Université de Montréal

Simulation du transport de neige

par
Alexandre Jubert

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Juillet, 2018

RÉSUMÉ

Que ce soit pour des scènes synthétiques de films ou de jeux vidéo, le réalisme des environnements passent par l'ajout d'effets visuels communs à nos expériences. La neige fait partie des phénomènes naturels complexes qui rendent leur simulation physique et visuelle d'autant plus difficile à traiter.

En effet, la neige tombe au gré du vent dans les scènes, elle s'entasse et se compacte à des endroits, se brise et retombe, se fait repousser par le vent une fois déposée, se liquéfie ou s'évapore, se glace, accumule des impuretés, etc. Le but de cette maîtrise est de développer un simulateur de transport de neige dans des espaces synthétiques. Ce simulateur se veut à la fois un outil automatique pour enneiger des scènes, mais offrant certains contrôles aux artistes afin de produire des résultats escomptés avec des paramètres simples comme le taux de précipitations de la neige ou la direction du vent.

Nous proposons une méthode de simulation du transport de la neige, comportant :

1. Une phase de précalculs afin d'estimer le vent et les précipitations dans la scène.
2. Un modèle physique de transport de la neige permettant même d'observer l'évolution de la neige dans le temps.
3. Une méthode d'implémentation sur processeur graphique.

Nos résultats montrent l'intérêt des divers phénomènes sur la simulation réaliste du transport de neige.

Mots clés: Infographie, Neige, Transport de Neige, Simulation de Fluides.

ABSTRACT

Whether it is for synthetic scenes in movies or video games, the realism of environments passes by the addition of visual effects common to our experiences. Snow is part of complex natural phenomena that make their physical and visual simulation all the more difficult to treat.

In fact, snow falls in scenes under the action of wind. It piles up and is compressed in some areas, breaks and falls, is repelled by wind once deposited, liquefies or evaporates, freezes, accumulates impurities, etc. The goal of this research is to develop a simulator of snow transport in synthetic spaces. This simulator is designed as an automatic tool to integrate snow in scenes by adding its precipitation, transport by wind, simulation of melting and evaporation, but also to offer some control to artists in order to produce the desired results with easy and understandable settings such as the snow precipitation rate or the wind direction and speed in the scene.

We propose a method for the simulation of snow transport that consists of :

1. A precomputation phase to estimate wind and precipitations in a scene.
2. A physical model of snow transport to capture its evolution through time.
3. A method of implementation on GPU.

Our results demonstrate the need to simulate the various phenomena on the realism of snow distribution.

Keywords: Computer Graphics, Snow, Snow Transport, Fluid Simulation.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

NOTATION

ζ	Epaisseur de la neige (m)
z	Hauteur (m)
θ	Direction du vent (rad)
W	Vitesse du vent ($\mathrm{m\,s^{-1}}$)
β	Angle de la pente du terrain (rad)
ξ	Angle azimutal du terrain (rad)
Ω_c	Influence de la courbure du terrain (sans unité)
Ω_p	Influence de la pente du terrain (sans unité)
ρ_s	Densité de la neige ($\mathrm{kg\,m^{-3}}$)
ρ_{ns}	Densité de la neige fraîche ($\mathrm{kg\,m^{-3}}$)
σ	Dureté de la neige (kPa)
ρ_w	Densité de l'eau ($\mathrm{kg\,m^{-3}}$)
ρ_a	Densité de l'air ($\mathrm{kg\,m^{-3}}$)
P	Taux de précipitations ($\mathrm{m\,s^{-1}}$)
Q_{turb}	Taux de transport par suspension ($\mathrm{kg\,m^{-1}\,s^{-1}}$)
Q_{salt}	Taux de transport par saltation ($\mathrm{kg\,m^{-1}\,s^{-1}}$)
T_{WB}	Température de l'air ($^\circ$ K)
u_*	Force d'érosion ($\mathrm{m\,s^{-1}}$)
u_{*t}	Seuil d'érosion critique ($\mathrm{m\,s^{-1}}$)

(dédicace) À ma famille

REMERCIEMENTS

Je tiens tout d'abord à remercier mes parents, mes frères et mes soeurs pour leur soutien dans cette longue aventure, pour m'avoir fait confiance et avoir cru dans mon rêve de venir au Canada, et pour avoir fait leur possible pour qu'il se réalise.

Merci à mon directeur de recherche Pierre Poulin pour son soutien autant moral que financier, ses conseils ainsi que sa patience, et pour ses nombreuses lectures et corrections de ce mémoire !

Merci aussi aux membres du LIGUM pour ces années inoubliables, notamment Adrien, Arnaud, Bruno, Chaitanya, Cihan, Etienne, Guochao, Jean, Jean-Philippe, Joël, Jonas, Luis, Melino, Nicolas, Olivier, Vincent et Yangyang, ainsi que les visiteurs et anciens membres du laboratoire, notamment Shuhei Kodama, Laurent Belcourt et Kevin Hinz.

Merci à Maël et Éléonore, à Audrey, ainsi qu'aux membres de la Petite Famille, Alexis, Andréa, Aurélien, Aymeric, Benjamin, Emma, Léo et Mathieu pour avoir cru en moi, et pour leur soutien inconditionnel dans mes péripéties !

Je tiens aussi à remercier aussi les professeurs du laboratoire (anciens comme nouveaux !), Bernhard Thomaszewski, Neil Stewart et Derek Nowrouzezahrai.

Finalement je tiens à remercier les membres du jury Michalis Famelis et Michel Boyer.

CHAPITRE 1

INTRODUCTION

La neige est un phénomène naturel extrêmement complexe. Depuis l'apparition de l'infographie, et dans notre cas depuis l'apparition de systèmes d'affichages complexes, la manière de la simuler, de l'afficher et de la manipuler dans la création de jeux vidéo, de films ou dans tout autre outil est une tâche compliquée. La modélisation de scènes enneigées par le biais d'outils de modélisation traditionnels est extrêmement coûteux en termes de temps et d'argent. Elle nécessite d'avoir une équipe d'artistes pour donner une apparence enneigée à une scène, et dans le cas de films d'animation, une simulation physique réaliste et coûteuse est nécessaire pour obtenir des résultats convaincants.

Dans le cas des jeux vidéo et des logiciels interactifs, il est impensable d'effectuer de lourdes simulations physiques, et de représenter chaque flocon de neige individuellement. Nous devons alors trouver des moyens d'approximer les modèles plus réalistes afin de produire une estimation satisfaisante des phénomènes réels, tout en permettant un affichage rapide et efficace, et en réduisant l'impact mémoire le plus possible.

Hormis la synthèse d'image, la neige et son transport ont été étudiés dans plusieurs domaines, notamment en glaciologie et en météohydrologie, où des modèles ont été développés afin d'estimer l'évolution des quantités de neige dans l'espace et dans le temps. Ces modèles que nous introduisons plus en détail dans le chapitre 5, ont permis de simuler avec une précision validée plusieurs années de transport de neige dans une multitude d'environnements différents, que ce soient les toundras de l'Arctique, en passant par des forêts tempérées, jusqu'aux Rocheuses canadiennes, en se basant sur des observations météorologiques sporadiques, et en fonction de plusieurs paramètres : types de végétations, types de sols, températures, couvertures nuagueuses, vents, etc.

L'intérêt de l'apparence de la neige et de son dépôt est important dans les productions audiovisuelles. En effet, de par sa complexité, obtenir une version enneigée d'une scène arbitraire demande un travail considérable aux artistes pour obtenir des scènes réalistes

qui duperont l'attention des spectateurs.

L'étude du transport de la neige a permis d'isoler les sources et les conséquences des mouvements de la neige. Le vent en déplace de grandes quantités par différents mécanismes que nous introduisons par la suite, sans compter la *sublimation* (c'est-à-dire la transformation des flocons de neige en gaz) et la fonte de la neige. Il y a bien entendu une multitude d'autres phénomènes influant les quantités de neige, mais ils sont négligeables ou peu fréquents (comme les avalanches) en comparaison. Certains de ces phénomènes varient en fréquence en fonction de la saison et du type d'environnement (plaines, montagnes). Par exemple, les avalanches transportent des tonnes de neige dans un laps de temps très court, mais ne sont présentes qu'en montagne. De plus, ces phénomènes apparaissent à différentes échelles : des précipitations de neige peuvent tomber sur des zones de plusieurs kilomètres, mais la *saltation* déplace de la neige seulement sur quelques centimètres. Nous développons un modèle de simulation que l'on souhaite général afin de pouvoir l'appliquer sur différents types d'environnements. C'est pour cela que les phénomènes contraints à seulement certains types d'environnements ne sont pas pris en compte dans notre solution. Nous nous concentrons sur les phénomènes les plus courants transportant de la neige.

Dans ce cadre, le transport de la neige s'effectue donc selon différents mécanismes, les principaux étant :

Saltation : Le transport des flocons de neige en contact périodique avec la surface de neige, provoqué par l'érosion de cette surface par la force du vent.

Suspension : Le transport des flocons de neige par le vent qui s'effectue au-dessus de la couche de saltation. Elle est provoquée par les vents généralement très turbulents qui déplacent les flocons sur de plus grandes distances que la saltation.

Sublimation : L'évaporation dans l'atmosphère de la neige, sans passer par l'état liquide. Elle dépend du pourcentage d'humidité de l'atmosphère, de l'exposition au soleil de par son angle et sa force, etc.

Précipitation : Tout simplement les flocons de neige qui tombent directement au sol.

Nous proposons un protocole de dépôt de la neige dans des scènes virtuelles consistant en une première phase de précalcul du vent et des précipitations de la neige associées à celui-ci, ainsi qu'un modèle de transport de la neige réaliste provenant de la glaciologie, validé par des observations sur le terrain. Nous simulons ce modèle avec différentes cartes de hauteurs. L'utilisation d'un tel modèle en infographie est en soit novateur, car la majorité des méthodes de simulation du transport de la neige basées sur des cartes de hauteurs ne suivent aucun modèle physique, limitant de ce fait le réalisme des distributions de neige obtenues. Malgré certaines limitations de notre méthode que nous discutons dans ce document, notamment le manque d'interactivité de notre solution actuelle, nous empêchant par exemple de simuler des avalanches, simuler les traces de pas d'un personnage dans la neige ou l'absence de la simulation des turbulences dans notre champ de vent, nous pouvons obtenir des distributions de neige influencées par les phénomènes physiques décrits précédemment, et ce, en temps interactif grâce à une implémentation sur processeur graphique.

Nous présentons dans un premier temps les contributions qui ont été faites en infographie dans le transport de la neige au chapitre 2. Nous introduisons ensuite comment simuler efficacement un vent dans le chapitre 3 afin d'obtenir des champs de vent qui seront nécessaires pour calculer des cartes de précipitations à partir d'une simulation de particules que nous présentons dans le chapitre 4. Nous introduisons notre modèle de transport de la neige dans le chapitre 5 qui raffine les distributions de neige avec plusieurs phénomènes, et de l'implémentation sur processeur graphique (GPU) au chapitre 6. Nous discutons des résultats obtenus par notre méthode au chapitre 7, et finalement concluons en discutant de notre méthode et en présentant de futures pistes d'investigations dans le chapitre 8.

CHAPITRE 2

ETAT DE L'ART

Dans ce chapitre, nous présentons les travaux antérieurs sur la simulation de la neige en infographie. Les travaux antérieurs spécifiques à la simulation du transport de la neige sont couverts dans le chapitre 5. Nous distinguons trois catégories principales de travaux : les contributions basées sur des simulations de particules, celles basées sur la subdivision de surfaces, et celles basées sur des cartes de hauteurs ou de la rasterisation.

2.1 Méthodes à base de particules

Les méthodes de simulation à base de particules, aussi nommées Lagrangiennes, font partie d'un domaine très étudié en informatique graphique. Depuis 1988 avec la parution de la méthode SPH [27] développée par Monaghan, les simulations à base de particules sont présentes dans une grande variété d'applications. Leurs fonctionnements, malgré l'augmentation de la complexité des effets et des comportements physiques simulés, ainsi que de la complexité de la gestion des ressources computationnelles, suivent globalement la même boucle de simulation : chaque particule représente une ou plusieurs données physiques, et pour un pas de temps donné, les mouvements de chacune d'elles sont résolus. Par la suite une reconstruction d'un maillage ou d'une surface implicite à partir de ces particules est effectuée afin d'obtenir une forme à afficher, ou simplement un volume traitable par la suite. Ce processus est ensuite répété afin d'obtenir des animations réalistes.

Figure 2.1 – Rendu de la méthode de Nishita et al. [30].

Dans le cas de la simulation de neige, Nishita et al. [30] sont les premiers à proposer une méthode de dépôt de neige dans des scènes arbitraires. Leur méthode consiste à déposer sur toutes les surfaces de la scène des *Metaballs*, qui sont des particules ayant un rayon d'influence, une densité et une position. De manière itérative, ces particules sont déposées par couche dans la scène. Chacune de ces particules possédant un champ d'influence, une surface est générée par la méthode de *Marching Cubes* [23]. Nishita et al. proposent aussi un modèle d'éclairage de la neige prenant en compte la translumines-cence. Nishita et al. [30] sont ainsi les premiers à proposer un modèle de simulation et d'éclairage de la neige, mais leur méthode possède plusieurs défauts, notamment l'absence d'influence du vent, de la gravité (les *Metaballs* étant simplement déposées sur les surfaces) ou de tout autre effet dynamique sur les particules, ainsi que l'absence de dynamique entre les particules.

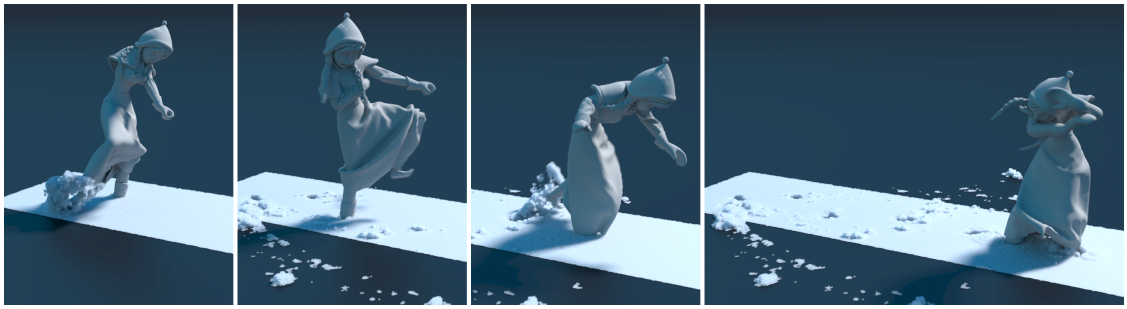


Figure 2.2 – Simulation avec la méthode de Stomakhin et al. [42].

Afin de pallier ces problèmes, Stomakhin et al. [42] proposent un modèle élasto-plastique pour simuler des amas de neige jusqu'au plus fin flocon de neige. Leur modèle prend en compte les particularités de la physique des flocons de neige, notamment les différences drastiques en fonction du type de neige. Cela devient important, notamment pour la neige dense et très humide, quant à son comportement lorsqu'elle se sépare, s'agglutine, tombe, etc. Pour ce faire, Stomakhin et al. [42] utilisent une méthode numérique alliant les deux points de vue utilisés en simulation de fluides : Lagrangien et Eulérien. En conservant les deux points de vue, Stomakhin et al. peuvent à la fois simuler les flocons de neige de manière individuelle ou en groupe, calculer le volume de chaque particule et simuler les fractures du point de vue Lagrangien. Ils peuvent efficacement mettre à jour les positions et les vélocités sur la grille, en prenant en compte les collisions avec les objets du point de vue Eulérien. Chacune des techniques respectives étant particulièrement adaptées à ces problèmes, Stomakhin et al. proposent un solveur stable et efficace, qui permet de simuler les dynamiques complexes de la neige. Par contre, dus à la complexité des effets qu'ils simulent, les temps de calcul et le coût mémoire, leur solution est inenvisageable pour des productions interactives. Par exemple, la simulation illustrée à la figure 2.2 a demandé 15.7 minutes de simulation par pas de temps en 2013.

Dagenais et al. [6] partent du constat que simuler tous les flocons de neige est beaucoup trop coûteux en temps et en mémoire, et que dans la majorité des productions visuelles, seulement la neige autour des personnages a besoin d'être animée, et que la majorité de la neige autour reste statique. Pour ce faire, ils proposent de découper la

simulation en trois parties : une couche de base statique, la simulation de la brume (les fines particules de neige en suspension) et la simulation de particules. Lors d'une itération de leur algorithme, ils calculent les collisions entre les objets dynamiques de la scène avec la couche de neige déposée et, en cas de collision et de pénétration dans la neige par un objet, ils génèrent des particules et les déplacent en fonction de la force de l'objet rentrant en collision, afin de laisser une empreinte dans la couche de neige. Pour éviter de devoir simuler les particules les plus fines, ils proposent de représenter la brume de neige par une simulation de fluide Eulérienne afin de prendre en compte la turbulence du mouvement de ces particules dans l'air, et d'obtenir un effet visuel intéressant. La méthode de Dagenais et al. [6] permet des résultats visuels réalistes là où cela est nécessaire, sans devoir simuler l'entièreté de la neige, tout en restant relativement efficace en comparaison à Stomakhin et al. [42]. Par exemple, les temps de simulation par pas de temps étant de l'ordre de quelques secondes en 2016, avec moitié moins de particules dans la majorité des cas.

2.2 Méthodes à base de carte de hauteurs

Depuis les débuts de l'infographie, la discrétisation de signaux continus est un outil indispensable pour pallier les limitations techniques afin de représenter n'importe quel type de données. De cette manière, les cartes de hauteurs représentent simplement la variation d'une quantité de matière sous forme de couche d'épaisseur variant dans l'espace. Cela implique des défauts inhérents à la discrétisation, notamment des problèmes d'aliassage et de précision, mais la théorie est que si l'on augmente de plus en plus la résolution de la carte, le résultat se rapprochera du signal continu d'origine. L'avantage principal d'utiliser des cartes de hauteurs est que les processeurs graphiques sont très efficaces à travailler avec des cartes en parallèle, ce qui donne des algorithmes qui peuvent produire des résultats satisfaisants dans un temps très contraint, par exemple en temps réel pour un jeu vidéo.

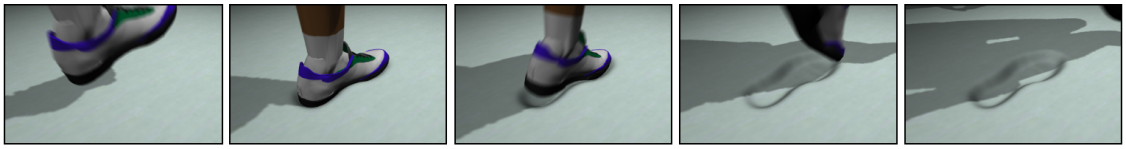


Figure 2.3 – Simulation avec la méthode de Sumner et al. [43].

Sumner et al. [43] sont les premiers à proposer une méthode de simulation pour appliquer des empreintes d'objets dynamiques dans du sable, de la boue et de la neige. Pour cela, ils discrétisent la surface du terrain de leur scène en une grille 2D uniforme afin d'obtenir une carte de hauteurs. Pour détecter une collision, ils lancent un rayon à partir du centre de chaque cellule de la grille jusqu'en haut de la colonne. S'ils obtiennent une collision, ils stockent l'information dans une carte de contours afin de déterminer la forme de l'empreinte. Par la suite, ils calculent le taux de compression du sol en fonction du type de surface, et répartissent la matière déplacée dans les cellules voisines. Finalement, une passe d'érosion est appliquée afin de diffuser progressivement la matière et de simuler par exemple le sable qui glisse vers l'intérieur après avoir été déplacé. De plus, des particules sont projetées à partir de la zone de contact afin de représenter la matière qui s'envole après collision. Une fois qu'elles retombent, cette matière est ajoutée aux colonnes dans lesquelles elles atterrissent. La méthode de Sumner et al. a l'avantage d'être facilement parallélisable, et d'avoir un faible coût en mémoire pour stocker la grille. Néanmoins, leur solution est empirique et les paramètres n'ont pas de représentation physique à proprement parler, ce qui oblige un artiste à changer itérativement les différents paramètres afin d'obtenir le résultat visuel souhaité. Nous pouvons aussi faire un parallèle avec la méthode de Dagenais et al. [6], qui au lieu d'utiliser une carte de hauteurs, utilise une surface comme couche de base, et applique des opérateurs booléens pour l'extraction de surface.

L'une des premières applications au rendu temps réel d'un algorithme d'accumulation de neige est présentée par Ohlsson et Seipel [31]. Leur contribution se scinde en deux parties : une fonction de prédiction d'accumulation de la neige, et une fonction de détermination de l'apparence de la neige. Leur implémentation de la fonction de prédic-

tion d'accumulation consiste à calculer une carte de distances depuis le ciel avec une projection orthographique (le plan de projection est parallèle au sol), afin de déterminer les zones qui sont occultées par des objets ou non. Lors du calcul de l'image finale, Ohlsson et Seipel vérifient que la surface à afficher n'est pas occultée dans la carte de distances. Si elle ne l'est pas, de la neige est affichée en utilisant leur fonction d'apparence. De plus, chaque sommet des surfaces de la scène est déplacé vers le haut en fonction de la quantité de neige, déplacement modulé par le facteur d'occultation. Cette façon empirique de procéder reste néanmoins intéressante, car elle permet de traiter de façon robuste n'importe quelle surface dans une scène, et n'est dépendante que de la résolution de la carte de distances de la scène. Par contre, elle ne permet pas d'accumuler la neige progressivement dans le temps, et ne prend pas en compte l'influence du soleil. De plus, l'occultation donne une information booléenne, qui empêche tout dégradé de la quantité de neige sous une branche d'arbre par exemple. Elle prévient aussi toute tentative d'obtenir de la neige qui se propage à l'intérieur d'un bâtiment par une porte ouverte, la carte de distances ne permettant pas de calculer cette propagation dans les zones occultées.

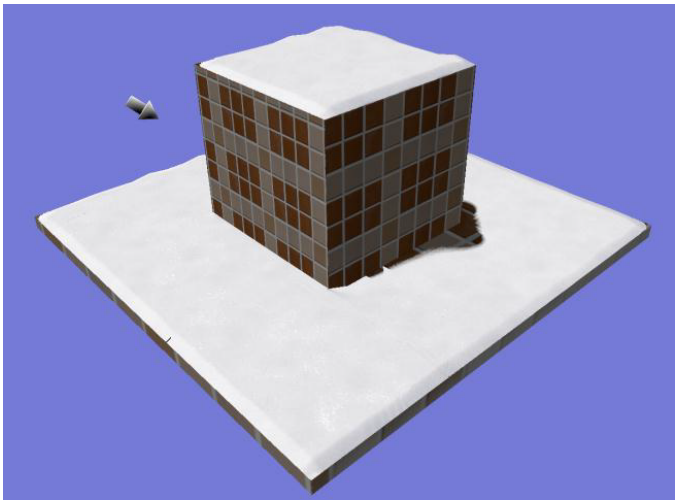


Figure 2.4 – Simulation avec la méthode de Moriya et Takahashi [28].

Afin de pallier ces défauts, Moriya et Takahashi [28] introduisent une approche qui prend en compte la fonte de la neige par le soleil. Leur méthode paramétrise automati-

quement les objets polygonaux de leur scène, et génère des coordonnées UVs de texture automatiquement. Ensuite, ils précalculent la luminance pour chaque texel (le plus petit élément d'une texture, issu de l'anglais *texture element*) de ces *lightmaps* à l'aide de la méthode PRT [39] (*Precomputed Radiative Transfer*), et conservent dans un même temps des cartes d'arêtes *(edge maps)*, qui stockent la distance aux arêtes des objets afin d'éviter des discontinuités sur les surfaces de neige générées. Dans la phase de simulation, la quantité de neige à ajouter à chaque texel des surfaces qui peuvent recevoir de la neige est déterminée en fonction de la luminance précalculée avec PRT [39] (moins de neige sera ajoutée en fonction de la quantité de luminance calculée), de la normale de la surface dans le texel, ainsi que la distance au bord. De cette manière, Moriya et Takahashi [28] sont capables d'animer l'accumulation de la neige dans le temps, en corrigeant le problème de Ohlsson et Seipel [31] tout en possédant une notion d'occultation plus précise. Néanmoins, l'influence du vent n'est pas non plus prise en compte.

Figure 2.5 – Simulation avec la méthode de Reynolds et al. [35].

Reynolds et al. [35] corrigent ce problème en proposant une méthode similaire à celle de Ohlsson et Seipel [31], se basant sur des projections orthographiques. L'ajout de l'influence du vent est fait en changeant la direction de la projection orthographique pour générer la carte d'occultations, ce qui leur permet d'influencer la direction dans

laquelle ils accumulent la neige. De plus, Reynolds et al. [35] introduisent des cartes d'accumulations pour chaque surface de la scène, une paramétrisation des objets de la scène en textures qui permet de stocker la quantité de neige tombée dans chacun des texels. Cette indirection découple la résolution de la carte d'occultations de celle des objets, permettant de stocker les quantités de neige par objet. De plus, cela leur permet d'avoir des objets dynamiques dans la scène, la neige étant stockée par objet. Une heuristique de stabilité est aussi proposée afin de redistribuer la neige en cas de mouvement dans la scène, par exemple un objet qui pivote pour faire tomber la neige de sa surface. Néanmoins, la neige n'est pas redistribuée ; elle est simplement retirée de la surface. Finalement, la neige est affichée sur les surfaces en les subdivisant et en déplaçant chaque sommet par la quantité de neige accumulée.

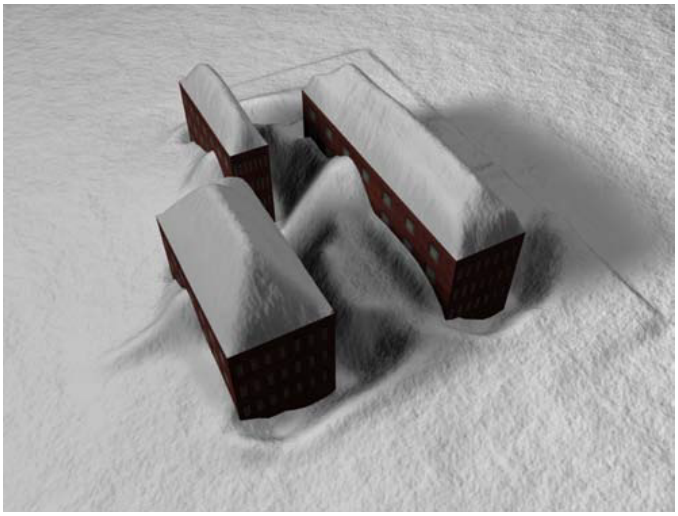


Figure 2.6 – Rendu de la méthode de Feldman et O'Brien [10].

Feldman et O'Brien [10] discrétisent entièrement leur scène dans une grille de voxels tri-dimensionnelle, et appliquent une simulation de vent en résolvant les équations de Navier-Stokes pour un flot incompressible non visqueux. Une fois ce champ de vent précalculé, des quantités de neige sont injectées dans les cellules sur les bords de la scène, puis déplacées par le champ de vent. Cette manière de procéder permet de diffuser la neige dans la scène à l'aide du vent, et d'obtenir des congères (des amas de neige

entassés par le vent) et des zones de dépression sans neige. Néanmoins, les quantités de neige sont injectées à partir de la bordure de la scène, et non déposées dans la scène depuis le ciel. Ce choix nécessite que pour que la neige atteigne les extrémités de la scène, il faut effectuer de multiples itérations. De plus, il est possible que la quantité de neige ne soit pas uniforme dans une plaine par exemple, ce mécanisme de diffusion de la neige provoquant un dégradé de neige d'un bord à l'autre de la scène.

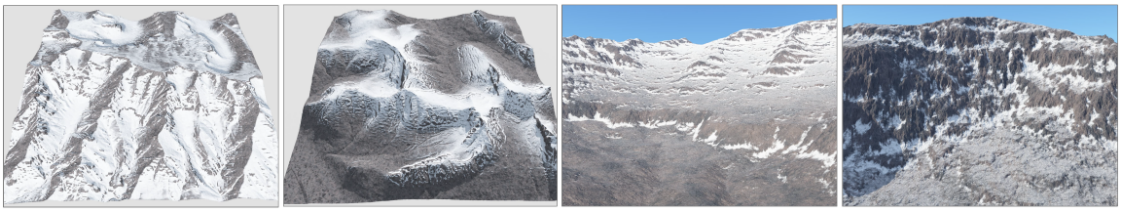


Figure 2.7 – Rendu de la méthode de Cordonnier et al. [5].

Cordonnier et al. [5] proposent une méthode prenant en compte le transport et le déplacement de la neige selon différents phénomènes physiques comme la fonte de la neige, les précipitations, la diffusion, le mouvement par le vent et la stabilité de la couche de neige. Leur méthode permet de simuler dans le temps l'épaisseur de la neige selon ces phénomènes et permet à l'utilisateur de directement interagir avec la neige afin de simuler des avalanches ou encore de générer des traces de ski dans la neige. Elle offre une grande variété de pinceaux afin de modifier la distribution de la neige. Leur implémentation nécessite le précalcul d'une carte de vent ainsi qu'une carte de températures, et le modèle de transport de la neige fonctionne en temps interactif à 30 Hz grâce à une implémentation sur GPU. Cordonnier et al. [5] ont ensuite validé les résultats avec des mesures prises aux États-Unis ; ils ont aussi effectué une étude perceptuelle sur une soixantaine de participants afin de valider si leur modèle paraît réaliste ou non. Nous n'avons pas pu comparer nos résultats sur les mêmes ensembles de données que celles utilisées dans les résultats présentés avec leur contribution, du fait de la découverte de celle-ci au moment de l'écriture de ce mémoire. Nos contributions respectives sont proches, notamment par :

1. La simulation se fait sur des cartes 2D d'informations, notamment des cartes de

vent, de hauteurs. Au contraire de notre méthode, ils proposent aussi l'utilisation d'une carte de températures ainsi que d'une carte d'expositions au soleil.

2. Un modèle physique pour simuler la neige qui consiste en plusieurs étapes :

   (a) Ajout des précipitations de neige

   (b) Mise à jour de la couche de neige

   (c) Diffusion de la neige "poudreuse", c'est-à-dire la neige fraichement déposée

   (d) Transport de la neige par le vent

   De plus ils permettent de simuler des avalanches et l'ajout de traces de ski dans la neige. Leur phase de mise à jour de la couche de neige prend en compte la stabilité de celle-ci, permettant de simuler l'érosion de la surface, un phénomène que nous ne prenons pas en compte dans notre méthode.

3. Cordonnier et al. utilisent un modèle plus réaliste pour simuler la fonte de la neige par rapport à celui que nous utilisons et que nous présentons dans ce document.

4. Le transport de la neige par le vent utilisé par Cordonnier et al. est similaire au modèle de saltation et de suspension que nous proposons d'utiliser.

5. La simulation de vent proposée par Cordonnier et al. est une interpolation d'un vent unidirectionnel sur le terrain, variant en fonction de la pente du terrain, similaire à notre méthode introduite au chapitre 5. Leur méthode ne permet pas de simuler le vent autour d'obstacles complexes car celle-ci ne supporte pas les changements drastiques d'élévation dans la carte de hauteurs, une limitation pour laquelle nous proposons une phase de précalcul d'un champ de vent 3D dans notre scène. Les turbulences dans le vent ne sont pas prises en compte, une limitation similaire de notre méthode.

En plus de ces points communs, Cordonnier et al. simulent la stabilité de la couche de neige ainsi que l'érosion de celle-ci, les avalanches et les traces de ski. Notre contribution est similaire en certains points avec la contribution de Cordonnier et al. [5], notamment

lors de la phase de transport de la neige par le vent. De plus, notre méthode bénéficierait probablement d'un modèle de fonte de la neige et d'un modèle calcul de la stabilité de la couche de la neige similaire aux leurs. Néanmoins, certains problèmes que nous avons tenté de résoudre avec notre contribution, notamment celui d'obtenir un champ de vent 3D plus réaliste et de prendre en compte l'occultation des précipitations de neige par la géométrie de la scène, restent non abordés par leur contribution.

2.3 Méthodes à base de subdivision

Finalement, les méthodes basées sur la subdivision de surfaces forment la troisième catégorie de méthodes de simulation d'accumulation et de transport de neige.

Figure 2.8 – Simulation avec la méthode de Fearing [9].

Fearing [9] propose d'accumuler la neige dans des scènes arbitraires en prenant avantage de la géométrie sous-jacente de la scène. Lorsque l'on cherche à déterminer où la neige tombe dans une scène, on peut intuitivement faire tomber des flocons de neige depuis le ciel jusqu'au sol. Fearing [9] propose de prendre le problème à l'envers, et détermine des surfaces de lancement à partir desquelles il teste l'accessibilité au ciel avec un lanceur de rayons. En fonction du résultat, il peut décider de subdiviser les zones de lancement là où son algorithme pense que plus de détails sont nécessaires. Ensuite, la surface de neige est créée, et un algorithme de stabilité est appliqué afin de s'assurer que

l'épaisseur de la neige est physiquement possible. La quantité de neige est déterminée à partir du nombre de particules qui ont pu se rendre jusqu'au ciel en partant des sites de lancement. La position initiale de ces particules est aléatoire sur la surface de la zone de lancement, et leurs trajectoires sont semi-aléatoires, afin de prendre en compte les turbulences que subissent les flocons de neige. La méthode de Fearing [9] permet d'obtenir des scènes enneigées réalistes, et subdivise les surfaces seulement dans les zones où cela est nécessaire, permettant de sauver du temps de calcul tout en maximisant les ressources pour les zones d'intérêt. Plusieurs critères rentrent en ligne de compte, notamment la distance de la zone de lancement à la caméra, et le nombre de particules n'ayant pas atteint le ciel. Par ailleurs, cette méthode inverse permet de traiter des cas considérés compliqués pour d'autres méthodes, comme l'accumulation de la neige sur de fines branches, ce qu'aucune méthode par carte de hauteurs n'arrive à obtenir. Malgré son potentiel, Fearing [9] ne prend pas en compte le déplacement de la neige par le vent.

Afin de pallier ce défaut, Moeslund et al. [26] intègrent une simulation de vent en employant la méthode de Stam [40] dans une grille, permettant de résoudre les équations de Navier-Stokes pour un fluide incompressible, tout en restant inconditionnellement stable. Ils proposent aussi une métrique de stabilité des surfaces. Afin de déterminer la quantité de neige dans chaque zone de lancement, des particules sont lancées depuis le ciel, jusqu'à entrer en contact avec ces zones, qui peuvent elles aussi êtres subdivisées au besoin. Finalement, un modèle d'éclairage de la neige est aussi proposé.

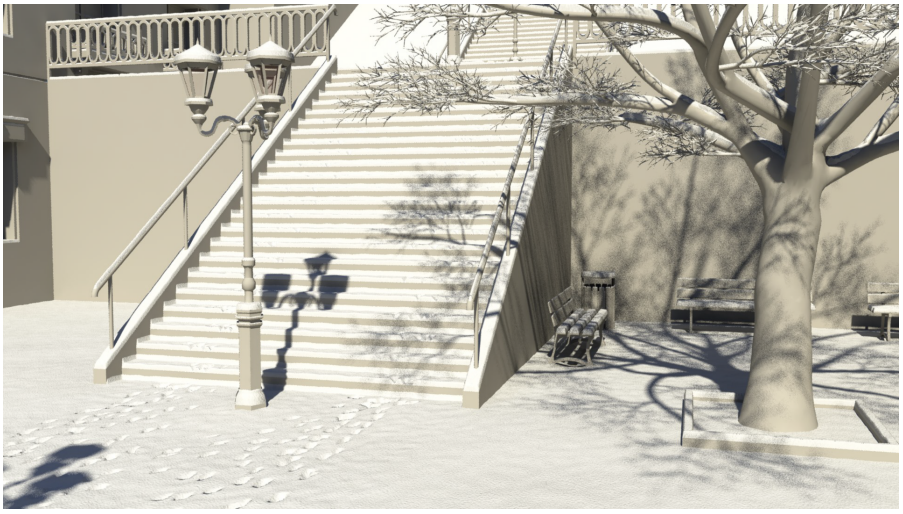


Figure 2.9 – Rendu de la méthode de Grosbellet et al. [13].

Grosbellet et al. [13] intègrent des surfaces de neige dans leur suite d'effets environmentaux, surfaces qui peuvent être influencées par d'autres effets environmentaux, comme la température ou la fonte de la neige. La quantité de neige sur les surfaces est déterminée par sommet de la surface de neige, à partir d'un paramètre utilisateur, ainsi que de la distance au bord de l'objet (distance aux arêtes), des champs de chaleur ou d'humidité placés par l'utilisateur, ainsi que d'un champ d'occultations qui détermine la distance du sommet aux autres objets de la scène. La géométrie de la surface de la neige peut être subdivisée au besoin afin d'obtenir plus de détails, mais au contraire de la méthode de Fearing [9], cette subdivision est globale et non adaptative. L'approche de Grosbellet et al. permet de donner un énorme contrôle à l'utilisateur, avec des résultats qui sont produits dans un temps interactif lors de changements dans la scène. Par contre, des effets environementaux plus complexes comme l'influence du vent ou la fonte de la neige ne sont pas pris en compte dans cette approche, et on peut reprocher la non-automatisation de la création des surfaces de neige qui demande une gestion des paramètres par objet.

Festenberg et al. [11] prennent une approche simulaire à la création des surfaces de neige, mais intègrent des paramètres physiques et des effets plus réalistes, en adéquation

avec la physique de la neige, notamment dû au comportement des flocons entre eux qui permet la création de ponts de neige suspendue. Afin de produire les surfaces enneigées, la scène est découpée en plusieurs couches de cartes de distances, ce qui leur permet d'obtenir une discrétisation de la géométrie de la scène. Ensuite, un maillage uniforme est généré sur les surfaces pouvant recevoir de la neige. Pour chaque sommet de ce maillage, un noyau estime la distance aux objets voisins à partir des cartes de distances, afin d'éviter toute intersection entre la surface de la neige et la géométrie voisine, ainsi que pour déterminer la distance à d'autres maillages de neige dans la scène. Si un autre maillage est suffisamment proche, les deux maillages sont reliés pour générer un pont de neige. La quantité de neige à chaque sommet est déterminée à partir d'une hauteur minimale de neige sur chaque sommet, ainsi que d'un solveur numérique afin de calculer la diffusion de la neige sur les maillages. Pour l'affichage final, la surface est générée par tessellation, et chaque sommet du maillage de neige est déplacé à la bonne hauteur de neige. L'approche de Festenberg et al. [11] permet d'obtenir dans un temps raisonnable des scènes enneigées réalistes, en prenant en compte des effets tels que les ponts de neige entre les surfaces et l'occultation par la géométrie voisine, sans devoir faire de simulations de particules ou de lanceurs de rayons complexes. Malgré tout, aucun effet environmental n'est pris en compte, comme le vent dans la scène.

CHAPITRE 3

SIMULATION DU VENT

Dans ce chapitre, nous voyons comment simuler le vent dans une scène virtuelle, à partir des équations de Navier-Stokes, en employant un point de vue Eulérien. Ce vent nous est utile afin d'obtenir les précipitations de neige dans notre scène par la méthode que nous décrivons dans le chapitre 4, et de simuler le transport de la neige par le modèle introduit au chapitre 5.

3.1 Simulation du vent

La simulation physique de fluides est un domaine très étudié dans de multiples domaines, que ce soit dans l'étude aérodynamique d'une aile d'avion ou de la simulation d'un fleuve en crue pour prévenir les désastres environnementaux. De multiples solutions existent. Notre intérêt dans la simulation de fluides réside dans le besoin de simuler le vent dans nos scènes virtuelles afin de transporter la neige. En effet, le vent est le principal vecteur de transport de neige (comme nous le verrons dans le chapitre 5), et son influence sur la distribution de la neige est non négligeable. De plus, nous souhaitons aussi obtenir des cartes de précipitations de neige, c'est-à-dire la distribution de la neige, due aux vents et à l'occultation par la géométrie de notre scène. Le trajet de nos flocons de neige est donc aussi influencé par ces mouvements d'air. Nous cherchons à obtenir un flot laminaire stable : notre simulation de neige pouvant être faite sur plusieurs jours, voire semaines, les turbulences dans le fluide ne sont pas une caractéristique que nous recherchons. Au contraire, les turbulences étant un phénomène local, bref, son impact a lieu sur une échelle de temps plus brève que celle à laquelle nous travaillons, qui peut s'étendre sur plusieurs jours ou semaines. Nous choisissons donc d'ignorer les turbulences afin de réduire la complexité de notre méthode, qui nécessiterait de simuler le vent en continu en parallèle à notre simulation de neige.

Avant d'introduire le modèle de simulation que nous utilisons, il est important d'introduire quelques opérateurs mathématiques indispensables pour la compréhension des équations qui suivent.

Toutes les équations et principes dans ce chapitre sont présentés en 2D par soucis de lisibilité et de compréhension, mais la généralisation en 3D est équivalente et généralement triviale.

3.2 Opérateurs vectoriels

Tous les **opérateurs** que nous définissons opèrent sur des champs vectoriels $\vec{u} = (u,v)$, des fonctions scalaires $f(x,y)$ ou des fonctions vectorielles $\vec{f}(f,g)$.

L'opérateur **gradient** ∇ d'une fonction $f(x,y)$ décrit sa variation spatiale sur chaque axe et est exprimé comme :

$$\nabla f(x,y) = \left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right) . \tag{3.1}$$

Il est plus commun de l'écrire directement comme $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)$.

Le gradient est aussi défini sur les fonctions vectorielles, et est communément appelé **Jacobien** de la fonction vectorielle. Il est exprimé comme :

$$\nabla (f,g) = \nabla \vec{f} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} . \tag{3.2}$$

L'opérateur **divergence** $\nabla \cdot$ permet de déterminer si le champ vectoriel diverge, converge ou reste neutre. Plus concrètement, dans le cas d'un fluide, cet opérateur permet de calculer son défaut de conservation de volume à n'importe quel point du champ vectoriel $\vec{u}$. Il est exprimé comme :

$$\nabla \cdot \vec{u} = \nabla \cdot (u,v) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} . \tag{3.3}$$

L'opérateur **rotationnel** $\nabla\times$ permet de mesurer à quel point le fluide tourne autour de n'importe quel point du flot. Il est exprimé sous la forme :

$$\nabla \times \vec{u} = \nabla \times (u, v) = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \,. \tag{3.4}$$

Finalement, l'opérateur **Laplacien** $\nabla \cdot \nabla$ ou ∇^2 est obtenu en chaînant l'opérateur divergence à l'opérateur gradient

$$\nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \,. \tag{3.5}$$

Avec ces outils en main, nous pouvons maintenant décrire les équations de Navier-Stokes pour un fluide homogène incompressible, qui sera le modèle principal de notre simulation de vent.

3.3 Équations de Navier-Stokes

Notre intérêt dans la mécanique des fluides provient du besoin de simuler l'écoulement de l'air dans nos scènes, afin de pouvoir transporter de la neige, ainsi que de pouvoir déterminer les précipitations de neige. Dans notre système, nous décrivons l'air comme un liquide visqueux non compressible. L'air dans la réalité est très légèrement visqueux (dans le sens mécanique du terme) et compressible. Mais lorsque l'on cherche à calculer l'écoulement de l'air dans une scène, entre des bâtiments par exemple, l'air se retrouve peu compressé. Cette simplification permet d'accélérer le temps de calcul en utilisant un système d'équations, celles de Navier-Stokes pour une fluide homogène et à viscosité constante. Elles décrivent l'évolution de la vélocité $\vec{u}$ dans le temps t, en fonction de la pression p, de la viscosité ν et des forces externes $\vec{f}$ (comme la gravité par exemple) :

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\,\vec{u} - \frac{1}{\rho}\nabla p + \vec{f} + \nu \nabla \cdot \nabla \vec{u} \,, \tag{3.6}$$

$$\nabla \cdot \vec{u} = 0 \,. \tag{3.7}$$

Obtenir une solution analytique de ces équations est une tâche très complexe, et réalisable que dans des cas très particuliers. Afin de faciliter ce processus, il est plus aisé dans un premier temps de découper les équations 3.6 et 3.7 en sous-équations plus connues, afin de simplifier notre système, et par la suite d'établir des méthodes numériques en fonction de leur type afin de simuler notre flot.

De plus, si l'on souhaite que le flot contourne les objets de notre scène, il faut appliquer des contraintes à nos équations pour assurer que le fluide ne pénètre pas à l'intérieur des objets. Pour ce faire, nous devons appliquer une contrainte supplémentaire à nos équations 3.6 et 3.7, qui prend la forme suivante :

$$\vec{u} \cdot \hat{n} = \vec{u}_{\text{solide}} \cdot \hat{n} \tag{3.8}$$

avec $\vec{u}$ la vitesse de notre fluide, $\hat{n}$ la normale de la surface en contact avec le fluide, et $\vec{u}_{\text{solide}}$ la vitesse de la surface. Dans le cas $\vec{u}_{\text{solide}} = 0$, la condition permet d'assurer que le fluide ne pénètre pas dans le solide, et qu'il prendra la direction tangentielle à la surface. Dans le cas $\vec{u}_{\text{solide}} \neq 0$, on assure que la vitesse relative de l'objet face au flot est tangentielle à la surface. De cette manière, notre fluide ne se déplace jamais dans nos surfaces. Ce type de condition est appelé condition de non-glissement aux bords.

Dans la section suivante, nous voyons comment résoudre et simuler notre fluide avec des méthodes numériques.

3.4 Fluides stables

Notre implémentation se base sur une méthode de simulation de fluides développée par Stam [40, 41], dénommée *Stable Fluids*. Stam part du constat que le champ de pression p est relié au champ de vélocité $\vec{u}$, grâce au théorème fondamental du calcul vectoriel (aussi appelé le théorème de Helmhotz-Hodge), ce qui implique que n'importe quel champ de vecteur $\vec{w}$ se décompose :

$$\vec{w} = \vec{u} + \nabla q \,, \tag{3.9}$$

avec $\vec{u}$ un champ non divergent ($\nabla \cdot \vec{u} = 0$) et q un champ scalaire. N'importe quel champ vectoriel est la somme d'un champ non divergent et d'un champ gradient. Grâce à cette observation, nous pouvons définir un opérateur **P**, qui projette n'importe quel champ $\vec{w}$ sur sa composante non divergente $\vec{u} = \mathbf{P}\vec{w}$. Cet opérateur est défini en multipliant chaque membre de l'équation 3.9 par l'opérateur gradient :

$$\nabla \cdot \vec{w} = \nabla^2 q \; . \tag{3.10}$$

Stam observe ainsi que l'équation 3.10 est une équation de Poisson pour le champ scalaire q, ce qui permet d'utiliser la solution à cette équation pour obtenir le champ non divergent $\vec{u}$:

$$\vec{u} = \mathbf{P}\vec{w} = \vec{w} - \nabla q \; . \tag{3.11}$$

Nous pouvons appliquer ce nouvel opérateur **P** aux deux côtés de l'équation 3.6 afin d'obtenir une équation unique pour les deux équations de Navier-Stokes, 3.6 et 3.7 :

$$\frac{\partial \vec{u}}{\partial t} = \mathbf{P}\left(-(\vec{u} \cdot \nabla)\,\vec{u} + \nu \nabla^2 \vec{u} + \vec{f} \right) \; . \tag{3.12}$$

L'équation 3.12 est résolue à partir d'un état initial $\vec{u}_0 = \vec{u}(x,0)$, en incrémentant par pas de temps Δt. Supposons que nous avons un champ de vent $\vec{w}_0(x) = \vec{u}(x,t)$ à un temps t. Afin d'obtenir le champ de vent au pas de temps suivant, nous appliquons chacun des termes de l'équation 3.12 puis en projetant le résultat grâce au nouvel opérateur **P**, nous obtenons un champ de vent non divergent. Cet algorithme prend la forme suivante :

1. Ajout des forces externes $\vec{w}_0(x) \rightarrow \vec{w}_1(x)$
2. Advection $\vec{w}_1(x) \rightarrow \vec{w}_2(x)$
3. Diffusion $\vec{w}_2(x) \rightarrow \vec{w}_3(x)$
4. Projection $\vec{w}_3(x) \rightarrow \vec{w}_4(x)$

La solution de l'algorithme au temps $t + \Delta t$ est $\vec{w}_4(x) = u(x, t + \Delta t)$. En appliquant de nouveau l'algorithme, on obtient notre vent au temps $t + 2\Delta t$, et ainsi de suite afin d'obte-

nir une animation de notre fluide dans le temps. Dans notre cas, vu que nous souhaitons obtenir le vent moyen dans la scène, nous faisons la moyenne de plusieurs de ces pas de temps afin d'obtenir un flot laminaire. Cette façon de procéder n'est pas entièrement correcte, mais elle donne efficacement un flot stable et généralement satisfaisant pour nos besoins. Nous étudions chaque étape, cas par cas avec nos ajouts pour prendre en compte des solides arbitraires, et nos améliorations à l'algorithme original d'advection.

Étape 1 : Ajout des forces externes

Afin de prendre en compte toutes les forces externes au fluide, comme la gravité par exemple, il est nécessaire de correctement les intégrer dans notre solveur. Ces forces, si elles ne varient pas pendant le pas de temps Δt, peuvent interagir avec le flot par l'équation suivante :

$$\vec{w}_1 = \vec{w}_0 + \Delta t f(x,t) \ , \tag{3.13}$$

avec $f(x,t)$ la fonction décrivant toutes les forces s'exerçant sur le fluide.

Étape 2 : Advection

Le second terme à résoudre est l'advection du vent par lui-même, exprimé par $(\vec{u} \cdot \nabla)\vec{u}$. Stam [40] propose que pour déterminer la vitesse du fluide au point x (l'un des centres des cellules de notre grille de simulation) au temps $t + \Delta t$, il suffit de remonter dans le temps dans notre champ vectoriel $\vec{w}_1(x)$ afin de trouver la vitesse à la position d'origine du point x au temps t, définissant un chemin $p(x,s)$ avec s un point dans le temps. De cette manière, nous évitons les problèmes de stabilité des méthodes d'advection plus classiques, car plus généralement, si le pas de temps Δt est trop grand, des problèmes de stabilité peuvent apparaître dus à la non-linéarité du terme d'advection. Dans l'algorithme proposé par Stam, ce problème ne se présente pas, car la vitesse maximale du champ vectoriel au temps $t + \Delta t$ ne peut pas dépasser la vitesse maximale du champ au temps t précédent, signifiant qu'il n'est pas possible que la simulation "explose" à cause d'un pas de temps trop grand. De cette manière aussi, le pas de temps peut être arbitrairement petit ou grand, cet algorithme d'advection reste inconditionnellement stable.

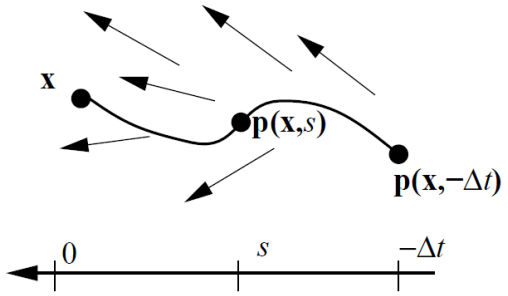


Figure 3.1 – Advection semi-Lagrangienne [40]. L'axe donne le point dans le temps, en partant du point x d'origine au temps t (équivalent au 0 dans le graphique, 0 est l'origine dans notre référentiel local), jusqu'à $-\Delta t$ dans le temps. s est un point quelconque dans le temps dans l'interval $[0, -\Delta t]$. On trouve la position précedente de x dans le temps, à une distance $-\Delta t$ de la position actuelle.

$$\vec{w}_2 = \vec{w}_1(p(x, -\Delta t)) \; . \tag{3.14}$$

Notre solution définie par l'équation 3.14 utilise notre fonction $p(x,s)$ et une intégration de Runge-Kutta d'ordre 4 afin de trouver la vitesse au temps $-\Delta t$. Stam propose initialement une intégration Eulérienne simple, mais dans le cas d'un pas de temps élevé, l'intégration Eulérienne peut facilement traverser des solides sur le chemin. L'intégration de Runge-Kutta d'ordre 4 (RK-4) estime une fonction en plusieurs points afin d'obtenir sa dérivée de manière plus précise. Cela a aussi l'avantage de tester en plusieurs points de notre chemin s'il passe à travers un solide, palliant le problème qu'une intégration d'Euler simple pose lorsque le pas de temps est trop élevé. Néanmoins notre solution ne résout pas totalement ce problème de détermination de pénétration des solides lors de l'intégration, mais nous permet de le mitiger. De plus, il est possible de changer l'ordre de l'intégrateur afin d'être plus précis. En pratique, nous avons observé qu'utiliser RK-4 est amplement suffisant dans nos simulations. L'intégration RK-4 prend la forme suivante :

$$x_{t+\Delta t} = x_t + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

avec

$$
\begin{aligned}
k_1 &= f(x_t, t) \\
k_2 &= f\left(x_t + \frac{\Delta t}{2} k_1, t + \frac{\Delta t}{2}\right) \\
k_3 &= f\left(x_t + \frac{\Delta t}{2} k_2, t + \frac{\Delta t}{2}\right) \\
k_4 &= f\left(x_t + \Delta t k_3, t + \Delta t\right) .
\end{aligned}
$$

Grâce à cet algorithme, nous pouvons déterminer la vitesse au temps $t + \Delta t$ de manière robuste, en prenant en compte les solides le long de la courbe du flot $p(x,s)$.

Étape 3 : Diffusion

Une fois l'advection calculée, il s'agit d'évaluer la diffusion du fluide en fonction de sa viscosité. Stam propose de résoudre l'équation

$$\frac{\partial \vec{w}_2}{\partial t} = \nu \nabla^2 \vec{w}_2 \tag{3.15}$$

en utilisant une méthode implicite, qui va rester stable au contraire de l'équation 3.15, et qui prend la forme :

$$\left(\mathbf{I} - \nu \Delta t \nabla^2\right) \vec{w}_3(x) = \vec{w}_2(x) . \tag{3.16}$$

En discrétisant l'opérateur Laplacien, on obtient un système linéaire avec $\vec{w}_3$ comme inconnue. Cette équation a la forme d'une équation de Poisson $\nabla^2 f = b$, qui est résolvable avec un algorithme de relaxation ou itératif, comme la méthode de Jacobi, permettant de résoudre les problèmes de la forme $\mathbf{A}\vec{x} = \vec{b}$.

Étape 4 : Projection

La dernière étape est celle de projection, pour transformer le champ $\vec{w}_3$ divergent vers le champ $\vec{w}_4$ non divergent, et afin d'appliquer au même moment les conditions aux bords définies dans l'équation 3.8. Pour ce faire nous calculons d'abord la divergence du

champ $\vec{w}_3$ grâce à l'opérateur $\nabla\cdot$ défini dans la section 3.2. Comme établi précédemment par le théorème de Helmhotz-Hodge, il est possible de reprojeter un champ divergent vers un champ non divergent. Pour ce faire, le problème consiste à résoudre l'équation de Poisson suivante :

$$\nabla^2 q = \nabla \cdot \vec{w}_3 \ , \tag{3.17}$$

afin d'obtenir le champ scalaire q, et de soustraire le gradient de celui-ci au champ $\vec{w}_3$ afin d'obtenir notre champ final $\vec{w}_4$:

$$\vec{w}_4 = \vec{w}_3 - \nabla q \ . \tag{3.18}$$

Afin de respecter nos contraintes aux bords, nous prenons en compte aussi l'équation 3.8 dans la résolution de l'équation 3.17. Pour ce faire, une fois le problème de Poisson discrétisé, il suffit de modifier les coefficients de la matrice **A** de notre méthode de relaxation, par exemple par l'algorithme de Jacobi, pour prendre en compte le type des cellules adjacentes.

3.5 Implémentation

Afin de résoudre nos équations 3.6 et 3.7 en appliquant les conditions aux bords définies par l'équation 3.8, il faut tout d'abord décider du type de point de vue que nous souhaitons prendre. Comme mentionné dans le chapitre 2, il existe deux points de vue : Lagrangien et Eulérien. Le premier simule des particules qui représentent notre fluide, dans notre cas des particules d'air, avec une méthode de solution comme SPH [27]. Ces méthodes demandent une grande quantité de particules afin d'obtenir des résultats satisfaisants. De plus, ces méthodes plus réalistes sont en général utilisées lorsque le résultat est affiché directement, comme par exemple la simulation d'une rivière ou d'un jet d'eau. Dans notre situation, nous sommes intéressés par le mouvement global du vent dans la scène, et nous n'allons pas l'afficher. Nous n'avons donc pas besoin de la précision de SPH, mais de rapidité. C'est pour cela que nous nous tournons vers les méthodes développées du point de vue Eulérien.

3.5.1 Grilles de simulation

Le point de vue Eulérien définit un domaine de simulation sur une grille régulière. Sur cette grille, nous stockons au centre des cellules des informations comme la vitesse, la pression, si la cellule est un solide ou non. Différents types de grilles existent, la plus commune étant simplement une grille régulière (ce que nous utilisons). Il existe aussi les grilles du type *M.A.C.* (*Marker And Cell*) [14], où la vélocité est stockée sur les bords des cellules, et la pression au centre, ce qui permet de simplifier les calculs des conditions aux bords. Aussi, il n'est pas forcément nécessaire de simuler toute la grille à la plus haute résolution, surtout dans les zones où le fluide n'est pas présent ou dans des zones où nous n'avons pas besoin de beaucoup de détails. Des stratégies d'optimisation ont donc été développées pour sauver du temps de calcul [4, 24] en utilisant des *Tall Cells* ou des *Octrees*. Dans notre cas, nous utilisons une grille simple en 3D ayant la forme illustrée à la figure 3.2 en 2D, qui consiste de cellules indexées i, j, k en 3D, chacune des cellules stocke sa position $x_{i,j,k}$, sa vélocité $\vec{u}_{i,j,k}$ et sa pression $p_{i,j,k}$.

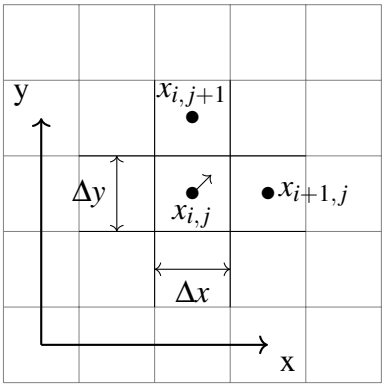


Figure 3.2 – Notre grille de simulation en 2D.

Afin de déterminer les conditions aux bords, notre grille stocke dans chaque voxel un booléen déterminant si la cellule est vide ou non. Ensuite, lors de la simulation, afin de prendre en compte les conditions aux bords déterminées par l'équation 3.8, il suffit simplement de tester les cellules voisines pour savoir si elles sont remplies ou non. Si

elles le sont, la normale est celle du mur de la cellule, comme dans la figure 3.3.

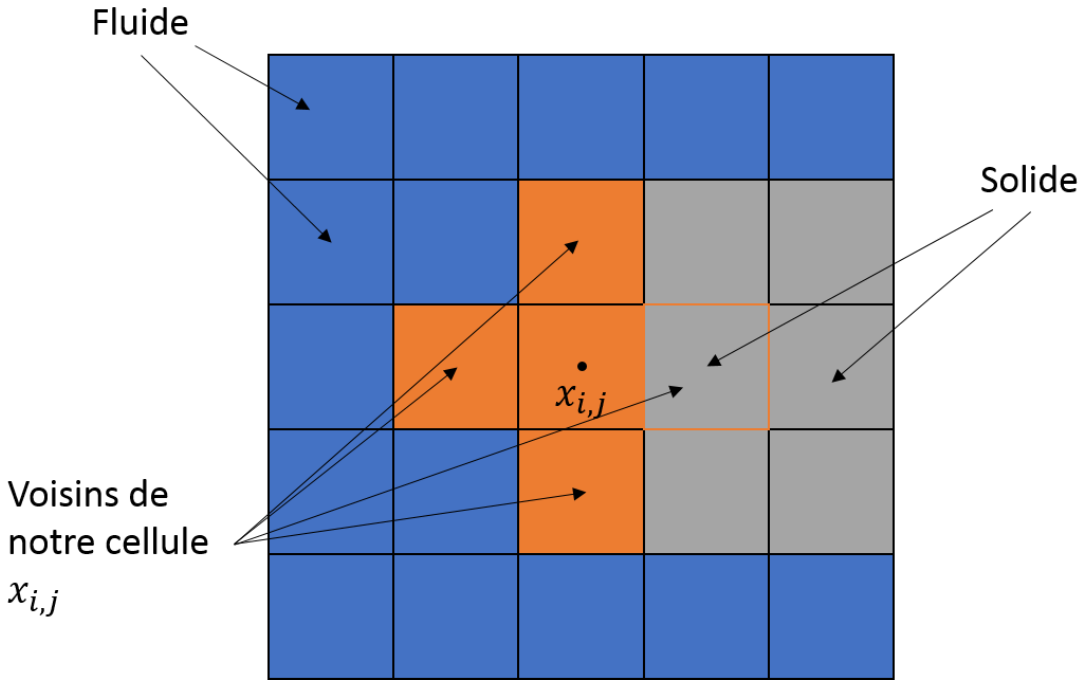


Figure 3.3 – Grille de solides.

Cette approche a néanmoins des limitations. Les normales de la géométrie sous-jacente n'étant pas prises en compte, la résolution des conditions aux bords en sera moins réaliste, et des problèmes d'aliassage peuvent apparaître ou nécessiter un traitement supplémentaire afin de corriger ces artefacts. En pratique, augmenter la résolution de la grille de simulation se révèle suffisant dans notre situation. Des structures de grilles différentes peuvent permettre de pallier ces problèmes, mais les coûts mémoire et computationnel de telles méthodes contre-balancent le gain de précision obtenu.

3.5.2 Solides

Afin de supporter les solides, des modifications doivent être effectuées dans les différentes routines que nous avons introduites dans les sections précédentes, plus particulièrement dans la phase de projection définie à la section 3.4. En effet, nous avons vu qu'il est possible de séparer les équations de Navier-Stokes en sous-systèmes résolvables séparément, le résultat final de ce système étant équivalent au résultat que nous

aurions obtenu en résolvant l'équation complète directement. Afin d'appliquer la condition de non-glissement aux bords, représentée par l'équation 3.8, nous pouvons imposer la condition lors de l'étape de projection, afin d'obtenir en résultat de notre algorithme de simulation de fluides un flot incompressible et respectant nos conditions aux bords.

Pour ce faire, Bridson [2] introduit lors de sa phase de projection le principe de pression *fantôme*. Lorsque l'on discrétise nos conditions aux bords (nous allons étudier le cas d'un solide avec une vélocité $\vec{u}_{\text{solide}} = 0$), et utilise une grille régulière et des cellules représentant la présence de solides ou non, il existe seulement quatre normales possibles (8 en 3D), qui sont orientées selon $(1,0),(0,1),(-1,0)$ et $(0,-1)$. Cela veut dire que la dérivée selon la normale du solide devient une différence finie avant ou arrière en fonction de quelle cellule est solide ou non.

Prenons le cas où la cellule à l'index $(i+1, j)$ est solide. Notre équation 3.8 devient alors :

$$\frac{\partial p}{\partial \hat{n}} = 0 \;\Leftrightarrow\; \frac{p_{i+1,j} - p_{i,j}}{\Delta x} = 0 \tag{3.19}$$

qui après simplification devient :

$$p_{i+1,j} = p_{i,j}\ . \tag{3.20}$$

Ce résultat est généralisable pour les trois autres cas de figure. Nous pouvons prendre avantage de cette nouvelle propriété lors de l'étape de projection, au moment de sa discrétisation. Lors de la résolution de l'équation 3.17, nous discrétisons l'opérateur ∇^2 par les différences finies pour obtenir l'équation suivante :

$$\nabla^2 p = \nabla \cdot \vec{w}_3 \;\Leftrightarrow\; \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2} = \nabla \cdot \vec{w}_3\ . \tag{3.21}$$

Notons que le champ de divergence n'est pas discrétisé dans cette équation par soucis de lisibilité. Ce champ est calculé en amont de cette étape et il agit comme un simple coefficient dans le système de Poisson de la forme $\mathbf{A}\vec{x} = \vec{b}$.

En réorganisant les termes, nous obtenons :

$$\frac{-2p_{i,j}}{\Delta x^2} + \frac{-2p_{i,j}}{\Delta y^2} + \frac{p_{i+1,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} + p_{i,j-1}}{\Delta y^2} = \nabla \cdot \vec{w}_3 \tag{3.22}$$

$$\frac{-2\Delta y^2 p_{i,j} - 2\Delta x^2 p_{i,j}}{\Delta x^2 \Delta y^2} + \frac{p_{i+1,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} + p_{i,j-1}}{\Delta y^2} = \nabla \cdot \vec{w}_3 \tag{3.23}$$

$$\frac{\left(-2\Delta y^2 - 2\Delta x^2\right)}{\Delta x^2 \Delta y^2} p_{i,j} + \frac{p_{i+1,j}}{\Delta x^2} + \frac{p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1}}{\Delta y^2} + \frac{p_{i,j-1}}{\Delta y^2} = \nabla \cdot \vec{w}_3 \, . \tag{3.24}$$

Nous résolvons l'équation 3.24 avec la méthode de Jacobi, permettant d'obtenir la pression p à la cellule (i, j), pour chaque cellule de la grille de simulation. Lors du calcul des coefficients, si l'un des voisins de la cellule contient un solide, nous utilisons l'équation 3.20 afin de modifier le coefficient de $p_{i,j}$ si la cellule voisine est solide, sinon on le soustrait à la divergence du flot $\vec{w}_3$.

Le champ de pression p calculé résultant de la méthode de Jacobi permet d'obtenir le champ final $\vec{w}_4$ non divergent et respectant les conditions aux bords.

Nous avons vu dans ce chapitre comment établir un système de simulation de vent. Nous voyons dans le chapitre 4 comment simuler des flocons de neige dans le champ de vent résultant afin d'obtenir des cartes de précipitations de neige, qui sont nécessaires au modèle de simulation du transport de la neige que nous introduisons dans le chapitre 5.

CHAPITRE 4

SIMULATION DE PARTICULES

4.1 Simulation de particules

Le vent est important dans le transport de la neige, mais encore plus important sont les précipitations de neige, car c'est le seul mécanisme qui fournit de la neige dans une scène. Les flocons de neige se forment à haute altitude dans les nuages, et ces flocons tombent durant des averses. Nous souhaitons simuler ces précipitations afin de déterminer tout d'abord où ces flocons tombent. Une approche simple peut être de simuler directement des flocons de neige, suivre où ils tombent, et les ajouter à une couche de neige. Néanmoins, une telle méthode est extrêmement coûteuse, et est généralement applicable seulement au cinéma d'animation, comme pour la méthode de Stomakhin et al. [42] discutée dans le chapitre 2. Nous voulons connaître les zones dans notre scène qui peuvent recevoir de la neige, de manière efficace, de sorte à obtenir une carte de précipitations, qui représente dans chaque texel la proportion des flocons qui atteignent cette zone. Nous avons deux avantages à procéder de la sorte. Premièrement, si nous souhaitons modifier le taux de précipitations de neige, afin qu'il tombe plus de neige dans notre scène, la carte de précipitations étant indépendante, nous n'avons pas à la recalculer, nous modifions simplement par un facteur linéaire toutes les quantités de neige dans les texels de celle-ci. Deuxièmement, nous nous intéressons à l'accessibilité de la scène pour la neige. Nous ne voulons donc pas simuler la collision des flocons de neige entre eux, ni la collision des particules avec la couche de neige. Nous assumons cette simplification par rapport à la réalité, car elle permet de simplifier notre moteur de simulation de sorte à pouvoir obtenir cette carte de précipitations plus rapidement. Les effets que nous ne prenons pas en compte, comme par exemple la collision des particules avec la surface de neige, sont pris partiellement en compte dans le modèle de simulation de transport de la neige (que nous introduisons dans le chapitre 5), compensant de cette manière certaines limitations de notre méthode.

Nous introduisons dans un premier temps les équations qui vont diriger le mouvement de nos flocons de neige, et par la suite les corrections apportées pour améliorer les cartes de précipitations obtenues.

4.2 Physique des flocons

Afin de simplifier les calculs et le temps de simulation, nous faisons plusieurs simplifications quant à la physique des flocons de neige. Premièrement, nous ne prenons pas en compte les diverses formes des flocons, et assumons que ce sont des sphères simples avec un rayon r (mm) et une masse m (kg). Les forces qui s'appliquent sur un flocon sont multiples, et sont principalement la force que le vent $\vec{v}$ applique sur la surface de la particule $\vec{F}_v$ en Newton (N), la force gravitationnelle $\vec{F}_g$ (N), et la friction $\vec{F}_f$ (N).

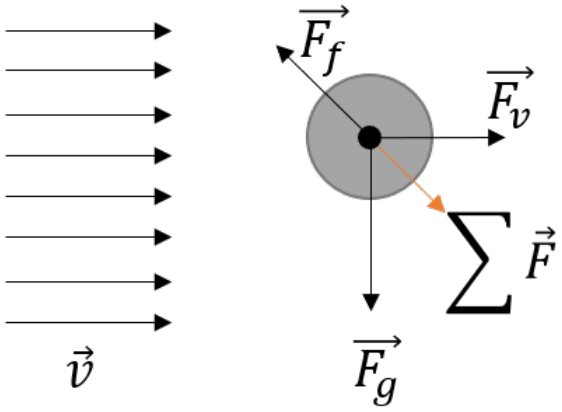


Figure 4.1 – Forces appliquées sur une particule.

Ces forces ne couvrent pas complètement toutes les forces qui s'appliquent sur un flocon, mais permettent d'obtenir une dynamique suffisamment réaliste pour nos besoins. Ces forces, illustrées à la figure 4.1, sont exprimées selon les équations suivantes :

$$A_p = 2\pi r^2 \,, \tag{4.1}$$

$$\vec{F}_f = \frac{1}{2}\rho_a \|\vec{v}_t\|^2 C_f A_p \,, \tag{4.2}$$

$$\vec{F}_v = \frac{1}{2}\rho_a \|\vec{v}_{\text{vent}}\|^2 A_p \,, \tag{4.3}$$

$$\vec{F}_g = M\vec{g} \,, \tag{4.4}$$

avec A_p (m^2) la surface de la sphère exposée dans la direction du vent (surface de l'hémisphère), M sa masse (kg), ρ_a ($\text{kg}\,\text{m}^{-3}$) la masse volumique de l'air, typiquement $1.225\,\text{kg}\,\text{m}^{-3}$, $\vec{v}_t$ la vitesse au temps t de la particule, C_f le coefficient de friction, par exemple 0.47, et $\vec{g}$ l'accélération gravitationnelle égale à $9.806\,65\,\text{m}\,\text{s}^{-2}$.

Avec ces équations en main, simuler les particules consiste à intégrer dans le temps la position et la vitesse de la particule, avec une intégration d'Euler, de sorte que pour obtenir la vitesse $\vec{v}_t$ et la position $\vec{x}_t$ de la particule au temps $t+1$, il suffit d'appliquer les forces afin de mettre à jour la vitesse, puis la position :

$$\vec{a}_{t+1} = \frac{\vec{F}_f + \vec{F}_v + \vec{F}_g}{M} \,, \tag{4.5}$$

$$\vec{v}_{t+1} = \vec{v}_t + \Delta t \vec{a}_{t+1} \,, \tag{4.6}$$

$$\vec{x}_{t+1} = \vec{x}_t + \Delta t \vec{v}_{t+1} \,, \tag{4.7}$$

avec Δt (s) notre pas de temps, et $\vec{a}$ l'accélération de notre particule. Il reste encore à déterminer notre carte de précipitations dans un système qui simule les particules avec le modèle physique que l'on vient de décrire.

4.3 Collisions et carte de précipitations

Afin de déterminer la carte de précipitations, nous devons calculer les collisions entre les particules et la scène. Pour ce faire, nous utilisons la grille de solide précédemment calculée dans le chapitre 3, afin d'accélérer les calculs, en évitant de devoir calculer

les collisions avec tous les polygones de la scène. De plus, déterminer les zones où les particules peuvent atterrir devient trivial, car cela revient à trouver tous les voxels ayant une face pointant vers le haut. Pour calculer la collision, au moment de calculer $\vec{x}_{t+1}$, il s'agit de déterminer si la future position se trouve dans un voxel. Si oui, deux cas apparaissent. Si le voxel a une face pointant vers le ciel, alors le flocon contribue à celui-ci, et est pris en compte dans la carte de précipitations. Dans le cas contraire, nous appliquons une condition de glissement, c'est-à-dire que la particule se déplace dans la direction tangente à la normale du voxel, et qu'elle continue son chemin pour atterrir dans un voxel plus loin dans le flot ou sortir du domaine de simulation. Les particules prennent des trajectoires proches du vent (vent qui respecte les conditions aux bords décrites dans le chapitre 3), mais qui tombent par l'action de la gravité, comme illustré à la figure 4.2.

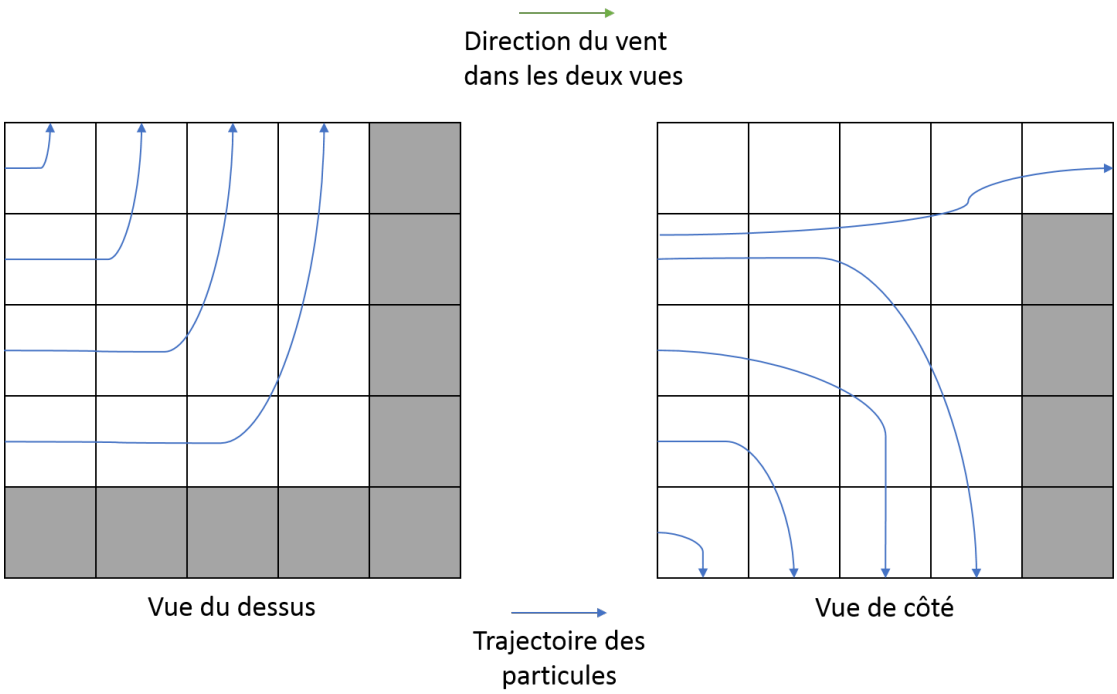


Figure 4.2 – Trajectoires typiques de particules (en bleu) en fonction de solides dans la scène (cellules grises), avec un vent initial donné (en vert).

Nous simulons notre vent et nos particules dans un espace 3D, mais notre simulation de neige se fait dans un espace 2D. Pour cela, nous déterminons une transformation de

l'espace 3D à l'espace 2D, illustrée à la figure 4.3, qui permet de faire correspondre un voxel de notre grille 3D à un texel dans notre carte de précipitations.

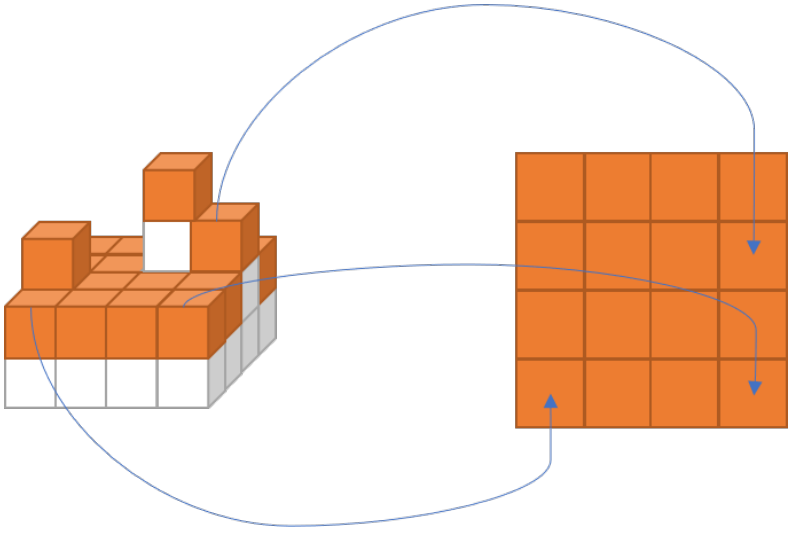


Figure 4.3 – Transformation de l'espace 3D (à gauche) à 2D (à droite).

Grâce à cette correspondance de voxel à texel, nous pouvons connaître dans quels voxels nos flocons tombent. Afin d'obtenir la carte de hauteurs finale, il s'agit de calculer le ratio de particules tombées/reçues dans chaque voxel de notre grille 2D, pour calculer le taux de précipitation dans chaque texel de notre carte de précipitations. En conservant notre neige sous forme de ratio par rapport à toute la neige tombée, assumant une distribution stable, il sera possible de modifier la quantité exacte de neige dans un texel de façon linéaire pour toute la carte. Une carte de vent aura une carte de précipitations. Plusieurs cartes de précipitations peuvent être combinées linéairement, permettant ainsi d'accumuler de la neige avec différents vents et différentes quantités de neige durant de longues périodes de temps.

CHAPITRE 5

TRANSPORT DE LA NEIGE

Le traitement de la distribution de la neige par la géométrie et le vent permet de simuler une partie de base des phénomènes affectant la distribution de neige. Plusieurs phénomènes affectent cette distribution, mais peuvent difficilement être intégrés de façon physique dans les équations de Navier-Stokes. Ce chapitre introduit un modèle qui approxime divers phénomènes physiques qui affectent cette distribution, mais de façon plus simple par a posteriori, tout en étant validé en glaciologie.

5.1 La physique de la neige

L'intérêt dans le transport de la neige n'est pas limité à l'informatique graphique. En effet, que ce soit pour prévoir les précipitations de neige dans une ville, ou comprendre comment la neige se rompt afin de prévoir les avalanches en montagne, connaître comment la neige se comporte est important dans de multiples domaines, et notamment en glaciologie.

La glaciologie est par définition :

> Glaciologie n.f. : Étude des glaciers, de la glace et des régions recouvertes de glace.
>
> (Dictionnaire Larousse)

La glaciologie est le fer de lance de la recherche sur tous les phénomènes reliés à la glace et à la neige. De nombreuses études ont été faites sur le transport de la neige, notre intérêt principal. Au fil des années, plusieurs mécanismes du transport de la neige ont été identifiés, et plus particulièrement, la **saltation**, la **suspension**, et à plus faible échelle, la **sublimation**, ainsi que l'**érosion**, les **précipitations** et la formation de congères. Nous voyons dans ce chapitre plusieurs modèles mathématiques simulant ces différents phénomènes, à partir desquels nous bâtissons notre simulation de transport de la neige présentée dans le chapitre 7.

5.2 Simulation du transport de la neige

Dans notre quête pour simuler le transport de la neige dans des scènes, il est nécessaire de choisir un modèle de transport de la neige, et de choisir une méthode de résolution à la fois efficace et procurant des résultats visuellement réalistes et intéressants, tout en étant facile à contrôler pour un utilisateur sans grande connaissance sur le transport de la neige en général. Les décisions que nous avons prises sont introduites dans les sections suivantes, notamment :

1. Le choix du modèle de transport de neige
2. Notre méthode de résolution numérique
3. Notre type de grille de simulation.

5.3 Grille de simulation

Avec notre modèle, nous devons choisir une forme de résolution de notre système, si nous voulons obtenir une solution analytique ou discrétisée. Dans le contexte de la simulation pour le rendu temps réel, nous choisissons de discrétiser notre domaine, comme nous avons effectué dans le chapitre 3. Nous simulons l'accumulation de la neige dans une grille régulière 2D. Nous discrétisons notre domaine dans une carte de hauteurs : notre grille est alignée sur les axes x et y, tandis que nous simulons l'accumulation de la neige sur l'axe z.

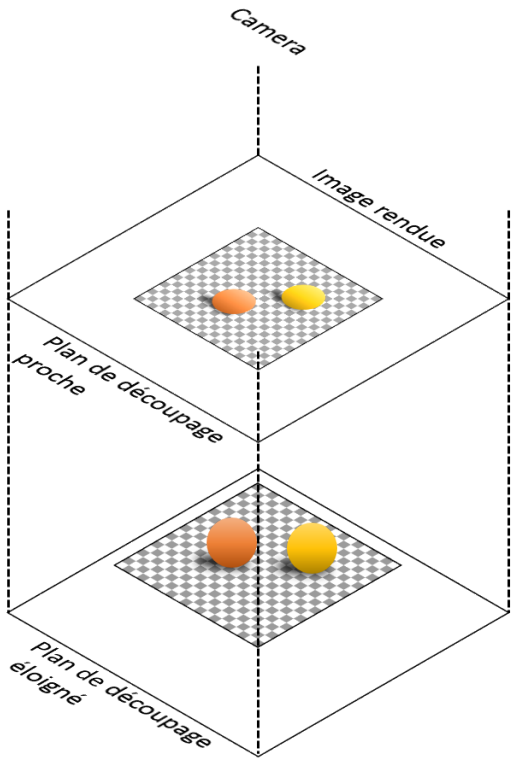


Figure 5.1 – Rendu de la scène orthographique.

Nous faisons le rendu des positions de notre scène comme l'illustre la figure 5.1, dans une texture qui fait office de grille de simulation, grâce à une projection orthographique parallèle à la hauteur sur l'axe z, qui englobe notre scène. Grâce au test de profondeur, nous obtenons dans chaque texel la hauteur maximale locale. Nous utilisons cette carte de hauteurs par la suite comme représentation de notre scène dans notre modèle de simulation du transport de la neige. Mais avant, nous devons extraire certaines informations de cette carte de hauteurs avant de simuler notre neige.

5.4 Informations du terrain

Avant de décrire notre modèle, nous devons tout d'abord extraire plusieurs informations de notre carte de hauteurs, afin de pouvoir calculer plusieurs cartes d'informations sur le terrain ainsi que plusieurs champs de vent, pour simuler en 2D le transport de la

neige. A partir de la carte de hauteurs et du champ de vent calculé dans le chapitre 3, nous extrayons différentes informations, dans cet ordre :

1. Direction et vitesse du vent initial
2. Dérivées de la hauteur en x et en y
3. Angle de la pente
4. Angle azimutal de la surface
5. Influence de la courbure du terrain sur le vent
6. Influence de la pente sur le vent
7. Direction et vitesse du vent résultant

5.4.1 Direction et vitesse du vent initial

Afin de calculer l'influence du terrain sur le champ de vent que nous avons simulé avec la méthode décrite dans le chapitre 3, nous encodons notre champ de vent 3D dans un champ de vent 2D, que l'on modifie par la suite en fonction de la pente du terrain et de sa courbure. Pour cela, nous mesurons à une hauteur que l'utilisateur spécifie le champ de vitesse du vent, et le convertissons en 2D dans une carte d'informations, afin de représenter la vitesse du vent en coordonnées cartésiennes u ($\mathrm{m\,s^{-1}}$) et v ($\mathrm{m\,s^{-1}}$). Nous utilisons aussi la représentation du vent en coordonnées polaires en convertissant le vent u et v, nous permettant d'obtenir la vitesse W ($\mathrm{m\,s^{-1}}$) et la direction θ (rad) du vent, comme illustré à la figure 5.2. Nous pouvons convertir d'une représentation à l'autre par les relations suivantes :

$$u = -W \sin\theta \,, \tag{5.1}$$

$$v = -W \cos\theta \,, \tag{5.2}$$

$$W = \sqrt{u^2 + v^2} \,, \tag{5.3}$$

$$\theta = \frac{3\pi}{2} - \arctan\left(\frac{v}{u}\right) \,, \tag{5.4}$$

avec u, v et W en $\mathrm{m\,s^{-1}}$, et θ en radians.

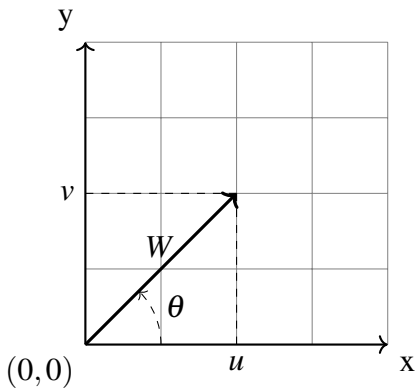


Figure 5.2 – Notre représentation du vent en 2D.

A partir de la carte de hauteurs de notre terrain et des informations que nous extrayons, nous modifions notre champ de vent 2D précalculé afin de mieux prendre en compte la pente et la courbure du terrain. Cette opération est surtout utile dans le cas où notre champ de vent initial est à plus basse résolution que la résolution de notre domaine de simulation de neige, ce qui contribue à ajouter à moindre coût des détails adaptés à la géométrie de la scène. De plus, rappelons que notre simulation de vent ne traite pas les collisions avec la géométrie directement, mais seulement avec des voxels pleins ou vides. De fait, des détails du terrain peuvent être manquants dans la simulation de vent, ce qui produit un champ de vent n'ayant pas de détails assez suffisants. Nous pouvons toutefois mitiger ce problème en modifiant le champ de vent en extrayant des informations de notre carte de hauteurs, afin de mieux prendre en compte l'influence du terrain.

5.4.2 Dérivées de la hauteur en x et en y

Nous avons besoin des dérivées en x et en y du terrain. Cette information permet d'évaluer par la suite la pente du terrain, son angle azimutal, ainsi que l'influence de la pente et de la courbure du terrain sur le vent. Nous obtenons simplement les dérivées en x et en y par une différence finie centrale pour chaque texel de notre domaine à partir de

notre carte de hauteurs $z(x,y)$:

$$\frac{\partial z}{\partial x} \approx \frac{z(x+\Delta x,y)-z(x-\Delta x,y)}{2\Delta x}\,, \tag{5.5}$$

$$\frac{\partial z}{\partial y} \approx \frac{z(x,y+\Delta y)-z(x,y-\Delta y)}{2\Delta y}\,. \tag{5.6}$$

5.4.3 Angle de la pente

Nous estimons l'angle β (rad) de la pente de notre terrain à partir de ces dérivées, par la relation suivante :

$$\beta = \tan^{-1}\left[\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2\right]^{\frac{1}{2}}. \tag{5.7}$$

5.4.4 Angle azimutal de la surface

Nous pouvons aussi calculer l'angle azimutal du terrain ξ (rad), c'est-à-dire dans quelle direction le terrain fait face, avec 0° vers le nord. Cette information est utile afin de calculer l'influence de l'orientation du terrain sur la vitesse du vent. L'azimut est calculé comme suit :

$$\xi = \frac{3\pi}{2} - \tan^{-1}\left(\frac{\frac{\partial z}{\partial y}}{\frac{\partial z}{\partial x}}\right). \tag{5.8}$$

5.4.5 Influence de la courbure du terrain sur le vent

Afin de déterminer le champ de vent utilisé pour la simulation du transport de la neige, nous devons déterminer l'influence de la courbure du terrain sur le vent. En effet, pour que notre vent suive notre terrain le plus fidèlement possible, nous devons prendre en compte sa courbure. Pour cela, nous modifions notre champ en fonction de la courbure en calculant son influence Ω_c selon la formule suivante :

$$\Omega_c = \frac{1}{4}\left[\frac{z-\frac{1}{2}(z_O+z_E)}{2\eta} + \frac{z-\frac{1}{2}(z_S+z_N)}{2\eta} + \frac{z-\frac{1}{2}(z_{SO}+z_{NE})}{2\sqrt{2}\eta} + \frac{z-\frac{1}{2}(z_{NO}+z_{SE})}{2\sqrt{2}\eta}\right], \tag{5.9}$$

avec η le rayon de la courbure moyenne de notre terrain, c'est-à-dire la distance typique dans notre scène entre le plus haut point et le plus bas point (entre le pic d'une montagne et le fond d'une vallée voisine par exemple), z_N, z_S, z_E, z_O, z_{NE}, z_{NO}, z_{SE} et z_{SO} l'élévation (m) respectivement au nord, sud, est, ouest, nord-est, nord-ouest, sud-est et sud-ouest de chaque point de notre carte de hauteurs. Ω_c ne représente pas la courbure du terrain, qui est calculée différemment, mais représente l'influence de celle-ci sur le vent proche du sol. Ce poids est normalisé dans l'interval $[-0.5, 0.5]$, en calculant Ω_{c_min} et Ω_{c_max}, qui sont respectivement le poids minimal et maximal de notre domaine que nous venons de calculer avec l'équation 5.9.

5.4.6 Influence de la pente sur le vent

La courbure n'est pas la seule influence sur la vitesse et la direction du vent, la pente est importante aussi. Nous devons calculer l'influence de la pente sur le vent, car plus la pente est importante, plus le vent à sa surface va accélérer. Afin de calculer le poids de la pente Ω_p, nous utilisons la relation suivante :

$$\Omega_p = \beta \cos(\theta - \xi) \ , \tag{5.10}$$

où ξ (rad) est l'azimut du terrain, β (rad) la pente du terrain, et θ (rad) la direction du vent, calculés précédemment.

5.4.7 Direction et vitesse du vent résultant

Avec les poids Ω_c et Ω_p en main, nous pouvons modifier notre carte de vent afin de prendre en compte plus adéquatement le terrain dans notre champ de vent. Comme indiqué plus tôt nous pouvons obtenir un champ de vent 2D plus détaillé à partir du champ 3D, en prenant en compte la forme de notre scène sous-jacente, à partir de la carte de hauteurs obtenue.

Nous calculons d'abord un facteur de modification du vent W_w, obtenu en moyennant

les poids Ω_c et Ω_p par la relation suivante :

$$W_w = 1 + \gamma_p \Omega_p + \gamma_c \Omega_c \ , \tag{5.11}$$

avec γ_p et γ_c des paramètres utilisateurs, afin de contrôler si la vitesse du vent est plus influencée par la pente ou la courbure du terrain, ou si elles sont aussi conséquentes l'une et l'autre. Puisque nous cherchons à interpoler linéairement entre les deux poids, la somme de γ_p et de γ_c doit toujours être égale à un :

$$\gamma_p + \gamma_c = 1 \ . \tag{5.12}$$

La vitesse modifiée de notre champ de vent W_t est obtenue par :

$$W_t = W_w W \ , \tag{5.13}$$

avec W le vent mesuré dans notre champ de vent 3D obtenu avec notre méthode décrite dans le chapitre 3 à une hauteur z_r (m) donnée en paramètre.

Ensuite, nous modifions la direction du vent par un coefficient θ_d (rad), qui représente la différence entre le vent initial et l'angle azimutal de la pente, obtenu par le rapport :

$$\theta_d = -0.5 \Omega_p \sin\left[2\left(\xi - \theta\right)\right] \ . \tag{5.14}$$

La direction modifiée θ_t (rad) du vent est obtenue en ajoutant le coefficient θ_d à la direction initiale θ (rad) mesurée elle aussi dans le champ de vent 3D à une hauteur z_r (m) :

$$\theta_t = \theta + \theta_d \ . \tag{5.15}$$

Les nouvelles vitesses W_t et directions θ_t sont converties en coordonnées cartésiennes dans les cartes d'informations u et v en utilisant les équations 5.1 et 5.2 décrites précédemment.

5.5 Modèle du transport de neige

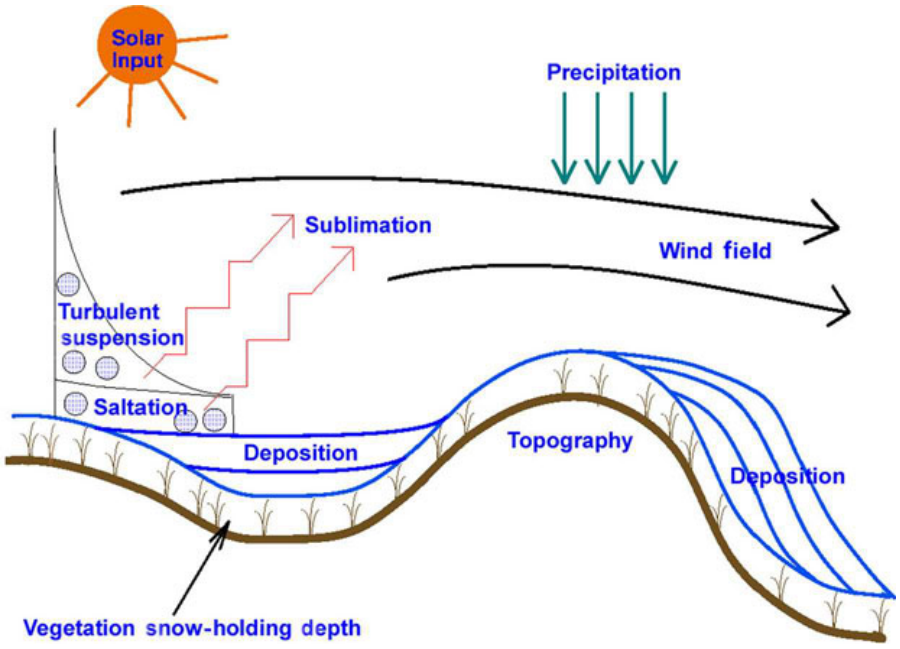


Figure 5.3 – Différents phénomènes de transport de la neige [20].

Une fois que nous avons obtenu le champ de vent modifié de la section 5.4.7, nous pouvons commencer à simuler notre neige. Rappelons que le modèle sur lequel se base notre transport est inspiré principalement par Liston et al. [1, 7, 8, 18–22, 32, 38]. Il consiste en une équation directrice décrivant l'évolution de l'épaisseur de la neige ζ en chaque point de la scène en termes de quantité de neige transportée par saltation Q_{salt}, par suspension Q_{susp} et par sublimation Q_{subl} dans la scène, tel qu'illustré dans la figure 5.3. Il est exprimé comme suit :

$$\frac{d(\rho_s \zeta)}{dt} = \rho_w P - \left(\frac{dQ_{salt}}{dx} + \frac{dQ_{salt}}{dy} + \frac{dQ_{susp}}{dx} + \frac{dQ_{susp}}{dy} \right) + Q_{subl} \,, \tag{5.16}$$

avec ζ l'épaisseur de neige (m), Q_{salt} ($\mathrm{kg\,m^{-1}\,s^{-1}}$) le taux horizontal de transport de masse de neige par saltation, Q_{susp} ($\mathrm{kg\,m^{-1}\,s^{-1}}$) le taux horizontal de transport de neige suspendue par turbulences, Q_{subl} ($\mathrm{kg\,m^{-2}\,s^{-1}}$) le taux de sublimation des particules de neige, ρ_s ($\mathrm{kg\,m^{-3}}$) la densité de la neige, $\rho_w = 1000\,\mathrm{kg\,m^{-3}}$ de l'eau, et P le taux de

précipitation ($\mathrm{m\,s^{-1}}$). t est le temps (s), x et y les coordonnées (m). Notons que le modèle est invariant dans le temps, c'est-à-dire que les résultats d'une itération de simulation n'influencent pas les taux de saltation, suspension et de sublimation dans l'itération suivante, seulement la quantité de neige ζ présente dans la scène à l'instant t.

Afin de simuler le transport de la neige dans notre scène, nous devons intégrer dans le temps l'équation 5.16. Comme mentionné dans le chapitre 3, plusieurs techniques d'intégration dans le temps existent, et nous utilisons une intégration d'Euler directe :

$$\zeta_{t+1} = \zeta_t + \Delta t \frac{d\zeta_t}{dt} \,. \tag{5.17}$$

Afin d'obtenir la nouvelle épaisseur de neige ζ_{t+1}, il faut simuler le transport de neige en fonction des effets précédemment mentionnés : pour cela notre algorithme de simulation du transport fonctionne comme suit :

1. Mise à jour du seuil d'érosion critique
2. Mise à jour de la force d'érosion du vent
3. Calcul du transport dû à la saltation
4. Calcul du transport dû à la suspension
5. Calcul du transport dû à la sublimation
6. Calcul de la fonte de la neige.

Avant d'appliquer notre algorithme nous assumons que nous avons auparavant mis à jour pour le temps t la carte de précipitations et la carte de hauteurs afin de prendre en compte la neige nouvellement déposée.

5.5.1 Seuil d'érosion critique

Le seuil d'érosion critique u_{*t} est le seuil critique à partir duquel le processus de saltation débute. Si la force de friction que le vent applique sur la surface de neige est supérieure à ce seuil, nous considérons que la saltation peut avoir lieu. En d'autres termes,

c'est le seuil critique qui détermine si le vent est assez fort pour soulever et déplacer des particules de neige de la surface. Afin de calculer ce seuil, nous devons tout d'abord calculer la densité de la neige ρ_s, afin de connaître sa résistance au vent. Si la neige est fraîche comme de la poudreuse, elle est facilement déplaçable ; au contraire une neige plus ancienne a pu glacer et durcir avec le temps.

5.5.1.1 Densité de la neige

Afin de calculer la densité de la neige ρ_s ($\mathrm{kg\,m^{-3}}$), il faut tout d'abord calculer la densité de la neige fraîche déposée ρ_{ns} ($\mathrm{kg\,m^{-3}}$), qui est donnée par :

$$\rho_{ns} = 50 + 1.7\,(T_{WB} - 258.16)^{1.5}\ , \tag{5.18}$$

avec T_{WB} (°K) la température de l'air. Par la suite, deux cas sont possibles pour calculer la densité de la neige : les périodes de précipitation, et les périodes sans précipitation.

Durant les périodes de précipitation, la densité de la nouvelle neige ρ_{ns} est utilisée comme densité pour ρ_s si la vitesse du vent est inférieure à $5\,\mathrm{m\,s^{-1}}$. Sinon, pour les vitesses supérieures ou égales à $5\,\mathrm{m\,s^{-1}}$, la densité de la neige ρ_s est donnée par :

$$\rho_s = \rho_{ns} + \rho_u\ , \tag{5.19}$$

avec

$$\rho_u = D_1 + D_2\,[1.0 - \exp\,[-D_3\,(W - 5.0)]]\ , \tag{5.20}$$

et $D_1 = 25.0$, $D_2 = 250.0$ et $D_3 = 0.2$ ($\mathrm{kg\,m^{-3}}$). D_1 est la déviation en densité pour un vent de 5 $\mathrm{m\,s^{-1}}$, D_2 le maximum de croissance de la densité due au vent, et D_3 une constante de contrôle de la croissance des vents faibles à élevés.

Durant les périodes sans précipitation, la densité de la neige évolue dans le temps selon l'équation différentielle 5.21, qui exprime la densité de la neige en fonction de la différence de température entre l'air et la neige, ainsi que plusieurs facteurs dus à la vitesse du vent et des constantes thermiques :

$$\frac{\partial \rho_s}{\partial t} = CA_1 U \rho_s \exp\left[-B\left(T_f - T_s\right)\right] \exp(-A_2 \rho_s) \,, \tag{5.21}$$

avec T_f (° K) la température de congélation, T_s (° K) la température de la neige (égale ou inférieure à la température de l'air), et les constantes $B = 0.08\ ^\circ\mathrm{K}^{-1}$, $A_1 = 0.0013\ \mathrm{m}^{-1}$ et $A_2 = 0.021\ \mathrm{m}^3\,\mathrm{kg}^{-1}$, $C = 0.10$. U est donné par :

$$U = \begin{cases} E_1 + E_2\left[1.0 - \exp\left[-E_3\left(W - 5.0\right)\right]\right] & W \geq 5.0\,\mathrm{m\,s}^{-1} \,, \\ 1.0 & W < 5.0\,\mathrm{m\,s}^{-1} \,, \end{cases} \tag{5.22}$$

avec $E_1 = 5.0$, $E_2 = 15.0$ et $E_3 = 0.2$ ($\mathrm{m\,s}^{-1}$).

De cette façon, la densité de la neige évolue dans le temps en fonction de la température et de la vitesse du vent.

5.5.1.2 Dureté de la neige

La dernière étape avant de pouvoir calculer le seuil critique d'érosion consiste à calculer la dureté σ (kPa) de la neige en fonction de sa densité :

$$\sigma = 1.36\ \exp(0.013\rho_s) \,. \tag{5.23}$$

Pour les densités comprises entre 300 et 450 $\mathrm{kg\,m}^{-3}$, une relation existe entre le seuil critique d'érosion u_{*t} et la dureté de la neige σ, qui est donnée par Kotlyakov [15] :

$$\sigma = 267 u_{*t} \,. \tag{5.24}$$

5.5.1.3 Seuil critique d'érosion u_{*t}

Le seuil critique d'érosion u_{*t} estime la vitesse de vent nécessaire afin de commencer à soulever les particules de neige, ce qui permet de décrire les quantités de neige transportées par saltation, suspension et sublimation. Ce seuil se calcule selon deux cas : pour les densités de neige comprises entre 50 et 300 $\mathrm{kg\,m}^{-3}$, et les densités comprises entre 300 et 450 $\mathrm{kg\,m}^{-3}$:

$$u_{*t} = \begin{cases} 0.005\ \exp(0.013\rho_s) & 300 < \rho_s \leq 450\ , \\ 0.10\ \exp(0.003\rho_s) & 50 < \rho_s \leq 300\ . \end{cases} \tag{5.25}$$

5.5.2 Force d'érosion du vent

Le vent est le principal vecteur de transport de neige. C'est lui seul qui déplace la majorité de la neige dans nos simulations. S'il est assez fort, il commence à transporter de la neige, soit par saltation, suspension ou sublimation. Afin de déterminer la force du vent sur la surface, on calcule la force de stress appliquée par le vent sur la surface de neige. Si elle est supérieure à u_{*t}, le vent est assez fort pour transporter de la neige, donc pour provoquer le mécanisme de saltation. Cette force d'érosion est exprimée en terme de vitesse du vent, et est représentée par u_* ($\mathrm{m\,s^{-1}}$). Elle est obtenue de la manière suivante :

$$u_* = u_r \frac{\kappa}{\ln(z_r/z_0)}, \tag{5.26}$$

avec u_r ($\mathrm{m\,s^{-1}}$) la vitesse du vent à une hauteur de référence z_r (m), z_0 (m) la longueur de rugosité de la surface, et κ la constante de von Kármán ($\kappa \approx 0.40$). u_r et z_r sont connues, u_r est extrait de la carte de vent que nous avons calculée précédemment.

Il faut cependant déterminer z_0, en fonction de la quantité de neige actuellement présente, par le type de la surface et le type de végétation présent ou non. Le type de végétation détermine la profondeur de neige C_v qu'elle peut retenir en protégeant du vent. Des hautes herbes peuvent retenir 50 cm de neige, tandis que du béton ne pourra retenir que quelques millimètres tout au plus. La longueur de rugosité de la surface z_0 est donnée par deux cas de figure : l'épaisseur de neige présente est inférieure à l'épaisseur de neige que la végétation peut retenir C_v, et le cas contraire, s'il y a plus de neige que de capacité disponible.

Dans le cas où la quantité de neige au sol est inférieure ou égale à la quantité de neige que la surface peut retenir, on calcule tout d'abord le ratio d'épaisseur de neige F_s entre

la quantité de neige disponible et la capacité de rétention de neige de la surface :

$$F_s = \frac{\zeta}{C_v}. \tag{5.27}$$

Par la suite, z_0 est simplement calculée en interpolant linéairement entre la longueur de rugosité de la neige z_{0_neige} (m) et la longueur de rugosité de la végétation z_{0_veg} (m) :

$$z_0 = F_s z_{0\_\text{neige}} + (1 - F_s) z_{0\_\text{veg}}. \tag{5.28}$$

Une fois z_0 calculée, on la réintègre dans l'équation 5.26 afin d'obtenir u_*.

Dans le cas où l'épaisseur de neige ζ est plus grande que l'épaisseur de neige que la surface peut retenir, la longueur de rugosité de la surface est donnée par l'équation suivante :

$$z_0 = 0.12 \frac{u_*^2}{2g}\,, \tag{5.29}$$

avec g la constante d'accélération gravitationnelle égale à $9,80665\ \mathrm{m\,s^{-2}}$.

On résout pour z_0 et u_* avec les équations 5.26 et 5.29, mais comme nous pouvons le remarquer, les deux équations sont dépendantes, et nécessitent un traitement supplémentaire afin de résoudre ce système. De plus, si u_* obtenue est supérieure au seuil de friction u_{*t} , alors la saltation est présente. Sinon, $z_0 = z_{0\_\text{neige}}$, et u_* est obtenue via l'équation 5.26.

Méthode de Newton-Raphson Les équations 5.26 et 5.29 forment le système suivant :

$$\begin{cases} u_* = u_r \frac{\kappa}{\ln(z_r/z_0)} \\ z_0 = 0.12 \frac{u_*^2}{2g}\,. \end{cases} \tag{5.30}$$

En réarrangeant les termes, nous obtenons :

$$\begin{cases} f_1(u_*, z_0) = u_* - u_r \frac{\kappa}{\ln(z_r/z_0)} = 0 \\ f_2(u_*, z_0) = z_0 - 0.12 \frac{u_*^2}{2g} = 0 \,. \end{cases} \tag{5.31}$$

Un tel système revient à trouver les racines des fonctions $f_1(u_*, z_0)$ et $f_2(u_*, z_0)$, problème que nous pouvons résoudre itérativement avec l'algorithme de Newton-Raphson. Cet algorithme est défini plus généralement sur les problèmes de la forme :

$$\begin{cases} f_1(x_1, \ldots, x_n) = f_1(\mathbf{x}) = 0 \\ \ldots \\ f_n(x_1, \ldots, x_n) = f_n(\mathbf{x}) = 0 \,, \end{cases} \tag{5.32}$$

avec $\mathbf{x} = [x_1, \ldots, x_n]^T$.

L'algorithme de Newton-Raphson donne la solution $\mathbf{x}^{k+1}$ après $k+1$ itérations en fonction du résultat $\mathbf{x}^k$ obtenu après k itérations comme étant :

$$\mathbf{x}^{k+1} \Leftarrow \mathbf{x}^k - \mathbf{J}_f^{-1}(\mathbf{x}^k) f(\mathbf{x}^k), \tag{5.33}$$

avec $\mathbf{J}_f$ le Jacobien de notre ensemble de fonction, défini comme :

$$\mathbf{J}_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} . \tag{5.34}$$

Dans notre situation, $\mathbf{J}_f$ est égal à :

$$\mathbf{J}_f = \begin{bmatrix} \frac{\partial f_1}{\partial u_*} & \frac{\partial f_1}{\partial z_0} \\ \frac{\partial f_2}{\partial u_*} & \frac{\partial f_2}{\partial z_0} \end{bmatrix} = \begin{bmatrix} 1 & \frac{-u_r \kappa}{z_0 \ln(z_r/z_0)^2} \\ -\frac{0.12}{g} u_* & 1 \end{bmatrix} . \tag{5.35}$$

et $\mathbf{J}_f^{-1}$ est donc égal à :

$$\mathbf{J}_f^{-1} = \frac{1}{1 - \frac{\partial f_1}{\partial z_0} \frac{\partial f_2}{\partial u_*}} \begin{bmatrix} 1 & -\frac{\partial f_1}{\partial z_0} \\ -\frac{\partial f_2}{\partial u_*} & 1 \end{bmatrix} . \tag{5.36}$$

Le principe de la méthode de Newton-Raphson est d'appliquer plusieurs itérations de l'algorithme jusqu'à convergence de la solution. De cette manière, grâce à l'équation 5.33, nous pouvons obtenir la solution pour z_0 ainsi que pour u_*. Avec cette information, nous pouvons déterminer si saltation se produit ou non.

5.5.3 Saltation

De la saltation est possible si $u_* \geq u_{*t}$, c'est-à-dire si la force du vent est suffisamment élevée pour dépasser le seuil critique d'érosion de la surface de neige. Il faut tout d'abord calculer le taux de transport horizontal maximum par saltation Q_{s_max} ($\mathrm{kg\,m^{-1}\,s^{-1}}$) que le vent peut supporter, qui est défini par :

$$Q_{s\_max} = \frac{0.68}{u_*} \left(\frac{\rho_a}{g} \right) u_{*t} \left(u_*^2 - u_{*t}^2 \right) \ , \tag{5.37}$$

avec ρ_a la densité de l'air ($\mathrm{kg\,m^{-3}}$). Ce taux maximal indique le volume maximal de neige que le vent peut transporter, en fonction de la densité de l'air ρ_a, ainsi que la différence entre le seuil limite d'érosion u_{*t} et le stress réel appliqué à la surface u_*. Q_{s_max} varie dans l'espace et dans le temps, dépendamment de la vitesse du vent ainsi que de la quantité de neige disponible.

À partir de ce taux de transport maximal via saltation, nous devons calculer le volume réel transportable Q_s dans notre domaine. Nous connaissons la force qu'applique le vent sur la surface de neige, ainsi que le taux maximal. De plus, l'usager peut spécifier si il y a assez de neige en dehors du domaine pouvant entrer dans celui-ci ou non. Nous pouvons itérer dans la direction du vent, et ajouter la quantité de neige transportable par le vent à ce point, jusqu'à atteindre le taux maximal de transport par saltation Q_{s_max}, soit ne plus avoir suffisamment de neige à transporter. Nous devons prendre en compte le cas où le stress sur la surface décroît, et le cas inverse :

$$Q_s(x^*) = \min \begin{cases} Q_s(x^* - \Delta x^*) \\ Q_{s\_max}(x^*) \end{cases} \quad \text{pour } \frac{\partial u_*}{\partial x^*} < 0\,, \tag{5.38}$$

$$Q_s(x^*) = Q_s(x^* - \Delta x^*) + \frac{\mu}{f}\Delta x^* \left(Q_{s\_max} - Q_s(x^* - \Delta x^*)\right) \quad \text{pour } \frac{\partial u_*}{\partial x^*} \geq 0\,, \tag{5.39}$$

avec x^* (m) la coordonnée horizontale dans la direction du vent, $\mu = 3$, $f = 300\,\text{m}$ [22] la distance suffisante pour atteindre le taux maximum de saltation Q_{s_max} si il y a assez de neige et de vent, Δx^* (m) la distance entre les points de notre grille de simulation. Dans le cas où le stress appliqué à la surface u_* décroît, l'équation 5.38 définit le taux de saltation comme étant le mimimum entre le taux de saltation en amont du vent de notre point x^* à une distance Δx^*, et le taux maximal que le vent peut transporter en notre point. Dans le cas où le stress appliqué à la surface augmente avec la distance, alors le taux de saltation Q_s est défini comme le taux de saltation à la cellule en amont du vent à une distance Δx^*, auquel on ajoute la quantité de neige transportable par notre vent au point x^* comme la différence entre le taux maximum et la quantité de saltation déjà obtenue précédemment.

Afin de résoudre ce système, il s'agit simplement d'itérer dans chaque direction du vent, c'est-à-dire dans la direction nord vers sud, sud vers nord, est vers ouest, et ouest vers est, afin d'obtenir pour chaque direction le taux de saltation pour cette direction du vent, et interpoler linéairement en fonction de la vitesse du vent dans cette direction. Une fois le taux de saltation obtenu Q_s, nous pouvons calculer le taux de suspension ainsi que le taux de sublimation.

5.5.4 Suspension

Nous venons de voir comment calculer le taux de transport par saltation Q_{salt}. En plus du transport par saltation, la neige est aussi transportée par du transport turbulent par suspension. Au-dessus de la surface et de la couche de saltation, des particules de neige sont transportées à de plus grandes distances par les turbulences locales du vent.

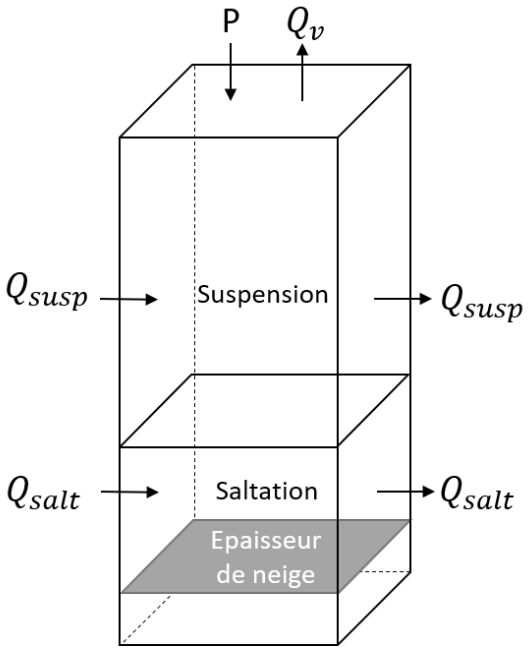


Figure 5.4 – Illustration des différents phénomènes simulés dans une cellule par notre modèle.

Cela est provoqué par la couche limite du vent, c'est-à-dire l'interface entre le flot et la surface, dépendamment de sa viscosité.

Afin de calculer le taux de transport par suspension, il faut estimer la concentration du nuage de particules de neige en suspension depuis la couche de saltation jusqu'au point où la concentration devient nulle. Le taux de transport par suspension Q_{susp} ($\mathrm{kg\,m^{-1}\,s^{-1}}$) est alors donné par :

$$Q_{susp}(x^*) = \int_{h_*}^{z_t} \phi_t(x^*, z) u(x^*, z) dz \,, \tag{5.40}$$

avec h_* (m) la limite supérieure de la couche de saltation, z_t (m) la limite supérieure de la couche de suspension où la concentration est nulle, z (m) la hauteur du point x^*, ϕ_t ($\mathrm{kg\,m^{-3}}$) la concentration du nuage de particules, u ($\mathrm{m\,s^{-1}}$) la vitesse du vent précalculée auparavant dans la carte de vent. Pour obtenir le taux de saltation en un point x^*, nous devons calculer la concentration ϕ_t à la hauteur z en question :

$$\phi_t(x^*,z) = \phi_r \left[\left(\frac{\phi^*}{\phi_r}\frac{u_*}{s} + 1 \right) \left(\frac{z}{z_{tr}} \right)^{-s/\kappa u^*} - \frac{\phi^*}{\phi_r}\frac{u_*}{s} \right] , \tag{5.41}$$

avec ϕ_r ($\mathrm{kg\,m^{-3}}$) la concentration à un niveau de référence z_{tr} (m), s ($\mathrm{m\,s^{-1}}$) la vitesse d'arrêt des particules de neige, assumée à 0.3 $\mathrm{m\,s^{-1}}$, et ϕ^* un paramètre de mise à l'échelle de la concentration. La concentration ϕ_t est donc obtenue comme un rapport entre la hauteur à laquelle nous estimons la concentration, la vitesse du vent à cette hauteur, et la vélocité moyenne des particules. Le ratio $\frac{\phi^*}{\phi_r}$ est proportionnel au rapport entre la vélocité de notre vent à la hauteur z et la vélocité de référence :

$$\frac{\phi^*}{\phi_r} = 0.5\frac{u_*}{u_r} . \tag{5.42}$$

Afin de résoudre l'équation 5.41, nous définissons la hauteur de référence z_{tr} comme étant h_*, afin d'assurer la continuité entre la couche de saltation et la couche de suspension. h_* est estimée par :

$$h_* = 1.6\frac{u_*^2}{2g} . \tag{5.43}$$

La concentration de référence ϕ_r est obtenue en fonction de la quantité de saltation Q_{salt}. En effet, nous assumons que la concentration de référence est obtenue au niveau h_*, à la limite de la couche de saltation et de suspension. En assumant que la concentration dans la couche de saltation est uniforme, alors ϕ_r est simplement exprimée en fonction du taux de saltation Q_{salt}, de la concentration de neige transportée, et de la vitesse du vent u_p à la hauteur h_*, donnant la vitesse horizontale des particules. u_p est formulé en fonction du seuil critique d'érosion u_{*t} :

$$u_p = 2.8u_{*t} . \tag{5.44}$$

Finalement, ϕ_r est obtenu par le rapport :

$$\phi_r = \frac{Q_{salt}}{h_* u_p} . \tag{5.45}$$

Grâce à cette modélisation de la suspension, nous pouvons résoudre l'équation 5.40 avec l'équation 5.41 afin d'obtenir le taux de transport horizontal dû à la suspension Q_{susp}. Mais le vent n'est pas le seul moteur de transport de la neige. Le soleil agit aussi sur la surface de neige, provoquant sa fonte et sa sublimation : la transformation des cristaux de glace en vapeur d'eau.

5.5.5 Sublimation

Après avoir calculé le taux de saltation Q_{salt} ainsi que le taux de sublimation Q_{subl}, prenant en compte ainsi le transport de la neige par le vent, nous devons encore prendre en compte la transformation de la neige par la température, l'exposition au soleil et l'humidité. Pour cela, le taux de sublimation Q_{subl} ($\mathrm{kg\,m^{-2}\,s^{-1}}$) est formulé en fonction de ces différents phénomènes, comme étant la quantité totale de neige transportée par unité d'air dans le temps dans chaque région de notre domaine. Cette quantité totale est exprimée par :

$$Q_{subl}(x^*) = \int_0^{z_t} \psi(x^*,z)\phi(x^*,z)dz\,, \tag{5.46}$$

avec ψ ($\mathrm{s^{-1}}$) le coefficient de perte de neige, et ϕ ($\mathrm{kg\,m^{-3}}$) la concentration verticale du nuage de particules de neige. Notons que nous intégrons de la hauteur zéro sur la surface de la neige, jusqu'à la hauteur z_t, où la concentration ϕ devient nulle. Il est possible d'exprimer séparément le taux de sublimation de la couche de saltation et de la couche de suspension, la relation devenant :

$$Q_{subl}(x^*) = \psi_s\phi_s h_* + \int_{h_*}^{z_t} \psi_t(x^*,z)\phi_t(x^*,z)dz\,, \tag{5.47}$$

avec ϕ_s et ϕ_t ($\mathrm{kg\,m^{-3}}$) respectivement la concentration de particules de neige dans la couche de saltation et dans la couche de suspension, ϕ_s étant obtenue par l'équation 5.45. ψ_s et ψ_t ($\mathrm{s^{-1}}$) sont les coefficients de perte de neige par sublimation respectivement dans la couche de saltation et dans la couche de suspension. Leur méthode de calcul est décrite dans l'annexe I.

5.5.6 Fonte de la neige

La sublimation donne la perte en volume de neige provoquée par la transformation des cristaux de glace de la neige en vapeur d'eau. Néanmoins, la fonte de la neige n'est pas prise en compte dans cette formulation, et nécessite de calculer le taux de fonte de la neige. Plusieurs modèles existent, et par soucis de simplicité dans la paramétrisation, nous choisissons d'utiliser le modèle *Degree-Day Method* [38]. Cette méthode exprime simplement le taux de fonte de la neige en fonction de la différence entre la température moyenne quotidienne, et une température de référence, multipliée par un taux global. L'avantage d'un tel modèle est qu'il expose des paramètres simples pour l'usager afin de représenter le taux de fonte :

$$M = C_M \left(T_a - T_b\right) \ , \tag{5.48}$$

avec M ($\mathrm{mm\,d^{-1}}$) l'épaisseur de neige qui fond quotidiennement, C_M ($\mathrm{mm\,^{\circ}C^{-1}}$) le coefficient de fonte quotidienne, T_a (°C) la température moyenne quotidienne, et T_b (°C) la température de référence, le plus souvent 0°C.

Nous venons de voir comment calculer le taux de fonte de la neige, complétant ainsi la liste des effets que nous simulons pour transporter notre neige. Nous pouvons maintenant évaluer l'équation 5.16 afin d'obtenir le changement d'épaisseur de la neige dans notre domaine comme fonction du vent, de la température et du soleil, et ce, spatialement dans notre domaine. Nous voyons dans le prochain chapitre comment appliquer ces modèles dans le contexte d'une simulation effectuée sur processeur graphique, afin d'obtenir une distribution de la neige dans des scènes arbitraires, avec seulement quelques paramètres exposés à l'utilisateur.

CHAPITRE 6

IMPLÉMENTATION SUR GPU

Dans ce chapitre, nous décrivons notre méthode de simulation du vent, des particules et du transport de la neige sur processeur graphique.

6.1 Pipeline de simulation du transport de neige

Avant de décrire en détail le fonctionnement de chacun des algorithmes, nous survolons la méthode générale de simulation que nous proposons. Elle consiste en une première phase de précalcul, et d'une phase de simulation du transport de la neige, illustrées dans la figure 6.1.

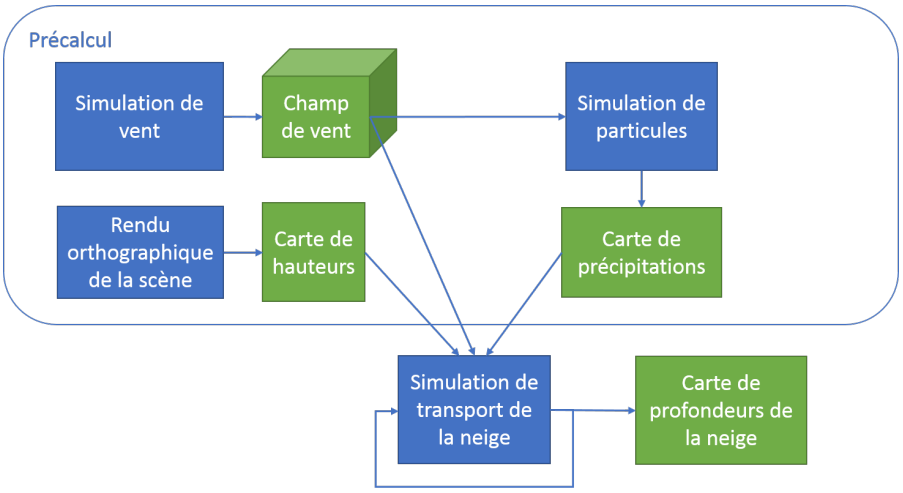


Figure 6.1 – Notre méthode de simulation. En bleu les phases de calcul et en vert les résultats temporaires ou définitifs obtenus de ces phases.

La première phase consiste à simuler le vent dans notre scène avec la méthode décrite dans le chapitre 3. Un champ de vent tri-dimensionnel est alors produit, qui est utilisé dans la simulation des particules et dans la simulation du transport de la neige. La simulation des particules permet de produire une carte de précipitations, information utile dans la simulation du transport de la neige. Le champ du vent ainsi que le carte de

précipitations sont toutes deux reliées, une carte de précipitations étant spécifique à une combinaison vitesse et direction du vent particulière dans une scène. Elles peuvent et devraient donc être calculées d'avance, et être utilisées au moment voulu dans la simulation de la neige. De cette façon, la simulation du transport de la neige est découplée de la simulation du vent et des particules, de sorte que nous pouvons simuler différentes conditions d'enneigement dans notre scène sans devoir tout simuler à nouveau si nous souhaitons obtenir une nouvelle épaisseur de neige dans la scène ; seulement le transport de la neige en tant que tel devra être resimulé, ce qui est peu coûteux en comparaison de la simulation du vent et des particules, comme nous le verrons dans le chapitre 7.

Nous introduisons d'abord brièvement le fonctionnement d'un GPU et des particularités spécifiques à ce type de matériel, puis nous discutons plus en détail notre implémentation de la simulation du vent, des particules et du transport de la neige sur processeur graphique.

6.2 Processeur graphique

Tout d'abord, un processeur graphique est un microprocesseur, soit embarqué dans le processeur central, ou séparé sur une carte dédiée. Il est généralement utilisé pour l'affichage d'images à l'écran, pour faire du rendu de formes en 2D ou en 3D, ou encore pour du calcul scientifique, comme de l'apprentissage profond ou encore afin de produire des crypto-monnaies. La communication avec ces microprocesseurs se fait par le biais d'interfaces (API) pour des applications graphiques comme *OpenGL*, *Vulkan* ou *DirectX*, ou avec des API de calculs parallèles comme *CUDA* ou *OpenCL*, permettant d'exprimer des requêtes bas-niveau au processeur graphique afin qu'il exécute une tâche bien précise, comme par exemple allouer un tampon mémoire pour stocker les sommets et triangles d'un maillage. L'utilisation d'un processeur graphique pour effectuer nos simulations comporte un grand avantage : le nombre possible de fils d'exécutions disponibles. Sur un processeur central classique, huit coeurs possiblement disponibles aux calculs permettent d'exécuter huit tâches en parallèle (en réalité il est possible d'en avoir beaucoup plus, mais le processeur partage le temps d'exécution de chaque coeur pour

pouvoir supporter ces tâches additionnelles), tandis que sur un processeur graphique, plusieurs centaines de coeurs sont disponibles pour exécuter nos calculs. Par exemple sur une NVIDIA GTX 970, 1664 coeurs sont disponibles. Si nos algorithmes sont exécutables en parallèle, nous pouvons alors utiliser un processeur graphique pour accélérer nos calculs. Avant d'introduire notre implémentation de nos modèles de simulation, nous introduisons brièvement plusieurs termes techniques :

Shader Un *shader* (nuanceur) est un programme pour un processeur graphique. Il en existe différents types, comme le *vertex shader*, qui s'occupe de traiter les sommets d'un maillage, ou encore le *fragment shader* qui contrôle l'apparence des triangles du maillage sur les pixels de l'écran.

Texture Une texture est une structure de données représentant plus généralement une image 1D, 2D ou 3D. Elle est stockée dans la mémoire de la carte graphique, pour être utilisée dans des calculs ou dans l'affichage.

Texel Unité fondamentale d'une texture, contient de l'information, le plus généralement une couleur.

Compute Shader Un *compute shader* est un type de *shader* permettant d'exécuter du code plus général avec le processeur graphique. Il est séparé du reste du pipeline graphique, et est utilisé pour des calculs ne traitant pas forcément de géométrie destinée pour de l'affichage.

Pipeline Graphique Le pipeline graphique consiste en plusieurs opérations successives permettant d'afficher de la géométrie à l'écran.

6.3 Simulation de vent sur GPU

Afin de simuler notre vent comme décrit dans le chapitre 3, quelques modifications doivent être effectuées afin de pouvoir le simuler sur un processeur graphique. Tout d'abord, notre grille régulière est représentée comme plusieurs textures 3D de même

résolution $N \times M \times O$ sur la carte graphique, contenant la vitesse, la divergence, la pression, ainsi qu'un booléen indiquant si la cellule est pleine ou non. Nous exécutons un *compute shader* sur ces $N \times M \times O$ cellules dans des fils d'exécutions séparés. Notre algorithme de simulation consiste premièrement en une étape de discrétisation de la scène dans notre texture 3D, puis par la suite d'une boucle principale de simulation effectuant chacune des étapes de notre méthode de simulation du vent, comme illustré à la figure 6.2.

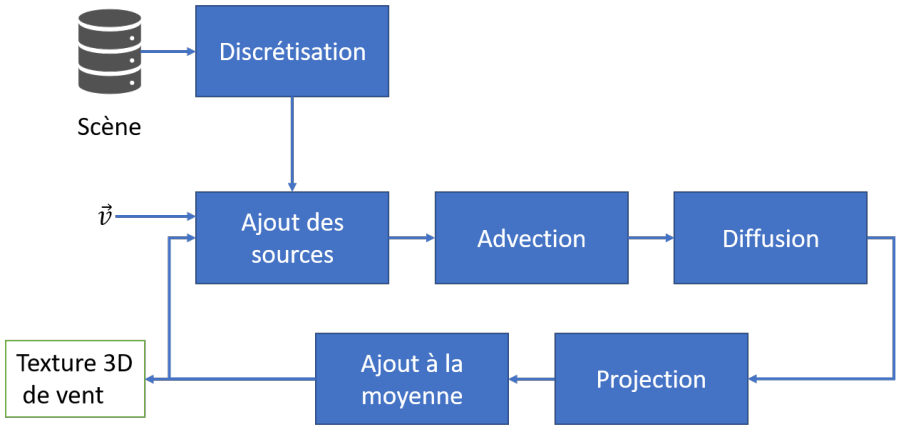


Figure 6.2 – Notre algorithme de simulation de vent.

La discrétisation de la scène se fait une seule fois avant la simulation, et teste l'intersection entre les triangles de notre scène et chacune des cellules de notre grille, comme l'illustre la figure 6.3. Si un triangle intersecte une des cellules, alors cette cellule est considérée occupée, même si réellement le vent pourrait quand même passer au travers de cette cellule. Ce choix de discrétisation permet d'accélérer les calculs sans qu'on n'ait à simuler et tester les collisions avec tous les triangles de la scène à chaque itération. De plus rappelons que notre objectif est d'obtenir le vent moyen dans notre scène, proche d'un flot laminaire, car nous acceptons de simplement considérer des mouvements globaux du vent dans la scène, afin de déplacer notre neige par la suite. Ensuite, nous devons remplir les intérieurs des maillages. Nous simplifions cette tâche non triviale en remplissant toutes les cellules en-dessous d'une cellule occupée. Ce choix est fait tout d'abord pour simplifier cette étape, car beaucoup de cas extrêmes sont possibles,

par exemple dans le cas d'un maillage non étanche. Cela permet d'être cohérent avec la forme de notre domaine de simulation de neige : nous simulons la neige sur un domaine 2D, nous ne prenons donc pas en compte les volumes vides en-dessous des surfaces les plus hautes. Par exemple nous ne simulons pas de la neige sous un pont, donc il ne nous est pas utile de connaître le vent sous celui-ci. En réalité nous devons prendre en compte ce vent, surtout dans le cas où nous souhaiterions simuler le transport de la neige dans une grille 3D, à la place de la simulation en 2D que nous effectuons. C'est une des limitations de notre méthode que nous discutons dans le chapitre 8.

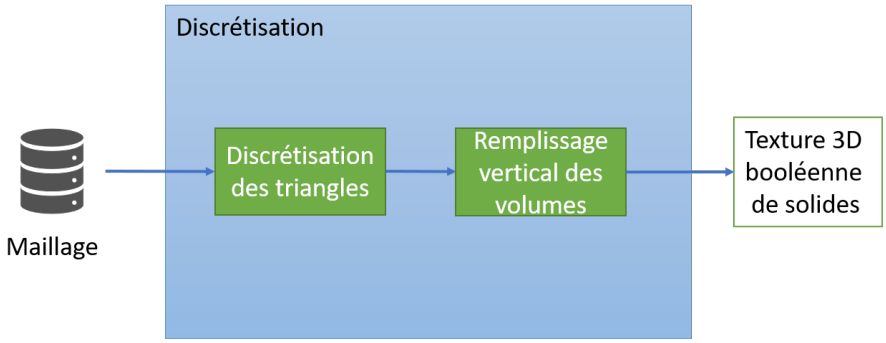


Figure 6.3 – Notre algorithme de discrétisation de la scène.

Une fois la scène discrétisée, nous pouvons simuler notre vent comme décrit dans le chapitre 3 : l'avantage d'utiliser une grille régulière est que nous pouvons effectuer les calculs dans chaque cellule en parallèle, les différents problèmes pouvant être résolus indépendamment dans chacune d'elle. Les systèmes de Poisson faisant partie de la phase de projection et de diffusion sont résolus itérativement avec la méthode de Jacobi. Cette méthode de résolution itérative est définie pour les problèmes de la forme $\mathbf{A}\vec{x} = \vec{b}$, un système linéaire avec :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

On cherche à résoudre pour $\vec{x}$, et la méthode de Jacobi définit la solution à l'itération

$k+1$ pour chaque élément x_i du vecteur $\vec{x}$ comme :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right) . \tag{6.1}$$

Or, dans l'expression de l'équation de la diffusion (3.16) et de la projection (3.24), nous n'avons besoin que des cellules directement voisines avec la cellule pour laquelle nous cherchons à résoudre ces équations. Autrement dit, la matrice **A** est de la forme :

$$A = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
a_{31} & a_{32} & a_{33} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & a_{44} & a_{45} & a_{46} & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & a_{54} & a_{55} & a_{56} & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & a_{64} & a_{65} & a_{66} & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & a_{(n-2)(n-2)} & a_{(n-2)(n-1)} & a_{(n-2)n} \\
0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & a_{(n-1)(n-2)} & a_{(n-1)(n-1)} & a_{(n-1)n} \\
0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & a_{n(n-2)} & a_{n(n-1)} & a_{nn}
\end{bmatrix} ,$$

ce qui signifie qu'elle est creuse. Nous n'avons pas besoin de bâtir la matrice **A** au complet, mais simplement de résoudre les équations 3.16 et 3.24 pour chacune des cellules, en prenant en compte les cellules voisines directes, puisque les cellules plus éloignées ne le sont pas dans la formulation du problème.

Une fois que nous avons simulé une itération de notre vent, nous ajoutons le nouveau vent obtenu dans une texture 3D qui conserve la moyenne des itérations. Nous ne sommes pas intéressés dans le vent immédiat dans la scène, mais par le mouvement global dans le temps. Notre simulation de neige pouvant s'effectuer sur plusieurs jours virtuels, voire semaines, simuler le vent puis les particules à chaque instant est beaucoup trop coûteux pour être envisagé. Vu que nous souhaitons précalculer le champ de vent et

les précipitations de neige, le vent doit être le plus général, et incorporer le moins de phénomènes temporels. C'est dans ce but que nous faisons une moyenne de nos itérations de vent. Ainsi les turbulences s'annulent avec le temps, laissant juste le mouvement général du vent. Malgré tout, la neige déplacée par plusieurs vents successifs turbulents n'aura pas la même distribution que celle qui serait déplacée par la moyenne de ces vents turbulents. Toutefois, cette méthode n'est pas exempte de problèmes non plus, par exemple si nous obtenons entre deux itérations deux vents opposés, la moyenne va annuler le vent, alors que temporellement le vent aurait évolué d'une direction à l'autre, au lieu de simplement s'annuler.

Grâce à notre implémentation, nous obtenons une texture 3D contenant le vent moyen dans notre scène après plusieurs itérations. Actuellement nous laissons un nombre arbitraire d'itérations avant d'arrêter d'accumuler, mais il est parfaitement possible de calculer la variance de la moyenne entre chaque itération et d'arrêter la simulation dès qu'elle est inférieure à un certain seuil de tolérance paramétrable.

6.4 Simulation de particules sur GPU

Avec le vent obtenu, nous devons calculer notre carte de précipitations, en simulant des flocons de neige comme décrit dans le chapitre 4. Pour cela, nous utilisons un *compute shader* qui simule le déplacement de plusieurs flocons en parallèle. Nous exécutons ce code pour simuler plusieurs milliers de flocons à la fois, jusqu'à qu'ils soient tous tombés, c'est-à-dire une fois qu'ils ont atteint une cellule de notre grille 2D de précipitations, ou qu'ils sont sortis du domaine de simulation. L'algorithme de simulation des particules fonctionne comme suit (illustré dans la figure 6.4) : calcul d'une fenêtre d'apparition des particules, simulation de la chute de ces particules dans la scène jusqu'à ce qu'elles aient toutes atterri, calcul d'une première carte de précipitations, rejet des aberrations dans la texture et filtrage.

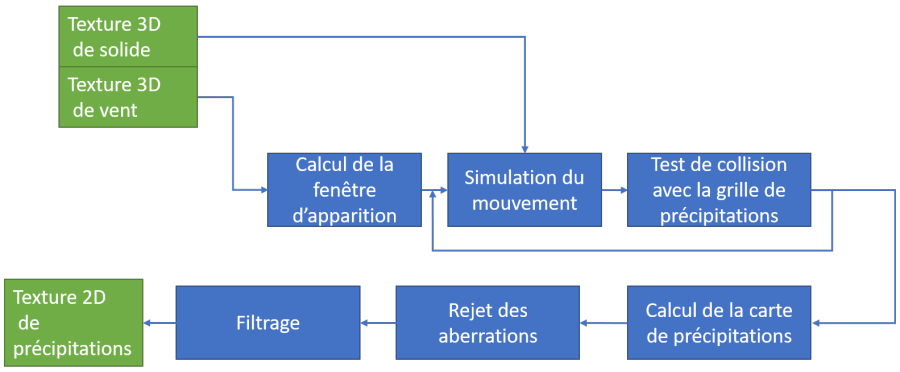


Figure 6.4 – Notre algorithme de simulation de particules.

La première étape est donc de calculer la fenêtre de lancement des particules (figure 6.5). Toutes les particules apparaissent aléatoirement dans cette zone, définie à partir du vent minimum et du vent maximum de notre champ de vent. On calcule l'intersection entre le plan à une hauteur h laissée en paramètre à l'utilisateur et des rayons partant des coins de notre domaine 2D définissant notre carte de précipitations, avec la direction inverse au vent minimum et maximum ajoutée à la gravité (la masse d'un flocon étant prise en compte dans l'expression de la gravité). En testant pour les directions minimum et maximum, nous nous assurons que les particules simulées tombent généralement dans notre scène. Nous gardons le minimum et le maximum dans toutes les directions des intersections entre ces rayons et le plan, et ces quatre points définissent notre fenêtre de lancement. Tout flocon qui apparaîtrait en dehors de cette fenêtre ne pourrait pas intersecter le plan de base de notre scène en suivant notre champ de vent. Malheureusement, de cette manière les particules peuvent aussi tomber en dehors de notre scène, mais nous décidons d'être conservateur, afin qu'elle soit recouverte uniformément dans tous les cas.

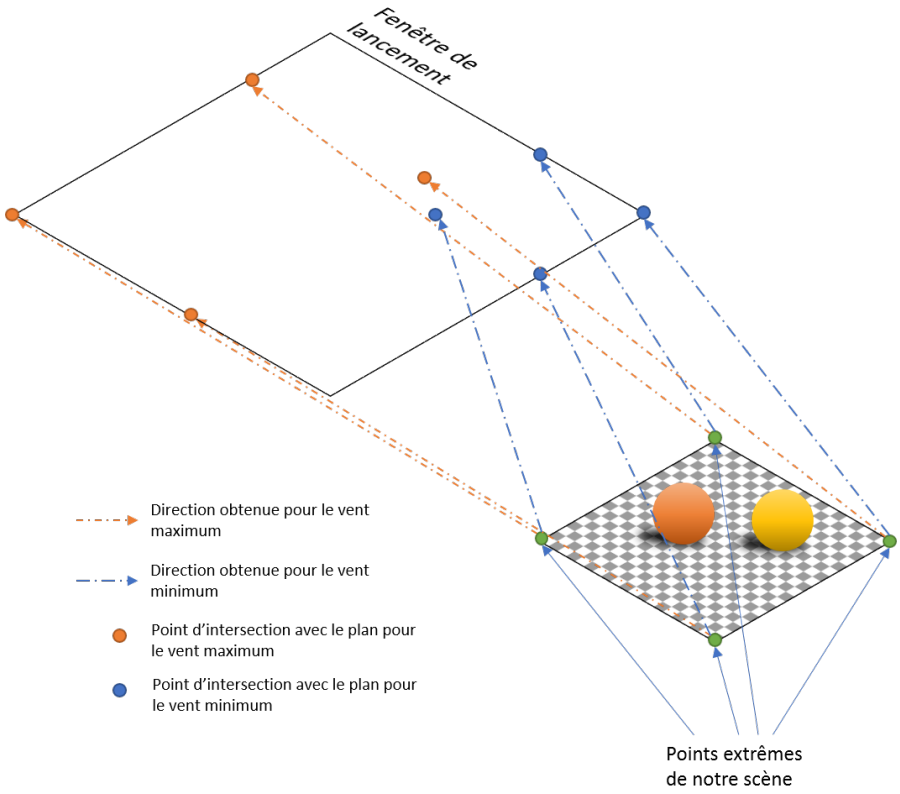
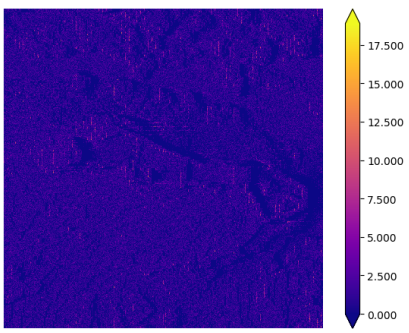
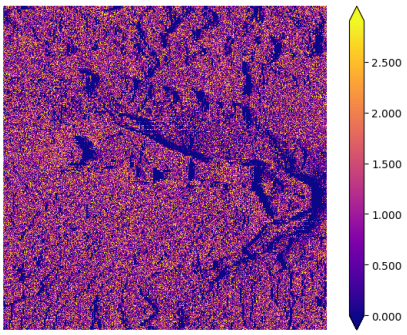


Figure 6.5 – Calcul de la fenêtre de lancement.

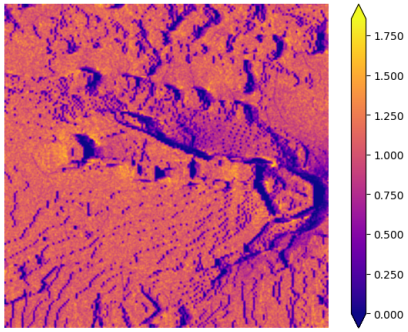
Ensuite, nous faisons apparaître nos particules aléatoirement dans cette zone de lancement. Nous simulons leur mouvement avec un *compute shader*, chaque particule ayant son propre fil d'exécution. Nous testons les collisions comme indiqué dans le chapitre 4. Une fois que le nombre de particules ayant atteint les cellules de notre grille de précipitations est satisfaisant, nous pouvons calculer la carte de précipitations. On convertit le nombre de particules par cellule en pourcentage du nombre total de particules ayant atterri dans toutes les cellules. Ce nombre est normalisé par rapport à la moyenne, de sorte que dans chaque cellule le pourcentage p de particules s'étant rendues dans celle-ci soit dans l'intervalle $[p_{\min}, p_{\max}]$. De cette façon, notre carte de précipitations peut avoir des zones où le ratio de précipitation est supérieur à 100%, permettant de représenter des zones où plus de flocons sont tombés que la moyenne. A ce stade nous obtenons une carte de précipitations qui ressemble à celle présentée dans la figure 6.6a.



(a) Carte de précipitations initiale après simulation



(b) Carte de précipitations avec les aberrations retirées



(c) Carte de précipitations finale après l'application d'un filtre gaussien

Figure 6.6 – Une carte de précipitations (%) à différentes étapes de notre processus de traitement, avec un vent allant de gauche à droite à $11\,\mathrm{m\,s^{-1}}$.

Comme nous pouvons le remarquer, cette carte est très bruitée malgré le nombre de particules (10 millions dans ce cas), dû à la résolution de la carte (1000×1000), ce qui mène à une moyenne de 10 particules par cellule. De plus, la distribution peut être très biaisée vers quelques zones où beaucoup de particules sont tombées par rapport à d'autres. C'est le cas de certaines de nos simplifications, comme par exemple la neige qui frappe un voxel, et qui tombe directement dans la cellule à sa base. Pour pallier plusieurs tels problèmes, nous avons préféré supprimer les distributions qui sont trop éloignées de l'écart interquartile en calculant la distance à celui-ci selon :

$$IQR = 1.5\,(Q_3 - Q_1)\ ,$$

avec Q_1 et Q_3 respectivement le premier et troisième quartile de notre distribution de précipitations. Si la distance à la moyenne de la distribution est plus grande que l'IQR, nous marquons la cellule comme aberrante. Une fois toutes les cellules aberrantes identifiées, nous remplaçons le taux de précipitation par la moyenne des cellules voisines qui ne sont pas aberrantes, si plus de 50% de celles-ci ne le sont pas. Nous procédons itérativement de sorte à propager les valeurs non aberrantes progressivement dans la carte de précipitations. Nous obtenons en résultat une carte mieux distribuée, comme dans la figure 6.6b. Typiquement, en fonction de la résolution de notre carte de précipitations, le pourcentage de cellules aberrantes est de l'ordre de 2 à 3% du nombre total de cellules.

Finalement nous appliquons simplement un filtre gaussien de 11×11 texels afin d'obtenir la carte de précipitations avec un bruit beaucoup plus basse fréquence qu'à l'origine, ce qui est plus souhaitable, car de la neige ne tombe pas parfaitement uniformément dans une scène. Néanmoins il reste nécessaire de simuler suffisamment de particules afin d'obtenir une carte de précipitations moins bruitée.

6.5 Simulation du transport de la neige sur GPU

Avec notre carte de vent 3D et notre carte de précipitations 2D, il ne manque plus qu'à obtenir une carte de hauteurs avant de pouvoir simuler le transport de la neige.

Pour l'obtenir, nous faisons un rendu orthographique de notre scène, comme illustré à la figure 6.7. Par la suite, nous pouvons commencer à simuler notre neige sur GPU. Pour ce faire, nous créons 26 cartes de données différentes, afin de conserver ou de simuler spatialement :

- la hauteur du terrain z
- l'angle de la pente du terrain β
- l'angle azimutal du terrain ε
- l'influence de la courbure du terrain Ω_c
- l'influence de la pente du terrain Ω_p
- les dérivées spatiales en x et en y
- la direction θ et la vitesse W du vent
- l'épaisseur de la neige ζ
- le seuil critique d'érosion u_{*t}
- la densité de la neige ρ_s
- l'érosion sur la surface u_*
- le taux de précipitations P
- le taux maximum de saltation Q_{s_max}
- le taux de saltation Q_s
- la hauteur de la couche de saltation h_*
- la concentration totale du nuage de particules ϕ_t
- le taux de suspension Q_t
- le taux de sublimation Q_v
- les dérivées spatiales de Q_s, Q_t et Q_v

Nous avons besoin de conserver ces informations dans des textures car le calcul de chacune de ces données est soit trop coûteux à effectuer à la volée, soit qu'elle varie spatialement. Notre implémentation n'est pas optimisée complètement pour économiser la mémoire, et il est tout à fait possible de réutiliser des cartes d'informations au fur et à mesure de l'évaluation d'une itération de notre algorithme, certaines données n'étant plus nécessaires comme par exemple l'angle azimutal du terrain une fois que la carte de vent 2D est produite.

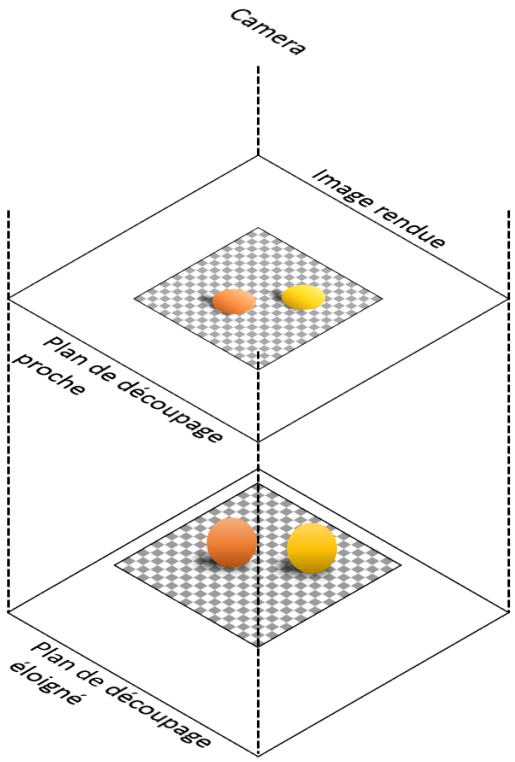


Figure 6.7 – Rendu orthographique de la scène.

Une fois ces cartes de données en main, nous effectuons plusieurs itérations de la simulation afin d'obtenir l'épaisseur de neige après une durée de temps t voulue. Pour cela, plusieurs paramètres sont disponibles à l'utilisateur (listés dans le tableau 6.I). Le système s'occupe d'inférer ou de simuler les autres paramètres du système de simulation. De cette manière, un utilisateur n'a besoin de comprendre que quelques variables simples, comme le taux de précipitations P, le pas de temps Δt ou la durée maximale de simulation t, représentant des données physiques afin de pouvoir simuler l'accumulation de la neige dans ses scènes.

Le taux de précipitation P est le scalaire qui, lors de l'étape de précipitation est utilisé pour multiplier la carte de précipitations afin d'obtenir la quantité de neige qui tombe dans la scène lors d'un pas de temps Δt. z_r est utilisé pour savoir à quelle hauteur à partir du sol nous devons mesurer le champ de vent. Un champ de vent basse résolution pour-

Tableau 6.I – Paramètres exposés à l'utilisateur.

Symbole	Unité	Exemple	Définition
P	$\mathrm{m\,s^{-1}}$	$1.5\,\mathrm{m\,s^{-1}}$	Taux de précipitations de la neige
z_r	m	5 m	Altitude de mesure du vent à partir du sol
γ_p	N/A	0.45	Poids de l'influence de la pente sur le vent
γ_c	N/A	0.55	Poids de l'influence de la courbure du terrain sur le vent
η	$^\circ$	45.5°	Latitude de la scène
RH	%	25 %	Humidité relative de la scène
ν	m	50	Rayon de la courbure moyenne de la scène
τ	$^\circ$	-35°	Angle horaire
σ_c	%	15 %	Pourcentage de couverture nuageuse
d	N/A	1er Janvier	Jour de l'année
Δt	s	300	Pas de temps de simulation
C_M	$\mathrm{mm\,^\circ C^{-1}}$	$1.6\,\mathrm{mm\,^\circ C^{-1}}$	Taux de fonte de la neige
T_b	°C	0.0 °C	Température de référence
C_v	m	0.05 m	Epaisseur de neige conservable dans la végétation
z_{0_veg}	m	0.05 m	Rugosité de la végétation

rait nécessiter de mesurer le vent plus haut, afin de nous assurer que nous ne mesurons pas le vent dans une cellule solide (le vent serait nul dans ce cas), et afin de prendre en compte à quel endroit dans la couche limite nous sommes afin d'exprimer le taux d'érosion. γ_p et γ_c permettent de laisser le choix à l'utilisateur s'il souhaite que le vent soit plus influencé par la pente ou par la courbure du terrain lors du calcul de la carte de vent 2D. ν est utilisé dans le calcul de l'influence de la courbure du terrain sur le vent. η est simplement la latitude de la scène, et est utilisé dans le calcul de la saltation expliqué dans l'annexe I (la longitude n'est pas nécessaire dans ce calcul). De même, le taux d'humidité relatif RH, l'angle horaire ν, le pourcentage de couverture nuageuse σ_c, le jour de l'année d sont utilisés afin de déterminer la quantité de radiation solaire que la neige reçoit sur sa surface, afin de calculer le taux de sublimation. Δt est le pas de temps de simulation. C_M est le taux d'évaporation de la neige. C_v permet de choisir

quelle épaisseur de neige est conservée par la végétation (l'herbe haute conserve un peu de neige par rapport au béton), et z_{0_veg} contrôle la rugosité de cette végétation. Finalement, T_b est la température de référence, le plus souvent 0.0 °C, utilisé dans le calcul de la fonte de la neige. Nous pourrions inférer automatiquement plus de ces paramètres depuis la scène, comme le rayon de la courbure moyenne de la scène ν, mais ce sont les paramètres que nous avons exposés dans l'implémentation actuelle de notre système, et nous n'avons pas établi de méthode pour en inférer d'autres.

De plus, certains de ces paramètres comme par exemple C_v et z_{0_veg} pourraient être appliqués en fonction du type de matériel de la géométrie. Par exemple différencier l'herbe et la terre sur le terrain d'une scène, tout comme le pourcentage de couverture nuageuse sont des données qui pourraient être stockées dans une texture, afin de simuler des éclaircies dans le ciel accélérant la fonte de la neige dans certains endroits de la scène.

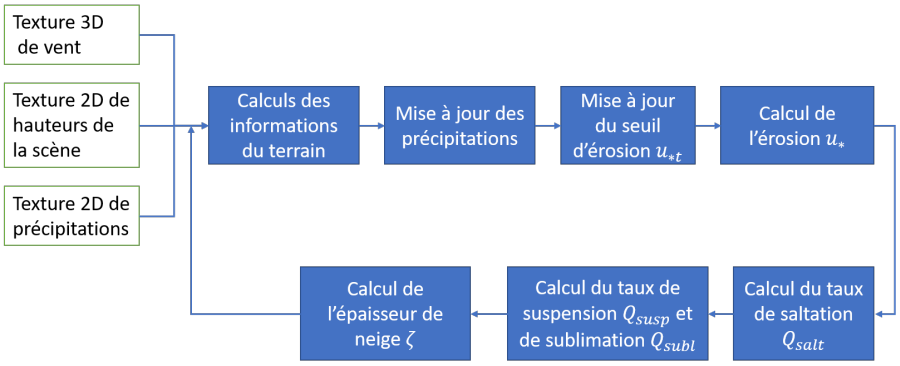


Figure 6.8 – Notre algorithme de simulation de la neige.

Une fois nos textures 2D créées, il suffit de simuler avec le modèle décrit dans le chapitre 5 sur plusieurs pas de temps afin d'obtenir l'épaisseur de neige après une durée de temps t désirée. Pour ce faire, nous exécutons plusieurs *compute shaders* en parallèle sur le GPU afin d'obtenir les résultats de nos équations. Nous procédons en plusieurs phases, comme illustré à la figure 6.8. Nous extrayons les informations du terrain de notre carte de hauteurs, afin d'obtenir notre carte 2D du vent mesuré à la hauteur z_r

depuis le sol dans notre champ de vent 3D obtenu par notre méthode du chapitre 3, modifié en fonction de la courbure et de la pente du terrain. Ensuite nous mettons à jour la carte de précipitations obtenue par la méthode décrite précédemment, que nous multiplions par le taux de précipitations P en paramètre, afin de pouvoir le modifier sans avoir à recalculer une nouvelle carte de précipitations. Par la suite, nous mettons à jour les différentes données stockées dans les textures 2D, notamment le seuil d'érosion critique u_{*t}, la vitesse d'érosion u_* et le taux de saltation Q_{salt}, de suspension Q_{susp} et de sublimation Q_{subl}. Finalement nous calculons les dérivées spatiales avec une différence centrale de la forme :

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+\Delta x,y) - f(x-\Delta x,y)}{2\Delta x} \,, \tag{6.2}$$

pour évaluer le taux de changement spatial de la saltation Q_{salt} et de la suspension Q_{susp} dans l'équation 5.16.

Lors du calcul du taux de suspension Q_{susp} et de sublimation Q_{subl}, nous devons intégrer la concentration de particules de neige en hauteur dans chacune des cellules de notre domaine. Nous résolvons la saltation et la suspension au même moment car les domaines d'intégration définis dans les équations 5.40 et 5.46 se chevauchent. Nous discrétisons notre intégrale comme une somme de Riemann pour obtenir une solution de la forme :

$$\int_0^{z_t} f(x^*,z)dz \approx \sum_{i=0}^{n} f(x^*,z) \times \Delta z \,, \tag{6.3}$$

avec Δz (m) un incrément de hauteur, défini comme :

$$\Delta z = z_t / n \,, \tag{6.4}$$

z_t (m) étant notre limite d'intégration supérieure, et n le nombre d'échantillons qui vont mesurer notre fonction $f(x^*,z)$.

Nous venons de voir les changements nécessaires à une implémentation sur GPU,

permettant d'effectuer nos simulations efficacement et en parallèle en prenant avantage de l'architecture hautement parallèle d'un processeur graphique. Nous obtenons de cette façon un algorithme de simulation du transport de la neige qui permet l'exécution de chacune de ses itérations en temps interactif, comme nous voyons dans le chapitre 7 présentant les résultats ainsi que les performances de notre méthode.

CHAPITRE 7

RÉSULTATS

Dans ce chapitre, nous présentons les résultats obtenus par notre méthode ainsi que ses performances. Tous les résultats ont été produits sur un ordinateur avec un processeur Intel® Core™ i5-4570 cadencé à 3.2 GHz sur 4 coeurs et un processeur graphique Nvidia GTX 970 cadencé à 1050 MHz, avec 4 GB de mémoire vive, et 1664 coeurs Cuda™.

7.1 Résultats

Figure 7.1 – Notre scène "Château" enneigée avec notre méthode, avec une carte d'épaisseurs de neige d'une résolution de 850×850.

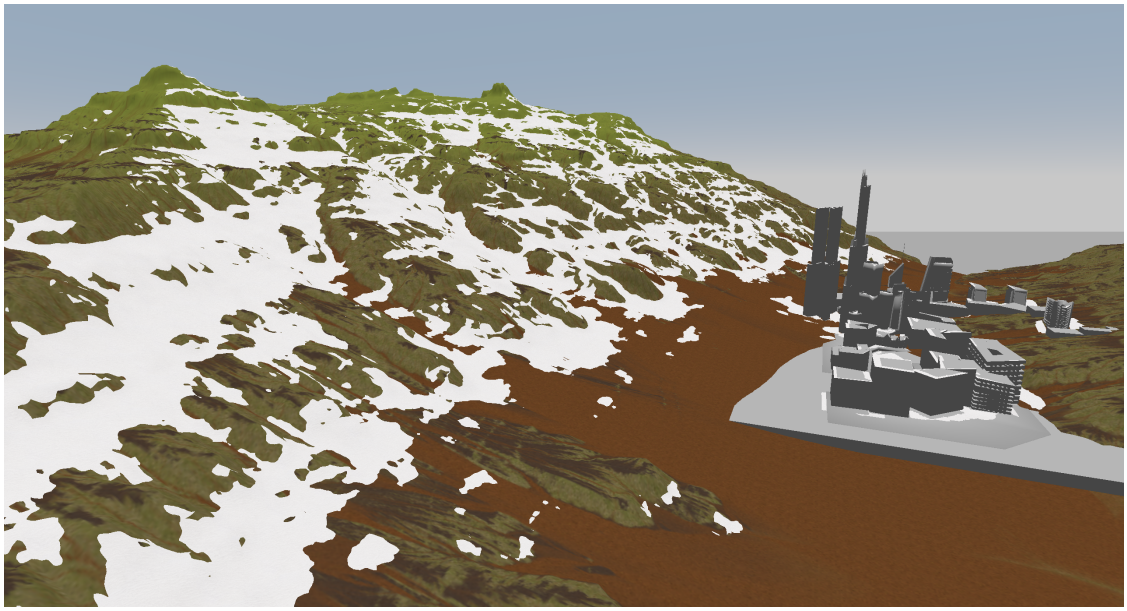


Figure 7.2 – Notre scène "Vallée" enneigée avec notre méthode.

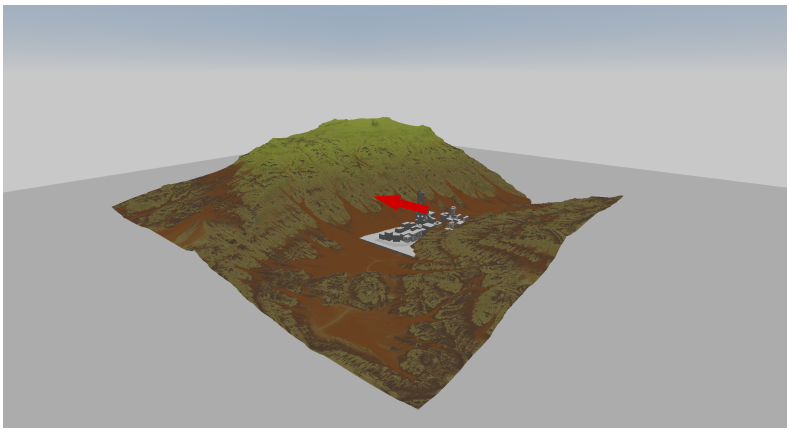
Nous présentons dans cette section les résultats obtenus avec notre méthode. Ils consistent en différentes simulations appliquées sur deux scènes illustrées dans la figure 7.3, avec les paramètres de simulation fournis dans le tableau 7.I. La première scène est un château dans un domaine de petite taille (850 m par 850 m), permettant de tester la robustesse de notre algorithme sur des petits incréments de grille (inférieurs à 1 m) ainsi que la robustesse à la complexité géométrique. La seconde scène est une vallée, avec quelques bâtiments au centre, dans un domaine de simulation de 8 km par 8 km, permettant de tester notre algorithme sur des terrains naturels, avec une complexité géométrique plus douce, et des résolutions de grilles plus élevées. Nous analysons l'influence de la résolution du domaine de simulation dans les résultats, de l'influence de chacun des effets décrits précédemment, plus particulièrement la saltation, la suspension, la sublimation, la fonte, ainsi que l'influence du vent et de sa vitesse et des précipitations dans la scène.

Nous avons effectué un certain nombre de simulations sur ces scènes, afin d'obtenir des distributions de neige qui peuvent être affichées comme l'illustrent les figures 7.1 et 7.2. Pour la suite des résultats, nous présentons les cartes d'épaisseurs de la neige obtenues par notre méthode.

Tableau 7.I – Paramètres utilisés pour nos scènes.

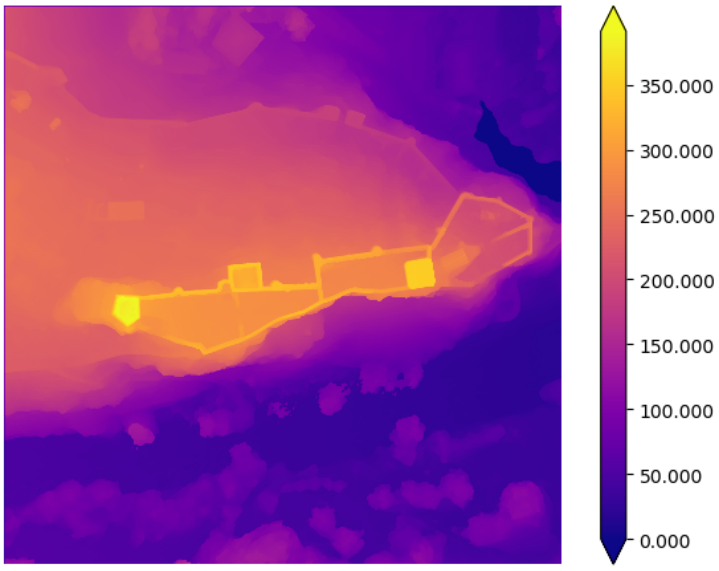
Valeur	Scène	
	7.3a Château	7.3b Vallée
Dimensions (m)	850×850	8000×8000
Résolution vent	$200 \times 200 \times 50$	$400 \times 400 \times 100$
Vent initial ($\mathrm{m\,s^{-1}}$)	(11,0,0)	(0,-10,0)
Précipitations ($\mathrm{cm\,h^{-1}}$)	$0.75\,\mathrm{cm\,h^{-1}}$ pendant 12h, $0\,\mathrm{cm\,h^{-1}}$ ensuite.	
Température (°C)	$-5\,°\mathrm{C}$ pendant 12h, $10\,°\mathrm{C}$ ensuite.	
Humidité (%)	40%	
Couverture nuageuse (%)	0%	
Taux de fonte ($\mathrm{mm\,°C^{-1}}$)	$3.2\,\mathrm{mm\,°C^{-1}}$	

(a) Château

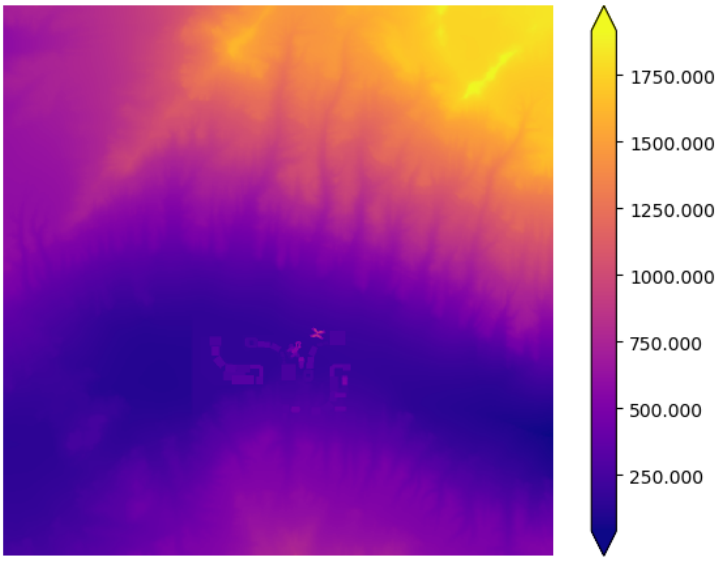


(b) Vallée

Figure 7.3 – Nos différentes scènes sans neige avec la direction du vent illustrée en rouge.



(a) Château



(b) Vallée

Figure 7.4 – Les cartes de hauteurs de nos différentes scènes, en mètres.

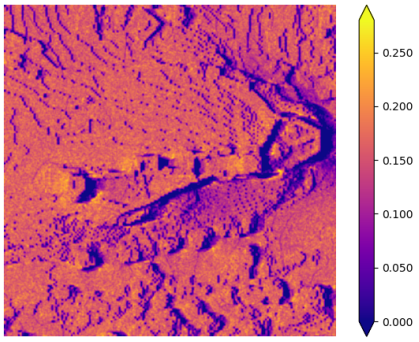
7.1.1 Résolution

Nous avons effectué des simulations à trois résolutions dans notre scène "Château" de la figure 7.3a, afin d'observer l'influence de la résolution, et donc de l'incrément entre chacune de nos cellules. Notre simulation consiste en 12 heures de précipitations de neige avec une température ambiante de $-5\,^{\circ}$C, suivies de 24 heures sans précipitation avec une température ambiante de $10\,^{\circ}$C. Avec ces paramètres, nous pouvons tester les changements dans les résultats pour tous les phénomènes que nous simulons. Nous pouvons voir dans la figure 7.5 le résultat de deux de ces simulations avec trois résolutions de grille différentes. Nous pouvons remarquer que malgré la différence en résolution des trois grilles, respectivement 1680×1680 pour les figures 7.5a et 7.5b, 840×840 pour les figures 7.5c et 7.5d et 168×168 pour les figures 7.5e et 7.5f, les distributions sont pratiquement identiques, la distribution de la neige étant plus diffuse dans les cartes à plus basse résolution, résultat attendu par ce type de méthode. Néanmoins, il reste intéressant de noter que malgré la différence en résolution, les résultats sont extrêmement semblables. Des différences commencent à apparaître lorsque la résolution devient beaucoup moins élevée car la carte de hauteurs comportant moins de cellules, la hauteur se retrouve lissée et l'on perd des détails au fur et à mesure que la résolution est réduite.

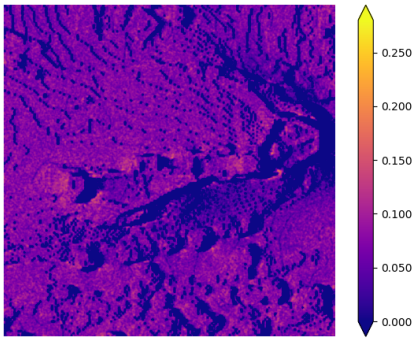
Une résolution plus fine de notre carte de simulation peut néanmoins produire plus de détails dans la carte d'épaisseurs de neige, et pourrait permettre plus des variations que nous ne pourrions obtenir dans une carte plus basse résolution. Le choix de la résolution est donc important et à définir en fonction des effets recherchés. Une carte plus basse résolution de la neige est plus rapide à obtenir comme nous le verrons dans la section 7.2, et pourrait être utilisée pour aider un artiste à prévisualiser ce que ses paramètres vont produire comme distribution de neige dans sa scène.

Figure 7.5 – Comparaison de trois différentes résolutions après deux durées de simulation. Les précipitations de neige se déroulent lors des premières 12 h (à gauche et à droite), puis dans la 2ème simulation (à droite), du vent et une température de 10 °C réduisent l'épaisseur de neige au sol pendant 24 h de plus.

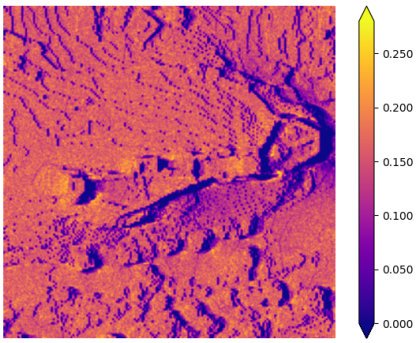
(a) Résolution de 1680×1680, incrément de 50 cm.



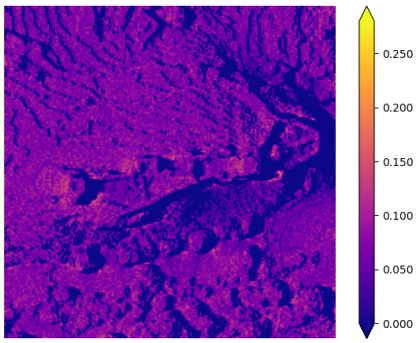
(b) Résolution de 1680×1680, incrément de 50 cm.



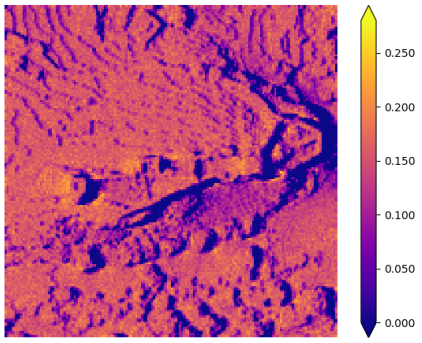
(c) Résolution de 840×840, incrément de 1 m.



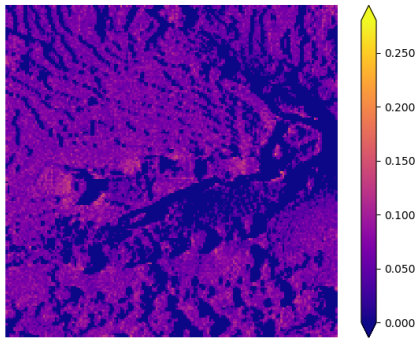
(d) Résolution de 840×840, incrément de 1 m.



(e) Résolution de 168×168, incrément de 5 m.



(f) Résolution de 168×168, incrément de 5 m.



7.1.2 Saltation

Parmi les phénomènes décrits dans notre modèle, le taux de saltation indique à la fois si la neige est déplacée par le vent, mais aussi le taux de suspension et de sublimation. Nous avons effectué une simulation sur notre scène "Vallée" illustrée à la figure 7.3b en activant la contribution de tous les phénomènes simulés ; nous utilisons l'épaisseur de neige obtenue comme référence. Nous avons effectué une autre simulation en ayant désactivé la saltation. Nous illustrons dans la figure 7.6 la différence entre les deux cartes d'épaisseurs, et nous pouvons constater que la quantité de neige dans la scène n'est pas fortement influencée par le taux de saltation, les différences étant de l'ordre du millimètre. La distribution finale est donc très peu modifiée par ce taux. Néanmoins, comme nous l'avons mentionné, la saltation détermine le taux de suspension ainsi que le taux de sublimation. Sa simulation est nécessaire afin de calculer l'influence de ces deux autres effets, que nous allons voir dans la prochaine section 7.2, mais la contribution finale de la saltation elle-même est extrêmement minime avec les conditions de cette simulation. Il serait donc possible de ne pas prendre en compte la saltation dans l'accumulation finale. Néanmoins le coût de cette opération étant moindre, comme nous le verrons dans la section 7.2, nous pouvons prendre en compte sa contribution dans l'épaisseur de neige finale. La faible influence de la saltation confirme aussi les observations de Liston et Sturm [22], qui indiquent que le changement total de l'épaisseur de la neige dû à la saltation n'est que de quelques centimètres sur l'échelle d'une saison hivernale complète. La figure 7.6 illustre l'épaisseur de la neige (m) pour une simulation complète versus une simulation avec la saltation désactivée, après 24 h de simulation suivant les paramètres indiqués au tableau 7.I.

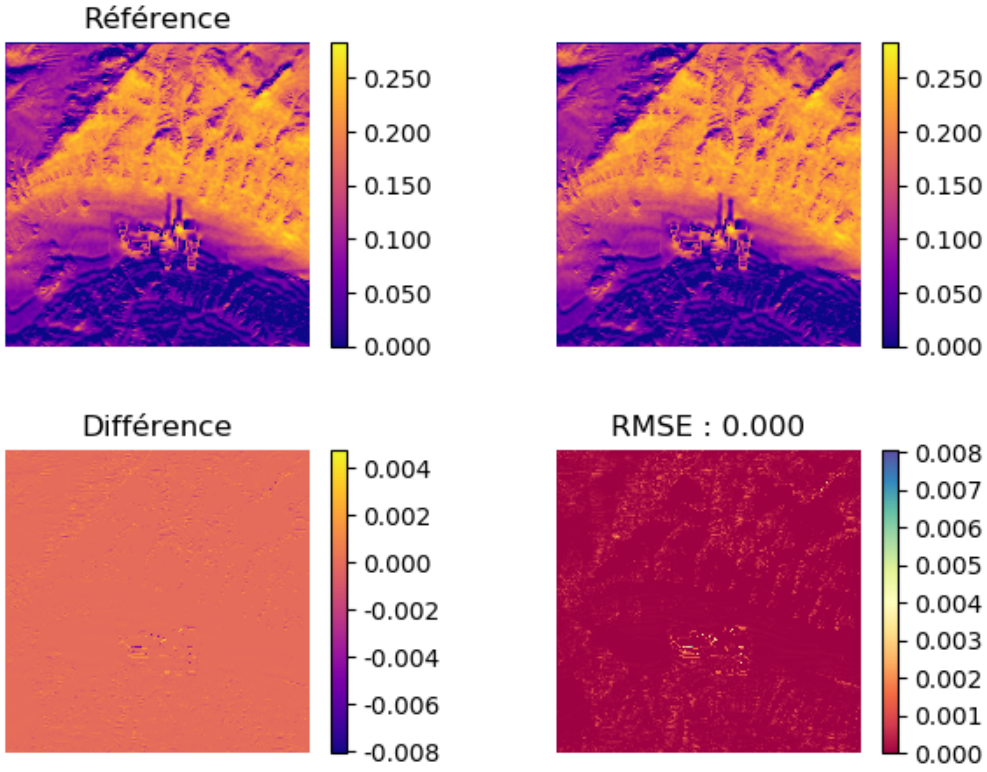


Figure 7.6 – Différence dans l'épaisseur de la neige (m) entre une simulation avec saltation (à gauche) et sans saltation (à droite) dans notre scène "Vallée".

7.1.3 Suspension et sublimation

Comme mentionné précédemment, le taux de saltation détermine aussi la quantité de neige déplacée par suspension et par sublimation. Nous avons effectué comme pour la saltation deux simulations sur notre scène "Vallée", une simulation complète et une deuxième en désactivant la contribution finale de la suspension et de la sublimation dans la carte d'épaisseurs. Dans la figure 7.7 nous présentons la différence entre les cartes d'épaisseurs obtenues après ces simulations. Nous pouvons voir cette fois que la différence est importante, jusqu'à 0.2 m de différence. La déviation quadratique moyenne (RMSE) est très importante, 10% dans cet exemple. Il devient évident qu'une simulation sans suspension ni sublimation finit par accumuler plus de neige qu'elle ne le devrait, la sublimation permettant la transformation de la neige en vapeur, et la suspension dépla-

çant la neige en dehors du domaine de simulation. De plus, nous pouvons aussi remarquer que la différence n'est pas uniforme car des zones de notre domaine n'ont pas été affectées par la suspension ni la sublimation. Ces zones sont plus protégées du vent que les cellules voisines. Du coup elles conservent la neige qu'elles ont accumulée. Nous remarquons aussi dans la figure 7.7 que la suspension et la sublimation retirent la neige sur le versant opposé au vent (la zone en violet, dans la figure de gauche), au contraire de la simulation sans ces deux phénomènes (dans la figure de droite). Sans ces deux phénomènes la neige n'est pas déplacée par le vent et nos distributions seraient moins justes par rapport à la réalité, comme observé par Liston et al. [20, 22].

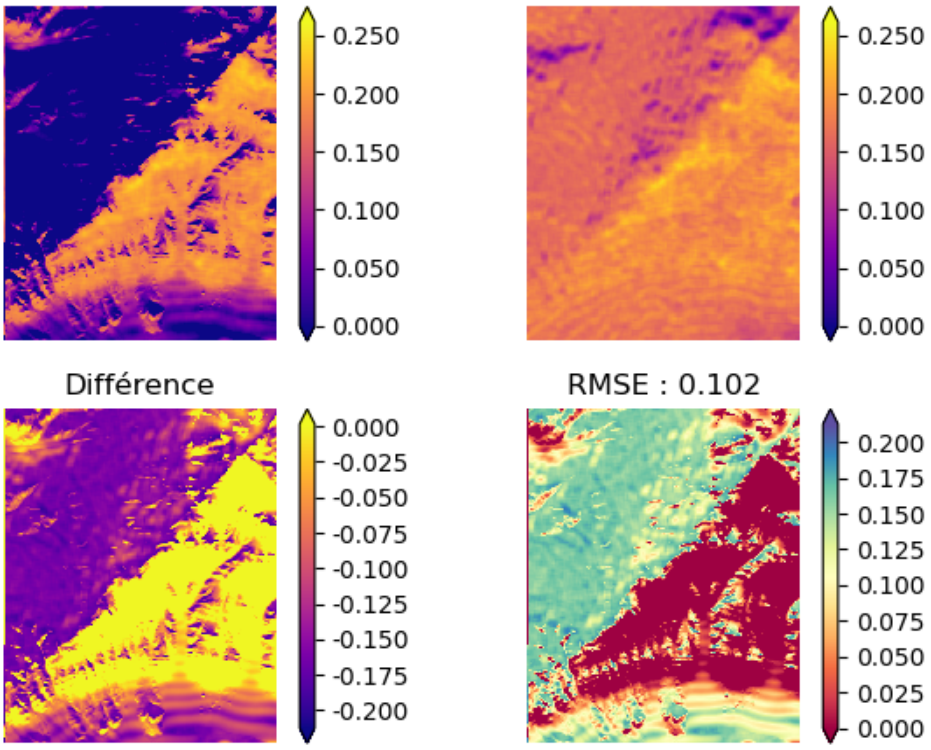


Figure 7.7 – Différence dans l'épaisseur de la neige (m) entre une simulation avec suspension et sublimation (à gauche) et une simulation sans (à droite) dans notre scène "Vallée".

7.1.4 Fonte

La fonte de la neige est le dernier phénomène que nous simulons directement. Il correspond à un taux de fonte uniforme. Dès lors, la différence consiste simplement à une constante, qui accumule la perte de neige avec le temps. Nous avons suivi le même protocole de simulation que pour les comparaisons présentées pour la saltation, la sublimation et la suspension, consistant en deux simulations, une complète et une avec la fonte de la neige désactivée. Nous comparons les deux cartes d'épaisseurs obtenues après ces deux simulations. Comme illustré à la figure 7.8, la fonte de la neige fait une différence moyenne de quelques centimètres entre nos deux simulations. Le taux étant uniforme sur notre domaine, nous n'avons pas de variation spatiale dans ce taux.

Il serait d'ailleurs intéressant de trouver un modèle de fonte de la neige intégrable dans notre système afin de mieux prendre en compte l'ensoleillement, à la place du modèle empirique que nous utilisons pour le moment. Un tel modèle pourrait être celui utilisé par Cordonnier et al. [5] introduit au chapitre 2. De plus, nous pourrions réutiliser plusieurs informations calculées pour la sublimation dans l'appendice I.1 afin de déterminer le taux de fonte.

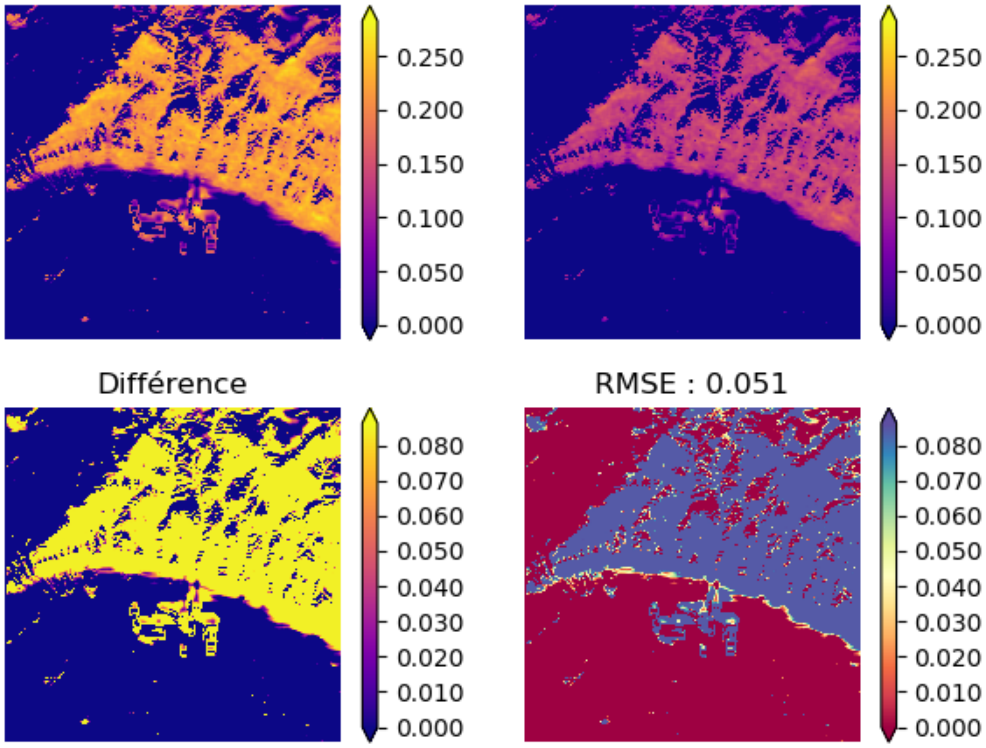


Figure 7.8 – Différence dans l'épaisseur de la neige (m) entre une simulation avec fonte (à gauche) et une simulation sans (à droite) dans notre scène "Vallée".

7.1.5 Vent

Nous pourrions ne pas vouloir précalculer le champ de vent 3D, et simplement utiliser un vent uniforme sur tout notre domaine, et appliquer notre méthode d'extraction des détails du terrain décrite dans la section 5.4 afin d'améliorer les détails de la carte de vent. Comme nous pouvons le voir dans la figure 7.9, l'accélération du vent proche des obstacles est inéxistante et moins détaillée avec le champ de vent uniforme (à droite). La différence se fait beaucoup sentir autour des obstacles, chose que nous pouvions prévoir. Dès lors, le précalcul de notre champ de vent est justifiable car malgré son coût, nous obtenons plus de détails qu'avec un champ de vent uniforme comme nous pouvons l'observer dans la figure 7.9. La déviation moyenne quadratique explose dans ce cas, avec des extrêmes jusqu'à 1400% de différence dans les texels proches des obstacles.

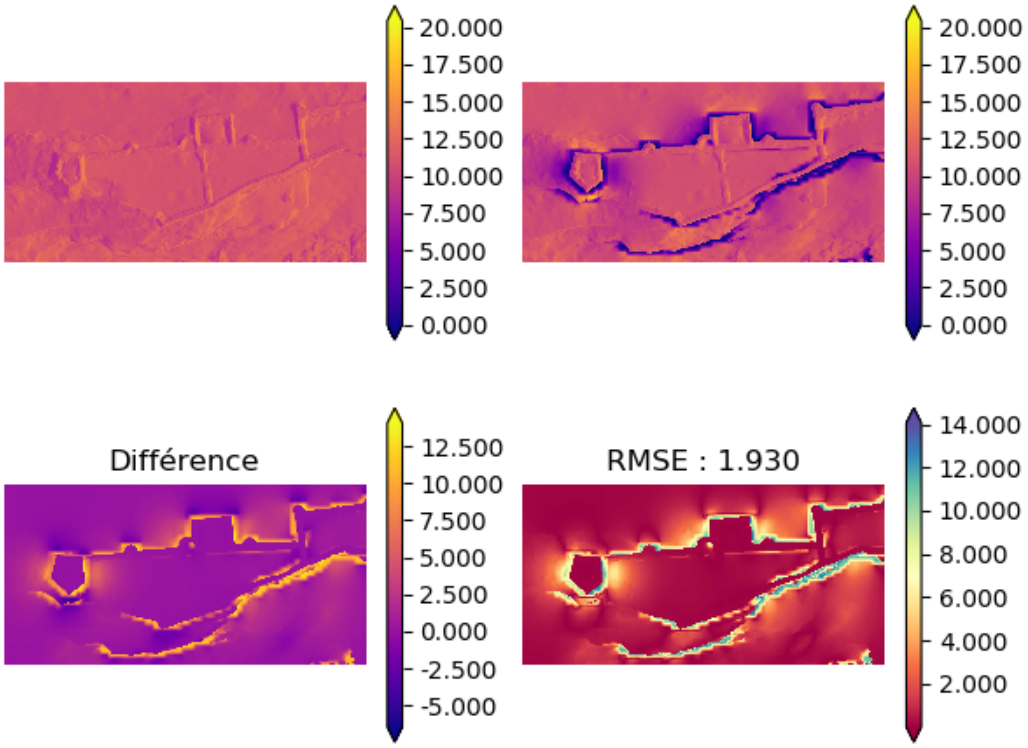


Figure 7.9 – Différence entre la vitesse du vent ($\mathrm{m\,s^{-1}}$) obtenue de notre champ 3D (à gauche) et un vent uniforme (à droite) dans notre scène "Château", pour un vent de $11\,\mathrm{m\,s^{-1}}$ allant de gauche à droite dans notre scène.

7.1.6 Précipitations

Comme pour notre simulation du vent, nous pourrions décider d'utiliser des précipitations uniformes dans notre domaine, au lieu de précalculer notre carte de précipitations. Nous pouvons voir dans la figure 7.10 que comme pour le vent, nous obtenons bien plus de détails dans notre carte d'épaisseurs de neige qu'avec des précipitations uniformes. Nous pouvons observer des différences importantes à droite de la tour et le long du mur de défense en bas à droite de celle-ci. De même que le vent, l'ajout de détails se fait surtout dans les zones proches d'obstacles, et non dans les plaines. Nous pouvons remarquer qu'à l'arrière des obstacles, peu ou pas de neige n'atteint certaines régions. Par contre, nous obtenons de l'aliassage dans notre résultat et plus de bruit, à cause de la génération

de la carte de précipitations. Les résultats approximatifs pourraient être encore améliorés en trouvant une manière d'obtenir l'occultation par la géométrie des précipitations sans aucun processus stochastique comme notre méthode actuelle, comme par exemple avec l'utilisation d'un *Horizon Map* [25].

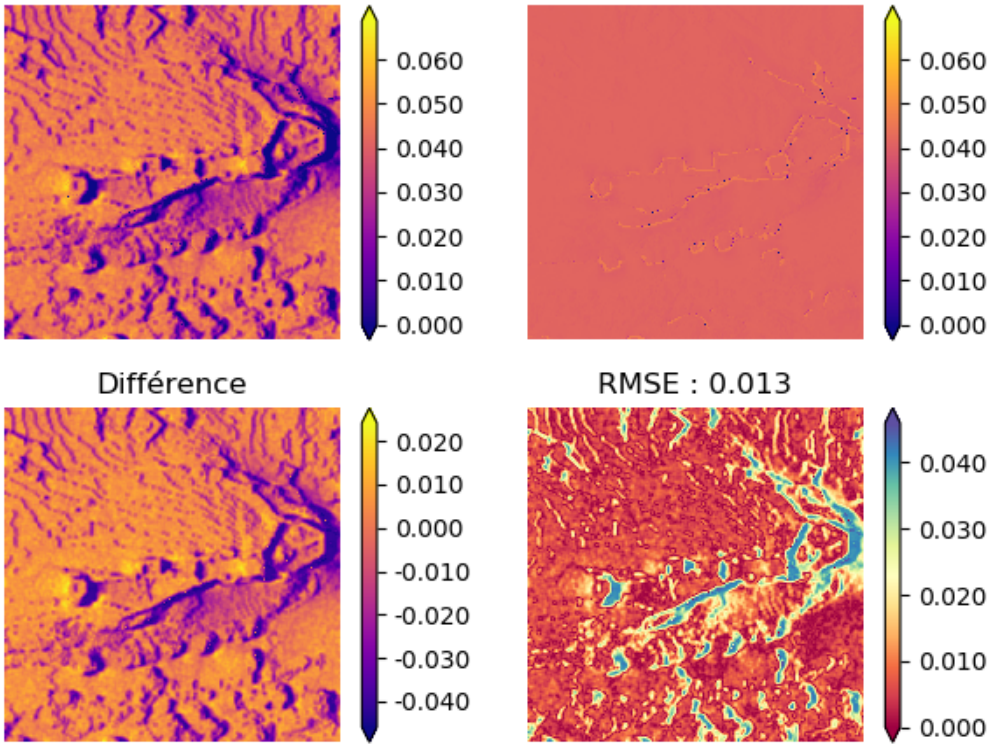


Figure 7.10 – Différence entre l'épaisseur de la neige (m) obtenue avec notre carte de précipitations (à gauche), et avec des précipitations uniformes (à droite) dans notre scène "Château", pour un vent de 11 $\mathrm{m\,s^{-1}}$ allant de gauche à droite dans notre scène.

Nous avons vu les résultats et l'influence de chaque phénomène sur la distribution finale de la neige, et nous pouvons en tirer plusieurs conclusions :

1. Il est apparent que la saltation n'a qu'une très légère influence sur la distribution de la neige, mais que son calcul reste néanmoins nécessaire au calcul des autres phénomènes.

2. La suspension et la sublimation ont la plus grande influence sur la distribution de

la neige, et permettent de redistribuer la neige en fonction de l'exposition au soleil, de la pente du terrain et de l'érosion par le vent.

3. Notre méthode est robuste sous différentes résolutions, nous permettant de simuler à plus basse résolution si nécessaire, sans obtenir des distributions drastiquement différentes de celles à plus haute résolution.

4. Notre méthode de calcul de fonte de la neige permet de prendre en compte l'évaporation dans notre distribution finale. Néanmoins une méthode plus réaliste serait nécessaire afin de permettre une variation spatiale du taux de fonte.

5. Le précalcul du vent nous permet d'obtenir des résultats plus réalistes et convaincants par rapport à un vent uniforme, et nous permet de justifier le coût de ce précalcul, même si nous avions souhaité pouvoir obtenir le champ de vent plus rapidement, comme nous le verrons dans la section 7.2.

6. De la même manière, le précalcul de la carte de précipitations nous permet de prendre en compte l'occultation par la géométrie des précipitations de neige, nous permettant d'obtenir des distributions plus réalistes et plus plausibles.

7.2 Performances

Dans cette section, nous présentons les performances de notre méthode, plus particulièrement lors de la phase de simulation du vent, de simulation des particules, et de simulation du transport de la neige. Nous discutons de l'impact sur les performances de la résolution de la grille de simulation du vent ainsi que celle de la simulation de particules et du transport de la neige. Nous présentons aussi l'impact de chacune des étapes des différents algorithmes de simulation sur les temps de calcul. Nous nous concentrons sur la complexité de chaque algorithme et des étapes les plus coûteuses. Nous discutons d'abord des performances des précalculs nécessaires à la simulation du transport de la neige (la simulation du vent ainsi que la simulation des particules), et justifions le fait de les précalculer dans notre système. Nous discutons ensuite les performances de la simulation de la neige, et expliquons pourquoi cet algorithme peut être considéré comme

interactif. Toutes les mesures de performances présentées sont les temps d'exécution mesurés sur GPU après chaque opération.

7.2.1 Simulation du vent

Afin d'estimer l'impact de la simulation du vent sur les performances, nous avons simulé le même vent avec plusieurs grilles de différentes résolutions, et avons mesuré la durée du temps de calcul après chaque étape de notre algorithme de simulation : après l'ajout des sources, la diffusion, la projection, l'advection et après l'ajout du résultat de l'itération à la moyenne du vent. Comme le montre le tableau 7.II, les deux étapes les plus coûteuses de notre algorithme de simulation du vent sont la diffusion et la projection, qui prennent dix fois plus de temps à résoudre que n'importe quelle autre étape de notre système de simulation. Cela est dû à la résolution des systèmes de Poisson définis plus tôt, la résolution étant faite grâce à plusieurs itérations de l'algorithme de Jacobi. Nous observons que la croissance du temps de simulation pour une itération est linéaire avec la résolution de la grille, comme illustré à la figure 7.11 mais néanmoins cette croissance est très rapide, à un facteur cubique près car l'augmentation d'une des trois dimensions de la grille augmente considérablement le nombre de cellules à traiter. Pour pallier cette croissance de la durée de la simulation, nous simulons notre vent avec une résolution moins élevée sur l'axe vertical, le vent variant moins en altitude qu'au niveau du sol. De cette manière, nous pouvons gagner un peu de temps lors de la simulation, comme l'illustre le temps mesuré pour la grille de résolution $100 \times 100 \times 20$ du tableau 7.II.

Tableau 7.II – Temps de simulation moyen (ms) par itération de la simulation du vent en fonction de la résolution de la grille de simulation.

Résolution	Pipeline de simulation du vent (ms)					TT (ms)
	Ajout des sources	Diffusion	Projection	Advection	Ajout à la moyenne	
16×16×16	0.034	1.246	3.193	0.025	0.015	4.51
32×32×32	0.135	8.119	20.616	0.135	0.070	29.08
100×100×20	0.787	48.300	120.847	0.863	0.411	171.21
64×64×64	0.999	61.953	156.638	1.116	0.497	221.21
128×128×128	8.49	492.221	1308.169	9.435	4.885	1823.20
256×256×256	60.325	4031.051	10183.842	75.979	31.499	14382.70

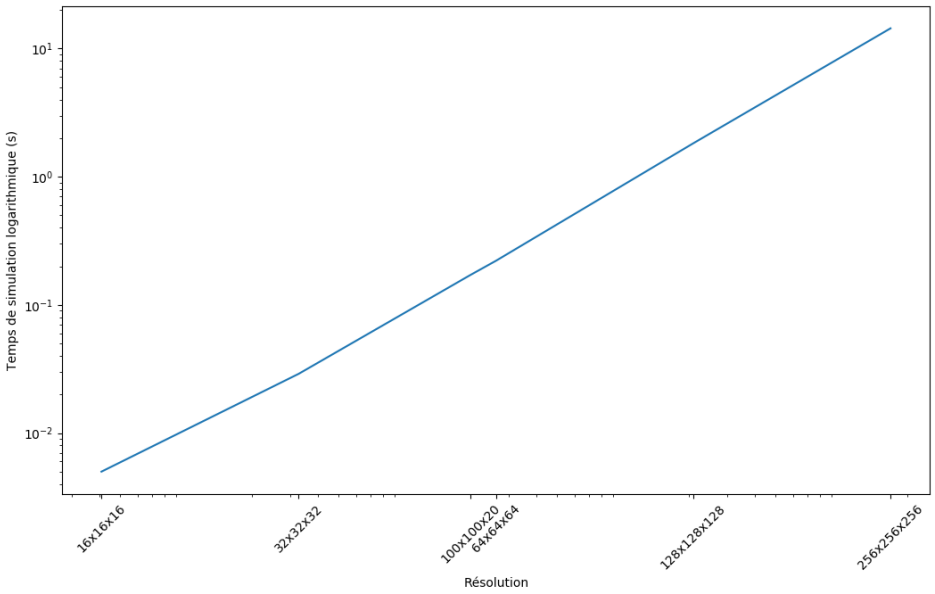


Figure 7.11 – Courbe du temps de simulation du vent en fonction de la résolution dans un repère log-log.

Le champ de vent résultat est stocké dans la mémoire du GPU comme une texture 3D au format *RGBA32F*, c'est-à-dire 4 canaux d'informations contenant pour chaque texel un nombre à virgule flottante sur 32 bits. Le coût mémoire d'une telle texture est donc le nombre de texels dans celle-ci $N \times M \times O$, multiplié par 4 puis par 32. Par exemple, la mémoire occupée pour un champ de vent de résolution 100×20×100 est de

3.84 MB. Notons que nous pourrions utiliser une texture 3D avec seulement 3 canaux d'informations, mais leur création sur GPU n'est pas possible car par conception les processeurs graphiques allouent de la mémoire pour les textures 3D seulement par blocs de puissance de 2, toute texture au format RGB est implicitement convertie vers le format RGBA.

7.2.2 Simulation des particules

La simulation de particules, au contraire de la simulation de vent, a une croissance du temps de simulation linéaire, comme illustré à la figure 7.12. Le nombre de particules nécessaires est relié à la densité de particules souhaitée dans le domaine, et dans la majorité des cas, un million de particules sont amplement suffisantes pour obtenir une carte de précipitations assez détaillée.

Tableau 7.III – Temps de simulation (ms) des particules pour différentes quantités de particules.

Nombre de particules	Temps de simulation (min :sec :ms)
10 000	00 :00 :932
100 000	00 :08 :683
500 000	00 :43 :354
1 000 000	01 :30 :599
2 000 000	02 :53 :109
4 000 000	05 :43 :400
6 000 000	08 :31 :189
8 000 000	11 :19 :154
10 000 000	14 :02 :249

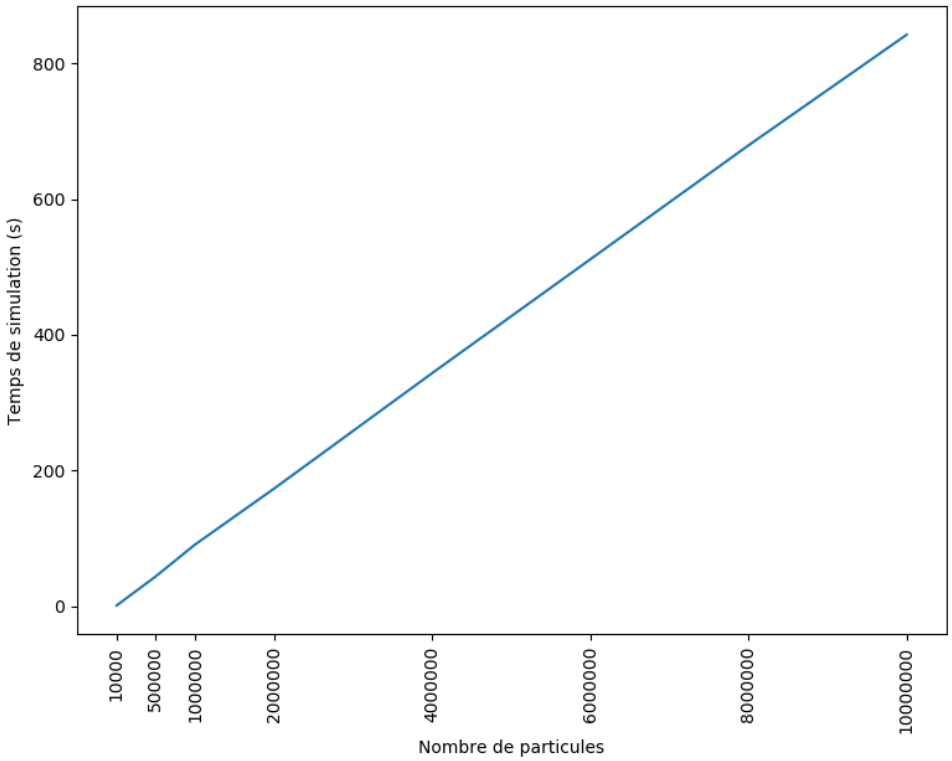


Figure 7.12 – Courbe du temps de simulation des particules en fonction de leur nombre.

7.2.3 Simulation du transport de la neige

Une fois les précalculs effectués, nous simulons notre neige avec la méthode présentée aux chapitres 5 et 6. Notre algorithme consiste en plusieurs phases, dans cet ordre :

1. Mise à jour des informations du terrain
2. Mise à jour des précipitations
3. Mise à jour du seuil d'érosion $u_* t$
4. Calcul de l'érosion u_*
5. Calcul du taux de saltation Q_{salt}

6. Calcul du taux de suspension Q_{susp} et de sublimation Q_{subl}

7. Calcul de l'épaisseur de neige ζ

Comme nous le voyons dans le tableau 7.IV, l'étape la plus coûteuse est le calcul des taux de suspension Q_{susp} et de sublimation Q_{subl}. Cela est dû à l'intégration de la concentration des particules de neige définie par les équations 5.40 et 5.46, nous obligeant à effectuer de multiples pas d'intégration afin de résoudre la somme de Riemann donnée par l'équation 6.3. Le taux de croissance du temps de simulation du transport de la neige illustré à la figure 7.13 est lui aussi linéaire en fonction de la résolution de la grille de simulation, avec un facteur quadratique étant donné que notre grille de simulation est bidimensionnelle. Les temps de simulation d'une itération de notre méthode de transport de la neige permettent de la considérer comme une méthode interactive, c'est-à-dire que nous pouvons simuler à jour l'épaisseur de notre neige dans une scène sans compromettre intégralement l'utilisabilité du logiciel de rendu à côté, et nous permet de visualiser la progression de l'épaisseur de la neige dans notre scène, sans devoir forcément la précalculer. Il peut être souhaitable dans certains contextes de vouloir simuler le transport de la neige pour une saison complète, et d'utiliser la carte d'épaisseurs de neige obtenue après la simulation pour l'affichage, sans visualiser la progression de la couche de neige dans la scène interactivement. Notre méthode ne permet pas d'obtenir de carte d'épaisseurs de neige à un moment précis dans le temps en une seule itération, il est nécessaire d'en effectuer plusieurs afin d'obtenir la distribution demandée. Mais en affichant l'épaisseur de la neige obtenue après chaque itération dans un moteur de visualisation nous pouvons obtenir un *timelapse* de la neige dans une scène.

Hormis les calculs de la suspension et de la sublimation qui sont relativement coûteux par rapport aux autres étapes de notre simulation, les autres temps restent parfaitement raisonnables. En effet, cette étape prend en général 50% de notre temps de simulation. Il serait intéressant de trouver une manière d'accélérer ce processus afin de réduire le temps de simulation.

Tableau 7.IV – Temps de simulation moyen (ms) par étape pour 3 jours de simulation de la neige à plusieurs résolutions dans la scène "Château".

Résolution	Pipeline de simulation de la neige (ms)							TT (ms)
	Terrain	P	u_{*t}	u_*	Q_{salt}	Q_{susp} et Q_{subl}	ζ	
100×100	0.935	0.060	0.045	0.188	0.337	1.410	0.224	3.198
200×200	3.959	0.180	0.145	0.713	0.799	5.449	0.803	12.048
50×500	10.800	0.972	0.826	4.434	5.139	33.849	3.656	59.677
1000×1000	51.101	3.857	3.332	17.655	17.646	135.087	13.194	241.870
2000×2000	176.586	15.044	12.868	70.489	64.825	540.323	48.697	928.833

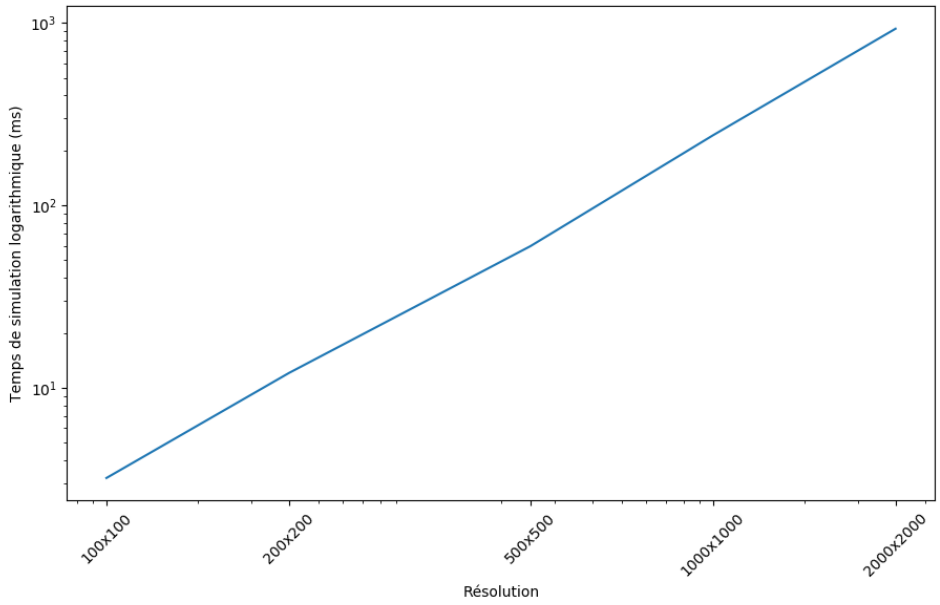


Figure 7.13 – Courbe de croissance du temps total de simulation de la neige en fonction de la résolution.

Notre méthode de simulation du transport de la neige nécessite l'utilisation de 26 textures 2D afin de contenir les informations qui varient spatialement. Chaque texture contient un seul canal d'informations, au format *R32F*, c'est-à-dire un nombre à virgule flottante par texel. La mémoire occupée pour une texture de résolution $N \times M$ est donc de $N \times N$ multiplié par 32 bits. La taille totale de l'information stockée sur la mémoire du GPU est donc vingt-six fois cette quantité. Par exemple, pour une grille de résolution

1024×1024, la mémoire totale occupée est de 151.99 MB. La mémoire utilisée par notre méthode est moyennement élevée, dans ce cas précis nous utilisons 7.5% de la place disponible sur un processeur graphique avec 2 GB de mémoire. Toutefois il est possible de conserver seulement la carte d'épaisseurs de la neige pour afficher la neige dans une scène et nous n'avons pas besoin de garder les vingt-cinq autres cartes si nous ne continuons pas la simulation par la suite. Dans ce cas, le coût mémoire n'est que de 4.19 MB pour une carte de résolution 1024×1024.

Nous venons de présenter les résultats ainsi que les performances obtenues par notre méthode de simulation. Nous avons démontré l'intérêt de l'utilisation d'un modèle physique pour obtenir des résultats réalistes et convaincants, l'intérêt et la nécessité de notre phase de précalculs pour le réalisme des résultats. La phase de simulation du transport de la neige peut être effectuée en temps interactif pour chaque itération de l'algorithme, et malgré le temps plus conséquent de simulation pour les précalculs, cette phase est entièrement découplée de la simulation du transport de la neige, nous permettant d'effectuer cette phase seulement pour chaque direction de vent souhaitée par l'utilisateur dans une scène. Que ce soit la saltation, la suspension, la sublimation, le vent, les précipitations ou la fonte de la neige, chaque effet que nous simulons apporte à la fois de la justesse par rapport au phénomène réel, ainsi que du réalisme dans les distributions de neige obtenues. Néanmoins, nous avons certaines limitations que nous discutons dans le chapitre 8.

CHAPITRE 8

CONCLUSIONS ET TRAVAUX FUTURS

La simulation de l'enneigement de scènes est un problème complexe auquel plusieurs solutions ont tenté de répondre. Pouvoir enneiger des scènes pour le cinéma, ou pouvoir visualiser et prévoir des avalanches sont d'un intérêt crucial, qui nécessitent encore du travail manuel afin d'obtenir des résultats satisfaisants.

Dans ce mémoire, nous avons tout d'abord discuté des différentes méthodes existantes sur la simulation de la neige dans des scènes virtuelles en infographie. Nous avons rappelé la théorie de la simulation de fluides, principalement du point de vue Eulérien, ainsi qu'un modèle physique afin de simuler des précipitations dans un flot de vent. Par la suite nous avons introduit et adapté un modèle de simulation de transport de la neige employé en glaciologie, permettant de simuler différents phénomènes affectant le transport de la neige par la saltation, la suspension, la sublimation et la fonte de la neige. Nous avons implémenté sur processeur graphique ces différents modèles et proposé un algorithme de simulation de la neige dans des scènes arbitraires en précalculant une simulation de vent et une carte de précipitations obtenue par une simulation de particules. Finalement nous avons présenté les résultats obtenus avec cette méthode, ainsi que discuté des performances d'une telle méthode.

Notre approche à la simulation du transport de neige est innovatrice en infographie de par l'utilisation pour la première fois dans les méthodes à base de carte de hauteurs d'un modèle physique plus sophistiqué du transport de la neige, ce modèle ayant été validé sur le terrain en glaciologie. De plus, notre méthode prend en compte l'occultation des précipitations de neige par la géométrie et le vent, nous permettant de calculer des cartes de précipitations selon plusieurs aspects plus réalistes que les autres méthodes jusque là envisagées pour le temps réel. Finalement, malgré la complexité du modèle de simulation de neige, nous exposons à l'utilisateur seulement quelques paramètres faciles à comprendre comme l'heure de la journée, la position de la scènes en coordonnées géo-

graphiques (latitude et longitude), et des paramètres variables dans le temps comme le taux de précipitation, ou encore la température de l'air dans la scène.

En résumé, nos contributions sont :

1. Un modèle physique de transport de la neige simple à contrôler et basé sur des cartes de hauteurs.
2. Un système de calcul de l'occultation de la neige par la géographie, dont les bâtiments, et selon le vent simulé.
3. Une implémentation sur processeur graphique.

8.1 Travaux futurs

Notre proposition n'est qu'un premier pas vers de nouvelles méthodes de simulation de la neige en infographie. Malgré ses avantages, elle comporte aussi des inconvénients, qui peuvent empêcher sa mise en application dans différents milieux, par exemple le domaine du jeu vidéo, où la présence de précalculs dans une méthode est en général à éviter. Notamment, notre méthode ne permettant pas de calculer le vent ni la carte de précipitations dynamiquement, le changement de la direction du vent nécessite le calcul d'un nouveau champ de vent ainsi que d'une nouvelle carte de précipitations, ajoutant du temps de précalcul. Il serait intéressant de trouver une méthode de génération de vent qui soit assez efficace et ne nécessite pas plusieurs itérations d'un algorithme de simulation de vent. Une méthode analogue à la méthode *Curl-noise for procedural fluid flow* de Bridson et al. [3] permettrait peut-être de générer un champ de vent 2D basé sur un générateur de bruit de Perlin non compressible qui respecte les conditions aux bords, de manière efficace en simplifiant les équations de Navier-Stokes.

Dans notre contexte de champ de vent invariant dans le temps, utiliser une telle méthode pour simuler le vent pourrait permettre de gagner en performance et potentiellement ne pas nécessiter de précalculer le champ de vent, qui pourrait être calculé à la

volée quand l'utilisateur demande le changement de sa direction dans notre méthode. Pour l'instant, notre champ de vent n'est qu'une moyenne, éliminant de ce fait les turbulences qui peuvent affecter la distribution de la neige en introduisant plus de variations locales. Notre le vent est simulé qu'à une seule échelle, de ce fait les distributions près des surfaces ou entre des détails fins ne sont pas affectés.

Pour l'instant, le vent n'est pas affecté par l'accumulation de la neige. Nous pourrions resimuler le vent avec la nouvelle couche de neige obtenue traitée comme de la géométrie, mais procéder de cette façon entraîne un coup considérable en temps et en mémoire, tout en brisant l'indépendance de la simulation de la neige et de la carte de vent.

De plus, la simulation des particules reste coûteuse pour obtenir une carte 2D de précipitations. Trouver un algorithme permettant d'estimer efficacement l'occultation par la géométrie et le vent dans un domaine 2D sans avoir recours à une méthode à base de particules permettrait de supprimer cette étape du précalcul. De plus un tel estimateur serait applicable en dehors du contexte de la simulation de neige, par exemple dans l'estimation de la pollution sur des bâtiments.

Notre méthode ne supporte pas d'interactions entre des objets de la scène et la neige. Nous ne pouvons pas faire marcher un personnage dans notre neige et voir les empreintes de pas, ni déplacer une boîte et faire tomber la neige qui s'était accumulée dessus. L'utilisation de carte de hauteurs spécialisées ou par objet permettrait d'appliquer des empreintes de pas comme le fait la méthode de Sumner et al. [43], et de pouvoir de cette manière interagir avec la neige dans notre scène.

Par ailleurs, afin d'améliorer les performances, il aurait été intéressant de pouvoir séparer notre domaine en sous-domaines et effectuer les simulations localement, en prenant en compte tout problème de discontinuité qui pourrait apparaître, comme des discontinuités dans la neige accumulée entre deux zones. Cela permettrait de pouvoir simuler le

transport de la neige sur des domaines beaucoup plus vastes, en effectuant le calcul sur des systèmes distribués.

Il aurait été intéressant de pouvoir proposer un modèle d'apparence de la neige efficace, permettant l'affichage de notre surface de neige adaptativement en fonction de la distance de la caméra, l'angle de la surface, et obtenir des effets visuels comme la transmission sous-surface de la lumière, voir les cristaux sur la surface de la neige au plus près, etc. Un tel modèle pourrait être utilisé pour afficher la surface de neige produite par notre méthode. Nous avons produit un premier modèle ad hoc, qui a permis de constater l'ampleur de la tâche avant de nous concentrer exclusivement sur les distributions de neige.

Malgré ses limitations, notre méthode demeure un premier pas vers des techniques de simulation interactive du transport de la neige basées sur un modèle physique, permettant l'automatisation de l'ajout de neige dans des scènes arbitraires, et de simuler des conditions de neige dans une scène sans modification de celle-ci ou travail manuel.

BIBLIOGRAPHIE

[1] Eric Andrew Anderson. A point energy and mass balance model of a snow cover. *NOAA Tech. Rep. NWS*, 19, 01 1976.

[2] R. Bridson. *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015. URL `https://books.google.ca/books?id=7MySoAEACAAJ`.

[3] Robert Bridson, Jim Houriham et Marcus Nordenstam. Curl-noise for procedural fluid flow. Dans *ACM Trans. Graph.*, volume 26, page 46. ACM, 2007.

[4] Nuttapong Chentanez et Matthias Müller. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph.*, 30:82 :1–10, 2011.

[5] Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes et Marie-Paule Cani. Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches. *Computer Graphics Forum*, 37(2):497–509, 2018.

[6] François Dagenais, Jonathan Gagnon et Eric Paquette. An efficient layered simulation workflow for snow imprints. *The Visual Computer*, 32(6-8):881–890, 2016.

[7] Glen E. Liston, Chris Polashenski, Anja Rösel, Polona Itkin, Jennifer King, Ioanna Merkouriadi et Jari Haapala. A distributed snow evolution model for sea ice applications. *Journal of Geophysical Research : Oceans*, 2018.

[8] Richard Essery, Long Li et John Pomeroy. A distributed model of blowing snow over complex terrain. *Hydrological Processes*, 13(1415):2423–2438, 1999.

[9] Paul Fearing. Computer modelling of fallen snow. Dans *Proceedings of SIGGRAPH*, pages 37–46. ACM Press/Addison-Wesley, 2000.

[10] Bryan E Feldman et James F O'Brien. Modeling the accumulation of wind-driven snow. Dans *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, page 218. ACM, 2002.

[11] Niels v. Festenberg et Stefan Gumhold. Diffusion-based snow cover generation. Dans *Computer Graphics Forum*, volume 30, pages 1837–1849. Wiley, 2011.

[12] Robert G. Fleagle et Joost A. Businger. *An introduction to atmospheric physics*, volume 25. Academic Press, 1981.

[13] Francois Grosbellet, Adrien Peytavie, Éric Guérin, Eric Galin, Stéphane Mérillou et Bedrich Benes. Environmental objects for authoring procedural scenes. Dans *Computer Graphics Forum*, volume 35, pages 296–308. Wiley, 2016.

[14] Francis H. Harlow et J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8 (12):2182–2189, 1965. URL https://aip.scitation.org/doi/abs/10.1063/1.1761178.

[15] V.M. Kotlyakov. Results of a study of the processes of formation and structure of the upper layer of the ice sheet in eastern antarctica. *Colloque sur la glaciologie antarctique*, 55:88–99, 1961.

[16] Lang Wah Lee. *Sublimation of snow in turbulent atmosphere*. Thèse de doctorat, University of Wyoming, 1975.

[17] Glen E. Liston. Local advection of momentum, heat, and moisture during the melt of patchy snow covers. *Journal of Applied Meteorology*, 34(7):1705–1715, 1995.

[18] Glen E. Liston. Interrelationships among snow distribution, snowmelt, and snow cover depletion : Implications for atmospheric, hydrologic, and ecologic modeling. *Journal of Applied Meteorology*, 38(10):1474–1487, 1999. URL https://doi.org/10.1175/1520-0450(1999)038<1474:IASDSA>2.0.CO;2.

[19] Glen E. Liston et Kelly Elder. A meteorological distribution system for high-resolution terrestrial modeling. *Journal of Hydrometeorology*, 7(2):217–234, 2006. URL https://doi.org/10.1175/JHM486.1.

[20] Glen E. Liston, Robert B. Haehnel, Matthew Sturm, Christopher A. Hiemstra, Svetlana Berezovskaya et Ronald D. Tabler. Simulating complex snow distributions in windy environments using snowtran-3d. *Journal of Glaciology*, 53(181):241–256, 2007.

[21] Glen E. Liston et Christopher A. Hiemstra. A simple data assimilation system for complex snow distributions. *Journal of Hydrometeorology*, 9(5):989–1004, 2008. URL https://doi.org/10.1175/2008JHM871.1.

[22] Glen E. Liston et Matthew Sturm. A snow-transport model for complex terrain. *Journal of Glaciology*, 44(148):498–516, 1998.

[23] William E. Lorensen et Harvey E. Cline. Marching cubes : A high resolution 3D surface construction algorithm. Dans *Proceedings of SIGGRAPH*, pages 163–169. ACM, 1987. URL http://doi.acm.org/10.1145/37401.37422.

[24] Frank Losasso, Frédéric Gibou et Ron Fedkiw. Simulating water and smoke with an octree data structure. Dans *ACM Trans. Graph.*, volume 23, pages 457–462. ACM, 2004.

[25] Nelson L. Max. Shadows for bump-mapped surfaces. Dans *Advanced Computer Graphics*, pages 145–156. Springer, 1986.

[26] Thomas B. Moeslund, Claus B. Madsen, Michael Aagaard et Dennis Lerche. Modeling falling and accumulating snow. Dans *Second International Conference on Vision, Video and Graphics*, volume 1, 2005.

[27] J.J. Monaghan. An introduction to SPH. *Computer Physics Communications*, 48 (1):89 – 96, 1988. ISSN 0010-4655. URL http://www.sciencedirect.com/science/article/pii/0010465588900264.

[28] Tomoaki Moriya et Tokiichiro Takahashi. A real time computer model for wind-driven fallen snow. Dans *ACM SIGGRAPH ASIA 2010 Sketches*, page 26. ACM, 2010.

[29] Francis W. Murray. On the computation of saturation vapor pressure. Rapport technique, Rand Corp Santa Monica Calif, 1966.

[30] Tomoyuki Nishita, Hiroshi Iwasaki, Yoshinori Dobashi et Eihachiro Nakamae. A modeling and rendering method for snow by using metaballs. *Computer Graphics Forum*, 16(3):C357–C364. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00173.

[31] Per Ohlsson et Stefan Seipel. Real-time rendering of accumulated snow. Dans *The Annual SIGRAD Conference. Special Theme-Environmental Visualization*, pages 25–31. Linköping University Electronic Press, 2004.

[32] J. W. Pomeroy et D. M. Gray. Saltation of snow. *Water Resources Research*, 26(7): 1583–1594, 1990. ISSN 1944-7973. URL http://dx.doi.org/10.1029/WR026i007p01583.

[33] J.W. Pomeroy et D.M. Gray. Snowcover accumulation, relocation and management. *Bulletin of the International Society of Soil Science*, 88(2), 1995.

[34] J.W. Pomeroy, D.M. Gray et P.G. Landine. The prairie blowing snow model : characteristics, validation, operation. *Journal of Hydrology*, 144(1-4):165–192, 1993.

[35] Daniel Tobias Reynolds, Stephen D. Laycock et A.M. Day. Real-time accumulation of occlusion-based snow. *The Visual Computer*, 31(5):689–700, 2015.

[36] R.A. Schmidt. Sublimation of wind-transported snow–a model. *USDA Forest Service research paper RM-United States, Rocky Mountain Forest and Range Experiment Station*, 1972.

[37] R.A. Schmidt. Sublimation of snow intercepted by an artificial conifer. *Agricultural and Forest Meteorology*, 54(1):1–27, 1991.

[38] Natural Resources Conservation Service. National engineering handbook. Dans United States Department of Agriculture, éditeur, *The Oxford Handbook of Innovation*, chaptitre 11. United States Department of Agriculture, 2004.

[39] Peter-Pike Sloan, Jan Kautz et John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. Dans *Proceedings of SIGGRAPH*, pages 527–536. ACM, 2002. URL http://doi.acm.org/10.1145/566570.566612.

[40] Jos Stam. Stable fluids. Dans *Proceedings of SIGGRAPH*, pages 121–128. ACM, 1999. URL http://dx.doi.org/10.1145/311535.311548.

[41] Jos Stam. Real-time fluid dynamics for games. Dans *Proceedings of Game Developer Conference*, volume 18, pages 25–42, 2003.

[42] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran et Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4): 102 :1–10, juillet 2013. ISSN 0730-0301. URL http://doi.acm.org/10.1145/2461912.2461948.

[43] Robert W. Sumner, James F. O'Brien et Jessica K. Hodgins. Animating sand, mud, and snow. Dans *Computer Graphics Forum*, volume 18, pages 17–26. Wiley, 1999.

[44] A.D. Thorpe et B.J. Mason. The evaporation of ice spheres and ice crystals. *British Journal of Applied Physics*, 17(4):541, 1966.

Annexe I

Taux de perte par sublimation ψ.

Dans cette annexe nous décrivons comment calculer les coefficients ψ_s et ψ_t de perte de neige par sublimation, comme définis par Liston et Sturm [22]. Ce taux est exprimé dans l'équation de la sublimation :

$$Q_v(x^*) = \int_0^{z_t} \psi(x^*,z)\phi(x^*,z)dz\,, \tag{I.1}$$

avec ϕ_s et ϕ_t ($\mathrm{kg\,m^{-3}}$) la distribution verticale de la concentration des particules de neige respectivement dans la couche de saltation et dans celle de suspension. La solution à l'équation I.1 est basée sur les travaux de Schmidt [36, 37], de Pomery et al. [34] et de Pomeroy et Gray [33]. $\psi(x^*,z)$ est exprimé par :

$$\psi(x^*,z) = \frac{\frac{d\bar{m}(z)}{dt}}{\bar{m}(z)}\,, \tag{I.2}$$

avec t (s) le temps, $\bar{m}$ (kg) la masse moyenne des particules à la hauteur z qui est estimée par :

$$\bar{m}(z) = \frac{4}{3}\pi\rho_i\bar{r}_r(z)^3\left(1+\frac{3}{\alpha}+\frac{2}{\alpha^2}\right) \tag{I.3}$$

avec ρ_i ($\mathrm{kg\,m^{-3}}$) la densité de la glace, $\bar{r}_r$ le rayon moyen des particules de neige à la hauteur z donné par :

$$\bar{r}_r(z) = 4.6\times10^{-5}z^{-0.258}\,, \tag{I.4}$$

et le coefficient α défini par :

$$\alpha = 4.08 + 12.6z\,. \tag{I.5}$$

Le taux de changement de la masse moyenne des particules $\bar{m}$ est défini en fonction du gradient d'humidité entre la particule de neige et l'atmosphère, les radiations solaires,

la taille de la particule et l'advection. Il est défini par :

$$\frac{d\bar{m}}{dt} = \frac{2\pi\bar{r}\sigma - \frac{S_p}{\lambda_t T_a Nu}\left[\frac{h_s M}{RT_a} - 1\right]}{\frac{h_s}{\lambda_t T_a Nu}\left[\frac{h_s M}{RT_a} - 1\right] + \frac{1}{D\rho_v Sh}} , \tag{I.6}$$

où $M = 18.01$ ($\mathrm{kg\,kmol^{-1}}$) est la masse moléculaire de l'eau, $R = 8313$ ($\mathrm{J\,kmol^{-1}\,K^{-1}}$) est la constante de gaz universelle, T_a ($^\circ$K) est la température de l'air, $\lambda_t = 0.024$ ($\mathrm{J\,m^{-1}\,s^{-1}\,K^{-1}}$) est la conductivité thermique de l'atmosphère et $h_s = 2.838 \times 10^6$ ($\mathrm{J\,kg^{-1}}$) est la chaleur résiduelle de la sublimation [36, 44]. La diffusivité de la vapeur d'eau dans l'atmosphère D ($\mathrm{m^2\,s^{-1}}$) est donnée par [44] :

$$D = 2.06 \times 10^{-5} \left(\frac{T_a}{273}\right)^{1.75} , \tag{I.7}$$

et la densité de saturation de la vapeur d'eau ρ_v ($\mathrm{kg\,m^{-3}}$) à la température T_a ($^\circ$K) est [12] :

$$\rho_v = 0.622 \frac{e_s}{R_d T_a} , \tag{I.8}$$

avec $R_d = 287$ ($\mathrm{J\,^\circ C^{-1}\,kg^{-1}}$) la constante de gaz pour l'air sec. La pression de la vapeur saturée sur la glace e_s (Pa) est approximée par [29] :

$$e_s = 610.78 \exp\left(\frac{21.875(T_a - 273.16)}{T_a - 7.66}\right) , \tag{I.9}$$

avec T_a ($^\circ$K) la température de l'air.

Le rayon d'une particule de neige $\bar{r}_z$ (m), avec une masse moyenne $\bar{m}(z)$ est :

$$\bar{r}(z) = \left(\frac{3\bar{m}(z)}{4\pi\rho_i}\right)^{\frac{1}{3}} . \tag{I.10}$$

La distribution verticale de la vapeur d'eau sous-saturée atmosphérique par rapport à la glace $\sigma(z)$ est donnée par Liston et Sturm [22], et est exprimée en fonction de la hauteur z comme :

$$\sigma(z) = \sigma_r\,(\Gamma + 0.027 \ln(z)) \tag{I.11}$$

avec σ_r la sous-saturation à la hauteur d'observation de l'humidité relative z_{rh} (m)

$$\sigma_r = RH(z_{rh}) - 1 \ , \tag{I.12}$$

où RH est l'humidité relative exprimée par :

$$\Gamma = 1 - 0.027 \ln(z_{rh}) \ . \tag{I.13}$$

Les nombres de Nusselt Nu et Sherwoord Sh, caractérisant respectivement le transfert convectif et le transfert par diffusion, sont donnés par [22] :

$$Nu(z) = Sh(z) = 1.79 + 0.606 Re(z)^{0.5} \tag{I.14}$$

et

$$Re(z) = \frac{2\bar{r}(z) V_v(z)}{\upsilon} \tag{I.15}$$

où Re est le nombre de Reynolds de la particule [16], υ est la viscosité kinématique de l'air, et $V_v(z)$ est exprimé selon la hauteur et dans quelle couche les particules se trouvent : soit dans la couche de saltation, soit dans la couche de suspension :

$$V_v(z) = \begin{cases} V_t(z) = \bar{w}(z) + 3x_r(z)\cos\left(\frac{\pi}{4}\right) & \text{si } z \text{ dans la couche de suspension} \\ V_s(z) = 0.68u_* + 2.3u_{*t} & \text{si } z \text{ est dans la couche de saltation} \ , \end{cases} \tag{I.16}$$

où $\bar{w}(z)$ ($\mathrm{m\,s^{-1}}$) est la vitesse terminale de chute de la neige en suspension donnée par

$$\bar{w}(z) = 1.1 \times 10^7 \bar{r}(z)^{1.8} \tag{I.17}$$

La composante de fluctuation de la velocité des particules $x_r(z)$ est exprimée comme

$$x_r(z) = 0.005 u(z)^{1.36} \ . \tag{I.18}$$

La radiation solaire absorbée par une particule de neige S_p (W) est donnée par :

$$S_p = \pi \bar{r}_r(z)^2 (1-\alpha_p)(1+\alpha_s) S_i \,, \tag{I.19}$$

avec $\alpha_p = 0.5$ l'albedo d'une particule de neige, et $\alpha_s = 0.8$ l'albedo de la surface de neige [36]. La radiation solaire atteignant la surface de la Terre S_i ($\mathrm{W\,m^{-2}}$) est décrite par Liston [17] comme :

$$S_i = S^* \Upsilon \sin\omega \,, \tag{I.20}$$

avec $S^* = 1370\ \mathrm{W\,m^{-2}}$ l'irradiance solaire au sommet de l'atmosphère, et Υ la transmissivité nette du ciel, autrement dit le ratio de radiations solaires qui se rendent à la surface. L'angle d'élévation solaire ω est exprimé comme :

$$\sin\omega = \sin\delta \sin\eta + cos\delta \cos\eta \cos\tau \,, \tag{I.21}$$

avec η la latitude, τ l'angle horaire mesuré et δ l'angle de déclinaison du soleil approximé par :

$$\delta = \eta_{\mathbf{T}} \cos\left[2\pi\left(\frac{d-d_r}{d_y}\right)\right] \tag{I.22}$$

où $\eta_{\mathbf{T}} = 23.45\ ^\circ\mathrm{N}$ est la latitude du Tropique du Cancer, d est le jour grégorien, $d_r = 173$ est le nombre de jours du solstice d'été et d_y est le nombre moyen de jours dans une année.

Finalement, afin de prendre en compte l'absorption, la diffusion et la réflection du rayonnement à ondes courtes par les nuages, la radiation solaire S^* est multipliée par :

$$\Upsilon = (0.6 - 0.2\sin\omega)(1.0 - 0.5\sigma_c) \,, \tag{I.23}$$

avec σ_c le pourcentage de couverture nuageuse.