Rapport de documentation de stage

Raoelisolonarivony - MISA M2

5 Février 2017

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

Résultats

Elle y décrit l'implémentation, les outils technologiques utilisés et l'interface du système.

L'implémentation est faite sur Unity3D en relation avec une base de données PostGreSQL via des scripts C#. Une qualité d'Unity3D est de garder les matériaux et les textures intactes pour les modèles 3D importés depuis Blender par exemple. Ensuite, la gestion de ressources d'Unity sous forme d'Assets lui procure la capacité de créer des modèles de terrain et de végétation, en plus de supporter la gestion de collisions grâce à l'intégration du moteur PhysX.

Pour l'interface, l'utilisateur doit modéliser des éléments de la ville : bâtiment, véhicule, etc. et attribuer les fonctions des éléments (fenêtres/portes) sous forme de propriétés personnalisées dans Blender. Un script python effectue la transformation des modèles via Blender. Les modèles de la scène sont décomposés en éléments individuels, lesquels sont inserés dans les tables de la base. Les données sont constituées de coordonnées de multi-polygones, de modèles complets uniformisés réutilisables et des informations géométriques et sémantiques.

Unity3D $\overset{NpgSQL}{\longleftarrow}$ BDD $\overset{APIPython}{\longleftarrow}$ Modèle $\longleftarrow$ Utilisateur

L'utilisateur choisit une région, une liste de villes relative à la région est proposée dans laquelle il en sélectionne une, la génération de la ville avec son style urbain caractéristique se charge. Pour chaque ville, 150 bâtiments sont générés sur une surface définie.

En premier, le terrain et la rue sont créés, puis les bâtiments sur les zones non occupées par la rue, cette dernière divise le terrain en deux parties. Les bâtiments sont génénés par des polygones et des transformations d'objets prédéfinis, placés de manière aléatoire sur un emplacement non occupé, leur orientation est décidée suivant la position de la porte orientée vers la rue, plus un petit aléa de rotation.

A partir des données de la base, les éléments thématiques sont générés et assemblés en entité d'Unity3D pour représenter les bâtiments, les rues, etc. Un mixage des éléments thématiques sous contrainte des dimensions et des positions des éléments permet d'avoir une diversification du paysage urbain. La taille des bâtiments est corrigée pour rentrer dans une fourchette de taille définie sur les espaces disponibles.

L'objet "rue", avec sa position et sa dimension, est récupéré. Les dimensions des véhicules sont proportionelles à la rue. Ils sont positionnés le long de la rue espacés les uns des autres, dans les

deux sens de la file. Chaque véhicule se déplace vers l'avant jusqu'à arriver au bout où il disparaît et réapparait au début de la rue.

Tiavina s'est mise en objectif de générer une ville à partir d'un ensemble d'éléments prédéfinis correspondant au style architectural choisi. Une de ses contributions est la mise en relation de la BDD et la visualisation. Elle a fusionné le standard CityGML et la génération procédurale.

Elle a obtenu la composition de villes avec des éléments avec des styles urbains contrôlables et variés. Les villes sont aléatoires grâce aux variations apportées par l'échange des éléments architecturaux. Le respect des normes établies avec une bonne initialisation lors de la modélisation ont permis d'obtenir son résultat.

Comme amélioration, elle suggère d'automatiser la génération de modèles, ajouter des éléments thématiques par classement (zones résidentielles, zones industrielles, ...) et des règles de placement d'immeubles pour avoir de la cohérence. Il serait bien d'avoir plusieurs rues, actuellement il n'y a qu'une route principale.

2 New generation crowd simulation algorithms (2014) of Julien Pettre and Ming Lin. (suite)

2.1 Etat de l'art

La simulation de la foule du point de vue macroscopique se fait par exemple grâce à la prise en compte de la dynamique des fluides et des champs de vélocités. Seule la contrainte de non-interpénétration est prise en compte pour ces approches. Les points négatifs de l'approche macroscopique sont visibles au niveau local : des agents se déplacant sur le mauvais côté, entrant en collision, subissant des interpénétrations résiduelles.

Pour l'approche microscopique, les contraintes et buts individuels sont plus importants. Le comportement global est alors le résultat qui émerge des interactions locales. Les méthodes pour simuler les interactions locales sont les suivantes :

1. **Discrétiser l'espace en cellules** : (cas des automates cellulaires) les agents sont les cellules occupées. Les interactions sont modélisées comme les règles de transition probabiliste entre les états. Des règles interdissent aux agents 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. Le **modèle de Boids** par exemple permet aux agents de décider de leur mouvement en respectant des règles :
   — séparation : éviter le surpeuplement et les attroupements
   — alignement : prendre une direction vers un but commun
   — cohésion : ne pas se séparer du groupe

Un autre exemple reposent sur les lois physiques, les agents sont soumis à des forces dites "sociales" (**SF** ou Social Forces model). Le premier avantage de ce modèle est qu'il permet d'éviter les collisions complexes, et ensuite de modéliser des phénomènes tels que l'amitié et les forces attractives.

3. Les **modèles basées sur la vélocité** sont des évolutions des modèles SF, appelés aussi modèles prédictives. Ils traitent plus d'informations et 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 la plus convenable pour atteindre l'objectif.

4. Le **Dynamic Window** est une méthode utilisée par les robots pour éviter les collisions : un algorithme d'exploration glouton explore les vélocités en associant un poids à chacune, une possible

collision correspondant à un poids élevé. Ce groupe de méthode a inspiré les méthodes basées sur la vélocité.

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.

6. Le modèle **RVO (Reciprocal Velocity Obstacle)** de Van den Berg fait intervenir deux agents avec un espace de vélocités relatives. Les vélocités entrainant une collision dans une certaine fenêtre de temps sont rendues indisponibles, formant un obstacle de vélocités.

7. La méthode **Linear Trajectory Avoidance** parcours un ensemble de mouvements possibles et associe un coût à chacun. Le coût est estimé en extrapolant les trajectoires suivant les positions et les vélocités. Cette technique est utilisée dans le suivi par vidéo des piétons réels comme un prédicteur pour le traqueur de vision par ordinateur, et donne des résultats satisfaisants, spécialement pour les problèmes d'occlusion.

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 en utilisant les positions relatives aux espaces personnels des agents, une décision est faite sur lequel des agents passe en premier et lequel cède le passage.

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'appuient sur les angles de portée et l'approximation du **ttc** (time-to-collision) pour prendre des décisions. Ce modèle est capable d'intéragir avec n'importe quel obstacle statique ou en mouvement aussi longtemps que l'agent peut percevoir l'obstacle.

Ces modèles sont à la base de plusieurs modèles développés récemment.

La validation expérimentale des modèles basés sur la vélocité

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 $25m \times 25m$, 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 minimum (appelée MPD) à laquelle ces deux personnes sont supposées se rencontrer sans se soucier d'éviter la collision est notée pour chaque paire 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. Les observations suivantes se dégagent de ces expériences :

(i) Le contrôle de la distance MPD commence aussitôt que les participants se voient
(ii) Cette distance ne cesse de croître positivement (une croissance monotone)
(iii) Cette distance MPD n'est pas exagérément augmentée, aucune manoeuvre superflue
(iv) Cette manoeuvre s'arrête aussitôt que la collision est évitée.

L'existence 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.

2.2 Trois examples de modèles basées sur la vélocité

L'article décrit, discute et compare trois solutions pour les modèles basées sur la vélocité dévoloppées par L'Inria :

(i) le modèle de Paris

(ii) le modèle "Tangent"

(iii) le modèle "Vision"

On rappèle que l'objectif est de définir un ensemble de vélocités sans risque de collision avec des obstacles statiques ou en mouvement (vélocités admissibles) et ensuite 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 de 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} et 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 voisin. 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 un 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 pas de sens de traiter s_{min} pour ce cas.

Le modèle "Tangent"

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 obtenue (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 et 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.

Ce modèle permet l'évitement des obstacles statiques avec des géométries décrites comme des ensembles de segments (les immeubles par exemples). 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.

Le modèle "Vision"

Le modèle "Vision" se base sur la superposition des données des vitesses angulaires et des **ttc** (time-to-collision) qu'un agent peut avoir avec ses voisins. L'espace est subdivisée en pixels, chaque pixel p_i possède une paire de propriétés (vitesse angulaire/ttc) notées $(\dot{\alpha}, ttc)$ qui varient à chaque instant. Sur la base de ces propriétés, deux domaines spécifiques pouvant conduire à des collisions sont différenciés : le premier est le domaine des risques de collisions imminentes, les cas des ttc_i proches de zéro (des agents très proches). Les collisions peuvent être éviter par l'agent en réduisant la vitesse. Le deuxième est le domaine des risques de futures collisions, les cas des $\dot{\alpha}_i$ proches de zéro (des agents sur le front). Ces futures collisions peuvent être éviter en changeant d'orientation. La combinaison de ces deux réactions forme un ensemble de processus itératif "perception-action" de la part de l'agent concerné. Ce modèle peut gérer différentes formes d'obstacles. Etant donné qu'il considère les pixels, la visibilité des obstacles est implicitement pris en compte. Il est capable de simuler l'émergence de patterns connus de mouvements des piétons et des conditions de traffic routier.

2.2.1 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 toutes les 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 sélectionne 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 demeurent sans réponse sur les modèles basés sur la vélocité, les modèles concernant le comportement de suivi d'agent. Comment combiner, par exemple, les comportement de suivi et d'évitement pour simuler des comportements complexes de groupes, crucial pour les foules simulées de manière naturelle.

Une simple extrapolation linéaire de trajectoires basée sur la position et la vélocité est-elle une bonne prediction ? spécialement pendant les manoeuvres : dans des places bondées où les gens

redéfinissent constamment leur mouvement, les modèles basés sur la vélocité sont-ils inutiles ?

2.3 Applications

2.3.1 L'industrie du cinéma et le design architectural (Golaem)

Le logiciel Golaem Crowd est un outil développé pour l'animation de personnages destiné aux artistes pour peupler leur scène. Golaem est une société fondée en 2009 par des chercheurs et ingénieurs de l'Inria.

Le logiciel prend en charge la génération de tous les éléments visuels, le placement de personnages, l'animation procédurale de créatures n-ped, la navigation de personnages individuels ou en groupes, les interactions physiques. L'outil permet également d'intéragir avec des logiciels de rendus via des plugins et offre un éditeur de comportement.

Pour gérer toutes ses situations, le logiciel utilise des algorithmes de navigation suivant les modèles Social Force, Reciprocal Velocity Obstacles et leur propre algorithme d'extrapolation. Ces algorithmes ont beaucoup d'impacts sur l'animation de personnage dans les grosses productions cinématographiques.

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

La simulation de foule basée sur la vélocité est adaptée aux créatures des jeux vidéos. Les framework d'évitement d'obstacles sont très demandés dans la production de jeux. La simulation du phénomène de foule réel et des comportements chorégraphiques demandent un design de jeu basé sur de tels frameworks. La qualité de perception de la locomotion des personnages de jeux s'en trouve améliorée.

3 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 (**MD** : 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 nombres pseudo-aléatoires pour le déplacement de point de milieu. Ainsi, ils associent les triangles disposés sur une même altitude en piochant le nombre dans la table de hachage.

Les mathématiques des fractals de Mandelbrot et la recherche de chemin simplifié dans un graphe planaire 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 méthode : les rivières trop applaties, 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).

4 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 de crête 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 étape 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*.