

MASTER 2

Informatique, Synthèse d'Images et Conception Graphique

2022-23

*Synthèse d'images réalistes de pluie
et de ses interactions.*

Allan Blanchard

Encadrant

Stéphane MERILLOU

Référent universitaire

Vincent JOLIVET

1 septembre 2023

Résumé

Ce rapport détaille un stage de recherche entrepris au sein du laboratoire XLIM à Limoges, où le principal objectif était la simulation réaliste de la pluie et de ses interactions avec son environnement. Notre étude a principalement porté sur la modélisation de la pluie sous l'influence du vent. Les résultats de notre approche basée sur un système de particules montrent une représentation visuelle convaincante de la pluie avec des variations d'intensité et des interactions avec un vent directionnel ainsi qu'avec les perturbations locales du vent dût au relief du terrain. Elle offre aussi des perspectives prometteuses pour des simulations météorologiques plus complexes, incorporant des phénomènes tels que la foudre, la brume, un cycle jour/nuit et l'évolution des nuages au fil du temps.

Table des matières

1	Introduction	3
1.1	Présentation du lieu de stage	3
1.2	Objectifs du stage	3
2	Rétrospective de phénomènes météorologiques	4
2.1	Les hydrométéores	4
2.1.1	Les corps en suspension	4
2.1.2	Les corps en chute	5
2.2	Les photométéores	6
2.2.1	La diffusion de la lumière dans l'atmosphère	6
2.2.2	Arcs, halos et mirages	7
2.3	Les électrométéores	8
2.3.1	La foudre	8
2.3.2	Les aurores polaires	9
2.4	Le vent	9
2.4.1	Les tempêtes	10
2.4.2	Les tornades	10
3	Travaux Connexes	11
3.1	Les précipitations	11
3.1.1	Approches utilisant un système de particules	11
3.1.2	Approches basées sur l'utilisation de texture	12
3.1.3	Approches hybrides	13
3.2	Le vent	15
3.2.1	Méthodes basées sur les équations de Navier-Stokes	15
3.2.2	Méthodes basées sur l'équation discrète de Boltzmann	16
3.2.3	Méthodes basées sur des primitives de vent	16
3.3	Les autres phénomènes	17
3.3.1	Diffusion de la lumière dans l'atmosphère	18
3.3.2	Génération et rendu de nuages, foudre et aurore polaire	19
3.3.3	Les conséquences des précipitations	20
4	Méthodes Développées	22
4.1	Première méthode : Génération de texture procédurale de corde	22
4.1.1	La texture procédurale	23
4.1.2	Le double cône	25
4.1.3	Les layers	27
4.2	Seconde méthode : Utilisation d'un système de particules	29
4.2.1	Création d'une boîte à goutte	29
4.2.2	Prise en compte de l'intensité de la pluie	35
4.2.3	Détection de collision entre les gouttes et les obstacles	38
4.2.4	Illumination des gouttes	40
4.2.5	Améliorations	41
4.2.6	Résultats	43
4.3	La brume	45
5	Conclusion et perspectives	47

1 Introduction

1.1 Présentation du lieu de stage

Le stage s'est déroulé au sein du laboratoire XLIM sur le site de Limoges, dans l'équipe ASALI/SIR. XLIM est un institut de recherche pluridisciplinaire localisé sur plusieurs sites : Angoulême, Poitiers, Brive et Limoges. Il regroupe une multitude de savoir-faire dans des domaines variés, notamment l'électronique et les hyperfréquences, l'optique et la photonique, les mathématiques, l'informatique et l'image, et bien d'autres champs de compétences touchant de près ou de loin au milieu spatial (les réseaux télécom, les environnements sécurisés, la bio-ingénierie, etc.).

L'institut regroupe environ 440 personnes, englobant des enseignants-chercheurs, chercheurs CNRS, ingénieurs, techniciens, personnels administratifs, post-doctorants et doctorants. Il est dirigé par Stéphane Bila et Stéphane Mérillou (directeur adjoint). Lors de mon stage, j'ai été encadré par Stéphane Mérillou, professeur des Universités en Informatique et Vincent Jolivet, maître de conférences en Informatique.

1.2 Objectifs du stage

Rappelons l'intitulé du stage : "Synthèse d'images réalistes de pluie et de ses interactions". Le but de celui-ci est comme son nom l'indique de réaliser une simulation de pluie la plus réaliste possible en prenant en compte ses interactions avec son environnement. Concernant les interactions, nous pouvons citer entre autres les écoulements, les égouttements, les éclaboussures, l'érosion ou encore l'absorption de l'eau par les surfaces, provoquant parfois un changement d'aspect de celles-ci. Modéliser tous ces phénomènes est une tâche complexe et nécessite de résoudre nombre de problèmes souvent traités indépendamment les uns des autres.

Puisque la simulation de tous ces phénomènes pourrait individuellement faire l'objet d'un stage, voire d'une thèse, nous avons dû nous concentrer sur un seul d'entre eux. Nous avons choisi d'axer ce stage sur la prise en compte du vent dans le mouvement des précipitations lors d'averse. Il va alors falloir trouver et comprendre les travaux précédemment menés sur le sujet et proposer ensuite une ou plusieurs méthodes se basant sur cet état de l'art permettant de modéliser le phénomène souhaité. Quelques contraintes viennent s'ajouter à cela. En effet, il serait préférable que la méthode puisse être compatible avec les applications de rendu temps réel et il faudrait qu'à terme elle supporte un vent variant au cours du temps et non uniforme dans l'espace.

A la suite de tout cela, une adaptation de la méthode sous le moteur Enreal Engine 5 était souhaitée pour diverses raisons, notamment afin de pouvoir réaliser de belles images et/ou vidéos de démonstration de la méthode sans avoir à re-développer un moteur de rendu entier. Malheureusement la complexité ainsi que l'opacité du moteur Unreal Engine 5 nous ont amenés à revoir cet objectif et nous avons convenu avec mes encadrants que le rendu de la méthode pourrait se faire via un moteur 3D en OpenGL qui à la base, a été confectionné durant le stage dans le but de faciliter le prototypage.

2 Rétrospective de phénomènes météorologiques

Avant de plonger au coeur de notre sujet, il est important de comprendre et répertorier les différents phénomènes météorologiques qui surviennent durant les intempéries afin de pouvoir cibler ceux que l'on souhaite simuler. Ce travail de compréhension nous permet ainsi de retranscrire les phénomènes sélectionnés le plus précisément et de la manière la plus réaliste possible. De nos jours, les phénomènes météorologiques sont de mieux en mieux compris. Pour faciliter leur étude, les météorologues se sont attelés à les classifier. Nous commençons par présenter ces différents phénomènes.

2.1 Les hydrométéores

Tout d'abord, il nous faut commencer par une brève explication sur la formation de la pluie. Pour diverses raisons, de la vapeur d'eau est relâchée dans l'atmosphère et charriée au grès des vents. En se condensant, l'eau se regroupe autour d'aérosols contenus dans l'air. On appelle ces regroupements des noyaux de condensation. Ces noyaux se regroupent alors pour former des nuages. Ces nuages vont ensuite évoluer en fonction des conditions de pression et de température auxquelles ils seront soumis durant leur existence. Lorsque les noyaux de condensation contenus dans le nuage seront trop lourds, ils se mettront à tomber sous forme de précipitation. Dans tout cela, les hydrométéores désignent les corps composés d'eau (sous forme liquide, solide ou gazeux), en suspension ou en chute libre dans l'atmosphère. On peut alors intuitivement déduire deux sous-groupes d'hydrométéores : les corps en suspension et les corps en chute.

2.1.1 Les corps en suspension

Dans ce premier groupe on trouve les nuages qu'on distingue à travers dix catégories principales. Tous d'abord les trois basiques :

Cirrus	Nuage formé d'une multitude de filament blancs. Annonciateur de beau temps.
Cumulus	Nuage le plus commun à l'aspect blanc éclatant et aux contours bien délimités. Il peut aisément faire plusieurs kilomètres d'épaisseur mais est bien souvent associé à du beau temps.
Stratus	Nuage grisâtre couvrant entièrement le ciel, il peut être à l'origine de faibles averses.

Par la suite cinq autres catégories de nuages contiennent en suffixe ces noms, cela signifie que leur apparence peuvent s'y appartenir : Stratocumulus, Altocumulus, Cirrocumulus, Altostratus et Cirrostratus. Leurs préfixes ont eut aussi une signification :

Cirro-	les nuages évoluent entre 6000m et 13000m d'altitude
Alto-	les nuages évoluent entre 2000m et 6000m d'altitude
Strato-	les nuages évoluent entre le sol et 2000m d'altitude

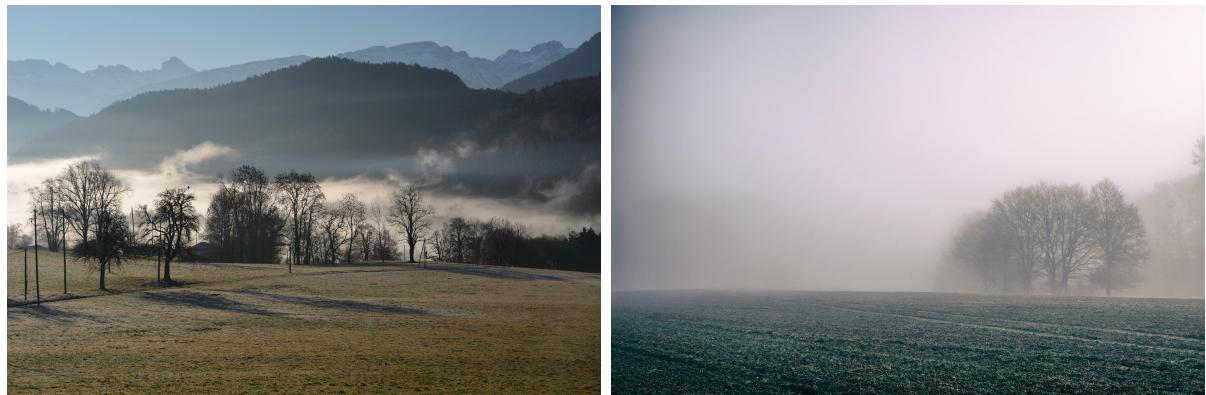
Les deux dernières catégories, Nimbostratus et Cumulonimbus, qui par leur appellation Nimbo-/nimbus indique que de fortes averses voire tempêtes sont en approche. Dans le tableau 1 ci-dessous, nous présentons des illustrations correspondant aux différents nuages.

				
Cirrus	Cumulus	Stratus	Nimbostratus	Cumulonimbus
				
Stratocumulus	Altocumulus	Cirrocumulus	Altostratus	Cirrostratus

TABLE 1 – Illustrations des différents types de nuages.

Image tirée de [wikipedia](#) et de [metoffice uk](#).

On retrouve aussi dans ce groupe la brume et le brouillard (illustrés sur les images 1). En réalité il s'agit d'un seul et même phénomène, seulement on parle de brume lorsque la visibilité est comprise entre un et cinq kilomètres, et de brouillard lorsqu'elle est inférieure à un kilomètre. Le diamètre des gouttes d'eau qui le composent varie d'un millième à un centième de millimètre. Il existe une large variété de processus qui permettent la condensation de la vapeur d'eau au voisinage de la surface de la terre pour aboutir à un tel phénomène, mais globalement son processus de formation est identique à celui des nuages (en particulier des stratus).



(a) Brume, image de [camptocamp](#).

(b) Brouillard, image de [meteocontact](#).

FIGURE 1 – Illustration de brume et de brouillard.

2.1.2 Les corps en chute

On distingue trois types de précipitations. Ces types dépendent uniquement des changements de température auxquels sera soumis l'eau durant sa chute. Comme l'illustre le schéma 2, on aura de la pluie si l'eau qui tombe du nuage est décongelée durant sa chute par des masses d'air chaud, on aura de la neige si l'eau n'est pas décongelée durant sa chute, et on aura de la grêle si l'eau est d'abord décongelée puis recongelée durant sa chute. Les précipitations peuvent être de taille variable mais là encore cela dépend de beaucoup de facteur. En règle générale, la goutte de pluie varie de 0.1 millimètres (bruine/crachin) à 6 millimètres. Et la taille des grêlons peut varier de quelques millimètres à quelques centimètres voire plusieurs dizaines de centimètres.

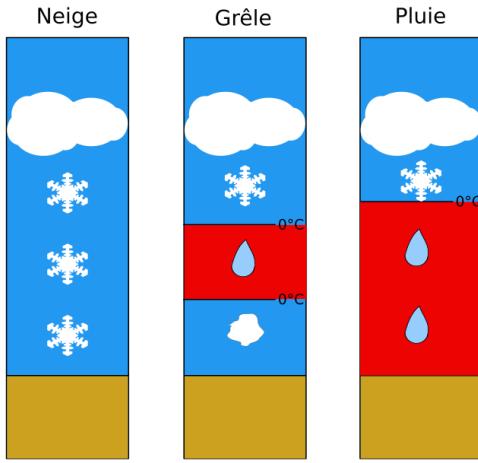


FIGURE 2 – Type de précipitation en fonction des conditions météo.
Image modifiée, tirée de [wikipedia](#).

La collision entre ces précipitations et la surface du sol ou d'un objet provoquent de nombreuses interactions. Premièrement dans le cas de goutte ou de grêlon, cette collision va créer un son. Ensuite la géométrie de cet objet peut se retrouver changée en raison de la violence de l'impact. Les grêlons et les gouttes peuvent se diviser et/ou rebondir (éclaboussure). Et l'eau peut aussi glisser sur les surfaces ou s'insinuer à l'intérieur de celles-ci et ainsi modifier ses propriétés physiques (absorption).

Au delà de ça, une fois arrivées sur le sol, les précipitations vont se déposer et former en fonction des conditions météorologiques :

- rosée : vapeur d'eau se liquéfiant proche du sol.
- verglas : dépôt de pluie qui gèle.
- flaqua : accumulation d'eau de pluie dans des zones creuses.
- givre ou la neige : monticule de flocons.

Au fur et à mesure que le temps avance, d'autres phénomènes liés à la mécanique des fluides peuvent émerger. On peut notamment voir apparaître des écoulements d'eau et des égouttements. Et sur une échelle de temps encore plus grande, des dégradations des surfaces en contact avec les précipitations voient le jour (érosion ou dépôt de matière).

2.2 Les photométéores

L'interaction entre ces hydrométéores et la lumière du soleil peuvent donner naissance à tout un tas de photométéores. On peut définir ces météores comme correspondant à tout phénomène optique atmosphérique engendré par une déviation de la lumière.

2.2.1 La diffusion de la lumière dans l'atmosphère

Commençons par le plus visible d'entre eux : la couleur du ciel. En effet, notre atmosphère se compose de cristaux de glaces (contenus au sein des nuages), de poussière, d'aérosols ainsi que de nombreux gaz. Avant de parvenir à nos yeux, la lumière émise par le soleil va interagir avec tous ces composants et va être absorbée, diffusée, dispersée, réfléchie et réfractée à plusieurs reprises. Au final, c'est cet amoncellement d'événements qui va venir définir la couleur du ciel.

2.2.2 Arcs, halos et mirages

Le plus populaire des photométéores est l'arc-en-ciel. C'est un phénomène courant qui est visible lorsque l'on regarde dans la direction opposée au soleil quand il brille par temps pluvieux. Il provient d'un jeu de réflexion-réfraction au sein des gouttes qui permettent la dispersion de la lumière et provoque cette célèbre arche multicolore dans le ciel ([3a](#)). Ce phénomène de diffraction de la lumière est aussi parfois observable en scrutant le sol à la suite d'une rosée ([3b](#)). En fonction des conditions dans lesquelles se trouve l'observateur et par le biais de mécanismes similaires à ceux qui régissent la création des arcs-en-ciel, il pourra avoir la chance d'admirer divers autres phénomènes comme des gloires ([3c](#)), des Heilgenscheins ([3d](#)), etc.



(a) Arc-en-ciel (b) Arc-en-ciel (c) Gloire (d) Heilgenschein

FIGURE 3 – Différents phénomènes en forme d'arc.

Image tirée de [wikipedia](#)

Les halos se manifestent comme des arcs faiblement colorés entourant le soleil ([4a](#)). Ils sont causés par la double réfraction de la lumière provenant du soleil et passant au travers d'une mince couche de cristaux de glace. Dans le même ordre d'idée, les rayons crépusculaires sont des phénomènes qui créent un genre de halo lumineux autour des nuages ou autres objets qui obstruent le soleil lorsqu'on le regarde. On peut y voir une multitude de rayon autour de la silhouette de l'objet. Il sont parfois appelés "god ray" ou "light shaft" dans la littérature anglo-saxonne ([4b](#)).



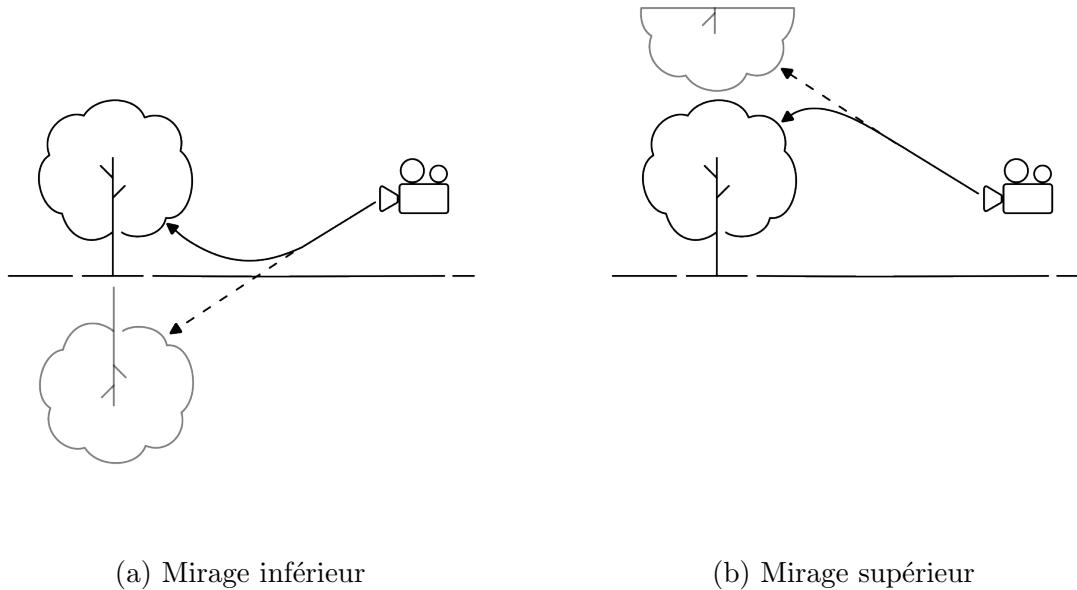
(a) Halo

(b) Rayons crépusculaire

FIGURE 4 – Différent phénomènes de en forme de halo.

Image tirée de [wikipedia](#)

Le mirage fait référence à une illusion d'optique qui donne l'impression d'observer un objet à un endroit différent de son emplacement réel. Il est dû à la déviation des faisceaux lumineux qui parviennent à nos yeux par une succession de couche d'air de température différentes. On appelle mirage inférieur un mirage dont l'image réfléchie est située au dessous de l'objet réel ([5a](#)), mirage supérieur un mirage dont l'image réfléchie est située au-dessus de l'objet réel ([5b](#)), et fata Morgana un mirage qui combine plusieurs mirages inférieurs et supérieurs.



(a) Mirage inférieur

(b) Mirage supérieur

FIGURE 5 – Schéma de fonctionnement des mirages.

2.3 Les électrométéores

Les météores qui composent ce groupe sont souvent les plus spectaculaires à observer dans la nature. Dans cette catégorie des électrométéores on regroupe tout phénomène lié à des décharges électriques dans l'atmosphère.

2.3.1 La foudre

Le plus évident d'entre eux est la foudre, ce phénomène se produit au sein du cumulonimbus quand une différence de potentiel immense se crée entre le haut et le bas du nuage. Pour évacuer ce surplus d'énergie, le nuage n'a alors d'autre choix que de se décharger dans le sol. L'événement se compose d'une ou plusieurs manifestations lumineuses (des éclairs ([6a](#)) et/ou des arcs électriques) et d'une manifestation sonore qu'on appelle tonnerre. Les arcs électriques peuvent se produire à l'intérieur d'un nuage, entre plusieurs nuages ou entre le sol et le nuage. Lorsque le champs magnétique produit par l'orage est suffisamment puissant, on peut observer sur les objets métalliques en hauteur qui se trouvent sur la trajectoire de l'éclair, le phénomène du feu de saint-elme ([6b](#)).



(a) Éclairs, image de [wallpaperflare](#).

(b) Feu de St.Elme, image de [couleur-science](#).

FIGURE 6 – Illustration de phénomènes liés à la foudre.

2.3.2 Les aurores polaires

Les aurores polaires (image 7) se produisent massivement aux dessus des lieux proches des pôles magnétiques terrestres. Elles proviennent de l'interaction entre des particules chargées contenues par les vents solaires et notre atmosphère. Il existe un multitude d'autres phénomènes similaires aux aurores polaires, se déroulant eux aussi en haute atmosphère et à la lisière entre les photométéores et électrométéores. Mais ces phénomènes (les farfadets, les jets bleus, etc.), sont bien plus inhabituels et il n'est pas utile pour nous de s'étendre sur le sujet.



FIGURE 7 – Aurore polaire, image tirée de [pxhere](#).

2.4 Le vent

Pour comprendre comment se crée le vent, on doit se pencher sur le comportement des masses d'air dans la troposphère¹. Il existe des masses d'air chaudes et des masses d'air froides qui circulent en permanence à la surface de la terre. À volume égal l'air chaud est moins dense et exerce moins de pression que l'air froid. Il en résulte que les masses d'air chaudes ont tendance à vouloir passer au-dessus des masses d'air froides sans se mélanger. C'est donc ces mouvements de masses d'air qui sont à l'origine du vent.

1. Partie de l'atmosphère allant de 0 à 10 kilomètres d'altitude.

Le vent généré de cette manière est relativement uniforme sur des dizaines voire des centaines de kilomètres. A l'inverse, il existe aussi un vent de surface qui se crée localement en fonction des reliefs et des échanges thermiques entre le terrain et le vent.

2.4.1 Les tempêtes

Lorsque un orage se met à dépasser les 200 kilomètres d'envergure, on parle de tempête. Ces tempête s'accompagnent de pluie ou de grêle mais aussi bien souvent de foudre et de tornade. On va plutôt parler de blizzard ou tempête de neige si ce sont des flocons qui tombent lors de ces tempêtes. Si les vents qui règnent à l'intérieur de ces événements sont supérieurs à 120 km/h on parlera de typhon, cyclone ou ouragan (ces trois termes désignent le même phénomène).

2.4.2 Les tornades

Les tornades sont des phénomènes qui peuvent être extrêmement spectaculaires et destructeurs. Ils ressemblent à des tourbillons de vents violents se prolongeant jusqu'au sol et se développant durant les périodes orageuses. Lorsque ce phénomène se manifeste au-dessus de la mer on préférera le terme de trombe marine.

Pour qu'une tornade voit le jour il faut d'abord qu'on fasse se rencontrer une masse d'air froide sèche en altitude et une masse d'air chaude et humide au sol. Si la différence de température et le taux d'humidité sont suffisants on va voir se former un orage qui accueillera en son sein un courant d'air ascendant et descendant (l'air chaud va monter et l'air froid va descendre). Il faut ensuite que des vents cisailant¹ viennent perturber notre orage. Cela va venir créer un vent tourbillonnant selon un axe horizontal dans notre nuage qui, additionné avec nos vents ascendants et descendants va faire tourner l'ensemble du nuage sur lui-même selon un axe vertical. En plus de cela, en dessous du centre du nuage, les deux masses d'air vont alors s'enrouler sans se mélanger pour former un entonnoir nuageux (tuba) jusqu'à toucher le sol pour former une tornade. Les vents à l'intérieur et autour de la tornade ainsi créée peuvent varier entre 100 et 500 km/h.



FIGURE 8 – Tornade, image tirée de [wikipedia](#).

1. Importante variation de vitesse des vents en fonction de l'altitude.

3 Travaux Connexes

Pour que notre simulation semble la plus réaliste possible il serait bénéfique qu'un grand nombre des phénomènes météorologiques présentés dans la partie 2 soit représenté. Cependant comme expliqué dans la partie 1, créer de toute pièce un modèle météorologique complexe capable d'inclure tout ces phénomènes se révèle être une tâche colossale qui dépasse de loin le cadre de ce stage. C'est pour cette raison que le modèle que nous développerons ne contiendra qu'un nombre restreint de manifestation météorologique, à savoir la représentation de précipitations sous forme de pluie et ses interactions avec le vent. La première tâche à accomplir avant de se lancer est bien évidemment de trouver et comprendre les travaux qui ont précédemment été réalisés sur le sujet. Bien que nous nous apprêtons à vous présenter un état de l'art répertoriant différentes techniques permettant la simulation de précipitation, nous allons tout de même aborder les travaux permettant la simulation d'autres phénomènes météorologiques présentés dans la partie 2.

3.1 Les précipitations

Puisqu'il faut bien commencer quelque part, nous sommes partis des deux thèses qui ont été menées sur ce même sujet à Limoges par Pierre Rousseau [Rou07] en 2007 et Yoann Weber [Web16] en 2016. Leurs écrits comportaient un état de l'art très complet sur les travaux menés à leur époque respective et sur lesquels nous nous sommes fortement basés. Il fut très instructif d'étudier les deux approches puisqu'ils ont choisi des chemins radicalement opposés pour simuler les précipitations de pluie. Grossièrement les travaux de Pierre Rousseau s'appuyaient sur l'utilisation de particules pour représenter les gouttes de pluie tandis que Yoann Weber à lui choisi de générer une texture procédurale qu'il appliquera ensuite comme un filtre sur l'image finale. Après analyse de l'état de l'art, trois tendances se dessinent en ce qui concerne la simulation des précipitations. A l'image de ces deux thèses nous avons des méthodes préférant l'utilisation d'un système de particules et d'autres optant pour l'utilisation de textures, mais on peut aussi retrouver des méthodes hybrides utilisant des particules pour simuler le trajet des gouttes et des textures pour définir la forme et l'illumination des cordes.

3.1.1 Approches utilisant un système de particules

Lorsque l'on souhaite se servir de particules dans une simulation il faut bien choisir où et comment nous allons les utiliser, puisque nous n'en disposons que d'un nombre limité. Dans le cadre de précipitation, ce qui va être important c'est que l'observateur ait en permanence des gouttes dans son champ de vision. Autrement dit, ici, il sera primordial de n'avoir des particules qu'autour de la caméra.

Globalement l'idée qui est privilégiée, est de ne faire évoluer nos particules qu'à l'intérieur d'une boîte située autour de la caméra. Les gouttes sortant de cette boîte seront immédiatement détruites et/ou recyclées. Rousseau et al. [RJG08] ont été les premiers à expliquer cette technique en profondeur et ont baptisé la boîte : "Rain box" ou "boîte à gouttes" en français. Ils proposent en plus dans leur article, une parallélisation de la méthode sur GPU, une gestion du vent et une détection des collisions pour les vents uniformes par le biais d'une carte de hauteur. Pour faire le rendu des gouttes ils choisissent de reprendre leurs précédents travaux [RJG06]. Dans cet article, ils ont montré que la réfraction est largement prédominante sur la réflexion dans l'apparence d'une goutte

d'eau et qu'à partir de ce postulat, on pouvait en déduire que le "champ de vision" d'une goutte s'étend sur 165 degré derrière elle. Ils proposent alors de rendre la scène une seconde fois avec une caméra virtuelle ayant un angle de vision de 165 et d'utiliser cette image pour calculer l'illumination des gouttes.

Peu après, Wang et al. [Cha+08] proposent eux aussi un modèle permettant de représenter la pluie à l'aide de particules. Ils choisissent de représenter les gouttes sous la forme de cordes et réalisent un travail visant à déterminer la forme de ces corde lorsque les gouttes sont soumises à la résistance de l'air durant leur chute. La deuxième partie de leurs travaux se concentre sur le rendu de brouillard. Ils remarquent qu'au-delà d'un certain seuil de distance à l'observateur, les gouttes ne deviennent plus perceptibles individuellement et peuvent alors être représentées sous la forme d'un milieu participant. Ils définissent alors un modèle de brume où les gouttes d'eau et les aérosols altèrent la diffusion de la lumière, et perturbent ainsi la visibilité.

Dans la réalité, puisque les précipitations proviennent des nuages, au sol les zones de pluie sont bien délimitées. Puig-Centelles et al. [PRC09] introduisent alors cette notion représentée sous la forme d'ellipses qui définissent les zones dans lesquelles la pluie peut tomber. Ainsi, les particules ne peuvent être générées que dans les zones où les ellipses et la boîte à goutte se chevauchent. Afin d'assurer une transition graduelle, lorsqu'une boîte à goutte se met à entrer dans une ellipse, l'intensité de la pluie et la taille des gouttes générées est progressivement augmentée au niveau de ses frontières.

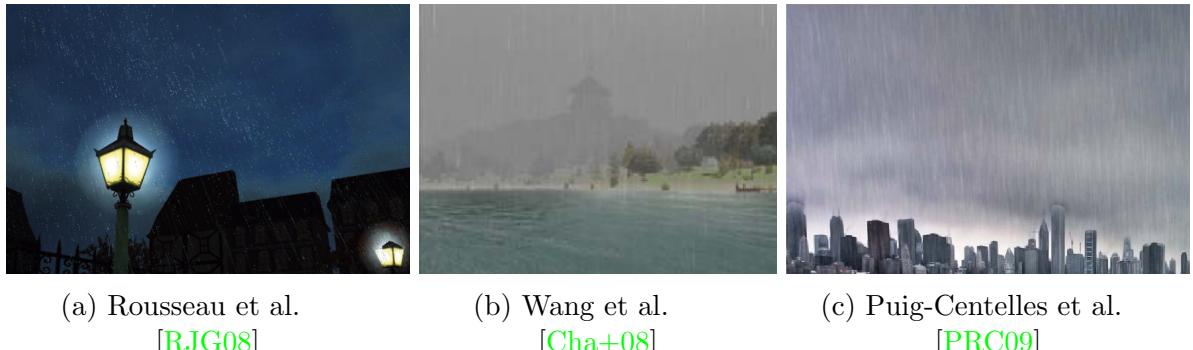


FIGURE 9 – Résultats des différentes méthodes utilisant des particules.

3.1.2 Approches basées sur l'utilisation de texture

L'utilisation d'un système de particule pouvant être considérée comme étant relativement coûteuse en termes de calcul, des alternatives bien moins coûteuses utilisant des textures ont été développées.

En 2004 Wang et Wade [WW04] proposent une technique à destination du simulateur "Flight Simulator" pour représenter les précipitations. Leur technique repose sur l'utilisation d'un double cône centré sur la caméra, sur lequel on viendra plaquer des textures créées par des artistes. Des transformations sont appliquées aux différentes textures qui sont empilées afin de donner un effet de parallaxe. Ces textures défilent continuellement vers le bas dans le but de simuler l'effet de la gravité sur les gouttes. Et pour finir le double cône est incliné de façon à coller au mouvement de la caméra.

Tatarchuck et Isidoro [TI06] proposent de faire coexister une multitude d'effets liés à la pluie au sein d'une même simulation. En s'inspirant des travaux de Wang et Wade [WW04], leur méthode représente les précipitations en utilisant une succession de texture tombant à des vitesses différentes selon la distance à l'observateur pour créer un effet de parallaxe tout en donnant une sensation de densité du phénomène. Ils ont ajouté à cela, une fine brume modélisée simplement en grisant les pixels de l'image finale en fonction de la distance à l'observateur. Et en ce qui concerne le reste des phénomènes, nous allons y revenir plus tard dans ce rapport.

Plus récemment, les travaux de Weber et al. [Web+15] proposent de générer une texture procédurale de pluie qui sera appliquée à l'image finale durant la phase de post-traitement. A la manière des travaux de Wang et al. ([Cha+08]), la méthode inclut une atténuation de la visibilité qui a elle aussi été modélisée sous la forme d'un milieu participant. Le bruit retenu pour la création de la texture procédurale permet d'utiliser les images de cordes provenant de la bibliothèque de texture de Garg et Nayar. Cette bibliothèque a été développée à la suite de leurs travaux de 2006 [GN06] et de 2007 [GN07] où ils ont réalisé une étude très poussée sur la forme des gouttes de pluie en chute.



FIGURE 10 – Résultats des différentes méthodes utilisant des textures.

3.1.3 Approches hybrides

L'inconvénient majeur de la plupart des techniques qui reposent sur l'utilisation de texture reste le fait qu'elles ne permettent pas de gérer les interactions entre gouttes et leur environnement. En revanche, leurs avantages par rapport au système de particule sont non négligeables. Premièrement leur temps de calcul est souvent bien moindre, et deuxièmement l'utilisation de texture permet d'avoir une forme et une illuminations des gouttes plus poussée que ce qui peut être proposé du côté des particules. Ce que l'on appelle méthode hybride correspond tout simplement à des techniques ayant recourt à la fois à des particules et à des textures. Elles naissent de la volonté de tirer parti du meilleur des deux camps.

En premier, nous allons présenté une méthode assez particulière. A la base créée pour représenter des chutes importantes de neige, la méthode de Langer et al. [Lan+04] est tout à fait viable pour représenter toute autre sorte de précipitation. L'idée est de simuler et rendre un faible nombre de particule pour ensuite utiliser des méthodes de synthèse spectrale sur la texture générée pour donner une impression de densité du phénomène.

S'en suivent les travaux de Tariq [Tar07] dirigé par Nvidia qui proposent une scène pluvieuse pour montrer les possibilités de leur matériel et de leur API graphique DirectX 10. Dans cette scène pas moins de 5 millions de particules sont simulés. Pour les illuminer, des textures provenant de la bibliothèque de Garg et Nayar sont plaquées sur des imposteurs placés au niveau du centre de ces particules. La méthode inclut aussi une brume comparable à celle présente dans les travaux de Tatarchuck et Isidoro [TI06].

Dans la même veine, Creus et Patow [CP13] proposent une méthode comparable pour simuler et rendre la pluie. Ils optent pour l'utilisation d'une boîte à goutte en forme de pavé centré sur la caméra et réalisent aussi une étude sur le mouvement des gouttes de pluie en chute libre. Tout comme Pierre Rousseau [RJG08], ils penchent pour l'utilisation d'une carte de hauteur pour gérer les collisions entre les précipitations et leur environnement. On peut aussi retrouver une approche très similaire dans des travaux récents proposés par Im et Sung [IS21]. A la différence qu'ils utilisent non plus une boîte à goutte en forme de pavé, mais une boîte à goutte collant parfaitement au frustum. Afin de réduire encore le nombre de goutte à simuler, ils négligent la zone du frustum se trouvant au-delà d'une certaine distance de l'observateur pour n'avoir à traiter que les gouttes étant le plus proche de lui.



(a) Langer et al.
[Lan+04]

(b) Tariq
[Tar07]

(c) Creus et Patow
[CP13]

FIGURE 11 – Comparaison des résultats entre les méthodes hybrides.

Du côté de la vision par ordinateur, nous avons aussi des approches intéressantes. Les problématiques qui sont évoquées dans ce domaine et qui concernent les précipitations sont globalement toujours les mêmes, il faut soit ajouter soit enlever les précipitations des photos ou vidéos. En réalité on peut faire un parallèle entre les méthodes développées ici et celles qui le sont dans le cadre de la synthèse d'image puisque si nous voulons employer ces méthodes, il nous suffit simplement de remplacer la vidéo ou l'image par le rendu de notre scène et de lui appliquer le même traitement post-processing. Parmi ces méthodes orientées vision, on peut citer le papier d'Halder et al. [HLC19] qui a pour but de transformer le beau temps en pluie dans des images ou des vidéos. Ils utilisent pour cela une simulation physique de particule sur lesquelles ils viennent plaquer des textures de corde issues de la fameuse bibliothèque de Garg et Nayar. Ils re-projectent ensuite les gouttes dans la scène à l'aide des données de profondeur extraites à l'aide d'algorithmes de vision. Puisque la luminosité d'une scène est bien plus atténue quand il pleut que quand il fait beau, ils assombrissent préalablement l'image et ajoutent une brume similaire à celle retrouvée dans les travaux de Tatarchuck et Isidoro [TI06]. Un second papier traitant exactement du même sujet est sorti à la suite en 2020 ([Tre+20]). Il apporte tout un tas

de précisions et de corrections sur la méthode, le tout en ajoutant quelques fonctionnalités à celle-ci.

3.2 Le vent

La méthode que l'on va développer n'aura pas pour vocation de contenir une simulation du vent mais simplement de prendre la représentation de ce vent comme une entrée. Il faut alors que nous nous penchions sur les différentes méthodes permettant la simulation d'un vent complexe afin de définir la nature de cette entrée.

3.2.1 Méthodes basées sur les équations de Navier-Stokes

Les équations de Navier-Stokes sont des équations qui régissent le mouvement des fluides dit newtonien. Ce terme englobe l'ensemble des gaz et une partie des liquides. En résumé, ces équations peuvent être utilisées pour simuler du vent, de l'eau, de la fumée, etc. En tant que tel, ces équations sont assez complexes et il n'existe pas encore de solution générale. Cependant, diverses approches se focalisant sur des cas bien spécifiques permettent de réaliser des simulations à l'aide de ces équations. Ce domaine étant encore bouillonnant à l'heure actuelle dans le monde de la recherche, il serait vraiment difficile de faire un état de l'art complet sur toutes ces méthodes. Heureusement pour nous, la majorité de celles-ci ne font qu'améliorer des techniques existantes. Et ces techniques sont elles-mêmes issues d'une poignée d'articles fondateurs.

Parmi ces articles, en tout premier nous trouvons la méthode SPH (pour "Smoothed particle hydrodynamics"), présentée en 1977 par le papier de Gingold et Monaghan [GM77]. Elle permet la simulation de fluide ou de gaz grâce à l'utilisation de particules. Chaque particule va représenter une partie du volume du fluide que l'on cherche à simuler. On va ensuite lui associer une densité et une pression puis on va la faire évoluer de manière à satisfaire les équations de Navier-Stokes. Il nous faut ensuite trouver un moyen de déterminer si un point de l'espace appartient ou non au fluide. Pour déduire ces informations à partir de celles que l'on a, la méthode propose d'utiliser une fonction de lissage qui va réaliser une moyenne pondérée avec toutes les particules du voisinage du point testé. Le poids choisi pour la moyenne sera dépendant de la distance entre la particule et le centre de recherche du voisinage. Cette méthode et ses dérivées sont toujours en vogue à l'heure actuelle malgré leur instabilité au niveau des limites du fluide.

Par la suite la méthode MPM ("material point method") est apparue dans l'article [SZS95]. Cette méthode permet aussi bien de simuler les fluides et gaz que les corps rigides ou déformables. C'est une méthode hybride qui se situe entre les méthodes dites Lagrangienne (à base de particules) et Eulérienne (espace discréte). Les objets sont ici représentés comme étant constitués d'un ensemble de points (particules) tous rangés dans une grille qui nous permet de calculer les déformations que subit l'objet. Par la suite, cette approche a été reprise par Disney ([Sto+13]) pour réaliser une simulation de neige pour le film d'animation "Frozen". Récemment la méthode MPM s'est faite complètement surpassée par la méthode du MLS-MPM ("moving least squares-material point method") provenant de l'article [Hu+18]. En plus d'être deux fois plus rapide à calculer que son homologue, cette nouvelle méthode offre de meilleurs résultats dans la plupart des cas et en particulier lors de fracture nette ou pour la simulation de fluide.

Et enfin la méthode PBD ("position based dynamics") présentée dans l'article [Müller+07] qui a été pensée pour la simulation de tissus et autres objets déformables. L'idée est de ne plus utiliser les schémas d'intégration classique comme Euler, Verlet ou Runge-Kutta mais plutôt d'appliquer un ensemble de contraintes au système en intervenant directement sur la position de ses particules. Du fait de sa grande stabilité au niveau des bordures, le modèle sera ensuite étendu aux fluides et au gaz par l'article [MM13] dans le but d'offrir une nouvelle alternative à la méthode SPH.

3.2.2 Méthodes basées sur l'équation discrète de Boltzmann

L'équation de Boltzmann est une alternative à la résolution des équations de Navier-Stokes. Afin de pouvoir l'utiliser dans la simulation de fluide, il nous faut d'abord discréteriser notre monde virtuel en un ensemble de noeud uniformément réparti dans l'espace. On va ensuite associer à chaque noeud une vitesse, un panel de liens vers des cellules voisines et une fonction de distribution qui indiquera la proportion de fluide qui se déplacera à chaque pas de temps vers les noeuds voisins. Il existe une multitude de manière de définir les liens entre chaque noeud, mais les plus couramment utilisés restent le D2Q9, D3Q15, D3Q19 et D3Q27 (figure 12). Cette nomenclature permet de distinguer et répertorier les différentes dispositions de lien qu'il existe. Elle permet d'indiquer à la fois la dimensionnalité du problème à l'aide du nombre qui vient après le "D" ainsi que l'arrangement et la quantité de lien grâce au nombre qui suit le "Q".

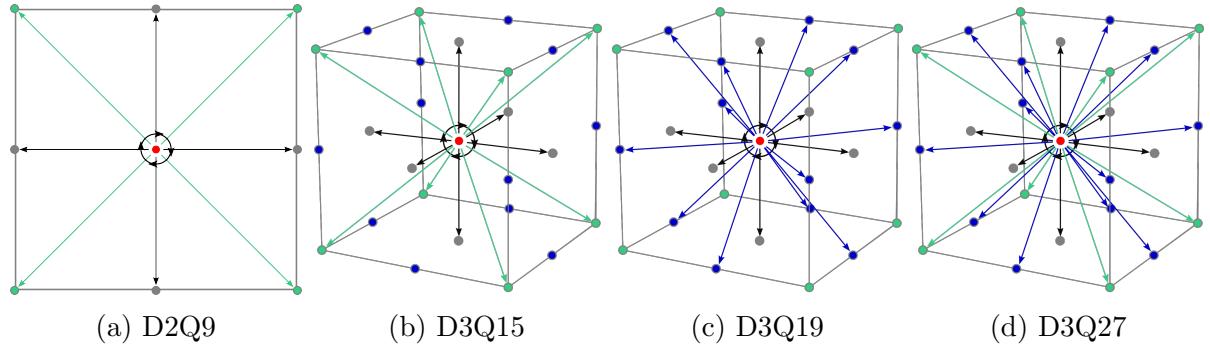


FIGURE 12 – Différents arrangements de liens entre cellules voisines.

De nombreux papiers utilisent cette technique afin de simuler les mouvements du vent. Nous pouvons citer entre autres [RMP16] qui utilise ce LBM (Lattice Boltzmann method) dans le but de reproduire les perturbations du vent au contact des bâtiments situés dans un environnement urbain. Il propose aussi une explication poussée sur le fonctionnement de cette fameuse méthode.

3.2.3 Méthodes basées sur des primitives de vent

Même si les approches présentées précédemment restent pour la plupart compatibles avec le temps réel, elles restent très coûteuses en temps de calcul. Il existe alors une dernière approche que nous n'avons pas encore évoquée qui est souvent celle préférée par les applications ayant de fortes contraintes liées au temps réel. L'histoire de cette technique commence avec l'article [WH91]. Ce papier nous propose d'utiliser quatre primitives de vents (image 13) qui se présentent sous la forme de fonctions mathématiques et qui peuvent s'assembler pour former un champs de vecteur plus ou moins complexe. Contrairement aux

méthodes précédentes, le réalisme du vent qui sera généré relèvera du talent de l'artiste en charge de mixer ces primitives. Nous avons les primitives suivantes :

- Un vent directionnel et uniforme dans tout l'espace (13a).
- Un vent redirigeant tout ce qu'il trouve vers un unique point (13b).
- Un vent partant d'un singularité qui repousse tout objet l'approchant (13c).
- Un vent tourbillonnant autour d'un point (13d).

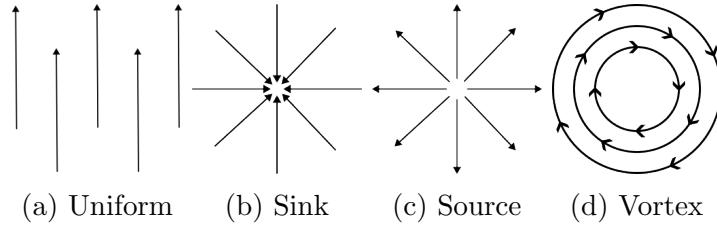


FIGURE 13 – Différents types de primitives de vents.

De nos jours, ces primitives de vents sont toujours d'actualité, au fur et à mesure des années, de nouvelles primitives ont même pu émerger. Dernièrement lors de la Game Dev Conférence 2019, Ruert Renard a pu expliquer la technique mise en oeuvre pour simuler le vent au sein d'un jeu de la série des God of War ([Ren19]). Elle reprend entièrement le principe du papier de 1991 mais ajoute des éléments majeurs. Il commence par ajouter trois nouvelles primitives (figure 14). La première (14a) représente le vent généré lors du mouvement d'un objet et les deux autres, en forme de cône tronqué illustre le phénomène d'aspiration (14b) et de soufflage (14c) de l'effet venturi. Ensuite une simulation physique du vent dans une zone restreinte autour de la caméra est réalisée afin de pouvoir tenir compte des obstacles de la scène.

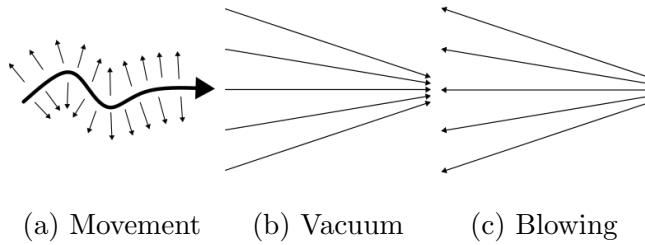


FIGURE 14 – Différents types de primitives de vents.

Au final, on remarque que quelle que soit la méthode choisie, le vent uniforme est en réalité un simple vecteur et les perturbations locales du vent peuvent être modélisées par un champ de vecteur. Ce champ de vecteur peut être représenté sous la forme d'une texture 2D ou 3D indiquant pour chacune des cases : une direction et une intensité. On peut éventuellement rajouter à ces cartes d'autres informations telles que la pression, la température ou l'humidité en fonction des besoins.

3.3 Les autres phénomènes

Même si au final notre méthode n'intégrera pas la simulation de ces phénomènes il peut être intéressant d'étudier les travaux les traitant, puisque cela pourrait s'avérer utile pour toute personne souhaitant reprendre et améliorer notre modèle. Sans trop nous étendre nous allons ici évoquer différents papier proposant des méthodes qui pourraient y être aisément incorporées.

3.3.1 Diffusion de la lumière dans l'atmosphère

C'est un phénomène auquel nous sommes confrontés au quotidien lorsque que nous regardons par la fenêtre ou lorsque nous sortons à l'extérieur. Pour des mondes virtuels qui se veulent un minimum réaliste, il est impératif de représenter cette diffusion puisque c'est elle qui est à l'origine de la couleur du ciel et permet une meilleure immersion de l'utilisateur.

Les premières méthodes qui visaient à reproduire la couleur du ciel consistaient soit à utiliser une couleur uniforme en fond du rendu, soit à utiliser une "environment map" issue de différentes photos du ciel. Ces méthodes très simples et qui ne demandent aucun calcul ne sont malheureusement pas suffisantes pour reproduire toutes les subtilités de ce phénomène. Par la suite de bien nombreux travaux ont tenté de simuler avec précision les interactions de la lumière avec les corps qui composent l'atmosphère. Ici, nous avons fait le choix de ne pas présenter l'ensemble de ces travaux mais uniquement ceux ayant permis des avancées majeures sur le sujet.

En 1997, Jackel et Walter [JW97] ont publié un papier dans lequel ils réduisent l'atmosphère en une suite de sphères concentriques contenant différents gaz et aérosols. Ils simulent ensuite le trajet de la lumière à l'intérieur de ce modèle d'atmosphère simplifiée. Cette méthode permet en plus de rendre la couleur du ciel (figure 15), la représentation d'autres phénomènes comme les arcs en ciel ou le brouillard. Malheureusement son principal problème réside dans son temps de calcul, où plusieurs minutes sont nécessaires pour rendre chaque image. Peu après, en 1999 Preetham et al [PSS99] publient à leur tour un papier traitant de ce sujet. Contrairement aux travaux de Jackel et al, ils ont fait du temps de calcul une priorité. En plus de cela, la méthode intègre un principe de dégradation de la visibilité qui permet de donner une impression de profondeur. Leurs résultats sont visibles sur les images de la figure 16.

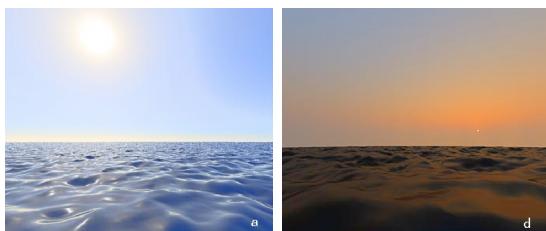


FIGURE 15 – Jackel et Walter [JW97]

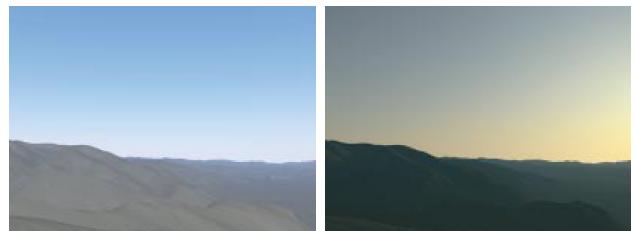


FIGURE 16 – Preetham et al. [PSS99]

Plus récemment en 2008 Bruneton et Neyret [BN08] présentent une méthode prenant en compte les multiples rebonds de la lumière à l'intérieur de l'atmosphère. Ce papier propose de pré-calculer un certain nombre de terme et cela pour tous les points de vue possible. Cette phase de pré-calculation est coûteuse (environ 6 secondes sur leur ordinateur) mais permet ensuite d'obtenir des effets d'ombres projetées venant des montagnes (image 17). Le rendu de l'image finale sera fait quant à lui dans un temps plus raisonnable de l'ordre de quelques millisecondes. Hillaire [Hil20] reprendra ensuite ses travaux pour réduire drastiquement le temps de calcul et arriver sur le rendu d'une image finale qui prendra moins d'une milliseconde sur son ordinateur et cela pour n'importe quel point de vue. On peut observer ses résultats sur l'image 18.



FIGURE 17 – Bruneton et Neyret [BN08]



FIGURE 18 – Hillaire [Hi20]

3.3.2 Génération et rendu de nuages, foudre et aurore polaire

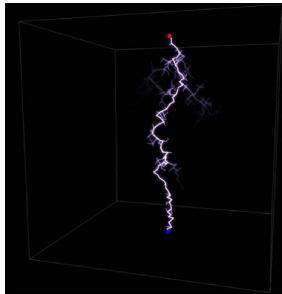
Au même titre que la couleur du ciel, les nuages sont eux aussi omniprésents dans notre vie de tous les jours. Leur représentation dans les mondes virtuels est étudiée depuis de nombreuses années. En 2019, une thèse a été menée sur ce sujet précis [Web19]. Elle visait à créer une méthode procédurale permettant de générer et rendre différents types de nuages évoluant au cours du temps et prenant en compte le relief du terrain. Aussi un état de l'art très détaillé sur les méthodes existantes pour simuler différents phénomènes météorologiques y est proposé. Nous invitons le lecteur à le consulter afin d'en apprendre davantage sur le sujet. Sur la figure 19 on peut observer respectivement des images de cumulus, stratocumulus, cirrocumulus et nimbostratus, modélisés grâce à la méthode proposée dans la thèse.



FIGURE 19 – Génération et rendu de nuages par Webanck [Web19].

Pour les éclairs, Tatarchuck [TI06] propose de simplement ré-hausser la luminosité de la scène et changer l'opacité des gouttes pour simuler le flash lumineux qui survient lorsque la foudre s'abat. Malheureusement dans cette méthode, ces flashes ne sont pas liés à de réels éclairs et il sera donc impossible pour un observateur de voir la foudre dans le ciel. Pour cela, en 1984 Niemeyer et al. [NPW84] construisent un modèle physique permettant entre autre de décrire le phénomène se cachant derrière la formation de la foudre. Par la suite de nombreux travaux se basent sur ce modèle pour calculer l'énergie potentielle électrique de chaque case d'une grille régulière. Ils peuvent alors reconstruire le chemin emprunté par l'éclair et générer une forme très réaliste de foudre. Cela demande néanmoins une quantité conséquente de calculs et fait donc de ces méthodes des techniques difficilement utilisables par les applications temps réel. C'est à partir de là que Yun et al. [Yun+17] ont sorti leurs travaux consistant à adapter ces méthodes afin de diminuer leur temps de calcul (résultat image 20a, pour une grille de 64 par 64 par 64). Quelques années plus tard Reis et Fernandes [RF22] sont arrivés avec une méthode assez novatrice qui reprend des algorithmes utilisés pour la génération d'arbre. Afin d'améliorer leur modèle ils reprendront aussi différents éléments des travaux précédemment menés sur le sujet. L'image 20b illustre leurs résultats.

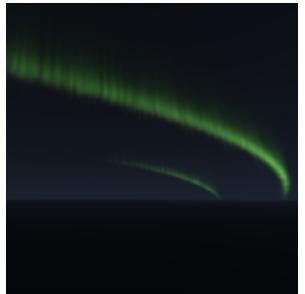
Les aurores polaires étant des phénomènes météorologiques bien moins fréquents que ceux évoqués précédemment, peu de chercheurs se sont intéressés à sa simulation. Dans l'état de l'art de la thèse d'Antoine Webanck [Web19] on peut retrouver plusieurs papiers traitant du sujet, avec notamment [Ish+11] et [LG11]. Là où le premier papier propose de simuler le phénomène en modélisant le flux de particule au sein de la magnétosphère en se basant sur des données réelles (résultat image 21a), le deuxième papier choisit une approche radicalement différente et propose de réaliser une simulation de fluide sur une grille 2D. La grille sera ensuite convertie en une texture qui sera alors étirée verticalement afin d'obtenir le rendu visible sur l'image 21b. Depuis, très peu d'articles sur le sujet sont sortis. Ishikawa a tout de même continué d'écrire des papiers en lien avec les aurores polaires comme par exemple [INM19] en 2019 qui offre une méthode permettant d'animer le phénomène.



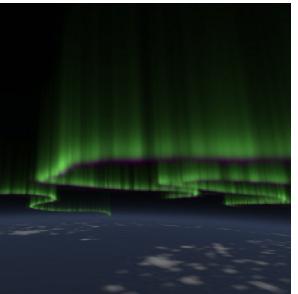
(a) Yun et al



(b) Reis et Fernandes



(a) Ishikawa et al



(b) Lawlor et Genetti

FIGURE 20 – Génération d'éclairs.

FIGURE 21 – Génération d'aurore polaire.

3.3.3 Les conséquences des précipitations

Durant les intempéries de nombreux phénomènes apparaissent. Parmi les travaux qui traitent de ceux-ci, le papier de Tatarchuck et Isidoro ([TI06]) se démarque particulièrement. Son objectif était de faire coexister un maximum de ces phénomènes au sein d'une même simulation. Parmi ces phénomènes on retrouve la génération d'éclaboussure lorsque les gouttes de pluie rencontrent un objet ou le sol. Les animations de ces éclaboussures sont ici créées en plaquant une séquence d'images sur des imposteurs. On retrouve aussi des écoulements d'eau et de l'égouttement le long des gouttières qui sont simplement créés en utilisant un système de particules. Sur le sol on peut également observer une accumulation d'eau donnant l'impression qu'une fine couche de liquide repose sur celui-ci. Cet effet est généré grâce à l'utilisation de différentes textures (displacement map, specular map et normal map). Si vous souhaitez examiner les résultats obtenus, une [vidéo](#) de démonstration est associée au papier.

Les égouttements ont aussi fait l'objet d'une étude par Weber et al. [Web+16]. Même s'ils ont fait le choix de représenter le phénomène uniquement au niveau des feuilles d'arbres, ils proposent une alternative à l'utilisation de système de particule qui peut tout à fait être étendue à un cas plus général. Leur méthode consiste à créer des imposteurs en dessous des sources d'égouttement sur lesquelles on viendrait calculer une texture représentant les gouttes en chute. Une [vidéo](#) accompagnant leur papier est fournie.

Au fur et à mesure des intempéries, l'eau qui ruisselle sur le sol va venir éroder et sculpter le paysage. Il existe un large panel de techniques visant à simuler cette érosion en utilisant des cartes de hauteur comme par exemple Mei et al. [MDH07] (résultats 22a). Ces méthodes sont malheureusement assez limitées puisque qu'elles ne permettent pas de représenter les zones creuses comme les arches ou les caves. Pour abroger les limitations liées à l'utilisation de ces cartes, d'autres papiers comme [Pey+09] font le choix de se détourner de la simulation physique pour offrir une méthode qui laisse plus de liberté aux artistes. Pour stocker les données sur la topographie du terrain, les auteurs optent pour une représentation implicite de la géométrie. Il ajoute à cela une grille 2D contenant des piles de matériaux pouvant être au choix de l'eau, de l'air, du sable, des roches ou la croûte terrestre. En plus, pour faciliter la construction du terrain ils proposent différents outils permettant d'éroder, de réaliser un texturing automatique ou de générer des piles de rocher. Sur l'image 22b vous pouvez observer une scène entièrement modélisée par le biais de cette méthode.



FIGURE 22 – Érosion de terrain.

A la suite du passage de l'eau, des sédiments contenus dans le liquide peuvent se déposer et être à l'origine d'un changement d'aspect des surfaces sur lesquelles ils se trouvent. Récemment Jonchier et al. [Jon+22] ont développé une technique permettant entre autre de reproduire le phénomène. Ils se basent sur la méthode SPH permettant de simuler le mouvement des particules. Ils prennent soin de faire la distinction entre les particules associées au liquide et celles associées aux sédiments, avant de réaliser une simulation physique de l'écoulement de l'eau (résultats image 23).



FIGURE 23 – Transport et dépôt de sédiment par l'eau par Jonchier et al. [Jon+22]

4 Méthodes Développées

Tout d'abord demandons-nous ce que nous pouvons attendre de la méthode que nous voudrions développer. Nous attendons de cette méthode qu'elle nous permette d'observer les précipitations visible par temps de pluie sous forme de corde. Cette méthode devra également prendre en compte un vent directionnel et une carte 3D de vent (champs de vecteur). Il serait aussi préférable que les effets tels que les bourrasques de vent (appelées train de vent) soient visibles au travers de la simulation. Dans l'idéal nous aimerais aussi que la méthode soit compatible avec des applications temps réel, que son utilisation et le changement de ses paramètres soient intuitifs pour l'utilisateur, et bien sûr nous voudrions qu'elle fournisse un résultat visuellement convaincant.

Pour synthétiser l'état de l'art, nous pouvons observer que globalement, deux tendances se dégagent en ce qui concerne le rendu des cordes : l'utilisation de textures et l'utilisation de système de particules. Ces deux types de méthodes comportent leur lot d'avantages et d'inconvénients, tout dépend du cas d'utilisation. C'est pourquoi afin de trouver une méthode qui puisse satisfaire nos attentes nous avons exploré ces deux voies et nous avons donc pu développer deux méthodes que nous allons vous détailler.

En ce qui concerne la brume observable au loin, nous trouvons que les travaux menés par Yoann Weber [Web+15] donnent un résultat tout à fait convaincant. De plus sa méthode est régie par seulement quelques paramètres intuitifs et pertinents. Nous reprenons donc cette partie de ses travaux afin de les incorporer dans notre simulation, il faudra cependant que nous mesurions l'impact du vent sur cette dernière afin de la modifier en conséquence avant de l'incorporer dans notre simulation. Mais nous reviendrons bien évidemment sur cela plus loin dans le rapport.

Bien que les méthodes que nous nous apprêtons à présenter ne permettent pas de simuler l'ensemble des phénomènes observables durant les périodes pluvieuses, il est tout à fait possible de les compléter avec d'autres techniques simulant ces phénomènes. Nous avons déjà exposé des éventuelles solutions dans l'état de l'art.

4.1 Première méthode : Génération de texture procédurale de corde

L'idée de cette méthode nous est venu de la volonté de poursuivre les travaux menés par Yoann Weber [Web16]. Sa méthode de rendu de corde proposait de générer une texture procédurale en espace écran ne rendant que les cordes visibles à l'écran. Cette méthode tiendrait compte de la profondeur de vue (distance à laquelle nous rencontrons un obstacle opaque) et des zones où la pluie ne peut pas accéder du fait qu'elle ait rencontré un obstacle plus en hauteur. De plus le nombre de corde à rendre serait calculé en fonction de paramètre facilement compréhensible par l'utilisateur. Cette méthode diverge des autres méthodes qui utilisaient des textures, du fait que la texture utilisée soit générée procéduralement. Cela permet à la fois de tenir compte des spécificités de la scène mais aussi de pouvoir changer dynamiquement l'intensité de la pluie.

Malheureusement cette technique en l'état comporte de nombreux défauts. Entre autre, elle ne permet pas d'avoir une cohérence spatiale des gouttes et cela se ressent notamment lorsque l'on bouge la caméra de droite à gauche ou que l'on se déplace dans

l'espace, on peut remarquer que les gouttes suivent la caméra. De plus la méthode n'a pas été développée pour tenir compte du vent, de ce fait les gouttes tombent parfaitement droit de haut en bas de l'écran (puisque la texture 2D défile simplement de haut en bas de l'écran). En plus de cela, on peut observer de grosses incohérences visuelles lorsque l'on se met à regarder vers le haut de notre monde, puisque les gouttes continueront de défiler du haut vers le bas de notre écran. Et en plus de ces défauts, il y a un flou autour de la manière dont la méthode fait pour illuminer ses cordes. Nous n'avons des informations claires que sur la manière dont sont déterminées les formes de corde.

Bien que cette méthode en l'état comporte pas mal de défauts, nous avons souhaité explorer cette voie puisque les problèmes présentés plus haut peuvent être réglés à l'aide d'autres méthodes évoquées dans l'état de l'art et que l'utilisation de texture procédurale pour simuler des cordes semble n'avoir été que très peu explorée. Pour bien comprendre comment marche notre méthode nous allons la construire pas à pas ensemble.

4.1.1 La texture procédurale

Pour commencer, partons de la méthode de Yoann Weber [Web+16]. Pour rappel Yoann utilise un bruit appelé LRPN (local random phase noise) présenté pour la première fois en 2014 ([Gil+14]). Ce bruit va découper notre texture en une grille régulière, sur laquelle seront appliqués des kernels. L'équation associée à ce bruit est la suivante :

$$LRPN(x) = \sum_i^I w\left(\frac{\|x - x_i\|}{\Delta}\right) \sum_j^J A_{ij} \cos(2\pi(f_{ij} \cdot x) + \phi_{ij})$$

$$w(y) = \frac{b_3(3\pi\sqrt{1 - (\frac{y}{1.5})^2})}{b_3(3\pi)}$$

$$b_3(z) = \sum_{m=0}^3 \frac{(0.5z)^{2m+3}}{m!(m+2)!}$$

Pour x les coordonnées de texture que l'on utilise pour échantillonner, x_i d'autres coordonnées de textures proches de x , Δ la taille des cellules de notre grille et A_{ij} , f_{ij} et ϕ_{ij} respectivement les amplitudes, fréquences en phases de nos kernels. Ces kernels représenteront nos cordes, plusieurs kernels peuvent être contenus dans chaque case de la grille et des transformations peuvent leur être appliquées comme des rotations, des homothéties, des translations, etc.

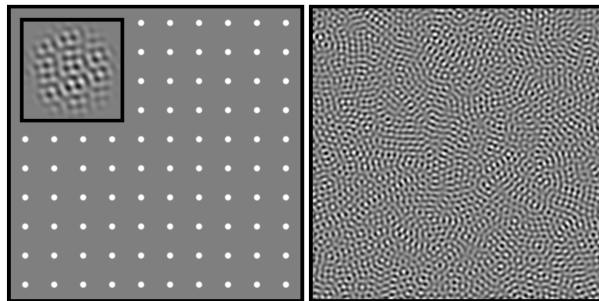


FIGURE 24 – Schéma de fonctionnement du bruit "LRPN".

Image tirée de la thèse d'Arthur Cavalier [Cav19].

Le schéma 24 illustre très bien le fonctionnement de ce bruit. Sur l'image de gauche on a en fond des points représentant les centres de nos grilles et en petit un kernel. Nous avons à droite le résultat de l'application des kernels sur cette grille. En réalité le type de bruit employé importe peu, cependant son utilisation permet de recourir à des kernels nécessitant une ou plusieurs phases, fréquences et amplitudes pouvant être extraites à l'aide de la théorie de fourrier sur des textures pré-calculées et permettant d'avoir des formes de cordes très réalistes. Si vous souhaitez en savoir plus sur les propriétés des différents bruits existants, nous vous conseillons d'aller lire la thèse d'Arthur Cavalier ([Cav19]) ou bien le papier de Lagae et Al. [Lag+10] proposant un état de l'art très complet sur le sujet.

Pour connaître le nombre de cordes/kernels à rendre par case de la grille, nous pouvons le déterminer en fonction de l'intensité de la pluie, de la taille des gouttes, de la profondeur de vue et d'une carte que l'on appellera "carte d'ombrage de pluie". Les informations de profondeur de vue correspondent simplement aux informations contenues dans le depth buffer après avoir rendu la scène. La carte d'ombrage de la pluie contiendra les informations sur les zones où la pluie ne peut pas accéder du fait qu'elle ait rencontré un obstacle plus en hauteur. Pour l'obtenir, l'idée est simple. On reprend le principe d'une shadow Map utilisée par les directionals lights. Il nous suffit donc de rendre notre scène du point de vue de la pluie avec une projection orthographique. Le depth buffer contiendra alors les données permettant de savoir pour tout point de notre espace de vue si la pluie doit y être rendue ou non. Et pour la taille des gouttes et l'intensité de la pluie, ce sont simplement des paramètres de la simulation.

En rassemblant les informations contenues dans la carte de profondeur et celle d'ombrage nous sommes en mesure d'estimer pour chaque pixel de notre écran le volume d'air dans lequel peuvent être contenues des cordes. La carte en résultant sera nommée par la suite la "carte de densité de pluie". Pour rassembler ses informations, plusieurs approches sont envisageables.

Nous avons imaginé une première approche "brute force" basée sur le ray marching. Pour chaque pixel de notre écran, nous lançons un rayon sur lequel nous allons avancer pas à pas jusqu'à atteindre la "profondeur de vue". A chaque pas nous allons évaluer notre carte d'ombrage de pluie. Si nous nous situons pas dans l'ombre, nous calculons un volume d'air à l'aide de la matrice de projection, de la taille du pas et de la distance au pixel. Si nous nous situons dans une ombre ce volume sera égal à 0. Pour finir tous ces volumes seront alors additionnés.

Une deuxième approche peut être envisagée, celle-ci reprend le principe des shadows volumes. Il nous suffit de rendre un à un les objets de la scène du point de vue de la pluie. Mais plutôt que de générer une shaodw map, il nous faut générer la géométrie englobant l'ombre projetée. Ensuite en lançant des rayons à travers la géométrie nouvellement formée, il nous est aisé de retrouver pour chaque pixel ses informations de densité.

Maintenant, déterminons le nombre de corde par pixel qu'il nous faut générer. Pour cela, partons de la distribution de Marshall et Palamars ([MP48]) :

$$N(D, I)\delta D = 8000e^{-4.1I^{-0.21}D}\delta D$$

Cette équation nous donne, pour des diamètres de goutte variant de D à $D + \delta D$, le nombre de goutte contenues par m^3 d'espace en fonction de l'intensité I de la pluie.

Il ne nous reste alors plus qu'à calculer : $Area * N(D, I)\delta D$ où $Area$ est le volume d'air contenu dans la carte de densité de pluie. Nous obtenons alors le nombre de gouttes contenues dans chaque pixel. Lors de l'évaluation du bruit, nous pouvons appliquer différentes transformations aux kernels pour donner une sensation de densité et de profondeur à la pluie.

Pour mieux se rendre compte étape après étape de l'évolution de notre modèle nous exposerons des schémas de plus en plus complets. Le premier est le schéma 25. Sur celui-ci nous avons en gris le frustum de vue, en nuances de bleu la texture de pluie et en fond notre rendu de la scène sans pluie. A ce stade nous avons quelque chose de très similaire aux résultats obtenus par Yoann Weber puisque nous ne faisons rien de plus que générer une texture de pluie à l'aide des informations sur la géométrie de la scène. Ensuite nous utilisons cette texture comme un "filtre" que nous plaçons par dessus le rendu notre scène.

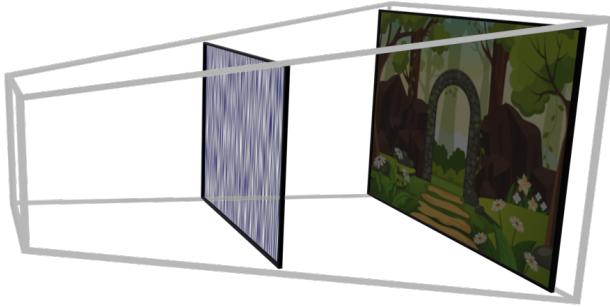


FIGURE 25 – Texture procédural de pluie plaquée sur un quad screen.

4.1.2 Le double cône

Il nous reste encore à régler les problèmes liés à la caméra évoqués précédemment. Dans un papier de 2004 [WW04] à destination du simulateur "flight simulator" une méthode de plaquage de texture de précipitation est proposée. Le principe est de créer un double cône centré sur la caméra et de venir y plaquer notre texture (comme illustré par la figure 26). La texture plaquée serait continuellement défilée de haut en bas du cône, à la vitesse de chute des précipitations. Cela permet de donner la sensation que les gouttes ne suivent pas la caméra lorsque celle-ci se tourne.

Il y a là deux problèmes majeurs qu'il va nous falloir résoudre si on veut pouvoir utiliser cette technique. D'une part, puisque la texture défile de haut en bas du cône, lorsque la caméra est immobile nous aurons la sensation que la pluie tombe vers le sol sans tenir compte de la direction et la force du vent. D'autre part, lorsque la caméra se met en mouvement la texture continue de défiler sur le cône sans se soucier de ce mouvement. Heureusement le papier nous propose une méthode pour calculer l'inclinaison du cône en fonction du mouvement effectué par l'observateur. On a donc :

$$\vec{Dir}_{prec} = (C_f \times \vec{Vel}_{camera} + \vec{G})\Delta t$$

Où \vec{Dir}_{prec} correspond au vecteur régissant l'orientation du cône, C_f correspond à un coefficient de damping, \vec{Vel}_{camera} à la vitesse de la caméra et \vec{G} à la force de gravité. Cette équation peut être adaptée pour tenir compte de la direction globale du vent :

$$\vec{Dir}_{prec} = (C_f \times \vec{Vel}_{camera} + \vec{G} + \vec{Dir}_{wind})\Delta t$$

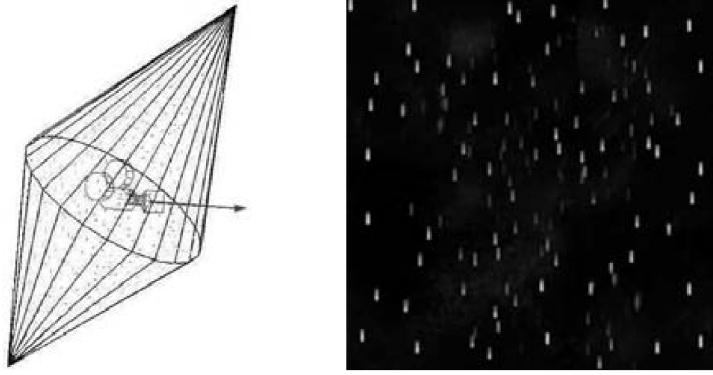


FIGURE 26 – Double cône centré sur la caméra et sa texture plaquée.
Image tirée du papier de 2004 [WW04].

Le deuxième problème survient lorsque l'on tente de plaquer notre texture sur le cône. Bien que l'on puisse définir les UVs de ce dernier comme cela nous arrange, nous risquons tout de même de voir des déformations de nos textures apparaître. Le papier fait part de ces problèmes et nous explique qu'ils peuvent tout à fait être réglés à l'aide des opérations classiques de traitement des textures.

De plus certaines adaptations de notre méthode seront nécessaires pour pouvoir utiliser cette technique du double cône. Premièrement, les UVs que nous utiliserons pour échantillonner notre bruit ne seront plus corrélés aux UVs de notre écran mais à ceux des fragments du cône projetés sur notre écran lors du processus de rastérisation. L'utilisation d'un double cône implique aussi une adaptation des transformations appliquées aux différents kernels de notre bruit afin de mieux correspondre à la nouvelle géométrie sur laquelle nous le plaquons (nous passons d'un rendu dans un "quad screen" plat à un double cône facétisé).

Sur le schéma 27 qui représente l'état actuel de notre simulation, nous pouvons observer que nous n'avons rien ajouté/enlevé à ce que nous avions précédemment. Nous avons simplement transformé la manière de représenter et de créer notre "filtre" de pluie.

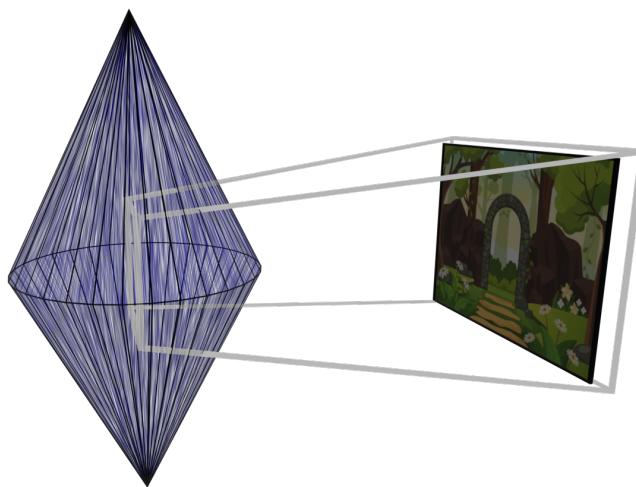


FIGURE 27 – Double cone texturé centré sur la caméra.

4.1.3 Les layers

Bien que la méthode que nous avons pour le moment puisse théoriquement donner des résultats convaincants quelle que soit la position et l'inclinaison de la caméra et la direction du vent, elle ne permet pas de prendre en compte réellement les lumières et objets transparents de la scène, les collisions avec celle-ci et les perturbations locales du vent. Tout cela est dû au fait qu'on ne puisse pas associer à nos gouttes une position en espace monde car elles sont générées à l'aide d'un bruit 2D.

La solution qu'on apporte, consiste à découper notre frustum en différents layers. Les premiers layers seraient en charge de rendre différentes "couches" de pluie et les derniers les différentes "couches" de brumes. La nécessité des différentes "couches" de brume sera expliquée dans une partie dédiée. Gardez seulement à l'esprit que spatialement ces couches se situent après les layers de pluie. La simulation ressemblerait alors à ce qui est représenté sur la figure 28.

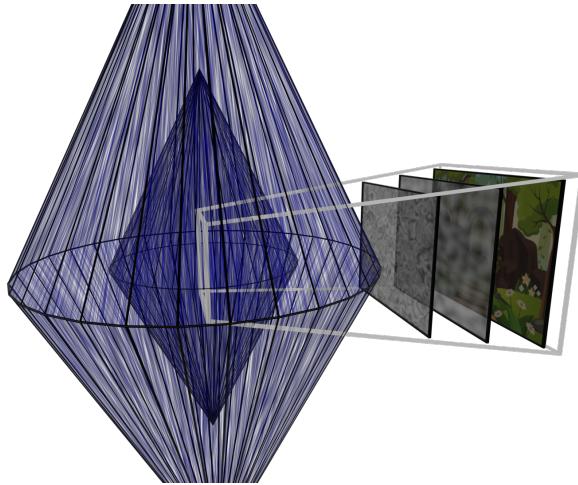


FIGURE 28 – Schéma des différents layers de pluie.

Maintenant, nous pouvons définir une distance entre chaque layer et pour chacun d'entre eux, leur associer une distance à la caméra ainsi qu'une carte de densité qui sera propre à l'espace dont il aura la charge. Le frustum serait alors découpé en différentes parties et chaque layer serait en charge de rendre la pluie pour sa part du frustum. L'idée maintenant serait pour chaque pixel des layers, d'approximer l'illumination présente dans l'espace dont il est en charge. C'est un peu comme si nous avions un frustum voxélisé où on pouvait associer un voxel à chaque pixel de chaque layer. Tout ceci est illustré par la figure 29. Nous allons donc considérer que chaque kernel contenu dans un même "voxel" est illuminé de la même manière.

Pour définir l'illumination de chaque voxel, il nous faut prendre une position à l'intérieur de celui-ci et trouver la quantité de lumière qui lui parvient à l'aide des différentes informations de la scène. On peut tout à fait choisir de prendre plusieurs points et de moyenner leurs résultats pour tendre vers une meilleure approximation.

Un problème va se poser, comment va t-on échantillonner ces layers ? On pourrait leur associer à chacun un double cône sur lequel on plaqueraient le layer. Malheureusement l'utilisation des doubles cônes ne va pas se faire sans soucis. Premièrement pour donner un effet de parallaxe aux gouttes qui vont tomber depuis les différents layers, il va falloir effectuer une homothétie de plus en plus importante pour les cônes étant de plus en

plus éloignés de la caméra. Et en même temps il faudra adapter le sampling des textures procédurales qui leur sont associé afin que les cordes paraissent avoir toujours une taille cohérente une fois plaquées sur les cônes. Ensuite, une fois les différents doubles cônes rastérisés, les fragments générés relatifs à un même pixel auront des UVs très similaires, une duplication de la texture sera alors nettement visible sur l'image finale. Pour résoudre ce problème, deux approches sont possibles : soit nous choisissons d'appliquer un offset au moment du sampling du bruit 2D qui dépendra de l'indice du layer et de la distance entre les layers, soit nous choisissons de changer notre bruit 2D pour un bruit 3D.

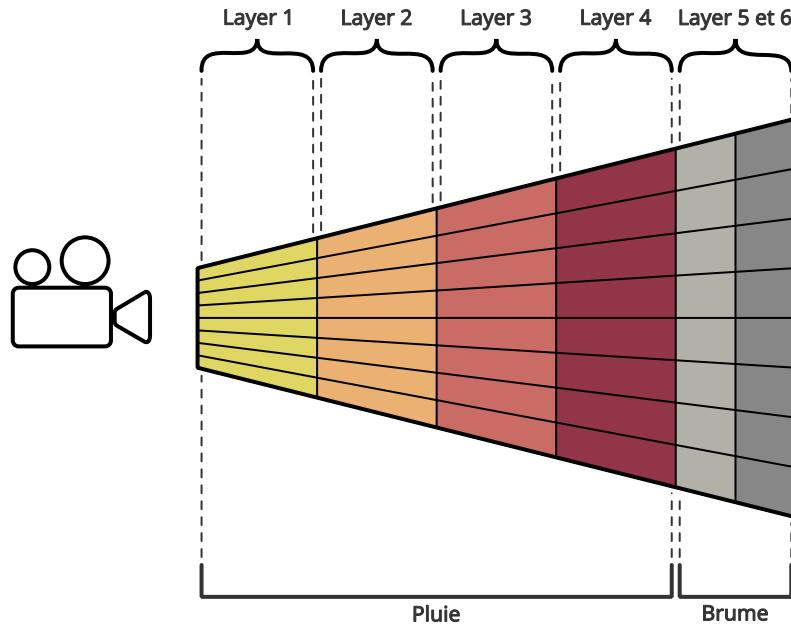


FIGURE 29 – Répartition des voxels du frustum en layers.

La première approche est très simple et règle le problème sans avoir à changer drastiquement la méthode, et semble donc être celle à privilégier. Mais avant de faire notre choix, penchons nous sur les problèmes qu'il nous reste à régler. Même si nous pouvons maintenant associer la profondeur d'un layer à chaque corde qu'il représente cela reste une approximation trop grossière pour espérer avoir une gestion des collisions précise avec la scène et une prise en compte des perturbations locales du vent. C'est en cela que l'utilisation d'un bruit 3D peut être intéressant, car contrairement à une bruit 2D, un bruit 3D est défini en tout point de l'espace tridimensionnel (figure 30).

L'utilisation d'un bruit 3D soulève en réalité plus de questions que de réponses apportées. Nous sommes en droit de nous demander :

- Comment sampler efficacement un bruit 3D dans notre espace ?
- Peut-on réutiliser le LRPN ? Si oui, comment adapter nos kernels à un bruit 3D ?
- Comment déformer notre bruit 3D pour permettre d'appliquer les perturbations locales du vent ?
- Est-il toujours utile d'utiliser des layers et des cônes ?
- N'est-il pas préférable d'utiliser des particules ?

Nous ne nous sommes pas encore assez penchés sur la question pour pouvoir y répondre, mais il peut être intéressant de creuser.

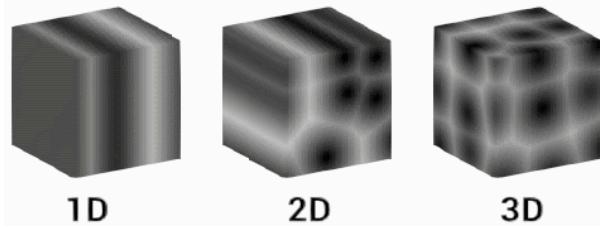


FIGURE 30 – Différence entre un bruit 1D, 2D et 3D.
Image tirée du site web d'[unity](#).

4.2 Seconde méthode : Utilisation d'un système de particules

La méthode que nous développerons dans cette partie découle de l'intérêt tout particulier que nous avons porté aux travaux de Pierre Rousseau [Rou07]. Ses travaux ont montré qu'il est tout à fait envisageable d'utiliser un système de particule pour représenter de la pluie dans des applications temps réel et cela en plus de prendre en compte les interactions de chacune des cordes avec le vent tout en réalisant un rendu physiquement réaliste de celles-ci dans des scènes plus ou moins complexes.

Bien que sa méthode paraisse déjà très complète en elle-même, de nombreuses améliorations peuvent être envisagées. En effet Pierre Rousseau a choisi de se concentrer uniquement sur le rendu des cordes pour une intensité de pluie constante. Et sa méthode n'intègre aucune autre simulation de phénomène qui pourrait survenir durant la pluie. Nous proposons alors de reprendre ses travaux en les mixant avec ceux de Yoann Weber et en y ajoutant au passage quelques améliorations développées pour l'occasion. Nous obtiendrons un modèle de pluie permettant de rendre des cordes, des égouttements ainsi que de la brume au loin, le tout régit par un nombre limité de paramètres intuitifs. Tout comme pour la méthode précédente, afin de bien comprendre comment elle marche, nous allons la construire pas à pas ensemble.

4.2.1 Création d'une boîte à goutte

Avant toute chose commençons par définir ce que l'on entend par particule. Nos particules représentent nos gouttes que l'on peut définir comme ayant une position, une vitesse, un rayon/diamètre et un état (mort ou vivant).

L'idée de la méthode de Pierre Rousseau [RJG08] est de lancer une simulation avec un nombre restreint de particules. Toutes ces particules ne pourront évoluer qu'à l'intérieur d'une boîte qui contient le frustum et appelée "boîte à goutte". Lorsque les gouttes sortiront de la boîte, elles seront immédiatement recyclées à l'intérieur de la boîte à goutte. L'utilisation d'une telle boîte permet de n'avoir à gérer que des particules qui ont une chance d'être visibles à l'écran. Pour bien faire, l'idée serait de créer une boîte à goutte qui collerait parfaitement au frustum pour réduire le plus possible les calculs inutiles.

Calcul du mouvement des gouttes

Avoir des gouttes statiques contenues dans une boîte c'est bien, mais cela ne suffit pas pour simuler de la pluie. Il va nous falloir trouver un moyen de leur appliquer des forces qui changeront leur position. Nous souhaiterons donc connaître pour chaque temps t les positions de nos gouttes à l'instant suivant $t + \Delta t$. Or nous savons que l'accélération est la dérivée de la vitesse par le temps et la vitesse est la dérivée de la position par le temps. On note aussi : $\frac{\partial \vec{x}_t}{\partial t^2} = \frac{\partial \vec{v}_t}{\partial t} = \vec{a}_t$. Comme nous cherchons à obtenir $x_{t+\Delta t}$ il nous faut

d'une, intégrer l'accélération et la vitesse et de deux, trouver l'accélération. Pour cela, la seconde loi de Newton nous indique que la somme des forces appliquées à un corps est égale à sa masse multipliée par son accélération ($\sum \vec{F}_{ext} = m\vec{a}$). Et pour intégrer, nous pouvons partir de différentes méthodes d'intégration :

Euler explicite :

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_t \Delta t$$

$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_t \Delta t$$

Euler semi-implicite :

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_t \Delta t$$

$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_{t+\Delta t} \Delta t$$

Euler implicite :

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_{t+\Delta t} \Delta t$$

$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_{t+\Delta t} \Delta t$$

$$\text{Runge-kutta 4 : } \vec{x}_{t+\Delta t} = \vec{x}_t + \frac{1}{6}(\vec{v}_t + 2\vec{v}_{t+0.25\Delta t} + 2\vec{v}_{t+0.5\Delta t} + \vec{v}_{t+0.75\Delta t})\Delta t$$

$$\text{Verlet : } \vec{x}_{t+\Delta t} = 2\vec{x}_t - \vec{x}_{t-\Delta t} + \vec{a}_t \Delta t^2$$

A l'aide de ces équations et à partir du moment où nous connaissons les différentes forces qui sont appliquées à notre système, ainsi que la masse de nos particules, nous sommes en mesure de trouver leurs futures positions. Commençons donc par calculer leur masse. En admettant que nos gouttes soient constituées uniquement d'eau pure et qu'elles contiennent un volume d'eau équivalent à une sphère de diamètre D . On a :

$$\begin{aligned} V_{sphere} &= \frac{4}{3}\pi r^3 & (1L_{eau} = 1Kg = 1dm^3) \\ V_{goutte} &= \frac{4}{3}\pi\left(\frac{D}{2}\right)^3 & (D \text{ en mm}) \\ &= \frac{4}{3}\pi\left(\frac{10^{-2}D}{2}\right)^3 & (V_{goutte} \text{ en } dm^3) \\ m_{goutte} &= \frac{1}{6}\pi D^3 10^{-6} & (m_{goutte} \text{ en Kg}) \end{aligned}$$

Maintenant écrivons le bilan de nos forces en présence.

$$\begin{aligned} m\vec{a} &= \sum \vec{F}_{ext} \\ m\vec{a} &= \vec{P} + \vec{F}_{vent} + \vec{F}_{frottement} \\ m\vec{a} &= m\vec{G} + \alpha \frac{\vec{W}_D}{\|\vec{W}_D\|} + \beta \frac{\vec{W}_L(x, t)}{\|\vec{W}_L(x, t)\|} - k\|\vec{v}\|\vec{v} \\ \vec{a} &= \vec{G} + \frac{\alpha \frac{\vec{W}_D}{\|\vec{W}_D\|} + \beta \frac{\vec{W}_L(x, t)}{\|\vec{W}_L(x, t)\|} - k\|\vec{v}\|\vec{v}}{m} \end{aligned}$$

Avec $\vec{P} = m\vec{G}$ la force de pesanteur, où m est la masse de la particule et \vec{G} la constante de gravité de la Terre soit : $g = 9.81$ (Newton) multipliée par la direction vers le centre de la Terre. $\alpha \frac{\vec{W}_D}{\|\vec{W}_D\|} + \beta \frac{\vec{W}_L(x, t)}{\|\vec{W}_L(x, t)\|}$ la force du vent que l'on décompose en deux phénomènes ; une direction globale (\vec{W}_D) et des perturbations locales ($\vec{W}_L(x, t)$). $\vec{F}_{frottement} = -k\|\vec{v}\|\vec{v}$ la force de friction de l'air qui est ici quadratique où \vec{v} est la vitesse de la particule. Et nous avons aussi α la force en Newton qu'applique le vent directionnel à la goutte, β la

force en Newton qu'appliquent les perturbations locales du vent à la goutte et $-k$ la force en Newton qu'applique le frottement de l'air à la goutte.

Dans cette équation nous avons choisi d'utiliser des forces de frottement de l'air quadratique tout simplement car les forces de frottement de l'air linéaire ($-k\vec{v}$) ne s'appliquent qu'en cas de vitesse très faible des particules. Dans notre cas, la différence entre les deux types de force de frottement reste assez imperceptible pour ces faibles vitesses. Il est donc préférable pour nous d'utiliser des forces de frottement quadratique quelle que soit la vitesse de la particule.

Il nous faut clarifier ce que représentent \vec{W}_D et $\vec{W}_L(x, t)$. D'abord \vec{W}_D représente une direction globale du vent multipliée par une vitesse. De la même manière $\vec{W}_L(x, t)$ représente un champ de vecteur pouvant varier au cours du temps, où chaque vecteur de ce champ représente une direction du vent multipliée par une vitesse.

A présent afin de pouvoir utiliser cette équation il nous faut trouver comment calculer α , β et $-k$. La formule utilisée pour calculer $-k$ est la suivante : $k = \frac{1}{2}C_x\rho S$ où ρ est la masse volumique de l'air que l'on fixera à $1.226 \text{ (kg/m}^3)$ qui est une approximation de sa valeur au niveau de la mer pour une température allant de 0 à 20 degré Celsius et une humidité variant de 0 à 100% . S est le maître couple soit l'aire de la section transversale de la goutte. Si on admet que notre goutte reste en toute circonstance ronde et qu'elle a pour diamètre D , on peut alors déduire que le maître couple de la goutte serait l'aire d'un cercle de diamètre D . Et C_x est le coefficient de traînée de la goutte, c'est un coefficient qui est très compliqué à déterminer analytiquement. Heureusement de nombreuses études en soufflerie ont été menées et nous indiquent que pour notre cas d'utilisation il est possible d'approximer C_x avec une constante variant entre 0.4 et 1 . Nous pouvons donc déduire k comme ceci :

$$\begin{aligned} k &= \frac{1}{2}S\rho C_x && (k \text{ en N}) \\ k &= \frac{1}{2} \times (\pi(\frac{10^{-3}D}{2})^2) \times 1.226C_x && (S \text{ en m}^2) \\ k &= \frac{1}{8}\pi 10^{-6}D^2 1.226C_x \end{aligned}$$

Il ne nous reste alors plus qu'à calculer α et β . Ces deux constantes représentent toutes deux la force en Newton du vent qui sera appliquée à la goutte. On peut le calculer à l'aide d'une formule assez similaire à la précédente : $F = SP C_x$. P représente la pression exercée par le vent en Pascal et dans notre cas on a $P = 0.613V^2$ où V est la vitesse du vent en $m s^{-1}$. Et tout comme la formule pour le calcul de k , S représente le maître couple et C_x le coefficient de traînée. En remplaçant nos variables on obtient alors :

$$\begin{aligned} \alpha &= (\pi(\frac{10^{-3}D}{2})^2) \times (0.613\|\vec{W}_D\|^2) \times C_x && (S \text{ en m}^2) \\ \beta &= (\pi(\frac{10^{-3}D}{2})^2) \times (0.613\|\vec{W}_L(x, t)\|^2) \times C_x && (S \text{ en m}^2) \end{aligned}$$

Si on reprend notre équation de l'accélération et qu'on remplace les constantes par leurs valeurs que l'on vient de calculer on obtient :

$$\begin{aligned}
\vec{a} &= \vec{G} + \frac{\alpha \frac{\vec{W}_D}{\|\vec{W}_D\|} + \beta \frac{\vec{W}_L(x,t)}{\|\vec{W}_L(x,t)\|} - k\|\vec{v}\|\vec{v}}{m} \\
\vec{a} &= \vec{G} + \frac{\left(\frac{0.613\pi D^2 C_x}{4 \cdot 10^6} \|\vec{W}_D\|^2\right) \frac{\vec{W}_D}{\|\vec{W}_D\|} + \left(\frac{0.613\pi D^2 C_x}{4 \cdot 10^6} \|\vec{W}_L(x,t)\|^2\right) \frac{\vec{W}_L(x,t)}{\|\vec{W}_L(x,t)\|} - \left(\frac{1.226\pi D^2 C_x}{8 \cdot 10^6}\right) \|\vec{v}\|\vec{v}}{\frac{\pi D^3}{6 \cdot 10^6}} \\
\vec{a} &= \vec{G} + \frac{6 \cdot 10^6 \times 0.613\pi D^2 C_x}{4 \cdot 10^6 \pi D^3} (\|\vec{W}_D\|\vec{W}_D + \|\vec{W}_L(x,t)\|\vec{W}_L(x,t)) - \frac{6 \cdot 10^6 \times 1.226\pi D^2 C_x}{8 \cdot 10^6 \pi D^3} \|\vec{v}\|\vec{v} \\
\vec{a} &= \vec{G} + \frac{6 \times 0.613C_x}{4D} (\|\vec{W}_D\|\vec{W}_D + \|\vec{W}_L(x,t)\|\vec{W}_L(x,t)) - \frac{6 \times 1.226C_x}{8D} \|\vec{v}\|\vec{v} \\
\vec{a} &= \vec{G} + 0.9195 \frac{C_x}{D} (\|\vec{W}_D\|\vec{W}_D + \|\vec{W}_L(x,t)\|\vec{W}_L(x,t) - \|\vec{v}\|\vec{v})
\end{aligned}$$

A présent, puisque les méthodes d'intégration que nous allons utiliser sont des suites, il nous faut définir les valeurs initiales de celles-ci. Et comme la goutte une fois créée est censée déjà avoir initié sa chute depuis bien longtemps, nous pouvons supposer que :

- cette chute n'a été influencée que par des forces constantes (\vec{G} et \vec{W}_d) et par les forces de friction de l'air ($-k\vec{v}^2$).
- la particule a eu le temps d'atteindre une vitesse à partir de laquelle les forces de friction de l'air compensent les autres forces en présence.

On aurait alors $\vec{a}_0 = \vec{0}$. Et à l'aide de l'équation de \vec{a} au dessus on peut en déduire \vec{v}_0 comme ceci :

$$\begin{aligned}
\vec{a} &= \vec{G} + 0.9195 \frac{C_x}{D} (\|\vec{W}_D\|\vec{W}_D + \|\vec{W}_L(x,t)\|\vec{W}_L(x,t) - \|\vec{v}\|\vec{v}) \\
\vec{a}_0 &= \vec{G} + 0.9195 \frac{C_x}{D} (\|\vec{W}_D\|\vec{W}_D - \|\vec{v}_0\|\vec{v}_0) \\
\vec{0} &= \vec{G} + 0.9195 \frac{C_x}{D} (\|\vec{W}_D\|\vec{W}_D - \|\vec{v}_0\|\vec{v}_0) \quad (\text{avec } \vec{a}_0 = \vec{0}) \\
\|\vec{v}_0\|\vec{v}_0 &= \frac{D}{0.9195C_x} \vec{G} + \|\vec{W}_D\|\vec{W}_D \\
\|\vec{v}_0\| &= \sqrt{\left\| \frac{D}{0.9195C_x} \vec{G} + \|\vec{W}_D\|\vec{W}_D \right\|} \quad (\text{avec } \|\vec{v}_0\|^2 = \|\|\vec{v}_0\|\vec{v}_0\|) \\
\vec{v}_0 &= \frac{\frac{D}{0.9195C_x} \vec{G} + \|\vec{W}_D\|\vec{W}_D}{\sqrt{\left\| \frac{D}{0.9195C_x} \vec{G} + \|\vec{W}_D\|\vec{W}_D \right\|}} \quad \left(\text{avec } \vec{v}_0 = \frac{\|\vec{v}_0\|\vec{v}_0}{\|\vec{v}_0\|} \right)
\end{aligned}$$

Nous avons donc déterminé nos conditions initiales pour \vec{a}_0 et \vec{v}_0 et la manière dont x_0 est choisi sera déterminée plus loin dans le rapport. Vient maintenant la question du choix de la méthode d'intégration. Contrairement à certaines simulations comme celles de tissu ou de fluide, nous n'avons pas forcément besoin d'une méthode extrêmement précise. Nous pouvons donc nous permettre de sacrifier un peu de précision afin de réduire les calculs nécessaires. Nous allons donc éliminer les méthodes implicites comme celle d'Euler et de Runge-Kutta qui donnent des résultats très précis mais demandent vraiment davantage de calculs. Les trois dernières méthodes qu'il reste demandent les mêmes quantités de calculs

et stockent un nombre de données équivalent. Il faut donc trouver un autre moyen de les départager. Pour bien faire notre choix il faudrait benchmarker les méthodes. Ceci dit, le choix de la méthode d'intégration n'étant pas un élément central dans notre méthode nous avons choisi d'utiliser l'intégration de Verlet qui est souvent plus stable et robuste que les méthodes d'Euler.

Détection de collision entre gouttes et boîte à goutte

Maintenant que nos particules peuvent évoluer dans notre monde, rien ne les empêchent de sortir de la boîte à goutte. Puisque nous souhaitons "recycler" les gouttes se trouvant en dehors de la boîte à goutte, il nous faut d'abord pouvoir déterminer comment détecter si une goutte est sortie de celle-ci. Puisque la boîte à goutte est construite de manière à coller au frustum, on peut en déduire qu'elle sera toujours de type hexahédrique et ses six faces seront toujours rectangulaires. De ce fait, pour chacune des faces d'une boîte à goutte il nous est possible de construire un plan à l'aide d'un point sur celui-ci et de sa normale.

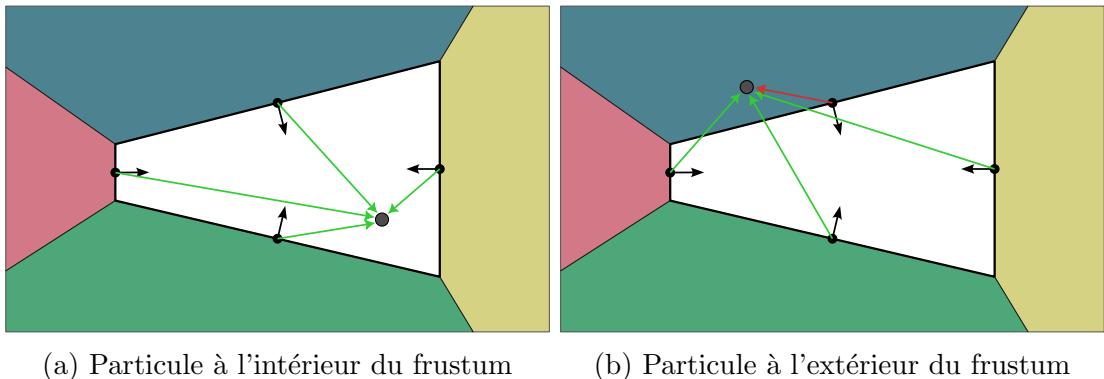


FIGURE 31 – Test d'appartenance d'une particule au frustum

Le schéma 31 a été réalisé en 2D dans le but de simplifier la compréhension de la méthode. Nous pouvons observer notre frustum sous forme de trapèze isocèle. Sur chacune de nos faces, en noir, on trouve les normales de nos plans partant du point milieu de la face. En testant le signe du cosinus entre la normale de notre plan et le vecteur partant d'un point appartenant à notre plan et allant vers un point à tester nous pouvons déterminer de quel côté du plan ce point se situe. Pour qu'un point soit considéré comme étant à l'intérieur de la boîte il faut que les six tests (quatre en 2D) avec les plans renvoient tous des chiffres positifs. On voit bien sur la figure 31a que les quatre tests nous fournissent des résultats positifs et que notre particule est bien à l'intérieur du frustum. Et à l'inverse sur la figure 31b l'un des quatre test nous renvoie une valeur négative et notre particule est bien à l'extérieur de notre frustum.

Recyclage des gouttes

Pour l'instant, comme nous souhaitons toujours avoir le nombre maximal de goutte à l'intérieur de notre boîte à goutte, quand une goutte est détectée comme étant sortie de la boîte, il nous faut lui réassigner une nouvelle position à l'intérieur de celle-ci. Si nous choisissons de lui donner une position déterminée au hasard à l'intérieur de la boîte, nous pourrions observer que des gouttes peuvent apparaître de n'importe où et semblent émerger du néant. Cet effet n'est pas souhaitable dans notre cas. Nous souhaiterions plutôt

que les gouttes semblent venir d'en dehors du champ visuel de l'observateur. Pour cela il est donc mieux de les faire apparaître proches des bords de la boîte à goutte.

Prenons un exemple pour illustrer la solution trouvée (figure 32). Prenons les 6 plans de la boîte à gouttes numérotés de 1 à 6 et calculons le cosinus entre le vecteur et la normale à chacun de ces plans et \vec{v}_0 . Nous obtiendrons des valeurs comprises entre $]-\infty, +\infty[$ puisque la valeur de \vec{v}_0 n'est pas normalisée. Notre objectif est de sélectionner un plan à partir duquel les nouvelles gouttes vont être générées. Il n'est donc pas souhaitable de générer des gouttes à partir des plans n'allant pas dans la même direction que notre vecteur car les gouttes seraient générées à l'extérieur du frustum. C'est pour cela que nous bornerons à 0 les valeurs des cosinus. Ensuite pour chaque plan nous allons calculer les valeurs cumulées obtenues par les plans ayant un numéro plus petit que lui et afin de normaliser les valeurs nous diviserons le résultat par la valeur obtenue par le 6ème plan. Nous obtenons alors une fonction discrète cumulative de densité (CDF). Il nous suffit alors pour échantillonner cette fonction discrète de tirer un nombre aléatoire entre 0 et 1, d'itérer sur notre tableau de 6 valeurs croissantes et de s'arrêter lorsque le nombre contenu dans le tableau est plus grand ou égal au nombre aléatoire. Nous sélectionnerons alors la face associée à cette case du tableau.

Plans	n°1	n°2	n°3	n°4	n°5	n°6
cos(Normal,v0)	-0,030	0,164	0,340	-0,120	-0,600	0,237
Cumule des valeurs précédentes	0,000 + 0,164 + 0,340 + 0,000 + 0,000 + 0,237					
Total	0,000	0,164	0,504	0,504	0,504	0,741
Total / Total n°6	0,000	0,221	0,680	0,680	0,680	1,000

FIGURE 32 – Exemple de calcul de la fonction discrète cumulative de densité

Comme nous savons que la face choisie est rectangulaire, il est facile de sélectionner un point aléatoire sur cette face. Il suffit de prendre deux valeurs u et v aléatoires entre 0 et 1 et de réaliser une interpolation linéaire des quatre sommets du rectangle avec ces valeurs. Il nous faut à présent déterminer un offset à cette nouvelle goutte afin d'une part, éviter les problèmes d'imprécision mathématique lors du test de collision avec le frustum et d'autre part, que les gouttes générées lors du même pas de temps ne soient pas alignées sur un même plan. Pour cela nous allons déplacer la particule en direction de la normale au plan d'un minimum d'une valeur epsilon très petite de l'ordre de 10^{-4} et d'une valeur maximale du projeté du vecteur $\Delta t \vec{v}_0$ sur la normale au plan (ce qu'on peut simplement exprimer par $(\Delta t \vec{v}_0) \cdot \vec{N}$ puisque la normale \vec{N} est normalisée). Cette valeur maximale simule le fait qu'à l'instant $t - \Delta t$ la particule se situait forcément sur la droite ayant pour vecteur directeur v_0 et passant par le point x_0 à une distance maximale lui permettant pour cet instant t d'apparaître dans la boîte à goutte.

Avec cette méthode on peut observer quelques problèmes. A cause de l'utilisation d'un offset, certaines gouttes peuvent se retrouver en dehors de la boîte à goutte à la suite de leur recyclage. Dans le même ordre d'idée, il peut arriver que des gouttes avec une trop grande vitesse traversent entièrement le champ de vision dès leur création. Il faut donc bien penser à la suite de la génération de la nouvelle goutte, de réaliser un test de sortie de la boîte à goutte pour tuer celles se retrouvant à l'extérieur.

Ensuite, lorsque la caméra se déplace ou bien tourne, on peut parfois remarquer une baisse de la densité des gouttes sur les côtés de la boîte à goutte (image 33a). Cela vient du fait que dans cette partie de l'espace il n'y a de base aucune goutte et il faut attendre que les gouttes y accèdent par elles-mêmes. Cela peut amener à briser la sensation d'immersion. Dans ce cas, on peut tout à fait se figurer que prendre en compte d'une quelconque manière le mouvement de la boîte à goutte lors du recyclage des goutte pourrait aider à résoudre le problème. Pour illustrer cela, les images de la figure 33 proviennent de capture de la même simulation et ont été prises au même moment. Dans cette simulation on retrouve une boîte à goutte allant de droite à gauche, un vent directionnel allant dans le même sens et la force de gravité allant de haut vers le bas. Sur l'image 33a, les gouttes évoluent dans la boîte à goutte à l'aide des forces décrites précédemment et sont recyclées avec la méthode qui vient tout juste d'être expliquée. On remarque que la partie gauche de la boîte à goutte contient une espace dénué de goutte. Sur l'image 33b, les gouttes évoluent de la même manière que sur l'image précédente, par contre la manière dont est choisie la face pour créer les gouttes et le choix de l'offset prennent en compte le déplacement de la boîte à goutte. Sur cette image l'espace qui ne contenait auparavant pas de goutte en contient désormais.

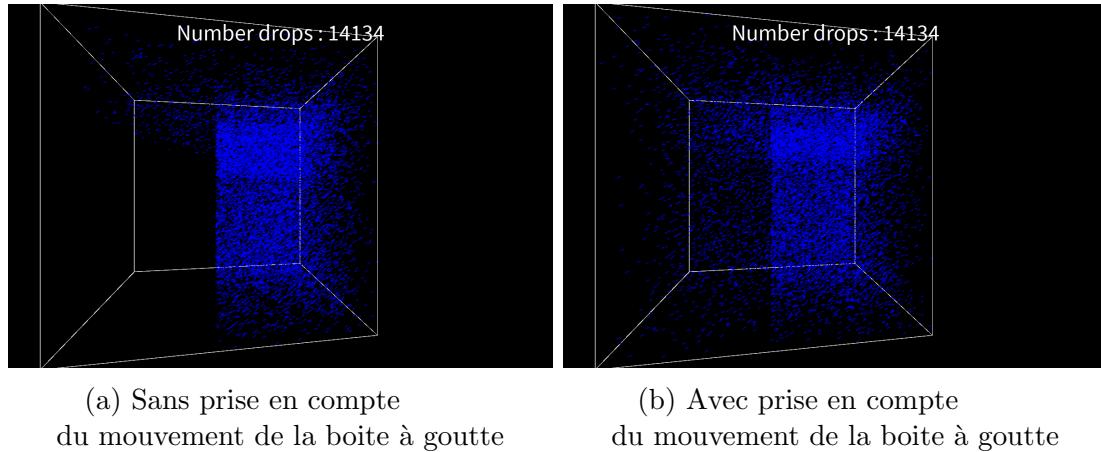


FIGURE 33 – Différence de densité des gouttes à l'intérieur de la boîte à goutte

4.2.2 Prise en compte de l'intensité de la pluie

Jusque là, le nombre de particules contenues dans notre boîte à goutte était constant. Maintenant nous allons faire en sorte que ce nombre puisse évoluer en fonction de l'intensité de la pluie souhaitée. Pour commencer sur de bonnes bases nous pouvons partir de la distribution de Marshall et Palmers ([MP48]) :

$$N(D) = N_0 e^{-\Lambda D}$$

Où $N(D)$ nous donne pour un diamètre D de goutte, le nombre de goutte par unité d'espace qui doit y être contenu. Ici $N_0 = 8000$ et $\Lambda = 4.1I^{-0.21}$ pour I l'intensité de

la pluie. Cette équation a été massivement reprise par la suite notamment par Garg et Nayar, Pierre Rousseau ou encore Yoann Weber du fait de sa précision pour les gouttes au delà de 1.5mm de diamètre et de sa simplicité de calcul par rapport aux autres équations existantes. Cette équation a été ensuite généralisée par les météorologues étudiant le phénomène ([Ulb83]) vers cette équation :

$$N(D) = N_0 D^\mu e^{-\Lambda D}$$

Où μ est un nouveau paramètre pouvant se retrouver dans le calcul de Λ et N_0 et complexifiant le calcul mais en devenant beaucoup plus précis pour les valeurs de D en dessous de 1.5mm. Dans notre cas d'utilisation, nous n'avons pas nécessairement envie de simuler les gouttes les plus petites. Leur simulation et shading pourrait signifier un surcoût énorme pour un résultat sensiblement égal. De ce fait à partir de maintenant nous allons utiliser l'équation de Marshall et Palmers que nous noterons $N(D, I)$.

Puisque notre équation $N(D, I)$ nous indique le nombre de goutte de diamètre D par m^3 d'espace. Il nous est possible d'en déduire le nombre de goutte que nous devrions avoir dans notre simulation pour une intensité de pluie donnée et un intervalle de taille de goutte $[d_{min}, d_{min} + \delta d]$. Il nous suffit alors de calculer la somme de tous les $N(D, I)$ pour D allant de d_{min} à $d_{min} + \delta d$ et de le multiplier par l'aire de notre boîte à goutte. On peut le réécrire comme ceci :

$$Area_{boîte} \int_{d_{min}}^{d_{min} + \delta d} N(D, I) dD$$

Puisque la fonction $N(D, I)$ est de la forme $a e^{bx+c}$ il est facile de retrouver sa primitive ($-\frac{N_0}{\Lambda} e^{-\Lambda D}$) et de surcroît il est très simple de résoudre notre intégrale :

$$\begin{aligned} \int_{d_{min}}^{d_{min} + \delta d} N(D, I) dD &= -\frac{N_0}{\Lambda} e^{-\Lambda(d_{min} + \delta d)} - -\frac{N_0}{\Lambda} e^{-\Lambda d_{min}} \\ &= \frac{N_0}{\Lambda} (e^{-\Lambda d_{min}} - e^{-\Lambda(d_{min} + \delta d)}) \end{aligned}$$

Dans le papier original de Marshall et Palmers, il nous est fourni une autre méthode pour obtenir le nombre de goutte que nous devrions avoir dans notre simulation pour une intensité de pluie donnée. Les auteurs nous indiquent que l'on aurait besoin de calculer uniquement $N(D, I)\delta d$. En traçant la fonction et en la comparant à la nôtre on peut en effet voir que si le paramètre δd est bien choisi, les deux courbes ont un comportement similaire. Cependant lorsque le paramètre n'est pas choisi avec soin leur approximation ne fonctionne plus du tout là où la nôtre reste robuste.

Lorsque l'on regarde plus attentivement la distribution de Marshall et Palmers l'utilisation d'un d_{min} dans nos équations se justifie afin de remédier aux erreurs de la distribution en dessous d'un certain seuil. Cependant l'utilisation d'un δd ne se justifie pas puisque on se rend compte qu'au delà de quelques millimètres, il est presque impossible qu'une goutte d'un tel diamètre existe. Il est donc raisonnable de penser que dans notre intégrale lorsque l'on va faire tendre notre paramètre δd vers l'infini nous allons converger vers une valeur. Cela nous permettra au passage d'enlever un paramètre à la simulation et d'obtenir alors :

$$\begin{aligned} \lim_{\delta d \rightarrow +\infty} \int_{d_{min}}^{d_{min} + \delta d} N(D, I) dD &= \frac{N_0}{\Lambda} (e^{-\Lambda d_{min}} - 0) \\ &= \frac{N_0}{\Lambda} e^{-\Lambda d_{min}} \end{aligned}$$

Nous obtenons donc le nombre de particules contenues dans notre boîte à goutte ayant un diamètre minimal de d_{min} par la formule $Area_{boîte} \frac{N_0}{\Lambda} e^{-\Lambda d_{min}}$.

Au passage, puisque la fonction $N(D, I)$ nous donne la distribution des tailles de gouttes, nous pourrions l'utiliser lorsque l'on génère une nouvelle goutte pour déterminer sa taille. L'idée serait de pouvoir réécrire notre fonction sous la forme d'une pdf que l'on sait échantillonner. Par chance notre fonction ressemble à une distribution de Laplace et nous savons comment échantillonner cette fonction. Nous pouvons donc la réécrire comme ceci :

$$\begin{aligned} N(D) &= N_0 e^{-\Lambda D} \\ &= N_0 \frac{1}{2\lambda^{-1}} 2\lambda^{-1} e^{-\frac{D-0}{\lambda^{-1}}} \\ &= \frac{2N_0}{\lambda} \frac{1}{2\lambda^{-1}} e^{-\frac{D-0}{\lambda^{-1}}} \\ &= \frac{2N_0}{\lambda} f(D|0, \lambda^{-1}) \quad \forall D \geq 0 \end{aligned}$$

Où on a $f(D|0, \lambda^{-1})$ qui correspond à la pdf d'une distribution de Laplace ayant pour paramètre $\mu = 0$ et $b = \lambda^{-1}$. Or on sait échantillonner une telle fonction avec :

$$X = \mu - b \operatorname{sign}(U) \ln(1 - 2|U|) \quad , U \text{ random } \in [-0.5, 0.5]$$

Seulement au final ce que nous voulons, c'est échantillonner $N(D, I)$ dans l'intervalle $[d_{min}, +\infty[$. Pour comprendre comment faire il nous faut manipuler un peu notre équation :

$$\begin{aligned} N(D) &= N_0 e^{-\Lambda D} \\ &= N_0 e^{-\Lambda(D+d_{min}-d_{min})} \\ &= N_0 e^{-\Lambda d_{min}} e^{-\Lambda(D-d_{min})} \\ &= N(d_{min}) e^{-\Lambda(D-d_{min})} \\ &= \frac{2N(d_{min})}{\lambda} f(D|d_{min}, \lambda^{-1}) \quad \forall D \geq d_{min} \end{aligned}$$

Nous avons réussi à associer notre fonction $N(D, I)$, une pdf de Laplace dont la courbe est centrée en d_{min} . Ce que nous voulons c'est un sample sur cette pdf pour les $D \geq d_{min}$. Il nous faut donc juste changer légèrement notre fonction de sampling afin qu'elle puisse satisfaire cette condition. Nous aurons alors :

$$\begin{aligned}
D_{goutte} &= \mu - b \operatorname{sign}(U) \ln(1 - 2|U|) & U \text{ random } \in [0.0, 0.5] \\
&= \mu - b \ln(1 - 2U) & U \text{ random } \in [0.0, 0.5] \\
&= \mu - b \ln(1 - U) & U \text{ random } \in [0.0, 1.0] \\
&= d_{min} - \frac{\ln(U)}{\Lambda} & U \text{ random } \in [0.0, 1.0]
\end{aligned}$$

Avant d'aller plus loin, il nous faut attirer votre attention sur un problème. Si nous utilisons telle quelle cette fonction pour déterminer la taille de nos nouvelles gouttes il se peut (avec une chance infime) que des gouttes très grandes apparaissent (de l'ordre de quelques centimètres voire plus). De telles gouttes ne sont pas souhaitables et peuvent nuire au réalisme de la simulation. De plus dans la réalité, de telles gouttes n'ont jamais été observées lors de période pluvieuse. Nous pouvons donc borner le résultat de la fonction à l'aide d'une valeur maximale qui peut être soit laissée à l'appréciation de l'utilisateur soit fixée en prenant appui sur des observation faites par des météorologue.

Certes, nous pouvons à présent déterminer le nombre de gouttes qui devraient être présentes dans notre simulation en fonction de l'intensité de pluie. Mais comment peut-on faire pour corrélérer ce chiffre avec le nombre de goutte que nous avons effectivement dans notre simulation ? La solution la plus simple serait d'associer à chaque goutte un identifiant, et lorsque l'on change le paramètre I , tuer toutes les gouttes dont l'identifiant serait plus grand que le nombre de goutte désiré et recycler toutes les gouttes mortes qui ont un identifiant plus petit que le nombre de goutte souhaité. Certes le nombre de goutte dans la simulation collerait toujours parfaitement au nombre de goutte désiré mais nous pourrions voir disparaître des gouttes sous nos yeux. Il est donc préférable d'attendre que les gouttes sortent de la boîte à goutte pour décider de les recycler ou non en fonction de leur identifiant.

Avant cette partie, nous avions un nombre fixe de particules à l'intérieur de notre boîte à goutte. L'initialisation de celles-ci était simple, il fallait toutes leur attribuer une position aléatoire dans la boîte à goutte. A présent, nous ne pouvons plus agir de la sorte. Lors du lancement de la simulation ou lors d'un changement de "zone" dans la simulation, si l'intensité de la pluie n'est pas nulle, il nous faut avoir la sensation que les cordes existaient déjà avant notre arrivée. Dans ces cas-là il va donc falloir que nous calculions d'abord notre nombre de goutte devant exister, et ensuite il nous faudra attribuer des positions aléatoires à celles ayant un identifiant inférieur au nombre de goutte que l'on souhaite avoir. Nous n'avons pas encore parlé de comment générer une position aléatoire dans la boîte à goutte. Une première idée très simple peut être mise en oeuvre. Puisque notre boîte à goutte colle à notre frustum, nous pouvons tout à fait générer des positions aléatoires à l'intérieur du frustum en NDC space pour les re-projeter en world space à l'aide des matrices de perspective et de vue de la caméra. Le problème est que la distribution obtenue ne sera alors pas uniforme. Pour cela d'autres méthodes plus complexes existent.

4.2.3 Détection de collision entre les gouttes et les obstacles

Puisque les scènes ne sont que très rarement vides, il est impératif de gérer les collisions avec celles-ci. Nous allons ici supposer que les objets qui peuplent notre scène peuvent bouger ou non et/ou se déformer ou non. Dans le cadre d'une pluie ne prenant pas

en compte les perturbations locales du vent il nous est possible de calculer un vecteur directeur de la pluie de la même manière que nous calculons le vecteur \vec{v}_0 . Comme expliqué dans la première méthode, avec ce vecteur nous sommes capable de construire ce que nous appelons "une carte d'ombrage de pluie" en rendant à nouveau notre scène du point de vue de la pluie. Cette technique est illustrée par la figure 34 (image de gauche). Sur cette figure on voit en noir la délimitation de la carte d'ombrage générée. Cette carte nous permet alors de savoir pour n'importe quel point de la boîte à goutte s'il se situe dans une "ombre" ou non. Ceci est illustré par la figure 34 (image de droite), où en rouge on a les gouttes se situant dans l'ombre. Pour ce cadre d'utilisation bien précis où l'on ignore les perturbations locales du vent, cela nous permet de savoir s'il y a eu une collision avec la scène et en quel point. Grâce à l'utilisation d'un système de particule nous sommes aussi en mesure de fournir tout un tas d'informations qui pourraient être utiles pour générer une réponse à la collision comme : la masse, le diamètre, la vitesse ou l'accélération de la goutte ayant été victime de la collision ; la normale et la position du point d'impact ; et éventuellement l'objet étant victime de la collision. Les réponses à apporter aux collisions peuvent être diverses : créations de nouvelles gouttes, modifications des propriétés de l'objet touché, apparition d'animations d'éclaboussures au niveau de la collision, changement de la trajectoire de la goutte, etc.

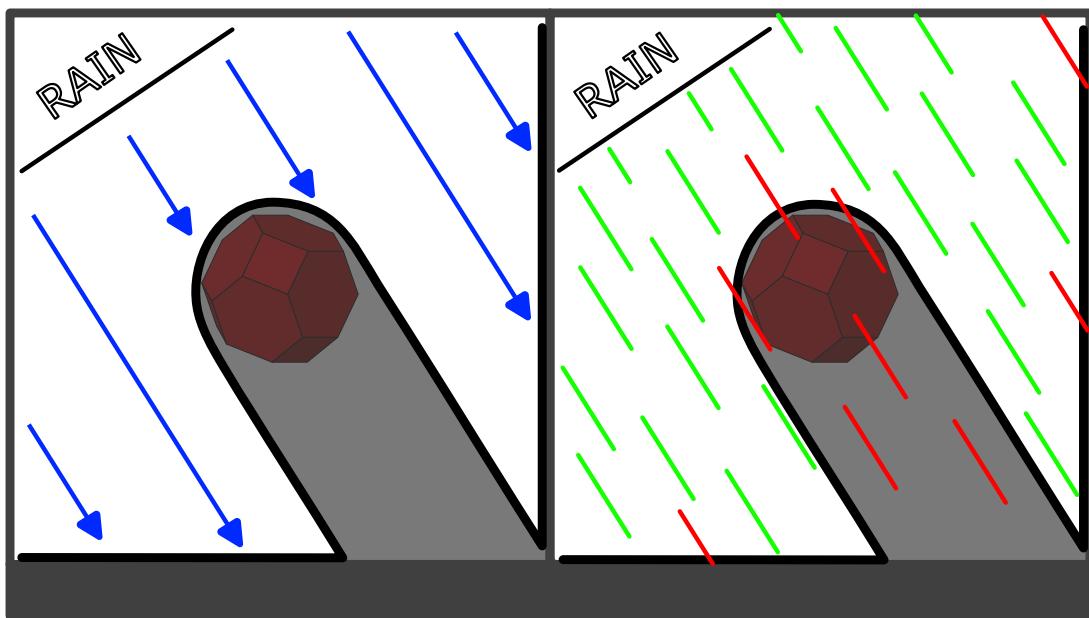


FIGURE 34 – Construction et utilisation de la carte d'ombrage de pluie

Dans le cas où nous gérons aussi les perturbations locales du vent, les gouttes peuvent potentiellement s'insinuer un peu partout. De ce fait il devient impossible de gérer les collisions à l'aide de cette "carte d'ombrage de pluie". Plusieurs alternatives s'offrent à nous.

Nous pouvons soit calculer une carte 3D décrivant pour chaque partie discrétisée de l'espace si elle contient un obstacle ou non. Cette carte peut être calculée à l'aide d'algorithmes utilisés massivement par les logiciels de slicing destinés aux imprimantes 3D. On peut citer entre autres des méthodes comme le Dual Depth Peeling ou des approches basées sur du lancer de rayon. Ces approches sont assez coûteuses en temps de calcul et sont aussi gourmandes en mémoire. De plus elles peuvent comporter des erreurs si la géométrie n'est pas fermée (lorsque l'on ne peut pas distinguer l'intérieur et l'extérieur).

La carte 3D en résultant peut être cependant assez précise mais nécessite d'être recalculée à chaque changement mineur de la scène. Malgré cela des collisions peuvent être manquées lors des mouvements des objets.

Une seconde approche consisterait à simplement considérer la trajectoire entre l'instant t et $t - \Delta t$ de la particule comme un rayon et simplement tester l'intersection entre ce rayon et notre scène. Bien que cette approche puisse être considérée comme coûteuse, la méthode pourrait tirer bénéfices des accélérations matérielles liées au ray tracing. Tout comme les autres méthodes proposées précédemment, des collisions peuvent être manquées lors des mouvements des objets.

Pour éviter de manquer des collisions, les approches classiques optent pour l'utilisation d'un Δt plus petit, ce qui n'est pas forcément la solution la plus adaptée au temps réel. Nous avons imaginé une approche plus envisageable pour notre cas d'utilisation. Si nous disposons de la géométrie d'un objet à l'instant $t - \Delta t$ et à l'instant t , nous sommes en mesure de relier ces deux géométries pour en former une nouvelle qui approximerait l'ensemble des volumes qu'a occupé cet objet durant cet intervalle de temps. Ensuite cette nouvelle géométrie pourrait être utilisée par l'une ou par l'autre des méthodes présentées au-dessus. Grâce à cela, peu d'intersections seraient manquées. Il est aussi possible (pour accélérer ces calculs) d'envisager de réaliser cette méthode, non plus avec la géométrie en tant que telle, mais avec des boîtes englobantes.

La technique de la carte d'ombrage n'est pas non plus à écarter puisque celle-ci peut tout à fait servir lors du recyclage des gouttes, de l'initialisation de la simulation et des changements de zone. A ces moments-là, il nous faut vérifier que les gouttes sont bel et bien générées dans une zone où la pluie a un accès direct. Il est aussi envisageable d'utiliser ici, des approches basées lancer de rayons.

4.2.4 Illumination des gouttes

Par souci de réalisme nous n'allons maintenant plus parler de gouttes mais de cordes. En effet lorsque nous observons de la pluie, ce que nous voyons ce ne sont pas des petites sphères/ellipses comme on pourrait s'y attendre, mais des cordes. Cela est dû à la persistance rétinienne dans le cas de la vision humaine et au temps d'exposition pour un dispositif électronique.

Il faut alors que nous puissions définir la forme des cordes en fonction du temps, de la position et trajectoire de la particule et du temps d'exposition de la caméra. Cette forme reste encore à définir mais nous pouvons nous appuyer sur les travaux réalisés par Garg et Nayar [GN06 ; GN07], Pierre Rousseau [RJG06] et Yoann Weber [Web+15] qui décrivent avec précision la forme qu'une corde peut prendre.

Si nous souhaitons calculer la géométrie de la corde puis la rendre à l'aide du rastériser, cela pourrait s'avérer fort coûteux pour les scènes comportant un grand nombre de particules. De ce côté-là, nous pouvons nous appuyer sur les travaux de Pierre Rousseau qui a opté pour l'utilisation d'imposteur centré sur la position de la particule. Cela permet en plus de réduire la géométrie à rendre, de gérer l'occlusion des cordes par la scène. Cependant, l'utilisation du billboard va encore complexifier la détection de collision puisque la détection va se faire à partir du centre des cordes et non avec toute la surface de l'imposteur. Une solution pourrait être de ne pas centrer le billboard sur la position actuelle de la particule mais plutôt d'utiliser les postions de la particule à l'instant $t - \Delta t$ et t pour délimiter l'imposteur.

Pour le moment nous avons simplement opté pour l'affichage de lignes dont les deux extrémités seraient l'ancienne et l'actuelle position de la goutte. Cette géométrie est très simple et est accélérée par le hardware. Pour simuler la persistance rétinienne on pourrait même imaginer réutiliser les précédentes images générées par les passes de pluies et les mixer avec l'images courante. On pourrait alors obtenir des géométries de cordes plus complexes qu'un simple trait.

Dans sa thèse, Pierre Rousseau nous propose une étude poussée sur la manière dont la réflexion et la réfraction de la lumière participe à l'illumination d'une corde. A la suite de ses études, il conclut que hormis pour les angles rasant, les rayons réfléchis ne participent pas de manière significative à l'apparence d'une goutte.

Il tente ensuite de comprendre la relation qui lie deux rayons entrant et sortant d'une goutte. Premièrement, il en déduit qu'il est possible de pré-calculer des masques indiquant pour un rayon donné, quelle sera la direction du rayon sortant. Et deuxièmement, il en déduit la portion d'une scène qui est "vue" au travers d'une goutte d'eau soit : 165 degré derrière elle par rapport à l'observateur. A l'aide d'une caméra "virtuelle" il est possible de rendre cette partie de la scène pour chaque goutte afin de calculer la manière dont elles sont illuminées. Cette opération s'avérant extrêmement coûteuse, il a choisi de rendre seulement une seconde fois la scène pour l'ensemble des gouttes. Cette approximation donne lieu à des bugs optiques qui induisent que certaine gouttes "voient" des parties de la scène qui leur sont pourtant caché.

C'est pour cela que nous proposons d'utiliser les informations de couleur et de profondeur calculées lors du rendu classique de la scène afin de réaliser des lancers des rayons en utilisant des méthodes Screen Space comme (Screen Space Global Illumination, Screen Space Reflection ou Screen Space Refraction) pour éviter d'avoir à rendre de nouveau la scène et régler ce souci. Ces méthodes n'arrivent pas sans problèmes. Premièrement, comme ce sont des effets Screen Space, si le rayon ne touche rien, cela ne signifie pas qu'il n'existe pas d'obstacle dans le monde, ces obstacles ne sont simplement pas présents dans le frustum. Que faudrait t-il faire dans ce cas ? Renvoyer une couleur fixe ? Sampler l'environnement map ? Prendre la couleur du pixel le plus proche du rayon durant son parcours ? Deuxièmement, du fait que ces effets screen space ne travaillent qu'avec un nombre réduit d'information sur la scène, certains rayons peuvent produire des résultats complètement erronés. Et troisièmement, la méthode de Pierre Rousseau nécessitait l'utilisation d'un seul texel fecht par fragment, là où nous en avons besoin d'une quantité bien plus importante. Pour réellement savoir si ces techniques peuvent être avantageuses, il va falloir les essayer et les comparer.

4.2.5 Améliorations

Bien sûr les méthodes proposées ici ne sont pas parfaites et des améliorations sont d'ores et déjà imaginables.

Forme de boite à goutte

Précédemment dans le rapport, nous avons expliqué qu'il était préférable de simuler uniquement les gouttes qui avaient une chance d'avoir un impact sur l'image finale. De ce fait, nous utilisions jusqu'alors une boite à goutte qui correspondait parfaitement au frustum. Malheureusement cela induit plusieurs problèmes. Premièrement, selon l'orientation

de la caméra, puisque le sampling de la surface du frustum ne dépend pas de la surface de ces faces et à cause de la petite taille de face "near" de la boîte à goutte, on peut observer une très grande quantité de goutte générée proche de l'observateur et placée au centre de l'écran. Pour tenter de régler cela, on peut choisir de changer le sampling de la boîte à goutte en prenant en compte la surface de chaque face. Ensuite, lorsque l'on tourne la caméra, la position de celle-ci ne change pas et pourtant l'ensemble de l'espace visible se retrouve changé. De ce fait une grande partie des gouttes doit être recyclée et une partie du frustum se retrouve temporairement sans goutte. C'est un problème assez similaire à celui que nous avions évoqué dans la partie 4.2.1 lorsque nous parlions du mouvement du frustum. Et dernier problème, les gouttes ne peuvent interagir avec le champ de vecteur qu'à l'intérieur du frustum. Suite à cela, en cas de fortes perturbations locales du vent, on peut observer un grand nombre d'aberrations visuelles auxquelles nous ne pouvons pas remédier dès lors que l'on choisit de garder ce type de boîte à goutte.

C'est cette difficulté qui nous a invité à réfléchir sur l'utilisation d'autres types de boîte à goutte. Pour commencer, il est certain que si l'on souhaite qu'il y ait une interaction avec le champ de vecteur et les particules en dehors du frustum, il va falloir que l'on utilise une boîte à goutte qui englobe le frustum. Aussi, pour minimiser le problème qui survient lors de la rotation de la caméra, il faudrait que la nouvelle boîte à goutte soit centrée sur la position de la caméra. Le but de la manœuvre serait de simuler des gouttes tout autour de la position de la caméra, ainsi lors d'une rotation de la caméra, quelle que soit la partie de l'espace observée, il existerait déjà des gouttes.

La première solution que nous avons essayée est assez simple. Nous avons réutilisé tout le processus de calcul de la boîte à goutte en forme de frustum, mais en changeant la matrice de projection perspective par une matrice de projection orthogonale, ainsi que le plan "near", qui se retrouve derrière la caméra à une distance égale du plan "far". Cela a permis de régler une grande partie de nos problèmes tout en réutilisant l'ensemble des méthodes mises en œuvre jusque là. Malgré cela, des artefacts visuels lors de la rotation de la caméra étaient toujours perceptibles. De plus, il faut à présent simuler un nombre bien plus conséquent de goutte pour une intensité de pluie similaire, puisque l'aire de la boîte à goutte en forme pavée est bien supérieure à son homologue en forme de frustum.

En partant du constat que l'utilisation d'une boîte à goutte en forme de sphère nous permettrait de régler entièrement ce problème de rotation, nous avons changé entièrement notre méthode pour tester ce type de boîte à goutte. En réalité la création et l'utilisation d'un tel type de boîte nous simplifie grandement la vie. Dorénavant nous n'aurons plus besoin de connaître les huit sommets, les six normales et les quatre points associés à chacune des faces de la boîte à goutte, mais uniquement le centre et le rayon de celle-ci. Alors, nous choisissons logiquement comme centre la position de la caméra, et comme rayon une valeur pouvant varier de 0 à la distance au plan "far". Il nous faut aussi changer la manière de détecter qu'une goutte est sortie de la boîte à goutte, pour cela nous pouvons utiliser la fonction de distance signée d'une sphère qui nous indiquera pour chaque point de l'espace si ce point est dans la sphère (la fonction renverra un nombre négatif), sur la sphère (la fonction renverra zéro) ou hors de la sphère (la fonction renverra un nombre positif). Il nous reste maintenant à trouver un moyen de générer des points sur la surface d'une sphère et pour cela le papier [Des04] nous propose une excellente solution.

Au final, lequel des ses types de boite est le meilleur ? Si l'on parle en terme d'aire, la boite en forme de frustum l'emporte sur celle en forme de cube, puisque en fonction de sa fovy elle sera toujours au minimum deux fois moins volumineuse et donc pourrait avoir besoin de simuler au minimum deux fois moins de gouttes pour une intensité de pluie similaire. Celle en forme de sphère l'emporte aussi toujours sur celle en forme de cube puisque elle y est inscrite. La place de première revient tout de même ici au frustum puisque en règle générale son aire est plus petite que celle en forme de sphère. Cependant il y a d'autres aspect à prendre en compte. Par exemple, le fait que la sphère ne souffre d'aucun artefact visuel dû à la rotation de la caméra contrairement aux deux autres types de boite. Ou encore la quantité de calcul qu'il est nécessaire de réaliser, de ce coté-là la sphère gagne encore puisque côté CPU seuls la position de la caméra et le rayon de la sphère sont à déterminer alors que pour les deux autres types, il faut calculer les 8 sommets du frustum ainsi que les normales et les centres des faces. Côté GPU le calcul et la génération sont aussi radicalement plus simples pour la sphère.

La carte d'ombrage de pluie

Dans la partie [4.2.3](#) nous sommes passés assez rapidement sur la création de la carte d'ombrage de pluie et nous avons omis un détail. Pour la créer il nous faut utiliser le vecteur $\vec{v_0}$, seulement ce vecteur n'est pas uniforme pour toutes nos gouttes puisqu'il va dépendre de la taille de celle-ci. Pour pouvoir continuer d'utiliser cette carte d'ombrage, nous avons simplement fait le choix d'utiliser le vecteur $\vec{v_0}$ associé à la taille moyenne de nos gouttes. Il en résultera une carte qui ne contiendra pas exactement le résultat escompté, mais cette approximation reste bien suffisante pour nous.

Affichage des cordes

Durant la frame où les cordes sortent de la boite à goutte, elles sont considérées comme mortes et ne sont pas affichées bien qu'une partie de leur trajectoire se trouve encore dans la boite. Pour palier ce problème, nous nous proposons de rajouter un nouvelle état possible aux gouttes qui aura pour effet de considérer durant le reste de la frame que cette particule est morte sauf durant la phase d'affichage.

A l'heure actuelle nous utilisons des lignes comme géométrie de nos cordes. Lorsque ces cordes foncent vers la caméra nous ne sommes plus en mesure de les percevoir du fait de leur finesse. Lors de pluie intense on peut donc remarquer dans le ciel une zone en forme de croix où aucune corde n'est visible. Pour y remédier il faudrait simplement décaler l'un des deux vertex de la ligne lors de son affichage.

4.2.6 Résultats

Au final nous avons réalisé une implémentation GPU de la méthode dans notre moteur où nous avons choisi d'utiliser une boite à goutte sphérique. Les cordes sont représentées par des lignes reliant leur ancienne position avec celle actuelle. Le shading consiste simplement à rendre chaque goutte comme étant grise et transparente. Les techniques présentées précédemment ont aussi été implémentées. Entre autre, le nombre de goutte dans la boite est corrélé à l'intensité de la pluie et nous pouvons faire varier cette intensité durant la simulation. Les images de la figure [35](#) montrent le rendu de notre pluie dans la célèbre scène nommée "Sponza" pour des intensités de pluie variables.

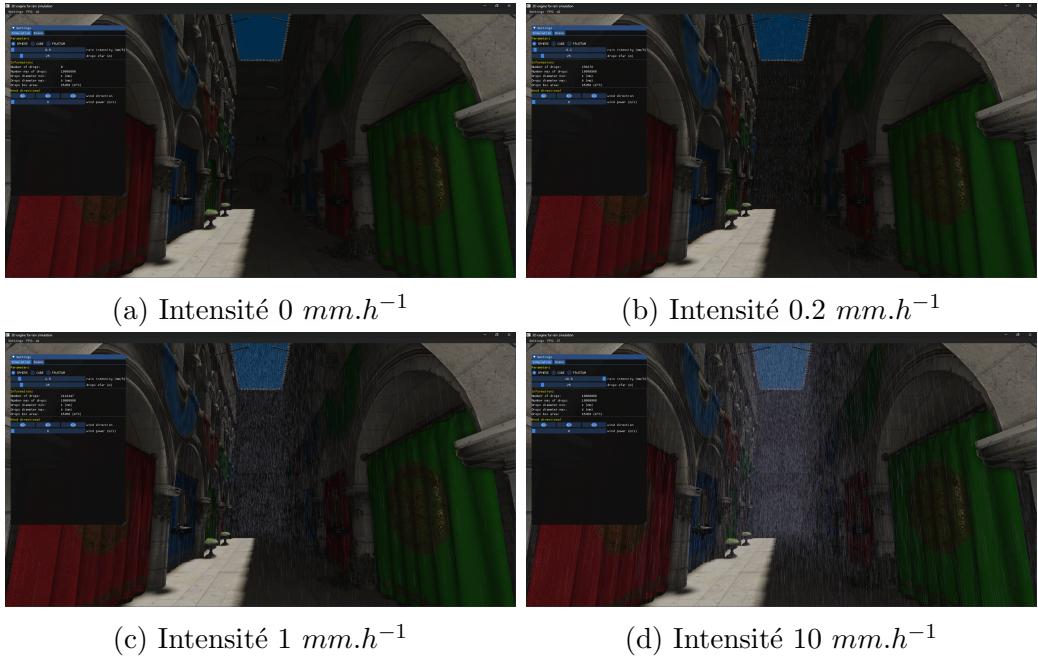


FIGURE 35 – Rendu de la pluie selon différentes intensités

Lorsqu'il n'y a qu'un vent directionnel dans la simulation, le moteur gère les collisions entre les gouttes et les objets de la scène à l'aide d'une carte d'ombrage de pluie. Sur la figure 36 vous pouvez observer le résultat pour différents vents directionnels.

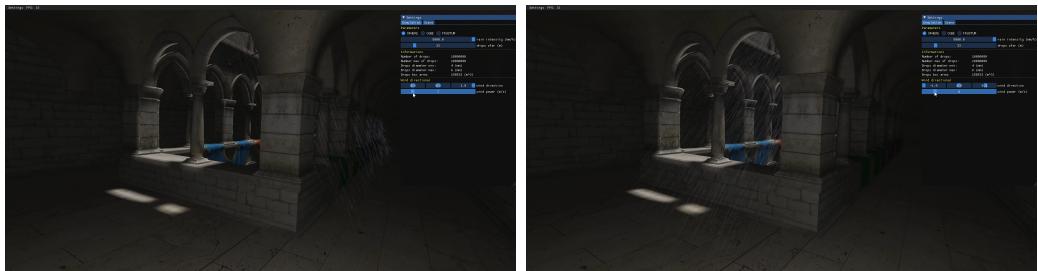


FIGURE 36 – Prise en compte d'un vent directionnel

Et enfin la figure 37 nous montre deux versions de la méthode intégrant la prise en compte des perturbations locales du vent. La première sur processing (à gauche) comprend la gestion des collisions entre les particules et la scène par ray casting. L'autre (à droite) provient de notre moteur et permet l'utilisation d'un bien plus grand nombre de goutte mais ne permet pas pour ce type de vent, la gestion des collisions avec l'environnement.

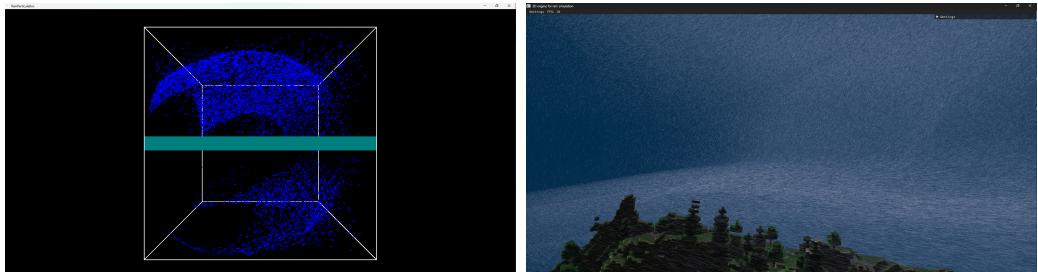


FIGURE 37 – Prise en compte des perturbations locales du vent

4.3 La brume

En raison de l'utilisation d'une vue perspective, lorsque les gouttes sont trop éloignées de l'observateur, la taille projetée sur l'écran des cordes peut passer en dessous de la taille d'un pixel. Au delà de cette distance, les gouttes deviennent difficilement perceptibles individuellement, on identifie alors la pluie comme une sorte de brouillard. Il est important de noter que l'atténuation de visibilité au loin durant les scènes pluvieuses participe grandement au réalisme et donne une sensation de densité du phénomène pour l'observateur.

L'idée suggérée par les travaux de Yoann Weber [Web16] serait de rendre ces gouttes sous la forme d'un brouillard modélisé par un milieu participant. Ce modèle permet à la fois de n'avoir à calculer qu'un nombre de particules restreint (pour les simulations utilisant des particules), mais aussi, il permet de garder la sensation que la pluie se prolonge tout le long du champ de vision. En plus de cela, le brouillard est régi par les mêmes paramètres que notre simulation et s'adapte au changement d'intensité de pluie et de taille minimale et maximale de goutte. En revanche cette méthode comporte quelques inconvénients. Pour simplifier le rendu, l'influence du vent sur ces cordes a été négligée. Les cordes sont alors, considérées comme tombant toujours de haut bas de notre écran. De ce fait, quelle que soit la direction vers laquelle la caméra regarde, la brume tiendra certes compte de la géométrie du terrain, mais ne comportera aucune variation et commencera toujours à partir de la même distance de l'observateur. Puisque notre modèle de précipitations cherche à inclure l'influence du vent, nous ne pouvons pas utiliser tel quel le brouillard proposé par Yoann. En effet nous pouvons tout à fait imaginer que le vent et surtout l'inclinaison des cordes par rapport à la caméra aient une importance non négligeable dans le rendu final de ce brouillard.

Puisque nous connaissons la taille minimale et maximale des gouttes rendues, nous sommes en mesure de calculer le seuil minimal et maximal de distance à l'observateur à partir duquel il n'est plus possible de distinguer individuellement ces gouttes. On découpe alors notre espace en trois zones. Dans la première zone, avant le seuil minimal, toutes les gouttes sont distinguables. Dans la seconde zone entre le seuil minimal et maximal, les gouttes les plus grosses peuvent encore être perçues individuellement là où les plus petites ne le sont plus. Au-delà du seuil maximal, aucune goutte ne peut encore être distinguée et cela quelle que soit leur taille. D'après ces observations et en admettant que des gouttes plus petites que la taille minimale que nous avons fixée ne peuvent exister, nous pouvons déduire que la brume devrait démarrer à partir de notre seuil minimal. On peut aussi déduire que le seuil maximal fixe la limite de distance à l'observateur au-delà de laquelle il est inutile de rendre des gouttes. Mais comme l'idée serait d'utiliser la brume pour réduire au maximum le nombre de gouttes à rendre, il serait encore mieux de fixer cette limite de création de goutte au niveau du seuil minimal et ainsi créer deux types de brume, où le premier niveau intégrerait dans son rendu les gouttes encore perceptibles de la seconde zone. De ce fait, la première couche de brume serait en charge de rendre la pluie dans la zone délimitée par le seuil minimal et maximal. Elle prendrait la forme d'un milieu participant dans lequel on pourrait percevoir des mouvements induits par le vent qui seraient modélisés par des différences de densité de la brume. A l'opposé, la seconde couche serait en charge de rendre une brume similaire à celle proposée par Yoann Weber, partant de notre seuil maximal et qui n'interagirait pas avec le vent.

Malheureusement, puisque nous n'avons pas choisi de nous concentrer sur cet aspect de la pluie, nous n'avons pu que théoriser cette méthode et rien de plus.

5 Conclusion et perspectives

Comme nous l'avons vu précédemment dans le rapport, la création d'un modèle météorologique complexe est un sujet très vaste qui englobe un grand nombre de phénomène et il est utopique de penser qu'en l'espace de cinq mois l'intégralité du sujet aurait pu être traité. C'est pour cela que l'objectif du stage consistait à ne simuler qu'une partie de ces phénomènes. Et même si au départ je ne connaissais pas grand chose sur ces évènements météorologiques, je me suis documenté pour au final produire deux modèles théoriques qui mixent les idées des différents papiers de l'état de l'art, permettant de modéliser la pluie. Durant ce stage j'ai aussi été amené à générer du code pour tester les limites de ces modèles et ainsi déceler leurs problèmes dans le but de pouvoir les améliorer.

Notre modèle reposant sur l'utilisation d'un système de particule peut être une bonne base vers la création d'un système météorologique complexe. Il est régi par des paramètres simples et offre un bon nombre de possibilités d'amélioration. Bien sûr il y a des améliorations plus urgentes que d'autres. En premier lieu il est impératif de réaliser un gros travail sur la manière d'illuminer les gouttes. Et en parallèle, l'ajout de la brume pourrait aussi apporter pas mal de réalisme. Dans un second temps l'ajout d'un cycle jour/nuit et la modélisation de la couleur du ciel ainsi que des nuages permettraient de donner à l'utilisateur la sensation d'évoluer dans un environnement plus familier et vivant. On pourrait imaginer ensuite que la pluie ne se mette qu'à tomber dans des zones situées en dessous des nuages, et que l'intensité de la pluie à ces endroits soit corrélée à la taille du nuages duquel elle provient. Il serait aussi appréciable que pour les plus imposants d'entre eux, des éclairs et autres arc-électriques puissent être générés. Et pour finir, l'incorporation de vent complexe au modèle permettrait d'insuffler encore plus de vie à la simulation en faisant bouger à la fois les objets de la scène mais aussi d'orienter le mouvement des gouttes et des nuages.

Références bibliographiques

- [BN08] Eric BRUNETON et Fabrice NEYRET. “Precomputed Atmospheric Scattering”. In : *Computer Graphics Forum* 27.4 (juin 2008), p. 1079-1086. DOI : [10.1111/j.1467-8659.2008.01245.x](https://doi.org/10.1111/j.1467-8659.2008.01245.x) (cf. p. 18, 19).
- [Cav19] Arthur CAVALIER. “Génération procédurale de textures pour enrichir les détails surfaciques”. Thèse de doctorat dirigée par Ghazanfarpour-Kholendjany, Djamchid et Gilet, Guillaume Informatique graphique Limoges 2019. Thèse de doct. 2019. URL : <http://www.theses.fr/2019LIM00108> (cf. p. 23, 24).
- [Cha+08] Wang CHANGBO et al. “Real-time modeling and rendering of raining scenes”. In : *The Visual Computer* 24.7-9 (juin 2008), p. 605-616. DOI : [10.1007/s00371-008-0241-0](https://doi.org/10.1007/s00371-008-0241-0) (cf. p. 12, 13).
- [CP13] Carles CREUS et Gustavo A. PATOW. “R4 : Realistic rain rendering in real-time”. In : *Computers & Graphics* 37.1-2 (fév. 2013), p. 33-40. DOI : [10.1016/j.cag.2012.12.002](https://doi.org/10.1016/j.cag.2012.12.002) (cf. p. 14).
- [Des04] Markus DESERNO. *How to generate equidistributed points on the surface of a sphere*. 2004. URL : https://www.cmu.edu/biolphys/deserno/pdf/sphere_equi.pdf (cf. p. 42).
- [GN06] Kshitiz GARG et Shree K. NAYAR. “Photorealistic rendering of rain streaks”. In : *ACM Transactions on Graphics* 25.3 (juill. 2006), p. 996-1002. DOI : [10.1145/1141911.1141985](https://doi.org/10.1145/1141911.1141985) (cf. p. 13, 40).
- [GN07] Kshitiz GARG et Shree K. NAYAR. “Vision and Rain”. In : *International Journal of Computer Vision* 75.1 (fév. 2007), p. 3-27. DOI : [10.1007/s11263-006-0028-6](https://doi.org/10.1007/s11263-006-0028-6) (cf. p. 13, 40).
- [Gil+14] Guillaume GILET et al. “Local random-phase noise for procedural texturing”. In : *ACM Transactions on Graphics* 33.6 (nov. 2014), p. 1-11. DOI : [10.1145/2661229.2661249](https://doi.org/10.1145/2661229.2661249) (cf. p. 23).
- [GM77] R. A. GINGOLD et J. J. MONAGHAN. “Smoothed particle hydrodynamics : theory and application to non-spherical stars”. In : *Monthly Notices of the Royal Astronomical Society* 181.3 (déc. 1977), p. 375-389. DOI : [10.1093/mnras/181.3.375](https://doi.org/10.1093/mnras/181.3.375) (cf. p. 15).
- [HLC19] Shirsendu HALDER, Jean-Francois LALONDE et Raoul De CHARETTE. “Physics-Based Rendering for Improving Robustness to Rain”. In : *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct. 2019. DOI : [10.1109/iccv.2019.01030](https://doi.org/10.1109/iccv.2019.01030) (cf. p. 14).
- [Hil20] Sébastien HILLAIRE. “A Scalable and Production Ready Sky and Atmosphere Rendering Technique”. In : *Computer Graphics Forum* 39.4 (juill. 2020), p. 13-22. DOI : [10.1111/cgf.14050](https://doi.org/10.1111/cgf.14050) (cf. p. 18, 19).
- [Hu+18] Yuanming HU et al. “A moving least squares material point method with displacement discontinuity and two-way rigid body coupling”. In : *ACM Transactions on Graphics* 37.4 (juill. 2018), p. 1-14. DOI : [10.1145/3197517.3201293](https://doi.org/10.1145/3197517.3201293) (cf. p. 15).
- [IS21] JinGi IM et Mankyu SUNG. “Efficient Rain Simulation based on Constrained View Frustum”. In : *International Journal of Advanced Computer Science and Applications* 12.5 (2021). DOI : [10.14569/ijacsia.2021.0120545](https://doi.org/10.14569/ijacsia.2021.0120545) (cf. p. 14).

- [INM19] Tomokazu ISHIKAWA, Ryota NAKAZATO et Ichiro MATSUDA. “Procedural Animation of Aurora and its Optimization for Keyframe Animation”. In : *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*. ACM, sept. 2019. DOI : [10.1145/3386164.3389098](https://doi.org/10.1145/3386164.3389098) (cf. p. 20).
- [Ish+11] Tomokazu ISHIKAWA et al. “Modeling of aurora borealis using the observed data”. In : (avr. 2011). DOI : [10.1145/2461217.2461220](https://doi.org/10.1145/2461217.2461220) (cf. p. 20).
- [JW97] D. JACKEL et B. WALTER. “Modeling and Rendering of the Atmosphere Using Mie-Scattering”. In : *Computer Graphics Forum* 16.4 (oct. 1997), p. 201-210. DOI : [10.1111/j.1467-8659.00180](https://doi.org/10.1111/j.1467-8659.00180) (cf. p. 18).
- [Jon+22] Theo JONCHIER et al. “Porous Flow and Sediment Transport Simulation for Physically-Based Weathering”. In : (2022). URL : <https://hal.science/hal-03696332/> (cf. p. 21).
- [Lag+10] A. LAGAE et al. *State of the Art in Procedural Noise Functions*. 2010. DOI : [10.2312/EGST.20101059](https://doi.org/10.2312/EGST.20101059) (cf. p. 24).
- [Lan+04] M. S. LANGER et al. *A spectral-particle hybrid method for rendering falling snow*. 2004. DOI : [10.2312/EGWR/EGSR04/217-226](https://doi.org/10.2312/EGWR/EGSR04/217-226) (cf. p. 13, 14).
- [LG11] Orion Sky LAWLOR et Jon D. GENETTI. “Interactive Volume Rendering Aurora on the GPU”. In : *J. WSCG* 19.1 (2011), p. 25-32. URL : http://wscg.zcu.cz/WSCG2011/_2011_J_WSCG_1-3.pdf (cf. p. 20).
- [MM13] Miles MACKLIN et Matthias MÜLLER. “Position based fluids”. In : *ACM Transactions on Graphics* 32.4 (juill. 2013), p. 1-12. DOI : [10.1145/2461912.2461984](https://doi.org/10.1145/2461912.2461984) (cf. p. 16).
- [MP48] J. S. MARSHALL et W. Mc K. PALMER. “THE DISTRIBUTION OF RAINDROPS WITH SIZE”. In : *Journal of Meteorology* 5.4 (août 1948), p. 165-166. DOI : [10.1175/1520-0469\(1948\)005<0165:tdorws>2.0.co;2](https://doi.org/10.1175/1520-0469(1948)005<0165:tdorws>2.0.co;2) (cf. p. 24, 35).
- [MDH07] Xing MEI, Philippe DECAUDIN et Bao-Gang HU. “Fast Hydraulic Erosion Simulation and Visualization on GPU”. In : (oct. 2007). DOI : [10.1109/pg.2007.15](https://doi.org/10.1109/pg.2007.15) (cf. p. 21).
- [Mül+07] Matthias MÜLLER et al. “Position based dynamics”. In : *Journal of Visual Communication and Image Representation* 18.2 (avr. 2007), p. 109-118. DOI : [10.1016/j.jvcir.2007.01.005](https://doi.org/10.1016/j.jvcir.2007.01.005) (cf. p. 16).
- [NPW84] L. NIEMEYER, L. PIETRONERO et H. J. WIESMANN. “Fractal Dimension of Dielectric Breakdown”. In : *Physical Review Letters* 52.12 (mars 1984), p. 1033-1036. DOI : [10.1103/physrevlett.52.1033](https://doi.org/10.1103/physrevlett.52.1033) (cf. p. 19).
- [Pey+09] A. PEYTAVIE et al. “Arches : a Framework for Modeling Complex Terrains”. In : *Computer Graphics Forum* 28.2 (avr. 2009), p. 457-467. DOI : [10.1111/j.1467-8659.2009.01385.x](https://doi.org/10.1111/j.1467-8659.2009.01385.x) (cf. p. 21).
- [PSS99] A. J. PREETHAM, Peter SHIRLEY et Brian SMITS. “A practical analytic model for daylight”. In : *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*. ACM Press, 1999. DOI : [10.1145/311535.311545](https://doi.org/10.1145/311535.311545) (cf. p. 18).

- [PRC09] Anna PUIG-CENTELLES, Oscar RIPOLLES et Miguel CHOVER. “Creation and control of rain in virtual environments”. In : *The Visual Computer* 25.11 (mai 2009), p. 1037-1052. DOI : [10.1007/s00371-009-0366-9](https://doi.org/10.1007/s00371-009-0366-9) (cf. p. 12).
- [RMP16] Eduard RANDO, Imanol MUÑOZ et Gustavo PATOW. *Interactive Low-Cost Wind Simulation For Cities*. 2016. DOI : [10.2312/UDMV.20161421](https://doi.org/10.2312/UDMV.20161421) (cf. p. 16).
- [RF22] Nuno REIS et Antonio Ramires FERNANDES. “Using a Space Colonization Algorithm for Lightning Simulation”. In : *2022 International Conference on Graphics and Interaction (ICGI)*. IEEE, nov. 2022. DOI : [10.1109/icgi57174.2022.9990427](https://doi.org/10.1109/icgi57174.2022.9990427) (cf. p. 19).
- [Ren19] Rupert RENARD. “Wind Simulation in God of War”. In : 2019. URL : <https://www.gdcvault.com/play/1026404/Wind-Simulation-in-God-of-War> (cf. p. 17).
- [Rou07] Pierre ROUSSEAU. “Simulation réaliste de pluie en temps-réel”. Thèse de doctorat dirigée par Ghazanfarpour-Kholendjany, Djamchid et Jolivet, Vincent Informatique Limoges 2007. Thèse de doct. 2007, 1 vol. (XVI-114 p.) URL : <http://www.theses.fr/2007LIM04019> (cf. p. 11, 29).
- [RJG06] Pierre ROUSSEAU, Vincent JOLIVET et Djamchid GHAZANFARPOUR. “Realistic real-time rain rendering”. In : *Computers & Graphics* 30.4 (août 2006), p. 507-518. DOI : [10.1016/j.cag.2006.03.013](https://doi.org/10.1016/j.cag.2006.03.013) (cf. p. 11, 40).
- [RJG08] Pierre ROUSSEAU, Vincent JOLIVET et Djamchid GHAZANFARPOUR. “GPU Rainfall”. In : *Journal of Graphics Tools* 13.4 (jan. 2008), p. 17-33. DOI : [10.1080/2151237x.2008.10129270](https://doi.org/10.1080/2151237x.2008.10129270) (cf. p. 11, 12, 14, 29).
- [Sto+13] Alexey STOMAKHIN et al. “A material point method for snow simulation”. In : *ACM Transactions on Graphics* 32.4 (juill. 2013), p. 1-10. DOI : [10.1145/2461912.2461948](https://doi.org/10.1145/2461912.2461948) (cf. p. 15).
- [SZS95] Deborah SULSKY, Shi-Jian ZHOU et Howard L. SCHREYER. “Application of a particle-in-cell method to solid mechanics”. In : *Computer Physics Communications* 87.1-2 (mai 1995), p. 236-252. DOI : [10.1016/0010-4655\(94\)00170-7](https://doi.org/10.1016/0010-4655(94)00170-7) (cf. p. 15).
- [Tar07] Sarah TARIQ. “Rain”. In : *Nvidia White Paper*. 2007. URL : <https://developer.download.nvidia.com/SDK/10/direct3d/Source/rain/doc/RainSDKWhitePaper.pdf> (cf. p. 14).
- [TI06] Natalya TATARUCHUK et John ISIDORO. *Artist-Directable Real-Time Rain Rendering in City Environments*. 2006. DOI : [10.2312/NPH/NPH06/061-073](https://doi.org/10.2312/NPH/NPH06/061-073) (cf. p. 13, 14, 19, 20).
- [Tre+20] Maxime TREMBLAY et al. “Rain Rendering for Evaluating and Improving Robustness to Bad Weather”. In : *International Journal of Computer Vision* 129.2 (sept. 2020), p. 341-360. DOI : [10.1007/s11263-020-01366-3](https://doi.org/10.1007/s11263-020-01366-3) (cf. p. 14).
- [Ulb83] Carlton W. ULBRICH. “Natural Variations in the Analytical Form of the Raindrop Size Distribution”. In : *Journal of Climate and Applied Meteorology* 22.10 (oct. 1983), p. 1764-1775. DOI : [10.1175/1520-0450\(1983\)022<1764:nvat>2.0.co;2](https://doi.org/10.1175/1520-0450(1983)022<1764:nvat>2.0.co;2) (cf. p. 36).

- [WW04] Niniane WANG et Bretton WADE. “Rendering falling rain and snow”. In : *ACM SIGGRAPH 2004 Sketches on - SIGGRAPH '04*. ACM Press, 2004. DOI : [10.1145/1186223.1186241](https://doi.org/10.1145/1186223.1186241) (cf. p. 12, 13, 25, 26).
- [Web19] Antoine WEBANCK. “Génération procédurale d’effets atmosphériques”. Thèse de doctorat dirigée par Galin, Eric et Guérin, Éric Informatique Lyon 2019. Thèse de doct. 2019. URL : <http://www.theses.fr/2019LYSE1109> (cf. p. 19, 20).
- [Web+16] Y. WEBER et al. “A phenomenological model for throughfall rendering in real-time”. In : *Computer Graphics Forum* 35.4 (juill. 2016), p. 13-23. DOI : [10.1111/cgf.12945](https://doi.org/10.1111/cgf.12945) (cf. p. 20, 23).
- [Web16] Yoann WEBER. “Rendu multi-échelle de pluie et interaction avec l’environnement naturel en temps réel”. Thèse de doctorat dirigée par Ghazanfarpour-Kholendjany, Djamchid et Jolivet, Vincent Informatique et applications Limoges 2016. Thèse de doct. 2016. URL : <http://www.theses.fr/2016LIM00109> (cf. p. 11, 22, 45).
- [Web+15] Yoann WEBER et al. “A multiscale model for rain rendering in real-time”. In : *Computers & Graphics* 50 (août 2015), p. 61-70. DOI : [10.1016/j.cag.2015.04.007](https://doi.org/10.1016/j.cag.2015.04.007) (cf. p. 13, 22, 40).
- [WH91] Jakub WEJCHERT et David HAUMANN. “Animation aerodynamics”. In : *Proceedings of the 18th annual conference on Computer graphics and interactive techniques - SIGGRAPH '91*. ACM Press, 1991. DOI : [10.1145/122718.122719](https://doi.org/10.1145/122718.122719) (cf. p. 16).
- [Yun+17] Jeongsu YUN et al. “Physically inspired, interactive lightning generation”. In : *Computer Animation and Virtual Worlds* 28.3-4 (mai 2017), e1760. DOI : [10.1002/cav.1760](https://doi.org/10.1002/cav.1760) (cf. p. 19).