Radial-basis implicit volumes offer smooth interpolation from scattered samples.

### Discrete voxel grid models

A voxel grid partitions space into a regular lattice $\mathbb{Z}^3$; each lattice point $v = (i, j, k)$ stores a datum describing the local state of the terrain. Because grids are topologically simple and cache-friendly, they interface well with GPU rasterisers, cellular-automata solvers, and image-processing operators extended to 3-D. The main drawback is cubic memory growth; sparse and hierarchical encodings (SVO, DAG, OpenVDB) alleviate this when the occupied volume is far below the bounding box [LK10, VMG17, Mus13].

Voxel grids support boolean set operations, distance transforms, morphological filters, and mesh extraction (Marching Cubes, Dual Contouring) in a very similar fashion as the their 2D conterparts in image processing (binary images, indexed images, and greyscale images).

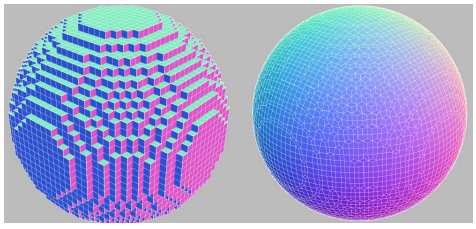


**Figure 2.8:** *Binary surface (left) vs. density-grid isosurface (right). Density grids preserve curved boundaries without refining the lattice.*

*Binary voxel grids*

Binary voxel grids record pure occupancy, defined as $o : \mathbb{Z}^3 \mapsto \{0, 1\}$. Their single-bit payload makes them the lightest form of volumetric storage, and simple bit-packing enables SIMD flood-fills or cellular automata. When combined with Sparse Voxel Octrees or voxel DAGs they scale to scenes containing $10^{12}$ voxels while keeping GPU residency practical [LK10, VMG17]. The downside is blocky surfaces: without post-processing, silhouettes follow axis-aligned voxel faces (see Figure 2.8, left). Smoothing kernels or surface nets can mitigate this at the cost of losing exact occupancy.

*Material voxel grids*

Material grids extend binary occupancy with categorical information (stone, sand, water, ...) with $\mu : \mathbb{Z}^3 \mapsto \mathcal{M}$ (with $\mathcal{M}$ a material from a finite material set). They are the volumetric analogue of indexed images and form the basis of block-based worlds such as *Minecraft* (Figure 2.9). Each update affects exactly one cell, making editing and cellular-automata simulation intuitive. The obvious limitation is abrupt boundaries: transitions remain voxel-aligned.

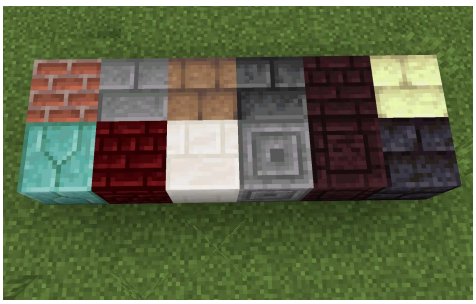


**Figure 2.9:** *Each cell of the voxel grid is encoded with a material, visible by different textures.*

*Density voxel grids*

Storing a real-valued scalar per voxel enables implicit isosurfaces inside the grid, defined as $d : \mathbb{Z}^3 \mapsto \mathbb{R}$ with $d$ typically an occupacy fraction or a signed distance. Marching Cubes or Dual Contouring produce triangle meshes with sub-voxel geometric accuracy (Figure 2.8, right), while the grid still supports topology-aware editing via direct field arithmetic. Applications range from high-fidelity VFX terrain in OpenVDB [Mus13] to destructive game mechanics and GPU-based fluid coupling. Memory cost is higher than its binary counterpart (4 or 8 bytes per voxel instead of one), but sparsity structures or narrow-band storage tame the overhead [FPRJ00].

### Layered models

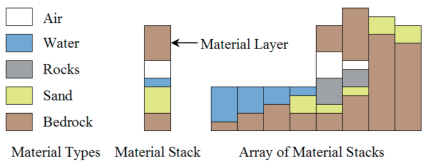



**Figure 2.10:** *Layer representation as described in [BF01]. [PGGM09] allows air and water strata to be inserted between solid material strata.*

Beneš and Forsbach introduced a layered-based terrain representation in which the landscape is discretized as a 2D grid of vertical material stacks, each composed of horizontal strata representing a single homogeneous material (e.g., water, sand, bedrock, ... presented in Figure 2.10) [BF01]. The terrain surface is given by the height of the top layer in each stack.

Peytavie et al. improved the structure in two critical ways [PGGM09]. First, they introduce an explicit air layer that may be inserted between solid strata to allow true 3D features such as caves, overhangs, and arches. Second, they couple this discrete stack data with a convolution-based implicit surface field $f(\mathbf{p})$, defined as the normalised local volume fraction of solid material $\mathcal{V}_\mathcal{M}$ within a cubic kernel $\mathcal{V}_\Omega$. The zero-level set $\mathcal{V}_\mathcal{M} = \frac{1}{2}\mathcal{V}_\Omega$ is polygonized to form a smooth, editable surface mesh (see Figure 2.11, using the empty volume $\mathcal{V}_\emptyset = \mathcal{V}_\Omega - \mathcal{V}_\mathcal{M}$).

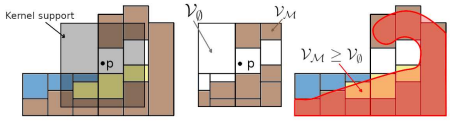



**Figure 2.11:** *The implicit volume (right, red) computed from layers-based terrain (left) is obtain by computing all points* $\mathbf{p}$ *for which a cubic kernel support (middle) contains more solid material* $\mathcal{V}_\mathcal{M}$ *than fluid material* $\mathcal{V}_\emptyset$ *in [PGGM09].*

## 2.3 User interaction for procedural terrain generation

Procedural algorithms are designed to find a balance between three main objectives: controllability, realism, and computation speed [EPW*03, STBB14, GGP*19]. In our work, we positioned our focus on the inclusion of the user in the generation process, resulting in a need to focus our attention on

controllability for authoring landscapes, and computation speed in order to quickly visualise results, allowing to efficiently interact with the system to reach the user goals. In this section, we briefly present the tools available for interacting with a procedural system.

Procedural terrain generation algorithms typically rely on user-defined parameters to produce results that meet specific needs. To effectively involve users in the creation process, it is essential to provide tools that allow intuitive interaction with these parameters. These parameters vary widely in nature, and each type benefits from different kinds of interaction tools. While prior surveys have discussed categories of procedural methods [SKG*09, STBB14, GGP*19], we propose to broadly classify interactive tools for procedural terrain generation along a global-to-local spectrum:

- **Global procedural parameters:** Values that reshape the whole terrain field at once.

This category includes *noise parameters* (e.g., frequency, amplitude, octaves) that drive the terrain base-shape variation [Per85, FFC82], and *physical constants* (e.g., erosion sediment capacity, wind strength) used by simulation passes [BF01, PPG*19].
These numerical values are usually exposed through spinboxes, sliders, or configuration files.

- **Regional controls (masks and density maps):** Binary or weighted maps that localise subsequent generators, indicating where and with what intensity to apply a process [dCB09, SS05].

Users paint or import these masks with brush tools, which are often using weighted values to avoid the hard edges of binary masks.

- **Assets and features placement:** Interactive positioning (or scattering) of discrete assets such as rock arches, buildings, or river splines [BKRE17, GPG*16].

Users click to set an initial $xyz$ location, then translate, rotate, or duplicate as needed.

- **Surface sculpting (manual and procedural brushes):** Point-level edits that raise, lower, smooth, or terrace the height field [GH91, GM05, CHCC21].

Users control the brush effect through parameters like radius, intensity, and falloff shape.

Each developer uses specific strategies for each type of parameters as the needs and objectives of each application are different. Providing the user with too many parameters can be overwhelming, while too few limits the output possibilities [TCL*13]. In the meantime, providing unintuitive parameters such as dimensionless values often results in a trial-and-error strategy, requiring the generation algorithm to be run many times before finding the appropriate values. This implies that the algorithms must be able to be executed fast. On the opposite side, removing user controls to use hard-coded constants instead increases processing speed. Finally, an algorithm that aims to be fast or let the user control numerous parameters may reduce the realism of the output. Most algorithms try to strike a balance between realism, speed, and control.

Realism refers to the extent to which generated content accurately represents real-world characteristics, such as visual details, physical processes, and natural patterns. This is especially important in simulations, visualisations, and training environments where physical accuracy is critical. Techniques to enhance realism include physical simulations, which model erosion processes and the integration of expert knowledge from geological and ecological fields. However, achieving realism is challenging due to the complexity of detailed simulations and the need for specialised expertise, which may demand substantial computational resources and inputs. Usually, a realistic algorithm tends to have low user control and speed.

On the other hand, speed is about the efficiency of content generation within acceptable time constraints. While no official categorisation has been set, we propose to describe levels of speed based on response time for generation:

- **Real-time:** less than 30 milliseconds, essential for interactive applications such as VR environments,
- **Interactive:** under 3 seconds, suitable for user-driven customisation in games and simulations,
- **Near-interactive:** less than 5 minutes, applicable for larger-scale simulations where some delay is acceptable,

- **Offline:** greater than 5 minutes, often used for precomputed content or offline rendering.

The execution speed of a procedural algorithm is crucial as the user fine-tunes the parameters before being satisfied [STBB14]. Optimising speed involves using efficient algorithms and parallel processing. Almost all recent works achieve fast execution times thanks to high parallelisation on GPU [Ols04]. Other types of algorithm may rely on a refinement paradigm, generating a coarse result at first and then iteratively adding finer details, such that the global output shape is available long before the real final result.

## 2.4 Coral biology

Coral reefs, once misunderstood as sea plants or lifeless rocks, are now known to be biologically rich structures formed by colonies of marine invertebrates. Technological advances in the 20th century, including scuba diving and sonar mapping, enabled direct observation of reefs, revealing their complex ecological and geological roles. Today, tools such as remote sensing and genetic analysis continue to enhance our understanding of their biodiversity and spatial patterns.

This section summarises key biological and ecological features of corals, focusing on elements relevant to reef morphology and terrain modelling.

### 2.4.1 Coral polyps and colonies

Coral polyps, illustrated in Figure 2.12, are small soft-bodied marine invertebrates that secrete calcium carbonate ($CaCO_3$) to build an external exoskeleton. Over time, successive secretion by many polyps produces the rigid framework of a reef. Symbiosis with zooxanthellae (photosynthetic algae) supplies most of the energy for calcification; this makes coral growth strongly light-dependent and tied to shallow, clear-water zones.

From a terrain generation perspective, the $CaCO_3$ secretion underlies the elevation of reef structures and the light-dependence of the polyps implies depth-based constraints on where reef heights appear, relative to water level. Moreover, coral-built substrate produced by accumulated skeleton, or "dead coral", increases landscape resistance to erosion.

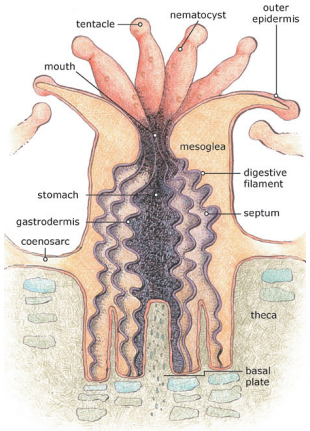



**Figure 2.12:** *Anatomy of a single polyp, ranging in size from one to three millimeters in diameter.*

Coral colonies adopt varied growth forms adapted to local conditions (light, water motion, depth). Classifying coral species has been done using different criteria [KLR*20]: by coverage [Kuc86, BBM*09, PGP*19], by dominant taxa [Sto69], or by coral growth morphology [MH99, GSW06]. We use the classification of hard corals from [GSW06] to describe five different coral morphologies and their typical properties in reefs: massive, branching, foliose, table and encrustring, which we detail in the following.

- **Massive:** These corals form large boulder-like shapes reaching several metres in diameter under calm conditions (Figure 2.13). Found on reef slopes and back-reefs in low to moderate energy, up to about 30 m depth, tolerating moderate to low light, their slow growth persists over centuries in stable conditions [CLR*20]. They are foundational reef-builders, providing long-term structural stability and substrate for other organisms to settle, while hosting diverse fishes in their crevices. Their bulk buffers wave energy, alters flow to form sediment-depositing lee zones, stabilises the substrate, and produces oxygen over large surfaces [BKS*14].

**Figure 2.13:** *Large Porite lobata - Photo credit: Luis Lamar.*

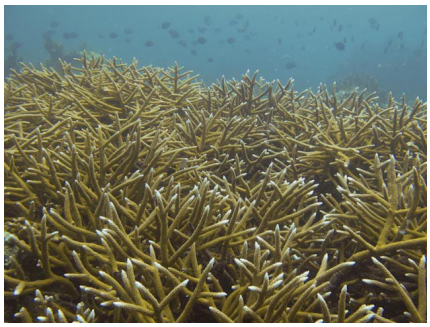


**Figure 2.14:** *A field of staghorn coral - Photo credit: NOAA.*

- **Branching:** These corals exhibit tree-like branches with lengths often exceeding 1-2 m (Figure 2.14). They occupy shallow reef surfaces, up to 20 m deep in high-light, high-energy environments. Fast-growing, they rapidly deposit carbonate, create 3D nurseries for juvenile fish, and support biodiversity. Their high rugosity increases turbulence and drag, altering currents, trapping sediment, and producing oxygen [BKS*14, OCTR25]. Fragile branches break in storms, aiding dispersal via fragmentation [MANS13]

- **Foliose:** These corals form leaf-like laminae often spanning over a metre wide (Figure 2.15). They inhabit back-reefs and reef flats between 2 to 20 m depth with moderate flow and light. They provide microhabitats, surface for symbionts, trap sediment, and contribute moderate carbonate.

**Figure 2.15:** *Pavona cactus - Photo credit: Benzoni, F.*

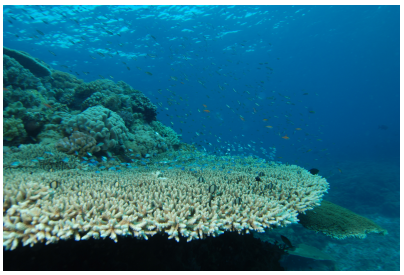


**Figure 2.16:** *Example of an Acropora pulchra, forming a large table - Photo credit: Albert Kok.*

- **Table/Plate:** These corals develop broad plates up to 3 m wide in shallow, well-lit reef crest and upper fore-reef zones above 15 m deep (Figure 2.16). They maximise photosynthesis and carbonate production, offer horizontal habitats and shaded refuges, and support encrusting life beneath which alters water flow by causing divergence above and shadows below, redistributing sediment, and producing oxygen. Fragile in storms, they form rubble that aids colonisation when fragmented.

- **Encrusting:** These corals grow as thin mats tightly adhering to substrate for about a metre wide and up to 40 m deep, tolerating various flow regimes. By resisting breakage and binding rubbles, they stabilise the substrate and cohesion, reducing erosion. *Psammocora*, displayed in Figure 2.17, spans over a part of massive coral and the soil surface, protecting the covered coral from erosion and stabilising the colony, but blocking the sunlight from reaching it.

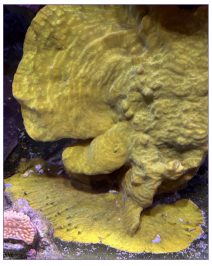


**Figure 2.17:** *Psammocora - Photo credit: Michael Paletta.*

### 2.4.2 Ecological interactions

Coral dispersal and colonisation occurs at multiple scales through budding (local polyp duplication), brooding (larvae settle near parent), polyp bail-out (stress-induced short-range resettlement), fragmentation (medium-range reattachment after disturbance), and broadcast spawning (long-range larval drift).


**Figure 2.18:** *Coral bleaching in the National Marine Sanctuary of American Samoa between 2014 and 2015 - Photo credit: NOAA/ XL Catlin Seaview Survey.*

This diffusion is balanced with their mortality and environmental factors limiting their spread. The most notable factors taken into account are elevation of temperature and $CO_2$ concentration, visible through the increase in their bleaching frequency and reduction of resilience, as well as pollution, which favours algal blooms, overcoming and shading polyps [MJD01, CCK*21]. Another key factor is sea-level changes, in response to which reefs' vertical accretion may "keep up" (the coral reef stays in the sunlight, just below water), "catch up" (growth slower but eventually matches sea level), or "give up" (cannot match water level and cannot receive enough sunlight) [KGC*11, RQT*13]. Figure 2.18 displays a field of staghorn coral in healthy shape, during bleaching, and finally dead, in which case the chances for the regrowth of the reef are almost null.

Coral and algae both compete for sunlight, substrate, and nutrients. However, algae grow faster, especially under elevated nutrients or reduced herbivory, leading to algal overgrowth that smothers coral, flattens reef topography, and reduces habitat complexity and biodiversity. An overwhelmed massive coral by algae, blocking access to light, is shown in Figure 2.19. On a small-scale level, other organisms such as parrotfish and certain invertebrates consume algae and break down dead coral, producing sediment that can fill gaps and support new settlement for polyps [MYG*14].

**Figure 2.19:** *Algae colonising massive coral.*

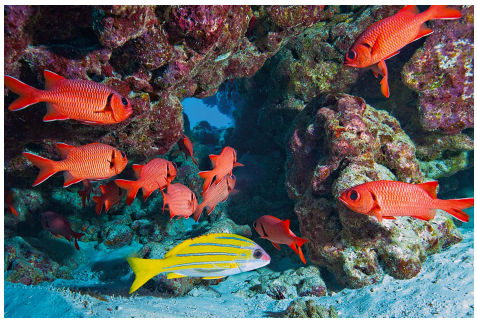


**Figure 2.20:** *Coral reefs are porous, resulting in biodiversity shelter, but also complex hydrodynamics.*

Structural complexity (rugosity, crevices) offers shelter for juveniles, spawning grounds, and supports diverse species (about 25% of marine species in $< 0.1\%$ ocean area) [OCTR25]. The true internal geometry of a reef is not currently completely understood but is known to influence the hydrodynamics, and by extension, biodiversity, of the whole biome (Figure 2.20 shows a small shelter for biodiversity of a few centimetres cubed from a few kilometres wide reef).

These fine-scale properties have influence on the large-scale environment as they introduce turbulences and drag in the water flow [SBD*20], and produce sediment deposition at small and large distances [RQT*13, Kre27].

For the same reasons, reef crests break incoming waves; morphological complexity (e.g., branching structures) slows water and reduces energy reaching shore, mitigating erosion [Woo03].

The erosion process is directly impacted by the small scale. Reef structures trap or stabilise sediments,

protecting mangroves and seagrass beds. Simulation of the evolution of the landscape and fluid dynamics would require taking into account, almost as importantly, the large-scale and small-scale bathymetry, as well as the geological and biological aspects of the ecosystem, to reach accuracy [CDP*17, CDRB15].

Overall, from these behaviours, we consider that corals act as important dynamic ecological agents, engaging in feedback loops with their environment. Modelling them requires accounting for spatial diffusion via diverse reproductive modes, competition for resources, and bidirectional environmental influences. We describe ecological agent simulation in Chapter 4's state of the art section.

### 2.4.3 Coral reef formations shape

In his travel journal, Darwin outlines his atoll reef formation theory, proposing that Earth's crust movements under the oceans created atolls [DJ42]. Darwin described a three-stage process in atoll formation: a fringing reef forms around a sinking volcanic island, evolving into a barrier reef, and finally an atoll reef as subsidence continues [Hop14]. In this thesis, the term "coral reef island" refers to an island of volcanic or continental origin that is surrounded by coral reefs, rather than a low island formed from reef-derived sediments (as used by [MBK20])

#### Fringing reefs

Fringing reefs are the most common and widespread coral reefs. They form narrow bands near the shore and are usually attached directly to land. Sometimes, they are separated by a shallow, narrow lagoon [KLR*20]. Their coastal location makes them accessible but also exposes them to land- and sea-based stressors.

Their structure includes the reef flat, reef crest, and reef slope. The reef flat is shallow, calm, and closest to shore, as seen by the clear blue region in Figure 2.21. The reef crest is higher, exposed at low tide, and faces strong wave action. Only hardy corals survive there. The reef slope descends into deeper water where coral diversity and density increase. It offers habitat for many marine organisms due to more light and water movement.

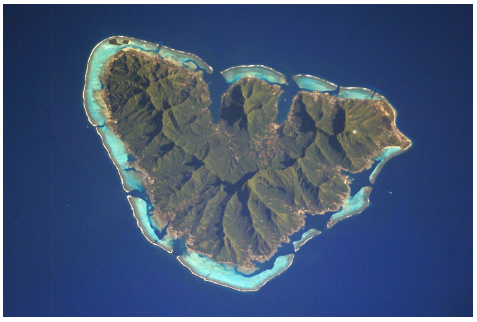


**Figure 2.21:** *Fringing reef around Mo'orea island.*

Fringing reefs grow in clear, shallow waters with high light, warmth, and moderate waves. Coastal currents bring nutrients, helping coral growth. Yet, their location makes them prone to sedimentation, runoff, and pollution [NWB*08]. Sediment can smother corals and block sunlight. Runoff adds nutrients, causing algal blooms that compete with corals (possibly causing some of the "holes" in the reef in Figure 2.21).

Zonation defines coral distribution across the reef. On the flat, corals handle temperature shifts and sediment. On the crest, species tolerate waves and low-tide exposure. The slope supports more diverse corals, often forming large structures that shelter marine life.

**Figure 2.22:** *Aerial image of Providencia Island, surrounded by a large barrier reef.*

### Barrier reefs

These reefs typically develop from fringing reefs that have grown over long periods, during which the coastline has either subsided or sea levels have risen. As the land subsides or the sea level rises, the original fringing reef is gradually separated from the shore, forming a lagoon between the reef and the coastline (the lagoon in Figure 2.22 is visible as a deep blue band between the clear blue back reef and the abovewater island).
The depth and size of the lagoon, as well as the height of the reef crest, are influenced by multiple factors such as wave action, currents, and the rate of coral growth. The lagoon often contains patch reefs (clear blue spots in the lagoon of Figure 2.22), seagrass beds, and sandy areas. The zonation within barrier reefs creates distinct ecological niches, with different species of corals, fish, and invertebrates adapted to the specific conditions found in each zone. The fore reef slope, with its high-energy environment, supports species that are resilient to strong waves and currents. In contrast, the more sheltered back reef and lagoon provide habitats for species that prefer calmer waters and stable conditions.

Barrier reefs also play a role in the connectivity between marine ecosystems. They act as a bridge between the open ocean and the coastal environments, facilitating the movement of marine species across different habitats. Many species of fish and invertebrates migrate between the reef and the lagoon during different life stages, using the barrier reef as a nursery or feeding ground.

### Atolls

The formation of atolls is a process that unfolds over millions of years. As the island gradually subsides, mainly due to tectonic activity, the coral continues to grow upward toward the sunlight. Eventually, the island disappears entirely beneath the ocean's surface, leaving behind a ring-shaped coral reef that encircles the central lagoon in a circular or almost circular shape (Figure 2.23 displays a deformed atoll with the presence of cay islands on its rim). This theory of atoll formation was first proposed by Charles Darwin and remains one of the most widely accepted explanations for the development of atolls.

The lagoon varies greatly in depth, ranging from shallow areas with sandy bottoms to deeper sections where patch reefs and seagrass beds may develop [Gol16]. The sheltered waters of the lagoon provide a calmer environment compared to the outer reef slope, allowing a different community of marine species to develop. These species often include a mix of corals, fish, and invertebrates that are weakly tolerant of the high-energy conditions. The lagoon may also contain small coral reef islands, known as motus, which form from the accumulation of coral debris and sand.

**Figure 2.23:** *Aerial view of Tetiaroa Atoll, French Polynesia.*

---

Understanding the biological mechanisms that shape coral reef morphologies underpins procedural

methods for generating and simulating reef island environments across multiple scales. At large scales, 2.5D elevation models for representing coral reefs suffice because cavities and overhangs can be neglected; at smaller scales, 3D representations are essential to capture erosive processes that evolve in all three dimensions.

Crucially, our aim is not only realism but controllable authoring. These environments are defined by surface geometry, material heterogeneity, and living components, so we combine intuitive controls with models that remain responsive for iterative design. In the speed-realism-control trade-off, we privilege representations and algorithms that preserve key biological and physical constraints while enabling interactive, multi-scale editing.

Because growth, sedimentation, erosion, and bioerosion act at distinct spatial and temporal scales, the computational representation must match the phenomenon and the intended authoring operations. This perspective motivates the multi-scale, multi-representation framework developed in the following chapters: automatic, user-guided generation of large-scale coral reef islands (Chapter 3), authorable ecosystem simulation with biologically informed rules (Chapter 4), and erosion modelling across terrain representations (Chapter 5).

Chapter 3

# Automatic generation of coral reef islands

**Figure 3.1:** *Given a free hand-drawn sketch of the different regions of an island (bottom inset of each example), our method generates a corresponding height field using neural networks. Subsidence (gradual sinking of land) is controlled by the user in the luminosity channel of the input. These examples show the gradual subsidence of the same island over time.*

## Abstract

In this chapter, we propose a procedural method for generating single volcanic islands with coral reefs based on user sketches from two projections commonly used in geological and remote sensing domains: a top view defining the island's shape, and a profile view, establishing its elevation. We add a user-defined wind field that deforms the island to mimic long-term effect of wind and wave, providing finer user control over the island's shape. Next, we model the coral growth on the island following biological observations. This results in a terrain height field of the island and its surrounding reef. Our method creates a large variety of coral reef island models which we used to train a conditional Generative Adversarial Network (cGAN). By applying data augmentation, the cGAN enhances the variety in the generated islands, providing users with greater freedom and intuitive controls over the shape and structure of the final output.

# Contents



## 3.1 Introduction

**Figure 3.2:** *Kauai Island, Tuvuca Island, Mascarene Island, and Cocos Island (from left to right) are islands formed through an identical geological process; however, their shapes vary greatly, from a full island with fringing coral reef (leftmost) to a complete atoll (rightmost).*

The scarcity of high-resolution data, the need for volumetric and multi-scale representations, and the strong influence of biological and hydrodynamic processes present unique modelling challenges for underwater landscapes. Among these environments, coral reef islands are of particular interest and complexity. They result from a long-term interplay of volcanic activity, coral growth, erosion, and subsidence, producing highly distinctive morphologies that procedural techniques must capture if they are to generate convincing seascapes. We propose a simplified user-centered interface to focus on the overall large-scale modelling of coral reef islands using sketching methods to avoid tedious user control over a large range of ecological parameters.

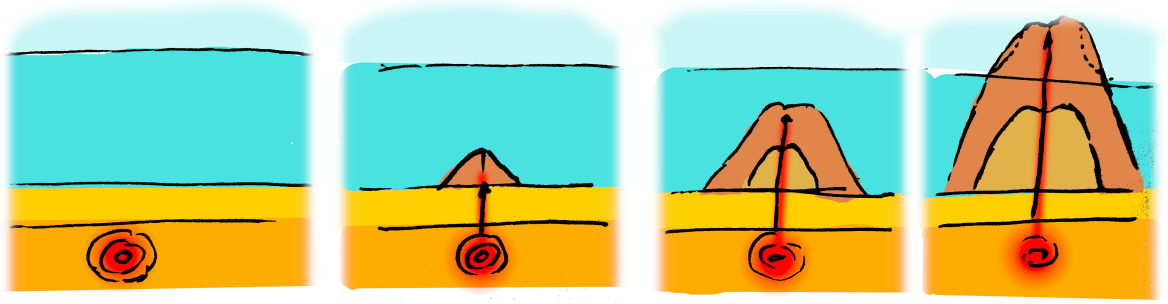


**Figure 3.3:** *Volcanic islands rise above hotspots. The rise of magma from the hotspot forms a seamount. Consecutive eruptions from the volcano grow the seamount until the peak emerges above sea level. The ground above and close to sea level is affected by hydraulic, aeolian, and coastal erosion.*

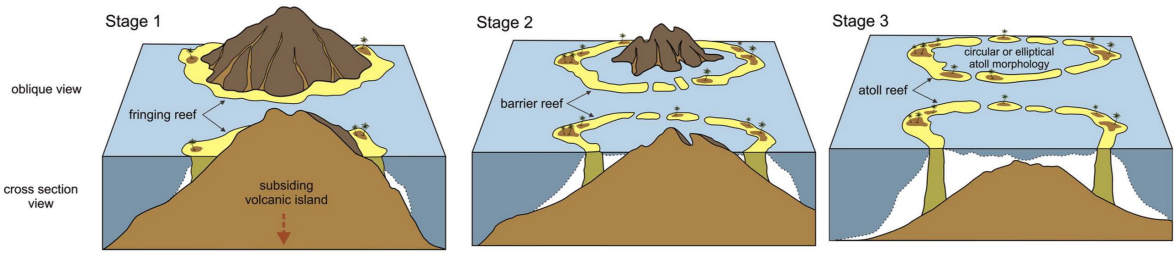



**Figure 3.4:** *Coral colonies grow near sea level, forming a fringing reef (Stage 1). The island sinks slowly (subsidence); the coral reef continues its growth to keep living coral colonies in the photic zone. The sinking island increases the size of the lagoon, forming a barrier reef island (Stage 2). When the peak of the island is below the surface of the lagoon, an atoll is formed (Stage 3) [TG13].*

To better understand these morphologies, we begin by recalling the classical geological explanation of their formation, as proposed by Charles Darwin in the 19th century [DJ42]. His subsidence theory remains a foundational description of how volcanic islands gradually transform into barrier reefs and atolls (such as the islands presented in Figure 3.2), and it provides the conceptual backdrop for our generative model.

According to this theory, these islands begin as volcanic landmasses, created when magma from the Earth's mantle erupts through the ocean floor and builds up layers of volcanic rock, eventually rising above sea level (Figure 3.3). In tropical waters, these volcanic islands create ideal conditions for coral reefs to develop. Corals, thriving in the shallow, sunlit waters around the island, initially form fringing reefs attached to the island's coastline (Stage 1 in Figure 3.4).

As time passes, the volcanic island undergoes subsidence, a slow sinking process caused by the cooling and contraction of the Earth's crust beneath the island. In response, corals continue to grow upwards to remain within the photic zone, where sunlight supports their survival. This upward growth combined with subsidence leads to the formation of barrier reefs, that become separated from the island by a lagoon as it sinks further (Stage 2 in Figure 3.4).

Eventually, the volcanic island may submerge completely, leaving only the coral structure visible above water. This process results in an atoll: a ring-shaped reef encircling a central lagoon (Stage 3 in Figure 3.4). Over geological time, the island's shape evolves from a prominent volcanic peak into a coral-dominated reef system shaped by subsidence, coral growth, and erosion.

Simulating the formation of coral reef islands presents significant challenges due to the complex interplay of geological, environmental, and biological factors [Hop14]. One major difficulty lies in capturing the long-term subsidence of volcanic islands, occuring over millions of years, and the

concurrent upward growth of coral reefs that rely on environmental conditions (e.g., water depth, temperature, sunlight, ...). This combination of slow geological processes and dynamic biological growth is inherently challenging to replicate in a computational model.

Additionally, the biological aspects of coral growth are inherently dependant on environmental factors. Coral reefs grow only within a specific range of water depth and sunlight, and their growth patterns are influenced by reef ecosystem's health and resources availability. Accurately modelling these biological dependencies in a procedural system is challenging, as these factors are numerous and difficult to generalise. Moreover, the scarcity of data available obstructs global understanding of these biomes. In a recent high-resolution mapping of shallow coral reefs [LMK*24], researchers estimated the total surface area of this biome to cover less than 0.7% of Earth's area, and more specifically that coral habitat represents less than 0.2%.

Existing terrain generation methods, such as Perlin noise-based algorithms or uplift-erosion models, are often ill-suited for these multi-disciplinary processes. Although they generate natural-looking landscapes (such as alpine landscapes, representing about a quarter of land area [KSBZ14]), they do not account for the unique geological and biological interactions critical for coral reef island development, resulting in a lack of coherence. Capturing these dynamics, while also providing user control during the modelling of a terrain, requires a balance between realism and procedural flexibility, allowing for both accurate computationally expensive simulation of natural processes and intuitive user control in interactive time.

The formation of these islands involves processes at multiple scales, from the growth patterns of coral colonies to large-scale sediment transport, which are difficult to simulate directly. As a result, purely procedural or physics-based simulations fail to produce convincing or diverse coral reef island landscapes. On the other hand, deep learning methods are inoperable due to the extremely small amount of data, and the scarcity of high-resolution DEMs of these regions.

Despite advances in terrain generation, current methods struggle to support user-controlled design of specific island shapes and achieving realism without real data. Coral reef islands exemplify this gap: we lack datasets to directly train deep learning models, and purely procedural methods require expert tuning to mimic their features.

To address these challenges, we use procedural generation as an initial step in our global pipeline. In this step we employ algorithmic rules to synthesise terrain features, allowing us to encode basic patterns of coral reef island formation. However, such algorithms are inherently rigid: they are tailored to a narrow family of shapes (in our case, radially organised and centred islands) and often rely on mathematical controls that are unintuitive for non-expert users. In our work, we use a procedural model not as the final solution, but as a means to efficiently create a large and diverse set of training examples for a learning-based model. Specifically, by adjusting procedural parameters, the procedural pipeline produces varied island scenarios. Each synthetic example is represented by a detailed terrain height field and a corresponding semantic label map that marks different regions, providing structured input-output pairs for the learning stage.

We then train and deploy a conditional Generative Adversarial Network (cGAN) as the core of our method. A cGAN is a type of deep learning model that learns to generate realistic data based on an input condition or context. In our case, the cGAN takes as input the semantic label map of an island (a label layout indicating regions such as ocean, reef, beach, and mountain) generated by the procedural step and learns to produce a plausible island height field that matches this layout. By training on a large set of examples from the procedural generator, the cGAN captures subtle terrain features and variations unique to coral reef islands, going beyond what hard-coded procedural rules can achieve thanks to the application of data augmentation.

Once the cGAN is trained on a sufficiently large and varied set of synthetic islands, it is used on its own to generate new island terrains. At this stage, our procedural generation module, only necessary to provide training data, is not required for creating new islands. Instead, a user provides a semantic map using digital drawing or another simple algorithm, and the cGAN generates a plausible island

terrain accordingly. To summarise, our pipeline leverages procedural modelling to create a training dataset, and then relies on the learned cGAN model for the final generation of coral reef islands. In this way, we overcome the scarcity of real data by procedurally generating training examples, and overcome the rigidity of procedural rules through cGAN-based learning.

The key contributions of this chapter are:

- a novel sketch-based procedural algorithm for shaping island terrains from top and profile views,
- the use of a deep learning model trained on synthetic data derived from procedural rules, acting as an abstraction layer that hides underlying complexity,
- a demonstration that the cGAN approach tolerates imprecise, low-detail user input sketches, broadening usability, without the need for cutting-edge network architectures,
- and an insight that procedural generation remains essential to produce training data in data-sparse domains, illustrated by the example of coral reef islands.

These contributions collectively show a pathway for blending user-driven design with learning-based generation for terrain modelling.

## 3.2 State of the art

Procedural terrain generation and sketch-based modelling have each seen significant advances over the past decades, yet neither alone fully addresses the particular challenges of coral reef island synthesis. On the one hand, classic noise-based and physically driven methods (Perlin noise, hydraulic erosion, uplift-erosion models) excel at producing broad, natural-looking landscapes but lack the biologically inspired reef geometries and dynamic island evolution governed by subsidence and coral growth. On the other hand, sketch-based tools give users intuitive control over silhouettes and terrain profiles, but typically require expert parameter tuning to achieve realism and do not model long-term geological or ecological processes. More recently, deep learning, especially GANs and cGANs, has emerged as a powerful way to learn terrain features and geological rules from data, yet it relies on large, labelled datasets that are scarce for coral reef islands.

In this section, we first review the key geological theories that explain reef formation, then examine traditional procedural terrain generators, followed by an overview of sketch-based terrain editing frameworks, and finally discuss recent advances in GAN-based terrain synthesis.

### 3.2.1 Coral reef island formation theories

Since Darwin first proposed his subsidence theory, presented in Section 3.1, geologists and biologists have debated how exactly coral reefs have formed around land masses. In this section we present three major alternatives to Darwin's theory, before explaining why Darwin's unified subsidence model provides the most direct foundation for our procedural generation.

- **Murray's stand-still theory [Mur80]**

Murray argued that reefs could develop on stable, non-sinking platforms, with coral growth keeping pace with modest sea-level changes rather than underlying land subsidence. He proposed that as long as water depth remained within the photic zone, reefs would accrete upward solely in response to environmental sea-level fluctuations, and continuously seaward, forming a gradual structure from a fringing reef to barrier reef (Figure 3.5). Atolls in this theory are mainly created by the degradation of the middle island through aeolian and coastal erosion until it becomes completely submeged. Murray's emphasis on environmental stability and gradual sea-level change was supported by early observations of island terraces and reef growth patterns. However, later drilling and seismic data revealed volcanic foundations buried beneath reef limestones on many atolls, which are evidence of true subsidence that Murray's theory cannot explain.

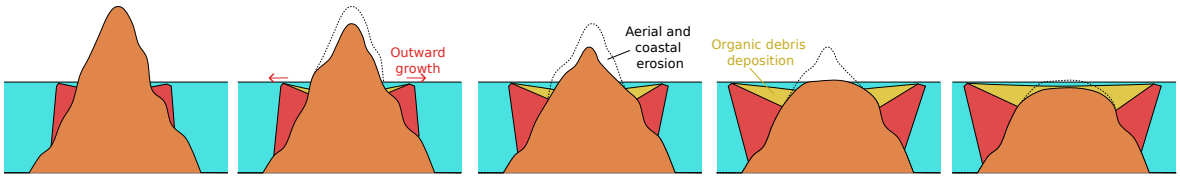



**Figure 3.5:** *In Murray's theory, with the island fixed in place but reducing in size due to erosion, reefs (red) simply remains at sea level and grows outwards (producing fringing reefs, then barrier reefs, and eventually atolls) without any subsidence, leaving only the inside reef filled with sand (yellow) forming a lagoon.*

- **Daly's glacial-control theory [Dal15]**

Daly shifts the focus from steady subsidence to global sea-level oscillations driven by glacial-interglacial cycles. He argues that reefs thrive during high sea levels in interglacials but are exposed and eroded during glacial lowstands. As sea levels rise again, the wave-cut benches left behind provide ideal habitats for new coral colonies, which grow upward with the water level and can develop into barrier reefs or atolls once sea level overtops the island (Figure 3.6). He supports this model by pointing to multiple reef terraces at different elevations and well-documented 100m sea-level drops during ice ages as support for this cycle-driven reef development. The plausibility of Daly's model lies in its direct link with climatic data and the clear geomorphic signatures of past sea-level stands. Yet, core samples often show uninterrupted reef accretion atop subsiding volcanic bases, indicating that glacial cycles alone cannot account for continuous reef "keep-up" growth.

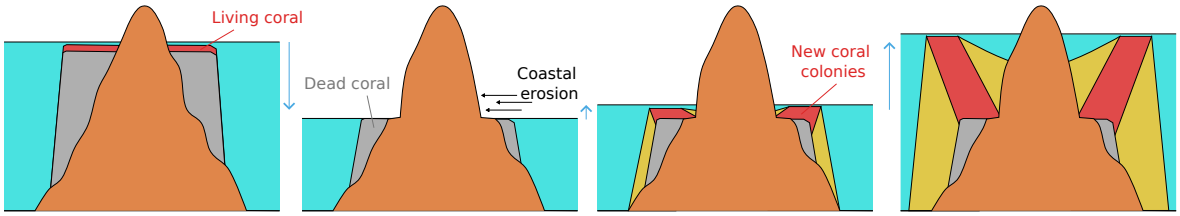



**Figure 3.6:** *In Daly's theory, repeated sea-level drops expose the reef to wave planation and erosion into a flat bench (grey), and subsequent high stands see new coral rims (red) accrete on that outer terrace to form a lagoon-separated barrier reef.*

- **Droxler's karstification theory [DJ21]**

In contrast to models based on volcanic subsidence, Droxler et Jorry attribute atoll formation to repeated karst dissolution of a broad carbonate platform during low-sea-level glacials, followed by renewed coral accretion in interglacials (Figure 3.7). High-resolution bathymetry and drill cores reveal karst-etched depressions beneath certain atoll crests, supporting this cyclic exposure-dissolution mechanism. While compelling for extensive carbonate shelves, this theory hinges on pre-existing flat platforms and meteoric water circulation (water originating from precipitation), which are conditions uncommon on volcanic seamounts. As a result, Droxler's model explains atoll rings on continental carbonate foundations but does not readily apply to volcanic island reef systems.

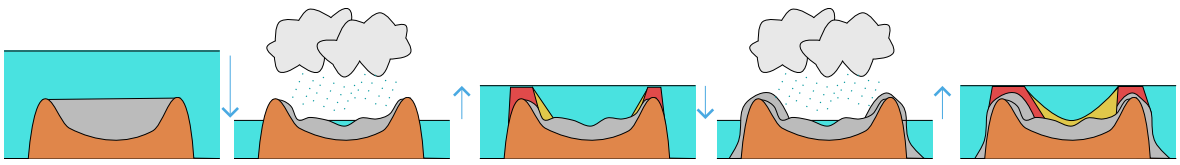


**Figure 3.7:** *A broad carbonate platform (grey) is exposed and karstified during low-sea-level glacials, then drowns and is rimmed by reef growth (red) at its outer edge during high stands, yielding an atoll-style rings.*

Of all the competing hypotheses, Charles Darwin's subsidence model offers several compelling advantages for our procedural island generation due to its simplicity, historical significance, and effectiveness [TMNM97]. This theory provides a straightforward framework outlining a clear progression from fringing reefs to barrier reefs to atolls as a volcanic island subsides, which can be easily modelled, making it practical for generating plausible island landscapes. This simplicity not only ensures predictability in simulation outcomes but also reduces computational demands, which is particularly beneficial for interactive edition applications. Additionally, Darwin's model provides a foundational basis upon which more complex phenomena (sea-level changes and climatic effects), considered as secondary factors in the geological formation of islands, could be layered. This allows us to start with a basic model and, in the future, add complexity as needed, offering flexibility in simulation design.

In our approach, we translate the core principles of Darwin's theory into a procedural generation model, emulating the gradual sinking of volcanic islands while coral reefs grow to keep pace with changing sea levels. This allows us to realistically model the progressive transformation of islands from volcanic landmasses to coral-dominated atolls. By capturing this interplay, we procedurally generate a wide variety of island structures that reflect real-world geological processes.

In generating synthetic coral reef islands, we adopt a set of simplifying assumptions that are interpreted by the formation process from Darwin's theory and field observations:

- **Radial symmetry and radial arrangement of features**

While volcanic islands do not have a perfect circular shape and reefs are not exact rings, they grow outward in roughly concentric belts (looking at the island in Figure 3.8, we see the typical layout mountain-beach-lagoon-reef-ocean). Anchoring our sketch-to-terrain engine on this radially organised structure offers two main benefits to the user: a single intuitive control over the overall island shape and while keeping distance computations simple and efficient.

**Figure 3.8:** *Aerial view of Tuvuca Island, Lau Archipelago, Fiji.*

- **Uniform profile shape**

Although real islands have ridges, valleys, local bumps and many other features, we simplify the model to a single elevation curve from centre to edge, making the profile-view sketch a one-dimensional drawing, enabling users to modify elevation with a single stroke.

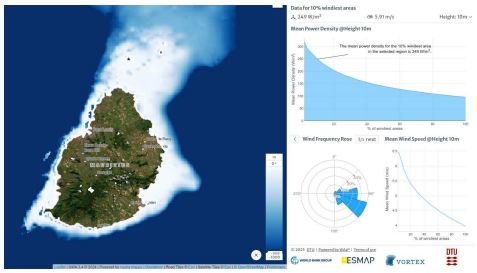


**Figure 3.9:** *Mauritius is deformed. We see a deep slope on the west side while the east side has a gentle slope. The wind rose, showing in which direction the mean wind velocity is pointing, is correlated with this deformation.*

- **Subsidence and coral "keep up" growth**

Actual reefs respond to light, nutrients, and biology, but the dominant effect is that corals grow vertically to stay near the photic zone. We decouple coral growth from subsidence and simply keep reef heights in a fixed band beneath the surface, capturing Darwin's core insight using minimal parameters.

- **Wind and wave deformation**

To represent coastal erosion resulting from sediment transport and complex hydrodynamics [TG13], we simplify the model by using a vector field painted by the user representing wind and wave directions, deforming the global island (Figure 3.9 suggest that island outlines are influenced by the average wind direction). This vector field allows radial symmetry to be broken in a controllable way, without relying on a full erosion simulator.

- **Independence of islands**

While archipelagos share currents, sediments, and flood each other's lagoons, simulating multi-island interactions is computationally expensive and hard to control. We treat each island as an isolated entity to generate multi-island scenes by using a simple blending of individual islands, keeping our method fast and predictable.

#### Conclusion

These assumptions, grounded in theories of coral reef island formation, provide a practical foundation for procedural modelling. While they simplify the full complexity of geological and biological processes, they capture the essential dynamics needed to produce plausible island structures. Simultaneously, they ensure our system remains intuitive, controllable, and computationally efficient, enabling the generation of diverse and plausible islands suitable for interactive design.

### 3.2.2 Traditional terrain generation methods

Procedural generation of terrain has been a well-researched area in computer graphics and simulations, where the goal is to create large, realistic landscapes with minimal manual input. Various methods have been developed over the years to generate terrains automatically, from noise-based methods to physically based erosion simulations, sketch-driven mechanisms, and more recently, deep learning techniques.

However, coral reef islands present unique challenges due to the combination of long-term geological processes (mainly subsidence and coral reef growth) and environmental interactions (i.e. erosion caused by wind and waves). In this section, we review the key techniques proposed for terrain generation, highlighting their limitations for coral reef island formation, and positioning our work as an approach that addresses these challenges.

#### Noise-based terrain generation

Noise-based procedural generation remains one of the most widely used techniques for creating natural-looking terrains on height fields. Perlin noise [Per85], Simplex noise [Per01], and the Diamond-square algorithm [FFC82] are foundational algorithms that generate pseudo-random yet continuous variations across a grid, producing terrain features that resemble organic landscapes. These techniques have been widely adopted in computer graphics and game development due to their efficiency and visual appeal.

Beyond basic noise functions, advanced techniques such as fractal Brownian motion (fBm) and multifractal noise have been introduced to add finer-scale variation and detail [MKM89, EPW*03]. fBm combines multiple layers, or "octaves," of noise at different frequencies and amplitudes, producing terrains that exhibit realistic and varied features. The combination of noise with domain warping and signal processing techniques has been explored in depth in procedural modelling literature [RLL*10], enabling further control over visual complexity and terrain realism.

Noise functions are often paired with falloff maps to produce island-like terrains, where elevation gradually decreases towards the edges of the domain, mimicking coastlines and basic island shapes (see Figure 3.10). Various methods for enhancing island generation using noise and falloff blending have been proposed for applications in games and virtual worlds [Ols04]. While these techniques excel at producing large, visually diverse landscapes quickly, they suffer from several key limitations when applied to the modelling of coral reef islands.

However, limitations arise when applying these methods to coral reef islands as noise-based modelling lack grounding in geological or biological reality. They generate spatial patterns through mathematical noise, not through simulations of real-world processes such as volcanic subsidence or coral accretion. The signal processing parameters typically involved (frequency, lacunarity, gain, amplitude, ...) are tuned for visual effect rather than scientific plausibility. As highlighted in procedural modelling

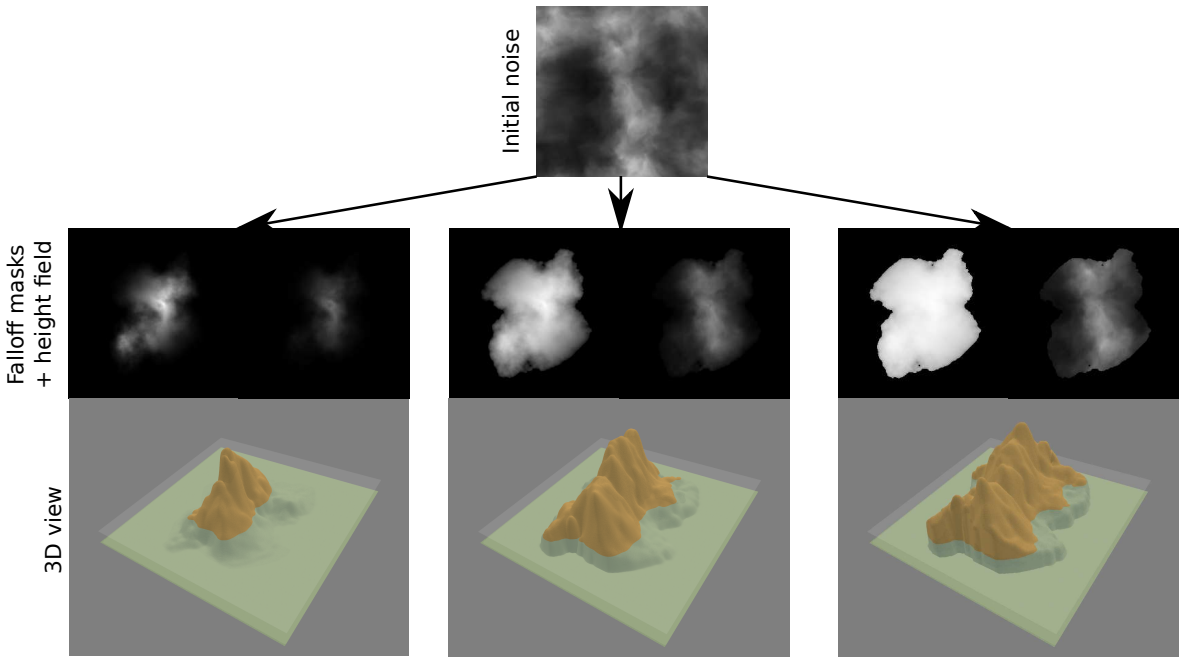



**Figure 3.10:** *Three different results of an island generated from noise functions. In each case, the initial height field is identical, computed using flow noise. The falloff masks are created by combining fBm noise, modulated by the Euclidean distance from the image centre, and further shaped through domain warping and power-law remapping of the mask values. The only parameter modified between the three examples is the exponent used in this remapping. The resulting islands differ significantly in shape and scale, and the method offers limited control over specific coastal features such as lagoons and reefs.*

surveys [SKG*09, GGP*19], this disconnect results in a lack of semantic control and poor correlation with actual environmental dynamics, making it difficult to represent phenomena such as reef rings, lagoons, or atoll structures in a biologically or geologically coherent way.

Moreover, the biological aspects of coral growth are inherently tied to environmental conditions. Coral reefs form and persist only within specific ranges of water depth, sunlight, salinity, and water quality. Their growth patterns are further influenced by ecological health, nutrient availability, and symbiotic relationships. These dependencies are extremely difficult to capture in procedural noise models, which are not designed to reproduce such complex and coupled dynamics.

Our approach goes beyond the randomness of noise-based generation by incorporating real-world geological and biological processes into the terrain formation pipeline. Specifically, we model the gradual subsidence of volcanic islands and the upward growth of coral reefs, both of which are central to the long-term evolution of coral reef islands. By embedding these natural processes directly into the generation algorithm, we produce terrains that are not only more realistic but also more controllable. This integration of scientific modelling with procedural flexibility allows us to overcome the inherent limitations of traditional noise-based techniques and more accurately represent the complex formation of coral reef island systems.

### Simulation-based modelling

Simulation-based terrain modelling methods aim to increase realism by replicating natural processes such as erosion, sediment transport, tectonic uplift, and vegetation growth. Unlike noise-based strategies, which rely on random functions, simulation-based processes model causality and temporal dynamics to describe how a terrain evolves over time under physical or biological forces. These methods are often used to enhance base terrains, adding geologically plausible detail and structure

[BTHB06, SKG*09].

*Hydraulic and thermal erosion*

Hydraulic erosion models simulate the impact of flowing water on the landscape by modelling erosion, sediment pickup, transport, and deposition. Early implementations by [MKM89] laid the groundwork for erosion in procedural generation, while more recent works have accelerated these simulations using GPU architectures [MDH07] and particle-based methods [NWD05]. These simulations either follow Eulerian fluid models or Lagrangian particle systems to capture terrain displacement.

Thermal erosion, by contrast, simulates mass movement due to gravity, redistributing material from steeper slopes to gentler gradients, akin to landslides or soil creep [BTHB06]. These erosion models generate realistic fluvial networks and landforms, but they are parameter-sensitive and computationally expensive.

Moreover, such models are generally designed for terrestrial landscapes and lack mechanisms for simulating underwater sedimentation, reef growth, or biogenic processes crucial to coral reef island formation. These models typically simulate time scales relevant to geomorphological processes (hundreds to thousands of years), which are mismatched with both the faster dynamics of biological processes (e.g., coral growth) and the slower geological evolution of reef islands.

We propose our new particle-based erosion simulation method, adapted for underwater and terrestrial landscapes, in Chapter 5.

*Tectonic uplift and geologic simulation*

Geological simulation approaches such as those proposed by [CBC*16, CCB*17] and extended by [SPF*23] model terrain evolution through crustal deformation and tectonic uplift. These methods simulate isostatic adjustments, plate tectonics, or local uplift phenomena, often over geological timescales.

Although well-suited for mountain-building processes or fault line modelling, these methods are not designed to account for biogenic terrain formation, such as coral reef accretion, which is critical for simulating coral reef islands. As a result, despite being physically grounded models, they do not capture the coupled geological and biological dynamics necessary for representing the long-term evolution of reef islands.

*Vegetation and ecosystem dynamics*

A number of simulation-based terrain models integrate ecological dynamics to reflect the feedback between terrain and living systems. For instance, [ECC*21] and [CGG*17] simulate interactions between vegetation and terrain erosion, modelling plant colonisation, growth, and their influence on soil stability and moisture retention.

These ecosystem simulations allow more complex landscape evolution by considering biotic agents; however, they are designed primarily for terrestrial plants and temperate ecosystems. Coral colonies, in contrast, are marine organisms with strict environmental requirements (e.g., limited depth, adequate sunlight, nutrients presence, warm water temperatures, ...). Accurately simulating these dependencies would require significant computational resources. We present a deeper analysis of current ecosystem simulations in Chapter 4's state of the art.

Furthermore, coral growth is not a passive process such as sediment accumulation, but an active accretion system that builds calcium carbonate structures over thousands of years. These unique growth mechanisms, constrained by marine ecology, fall outside the scope of existing vegetation or soil-plant-water feedback models.

Li propose to procedurally synthesizes coral colonies structures by emulating stochastic reef growth with a Diffusion-Limited Aggregation process, producing reef patterns but does not propose control

over the simulation [Li21]. The integration of user input such as sketch interfaces remains essential for controlling the generated output.

#### Conclusion

While simulation-based models represent a significant advancement over purely procedural models, they fall short in capturing the coupled geological and biological dynamics that shape coral reef islands. They are either computationally intensive, domain-specific, or biologically inapplicable, highlighting the need for a new class of terrain generation tools that embed long-term marine biogeomorphological processes into the procedural pipeline.

### 3.2.3 Sketch-based terrain modelling

The term "sketching" encompasses several definitions: either refering to performing hand or body gestures, creating a rough drawing, or outlining an idea in a simplified form. Accordingly, sketch-based modelling in 3D computer graphics can be understood through three complementary perspectives, each centred around a distinct core concept: interaction, construction and interpretation.

First, "sketching" may focus on interaction, where gestures captured through hand or body motion are used to manipulate virtual objects, often in immersive environments (virtual or augmented reality), using established techniques such as sculpting and distortion [OSSJ09, CA09]. Second, "sketching" may involve construction, where simple geometric primitives (curves, parametric shapes, implicit surfaces, ...) are combined under constraints to build a more complex model. Finally, "sketching" may centre on interpretation, where the user draws strokes on a 2D canvas and the system analyses their meaning to generate a plausible 3D model.

While sketch-based modelling encompasses a wide range of techniques, including gesture-driven interaction in immersive environments, this work focuses primarily on construction and interpretation aspects. In particular, construction serves as the foundation for procedural generation techniques using geometric primitives and constraints, while interpretation becomes relevant when exploring data-driven approaches using deep learning to infer terrain structure from sketches. Interaction-based techniques, though significant in other contexts, fall outside the scope of this work.

To clarify terminology in this chapter, we refer to the constructive approach as sketch-based, and to the interpretive, learning-driven approach as learning-based. It is important to note, however, that the boundaries between these categories are inherently blurry and often overlap in practice.

In procedural terrain generation, sketch-based construction workflows enable users to shape landscapes by manipulating high-level geometric primitives through intuitive sketching interfaces. These methods allow the definition of key terrain features (mountains, valleys, coastlines, ...) by drawing their outlines on a two-dimensional canvas, which are then procedurally transformed into 2.5D or 3D terrain representations. This technique offers a high degree of control, making it particularly effective for creative applications such as video games and robotics simulations, where modelling is primarily user-driven.

#### Curve-based modelling

Sketch-based terrain generation often begins with user-defined curves which act as high-level constraints to guide the shape of the terrain. These curves represent silhouettes, ridgelines, valleys, or feature outlines. Once defined, they are interpreted by the system and translated into elevation changes through various computational techniques. This type of approach allows for intuitive control over large-scale landforms while maintaining a procedural foundation for terrain synthesis.

One of the earliest and most influential works in this domain was introduced by Gain et al. (Figure 3.11), enabling users to sketch silhouettes, ridges, and spine curves to define complex terrain structures [GMS09]. The method employs multiresolution surface deformation and propagates wavelet-based noise from the sketched features to their surroundings, allowing users to generate detailed, natural-looking terrains from minimal input. This method demonstrated the effectiveness of combining intuitive sketch input with procedural detail synthesis.

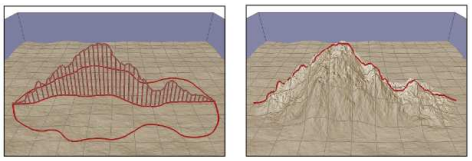


**Figure 3.11:** *Gain et al. propose to edit terrains through sketching by diffusing a noisy height function over a bounded region (left), effectively modelling mountain ranges (right) [GMS09].*

We draw direct inspiration from this work's dual-view sketching strategy, combining top-view and profile sketches, which closely aligns with our concentric curve and height-profile input approach.

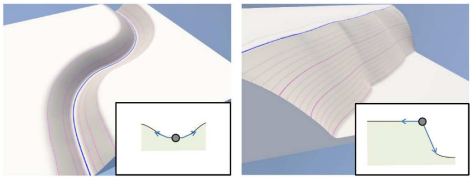


**Figure 3.12:** *Hnaidi et al. define the surface by elevation curves with constrained slopes, forming with ease river beds and cliffs [HGA*10].*

Building on this idea, Hnaidi et al. proposed a technique based on diffusion equations [HGA*10]. In their method, curves are annotated with geometric constraints (elevation, slope, and roughness), and a diffusion process is used to interpolate these constraints across the terrain surface. This results in smooth, continuous elevation fields that conform to user-defined features such as rivers, ridgelines, or cliffs (Figure 3.12). The use of parameterised curves as terrain anchors allows for precise control over landform shaping, while maintaining a high degree of automation.

In a different interaction paradigm, Tasse et al. introduced a first-person sketching interface, where users draw terrain silhouettes from a particular camera viewpoint [TEC*14]. These silhouettes are then projected into 3D, and a deformation algorithm adjusts the terrain so that the drawn features are visible exactly as intended from the user's perspective (Figure 3.13). This method supports complex silhouettes with occlusions, T-junctions, and cusps, and represents an immersive and perceptually grounded approach to sketch-based terrain editing.

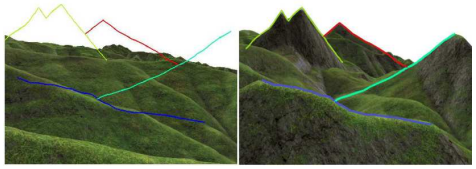


**Figure 3.13:** *Tasse et al. use sketching in association with camera position and direction to constrain the edition of height fields [TEC*14].*

These methods demonstrate the expressive power of curves as terrain-defining elements. By enabling users to sketch intuitive shapes and constraints, they bridge the gap between artistic intent and procedural complexity. Curve-driven methods remain foundational in terrain modelling, particularly when user control over large-scale structure is essential. While we do not use the diffusion model, the idea of sketch-defined elevation constraints along curves informs our use of user-defined shape boundaries.

### Constraint-based modelling

Constraint-based and gradient-based families of methods focus on exerting precise control over terrain features via formal specifications such as elevation values, slopes, or gradient fields. These methods prioritise structural accuracy and procedural consistency, making them particularly suited to applications that demand terrain realism, integration with geographic data, or fine-grained editing capabilities.

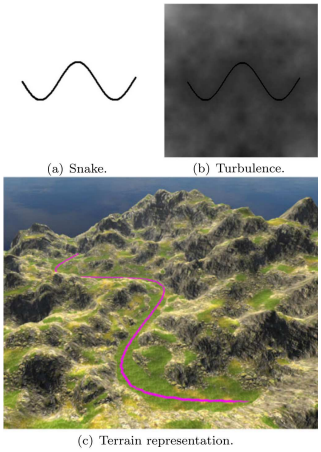



**Figure 3.14:** *Gasch et al. use a top-view sketch to apply constraints (a) on the noise function modelling a terrain, creating an intuitive means to create paths and roads in the resulting terrain (b and c) [GCRR20].*

A representative example of constraint-based modelling is presented by Gasch et al., proposing a method for procedural terrain generation that respects user-defined elevation constraints [GCRR20]. Their system allows users to fix values at specific control points (e.g., paths, landmarks) and then solve a system of equations to propagate these constraints throughout the terrain. The method integrates these constraints with a noise-based procedural function to preserve natural randomness while conforming to user intent (Figure 3.14). This approach is especially valuable when generating terrains that must align with real-world data or gameplay constraints.

We do not adopt this constraint-solving mechanism, but conceptually relate our profile sketch input to a localised height constraint.

Extending this idea, Talgorn and Belhadj introduce a real-time sketch-based terrain generation system based on a GPU-accelerated implementation of midpoint displacement [TB18]. Users sketch curves with explicit elevation values, which act as absolute constraints, and the system extrapolates and interpolates terrain surfaces in real-time. Moreover, the user input influences the elevation constraint and interpolation method properties of the midpoint displacement algorithm using the Red, Green and Blue channels (Figure 3.15). Their model supports both global and local control over interpolation curvature and roughness, and introduces semantic labelling of sketched features (e.g., ridges vs. rivers) to influence how terrain propagates around constraints. This combination of sketch-based input, constraint propagation, and semantic control enables expressive, large-scale terrain modelling at interactive speeds. We adopt their notion of semantic labels and hierarchical constraint propagation to support real-time terrain shaping with sketch-defined features without the fractal interpolation.

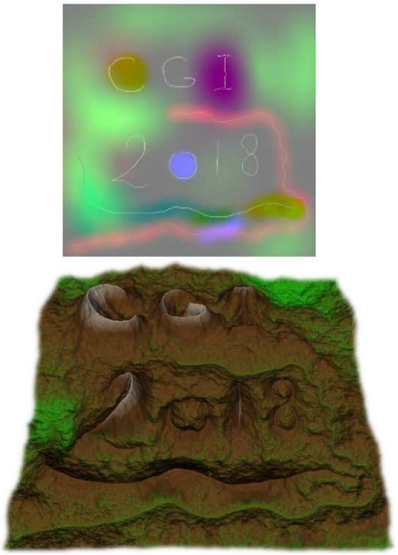


**Figure 3.15:** *Talgorn and Belhadj use top-view sketching to encode properties in the noise function used to model the terrain surface [TB18].*

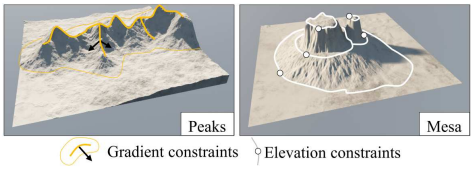



**Figure 3.16:** *Guérin et al. apply gradient (left) and height (right) constraints on curves and diffuse them on the terrain to create landscapes with lightweight and multi-scale representations [GPM*22].*

Building on the need for more intuitive editing, Guérin et al. introduce gradient domain paradigm for terrain modelling [GPM*22]. Rather than specifying elevation values directly, users interact with slope-based representations, allowing for the manipulation of terrain inclination and the integration of local edits into global terrain structure (Figure 3.16). By controlling terrain gradients and reconstructing elevation through integration, this method enables seamless blending between regions and supports a natural editing workflow, particularly for sculpting realistic mountain ridges, valleys, or plateaus. This gradient-domain editing paradigm offers interesting insights; we adopt an alternative approach as we operate in the elevation domain with semantic control.

Both paradigms offer complementary strengths: constraint-based methods ensure precise adherence to user-defined features or data sources, while gradient-based systems provide fluid, perceptual control over terrain shaping. Together, they represent a shift toward high-level modelling tools that maintain procedural expressiveness while granting users a deeper degree of terrain control.

### Semantic terrain representation

Beyond geometric sketching and low-level constraints, a third class of methods explores high-level terrain construction, where users guide terrain generation using abstract or semantic inputs. These approaches aim to simplify the authoring process by allowing users to describe what a terrain should contain (e.g., a mountain or a valley) without specifying how to generate its geomety. Such methods often rely on symbolic sketching, sparse representations, or domain-specific visual cues, offering powerful tools for conceptual design and inverse procedural modelling.

In this vein, Génevaux et al. propose a method for representing terrains as sparse combinations of procedural primitives, referred to as "terrain atoms" [GGP*15]. These atoms are stored in a dictionary and are either extracted from real-world data or generated synthetically. The terrain is modelled as a linear combination of these features, forming a Sparse Construction Tree that blends primitives in a compact and expressive form (Figure 3.17). This representation facilitates terrain editing, amplification, and reconstruction from coarse user input.

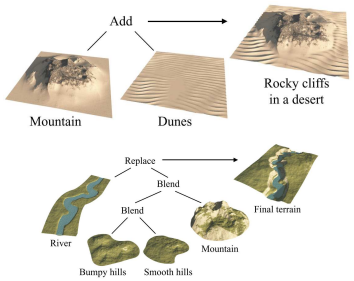



**Figure 3.17:** *Génevaux et al. symbolically define a terrain by an aggregation of patches with real-world meaning [GGP*15].*

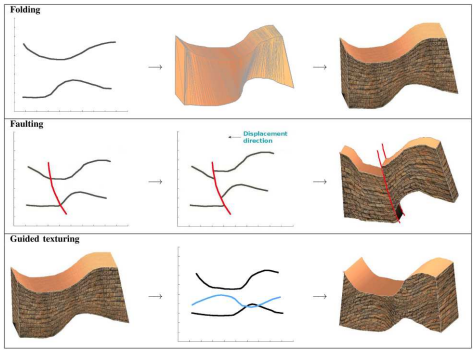



**Figure 3.18:** *Natali et al. use sketching to describe geological phenomena: (top) geological layer deformation, (middle) abrupt displacement from faulting, and (bottom) texture deformation [NVP12].*

An illustrative and domain-specific use case is presented by Natali et al., who introduce a system for rapid visualisation of geological concepts [NVP12]. Here, users sketch schematic representations of subsurface structures such as faults, folds, or strata, and the system generates plausible 3D visualisations of geological terrains (Figure 3.18). Their tool is designed primarily for educational and exploratory purposes, enabling geoscientists and students to create, manipulate, and communicate complex geological scenarios through intuitive sketch input. Although it extends beyond traditional terrain elevation modelling, the work exemplifies how sketch-based systems can operate on a conceptual level and support domain-specific semantics. This work inspired our use of sketch strokes to define deformation fields, although our implementation targets structured terrain generation rather than schematic visualisation.

These high-level approaches demonstrate the potential of sketch-based modelling not only as a geometric tool, but as a semantic interface between human intention and terrain synthesis. By abstracting terrain construction into symbolic or feature-based representations, they allow users to create rich, expressive landscapes without directly engaging with low-level geometry, making them particularly valuable for tasks involving conceptual design, education, and inverse procedural modelling.

#### Conclusion

The methods presented in this section illustrate the diversity of sketch-based approaches for constructive terrain modelling, from curve-driven shape control to constraint-based editing and semantic abstractions. While these methods offer valuable tools for intuitive user interaction and procedural shaping, they often lack ecological grounding, multi-view integration, or the ability to produce structured data suitable for training generative models. In our work, we reinterpret and adapt elements from these approaches (dual-view sketching, curve-based region definition, and deformation fields) to support the generation of coral reef islands through a hybrid procedural and learning-based pipeline. This constructive sketch-based foundation enables us to balance user control with scalable terrain generation in data-sparse domains.

### 3.2.4 Deep learning

Over the past decade, deep learning has revolutionised almost all areas of computer graphics and procedural content creation by learning complex, data-driven priors directly from examples. Unlike purely procedural or sketch-based methods, which rely on hand-tuned noise functions or geometric constraints, neural networks are trained to capture subtle patterns and high-frequency details without explicit programming of each effect. In terrain synthesis, this enables models to infer realistic elevation structures, textures, and region transitions from training data, even when that data is sparse or synthetic. In the context of coral reef islands, where high-resolution digital elevation models are rare, deep learning offers a way to abstract away low-level procedural rules and directly learn the mapping from semantic layouts (label maps) to plausible height fields. In the following sections, we first review general Generative Adversarial Networks (GANs) and then focus on their conditional variant, cGANs, which forms the backbone of our sketch-to-terrain translation pipeline.

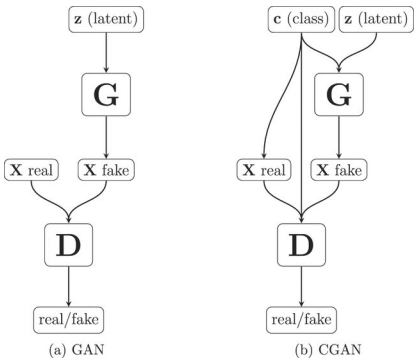



**Figure 3.19:** *The general structure of GAN and cGAN networks is similar: (a) a generator network G is trained to take some noise z as input to try to create a "realistic" output $X_{fake}$, and a discriminator network D is trained in parallel to distinguish generated data from real data. (b) cGAN networks introduce a class input c, used by both the generator and discriminator in their inference process. In the end, only the generator is used to create new data.*

### Generative Adversarial Networks

GANs, introduced in [GPM*14], are a class of generative models in which two neural networks are trained in opposition (Figure 3.19):

- a generator $G$ learns to produce synthetic "fake" data samples that resemble those from a target distribution given a random noise $z$,
- a discriminator $D$ learns to distinguish "real" samples from those generated.

Through this adversarial process, the generator gradually improves its ability to mimic the underlying data distribution, enabling the creation of realistic outputs from random input.

GANs have been widely adopted for image synthesis, texture generation, and data augmentation, among other tasks. In terrain modelling, they offer the potential to generate realistic landforms by learning directly from real-world data, without requiring hand-crafted procedural rules. The following works demonstrate how different GAN variants have been applied to terrain synthesis, each with its own assumptions, design trade-offs, and limitations.

Early terrain GANs used unconditional generation, synthetising elevation maps from pure latent noise, without any spatial or semantic guidance. Wulff-Jensen et al. train a Deep Convolutional GAN (DCGAN) to generate realistic digital elevation models of mountainous landscapes, showing that terrain-like structures could emerge from purely data-driven learning [WRMB18]. The model captures local elevation statistics and allowed for latent space interpolation, enabling smooth variations across generated terrains. However, the lack of spatial conditioning make it difficult to control or constrain features such as ridges, valleys, and coastlines, resulting in landscapes that reflected training set distributions but offered no means for intentional design.

Spick and Walker extend this approach with a Spatial GAN that generates height and texture maps jointly [SW19]. By conditioning the generation process on spatial coordinates, their model enforced local consistency and reduced structural artefacts. This integration simplified the content pipeline by fusing geometry and appearance into a single pass.

Yet despite improved quality, the generation process remained fundamentally uncontrolled: users are unable to specify terrain layout, features, or semantics. As with earlier GANs, the model learned to mimic terrain distributions but could not support authoring or guided synthesis.

Later works introduce multi-stage architectures in which a second conditional GAN refines or interprets the output of a first-stage generator. Beckham and Pal propose a two-step pipeline where a DCGAN generates bare terrain heightmaps from noise, and a conditional pix2pix network adds texture based

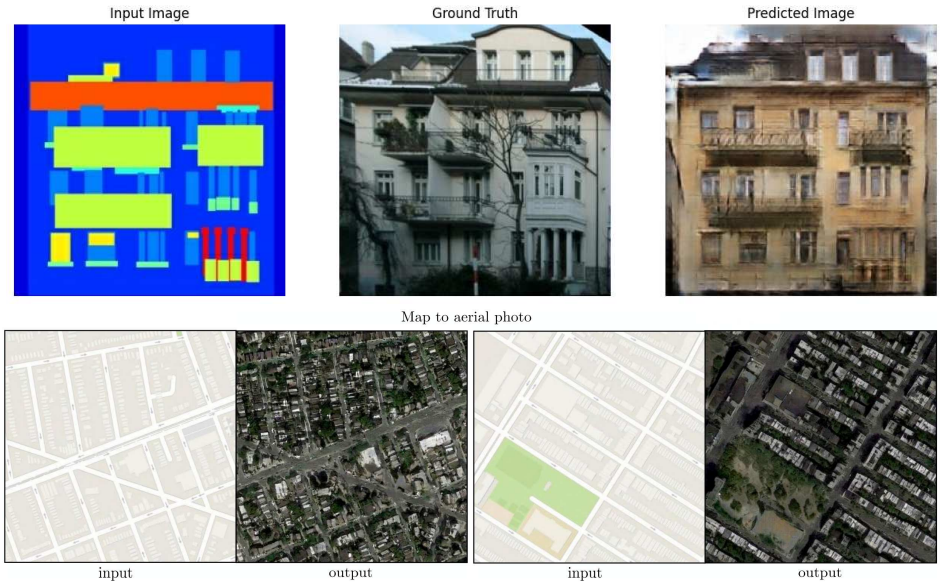



**Figure 3.20:** *Example of pix2pix image-to-image translation: (top) a trained model converts semantic label maps into realistic façade images, and (bottom) constructs highly plausible aerial images from navigation map sketches. This paradigm generalises to many tasks, including terrain generation.*

on semantic cues [BP17]. This separation of geometry and appearance allows for basic stylisation and terrain remixing, but control for the initial terrain remains limited due to purely noise-driven GAN. Using a conceptually similar structure, Panagiotou and Charou reverse the mapping [PC20]: an unconditional GAN first synthesises aerial RGB imagery from noise, which is then passed to a cGAN trained to predict plausible DEMs. While this image-to-DEM translation enables realistic terrain reconstruction, it lacks any semantic or structural user control and relies on large paired datasets, limiting their applicability in settings such as coral reef islands, where training data are scarce. In both cases, despite the introduction of a conditional refinement stage, the generation process remains fundamentally anchored in an unconstrained latent input, offering limited authoring capability and no intuitive tool to shape landform structures.

These methods show a shift towards using conditional models to guide terrain generation with more structure. But because they start from random noise, they offer little real control over the layout or semantic of the terrain. A natural next step is to guide the generation process directly from user-defined semantic inputs, such as sketches or label maps, to produce terrain that reflects both the training data and the user's intent.

### Conditional GANs for terrain generation

While traditional GANs generate data from noise, cGANs extend this concept by incorporating additional information, often called a class map or label map, to guide generation toward user-specified outcomes [MO14] (the $c$ class in Figure 3.19). This makes them particularly attractive for structured content synthesis tasks, including terrain generation, where user input often defines large-scale layout and realism must emerge from learned detail. The pix2pix framework [IZZE17] is the canonical cGAN formulation for image-to-image translation. It uses a U-Net generator conditioned on an input image (a sketch or a label map, for example), and a PatchGAN discriminator that evaluates realism at the patch level, thus encouraging fine detail and local consistency (Figure 3.20 presents two image-to-image translations from the same pix2pix network architecture).

In terrain generation, the use of cGANs remains surprisingly rare. The most directly relevant precedent is the work of Guérin et al., who train a pix2pix-style cGAN to map sketched terrain features such as valleys, ridgelines, or peaks into full-resolution digital elevation models (DEMs) [GDG*17]. Their results demonstrate that cGANs can plausibly reconstruct complex topographic forms from sparse

semantic cues, offering a promising balance between user control and learned realism. Lochner et al. updates the image-to-image process [LGP*23] by replacing the cGAN with a Denoising Diffusion Probabilistic Model [HJA20, ND21], improving the percieved quality of the resulting terrains through an iterative noising and denoising process, trained on a large DEM dataset. Diffusion models are however known to be multiple orders of magnitude slower than GANs or cGANs [XKV22, HRJ*23], making them unusable for interactive sessions on a personal computer.

Naik et al. inserts a Variational Autoencoder [KW22] before the cGAN phase to first compress the user input in a latent space [NJSR22], allowing to forward pass the cGAN with variations of the user input sketch or interpolate between different inputs in the 128 dimensions offered by the latent space.

An alternative approach from Voulgaris et al. uses a GAN-based system that maps sparse "altitude dot" maps to plausible terrain imagery, acting as a minimal-interaction generative authoring tool [VMP21]. While not strictly a cGAN, their system reflects a similar spirit: conditioning generation on lightweight user constraints.

Despite these advances, no standard pipeline exists for generating detailed terrains from semantic layout maps using cGANs, particularly in biologically driven environments. The potential of this family of methods remains underexploited, especially when it comes to coupling user control with long-term geological plausibility.

### Conclusion

In our method, we address this gap by training a pix2pix cGAN to transform label maps (semantic region maps produced by procedural sketch-based modelling) into plausible coral reef island height fields. Each input map encodes key zones of an island (e.g., lagoon, reef crest, beach, island core) using categorical labels, serving as semantic constraints for terrain synthesis. The cGAN generator learns to condition elevation details on both the global layout and the implicit patterns encoded in the training data. By generating our own synthetic dataset with procedurally modelled coral reef islands (see Figure 3.21), we overcome the shortage of labelled elevation data and enforce geological coherence via data generation design. The choice of the pix2pix architecture is tailored by its short training and inference time and the availability of well-maintained implementations that support reproducibility, but any other conditional architecture could be substitued such as pix2pixHD or BicycleGAN [WLZ*18, ZZP*18].

This setup offers several key advantages:

- respecting user-defined structure while allowing the generator to introduce realistic variation,
- removing procedural biases (radial symmetry and fixed island typologies),
- enabling the generation of irregular, non-circular landforms while implicitly modelling geological processes such as subsidence and coral accretion through training-time priors.

---

In this context, conditional GANs emerge as a powerful yet underutilised tool for terrain modelling. By training on procedurally generated coral reef island data, we show that sketch-conditioned learning can effectively bridge the gap between high-level user intent and geologically plausible terrain synthesis. We note that few-shot or one-shot learning methods are emerging such as Liu and Benes's Graph Neural Network usage, possibly solving the dataset scarcity problem, at the cost of methods falling out of the interactive-time scope with multiple seconds at inference time [LB25].

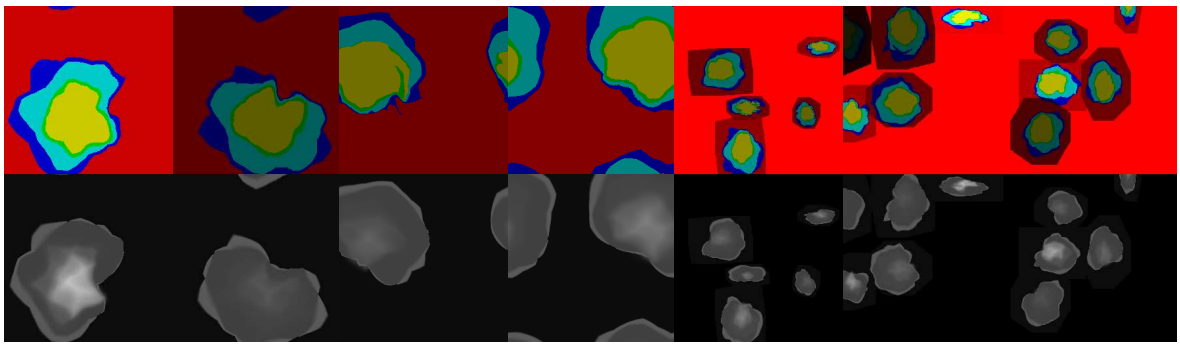


**Figure 3.21:** *Examples of procedural region maps used for training: left to right, canonical island, off-centre island, elongated shapes, and multi-island scenes. These maps serve as semantic inputs to our cGAN.*

## 3.3 Description of our method

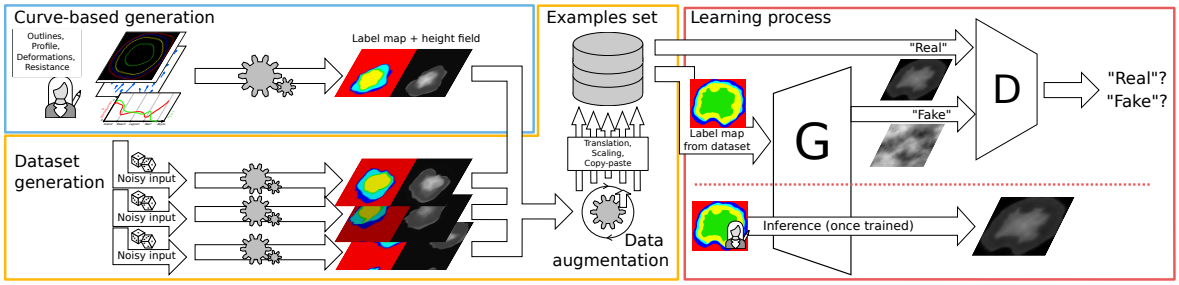



**Figure 3.22:** *Blue box: our pipeline first prompts the user to create single pairs of coral reef island height fields and label maps using our proposed curve-based modelling algorithm (Section 3.4). Yellow region: the same algorithm is applied multiple times on randomly altered versions of the user input to create a dataset, which we further enhance with data augmentation techniques (Section 3.5). Redbox: finally, we train a conditional Generative Adversarial Network, which can then be used standalone to create height fields from labelled sketches (Section 3.6).*

Our method for procedural generation of coral reef islands is composed of two independent modelling techniques shown in Figure 3.22. First, a curve-based algorithm (top-left blue block, presented in Section 3.4) parametrises the surface of islands via a top-view sketch interface, describing the outlines of its constituent regions and a stroke-based wind field deforming them, as well as a profile-view sketch describing, for each region, its altitude and resistance to deformations. User inputs are given as 1D functions (altitude and resistance), a 1D polar function (island outlines), and a 2D vector field (wind field), as shown in Figure 3.23, which are convenient to randomise through the use of noise. While effective for encoding geological principles and generating structured examples, this algorithm cannot by itself provide the full freedom and irregularity required for realistic island design. This motivates the use of a learning-based component capable of generalising beyond the procedural rules.

Second, we propose a learning-based generation algorithm (right red blocks Figure 3.22, developed in Section 3.6), which trains a neural generator $G$ to transform a labelled image into a height field and a discriminator $D$ to distinguish height fields provided from the training set against height fields generated by $G$. This adversarial training strengthens the outputs of the generator, up to the point where we discard the discriminator and training data, only keeping the generation step (bottom-right). Users are then able to provide unseen label maps and obtain height fields as close as possible to the training dataset. The procedural method lacks expressiveness but provides a large set of varied synthetic data; the cGAN offers expressiveness but requires a large dataset. Their combination is

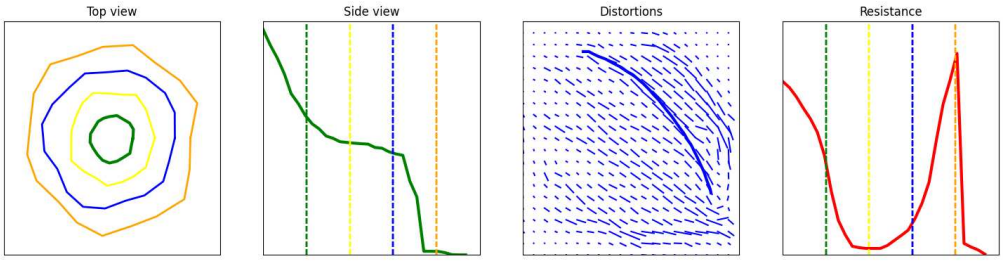



**Figure 3.23:** *The user interacts directly on the island by editing the different canvases in any order. This UI shows, from left to right: the top-view sketch with the different outlines of each region, the profile-view sketch with the outlines represented by dotted lines, the wind velocity sketch drawn with strokes (last stroke is visible), and the resistance function showing here a high resistance at the top of the island and on the front reef.*

therefore natural: the procedural model acts as a data generator, while the cGAN removes its structural biases.

We connect these two algorithms through a process of dataset generation (central orange block Figure 3.22, described in Section 3.5) in order to enforce the benefits of each while reducing their limitations. Given the initial curve-based user inputs, we create a dataset composed of similar samples processed by our first algorithm, rasterised into a 2D image of the parametrised regions, and paired with the resulting height field. We then significantly augment the dataset by employing various data augmentation techniques: translations, scaling, and copy-pasting of multiple islands in a single sample. We then obtain a dataset large enough for training our cGAN and enable users to procedurally create coral reef islands with a high level of control.

## 3.4 Procedural island generation

We propose a structured process for generating coral reef island terrains that takes the user's sketches and produces a complete 3D terrain model. This process begins with the creation of an initial height field based on user inputs, and applies wind deformation to introduce natural variations, and finally integrate coral reef features via subsidence and coral growth modelling.

The generation of coral reef islands with our system begins with two intuitive sketch-based inputs from the user: a top-view sketch and a profile-view sketch, which define the island's horizontal layout and vertical elevation profile. Users further refine the terrain by applying wind deformation strokes, approximating wind and waves effects on the island's shape. This combination of sketches and wind inputs gives users precise control over both the island's structure and its natural variations, such as irregular coastlines or concave features. In this section we demonstrate the usefulness of these sketches, describe the technical details, and present the method in Algorithm 1.

### 3.4.1 Initial height field generation

The top-view sketch defines the island's outline as seen from above. Using a simple drawing interface, the user can delineate the boundaries between key regions of the island, including the island itself, beaches, lagoons, and surrounding abysses (Figure 3.24). Our system assumes that these regions are arranged concentrically around the centre of the island, with each boundary defined by a radial distance from the centre (Figure 3.27).

Each region's boundary is represented in polar coordinates $(r_{\mathbf{p}}, \theta_{\mathbf{p}})$, with $r_{\mathbf{p}}$ indicating the radial distance from the island's centre and $\theta_{\mathbf{p}}$ representing the angular position. This polar representation allows us to map the user's sketch onto a circular framework, ensuring smooth transitions between regions and maintaining a coherent island layout.

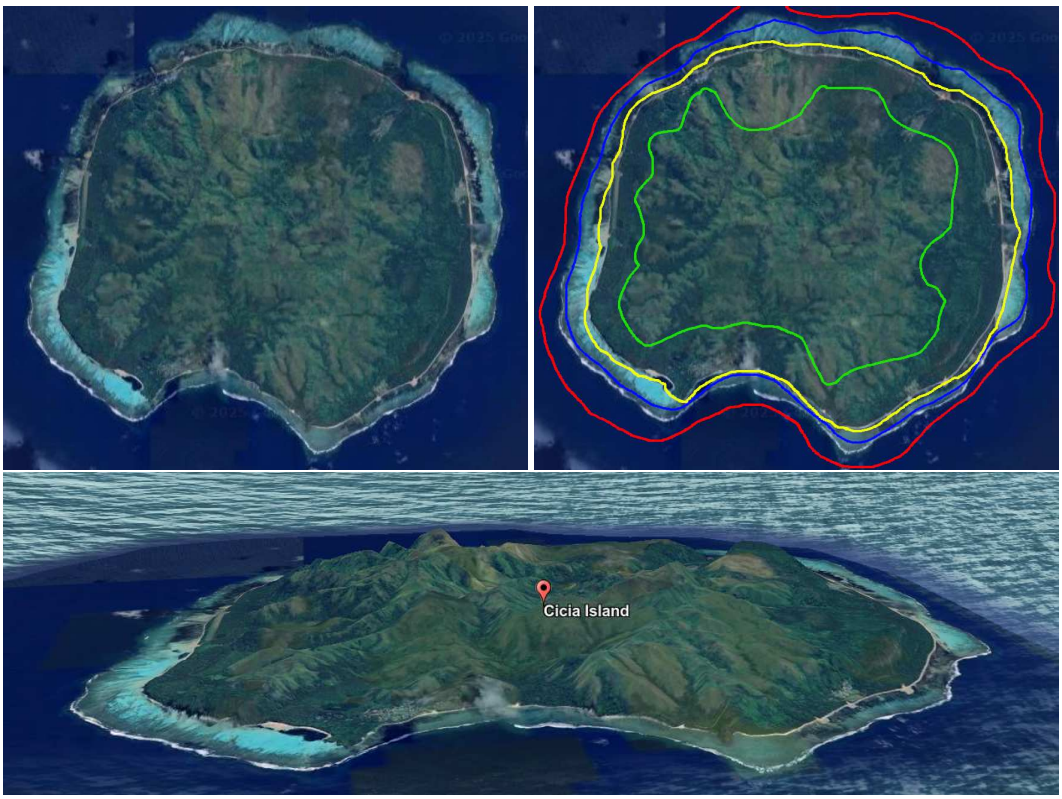



**Figure 3.24:** *(Left) A real-world example of aerial image (and 3D visualisation on bottom) of an island (Cicia Island) may be segmented into regions. (Right) We represent the different regions boundaries.*

In this sketch, the user defines the overall horizontal layout of the island, including the size and shape of each feature (Figure 3.25). Variations in the outline are introduced by allowing the radial distances to vary with angle, ensuring that the island is not strictly symmetrical and introducing more natural, irregular shapes.

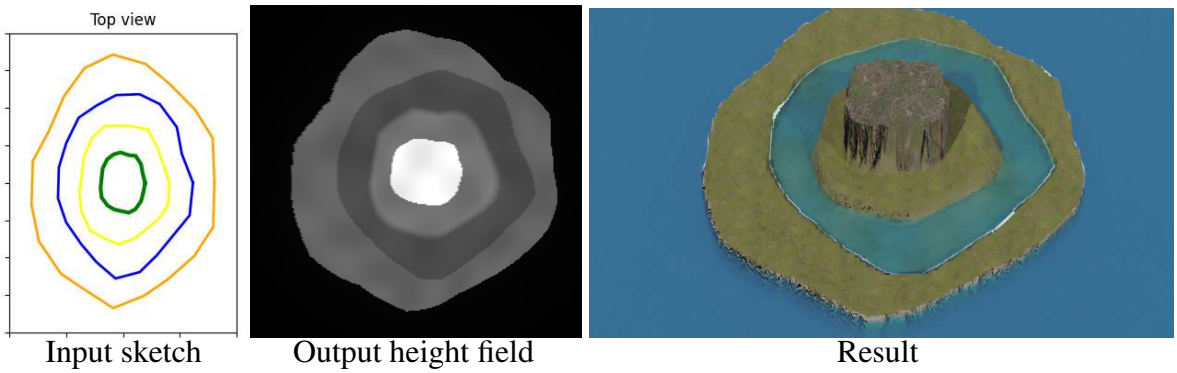



**Figure 3.25:** *Using only the outlines of the island as an input sketch (left), we provide a height field depending on the region in which it relies (middle). The resulting terrain is displayed (right).*

The profile-view sketch defines the vertical elevation profile of the island along any radial direction, offering control over the island's height. In this view, the user specifies the elevation of different regions of the island, such as the island peak, beach, lagoon, abyss, and everything in between, by drawing the corresponding profile curve (Figure 3.26).

These regions correspond to key terrain transitions: the highest point of the island (centre), the island border, the beach, the lagoon, and the deep-sea abyss. We interpolate these milestones into a

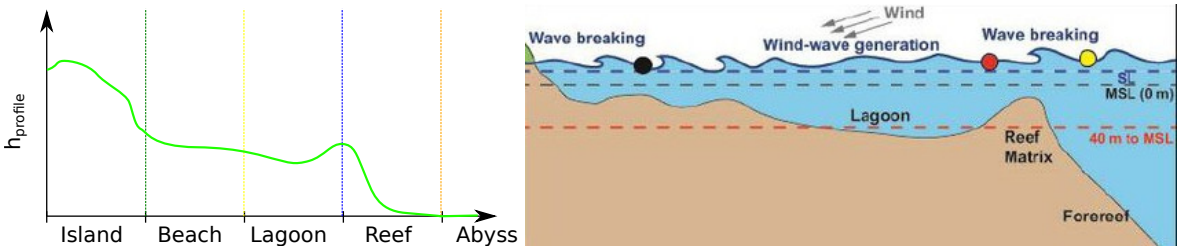



**Figure 3.26:** *(Left) A profile function $h_{profile}$ is defined as a 1D function and represents the surface from the centre of the island to the abysses. (Right) The cross-section representation of an island is often represented as a 1D function defined using terrain features as landmarks.*

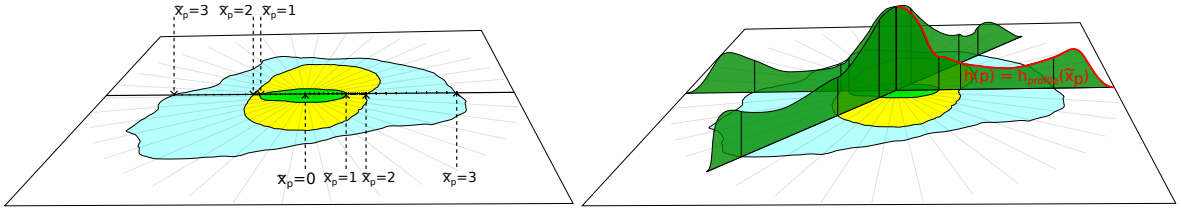



**Figure 3.27:** *The $\tilde{x}_{\mathbf{p}}$ parameter is used to stretch the 1D height function $h_{profile}(x)$ to fit the distances from the centre to the outlines of each region defined in the top-view sketch.*

continuous 1D height function $h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$, where $\tilde{x}_{\mathbf{p}}$ represents a non-uniform region distance from the island's centre, and $h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$ gives the height at each point. This continuous profile ensures smooth elevation transitions across the island.

By combining the top-view and profile-view sketches, our method generates a coherent 3D terrain model that accurately reflects the user's design by revolution modelling.

The generation of the coral reef island terrain begins by transforming the user-defined top-view and profile-view sketches into a coherent 3D height field. This process combines the radial layout of the top-view sketch with the elevation information provided by the profile-view sketch, creating a terrain that accurately represents the desired features (i.e. the island, beaches, lagoons, and abyss).

For any point $\mathbf{p} \in \mathbb{R}^2$ on the terrain, we define its polar coordinates $(r_{\mathbf{p}}, \theta_{\mathbf{p}})$ relative to the center of the island $\mathbf{c}$ as:

$$r_{\mathbf{p}} = \|\mathbf{p} - \mathbf{c}\|, \qquad \theta_{\mathbf{p}} = \text{atan2}(\mathbf{p}.y - \mathbf{c}.y,\ \mathbf{p}.x - \mathbf{c}.x).$$

The radial distance $r_{\mathbf{p}}$ determines which region the point belongs to (island, beach, lagoon, reef, or abyss) based on the user-defined radial limits. Those outlines from the top-view sketch provide the exact boundaries between regions.

Each point's height is computed using the profile function $h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$. Instead of using the raw radial distance $r_{\mathbf{p}}$, we define a parametric region distance $\tilde{x}_{\mathbf{p}}$ that maps each point to a normalised position along the concentric regions (see Figure 3.27). The radial space is divided by user-defined boundaries $R_0, R_1, \ldots, R_n$, corresponding to the island centre, border, beach, lagoon, reef, and abyss.

When a point $\mathbf{p}$ lies between two boundaries $R_i$ and $R_{i+1}$, its parametric distance is

$$\tilde{x}_{\mathbf{p}} = i + \frac{r_{\mathbf{p}} - R_i}{R_{i+1} - R_i}, \tag{3.1}$$

where $i$ is the index of the region containing $\mathbf{p}$ (i.e., $R_i \leq r_{\mathbf{p}} < R_{i+1}$). This linear mapping stretches each region's radial span to the interval $[i, i+1[$, ensuring smooth interpolation across region boundaries. For any point $\mathbf{p}$, the height is finally computed as:

$$h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}}). \tag{3.2}$$

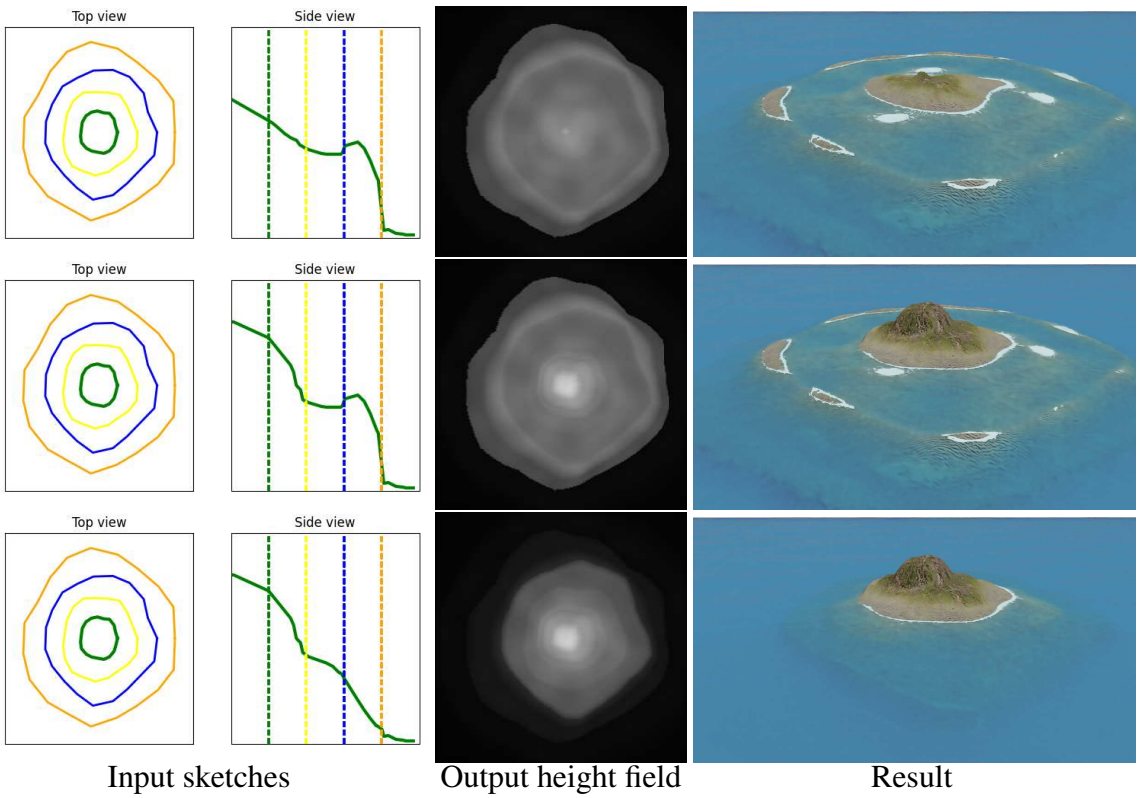



**Figure 3.28:** *Providing a smooth function between each region results in islands with plausible reliefs. We fixed the outlines (first column) while editing only the height function (second column) in order to produce, from top to bottom, a low island, a coral reef island, and finally an identical island without the reef.*

This approach ensures that the height field accurately follows the elevation profile specified by the user while maintaining smooth transitions between different regions of the island.

The result is a height field that captures both the radial structure of the island (from the top-view sketch) and the vertical elevation profile (from the profile-view sketch), producing an organic representation of islands with smooth transitions between the key terrain features. Three slightly edited height functions on an identical island layout and their associated outputs are presented in Figure 3.28. To avoid perfectly smooth surfaces, we also apply a very low-amplitude Perlin noise perturbation to the final height field, adding fine-scale irregularities.

### Wind and wave deformation

In island environments, wind and waves reshape islands through coastal erosion, which removes and redistributes material over time. Simulating this physically requires sediment transport and hydrodynamics, which is both computationally intensive and difficult to control interactively. Instead, we approximate the morphological outcome of erosion with a deformation field: user-drawn strokes define a wind velocity field that warps the island, introducing large-scale asymmetries without explicitly removing matter. To account for the fact that not all regions respond equally to erosion, we also introduce a resistance function, which modulates deformation's strength across regions. For example, shallow coastal areas are strongly deformed, while the deep abyss or resistant volcanic core remain largely unaffected. This combination provides an efficient and intuitive proxy for the long-term shaping role of wind and waves. Although real coastlines are shaped by both wind and wave processes, we represent them jointly through a single user-defined wind field, which serves as a controllable proxy for their combined morphological effect.

The wind field is represented as a series of strokes drawn by the user on a 2D canvas. Each stroke represents a parametric curve, where the direction and strength of the wind are encoded as a vector field. The user controls the wind's direction by drawing a set of curves, and we interpret them to create a velocity field that defines how the terrain is deformed.

As the user draws a stroke, a set of control points are generated along it to create a curve, with the option to adjust the stroke's width. The width of each stroke determines the area of influence around the curve, where wider strokes result in broader deformations of the terrain. The deformation strength decreases with distance from the curve using a Gaussian falloff function using the stroke width as standard deviation. This ensure that the terrain transitions smoothly from deformed regions to non-deformed areas.

Once the strokes are applied to the velocity field, we displace each points of the terrain accordingly. The height field, originally generated from the user's sketches, is modified by the wind field to create non-radial features, breaking the initial radial symmetry and producing a more organic island shape (Figure 3.29).

The deformation is controlled via a user-defined vector field representing the direction and strength of wind flows across the terrain. Users interact with our system by drawing strokes on a 2D canvas represented as centripetal Catmull-Rom splines $C$ [CR74], a type of parametric curve that allows for smooth, continuous paths representing wind patterns. Each stroke defines a wind flow in the curve's direction $C'$, a strength $S_C$, and an effect width $\sigma$ (Figure 3.30); these wind flows are used to displace the terrain, approximating the gradual reshaping of the island due to wind and wave erosion. The strength of the displacement is controlled by a Gaussian scaling function $G_\sigma(x)$ that smoothly decreases with the distance from $\mathbf{p}_C^*$, the closest point on the parametric curve $C$, as follows:

$$\begin{aligned} \Phi_C(\mathbf{p}) &= S_C \cdot \frac{C'(\mathbf{p}_C^*)}{\|C'(\mathbf{p}_C^*)\|} \cdot G_\sigma\left(\|\mathbf{p} - \mathbf{p}_C^*\|\right), \\ G_\sigma(x) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}. \end{aligned} \tag{3.3}$$

The final deformation vector $\Phi(\mathbf{p})$ is computed as a sum of the influences from all strokes:

$$\Phi(\mathbf{p}) = \mathbf{p} + \sum_{C \in \text{curves}} \Phi_C(\mathbf{p}). \tag{3.4}$$

Once the deformation vector $\Phi(\mathbf{p})$ is computed, the terrain height at point $\mathbf{p}$ is adjusted by displacing $\mathbf{p}$ to a new point $\Phi(\mathbf{p})$. We then compute the final height $h(\Phi(\mathbf{p})) = h_{\text{profile}}(\tilde{x}_\mathbf{p})$, or

$$\tilde{h} = h \circ \Phi^{-1}. \tag{3.5}$$

This process introduces variations in the terrain, distorting the coastline, creating concave regions, and breaking the original radial symmetry defined by the top-view and profile-view sketches.

To ensure that certain regions of the terrain such as deep-water areas remain relatively unaffected by the wind, a resistance function $\rho(\tilde{x}_\mathbf{p})$ is applied. The resistance function modulates the effect of deformation based on the previously computed piecewise parametric distance $\tilde{x}_\mathbf{p}$, with the same interaction means as the $h_{\text{profile}}$ function (Figure 3.31).

For example, regions near the coastline (such as the beach and lagoon) have lower resistance, allowing for significant deformation (emulating coastal erosion from wave energy), while regions farther away (such as the abyss) have higher resistance, limiting the wind and coastal erosion impact.

The deformation vector previously described is then scaled by the resistance function at each point $\mathbf{p}$, such that the final deformation vector becomes

$$\tilde{\Phi}(\mathbf{p}) = \left(1 - \rho(\tilde{x}_\mathbf{p})\right) \cdot \Phi(\mathbf{p}). \tag{3.6}$$

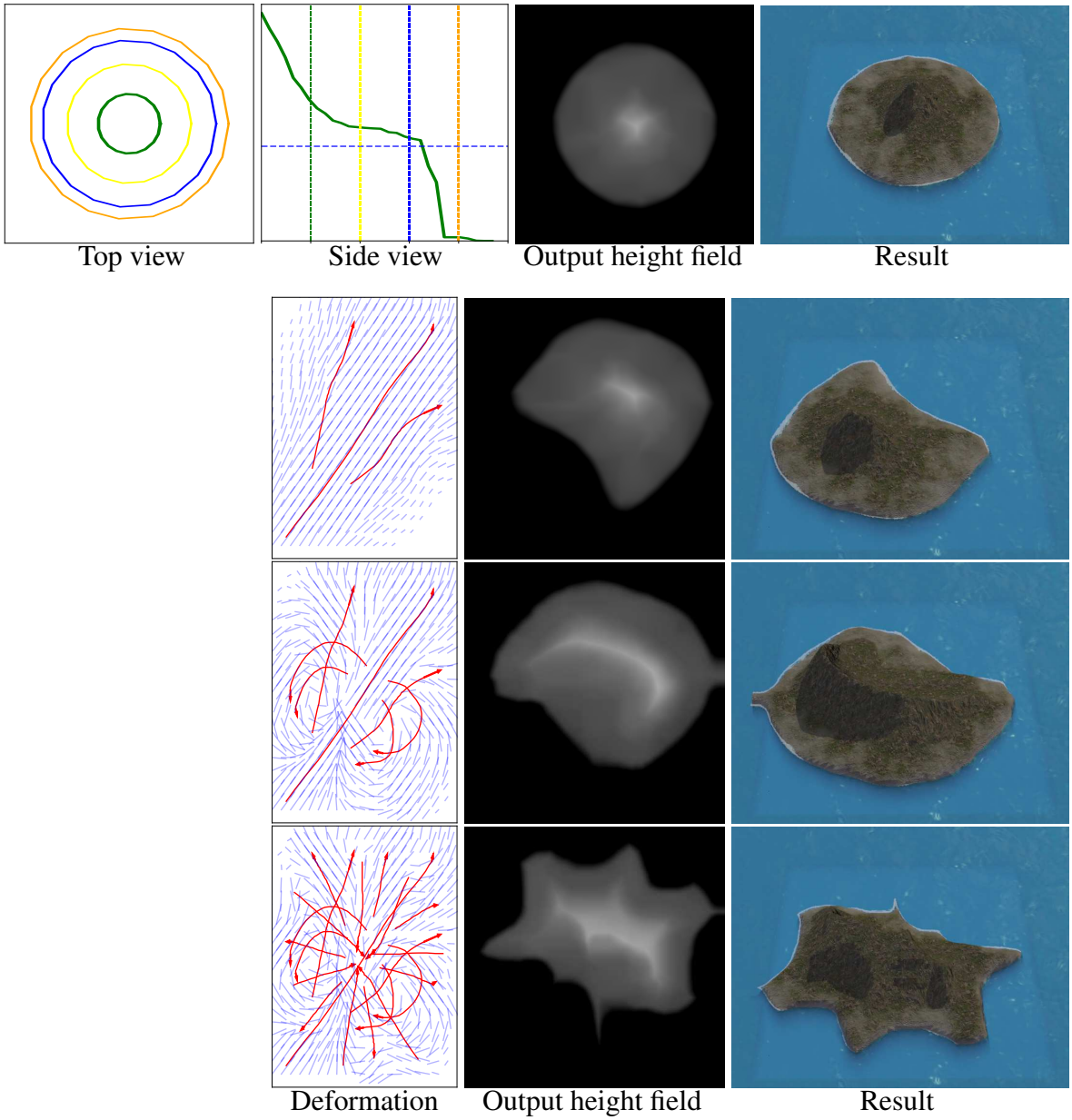



**Figure 3.29:** *Top row: A circular island layout and a given profile sketch creates a simple round island height field. Each following row shows a progression in an edition session, conserving the initial top-view and side-view sketch, by manipulating the deformation field (blue lines) with respectively 3, 7 and 20 user strokes (red arrows). The final result breaks the initial radial symmetry of the island layout.*

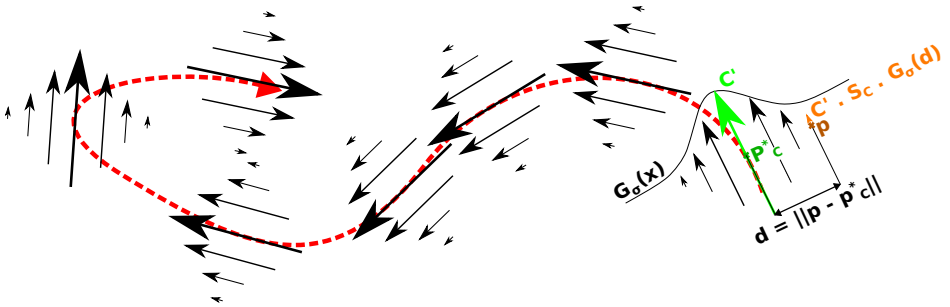



**Figure 3.30:** *From the parametric curve defined by a user (red), we define the velocity field by considering the directional derivative $C'$ of the curve $C$ at the closest point $\mathbf{p}_C^*$, modulated by a Gaussian distance function $G_\sigma(x)$ and scaled by the user-defined strength $S_C$.*

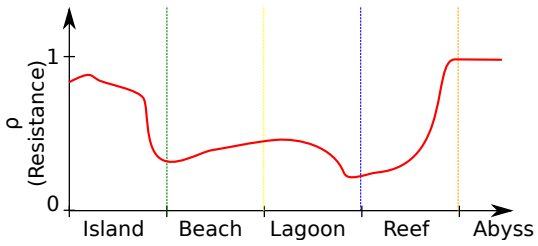



**Figure 3.31:** *The resistance function of the island is defined with a similar strategy as the $h_{profile}$ function. The resistance to erosion and deformation arises from multiple factors such as depth, materials, wind shadowing, biotic and abiotic factors, ... Modelling all these factors is complex. As such, using a user-defined approximation through a resistance function $\rho$ allows for more control.*

**Input:** Top-view boundaries $\{R_0(\theta), \ldots, R_n(\theta)\}$, profile function $h_{\text{profile}}$, resistance $\rho(\tilde{x}_{\mathbf{p}})$, user strokes $\{C\}$ with width $\sigma$ and strength $S_C$
**Output:** Deformed height field $\tilde{h}$ and label map $\tilde{I}$

`// (1) Initial height field and label map from sketches`
**foreach** *pixel* $\mathbf{p} \in \mathbb{R}^2$ **do**
  $r_{\mathbf{p}} \leftarrow \|\mathbf{p} - \mathbf{c}\|$
  $\theta_{\mathbf{p}} \leftarrow \text{atan2}(\mathbf{p}.y - \mathbf{c}.y,\ \mathbf{p}.x - \mathbf{c}.x)$
  Find $i$ such that $R_i(\theta_{\mathbf{p}}) \le r_{\mathbf{p}} < R_{i+1}(\theta_{\mathbf{p}})$
  $\tilde{x}_{\mathbf{p}} \leftarrow i + \dfrac{r_{\mathbf{p}} - R_i(\theta_{\mathbf{p}})}{R_{i+1}(\theta_{\mathbf{p}}) - R_i(\theta_{\mathbf{p}})}$
  $h(\mathbf{p}) \leftarrow h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$
  $I(\mathbf{p}) \leftarrow i$

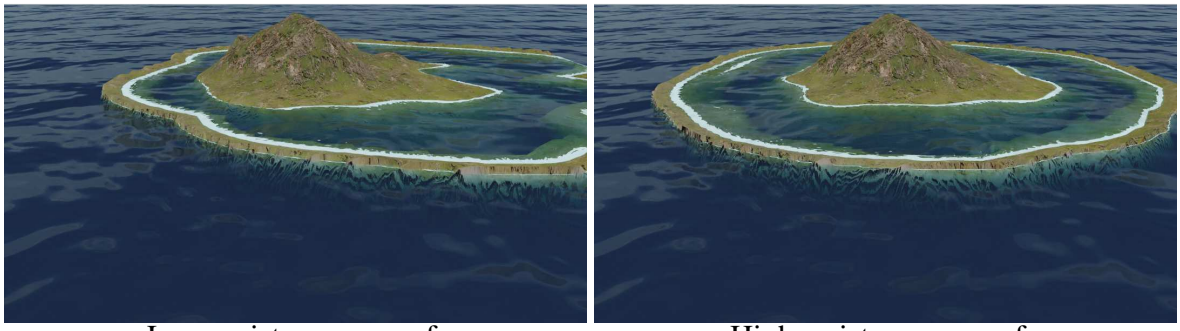
`// (2) Wind-field deformation from strokes`
**foreach** *pixel* $\mathbf{p} \in \mathbb{R}^2$ **do**
  $\Phi(\mathbf{p}) \leftarrow \mathbf{0}$
  **foreach** *stroke* $C$ **do**
    $\mathbf{p}_C^* \leftarrow \arg\min_{\mathbf{q} \in C} \|\mathbf{p} - \mathbf{q}\|$  `//` $\mathbf{p}_C^* \leftarrow$ `closest point of the curve` $C$ `to the point` $\mathbf{p}$
    $\Phi_C(\mathbf{p}) \leftarrow S_C \cdot \dfrac{C'(\mathbf{p}_C^*)}{\|C'(\mathbf{p}_C^*)\|} \cdot G_\sigma(\|\mathbf{p} - \mathbf{p}_C^*\|)$  `//` $G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$
    $\Phi(\mathbf{p}) \leftarrow \Phi(\mathbf{p}) + \Phi_C(\mathbf{p})$
  $\tilde{\Phi}(\mathbf{p}) \leftarrow \big(1 - \rho(\tilde{x}_{\mathbf{p}})\big) \cdot \Phi(\mathbf{p})$ `// Modulation from resistance function`
$\tilde{h} \leftarrow \tilde{\Phi}^{-1} \circ h$  `//` $\tilde{h}(\mathbf{p}) \leftarrow h(\mathbf{p} - \tilde{\Phi})$ `for backward warping`
$\tilde{I} \leftarrow \tilde{\Phi}^{-1} \circ I$
**return** $\tilde{h}, \tilde{I}$

**Algorithm 1:** Sketches to height field with wind deformation.

The deformation process results in a modified height field where the terrain has been warped according to user-defined strokes; Figure 3.32 illustrates the difference between non-resistant and resistant islands. This deformation introduces non-radial features, such as concave coastlines or irregularities along the beach and lagoon, making the island appear more natural and varied.

Both the height field and the label map (which tracks the terrain regions) are updated to reflect the deformation (Figure 3.33). This ensures that the semantic information of the terrain remains consistent even after the terrain has been warped. The label map is deformed in the same way as the height field, preserving the logical structure of the island for further post-processing, such as texturing or mesh instancing.

In Figure 3.29, a simple circular island is generated from the initial height field. By applying strokes along one side of the island, the deformation process can create concave regions along the coastline,



Low resistance on reefs High resistance on reefs

**Figure 3.32:** *(Left) Given a uniform velocity field and a resistance function similar to Figure 3.31, the coasts are smoothly eroded while the interior of the island is almost unaffected. (Right) Modifying the resistance function to affect a strong resistance to borders emulates the effect of coast reinforcements.*

making the shape more irregular and mimicking the effects of real-world wind and wave erosion. The resistance function ensures that while the beach and lagoon areas are deformed, the abyss remains largely unaffected as they are far from the wind and wave effective areas, preserving the island's overall structure.

### 3.4.2 Coral reef modelling

Once the terrain has been generated and deformed by the wind, we emulate the subsidence of the volcanic island and, in parallel, the growth of coral reefs. These processes reflect the long-term geological evolution of coral reef islands, where the volcanic island gradually sinks (subsides) while coral reefs grow upward to "keep up" with the sinking landmass.

#### Subsidence

To account for the island subsidence due to tectonic activity we propose to scale the initial height field downward, modelling procedurally the effect of the volcanic island slowly sinking into the ocean. The user provides a subsidence rate $\lambda \in [0,1]$, which represents the proportion by which the island has sunk. The subsidence is applied uniformly, meaning all terrain points on the island sink with the same intensity, regardless of their original height or location.

The subsided height field $h_{\text{subsid}}(\mathbf{p})$ is computed by scaling the original height field $h(\mathbf{p})$ with the subsidence factor:

$$h_{\text{subsid}}(\mathbf{p}) = (1-\lambda) \cdot h(\mathbf{p}).$$

#### Coral reef growth

As the volcanic island subsides, coral reefs grow upward to remain close to the water surface, following the "keep-up" strategy observed in most real-world coral formations. Coral growth is restricted to regions where the depth is within an optimal range for coral development, typically from the water surface to around 30 metres below before becoming much scarcer.

The coral reef features (reef crest, back reef, and fore reef) are modelled separately from the subsidence process. We generate a coral feature height field $h_{\text{coral}}(\mathbf{p})$, which remains unaffected by the island's subsidence.

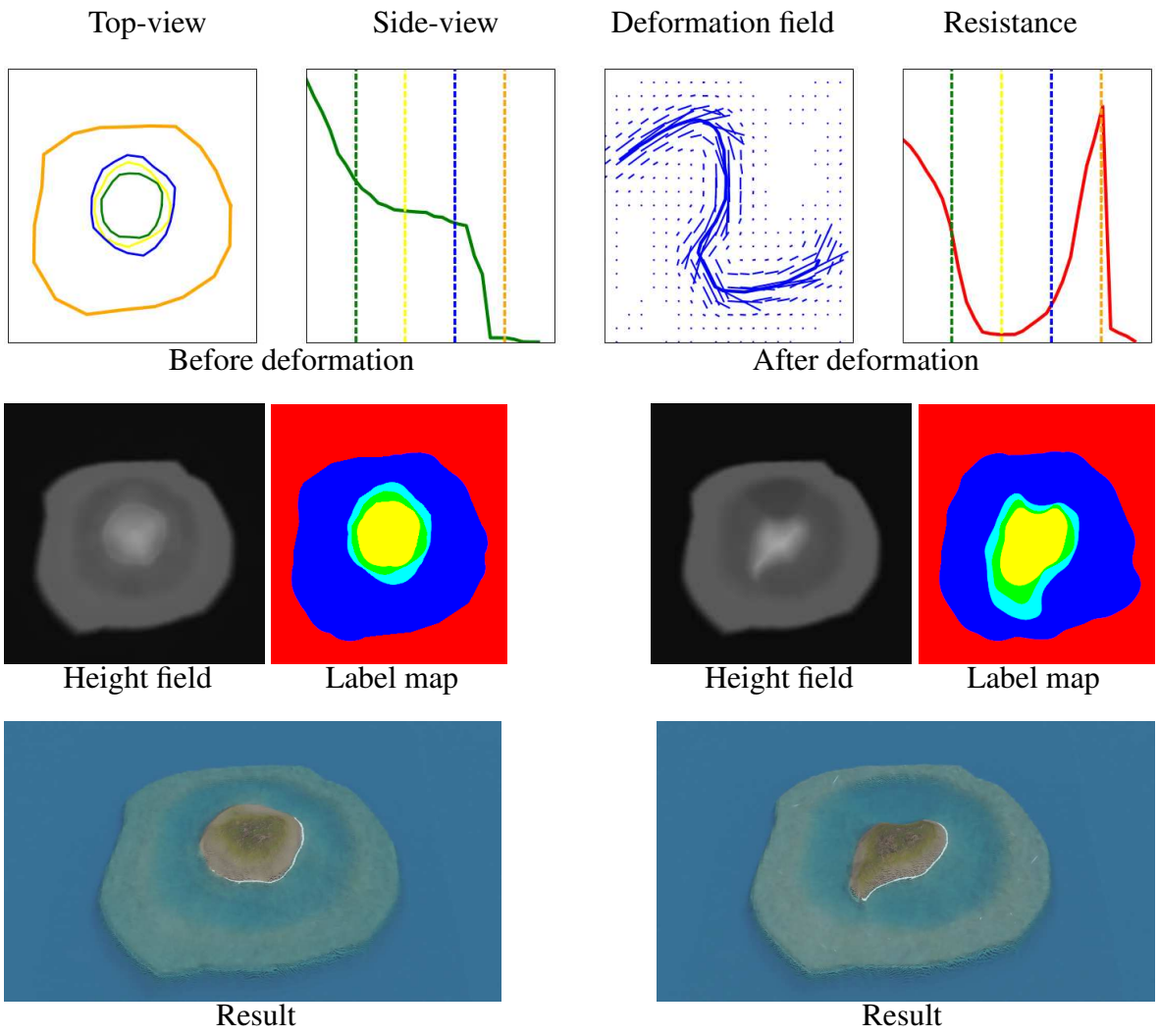



**Figure 3.33:** *A top-view velocity field is defined from user-provided strokes (top, blue), in association with a resistance function (top, red); a height field is deformed accordingly. Left: original height field and render; right: altered results. The beach and lagoon regions are defined with low resistance, which is visible by having only these regions deformed in bottom results.*

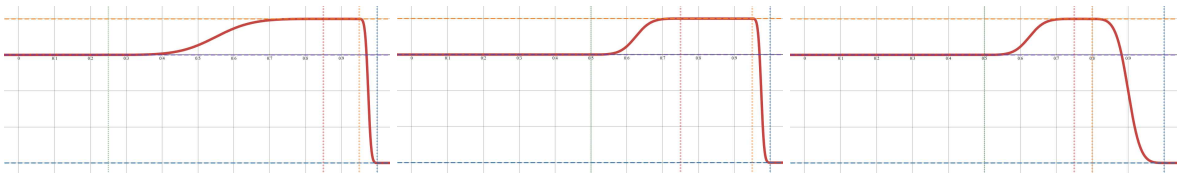


**Figure 3.34:** *The modelling of the reef growth in our model is described by a piecewise function $h_{coral}$ which is flat in the lagoon, the crest and abyss, and follows a smoothstep function as transitions for the back reef and fore reef regions. The model is easily edited by tuning the height values and delimitations of the subregions. Here, three examples of reef growth function with different delimitations (vertical dotted lines), providing more or less smooth transitions between reef subregions.*

In our model, coral reef growth is entirely independent of the subsided terrain. Even as the volcanic island sinks, coral growth is driven only by the proximity of terrain to the water surface, ensuring that coral features always remain near the surface, irrespective of how much the island subsides.

We define distinct reef zones anchored at specific depths:

- Reef crest near sea level: $h_{\text{crest}} = -2$,m,
- Back reef and lagoon: $h_{\text{back}} = -20$,m,
- Fore reef sloping to abyss: $h_{\text{abyss}} = -100$,m.

Each reef subregion is defined over a parametric domain $x \in [0,1]$, with $x = 0$ the beginning of the reef region and $x = 1$ its end, directly inputting the parametric distance $x = \tilde{x}_{\mathbf{p}} - i_{\text{reef}}$ with $i_{\text{reef}}$ the ID of the reef region, resulting in a linear interpolation between the begining and the end of the reef region in the top-view sketch. For instance:

- Back reef: $x_{\text{back,start}} = 0$, $x_{\text{back,end}} = 0.5$,
- Reef crest: $x_{\text{crest,start}} = 0.75$, $x_{\text{crest,end}} = 0.8$,
- Abyss begins at $x_{\text{abyss,start}} = 1$.

We model transition zones between these regions using a smoothstep operator [Per02]:

$$\text{smoothstep}(x) = 3x^2 - 2x^3.$$

For conciseness, denote the interpolating function as:

$$S(a,b,x_0,x_1,x) = a + (b-a)\,\text{smoothstep}\left(\frac{x-x_0}{x_1-x_0}\right).$$

The complete coral height field, as displayed in Figure 3.34, is built as a piecewise function:

$$\begin{aligned} h_{\text{coral}}(x) = &\sum_{r \in \text{subregions}} \begin{cases} h_r & \text{if } x_{\text{r,start}} \le x \le x_{\text{r,end}} \\ 0 & \text{otherwise} \end{cases} \\ &+ \sum_{t \in \text{transitions}} \begin{cases} S(h_t, h_{t+1}, x_{\text{t,end}}, x_{\text{t+1,start}}, x) & \text{if } x_{\text{t,end}} < x < x_{\text{t+1,start}} \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{3.7}$$

### Blending height fields

The final step is to blend the subsided height field $h_{\text{subsid}}(\mathbf{p})$ with the coral feature height field $h_{\text{coral}}(\mathbf{p})$ to produce the final terrain. The goal is to ensure that coral features remain near the water surface while allowing the rest of the island to subside.

To achieve this, we use a smooth maximum function, which smoothly blends the two height fields, ensuring that the coral regions dominates where coral growth is present, while the subsided island terrain dominates in other regions. This blending method ensures that the transition between the coral and subsided regions is smooth and visually consistent.

We define our smooth maximum function $\text{smax}(a,b) \in \mathbb{R}$ for $(a,b) \in \mathbb{R}^2$ as the mean of two functions, $\text{smax}^-$ and $\text{smax}^+$ (shown in Figure 3.35), adapted from Inigo Quilez's smooth min function [Qui13], that respectively underestimate and overestimate the function max:

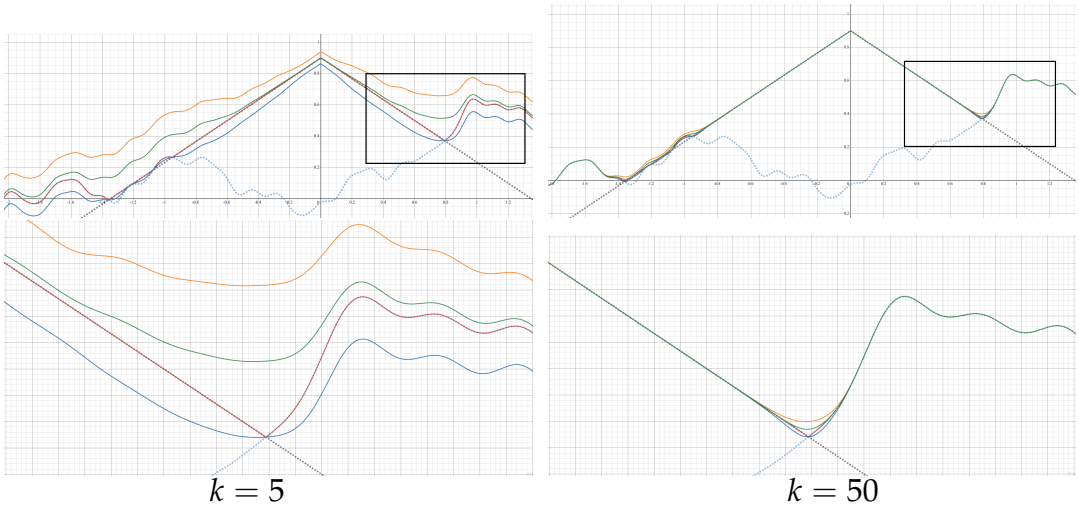



**Figure 3.35:** *The* $\text{smax}^+$ *operator (orange curve) is a function that is used to overestimate the maximum value of two input functions (dotted curves), especially when the difference between the two functions is small, while the* $\text{smax}^-$ *function (blue curve) tends to underestimate the* max *operator. Taking* smax *(green curve) as the average of* $\text{smax}^-$ *and* $\text{smax}^+$ *creates a smooth function closely approximating the* max *operator, even with low values of sharpness* $k$ *(Left:* $k = 5$*, right:* $k = 50$*). Bottom graphs show a zoom on the domain* $x \in [0.5, 1.3]$.

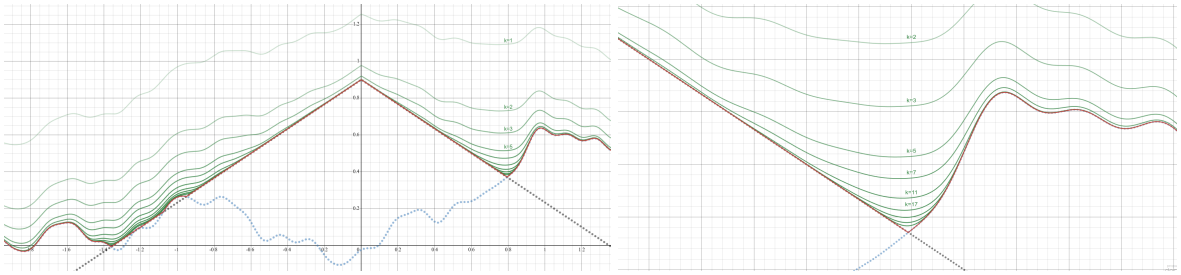


**Figure 3.36:** *Blending two functions* $f : \mathbb{R} \mapsto \mathbb{R}$ *(black, dotted) and* $g : \mathbb{R} \mapsto \mathbb{R}$ *(blue, dotted) with the* max *operator (red curve), causing discontinuities when* $f(x) = g(x)$*, and with the* smax *operator (green curves) with various sharpness values* $k$*, resolving the issue at the cost of slight overestimations with low values of* $k$*. Right presents a zoom on the domain* $x \in [0.5, 1.3]$.

$$\text{smax}^-(a,b) = a + \frac{b-a}{1+\exp\left(-k\cdot(b-a)\right)},$$
$$\text{smax}^+(a,b) = a + \frac{b-a}{1-\exp\left(-k\cdot(b-a)\right)},$$
$$\text{smax}(a,b) = a + \frac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2}.$$

Here, $a = h_{\text{subsid}}(\mathbf{p})$ is the height from the subsided island, $b = h_{\text{coral}}(\mathbf{p})$ is the height from the coral reef feature, and $k$ controls the smoothness of the transition (fixed arbitrarly here at $k = 5$ for heights ranging between 0 and 1). Higher values of $k$ bring the smax function closer to the max function (Figure 3.36).

This smooth max function guarantees continuity by preventing abrupt height and slope differences between coral regions and the subsided terrain, creating a smooth, gradual transition that mimics the natural blending of coral reefs with deeper areas. The coral feature height field takes precedence where coral can grow, typically in shallow regions. In deeper regions, such as the abyss, the subsided

height field naturally dominates, ensuring that the final terrain accurately reflects both subsidence and coral growth processes (Figure 3.37).

Note that the smax function is undefined for $a = b$, however, a proof of continuity for $\text{smax} \in C^\infty$ is provided in Annex A (along with a deeper analysis of the function), resulting in

$$\text{smax}(a,b) = \begin{cases} a + \dfrac{1}{2k} & \text{if } a = b, \\ \dfrac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2} & \text{otherwise} \end{cases}. \tag{3.8}$$

**Input:** Deformed height field $\tilde{h}$, subsidence rate $\lambda \in [0,1]$, coral growth function $h_{\text{coral}}$ (piecewise with transitions via smoothstep)
**Output:** Final height field $\hat{h}$

```
// (1) Subsidence
```
**foreach p do**
  $h_{\text{subsid}}(\mathbf{p}) \leftarrow (1-\lambda) \cdot \tilde{h}(\mathbf{p})$

```
// (2) Coral growth evaluation
```
**foreach p do**
  `// Use same` $\tilde{x}_{\mathbf{p}}$ `as in Algorithm 1`
  $x \leftarrow \tilde{x}_{\mathbf{p}} - i_{\text{reef}}$ `// ` $i_{\texttt{reef}}$ `= reef region ID`
  $c(\mathbf{p}) \leftarrow h_{\text{coral}}(x)$

```
// (3) Smooth maximum blend
```
**foreach p do**
  $\hat{h}(\mathbf{p}) \leftarrow \text{smax}\left(h_{\text{subsid}}(\mathbf{p}),\, c(\mathbf{p})\right)$

**return** $\hat{h}$

**Algorithm 2:** Subsidence, coral growth, and smooth blending.

### Output

The resulting terrain represents a plausible coral reef island, where the volcanic island has subsided, and coral reefs have grown upward to keep pace with the water level. The smooth blending between the subsided terrain and the coral features ensures a natural transition between regions (island, lagoon, coral reefs, and abysses).

One of the key strengths of this method is its flexibility, as the subsidence and coral reef growth processes are modelled independently, allowing for a wide range of configurations. Users can generate plausible island terrains with or without coral features, or apply the coral reef growth modelling to existing height fields from other sources. Note that each of the pseudo-code section proposed in Algorithm 2 can be replaced with surrogate methods if desired (e.g., realistic subsidence or coral reef simulations, or the use of another blending function).

## 3.5 Data generation

The creation of the dataset is done through the use of the procedural algorithm for which we alter the input parameters.

For each generation, the top-view and profile-view sketches use the user-given layout. Each outline of the top-view sketch $r^*$ is modified by adding values of Perlin noise $\eta$, giving the final outlines:

$$r(\theta) = r^* + \eta_{\text{outlines}}(\theta).$$

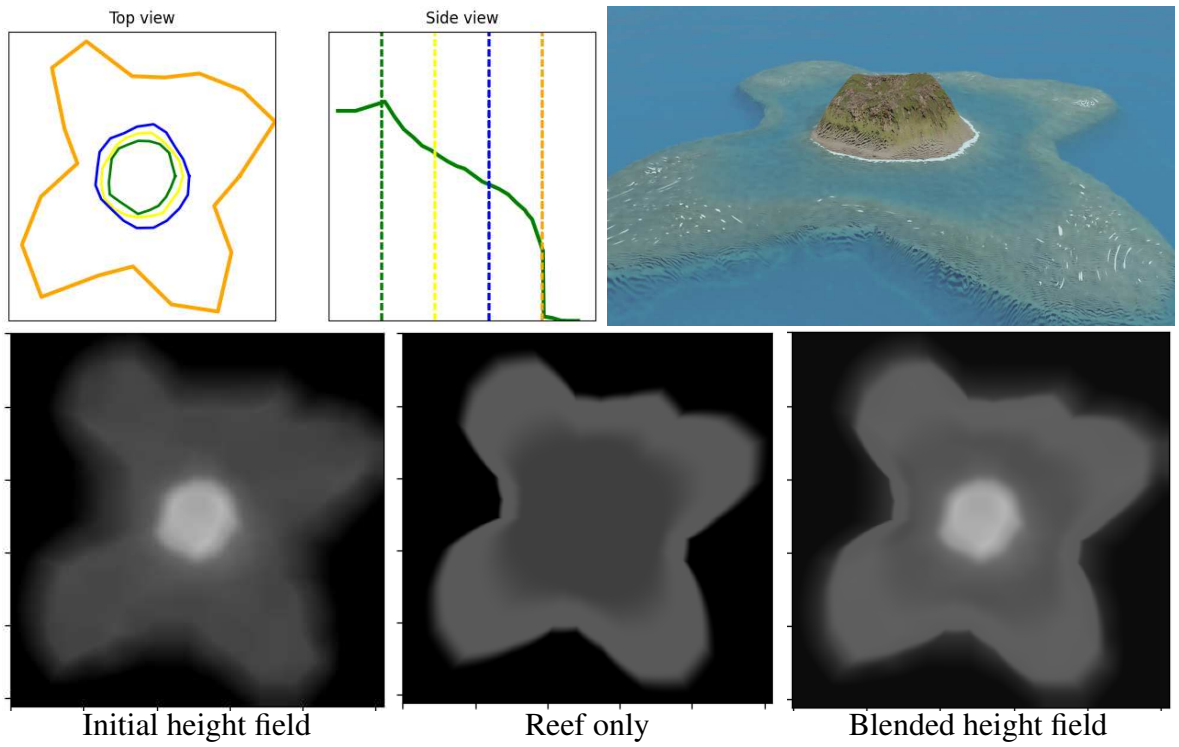



**Figure 3.37:** *Volcano with single vent. Bottom left: the initial height field is computed directly from the user input. Bottom centre: the reef height field is output from Equation* (3.7)*. Bottom right: blended result with Equation* (3.8)*.*

On the other hand, we define an initial profile-view sketch by defining $h^*_{\text{profile}}(\tilde{x}_{\mathbf{p}})$, the initial height function, for which fBm noise $\eta_{\text{height}}$ ($|\eta_{\text{height}}| \leq 0.2$ in the examples) is applied to obtain

$$h_{\text{profile}}(\tilde{x}_{\mathbf{p}}) = h^*_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \cdot \eta_{\text{height}}(\tilde{x}_{\mathbf{p}}).$$

An identical process is done for the resistance function:

$$\rho(\tilde{x}_{\mathbf{p}}) = \rho^*(\tilde{x}_{\mathbf{p}}) \cdot \eta_{\text{resistance}}(\tilde{x}_{\mathbf{p}}).$$

In the examples below, all terrains are defined in normalized coordinates $(x, y) \in [-1, 1]^2$ and $z \in [0, 1]$. Noise amplitudes used are bounded by $|\eta_{\text{outlines}}| \leq 0.2$, $0.8 \leq \eta_{\text{height}} \leq 1.2$, $0.8 \leq \eta_{\text{resistance}} \leq 1.2$.

Next, we generate a random deformation field. We do not target realism in stroke generation. Instead we generate a random number $n$ of strokes with random paths. Spread and intensity of each stroke is also randomised, providing complexity and variety in the results. The results presented in this chapter uses between 2 to 6 random wind strokes. Our random wind stroke paths are smooth, simplex noise-driven trajectories biased toward the island's center, ensuring that distortions remain mainly around the main landmass, with uniform sampling of spread $0.1 \leq \sigma \leq 0.3$ and strength $0.05 \leq S_C \leq 0.3$.

Finally, to further enrich the dataset with erosion-like structures, we add a set of radial strokes of small width and low strength, extending between the island centre and the exterior of the canvas, oriented alternately landward and seaward. This produces simple radial drainage patterns illustrated in Figure 3.38.

Once all inputs are set, we generate an example for multiple levels of subsidence $\lambda \in [0, 1]$ producing height fields that incorporate coral reef modelling along with its associated label map.

The pix2pix model was originally pretrained using RGB images. In this training phase, the images were labelled using the HSV (Hue, Saturation, Value) colour space, where the Hue component

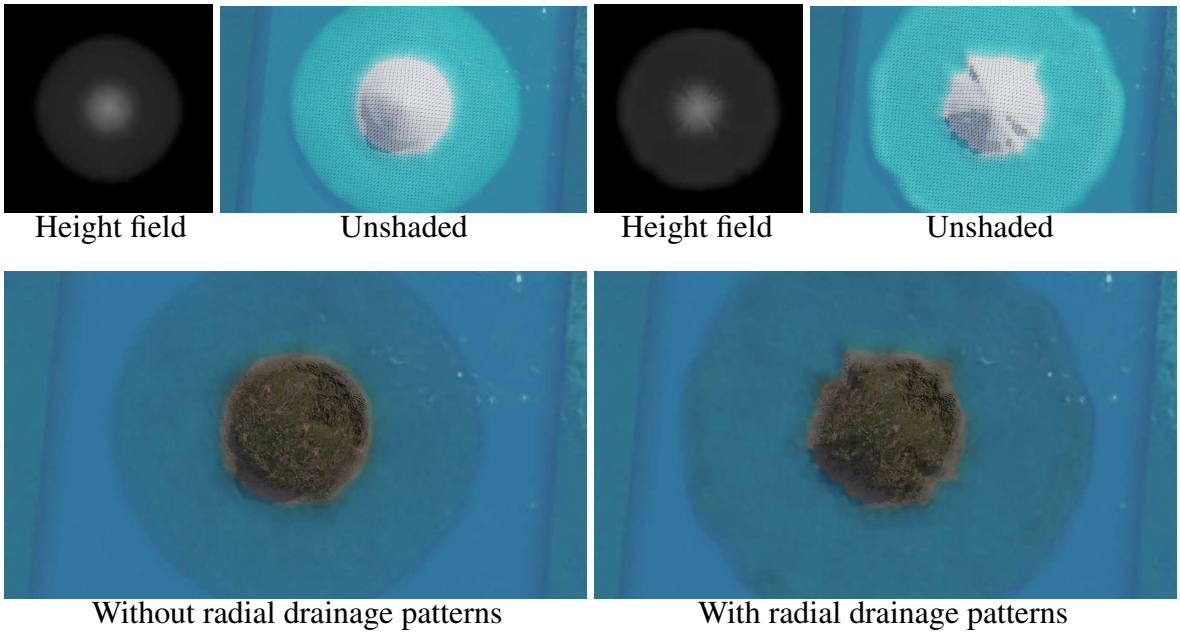



**Figure 3.38:** *Multiple procedural strokes are added between the island centre and the surrounding sea, alternately landward and seaward. This transforms a simple circular island (left) into a more realistic volcanic island (right) by cheaply simulating star-shaped radial drainage patterns.*

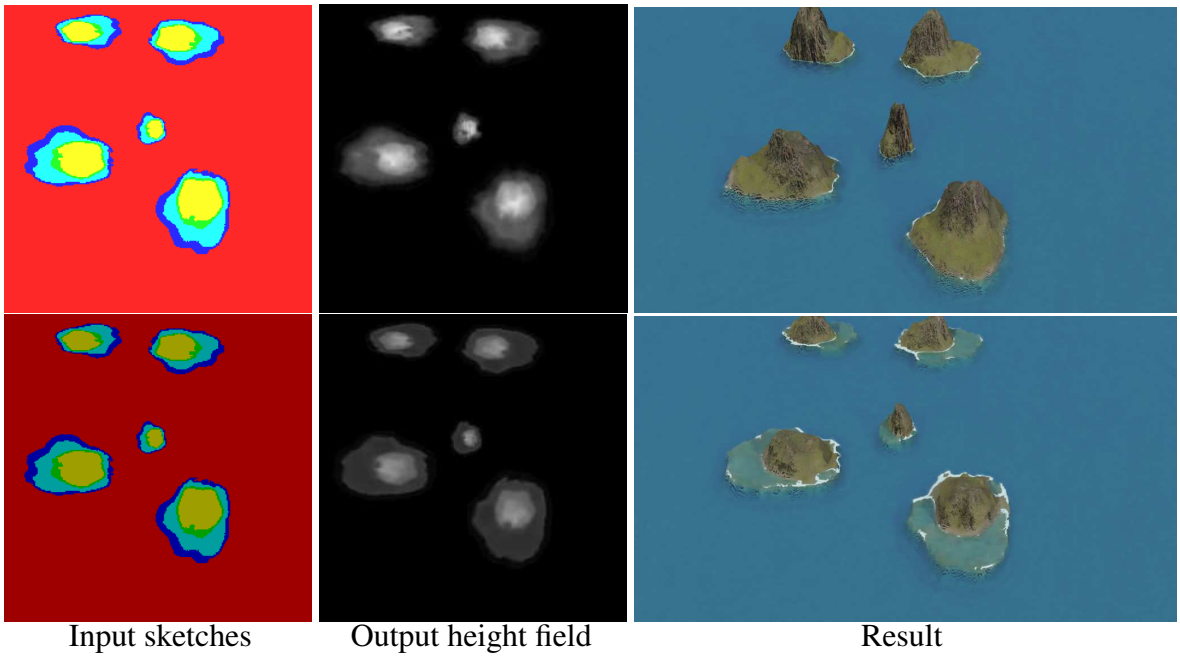



**Figure 3.39:** *An identical label map yields similar height fields over multiple inferences from the model, even after modifying the subsidence factor (visible in the luminosity of the input image).*

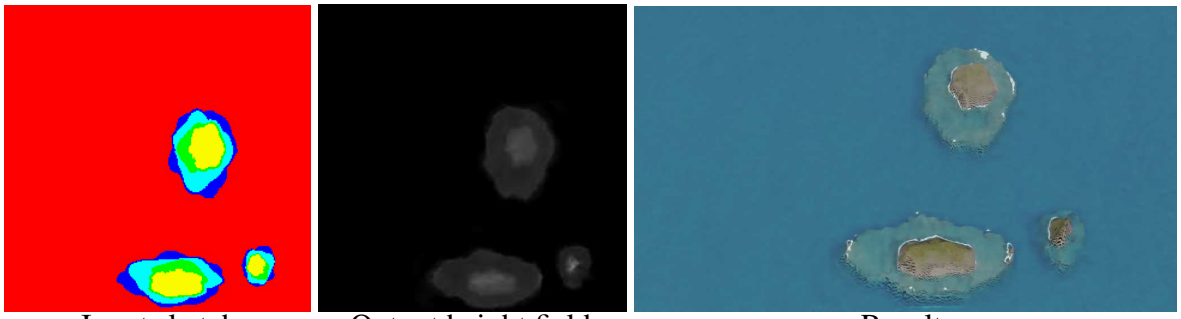
specifically carried the label information. Both the Saturation and Value components were kept neutral, meaning they did not convey any significant label-related data. The target images, the ones the model aimed to reproduce, were formatted in RGB.

For the purpose of fine-tuning the model, we continue to use the Hue component to encode the labels from the label map. We introduced a new dimension to the model's learning capabilities by incorporating the subsidence rate, denoted as $\lambda$, into the Value component. This addition not only utilises the model's existing capability to interpret the HSV format but also enriches the input data, which now carries additional, valuable environmental information.

Moreover, we purposefully left the Saturation component unchanged at this stage, reserving space for potentially including another parameter in the future, which would allow us to expand the model's utility without altering the foundational HSV encoding scheme established during its initial training. By adhering to this encoding format, we ensure continuity in data representation, which maximises the efficiency of the pretrained model. This strategic update enhances the model's adaptability and broadens its applicability to tackle new, complex challenges more effectively.

This configuration enables rapid generation of large quantity of samples, with multiple parameters, of a single island centred in the image.

### 3.5.1 Data augmentation



Input sketch Output height field Result

**Figure 3.40:** *A sample of the dataset with the input and output of the island copied and scaled at multiple positions of the image, creating three different islands in a single image.*

To enhance the variety of the dataset and improve the model's ability to generalise, we apply several data augmentation techniques in addition to the usual affine transformations (rotation, scaling, and flipping):

- **Translation:** Since the original algorithm always centres the island, we translate the islands within the image to remove this constraint (Figure 3.40). This ensures that the cGAN can generate islands in any position within the frame. By wrapping the island on the borders (see example Figure 3.41), the model learns that data can appear on the edges of the image.

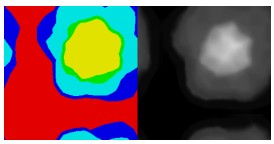


**Figure 3.41:** *As we translate, we wrap the height field and the label map, reducing border artefacts appearing when datasets rarely contain content on edges.*

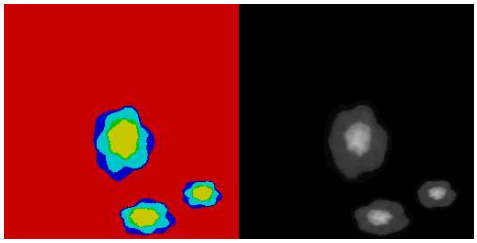


**Figure 3.42:** *We enforce the possibility to have multiple islands at once.*

• **Copy-paste:** In some cases, we combine multiple islands into a single sample (Figure 3.42), with only a maximum intersection over union (IoU) of 10%, allowing the abysses to overlap but without obtaining other region types to merge. Height fields are blend using the smooth maximum function from Equation (3.8), and label blending uses the usual max operator. Regions not covered by any island are assigned the abyss ID, which correspond to the minimal height. In this case specifically, the subsidence factor (Value channel) is close to irrelevant (Figure 3.40, rightmost).

All augmentation techniques are simultaneously applied to both the height field and the label map ensuring consistency between input (the label map) and output (the height field). We summarise the complete dataset generation in Algorithm 3.

**Input:** Base outline radii $r^*$, profile $h^*_{\text{profile}}$, resistance $\rho^*$, noise $\eta$, stroke generator, subsidence grid $\{\lambda\}$
**Output:** Pairs $\{(\text{label map},\ \text{height field})\}$

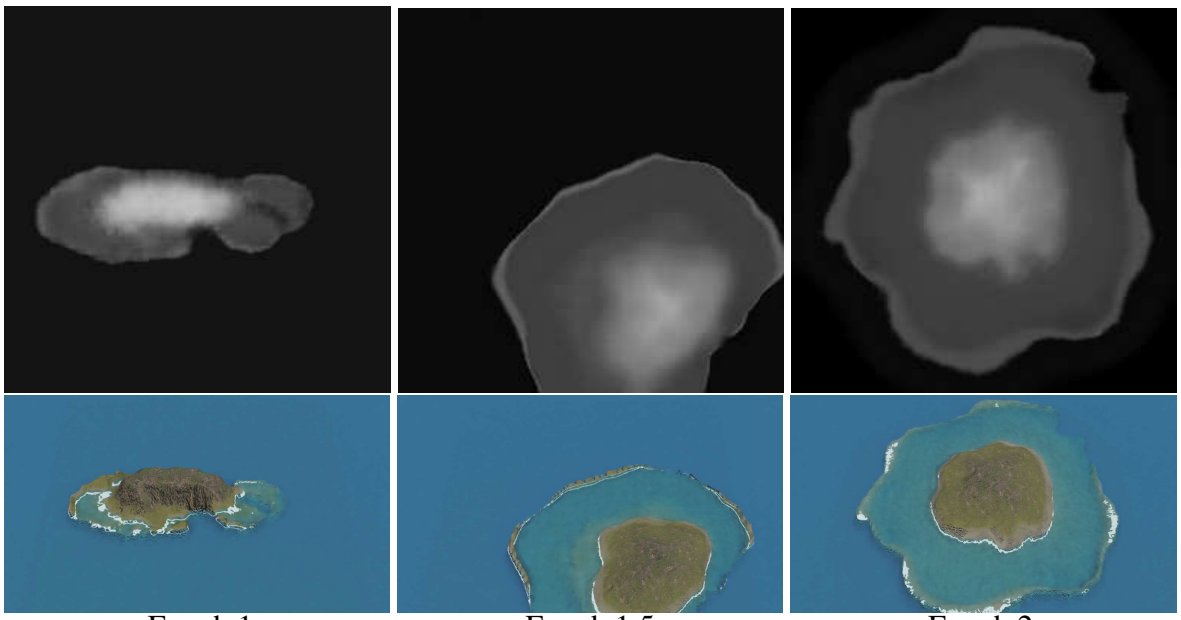
`// (1) Randomise sketches`
**foreach** *outline* **do**
  $r(\theta) \leftarrow r^* + \eta(\theta)$
$h_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \leftarrow h^*_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}})$
$\rho(\tilde{x}_{\mathbf{p}}) \leftarrow \rho^*(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}})$
`// (2) Random strokes`
Generate $n$ strokes $\{C\}$ by uniformly sampling $m$ control points each; sample $\sigma$ and $S_C$ per stroke
`// (3) Synthesis over subsidence levels`
**foreach** $\lambda \in [0,1]$ **do**
  Build $h$ and $I$ via Algorithm 1 and apply Algorithm 2 to get $\hat{h}$
  label map $\leftarrow$ HSV image with $I$ in the Hue channel and $\lambda$ in the Value channel
  Store pair $(\text{label map}, \hat{h})$
`// (4) Augmentation`
Apply translations with wrapping, rotations, scalings, flips to both label map and height field consistently
Copy-paste multiple islands with IoU$\leq$ 0.1; blend heights with smax and labels with max; and assign abyss ID for uncovered regions
**return** dataset

**Algorithm 3:** Procedural dataset generation.

## 3.6 Learning-based generation

In this section, we introduce the use of a cGAN, specifically the pix2pix model, to enhance the island generation process by increasing the variety and flexibility of terrains. While the initial procedural algorithm generates diverse island examples (convex and concave outlines), cGAN provides additional flexibility in generating more complex terrain without the rigid constraints of the procedural algorithm that stem from our initial assumptions based on coral reef formation theory.

Our model is trained using a batch size of 1, and optimised using the Adam optimiser with a learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The loss function combined a conditional adversarial loss and an L1 loss, where the L1 loss was weighted by a factor $\lambda = 100$. The generator follows a U-Net architecture with encoder-decoder layers and skip connections, while



Epoch 1 (~10 000 samples) | Epoch 1.5 (~15 000 samples) | Epoch 2 (~20 000 samples)

**Figure 3.43:** *After the first epoch (left), grid artefacts similar to a low-resolution image are visible, but are being corrected during a second epoch (middle) where erosion patterns appear in the height fields (right).*

the discriminator was based on a PatchGAN, classifying local image patches. These parameters are directly adopted from the original pix2pix paper [IZZE17]. We use a dataset of 1 000 single islands, augmented with the various methods seen in Section 3.5.1 to obtain about 10 000 samples that are all seen once during an epoch (representing about 30 minutes of training).

During inference after a single epoch, our model still outputs grid artefacts, visible in Figure 3.43. After the second epoch, visual artefacts are already rare. Illustrations displayed in the remainder of this chapter are results of models trained over less than ten epochs. The training time is relatively short compared to typical deep-learning terrain generation pipelines, largely thanks to the synthetic nature and consistency of our dataset.

Once the model is trained, the user colours a $256 \times 256$ RGB image to sketch the different regions. Each region is given a specific colour based on the HSV encoding used for the dataset generation. The examples presented in this paper use red for abysses, blue for reefs, cyan for lagoons, green for beaches, and yellow for mountains. The subsidence factor presented in Section 3.4.2 is controllable through the luminosity of the input image.

The Python script for the initial island dataset generation is unoptimised and takes about 2.5 s per island of size $256 \times 256$ as we do not perform parallelisation here. Implementing an optimised C++ version of the initial generation process reduces this execution time to 50 ms per generation.

On the other hand, the inference time of our deep learning model for a single input image of dimension $256 \times 256$ remains constant regardless of scene complexity. Using the NVIDIA GeForce GTX 1650 Ti GPU with Python 3.10 and PyTorch version 2.5.1+cu121, the inference time measured is 5 ms (std 1.1 ms).

Once trained, the cGAN model generates a height field from a given label map. The output is a complete 2D height map representing the terrain's elevation at each point. Although the cGAN's internal logic is not easily interpretable, the label map remains available after inference and retains semantic terrain structure for the user.


**Figure 3.44:** *User applied fuzzy brushes to draw the label map, resulting in some pixels that are inconsistent with the dataset and illogical island layouts: abyss regions (red) are found between beach (green), lagoon (cyan) and reef regions (blue). The neural model ignores the layout inconsistencies, and partial over-brightness as long as the pixel is not completely white.*

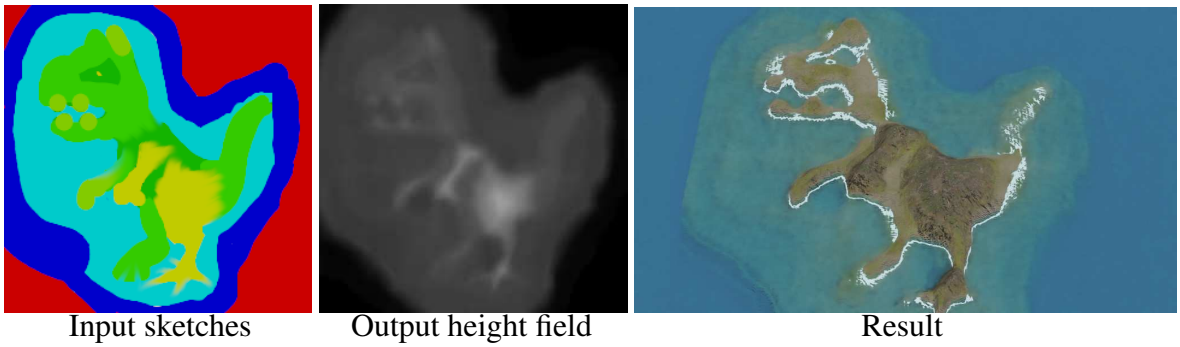



**Figure 3.45:** *Without constraints on the generation, the user may use unrealistic layout and the neural network will however output a plausible result. Here new colours have been added (pistachio), but the network naturally considers it as a transition from mountain to beach.*

## 3.7 Results

The resulting model for coral reef island generation enables a high level of user control as unconstrained painting allows complex scenarios while producing height fields in real-time. In this paper we used Blender to provide renders directly from the output height fields. As our pix2pix model is trained to output $256 \times 256$ images, the resolution of the 3D models is limited by this architecture.

By using deep-learning-based models, most initial constraints are removed (radial layout, isolated islands, ...). The control over the overall shapes of the island regions is given through digital painting with any software, here using GIMP (first column of Figure 3.44) and Paint (such as in Figure 3.45). Each pixel of the image is encoded in HSV, with the region identifier encoded in the Hue channel. The user may increase or decrease the subsidence level of the island by modifying the Value channel over the whole image (see Figure 3.39).

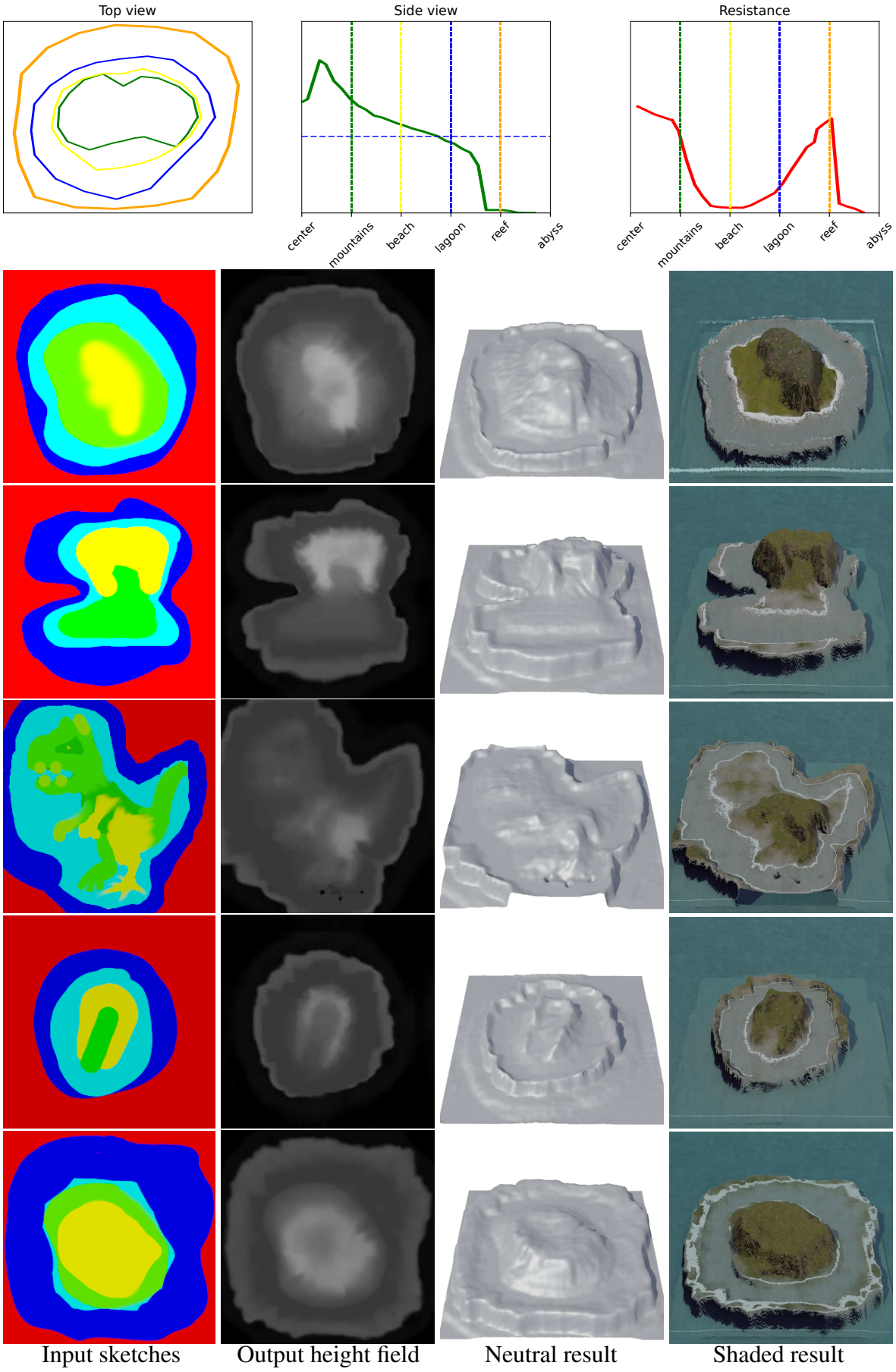



**Figure 3.46:** *Top row: user input used to generate the "volcanic" dataset. Bottom: User sketches creates various islands, following the initial user inputs.*

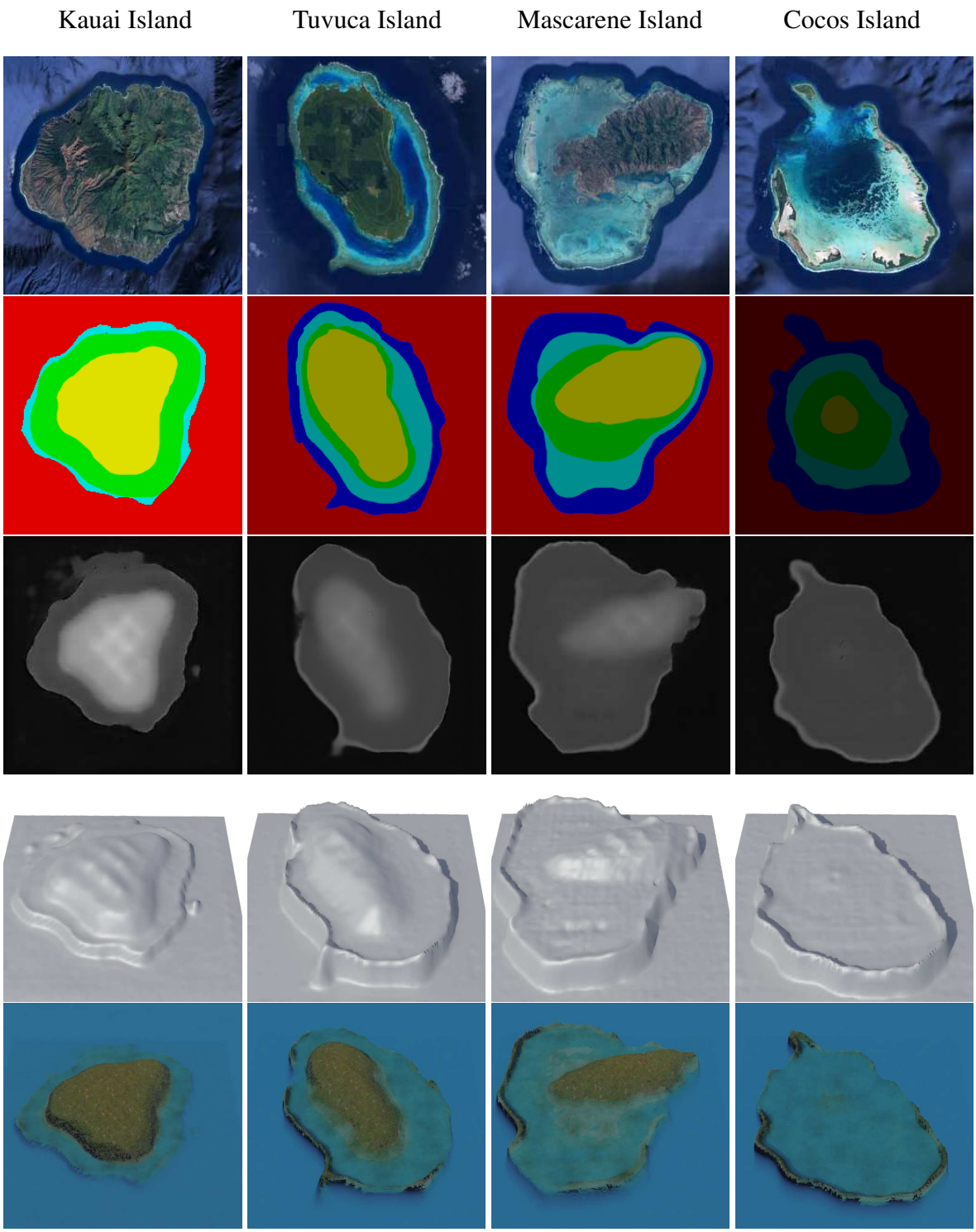



**Figure 3.47:** *Reproduction of islands from Figure 3.2.*

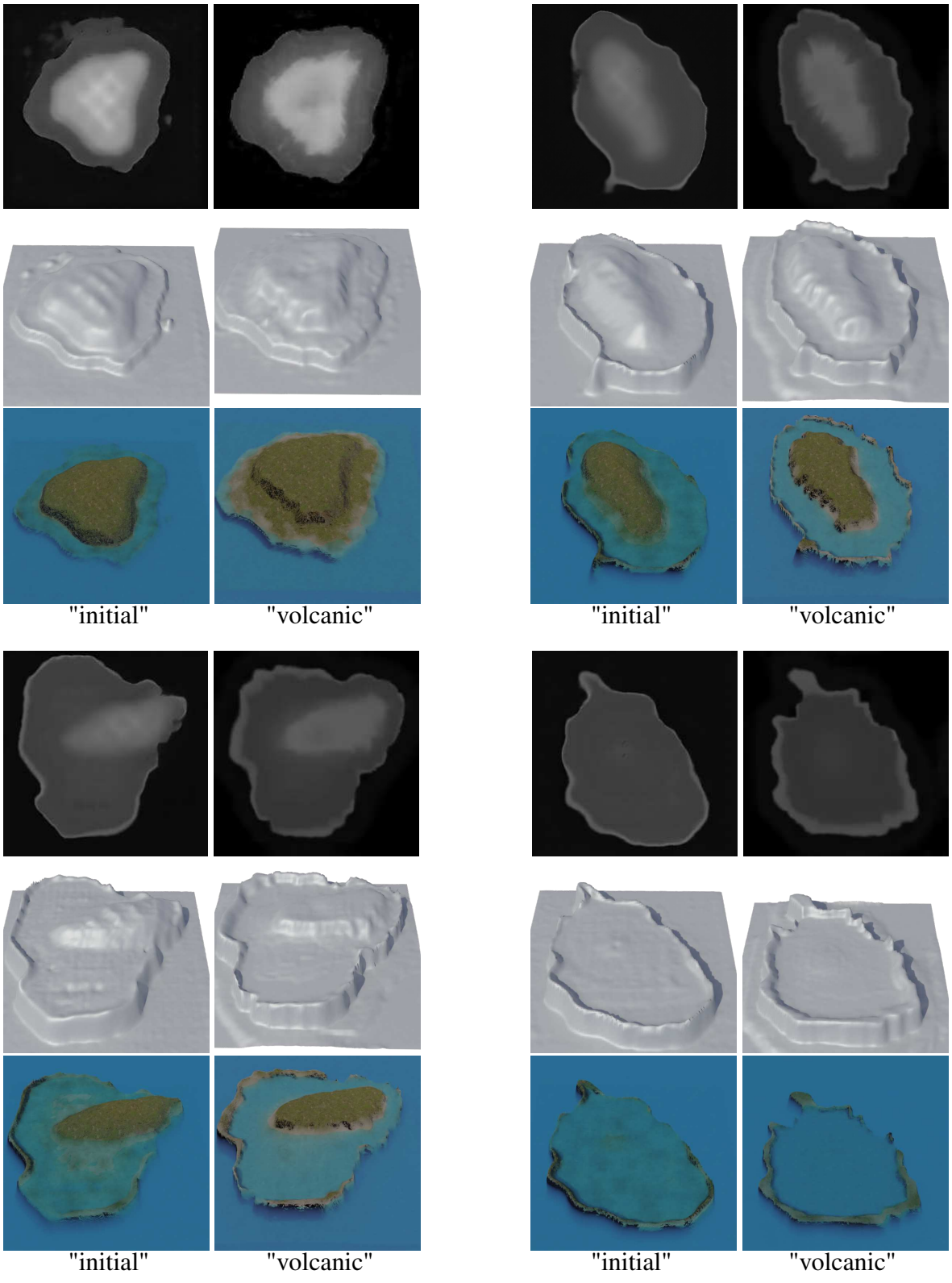



**Figure 3.48:** *Comparison between the "initial" cGAN trained model (dataset generated from parameters visible in Figure 3.33) and the "volcanic" cGAN (Figure 3.46, top). The input label maps are identical as Figure 3.47.*

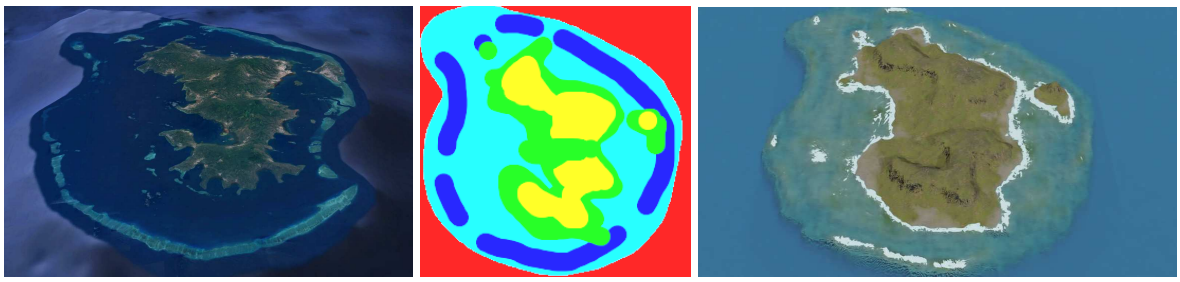


**Figure 3.49:** *Comparison between real (left, from Google Earth) and synthetic (right) islands of Mayotte. The label map (centre) is hand-drawn to match approximately its real-world counterpart.*

Since the model relies on pixel-level statistics rather than strict class values, it is robust to noise caused by compression, anti-aliasing from brush tools, or resizing artefacts introduced by image editors. The example displayed in Figure 3.44 (top) displays a sketch with blurry outlines and unexpected layouts (see the small red regions inside the southern lagoon region or the adjacency of beach regions directly with the abyssal region). The learned model does not include inconsistencies and results in plausible 3D models. We can note that the input label map is not required to be hand-drawn, as a simple Perlin noise can produce it randomly, as shown in the examples of Figure 3.50. Figure 3.44 (bottom) shows highlight clipping (white pixels due to artificial oversaturation, losing the Hue value) that is not interpretable by the model and resulting in black patches in the result.

The tolerance to imprecise input values enables more control about the transitions between two regions than the procedural module. Figure 3.45 shows an example of input map with regions that are leaking over neighbouring regions, and the introduction of new hue values nonexistent in the dataset (light green and dark green) but are the interpolated hue values of mountain regions and beach regions.

Because our curve-based procedural phase included low randomness, the output of the cGAN is limiting its unpredictability, meaning that small input changes result only in minor changes on the output, preventing unexpected results. Figure 3.39 shows the result of an input map with only a variation on the subsidence level, the resulting height fields are very similar. Adding the real-time computation of outputs, it becomes possible to construct progressively a landscape and correct small inconsistencies to intuitively design islands inspired by real-world regions. A creation of an island similar to Mayotte, presented in Figure 3.49 was hand-made in just a few minutes. The islands presented at the beginning of the chapter in Figure 3.2 are also reproduced in Figure 3.47 with different subsidence rates.

Figure 3.46 (bottom) presents results of a second cGAN training for which the procedural parameters used for the generation of the training dataset "volcanic" are slightly adjusted for lower overall resistance and to have a crater in the center of islands (Figure 3.46, top). The $h_{\text{profile}}$ function include a hole at the center of the island to emulate a volcaninc crater, which we can see strongly in the first and last examples, and more slightly in the second example; the resistance function $\rho$ is lowered on the reefs, creating visible dents in the reef and dendritic patterns on the island sides.

### 3.7.1 Advantages of the approach

One of the main strengths of this approach is its ability to produce a wide variety of island terrains, even in the absence of real-world data. The procedural generation methods allow for high flexibility in designing both the shape and features of the island, while the use of cGAN enables further refinement and the generation of terrains not bound by the original constraints of the procedural model. By combining these two methods, we leverage the complementary advantages of both: the structured control of procedural techniques and the pattern-learning capabilities of deep learning.

A key advantage of this approach is the preservation of semantic information throughout the generation process. The label map, which serves as the input to the cGAN, can also be used after terrain generation

to provide a detailed semantic representation of the different regions of the island (island, beach, lagoon, coral reef, and island body). This label map can be useful to guide post-processing operations, such as applying different textures based on terrain features or adding other environmental elements, vegetation for instance. The preservation of semantic information provides a useful connection to the next stage of terrain manipulation, making the process more versatile and adaptable to different use cases.

Furthermore, the use of an out-of-the-box cGAN model highlights the feasibility of employing existing neural network architectures with minimal modifications in the field of procedural generation. This is particularly important for domains where real-world data is scarce, such as coral reef islands, allowing synthetic data to be effectively used for training purposes.

### 3.7.2 Limitations

While the cGAN model provides increased flexibility and variety in island generation, it does come with certain limitations:

- **Data-driven dependence:** Just as any deep learning method, the quality of generated terrains strongly depends on the quality of the dataset. Unusual layouts or out-of-distribution inputs are not entirely interpretable by the models. Our dataset generation method lifts some restrictions, but some remain such as the required presence of all regions in the input map. This limitation reduces the true diversity of the generated terrains, as the cGAN cannot generate terrains that deviate too far from the examples in the training set.
- **Biases from the synthetic dataset:** Since the cGAN model is trained entirely on a procedurally generated dataset, it inherits the biases present in the initial algorithm. For example, while the model breaks free from the radial symmetry constraint and centre positioning, it remains dependant on the structure and patterns of the synthetic data (e.g., our procedural drainage patterns, our reef analytic morphology, etc.). The "realistic" aspect of the results thus depends on the generation methods used to create the dataset.
- **Lack of user control:** Another limitation of using cGAN in this context is the lack of real-time user control during terrain generation. While traditional procedural generation methods allow users to adjust parameters during the generation process, the cGAN model operation is abstracted from the user, providing no mechanism for direct interaction beyond the initial label map. This reduces the level of customisation available to the user.

## 3.8 Conclusion and future works

In this work we presented a novel approach to generating coral reef island terrains by combining traditional procedural methods with deep learning techniques. We first developed a procedural generation algorithm capable of creating a wide variety of island terrains through a combination of top-view and profile-view sketches, wind deformation, and subsidence and coral reef growth modelling. By applying these methods, we produced plausible terrains based on geological processes, capturing key features of coral reef islands: island, beaches, lagoons, and coral reefs.

To further enhance flexibility and realism in the generation process, we incorporated a Conditional Generative Adversarial Network, using the pix2pix model to generate height maps from label maps of island features. The cGAN model allowed us to overcome some of the constraints inherent in the procedural algorithm, such as radial symmetry and fixed island positioning. Through data augmentation, we trained the cGAN on a synthetic dataset, enabling the generation of varied and plausible island terrains.

There are several directions for future research and improvements. One promising avenue is to incorporate the wind velocity field directly into the cGAN training process, potentially as an additional

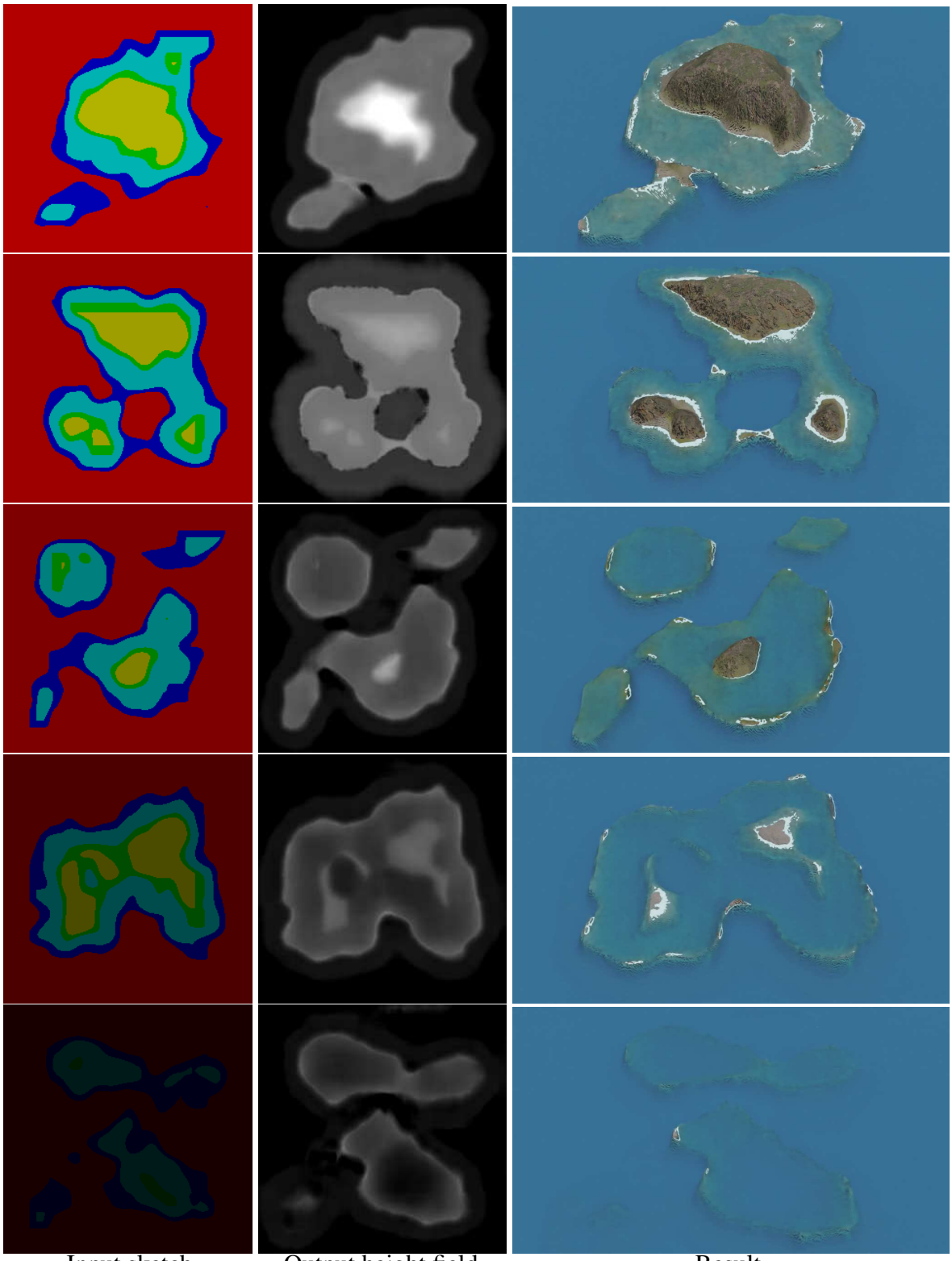



**Figure 3.50:** *Starting from random Perlin noise, transformed into a label map, we can generate a large variety of results.*

input condition. This would allow the model to better capture wind-driven terrain features such as cliffs or other deformations influenced by wind patterns.

Another area for exploration is improving user interaction during the terrain generation process. While the current model allows for rapid terrain generation, adding more options for users to interact with the cGAN, such as adjusting parameters (wind strength and main direction, erosion, reef function parameters, ...), could improve control over the terrain generation process of our system and help the user to generate his target results.

Finally, further improvements could be made to the synthetic dataset. Incorporating additional geological processes, such as higher fidelity physical simulation of wave and rain erosion, and tidal influences, could lead to even more realistic terrains at the cost of interactivity. Additionally, refining the way islands are blended in multi-island samples, or adding more diverse input conditions (e.g., different geological settings), could help the model generalise and produce more varied and dynamic landscapes.

Various other neural network models could be exploited to increase user interactions and improve user control, with newer variants of cGANs [PLWZ19], or models with style transfer functionalities [GEB15, ZCZ*20] in order to change the overall aspect of a terrain [PPB*23, PPB*23], use text-to-image models [RBL*21, RKH*21] to generate height fields from a verbal prompt, or super-resolution models [DLHT14] to increase the definition of details in the final output [GDGP16].

In summary, this chapter has demonstrated how procedural rules and deep learning can produce convincing island terrains. Yet, terrains alone are not enough: seascapes also depend on the biotic and abiotic features that inhabit them. The following chapter therefore moves beyond geometry to explore ecosystem generation.

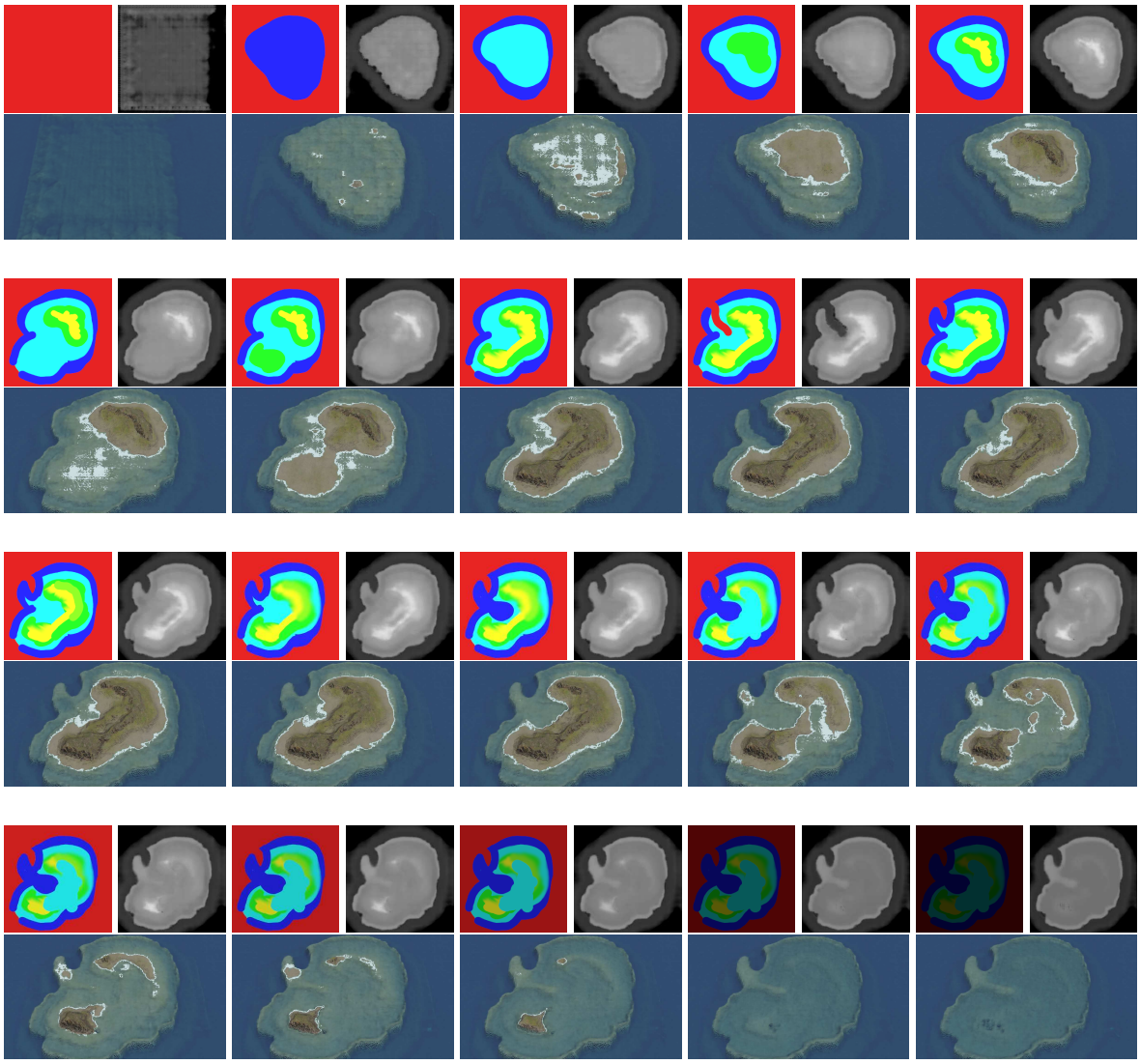


**Figure 3.51:** *Few frames of an editing session for the generation of a coral reef island on Paint. The resulting height fields are output in real-time and each generation is consistant, allowing to interact fluently with the system without unexpected behaviour. The shaded outputs (bottom of each frame) are rendered in Blender, which are not real-time.*

Chapter 4

# Semantic terrain representation

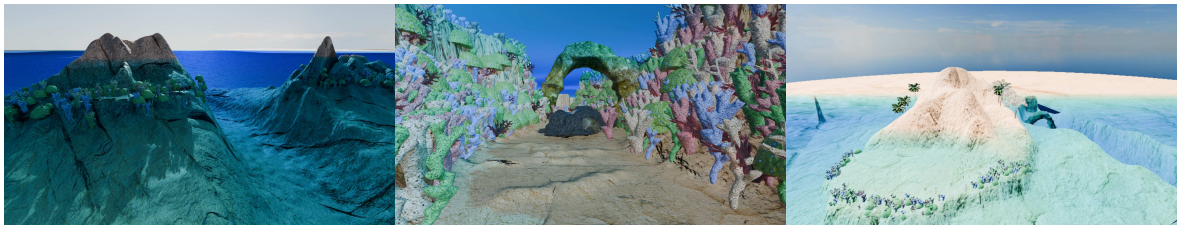


**Figure 4.1:** *Three underwater scenes using our environmental objects representation. From left to right: an island cut in half by a canyon, the inside of a canyon with rocks and corals, an island with corals, algae and a trench.*

## Abstract

This chapter introduces a novel method for procedural terrain generation, which leverages a sparse representation of environmental features to produce landscapes that are lightweight, plausible and adaptable to user desires. The method differs from traditional terrain generation approaches by emphasising multi-scale user interaction and incorporating expert knowledge to model the evolution of terrain features over time. By representing terrain features as discrete entities, or "environmental objects", our method enables dynamic interaction between these entities and their surrounding environment, represented through continuous scalar and vector fields. The generation process is iterative and allows for user-guided modifications at each step, including the introduction of environmental events that influence terrain evolution. The proposed approach is particularly flexible, capable of generating underwater landscapes with a focus on large-scale plausibility and detailed, localised feature representation.

# Contents



# 4.1 Introduction

While the previous chapter focused on generating the large-scale geomorphology of coral reef islands through sketch-based procedural rules and learning-based generation, these terrains remain purely geometric. In practice, convincing seascapes require not only landforms but also the biotic and abiotic elements that populate them (coral colonies, algae, sediments, rivers, and other ecological features that interact across scales). Addressing this broader challenge calls for a representation that integrates semantic structure, ecological constraints, and user control. To address this challenge, we turn to ecosystem generation in this chapter, introducing a semantic framework that unifies terrain and ecology through symbolic primitives and environmental fields.

Procedural ecosystem generation in computer graphics remains fragmented. Classical terrain algorithms produce heightfields or meshes without semantic structure, plant-placement systems use fixed-radius or zone-of-influence rules for foliage alone, and full ecosystem simulators, although rich in biotic and abiotic processes, are too costly for real-time or interactive content creation. Consequently, no prior work combines sparse, semantic primitives for both living and non-living features, multi-material field coupling driven to steady state, and multi-scale, user-interactive authoring into a single, lightweight ecosystem generator.

To fill this gap, we introduce a pipeline that constructs both biotic (corals, algae, trees) and abiotic (islands, rivers, canyons) elements by instantiating environmental objects (points, curves, regions) which read from and write to continuous environmental fields (elevation, water flow, resource stocks) in an iterative, steady-state diffusion loop. At each step, environmental objects candidates spawn or die via expert-tuned generation and fitness rules, and users can inject global geomorphological events or

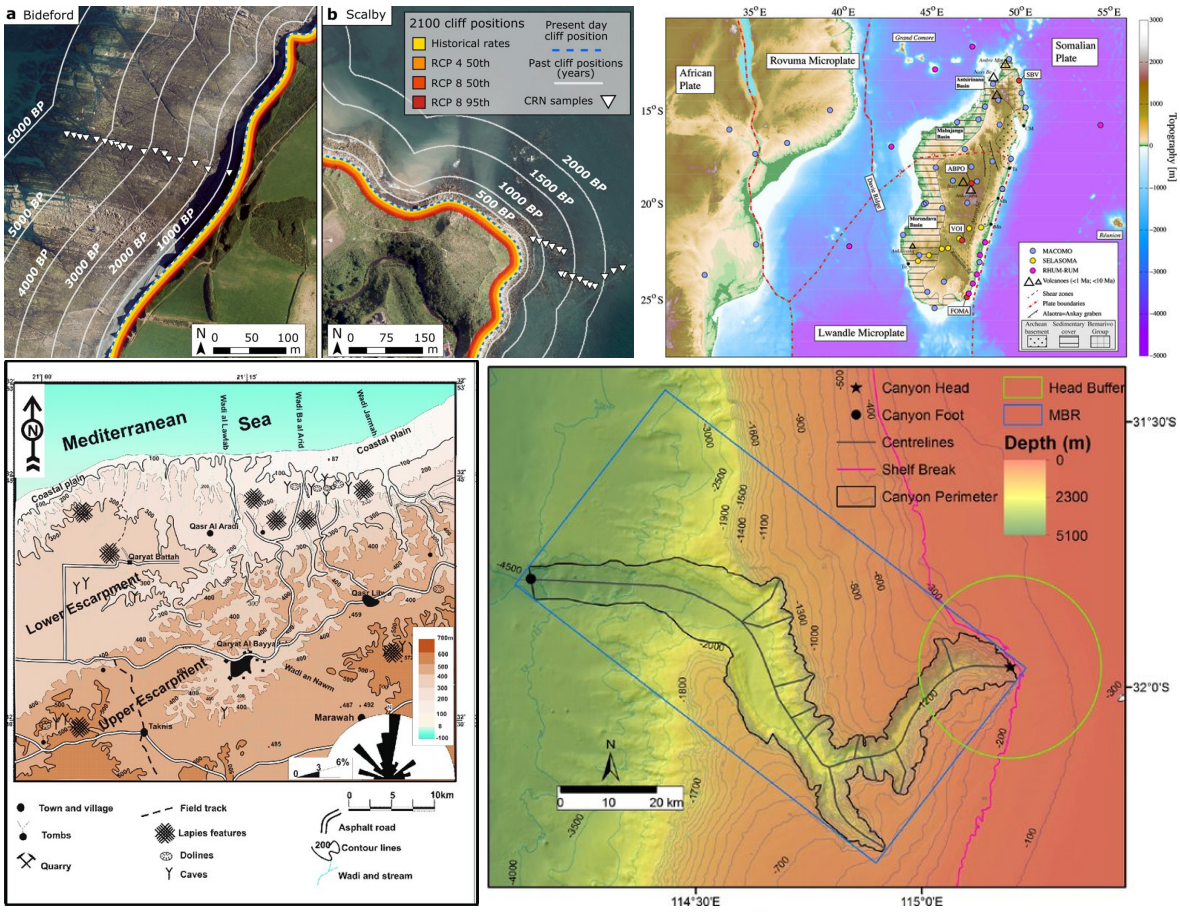



**Figure 4.2:** *Examples of cartographies used in different fields of natural science. From left to right, and top to bottom: Evolution of coastlines at Bideford, UK and Scalby, UK over the last 6000 years and 2000 years respectively (BP = Before Present) [SRH*22]; sedimentary distribution over Madagascar island [PWA*17]; geological features distribution of karstic landscape at Qasr Libya, Libya [AM09]; localisation of key parameters of Perth Submarine Canyon, Australia [HNHC14].*

manually refine any primitive, all while maintaining interactive performance for thousands of objects.

We draw qualitative inspiration from cartographic symbology, where topographic and bathymetric charts employ 0D points (volcanoes, observatories), 1D lines (canyon heads, reef crests), and 2D regions (soil or sediment covers) to strip away geometric complexity and preserve the spatial relationships that underpin expert reasoning (see Figure 4.2). Similarly to these map primitives, our environmental object abstractions enable a "sketch-first, refine-later" workflow: designers begin with a coarse semantic layout, then progressively zoom in, from mountain ridges down to individual boulders or coral colonies, without ever losing global coherence.

In the following state of the art, we first review classical radius-based heuristics used for ecosystem generation in Computer graphics (FRN, ZOI), Ecological-Field Theories (EFT) and reaction-diffusion equations that inform our local interaction and diffusion mechanisms. We then present our environmental objects-environmental attributes pipeline in detail and demonstrate its use for underwater coral-reef environments.

Unlike full-fledged ecosystem simulations, which model temporal dynamics (population growth, predator-prey interactions, nutrient cycling, and fluid flows, ...) over successive time steps to study emergent behavior, our work centers on ecosystem generation, where the goal is to produce a plausible, editable snapshot of a landscape by iteratively placing and culling semantic primitives (environmental objects) that read from and write to steady-state environmental fields; this generation process trades off dynamic fidelity for interactive performance and multi-scale user control, delivering plausible biotic and abiotic scene layouts without the overhead of continuous process simulation.

Building on these observations, the contributions presented in this chapter are:

- a sparse, domain-agnostic symbol set (points, curves, regions) that both read continuous environmental fields and write local modifications back to them, creating a closed feedback loop,
- a decay-stabilised reaction-diffusion solver for scalar "materials" and analytical deformation of the vector-flow field yield interactive, steady-state updates without heavy fluid or erosion simulation,
- a focus-and-refine pipeline lets users jump from kilometre-scale planning to sub-metre edits while preserving global coherence.

## 4.2 State of the art

Simulating ecosystems involves a range of modelling approaches, each balancing biological realism and computational efficiency. Broadly, biological models treat populations as continuous fields using reaction-diffusion equations, or model each organism individually through agent-based methods. For clarity, we classify existing work in three categories: site-based (grid or tessellation) models, Individual-Based Models (IBMs), and continuous field models, each emphasising a different balance between local level details and scalability. Site-based models discretise space to model local interactions efficiently. In contrast, IBMs capture fine-scale behaviours and heterogeneity, often using spatial interaction rules (fixed-radius, zone-of-influence, or field of neighborhood methods). Ecological Field Models (EFMs) bridge these paradigms by representing environmental properties as continuous fields influenced by organisms, enabling dynamic feedback between species and their surroundings. These diverse frameworks provide flexible tools for capturing complex ecological processes at varying scales and resolutions.

Two main modelling paradigms emerge in the literature: modelling dense populations as continuous media [Tur52], or representing each individual explicitly [Cza98]. The former excels in broad predictions; the latter in local realism.

From a large-scale perspective, where billions of individuals are indistinguishable (e.g., grass shoots, bacteria, whole populations), we treat species as continua and borrow reaction-diffusion models from physics and chemistry.

Conversely, when the number of individuals is smaller or when populations are sparse enough that individuals and their interactions must be tracked explicitly, IBMs become appropriate. These methods treat organisms as agents that sense and interact with neighbours, making decisions (spawning, growing, and dying for vegetation).

A hybrid family, the EFMs [WSWP85], describes how each individual modifies the state of its surrounding environment. Borrowing field theory from physics, they let organisms deposit or absorb continuous "influence fields", creating explicit feedback loops with the environment.

Below, we review the three modelling families (site-based, individual-based, field-based) because our method borrows the sparse symbolic control of IBMs yet retains the continuous feedback of EFMs. We then discuss diffusion models, the mathematical backbone for our environmental fields.

### 4.2.1 Site-based models

Site-based models represent the environment as discrete regions with uniform properties, interacting primarily with their immediate neighbours (Figure 4.3). The concept generalises to other tessellations such as Voronoi diagrams or arbitrary graphs.

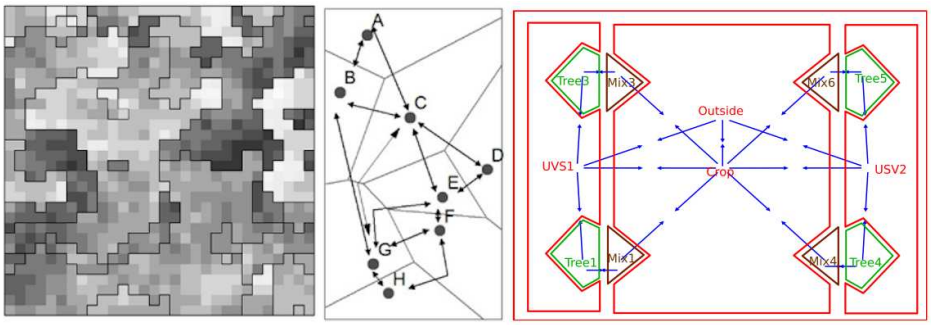



**Figure 4.3:** *Site-based models partition the environment into discrete regions with locally uniform properties. Examples include (left) regular grids, (middle) Voronoi cells, and (right) generalised topological maps [NR12, LJGS23].*

### Grid-based models

Grid-based models represent ecosystems by discretising continuous space into a regular arrangement of cells, typically squares or rectangles, each containing uniform environmental properties and ecological conditions (Figure 4.3, left). In this approach, each cell represents a spatially explicit patch hosting one or multiple populations, with interactions primarily occurring between neighbouring cells. The central ecological rationale for employing grid-based models is their conceptual simplicity, computational efficiency, and ease of representing spatial processes such as diffusion, dispersal, competition, or predation [GR05, CCR10, NR12].

Mathematically, grid-based ecological dynamics are commonly described using discrete-time state equations that represent how the state of a cell (e.g., population density, biomass, or nutrient availability) changes according to its local interactions. A general formulation of these dynamics are expressed as:

$$N_{t+1}(x,y) = f(N_t(x,y), \overline{N_t}(x,y), E_t(x,y)),$$

where $N_t(x,y)$ is the state variable at cell coordinates $(x,y)$ at time $t$, $\overline{N_t}(x,y)$ is the neighbourhood of the cell at coordinates $(x,y)$, $E_t(x,y)$ is the explicit environmental factors (e.g., precipitation and soil nutrients), and $f$ is the update function that describes the biological changes given the current state, the neighbourhood and the environmental factors around a cell.

Grid-based models are particularly suited for simulations involving diffusion-like processes or phenomena with clear spatial boundaries. Classic examples in ecology include modelling vegetation dynamics in arid ecosystems, spatially explicit predator-prey interactions, forest-fire spread, and habitat fragmentation effects. Additionally, grid-based representations form the basis for Cellular Automata models, where cells exhibit discrete states (e.g., occupied, empty, burning), updated synchronously according to predefined state-transition rules.

Grid-based models are commonly used for their computational simplicity, easy integration with Geographic Information Systems (GIS), and suitability for parallel computing [Cza98, NR12]. They facilitate large-scale ecological modelling by clearly defining spatial relationships and boundary conditions, thus making them practical for exploring broad-scale ecological hypotheses or management scenarios.

However, significant limitations exist, as ecological processes may differ drastically at varying scales (e.g., local vs. landscape-level processes). Incorrect spatial resolution of the grid cells can lead to ecological inaccuracies or oversimplifications [Wie89]. Moreover, grid cells assume ecological homogeneity within each cell, which may poorly reflect natural ecological heterogeneity, especially in large cells [Lev92]. Complex ecological interactions, such as asymmetric competition or directional effects, are typically simplified or lost in grid-based discretisation [DL94]. Finally, regular grids impose artificial directional biases (e.g., horizontal/vertical) that impact ecological interpretations,

especially for processes that depend on continuous spatial gradients or isotropy.

### Tessellation models

Tessellation models generalise the idea of grid-based methods by partitioning continuous ecological space into discrete regions based on proximity rather than regular shapes (Figure 4.3, centre). Among these methods, Voronoi diagrams (also known as Thiessen polygons in ecological studies) and their dual, Delaunay triangulations, are particularly relevant and widely applied [LJGS23, Fol19, MU08].

Voronoi diagrams partition a plane based on a set of points (e.g., tree positions, animal territories, or resource centres). Each region in a Voronoi diagram is defined as the set of all points closer to a specific focal point (seed) than to any other point. Formally, given a set of points $p_1, p_2, ..., p_n$, the Voronoi region $V_i$ associated with the point $p_i$ is defined as:

$$V_i = x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j), \forall j \neq i,$$

with $d(x, p_i)$ the distance between the point $x$ and the seed $p_i$.

Conversely, the Delaunay triangulation is the dual graph structure of a Voronoi diagram. It connects points whose Voronoi regions share a common boundary, thereby explicitly defining neighbourhoods among entities (e.g., individuals or resource patches). This duality between Voronoi and Delaunay representations provides two complementary perspectives: territorial partitioning (Voronoi) and direct adjacency or connectivity (Delaunay).

In ecological modelling, these tessellation methods have significant implications and applications. Voronoi diagrams naturally represent resource partitioning among individuals or populations, capturing territorial or competitive interactions without the artificial constraints imposed by fixed grids [CC06]. For instance, they have been effectively employed in forest ecology to model individual-tree resource competition, where the area of a tree's Voronoi cell directly correlates to available resources and competitive interactions.

The use of tessellated models may become very useful as they adapt to irregular spatial distributions, accurately modelling ecological territories and resource accessibility based on actual entity locations. These models inherently define spatial neighbourhoods without arbitrary distances or predefined shapes, enabling more precise representation of interactions among entities.

Unlike grids, Voronoi cells automatically adjust spatial resolution according to entity density, allowing finer resolution in densely populated areas and coarser resolution in sparse ones.

However, Voronoi-Delaunay models also present several limitations and challenges:

- updating dynamically Voronoi diagrams and Delaunay triangulations is computationally more intensive than fixed-grid models, especially in large-scale or highly dynamic ecosystems where individuals frequently appear, disappear, or move.
- Voronoi models implicitly assume isotropic interactions (competition or resource usage is equal in all directions) and strictly distance-based territories, neglecting directional biases such as prevailing winds, slopes, or anisotropic resource gradients.

Generalised topological maps represent ecosystems using abstract spatial relationships and connectivity structures, emphasising the importance of topological adjacency or interactions rather than explicit geometric positions or shapes [UMTS09]. Unlike grid-based or Voronoi-Delaunay tessellation models, generalised topological maps prioritise topological relationships over exact geometric distances or spatial coordinates (Figure 4.3, right). Formally, these maps are defined as graphs $G = (V, E)$, where vertices $V$ represent spatial entities (e.g., habitat patches, resources, individuals, or territories), and edges $E$ define ecological interactions, such as connectivity, dispersal pathways, competitive influence, or predator-prey relations [HACV21, MU08, PSP*24].

By abstracting space into relational networks, these maps enable efficient representation and analysis of ecological processes in scenarios where exact geometry may be unknown, less relevant, or overly complex. They are particularly valuable in modelling metapopulation dynamics, habitat connectivity, or animal movement patterns, where the critical ecological factor is the presence or absence of connectivity rather than explicit spatial positions. Furthermore, generalised topological approaches facilitate integration with network theory, providing tools to analyse ecological resilience, robustness, fragmentation, and connectivity dynamics directly through graph-based metrics (e.g., connectivity indices, centrality measures, or modularity analyses) [LJGS23, GBH*12].

However, the use of generalised topological maps also presents several limitations and ecological assumptions:

- Removing explicit geometry sacrifices detailed spatial realism, potentially limiting the ability to accurately model distance-dependent ecological processes such as diffusion or continuous dispersal.
- Ecological outcomes can be highly sensitive to how vertices and edges are defined, necessitating careful selection or validation of the underlying graph structure to avoid artificial ecological patterns or biased results.
- Empirical calibration or validation may be difficult, as topological maps represent abstract ecological relationships rather than measurable spatial distances or tangible boundaries.

Despite these limitations, generalised topological maps remain valuable tools, especially when combined with explicit spatial approaches [ECC*21] or when explicit geometry is unavailable or unneeded such as in metapopulation studies where only patch connectivity is known [DARB18] or where movement follows network-like pathways shaped by the environment rather than simple Euclidean distance [BP22, MCB*14, CARR12].

#### Conclusion

While site-based models (grids, tessellations, topological maps) provide an efficient and conceptually simple way to represent ecosystems, they impose discretisations that may introduce artificial scales, isotropy assumptions, or directional biases. Our approach differs in that it does not partition space into fixed cells. Instead, we represent the environment as continuous fields updated by sparse environmental objects anchored on points, curves, or regions. This hybrid representation avoids the rigidity of predefined spatial partitions: continuous fields capture diffusion- and transport-like processes at any resolution, while objects encode semantic features (rivers, reefs, sand patches) that interact with those fields. Conceptually, this places our method between grid-based and individual-based models: similarly as site-based approaches, it supports efficient simulation of field processes, but in the same manner as IBMs, it maintains the individuality and heterogeneity of discrete entities.

### 4.2.2 Individual-based models

In contrast to aggregated or spatially discrete models, IBMs explicitly represent each organism as a distinct unit [CHM17]. IBMs have gained popularity in vegetation modelling because they capture variability in size, spatial positioning, and competitive interactions among individuals. Each tree or plant is modelled independently, with processes such as growth, mortality, and reproduction shaped by local environmental conditions and the presence of neighbours [Chn13, PGG*24].

This fine-grained representation enables IBMs to reproduce emergent stand-level dynamics from simple local rules. They are particularly effective for modelling competition for light, nutrients, and space, where outcomes depend on relative size or distance among neighbours [MSMM11, ZD20]. To reduce computational costs while maintaining ecological realism, vegetation IBMs often employ distance-based competition methods (e.g., FRN, ZOI, FON) or stochastic simplifications. Recent advances in parallel computing and GPU simulation have further extended the applicability of IBMs to large-scale virtual ecosystems.

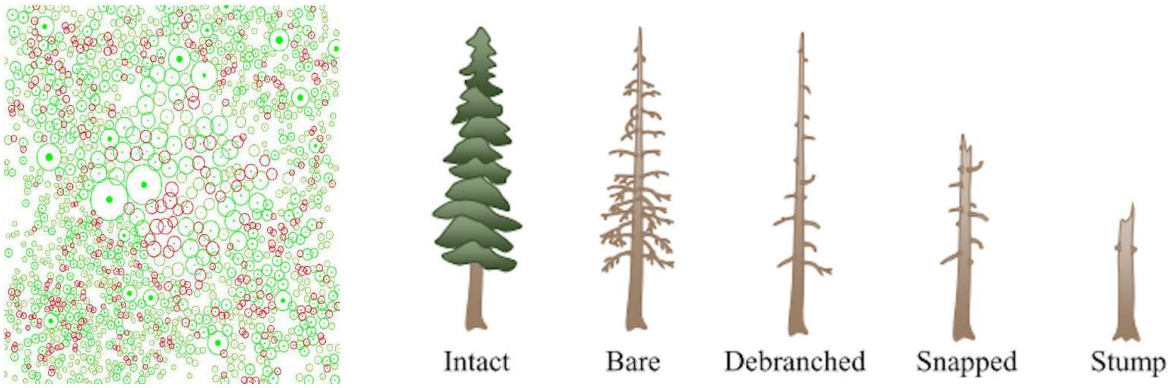



**Figure 4.4:** *Individual-based models focus attention on each element of the scene, with interaction between neighbouring entities. This perspective allows for more precision on (left) the positioning of each tree [AD05], but also on (right) the life cycle and geometry of the individuals [PGG*24].*

Although our discussion is focused on vegetation, IBMs are a general ecological framework. They are applied to animal movement and foraging, predator-prey dynamics, and even disease spread in heterogeneous populations [GR05, Chn13, PD09]. These examples underline the versatility of IBMs, but here we emphasise their role in vegetation modelling and spatial competition, which is most relevant for our purposes.

### Fixed-Radius Neighbourhood

The Fixed-Radius Neighbourhood (FRN) method is the first appearance of a distance-based method. Considering an application to forest ecosystems, each tree has a radius inside which we consider that there is shade and competition for nutrients. If two trees' radii intersect, a competition allows the stronger tree to survive while the weaker one dies [DHL*98].

In graphics, FRN-based exclusion is realised by Poisson-disk sampling (Figure 4.5) or blue noise approximations [DHOS00] to generate entities as points that do not fall within another entity's radius. The algorithm can be accelerated using a regular grid structure as proposed by Bridson [Bri07]: each cell of the grid has a width of $\frac{r}{\sqrt{n}}$ with $n$ the number of dimensions (usually $n = 2$). Each cell stores at most one sample, limiting neighbour checks to adjacent cells. Each point added to the grid recursively instantiates new random samples in a spherical annulus between radius $r$ and $2r$. Variants of this method guarantee maximal coverage [EDP*11] or propose a GPU parallelisation [Wei08]. However, in the presence of at least two species with different radii, the circle packing problem becomes more complicated and requires a larger number of collision checks, while still relying on the spatial indexing method from Bridson's algorithm.

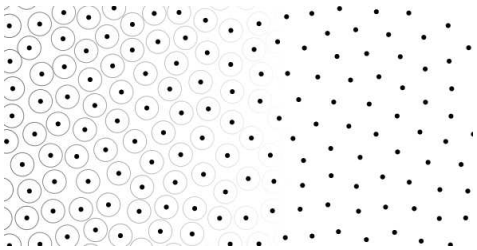


**Figure 4.5:** *Poisson Disk Sampling iteratively adds points in space and rejects a candidate if it is within radius r of any existing point.*

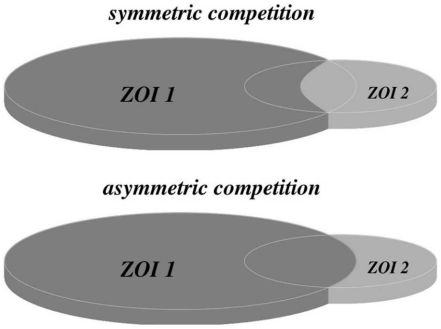



**Figure 4.6:** *ZOI is represented by regions around entities. The intersection area between two discs defines the competition between the two entities. Top: Symmetric competition (equal resource sharing, with the competition at the intersection divided equally); bottom: Asymmetric competition (larger entity captures more resources, such as light) [PWL*20].*

### Zone of Influence

The Zone of Influence (ZOI) model, used in Figure 4.4 (left) [AD05, AD06] is an ecological method designed to quantify competition among trees by modelling their interactions as gradual rather than binary processes [Cza98]. In contrast to a simple binary approache proposed with FRN, the ZOI model assigns each tree a circular influence zone, typically defined based on measurable attributes such as Diameter at Breast Height (DBH), crown dimensions, or species-specific ecological characteristics.
In this framework, competition between adjacent trees is quantified by calculating the fractional overlap between their influence zones (visible in Figure 4.6).

Mathematically, the intensity of competition $C_{ij}$ exerted by a neighbouring tree $j$ on a focal tree $i$ is expressed as:

$$C_{ij} = \frac{A_{\text{overlap},ij}}{A_{\text{ZOI},i}},$$

where $A_{\text{overlap},ij}$ is the area of overlap between the radii of trees $i$ and $j$, and $A_{\text{ZOI},i}$ is the total area of the influence zone of tree $i$. This intersection area is computed directly with

$$\begin{aligned} A_{\text{overlap},ij} =& r_i^2 \cos^{-1}\left(\frac{d^2 + r_i^2 - r_j^2}{2dr_i}\right) + r_j^2 \cos^{-1}\left(\frac{d^2 + r_j^2 - r_i^2}{2dr_j}\right) \\ & - \frac{1}{2}\sqrt{(-d + r_i + r_j)(d + r_i - r_j)(d - r_i + r_j)(d + r_i + r_j)}, \end{aligned}$$

assuming $r_i$ and $r_j$ are the radii of trees $i$ and $j$ respectively, and $d$ the distance between the two centres.

The biological consequences of competition quantified by ZOI, such as reduced growth or increased mortality risk, are represented through growth-response equations. For instance, tree growth under competition might be modelled as [Das12, UCCH04]:

$$G_i = G_{\max,i}\left(1 - \alpha \sum_j C_{ij}\right),$$

with $G_i \in [0,1]$ the growth of the tree, $G_{\max,i}$ the maximal potential growth if no competition affects it, and $\alpha$ a sensitivity parameter determining how competition impacts growth.

The ZOI model typically assumes symmetric (isotropic) competition, meaning all trees compete equally in every direction, and the competition effect between two trees is reciprocal if their zones and dimensions are similar. However, ecological realism can be increased by adopting an asymmetric (anisotropic) version of the model, which acknowledges that competition is often directionally biased or size-dependent (Figure 4.6). For asymmetric competition, the equation is modified by introducing weighting based on differences in tree size or dominance, for example:

$$C_{ij} = \frac{A_{\text{overlap},ij}}{A_{\text{ZOI},i}} f(DBH_i, DBH_j),$$

with $f : \mathbb{R}^2 \mapsto [0,1]$ weighting function that introduces asymmetry between individuals based on size or dominance traits. For example, $f(\text{DBH}_i, \text{DBH}_j)$ can bias competition towards larger trees by giving them a higher share of contested resources, while $f(h_i, h_j)$ with tree heights $h_i$ and $h_j$ can represent light competition by favouring taller individuals. The exact form of $f$ depends on the ecological process of interest (e.g., size dominance, shading, or nutrient uptake).

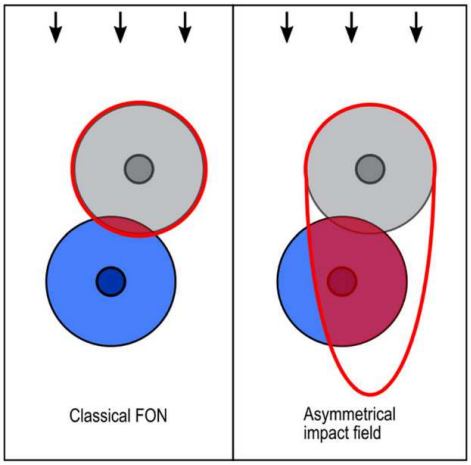



**Figure 4.7:** *A common simplification is isotropy; real stands often show directional effects (slope, wind, light). Left: competition in classical FON models is identical for each entity; right: a flow direction (black arrows) induce a deformed competition for resources. Asymmetric interaction area is much more complex to compute [PWL*20].*

The ZOI model is theoretically rooted in the understanding that trees compete gradually and continuously, rather than abruptly. However, it does rely on simplifying assumptions (circular, isotropic zones, homogeneous environmental conditions, and static radii within measurement intervals) which can limit its applicability to ecological scenarios involving directional resource gradients or heterogeneous environments (Figure 4.7).

Empirical studies across diverse forest ecosystems have demonstrated that the ZOI model significantly improves predictions of tree growth, survival, and mortality compared to simpler binary models such as FRN. The nuanced representation of competition offered by ZOI models better captures complex ecological dynamics, particularly in mixed-species stands, uneven-aged forests, or structurally diverse ecosystems [Bel72, BD92].

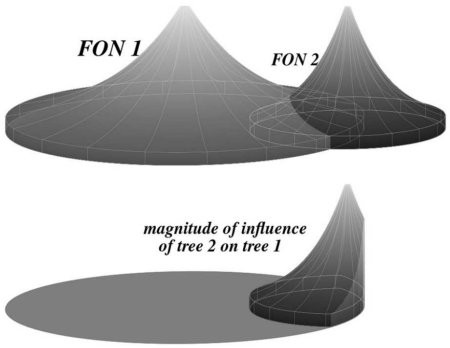



**Figure 4.8:** *The FON model proposes a smoother version than the ZOI, where the intersection between two radii also accounts for the distance from the tree trunk [PWL*20].*

### Field of Neighbourhood

The Field of Neighbourhood (FON) approach provides an alternative method to quantify tree competition by explicitly incorporating distance-dependent and size-dependent effects of neighbouring trees on a focal tree's growth or survival [BH00]. Unlike the ZOI model, which uses circular zones and geometric overlaps, FON models typically rely on explicit indices that incorporate both the distance and the relative sizes of neighbouring trees, without necessarily defining explicit geometric overlap areas.
In the FON approach, competition intensity affecting a given focal tree is calculated by summing the competitive effects of neighbouring trees, typically as a weighted function of their size and distance (Figure 4.8).

A commonly used formulation of competition intensity $FON_i$ for a tree $i$ at distance $r$ from the trunk is expressed as:

$$FON_i(r) = \begin{cases} 1 & \text{for } 0 \leq r < \text{RBH}, \\ e^{-C(r-\text{RBH})} & \text{for RBH} \leq r \leq R, \\ 0 & \text{otherwise} \end{cases}$$

, with RBH the radius at breast height (RBH $= \frac{1}{2}$DBH) (Figure 4.9), $R$ the radius of influence of the tree (usually proposed as $R = a\sqrt{\text{RBH}}$ with a scaling parameter $a$), and $C$ an attenuation coefficient, defined for each species.

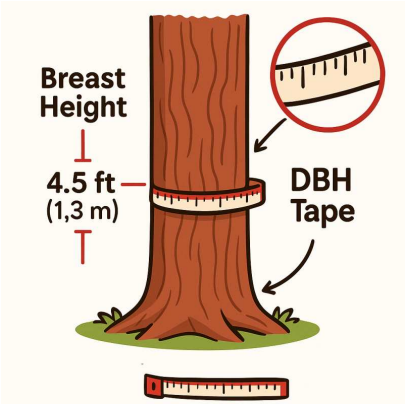



**Figure 4.9:** *DBH (and RBH) is measured at 1.3 m above ground level.*

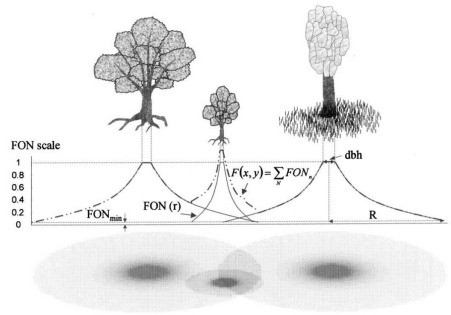



**Figure 4.10:** *The evaluation of the FON value at any point is done by accumulating all individual fields [BH00].*

The overall competition at any point in space can now be described as:

$$F(x,y) = \sum_i FON_i(x,y).$$

The competition perceived by a tree $i$ is the average of all $FON$ values intersecting its area $A$:

$$F_A^i = \frac{1}{A} \sum_{j \neq i} \int_{A_{\text{overlap},ij}} FON_j(x,y)\, da.$$

Unlike the ZOI method, the FON model does not assume a hard boundary around each tree, instead explicitly capturing a smooth decline in competitive impact with increasing distance. It also inherently accommodates asymmetric competition, since larger and closer neighbours exert stronger competitive influences than smaller or more distant trees (Figure 4.10).

The FON method does not rely explicitly on geometric overlap and is thus flexible and particularly useful in heterogeneous forests. However, to capture the self-thinning property of silviculture, an empirical law in forestry describing how tree density decreases as average tree biomass increases [MHS*19], satisfying $W = CN^{-\frac{3}{2}}$ (with $W$ the average biomass of an individual, $N$ the density, and $C$ a constant) [Wes84], the generation of the ecosystem should be iteratively simulated with small time steps [AD05], which is computationally expensive. Rather than simulating the full growth model of vegetation, most ecosystem generation algorithms focus on the distribution of the elements of the terrain, abstracting the competition into distance function deformations or statistical optimisation.

By designing non-monotonic field functions around trees, emergent behaviours can appear. Lane and Prusinkiewicz proposed to apply what they call a deformation kernel on the distance function, resulting in a larger variety of patterns, such as the clustering of bushes and the competition of larger trees (Figure 4.11, top left) [LP02]. Building on this idea, defining deformation kernels for each pair of species (Figure 4.11, bottom right, presents a kernel matrix for two species) allows the system to include ecological interactions between different species in the terrain. As a result, the simulation can introduce intra-species clumping while maintaining inter-species segregation.

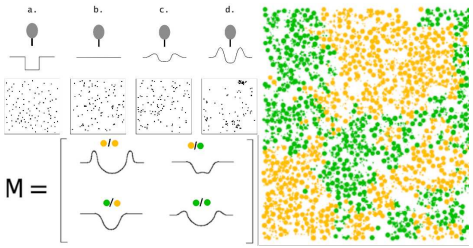



**Figure 4.11:** *Deforming neighbourhood kernels models species behaviours and interactions. Left: different kernels (top) yield distinct layouts (middle), that can be encoded in a inter-species matrix M (bottom); Right: resulting distribution [LP02].*

Alternatively, learning the spatial distribution of distances between individuals from examples can be used as a statistical metric to introduce interactive tools for ecosystem generation [EVC*15] (Figure 4.12). In this work, a user-provided patch serves as an example: the system extracts descriptors such as histograms of inter- and intra-species distances, or correlations with environmental variables such as slope, and reproduces these distributions in other regions. Unlike kernel-based models, which require explicit design of influence functions, this approach learns statistical descriptors directly from data.

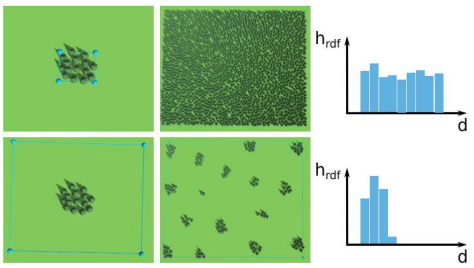



**Figure 4.12:** *Using statistical distribution from an input terrain, a different terrain with similar distributions is synthesised [EVC*15]. Two examples are given (delimited by blue rectangular boundaries): uniform distribution (top) and clustered distribution (bottom). Differences in distribution arise from the voids in the regions of interest.*

This method is not limited to vegetation but generalises to any scene elements (e.g., houses, rocks, roads), making it highly flexible for procedural content generation. From an ecological standpoint, it resonates with point-process modelling, where distributions of distances or clustering statistics are fitted to observed spatial patterns [Ben21, Bad10]. For computer graphics, the strengths of this method are interactivity and scalability: complex, realistic layouts can be synthesised quickly without costly simulation, while retaining ecological plausibility through statistical consistency.

While the FON model provides a biologically detailed representation of competition, its computational complexity makes it significantly more challenging to implement in large-scale simulations. Unlike the ZOI model, which relies on predefined geometric overlaps that can be calculated efficiently, the FON approach requires integrating a continuously decaying competition function over an arbitrary area of influence. This necessitates evaluating the function at multiple points across overlapping zones, increasing computational demand. Moreover, the additive nature of FON, where each tree contributes to the total competition field at every spatial point, further exacerbates the computational cost, scaling poorly in dense forest stands. The need for numerical integration or spatial discretisation makes FON computationally expensive compared to ZOI, which only requires simple geometric calculations. Consequently, while FON is useful for small-scale or highly detailed ecological studies, it is generally impractical for large-scale forest simulations.

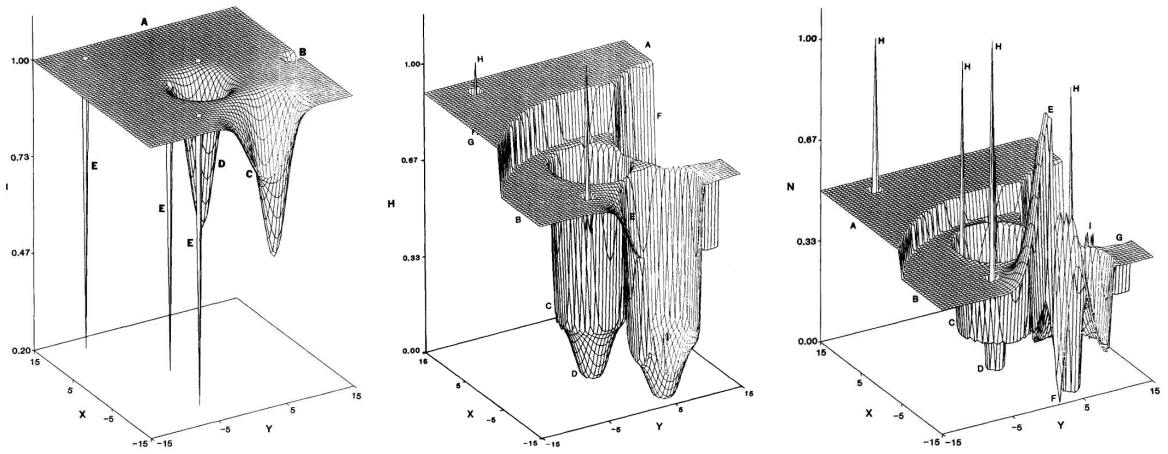



**Figure 4.13:** *EFMs represent the environment as a mathematical function where environmental properties such as humidity, shade, and nutrients are seen as distinct fields (from left to right respectively) created by the presence of trees, bushes, and flowers [WSWP85].*

### Conclusion

Whereas FRN, ZOI, and FON define interactions through explicit neighbour relations, our approach shifts this logic into continuous environmental fields. Individuals still act as discrete units, but they interact indirectly by modifying and sensing shared fields, aligning IBMs with the EFM perspective.

## 4.2.3 Ecological Field Models

While the site-based and individual-based approaches have been successfully used in various ecological studies, real-world ecological processes do no work on a discrete grid or as a set of independent agents. Instead, they operate in a continuous spatial domain, where organisms interact with smooth environmental gradients rather than discrete neighbours. For example, competition for light and nutrients is not limited to direct neighbours.

To address these challenges, EFMs provide an alternative framework where environmental properties are represented as continuous, spatially explicit fields [WSWP85]. Rather than defining interactions through direct contact (as in IBMs) or discrete adjacency (as in SBMs), EFMs describe how organisms modify and respond to smooth environmental gradients [Chn11, SRSS12].

From a computer graphics perspective, EFMs share similarities with Signed-Distance Fields (SDFs). Just as SDFs represent geometry through continuous distance fields, EFMs represent organism-environment interactions through continuous ecological fields, allowing interactions to be modelled as smooth gradients rather than discrete neighbour effects.

EFMs are based on the fundamental idea that organisms are not isolated agents but rather integral components of their environment, influencing and being influenced by it in a spatially explicit manner. Figure 4.13 shows the effect of multiple vegetation entities on the humidity, shade, and nutrient fields of the environment. The core ecological principles behind EFMs include two key points:

- **Continuous organism-environment interactions:** organisms modify their surroundings beyond their immediate location. Environmental factors such as soil nutrients, water availability, light exposure, and chemical signals influence other species gradually across space,
- **Environmental feedback:** species respond to spatially varying gradients rather than direct neighbour-based interactions, as seen in phototropism (guiding growth toward light) [PSK*12] or root development towards nutrient-rich soils [LKM*23].

The combination of these two principles creates natural feedback loops that are explicitly modelled, allowing for a more realistic representation of ecological interactions compared to static environmental assumptions in traditional models.

EFMs provide a continuous representation of environmental factors, bridging the gap between grid-based and individual-based models by modelling smooth ecological gradients, instead of forcing interactions into discrete cells. This method allows organisms to influence and respond to fields dynamically, while providing a scalable framework that avoids the exponential computational cost of tracking every individual separately.

While EFMs provide a biologically grounded approach to modelling spatial interactions, their implementation requires an analytical framework to define how organisms modify and respond to environmental fields.

An ecological field represents a spatially varying environmental property across a landscape. Mathematically, we can define an $n$-dimensional ecological field as a scalar field $F$ for each point $\mathbf{p} \in \mathbb{R}^n$:

$$F(\mathbf{p}) : \mathbb{R}^n \mapsto \mathbb{R}.$$

Any organism $i$ in the scene has an influence on the ecological fields through an influence kernel $K_i$. These functions describe how an organism's presence affects its surroundings. We can then evaluate the ecological field with $N$ entities:

$$F(\mathbf{p}) = F_0(\mathbf{p}) + \sum_{i=0}^{N} K_i(\mathbf{p}), \tag{4.1}$$

with $F_0(\mathbf{p})$ the baseline environmental state.

The primary challenge in computer graphics is how to efficiently store, sample, and modify continuous environmental fields. Since EFMs define spatially varying properties, they require a structured representation that balances accuracy, efficiency, and scalability.

Several strategies have been proposed to implement ecological fields in practice, ranging from grid discretisations [WSWP85, SRSS12] to procedural functions inspired by noise and implicit surfaces [Per85, FPRJ00]. In computer graphics, field-based representations are long established as a means to control geometry or appearance, for instance through implicit modelling, distance fields, or scalar textures. Building on these ideas, the Environmental Objects framework [GPG*16] can be seen as a concrete instantiation of EFM principles: fields are directly attached to objects, enabling them both to emit and sense spatial influences. Although originally designed for urban scenes, its conceptual alignment with EFMs illustrates how continuous organism-environment feedbacks can be embedded into procedural modelling systems.

### Environmental Objects

The Environmental Objects system [GPG*16] implements this principle in practice by extending scene assets with scalar fields (e.g., heat, humidity, occlusion). These fields govern local interactions, such as snow, leaves, or soot accumulating on surfaces, and can in turn influence the appearance and behaviour of nearby objects. Originally developed for interactive urban set dressing, the method introduces a hierarchical framework for procedural scenes in which object-level fields control both visual details and environmental coherence (Figure 4.14).

Environmental Objects in a virtual scene dynamically adjust their appearance based on surrounding scalar fields, allowing for realistic procedural effects illustrated with snow deposition, leaf accumulation, and icicle formation. Unlike global simulations, which are computationally demanding, this model uses local environmental fields to procedurally generate details without requiring extensive physics-based calculations.

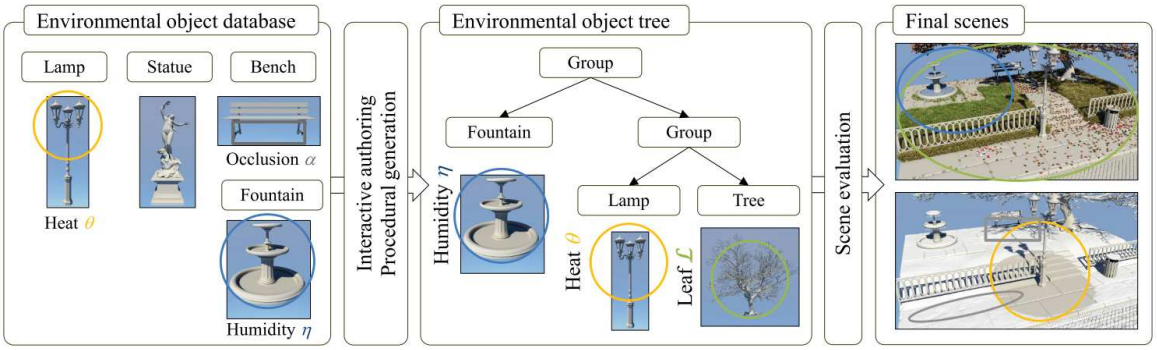



**Figure 4.14:** *The Environmental Objects framework [GPG*16] presents a practical methodology for integrating spatially explicit environmental interactions into procedural modelling. The Environmental Objects concept aligns closely with EFMs, providing a computationally feasible way to embed field interactions into simulated ecosystems.*

Users can modify environmental parameters, such as temperature fields, to influence scene aesthetics in an intuitive and predictable manner. The model enables scene composition through level of details using object groupings, optimising the computation of large-scale environments while maintaining local control over environmental interactions.

By employing implicit primitives to define the environmental fields and procedural blending techniques, this framework enables efficient simulation of environmental changes while allowing for real-time scene modifications.

In this method, each 3D model can be associated with "procedural effects", which regroup scalar fields representing changes in the environment (Figure 4.14, left). These fields are defined using implicit primitives (point-, spline-, surface- or volume-based) allowing the environment properties to be changed along a path, a surface or a volume, instead of only using points as in the initial EFM. This enables the system to represent, for example, the diffusion of heat around a pipe, which is not possible with the latter. The computation of an environment property is then an agglomeration of all the scalar fields associated with the given property (Figure 4.14, middle). The authors propose computing all scalar fields, such as the heat field $\theta$, at any position $\mathbf{p}$ by the accumulation of objects' fields, in a similar manner as Equation (4.1):

$$\theta(\mathbf{p}) = \theta_0(\mathbf{p}) + \sum_{i=1}^{n} \theta_i(\mathbf{p}),$$

and the occlusion field $\alpha$ as the product of all objects' occlusion fields:

$$\alpha(\mathbf{p}) = \alpha_0(\mathbf{p}) \prod_{i=1}^{n} \alpha_i(\mathbf{p}),$$

assuming $\theta_0$ and $\alpha_0$ to be global scalar fields for temperature and shade, symbolising the baseline environmental state in the ecological domain.

Using the field values, the system instantiates small details such as leaves by sampling the space with candidate "anchor positions" (Figure 4.14, right). If the field values satisfy certain conditions at the anchor position $\mathbf{p}_i$, the small-scale objects are instantiated at this position (for example, for leaves: $\alpha(\mathbf{p}_i)l(\mathbf{p}_i) < d_i$ using the occlusion field $\alpha$, the leaves deposition field $l$ and the distance of the anchor position to the depositing surface $d_i$ as shown in Figure 4.15). This method was extended to modify the geometry or texture of the object based on the field values (in Figure 4.15, the heat field and the humidity field), to manually move or rotate each of the objects, or even integrate user-defined density fields to populate entangled details on the ground [GGG*16].

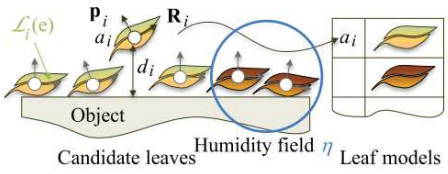



**Figure 4.15:** *Leaves are instanced if the field values are fitted for leaves presence, and the geometry and texture are altered depending on the environment [GPG*16].*

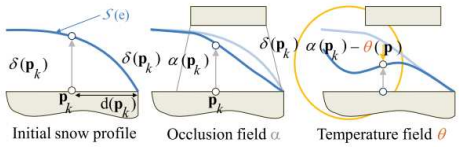



**Figure 4.16:** *Snow deposition is a displacement of vertices along the surface normals of the mesh (left), modulated by the value of fields such as occlusion (middle) and heat (right) [GPG*16].*

On the other hand, the geometry of large objects is altered using the scalar fields. The authors present a simulation of snow deposition on the objects by displacing the vertices $i$ in the direction of the original surface normals $\vec{n}_i$ by an amount depending on the environment (Figure 4.16):

$$\mathbf{p}'_i = \mathbf{p}_i + (\alpha(\mathbf{p}_i)s(\mathbf{p}_i)g(d(\mathbf{p}_i)) - \theta(\mathbf{p}_i))\,\vec{n}_i,$$

with $s$ the snow field, and $g$ a snow elevation function based on the distance between the original vertex and the border of the mesh.

In addition to the EFM framework, environmental effects from environmental objects are not limited to point sources (fields can be emitted by splines and volumes) as the scalar functions may use a spline primitive or be extended to volume primitives (Figure 4.17). Translated to the ecological domain, this methodology extends the EFM as the inclusion of water bodies, essential for realistic vegetation simulation, could be taken into account by considering environmental effects from rivers and lakes which are often modelled as splines and regions in 3D modelling [EPCV15, GGG*13, GGP*15].

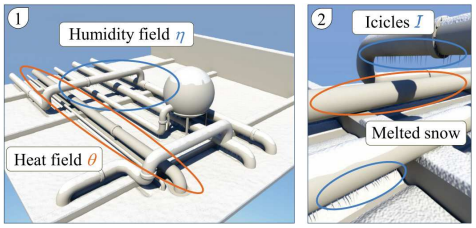



**Figure 4.17:** *The heat field produced by a pipe uses a distance function from a line instead of a single point in space [GPG*16].*

The Environmental Objects method can be regarded as a variant of EFMs, since both rely on scalar fields that objects emit and sense. They share a common substrate of composable fields, but their purposes differ: Environmental Objects mainly enrich visual appearance, while EFMs model ecological dynamics. This overlap suggests cross-overs: Environmental Objects techniques can improve the scalability and visualisation of EFMs, while EFMs can provide Environmental Objects biologically grounded feedback loops.

Our approach adopts this fields-attached-to-objects view, but upgrades it to a bi-directional simulator: objects both read fields to adapt and write fields to decide where/whether they should exist, yielding emergent spatial distributions.

However, classical EFMs assume static fields, whereas many ecological processes evolve dynamically (e.g., soil moisture depletion, growth-induced shading, or decaying scent trails). By integrating biologically inspired feedback loops, procedural generation could extend beyond static environmental effects, enabling simulations of growth, decay, and species interactions [OL01, PMG*22]. Capturing such time-dependent feedbacks requires a mathematical framework for field dynamics. The next section therefore turns to reaction-diffusion equations, which provide the computational backbone for

modelling growth, depletion, and decay in ecological fields.

### Advection-Reaction-Diffusion

Reaction-diffusion models describe how a quantity $u(x,t)$ (e.g., population density, nutrient concentration, sediment stock) evolves under two coupled processes: local reactions and spatial diffusion. In its classical isotropic form [Tur52, OL01]:

$$\frac{\partial u}{\partial t} = D\nabla^2 u + R(u), \tag{4.2}$$

where $D$ is the diffusion coefficient and $R(u)$ encodes local growth, decay, or interactions. Adding transport by external flows yields the full advection-reaction-diffusion formulation.

#### *Diffusion*

Diffusion models the random spread of material, following Fick's law:

$$\text{Flux} = -D\nabla u. \tag{4.3}$$

This produces isotropic spreading, equivalent to convolving point inputs with a Gaussian kernel. In heterogeneous environments, anisotropic diffusion generalises $D$ to a tensor $\mathbf{D}$ [Ram24], introducing directional bias (e.g., slope-driven seepage, anisotropic dispersal in currents). In graphics, diffusion has long been exploited for mesh smoothing and fairing [Tau95], anisotropic image filtering [PM90], and texture synthesis based on reaction-diffusion patterns [Tur91]. GPU acceleration allows large grids to be updated interactively, which directly informs our use of diffusion for transporting environmental materials.

#### *Reaction*

The reaction term governs local dynamics:

- **Linear reaction:** $R(u) = \lambda u$, describing exponential growth ($\lambda > 0$) or decay ($\lambda < 0$).
- **Nonlinear reaction:** logistic growth, predator-prey interactions, or Lotka-Volterra competition terms capture resource limitation or trophic interactions [BC12, Ver44].

In our framework, the reaction term corresponds directly to the production or consumption of environmental materials by environmental objects. For example, a coral colony deposits calcium carbonate ($\lambda > 0$), while erosion removes sediment ($\lambda < 0$). In graphics, reaction-diffusion systems have been widely used for texture and pattern synthesis, where extending $u(x,t)$ to multiple interacting fields enables the emergence of Turing-like patterns such as stripes or spots [Tur91, WK91, SKJY06].

#### *Advection*

Advection introduces directed transport under an external velocity field $\mathbf{v}(x)$ (e.g., currents, wind, gravity-driven runoff):

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D\nabla^2 u + R(u). \tag{4.4}$$

Unlike diffusion's random spread, advection shifts material coherently along streamlines, aligning with our processing of water currents as vector fields [BGI*20, OAL*17]. In graphics, this formulation underpins fluid solvers such as Stam's "stable fluids" [Sta99], where advection steps move smoke, fire, or water density across a grid. GPU-accelerated advection is also widely used in real-time effects such

as smoke trails or rivers, making it a natural tool to simulate ecological transport processes, which is presented in depth in Chapter 5's state of the art.

*Decay and steady state*

To avoid indefinite accumulation, a decay term $-\mu u$ is added:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D\nabla^2 u + R(u) - \mu u. \tag{4.5}$$

The decay term $\mu > 0$ guarantees that even with continuous deposition, the system relaxes toward a steady state where inputs balance dispersal and losses. This makes advection-reaction-diffusion models particularly suitable for efficient simulations of environmental materials, where explicit time-stepping can be replaced by directly solving for equilibrium. In graphics, decay is often added for numerical stability and realism: without it, transported quantities such as heat, smoke, or density fields accumulate without bound. In our case, decay serves the same dual purpose: guaranteeing stability and providing interpretable equilibrium states.

*Green's function solutions*

For point sources, the advection-reaction-diffusion equation admits analytic solutions via Green's functions. For example, in 2D with isotropic diffusion, constant advection $\mathbf{v}$, and decay $\mu$, the fundamental solution is a shifted, damped Gaussian:

$$G(x,t) = \frac{1}{4\pi D t} \exp\left(-\frac{\|x - \mathbf{v}t\|^2}{4Dt} - \mu t\right). \tag{4.6}$$

This solution represents a spreading plume: diffused by $D$, advected along $\mathbf{v}$, and exponentially damped by $\mu$.

For extended sources such as curves $C$ or regions $\Omega$, the solution is the convolution of the point-source Green's function with the source geometry:

$$u(x,t) = \int_C G(x-y,t)\, dy, \tag{4.7}$$

$$u(x,t) = \int_\Omega G(x-y,t)\, dy. \tag{4.8}$$

Closed-form solutions exist only for simple shapes such as infinite lines and disks. For arbitrary geometries, these integrals must be evaluated numerically.

## Conclusion

The advection-reaction-diffusion formalism provides a compact vocabulary to describe four fundamental mechanisms of ecosystem dynamics: diffusion (random spread), reaction (local production or consumption of resources as environmental materials deposition and uptake), advection (directional transport by flows) and decay (loss mechanisms ensuring stability).

These mechanisms are the backbone of how we model the interaction between discrete environmental objects and continuous ecological fields, combining analytic tractability for simple cases with numerical flexibility for complex geometries.

## 4.3 Our pipeline

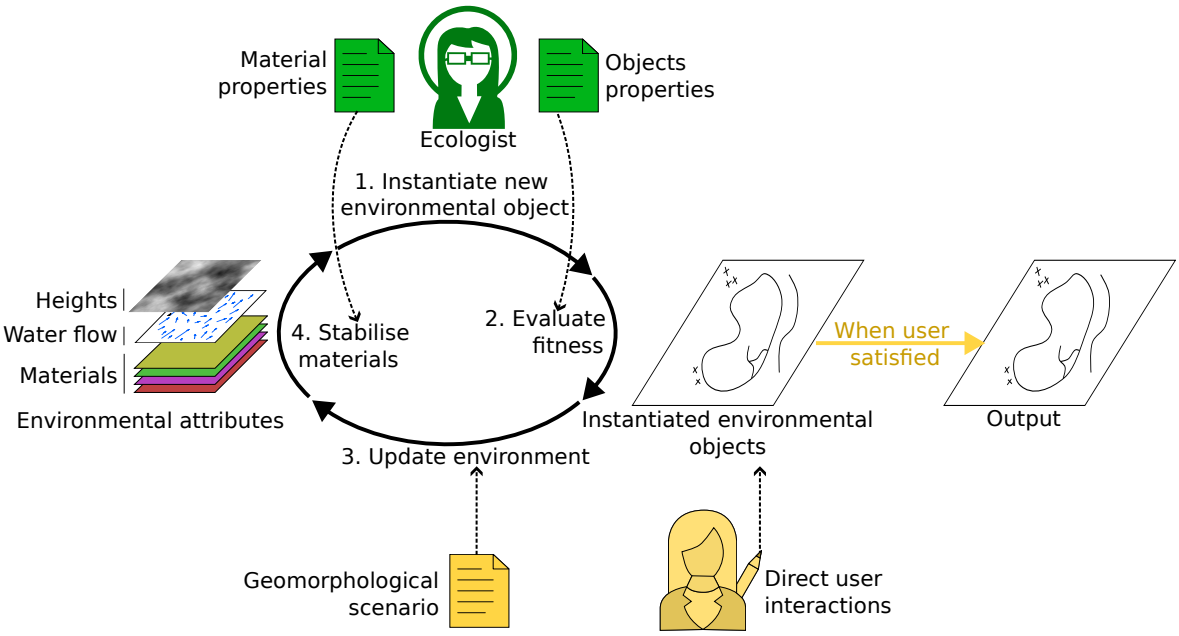



**Figure 4.18:** *Overview of the pipeline of our iterative generation process: at each iteration, (1) the environment instantiate a new environmental objects, (2) environmental objects evaluate their fitness with respect to user modifications then (3) update the environment (Section 4.3.4) until (4) stabilisation with user-defined geomorphological events (Section 4.3.5). When the resulting scene suits the user, the sparse representation of generated environmental objects is returned (Section 4.3.6) and can be instantiated and rendered. Ecologist knowledge (green) is injected through material and object properties, while user control (yellow) appear on instantiated objects, environmental attributes, and stopping conditions.*

### 4.3.1 Overview

Our method generates plausible, editable environments by coupling discrete semantic primitives with continuous environmental fields. As illustrated in Figure 4.18 and summarised in Algorithm 4, the system forms a closed feedback loop in which sparse objects (environmental objects) interact with environmental quantities (environmental attributes) through local modifiers (environmental modifiers). This captures how terrain features both depend on and reshape their surroundings.

The framework rests on three complementary abstractions (introduced in Section 4.3.2):

- environmental attributes describe the environment as continuous scalar and vector fields,
- environmental objects are sparse, scale-independent primitives with a skeleton and per-class rules,
- and environmental modifiers mediate their mutual influence by translating object presence into local field updates.

Generation proceeds iteratively: starting from an initial configuration of terrain, water level, materials, and an object catalogue, the system alternates between object instantiation and environment updates (Section 4.3.4). Users can intervene at any time by editing objects or triggering geomorphic events (Section 4.3.5). The process converges to a steady state and yields a sparse semantic scene suitable for meshing or rendering (Section 4.3.6).

**Input:** Base height field $h$, water level $\mathcal{L}$, user flow $\mathcal{W}_{\text{user}}$, catalogue $\widetilde{\mathcal{O}}$, target multiset $\hat{\mathcal{O}}$, material params for $\mathcal{M}$
**Output:** Sparse set of instantiated environmental objects

**Initialisation:**
$\mathcal{H} \leftarrow h - \mathcal{L}$
Initialise $\mathcal{M}$ scalar fields (zeros or expert priors)
$\mathcal{W}_{\text{simulation}} \leftarrow$ terrain-aware flow from $\mathcal{H}$ $\mathcal{W} \leftarrow \mathcal{W}_{\text{user}} + \mathcal{W}_{\text{simulation}}$
$\mathcal{E} \leftarrow (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$
$\mathcal{O} \leftarrow \varnothing$
**Generation loop:**
**while** *not stop (max iters / $\hat{\mathcal{O}}$ reached / user approval)* **do**
  `// (1) Instantiate new objects`
  **foreach** *class* $\in \widetilde{\mathcal{O}}$ **do**
    sample position: $\mathbf{p}_s \leftarrow \arg\max_{\mathbf{p}_k}\ \omega_{\text{class}}(\mathbf{p}_k, \mathcal{E})$
    **if** $\omega_{class}(\mathbf{p}_s, \mathcal{E}) > threshold_{class}$ **then**
      compute object's skeleton: $skeleton_o \leftarrow \Gamma_{\text{class}}(\mathcal{E})$
      add new object $o$ to $\mathcal{O}$
  `// (2) Environment update from environmental modifiers`
  $\mathcal{H} \leftarrow$ recompute from all objects height modifiers $\mathcal{H}_o^+$
  $\mathcal{W}_{\text{simulation}} \leftarrow$ recompute from $\mathcal{H}$
  $\mathcal{W}^+_{\text{objects}} \leftarrow \sum_{o\in\mathcal{O}} \lambda_o\, \mathbf{u}_o$
  $\mathcal{W} \leftarrow \mathcal{W}_{\text{user}} + \mathcal{W}_{\text{simulation}} + \mathcal{W}^+_{\text{objects}}$
  $\mathcal{M} \leftarrow$ advection-reaction-diffusion
  $\mathcal{E} \leftarrow (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$
  apply direct user edits to skeleton
  apply geomorphic events analytically on $\mathcal{E}$
  $\mathcal{O} \leftarrow$ remove unfit objects
**return** $\mathcal{O}$ `// sparse environmental objects`

**Algorithm 4:** environmental objects generation pipeline.

### 4.3.2 Core concepts

Our approach extends the concept of environmental objects, simplified, scale-agnostic terrain features that interact with their surroundings to simulate complex ecosystem dynamics. These environmental objects, which represent natural features (corals, algae, islands, rocks, ...) influence and are influenced by environmental fields (temperature, humidity, elevation, ...).

In this section, we present the different related concepts that underpins our method. We first introduce the continuous environmental fields (environmental attributes), then the discrete actors that perceive and affect these fields (environmental objects), before detailing how they modify the fields (environmental modifiers) and how the full pipeline orchestrates their interaction.

#### Environmental attributes

In the field of geography and for EFMs, a "field" is a spatially continuous variable defined at every point of the domain, encompassing scalar fields (elevation, temperature) and vector fields (wind,

current). In the following, we name every such quantity an environmental attribute.

In an ecosystem simulation, each actor has an impact on all other actors, which results in an exponentially growing computation effort which becomes problematic as the number of elements of the terrain increases. Direct pairwise interaction between $n$ organisms scales as $\mathcal{O}(n^2)$. Instead, we adopt the standard EFM strategy: every environmental object writes to local environmental attributes through its environmental modifier. Each object then reads them, turning the global complexity into a linear complexity per simulation step and thus remaining interactive even for thousands of objects.

We bundle the currently supported fields under the unified term "environment", denoted as $\mathcal{E} = (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$, where $\mathcal{H}$ (height) and $\mathcal{L}$ (water level) are scalars, $\mathcal{W}$ (water currents) is a 2D vector field, and $\mathcal{M}$ (environmental materials) is a vector of per-point resource stocks (sand, salt, moisture, limestone, ...).

### Environmental objects

A geographical feature, also called object or entity, is defined as a discrete phenomenon located at or near the Earth's surface. It represents geographic information that are depicted in maps, GIS, and other forms of geographic media. This term includes both natural and human-made objects, ranging from tangible items (e.g., buildings or trees) to intangible concepts (e.g., neighbourhoods or savanna). Features are distinct entities with defined boundaries, differentiating them from continuous geographic masses or processes occurring over time. They can be categorised as natural features, such as ecosystems, biomes, water bodies, and landforms, or artificial features (e.g., settlements, administrative regions, and engineered constructs). We use the term environmental object in this work to avoid the ambiguous term "feature".

Each environmental object is anchored by a skeleton (a point, curve, or region in the cartographic sense) which captures its footprint while postponing heavy geometry generation (Figure 4.19). Our objects are:

- **Point-based**: isolated boulders, individual trees, individual corals, ...
- **Curve-based**: river, coral reefs, fault scarps, ...
- **Region-based**: islands, forests, sediment patches, ...

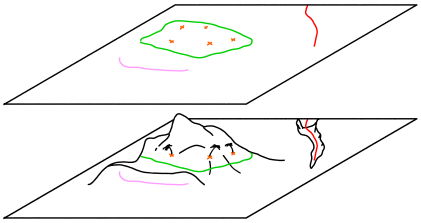


**Figure 4.19:** *Environmental objects, from 2D abstract geometric shapes (top) to a possible 3D geometry (bottom).*

Via its specified environmental modifiers, an environmental object locally deposits, removes, or diverts environmental materials, thereby updating the surrounding environmental attributes. Conversely, its viability is re-evaluated through a user-defined fitness function that samples those same fields. If viable, a skeleton fitting function refines the skeleton (e.g., a river streams downhill, and seagrass meadow patches are limited to luminous areas).

### Environmental modifiers

The environment determines if an environmental object can survive at a certain position. When an environmental object is placed, its surrounding environmental attributes are affected through environmental modifiers $\mathcal{E}^+ = (\mathcal{H}^+, \mathcal{W}^+, \emptyset, \mathcal{M}^+)$ where $\mathcal{H}^+$ defines a change of height, $\mathcal{W}^+$ the modifications of the water current field, and $\mathcal{M}^+$ the environmental materials alterations. The water level $\mathcal{L}$ is not modified by an object, hence $\emptyset$.

Environmental objects are subject to altitude constraints. However, to maintain a clear distinction between semantic modelling and 3D modelling, we do not compute the exact physical shape or detailed height field of each environmental object. Instead, we define a coarse height function $\mathcal{H}^+$,

a parametric surface derived from the skeleton, that provides a coarse estimate of local elevation changes. This coarse proxy lets us evaluate altitude-dependent rules without the cost of globally computing the height field.

Altering the vector field $\mathcal{W}$ is performed by composing the individual contributions $\mathcal{W}^+$ of each object at position $\mathbf{p}$, following [WH91]. For efficiency we adopt an analytical deformation kernel inspired by Kelvinlet solutions [GJ17] to aggregate these local disturbances.

Each environmental object possesses intrinsic environmental materials that both spread and are absorbed around its skeleton over time. For example, a coral colony deposits limestone while simultaneously slowing currents; the reef then uptakes that limestone for growth. In our model the colony contributes a deposition term $\mathcal{M}^+_{\text{limestone}}$, and the reef absorbs it. No direct object-object communication is required.

Scalar environmental attributes are updated by adding or removing mass around the skeleton and then diffusing it, biased by $\mathcal{W}$. We assume a steady state regime, guaranteed by a decay rate $\mu \in [0, 1[$ that prevents unbounded accumulation during advection-diffusion.

### 4.3.3 Initialisation

We initialise the terrain with a height field $h$ and a water level $\mathcal{L}$. As a large number of underwater environmental objects depends on altitude or depth, we use a shorthand notations $\mathcal{H} = h - \mathcal{L}$ (signed height above water) and $\mathcal{D} = -\mathcal{H}$ (signed depth below water). The height field provides variation in altitude, which influence the generation process of the scene.

A catalogue of available environmental objects $\widetilde{\mathcal{O}}$ is supplied by field experts; the user provides an optional target list of environmental objects $\hat{\mathcal{O}}$ used as a stop condition of the process.

Finally, different environmental materials are defined with properties: diffusivity, density, decay rate, and advection sensitivity. These parameters are chosen with field experts to reflect plausible physical behaviour.

An initial environment configuration, resulting from the initial height field $h$ and water level $\mathcal{L}$, the environmental materials distribution $\mathcal{M}$ represented as scalar fields $\mathcal{M} : \mathbb{R}^2 \mapsto \mathbb{R}$, and water currents $\mathcal{W}$ as a vector field $\mathcal{W} : \mathbb{R}^2 \mapsto \mathbb{R}^2$, is then available for all environmental objects to evaluate growth and spawning. The set of environmental attributes is noted $\mathcal{E} = (\mathcal{H}, \mathcal{W}, \mathcal{L}, \mathcal{M})$.

The definition of environmental objects properties and environmental attributes is done with field experts, who first decide which environmental materials layers are relevant for the target biome and then anchor the corresponding parameters in the system. The generation phase starts from this environment plus an optional seed set of environmental objects.

### 4.3.4 Generation process

Once initialization is complete, we enter an iterative generation phase in which objects and environment co-evolve until a steady state is reached. Each iteration alternates between two main stages:

- **object instantiation and shaping**, where new environmental objects are proposed, evaluated, and fitted to their surroundings and
- **environment update**, where these objects modify the environmental fields through their environmental modifiers.

The iterative loop continues until user-defined termination criteria are met such as a fixed number of iterations, the completion of a target object set $\hat{\mathcal{O}}$, or explicit user approval. During this process, users may intervene at any time to steer the generation, either by editing individual environmental objects

**Figure 4.20:** *While the fitness function guides the position of the environmental objects, the skeleton fitting function provides an indication of the suitability of the environmental object in its surroundings. This information is used to determine if the environmental object should be removed, but may also be used for visual purposes to indicate unhealthy corals or eroded rocks.*

interactively or by triggering large-scale geomorphic events that globally affect the environmental fields.

The following subsections detail these mechanisms: the instantiation of candidate objects and their fitness-based adjustment, the update of height, flow, and material fields via local modifiers, and the interactive and event-driven interventions that allow continuous user control.

### Placement of environmental objects in an environment

At each iteration of our algorithm, we aim to place new environmental objects at plausible locations. Because we do not enforce physical time continuity, our goal is to progressively enrich the scene according to user intent while maintaining ecological and geological plausibility. Since each environmental object modifies the environment when instantiated, potentially destabilising previous placements, the goal is to insert new elements where they minimally disturb the existing scene.

Our placement algorithm proceeds in two steps: first, it identifies the globally most suitable position using the fitness function $\omega$, which depends on environmental attribute (altitude $\mathcal{H}$, water flow $\mathcal{W}$, water level $\mathcal{L}$, and environmental materials availability $\mathcal{M}$). Second, we refine the object's shape using its skeleton fitting function $\Gamma$.

**Fitness function** Our placement method take inspiration from "Darwinian fitness", refering to an organism's ability to survive and reproduce in its environment. It is a measure of how well-suited an organism is to its surroundings, and those with higher fitness are more likely to pass on their genes to the next generation. In a similar manner, the fitness function of an environmental object evaluates its suitability at a location in the terrain, extending the meaning to living features (e.g., forests and corals) and non-living elements (e.g., reefs and lagoons).

The fitness function $\omega(\mathcal{E}) : \mathbb{R}^2 \mapsto \mathbb{R}$ quantitatively evaluates how well a given environment location satisfies the requirements of environmental objects. In other words, $\omega$ returns a numeric score representing the suitability of local environmental conditions for each object. This function considers various environmental variables such as altitude, the presence of specific materials, water current strength, and their respective gradients. Defining separately $\omega$ for each environmental object class allows the algorithm to be tailored to the unique needs and tolerances of different object types. High $\omega$ values indicate favourable sites for the object, while low values correspond to unsuitable or inhospitable locations.

Different types of environmental objects require different criteria for their fitness evaluation. For example, a river might prioritise lower altitude and proximity to water currents, while a forest might prioritise higher altitude and specific material availability. The flexibility of the fitness function allows it to be customised for each environmental object, ensuring that the generated terrain remains coherent and realistic. To ensure that a coral entity appear at at position around 10m deep, with low wave energy, low sand concentration and on a flat surface, we encode these constraints as:

$$\omega_{\text{coral}} = \text{normal}(\mathcal{D} - 10) - \|\nabla\mathcal{D}\| - \mathcal{M}_{\text{sand}} - \|\mathcal{W}\|.$$

In practice, to find suitable locations for an object, we evaluate its $\omega$ across the environment's spatial domain by randomly sampling $\omega$ values over the terrain, effectively producing a fitness map for that object class, where peaks indicate highly suitable locations. Each object class thus yields its own fitness map. The position with the best fitness value serves as seeding point to shape the environmental object's skeleton thanks to the skeleton fitting function

The value of the fitness function can be used later on to modify the geometric representation or the appearance of the environmental object to appear more eroded (Figure 4.20).

**Skeleton fitting function** The candidate object is first seeded at its optimal the position $\mathbf{p}_s = \arg\max_{\mathbf{p}} \omega(\mathbf{p})$. This seed acts as the initial guess for the subsequent shape optimisation for the skeleton fitting function $\Gamma(\mathcal{E}) : \mathbb{R}^2 \mapsto \mathbb{R}$ described below. Moreover, this two-step optimisation allows a coarse approximation of the global maximum of the fitness function with few samples, reducing its computational demand, as the skeleton fitting function optimisation is focused on a single seed.

*Point-based skeleton*

For objects whose skeleton reduces to a single point (boulders, individual corals, etc.), we refine the seed by hill-climbing on a (possibly different) fitting field $\Gamma$. In most cases, we simply set $\Gamma = \omega$, but the user may supply a sharper or multi-objective field when needed. Optimisation proceeds by gradient ascent on $\nabla\Gamma$.

*Curve-based skeleton*

The skeleton of a curve-based environmental object is best fitting shape in the environment it is added to. Using an Active-Contour ("snake") formulation [KWT88], we seek a parametric curve $C$ that minimises the following composite energy:

$$E(C) = E_{\text{internal}} + E_{\text{external}} + E_{\text{shape}} + E_{\text{gradient}}, \tag{4.9}$$

In this configuration, $E_{\text{internal}}$ induces a smooth continuity of the curve by reducing the spacing of each point while reducing the curvature. Another energy, $E_{\text{external}}$, integrates the skeleton fitting function over the curve, often seen as an attractor of the points, that tries to descend the gradient to find local minima. At the same time, $E_{\text{shape}}$ applies constraints on the curve shape, which, in this case, is to target a specific length $L$. As such, the curve searches for an optimised shape given constraints in $E_{\text{shape}}$ for minimising $E_{\text{external}}$. We introduce a new term $E_{\text{gradient}}$ in the energy computation that pushes the points of the curve in the direction of the slope of the skeleton fitting function, also providing an orientation for the curve.

- Internal energy:

The internal energy introduced in [KWT88] is composed of two components imposing penalties on the local properties of the points of the curve. The first derivative forces the points along the curve to be evenly spaced by minimising the curve tension, while the second derivative restricts it from forming sharp corners, increasing the rigidity of the curve. As our aim is to represent natural elements, we rarely find sharp features and thus we use the original definition from the Snake formulation:

$$E_{\text{internal}} = \alpha_{\text{i}} \left( \int_C \left\| C'(s) \right\|^2 ds + \int_C \left\| C''(s) \right\|^2 ds \right). \tag{4.10}$$

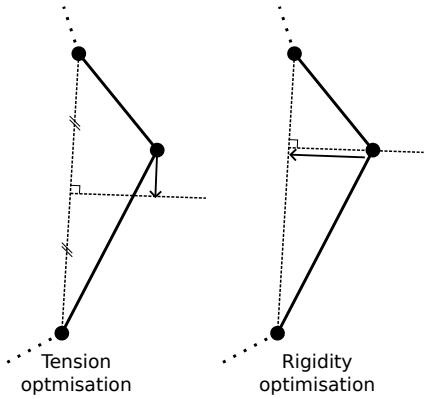



**Figure 4.21:** *Internal energy minimisation for one vertex is done by reducing the difference of length with the next and with the previous vertices (tension, left), while reducing the curvature at this point (rigidity, right).*

In Figure 4.21 we see the two different components of this energy minimisation: averaging the vector formed by the previous segment with the vector formed by the next segment, we move the vertex to a position for which the two new segments are equal in length. On the other hand, translating the vertex towards its projection minimises the curvature at this vertex.

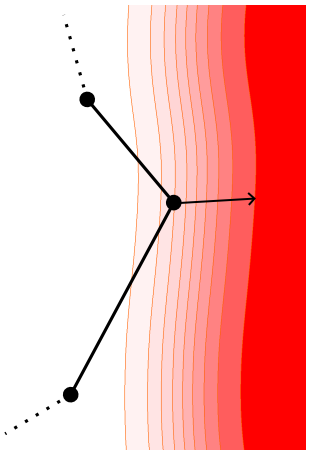


**Figure 4.22:** *The central vertex external energy is optimised by translating in the same direction as the function's gradient $\nabla\omega$.*

- External energy:

The external energy is also present in the original work. Using an external scalar field, each point of the curve is forced to follow the steepest slope of the field. The external field can be seen as an attractor for the curve (Figure 4.22).

$$E_{\text{external}} = -\alpha_{\text{e}} \int_C \Gamma(C(s))\, ds \tag{4.11}$$

The energy is minimised when the vertex is positioned at the global maximum of the scalar field.

- Shape energy:

We introduce the shape energy, an energy defined on the whole curve to apply constraints on its final shape. As natural features often have a given dimension, we propose to add a constraint on the length of the skeleton (Figure 4.23):

$$E_{\text{shape}} = \alpha_s \left( L - \int_C \|C'\| \, ds \right)^2. \tag{4.12}$$

The original Snake algorithm is biased, forcing the curve to shrink as the internal forces attract the vertices towards their neighbours. The introduction of a target length cancels this effect.

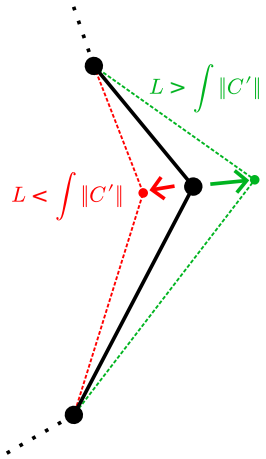



**Figure 4.23:** *The shape energy is minimised when the arc length of the curve $C$ is equal to a parameter $L$, obtainable by moving towards or in opposition to its neighbours.*

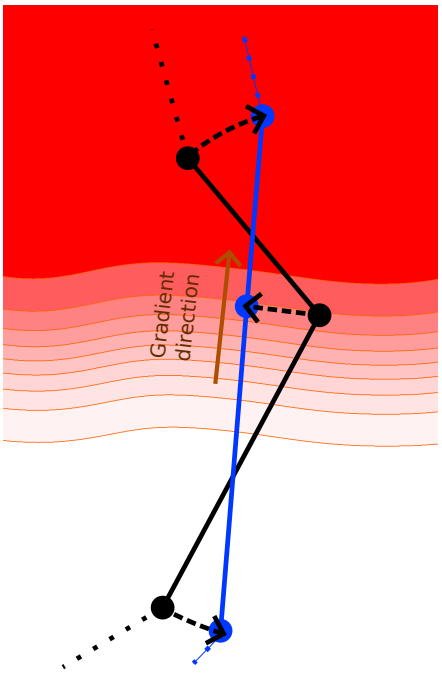



**Figure 4.24:** *The gradient energy is minimised when the curve crosses perpendicularly the isolines of the function. Black: the initial curve, Blue: a possible curve crossing the gradient obtained by displacing the vertices.*

- Gradient energy:

In our application, having information about the orientation of environmental objects may be essential, such as guiding a river in the direction of the slope. For this purpose, we introduced a gradient energy component in the formulation. The equation imposes that the direction of the curve at any point should be directed towards the gradient of the scalar field (Figure 4.24):

$$E_{\text{gradient}} = \alpha_g \int_C -\langle C'(s), \nabla\Gamma\left(C(s)\right)\rangle^2 ds. \tag{4.13}$$

As the external energy pushes points towards the lowest point of the scalar field, the gradient field restricts the gradient descent for the global curve into a specific way, which may feel more natural. Figure 4.25 presents the effect of gradient energy with three identical initial curves: a strong gradient energy factor (top) drives the curve uphill while a negative coefficient (bottom) forces the curve to avoid steep slopes.

During the optimisation process, the gradient of the scalar field is already evaluated by the external energy optimisation, so the addition of the gradient component comes without additional cost.

*Region-based skeleton*

When the skeleton is a closed boundary (island, forest patch, lagoon), we adapt Chan-Vese segmentation [CV01]: the gradient term vanishes and the external energy integrates over the enclosed domain.

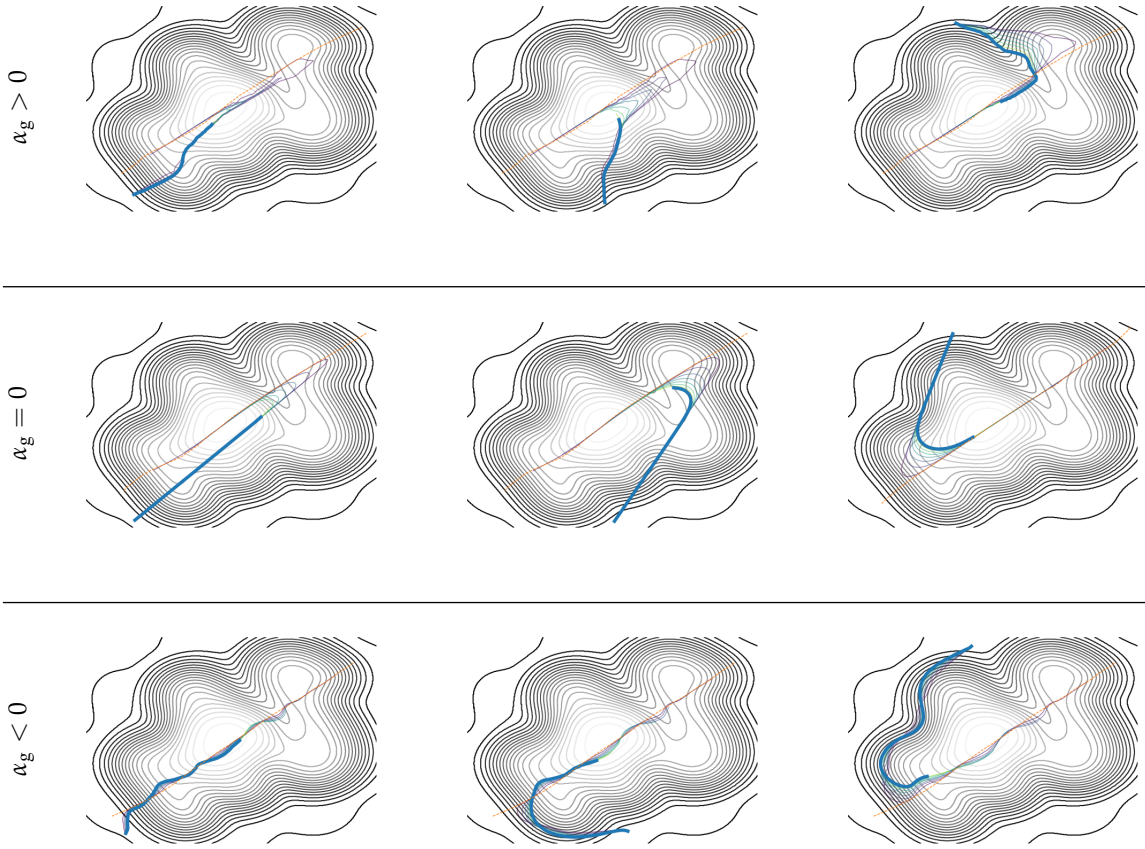



**Figure 4.25:** *Three examples of Snake optimisation from an initial curve (orange line). One vertex of the curve is fixed. Top row has a strong gradient energy coefficient $\alpha_g > 0$, middle row has no gradient energy coefficient $\alpha_g = 0$, and bottom row has negative gradient energy coefficent $\alpha_g < 0$.*

The resulting energy $E$ to minimise for a closed region whose borders are defined by the curve $C$ is then expressed as $E(C) = E_{\text{internal}} + E_{\text{external}} + E_{\text{shape}}$.

The internal energy is expressed identically as for curve-based environmental objects. The external energy, however, is modified to take into account the interior of the region instead of only the borders. We use the idea of the Chan-Vese algorithm, separating the energy value for the inside $\Omega$ and the borders $C$ of the region [CV01, Get12]:

$$E_{\text{external}} = \lambda_1 \int_{\Omega} \Gamma(s)\, ds + \lambda_2 \int_{C} \Gamma(C(s))\, ds. \tag{4.14}$$

$\lambda_1$ emphasises interior homogeneity (e.g. constant humidity for a forest), while $\lambda_2$ attracts the boundary to strong gradients (reef rim at the lagoon edge).

Finally, the shape constraint energy $E_{\text{shape}}$ targets an area $A$:

$$E_{\text{shape}} = \alpha_s \left( A - \int_{\Omega} 1\, ds \right)^2.$$

In our implementation, each environmental object's skeleton is a connected component, as we define the boundaries by the connected curve $C$, but it can be convex or concave. An infinite penalty is added for the curve's self-intersection.

At every global iteration, each object re-evaluates $\omega(C, \mathcal{E}) < T_{\text{fit}}$ with $T_{\text{fit}}$ a threshold for which the object is removed, fixed for each object class; otherwise we take advantage of the iterative nature of

the Snake algorithm and improve the shape of the skeleton by optimising their skeleton fitting function again. If the user fixed the position of vertices, we simply set $\nabla E = \vec{0}$ at the given vertices.

### Environmental material modifiers

Every instantiated environmental object contributes a local environmental modifier tuple $\mathcal{E}_o^+ = (\mathcal{H}_o^+,\ \mathcal{W}_o^+,\ \mathcal{M}_o^+)$, affecting elevation, flow, and resource fields respectively. At the end of each iteration we assemble the provisional environment:

$$\mathcal{E}^\star = \mathcal{E} + \sum_{o \in \mathcal{O}} \mathcal{E}_o^+, \tag{4.15}$$

before advection-diffusion relaxation and viability tests.

For every material layer $m \in \mathcal{M}$ (sand, limestone, nutrients, ...), we store a scalar availability field $c_m(\mathbf{p}, t)$ defined on the plane.

Each material has four transport parameters $(\rho_m, \nu_m, D_m, \mu_m)$: mass density $\rho_m$, flow-coupling factor $\nu_m$, molecular diffusivity $D_m$, and first-order decay rate $\mu_m$.

In our method, we consider that living or abiotic objects deposits (positive source) or absorbs (negative source) material, thereby mediating indirect interaction with its neighbours. For object $o$ deposing or absorbing the material $m$, we denote the net source term $S_{m,o}(\mathbf{p})$ at a given point $\mathbf{p}$:

$$S_{m,o}(\mathbf{p}) = (D_{m,o} - A_{m,o})\, G(d(\mathbf{p})), \tag{4.16}$$

where $G$ is a Gaussian kernel centred on the skeleton, $d(\mathbf{p})$ is the shortest distance of $\mathbf{p}$ to the skeleton, and $D_{m,o}$ and $A_{m,o}$ are the deposition and absorption rates.

Material advection combines gravity-driven downslope drift and water-borne transport. We approximate the velocity field:

$$\Phi(\mathbf{p}, t) = \rho_m \nabla \mathcal{H}(\mathbf{p}, t) + \nu_m \mathcal{W}(\mathbf{p}, t). \tag{4.17}$$

The environmental materials are also dispersed at a diffusion rate $D_m$, for which we use the advection-diffusion-reaction equation to evaluate the distribution over time $t$:

$$\frac{\partial c_m}{\partial t} + \Phi \cdot \nabla c_m = D_m \nabla^2 c_m - \mu_m c_m + \sum_{o \in \mathcal{O}} S_{m,o}. \tag{4.18}$$

We solve Equation (4.18) numerically using an Euler integration:

$$\begin{aligned} c_m(\mathbf{p}, t + dt) = \max\big(0, c_m(\mathbf{p}, t) \\ + dt\big(D_m \nabla^2 c_m(\mathbf{p}, t) - \mu_m c_m(\mathbf{p}, t) - \Phi(\mathbf{p}, t) \cdot \nabla c_m(\mathbf{p}, t) + \sum_{o \in \mathcal{O}} S_{m,o}(\mathbf{p})\big)\big). \end{aligned} \tag{4.19}$$

The linear decay $\mu_m$ prevents indefinite mass build-up and drives the system toward a bounded equilibrium, following the practice of EFMs [SRSS12]. Figure 4.28 shows the stability of a diffusion simulation over 200s with and without decay, with and without a flow field. A steady state is visible as the MAE (Mean Absolute Error, i.e. discrete approximation of $L^1$-norm) from one time step to another drops at $t = 50s$ in the case of a small decay, while the MAE diverges in the case without decay. The implementation of this process is straightforward, as summarized by the concise pseudo-code in Algorithm 5, which integrates advection, diffusion, decay, and source terms in an iterative loop.

An example of deposition of sand from an environmental object is provided in Figure 4.26, which is mainly affected by the water flow generated by the islands.

**Figure 4.26:** *(Left) A single island deposits sand over its region, which is displaced by the water flow. (Centre) With two adjacent islands, we see an accumulation of sand (green) in the valley they form, as we would see in nature. (Right) This phenomenon is directly due to the cancelling of their respective water flow influence, producing a region with low currents, allowing sand to settle, without any need for complex computation.*

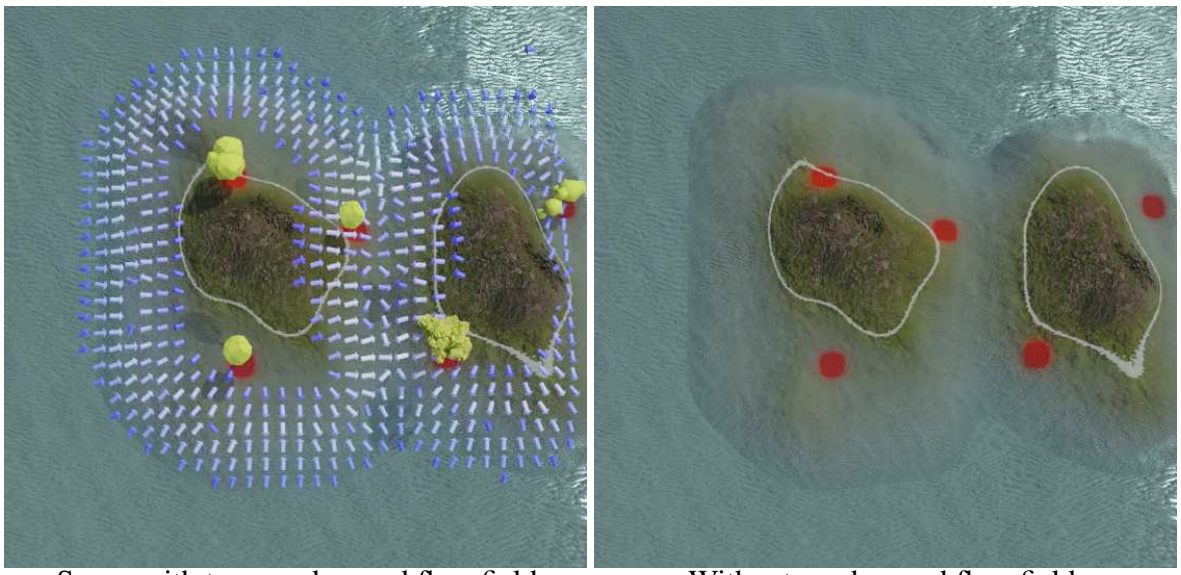
The effects of varying the different material properties of sand spread from rocks in a canyon with high water currents are presented in Figure 4.47. The special case of shade and coral calcareous deposition (reef-building material) only relies on diffusivity and decay to conserve a static modification on the surface, without the need for another framework for this unique type of environmental change (Figures 4.27 and 4.44).

**Input:** $c(\cdot)$, $\mathcal{W}$, $\mathcal{H}$, params $(\rho, \nu, D, \mu)$, sources and sinks $S(\cdot)$, step size $dt$
**Output:** Steady state environmental material

$\Phi(\mathbf{p}) \leftarrow \rho\nabla\mathcal{H}(\mathbf{p}) + \nu\mathcal{W}(\mathbf{p})$ for all $\mathbf{p}$
$\hat{c} \leftarrow c$
**do**
  $c \leftarrow \hat{c}$
  **foreach p do**
    $\hat{c}(\mathbf{p}) \leftarrow \max\{0,\ c(\mathbf{p}) + dt\,[D\nabla^2 c - \mu c - \Phi \cdot \nabla c + S(\mathbf{p})]\}$
**while** $\frac{1}{|\Omega|}\sum_{\mathbf{p}} |\hat{c}(\mathbf{p}) - c(\mathbf{p})| > \tau$ // MAE < $\tau$

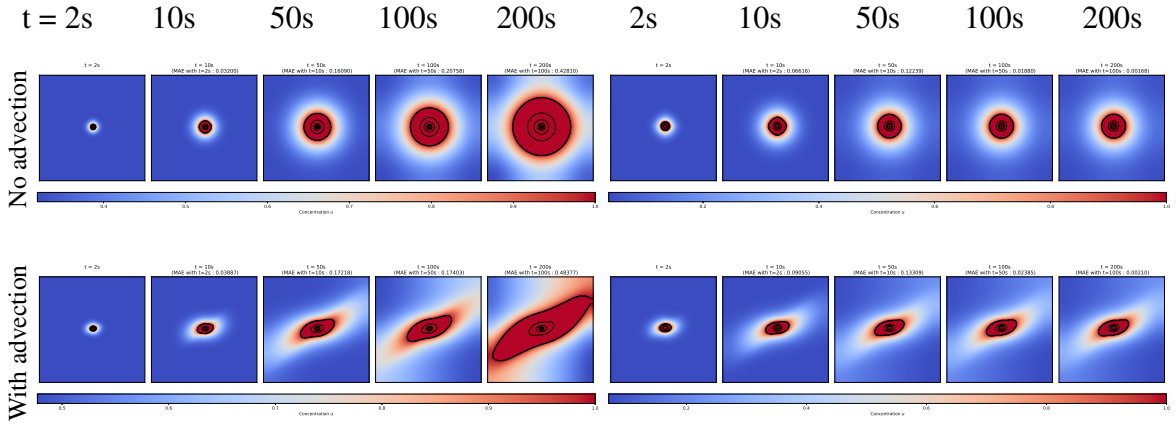
**return** $\hat{c}$

**Algorithm 5:** Advection-diffusion-reaction for a single environmental material.



Scene with tree meshes and flow field | Without meshes and flow field

**Figure 4.27:** *While we do consider shade (red) as being a environmental material in the same manner as sand or pebbles, setting the influence of water currents and slope to null makes it independent of the environment, staying at the position of the yellow environmental object that generates it.*




No decay | With decay $\mu = 0.05$

**Figure 4.28:** *With the addition of a decay term in the advection-diffusion-reaction equation, the spread of the environmental materials tends towards stability after a few iterations (right), whereas the total mass would otherwise increase indefinitely (left). Concentration values are clamped to 1 for visualization. This behavior also holds in the presence of a flow field (bottom), here with bounded diagonal vector field oriented along $x = y$. Moreover, decay reduces boundary-related instabilities; bottom left uses a zero-gradient boundary condition, which leads to unstability from the borders.*

*Approximation of distances from environmental materials.*

The use of environmental materials provides an efficient, on-demand approximation of the distance to the closest instance of an object type $\mathcal{O}$, with computational complexity linear in the number of objects in the scene. For each object type (e.g., coral, rock, reef, island, ...), we maintain a separate environmental material field $\mathcal{M}_{\text{object}}$ that evolves under diffusion-reaction only (advection disabled by setting $v = 0$ and $\rho = 0$). The object's skeleton acts as a Dirichlet boundary: for every skeleton point $\mathbf{q}$, the concentration is fixed to a constant value $c_{\text{object}}(\mathbf{q}) = C_s$ with $C_s > 0$.

Let $\Gamma$ denote the skeleton of all objects of a given type and $d = \|\mathbf{p} - \mathbf{q}\|$ with $\mathbf{q} \in \Gamma$ the closest skele-

ton point to $\mathbf{p}$. The diffusion-reaction field is updated using forward Euler steps of Equation (4.19):

$$c_{\text{object}}(\mathbf{x}, t + \Delta t) = c_{\text{object}}(\mathbf{x}, t) + \Delta t \left( D\nabla^2 c_{\text{object}}(\mathbf{x}, t) - \mu\, c_{\text{object}}(\mathbf{x}, t) \right) \tag{4.20}$$

$$c_{\text{object}}(\mathbf{q}, t) = C_s, \quad \mathbf{q} \in \Gamma. \tag{4.21}$$

To approximate the distance from a query point $\mathbf{p}_0$ to the nearest object, we perform gradient ascent on the concentration field: starting at $\mathbf{p}_0$, we follow the discrete gradient $\nabla c_{\text{object}}$ until reaching a location $\mathbf{p}$ where $c_{\text{object}}(\mathbf{p}) = C_s$. The distance estimate is then

$$d \approx \|\mathbf{p} - \mathbf{p}_0\|.$$

This setup allows direct queries for the distance to any specific type without additional preprocessing on the complete terrain, while alternative strategies such as using Euclidean distance transforms could also be applied to avoid the gradient ascent with a single lookup at the cost of heavier computation. The complete distance-query procedure is given in Algorithm 6.

**Input:** Screened field $c_{\text{object}}(\cdot)$ with Dirichlet $c_{\text{object}} = C_s$ on skeleton $\Gamma$, query point $\mathbf{p}_0$
**Output:** Approximated distance $\hat{d}$

$\mathbf{p} \leftarrow \mathbf{p}_0$
**while** $c_{object}(\mathbf{p}) < C_s - \varepsilon$ **do**
  $\mathbf{p} \leftarrow \mathbf{p} + \eta \nabla c_{\text{object}}(\mathbf{p})$ `// gradient ascent`
**return** $\hat{d} \leftarrow \|\mathbf{p} - \mathbf{p}_0\|$

**Algorithm 6:** Distance query to nearest object of a given type.

### Height modifiers

Terrain elevation is updated by blending the coarse height function functions of all objects that overlap a query point $\mathbf{p}$. We simplify the shape of environmental objects to analytical functions such as a mountain to a cone, a reef as a curved cylinder, or a coral boulder as a sphere, enabling fast, on-demand, and multi-scale computation of the altitude at any point. Because we work with a DEM, overhangs are ignored.

To preserve ordering constraints, we partition objects into three mutually exclusive groups:

- $\mathcal{G}$: grounded data (e.g. islands) defined from the absolute zero level,
- $\mathcal{A}$: altitude-relative shapes (depth-constrained corals, tidal flats),
- $\mathcal{S}$: fine surface details that sit on the current terrain, which can be seen as bumps or carves from an aerial view.

Groups $\mathcal{G}$ and $\mathcal{A}$ are combined with our smooth maximum/minimum operators as presented in the previous chapter to avoid $C^0$ discontinuities, while surface details in $\mathcal{S}$ are simply summed:

$$\mathcal{H}^+_{\mathcal{G}}(\mathbf{p}) = \text{smax}_{o\in\mathcal{G}}\, \mathcal{H}^+_o(\mathbf{p}), \tag{4.22}$$
$$\mathcal{H}^+_{\mathcal{A}}(\mathbf{p}) = \text{smin}_{o\in\mathcal{A}}\, \mathcal{H}^+_o(\mathbf{p}), \tag{4.23}$$
$$\mathcal{H}^+_{\mathcal{S}}(\mathbf{p}) = \sum_{o\in\mathcal{S}} \mathcal{H}^+_o(\mathbf{p}). \tag{4.24}$$

The final coarse elevation and its analytic gradient (for slope queries) are then given by the smooth maximum between $\mathcal{H}^+_{\mathcal{G}}$ and $\mathcal{H}^+_{\mathcal{A}}$, and the addition of $\mathcal{H}^+_{\mathcal{S}}$ (Figure 4.29):

$$\mathcal{H}^\star(\mathbf{p}) = \text{smax}\left(\mathcal{H}^+_{\mathcal{G}},\ \mathcal{H}^+_{\mathcal{A}}\right) + \mathcal{H}^+_{\mathcal{S}}. \tag{4.25}$$

The gradient $\nabla\mathcal{H}^\star$ is used by fitness rules that depend on slope, as well as the analytic water flow simulation $\mathcal{W}_{\text{simulation}}$.

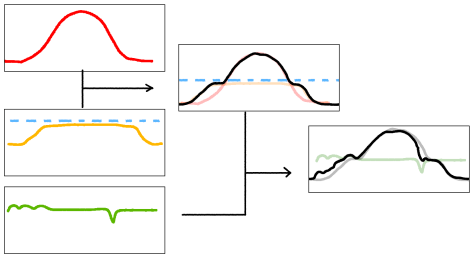


**Figure 4.29:** *The final coarse height function is a combination of the three groups of environmental objects: grounded data $\mathcal{G}$ (red) and altitude-relative shapes $\mathcal{A}$ (orange) are aggregated by the* smax *operator, then the surface elements $\mathcal{S}$ (green) are added.*

### Influence on water currents

We define the global water current as a vector field composed of three components:

$$\mathcal{W}(\mathbf{p}) = \mathcal{W}_{\text{user}}(\mathbf{p}) + \mathcal{W}_{\text{simulation}}(\mathbf{p}) + \mathcal{W}^+_{\text{objects}}(\mathbf{p}). \tag{4.26}$$

Here, $\mathcal{W}_{\text{user}}(\mathbf{p})$ is a user-defined vector field that provides direct control over the flow direction and intensity using stroke paths, as presented in the previous chapter. The term $\mathcal{W}_{\text{simulation}}(\mathbf{p})$ is an analytically defined component directly adopted from the terrain-aware wind simulation approach introduced byParis et al. [PPG*19], which we use for large water body fluid simulation. Finally, $\mathcal{W}^+_{\text{objects}}(\mathbf{p})$ represents the deformation introduced by environment objects.

The simulated component $\mathcal{W}_{\text{simulation}}$ captures how water flow is diverted and shaped by the underlying terrain. It starts from a uniform flow direction $a$ and introduces warping based on the terrain gradient evaluated at multiple spatial scales (Equation (4.29)). The resulting vector field is expressed as:

$$\mathcal{W}_{\text{simulation}}(\mathbf{p}) = \sum_{i=0}^{n} c_i \Phi_i \cdot v. \tag{4.27}$$

In this formulation, $v$ is a base velocity vector adjusted according to the terrain elevation:

$$v = a\left(1 + k_w\left|\mathcal{H}(\mathbf{p})\right|\right), \tag{4.28}$$

where $a$ is the global flow direction and velocity, $|\mathcal{H}(\mathbf{p})|$ is the vertical distance from the terrain to the water level at point $\mathbf{p}$, and $k_w$ is a scaling factor simulating the Venturi effect, which increases velocity in regions of compression.

Each $\Phi_i \cdot v$ term represents the warping of $v$ at scale $i$, defined as:

$$\Phi_i \cdot v = (1-\alpha)v + \alpha k_i \nabla\widetilde{\mathcal{H}_i}^{\perp}(\mathbf{p}) \quad \text{with} \quad \alpha = \left\|\nabla\widetilde{\mathcal{H}_i}(\mathbf{p})\right\|. \tag{4.29}$$

Here, $\widetilde{\mathcal{H}_i}$ denotes the smoothed terrain elevation at scale $i$, and $\nabla\widetilde{\mathcal{H}_i}^{\perp}(\mathbf{p})$ is the vector orthogonal to the terrain gradient, directing flow along paths that curve around elevation changes. The blending

coefficient $\alpha$ controls the interpolation between the original flow and the deviated flow, based on the steepness of the terrain. The deviation coefficient $k_i$ determines the strength of the flow deflection at each scale.

As proposed in the original method, we employ two spatial scales ($n = 2$), using Gaussian smoothing kernels with radii of 200m and 50m. The associated weights are set to $c_0 = 0.8$ and $c_1 = 0.2$, with deviation coefficients $k_0 = 30$ and $k_1 = 5$ respectively. This multi-scale formulation enables the simulation to account for both large topographic features and finer terrain details, resulting in a more nuanced and plausible flow field.

The proposed formulation offers a terrain-aware approach to modelling water currents, recognising that topography is the dominant influence on flow direction in natural environments, particularly in the absence of dynamic fluid interactions. By analysing terrain gradients at multiple spatial scales, the solver captures both broad elevation patterns and fine landform features, resulting in a plausible, nuanced flow field that naturally conforms to hills, valleys, and river paths without requiring a full physical simulation.

Unlike physics-based solvers (discussed further in Chapter 5), the analytical formulation of $\mathcal{W}_{\text{simulation}}$ exposes intuitive parameters such as flow direction, elevation sensitivity, and gradient deviation that can be directly adjusted by the user, making it especially suitable for terrain generation pipelines where user intent or artistic direction must be encoded. The solver is also computationally efficient and supports on-demand evaluation, avoiding the need for dense precomputed data and aligning well with the requirements of real-time systems, streaming terrains, and interactive editors.

While the model assumes steady-state flow and does not simulate dynamic phenomena such as rainfall, flooding, or erosion, these effects can be layered atop the static flow if needed, preserving a lightweight design that still conveys physically plausible behaviour. Finally, our system is purposefully modular: $\mathcal{W}_{\text{simulation}}$ addresses large-scale flow patterns, while small-scale variations (turbulence, vortices, and flow deviations caused by small-scale features) are handled separately by the $\mathcal{W}^{+}_{\text{objects}}$ term, allowing each component to specialise in its relevant spatial scale.

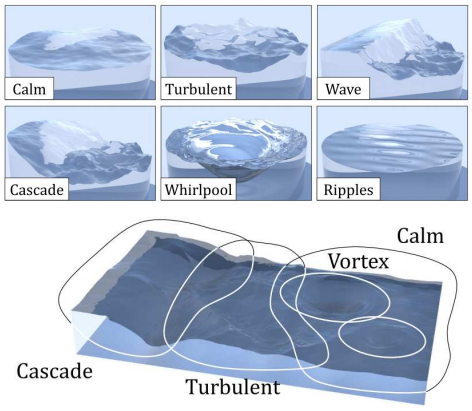



**Figure 4.30:** *Peytavie et al. uses a sparse representation of a river surface to include details at the water surface that are aggregated in a construction tree [PDG*19].*

*Vector field deformation through Kelvinlets*

$\mathcal{W}^{+}_{\text{objects}}$ is a deformation field defined as the accumulation of flow primitives. The use of analytical primitives to represent localised variations within large-scale scenes has been explored in various contexts through the edition of wind fields (Figure 4.31) [WH91] and authoring of rivers' water surface geometry (Figure 4.30) [PDG*19]. In the latter, flow primitives are primarily used to generate the geometry of the water surface, but the same primitives are also reused to produce a flow field, enabling effects such as floating debris or leaves drifting along the surface. Because geometry is sensitive to overlapping influences, the authors organise primitives into a blending tree, where merge and replace operations control how individual contributions interact.

On the other hand, Wejchert and Haumann showed that simply summing flow primitives generates velocity fields that approximate Navier-Stokes dynamics (Figure 4.31) [WH91]. Because this method affects motion rather than visible geometry, this additive approach offers a good trade-off between plausibility, user control, and computation time. In our method, we adopt a similar additive approach to combine Kelvinlet-based primitives. Our goal is to define a deformation field, and as such, summation offers an effective and intuitive way to accumulate the influence of multiple localised deformations at various scales, without requiring additional structure to resolve overlaps.

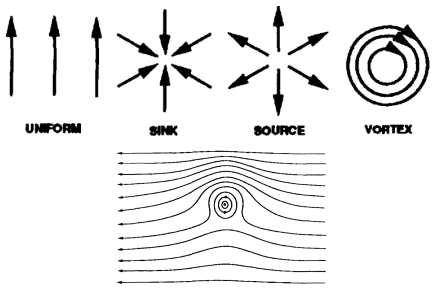



**Figure 4.31:** *Wejchert and Haumann propose to describe the flow field of the environment by a sum of primitives, resulting in a simulation of wind field controllable by the user at very small memory cost [WH91].*

Kelvinlets were introduced to computer graphics as an efficient means of producing physically plausible and interactive deformations [GJ17]. In our context, they provide a smooth and compact representation of flow alterations, ideal for integrating environmental features into vector field modelling.

Kelvinlets are based on the Kelvin solution, which is the Green's function of the Navier-Cauchy equations of linear elasticity. It describes the displacement $\mathbf{u}(\mathbf{p})$ at a point $\mathbf{p} \in \mathbb{R}^3$ caused by a force applied at $\mathbf{q}$, in an isotropic, homogeneous elastic material:

$$\mu \nabla^2 \mathbf{u} + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}) + \delta(\mathbf{p} - \mathbf{q}) \mathbf{F} = 0,$$

where $\lambda$ and $\mu$ are the Lamé parameters, with $\mu$ also known as the shear modulus, and $\mathbf{F}$ is the force vector applied at a single point $\mathbf{q}$ via the Dirac delta function $\delta(\mathbf{p} - \mathbf{q})$.

To regularise the singularity at $\mathbf{p} = \mathbf{q}$, Goes and James introduced a smoothed form known as the regularised Kelvinlets, which allow deformation effects to fall off smoothly with distance, and prevent numerical instabilities [GJ17].

Given a centre $\mathbf{q}$, an evaluation point $\mathbf{p}$, and a regularisation $\varepsilon$, we define $\mathbf{r} = \mathbf{p} - \mathbf{q}$ and the regularised distance $r_\varepsilon = \sqrt{\|\mathbf{r}\|^2 + \varepsilon^2}$.

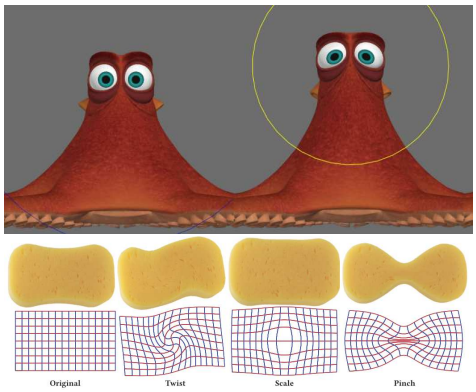



**Figure 4.32:** *Goes and James presents four types of deformations using Kelvinlets, resulting in organic pinch-like interaction with the original field. Top: a grab on a ©Disney/Pixar character. Bottom, from left to right: a twist, a scale, and a pinch operation [GJ17].*

We use the scale, grab, and twist formulations of the regularised Kelvinlets brushes (Figure 4.32), denoted as $s_\varepsilon(\mathbf{r})$, $g_\varepsilon(\mathbf{r})$, and $t_\varepsilon(\mathbf{r})$ respectively, to simulate obstruction, diversion, and vortex formation. Each environmental object may combine several instances of these primitives at different positions and orientations.:

$$\begin{aligned} s_\varepsilon(\mathbf{r}) &= (2b - a)\left(\frac{1}{r_\varepsilon^3} + \frac{1}{2r_\varepsilon^5}\right)(s\mathbf{r}), \\ g_\varepsilon(\mathbf{r}) &= \left[\frac{a-b}{r_\varepsilon}\mathbf{I} + \frac{b}{r_\varepsilon^3}\mathbf{r}\mathbf{r}^t + \frac{a\varepsilon^2}{2r_\varepsilon^3}\mathbf{I}\right]\mathbf{F}, \\ t_\varepsilon(\mathbf{r}) &= -a\left(\frac{1}{r_\varepsilon^3} + \frac{3\varepsilon^2}{2r_\varepsilon^5}\right)\mathbf{q} \times \mathbf{r}, \end{aligned} \tag{4.30}$$

with $a = \frac{1}{4\pi\mu}$ and $b = \frac{a}{4(1-v)}$, provided $\mu$ is a shear modulus and $v$ a Poisson ratio specified for each Kelvinlet, $s$ a scaling factor, and $\mathbf{F}$ the force vector of the grab operation. These values are tunable per object to simulate different material or flow resistance behaviours.

Deformations defined on curves use $\mathbf{q} = \mathbf{p}_C^*$ with $\mathbf{p}_C^*$ the closest point on the curve from the point $\mathbf{p}$ and $\mathbf{F} = C'(\mathbf{p})$. We then obtain each environmental object's flow field as the sum of all its Kelvinlet primitives distributed according to the environmental objects shape and influence: $\mathbf{u}_o(\mathbf{p}) = \sum s_\varepsilon + \sum g_\varepsilon + \sum t_\varepsilon$.

Finally, we retrieve the velocity field from the objects, with a per-object coefficient $\lambda_o$ for the strength of its flow effects (due to its size, porosity, surface roughness, etc.):

$$\mathcal{W}^+_{\text{objects}}(\mathbf{p}) = \sum_{o \in \mathcal{O}} \lambda_o \mathbf{u}_o(\mathbf{p}). \tag{4.31}$$

The different configurations of Kelvinlet compositions used for point-, curve-, and region-based objects are illustrated in Section 4.3.4, showing how each environmental object modifies the base flow according to its geometry and role in the environment.

The use of regularised Kelvinlets is well-suited to our model. They offer a compact representation thanks to their closed-form definition, which allows each deformation to be described analytically without the need for precomputed data or large simulation grids. They maintain continuity, ensuring that the resulting vector field is smooth and free of visual or numerical artefacts, which is crucial when blending influences from multiple environment objects. Their behaviour is physically plausible, as they are derived from fundamental solutions to elasticity equations, and can convincingly simulate natural flow behaviours such as deflection around obstacles or local flow obstruction. They are also computationally efficient, since their evaluation is lightweight and independent of the size or complexity of the terrain, making them particularly suitable for integration into large-scale procedural generation pipelines and real-time interactive applications.

Figure 4.34 presents a rock acting as an obstacle to the incoming water flow, which guides the flow, and more importantly the environmental materials, in a plausible direction on the side of the obstacle, with the introduction of artificial vortices at its back. At a larger scale, the same additive formulation extends to multiple interacting environment objects. Figure 4.35 shows how islands, reefs, and coastal structures jointly reshape the flow, their individual Kelvinlet-based influences blending into a coherent global current pattern. A curve-based water influence is presented in Figure 4.45 and Figure 4.46, which is a unique "grab Kelvinlet" in the direction of the curve skeleton of the canyon, and thus inserting new environmental objects downstream, where the environmental materials are transported.

To assess the plausibility of our analytical formulation, Section 4.3.4 compares our 2D projection of the velocity field around a reef with a full 3D CFD simulation, showing that the principal flow deflections and side vortices are well reproduced.

Finally, Figure 4.37 demonstrates the scalability of the approach to larger scenes such as an archipelago. Each island and reef acts as an independent Kelvinlet composition, collectively producing a plausible large-scale oceanic flow without requiring full-domain simulation.

### 4.3.5 User interactions

The user can guide the generation process. The use of simple shapes as environmental objects facilitates the edition of the simulation, as we can interactively add, remove or modify environmental objects, or focus the generation process in a restricted area. Interaction with the environmental attributes is also provided as geomorphic events, that the user can invoke during the simulation. While the direct interactions on the environmental objects are instantaneous, the geomorphic events are active over a given duration.

#### Direct interactions with the environmental objects

The interactive nature of our simulation enables the user to modify the state of the terrain by manipulating directly the environmental objects of the scene. We assume the modifications are applied

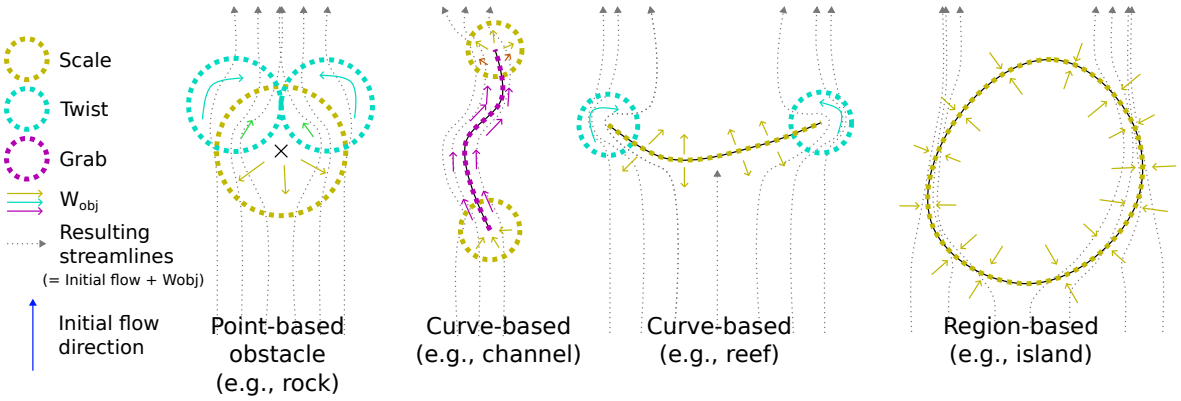



**Figure 4.33:** *Examples of point-, curve- and region-based environmental objects' Kelvinlet operators composition. The $\mathcal{W}^{+}_{objects}$ field modifies the initial water flow. Final water flow is reprented by streamlines (dashed grey arrows).*

between two iterations of the simulation.

Translating an environmental object is trivial, it simply requires evaluating the fitness function of the environmental objects at a translated position to verify that the environment is still suitable for its survival.

The deformation of environmental objects is applied on curve and region environmental objects by displacing directly the control points of the skeleton (Figure 4.38 deforming a canyon, and Figure 4.39 editing an island). As the closed or open geometry shape of the skeleton changes, we run again the skeleton fitting function optimisation through our Snake algorithm (Section 4.3.4).

Locking the vertices from the deformed skeleton edited manually by the user is achieved by cancelling the energy $E(C)_v$ from the Snake algorithm for this specific vertex $v$, disabling it from any displacement during the optimisation process. Finer control can be given by applying a coefficient on the energy gradient, such that the vertices are not locked but their displacement is only hindered.

By modifying an environmental object, environmental attributes may change, which can result in the destruction of the now incompatible environmental objects in the scene (Figure 4.38, middle). The removal of environmental object is subject to chain reactions, often visible in nature. Reducing the

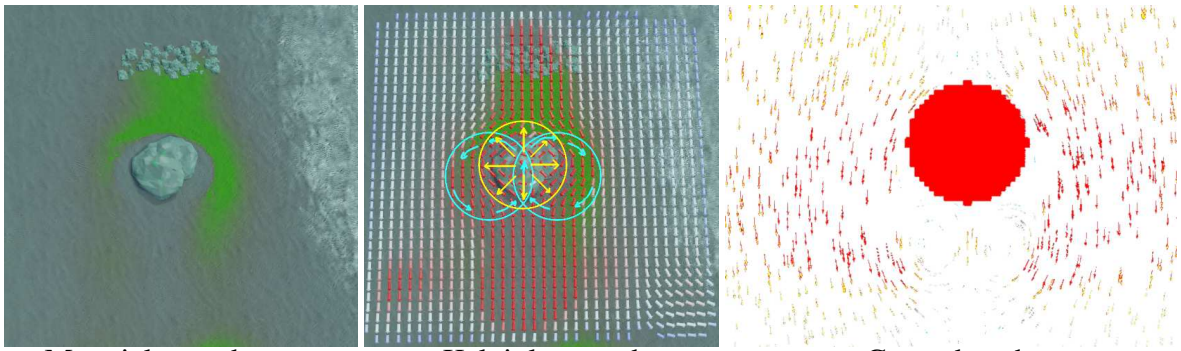



**Figure 4.34:** *A strong current is affected by a rock, visible by the deviation of sand (green) produced by smaller rocks. The flow is directly computed by the composition of three Kelvinlet objects assigned to the rock: a "scale Kelvinlet" (yellow) deflects the materials as an obstacle would; and two "twist Kelvinlets" (cyan) approximate the vortices of the flow, pushing the sand back behind the rock. Rightmost displays streamlines of CFD on a cylinder, matching approximatively our analytical solution.*

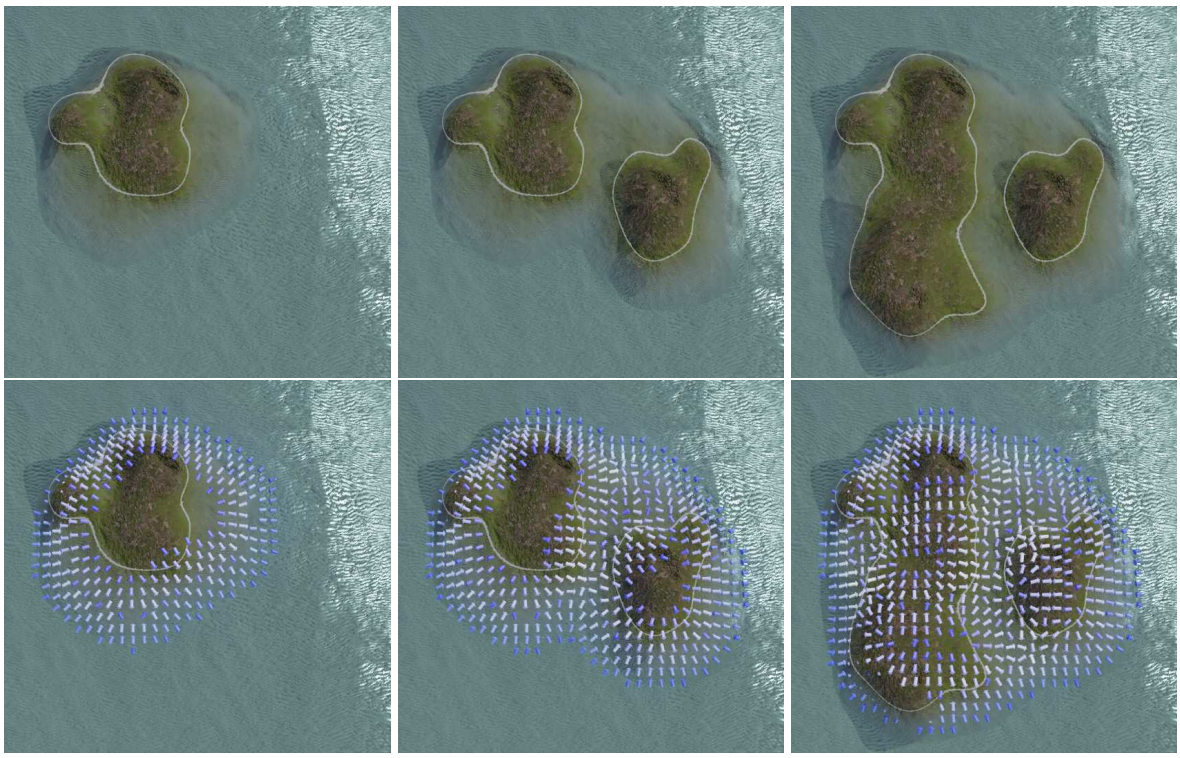
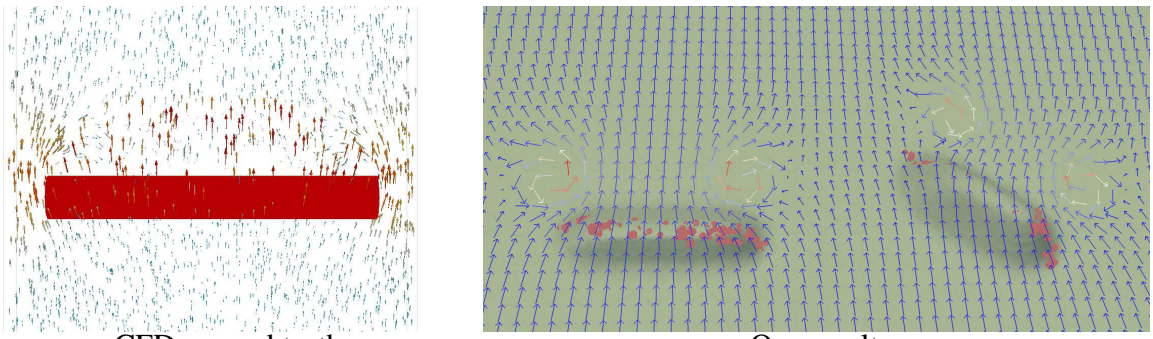


**Figure 4.35:** *Each environmental object has an influence on the water currents. Islands, defined with an area, add a force towards themselves (waves crashing on coasts) and simultaneously a force towards the ocean that acts as an obstacle flow. The resulting vector field of the scene is the sum of each environmental object's influence.*

reaction is achievable by alternating, until a steady state is reached again, a step in the environment update (Section 4.3.4) and a step of skeleton refinement for each environmental object. As a counterbalance to the removal, each deleted environmental object is reinserted if possible in the scene (Figure 4.38, right) following the process described in Section 4.3.4.

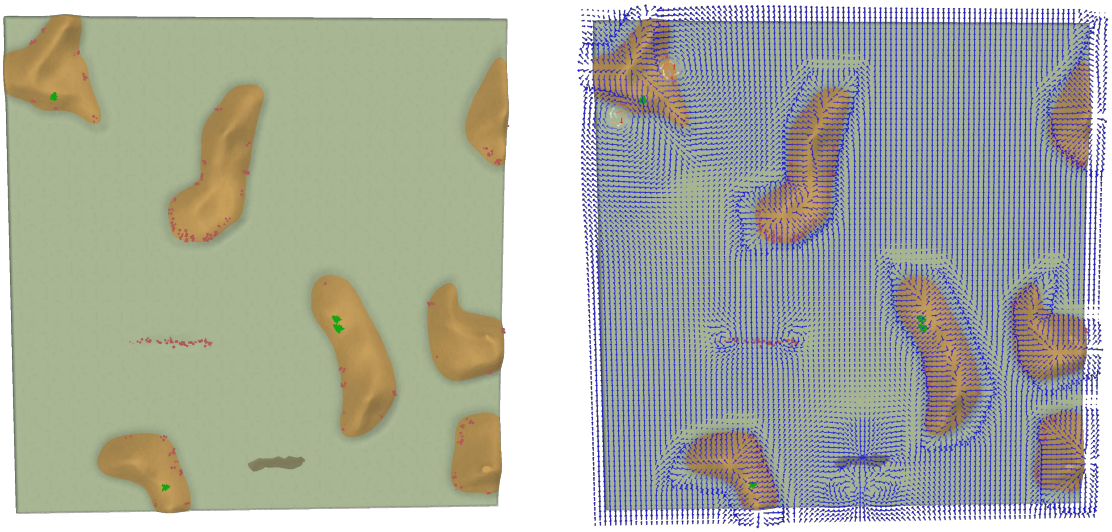
As long as a non-zero fitness function is defined in the terrain, new environmental objects can be forced by the user at any point of the simulation.

Control over the region of the terrain that should be updated is given by adjusting all fitness functions through a scalar field $\lambda : \mathbb{R}^2 \mapsto \mathbb{R}$ such that the fitness function $\omega(\mathbf{p})$ of any new environmental object is evaluated as $\omega^*(\mathbf{p}) = \lambda\mathbf{p}\omega(\mathbf{p})$. This is especially useful in the planning of robotic simulations, as



CFD ground truth Our result

**Figure 4.36:** *3D fluid simulation over a reef. Our method approximates the 2D projection of this output, with the introduction of turbulences on the side of the reefs.*



Environmental objects Resulting water flow field

**Figure 4.37:** *Archipelago with few islands, reefs and individual corals under an initial uniform water flow (going bottom to top) result in an analytical flow with each environmental object composed of Kelvinlet operators.*

we can first generate the overall shape of our terrain and then focus the generation process around the areas that may be visited by the robot, avoiding useless simulations and computational power. Figure 4.49 shows an example of colonisation of the coral polyps that we manually limited to an annulus shape.

Our water current simulation is modelled as a simple vector field. As such, the user is able to interact with it at any moment of the simulation, allowing for the death of sensitive environmental objects while guiding the simulation into a new landscape. By modifying the water currents, the user also modifies the transport rate of environmental materials at this position. The modification of currents is given as a stroke, a parametric curve $C$ for which we evaluate $\Delta\mathcal{W}_{\text{user}}(\mathbf{p})$ just as described in Chapter 3's user-driven wind velocity field construction. A simple user stroke shows the impact of a strong underwater current on coral colonies in Figure 4.40.

### Indirect interaction with environmental objects

A configuration file defines in advance the different geomorphic events that should be triggered during the simulation. This enables the simulation to reproduce the geological evolution of real-world

**Figure 4.38:** *Starting from a coral colony developed around a canyon (left), the user edits the shape of the canyon, resulting in a different configuration of the scene, killing the corals that end up too deep in the water (centre) and the development and growth of new corals at the previous location of the canyon (right).*

**Figure 4.39:** *Deformation of an environmental object is applied by displacing the control points of its outlines. All environmental objects in the scene evaluate their new fitness in the environment.*

landscapes. Multiple geomorphic events can be triggered either as sudden or continuous environmental changes. These changes have a major impact on landscape morphology. We define geomorphic events with a starting point and an ending point, such that at any time of the simulation we compute the progress of the geomorphic event as $t_e \in [0,1]$ and linearly interpolate the effects on environmental attributes.

*Water level events*

Water level changes are key geomorphic events that shape underwater landscapes. As previously submerged environmental objects become elevated above the water level, flora and fauna terrain features dry and die. Deprived of the living part of the features, everything is more affected by terrestrial erosion. By updating the value of the depth $\mathcal{D}$ evaluated in the fitness functions, any environmental object that is sensitive to depth and altitude are automatically impacted, which may

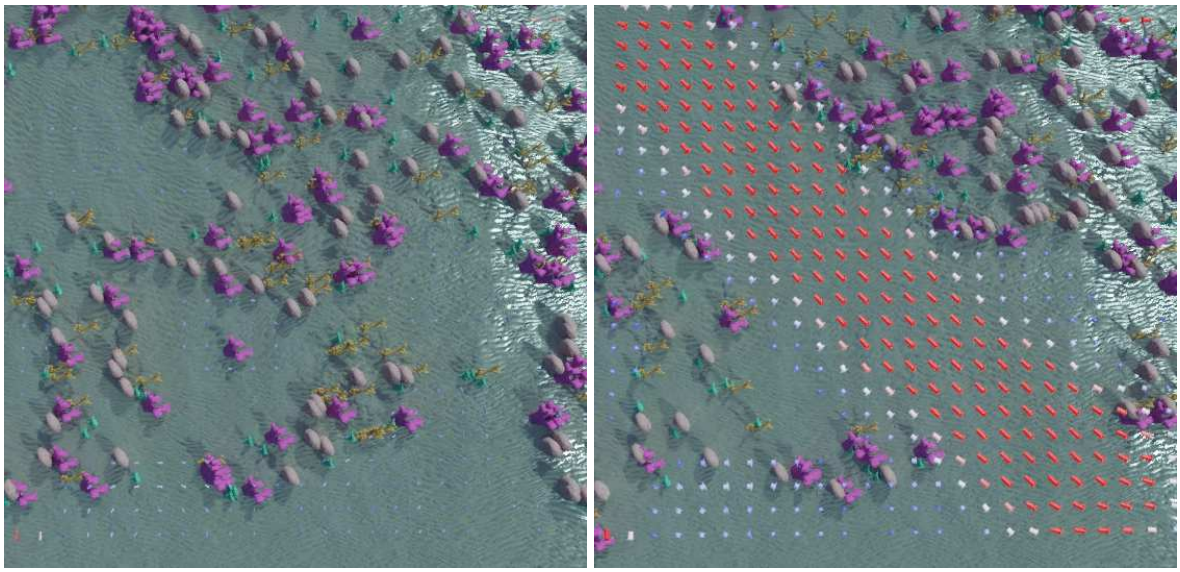


**Figure 4.40:** *A strong water current created by the user has a direct influence on the corals that are sensitive. Environmental objects that are far enough from the user stroke are unaffected.*

**Figure 4.41:** *Lowering the water level by a few metres caused most of the coral objects to satisfy $\omega \leq 0$, causing their death. Since the water level (blue) decreases slowly, new coral objects spawn progressively at a lower altitude.*

cause death (Figure 4.41). The modification of the water level is defined as:

$$\mathcal{D}(\mathbf{p}) = \mathcal{D}_0(\mathbf{p}) + \sum_{e \in \text{events}} \Delta\mathcal{D}_e t_e, \tag{4.32}$$

with $\Delta\mathcal{D}_e$ the amount of water rising or lowering during a geomorphic event. We assume a linear evolution of the water level during a geomorphic event. This allows the evaluation of depth at any point in space and time.

*Subsidence and uplift events*

Subsidence and uplift are the main geomorphic events that create or destroy islands in the long term, as presented in Chapter 3. These geomorphic events are simulated as a simple factor on the height field of the generated terrain (Figure 4.42). Subsidence is not always uniform across the terrain. As such, the user can provide a position $\mathbf{q}$ at which the subsidence is strongest, the amount of subsidence applied $\Delta\mathcal{H}_e$ and a standard deviation $\sigma$, from which we then compute, at any point in space and time of the simulation, the height of the terrain:

$$\mathcal{H}(\mathbf{p}) = \mathcal{H}_0(\mathbf{p}) \cdot \sum_{e \in \text{events}} G(\|\mathbf{p} - \mathbf{q}\|)\Delta\mathcal{H}_e t_e, \tag{4.33}$$

with $G_\sigma(x)$ the Gaussian function:

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

The Gaussian analytic formulation guarantees continuity of the changes on the environmental attributes, with smooth transitions from the epicentre of the geomorphic events as the distance increases. Any other analytical formulation could be used, but we chose the Gaussian formulation for its simplicity since the only parameter needing parametrisation is the standard deviation $\sigma$.

*Storm events*

Storms are significant factors in the geomorphology of coral reefs [VK16, OATS23] and coasts [DAG05, CWC10]. Due to the extreme wind and wave velocities, coasts are highly eroded in a short time period and the more fragile corals near the water surface are broken, possibly causing breaches in the reefs and spreading polyps in the current's direction. While there are various factors at play to understand the appearance of storms and the hydrodynamics affecting them, we simplify the model of

**Figure 4.42:** *Simulating subsidence on a part of the terrain (brown area) causes depth value to change locally, resulting in the death of coral objects that find themselves too deep to survive. Here two subsidence geomorphic events are triggered in parallel.*

storms to a single epicentre $\mathbf{q}$ with a wind velocity $v_{\text{wind}}$ and a standard deviation $\sigma$ representing the spread around the epicentre (Figure 4.43). The computation of water currents is then given as:

$$\mathcal{W}_{\text{user}}(\mathbf{p}) = \mathcal{W}^{*}_{\text{user}}(\mathbf{p}) + \sum_{e \in \text{events}} v_{\text{wind}} G(\|\mathbf{p} - \mathbf{q}\|). \tag{4.34}$$

In this case, we did not include the linear factor $t_e$ as storms are usually conserving a constant force for the time of the few weeks or months of their occurrence.

### Conclusion

Our framework can easily be extended as geomorphic event system remains similar for all geomorphic events as we see in Algorithm 7. Including higher-level simulations in the geomorphic event system can be added, such as the simulation of tectonic activity, the use of fluid dynamics for tsunami

**Figure 4.43:** *The result of a storm localised on one side of the island (red area) modifies the result of the evaluation of environmental objects around its epicentre for a short period of time. Most of the coral objects died from the geomorphic event, except for a few environmental objects less sensitive to the strength of the water currents.*

geomorphic events, the integration of human activity to alter specific environmental materials, etc...

**Input:** $\mathcal{H}, \mathcal{L}, \mathcal{W}_{\text{user}}$, events, $t$
**Output:** $\mathcal{H}, \mathcal{L}, \mathcal{W}_{\text{user}}$ after applying geomorphic events changes

**foreach** *e in events* **do**
  $t_e \leftarrow \frac{t - t_{e,\text{start}}}{t_{e,\text{end}} - t_{e,\text{start}}}$
$\mathcal{L} \leftarrow \mathcal{L} + \sum_{e \in \text{water events}} \Delta\mathcal{L} t_e$
$\mathcal{H}(\mathbf{p}) \leftarrow \mathcal{H}(\mathbf{p}) + \sum_{e \in \text{subsidence events}} \Delta\mathcal{H}_e \, G_{\sigma_e}(\|\mathbf{p} - \mathbf{q}_e\|) t_e$
$\mathcal{W}_{\text{user}}(\mathbf{p}) \leftarrow \mathcal{W}_{\text{user}}(\mathbf{p}) + \sum_{e \in \text{storm events}} v_e \, G_{\sigma_e}(\|\mathbf{p} - \mathbf{q}_e\|)$
**return** $\mathcal{H}, \mathcal{L}, \mathcal{W}_{user}$

**Algorithm 7:** Applying geomorphic events.

### 4.3.6 Output

The output of our system is a set of environmental objects stored as 2D skeletons in the plane. We deliberately stop at this sparse, symbolic level; meshing, texturing, and final rendering are left to downstream tools chosen by the end-user. For each object, we export its skeleton geometry, semantic class, and environment-modifier parameters, enough to drive terrain synthesis or placement of assets in a game engine or simulator. Figures in this chapter illustrate possible renderings generated with a mix of implicit surfaces and triangular meshes.

## 4.4 Results

We show that the method scales from islands down to coral colonies. The three examples cover different regimes: large-scale feedbacks between transport and terrain, mid-scale coupling between geology and biology, and fine-scale organisation driven by simple competition and species preferences.

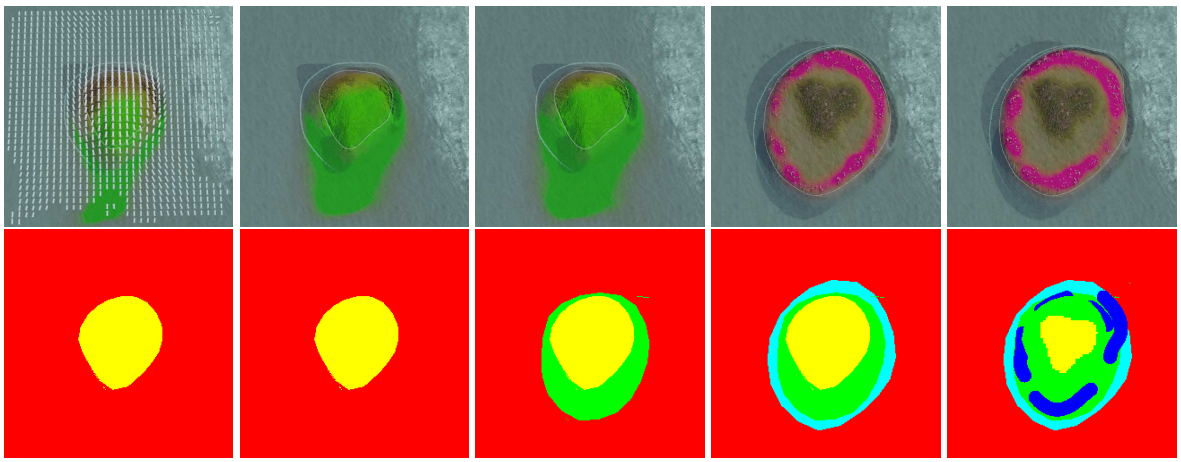


**Figure 4.44:** *Iterative construction of an island under strong prevailing wind. Sand is advected along the wind direction, forming an elongated beach ridge that shelters a low-energy lagoon. Coral colonies subsequently develop along the lagoon rim where sedimentation is minimal. Bottom: label map extracted from the environmental objects of the scene.*

### 4.4.1 Large-scale generation

In the example presented in Figure 4.44, we start with a single "Island" environmental object that diffuse sand, and a uniform wind that sets a transport direction. Sand moves according to the advection-diffusion-reaction model: advection couples sand to water flow field, diffusion smooths local variations, and decay drives a steady state.

As iterations proceed, sand drifts leeward where effective flow weakens, and it piles up into an elongated ridge along the island. We interpret that ridge as a "Beach". On its sheltered side, the flow falls below a threshold and a "Lagoon" is instantiated.

Once the lagoon is present, we instantiate "Coral colonies". Their fitness function combines a preference for moderate currents with an avoidance of sand concentration and large quantities of "Coral polyp"; the skeleton fitting function keeps them between water level $\mathcal{L}$ and 10m deep $\mathcal{L} - 10$ . Colonies deposit "Coral polyp" and "Dead coral", a calcareous material that allows the instantiation of a "Coral reef". The reef then acts as an obstacle, further calming the lagoon and closing the loop between morphology and ecology.

### 4.4.2 Medium-scale generation

In Figure 4.45 and Figure 4.46 the substrate contains a noisy "hard bedrock" material with no diffusion nor decay (constantly present). The canyon is a curve-based environmental object whose skeleton fitting function follows a path that minimises hard rock so it travels through softer corridors. Its water influence creates a flow lane along its path (Figure 4.45).

"Arches" are point-based environmental objects with fitness function maximised with current strength and bedrock abundance, making their seed placement in the canyon. The skeleton fitting function favours deeper spots, moving the seed placement in a local depth maximum, inside the canyon. An arch produces shade and deposits a "rock fragment" material. That deposit "Pebble" material, spawning "Rock" objects, which in turn act as sources for sand. Figure 4.48 adds smaller rocks at each iteration.

The result is a spawning cascade, with canyons allowing the instantiation of arches, which in turn generate rocks.

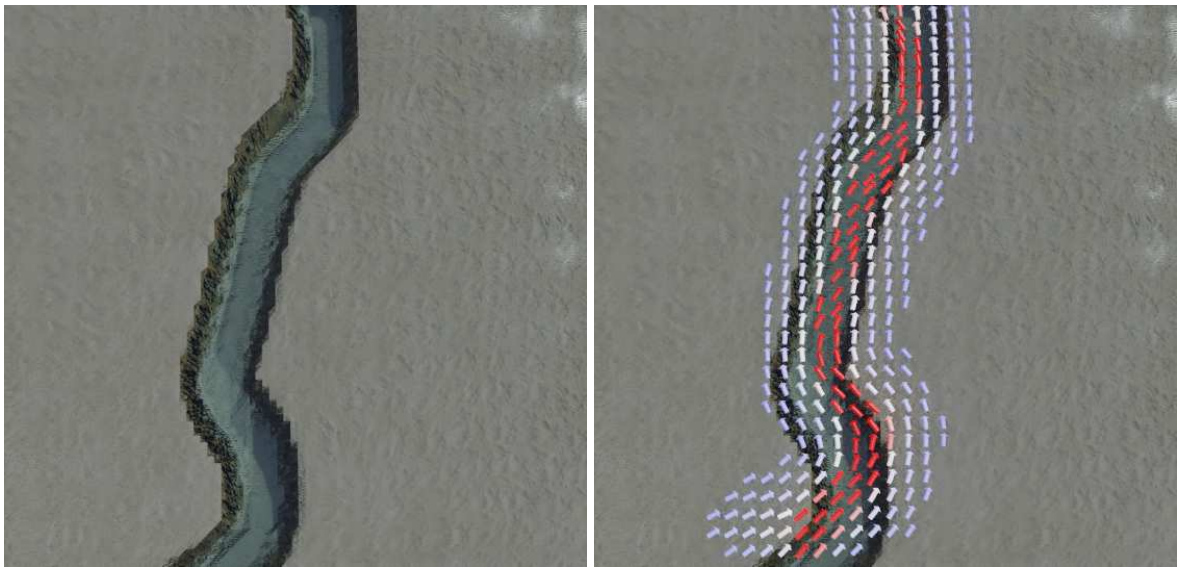


**Figure 4.45:** *Water flow oriented along the canyon curve. The analytic flow field aligns currents with the canyon's skeleton direction.*

**Figure 4.46:** *Inside view of a canyon with an arch. The arch produces shade (red) and pebbles (green), creating favourable conditions for rocks and sand deposition.*

**Figure 4.47:** *Influence of material parameters on sand distribution. From left to right, and top to bottom: reference configuration, low decay, no water transport, high diffusivity, and reversed water flow.*

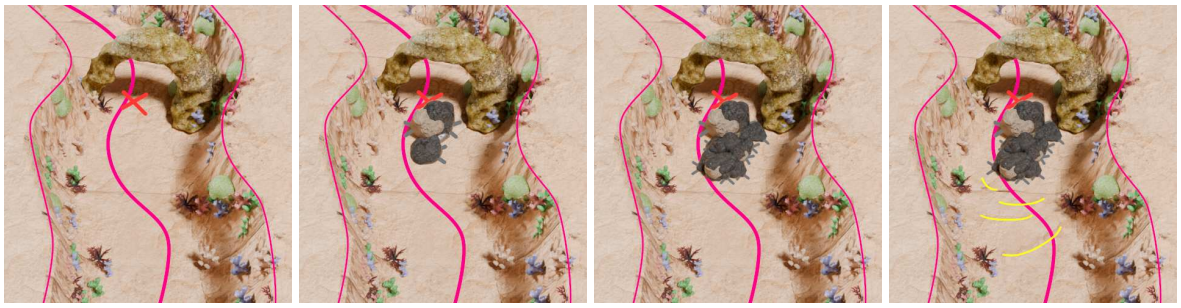


**Figure 4.48:** *Evolution of a canyon scene at different iterations. The formation of an arch triggers rock and pebble spawning, followed by sand deposition and ripple emergence at the canyon base.*

We also vary the transport model to probe sensitivity (Figure 4.47). Lower decay spreads sand farther; high diffusivity blurs contrasts; removing or reversing the water-driven advection flips directionality but remains stable. The steady-state formulation gives predictable, controllable patterns.

### 4.4.3 Small-scale generation

We present two complementary setups that use region-based environmental objects and simple material feedback to produce self-organised patterns.

In Figure 4.49, three generic coral colonies "Coral A", "Coral B", and "Coral C" are restricted to a user-drawn control field $\lambda$ in the shape of an annulus. Each species deposits generic environmental material "Polyp" and its own environmental material "Polyp A", "Polyp B", or "Polyp C". The fitness function $\omega_A = \frac{\text{Polyp}}{\text{Polyp A}}$ depends only on these materials: self-material raises fitness; other species materials reduce it. Starting from random seeds, colonies expand until their fronts meet. At the border between two colonies, none of the colonies make progression due to the amount of coral polyp specific from the other colony.

In Figure 4.50 we assign ecological preferences to different coral types via their fitness function: preferred depth band, a current range that balances supply and stress, a slope tolerance, and an aversion

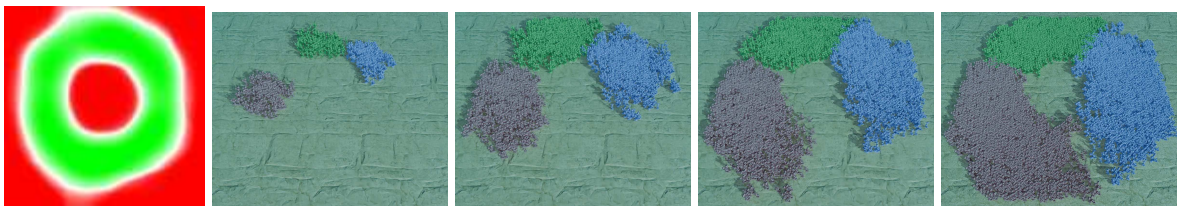


**Figure 4.49:** *Three colonies of coral (red, blue, green) restricted to an annulus (leftmost). The colonies expand until their borders meet, forming stable boundaries where no further progression occurs.*

to sand. Each coral entity finds a place that fits the distribution of species presented in Section 2.4.

### Conclusion

The proposed method aims to generate plausible landscapes using simplified versions of the evolution of an ecosystem and of the 3D representation. The biological realism of the result is highly correlated to the amount of simplification and assumptions, while the visual realism is completely dependent on the geometric functions used for the 3D modelling of the environmental objects. We propose a flexible generic approach for terrain generation where close collaboration with field experts and with graphists is needed to achieve optimal results.

## 4.5 Conclusion

This chapter introduce a procedural terrain generation method grounded in a sparse and semantic representation of landscapes. This framework, centred on the environmental objects, provides a unified abstraction for terrain and ecosystem features through parametric points, curves, and regions. By embedding expert knowledge into the fitness functions and maintaining a steady-state formulation of the environment, our approach allows both automatic simulation and interactive user guidance,

**Figure 4.50:** *Distribution of coral morphotypes from generation rules. Branching corals (purple) dominate back reefs, foliose corals (cyan) the reef crest, and massive corals (brown) extend toward the fore reef.*

while preserving coherence across multiple scales. Each environmental object locally modifies and queries environmental fields, enabling a computationally efficient yet lifelike synthesis of terrains and ecosystems.

Despite these strengths, several aspects of the current approach remain simplified and open research directions:

**Dimensional simplification.** Our framework relies on two-dimensional scalar and vector fields, which ensures efficiency and interpretability but restricts the representation to heightfield-like surfaces. Extending this model to volumetric environmental fields would enable the generation of more complex geological and biological structures such as caves, overhangs, and the internal architecture of coral reefs.

**Dynamic evolution.** The current formulation focuses on steady-state snapshots of plausible environments rather than on their continuous evolution. Incorporating dynamic processes including erosion, sediment transport, or biological growth, would bridge the gap between procedural generation and environmental simulation, allowing the depiction of terrains that change over time under natural forces.

**Shape optimisation and structural realism.** In the current implementation, the optimisation of environmental objects geometry is guided by target parameters such as a desired length or surface area, which ensures controllability but does not reflect biological or geomorphological growth patterns. Future work could investigate alternative approaches to structure generation, for instance by extracting medial-axis or skeleton structures from fitness fields to instantiate new objects, rather than relying solely on our Snake optimisation, or to be used as the initial shape for our Snake optimisation. Such biologically inspired mechanisms could produce more natural shapes and spatial organisations.

**Direct interaction and deformation.** The deformation of curve and region environmental objects is currently achieved by directly displacing the control points of their skeletons, after which the skeleton fitting function optimisation is rerun. More advanced deformation models such as sharp Kelvinlets, cage-based deformation, or Gaussian influence fields, could improve the continuity of user edits, allowing intuitive reshaping while preserving physical and ecological constraints.

**Expert knowledge requirements.** The plausibility of the results currently depends on expert-defined parameters that require manual tuning. Future work could explore learning-based estimation or inverse modelling strategies to derive these parameters automatically from real data or target exemplars, thus improving generality and ease of use.

In summary, the environmental objects framework offers a semantic, sparse, and interactive foundation for procedural terrain and ecosystem generation. By combining rule-based reasoning with analytical field interactions, it achieves a balance between plausibility, control, and performance. Its current simplifications opens the way toward richer extensions that integrate volumetric representations, evolutionary processes, and data-driven calibration.

Because the framework does not explicitly model surface geometry, the resulting terrains can appear synthetic and lack fine-scale realism. The following chapter addresses this limitation by introducing an erosion simulation method based on particle simulation. This approach generalises detachment, transport, and deposition processes across both surfacic and volumetric domains, providing a flexible and efficient mechanism for integrating erosion into procedural terrain generation.

Chapter 5

# Erosion simulation


**Figure 5.1:** *Applying shading and textures on the generated geometry produces a plausible aspect of a coast eroded by waves on a long timespan, or a desertic landscape eroded by wind, or a mountainous area flatten by thermal erosion.*

## Abstract

In this chapter, we present a novel particle-based method for simulating erosion on various terrain representations, including height fields, voxel grids, material layers, and implicit terrains. Our approach breaks down erosion into two key processes (terrain alteration and material transport) allowing for flexibility in simulation. We utilise independent particles governed by basic particle physics principles, enabling efficient parallel computation. For increased precision, a vector field can adjust particle speed, adaptable for realistic fluid simulations or user-defined control. We address material alteration in 3D terrains with a set of equations applicable across diverse models, requiring only per-particle specifications for size, density, coefficient of restitution, and sediment capacity. Our modular algorithm is versatile for real-time and offline use, suitable for both 2.5D and 3D terrains.

# Contents



# 5.1 Introduction

We finally turn to the problem of terrain evolution, focusing on erosion as a key process shaping both aerial and underwater landscapes.

A standard approach for terrain modelling is to first generate a base terrain geometry and amplify it

to improve realism. Solely using noise [MKM89, Ols04, RPP93] or to our methods proposed in our previous chapters (Chapter 3 and Chapter 4) to define the height on the input domain will most likely lack details and thus realism and feel synthetic. To tackle this problem, erosion simulation algorithms are applied to simulate thousands of years of ageing by reproducing physical phenomena (i.e. effects of the rain, wind, and running water erosion agents) affecting the terrain making it more believable [SS05, SKG*09, GGP*19].

The process of terrain alteration caused by the effect of water, air, or any other element, natural or not, over time is usually performed in three steps [NWD05]:

- **Detachment:** pieces of the ground of variable dimensions, ranging from complete ledges to grains of sand, are removed from the terrain depending on the simulated meteorological phenomenon,
- **Transport:** pieces of ground fallen from their initial position are moved to a different one such as a cornice falling down a slope or a grain of sand thrown into the air,
- **Deposition:** transported pieces of land are accumulated at a new part of the landscape.

Various phenomena cause these alterations: thermal erosion (bursting of rocks caused by expansion of water under frost, then falling of debris to the bottom of a slope), hydraulic erosion (detachment caused by the impact of water particles on surfaces and the transport of sediments by the flow of runoff), wind erosion (fine particles carried away in the wind and hit surfaces on their way, creating new fine particles which then also fly away), chemical erosion (chemical decomposition of rocks caused by rainwater or other fluids), other exceptional phenomena such as avalanches, animals, lightning, etc... modify the terrain [CCB*17, AGP*20, CEG*18, CGG*17, CJP*23].

In practice, the core idea to simulate erosion is to add or remove material from the terrain at given positions on the interface between the terrain and the fluid eroding it (e.g., air or water). Hence, the two major problems to tackle are: how to locally alter the terrain geometry for material detachment and deposition and where to perform these alterations given the properties of the environment (terrain slope, fluid density and velocity). A terrain is more than often represented in 2.5D using a 2D image called a heightmap whose greyscale values define terrain elevation. While being the major terrain representation, only a limited number of types of environments can be modelled. Indeed, natural landscapes are intrinsically 3D (overhangs, cavities or geological structures such as arches or goblins), this is particularly true for underwater environments generation. Alternate representations such as voxel grids, material layers or implicit surfaces can be used. A wide variety of methods have been proposed to simulate natural erosion phenomena on heightmaps as the partial differential equations to model erosion can be discretised and solved in 2D and the material detachment and deposition at a given point of the terrain surface can be easily performed by elevating or lowering the ground level i.e. changing locally pixel intensities. For volumetric representations, the alteration of the terrain is not as trivial. To define where to perform the erosion process, the local slope variations are more than often used combined with eroding medium information. This fluid can be simulated using particle systems, Smoothed Particle Hydrodynamics (SPH) [KBKŠ09] or approximated using a simple vector field. Proposed methods offer a specific erosion effect tailored to a single terrain representation and fluid simulation.

In this work, we propose an approach to simulate a large range of the geomorphological and meteorological phenomena present in the literature of terrain generation (including 3D and volumetric effects). We introduce a generalised algorithm performing the three stages of erosion on surface and volume representations alike, and expose very few intuitive parameters to be adjusted by the user. We propose to tackle separately the material variation and the fluid simulation. Our method relies on a particle system to simulate eroding agents, each thrown particle collides with the terrain, performs terrain alteration at the collision point and transports material along its path. Particle motion is computed using simple particle physics accounting for the medium density and particle properties (buoyancy and gravity forces). We consider each particle as independent, hence, they do not interact with each other, no collision detection or response. This simplification allows for efficient parallel computation. When more accuracy or control is needed, we propose to provide a vector field used to modify the

particle speed at each time step. The nature of this vector field is flexible, it can be computed using a more or less accurate fluid simulation (SPH, FLIP,...) or be manually defined by the user. We propose a particle-based strategy for material alteration that can be applied on surface and volumetric representations.

The main contributions presented in this chapter are:

- a generalised particle-based algorithm performing the three stages of erosion on surface and volume representations,
- decoupling the erosion system from the fluid simulation, making the process more flexible in its usage and implementation and opening the door for richer effects that can easily be produced.

## 5.2 State of the art

In this section, we first review a subset of the major simulated phenomena used to erode terrains, we then cover different fluid simulation algorithms used in procedural terrain generation. We highlight the fact that, in the literature, a specific erosion method, tailored to a given terrain representation, is proposed for given phenomena which might lead to limitation in term of terrain modeling. Indeed, changing representation costs information and precision loss.

### 5.2.1 Erosion processes

Driven by an array of natural forces and processes, erosion varies significantly across environments, from the intense carving of river valleys to the subtle reshaping of slopes in arctic regions. In this section, we present the primary types of erosion (thermal, hydraulic, and wind erosion) alongside other significant processes that contribute to landscape changes. Each erosion type not only influences distinct terrain forms but also varies in applicability depending on terrain representation in simulations. Notably, not all erosion types are easily adaptable to all forms of terrain representation due to inherent limitations in data resolution and computational methods.

#### Gravity-driven erosion

Gravity-driven erosion encompasses processes that involve the downslope movement of soil, rock, or debris due to the force of gravity. These processes, including landslides and talus slope formation, play a crucial role in reshaping landscapes by redistributing materials across slopes, valleys, and cliffs, influencing terrain stability and morphology.

*Thermal erosion*

Freeze-thaw weathering, also known as frost wedging, frost shattering, or more simply thermal erosion in Computer Graphics, is a process that occurs when water infiltrates cracks and pores within rocks and then freezes (Figure 5.2). As the water freezes, it expands and exerts pressure on the rock, causing it to fracture and break apart over time. This cycle of freezing and thawing is especially prevalent in regions with large temperature variations between day and night or between seasons, such as alpine and polar climates. Over time, freeze-thaw weathering contributes to the breakdown of large rocks into smaller fragments, creating loose rock material that can accumulate and gradually move downslope.

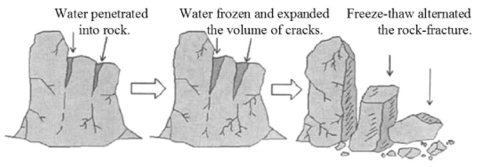



**Figure 5.2:** *Freeze-thaw process [WXF*17].*

Freeze-thaw weathering plays an important role in shaping landscapes, as it weakens rock faces and cliff edges, contributing to the formation of loose rock debris that eventually becomes part of other erosive processes, such as the development of talus slopes. This process, involving freeze-thaw cycles, can quickly fragment rock surfaces, with cumulative landscape impacts such as the formation of talus slopes observable over decades to centuries.

**Figure 5.3:** *Talus cones on north shore of Isfjord, Svalbard, Norway - Photo credit: Mark A. Wilson.*

*Talus slopes*

Talus slopes, also known as scree slopes, are accumulations of loose, angular rock debris at the base of cliffs, steep slopes, or mountainous areas. These slopes form as fragments of rock break off due to weathering processes such as freeze-thaw, and gravity pulls them downslope, where they accumulate in a cone-shaped deposit (Figure 5.3). Talus slopes are common in high-altitude or cold regions where physical weathering of rock faces is intense, and they contribute to the visual ruggedness of mountainous landscapes. Formed by the accumulation of rock debris from higher elevations, talus slopes develop gradually as materials accumulate, influencing terrain over centuries.

*Landslides*

Landslides encompass a range of processes where rock, soil, or debris moves downslope due to gravity. These events vary in scale and speed, ranging from rapid, sudden rockfalls to slow, gradual soil creep (Figure 5.4 illustrate different landslide processes). They can be triggered by factors such as heavy rainfall, seismic activity, or thawing permafrost, which destabilise slopes and initiate movement. Key types of landslides include:

- **Rockfalls:** sudden detachment of rock from steep faces, often triggered by weathering, freeze-thaw cycles, or seismic activity, leading to rapid downslope movement,
- **Soil creep:** slow, continuous downslope movement of soil and rock, caused by repeated cycles of expansion and contraction due to changes in moisture and temperature, often imperceptible over short timescales,
- **Mudflows and debris flows:** rapid flows of water-saturated soil and debris, typically triggered by heavy rainfall or snowmelt, which transport large volumes of material downslope in a short period.

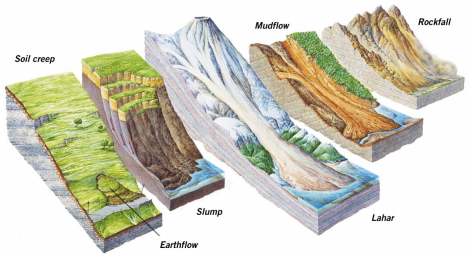



**Figure 5.4:** *Different landslide processes: soil creep, slump, lahar, mudflow, and rockfall.*

Landslides are a major force in landscape evolution, rapidly reshaping terrain and redistributing materials across slopes and valleys. Triggered by factors such as heavy rain or seismic activity, landslides can reshape landscapes almost instantaneously, though their frequency and impact may vary widely.

*Modelling implications*

Realistic simulation of these effects is achieved by applying multi-flow Computational Fluid Dynamics on the internal rock fragments or sediments, considering them as fluid particles with an evaporating viscosity [FFRL24, HD01, LD09] or as inelastic frictional spheres [Wal93], but at the cost of very high computation times.

In procedural terrain generation, however, freeze-thaw, talus slopes, and landslides are all similarly considered and generalised as "thermal erosion". The use of "thermal erosion" or "thermal weathering" in procedural generation, introduced by Musgrave et al. [MKM89], is a misnomer initially used in opposition to "hydraulic erosion". However, the effect of freeze-thaw can be seen as a reduction in the ground resistance to detachment, as the critical shear stress is highly reduced in cold weather regions, while talus slopes and landslides involve finer soil particles or the mixing of liquid fluids, introducing viscosity into the system, resulting in a quite similar output [HĎ11].

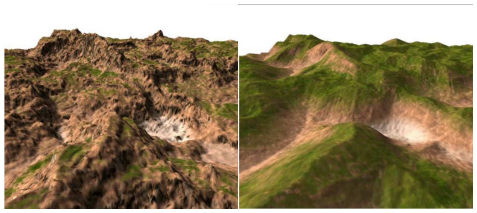


**Figure 5.5:** *Left: initial fractal terrain; right: thermal erosion simulation from [Ols04].*

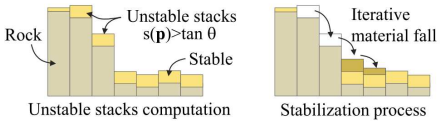



**Figure 5.6:** *Layered terrain thermal erosion simulation iteratively distributeunstable material from a cell to its neighbours until all slopes satisfy a maximum repose angle θ [GGP*19].*

Thermal erosion can be simulated by redistributing rock fragments or particles to accumulate at the base of cliffs or steep inclines, taking into account only the surface of the terrain. This effect is achieved by applying gravity-based algorithms that allow loose materials to fall and settle, forming natural slopes of debris at the base of rocky terrain [MKM89, BBFJ10]. Initially proposed for discrete height field terrain representations, thermal erosion simulation proposed by Musgrave et al. [MKM89] and improved for GPUs by Jákó and Tóth iteratively displaces a small amount of the height at each cell of the terrain and redistributes it to its direct neighbours if a repose angle is not satisfied (see Figure 5.6, used to generate the result of Figure 5.5) [JT11].

Beardall et al. adapts this definition of thermal erosion for density-voxel representations[BBFJ10], and Beneš and Forsbach for layered representations [BF01]. The importance of keeping track of scree areas allows for more detailed modelling such as the addition of geometry of rock piles illustrated in Figure 5.7, mimicking talus blocks at the bottom of a canyon [PGGM09, PPG*20].

**Figure 5.7:** *Procedural rock piles from Peytavie et al. [PGGM09].*

## Hydraulic erosion

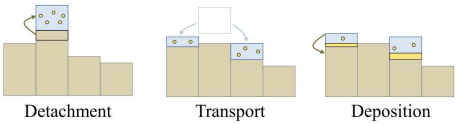



**Figure 5.9:** *Hydraulic erosion simulations uses water as a medium to move sediments from one position to another.*

Hydraulic erosion is the process by which moving water dislodges and transports soil, sediment, and rock from the Earth's surface (Figure 5.9). Occurring in multiple forms, including river-based, rainfall-induced, and coastal erosion, hydraulic erosion is driven by factors such as water velocity, volume, and surface composition.

This process plays a primary role in reshaping landforms, forming valleys, river channels, and coastlines, and significantly contributes to sediment redistribution in terrestrial and coastal environments.

### *Fluvial erosion*

Fluvial erosion is the process by which rivers and streams reshape the landscape by eroding, transporting, and depositing sediment. This phenomenon occurs as the kinetic energy of moving water exerts

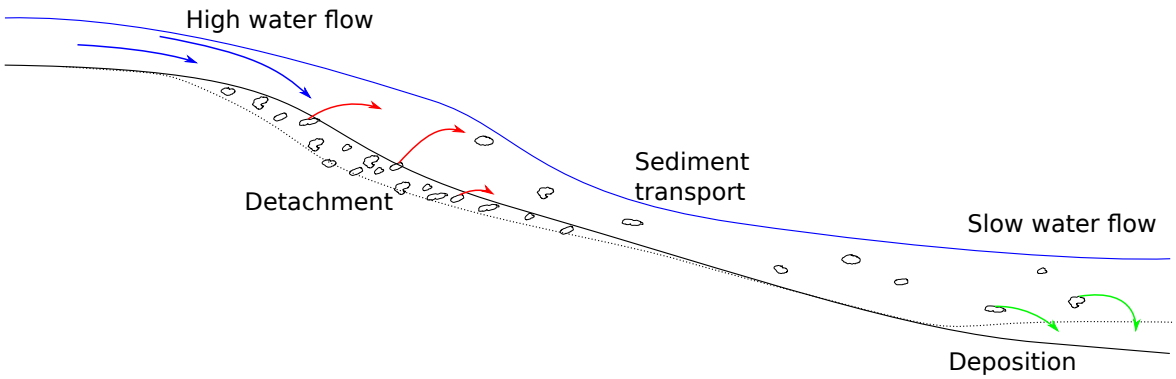



**Figure 5.8:** *Hydraulic erosion is caused by the friction of water displacing sediments on a slope from intense water flow regions to calmer downstream areas [GGP*19].*

mechanical forces on the riverbed and banks, dislodging soil, rock, and sediment particles (Figure 5.8). The intensity of fluvial erosion is influenced by factors such as water velocity, discharge (volume of water flowing per unit time), channel slope, and the composition of the riverbed and banks.

In steep, fast-flowing sections of a river, higher water velocities generate turbulent flow, which increases the river's capacity to dislodge and carry large particles. These particles, including gravel and pebbles, collide with the riverbed in a process called abrasion, grinding and wearing down the bedrock over time. Additionally, water exerts direct hydraulic pressure, especially in areas where currents are swift, prying apart rocks and sediment through hydraulic action. This is especially effective in widening channels and undercutting banks.

Fluvial erosion processes contribute to the dynamic reshaping of river channels, forming distinct landforms such as V-shaped valleys, canyons, and river meanders. Over time, rivers naturally balance their erosive energy with sediment transport and deposition, forming floodplains where sediment is deposited during seasonal overflows. In meandering rivers, erosion typically occurs on the outer curves of bends, where flow velocity is highest, while sediment deposition takes place on the inner curves, forming point bars. This continual interaction between erosion and deposition drives the lateral migration of meanders, altering the river's course across the landscape. The river's competence, or its ability to transport particles of a certain size, depends on the flow's velocity and discharge. During periods of high flow, such as after heavy rainfall or snowmelt, rivers gain greater erosive power, enabling them to transport larger particles and increase their erosion rates. River systems reshape landscapes methodically over years to millennia, deeply engraving river paths and altering regional topographies.

*Rainfalls*

Rainfall-induced hydraulic erosion begins as raindrops strike exposed soil surfaces, causing splash erosion, where particles are dislodged and displaced by the impact of individual raindrops, creating tiny craters [VHLL05, LFW*24]. As rainfall accumulates and flows overland, it transitions into sheet erosion, where a thin layer of water, known as sheet flow, moves across the land surface. This process is often intensified on sloped terrain, where the water gains momentum as it descends, picking up and carrying loose particles downslope. Sheet erosion can remove a uniform layer of soil across a large area, gradually depleting soil fertility and weakening the structure of the soil surface.

**Figure 5.10:** *Rill and gully erosion along the Chilcotin River [GSP*10].*

On steep or prolonged slopes, sheet flow may concentrate into small channels, initiating rill erosion [Gat00]. Rills are narrow, shallow channels that cut into the soil as water flow converges, carving miniature stream-like paths down the slope visible in Figure 5.10. As rills deepen and widen, they can evolve into larger channels in a process called gully erosion, where channels become deep and wide enough that normal agricultural or natural processes cannot easily repair them. Gullies disrupt the landscape, fragmenting ecosystems, and accelerating the removal of topsoil.

The extent of erosion depends on factors such as rainfall intensity and duration, soil type, vegetation cover, and slope steepness. Sandy soils, for instance, are prone to erosion due to their low cohesion, whereas clay-rich soils, while more resistant to initial splash erosion, are highly susceptible to rill and gully formation once water begins to concentrate. Splash erosion and sheet erosion can cause significant soil degradation and landscape changes over months to decades.

*Modelling implications*

Hydraulic erosion simulation dominates procedural terrain work because it produces large, coherent landforms and can be simulated efficiently with surface-based models using simple rainfall fields.

In procedural terrain generation, fluvial erosion is focused on the generation of rivers and cascades defined as feature-curves [EPCV15]. First, the path of the river is drawn either by the user [HGA*10] or through simulation [Par23]. Second, the shape of the rivers' beds are modelled [GGG*13]. It is then finally possible to provide a geometric representation of the water surface taking into account flow rate, depth, obstacles, and user-defined details [PDG*19].

The computation of rivers' properties is defined by understanding the flow of water, which is directly related to the amount of rainfall at each point of the terrain [KMN88], then approximated by simplified Shallow Water modelling [MDH07], but is mainly achieved in acceptable computation time by considering all rivers as a drainage network, an oriented graph of rivers as introduced by Roudier et al. [RPP93]. Recent works focus on the acceleration of the computation of these graphs and the drainage area associated at each point [CBC*16, SPF*23].

For most proposed methods, the amount of water falling at each point of the terrain is taken into account. However, this strategy is divided among works that consider that rain falls uniformly on the whole terrain or using a random noise function, that rain is more present in regions of high altitude [NWD05], or uses a more accurate weather simulation to define areas more prone to rainfall [PMG*22].

As the motion of water is easier to model on the surface of the terrain, almost all algorithms consider the terrain with a 2.5D representation. On a large scale, this assumption is beneficial but limits their scope to this type of representation, making them unavailable to volumetric models. Particle-based methods are then proposed to overcome this issue for smaller-scale terrains, using 3D Eulerian fluid simulations on voxel terrain representations [BTHB06] or Lagrangian simulations on TIN terrains [KBKŠ09], at the expense of much higher computational time.

### Chemical erosion and caves

Chemical erosion, also known as chemical weathering, involves the chemical reactions between water and rock that lead to the dissolution and alteration of minerals within the rock. This process is especially significant in river and coastal environments, where water, often containing dissolved carbon dioxide, interacts with rocks such as limestone and dolostone to form weak carbonic acid. This acid reacts with carbonate minerals, gradually breaking down the rock structure through a process called dissolution. Over time, this process weakens rock formations, making them more susceptible to mechanical erosion by water.

Chemical erosion is particularly influential in the formation of karst landscapes, where the dissolution of carbonate rocks creates unique features such as caves, sinkholes, and underground drainage systems. As acidic water seeps into fractures within the rock, it slowly enlarges these cracks, leading to the

creation of hollow spaces and intricate cave networks. In addition to carbonic acid, other dissolved ions, such as sulphur and organic acids, also contribute to the chemical weathering of rocks in various environments. The dissolution of soluble rocks such as limestone forms extensive karst landscapes, including networks of underground caves, over thousands to millions of years.

**Figure 5.11:** *Akun Island basalt sea cave - Photo credit: Steve Hillebrand.*

Sea caves are formed through the mechanical force of hydraulic action as waves continuously impact the shore. These sea caves develop along coastlines with cliffs, where wave energy focuses on weak points in the rock, such as fractures or softer rock layers. Over time, the pressure from waves and tides pries apart rock fragments, carving out hollow spaces within the cliffside or even a whole arch as presented in Figure 5.11. Cave formation by the erosive power of waves against rock cliffs containing zones of weakness is observed over hundreds to thousands of years.

In desert environments, caves can also be form through aeolian erosion, where wind-driven sand particles abrade rock surfaces. These aeolian caves are typically found in sandstone or other softer rocks, where strong, consistent winds gradually wear away the rock. Unlike chemical or hydraulic caves, aeolian caves are usually shallower and smaller, as the erosive force of wind is less powerful than water or acid-driven processes. However, these caves add unique features to desert landscapes, creating sheltered hollows that sometimes serve as habitats for desert wildlife. Created by wind erosion primarily in softer rock formations, these caves can develop over centuries, depending on the consistency and strength of wind.

*Modelling implications*

In procedural terrain generation, chemical erosion has been rarely studied, as this interaction mostly occurs in shallow to deep waters and requires a volumetric representation, while most terrain generation algorithms are tailored to aerial landscapes and are limited by 2.5D representations. However, as visible in Figure 5.12, using 3D cellular automata provides an explicit representation of this effect [MKL20, WCMT07] (and can be extended to a simulation of coastal erosion [Haw14]).

**Figure 5.12:** *Wojtan et al. simulates chemical erosion on a regular grid [WCMT07].*

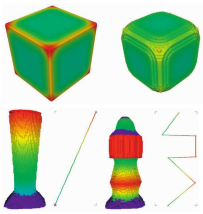


**Figure 5.13:** *Top: spheroidal weathering proposed by Beardall et al. on voxel grids iteratively removes most exposed voxels (red) [BBFJ10]. Bottom: modulating exposition with a resistance map (red/green curve) creates specific weathering patterns.*

An approximation of this phenomenon is proposed in Beardall et al. on voxel grids by considering that the most vulnerable voxels of the terrain are the ones that have the most neighbours with air voxels, in a very similar manner than the computation of voxel-grid ambient occlusion (Figure 5.13) [BFO*07, BBFJ10]. The simulation of sea caves for implicit terrains has been proposed by Paris et al. by considering a resistance function in the bedrock, usually a function of height, and adding spherical holes in the construction tree at the least resistant areas of the terrain [PGP*18]. The inverse percolation method enables the simulation to dig longer cavities, resembling coastal karsts.

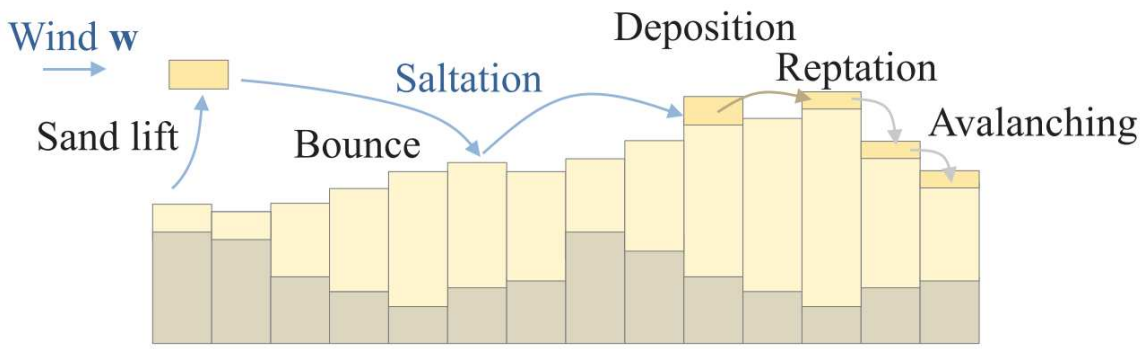



**Figure 5.14:** *Wind erosion includes the lifting of the sand, the transport through the wind, and its deposition [PPG*19].*

## Wind erosion

Aeolian erosion, also known as wind erosion, is the process by which wind transports, dislodges, and deposits particles of soil, sand, and rock, particularly in arid and semi-arid regions where vegetation is sparse. This form of erosion is driven by the movement of loose particles across the surface and the abrasive impact of wind-driven sand. Aeolian erosion leads to distinctive landforms that shape desert landscapes, coastal areas, and regions downwind of deserts. Wind erosion typically occurs through three main processes illustrated in Figure 5.14: deflation (removal of fine particles), saltation (bouncing movement of medium-sized particles), and abrasion (wearing down of rock surfaces by wind-driven particles).

*Sand dunes*

One such feature is sand dunes, mounds or ridges of sand formed by the accumulation of wind-transported particles (Figure 5.15). As wind moves sand across a surface, obstacles such as rocks or vegetation slow down particles, causing them to settle and accumulate into dunes. The shape and type of dune depend on wind direction, strength, sand availability, and landscape features [PPG*19]. Common types of dunes include barchan dunes, which are crescent-shaped with tips pointing downwind; transverse dunes, long ridges perpendicular to the wind; and star dunes, which have multiple arms formed by shifting winds. These dunes are dynamic, migrating over time as wind continues to move sand particles, contributing to the constantly evolving desert landscape.

**Figure 5.15:** *Sand dunes [SM06].*

**Figure 5.16:** *A yardang near Meadow, Texas - Photo credit: U.S. Departement of Agriculture.*

*Yardangs*

Yardangs are elongated ridges formed where softer material is eroded faster than more resistant rock. The wind-carved ridges align with the prevailing wind direction, creating streamlined shapes with steep sides facing into the wind and gentler slopes on the leeward side as visible in Figure 5.16. Yardangs vary widely in size, from small ridges a few metres long to massive formations stretching for kilometres, as seen in desert regions of Iran and Egypt. They illustrate the power of wind erosion in shaping landscapes over long periods, with their formation largely dependent on rock hardness, wind intensity, and particle size. These streamlined rock formations are carved by persistent wind eroding softer material faster than harder material, typically forming over centuries.

*Ventifacts*

Ventifacts are rocks that have been shaped and polished by the abrasive action of wind-driven sand. These rocks typically exhibit flat, smooth surfaces that are often oriented in the same direction as the prevailing winds. Their formation occurs predominantly in arid environments where loose sand is available to act as a natural sandblasting tool. This erosive process highlights the power of wind as a geomorphic agent, capable of sculpting rocks into distinctive shapes over time as illustrated in Figure 5.17. Rocks shaped by wind-driven sand, ventifacts can take decades to centuries to form, depending on local wind conditions and rock exposure.

**Figure 5.17:** *Ventifact at White Desert National Park, Egypt - Photo credit: Christine Schultz.*

*Modelling implications*

Wind erosion shifts material through wind force, notably impacting areas with fine surface particles such as deserts. Sand dune generation was modelled on discrete height fields by mimicking sand's wind-driven trajectory and using thermal erosion to correct the slope [RB04]. These algorithms consider the wind coming from a unique direction, but introducing variations in the wind flow simulation, new dune formations emerge. Paris et al. adapt this method for layer-based representations in which the layers of sand and the layers of bedrock are defined, includes the definition of a wind simulation consisting of warping a uniform flow with the terrain slopes [PPG*19]. This allows for the fast generation of large-scale desertic landscapes. More recently, Rosset et al. associat a fine-resolution 3D fluid simulation for wind modelling, enabling the simulation of dune formation at a small scale with high fidelity, albeit at a higher computational cost due to fluid dynamics [RDBC24].

While most aeolian terrain models account for the effect of wind on sand transport, far fewer simulate the feedback of transported material abrading obstacles. Early voxel-based approaches applied analytical heuristics such as spheroidal weathering [BFO*07] and curvature-driven removal [BBFJ10] to progressively reshape volumetric rock, in the spirit of exposure-based methods on voxels and implicit fields [PGP*19]. By contrast, Krs et al. adopt a physics-inspired formulation [KHM*20]: polygonal meshes are converted into a layered-terrain representation, erode its layer stacks when flow stresses exceed material resistance, advect the detached mass as particles within an SPH fluid, and finally redeposit it into the volume. This particle-fluid coupling parallels hydraulic erosion work [KBKŠ09], but targets wind-driven abrasion and deposition. Despite their different strategies

(analytic voxel criteria versus particle-fluid transport) all three approaches ultimately rely on volumetric discretizations to modify geometry. Taken together, they outline a design space where voxel analytics offer controllable stylization, while physics-based enables obstacle-sand interactions.

### Glacial erosion

Glacial erosion is a dominant geomorphic force in cold regions, characterised by massive ice sheets and glaciers that sculpt the Earth's surface over thousands of years. As glaciers advance and retreat, they transform landscapes through two primary mechanisms: plucking and abrasion.

Plucking occurs when glaciers freeze to the bedrock and, as they move, exert tremendous force that tears away blocks of rock. This process is facilitated by water that infiltrates cracks in the rock, freezes, and expands, helping to dislodge rock pieces as the glacier flows. Abrasion happens concurrently as rocks and debris embedded in the ice act as sandpaper, grinding and smoothing the rock surface beneath the glacier. This abrasion not only polishes the rock but also carves deep grooves and striations (linear marks that align with the direction of ice movement).

These erosive processes produce distinctive glacial landforms. U-shaped valleys, unlike the V-shaped valleys formed by river erosion, are wide and deep with a flat bottom, shaped by the broad, sweeping movement of glacier ice. Fjords are deep, narrow inlets of the sea set between high cliffs, created by the deepening of U-shaped valleys by glaciers that then become submerged as sea levels rise. Cirques are amphitheatre-like hollows situated at the heads of glacial valleys, formed by the erosion caused by the rotational movement of ice within them. Moraines are accumulations of dirt and rocks scraped up and deposited by moving glaciers, typically appearing as ridges along the sides of glaciers or as mounds of debris left behind after a glacier retreats.

The timescale of glacial processes spans over thousands to millions of years, allowing glaciers to leave a lasting impact on the landscape.

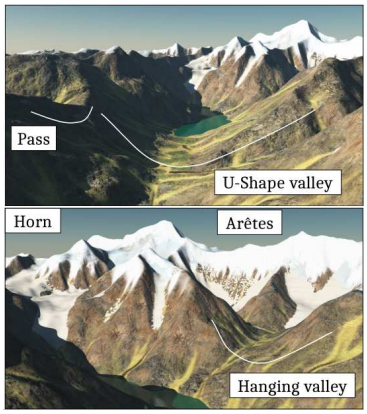



**Figure 5.18:** *Glacial erosion from Cordonnier et al. erodes smoothly valleys, shaping the terrain with natural looking valleys and mountain peaks [CJP*23].*

Few prior works focused their effort on the understanding and modelling of glacial erosion phenomena. In these conditions, Argudo et al. consider the glacier as a fluid with no inertia, called the Shallow-Ice Approximation, taking only the terrain surface gradient and ice thickness to deduce the velocity of an ice column [AGP*20]. The shear stress applied beneath the ice sheet is directly computed by the estimation of the ice column and its velocity. This method results in V-shaped valleys, but when integrating more factors into the simulation, such as debris flow, fluvial erosion and talus slopes, the improved method presented by Cordonnier et al. generates a large variety of features present in glacial landscapes as presented in Figure 5.18 [CJP*23].

### Volcanic activity

Volcanic activity is a powerful tectonic process that alters landscapes by creating new landforms through the eruption of magma from the Earth's mantle [RQT*13]. Volcanoes form primarily at tectonic boundary zones, either where plates diverge, allowing magma to rise and fill the gaps, or where one plate subducts beneath another, melting into magma due to high pressure and temperature.

The surface changes caused by volcanic activity are substantial and varied. When a volcano erupts, it

deposits layers of lava that solidify into new rock formations, gradually building the volcanic cones and mountainous structures that are characteristic of volcanic islands and mountain ranges [Woo03].

Pyroclastic flows are another aspect of volcanic activity that dramatically changes the terrain. These fast-moving currents of hot gas and volcanic matter can race down the sides of a volcano, destroying everything in their path and laying down thick deposits that can form natural barriers or change drainage patterns by damming rivers and creating lakes almost instantaneously (see Figure 5.19).

**Figure 5.19:** *Piton de la Fournaise, Réunion Island, in eruption - Photo credit: Richard Roquejoffre.*

Volcanic activity also leads to the formation of calderas, large depressions that occur when the summit of a volcano collapses into the emptied magma chamber below after an eruption. Calderas can be several kilometres in diameter and significantly alter the local landscape.

Although lava and pyroclastic flows have been studied in Computer Graphics, resulting in realistic eruption simulations in recent works, these simulations are not used in the case of procedural terrain modelling but focused on rendering and animation [ZKL*17, CBL*09, SAC*99, Las23].

### Biological processes

Biological activity both accelerates and mitigates erosion. On land, burrowing fauna (e.g., rodents, moles, earthworms) loosen and reorganise soil, increasing porosity and altering infiltration; this can expose fine material to wind and runoff. Grazers reduce vegetative cover and compact the soil through trampling, boosting overland flow and sediment detachment. At very small spatial scales, animal trails and human desire paths locally concentrate shear and degrade soil structure, but remain understudied compared to other drivers [CEG*18, JKA*19, AAR*24, ECC*21]. In coastal and marine settings, bioeroders such as urchins, molluscs, and parrotfish mechanically abrade rock and coral frameworks, gradually lowering relief and reshaping habitat complexity.

Vegetation generally counteracts erosion through three coupled mechanisms. First, roots stabilise soil: deep, woody systems anchor slopes and resist mass wasting, while fibrous grass mats bind topsoil and limit detachment. Second, canopies intercept rainfall, dissipating raindrop energy and reducing splash erosion. Third, cover increases surface roughness and promotes infiltration while transpiration lowers antecedent moisture, collectively reducing runoff volume and critical shear stress. Together, these processes reduce both detachment rate and the frequency of erosive events and should be taken into account for realistically shaping virtual landscapes. However, these factors are seldom integrated in erosion simulations methods [CGG*17].

### Conclusion

In conclusion, various erosion processes occur on terrains over very different timescales. However, each procedural generation algorithm is mimicking a small number of these processes at once.

The mapping in Table 5.1 highlights a clear imbalance in existing work: most erosion models are confined to height field representations, while layered, voxel, and implicit approaches remain sparsely explored. This uneven distribution reveals a methodological limitation rather than a physical one as current algorithms are tightly coupled to the data structures they operate on, making them difficult to generalise across terrain types. As a result, many process-representation combinations, such as aeolian or chemical erosion on volumetric terrains, remain absent. Our method directly addresses these gaps by expressing erosion through its three fundamental stages using a unified particle-based formulation: the detachment of matter, its transport, and the deposition in smaller sediments. Because particles interact with terrain through local sampling and deposition functions rather than explicit grid

| Process | Height field | Layered | Voxels | 3D implicit | Specific representation |
|---|---|---|---|---|---|
| Thermal | [MKM89] | [BF01] | [BBFJ10] | | |
| Landslide | [CEG*18] | | | | [HĎ11] |
| Hydraulic | [KMN88, GGG*13] | [ŠBBK08] | [BTHB06] | | [KBKŠ09] |
| Chemical | | | [WCMT07] | [PGP*19] | |
| Aeolian | [PPG*19] | | | | |
| Glacial | [CJP*23, AGP*20] | | | | |
| Volcanic | | | | | |
| Biological | [ECC*21, AAR*24] | | | | |

Table 5.1: Mapping of erosion processes with terrain representations. Most works target height fields; volumetric and implicit approaches exist but are rarer and typically computationally demanding.

updates, the same erosion mechanism can be applied consistently to height fields, layered terrains, voxel grids, or implicit surfaces. This decoupling between the erosion logic and the underlying representation allows the simulation of processes that were previously constrained to specific data structures, effectively filling the blank cells of the table and enabling cross-representation erosion modelling.

The displacement is guided by the environment in which the erosion occurs. It may be only the force of gravity, the trajectories of glaciers, the water flow, or the wind. Water and wind forces require fluid simulations to simulate accurately.

### 5.2.2 Fluid simulations

Fluid simulation constitutes an important field in Computer Graphics and in terrain generation, providing a physical basis for modelling water behaviour and dynamics for real-time applications and offline rendering [WXL*24]. The mathematical foundations of fluid simulations lie in the Navier-Stokes equations, or in the case of hydrodynamics, the incompressible Navier-Stokes equations. An accurate computation of these dynamics implies an expensive computational cost; thus, various solver variants have been proposed, relying on grids, particles, or hybrid frameworks, each balancing trade-offs between simulation stability, dissipation control, computational scalability, and user control. In this section we propose an overview of different solvers and their characteristics.

#### Governing equations

The governing equations for fluid simulation are rooted in the conservation laws of mass and momentum for an incompressible Newtonian fluid. In three dimensions, these are expressed by the incompressible Navier-Stokes equations, which consist of the momentum equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{g}, \tag{5.1}$$

and the incompressibility constraint:

$$\nabla\cdot\mathbf{u} = 0. \tag{5.2}$$

Here $\mathbf{u}(\mathbf{x}, t)$ denotes the velocity field, $p(\mathbf{x}, t)$ the pressure field, $\rho$ the fluid density, $\mu$ the dynamic viscosity, and $\mathbf{g}$ the gravitational acceleration vector. The momentum equation enforces conservation of momentum under advective transport, pressure gradient forces, viscous diffusion, and body forces; the divergence-free condition enforces mass conservation by ensuring volume preservation of fluid elements. In computer graphics contexts, these equations form the basis for physically grounded fluid behaviour, although various approximations or discretisation strategies are typically applied to achieve stability, efficiency, or control in practice.

When altitude variations are negligible compared to horizontal scales, a depth-averaged approximation known as the Shallow Water Equations (SWE) is often employed [Par19]. Under the assumption that the fluid column height $\eta(x, y, t)$ varies slowly in the vertical direction, the (non-conservative) SWE are expressed as:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\mathbf{g} \nabla \eta, \tag{5.3}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (\eta \mathbf{u}) = 0, \tag{5.4}$$

where $\mathbf{u}(x, y, t)$ is the horizontal velocity at the surface, and $\mathbf{g}$ denotes gravitational acceleration. The first equation enforces conservation of mass in the depth-integrated sense, and the momentum equation balances advective transport and pressure forces arising from fluid depth. This model offers substantial computational savings and is widely used for real-time or large-domain simulations when three-dimensional effects such as vertical vortices and breaking waves are not critical [Par19].

Modelling assumptions are made explicit in selecting the governing equations. The fluid is typically assumed to be Newtonian and incompressible, with constant density and viscosity. Body forces are often limited to gravity, and surface tension is neglected. The incompressibility assumption simplifies the mathematical treatment and enhances numerical stability; its enforcement commonly involves a pressure projection step that ensures the velocity field remains divergence-free. Viscosity may be incorporated implicitly or explicitly depending on stability requirements, and external forces or boundary conditions are specified to match the intended scenario.

### Numerical solvers

Hydrodynamics are continuous but numerical solvers are commonly organised according to their discretisation paradigm, with each category offering different trade-offs in stability, adaptivity, and computational cost. The principal categories comprise grid-based Eulerian methods, particle-based Lagrangian techniques, hybrid schemes that combine grid and particle representations, and reduced models tailored for simplified scenarios or interactive control.

On one hand, grid-based Eulerian methods discretise the fluid domain on a fixed grid and approximate the incompressible Navier-Stokes equations via operator splitting or projection schemes. Historically, the Marker-and-Cell (MAC) approach established the staggered-grid representation [HW65], storing velocities on cell faces and pressure at cell centres to enforce divergence-free constraints accurately in free-surface flows. Subsequent developments in graphics introduced semi-Lagrangian advection with implicit viscosity integration [Sta99], which achieves unconditional stability permitting large time steps at the expense of increased numerical dissipation. Lattice Boltzmann methods (LBM) offer a mesoscopic viewpoint on fluid dynamics, leveraging local collision and streaming operations on a lattice to recover macroscopic behaviour [CD98]; they are notable for parallel efficiency and natural handling of moderate boundary complexity, though three-dimensional or highly irregular domains carry significant computational overhead. Finite-volume or finite-element variants appear in engineering CFD frameworks such as OpenFOAM and can be adapted for graphics applications, but typically require careful optimisation to remain feasible for large-domain or interactive contexts.

On the other hand, particle-based Lagrangian methods represent fluid as discrete particles carrying mass, momentum, and other properties. Smoothed Particle Hydrodynamics (SPH) discretises the

continuum via kernel-weighted interactions among particles [Mon05], naturally handling free surfaces and complex boundaries without explicit interface tracking; however, SPH demands high particle counts for fidelity, and stability challenges (e.g., tensile instability, pressure oscillations) necessitate specialised corrective formulations [Mon05, KBST22]. Pure Particle-In-Cell (PIC) schemes map particle information to a grid for pressure projection but tend to introduce substantial numerical dissipation through frequent interpolation [Har62]. The Fluid-Implicit Particle (FLIP) method mitigates dissipation by updating particle velocities using grid-derived increments rather than full replacements, thereby preserving kinetic energy and small-scale turbulence [BKR88]; FLIP is well suited for detailed free-surface phenomena such as splashes yet requires careful tuning near boundaries to avoid instability. Other particle variants, including Moving Particle Semi-implicit (MPS) methods, extend the Lagrangian paradigm with implicit pressure solves, but share similar demands for neighbour search and stabilisation [KO96].

Hybrid approaches aim to combine the stability and incompressibility enforcement of grid-based solvers with the adaptivity and natural boundary handling of particles. In common hybrid pipelines, particles carry momentum and advect passive quantities, while a background grid enforces the divergence-free condition via projection. Affine Particle-in-Cell (APIC) refines the transfer between particles and grid by representing particle velocity fields affinely, improving momentum conservation and reducing dissipation relative to FLIP [JSS*15]. Such hybrids leverage the strengths of each paradigm but introduce overhead in particle-grid transfers and require careful design to maintain consistency and numerical robustness in dynamic domains.

Reduced fluid models are employed when full three-dimensional simulation is unnecessary or prohibitively expensive. Depth-averaged shallow water equations capture large-scale horizontal flows over terrain under the assumption of negligible vertical variations; their lower-dimensional form yields significant computational savings and is widely used in real-time or large-domain scenarios where vertical vortices or breaking waves are not critical [Vre94, PHTB12]. Potential flow approximations or other simplified models may be invoked for wave-like phenomena where vorticity and viscous effects can be ignored. Procedural or heuristic approximations, such as cellular automata-based flow or ad hoc velocity fields, support highly interactive or stylised effects by sidestepping full PDE solves, at the cost of physical fidelity. These reduced methods serve both as initial terrain-shaping tools and as fallback options for real-time applications where performance constraints dominate.

### Solver characteristics

Numerical characteristics of solvers critically influence the realism, stability, and performance of fluid simulation. These include time integration and stability properties, processing of free surfaces and boundary conditions, control of numerical dissipation to preserve detail, and computational efficiency and scalability strategies.

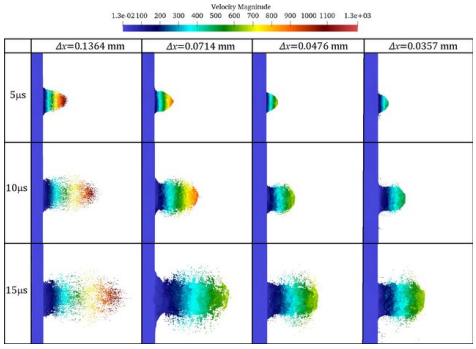



**Figure 5.20:** *Results of a fluid simulation after 5µs, 10µs and 15µs with different spatial resolutions (from left to right: 0.14mm, 0.07mm, 0.05mm and 0.04mm) shows large discrepancies.*

Time integration schemes must balance stability and accuracy. Explicit methods compute updates directly from known states but impose restrictive time-step limits proportional to grid spacing ($\Delta x, \Delta y, \Delta z$) or particle spacing in relation with the computed velocity, rendering fine resolutions costly. Figure 5.20 illustrate the importance of adequat spatial resolution with respect to temporal resolution. The CFL condition states that the dimensionless Courant number $C$ should not exceed a maximal value $C_{max}$ given depending on the PDE to solve (typically $C_{max} = 1$ for explicit integrations):

$$C = \Delta t \left( \sum_{i=1}^{n} \frac{u_i}{\Delta x_i} \right) \leq C_{max}.$$

Thus, when looking at 2D and 3D explicit methods, the main constraint is finding a valid time-step $\Delta t$:

$$\Delta t_{2\mathrm{D}} \leq \left(\frac{\Delta x}{u_x} + \frac{\Delta y}{u_y}\right) C_{max}, \quad \Delta t_{3\mathrm{D}} \leq \left(\frac{\Delta x}{u_x} + \frac{\Delta y}{u_y} + \frac{\Delta z}{u_z}\right) C_{max}. \tag{5.5}$$

Implicit or semi-implicit treatments of diffusion or pressure terms allow larger time steps by solving linear or nonlinear systems ($C_{max} > 1$), as exemplified by semi-Lagrangian advection with implicit viscosity in [Sta99]. Pressure projection typically entails solving a Poisson equation, for which direct solvers offer accuracy but scale poorly to large grids, while iterative solvers (e.g., conjugate gradient, multigrid) afford scalability at the expense of convergence concerns that must be managed via preconditioning or adaptive tolerance. Time-stepping strategies may incorporate adaptive substepping or projection frequency adjustments to maintain stability without excessive computation.

Accurate handling of free surfaces and boundary conditions is essential for plausible fluid behaviour. Interface-capturing methods in Eulerian solvers, such as level-set or volume-of-fluid, track the free surface implicitly but can suffer volume loss or smearing; corrective reinitialisation or particle-based markers near the interface often mitigate these deficits. Particle-based schemes represent the free surface naturally through particle distribution but require surface reconstruction for rendering and may exhibit clumping or void regions without density control.

Solid boundary conditions in both paradigms demand robust collision and pressure treatment: Eulerian grids enforce no-slip or free-slip via ghost cells or immersed boundary techniques, while particles interact with geometry via repulsion forces or dynamic boundary particles. Hybrid methods must synchronise interface representation between particles and grid, ensuring that evolving boundaries remain consistent with divergence-free constraints. Dynamic domains, such as moving obstacles or adaptive meshes, necessitate regridding or particle reseeding procedures that preserve mass and momentum.

Numerical dissipation and detail preservation influence the visual richness of simulated flows. Semi-Lagrangian advection and grid-particle interpolation in PIC introduce artificial smoothing that dampens small-scale vortices and surface detail. Techniques to mitigate dissipation include the FLIP update, which applies only the change in grid velocity to particles, and higher-order advection schemes that reduce numerical diffusion. Vorticity confinement or turbulence-enhancement terms may reintroduce fine-scale structures lost to dissipation. In Eulerian solvers, divergence-free interpolation schemes and improved projection methods help maintain kinetic energy. SPH and other particle methods can preserve detail inherently but may suffer from noise or instability if neighbour sampling is irregular; stabilisation strategies, such as density reinitialisation, kernel correction, or pressure regularisation, seek to retain fidelity without sacrificing stability. Machine learning-based super-resolution methods have recently emerged to reconstruct fine details atop coarse simulation outputs, however, care must be taken when applying them to simulations that differ significantly from the data they were trained on, as they may not produce reliable results.

### Non-physical fluid simulations

Recent advances in fluid simulation increasingly integrate machine learning techniques to accelerate or augment traditional numerical methods. Surveys of machine learning applications in Computational Fluid Dynamics identify roles for data-driven methods, physics-informed models, and ML-assisted numerical solvers that improve convergence or predict intermediate quantities such as pressure fields and subgrid turbulence closures [HFTL22].

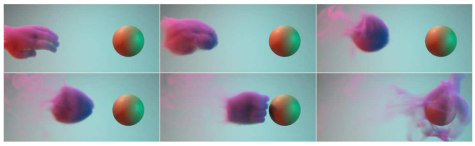


**Figure 5.21:** *An incompressible force field is computed to guide the smoke simulation from a user-defined shape to another [TACS21].*

Neural surrogates were proposed to approximate costly components of the solver, yielding substantial speed-ups while maintaining acceptable fidelity in graphics contexts; such approaches often leverage

convolutional networks for pressure projection or learn residual corrections to coarse simulations [TSSP17, SRA24]. Physics-informed neural networks and related frameworks enable embedding governing equations into the learning process, facilitating generalisation and adherence to conservation laws, though their applicability to large-scale, real-time graphics remains an active research area [BNK25].

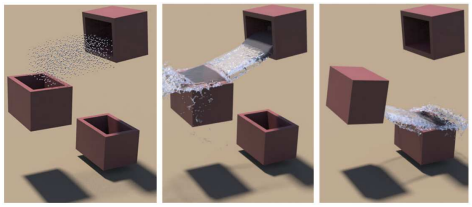


**Figure 5.22:** *Lu et al. uses fluid rigging-skinning on water particles to guide the animation of particles [LCY*19].*

Procedural or artistic fluid control has been an emerging topic for the last few decades, striking a balance between user interaction and plausibility of the fluid behaviour without full PDE solving. Chapter 4 used the Kelvinlet paradigm [GJ17], an analytic formulation for material deformation derived from Kelvin's state using Green's function of linear elasticity of material, providing more control types than the fluid-specific Stokelets [CY76]. Wejchert and Haumann shows that these analytic fluid primitives are an efficient and lightweight approximation of highly controlled fluid simulations [WH91]. Stroke-based fluid models are closer to the storyboard design of animation artists, making them intuitive to use, but are usually for subsets of frames of an animation, which become time-consuming for the price of high-level control [XKG*16, PT05, YCYW20, PHT*13].

Procedural fluids are usually defined by noise functions to artificially introduce motion [BHN07], but can also be added after computing an accurate fluid simulation to introduce more vortices [WZF*25]. Vortices are usually too small for Eulerian simulations as the grid's cell width used may be larger than a vortex, and the dissipation removes their presence. Editing velocity fields procedurally can also come through the use of frequency-domain editing of forces as used in Figure 5.21 [FN20, TACS21], blending and interpolation techniques to merge a predefined fluid with another [RWTT14], or even morphing techniques to deform a velocity or force field as shown in Figure 5.22 [LCY*19, RTW*12, FEHM19].

### Conclusion

Non-physical fluid simulations include procedural and artistic fluids during or after a more computationally expensive simulation, as they introduce variation on different levels: force field and velocity field of Eulerian simulation methods, or the force, velocity, and even position of simulation particles of Lagrangian methods [Sim90].

---

Fluid simulations come in a large variety of forms, each with advantages and drawbacks: accuracy, energy preservation, user control, and boundary representations. Erosion on terrains, and more importantly submerged terrains, requires the computation of the motion of fluid in the environment, whether it represents air for wind-driven phenomena or water for hydraulic erosions. Erosion simulation algorithms proposed in literature do not offer alternatives to the method-specific fluid simulation, constraining the user to only a subset of possibilities for a given terrain representation.

Our proposed method computes particle motion from any vector field by integrating external forces, namely gravity, buoyancy, and drag. These forces only require the evaluation of the fluid velocity at particle positions, which can be obtained from any chosen solver. As a result, our framework is independent of the underlying fluid simulation technique, allowing users to select between computationally intensive CFD solvers for high accuracy, lightweight models for fast computation and ease of implementation, or procedural and artistic simulations to enhance control and flexibility.

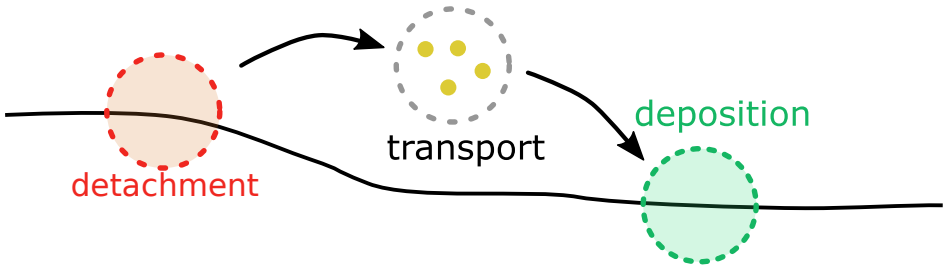



**Figure 5.23:** *Three steps of the erosion process from the sediment point of view: detachment from its original location (dotted red circle), transport in a fluid (dotted black circle), deposition at a new location (dotted green circle).*

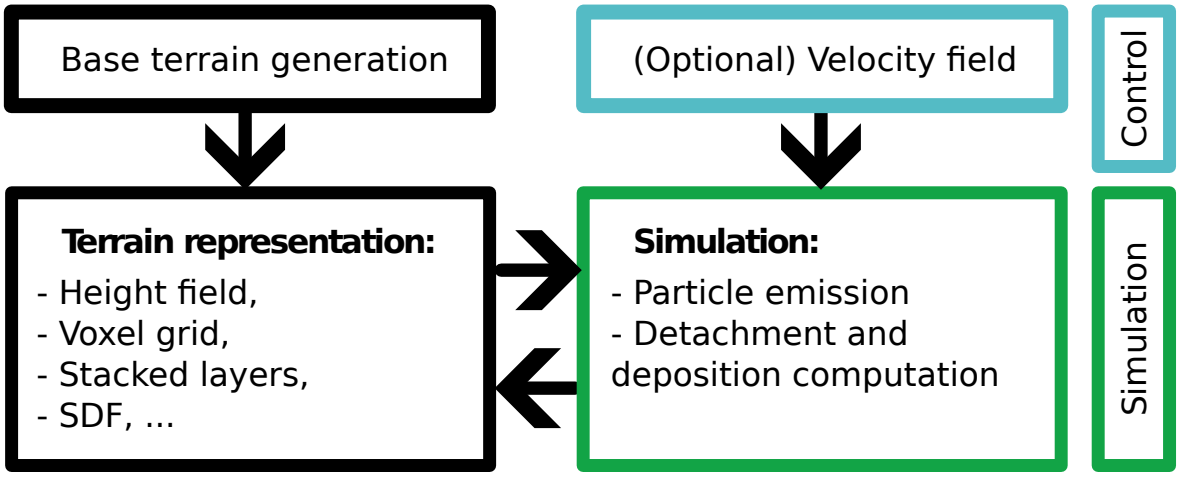



**Figure 5.24:** *Our overall pipeline: our erosion process compute matter displacement of a terrain using an arbitrary representation as long as intersections between particles and the ground can be detected. An optional velocity field, provided by the user, guides the particles trajectories. We propose surface alteration methods to apply the erosion to the terrain in a coherent way between possible representations.*

## 5.3 Particle erosion

Erosion occurs in three stages: material detachment, transport and deposition (respectively in red, black and green in Figure 5.23). In our approach, particles move through the medium following its flow (i.e. wind in air or currents in water) and then absorb or deposit a small amount of material upon contact with the land surface, effectively fulfilling the three stages of erosion, separating the transport process from the detachment and deposition processes (Figure 5.24). The overall algorithm presented in Algorithm 8 illustrates the separation of these processes.

### 5.3.1 Overview

**Input:** Terrain $\mathcal{T}$, particle parameters $(R, \rho_{\text{particle}}, COR, C_{\max}, K_\varepsilon, \omega)$, medium params $(\rho_{\text{fluid}}, \mu)$, velocity field $v_{\text{fluid}}$
**Output:** Updated terrain $\widetilde{\mathcal{T}}$

```
for iter ← 1 to N_iters do
    C ← ∅                                              // Collisions list
    // (1) Emit particles in domain
    P ← SampleParticles(T, params)
    // (2) Particle transport
    foreach p ∈ P do
        while p in terrain do
            integrate motion                  // Equations (5.14) and (5.15)
            if p intersects T then
                add collision data to C     // particle + terrain properties
                apply collision response                       // Algorithm 9
    // (3) Terrain updates
    foreach collision in C do
        accumulate terrain change Δh, Δf, orΔμ                 // Section 5.4
    commit to T and recompute normals
return T̃
```

**Algorithm 8:** Particle-based erosion iterations.

Particles are transported through the medium and may traverse several different media. Each medium is defined by a density and a flow. Consider, for example, water density to be $1000\,\mathrm{kg\,m^{-3}}$ and that of air to be $1\,\mathrm{kg\,m^{-3}}$. The gravity applied to the particles is then very different between open and submerged environments due to the difference in buoyancy, while the process remains similar. Using a pre-calculated flow field to guide particle movement simplifies the simulation by treating particles as independent entities, eliminating the need for inter-particle calculations. This not only significantly reduces the overall execution time but also offers users high flexibility over the quality of the simulation and simplifies the implementation. Moreover, we consider that particles in a fluid do not influence the flow as it would become "overkill" for such lightweight objects [WZF*03].

### 5.3.2 Erosion process

Every time the particle hits the ground, a given amount $q_{\text{detachment}}$ of sediment is detached from the ground while another amount $q_{\text{deposit}}$ of sediment is deposited at this location (respectively red and green arrows in Figure 5.25). Our erosion model is based on the work of [WCMT07] where regular 3D grids are used to estimate the fluid velocity and sediment transport. In the spirit of [KBKŠ09], we transposed their method into a particle-based erosion simulation, but in our proposition, we decouple the particle system from the fluid simulation, making the process more flexible and opening the door for richer effects that can easily be produced.

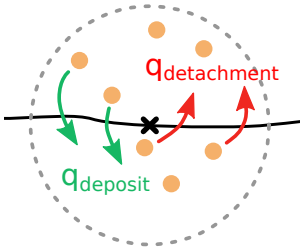



**Figure 5.25:** *When a particle is in contact with the terrain surface, an amount $q_{deposit}$ of sediments contained in the particle is released in the ground, and $q_{detachment}$ is absorbed.*

### Detachment

As a particle approaches the surface of the terrain, its motion applies friction at the interface between fluid and ground, causing bedrock to dislocate microscopic parts, which we call abrasion. We use pseudoplastic models to approximate the amount of matter removed due to the shear forces while considering the physical properties of the fluid and the ground [WCMT07].

The shear rate $\theta$ is approximated by the relative velocity of the fluid to the solid boundary $\vec{v}_{rel}$ over a short distance $l$. We approximate the shear stress $\tau$ at the solid boundary by a power law:

$$\tau = K\theta^n, \tag{5.6}$$

where $\theta = \vec{v}_{rel}/l$, $K$ is the shear stress constant (often set to 1), and $n \in [0,1]$ is the flow behaviour index. Shear-thinning models typically assume $n$ close to $\frac{1}{2}$, which is why we used this value as a constant.

We can then compute the erosion rate $\varepsilon$ at any contact point between a fluid and a solid boundary using Equation (5.6) by

$$\varepsilon = K_\varepsilon(\tau - \tau_{\text{critical}})^a, \tag{5.7}$$

with $K_\varepsilon \in [0,1]$ a user-defined erosion constant, $\tau_{\text{critical}}$ the critical shear stress value for which the matter starts to behave in similar manner as a fluid, and $a$ a power-law constant, typically considered as $a = 1$.

In our method, the eroded quantity is approximated as the material contained in the hemisphere of radius $R$, in the normal opposite direction at the particle impact point (Figure 5.29). We then use Equation (5.7) to get the final eroded amount $q_{\text{detachment}}$:

$$q_{\text{detachment}} = \varepsilon\frac{2\pi R^3}{3}. \tag{5.8}$$

The particle is also defined by a maximal amount of sediment that can be contained in its volume before being saturated, noted $C_{\text{max}}$.

### Deposition

The eroded sediment is considered to be in suspension in a fluid and is affected by its velocity. A fluid particle then transports the sediment in its flow until gravity settles it onto the ground again. The effect of gravity is modelled by a settling velocity $w_s$:

$$w_s = \frac{2}{9}\vec{g}R^2\frac{(\rho_{\text{particle}} - \rho_{\text{fluid}})}{\mu}f(C), \tag{5.9}$$

where $f(C)$ is a hindered-settling factor that reduces the effective body force as the particle's carried sediment approaches its maximum capacity. Following Richardson-Zaki [RZ54], we take

$$f(C) = \left(1 - \frac{C}{C_{\text{max}}}\right)^n, \tag{5.10}$$

with exponent $n$ typically in the range 4-5.5, and we set $n = 5$, similarly as [WCMT07].

We consider that the amount of sediment settled is proportional to the norm of the settling velocity as proposed in [WCMT07], with $\omega \in [0, 1]$:

$$q_{\text{deposit}} = \omega \|w_s\|. \tag{5.11}$$

### 5.3.3 Transport

Our simulation is computed by integrating the full trajectory of multiple particles at each iteration, unlike most other erosion methods. This allows us to constantly have a terrain in a plausible state, while giving the possibility to increase the ageing effect by running more iterations. Note that progressively reducing the overall erosion strength can be used as a strategy to adapt the computation time to a chosen level of detail.

We first present how to compute the particle speed using particle physics, then how to add an optional medium velocity field to add a fluid simulation or user control.

#### Particle physics

Due to their independence from other particles, we consider each particle to follow Newton's laws of motion. Each particle has a volume $V$, an effective density $\rho_{\text{particle}}$ (which may depend on the amount of carried sediment), and thus a mass $m_{\text{particle}} = \rho_{\text{particle}} V$. The surrounding medium has density $\rho_{\text{fluid}}$, viscosity $\mu$, and velocity field $v_{\text{fluid}}(\mathbf{p}, t)$.

The exerced forces applied to a particle consist of external forces $\vec{F}_{\text{ext}}$ (gravity and buoyancy), and drag force $\vec{F}_{\text{drag}}$ (Figure 5.26). Gravity acts as:

$$\vec{F}_{\text{gravity}} = m_{\text{particle}} \vec{g},$$

with $\vec{g}$ the gravitational acceleration vector. Buoyancy opposes gravity as:

$$\vec{F}_{\text{buoyancy}} = -\rho_{\text{fluid}} V \vec{g}.$$

The net body external force is then

$$\vec{F}_{\text{ext}} = V \left(\rho_{\text{particle}} - \rho_{\text{fluid}}\right) \vec{g}. \tag{5.12}$$

In addition, drag relaxes the particle velocity $\vec{v}$ toward the local fluid velocity $v_{\text{fluid}}$. For small Reynolds numbers we use Stokes' drag formula for spheres in a fluid:

$$\vec{F}_{\text{drag}} = 6\pi\mu R \left(v_{\text{fluid}} - \vec{v}\right), \tag{5.13}$$

where $R$ denotes the particle radius.

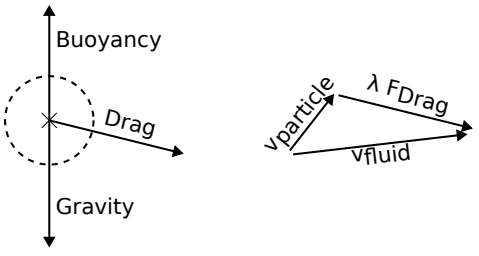



**Figure 5.26:** *Particle's motion in a fluid consist of three forces: gravity, buoyancy, and drag. The later is acting opposite direction to the relative particle velocity in the fluid.*

The full equation of motion is therefore

$$\dot{\vec{v}} = \frac{f(C)\vec{F}_{\text{ext}} + \vec{F}_{\text{drag}}}{m_{\text{particle}}}, \tag{5.14}$$

with $f(C)$ resulting from Equation (5.10).

The particle position $\mathbf{p}$ evolves as:

$$\dot{\mathbf{p}} = \vec{v}. \tag{5.15}$$

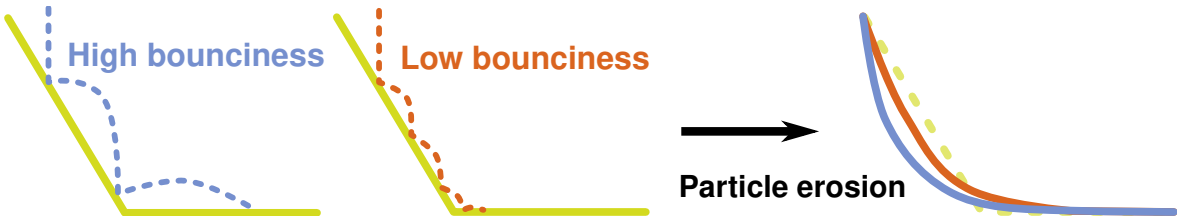



**Figure 5.27:** *The coefficient of restitution affects the amount of energy absorbed from the particle when hitting the ground. Here, rain is applied on an initial slope (yellow). Only two particles are displayed, with a high (blue) and low (red) coefficient of restitution. The resulting slope after erosion is displayed in blue and red (right).*

When a particle collides with the ground, its post-impact velocity is modified by a coefficient of restitution $COR$. This value depends on the ground material as it is influenced mainly by the material's particle shape, coefficient of friction and density [YZKF20]. Less bouncy particles lose speed quickly and settle down sooner, forming a steeper pile (Figure 5.27 blue), or a higher talus angle such as chalk. On the other hand, more bouncy particles disperse more widely upon hitting a surface, resulting in a gentler accumulation such as clay (Figure 5.27 red). We propose this simple collision response in Algorithm 9.

**Input:** Particle $(\mathbf{p}, \vec{v}, C)$, terrain $\mathcal{T}$, restitution $COR$
**Output:** Updated particle state $(\vec{v}, C)$

$\vec{n} \leftarrow \mathcal{T}.\vec{N}(\mathbf{p})$
$\vec{v} \leftarrow COR\,(\vec{v} - 2\,\langle \vec{v}, \vec{n} \rangle\,\vec{n})$
`// Update sediment capacity at this point`
compute $q_{\text{detachment}}$, $q_{\text{deposit}}$ `// Equations (5.8) and (5.11)`
$C \leftarrow C + q_{\text{detachment}} - q_{\text{deposit}}$
**return** $(\vec{v},\ C)$

**Algorithm 9:** Particle-terrain collision response.

### Velocity field

With our particle system, we propose to model intricate scenarios, such as the erosion caused by water currents on the seabed or aeolian erosion. The velocity field remains static during the erosion, which may cause inconsistencies in the fluid velocity field. However, minor changes can be overlooked to maintain a balance between realism and computational efficiency [TJ10]. We suggest three velocity improvement methods:

- **Fluid simulation refinement:** Various erosion mathods incorporate fluid simulation, requiring regular updates for erosion and velocity [KBKŠ09, WCMT07]. Our method can use fluid simulations with multi-resolution refinement, with the possibility to focus the velocity field adjustments near the updated boundaries of the surface [RLL11].

- **Particle velocities in fluid simulation:** With a Lagrangian fluid simulation relying on particle systems [KBST22], our particle velocities can be incorporated into its computation. This approach is only a provisional solution due to potential parameter mismatches with the main fluid simulation.

- **Velocity field diffusion:** Given the minor changes to the surface level at each erosion iteration, which reflect the gradual alterations in terrain surface, we estimate that the velocity at a fixed point transitioning between the inside and outside of the terrain closely mirrors the velocities observed in its surrounding area. In this context, we simply interpolate the velocity field at any transitioning point. This simple method, as used in Figure 5.46, allows us to find a balance between achieving realistic flow simulations and maintaining computational efficiency.

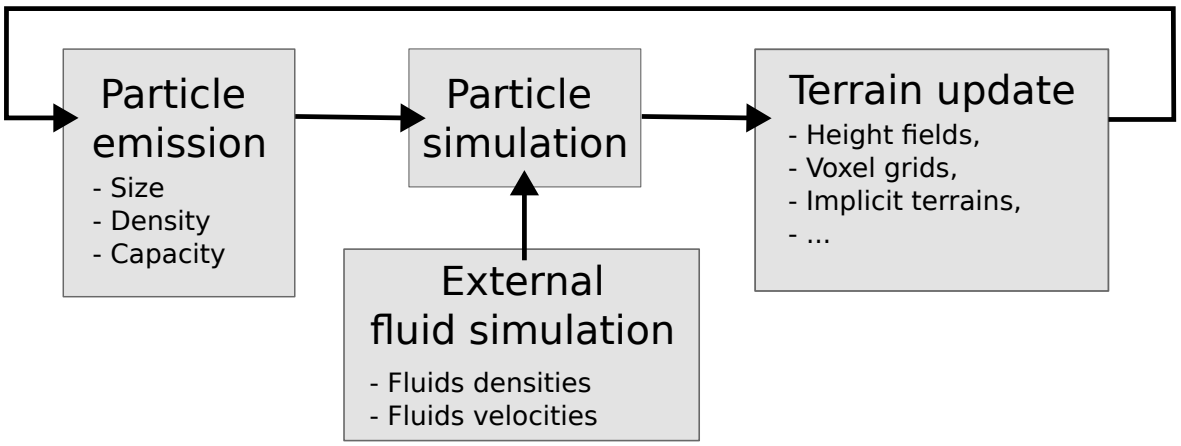



**Figure 5.28:** *Our method requires a base geometry, a small number of parameters for the particles and the medium used for the erosion simulation. It can be easily adapted to be compatible with different mediums and terrain representations.*

## 5.4 Our erosion method

In this section, we describe how to apply detachment and deposition to different terrain representations with our method (Figure 5.28). We cover the most commonly used representations, namely height fields, layered terrains, voxel grids and implicit surfaces. Note that our work could be extended to additional representations. Two conditions need to be satisfied for a representation to be eligible for our erosion method: the ability to evaluate the intersection of a particle with the ground and to compute the normal of the terrain at this point. To the best of our knowledge, all representations do.

We use Verlet integration for the particle physics [Ver67], with low error rate and stability even for high $dt$, reducing computation time for negligible imprecision [BW98, SABW82].

For all the representations, the amount of material absorbed by the particle, i.e. the erosion value $q_{\text{detachment}}$ from Equation (5.8), is taken around the particle at a radius $R$, meaning that the modification of the terrain by a particle at position $\mathbf{c}$ will only occur for the positions $\mathbf{p}$ satisfying $||\mathbf{p}-\mathbf{c}|| < R$. At the same time, the amount $q_{\text{deposit}}$ from Equation (5.11) is deposited, resulting in a change $Q = q_{\text{deposit}} - q_{\text{detachment}}$.

In our simulation, while the dynamics are informed by physical principles, the particle size is conceptualised within a dimensionless framework. This provides the flexibility to adapt our results to various real-world scales, ensuring the applicability of our model across diverse scenarios. Note that, for a 2.5D terrain, we can consider that half of the sphere surrounding the particle is affected, which has a volume of $V_{2.5D} = \frac{2\pi R^3}{3}$, while a 3D terrain is affected by the full sphere $V_{3D} = \frac{4\pi R^3}{3}$ (as illustrated in Figure 5.29). In the following sections, we will describe the strategies used to modify the amount of matter for different representations.

### 5.4.1 Application on height fields

On a height field defined by $h(\mathbf{p}) = z$, the intersection point with the surface is verified at $\mathbf{p}_z = h(\mathbf{p})$, and the normal can be computed at the intersection point.

For this representation, the half-sphere is scaled in the $z$ direction to fit $\alpha V = Q$ using $\alpha = \frac{Q}{V}$. We then decrease the height $h'(\mathbf{p})$ at all points $\mathbf{p}$ by the height of the scaled half-sphere at position $\mathbf{p}$. Given the height of the scaled half-sphere of centre $\mathbf{c}$ and the distance of the particle to the centre $d = ||\mathbf{p}-\mathbf{c}||$ by $h_{\text{half sphere}}(\mathbf{p}) = \alpha\sqrt{R^2-d^2}$ for all $\mathbf{p}$ such that $d \leq R$, the radius around a particle.

This change of height can be sampled at all points of the 2D grid by reducing the height by

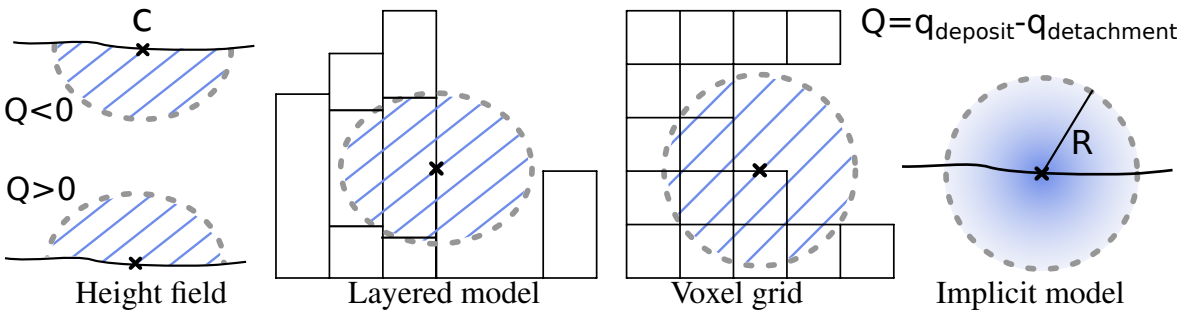



**Figure 5.29:** *Illustration of the material detachment in the (half-)sphere at contact point C (cross) on different representations. (height field) When $Q < 0$, material detachment happens in the bottom scaled half-sphere of the particle's contact with the ground, while the deposition is applied on the upper half-sphere of volume when $Q > 0$. Unlike the height field, for 3D terrains detachment and deposition are applied in the full sphere around the contact point.*

$$\Delta h(\mathbf{p}) = \frac{\sqrt{R^2 - d^2}}{\alpha} = \frac{Q}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2}. \tag{5.16}$$

The height at each point after an erosion is then computed as $\tilde{h}(\mathbf{p}) = h(\mathbf{p}) + \Delta h(\mathbf{p})$.

### 5.4.2 Application on layered terrains

Layered terrains are defined as $\mu : \mathbb{R}^3 \mapsto \mathbb{N}$, assigning a discrete material index $\mu$ for any point in space [BF01, PGGM09]. In the original work, outer borders stack elements of the terrain are transformed into density voxels to enable global erosion through height changes. We enable the erosion/deposition process directly on the layers, hence removing the need for representation changes.

When intersecting the terrain, the amount eroded for each material stack should be the integration of the volume of the intersection between the sphere surrounding the particle and the cubicle represented by the stack. Since there is no easy solution [JW17], we approximate the volume of the stack we need to alter using the previously defined height field equation Equation (5.16). At a distance $d$ from the particle, the height is defined as:

$$H(d) = \frac{|Q|}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2}. \tag{5.17}$$

If $Q > 0$ (more deposition is applied than detachment), then we transform the materials in the stack contained in the sphere to become ground material. For $Q < 0$, the materials are transformed into background material.

### 5.4.3 Application on implicit terrains

Implicit terrains are defined using a function $f(\mathbf{p})$ and its variation resulting from the erosion process using $\Delta f(\mathbf{p})$. We propose a strategy to compute $\Delta f(\mathbf{p})$ at any point of the sphere surrounding the erosion point based on metaball primitives. At each contact point, a metaball is added to create a hole or a bump in the terrain. A metaball is defined as:

$$\Delta f(\mathbf{p}) = \frac{3Q}{\pi R^3}\frac{(1-d)}{R}, \tag{5.18}$$

with $d$ the distance of the point $\mathbf{p}$ to the sphere centre for all points $\mathbf{p}$ for which $d \geq R$, $\Delta f(\mathbf{p}) = 0$ (see Annex B).

As they are the most commonly used representations, we propose a formulation to erode implicit terrains defined by Signed Distance Functions (SDFs) and by gradient or vector fields.

### Signed Distance Functions

Considering SDFs, the terrain is defined as the 0-set of the signed distance function $f : \mathbb{R}^3 \mapsto \mathbb{R}$, hence, for $f(\mathbf{p}) = 0$, the inside is $f(\mathbf{p}) < 0$ and the outer part (i.e. air or water) is $f(\mathbf{p}) > 0$.

The particle erosion applies at impact points at discrete positions, so we propose to add or subtract metaballs defined using equation Equation (5.18) to respectively deposit or erode material using a composition tree: $metaball(\mathbf{p}) = -\Delta f(\mathbf{p})$.

Now the eroded terrain function $\tilde{f}(\mathbf{p})$ will be evaluated at each point $\mathbf{p}$ from the initial terrain value $f(\mathbf{p})$, the erosion function $metaball(\mathbf{p})$ and the composition function $g(f_1, f_2)$:

$$\tilde{f}(\mathbf{p}) = g(f(\mathbf{p}), metaball(\mathbf{p})). \tag{5.19}$$

As a metaball is added for each particle bounce on the terrain, space partitioning optimisation algorithms such as k-d trees, BSP trees or BVH can easily be used to improve performance.

### Other implicit terrains

Other implicit terrain models are present in the literature, notably a 2.5D representation based on the surface gradient [GPM*22] and a 3D representation based on curves [BKRE17], for which the trajectory of each particle projected to the closest surface could be used to define the alteration of the terrain.

In the case of gradient-based representation, we propose to use the partial derivative from the equation of the 2D scalar fields Equation (5.16), which gives, for $d \leq R$,

$$\nabla h' = -\frac{Q}{\frac{2}{3}\pi R^3}\frac{1}{\sqrt{R^2 - d^2}}\vec{CP}, \tag{5.20}$$

with $\vec{CP}$ the vector from the centre of the erosion point $\mathbf{c}$ to evaluate to the position $\mathbf{p}$.

Avoiding the singularity for $R = d$ can be done using a regularisation by introducing $\varepsilon \ll R$ in Equation (5.20):

$$\nabla h' = -\frac{Q}{\frac{2}{3}\pi R^3}\frac{1}{\sqrt{R^2 - d^2 + \varepsilon^2}}\vec{CP}. \tag{5.21}$$

Now the new gradient field can be computed as:

$$\nabla \tilde{h}(\mathbf{p}) = \nabla h(\mathbf{p}) + \nabla h'(\mathbf{p}).$$

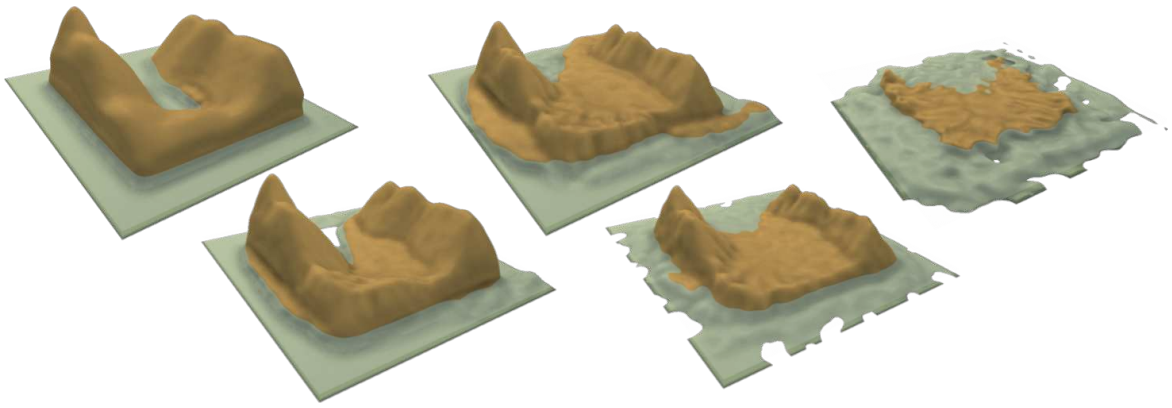


**Figure 5.30:** *Our erosion method is applied iteratively on a completely synthetic island; the terrain is altered to obtain a plausible shape by forming rills. The use of particles with hydraulic densities dropped from the sky results in strong erosion on the sides of the mountains, and the particles that slide to the sea mainly drift offshore, resulting in the formation of small beaches and weaker erosion on the bottom of the water body. Repeating the process causes the island height to decrease progressively, up to the point where only the submerged part of the terrain is sheltered from erosion.*

### 5.4.4 Application on voxel grids

We consider two of the voxel grid representations: density-voxel grids and binary voxel grids, for which we present our material alteration strategy.

#### Density voxels

We consider "density-voxel" grids defined on $f : \mathbb{Z}^3 \mapsto [-1,1]$, for which a voxel is full for $f(\mathbf{p}) = 1$, partially full for $-1 < f(\mathbf{p}) < 1$, or empty for $f(\mathbf{p}) \leq -1$. This definition allows us to erode them smoothly. Since this kind of grid is a discretisation of a scalar function, we could directly use Equation (5.18), as described previously, but we take advantage of the discrete nature of the representation to avoid expensive computation.

We apply the erosion from a particle at position $\mathbf{c}$ on all points $\mathbf{p}$ in the volume, proportionally to the distance from the centre of the sphere $d = ||\mathbf{p} - \mathbf{c}||$, to find an approximation to the real erosion value per voxel $Q_{approx} = Q\frac{1-d}{R}$. Using their discrete nature, we rectify this value to sum up the total erosion value to $Q$ by dividing each value by the sum of the distances. We now consider eroding the "empty" voxels since their density can drop to $-1$. We then have for all surrounding voxels

$$\Delta f(\mathbf{p}) = Q\frac{(1-\frac{d}{R})}{\sum(1-\frac{d}{R})}. \tag{5.22}$$

The resulting voxel value is computed as $\tilde{f}(\mathbf{p}) = f(\mathbf{p}) + \Delta f(\mathbf{p})$. In our implementation, presented in Algorithm 10, when $f(\mathbf{p}) > 1$ we simply transport the density excess to the voxel above, giving it a very close analogy to height fields as long as $|\Delta f| < 1$.

**Input:** Voxel grid $f : \mathbb{Z}^3 \mapsto [-1,1]$, contact point $\mathbf{c}$, erosion radius $R$, volume change $Q$
**Output:** Updated grid $\tilde{f}$

total $\leftarrow \sum_{\mathbf{q} \in \|\mathbf{q}-\mathbf{c}\|<R} (1 - \frac{\|\mathbf{q}-\mathbf{c}\|}{R})$
**foreach** *voxel* $\mathbf{p}$ *with* $\|\mathbf{p}-\mathbf{c}\| < R$ **do**
  $d \leftarrow \|\mathbf{p}-\mathbf{c}\|$
  $\Delta f(\mathbf{p}) \leftarrow Q \cdot \frac{(1-\frac{d}{R})}{\text{total}}$ `// Equation (5.22)`
  $f(\mathbf{p}) \leftarrow f(\mathbf{p}) + \Delta f(\mathbf{p})$
  **if** $f(\mathbf{p}) > 1$ **then**
    spill excess upward voxel
**return** $\tilde{f}$

**Algorithm 10:** Density-voxel terrain update.

### Binary voxels

The terrain can be represented using an occupancy function as $f : \mathbb{Z}^3 \mapsto \{0,1\}$, where a voxel $f = 1$ defines the ground and $f = 0$ the background.

We propose to apply particle erosion by assigning voxels a number of hits, and transforming them into air or ground when this number reaches a critical value $C$ that is proportional to the particle's strength parameter $K_\varepsilon$ [BBFJ10].

On a hit, all voxels in a radius $R$ receive a hit number

$$\Delta hits = \lfloor \alpha \Delta f \rfloor, \tag{5.23}$$

with $\Delta f$ the erosion per voxel computed using Equation (5.22) and $\alpha$ a coefficient high enough to obtain values above 1.

All voxels with $\#hits > C$ are transformed into background, and voxels with $\#hits < -C$ are transformed into ground.

Note that a binary voxel grid can also be transformed into a density-voxel grid to be eroded smoothly.

Our formulation for height fields Equation (5.16) can be used to erode 2D scalar field-based representations. Similarly, our proposition for SDFs Equation (5.18) enables erosion for continuous 3D scalar fields, and voxels Equation (5.22) for discrete 3D scalar fields, respectively.

## 5.5 Results

Our erosion process supports the simulation of a wide range of erosion effects across different terrain representations. In this section, we demonstrate the versatility of the method by varying parameters such as particle effect size, quantity, density, maximum capacity, deposition factor, and velocity fields.

The influence of each parameter on the erosion process is illustrated in Figures 5.31 to 5.35. In Figure 5.31, particle density controls the particle inertia and thus its sensitivity to the velocity field. Low-density particles follow the wind closely, generating fine and widely distributed erosion, whereas high-density particles travel straighter and carve deeper, more localized channels. In Figure 5.32, increasing the particle radius produces smoother erosion and deposition patterns because larger particles average the terrain gradients over a wider area. Figure 5.33 highlights the combined effect of gravity and the coefficient of restitution: low values of COR cause particles to lose energy rapidly

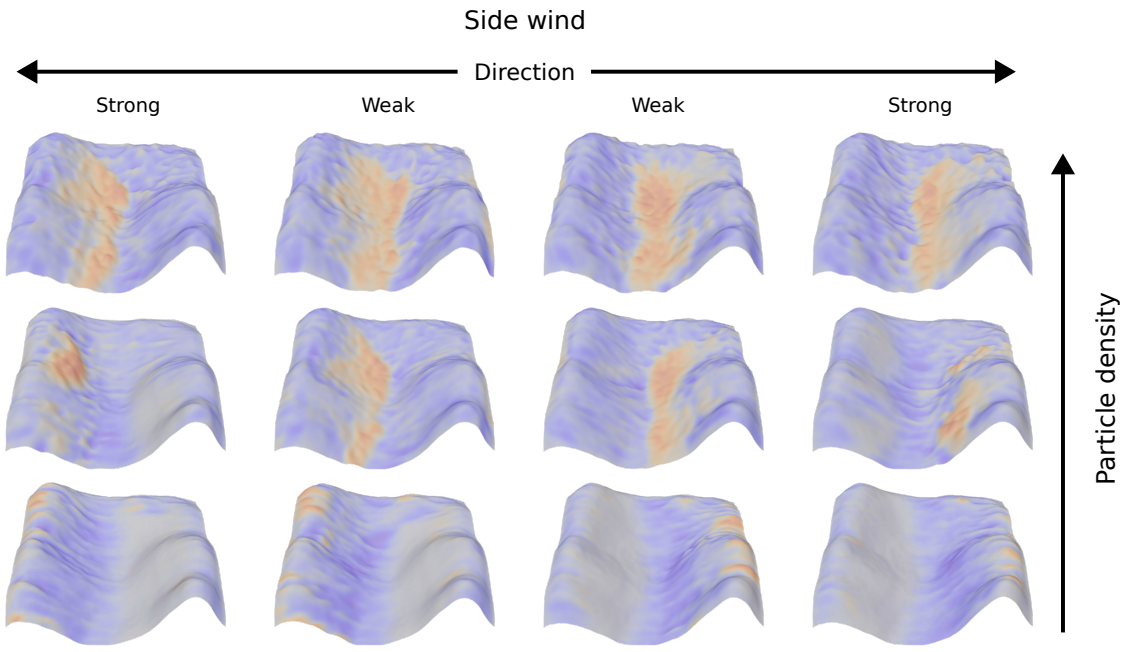



**Figure 5.31:** *Uniform wind field applied laterally on a terrain with different particle densities. Deposition and erosion are visible in red and blue respectively.*

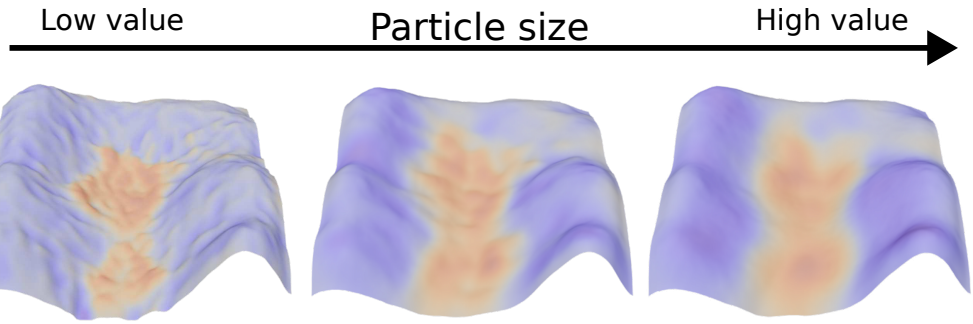



**Figure 5.32:** *Particle radius influence the smoothness over the final result. Red and blue indicate deposition and erosion, respectively.*

and accumulate at the base of slopes, emulating scree or debris deposits, while higher COR values allow particles to travel farther and mimic viscous, fluid-like transport. The influence of critical shear and shear power parameters is shown in Figure 5.34; larger critical shear thresholds limit erosion to steep regions, whereas higher shear power amplifies the rate of material removal over the entire terrain. Finally, in Figure 5.35, the erosion and deposition factors directly control the global mass transfer. High erosion rates deepen valleys and accentuate roughness, while high deposition rates quickly smooth the terrain and fill cavities. Together, these examples demonstrate that the proposed model can reproduce a continuum of erosion behaviours ranging from diffusive smoothing to strongly channelized material transport.

Parameters used for each result are available in Table 5.2. It is important to note that all erosion examples presented in this section are available for any 3D terrain representation. However, we cannot create volumetric structures, such as overhangs, using 2.5D representations (height fields).

The environment density $\rho_{\text{fluid}}$ is set to $1\,\text{kg}\,\text{m}^{-3}$ above the water level (blue regions of the terrain) and $1000\,\text{kg}\,\text{m}^{-3}$ below it. The velocity field is refined using the diffusion strategy described previously.

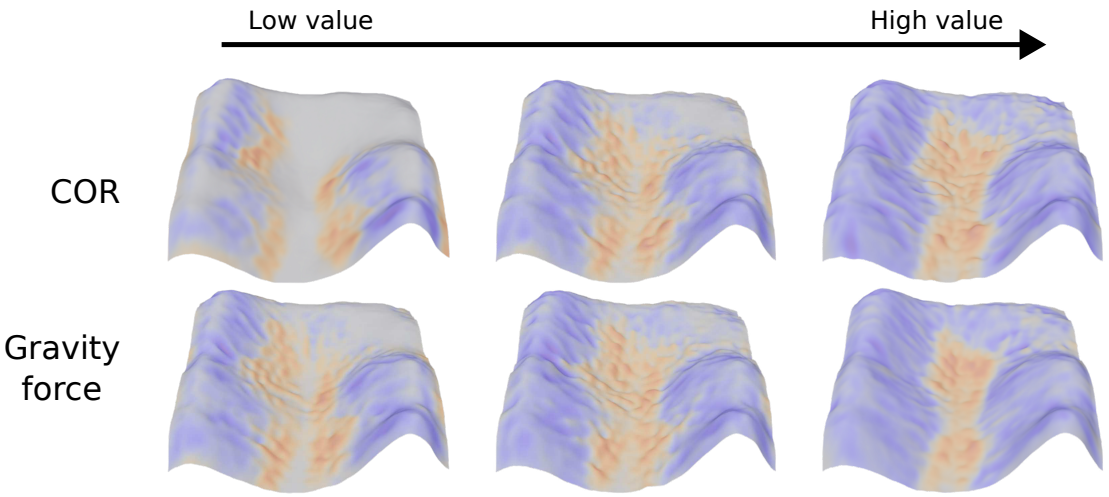



**Figure 5.33:** *The coefficient of restitution (COR) and the force of gravity influence similarly the localisation of deposits. A low COR emulate debris scree at the bottom of slopes; high COR values approximate viscous transports such as hydraulic processes. Red and blue indicate deposition and erosion, respectively.*

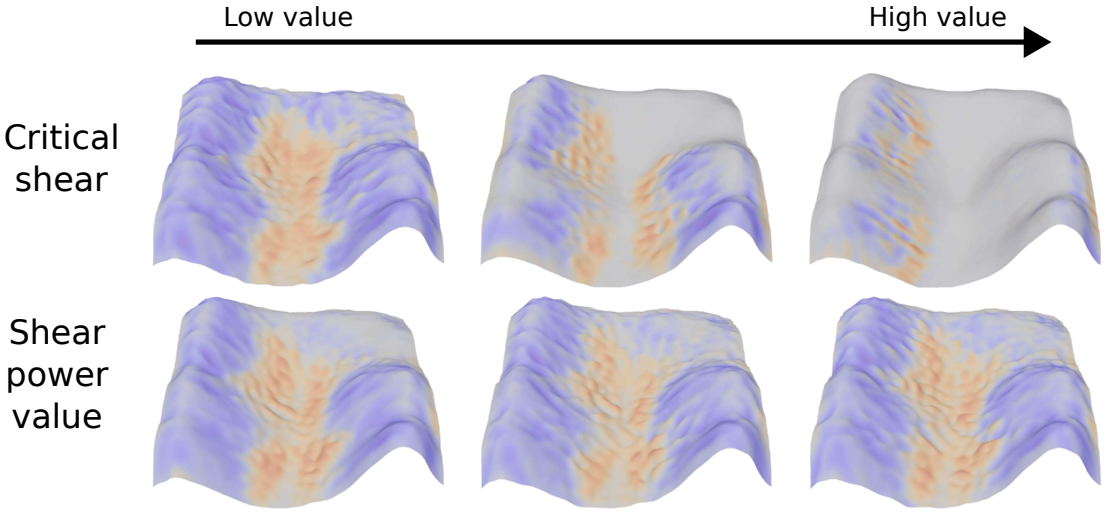



**Figure 5.34:** *Large critical shear values restrict erosion to steep slopes, while large shear power values increase the effect of erosion on the whole terrain. Red and blue indicate deposition and erosion, respectively.*

### 5.5.1 Rain

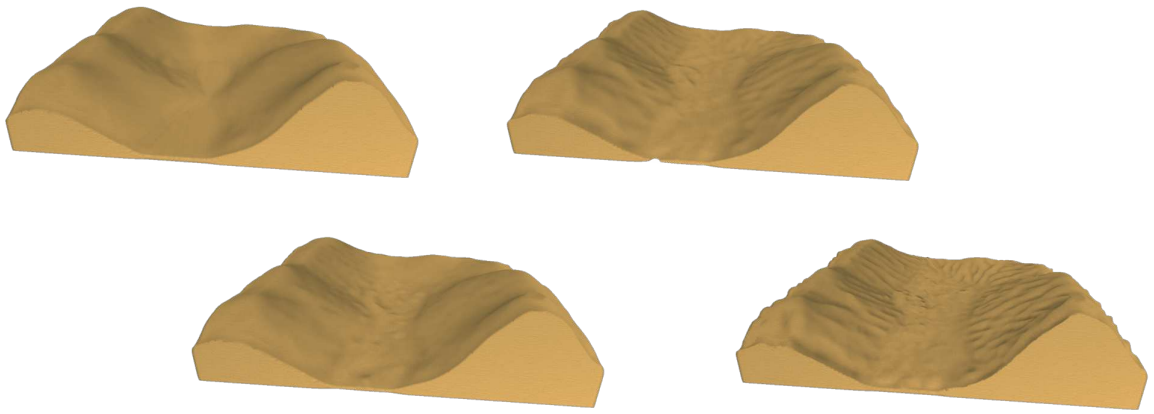


**Figure 5.36:** *Rain erosion on a synthetic heightmap generated by Perlin noise using small water particles.*

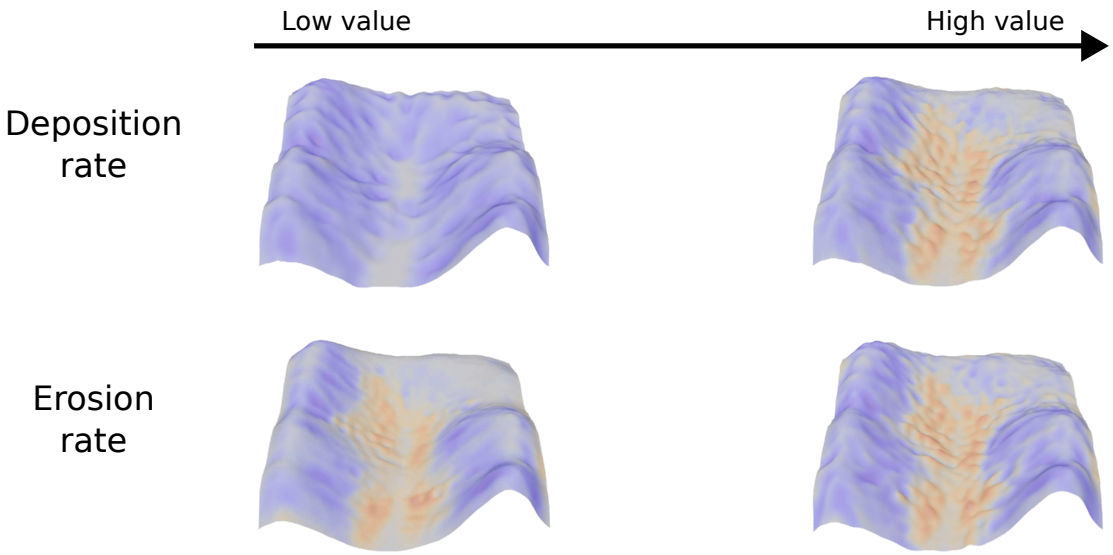



**Figure 5.35:** *Erosion and deposition rates directly influence the displacement of matter from the particles on the terrain. Red and blue indicate deposition and erosion, respectively.*

Hydraulic erosion from rain is the most common process used in terrain generation. In this case, particles are seen as water droplets falling from the sky and rolling downhill due to Earth's gravitational force. No velocity field is required from fluid simulation. These parameters result in detailed geometry of the rills on the sides of mountains that quickly emerge and deposit many sediments in the valley. We demonstrate the result of rain erosion in Figure 5.36 with a computation time of 4 seconds.

Using these erosion parameters in combination with water bodies results in different outcomes (Figure 5.30). The terrain above water is directly affected by the erosion process, while particles colliding with the underwater part of the terrain are slowed down and filled with sediment, leading mainly to deposition. The result is typical hydraulic erosion on mountains and the formation of slopes and beaches near the water level.

### 5.5.2 Coastal erosion

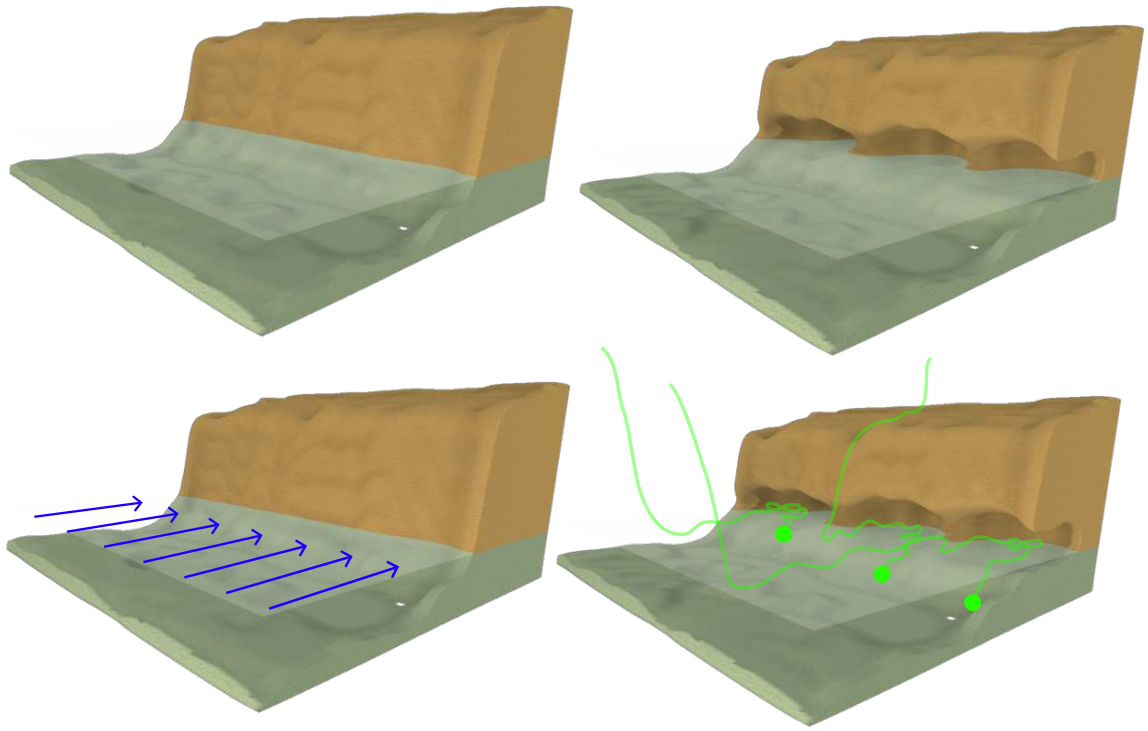


**Figure 5.37:** *Coastal erosion on cliff from density-voxel representation using large particles with density slightly lower than water to target target collision on the air-water interface. A uniform flow field (blue arrows on bottom left) force the particles towards the cliff (green paths of three particles visible in bottom right).*

The repeated motion of waves creates coastal erosion, which can be seen as cliffs with holes at the water level.

We apply a uniform velocity field in the water pointing towards the coast to simulate waves and emit particles from the water area with a large size, a density between air and water densities, a high capacity factor, and a low deposition factor $\omega$. Using these parameters, the erosion process is focused at the interface of air and water, applying coarse detachment while depositing a very small quantity of sediment, simulating the corrosive effect of water on limestone.

This effect can only be simulated on 3D terrain representations since it would create cliffs on a 2D representation. Figure 5.37 presents the result of coastal erosion on a density-voxel grid that creates overhangs around sea level using a small number of particles. Note that the same effect using an alternate implicit representation based on SDF is displayed in Figure 5.48. A shaded version of this effect is presented in Figure 5.1.

### 5.5.3 Rivers

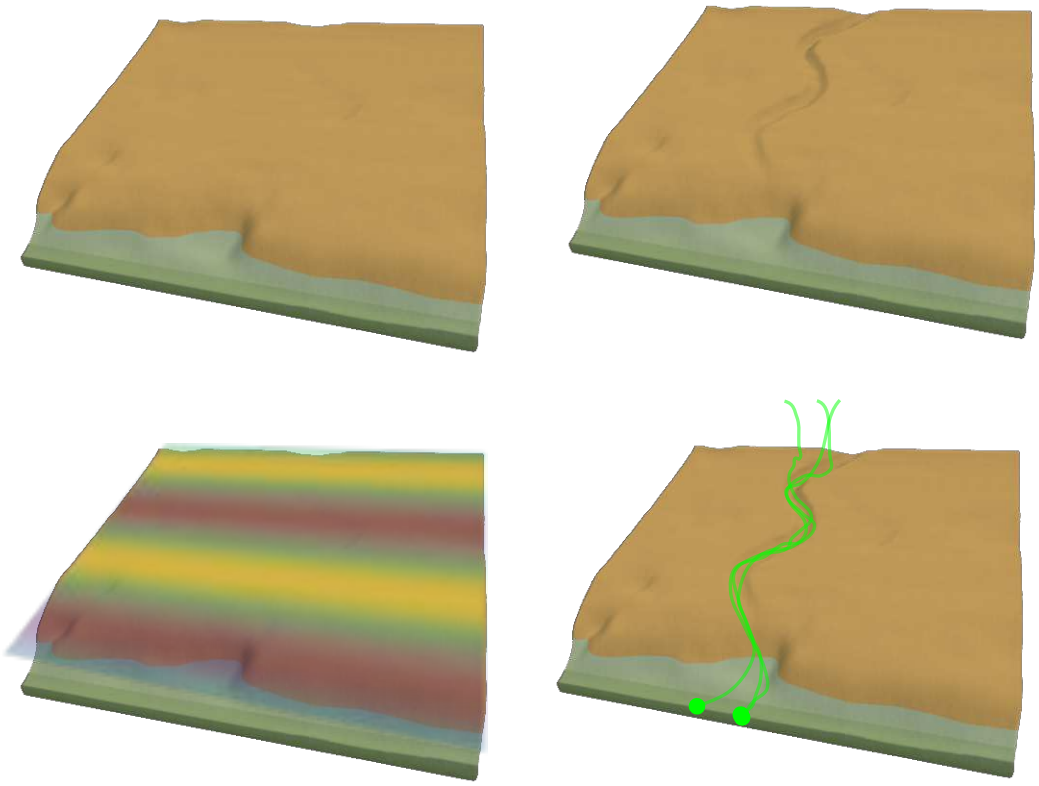


**Figure 5.38:** *Meandering river simulation on an implicit terrain representation using a user-defined sinusoidal flow field. Bottom left illustrate the alternating variation on the velocity field, guiding the particles as illustrated with green paths in bottom right.*

Given a source point, we generate particles that run downhill, simulating the formation of a river. More complex erosion simulations using fluid simulations such as SPH [KBKŠ09] would create realistic results at the cost of high processing time. Our method offers the flexibility to be applied either with a velocity field (simple, user-given, or resulting from a fluid simulation) or without, allowing for simplicity and efficiency.

When provided with a hand-made or procedural velocity field, our particle system reproduces simple river meanders (Figure 5.38).

Figure 5.39 presents a river that has been modelled by emitting water particles with different sizes ranging from 1.5 m to 5 m, a high coefficient of restitution, and a low capacity factor. Random sizes are used to simulate a river for which the flow rate fluctuated over formation time, while the low capacity ensures that the banks of the river stay smooth. A high coefficient of restitution is a strategy that lets the particles flow with low friction, approaching water-like behaviour. Our particles are affected only by gravity, without fluid simulation.

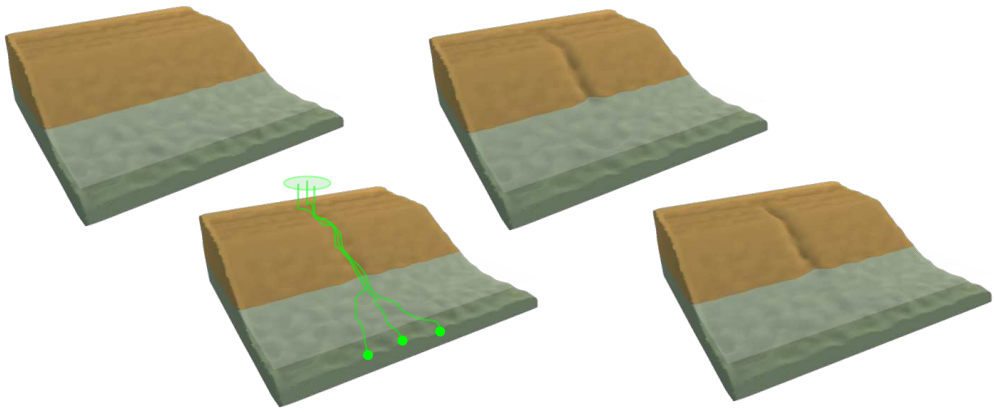


**Figure 5.39:** *A single river simulation on a layered representation with water particles emitted from a small disk (green).*

### 5.5.4 Landslide

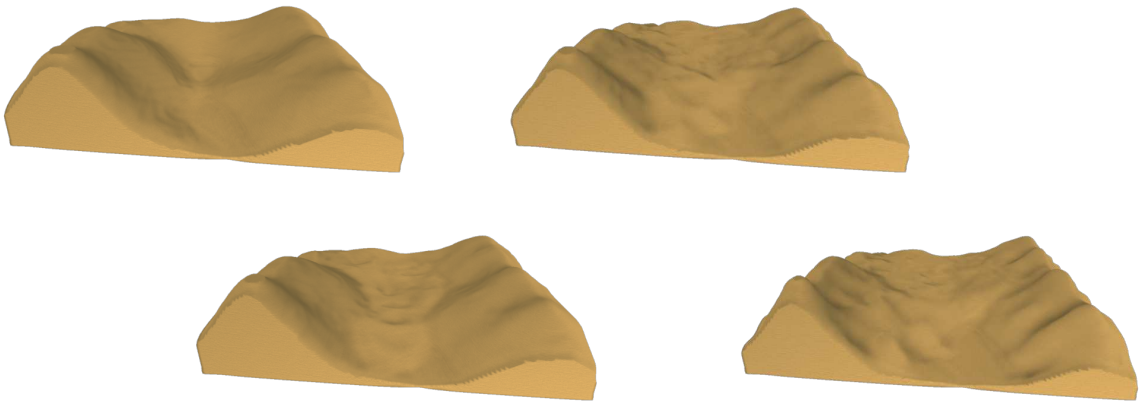


**Figure 5.40:** *Landslides on a height field representation is emulated by using large particles with high deposition factor and low capacity.*

Landslides are mainly caused by large amounts of water saturating the ground and flowing downhill, transporting matter in its path.

By using water particles with a medium size, a low coefficient of restitution, a low capacity factor, and a high deposition factor $\omega$, they transport sediment over short distances as the velocity quickly drops to 0, and ground material is completely spread along the path, since it is easier to deposit the same amount of sediment as the eroded amount at each collision point. Reducing the density of the particle simulates a rise in viscosity in the settling velocity formula, again increasing the quantity of matter to deposit at contact with the ground. By this means, we can simulate landslides as illustrated in Figure 5.40. A smoother surface results compared to rain erosion, as the rills are filled with sediment as soon as they begin to form. By setting the initial capacity of the particle equal to 10% of its max capacity, the mass of the terrain increases, simulating a volcanic eruption as illustrated in Figure 5.41.

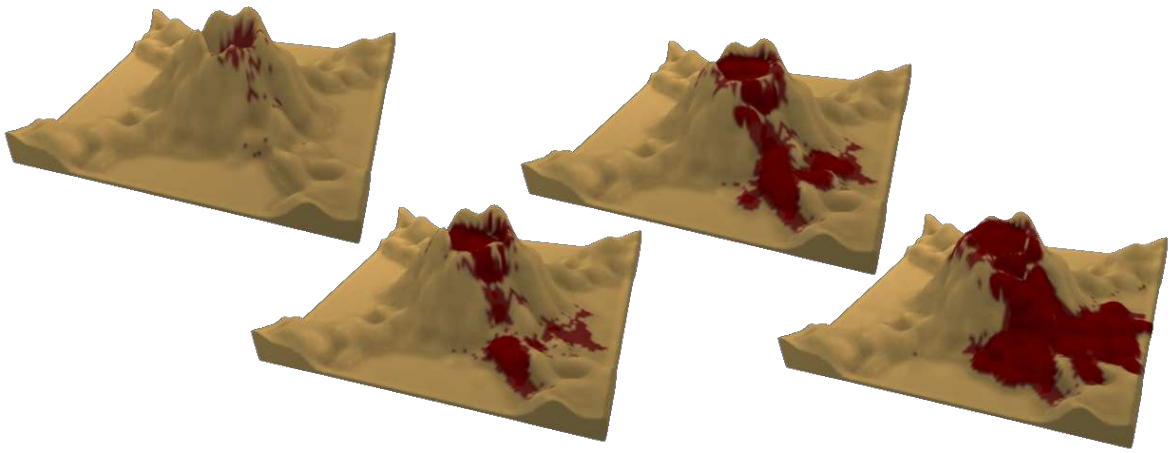


**Figure 5.41:** *A lava flow emulation by the emission of particles already containing matter from the crater center.*

### 5.5.5 Karsts

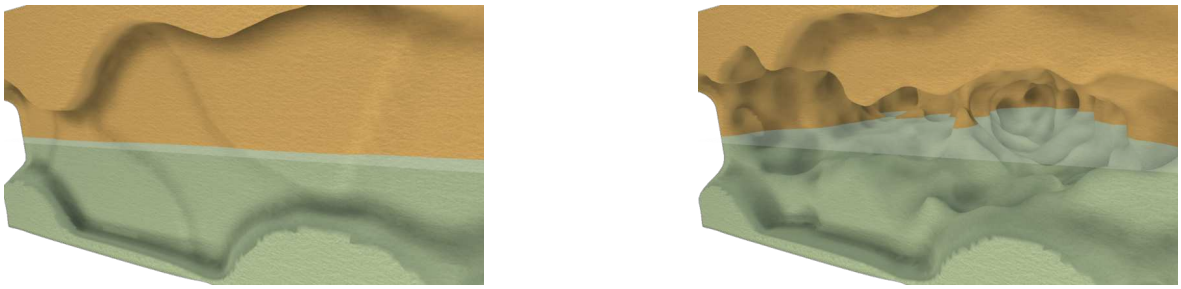


**Figure 5.42:** *Karstic cavities dug from a plain terrain on a binary-voxel grid representation.*

Karst networks are created over hundreds of years from the corrosion of water on the limestone in the ground. A limited number of methods have been proposed for the procedural generation of karsts [PGP*21].

By reducing the deposition factor $\omega$, the particles simulate corrosion (without mass conservation). We use the same particle parameters as in coastal erosion (large size, a density between air and water densities, a high capacity factor, and a low $\omega$) and optionally provide a 3D shear stress map. The karst will automatically follow the softest materials, which is geologically coherent, as given in the example in Figure 5.42, where we observe a "pillar" formed in the centre, and thus the karst forms two corridors that finally merge partially.

Underground results are only available for representations allowing 3D structures. Another underground terrain simulation is shown in Figure 5.43, in which a water runoff is eroding a tunnel without the use of a fluid simulation. Here, when particles bounce frequently on the terrain surface, the coefficient of restitution may be interpreted as a viscosity parameter.

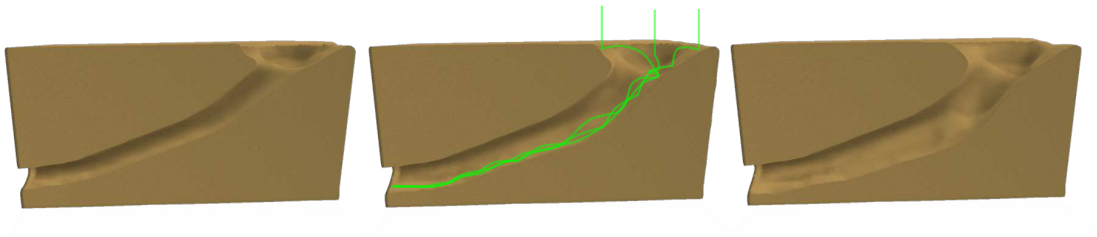


**Figure 5.43:** *Particles emitted from the entry of a tunnel dig their way inside the volumetric conduit. Green lines: example of particle trajectories.*

### 5.5.6 Wind

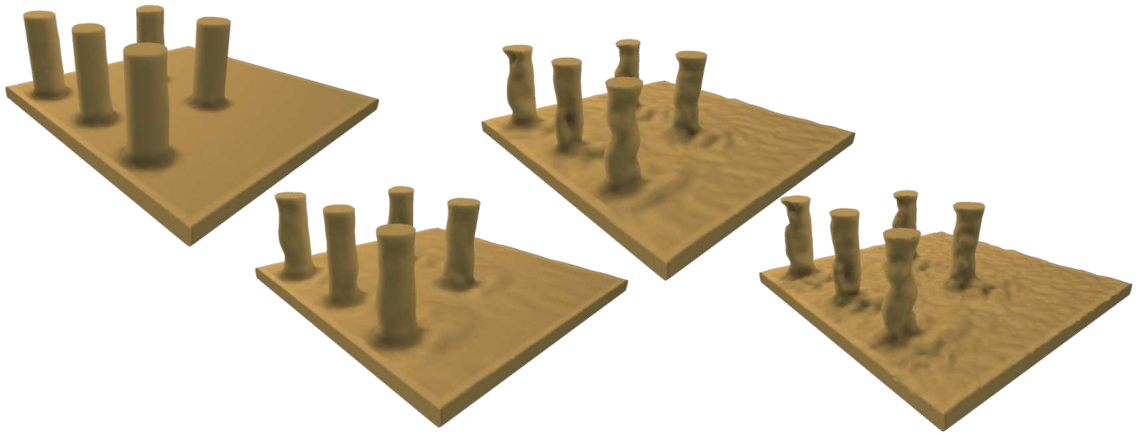


**Figure 5.44:** *High wind erosion applied on a density-voxel grid simulating abrasion on static pillars using a uniform flow field. Sand is naturally deposited at in front and on the sides of the obstacles.*

Wind erosion is a significant process in desertscape shaping, since there are no obstacles in the airflow path. Air particles can reach high velocities, transporting sand over long distances, forming either dunes or blasting into rocks, eroding them into goblins.

By setting the density of our particles close to $1\,\mathrm{kg\,m^{-3}}$, two erosion simulations can be applied at once. Air particles follow closely the flow field given by the user. This flow field can be provided by a complex simulation, a user-defined wind rose [PPG*19], or a random flow field with a general direction.

The generation of different sand structures depends on the velocity field provided, and a simple field will easily generate linear dunes. On contact with a rock block, the simulation will automatically erode block borders, creating shapes resembling goblins.

Figure 5.44 gives an example of wind erosion on a flat surface with rock columns being eroded. Given a strong 2D velocity field computed by the high wind simulation proposed in [PPG*19] and used on light particles, the simulation is fast thanks to the low number of collisions each particle has with the ground.

### 5.5.7 Underwater currents

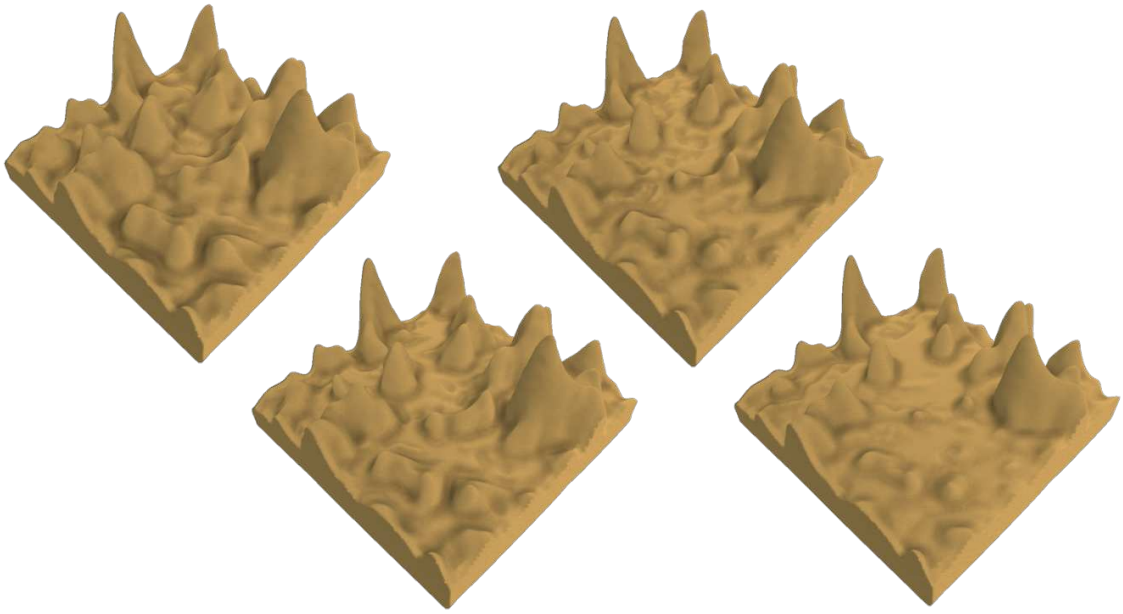


**Figure 5.45:** *Simulation of underwater erosion rapidly deposits sediments, filling in hollows and smoothing the surface on a height field.*

Procedural generation of underwater 3D terrains has received little attention. The difference between underwater and surface environments lies in the buoyancy force, which is much stronger, meaning that water flow has a much more significant effect on erosion than wind. Taking into account the density of the environment and the velocity field of water in our formulas is key to being able to apply erosion in this environment. Our method works in a water environment by assigning at least water density to particles. Given a velocity field describing underwater currents from a complex simulation or from a sketch, the particle system erodes the terrain.

In the example presented in Figure 5.45, the velocity field is given by a simple 3D fluid simulation [Sta99] applied to the terrain.

A complex water flow simulation is computed using SIMPLE [CGPS73] fluid simulation with OpenFOAM. The resulting erosion can then follow complex water movement and erode the terrain at the most affected parts of the 3D terrain, as the trajectories of the particles (green) are highly affected by the fluid velocity (blue). The densities of the particles and the environment being close, the buoyancy cancels most of the gravity force, leaving the velocity of the particles computed by the fluid velocity $v_{\text{fluid}}$ (Figure 5.46).

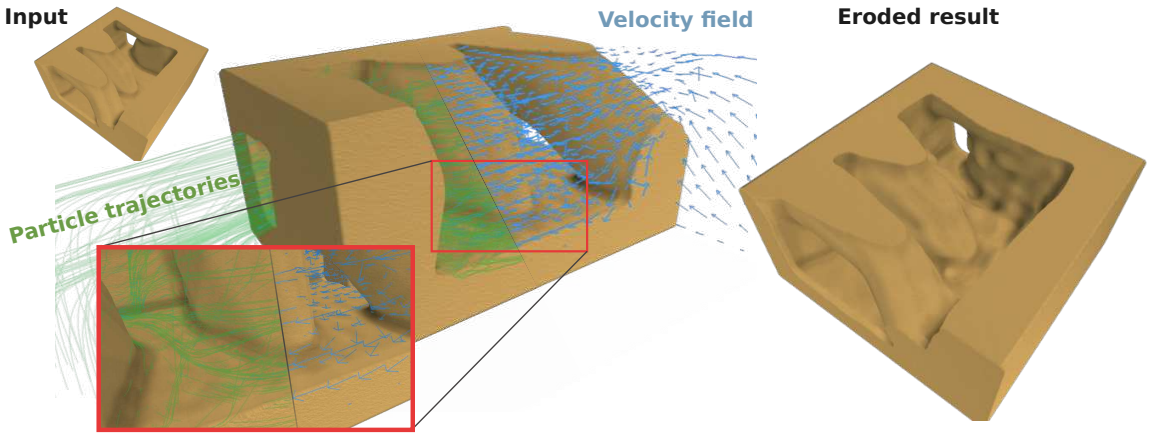



**Figure 5.46:** *A complex water flow simulation is computed using OpenFOAM. Particle trajectories (green) are highly affected by the fluid velocity (blue). Most of the terrain's exposed surfaces are eroded (bottom).*

| **Name** | **Rep.** | **Dimensions** | **Res** | $\#P$ | $\#N$ | $R$ | $COR$ | $\rho_{\textbf{particle}}$ | $C_{\textbf{factor}}$ | $\varepsilon$ | $\omega$ | **Vel field** | $t$ | Figure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rain | H | 100x100 | 20 | 100 | 10 | 1.0 | 1.0 | 1000 | 10.0 | 2.5 | 0.3 | None | 4.0 | Figure 5.36 |
| Coastal | DV | 100x100x30 | 10 | 80 | 3 | 5 | 0.1 | 500 | 10.0 | 5.0 | 0.5 | Uniform | 0.5 | Figure 5.37 |
| Meanders | I | N/A | N/A | 10 | 20 | 5.0 | 1.0 | 1000 | 1.0 | 1.0 | 1.0 | (1) | 1 | Figure 5.38 |
| River | L | 100x100 | 5 | 100 | 50 | 1.5-5 | 0.5 | 900 | 0.1 | 1.0 | 1.0 | None | 2.5 | Figure 5.39 |
| Landslide | H | 100x100 | 20 | 200 | 10 | 2.5 | 0.2 | 500 | 0.1 | 1.0 | 1.0 | None | 4 | Figure 5.40 |
| Volcano | DV | 100x100x40 | 50 | 150 | 30 | 1.0 | 5.0 | 2000 | 1.0 | 1.0 | 5.0 | None | 0.8 | Figure 5.41 |
| Karst | BV | 100x100x50 | 2 | 1000 | 40 | 5 | 0.5 | 500 | 10.0 | 5.0 | 0.5 | Uniform | 20 | Figure 5.42 |
| Tunnel | DV | 100x100x50 | 1 | 100 | 100 | 2.5 | 0.1 | 500 | 1.0 | 1.0 | 1.0 | None | 0.8 | Figure 5.43 |
| Wind | DV | 100x100x50 | 0.2 | 100 | 10 | 1.5 | 0.9 | 1.5 | 1.0 | 1.0 | 1.0 | [PPG*19] | 0.5 | Figure 5.44 |
| Underwater | H | 100x100 | 10 | 100 | 50 | 2.5 | 0.9 | 1000 | 1.0 | 1.0 | 1.0 | [Sta03] | 4 | Figure 5.45 |

Table 5.2: Parameters used for the generation of the terrains presented in Section 5.5, with "Rep" the representation (H: Height field, DV: Density voxels, BV: Binary voxels, L: Layered, I: Implicit), "Res" the resolution in metres per voxel or cell, $\#P$ the number of particles per iteration, $\#N$ the number of iterations, $R$ the particle radius (in voxel or cell unit), $COR$ the coefficient of restitution, $\rho_{\text{particle}}$ the particle density in $\mathrm{kg\,m^{-3}}$, $C_{\text{factor}}$, $\varepsilon$ and $\omega$ respectively the capacity, erosion and deposition factors, "Vel field" the type of velocity field used, and $t$ the computation time of the simulation in seconds on CPU. (1) The velocity field is a vector field defined as $v_{\text{fluid}}(\mathbf{p}) = [0\ \sin(\mathbf{p}_x)\ 0]^\intercal$.

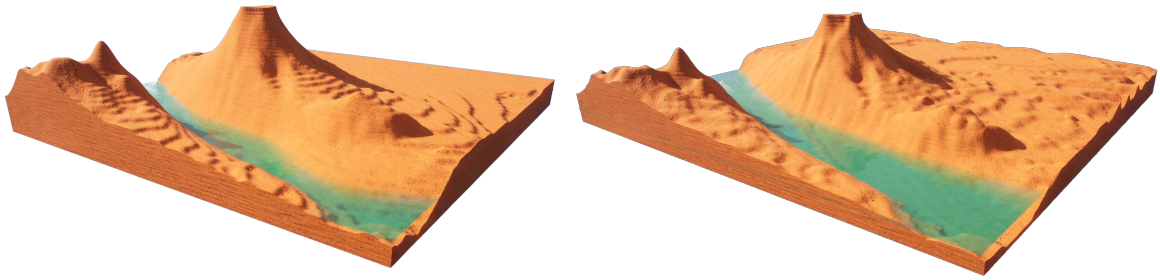


**Figure 5.47:** *Combination of multiple erosion types. On an initial synthetic 500x500x50 density voxel grid, wind erosion is applied on the surface of the terrain while hydraulic erosion shapes the rills and the base of the mountains. A water current digs its borders and spreads sediment at the bottom.*

### 5.5.8 Multiple phenomena

A terrain eroded with multiple erosion phenomena applied on a 500x500x50 density-voxel grid is illustrated in Figure 5.47. Here, water-density particles apply rain on the terrain while the coasts of the river are eroded due to a velocity field defined at the water level. The velocity field defined in the air mainly affects particles with air density, allowing wind erosion to be applied at the same time. The computation of these effects took 7 seconds on CPU.

The particle density directly influence the impact of the velocity field on their trajectory, as we can see in Figure 5.31. Denser particles fall to the bottom of the basin while lighter particles are transported by the wind, increasing the abrasion on the windward slopes and leaving the shaded slopes untouched.

## 5.6 Comparisons

In the following section, we compare our method with existing ones to show that while we are versatile on terrain representation, we are also able to reproduce various effects without applying specific algorithms. The other works are displayed in blue to distinguish them from ours.

### 5.6.1 Coastal erosion on implicit terrain representation

Paris et al. present an erosion simulation method applied to implicit terrains able to create coastal erosion, karsts and caves by adding negative sphere primitives in the terrain's construction tree [PGP*19]. The positions of the spheres are determined using Poisson disk sampling at the weakest terrain areas defined by the Geology tree of their model. They simulate the corrosion effect of water on rocks. Our work is also able to approximate this phenomenon by defining the position of these sphere primitives at the locations where the water particles hit the surface. While the computation time for the positions of the spheres is higher due to the fact that we evaluate the position of our particles at every time step in the implicit model (which could be improved by triangulation of the implicit surface, or better, a dynamic triangulation), the distribution of our erosion primitives is based on a physical model instead of a mathematical one. This means we can more easily integrate the direction and strength of waves, for example. The management of their sphere primitives can be replicated with our method by considering that a particle exists until a collision occurs, at which point it disappears. Their method does not conserve the mass of the terrain, which is acceptable for corrosion simulation but limits its validity for other erosion simulations. In our method, the particle can be tracked until it settles, ensuring mass conservation. Figure 5.48 displays a comparison of our method with [PGP*19]. Observing real world eroded cliffs in Figure 5.49, our method is able to emulate both plateforms and cavities.

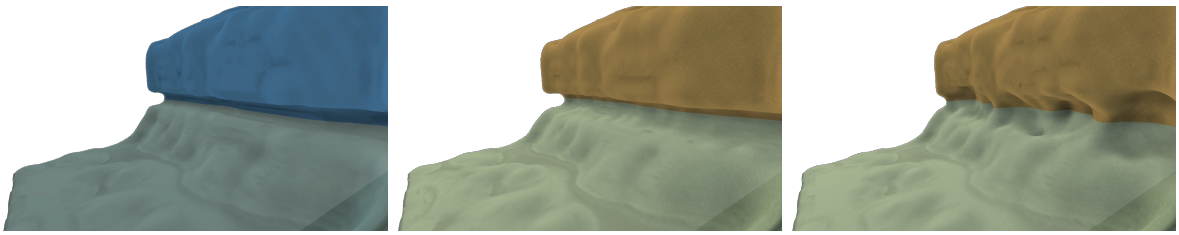


**Figure 5.48:** *The algorithm proposed by [PGP*19] allows for the simulation of coastal erosion (left), which we can reproduce almost identically by allowing our particles to collide only once with the ground and applying only erosion (centre). If we apply our erosion with full tracking of our particles and using deposition, we can achieve more diverse results (right).*

Abrasion platform associated with the present sea-level in Cyprus [GŞS*16]

John Smith's Bay cave in Bermuda (Photo credit: Akil Simmons)

**Figure 5.49:** *(Left) Paris et al. emulate platform coastal erosion, (right) but our method is also able to reproduce the irregular wave-cuts in reef and cliff erosions.*

### 5.6.2 Weathering on voxel grid representation

Beardall et al. propose a weathering erosion on voxel grids by approximating and eroding continuously the most exposed voxels [BBFJ10]. When a solid voxel is decimated, it is considered deposited and is displaced down the slope until a minimal talus angle in the terrain is reached. If the deposition is eroded again, it disappears. Our work is able to reproduce their algorithm by sending our particles from a close distance to the terrain surface. By doing so, we reproduce both the erosion and the deposition processes since the air particles, filled with sediment, fall automatically towards the local minimum of the erosion point. Similarly as their work, we can easily define the resistance value of materials to add diversity in the results. By adding the possibility of a wind field (even a very simple uniform vector field) to the simulation, we naturally add the wind shadowing effect that protects a goblin surrounded by larger goblins and also allows the deposit slope to align more closely with the wind direction (Figure 5.50).

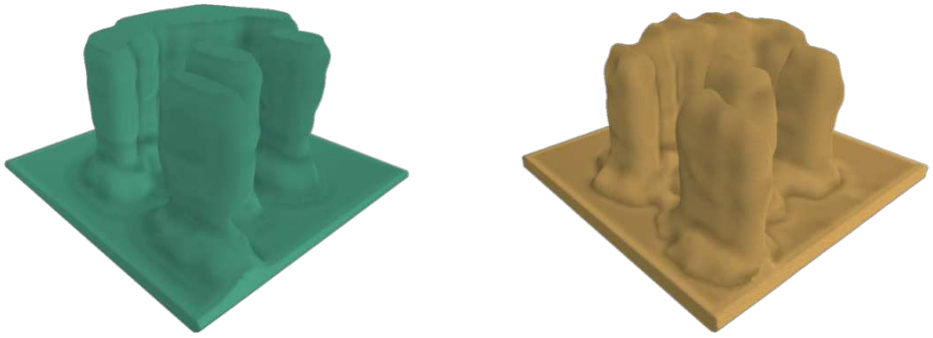


**Figure 5.50:** *The algorithm proposed by Beardall et al. allows for an efficient simulation of spheroidal erosion, making the creation of goblins on voxel grids in a plausible way (left) [BBFJ10]. Our algorithm naturally erodes the most exposed areas of the terrain when particles are affected by wind (right).*

### 5.6.3 Hydraulic erosion on height field representation

The method proposed by Mei et al. [MDH07] integrates and adapts to the GPU the pipe model proposed by O'Brien and Hodgins for fluid simulation [OH95]. This simulation is simple but efficient enough to approximate the Shallow Water Equations in real time and uses the speed of columns of water to compute the erosion and deposition rate on the 2D grid of the terrain at each time step. Using columns of water even allows the flow to overpass small bumps on the terrain over time. Our method initially relies on a stable fluid flow that is consistent throughout the lifetime of a particle, but by refining the simulation at each time step instead of at the end of the particle's lifetime, our erosion model is able to reproduce this effect, allowing the terrain to have a single batch of fluid moving through it. Our method can be seen as a generalisation of Mei et al., which can then be used on more than just discrete 2D grids. Figure 5.51 presents a comparison between [MDH07] and our method on an identical terrain. We observe that our terrain surface is not completely smooth, but rather uneven as is the case in real slopes, with rills and gullies, as shown in Figure 5.52.

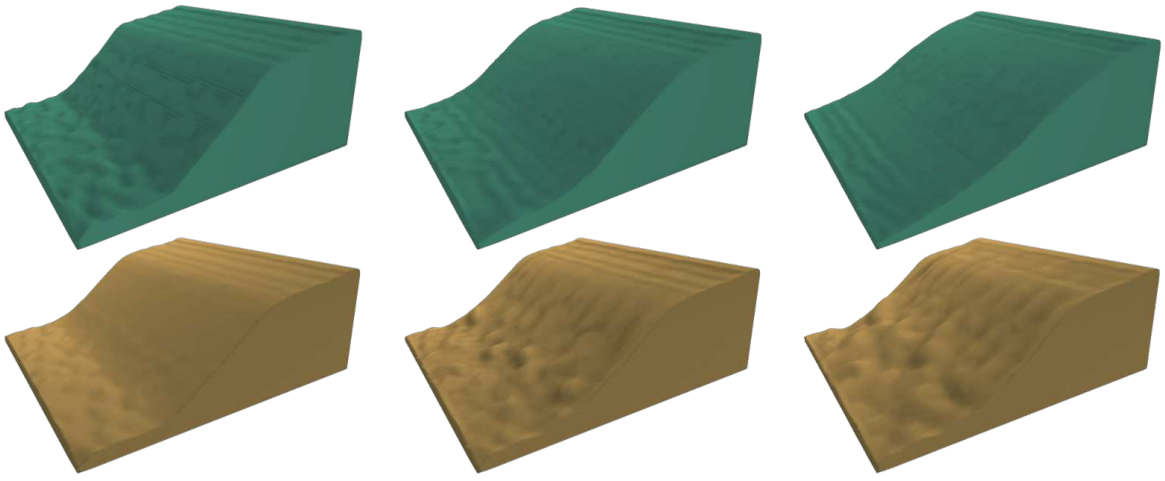


**Figure 5.51:** *While our resulting geometry for hydraulic erosion (bottom) is less smoothed than that proposed by Mei et al. (top) [MDH07], our method emulate the presence of rills at the ground surface, and allows its application on more terrain representations than just height fields.*

**Figure 5.52:** *Hydraulic erosion on slope without human intervention tends to create rills or gullies instead of smooth surfaces - Photo credit: [RSW*25] (left and center) and Geoparque Villuercas-Ibores-Jara, Portugal (right).*

### 5.6.4 Wind erosion on stacked materials representation

Paris et al. [PPG*19] simulate the effect of wind over sand fields defined on stacked materials, creating dune structures, even taking into account obstacles alike Roa and Benes [RB04] and different material layers such as vegetation [CCB*17], which are not affected by abrasion. A wind field simulation is required to produce results, and while Roa and Benes and Onoue and Nishita [ON00] consider a uniform vector field, this work considers a dynamic vector multi-scaled warped field derived from terrain height. The sand grains then apply multiple moves: sand lift, bounces, reptation and avalanching. Once the sand is lifted by the wind, the trajectory of the grains can be seen as the displacement of particles, fitting completely with our model, as illustrated in Figure 5.53.

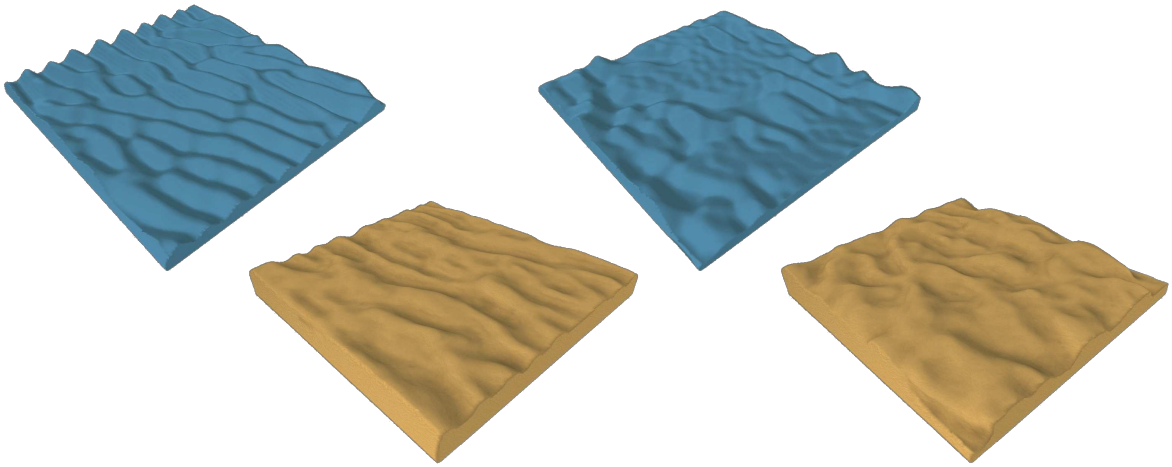


**Figure 5.53:** *The algorithm from Paris et al. allows for the generation of desertscapes (top), which we can (at least partially) reproduce with our erosion simulation (bottom). The different effects are achieved by altering the wind direction and strength [PPG*19].*

## 5.7 Conclusion

We presented a general particle-based erosion framework that decouples terrain alteration from material transport and operates uniformly across height fields, layered models, voxel grids, and implicit 3D terrains. By driving independent particles with simple physics and optionally an external velocity field, our method reproduces a wide range of geomorphological effects using a single set of abstractions and a small, intuitive parameter set: rain-driven rilling, coastal undercutting, river meanders, landslides, aeolian abrasion, and underwater smoothing.

The key advantage is representation-agnosticism: the same erosion logic applies whether the terrain is a 2.5D image or a fully volumetric implicit and voxel field. This enables users and developers to switch representations without redesigning the erosion algorithm, and to trade speed for fidelity by swapping in different flow fields (from hand-authored vectors to CFD) without modifying the erosion core.

In practice, while similar particle physics is used on different terrain representations, using similar parameters does not ensure the same eroded terrain. Surfaces and normals being approximated differently have a rippling effect on particle trajectories. Note that not all effects can be applied to all representations due to their 3D nature. For instance, karst generation, an intrinsically volumetric structure, is not applicable on 2.5D data structures.

**Realism.** The realism of the erosion simulation is highly correlated to the size and quantity of particles used and their distribution. Using too few or distributing them too sparsely results in a terrain that is unrealistic, since the alteration will have localised effects, breaking process homogeneity.

The resolution is also limited by the number and size of the particles, which can be problematic on implicit terrains that can theoretically have infinite resolution.

Our method allows erosion on implicit terrains. However, in its current form, our algorithm is time-expensive on implicit representations, since a large number of primitives are added to the composition tree. Using skeleton-defined primitives [Hon13, RGF00] from particle trajectories and erosion/deposition values could be a solution to optimise computation time.

**Usage of velocity fields.** In our erosion algorithm, we simplify particle physics to enhance computational efficiency and facilitate parameterisation. We use the velocity field from fluid simulations to approximate particle velocities. Sediment mass is harnessed to compensate for this approximation, allowing compatibility with various fluid simulation algorithms. Velocity fields can be recomputed at a frequency that meets the application's needs, ranging from "classic erosion simulation" (recomputed at each time step) to "simple simulation" (never recomputed). We addressed provisional adjustments to mitigate discrepancies when terrain changes due to erosion are not reflected in a static velocity field in section Section 5.3.3. However, it is important to note that these are expedient solutions and may not fully capture the precise dynamics of an evolving terrain.

**Performances.** To facilitate parallelisation, we intentionally overlook particle interactions and sediment exchanges, albeit at the expense of achieving smoother results. Surface collisions are simplified to basic bounces with a damping parameter instead of relying on complex particle and ground properties (Young's modulus, friction, material, ...) [YZKF20], further easing the parameterisation process. However, these simplifications, combined with the inherent discrete nature of particles, as opposed to the continuous nature of erosion, result in a correlation between realism and particle count.

The performance of our method is influenced by the time required for collision detection. Consequently, we mainly observe better performance with explicit terrain models than with implicit models.

**Particle's atomicity.** While we can replicate various effects, the "fan" shape commonly observed in natural erosion patterns is not perfectly represented. This limitation arises because we do not account for the splitting of a particle, a process that significantly influences the multidirectional dislocation and trajectory of individual particles [RTG60]. Additionally, we acknowledge an issue where particles may collide with the ceiling and the deposition becomes stuck. While a potential resolution involves splitting particles upon impact rather than simply depositing sediment, this introduces complexities to the parallelisation layer of the method. Allowing particles to split introduces unpredictability in the total number of particles that will exist in the simulation. This unpredictability complicates the use

of multi-threading. Future work includes finding a data structure allowing this splitting efficiently, leading to more realistic erosion patterns.

**Simulation with multiple materials.** One aspect we have not addressed is a layered terrain with multiple materials. In the native form of our method, we do not consider the transport of different materials (all sediment is considered as sand), but by storing a list of the different materials and the quantity transported by each particle, the same simulation process could be done at the cost of some memory and performance overhead.

Another possible adaptation of the erosion strategy for material voxels is to extend the erosion computation from binary voxels by defining transformation rules from one material to another when a voxel is eroded a number of times $\#hits < -C$ or $\#hits > C$. For example, the material "clay" may transform to "sand" when eroded, or to "rock" when many depositions occur.

# Conclusion

This thesis presents our work on the procedural generation of underwater environments, focusing particularly on coral reef islands. By situating our work at the intersection of computer graphics and marine biology, we addressed the unique challenges posed by seascapes: sparse and incomplete data, multi-scale structural complexity, dynamic biological and hydrodynamic processes, and the tension between automation and user control.

Our first contribution introduced a user-guided procedural framework for coral reef island generation (Chapter 3). The method begins with sketch-based input, where the user provides two orthogonal projections defining the island's outline and elevation. A long-term environmental deformation field representing factors like prevailing wind or current is then applied to shape the coarse geometry realistically over time. The resulting heightmaps serve as training data for a conditional Generative Adversarial Network (cGAN), which learns to produce diversified yet controllable variations of reef islands. This hybrid approach leverages both deterministic user input and stochastic neural synthesis, ensuring that the generated terrains remain ecologically plausible while enabling rapid, large-scale seascape generation.

The second contribution proposed a semantic representation of terrains using environmental objects (Chapter 4). Here, underwater features such as canyons, coral colonies, or rocky outcrops are abstracted into symbolic entities governed by rule-based interactions. Instead of simulating full multiphysics processes, these semantic objects encode simplified ecological and geological relationships. By decoupling terrain structure from heavy physical simulations, this representation enables interactive editing: users can rearrange symbolic layouts or modify ecosystem composition without committing to a specific geometric representation. The result is a fast, low-cost synthesis pipeline that preserves expert knowledge and bridges the gap between domain specialists and procedural modelling tools.

Our third contribution introduced a particle-based erosion model that is parallelisable, representation-agnostic, and applicable to both terrestrial and underwater terrains (Chapter 5). Designed with user control in mind, the model exposes intuitive parameters and guiding fields that allow users to steer erosion behaviour and achieve desired landscape outcomes by adjusting flow direction, particle emission positions and erosive agent properties. Sediment transport and deposition are simulated using discrete particles that move under simplified hydrodynamic and gravitational forces: they erode material from steep slopes, carry it along currents, and deposit it in calmer regions, reproducing processes such as coastal cliff retreat, reef slope scouring, and lagoon sedimentation. Implemented for efficient execution on modern hardware, the algorithm operates across various data representations (heightmaps, voxel grids, or layered models) while maintaining scalability. This controllable erosion stage enriches procedural terrains with naturalistic details and realistic ageing effects, reinforcing the creative and interpretative role of the user in shaping virtual environments.

Together, these contributions form a coherent pipeline that keeps the user central in the content creation loop, from large-scale landscape definition to robot-scale detail refinement. This philosophy of guiding rather than replacing the human designer responds directly to the need for controllable, transparent, and explainable virtual environments.

Beyond the specific methods presented, this work underscores a broader insight: procedural generation for underwater environments is most powerful when it blends process-informed models, simplified ecological rules, and human expertise. It is this combination that enables both scientific discovery and creative exploration in domains where direct observation is difficult or even impossible.

# Future work and research perspectives

Future work and research perspectives span several directions that extend the foundations established in this thesis. These directions aim to deepen the modular, user-centric approach to terrain generation by incorporating recent advances in data-driven modelling, procedural inference, and simulation surrogates. Each perspective addresses a distinct aspect of the pipeline. While technically diverse, these perspectives share a common goal: to enhance user control while reducing reliance on manual tuning or exhaustive simulation, and promote adaptability across creative and scientific domains.

**Inverse procedural generation** An interesting direction would be to infer procedural rules from an observed terrain and its semantic elements by inverting the symbolic framework of Chapter 4. The aim is to analyse a single scene, and abstract it as environmental objects to hypothesise placement, implicit geometry [GDGP16] and interaction rules that could generate similar patterns [ŠBM*10, ŠKC*14]. Scene analysis involves mapping observed entities and spatial distributions [EVC*15] into the symbolic vocabulary. Rule hypothesis then explores candidate constructs, followed by forward simulation to validate and adjust parameters until regenerated outputs resemble the original scene. Once validated, the inferred rules can be used to extend the terrain beyond observed boundaries, simulate temporal evolution, or perform inpainting.

**User-data-driven generation** We could develop generative models that learn terrain design style from a limited archive of prior creations of a single user by building on the sketch-based cGAN method of Chapter 3 or through neuroevolution [SBM05, CLM24] in association with the generation rules inferred on the semantic representation of the user terrains. Generating label maps from point, curve and region environmental objects may provide a way to produce a dataset with data generation and data augmentation. Once trained, freehand sketching of a label map generates the environmental objects to populate the landscape, which stays conform to prior creations. The challenge is to propose a representation scheme for the cGAN architecture that outputs environmental objects' skeleton and properties.

**Learned erosion model** A final work proposed from this thesis is the learned approximation of erosion from Chapter 5 to enable fast terrain ageing with controllable fidelity. Provided an initial flow field [TSSP17] and terrain as a voxelised image and the resulting grid after erosion iterations with varied particle counts, the training of a 3D cGAN [ÖT18] or Transformer [VSP*17] could result in a learned-based erosion model. If stable enough, this model would enable interpolation directly on a desired number of particles and wind flow direction in the way of a Nerf [MST*20]. Providing a model for each possible environmental object of a terrain could be a means to generate each 3D model plausibly.

# Publications and Conferences

The erosion method presented in Chapter 5 led to a publication in an international journal:

- **M. Hartley**, N. Mellado, C. Fiorio, N. Faraj. *Flexible Terrain Erosion*. *The Visual Computer* **40**, 4593–4607 (2024). **Best Student Paper Award.**

Additional works are currently in preparation for publication:

- **M. Hartley**, L. Jean, K. Godary-Dejean, O. Deussen, B. Benes, N. Faraj. *Procedural Underwater Terrain Generation with Environmental Objects*. *In preparation.*
- **M. Hartley**, N. Faraj. *Procedural and Learning-Based Generation of Coral Reef Islands*. *In preparation.*

These works have also been presented on multiple occasions:

- **International Conferences:**
  - **M. Hartley**, N. Mellado, C. Fiorio, N. Faraj. *Flexible Terrain Erosion*. *Computer Graphics International*, Geneva, Switzerland, 2024.
- **National Conferences:**
  - **M. Hartley**, N. Mellado, C. Fiorio, N. Faraj. *Algorithmes d'érosion pour la génération de terrains 3D*. *Journées Françaises d'Informatique Graphique*, Bordeaux, France, 2022.
  - **M. Hartley**, L. Barthe, B. Benes, O. Deussen, C. Fiorio, N. Faraj, K. Godary-Dejean. *Génération de terrains par processus itératif : application à la génération d'îles coralliennes*. *Journées Françaises d'Informatique Graphique*, Montpellier, France, 2023.
- **Workshops:**
  - **M. Hartley**. *Procedural Generation of 3D Underwater Environments*. *Computer Science and Robotics Tools for 3D Marine Underwater Environment Monitoring and Modelling*, Montpellier, France, 2022.
  - **M. Hartley**. *Procedural Generation of 3D Underwater Environments*. *Workshop on Underwater Environment Monitoring and Modelling*, Montpellier, France, 2024.

This work was carried out in collaboration with national and international partners, whom I would like to thank sincerely:

- **IRIT**, Université de Toulouse, France: with N. Mellado and L. Barthe (including a one-month visit),
- **Universität Konstanz**, Germany: with O. Deussen,
- **Purdue University**, USA: with B. Benes (including a one-month visit).

The overall goal of this work is to propose methods for generating underwater environments for mission simulation and planning in the context of autonomous robot validation. To enable practical use, our methods were integrated into a mission editing tool based on Unreal Engine 5, developed by our software engineer Q. Guillot.

# References


[AAR*24] ALVARADO, EDUARDO, ARGUDO, OSCAR, ROHMER, DAMIEN, et al. **TRAIL: Simulating the impact of human locomotion on natural landscapes**. July 2024 Cited on pages 139, 140.

[ACG22] ATAS, FETULLAH, CIELNIAK, GRZEGORZ and GRIMSTAD, LARS. "**'Elevation State-Space: Surfel-Based Navigation in Uneven Environments for Mobile Robots'**". *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2022-October. Institute of Electrical and Electronics Engineers Inc., 2022, 5715–5721. ISBN: 9781665479271 Cited on page 24.

[AD05] ALSWEIS, MONSSEF and DEUSSEN, OLIVER. "**'Modeling and Visualization of symmetric and asymmetric plant competition'**". *Eurographics Workshop on Natural Phenomena*. Ed. by POULIN, PIERRE and GALIN, ERIC. The Eurographics Association, 2005. ISBN: 3-905673-29-0. URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-26559 Cited on pages 88, 89, 91.

[AD06] ALSWEIS, MONSSEF and DEUSSEN, OLIVER. "**'Wang-Tiles for the Simulation and Visualization of Plant Competition'**". *Proceedings of the 24th International Conference on Advances in Computer Graphics*. Springer-Verlag, 2006, 1–11. ISBN: 3-540-35638-X,978-3-540-35638-7. URL: http://link.springer.com/10.1007/11784203_1 Cited on page 89.

[AF00] ANTHONY, KENNETH R N and FABRICIUS, KATHARINA E. **Shifting roles of heterotrophy and autotrophy in coral energetics under varying turbidity**. Tech. rep. 2000, 221–253. URL: www.elsevier.nl/locate/jembe Cited on page 18.

[AFD*21] ARNONE, E., FRANCIPANE, A., DIALYNAS, Y. G., et al. "**'Implications of terrain resolution on modeling rainfall-triggered landslides using a TIN- based model'**". *Environmental Modelling and Software* 141 (July 2021). ISSN: 13648152 Cited on page 24.

[AGP*20] ARGUDO, OSCAR, GALIN, ERIC, PEYTAVIE, ADRIEN, et al. "**'Simulation, modeling and authoring of glaciers'**". *ACM Transactions on Graphics* 39 (6 Dec. 2020), 1–14. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/3414685.3417855 Cited on pages 129, 138, 140.

[ALY15] ABELA, RYAN, LIAPIS, ANTONIOS and YANNAKAKIS, GEORGIOS N. "**'A Constructive Approach for the Generation of Underwater Environments'**". *Proceedings of the FDG workshop on Procedural Content Generation in Games*. 2015. URL: https://code.google.com/p/lsystems-csharp-lib Cited on page 18.

[AM09] AMAWY, MAHER EL and MUFTAH, AHMED M. "**'Karst development and structural relationship in the tertiary rocks of the Al Jabal Al Akhdar, NE Libya: A case study in Qasr Libya area'**". *3rd International Symposium Karst Evolution in the South Mediterranean Area* 14 (Mar. 2009), 173–189 Cited on page 83.

[Bad10] BADDELEY, ADRIAN. **Analysing spatial point patterns in R**. Tech. rep. 2010 Cited on page 92.

[BBFJ10] BEARDALL, MATTHEW, BUTLER, JOSEPH, FARLEY, MCKAY and JONES, MICHAEL D. "**'Directable weathering of concave rock using curvature estimation'**". *IEEE Transactions on Visualization and Computer Graphics* 16 (1 2010), 81–94. ISSN: 10772626. URL: https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=1109&context=facpub Cited on pages 132, 135, 137, 140, 154, 167, 168.

[BBM*09] BAUER, LJ, BATTISTA, TA, MUELLER, PW, et al. **Shallow-Water Benthic Habitats of St. John, U.S. Virgin Islands**. Tech. rep. NOAA TECHNICAL MEMORANDUM NOS NCCOS 96, 2009 Cited on page 29.

[BC12] BRAUER, FRED and CASTILLO-CHAVEZ, CARLOS. **Mathematical Models in Population Biology and Epidemiology**. Vol. 40. Springer New York, 2012. ISBN: 978-1-4614-1685-2. URL: https://link.springer.com/10.1007/978-1-4614-1686-9 Cited on page 97.

[BD92] BIGING, GREG S. and DOBBERTIN, MATTHIAS. "**'A Comparison of Distance-Dependent Competition Measures for Height and Basal Area Growth of Individual Conifer Trees'**". *Forest Science* 38 (3 Aug. 1992), 695–720. ISSN: 0015-749X. URL: https://academic.oup.com/forestscience/article/38/3/695/4642125 Cited on page 90.

[Bel72] BELLA, I E. **A New Competition Model for Individual Trees**. Tech. rep. 1972. URL: https://academic.oup.com/forestscience/article/17/3/364/4709976 Cited on page 90.

[Ben21] BEN-SAID, MARIEM. **Spatial point-pattern analysis as a powerful tool in identifying pattern-process relationships in plant ecology: an updated review**. Dec. 2021 Cited on page 92.

[BF01a] BENEŠ, BEDŘICH and FORSBACH, RAFAEL. "**'Layered data representation for visual simulation of terrain erosion'**". *Proceedings - Spring Conference on Computer Graphics, SCCG 2001* (2001), 80–86. URL: https://data.exppad.com/public/papers/Layered_data_representation_for_Visual_Simulation_of_Terrain_Erosion.pdf Cited on pages 27, 132, 140, 151.

[BF01b] BENEŠ, BEDŘICH and FORSBACH, RAFAEL. "**'Parallel implementation of terrain erosion applied to the surface of Mars'**". *Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation*. ACM, Nov. 2001, 53–57. ISBN: 1581134460. URL: https://dl.acm.org/doi/10.1145/513867.513880 Cited on page 28.

[BFO*07] BEARDALL, MATTHEW, FARLEY, MCKAY, OUDERKIRK, D., et al. "**'Goblins by spheroidal weathering'**". *Natural Phenomena* (2007), 7–14. ISSN: 18160867. URL: https://diglib.eg.org/bitstream/handle/10.2312/NPH.NPH07.007-014/007-014.pdf?sequence=1&isAllowed=y#:~:text=to%20additional%20weathering.-,weathering.,Weathering%20weakens%20the%20rock. Cited on pages 135, 137.

[BGI*20] BÜRGER, RAIMUND, GAVILÁN, ELVIS, INZUNZA, DANIEL, et al. "**'Exploring a convection–diffusion–reaction model of the propagation of forest fires: Computation of risk maps for heterogeneous environments'**". *Mathematics* 8 (10 Oct. 2020), 1–20. ISSN: 22277390 Cited on page 97.

[BH00] BERGER, UTA and HILDENBRANDT, HANNO. "**'A new approach to spatially explicit modelling of forest dynamics: spacing, ageing and neighbourhood competition of mangrove trees'**". *Ecological Modelling* 132 (3 Aug. 2000), 287–302. ISSN: 03043800. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380000002982 Cited on pages 90, 91.

[BHN07] BRIDSON, ROBERT, HOURIHAM, JIM and NORDENSTAM, MARCUS. "**'Curl-noise for procedural fluid flow'**". *ACM Transactions on Graphics* 26 (99 2007), 46. ISSN: 07300301 Cited on page 144.

[BKR88] BRACKBILL, J. U., KOTHE, D. B. and RUPPEL, H. M. "**'Flip: A low-dissipation, particle-in-cell method for fluid flow'**". *Computer Physics Communications* 48 (1 1988), 25–38. ISSN: 00104655 Cited on page 142.

[BKRE17] BECHER, MICHAEL, KRONE, MICHAEL, REINA, GUIDO and ERTL, THOMAS. "**'Feature-based volumetric terrain generation'**". *Proceedings - I3D 2017: 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2017) Cited on pages 28, 152.

[BKS*14] BALDOCK, TOM E., KARAMPOUR, HASSAN, SLEEP, RACHAEL, et al. "**'Resilience of branching and massive corals to wave loading under sea level rise - A coupled computational fluid dynamics-structural analysis'**". *Marine Pollution Bulletin* 86 (1-2 2014), 91–101. ISSN: 18793363 Cited on page 30.

[BNK25] BRUNTON, STEVEN L, NOACK, BERND R and KOUMOUTSAKOS, PETROS. "**'Machine Learning for Fluid Mechanics'**". *Annual Review of Fluid Mechanics Downloaded from www.annualreviews.org. Guest* 7 (2025), 29. URL: https://doi.org/10.1146/annurev-fluid-010719- Cited on page 144.

[BP17] BECKHAM, CHRISTOPHER and PAL, CHRISTOPHER. "**'A step towards procedural terrain generation with GANs'**". (July 2017). URL: http://arxiv.org/abs/1707.03383 Cited on pages 50, 51.

[BP22] BOUSSANGE, VICTOR and PELLISSIER, LOÏC. "**'Eco-evolutionary model on spatial graphs reveals how habitat structure affects phenotypic differentiation'**". *Communications Biology* 5 (1 Dec. 2022). ISSN: 23993642 Cited on page 87.

[Bri07] BRIDSON, ROBERT. "**'Fast poisson disk sampling in arbitrary dimensions'**". *ACM SIGGRAPH 2007 Sketches, SIGGRAPH'07* (2007). URL: https://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph07-poissondisk.pdf Cited on page 88.

[BTHB06] BENEŠ, BEDŘICH, TĚŠÍNSKÝ, VÁCLAV, HORNYŠ, JAN and BHATIA, SANJIV K. "**'Hydraulic erosion'**". *Computer Animation and Virtual Worlds* 17 (2 2006), 99–108. ISSN: 15464261 Cited on pages 44, 134, 140.

[BW98] BARAFF, DAVID and WITKIN, ANDREW. "**'Large steps in cloth simulation'**". *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998* (1998), 43–54. URL: https://www.cs.cmu.edu/~baraff/papers/sig98.pdf Cited on page 150.

[CA09] COOK, MATTHEW T. and AGAH, ARVIN. "**'A survey of sketch-based 3-D modeling techniques'**". *Interacting with Computers* 21 (3 2009), 201–211. ISSN: 09535438 Cited on page 45.

[CARR12] CARRARA, FRANCESCO, ALTERMATT, FLORIAN, RODRIGUEZ-ITURBE, IGNACIO and RINALDO, ANDREA. "**'Dendritic connectivity controls biodiversity patterns in experimental metacommunities'**". *Proceedings of the National Academy of Sciences of the United States of America* 109 (15 Apr. 2012), 5761–5766. ISSN: 00278424 Cited on page 87.

[CBC*16] CORDONNIER, GUILLAUME, BRAUN, JEAN, CANI, MARIE-PAULE, et al. "**'Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion'**". *Computer Graphics Forum* 35 (2 May 2016), 165–175. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12820 Cited on pages 44, 134.

[CBL*09] CHANG, YUANZHANG, BAO, KAI, LIU, YOUQUAN, et al. "**'A particle-based method for viscoelastic fluids animation'**". *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. ACM, Nov. 2009, 111–117. ISBN: 9781605588698. URL: https://dl.acm.org/doi/10.1145/1643928.1643954 Cited on page 139.

[CC06] CASTLE, CHRISTIAN J E and CROOKS, ANDREW T. "**'Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations'**". Sept. 2006 Cited on page 86.

[CCB*17] CORDONNIER, GUILLAUME, CANI, MARIE-PAULE, BENEŠ, BEDŘICH, et al. "**'Sculpting Mountains: Interactive Terrain Modeling Based on Subsurface Geology'**". *IEEE Transactions on Visualization and Computer Graphics* 24 (2017). URL: http://www.ieee.org/publications_standards/publications/rights/index.html Cited on pages 44, 129, 169.

[CCK*21] CORNWALL, CHRISTOPHER E., COMEAU, STEEVE, KORNDER, NIKLAS A., et al. "**'Global declines in coral reef calcium carbonate production under ocean acidification and warming'**". *Proceedings of the National Academy of Sciences* 118 (21 May 2021). ISSN: 0027-8424. URL: https://pnas.org/doi/full/10.1073/pnas.2015265118 Cited on page 31.

[CCR10] CANTRELL, STEPHEN, COSNER, CHRIS and RUAN, S A SHIGUI. **Spatial Ecology**. 2010 Cited on page 85.

[CD98] CHEN, SHIYI and DOOLEN, GARY D. "**'Lattice Boltzmann method for fluid flows'**". *Annual Review of Fluid Mechanics* 30 (1 Jan. 1998), 329–364. ISSN: 0066-4189. URL: https://www.annualreviews.org/doi/10.1146/annurev.fluid.30.1.329 Cited on page 141.

[CDP*17] CHEVALIER, C., DEVENON, J. L., PAGANO, M., et al. "**'The atypical hydrodynamics of the Mayotte Lagoon (Indian Ocean): Effects on water age and potential impact on plankton productivity'**". *Estuarine, Coastal and Shelf Science* 196 (2017), 182–197. ISSN: 02727714 Cited on page 32.

[CDRB15] CHEVALIER, CRISTELE, DEVENON, JEAN LUC, ROUGIER, GILLES and BLANCHOT, JEAN. "**'Hydrodynamics of the Toliara Reef Lagoon (Madagascar): Example of a Lagoon Influenced by Waves and Tides'**". *Journal of Coastal Research* 31 (6 2015), 1403–1416. ISSN: 15515036 Cited on page 32.

[CEG*18] CORDONNIER, GUILLAUME, ECORMIER-NOCCA, PIERRE, GALIN, ÉRIC, et al. "**'Interactive generation of time-evolving, snow-covered landscapes with avalanches'**". *Computer Graphics Forum* 37 (2 2018), 497–509. ISSN: 14678659. URL: https://hal.inria.fr/hal-01736971/file/interactive-generation-time.pdf Cited on pages 129, 139, 140.

[CGG*17] CORDONNIER, GUILLAUME, GALIN, ÉRIC, GAIN, JAMES, et al. "**'Authoring landscapes by combining ecosystem and terrain erosion simulation'**". *ACM Transactions on Graphics* 36 (4 2017). ISSN: 15577368. URL: https://hal.archives-ouvertes.fr/hal-01518967/file/authoring-landscapes-combining.pdf Cited on pages 44, 129, 139.

[CGPS73] CARETTO, L. S., GOSMAN, A. D., PATANKAR, SUHAS V. and SPALDING, D. B. "**'Two calculation procedures for steady, three-dimensional flows with recirculation'**". *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Vol. 19. 1973, 60–68. ISBN: 978-3-540-06171-7. URL: http://www.springerlink.com/content/x46n27271791j492/ Cited on page 163.

[CHCC21] CHEN, CHIEN WEN, HU, MIN CHUN, CHU, WEI TA and CHEN, JUN CHENG. "**'A Real-Time Sculpting and Terrain Generation System for Interactive Content Creation'**". *IEEE Access* 9 (2021), 114914–114928. ISSN: 21693536 Cited on page 28.

[CHM17] CROOKS, ANDREW T, HEPPENSTALL, ALISON and MALLESON, NICK. "**'Agent-Based Modeling'**". *Comprehensive Geographic Information Systems*. Vol. 3. Elsevier Inc., July 2017, 218–243. ISBN: 9780128046609 Cited on page 87.

[Chn11] CH'NG, EUGENE. "**'An Artificial Life-Based Vegetation Modelling Approach for Biodiversity Research'**". *Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery*. IGI Global, 2011, 68–118. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-705-8.ch004 Cited on page 93.

[Chn13] CH'NG, EUGENE. "**'Model resolution in complex systems simulation: Agent preferences, behavior, dynamics and n-tiered networks'**". *SIMULATION* 89 (5 May 2013), 635–659. ISSN: 0037-5497. URL: http://journals.sagepub.com/doi/10.1177/0037549712470582 Cited on pages 87, 88.

[CJP*23] CORDONNIER, GUILLAUME, JOUVET, GUILLAUME, PEYTAVIE, ADRIEN, et al. "**'Forming Terrains by Glacial Erosion'**". *ACM Transactions on Graphics* 42 (4 2023), 14. URL: https://doi.org/10.1145/3592422. Cited on pages 129, 138, 140.

[CLM24] CORTÊS, GABRIEL, LOURENÇO, NUNO and MACHADO, PENOUSAL. "**'Towards Physical Plausibility in Neuroevolution Systems'**". Vol. 2. Apr. 2024, 76–90. URL: https://link.springer.com/10.1007/978-3-031-56855-8_5 Cited on page 174.

[CLR*20] COWARD, GEORGIA, LAWRENCE, ALICE, RIPLEY, NATASHA, et al. "**'A new record for a massive Porites colony at Ta'u Island, American Samoa'**". *Scientific Reports* 10 (1 Dec. 2020). ISSN: 20452322 Cited on page 30.

[CLY*24] CHEN, MEIDA, LAL, DEVASHISH, YU, ZIFAN, et al. "**'Large-Scale 3D Terrain Reconstruction Using 3D Gaussian Splatting for Visualization and Simulation'**". *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. Vol. 48-2-2024. International Society for Photogrammetry and Remote Sensing, June 2024, 49–54 Cited on page 24.

[CR74] CATMULL, EDWIN and ROM, RAPHAEL. "**'Class of local interpolating splines'**". *Computer Aided Geometric Design*. Academic Press, 1974, 317–326 Cited on page 58.

[CSB*25] CHEN, TIMOTHY, SHORINWA, OLA, BRUNO, JOSEPH, et al. "**'Splat-Nav: Safe Real-Time Robot Navigation in Gaussian Splatting Maps'**". *IEEE Transactions on Robotics* 41 (2025), 2765–2784. ISSN: 19410468 Cited on page 24.

[CV01] CHAN, TONY F and VESE, LUMINITA A. "**'Active Contours Without Edges'**". *IEEE Transactions on image processing* 10 (2 2001) Cited on pages 106, 107.

[CWC10] COWART, LISA, WALSH, J. P. and CORBETT, D. REIDE. "**'Analyzing Estuarine Shoreline Change: A Case Study of Cedar Island, North Carolina'**". *Journal of Coastal Research* 265 (September 2010), 817–830. ISSN: 0749-0208 Cited on page 120.

[CWL*23] CHEN, KAI, WANG, CHUN, LU, MINGYUE, et al. "**'Integrating Topographic Skeleton into Deep Learning for Terrain Reconstruction from GDEM and Google Earth Image'**". *Remote Sensing* 15 (18 Sept. 2023). ISSN: 20724292 Cited on page 15.

[CY76] CHWANG, ALLEN T and YAO-TSU, T. **Hydromechanics of low-Reynolds-number flow. Part 2. Singularity method for Stokes flows**. Tech. rep. 1976, 787–815 Cited on page 144.

[Cza98] CZARAN, TAMAS. **Spatiotemporal models of population and community dynamics**. Chapman & Hall, Jan. 1998, 284. ISBN: 0412575507 Cited on pages 84, 85, 89.

[DAG05] DOMÍNGUEZ, L., ANFUSO, G. and GRACIA, F. J. "**'Vulnerability assessment of a retreating coast in SW Spain'**". *Environmental Geology* 47 (8 2005), 1037–1044. ISSN: 09430105 Cited on page 120.

[Dal15] DALY, REGINALD A . "**'The Glacial-Control Theory of Coral Reefs'**". *Proceedings of the American Academy of Arts and Sciences* 51 (4 1915), 157–251. URL: http://www.jstor.org/s Cited on page 40.

[dALJ*15] De ARAÚJO, B. R., LOPES, DANIEL S., JEPP, PAULINE, et al. "**'A Survey on Implicit Surface Polygonization'**". *ACM Computing Surveys* 47 (4 July 2015), 1–39. ISSN: 0360-0300. URL: https://dl.acm.org/doi/10.1145/2732197 Cited on page 26.

[DARB18] DUFLOT, RÉMI, AVON, CATHERINE, ROCHE, PHILIP and BERGÈS, LAURENT. "**'Combining habitat suitability models and spatial graphs for more effective landscape conservation planning: An applied methodological framework and a species case study'**". *Journal for Nature Conservation* 46 (Dec. 2018), 38–47. ISSN: 16171381 Cited on page 87.

[Das12] DAS, ADRIAN. "**'The effect of size and competition on tree growth rate in old-growth coniferous forests'**". *Canadian Journal of Forest Research* 42 (11 Nov. 2012), 1983–1995. ISSN: 00455067 Cited on page 89.

[dCB09] De CARPENTIER, GILIAM J. P. and BIDARRA, RAFAEL. "**'Interactive GPU-based procedural heightfield brushes'**". *Proceedings of the 4th International Conference on Foundations of Digital Games*. Vol. 8. ACM, Apr. 2009, 55–62. ISBN: 9781605584379. URL: https://dl.acm.org/doi/10.1145/1536513.1536532 Cited on page 28.

[DGG24] DAI, ADAM, GUPTA, SHUBH and GAO, GRACE. "**'Neural Elevation Models for Terrain Mapping and Path Planning'**". (May 2024). URL: http://arxiv.org/abs/2405.15227 Cited on page 24.

[DHL*98] DEUSSEN, OLIVER, HANRAHAN, PAT, LINTERMANN, BERND, et al. "**'Realistic modeling and rendering of plant ecosystems'**". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. ACM Press, 1998, 275–286. ISBN: 0897919998. URL: http://portal.acm.org/citation.cfm?doid=280814.280898 Cited on page 88.

[DHOS00] DEUSSEN, OLIVER, HILLER, STEFAN, OVERVELD, CORNELIUS VAN and STROTHOTTE, THOMAS. "**'Floating points: a method for computing stipple drawings'**". *Computer Graphics Forum* 19 (3 Aug. 2000), 41–50. ISSN: 01677055 Cited on page 88.

[DJ21] DROXLER, ANDRÉ W and JORRY, STÉPHAN J. "**'The Origin of Modern Atolls : Challenging Darwin's Deeply Ingrained Theory'**". *Annual Review of Marine Science* (2021) Cited on page 40.

[DJ42] DARWIN, CHARLES and JACKSON, COLONEL. "**'The Structure and Distribution of Coral Reefs: Being the First Part of the Geology of the Voyage of the 'Beagle,' under the Command of Capt. Fitzroy, R. N., during the Years 1832 to 1836'**". *The Structure and Distribution of Coral Reefs*. Vol. 12. 1842, 115 Cited on pages 32, 37.

[DL94] DURRETT, R. and LEVIN, S. "**'The Importance of Being Discrete (and Spatial)'**". *Theoretical Population Biology* 46 (3 Dec. 1994), 363–394. ISSN: 00405809. URL: https://linkinghub.elsevier.com/retrieve/pii/S004058098471032X Cited on page 85.

[DLHT14] DONG, CHAO, LOY, CHEN CHANGE, HE, KAIMING and TANG, XIAOOU. "**'Image Super-Resolution Using Deep Convolutional Networks'**". (Dec. 2014). URL: http://arxiv.org/abs/1501.00092 Cited on page 78.

[EBM04] EDGAR, GRAHAM J., BARRETT, NEVILLE S. and MORTON, ALASTAIR J. "**'Biases associated with the use of underwater visual census techniques to quantify the density and size-structure of fish populations'**". *Journal of Experimental Marine Biology and Ecology* 308 (2 Sept. 2004), 269–290. ISSN: 00220981 Cited on page 13.

[ECC*21] ECORMIER-NOCCA, PIERRE, CORDONNIER, GUILLAUME, CARREZ, PHILIPPE, et al. "**'Authoring consistent landscapes with flora and fauna'**". *ACM Transactions on Graphics* 40 (4 2021). ISSN: 15577368. URL: https://www-sop.inria.fr/reves/Basilic/2021/ECCMMBC21/Authoring_Consistent_Landscapes_with_Flora_and_Fauna.pdf Cited on pages 44, 87, 139, 140.

[EDP*11] EBEIDA, MOHAMED S., DAVIDSON, ANDREW A., PATNEY, ANJUL, et al. "**'Efficient maximal poisson-disk sampling'**". *ACM Transactions on Graphics* 30 (4 July 2011). ISSN: 07300301 Cited on page 88.

[EPCV15] EMILIEN, ARNAUD, POULIN, PIERRE, CANI, MARIE-PAULE and VIMONT, ULYSSE. "**'Interactive Procedural Modelling of Coherent Waterfall Scenes'**". *Computer Graphics Forum* 34 (6 Sept. 2015), 22–35. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12515 Cited on pages 96, 134.

[EPW*03] EBERT, DAVID S., PEACHEY, DARWYN, WORLEY, STEVEN, et al. **Texturing and Modeling, A Procedural Approach**. Vol. Third edition. Morgan Kaufmann, 2003. ISBN: 978-1-55860-848-1 Cited on pages 27, 42.

[EVC*15] EMILIEN, ARNAUD, VIMONT, ULYSSE, CANI, MARIE-PAULE, et al. "**'WorldBrush'**". *ACM Transactions on Graphics* 34 (4 July 2015), 1–11. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/2766975 Cited on pages 92, 174.

[FBS*14] FERRARIO, FILIPPO, BECK, MICHAEL W., STORLAZZI, CURT D., et al. "**'The effectiveness of coral reefs for coastal hazard risk reduction and adaptation'**". *Nature Communications* 5 (May 2014). ISSN: 20411723 Cited on page 13.

[FEHM19] FLYNN, SEAN, EGBERT, PARRIS, HOLLADAY, SETH and MORSE, BRYAN. "**'Fluid Carving: Intelligent Resizing for Fluid Simulation Data'**". *ACM Trans. Graph* 38 (2019), 14. URL: https://doi.org/10.1145/3355089.3356572 Cited on page 144.

[FFC82] FOURNIER, ALAIN, FUSSELL, DON and CARPENTER, LOREN. "**'Computer rendering of stochastic models'**". *Communications of the ACM* 25 (6 June 1982), 371–384. ISSN: 0001-0782. URL: https://dl.acm.org/doi/10.1145/358523.358553 Cited on pages 15, 28, 42.

[FFRL24] FENG, RUOFENG, FOURTAKAS, GEORGIOS, ROGERS, BENEDICT D. and LOMBARDI, DOMENICO. "**'Modelling internal erosion using 2D smoothed particle hydrodynamics (SPH)'**". *Journal of Hydrology* 639 (Aug. 2024). ISSN: 00221694 Cited on page 131.

[FN20] FOROOTANINIA, ZAHRA and NARAIN, RAHUL. "**'Frequency-domain smoke guiding'**". *ACM Transactions on Graphics* 39 (6 Nov. 2020). ISSN: 15577368 Cited on page 144.

[Fol19] FOLTÊTE, JEAN CHRISTOPHE. "**'How ecological networks could benefit from landscape graphs: A response to the paper by Spartaco Gippoliti and Corrado Battisti'**". *Land Use Policy* 80 (Jan. 2019), 391–394. ISSN: 02648377 Cited on page 86.

[FPRJ00] FRISKEN, SARAH F, PERRY, RONALD N, ROCKWOOD, ALYN P and JONES, THOUIS R. **Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics**. Tech. rep. 2000 Cited on pages 26, 27, 94.

[Gat00] GATTO, LAWRENCE W. "**'Soil freeze–thaw-induced changes to a simulated rill: potential impacts on soil erosion'**". *Geomorphology* 32 (1-2 Feb. 2000), 147–160. ISSN: 0169555X. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169555X99000926 Cited on page 134.

[GBH*12] GAUCHEREL, CÉDRIC, BOUDON, FRÉDÉRIC, HOUET, THOMAS, et al. "**'Understanding Patchy Landscape Dynamics: Towards a Landscape Language'**". *PLoS ONE* 7 (9 2012). ISSN: 19326203. URL: https://halshs.archives-ouvertes.fr/halshs-00750971/file/Gaucherel_al-PONE.pdf Cited on page 87.

[GBR*16] GONZÁLEZ-RIVERO, MANUEL, BEIJBOM, OSCAR, RODRIGUEZ-RAMIREZ, ALBERTO, et al. "**'Scaling up ecological measurements of coral reefs using semi-automated field image collection and analysis'**". *Remote Sensing* 8 (1 2016). ISSN: 20724292 Cited on page 17.

[GCRR20] GASCH, CRISTINA, CHOVER, MIGUEL, REMOLAR, INMACULADA and REBOLLO, CRISTINA. "**'Procedural modelling of terrains with constraints'**". *Multimedia Tools and Applications* 79 (41-42 2020), 31125–31146. ISSN: 15737721 Cited on page 47.

[GDG*17] GUÉRIN, ÉRIC, DIGNE, JULIE, GALIN, ÉRIC, et al. "**'Interactive example-based terrain authoring with conditional generative adversarial networks'**". *ACM Transactions on Graphics* 36 (6 2017). ISSN: 15577368 Cited on page 51.

[GDGP16] GUÉRIN, ÉRIC, DIGNE, JULIE, GALIN, ÉRIC and PEYTAVIE, ADRIEN. "**'Sparse representation of terrains for procedural modeling'**". *Computer Graphics Forum* 35 (2 2016), 177–187. ISSN: 14678659 Cited on pages 78, 174.

[GEB15] GATYS, LEON A., ECKER, ALEXANDER S. and BETHGE, MATTHIAS. "**'A Neural Algorithm of Artistic Style'**". (Aug. 2015). URL: http://arxiv.org/abs/1508.06576 Cited on page 78.

[Ger25] GERIGK, CHRISTOPH. "**'Underwater Digital Twin of Yuba Island: Mapping a Large-Scale Marine Ecosystem through Photogrammetry and 3D Modeling'**". *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. Vol. 48. International Society for Photogrammetry and Remote Sensing, July 2025, 87–94 Cited on page 15.

[Get12] GETREUER, PASCAL. "**'Chan-Vese Segmentation'**". *Image Processing On Line* 2 (Aug. 2012), 214–224 Cited on page 107.

[GGC*20] GONZÁLEZ-GARCÍA, JOSUÉ, GÓMEZ-ESPINOSA, ALFONSO, CUAN-URQUIZO, ENRIQUE, et al. **Autonomous underwater vehicles: Localization, navigation, and communication for collaborative missions**. Feb. 2020 Cited on page 13.

[GGG*13] GÉNEVAUX, JEAN-DAVID, GALIN, ÉRIC, GUÉRIN, ERIC, et al. "**'Terrain generation using procedural models based on hydrology'**". *ACM Transactions on Graphics* 32 (4 July 2013), 1–13. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/2461912.2461996 Cited on pages 96, 134, 140.

[GGG*16] GUÉRIN, ERIC, GALIN, ERIC, GROSBELLET, FRANÇOIS, et al. "**'Efficient modeling of entangled details for natural scenes'**". *Computer Graphics Forum* 35 (7 Oct. 2016), 257–267. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13023 Cited on page 96.

[GGP*15] GÉNEVAUX, JEAN-DAVID, GALIN, ÉRIC, PEYTAVIE, ADRIEN, et al. "**'Terrain Modelling from Feature Primitives'**". (2015) Cited on pages 25, 48, 96.

[GGP*19] GALIN, ERIC, GUÉRIN, ERIC, PEYTAVIE, ADRIEN, et al. "**'A Review of Digital Terrain Modeling'**". *Computer Graphics Forum* 38 (2 May 2019), 553–577. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13657 Cited on pages 15, 24, 27, 28, 43, 129, 132, 133.

[GH91] GALYEAN, TINSLEY A. and HUGHES, JOHN F. "**'Sculpting: An Interactive Volumetric Modeling Technique'**". *ACM SIGGRAPH Computer Graphics* 25 (4 July 1991), 267–274. ISSN: 0097-8930. URL: https://dl.acm.org/doi/10.1145/127719.122747 Cited on page 28.

[GJ17] GOES, FERNANDO DE and JAMES, DOUG L. "**'Regularized Kelvinlets: Sculpting Brushes based on Fundamental Solutions of Elasticity'**". *ACM Transactions on Graphics* 36 (4 2017), 401–411. ISSN: 15577368 Cited on pages 102, 114, 144.

[GM05] GAIN, J. and MARAIS, P. "**'Warp sculpting'**". *IEEE Transactions on Visualization and Computer Graphics* 11 (2 Mar. 2005), 217–227. ISSN: 1077-2626. URL: http://ieeexplore.ieee.org/document/1388232/ Cited on page 28.

[GMS09] GAIN, JAMES, MARAIS, PATRICK and STRASSER, WOLFGANG. "**'Terrain sketching'**". *Proceedings of I3D 2009: The 2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* 1 (212 2009), 31–38 Cited on page 46.

[Gol16] GOLDBERG, WALTER M. "**'Atolls of the world: Revisiting the original checklist'**". *Atoll Research Bulletin* 2016 (610 2016), 1–47. ISSN: 00775630 Cited on page 33.

[GPG*16] GROSBELLET, FRANÇOIS, PEYTAVIE, ADRIEN, GUÉRIN, ÉRIC, et al. "**'Environmental Objects for Authoring Procedural Scenes'**". *Computer Graphics Forum* 35 (1 Feb. 2016), 296–308. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12726 Cited on pages 28, 94, 95, 96.

[GPM*14] GOODFELLOW, IAN, POUGET-ABADIE, JEAN, MIRZA, MEHDI, et al. "**'Generative adversarial networks'**". *Advances in Neural Information Processing Systems* 3 (11 June 2014), 139–144. URL: https://dl.acm.org/doi/10.1145/3422622 Cited on page 50.

[GPM*22] GUÉRIN, ERIC, PEYTAVIE, ADRIEN, MASNOU, SIMON, et al. "**'Gradient Terrain Authoring'**". *Computer Graphics Forum* 41 (2 May 2022), 85–95. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14460 Cited on pages 48, 152.

[GR05] GRIMM, VOLKER and RAILSBACK, STEVEN F. **Individual-based Modeling and Ecology**. Princeton University Press, 2005, 448 Cited on pages 85, 88.

[GSP*10] GEERTSEMA, MARTEEN, SCHWAB, JAMES W., PETER, JORDAN, et al. "**'Hillslope processes'**". *Compendium of forest hydrology and geomorphology in British Columbia*. Jan. 2010 Cited on page 133.

[GŞS*16] GALILI, E., ŞEVKETOĞLU, M., SALAMON, A., et al. "**'Late quaternary beach deposits and archaeological relicts on the coasts of Cyprus, and the possible implications of sea-level changes and tectonics on the early populations'**". *Geological Society Special Publication* 411 (1 2016), 179–218. ISSN: 03058719 Cited on page 167.

[GSW06] GÖLTENBOTH, FRIEDHELM, SCHOPPE, SABINE and WIDMANN, PETER. "**'CORAL REEFS'**". *Ecology of Insular Southeast Asia: THE INDONESIAN ARCHIPELAGO*. 2006, 47–69 Cited on page 29.

[HACV21] HAMONIC, FRANÇOIS, ALBERT, CÉCILE, COUËTOUX, BASILE and VAXÈS, YANN. "**'Optimizing the ecological connectivity of landscapes with generalized flow models and preprocessing'**". (Sept. 2021). URL: http://arxiv.org/abs/2109.06622 Cited on page 86.

[Har62] HARLOW, FRANCIS. **The particle-in-cell method for numerical solution of problems in fluid dynamics**. Tech. rep. Los Alamos National Laboratory (LANL), Mar. 1962. URL: https://www.osti.gov/servlets/purl/4769185/ Cited on page 142.

[Haw14] HAWICK, KEN A. "**'Modelling flood incursion and coastal erosion using cellular automata simulations'**". *Proceedings of the IASTED International Conference on Environmental Management and Engineering, EME 2014* (2014), 158–165 Cited on page 135.

[HD01] HARMON, R. S. (RUSSELL S.) and DOE, WILLIAM W. **Landscape erosion and evolution modeling**. Kluwer Academic/Plenum Publishers, 2001, 540. ISBN: 0306467186 Cited on page 131.

[HĎ11] HUDÁK, M. and ĎURIKOVIČ, R. "**'Terrain models for mass movement erosion'**". *Theory and Practice of Computer Graphics 2011, TPCG 2011 - Eurographics UK Chapter Proceedings* (2011), 9–16. URL: https://diglib.eg.org/bitstream/handle/10.2312/LocalChapterEvents.TPCG.TPCG11.009-016/009-016.pdf?sequence=1 Cited on pages 132, 140.

[Her22] HEREAU, ADRIEN. "**'Multi-level fault tolerance for autonomous robots-Application to an underwater robot'**". PhD thesis. Université de Montpellier, June 2022 Cited on page 14.

[HFTL22] HUANG, SHUDONG, FENG, WENTAO, TANG, CHENWEI and LV, JIANCHENG. "**'Partial Differential Equations Meet Deep Neural Networks: A Survey'**". (Oct. 2022). ISSN: 21622388. URL: http://arxiv.org/abs/2211.05567 Cited on page 143.

[HGA*10] HNAIDI, HOUSSAM, GUÉRIN, ERIC, AKKOUCHE, SAMIR, et al. "**'Feature based terrain generation using diffusion equation'**". *Computer Graphics Forum* 29 (7 Sept. 2010), 2179–2186. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2010.01806.x Cited on pages 46, 134.

[HGG*20] HEREAU, ADRIEN, GODARY-DEJEAN, KAREN, GUIOCHET, JÉRÉMIE, et al. "**'Testing an Underwater Robot Executing Transect Missions in Mayotte'**". *21st Annual Conference Towards Autonomous Robotic Systems*. Sept. 2020. URL: http://syera.fr Cited on page 13.

[HJA20] HO, JONATHAN, JAIN, AJAY and ABBEEL, PIETER. "**'Denoising Diffusion Probabilistic Models'**". *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851. URL: https://github.com/hojonathanho/diffusion. Cited on page 52.

[HNHC14] HUANG, ZHI, NICHOL, SCOTT L., HARRIS, PETER T. and CALEY, M. JULIAN. "**'Classification of submarine canyons of the Australian continental margin'**". *Marine Geology* 357 (Nov. 2014), 362–383. ISSN: 00253227 Cited on page 83.

[Hon13] HONG, QINGQI. "**'A skeleton-based technique for modelling implicit surfaces'**". *Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013* 2 (Cisp 2013), 686–691 Cited on page 170.

[Hop14] HOPLEY, DAVID. **Encyclopedia of modern coral reefs : structure, form and process**. Springer, Credo Reference, 2014. ISBN: 9789048126385 Cited on pages 17, 32, 37.

[HPSD17] HOEGH-GULDBERG, OVE, POLOCZANSKA, ELVIRA S., SKIRVING, WILLIAM and DOVE, SOPHIE. **Coral reef ecosystems under climate change and ocean acidification**. May 2017 Cited on page 13.

[HRJ*23] HUANG, RONGJIE, REN, YI, JIANG, ZIYUE, et al. "**'FastDiff 2: Revisiting and Incorporating GANs and Diffusion Models in High-Fidelity Speech Synthesis'**". *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023, 6994–7009. URL: https://aclanthology.org/2023.findings-acl.437 Cited on page 52.

[Hus85] HUSTON, M. A. "**'Patterns of species diversity on coral reefs.'**" *Annual review of ecology and systematics. Vol. 16* (1985), 149–177. ISSN: 0066-4162 Cited on page 16.

[HW65] HARLOW, FRANCIS H. and WELCH, J. EDDIE. "**'Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface'**". *The Physics of Fluids* 8 (12 Dec. 1965), 2182–2189. ISSN: 0031-9171. URL: https://pubs.aip.org/pfl/article/8/12/2182/951500/Numerical-Calculation-of-Time-Dependent-Viscous Cited on page 141.

[IZZE17] ISOLA, PHILLIP, ZHU, JUN-YAN, ZHOU, TINGHUI and EFROS, ALEXEI A. "**'Image-to-Image Translation with Conditional Adversarial Networks'**". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, 5967–5976. ISBN: 978-1-5386-0457-1. URL: http://arxiv.org/abs/1611.07004%20http://ieeexplore.ieee.org/document/8100115/ Cited on pages 51, 70.

[JKA*19] JAISWAL, SURAJ, KORKEALAAKSO, PASI, AMAN, RAFAEL, et al. "**'Deformable Terrain Model for the Real-Time Multibody Simulation of a Tractor with a Hydraulically Driven Front-Loader'**". *IEEE Access* 7 (2019), 172694–172708. ISSN: 21693536 Cited on page 139.

[JSS*15] JIANG, CHENFANFU, SCHROEDER, CRAIG, SELLE, ANDREW, et al. "**'The affine particle-in-cell method'**". *ACM Transactions on Graphics* 34 (4 July 2015), 1–10. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/2766996 Cited on page 142.

[JT11] JÁKÓ, BALÁZS and TÓTH, BALÁZS. "**'Fast Hydraulic and Thermal Erosion on GPU'**". PhD thesis. 2011, 4 Cited on page 132.

[JW17] JONES, BRUCE D. and WILLIAMS, JOHN R. "**'Fast computation of accurate sphere-cube intersection volume'**". *Engineering Computations* 34 (4 June 2017), 1204–1216. ISSN: 0264-4401. URL: https://www.emerald.com/insight/content/doi/10.1108/EC-02-2016-0052/full/html Cited on page 151.

[KBKŠ09] KRIŠTOF, PETER, BENEŠ, BEDŘICH, KŘIVÁNEK, J. and ŠŤAVA, ONDŘEJ. "**'Hydraulic erosion using smoothed particle hydrodynamics'**". *Computer Graphics Forum* 28 (2 2009), 219–228. ISSN: 14678659 Cited on pages 129, 134, 137, 140, 146, 149, 159.

[KBST22] KOSCHIER, DAN, BENDER, JAN, SOLENTHALER, BARBARA and TESCHNER, MATTHIAS. "**'A Survey on SPH Methods in Computer Graphics'**". *Computer Graphics Forum* 41 (2 2022), 737–760. ISSN: 14678659 Cited on pages 142, 149.

[KGC*11] KINDLER, PASCAL, GODEFROID, FABIENNE, CURRAN, H. ALLEN, et al. **Modern and quaternary carbonate environments**. Tech. rep. Gerace Research Center, Feb. 2011 Cited on page 31.

[KHM*20] KRS, V., HÄDRICH, TORSTEN, MICHELS, D. L., et al. "**'Wind Erosion: Shape Modifications by Interactive Particle-based Erosion and Deposition'**". *19th ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2020, SCA 2020 - Posters*. The Eurographics Association, 2020, 9–11 Cited on page 137.

[Kir94] KIRK, JOHN T. O. **Light and photosynthesis in aquatic ecosystems**. Cambridge University Press, 1994. ISBN: 9780511623370 Cited on page 17.

[KLR*20] KOVACS, E, LYONS, M, ROE, BORREGO-ACEVEDO R, et al. **Reef Cover Classification Coral reef internal class descriptors for global habitat mapping**. Tech. rep. 2020 Cited on pages 29, 32.

[KMN88] KELLEY, ALEX D., MALIN, MICHAEL C. and NIELSON, GREGORY M. "**'Terrain simulation using a model of stream erosion'**". *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1988* M (July 1988), 263–268 Cited on pages 134, 140.

[KO96] KOSHIZUKA, S. and OKA, Y. "**'Moving-Particle Semi-Implicit Method for Fragmentation of Incompressible Fluid'**". *Nuclear Science and Engineering* 123 (3 July 1996), 421–434. ISSN: 0029-5639. URL: https://www.tandfonline.com/doi/full/10.13182/NSE96-A24205 Cited on page 142.

[Kre27] KREMPF, ARMAND. **La forme des récifs coralliens et le régime des vents alternants**. Tech. rep. Gouvernement général de l'Indochine, 1927. URL: https://aquadocs.org/bitstream/handle/1834/35861/memoire2.pdf?sequence=1 Cited on page 31.

[KSBZ14] KÖRNER, CHRISTIAN, SPEHN, EVA M, BUGMANN, HARALD and ZURICH, ETH. **Mountain Systems**. Tech. rep. 2014. URL: https://www.researchgate.net/publication/238663492 Cited on page 38.

[Kuc86] KUCHLER, D A. **GREAT BARRIER REEF MARINE PARK AUTHORITY TECHNICAL MEMORANDUM GBRMPA-TM-7 REEF COVER AND ZONATION CLASSIFICATION SYSTEM FOR USE WITH REMOTELY SENSED GREAT BARRIER REEF DATA**. Tech. rep. 1986 Cited on page 29.

[KW22] KINGMA, DIEDERIK P and WELLING, MAX. "**'Auto-Encoding Variational Bayes'**". (Dec. 2022). URL: http://arxiv.org/abs/1312.6114 Cited on page 52.

[KWT88] KASS, MICHAEL, WITKIN, ANDREW and TERZOPOULOS, DEMETRI. "**'Snakes: Active contour models'**". *International Journal of Computer Vision* 1 (4 Jan. 1988), 321–331. ISSN: 0920-5691. URL: http://link.springer.com/10.1007/BF00133570 Cited on pages 104, 105.

[Las23] LASTIC, MAUD. "**'Efficient visual simulation of volcanic phenomena'**". PhD thesis. École Polytechnique, July 2023. URL: https://theses.hal.science/tel-04530320v1 Cited on page 139.

[LB25] LIU, Y. and BENES, B. "**'Single-Shot Example Terrain Sketching by Graph Neural Networks'**". *Computer Graphics Forum* 44 (1 Feb. 2025). ISSN: 14678659 Cited on page 52.

[LBZ*24] LILKENDEY, JULIAN, BARRELET, CYRIL, ZHANG, JINGJING, et al. "**'Herbivorous fish feeding dynamics and energy expenditure on a coral reef: Insights from stereo-video and AI-driven 3D tracking'**". *Ecology and Evolution* 14 (3 Mar. 2024). ISSN: 20457758 Cited on page 13.

[LCY*19] LU, JIA MING, CHEN, XIAO SONG, YAN, XIAO, et al. "**'A Rigging-Skinning Scheme to Control Fluid Simulation'**". *Computer Graphics Forum* 38 (7 2019), 501–512. ISSN: 14678659. URL: https://cg.cs.tsinghua.edu.cn/papers/CGF-2019-fluid.pdf Cited on page 144.

[LD09] LENAERTS, TOON and DUTRÉ, PHILIP. "**'Mixing fluids and granular materials'**". *Computer Graphics Forum* 28 (2 2009), 213–218. ISSN: 14678659 Cited on page 131.

[Lev92] LEVIN, S. A. "**'The problem of pattern and scale in ecology'**". *Ecology* 73 (6 1992), 1943–1967. ISSN: 00129658 Cited on page 85.

[LF15] LOWE, RYAN J. and FALTER, JAMES L. "**'Oceanic forcing of coral reefs'**". *Annual Review of Marine Science* 7 (Jan. 2015), 43–66. ISSN: 19410611 Cited on page 16.

[LFMA09] LOWE, RYAN J., FALTER, JAMES L., MONISMITH, STEPHEN G. and ATKINSON, MARLIN J. "**'Wave-driven circulation of a coastal reef-lagoon system'**". *Journal of Physical Oceanography* 39 (4 2009), 873–893. ISSN: 00223670 Cited on page 17.

[LFW*24] LI, XINGYAO, FAN, HENGHUI, WANG, PENGWEI, et al. "**'Interactive effect of soil dispersity and rainfall intensity on splash erosion: Insights from laboratory tests'**". *Catena* 238 (Apr. 2024). ISSN: 03418162 Cited on page 133.

[LGP*23] LOCHNER, J., GAIN, J., PERCHE, S., et al. "**'Interactive Authoring of Terrain using Diffusion Models'**". *Computer Graphics Forum* 42 (7 Oct. 2023). ISSN: 14678659 Cited on page 52.

[Li21] LI, WANWAN. "**'Procedural Modeling of the Great Barrier Reef'**". *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13017 LNCS (2021), 381–391. ISSN: 16113349 Cited on pages 44, 45.

[LJGS23] LEMIERE, LAËTITIA, JAEGER, MARC, GOSME, MARIE and SUBSOL, GÉRARD. "**'Combinatorial Maps, a New Framework to Model Agroforestry Systems'**". *Plant Phenomics* 5 (2023), 0120. ISSN: 26436515. URL: https://linkinghub.elsevier.com/retrieve/pii/S2643651524001158 Cited on pages 85, 86, 87.

[LK10] LAINE, SAMULI and KARRAS, TERO. "**'Efficient sparse voxel octrees'**". *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, Feb. 2010, 55–63. ISBN: 9781605589398. URL: https://dl.acm.org/doi/10.1145/1730804.1730814 Cited on page 26.

[LKM*23] LI, BOSHENG, KLEIN, JONATHAN, MICHELS, DOMINIK L., et al. "**'Rhizomorph: The Coordinated Function of Shoots and Roots'**". *ACM Transactions on Graphics* 42 (4 Aug. 2023). ISSN: 15577368 Cited on page 93.

[LMK*24] LYONS, MITCHELL B., MURRAY, NICHOLAS J., KENNEDY, EMMA V., et al. "**'New global area estimates for coral reefs from high-resolution mapping'**". *Cell Reports Sustainability* 1 (2 Feb. 2024), 100015. ISSN: 29497906 Cited on page 38.

[LP02] LANE, BRENDAN and PRUSINKIEWICZ, PRZEMYSLAW. "**'Generating Spatial Distributions for Multilevel Models of Plant Communities'**". *Proceedings of the Graphics Interface 2002 Conference*. May 2002, 69–80 Cited on page 92.

[LYZ*22] LI, HAI, YANG, XINGRUI, ZHAI, HONGJIA, et al. "**'Vox-Surf: Voxel-Based Implicit Surface Representation'**". *IEEE Transactions on Visualization and Computer Graphics* 30 (3 2022), 1743–1755. ISSN: 19410506 Cited on page 24.

[Man83] MANDELBROT, BENOIT B. **The fractal geometry of nature**. Ed. by BOOKS, TIME. 1st ed. W. H. Freeman and Company, 1983, 468. ISBN: 9780716711865 Cited on page 22.

[MANS13] MUKO, S., ARAKAKI, S., NAGAO, M. and SAKAI, KAZUHIKO. "**'Growth form-dependent response to physical disturbance and thermal stress in Acropora corals'**". *Coral Reefs* 32 (1 Mar. 2013), 269–280. ISSN: 07224028 Cited on page 30.

[MBK20] MASSELINK, GERD, BEETHAM, EDDIE and KENCH, PAUL. "**'Coral reef islands can accrete vertically in response to sea level rise'**". *Science Advances* 6 (24 2020). ISSN: 23752548 Cited on page 32.

[MCB*14] MARI, LORENZO, CASAGRANDI, RENATO, BERTUZZO, ENRICO, et al. "**'Metapopulation persistence and species spread in river networks'**". *Ecology Letters* 17 (4 2014), 426–434. ISSN: 14610248 Cited on page 87.

[MDH07] MEI, XING, DECAUDIN, PHILIPPE and HU, BAO GANG. "**'Fast hydraulic erosion simulation and visualization on GPU'**". *Proceedings - Pacific Conference on Computer Graphics and Applications* (2007), 47–56. ISSN: 15504085. URL: https://xingmei.github.io/files/erosion.pdf Cited on pages 44, 134, 168.

[MH99] MUMBY, PETER J. and HARBORNE, ALASTAIR R. "**'Development of a systematic classification scheme of marine habitats to facilitate regional management and mapping of Caribbean coral reefs'**". *Biological Conservation* 88 (2 May 1999), 155–163. ISSN: 00063207. URL: https://linkinghub.elsevier.com/retrieve/pii/S0006320798001086 Cited on page 29.

[MHS*19] MAKOWSKI, MIŁOSZ, HÄDRICH, TORSTEN, SCHEFFCZYK, JAN, et al. "**'Synthetic silviculture: Multi-scale modeling of plant ecosystems'**". *ACM Transactions on Graphics* 38 (4 July 2019). ISSN: 15577368 Cited on page 91.

[Mil86] MILLER, GAVIN S P. "**'The definition and rendering of terrain maps'**". *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. ACM, Aug. 1986, 39–48. ISBN: 0897911962. URL: https://dl.acm.org/doi/10.1145/15922.15890 Cited on page 15.

[MJD01] MCCOOK, L. J., JOMPA, J. and DIAZ-PULIDO, G. **Competition between corals and algae on coral reefs: A review of evidence and mechanisms**. May 2001 Cited on page 31.

[MKL20] MENSHUTINA, NATALIA V, KOLNOOCHENKO, ANDREY V and LEBEDEV, EVGENIY A. "**'Cellular Automata in Chemistry and Chemical Engineering'**". (2020). URL: https://doi.org/10.1146/annurev-chembioeng- Cited on page 135.

[MKM89] MUSGRAVE, FOREST KENTON, KOLB, CRAIG E. and MACE, ROBERT S. "**'The synthesis and rendering of eroded fractal terrains'**". *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989* (1989), 41–50. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.8939&rep=rep1&type=pdf Cited on pages 15, 42, 44, 129, 132, 140.

[MLD*21] MASLIN, MATHILDE, LOUIS, SILVAIN, DEJEAN, KAREN GODARY, et al. "**'Underwater robots provide similar fish biodiversity assessments as divers on coral reefs'**". *Remote Sensing in Ecology and Conservation* 7 (4 Dec. 2021), 567–578. ISSN: 20563485 Cited on page 13.

[MO14] MIRZA, MEHDI and OSINDERO, SIMON. "**'Conditional Generative Adversarial Nets'**". (Nov. 2014). URL: http://arxiv.org/abs/1411.1784 Cited on page 51.

[Mon05a] MONAGHAN, J. J. **Smoothed particle hydrodynamics**. Aug. 2005 Cited on page 142.

[Mon05b] MONTAGGIONI, LUCIEN F. "**'History of Indo-Pacific coral reef systems since the last glaciation: Development patterns and controlling factors'**". *Earth-Science Reviews* 71 (1-2 2005), 1–75. ISSN: 00128252 Cited on page 17.

[MRYR18] MODASSHIR, MD, RAHMAN, SHARMIN, YOUNGQUIST, OSCAR and REKLEITIS, IOANNIS. "**'Coral Identification and Counting with an Autonomous Underwater Vehicle'**". *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*. Institute of Electrical and Electronics Engineers Inc., July 2018, 524–529. ISBN: 9781728103761 Cited on page 17.

[MSMM11] MCLANE, ADAM J., SEMENIUK, CHRISTINA, MCDERMID, GREGORY J. and MARCEAU, DANIELLE J. **The role of agent-based models in wildlife ecology and management**. Apr. 2011 Cited on page 87.

[MST*20] MILDENHALL, BEN, SRINIVASAN, PRATUL P., TANCIK, MATTHEW, et al. "**'NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis'**". (Mar. 2020). URL: http://arxiv.org/abs/2003.08934 Cited on page 174.

[MU08] MINOR, EMILY S. and URBAN, DEAN L. "**'A graph-theory framework for evaluating landscape connectivity and conservation planning'**". *Conservation Biology* 22 (2 Apr. 2008), 297–307. ISSN: 08888892 Cited on page 86.

[Mur80] MURRAY, JOHN. "**'On the Structure and Origin of Coral Reefs and Islands'**". *Proceedings of the Royal Society of Edinburgh* 10 (1880), 505–518. ISSN: 0370-1646 Cited on page 39.

[Mus13] MUSETH, KEN. "**'VDB: High-resolution sparse volumes with dynamic topology'**". *ACM Transactions on Graphics* 32 (3 June 2013). ISSN: 07300301 Cited on pages 26, 27.

[MYG*14] MCCAULEY, DOUGLAS J., YOUNG, HILLARY S., GUEVARA, ROGER, et al. "**'Positive and Negative Effects of a Threatened Parrotfish on Reef Ecosystems'**". *Conservation Biology* 28 (5 Oct. 2014), 1312–1321. ISSN: 15231739 Cited on page 31.

[ND21] NICHOL, ALEX and DHARIWAL, PRAFULLA. **Improved Denoising Diffusion Probabilistic Models**. Tech. rep. 2021. URL: https://github.com/openai/ Cited on page 52.

[NJSR22] NAIK, SHANTHIKA, JAIN, ARYAMAAN, SHARMA, AVINASH and RAJAN, K. S. "**'Deep Generative Framework for Interactive 3D Terrain Authoring and Manipulation'**". *International Geoscience and Remote Sensing Symposium (IGARSS)*. Vol. 2022-July. Institute of Electrical and Electronics Engineers Inc., 2022, 6410–6413. ISBN: 9781665427920 Cited on page 52.

[NMG*20] NOCERINO, ERICA, MENNA, FABIO, GRUEN, ARMIN, et al. "**'Coral reef monitoring by scuba divers using underwater photogrammetry and geodetic surveying'**". *Remote Sensing* 12 (18 Sept. 2020). ISSN: 20724292 Cited on page 13.

[NR12] NELSON, TRISALYN A. and ROBERTSON, COLIN. "**'Refining spatial neighbourhoods to capture terrain effects'**". *Ecological Processes* 1 (1 2012), 1–11. ISSN: 21921709 Cited on page 85.

[NVP12] NATALI, MATTIA, VIOLA, IVAN and PATEL, DANIEL. "**'Rapid visualization of geological concepts'**". *Brazilian Symposium of Computer Graphic and Image Processing* (2012), 150–157. ISSN: 15301834 Cited on page 49.

[NWB*08] NICHOLLS, ROBERT J., WONG, POH POH, BURKETT, VIRGINIA, et al. "**'Climate change and coastal vulnerability assessment: Scenarios for integrated assessment'**". *Sustainability Science*. Vol. 3. Apr. 2008, 89–102 Cited on page 32.

[NWD05] NEIDHOLD, B., WACKER, M. and DEUSSEN, OLIVER. "**'Interactive physically based fluid and erosion simulation'**". *Natural Phenomena* (2005), 25–32. ISSN: 18160867. URL: http://graphics.uni-konstanz.de/publikationen/Neidhold2005InteractivePhysicallyBased/Neidhold2005InteractivePhysicallyBased.pdf Cited on pages 44, 129, 134.

[OAL*17] OGDEN, FRED L., ALLEN, MYRON B., LAI, WENCONG, et al. "**'The soil moisture velocity equation'**". *Journal of Advances in Modeling Earth Systems* 9 (2 June 2017), 1473–1487. ISSN: 19422466 Cited on page 97.

[OATS23] ORON, SHAI, AKKAYNAK, DERYA, TCHERNOV, BEVERLY N. GOODMAN and SHAKED, YONATHAN. "**'How monster storms shape fringing reefs: Observations from the 2020 Middle East Cyclone'**". *Ecosphere* 14 (7 July 2023). ISSN: 21508925 Cited on page 120.

[OCTR25] OH, DAPHNE, CRESSWELL, ANNA K., THOMSON, DAMIAN P. and RENTON, MICHAEL. "**'Do greater coral cover and morphological diversity increase habitat complexity?'**": *Coral Reefs* 44 (1 Feb. 2025), 257–272. ISSN: 14320975 Cited on pages 30, 31.

[OH95] O'BRIEN, JAMES F. and HODGINS, JESSICA K. "**'Dynamic simulation of splashing fluids'**". *Proceedings Computer Animation, CA 1995* (1995), 198–205. URL: https://arxiv.org/pdf/2302.06087.pdf Cited on page 168.

[OL01] OKUBO, AKIRA and LEVIN, SIMON A. **Diffusion and Ecological Problems: Modern Perspectives**. Vol. 14. Springer New York, 2001. ISBN: 978-1-4419-3151-1. URL: http://link.springer.com/10.1007/978-1-4757-4978-6 Cited on pages 96, 97.

[Ols04] OLSEN, JACOB. "**'Realtime procedural terrain generation'**". *Department of Mathematics And Computer Science* (2004), 20. URL: https://pdfs.semanticscholar.org/5961/c577478f21707dad53905362e0ec4e6ec644.pdf%5Cnhttp://www.tsi.enst.fr/~bloch/P6/PRREC/terrain_generation.pdf%5Cnhttp://web.mit.edu/cesium/Public/terrain.pdf Cited on pages 29, 42, 129, 132.

[ON00] ONOUE, K. and NISHITA, T. "**'A method for modeling and rendering dunes with wind-ripples'**". *Proceedings - Pacific Conference on Computer Graphics and Applications* 2000-Janua (2000), 427–428. ISSN: 15504085 Cited on page 169.

[OSSJ09] OLSEN, LUKE, SAMAVATI, FARAMARZ F., SOUSA, MARIO COSTA and JORGE, JOAQUIM A. "**'Sketch-based modeling: A survey'**". *Computers and Graphics (Pergamon)* 33 (1 2009), 85–103. ISSN: 00978493 Cited on page 45.

[ÖT18] ÖNGÜN, CIHAN and TEMIZEL, ALPTEKIN. "**'Paired 3D Model Generation with Conditional Generative Adversarial Networks'**". 11129 (Aug. 2018). Ed. by LEAL-TAIXÉ, LAURA and ROTH, STEFAN. URL: http://arxiv.org/abs/1808.03082 Cited on page 174.

[Par19] PARNA, PEETER. "**'Shallow Water Equations in Real-Time Computer Graphics'**". PhD thesis. Abertay University, Nov. 2019 Cited on page 141.

[Par23] PARIS, AXEL. "**'Modeling and simulating virtual terrains'**". PhD thesis. Université Claude Bernard Lyon 1, Mar. 2023. URL: https://theses.hal.science/tel-04502530 Cited on pages 17, 134.

[PC20] PANAGIOTOU, EMMANOUIL and CHAROU, ELENI. "**'Procedural 3D Terrain Generation using Generative Adversarial Networks'**". (Oct. 2020). URL: http://arxiv.org/abs/2010.06411 Cited on page 51.

[PCBK11] PLAISANCE, LAETITIA, CALEY, M. JULIAN, BRAINARD, RUSSELL E. and KNOWLTON, NANCY. "**'The diversity of coral reefs: What are we missing?'**": *PLoS ONE* 6 (10 Oct. 2011). ISSN: 19326203 Cited on page 13.

[PD09] PEREZ, LILIANA and DRAGICEVIC, SUZANA. "**'An agent-based approach for modeling dynamics of contagious disease spread'**". *International Journal of Health Geographics* 8 (1 Aug. 2009). ISSN: 1476072X Cited on page 88.

[PDG*19] PEYTAVIE, ADRIEN, DUPONT, THIBAULT, GUÉRIN, ÉRIC, et al. "**'Procedural Riverscapes'**". *Computer Graphics Forum* 38 (7 Oct. 2019), 35–46. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13814 Cited on pages 113, 134.

[Per01] PERLIN, KEN. "**'Noise Hardware'**". *Real-Time Shading SIGGRAPH Course Notes* (2001) Cited on page 42.

[Per02] PERLIN, KEN. "**'Improving noise'**". *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM, July 2002, 681–682. ISBN: 1581135211. URL: https://dl.acm.org/doi/10.1145/566570.566636 Cited on pages 63, 205.

[Per85] PERLIN, KEN. "**'An image synthesizer'**". *ACM SIGGRAPH Computer Graphics* 19 (3 July 1985), 287–296. ISSN: 0097-8930. URL: https://dl.acm.org/doi/10.1145/325165.325247 Cited on pages 22, 28, 42, 94.

[PEV20] PENDLETON, LINWOOD, EVANS, KAREN and VISBECK, MARTIN. **We need a global movement to transform ocean science for a better world**. May 2020 Cited on page 13.

[PGG*24] PEYTAVIE, ADRIEN, GAIN, JAMES, GUÉRIN, ERIC, et al. "**'DeadWood: Including Disturbance and Decay in the Depiction of Digital Nature'**". *ACM Transactions on Graphics* 43 (2 Apr. 2024), 1–19. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/3641816 Cited on pages 87, 88.

[PGGM09a] PEYTAVIE, ADRIEN, GALIN, E., GROSJEAN, J. and MERILLOU, S. "**'Arches: A framework for modeling complex terrains'**". *Computer Graphics Forum* 28 (2 2009), 457–467. ISSN: 14678659 Cited on pages 27, 151.

[PGGM09b] PEYTAVIE, ADRIEN, GALIN, ERIC, GROSJEAN, J. and MERILLOU, S. "**'Procedural Generation of Rock Piles using Aperiodic Tiling'**". *Computer Graphics Forum* 28 (7 Oct. 2009), 1801–1809. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01557.x Cited on page 132.

[PGP*18] PARIS, AXEL, GALIN, ÉRIC, PEYTAVIE, ADRIEN, et al. "**'Amplification de Terrains avec des caractéristiques implicites 3D'**". *Journées de l'Association Française d'Informatique Graphique* (2018) Cited on page 135.

[PGP*19a] PARIS, AXEL, GALIN, ERIC, PEYTAVIE, ADRIEN, et al. "**'Terrain Amplification with Implicit 3D Features'**". *ACM Transactions on Graphics* 38 (5 Oct. 2019), 1–15. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/3342765 Cited on pages 26, 137, 140, 166, 167.

[PGP*19b] PURKIS, SAM J., GLEASON, ARTHUR C.R., PURKIS, CHARLOTTE R., et al. "**'High-resolution habitat and bathymetry maps for 65,000 sq. km of Earth's remotest coral reefs'**". *Coral Reefs* 38 (3 June 2019), 467–488. ISSN: 14320975 Cited on page 29.

[PGP*21] PARIS, AXEL, GUÉRIN, ÉRIC, PEYTAVIE, ADRIEN, et al. "**'Synthesizing Geologically Coherent Cave Networks'**". *Computer Graphics Forum* 40 (7 Oct. 2021), 277–287. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14420 Cited on page 161.

[PHT*13] PAN, ZHERONG, HUANG, JIN, TONG, YIYING, et al. "**'Interactive localized liquid motion editing'**". *ACM Transactions on Graphics* 32 (6 Nov. 2013). ISSN: 07300301 Cited on page 144.

[PHTB12] PAN, ZHERONG, HUANGY, JIN, TONG, YIYING and BAO, HUJUN. "**'Wake synthesis for shallow water equation'**". *Eurographics Symposium on Geometry Processing*. Vol. 31. Eurographics Association, 2012, 2029–2036 Cited on page 142.

[PL92] PRUSINKIEWICZ, PRZEMYSLAW and LINDENMAYER, ARISTID. "**'The Algorithmic Beauty of Plants'**". *SIAM Review* 34 (1 1992), 142–143. ISSN: 0036-1445. URL: http://algorithmicbotany.org/papers/abop/abop.pdf Cited on page 18.

[PLWZ19] PARK, TAESUNG, LIU, MING-YU, WANG, TING-CHUN and ZHU, JUN-YAN. "**'GauGAN'**". *ACM SIGGRAPH 2019 Real-Time Live!* Vol. 405. ACM, July 2019, 1–1. ISBN: 9781450363150. URL: https://dl.acm.org/doi/10.1145/3306305.3332370 Cited on page 78.

[PM90] PERONA, P. and MALIK, J. "**'Scale-space and edge detection using anisotropic diffusion'**". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (7 July 1990), 629–639. ISSN: 01628828. URL: http://ieeexplore.ieee.org/document/56205/ Cited on page 97.

[PMG*22] PAŁUBICKI, WOJTEK, MAKOWSKI, MIŁOSZ, GAJDA, WERONIKA, et al. "**'Ecoclimates: Climate-Response Modeling of Vegetation'**". *ACM Transactions on Graphics* 41 (4 July 2022). ISSN: 15577368 Cited on pages 96, 134.

[PMK*13] PERRY, CHRIS T., MURPHY, GARY N., KENCH, PAUL S., et al. "**'Caribbean-wide decline in carbonate production threatens coral reef growth'**". *Nature Communications* 4 (2013). ISSN: 20411723 Cited on page 16.

[PPB*23a] PERCHE, SIMON, PEYTAVIE, ADRIEN, BENES, BEDRICH, et al. "**'Authoring Terrains with Spatialised Style'**". *Computer Graphics Forum* 42 (7 Oct. 2023). ISSN: 14678659 Cited on page 78.

[PPB*23b] PERCHE, SIMON, PEYTAVIE, ADRIEN, BENES, BEDRICH, et al. "**'StyleDEM: a Versatile Model for Authoring Terrains'**". (Apr. 2023). URL: http://arxiv.org/abs/2304.09626 Cited on page 78.

[PPG*19] PARIS, AXEL, PEYTAVIE, ADRIEN, GUÉRIN, ÉRIC, et al. "**'Desertscape Simulation'**". *Computer Graphics Forum* 38 (7 Oct. 2019), 47–55. ISSN: 0167-7055. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13815 Cited on pages 28, 112, 136, 137, 140, 162, 165, 169.

[PPG*20] PARIS, AXEL, PEYTAVIE, ADRIEN, GUÉRIN, ERIC, et al. "**'Modeling Rocky Scenery using Implicit Blocks'**". *The Visual Computer* 36 (10 2020), 2251–2261. URL: https://hal.science/hal-02926218v2 Cited on page 132.

[PSK*12] PIRK, SÖREN, STAVA, ONDREJ, KRATT, JULIAN, et al. "**'Plastic trees: Interactive self-adapting botanical tree models'**". *ACM Transactions on Graphics* 31 (4 July 2012). ISSN: 07300301 Cited on page 93.

[PSP*24] PETERSON, EMILY A., STUART, COURTNEY E., PITTMAN, SIMON J., et al. "**'Graph-theoretic modeling reveals connectivity hotspots for herbivorous reef fishes in a restored tropical island system'**". *Landscape Ecology* 39 (8 Aug. 2024). ISSN: 15729761 Cited on page 86.

[PT05] PATEL, MAYUR and TAYLOR, NOAH. "**'Simple Divergence-Free Fields for Artistic Simulation'**". *Journal of Graphics Tools* 10 (4 2005), 49–60. ISSN: 1086-7651. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=18183ace6beab86bb0d9a505bfeb6ab7bb3b840b Cited on page 144.

[PWA*17] PRATT, MARTIN J., WYSESSION, MICHAEL E., ALEQABI, GHASSAN, et al. "**'Shear velocity structure of the crust and upper mantle of Madagascar derived from surface wave tomography'**". *Earth and Planetary Science Letters* 458 (Jan. 2017), 405–417. ISSN: 0012821X Cited on page 83.

[PWL*20] PETERS, RONNY, WALTHER, MARC, LOVELOCK, CATHERINE, et al. **The interplay between vegetation and water in mangroves: new perspectives for mangrove stand modelling and ecological research**. Aug. 2020 Cited on pages 89, 90.

[Qui13] QUILEZ, INIGO. **Smooth minimum**. 2013. URL: https://iquilezles.org/articles/smin/ Cited on pages 63, 202.

[Ram24] RAMOS, JUAN I. "**'Effects of Anisotropy, Convection, and Relaxation on Nonlinear Reaction-Diffusion Systems'**". *Computation* 12 (11 Nov. 2024). ISSN: 20793197 Cited on page 97.

[RB04] ROA, TONEY and BENES, BEDRICH. "**'Simulating desert scenery'**". *Winter School of Computer Graphics SHORT communication Papers Proceedings* (2004), 17–22. URL: https://dspace5.zcu.cz/bitstream/11025/6180/1/C79.pdf Cited on pages 137, 169.

[RBL*21] ROMBACH, ROBIN, BLATTMANN, ANDREAS, LORENZ, DOMINIK, et al. "**'High-Resolution Image Synthesis with Latent Diffusion Models'**". (Dec. 2021). URL: http://arxiv.org/abs/2112.10752 Cited on page 78.

[RDBC24] ROSSET, NICOLAS, DUVIGNEAU, REGIS, BOUSSEAU, ADRIEN and CORDONNIER, GUILLAUME. "**'Windblown sand around obstacles – simulation and validation of deposition patterns'**". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 7 (1 May 2024). ISSN: 25776193 Cited on page 137.

[RGF00] RIGAUDIÈRE, DOMINIQUE, GESQUIÈRE, GILLES and FAUDOT, DOMINIQUE. "**'Shape Modelling with Skeleton based Implicit Primitives'**". *Methods* (2000). URL: https://www.graphicon.ru/html/2000/2D%20GRAPHICS/Rigaudiere.pdf Cited on page 170.

[RHRH22] RUDIN, NIKITA, HOELLER, DAVID, REIST, PHILIPP and HUTTER, MARCO. "**'Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning'**". (Aug. 2022). URL: http://arxiv.org/abs/2109.11978 Cited on page 15.

[RKH*21] RADFORD, ALEC, KIM, JONG WOOK, HALLACY, CHRIS, et al. "**'Learning Transferable Visual Models From Natural Language Supervision'**". (Feb. 2021). URL: http://arxiv.org/abs/2103.00020 Cited on page 78.

[RLL*10] REINHARD, E, LAGAE, A, LEFEBVRE, S, et al. "**'State of the Art in Procedural Noise Functions'**". *Eurographics 2010 - State of the Art Reports*. Ed. by HAUSER, HELWIG and REINHARD, ERIK. The Eurographics Association, 2010 Cited on page 42.

[RLL11] ROOSE, DIRK, LEUVEN, K U and LÓPEZ, YAIDEL REYES. **Dynamic refinement for fluid flow simulations with SPH Particle refinement for fluid flow simulations with SPH**. Tech. rep. 2011. URL: https://www.researchgate.net/publication/228531954 Cited on page 149.

[RPP93] ROUDIER, P., PEROCHE, B. and PERRIN, M. "**'Landscapes Synthesis Achieved through Erosion and Deposition Process Simulation'**". *Computer Graphics Forum* 12 (3 1993), 375–383. ISSN: 14678659 Cited on pages 129, 134.

[RQT*13] RAMALHO, RICARDO S., QUARTAU, RUI, TRENHAILE, ALAN S., et al. **Coastal evolution on volcanic oceanic islands: A complex interplay between volcanism, erosion, sedimentation, sea-level change and biogenic production**. Dec. 2013 Cited on pages 31, 138.

[RSW*25] REN, CHAO, SONG, CHANGJUN, WU, LIJIAN, et al. "**'Stability evaluation and reinforcement of ultra-high fill embankment slope of GRS'**". *Frontiers in Earth Science* 13 (2025). ISSN: 22966463 Cited on page 169.

[RTG60] RANZ, W. E., TALANDIS, G. R. and GUTTERMAN, BERNARD. "**'Mechanics of particle bounce'**". *AIChE Journal* 6 (1 Mar. 1960), 124–127. ISSN: 0001-1541. URL: https://aiche.onlinelibrary.wiley.com/doi/10.1002/aic.690060123 Cited on page 170.

[RTW*12] RAVEENDRAN, KARTHIK, THUEREY, NILS, WOJTAN, CHRIS, et al. **Controlling Liquids Using Meshes**. Tech. rep. 2012 Cited on page 144.

[RWTT14] RAVEENDRAN, KARTHIK, WOJTAN, CHRIS, THUEREY, NILS and TURK, GREG. "**'Blending liquids'**". *ACM Transactions on Graphics* 33 (4 July 2014), 1–10. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/2601097.2601126 Cited on page 144.

[RZ54] RICHARDSON, J. F. and ZAKI, W. N. "**'The sedimentation of a suspension of uniform spheres under conditions of viscous flow'**". *Chemical Engineering Science* 3 (1954) Cited on page 147.

[SABW82] SWOPE, WILLIAM C., ANDERSEN, HANS C., BERENS, PETER H. and WILSON, KENT R. "**'A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters'**". *The Journal of Chemical Physics* 76 (1 1982), 637–649. ISSN: 00219606 Cited on page 150.

[SAC*99] STORA, DAN, AGLIATI, PIERRE-OLIVIER, CANI, MARIE-PAULE, et al. "**'Animating Lava Flows'**". *Graphics Interface (GI'99) Proceedings*. 1999, 203–210. URL: https://inria.hal.science/inria-00510066v1 Cited on page 139.

[ŠBBK08] ŠT'AVA, ONDŘEJ, BENEŠ, BEDŘICH, BRISBIN, MATTHEW and KRIVÁNEK, JAROSLAV. "**'Interactive terrain modeling using hydraulic erosion'**". *Computer Animation 2008 - ACM SIGGRAPH / Eurographics Symposium, SCA 2008 - Proceedings* (2008), 200–210. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.5239&rep=rep1&type=pdf Cited on page 140.

[SBD*20] SOUS, DAMIEN, BOUCHETTE, FRÉDÉRIC, DOERFLINGER, ERIK, et al. "**'On the small-scale fractal geometrical structure of a living coral reef barrier'**". *Earth Surface Processes and Landforms* 45 (12 2020), 3042–3054. ISSN: 10969837 Cited on page 31.

[ŠBM*10] ŠT'AVA, ONDŘEJ, BENEŠ, BEDŘICH, MĚCH, RADOMIR, et al. "**'Inverse procedural modeling by automatic generation of L-systems'**". *Computer Graphics Forum* 29 (2 2010), 665–674. ISSN: 14678659. URL: https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Stava10.pdf Cited on page 174.

[SBM05] STANLEY, KENNETH O, BRYANT, BOBBY D and MIIKKULAINEN, RISTO. **Real-Time Neuroevolution in the NERO Video Game**. Tech. rep. 2005. URL: http://nerogame.org Cited on page 174.

[SBW*17] SPALDING, MARK, BURKE, LAURETTA, WOOD, SPENCER A., et al. "**'Mapping the global value and distribution of coral reef tourism'**". *Marine Policy* 82 (Aug. 2017), 104–113. ISSN: 0308597X Cited on page 13.

[SHM*13] SOH, YOUNGSUNG, HAE, YONGSUK, MEHMOOD, AAMER, et al. "**'Performance Evaluation of Various Functions for Kernel Density Estimation'**". *Open Journal of Applied Sciences* 03 (01 2013), 58–64. ISSN: 2165-3917. URL: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/ojapps.2013.31B012 Cited on page 207.

[Sil98] SILVERMAN, B.W. **Density Estimation for Statistics and Data Analysis**. Routledge, Feb. 1998. ISBN: 9781315140919. URL: https://www.taylorfrancis.com/books/9781351456173 Cited on page 206.

[Sim90] SIMS, KARL. "**'Particle animation and rendering using data parallel computation'**". *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. Vol. 24. ACM, Sept. 1990, 405–413. ISBN: 0897913442. URL: https://dl.acm.org/doi/10.1145/97879.97923 Cited on page 144.

[ŠKC*14] ŠT'AVA, ONDŘEJ, KRATT, JULIAN, CHEN, BAOQUAN, et al. "**'Inverse procedural modelling of trees'**". *Computer Graphics Forum* 33 (6 2014), 118–131. ISSN: 14678659. URL: https://cfcs.pku.edu.cn/baoquan/docs/20180621170343624947.pdf Cited on page 174.

[SKG*09] SMELIK, RUBEN M., KRAKER, KLAAS JAN DE, GROENEWEGEN, SASKIA A, et al. "**'A survey of procedural methods for terrain modelling'**". *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)* (2009) Cited on pages 28, 43, 44, 129.

[SKJY06] SANDERSON, ALLEN R., KIRBY, ROBERT M., JOHNSON, CHRIS R. and YANG, LINGFA. "**'Advanced Reaction-Diffusion Models for Texture Synthesis'**". *Journal of Graphics Tools* 11 (3 Jan. 2006), 47–71. ISSN: 1086-7651. URL: https://www.tandfonline.com/doi/full/10.1080/2151237X.2006.10129222 Cited on page 97.

[SM06] SUN, J. and MUHS, D. R. "**'Dune fields / Mid-latitudes'**". *Encyclopedia of Quaternary Science*. Elsevier Science Ltd., Jan. 2006, 607–626. ISBN: 9780444527479 Cited on page 136.

[SPF*23] SCHOTT, HUGO, PARIS, AXEL, FOURNIER, LUCIE, et al. "**'Large-scale Terrain Authoring through Interactive Erosion Simulation'**". *ACM Transactions on Graphics* 42 (5 Oct. 2023), 1–15. ISSN: 0730-0301. URL: https://dl.acm.org/doi/10.1145/3592787 Cited on pages 44, 134.

[SRA24] SOUSA, PAULO, RODRIGUES, CARLOS VEIGA and AFONSO, ALEXANDRE. "**'Enhancing CFD solver with Machine Learning techniques'**". *Computer Methods in Applied Mechanics and Engineering* 429 (Sept. 2024). ISSN: 00457825 Cited on page 144.

[SRH*22] SHADRICK, JENNIFER R., ROOD, DYLAN H., HURST, MARTIN D., et al. "**'Sea-level rise will likely accelerate rock coast cliff retreat rates'**". *Nature Communications* 13 (1 Dec. 2022). ISSN: 20411723 Cited on page 83.

[SRSS12] SEIDL, RUPERT, RAMMER, WERNER, SCHELLER, ROBERT M. and SPIES, THOMAS A. "**'An individual-based process model to simulate landscape-scale forest ecosystem dynamics'**". *Ecological Modelling* 231 (Apr. 2012), 87–100. ISSN: 03043800 Cited on pages 93, 94, 108.

[SS05] STACHNIAK, S and STUERZLINGER, W. "**'An Algorithm for Automated Fractal Terrain Deformation'**". *In Proceedings of Computer Graphics and Artificial Intelligence* (2005), 64–76. URL: www.cs.yorku.ca/~wolfgang/%5Cnhttp://alter-unilim.teiath.gr/3ia_previous_conferences_cds/2005/Papers/Papers/Paper03.pdf Cited on pages 28, 129.

[Sta03] STAM, JOS. "**'Real-Time Fluid Dynamics for Games'**". *Proceedings of the Game Developer Conference* 18 (11 2003), 17. ISSN: 09574174. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.6736&rep=rep1&type=pdf Cited on page 165.

[Sta99] STAM, JOS. "**'Stable fluids'**". *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999* (1999), 121–128 Cited on pages 97, 141, 143, 163.

[STBB14] SMELIK, RUBEN M., TUTENEL, TIM, BIDARRA, RAFAEL and BENES, BEDRICH. "**'A survey on procedural modelling for virtual worlds'**". *Computer Graphics Forum* 33 (6 2014), 31–50. ISSN: 14678659 Cited on pages 27, 28, 29.

[Sto69] STODDART, D. R. "**'Ecology and morphology of recent coral reefs'**". *Biological Reviews* 44 (4 Nov. 1969), 433–498. ISSN: 1464-7931 Cited on page 29.

[Stu11] STUDIOS, MOJANG. **Minecraft**. 2011 Cited on page 27.

[SW19] SPICK, RYAN and WALKER, JAMES. "**'Realistic and Textured Terrain Generation using GANs'**". *European Conference on Visual Media Production*. Vol. 2022-March. ACM, Dec. 2019, 1–10. ISBN: 9781450370035. URL: https://dl.acm.org/doi/10.1145/3359998.3369407 Cited on page 50.

[TACS21] TANG, JINGWEI, AZEVEDO, VINICIUS C., CORDONNIER, GUILLAUME and SOLENTHALER, BARBARA. "**'Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization'**". *Computer Graphics Forum* 40 (2 May 2021), 339–353. ISSN: 14678659 Cited on pages 143, 144.

[Tau95] TAUBIN, G. "**'Curve and surface smoothing without shrinkage'**". *Proceedings of IEEE International Conference on Computer Vision*. IEEE Comput. Soc. Press, 1995, 852–857. ISBN: 0-8186-7042-8. URL: http://ieeexplore.ieee.org/document/466848/ Cited on page 97.

[TB18] TALGORN, FRANÇOIS XAVIER and BELHADJ, FARÈS. "**'Real-time sketch-based terrain generation'**". *ACM International Conference Proceeding Series* (March 2021 2018), 13–18. URL: https://www.researchgate.net/profile/Fares-Belhadj/publication/325327322_Real-Time_Sketch-Based_Terrain_Generation/links/6054c4b4a6fdccbfeaf0a8b2/Real-Time-Sketch-Based-Terrain-Generation.pdf Cited on page 47.

[TCL*13] TOGELIUS, JULIAN, CHAMPANDARD, ALEX J, LANZI, PIER LUCA, et al. "**'Procedural Content Generation: Goals, Challenges and Actionable Steps'**". *Artificial and Computational Intelligence in Games* 6 (2013). URL: https://groups.google.com/group/proceduralcontent/ Cited on page 28.

[TEC*14] TASSE, FLORA PONJOU, EMILIEN, ARNAUD, CANI, MARIE-PAULE, et al. "**'First Person Sketch-based Terrain Editing'**". *Graphics Interface*. 2014. URL: https://hal.inria.fr/hal-00976689/file/FirstPersonSketchBasedTerrainEditing_GI2014.pdf Cited on page 46.

[TG13] TERRY, JAMES P and GOFF, JAMES. "**'One hundred and thirty years since Darwin: 'Reshaping' the theory of atoll formation'**". *The Holocene* 23 (4 Apr. 2013), 615–619. ISSN: 0959-6836. URL: http://journals.sagepub.com/doi/10.1177/0959683612463101 Cited on pages 37, 41.

[THR23] TZACHOR, ASAF, HENDEL, OFIR and RICHARDS, CATHERINE E. "**'Digital twins: a stepping stone to achieve ocean sustainability?'**": *npj Ocean Sustainability* 2 (1 Oct. 2023), 16. ISSN: 2731-426X. URL: https://www.nature.com/articles/s44183-023-00023-9 Cited on page 15.

[TJ10] TYCHONIEVICH, L. A. and JONES, M. D. "**'Delaunay deformable mesh for the weathering and erosion of 3D terrain'**". *Visual Computer* 26 (12 2010), 1485–1495. ISSN: 01782789 Cited on page 149.

[TMNM97] TOMASCIK, TOMAS, MAH, ANMARIE JANICE, NONTJI, ANUGERAH and MOOSA, MOHAMMAD KASIM. "**'Coral Reef Origins: The Theories'**". *The Ecology of the Indonesian Seas*. Oxford University PressOxford, July 1997, 207–232. URL: https://academic.oup.com/book/52768/chapter/421867308 Cited on page 41.

[TSSP17] TOMPSON, JONATHAN, SCHLACHTER, KRISTOFER, SPRECHMANN, PABLO and PERLIN, KEN. "**'Accelerating eulerian fluid simulation with convolutional networks'**". *34th International Conference on Machine Learning, ICML 2017* 7 (2017), 5258–5267 Cited on pages 144, 174.

[Tur52] TURING, M. "**'The chemical basis of morphogenesis'**". *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 237 (641 Aug. 1952), 37–72. ISSN: 2054-0280. URL: https://royalsocietypublishing.org/doi/10.1098/rstb.1952.0012 Cited on pages 84, 97.

[Tur91] TURK, GREG. "**'Generating textures on arbitrary surfaces using reaction-diffusion'**". *ACM SIGGRAPH Computer Graphics* 25 (4 July 1991), 289–298. ISSN: 0097-8930. URL: https://dl.acm.org/doi/10.1145/127719.122749 Cited on page 97.

[UCCH04] URIARTE, MARÍA, CONDIT, RICHARD, CANHAM, CHARLES D. and HUBBELL, STEPHEN P. "**'A spatially explicit model of sapling growth in a tropical forest: Does the identity of neighbours matter?'**": *Journal of Ecology* 92 (2 Apr. 2004), 348–360. ISSN: 00220477 Cited on page 89.

[UMTS09] URBAN, DEAN L., MINOR, EMILY S., TREML, ERIC A. and SCHICK, ROBERT S. **Graph models of habitat mosaics**. Mar. 2009 Cited on page 86.

[Ver44] VERHULST, P.-F. "**'Recherches mathématiques sur la loi d'accroissement de la population'**". *Nouveaux mémoires de l'Académie Royale des sciences et Belles-Lettres de Bruxelles* (1844) Cited on page 97.

[Ver67] VERLET, LOUP. "**'Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules'**". *Physical Review* 159 (1 July 1967), 98–103. ISSN: 0031-899X. URL: https://link.aps.org/doi/10.1103/PhysRev.159.98 Cited on page 150.

[VHLL05] VALETTE, GILLES, HERBIN, MICHEL, LUCAS, LAURENT and LÉONARD, JOËL. "**'A preliminary approach of 3D simulation of soil surface degradation by rainfall'**". *Natural Phenomena* (2005), 41–50. ISSN: 18160867. URL: https://diglib.eg.org/bitstream/handle/10.2312/NPH.NPH05.041-049/041-049.pdf?sequence=1 Cited on page 133.

[Vis18] VISBECK, MARTIN. **Ocean science research is key for a sustainable future**. Dec. 2018 Cited on page 13.

[VK16] VILA-CONCEJO, ANA and KENCH, PAUL. "**'Storms in Coral Reefs'**". *Coastal Storms: Processes and Impacts*. Wiley Blackwell, Aug. 2016, 127–149. ISBN: 9781118937099 Cited on page 120.

[VMC*20] VILLON, SÉBASTIEN, MOUILLOT, DAVID, CHAUMONT, MARC, et al. "**'A new method to control error rates in automated species identification with deep learning algorithms'**". *Scientific Reports* 10 (1 Dec. 2020). ISSN: 20452322 Cited on page 13.

[VMG17] VILLANUEVA, ALBERTO JASPE, MARTON, FABIO and GOBBETTI, ENRICO. "**'Symmetry-aware Sparse Voxel DAGs ( SSVDAGs ) for compression-domain tracing of high-resolution geometric scenes'**". *Journal of Computer Graphics Techniques* 6 (2 2017), 1–30 Cited on page 26.

[VMP21] VOULGARIS, GEORGIOS, MADEMLIS, IOANNIS and PITAS, IOANNIS. "**'Procedural Terrain Generation Using Generative Adversarial Networks'**". *European Signal Processing Conference*. Vol. 2021-August. European Signal Processing Conference, EUSIPCO, 2021, 686–690. ISBN: 9789082797060 Cited on page 52.

[Vre94] VREUGDENHIL, C. B. **Numerical Methods for Shallow-Water Flow**. Vol. 13. Springer Netherlands, 1994. ISBN: 978-90-481-4472-3. URL: http://link.springer.com/10.1007/978-94-015-8354-1 Cited on page 142.

[VSP*17] VASWANI, ASHISH, SHAZEER, NOAM, PARMAR, NIKI, et al. "**'Attention Is All You Need'**". (June 2017). URL: http://arxiv.org/abs/1706.03762 Cited on page 174.

[Wal93] WALTON, OTIS R. "**'Numerical simulation of inclined chute flows of monodisperse, inelastic, frictional spheres'**". *Mechanics of Materials* 16 (1-2 Aug. 1993), 239–247. ISSN: 01676636. URL: https://linkinghub.elsevier.com/retrieve/pii/016766369390048V Cited on page 131.

[WCMT07] WOJTAN, CHRIS, CARLSON, MARK, MUCHA, PETER J. and TURK, GREG. "**'Animating corrosion and erosion'**". *Natural Phenomena* (2007), 15–22. ISSN: 18160867. URL: https://pub.ist.ac.at/group_wojtan/projects/acid/acid_FINAL_nocolorplate.pdf Cited on pages 135, 140, 146, 147, 148, 149.

[Wei08] WEI, LI YI. "**'Parallel Poisson disk sampling'**". *ACM Transactions on Graphics* 27 (3 Aug. 2008). ISSN: 07300301 Cited on page 88.

[Wen95] WENDLAND, HOLGER. "**'Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree'**". *Advances in Computational Mathematics* 4 (1 Dec. 1995), 389–396. ISSN: 1019-7168. URL: http://link.springer.com/10.1007/BF02123482 Cited on page 206.

[Wes84] WESTOBY, MARK. "**'The Self-Thinning Rule'**". Academic Press, 1984, 167–225. ISBN: 0120139146. URL: https://linkinghub.elsevier.com/retrieve/pii/S0065250408601713 Cited on page 91.

[WH91] WEJCHERT, JAKUB and HAUMANN, DAVID. "**'Animation aerodynamics'**". *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991* 25 (4 1991), 19–22. URL: https://dl.acm.org/doi/pdf/10.1145/122718.122719 Cited on pages 102, 113, 114, 144.

[Wie89] WIENS, J A. **Spatial Scaling in Ecology**. Tech. rep. 1989, 385–397 Cited on page 85.

[WJ93] WAND, M P and JONES, M C. **Comparison of Smoothing Parameterizations in Bivariate Kernel Density Estimation**. Tech. rep. 1993, 37 Cited on page 206.

[WK91] WITKIN, ANDREW and KASS, MICHAEL. "**'Reaction-diffusion textures'**". *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. Vol. 25. ACM, July 1991, 299–308. ISBN: 0897914368. URL: https://dl.acm.org/doi/10.1145/122718.122750 Cited on page 97.

[WLZ*18] WANG, TING-CHUN, LIU, MING-YU, ZHU, JUN-YAN, et al. "**'High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs'**". (Aug. 2018). URL: http://arxiv.org/abs/1711.11585 Cited on page 52.

[Woo03] WOODROFFE, COLIN D. **Coasts: Form, process and evolution**. Press syndicate of the University of Cambridge, 2003. ISBN: 0521812542 Cited on pages 31, 139.

[WRMB18] WULFF-JENSEN, ANDREAS, RANT, NICLAS NERUP, MØLLER, TOBIAS NORDVIG and BILLESKOV, JONAS AKSEL. "**'Deep Convolutional Generative Adversarial Network for Procedural 3D Landscape Generation Based on DEM'**". *Interactivity, Game Creation, Design, Learning, and Innovation*. Springer, Jan. 2018, 85–94 Cited on page 50.

[WSWP85] WU, HSIN-I, SHARPE, PETER J.H., WALKER, JOE and PENRIDGE, LES K. "**'Ecological field theory: A spatial analysis of resource interference among plants'**". *Ecological Modelling* 29 (1-4 Sept. 1985), 215–243. ISSN: 03043800. URL: https://linkinghub.elsevier.com/retrieve/pii/0304380085900547 Cited on pages 84, 93, 94.

[WXF*17] WANG, PENG, XU, JINYU, FANG, XINYU, et al. "**'Ultrasonic time-frequency method to evaluate the deterioration properties of rock suffered from freeze-thaw weathering'**". *Cold Regions Science and Technology* 143 (Nov. 2017), 13–22. ISSN: 0165232X Cited on page 130.

[WXL*24] WANG, XIAOKUN, XU, YANRUI, LIU, SINUO, et al. **Physics-based fluid simulation in computer graphics: Survey, research trends, and challenges**. Oct. 2024. URL: https://github.com/WillDreamer/Awesome-AI4CFD. Cited on page 140.

[WZF*03] WEI, X, ZHAO, Y, FAN, Z, et al. "**'Blowing In the Wind'**". *Symposium on Computer Animation* (July 2003), 75–85. ISSN: 15383598. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=de7c0de4040df7725de7d11f25f2d96258fee5b8 Cited on page 146.

[WZF*25] WANG, SINAN, ZHOU, JUNWEI, FENG, FAN, et al. "**'Fluid Simulation on Vortex Particle Flow Maps'**". *ACM Transactions on Graphics* 44 (2025), 24. URL: https://doi.org/10.1145/3731198 Cited on page 144.

[XKG*16] XING, JUN, KAZI, RUBAIAT HABIB, GROSSMAN, TOVI, et al. "**'Energy-brushes: Interactive tools for illustrating stylized elemental dynamics'**". *UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), 755–766 Cited on page 144.

[XKV22] XIAO, ZHISHENG, KREIS, KARSTEN and VAHDAT, ARASH. "**'Tackling the Generative Learning Trilemma with Denoising Diffusion GANs'**". (Apr. 2022). URL: http://arxiv.org/abs/2112.07804 Cited on page 52.

[YCYW20] YAN, GUOWEI, CHEN, ZHILI, YANG, JIMEI and WANG, HUAMIN. "**'Interactive liquid splash modeling by user sketches'**". *ACM Transactions on Graphics* 39 (6 Nov. 2020). ISSN: 15577368 Cited on page 144.

[YLPY24] YUAN, MEI HUA, LIN, KUAN TING, PAN, SHU YUAN and YANG, CHIH KAI. **Exploring coral reef benefits: A systematic SEEA-driven review**. Nov. 2024 Cited on page 13.

[YSL05] YANG, B., SHI, W. and LI, Q. "**'An integrated TIN and Grid method for constructing multi-resolution digital terrain models'**". *International Journal of Geographical Information Science* 19 (10 Nov. 2005), 1019–1038. ISSN: 13658816 Cited on page 24.

[YZKF20] YAN, PENG, ZHANG, JINHUA, KONG, XIANGZHEN and FANG, QIN. "**'Numerical simulation of rockfall trajectory with consideration of arbitrary shapes of falling rocks and terrain'**". *Computers and Geotechnics* 122 (June 2020). ISSN: 18737633 Cited on pages 149, 170.

[ZCZ*20] ZHU, DI, CHENG, XIMENG, ZHANG, FAN, et al. "**'Spatial interpolation using conditional generative adversarial neural networks'**". *International Journal of Geographical Information Science* 34 (4 Apr. 2020), 735–758. ISSN: 13623087 Cited on page 78.

[ZD20] ZHANG, BO and DEANGELIS, DONALD L. **An overview of agent-based models in plant biology and ecology**. Sept. 2020 Cited on page 87.

[ZKL*17] ZHANG, SHENFAN, KONG, FANLONG, LI, CHEN, et al. "**'Hybrid modeling of multiphysical processes for particle-based volcano animation'**". *Computer Animation and Virtual Worlds*. Vol. 28. John Wiley and Sons Ltd, May 2017 Cited on page 139.

[ZZL08] ZHU, QING, ZHANG, YETING and LI, FENGCHUN. "**'Three-dimensional TIN algorithm for digital terrain modeling'**". *Geo-Spatial Information Science* 11 (2 June 2008), 79–85. ISSN: 10095020 Cited on page 24.

[ZZP*18] ZHU, JUN-YAN, ZHANG, RICHARD, PATHAK, DEEPAK, et al. "**'Toward Multimodal Image-to-Image Translation'**". (Oct. 2018). URL: http://arxiv.org/abs/1711.11586 Cited on page 52.

Appendix A

# Smoothmax Function

*Boolean operations such as union, intersection, and difference are fundamental in logic and have been widely adopted in computer graphics, particularly in Constructive Solid Geometry (CSG). In this context, they are used to combine geometric primitives into complex shapes. However, while boolean functions are naturally discontinuous, modern rendering pipelines typically favor smoothness to enable effects such as anti-aliasing, gradient shading, and physical simulation. This has led to the development of smooth operators: continuous, differentiable approximations of boolean functions. The ideal objective is to construct operators that are not only smooth but infinitely differentiable, also known as functions belonging to the class $C^\infty$.*

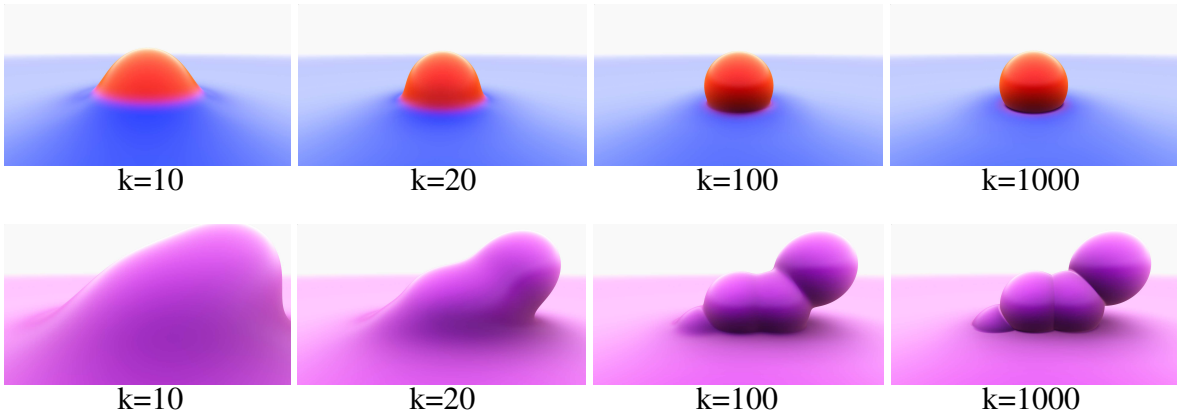



**Figure A.1:** *Top: Smooth union of a sphere of radius 0.1 with a plane rendered with Shadertoy shows the effect of the sharpness parameter k. Lower values blends smoothly while larger values approximate the boolean union operator. Bottom: Multiple spheres blended together with the same setup with the same k values by chaining the* smax *operator.*

## A.1 Definition of the `Smoothmax` function

In Section 3.4.2, we introduced the `Smoothmax` operator $\text{smax} : \mathbb{R}^2 \mapsto \mathbb{R}$ as a smooth approximation of the max operator, defined with the sharpness parameter $k > 0$ as:

$$\text{smax}(a,b) = a + \frac{1}{2} \cdot \frac{b-a}{1+e^{-k(b-a)}} + \frac{1}{2} \cdot \frac{b-a}{1-e^{-k(b-a)}}. \tag{A.1}$$


**Figure A.2:** *Plotting the function $f(x)$ suggests continuity, with the sharpness parameter k equal to 1, 2, 5, 10 for the curves red, green, orange and blue respectively; although it is initially undefined at $x = 0$.*

We observe that for $a = b$, the final term becomes undefined due to a removable singularity in the expression $\frac{b-a}{1-e^{-k(b-a)}}$, potentially introducing a discontinuity. In this section, we demonstrate that this function is in fact continuous on $\mathbb{R}^2$ and is infinitely differentiable, i.e., $\text{smax} \in C^\infty$.

It is evident that the only potentially problematic term in smax is $\frac{b-a}{1-e^{-k(b-a)}}$, as the rest of the expression is composed of standard smooth ($C^\infty$) functions.

To simplify our analysis, define the auxiliary function:

$$f(x) = \frac{x}{1 - e^{-kx}}. \tag{A.2}$$

Then we express

$$\text{smax}(a, b) = a + \frac{1}{2} \cdot \frac{b - a}{1 + e^{-k(b-a)}} + \frac{1}{2} f(b - a).$$

To assess the continuity and differentiability of smax, it suffices to analyze $f$ in a neighborhood of $x = 0$, where the apparent singularity arises.

## A.2 Continuity: $C^0$

From Figure A.2, we observe that $f$ appears continuous at $x = 0$. However, directly evaluating $f(0)$ yields the indeterminate form $\frac{0}{0}$. To resolve this, we apply L'Hôpital's Rule:

$$\lim_{x \mapsto 0} f(x) = \lim_{x \mapsto 0} \frac{x}{1 - e^{-kx}} = \frac{g'(x)}{h'(x)},$$

where

$$\begin{aligned} g(x) &= x, & g'(x) &= 1, \\ h(x) &= 1 - e^{-kx}, & h'(x) &= ke^{-kx}. \end{aligned}$$

Thus,

$$\lim_{x \mapsto 0} \frac{g'(x)}{h'(x)} = \lim_{x \mapsto 0} \frac{1}{ke^{-kx}} = \frac{1}{k}. \tag{A.3}$$

Since the limit exists and equals $\frac{1}{k}$, we define

$$f(0) := \frac{1}{k}. \tag{A.4}$$

This definition removes the singularity and ensures that $f$ is continuous on $\mathbb{R}$. For the case $a = b$, we then have

$$\text{smax}(a, b) = a + \frac{1}{2k}. \tag{A.5}$$

The complete definitions of the functions using Equations (A.2) and (A.4) and Equations (A.1) and (A.5) are:

$$f(x) = \begin{cases} \dfrac{x}{1 - e^{-kx}} & , x \neq 0 \\ \dfrac{1}{k} & , x = 0 \end{cases},$$

$$\text{smax}(a, b) = \begin{cases} a + \dfrac{1}{2} \cdot \dfrac{b - a}{1 + e^{-k(b-a)}} + \dfrac{1}{2} \cdot \dfrac{b - a}{1 - e^{-k(b-a)}} & , a \neq b \\ a + \dfrac{1}{2k} & , a = b \end{cases}.$$

We have shown that $f$ is continuous on $\mathbb{R}$, and thus smax is continuous on $\mathbb{R}^2$, meaning $\text{smax} \in C^0$.

## A.3 Smoothness: $C^\infty$

In this section, we prove that smax is infinitely differentiable. Since it is composed of standard smooth functions and the auxiliary function $f(x) = \frac{x}{1-e^{-kx}}$, it suffices to show that $f \in C^\infty(\mathbb{R})$.

We prove this by induction on the number of derivatives of $f$.

- **Base case:** As shown earlier, $f$ is continuous on $\mathbb{R}$ with the singularity at $x = 0$ removed by defining $f(0) := \frac{1}{k}$. Hence, $f \in C^0$.
- **Inductive step:** Assume that $f^{(n)}$ exists, is continuous on $\mathbb{R}$, and is expressible in the form

$$f^{(n)}(x) = \frac{P_n(x, e^{-kx})}{(1 - e^{-kx})^{n+1}},$$

where $P_n$ is a smooth function of $x$ and $e^{-kx}$. This form arises naturally from repeated applications of the quotient and chain rules:

- The denominator gains a factor of $(1 - e^{-kx})$ with each differentiation, hence the power $(n+1)$.
- The numerator $P_n$ is a recursively constructed smooth function resulting from differentiation of $P_{n-1}$ and the chain rule applied to $e^{-kx}$. Since $e^{-kx}$ is smooth and all operations preserve smoothness, $P_n(x, e^{-kx})$ remains smooth.

Differentiating $f^{(n)}$ using the quotient rule $\left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$, we get

$$f^{(n+1)}(x) = \frac{d}{dx}\left(\frac{P_n(x, e^{-kx})}{(1 - e^{-kx})^{n+1}}\right),$$

which results in a new expression:

$$f^{(n+1)}(x) = \frac{P_{n+1}(x, e^{-kx})}{(1 - e^{-kx})^{n+2}},$$

where $P_{n+1}$ is again a smooth function of $x$ and $e^{-kx}$, obtained via product, chain, and sum rules. Importantly, we do not need the explicit form of $P_{n+1}$, it suffices to know it is smooth.

At $x = 0$, both numerator and denominator vanish, but to the same order. The denominator vanishes to order $n + 2$ due to the Taylor expansion:

$$1 - e^{-kx} = kx - \frac{k^2x^2}{2} + \frac{k^3x^3}{6} - \frac{k^4x^4}{24} + \frac{k^5x^5}{120} + O(x^6),$$

and $P_{n+1}(x, e^{-kx})$ also vanishes to order at least $n + 2$ due to the construction. Thus, any singularity at $x = 0$ is removable, and $f^{(n+1)}$ can be continuously extended there.

By induction, $f^{(n)}$ exists and is continuous for all $n \geq 0$, so $f \in C^\infty(\mathbb{R})$. Since smax is built from smooth functions and $f$, we conclude that $\text{smax} \in C^\infty(\mathbb{R}^2)$.

Moreover, since $f(x) = \frac{x}{1-e^{-kx}}$ is composed of analytic functions with a removable singularity at $x = 0$, $f$ is real-analytic on $\mathbb{R}$. Consequently, smax is also real-analytic on $\mathbb{R}^2$.

## A.4 Operator's symmetry

We will briefly prove the symmetry of smax by showing that $\text{smax}(a, b) = \text{smax}(b, a)$. The case of $a = b$ is obviously true.

Let's note $x = b - a$, and decompose $\text{smax}(a, b) = a + g(x)$. We want to show that for all $a, b$ we have

$$\begin{aligned} &\text{smax}(a, b) = \text{smax}(b, a) \\ \iff &a + g(x) = (a + x) + g(-x) \\ \iff &g(x) - g(-x) = x. \end{aligned} \tag{A.6}$$

By rearranging the function $g$, we obtain

$$\begin{aligned} g(x) &= \frac{x}{2}\left(\frac{1}{1 - e^{-kx}} + \frac{1}{1 + e^{-kx}}\right) \\ &= \frac{x}{2}\left(\frac{1 - e^{-kx}}{(1 + e^{-kx})(1 - e^{-kx})} + \frac{1 + e^{-kx}}{(1 + e^{-kx})(1 - e^{-kx})}\right) \\ &= \frac{x}{2} \cdot \frac{2}{1 - e^{-2kx}} \\ &= \frac{x}{1 - e^{-2kx}}. \end{aligned} \tag{A.7}$$

Developing Equation (A.6) and let $r = e^{2kx} > 0$:

$$g(x) - g(-x) = \frac{x}{1-e^{-2kx}} - \frac{-x}{1-e^{2kx}} \tag{A.8}$$

$$= \frac{x}{(1-e^{-2kx})} \cdot \frac{e^{2kx}}{e^{2kx}} + \frac{x}{1-r} \tag{A.9}$$

$$= x\left(\frac{r}{r-1} + \frac{1}{1-r}\right) \tag{A.10}$$

$$= x\left(\frac{1-r}{1-r}\right) = x. \tag{A.11}$$

We shown that $g(x) - g(-x) = x$, resulting in the proof that $\operatorname{smax}(a,b) = \operatorname{smax}(b,a)$ for all $a, b$.

## A.5 Practical expressions for $\operatorname{smax}(a,b)$ and its derivatives

In applications such as shading, soft blending, and physical simulation, efficient evaluation of $\operatorname{smax}(a,b)$ and its first and second derivatives is often required. This section provides compact, implementation-ready expressions for these quantities.

### Function value

$$\operatorname{smax}(a,b) = \begin{cases} a + \frac{(b-a)}{2}\left(\frac{1}{1+e^{-k(b-a)}} + \frac{1}{1-e^{-k(b-a)}}\right) & \text{if } a \neq b \\ a + \frac{1}{2k} & \text{if } a = b \end{cases} \tag{A.12}$$

### First derivatives

$$\frac{\partial}{\partial a}\operatorname{smax}(a,b) = -g_1(a,b) \tag{A.13}$$

$$\frac{\partial}{\partial b}\operatorname{smax}(a,b) = g_1(a,b) \tag{A.14}$$

with

$$g_1(a,b) = \begin{cases} \frac{e^{2k(b-a)}\left(e^{2k(b-a)}-1\right)}{\left(e^{2k(b-a)}-1\right)^2} & \text{if } a \neq b \\ \frac{1}{2} & \text{if } a = b \end{cases}$$

### Second derivatives

$$\frac{\partial^2}{\partial a^2}\operatorname{smax}(a,b) = \frac{\partial^2}{\partial b^2}\operatorname{smax}(a,b) = -g_2(a,b) \tag{A.15}$$

$$\frac{\partial^2}{\partial a \partial b}\operatorname{smax}(a,b) = \frac{\partial^2}{\partial b \partial a}\operatorname{smax}(a,b) = g_2(a,b) \tag{A.16}$$

where the second derivative of the core function is given by

$$g_2(a,b) = \begin{cases} \frac{4ke^{2ka}e^{2kb}\left(e^{2ka}(ka-kb-1)+e^{2kb}(ka-kb+1)\right)}{\left(e^{2kb}-e^{2ka}\right)^3} & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases}.$$

The expression is algebraically heavy, but all terms are composed of simple exponentials, products, and powers, making it fast and safe to compute in practice, and the redundant components can be cached (in particular $e^{2ka}$ and $e^{2kb}$). At $x = 0$, we have $g_2(0) = 0$ by symmetry and smoothness of the underlying expression.

A smin operator can be derived directly from the smax by reversing the sign of the sharpness parameter $k_{\text{smoothmin}} = -k_{\text{Smoothmax}}$.

# A.6 Comparison with other smooth maximum functions

To evaluate the practical performance of the `Smoothmax` operator, we compared it with two commonly used alternatives:

- `LogSumExp`:

$$\texttt{LSE}(a,b) = \frac{1}{k}\log\left(e^{ka}+e^{kb}\right) \tag{A.17}$$

  This is a popular smooth approximation to $\max(a,b)$, widely used in machine learning.

- `Smooth Absolute Max`:

$$\texttt{AbsMax}(a,b) = \frac{a+b}{2} + \frac{1}{2k}\log\left(1+e^{-k|a-b|}\right) \tag{A.18}$$

  A symmetric blend of the average and the absolute difference.

- `Smoothmax`:

$$\text{smax}(a,b) = \begin{cases} a + \frac{(b-a)}{2}\left(\frac{1}{1+e^{-k(b-a)}} + \frac{1}{1-e^{-k(b-a)}}\right) & \text{if } a \neq b \\ a + \frac{1}{2k} & \text{if } a = b \end{cases}$$

  Our symmetric and infinitely differentiable approximation, designed to behave well under chaining and small sharpness variations.

We compare these functions across four benchmarks: chaining behavior, sharpness resolution, runtime performance, and aggregation error.

## Chaining behavior

We evaluate each operator by chaining it over $n$ inputs drawn randomly from $[0, 100]$. The true maximum is compared to the approximate maximum, and errors are averaged over 100 trials per $n$. Figure A.3 shows that the `Smoothmax` operator consistently provide an accuracy multiple order of magnitude higher than the Smooth Absolute Max operator, and similar order of magnitude than the `LogSumExp` operator.

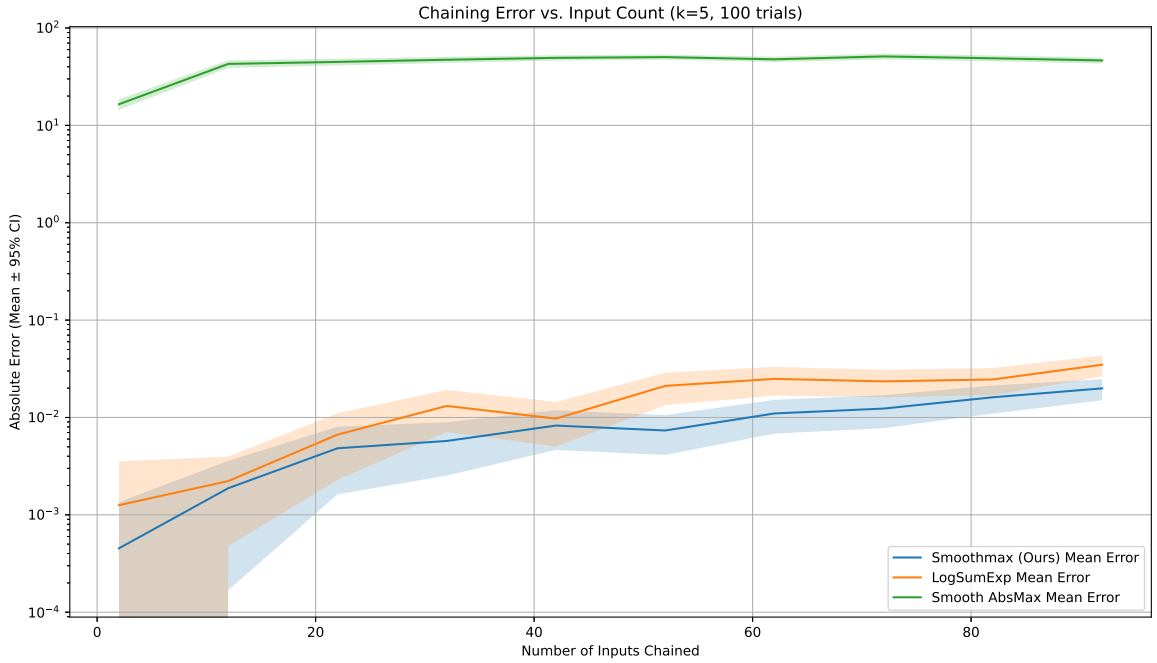



**Figure A.3:** *Mean absolute error as more values are chained together ($k = 5$).*

## Precision and sharpness

We also test how each function converges to $\max(a, b)$ as sharpness $k$ increases. We fix two very close inputs $a = 1.0000$ and $b = 1.0001$ and sweep $k$ over several orders of magnitude. We see in Figure A.4 that for small values of $k$, the `AbsMax` operator is closer to the ground truth, but as $k$ tends to exaggerated high values, the `Smoothmax` operator gets significantly more accurate. When fixing $a = 1.0000$ and $b = 1.01$ or $b = 2.0$, `AbsMax` operator's error becomes higher by multiple order of magnitude, while the `LogSumExp` and the `Smoothmax` operators have identical behaviours.

## Runtime benchmark

To get a rough idea of performance, we measured the average time (in microseconds) for a single call of each function over 10 000 repetitions, using scalar inputs and $k = 5$. Results are presented in Table A.1. This test was conducted with using Python 3.10 and NumPy 1.25 on a Linux system with a modern multi-core CPU and 14GB of RAM. We note that the `LogSumExp` function is more time consuming as it contains two exponential componants and a log function, with a risk of overflow that for large values of $a$, $b$ or $k$. On the other hand, the `Smoothmax` operator contains a single unique exponential componant that quickly fall to 0 for large values of $k$ or difference between $a$ and $b$, keeping it stable (Figure A.5 presents floating point overflows).

Fixing $k = 2.0$, $a = 0.0$ and limiting data type to 4 bytes floating points representation, `LogSumExp` and AbsSumMax overflows at $b = 45.0$, and $b = 355.0$ for 8 bytes representations, while the `Smoothmax` function does not have limit as `exp(-k * (b - a))` is simply truncated to 0 as $k(b-a)$ gets large. Moreover, the `Smoothmax` computation rely on the difference between the input (scaled by $k$) in the exponential operators, while `LogSumExp` rely on the sum of $e^{ak}$ and $e^{bk}$, which overflow quickly if $a$ and $b$ are both larger than 1.

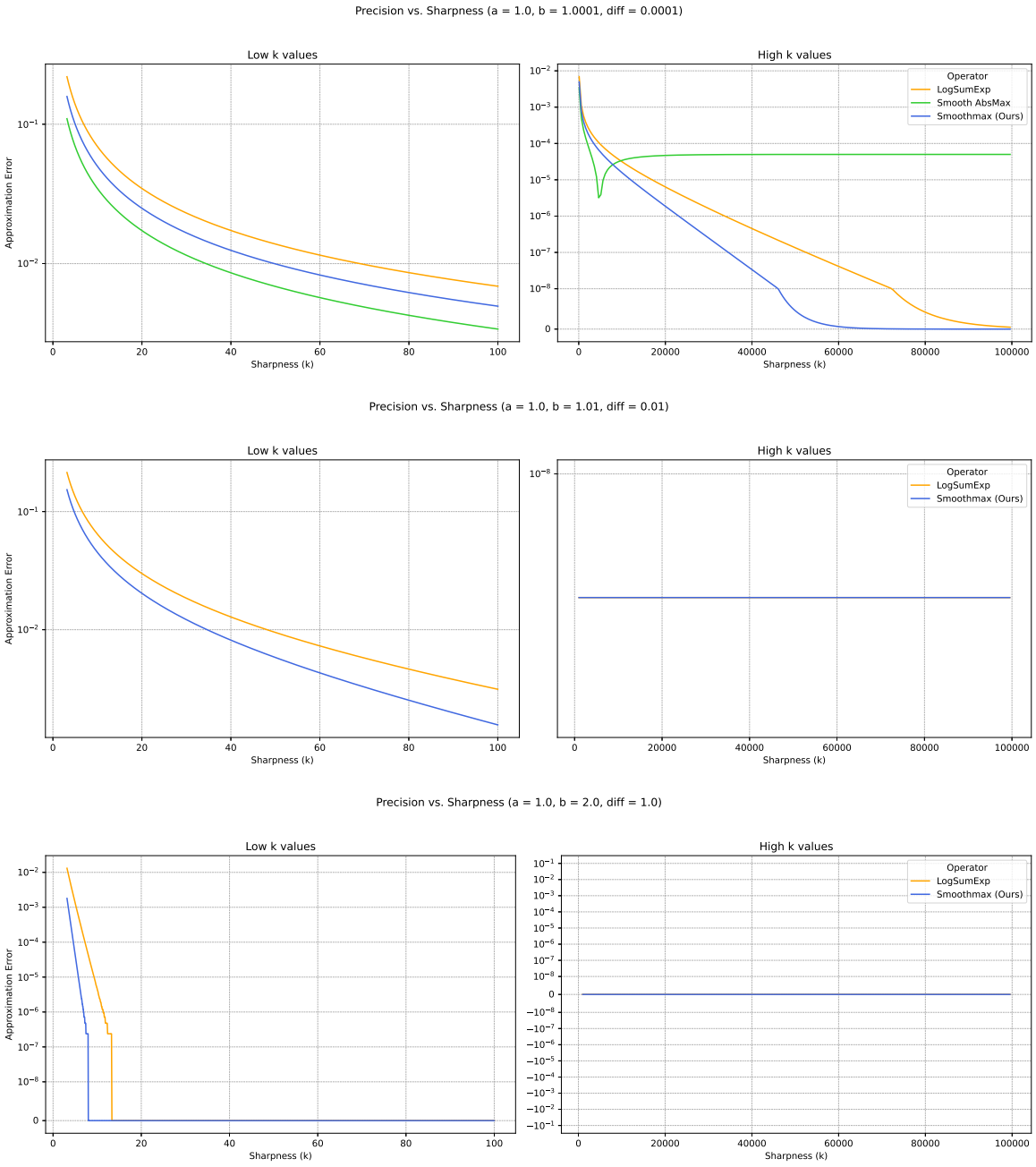



**Figure A.4:** *Approximation error as a function of $k$ for different values of $a$ and $b$ compared to* $\max(a, b)$. *Error is greater for smaller differences between $a$ and $b$. Left shows that the operator is comparable to all other operators for sharpness values below 100; right show that for large sharpness values, the* `Smoothmax` *operator behave similarly as* `LogSumExp`*, while the* `Smooth Absolute Maximum` *operator stagnate with a lower accuracy than the others (error of* `Smooth Absolute Maximum` *is too large to be displayed when the difference between $a$ and $b$ exceeds 0.001).*

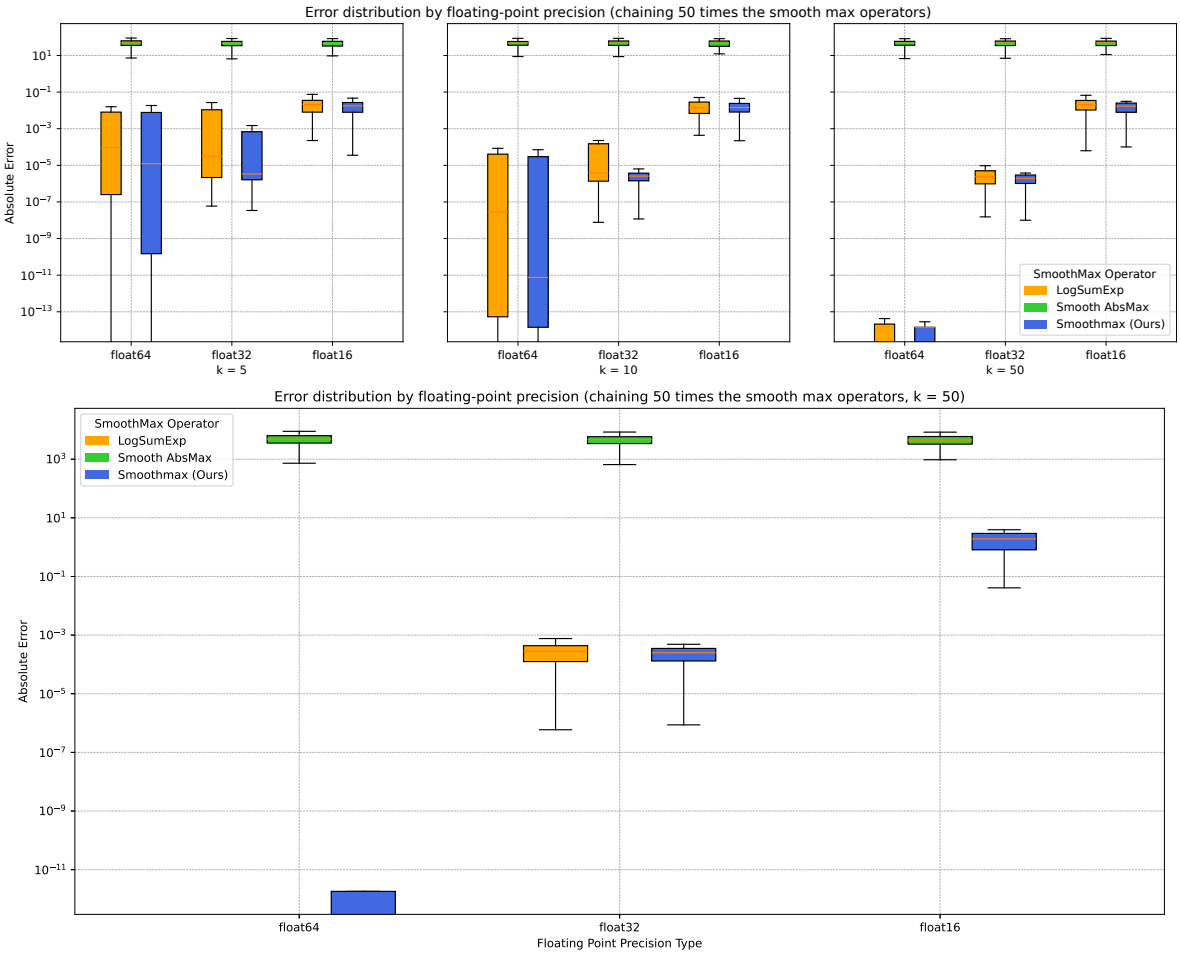



**Figure A.5:** *Top: Comparison of error between the* `LogSumExp`*, Smooth* `AbsMax` *and the* `Smoothmax` *operators when chaining the operators on data between 0 and 100 while restricting the floating point precision, with from left to right respectively* $k = 5$*,* $k = 10$*, and* $k = 50$*. Bottom: We chained the operators for values between 0 and 1000, which raise overflows for* `LogSumExp` *(thus the data not displayed for 'float64' and 'float16').*

| Function | Time |
|---|---|
| `Smoothmax` | 1.40 |
| `LogSumExp` | 2.68 |
| `AbsMax` | 1.80 |

Table A.1: Average runtime per call (in $\mu$s)

---

Overall, `Smoothmax` performs best across the board:

- It keeps errors low even when chaining over many values.
- It converges smoothly and accurately as sharpness increases.

- It is fast to compute and numerically stable.

This makes it a practical and robust replacement for $\max(a, b)$ in simulation and graphics tasks where smoothness matters (such as raymarching, as illustrated in Figure A.1).

## A.7 Relationship to prior work

After deriving the formula in Equation (A.7):

$$\operatorname{smax}(a, b) = a + \frac{b - a}{1 - e^{-2k(b-a)}},$$

we found it is algebraically equivalent to Quílez's "sigmoid" smooth-min function [Qui13] under the standard transformation $\operatorname{smax}(a, b; k) = -\operatorname{smin}(-a, -b; k_{\text{Quilez}})$ with the parameter mapping $k_{\text{Quilez}} = \frac{1}{2k}$. Accordingly, we do not claim novelty for the functional form. Our contributions are the rigorous smoothness/analyticity proofs, closed-form derivatives with $a = b$ limits, and numerically stable implementations and benchmarks.

Appendix B

# Computation of a metaball

*In this section we develop in more detail the formulation used for metaballs in Chapter 5 for the simulation of particle erosion on implicit terrains.*

## B.1 Linear metaball derivation

We use the following formula to evaluate a metaball in space with a center $\mathbf{c}$ and radius $R$:

$$g(\mathbf{p}) = \max\left(1 - \frac{|\mathbf{p}-\mathbf{c}|}{R},\, 0\right), \tag{B.1}$$

where the Euclidean distance is used, and $g$ is clamped to zero outside the sphere of radius $R$.

We have a total amount $Q$ to define in this space, so the final metaball function $f$ must satisfy

$$f(\mathbf{p}) = \lambda\, g(\mathbf{p}), \quad \int_{\mathbf{p}\in B_R(\mathbf{c})} f(\mathbf{p})\, d\mathbf{p} \qquad = Q, \tag{B.2}$$

where $B_R(\mathbf{c})$ denotes the ball of radius $R$ centered at $\mathbf{c}$.

Exploiting radial symmetry, we set $r = \frac{|\mathbf{p}-\mathbf{c}|}{R} \in [0,1]$ and write $g(\mathbf{p}) = 1 - r$. In spherical coordinates of the normalised point $\frac{(\mathbf{p}-\mathbf{c})}{R}$, the volume element is $R^3\, r^2 \sin(\theta)\, dr\, d\theta\, d\phi$. The integral of $g$ over $B_R(\mathbf{c})$ becomes

$$\begin{aligned}
&R^3 \int_0^1 \int_0^\pi \int_0^{2\pi} (1-r)\, r^2\, \sin(\theta)\, dr\, d\theta\, d\phi \\
&= R^3 \left[\int_0^1 (1-r) r^2\, dr\right] \left[\int_0^\pi \sin\theta\, d\theta\right] \left[\int_0^{2\pi} 1\, d\phi\right] \\
&= R^3 \times \frac{1}{12} \times 2 \times 2\pi \\
&= \frac{\pi}{3}\, R^3.
\end{aligned} \tag{B.3}$$

Given $\int f = Q$ from Equation (B.2) and $\int f = \lambda \int g$, we obtain

$$\lambda = \frac{Q}{\int g} = \frac{3}{\pi\, R^3}\, Q. \tag{B.4}$$

From Equation (B.2), the normalised linear metaball is

$$f(\mathbf{p}) = \frac{3\,Q}{\pi\,R^3}\ \max\!\left(1 - \frac{|\mathbf{p}-\mathbf{c}|}{R},\, 0\right), \tag{B.5}$$

representing the rate of change applied to the terrain's evaluation function.

### Derivatives

Let $\rho = |\mathbf{p}-\mathbf{c}|$, $r = \frac{\rho}{R}$, and $\vec{u} = \frac{(\mathbf{p}-\mathbf{c})}{\rho}$ (the unit radial vector). For $0 < r < 1$,

$$\nabla f(\mathbf{p}) = -\frac{\lambda}{R}\,\vec{u}, \qquad \Delta f(\mathbf{p}) = -\frac{2\lambda}{R^2\,r}. \tag{B.6}$$

We define by convention $\nabla f(\mathbf{c}) = 0$. The function $f$ is only $C^0$: the derivative $\phi'(r)$ of its radial profile is discontinuous at $r = 0$ and $r = 1$, and the Laplacian diverges as $r \mapsto 0$.

The integration in voxel space is out of the scope of this appendix; a numerical solution is instead proposed in Section 5.4.4.

## B.2 Other possible radial falloff functions

The linear falloff in Equation (B.5) is only one possible choice of radial profile. We write

$$x = \frac{|\mathbf{p}-\mathbf{c}|}{R}, \qquad g(\mathbf{p}) = \phi(x),$$

where $\phi : [0,\infty) \mapsto \mathbb{R}_{\geq 0}$ is compactly supported, i.e. $\phi(x) = 0$ for $x \geq 1$. If a smoother boundary is desired, pick $\phi$ such that $\phi(1) = 0$ and $\phi'(1) = 0$ (or higher-order vanishing derivatives).

We want to decompose the final function as $f(\mathbf{p}) = \lambda\,\phi(x)$ with $\lambda$ a scaling factor. Using spherical coordinates and radial symmetry,

$$\int_{\mathbf{p}\in\mathbb{R}^3} g(\mathbf{p})\,d\mathbf{p} = 4\pi\,R^3 \underbrace{\int_0^1 \phi(x)\,x^2\,dq}_{J_\phi}, \tag{B.7}$$

so that, given the total amount $Q$, the normalisation reads

$$\lambda = \frac{Q}{4\pi\,R^3\,J_\phi}, \qquad f(\mathbf{p}) = \lambda\,\phi\!\left(\frac{|\mathbf{p}-\mathbf{c}|}{R}\right). \tag{B.8}$$

For implementation, it is convenient to record the radial derivatives once. Let $\rho = |\mathbf{p}-\mathbf{c}|$, $x = \frac{\rho}{R}$, and $\vec{u} = \frac{(\mathbf{p}-\mathbf{c})}{\rho}$ (defined for $\rho > 0$). For $0 < x < 1$,

$$\nabla f(\mathbf{p}) = \frac{\lambda}{R}\,\phi'(x)\,\vec{u}, \qquad \Delta f(\mathbf{p}) = \frac{\lambda}{R^2}\left[\phi''(x) + \frac{2}{x}\,\phi'(x)\right]. \tag{B.9}$$

By convention, we set $\nabla f(\mathbf{c}) = 0$. The smoothness at $x = 0$ and $x = 1$ depends on $\phi$; if $\phi'(0) \neq 0$ or $\phi'(1) \neq 0$, the gradient is discontinuous at the corresponding point.

**Linear** (Figure B.1)

$$\begin{aligned} \phi(x) &= \max(1-x,0) \\ \phi'(x) &= -1 \\ \phi''(x) &= 0 \\ J_\phi &= \tfrac{1}{12} \\ \lambda &= \frac{3\,Q}{\pi\,R^3} \end{aligned} \tag{B.10}$$

*Continuity: $C^0$; gradient discontinuous at $x = 0$ and $x = 1$. A standard piecewise-linear choice.*

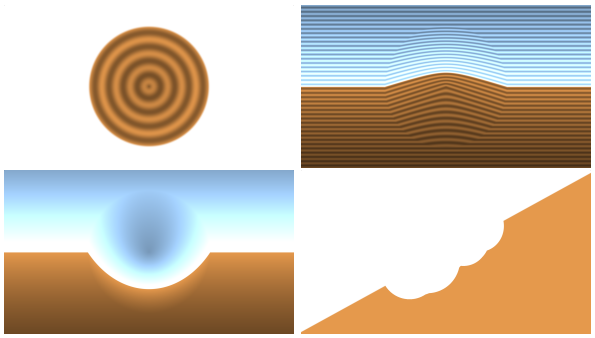


**Figure B.1:** *Top left: Linear profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Smoothstep** (Figure B.2)

$$\begin{aligned} \phi(x) &= \max\left(1-3x^2+2x^3,0\right) \\ \phi'(x) &= -6x+6x^2 \\ \phi''(x) &= -6+12x \\ J_\phi &= \tfrac{1}{15} \\ \lambda &= \frac{15\,Q}{4\pi\,R^3} \end{aligned} \tag{B.11}$$

*Continuity: $C^1$ with $\phi'(0) = \phi'(1) = 0$; widely used in graphics as the cubic smoothstep [Per02].*

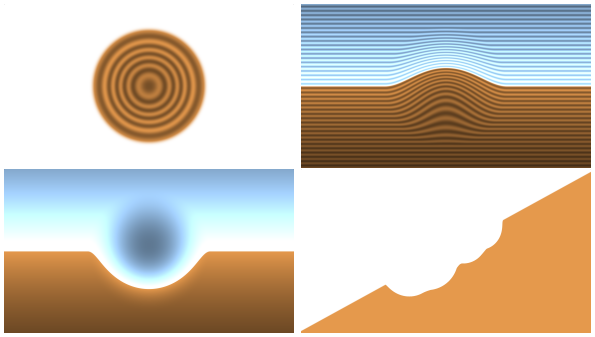


**Figure B.2:** *Top left: Smoothstep profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Smootherstep** (Figure B.3)

$$\begin{aligned} \phi(x) &= \max\left(1-10x^3+15x^4-6x^5,0\right) \\ \phi'(x) &= -30x^2+60x^3-30x^4 \\ \phi''(x) &= -60x+180x^2-120x^3 \\ J_\phi &= \tfrac{5}{84} \\ \lambda &= \frac{21\,Q}{5\pi\,R^3} \end{aligned} \tag{B.12}$$

*Continuity: $C^2$ with $\phi'(0) = \phi'(1) = 0$; quintic smootherstep popularized by Perlin for higher-order smoothness [Per02].*

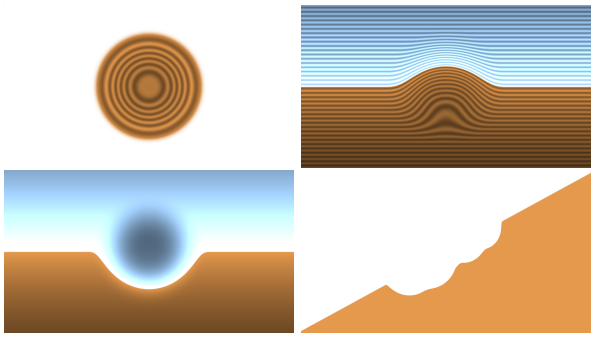


**Figure B.3:** *Top left: Smootherstep profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Wendland** (Figure B.4)

$$\begin{aligned} \phi(x) &= \max\left((1-x)^4(1+4x), 0\right) \\ \phi'(x) &= -20x(1-x)^3 \\ \phi''(x) &= -20(1-x)^2(1-4x) \\ J_\phi &= \tfrac{1}{42} \\ \lambda &= \frac{21\,Q}{2\pi\,R^3} \end{aligned} \tag{B.13}$$

*Continuity: $C^2$ in 3D; a compactly supported, positive definite RBF of minimal degree [Wen95].*

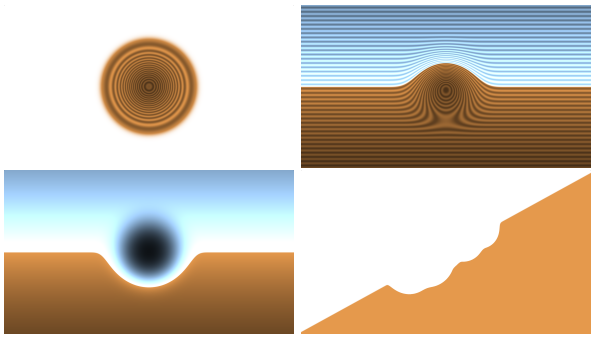


**Figure B.4:** *Top left: Wendland profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Biweight kernel** (Figure B.5)

$$\begin{aligned} \phi(x) &= \max\left((1-x^2)^2, 0\right) \\ \phi'(x) &= -4x(1-x^2) \\ \phi''(x) &= -4+12x^2 \\ J_\phi &= \tfrac{8}{105} \\ \lambda &= \frac{105\,Q}{32\pi\,R^3} \end{aligned} \tag{B.14}$$

*Continuity: $C^1$ at $x = 1$ (and $\phi'(0) = 0$). A quartic kernel from density estimation [Sil98, WJ93]).*

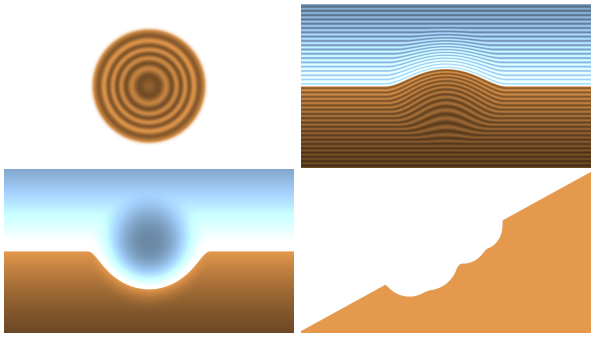


**Figure B.5:** *Top left: Biweight kernel profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Triweight kernel** (Figure B.6)

$$\begin{aligned} \phi(x) &= \max\left((1-x^2)^3, 0\right) \\ \phi'(x) &= -6x(1-x^2)^2 \\ \phi''(x) &= -6(1-x^2)(1-5x^2) \\ J_\phi &= \tfrac{16}{315} \\ \lambda &= \frac{315\,Q}{64\pi\,R^3} \end{aligned} \tag{B.15}$$

*Continuity: $C^2$ at $x = 1$ (and $\phi'(0) = 0$). A sextic kernel in the KDE literature [Sil98, WJ93].*

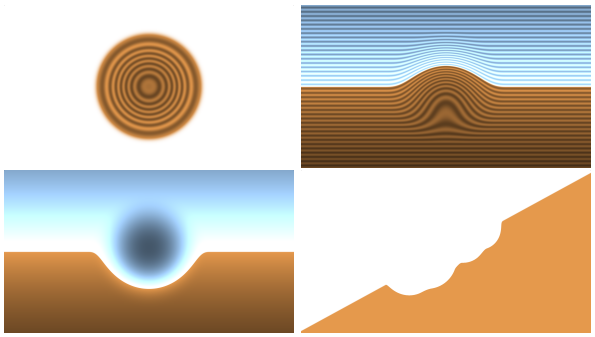


**Figure B.6:** *Top left: Triweight kernel profile. Top right: deposition on a plane $f_{plane}(x,y) = -y$ with $Q = 0.1$ and $R = 0.75$. Bottom left: erosion with $Q = -0.5$ and $R = 0.75$. Bottom right: multiple eroding metaballs on an inclined plane.*

**Raised cosine** (Figure B.7)

$$\begin{aligned} \phi(x) &= \max\left(\tfrac{1+\cos(\pi x)}{2}, 0\right) \\ \phi'(x) &= -\tfrac{\pi}{2}\sin(\pi x) \\ \phi''(x) &= -\tfrac{\pi^2}{2}\cos(\pi x) \\ J_\phi &= \tfrac{1}{6} - \tfrac{1}{\pi^2} \\ \lambda &= \frac{3\pi Q}{(2\pi^2 - 12) R^3} \end{aligned} \tag{B.16}$$

*Continuity:* $C^\infty$ *on* $(0,1)$ *with* $\phi'(0) = \phi'(1) = 0$*; a standard compactly supported kernel in statistics [SHM*13].*

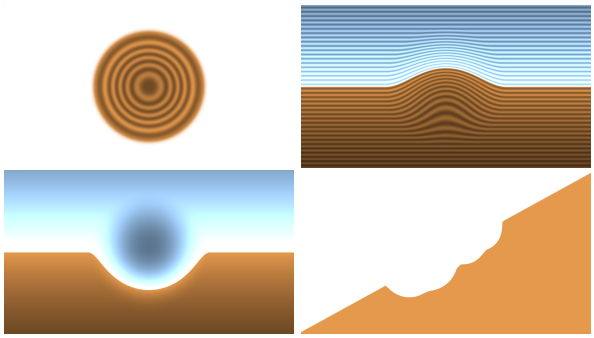


**Figure B.7:** *Top left: Raised cosine profile. Top right: deposition on a plane* $f_{plane}(x,y) = -y$ *with* $Q = 0.1$ *and* $R = 0.75$*. Bottom left: erosion with* $Q = -0.5$ *and* $R = 0.75$*. Bottom right: multiple eroding metaballs on an inclined plane.*

Any other compactly supported $\phi$ can be used: we just need to compute $J_\phi$ once (analytically or numerically) and plug it into Equation (B.8).

# List of Figures

# List of Tables

# List of Algorithms

# List of Equations