Shape morphing 2D

Introduction

Dans ce document je vais répertorier quelques algorithmes que j'ai essayé dans le but d'interpoler une forme dans une autre en conservant la topologie initiale. L'interpolation doit conserver un aspect organique qu'un humain imaginerait intuitivement.

Dans notre génération de réseaux karstiques, nous souhaitons pouvoir définir un profil d'une tranche d'un embranchement, puis de définir le profil à la sortie de la branche. La transition du début jusqu'à la fin de la branche doit rester cohérente. Si une perte de topologie se crée (i.e. aucune nouvelle intersection se créée entre deux segments non-consécutifs), les conséquences dans la modélisation du réseau peuvent créer des incohérences comme des cul-de-sac, inversions de normales, etc…

Restrictions

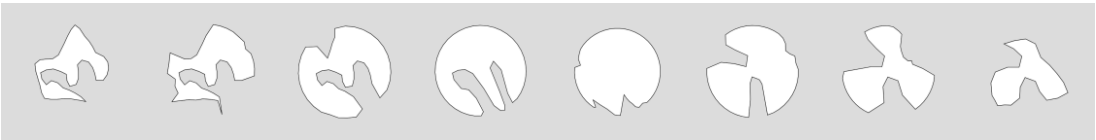
Dans notre cas, nous nous concentrons sur la géométrie discrète en deux dimensions. Cela signifie que nous travaillons sur des polygones en deux dimensions définies par des ensembles de points discrets.

De plus, nous supposerons les formes à interpoler étant définies comme « simplement connexe », c'est-à-dire n'ayant aucun segment sécant. Par conséquence, les formes durant l'interpolation doivent éviter au maximum de créer de nouveaux croisements.

L'algorithme final semble permettre d'interpoler une forme composée d'un unique croisement vers une forme de même topologie, en conservant cette topologie tout au long de l'interpolation. L'algorithme échoue pour des polygones ayant plus de 1 croisement.

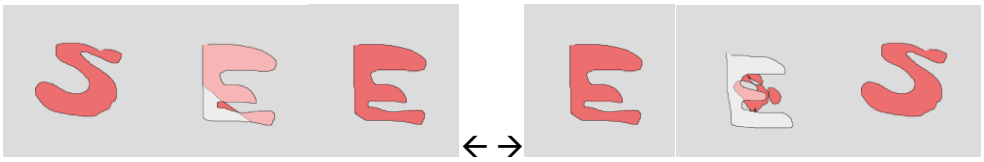
Résultats

Ci-dessous je présente 3 algorithmes essayés. Seul le dernier (« projection-répartition ») offre des résultats qui conservent complètement la topologie des formes durant l'interpolation, mais je suppose que la raison principale est qu'il intègre l'algorithme « rotation » dans sa conception. Néanmoins, cet algorithme impose une forme intermédiaire circulaire, qui peut ne pas être désirable en réalité.



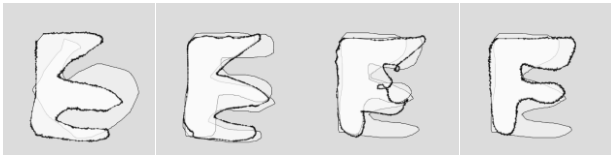
Interpolation entre une forme initiale (gauche) vers une forme finale (droite) en utilisant l'algorithme de « projection-répartition ». On remarque une forme intermédiaire proche du disque à la moitié de l'interpolation.

L'algorithme « rotation » propose une interpolation linéaire point-à-point. Dans la pratique, pour des formes plutôt simples, on obtient probablement les meilleurs résultats. Il semblerait aussi que l'algorithme ne soit pas vraiment sensible au nombre de segment sécants tant que la source et la destination en partagent le même nombre.



Interpolation d'une forme initiale (gauche) vers une forme finale (milieu) en utilisant l'algorithme de « rotation ». Bien que relativement petit, on remarque la création de croisements durant l'interpolation. Sur des formes très similaires, le résultat peut néanmoins être désastreux.

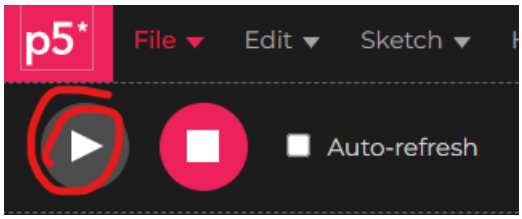
L'algorithme de système de particules offre une interpolation très organique par sa mécanique de gestion de forces multi-agents.



Interpolation d'une forme initiale (gauche) vers une forme finale (droite) en utilisant un système de particules. Des petits croisements peuvent apparaitre, mais j'estime que cela tient du fait que certains aspects de l'algorithme n'ont pas été codés complètement.

Afin d'appliquer des transformations sur chaque vertex source vers chaque vertex destination, on applique des subdivisions sur les formes jusqu'à atteindre le même nombre de points dans chaque polygone. Cette subdivision aléatoire peut causer des distributions non-uniformes le long du polygone, et possiblement des conséquences dans la recherche de plus courtes distances d'une forme à l'autre. J'aurais pu supposer que la subdivision soit faite de manière à diviser en priorité les segments les plus longs. De plus, on pourrait simplifier chaque forme par l'application d'un algorithme de simplification de polyligne (comme Ramer-Douglas-Peucker ou encore Visvalingam–Whyatt).

Une version web interactive a été codée ici :
https://editor.p5js.org/kemar74/collections/STtRw-Kbx
Pour les utiliser, accedez au lien, cliquez sur l'un des 3 algorithmes, puis lancer l'application en appuyant sur Play en haut à gauche.



Liste des algorithmes :

Système de particules :

Description :

- Chaque vertex est considéré comme une particule attachée par des ressorts à ses voisins.
- Pour changer de forme, chaque particule devient attirée par sa nouvelle position, tout en gardant une force attractive vers ses voisins directs.

Objectif :

La transition d'une forme à l'autre se fait de manière plus "organique" grâce aux contraintes imposées à chaque particule (ressorts). Avec une contrainte de torsion entre les voisins, on peut espérer voir disparaitre les "mini-boucles", ainsi que voir les vertex avoir une trajectoire non-linéaire qui peuvent éviter les croisements.

Réalité :

- Je n'ai pas recodé le système de ressorts, mais seulement une attraction avec les voisins. Je n'ai pas minimisé les distances à parcourir pour chaque vertex, donc on voit de gros croisements si les formes source et destination sont définies en sens contraire.
- On pourrait donc espérer voir certaines « anomalies » disparaitre en appliquant une rotation des indices de vertex comme décrit dans l'algorithme « rotation ».

Algorithme de rotation :

Description :

- Chaque vertex dans la forme source et la forme destination se voit assigné un indice de 0 à N.
- On calcule la distance à parcourir pour chaque vertex pour aller de la position i de la forme source à la position i de la forme destination.
- On répète l'étape 1 et 2 en incrémentant de 1 les indices de la forme source, sans toucher à la forme destination. On répète aussi l'opération en inversant l'ordre des indices dans la forme source.
- À partir de la configuration qui minimise la distance à parcourir pour chaque vertex, on peut interpoler linéairement la forme source vers la forme destination.

Objectif :

En déterminant une distance minimale, on espère trouver le meilleur matching entre les deux formes. Pour deux formes identiques simplement translatées et/ou pivotées, on est certain d'obtenir une interpolation sans boucle (preuve nécessaire). En tous cas, cela permet d'obtenir une configuration de base optimale pour l'interpolation (preuve nécessaire).

Réalité :

- La minimisation des distances peut être faite en MAE ou MSE ou en utilisant n'importe quelle métrique de distance, mais je ne sais pas ce qui est le plus efficace.
- Pour des formes complexes, beaucoup de croisements "inutiles" peuvent apparaitre, dûe à l'interpolation linéaire appliquée sur une rotation. L'utilisation de coordonnées polaires peuvent possiblement résoudre ce problème.
- Pour des formes simples (ex : carré sur carré translaté), on peut avoir une impression de rotation inutile des vertex, mais c'est possiblement juste visuel.
- La densité et la répartition des vertex joue un rôle dans le résultat des interpolations.

Algorithme de « projection-répartition » :

Description :

- L'algorithme cherche à interpoler premièrement une forme source vers un cercle avant d'être de nouveau interpolée vers la forme destination.
- À partir de la forme source, on calcule tous les points capables d'être déplacés directement au bord du cercle sans causer de croisement avec les autres segments. Plusieurs "bords" sont essayés : le point le plus proche sur le cercle, ainsi que le point central des "trous" du cercle (un trou étant défini par l'angle entre deux vertex plaqués sur le cercle mais non-liés [i.e. dont les indices ne sont pas successifs). Pour cela on vérifie que le triangle ABC défini par le point précédent, le point suivant et le bord de cercle ne contient aucun point du polygone (excepté le vertex que l'on cherche à déplacer). C'est l'étape de « projection ».

- Les points du bord de cercle sont ensuite répartis uniformément tout le long du cercle "occupé". C'est l'étape de « répartition ».
- Les étapes de projection sur le cercle et répartition des points "fixés" est répétée jusqu'à ce que tous les points soient fixés et uniformément répartis.
- On applique le même algorithme entre la forme destination et un cercle. On ajoute une rotation des indices du cercle final pour que les indices du cercle "source" et le cercle "destination" correspondent aux coordonnées les plus proches (voir algorithme "de rotation").
- On applique une interpolation de la forme source vers le cercle "source", puis du cercle "source" vers le cercle "destination", puis enfin du cercle "destination" vers la forme destination.
- Afin de ne pas voir la forme "cercle" de transition, on peut appliquer une interpolation par courbe de Bézier avec comme point de départ et point final de chaque vertex sa position dans les formes source et destination respectivement. Les points de contrôle seront les coordonnées sur le cercle calculées à chaque itération de la projection-répartition.

Hypothèses utilisées (mais aucune n'a été vérifiée avant) :

- Hypothèse 1 : on peut interpoler efficacement n'importe quelle forme "simplement connexe" vers un cercle. Inversement, on peut interpoler efficacement un cercle vers n'importe quelle forme "simplement connexe".
- Hypothèse 2 : On va supposer qu'une interpolation efficace est seulement définie par une le fait que pour tout temps t de l'interpolation d'une forme à l'autre, la forme reste simplement connexe (donc sans croisement).
- Hypothèse 3 : Je suppose que l'algorithme termine en moins de N itérations car si un point Pi est déplacé sur le bord du cercle, le point Pi-1 et Pi+1 peut être déplacé sur le bord de cercle à l'itération suivante.
- Hypothèse 4 : Sans le prouver ou l'avoir testé, j'espère qu'un point ayant été interpolé par une courbe de Bézier définie par les points de contrôle P0, P1, P2, ..., Pm n'intersecte pas une seconde courbe de Bézier définie par les points de contrôle Q0, Q1, Q2, ..., Qm si aucune paire de segment [Pn, Pn+1] n'intersecte la paire analogue [Qn, Qn+1]. <---- Nope, c'est FAUX, d'après les tests.
- Vérification d'un point du polygone dans un triangle : pour cela, on calcule le point d'intersection avec la ligne définie par le vertex courant et le bord du cercle, ainsi que le segment défini par chaque paire successive de vertex. S'il y a intersection, on rejette l'hypothèse de ce point pouvant être plaqué sur le bord du cercle. La vérification est aussi effectuée avec le point prédécesseur et avec le point successeur. On peut transformer ces 3 vérifications en disant que l'on vérifie que le triangle ABC (avec A le prédécesseur, B le bord de cercle et C le successeur) ne contient aucun point intérieur de la forme.

Réalité :

- Dans la pratique, je ne projette qu'un seul point par étape de projection. Après chaque projection ainsi qu'après chaque répartition, un point de control des courbes de Bézier est ajouté. Cela impose peut-être trop la forme circulaire intermédiaire, mais on est quasiment certains de ne pas obtenir de croisement.