Génération procédurale de mondes virtuels : Scène de quartier enneigé

Olivier BERTHIER

Janvier 2013

Résumé : Ce projet a été réalisé dans la perspective de créer une architecture de classes permettant la génération d'une scène crédible en trois dimensions d'un quartier de ville enneigé en utilisant le framework Arches développé par l'équipe GeoMod. Cette architecture devra minimiser la consommation de mémoire tout en maximisant la précision des détails.

Mots-clés : 3D, neige, crédible, modélisation, primitive, procédurale, ville, Arches

Table des matières

1 Introduction

C'est dans le cadre du projet de Recherche du Master 1 Informatique de Lyon 1 que j'ai rejoint l'équipe GeoMod pour participer à ses travaux. Cette équipe travaille actuellement sur un framework nommé Arches dont l'objectif est de générer des scènes 3D probables en temps réel avec une excellente précision des détails. Au cours de ce TER j'ai été encadré par Eric Galin professeur à l'Université Lumière Lyon 2 et François Grosbellet un de ses doctorants, tous deux membres de l'équipe.

La modélisation réaliste d'un manteau de neige est un des nombreux défis de ce framework d'autant que celui-ci devra être généré en temps réel dans une scène procédurale, c'est à dire que lorsqu'un objet de la scène est modifié la neige doit automatiquement s'adapter au nouveau décor.

L'objectif de ce TER sera donc de réaliser une architecture de classes permettant de modéliser une scène de quartier d'une ville enneigée en utilisant les fonctionnalités de ce framework. La scène final ainsi que les primitives qui la composent devront être réutilisables dans d'autres scènes en tant que primitives. Il faudra ensuite optimiser cette arborescence de classes de façon à utiliser le moins de mémoire possible.

2 État de l'art

Depuis plusieurs années, plusieurs travaux ont été réalisés présentant différentes méthodes de génération de neige dans une scène virtuelle. Deux méthodes sont souvent utilisées :

- La méthode de simulation par particules, qui est très coûteuse surtout lorsqu'il s'agit de prendre en compte l'environnement (vent, collisions).
- La méthode de déplacement de surface, particulièrement simple et peu coûteuse mais peu réaliste.

L'efficacité de ces méthodes se perd un peu plus lorsqu'il s'agit d'obtenir un rendu réaliste dans des scènes complexes (forêts, villes). En effet, même si nous partons du principe que la même quantité de neige est tombée sur toute la scène, certaines surfaces sont soit recouvertes par divers obstacles diminuant la quantité de neige sur celle-ci (arbre, grille), soit sont trop petite pour toute cette neige puisse s'y accumuler (bordure de balcon, dossier de banc, branche d'arbre).

Dans [FG09] une solution est proposée pour ces problèmes. Elle permet de réguler la hauteur de neige sur un objet en fonction de la hauteur de neige tombée en utilisant la courbure que l'on peu constater sur les éléments dépassant du manteau de neige. Une courbure inverse est utilisée pour les surfaces couvertes. Ce travail permet d'obtenir un rendu rapide et plus agréable grâce à l'adoucissement des bords, mais qui manque parfois de réalisme.

Dans [FG11] une autre méthode mathématique basé sur les équations de diffusion est proposée permettant de résoudre d'avantages ces problèmes. De plus Cette méthode permet de gérer les ponts de neige, ce qui augmente fortement le réalisme (exemple : sur une table constituée de plusieurs planches espacées) (cf. figure 1). L'utilisation d'un masque alpha lors de l'application des textures permet d'améliorer le rendu des limites de la neige (cf. figure 2). Cette approche permet d'avoir plus de réalisme, mais est plus coûteuse en ressources.

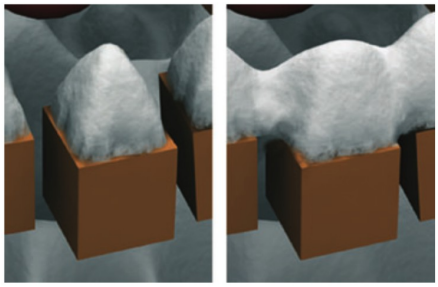


Figure 1 : Comparaison d'une scène sans et avec pont de neige

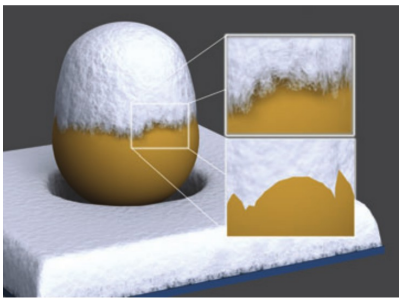


Figure 2 : Comparaison de rendu avec et sans application du masque alpha

Enfin le travail réalisé dans [MGGMM10] permet de pousser un peu plus loin le réalisme, en attribuant certaines propriétés physiques aux éléments d'une scène. Par conséquence la neige va fondre plus rapidement sur certains objets, ou lorsqu'elle se trouve exposée à une source de chaleur (cf. figure 3). Il s'agit dans ce travail d'une simulation très longue et coûteuse, qu'il n'est donc pas possible de l'utiliser dans Arches.

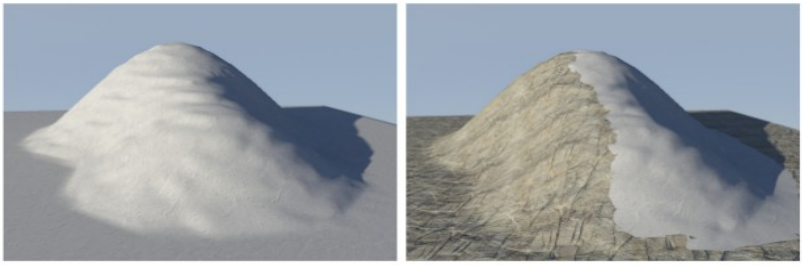


Figure 3 : Influence du soleil sur une scène enneigé avec reliefs

3 Le framework Arches

3.1 Présentation du framework

Arches est un framework de génération procédurale qui permet de créer et de manipuler des scènes en trois dimensions construites à base de primitives. L'objectif de ce framework est d'arriver à générer des scènes crédibles en temps réel en obtenant un niveau de détails de 10^{-2} m à 10^{3} m de distance. Le défit est de taille, car de nombreux phénomènes sont très complexe à modéliser (par exemple la neige et les rivières). Jusqu'à présent les résultats les plus réalistes étaient soit le produit d'un infographe qui aura dû intervenir manuellement pour modifier la scène, soit le résultat d'une simulation lourde en mémoire et en CPU qu'il est impensable d'utiliser en temps réel même avec l'évolution constante de la puissance de calcul et de la quantité de mémoire dans les ordinateurs. D'un autre côté les méthodes les plus rapides et les moins lourdes en mémoire ne donnent pas un résultat assez satisfaisant.

Ce framework utilise l'API OpenGL, ce qui permet exploiter directement la puissance de calcul de la carte graphique pour les rendus. Il est possible d'exporter les scènes sous un format compatible avec Maya afin d'obtenir un rendu de la scène après y avoir ajouté des textures. Ce projet est essentiellement développé en C++, le gestionnaire de version SVN est utilisé afin de permettre le travail collaboratif et préserver un historique des différentes versions de chaque fichier.

3.2 Modélisation du manteau de neige

Le fonctionnement de ce module dans une scène va beaucoup dépendre des primitives utilisées et de l'arborescence des classes. Pour ajouter de la neige sur une primitive il faut définir manuellement son emplacement utilisant des primitives de neige prédéfinis telle que les « SnowQuad », les « SnowDisk » ou les « SnowSpheres » qui permettent respectivement de placer la neige sur un quad, un disque et une sphère.

Pour pouvoir générer la neige sur une primitive composée de plusieurs primitives il suffit d'utiliser la fonction permettant la génération de la neige pour chacune de ses sous-primitives. Si les surfaces de neige sont correctement placées, l'architecture de construction du maillage des primitives étant semblable à celle de la neige, lorsque l'on modifie un paramètre de la scène la neige va automatiquement s'adapter aux nouvelles surfaces. Toutes les classes héritant d'une même classe, la classe « Node », il n'y a pas de conflits possibles par rapport aux noms des fonctions utilisées pour la génération du maillage et de la neige. Pour augmenter le réalisme et ne pas avoir de coupure net dans la neige, il faut définir les bords des primitives de neige qui seront adoucis vers le haut ou vers le bas (cf. figure 4).

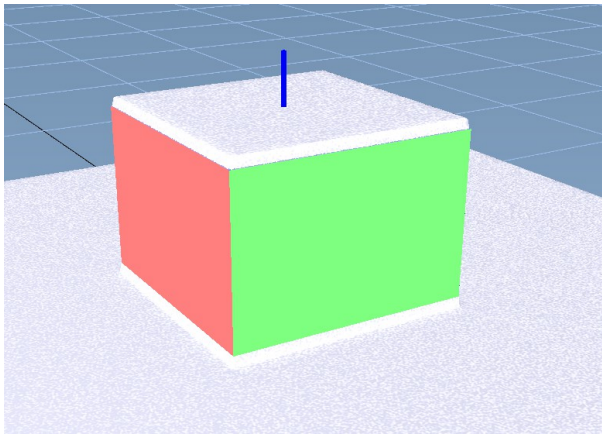


Figure 4 : Adoucissement positif contre les parois du cube et négatif sur bords du cube

Enfin le framework permet de définir des zones d'occlusion où la neige sera moins épaisse à cause des obstacles qui se trouvent au-dessus (par exemple sous une table, un arbre ou une grille) (cf. figure 5). Il est aussi possible d'intégrer des zones de chaleur où la neige sera moins épaisse due à sa fonte plus rapide.

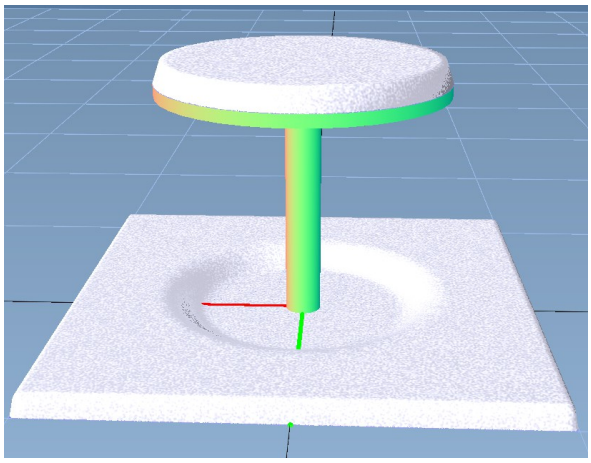


Figure 5 : Illustration d'occlusion

4 Réalisations

4.1 Problématique

L'objectif du module de modélisation de neige du framework Arches est de générer une scène crédible avec un excellent niveau de détails en temps réel en utilisant un minimum de mémoire. Ce module est encore en phase de développement, cependant il est essentiel pour pouvoir l'optimiser de le mettre à l'épreuve dans divers situations. Ce projet permettra d'avoir une nouvelle scène pour de futures tests ou utilisations du framework.

Ses performances de rendu ne dépendant pas que du module de neige, il faudra aussi trouver une architecture de classes qui permette cela. Il sera donc proposé de développer une scène de neige dans un quartier du 18ème arrondissement de Paris. Le rendu de cette scène devra être procédurale, c'est à dire qu'en modifiant des paramètres l'aspect de la scène doit changer et toutes les primitives doivent s'y adapter automatiquement.

4.2 Mise en œuvre

4.2.1 Phase d'analyse

La première étape consiste à schématiser la scène que l'on souhaite modéliser. Il faut pour cela procéder par étapes en partant d'un très faible niveau de détails vers un niveau de détails de plus en plus élevé. Il faut aussi, en particulier dans le cas de d'une scène de quartier enneigé, trouver un modèle à respecter, ceci permet d'avoir des mesures crédibles après quelques recherches sur le site du cadastre. La scène choisie est un escalier dans une rue de Montmartre à Paris (plus précisément le sommet de la rue du Mont Cenis) (cf. figure 6).

On identifie 5 éléments principaux :

- L'escalier
- Le haut de l'escalier
- Le bas de l'escalier
- Les immeubles de droites
- Les immeubles de gauche

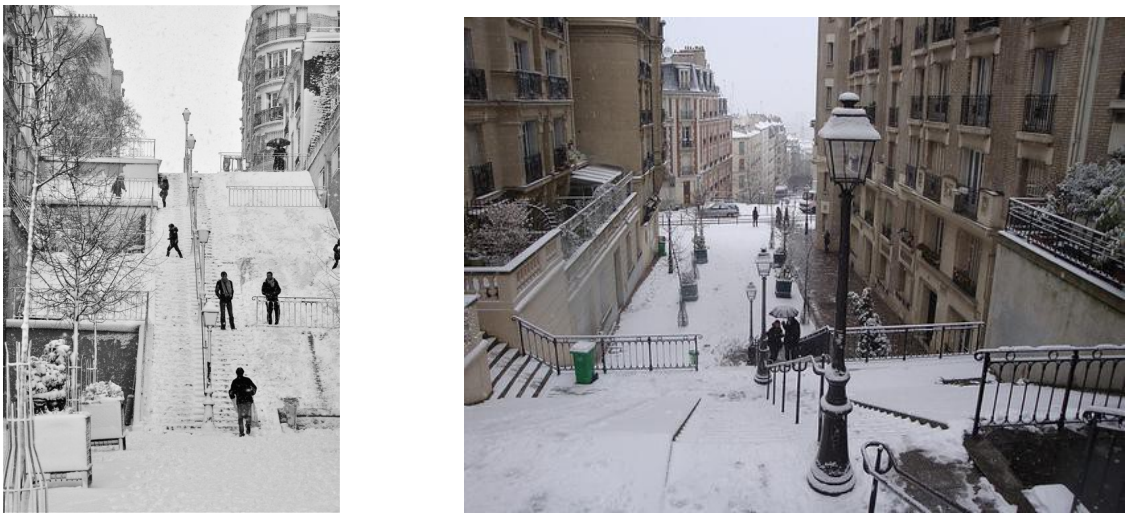


Figure 6 : Photos de la scène réalisée vue du bas et du haut (sources http://www.flickriver.com et http://pascalinedargant.blogspot.fr)

Pour mon travail je me suis surtout concentré sur l'escalier. Celui-ci se compose de quatre niveaux différents, chacun comprenant un escalier au centre et un espace à droite et à gauche. La zone centrale de chacun de ces niveaux est constante, il sera donc possible d'utiliser la même classe. Pour ce qui est des bords de l'escalier il y a des variantes, il faudra donc une classe pour chacune d'entre elles (cf. figure 7). L'étape suivante est de détailler chacun de ces sous-éléments. A la fin on se retrouve avec une arborescence de classes, chacune d'elle permettant de modéliser une primitive de l'escalier.

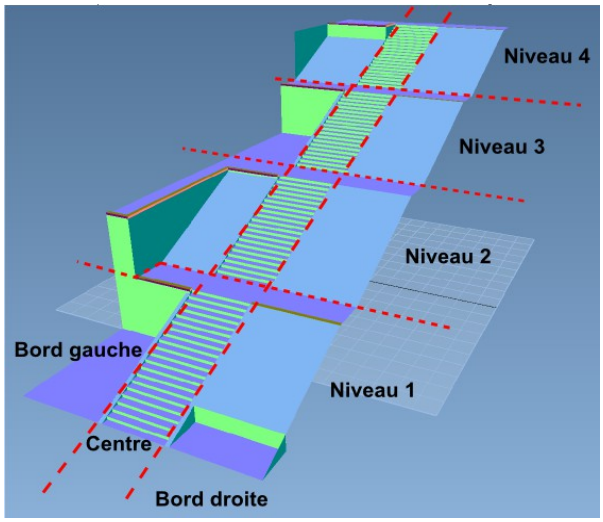


Figure 7 : Découpage de l'escalier

L'étape suivante est de définir les paramètres des primitives de la scène qu'il sera possible de modifier. Certains primitives resteront statiques dans la mesure ou leur modification n'apporterait pas grand chose à la scène, comme la largeur des barrières dans ce cas. En revanche il est importants de pouvoir modifier certains paramètres comme le nombre de marches, le nombre de niveaux ou la largeur des escaliers.

Dans cette scène, la gestion des façades et des bordures de droite ne pose pas trop de problèmes, mais ce n'est pas le cas pour ceux de gauche. En effet le relief est assez irrégulier et le premier et le deuxième niveau ne doivent pas être séparés. La figure 8 et les rendus de l'annexe 4 illustrent le rendu du même

escalier avec une simple modification de quelques paramètres montrant également la gestion des niveaux.

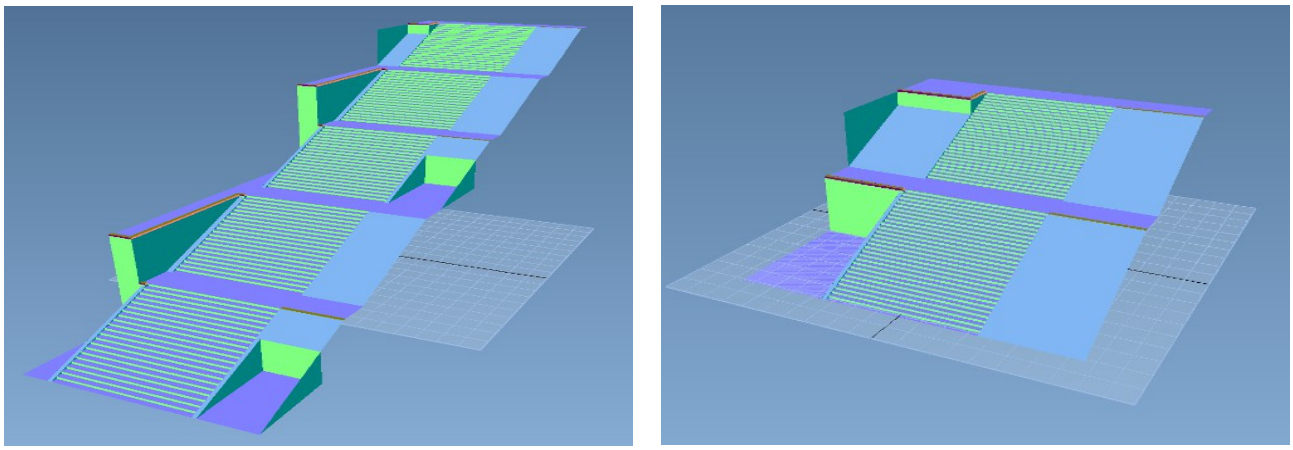


Figure 8 : Rendu de l'escalier avec 5 niveaux puis avec 2 niveaux

4.2.2 Phase de développement

La principale idée qu'il faut préserver tout au long du développement est la réutilisation de ce qui a été fait et de rendre possible la réutilisation de la scène en cours de développement comme primitive. Il faut aussi faire attention à toujours placer le point d'entré de l'objet à un endroit stratégique facilitant sa manipulation, en générale il s'agit du centre de la primitive en X, Y et Z.

Chaque primitive est réalisée en surchargeant certaines fonctions de la classe « Node » dont elles héritent toutes. La première de ces fonctions est « QueryShapeMesh », dans laquelle il faut créer le maillage de la primitive en utilisant des primitives de base ou celles existantes. A ce stade on se retrouve avec une scène nue. Cependant il s'agit d'une étape très importante car les futures méthodes surchargées, en particulier pour le placement de la neige, dépendront fortement du maillage.

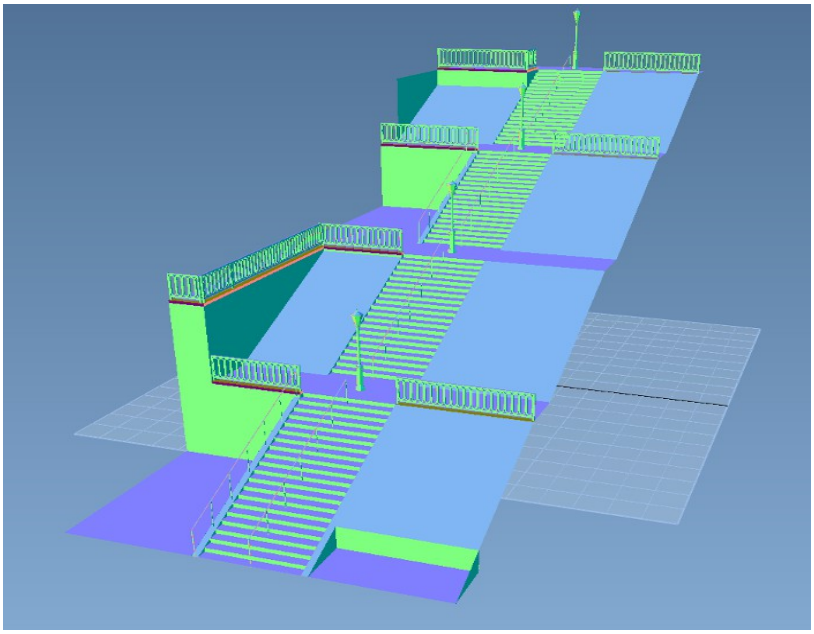


Figure 9 : Vue de l'escalier avec détails.

Ensuite le niveau de détails est augmenté au fur et à mesure du développement (cf. figure 9), il n'est cependant pas nécessaire d'aller à un niveau de détails trop élevé dans un premier temps, surtout qu'il s'agit d'une scène de neige ou certains éléments seront recouverts.

Une fois que le maillage est posé, il est possible de mettre en place les primitives de neige. On supposera au cours du projet que la quantité de neige sera la même partout et qu'aucun phénomène physique tel que le vent ou la pluie n'ont eu lieu. Ces primitives de neige sont initialisées en surchargeant la méthode « QuerySnowMesh » dans chacune des classes définies précédemment. En générale il suffit de reprendre toutes les surfaces non verticale et d'y placer une ou plusieurs primitives de neige adaptées à la surface. Il faut découper la surface en plusieurs morceaux si une ou plusieurs primitives se trouvent sur cette surface afin d'en définir le contour (sans cela une partie de la surface de neige serait calculée inutilement et cela permet d'arrondir la neige le long de la paroi de la primitive). Il faut ensuite appeler cette même fonction pour chacune des primitives qui la compose. Le module de neige n'étant pas encore optimisé, cela augmente plus ou moins fortement le temps de rendu, en particulier lorsqu'il s'agit de grandes surfaces ou de sphères. Il vaut mieux procéder élément par élément au cours de cette étape pour gagner du temps. L'adoucissement des bords ou des surfaces de neige en contact avec d'autres primitives permettra un rendu plus crédible (cf. figure 10).

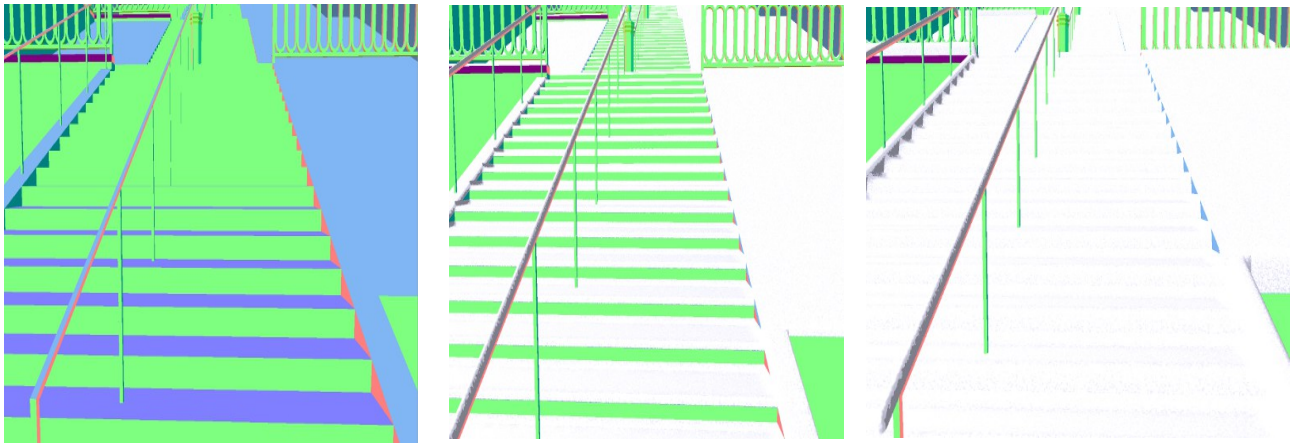


Figure 10 : Vue de l'escalier sans neige, avec 4 cm de neige et 20 cm de neige

On observe sur la figure 10 qu'au plus la quantité de neige est importante, au plus la pente de l'escalier s'aplatit. Ce phénomène est lié à l'adoucissement, sur le bord extérieur de chaque marche on va constater un arrondissement vers la marche du dessous, alors que contre la paroi intérieur on aura un arrondissement vers la marche du dessus (cf. figure 11). A partir d'une certaine quantité de neige toutes les marches seront jointes et la surface de neige tendra à s'aplatir (cf. figure 12).

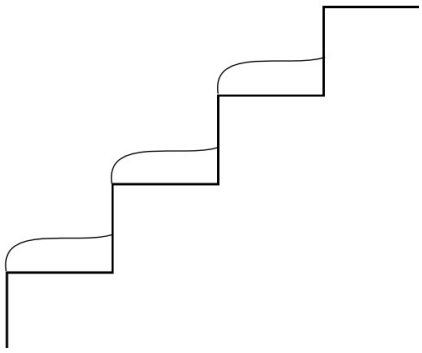


Figure 11 : Schéma d'un escalier avec peu de neige

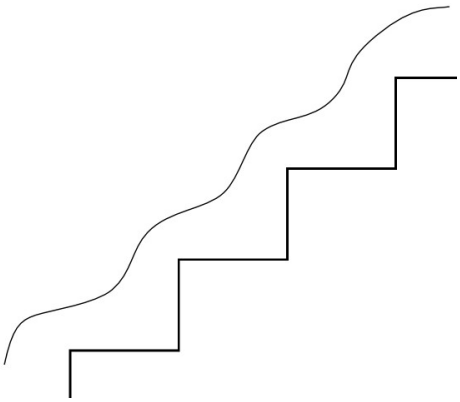


Figure 12 : Schéma d'un escalier avec beaucoup de neige

En fait il ne s'agit pas d'un problème qui ne se limite qu'aux escaliers, mais qui se retrouve à de nombreux endroits dans la scène (pour les murets par exemple). Dès lors que l'on dépasse une certaine quantité de neige on constate un décalage. Il faut donc faire attention à bien adapter les primitives de neige pour ne pas obtenir de cassure comme illustré en rouge sur la figure 13. Le trait vert représente une quantité de neige inférieur à la taille du muret, le trait bleu représente le résultat de rendu attendu en cas de grosse quantité de neige.

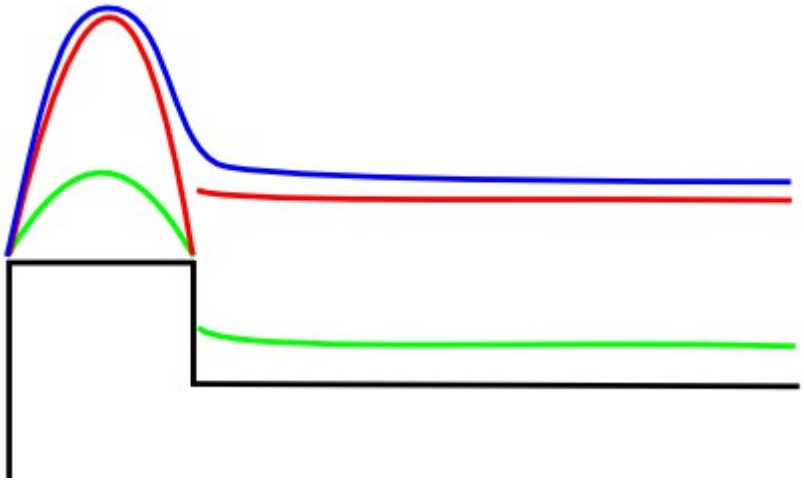


Figure 13 : Illustration du problème d'adaptation de la neige en fonction de la hauteur de neige

Enfin on procède de la même façon que pour l'intégration de la neige pour mettre en place les zones d'occlusion et de chaleur. Il suffit de surcharger certaines méthodes avec des objets de type OcclusionEmitter et TempEmiters et d'appeler les méthodes correspondantes pour chacune de ses composantes, du moins lorsque c'est nécessaire.

La façon la plus efficace de debuger une primitive est de lui faire subir divers transformations et en la plaçant dans une autre scène, cela permet de mettre en évidence des problèmes de décalage qui n'apparaissent pas lorsque la primitive se trouve au centre de la scène.

Pour la scène final plusieurs primitives déjà existantes ont été réutilisées notamment les façades des immeubles, les barrières et les lampadaires. Les paramètres configurables lors du rendu concernent essentiellement l'escalier. Cependant le reste du décor s'adapte aussi par rapport à ces paramètres, par exemple lorsque l'on modifie la longueur ou la largeur des escaliers les façades des immeubles ainsi que le haut et le bas des escaliers vont s'adapter aux nouveaux paramètres.

4.2.3 Phase d'optimisations

L'optimisation est l'étape finale de ce TER. L'objectif du module neige du framework étant de générer en temps réel des scènes de neige en utilisant un minimum de mémoire, il y a forcement beaucoup de travail à faire pour optimiser cela du côté de l'architecture des classes utilisées. Concernant le maillage il faut toujours chercher à réutiliser au maximum les mêmes primitives. Une fois utilisée, celle-ci est stockée en mémoire avec un identifiant. Si ce même identifiant est de nouveau demandé le même objet sera utilisé par la carte graphique. En utilisant cette propriété il est possible d'économiser de la mémoire et du temps de calcul. Il faut donc vérifier que la fonction permettant de contrôler si l'identifiant de la classe n'a pas déjà été référencé dans la scène soit présent dans les fonctions que l'on a surchargé dès que nécessaire.

Une autre optimisation à faire concerne la neige. Il faut diminuer au maximum le nombre de primitives de neige. C'est entre autre pour cela que le découpage des niveaux des escaliers a été fait en vue de face plutôt qu'en vue de dessus, cela permet de diminuer légèrement le nombre de primitives de neige à poser.

5 Conclusion

Le résultat de la scène 3D d'un quartier d'une ville enneigée réalisée au cours de ce projet de recherche de Master 1 me semble assez satisfaisant (cf. annexe 1, annexe 2 et annexe 3). L'ensemble des contraintes du cahier des charges ont été respectées. Il est possible de réutiliser la scène en tant que primitive tout en modifiant son apparence en modifiant les paramètres appropriés et des optimisations ont été faite pour utiliser moins de mémoire. Je pense cependant qu'avec un peu plus de temps le résultat aurait été meilleur. Plus de primitives auraient été ajoutés afin d'augmenter la crédibilité de la scène. Il aurait aussi été possible de passer plus de temps pour mieux optimiser les classes et pour réparer toutes les cassures mineurs qui apparaissent en cas de forte quantité de neige évoqué dans le chapitre 4.2.2.

Je pense que le fait que le module de neige ne soit pas encore optimisé a pu ralentir ma progression, la scène final mettant près de 2 minutes à s'afficher cependant cela ne m'a que peu ralenti car je travaillais la plus part du temps sur de petites primitives.

J'avais déjà eu l'occasion de faire des scènes en 3D avec Blender auparavant, ce TER m'aura permis d'apprendre une nouvelle façon de les modéliser en programmant la scène en C++, ce qui pourra m'aider pour le choix de mon Master 2. Enfin ce projet m'aura permis d'avoir un aperçu du travail réalisé dans les laboratoires de recherche en informatique.

6 Références

[FG09] N. v. Festenberg et S. Gumhold, A Geometric Algorithm for Snow Distribution in Virtual Scenes, 2009

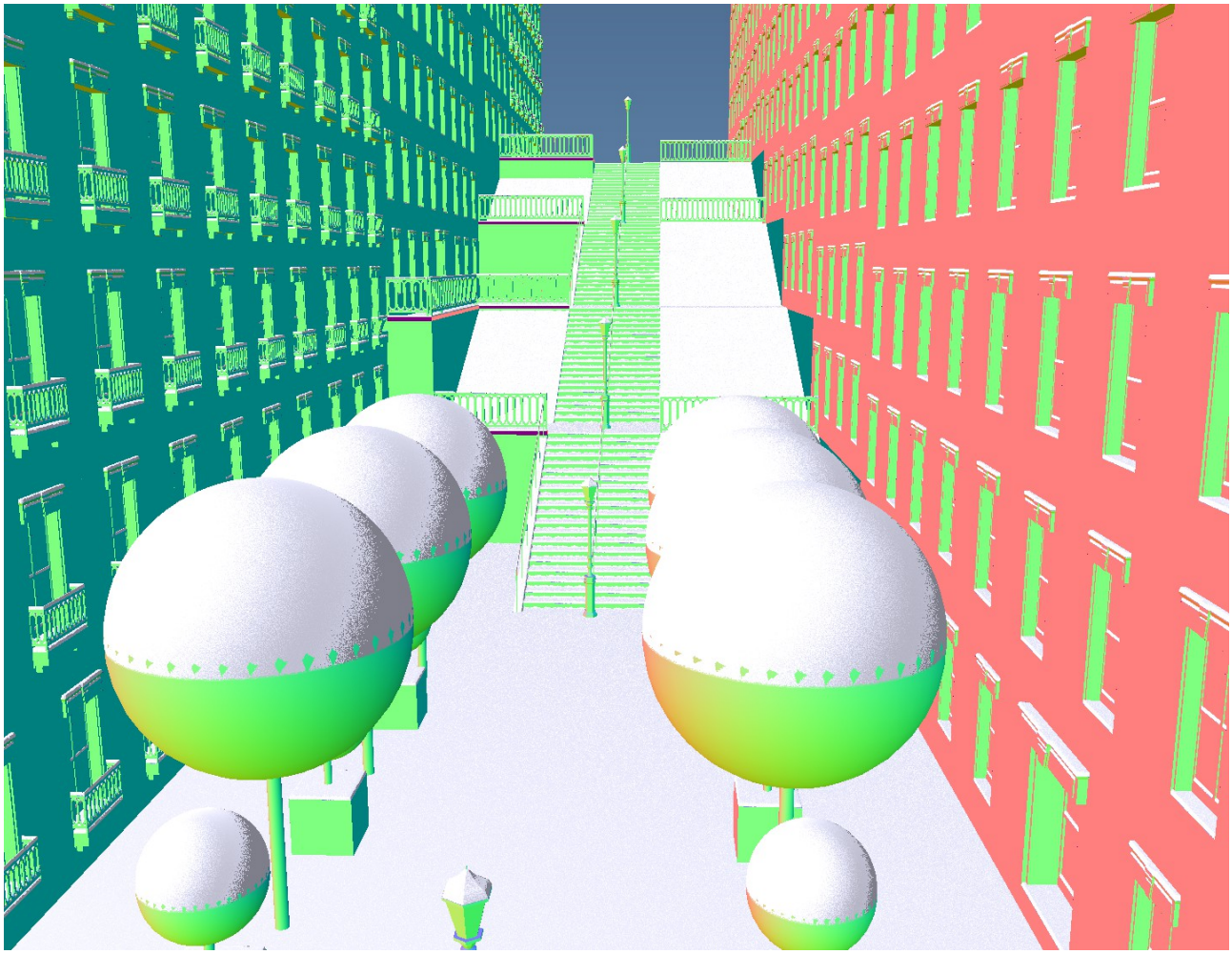
[FG11] N. v. Festenberg et S. Gumhold, Diffusion-Based Snow Cover Generation, 2011

[MGGMM10] N. Maréchal, E. Guérin, E. Galin, S. Mérillou, N. Mérillou, Heat Transfer Simulation for Modeling Realistic Winter Sceneries, 2010

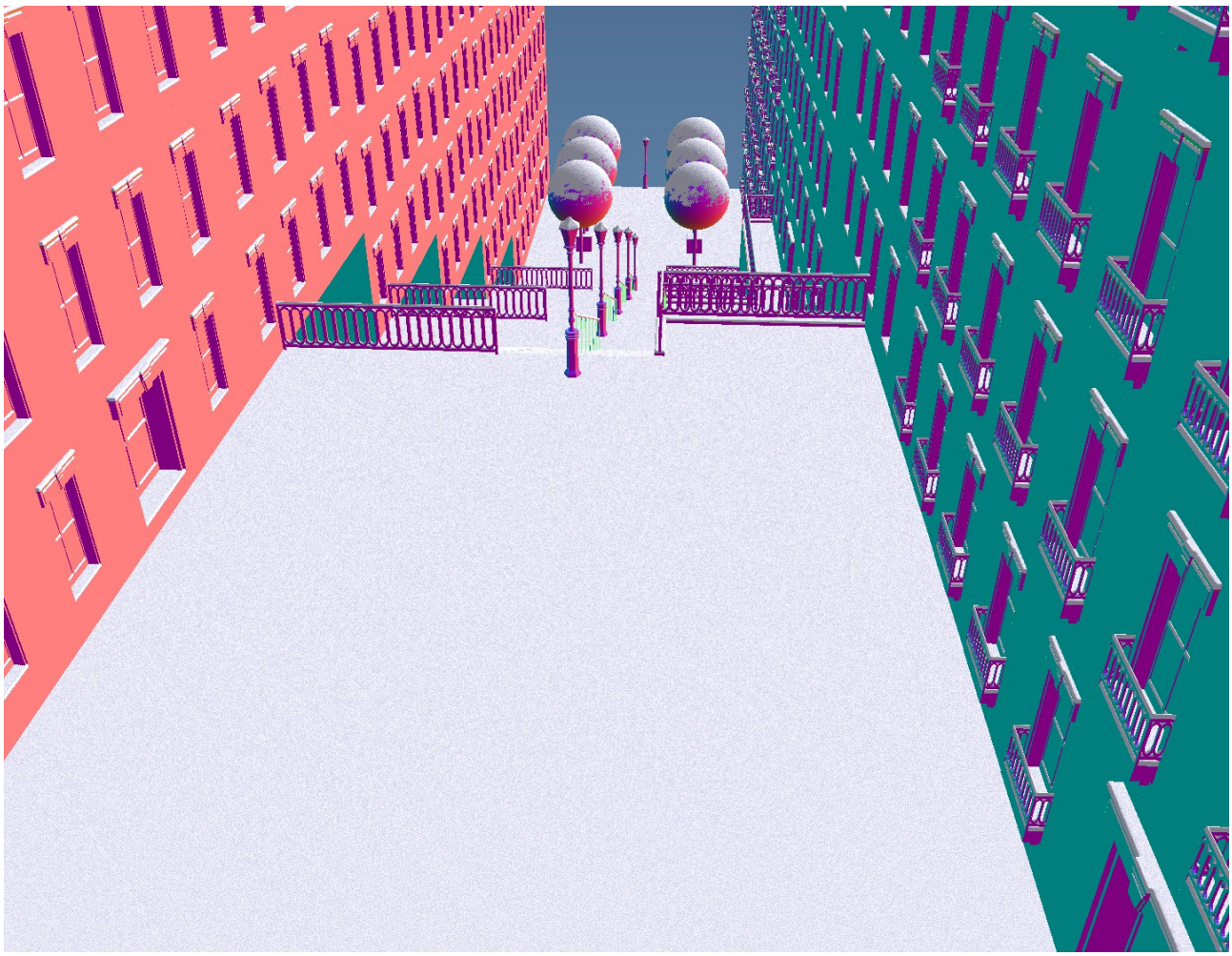
7 Annexes

7.1 Annexe 1 : Rendus de la scène sous Maya

7.2 Annexe 2 : Rendu depuis le bas de l'escalier



7.3 Annexe 3 : Rendu depuis le haut de l'escalier



7.4 Annexe 4 : Rendus procédurales de l'escalier