

Rendu Volumétrique et Simulation de Nuage

Équipe : J.J. Parra Díaz, M. Longlade, N. Souvignet, X.-H. Wang

Liens du dépôt : <https://github.com/TER-VOX/cloud.git>

Encadrante : N. Faraj

Université de Montpellier
Faculté des sciences
Master Mention Informatique

Mai 2025

Résumé

Ce projet, réalisé dans le cadre de l'unité d'enseignement de Travaux d'études et recherche, consistait en la réalisation d'un moteur de rendu, permettant la visualisation de milieux participants tels que des nuages. Pour cela, nous avons exploré et mis en œuvre différentes méthodes de modélisations et de rendu adaptées à visualiser des objets volumétriques. Pour nous accompagner dans cette aventure, nous avons eu recours au langage C++, l'API Graphique OpenGL et par conséquent à son langage standard de shading GLSL. Le présent rapport fait état de notre avancée sur ce projet à l'issue de ce semestre. Il sert de conclusion aux mois de travaux écoulés mais également d'ouverture à d'éventuelles améliorations. Nous y ferons un tour d'horizon des méthodes implémentées et des concepts s'y afférants.

“Mais, dans ces conditions, s’ils pouvaient se parler les uns aux autres, ne penses-tu pas qu’ils croiraient nommer les objets réels eux-mêmes en nommant ce qu’ils voient ?”

– Platon, *Allégorie de la Caverne*

Avant de détailler plus avant ce rapport, nous tenions à adresser nos remerciements à Noura Faraj qui a accepté quasiment au pied levé de prendre de son temps pour nous encadrer et nous tutorer sur ce projet. Le nom Nurage est d’ailleurs un clin d’œil rempli de reconnaissance que nous lui adressons.

Nous espérons que la lecture de ce rapport vous sera agréable.

L’équipe Nurage.

Table des matières

1	Introduction	3
1.1	Objectifs du projet	3
1.2	Les nuages	3
1.3	Enjeux de la simulation et du rendu de nuages	4
1.4	Évolution des méthodes simulations et de rendu de nuages	5
1.5	Annonce du plan	5
2	Notions et concepts	6
2.1	Pipeline de rendu	6
2.1.1	Pipeline de rendu par rastérisation	6
2.1.2	Pipeline de rendu par lancer de rayon	9
2.2	Les modèles d'ombrages	12
2.3	Rendu Volumique	12
3	État de l'art	13
3.1	Un panorama sur le rendu de nuages.	13
4	Implémentations	15
4.1	Modélisation	15
4.1.1	Bruits et Textures 3D	15
4.1.2	Génération des formes	16
4.1.3	Système de Particules	19
4.2	Rendu	20
4.2.1	Imposteurs	20
4.2.2	Ray-Marching	22
4.2.3	Calculs de ray marching volumétrique.	23
5	Comparaison des approches	31
5.1	Ray-marching	31
5.2	Particules et imposteurs	31
5.3	Synthèse	32
6	Conclusion	33

Chapitre 1

Introduction

1.1 Objectifs du projet

Ce projet s'inscrit dans une démarche expérimentale d'exploration techniques autour du rendu volumétrique. Il a pour objectifs les points suivants :

- Comprendre les enjeux spécifiques au rendu volumétrique, notamment les questions liées à la transparence, à l'éclairage et à la simulation de densités dans un volume.
- Découvrir et expérimenter et mettre œuvre différentes approches et techniques de simulation et de rendu de nuages.
- Analyser et comparer ces approches selon plusieurs critères : qualité visuelle, performances, réalisme et facilité d'intégration dans un pipeline de rendu.

1.2 Les nuages

Les nuages, décrit physiquement, comme des masses visibles composées essentiellement d'un nombre important de particules d'eau en suspension, condensées sous forme liquide, forment un ensemble d'éléments essentielles de nos paysages. En effet, cette omniprésence nuageuse, couplée à leur variabilité de caractéristiques en terme notamment d'opacité, de forme et de couleur, fluctuant au gré de facteurs géographiques et ou climatiques tel que la pression atmosphérique, la température ou encore leurs altitudes en font une part inhérente de la façon que nous avons de percevoir une scène visuelle. En plus de leur impact manifeste sur la météo, qui justifie entre autre leur étude néphologique, la manipulation de ces entités brumeuses dans une image notamment permet également de transmettre des informations et même de véhiculer des émotions.

De ce fait, l'intérêt pour les entités volumétriques que sont les nuages n'est pas récent, les premières études et tentatives d'appréhension concrètent datant du XIX^e siècle. Cependant, exhausté par l'avènement de l'informatique ayant permis des avancées notamment en matière de simulation et de visualisation computationnelle de corps fluides leur étude a pris une autre dimension. Aujourd'hui, nous pouvons constater de nombreux domaines scientifiques et artistiques dans lesquels la pertinence de représenter de manière réaliste les comportements et spécificités visuels des nuages devient de plus en plus probante. C'est domaines varie de la prévision météorologique aux jeux vidéos en passant par le cinéma.

1.3 Enjeux de la simulation et du rendu de nuages

Les nuages étant des composés d'éléments décrivant des volumes complexes ayant des interactions riches et singulières avec leur environnement notamment avec la lumière, leur simulation présente déjà en elle-même de véritables enjeux. De plus, la simulation numérique cherchant à émuler des phénomènes réels et bien souvent continues au travers d'un prisme discret, se heurtant bien souvent à ce plafond de verre que symbolise la calculabilité, la complexité et les limitations matérielles, appelle à un certain nombre d'approximations susceptibles d'affecter la qualité effective de la simulation. Cette section cherchera à définir et à mettre en lumière les différents questionnements que soulève cet objectif.

La forme

De part leur composition détaillé et caractéristique, l'une des premières interrogation qu'amène la simulation de nuage est celle de génération de la forme. Nous entendons par là, la recherche de structures et ou d'algorithmes capables de saisir leur nature intrinsèquement volumique nous permettant ainsi à terme de nous approcher de façon signifiante de leur apparence. Pour ce faire, il est nécessaire, de prendre encrage dans le réel, mais ce faisant, nous heurtons à ce dernier, constatant qu'il existe de nombreux types de nuages ayant la possibilité d'arborer un nombre quasi-infini de formes et de motifs. Il nous faut donc prendre compte cette aléa avant d'envisager toute autres choses.

Le mouvement

Il reste également important, lorsque l'on s'attaque à ce sujet, de garder à l'esprit que les nuages, loin d'être des entités statiques, se mouvoient. Leur dynamique de mouvement répondant à celles des corps fluides. Leur simulation appelle donc à prendre en compte cette composante mouvement capable en autres de donner au ciel une certaine richesse visuelle.

Le visuel

Comme introduit précédemment, les volumes nuageux sont en réalité composés de particules d'eau. Particules, qui entretiennent notamment avec la lumière les traversant, des interactions complexes. La densité et la répartition de ces particules est responsable du visuel caractéristique des nuages. En effet, ces interaction, couplées à la nature semi-transparente de ces volumes participes à leur richesse visuel ainsi, qu'à la variabilité de leur palette de couleur.

En somme, l'équilibre entre ces différentes composantes, tenant compte des contraintes de performances sera gage de la qualité de rendu d'une simulation. Forts de ce constat, différentes méthodes ont vu le jour permettant d'obtenir des résultats satisfaisant à mesure des années.

1.4 Évolution des méthodes simulations et de rendu de nuages

Les premières tentatives de représentation numérique des nuages étaient assez simpliste et ne consistant qu'en un rendu d'images statiques. Un exemple ce genre de rendu se trouve dans le jeu vidéo Super Mario Bros datant de 1985 où le même sprite en 2D représentant des buissons était utilisé, avec une couleur différente pour simuler les nuages.

Plus tard, des méthodes de rendu plus poussées ont vu le jour cherchant à donner cette impression tridimensionnelle et volumétrique, comme le rendu en couches de sprites, qui par superposition d'image créait une illusion de profondeur rendant les nuages plus réalistes. Les imposteurs ou billboard ont également été introduits : des images de nuages, prises sous différents angles faisant toujours face au point de vu, créant ainsi non seulement une impression de 3D mais donnant également l'illusion de dynamisme.

Parmi les premières représentations volumétriques, on trouve également des nuages générés à partir de formes simples, telles que des sphères et des cubes. Le film Là-haut de Pixar en est un exemple [5]. Pour créer les nuages, ces formes simples ont été couplée à un processus de fragmentation, aboutissant à une distribution fractale de formes de différentes dimensions.

Par la suite, des méthodes de rendu et de génération de nuages plus sophistiquées on vu le jour. Parmi elles, des techniques tel que les systèmes de particules, le grilles d'occupations et le ray marching. Techniques, sur lesquels nous reviendrons dans la suite de ce rapport.

L'avènement de ces méthodes a permis d'une part la simulation de nuages dont le mouvement et la position sont influencés par des facteurs physiques réels, simulant ainsi le comportement de fluides atmosphériques. D'autre part, elles ont permis d'obtenir des résultats visuellement réalistes. Un exemple probant peut être vu dans le jeu vidéo Horizon : Zero Dawn [1], où, en combinant plusieurs techniques de générations, basées sur la notion de textures 3D et bruit ainsi que sur des méthodes d'éclairage appliquant notamment la loi physique de Beer-Lambert, des résultats réalistes ont été obtenus.

1.5 Annonce du plan

Dans le présent rapport après une présentation des notions fondamentales nécessaires à la compréhension des parties techniques, nous entamerons un état de l'art synthétique autour du rendu de nuages et plus largement de la simulation informatique puis, poursuivrons en faisant un tour d'horizon plus large autour de concepts fondateurs en infographie 3D. Nous consacrerons au cœur de ce rapport une section faisant état de nos implémentations des différentes approches explorées. Cette partie technique donnera lieu à une discussion autour de ces différentes méthodes, en termes notamment de performances et de réalisme. Nous conclurons en rappelant les enjeux de ce domaine d'études et exposerons les limitations auxquelles nous avons fait face.

Chapitre 2

Notions et concepts

Ce rapport s'attelant à développer un certains nombre de concepts de natures assez abstruses, ce chapitre vise à réduire si non à entièrement dissiper le voile abscons entourant ces notions en leurs apposant des explications claires. Explications qui aideront à la compréhension profonde de ce rapport.

2.1 Pipeline de rendu

Au coeur de l'infographie 3D, il y a la notion de pipeline de rendu. Un de pipeline de rendu, décrit un ensemble d'opérations permettant de visualiser à l'écran une scène 3D, définie par un ensemble de primitives discrètes. Le pipeline de rendu se décompose en une série d'étapes dont les données d'entrée de chacune se composent des données de sortie de la précédente.

2.1.1 Pipeline de rendu par rastérisation

Ce pipeline, représente une des méthodes de rendu les plus courantes si ce n'est la plus courante. L'objectif de cette section est de faire un tour d'horizon de ce pipeline et de mettre en lumière ses limitations quant à la visualisation de milieux participants que sont les nuages.

Définitions

Il paraît presque impossible de rentrer dans le vif du sujet concernant ce pipeline de rendu sans disposer au préalable des armes nécessaires pour l'appréhender. Par armes, nous entendons là, un ensemble minimal et fondamental de définitions et de concepts qui nous permettront de le saisir dans toute son essence. Le rôle de cette partie est donc, comme son nom l'indique, de définir.

- **Systèmes de coordonnées et espaces :** Un système de coordonnées est un cadre composé d'une origine et d'une base vectorielle formant un repère permettant de faire correspondre à tout point d'un espace un n-uplet de scalaire définissant ces coordonnées. Bien que ce rapport n'est pas la visée didactique d'un cours d'algèbre, il paraissait pertinent d'introduire cette notion car elle sert de pivot pour introduire un certain nombre «d'espaces 3D» notable dans ce pipeline :
 1. *Espace model* : Qui représente l'espace de conception du modèle dans lequel les points sont définis à partir du système de coordonnées de l'objet
 2. *Espace monde* : Espace dans lequel, les éléments de la scène, objets, lumière et caméra sont placé par rapport à l'origine du monde (0, 0, 0) et au repère cartésien.

3. *Espace de vue* : Espace dans lequel la géométrie est exprimée en fonction du point de vu et dans le repère de la caméra.
 4. *Espace de Clip* : Espace intermédiaire dans lequel les coordonnées sont exprimées après projection. Espace écran : Espace à 2 dimensions ayant pour origine le centre de l'écran faisant correspondre aux coordonnées 3D les coordonnées 2D de l'écran.
- **Maillage** : Un maillage ou mesh en anglais est une structure discrète permettant de décrire une forme en 3D. Il se compose, d'un ensemble de *Sommets* (*Vertices*), connectée par des *Arrêtes* (*Edges*), formant des *Faces* définissant la surface de l'objet décrit. Bien que les maillages puissent suivre différentes topologies et que leurs faces puissent prendre un grand nombre de formes, la forme la plus courante étant le triangle, par soucis de concision, nous ne considérerons que les maillages triangulaires dans la suite de ce rapport.
 - **Texture** : Une texture est essentiellement une image numérique appliquée sur la surface d'un modèle 3D (son maillage) permettant de donner des détails visuels, une couleur, un motif et des effets de surface qui ne seraient pas pratiques à créer avec la géométrie seule. Bien qu'il existe des textures qui ne soient pas de 2 dimensions, pour cette section, nous nous y limiterons et reviendrons par la suite sur des définitions plus approfondies notamment en matière de textures 3D. L'accès aux données d'une texture se fait par des coordonnées de texture comprises entre 0 et 1 sur chaque axe. OpenGL utilise ces coordonnées normalisées pour échantillonner la texture et retourne un texel — une valeur éventuellement interpolée à partir des texels voisins, selon la méthode de filtrage choisie.
 - **Caméra** : En infographie 3D, cherchant à se rapprocher des caractéristiques de son homologue réel la caméra définie, le point de vu, ce que l'on peut voir. Une caméra classique se compose de différents paramètres dont :
 - *Une position* : un point dans l'espace symbolisant l'endroit d'où l'on regarde.
 - *Un repère* : 3 vecteurs (Front, Right, Up), définissant l'orientation de la caméra.
 - *Un champs de vision (Field of Vision)* : Un angle en degrés décrivant la latitude de vision de la caméra.
 - *Des plans de coupe (Clipping planes)* : 2 plans décrivant les limites avant et arrière de l'espace de rendu. Ils définissent la profondeur de champs que la caméra est capable de voir.
 - **Scène** : Introduite en début de section, elle représente de manière structurée le point de convergence de tous les éléments graphiques, le cadre structurant l'ensemble des éléments que l'on veut rendre visuellement, des maillages à l'éclairage en passant par les textures à plaquer sur nos objets.
 - **Shaders** : Un shader est un petit programme informatique qui s'exécute directement sur la carte graphique (Graphics Processing Unit). Il prend une part inhérente dans le pipeline graphique, et dans la façon qu'un objet 3D a d'être rendu à l'écran.

Structures et étapes de rendu

Comme introduit précédemment, les étapes du pipeline sont exécutées de manière séquentielle, les données s'écoulant d'une étape à une autre. Tirant partie des qualités hautement parallèles du processeur graphique certaines de ces étapes s'exécutent de manière simultanée sur l'ensemble de leurs données d'entrée. Nous ferons alors ici un état des lieux ne se voulant pas exhaustif mais suffisant des étapes essentielles le composant. La figure 2.1 illustre ces étapes.

A. Manipulation Applicative Cette première étape, généralement exécutée sur le Central Processing Unit (CPU) est relative à la logique intrinsèque. Elle se charge de la gestion d'un certain nombre d'opérations afférentes à la scène dont notamment le placement des objets, de la lumière, la définitions de leurs propriétés, leurs animations, les calculs de physiques etc... Elle permet de définir ce que l'on doit voir, initialisant et influant sur les paramètres de la caméra et se charge également de transmettre à la carte graphique l'ensemble des données géométriques (Sommets, Normals, Triangles, Matrices de Transformations, Coordonnées de Textures, etc...).

B. Traitement de la géométrie Cette seconde étape massivement parallélisée par le GPU se décompose en plusieurs sous-étapes.

1. **Transformation des sommets :** A cette étape effectuée pour chaque sommet par le *Vertex Shader*, lors d'un rendu classique, transforme à l'aide de différentes matrices de transformations chaque sommet de son *espace model*, vers l'*espace monde* puis vers l'*espace caméra*, et enfin vers l'*espace de clip*. C'est également à cette étape, qu'un éclairage est généralement calculé aux sommets, que les couleurs de ceux-ci ainsi que le mappage des textures sont traitées puis passée à l'étape suivante.
2. **Assemblage des primitives :** Les sommets transformés, sont assemblés à partir de leurs informations de connexité en un ensemble de primitives. Comme spécifié plus tôt par soucis de concision, nous considérerons que ces primitives sont des triangles.
3. **Découpage et projection 2D :** A cette étape en réalité composée de plusieurs étapes que nous éviterons de détailler pour ne pas s'encombrer d'informations non nécessaire à la compréhension de la suite, les primitives résultantes des étapes précédentes qui sont partiellement ou entièrement en dehors du volume de vision de la caméra sont découpées. Seule la partie visible de chaque primitive est conservée. De plus, les coordonnées 3D des primitives dont les sommets ont été transformés par les étapes précédentes, et expirmées dans l'*espace de clip*, sont à nouveau modifiés en coordonnées 2D et normalisé dans l'*espace écran*. Une division perspective est couramment effectuée, ce qui entraîne que les primitives les plus éloignées apparaissent plus petites.

C. Rastérisation Cette étape fondamentale dans le pipeline de rendu, transforme les primitives en un ensemble de fragments. Pour chaque triangle, la rastérisation détermine quels pixels de l'écran sont couverts par celui-ci. Pour chaque pixel couvert, un "fragment" est généré, interpolant à celui-ci les attributs des sommets (couleur, normales, coordonnées de texture) transmises par les étapes précédentes. En définitive, la rastérisation est le processus qui convertit les formes vectorielles décrites en une image matricielle.

D. Shading des pixels A ce stade, pour chacun des fragments générés par la rastérisation un *Fragment Shader* est exécutée. Ce shader, tirant partie des attributs interpolés comme les coordonnées de textures et les normales calcule et détermine la couleur des pixels. Permettant la génération notamment d'effets d'ombrage donnant l'impression de 3D et la créations de détails visuels à la surfaces des objets.

E. Test de Profondeur et Fusion Ici, pour chacun des fragments, on effectue un test de profondeur, sa distance par rapport à la caméra est comparée à la valeur de profondeur contenu dans le *Z-Buffer*. Ce test, permet de déterminer quelles sont les fragments visibles dans la scène, ceux qui ne sont pas masqués par d'autres. De plus, de manière courante si pour un fragment rendu $\alpha < 1.0$, le symbole α représentant la composante de transparence dans l'espace couleur RGBA

la couleur de ce fragment "translucide" est interpolée avec celle du ou des fragments se trouvant à l'arrière.

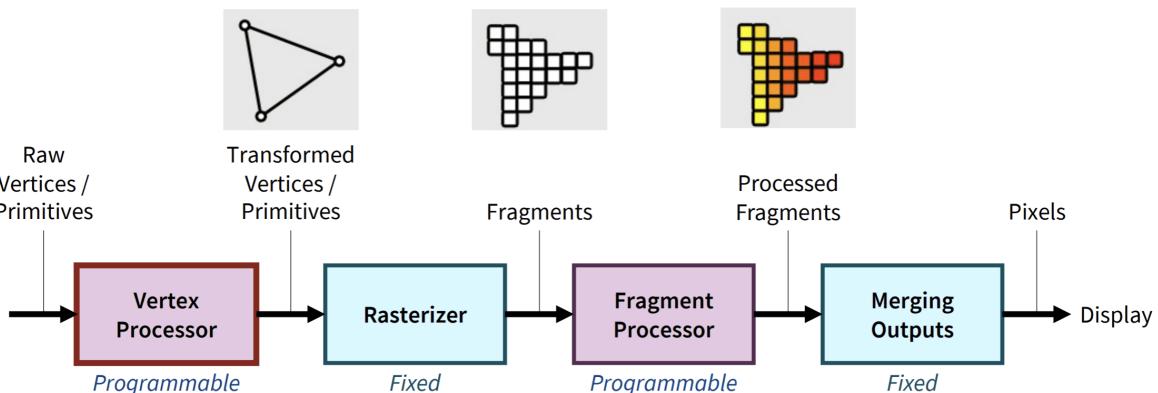


Figure 2.1 – Étapes du rendu de la géométrie par rastérisation.

2.1.2 Pipeline de rendu par lancer de rayon

Les premières mentions du lancer de rayon ou ray tracing en anglais sont datées des années 60. Conceptualisée par l'américain Arthur Appel, cette méthode de rendu graphique simule le parcours physique de la lumière pour créer des images photo-réalistes. Contrairement aux méthodes basées sur la rastérisation qui travaillent des modèles 3D vers l'écran, le ray tracing opère lui de l'écran vers la scène. Cette section permet d'exposer les étapes fondamentales de ce pipeline car le principe de ray tracing, sert d'introduction à un concept que nous avions chevillé au corps tout au long de ce projet : le ray marching.

Définitions

Certaines des définitions exposées plus tôt étant communes aux deux pipelines de rendu, il ne paraît pas pertinent d'y refaire mention ici. Cependant cette méthode apporte avec elle son lot de spécificités et de concepts que nous allons nous atteler à définir.

- **Rayon :** Dans le monde réel, un rayon de lumière matérialise le trajet suivi par la lumière pour aller d'un point à l'autre. Forts de cette analogie, les "rayons" en ray tracing sont en réalité des demi-droites possédant une origine et une direction, exprimés par l'équation paramétrique $P(t) = O + t \cdot D$ où :
 - $P(t)$ est un point sur le rayon,
 - O son origine,
 - D sa direction et
 - t un paramètre scalaire souvent interprété comme le temps ou la distance déterminant une position le long du rayon.
- **Intersection :** En lancer de rayon, le calcul d'une intersection désigne le processus fondamental qui consiste à déterminer si un rayon rencontre les objets de la scène, et le cas échéant, à calculer, les points d'entrée et de sortie à du rayon dans ces objets.
- **Réflexion et réfraction** Comme évoqué plus tôt ce rapport n'a pas la prétention de se substituer à un cours de sciences mais cherche à donner une définition claire de termes aidant

sa compréhension. La réflexion et la réfraction sont deux phénomènes fondamentaux qui dictent la manière qu'a lumière d'interagir avec certaines surfaces, permettant de simuler des effets visuelles réalistes. Une réflexion se produit lorsqu'un rayon lumineux frappe une surface et rebondit sur celle-ci. Tandis qu'une réfraction elle, se produit lorsque qu'un rayon de lumière est dévié lorsqu'il traverse un certains type de matériaux comme le verre ou l'eau.

Structures et étapes de rendu

Comme effectué pour le précédent pipeline nous allons ici également faire un état des lieux des étapes fondamentales entrant dans la composition celui-ci. Éludant au passage l'étape A. commune aux deux. La figure 2.2, illustre ces étapes.

B. Génération des rayons A cette étape, initialisant réellement le mécanisme de lancer de rayon, pour chaque pixels constituant la zone de vue au moins un rayon l'intersectant est tirer depuis la caméra. L'algorithme 1, expose cette procédure.

Algorithm 1 Génération d'un rayon

- 1: **Entrée :** Position p d'un point sur le pixels, Position c de la caméra
 - 2: **Sortie :** Rayon partant de la caméra et intersectant le pixel
 - 3: $rayOrigin \leftarrow c$
 - 4: $rayDirection \leftarrow \text{normaliser}(p - rayOrigin)$
 - 5: **Retourner** Rayon($rayOrigin, rayDirection$)
-

C. Calcul des intersections Ici, pour chaque rayon généré, l'on doit déterminer quels objets dans la scène sont intersectés par ceux-ci. Lorsqu'un rayon intercepte un objet, il faut déterminer le ou les points d'intersection. L'algorithme 2, expose cette procédure.

Algorithm 2 Calcul d'une intersection

- 1: **Entrée :** un objet M de la scene, un rayon R
 - 2: **Sortie :** Une intersection I
 - 3: $isIntersect \leftarrow M.\text{intersect}(R)$
 - 4: **if** $isIntersect$ **then**
 - 5: $enterPoint, exitPoint \leftarrow M.\text{getPoints}(R)$
 - 6: **end if**
 - 7: **Retourner** Intersection($isIntersect, enterPoint, ExitPoint$)
-

D. Génération des rayons secondaires Tirant avantage, des intersections calculé en amont, cette étape, se centre sur la génération de rayons secondaires au point d'intersection les plus proches d'un rayons s'il y en a un. Ces rayons responsables des calcules d'Ombrage, des effets de réflexion et de réfraction.

E. Composition des couleurs et débruitage Accumulation et Composition (Accumulation and Composition) : Les résultats de ces différents lancées de rayon (couleur, intensité lumineuse, reflet) sont à cette étape accumulés et combiné pour déterminer la couleur final de chaque pixels. L'image final résultante laissant bien souvent entrevoir l'apparition d'artefacts visuel du fait d'une

part d'imprécisions de calcul numériques et d'une autre d'optimisations de performances, des techniques de débruitage (denoising) sont souvent appliquée avec pour objectif de lisser ses imperfections. La figure 2.3, illustre ce phénomène.

Ces deux pipelines bien différents, ayant fait leurs preuves et étant même devenu, depuis plusieurs années maintenant des références, s'établissent avec une vision surfacique du rendu, vision qui pose quelque limitations quant à la visualisation de volumes ne possédant pas une surface propre au sens où on l'entend pour un objet solide.

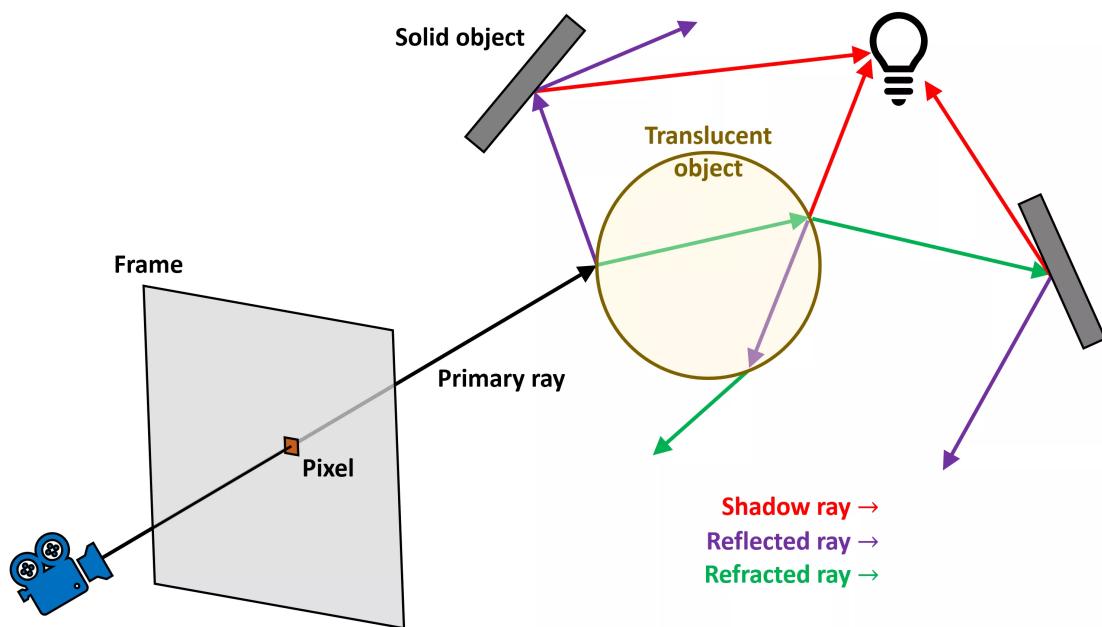


Figure 2.2 – Étapes du rendu de la géométrie par lancé de rayons

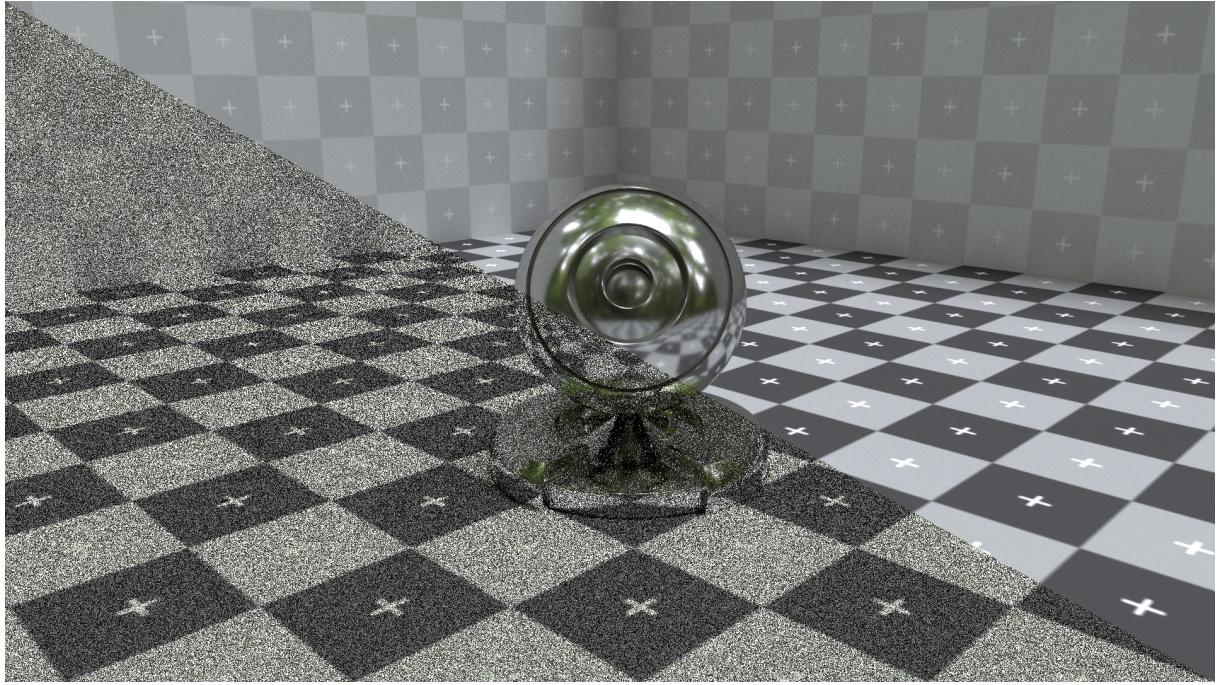


Figure 2.3 – Étapes du rendu de la géométrie par lancé de rayons

2.2 Les modèles d'ombrages

Cette courte section vient éclairer sans entrer dans trop de détails superflues une notion que nous n'avons fait qu'effleurer au cours des précédentes sections : Les modèles d'éclairage. En infographie 3D, est défini comme un modèle d'éclairage, un ensemble d'algorithmes et de formules qui régissent la façon dont la lumière interagit avec des objets d'une scène virtuelle. Leur but est de simuler de manière réaliste ou stylisée la façon dont la lumière est réfléchie, absorbée et diffusée par les matériaux, afin de donner aux objets rendu leur apparence visuelle (couleur, brillance, ombres, etc...) participant entre autre à l'impression tridimensionnelle. Ils oscillent de modèles simplifiés comme celui de Lambert à des modèles exhaustifs la physique (Physique Based Rendering) que nous détaillerons plus tard.

2.3 Rendu Volumique

Le rendu volumique par opposition au rendu surfacique regroupe l'ensemble des méthodes permettant de visualiser directement un ensemble de données 3D sans avoir à en extraire explicitement une surface. Il trouve son intérêt dans la visualisation d'entité volumétrique comme les nuage le plaçant au coeur de ce rapport mais aussi dans le médical notamment dans lequel il permet entre autre de restituer visuellement des données 3D issues de système d'acquisition comme les CT-Scan ou les IRM. Il introduit avec lui la notion de voxels remplissant en dimension 3 le rôle des pixels en dimension 2.

Chapitre 3

État de l'art

Nous voilà maintenant outillé pour appréhender les concepts inhérents au rendu de nuages. Dans ce chapitre nous ferons un pas dans la compréhension du contexte qui entoure la simulation volumétrique de nuages et ferons comme l'indique son intitulé état de l'art de ce domaines d'étude.

3.1 Un panorama sur le rendu de nuages.

Pour rendre des nuages, il existe donc plusieurs solutions adaptées au besoins et à l'objectif en terme de mouvement, de réalisme visuelle, et de performances :

- **La SkyBox :** représente la solution la plus simple mais aussi la plus limitée. Il s'agit d'une projection de l'environnement céleste sur un cube centré sur la caméra, utilisée en arrière-plan lorsque aucun objet n'est visible. Cette méthode convient à des scènes statiques, mais devient problématique si l'on souhaite animer les nuages. Faire défiler les textures des faces briserait la cohérence visuelle, car cela ne simulerait pas un déplacement réel dans un espace tridimensionnel.
- **Les Imposteurs :** consiste à placer des images (sprites ou textures) orientées dans l'espace pour simuler des volumes. On peut utiliser des formes irrégulières ou déformées pour évoquer des nuages, parfois animées autour d'un centre de scène. L'ajout d'un effet parallaxe améliore la profondeur perçue, mais l'ensemble donne souvent une impression de palteur. Cette méthode permet toutefois d'afficher efficacement certains effets comme de la fumée avec un temps de calcul raisonnable.
- **Les Maillages 3D :** il est effectivement possible pour représenter des nuages de les modéliser avec le concourt d'un maillage et de les afficher après certains traitements pour leur donner un coté "nuageux" au maillage. Cette approche à pour avantage d'offrir un certain contrôle aux artistes sur la forme et la couleur du nuage.
- **Rendu par tranches :** le volume du nuage est découpé en plusieurs plans parallèles à la caméra, chacun avec une transparence partielle. En les superposant, on obtient une illusion de volume. Cependant, si l'angle de vue devient trop oblique, les tranches peuvent devenir perceptibles, ce qui dégrade le réalisme.
- **Ray marching :** cette méthode repose sur le lancer rayon depuis la caméra. Avec la spécificité de marcher à intervalles réguliers de le volume nuages le long de ce rayon, échantillonnant au passage des informations telles que la densité ou l'éclairage du volume. Cela confère au rendu un aspect très réaliste, avec des effets de lumière volumétrique et une intégration naturelle des nuages dans la scène, quelle que soit la position de la caméra.

Les maillages et les imposteurs offrent tout un contrôle plus direct sur la forme des nuages, avec une complexité réduite. Les imposteurs, sont notamment, bien adaptés à des systèmes nécessitant des optimisation accru en termes de performances. Ces méthodes, ne sont cependant pas figées, il est tout à fait possible de les combiner, tirant partie du meilleur de chacune d'elles. Par exemple en utilisant des imposteurs pour les nuages lointains et du ray marching pour ceux proches de la caméra.

Chapitre 4

Implémentations

4.1 Modélisation

4.1.1 Bruits et Textures 3D

Un nuage étant un volume à représenter, une approche de modélisation implémentées consiste à interpoler la densité de celui-ci grâce à un composé de plusieurs bruit, utilisés soit pour remplir une grille 3D de voxels envoyée au GPU sous forme de texture 3D, soit en le produisant directement depuis le GPU.

Bruits et approximation du nuage

Avant de parler plus en détails dans cette approche, il semble important préciser ce qu'est un bruit et d'en connaître le fonctionnement. Bien souvent utilisé dans les rendus numériques les bruits constituent une ensemble d'outils fondamentales utilisés pour introduire de la variation et du réalisme dans les images synthétiques. Ils permettent d'ajouter des variations pseudo-aléatoires cohérentes, dans un tableau simple, tel qu'une texture 2D ou 3D. Dans le cadre de la visualisation de milieux participants tel que les nuages ils permettent de simulé la forme et les détails visuels présent dans les nuages. Les bruits pertinent dans ce cadre étant l' bruit de Perlin, le Mouvement Fractal Brownien, le bruit de Worley et le bruit de Perlin-Worley.

Le Bruit de Perlin développé par Ken Perlin en 1985 dans le cadre de son travail sur le film Tron. Ce bruit se basant sur un ensemble de vecteurs directionnel aléatoire, nommés vecteur de gradient apposés aux coins des cellules d'une grille permettent en n'importe quel point présent dans celle-ci, de calculer une valeur en interpolant les contributions des vecteurs de gradient des coins de la cellule dans laquelle se trouve le point. Cette interpolation confère à ce bruit son aspect fluide et organique caractéristique.

Le Mouvement Fractal Brownien n'est pas une fonction de bruit en lui même, mais en réalité une technique intrinsèquement liées à la notion de fractale qui combinée au bruit de Perlin permet de créer des textures beaucoup plus riches et détaillées.

Bruit de Worley Le bruit de Worley, aussi appelé bruit de Voronoï, est un autre type de bruit procédurale très utilisé en infographie, il permet la génération de motifs assez différents du bruit de Perlin, particulièrement adapté pour simuler des structures cellulaires ou granuleuses lui valant l'appellation par certains de bruit cellulaire.

Le bruit de Perlin-Worley Résultant de la composition linéaire de bruit de Perlin et de Worley, il tire partie forces des deux bruits et est particulièrement adapté pour la génération détails caractéristiques de nuages.

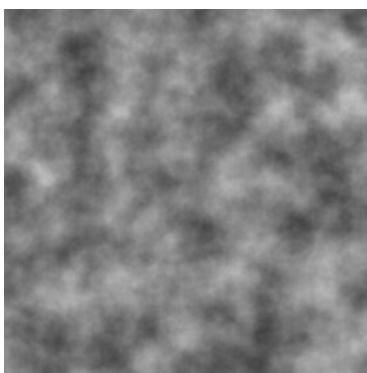


Figure 4.1 – Bruit de Perlin

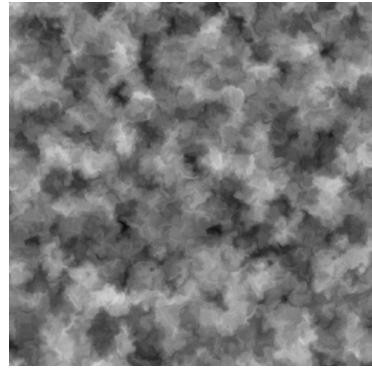


Figure 4.2 – Bruit de Perlin
Fbm

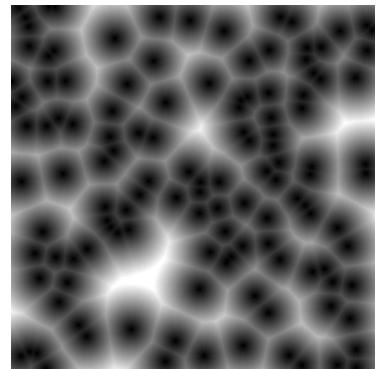


Figure 4.3 – Bruit de Worley

4.1.2 Génération des formes

Formes

Dans notre pipeline de génération de forme, l'idée était de « fragmenter » progressivement un volume pouvant s'apparenter entre autre à une sphère pour obtenir un volume nuageux à l'aspect naturel, à la fois aléatoire et cohérent. Le processus se décompose en quatre étapes successives.

- Premièrement, il faut déterminer si le rayon issu de la caméra pénètre dans le volume. Nous réalisons un test d'intersection rayon–sphère sur le centre `centre` et le rayon `sphere_radius`, ce qui nous donne les paramètres du temps d'entrée et de sortie t_0 et t_1 . En l'absence d'intersection, on renvoie la couleur de fond; sinon, on répartit uniformément le pas `step_size` sur l'intervalle $[t_0, t_1]$, définissant ainsi la zone où le nuage sera échantillonné. Cette première étape fixe la base géométrique de l'algorithme.

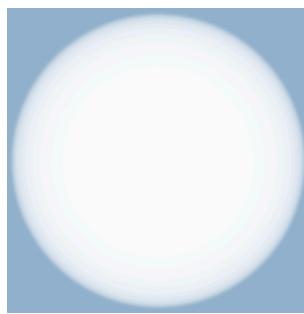


Figure 4.4 – Le rayon intersecte la sphère, ce qui donne une sphère de densité uniforme.

- La deuxième étape consiste à injecter un bruit 3D à l'intérieur de la sphère. On utilise un bruit de Perlin (figure 4.5) enrichi par une *fractal Brownian motion - fBm* (figure 4.6) : chaque octave multiplie la fréquence par 2 et réduit l'amplitude de moitié, de sorte que les basses fréquences créent de larges zones d'ombre et les hautes fréquences ajoutent des détails fins. Le résultat est un champ aléatoire à la fois « grumeleux » et continu. Pour chaque point

d'échantillonnage, on calcule une densité de base *base_density* en fonction de sa distance au centre, puis on multiplie par $(0.8 \times noise + DensityOffset)$, où *noise* est le résultat de la fBm. À ce stade, la sphère conserve son contour général, mais sa surface présente déjà les grappes caractéristiques des nuages.



Figure 4.5 – La forme brute avec la texture de bruit sans fbm

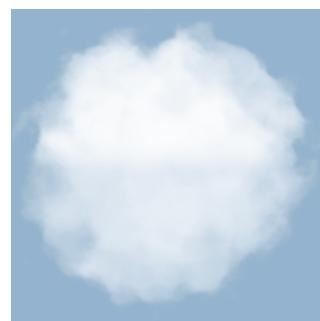


Figure 4.6 – La forme avec l'utilisation fbm

- La troisième étape applique un déplacement « warp » au nuage : Dans un shader, le warp consiste à déformer les coordonnées de l'espace : on ajoute au point d'échantillonnage initial un petit vecteur de déplacement généré par une fonction de bruit, puis on rééchantillonne à cette nouvelle position[4]. Par exemple, appliqué à une texture de bruit 2D, cela produit l'effet illustré en passant de la Figure 4.7 à la Figure 4.8.



Figure 4.7 – Une texture de bruit

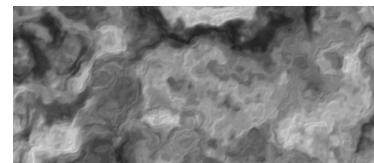


Figure 4.8 – La texture de bruit après warpped

À chaque pas de ray-marching, on génère un petit vecteur de décalage issu d'un bruit supplémentaire et on rééchantillonne la densité à la nouvelle position. Ce procédé plisse la structure, brisant les couches concentriques et créant des arêtes déchiquetées. Le paramètre *warp_scale* module l'amplitude de la déformation, et *warp_offset* en ajuste la fréquence : en augmentant *warp_scale*, on obtient un nuage plus lâche(figure 4.9 à figure 4.10), tandis qu'un *warp_offset* élevé la discrétisation (figure 4.10 à figure 4.11).

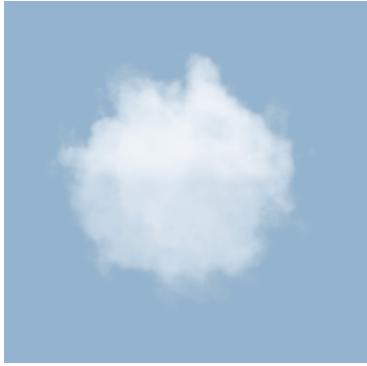


Figure 4.9 – Forme obtenue après application du warp : warp scale = 0.7 , warp offset = 0.5



Figure 4.10 – Forme obtenue après application du warp : warp scale = 1 , warp offset = 0.5

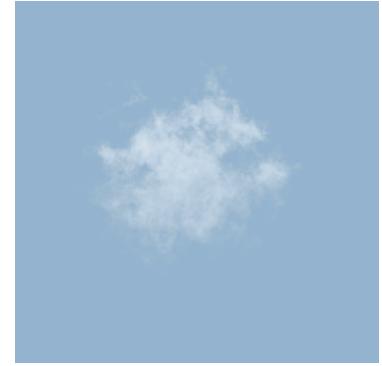


Figure 4.11 – Forme obtenue après application du warp : warp scale = 1 , warp offset = 1.5

- Enfin, on affine la silhouette avec une transformation linéaire $R(v, l_0, h_0, l_n, h_n)$ qui remappe une valeur $v \in [l_0, h_0]$ sur $[l_n, h_n]$. En particulier, on calcule

$$R(\text{height}, 0.5, 1.0, 0.0, 1.0)$$

où height est le pourcentage de hauteur du point ($0 \leq \text{height} \leq 1$). Cette fonction envoie la partie inférieure $[0, 0.5]$ sur 0 (suppression de la moitié basse) et la partie supérieure $[0.5, 1.0]$ sur $[0, 1]$, donnant l'illusion que l'on a taillé la sphère horizontalement. C'est un moyen simple de gérer l'apparition ou la disparition de nuages selon l'altitude.

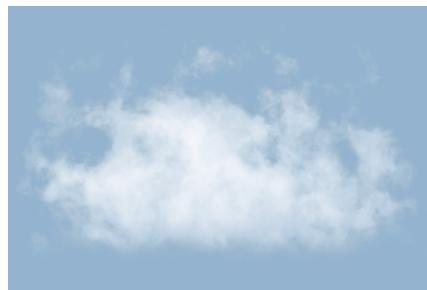


Figure 4.12 – Forme avec pipeline fmb, warp et R.

Grâce à ces quatre phases — intersection géométrique, bruit fractal, déformation par warp et remappage linéaire — une sphère parfaite se transforme en un nuage volumétrique à la fois complexe, éparsé sur les bords et doté d'un dégradé naturel de densité.

Texture 3D

Pour générer la texture 3D qui représente le volume qui sera ensuite rendu sous forme de nuage, nous avons utilisé l'outil FastNoiseLite [2], qui nous permet de créer différents types de bruit et d'ajuster divers sous-paramètres, offrant ainsi un contrôle précis sur le résultat final. Cet outil a permis de générer un bruit tridimensionnel.

Après avoir analysé les différentes options de bruit, ainsi que les méthodes et exemples existants dans la littérature sur la génération de nuages, il a été conclu que le type de bruit le plus approprié à cette fin était le bruit de Perlin.

Ce type de bruit permet de former des grappes en attribuant des valeurs à chaque coordonnée en fonction de leur proximité avec certains points clés, ce qui produit un effet de grappe qui ressemble à la texture d'un nuage. Pour ajouter davantage de réalisme et de dynamisme, une technique de mouvement brownien fractionnaire (fBm) a également été appliquée, qui introduit des variations douces et cohérentes dans le bruit.

Une fois le bruit généré, la texture 3D a été construite et paramétrée dans le programme, ce qui permet de contrôler sa fréquence. Cela permet d'ajuster visuellement l'aspect clair ou nuageux du ciel.

Chaque fois que l'un de ces paramètres est modifié, la texture est régénérée et renvoyée au GPU pour traitement.

En outre, un paramètre de vitesse a été ajouté, qui déplace les coordonnées de génération de la texture dans le temps, simulant ainsi le mouvement des nuages.

Dans une première version, la texture était envoyée au GPU à chaque image, ce qui ralentissait la simulation. Pour optimiser les performances, le déplacement a été déplacé vers le shader, générant et chargeant une nouvelle texture uniquement lorsqu'un paramètre affectant directement la génération de bruit est modifié.

Pour donner une impression de mouvement au volume, il suffit de rajouter dans une direction, un offset de plus en plus grand dans les coordonnées de textures échantillonées. Ce qui a pour conséquence de faire défiler la texture. On peut ajouter l'offset seulement lors du calcul du bruit plus fin.

4.1.3 Système de Particules

Une autres approches, nommées système de particules, consistent en la génération aléatoire dans un volume d'un ensemble ponctuels nommés particules apportant avec elles des informations de positions, de densité de mouvement et de tailles. Les particules sont souvent traitées comme des éléments individuels bien influencées dans leurs mouvements par leur voisinage les rendant pertinentes quant la simulation de mouvements physiques. Ces particules en elles même sont rarement défini comme des objets "concrets" en 3D au sens géométrique, mais plutôt comme des éléments visuels qui, une fois nombreux et animés, créent des effets complexes. Ce système généralement géré par le CPU à su tirer partie du parallélisme des GPU grâce à l'utilisation de shader de calcul (compute shader).

Utilisation des compute shaders pour la simulation de particules

La génération d'une grille de densité volumique à partir d'un ensemble de particules nécessaire, à chaque frame (affichage de la scène), de connaître leur position et leur contribution locale à l'opacité du nuage. Réaliser cette opération côté CPU puis transférer les données vers le GPU représente un coût important en termes de bande passante et de synchronisation.

Pour optimiser ce processus, nous avons migré l'intégralité de la simulation des particules sur le GPU, en exploitant les *compute shaders*. Ces shaders, exécutés en dehors du pipeline graphique classique, permettent de réaliser des calculs généralistes hautement parallélisables.

Notre implémentation repose sur deux passes distinctes :

- Une première passe met à jour les particules : position, vitesse, durée de vie, etc.
- Une seconde passe reconstruit la grille d'opacité tridimensionnelle à partir des nouvelles positions des particules.

La séparation en deux passes est indispensable car la mise à jour de la grille dépend de la totalité des positions mises à jour lors de la première phase. Toute tentative de lecture/écriture



Figure 4.13 – Échec de l’accumulation de transparence à cause du test de profondeur.

concurrente à l’intérieur d’une seule passe introduirait des conflits ou des artefacts visuels, en particulier sur les zones de densité élevée.

L’utilisation des compute shaders permet ainsi d’éviter toute copie CPU-GPU entre les étapes et garantit une exécution entièrement parallèle et localisée sur la mémoire du GPU.

4.2 Rendu

4.2.1 Imposteurs

Une autre approche avec laquelle nous avons voulu expérimenter concernant l’utilisation de particules reposait sur l’utilisation d’imposteurs pour visualiser les particules. Cette approche constitue notre première tentative pour visualiser ces particules.

Principe général

Un imposteur est une géométrie simple (souvent un triangle ou un quad) sur laquelle une texture est affichée pour simuler un objet plus complexe. Cette approche permet de représenter visuellement des entités riches tout en conservant un coût de rendu faible, ce qui la rend particulièrement adaptée pour le rendu de particules en grand nombre.

Il suffit de connaître la position 3D de chaque particule ainsi que les coordonnées 2D de la texture à appliquer. Le fragment shader peut alors afficher cette texture avec transparence sur chaque quad.

Afin d’optimiser les échanges entre le CPU et le GPU, nous n’envoyons au GPU que la position des particules, en laissant celui-ci générer la géométrie des quads. Pour cela, nous utilisons un geometry shader, qui intervient avant le vertex shader dans la chaîne de traitement graphique. Ce shader reçoit un point par particule et génère un quad centré sur ce point, en construisant les quatre sommets ainsi que leurs coordonnées de texture associées. Le vertex shader positionne ensuite les sommets dans l’espace, et le fragment shader applique la texture.

Gestion de la transparence

Le rendu de la transparence avec des imposteurs nécessite une attention particulière. Par défaut, le pipeline graphique applique un test de profondeur (Z-test) qui empêche le rendu cor-

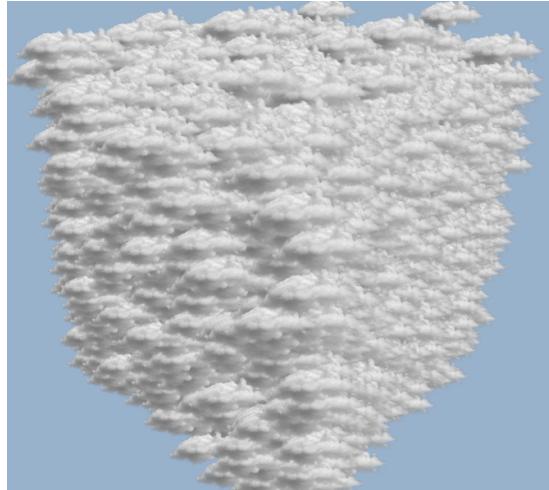


Figure 4.14 – Empilement correct des imposteurs grâce au tri par profondeur.

rect des éléments transparents. Si une particule est située devant une autre, et qu'elle est affichée avant, elle risque d'empêcher la particule de derrière d'être affichée qui sera masquée par le test de profondeur. Empêchant ainsi toute accumulation visuelle de transparence comme l'illustre la figure 4.13.

Pour corriger cela, il est nécessaire de désactiver l'écriture dans le depth buffer pendant le rendu des imposteurs, et surtout de trier les particules par distance à la caméra (du plus éloigné au plus proche) avant de les envoyer au GPU. Ce tri garantit que les fragments les plus lointains sont dessinés en premier, assurant une composition correcte des transparences. Le résultat du tri est visible sur la figure 4.14.



Figure 4.15 – Texture de particule générée par ray-marching.

Génération de textures dynamiques

Pour améliorer l'apparence des nuages, nous avons choisi de générer dynamiquement les textures utilisées par les imposteurs en réutilisant nos techniques de rendu volumique. Cela permet un contrôle visuel plus précis sur la forme et la densité de chaque particule (figure 4.15).

Le problème de cette approche concerne le traitement des lumières, les particules ont une ombre sur le coté gauche et elles sont exposées à la lumière sur le coté droit, et ce, même si il y a d'autres particules sur leur droite. Mettre à jour dynamiquement les textures permet d'obtenir un éclairage cohérent avec la direction de la lumière, mais ne résout pas le manque d'interactions entre particules (occlusion mutuelle, auto-ombres, etc.). Le résultat final sur les imposteurs illustré avec la figure 4.16.

4.2.2 Ray-Marching

Pour chaque pixel de l'image finale, il nous faut définir un rayon à lancer. Un rayon est défini par deux éléments :

- un point d'origine.
- une direction, qui dépend du pixel à l'écran.

Pour implémenter du ray-marching, nous avons testé 2 approches différentes.

Ray-Marching 1 : rastérisation d'un cube

Cette méthode consiste à rastériser un cube englobant le volume, en attribuant à chaque sommet à la fois sa position dans l'espace et des coordonnées de texture 3D (variant de 0 à 1 sur les axes x, y et z). 4.17 schématisé l'interpolation de coordonnées de texture lors du rendu d'un pixel d'une face. La figure 4.18 permet de visualiser les coordonnées de textures sur les faces du cube. Grâce à l'interpolation effectuée durant la rastérisation, on peut reconstruire pour chaque pixel :

- les coordonnées d'entrée dans le volume (issues des coordonnées de texture).
- la position correspondante dans l'espace monde.

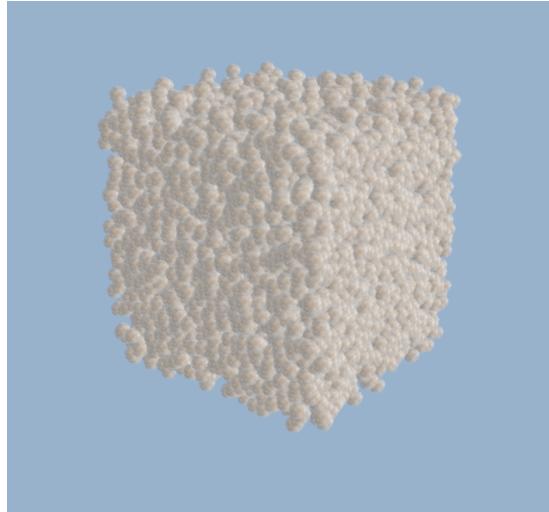


Figure 4.16 – Bloc de particule final composé d’imposteurs utilisant des textures générées par ray-marching.

La direction du rayon est obtenue en soustrayant la position interpolée du pixel à la position de la caméra, puis en normalisant ce vecteur. On peut alors effectuer le ray marching : avancer pas à pas depuis le point d’entrée, dans la direction du rayon, en échantillonnant la densité du volume.

Cette première approche a deux défauts :

- Lorsque le rayon est presque parallèle à une face du cube, l’interpolation devient imprécise, provoquant une distorsion de la direction du rayon. Cela entraîne des artefacts visuels, notamment un aplatissement des volumes.
- Cette méthode ne permet pas de visualiser l’intérieur du volume : les rayons ne sont générés qu’à partir des faces du cube vers l’extérieur. Si la caméra se trouve à l’intérieur du volume, aucun rayon n’est correctement généré.

Ray-Marching 2 : utilisation de la transformation inverse de la projection

Contrairement à la première méthode, ici on ne rastérise pas un cube, mais un simple rectangle couvrant l’écran. Cela permet de générer un rayon par pixel sans passer par l’interpolation de coordonnées 3D sur un volume.

Chaque sommet de ce rectangle correspond à un coin de l’écran. Lors du rendu, on reconstruit pour chaque pixel sa position dans l’espace caméra (en appliquant l’inverse de la projection). Cela permet de déterminer directement le point d’origine du rayon (la caméra) ainsi que la direction du rayon (calculée à partir de la position du pixel dans l’espace caméra).

Cette variante résoud ainsi les défauts de distorsions en plus de permettre le rendu du volume depuis l’intérieur de celui-ci.

4.2.3 Calculs de ray marching volumétrique.

Diffusion entrante

L’éclairage joue un rôle essentiel dans le rendu des nuages : il donne du aspect volumique et permet de moduler leur teinte, en les rendant plus sombres ou plus lumineuses à volonté.

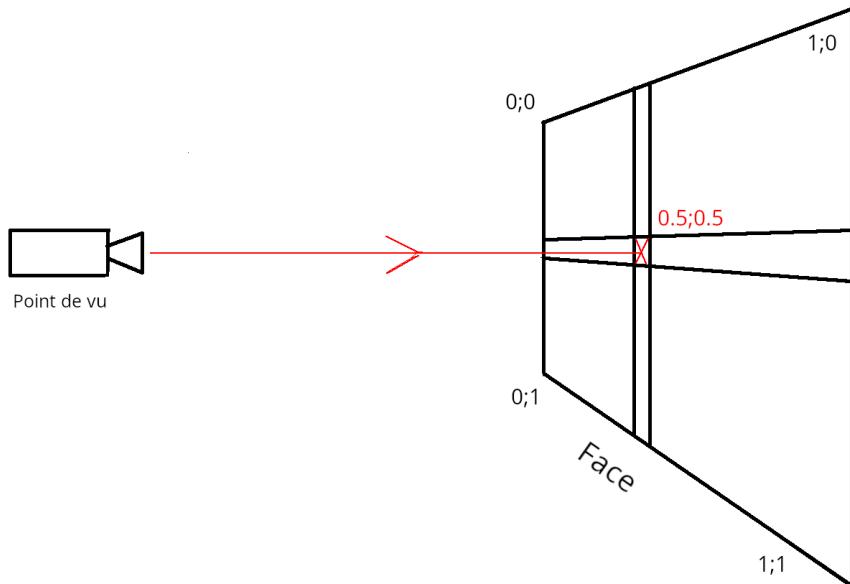


Figure 4.17 – Schématisation de l’interpolation des coordonnées de texture sur une face. La case englobant une croix rouge représente un pixel rendu avec les coordonnées associées (en rouge) qui sont issues des valeurs des sommets constituant la face.

Dans cette même optique, l’article [3] introduit les notions d’*in-scattering* et d’*out-scattering* ainsi que leurs formulations analytiques. Diffusion entrante(*in-scattering*) désigne le phénomène par lequel, lorsqu’un rayon lumineux traverse un nuage, une partie de la lumière provenant d’autres directions (par exemple du Soleil) est diffusée vers la ligne de vue de l’observateur. Il « éclaire » ainsi le volume traversé par le rayon, donnant aux nuages un effet de bord argenté ou de halo.

Application du modèle d’éclairage dans Ray-Marching Pour rappel, le processus de ray-marching additionne la contribution lumineuse de chaque pas successif; la somme courante est stockée dans la variable `accumulated_color`. Notre modèle d’éclairage s’insère naturellement dans ce mécanisme : à chaque échantillon, la quantité de lumière est calculée puis ajoutée à `accumulated_color`, de sorte que le résultat final réunit l’effet de tous les pas le long du rayon.

Extinction totale Lorsque l’on simule la propagation de la lumière à l’intérieur d’un nuage, deux mécanismes responsables de la perte d’énergie lumineuse doivent être pris en compte :

- Absorption : Les photons sont absorbés par les gouttelettes d’eau ou les cristaux de glace du nuage et leur énergie est convertie principalement en chaleur. La probabilité, par unité de longueur, qu’un tel événement se produise est notée σ_a .
- Diffusion : Les collisions entre photons et particules dévient les photons de leur direction initiale, ce qui réduit le flux lumineux le long du faisceau. La probabilité linéaire associée est notée σ_s .

En additionnant ces deux contributions, on obtient le coefficient d’extinction total :

$$\sigma_t = \sigma_a + \sigma_s$$

qui quantifie, par unité de longueur, la probabilité que la lumière quitte le faisceau initial — que ce soit par absorption ou par diffusion.

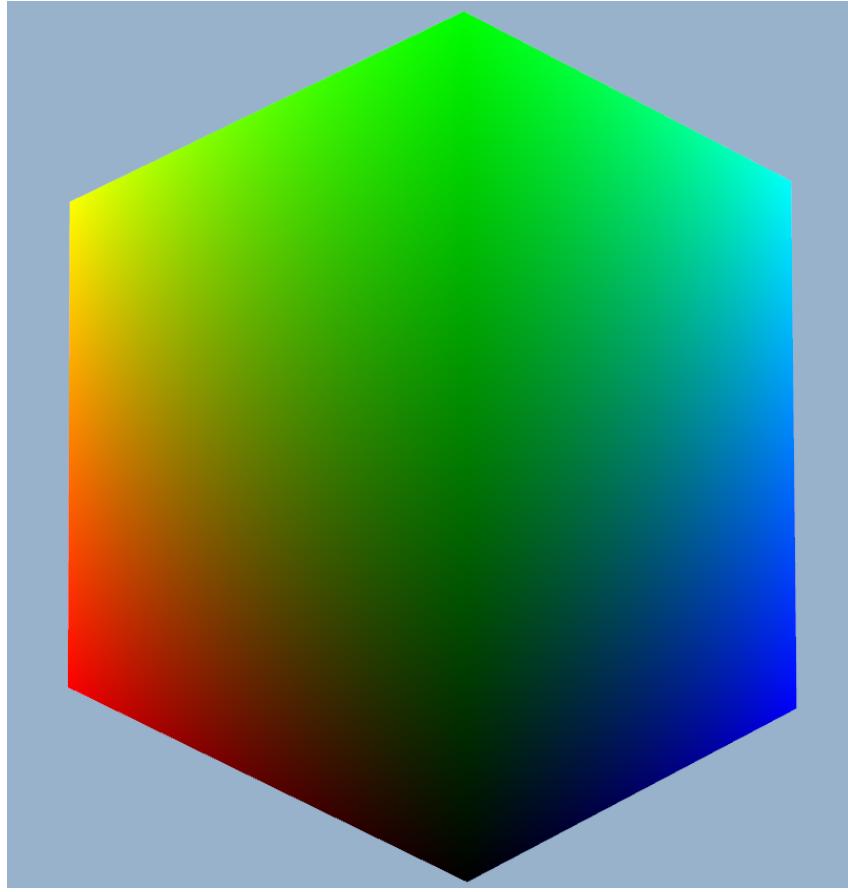


Figure 4.18 – Visualisation des coordonnées sur le cube. Le sommet supérieur au centre à une valeur rattachée de $(0,1,0)$ tandis que le point inférieur droit a une valeur $(0,0,1)$. Les pixels allant d'un point à l'autre varient donc du vert au bleu.

Atténuation depuis la source Ce terme simule l'énergie perdu par le rayon pendant son trajet *soleil → point d'échantillonnage dans nuage*. Plus cette valeur est élevée, moins de flux lumineux arrive dans le nuage, donc l'in-scattering y est plus sombre.

Si un rayon parcourt un petit segment Δs dans un voxel de densité ρ , l'épaisseur optique correspondante est

$$\tau = \sigma_t \rho \Delta s,$$

où τ est une grandeur qui quantifie la probabilité cumulée qu'un photon soit absorbé ou diffusé au long de ce trajet élémentaire (plus τ est grand, plus la lumière est atténuée).

D'après la loi de Beer–Lambert, l'atténuation de l'intensité lumineuse vérifie

$$E(\tau) = e^{-\tau}.$$

En réécrivant cette relation pour un usage pratique dans notre shader, on obtient la forme simplifiée :

$$E(b, d_s) = e^{-b d_s},$$

où $b = \sigma_t$ est le coefficient d'extinction total, et d_s la distance optique cumulée, c'est-à-dire la longueur effective parcourue par le rayon entre son point d'entrée et son point de sortie dans le nuage.



Figure 4.19 – Rendu avec couleur simple. $R : G : B = 1.3 : 1.2 : 1.1$



Figure 4.20 – Rendu après l'ajout de l'atténuation depuis la source , $\sigma_a = 0.6$, $\sigma_s = 0.4$

On distingue nettement la différence entre le rendu sans la loi de Beer–Lambert (figure 4.19) et celui avec (figure 4.20). Dans le second cas, la lumière n'est plus uniforme : les contours et les détails du nuage deviennent visibles, et l'on observe des zones plus claires ou plus sombres qui varient selon la direction et la position de la source lumineuse.

Atténuation depuis la source VS. Accumulation des pas Bien que les deux termes reposent sur la loi de Beer–Lambert, ils ne représentent pas la même portion du trajet optique (voir figure 4.21) :

- **Accumulation des pas**

Il s'agit de la transmittance *caméra* → *position échantionné*. À chaque incrément de marche Δs , on multiplie par $\exp(-\sigma_t \rho \Delta s)$. c'est-à-dire la fraction de lumière qui subsiste tout au long du chemin visuel.

- **Atténuation depuis la source**

Ici, on calcule la transmittance *soleil entrer du nuage* → *soleil sortir du nuage*. On lance un « rayon ombre » vers la source, on mesure la longueur L (exit - enter) et on applique

$$\text{light_atten} = e^{-\sigma_t L}.$$

Cette valeur exprime la quantité de flux lumineux.

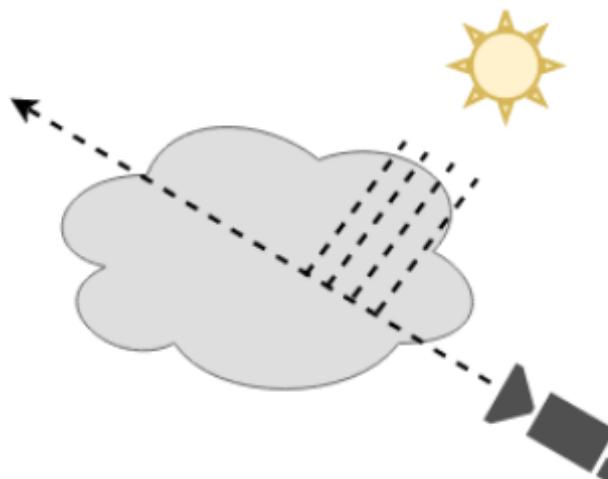


Figure 4.21 – Le modèle nuage - éclairage - caméra

Effet de bord argenté brillant Lorsqu'un nuage se place devant la source lumineuse, on observe que la lumière fait ressortir de façon particulièrement brillante les contours du nuage. Ce phénomène est appelé l'effet de bord argenté des nuages (cloud silver lining).



Figure 4.22 – Un exemple de l'effet de bord argenté brillant dans le monde réel

L'association de la **fonction de phase de Henyey–Greenstein** avec la loi de Beer–Lambert appliquée précédemment nous permet de reproduire cet effet lumineux (silver lining) autour des nuages.

$$p(\theta) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g \cos \theta)^{3/2}}$$

$p(\theta)$ module la diffusion selon l'angle θ entre la direction incidente et le regard : θ petit \Rightarrow intensité forte (liseré argenté). Le paramètre g contrôle l'anisotropie :

- $g \simeq 0$: diffusion isotrope;
- $g > 0$: diffusion avant (plus de lumière vers l'observateur);
- $g < 0$: diffusion arrière (réhausse un léger *rim-light*).

En ajustant g , on règle l'intensité du bord lumineux et des contre-jours.

En comparant les Fig. 4.23 et Fig. 4.24, on perçoit clairement la différence entre le rendu sans effet de bord argenté et celui intégrant cet effet.

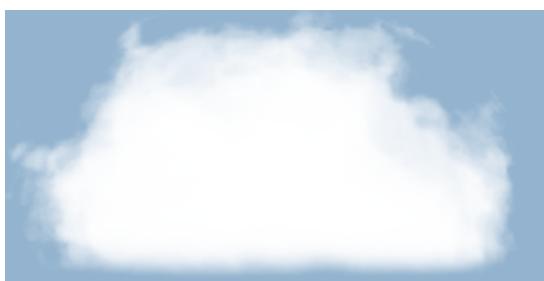


Figure 4.23 – Le rendu sans l'effect de bord argenté

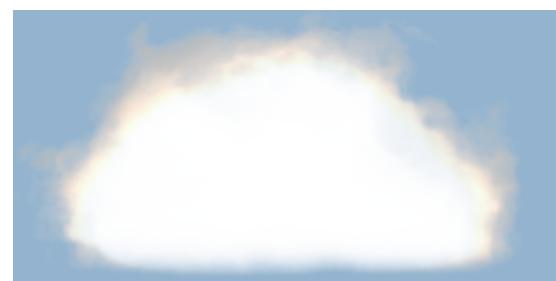


Figure 4.24 – Le rendu avec l'effect de bord argenté. $g = 0.8$

Diffusion sortante

Diffusion sortante (out-scattering) désigne le phénomène par lequel la lumière initialement propagée le long de la ligne de vue de l'observateur est diffusée par le milieu nuageux vers d'autres directions, réduisant ainsi la lumière qui atteint l'oeil. Il rend les contours du nuage ou les zones plus épaisses visiblement plus sombres.

Effet de bord argenté brillant PLUS

Parfois, on souhaite obtenir un effet de « silver lining » plus marqué, voire un liseré doré au crépuscule(figure 4.26). Dans ce cas, on peut combiner l'in-scattering et l'out-scattering pour obtenir cet effet.

- L'in-scattering se calcule de la même façon que précédemment.
- On ajoute ensuite une diffusion arrière via

$$HG(\theta, -out_s)$$

pour contrôler le niveau en noir du contour des nuages.

- Enfin, on interpole linéairement entre ces deux contributions pour les fusionner.



Figure 4.25 – Sans out-scattering



Figure 4.26 – in-scattering et out-scattering ensemble

Effet de poudre En générale, les nuages que l'on observe dans la vie réelle ne sont pas de simples nappes d'un blanc dans le ciel(figure 4.27). En particulier, pour les nuages relativement épais, on constate que leur surface présente des zones d'ombre(figure 4.29), à l'image de poudre illustrée à la figure 4.28. Nous désignons ce phénomène sous le nom d'« effet de poudre ».

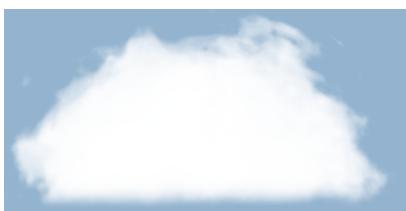


Figure 4.27 – Le rendu brut : peut perdre en réalisme s'il est trop blanc.



Figure 4.28 – L'effet de l'empiement des poudres est similaire à l'effet volumétrique des nuages. Illustré d'après [1]



Figure 4.29 – Des nuages épais dans la réalité

Il est difficile de reproduire ce phénomène avec une fonction purement physique; nous avons donc utilisé une fonction non-physique pour approximer cet out-scattering.

$$OS_{\text{ambient}} = 1 - \text{SAT}(o_{sa} \times d^{R(p_h, 0.3, 0.9, 0.5, 1.0)}) \times (\text{SAT}(R(p_h, 0, 0.3, 0.8, 1.0)))^{0.8}$$

- $SAT(x)$: $\text{clamp}(x, 0, 1)$;
- $Osa \in [0, 1]$: la quantité appliquée de l'environnement de diffusion sortant;
- d : la densité de nuage au point d'échantillonnage courant
- $p_h \in [0, 1]$: le pourcentage de hauteur.
- $R(x, A, B)$: Conversion linéaire du x de l'intervalle A à l'intervalle B

En multipliant cette OSambient au résultat de l'in-scattering, on obtient un renforcement des contours des nuages, avec des zones d'ombre plus prononcées sur leurs bords (figures 4.30 4.31 4.32).



Figure 4.30 – $o_{sa} = 0.83$



Figure 4.31 – $o_{sa} = 0.92$



Figure 4.32 – $o_{sa} = 0.96$

Approximation de la diffusion multiple

Pourquoi a-t-on besoin de la multi-diffusion?

Une diffusion simple (*single scattering*) éclaire uniquement la première rencontre soleil → voxel → œil . Le cœur des nuages reste alors fortement atténue et paraît parfois trop sombre.

Pour restituer la lumière qui rebondit plusieurs fois dans le volume — et donc l'aspect réel, plus *épais* et *doux* — il faut tenir compte des diffusions d'ordre supérieur.

Pourquoi une *approximation* de la multi-diffusion?

Une simulation exacte devrait relancer des rayons après chaque rebond, ce qui est prohibitif en temps réel.

Nous répétons plutôt l'intégrale dans le *même voxel*; à chaque ordre m les coefficients $\sigma_t^{(m)}, \sigma_s^{(m)}, g^{(m)}$ sont *exponentiellement réduits* ($a, b, c < 1$).

Ainsi, les hautes diffusions perdent rapidement de l'énergie :

$$\sigma_t^{(m)} = a^m \sigma_t, \quad \sigma_s^{(m)} = b^m \sigma_s, \quad g^{(m)} = c^m g, \quad 0 < a, b, c < 1$$



Figure 4.33 – Le rendu sans utilisant multi-diffusion, $\alpha_{sa} = 0.9$



Figure 4.34 – Le rendu avec l'utilisation de multi-diffusion, $\alpha_{sa} = 0.9, m = 3$

Chapitre 5

Comparaison des approches

Nous allons aborder dans ce chapitre une comparaison des différentes techniques implémentées au cours de ce projet. L'analyse repose sur plusieurs critères : la qualité visuelle, la complexité d'implémentation, l'aptitude à l'animation ainsi que les performances.

5.1 Ray-marching

Le ray marching s'impose comme la solution la plus complète et offre la meilleure restitution visuelle parmi les méthodes testées. Il permet un rendu réaliste avec une bonne perception du volume, en particulier grâce aux modèles physiques utilisés (loi de Beer–Lambert, fonction de phase de Henyey–Greenstein). Les effets lumineux comme le silver lining ou les ombres douces y sont reproduits avec précision.

Il s'agit cependant d'une solution difficile à intégrer dans un pipeline de rendu rasterisé classique. Il faut également faire attention à certains paramètres, le nombre de pas le long du rayon est à surveiller et peu rapidement ralentir le processus. Si l'on prend en compte qu'il faut échantillonner en direction de la source lumineuse à chaque pas, la complexité n'en est que plus grande. Il est donc impératif d'inclure certaines optimisations dès lors que l'on souhaite implémenter du ray marching (pas adaptatif, early-exit, amortissement du nombre de samples). D'autres solutions existent quant à l'optimisation comme présentées dans cet article [3].

5.2 Particules et imposteurs

L'approche par particules se distingue par sa flexibilité et sa capacité à produire des animations fluides. Le comportement dynamique des particules peut être contrôlé et simulé à l'aide des compute shaders pour gagner en performances et permettre un affichage des particules sous forme de grille d'occupation.

La visualisation des particules a été réalisée à l'aide d'imposteurs : des quads texturés orientés vers la caméra. Cette méthode est simple à implémenter et très performante. Néanmoins, elle présente une limite majeure en termes de réalisme. Les nuages apparaissent plats, les effets de volume sont absents, et l'éclairage ne tient pas compte des interactions entre particules (pas d'auto-ombres).

Une tentative de combinaison entre particules et grille de densité pour un rendu via ray marching a été étudiée, mais se heurte à une complexité d'implémentation notable (grille 3D dynamique, synchronisation).

5.3 Synthèse

En résumé :

- Le ray marching est la méthode la plus fidèle visuellement, au prix d'une forte complexité et de contraintes de performances.
- Les imposteurs sont simples, rapides et adaptés à l'animation, mais limités en qualité visuelle.
- Les particules offrent une base robuste pour simuler le mouvement, mais nécessitent un travail important pour restituer un rendu volumique convaincant.

Ces méthodes ne s'excluent pas mutuellement : dans une application temps réel, il serait pertinent d'utiliser le ray marching pour les nuages proches, et les imposteurs pour les masses lointaines, réduisant ainsi la charge de calcul tout en maintenant un bon niveau de réalisme.

Chapitre 6

Conclusion

Le rendu volumétrique des nuages constitue un domaine stimulant de l’infographie, combinant modélisation, éclairage et performance. Il soulève des défis techniques liés à la représentation de milieux diffus, à la gestion de la lumière dans les volumes, et à la contrainte de temps réel.

Dans ce projet, nous avons exploré plusieurs approches de génération de nuages : l’utilisation de formes géométriques enrichies par du bruit, les textures 3D, et la simulation de particules. Côté rendu, deux techniques principales ont été mises en œuvre : le ray marching, pour un rendu volumique réaliste, et les imposteurs, pour une visualisation plus légère.

Parmi ces méthodes, l’utilisation de textures 3D combinée au ray marching s’est révélée la plus convaincante visuellement. Elle permet de simuler avec précision les effets de diffusion de la lumière, et donne une forte impression de profondeur. Malgré son coût élevé, cette méthode reste pertinente, notamment si des optimisations adaptées sont mises en place.

La simulation par particules offre quant à elle une animation plus réaliste du mouvement des nuages, mais reste difficile à coupler efficacement avec un rendu volumétrique. L’utilisation d’imposteurs permet de contourner cette difficulté, au prix d’un réalisme limité mais avec un gain significatif en performances.

Ce travail a permis de mieux comprendre les enjeux liés à la représentation de milieux volumétriques, et de confronter plusieurs approches selon leurs atouts respectifs. Des perspectives d’amélioration subsistent, notamment dans la fusion des techniques ou l’optimisation du ray marching pour une utilisation temps réel.

Références

- [1] Nathan Vos Andrew Schneider. The real-time volumetric cloudscapes of horizon : zero dawn. <https://advances.realtimerendering.com/s2015/The%20Real-time%20Volumetric%20Cloudscapes%20of%20Horizon%20-%20Zero%20Dawn%20-%20ARTR.pdf>, 2015.
- [2] Auburn. Fastnoiselite - open source noise generation library. <https://github.com/Auburn/FastNoiseLite/tree/master>.
- [3] Fredrik Häggström. Real-time rendering of volumetric clouds. <https://www.diva-portal.org/smash/get/diva2:1223894/FULLTEXT01.pdf>, 2018.
- [4] Inigo Quilez. domain warping - 2002. <https://iquilezles.org/articles/warp/>, 2002.
- [5] Barbara Robertson. Volume : 32 issue : 6, the shape of animation. <https://www.cgw.com/Publications/CGW/2009/Volume-32-Issue-6-June-2009-/The-Shape-of-Animation.aspx>, 2009.