The Visual Computer
https://doi.org/10.1007/s00371-024-03444-w

RESEARCH

Flexible terrain erosion

Marc Hartley[1] · **Nicolas Mellado**[2] · **Christophe Fiorio**[1] · **Noura Faraj**[1]

Accepted: 30 April 2024

Abstract
In this paper, 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 utilize 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.

Keywords Procedural modeling · Terrain morphing · Natural phenomena · Erosion processes

1 Introduction

Automated terrain generation is a key component of natural scene digital modeling for animated movies and video games. A standard approach is to first generate a base terrain geometry using noise to define the height on the input domain [28, 31, 42], the result will most likely lack realism and feel synthetic. Erosion simulation algorithms are applied, to simulate thousands of years of aging by reproducing physical phenomena—i.e., effects of the elements (rain, wind, running water...)—affecting the terrain making it more believable [17, 44, 45]. 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 [29]: **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 (e.g. a cornice falls down a slope or a grain of sand is thrown into the air)—and **deposition**—transported pieces of land are accumulated at a new part of the landscape. Various phenomena can 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, modify the terrain [1, 8–11].

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 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 alteration 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 which grayscale values define terrain elevation. While being the major terrain

✉ Marc Hartley
marc.hartley@umontpellier.fr

Nicolas Mellado
nicolas.mellado@irit.fr

Christophe Fiorio
christophe.fiorio@lirmm.fr

Noura Faraj
noura.faraj@umontpellier.fr

[1] LIRMM, CNRS, Université de Montpellier, Montpellier, France

[2] IRIT, CNRS, Université de Toulouse, Toulouse, France

Published online: 05 June 2024

Fig. 1 Applying shading and textures on the generated geometry can produce 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

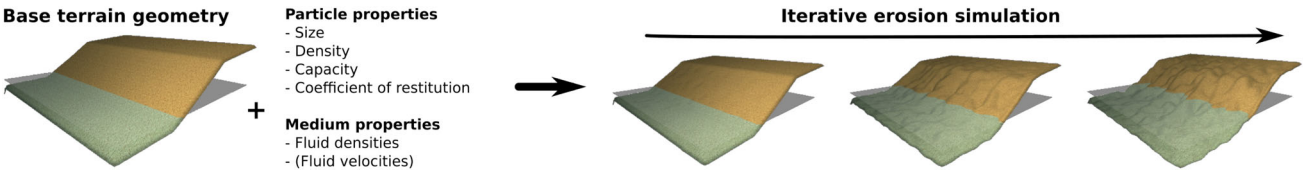


Fig. 2 Our method require a base geometry, a small number of parameters for the particles and the medium used for the erosion simulation. It can be easily adapted to be compatible with different mediums and terrain representations

representation, only a limited number of environments can be modeled. Indeed, natural landscapes are intrinsically 3D (overhangs, cavities or geological structures such as arches or gobelins), this is particularly true for underwater environments generation. Alternate representation such as voxel grids, material layers or implicit surfaces can be used. A wide variety of method have been proposed to simulate natural erosion phenomena on heightmaps as the partial differential equations to model erosion can be discretized 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) [25] 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 part of the geomorphological and meteorological phenomena present in the literature of terrain generation (including 3D and volumetric effects). We introduce a generalized 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 (Fig. 2). 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 will collide with the terrain, perform terrain alteration at the collision point and transport material along its path. Their 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 representation.

The main contributions of this paper are:

- a generalized 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.

2 State of the art

In this section, we first present the major terrain representations (height fields, layered representations, voxel grids, and scalar functions) and a subset of the major simulated phenomena used to erode terrains. 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.

2.1 Terrain representations

A terrain can be represented in various ways, each of them suited for a given application of which we give an brief overview, more details can be found in [17].

Height Fields represents the surface of the terrain by defining the elevation at each point in a 2D grid. This representation is simple, regular, and fast to process allowing for easy manipulation, such as raising or lowering the terrain [15, 16].

Layered Representations are an extension of height fields using a 2D grid where each cell represents a stack of different materials instead of a simple height [6, 36] allowing for memory efficient representation of volumetric terrains. To create complex structures, arches or caves, solid materials can be transformed into more granular ones, that can be stabilized [36].

Voxel Grids are regular, uniform volumetric grids that encode information on the presence or absence of material for each 3D point in the domain. Voxel grids are advantageous due to their regularity [13] and ability to represent volumetric models at the cost of high memory footprint, which has limited their use in terrain generation [21, 23, 26]. We consider two voxel grid representations: *density-voxel* grids for which each voxel contains a scalar value, for instance the occupation percentage [14] and *binary voxel* grids that can be seen as a mask containing the presence of material information.

Implicit terrains represent landscapes as an implicit surface defined by a scalar function. This allows for high definition large terrain modeling. The application of combinable scalar function overlays [18] or the definition of user-defined gradients [19] can be used to create complex terrain features. Altering a implicitly defined surface is a challenging task hence limited option exist for erosion simulation [33].

2.2 Erosion processes

Erosion processes play a crucial role in shaping landscapes over time. We present different kind of erosion and how they apply to given terrain representations. Note that using existing methods all erosion methods cannot be used on all representation.

Thermal Erosion is driven by large temperature shifts, transferring material based on slope thresholds. The process is iterative, redistributing material until slopes stabilize. It can be computed efficiently on height fields and layered terrains due to their manipulable height nature [6, 28, 36]. However, its application on voxel grids is challenging due to limited Z-axis resolution.

Hydraulic Erosion stems from water movement, eroding and depositing sediment based on water flow intensity. 2.5D terrains are widely studied for this simulation, using either water slope velocities [29] or water simulations for erosion effects [27]. For smaller scales, 3D fluid simulations on voxel grids have been proposed [5]. Kristof et al [25] used SPH (Smoothed Particle Hydrodynamics) for meshless erosion simulations on various terrains. Their method involves numerous particle interactions, demanding significant computational power. Our approach draws inspiration from this but enhances efficiency by removing certain particle interactions.

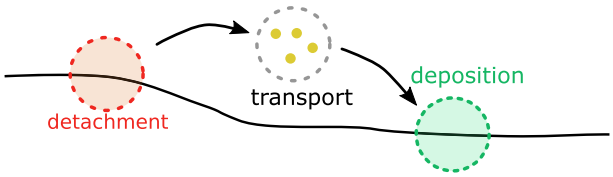


Fig. 3 Three steps of the erosion process from the sediment point of view: detachment from its original location—dotted red circle—, transport in a fluid—dotted black circle—, deposition at a new location—dotted green circle

Wind Erosion shifts material through wind force, notably impacting areas with fine surface particles like deserts. It has been modeled on discrete height fields [35, 40] by mimicking sand's wind-driven trajectory and using thermal erosion for corrections. This process is simulated by iteratively displacing small amounts of matter, which make it less suitable for representations with discrete height resolution.

Erosion by Other Forces includes influences like glaciers, snow, tectonic movements, and fauna, each introducing distinctive terrain patterns, enriching its intricacy [8–12, 43]. However, most methods are tailored by a given terrain representation, often the height fields, and might not be applicable to other representations due to their intrinsic properties.

3 Particle erosion

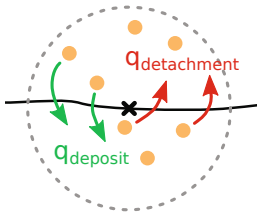
Erosion occurs in three stages: material detachment, transport and deposition (respectively in red, black and green in Fig. 3). In our approach, particles move through the medium following its flow (i.e. wind in air or currents in water) and then absorb or deposit a small amount of material upon contact with the land surface, effectively fulfilling the three stages of erosion.

3.1 Overview

Particles are transported through the medium and can pass through several different media. Each medium is defined by a density and a flow. Consider, for example, water density to be 1000kg m^{-3} and that of air to be 1kg 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 precalculated 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 reduces significantly the overall execution time but also offers users high flexibility over the quality of the simulation and simplify the implementation.

3.2 Erosion process

Every time the particle hits the ground, a given amount $q_{\text{detachment}}$ of sediment is detached from the ground (red arrows) while another amount q_{deposit} of sediments is deposed at this location (green arrows). Our erosion model is based on the work of Wojtan et al where regular 3D grids are used to estimate the fluid velocity and sediment transport [52]. In the spirit of [25], 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.



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, that we call abrasion. We use pseudoplastics model to approximate the amount of matter removed due to the shear forces while considering the physical properties of the fluid and the ground [52].

The shear rate θ is approximated by the relative velocity of the fluid to the solid boundary υ_{rel} over a short distance l. We approximate the shear stress τ at the solid boundary by a power-law:

$$\tau = K\theta^n \tag{1}$$

where $\theta = \upsilon_{\text{rel}}/l$, K is the shear stress constant (often set to 1) and $n \in [0, 1]$ is the flow behavior 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 ε at any contact point between a fluid and a solid boundary using (1) by

$$\varepsilon = K_\varepsilon(\tau - \tau_c)^a \tag{2}$$

with $K_\varepsilon \in [0, 1]$ a user-defined erosion constant, τ_c the critical shear stress value for which the matter starts to behave like 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 half sphere, of radius R, in the normal opposite direction at the particle impact point (Fig. 6). We then use (2):

$$q_{\text{detachment}} = \varepsilon\frac{2\pi R^3}{3} \tag{3}$$

to get the final eroded amount $q_{\text{detachment}}$. The particle is also defined by a maximal amount of sediments that can be contained in its volume before being saturated noted $C_{\max}$. Note that this constant will be used for the settling velocity computation (7).

Deposition. The eroded sediments are considered in suspension in a fluid and are affected by its velocity. A fluid particle then transports the sediments in its flow until gravity settles it onto the ground again. The effect of gravity is modeled by a settling velocity w_s defined in Eq (7). We consider that the amount of sediment settled is proportional to the norm of the settling velocity as proposed in [52] with $\omega \in [0, 1]$:

$$q_{\text{deposit}} = \omega||w_s||. \tag{4}$$

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 to constantly have a terrain in a plausible state, while giving the possibility to increase the aging effect by running more iterations. Note that, reducing progressively the overall erosion strength can be used as a strategy to adapt the computation time to a chosen level-of-details.

We first present how to compute the particle speed using particle's physics then how to add optional medium velocity field to add a fluid simulation or user control.

Particle's physics. From its independence with other particles: we consider each particle following Newton's laws of motion.

First, we define the external forces F_{ext} applied on each particle, we consider gravity and buoyancy. We calculate the buoyancy force $B = -\rho_f V\vec{g}$ with ρ_f the density of the fluid, V the volume of the particle and g the gravitational acceleration, but we can also calculate the force of gravity $G = M_p\vec{g}$ with M_p the mass of the particle. We then have the final external force $F_{\text{ext}} = \vec{G} + \vec{B} = M_p\vec{g} - \upsilon_f V\vec{g}$ knowing the density of an object $\rho_p = \frac{M_p}{V}$, we have:

$$F_{\text{ext}} = V\vec{g}(\rho_p - \rho_f). \tag{5}$$

The particle velocity υ can be integrated from (5) by:

$$\upsilon = \int F_{\text{ext}}\,\mathrm{d}t + w_s + \upsilon_0, \tag{6}$$

Fig. 4 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)

with w_s the settling speed of sediments in a fluid with a viscosity μ given by Stoke's Law [48]:

$$w_s = \frac{2}{9} g R^2 \frac{(\rho_p - \rho_f)}{\mu} f(C). \tag{7}$$

We use the Richardson–Zaki relation as the hindered settling coefficient:

$$f(C) = 1 - \left(\frac{C}{C_{\max}}\right)^n$$

with C and $C_{\max}$, respectively, the fraction of volume of sediments contained and the maximal fraction of sediments the particle can contain, and n an exponent typically 4–5.5, which we set to 5 [38, 52].

Finally, the particle position can be integrated as:

$$p = \int v \, \mathrm{d}t + p_0.$$

When the particle hits the ground, a coefficient of restitution affects its behavior by reducing its velocity post-collision. This value depends on ground material as it is influenced mainly by the material's particle shape, coefficient of friction and density [53]. Less bouncy particles lose speed quickly and settle down sooner, forming a steeper pile (Fig. 4 blue), or a higher talus angle like chalk. On the other hand, more bouncy particles disperse more widely upon hitting a surface, resulting in a gentler accumulation like clay (Fig. 4 red).

Velocity field. In our model, we allow the user to add a velocity field to the environment that influences particles motion. This velocity field can be the result of a complex fluid simulation, a uniform vector field, or an artistic motion field. We modify Equation (6) such that the particle's speed will be influenced by the velocity field as follows:

$$v = \int F_{\text{ext}} \, \mathrm{d}t + w_s + \alpha v_f + v_0, \tag{8}$$

with v_f medium velocity field modulated by $\alpha \in [0, 1]$.

Our particle system can model intricate scenarios, like 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.

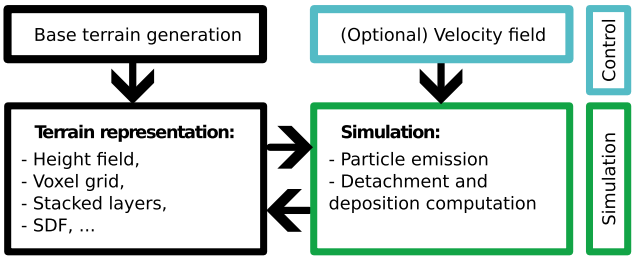


Fig. 5 Our overall pipeline: our erosion process compute matter displacement of a terrain using an arbitrary representation as long as intersections between particles and the ground can be detected. An optional velocity field, provided by the user, guides the particles trajectories. We propose surface alteration methods to apply the erosion to the terrain in a coherent way between possible representations

However, minor changes can be overlooked to maintain a balance between realism and computational efficiency [50]. We offer several velocity improvement methods:

-Fluid simulation refinement: Many erosion systems incorporate fluid simulation, requiring regular updates for erosion and velocity [25, 52]. 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 [41].

-Particle velocities in fluid simulation: With a Lagrangian fluid simulation relying on particle systems [24], our particle velocities can be incorporated in its computation. This approach is only a provisional solution due to potential parameter mismatches with 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 can 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 can simply interpolate the velocity field at any transitioning point. This simple method, as used in Fig. 9, allows us to find a balance between achieving realistic flow simulations and maintaining computational efficiency.

4 Our erosion method

In this section, we describe how to apply detachment and deposition to different terrain representations with our method (Fig. 5). We cover the most commonly used representations namely height fields, layered terrains, voxel grid 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: being able to evaluate the intersection of a particle with the ground and compute the normal of the terrain at this point. To the best of our knowledge, all representation do.

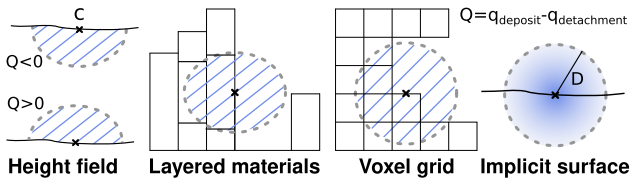


Fig. 6 Illustration of the material detachment in the (half-)sphere at contact point C (cross) on different representations. (height field) When $Q < 0$ material detachment happen in the bottom scaled half sphere of the particle's contact with the ground, while the deposition is applied on the upper half sphere of volume when $Q > 0$. Unlike the height field, for 3D terrains detachment and deposit are applied in the full sphere around the contact point

We use Verlet integration for the particle's physics [51], with low error rate and stability even for high dt, reducing computation time for negligible imprecision [2, 49].

For all the representations, the amount of material absorbed by the particle, i.e., the erosion value $q_{\text{detachment}}$ from (3), is taken around the particle at a radius R, meaning that the modification of the terrain by a particle at position c will only occur for the positions p satisfying $||p-c|| < R$. At the same time, the amount q_{deposit} from (4) 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 conceptualized 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 Fig. 6). In the following sections, we will describe the strategies used to modify the amount of matter for different representations.

4.1 Application on height fields

On a height field defined by $h(p) = z$, the intersection point with the surface is verified at $p_z = h(p)$, 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 can decrease the height $h'(p)$ at all points p by the height of the scaled half sphere at position p. Given the height of the scaled half sphere of center c and the distance of the particle to the center $d = ||p - c||$ by $h_{\text{halfsphere}}(p) = \alpha\sqrt{R^2 - d^2}$ for all 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

$$\Delta h(p) = \frac{\sqrt{R^2 - d^2}}{\alpha} = \frac{Q}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2} \tag{9}$$

The height at each point after an erosion is then computed as $\tilde{h}(p) = h(p) + \Delta h(p)$.

4.2 Application on layered terrains

Layered terrains are defined as $\mu : \mathbb{R}^3 \to \mathbb{N}$ assigning a discrete material index μ for any point in space [6, 36]. 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 [22], we approximate the volume of the stack we need to alter using the previously defined height field equation (9). At a distance d from the particle, the height is defined as:

$$H(d) = \frac{|Q|}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2}. \tag{10}$$

If $Q > 0$ (more deposition is applied that detachment), then we transform the materials in the stack contained in the sphere to become ground material. For $Q < 0$ the materials are transformed in background material.

4.3 Application on implicit terrains

Implicit terrain are defined using a function $f(p)$ and its variation resulting from the erosion process using $\Delta f(p)$. We propose a strategy to compute $\Delta f(p)$ 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:

$$\Delta f(p) = \frac{3Q}{\pi}\frac{(1-d)}{R} \tag{11}$$

with d the distance of the point p to the sphere center. For all point p for which $d \geq R$, $\Delta f(p) = 0$ (see A).

As they are the most commonly used representations, we propose a formulation to erode implicit terrains defined by signed distance functions (SDF) and by gradient or vector fields.

Signed distance functions Considering SDF, the terrain is defined as the 0-set of the signed distance function $f : \mathbb{R}^3 \to \mathbb{R}$, hence, for $f(p) = 0$, the inside as $f(p) < 0$ and outer-part (i.e. air or water) as $f(p) > 0$.

The particle erosion applies at impact points at discrete positions, so we propose to add or subtract metaballs defined

using Eq. (11) to, respectively, deposit or erode material using a composition tree:

$$metaball(p) = -\Delta f(p).$$

Now the eroded terrain function $\tilde{f}(p)$ will be evaluated at each point p from the initial terrain value $f(p)$, the erosion function $metaball(p)$ and the composition function $g(f_1, f_2)$:

$$\tilde{f}(p) = g(f(p), metaball(p)).$$

As a metaball is added for each particle bounce on the terrain space partitioning optimization algorithms such as k-d trees, BSP trees or BVH can easily be used to improve performances.

Other implicit terrains are present in the literature, notably a 2.5D representation based on the surface gradient [19] and a 3D representation based on curves [4] 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 (9) that gives:

$$\nabla h' = -\frac{Q}{\frac{2}{3}R^3}\frac{1}{\sqrt{R^2 - d^2}}\vec{CP} \tag{12}$$

with $\vec{CP}$ the vector from the position p to evaluate to the center of the erosion point c. Now the new gradient field can be computed as:

$$\nabla\tilde{h}(p) = \nabla h(p) + \nabla h'(p).$$

4.4 Application on voxel grids

We consider two of the voxel grids representations: density-voxel grids and binary voxel grids for which we present our material alternation strategy.

Density voxels. We consider "density-voxel" grids defined on $f : \mathbb{Z}^3 \rightarrow [-1, 1]$ for which a voxel is be full for $f(p) = 1$, partially full for $-1 < f(p) < 1$ or empty for $f(p) \leq -1$. This definition allows us to erode them smoothly. Since this kind of grid is a discretizaion of a scalar function, we could directly use (11), 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 c on all points p in the volume proportionally to the distance from the center of the sphere $d = ||p - c||$ to find an approximation to the real erosion value per voxel $q_{\text{approx}} = Q\frac{1-d}{R}$. 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 until -1. We then have for all surrounding voxels:

$$\Delta f(p) = Q\frac{(1 - \frac{d}{R})}{\sum(1 - \frac{d}{R})}. \tag{13}$$

Resulting voxel value is computed as $\tilde{f}(p) = f(p) + \Delta f(p)$. In our implementation, when $f(p) > 1$, we simply transport the density excess to the above voxel, giving it a very close analogy to height fields as long as $|\Delta f| < 1$.

Binary voxels The terrain can be represented using an occupancy function as $f : \mathbb{Z}^3 \rightarrow \{0, 1\}$ 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 transform them as air or as ground when this number reaches a critical value C that is proportional to the particle's strength parameter K_ε [3].

On a hit, all voxels in a radius R receive a hit number:

$$\Delta hits = \lfloor \alpha \Delta f \rfloor \tag{14}$$

with Δf the erosion per voxel computed using (13) and α a coefficient high enough to obtain values above 1.

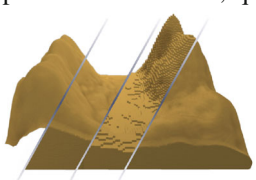
All voxels with $\#hits > C$ are transformed to background and voxels with $\#hits < -C$ are transformed to ground.

Note that, a binary voxel grid can also be transformed into a density-voxel grid to be eroded smoothly.

Our formulation for height fields (9) can be used to erode 2D scalar field-based representations. Similarly, our proposition for SDF (11) enables erosion for continuous 3D scalar fields and voxels (13) for discrete 3D scalar fields, respectively.

5 Results

Our erosion process enables the simulation of a wide range of erosion effects on the major terrain representations alike. In this section, we present applications that demonstrate the versatility of our method by changing the particle's effect size, quantity, density, maximum capacity, deposition factor and the velocity fields. The results of each process are presented in Fig. 14, parameters used are available at Table 1. 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 structure, such as overhangs, using 2.5D representations (height fields).



Environment density ρ_f is set to 1kg m^{-3} above water level (terrain blue part) and to 1000kg m^{-3} below it. Velocity

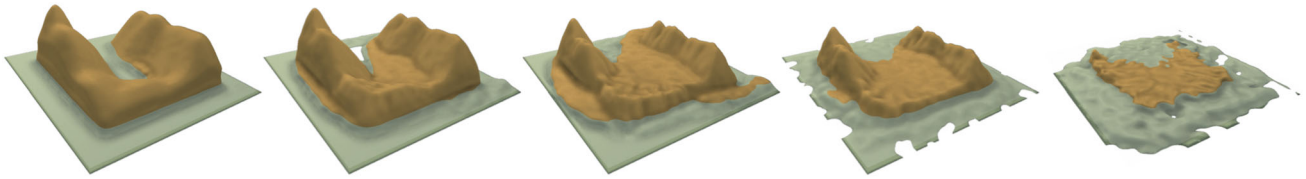


Fig. 7 Our erosion method is applied iteratively on a completely synthetic island, the terrain is altered to obtain a plausible shape by forming rills. The use of particles with hydraulic densities dropped from the sky results in a strong erosion on the sides of the mountains, and the particles that slide to the sea are mainly drifting offshore resulting in the formation of small beaches and a 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

Table 1 Parameters used for the generation of the terrains presented in Fig. 14, with "Rep" the representation (H: Heightmap, DV: Density-voxels, BV: Binary voxels, I: Implicit) "Res" the resolution in meter per voxel or cell, $\#P$ the number of particles per iteration, $\#N$ the number of iterations, R the particles radius (in voxel or cell unit), COR the coefficient of restitution, ρ_p the particle density in kg m^{-3}, C_{factor}, ε and ω 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

Name	Rep.	Dimensions	Res	$\#P$	$\#N$	R	COR	ρ_p	C_{factor}	ε	ω	Vel field	t
Rain	H	100×100	20	100	10	1.0	1.0	1000	10.0	2.5	0.3	None	4.0
Coastal	DV	$100 \times 100 \times 30$	10	80	3	5	0.1	500	10.0	5.0	0.5	Uniform	0.5
Meanders	I	N/A	N/A	10	20	5.0	1.0	1000	1.0	1.0	1.0	[a]	1
River	H	100×100	5	100	50	1.5–5	0.5	900	0.1	1.0	1.0	None	2.5
Landslide	H	100×100	20	200	10	2.5	0.2	500	0.1	1.0	1.0	None	4
Volcano	DV	$100 \times 100 \times 40$	50	150	30	1.0	5.0	2000	1.0	1.0	5.0	None	0.8
Karst	BV	$100 \times 100 \times 50$	2	1000	40	5	0.5	500	10.0	5.0	0.5	Uniform	20
Tunnel	DV	$100 \times 100 \times 50$	1	100	100	2.5	0.1	500	1.0	1.0	1.0	None	0.8
Wind	DV	$100 \times 100 \times 50$	0.2	100	10	1.5	0.9	1.5	1.0	1.0	1.0	[33]	0.5
Underwater	H	100×100	10	100	50	2.5	0.9	1000	1.0	1.0	1.0	[47]	4

[a] The velocity field is a vector field defined as $v_f(p) = [0\ sin(p.x)\ 0]^T$

field's refinement is done by using the presented diffusion strategy.

Rain. 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 the gravitational force of Earth. No velocity field is required from fluid simulation. These parameters result in a detailed geometry of the rills on the side of mountains that quickly emerge and deposit many sediments in the valley. We demonstrate the result of rain erosion in Fig. 14: Rain with a computation time of 4 s.

Using this erosion parameters in combination with water bodies results in different outcomes (Fig. 7). 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 sediments, leading to mainly apply deposition. The result is a typical hydraulic erosion on mountains and the formation of slopes and beaches near water level.

Coastal erosion. Waves repeated motion creates coastal erosion, that can be seen as cliffs with holes at the water level.

We apply a uniform velocity field in the water pointing toward 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 ω. Using these parameters, the erosion process is focused at the interface of air and water, and apply a coarse detachment while depositing a very small quantity of sediments, simulating the corrosive effect of water on limestone.

This effect can only be simulated on 3D terrain representations, but will create cliffs on a 2D representation. Figure 14: Coastal presents the result of coastal erosion on a density-voxel grid that creates overhangs around sea level using a small amount of particles. Note that, the same effect using an alternate implicit representation based on SDF is displayed in Fig. 10. A shaded version of this effect is presented in Fig. 1.

Rivers. Given a source point, we generate particles that run downhill, simulating the formation of a river. More complex erosion simulation using fluid simulations like SPH [25] 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, used 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 can reproduce simple river meanders (Fig. 14: Meanders).

Figure 14: River presents a river that has been modeled by emitting water particles with different sizes that ranges from 1.5 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 had fluctuated over formation time, while the low capacity ensure that the banks of the river stays smooth. A high coefficient of restitution is a strategy that let the particles flow with low friction, approaching a water behavior. Our particles are affected only by gravity, without fluid simulation.

Landslide are mainly caused by large amount 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 and a low capacity factor but a high deposition factor ω, they transport sediments on short distances as the velocity quickly drops to 0, and ground material is completely spread along its path since it is easier to deposit the same amount of sediment than the eroded amount at each collision point. Reducing the density of the particle simulates a rise of viscosity in the settling velocity formula, increasing again the quantity of matter to deposit at contact with the ground. By this means, we can simulate landslides as illustrated on Fig. 14: Landslide. A smoother surface is resulting, compared to the rain erosion as the rills are filled with sediments 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 volcano eruption as illustrated on Fig. 14: Volcano.

Karsts 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 [34].

By reducing the deposition factor ω, the particles simulate corrosion (without mass conservation). We can use the same particle parameters than the coastal erosion (big size, a density between air and water densities, a high capacity factor and a low ω) and optionally provide a 3D shear stress map. The karst will automatically follow the softest materials, which is geologically coherent as given in example in Fig. 14: Karst, where we can observe a "pillar" that is formed in the center, 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 Fig. 14: Tunnel in which a water runoff is eroding a tunnel without the use of a fluid simulation. Here, when particles bounce often on the terrain surface, the coefficient of restitution may be seen as a viscosity parameter.

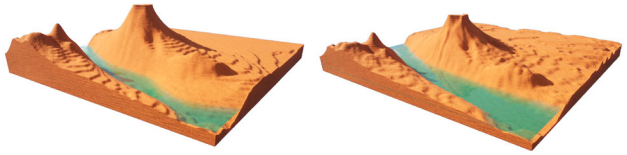


Fig. 8 Multiple erosion types can be combined. On an initial synthetic $500 \times 500 \times 50$ density voxel grid, the a wind erosion is applied on the surface of the terrain while hydraulic erosion shapes the rills and the base of the mountains. A water current digs its borders and spreads sediments at the bottom

Wind erosion is a significant process in desertscapes shaping since there are no obstacles on the airflow path. Air particles can reach high velocities, transporting sand over long distances forming either dunes or are blasted into rocks, eroding into goblins.

By setting the density of our particles close to 1kg m^{-3}, two erosion simulations can be applied at once. Air particles follow closely the flowfield given by the user in air. This flowfield can be given from a complex simulation, a user-defined wind rose [35] or a random flowfield with a general direction.

The generation of the 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 looking like gobelins.

Figure 14: Wind 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 [35] is used on light particles, the simulation is fast thanks to the low number of collisions each particle has with the ground.

Multiple phenomena A terrain eroded with multiple erosion phenomena applied on a $500 \times 500 \times 50$ density-voxel grid is illustrated in Fig. 8. Here, water-density particles are applying rain on the terrain while the coasts of the river are being eroded thanks to a velocity field defined at the water level. The velocity field defined in the air mainly affects particles with air-density, such that wind erosion can be applied at the same time. The computation of these effects took 7s on CPU.

Underwater currents Procedural generation of underwater 3D terrains has received little attention. The difference between the underwater and the surface rely on the buoyancy force that is much stronger, meaning that the water flow has a much more impacting effect on erosion than wind. Taking into account the density of the environment and the velocity field of water in our formulas are the keys to be able to apply any erosion in this environment. Our method works in a water environment by giving at least water density to

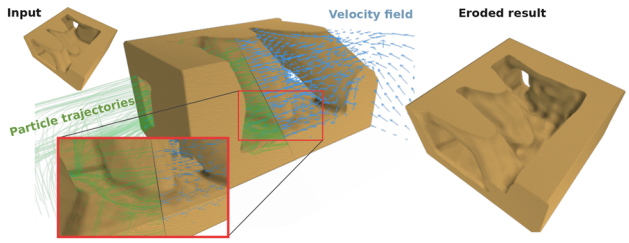


Fig. 9 A complex water flow simulation is computed using OpenFOAM. Particle trajectories (green) are highly affected by the fluid velocity (blue). Most the terrain exposed surfaces is eroded (bottom)

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 Fig. 14: Underwater, the velocity field is given by a simple 3D fluid simulation [46] applied on the terrain.

A complex water flow simulation is computed using SIMPLE [7] 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) is highly affected by the fluid velocity (blue). The density 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_f and settling velocity w_s from (8) (Fig. 9).

6 Comparisons

In the following section, we compare our method with existing ones to show that while we are versatile on the 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.

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 primitive in the terrain's construction tree [33]. The positions of the spheres are determined using a Poisson disk sampling at the weakest terrain area defined by the Geology tree of their model. They are simulating the corrosion effect of water on the rocks. Our work is also able to approximate this phenomena by defining the position of these sphere primitives at the position where the water particles hit the surface. While the computation time of the positions of the sphere is higher due to the fact that we are evaluating the position of our particles at every time step in the implicit model (which could be improved by the triangulation of the implicit surface, or better, a dynamic triangulation), the distribution of our erosion primitives is

Fig. 10 The algorithm proposed by Paris et al [33] allows for the simulation of coastal erosion (left) that we can reproduce almost identically by allowing our particles to collide only once with the ground and applying only erosion (center). If we apply our erosion with the full tracking of our particles and using deposition, we can achieve more diverse results (right)

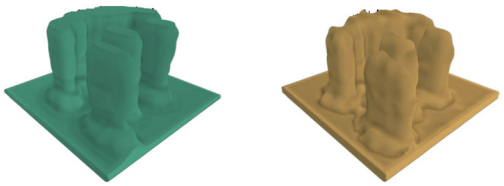


Fig. 11 The algorithm proposed by Jones et al [3] allows for an efficient simulation of the spheroidal erosion, making the creation of gobelins on voxel grids in a plausible way (left). Our algorithm naturally erodes the most exposed areas of the terrain when particles are affected by the wind (right)

based on a physical model instead of a mathematical model, meaning that we can integrate more easily the direction and strength of the 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 is not conserving the mass of the terrain, which is acceptable for the corrosion simulation, but limits its validity for other erosion simulations. In our method, the particle can be tracked until it settles, ensuring mass conservation (Fig. 10);

Wind erosion on voxel grid representation: Jones et al propose a weathering erosion on voxel grids by approximating and eroding continuously the most exposed voxels [3]. When a solid voxel is decimated, it is considered deposit and is displaced down the slope until a minimal talus angle in the terrain is reached and 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 the erosion process as much as the deposition process since the air particles, filled with sediments, is falling automatically toward the local minimum of the erosion point. Just like in their work, we can easily define the resistance value of the 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 gobelin surrounded by bigger gobelins, and also allows the deposit slope to fit more closely to the wind direction (Fig. 11).

Hydraulic erosion on height field representation: Mei et al integrate and adapt to the GPU the pipe model proposed

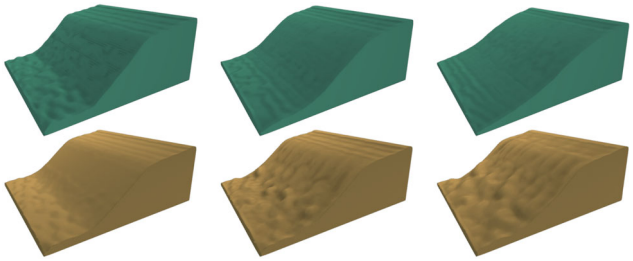


Fig. 12 While our resulting geometry on the hydraulic erosion (bottom) is less smoothed than the one proposed by Mei et al. [27] (top), our method allows the application on more terrain representations than the height fields only

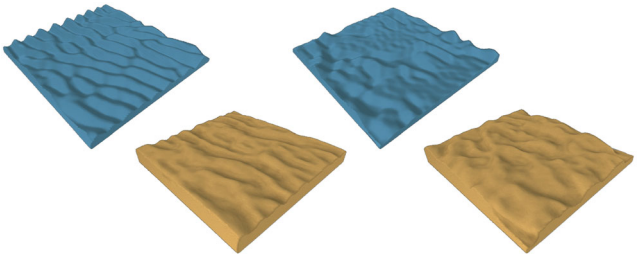


Fig. 13 The algorithm from Paris 2020 allow the generation of desertscapes (top), which we can (at least partially) reproduce with our erosion simulation (bottom). The different effects are achieved by affecting the wind direction and strength

in [30] for the fluid simulation [27]. This simulation is simple but efficient enough to approximate the Shallow-Water equations in real time and use 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 rely on a stable fluid flow that is consistent during the whole life time of a particle, but by refining the simulation at each time step instead of at the end of the particles lifetime, our erosion model is able to reproduce this effect, allowing the terrain to have a single batch of fluid going through it. Our method can be seen as a generalization of Mei at al. that can then be used on more than discrete 2D grids (Fig. 12).

Wind erosion on stacked materials representation: Paris et al [35] simulate the effect of wind over sand fields defined on stacked materials, creating dune structures, even taking into account obstacles like [40] and different material layers like vegetation [11] that are not affected by abrasion [35]. A wind field simulation is required to produce results, and while [40] and [32] consider a uniform vector field, this work consider a dynamic vector multi-scaled warped field from the 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 Fig. 13.

7 Discussion

This work is a generalization of erosion that is applicable to any terrain representation. In practice, while similar particle physics is used on different terrain representations, using similar parameters does not ensure resulting in the same eroded terrain. Surfaces and normals being approximated differently have rippling effect on particle trajectories. Note that, not all effects can be applied to all representations, for instance, karsts generation on 2.5D data structures.

Realism 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 will result in a terrain that is unrealistic since the alteration will have localized 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 a infinite resolution.

Our method allows to perform 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 in the composition tree. Using skeletons-defined primitives [20, 39] from particles trajectories and erosion/deposition values could be a solution to optimize the computation time.

Usage of velocity fields In our erosion algorithm, we simplify particle physics to enhance computational efficiency and facilitate parameterization. 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 meeting the applications 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 Sect. 3.3. However, it is important to note that these are expedient solutions and may not fully capture precise dynamics of an evolving terrain.

Performances To facilitate parallelization, 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 particles and ground properties (Young's modulus, friction, material,...) [53], further easing the parameterization process. However, these simplifications, combined with the inherent discrete nature

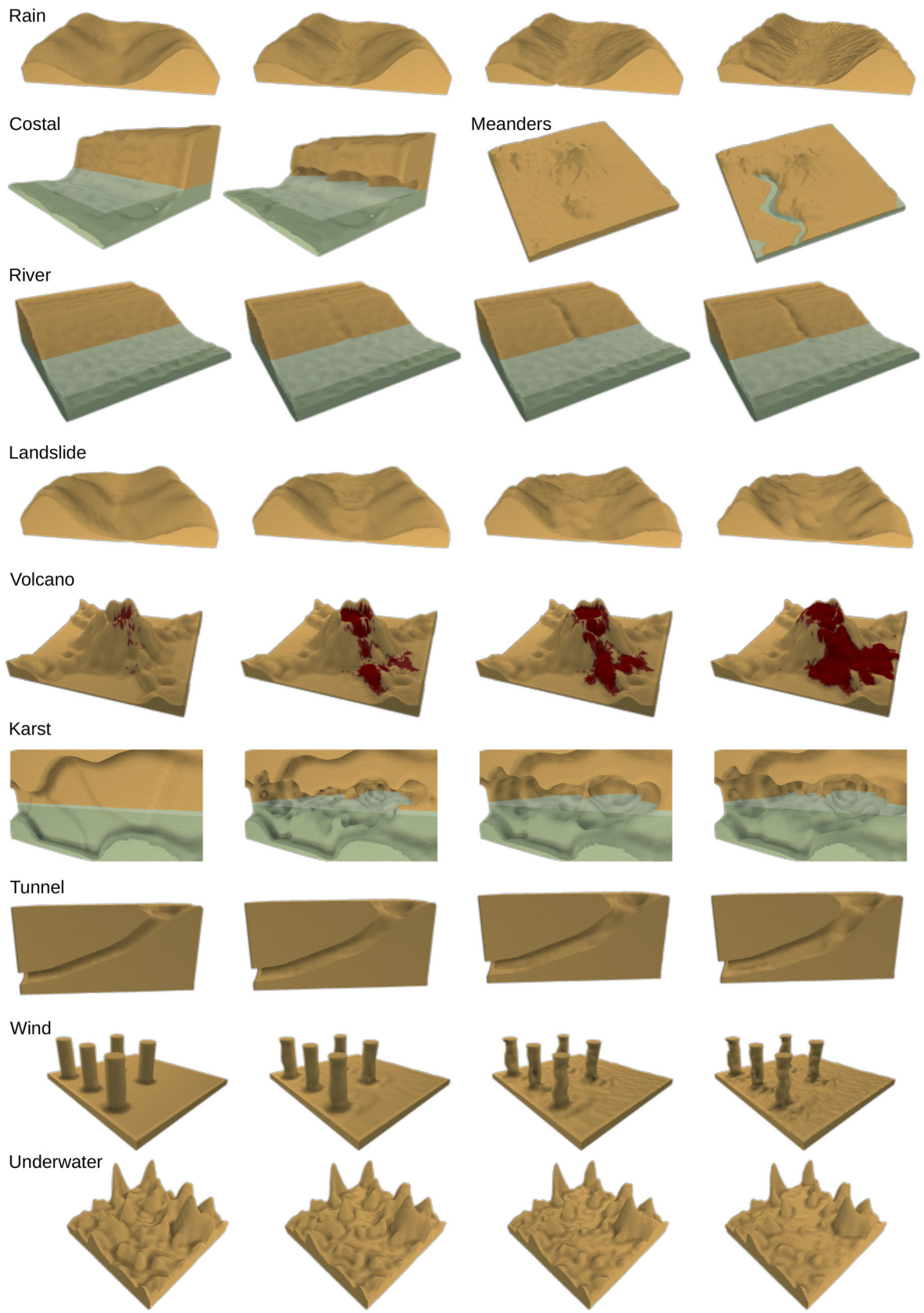


Fig. 14 Erosion processes results on various representations presented in Sect. 5. Used parameters used are detailed in Table 1

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 performances 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 [37]. Additionally, we acknowledge an issue where particles may collide with the ceiling and the deposition is stuck. While a potential resolution involves splitting particles upon impact rather than simply depositing sediments, this introduces complexities to the parallelization layer of the method. Allowing particles to split introduces unpredictability in the total number of particles that will exist in the simulation. This unpredictability can complicate the use of multi-threading. Future works include 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 way our method is done, we do not consider the transport of different materials (all sediments are 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 define transformation rules from one material to another when a voxel is eroded a number $\#hits < -C$ or $\#hits > C$. For example, the material "clay" may transform to "sand" when eroded or to "rock" when many depositions occurred.

8 Conclusion

We introduced a flexible particle-based erosion system that is easy to use and simple to implement. We have presented how to adapt the process for various terrain representations and generate a variety of erosion phenomenon due to rain, wind, water bodies... by adjusting intuitive parameters hence generate automatically realistic 2.5D and 3D terrains. The use of external velocity fields provides a high flexibility, i.e. using the simulations that best fits the user's needs (precision, control, implementation efficiency...). Our method can also be applied to underwater environments with identical physics simulation since our erosion method can be applied on 3D representations. Erosion algorithms are often limited to the use of height fields, but by finding more generalized methods, we can go toward a global use of 3D terrains, which can offer richer and more diverse landscapes.

9 Revision

We have addressed the reviewers concerns in red namely a typography in section 5.

A Computation of a metaball

We use the following formula to evaluate a metaball in space with a center c and of radius R:

$$g(p) = 1 - \frac{||p - c||}{R}$$

using the Euclidean distance.

We have a total amount Q to define in this space, so the final metaball function f needs to satisfy Eqs. (15) and (16):

$$f(p) = \lambda g(p) \tag{15}$$

$$\int_{p \in V_{3D}} f \, dp = Q \tag{16}$$

First, let's exploit the radial symmetry of the metaball and rewrite $g(p) = 1 - r$ by using the polar coordinates of the point $p - c$.

We can then integrate g over the volume V_{3D} as

$$\begin{aligned}
&\int_0^1 \int_0^\pi \int_0^{2\pi} g(r) r^2 \sin(\theta) \, \mathrm{d}r \, \mathbf{d}\theta \, \mathrm{d}\phi \\
&= \int_0^1 \int_0^\pi \int_0^{2\pi} (1 - r) r^2 \sin(\theta) \, \mathrm{d}r \, \mathrm{d}\theta \, \mathrm{d}\phi \\
&= \int_0^1 (1 - r) r^2 \, \mathrm{d}r \times \int_0^\pi \sin\theta \, \mathrm{d}\theta \times \int_0^{2\pi} 1 \, \mathrm{d}\phi
\end{aligned}$$

We then break down the integrals one by one such as

$$\begin{aligned}
&\int_0^1 (1 - r) r^2 \, \mathrm{d}r = \frac{1}{12} \\
&\int_0^\pi \sin\theta \, \mathrm{d}\theta = 2 \\
&\int_0^{2\pi} 1 \, \mathrm{d}\phi = 2\pi
\end{aligned}$$

By combining all these integrals, we get $\int g = \frac{1}{12} \times 2 \times 2\pi = \frac{\pi}{3}$.

So given $\int f = q_{\text{detachment}}$ and $\int f = \lambda \int g$, we can deduce that $\lambda = \frac{Q}{\int g} = \frac{3}{\pi} Q$.

From (15) we finally get

$$f(p) = \frac{3Q}{\pi}\left(1 - \frac{||p-c||}{R}\right) \tag{17}$$

, representing the rate of change on the evaluation function of the terrain surface.

The integration in the voxel space is out of the scope of this paper and a numerical solution is instead proposed in Sect. 4.4.

References

1. Argudo, O., Galin, E., Peytavie, A., Paris, A., Guérin, E.: Simulation, modeling and authoring of glaciers. ACM Trans. Gr. **39**, 1–14 (2020). https://doi.org/10.1145/3414685.3417855
2. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, pp. 43–54 (1998). https://doi.org/10.1145/280814.280821
3. Beardall, M., Butler, J., Farley, M., Jones, M.D.: Directable weathering of concave rock using curvature estimation. IEEE Trans. Vis. Comput. Gr. **16**(1), 81–94 (2010). https://doi.org/10.1109/TVCG.2009.39
4. Becher, M., Krone, M., Reina, G., Ertl, T.: Feature-based volumetric terrain generation. In: Proceedings - I3D 2017: 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (2017). https://doi.org/10.1145/3023368.3023383
5. Beneš, B., Těšínský, V., Hornyš, J., Bhatia, S.K.: Hydraulic erosion. Comput. Anim. Virtual Worlds **17**(2), 99–108 (2006). https://doi.org/10.1002/cav.77
6. Beneš, B., Forsbach, R.: Layered data representation for visual simulation of terrain erosion. In: Proceedings - Spring Conference on Computer Graphics, SCCG 2001, pp. 80–86 (2001). https://doi.org/10.1109/SCCG.2001.945341
7. Caretto, L.S., Gosman, A.D., Patankar, S.V., Spalding, D.B.: Two calculation procedures for steady, three-dimensional flows with recirculation. In: Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, vol. 19, pp. 60–68 (1973)
8. Cordonnier, G., Braun, J., Cani, M.P., Beneš, B., Peytavie, A., Guérin, É.: Large scale terrain generation from tectonic uplift and fluvial erosion. IEEE Trans. Vis. Comput. Graph. (2017). https://doi.org/10.1109/TVCG.2017.2689022
9. Cordonnier, G., Cani, M.P., Beneš, B., Braun, J., Galin, É.: Sculpting mountains: interactive terrain modeling based on subsurface geology. IEEE Trans. Vis. Comput. Graph. (2018). https://doi.org/10.1109/TVCG.2017.2689022
10. Cordonnier, G., Ecormier-nocca, P., Galin, É., Gain, J., Beneš, B., Cani, M.P.: Interactive generation of time-evolving, snow-covered landscapes with avalanches. Comput. Graph. Forum **37**(2), 497–509 (2018). https://doi.org/10.1111/cgf.13379
11. Cordonnier, G., Galin, É., Gain, J., Beneš, B., Guérin, É., Peytavie, A., Cani, M.P.: Authoring landscapes by combining ecosystem and terrain erosion simulation. ACM Trans. Graph. (2017). https://doi.org/10.1145/3072959.3073667
12. Cordonnier, G., Jouvet, G., Peytavie, A., Braun, J., Cani, M.P., Benes, B., Galin, E., Guérin, E., Gain, J.: Forming terrains by glacial erosion. ACM Trans. Graph. **42**, 14 (2023). https://doi.org/10.1145/3592422
13. Dey, R., Doig, J.G., Gatzidis, C.: Procedural feature generation for volumetric terrains using voxel grammars. Entertain. Comput. **27**, 128–136 (2018). https://doi.org/10.1016/j.entcom.2018.04.003
14. Eisemann, E., Decoret, X.: Single-pass gpu solid voxelization for real-time applications, pp. 73–80 (2008)
15. Emilien, A., Poulin, P., Cani, M.P., Vimont, U.: Interactive procedural modelling of coherent waterfall scenes. Comput. Graph. Forum **34**, 22–35 (2015). https://doi.org/10.1111/cgf.12515
16. Gain, J., Marais, P., Straßer, W.: Terrain sketching. In: Proceedings of I3D 2009: the 2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, vol. 1(212), pp. 31–38 (2009). https://doi.org/10.1145/1507149.1507155
17. Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.P., Benes, B., Gain, J.: A review of digital terrain modeling. Comput. Graph. Forum **38**, 553–577 (2019). https://doi.org/10.1111/cgf.13657
18. Guérin, É., Digne, J., Galin, É., Peytavie, A.: Sparse representation of terrains for procedural modeling. Comput. Graph. Forum **35**(2), 177–187 (2016). https://doi.org/10.1111/cgf.12821
19. Guérin, E., Peytavie, A., Masnou, S., Digne, J., Sauvage, B., Gain, J., Galin, E.: Gradient terrain authoring. Comput. Graph. Forum **41**, 85–95 (2022). https://doi.org/10.1111/cgf.14460
20. Hong, Q.: A skeleton-based technique for modelling implicit surfaces. In: Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013, vol. 2(Cisp), pp. 686–691 (2013). https://doi.org/10.1109/CISP.2013.6745253
21. Ito, T., Fujimoto, T., Muraoka, K., Chiba, N.: Modeling rocky scenery taking into account joints. In: Proceedings of Computer Graphics International Conference, CGI 2003-Janua(July 2014), pp. 244–247 (2003). https://doi.org/10.1109/CGI.2003.1214475
22. Jones, B.D., Williams, J.R.: Fast computation of accurate sphere-cube intersection volume. Eng. Comput. **34**, 1204–1216 (2017). https://doi.org/10.1108/EC-02-2016-0052
23. Kaufman, A., Cohen, D Yagel, R,: Volume graphics. Computer **26**(7), 51–64 (1993). https://doi.org/10.1109/MC.1993.274942
24. Koschier, D., Bender, J., Solenthaler, B., Teschner, M.: A survey on SPH methods in computer graphics. Comput. Graph. Forum **41**(2), 737–760 (2022). https://doi.org/10.1111/cgf.14508
25. Krištof, P., Beneš, B., Křivánek, J., Šťava, O.: Hydraulic erosion using smoothed particle hydrodynamics. Comput. Graph. Forum **28**(2), 219–228 (2009). https://doi.org/10.1111/j.1467-8659.2009.01361.x
26. Lengyel, E.: Voxel-based terrain for real-time virtual simulations, p. 148 (2010)
27. Mei, X., Decaudin, P., Hu, B.G.: Fast hydraulic erosion simulation and visualization on GPU. In: Proceedings - Pacific Conference on Computer Graphics and Applications, pp. 47–56 (2007). https://doi.org/10.1109/PG.2007.27
28. Musgrave, F.K., Kolb, C.E., Mace, R.S.: The synthesis and rendering of eroded fractal terrains. In: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989, pp. 41–50 (1989). https://doi.org/10.1145/74333.74337
29. Neidhold, B., Wacker, M., Deussen, O.: Interactive physically based fluid and erosion simulation. Natural Phenomena, pp. 25–32 (2005)

30. O'Brien, J.F., Hodgins, J.K.: Dynamic simulation of splashing fluids. In: Proceedings Computer Animation, CA, pp. 198–205 (1995). https://doi.org/10.1109/CA.1995.393532
31. Olsen, J.: Realtime procedural terrain generation. Department of Mathematics And Computer Science (...) p. 20 (2004)
32. Onoue, K., Nishita, T.: A method for modeling and rendering dunes with wind-ripples. In: Proceedings - Pacific Conference on Computer Graphics and Applications **2000-January**, pp. 427–428 (2000). https://doi.org/10.1109/PCCGA.2000.883978
33. Paris, A., Galin, E., Peytavie, A., Guérin, E., Gain, J.: Terrain amplification with implicit 3d features. ACM Trans. Graph. **38**, 1–15 (2019). https://doi.org/10.1145/3342765
34. Paris, A., Guérin, E., Peytavie, A., Collon, P., Galin, E.: Synthesizing geologically coherent cave networks. Comput. Graph. Forum **40**, 277–287 (2021). https://doi.org/10.1111/cgf.14420
35. Paris, A., Peytavie, A., Guérin, E., Argudo, O., Galin, E.: Desertscape simulation. Comput. Graph. Forum **38**, 47–55 (2019). https://doi.org/10.1111/cgf.13815
36. Peytavie, A., Galin, E., Grosjean, J., Merillou, S.: Arches: a framework for modeling complex terrains. Comput. Graph. Forum **28**, 457–467 (2009). https://doi.org/10.1111/j.1467-8659.2009.01385.x
37. Ranz, W.E., Talandis, G.R., Gutterman, B.: Mechanics of particle bounce. I.Ch.E. J **6**, 124–127 (1960)
38. Richardson, J.F., Zaki, W.N.: The sedimentation of a suspension of uniform spheres under conditions of viscous flow. Chem. Eng. Sci. **3** (1954)
39. Rigaudière, D., Gesquière, G., Faudot, D.: Shape Modelling with Skeleton based Implicit Primitives. Methods (2000)
40. Roa, T., Benes, B.: Simulating desert scenery. Winter School of Computer Graphics SHORT communication Papers. In: Proceedings, pp. 17–22 (2004)
41. Roose, D., Leuven, K.U., López, Y.R.: Dynamic refinement for fluid flow simulations with sph particle refinement for fluid flow simulations with sph (2011)
42. Roudier, P., Peroche, B., Perrin, M.: Landscapes Synthesis Achieved through Erosion and Deposition Process Simulation. Computer Graphics Forum **12**(3), 375–383 (1993). https://doi.org/10.1111/1467-8659.1230375
43. Schott, H., Paris, A., Fournier, L., Guérin, E., Galin, E.: Large-scale terrain authoring through interactive erosion simulation (2023)
44. Smelik, R.M., Kraker, K.J.D., Groenewegen, S.A., Tutenel, T., Bidarra, R.: A survey of procedural methods for terrain modelling. Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS) (2009)
45. Stachniak, S., Stuerzlinger, W.: An algorithm for automated fractal terrain deformation. In Proceedings of Computer Graphics and Artificial Intelligence pp. 64–76 (2005)
46. Stam, J.: Stable fluids. Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999 pp. 121–128 (1999). https://doi.org/10.1145/311535.311548
47. Stam, J.: Flows on surfaces of arbitrary topology. ACM Transactions on Graphics **22**(3), 724–731 (2003). https://doi.org/10.1145/882262.882338
48. Stokes, G.G.: On the Effect of the Internal Friction of Fluids on the Motion of Pendulums, pp. 1–10. Cambridge University Press (2009). https://doi.org/10.1017/CBO9780511702266.002
49. Swope, W.C., Andersen, H.C., Berens, P.H., Wilson, K.R.: A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. J. Chem. Phys. **76**(1), 637–649 (1982). https://doi.org/10.1063/1.442716
50. Tychonievich, L.A., Jones, M.D.: Delaunay deformable mesh for the weathering and erosion of 3D terrain. Visual Computer **26**(12), 1485–1495 (2010). https://doi.org/10.1007/s00371-010-0506-2
51. Verlet, L.: Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. Phys. Rev. **159**, 98–103 (1967). https://doi.org/10.1103/PhysRev.159.98
52. Wojtan, C., Carlson, M., Mucha, P.J., Turk, G.: Animating corrosion and erosion. Natural Phenomena pp. 15–22 (2007)
53. Yan, P., Zhang, J., Kong, X., Fang, Q.: Numerical simulation of rockfall trajectory with consideration of arbitrary shapes of falling rocks and terrain. Computers and Geotechnics **122** (2020). https://doi.org/10.1016/j.compgeo.2020.103511

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com