Génération procédurale de bâtiments

Alexandre Brochu

24 / 04 / 2015

Département d'informatique
Université de Sherbrooke
Sherbrooke (Qc), Canada, J1K 2R1
Alexandre.Brochu@usherbrooke.ca

- Rapport de recherche no 1 -

Résumé

le travail relié à la création de modèles 3D de villes pour créer un réalisme dans un environnement urbain dans les films ainsi que dans les jeux vidéo devient de plus en plus complexe. On recherche des façons d'automatiser ce travail à l'aide d'algorithmes qui se basent sur la génération procédurale. Ici, on parle d'un de ces algorithmes un peu plus en détail.

Table des matières

Liste des figures

1 Mise en contexte

On met en contexte en ce moment. De nos jour, il est possible de créer de films et des jeux vidéo qui donnent toujours un meilleur effet de réalisme au fil des années. Ce réalisme provient de l'amélioration constantes des techniques utilisées lors de la création de ces médias. Un des aspects qui peut beaucoup influencer le réalisme est la modélisation des objets dans la scène. L'industrie tente toujours d'ajouter plus de détails aux modèles 3D utilisés dans le projet en ajoutant toujours plus de points et de triangles pour obtenir des formes qui imitent le mieux possible la forme réelle. Souvent, on concentre les efforts surtout sur les détails dans les personnages principaux mais travailler sur le réalisme de l'environnement peut aussi donner de bon, sinon meilleur résultat.

FIGURE 1.1 – Des exemples d'environnements urbains dans les jeux GTA IV et Assassin's Creed

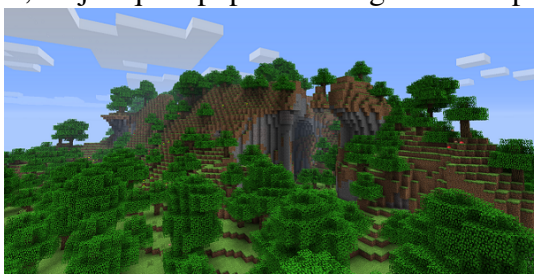
Puisqu'on améliore le réalisme de l'environnement autour des personnages, il faut ajouter plus d'information aux formes de la scène. De plus, l'environnement doit toujours être plus grand pour donner un meilleur sentiment d'immersion. Il y a un problème en ce sens puisque plus il faut d'information pour spécifier des formes dans la scène, plus il faut beaucoup de temps pour les artistes de créer cette information. Aussi, créer des environnement toujours plus grand demande aussi beaucoup de temps. Un moment donné, ce n'est plus envisageable de demander à des gens de faire ce travail. Il faut donc trouver une façon d'utiliser la technologie de façon intelligente pour faire ce travail automatiquement ou semi-automatiquement. Si on regarde la figure 1.1, on voit qu'il y a beaucoup trop de détail dans l'environnement pour que quelqu'un ou une équipe spécifie la position de tous les points de ce modèle. Ce jeu se déroule dans un environnement urbain c'est pourquoi l'étude se poursuit pour créer des modèles d'un environnement urbain. Pour obtenir un bon résultat, il est important d'avoir un bon sentiment d'immersion. Pour cela, il faut un bon réalisme ainsi qu'une bonne variété dans les bâtiments de la ville. Par exemple, les images des jeux "Assassin's Creed" et "GTA IV" sont des bons exemples d'environnements réussis. La question ici est, comment utiliser la technologie pour arriver a de bons résultats de façon automatisée ?

2 Présentation de l'état de l'art

Il existe des méthodes pour aider à la création d'environnement urbains très grands. Ces techniques sont basés sur la génération procédurale.

2.1 Génération Procédurale

FIGURE 2.1 – Minecraft, un jeu qui a popularisé la génération procédurale récement [5]



La réponse à la question est la génération procédurale. Cette technique a été beaucoup popularisé par un jeu vidéo indie nommé Minecraft (voir figure 2.1). Par contre, il existe beaucoup de variantes à la génération procédurale. L'article étudié se concentre sur la génération procédurale de bâtiments pour générer un environnement urbain réaliste. Pour commencer, le fonctionnement de cette technique est d'utiliser des algorithme qui s'exécutent sur l'unité de traitement d'un ordinateur pour générer un élément. On ne spécifie pas directement le résultat de la génération ici parce qu'il est possible de générer pratiquement n'importe quel élément grâce à cette technique. Elle offre aussi la possibilité d'ajouter du hasard dans la génération pour ne pas obtenir toujours le même résutlat à chaque exécution. Elle est beaucoup utilisée puisqu'il est possible d'ajouter de la variété et de la rejouabilité dans les jeux de nos jours. Par contre, au départ, cette technique était beaucoup utilisée pour économiser de l'espace lors de la création de jeux. Puisque certains éléments étaient générés lorsque le jeu est en marche, il y avait moins de ressources à conserver en mémoire. Le premier exemple de génération procédurale se trouve dans le jeu Akalabeth (1980) où les cartes étaient générées [4]. La génération procédurale a beaucoup été popularisée par l'arrivée du jeu Minecraft. Certains jeux en font même un usage très intéressant. Par exemple, dans le jeu "Left 4 Dead", on génère la difficulté de la mission courante d'après les statistiques des joueurs ainsi que les équipements qu'ils ont sur eux [1]. On remarque donc que l'on peut générer n'importe quel élément grâce à cette technique.

Avant l'évolution de la technique élaborée dans l'article étudié, deux autres façon de générer des bâtiments de façon procédurale existaient.

2.2 Première Technique

Cette technique a été crée pour générer des grandes villes avec un minimum de données statistiques et géographiques. Elle voulait aussi offrir un grand contrôle aux usagers [8]. Cette technique se sert du

principe des ”L-system”. Cette technique tient aussi compte de la création de routes pour connecter les différents bâtiments. Par ocntre, cette technique ajoute des détails sur les façades des bâtiments générés à l'aide de nuanceurs. Ces chercheurs ont montré une méthode pour générer un grand environnements urbains composés d'un grand nombre de bâtiments. Cependant, il y a un problème. Les détails que ajoutés aux façades des bâtiments peuvent causer des intersections dans les modèles ce qui peut enlever l'effet de réalisme recherché.

2.3 Deuxième Technique

Ici, les chercheurs ont voulu s'attaquer au problème de la génération de détails sur les façade des bâtiments générés. Ils ont réussi à créer des détails qui sont en fait générés à l'étape géométrique. Ces détails sont donc compris dans le modèle résultant et il n'est pas nécessaire d'utiliser des nuanceurs pour ajouter du réalisme aux façades des bâtiments. Par contre, avec cette technique, on s'attarde aux détails d'un seul bâtiment à la fois. Il n'est donc pas possible de générer un grand environnement urbain comme dans la première technique.

On remarque que l'on désire obtenir un mélange de ces deux techniques. On désire un grand nombre de bâtiments dans notre ville et que ces bâtiments possèdent des détails géométriques pour augmenter le réalisme. Nous allons maintenant nous pencher sur la technique principale de l'article. On précisera aussi en quoi cette méthode est meilleure que les autres.

3 Méthode Choisie

La technique utilsée ici se base sur les grammaires de formes. La première partie de ce rapport se penche plus précisément sur le fonctionnement de base d'une grammaire. Cette technique ,comparée aux deux autres présentées au cours de ce rapport, génère les détails des bâtiments de façon géométrique par opposition aux nuanceurs. Il est donc possible de vérifier s'il existe des intersection entre les détails générés et les annuler puisque ces intersections enlèvent beaucoup de réalisme à une scène. Pour la deuxième technique présentée, l'avantage ici est que cette technique permet la création de grands environnements en gardant un temps d'exécution raisonnable. On présente donc quelques éléments de base sur lesquels l'algorithme est basé et plus tard, on montre les différentes étapes de l'algorithme plus en détail.

3.1 Théorie

3.1.1 Une grammaire

Dans le domaine informatique, une grammaire est définie par deux ensembles de symboles, un ensemble de règles et un symbole de départ. Le but d'une grammaire est de passer du symbole de départ et utiliser les règles pour passer à un état où il n'y a que des symboles terminaux [2]. Le premier ensemble de symbole sont les symboles terminaux. Ces symboles servent à marquer la fin d'une série de changements sur un symbole. Un symbole terminal ne sera plus modifié par les règles de la grammaire.

Ensuite, il y a l'ensemble des symboles non-terminaux. Ces symboles représentent des état transitifs qui seront modifiés lors de des itérations d'exécution de la grammaire. Pour finir, on définit les règles de la grammaires. Ces règles ont un format spécifique : non-terminal $\rightarrow$ serie de terminaux et non terminaux. Cette règle commence par annoncer sur quel symbole non-terminal il va agir et en quoi ce symbole sera transformé. On applique itérativement une règle à la fois sur la chaine de symbole courante jusqu'à obtenir une chaîne qui n'a que des symboles terminaux.

3.1.2 Une grammaire de forme

Dans le domaine de génération procédurale de bâtiment, on utilise la même idée de base mais on adapte les principes au rendu 3D. Par exemple, les symboles terminaux et non-terminaux sont des formes de base [6]. Aussi, les règles donnent les transformation géométriques qu'il faut faire et les formes qu'il faut ajouter pour obtenir une forme terminale pour terminer l'exécution de la grammaire de forme. Voici quelques exemples de règles dans une grammarie de forme [7] :

$$A \rightarrow [T(0,0,6)S(8,10,18)I(\text{"}cube\text{"})]T(6,0,0)S(7,13,18)I(\text{"}cube\text{"})T(0,0,16)S(8,15,8)I(\text{"}cylinder\text{"})$$

$$fac \rightarrow Subdiv(\text{"}Y\text{"}, 3.5, 0.3, 3, 3, 3)\{floor|ledge|floor|floor|floor\}$$

On peut aussi spécifier des règles pour créer des sous-sections de formes géometriques. Généralement, on donne une séries de longeurs (les différentes séparations) et sur quel axe (x, y ou z) on fait la séparation. On peut utiliser ces séparation pour créer des détails sur les façades des bâtiments qu'on génère en insérant d'autres modèles alignés sur ces séparations.

3.1.3 Les portées

Un autre élément important de l'algorithme utilisé pour cette technique sont les portées. Une portée est une partie de l'espace 3D. Les transformations spécifiées dans les règles de production sont affectées sur les portées [7]. Après les transformations géométriques, si on insère un nouveau modèle, il sera placé dans la portée et prendra la dimension et la position de la portée. On sait donc que chaque forme dans une configuration de l'algorithme possède une portée. De plus, il existe aussi une utilité pour des portées qui représentent un espace 2D. Ces portées peuvent être utilisées pour générer un toit sur un bâtiment de base. On défini une portée en 2D aligné sur un des murs du bâtiment ou le dessus de la forme et on applique des transformations et on insère des nouvelles formes pour obtenir un toit sur notre structure de base.

3.1.4 L'algorithme

Pour commencer, une configuration est le nom qu'on donne à un nombre fini de formes dans l'espace 3D. Aussi, dans notre cas, un symbole est représenté par une forme et ces symboles peuvent être soit non-terminaux ou terminaux. La définition plus précise des symboles se trouve plus haut dans ce document. Au départ l'algorithme doit recevoir une spécification d'un axiome qui est une configuration de formes simples pour démarrer la génération. On procède ensuite par étapes [7] :

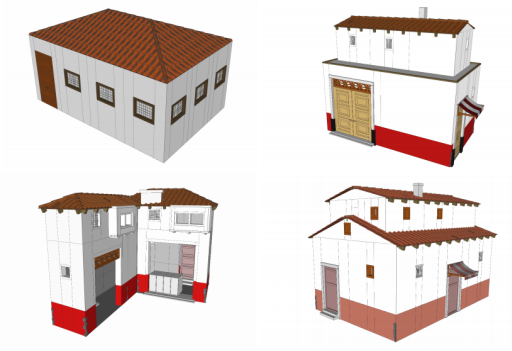
1. On sélectionne une forme (état) non-terminale qui est toujours active. À titre d'exemple, ce symbole est nommé n
2. On choisit ensuite une règle de production de la grammaire de forme à appliquer. Il faut que cette règle soit composée du symbole n comme partie gauche. Après on génère (à partir de la spécification de la règle choisie) un nouvel ensemble de formes. On nomme ce nouvel ensemble nNew
3. On place la forme n comme inactive dans la liste des formes de la configuration courante. Pour finir on ajoute l'ensemble de formes nNew à la configuration et on recommence à l'étape no.1.

3.2 Exemples

Les possibilités de génération sont pratiquement infinie. L'article étudié se penche plus en détails sur types d'exemples. Les trois premiers exemples montrés dans l'article sont volontairement simples pour montrer la base de l'algorithme. Ils montrent 3 types de constructions : une maison simple, un bloc d'édifices à bureaux et une maison un peu plus poussée. Par après, les exemples montrés sont plus complexes et montrent une utilisation plus poussée de l'algorithme. Le premier exemple plus complexe est une extension du 3e exemple qui assemble plusieurs éléments pour créer un quartier de banlieue. Ensuite, le dernier exemple de l'article montre la génération d'une ville entière basée sur le style de Pompeii.

3.2.1 Exemple 1

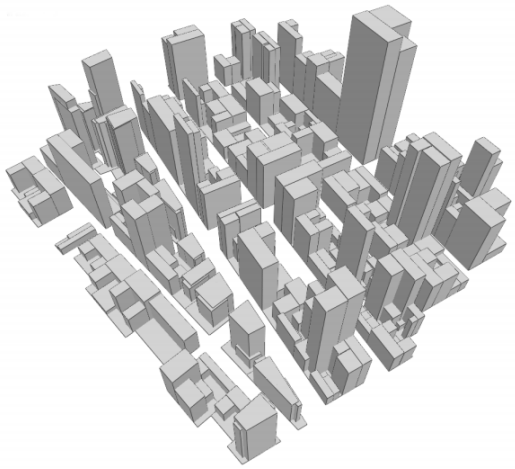
FIGURE 3.1 – Une maison simple [7]



Cet exemple montre la façon de créer la structure la plus simple avec cet algorithme pour le corps principal de la maison et la forme la plus simple de toits. Les murs sont simplement construit à partir d'une forme deux dimensions avec laquelle une extrusion a été faite pour obtenir un prisme en trois dimensions. Pour ce qui est du toit, il est construit à partir deux 2 plans en 2D qui se rencontrent au milieu du prisme de la structure du bâtiment.

3.2.2 Exemple 2

FIGURE 3.2 – Un bloc d'édifices a bureaux [7]



Cet exemple tente de montrer qu'il est possible de générer plus qu'un seul bâtiment à la fois à l'aide de cet algorithme. L'algorithme utilise encore des formes 2D comme une base et l'extrusion de ces formes donne la forme générale des bâtiments. La distribution des formes 2D au départ définissent la densité de la ville que l'on veut obtenir. Par exemple, dans la figure 3.2 il y a une grande densité de grand bâtiments. Aucun détails n'a été générer sur les façades des bâtiments dans cet exemple puisque les détails seront montrés dans les exemples suivants.

3.2.3 Exemple 3

FIGURE 3.3 – Une maison de banlieue plus évoluée [7]

Dans cet exemple, on ajoute des détails autour de la maison. Par exemple, on ajoute une cour, un stationnement, et de la végétation. Ces détails sont générés de la même façon que les éléments de base. On ajoute aussi des trottoirs autour de la propriété.

On peut aussi utiliser plusieurs maisons de ce type et les rassembler pour créer un quartier de banlieue (figure 3.3). Ce rassemblement ne veut pas seulement dire qu'on execute plusieurs fois l'algorithme et qu'on combine les modèles résultants en un seul. On peut tout simplement modifier les règles de l'algorithme pour faire en sorte qu'il soit possible de générer plus qu'un seul bâtiment, mais conserver les règles qui ajoutent des détails de végétation et des trottoirs. De la même façon, on peut générer une ville plus grande en utilisant les règles de l'exemple 2 comme une base.

4 Conclusion

Voilà une courte explication du fonctionnement de l'algorithme présenté dans l'article étudié. L'algorithme offre des possibilités de génération infinie. La puissance de l'algorithme provient de la définition des règles. Comme prouvé dans l'exemple de la ville dans le style de Pompeii [7]. Cette technique de génération a été développée pour combiner les avantages de deux autres techniques qui ont été mises au point auparavant. De plus, il est facilement possible d'ajouter des éléments de hasard dans la génération. De cette façon, on peut créer plus d'une ville avec la même série de règles.

Il reste quelques aspects de l'algorithme qui n'ont pas été couvert dans l'article étudié au cours de la session. Par exemple, la performance possible de l'algorithme ainsi que la parallèlisation de ce dernier. Il aurait été intéressant de voir quelles parties de l'algorithme sont parallèlisable et quel facteur d'amélioration il est possible d'obtenir avec la version parallèle de l'algorithme.

Pour finir, il existe un outil qui utilise cet algorithme pour faire une génération de modèles de villes automatiquement [3]. Dans le vidéo d'introduction de cet outil, on remarque la plupart des sujets qui on été abordés dans ce rapport. On remarque aussi que la performance de l'algorithme est assez bonne puisque les résutlats de l'algorithme sont rendus en temps réel.

Bibliographie

[1] The ai systems of left 4 dead. http://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/. Visite : 2015-04-24.

[2] Context-free grammar. https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html. Visite : 2015-04-24.

[3] Esri cityengine. http://www.esri.com/software/cityengine. Visite : 2015-04-24.

[4] Historic rpg akalabeth : World of doom free on gog. http://segmentnext.com/2014/12/24/historic-rpg-akalabeth-world-doom-free-gog/. Visite : 2015-04-24.

[5] How procedural generation took over the gaming industry. http://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/. Visite : 2015-04-24.

[6] Introduction to shape grammars. http://home.fa.utl.pt/~albertidigital/AD_VER1011/files/aula05/aula05_shapeGrammars.pdf. Visite : 2015-04-24.

[7] Procedural modeling of buildings. http://info.usherbrooke.ca/ogodin/enseignement/imn538/articles/NON%20DISPONIBLE%20-%20Procedural%20Modeling%20of%20Buildings.pdf. Visite : 2015-04-24.

[8] Procedural modeling of cities. http://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/. Visite : 2015-04-24.