Génération procédurale de monde

Adrien Peytavie

$N°112 - 2010$ **Année 2010**

UNIVERSITÉ CLAUDE BERNARD – LYON 1

UFR INFORMATIQUE

École doctorale "Informatique et Information pour la Société" (EDIIS) de Lyon

THÈSE DE L'UNIVERSITÉ DE LYON

présentée et soutenue publiquement le 7 juillet 2010

pour l'obtention

du DIPLÔME DE DOCTORAT

(Arrêté du 7 août 2006)

Spécialité Informatique

par

Adrien Peytavie

Génération procédurale de Monde

Composition du jury

Rapporteurs :		
	M^{me} Marie-Paule Cani	Professeur, INP de Grenoble
	M. Jean-Pierre Jessel	Professeur, Université de Toulouse
Examinateurs :		
	M^{me} Ming Lin	Professeur, University of North Carolina
	M. Jean-Michel Dischler	Professeur, Université Louis Pasteur de Strasbourg
	M. Samir Akkouche	Professeur, Université Claude Bernard Lyon 1
Directeur :		
	M. Éric Galin	Professeur, Université Lumière Lyon 2

Laboratoire d'InfoRmatique de l'Image et Systèmes d'information (LIRIS) – UMR CNRS 5205

Table des matières

Introduction

La génération de monde est une thématique importante en informatique graphique car centrale et récurrente dans l'ensemble de ces filières que sont la modélisation, l'animation et la visualisation. De nombreuses applications découlent de ces travaux de recherche comme la création de scènes pour l'industrie du cinéma et du jeu vidéo, la simulation, la planification de territoire ou encore la cartographie.

Générer un monde consiste à définir une scène composée d'une multitude d'éléments de différents types qui interagissent entre eux pour obtenir un résultat le plus réaliste possible. Durant les 30 dernières années, de nombreux travaux ont été menés pour résoudre les verrous scientifiques et techniques. Afin de diminuer la complexité, cette thématique a été divisée en trois domaines principaux qui sont la génération de terrains, la création de la végétation et la modélisation du milieu urbain. En divisant le problème de la génération de monde en trois sous domaines de recherche, les différents travaux traitant de problèmes spécifiques et ciblés comme la modélisation d'un arbre ou encore la définition du relief du terrain ont pu aboutir à des résultats très réalistes. Cependant, cette approche n'a pas permis de générer des éléments qui interagissent entre eux pour s'adapter à leur environnement.

Dans le chapitre 1, nous présentons un état de l'art des techniques de modélisation qui peuvent être classées en trois catégories : la génération procédurale, la simulation et la modélisation par esquisses ou par édition. La génération procédurale permet de créer très rapidement un nombre important d'élément à l'aide d'un ensemble de procédures. Cette catégorie est utilisée pour modéliser des scènes de très grandes tailles de plusieurs dizaines de kilomètres comme un terrain ou une forêt. La simulation est utilisée pour obtenir des résultats réalistes pouvant évoluer au cours du temps. Cette approche utilise des lois physiques, chimiques ou biologique qui sont longues et coûteuses à évaluer. Les équations sont appliquées soit sur des petites scènes de quelques mètres avec une discrétisation fine, soit sur de grandes scènes de faible résolution pour traiter un phénomène spécifique. La modélisation par esquisses ou par édition est utilisée pour la création d'éléments pour des scènes de grandes tailles ou de tailles intermédiaires. Dans ce cas le temps de modélisation nécessaire pour obtenir un résultat réaliste est long et le travail des graphistes est très fastidieux. Les scènes de petites et de grandes échelles

sont définies avec une faible variété d'élément car la précision de modélisation ne permet pas d'en contenir d'avantage. Par contre, les scènes de tailles intermédiaires nécessitent de contenir un nombre important d'éléments variés qui sont définis en fonction de leur environnement pour obtenir un résultat réaliste.

Actuellement, très peu d'approches existent pour générer du contenu graphique ayant un très grand niveau de détail et de réalisme pour des scènes de tailles intermédiaires. Dans cette thèse, nous proposons des techniques hybrides afin de répondre à cette problématique. Notre approche consiste à combiner l'ensemble des techniques de modélisation (procédurale, simulation, édition) afin d'optimiser la génération de ces scènes et pour obtenir un résultat le plus réaliste possible. Pour créer nos scènes, nous modélisons la géométrie de nos éléments jusqu'à une précision de l'ordre de dix centimètres, les éléments plus petits étant considérés comme de la texture. Nos méthodes permettent de créer des scènes complexes composées de différents éléments qui interagissent entre eux. L'ensemble de nos contributions est présenté dans trois chapitres.

Dans le chapitre 2, nous proposons une représentation originale pour créer des terrains de formes complexes en définissant l'ensemble des caractéristiques de la scène. Notre approche permet de modéliser des surplombs, des arches, des grottes et les nombreux éléments composant la scène sous la forme de couches. Ces couches servent à manipuler les caractéristiques du terrain sur les différentes échelles simplement en considérant les éléments comme un ensemble continu. Des outils intuitifs sont proposés aux utilisateurs afin d'éditer l'ensemble des caractéristiques de la scène. Ces outils de haut niveau permettent de contrôler l'emplacement des éléments de la scène sans avoir à ce soucier des détails qui sont automatiquement gérés par les modèles de simulation et de génération procédurale. La simulation est utilisée pour manipuler automatiquement l'ensemble des couches à chaque édition de l'utilisateur pour obtenir un résultat plus réaliste. La simulation reproduit les phénomènes physiques, mécaniques ou biologiques de notre scène comme la stabilisation de matériaux. Nous utilisons les modèles de génération procédurale afin de créer automatiquement les détails géométriques de la scène par exemple pour générer l'empilement de milliers de pierres.

Dans le chapitre 3, nous proposons un modèle de génération d'empilement de pierres pour ajouter automatiquement des détails à une scène. Notre modèle consiste à créer la géométrie de l'ensemble des pierres et de les positionner dans la scène. Nous utilisons des modèles procéduraux afin de distribuer les instances de pierres dans la scène par une méthode de pavage apériodique. Ce modèle permet de déposer rapidement une grande quantité d'instances dans la scène. Nous appliquons une simulation d'érosion lors de la création de la géométrie des pierres afin d'augmenter le réalisme de l'empilement. La forme de chacune des pierres et de l'empilement est contrôlée par plusieurs paramètres.

Enfin dans le chapitre 4, nous proposons un modèle de génération de routes de campagnes s'adaptant aux contraintes définies par les caractéristiques de la scène. Ce processus consiste à trouver une trajectoire entre un point initial et un point final. Cette génération s'effectue sur une scène à grande échelle tout en nécessitant une grande précision de modélisation pour créer les éléments de la route. Pour définir la trajectoire, nous effectuons une recherche de plus court chemin anisotropique sur

l'ensemble du terrain discrétisé. La trajectoire de la route est contrôlée par des fonctions pondérant les caractéristiques du terrain. La génération de la géométrie de la route est obtenue à l'aide de modèles procéduraux. La route est ensuite intégrée au décors par une succession d'opérations de terrassement et d'excavation du terrain et les éléments comme le bitume, les ponts, les tunnels et les barrières sont obtenus par des fonctions procédurales paramétrées.

Pour l'ensemble des modèles que nous proposons, nous utilisons les avantages de chaque technique de modélisation afin d'optimiser, de simplifier et de rendre plus réaliste nos scènes. La simulation est utilisée pour alléger les étapes d'édition tout en augmentant le réalisme de la scène. Les techniques de génération procédurales permettent d'obtenir rapidement une grande quantité de données pour augmenter les détails de la scène automatiquement. Ces modèles sont associés à des paramètres de contrôle qui ont un sens sous-jacent afin de permettre une édition intuitive et naturelle de la scène.

Chapitre 1

Etat de l'art



Sommaire

La génération de mondes virtuels est un vaste domaine de recherche contenant de nombreux défis. Un des principaux challenges consiste à modéliser l'ensemble des éléments graphiques composant notre monde en prenant en compte l'ensemble des interactions entre les différents éléments. Afin de réduire la complexité de ce problème, les méthodes proposées génèrent indépendamment chacun des éléments. Reeves et Blau (RB85) ont proposé un processus de génération de monde en plusieurs étapes pour éviter les conflits lors de la création. Le processus suit un modèle hiérarchique en partant du plus gros élément aux plus petits. Ce processus a été repris par Chiba (CMDH97) et se décompose en trois étapes :

1. Modélisation du terrain qui sert de structure à l'ensemble des autres éléments.
2. Création des instances des éléments de plus petite taille comme les arbres et les bâtiments.
3. Définition des emplacements de chaque instance. Pour les végétaux, cela consiste à définir la position de chaque arbre dans la scène. Pour les paysages urbains, cela consiste à délimiter les emplacements des bâtiments d'après un réseau de rues.

Nous pouvons classer les techniques existantes en trois grandes catégories (Figure 1.1) : les méthodes d'édition et d'esquisse, les techniques de simulation et la génération procédurale. Certaines méthodes sont parfois à l'intersection de différentes catégories et combinent plusieurs avantages pour s'adapter aux contraintes de modélisation fixées par le problème posé.

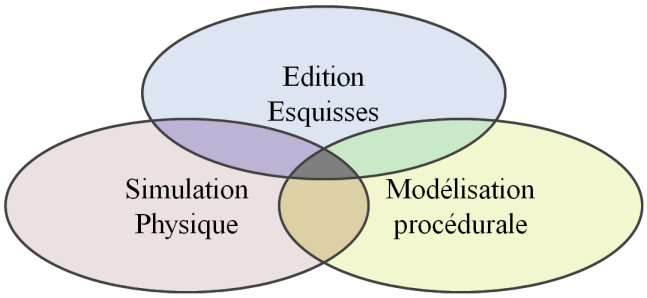


FIGURE 1.1 – Techniques de modélisation des éléments.

Les méthodes à base d'édition ou d'esquisses permettent de contrôler précisément la forme des éléments. Elles consistent à en modifier itérativement la surface afin de contraindre l'objet vers une forme souhaitée. Ce type de modélisation demande une forte intervention de l'utilisateur et s'avère longue et coûteuse lors de la création d'un grand nombre d'objets. Malgré ces inconvénients, cette méthode est la plus utilisée dans les industries du film et des jeux vidéo.

La génération procédurale permet de modéliser rapidement et automatiquement un nombre important d'éléments en définissant un ensemble de procédures et avec une intervention de l'utilisateur réduite. Ces procédures sont contrôlées par des paramètres soit définis en entrée par un utilisateur, soit calculés aléatoirement dans un intervalle également spécifié par l'utilisateur.

Enfin, les méthodes de simulation physique permettent de modéliser un élément évoluant au cours du temps. Ces méthodes reproduisent des phénomènes physiques, chimique, biologique appliqués à un objet pour obtenir un résultat réaliste. Le résultat final ne peut pas être contrôlé car il dépend uniquement des équations physiques. La création d'un élément est long et coûteux car elle nécessite de résoudre de nombreuses équations physiques.

La plupart des méthodes existantes sont aux frontières de ces catégories afin d'en combiner les différents avantages. Ces techniques consistent à diminuer la contrainte d'une catégorie par des approximations du modèle. Par exemple, afin d'obtenir des temps de génération interactifs, les équations physiques de simulations sont simplifiées.

L'état de l'art que nous proposons suit le processus de génération de monde (RB85, CMDH97). Nous détaillons tout d'abord les méthodes de création de terrain, ensuite la génération de la végétation et pour finir la génération de paysages urbains. Nous détaillons principalement les travaux portant sur la modélisation et la génération des objets et nous ne présenterons pas les méthodes de visualisation.

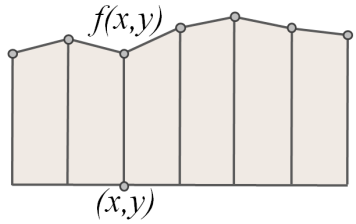
1.1 Création de terrain

Le terrain est l'élément le plus important de la scène car il sert de support à la génération de l'ensemble des éléments de la scène. De ce fait, le terrain est le premier élément devant être modélisé. De nombreux travaux ont été réalisés pour créer, manipuler et visualiser des terrains. Dans cette partie, nous commençons par présenter l'ensemble des structures de données mises en place pour définir différents types de terrains. Ensuite, nous présentons les processus utilisés pour générer ces terrains.

1.1.1 Structures de données

De nombreuses structures de données ont été proposées pour définir un terrain. Ces structures de données permettent de représenter différents types de terrain ayant des caractéristiques qui leurs sont propres. Les structures de données proposées essaient de trouver un compromis entre les différentes caractéristiques qui sont le volume des données, la simplicité de manipulation et la complexité et variété de la donnée.

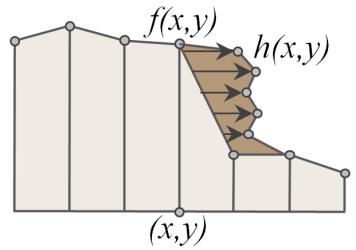
1.1.1.1 Cartes d'élévation



Une carte d'élévation représente le terrain comme une grille régulière à deux dimensions (MKM89, LH04). La carte est définie comme une fonction de hauteur f qui associe à chaque nœud (x, y) de la grille une altitude $f(x, y)$. Cette représentation permet de connaître en tout point de l'espace la hauteur du terrain. Cette représentation est la plus utilisée en informatique graphique car elle est simple et compacte. De plus, cette structure peut être représentée sous la forme d'une image où la couleur définie l'altitude du terrain, ce qui permet d'optimiser certaines opérations en utilisant les capacités des cartes graphiques actuelles (NWD05). Les opérations de modification du terrain sont également simplifiées. La génération de la carte consiste uniquement à créer une fonction de hauteur en tout point de l'espace. Une limitation de ce modèle est de ne pas représenter des terrains avec des caractéristiques géologiques complexes comme les surplombs, les arches ou encore les grottes. En pratique, cette structure est utilisée uniquement pour des terrains dont la pente est inférieure à 30°. Lorsque la pente est supérieure à cette limite, des artefacts d'étirements du maillage et de la texture peuvent apparaître.

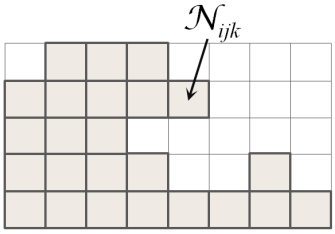
Un empilement de cartes d'élévation (BF02) permet d'obtenir un modèle plus évolué et de définir les différentes couches de matière composant le sol comme la roche, le sable et la terre. Cette représentation a été utilisée pour représenter les couches de sédiments produites par les phénomènes d'érosion.

1.1.1.2 Cartes de surplombs



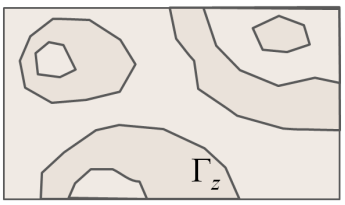
Gamito et Musgrave (GM01) ont proposé une structure de données originale pour créer des surplombs à partir des cartes d'élévation de données. Cette technique applique des déformations horizontales $h(x, y)$ sur la surface initiale représentée par une carte d'élévation. La déformation de la surface du terrain est guidée par un champ vectoriel défini par un utilisateur sur l'ensemble de la scène. La nouvelle surface est définie comme la fonction $h(x, y)$ à deux dimensions pour modéliser le champ vectoriel. Cette méthode permet de créer des surplombs mais ne permet pas de modéliser des arches ou des grottes qui sont des caractéristiques plus complexes nécessitant de modifier la topologie du terrain. Une limitation supplémentaire est la difficulté d'édition du champ vectoriel $h(x, y)$.

1.1.1.3 Énumération spatiale



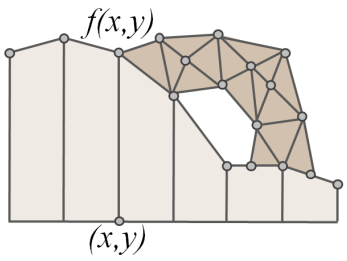
L'utilisation d'une énumération spatiale en voxels permet d'obtenir une représentation tridimensionnelle de la scène (IFMC03, BFO+07). Cette méthode consiste à discrétiser l'ensemble de l'espace sous la forme d'une grille tridimensionnelle pour créer des scènes de géométrie et de topologie quelconque comportant des surplombs, des arches ou des grottes. Les nœuds $\mathcal{N}_{ijk}$ de la grille stockent la présence de matière à la position (i, j, k) de l'espace. Cette structure est bien adaptée pour créer des petits terrains de géométries et de topologies quelconques composés de nombreuses grottes et arches. Elle permet également d'effectuer des simulations d'érosion ou de fluides par éléments finis (MrGrG+10). Par contre, elle est très volumineuse ce qui la limite à la création de scènes de petite taille.

1.1.1.4 Courbes



Afin d'obtenir une représentation compacte, les terrains peuvent être représentés comme un ensemble de courbes. Ces courbes définissent soit le terrain par des courbes de niveau (ZKT05, HSS03) soit par les caractéristiques du terrain comme les vallées ou les crêtes (RME09, GMe09). En cartographie, les courbes de niveau sont très utilisées pour représenter les terrains. Une courbe de niveau, notée Γ, est définie comme la ligne d'intersection entre un plan horizontal et le relief du terrain. Les courbes caractéristiques pour définir les lignes de crêtes ou de vallées permettent d'obtenir une représentation plus compacte que les courbes de niveau. Cette représentation est généralement utilisée dans les méthodes de création de terrain par esquisses. Une reconstruction (souvent coûteuse en temps de calcul) à partir des courbes est nécessaire afin de pouvoir générer la surface du terrain et la visualiser en trois dimensions.

1.1.1.5 Données Hybrides



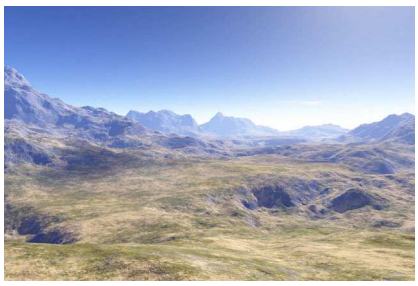
Il est possible de combiner différentes structures de données afin d'utiliser les avantages de plusieurs modèles. Cette technique consiste à créer des scènes de grandes tailles qui contiennent localement des caractéristiques complexes nécessitant plus de précision comme des arches ou des surplombs. Pour ce faire, ces méthodes combinent le modèle par carte d'élévation de données avec une structure complémentaire comme les maillages ou les voxels. Cette structure complémentaire est ajoutée à la carte d'élévation afin de définir ces nouvelles caractéristiques. Dans l'industrie du cinéma et du jeu vidéo, le maillage est la donnée complémentaire la plus

utilisée. Ces maillages sont définis manuellement par des graphistes qui éditent l'ensemble des points de chaque triangle afin d'obtenir la forme souhaitée se raccordant avec le terrain. Bien que cette approche permette de définir l'ensemble des caractéristiques du terrain comme les surplombs, les arches et les grottes, elle nécessite un travail fastidieux de la part du graphiste pour raccorder les différents modèles. Les cartes d'élévation peuvent être également combinées à une représentation par voxel. Le graphiste définit une région de la carte d'élévation de données qui est convertie en une grille tridimensionnelle de voxels pouvant être sculptés. La graphiste sculpte interactivement la grille avec des outils de sculpture spécifiques à l'édition de voxel. Cette méthode a été utilisée avec succès pour la création des scènes du jeu vidéo Crysis de EA Games. Par contre, ces caractéristiques sont définies sur des petites zones fixes afin d'obtenir un modèle final pas trop coûteux.

1.1.2 Algorithmes de génération de terrains

De nombreuses techniques de génération automatique de terrains ont été proposées. La majorité des techniques existantes utilisent les cartes d'élévation de données pour générer les terrains et s'appuient soit sur des modèles procéduraux, des simulations physiques ou encore des méthodes à partir d'esquisses pour créer les terrains.

1.1.2.1 Méthodes procédurales



Les méthodes procédurales ont été les premières méthodes utilisées pour créer des terrains. Ces techniques génèrent des cartes d'élévation de données à partir d'une fonction f qui renvoie une hauteur $f(x, y)$ pour tout point (x, y) de la grille.

Le Mouvement Brownien Fractionnaire (EMP+98, EMP+02) est une approche classique pour générer des cartes d'élévation. Cette fonction est une généralisation des fonctions de bruit. La hauteur d'un point $\mathbf{p}$ de la surface est calculée à partir de la somme pondérée de fonctions de bruits :

$$f(\mathbf{p}) = \sum_{i=0}^{o-1} \frac{n(\mathbf{p}.\lambda^i)}{\lambda^h}$$

où o est le nombre d'octaves, λ est l'écart entre les fréquences successives, $n(\mathbf{p})$ est la valeur de la fonction de bruit au point $\mathbf{p}$, et h est le paramètre incrémental de la fonction fractale. La fonction de bruit de Perlin est obtenue avec $h = 1$ et $\lambda = 2$. La fonction f est homogène et isotrope, ce qui a pour effet de produire une rugosité homogène sur l'ensemble du terrain.

Pour obtenir des terrains plus réalistes, les multifractales (PJ95) combinent des fonctions de bruit par multiplications au lieu d'additions successives. De ce fait, la

présence de hautes fréquences dépendent des valeurs de bruit de basses fréquences. Des modèles de terrains hétérogènes (MKM89) ont été développés pour créer des terrains plus réalistes en approximant des modèles d'érosion en fonction de la pente du terrain. D'autres modèles de génération de terrains dérivés de ceux ci ont été définis pour obtenir des caractéristiques spécifiques comme des falaises (Ols04), des rivières (PHH93, Bel07), ou pour contraindre la génération (SS05, Bel07).

1.1.2.2 Simulation d'érosion

Les méthodes de simulation d'érosion permettent d'augmenter le réalisme des terrains générés procéduralement. Deux principaux types d'érosion existent l'érosion thermique et l'érosion hydraulique. L'érosion thermique crée des sédiments par des chocs thermiques. En se séparant de la roche, les sédiments se stabilisent en fonction de la pente locale du terrain. L'érosion hydraulique dissout une petite quantité de matière du terrain par son frottement avec de l'eau puis les sédiments sont transportés par le mouvement du fluide.

Plusieurs méthodes (RPP93, Nag98, BF02) simulent le mouvement de l'eau comme un simple algorithme de diffusion. Cette approximation est souvent trop grossière afin d'obtenir un résultat réaliste. De ce fait des méthodes de simulation de fluides ont été mises en place pour compléter le processus d'érosion (CMF98). La combinaison de la simulation d'érosion et de la simulation d'écoulement des fluides est très coûteuse. Ces simulations ont été optimisées et portées sur les cartes graphiques qui offrent une puissance de calcul hautement parallèle pour obtenir des temps de calculs interactifs (NWD05, MDH07, SBBK08). Afin d'obtenir un résultat plus réaliste, plusieurs méthodes d'érosion définissent la stratification du terrain par un ensemble de couches empilées (RPP93, IFMC03, BFO+07). La majorité des méthodes utilise les cartes d'élévation de données pour simuler l'érosion sur de grands terrains mais quelques méthodes utilisent des grilles de voxels afin d'obtenir des caractéristiques plus complexes comme la création de falaises ou de Goblins (IFMC03, BTHB06, BFO+07). Ces méthodes consistent à éroder progressivement les voxels en définissant une valeur d'érosion pour chaque voxel. Lorsque la valeur associée au voxel est nulle, alors le voxel est érodé.

Une limitation des modèles d'érosion est qu'ils représentent uniquement une partie des sédiments obtenus par l'érosion (sable). Les blocs de pierres issus de l'érosion thermique ne sont pas modélisés ce qui ne permet pas d'obtenir une simulation exacte et qui diminue le réalisme des scènes.

1.1.2.3 Synthèse à partir d'exemple

Pour générer un terrain ayant les même caractéristiques qu'un autre terrain, Zhou (ZSTR07) a proposé une méthode de génération de terrains à partir d'exemple. Le processus consiste à générer un terrain dont les traits caractéristiques de la scène suivent les contraintes de l'utilisateur. La méthode s'inspire des techniques de génération de texture par l'exemple (EF01, WY04) et s'exécute sur une carte d'élévation de données représentée comme une texture. Le processus de génération utilise en entrée une scène exemple issue de bases de données pour recréer un terrain de même type mais suivant les contraintes de l'utilisateur. La carte d'élévation en entrée est découpée en un ensemble de morceaux rectangulaires. Les morceaux obtenus sont assemblés sur le terrain résultat en suivant les contraintes définies par l'utilisateur tout en minimisant les coûts des différentes altitudes le long des jointures. Les jointures entre les morceaux sont finalement lissées en utilisant les équations de Poisson afin d'obtenir une transition sans discontinuité. Cette méthode est efficace pour créer de grands terrains en suivant globalement la contrainte de l'utilisateur mais ne permet pas de contrôler précisément la forme des montagnes et leur emplacement exact. Une méthode similaire à celle ci génère le terrain en partant directement d'un ensemble restreint de scènes exemples passées en entrée (SS05).

1.1.2.4 Génération par esquisses

Quelques travaux (CHZ00, WI04) ont étudié des méthodes de modélisation à partir d'esquisses afin de contrôler la forme du relief. Ces méthodes définissent les crêtes et les vallées de la scène par un ensemble de courbes qui sont ensuite interprétées par le système pour reconstruire le relief. La reconstruction du relief est effectuée par des méthodes de diffusion ou de déformation multirésolution de la surface. Gain (GMe09) propose un système complet de création de terrain par esquisses. Le système prend en entrée l'esquisse d'une crête de montagne, et le système génère automatiquement l'empreinte au sol de la montagne. Cette empreinte peut être modifiée par l'utilisateur pour redéfinir la forme de la montagne. Une fois que l'utilisateur a défini les lignes, le système génère le relief se trouvant à l'intérieur de l'empreinte tout en s'adaptant à l'esquisse. Des détails géométriques sont ensuite ajoutés au relief par propagation de bruit pour rendre le terrain plus réaliste (CD05).

Une limitation de ce modèle est que la génération s'effectue par un ensemble d'opérations incrémentales qui ne permet pas de modifier les esquisses précédentes. Le résultat obtenu ne peut donc pas être stocké de manière compacte comme un ensemble de courbes.

1.1.3 Conclusion

Les travaux existants permettent de modéliser soit de très grands terrains (plusieurs centaines de kilomètres carrés) soit de simuler l'évolution de petits terrains (quelques dizaines de mètres carrés). Ces échelles extrêmes sont inadaptées à la représentation de scènes à échelle humaine (quelques hectares) qui ont la particularité de nécessiter la prise en compte des très nombreux détails au sol (pierres, feuilles, branches). En effet, ces détails ne peuvent pas être approximés uniquement à l'aide de textures ou d'imposteurs, et requièrent non seulement des modèles géométriques précis, mais également en très grand nombre. En outre, ces scènes nécessitent de générer d'importantes masses de données et de faire interagir l'ensemble des éléments de la scène pour les intégrer de manière naturelle.

Dans le chapitre 2, nous proposons une nouvelle représentation de terrain pour permettre de créer ces scènes complexes avec un haut niveau de détail. Notre modèle volumique permet de créer des terrains de géométrie quelconque (arches, surplombs et grottes) ce que ne permettent pas la plupart des modèles existants s'appuyant sur des champs de hauteur. Notre modèle permet également de définir les différentes couches de matériaux de la scène pour pouvoir reproduire des opérations complexes comme l'érosion ou l'écoulement d'eau. Enfin, dans le chapitre 3, nous proposons une méthode originale pour la création d'empilement de pierres. Cette approche permettra d'ajouter automatiquement des détails géométriques visuellement très importants et jouant un rôle clef dans le réalisme global d'une scène.

1.2 Génération de la végétation

La végétation est un élément important à prendre en compte dans le processus de création de monde. La végétation est définie comme un ensemble de plantes et d'arbres interagissant avec l'environnement pour former un écosystème. La modélisation de l'ensemble de ces instances est très difficile car nécessite de prendre en compte de nombreux paramètres comme le type de végétation, les ressources disponibles ou encore la présence d'obstacles. Les travaux effectués sur ce domaine peuvent être classifiés en deux parties : la création d'une instance de plante et la distribution des éléments dans la scène.

1.2.1 Génération de plantes

Les plantes sont difficiles à modéliser car elles sont définies par une structure complexe pouvant être très différente en fonction des espèces. Par exemple, un arbre est défini par des racines, un tronc, un sous ensemble de branches, des fleurs et fruits et également des feuilles. Les approches proposées essaient de définir un modèle générique pour permettre de créer le maximum d'arbres différents d'une même espèce mais

également d'espèces différentes. Les deux principales approches sont la génération automatique d'instances par des méthodes procédurales et l'édition bas niveau pour mieux contrôler la forme de l'instance.

1.2.1.1 Génération procédurale

Les origines de la modélisation d'arbres en informatique peuvent être attribuées aux études de Ulam (Ula62) et Honda (Hon71). Honda (Hon71) considère un arbre comme une structure récursive. La structure est caractérisée par des paramètres tels que les angles des ramifications, et le ratio de la taille des ramifications entre chaque niveau de récursion. De nombreux autres algorithmes de génération récursifs ont également été proposés comme les L-Systèmes (Lin74) mais aussi d'autres systèmes (AK84, Blo85, RB85, Opp86, WP95, LD99, PMKL01). En revanche, Ulam (Ula62) considère les arbres comme des structures auto-organisées où les modèles de ramifications se développent par une compétition pour l'espace. Le processus de génération utilise des automates cellulaires pour définir l'ensemble des branches de l'arbre (Gre89, BM02, Pal07, BL09).

L'analyse théorique de la ramification des modèles (BS81) donne un aperçu des principales différences entre les modèles fabriqués avec les techniques d'auto-organisation et les techniques récursives. Cette étude montre que le nombre de nœuds croît de façon exponentielle avec l'âge (profondeur de récursivité), mais que le volume total du modèle ne croît que légèrement. Par conséquent, la densité des éléments augmente indéfiniment avec le temps. Honda a abordé ce problème en diminuant la taille des branches avec le niveau de récursivité, tout en préservant la topologie répétitive à tous les niveaux de la récursivité. En revanche, Ulam utilise des branches de taille constante, mais a permis aux segments terminaux d'avoir des destins différents. Certains donnent lieu à de nouvelles branches et d'autres non, en fonction des résultats de la compétition pour l'espace.

Pour s'adapter aux applications de synthèse d'image, les méthodes de modélisation d'arbres doivent trouver un compromis entre les procédures de génération et le contrôle de l'utilisateur sur les structures. Les premiers modèles récursifs (Hon71, Opp86) offrent un contrôle limité sur la structure arborescente, comme les angles de ramification et les rapports de longueurs de branches d'arbres. Des techniques global-à-local (RB85, WP95, LD99, PMKL01) proposent de spécifier explicitement la silhouette de l'arbre et la densité des branches. Ces techniques permettent de créer des modèles très réalistes dans des mains d'expert. Toutefois, l'utilisateur doit contrôler un grand nombre de paramètres et la modélisation d'arbres âgés contenant des branches irrégulières est difficile à obtenir. Palubicki (PHL+09) résout ce problème en utilisant une technique d'auto-organisation en permettant aux branches de s'adapter de façon autonome à l'espace disponible.

1.2.1.2 Contrôle de la génération

Power (PBPS99) et Boudon (BPF+03) sont les premiers à proposer des techniques pour manipuler et éditer directement les végétaux. Ces techniques permettent à un utilisateur de tailler et de plier les branches lors du processus de génération. Des contrôles de la forme encore plus strict ont été réalisés dans des modèles d'édition par esquisses de petites plantes (IOOI05, IOI06a, OOI06, ASSJ06). Des extensions basées esquisses pour générer de grands arbres ont également été proposées (IOI06b, ZS07, WBCG09). Ces modèles consistent à définir par des esquisses les premiers niveaux de récursion de l'arbre et de générer ensuite l'ensemble des branches par un modèle procédural. D'autres travaux ont été effectués récemment sur la modélisation par l'exemple pour modéliser des arbres à partir des données numérisées (CWFV09) ou des photographies (NFD07).

Les modèles d'arbres peuvent encore être contrôlés en modifiant l'environnement lui-même. Greene (Gre89) crée des plantes grimpantes et des vignes qui sont guidées par des structures d'appui. Un modèle de contrôle de développement (MP96) guide les racines par une distribution de l'eau dans le sol spécifiée par un programme de dessin. Un autre modèle (PMKL01, RLP07) consiste à couper l'ensemble des branches qui poussent à l'extérieur d'un volume de taille prédéfinie, les branches latérales remplissent alors le reste du volume.

1.2.2 Distribution de végétaux

Trois approches ont été proposées pour distribuer les plantes dans une scène naturelle. La première approche consiste à simuler la croissance d'un ensemble de plantes par une compétition de ressources. La deuxième approche consiste à contrôler la distribution de végétaux à l'aide de cartes de densité. Enfin, la dernière approche consiste à utiliser les méthodes de pavage pour optimiser la distribution des végétaux en utilisant plusieurs fois la même instance.

1.2.2.1 Simulation d'écosystème

En informatique graphique, les premiers travaux de simulation d'écosystèmes ont été présentés par Deussen (DHL+98). Cette approche consiste à faire croître progressivement un ensemble de plantes interagissant entre elles. Une plante est représentée par un disque définissant sa position et sa taille. La simulation est initialisée par une distribution aléatoire de plantes de différentes tailles sur un plan. Lorsque deux disques s'intersectent, la plante ayant la plus petite taille de disque meurt et est supprimée. La simulation fait croître incrémentalement les disques au cours du temps et lorsqu'une plante a atteint son espérance de vie maximale elle est supprimée. Ce processus permet de reproduire le phénomène d'éclaircissage dans la nature. Ce phénomène correspond à l'étouffement et à la mort des jeunes plantes dominées par les plus grosses lors de la compétition aux ressources. Lorsqu'une instance est morte une nouvelle distribution de plante est effectuée dans l'espace disponible.

Afin d'obtenir des résultats réalistes, Lane (LP02) a proposé un modèle permettant de prendre en compte les phénomènes de préférence (symbiose) dans un groupe de plantes. Dans la nature, nous pouvons observer que les plantes de la même famille ont tendance à se développer par groupe. Ce phénomène a un impact important sur l'aspect de la distribution. Lane a généralisé le processus de création par L-Système (Lin74) afin de définir des règles de croissance pour un groupe de plantes. Dans son multiset L-système, Lane utilise trois phénomènes pour décrire le développement d'un groupe de plantes. Le premier phénomène est l'éclaircissement comme celui utilisé par Deussen (DHL+98). Le deuxième phénomène est la succession d'une plante qui est remplacée par un ensemble de plantes plus petites lorsque celle ci meurt. Le dernier phénomène consiste à propager une espèce pour qu'une plante donne naissance à d'autres plantes de même espèce dans son voisinage.

Une limitation de ces deux modèles est que le processus ne permet pas aux instances de se partager les ressources, car la plante dominante monopolise les ressources et détruit ses concurrents. Ce processus ne permet pas d'obtenir une évolution des plantes réaliste au cours du temps. De ce fait, aucun jeune plant d'arbre ne peut se trouver à proximité d'un arbre dominant.

Une méthode plus réaliste a été proposé par Alsweis (AD05, AD06) pour mieux modéliser la compétition de ressources. Leur méthode utilise globalement la même procédure que dans (DHL+98, LP02) par contre les instances ne sont pas supprimées lors de l'intersection des disques. La simulation permet aux plantes de se partager les ressources pour obtenir une simulation symétrique lorsque les disques des plantes s'intersectent. La ressource de la plante est définie par une intégration du volume d'intersection des plantes. Le volume d'intersection est pondéré en fonction de la distance au tronc de l'arbre. Lorsqu'une plante à le monopole sur une autre, elle se développe au détriment de ces voisines pour devenir une simulation asymétrique et continuer son développement. Avec ce comportement de partage de ressource, le résultat obtenu est beaucoup plus réaliste que les méthodes précédentes.

1.2.2.2 Distribution à partir de cartes de densité

Les méthodes reposant sur la simulation de comportement individuel offrent peu de contrôle sur le placement des instances. Afin de contrôler la distribution de la végétation dans l'espace, quelques approches proposent d'utiliser des cartes de densité. La difficulté de cette approche réside dans le processus d'instanciation des végétaux à partir de la carte de densité.

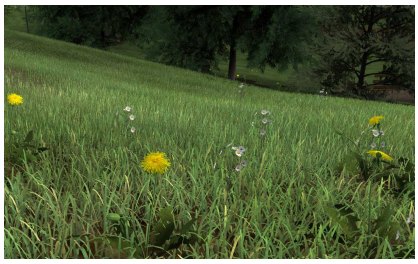
Deussen (DHL+98) propose d'utiliser des cartes de densités représentant les ressources d'eau disponibles. Ces cartes sont créées à partir de données réelles, simulées par un processus d'écoulement sur une carte d'élévation ou bien peintes à la main. La croissance de la plante s'effectue en fonction de la quantité d'eau présente dans le voisinage et en fonction du paramètre d'absorption défini pour chaque espèce de plantes. Le réalisme de la distribution est donc fortement dépendant des données récoltées. Lane (LP02) défini un algorithme d'instanciation permettant de tenir compte de l'éclaircissement. Cet algorithme repose sur une probabilité de présence d'une plante dans l'espace. La probabilité est définie sur une image de niveau de gris pour chacune des espèces de plantes. L'ajout d'une plante modifie la probabilité d'apparition pour les autres plantes dans son voisinage en respectant la contrainte de densité. Cette approche permet de définir le seuil d'éclaircissement entre chaque plante et de contrôler ainsi leurs interactions et leurs placements. Une limitation de ces modèles est qu'ils ne permettent pas de contrôler l'impact de l'homme sur la simulation.

Benes (BCS03) propose une approche différente afin de contrôler la modélisation de l'écosystème par des agents. Les agents virtuels entrent dans l'écosystème pour effectuer des actions favorisant certaines espèces végétales et afin de provoquer des instabilités dans le système. Les agents libèrent de l'espace en éliminant des plantes lorsqu'une zone est surpeuplée, ils plantent des graines et arrosent les plantes pour agir comme des jardiniers. Des règles sont définies pour chaque agent et permettent ainsi de bien contrôler le positionnement des instances tout en laissant part à la simulation.

Le processus se déroule jusqu'à ce que l'écosystème soit globalement stabilisé.

1.2.2.3 Méthodes de pavage

Les méthodes de simulation et de distribution à partir de cartes de densités nécessitent de placer et de simuler la croissance de chaque plante individuellement. Ces méthodes sont très coûteuses et ne permettent pas de distribuer un nombre très important d'instances sur le terrain. Afin de distribuer une grande quantité de plantes sur le terrain et de contrôler leur emplacement, des méthodes de pavage ont été proposées (GC01, CSHD03).



L'objectif de ces méthodes de pavage est de minimiser le nombre d'instances dans la scène en les distribuant par groupes tout en dupliquant les instances. La distribution des instances sur le terrain ne doit pas faire apparaître de répétition d'instances et de discontinuité entre les pavés. Ces méthodes nécessitent alors de construire des pavés de telle sorte que les pavés voisins aient une distribution continu entre eux. Pour cela plusieurs méthodes ont été mises en place (NC99, DN04, CSHD03, LD06b, BPB09). Le principe général est de construire des pavés composés d'un ensemble d'instances, en vérifiant que les instances soient distantes d'un certain rayon. La propriété de distance entre les instances doit être respectée à l'intérieur du pavé mais également aux niveaux des interfaces entre les différents pavés. Afin de respecter ces contraintes, Lagae (LD06b) a proposé un modèle de création de pavés rectangulaires qui respectent les conditions de proximité entre les 8 voisins du pavé. Sa méthode consiste à distribuer un ensemble de points à l'aide d'une distribution de Poisson avec des contraintes aux interfaces.

Ces approches sont bien adaptées à la création de grandes étendues de végétaux homogènes comme des champs, des forêts ou des gazons. Un avantage de cette méthode est que le coût mémoire est beaucoup plus faible que les autres techniques. Cette approche permet également d'être combinée à une carte de densité pour contrôler les zones de distribution. Une limitation de ce modèle est que la distribution des instances de végétation est homogène ce qui ne permet pas de créer des scènes réalistes car elles manquent de variétés.

1.2.3 Conclusion

Les techniques de modélisation et de distribution d'espèces végétales permettent d'obtenir des résultats réalistes et de reproduire fidèlement l'évolution d'un écosystème. Une limitation importante à ces travaux est que la gestion des ressources pour la croissance des plantes ne soit généralement pas prise en compte ou sinon de manière très imprécise.

Dans le chapitre 2, nous présentons un modèle original qui permettrait d'évaluer quantitativement l'ensemble des ressources accessibles à une plante (eau, soleil, terre) pour pouvoir simuler très précisément les conditions de développement des organismes de l'écosystème. Ce travail dépasse le cadre de cette thèse, mais constitue une piste de recherche sur laquelle nous reviendrons en conclusion de ce mémoire.

1.3 Génération de paysages urbains

La modélisation de paysages urbains est un domaine récent en informatique graphique. Les travaux effectués dans ce domaine se sont principalement orientés sur la génération de villes. Par contre peu de travaux ont étudié l'intégration de maisons ou de routes dans un paysage rural. La modélisation d'un espace urbain est très difficile car nécessite une structure sous-jacente définie par un nombre important de paramètres. Le processus global de génération de ville est le suivant :

1. Génération du réseau de routes composé des routes principales et secondaires
2. Création des quartiers défini par un ensemble de blocs obtenus par les croisements de routes
3. Génération des instances de bâtiments à l'intérieur des quartiers pour définir la géométrie

Dans cette section, nous détaillons les deux principales étapes de génération de la ville qui sont la génération des bâtiments et la création du réseau de routes. Nous détaillons également les méthodes existantes d'édition de routes de campagne.

1.3.1 Création de bâtiments

Plusieurs processus ont été proposés pour créer des instances de bâtiments. Les architectes utilisent des logiciels pour ajouter et éditer interactivement l'ensemble des pièces d'un bâtiment. Ces logiciels demandent un fort investissement de l'utilisateur afin de pouvoir créer un modèle final. En informatique graphique, les travaux effectués consistent à créer rapidement un nombre important d'instances de bâtiments. Ces bâtiments sont définis par un nombre limité de paramètres afin de faciliter l'interaction avec l'utilisateur. Les techniques proposées pour créer ces bâtiments utilisent principalement des méthodes procédurales afin de respecter ces contraintes. Récemment, d'autres travaux ont consisté à modéliser des bâtiments à partir d'exemples.

1.3.1.1 Génération procédurale

La modélisation procédurale est utilisée pour automatiser la génération de structures urbaines complexes comme les maisons et les bâtiments afin de produire des contenus numériques à partir d'un ensemble restreint de paramètres et de règles. Les

procédures de modélisation architecturale peuvent utiliser l'un des systèmes de production existant tels que les grammaires de Chomsky (Sip96), les grammaires de graphes (EKG99), les grammaires de forme (Sti75), les L-systèmes (GMB06). En architecture, Stiny lance l'idée des grammaires de forme (Sti75, Sti80). Ces grammaires de forme ont été utilisées avec succès pour la construction et l'analyse de la conception architecturale. La formulation initiale de la grammaire de forme fonctionne directement sur un arrangement labellisé de lignes et de points. Les règles de dérivation peuvent être modélisées en dessinant un ensemble de lignes et de points labellisés. En pratique, ce processus conduit à un problème de dérivation complexe qui nécessite une évaluation à chaque étape de l'itération afin de choisir parmi un nombre important de transformations quelle règle doit être appliquée.

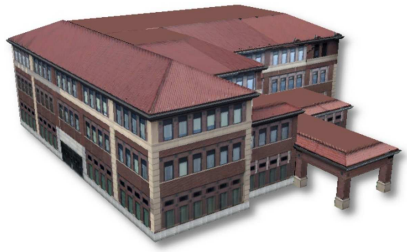
Pour faire du concept de grammaire de forme un modèle plus applicable, Wonka (WWSR03) et Müller (MWH$^+$06) ont introduit un cadre composé de règles permettant de remplacer une forme par une ou plusieurs autres formes, ainsi que des mécanismes de dérivation automatique de règles. La grammaire originale de Wonka (WWSR03) est essentiellement axée sur les règles de conception de façades par des opérations de fractionnement. L'opération de fractionnement permet de briser les formes élémentaires (cubes et cylindres) en divisant le long des plans de formes élémentaires. La taille des règles de conception permet de préciser l'emplacement du plan de division lorsque la taille de la forme élémentaire doit être changée et le nombre de division devant être appliquée en fonction de la taille de l'élément. L'auteur propose également des mécanismes de sélection de règles pour assurer la cohérence entre les éléments tout en permettant des variations aléatoires.

Müller (MWH$^+$06) repose sur les même règles de fractionnement que celle proposée par Wonka (WWSR03) mais ajoute de nouveaux éléments. Les auteurs ont étendu les commandes utilisées dans les L-systèmes (GMB06) pour définir des modèles de masse. Les modèles de masse sont généralement créés par la composition de plusieurs formes élémentaires. Cette modélisation de masse est assez intuitive et reflète les processus de conception utilisés dans l'architecture. Les auteurs ont également introduit des règles sensibles au contexte en fonction des masses des différents bâtiments. Cette considération permet de générer une grande variété de bâtiments. Lipp (LWW08) propose un modèle pour éditer interactivement la grammaire. Au lieu d'écrire les règles avec un éditeur de texte, le modèle (LWW08) permet de concevoir et de modifier entièrement les règles à l'aide d'une interface utilisateur. Cette extension permet de rendre la procédure de modélisation accessible à un plus grand public.

1.3.1.2 Modélisation à partir d'exemple

Afin de générer un modèle de bâtiment, des travaux récents combinent les concepts de la vision par ordinateur et la modélisation procédurale. Ces méthodes consistent

à reconstruire le modèle tridimensionnel à partir d'une image de bâtiment passée en exemple.



Müller (MZWVG07) utilise la notion de règles de fractionnement afin de définir un cadre pour analyser la façade à partir d'une seule image orthographique en entrée. Cette méthode consiste à définir une simple image de façade non réductible, puis de trouver les lignes de fractionnement en détectant les lignes et les colonnes des éléments de la façade. La subdivision s'effectue exactement comme les règles de fractionnement utilisée dans (MWH+06). Les règles de grammaire peuvent ensuite être extraites pour redéfinir la forme. Une approche similaire a été suggérée par Aliaga (ARB07). Cette approche prend en entrée plusieurs images d'un même bâtiment afin de pouvoir reconstruire l'ensemble du bâtiment. Ce modèle permet également de générer un nouvel immeuble du même style que celui passé en exemple à partir de la grammaire extraite.

Merrell (Mer07) a introduit une méthode de synthèse d'objets pour créer des bâtiments à partir d'un modèle tridimensionnel. Cette méthode consiste à découper le modèle en différents morceaux cubique et de combiner l'ensemble de ces morceaux pour définir un nouvel objet. La combinaison des morceaux s'effectue en plaçant les cubes les uns à cotés des autres en respectant la continuité entre les cubes. Une restriction de cette méthode est que les objets d'entrée doivent être alignés sur une grille pour permettre la création des différents morceaux. Dans une seconde méthode (MM08), l'extraction des morceaux a été améliorée pour prendre en compte la connectivité entre les différentes caractéristiques du modèle en entrée. Les auteurs ont également amélioré la génération des éléments en permettant d'obtenir des orientations arbitraires.

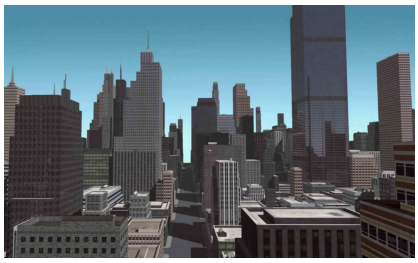
1.3.2 Modélisation des réseaux de rues

La modélisation de réseaux de rues consiste à définir un graphe composé de routes principales et de routes secondaires. Le graphe définit la structure de la ville pour représenter l'ensemble des caractéristiques de la ville comme les centres villes ou les banlieues. Deux processus principaux ont été proposés pour créer ce graphe : une méthode de génération procédurale du réseau et une méthode de modélisation par l'exemple.

1.3.2.1 Génération procédurale

Générer un réseau de rue consiste à générer un graphe. De nombreux travaux de génération de graphes ont été proposés dans la littérature qui pourrait s'apparenter à un graphe de rues. Parmi ces méthodes, une classe intéressante d'algorithmes est l'utilisation des diagrammes de Voronoï de points distribués aléatoirement (dBvKOS00).

Cela permet de générer un réseau de rues non uniforme. D'autres modèles ont été proposés pour créer des textures (Wor96), des mosaïques (Hau01), des fractures (Mou05) et même des motifs de rues (SYBG02, GMB06). Un autre algorithme de génération de graphe est proposé dans le cadre de la modélisation de nervures de feuilles (ARP05). Si certains de ces algorithmes peuvent correspondre à un modèle de génération de rues, ils ne permettent pas d'obtenir les motifs spécifiques des réseaux de rues.



A ce jour, l'algorithme le plus efficace pour générer un graphe de rue est celui proposé par Parish et Müller (PM01). Ce modèle consiste à étendre le concept des L-systèmes (GMB06) afin de faire croître les segments de rue comme les branches d'un arbre jusqu'à ce que les branches croisent un segment de rue existant. Le processus prend en entrée différentes cartes de données pour définir les frontières entre la terre et l'eau ainsi qu'une carte de densité de population. Les L-Systèmes ont été étendus pour prendre en compte les caractéristiques définis par ces cartes pour s'adapter aux contraintes globales et locales de la scène et afin de réduire la complexité des règles de production. Les auteurs utilisent également des règles spécifiques pour obtenir des motifs de rues par exemple pour représenter des motifs rectangulaires (New York), radiales (Paris) ou en fonction du relief (San Francisco). L'ensemble de ces motifs de rues ont été étudiés par Focas (Foc98).

Chen (CEW+08) utilise la notion de champs d'écoulement et de champs de tenseur pour définir le tracé des rues de la ville. Cette méthode permet de contrôler la direction des lignes caractéristiques du graphe. Les champs de tenseurs donnent lieu à deux séries de direction qui sont la direction des vecteurs propres principaux et les vecteurs propres mineures pour définir respectivement les routes principales et les routes secondaires. Le processus de modélisation consiste à modéliser tout d'abord le champ de tenseur puis de générer le graphe des rues. L'étape de modélisation du champ de tenseurs utilise plusieurs opérations d'édition comme l'ajout de bruits, le lissage et la génération de champ de tenseur à partir de cartes topographiques. Le graphe de rues est ensuite obtenu en combinant l'algorithme de Parish (PM01) et les contraintes définies par le champ de tenseur. Afin d'obtenir un résultat plus réaliste, les auteurs ajoutent des contraintes de proximité entre chaque rue et en définissant un nombre d'impasse fixe.

Une récente approche (WMWG09) propose de combiner des techniques de modélisation procédurale avec de la simulation urbaine pour obtenir des réseaux de rues évoluant avec le temps. Le système comprend un algorithme d'expansion de rues, une simulation d'utilisation des terres, et une simulation de trafic. Bien que le modèle soit plus simple que les simulations urbaines complètes, la simulation est interactive et l'utilisateur peut appliquer des modifications durant la génération. Les modifications sont de natures différentes telles que le contrôle des zones de croissance, l'ajout de routes

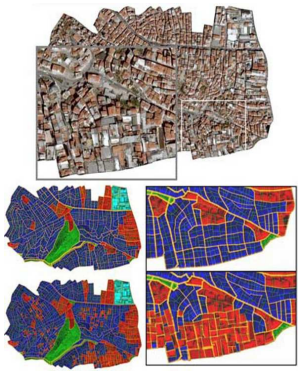
de montagne, l'édition de l'utilisation des terres et le changement des paramètres de la simulation. L'objectif de ce travail est de fournir un cadre générique permettant d'être configuré pour différentes catégories de génération. Contrairement à la méthode proposée par Chen (CEW+08), l'utilisateur ne contrôle pas l'ensemble de la génération ce qui simplifie son interaction.

1.3.2.2 Modélisation à partir d'exemple

La modélisation par l'exemple permet d'obtenir un nouveau graphe de rues à partir d'un modèle existant tout en respectant les motifs de l'exemple. Ces méthodes consistent à extraire le graphe de l'exemple pour ensuite synthétiser un nouveau graphe.

Hertzmann (HJO+01) introduit un processus de conception à deux phases qui peut être directement appliqué à la synthèse de texture par l'exemple et est applicable à l'imagerie aérienne en milieu urbain. La première phase de son processus consiste à obtenir une paire d'image comme donnée d'apprentissage dont une image est la version filtrée de l'autre. Dans la deuxième phase, le système utilise le filtre d'apprentissage afin de créer un résultat analogue sur une image cible. Les nouvelles images sont ensuite synthétisées en appliquant des statistiques de reconstruction à partir d'un exemple labellisé. Une limitation de cette méthode est qu'elle ne garde pas l'information de la structure du réseau afin de reconstruire l'image ce qui a pour effet de générer un réseau non réaliste.

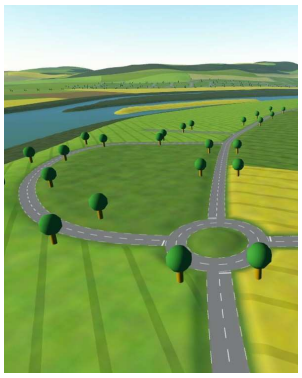
Plus récemment, des approches ont été adaptées pour exploiter l'organisation de l'espace urbain. La synthèse de textures traditionnelles ne prend pas en compte la structure géométrique du paysage urbain tel que les rues, les quartiers ou encore l'empreinte des bâtiments. Par conséquent, l'image résultante n'est pas uniforme et ne permet pas d'obtenir un réseau de rues réalistes. Aliaga (AVB08) propose une méthode de synthèse à partir d'exemple en fonction des tracés urbains pour prendre en compte la structure de l'espace urbain. Cette méthode utilise en entrée un ensemble de fragments d'aménagement urbain et effectue simultanément une synthèse fondée sur la structure et une synthèse basée sur l'image pour générer un réseau de rues complet. Les données de la structure et de l'image de la ville sont utilisées par l'algorithme de synthèse pour effectuer plusieurs opérations de haut niveau afin de générer interactivement de nouvelles structures complexes. L'utilisateur peut créer de nouvelles caractéristiques par une suite d'opérations telles que des jointures, des expansions, et des mélanges sans se soucier des détails de la structure de bas niveau qui sont reconstruit automatiquement.



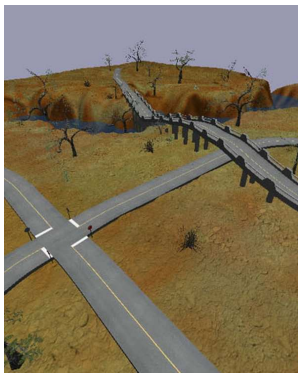
Dans des travaux connexes, les mêmes auteurs proposent une méthode de reconstruction interactive de tracés urbains (ABVA08). L'image de l'aménagement urbain peut être modifiée mais le système d'édition permet de reconnaître la structure urbaine. La méthode prend en entrée les données vectorielles des rues, des blocs et des parcelles de l'espace urbain, avec les images aériennes correspondante. Plusieurs opérations d'édition, telles que l'expansion, la mise à l'échelle, le remplacement et le déplacement de zones sont pris en charge. Ces opérations sont effectuées sur l'ensemble des blocs de la ville qui sont définis par les limites de parcelles. La transformation est réalisée par une distribution de la déformation sur l'ensemble des blocs adjacents tout en préservant la connectivité et en minimisant les distorsions individuelles. Ce processus est semblable à la minimisation de distorsions de texture utilisée pour texturer des objets complexes. Cette méthode est très efficace pour redéfinir une partie de la structure de la ville.

1.3.3 Création de routes

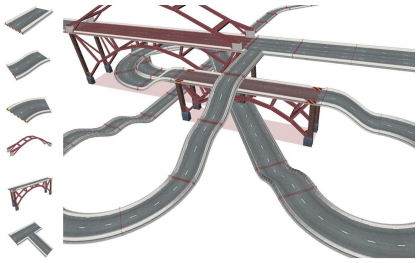
Plusieurs techniques ont été proposées pour créer une route, une trajectoire de route ou pour éditer une trajectoire existante. Ces modèles permettent de définir des routes dans des paysages urbains ou naturels. L'ensemble de ces méthodes permettent d'éditer interactivement le modèle obtenu.



Bruneton (BN08) propose une méthode de visualisation de terrain à grande échelle à partir de données vectorielles en entrée. Les données vectorielles représentent différentes caractéristiques de la scène comme le relief, les cours d'eau mais également l'ensemble du réseau de routes. Cette méthode permet de reconstruire la scène tridimensionnelle à partir de l'ensemble des données vectorielles. L'utilisateur peut manipuler les courbes vectorielles pour éditer les trajectoires de routes à partir de différents points de contrôle. A chaque édition, le système définit un nouveau relief pour adapter le terrain à la route. Une limitation de ce modèle est qu'aucune contrainte n'est appliquée à la trajectoire de la route, de ce fait les trajectoires éditées ne respectent pas les contraintes de génération de routes comme la pente du terrain ou la courbure de la route.



McCrae (MS09a) propose un système pour créer et éditer un réseau de routes en contrôlant la trajectoire des courbes. Afin d'obtenir des trajectoires réalistes, le système analyse les esquisses entrées par l'utilisateur pour les convertir en un ensemble de courbes de Clothoïdes (MS09b). Les Clothoïdes sont des courbes dont la courbure varie linéairement en fonction du paramètre de longueur d'arc. Le système permet également de générer automatiquement des ponts traversant d'autres routes en définissant les hauteurs du pont afin de respecter les contraintes de courbures. Le modèle modifie l'entourage du tracé afin de retirer la végétation trop proche, créer les ponts et tunnels pour traverser des caractéristiques spécifiques comme un lac et de construire les panneaux de signalisation appropriés en fonction du chemin. Une limitation de ce modèle est que la création des routes s'adapte uniquement à la contrainte de courbure mais ne prend pas en compte les caractéristiques tel que la pente, l'eau, la végétation qui peuvent influer sur la trajectoire. De ce fait, la trajectoire des routes de montagne est souvent non réaliste.



Cabral (CLDD09) a proposé une approche pour modéliser des scènes architecturales complexes en affectant des dépendances entre les différentes structures. Cette approche permet d'éditer et de combiner des modèles de scènes architecturales pour définir de nouvelles géométries. L'utilisateur peut modifier interactivement les modèles en préservant les contraintes de modélisation. L'utilisateur utilise ce modèle pour créer un ensemble de morceaux de routes qu'il combine les un aux autres afin de définir la route. Les morceaux sont assemblés en appliquant une contrainte de continuité entre chaque morceau.

1.3.4 Conclusion

Les méthodes de génération de paysages urbains sont plus récentes et sont, par rapport aux techniques de génération de terrains ou d'écosystèmes, moins abouties. Un problème récurent dans les techniques de génération de réseaux de rues ou de bâtiments est l'absence de prise en compte de l'environnement dans la génération. Les scènes obtenues sont définies sur des zones plates de terrain, sans prendre en compte des contraintes de pente, de nature des sols, ou de préservation de l'environnement.

Une amélioration possible serait d'ouvrir les méthodes existantes à la prise en compte de paramètres de l'environnement, un peu à la manière des Open L-Systems par rapport aux L-Systems. Dans le chapitre 4, nous proposons un modèle de génération de routes de campagne dont l'originalité est d'adapter la trajectoire de la route à l'ensemble des contraintes de l'environnement (pente du terrain, nature des sols, présence d'obstacles naturels comme les montagnes ou les cours d'eau).

1.4 Synthèse

De nombreuses méthodes de génération ont été proposées pour créer les éléments qui composent notre monde. Trois grandes techniques de génération sont utilisées pour créer ces éléments : la génération procédurale, la simulation et l'édition. La génération procédurale permet de créer rapidement une grande quantité d'éléments comme des bâtiments, des forêts ou des terrains. Cette méthode est principalement utilisée pour créer des grandes scènes de plusieurs dizaines de kilomètres. La simulation est utilisée pour obtenir des résultats réalistes et pouvant varier au cours du temps comme l'érosion d'un terrain ou la croissance d'un arbre. La simulation est appliquée sur des petites scènes ou sur des grandes échelles de faible résolution comme pour l'érosion. Les méthodes d'édition permettent de contrôler les formes à créer. Elles sont utilisées sur des grandes scènes comme les terrains et sur des petits éléments comme une plante.

Très peu de modèles ont été proposés pour créer des scènes de tailles intermédiaires, car cette catégorie pose de nombreux défis. Elle nécessite de manipuler une masse de données très importante composée d'éléments de nature variée. Afin d'obtenir un résultat réaliste, chaque élément doit prendre en compte son environnement lors de sa création afin d'être correctement intégré à la scène.

Dans les chapitres suivants, nous avons étudié la création de scène à l'échelle humaine en prenant en compte l'ensemble de ces problématiques. Dans le chapitre 2, nous proposons une nouvelle structure de terrain pour représenter les différentes caractéristiques de la scène ainsi que la définition de l'ensemble des détails par une représentation par couche. Dans le chapitre 3 et 4, nous proposons un modèle de génération d'empilement de pierres et de trajectoire de routes pour intégrer les détails géométriques à la scène en respectant les différentes caractéristiques de la scène.

Chapitre 2

Création de Terrains Complexes



Sommaire

Les résultats présentés dans ce chapitre ont fait l'objet d'une publication dans la revue internationale Computer Graphics Forum (PGMG09a).

Les cartes d'élévation de données sont généralement les structures de données les plus adaptées pour représenter des terrains de très grande taille avec un faible niveau de détail. Par contre, elles ne permettent pas de modéliser des caractéristiques complexes telles que les surplombs, les grottes ou encore des arches qui sont des caractéristiques essentielles lors de la création de scènes complexes. De plus, les cartes d'élévation de données définissent une unique surface qui ne permet pas de différencier les matériaux composant le sol comme la roche, le sable, le terre. Pour pouvoir créer des terrains complexes, nous avons mis en place une nouvelle représentation de terrain.

Dans ce chapitre, nous introduisons une représentation originale pour créer des terrains de forme complexe. Notre approche permet de modéliser des surplombs, des arches, des grottes et une multitude de matériaux différents représentés sous formes de couches comme la roche, le sable, la terre, les pierres ou l'eau.

Les terrains sont définis à l'aide d'un modèle hybride combinant une structure volumique discrète pour stocker les différentes couches de matériaux et une représentation implicite pour sculpter et générer la surface du terrain. Ce modèle hybride sert à obtenir une information volumique pour générer la surface du terrain et instancier les détails composant le sol comme les empilements de pierres.

L'édition du terrain est effectuée en travaillant soit sur le modèle discret, soit sur la représentation implicite pour tirer parti des avantages de ces deux modèles combinés. Cette approche permet d'optimiser les calculs en utilisant la meilleure représentation. La représentation implicite sert à sculpter la roche à l'aide d'outils propres aux surfaces implicites (FCG02). Elle sert également à générer le maillage de la surface du terrain pour la visualiser grâce à des algorithmes classiques de triangulation de surfaces implicites (BS91, BW97). La représentation discrète par couches de matériaux permet d'effectuer des opérations d'édition de haut niveau nécessitant par exemple une préservation des différents volumes de matière. Par exemple, nous utilisons ce modèle pour rajouter une quantité précise de sable ou de pierres au terrain. La représentation discrète est également utilisée pour distribuer naturellement les matériaux sur le terrain. Cette distribution est obtenue par une étape de simulation physique pour organiser et stabiliser les couches de matériaux en fonction de leurs caractéristiques.

2.1 Structure du modèle

Notre structure hybride est composée d'un modèle discret et d'une représentation implicite (Figure 2.1). Le modèle discret est utilisé principalement pour modifier le volume du terrain avec une information quantitative de la matière à éditer. Le modèle implicite est utilisé pour sculpter la forme de la surface du terrain avec des outils propres aux surfaces implicites. La surface du terrain est obtenue en générant le maillage triangulaire à partir du champ de potentiel du modèle implicite.

L'ensemble de la structure est couplé à un moteur de simulation physique qui définit les comportements de l'ensemble des matériaux par des opérations d'organisation et de

stabilisation. Ceci nous permet d'obtenir une représentation compacte et une distribution naturelle des matériaux sur le sol. Les modifications effectuées sur un modèle se répercutent sur la structure complémentaire en utilisant des opérations de convolution et de discrétisation.

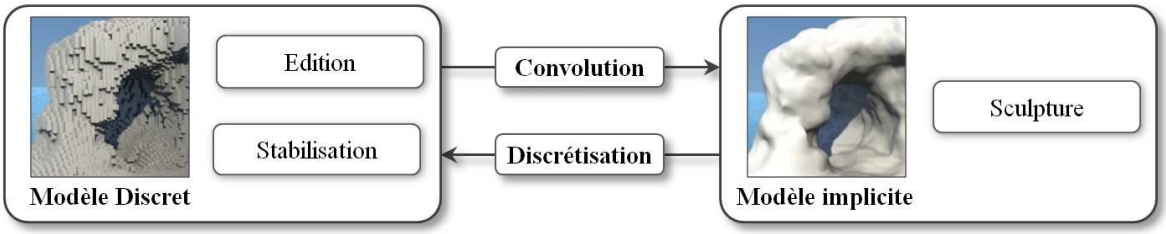


FIGURE 2.1 – Structure hybride du modèle de terrain composée d'un modèle discret et d'une représentation implicite auxquels sont associés leurs propres outils d'édition.

2.1.1 Représentation discrète

2.1.1.1 Structure de données

Un terrain est défini comme une grille à deux dimensions de piles de matériaux (Figure 2.1). Une pile de matériaux est définie comme un ensemble ordonné de couches de matière caractérisées par leur épaisseur e et le type du matériau $\mathcal{M}$ correspondant (BF01). Notre modèle prend en compte différents types de matériaux : l'air, l'eau, la neige, la glace, le sable, la terre, les pierres et la roche. Les surplombs, les arches et les grottes peuvent être facilement créés en insérant une couche d'air entre deux couches de roche (Figure 2.2). L'utilisation d'un grand nombre de types de matériaux différents permet de générer des scènes complexes d'une grande variété. Notre structure est ouverte et peut être aisément enrichie de nouveaux matériaux.

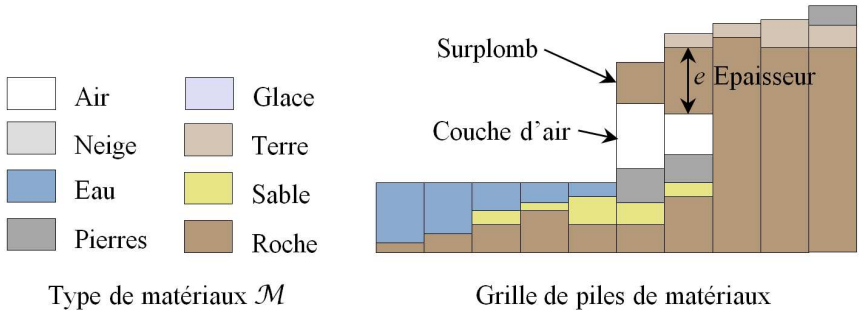


FIGURE 2.2 – Ensemble des matériaux présents sur le terrain, et la structure des matériaux en pile.

Notre structure de données permet d'obtenir un modèle 3D compact et moins coûteuse en mémoire qu'une décomposition de l'espace en voxel (IFMC03, BFO+07). Elle permet de générer des terrains contenant des caractéristiques différentes comme les

grottes, les falaises, des arches ou des paysages surréalistes. Le tableau 2.1 présente la quantité de mémoire utilisée pour modéliser différentes scènes. Le coût est comparé à la résolution effective définie par une décomposition de l'espace en voxels. Par exemple, le stockage d'une scène de résolution effective excédant $1\,000 \times 750 \times 8\,000$ (6 milliards de voxels) prend moins de 21 mégaoctet avec notre modèle. Cette structure de données est utilisée pour stocker les données de la scène définies par les épaisseurs et types de chaque couches de la grille, la représentation implicite est construite à la volée par une convolution des fonctions caractéristiques représentant les matériaux du terrain avec un noyau de convolution adapté.

Terrain	Résolution en voxel	Mémoire en voxel	Mémoire
Grotte	$1\,000 \times 750 \times 8\,000$	6 000 Mo	21 Mo
Canyon	$1\,000 \times 500 \times 5\,000$	2 500 Mo	14 Mo
Falaises	$2\,000 \times 500 \times 4\,000$	4 000 Mo	26 Mo

TABLEAU 2.1 – Comparatif du volume de données occupé en énumération spatiale et avec notre structure par couches sur plusieurs scènes.

Pour chaque matériau noté $\mathcal{M}$, nous définissons la fonction caractéristique notée $g(\mathbf{p})$ valant 1 si la matière au point $\mathbf{p}$ est de type $\mathcal{M}$ et 0 sinon.

$$g(\mathbf{p}) = \begin{cases} 1 \text{ si } \mathbf{p} \in \mathcal{M} \\ 0 \text{ sinon} \end{cases}$$

Cette fonction nous permet de définir les fonctions potentiel nécessaires $g_{\mathcal{M}}(\mathbf{p})$ pour définir la représentation implicite. Nous considérons que les matériaux en dehors des limites du terrain et au dessus des piles des matières existantes sont de type air.

Notre structure de données est combinée à un moteur physique pour traiter l'ensemble des matériaux du terrain. Le moteur physique opère sur l'ensemble des matériaux sauf la roche. Nous considérons les couches de roche comme fixes pour obtenir la structure du terrain et créer les surplombs. Les autres matériaux, dits de dépôt, sont considérés comme déposés sur la couche de roche. Le moteur physique gère la stabilité de l'ensemble des matériaux granuleux et liquides afin d'obtenir une distribution de la matière réaliste.

2.1.1.2 Gestion des couches de matières

Les différents matériaux sont caractérisés par leurs paramètres de granularité, de densité et d'adhérence. Ces paramètres définissent les comportements de stabilisation et d'interaction avec l'environnement. Ces paramètres peuvent varier lors d'un mélange avec d'autres matériaux. Ces mélanges sont obtenus lors du transport de la matière par le vent ou les précipitations. Ce phénomène est d'autant plus complexe à modéliser

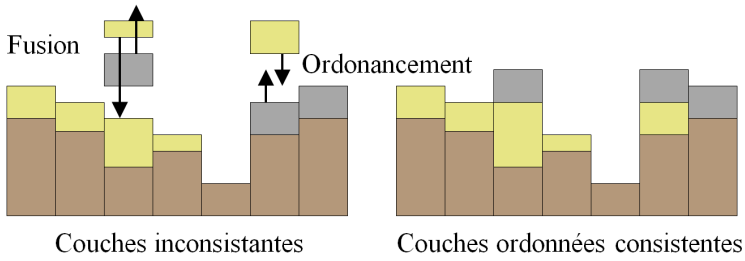


FIGURE 2.3 – Processus de tri et de fusion des couches.

qu'il nécessite de modéliser une quantité infinie de mélanges de matières contenant des paramètres spécifiques.

Rappelons que notre but n'est pas d'effectuer une simulation exacte, mais de générer simplement des scènes physiquement plausibles. Dans notre modèle, nous simplifions ces interactions en n'autorisant pas les mélanges de matériaux. Ceci permet également de réduire les coûts de traitements. Nous ordonnons et fusionnons l'ensemble des couches de matières de même type. De ce fait, l'ensemble des matériaux définis suffit à obtenir des scènes variées et réalistes. L'algorithme (Figure 2.3) procède en trois étapes :

1. L'ensemble des couches de matières se situant entre une couche d'air et de roche ou de deux couches de roches sont ordonnées selon l'ordre suivant (du plus profond vers la surface) : sable, terre, pierres, eau, glace, neige et air. Cette opération permet de trier l'ensemble des couches de matière en fonction de leur type.
2. Pour obtenir un modèle compact, l'ensemble des couches de même type sont fusionnées.
3. Nous supprimons ensuite l'ensemble des données incohérentes. Cette étape est effectuée en utilisant des règles arbitraires de bon sens afin d'obtenir des cohérences de superposition de matière s'approchant de la réalité. Par exemple, les couches de neige se situant sur une couche d'eau sont supprimées, sauf si une couche de glace les sépare.

Ce processus permet d'obtenir une structure de données compacte et cohérente en simplifiant l'interaction de l'utilisateur avec le terrain afin de mieux contrôler le positionnement des matériaux. Cette étape est effectuée à chaque modification de la structure de données soit lors de l'édition des matériaux de dépôts ou de la roche soit lors de simulation.

2.1.2 Représentation implicite

Nous définissons la surface des différentes couches de matériaux comme une surface de convolution (BS91) lissant la structure de données discrète. Cette représentation implicite nous permet de générer une surface de terrain continue entre chaque matériau. Le maillage du terrain est obtenu simplement en utilisant des techniques classiques de maillages de surface implicite (BS91, BW97). Dans cette partie, nous détaillons comment obtenir la surface de convolution d'un matériau qui est nécessaire pour l'édition d'un type de matières. Dans la partie Visualisation 2.5, nous présentons notre méthode pour obtenir une unique surface à partir des différentes surfaces de convolutions. Nous rappelons que la surface implicite S est définie par :

$$S = \left\{\mathbf{p} \in \mathsf{R}^3 | f(\mathbf{p}) = 0\right\}$$

La fonction potentiel f est définie comme une fonction linéaire dépendant d'une fonction de convolution $i(\mathbf{p})$. Nous définissons tout d'abord la fonction de convolution $i = (h * g)$ où g représente la fonction squelette et h représente le noyau de convolution.

$$i(\mathbf{p}) = (h * g)(\mathbf{p}) = \int_{\mathsf{R}^3} g(\mathbf{q})\, h(\mathbf{p} - \mathbf{q}) \mathrm{d}\mathbf{q}$$

La fonction squelette g est définie par la fonction d'appartenance à un matériau $g(\mathbf{p})$ présentée dans la partie représentation discrète.

2.1.2.1 Etat de l'art des noyaux de convolution

Dans la littérature, plusieurs fonctions noyaux h ont été proposées : des fonctions gaussiennes (BS91), des fonctions de Cauchy (She99), des fonctions inverses (WO96), des fonctions inverses au carré, des Metaballs (NHK+85) ou encore des polynômes quadratiques (Tableau 2.2). Leurs propriétés et la complexité de l'évaluation de l'intégrale associée sont présentées dans les travaux de Sherstyuk (She99) et de Jin (Jin02). La complexité de ces noyaux a été étudiée sur des primitives ponctuelles et les segments de droites. Dans notre cas nous étudions un modèle volumique composé d'un ensemble de boîtes réparties dans l'espace ce qui augmente le coût de l'évaluation de ces fonctions. Des statistiques comparant les temps de génération en utilisant les différents noyaux sont détaillées dans le tableau 2.2.

Les noyaux utilisant un support infini comme les fonctions gaussiennes sont très coûteuses en temps de calcul car l'évaluation de la convolution demande de calculer une intégration sur l'ensemble du terrain. Par contre, les noyaux à support compact, noté Ω, permettent d'effectuer un calcul local de l'intégration. Le support compact Ω est associé et centré au point $\mathbf{p}$. C'est le cas de la fonction quadratique polynomiale. Une fonction de convolution utilisant un support compact s'écrit :

$$(h * g)(\mathbf{p}) = \int_{\Omega} g(\mathbf{q})\, h(\mathbf{p} - \mathbf{q}) \mathrm{d}\mathbf{q}$$

Noyaux	Fonctions	Point	Segment
Gaussienne	$h(r) = exp(-a^2\, r^2)$ si $r > 0$	4.54	20.29
Cauchy	$h(r) = 1/(1 + s^2\, r^2)^2$ si $r > 0$	4.51	23.35
Inverse	$h(r) = 1/r$ si $r > 0$	5.40	14.52
Carré	$h(r) = 1/r^2$ si $r > 0$	4.24	17.48
Polynomiale	$h(r) = (1 - r^2)^2$ si $r \leq 1$	3.45	8.80

TABLEAU 2.2 – Temps d'évaluation des différents noyaux pour une primitive ponctuelle et un segment (secondes).

Quelques noyaux à support compact ont également été étudiés dans (She99, Jin02). Ces fonctions utilisent une distance Euclidienne à un point, ainsi le support compact est défini comme une sphère (Figure 2.4). Même pour des squelettes de faible dimension comme les segments, l'évaluation de l'intégration reste complexe et très coûteuse. Dans notre cas, le squelette est encore plus complexe puisqu'il est défini comme une union de boîtes noté $\mathcal{B}$. La fonction squelette g correspondant est un ensemble de fonctions constantes dont la valeur est égale à 1 à l'intérieur de l'union des boites $\mathcal{B}$ et égale à 0 en dehors. De ce fait, nous proposons d'utiliser une fonction noyau très simple permettant d'accélérer les calculs.

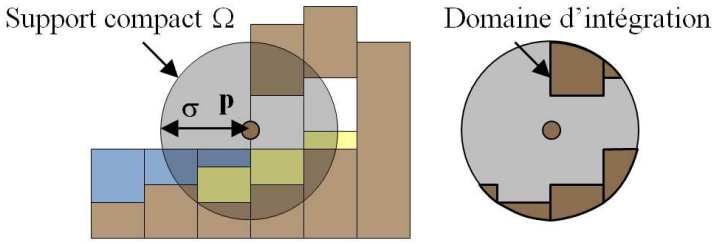


FIGURE 2.4 – Évaluation du domaine d'intégration avec un support compact sphérique.

2.1.2.2 Noyau de convolution à support compact rapide

La principale cause de la complexité de l'évaluation de l'intégrale est l'utilisation du support compact sphérique. Pour obtenir une continuité de classe C^1, cette intégration revient à évaluer le nouveau noyau suivant :

$$(h * g)(\mathbf{p}) = \int_{\Omega \cap \mathcal{B}} g(\mathbf{q})\, h(\mathbf{p} - \mathbf{q}) \mathrm{d}\mathbf{q} \qquad h(\mathbf{q}) = \begin{cases} 1 - \dfrac{\|\mathbf{q}\|}{\sigma} \text{ si } \|\mathbf{q}\| < \sigma \\ 0 \text{ sinon} \end{cases}$$

Le paramètre σ réfère au rayon du support compact du noyau de convolution. Cette

équation peut être simplifiée en considérant seulement une continuité de classe C^0. Dans notre cas, une continuité de classe C^0 est suffisante pour représenter des terrains. Cette simplification revient à évaluer la convolution par une intégration du domaine définie par l'intersection d'une sphère et l'union des boîtes :

$$(h * g)(\mathbf{p}) = \int_{\Omega \cap \mathcal{B}} g(\mathbf{q})\, h(\mathbf{p} - \mathbf{q}) \mathrm{d}\mathbf{q} \qquad h(\mathbf{q}) = \begin{cases} 1 \text{ si } \|\mathbf{q}\| < \sigma \\ 0 \text{ sinon} \end{cases}$$

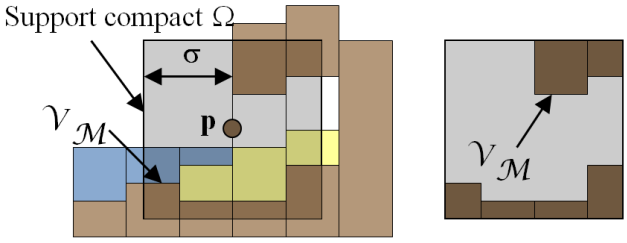


FIGURE 2.5 – Évaluation de la fonction de convolution : $i(\mathbf{p})$ est défini comme le volume de matière $\mathcal{V}_{\mathcal{M}}$ à l'intérieur du support compact Ω de la fonction noyau.

L'évaluation de l'intersection entre l'union des boîtes $\mathcal{B}$ et une sphère est relativement difficile. Pour accélérer encore les calculs et simplifier le calcul de l'intégrale, nous proposons d'utiliser un support compact cubique Ω pour définir la fonction noyau (Figure 2.5). L'arête du cube a une longueur de $2\,\sigma$. L'équation de l'évaluation de l'intégration sur un support compact cubique est la suivante :

$$h(\mathbf{q}) = \begin{cases} 1 \text{ si } \|\mathbf{q}\|_\infty < \sigma \\ 0 \text{ sinon} \end{cases} \quad \text{avec } \|\mathbf{q}\|_\infty = max(|x|, |y|, |z|)$$

Le terme $\|\mathbf{q}\|_\infty$ correspond à la norme de $\mathbf{q}$ sur une métrique de longueur infinie $\mathcal{L}^\infty$. Cette définition simplifie le calcul de l'intégrale $i(\mathbf{p})$ en une évaluation du volume de matière $\mathcal{V}_{\mathcal{M}}$ à l'intérieur du support compact de la fonction noyau (Figure 2.5). Cette évaluation est effectuée très efficacement en calculant la somme des volumes des boîtes des couches de matériaux qui intersectent le support compact cubique. Cette opération revient à évaluer le volume obtenu par l'intersection de plusieurs boîtes.

Le paramètre $\mathcal{V}_\Omega = 8\,\sigma^3$ représente le volume du support compact cubique Ω, la fonction potentiel $f(\mathbf{p})$ est alors définie par :

$$f(\mathbf{p}) = \frac{i(\mathbf{p})}{4\,\sigma^3} - 1 = 2\frac{\mathcal{V}_{\mathcal{M}}}{\mathcal{V}_\Omega} - 1$$

Cette définition garantit que $f(\mathbf{p}) = 1$ si la région cubique Ω autour de $\mathbf{p}$ contient seulement le matériau $\mathcal{M}$, et que $f(\mathbf{p}) = -1$ si Ω ne contient aucune couche de ce matériau. Lorsque la moitié du volume de Ω est rempli du matériau $\mathcal{M}$ alors $f(\mathbf{p}) = 0$.

La fonction $i(\mathbf{p})$ est la convolution de fonctions constantes par morceaux et est de classe C^0. Par conséquent, la fonction potentiel f est définie comme une fonction

linéaire de $i(\mathbf{p})$ est bien de classe C^0. Bien que des discontinuités puissent exister lors de l'évaluation du gradient ce qui a pour conséquence d'obtenir des discontinuités dans le calcul des normales à la surface et donc des discontinuités lors du calcul d'éclairement, les artefacts ne sont pas visibles après application de la texture.

2.2 Edition de la roche

Dans cette section, nous présentons les outils utilisés pour sculpter la couche de roche. Dans notre système, l'utilisateur peut sculpter la roche dans l'espace sans aucune contrainte de gravité ou de validité physique. Ceci lui permet de créer librement les surplombs qu'il souhaite et même de créer des rochers flottants dans l'air.

Les outils proposés permettent de sculpter la couche de roche intuitivement et interactivement. Dans les algorithmes détaillés ici, notre but est de proposer des outils de haut niveau pour créer des scènes complexes sans avoir le fardeau d'éditer manuellement l'ensemble des détails. Outre les outils classiques de sculpture utilisés pour les cartes d'élévation de données, pour la sculpture par ajout ou retrait de matière à l'aide de formes géométriques simples comme une sphère ou encore la conversion de maillage d'objet en roche, nous détaillons deux outils plus complexes adaptés à notre modèle en utilisant efficacement le modèle hybride. La première technique consiste à déformer et sculpter localement le terrain en direction de la normale à la surface. Comme la sculpture par ajout successif de matière est longue et fastidieuse pour sculpter de grands volumes, nous avons développé une deuxième technique pour créer des fissures et percer des réseaux de grottes.

2.2.1 Sculpture de la couche de roche

Rappelons que la roche est définie comme une surface implicite.

$$S = \{\mathbf{p} | f(\mathbf{p}) = 0\}$$

Cette représentation implicite, nous permet d'implémenter des techniques de sculpture implicite (FCG99) facilement dans notre système. La plupart des méthodes existantes ne sont pas adaptées à la surface de grandes scènes mais sont plutôt spécifiques à l'édition d'objets. Nous avons donc développé un outil spécifique pour ajouter ou retirer un volume de roche en direction de la normale à la surface du terrain.

L'outil de sculpture est paramétré par une région d'influence sphérique noté $\mathcal{R}$ de centre $\mathbf{c}$ et de rayon R (Figure 2.6). La fonction potentiel correspondante est définie à partir d'une primitive implicite créée à partir d'un squelette ponctuel (WGG99) : $f^\star(\mathbf{p}) = g \circ d(\mathbf{p})$, où $d(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}\|$ représente la distance euclidienne et $g(r)$ représente

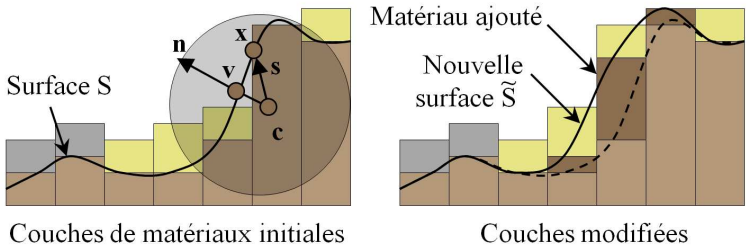


FIGURE 2.6 – Déplacement local de la surface de roche en mettant à jour les matériaux correspondant à l'intérieur de la région d'édition.

la distribution du potentiel autour du squelette :

$$g(r) = \begin{cases} \left(1 - \left(\frac{r}{R}\right)^2\right)^2 & \text{si } r < R \\ 0 \text{ sinon} \end{cases}$$

Lors de la modification de surfaces plus fines que les primitives implicites, l'ensemble de la surface va être déformée ce qui n'est souvent pas souhaitable pour un outil de sculpture. De ce fait, il est nécessaire de créer des primitives implicites orientées pour rendre plus intuitifs l'édition. Nous définissons automatiquement l'orientation de la surface implicite en fonction de la normale à la surface. Pour extraire la surface uniquement en direction de la normale, nous vérifions que le produit scalaire $\mathbf{n}.\mathbf{s}$ est positif ou $\mathbf{n}$ est la normale et $s = \mathbf{x} - \mathbf{c}$ le vecteur défini par le point courant sur la surface du terrain $\mathbf{x}$ et le centre de l'outil $\mathbf{c}$.

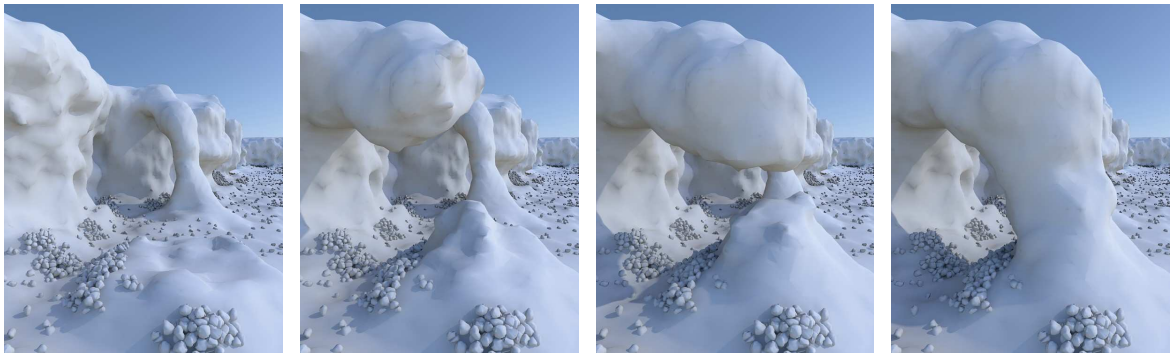


FIGURE 2.7 – Un surplomb sculpté en déplaçant progressivement la surface de roche en direction de sa normale.

Pour effectuer cette opération, l'utilisateur place l'outil au point $\mathbf{p}$ situé sur la surface du terrain. Avant l'édition, la fonction de potentiel du terrain est représentée par f et après le processus d'édition le champ de potentiel modifié est représentée par $\widetilde{f}$. Le processus global de sculpture pour ajouter de la matière à la surface de roche procède en 5 étapes :

1. Calculer la normale $\mathbf{n} = \nabla f(\mathbf{p}) / \|\nabla f(\mathbf{p})\|$ de la surface de roche au point $\mathbf{p}$.

2. Créer un point comme primitive implicite avec un centre légèrement à l'intérieur de la surface du terrain $\mathbf{c} = \mathbf{p} - \delta\,\mathbf{n}$ où δ représente la distance à la surface.
3. Calculer la fonction implicite modifiée $\widetilde{f} = f + f^{\star}$ en mélangeant le terrain avec la primitive implicite.
4. Mettre à jour localement les couches de matière de la roche et d'air qui intersectent la région d'influence $\mathcal{R}$ et redéfinir la fonction de potentiel f à partir des nouvelles couches de matières.
5. Enfin, stabiliser les différentes couches de matières à partir des couches modifiées dans R. Nous remarquons que l'étape 5 peut engendrer des calculs de stabilisation en dehors de R par éboulement.

Le paramètre $0 \leq \delta \leq R$ et le rayon R contrôlent le volume de matière ajouté à la surface. En pratique, $\delta = R/2$ donne de bons résultats pour sculpter la surface. L'algorithme fonctionne de la même manière pour creuser la roche : le centre est placé en dehors de la surface $\mathbf{c} = \mathbf{x} + \delta\,\mathbf{n}$ et la fonction potentiel $\widetilde{f}$ est modifiée en utilisant une fonction de mélange négative $\widetilde{f} = f - f^{\star}$.

Les matériaux qui intersectent $\mathcal{R}$ sont subdivisés en couche d'air ou de roche selon le potentiel de la fonction champ de potentiel modifié $\widetilde{f}$. Ce résultat est obtenu en résolvant l'équation $\widetilde{f} = 0$ qui représente les limites supérieures et inférieures des couches de roches le long de l'axe central de la couche de matière.

2.2.2 Création de fissures et de grottes par balayage

En pratique, il est très long et fastidieux de sculpter manuellement de grandes scènes par ajout ou retrait successif de matière avec l'outil de sculpture présenté précédemment. En effet, la sculpture se prête plus à l'édition de détails sur la surface. C'est pour cela qu'il y a des besoins d'outils génériques pour créer une variété de caractéristiques de terrain. Nous proposons des techniques spécifiques de balayage pour créer des fissures, des gorges et des tunnels.

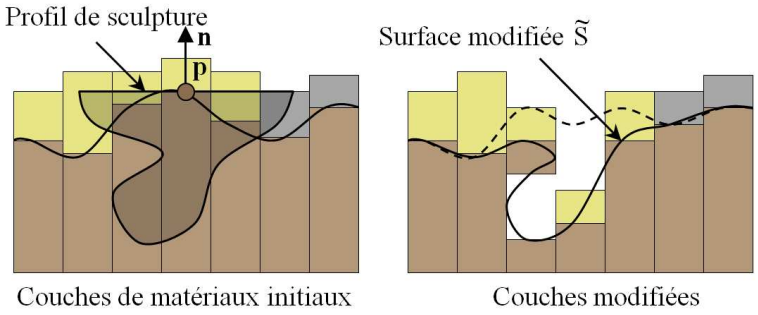


FIGURE 2.8 – Exemple de fissure sculptée dans la couche de roche avec un profil de courbe non convexe.

Notre méthode reprend l'idée générale de la génération de fissures proposée dans les travaux de Desbenoit (DGA05). Leur méthode permet de créer des fissures sur des

maillages comme des bancs de bois, des statues de pierres ou encore sur du verre. Nous avons adapté leur méthode afin de l'intégrer dans notre structure de données. L'outil est défini par un graphe $\mathcal{G}$ représentant la structure des branches du chemin. La forme de l'outil est paramétrée par un ensemble de courbe de profil $\mathcal{C}$ représentant les sections de l'outil de sculpture. Les nœuds du graphe stockent différentes courbes de profils définissant les sections. Le volume correspondant est produit en traversant récursivement le graphe et en interpolant les courbes de profils en suivant les arcs des graphes. Les courbes de profils sont définies par des splines cubiques par morceaux et peuvent définir des sections non convexes (Figure 2.8).

FIGURE 2.9 – Un ravin généré en balayant des courbes de profil complexe.

Soit f la fonction potentiel du terrain. L'algorithme procède en 4 étapes :

1. Les fissures sont créées en projetant le graphe $\mathcal{G}$ sur la surface du terrain pour créer un squelette $\mathcal{S}$. Cette étape peut être effectuée simplement en utilisant la représentation implicite du terrain.
2. Ensuite, nous définissons les courbes de profils aux nœuds du squelette en plaçant les courbes de profils du modèle définies à chaque nœud et en les orientant selon la normale locale ∇f de la surface de l'objet et la tangente de la courbe du squelette. Nous définissons le volume de la fissure $\mathcal{V}$ en balayant et interpolant les courbes de profils le long du squelette. La fonction potentiel définie par le volume sera notée $f^\star$.
3. Nous creusons le volume de fissure $\mathcal{V}$ sur le terrain en calculant la différence Booléenne entre le terrain et le volume $\mathcal{V}$. Cette étape est effectuée efficacement avec la représentation implicite de la surface en définissant la fonction de champ de potentiel comme $\widetilde{f} = \min(f, -f^\star)$.
4. Nous stabilisons l'ensemble des piles de matériaux modifiées (Figure 2.9). Nous remarquons que l'étape 5 peut engendrer des calculs de stabilisation en dehors de R par éboulement.

Comme pour la sculpture de la roche, la hauteur des couches de matériaux modifiées est calculée en résolvant l'équation $\widetilde{f} = 0$ le long de l'axe central de la couche de matière modifiée.

Le processus de modélisation est le même pour les grottes et tunnels, excepté que nous ne projetons pas le graphe $\mathcal{G}$ sur la surface du terrain. Dans ce cas, le graphe

représente directement la trajectoire du tunnel à l'intérieur de la roche. Cet outil est utilisé pour définir des falaises en utilisant une fonction potentiel $\widetilde{f} = \max(f, f^\star)$ pour ajouter de la matière.

2.3 Édition des matériaux granuleux

Dans cette section, nous présentons des outils spécifiques pour manipuler les couches de matériaux de dépôt (sable, pierres, neige). Ces matériaux permettent d'ajouter des détails dans la scène afin de la rendre plus réaliste. Nous proposons des outils classiques de dépôt de matière mais également des outils de simulation d'érosion pour obtenir des résultats réalistes rapidement et intuitivement. Les outils d'érosion permettent de transformer la roche en matériaux de dépôts pour obtenir les sédiments. Ces outils s'appuient sur un processus de stabilisation afin de distribuer la matière naturellement selon son angle au repos.

2.3.1 Ajout de matériaux granuleux

Nous avons développé des outils génériques pour déposer des couches de matières granuleux dans une scène. Notre approche est une généralisation tridimensionnelle des outils utilisés pour éditer des cartes d'élévation de données. Elle permet de déposer sur le terrain l'ensemble des matériaux granuleux que ce soit le sable, la terre ou les pierres. Notre modèle est fortement lié au moteur physique de stabilisation des couches de matériaux pour distribuer naturellement la matière dans la scène tout en simplifiant l'interaction de l'utilisateur avec le système.

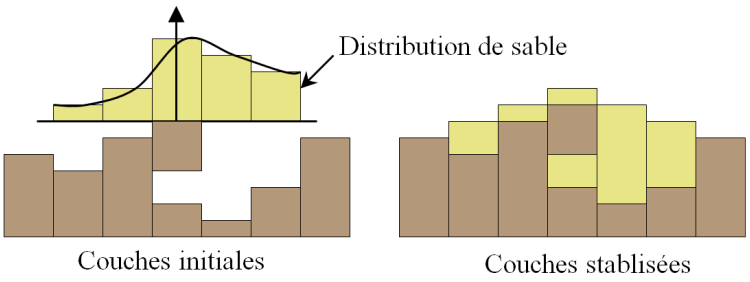


FIGURE 2.10 – Ajout de sable sur une couche de roche.

L'outil de dépôt de matière est caractérisé par un profil de distribution(Figure 2.11). Notre outil de dépôt de matière est défini comme une brosse. Cette brosse est soit projetée sur la surface du sol comme pour les cartes d'élévation, soit elle est positionnée dans l'espace tridimensionnel pour ajouter de la matière dans une grotte par exemple. La brosse est caractérisée par une région de dépôt $\mathcal{R}$ et est centrée autour d'un point $\mathbf{p}$. La distribution de matière dans la région $\mathcal{R}$ est définie comme une carte d'élévation de données pour représenter la quantité à déposer sur le terrain (Figure 2.10).

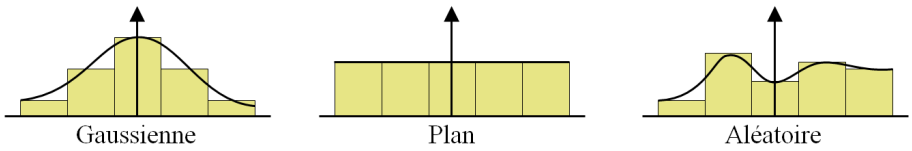


FIGURE 2.11 – Différents profils de distribution.

Le dépôt de matière procède en deux étapes :

1. Nous ajoutons les couches de matières sur le plan horizontal contenant $\mathbf{p}$ en insérant l'épaisseur de matière δh_i. L'épaisseur de matière $\delta\tilde{h}_i$ ajoutée à la scène est définie en fonction de l'épaisseur d'air δh_{air} disponible :

$$\Delta\tilde{h}_i = \begin{cases} \Delta h_{air} \text{ si } \Delta h_{air} < \delta h_i \\ \Delta h_i \text{ sinon} \end{cases}$$

   Aucune épaisseur de matière $\Delta\tilde{h}_i$ n'est ajoutée à la scène si le point $\mathbf{p}$ se situe dans une couche de roche.

2. Le moteur physique stabilise les couches instables pour obtenir une distribution réaliste des matériaux sur le terrain, tout en supprimant les superpositions de matériaux incompatibles.

FIGURE 2.12 – Couches de sable, de pierres et de terre (couverte d'herbe) ajoutées successivement sur des couches de roches.

Cette méthode permet de contrôler la quantité de matière déposée ainsi que son placement dans l'espace. Pour éviter que l'utilisateur ne définisse la forme de la brosse, des formes par défaut sont proposées.

2.3.2 Érosion globale

La majorité des techniques de simulation d'érosion à grande échelle érode la couche de roche par des frottements de la roche avec un fluide pour produire des sédiments qui sont déplacés dans la scène par une simulation de transport (NWD05, BF02, MDH07). Ces techniques sont efficaces pour des grands terrains de l'ordre de plusieurs dizaines

de kilomètres afin de les lisser ou de créer des vallées. Comme nous nous situons sur des terrains de moyenne échelle, nous ne pouvons pas utiliser les modèles classiques d'érosion de terrain. Nous proposons une méthode permettant d'éroder la scène pour sculpter la roche et ajouter automatiquement des détails sur des reliefs de moindre échelle : 100m à 2km. Notre méthode s'apparente à la technique d'érosion de Beardall (BFO+07) pour générer des Goblins. Les Goblins sont des monticules de roches s'apparentant à des champignons. Ils sont obtenus par une forte érosion de la roche composée de roche très dure et moins dure. De nombreux Goblins se situent dans la région de l'Utah aux Etats Unis.

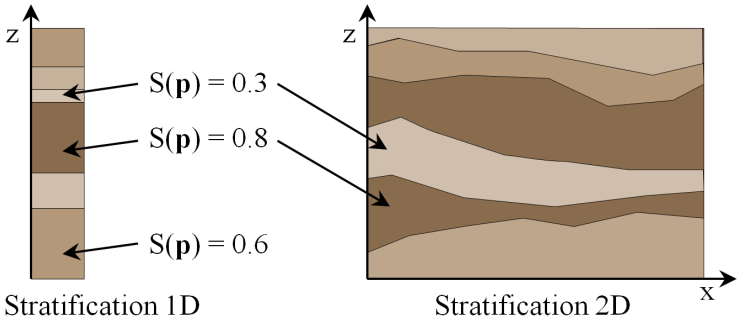


FIGURE 2.13 – Deux représentations de stratifications de roche utilisées.

Pour pouvoir éroder le terrain, nous utilisons un modèle de stratification du terrain défini par une fonction tridimensionnel $\mathcal{S}(\mathbf{p})$ retournant pour tout point de l'espace $\mathbf{p}$ la granularité et la dureté de la roche. Ces fonctions de strates peuvent être définies manuellement ou procéduralement. Une représentation de stratification tridimensionnelle a été proposée dans (RPP93) pour générer des cartes d'élévation de données par une érosion des couches supérieures. Plusieurs styles de strates peuvent être obtenus en fonction de l'état géologique et de l'activité sismique de la région. Les stratifications peuvent être régulières comme sur les falaise de bord de mer. Ce type de stratification peut être représenté par une fonction à une dimension définie par une pile de paramètres définissant la dureté et la granularité de la roche. Les stratifications peuvent être plus chaotiques à cause des collisions des plaques tectoniques en montagne. Nous définissons ce type de stratification à l'aide de textures 3D, ces textures sont générées procéduralement par des fonctions de bruits, ou par combinaison successive de cartes d'élévation générées procéduralement.

Pour éroder la surface du terrain, nous effectuons une discrétisation en voxel de la surface de notre terrain afin de affecter à tout voxel de centre $\mathbf{p}$ une valeur d'érosion η. Étant donné la fonction de stratification du terrain $\mathcal{S}(\mathbf{p})$ sur l'ensemble du terrain, le processus d'érosion est le suivant :

1. Nous générons uniquement les voxels sur la surface de roche apparente. Seul les couches de roches voisines d'une couches d'air sont converties en voxel afin de ne pas surcharger la mémoire. Nous associons à chaque voxel une valeur d'érosion η qui est initialisée à $\eta = 1$ pour définir un voxel non érodé.
2. Nous érodons itérativement les voxels selon un nombre fixé par l'utilisateur. A

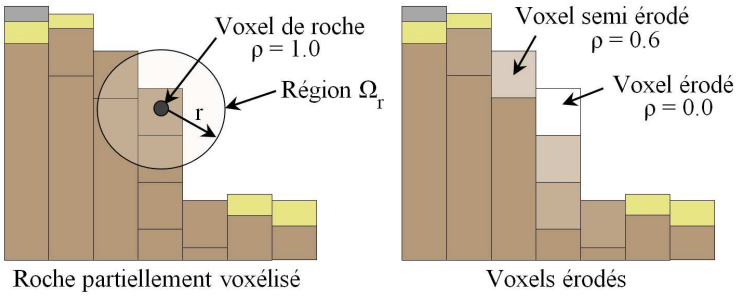


FIGURE 2.14 – Génération des voxels de la surface de la roche et processus d'érosion itératif.

chaque itération, nous calculons la nouvelle valeur d'érosion $\tilde{\eta}$:

$$\tilde{\eta} = \eta - a(\mathbf{p})\, S(\mathbf{p})$$

avec a l'accessibilité du voxel dans une région sphérique Ω de rayon r (BFO+07). Lorsque $\eta = 0$ le voxel de roche est transformé en sédiment, les matériaux de sédiment sont obtenus en fonction de la rugosité de la matière. Le voxel est transformé en couche de pierres ou en sable en fonction d'un seuil associé à la rugosité du matériau.

3. Enfin, nous calculons le transport et la stabilisation des matériaux.

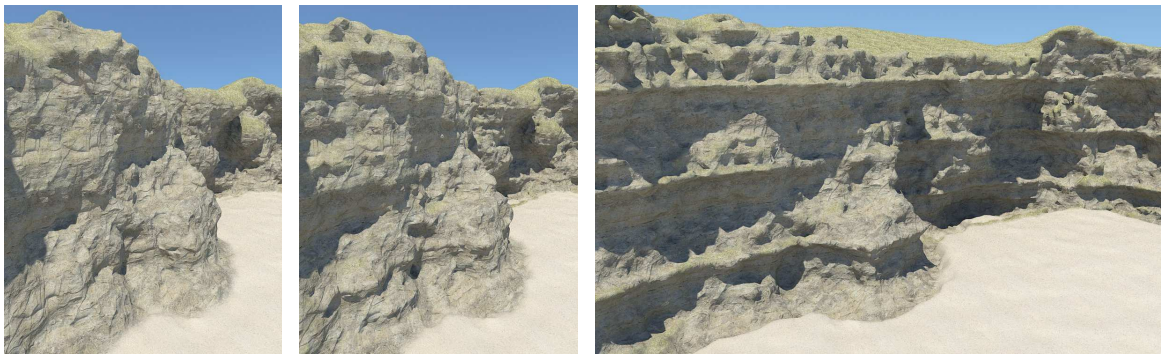


FIGURE 2.15 – Etapes d'érosion d'une falaise, les couches molles définies par la fonction de stratification $\mathcal{S}(\mathbf{p})$ ont été érodées plus rapidement que les couches dures qui apparaissent dans l'image comme des traits fins.

2.3.3 Érosion locale

Nous proposons un outil d'érosion local pour reproduire le détachement des pierres de la paroi se rassemblant en tas sur le sol. Cet outil permet à l'utilisateur de déclencher manuellement l'éboulement d'une partie de la roche. Ce mode d'érosion est souvent dû dans la nature à des successions de gel et dégel de la roche. Ce phénomène de gel et dégel est très complexe car il dépend des diaclases dans lesquels l'érosion est effectuée. Pour

l'érosion d'une falaise, ce phénomène est directement lié à l'épaisseur de la diaclase. Les diaclases sont des fractures de la roche qui sont souvent orientées perpendiculairement aux limites de stratification.

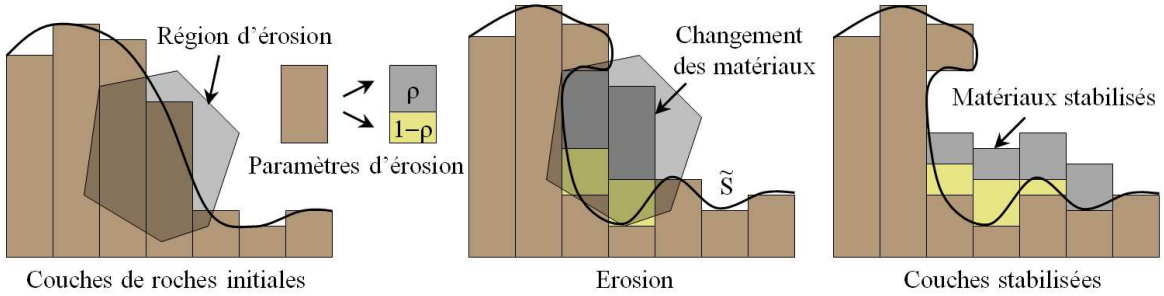


FIGURE 2.16 – Processus d'érosion global : la roche se transforme en sable et en pierres qui sont transportés par le processus de stabilisation.

Le graphiste définit un outil d'érosion caractérisé par la forme de sa région d'influence, notée $\mathcal{R}$, de façon à transformer la roche en différents matériaux de type granuleux (Figure 2.16). Dans notre implémentation, la roche peut être érodée en sable et en pierres. Notre outil est contrôlé par le paramètre $\rho \in [0, 1]$ définissant le volume relatif de roche ou de sable produit par l'érosion. L'algorithme d'érosion procède en deux phases :

1. L'érosion est effectuée en déplaçant l'outil sur la surface du terrain. Nous simulons le détachement des pierres de la roche en modifiant les couches de roche qui intersectent la région d'érosion $\mathcal{R}$ et en les transformant en matériau de sable et de pierre selon le paramètre ρ définissant le pourcentage effectif de pierre et de sable issu de l'érosion (Figure 2.16).
2. La stabilisation des couches de matière transporte automatiquement les matériaux érodés à leur position de repos.

FIGURE 2.17 – Etapes d'érosion locale d'une falaise.

Nous proposons un processus de génération de la région d'influence $\mathcal{R}$ par une forme convexe. Cette forme est générée comme la cellule de Voronoï issue du point central $\mathbf{p}$ de l'outil. La cellule est obtenue en distribuant aléatoirement autour du point $\mathbf{p}$ un ensemble de points. Ces points sont générés à une distance minimale $r + \epsilon$ pour définir

une taille moyenne de la région d'édition. Pour obtenir un résultat qui s'adapte à la stratification, nous définissons r tel que r soit supérieur ou égal à l'épaisseur de la strate courante.

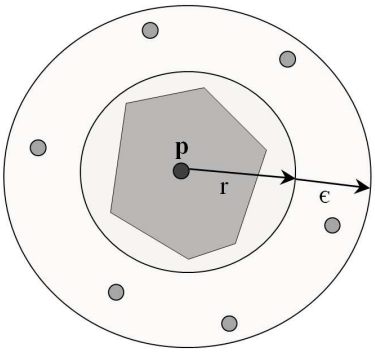


FIGURE 2.18 – Génération de la région d'érosion.

2.3.4 Simulation de la stabilisation des matériaux granuleux

La stabilité des matériaux est définie par un angle de repos de la matière. L'angle de repos entre deux matériaux est défini comme l'angle entre le plan horizontal et le plan de contact entre deux matériaux. Cet angle est défini selon les caractéristiques du matériau, à savoir la nature des particules (adhérence) et la géométrie des particules.

Pour obtenir un résultat réaliste lors de l'édition des couches de matériaux granuleux (sable, terre, pierres), nous effectuons une étape de simulation de stabilisation sur ces matériaux. Dans la littérature plusieurs travaux ont été proposés pour stabiliser un matériaux transporté par un fluide ou non (MKM89, BF01). Nous proposons une méthode pour stabiliser les matériaux granuleux dans un environnement de topologie arbitraire.

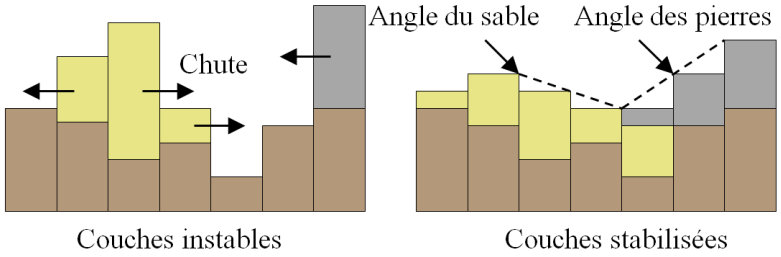


FIGURE 2.19 – Stabilisation des couches de matériaux selon l'angle au repos des différents matériaux.

Nous stabilisons tout d'abord les couches inférieures de la pile puis les couches supérieures. Cela permet de traiter à chaque étape une seule fois chaque couche de matière. Notre simulation de stabilisation est effectuée en déplaçant la matière d'une pile à une autre selon son angle de repos α défini en fonction de chaque matériau

(Figure 2.19). Dans notre implémentation, nous utilisons les paramètres suivant d'angle au repos en considérant les matériaux secs et homogènes (Tableau 2.3).

Matériau	Angle
Sable	$30 - 35$
Pierres	$40 - 45$
Terre	$25 - 40$
Neige	$40 - 80$

TABLEAU 2.3 – Angle au repos (en degrés) pour la stabilisation des différents matériaux : degrés pour le sable, degrés pour les pierres et degrés pour la terre.

Soit h la hauteur d'un matériau donné de sable, de pierre ou de terre et soit h_i avec $i \in [1, 8]$ la hauteur de ses 8 voisins(Figure 2.20). Nous définissons la différence de hauteur entre la pile centrale et ses voisins par $\Delta h_i = h_i - h$. Nous déterminons $\Delta h = \max(\Delta h_i)$ comme la différence de hauteur maximale.

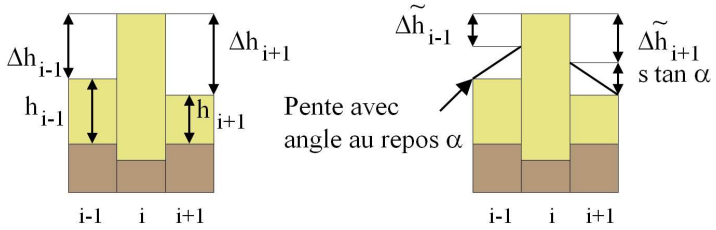


FIGURE 2.20 – Notations pour l'algorithme de stabilisation.

Le processus de stabilisation est le suivant :

1. Détection des piles instables lors de l'édition du terrain en comparant l'angle entre les piles et l'angle de repos a du matériau.
2. Calcul de l'épaisseur de matière $\Delta\tilde{h}_i$ à transporter. $\Delta\tilde{h}_i$ est défini en fonction de la quantité d'air disponible Δh_{air} sur la pile voisine (Figure 2.21) et les caractéristiques du matériau.

$$\Delta\tilde{h}_i = \begin{cases} 0 \text{ si } \Delta h_i < s \tan\alpha \\ \Delta h_{air} \text{ si } \Delta h_{air} < s \tan\alpha \\ \Delta h_i - s \tan\alpha \text{ sinon} \end{cases}$$

3. Nous déplaçons le volume de matière Δz_i sur la pile voisine i. L'épaisseur de matière à déplacer est définie comme une constante a pour éviter les oscillations de l'algorithme. L'épaisseur de matière à déplacer de la pile centrale aux piles voisines Δz_i est définie comme une moyenne pondérée proportionnelle à la différence de hauteur :

$$\Delta z_i = a \frac{\Delta\tilde{h}_i}{\sum_{i\in[1,8]} \Delta\tilde{h}_i}$$

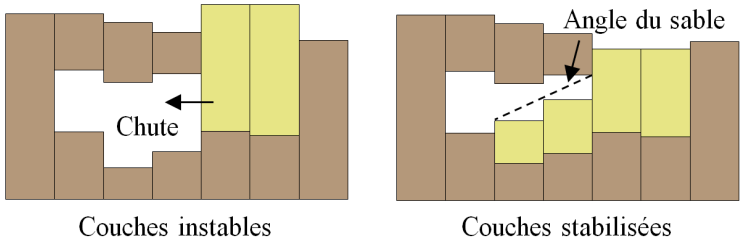


FIGURE 2.21 – Stabilisation du sable dans une grotte.

Ce processus est répété itérativement jusqu'à ce que l'ensemble des matériaux soit stabilisé.

FIGURE 2.22 – Simulation de stabilisation d'un tas de terre et d'un tas de pierres.

2.4 Création de lacs et de plans d'eau

Dans cette partie, nous proposons des outils de remplissage et d'écoulement d'eau et de changement de phase qui ne s'appuient pas sur une simulation physique de stabilisation (Figure 2.23). Dans notre étude, nous nous sommes intéressés à la position statique de ces éléments pour simplifier les problèmes liés à la dynamique des fluides. Donc dans cette partie, nous n'étudions pas la simulation d'écoulement continu de l'eau dans les rivières (YNBH09).

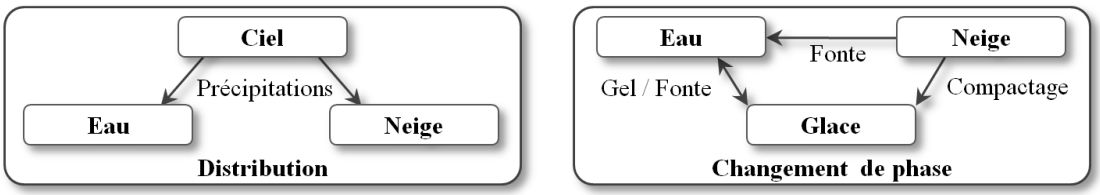


FIGURE 2.23 – Différents outils d'édition de l'eau définis dans notre système : distribution et changement de phase.

Nous proposons des outils de remplissage s'appuyant sur un moteur de stabilisation

pour approximer l'écoulement des fluides et obtenir un résultat final réaliste. La stabilisation de l'eau nécessite un algorithme spécifique différent de celui des matériaux granuleux. Enfin, une originalité de notre modèle multi-matériau est de permettre la mise au point d'outils de changement de phase pour faire fondre la neige et la glace en eau ou pour geler l'eau en glace.

2.4.1 Remplissage d'une région par propagation

L'outil de dépôt proposé pour ajouter des matériaux granuleux est également utilisé pour ajouter ou retirer un volume d'eau à la scène. Ce volume d'eau nécessite d'être ensuite stabilisé, cette étape de stabilisation est longue et coûteuse sur de grands volumes d'eau. Nous proposons un outil de remplissage d'eau pour générer rapidement des étendues d'eau telles que des flaques ou des lacs. En comparaison à l'outil de dépôt de matière, l'utilisateur contrôle la hauteur maximale du lac mais ne contrôle pas la quantité de matière générée.

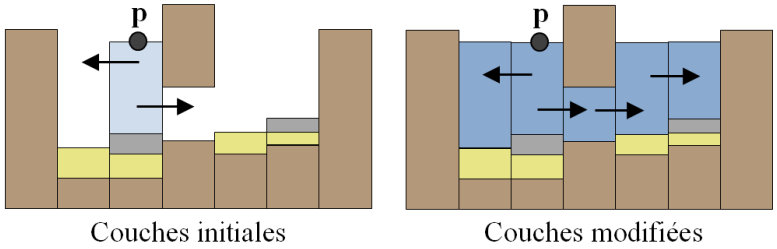


FIGURE 2.24 – Remplissage par connexité à une hauteur prédéfinie.

Une couche d'eau l_i est définie par un intervalle de hauteur $[a_i, b_i]$ où a_i et b_i représentent respectivement la hauteur minimale et maximale de la couche l_i. Étant donné une position $\mathbf{p}$ située dans une couche d'air pour définir la hauteur h représentant la hauteur maximale de l'étendue d'eau, le processus de remplissage d'eau est le suivant (Figure 2.24). Nous ajoutons une couche d'eau de hauteur h_i sur la pile i telle que la hauteur maximale de la couche d'eau l soit égale à h. Nous initialisons la liste $\mathcal{Q}$ avec la couche d'eau courante l_i. La liste $\mathcal{Q}$ correspond au front de propagation du lac à générer. L'algorithme s'arête lorsque la liste $\mathcal{Q}$ est vide.

1. Pour toutes les couches l_i de la liste $\mathcal{Q}$, nous évaluons la quantité d'eau à ajouter sur l'ensemble des couches voisines l_j avec $j \in [0, 8[$ ses 8 voisins. Cette quantité correspond au volume d'air disponible entre le sol et le point $\mathbf{p}$.
2. La propagation sur une couche voisine est effectuée lorsque l'intersection de l'intervalle de hauteur entre la couche courante l_i et la couche voisine l_i est non nulle. Si $l_i \cap l_j \neq \emptyset$ alors la couche d'eau est générée avec l'intervalle $[a_i, b_i]$ si $b_i < h$ ou $[a_i, h]$ si $b_i > h$. La couche d'eau générée l_j est ajoutée à la liste $\mathcal{Q}$.
3. Après avoir analysé l'ensemble des voisins, la couche l_i est retirée de la liste $\mathcal{Q}$.

Comme nous étudions seulement la position statique des éléments, cette méthode est très efficace pour générer rapidement de grandes étendues d'eau directement stable.

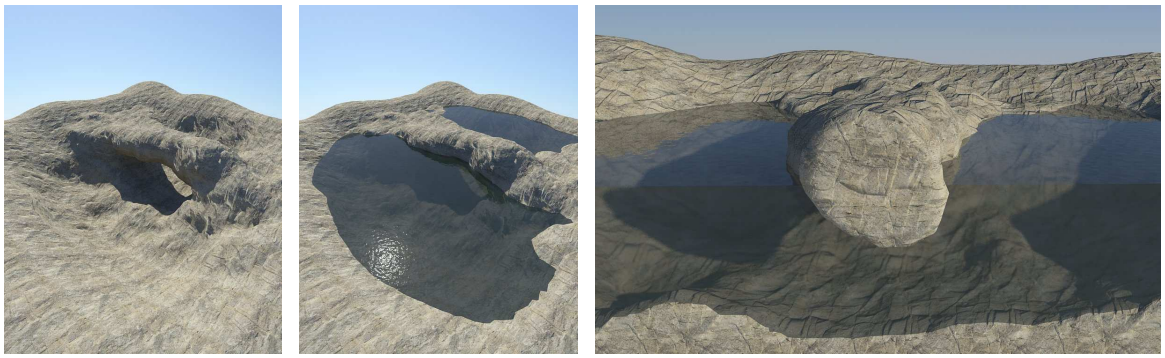


FIGURE 2.25 – Génération automatique d'un lac par la méthode de propagation.

2.4.2 Simulation de l'écoulement

Nous proposons un processus simplifié d'écoulement de l'eau pour que l'ensemble des couches d'eau connexes aient la même hauteur. Notre méthode permet de reproduire correctement la stabilisation de l'ensemble du volume d'eau même avec la présence de siphons. Notre méthode ne simulant pas la mécanique des fluides ni l'interaction fluide matière, les matériaux granuleux ne sont pas entrainés par l'écoulement.

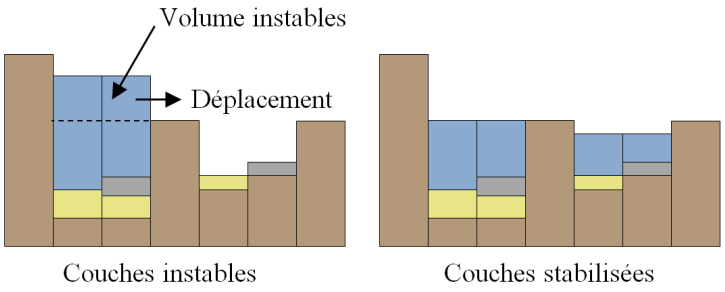


FIGURE 2.26 – Écoulement de l'eau.

Pour stabiliser les volumes d'eau, nous cherchons à garantir que la différence de hauteur Δ_h entre les piles connexes soit proche de 0. Le processus de stabilisation d'un volume d'eau que nous proposons est le suivant :

1. Nous définissons l'ensemble $\mathcal{E}$ regroupant l'ensemble des listes de couches $\mathcal{Q}$. Chaque liste $\mathcal{Q}$ regroupe l'ensemble des couches d'eau l_i connexes. Une couche d'eau est voisine d'une autre lorsque l'intersection des intervalles de hauteur des deux couches n'est pas nulle. L'opération de propagation s'effectue sur l'ensemble des 8 voisins de la couches d'eau l_i courante.
2. Pour chaque liste de couches $\mathcal{Q}$, nous recherchons les couches l_i instables. Une couche l_i est instable lorsque une couche d'air voisine intersecte l'intervalle de hauteur de la couche d'eau l_i. Chaque couche d'air est stockée dans une liste $\mathcal{L}$.
3. Ajout de nouvelles couches d'air dans la liste $\mathcal{L}$, lorsque la couche d'air courante contient une couche d'air voisine de hauteur minimale inférieure à la couche

courante. La recherche s'effectue itérativement en utilisant la couche d'air ayant la hauteur minimale. La recherche s'arête lorsque la couche d'air courante n'a pas de voisins vérifiant les critères d'extension.

4. Nous déplaçons une partie de la couche d'eau de hauteur maximale de $\mathcal{Q}$ vers la couche d'air l_k de hauteur minimale de la liste $\mathcal{L}$ ou si $\mathcal{L}$ est vide vers une couche d'eau de hauteur minimale de $\mathcal{Q}$. Si la couche l_k n'est pas connexe à une couche d'eau de $\mathcal{Q}$, nous générons une nouvelle liste. L'algorithme reprend à l'étape 3. tant qu'elle ne vérifie pas la condition de stabilité.

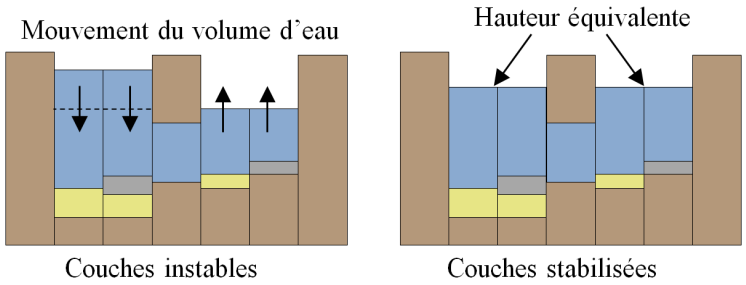


FIGURE 2.27 – Processus de stabilisation des couches d'eau à un même niveau.

L'obtention d'un état stable pour l'ensemble des couches d'eau peut prendre quelques secondes voir quelques minutes selon la quantité d'eau à traiter et la géométrie du terrain. Par contre nous obtenons des effets se rapprochant plus de la réalité que les méthodes proposées précédemment.

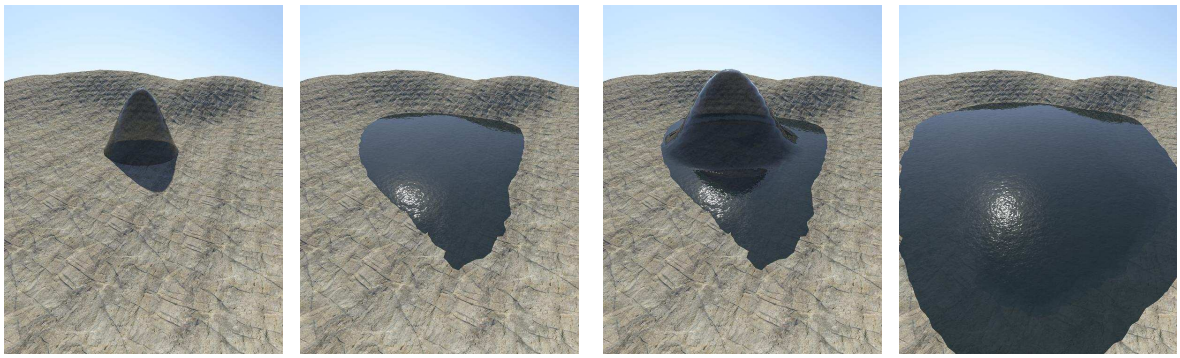


FIGURE 2.28 – Deux étapes de stabilisation de l'eau.

2.4.3 Changement de phase : outil de glaciation et de fonte

En fonction de la température ambiante, l'eau change de phase pour être soit en phase liquide (eau), soit en phase solide (glace, neige). Récemment un système de génération de paysage hivernal (MrGrG$^+$10) a été proposé pour simuler l'ensemble des transferts thermiques afin d'obtenir une grande variété de scène évoluant selon une

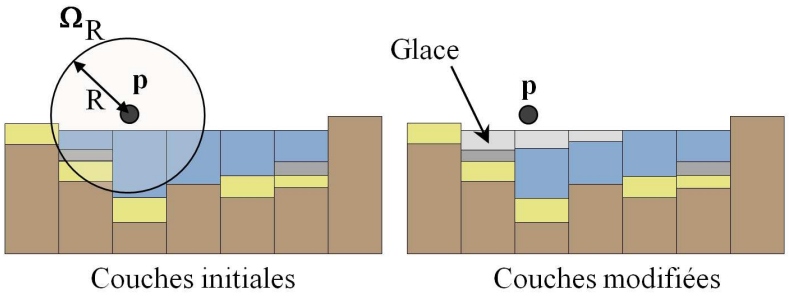


FIGURE 2.29 – Processus de changement de phase.

trame météorologique. Cette méthode est cependant très couteuse en temps de calcul et est inadaptée à l'édition interactive.

Nous proposons donc un outil de changement de phase permettant de modifier interactivement et intuitivement les couches d'eau, de glace et de neige. Cet outil est paramétré par une zone d'influence Ω de rayon R. Nous définissons un champ de potentiel $C(\mathbf{p})$ à l'intérieur de la zone Ω_R afin de contrôler la puissance de changement de phase qui s'opère aux interfaces. Ce champ de potentiel est contrôlé par une intensité de changement de phase I pouvant être définie en fonction d'une température ambiante t.

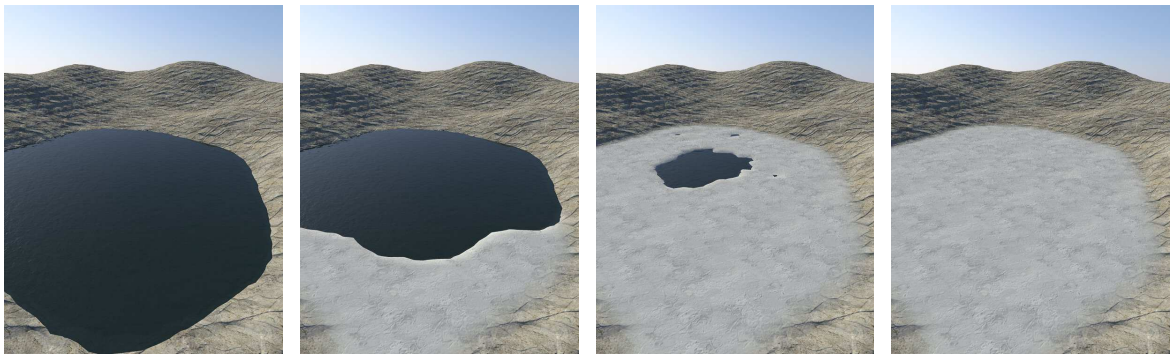


FIGURE 2.30 – Changement de phase de l'eau en glace d'un lac.

Le graphiste définit une position $\mathbf{p}$ dans l'espace à l'interface entre une couche d'eau et d'air. L'édition est effectuée sur un domaine Ω de rayon R avec une intensité I représentant l'épaisseur maximale de glaciation ou de fonte. Sans perte de généralité, nous détaillons le processus de changement de phase pour transformer l'eau en glace (Figure 2.29) :

1. Pour l'ensemble des couches d'eau l_i tel que $l_i \cap \Omega_R \neq \emptyset$, nous évaluons l'épaisseur e de glace à créer à la position du point $\mathbf{q}$ situé entre l'interface d'eau de la couche l_i et de la couche supérieure (air, glace). Nous analysons le volume d'eau $\mathcal{V}$ dans un domaine Ω_r de rayon r afin de définir l'épaisseur de glace e à générer. L'épaisseur

de glace finale est obtenue par l'équation suivante :

$$e = \left(1 - \frac{d(\mathbf{p}, \mathbf{q})}{R}\right)^2 \left(1 - \frac{\mathcal{V}}{\mathcal{V}_{max}}\right)^2 I$$

Le premier terme de l'équation permet de pondérer l'intensité du changement de phase en fonction de la distance au point $\mathbf{p}$ et le deuxième en fonction du volume d'eau présent pour transformer l'eau en glace plus rapidement lorsque le volume d'eau est faible.

2. Une partie de la couche d'eau d'épaisseur e est convertie en glace.
3. L'étape de stabilisation est effectuée pour stabiliser les couches d'eau devenues instables lors de la fonte de la glace ou de la neige.

L'utilisateur peut contrôler naturellement la zone de propagation de la glace, car l'analyse du volume d'eau environnant a pour effet de rajouter une plus importante épaisseur de glace lorsque la couche se trouve sur une rive. Les calculs sont les mêmes pour transformer la neige en glace ou pour faire fondre la neige ou la glace en eau, à la seule différence que le point $\mathbf{q}$ se situe entre la couche courante l_i et la couche inférieure.

2.5 Visualisation

Dans cette section, nous présentons une méthode pour visualiser l'ensemble de notre structure de données afin d'obtenir un résultat réaliste. En entrée, la scène est composée de notre structure hybride définie par plusieurs matériaux. L'ensemble de ces matériaux sont de nature varié et demandent d'être visualisés différemment en fonction de leurs caractéristiques. Dans notre modèle, nous pouvons extraire trois types de visualisation. Le premier type regroupe les matériaux formant le sol. Nous considérons que la roche et l'ensemble des matériaux continus (sable, terre, neige, glace) font partie du sol. Le deuxième type est défini par les matériaux liquides, soit l'eau. Enfin le troisième type regroupe les matériaux nécessitant des instances pour être visualisés pour obtenir un résultat plus réaliste comme les pierres.

FIGURE 2.31 – Processus de visualisation d'un terrain complexe.

Le pipeline de notre processus de visualisation s'effectue en 3 étapes (Figure 2.31) :

1. Génération du maillage de la surface du sol
2. Génération du maillage des surfaces des volumes d'eau
3. Génération des maillages de chaque pierre

Nous générons un unique maillage texturé pour rendre l'ensemble des couches de roche, de sable, de terre, de glace et de neige. La texture du maillage de la surface du sol est obtenue par une méthode de mélange de textures. Le maillage de l'eau est obtenu par une extraction du plan des couches d'eau connexes. La création du maillage de l'ensemble des pierres empilées les unes sur les autres est très complexe. Pour résoudre ce problème particulier, nous proposons une méthode originale de création d'empilement de maillage de pierres dans le Chapitre 3.

2.5.1 Génération du maillage du terrain

Nous devons extraire deux types de maillage de notre structure de données. Le premier type correspond à un maillage de géométrie et de topologie quelconque pour représenter le sol. Ce type est défini par le matériau roche définissant les arches, surplombs et grottes et les matériaux sable, terre, neige et glace. Le deuxième type de maillage à générer représente les différents volumes d'eau contenus dans la scène. En considérant que les couches d'eau sont stabilisées, les volumes peuvent être représentés par des cartes d'élévation de données.

2.5.1.1 Génération du maillage

Nous générons un unique maillage triangulaire pour représenter le sol. Le sol peut être de topologie et de géométrie quelconque car il est composé de surplombs, d'arches et de grottes. Nous obtenons ce maillage par une triangulation de la représentation implicite $S = \{\mathbf{p} \in \mathsf{R}^3 | f(\mathbf{p}) = 0\}$ du terrain. Nous générons une unique surface implicite en utilisant le squelette défini par l'union des matériaux de surface $\cup\{\mathcal{M}_i\}$ contenant la roche, le sable, la terre, la neige et la glace. Ce squelette est composé de l'ensemble des boîtes définissant les couches de matières (Figure 2.32) :

$$g(\mathbf{p}) = \begin{cases} 1 \text{ si } \mathbf{p} \in \cup\{\mathcal{M}_i\} \\ 0 \text{ sinon} \end{cases}$$

Pour générer le maillage du sol, nous utilisons les techniques classiques de triangulation de surface implicite (BS91, BW97) par Marching Cubes. Ce choix nous permet de contrôler le nombre de triangles générés pour nos scènes en ajoutant la taille de la grille. Les maillages de terrains représentés dans ce chapitre comptent un peu moins de 1 million de triangles (Tableau 2.4).

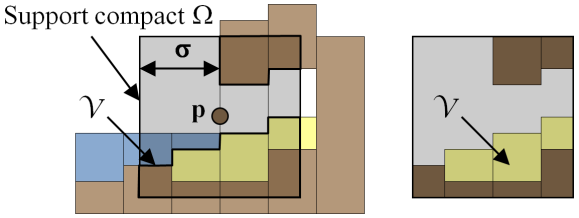


FIGURE 2.32 – Évaluation de la fonction de convolution : i(p) est défini comme la somme des volumes de matières de la surface à l'intérieur du support compact de la fonction noyau.

Terrain	Triangles	Mémoire
Grotte	818 940	21 Mo
Canyon	760 769	14 Mo
Falaises	742 615	26 Mo

TABLEAU 2.4 – Statistiques sur le nombre de triangles générés et la quantité de mémoire de stockage de notre modèle.

2.5.1.2 Génération de la texture

Nous obtenons à cette étape, un unique maillage du sol de géométrie et de topologie quelconque issu de plusieurs matériaux. Deux méthodes sont envisagées pour texturer ce maillage : un modèle par projection de texture et un modèle par génération procédurale de la texture (EMP+98). Actuellement, générer procéduralement la texture du terrain est long et coûteux car nécessite de définir pour tout point du maillage une couleur en fonction du matériau environnant. Cette couleur est obtenue par une combinaison de fonctions complexes définissant l'ensemble des matériaux. Cette méthode ne permet pas de visualiser la scène en temps réel. C'est pour cela que nous avons défini une nouvelle méthode de projection de texture afin de visualiser le terrain en temps réel.

Projection de texture

Pour visualiser l'ensemble des matériaux sur un maillage de topologie et géométrie quelconque, nous proposons de projeter des textures sur le terrain. Comme nous utilisons des terrains de topologie et de géométrie arbitraire, calculer les coordonnées (u, v) de la projection de texture devient un problème difficile. Plusieurs méthodes de projection de texture existent (MK03, ZDW+05). Ces méthodes peuvent être longues et coûteuses et peuvent contenir des étirements de texture qui ne sont pas souhaitables pour obtenir un résultat réaliste. Ces méthodes texturent un objet avec un unique matériau et n'utilisent donc pas l'information de l'environnement pour modifier le type de texture.

Nous proposons une technique pour projeter automatiquement les textures de matériaux sur le maillage du sol en temps réel en calculant à la volée les coordonnées uv. Notre approche est inspirée de (Gei07). Soit $\mathbf{p}$ un point d'un triangle du maillage du sol, et $\mathbf{n}(x, y, z)$ sa normale. La fonction $T(u, v)$ représente la couleur de la texture aux coordonnées (u, v) et $T(\mathbf{p})$ la couleur de la texture au point $\mathbf{p}$. Nous définissons la couleur au point $\mathbf{p}$ en mélangeant les trois couleurs obtenues par la projection de texture selon les trois axes principaux du monde :

$$T(\mathbf{p}) = \alpha\, T(|x|, |y|) + \beta\, T(|x|, |z|) + \gamma\, T(|y|, |z|)$$

Les coefficients de mélange α, β et γ sont calculés en fonctions des normales $\mathbf{n}$ en vue de mélanger les différentes textures (Figure 2.33). $\|\mathbf{n}\|_p = (|x|^p + |y|^p + |z|^p)^{1/p}$ représente la $\mathcal{L}^p$ norme de $\mathbf{n}$, nous définissons les coefficients ainsi :

$$\alpha = |x|^p / \|\mathbf{n}\|_p \qquad \beta = |y|^p / \|\mathbf{n}\|_p \qquad \gamma = |z|^p / \|\mathbf{n}\|_p$$

FIGURE 2.33 – Projection de texture sur un objet. Les plans représentent les différents axes de projection.

L'exposant $p \geq 1$ caractérise le contrôle du mélange entre les axes de projection de la texture. Dans notre implémentation, nous utilisons $p = 8$. Nous avons implémenté cette technique de texture dans un shader HLSL shader pour du rendu temps réel et en shader Mental Ray ® pour produire des images de haute qualité.

Cette méthode permet d'obtenir une projection de texture sur un maillage de géométrie et topologie arbitraire.

Mélange de texture

La texture du sol est différente en fonction du type de matériau qui se trouve dans le voisinage. Nous devons donc analyser pour un point $\mathbf{p}$ l'ensemble du voisinage afin de trouver la texture correspondante. Notre approche consiste à mélanger la texture des différents matériaux selon le ratio des matériaux dans le voisinage.

Pour chaque point $\mathbf{p}$ du maillage, nous évaluons le ratio de matière visible dans un domaine cubique Ω centré en $\mathbf{p}$ (Figure 2.34). Le ratio de matière visible est défini en calculant le rapport de surface visible S_m pour chaque matériau sur la surface totale S_t

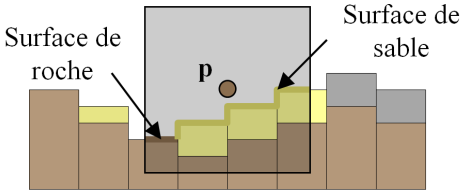


FIGURE 2.34 – Évaluation de la surface visible pour les différents matériaux.

de matière visible. Pour chaque couche intersectant le domaine cubique, nous évaluons l'aire de surface visible. L'air de surface visible est défini pour toute partie de couche qui est en contact avec une couche d'air. La surface totale de matière visible est définie par :

$$S_t = \Sigma S_m$$

Le ratio de couleur pour chaque matériau est défini par l'équation :

$$r_m = \frac{S_m}{S_t}$$

Cette méthode permet d'obtenir la texture correspondante au matériau courant, ou le mélange de plusieurs textures au niveau des interfaces (Figure 2.35).

FIGURE 2.35 – Mélange de 4 textures sur différents objets.

2.5.2 Génération du maillage de l'eau

Nous ne pouvons pas utiliser les surfaces implicites pour définir les volumes d'eau. En effet, la surface implicite a des propriétés de lissage du maillage qui ne sont pas adaptées à la visualisation d'un volume d'eau. L'effet de lissage est illustré dans la figure 2.36.

Pour résoudre ce problème, nous représentons la surface d'eau par une carte d'élévation locale pour définir chaque volume d'eau $\mathcal{Q}$ par l'ensemble des couches connexes. Le processus d'extraction de la carte d'élévation est le suivant :

1. Pour chaque couche d'eau l_i de l'ensemble $\mathcal{Q}$, nous modifions la hauteur de la carte d'élévation de données.

FIGURE 2.36 – Triangulation du volume d'eau par surface implicite (gauche) et par extraction et extension du plan d'eau (droite).

2. Nous calculons la hauteur moyenne h du volume d'eau pour pouvoir définir une surface planaire sur l'ensemble du volume d'eau.
3. L'ensemble des hauteurs stockées dans la carte d'élévations est modifié pour obtenir sur chaque position la hauteur h.
4. Nous étendons la zone d'eau au niveau des interfaces afin de coller le maillage d'eau au maillage du sol. Cette dilatation est effectuée tant que la fonction d'appartenance n'est pas dans le matériau roche.

Ce processus d'extraction de maillage d'eau est effectué pour chaque volume d'eau $\mathcal{Q}$. Nous associons à chaque carte d'élévation un masque afin de générer uniquement le maillage de la zone d'eau. Ce masque supprime tous les points contenant une hauteur différente de h.

2.6 Conclusion et discussion

Nous avons implémenté notre modèle de terrain hybride et les outils de modélisation dans une application codée en C++. Nous avons appliqué notre méthode pour créer différentes variétés de scènes rocheuses complexes avec des caractéristiques géologiques complexes incluant les grottes (Figure 2.39), les falaises avec des surplombs (Figure 2.37) et des arches (Figure 2.38).

2.6.1 Contrôle

Notre structure de données permet de créer efficacement des terrains complexes de géométrie et de topologie quelconque. Les outils proposés à l'utilisateur permettent de contrôler facilement et intuitivement la forme finale du terrain. Le moteur de stabilisation simplifie les interactions de l'utilisateur avec le terrain pour obtenir un résultat réaliste et cohérent à chaque édition du terrain. Le modèle d'instanciation de pierres (Chapitre 3) permet à l'utilisateur de contrôler le placement d'un nombre important

d'instances de pierres. La manipulation de l'ensemble des instances de pierres est effectuée de façon transparente pour simplifier l'interaction avec l'utilisateur. L'utilisateur contrôle alors uniquement le volume de l'empilement lors de l'édition des couches de pierres.

2.6.2 Réalisme

La possibilité de créer une structure complexe composée de nombreux matériaux permet de créer des scènes très réalistes. Notre modèle permet d'augmenter le réalisme des cartes d'élévation de données qui restent très limitées. Notre modèle améliore le réalisme des cartes d'élévation en permettant de rajouter différents matériaux à la scène et d'améliorer la visualisation des terrains contenant des pentes de plus de 30° grâce à la création du maillage par Marching Cubes. Par ce procédé, nous obtenons une scène plus complexe et par la même occasion plus réaliste. Par exemple, nous pouvons charger dans notre modèle une carte d'élévation représentant une Mesa. Une Mesa est une élévation de terre ou de roche dont le dessus est plat et les côtés constitués de falaises pouvant ressembler à une table. Les falaises produites par la carte d'élévation ne sont pas du tout réalistes. Notre modèle permet d'éroder la forme obtenue par la carte d'élévation pour obtenir automatiquement des détails pour pouvoir la visualiser.

FIGURE 2.37 – Un canyon contenant un lit de rivière asséchée.

2.6.3 Perspectives

De nombreuses perspectives découlent de ces travaux qui proposent une nouvelle représentation de terrain entièrement tridimensionnelle.

Une des perspectives nécessitant d'être étudiée est la simulation d'un écosystème tridimensionnel complète afin de modéliser des scènes évoluant au cours du temps (saisons et années). Simuler l'évolution complète d'un écosystème consiste à évaluer chaque individu de chaque espèce en prenant en compte les interactions entre le voisinage et la scène. Ce processus est d'autant plus complexe qu'il nécessite de simuler la vie d'un

FIGURE 2.38 – Des arches créées en sculptant tout d'abord une falaise par des courbes de profils puis en extrudant et sculptant interactivement les arches. L'apparence finale a été obtenue en érodant légèrement la surface de la paroi.

FIGURE 2.39 – Des grottes reliées par un réseau de tunnels et de ponts de pierres sculptées par balayage et par l'outil d'érosion. Les piles de pierres dans le tunnel et au fond du grand gouffre ont été ajoutées en érodant des parties du mur.

individu consistant à modéliser sa croissance au cours du temps, la génération de déchets tels que les feuilles ou les branches en fonction de la saison, jusqu'à sa mort et sa détérioration.

Pour se développer, chaque plante utilise des ressources de terre, d'eau et de lumière. Notre modèle permettrait de représenter précisément l'ensemble de ces ressources en fonction de l'accessibilité de la lumière dans des scènes aux reliefs complexes, en fonction de la quantité de terre disponible dans le voisinage de l'individu et en ajoutant un paramètre d'humidité à la terre. Ce paramètre d'humidité peut varier en fonction de l'absorption de l'eau issue de précipitations préalablement définies par une trame météorologique. Cette approche permettrait d'effectuer des simulations plus précises que celles existant actuellement.

Un autre aspect intéressant à étudier est la modélisation de l'ensemble des instances au cours des saisons pour obtenir une grande variété de scène. Cette aspect consiste à gérer l'ensemble des déchets (feuilles et branches) issu de chaque individu. Ce phénomène pourrait être intégré dans notre modèle en définissant des couches de feuilles et de branches obtenues lors de l'automne. La manipulation par couche permettrais

de simplifier l'interaction avec les milliers d'instances nécessaire pour les intancier uniquement au moment de la visualisation. Au cours du temps les couches de déchets se décomposent en couches de terre pour apporter de nouvelles ressources aux individus et modéliser une nouvelle croissance lors du printemps.

D'autres travaux nécessiteraient d'être effectués sur ce modèle afin d'utiliser cette nouvelle structure comme la génération automatique de scènes complexes ou encore la modélisation de rivières complexes contenant des cascades.

Chapitre 3

Génération d'empilements de pierres



Sommaire

Les résultats présentés dans ce chapitre ont fait l'objet d'une publication dans la revue internationale Computer Graphics Forum (PGMG09b).

Notre approche de génération d'empilements de pierres est complémentaire à la méthode détaillée dans le Chapitre 2. Rappelons que les matériaux granuleux tels que le sable, la terre et les pierres sont traités comme un milieu continu dans notre structure. Le sable et la terre sont visualisés grâce à un mélange de textures sur la surface du sol. Par contre, les détails contenus dans les couches de pierres doivent être modélisés par de la géométrie pour obtenir une scène réaliste.

Dans ce chapitre, nous proposons une méthode procédurale permettant de remplacer un volume par un ensemble d'instances de pierres en contact les unes avec les autres(Figure 3.1). Comme nous utilisons une contrainte de placement des pierres dans l'espace, nous ne pouvons pas utiliser de simulation physique. En effet, aucun contrôle ne peut être obtenu sur la forme de l'empilement en effectuant une stabilisation physique sur les pierres. De plus la simulation de la stabilisation d'un grand nombre de pierres nécessite des calculs longs et coûteux.

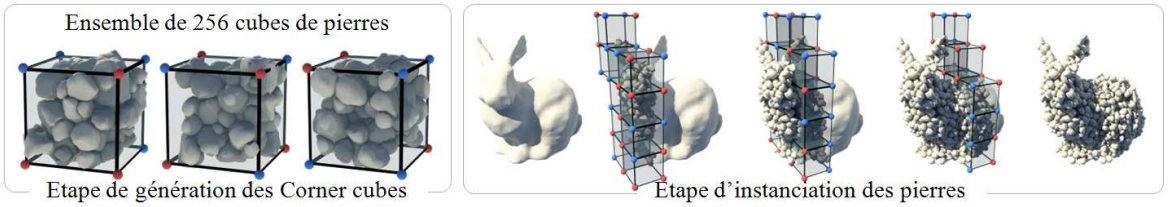


FIGURE 3.1 – Processus de création d'un empilement de pierre.

Notre processus prend en compte les contraintes de contrôle de forme de chacune des pierres, les contraintes de placements de l'ensemble des pierres ainsi que la contrainte de manipulation d'un nombre très élevé de pierres. Notre méthode procède en deux étapes.

La première étape consiste à générer un ensemble de cubes apériodiques contenant des pierres en contact les unes avec les autres de façon à garantir le raccordement dans l'empilement des différents cubes. Les formes des pierres sont contrôlées dans notre processus de génération en utilisant une fonction de distance anisotrope pour obtenir une plus grande variété de pierres. Les pierres sont générées de façon à ce qu'elles soient en contact avec leurs voisines pour obtenir un résultat plausible de stabilité. Nous rappelons que nous voulons obtenir un résultat visuellement plausible et non physiquement réaliste pour accélérer les temps de calculs.

La deuxième étape consiste à paver l'ensemble de l'espace avec les cubes obtenus pour générer les empilements. Seules les pierres contenues dans le volume de matériau sont conservées.

3.1 Vue d'ensemble et notations

Dans cette partie, nous présentons un rapide état de l'art des techniques de pavage, des méthodes de distributions d'objets dans l'espace à l'aide de pavages apériodiques

ainsi que leurs limitations. Puis nous proposons une méthode procédurale pour générer un ensemble de cubes avec une distribution apériodique contenant des pierres en contact. Nous proposons d'utiliser ces cubes pour générer nos empilements de pierres dans nos scènes. Les cubes permettent de stocker l'ensemble de la géométrie des pierres afin de les distribuer interactivement dans une scène sans avoir à les générer à chaque édition. La non périodicité est nécessaire afin d'éviter les artefacts de répétition obtenus avec des motifs périodiques (Figure 3.2).

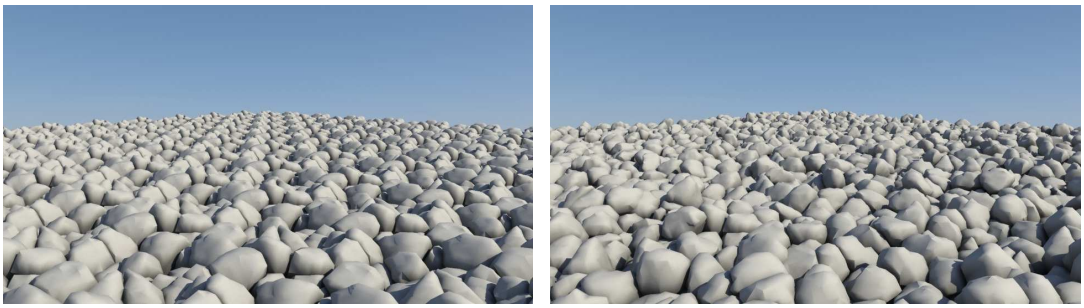


FIGURE 3.2 – A gauche, pavage périodique montrant des artefacts de répétition d'instances et à droite le modèle apériodique.

3.1.1 État de l'art des techniques de pavage

Un problème commun en informatique graphique est la synthèse et le stockage de signaux complexes. Ces signaux peuvent être bidimensionnels ou tridimensionnels pour définir des textures, une distribution de points ou d'objets. Les méthodes de pavages permettent de générer de telles distributions par composition ou répétition d'un ensemble de signaux simples. La plupart des recherches consiste à supprimer les artefacts visibles de répétition de signaux qui apparaissent lors du pavage d'un même signal.

3.1.1.1 Pavage général

De nombreux travaux ont été réalisés pour définir des signaux complexes sur un plan. Ces travaux consistent à générer une texture 2D par composition de formes géométriques ou par combinaison de textures ou de cartes d'élévation de données. Ces méthodes permettent également de distribuer un ensemble d'objets sur une surface, par exemple lors de placement de plusieurs milliers d'arbres dans une scène. Plusieurs méthodes de pavage ont été proposées pour générer un signal complexe à partir de pavés de formes arbitraires (GS86, Gla98). Cette approche consiste à trouver une solution de combinaison des pavés pour obtenir un pavage de l'ensemble de l'espace. En informatique graphique, les pavés triangulaires (NC99) et carrés (Cul96) sont préférés afin de faciliter l'assemblage des pavés. Cette approche consiste à obtenir une continuité de texture ou de distribution entre chaque pavé.

Les pavés de Wang sont les plus utilisés pour la synthèse de textures (Cul96, CSHD03, KCODL06, Lag09). La méthode consiste à définir un ensemble de pavés carrés, en associant une couleur à chaque arête. L'assemblage s'effectue simplement en raccordant deux pavés ayant la même couleur d'arête. La génération des pavés s'effectue en définissant une contrainte de continuité entre les arêtes de même couleur. Le problème de cette méthode est que la continuité est uniquement valable entre les arêtes des carrés. De ce fait, la continuité n'est pas assurée au niveau des voisins se situant sur la diagonale. Pour assurer cette continuité, Lagae a proposé les Corner Tile (LD06b). Cette méthode consiste à diviser le carré en sous parties pour prendre en compte l'ensemble du voisinage. Les carrés sont définis en associant une couleur à chaque sommet du carré. Deux carrés sont raccordés lorsque la couleur des deux sommets concordent.

3.1.1.2 Pavage à partir de cellules cubiques

Deux modèles de distribution apériodique de cubes ont été proposés : les Wang cubes et les Corner cubes. Ces modèles sont des extensions des modèles 2D qui traitent les textures.

Les Wang Cubes (CK95, LEQ+07) sont une extension des Wang Tiles (CSHD03, Lag09) traitant le cas 2D pour générer des textures. Les cubes ont une orientation fixe, et sont définis en associant une couleur sur les faces des cubes (Figure 3.3). Un cube est joint à un autre cube seulement si la couleur des faces connexes est la même. Les Wang Cubes ont été les premiers à être utilisés pour distribuer de la géométrie dans l'espace.

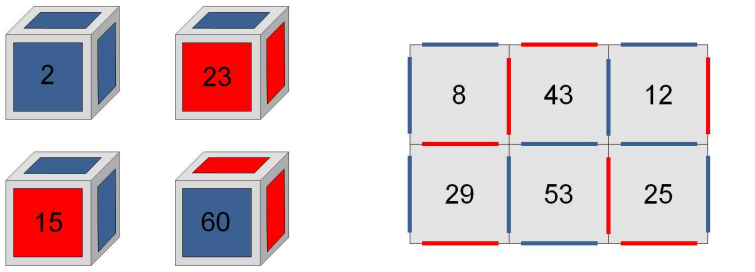


FIGURE 3.3 – Quelques Wang Cubes différents à partir d'un ensemble complet de 64 configurations, et une distribution de Wang cubes dans l'espace.

En affectant n couleurs différentes par face, on obtient par combinaison de couleurs n^6 cubes. En définissant 2 couleurs par face, on a 2^6 cubes soit 64 cubes. En utilisant 3 couleurs par face, on obtient 3^6 cubes soit 729 cubes. Le nombre de cubes augmente exponentiellement avec le nombre de couleurs possibles par face. Une méthode (CK95) a été proposée pour obtenir un faible nombre de cubes tout en gardant une combinaison assez grande pour avoir une distribution apériodique. Cette méthode réduit le nombre de cubes à 21. Cette méthode utilise seulement 2 couleurs par face.

Le problème des Wang cubes provient de la connexité avec les cubes voisins : les cubes sont connectés entre eux seulement par leurs faces. De ce fait, des artefacts apparaissent lors de la génération de géométrie à l'intérieur des cubes. La géométrie aura une continuité seulement au niveau des faces, par contre aucune continuité ne pourra être obtenue au niveau des arêtes et des sommets. Par exemple, pour générer des sphères dans l'espace avec une distribution de Poisson, la propriété de distribution pourra être maintenue seulement sur les faces des cubes. Ce problème a été résolu par les Corner Cubes.

Les Corner Cubes ont été proposés dans (LD06a) pour distribuer des sphères dans l'espace en vérifiant une distribution de Poisson. La méthode a été utilisée pour distribuer des objets dans l'espace et pour générer des textures volumiques. Les Corner cubes sont une extension du cas 2D des Corner Tiles (LD06b) proposant de résoudre le problème de connexité pour les textures. Les cubes ont également une orientation fixe. Contrairement aux Wang Cubes qui sont définis par les faces du cubes, les Corner Cubes sont définis par les 8 sommets du cube (Figure 3.4). Un cube est défini en affectant une couleur à chaque sommet. Un cube est joint à un autre cube lorsque les couleurs des 4 sommets d'une face concordent avec les couleurs des sommets de la face de l'autre cube.

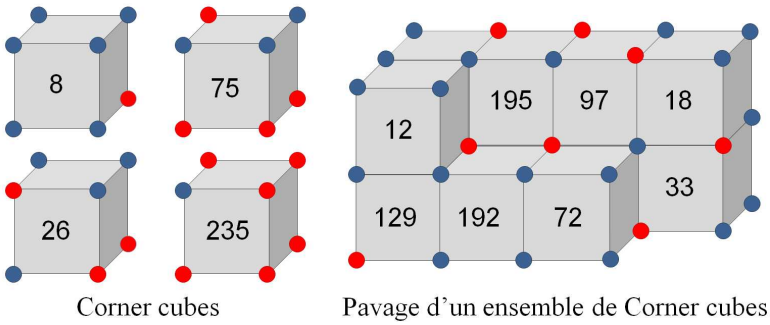


FIGURE 3.4 – Quelques Corner Cubes différents à partir d'un ensemble complet de 256 configurations, et un exemple de pavage généré à partir d'un sous ensemble de Corner Cubes.

Afin de prendre en compte l'ensemble du voisinage, le cube est divisé en parties pour définir proprement les interfaces entre les cubes. Des régions sommets, arêtes, faces et centres sont alors définies (Figure 3.5). Pour n couleurs différentes par sommets, n^8 cubes doivent être définis soit 256 cubes pour 2 couleurs et 6 561 cubes pour 3 couleurs. Le nombre de cubes à définir est plus grand que pour les Wang Cubes.

3.1.2 Processus de génération

Dans cette section nous présentons le problème de génération d'objets au niveau des interfaces des cubes, et nous proposons une représentation pour résoudre ce problème.

Les Corner Cubes (LD06a), qui ont été proposés pour générer un ensemble de

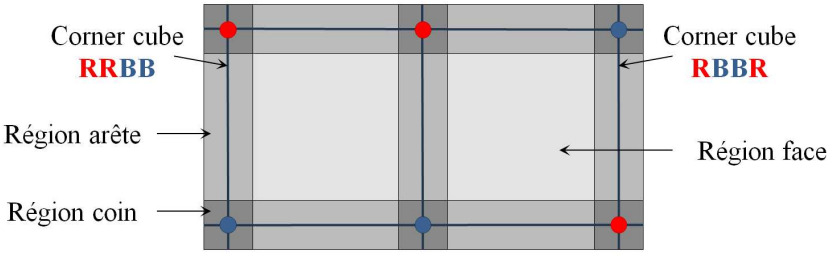


FIGURE 3.5 – Pavage de deux Corner Cubes avec leurs régions d'interface.

sphères dans l'espace avec une distribution de Poisson, ne sont pas adaptés pour générer un ensemble de maillage ayant des faces en contact qui est nécessaire pour les pierres. Le problème du Corner Cube est qu'il n'est pas possible de créer des volumes de pierres $\mathcal{V}_k$ directement à partir des cubes de façon cohérente. En effet, la technique de génération, même locale, nécessite de connaître le proche voisinage de la pierre à construire. Pour illustrer le problème, nous proposons de définir les volumes $\mathcal{V}_k$ en divisant l'espace par l'algorithme de Voronoï sur un ensemble de points germes $\mathbf{p}_k$. Utiliser une propriété de distance entre les éléments n'est pas suffisant car des artefacts de génération vont se produire au niveau des interfaces. Pour un Corner Cube $\mathcal{C}$, la géométrie du volume $\mathcal{V}_k$ dépend des points germes voisins $\mathbf{p}_k$ qui peuvent être différents selon les combinaisons de Corner Cubes.

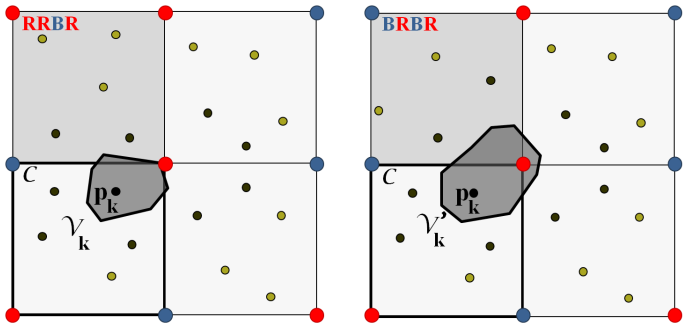


FIGURE 3.6 – Deux volumes différents de pierres $\mathcal{V}_k$ et $\mathcal{V}'_k$ peuvent être générés pour un même point germe $\mathbf{p}_k$ avec des Corner Cubes compatibles mais différents.

La figure 3.6 illustre le problème de génération des volumes en deux dimensions : deux volumes différents de pierres $\mathcal{V}_k$ et $\mathcal{V}'_k$ sont générés pour un même point germe $\mathbf{p}_k$ dépendant des Corner Cubes voisins dont les configurations, notés BRBR et RRBR respectivement, sont différentes bien que compatibles avec les couleurs des sommets du Corner Cube $\mathcal{C}$.

Nous proposons une nouvelle structure, les Corner Cubes Refined, comme un moyen pour résoudre le problème de création des volumes de pierres. En décomposant le Corner Cube en une grille de n^3 cellules $\mathcal{C}_{ijk}$, nous pouvons garder une trace des points germes voisins dans le schéma de génération des Corner Cubes. Nous proposons un processus de construction qui évite les ambigüités dans la création des volumes des pierres.

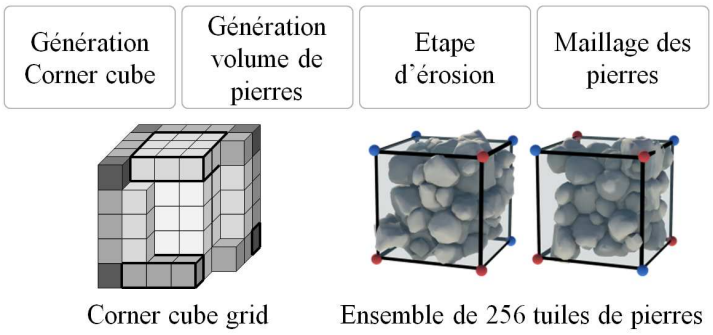


FIGURE 3.7 – Procédure de génération des pierres.

Notre méthode permet de créer un ensemble apériodique de cubes de pierres, avec des pierres en contacts les unes avec les autres tout en se chevauchant, même au niveau des interfaces des cubes pour obtenir une continuité dans les empilements de pierres. Notre méthode se déroule en quatre étapes (Figure 3.7) :

1. Nous générons un ensemble de 256 Corner Cubes. Chaque Corner Cube est décomposé en une grille de cellules, noté $\mathcal{C}_{ijk}$. Un ensemble de points germes $\mathbf{p}_k$ est généré pour chaque cellule.
2. Nous créons le volume $\mathcal{V}_k$ pour chaque point germe $\mathbf{p}_k$. Le volume est obtenu en calculant la cellule de Voronoï à partir de l'ensemble des points germes voisins en utilisant une fonction de distance anisotrope pour contrôler la forme des pierres.
3. Nous générons une représentation implicite des pierres en érodant les cellules de Voronoï $\mathcal{V}_k$ avec une érosion sphéroïdale (BFO+07).
4. Finalement, le maillage est obtenu par une étape d'extraction des triangles de la représentation implicite de chacune des pierres.

La structure proposée permet de définir le volume $\mathcal{V}_k$ des pierres localement pour chaque cellule $\mathcal{C}_{ijk}$. Le volume $\mathcal{V}_k$ généré peut intersecter une cellule voisine à $\mathcal{C}_{ijk}$. Cette propriété permet d'obtenir un chevauchement de l'ensemble des pierres entre elles pour conserver une continuité d'empilement entre les Corner Cubes.

3.2 Génération des Corner Cubes

Dans cette section, nous présentons le processus de génération des Corner Cubes. Dans le reste de ce chapitre, nous utiliserons 2 couleurs pour définir les sommets, nous devons ainsi générer un ensemble de 256 cubes.

3.2.1 Définition du Corner Cube Refined

Soit $\mathcal{C}$ un Corner Cube générique. Nous décomposons $\mathcal{C}$ en une grille de n^3 cellules notées $\mathcal{C}_{ijk}$. Les différentes cellules $\mathcal{C}_{ijk}$ sont définies comme des cellules sommets, arêtes,

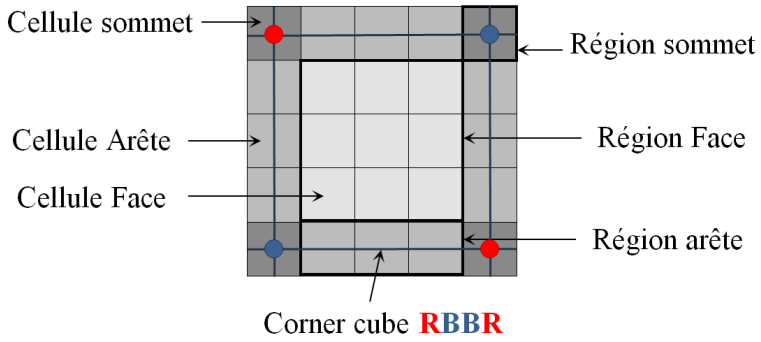


FIGURE 3.8 – La structure du Corner Cube Refined ($n = 5$).

faces et centre (Figure 3.8). Nous décomposons le Corner Cube en un ensemble de régions nommées régions sommets, arêtes, faces et centre. Elles sont respectivement définies comme les 8 cellules sommets, les $n-2$ cellules le long des 12 arêtes du Corner Cube, les $(n-2)^2$ cellules des 6 faces du Corner Cube et les $(n-2)^3$ cellules restantes au centre du cube.

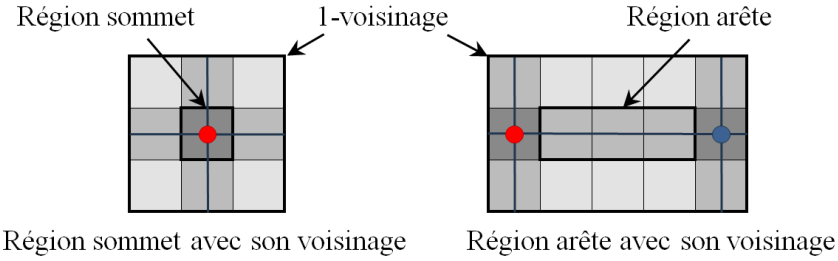


FIGURE 3.9 – Représentation 2D du voisinage des régions avec une région sommet et une région arête.

Nous définissons les cellules voisines d'une région donnée comme les cellules situées dans le 1-voisinage des cellules de la région (Figure 3.9). Nous pouvons noter que le 1-voisinage d'une région contient des cellules de types différents de la région étudiée qui peuvent appartenir à plusieurs Corner Cubes. Cette propriété clef dans l'algorithme de génération des points germes joue un rôle important dans la construction des 256 Corner Cubes : les cellules voisines sont utilisées pour conserver l'historique des points $\mathbf{p}_k$ déjà créés pour éviter des configurations ambiguës dans la définition des volumes $\mathcal{V}_k$.

3.2.2 Génération des points germes

Lors de la génération des 256 Corner Cubes, nous effectuons une distribution aléatoire des points germe $\mathbf{p}_k$ dans les différentes cellules $\mathcal{C}_{ijk}$ du voisinage des régions. Un ou plusieurs points germes peuvent être créés dans une cellule. Le nombre maximal de points germes par cellule est représenté par le paramètre s et le nombre de points

germes dans la cellule est représenté par n_k. Les points germes $\mathbf{p}_k$ sont créés aléatoirement dans chaque cellule. Le paramètre n_k définit la densité relative des points germes dans une cellule, et ainsi la taille relative des volumes de Voronoï $\mathcal{V}_k$ (Section 3.3).

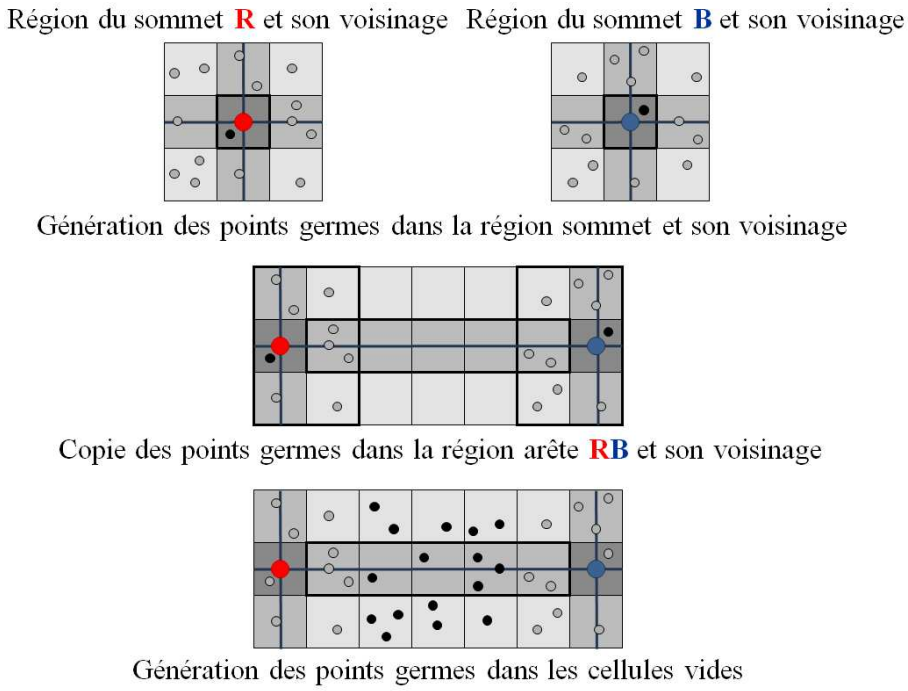


FIGURE 3.10 – Principe des deux premières étapes de l'algorithme de génération des points germes pour une arête RB.

Par combinaison, pour créer les 256 configurations des Corner Cubes, nous avons besoin de générer 2 régions sommets, 12 régions arêtes, 48 régions faces et 256 régions centre. Les points germes $\mathbf{p}_k$ sont séquentiellement créés dans les régions sommets, arêtes, faces et centre. L'algorithme peut être décomposé en quatre étapes :

1. Générer les points germes $\mathbf{p}_k$ dans les 2 régions sommets et dans les cellules voisines.
2. Pour l'ensemble des 12 régions arêtes, copier les points germes $\mathbf{p}_k$ des cellules sommets et des cellules du voisinages du sommets sur les cellules de la région arête correspondante. Générer de nouveaux points germes seulement dans les cellules vides de la région arête et dans les cellules du voisinage de l'arête.
3. Pour l'ensemble des 48 régions faces, copier les points germes $\mathbf{p}_k$ des cellules voisines des sommets et arêtes dans les cellules correspondantes à la région face. Générer de nouveaux points germes seulement dans les cellules vides de la région face et des cellules dans le voisinage de faces.
4. Finalement, pour l'ensemble des 256 régions centre, copier les points germes $\mathbf{p}_k$ des régions voisines des sommets, arêtes et faces avant de générer de nouveau points germes dans les cellules de la région centre.

La figure 3.10 illustre le processus pour une arête RB. A la fin du processus de construction, nous créons un ensemble de 256 Corner Cubes en assemblant les régions correspondantes de sommets, arêtes, faces et centre. Les Corner cubes contiennent une distribution consistante des points germes qui garantissent que les volumes des pierres $\mathcal{V}_k$ se raccordent parfaitement, même sur les bords du cube.

3.3 Génération du volume des pierres

Dans cette section, nous détaillons le calcul de génération des volumes des pierres $\mathcal{V}_k$ à partir des points germes $\mathbf{p}_k$.

3.3.1 Définition du volume des pierres

Nous définissons les volumes $\mathcal{V}_k$ comme les cellules de Voronoï des points $\mathbf{p}_k$. Au lieu d'utiliser une distance euclidienne qui crée des formes de pierres globalement ronde, nous proposons d'utiliser une distance anisotrope. Plusieurs fonctions de distances anisotropes ont été proposées dans les travaux de Crespin (CBS96) pour modéliser des primitives implicites. Nous proposons d'utiliser des fonctions de distances anisotropes ellipsoïdales pour contrôler la forme globale et l'orientation des pierres. L'utilisation de cette fonction nous permet de générer une plus grande variété de pierres.

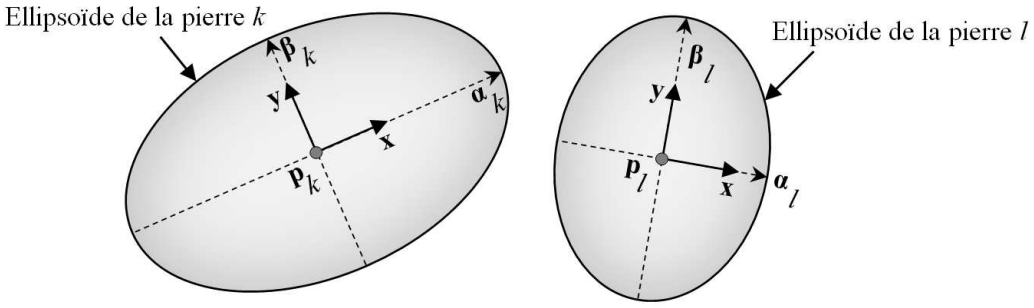


FIGURE 3.11 – Représentation des fonctions ellipsoïdales selon leurs repères ($\mathbf{x}$, $\mathbf{y}$) et leurs paramètres α et β.

Pour chaque point germe $\mathbf{p}_k$, nous définissons la fonction de distance anisotrope ellipsoïdale (Figure 3.11), noté $d_k(\mathbf{p})$, qui va nous servir à créer le volume des pierres par :

$$d_k(\mathbf{p}) = \frac{1}{g_k} f_k(\mathbf{p})$$

Soient $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ les vecteurs du repère local attachés sur chaque point germe $\mathbf{p}_k$, et α, β, γ les paramètres de l'ellipsoïde caractérisant la taille relative selon les axes du repère. La fonction $f_k(\mathbf{p})$ est définie par :

$$f_k(\mathbf{p}) = \alpha_k((\mathbf{p} - \mathbf{p}_k)\mathbf{x})^2 + \beta_k((\mathbf{p} - \mathbf{p}_k)\mathbf{y})^2 + \gamma_k((\mathbf{p} - \mathbf{p}_k)\mathbf{z})^2$$

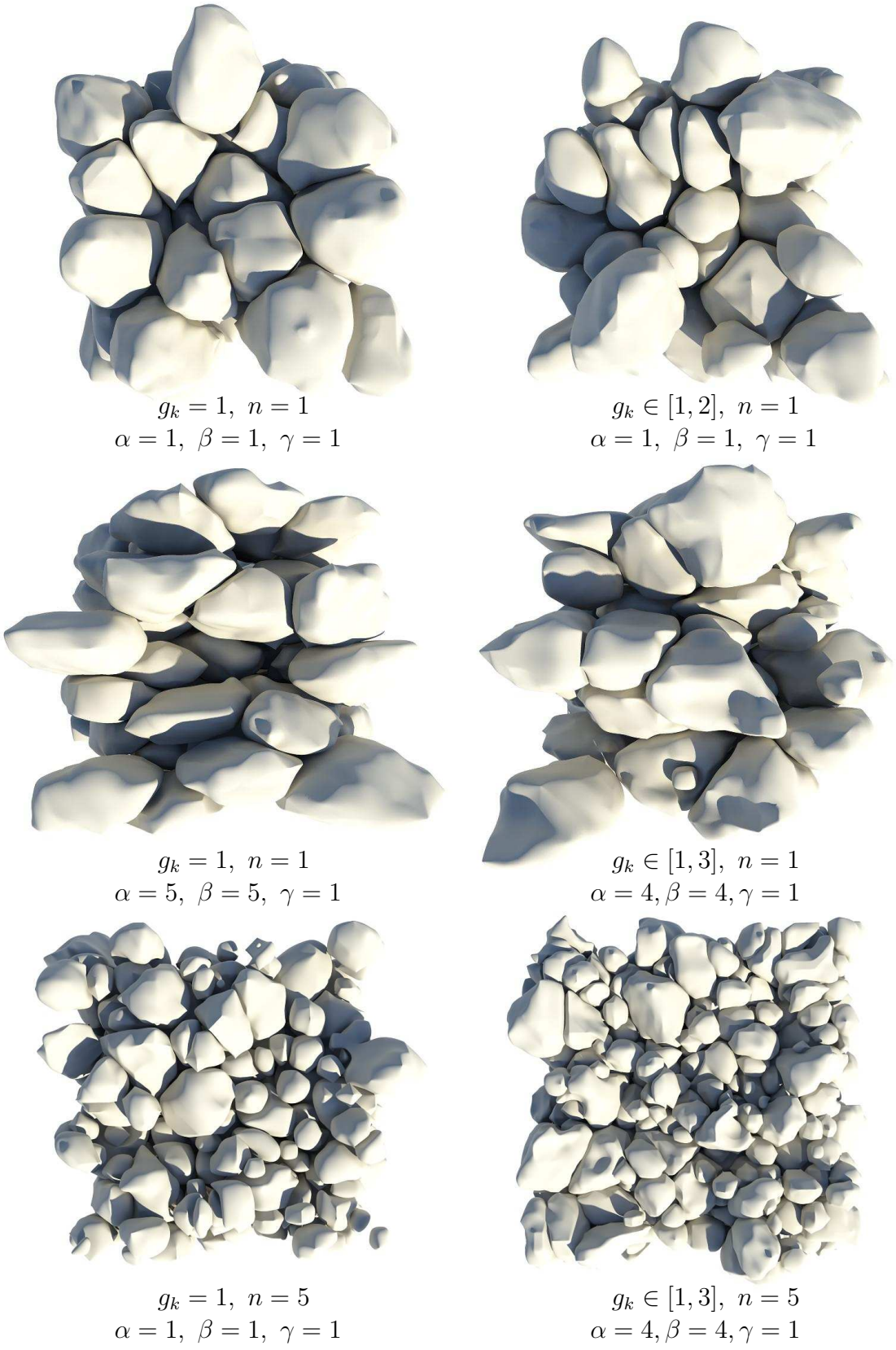


FIGURE 3.12 – Pierres de formes variées obtenues en utilisant différents paramètres.

Le paramètre g_k et le nombre de points par cellule n_k sont utilisés pour contrôler la taille relative des pierres entre elles : plus n_k est grand, et plus les volumes de Voronoï $\mathcal{V}_k$ seront petits. Le paramètre $g_k > 0$ règle la taille relative des pierres. Dans notre système, la valeur de g_k est définie en utilisant une distribution uniforme aléatoire dans un intervalle de contrôle $[1, g_{\max}]$. Dans notre implémentation, nous définissons $g_{\max} = 4$ pour limiter l'écart entre la taille relative des pierres.

Si n_k à la même valeur pour chaque cellule, alors les volumes générés $\mathcal{V}_k$ des pierres et les modèles finaux des pierres auront approximativement la même taille. Par contre, si n_k est obtenu en échantillonnant aléatoirement l'ensemble des valeurs dans $n_k \in [1, s]$, nous pouvons créer des volumes des pierres avec des tailles très différentes, comme un mélange de pierres et de graviers.

Le repère local du système $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ est généré par une distribution aléatoire uniforme. Les coefficients α_k, β_k et γ_k sont obtenus en utilisant également une distribution aléatoire uniforme dans un certain intervalle d'entrée. Cet intervalle est caractérisé par une limite minimum et maximum de taille sur chaque axe du repère local α_k, β_k et $\gamma_k \in [l_{min}, l_{max}]$.

La figure 3.12 montre plusieurs modèles de pierres générées selon les différents paramètres de l'ellipsoïde g_k, α_k, β_k et γ_k et du nombre de pierre n_k.

3.3.2 Génération des cellules de Voronoï

La création des cellules de Voronoï en utilisant des distances anisotropes est un problème très complexe. Nous effectuons l'étape de génération du volume de pierre en calculant un diagramme de Voronoï discret. Nous subdivisons le Corner Cubes $\mathcal{C}$ de n^3 cellules en une grille $\mathcal{G}$ de $n_{\mathcal{G}}^3$ voxels avec $n^3 < n_{\mathcal{G}}^3$. Le paramètre $n_{\mathcal{G}}$ définit la résolution de la grille qui est utilisée pour calculer les cellules de Voronoï et pour effectuer le processus d'érosion qui sera présenté dans la section suivante (Section 3.4). Par conséquent, $n_{\mathcal{G}}$ doit être assez grand pour garantir que la grille capture tous les volumes de Voronoï . L'opération est effectuée en calculant la distance minimum entre tous les points germes $\mathbf{p}_k$.

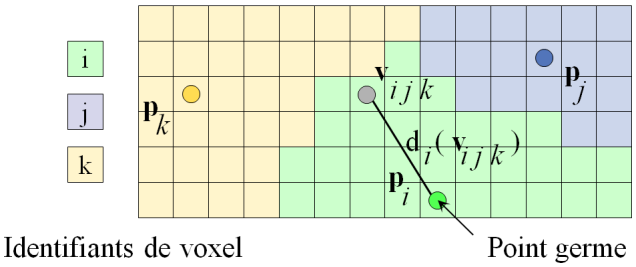


FIGURE 3.13 – Définition des volumes de Voronoï pour chaque pierre en affectant un identifiant à chaque voxel $\mathcal{G}_{ijk}$.

$\mathcal{G}_{ijk}$ représente un voxel de la grille et $\mathbf{v}_{ijk}$ représente le centre du voxel. Pour

chaque voxel, nous calculons d'abord un unique identifiant qui stocke l'index k de son point germe correspondant $\mathbf{p}_k$ (Figure 3.13). Cette étape est effectuée efficacement en évaluant la distance minimum entre le centre du voxel et les points germes avoisinant $d_k(\mathbf{v}_{ijk})$ avec $d_k(\mathbf{v}_{ijk})$ la distance anisotrope à la pierre k. La représentation discrète nous permet d'effectuer l'érosion des volumes de pierres pour générer la forme finale des pierres en contact.

3.4 Processus d'érosion

Le processus de génération du volume des pierres nous fournit une représentation à base de voxel du volume des pierres $\mathcal{V}_k$. L'étape d'érosion vise à creuser et sculpter la forme des pierres hors de leurs volumes grossiers mais en maintenant les points de contacts entre les formes des pierres finales avec pour but de générer des piles de pierres convaincantes. L'algorithme d'érosion peut être décomposé en quatre étapes :

1. Tout d'abord nous générons un ensemble de points de contact entre les volumes de pierres.
2. Ensuite, nous définissons un champ d'atténuation qui va affaiblir l'érosion dans le voisinage des points de contact pour conserver des zones de contact entre les pierres (Figure 3.15).
3. Plusieurs itérations d'érosion sont effectuées en utilisant une érosion sphéroïdale. Le nombre d'itérations peut être contrôlé par l'utilisateur (BFO+07).
4. Finalement, nous générons une représentation implicite des pierres pour créer le maillage des pierres en utilisant des techniques de génération de maillage de surface implicite (LC87).

3.4.1 Points de contact

Les points de contact entre les différentes pierres sont générés en deux étapes. Nous générons un point de contact, noté $\mathbf{c}_{ij}$, pour chaque volumes de pierres $\mathcal{V}_i$ et $\mathcal{V}_j$ en contact. Le point de contact est créé sur le segment de droite $\mathbf{p}_i\,\mathbf{p}_j$ en trouvant la position qui satisfait $d_i(\mathbf{c}_{ij}) = d_j(\mathbf{c}_{ij})$ en utilisant l'algorithme par dichotomie avec $d_k(\mathbf{c}_{ij})$ la distance anisotrope associée à la pierre k.

Il est également possible de supprimer quelques points de contact pour créer plus d'espace entre les pierres. Par contre cette opération ne garantit plus la stabilité des pierres entre elles. Dans notre système, nous gardons un point de contact entre chaque couples pour obtenir une configuration visuellement plausible.

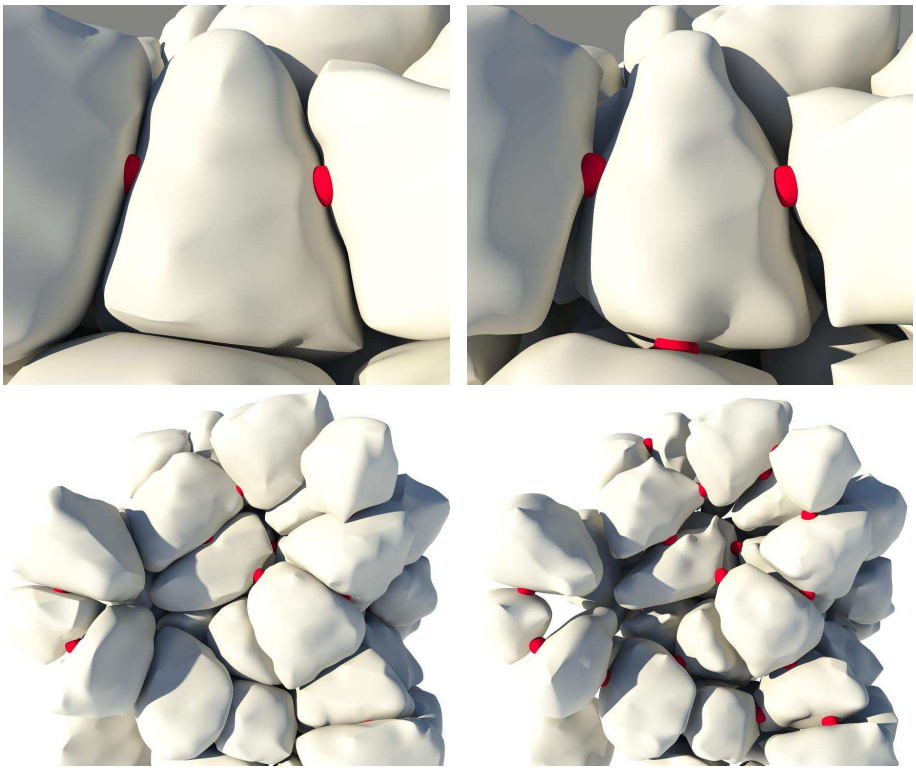


FIGURE 3.14 – Exemple de pierres en contact après le processus d'érosion : les points de contact ont été représentés par des sphères rouges.

3.4.2 Génération du champ de potentiel d'érosion

Les pierres sont obtenues en érodant les volumes des pierres $\mathcal{V}_k$ tout en préservant les points de contact entre les volumes. L'érosion doit être réduite dans le voisinage des points de contact (Figure 3.15). Pour chaque point de contact $\mathbf{c}_{ij}$, nous définissons le support compact de la fonction d'atténuation $\omega_{ij}(\mathbf{p})$ comme une fonction linéaire par morceaux de la distance aux points de contact, $\omega_{ij}(\mathbf{p}) = \omega_{ij}(r)$ avec $r = \|\mathbf{p} - \mathbf{c}_{ij}\|$:

$$\omega_{ij}(r) = \begin{cases} 0 & \text{si } r < a \\ (r-a)/(b-a) & \text{si } a \leq r < b \\ 1 & \text{sinon} \end{cases}$$

Le paramètre a définit le rayon de contact de la région sphérique où aucune érosion n'est effectuée. Ce paramètre nous permet de contrôler la taille de la surface de contact entre les pierres. La paramètre b contrôle la taille de la région de mélange entre la zone d'érosion complète et la zone sans érosion.

Le champ d'atténuation final $\omega(\mathbf{p})$ est défini comme la combinaison de l'ensemble des champs d'atténuation $\omega_{ij}(\mathbf{p})$ ainsi :

$$\omega(\mathbf{p}) = \min_{(i,j) \in [0, n_{\mathcal{G}} - 1]^2} (\omega_{ij}(\mathbf{p}))$$

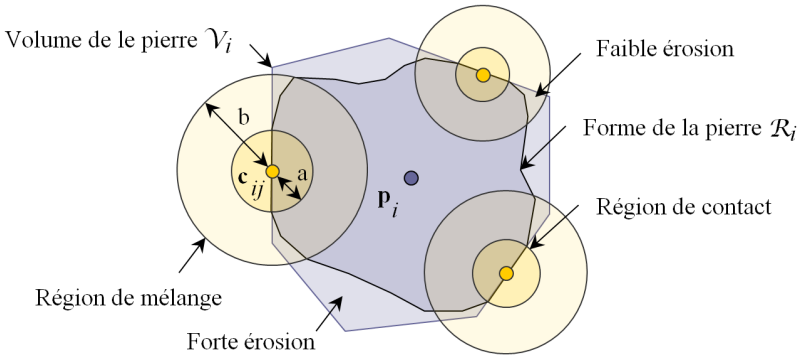


FIGURE 3.15 – Définition de la région d'atténuation de l'érosion autour des points de contact.

En utilisant la fonction minimum nous garantissons qu'aucune érosion n'est appliquée à l'intérieur de la région sphérique $r < a$ ce qui est une condition suffisante pour préserver les contacts entre les pierres.

3.4.3 Processus d'érosion

Le processus d'érosion s'effectue en deux étapes. Tout d'abord, nous initialisons les voxels $\mathcal{G}_{ijk}$ de la grille de voxel avec le paramètre $\rho_{ijk} = 1$ qui correspond à l'état d'érosion du voxel. Si $\rho_{ijk} = 0$, alors le voxel est complètement érodé et devient vide (Figure 3.16).

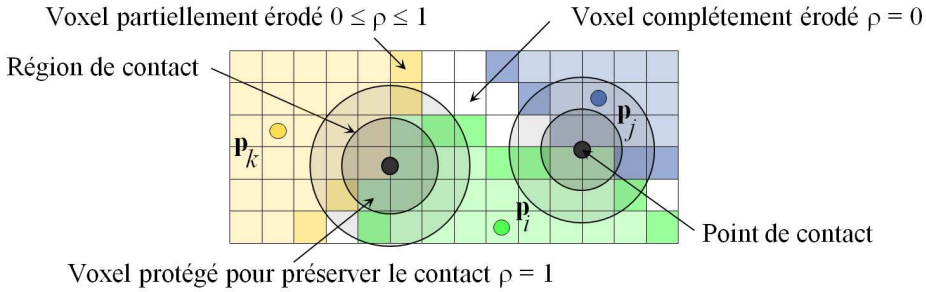


FIGURE 3.16 – Érosion des volumes de Voronoï discret

Le processus d'érosion est effectué en calculant itérativement la quantité d'érosion pour chaque cellule. Seuls les voxels à la surface du volume des pierres, *i.e.* qui partagent une face avec un voxel vide ou d'un voxel d'identifiant différent, vont s'éroder progressivement. Le terme $\epsilon(\mathbf{p})$ représente la quantité d'érosion en un point $\mathbf{p}$ dans l'espace à une certaine itération avec $\epsilon(\mathbf{p}) \in [min, max]$. Nous définissons l'évolution de l'état d'un voxel $\mathcal{V}_{ijk}$ comme :

$$\rho_{ijk} \leftarrow \rho_{ijk} - \epsilon(\mathbf{p})\,\omega_{ij}(\mathbf{p})$$

3.4.4 Processus de génération du maillage

Pour chaque pierre, nous caractérisons les voxels de la grille avec les identifiants correspondant et nous définissons une représentation implicite de la surface du volume de la pierre. Nous définissons la valeur de la fonction du champ de potentiel au centre de la cellule $\mathbf{v}_{ijk}$ comme :

$$r(\mathbf{v}_{ijk}) = 2\,\rho_{ijk} - 1 \qquad r(\mathbf{v}_{ijk}) \in [-1, 1]$$

La fonction de potentiel $r(\mathbf{p})$ pour chaque point de l'espace est obtenue par une interpolation tri-linéaire de ces valeurs. Ainsi, nous pouvons créer facilement le maillage des pierres par différentes techniques de maillage de surface implicite (BW97). Nous obtenons ainsi un maillage pour l'ensemble des pierres de l'ensemble des Corner Cubes.

3.5 Instanciation des pierres

Les empilements de pierres sont créés en utilisant un objet support $\mathcal{O}$ qui définit la forme des empilements de pierres. Nous pouvons utiliser n'importe quel support qui nous fournit une fonction d'appartenance d'un point à l'objet, noté $s(\mathbf{p})$, la fonction retourne 0 si $\mathbf{p}$ est en dehors et 1 si $\mathbf{p}$ est à l'intérieur de l'objet. Dans notre système, nous utilisons des représentations implicites (Figure 3.23, 3.24), des modèles procéduraux (Figure 3.25) et des maillages de triangles fermés (Figure 3.19).

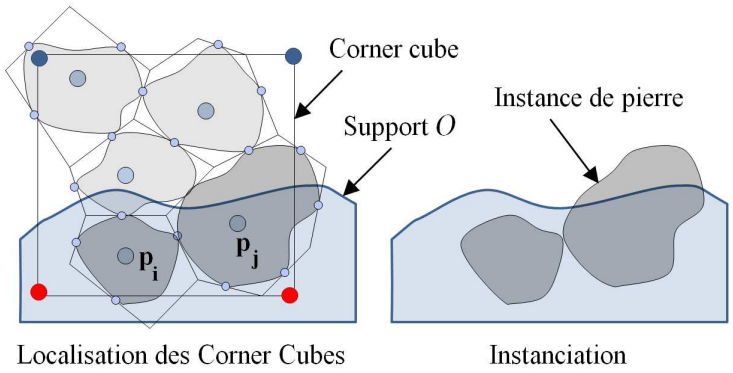


FIGURE 3.17 – Processus d'instanciation de pierres

3.5.1 Principe de distribution et sélection des pierres

Les Corner Cubes sont alignés sur une grille tridimensionnelle. Une couleur est définie pour chaque point de cette grille afin de déterminer la couleur des sommets des cubes. La couleur est obtenue en effectuant une distribution aléatoire stochastique pour chaque point de la grille. Nous générons la forme de l'empilement en analysant uniquement les Corner Cubes intersectant l'objet $\mathcal{O}$ (Figure 3.17).

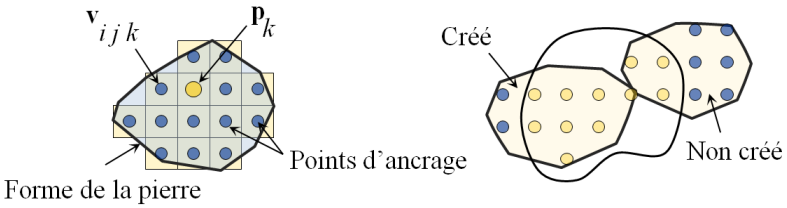


FIGURE 3.18 – Processus de sélection de pierre

Nous associons à chaque pierre un ensemble de points d'ancrage $\mathcal{A} = \{\mathbf{a}_i\}$ qui sont définis comme l'union des centres $\mathbf{v}_{ijk}$ de chaque voxel et le point germe $\mathbf{p}_k$. Pour tous les points d'ancrage, nous évaluons la fonction d'appartenance des points à l'objet et nous calculons le ratio du nombre de points d'ancrage à l'intérieur et à l'extérieur de l'objet $\mathcal{O}$ (Figure 3.18). Nous instancions seulement les pierres qui ont un nombre de points d'ancrage à l'intérieur de l'objet supérieur à un seuil. En pratique, nous instancions les pierres ayant plus de la moitié des points d'ancrage à l'intérieur de l'objet. Cette méthode permet d'éviter la création de pierres contenant une petite intersection avec la géométrie de l'objet support.

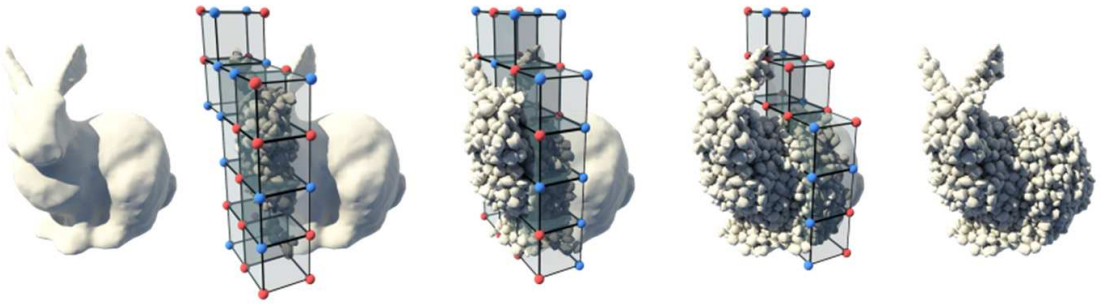


FIGURE 3.19 – Un maillage de lapin converti en un ensemble de maillages de pierres.

Afin d'optimiser les performances d'affichage des empilements de pierres, nous instancions uniquement les pierres se situant à la surface de l'objet $\mathcal{O}$ (Figure 3.20). Pour ne pas voir les instances de pierres manquantes de l'empilement, nous supprimons uniquement les instances se situant à une distance minimale d de la surface. Comme les pierres sont imbriquées les unes dans les autres, il suffit d'utiliser d tel que $d > 3\,r_p$ avec r_p le rayon maximal du volume de Voronoï moyen.

3.5.2 Résolution des problèmes de placement

Lors du placement des instances de pierres sur un terrain, des artefacts visuels d'intersection ou de non stabilité des pierres avec le terrain peuvent se produire. Lors de la génération des pierres, l'empilement vérifie que toutes les pierres sont en contact les unes des autres et que cet empilement soit visuellement plausible. Par contre cette propriété reste vraie seulement à l'intérieur de l'empilement, et n'est pas vérifiée avec les éléments extérieurs de la scène, en particulier le sol. En effet, lors de l'instanciation

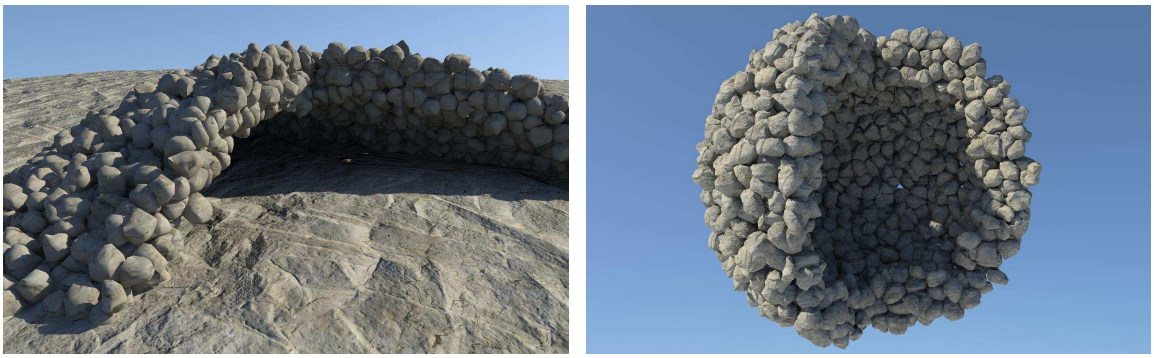


FIGURE 3.20 – Plusieurs coupes d'empilement de pierres permettant de voir la structure des pierres instanciées.

des pierres dans un volume, les pierres ne vérifient pas la propriété de contacts avec la surface du volume (Figure 3.22). Ces artefacts diminuent le réalisme de la scène lors de la création d'empilement contenant peu d'instances.

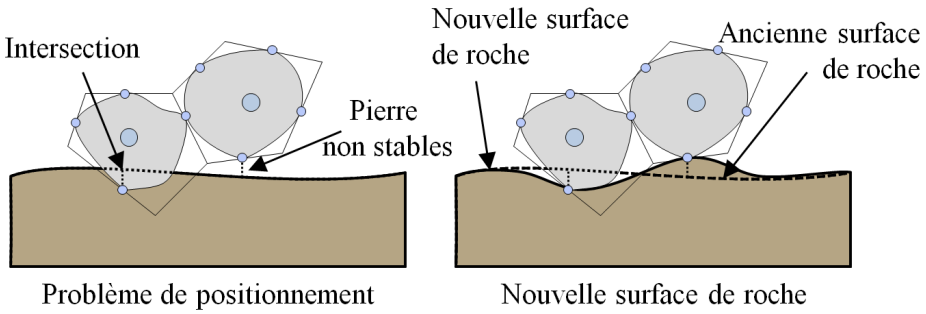


FIGURE 3.21 – Deux configuration non réalistes où les pierres intersectent la surface du terrain ou ne sont pas stables.

Dans le cas d'empilement de pierres sur un terrain, les pierres qui ne sont pas en contact avec le sol diminuent le réalisme de la scène. Par contre les intersections des pierres avec le terrain sont plausibles en fonction du matériau intersecté. En effet, dans la nature, les pierres se mélangent avec d'autres matériaux granuleux comme la terre, le sable ou encore des graviers. En revanche, l'intersection d'une pierre avec de la roche n'est pas réaliste et nécessite un traitement particulier.

Pour rendre la scène plus réaliste, nous proposons de modifier la surface du terrain afin d'obtenir un contact entre les éléments. Notre méthode opère en 2 étapes (Figure 3.22) :

1. Nous recherchons l'ensemble des instances de pierres qui ne sont pas stables ou qui intersectent la roche. Cette recherche s'effectue en analysant les points de contact de toutes les instances. Pour tous les points de contact dont une pierre voisine a été supprimée, nous analysons leurs positions par rapport au terrain. Nous pouvons ainsi définir si un point est à l'intérieur d'une couche de roche ou si un point est dans une couche d'air et donc pas stable.

2. La surface du terrain est ensuite déformée afin de raccorder la surface du terrain au point de contact. Cette déformation s'effectue sur un voisinage de rayon équivalent au rayon moyen de la pierre r_p. Elle est atténuée en fonction de sa distance au point de contact.

FIGURE 3.22 – Les deux premières images montrent les artefacts d'instabilité et d'intersection entre un rocher et le sol. La troisième image montre la solution obtenue après la déformation du sol. La dernière image montre une pierre intersectant une couche de sable qui n'a pas besoin d'être déformée.

En général,la taille de l'artefact est de petite taille. De ce fait, une déformation locale de la surface du terrain reste suffisante pour garder un résultat réaliste.

3.6 Résultats et discussion

Nous avons implémenté notre algorithme de génération apériodique de pierres en C++ et nous l'avons intégré au modeleur de terrain. Nous avons appliqué notre modèle pour créer différentes scènes : un canyon (Figure 3.23), un jardin zen (Figure 3.25) et une ancienne borie faite de pierres (Figure 3.24). Les rendus ont été effectués en utilisant le logiciel Mental Ray pour bénéficier d'un éclairage global sur le maillage produit par notre méthode. La création des empilements de pierres augmente le réalisme de la scène comparé aux modèles proposant de représenter les empilements par des textures. Le pavage permet d'obtenir des empilements de formes variées sur de grandes étendues, tout en conservant la propriété de contact entre les pierres.

3.6.1 Contrôle de la génération

Lors de la création des Corner Cubes, l'utilisateur peut contrôler simplement et avec un nombre réduit de paramètres les caractéristiques de pierres composant l'empilement. L'utilisateur peut ainsi contrôler la densité de pierres au sein de l'empilement pour générer soit des empilements contenant des pierres de tailles homogènes, soit contenant des pierres de différentes tailles en modifiant la densité de génération. L'utilisateur

FIGURE 3.23 – Un canyon virtuel avec des empilement des pierres et de rocher déposé au fond de la rivière (25 313 pierres).

FIGURE 3.24 – Bories (ancienne hutte de pierres) générés avec notre technique de pavage de pierre (28 032 pierres).

contrôle également la forme de chacune des pierres afin d'augmenter la variété au sein de l'empilement pour définir des pierres rondes, plates ou en forme de galets.

Notre méthode se prête à la modélisation de scène complexe pour instancier interactivement les empilements et en contrôlant à chaque étape l'emplacement précis des pierres ce qui n'est pas possible avec la simulation physique de stabilisation. De plus la simulation physique nécessite d'instancier l'ensemble des pierres de l'empilement ce qui n'est pas nécessaire avec notre modèle qui optimise le nombre d'instance dans la scène. L'utilisateur peut créer des empilements de pierres de formes quelconques. Les empilements peuvent définir les couches de pierres du terrain. L'ancienne borie démontre que notre méthode peut instancier efficacement des milliers de pierres pour créer des modèles architecturaux. Le modèle géométrique représentant la borie a été créé en utilisant notre système Arches (PGMG09a) en désactivant le moteur physique agissant sur les pierres. De ce fait, nous avons pu sculpter le volume de la borie très rapidement. Enfin, des empilements de pierres à finalité artistique peuvent être réalisés en transformant des maillages en empilements (Figure 3.26).

FIGURE 3.25 – Un jardin zen rassemblant 94 658 petites pierres ratissées selon des motifs linéaires et circulaires.

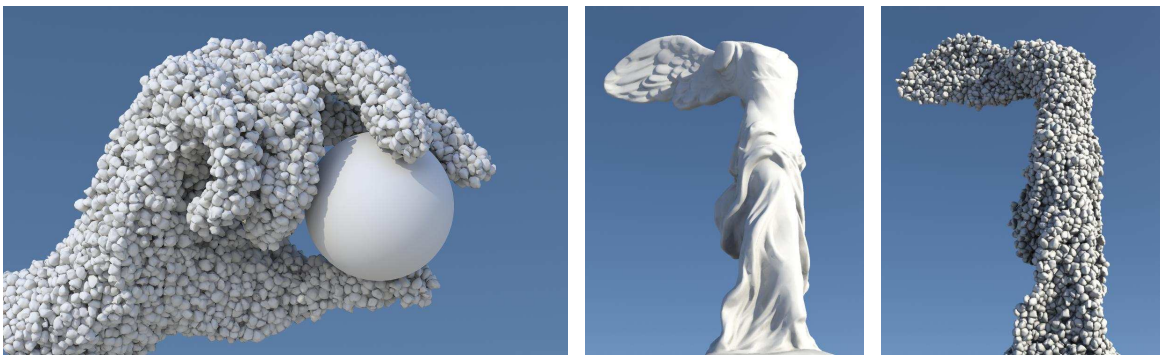


FIGURE 3.26 – Un ensemble de modèles représentant une main et une statue.

3.6.2 Temps de calcul

Notre technique de génération d'empilement peut créer des milliers de pierres empilées les unes sur les autres très efficacement. Le tableau 3.1 rapporte les temps de génération des maillages de pierres sur l'ensemble des cubes apériodiques en fonction du paramètre du nombre de cellules n. Le calcul du volume de Voronoï et le processus d'érosion sont exécutés en utilisant une grille plus fine $\mathcal{G}$, avec $n_{\mathcal{G}} = 10\,n$ pour que la résolution puisse être assez fine pour capturer les détails des petites pierres. Le nombre de pierres minimal m en utilisant une seule pierre par cellule et c couleurs et n cellules est défini par l'équation :

$$m = c + 3\,(n-2)\,c^2 + 3\,(n-2)^2\,c^4 + (n-2)^3\,c^8$$

Nous avons utilisé uniquement $n = 2$ couleurs par sommets du Corner Cube pour minimiser le nombre de combinaisons. Même avec $n = 2$, le nombre de pierres reste élevé : $m = 7\,386$.

Nous pouvons noter que le temps de création des points germes des Corner Cubes est négligeable : la plupart du temps est passé dans la génération des volume de Voronoï , le processus d'érosion et la création des maillages de chaque pierre à partir de leur représentation implicite.

n	Pierres	Points	Voronoï	Erosion	Maillage	Total
5	7 386	0	19	1	3	23
6	17 202	0	63	3	7	73
7	33 262	0	152	6	18	176

TABLEAU 3.1 – Temps (en secondes) pour la génération des cubes en fonction du niveau de discrétisation du cube n (le tableau montre également le nombre de pierres générées) : le tableau présente les temps pour la génération des points, la création des volumes de Voronoï , l'érosion, et la génération du maillage.

Rappelons que la géométrie des pierres dans les cubes est générée une seule fois dans une étape de pré-traitement. Par contre, l'instanciation des pierres dans la scène peut être effectuée très efficacement. Ce temps d'instanciation dépend de l'efficacité de la fonction d'appartenance qui définit si un point appartient ou non au volume. En fonction des modèles à instancier, le temps peut donc varier, mais dans la majorité des cas le temps d'instanciation est inférieur à 1 seconde.

3.6.3 Perspectives

Un problème lié à la structure actuelle du modèle est l'importance du volume des données à traiter en mémoire. Pour obtenir une distribution apériodique, le nombre de cellules n doit être supérieur ou égal à 5. Pour 5 cellules, cela revient à stocker en mémoire 7 355 pierres pour 2 couleurs par sommets soit 256 cubes. En comptant une moyenne minimum de 200 triangles par maillage de pierres, nous obtenons approximativement un total de 1 400 000 triangles. Ce nombre élevé de triangles est un facteur limitatif dans la création de scènes complexes même avec les capacités des machines actuelles. De plus lors de la création de petites scènes contenant moins de 7 000 instances, la taille de la structure de données est plus importante que la taille finale des données ce qui n'est pas le résultat souhaité pour une méthode de pavage.

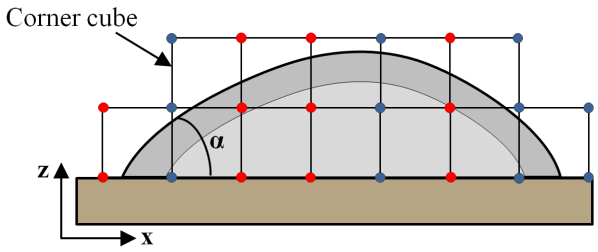


FIGURE 3.27 – Instanciation d'un empilement de pierre sur un terrain.

Pour réduire les coûts mémoires, nous proposons de diminuer le nombre de cube à générer. Plusieurs solutions peuvent être envisagées. La première consiste à extraire un sous ensemble de cubes par une analyse combinatoire afin de garder l'apériodicité. Une étude proche à celle de Culik(CK95) serait nécessaire. Nous rappelons que Culik

a proposé une solution pour réduire le nombre de combinaisons des Wang Cubes de 64 à 21 cubes en utilisant 2 couleurs par axe. Cette méthode permet de préserver l'apériodicité des cubes sur l'ensemble des dimensions.

n	16 Corner Cubes	256 Corner Cubes
5	788	7 386
6	1 642	17 202
7	2 952	33 262
8	4 814	57 098
9	7 324	90 246

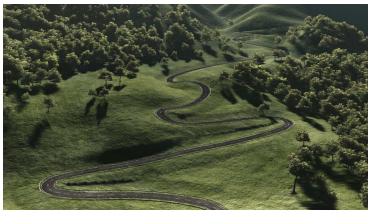
TABLEAU 3.2 – Nombre de pierres selon un nombre différent de Corner Cubes.

La deuxième solution que nous proposons pour diminuer le nombre de Corner Cubes est d'enlever une dimension à l'apériodicité. Cette solution se place uniquement dans le cadre de la génération d'empilements de pierres sur un terrain. Les empilements de pierres sur un terrain peuvent être définis comme une carte d'élévation de données ayant un angle de repos de 40°. Dans ce contexte, la périodicité sur l'axe **z** n'est pas visible au sein d'un même empilement. Nous rappelons que seul la surface supérieure de l'empilement est instanciée. De ce fait, la périodicité est visible uniquement sur les axes **x** et **y** lors d'instanciation de surfaces planes. En supprimant l'apériodicité sur l'axe **z**, nous pouvons réduire le nombre de Corner Cubes de $2^8 = 256$ cubes à $2^4 = 16$ cubes. L'instanciation est effectuée en associant à chaque point de la grille la même couleur sur l'axe **z** (Figure 3.27). Cette représentation permet de diminuer énormément le nombre de pierres à générer 3.2. Nous pouvons observer que le nombre de Cubes générés par cette méthode diminue d'un facteur d'environ 10.

Chapitre 4

Génération de routes





Sommaire

Les résultats présentés dans ce chapitre ont fait l'objet d'une publication dans la revue internationale Computer Graphics Forum (GPMG10).

De nombreuses méthodes ont été introduites pour créer des paysages naturels et urbains. Cela fait plus de trente ans que des travaux sont proposés pour créer des terrains (MKM89, GMe09) ou de la végétation (PHL+09, DHL+98). A l'inverse, la création de paysages urbains est récente et n'a été étudiée que dans la dernière décennie sur la génération de villes. Pour obtenir une ville réaliste, il est nécessaire de définir la structure de la ville représentée par son réseau de rues. Plusieurs méthodes (PM01, CEW+08) proposent de créer ce réseau de rues par des règles de croissance ou en se basant sur des champs de tenseur. Par contre, il n'existe à ce jour aucune technique de génération automatique de routes de campagne mais uniquement des méthodes d'édition manuelles ou par esquisses de routes de campagne. Ces méthodes produisent des trajectoires non réalistes car ne permettent pas de respecter facilement les contraintes de création de routes s'adaptant aux caractéristiques du terrain.

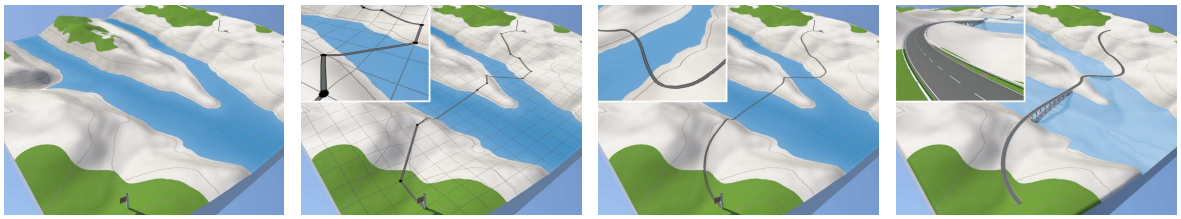


FIGURE 4.1 – Processus global de génération de routes.

Dans ce chapitre, nous présentons une méthode pour résoudre le problème de génération de routes de campagne dans un environnement défini par de multiples caractéristiques (relief, végétation, eau). Pour ce faire, nous utilisons un nouveau modèle s'appuyant sur un algorithme de plus court chemin. Notre méthode consiste à générer une trajectoire réaliste entre un point initial et un point final en prenant en compte l'environnement. Les paramètres de la scène pris en compte sont le relief du terrain, la nature du terrain, la densité de végétation et la profondeur de l'eau. La trajectoire est obtenue en minimisant une fonction de coût définie par l'ensemble des caractéristiques du terrain. Elle est ensuite convertie en un ensemble de clothoïdes pour définir une route réaliste. Le maillage de la route est généré procéduralement en suivant la trajectoire de la route. Le terrain est également modifié par des opérations d'excavation et de remblai adaptés afin de pouvoir déposer le bitume de la route.

4.1 Plus court chemin anisotrope discret

Dans cette section, nous présentons le problème du calcul du plus court chemin anisotrope, et les algorithmes pour trouver le plus court chemin sur un graphe pour la génération de routes, tunnels et ponts sur des terrains complexes.

4.1.1 Problème du plus court chemin anisotrope

Le problème du plus court chemin anisotrope pondéré sur un domaine continu consiste à calculer le chemin entre deux points en minimisant l'intégrale curviligne d'une fonction de coût le long de ce chemin.

Considérons une région compacte $\Omega \in \mathsf{R}^2$, un point initial $\mathbf{a}$ et un point final $\mathbf{b}$. Notre objectif est de calculer un chemin continu ρ de $\mathbf{a}$ vers $\mathbf{b}$ qui minimise l'intégrale curviligne de la fonction de coût $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ qui dépend de la position $\mathbf{p}$ et des deux premières dérivées notées respectivement $\dot{\mathbf{p}}$ et $\ddot{\mathbf{p}}$ sur ce chemin.

Pour formaliser le problème, nous noterons $\mathcal{P}$ l'ensemble de tous les chemins continus dans Ω de $\mathbf{a}$ vers $\mathbf{b}$ qui sont deux fois continus dérivables par morceaux. Cela signifie que $\mathcal{P}$ désigne l'ensemble des fonctions continues $\rho : [0, T] \to \Omega$, avec $T > 0$ pour lesquelles $\rho(0) = \mathbf{a}$ et $\rho(T) = \mathbf{b}$. Soit $C : \mathcal{P} \to [0, \infty($ la fonction caractérisant le coût d'un chemin $\rho \in \mathcal{P}$:

$$C(\rho) = \int_0^T c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, \mathrm{d}t$$

Le problème continu du plus court chemin anisotrope consiste à trouver un chemin ρ^* qui minimise la fonctionnelle $C(\rho)$:

$$C(\rho^*) = \min_{\rho \in \mathcal{P}} C(\rho)$$

4.1.1.1 Rappel sur les méthodes de recherche de plus court chemin

Dans le domaine de la géométrie algorithmique, plusieurs travaux ont été réalisés pour rechercher le plus court chemin sur un graphe. De nombreuses techniques se concentrent sur le cas isotrope c'est à dire lorsque la fonction de coût ne dépend que de la position. Plusieurs algorithmes ont été proposés pour résoudre le problème sur des régions planaires composées d'obstacles et quand la fonction de coût est indépendante de la vitesse (MP91, AMS00). La complexité du problème augmente lorsque la fonction de coût est continue mais reste indépendante de la vitesse. Une autre famille d'algorithmes a été proposée pour résoudre le problème du plus court chemin isotrope en résolvant l'équation discrète de Hamilton Jacobi Bellmann (Tsi95, PBT98). Le problème discret permet de diminuer la complexité pour accélérer les calculs. Par contre ces méthodes ne peuvent pas se généraliser au cas anisotrope.

Peu de travaux ont été consacrés au cas anisotrope lorsque la fonction de coût dépend de la position, de la vitesse et de l'accélération. Quelques techniques ont été proposées pour prendre en compte la pente du terrain (RR90, LMS99). Aleksandrov (AMS00, AMS05) propose une technique de discrétisation pour résoudre le problème du plus court chemin sur un terrain pondéré. En utilisant un maillage triangulaire, la méthode consiste à insérer des points de Steiner sur les arêtes du graphe de recherche et sur la bissectrice des triangles (Figure 4.2). Les points de Steiner sont des points

ajoutés au graphe qui ne sont pas membres du graphe d'entrée. Cette méthode permet d'augmenter le nombre de trajectoires possibles pour mieux contrôler la courbure.

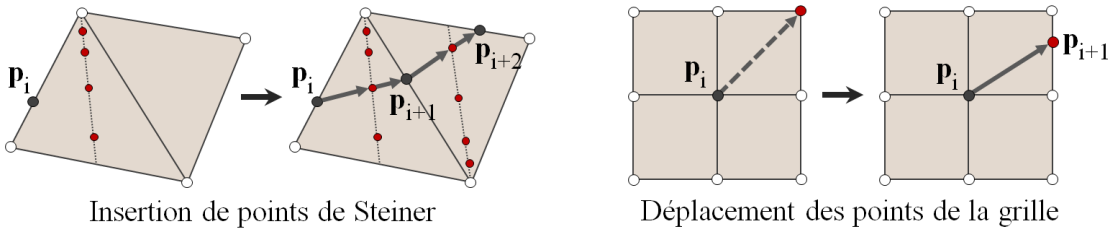


FIGURE 4.2 – Recherche d'un plus court chemin se rapprochant du modèle continu sur une grille et un maillage polygonal.

Kim et al. (KH03) proposent une méthode sur une discrétisation non uniforme de la région continue par un algorithme d'échantillonnage en nid d'abeilles. Une autre technique présentée par Jia (JV04) utilise une grille régulière comme base de discrétisation. Comme le coût est anisotrope, les chemins rectilignes reliant les points adjacents de la grille ne permettent pas d'obtenir une trajectoire optimale. Pour surmonter cette limitation de limite de direction, la méthode recherche la trajectoire sur un graphe dont la position des points peut être modifiée (Figure 4.2).

Le problème de ces méthodes est qu'elles sont coûteuses en temps de calcul et donc pas applicables sur de grands domaines.

4.1.1.2 Plus court chemin anisotrope discret

Pour résoudre le problème de plus court chemin anisotrope pondéré, nous approximons la solution en discrétisant uniformément la région Ω sous forme de grille. Nous définissons le chemin comme un enchaînement de segments entre les points de la grille. Pour un nombre fini de points de la grille, cette procédure convertit le problème continu de plus court chemin en un problème de plus court chemin sur un graphe fini $\mathcal{G}$.

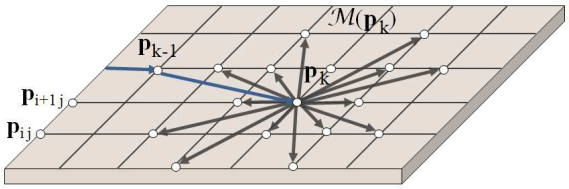


FIGURE 4.3 – Notations pour les points de la grille $\mathbf{p}_{ij}$, le masque $\mathcal{M}(\mathbf{p}_k)$ et le chemin $\rho^* = \{\mathbf{p}_k\}_{k\in[0,n]}$.

Soit $\mathbf{p}_{ij}$, $(i,j) \in [0, n-1]^2$ les sommets de la grille distribués selon un échantillonnage uniforme sur le domaine de recherche. Ces points correspondent aux nœuds du graphe $\mathcal{G}$. La fonction de coût C étant anisotrope, les segments connectant un point $\mathbf{p}_{ij}$ aux points adjacents de la grille ne donneront pas une trajectoire optimale (JV04).

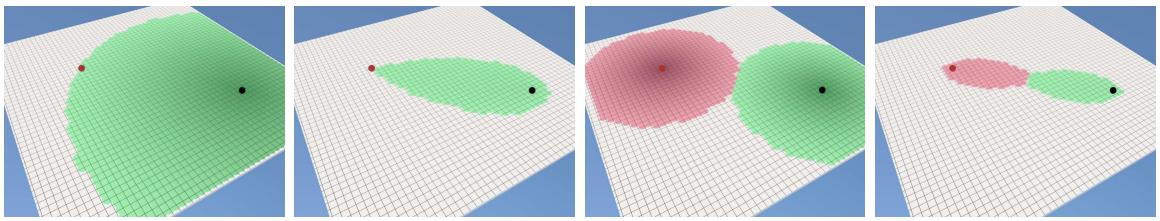
L'originalité de notre approche est de considérer que chaque point $\mathbf{p}_{ij}$ est implicitement connecté à un vaste ensemble de points voisins distant d'une distance r. Nous définissons ce sous-ensemble comme le r-voisinage de $\mathbf{p}_{ij}$:

$$\mathcal{M}(\mathbf{p}_{ij}) \subset \{\mathbf{q},\, \|\mathbf{p}_{ij} - \mathbf{q}\| \leq r\}$$

Stocker explicitement ces arcs serait trop coûteux en mémoire. Le nombre d'arcs n_a à explorer pour un point de la grille $\mathbf{p}_{ij}$ est défini par le rayon r et croît en $\mathcal{O}(r^2)$. Pour l'ensemble du graphe, le nombre d'arcs total à stocker serait équivalent à $n^2\, n_a^2$ arcs. Par exemple, pour une grille de taille $n = 512$ et avec $n_a = 5$ arcs par points, on obtient un total de 6 553 600 arcs. Par conséquent, nous utilisons des masques génériques de segments de chemin, notés $\mathcal{M}_k$, permettant de stocker implicitement la connectivité entre les points de la grille (Section 4.3). Cette technique nous permet de résoudre le problème de limite de direction en considérant un r-voisinage quelconque entre les points de la grille $\mathbf{p}_{ij}$, même avec r grand, ce qui sera utile pour la génération de ponts et de tunnels.

4.1.2 Calcul du plus court chemin

En théorie des graphes, plusieurs méthodes ont été proposées pour rechercher des plus courts chemins sur des graphes. Les algorithmes de Dijkstra, A*, Bellman-Ford, Floyd-Warshall et par recherche bidirectionnelle recherchent un chemin entre un point initial $\mathbf{a}$ et un point final $\mathbf{b}$ en minimisant la fonctionnelle $C(\rho)$ afin que la somme des coûts des arcs reliant les nœuds soit minimale.



Dijkstra A* Dijkstra Bidirectionnel A* Bidirectionnel

FIGURE 4.4 – Les différents parcours d'exploration selon les algorithmes correspondant : Dijkstra, A* et en bidirectionnel. Le point de départ est représenté par un point noir et le point d'arrivée par un point rouge.

4.1.2.1 Algorithme de Dijkstra

Cet algorithme calcule le plus court chemin sur un graphe $\mathcal{G}$ entre un point de départ $\mathbf{a}$ et un point d'arrivée $\mathbf{b}$. Chaque nœud de la grille contient la valeur de son coût $l(\mathbf{p}_{ij})$ et le nœud de son prédécesseur $p(\mathbf{p}_{ij})$ afin de retrouver le chemin. Le coût $l(\mathbf{p}_{ij})$ correspond au coût du chemin pour arriver au point $\mathbf{p}_{ij}$ depuis le point de départ. La valeur du coût de tous les points $l(\mathbf{p}_{ij})$ est initialisée à l'infini, alors que la valeur

du point de départ $l(\mathbf{a})$ est fixée à 0. La file d'attente $\mathcal{Q}$ va contenir à chaque itération l'ensemble des nœuds de frontière à explorer. Cette file d'attente $\mathcal{Q}$ est initialisée avec le point initial $\mathbf{a}$. La boucle principale de l'algorithme est la suivante :

1. Tant que $\mathcal{Q}$ n'est pas vide, sélectionner le point $\mathbf{p}_{ij}$ ayant la plus petite valeur de coût $l(\mathbf{p}_{ij})$ à partir de la file d'attente.
2. Si la destination a été trouvée, c'est à dire si $\mathbf{p}_{ij} = \mathbf{b}$, l'algorithme est arrêté.
3. Pour tous les points $\mathbf{q} \in \mathcal{M}_k(\mathbf{p}_{ij})$, évaluer le coût $e(\mathbf{p}_{ij}, \mathbf{q})$ sur le segment $[\mathbf{p}_{ij}, \mathbf{q}]$. Si $l(\mathbf{p}_{ij})+e(\mathbf{p}_{ij}, \mathbf{q}) < l(\mathbf{q})$ alors le prédécesseur de $\mathbf{q}$ est $\mathbf{p}_{ij}$, $l(\mathbf{q}) = l(\mathbf{p}_{ij})+e(\mathbf{p}_{ij}, \mathbf{q})$ et le nœud $\mathbf{q}$ est ajouté à la file d'attente.

Cet algorithme génère un chemin discret caractérisé par un ensemble de points de la grille, noté $\rho^* = \{\mathbf{p}_k\}$, $k \in [0, n]$. L'ensemble des points de la grille ρ^* est obtenu en partant du nœud $\mathbf{p}_{ij} = \mathbf{b}$ puis en suivant itérativement les prédécesseurs. Cet étape s'arrête lorsque le nœud courant $\mathbf{p}_{ij} = \mathbf{a}$.

4.1.2.2 Algorithme A*

A* est considéré comme un des meilleurs algorithmes de recherche du plus court chemin. La méthode utilisée est similaire à celle de Dijkstra à la différence que l'algorithme utilise une fonction heuristique $h(\mathbf{p})$ afin d'accélérer la recherche du plus court chemin en déterminant un ordre d'exploration du graphe. Un algorithme de recherche qui garantit de toujours trouver le chemin le plus court à un but s'appelle un algorithme admissible. Si A* utilise une heuristique qui ne surestime jamais le coût du but, A* est avéré admissible. Dans plusieurs applications, $h(\mathbf{p})$ représente la distance euclidienne du point courant au point d'arrivée. Dans notre modèle, nous utilisons la distance euclidienne comme heuristique car c'est la seule à ne pas surestimer la coût même avec l'ensemble des caractéristiques de notre modèle. L'heuristique doit vérifier $h(\mathbf{p}) \leq e(\mathbf{p}, \mathbf{q}) + h(\mathbf{q})$ pour chaque point $\mathbf{p}$, $\mathbf{q}$ du graphe avec $e(\mathbf{p}, \mathbf{q})$ le coût de l'arc entre $\mathbf{p}$ et $\mathbf{q}$. Ainsi h est considéré comme monotone et consistant. La valeur du coût de tous les points $l(\mathbf{p})$ est initialisée à l'infinie, tandis que la valeur du point $l(\mathbf{a})$ est mise à $h(\mathbf{b})$. La boucle principale de l'algorithme est la suivante :

1. Tant que $\mathcal{Q}$ n'est pas vide, sélectionner le point $\mathbf{p}_{ij}$ ayant la plus petite valeur de coût $l(\mathbf{p}_{ij})$ à partir de la file d'attente.
2. Si la destination a été trouvée, c'est à dire si $\mathbf{p}_{ij} = \mathbf{b}$, l'algorithme est arrêté.
3. Pour tous les points $\mathbf{q} \in \mathcal{M}_k(\mathbf{p}_{ij})$, évaluer le coût $e(\mathbf{p}_{ij}, \mathbf{q})+h(\mathbf{q})$ sur les segments $[\mathbf{p}_{ij}, \mathbf{q}]$. Si $l(\mathbf{p}_{ij}) + e(\mathbf{p}_{ij}, \mathbf{q}) + h(\mathbf{q}) < l(\mathbf{q})$ alors le prédécesseur de $\mathbf{q}$ est $\mathbf{p}_{ij}$, $l(\mathbf{q}) = l(\mathbf{p}_{ij}) + e(\mathbf{p}_{ij}, \mathbf{q})$ et le nœud $\mathbf{q}$ est ajouté à la file d'attente.

Cet algorithme est plus efficace que celui de Dijkstra car il explore moins de nœuds du graphe.

4.1.2.3 Algorithme de recherche bidirectionnelle

Cet algorithme est une accélération des algorithmes de plus court chemin s'appuyant sur deux recherches simultanées : une en partant de l'état initial $\mathbf{a}$ et l'une en partant de l'état final $\mathbf{b}$. L'algorithme s'arête lorsque un nœud $\mathbf{p}_{ij}$ a été évalué par les deux recherches. Cette recherche nécessite deux files d'attente $\mathcal{Q}_\mathbf{a}$ et $\mathcal{Q}_\mathbf{b}$ afin d'explorer les nœuds sur les deux fronts de recherche. Cet algorithme s'adapte également à l'algorithme A* en utilisant des heuristiques spécifique à la liste de recherche. Les heuristiques utilisées sont $h_\mathbf{a}(\mathbf{p})$ et $h_\mathbf{b}(\mathbf{p})$ pour évaluer le coût pour tout point $\mathbf{p}$ pour aller au point de départ $h_\mathbf{a}(\mathbf{p})$ et le coût pour aller au point d'arrivée $h_\mathbf{b}(\mathbf{p})$.

4.1.2.4 Modèles utilisés

Nous utilisons les algorithmes de Dijkstra, A* et de recherche bidirectionnelle appliqués aux algorithmes de Dijkstra et A*. Dans notre cas de recherche de trajectoire de route, le coût entre deux nœuds est toujours positif, nous ne détaillons donc pas les résultats obtenus avec Bellman-Ford. L'heuristique utilisée pour l'algorithme A* est défini par le coût de distance entre $\mathbf{a}$ et $\mathbf{b}$ avec une différence de hauteur $\Delta_z = 0$ pour obtenir une heuristique admissible.

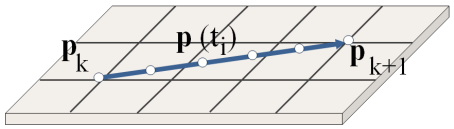


FIGURE 4.5 – Évaluation de la fonction de coût le long d'un segment $[\mathbf{p}_k, \mathbf{p}_{k+1}]$.

L'étape d'évaluation d'un nœud implique le calcul de l'ensemble des points du voisinage $\mathcal{M}_k(\mathbf{p}_{ij})$ afin de définir les arcs entre les nœuds. Cette étape nécessite également l'évaluation de la fonction de coût le long des segments de droite reliant deux points de la grille (Figure 4.5). Soit $[\mathbf{p}_k, \mathbf{p}_{k+1}]$ un segment du chemin et t_k, t_{k+1} les paramètres correspondant à $\mathbf{p}_k$ et $\mathbf{p}_{k+1}$. L'intégrale curviligne est définie comme suit :

$$e(\mathbf{p}_k, \mathbf{p}_{k+1}) = \int_{t_k}^{t_{k+1}} c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, \mathrm{d}t$$

Nous approximons l'intégrale par une somme finie en discrétisant le domaine d'intégration en n intervalles. L'évaluation du coût $c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))$ en fonction des caractéristiques du terrain s'effectue sur l'ensemble des points de la courbes qui sont définis par les intervalles.

4.2 Fonctions de coût

Dans cette section, nous présentons un ensemble de fonctions de coût permettant de définir l'influence des caractéristiques du terrain sur la trajectoire de la route. Ces

caractéristiques sont définies par la pente du terrain et les obstacles naturels comme les rivières, les lacs ou encore la forêt. Nous définissons la fonction de coût global $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ par une somme pondérée de fonctions évaluant l'influence des différentes caractéristiques du terrain sur la route :

$$c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}) = \sum_{i=0}^{i=n-1} \mu_i \circ \kappa_i(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$$

Les fonctions $\kappa_i : \mathsf{R}^3 \times \mathsf{R}^3 \times \mathsf{R}^3 \rightarrow \mathsf{R}$ évaluent les différentes caractéristiques du terrain ainsi que les caractéristiques géométriques de la trajectoire de la route au point $\mathbf{p}$. Les fonctions μ_i sont des fonctions de transfert pondérant et combinant l'influence des caractéristiques $\kappa_i(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$. Les fonctions de transfert permettent à l'utilisateur de contrôler l'influence de la scène par ces paramètres et donc de contrôler le comportement de l'algorithme du plus court chemin.

4.2.1 Fonctions caractéristiques

Les fonctions caractéristiques sont évaluées pour un point $\mathbf{p}$ situé sur la trajectoire d'un chemin. Ces fonctions sont dépendantes de la scène dans laquelle est définie l'ensemble des obstacles mais également la trajectoire courante. Nous calculons l'ensemble des fonctions caractéristiques suivantes :

1. La pente du terrain $s(\mathbf{p}, \dot{\mathbf{p}})$ est obtenue en évaluant le pourcentage de pente défini par l'angle entre le sol et la dérivée première. La fonction $\kappa_s(\mathbf{p}, \dot{\mathbf{p}})$ retourne l'évaluation de la pente au point $\mathbf{p}$ de la trajectoire courante.
2. La densité de végétation $v(\mathbf{p})$ est calculée en évaluant le nombre d'arbres se trouvant dans $\Omega(\mathbf{p}, r)$. La fonction $\kappa_v(\mathbf{p})$ représente la densité d'arbre présent dans $\Omega(\mathbf{p}, r)$.
3. La profondeur de l'eau $w(\mathbf{p})$ est définie comme la hauteur maximale d'eau se trouvant dans le voisinage $\Omega(\mathbf{p}, r)$. La fonction $\kappa_w(\mathbf{p})$ représente la profondeur maximale d'eau dans $\Omega(\mathbf{p}, r)$.
4. La courbure de la route $\gamma(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ est calculée en évaluant les deux premières dérivées au point $\mathbf{p}$. La fonction $\kappa_\gamma(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ représente le rayon de courbure de la courbe au point $\mathbf{p}$.
5. La hauteur au sol $h(\mathbf{p})$ est définie comme la hauteur maximale Δz définie par la différence entre $\mathbf{p}$ et le sol. La fonction $\kappa_{\mathrm{h}}(\mathbf{p})$ représente la différence de hauteur en mètre au point $\mathbf{p}$.
6. La profondeur dans le sol $d(\mathbf{p})$ définit la profondeur maximale Δz définie par la différence entre $\mathbf{p}$ et le sol. La fonction $\kappa_{\mathrm{d}}(\mathbf{p})$ représente la différence de hauteur en mètre au point $\mathbf{p}$.

L'évaluation des fonctions de densité de végétation, de profondeur d'eau s'effectue sur un domaine $\Omega(\mathbf{p}, r)$. Le domaine $\Omega(\mathbf{p}, r)$ est défini par un disque centré en $\mathbf{p}$ et de rayon r. Le rayon r est défini en fonction de la largeur de la route l et d'une région de

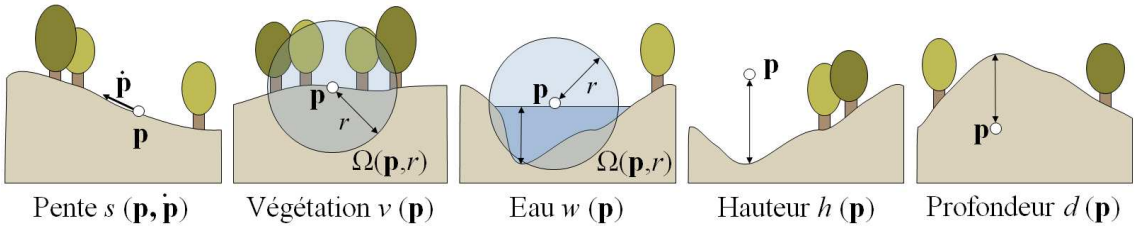


FIGURE 4.6 – Ensemble des caractéristiques analysées pour évaluer le coût de traversée de la scène.

raccord e, soit $r = l + e$. L'évaluation est effectuée sur un terrain continu composé d'un ensemble de couches définissant l'ensemble des caractéristiques. Le temps d'évaluation dépend de la précision du terrain en entrée.

4.2.2 Fonctions de transfert

Les fonctions caractéristiques de la scène sont pondérées par les fonctions de transfert pour contrôler l'influence d'une caractéristique sur la trajectoire globale de la route. Dans notre système, les fonctions de transfert sont caractérisées par des graphiques édités interactivement (Figure 4.7).

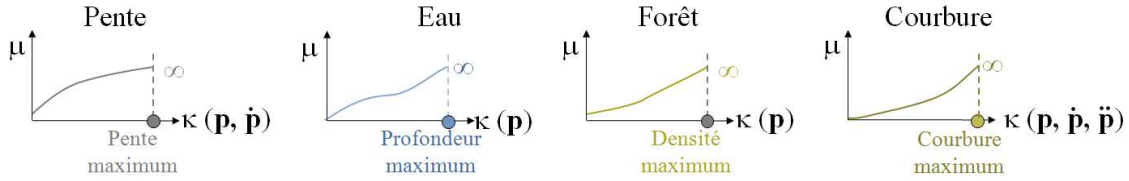


FIGURE 4.7 – Représentation des fonctions de transfert pour les routes de surface.

Les fonctions de transfert sont caractérisées par un seuil, notée κ_0. Si la valeur caractéristique κ est supérieure à κ_0, la fonction de transfert correspondante $\mu(\kappa)$ retourne une valeur infinie. Cela nous permet de contrôler les régions où le chemin ne peut pas être créé. Nous détaillons ici l'ensemble des fonctions de transfert.

La fonction de transfert pour la pente retourne un coût dépendant du degré de la pente. La fonction est représentée par une courbe logarithmique pour pénaliser rapidement le coût de création lorsque la pente augmente. Lorsque la pente est trop raide, la fonction de transfert retourne une valeur infinie. Nous avons défini cette limite par $\kappa_0 = 15°$.

La fonction de transfert pour l'eau retourne un coût dépendant de la profondeur de l'eau. En fonction du type de route à créer que ce soit une route de surface, un pont ou un tunnel, la fonction de transfert est adaptée. Le paramètre κ_0 représente la profondeur d'eau maximale qu'un type de route peut traverser. Par exemple, $\kappa_0 = 0.5m$ pour les routes de surface, par contre $\kappa_0 = 50m$ pour la création de ponts.

La fonction de transfert pour la végétation retourne un coût relatif à la densité de végétation. Cette fonction ne nécessite pas la création d'un seuil, mais pénalise le coût selon la densité de végétation traversée. Dans des contextes particulier, l'utilisateur peut ajouter un seuil κ_0 afin de contourner les forêts trop denses.

La fonction de transfert pour la courbure retourne un coût en fonction de l'angle de courbure de la trajectoire. La fonction retourne un coût élevé lorsque l'angle est élevé. Nous avons défini le seuil κ_0 pour la création de route de surface à $\kappa_0 = 170°$ pour permettre de créer des lacets en montagne. Par contre pour la création de ponts et de tunnels, le seuil κ_0 est plus bas, $\kappa_) = 50°$ pour la création de pont et $\kappa_0 = 70°$ pour la création de tunnel. Pour créer un chemin de fer, l'utilisateur doit mettre un seuil très faible, par exemple $\kappa_0 = 20°$

La fonction de transfert de hauteur retourne un coût dépendant de la position du point $\mathbf{p}$ par rapport au sol. La fonction retourne un coût proportionnel à la hauteur du point. Le seuil κ_0 est défini en fonction de la hauteur maximale que peut supporter la création d'un pont, défini avec $\kappa_0 = 150m$.

La fonction de transfert de profondeur retourne un coût dépendant de la profondeur d'un point dans le sol. La fonction de transfert retourne un coût de construction fixe lorsque la profondeur a dépassée un seuil. Ce seuil a été défini à $15m$. Le coût retourné de $0m$ à $15m$ de profondeur est linéaire. Nous avons défini une limite de profondeur de création de tunnel tel que $\kappa_0 = 1000m$.

4.2.3 Calcul du coût pour les routes, les ponts et les tunnels

4.2.3.1 Routes de surface

La trajectoire d'une route de surface est évaluée sur la plupart des caractéristiques présentées précédemment. Ces caractéristiques sont la pente du terrain et l'ensemble des obstacles naturels tels que les rivières, les lacs et les forêts (Figure 4.8).

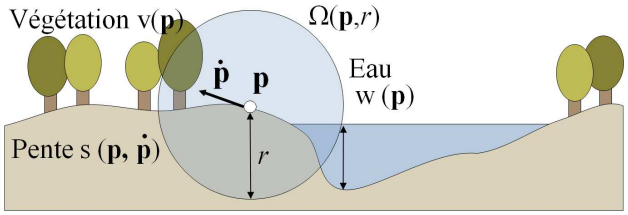


FIGURE 4.8 – Vue d'ensemble des fonctions de coût pour une route de surface : nous évaluons les fonctions caractéristiques $\kappa_i(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ de la scène dans le voisinage de $\mathbf{p}$ en appliquant les fonctions de transfert μ_i pour pondérer leur influence.

La caractéristique de la pente est un paramètre important pour la route de surface car c'est elle qui va permettre d'adapter la trajectoire au relief du terrain. En complément à la pente, la courbure de la route doit être évaluée pour diminuer le nombre de

lacets de la route. La pondération de la traversée de l'eau est très élevée pour que la route ne puisse pas traverser une zone d'eau.

4.2.3.2 Ponts

Les ponts sont des structures complexes qui permettent de traverser des obstacles comme des fleuves, des lacs, ou encore une vallée. La création d'un pont nécessite alors des caractéristiques différentes de la génération de routes. Les caractéristiques utilisées pour créer un pont sont bien évidemment les mêmes que celles de la route mais à laquelle s'ajoute la hauteur du pont par rapport au sol $h(\mathbf{p})$ (Figure 4.9).

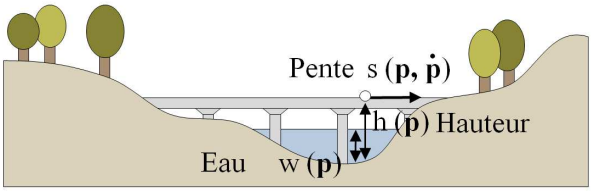


FIGURE 4.9 – Fonction d'évaluation du coût d'un pont.

La fonction de hauteur par rapport au sol $h(\mathbf{p})$ va permettre de ne pas créer de ponts de très basse ou trop haute altitude. Si la trajectoire du pont intersecte le terrain, alors le pont ne peut pas être construit. La pondération des autres caractéristiques comme la pente, la courbure, la végétation et l'eau est différente de celle de la route. La pondération de la pente et de la courbure est plus élevée que celle de la route afin de ne pas créer de trajectoires trop dangereuses pour un pont. Par contre, l'évaluation de la végétation et de l'eau est moins coûteuse que celle de la route.

4.2.3.3 Tunnels

Un tunnel est également une structure complexe qui permet de traverser un autre type d'obstacle tel que les montagnes. Pour évaluer les tunnels, nous utilisons une nouvelle caractéristique qui est la profondeur dans le sol $d(\mathbf{p})$ (Figure 4.10). Si la trajectoire du tunnel est hors du sol, il ne peut être créé.

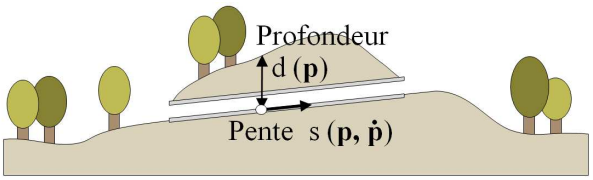


FIGURE 4.10 – Fonction d'évaluation du coût d'un tunnel.

Comme les tunnels s'intègrent dans le paysage, nous n'évaluons pas la caractéristique de végétation. Par contre nous évaluons la caractéristique de la pente, de la

courbure et de l'eau. La pondération utilisée pour la pente et la courbure est la même que celle des routes de surface car elles ont moins de risques que pour les ponts. Nous considérons que le coût de création d'un tunnel sous une rivière ou un lac est élevé car nécessite des matériaux spécifiques, de ce fait nous le traduisons dans notre modèle en augmentant la pondération.

4.3 Masques de segments de chemin

Dans cette section, nous décrivons notre méthode de calcul du plus court chemin anisotrope sur une grille uniforme.

Un problème important de l'échantillonnage sur une grille uniforme est la difficulté à se rapprocher de la position et de la direction d'un plus court chemin continu. Ce problème est dû à la limitation de la direction en se déplaçant sur une grille uniforme (Figure 4.11).

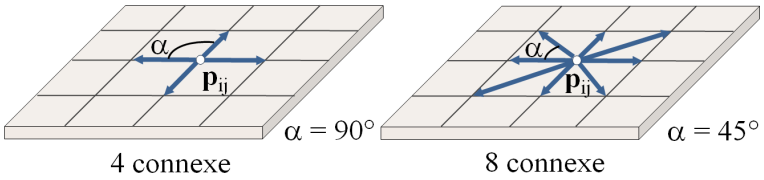


FIGURE 4.11 – La limitation sur le problème de direction en utilisant la 4 et 8 connexité entre les points d'échantillonnage.

Rappelons que la route discrète est un chemin composé de segments de droites reliant les points d'échantillonnage. Les techniques existantes considèrent seulement la 4 et 8 connexité entre les points d'échantillonnage. Par conséquent, les segments de route discrète ne peuvent avoir que 4 ou 8 directions avec une résolution d'angle maximale de 45 degrés. Une méthode de déplacements de nœuds sur le chemin (JV04) peut résoudre en partie ce problème en permettant aux chemins de s'éloigner des points de la grille, mais exigent des itérations de relaxation coûteuses pour déplacer les points d'ancrage du chemin.

Pour résoudre cette limite sur le problème d'orientation, nous proposons d'augmenter la distance de sélection du voisinage afin d'ajouter plus de directions. Cette solution nous permet d'améliorer la précision de l'angle maximal entre deux directions. Cette approche nous permet également de définir des segments reliant des points de la grille très éloignés les uns des autres pour créer des ponts et des tunnels.

4.3.1 Masques de segment

Comme stocker l'ensemble des segments entre les points de la grille serait trop coûteux, nous proposons de stocker l'information de la connectivité entre les points de

la grille par un ensemble de masques de segments, noté $\mathcal{M}_k$ (Figure 4.12).

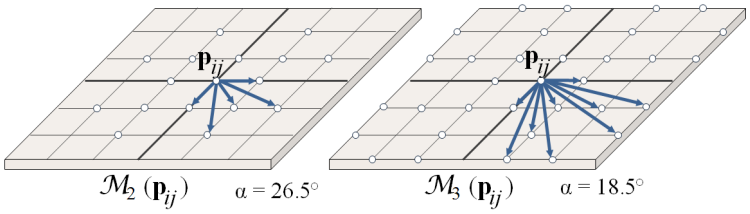


FIGURE 4.12 – Représentation des segments de route pour les masques $\mathcal{M}_2$ et $\mathcal{M}_3$. Seulement quelques segments ont été tracés pour plus de clarté.

Nous définissons les masques de segment $\mathcal{M}_k$ comme l'ensemble des segments reliant un point d'origine $(0, 0)$ à un point $(i, j) \in [-k, k]^2$ tels que le plus grand commun diviseur de i et j est 1 (Figure 4.12). Utiliser cette définition au lieu des $(2\,k + 1)^2$ segments avec $(i, j) \in [-k, k]^2$, permet de diminuer le nombre d'arcs en évitant de vérifier les chemins redondant lors du calcul de l'algorithme du plus court chemin discret.

k	1	2	3	4	5	6	7	8	9	10	11
α	45	26.5	18.5	14.0	11.3	9.5	8.1	7.1	6.3	5.7	5.1
n_k	8	16	32	48	80	96	144	176	224	256	336

TABLEAU 4.1 – Statistiques sur les masques de segment : α représente la résolution de l'angle et n_k le nombre de segment par masques.

Augmenter la taille des masques $\mathcal{M}_k$ nous permet de diminuer l'angle maximum α entre deux segments ce qui réduit la limitation sur le problème de direction au prix d'un nombre plus important d'itérations de l'algorithme. Le Tableau 4.1 indique pour un masque $\mathcal{M}_k$ le nombre n_k de segments et l'angle maximum α. La résolution de l'angle maximum α est définie par l'équation suivante :

$$\alpha = \arctan\left(\frac{1}{k}\right)$$

La Figure 4.13 montre l'influence du paramètre k sur le calcul du plus court chemin entre un point initial dans une vallée et un point final au sommet d'une colline. Pour $k = 1$, la connectivité limitée entre les points de la grille a pour effet de créer un plus court chemin non réaliste. Lorsque k augmente, le chemin devient de plus en plus réaliste en suivant les courbes de niveau du terrain pour générer des lacets afin d'atteindre la position d'arrivée à moindre coût. Nous rappelons que la fonction d'évaluation de la pente pénalise le coût lors de pentes trop élevées. De ce fait la trajectoire suit les lignes de niveau du terrain pour obtenir un coût plus faible que de passer par une pente forte.

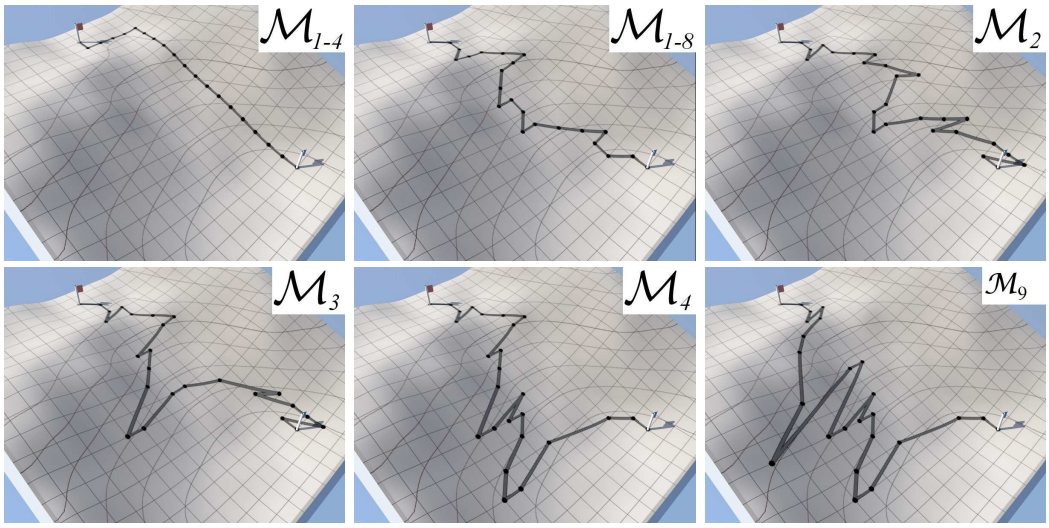


FIGURE 4.13 – Influence de la taille des masques $\mathcal{M}_k$: les résultats pour $\mathcal{M}_5$,$\mathcal{M}_6$,$\mathcal{M}_7$,$\mathcal{M}_8$ sont les mêmes que pour $\mathcal{M}_4$ ce qui signifie que l'augmentation de la taille du masque de voisinage dans la recherche du plus court chemin ne permet pas dans ce cas de trouver une meilleure solution avant le masque $\mathcal{M}_9$ dans ce cas.

Le Tableau 4.2 rapporte le temps (secondes) ainsi que le coût (unité arbitraire) et la longueur du chemin (mètres) pour le calcul du plus court chemin avec des masques de segments différents.

k	Coût	Temps	Longueur
1	103 104	0.42	4 129
2	80 573	0.86	3 731
3	74 265	1.75	3 740
4	73 667	2.95	3 701
5	73 481	5.65	3 717

k	Coût	Temps	Longueur
6	73 351	8.12	3 792
7	73 286	13.43	3 786
8	73 219	18.53	3 786
9	73 160	26.59	3 785
10	73 160	35.07	3 785

TABLEAU 4.2 – Statistiques sur les différentes tailles de masques. Les calculs ont été effectués sur une scène complexe afin de ne pas limiter la recherche de chemin pour des arcs de longues distances. Nous pouvons remarquer que le temps augmente en fonction du nombre d'arcs par contre la longueur du chemin n'est pas dépendante de la taille des masques.

L'augmentation des temps de calcul en fonction de la taille du masque confirme une complexité en $O(k^2)$. Le coût du plus court chemin converge vers une limite lorsque la taille du masque augmente. Nous pouvons observer sur le tableau que les coûts de génération de la trajectoire se stabilisent autour de $k = 5$. La différence des trajectoires obtenues pour des masques allant de $k = 5$ à $k = 10$ est faible comparée aux temps de génération de la trajectoire. En pratique utiliser un masque de taille $k = 5$ va être

un bon compromis entre le temps de génération et la qualité de la trajectoire obtenue. Pour un masque de taille $k = 5$ la résolution de l'angle maximal α est de 11 degrés.

4.3.2 Courbure

Alors que les masques de segments nous permettent de surmonter la limite du problème de direction, le chemin généré comporte souvent trop de lacets diminuant le réalisme de la trajectoire (Figure 4.13). Par conséquent, il est nécessaire d'évaluer la courbure pour limiter les virages.

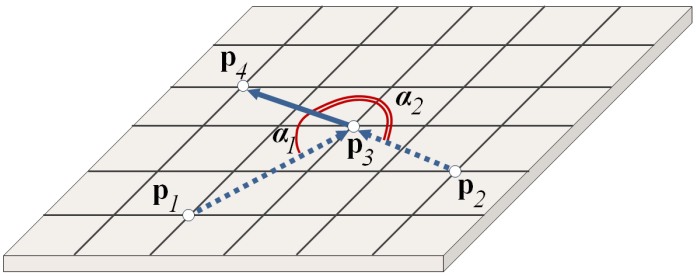


FIGURE 4.14 – Problème de l'évaluation de la courbure.

Pour évaluer le coût de la courbure en un point $\mathbf{p}_i$, il est nécessaire de connaître le point précédent $\mathbf{p}_{i-1}$ et le point suivant $\mathbf{p}_{i+1}$. Le problème lors de l'évaluation d'un nœud vient du fait qu'un nœud à une position (i, j) contient un seul prédécesseur $p(\mathbf{p}_{ij})$. Par exemple (Figure 4.14), lors de l'évaluation d'un nœud $\mathbf{p}_3$ avec $c(\mathbf{p}_1, \mathbf{p}_3) = c(\mathbf{p}_2, \mathbf{p}_3) + \epsilon$ le prédécesseur de $\mathbf{p}_3$ est $p(\mathbf{p}_3) = \mathbf{p}_1$. Les algorithmes actuels de recherche de plus court chemin ne considèrent pas le point $\mathbf{p}_2$ qui donne un coût localement plus élevé. Par contre lors de l'évaluation de la courbure, nous pouvons avoir $c(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4) > c(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) + \epsilon$, le plus court chemin pour aller au point $\mathbf{p}_4$ passe par $\mathbf{p}_2$. Les algorithmes actuels ne prennent pas en compte tous les chemins lors de l'évaluation de la courbure. De ce fait il est important de connaître l'ensemble des prédécesseurs d'un point $\mathbf{p}_i$ pour évaluer le coût. La grille à deux dimension n'est plus suffisante pour calculer tous les chemins.

Au lieu d'utiliser une discrétisation à deux dimensions du domaine $\Omega \subset \mathsf{R}^2$, nous effectuons une discrétisation en trois dimensions du domaine continu $\Omega \times [0, 2\,\pi]$ afin de représenter toutes les positions dans Ω et toutes les orientations possibles (Figure 4.15). Nous discrétisons le domaine angulaire $[0, 2\,\pi]$ en m intervalles. Ainsi, notre approche consiste à calculer le plus court chemin discret anisotrope sur une $n^2\,m$ grille tridimensionnelle de points orientés, noté $\mathbf{p}_{ija}$, $(i, j) \in [0, n]^2$, $a \in [0, m-1]$. Les masques de segments $\mathcal{M}$ sont généralisés pour étendre les masques existant, noté $\mathcal{E}$, tel que $\mathcal{E}(\mathbf{p}_{ija})$ se rapporte à l'ensemble des points $\mathcal{M}(\mathbf{p}_{ij})$ dupliqués pour tous $a \in [0, m-1]$.

Cette technique combinée aux fonctions de coût permet de contrôler la courbure pour générer des routes plus réalistes, au détriment de calculs plus coûteux (Figure 4.16). Le Tableau 4.3 rapporte les statistiques de temps d'exécution en utilisant l'orientation

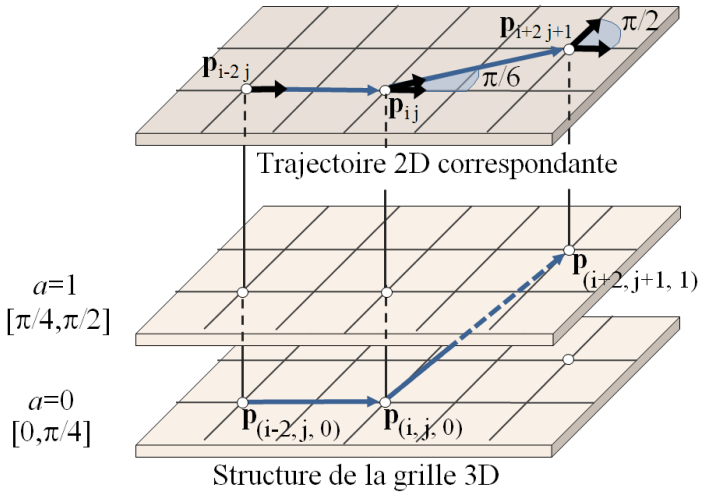


FIGURE 4.15 – Représentation discrète du domaine $\Omega \times [0, 2\,\pi]$ avec $m = 8$, seulement les sous-domaines $\Omega \times \{0\}$ et $\Omega \times \{1\}$ ont été représentés pour plus de clarté.

avec différentes valeurs m de discrétisation de l'angle. Pour obtenir toutes les variations de courbures, la discrétisation de l'angle doit être supérieure ou égale à celle des masques soit $m \geq \frac{180}{\alpha}$.

m	Coût	Temps
0	113 857	0.56
1	97 077	0.72
2	97 023	1.42
3	97 010	2.09

m	Coût	Temps
4	97 000	2.75
5	96 996	3.41
10	96 840	8.07
20	96 840	16.56

TABLEAU 4.3 – La courbure contraint le plus court chemin : coût (unité arbitraire), temps (secondes) et la longueur de la route (mètres).

Comme nous pouvons observer sur le tableau 4.3, le temps d'exécution est proportionnel aux nombres de nœuds à explorer. Le nombre de nœuds est un facteur de m. En pratique, nous utilisons une discrétisation angulaire m avec $m = 8$.

4.3.3 Tunnels et ponts

Les tunnels et les ponts peuvent être traités avec une méthode similaire en utilisant une extension du masque de segments. Le concept principal est d'examiner les ponts et les tunnels en tant que long segment reliant deux points éloignés de la grille.

Sans perte de généralité, considérons d'abord les ponts. Comme pour les routes en surface, nous introduisons des masques de ponts génériques, notés $\mathcal{A}$, représentant les

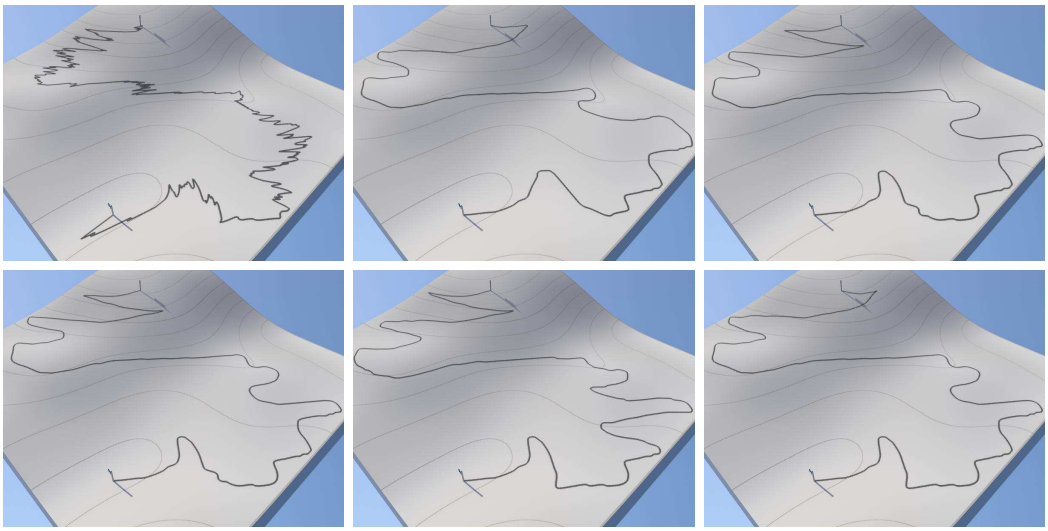


FIGURE 4.16 – Influence de la discrétisation de la direction avec une taille des masques $\mathcal{M}_4$. Les trajectoires obtenues sont respectivement calculées sans courbure et avec courbure avec une discrétisation de l'angle $m = 1, 2, 3, 10, 20$.

connections entre deux points de la grille qui doivent être traités comme des ponts par l'algorithme du plus court chemin (Figure 4.17).

Ces segments ont une distance minimale et maximale, noté r_i et r_e. Ces distances correspondent à la longueur minimale et maximale qu'un type donné de ponts ou de tunnels peuvent avoir. Nous considérons que les ponts de faibles longueurs sont définis par les masques de segment classique, par exemple pour traverser de petites rivières. Par conséquent, nous définissons $\mathcal{A}(\mathbf{p}_{ij})$ tel que :

$$\mathcal{A}(\mathbf{p}_{ij}) = \{\mathbf{q} \neq \mathbf{p}_{ij} | \, r_i \leq |\mathbf{p}_{ij} - \mathbf{q}| \leq r_e\}$$

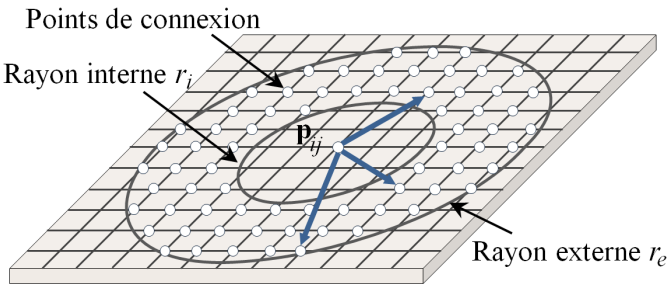


FIGURE 4.17 – Masques de segments de ponts $\mathcal{A}(\mathbf{p}_{ij})$. Seuls trois segments ont été tracés pour plus de clarté.

L'étape 3 de l'algorithme de Dijsktra est modifiée ainsi. Lors du traitement d'un point de la grille $\mathbf{p}_{ij}$, nous calculons et actualisons la valeur des points de la grille dans les deux ensembles $\mathbf{q} \in \mathcal{M}(\mathbf{p}_{ij})$ et $\mathbf{q} \in \mathcal{A}(\mathbf{p}_{ij})$.

3. Pour chaque point $\mathbf{q} \in \mathcal{M}(\mathbf{p}_{ij})$, nous évaluons le coût du chemin en surface $c(\mathbf{p}_{ij}, \mathbf{q})$, et pour tous les points $\mathbf{q} \in \mathcal{A}(\mathbf{p}_{ij})$, nous évaluons le coût du pont $c(\mathbf{p}_{ij}, \mathbf{q})$. Si $c(\mathbf{p}_{ij}, \mathbf{q}) < C(\mathbf{q})$ alors le prédécesseur de $\mathbf{q}$ est $\mathbf{p}_{ij}$.

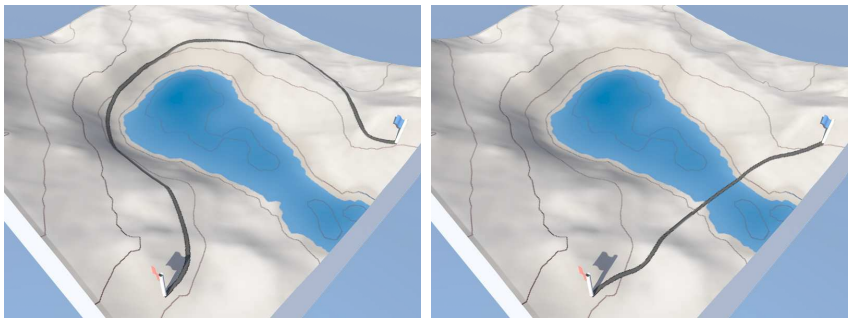


FIGURE 4.18 – L'image de gauche montre une longue route sans pont, alors que l'image de droite montre un chemin plus court obtenu par la création d'un pont.

La Figure 4.18 représente deux routes différentes générées avec et sans pont. Les statistiques correspondantes (Tableau 4.4) démontrent que les ponts permettent de réduire efficacement le coût de la trajectoire, au détriment d'un calcul plus long.

Technique	Coût	Temps	Longueur
Ponts autorisés	45 974	47.8	1 067
Route seule	65 778	0.8	1 773

TABLEAU 4.4 – Statistiques pour deux trajectoires différentes : coût (unité arbitraire), temps (secondes) et longueur de la route (mètres).

Les tunnels sont définis de la même manière, les masques correspondants pour les tunnels sont appelés $\mathcal{T}$.

4.3.4 Échantillonnage stochastique

La détection de tous les tunnels et les ponts de la scène implique d'évaluer la fonction de coût correspondante pour un nombre élevé de segments. Nous rappelons que la complexité d'un algorithme comme Dijkstra mis en œuvre en utilisant un tas de Fibonacci est en $O(a\, n \ln n)$ où a désigne le nombre d'arcs et n se rapporte au nombre de nœuds. Bien que la complexité soit linéaire par rapport au nombre d'arcs, le nombre d'arcs augmente de façon quadratique lorsque le rayon extérieur de la zone de recherche r_e augmente.

Nous proposons d'utiliser une technique d'échantillonnage stochastique pour accélérer les calculs. Au lieu d'évaluer les fonctions de coût des tunnels et des ponts pour l'ensemble des points $\mathcal{A}(\mathbf{p}_{ij})$ à chaque étape de l'algorithme, nous réalisons ces calculs

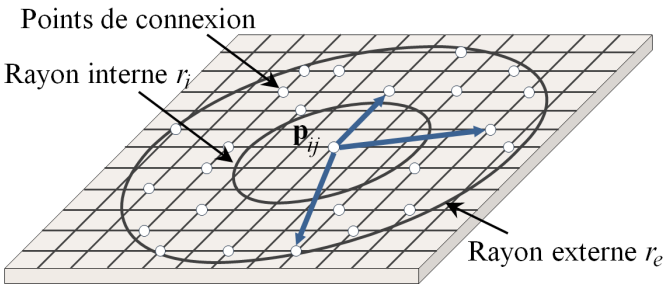


FIGURE 4.19 – Masques de tunnel stochastiques $\mathcal{S}(\mathbf{p}_{ij})$. Seuls trois segments ont été tracés pour plus de clarté.

sur un sous-ensemble restreint de segments $\mathcal{S}(\mathbf{p}_{ij}) \subset \mathcal{A}(\mathbf{p}_{ij})$. Soit $s = |\mathcal{S}|$ le cardinal de $\mathcal{S}$ et $\rho = |\mathcal{A}|$ le cardinal de $\mathcal{A}$ pour un pont. Les points de la grille sont obtenus en sélectionnant stochastiquement certains points à l'intérieur des disques selon une certaine séquence à chaque itération de l'algorithme (Figure 4.19).

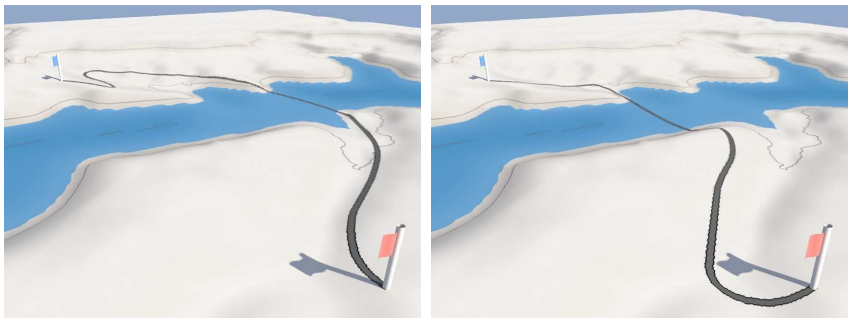


FIGURE 4.20 – Utiliser les masques stochastiques donnent des trajectoires de route légèrement différentes à chaque génération de trajectoire.

Le Tableau 4.5 rapporte des statistiques sur les plus courts chemins illustrés (Figure 4.20) et démontre l'efficacité de la technique d'échantillonnage. La zone du masque du tunnel et du pont a été fixée à $r_i = 50\,\text{m}$ et $r_e = 300\,\text{m}$ avec une grille d'échantillonnage de 10 m. Le nombre de points de la grille visités à chaque itération est égal à $\rho = 2728$. En revanche, le nombre de points visités dans l'approche stochastique varie selon la variable s. L'expérience montre que le temps de calcul est proportionnel au ratio $s/\ \rho$.

Les temps de calcul prouvent que notre algorithme stochastique accélère considérablement les temps de calcul des tunnels et des ponts. Par contre, le coût est légèrement supérieur à la solution optimale.

4.4 Génération procédurale du maillage de routes

Dans cette section, nous présentons notre méthode pour générer les modèles géométriques de routes, de tunnels et de ponts à partir du plus court chemin discret. Notre

s	%Arcs	Coût	Temps	%Coût
0	0	87 096	0.7	100
10	0.3	55 973	0.3	48
25	1	43 918	0.4	27
50	2	44 782	0.6	29
100	4	38 872	1.0	19
250	8	31 583	1.8	7
500	17	30 835	3.4	6
1 000	35	31 038	6.4	7
2 000	71	29 212	12.3	3
ρ	100	27 136	14.2	0

TABLEAU 4.5 – Statistiques pour le masque de segments : coût (unité arbitraire), temps (secondes), pourcentage d'arcs en fonction de ρ et pourcentage du coût en fonction du coût obtenu sans pont.

méthode se décompose en trois étapes.

1. Nous convertissons le plus court chemin discret en un ensemble d'arcs de clothoïdes pour obtenir une trajectoire de route lisse et réaliste.
2. Nous segmentons la trajectoire pour identifier les section de routes de tunnels et de ponts.
3. Nous générons le route en effectuant des opérations de modification de relief et en générant le maillage de la route.

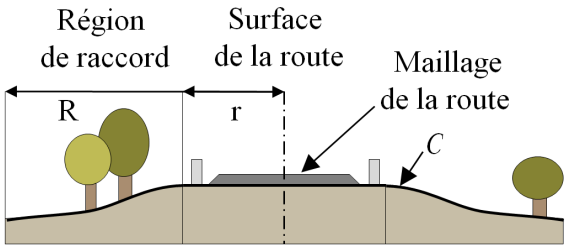


FIGURE 4.21 – Coupe transversale de notre modèle de route de surface.

Les routes, tunnels et ponts sont implémentés comme des modèles géométriques procéduraux. Cette technique nous permet d'adapter la géométrie aux contraintes de la trajectoire. Les routes sont caractérisées par une courbe de profil $\mathcal{C}$ paramétrée par la largeur r de l'asphalte et la largeur R d'une région de mélange (Figure 4.21). La région de mélange est utilisée pour caractériser la forme de l'excavation et du remblai le long de la trajectoire de la route.

4.4.1 Conversion en clothoïdes

La méthode du plus court chemin définit un chemin discret reliant les nœuds du graphe de recherche. Ce chemin discret ne permet pas de construire la route directement. En effet, la trajectoire de la route doit être lisse et respecter des propriétés de courbure afin qu'elle ne soit pas dangereuse. Pour cela nous proposons de transformer la trajectoire discrète obtenue en un ensemble de clothoïdes. Les clothoïdes sont des courbes dont la courbure varie linéairement en fonction d'une longueur d'arc. Ces courbes sont également connues sous le nom de spirale d'Euler, de spirale de cornu ou d'arcs linéaires. Ces courbes sont effectivement utilisées en génie civil pour construire les routes. L'équation de la courbe est la suivante :

$$p(\mathbf{t}) = (x(\mathbf{t}), y(\mathbf{t})) \qquad \begin{cases} x(\mathbf{t}) = k \int_0^{\mathbf{t}} cos(\mathbf{u}^2) \mathrm{d}\mathbf{u} \\ y(\mathbf{t}) = k \int_0^{\mathbf{t}} sin(\mathbf{u}^2) \mathrm{d}\mathbf{u} \end{cases}$$

Nous utilisons la méthode de McRae (MS09b) pour générer nos courbes de clothoïdes. McRae a proposé une méthode pour créer des courbes de clothoïdes à partir d'un ensemble de traits entré par un utilisateur. L'algorithme se déroule en deux étapes :

1. Nous cherchons tout d'abord une approximation de la courbe discrète par un ensemble de morceaux dont la courbure varie linéairement. Cette approximation est définie comme une fonction contrôlant le compromis entre l'erreur de la génération et le nombre de courbes utilisées. Les segments de courbes sont composées de lignes, d'arcs circulaires et de segment de courbes de clothoïdes s'assemblant avec une continuité de classe G^2.
2. Nous déterminons une trajectoire unique à partir de l'ensemble des pièces obtenues. Le mélange est effectué par un mélange linéaire des pièces adjacentes.

Le rayon r de l'arc est défini pour être supérieur à la largeur de la route l de façon à ne pas obtenir d'intersection sur le maillage de la route lors de la création de lacets en montagne. Ce processus permet d'obtenir une trajectoire de route réaliste respectant les normes utilisées en génie civil.

4.4.2 Segmentation de la trajectoire

Rappelons que notre algorithme du plus court chemin anisotrope discret génère une trajectoire définie par $\rho^* = \{\mathbf{p}_k\}_{k\in[0,N]}$ où les points $\mathbf{p}_k$ sont situés sur les points de la grille $\mathbf{p}_{ij}$ discrétisant le domaine de recherche Ω. Nous créons une clothoïde par morceaux, noté Γ, à partir de points de contrôle $\mathbf{p}_k$. Cela nous permet de générer des trajectoires lisses et réalistes de routes.

Du fait du processus de lissage, certaines parties de la courbe Γ correspondant aux segments de chemin de surface peuvent se situer légèrement à l'intérieur ou au-dessus du

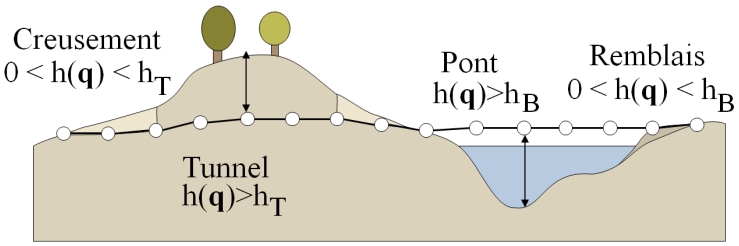


FIGURE 4.22 – La segmentation de la trajectoire de la route et les modifications du terrain correspondant.

terrain. Par conséquent, nous effectuons une segmentation de la trajectoire en fonction de l'élévation du terrain afin d'identifier les parties qui doivent être générées en utilisant les modèles génériques de routes, tunnels ou ponts (Figure 4.22).

La segmentation est réalisée par un échantillonnage uniforme de Γ afin de générer une courbe linéaire par morceaux notée $\gamma^* = \{\mathbf{q}_k\}$, $k \in [0, N]$. Pour chaque point de la courbe discrète, nous calculons $\delta\mathbf{q}_k$ comme étant la différence entre la hauteur du point $\mathbf{q}_k$ et la hauteur de sa projection sur la surface du terrain. Soit $h_{\mathcal{T}}$ la valeur de hauteur minimum pour la création d'un tunnel, et $h_{\mathcal{B}}$ la valeur de la hauteur minimum pour la création d'un pont. La segmentation est réalisée comme suit : si $h(\mathbf{q}_k) < h_{\mathcal{T}}$: marquer $\mathbf{q}_k$ comme point de tunnel, si $h_{\mathcal{T}} < h(\mathbf{q}_k) < h_{\mathcal{B}}$: marquer $\mathbf{q}_k$ comme point de route, sinon marquer $\mathbf{q}_k$ comme point de pont.

À la fin de ce processus, la trajectoire γ^* est discrétisée à nouveau en une courbe polygonale à partir de clothoïdes et chaque point $\mathbf{q}_k$ est marqué en tant que route, tunnel ou pont sans ambiguïté.

4.4.3 Génération de la route

Rappelons que R représente la largeur du modèle de route. Nous définissons le voisinage de la courbe γ^* tel que :

$$\Omega_\Gamma = \{\mathbf{p} \in \Omega \,|\, d(\mathbf{p}, \gamma) < R\}$$

La génération de la route s'effectue en deux étapes. Tout d'abord, nous enlevons la végétation existante dans les environs de la route en supprimant les instances d'arbres qui se trouvent sur Ω_Γ. Le terrain est également modifié dans le voisinage de la trajectoire comme suit (BN08). Premièrement, nous effectuons un raffinement adaptatif du maillage du terrain dans la région de Ω_Γ. Pour chaque sommet $\mathbf{v}$ du maillage raffiné, nous calculons la distance $d(\mathbf{v}, \gamma)$ à la trajectoire. L'excavation et le remblai sont effectués comme suit :

1. Si $d(\mathbf{v}, \gamma) < r$, fixer la hauteur du sommet $\mathbf{v}$ à la hauteur du point correspondant le plus proche sur la courbe.
2. Si $r < d(\mathbf{v}, \gamma) < R$, calculer la hauteur du sommet par interpolation de la courbe de profil de la route $\mathcal{C}(d(\mathbf{v}, \Gamma))$ avec la hauteur du terrain.

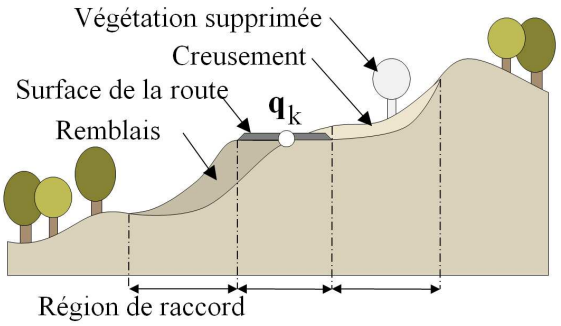


FIGURE 4.23 – Coupe représentant les modifications de terrain effectuées après le processus de segmentation de la route.

Pour finir, nous générons les maillages des routes, des tunnels et des ponts à partir de leur définition paramétrée et de la trajectoire discrète segmentée permettant ainsi une adaptation automatique des maillages aux caractéristiques du terrain.

4.5 Résultats et discussion

Notre méthode de génération procédurale de routes a été implémentée dans une plateforme intégrée à Arches et développée en C++. Nous avons appliqué notre méthode pour créer les différentes images présentées dans ce document. Les rendus ont été effectués en utilisant Mental Ray sur les maillages texturés et générés par notre méthode.

4.5.1 Réalisme

Notre méthode crée des trajectoires réalistes et produit de meilleurs résultats que les méthodes d'édition et de sketching actuelles (MS09b) à la fois en terme d'efficacité que de qualité. La raison principale est que notre algorithme génère une trajectoire optimale qui satisfait une combinaison de contraintes de coûts réalistes. En particulier, la combinaison des fonctions de coût de pente et de courbure nous permet de créer automatiquement des routes de montagne très réalistes (Figure 4.24). Notre algorithme trouve également les endroits les mieux adaptés pour créer des ponts et des tunnels afin d'éviter de longs détours (Figure 4.26).

Nous avons comparé une trajectoire de route réelle avec une trajectoire générée par notre méthode (Figure 4.25). La génération de la trajectoire a été effectuée uniquement sur les caractéristiques du relief du terrain. Les trajectoires obtenues ne sont pas exactement équivalentes aux routes réelles par contre elles reproduisent les mêmes caractéristiques comme des lacets ou des virages mais avec quelques variantes. Des trajectoires totalement différentes peuvent être obtenues tout en donnant des solutions alternatives réalistes. Plusieurs paramètres peuvent influer sur la différence de trajec-

FIGURE 4.24 – Une route de montagne traversant un fleuve.

toire comme l'utilisation de la grille discrète ou encore le manque d'information sur les caractéristiques de la scène (forêts, champs, villages, ...).

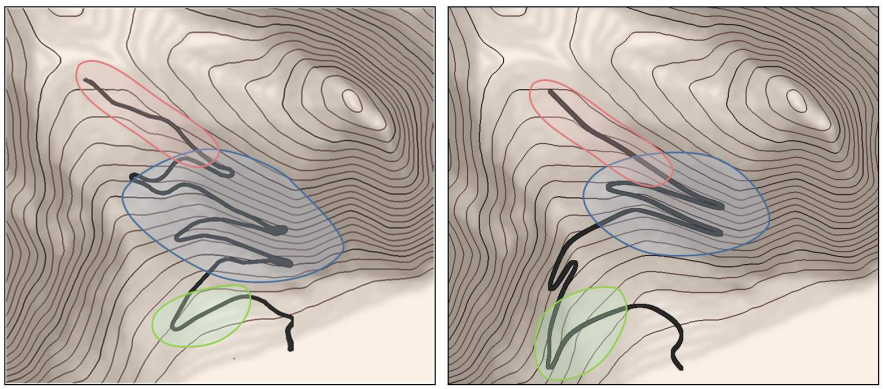


FIGURE 4.25 – Comparaison entre une route réelle à gauche et une route générée à droite. La trajectoire reproduit quelques caractéristiques de la route réelle : le virage de montée (vert), les lacets au milieu de la pente (bleu) et la trajectoire linéaire à l'arrivée (rouge).

4.5.2 Contrôle

Une fonctionnalité très intéressante et puissante de notre approche est sa simplicité d'utilisation et son contrôle. Le processus de génération de route peut être contrôlé de façon très efficace en définissant peu de fonctions de transfert (pente, courbure, végétation, hauteur d'eau). Pour ajouter du contrôle, l'utilisateur peut ajouter simplement de nouvelles fonctions de transfert pour prendre en compte de nouvelles caractéristiques comme la présence de champs, d'espèces protégées ou encore de zones urbaines. Dans notre système, les courbes de ces fonctions sont modifiées de manière interactive. Un exemple de contrôle de la trajectoire est illustrée dans la Figure 4.27. Il suffit de modifier les valeurs des fonctions de transfert pour donner plus d'influence à une caractéristique ou à une autre. Par exemple, pour éviter de déboiser trop de forêt, il faut lui affecter un fort poids.

FIGURE 4.26 – Exemples de routes, ponts et tunnels générés par notre système.

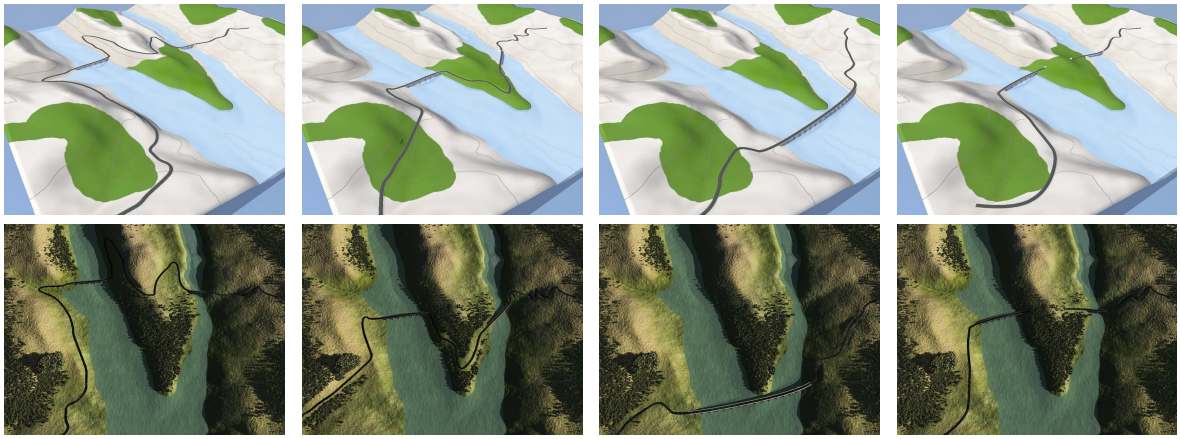


FIGURE 4.27 – Influence des fonctions de coût de pente, d'eau et de forêt. La première image a un fort coût de traversée d'eau et de forêt, la deuxième à un fort coût de traversée d'eau mais faible de forêt, la troisième a un faible coût de traversée d'eau et de forêt et la dernière image a une forte valeur de pente et une forte valeur de forêt.

4.5.3 Temps de calcul

Le temps de génération de la trajectoire varie en fonction du nombre de nœuds à visiter. Le nombre de nœuds à visiter est fortement lié aux caractéristiques de la scène et à la discrétisation de la grille. L'exploration d'un nœud comprend l'évaluation de l'ensemble des arcs voisins. L'évaluation d'un arc revient à évaluer l'ensemble des fonctions caractéristiques du terrain dans un domaine autour de l'arc selon une précision donnée de la scène. Pour nos exemples, nous avons une distance d'un nœud avec son voisin de $10m$ et nous effectuons une intégration de nos fonctions tous les mètres. Les statistiques d'exploration de 1000 nœuds selon les différentes tailles de k des masques sont définies dans le tableau 4.6. Le temps d'évaluation est proportionnel au nombre de segments à évaluer.

Le nombre de nœuds visité peut être réduit en fonction de l'algorithme de recherche du plus court chemin sur le graphe utilisé. Nous avons implémenté les algorithmes de Dijkstra, A* et évalué ces algorithmes avec une recherche bidirectionnelle. Quelques statistiques sont détaillées dans le tableau 4.7 sur plusieurs scènes.

k	1	2	3	4	5	6	7	8	9	10
n_k	8	16	32	48	80	96	144	176	224	256
Temps	6	15	38	65	130	184	315	433	630	838

TABLEAU 4.6 – Temps d'évaluation de 1000 nœuds pour les différents masques : temps (en milli secondes) et n_k le nombre de segment par masques.

	Unidirectionnel		Bidirectionnel	
Scene	Dijkstra	A*	Dijkstra	A*
Vallée	13 722	11 856	8 527	7 701
Traversée d'un col	45 955	43 877	34 973	33 429
Lacs	34 921	29 153	14 846	8 435

TABLEAU 4.7 – Nombre de nœuds visités pour différentes scènes en fonction des différents algorithmes Dijkstra, A* et en recherche bidirectionnelle.

La première scène représente un relief pour calculer une trajectoire dans une vallée. La deuxième scène représente un relief pour calculer une trajectoire traversant un col. La troisième scène représente un relief contenant plusieurs lacs. Nous pouvons observer que le nombre de nœuds explorés est minimal lorsque nous utilisons l'algorithme A* en bidirectionnel. En fonction des scènes, l'algorithme A* est de 5% à 15% meilleur que l'algorithme de Dijkstra. L'algorithme est beaucoup plus rapide (15%) lors de la recherche d'une trajectoire guidée par des obstacles, par exemple lors de la recherche entre deux points se situant dans une même vallée. L'utilisation de la recherche bidirectionnelle est de 24% à 37% meilleure que la recherche unidirectionnelle. De même, la recherche est optimisée lorsque nous recherchons une trajectoire entre deux obstacles qui réduisent le champ de recherche. Dans un cas extrême où le terrain n'a pas de relief et est composé d'obstacle, l'algorithme A* en unidirectionnel est le meilleur des algorithmes. Dans notre cas, où nous voulons créer des trajectoires sur des scènes complexes, nous utilisons l'algorithme A* en bidirectionnel. L'arbre de recherche pour toutes les scènes et pour chaque algorithme est illustré par la figure 4.28.

4.5.4 Perspectives

Nous avons présenté un modèle de génération de route prenant en compte des caractéristiques complexes pour générer une trajectoire réaliste. Notre modèle permet de générer uniquement une trajectoire d'un point **a** à un point **b**. Dans le cadre de la génération complexe de monde, notre méthode nécessite des travaux complémentaires afin de créer un réseau routier complexe composé de plusieurs routes et contenant des in-

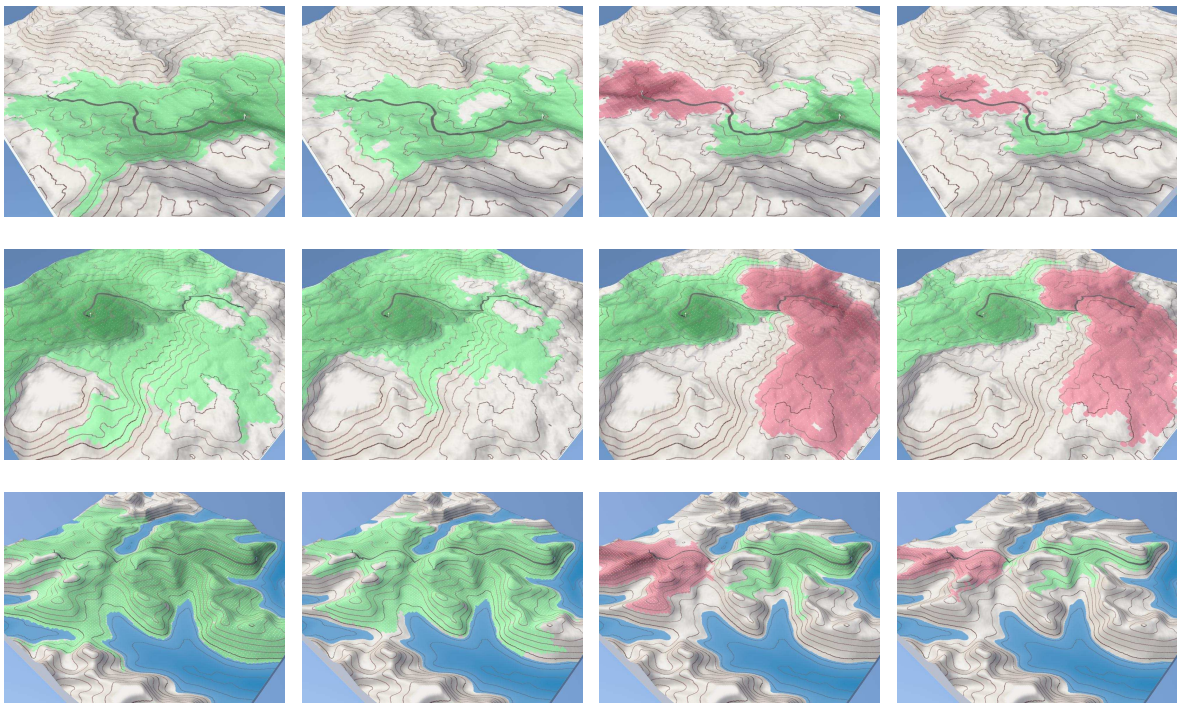


FIGURE 4.28 – Les différents arbres d'exploration selon les algorithmes correspondant : Dijkstra, A* et en Bidirectionnel. Les images représentent successivement les scènes de recherche de plus court chemin dans la Vallée, pour traverser le col et pour contourner plusieurs obstacles.

tersections. L'étude d'un réseau routier complexe apporte de nouvelles problématiques comme la gestion de flux entre les différents points d'intérêts, la minimisation de plusieurs trajectoires afin de les regrouper pour optimiser le coût de création et la création de modèles géométriques complexes définissant les intersections entre les routes comme des ronds points, des croisements ou des ponts.

Conclusion et perspectives

Contributions

Dans cette thèse, nous avons proposé des méthodes pour créer des scènes réalistes avec un très haut niveau de détail. Notre démarche s'appuie sur la combinaison des différentes techniques de modélisation (génération procédurale, simulation, édition) en tirant parti de leurs avantages de chacune pour permettre de contrôler intuitivement la génération et la forme des objets, créer rapidement l'ensemble des détails de la scène et pour obtenir le meilleur réalisme possible. Nous avons montré dans les chapitres 2, 3 et 4 qu'il était possible d'utiliser cette approche pour modéliser des éléments complexes et de natures diverses comme les terrains et les routes. L'approche que nous avons développée permet de résoudre plusieurs problèmes liés à la génération de monde : le réalisme de la scène, la gestion des masses de données, et le contrôle de la génération.

Les modèles de génération procédurale sont associés à des techniques de simulation afin d'obtenir des résultats réalistes. Cette combinaison permet de simplifier la manipulation des données et d'obtenir une scène physiquement plausible. Pour les terrains, nous simulons l'érosion non seulement afin de sculpter les parois des falaises mais également pour obtenir l'ensemble des détails issus de cette érosion comme les matériaux granuleux et les pierres. Nous simulons également la stabilisation des couches de matière afin d'avoir une distribution réaliste de la matière dans la scène. Nous avons proposé un processus de génération de pierres afin d'ajouter automatiquement un nombre important de pierres à la scène. Lors de la génération des pierres, nous simulons l'érosion pour définir des formes variées tout en conservant les contacts entre les pierres de façon à obtenir un empilement visuellement réaliste.

Une difficulté inhérente à la modélisation de scène avec un grand niveau de détail est la grande quantité de géométrie qui doit être produite. Les méthodes de simulation ne permettent pas de gérer cette complexité car les temps de calculs sont trop importants. Par exemple, nous ne pouvons pas simuler le vieillissement d'une route pour définir les craquelures car l'étendue de la route est trop importante. Pour produire cette quantité de géométrie, nous utilisons des modèles procéduraux. Nous utilisons également cette approche pour ajouter les empilements de pierres au terrain. Les techniques de pavages

apériodiques permettent de limiter le coût mémoire de la géométrie tout en permettant de manipuler un très grand nombre d'instance. Les procédures permettent également de générer une quantité de géométrie importante et de grande précision à partir de la trajectoire de la route comme le bitume, les barrières, les ponts et les tunnels à partir de simple plan.

Un élément important de notre approche est le contrôle de la création et de la génération des éléments. De nombreuses techniques totalement automatiques existent mais ne sont pas prévisibles et produisent des scènes non conformes à l'attentes des graphistes. A l'inverse, un utilisateur ne peut pas créer manuellement l'ensemble des détails de la scène car cette solution est fastidieuse, longue et répétitive. Pour résoudre ce problème, nous proposons de contrôler les méthodes de génération procédurales et de simulation par un ensemble de paramètres intuitifs. Pour les terrains, nous proposons des outils spécifiques et intuitifs permettant de les éditer. Ces outils permettent d'éditer directement de grandes zones comme la création de canyon, ou des zones plus locales comme la déformation de la roche. Pour créer des empilements de pierres, l'utilisateur doit simplement définir le volume englobant de l'empilement. Il peut également modifier le type de l'empilement en paramétrant la grosseur et la forme des pierres. La génération de la route s'effectue en fonction de l'ensemble des caractéristiques de la scène qui sont pondérées relativement les unes avec les autres.

Perspectives

Nous n'avons exploré qu'une partie des solutions pour créer des scènes avec un haut niveau de détail. Il reste encore de nombreux et importants verrous scientifiques et techniques à résoudre. Tout d'abord, l'ensemble des approches spécifiques aux terrains, rochers et routes sont encore perfectibles.

Pour les terrains, nous avons proposé un modèle permettant de définir un nombre important de caractéristiques. L'ajout de nouvelles caractéristiques permettrait d'augmenter le réalisme de ces scènes comme l'ajout de feuilles ou de branches. La définition de l'ensemble de ces caractéristiques permet de prendre en compte plus facilement les interactions entre l'ensemble des éléments. Une amélioration de notre modèle serait de simuler un écosystème tridimensionnel prenant en compte l'ensemble de son environnement. L'information volumique de notre modèle améliorerait la précision des ressources avoisinantes (lumière, terre eau) comparés aux modèles existant pour obtenir des résultats plus réalistes.

Pour les empilements de pierres, une amélioration directe serait d'optimiser le coût mémoire et le volume de données qu'occupe actuellement notre représentation afin d'obtenir un modèle plus compact et tout aussi performant. Des travaux plus poussés seraient également nécessaires pour généraliser notre modèle afin de créer des empilements de maillages quelconques, par exemple pour créer des tas de feuilles et de branches.

Une extension du modèle de génération de route serait de créer l'ensemble du réseau

routier reliant les différentes villes entre elles. Ce problème est beaucoup plus complexe car il nécessite d'analyser l'ensemble des trajectoires possibles entre chaque villes et de choisir les routes qui doivent être créées. Ce modèle nécessite également de définir des modèles géométriques plus complexes.

L'enjeu de cette thèse est la création de scènes contenant un ensemble d'éléments variés qui interagissent et cohabitent entre eux. Les méthodes existantes permettent de créer efficacement des forêts, des centre-villes ou des terrains. Par contre, ces méthodes ne parviennent pas à générer les éléments à la frontière de deux domaines. Cette problématique consiste à définir des modèles précis pour obtenir une continuité entre l'ensemble de ces interfaces, par exemple créer la périphérie d'une villes, créer un village de montagne qui s'adapte et s'intègre à leur environnement ou encore créer des caractéristiques plus complexe comme la génération de rivières contenant des cascades.

Bibliographie

[ABVA08] Daniel Aliaga, Bedrich Benes, Carlos A. Vanegas, and Nathan Andrysco. Interactive reconfiguration of urban layouts. *IEEE Computer Graphics and Applications*, 28 :38–47, 2008.

[AD05] M Alsweis and O Deussen. Modeling and visualization of symmetric and asymmetric plant competition. *Eurographics workshop on natural phenomena*, pages 83–88, 2005.

[AD06] Monssef Alsweis and Oliver Deussen. Efficient simulation of vegetation using light and nutrition competition. In *SimVis*, pages 35–48, 2006.

[AK84] Masaki Aono and Tosiyasu Kunii. Botanical tree image generation. *IEEE Comput. Graph. Appl.*, 4(5) :10–34, 1984.

[AMS00] L. Aleksandrov, A. Maheshwari, , and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In *Proceedings of the 32nd ACM symposium on Theory of Computing*, pages 286–295, 2000.

[AMS05] L. Aleksandrov, A. Maheshwari, , and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM*, 52 :25–53, 2005.

[ARB07] Daniel G. Aliaga, Paul A. Rosen, and Daniel R. Bekins. Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4) :786–797, 2007.

[ARP05] B. Lane P. Federl A. Rolland-Lagan A. Runions, M. Fuhrer and P. Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics*, 24(3) :702–711, 2005.

[ASSJ06] Fabricio Anastacio, Mario Costa Sousa, Faramarz Samavati, and Joaquim A. Jorge. Modeling plant structures using concept sketches. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 105–113, 2006.

[AVB08] Daniel Aliaga, Carlos Vanegas, and Bedrich Benes. Interactive example-based urban layout synthesis. *ACM Transaction on Graphics*, 27(5) :1–10, 2008.

[BCS03] Bedrich Benes, Javier Abdul Cordóba, and Juan Miguel Soto. Interacting agents with memory in virtual ecosystems. In *WSCG*, 2003.

[Bel07] Farès Belhadj. Terrain modeling : a constrained fractal model. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 197–204, 2007.

[BF01] Bedrich Benes and Rafael Forsbach. Layered data representation for visual simulation of terrain erosion. In *Proceedings of the 17th Spring conference on Computer graphics*, pages 80–85, 2001.

[BF02] Bedrich Benes and Rafael Forsbach. Visual simulation of hydraulic erosion. In *Journal of WSCG*, volume 10, pages 79–86, 2002.

[BFO+07] Mathew Beardall, Mckay Farley, Darius Ouderkirk, Jeremy Smith, Michael Jones, and Parris Egbert. Goblins by spheroidal weathering. In *Eurographics Workshop on Natural Phenomena*, pages 7–14, 2007.

[BL09] Stefan Bornhofen and Claude Lattaud. Competition and evolution in virtual plant communities : a new modeling approach. *Natural Computing : an international journal*, 8(2) :349–385, 2009.

[Blo85] Jules Bloomenthal. Modeling the mighty maple. *Proceedings of SIGGRAPH*, 19(3) :305–311, 1985.

[BM02] Bedrich Beneš and Erik Uriel Millán. Virtual climbing plants competing for space. In *Proceedings of the Computer Animation*, page 33, 2002.

[BN08] Eric Bruneton and Fabrice Neyret. Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum (Proceedings Eurographics)*, 27(2) :311–320, 2008.

[BPB09] Kevin Boulanger, Sumanta N. Pattanaik, and Kadi Bouatouch. Rendering grass in real time with dynamic lighting. *IEEE Computer Graphics and Applications*, 29(1) :32–41, 2009.

[BPF+03] F. Boudon, P. Prusinkiewicz, P. Federl, C. Godin, and R. Karwowski. Interactive design of bonsai tree models. *Computer Graphics Forum.*, 22(3) :591–599, 2003.

[BS81] R Borchert and N. Slade. Bifurcation ratios and the adaptative geometry of trees. *Botanical gazette*, 142(3) :394–401, 1981.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4) :251–256, 1991.

[BTHB06] Bedřich Beneš, Václav Těšínský, Jan Hornyš, and Sanjiv K. Bhatia. Hydraulic erosion : Research articles. *Comput. Animat. Virtual Worlds*, 17(2) :99–108, 2006.

[BW97] Jules Bloomenthal and Brian Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[CBS96] Benoît Crespin, Carole Blanc, and Christophe Schlick. Implicit sweep objects. *Computer Graphics Forum*, 15(3) :165–174, 1996.

[CD05] R. Cook and T. DeRose. Wavelet noise. *ACM Transaction on Graphics (Proc. SIGGRAPH)*, 24(3) :803–811, 2005.

[CEW+08] G. Chen, Greg Esch, Peter Wonka, P. Muller, and Eugene Zhang. Interactive procedural street modeling. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 103–113, 2008.

[CHZ00] J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold : a world made of drawings. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 83–90, 2000.

[CK95] K. Culik and J. Kari. An aperiodic set of wang cubes. *Journal of Universal Computer Science*, 1(10) :675–686, 1995.

[CLDD09] Marcio Cabral, Sylvain Lefebvre, Carsten Dachsbacher, and George Drettakis. Structure preserving reshape for textured architectural scenes. *Computer Graphics Forum (Proceedings of Eurographics)*, 2 :469–480, 2009.

[CMDH97] Norishige Chiba, Kazunobu Muraoka, Akio Doi, and Junya Hosokawa. Rendering of forest scenery using 3d textures. *Journal of Visualization and Computer Animation*, 8(4) :191–199, 1997.

[CMF98] N. Chiba, K. Muraoka, and K. Fujita. An erosion model based on velocity fields for the visual simulation of mountain scenery. *Journal of Visualization and Computer Animation*, 9(4) :185–194, 1998.

[CSHD03] M. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 22(3) :289–294, 2003.

[Cul96] Karel Culik. An aperiodic set of 13 wang tiles. *Discrete Mathematics*, 160(1-3) :245–251, 1996.

[CWFV09] Jean-François Côté, Jean-Luc Widlowski, Richard A. Fournier, and Michel M. Verstraete. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment*, 113(5) :1067 – 1081, 2009.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry : Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[DGA05] Brett Desbenoit, Eric Galin, and Samir Akkouche. Modeling cracks and fractures. *The Visual Computer*, 21(8-10) :717–726, 2005.

[DHL+98] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proceedings of ACM SIGGRAPH*, pages 275–286, 1998.

[DN04] Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 93–102, 2004.

[EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 341–346, 2001.

[EKG99] H. Ehrig, Engels H. K., and Rozenberg G. *Handbook of Graph Grammars and Computing by Graph Transformation : Applications, Languages and Tools*. World Scientific Publishing Company, 1999.

[EMP+98] David Ebert, Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling : A Procedural Approach*. Academic Press Professional, 1998.

[EMP+02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. 3 edition, 2002.

[FCG99] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Virtual sculpture. In *Eurographics : Short paper*, 1999.

[FCG02] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Resolution adaptive volume sculpting. *Graphical Models*, 63 :459–478, 2002.

[Foc98] C. Focas. The four world cities transport study. 1998.

[GC01] Stéphane Gobron and Norishige Chiba. Simulation of peeling using 3d-surface cellular automata. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, pages 338–347, 2001.

[Gei07] Ryan Geiss. Generating complex procedural terrains using the GPU. In *GPU Gems 3*. Addison-Wesley, 2007.

[Gla98] Andrew Glassner. Aperiodic tiling. *IEEE Comput. Graph. Appl.*, 18(3) :83–90, 1998.

[GM01] Manuel Gamito and F. Kenton Musgrave. Procedural landscapes with overhangs. In *10th Portuguese Computer Graphics Meeting*, pages 33–42, 2001.

[GMB06] Kevin R. Glass, Chantelle Morkel, and Shaun D. Bangay. Duplicating road patterns in south african informal settlements using procedural techniques. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 161–169, 2006.

[GMe09] J. Gain, P. Marais, and W. Straßer. Terrain sketching. In *Proceedings of the symposium on Interactive 3D graphics and games*, pages 31–38, 2009.

[GPMG10] Eric Galin, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin. Procedural Generation of Roads. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2) :429–438, 2010.

[Gre89] N. Greene. Voxel space automata : modeling with stochastic growth processes in voxel space. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 175–184, 1989.

[GS86] Branko Grünbaum and G C Shephard. *Tilings and patterns.* W. H. Freeman & Co., 1986.

[Hau01] Alejo Hausner. Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 573–580, 2001.

[HJO+01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[Hon71] H. Honda. Description of the form of trees by the parameters of the tree-like body : Effects of the branching angle and the branch length on the shape of the tree-like body. pages 331–338, 1971.

[HSS03] K. Hormann, S. Spinello, and P. Schröder. C^1-continuous terrain reconstruction from sparse contours. In *Proceedings of Vision, Modeling, and Visualization 2003*, pages 289–297, 2003.

[IFMC03] Tomoya Ito, Tadahiro Fujimoto, Kazunobu Muraoka, and Norishige Chiba. Modeling rocky scenery taking into account joints. *Computer Graphics International Conference*, 0 :244–247, 2003.

[IOI06a] T. Ijiri, S. Owada, and T. Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. In *Computer Graphics Forum*, pages 617–624, 2006.

[IOI06b] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. The sketch l-system : Global control of tree modeling using free-form strokes. In *Smart Graphics*, pages 138–146, 2006.

[IOOI05] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences : interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3) :720–726, 2005.

[Jin02] Analytical methods for polynomial weighted convolution surfaces with various kernels. *Computers and Graphics*, 26(3) :437 – 447, 2002.

[JV04] Zhanfeng Jia and P. Varaiya. Grid discretization based method for anisotropic shortest path problem over continuous regions. In *IEEE Conference on Decision and Control*, pages 3830–3837, 2004.

[KCODL06] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. *ACM Trans. Graph.*, 25(3) :509–518, 2006.

[KH03] Jongrae Kim and João P. Hespanha. Discrete approximations to continuous shortest-path : Application to minimum-risk path planning for groups of UAVs. In *42nd IEEE Conference on Decision and Control*, volume 2, pages 1734–1740, 2003.

[Lag09] Ares Lagae. *Wang Tiles in Computer Graphics*. Synthesis Lectures on Computer Graphics and Animation. Morgan and Claypool Publishers, 2009.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH*, pages 163–169, 1987.

[LD99] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Comput. Graph. Appl.*, 19(1) :56–65, 1999.

[LD06a] Ares Lagae and Philip Dutrï¿½. Poisson sphere distributions. In *Vision, Modeling, and Visualization*, pages 373–379, 2006.

[LD06b] Ares Lagae and Philip Dutré. An alternative for wang tiles : colored edges versus colored corners. *ACM Transactions on Graphics*, 25(4) :1442–1459, 2006.

[LEQ+07] Aidong Lu, David S. Ebert, Wei Qiao, Martin Kraus, and Benjamin Mora. Volume illustration using wang cubes. *ACM Transanctions on Graphics*, 26(2) :11, 2007.

[LH04] Frank Losasso and Hugues Hoppe. Geometry clipmaps : Terrain rendering using nested regular grids. *ACM Transactions on Graphics*, 23(3) :769–776, 2004.

[Lin74] Aristid Lindenmayer. Adding continuous components to l-systems. In *L Systems*, pages 53–68, 1974.

[LMS99] M. Lanthier, A. Maheshwari, and J.-R. Sack. Shortest anisotropic paths on terrains. In *26th International Colloquium on Automata, Languages and Programming*, volume 1644, pages 524–533. Lecture Notes in Computer Science, 1999.

[LP02] Brendan Lane and Przemyslaw Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface*, pages 69–80, 2002.

[LWW08] Markus Lipp, Peter Wonka, and Michael Wimmer. Interactive visual editing of grammars for procedural architecture. In *ACM Trans. Graph.*, pages 1–10, 2008.

[MDH07] Xing Mei, Philippe Decaudin, and Baogang Hu. Fast hydraulic erosion simulation and visualization on GPU. In *Pacific Graphics*, pages 47–56, 2007.

[Mer07] Paul Merrell. Example-based model synthesis. In *I3D '07 : Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 105–112, 2007.

[MK03] Sebastian Magda and David Kriegman. Fast texture synthesis on arbitrary meshes. In *ACM SIGGRAPH : Sketches & Applications*, page 1. ACM, 2003.

[MKM89] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *Proceedings of SIGGRAPH*, pages 41–50, 1989.

[MM08] Paul Merrell and Dinesh Manocha. Continuous model synthesis. *ACM Trans. Graph.*, 27(5) :1–7, 2008.

[Mou05] David Mould. Image-guided fracture. In *Proceedings of Graphics Interface 2005*, pages 219–226, 2005.

[MP91] J. Mitchell and C. Papadimitriou. The weighted region problem : finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38 :18–73, 1991.

[MP96] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410, 1996.

[MrGrG+10] Nicolas Maréchal, Éric Guérin, Éric Galin, Stéphane Mérillou, and Nicolas Mérillou. Heat Transfer Simulation for Modeling Realistic Winter Sceneries . *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2) :449–458, 2010.

[MS09a] James McCrae and Karan Singh. Sketch-based path design. In *Proceedings of Graphics Interface 2009*, pages 95–102, 2009.

[MS09b] James McCrae and Karan Singh. Sketching piecewise clothoid curves. *Computers and Graphics*, 33(4) :452 – 461, 2009.

[MWH+06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3) :614–623, 2006.

[MZWVG07] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3) :85, 2007.

[Nag98] K. Nagashima. Computer generation of eroded valley and mountain terrains. *The Visual Computer*, 13(9-10) :456–464, 1998.

[NC99] Fabrice Neyret and Marie-Paule Cani. Pattern-based texturing revisited. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 235–242, 1999.

[NFD07] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. In *ACM Transactions on Graphics*, page 88, 2007.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Trans. IECE Japan, Part D*, J68-D(4) :718–725, 1985.

[NWD05] B. Neidhold, M. Wacker, and Olivier Deussen. Interactive physically based fluid and erosion simulation. In *Eurographics Workshop on Natural Phenomena*, pages 25–32, 2005.

[Ols04] Jacob Olsen. Realtime procedural terrain generation. *Department of Mathematics And Computer Science (IMADA) University of Southern Denmark*, 2004.

[OOI06] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH : Courses*, page 18, 2006.

[Opp86] Peter E. Oppenheimer. Real time design and animation of fractal plants and trees. *Proceedings of SIGGRAPH*, 20(4) :55–64, 1986.

[Pal07] Wojtek Palubicki. *Fuzzy Plant Modeling with OpenGL- Novel Approaches in Simulating Phototropism and Environmental Conditions.* 2007.

[PBPS99] Joanna L. Power, A. J. Bernheim Brush, Przemyslaw Prusinkiewicz, and David H. Salesin. Interactive arrangement of botanical l-system models. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 175–182, 1999.

[PBT98] L. C. Polymenakos, D. P. Bertsekas, and J. N. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 43(2) :278–283, 1998.

[PGMG09a] Adrien Peytavie, Eric Galin, Stephane Merillou, and Jerome Grosjean. Arches : a Framework for Modeling Complex Terrains. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, 28 :457–467, 2009.

[PGMG09b] Adrien Peytavie, Eric Galin, Stephane Merillou, and Jerome Grosjean. Procedural Generation of Rock Piles Using Aperiodic Tiling. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, 28(7) :1801–1810, 2009.

[PHH93] Przemyslaw Prusinkiewicz, Mark Hammel, and Mark Hammel. A fractal model of mountains with rivers. In *In proceedings of Graphics Interface*, pages 174–180, 1993.

[PHL+09] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch, and Przemyslaw Prusinkiewicz. Self-organizing tree models for image synthesis. In *ACM Trans. Graph.*, number 3, pages 1–10, 2009.

[PJ95] Joost Van Lawick Van Pabst and Hans Jense. Dynamic terrain generation based on multifractal techniques. *International Workshop on High Performance Computing for Computer Graphics and Visualisation*, 1995.

[PM01] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, 2001.

[PMKL01] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300, 2001.

[RB85] William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Comput. Graph.*, 19(3) :313–322, 1985.

[RLP07] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Eurographics Workshop on Natural Phenomena*, pages 63–70, 2007.

[RME09] Brennan Rusnell, David Mould, and Mark Eramian. Feature-rich distance-based terrain synthesis. *The Visual Computer*, 25(5-7) :573–579, 2009.

[RPP93] P. Roudier, Bernard Peroche, and M. Perrin. Landscapes synthesis achieved through erosion and deposition process simulation. *Computer Graphics Forum*, 12(3) :375–383, 1993.

[RR90] Neil Rowe and Ron Ross. Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects. *IEEE Transactions on Robotics and Automation*, 6(5) :146–152, 1990.

[SBBK08] Ondřej Śtava, Bedřich Beneš, Matthew Brisbin, and Jaroslav Křivánek. Interactive terrain modeling using hydraulic erosion. In *ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 201–210, 2008.

[She99] Andrei Sherstyuk. Kernel functions in convolution surfaces : a comparative analysis. *The Visual Computer*, 15 :15–4, 1999.

[Sip96] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[SS05] S. Stachniak and W. Stuerzlinger. An algorithm for automated fractal terrain deformation. In *In Proceedings of Computer Graphics and Artificial Intelligence*, pages 64–76, 2005.

[Sti75] George Nicholas Stiny. *Pictorial and formal aspects of shape and shape grammars and aesthetic systems.* PhD thesis, 1975.

[Sti80] G. Stiny. Introduction to shape and shape grammars. *Environment and Planning B*, 7(3) :343–351, 1980.

[SYBG02] Jing Sun, Xiaobo Yu, George Baciu, and Mark Green. Template-based generation of road networks for virtual city modeling. In *VRST '02 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 33–40, 2002.

[Tsi95] John Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9) :1528–1538, 1995.

[Ula62] S. Ulam. On some mathematical properties connected with patterns of growth of figures. pages 215–224, 1962.

[WBCG09] Jamie Wither, Frédéric Boudon, Marie-Paule Cani, and Christophe Godin. Structure from silhouettes : a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum*, 28(2) :541–550, 2009.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree (warping, blending and boolean operations in an implicit surface modeling system). *Computer Graphics Forum*, 18(2) :149–158, 1999.

[WI04] N. Watanabe and T. Igarashi. A sketching interface for terrain modeling. In *ACM SIGGRAPH Posters*, page 73, 2004.

[WMWG09] Basil Weber, Pascal Mueller, Peter Wonka, and Markus Gross. Interactive geometric simulation of 4d cities. *Computer Graphics Forum*, pages 481–492, 2009.

[WO96] B. Wyvill and K. Van Overveld. Tiling techniques for implicit skeletal models. *SIGGRAPH Course*, 11 :C1.1-C1.26, 1996.

[Wor96] Steven Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, 1996.

[WP95] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, 1995.

[WWSR03] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Trans. Graph.*, 22(3) :669–677, 2003.

[WY04] Q. Wu and Y. Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(3) :364–367, 2004.

[YNBH09] Qizhi Yu, Fabrice Neyret, Eric Bruneton, and Nicolas Holzschuch. Scalable real-time animation of rivers. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2) :239–248, 2009.

[ZDW+05] Kun Zhou, Peng Du, Lifeng Wang, Yasuyuki Matsushita, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 11(5) :519–528, 2005.

[ZKT05] Zhiyi Zhang, Kouichi Konno, and Yoshimasa Tokuyama. 3d terrain reconstruction based on contours. *Computer Aided Design and Computer Graphics, International Conference on*, 0 :325–330, 2005.

[ZS07] M Nordin Zakaria and Siti Rokhmah Shukri. A sketch-and-spray interface for modeling trees. In *Proceedings of the 8th international symposium on Smart Graphics*, pages 23–35, 2007.

[ZSTR07] Howard Zhou, Jie Sun, Greg Turk, and James Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4) :834–848, 2007.

Publications

Revues internationales avec comité de lecture

A. Peytavie, E. Galin, S. Mérillou, J. Grosjean. Arches : a Framework for Modeling Complex Terrains.*Computer Graphics Forum (Proceedings of Eurographics)*, 28(2), pages 457-467, 2009.

A. Peytavie, E. Galin, S. Mérillou, J. Grosjean. Procedural Generation of Rock Piles using Aperiodic Tiling.*Computer Graphics Forum (Proceedings of Pacific Graphics)*,28 (7), pages 1801-1809, 2009.

E. Galin, A. Peytavie, N. Maréchal, and E. Guérin. Procedural generation of roads. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2), pages 429-438, 2010.

Conférences nationales sans comité de lecture

A. Peytavie, E. Galin, S. Mérillou, J. Grosjean. Modélisation de terrains complexes 3D. In *AFIG'08 : 21e journées de l'Association Française d'Informatique Graphique*, pages 43-52, 2008.

A. Peytavie, N. Maréchal, E. Guérin, and E. Galin. Génération procédurale de routes. In *AFIG'09 : 22e journées de l'Association Française d'Informatique Graphique*, pages 123-132, 2009.

Génération procédurale de Monde

Résumé :

Dans cette thèse, nous abordons le problème de la génération automatique de contenu graphique avec un haut niveau de détails pour la génération de mondes. Dans cette quête du réalisme, les principaux verrous scientifiques et techniques sont : la gestion des masses de données géométriques nécessaires à la création de variétés d'objets naturels, la prise en compte des interactions en les différents objets et le contrôle utilisateur.

Notre approche s'appuie sur un modèle volumique original et unificateur permettant de représenter les différents matériaux d'un terrain. Nous proposons un ensemble d'algorithmes de haut niveau, combinant des étapes de simulation pour assurer la plausibilité physique et le réalisme visuel à des techniques de génération procédurale permettant de gérer efficacement la complexité. Nos algorithmes permettent ainsi de sculpter des terrains complexes avec des grottes ou des surplombs et de générer des routes avec des tunnels et des ponts permettant le franchissement d'obstacles naturels.

Procedural generation of world

Abstract :

In this thesis, we address the problem of automatic generation of graphical content with a high level of detail for the generation of worlds. In this quest for realism, the major scientific and technical challenges include : managing the mass of geometric data needed to create varieties of natural objects, taking into account interactions with the various objects and user control.

Our approach is based on an original volumetric model for representing and unifying the different materials of land. We offer a range of high-level algorithms, combining simulation steps to ensure physical plausibility and visual realism, and procedural generation techniques to effectively manage complexity. Our algorithms are used to sculpt and complex terrain with caves and overhangs and generate roads with tunnels and bridges to cross natural barriers.