Village - Fiche de lecture

Raoelisolonarivony - MISA M2

Janvier 2017

Première partie

A Survey of Procedural Methods for Terrain Modelling (2009) de Ruben M. Smelik et al.

Chapitre 1

Survey

1.1 Abstract

Les **méthodes procédurales** sont des alternatives négligées pour la création manuelle de contenu. Les points négatifs de ces méthodes concernent surtout l'*imprévisibilité et le manque de contrôle sur les résultats et l'absence de solutions intégrées*, bien que des publications récentes traitent de plus en plus la résolution de ces problèmes. Cette note recense les méthodes procédurales appliquées sur la **modélisation de terrain**, évaluant le réalisme de leur résultat, la performance et le contrôle que les utilisateurs peuvent avoir sur la procédure.

1.2 Introduction

Les mondes virtuels 3D sont de plus en plus complexes. Cependant, le processus, les outils et les techniques de modélisation de ces mondes sont laborieuses et répétitives, nécessitant des qualités de spécialiste en modélisation graphique 3D. Pour la modélisation procédurale, la philosophie est au lieu de créer les contenus à la main, *créer ses contenus automatiquement.* Cette approche a été appliquée avec succès pour générer, par exemple, *les textures, les modèles géométriques, les animations et même les sons.* Un sujet majeur dans la modélisation procédurale est la **génération automatique de modèles de terrain**, qui a commencé par les *phénomènes naturels comme les élévation de terrain et la croissance des plantes* dans les années 80 et 90 puis a étendue son champ aux environnements urbains au début du nouveau millénaire.

En dépit de résultats encourageants, la modélisation procédurale n'est pas souvent appliquée à la modelisation entière de terrain (mainstream terrain modelling). Plusieurs facteurs limitent cette transition de la modélisation manuelle vers la modélisation automatisée. Parmi eux, les articles de recherche autant que les outils commerciaux se concentrent sur un seul aspect de la modélisation de terrain (mt), (par exemple, la génération de profils d'élévation d'intérêt) et gère les aspects restants à

un degré limité ou pas du tout. L'intégration et l'ajustement des méthodes procédurales (mp) pour qu'elles puissent automatiquement générer un mt complet et consistent demeurent jusqu'à maintenant irrésolus. Un autre problème connue des méthodes procédurales est le manque de contrôle qu'elles procurent. L'aléa inhérent au résultat obtenu force souvent les utilisateurs à modéliser par "trial and error" (generate and test). De récentes publications traite ce problème avec des solutions spécifiques, mais elle n'a pas assez reçu encore d'attention suffisante.

Cette note présente un court recensement des méthodes procédurales appliquées à la mt. Nous discutons sur les propriétés importantes des méthodes, comme le réalisme du résultat, la performance de l'algorithme et les facilités qu'il offre aux utilisateurs sur le contrôle du processus de génération. L'objectif de cette note est double : donner aux lecteurs intéressés par le domaine de la modélisation de terrain procédurale une vue d'ensemble des recherches faites jusqu'à maintenant, vérifier les possibilités que les problèmes connus liés aux modèles de terrain sont en cours de traitement et ce qui reste à faire.

Dans (Smellik et al., 2008 [8], 2010 [7]), nous avons identifié plusieurs conditions à remplir qu'un framework de modélisation procédurale doit remplir pour être qualitativement acceptable et représenter une alternative productive au workflow de modélisation existant. Nous décrivons le design conceptuel d'un tel framework. Il intègre les modélisations procédurales et gère les dépendances entre les particularités des terrains dans le but de générer un modèle de terrain complet et consistent, qui correspond à un schéma grossier dessiné du terrain fait par l'utilisateur. Il concerne la distinction de plusieurs couches dans le modèle du terrain, chacune contenant des caractéristiques naturelles (terre, eau, couches de végétation) et faites par l'homme (route et couches urbaines). Nous présentons le design, l'implémentation et les résultats de deux de ces couches. Dans cette note, la distinction des couches du terrain est aussi pratique pour structurer le travail que nous recensons. Les sections suivantes discutent des méthodes procédurales pour les données d'élévation, les corps d'eau, la végétation, les réseaux routiers et les environnements urbains, suivi d'une concludion de l'état de l'art.

1.3 Height-maps

Height-maps, i.e. des grilles à deux dimensions contenant des valeurs d'élévation, sont souvent utilisés comme la base d'un modèle de terrain. Il y a beaucoup d'algorihtmes procédurales pour créer les height-maps.

Parmi les plus anciens algorithmes figurent les méthodes basées sur la subdivision. Un height-map grossier est itérativement subdivisé, chaque itération utilise un aléa controllé pour ajouter les détails. Miller (1986) [4] décrit plusieurs variantes de cette méthode de déplacement de point de milieu (midpoint), dans laquelle un nouveau point d'élévation est assigné à la moyenne de ces coins dans un triangle ou une forme de diamant additionnée d'une correction aléatoire. L'intervalle de la correction

diminue à chaque itération dépendant d'un paramètre qui contrôle la rugosité du height-map résultant.

La génération height-map est de nos jours souvent basée sur les générateurs de bruits fractals (Fournier et al., 1982 [2] ; Voss, 1985 [9]), comme le bruit de Perlin (Perlin 1985 [6]), qui génère des bruits en échantillonant et en interpolant des points de la grille de vecteurs aléatoires. Redimensionner et ajouter plusieurs niveaux de bruit pour chaque point du height-map conduit à des structures naturelles, pseudo montagneux. Un livre recommandé pour les bruits fractals et la génération de height-map et celui d'Ebert et al. (2003)[1].

Ces height-maps peuvent être transformés sur la base de filtres communs d'image (ex. lissage) ou sur les simulations de phénomène physique, comme l'érosion. L'érosion thermique réduit les changements aigus pour l'élévation, en distribuant itérativement le matériau depuis les points les plus hauts vers les plus bas, jusqu'à ce qu'un angle de talus, i.e. un angle maximale de stabilité pour le matériau comme la pierre ou le sable, soit atteint. L'érosion causé par les pluies (érosion fluviale) peut être simulée en utilisant, par exemple, l'automate cellulaire, où la quantité d'eau et de matériel dissout qui s'écoule en dehors des autres cellules est calculée sur la base de la descente (pente) locale de la surface du terrain. Musgrave traite tous les types d'érosion (Musgrave et al., 1989 [5] ; Musgrave, 1993) et Olsen (2004) discutent de plusieurs optimisations de calcul avec une qualité réduite mais acceptable. Benes et Forsbach (2001) introduisent une structure de terrain convenable pour des algorithmes d'érosion plus réaliste. Leur modèle de terrain consiste en des tranches de matériaux empilées horizontalement, chacune ayant des valeurs d'élévation et des propriétés des matériaux, ex. la densité. C'est une compromis entre la structure height-map limitée mais efficace et le terrain tout en voxel. Le modèle permet également d'avoir des couches d'air, par ce moyen il supporte des structure caverneux (des grottes).

Bien que ces algorithmes d'érosions ajoutent beaucoup de réalisme aux terrains montagneux, ils sont réputés lents, ayant besoin de faire des centaines, voire des milliers d'itérations. Une récente recherche s'est penchée sur des algorithmes d'érosion interractifs, souvent en portant l'algorithmes sur le GPU. Des exemples prométeuses comprennent (Anh et al. 2007) et (Stava et al., 2008).

La génération de height-map basée sur le bruit basique produit des résultats bien aléatoires ; les utilisateurs contrôlent le résultat seulement à un niveau global, souvent utilisant des paramètres non intuitifs.

Plusieurs chercheurs se sont penchés sur ce problème. Stachniak and Stuerwlinger (2005) propose une méthode qui intègre les contraintes (exprimées comme des masques d'images) dans le processus de génération de terrain. Il emploie un algorithme de recherche qui cherche un ensemble d'opérations de déformation acceptable à appliquer à un terrain aléatoire dans le but d'obtenir un terrain qui est conforme à ces contraintes.

Schneider et al. (2006) introduisent un environnement d'édition dans lequel l'utilisateur peut

changer le terrain en modifiant interactivement les fonctions de base du générateur de bruits (en remplaçant le bruit de Perlin par un ensemble d'images en niveau de gris dessinées par l'utilisateur).

Zhou et al. (2007) décrivent une technique qui génère le terrain basé sur un exemple en entrée height-map et une ligne dessinée par l'utilisateur qui définie l'occurence de caractéristiques de ligne courbée en niveau de gris, comme une crête de montagne. Les caractéristiques sont extraites de l'exemple height-map et doivent correspondre aux courbes dessinées et être recousu sur le height-map résultant.

De Carpentier et Bidarra (2009) introduisent les pinceaux procéduraux (procedural brushes) : les utilisateurs peignent le terrain en height-map directement en 3D en appliquant simplement les pinceaux élévante de terrain (terrain raising brushes) mais également des pinceaux basés sur le GPU qui génère plusieurs types de bruit en temps réel (voir Fig. 1a).

Saunders (2006) propose un méthode très différente, laquelle synthétise un height-map en se basant sur des Modèles d'élévation digitale (ou DEM : Digital Elevation Models) du terrain réel. Un utilisateur de son système Terrainosaurus dessine un carte 2D de régions polygonales, chacune d'elle est marquée pour avoir un certain profile d'élévation. Pour le réalisme, les bords direct (straight boundaries) de la région sont perturbés puis rastériser dans une grille. Un height-map est instantié en utilisant un algorithme génétique qui sélectionne les données DEM qui correspondent au profil d'élévation requis.

Kamal et Uddin (2007) présente un algorithme de déplacement en milieu de point sous contrainte (constrained mid-point displacement algorithm) qui crée une seule montagne correspondante à des propriétés comme l'élévation et l'étalement de la base. Belhadj (2007) introduit un système plus général où un ensemble de valeurs élévations connues contraignent le processus de déplacement en milieu de point (mid-point deplacement). Des applications possibles sont l'interpolation grossière et incomplète de DEM ou des lignes dessinées par l'utilisateur.

Une limitation inhérente aux height-map est qu'ils ne supportent pas les rochers surplombants et les grottes (cavités). Gamito et Musgrave (2001) proposent un système de gauchissement de terrain qui résulte à un surplomb régulier artificiel. Une méthode récente (Peytavie et al., 2009) procure une structure plus élaborée avec différentes couches de matériaux qui supportent les rochers, les arches, les surplombs et les grottes. Leurs modèles de terrain sont visuellement plausibles et naturels.

Comme illustration de l'état-de-l'art parmi les outils de support, (Voir Fig. 1d) pour un rendu d'un height-map généré par L3DT (Torpy, 2009), un des outils commerciaux pour la création de height-maps.

1.4 Rivières, Océans et Lacs

Pour générer des rivières, plusieurs auteurs ont proposés des algorihtmes qui s'exécutent soit pendant soit après la génération height-map. Kelley et al (1988) [3] prennent un réseau de rivière

comme la base d'un height-map. Ils commencent par une unique rivière bien droite et le subdivisent récursivement, produisant un réseau de courants. Cela forme un squelette pour le height-map, qui est complété en utilisant une fonction d'interpolation de données éparpillées. Le type de climat et le matériau du sol influencent la forme du réseau du courant.

Prusinkiewicz et Hammel (1993) combinent la génération d'une rivière courbée avec un schéma de subdivision de height-map. Du triangle de départ d'une rivière, un bord est marqué comme l'entrée et l'autre bord comme la sortie d'une rivière. Dans l'étape de subdivision, le triangle est divisé en triangles plus petites, et la course de la rivière à partir de l'entrée jusqu'à la sortie peut prendre maintenant plusieurs formes alternatives. L'élévation des triangles contenant la rivière est prise comme la somme des déplacements négatives de la rivière sur tous les niveaux de récursion (produisant le lit de la rivière), les autres triangles sont traités en utilisant le déplacement de milieu de point standard. Après huit ou plus récursions, la course de la rivière résultante parait raisonnablement naturelle. Un point négatif majeur de l'approche est que la rivière est placée à un niveau d'élévation constant et bas, et alors les cisailements s'enfoncent à travers un terrain montagneux de manière non naturelle.

Une approche plus avancée décrite par Belhadj et Audibert (2005) crée un height-map avec les crêtes de montagne et les réseaux de rivières. Commençant avec une carte vide, ils placent les paires de particules de crête sur une élévation particulièrement haute et les déplacent dans les directions opposées pour plusieurs itérations. Une courbe de Gauss est dessinée sur le height-map le long des positions de la particule pour chaque itération. Ensuite, ils placent les particules de la rivière suivant le sommet de la crête de la montagne et les laissent s'écouler vers le bas suivant des règles simples de la physique (comparable à l'érosion hydraulique). Le points restants entre les crêtes et les rivières sont complétés avec une technique de déplacement de point milieu inversé. Pour ce type de terrain spécifique, i.e. les crêtes de montagnes et les vallées irriguées par un réseau de rivière dense, la méthode est rapide et efficace.

A l'exception des rivières, les corps d'eau procédurale, comme les océans et les lacs et leurs connections, réseaux de courants, deltas et chutes d'eau, ont retenu peu d'attention jusqu'à maintenant. La formation des lacs n'est pas considéré du tout. Les océans sont communément générés en prenant un niveau fixe d'eau (ex. $0m$) ou en commençant par un algorithme d'inondation à partir des points à faible élévation. Teoh (2008) a également déclaré que la recherche sur ce domaine est incomplète : plusieurs rivières et caractéristiques côtières n'ont pas été traitées. Il propose des algorithmes simples et rapides pour les rivières sinueux, les deltas et la formation de la plage, qui requiert plus de travail pour augmenter le réalisme.

1.5 Les modèles de plantes et la distribution de la végétation

A propos de la végétation, des auteurs ont développé plusieurs procédures de génération d'arbres et de modèles de plantes et des méthodes de placement automatique de la végétation dans un modèle de terrain. Les premières sont utilisées pour obtenir rapidement un ensemble de modèles de plantes similaires mais variables d'une même espèce ; les secondes délestent les modeleurs de terrain de la tâche laborieuse de placer manuellement tous ces modèles de végétation individuels dans une forêt large.

Les modèles de plantes procéduraux croissent, en commençant à la racine, ajoutant de plus en plus de petites branches et terminant par les feuilles. Ils sont fondés sur la grammaire formelle (grammar rewriting). Prusinkiewicz et Lindenmayer (1990) discutent du Lindenmayer-system, ou *L-system*, un système de réécriture souvent utilisé. Ils expliquent comment les règles de production peuvent être appliquées en 3D, et présentent plusieurs exemples d'arbres générés ensemble avec leur grammaire.

Lintermann et Deussen (1999) proposent un système plus intuitif pour modéliser procéduralement les plantes, en plaçant les composantes de la plante (ex. une feuille) dans un graphe. Les composantes connexes peuvent être structurées dans un sous-graphe (ex. une brindille). Le système parcours le graphe, générant et plaçant les instances des composantes dans un graphe intermédiaire qui est utilisé pour la génération géométrique. La figure 1e) montre un arbre créé avec leur logiciel de modélisation de plante commercial XFrog.

Deussen et al. (1998) décrivent un modèle de simulation d'écosystème pour remplir une surface de végétation. L'entrée pour le modèle de simulation est le height-map et une carte de l'eau, plusieurs propriétés d'espèces de plantes, comme le taux de croissance, et, optionnelement une distribution initiale de plantes. Basé sur cela et en prenant en compte les règles sur la compétition du sol, des rayons de lumière et de l'eau, une distribution de plantes à l'intérieur d'une surface est itérativement déterminée (voir Fig. 1b), s'exécutant pendant plusieurs minutes.

Une autre procédure de placement de la végétation par Hammes (2001) est fondée sur les écosystèmes. Il utilise des données d'élévation, des élévations relatives, des pente, direction de la pente et des bruits multi fractals pour sélectionner un des écosystèmes définis. Les textures des végétations du sol sont générés à la volée, dépendant du niveau de détails et de l'écosystème. L'écosystème détermine également le nombre de plantes par espèce, lesquelles sont placées aléatoirement.

La modélisation procédurale de la végétation produit des résultats fiables et sont déjà bien appliquées dans les jeux vidéos modernes, par exemple en utilisant le package commercial SpeedTree.

1.6 Les réseaux routiers

La génération de réseaux routiers pour les villes peut être faite par des méthodes variées, parmi lesquelles nous traitons les approches basées sur les pattern, L-systems, les simulations d'agents et les champs tensoriels. La plus simple technique est de générer une grille de carré dense (comme dans Greuter et al. (2003)). Le bruit de déplacement peut être ajouté aux points de la grille pour créer un réseau moin répétitif, mais le réalisme de cette technique est toujours limité.

Une méthode plus élaborée pour créer des routes est par le biais des modèles (ou "templates"), comme proposés par Sun et al. (2002). Ils observent plusieurs patterns fréquent dans les réseaux routiers réels et tentent de les reconstruire. Pour chaque pattern, il y a un template correspondant : un template basé sur la population (implémenté comme un diagramme de Voronoi des centres de population), un raster et un template radial, ou une template mixte. Les principales artères de la carte de la route sont les autoroutes, qui sont générées en premier en utilisant ces templates de pattern. Des règles simples sont appliquées pour vérifier leur validité. Quand des zones d'impasse sont rencontrées (ex. les océans), elles sont annulées ou déviées. Ensuite, les routes principales sont souvent courbées pour éviter les gradients d'élévation large. Les régions qu'elles englobent sont complétées avec un raster de rues.

Parish et Müller (2001) utilisent un L-system étendu pour faire croître leur réseau routier. Le L-system est conduit par des objectifs précis (goal-driven) et les objectifs en question sont la densité moyenne de la population (les rues tentent de connecter les centres de population et les patterns spécifiques de routes). Des exemples de tels patterns sont le raster et le pattern radial. Leur L-system est étendu avec des règles qui ont une tendance à connecter les routes nouvellement proposées avec les intersections existantes et des règles qui vérifient la validité de la route en respectant le terrain impassable et les contraintes d'élévation. Les rues sont aussi insérées dans les zones restantes comme des grilles simples.

Kelly et McCabe (2007) introduisent l'éditeur de ville interactive CityGen, dans lequel un utilisateur défini les routes principales en plaçant des noeuds dans le terrain 3D. Les régions inclues par ces routes peuvent être complétées avec une de ces trois patterns : les grilles style Manhattan, les routes de croissance industrielle avec des routes sans issues et des routes organiques comme par exemple les suburbaines nord-américaines.

Glass et al. (2006) décrivent plusieurs expérimentations de réplication de structures de route rencontrées dans les colonies informelles d'Afrique du Sud en utilisant la combinaison d'un diagramme de Voronoi pour les routes majeures avec des L-systeme ou des subdivisions régulières avec ou sans bruit de déplacement pour les routes mineures. Ils ont raisonnablement réussi à recréer les patterns observés.

A la différence des approches basées sur les grammaires ou les patterns ci-dessus, Lechner et al. (2003) prennent une approche basée sur l'agent, dans laquelle ils divisent la ville en zones incluant non seulement les zones résidentielles, commerciales ou industrielles, mais également des zones spéciales comme les bâtiments gouvernementaux, les squares, et les institutions. Ils placent deux agents, appelé l'*extender* et le *connector*, sur une position de semence dans la carte du terrain. L'extender recherche les zones non connectées dans la ville. Quand il trouve une telle zone qui est localisée non loin d'un réseau routier existant, il cherche le chemin le plus convenable pour connecter cette zone au réseau routier. Dans Lechner et al. (2006), les auteurs étendent cette méthode avec, entre autres choses, des agents qui développent les petites rues. Cette méthode donne des résultats plausibles, mais son désavantage est son temps d'éxectution long.

Chen et al. (2008) proposent une modélisation interactive de réseaux routier en utilisant les champs tensoriels. Ils définissent comment créer des patterns communs de routes (grille, radial, suivant les bords) en utilisant des champs tensoriels. Un réseau routier est généré à partir d'un champ tensoriel, en traçant des lignes aérodynamiques à partir des points de semence dans les directions majeures des vecteurs propres jusqu'à ce qu'une condition d'arrêt soit remplie. Ensuite, suivant la courbe tracée, de nouveaux points de semence sont placés pour tracer les lignes aérodynamiques dans la direction perpendiculaire (vecteur propre mineur). Des utilisateurs peuvent placer des bases de champs tensoriels, comme un pattern radial, lisser le champ, ou utiliser un pinceau pour contraindre localement le champ vers une direction spécifique. Du bruit peut être appliqué pour rendre le réseau routier moins régulier et en conséquence plus plausible.

Dans les méthodes discutées, l'influence de la carte du terrain sous-jacente et du profil d'élévation est à degré variable prise en compte. La plupart des méthodes prennent seulement des mesures basiques pour éviter des routes montantes avec des pentes trop raides et des routes qui traversent des corps d'eau. Kelly and McCabe (2007) planifient le chemin précis de leurs routes principales entre les noeuds mis en place par l'utilisateur pour avoir autant de possibilités dans l'élévation que possible. Seulement, pour les terrains durs cette mesure ne sera pas adéquate et le terrain doit être modifié pour l'accomodation de la route. Bruneton and Neyret (2008) proposent une méthode simple et efficace pour mixer les profiles des routes dans les height-map en utilisant les shaders.

1.7 Les environnements urbains

Kelly and McCabe (2006) donnent un point de vue élaboré de plusieurs approches pour la génération d'environnement urbains. Watson et al. (2008) donne un point de vue pratique de l'état de l'art.

L'approche la plus commune pour générer des villes procéduralement est de commencer par un réseau de routes dense et identifier les régions polygonales inclues dans les rues. La subdivision de

ces régions produisent des lots, pour lesquels différentes méthodes de subdivisions existent, voir par ex. Parish and Müller (2001) ou Kelly and McCabe (2007). Pour remplir ces lots avec des immeubles, soit la forme du lot est utilisée directement comme empreinte d'un immeuble, ou l'empreinte d'un immeuble est adapté pour le lot. En faisant simplement l'extrusion de l'empreinte jusqu'à une hauteur aléatoire, on peut générer une ville avec des gratte-ciels ou des immeubles de bureau. Pour obtenir des formes d'immeubles intéressantes, plusieurs approches ont été mises en oeuvre.

Greuter et al. (2003) génère des immeubles de bureau en combinant plusieurs formes primitives dans un plan d'étage et en faisant l'extrusion de cette forme pour différentes hauteurs. Parish and Müller (2001) commençent par un plan d'étage rectangulaire et appliquent un L-system pour affiner l'immeuble. Ces approches sont surtout utiles pour des modèles simples d'immeubles de bureau.

Wonka et al. (2003) introduisent le concept de *split grammar*, une grammaire formelle indépendante du contexte (context-free formal grammar) dont le design est fait pour produire des modèles d'immeubles. Le split grammar ressemble à un L-system, mais il est bas" sur des formes comme éléments primitifs à la place des symboles. Dans leur système, un style spécific d'immeuble peut être acquise en prennant un attribut au début du symbole, lequel est propagé pendant la réécriture. Dans un modèle d'immeuble, le style peut être différent pour chaque étage (ex. un immeuble d'appartement avec des boutiques au le rez-de-chaussé). Leur approche repose surtout sur la génération cohérente et plausible des façades pour des immeubles au forme relativement simple. Larive and Gaildrat (2006) utilise un grammaire de la même sorte, appelé *wall grammar*. Avec cette grammaire, ils sont capables de générer des murs d'immeuble avec des détails géométriques additionnels, comme les balcons.

Müller et al. (2006) appliquent une autre type de grammaire, appelée *shape grammar*. La propriété principale d'un shape grammar est qu'il utilise des règles dépendant du contexte (context-sensitive), tandis qu'une split grammar utilisent des règles indépendantes du contexte (context-free), qui dans ce cas permet la possibilité de modéliser les toits et les formes arrondies (rotated shapes). Ils commencent avec une réunion de plusieurs formes volumétriques lesquelles définissent les bords des immeubles. Cette forme est ensuite divisée en étages et les façades produites sont subdivisées en murs, fenêtres, et portes par le moyen d'un système de grammaire. Dans une étape finale, le toit est construit sur le sommet de l'immeuble. Fig 1f montre un réseau routier et Fig 1g la ville correspondante générée par leur produit commercial, CityEngine (Procedural, inc. 2009). En plus des immeubles d'affaire bien connus, la grammaire peut aussi modéliser des immeubles résidentiels, ex. les maisons suburbaines ou les anciennes villas romaines.

Bien que les shape grammars dans Müller et al. (2006) peuvent générer des modèles d'immeubles visuellement convaiquants, Finkenzeller and Bender (2008) notent que l'information sémantique à propos du rôle de chaque forme dans l'immeuble final est manquante. Ils proposent de capturer cette information sémantique dans un graph typé. Leur workflow comprend trois étapes. En commençant

par des traits de contour brute, un graphe de style d'immeuble peut être appliqué à ce modèle. Cela produit une représentation intermédiaire de graphe sémantique de l'immeuble, laquelle peut être modifiée ou regénérée avec un style différent. Dans la dernière étape, la géométrie est crée sur la base du modèle intermédiaire, et les textures sont appliquées, produisant l'immeuble 3D final. Finkenzeller (2008) décrit avec plus de détails la génération des façades et des toits de leur système (voir fig. 1c).

Young et al. (2004) décrivent une méthode pour créer des maisons au style vernaculaire du sud-est de la Chine en utilisant un shape grammar étendue. La grammaire est hierarchique et commencent au niveau de la ville (tandis que dans d'autres méthodes un shape grammar est appliqué à l'empreinte d'un immeuble individuel). La grammaire produit ensuite les rues, les blocs d'habitation, les routes, et même les maisons de productions avec des composantes comme les portails, les fenêtres, les murs et les toîts. A travers des règles de contrôles (définissant par exemple, les contraintes de ratio de composants) la validité des immeubles peut être testée. En appliquant ce système de grammaire, une ville ancienne typique du sud-est de la Chine peut être généré avec des résultats plausibles, puisque les style d'immeuble de ces villes est très rigidement structuré.

Müller et al. (2007) utilisent une approche très différente pour la construction des façades des immeubles. Leur méthode prend un image unique de la façade de l'immeuble réel comme entrée et est capable de reconstruire un modèle détaillé de la façade en 3D, en utilisant une combinaison d'image et de génération de shape grammar.

Bien que les méthodes ci-dessys donne rapidement et visuellement des résultats plaisants, les villes qu'elles génèrent manque de structure réaliste. Une nouvelle recherche incorpore les théories et les modèles d'aménagement de territoire urbain existantes dans le processus de génération. Groenewegen et al. (2009) présentent une méthode qui génère une distribution de différents types de districts selon les modèles d'aménagement urbain dans l'Europe de l'Ouest et l'Amérique du Nord. Ils prennent en compte un grand éventail de facteurs significatifs, incluant le plan historique de la ville et de l'attraction que certains types de terrain (côteaus, océans, rivières) peuvent avoir par ex. les districs industriels ou résidentiels de grande classe. Weber et al. (2009) utilisent des modèles comparables (comparaison un peu simplifiée) pour une simulation d'une ville dans le temps. Leur système est rapide (environ 1s pour un an de simulation) et interactif, ce qui veut dire que l'utilisateur peut guider la simulation en changeant les routes ou en mettant une peinture pour le sol utilisé sur le terrain.

1.8 Conclusions

Les méthodes procédurales pour la modélisation de terrain deviennent de plus en plus attractif pour l'industrie et le domaine académique. Nous avons classifié ces méthodes dans cinq domaines principales, et discuté d'une grande variété d'approches de recherche et des résultats obtenus grâce à elles :

l'élévation de terrain, les éléments d'eau, la végétation, les réseaux routiers et les environnements urbains.

Depuis les tous débuts, où l'intérêt se focalisait surtout sur la génération de height-map, jusqu'à maintenant, avec un déplacement vers de plus en plus d'environnements urbains réalistes, il y a un considérable corps de résultats disponibles. Beaucoup de méthodes procédurales basiques déploient des blocs communs de construction comme le bruit, la réécriture formelle ou les simples systèmes de simulation, pour chacune un large nombre de variantes, souvent spécifique à un domaine, sont de nos jours proposées, en particulier pour la route et les catégories urbaines. Parmi toutes les catégories discutées, les zones relatives à l'eau sont clairement les plus sous-développées.

Concernant les recherches à venir, plusieurs intéressantes tendances ont été identifiées. Parmi elles, trois directions prometeuses peuvent être résumées comme suit. Premièrement, la performance et l'intéractivité des méthodes procédurales continuent de progresser, souvent au moyen de programmation parallèle sur le GPU. Deuxièmement, les réseaux routiers et les zones urbaines continueront certainement de s'améliorer dans le sens de la variété et des niveaux de détails, mais le bond du réalisme sera fourni probablement par le déployement de plus de sémantiques autant dans le processus de génération procédurale que dans les modèles générés (Tutenel et al., 2008). Et enfin, la clé du déploiement répandu des méthodes procédurales par les non-experts (ex. les designers de jeux, les artistes, les designers de scénarios) sera l'intégration des méthodes procédurales dans un framework, offrant entre autre chose, plus de contrôles intuitifs, d'outils pour générer des modèles complets de terrain et des mécanismes non intrusives pour le maintien de la consistence parmi les caractéristiques générés (Smelik et al., 2009)

Deuxième partie

Génération automatique d'environnements virtuels urbains (2016) de Tiavina NIVOLALA

Résumé

La génération automatique d'environnements virtuels urbains trouve une application dans la planification urbaine.

Deux approches de génération :
— la génération direct à partir d'un modèle statique
— la génération procédurale
La méthode de Tiavina est une combinaison de ces deux approches. En deux étapes :
i) créer un *modèle de base de données hiérarchisée* contenant les *éléments fondamentaux d'un environnement urbain*
ii) appliquer des techniques procédurales pour générer automatiquement un environnement urbain à partir du *style architectural* correspondant à des éléments de la base nouvellement créée

Introduction générale

A partir des villes existantes, on a simulé les villes virtuelles. Les villes virtuelles sont utilisées et représentent des plateformes pour l'intégration d'information.

La visualisation et l'exploration des paysages urbains pour assister à :
— la planification urbaine
— le contrôle de trafic
— la gestion de désastres
— la création de mondes virtuels
— la réalité augmentée (superposition au monde réel)
Les villes ont des structures complexes d'où le besoin d'avoir un *modèle de données normalisé* pour obtenir une **structuration de données cohérente et interopérable ?**.

L'augmentation de la qualité des jeux vidéo à base de polygones en conjonction avec l'augmentation des performances des cartes graphiques demandent et permettent la **visualisation de grande quantité de données**. Il n'est pas commode de créer manuelle de grandes quantités de modèles, d'où le besoin de techniques procédurales. Sinon construite à la main, la construction de la ville virtuelle prendrait d'énorme délai.

La génération des objets avec une approche procédurale est aggrémentée de caractère aléatoire contrôlable. Le souci est qu'elle produit des résultats imprévisibles en forme et en apparence.

Chapitre 1 :

— Aperçu des différents standards et modèles déjà crées pour le stockage et la représentation des villes

— Quelques techniques de visualisation basés sur des modèles déjà existants

— Méthodes de visualisation purement procédurales.

Chapitre 2 : son approche pour la création du système

— La conception et l'alimentation de la base

— La génération des villes à partir de cette base

— Méthodes de visualisation purement procédurales.

Chapitre 3 :

— Outils utilisés pour la création du système

— Interface du système

Chapitre 2

Etat de l'art

2.1 Introduction

Generation de villes :

— depuis modèle existant → normes de représentation → représentation du modèle → visualisation (lecture du modèle)
— depuis photo (aériennes ou terrestres)
— de façon procédurale (sans modèle)

En deux parties :
— les standards dans la modélisation des villes
— les techniques de visualisation de ces dernières

2.2 Représentation des bâtiments et des villes

Plusieurs sources (modèles, procédures) ⇒ besoin d'adoption d'une forme de représentation.

Objectifs :
× décrire (représentation)
× gérer
× stocker

2.2.1 IFC

Format neutre ? pour :
× décrire + échanger des infos de construction de bâtiment
× norme pour openIBM ?

× objectif : fournir une base universelle pour échanger des données concernant les bâtiments
× orienté BdD, sur des définitions de classes d'objets (éléments, processus, formes) pour les projets de construction (ex. d'entités : *IfcWall*, de repr. de géométrie : *IfcExtrudedAreaSolid*
× IFC structure qui renfèrme des espaces : *IfcSpatialStructureElement* ⇒ résultent en sites/buildings/étages (⟵ dérive) *IfcProduct* ⇒ a un lieu, une représentation géométrique (modèles solides)

2.2.2 CityGML

Vue d'ensemble

Standard d'échanges et de représentation de données 3D Pour les modèles virtuels de villes et de paysages

× infos : géométrie/apparence/sémantique/topologie des objets
× Basé sur XML, ISO 19100 ? ?, Version 3.1.1 de Geography Markup Language (GML 3.1.1)
× Norme de l'Open Geospatial Consortium (OGC)

Modularisation de l'info géospatiale urbaine

Modules

Ville complexe ≈ Couverture de plusieurs thématiques des objets urbains CityGML rassemble :
× Modèle de base : concepts et composantes basiques
× Module d'ext. thématique : dépend du module de base

Les onzes modules d'extension thématiques :
× Building
× Apparences ? ?
× CityFurniture ? ?
× CityObjectGroup ?
× GenericCityObject ?
× Landuse
× Relief
× Transportation
× Vegetation
× WaterBody
× TexturedSurface

LoD

Pour faire les actions suivantes, CityGML utilise 5 LoD :

— mode de collecte
— application diverses

× Building
× Apparences ? ?
× CityFurniture ? ?
× CityObjectGroup ?
× GenericCityObject ?
× Landuse
× Relief
× Transportation
× Vegetation
× WaterBody
× TexturedSurface

2.3 Résultats

2.3.1 Introduction

La partie résultat décrit l'implémentation et les outils technologiques utilisés. Tiavina décrit ensuite l'interface du système.

2.3.2 Outil et langage

L'implémentation est faite sur Unity3D en relation avec une base de donnée PostGreSQL par via le language C#.

Unity3D est un moteur de jeu multi-plateforme possédant un gestionnnaire d'import robuste et intelligent acceptant les fichiers depuis les logiciels Maya, Blender, 3DSMax. Il garde les matériaux et textures intactes pour les modèles 3D.

Rendu : ses moteurs graphiques utilisent OpenGL, Direct3D et OpenGL ES pour l'usage multi-plateforme. Ils possède des fonctions de rendu de texte, des cartes d'ombres pour les ombres dynamiques.

Scripting : il est construit sur Mono, sur le framework .NET opensource et permet diverses languages possibles : Uniscript (similaire à javascript), C# et Boo.

Gestion de ressources : Unity3D possède un serveur de contrôle d'Assets pour les scripts de développement et les ressources de jeu. PostgreSQL pour gérer l'audio et Theora Codec pour la lecture vidéo. Il est capable de créer des modèles de terrain et de végétation.

Physique : bénéficie du support du moteur PhysX pour la simulation de tissu en temps réel sur

différents meshes, RayCasts et la gestion de collisions.

2.3.3 Présentation de l'interface du système

L'utilisateur doit modéliser un élément de la ville : bâtiment, véhicule ou autre objet urbain. Ce modèle alimente la base utilisée pour la visualisation de modèles 3D.

L'utilisateur doit attribuer les fonctions des éléments (fenêtres / portes) sous forme de propriétés personnalisées dans Blender. Les éléments peuvent être ajoutés à des moments différents pour le bâtiment. [1] .

Script python : effectue la transformation des modèles via Blender

Après éxécution du script, l'étape de décomposition des modèles de la scène en éléments individuels ; Ensuite vient l'étape d'insertion des éléments dans les tables de la base selon leur propriétés.

Les coordonnées de multi-polygones sont utilisés, certains modèles sont stockés en tant que tel après uniformisation s'ils sont susceptibles d'être réutilisés plusieurs fois dans la même scène. Intégration des informations géométriques et sémantiques pour reconstituer les données de la base pour la visualisation.

Unity3D <—(NpgSQL)—BDD<—API Python—Modèle<——Utilisateur

Générer une ville à partir d'un ensemble d'éléments prédéfinis correspondant au style architectural de la ville séléctionnée. Le gros volume de données et le besoin d'obtenir des résultats variés nécessitent des techniques procédurales.

L'utilisateur choisit une région, liste de villes est proposée, il choisit la ville et valide, à la prochaine scène, la génération de la ville se charge. [2]

Ustyle urbain est caractéristique d'une ville. Ici ont été prises Antananarivo, une ville américaine et une ville de l'Ile de Ré. Par exemple, les maisons sont sans étages et les toits en tuile, les volets et portes sont peintent en bleus et verts à l'Ile de Ré.

Pour chaque ville, 150 bâtiments sont générés sur une aire définie.

En premier, le terrain et la rue sont générés. Ensuite viennent les bâtiments sur les zones non occupées par la rue, qui divise le terrain en deux parties de bâtiments et objets urbains. Les bâtiments sont génénés par des polygones et des transformations d'objets prédéfinis, placés de manière aléatoire sur un emplacement vide, leur orientation est décidée suivant la position de la porte orientée vers la rue avec une petite aléa de rotation.

A partir des données de la BdD, les éléments thématiques sont générés et assemblés. Alors, ils

1. Tiavina ? inona no tinao ho ovaina (ampiana na esorina) ankoatran le hoe atao procédural daholo .. inona sisa ny anjaran ny utilisateur
2. Le génération an le ville v vao misafidy pays(région) dia efa manomboka sa rehefa mi-clic Générer ?

sont convertis en entité d'Unity3D pour représenter les bâtiments, les rues, les objets urbains,...

Un mixage des éléments thématiques en respectant les dimensions et les positions des éléments qu'ils remplacent permet d'avoir une diversification du paysage urbain.

L'éparpillation des objets urbains sur une plus grande surface et entre les bâtiments où de l'espace est disponible. La taille des bâtiment est vérifiée et corrigée pour rentrer dans une fourchette de taille définie.

Une fois les bâtiments placés, l'objet rue est identifié et récupéré. A partir de la position et des dimensions de la rue, ainsi que des dimensions de chaque véhicule correspondant à la ville, des véhicules sont positionnés le long de la rue à une certaine distance les unes des autres, dans les deux sens de la file. Chaque véhicule se déplace vers l'avant jusqu'à arriver au bout ou il disparaît et réapparait au début de la rue et ainsi de suite.

2.3.4 Conclusion

Le système est capable de recréer des variations de ces bâtiments en respectant la fidélidé du style de ville, les villes sont reconnaissables selon les modèles de base incorporant des caractéristiques (couleurs/textures, formes d'ouvertures).

2.4 Conclusion et perspectives

L'objectif de Tiavina : générer une ville à partir d'un ensemble d'éléments prédéfinis correspondant au style architectural choisi [3]

Elle a décrit les standards de modélisation existants et les techniques de visualisation. Elle a explicité le problème de création de contenus et les avantages des techniques procédurales.

Son traivail en deux étapes : la première consiste en la conception/modélisation d'une BDD hierachisée (infos + éléments constituant une ville) pour la génération de la ville. C'est une de ses contributions, la mise en relation de la BDD et la visualisation.

Elle s'est inspirée du standard CityGML et l'a couplé à la génération procédurale. Cet approche hybride de Tiavina est en d'autre mots le couplage entre éléments statiques (modèles manuelles) et procédurales.

Les résultats sont des villes avec éléments variés et des styles urbains contrôlables.

Petite remarque : les textures plaquées sur les facades extérieures des bâtiments, les ouvertures, décorations sont traitées de façon à part entière, détacher et ré-associer les textures des surfaces.

3. Afaka hazavainao ahy v ny fomba hahazahoana ny style architectural (efa défini ao @ le base zan le iz dia ampiasaina fotsiny, am le fotoana mi selectionne liste région iny ? fa techniquement

Le résultat : les villes sont aléatoires, fruit de l'échange des éléments architecturaux permettant d'obtenir des variations tout en maintenant le nombre de modèles stockés dans la base est réduit au minimum.

Le système a un rôle utile dans la construction d'un modèle urbain. Le respect des normes établies avec une bonne initialisation lors de la modélisation ont permis d'obtenir son résultat.

Les lacunes du systèmes :

Une perspective d'évolution : automatiser l'import des modèles vers la base, cela permet d'éviter le nommage manuel des éléments descriptifs d'un bâtiment. Un algorithme pour reconnaître les éléménts thématiques à partir des caractéristiques distinctes comme la position rapport à certains éléments, de réduire la tâche de l'utilisateur au minimum.

L'intégration de plus d'éléménts thématiques 'objets urbains' et bâtiments : le classement des bâtiments par fonction (résidentiel, commercial, industriel), l'ajout de règles de placement pour assurer la cohérence de l'ensemble des environnements.

Les styles des bâtiments, la forme de rues, et la subdivision en rue secondaires. Pour le moment une seule rue pour diviser le terrain en deux. Pour avoir des environnements réalistes : il faut générer les rues et la construction rapide de réseau routier primaire. Il est envisageable de diviser les rues en cellules pour définir les voisinages.

Troisième partie

Level detail for 3D Graphics - Terrain Level of detail (2003) de David Luebke et al. - page 211 - 253

[4]

algorithmes géospatial data

Terrain Level of Detail

Terrain visualisation has received a large amount of interest. Important for :

— Flight simulators
— Terrain based-computer games
— GIS (Geographic Information System)
— Military mission planning

Flight simulators (Schachter) : need to optimize the number of primitives for a scene, display obj in lower detail as they appeared further away.

GIS ([Weibel) : generalisation ≈ LOD (simplification of map info at different map scales

2.5 Introduction

Terrain simplification : Reference to Heckbert and Garland course SIGGRAPH'97.

this chapter : recent techniques (real-time view dependent) to have accurate vizualisation of massive, distributed terrain models.

Terrain have more constrained geometry : uniform grids of height values, so allows simpler algorithms.

Terrain data bring complications :

— eg1 because terrain is **continue**, a large amount of terrain is visible at any point, so importance of *view-dependent LOD techniques*.
— eg2 terrain meshes can be **dense**, so need of *paging techniques*, so that can be viewed on common desktop. Illustration : USGS elevation data set at 30-arc-second (1 km at the equator) = height grid of $43,200 \times 21,600 = 933$ million points, around 1.8 billion triangles over the entire planet [5].

4. Misy article récent v noho ity 2003 ty ?

5. Ask the specialist of programmer of Cities : Skylines, or look for a forum : Hello, I'm perfect stranger to you, I'm preparing a master Thesis about procedural generation of village and the simulations of people, water and electricity at the University of Antananarivo, Madagascar, I will use Unity3D and a database of rules, could you please advise me on the litterature first to read or the useful techniques to simulate functionality (electrical) or the tools I need to use, I have 6 months to give a little demo!! You're work is amazing and I think it's a nice start to look at your methods for Cities : Skylines ... I hope to establish a solid base of the simulation that can be extended in the future ... thanks in advance for any of your advices .. and keep up the nice work ... and sorry for the disturbing mail of a stranger like me, I can imagine that you have more important and difficult problem to solve ...

2.6 Multiresolution techniques for terrain

variables faced by developer when implementing a terrain LOD algo. 1.specific techniques valuable 2. background on geographic coordinate systems and files formats (producing geographically accurate sim) 3. useful resources on the Web relate to terrain LOD.

2.6.1 Top down and bottom up

differentiators of LOD algo : whether they are **top-down or bottom-up** approach to the simplification problem.

top-down algo, begin with two or four triangles for the entire region and then progressively add new triangles until the desired. (=subdivision or refinements meth)

bottom-up algo begins with the highest-resolution mesh and iteratively removes vertices from the triangulation until the desired level of simplification is gained. (=decimation or simplification meth)

bottom-up tend to be able to find the minimal number of triangles required (-) necessitate the entire model being available at the first step and higher memory and computational demands

distinction between sense of bottom-up vs top-down :
— preparation-time
— run-time
bottom-up : used during the initial offline hierachy construction

at run-time : a top-down approach might-be favored (support of view culling)

rmq : (1)interactive bottom-up = hybrid combination with top-down quatree block framework (2) a few systems perform incremental coarsening or refining of the terrain at each frame (taking advantage of frame-to-frame coherency) : syst unclassified strictly as top-down or bottom-up

2.6.2 Regular Grids and TINs

Another distinction between terrain LOD algo : the **structure** to represent the terrain.

Two major approaches :
— regular gridded height fields : array of height values at regularly spaced x and y coord (eg grid of $65 \times 65 = 4,225$ height values
— triangulated irregular networks (TINs) : variable spacing between vertices (512-vertex TIN with the same accuracy of grids 65×65)

TINs approximate surface to a required accuracy with fewer polygones (eg allow large flat regions to be represented with a coarse sampling, while reserving higher sampling for more bumpy regions). Regular grids tend to be far less optimal than TINs because the same resolution is used accross the

entire terrain, at flat places as well as high-curvature regions. TINs flexibility in the range and accuracy features that can be modeled, such as maxima, minima, saddle points, ridges, valleys, coastlines, overhangs, and caves.

However, grids simple to store and manipulate : - elevation at any point by bilinear interpollation of the four nearest neighbor points - easily integrated with raster databases and file formats, DEM, DTED, GeoTIFF, - less storage for the same number of points : only an array of z values needs to be stored rather than the full (x,y,z) coord. - TINs make implementing related functions (such as view culling, terrain following, collision detection and dynamic deformations) complex because of the lack of simple overaching spatial organization - TIN less applicable to view-dependent LOD [6] vs grid

so contemporary (2003) favor grids over TINS

nb :(1) hybrid schemes (hierachical triangulation based on reg grid, Evans et al. [7] triangular irregural network (RTINs) : binary tree data structure to impose spatial structure on a triangle mesh, compact repr similar to grid but supports nonuniform sampling feature of TINs.

(2)wavelet transform of gridded data to produce an adaptative mesh tessellation at near-interactive rates

2.6.3 Quatrees and Bintrees

Represent different parts of the grid at different resolutions : hierarchical repr. options : quadtree and binary triangle tree.

Quadtree : a rectangular region is divided uniformly into four quadrants. Each of these quadrants divided into four smaller regions, and so on. NB : employ a quadtree structure and use triangles as your primitives : decompose each rectangle into two or more triangles. rmq : many triangulation schemes possible independent of quadtrees or bintrees

A binary triangle tree (BTT) : the same way as a quadtree, instead of segmenting a rectangle into four it segments a triangle into two halves. The root triangle is a right-isosceles triangles (two of the three sides are equal and they join at a 90-degree angle), and the subdivision is performed by splitting this along the edge formed between its apex vertex and the midpoint of its base edge.

6. To be analyse, view-dependant LOD
7. io v nen Tiavina ?

Quatrième partie

Julien Pettre and Ming Lin. 2014. New generation crowd simulation algorithms. ACM SIGGRAPH 2014 Courses

Les foules sont souvent simulées par des algorithmes microscopiques parce qu'ils permettent de générer des trajectoires fluides pour les individus, ils sont basés sur les intéractions locales entre les agents (le mouvement est la combinaison entre les mouvements locaux et les interactions).

Les principaux problèmes sont les mouvements émergents (l'apparition de nouveau mouvement) qui sont imprévisibles. Des algorithmes de simulation de modèles de mouvements et d'interaction consistent à associer chaque modèle à un mouvement spécifique de la foule avec des trajectoires spécifiques. Le point négatif est le manque d'oscillations dans les trajectoires, les trajectoires improbables, les collisions résiduelles et les agents tombant dans des cul-de-sac.

Le document détaille les progrès effectués dans les algorithmes de simulation basés sur la vélocité (velocity-based), l'impact de la qualité du réalisme du mouvement dans l'industrie et ses applications.

— Première partie : la définition des modèles microscopiques de simulation avec les principales solutions proposées (les algorithmes basés sur la vélocité par rapport aux autres approches).
— Deuxième partie : La technique : les solutions les plus utilisées basées sur la RVO (Reciprocal Velocity Obstacles)
— Troisième partie : Comment utiliser les algorithmes en pratique : dans les situations d'urgence, les films

2.7 Introduction

La foule est constituée d'un ensemble d'individus dans un même endroit partageant les mêmes objectifs.

Chaque individu a des interactions avec ses voisins (dépendant de facteurs individuels et du milieu ie. physique, psychologique, social et environnemental)

La foule en mouvement dans les places publiques, les immeubles ou les évènements forment surtout des interactions physiques (les individus s'évitent, se suivent, se regroupent ou se dispersent). Combinées, ces interactions forment une structure émergente à plus grande échelle qui consistuent les principales caractéristiques des mouvements.

Ces études sont importantes pour :
— les architectes afin de prédire le traffic des piétons et améliorer la qualité de service dans les immeubles publics
— les designers dans le but de simuler le champ de bataille ou de créer les populations dans les scènes virtuelles
— les designers de jeux vidéos créant des scénarios convaincants de villes virtuelles

Deux classes majeures d'approches : l'approche macroscopique (considérer l'aspect global du mouvement de la foule comme la modélisation comme un fluide visqueux) et l'approche microscopique

L'approche microscopique : donne des trajectoires d'agent individuel lisses, cette approche est basée

sur les modèles d'interactions entre les agents, le comportement global est un phénomène émergent (plusieurs interactions entre les agents)

Les éléments de la simulation sont :

1. La séléction des voisins : trouver l'ensemble des voisins en interaction avec l'agent
2. Le modèle d'interaction locale : comment chaque interaction influence le mouvement des agents
3. La combinaison des interactions : intégration de toutes les interactions pour un agent donné

Les nouveaux modèles d'interaction locale : les nouveautés des modèles basées sur la vélocité (datant cependant de plus de dix ans) sont capables de simuler comment les humains dirigent leur mouvement en anticipant ceux des autres, une percée dans le niveau de réalisme.

Partie 1 : Définition du modèle local d'intéractions basé sur la vélocité, de quelle manière permet-il d'anticiper les réactions

Partie 2 : Un modèle populaire : RVO et ses variantes

Partie 3 : Les applications, chez la compagnie Gelaem avec les scènes de foules dans les films. Les bénéfices de la RVO et Unity3D.

2.8 L'algorithme de simulation basé sur la vélocité

Objectif de l'approche microscopique : calculer les comportements macroscopiques en simulant les interactions entre les individues à une échelle locale (par exemple, la simulation des piétons).

Prédiction des conditions du trafic global depuis des modèles numériques d'interactions physiques d'une foule en mouvement : éviter les obstacles statiques ou en mouvement dans le voisinage, ie. éviter les collisions et l'interpénétration entre les corps simulés.

La simulation basée sur la vélocité : un nouveau type de modèles numériques (2007) pour le **besoin de réalisme dans toutes les échelles** (même les plus petites).

Les soucis des techniques précédentes : manque d'anticipation, les agents bloqués dans des situations de cul-de-sac, les oscillations de mouvement étranges

Eviter la collision avec anticipation : les manoeuvres d'évitement sont effectuées bien avant que les agents soient proches les uns des autres.

L'anticipation ne peut pas provenir de modèles microscopiques formulant l'interaction comme une fonction de la distance entre les agents (comme les systèmes de particules et les modèles inspirés par la physique)

Dans les modèles basés sur la vélocité, les intéractions sont définies en fonction de la position de

chaque agent (son état) et de sa vélocité (dérivée de la position par rapport au temps)

Une décomposition, pour chaque agent, du domaine de vélocités atteignables (tous les mouvements globaux qu'un agent peut effectuer) en deux composantes : le domaine des vélocités admissibles et celui des vélocités inadmissibles.

L'espace admissible est l'ensemble des vélocités avec lesquelles un agent peut se déplacer sans risque de future collision. Le domaine inadmissible contient les vélocités provoquant des collisions.

La notion de temps est liée à celle du risque de collision par la TTC (Time-to-collision) qui décrit le risque suivant la dimension de temps.

Comment trouver les domaines, quand il y a des obstacles en mouvement, quand chaque agent effectue une adaptation indépendamment les un des autres ?

En travaillant sur de courtes fenêtres de temps et en mettant à jour constamment l'état et les décisions de l'agent, on obtient des résultats de simulations fluides. Les modèles peuvent être ensuite évaluer avec des données réelles pour avoir des simulations réalistes.

Les travaux récents sur les modèles d'évaluation[8] sur des données réels ont montré que cette catégorie d'approche est prométeuse.

L'objectif de la note : un aperçu des solutions existantes

Section 2 les modèles basés sur la vélocité par rapport aux autres méthodes et une liste des modèles basés sur la vélocité
Section 3 Les modèles basés sur la vélocité appliqués à la réalité
Section 4 Trois modèles basés sur la vélocité pour la description pédagogique et leurs singularités
Section 5 Les différences entre les modèles + les axes futures de développement pour ces nouvelles catégories de modèles

2.8.1 Etat de l'art

La simulation de la foule est possible à partir de deux perspectives :
— macroscopique : considérant la foule comme un corps cohérent
— microscopique : les interactions locales entre les agents
le point de vue macroscopique voit la foule comme un fluide, et traite son mouvement avec la dynamique des fluides par les champs de vélocités potentiels. Le principe étant d'obtenir un comportement cohérent de la foule, laissant de côté les buts individuels et les contraintes mais en restreignant l'interpénétration (qui est d'ailleurs la contrainte principale pour la simulation de la foule) au dernier moment. Le point négatif de l'approche macroscopique est rencontré au niveau locale quand par exemple des agents se déplacent sur le mauvais côté (sideways), entrent en collision, subissent des inter-pénétrations résiduelles.

8. Modele d'évaluation de méthodes ? ? sur quelle base ?

Pour l'approche microscopique, les contraintes et buts individuels sont plus importants. Le comportement global est juste le résultat qui émerge des interactions locales.

Les méthodes pour simuler les interactions locales :

1. **Discrétiser l'espace en cellules** (comme pour les automates cellulaires) telles que les agents sont les cellules occupées. Les interactions sont modélisées comme les règles de transition (probabiliste) entre les états et la compléxité varie avec le niveau de discrétisation. [9]. Différents comportements peuvent être modélisés de cette manière : par exemple la formation de rangée entre deux groupes qui se croisent. Le point négatif de cette méthode est le besoin d'avoir des niveaux de discrétisation, ce qui n'est pas pratique pour modéliser des stratégies d'évitement de collisions complexes [10]. Pour résoudre ce problème, les agents sont interdits de se déplacer dans des cellules occupées.

2. **Utiliser des modèles non discrets**, laissant les agents prendre des décisions selon les données entrées et des critères. Un premier exemple est le **modèle de Boids** où les agents décident en respectant les règles :

— séparation : éviter le surpeuplement (attroupement)
— alignement : aller vers un but commun (une direction commune)
— cohésion : rester toujours grouper, ne pas se séparer du groupe

Un deuxième exemple reposent sur les lois physiques, les agents sont soumis à des forces (**Social Forces model** - SF), c'est un modèle basé sur la position et toutes les forces qui agissent sur les agents en ces positions. Le premier avantage de ce modèle est qu'il permet d'avoir des comportements pour éviter les collisions complexes, ensuite il permet de modéliser des phénomènes tels que l'amitié et les forces attractives (au lieu des seules forces répulsives rencontrées devant les obstacles)

3. Les **modèles basées sur la vélocité** sont des modèles évolués des modèles SF, appelés aussi modèles prédictives. Ils traitent plus d'information, ils sont capables de prédire les trajectoires. Le moyen le plus populaire est l'établissement d'un domaine de vélocités admissibles. Il ne reste alors qu'à choisir la vélocité (par exemple celle qui mène au but) avec la plus petite accélération [11]

4. Le **Dynamic Window** est une méthode utilisée par les robots pour éviter les collisions, c'est un algorithme d'exploiration [12] greedy des vélocités les plus possibles en mettant un poids (les dynamiques des robots, la collision correspond à un poids élevé). Cela a inspiré les méthodes velocity-obstacle et les premières approches d'évitement de collision dans les simulations de foules.

5. La technique de **Paris** pour éviter les collisions repose sur le concept d'espace de vélocité : diviser l'espace devant l'agent en secteurs de vélocités, chacun ayant un couple maximum/minimum de vitesses définissant l'intervalle des vitesses de collision. L'agent est alors libre de choisir la vélocité convenable en dehors de ces intervalles.

9. Rechercher des exemples sur le net [12] à voir
10. existe-t-il différents types de collision ?
11. pourquoi la plus petite accélération ?
12. cf Vlady/Vonjy

6. Le modèle **RVO (Reciprocal Velocity Obstacle)** de Van den Berg fait intervenir deux agents avec un espace de vélocités relatives, chaque vélocité (un élément dans l'espace des vélocités) entrainant une collision dans une certaine fenêtre de temps est rendue indisponible, formant un obstacle de vélocités. L'agent choisit une vélocité hors de l'obstacle de vélocités. Cette méthode a été mis à jour en ajoutant la capacité de diviser les efforts d'évitement, et ensuite en utilisant des informations sur l'accélération (origine des modèles Acceleration-Velocity Obstacle).

7. La méthode **Linear Trajectory Avoidance** parcours un ensemble de mouvements possibles et associe un coût (une fonction des distances de croissements futures) à chacun. Le coût est estimé en extrapolant les trajectoires étant donné les positions courantes et les vélocités. Cette technique est utilisée dans le suivi par vidéo des piétons réels comme un prédicteur pour le traqueur, et donne des résultats améliorés, spécialement pour les occlusions.

8. Le modèle **Tangent** : les agents (et leur espace personnel) sont représentés dans l'espace de vélocité relative. Le but étant de trouver l'espace d'interaction (là où les agents sont les plus rapprochés) en utilisant les positions relatives aux espaces personnels des marcheurs, une décision est faite sur lequel des agents passe en premier et lequel cède le passage. L'effort d'adaptation est partagé (share ?) et asymétrique comme dans les scénarios de la vraie vie.

9. Le model **Vision** d'Ondrej : essaye de résoudre les interactions en se basant sur des informations accessible depuis le flot visuel, ainsi simulé une boucle de perception-réaction. Les agents s'appuyent sur les angles de portée et l'approximation du temps jusqu'à la collision (time-to-collision) pour prendre des décisions. Ce modèle est capable d'intéragir avec n'importe quel obstacle static ou dynamic aussi longtemps que l'agent peut le voir.

Ce sont les modèles plus antérieures qui sont les bases de plusieurs développement.

Il faut parcourir les derniers développement sur les algorithmes de la famille RVO.

La section suivante discute du pourquoi et du comment ces nouvelles classes d'algorithmes améliorent le réalisme de la simulation de la foule.

2.8.2 Experimental validation of velocity based models

principle of velocity based models = compute the admissible velocity domain (velocities at which an agent can move without provoking collisions. Each model proposes -meth to compute this domain -meth to select a solution

all : based on linear extrapolation of the current situation

do humans perform such a prediction ? anticipate future and react accordingly ? this hypothesis made by velocity-based models is validated by an experiment :

Une expérimentation a prouvé que les humains anticipent les futures collisions et réagissent

rapidement pour les éviter. L'expérimentation consiste à mettre deux personnes dans deux coins adjacents d'une pièce 25mx25m, séparées par une cloison partielle. Les deux personnens partent chacun de leur coin et se déplacent vers le coin opposé de la pièce tel que le centre du carré soit leur point de rencontre. La distance (appelée MPD) à laquelle ces deux personnes sont supposées se rencontrer est notée pour chaque pair de 400 participants. A la fin de la cloison, dès que les deux personnes se rendent comptent de la direction et de la vitesse de chacun, la distance prédite augmente pour éviter la collision. Ils [6] observent que (i) Le contrôle de la distance MPD comment aussitôt que les participants se voient (ii) Cette distance ne cesse de croître positivement (monotone croissante) (iii) Cette distance MPD n'est pas exagérément augmentée, aucune manoeuvre superflue (iv) Cette manoeuvre s'arrête aussitôt que la collision est évitée.

L'existance de ces manoeuvres d'évitement valide les méthodes basées sur la vélocité pour la simulation de la foule, rendant le comportement plus réalistes par rapport à celui des méthodes basées sur les distances comme les SF ou les modèles de particules.

2.8.3 3 examples de modèles basées sur la vélocité

L'Inria a dévoloppée trois solutions pour les modèles basées sur la vélocité, à décrire, discuter et comparer : (i) Paris (ii) Tangent (iii)Vision-based

On rappèle que l'objectif est de définir un ensemble de vélocités sans collision avec des obstacles statiques ou en mouvement (vélocités admissibles) et de choisir une solution spécifique parmi ces vélocités.

Le modèle de Paris

Le modèle de Paris consiste à définir le domaine des vélocités inadmissibles en prenant dans des intervalles de temps la vélocité et la position de deux agents A et B. Dans un intervalle de temps $[t_i, t_{i+1}]$, l'intervalle d'angle la zone concernée par l'interaction entre A et B $[\theta_i, \theta_{i+1}]$ est calculée ainsi que la vitesse maximale a ne pas dépasser pour céder le passage s_max en plus de la vitesse minimale a respecter pour pouvoir passer avant l'autre agent s_min. Pour l'agent A, le processus est itérativement effectué pour chaque intervalle de temps et pour chaque agent. Au final l'intersection des domaines inadmissibles déterminés à chaque intervalle de temps fournit les vélocités inadmissibles et une fonction de coût minimise les deviations d'angle et les distances pour obtenir la meilleur solution. Si les vitesses s_min et s_max doivent être en fonction du temps, seules les pires cas sont traités pour un intervalle de temps et un agent donné. Plus l'intervalle de temps est courte, plus la précision du modèle de Paris est grande. Pour trouver vrai compromis entre précision et performance, Paris suggère un échantillonage de temps à intervalle irrégulier ($t_1 = 1s, t_2 = 2s, t_3 = 4s$). Le modèle de Paris gère également l'évitement de la collision avec des obstacles statiques, seules les vitesses s_{max} sont traitées parce qu'il n'y a de

sens de traiter s_{min} pour ce cas. [13]

Tangent model

h [14] h [15] h [16]

Le modèle "Tangent" reproduit également des interactions expérimentales entre des piétons avec un haut niveau de réalisme, et se repose sur la perception par un agent des vélocités des agents en interaction avec lui. En particulier, il introduit un terme d'erreur ϵ permettant de reporter les manoeuvres d'évitement de collision à moins qu'une perception précise ne soit obtenu (la vitesse relative d'un agent B par rapport à un agent A qui fait l'appréciation : $V_{B/A} + \epsilon$ doit se trouver hors de la zone d'interaction). Le timing des interactions est reproduit.

Le principe de se modèle est de calculer la vélocité relative de l'agent B (observé par A) par rapport à la vélocité de l'agent A : $V_{B/A}$ (rappelons que $V_{B/A} = V_{B/W} - V_{A/W}$ où W est le système de référence du monde. Cette vélocité sert de référence à A pour adapter sa propre vélocité pour éviter les futures collisions, en changeant par exemple l'amplitude (augmenter ou réduire la norme de la vélocité de A : $V_{A/W}$) et la direction (gauche ou droite : ou en $rad.s^{-1}$ pour la vitesse angulaire) de sa vélocité. L'agent A établit alors une projection des espaces d'interactions à risque par rapport à la vélocité qu'elle doit prendre. Un système linéaire d'inégalités est résolu pour obtenir l'espace des vélocités hors risque de collision.

L'évitement des obstacles statiques avec des géométries décrites comme des ensembles de segments [15] A la différence du modèle de Paris, les risques de futures collisions sont explicitées mais l'intervalle de temps des possibles collisions n'est pas définie. En revanche, grâce à l'introduction d'erreur, un temps de réaction est simulée. Le système linéaire est progressivement construit jusqu'à ce qu'un certain nombre d'interactions soient atteintes où que l'espace des solutions soit vide. A ce point, les agents réagissent avec un risque élevé de collision.

Vision model

Le modèle "Vision" se basent sur l'établissement d'un rapport rassemblant les vitesses angulaires et les time-to-collision (ttc) qu'un agent A peut avoir avec ses voisins. L'espace est alors subdivisée en pixels et chaque pixel p_i possède les propriétés de vitesse angulaire et de ttc qui peut varier pendant le temps. Sur la base des valeurs de ces propriétés $(\dot{\alpha}, ttc)$, deux domaines spécifiques définissent les

13. il y aurait plus de réalisme en simulant également les collisions, juste une idée ? besoin de documentation : piétons distraits par le téléphone, les belles filles, les boutiques, piétons agées ou enfants avec une vue et une locomotion réduite, handicapées

14. represent an agent human as not an circle but as an segment view from the top

15. how about the collision with person who are moving in the same sens, ie you see the back of the agent : the $v_{B/A}$ should pointed in the same direction as the A, like in a street, troittoir,

16. collision avoidance is different of fear of a person : real life ... persons are keeping their direction, in particular in a étroit street, but just ajust the vélocity and apply a self rotation to almost enter in contact with the other agent

ensembles des données pouvant conduire à des collisions : le domaine des risques collisions imminentes (éviter par l'agent en réduisant la vitesse, cas des ttc_i avec des valeurs très basses proches de zéro : des agents très proches) et celles des risques de futures collisions (éviter en changent son orientation, cas des $\dot{\alpha}_i$ très basses proches de zéro : des agents sur le front). Une combinaison de ces deux réactions forme un ensemble de processus itératif "perception-action" de la part de l'agent concerné. Ce modèle a des propriétés intéressantes : il peut gérer différentes formes d'obstacles, vu qu'il considère les pixels, il prend implicitement en compte la visibilité des obstacles. Il est capable de simuler l'émergence de pattern connu des mouvements des piétons et des conditions de traffic [17].

2.8.4 Discussion

Les trois modèles ont été développés dans des buts différents : celui de Paris, développé en premier pour fournir une adapatation anticipée de la simulation de la foule, le modèle "Tangent" pour définir le début de la réaction d'un agent, et celui "Vision" qui définit le processus itératif réel de perception-action que les deux autres modèles ne considèrent pas.

En second lieu, ils sont basés sur des espaces de contrôle différents : le modèle de Paris opère dans un espace orientation-vitesse (module de la vélocité), celui "Tangent" dans un espace angulaire-tangentielle de vélocité, et enfin le modèle "Vision" contrôle indépendamment les mouvements (orientation) et la vitesse. Ces trois modèles prennent en compte les géométries variées, propriété souvent négligée : Paris échantillonne les obstacles comme un ensemble d'agents statiques, le modèle "Tangent" permet de simuler des interactions avec des segments de lignes (utile pour les environnements avec des immeubles), et le modèle "Vision" traite inconditionnellement tous les sortes et formes d'obstacles car le contrôle du mouvement de l'agent est exécuté suivant une représentation graphique.

En troisième lieu, ces modèles gère le temps différemment : Paris produit les risques de collision avec les instants de collision successifs, alors que le modèle "Tangent" indique ces risques avec un seul time-to-collision, explorant pourtant les instants futures.

Finallement, le modèle de Paris et celui "Tangent" nécessite des techniques additionnelles de filtrage, de triage et de sélections d'interactions. Un agent ne doit pas intéragir avec des agents trop nombreux. L'utilisation du notion de voisinage dans la simulation de foule est importante concernant les comportements émergents et le modèle "Vision" est satisfaisant vu qu'il sélection l'action à entreprendre en se basant sur la visibilité de l'obstacle et son importance du point de vue de la taille de l'image percue. Si ce modèle simule des phénomènes similaires à la réalité, est-ce cette propriété qui en est la cause ?

D'autres questions en attente de réponse sur les modèles basés sur la vélocité comme récemment, les modèles sur le comportement de suivi, évalué à un haut degré de réalisme par comparaison sur des données expérimentales. Comment combiner, par exemple, les comportement de suivi et d'évitement pour simuler des comportements complexes de groupes, crucial pour les foules simuler de manière

17. A vérifier : well known pedestrians patterns ? ?

naturelle.

Le fondement des modèles basés sur la vélocité doit être remis en question : un simple extrapolation linéaire de trajectoires basée sur la position actuelle et la vélocité est parfois une mauvaise prediction, spéciallement pendant les manoeuvres (une petite déviation peut modifier la prédiction entre deux instants donnés) : dans des places bondées où les gens redéfinissent constamment leur mouvement, les modèles basés sur la vélocité sont-ils inutiles ? Probablement pas, parce que les calculs sont constament mis à jour. Seulement, des prédictions, au niveau des capacités réelles de l'homme, pourrait rendre ce nouveau type de modèle plus réaliste et utile.

2.9 Interactive modeling, simulation and control of large-scale crowds and traffic

cf 2.11.2

2.10 Applications

2.10.1 L'industrie du cinéma et le design architectural (Golaem

Golaem a développé un logiciel d'animation de personnages pour les artists VFX, les studios d'animation et de développement de jeux vidéos pour peupler le fond, le mi-fond de leur scène (Technicolo, Toei Animation, Game of Thrones, zombies for Walking dead). Golaem a été fondé en 2009 par des chercheurs et ingénieurs de l'Inria (déjà plus de 15 ans d'existence). Le logiciel Golaem Crowd est facile à appréhender pour les artistes (intégration aisée dans Autodesk Maya) dont les principales caractéristiques sont :

- un Manager d'Asset pour générer rapidement des éléments visuels variés : meshes, props, textures et shaders ;
- un outil de placement de personnage pour facilement peupler des stades, definir des formations ou automatiquement des couples ou groupes d'amis ;
- un moteur d'animation pour les bipeds, les quadrupeds ou les n-ped procédurallement générant des nouveaux mouvement en reciblant, courbant, mirroring et avec des contraintes IK (inverse kinematik)
- un moteur de navigation procurant un recherche de chemin automatique et des évitement de collision pour des personnages individuels, des groupes ou des hordes ;
- un moteur de Physique qui prend en compte les ragdolls et les interactions physiques ;
- un editeur de comportements basés sur des noeuds
- des plugins pour les rendus proceduraux pour Arnold, V-Ray, Mental Ray, Renderman, 3Delight

Au moment de la création du logiciel, aucun outil ne permettait d'avoir un algorithme de navigation capable de gérer toutes ses situations, d'où l'implémentation de plusieurs algorithmes, suivant les modèles Social Force, Reciprocal Velocity Obstacles et notre propre algorithme d'extrapolation.

Dans cette classe, nous allons d'abord donnée un apercu de l'architecture de Golaem Crowd et l'illustrer par une démonstration. Ensuite nous présenterons les algorithmes de navigation à partir de plusieurs cas d'utilisation. Finallement, nous discuterons de l'impact de ces algorithmes sur l'animation de personnage.

2.10.2 L'industrie du divertissement (Unity Technologies) : La Velocity Based Obstacle Avoidance dans le jeu

La Velocity Based Avoidance est adpaté pour la simulation de personnages crédible dans les jeux vidéos. Cependant, la plupart des algorithmes suivent les comportements de Raynold. Dans cette présentation, une revue de l'intégration des techniques basées sur les obstacles de vélocités dans les jeux vidéos, et l'évolution des besoins de framework d'obstacle avoidance depuis la pré-prod, la prod à la version finale. Bien que les simulations tentent de reproduire le phénomène de foule réel, les créateurs de jeux veulent souvent chorégraphier les comportements. Nous allons discuter des implications du design du jeu sur les framework obstacle avoidance, les barrières à franchir et les solutions produites. Nous citerons les problèmes courants dans les situations de game play et les suggestions basées sur les techniques développées récemment. Finallement, nous nous concentrerons sur l'interplay entre la vélocité pour planifier une animation. La qualité de perception de la locomotion est une mesure importante de qualité pour les jeux, et souvent directement traduit comme à quel point les personnages du jeux paraissent intélligents. Nous verrons comment la qualité de l'animation peut être amélioré en impliquant plus de contraintes à l'étape de planning des vélocités. La qualité de l'animation est l'ensemble des résultats du système entier de simulation commençant par la recherche de chemin jusqu'à l'évitement d'obstacle.

2.11 Annexes

2.11.1 Pedestrian Reactive Navigation for Crowd Simulation : a Predictive Approach (2007) of S. Paris, J. Pettré and S.Donikian

Abstract

L'article décrit l'approche de résolution du *problème de déplacement autonome des piétons* dans le domaine de la simulation de la foule.

La méthode a pour but de *résoudre les interactions entre les piétons et l'évitement de collision.* Elle est basée sur les agents et est prédictive, c'est-à-dire que chaque agent percoit les trajectoires des

agents environants et fait leur extrapolation pour réagir en avance à des collisions potentielles.

Les auteurs veulent des *résultats réalistes*, ils effectuent une calibration du modèle depuis des données expérimentales de capture de mouvement. Les résultats obtenus sont valides et résolvent les points négatifs (le manque d'anticipation) des approches précédentes (les oscillations).

Dans l'article, ils font une représentation mathématique et décrivent l'implémentation du modèle. Pour finir, ils discutent de la calibration et de la validation par rapport à des données réels.

Introduction

Simuler les interactions entre les piétons est difficile à cause de la *complexité croissante à mesure que la densité de la population augmente*. Le *réalisme est mis à mal par notre capacité à détecter les artifices de la simulation* vu notre habitude à cotoyer de mouvements réels de piétons dans notre quotidien. Cette technique de navigation réactive peut être appliquée dans le *domaine de l'architecture, de la sécurité, de l'ergonomie de l'espace, du divertissement*. L'attente de voir *émerger un mouvement de foule naturel* depuis notre simulation microscopique de piétons est compréhensible.

Les inputs du modèle sont :
— la définition de l'environement
— un plan de navigation
  — l'état courant de chaque piéton
  — la destination de chaque piéton

La méthode consiste à vérifier les interactions futures des piétons : l'évolution de leurs positions par extrapolation de leur état actuel.

Le modèle est calibré par les données de capture de mouvement effectuée en deux temps :
— la calibration du modèle avec les premières données acquises en mesurant les interactions entre deux participants
— la capture des mouvements de plusieurs participants (le maximum supporté par notre système) pour tester et valider la simulation avec les données acquises.

Nous avons contribuer sur deux plans :
— la résolution des points négatifs des approches précédentes (oscillations, jams) qui manquent d'anticipation et réagissent trop simplement face aux collisions
— la mise en place d'une calibration/validation par capture de mouvement qui permet la décomposition en temps et dans l'espace de la réaction vis-à-vis des collisions potentielles. Les méthodes de validation classiques reposant sur des séquences de vidéos configurées manuellement. Elles manquent de précision. Notre méthode permet d'extraire un critère pour détecter le besoin de réaction et calculer les corrections pour une trajectoire adéquate.

2.11.2 Interactive modeling, simulation and control of large-scale crowds and traffic

Cinquième partie

A Fractal Model of Mountains with Rivers (1993) by Prusinkiewicz and Hammel

Leur modèle combine en un seul algorithme la génération de terrain montagneux et d'une rivière.

Ils constatent la grande similarité entre une famille d'algorithmes de simulation de montagnes (midpoint-displacement model) et de la rivière (squig-curve model), d'un point de vue des mécanismes d'écriture. Ils se basent sur des systèmes de générations fractales de Mandelbrot (context-sensitive rewriting). Ils étudient la génération basée sur les contextes qui décrit une interaction avec les triangles composants le terrain. Les côtés des triangles voisins partagent des propriétés communes (l'altitude à mettre en place, le caractère d'entrée/sortie de la rivière).

D'abord ils décrivent chaque méthode séparément en détails et ensuite font l'intégration des deux méthodes dans un processus commun.

Côté technique, ils notent les affirmations de Mandelbrot selon lequel les modèles de terrain fractal manquent l'inclusion des réseaux de rivières. Ils ajoutent qu'avant les méthodes de génération étaient séparées : une pour la génération du terrain et une autre pour les rivières. Bardeen, selon eux, a commencé à intégrer les deux générations en une méthode (Panorama).

Ils utilisent une table de nombre aléatoires (pseudo) pour le déplacement de point de milieu. Ainsi, il associe les triangles disposés sur une même altitude en piochant le nombre dans la table de hachage.

Les mathématiques des les fractals de Mandelbrot et la recherche de chemin dans un graphe planaire (simplifié) pendant l'intégration des deux méthodes sont relativement expliqués. Ce dernier servant par exemple à décrire la création du lit de la rivière.

Ils indiquent les points négatifs de leur méthodes : les rivières trop aplaties, les montagnes avec de pentes abruptes, la rivière sans source, le rendu graphique à améliorer.

Finallement, ils donnent de bonnes explications détaillées des deux principes (Midpoint-displacement et Squig curve) avec des illustrations très claires.

C'est un article intéressant car il réunit des techniques ayant des similarités (ex. les tiles).

Sixième partie

Procedural Content Generation for Games, a survey (2013) by Hendrikx

2.12 Taxonomy des méthodes pour la génération procédurale de contenu

2.12.1 Pseudo-Random Number Generators (PRNG)

La nature donne souvent l'illusion d'aléa, par exemple dans la forme d'une montagne, un nuage, ou une fleur. Les PRNGs peuvent, cependant, être utilisés pour mimer cet aléa rencontré dans la nature.

Perlin Noise [Perlin 1985 ; 1990] est un générateur de bruit basé sur PRNGs avec une vaste utilisation dans les médias et le divertissement. Ce bruit génère une carte de point de données (des valeurs aléatoires). Cette carte de données est générée par un seeded PRNG (ajout d'un seed) par l'interpolation. Plus de détail peut être ajouté à la carte de bruit en combinant plusieurs couches de bruit de Perlin et en utilisant un redimensionnement (scaling). Le Perlin noise est représentative pour beaucoup de techniques basées sur les PRNG utilisées dans les jeux vidéos [Van Verth and Bishop 2008] ; d'autres PRNGs populaires existent [Lecky-Thompson 2001].

2.12.2 Generative Grammars (GG)

Les Generative Grammars, concoctés depuis l'étude de langages de Noam Chomsky vers les années 1960, sont des ensembles de règles qui, opérant sur des mots individuels, peuvent générer des phrases strictemment et grammaticalement correctes. Ils peuvent être utilisés pour créer des objets corrects à partir d'éléments codés comme des lettes/mots. Dans cette section nous discutons des L-systems, split grammars, wall grammars, et shape grammars, lesquels ont été utilisés pour la génération de contenu dans le cinéma.

Lindenmayer-systems

(L-systems) est une grammaire consistuée de symboles qui décrivent les caractéristiques d'un objet. Un string généré par la grammaire décrit la structure ou le comportement d'un objet.

Split Grammars

Similaires aux L-systems, les split grammars [Wonka et al.2003] travaillent sur des formes string-encoded. De nouvelles formes sont générées à partir d'un ensemble basique de formes en appliquant des règles de réécriture régissant la conversion forme-à-forme (shape-to-shape), le split grammar génère une nouvelle forme. Par exemple, en utilisant un split grammar une forme de mur initial (wall-shape) peut être divisée en deux formes plus petites, lesquelles à leur tour peuvent être réécrites (converties) en une forme de contour de fenêtre et une forme de fenêtre. Les split grammars sont context-free, c'est-à-dire que le processus de réécriture garde le même résultat étant donné un ensemble de règle de réécriture, quel que soit l'ordre dans laquelle le string des symboles est évalué.

Wall Grammars

[Larive and Gaildrat 2006] sont spécifiquement créés pour la création des extérieurs des immeubles. Les formes sont manipulées pour former une extérieur d'immeuble similairement aux split grammars, mais les wall grammars peuvent générer des formes plus élaborées - par exemple, la règle d'extrusion de mur peut conduire à des formes complex 3D comme les balcons et les issues de secours (fire escapes).

Shape Grammars

[Müller et al. 2006] sont des grammaires context-sensitive et séquentielles dont l'origine est le travail de Stiny vers les années 1970s. Pour chaque étape de réécriture, le symbole et ses voisins dans le string déterminent quel(s) symbole(s) remplace le symbole original. Cependant, le processus de réécriture des shape grammars est différent des L-systems ou des split grammars. Similairement aux wall grammars, les shape grammars peuvent générer des structures plus complexes.

2.12.3 Image Filtering (IF)

L' Image Filtering a pour but principal l'amélioration d'une image par rapport à une mesure (subjective), ou pour mettre en valeur certaines caractéristiques d'une image pour afficher (partiellement) des informations cachées. Beaucoup de techniques ont été développées pour IF ; les tutoriaux annuels de l'image processing de SIGGRAPH forme un bon recensement/évaluation (survey) de l'état de l'art avec des applications dans les domaines de l'animation et des films. Dans cette section, nous présontons deux techniques relatives à l'IF, la morphologie binaire et les filtres de convolution.

Binary Morphology

est un ensemble de techniques utilisé pour les opérations binaires sur les images. L'image binaire, souvent nécessaire aux opérations binaires, peut être obtenue par seuillage (thresholding), un processus où les pixels ayant une intensité inférieure à une seule d'intensité définie sont mis à zéro et les pixels restants à un, créant effectivement une image binaire. Les exemples typiques d'opérations de morphologie binaires incluent la dilatation, où des pixels sont ajoutées aux coins d'un élément d'une image, et l'érosion, qui fait l'opposé. En combinant des opérateurs binaires basiques, des opérations plus complexes peuvent être réalisées. Par exemple, en faisant la dilataion d'une image et ensuite en soustrayant l'image originale du résultat, les coins de chaque élément de l'image originelle sont mis en évidence.

Convolution Filters

sont des filtres qui peuvent être représentés par une image simple (fonction ou ensemble de données discret). La convolution est un opérateur mathématique sur deux fonctions ou des ensembles de données discrets, où une fonction peut être utilisée pour modifier l'autre et ainsi créer une nouvelle fonction ou ensemble de données discret. Ces types de filtres peuvent être utillisés par exemple pour enlever les bruits, lisser, rendre pointue, détecter des coins des objets, ou même détecter la direction du mouvement des objets sur une image. En utilisant les filtres de convolution, une simple texture peut être manipulée pour créer un texture entièrement nouvelle, et ainsi économiser de l'espace de stockage.

2.12.4 Spatial Algorithms (SA)

Les Spatial Algorithms manipule l'espace pour générer le contenu du jeu. Le résultat est créé en utilisant une entrée ayant une structure, par exemple une grille, une récurrence. Dans cette section, nous discutons du tiling et du layering, de la grid subdivision, des fractals, et des Voronoi diagrams.

Tiling and Layering

Tiling est une technique utilisée pour créer une espace de jeu en décomposant la carte en une grille. La grille n'est pas limitée à une taille rectangulaire - les formes hexagonaux sont aussi très communes. Les grilles sont des structures de données 2D, mais des projections isométriques [18] peuvent être couplées aux grilles pour créer l'illusion d'une carte 3D. Le layering est une technique qui intègre dans la même carte plusieurs grilles, appelées layers (couches). Un tile est alors construit en superposant les parties de chaque couche, parmi lesquelles doivent contenir des parties transparentes. Cette approche permet la création d'effets overlay, comme l'écoulement de l'eau, et des espaces de jeu 3D-looking en utilisant une quantité limitée de textures sources de terrain.

Grid subdivision

est une technique iterative et dynamique pour la génération d'objet. Un objet est d'abord divisé en une grille uniforme avec les textures appropriées. Un algorithme de grid subdivision, par exemple l'algorithme Patch-LOD [Pi et al. 2006], est utilisée pour rajouter itérativement du détail à l'objet. Le côté dynamique de cette technique lie la génération itérative avec le point de vue - seules les cellules de la grille proches du point de vue courant doivent être divisées en plus petites cellules pour créer le niveau de détail demandé. Une exemple utilisant cette technique est le rendu d'un terrain généré procédurallement : seules les cellules proches du joueur sont détaillées (générées), alors que le reste du terrain est plus flouté (coarse), ainsi économisant la puissance de calcul.

18. isometric projection ? ? ?

Fractals

sont des figures récursives qui se copient elles-mêmes, par exemple une flocon de neige de Koch (snowflake). Avec les fractals, peu de paramètres contrôlent une vaste portée de résultats possibles. Un avantage des fractals est que les objets qui semblent avoir des détails infinis peuvent être stockés comme une simple fonction récursive. La génération de fractals est processus resource-intensive (utilisant intensivement des resources), à cause de la récursion. En utilisant les Iterated Function Systems [19], l'image résultante peut être générer plus rapidement, utilisant un processus itérative au lieu de récursive.

Voronoi Diagrams

[Aurenhammer 1991] sont des décompositions des espaces métriques (metric spaces) en des parties ayant des tailles et des formes déterminées par la position de seed points (point d'intérêt) dans l'espace métrique. La décomposition établit les bords des points à égale distance des seed points les plus proches, et le territoire contenant exactement un seed point et toutes les localisations pour lesquelles le seed point du territoire est le point d'intérêt le plus proche. Pour une petite carte avec peu de points, un diagramme peut être facilement calculé en utilisant une approche itérative. Cependant, quand la collection de points augmentent en taille, une approche de calcul plus élaboré est requise. Une variété d'algorithmes ont déjà été proposés pour rendre les diagrammes de Voronoi utilisable (presque) en temps réel [De Berg et al. 2008, Ch.7].

2.12.5 Modeling and Simulation of Complex Systems

Il n'est pas pratique dans certains cas de décrire un phénomène naturel avec des équations mathématiques. Les modèles et les simulations peuvent être utilisés pour combler ce problème. Dans cette section nous décrivons le cellular automata, les tensor fields, et l'agent-based simulation.

Cellular Automata

[Chopard and Droz 1998] Un cellular automaton (automate cellulaire) est un model de calcul discret basé sur les cellules alignées sur une grille, dans laquelle chaque cellule possède un état et est soumise à un ensemble de règles communes. Le model de calcul est appliqué à des étapes de temps discrets. Les règles de table (board) détermine comment le voisinage de la cellule et son état influencent son état au pas de temps suivant. Le comportement résultant peut être aléatoire, mais aussi périodique. Cellular automata peut être combiné avec d'autres systèmes pour former de nouveaux modèles de calcul. Un exemple d'un système combiné est l'open L-system, une combinairson d'automate cellulaire et de L-Systems. Dans les open L-systems, le comportement des objets est largement déterminé par l'interaction avec l'environnement contenant l'objet et par l'interaction avec les autres objets de

19. Iterated Function Systems ? ?

l'environnement [Hidalgo et al. 2008]. Cela permet la modélisation de contraintes spaciales, comme la distance minimum entre deux objets.

Tensor Fields

[Chen et al. 2008, Sec.4] sont des généralisations en 2D de vecteurs, qui peuvent être utilisées pour spécifier la forme d'un espace de jeu. Les tensors décrivent la direction de l'élévation de la carte. Comme les lignes de tensor peuvent être visualisées, elles sont souhaitables pour le design interactif et la manipulation de réseaux routiers.

Agent-based simulation (ABS)

[Davidsson 2001] est basée sur la modélisation d'une situation complexe en utilisant des individus, appelés agents. Le comportement émergeant, qui est, le comportement complexe qui résulte à partir des interactions simple entre agent, est un caractère d'une ABS qui le différencie du comportement moyen observé avec les techniques de modélisations traditionnelles. Les agents peuvent être ajoutés, supprimés, ou remplacés pendant la simulation ; les agents doivent également apprendre en temps.

D'autres systèmes complexes et théories

Beaucoup d'autres systèmes complexes et théories font et devraient faire leur chemin dans la génération de contenu procédural pour les jeux vidéos, incluant les théories de la dramatic act (pour la génération d'histoire - story generation), du processus cognitive - cognitive process (pour le comportement d'une entité), etc

2.12.6 Artificial Intelligence (AI)

Artifial Intelligence est un domaine très large dans l'informatique qui tente de mimer l'intelligence animale ou humaine. Les exemples incluent la reconnaissance de paroles (speech recognition), le planning et l'exécution de tâches physiques par les robots.

Genetic Algorithms (GA)

[Goldberg 1989] sont utilisés pour résoudre des problèmes d'optimisation en imitant l'évolution biologique. Des solutions possibles sont codées comme des strings (chromosomes), et une fonction d'évaluation (fitness function) est utilisée pour juger de la qualité de la solution. Une mutation et une fonction de crossover sont appliquées pour créer de nouvelles solutions. La fonction de mutation convertit une solution en une nouvelle. La fonction de crossover spécifie l'échange des parties de chromosome entre un ensemble de chromosomes parents. Le taux de mutation et le taux de crossover

déterminent à quelle fréquence ces opérations surviennent. Pour des problèmes d'optimisation, un group de N solutions candidates initiales sont d'abord générées. Ensuite, chaque solution est notée (assignée à une note) par la fonction de fitness, après quoi N nouvelles solutions sont créées en appliquant les fonctions de mutation et de crossover sur des ensembles de parents séléctionnés de manière aléatoire basés sur le fitness. Le processus continue jusqu'à ce qu'une solution satisfaisante soit trouvée ou qu'un nombre de tour de compte ait été atteint.

Artificial Neural Networks (ANN)

[Haykin 1994] sont des modèles de calcul avec une capacité d'apprendre les relations entre une entrée et un résultat en minimisant l'erreur entre le résultat obtenu et le résultat attendu. Les ANN peuvent être utilisés pour trouver des patterns, et classifier, se souvenir, et structurer des données. Un ANN consiste en d'unités de calcul appelés neurones, qui sont connectés par des arcs valués (weighted edges). Un neurone peut avoir plusieurs arcs entrants et sortants. Quand un neurone recoit une entrée, il combine d'abord les entrées depuis tous les arcs entrants et teste si le déclenchement est issu de cette entrée. Si le neurone est déclenché, il envoie le signal combiné vers toutes les lignes sortantes. L'ANN fonctionne dans un environnement, lequel fournit des signaux d'entrée, traite les signaux de sortie, et calcule l'erreur que l'ANN peut utiliser pour ajuster les poids (weight) des arcs et ainsi apprendre.

Constraint Satisfaction and Planning (CSP)

[Russell et al.1995] consiste à trouver un chemin d'un état initial à un état final en appliquant des actions. Un problème de planning consiste en un état initial, des actions, et un test de but (goal test) [20]. Le Planning Domain Definition Language (PDDL) [21] est communément utilisé pour exprimer des problèmes de planning. Une action peut être exécutée quand la condition initiale est satisfaite. L'effet d'une action peut être l'ajout ou la suppression de variables donnant un nouvel état. Les algorithmes de recherche Forward state-space commençent le planning à partir de l'état initial. En contraste, les algorithmes de recherche backward state-space commençent par l'état final. En dépit de ces deux types d'algorithmes, le planning est NP-hard en général, ce qui explique l'importance des heuristiques dans le planning.

20. goal test ? ?
21. Planning Domain Definition Language ? ?

Septième partie

Comptabilité du capital naturel écosystémique : Trousse de démarrage rapide (2014) de Jean-Louis Weber

Huitième partie

A dipole field object delivery by pushing on a flat surface () of Takeo Igarashi et al.

Cet article introduit un algorithme simple pour la transportation d'objet par un robot pousseur sur une surface plane. Nous assumons que la position globale et l'orientation du robots et des objets sont connus. Le système établit alors un champ de dipole autour de l'objet et déplace le robot le long de ce champ. Cet algorithme simple résout des problèmes subtiles dans l'implémentation de comportements de poussée "confident", comme l'évitement de la collision, l'utilisation de l'erreur, la coordination de plusieurs robots. Nous vérifions l'efficacité de l'algorithme à travers plusieurs expériences avec des robots variés et des objets de formes diverses. Bien que la livraison d'objet par la poussée et le contrôle de mouvement par un champ de vecteur n'est pas nouveau, l'algorithme proposé offre une implémentation plus aisée avec des ajustements de paramètre minimes à cause de sa définition indépendante de la mode (mode-less) et sa formulation indépendante de la taille des objets (scale-invariant).

Le contrôle local : plusieurs approches : 1- méthode binaire : pousser l'objet de l'arrière vers le but (naïve), quand l'objet n'est poussée que d'un côté, il effectue une rotation qui annule les efforts du robot, il faut repositionner le robot à l'arrière pour recommencer la tâche de poussée. (-) le réalignement prend beaucoup de temps, difficile d'assigner la distance excédante (offset) et le seuil de basculement entre les deux modes (pousser-repositionnement). 2- un seul mode : chasing, le robot cherchent continuellement à atteindre une position cible à l'intérieur de l'objet et située un peu derrière le centre de celui-ci. La position cible bouge avec l'objet. Cette technique implique que quelques petits réalignements, seulement il faut choisir soigneusement la distance offset, en prenant en compte la taille du robot et de l'objet 3- dipole : introduit une transition fluide entre l'étape de poussée et celui de se réaligner pour mitiger les problèmes de l'approche naïve.

Neuvième partie

Obstacle avoidance using velocity dipole field method (2005) of Munasinghe et al., ICCAS

Dixième partie

Modeling Landscapes with Ridges and Rivers : bottom up approach (2005) of Farès Belhadj and Pierre Audibert

Cet article décrit une génération procédurale de terrains à partir des chaînes de montagnes et des réseaux de rivières. Elle se déroule en trois étapes : (1) la création de réseaux de chaînes de montagnes et de rivières qui constituent le squelette de départ du terrain sous forme de carte **DEM** (Digital Elevation Map), (2) l'extrapolation préliminaire des coordonnées manquantes à partir d'une **MDI** (Midpoint Displacement's Inverse) pour enrichir les données d'élévation de base (3) suivi d'une subdivision normale **MD** (Midpoint displacement) pour combler les données des points restants.

La génération de chaînes de montagnes se fait en utilisant des paires de particules (r_i, r_{i+1}) placées initialement à une même hauteur y_0, à une position commune P_0 aléatoirement définie, lancées avec des angles initiaux opposés et influencées pour décrire un mouvement "Fractional Brownian" sur le plan (xz). A chaque étape notée par la distance δ et chaque itération i , c'est-à-dire sur la distance parcourue $i \times \delta$, l'altitude de chaque particule diminue suivant une courbe gaussienne $\mathcal{G}_1$ centrée sur la position de départ P_0 dont la déviation est $\sigma = \frac{1}{4} \times largeur(DEM)$. Sur le parcours de la section $[P_i, P_{i+1}]$, un DEM initialement vide stocke les valeurs successives des altitudes de chaque particule. Chaque point $p \in [P_i, P_{i+1}]$ décrit à leur tour une deuxième courbe gaussienne $\mathcal{G}_2(p, \sigma' = \frac{1}{4}\sigma)$ perpendiculaire à la section $[P_i, P_{i+1}]$ avec une amplitude égale à $altitude(p)$. Le processus de traçage de crêtes de montagne pour une particule se termine quand la particule devient statique (altitude nulle) ou qu'elle intersecte le chemin d'une autre particule.

Le réseau de rivières est similairement généré, procédant cette fois-ci avec des particules de rivière, possédant une masse (utilisée pour la génération de la largeur/profondeur) et soumises à la force de la gravité avec une vélocité, et un identifiant. Ces particules de rivières sont disposées aléatoirement sur les sommets des chaînes de montagnes, dont le mouvement est simulé par des modèles de la physique. Chaque position de la particule est stockée dans une liste de coordonnées $path(r_i)$ et sur la carte. Quand une particule r_i croise un chemin $path(r_j)$, il s'arrête et la particule r_j est repositionnée (par backtracking) à l'intersection telle que sa masse soit la somme des masses des deux particules $mass(r_j) = mass(r_j) + mass(r_i)$ et sa vélocité $\overrightarrow{velocity(r_j)} = \overrightarrow{velocity(r_j)} + \overrightarrow{velocity(r_i)}$. Le processus se termine quand toutes les particules sont statiques ou qu'elles dépassent les limites du terrain.

Après les deux premières étapes, seules les coordonnées du squelette du réseaux de chaîne et de rivières sont préservées sur la carte DEM, les autres coordonnées sont réinitialisées. Ils définissent quatre états : l'état *NULL* pour les coordonnées non calculées, l'état RIDGE dont la coordonnée est l'altitude de la particule à cette position, l'état RIVER valué avec l'altitude d'une particule de rivière et l'état DONE indiquant qu'elle est déterminée. Seules les coordonnées à l'état NULL sont à déterminer pour la carte DEM. L'interpolation des données manquantes pour générer le terrain fractale se déroule en deux étapes. La première permet d'éviter des discontinuités sur le modèle de terrain, cette étape utilisant la méthode MDI consiste à trouver les points parents (coins initiaux) des triangles enfants de la subdivision en mettant les parents possibles dans une liste, les données sont enrichies. La deuxième étapes est l'interpolation par subdivision MD pour calculer toutes valeurs des coordonnées à l'état NULL : deux modèles de subdivision ont été testés, *Triangle-Edge* et *Diamond-Square*.

Bibliographie

[1] Ebert, D. S., Worley, S., Musgrave, F. K., Peachey, D., and Perlin, K. 2003. *Texturing & Modeling, a Procedural Approach*. Elsevier, 3 rd edition.

[2] Fournier, A., Fussell, D., and Carpenter, L. 1982. *Computer Rendering of Stochastic Models*. Communications of the ACM, 25(6) :371–384.

[3] Kelley, A. D., Malin, M. C., and Nielson, G. M.. 1988. *Terrain Simulation Using a Model of Stream Erosion*. In SIGGRAPH '88 : Proceedings of the 15 th Annual Conference on Computer Graphics and Interactive Techniques, pages 263–268, New York, NY, USA. ACM.

[4] Miller, G. S. P.. 1986. *The Definition and Rendering of Terrain Maps*. In SIGGRAPH '86, Proceedings of the 13 th Annual Conference on Computer Graphics and Interactive Techniques, volume 20, pages 39–48, New York, NY, USA. ACM.

[5] Musgrave, F. K., Kolb, C. E., and Mace, R. S. 1989. *The Synthesis and Rendering of Eroded Fractal Terrains*. In SIGGRAPH '89 : Proceedings of the 16 th Annual Conference on Computer Graphics and Interactive Techniques, pages 41–50, New York, NY, USA. ACM.

[6] Perlin, K. 1985. *An Image Synthesizer*. In SIGGRAPH '85 : Proceedings of the 12 st Annual Conference on Computer Graphics and Interactive Techniques, volume 19, pages 287–296. ACM.

[7] Smelik, R., de Kraker, K. J., Tutenel, T., and Bidarra, R.. 2010. *Declarative Terrain Modeling for Military Training Games*. International Journal of Computer Games Technology

[8] Smelik, R., Tutenel, T., de Kraker, K. J., and Bidarra, R.. 2008. *A Proposal for a Procedural Terrain Modelling Framework*. In Poster Proceedings of the 14 th Eurographics Symposium on Virtual Environments EGVE08, pages 39–42, Eindhoven, The Netherlands.

[9] Voss, R. F. 1985. *Fundamental Algorithms for Computer Graphics*. chapter Random Fractal Forgeries, pages 805–835. Springer-Verlag, Berlin.