PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

| | |
|---|---|
| **Titre:** Title: | Visualisation de phénomènes gazeux |
| **Auteur:** Author: | Étienne Charbonneau-Lefort |
| **Date:** | 2004 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Charbonneau-Lefort, É. (2004). Visualisation de phénomènes gazeux [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7354/ |

**Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/7354/ |
| **Directeurs de recherche:** Advisors: | Benoît Ozell |
| **Programme:** Program: | Unspecified |

**Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal**
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

# VISUALISATION DE PHÉNOMÈNES GAZEUX

ÉTIENNE CHARBONNEAU-LEFORT
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
NOVEMBRE 2004

© Étienne Charbonneau-Lefort, 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

*Your file Votre référence*
*ISBN: 0-494-01297-8*
*Our file Notre référence*
*ISBN: 0-494-01297-8*

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

VISUALISATION DE PHÉNOMÈNES GAZEUX

présenté par : CHARBONNEAU-LEFORT, Étienne
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GRANGER, Louis, M.Sc., président
M. OZELL, Benoît, Ph.D., membre et directeur de recherche
M. REGGIO, Marcelo, Ph.D., membre

# REMERCIEMENTS

Je tiens à remercier de manière particulière mon directeur de maîtrise, M. Benoît Ozell, sans qui ce projet n'aurait pas pu être mené à bien. J'apprécie la grande latitude qu'il m'a offerte au niveau de la démarche et de l'avancement de mes travaux de recherche.

Je désire également remercier M. Jean-Philippe Hardy de m'avoir fait bénéficier de ses résultats de simulations numériques lors de mon projet. Ceux-ci m'ont permis de valider et d'améliorer mes algorithmes de modélisation.

# RÉSUMÉ

Dans le domaine de la visualisation scientifique, la modélisation des phénomènes gazeux suscite de plus en plus d'intérêt, mais demeure un défi de taille en raison de leur complexité. Trouver un compromis adéquat entre le réalisme et la performance est souvent indispensable. Ce travail se situe dans cette perspective. Nous avons développé un modèle permettant de représenter par un algorithme commun différents phénomènes gazeux à partir de données de densité de gaz. Deux paramètres, l'albedo et le niveau de transparence, permettent de spécifier l'apparence du gaz désirée.

La modélisation utilise comme structure de base une texture 3D comprenant des valeurs de transparence et de luminosité. La transparence est calculée à partir des valeurs de densité. La luminosité est obtenue par un algorithme d'illumination du gaz à partir d'une source lumineuse. Afin d'obtenir un niveau de réalisme intéressant tout en minimisant le plus possible le coût en calcul, nous effectuons un compromis au niveau du calcul de la diffusion de la lumière. Plutôt que d'évaluer la diffusion multiple réelle, nous utilisons deux phases de calculs. La première concerne la diffusion de la lumière dans la direction des rayons lumineux. La seconde calcule l'intensité de la lumière diffusée en direction de l'observateur. Pour y parvenir, les nuanceurs de sommets et de pixels sont mis à contribution.

Nous évaluons notre méthode selon trois critères : le réalisme du rendu, l'exactitude des données et l'interactivité avec l'observateur. Or, cette modélisation ne permet pas d'obtenir un niveau de réalisme parfait en raison des compromis effectués au niveau de l'illumination. Toutefois, elle engendre un rendu qui se rapproche suffisamment de la réalité pour être convaincant. D'autre part, cette méthode demeure représentative des données numériques et permet une interactivité fluide avec un observateur.

# ABSTRACT

In the field of scientific visualization, the modeling of gaseous phenomena rises great interest, but remains a challenge due to their complexity. There is a trade-off between realism and performance, and it is often essential to find an acceptable compromise between the two. This work aims to explore this question. We developed a model which uses a single algorithm to represent various gaseous phenomena given a certain gas density distribution. Two parameters, albedo and transparency, are used to specify the desired appearance of the gas.

The model generates a 3D texture containing the transparency and luminosity data. The transparency is calculated from the gas density, while the luminosity is obtained using an algorithm for gas illumination from a light source. In order to maintain an acceptable level of realism while minimizing the computational cost, we adopt a compromise regarding light scattering. Instead of computing the real multiple scattering, we use a two–step algorithm: we first relate to light scattering in the direction of the rays and then we calculate the intensity of the scattered light in the direction of the observer. The latter step is achieved using vertex shaders and pixels shaders.

We evaluate our method according to three criteria: realism, exactitude and interactivity. This modeling technique does not allow to reach a perfect realism because of the compromises regarding light scattering. However, it approaches reality sufficiently well to be convincing. In addition, this method remains representative of the numerical data and allows a fluid interactivity with the observer.

# TABLE DES MATIÈRES

REMERCIEMENTS ..... iv
RÉSUMÉ ..... v
ABSTRACT ..... vi
TABLE DES MATIÈRES ..... vii
LISTE DES TABLEAUX ..... x
LISTE DES FIGURES ..... xi
INTRODUCTION ..... 1

CHAPITRE 1 - DÉFINITIONS ET MISE EN CONTEXTE ..... 4
1.1 La visualisation scientifique ..... 4
1.2 Les textures 3D ..... 6
1.3 Nuanceurs de sommets, nuanceurs de pixels et le langage Cg ..... 8
1.4 Simulations numériques et FDS V3.10 ..... 10
1.5 Les phénomènes optiques ..... 12
1.5.1 Absorption ..... 12
1.5.2 Diffusion ..... 14
1.5.3 Fonction de phase ..... 16
1.5.4 Albedo ..... 18

CHAPITRE 2 - ÉTAT DE L'ART ..... 19
2.1 Modélisations représentatives des résultats numériques ..... 19
2.1.1 Champs vectoriels ..... 19
2.1.2 Surfaces iso-densité ..... 21
2.1.3 Lancer de rayon ..... 23
2.1.4 Modélisation par nuées (*blobs*) ..... 24
2.2 Modélisations « ontogénétiques » ..... 29
2.2.1 Systèmes de particules ..... 30

2.2.2 Utilisation de textures ..... 31
2.2.3 Méthode des ellipsoïdes ..... 34
2.2.4 Méthodes de projection ..... 39
2.3 La modélisation des phénomènes optiques ..... 43
2.3.1 Méthodes analytiques ..... 43
2.3.2 Méthodes statistiques ..... 46
2.3.3 Méthodes zonales ..... 47
2.3.4 Méthode des ordonnées discrètes ..... 48
2.3.5 Méthodes de rendu ontogénétiques ..... 49
2.3.6 Méthodes hybrides ..... 50

CHAPITRE 3 - SOLUTION ADOPTÉE 58
3.1 Objectifs visés ..... 58
3.2 Retour sur la revue de littérature ..... 59
3.3 Hypothèses soulevées ..... 62
3.4 Structure de la modélisation ..... 63
3.5 Calcul de la transparence ..... 64
3.6 Calcul de la luminosité ..... 68
3.7 Implémentation de l'algorithme de rendu ..... 70
3.7.1 Initialisation de la structure de données ..... 71
3.7.2 Calcul du rendu final ..... 73
3.8 Interface et design du logiciel de visualisation ..... 78
3.8.1 Fonctionnalités de l'outil de visualisation ..... 79
3.8.2 Design de l'implémentation ..... 85

CHAPITRE 4 - RÉSULTATS ET DISCUSSION 88
4.1 Exemples d'images obtenues ..... 88
4.1.1 Modélisation de fumée ..... 88
4.1.2 Modélisation d'explosion ..... 96

4.1.3 Modélisation de nuages ..... 98
4.2 Avantages et limites de la modélisation proposée ..... 101
4.3 Comparaison avec d'autres approches ..... 103
4.3.1 Comparaison avec VU ..... 103
4.3.2 Comparaison avec Smokeview ..... 104

CONCLUSION ET TRAVAUX FUTURS ..... 108
BIBLIOGRAPHIE ..... 110
ANNEXE ..... 115

# LISTE DES TABLEAUX

Tableau 1.1 Classement du type de fonction de phase utilisé ..... 16
Tableau 3.1 Évaluation des techniques de modélisation ..... 61
Tableau 4.1 Paramètres de simulation pour l'incendie dans une cuisine ..... 89
Tableau 4.2 Paramètres de simulation pour l'incendie dans une cuisine ..... 91
Tableau 4.3 Paramètres de la simulation de la pièce enfumée ..... 93
Tableau 4.4 Paramètres de la simulation de la maison incendiée ..... 95
Tableau 4.5 Paramètres de la simulation d'explosion ..... 97
Tableau 4.6 Paramètres de simulation de nuages ..... 99
Tableau 4.7 Évaluation de l'atteinte des objectifs ..... 103
Tableau 4.8 Comparaison avec les autres approches ..... 107

# LISTE DES FIGURES

Figure 1.1 Coordonnées d'une texture 3D ..... 7
Figure 1.2 Extraction d'une texture planaire à partir d'une texture 3D ..... 7
Figure 1.3 Étapes de rendu du pipeline graphique ..... 8
Figure 1.4 Processeur du *vertex shader* ..... 9
Figure 1.5 Exemple de division du domaine en volumes de contrôle ..... 11
Figure 1.6 Absorption ..... 13
Figure 1.7 Diffusion sortante ..... 15
Figure 1.8 Diffusion entrante ..... 15
Figure 2.1 Champ vectoriel pour un espace 2D ..... 20
Figure 2.2 Champ vectoriel pour un espace 3D ..... 21
Figure 2.3 Simulation d'explosion utilisant des surfaces iso-densité ..... 22
Figure 2.4 Surfaces de contour pour un ennuagement de 15% ..... 23
Figure 2.5 Ajout de la texture de transparence sur les surfaces de contour ..... 23
Figure 2.6 Modélisation hiérarchique à l'aide de blobs ..... 25
Figure 2.7 Blob se déformant dans le temps ..... 26
Figure 2.8 Fumée obtenue par la modélisation par blobs déformables ..... 27
Figure 2.9 Éclairage d'une scène utilisant des blobs ..... 29
Figure 2.10 Système de particules vu de très près ..... 30
Figure 2.11 Fumée générée par des textures procédurales ..... 31
Figure 2.12 Vapeur s'élevant d'une tasse ..... 32
Figure 2.13 Effet de brouillard non uniforme avec les textures de pixels ..... 33
Figure 2.14 Animation de nuages 3D ..... 34
Figure 2.15 Ellipsoïde avec une intensité texturée ..... 35
Figure 2.16 Ellipsoïde avec un seuil de transparence en 2D ..... 35
Figure 2.17 Ellipsoïde avec un seuil de transparence en 3D ..... 35
Figure 2.18 Combinaison de plusieurs ellipsoïdes ..... 35
Figure 2.19 Nuage formé par 27 ellipsoïdes ..... 36

Figure 2.20 Cumulus obtenus par Taxen ..... 37
Figure 2.21 Rendu d'une explosion ..... 39
Figure 2.22 Méthode de rendu par projection ..... 40
Figure 2.23 Formation de nuages ..... 40
Figure 2.24 Formation d'un nuage ..... 41
Figure 2.25 Utilisation d'imposteurs pour afficher des nuages ..... 42
Figure 2.26 Calcul d'illumination ..... 44
Figure 2.27 Nuages avec diffusion multiple ..... 46
Figure 2.28 Calcul de l'illumination d'un nuage par radiosité ..... 47
Figure 2.29 Simulation de diffusion multiple ..... 49
Figure 2.30 Diffusion multiple obtenue après 15 itérations ..... 51
Figure 2.31 Diffusion multiple de la lumière dans un gaz ..... 53
Figure 2.32 Illumination de fumée par tracé de photons ..... 56
Figure 2.33 Diffusion multiple (3 ordres) dans des nuages ..... 57
Figure 3.1 Application d'une texture 3D sur des plans orthogonaux ..... 63
Figure 3.2 Relation linéaire d'opacité ..... 65
Figure 3.3 Relation logarithmique d'opacité ..... 65
Figure 3.4 Relation exponentielle d'opacité ..... 65
Figure 3.5 Exemple de décalage des valeurs de densité ..... 66
Figure 3.6 Lancer de rayon pour calculer l'illumination ..... 69
Figure 3.7 Diffusion de la lumière vers l'observateur ..... 69
Figure 3.8 Sélection de plusieurs fichiers de simulation ..... 80
Figure 3.9 Spécification des paramètres de lecture des fichiers ..... 81
Figure 3.10 Spécification des paramètres de la texture 3D ..... 82
Figure 3.11 Spécification des paramètres d'illumination ..... 82
Figure 3.12 Édition de murs externes ..... 83
Figure 3.13 Édition de divisions internes ..... 84
Figure 3.14 Édition d'objets ..... 84
Figure 3.15 Principaux objets du programme ..... 85

Figure 4.1 État de la fumée après 100 secondes ..... 89
Figure 4.2 État de la fumée après 160 secondes ..... 90
Figure 4.3 État de la fumée après 250 secondes ..... 90
Figure 4.4 Fumée après 3 minutes ..... 92
Figure 4.5 Fumée après 15 minutes ..... 92
Figure 4.6 La fumée atteint la caméra ..... 93
Figure 4.7 La caméra se déplace dans la fumée ..... 94
Figure 4.8 Maison incendiée au temps de 85 minutes ..... 95
Figure 4.9 Maison incendiée au temps de 135 minutes ..... 96
Figure 4.10 Début d'explosion ..... 97
Figure 4.11 Fin de l'explosion ..... 98
Figure 4.12 Modélisation de nuages ..... 99
Figure 4.13 Modélisation de nuages légèrement ombragés ..... 100
Figure 4.14 Modélisation de nuages ..... 100
Figure 4.15 Fumée formée de plans semi-transparents créée par VU ..... 104
Figure 4.16 Fumée formée de plans semi-transparents créée par Smokeview ..... 105
Figure 4.17 Fumée formée de particules créée par Smokeview ..... 105
Figure 4.18 Fumée formée d'iso-surfaces créée par Smokeview ..... 106

# INTRODUCTION

## Motivation

Les phénomènes gazeux ont toujours suscité beaucoup d'intérêt et de curiosité. Par phénomènes gazeux, on peut regrouper les nuages, la fumée, la vapeur, le brouillard, le smog et même la poussière. Tous ces éléments possèdent un comportement complexe souvent indépendant de la volonté de l'observateur ou difficilement contrôlable ainsi qu'une structure non rigide difficile à modéliser. Cette complexité est la raison pour laquelle encore aujourd'hui plusieurs scientifiques s'évertuent à les étudier et à proposer différents modèles pour les représenter.

Les études scientifiques visant à comprendre et à simuler le comportement des gaz ont souvent recours à la visualisation scientifique pour valider leurs approches. La visualisation scientifique est également mise à profit pour l'étude de dispositifs qui interagissent avec des phénomènes gazeux, que ce soit la ventilation, la simulation d'incendies, ou des procédés industriels. En effet, l'objectif de la visualisation scientifique appliquée à ces domaines d'étude consiste à transformer des données strictement numériques en images pour pouvoir les étudier plus facilement. Dans bien des cas, la simulation par informatique s'avère essentielle, car reproduire la situation deviendrait beaucoup trop coûteux ou dangereux.

Il n'y a pas que dans le domaine scientifique où la représentation des phénomènes gazeux est devenue un défi de taille. Dans l'industrie du film, on tente de plus en plus de reproduire la nature pour créer des effets spéciaux. De même, pour les jeux vidéo, la demande toujours plus grande sur le plan du réalisme a amené les concepteurs à trouver de nouveaux algorithmes pour les représenter. On cherche alors à créer un rendu

convaincant de gaz, mais sans nécessairement respecter de façon rigoureuse les lois de la physique afin de conserver une interactivité acceptable.

Le réalisme, l'exactitude du rendu et l'interactivité avec l'observateur sont les trois éléments à tenir compte lors de la modélisation des phénomènes gazeux. Un rendu sera réaliste s'il laisse croire à l'observateur qu'il voit bel et bien une image de gaz tel qu'il serait possible de l'apercevoir dans une simulation réelle. Dans une situation idéale, le rendu devrait ressembler à une photographie. L'exactitude concerne le respect des données numériques et des lois de la physique. Il s'agit d'un concept crucial dans un contexte de visualisation scientifique où l'on désire habituellement représenter une situation concrète ou prédire un comportement réel. Pour une application aux jeux vidéo, on pourra sacrifier une partie de l'exactitude, en tentant toutefois de conserver un comportement raisonnablement exact. Finalement, une visualisation offrira un bon niveau d'interactivité si l'observateur peut influencer l'état de la visualisation, que ce soit par des déplacement ou des rotations, sans délais notables au niveau de l'affichage.

Ce travail s'effectue en coopération avec un projet de recherche en mécanique des fluides portant sur la simulation d'incendies. Ces simulations fourniront les résultats numériques dont nous nous servirons pour en calculer le rendu. Ainsi, ce travail porte uniquement sur la modélisation en vue de la visualisation de phénomènes gazeux et non pas sur leur simulation numérique.

**Méthodologie**

Dans un premier temps, nous effectuerons un survol de l'état de l'art. Les techniques de modélisation les plus fréquentes seront étudiées. Nous prendrons en considération à la fois les techniques provenant du domaine de la visualisation scientifique et celles

proposées pour les jeux vidéo et l'animation par ordinateur. Ainsi, nous obtiendrons un aperçu de modélisations proposant des degrés de réalisme et d'exactitude variés.

À la lumière de cette revue de littérature, nous décrirons la solution que nous proposons utilisant les textures tri-dimentionnelles comme éléments de base pour la modélisation. Cette avenue s'inspire de différentes techniques déjà existantes que nous avons adaptées pour nos besoins. Finalement, nous comparerons différents résultats obtenus suite à l'implémentation de cette solution avec ceux d'autres applications et nous discuterons de ses forces et ses lacunes.

# CHAPITRE 1 - DÉFINITIONS ET MISE EN CONTEXTE

Ce chapitre a pour but d'éclaircir certaines notions qui s'avéreront essentielles à la compréhension de ce travail et de mettre le lecteur en contexte. Ainsi, nous définirons brièvement ce qu'est la visualisation scientifique en mettant l'accent sur ses objectifs, les avantages que la recherche scientifique peut tirer de son utilisation de même que ses faiblesses. Ensuite, nous expliciterons une technique largement utilisée en infographie : les textures. De façon plus spécifique, nous nous concentrerons sur les textures 3D, lesquelles forment le pilier sur lequel repose notre technique de modélisation. La section qui suit traitera des nuanceurs de sommets et de pixels, ainsi que du langage Cg. Nous tiendrons compte uniquement du processus général du rendu graphique, sans entrer dans les détails d'implémentation matérielle. Par la suite, nous exposerons rapidement le fonctionnement du logiciel FDS V3.10 [38][39] ayant servi à générer les résultats numériques de phénomènes gazeux que nous utiliserons par la suite. Finalement, une section traitera de l'aspect théorique de l'illumination d'un gaz en décrivant les principaux phénomènes optiques à tenir compte.

## 1.1 La visualisation scientifique

Le principe de la visualisation scientifique est d'interpréter de l'information sous une représentation visuelle pour en permettre une analyse plus facile et efficace. Elle se base sur la capacité de l'être humain à comprendre et à analyser les images. Le défi de la visualisation consiste donc à calculer une représentation adéquate pour un ensemble de données brutes. La difficulté repose souvent au niveau de la très grande taille du volume de données et potentiellement de son aspect multidimensionnel. Une autre difficulté courante reliée à la visualisation scientifique concerne la reconstruction des données. En effet, la résolution d'équations par méthode numérique implique une forme de discrétisation, puis une reconstruction par interpolation. Optimiser la transformation des

données pour augmenter la qualité visuelle et l'efficacité du temps d'affichage est un problème crucial.

Les bénéfices issus de la visualisation scientifique sont variés. Premièrement, la représentation visuelle requiert souvent moins d'espace mémoire que les données brutes. De plus, la représentation visuelle est beaucoup plus intuitive et plus universelle, qu'une codification scientifique spécialisée, ce qui facilite la communication d'informations. Les chercheurs ont souvent recours à la visualisation scientifique pour explorer les résultats de simulations numériques, pour les manipuler ainsi que pour en analyser les détails. Typiquement interactive, elle peut être utilisée tout au long du processus de recherche pour la compréhension du phénomène et la diffusion des résultats [30]. Elle peut également être très avantageuse au niveau de la vulgarisation scientifique.

Les apports de la visualisation scientifique touchent de nombreux secteurs d'activité, tant au niveau de l'exploration de grands volumes de données qu'à la conception de simulateurs. Nous pouvons entre autres mentionner les simulateurs de vol, l'étude de la mécanique des fluides ainsi que les simulateurs chirurgicaux où la visualisation est particulièrement bénéfique. En effet, la simulation sur ordinateur tend de plus à plus à remplacer la réalisation de coûteux prototypes [31].

Pour être efficace, la visualisation doit répondre à trois critères : le réalisme, l'exactitude et l'interactivité. L'image projetée doit permettre à l'observateur de bien comprendre le phénomène étudié et doit également être fidèlement représentative des données numériques. Elle doit également pouvoir donner l'impression que l'objet simulé est une entité réelle que l'on peut manipuler, faire pivoter sous différents angles ou même modifier. Si l'outil de visualisation permet une navigation intuitive et aisée de l'espace virtuel, l'image graphique devient donc un instrument de recherche des plus utiles.

Afin d'éviter les confusions, nous établirons pour ce travail la signification de quelques termes qui reviendront fréquemment. Tout d'abord, nous considérerons une « représentation » comme étant une manière de traduire de l'information sous une forme visuelle pour des fins de visualisation. Par « modélisation » ou « modèle », nous décrirons de façon générale toute procédure utilisée afin d'obtenir un rendu final du phénomène gazeux. Ainsi, une modélisation peut faire appel à différentes représentations. Par exemple, si on représente la présence de gaz par des sphères et le niveau de densité par une échelle de couleurs, un modèle pourrait combiner ces représentations pour décrire l'état d'un gaz. Il est important de mentionner que nous considérerons la modélisation comme étant distincte de la simulation numérique.

## 1.2 Les textures 3D

Une texture est un tableau de données que l'on peut appliquer sur une primitive géométrique (point, ligne ou polygone) [33]. Ces données peuvent être une couleur, une luminosité ou une transparence. Les valeurs individuelles d'une texture sont habituellement appelées texels. Lorsqu'une texture est appliquée à une structure géométrique, les sommets reçoivent une valeur de la texture et des interpolations sont effectuées pour calculer le rendu des pixels situés entre ceux-ci. Une texture peut posséder de une à trois dimensions. Les textures 2D sont les plus utilisées, afin d'appliquer des images sur des surfaces.

Les textures 3D sont souvent utilisées comme technique pour effectuer du rendu de volume afin de visualiser des tableaux de données en trois dimensions [32]. Les textures 3D peuvent être vues comme étant un ensemble de textures 2D, tel qu'illustré la Figure 1.1.

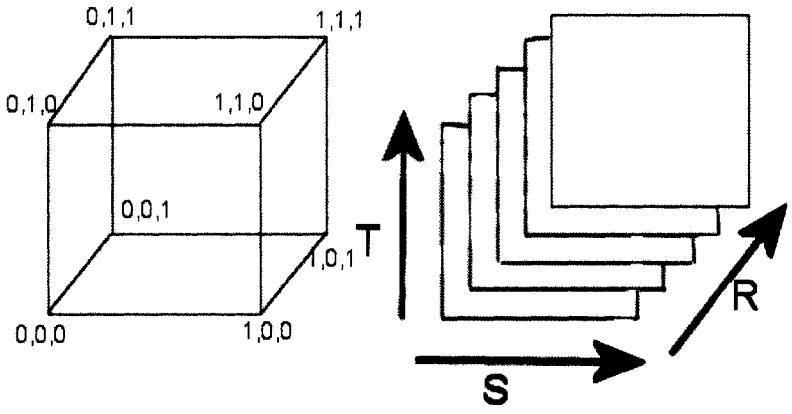



**Figure 1.1 Coordonnées d'une texture 3D**

La texture 3D est appliquée aux coordonnées (r,s,t) allant de (0,0,0) à (1,1,1) spécifiées pour les sommets d'un volume donné. Les textures 3D ont comme avantage par rapport aux textures 2D que le calcul d'interpolation est effectué dans les trois dimensions de l'espace par la carte graphique. La texture 3D peut également subir les transformations courantes (rotation, déformation, déplacement) selon les trois axes. Les valeurs de textures affichées correspondent à leur intersection avec le volume sur lequel elle est appliquée, comme le montre la Figure 1.2.

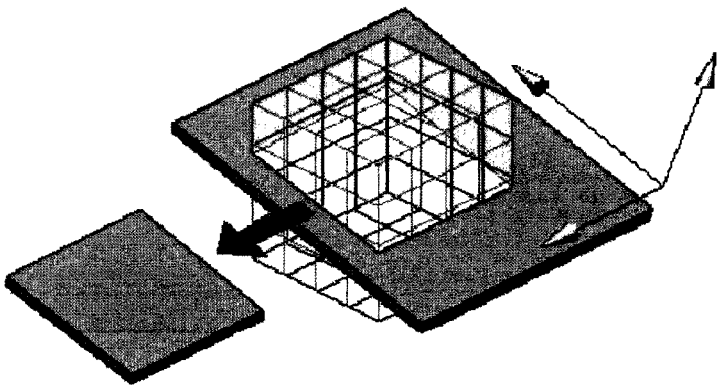


**Figure 1.2 Extraction d'une texture planaire à partir d'une texture 3D**

## 1.3 Nuanceurs de sommets, nuanceurs de pixels et le langage Cg

De façon simplifiée, le calcul du rendu graphique peut être divisé en trois étapes (Figure 1.3).

**Figure 1.3 Étapes de rendu du pipeline graphique**

Dans la phase du traitement des primitives géométriques, les sommets sont produits à partir des structures de données. Ces sommets décrivent des triangles, des points ou des lignes. Ensuite, le traitement des sommets peut être subdivisé en plusieurs étapes : les transformations, le calcul de l'éclairage et possiblement la transformation des coordonnées de texture. Les transformations consistent à transférer les coordonnées du monde virtuel aux coordonnées d'affichage. Le calcul de l'éclairage prend en considération les sources lumineuses pour déduire la couleur des sommets. Enfin, si des textures sont utilisées, leurs coordonnées sont également transformées pour être appliquées à l'espace d'affichage. L'étape finale consiste à prendre les données résultant de l'étape du traitement des sommets et à les placer dans la mémoire de trame comme valeurs de pixels.

Les opérations pour le traitement des sommets et des pixels sont à présent programmables. Au niveau du nuanceur de sommets, les sommets sont reçus en entrée et, suite à un traitement arbitraire, les valeurs de sortie (position, couleur, coordonnées de texture...) sont inscrites dans les registres. La Figure 1.4 schématise le traitement effectué par le nuanceur de sommets.

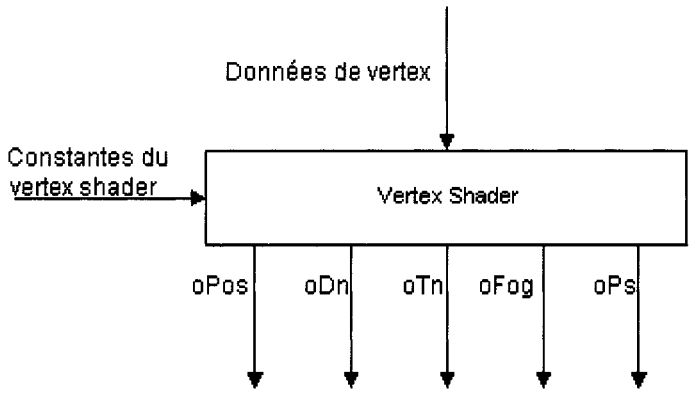



**Figure 1.4 Processeur du nuanceur de sommets**

Également programmable, le nuanceur de pixels calcule la couleur finale du pixel à afficher à partir de la couleur des sommets ainsi que des coordonnées de texture.

Les cartes graphiques récentes possèdent leur propre processeur (GPU) optimisé pour les calculs de rendu, ce qui le rend beaucoup plus performant que le processeur sur lequel fonctionne le système d'exploitation (CPU), conçu pour des traitements plus généraux. Le langage Cg (C for Graphics) [35] a été développé afin de permettre une programmation haut niveau des nuanceurs de sommets et de pixels. Il devient donc possible d'effectuer des calculs au niveau des sommets et des pixels directement sur le GPU plutôt que sur le CPU.

Le langage Cg peut être utilisé en combinaison avec OpenGL ou DirectX. Plusieurs effets de rendu peuvent ainsi être créés facilement et efficacement [35]. Mentionnons entre autres le placage d'environnement avec réflexion et réfraction ainsi que le placage de relief.

## 1.4 Simulations numériques et FDS V3.10

Dans cette section, nous décrirons sommairement le fonctionnement du logiciel FDS V3.10 ayant fourni les résultats numériques utilisés lors de nos modélisations. Il ne s'agit pas d'une description exhaustive des fonctionnalités et capacités du logiciel, mais plutôt d'un aperçu permettant de comprendre les principales techniques utilisées. Pour plus de précisions, il serait préférable de consulter le manuel d'utilisation [38] et le guide de référence [39] de FDS.

FDS simule les écoulements de fluide en divisant l'espace en petits volumes de contrôle. Les équations fondamentales de conservation de la masse, de la quantité de mouvement, des espèces et de l'énergie y sont appliquées et résolues pour chaque volume. Les propriétés de ces volumes (température, densité, vitesse...) sont mises à jour selon l'évolution du temps. La fiabilité des résultats dépend en grande partie du raffinement utilisé au niveau de la division en volumes : de très petits volumes permettront de s'approcher de la réalité, mais nécessiteront beaucoup plus de ressources informatiques. FDS ne permet l'utilisation que de parallélépipèdes rectangulaires comme volumes de contrôle (illustré à la Figure 1.5) afin de simplifier la résolution des équations. Une fois la géométrie créée, l'utilisateur doit spécifier les conditions frontières, les conditions initiales ainsi que les propriétés des matériaux.

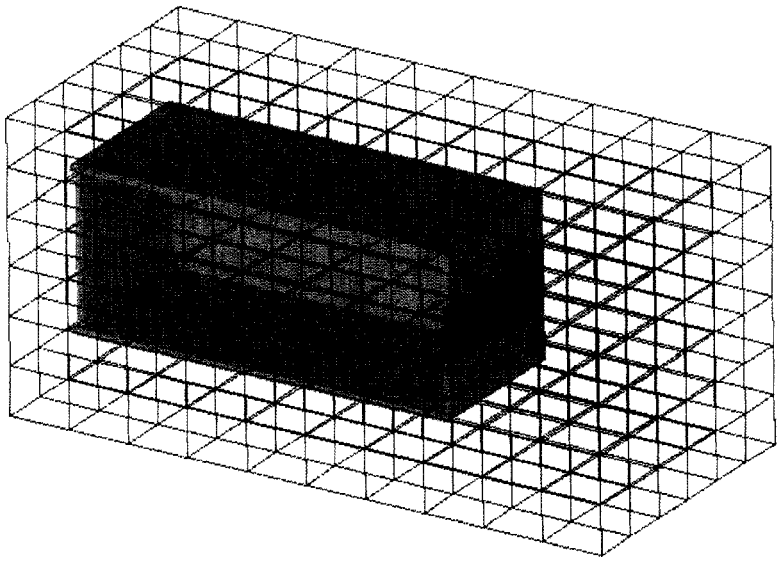


**Figure 1.5 Exemple de division du domaine en volumes de contrôle**[1]

Le logiciel FDS propose deux méthodes de résolution de turbulence : la méthode DNS (« Direct Numerical Simulation ») et la méthode LES (« Large Eddy Simulation ») [37]. La méthode DNS résout directement les termes dissipatifs, alors que la méthode LES résout la turbulence de grande échelle à partir d'une approximation des équations de Navier-Stokes et modélise les structures de petite échelle. L'utilisation de la méthode DNS se limite en général à des simulations de petite taille, car elle demande beaucoup de ressources informatiques. Quant à la méthode LES, elle est envisageable pour la plupart des simulations. Les résultats numériques que nous avons utilisés ont été obtenus uniquement par la méthode LES.

À chaque pas ($\Delta t$) de simulation, les approximations d'équations de Navier-Stokes sont résolues pour chaque volume, les différents paramètres du domaine sont mis à jour et ceux-ci sont enregistrés dans un fichier :

[1] Hardy [37]

- Température
- Vélocité selon les trois axes
- Densité
- Composition chimique
- Pression

La composition chimique Z varie entre 0 et 1, Z=0 correspondant à l'air pur, et Z=1 à du carburant pur. La densité de gaz ainsi que la composition chimique serviront lors de notre modélisation.

## 1.5 Les phénomènes optiques

Lorsqu'une onde lumineuse est propagée à l'intérieur d'un médium gazeux, différents phénomènes optiques surviennent, modifiant son énergie et sa direction. Nous allons décrire les deux phénomènes engendrant le plus de répercussions sur le plan visuel : l'absorption et la diffusion. Ensuite, nous parlerons de la fonction de phase et de l'albedo, qui sont deux autres éléments jouant un rôle déterminant au niveau de l'illumination d'un gaz.

### 1.5.1 Absorption

L'absorption correspond à une transformation d'énergie de sa forme ondulatoire à une forme vibratoire. Elle dépend des propriétés du gaz (température, pression, constituants…) et sa conséquence visuelle correspond à une diminution de l'intensité lumineuse (illustré à la Figure 1.6).

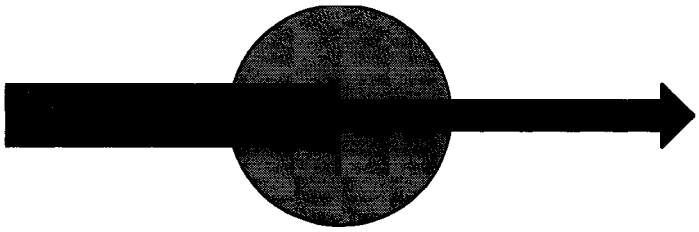


**Figure 1.6 Absorption**[2]

La lumière est atténuée en fonction de la grosseur et de la densité des particules. Le coefficient d'extinction, $\beta_{ex}(\lambda)$ est la quantité totale de lumière perdue par unité de distance par particule. La lumière restante après absorption le long du parcours du rayon allant du point « a » vers le point « b » dépend donc de la profondeur optique[3] :

$$I_{ab}(\lambda) = I_0(\lambda)\, exp(-\tau(P_a, P_b, \lambda))$$

$$\tau(a,b,\lambda) = \int_a^b \beta_{ex}(\lambda)\rho(s)ds$$

où $\lambda$ : Longueur d'onde de la lumière

$I_0(\lambda)$ : intensité initiale de la lumière (entrante)

$P_a$ : Point d'entrée « a »

$P_b$ : Point de sortie « b »

$\tau(P_a, P_b, \lambda)$ : Profondeur optique

$\rho(s)$ : Densité du médium à s

$\beta_{ex}(\lambda)$ : Coefficient d'extinction

[2] Da Dalto [27]
[3] Taxen G. [4]

1.5.2 Diffusion

La diffusion survient lorsque l'intensité lumineuse est retransmise dans une direction différente de la direction initiale. La lumière peut être diffusée par les particules atmosphériques dans différentes directions dépendamment de la grosseur des particules et de la longueur d'onde de la lumière. Pour une diffusion simple, l'intensité restante $I(\lambda)$ après diffusion peut être écrite[4]:

$$I(\lambda) = I_0(\lambda)\, \beta_{sc}(\lambda, \theta)$$

où $\lambda$ : Longueur d'onde de la lumière

$I_0(\lambda)$ : intensité initiale de la lumière (entrante)

$\theta$ : angle de diffusion

$\beta_{sc}(\lambda, \theta)$ : coefficient de diffusion (ou fonction de dispersion)

Si l'on ajoute le phénomène d'absorption, on peut calculer l'intensité de la lumière $I(\lambda)$ atteignant l'œil[5] :

$$I(\lambda) = I_{ab}(\lambda) + I_{sc}(\lambda)$$

$$I_{sc}(\lambda)=I_{sun}(\lambda)\int_{P_a}^{P_b}\beta_{sc}(\lambda,\theta)\exp(-\tau(P_b,l,\lambda)-\tau(l,P_a,\lambda))dl$$

où $I_{ab}(\lambda)$ : Lumière restante après absorption

$I_{sc}(\lambda)$ : Lumière restante après diffusion

$I_{sun}(\lambda)$ : Lumière de la source lumineuse (soleil)

$P_a$ : Point d'entrée

$P_b$ : Point de sortie

$l$ : Point d'observation

[4] Taxen G. [4]
[5] Taxen G. [4]

La diffusion sortante est l'émission de la lumière entrante dans différentes directions, comme le montre la Figure 1.7. Cela engendre donc une diminution de l'énergie transportée par le rayon.

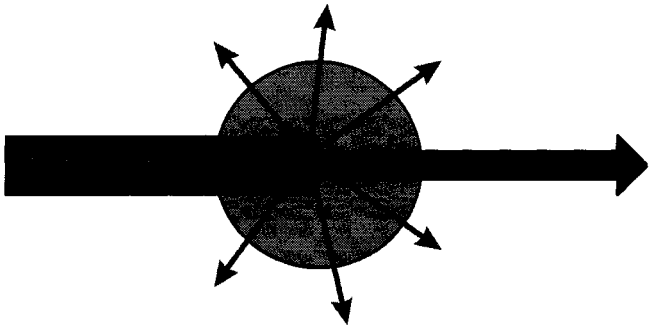


**Figure 1.7 Diffusion sortante**[6]

De même, si une particule reçoit la diffusion provenant de particules voisines, une augmentation du flux lumineux sera créée. Il s'agit de la diffusion entrante, telle qu'illustrée à la Figure 1.8.

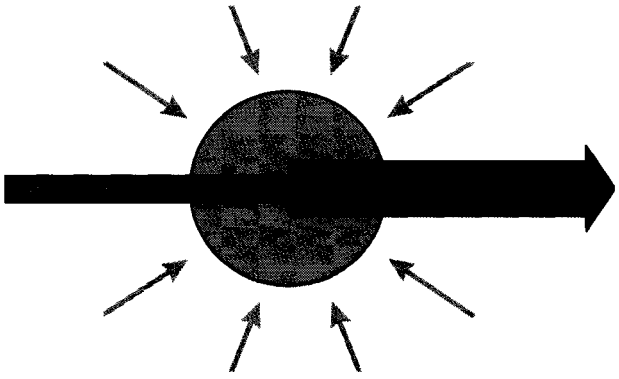


**Figure 1.8 Diffusion entrante**[7]

[6] Da Dalto [27]
[7] Da Dalto [27]

1.5.3 Fonction de phase

La fonction de phase détermine, pour un angle entre les directions incidentes et de sortie, la diffusion de l'intensité de la lumière. En général, la dispersion de la lumière s'effectue beaucoup plus dans le sens « avant » des rayons que par en arrière. Le choix de la fonction de phase théorique pour représenter le comportement du milieu est habituellement basé sur le rapport entre la grosseur de la particule (de rayon r) et la longueur d'onde de la lumière ($\lambda$). Normalement, il est préférable de définir une taille de particule moyenne et d'utiliser un seul type de fonction de phase pour tout le gaz. Inakage [29] propose le classement illustré par le Tableau 1.1.

**Tableau 1.1 Classement du type de fonction de phase utilisé**

| | |
|---|---|
| $r << \lambda$ | Absorption atmosphérique (pas de diffusion) |
| $r < \lambda$ | Diffusion de Rayleigh |
| $r \approx \lambda$ | Diffusion de Mie |
| $r >> \lambda$ | Optique géométrique (pas de fonction de phase) |

Taxen [4] considère que pour des particules de rayon allant jusqu'à environ $0{,}05\lambda$, la dispersion de Rayleigh est prédominante. Le coefficient d'extinction par molécule par unité de longueur est approximativement :

$$\beta_{ex}(\lambda) \approx \frac{8\pi^3 (n^2-1)^2}{3 \lambda^4 N}$$

où $N$ : nombre de particules (densité)

$n$ : index de réfraction

La fonction de diffusion par particule est approximativement :

$$\beta_{sc}(\lambda,\theta) \approx \frac{\pi^2 (n^2-1)^2}{2 \lambda^4 N}(1+\cos^2(\theta))$$

où $\theta$ est l'angle de diffusion

Le terme $1/\lambda^4$ des coefficients explique la couleur bleue durant le jour et le rouge au coucher du soleil.

Pour des particules plus grosses (brume, gouttelettes de nuage), la dispersion de Mie prédomine. On assume que les coefficients d'extinction et de dispersion sont indépendants de la longueur d'onde. Le coefficient d'extinction par kilomètre pour un nuage typique est :

$$\beta_{ex} = 16,33$$

Le coefficient de diffusion est approximativement :

$$\beta_{sc}(\theta) \propto \frac{3(1-g^2)}{2(2+g^2)} \frac{(1+\cos^2\theta)}{(1+g^2-2g\cos\theta)^{3/2}}$$

$$g = \frac{5}{9}u + \left(\frac{4}{3} - \frac{25}{81}u^2\right)x^{-1/3} + x^{1/3}$$

$$x = \frac{5}{9}u + \frac{125}{729}u^3 + \left(\frac{64}{27} - \frac{325}{243}u^2 + \frac{1250}{2187}u^4\right)^{1/2}$$

où $u$ varie entre 0,7 et 0,85

### 1.5.4 Albedo

On définit l'extinction comme étant l'atténuation de la lumière par absorption et par diffusion.

$$Extinction = Diffusion + Absorption$$

Le coefficient de diffusion est appelé « albedo ». Il représente le pourcentage d'atténuation par extinction due à la diffusion plutôt que par absorption. Une valeur d'albedo près de zéro correspond à une fumée très sombre, alors qu'une valeur d'albedo élevée (près de 1) convient à de la vapeur et à un nuage.

$$Albedo = Diffusion / Extinction$$

# CHAPITRE 2 - ÉTAT DE L'ART

Ce chapitre décrit différentes méthodes de visualisation des phénomènes gazeux proposées dans la littérature. Premièrement, nous allons exposer celles qui modélisent la structure du gaz. Nous avons subdivisé ces dernières en deux catégories : les méthodes permettant de respecter intégralement les données numériques et celles qui utilisent des procédures stochastiques. Finalement, nous allons décrire les modélisations qui traitent de l'illumination du gaz et qui tiennent compte des phénomènes optiques.

## 2.1 Modélisations représentatives des résultats numériques

### 2.1.1 Champs vectoriels

Une des premières modélisations utilisées en visualisation scientifique pour représenter un mouvement de fluide est le champ vectoriel. Les particules de gaz étant caractérisées par une direction et un module, il est possible d'échantillonner l'espace et de les représenter par des vecteurs.

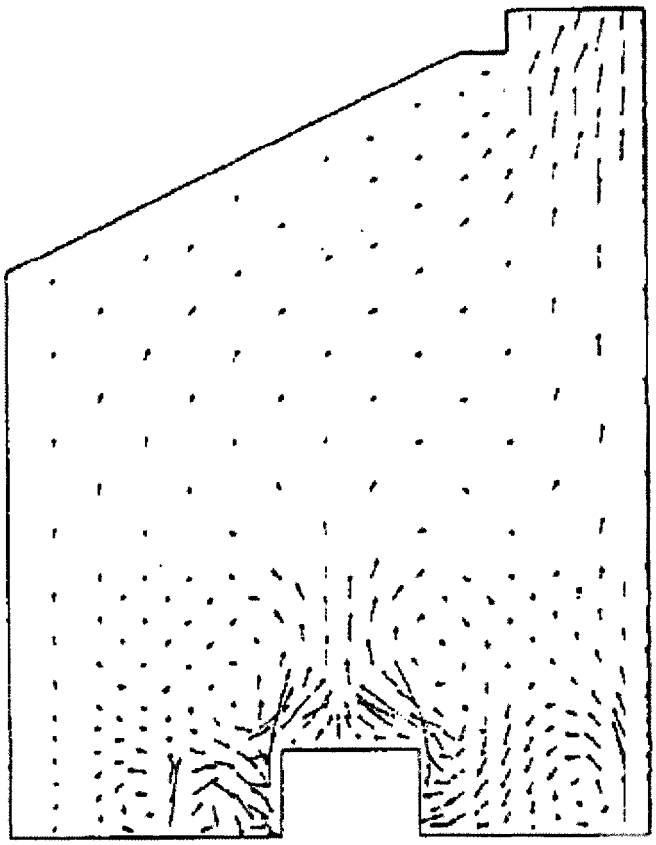


**Figure 2.1 Champ vectoriel pour un espace 2D[8]**

La Figure 2.1 donne un exemple de champ vectoriel pour un espace 2D. Ce type de représentation peut s'avérer utile pour déceler certaines tendances des données et extraire des informations rapidement. Par contre, elle présente des lacunes importantes. Premièrement, si les vecteurs reflètent bien les données numériques, l'aspect visuel obtenu, quant à lui, n'est pas réaliste. De plus, pour un espace en trois dimensions, les résultats visuels deviennent beaucoup plus ambigus, comme l'illustre la Figure 2.2.

[8] Post F. H. [36]

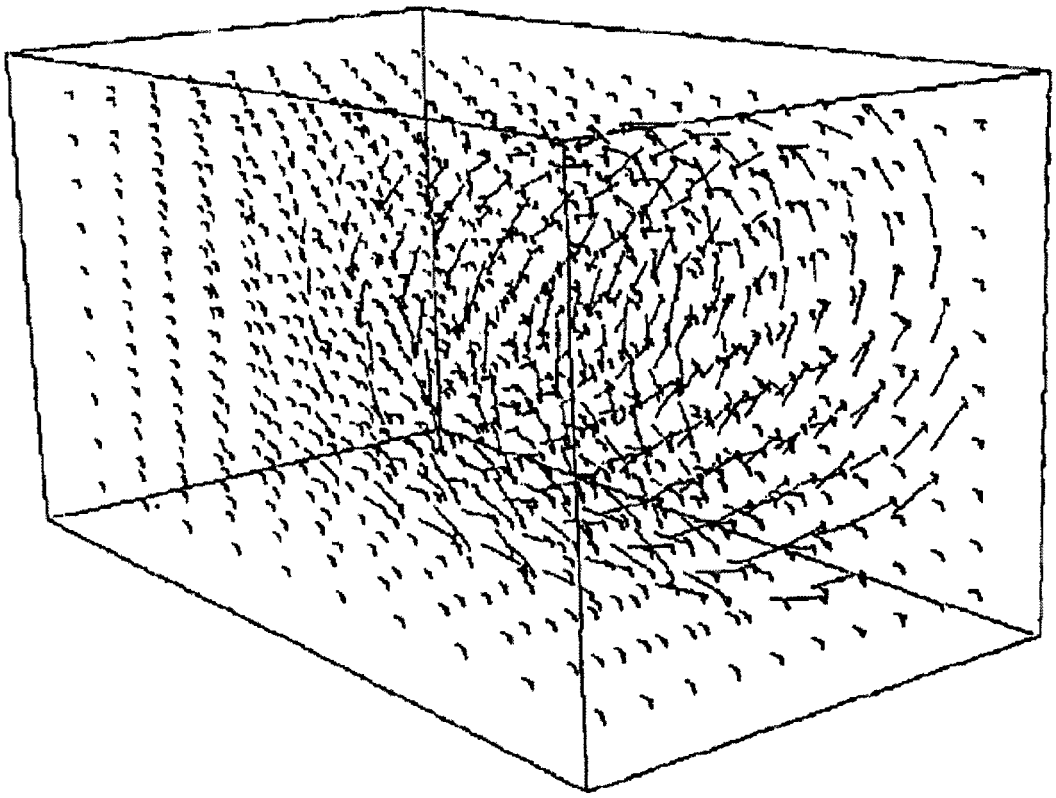


**Figure 2.2 Champ vectoriel pour un espace 3D[9]**

## 2.1.2 Surfaces iso-densité

Si des surfaces sont formées à partir de points de coordonnées offrant une même densité de gaz, il est possible d'obtenir un aperçu global du médium. Il s'agit d'une représentation simple nécessitant peu de calculs et permettant de retirer rapidement des informations visuelles. Par contre, cette méthode présente des lacunes importantes au niveau du réalisme et de l'exactitude des résultats. En effet, le calcul de la propagation de la lumière à travers le gaz n'est pas considéré lors du rendu des surfaces. De plus, les surfaces sont générées uniquement pour certaines valeurs de densité, délaissant les

[9] Post F. H. [36]

autres valeurs. Ceci engendre une perte de l'information lors de la visualisation. La Figure 2.3 présente l'affichage de cinq surfaces iso-densité obtenu par le logiciel VU[10] :

**Figure 2.3 Simulation d'explosion utilisant des surfaces iso-densité**

Max et Crawfis [3] proposent d'utiliser des surfaces de contour afin de représenter un ciel nuageux. Se basant sur des données climatiques réparties dans une grille tridimensionnelle, des tétraèdres sont construits. Ces derniers sont ensuite reliés entre eux en interpolant linéairement la fonction d'ennuagement (Figure 2.4). Finalement, une texture procédurale est utilisée pour contrôler la transparence des nuages (Figure 2.5).

[10] Ozell, B. [41]

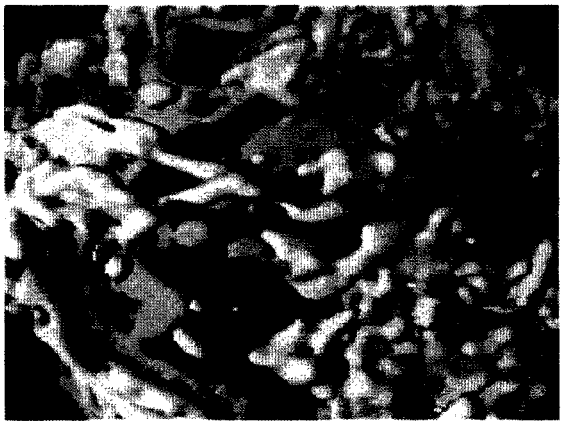


**Figure 2.4 Surfaces de contour pour un ennuagement de 15%**[11]

**Figure 2.5 Ajout de la texture de transparence sur les surfaces de contour**[12]

2.1.3 Lancer de rayon

Le lancer de rayon (« ray tracing ») est une méthode fréquemment utilisée pour obtenir des images de haute qualité visuelle. Pour chaque pixel d'affichage, un rayon est lancé et la couleur résultante dépend des objets de la scène rencontrés le long de son parcours.

Stam [13] utilise une grille 3D afin de calculer l'évolution de la densité en interpolant linéairement des données de densité entre deux étapes de temps. Par lancer de rayon, la transparence (*alpha*) et l'intensité (*intens*) des volumes sont ensuite obtenues comme suit :

$$alpha[i][j][k] = 1 - exp( -ex*coul*dens[i][j][k] * (l_x + l_y + l_z )/3 )$$

$$intens[i][j][k] = ( albedo * shadow( i, j, k, light ) * intens( light ) + em ) * coul$$

[11] Max et Crawfis [3]
[12] Max et Crawfis [3]

où *l_x*, *l_y* et *l_z* sont les espacements de la grille tridimensionnelle
*ex* est le coefficient d'extinction
*em* est l'émission du gaz
*coul* est la couleur du gaz
*dens* est la densité du gaz aux coordonnées (i,j,k)
*shadow()* est l'opacité causée par le gaz ou par d'autres objets gazeux
*light* est l'intensité du rayon lumineux

La méthode du lancer de rayon permet de générer un rendu très précis et de tenir compte d'éléments optiques tels que la réflexion et la réfraction. Toutefois, les ressources informatiques à ce jour ne sont pas suffisantes pour permettre un affichage fluide en utilisant uniquement cette technique.

### 2.1.4 Modélisation par nuées (*blobs*)

Une nuée peut être définie comme étant une sphère dans laquelle la densité de gaz est homogène. La modélisation par nuées échantillonne le volume gazeux uniquement aux endroits contenant de l'information, ce qui permet de minimiser l'espace mémoire requis. Par contre, plus le milieu est perturbé, plus les nuées devront être petites. De plus, pour éviter les artéfacts, un grand nombre de nuées est nécessaire. Pour limiter le temps de calcul et offrir un raffinement plus ou moins fin, une hiérarchisation des nuées peut être utilisée. Une arborescence utilisant des « macro-nuées » contenant des nuées plus petites est illustrée à la Figure 2.6.

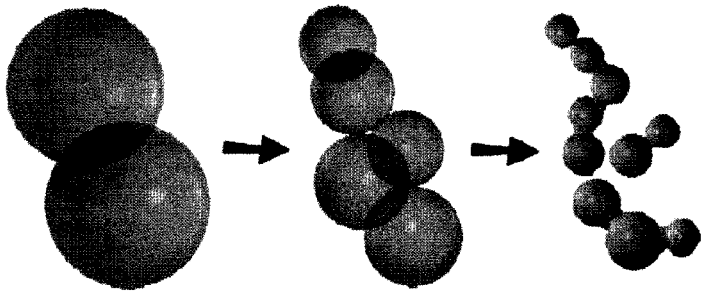


**Figure 2.6 Modélisation hiérarchique à l'aide de nuées**[13]

Stam [16] propose une méthode de rendu stochastique pour afficher des phénomènes gazeux représentés par des champs aléatoires de densité. Un champ de densité homogène ayant une covariance gaussienne est premièrement créé. Ce champ de densité aléatoire est vu comme une somme de nuées aléatoires, la variance de densité de chaque nuée étant proportionnelle à sa moyenne. Le rendu final est effectué par lancer de rayon. Pour chaque lancer de rayon, l'intensité de la lumière en absence de champ de densité est initialement calculée. Elle est ensuite modifiée selon les nuées. Pour chaque nuée *i* intersectant le rayon, on calcule la moyenne et la covariance le long du rayon au point d'intersection et on en effectue l'intégrale :

$$T_i = T_{i-1} + \overline{T_i(x,y)} + \sigma_i(x,y) Q_i(x,y)$$

où $\overline{T_i(x,y)}$ est la moyenne de la nuée au point d'intersection

$\sigma_i(x,y)$ est la covariance de la nuée au point d'intersection

$Q_i(x,y)$ est la valeur de densité du champ homogène au point (x,y)

$T_0 = 0$

La transparence est ensuite définie par une relation exponentielle :

$$\tau = \exp(-\kappa_t T)$$

[13] Barrero [28]

où $\kappa_t$ caractérise la quantité de lumière absorbée ou diffusée par unité de longueur

$T$ est l'accumulation de la densité le long du rayon

On prend ensuite en considération l'augmentation d'intensité (notée J) causée par l'émission et la diffusion de la lumière provenant d'autres directions. L'intensité finale du rayon sera :

$$I = \tau I_0 + (1-\tau)J$$

où $I_0$ est l'intensité en absence de champ de densité

La complexité de l'algorithme est de l'ordre de O(n) où n est le nombre de nuées.

Stam et Fiume [10][11] modélisent des ensembles de particules par des nuées. Sphériques initialement, les nuées grossissent et se déforment dans le temps sous l'effet du vent (Figure 2.7). À chaque étape de l'animation, la température et la densité des nuées sont mises à jour. L'illumination du gaz est calculée par un algorithme de lancer de rayon entre les sources lumineuses et les nuées. L'absorption, l'émission et la diffusion multiples sont prises en considération. Un exemple de rendu obtenu par Stam et Fiume [10][11] est illustré à la Figure 2.8.

**Figure 2.7 Nuée se déformant dans le temps**[14]

[14] Stam et Fiume [11]

**Figure 2.8 Fumée obtenue par la modélisation par nuées déformables[15]**

Stam [14] précise sa méthode de rendu utilisant la modélisation par nuées déformables. Cette modélisation ne fournit pas une précision numérique exacte, mais une approximation. Une structure hiérarchique de nuées est d'abord construite permettant de calculer la transparence et l'intensité par lancer de rayon traversant les nuées. La transparence du rayon traversant la nuée $k$ entre les points d'intersection $a$ et $b$ possède une allure gaussienne calculée comme suit:

$$T_k(b,a) \approx \frac{b-a}{\Delta_k} W(d_k,\varepsilon)$$

où $\Delta_k$ est la profondeur traversée dans la nuée

$d_k$ est la distance entre le centre de la nuée et le point d'intersection du rayon

$W(d_k,\varepsilon)$ est une fonction d'adoucissement

$\varepsilon$ est la plus petite valeur résolue par la méthode numérique

[15] Stam et Fiume [11]

Calculer la transparence totale du rayon correspond à effectuer la sommation de la contribution de chaque nuée interceptée et d'en calculer ensuite l'exponentielle inverse. L'intensité est ensuite obtenue par calcul de radiosité. À chaque étape de l'algorithme, l'énergie est propagée des sources de lumière aux nuées ainsi que de nuées à d'autres nuées.

Barrero [28] réutilise la méthode des nuées diffuses proposée par Stam [14][16] pour simuler l'animation de gaz dans le temps, mais y apporte des modifications pour corriger les problèmes causés par des nuées de trop grande taille. Lorsqu'une nuée atteint une taille maximale, elle est subdivisée en plusieurs nuées. De plus, lorsqu'une nuée a une densité plus petite qu'un seuil minimal, elle est supprimée. Le calcul d'illumination est effectué par un algorithme de radiosité. Les échanges d'énergie lumineuse sont calculés des sources aux nuées et des nuées à d'autres nuées. Finalement, lors du rendu, un lancer de rayon est utilisé en tenant compte de l'absorption et de la diffusion. La transparence $T_{tot}$ d'un pixel [x,y] à l'écran est le résultat de la multiplication des transparences de chaque nuée $T_j$ pour ce pixel :

$$T_{tot}[x,y]=\prod_{j=1}^{N} T_j[x,y]$$

La Figure 2.9 montre un exemple de fumée obtenue par Barrero [28].

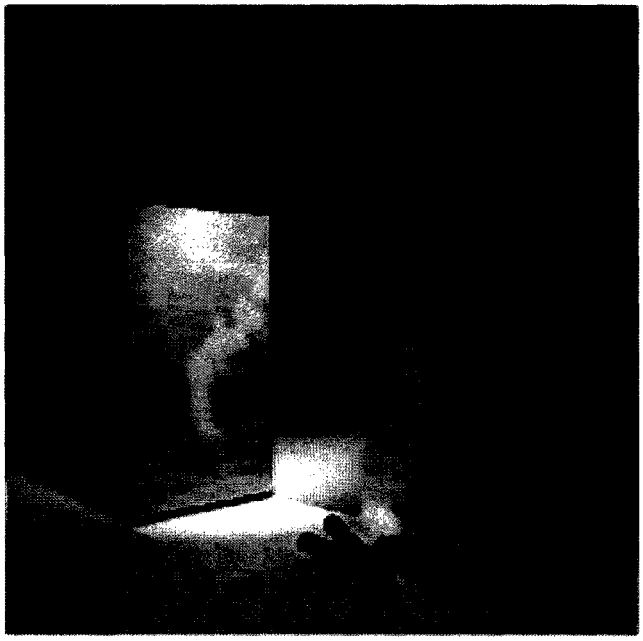


**Figure 2.9 Éclairage d'une scène utilisant des nuées[16]**

## 2.2 Modélisations « ontogénétiques »

La modélisation ontogénétique réfère à une modélisation se basant sur les caractéristiques morphologiques visibles. Elle vise l'atteinte des ressemblances subjectives plutôt que de viser la véracité issue des principes scientifiques. De nombreuses modélisations ontogénétiques de phénomènes gazeux ont été développées dans le but d'obtenir un rendu convaincant pour un observateur non scientifique. En effet, dans de nombreux secteurs tels que ceux du jeu vidéo et du cinéma, il n'est pas nécessaire de s'assurer que les gaz possèdent un comportement respectant de façon pointilleuse les lois de la physiques. L'effort est plutôt mis sur la beauté du rendu. Les modélisations ontogénétiques se basent souvent sur des techniques stochastiques.

[16] Barrero [28]

2.2.1 Systèmes de particules

La modélisation la plus simple et qui demeure très utilisée est le système de particules. Basée sur le modèle moléculaire des gaz, elle consiste à représenter le médium par un nuage de petites sphères semi-transparentes. Les forces entre les particules peuvent être facilement définies ainsi que leur trajectoire à travers un champ de turbulence. De plus, cette méthode est très simple à implémenter. Toutefois, pour que l'effet soit convaincant, un très grand nombre de particules est requis, ce qui représente une capacité de stockage mémoire importante. De plus, le nombre de particules ainsi que leur taille étant limités, il demeure toujours possible de distinguer les sphères si l'observateur s'en approche suffisamment près. La Figure 2.10 présente cet inconvénient :

**Figure 2.10 Système de particules vu de très près**

### 2.2.2 Utilisation de textures

Une technique souvent utilisée pour représenter des phénomènes gazeux consiste à décrire le volume gazeux par un espace solide à travers lequel on fait varier la couleur et la transparence. Des fonctions de densité habituellement aléatoires sont utilisées pour créer des variations au niveau de l'intensité et de l'opacité. L'un des inconvénients de cette technique réside dans le fait que le résultat visuel dépend entièrement du talent de l'artiste et que la procédure pour trouver une fonction de densité offrant des résultats réalistes s'effectue souvent par essais et erreurs.

King et al. [7] proposent une méthode simple pour créer une animation pour des phénomènes gazeux. Un ensemble de textures procédurales (bruit de Perlin ou autre) en teintes de gris est d'abord généré. Un volume contenant le gaz est ensuite défini et divisé en voxels espacés régulièrement. Une opacité est spécifiée pour chaque voxel et une texture initiale leur est appliquée. Pour chaque étape de l'animation, une nouvelle texture est apposée et l'image est affichée de l'arrière vers l'avant. Le volume peut également se déformer dans le temps. La Figure 2.11 illustre une fumée définie à l'intérieur d'un volume de forme elliptique.

**Figure 2.11 Fumée générée par des textures procédurales**[17]

[17] King et al. [7]

Ebert [22][23] construit des textures procédurales afin de spécifier les valeurs de transparence pour un volume de gaz. Deux fonctions sont à la base du calcul de la texture :

*densité = pow( turbulence * max, exp )*

*turbulence = ( 1 + sin( fast_turbulence( pnt )PI * 5 ) )0.5*

où *max* est la valeur de densité maximale (entre 0 et 1)

*exp* est le degré de l'exposant

*fast_turbulence* est une fonction aléatoire

Différentes modifications sont appliquées à la procédure de création de la texture selon le type de gaz désiré. Par exemple, pour simuler la vapeur s'élevant d'une tasse (Figure 2.12), la densité est amenuisée en fonction de la hauteur. Un paramètre de déplacement est également utilisé pour créer une animation.

**Figure 2.12 Vapeur s'élevant d'une tasse[18]**

[18] Ebert [22]

De leur côté, Heidrich et al. [8] utilisent les textures de pixels (une extension d'OpenGL) dans le but de générer du brouillard dont la densité n'est pas uniforme selon l'altitude. Une texture 3D contenant un calcul de distances selon la direction du regard est générée et appliquée aux pixels lors de l'affichage de la scène. Le résultat est illustré à la Figure 2.13.

**Figure 2.13 Effet de brouillard non uniforme avec les textures de pixels**[19]

Pour sa part, Stam [12] simule le mouvement de la fumée sous l'effet d'un écoulement d'air. Les valeurs de densité de gaz sont placées dans une grille 3D. Un résoluteur calcule les nouvelles valeurs de densité pour un intervalle de temps plus éloigné et des interpolations linéaires sont effectuées entre les deux grilles de densité. Ceci permet d'obtenir un mouvement fluide du gaz. Initialement, des coordonnées de texture 3D allant de 0 à 1 sont appliquées à la grille 3D. Durant la simulation, les coordonnées subissent les mêmes déformations que les valeurs de densité. Finalement, l'affichage est généré en faisant correspondre les coordonnées de texture à une texture 3D de type fractal et en multipliant la densité par une texture 2D (une image quelconque). La Figure 2.14 présente une animation de nuages obtenue par la méthode de Stam [12].

[19] Heidrich et al. [8]

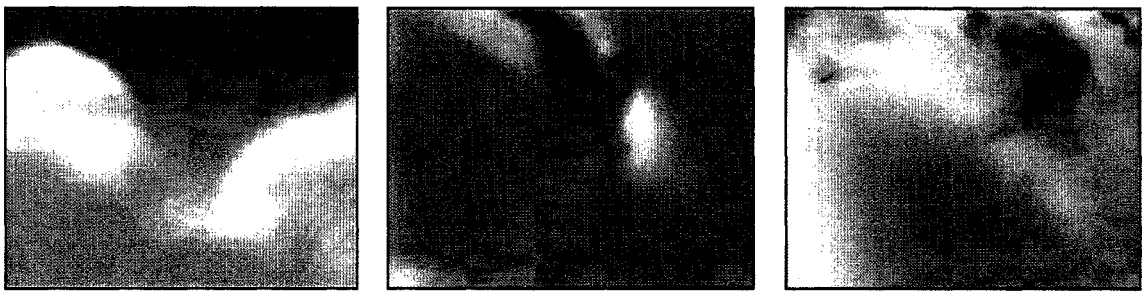


**Figure 2.14 Animation de nuages 3D[20]**

## 2.2.3 Méthode des ellipsoïdes

En raison de la forte similitude entre la forme d'un ellipsoïde et celle d'un nuage, de nombreuses techniques ont été développées utilisant cette géométrie comme structure pour le volume gazeux. Les ellipsoïdes présentent une forme simple et facile à contrôler.

Gardner [1] a été le premier à proposer la modélisation de nuages à l'aide d'ellipsoïdes et d'une texture générée par une fonction mathématique. La texture est créée par une série de Fourier composée d'une courte somme de sinus (entre quatre et sept) disposant de décalages aléatoires. Cette texture est appliquée sur les ellipsoïdes (Figure 2.15) pour en moduler l'intensité et la transparence (Figure 2.16 et Figure 2.17). Une transparence plus élevée est spécifiée près de la frontière des ellipsoïdes. En modifiant certains paramètres de la fonction mathématique, différents aspects visuels peuvent être obtenus, pouvant correspondre à un cirrus, un stratus ou un cumulus. Le nuage est finalement formé par la combinaison de plusieurs ellipsoïdes (Figure 2.18), disposant tous d'une couleur et de paramètres de texture communs. Par contre, leur grandeur, leur position ainsi que leur orientation sont modifiées par une variation aléatoire.

[20] Stam [12]

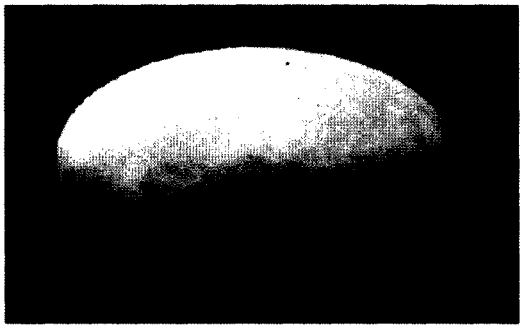


**Figure 2.15 Ellipsoïde avec une intensité texturée[21]**

**Figure 2.16 Ellipsoïde avec un seuil de transparence en 2D[22]**

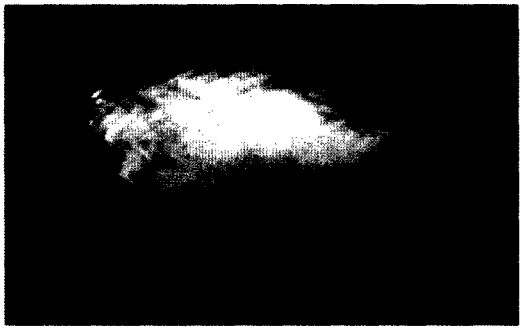


**Figure 2.17 Ellipsoïde avec un seuil de transparence en 3D[23]**

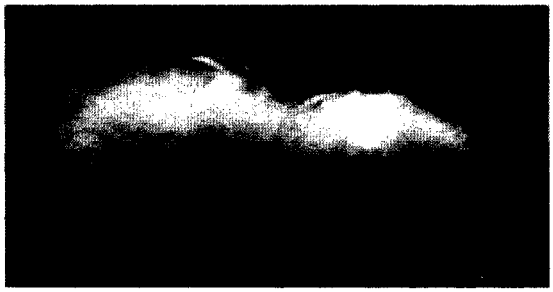


**Figure 2.18 Combinaison de plusieurs ellipsoïdes[24]**

Cette méthode peut être effectuée avec un coût de calcul suffisamment faible pour permettre l'affichage fluide d'une séquence d'images. De plus, elle permet une modélisation tridimensionnelle. Toutefois, elle ne considère pas l'illumination de la lumière à travers le gaz.

Elinas et Stuerzlinger [1] reprennent la méthode de Gardner utilisant un ensemble d'ellipsoïdes pour simuler la structure d'un nuage. Chaque ellipsoïde possède une texture irrégulière et est partiellement transparent. Toutefois, la texture est générée par

[21] Gardner [1]
[22] Gardner [1]
[23] Gardner [1]
[24] Gardner [1]

un bruit de Perlin plutôt qu'une série de sinus, permettant une plus grande flexibilité et un meilleur contrôle. Afin de s'assurer que l'intérieur de l'ellipsoïde offre une plus grande opacité que l'extérieur, une texture semblable à un éclairage de projecteur est préalablement appliquée sur chaque ellipsoïde. La Figure 2.19 illustre un rendu de nuages obtenu par Elinas et Stuerzlinger [1].

**Figure 2.19 Nuage formé par 27 ellipsoïdes**[25]

Taxen [4] reprend l'approche de Gardner [1] en ce qui concerne la modélisation du nuage, sauf qu'il propose un bruit de Perlin comme texture. Pour les calculs d'intensité, il effectue un lancer de rayon où chaque rayon est modifié par une fonction d'opacité plutôt que de calculer la véritable intensité de voxel à voxel. Les nuages de la Figure 2.20 résultent de cette méthode.

[25] Ellinas [1]

**Figure 2.20 Cumulus obtenus par Taxen[26]**

Rasmussen et al. [18] construisent un champ de vélocité 3D à partir de solutions d'équations de Navier-Stokes 2D. Des particules sont ensuite propagées à travers ce champ. Une grille tridimensionnelle est également créée, chaque voxel ayant la forme d'une pyramide tronquée alignée dans la même direction que celle de la caméra. Les densités de particules sont ensuite échantillonnées dans la grille en les considérant comme étant des ellipsoïdes. Chaque ellipsoïde possède un rayon pour chaque axe de coordonnées ainsi qu'une densité au point $p$ (inclus dans l'ellipsoïde) tel que :

$$D(p)=1-f(1-s,1,|p|/r)$$

$$f(a,b,t)=\begin{cases} 0 & \text{si } t\le a \\ 1 & \text{si } t\ge b \\ -2\left(\frac{t-a}{b-a}\right)^3+3\left(\frac{t-a}{b-a}\right)^2 & \text{si a<t<b} \end{cases}$$

où $0\le s\le 1$ est un facteur de dégradation

$r$ est le rayon de l'ellipsoïde passant par $p$

Une fonction de turbulence (bruit de Perlin) est utilisée pour moduler la fonction de densité. L'illumination directe du volume est calculée à chaque voxel par lancer de

[26] Taxen [4]

rayon. Après avoir emmagasiné les valeurs d'illumination à chaque voxel, la diffusion isotropique de la lumière est ensuite simulée en utilisant une méthode hiérarchique. Le rendu du volume est finalement effectué par lancer de rayon en accumulant l'intensité et l'opacité. L'opacité d'un voxel centré au point x est :

$$a = 1 - exp( -\tau * D(x) * dz )$$

où $D(x)$ est la densité du voxel

$dz$ est la profondeur du voxel dans la direction du rayon

$\tau$ est une constante contrôlant la conversion entre la densité et l'opacité

L'opacité est accumulée le long du rayon tel que :

$$A_{n+1} = A_n + a( 1 - A_n )$$

Finalement, à chaque point du volume la couleur est pondérée par l'opacité :

$$C_{n+1} = C_n + a( 1 - A_n ) I(x)$$

où $I(x)$ est l'illumination du voxel

La Figure 2.21 illustre un rendu d'explosion simulée par Rasmussen et al. [18].

**Figure 2.21 Rendu d'une explosion[27]**

2.2.4 Méthodes de projection

La nature tri-dimensionnelle des phénomènes gazeux est grandement responsable de la complexité du calcul de rendu d'une scène nécessitant l'interaction avec un observateur. Les méthodes de projection visent à réduire ce coût en performance en réduisant le volume gazeux à une surface 2D.

Dobashi et al. [5] présentent une méthode pour produire une animation de nuages (création, extinction, déplacement) dans le temps. L'espace est divisé en voxels, chacun possédant trois paramètres booléens (humidité, nuage, activation) qui effectuent des transitions. Cette méthode ne donne pas des résultats physiquement exacts, mais permet d'être visuellement convaincante. La fonction de distribution de densité est exprimée par un ensemble de métaballes. Pour calculer l'intensité lumineuse des pixels, une diffusion simple de la lumière à travers le nuage est considérée en calculant l'intensité

[27] Rasmussen et al. [18]

lumineuse traversant les métaballes. Ensuite, la transparence est obtenue en projetant chacune d'elles sur une surface plane perpendiculaire à l'angle de vue. L'affichage final résulte de la combinaison de ces projections. Les étapes de cette méthode sont présentées à la Figure 2.22 et la Figure 2.23 illustre un exemple de nuages obtenus par Dobashi [5].

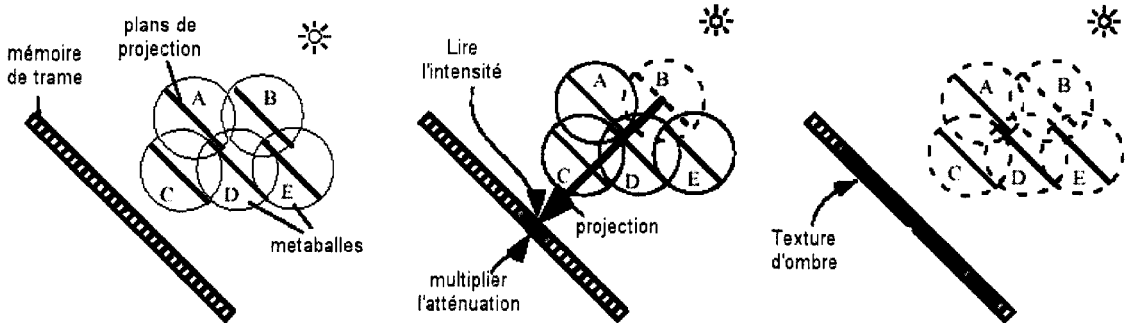



**Figure 2.22 Méthode de rendu par projection**

**Figure 2.23 Formation de nuages[28]**

Heinzlreiter et al. [6] utilisent des ellipsoïdes déterminés par la position du centre, la longueur des axes, la densité au centre et la densité à la surface pour approximer la

[28] Dobashi et al. [5]

distribution spatiale de la densité de vapeur d'eau. La distribution de densité à l'intérieur de chaque ellipsoïde est calculée par interpolation linéaire entre ses valeurs de densité au centre et à la surface. Le calcul de l'absorption de la lumière entre les voxels est effectué de façon récursive. La diffusion est également calculée en utilisant la fonction de phase de Mie avec trois itérations. Toujours dans la phase de pré-traitement, des images sont générées par lancer de rayon selon différents points de vue (un minimum de 32) autour du nuage pour créer des textures de projection. Une opacité est attribuée à chaque texture selon leur position avant de les superposer sur un plan de projection. Une texture pour l'ombre sur le sol peut également être créée en utilisant un point de vue directement au-dessus du nuage. L'image de la Figure 2.24 résulte de cette méthode de rendu.

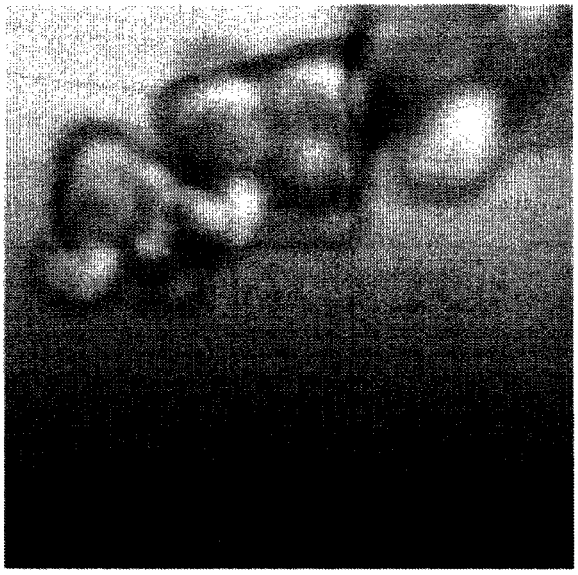


**Figure 2.24 Formation d'un nuage[29]**

Harris et Lastra [9] proposent d'utiliser des « imposteurs » pour diminuer la complexité des calculs lors de l'affichage de chaque image d'une animation incorporant des nuages. Un imposteur remplace un objet dans la scène par un plan semi-transparent sur lequel on

[29] Heinzlreiter et al. [6]

appose une texture. Ces plans de projection sont illustrés à la Figure 2.25. La texture appliquée aux plans correspond à l'image de l'objet remplacé vu de l'endroit où se situe la caméra. Étant donné que le point de vue par rapport à certains nuages change peu, un même imposteur peut être utilisé pour plusieurs images de l'animation. Le temps de calcul pour le rendu de nuages tridimensionnels s'en trouve amenuisé.

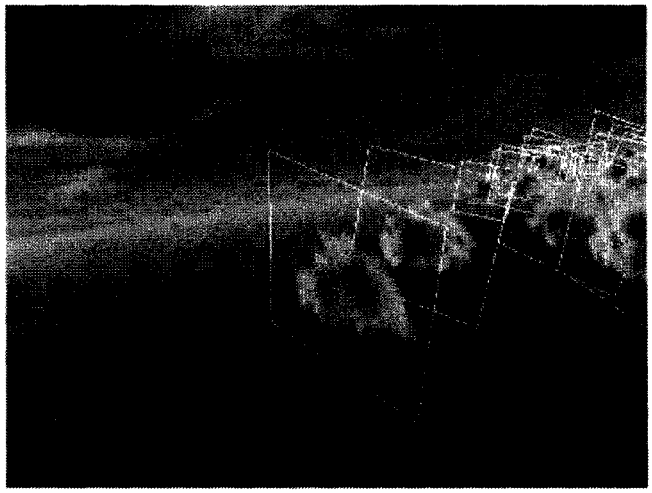


**Figure 2.25 Utilisation d'imposteurs pour afficher des nuages[30]**

[30] Harris et Lastra [9]

## 2.3 La modélisation des phénomènes optiques

### 2.3.1 Méthodes analytiques

Le calcul de l'illumination ne peut être résolu sous sa forme théorique de façon analytique. Il est nécessaire d'intégrer les équations de transfert d'énergie à l'intérieur du milieu pour finalement en connaître le bilan. Le principe consiste à échantillonner le milieu par lancer de rayon et d'en obtenir un système que l'on peut résoudre analytiquement. Ceci permet de calculer la diffusion simple, mais non la diffusion multiple. Toutefois, l'efficacité de ces algorithmes est élevée.

Kajiya et Herzen [24] proposent un algorithme de lancer de rayon pour simuler l'illumination dans le cas d'un albedo faible. Dans cette procédure, le calcul de rendu est divisé en deux étapes. Premièrement, la propagation de la lumière provenant des sources i à travers un tableau de densité $\rho(x,y,z)$ est calculée dans un second tableau $I_i(x,y,z)$ en tenant compte de la contribution de chaque lumière à l'illumination de chacun des points de l'espace. Ceci est effectué en calculant en parallèle l'intégrale pour chaque parcours $T_{x,y,z}=(x(t),y(t),z(t))$ de la source de lumière à travers $\rho(x,y,z)$.

$$I_i(x,y,z)=\exp\left(-\tau \int_{T_{x,y,z}} \rho(x(t),y(t),z(t))dt\right)$$

$$\tau = \kappa / \rho$$

où $\kappa$ est le coefficient d'absorption

Pour la seconde étape, chaque rayon est recueilli par une frontière en prisme rectangulaire, la clarté du rayon correspondant à la somme des contributions de chaque voxel.

$$B=\int_{\lambda 1}^{\lambda 2} e^{-\tau\int_{\lambda 1}^{t}\rho(x(\mu),y(\mu),z(\mu))d\mu}\times\left[\sum_i I_i(x(t),y(t),z(t))p(\cos\theta_i)\right]\times\rho(x(t),y(t),z(t))dt$$

où $\lambda_1$ et $\lambda_2$ sont le début et la fin du trajet entre l'œil et le voxel visible le plus éloigné.

$\theta_i$ est l'angle entre la direction du regard et le rayon pour le voxel $i$

$p(cos(\theta_i))$ est la fonction de phase

Le terme exponentiel de l'intégrale donne l'atténuation causée par l'absorption et la diffusion, tandis que la sommation fournit la contribution en clarté de chaque source de lumière pour un point donné. La Figure 2.26 donne un exemple de rendu obtenu par la méthode de Kajiya et Herzen [24].

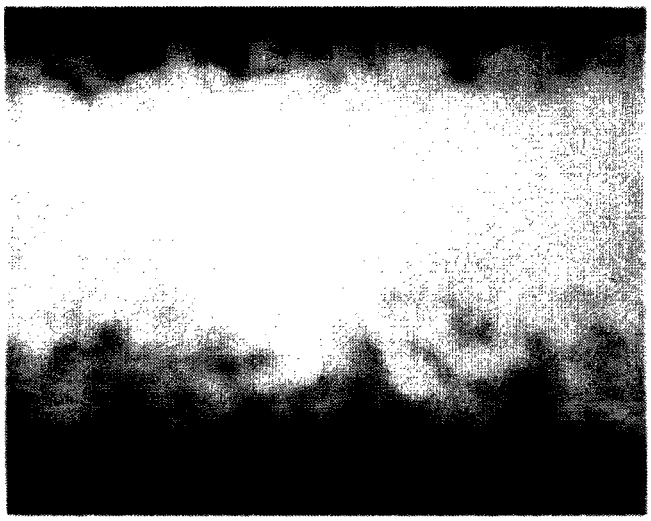


**Figure 2.26 Calcul d'illumination**[31]

La diffusion par des particules plus grosses est plus complexe et est décrite par Mie. C'est notamment le cas des particules de nuage, qui s'apparentent davantage au régime de Mie qu'à celui de Rayleigh. Toutefois, Harris et Lastra [9] considèrent qu'il est

[31] Kajiya et Herzen [24]

possible d'obtenir des résultats visuels intéressants par la formule de Rayleigh, mais en utilisant une simplification de cette fonction de phase :

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

où $\theta$ est l'angle entre la direction incidente et la direction de diffusion

Étant donné que le calcul de la diffusion anisotropique multiple est coûteux, les auteurs simplifient en calculant uniquement la diffusion dans la direction de la lumière. Par la suite, la diffusion simple vers l'observateur est ajoutée :

$$E_k = S_k + T_k * E_{k-1}, \qquad 1 \leq k \leq N$$
$$S_k = a_k * \tau_k * p(\theta) * I_k / 4\pi$$

où $E_k$ : Lumière sortant de la particule $p_k$

$S_k$ : Lumière dispersée

$T_k$ : Transparence de la particule

$T_k * E_{k-1}$ : Lumière non absorbée

$\theta$ : Angle entre la direction de l'observateur et celle du rayon

$p(\theta)$ : Fonction de phase

$\tau_k$ : Coefficient d'extinction (1/longueur) du nuage de profondeur t

$a_k$: Albedo

$I_k$ : Intensité de la lumière

La Figure 2.27 illustre un exemple de nuages obtenus par Harris et Lastra [9]. Les auteurs utilisent un coefficient d'extinction $\tau = 8{,}0$ et un albedo $a = 0{,}9$. De plus, une illumination globale du ciel est modélisée par de multiples sources de lumière.

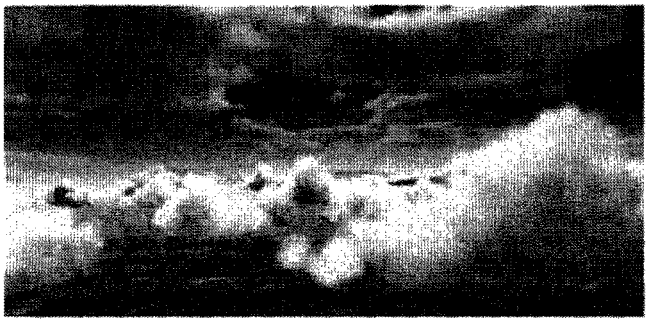


**Figure 2.27 Nuages avec diffusion multiple[32]**

## 2.3.2 Méthodes statistiques

La méthode statistique la plus utilisée pour la modélisation de phénomènes gazeux est l'algorithme de Monte-Carlo. Son principe consiste à simuler de manière aléatoire les phénomènes lumineux à l'intérieur du gaz. Il s'agit d'un algorithme non déterministe. Plusieurs chemins à travers le médium sont parcourus pour simuler la diffusion simple.

Pattanaik et Mudur [21] présentent une technique de simulation de Monte-Carlo pour calculer l'illumination à travers un médium gazeux. Un nombre fini de particules est généré à différentes positions sur la surface (ou sur le volume pour une simulation en 3D) émettant la lumière et différentes directions de propagation leur sont assignées. Pour chaque particule, on calcule la surface d'intersection la plus près. À ce point, la particule est aléatoirement absorbée ou diffusée avec une probabilité basée sur le coefficient d'extinction, l'albedo et la fonction de phase. Le processus est poursuivi pour chaque particule jusqu'à ce qu'elles soient toutes absorbées ou qu'elles aient parcouru tout l'environnement.

[32] Harris et Lastra [9]

Pour que l'algorithme de Monte-Carlo génère un rendu réaliste, il faut que le nombre de lancers de photon soit très grand, sinon des erreurs visuelles restent présentes. Une simulation réaliste nécessite un coût de calcul très élevé.

### 2.3.3 Méthodes zonales

Les méthodes zonales divisent l'espace du volume gazeux en voxels afin de simuler les échanges d'énergie par radiosité. Sillion [17] présente une technique pour le calcul de la radiosité dans un volume complexe où s'effectue une diffusion d'énergie. La stratégie consiste à construire une hiérarchie ascendante (« bottom-up ») regroupant les surfaces proches. L'algorithme utilise une procédure hiérarchique : considérant deux volumes, soit qu'il est décidé que le transfert d'énergie est correctement représenté au niveau actuel, soit qu'un des deux volumes est subdivisé. Le transfert d'énergie est calculé en traversant la structure hiérarchique et en accumulant l'opacité le long du parcours. Chaque volume possède une densité et un albedo. L'image finale est générée par lancer de rayon. La Figure 2.28 montre un exemple d'illumination obtenue par la méthode zonale.

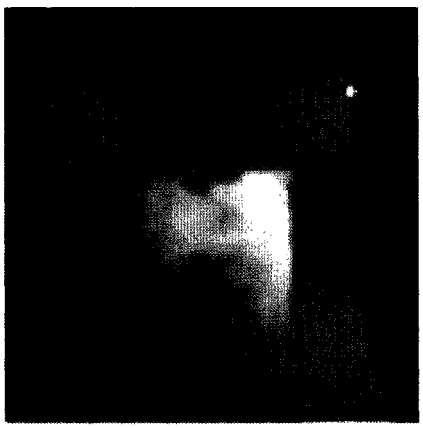


**Figure 2.28 Calcul de l'illumination d'un nuage par radiosité[33]**

[33] Sillion [17]

Les méthodes zonales peuvent engendrer des résultats intéressants, mais requièrent des coûts en calcul très élevés qui ne permettent pas une visualisation fluide. De plus, le découpage en voxels peut créer des problèmes de crénelage (« aliasing ») s'il n'est pas suffisamment fin, ce qui en diminue la performance.

### 2.3.4 Méthode des ordonnées discrètes

La méthode des ordonnées discrètes divise l'espace en voxels et sépare l'espace angulaire en angles solides élémentaires. Max [20] utilise cette méthode pour simuler la diffusion multiple. Un ensemble de M directions discrètes est utilisé pour propager l'intensité lumineuse entre les voxels. Si le volume est divisé en $N=n^3$ voxels, il y a NM intensités à calculer, formant un système d'équations linéaires. Chaque voxel possède une densité uniforme et ses coefficients d'absorption et de diffusion sont constants.

L'algorithme se divise en trois grandes étapes. Premièrement, l'énergie environnante est captée par la surface du volume gazeux. Cette énergie est ensuite diffusée à l'intérieur du milieu. Pour restreindre les artéfacts des rayons, le flux entrant chaque voxel est multiplié par sa transparence et ensuite distribué à quatre voxels adjacents. Les itérations sont effectuées jusqu'à l'atteinte de la convergence. Finalement, l'énergie résultante de la propagation est ré-émise vers la scène.

Encore une fois, plus le découpage en voxels est fin, moins de crénelage sera engendré. Par contre, il faut en payer le prix en performance. La Figure 2.29 a été obtenue en utilisant l'algorithme des ordonnées discrètes.

**Figure 2.29 Simulation de diffusion multiple**

2.3.5 Méthodes de rendu ontogénétiques

Taxen [4] considère que la diffusion simple n'est pas suffisante pour le rendu de nuages. Pour pallier aux temps de calcul trop importants dus à la diffusion multiple, l'auteur propose un rendu ontogénétique. Il divise la couleur en deux catégories :

La couleur du nuage éclairé : $I_c = (R_c, G_c, B_c)$

La couleur du nuage ombragé : $I_s = (R_s, G_s, B_s)$

Ces deux couleurs sont fournies par l'usager. $I_c$ et $I_s$ sont alors combinées avec la couleur de l'arrière-plan.

$I_{bg} = (R_{bg}, G_{bg}, B_{bg})$ :

$I = k_{bg}(o)I_{bg} + k_c(o)I_c + k_s(o)I_s$

où $k_{bg}(o)$, $k_c(o)$ et $k_s(o)$ sont des fonctions de poids de l'opacité

$$o=1-\exp\left(-\kappa\int_a^b \rho(s)ds\right)$$

où $\kappa$ est une constante fournie par l'usager

$a$ et $b$ sont les points d'entrée et de sortie du rayon dans le nuage

$\rho(s)$ est la densité au point s

Les fonctions de poids suivent une spline d'Hermite de la forme :

$$k(t) = (2t^3 - 3t^2 + 1)p_0 + (-2t^3 + 3t^2)p_1 + (t^3 - 2t^2 + t)r_0 + (t^3 - t^2)r_1$$

où $p_0$ et $p_1$ dénotent la valeur de $k(t)$ à $t=0$ et $t=1$

$r_0$ et $r_1$ dénotent la dérivée de $k(t)$ à $t=0$ et $t=1$

Étant donné que les parties basses d'un nuage sont souvent ombragées, $I$ est modifiée par une couleur fournie par l'usager $I_l$.

$$I = (1 - a)I + (a_k * a)I_l$$

$$a=alt^{a_e}$$

où $a_k$ et $a_e$ sont des constantes fournies par l'usager

$alt$ est l'altitude normalisée tel que $alt=0$ au bas du nuage et $alt = 1$ au dessus du nuage.

## 2.3.6 Méthodes hybrides

Plusieurs méthodes pour calculer l'illumination ont été développées et ne peuvent pas être restreintes aux catégories décrites précédemment. La combinaison de plusieurs algorithmes a souvent été utilisée afin d'obtenir un équilibre plus juste entre l'efficacité en calculs et le réalisme du rendu.

Max et Crawfis [3] simulent la diffusion multiple de la lumière du soleil et de la lumière atmosphérique dans un nuage. Des densités de gouttelettes sont d'abord définies dans une grille 3D par une texture 3D provenant d'un bruit de Perlin juxtaposé à trois densités ellipsoïdales. Une fonction de phase de Henyey-Greenstein est utilisée pour spécifier la probabilité directionnelle de diffusion. Pour chaque voxel, le flux total propagé dans 96 directions est comptabilisé. À chaque itération, le flux est propagé aux voxels pouvant être atteints, et ce une couche à la fois. Chaque voxel intercepte une fraction du flux déterminée par la densité de gouttelettes. L'autre fraction est re-diffusée selon la fonction de phase. La propagation est amortie afin que les calculs se limitent à un ordre de $O(N^2)$ plutôt que O(NlogN) (où N est le nombre de voxels). Une fois le flux total déterminé, l'image résultante est produite par lancer de rayon. Un exemple de rendu calculé par cette méthode est présenté à la Figure 2.30.

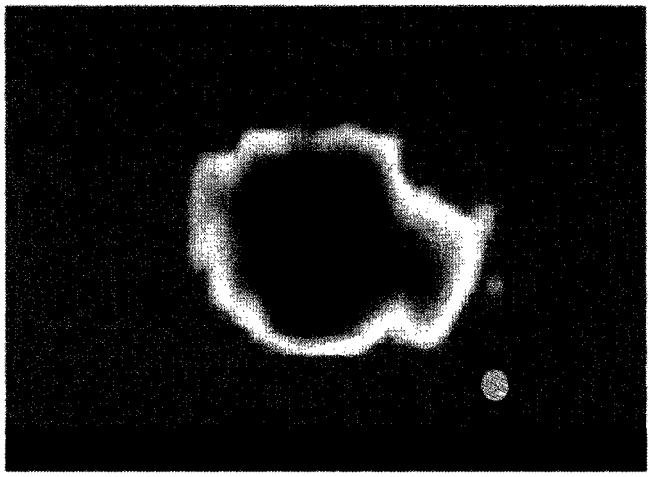


**Figure 2.30 Diffusion multiple obtenue après 15 itérations**[34]

Da Dalto [25][26][27] simplifie les calculs de diffusion multiple en calculant la diffusion entrante pour certains points du médium pour en déduire la valeur des autres points. L'algorithme utilisé présente deux étapes : le processus d'illumination et celui du rendu.

[34] Max et Crawfis [3]

L'illumination du volume gazeux s'effectue par calcul de radiosité. Premièrement, la distance entre la source lumineuse et le gaz permet d'évaluer la perte d'énergie due à la traversée de l'air. Ensuite, si la quantité d'énergie est suffisante, la diffusion entrante pour la première particule frappée par le rayon est calculée. L'auteur assume que toutes les particules voisines de celle sélectionnée possède la même taille et les mêmes propriétés. La diffusion entrante est obtenue par un algorithme de convergence. Initialement, seule l'énergie provenant de la source lumineuse est emmagasinée. Ensuite, la lumière de la particule est diffusée dans certaines directions échantillonnées vers les particules voisines. Ces particules mettront à jour leur fonction de diffusion multiple et les itérations se poursuivront ainsi jusqu'à ce que la diffusion atteigne une valeur minimale. Ces étapes sont effectuées pour chaque source de lumière à certains points de l'enveloppe gazeuse.

Lors du processus de rendu, des rayons sont lancés de l'observateur vers la scène. Si un rayon traverse le volume gazeux, sa couleur sera modifiée en calculant la variation d'énergie. Le rayon traversant le gaz est échantillonné selon des espaces réguliers le long de son parcours. La valeur de chaque propriété du médium (émission, absorption, diffusion entrante et diffusion sortante) est évaluée pour chaque point d'échantillonnage. La règle de Bouguer exprime l'atténuation A entre les points P et P' :

$$A(P, P') = e^{-\tau(P, P')}$$

$$\tau(P,P') = \int_{P}^{P'} K(P'')dP''$$

$$K = \alpha + \sigma$$

où $\tau$ est la profondeur optique entre $P$ et $P'$

$\alpha$ est le coefficient d'absorption

$\sigma$ est le coefficient de diffusion

Pour tenir compte de la diffusion entrante, une interpolation est utilisée pour la valeur de chaque point d'échantillonnage le long du rayon afin d'en additionner la diffusion. Pour chaque point d'échantillonnage, un rayon est lancé vers chaque source de lumière afin de calculer la perte d'énergie et l'émission de lumière jusqu'à ce point. La Figure 2.31 illustre un calcul de diffusion multiple pour un médium gazeux en utilisant la méthode de Da Dalto [25][26][27].

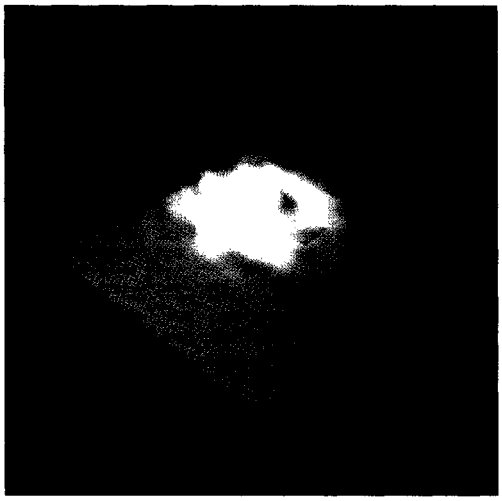


**Figure 2.31 Diffusion multiple de la lumière dans un gaz[35]**

La méthode de Da Dalto [25][26][27] est facile à utiliser et son coût en mémoire est raisonnable. Par contre, l'échantillonnage des rayons lors du calcul de rendu doit être suffisamment fin.

De son côté, Fedkiw [15] reprend la méthode proposée par Stam [12] en simulant le flux de gaz dans une grille 3D. Pour chaque intervalle de temps, la densité et la température de chaque voxel sont calculées, mais en utilisant des équations d'Euler plutôt que des équations de Navier-Stokes afin de diminuer le coût en calcul. Deux méthodes de rendu tenant compte de l'illumination sont proposées.

---

[35] Da Dalto [25][26]

La première, qui permet d'obtenir un temps de calcul plus rapide, utilise un algorithme à deux étapes. La quantité de lumière atteignant chaque voxel de la grille est calculée par un lancer de rayon. Initialement, la transparence de chaque rayon est fixée à 1 ($T_{ray} = 1$). Ensuite, chaque fois qu'un voxel est atteint, la transparence est calculée en fonction de la densité par une relation exponentielle :

$$T_{vox} = exp(\rho * h)$$

où $h$ est l'espacement de la grille

$\rho$ est la densité

Ensuite, la luminosité de chaque voxel est définie, alors que la transparence du rayon est multipliée par la transparence du voxel. La transparence du rayon diminue pendant son parcours de la densité, engendrant ainsi l'effet d'ombrage sur la fumée.

$$L_{vox} = albedo * L_{light} * (1 - T_{vox}) * T_{ray}$$
$$T_{ray} = T_{ray} * T_{vox}$$

La deuxième étape consiste à faire afficher les voxels d'avant en arrière par la carte graphique. La grille est décomposée en une série de plans semi-transparents alignés selon l'axe de direction du regard. La couleur et l'intensité de chaque sommet correspond à $L_{vox}$ et $(1 - T_{vox})$ respectivement.

Le deuxième algorithme, appelé « tracé de photons » tient compte de la diffusion multiple. Il modélise l'interaction des photons avec le médium gazeux. À chaque interaction, le photon est soit diffusé, soit absorbé. Deux étapes de calculs composent cet algorithme. Dans un premier temps, un volume est construit en émettant des photons à travers le médium et en les emmagasinant au fur et à mesure de leur interaction avec ce dernier. Seuls les photons correspondant à l'illumination indirecte sont conservés. Ensuite, dans la phase de rendu, un lancer de rayon est effectué.

$$L_n(x_n,\vec{\omega})=L_{n-1}(x_{n-1},\vec{\omega})+e^{-\tau(x_n)}\Delta x_n(\vec{\omega}*\vec{\nabla})L_s(x_n,\vec{\omega})$$

$$(\vec{\omega}*\vec{\nabla})L_s(x,\vec{\omega})=\Omega\sigma_t(x)\int_{4\pi}L(x,\vec{\omega}_s)p(x_n,\vec{\omega}',\vec{\omega})d\omega'$$

où $\tau(x_n)=\int_{x0}^{x_n}\sigma_t dx$ correspond à la profondeur optique (opacité du médium)

$L_s$ est la fraction de la luminosité diffusée dans la direction $\vec{\omega}$

$\Delta x_n>0$ est le pas, $x_{n+1} = x_n + \Delta x_n$

$x'_n$ est une localisation aléatoire sur le $n^{\text{ième}}$ segment

La luminosité diffusée $L_i$ est divisée en deux termes : la diffusion simple ($L_d$) et la diffusion multiple ($L_m$). La diffusion simple est calculée par lancer de rayon. La diffusion multiple, quant à elle, est calculée à partir du volume de luminosité estimé par tracé de photons (première étape) en localisant les $n_p$ photons voisins :

$$(\vec{\omega}\cdot\vec{\nabla})L_m(x,\vec{\omega})=\sum_{1}^{n_p}\frac{\phi_p(\vec{\omega}')p(x,\vec{\omega}',\vec{\omega})}{\frac{4}{3}\pi r^3}$$

où $\phi_p$ est la puissance du $p^{\text{ième}}$ photon

$r$ est la plus petite sphère englobant les $n_p$ photons

La fumée illustrée à la Figure 2.32 a été obtenue en utilisant l'algorithme de tracé de photons.

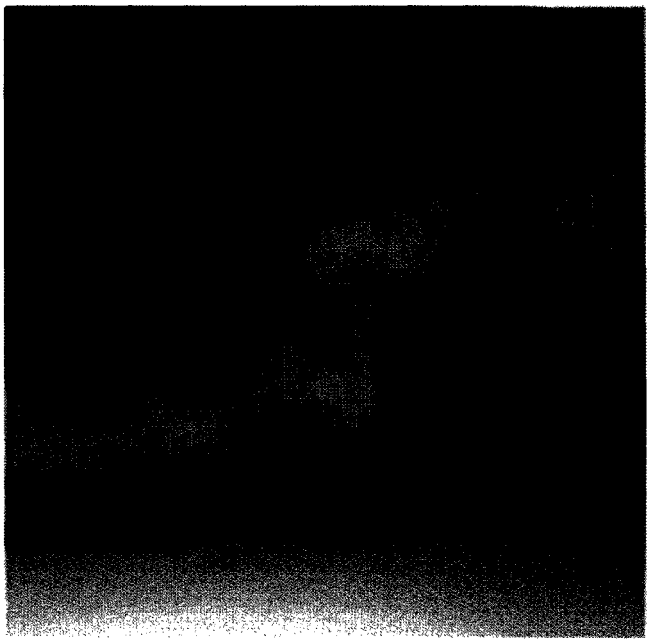


**Figure 2.32 Illumination de fumée par tracé de photons**[36]

Nishita [19] propose un algorithme pour calculer la diffusion multiple dans un nuage. Premièrement, la grille 3D contenant le champ de densité est parcourue pour calculer l'intensité de la lumière diffusée à chaque voxel par les autres voxels (premier ordre). Seuls les voxels dont la contribution est supérieure à un seuil minimal sont considérés. La fonction de phase utilisée provient d'une fonction d'Henyey-Greenstein :

$$F(\theta,g)=\frac{3(1-g^2)(1+\cos^2\theta)}{2(2+g^2)(1+g^2-2g\cos\theta)^{3/2}}$$

où $g$ est un facteur d'asymétrie déterminé par le type de nuage

Le ratio d'atténuation est ensuite enregistré dans chaque voxel. Pour chacun d'eux, la lumière diffusée dans la direction de l'observateur (deuxième et troisième ordre) est calculée en tenant compte de l'atténuation. Finalement, pour chaque pixel, l'intensité est obtenue en intégrant linéairement la grille. L'intensité de la diffusion de premier ordre

[36] Fedkiw [15]

pour un point sur le rayon de la vue est obtenue en utilisant le ratio d'atténuation et les intensités de deuxième et troisième ordre sont interpolées. La Figure 2.33 montre un exemple de résultat obtenu par Nishita [19].

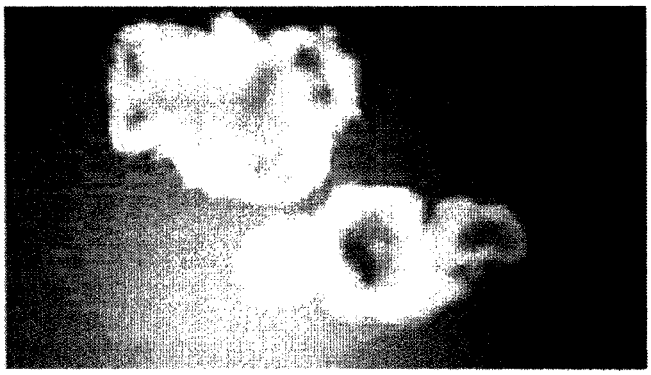


**Figure 2.33 Diffusion multiple (3 ordres) dans des nuages**[37]

[37] Nishita et al. [19]

# CHAPITRE 3 - SOLUTION ADOPTÉE

## 3.1 Objectifs visés

Pour que notre modélisation soit considérée comme étant adéquate, elle devra répondre à trois critères principaux : le réalisme du rendu, l'exactitude des résultats et l'interactivité.

Le réalisme du rendu doit permettre à l'observateur de croire à la scène visualisée. Que le phénomène étudié soit de la fumée ou un nuage, l'image affichée devrait être semblable au phénomène se déroulant normalement dans un milieu réel.

Tel que mentionné dans la section 1.4, les simulations numériques effectuées par FDS fournissent la densité et la composition chimique pour une discrétisation de l'espace. Notre modélisation doit être le reflet des données numériques et afficher une image qui corresponde exactement à ces résultats.

L'interactivité entre l'observateur et la modélisation est un critère important dans un contexte de visualisation scientifique. L'usager doit pouvoir se déplacer autour du phénomène observé, le faire pivoter et s'en rapprocher arbitrairement dans le but d'étudier les tendances de la simulation et d'effectuer les vérifications désirées. La modélisation doit donc pouvoir être générée de façon rapide en tenant compte des ressources informatiques actuelles. De même, l'affichage de l'image doit pouvoir être effectué à une fréquence suffisante pour permettre une fluidité lors de l'interaction. Idéalement, l'évolution du phénomène dans le temps devrait également pouvoir être tenue en compte.

### 3.2 Retour sur la revue de littérature

Il est clair que certains compromis devront être établis entre les trois critères précédents. En effet, certaines modélisations permettant un rendu très semblable à la réalité ne respectent aucunement l'exactitude des données, puisqu'elles utilisent des variables aléatoires. De même, calculer avec précision l'illumination d'un gaz nécessiterait des temps de calcul beaucoup trop élevés qui ne permettraient pas d'interaction fluide de la part de l'usager. Dans le but de mieux définir les avenues possibles, nous allons reprendre brièvement les techniques de modélisation présentées dans le chapitre précédent en départageant celles qui pourraient satisfaire nos objectifs.

Champs vectoriels : Cette technique doit être rejetée, puisqu'elle ne présente aucun réalisme du rendu.

Surfaces iso-densité : Cette modélisation n'offre pas assez d'exactitude par rapport aux données numériques, car il y a perte d'information par rapport aux densités non affichées.

Méthode des nuées : Cette méthode pourraient convenir à nos besoins, à condition de porter une attention spéciale à la performance.

Lancer de rayon : Le lancer de rayon est la méthode offrant le meilleur rendu. Par contre, compte tenu des ressources informatiques actuelles, elle n'est pas suffisamment rapide pour permettre une interaction adéquate.

Système de particules : Cette méthode pourrait être envisageable, mais elle présente deux faiblesses majeures : les particules restent visibles si on les regarde de près, et le nombre requis de particules pour

obtenir une image réaliste doit être très grand, ce qui limite la performance.

Hypertextures : Considérant que les cartes graphiques récentes sont optimisées pour le calcul de textures, cette avenue semble toute adéquate à condition que les textures correspondent bien aux données numériques.

Méthode des ellipsoïdes : Jusqu'à présent, cette méthode est associée à l'utilisation de variables aléatoires pour le calcul de rendu, ce qui demeure pour nous inacceptable au niveau de l'exactitude des résultats.

Méthodes de projection : Semblables aux méthodes utilisant les textures, ce type de modélisation pourrait s'avérer efficace. Par contre, elle perd un peu de sa raison d'être dans une situation où l'on désire visualiser le phénomène sous différents angles.

Au niveau du calcul de l'illumination, il est clair que les phénomènes d'absorption et de diffusion doivent être considérés pour obtenir le réalisme désiré. Par contre, la diffusion multiple s'avère être un obstacle majeur, puisque son calcul ne peut être réalisé qu'à un coût élevé en ressources informatiques. Certains compromis devront donc être envisagés.

Le Tableau 3.1 dresse un sommaire des différents types de méthodes à des fins comparatives. Pour chacune d'elles, nous avons évalué leur capacité de répondre aux critères de réalisme, d'exactitude et d'interactivité selon une échelle subjective : mauvais, moyen ou bon.

**Tableau 3.1 Évaluation des techniques de modélisation**

| Modélisation | Réalisme | Exactitude | Interactivité |
|---|---|---|---|
| **Champs vectoriels** | **Faible** | **Bonne** | **Bonne** |
| **Surfaces iso-densité** | **Moyen** | **Faible** | **Bonne** |
| Max et Crawfis [3] | | | |
| **Méthode des nuées** | **Bon** | **Bonne** | **Moyenne** |
| Stam [16] | Moyen | Faible | Faible |
| Stam et Fiume [10][11] | Bon | Bonne | Bonne |
| Stam [14] | Bon | Bonne | Faible |
| Barrero [28] | Bon | Bonne | Faible |
| **Lancer de rayon** | **Bon** | **Bonne** | **Faible** |
| **Système de particules** | **Moyen** | **Bonne** | **Moyenne** |
| **Hypertextures** | **Bon** | **Moyenne** | **Bonne** |
| King [7] | Bon | Faible | Bonne |
| Ebert [22][23] | Bon | Faible | Bonne |
| Heidrich [8] | Bon | Faible | Bonne |
| Stam [12] | Bon | Faible | Bonne |
| **Méthode des ellipsoïdes** | **Moyen** | **Faible** | **Moyenne** |
| Gardner [1] | Moyen | Faible | Bonne |
| Elinas et Stuerzlinger [2] | Moyen | Faible | Bonne |
| Taxen [4] | Moyen | Faible | Bonne |
| Rasmussen [18] | Bon | Faible | Faible |
| **Méthodes de projection** | **Bon** | **Moyenne** | **Moyenne** |
| Dobashi [5] | Bon | Moyenne | Moyenne |
| Heinzlreiter [6] | Moyen | Faible | Faible |
| Harris et Lastra [9] | Bon | Moyenne | Bonne |

La méthode des nuées et celle des hypertextures constituent les modélisations qui satisfont le plus à nos critères d'évaluation. La différence se situe surtout au niveau de l'interactivité et du réalisme. La plupart des auteurs ayant utilisé la méthode des nuées font appel à une procédure de lancer de rayon pour obtenir le rendu final. Toutefois, il

serait envisageable d'utiliser OpenGL ou DirectX afin d'obtenir une meilleure interactivité. En ce qui concerne les hypertextures, les auteurs génèrent leurs textures par des méthodes stochastiques. Si elles étaient conçues à partir des données numériques, cette lacune s'en trouverait remédiée.

### 3.3 Hypothèses soulevées

Pour ce travail, nous nous sommes fixé comme objectif de déterminer une méthode de modélisation pouvant s'avérer être un compromis idéal entre la performance et le réalisme. Nous assumons qu'il n'est pas essentiel que le calcul de la diffusion multiple soit exact pour que les besoins de la visualisation soient comblés. Plus encore, nous prétendons qu'il est possible de simplifier le processus de diffusion en des calculs d'approximation. Nous supposons également que la modélisation doit pouvoir être construite à l'aide de ressources informatiques à la portée du grand public.

Nous avons opté pour l'utilisation d'hypertextures comme structure de notre modélisation. Nous croyons que cette avenue, compte tenu des développements au niveau des cartes graphiques, s'avère être un moyen simple et flexible permettant de visualiser différents types de gaz. Nous posons comme hypothèse que tous les types de phénomènes gazeux peuvent être modélisés par une procédure commune utilisant les textures 3D, et que la différence au niveau du rendu peut être prise en compte uniquement par l'utilisation de quelques variables.

Nous allons décrire notre modélisation permettant d'effectuer les compromis visés, puis nous présenterons différents résultats que nous avons obtenus. Finalement, nous serons plus en mesure de critiquer cette modélisation en la comparant avec d'autres applications de visualisation.

### 3.4 Structure de la modélisation

Notre modélisation utilise les textures 3D d'OpenGL comme structure de base. Étant donné qu'au niveau des données numériques l'espace est divisé en petits hexaèdres, cette structure convient particulièrement bien, puisque les textures 3D possèdent une division orthogonale 3D. La taille de la texture est donc reliée au nombre de divisions du domaine. Pour chaque voxel (unité de volume) de la texture, des valeurs d'intensité (GL_LUMINANCE) et d'opacité (GL_ALPHA) sont déterminées à partir des données de densité. Finalement, tel qu'illustré à la Figure 3.1, une série de plans orthogonaux à la direction du regard de l'observateur est créée sur laquelle la texture 3D est appliquée.

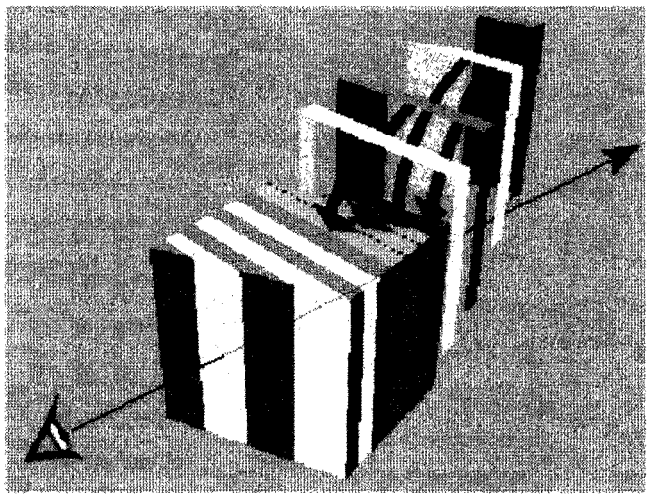


**Figure 3.1 Application d'une texture 3D sur des plans orthogonaux**

Chaque polygone reçoit une valeur de texture correspondant aux données du volume. Les rotations et les interpolations tri-linéaires sont calculées automatiquement par OpenGL. Il est à noter que la performance dépend du nombre de plans utilisés. Un nombre élevé de plans engendrera plus de temps de calcul, mais générera un raffinement plus précis au niveau de l'interpolation de la texture. Il est donc possible de faire varier ce paramètre en fonction des besoins.

Lors d'un changement de direction de la caméra, les plans doivent être pivotés afin de demeurer orthogonaux à la direction du regard. Pour conserver la position du gaz dans la scène, une rotation égale mais de sens opposé doit être appliquée au niveau de la texture. Les matrices de transformation d'OpenGL permettent d'effectuer ces opérations facilement. Finalement, il est à noter que la taille des plans doit être supérieure à celle du domaine afin que la texture n'excède pas à l'extérieur lors des rotations.

Nous avons mentionné que la texture 3D était composée de deux paramètres : l'intensité et l'opacité. Les deux sections suivantes expliquent comment ces valeurs sont calculées à partir des données numériques.

### 3.5 Calcul de la transparence

Nous avons essayé trois différents types de relation entre les valeurs de densité et celles de transparence pour déterminer laquelle offre le rendu le plus réaliste : la relation linéaire, la relation exponentielle et la relation logarithmique. Une image pour laquelle une relation linéaire a été utilisée est illustrée à la Figure 3.2, tandis que la Figure 3.3 représente une relation logarithmique. On peut voir que toutes deux offrent un rendu similaire relativement uniforme, la fumée obtenue par relation logarithmique étant toutefois légèrement plus opaque. On observe une plus grande variation d'opacité lorsque la relation exponentielle est utilisée, comme le montre la Figure 3.3. Ainsi, la relation exponentielle nous a donné des résultats plus convaincants que les deux autres, car elle permet d'accentuer d'avantage les valeurs de densité supérieures, laissant plus de transparence aux valeurs inférieures. D'ailleurs, la relation exponentielle était celle préconisée par de nombreux auteurs (Stam [13][16], Ebert [22][23], Rasmussen [18], Taxen [4], Fedkiw [15]).

**Figure 3.2 Relation linéaire d'opacité**

**Figure 3.3 Relation logarithmique d'opacité**

**Figure 3.4 Relation exponentielle d'opacité**

Pour calculer la transparence à partir des valeurs de densité, l'usager doit spécifier des seuils de densité minimale et maximale. Le rapport entre chaque valeur de densité (ρ) et les seuils (*ValMin* et *ValMax*) est transposé dans l'intervalle [0, 1] par une relation directe.

$$\rho = \frac{\rho - ValMin}{ValMax - ValMin}$$

Il arrive pour certaines simulations que les valeurs de densité se retrouvent massivement trop près d'une des bornes de l'intervalle. Il est alors préférable de les décaler de façon

arbitraire le long de la courbe exponentielle pour obtenir un meilleur rendu (Figure 3.5). Malheureusement, le décalage nécessaire peut varier beaucoup d'une simulation à une autre et est fonction du type de phénomène gazeux à générer. Par exemple, un rendu de brouillard correspondra à des valeurs de densité près du seuil minimal, tandis qu'une épaisse fumée, au contraire, pourra posséder des valeurs de densité élevées. De plus, le niveau de dispersion des valeurs de densité à l'intérieur de l'intervalle [*ValMin*, *ValMax*] peut limiter la possibilité de décalage. Si la répartition possède une allure gaussienne, il est plus facile de déplacer les valeurs le long de la courbe. Autrement, si les valeurs sont très dispersées sur la courbe, le jeu de décalage s'en trouve réduit. Compte tenu de ces paramètres difficilement contrôlables, nous n'avons pas automatisé le décalage des valeurs de densité, laissant à l'usager la responsabilité de le spécifier en fonction du rendu désiré.

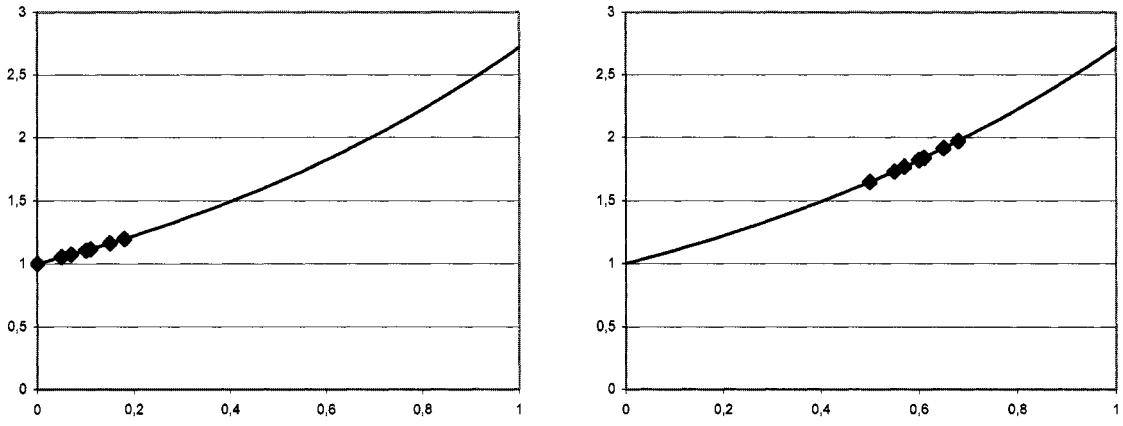



**Figure 3.5 Exemple de décalage des valeurs de densité**

Le calcul de l'opacité est effectué sous forme de rapports de valeurs exponentielles afin d'obtenir un résultat situé entre [0,1].

$$Opacité = \frac{\exp(\rho) - \exp(0)}{\exp(1) - \exp(0)} = \frac{\exp(\rho) - 1}{1{,}718}$$

Afin que l'opacité totale du gaz ne soit pas affectée par le nombre de plans utilisés par la texture, nous divisons l'opacité par le rapport entre le nombre de plans et la profondeur du gaz. L'opacité est multipliée par un facteur de 255 afin d'obtenir une opacité finale située dans l'intervalle [0,255].

$$Opacité = \frac{Opacité * 255,0}{(float)NombrePlans/(float)Dim_z}$$

Une fois l'opacité calculée, il est également facile d'en déduire la transparence située dans l'intervalle [0,255].

$$Transparence = 255,0 - Opacité$$

Le calcul de la transparence est donc simple et rapide, mais doit être effectué pour chaque valeur de densité située à l'intérieur de l'intervalle désiré, les valeurs à l'extérieur de l'intervalle étant considérées comme nulles. Outre cet intervalle, une seconde contrainte doit être satisfaite : la composition chimique (Z). La composition chimique indique la proportion de gaz pour chaque coordonnée de l'espace. Une composition chimique nulle correspond à de l'air pur, tandis qu'une valeur de 1 indique du carburant pur. Un mélange stoechiométrique correspondra à l'affichage d'une flamme. Habituellement, cette valeur est fixée à Z=0,147. Étant donné que nous nous préoccupons uniquement du rendu de gaz et non pas de celui du feu, seules les valeurs situées dans l'intervalle $0<Z<0,147$ recevront une valeur d'opacité non nulle.

### 3.6 Calcul de la luminosité

Calculer l'illumination multiple réelle ne permettrait pas d'atteindre la rapidité de calcul souhaitée. D'autre part, la diffusion simple ne permet pas de rendu suffisamment réaliste pour la modélisation de gaz doté d'un albedo élevé. C'est pourquoi nous utilisons un algorithme qui effectue un compromis entre la performance et le réalisme. La méthode est inspirée de celle proposée par Harris et Lastra [9] et décrite à la section 2.3.1.

L'algorithme que nous proposons pour déterminer l'intensité du gaz utilise deux étapes de calculs. Dans un premier temps, nous effectuons une itération de la propagation de la lumière provenant de la source de lumière principale à travers le medium gazeux. Pour y parvenir, un lancer de rayon passant par chacun des voxels de la grille de la texture 3D est effectué et la position d'intersection sur le plan de voxel suivant est calculée (Figure 3.6). Il est à noter que la texture 3D doit déjà contenir les valeurs de transparence (GL_ALPHA). Les premiers voxels interceptés par les rayons reçoivent l'intensité maximale de 255 (lumière blanche). On s'éloigne graduellement jusqu'à ce que toute la texture soit parcourue. Chaque voxel reçoit comme valeur de luminosité la quantité de lumière qu'a laissé passer le voxel précédent suivant le vecteur direction de la lumière.

$$L_i = L_{i-1} * Alpha_{i-1}$$

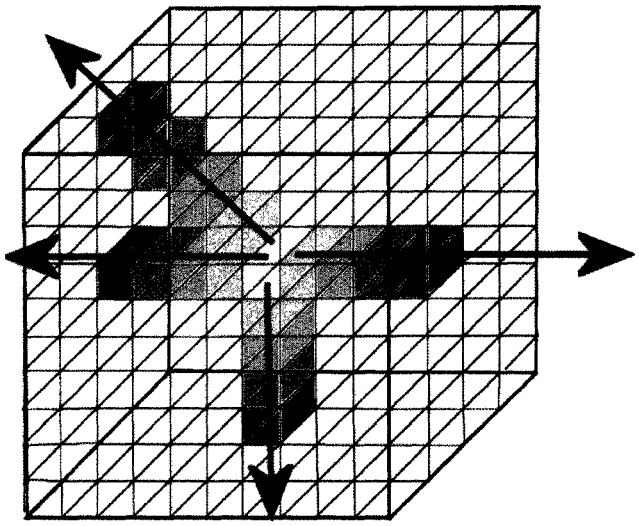


**Figure 3.6 Lancer de rayon pour calculer l'illumination**

La seconde phase de l'algorithme calcule l'illumination diffuse dans la direction des rayons de vue. On tient compte uniquement de la diffusion vers l'observateur, comme l'illustre la Figure 3.7.

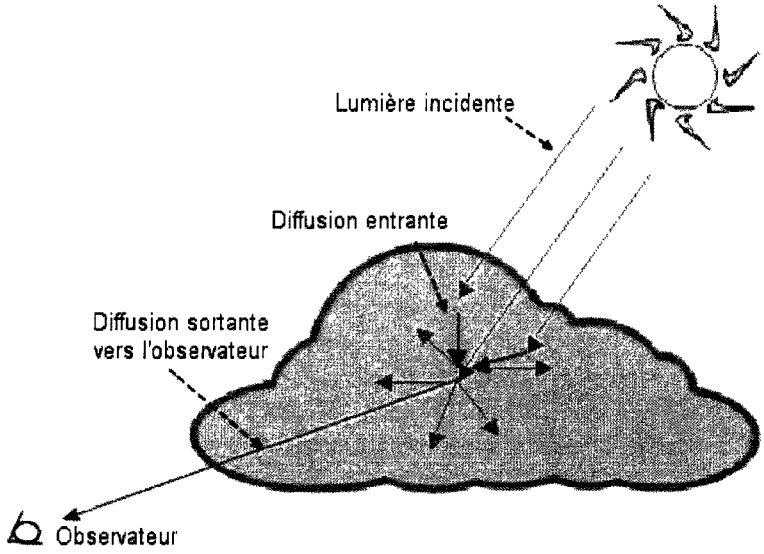



**Figure 3.7 Diffusion de la lumière vers l'observateur**

Pour chaque voxel de la grille 3D, l'intensité lumineuse issue de la propagation de la lumière (obtenue par la première étape de l'algorithme) est multipliée par la fonction de phase et par la constante de l'albedo. Nous avons vu dans la section 1.5.3 que le type de fonction de phase à utiliser était lié à la grosseur des particules du gaz. Pour de petites particules, le régime de Rayleigh était prépondérant, tandis que pour des particules plus volumineuses, le régime de Mie était plus adéquat. Ainsi, le choix de la fonction de phase devrait normalement être dépendante du type de phénomène modélisé. Toutefois, Harris et Lastra [9] ont montré qu'il était possible d'utiliser une approximation de la règle de Rayleigh plutôt que la fonction de phase de Mie sans que le résultat visuel en soit grandement affecté (voir section 2.3.1). Étant donné que l'utilisation de la fonction de phase de Mie ne satisferait pas nos objectifs de performance au niveau du temps de calcul et dans le but d'utiliser une procédure commune pour tous les types de phénomènes gazeux, nous avons adopté le calcul de fonction de phase proposé par Harris et Lastra [9]. Ce calcul est très facile d'implémentation et suffisamment rapide pour être répété pour chaque coordonnée de la grille 3D contenant une valeur de densité non nulle.

$$L_i = L_i * \tfrac{3}{4} (1 + \cos^2(\theta)) * Albedo$$

où $\theta$ est l'angle entre la direction incidente et la direction vers l'observateur

$L_i$ est la luminosité du voxel i

## 3.7 Implémentation de l'algorithme de rendu

Cette section décrira plus en détail la procédure adoptée telle qu'implémentée dans notre outil de visualisation. Nous traiterons uniquement des algorithmes utilisés au niveau du rendu du phénomène gazeux. La description de l'interface et du design de l'application fera l'objet de la section suivante.

Pour chaque fichier de simulation numérique correspondant à l'état du phénomène gazeux pour un temps déterminé, le traitement à effectuer se divise en deux parties. La première consiste à initialiser la structure de données, alors que la seconde calcule le rendu final.

### 3.7.1 Initialisation de la structure de données

Pour que l'initialisation de la structure de données puisse être effectuée, différents paramètres doivent être fournis par l'usager :

- Nom du fichier de données;
- Position de début de lecture dans le fichier pour les valeurs de densité;
- Position de début de lecture dans le fichier pour les valeurs de composition chimique;
- Nombre de discrétisations en x, y et z de l'espace 3D ($n_i$, $n_j$, $n_k$);
- Dimensions de l'espace selon les axes x, y et z;
- Nombre de plans à utiliser pour la texture 3D;
- Seuils de densité minimal et maximal;
- Valeur maximale de Z (composition chimique);
- Position de la source de lumière principale;
- Couleur de la lumière incidente;
- Valeur de décalage des données de densité;
- Valeur d'albedo du gaz.

Une fois ces valeurs fixées, il est possible de créer la texture 3D modélisant le phénomène gazeux. Cette texture est composée d'une intensité et d'une transparence. La transparence est indépendante de l'éclairage et de la position de l'observateur, car elle est calculée uniquement à partir de la valeur de densité. Elle est donc évaluée une

seule fois, lors de la lecture du fichier contenant les valeurs de densité. Toutefois, il faut également tenir compte de la composition chimique. Si la valeur Z ne se situe pas dans l'intervalle d'acceptation $]0,Z_{max}]$, l'opacité est fixée à 0.

**Algorithme 1 – Calcul de la transparence**

```
Calculer la plus petite puissance de 2 supérieure à la diagonale de l'espace subdivisé
Créer la texture avec les dimensions calculées
        ImageText = new GLubyte[TailleTex_x*TailleTex_y*TailleTex_z*2]
Créer un tableau 3D pour les valeurs de transparence (nombres réels)
Initialiser le tableau de transparence à 0
Ouvrir le fichier de données
Se positionner au début des données de composition chimique
Tant que les valeurs de composition chimique ne sont pas toutes lues
        Lire la valeur de Z
        Convertir Z en petit boutiste ou grand boutiste si nécessaire
        Si 0 < Z =< Zmax
                Mettre la valeur de transparence à 1.0
        Fin Si
Fin Tant
Se positionner au début des données de densité
Tant que les valeurs de densité ne sont pas toutes lues
        Lire la valeur de densité ρ
        Convertir ρ en petit boutiste ou grand boutiste si nécessaire
        Si ValeurMin <= ρ <= ValeurMax
                Situer ρ entre [0,1]
                Décaler ρ dans l'intervalle
                // Calcul de l'opacité selon la relation choisie par l'interface
                Si on veut une relation linéaire
```

$Opacité = \rho$

**Sinon si** on veut une relation exponentielle (intervalle [1,6])

$$Opacité = \frac{\exp(\rho * 5{,}0+1{,}0)-\exp(1{,}0)}{\exp(6{,}0)-\exp(1{,}0)}$$

**Sinon si** on veut une relation logarithmique (intervalle [1,6])

$$Opacité = \frac{\log(\rho * 5{,}0+1{,}0)}{\log(6{,}0)}$$

**Fin Si**

Multiplier l'opacité par la valeur de transparence déjà calculée (0 ou 1)

Calculer l'opacité en fonction du nombre de plans utilisés

$$Opacité = \frac{Opacité * 255{,}0}{(float)NombreTranches/(float)Dim_z}$$

Écrire la valeur d'opacité dans le tableau de transparence

**Sinon** mettre la valeur de transparence à 0

**Fin Tant que**

Fermer le fichier de données

3.7.2 Calcul du rendu final

Une fois le paramètre de transparence déterminé, il est possible de calculer l'intensité par les deux phases de calculs de l'illumination présentées précédemment. Le calcul de la propagation de la lumière dans le gaz dépend de la position de la source de lumière par rapport au médium. Il doit donc être effectué chaque fois qu'il y a déplacement d'un de ces deux éléments. Par contre, il n'est pas affecté par un déplacement de l'observateur. L'algorithme utilisé afin de simuler la propagation du flux lumineux à l'intérieur du gaz est décrit ci-dessous.

**Algorithme 2 – Calcul de la propagation de la lumière**

```
Initialiser la texture 3D avec les valeurs de transparence et une intensité de 255,0
Trouver la position de la lumière
Déterminer les huit sommets du plus petit cube autour de la lumière
Tant que toute la texture 3D n'est pas parcourue
    Pour chacune des faces du cube entourant la source de lumière
        Si il y a encore un plan d'intersection possible dans cette direction
            Si on a atteint une zone occupée par la texture
                Pour chaque voxel de la face (vf)
                    Calculer la direction du rayon de la source vers le voxel vf
                    Calculer le point d'intersection vi sur le plan un pas plus loin
                    Si le voxel vi est situé dans une zone valide
                        Déterminer la coordonnée de vi dans la texture 3D
                        Calculer la lumière atteignant le nouveau voxel
                            Luminosité de vi = Luminosité de vf * transparence de vf
                        Écrire l'intensité calculée dans la texture
                    Fin Si
                Fin Pour
            Fin Si
        Fin Si
        Déplacer les coins de la face traitée un pas de distance plus loin de la source
    Fin Pour
Fin Tant que
```

La seconde partie du calcul de l'intensité considère la diffusion simple en direction de l'observateur. Ceci implique que l'intensité doit être recalculée à chaque déplacement et rotation de la caméra par rapport à la scène, et ce pour chaque point de la texture. Ce traitement engendre un coût en calculs trop élevé pour être implémenté en langage C ou par OpenGL. Par contre, en utilisant directement les nuanceurs de sommets et de pixels,

il est possible d'obtenir une bonne performance. Ainsi, la seconde phase du calcul de l'illumination a été implémentée en utilisant le langage Cg afin de profiter de la performance du GPU. Le facteur de dispersion est calculé au niveau du nuanceur de sommets et l'intensité finale de la lumière atteignant l'observateur est obtenue au niveau du nuanceur de pixels.

Le nuanceur de sommets reçoit en entrée les matrices de transformation, la position de la source de lumière et celle de la caméra, la constante d'albedo du gaz ainsi que la couleur de la lumière incidente et celle du gaz. Il retourne la position des sommets et des coordonnées de texture après l'application des matrices de transformation ainsi que la couleur des pixels selon l'intensité de dispersion de la lumière dans la direction de l'observateur.

**Algorithme 3 – Opérations effectuées par le nuanceur de sommets**

Calculer la coordonnée de texture après l'application des transformations

Out.TexCoord0 = mul(TextureMatrix, In.Texture0);

Calculer la coordonnée du sommet après l'application des transformations

Out.Position = mul(modelViewProjection, In.Position);

Calculer la position de la lumière après l'application des transformations

PositionLum = mul(modelViewProjection, PositionLum);

Calculer la position de la caméra après l'application des transformations

PositionVue = mul(modelViewProjection, PositionVue);

Calculer l'angle entre la direction de la lumière incidente et celle de la caméra

Calculer l'approximation de la fonction de phase de Rayleigh

$\Omega = \frac{3}{4} (1 + \cos^2(\theta)) * Albedo$

Calculer le facteur multiplicatif de la couleur du sommet tenant compte de la dispersion

CouleurSortie = CouleurLumière * CouleurGaz * Ω

Le traitement effectué au niveau du nuanceur de pixels est simple. À partir de la texture 3D contenant la transparence et l'intensité de la lumière issue d'une itération de propagation (première phase de calculs), la couleur finale du pixel est calculée. La couleur RGBA issue de la texture est simplement multipliée par la couleur de sortie du nuanceur de sommets. Le code suivant est effectué par le nuanceur de pixels :

**Algorithme 4 – Code effectué par le nuanceur de pixels**

```
struct inputs
{
   float4 Texture0 : TEXCOORD0;
        float4 Couleur : COLOR0;
};

struct outputs
{
   float4 Couleur    : COLOR0;
};

outputs main(inputs In, uniform sampler3D TextureNuage )
{
 outputs Out;

 // calcule la couleur du pixel selon la texture
 Out.Couleur.rgba = tex3D(TextureNuage, In.Texture0 ).rgba;

 // multiplie par facteur obtenu du nuanceur de sommets
 Out.Couleur.rgba = In.Couleur * Out.Couleur;

 return Out;
}
```

Afin que le gaz soit toujours bien visible peu importe la position et l'orientation de la caméra, il est essentiel que les plans sur lesquels est apposée la texture 3D soient toujours placés de façon orthogonale à la direction de la caméra. Ainsi, à chaque rotation de la part de l'observateur, une rotation doit également être appliquée au niveau de ces plans. De façon conséquente, une rotation égale mais dans le sens opposé doit

être effectuée au niveau de la texture 3D. Finalement, la série de plans doit également être soumise à autre transformation. En effet, la texture 3D possède comme dimensions le pas d'échantillonnage de l'espace, ce qui ne correspond pas nécessairement aux dimensions du volume. Une mise à l'échelle doit donc être appliquée aux plans pour respecter les bonnes proportions. Ces transformations sont effectuées par des instructions OpenGL lors de la procédure d'affichage du gaz.

**Algorithme 5 – Affichage des plans pour la texture 3D**

```
// en mode de transformation GL_MODELVIEW
Appliquer la mise à l'échelle DimensionsEspace / DimensionsTexture
Se placer au centre du cube formé par la texture 3D
Calculer les angles de rotation selon les axes y et z
Effectuer les rotations
Se replacer à l'origine
// en mode de transformation GL_TEXTURE
Se déplacer au centre de la texture
Appliquer les rotations égales, mais de sens contraire à celles des plans
Se replacer à l'origine
// en mode de transformation GL_MODELVIEW
Activer les programmes de Cg pour les nuanceurs de sommets et de pixels
Établir les matrices et les autres paramètres d'entrée pour les programmes de Cg
Afficher la série de plans
```

### 3.8 Interface et design du logiciel de visualisation

Notre logiciel de visualisation a été implémenté en langage C++ sous Linux avec OpenGL et Cg comme librairies graphiques. Il pourrait toutefois être facilement supporté par Windows, qui permet également l'utilisation de ces librairies. De plus, une interface personne-machine a été construite afin de permettre à l'usager de contrôler la visualisation du phénomène gazeux. Il est à noter que le design de l'ensemble de l'application est grandement indépendant des algorithmes utilisés au niveau rendu et décrits dans les sections précédentes. Ainsi, la structure du programme ainsi que la présentation graphique auraient pu être bien différentes de celles que nous avons adoptées. Nous allons néanmoins la décrire afin de démontrer l'interactivité et la souplesse d'utilisation permises par notre modélisation.

Premièrement, nous allons exposer les principales fonctionnalités offertes à l'usager par l'entremise de notre interface personne-machine. Cette interface à été conçue en utilisant la librairie Qt. Cette dernière produit de belles présentations graphiques, est facile d'utilisation et peut être utilisée en combinaison avec le langage C++. De plus, elle est compatible avec un grand nombre de systèmes d'exploitation dont Linux et Windows. La section suivante traitera du design de l'application. Nous nous limiterons à une description générale de la structure du programme sans entrer dans les détails d'implémentation.

### 3.8.1 Fonctionnalités de l'outil de visualisation

❖ **Évolution de la visualisation dans le temps**

Un fichier de simulation contient les informations de l'état du phénomène gazeux pour un instant précis. Or, il peut s'avérer avantageux d'être capable de visualiser l'évolution du phénomène dans le temps. Ainsi, il est possible de sélectionner une liste de fichiers de simulation qui sera parcourue afin de permettre une animation automatique de l'état du gaz. La Figure 3.8 illustre une vue de l'application où une liste de trois fichiers de simulation est sélectionnée.

La modélisation proposée permet d'obtenir un résultat suffisamment rapide pour pouvoir afficher facilement un état de simulation par seconde sur un ordinateur disposant d'un processeur et d'une carte graphique commune de nos jours. En ce qui nous concerne, nous avons effectué nos visualisations sur une station AMD Athlon XP1600+ disposant d'une carte graphique NVIDIA GeForce3. L'usager peut spécifier la fréquence d'affichage désirée selon ses besoins et des ressources informatiques à sa disposition.

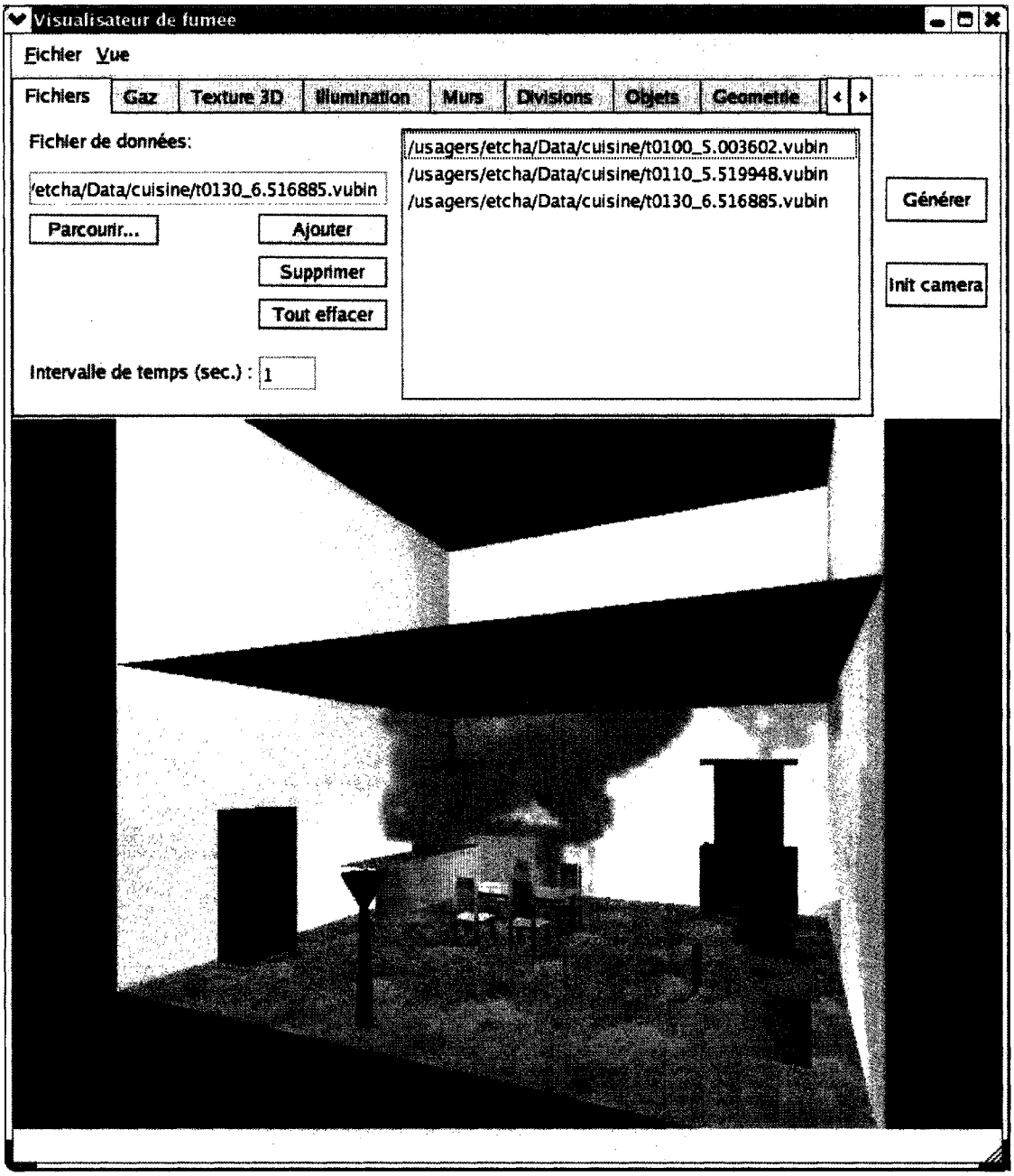



**Figure 3.8 Sélection de plusieurs fichiers de simulation**

### ❖ Contrôle des paramètres

Un onglet permet à l'usager de spécifier les endroits de début de lecture pour les valeurs de densité de gaz et de composition chimique à l'intérieur des fichiers de données. Il permet également de choisir des valeurs de densité minimale et maximale ainsi que de $Z_{max}$. Cette vue est illustrée à la Figure 3.9.

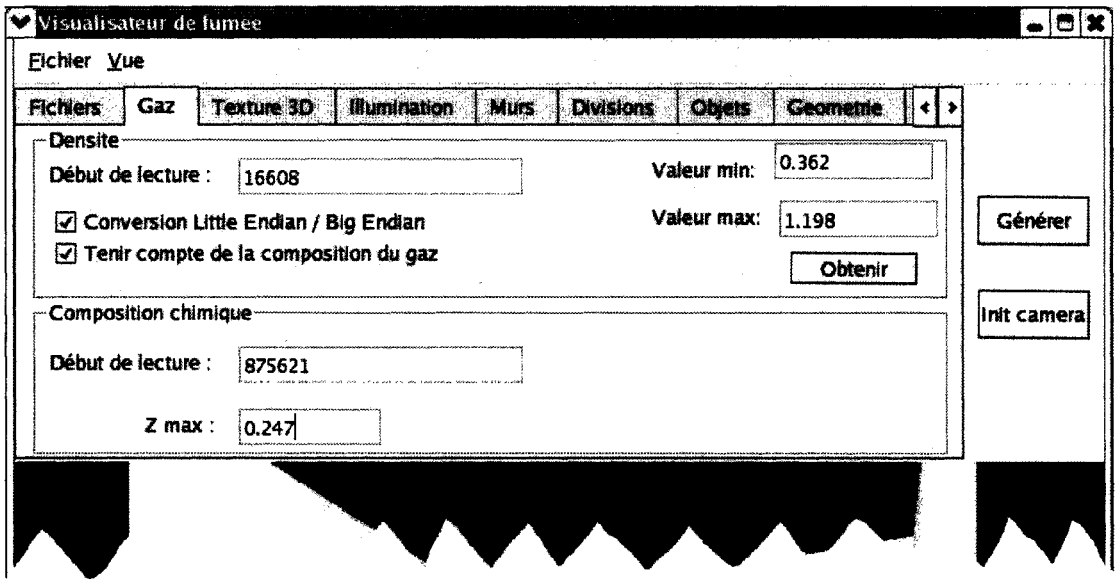



**Figure 3.9 Spécification des paramètres de lecture des fichiers**

Un onglet permet de définir les paramètres à utiliser lors de la construction de la texture 3D et de son affichage (Figure 3.10). Ces valeurs contrôlent l'aspect du gaz selon le type de phénomène à observer. Elles influencent le degré de transparence et d'intensité de la texture.

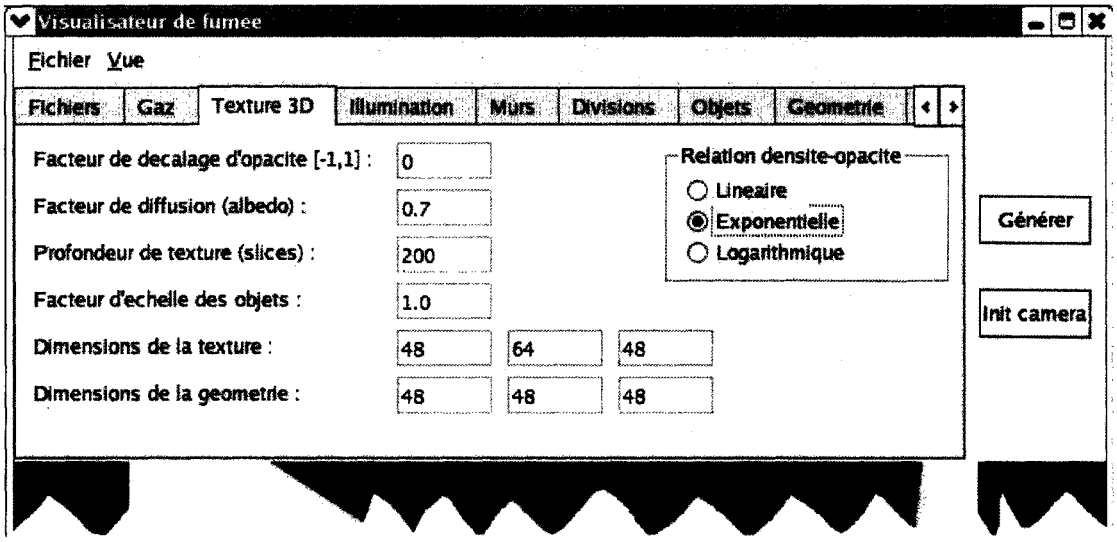



**Figure 3.10 Spécification des paramètres de la texture 3D**

La Figure 3.11 présente l'onglet permettant à l'utilisateur de modifier la couleur de l'éclairage, la couleur du gaz et de spécifier la position de la source lumineuse principale pour laquelle l'illumination du gaz est calculée.

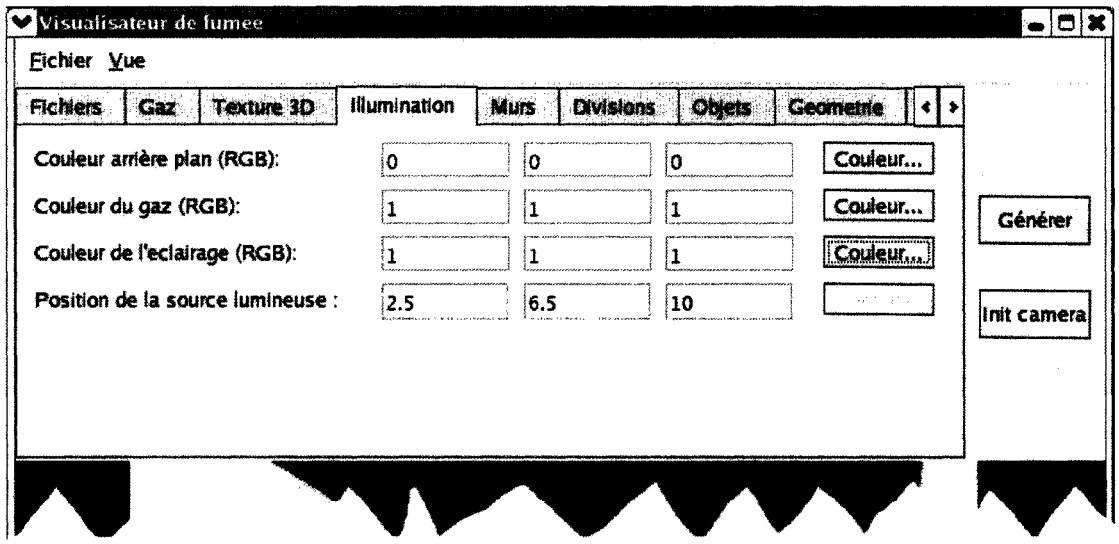



**Figure 3.11 Spécification des paramètres d'illumination**

### ❖ Édition d'environnement

Afin de rendre la simulation plus vraisemblable, il convient de pouvoir afficher, en plus du phénomène gazeux, l'environnement dans lequel évolue la simulation. En effet, le milieu peut parfois être indissociable du phénomène gazeux. Par exemple, dans une simulation de fumée se propageant dans une pièce, il est nécessaire de reproduire les objets, puisque ces derniers sont des obstacles que le gaz doit contourner. Pour la bonne compréhension de la visualisation, nous avons ajouté des fonctionnalités à l'application permettant à l'utilisateur d'éditer des murs (Figure 3.12 et Figure 3.13) et des objets (Figure 3.14) à l'intérieur de la scène.

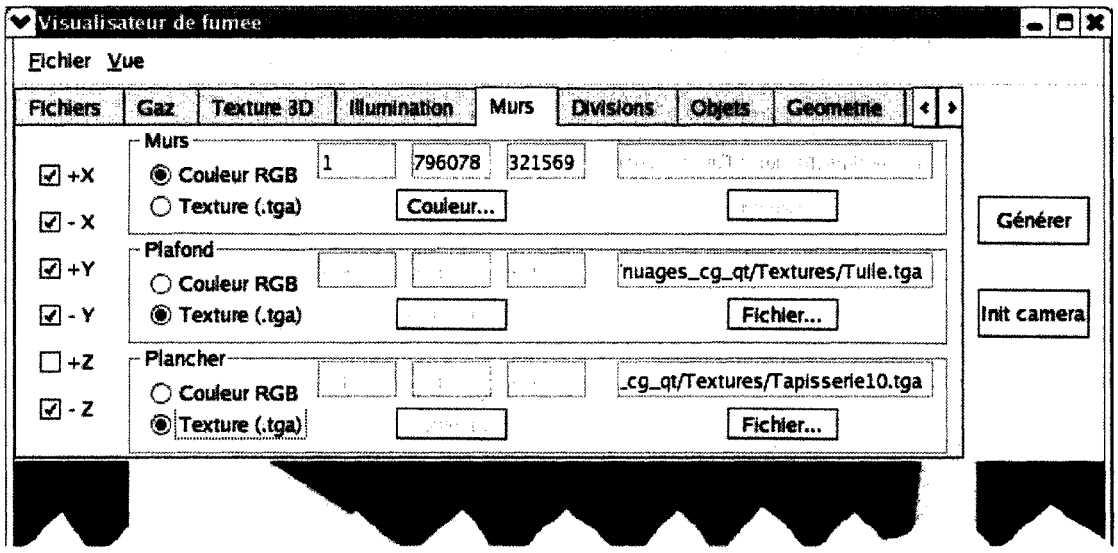



**Figure 3.12 Édition de murs externes**

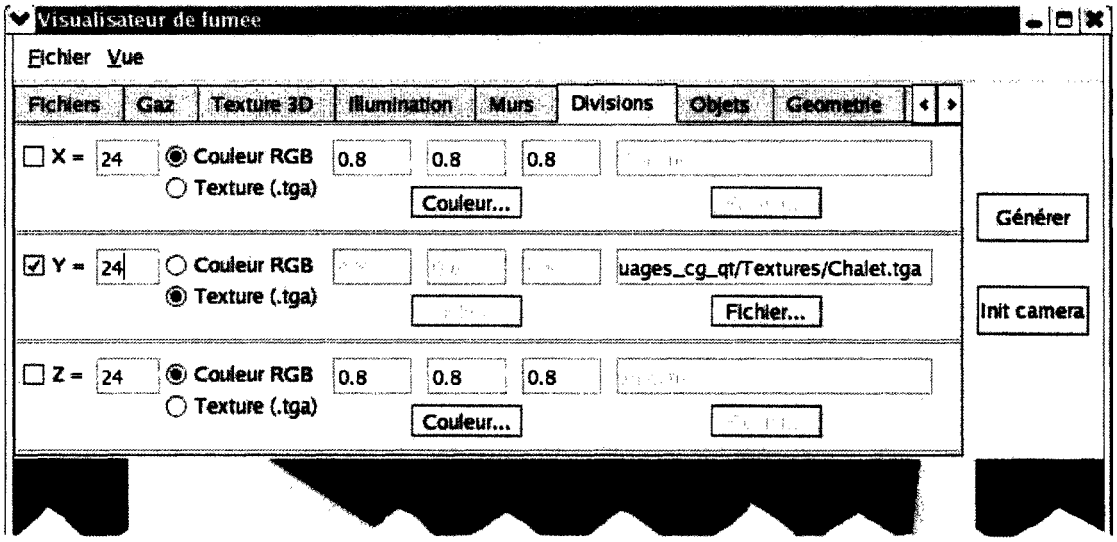



**Figure 3.13 Édition de divisions internes**

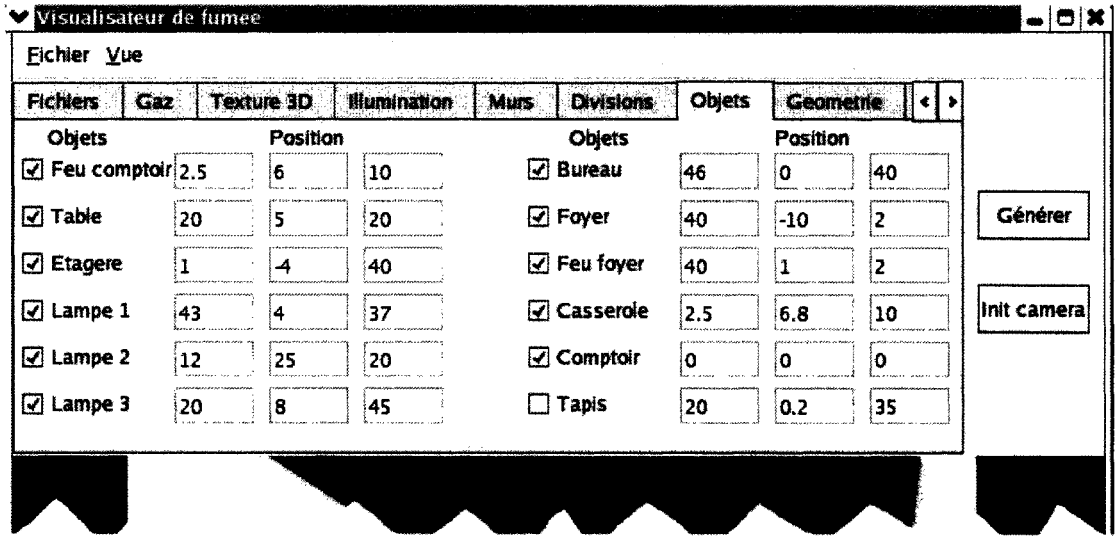



**Figure 3.14 Édition d'objets**

Il est également possible d'incorporer un maillage représentant la géométrie de la scène. Nous avons utilisé le même format de fichier que pour les géométries pouvant être visionnées par le logiciel VU [41].

3.8.2 Design de l'implémentation

La Figure 3.15 présente les principaux objets composant la structure du programme. Nous allons en faire une brève description. Des diagrammes plus complets sont insérés à l'annexe 1.

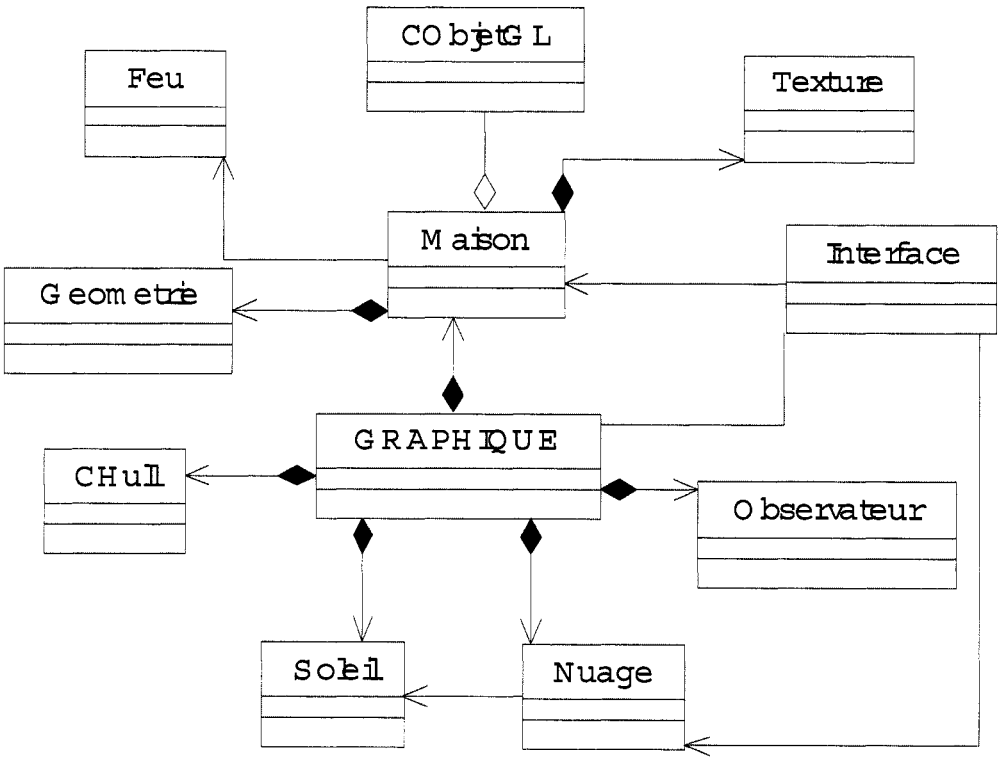



**Figure 3.15 Principaux objets du programme**

Classe *Interface* : La classe *Interface* permet de recevoir les informations provenant de l'usager et de les retransmettre aux classes concernées : GRAPHIQUE, Maison et Nuage. Elle implémente l'interface graphique de l'application en utilisant la librairie Qt. Elle initialise une entité de la classe GRAPHIQUE pour ensuite établir le lien entre l'utilisateur et les différentes fonctionnalités de l'application.

Classe *GRAPHIQUE* : La classe *GRAPHIQUE* gère l'ensemble de la visualisation. Utilisant les librairies OpenGL et Cg, elle est responsable de l'affichage de l'image en appelant les procédures d'affichage des différents éléments de la scène, notamment celle du phénomène gazeux, ainsi que celle de l'illumination de la scène. Elle capte les événements de la souris et du clavier pour permettre à l'utilisateur de déplacer la caméra dans le monde virtuel.

Classe *Nuage* : La classe *Nuage* représente le phénomène gazeux à visualiser. Elle encapsule les opérations effectuant la construction de la texture 3D ainsi que de la série de plans. Cette classe est également responsable du calcul de la transparence et de la propagation de la lumière dans le gaz.

Classe *Observateur* : La classe *Observateur* encapsule la caméra dans le monde virtuel. Elle calcule les déplacements et rotations de la caméra en fonction du point de vue demandé par l'usager. Elle permet également d'obtenir les informations sur les propriétés de la caméra.

Classe *Soleil* : La classe *Soleil* est une classe très simple permettant de représenter la source de lumière à partir de laquelle l'illumination du gaz sera calculée. Elle est un attribut de la classe Nuage.

Classe *Maison* : La classe Maison contrôle l'ensemble des éléments de la scène, à l'exception du gaz. Elle possède une liste d'objets, hérités de la classe *CObjetGL* dont elle contrôle l'affichage. Elle permet également l'affichage de murs, de feu et de l'instance de la géométrie, selon les indications de l'utilisateur.

Classe *Geometrie* : La classe *Geometrie* forme un maillage de scène obtenu par le parcours d'un fichier contenant une liste de surfaces. Le format de fichier est le même que celui supporté par le logiciel VU. Une liste d'affichage OpenGL est créée contenant cet ensemble de sommets afin d'en accélérer le rendu.

# CHAPITRE 4 - RÉSULTATS ET DISCUSSION

## 4.1 Exemples d'images obtenues

Nous nous sommes servis de notre application pour visualiser différentes simulations. Les premières portent sur la modélisation de fumée engendrée par un incendie. Une autre porte sur une explosion dans un milieu fermé, tandis que les dernières modélisent des nuages. Voici quelques exemples d'images obtenues.

### 4.1.1 Modélisation de fumée

La première simulation illustre un feu prenant naissance sur un poêle dégageant une fumée qui se propage graduellement à l'intérieur de la cuisine. La cloison séparant les deux pièces a été enlevée pour permettre de mieux voir la fumée. Les paramètres utilisés lors de cette simulation se retrouvent dans le Tableau 4.1. La Figure 4.1 illustre l'état de la fumée après 100 secondes, tandis que la Figure 4.2 et la Figure 4.3 correspondent à des temps de simulation de 160 et 250 secondes respectivement.

**Tableau 4.1 Paramètres de simulation pour l'incendie dans une cuisine**

| | |
|---|---|
| Simulation : | Fumée dans une cuisine |
| Seuil de densité minimale : | 0,362 |
| Seuil de densité maximale : | 1,198 |
| Décalage sur la courbe d'opacité : | 0,0 |
| Type de relation d'opacité : | Exponentielle |
| Nombre de plans de projection : | 200 |
| Dimensions de la texture : | 48x64x48 |
| Valeur d'albedo : | 0,7 |
| Créateur des données numériques : | Jean-Philippe Hardy |
| Créateur de l'environnement : | Étienne Lefort |

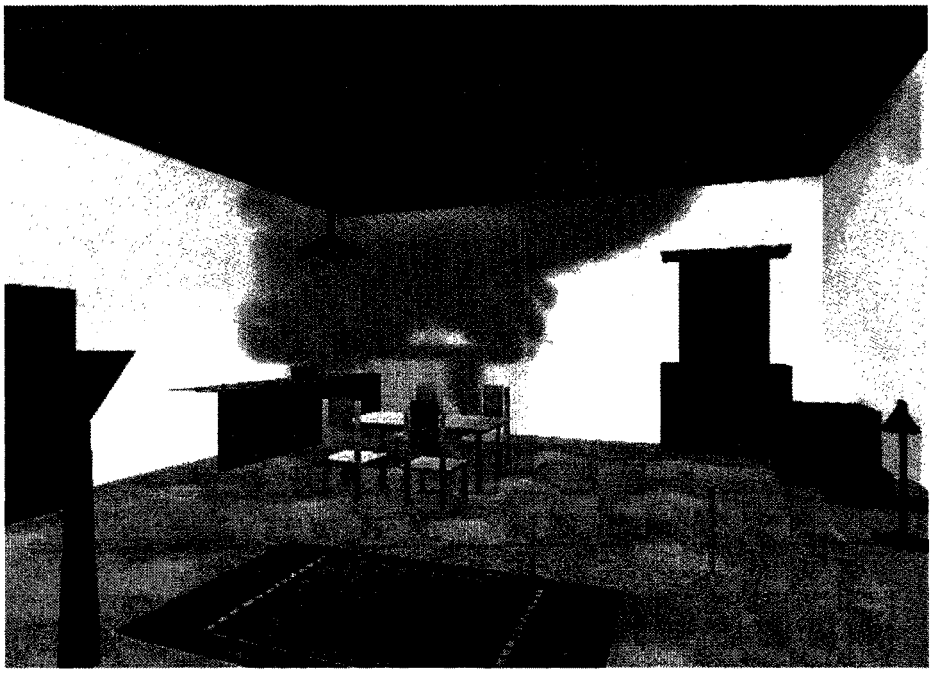


**Figure 4.1 État de la fumée après 100 secondes**

**Figure 4.2 État de la fumée après 160 secondes**

**Figure 4.3 État de la fumée après 250 secondes**

La seconde simulation présente également un rendu de fumée dans une cuisine. Cette fois, la scène est formée par l'incorporation d'une géométrie cubique. La Figure 4.4 et la Figure 4.5 montrent la fumée se propageant graduellement dans la pièce. Le Tableau 4.2 regroupe les paramètres de simulation utilisés.

**Tableau 4.2 Paramètres de simulation pour l'incendie dans une cuisine**

| Simulation : | Fumée dans une cuisine |
|---|---|
| Seuil de densité minimale : | 1,19848 |
| Seuil de densité maximale : | 4,3605 |
| Décalage sur la courbe d'opacité : | +0,7 |
| Type de relation d'opacité : | Exponentielle |
| Nombre de plans de projection : | 128 |
| Dimensions de la texture : | 65x65x65 |
| Valeur d'albedo : | 0,3 |
| Créateur des données numériques : | Jean-Philippe Hardy |
| Créateur de l'environnement : | Daniel Barrero |

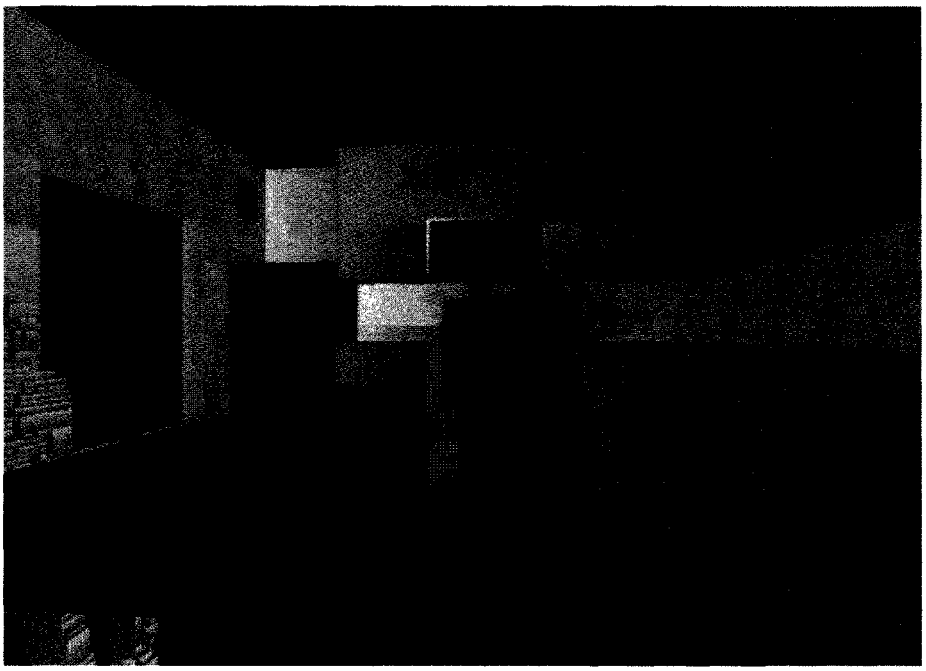


**Figure 4.4 Fumée après 3 minutes**

**Figure 4.5 Fumée après 15 minutes**

La Figure 4.6 et la Figure 4.7 illustrent des images prises lorsque la caméra se déplace à l'intérieur de la fumée. Les paramètres utilisés se retrouvent dans le Tableau 4.3.

**Tableau 4.3 Paramètres de la simulation de la pièce enfumée**

| | |
|---|---|
| Simulation : | Pièce enfumée |
| Seuil de densité minimale : | 1,1985 |
| Seuil de densité maximale : | 1,6 |
| Décalage sur la courbe d'opacité : | +0,8 |
| Type de relation d'opacité : | Exponentielle |
| Nombre de plans de projection : | 200 |
| Dimensions de la texture : | 76x73x65 |
| Valeur d'albedo : | 0,5 |
| Créateur des données numériques : | Jean-Philippe Hardy |
| Créateur de l'environnement : | Daniel Barrero |

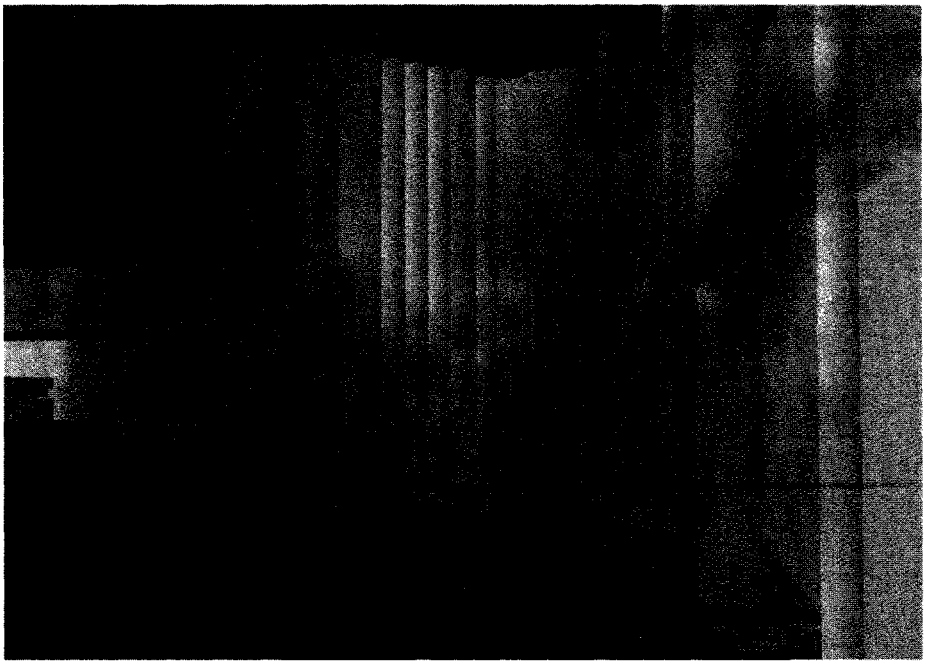


**Figure 4.6 La fumée atteint la caméra**

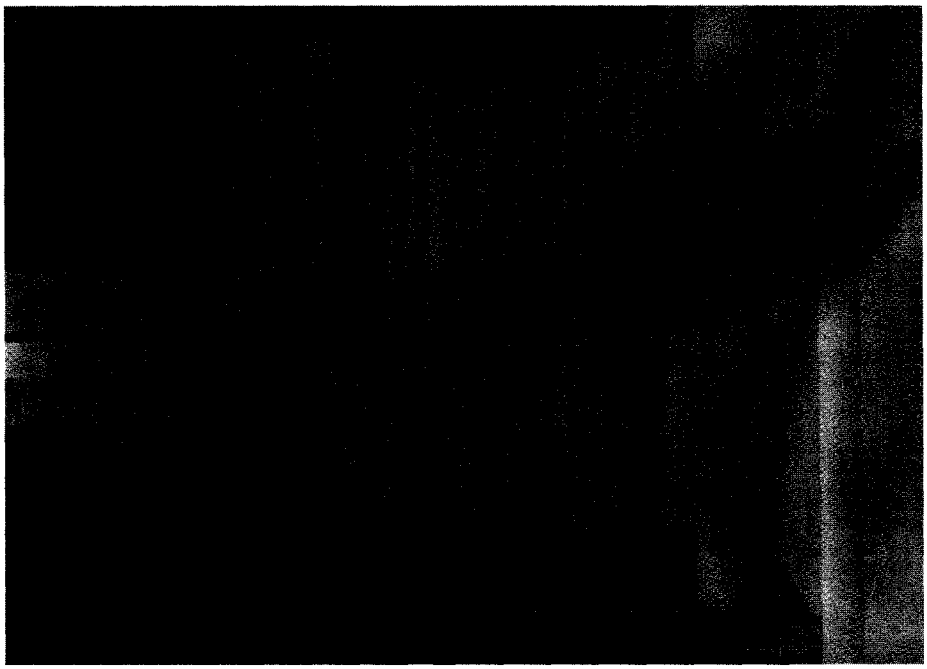


**Figure 4.7 La caméra se déplace dans la fumée**

Finalement, la Figure 4.8 ainsi que la Figure 4.9 montrent une simulation dans laquelle une maison est incendiée. Il est à noter que notre projet portant uniquement sur le rendu de phénomènes gazeux, nous avons délaissé le rendu de flammes. Les paramètres de la fumée sont inscrits dans le Tableau 4.4.

**Tableau 4.4 Paramètres de la simulation de la maison incendiée**

| | |
|---|---|
| Simulation : | Maison incendiée |
| Seuil de densité minimale : | 1,19848 |
| Seuil de densité maximale : | 1,4 |
| Décalage sur la courbe d'opacité : | +0,8 |
| Type de relation d'opacité : | Exponentielle |
| Nombre de plans de projection : | 250 |
| Dimensions de la texture : | 61x81x61 |
| Valeur d'albedo : | 0,7 |
| Créateur des données numériques : | Jean-Philippe Hardy |
| Créateur de l'environnement : | Juan Abanto |

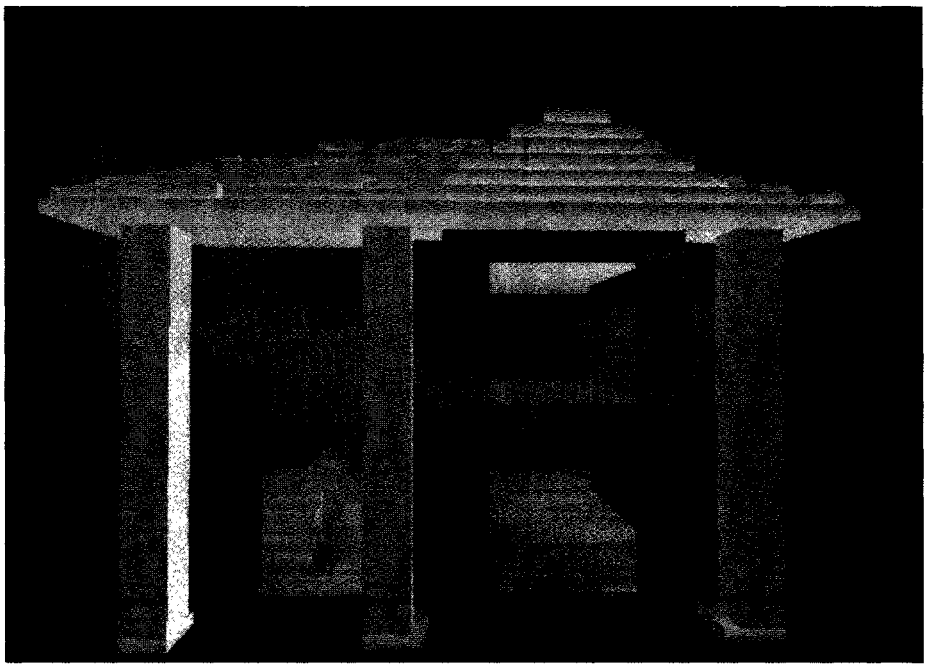


**Figure 4.8 Maison incendiée au temps de 85 minutes**

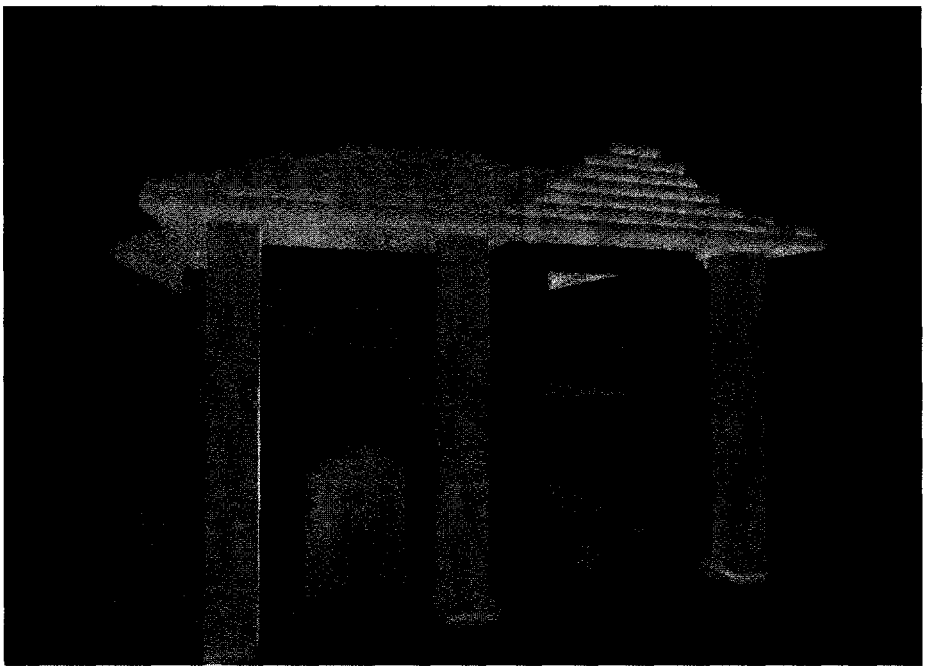


**Figure 4.9 Maison incendiée au temps de 135 minutes**

4.1.2 Modélisation d'explosion

La Figure 4.10 et la Figure 4.11 montrent de la fumée dégagée par le début et la fin d'une explosion. La simulation se déroule dans un milieu fermé, mais un mur a été supprimé de la visualisation pour mieux dévoiler le comportement du gaz. Le Tableau 4.5 renferme les différents paramètres de simulation.

**Tableau 4.5 Paramètres de la simulation d'explosion**

| | |
|---|---|
| Simulation : | Explosion |
| Seuil de densité minimale : | 0,0 |
| Seuil de densité maximale : | 100 |
| Décalage sur la courbe d'opacité : | +0,9 |
| Type de relation d'opacité : | Logarithmique |
| Nombre de plans de projection : | 200 |
| Dimensions de la texture : | 32x32x32 |
| Valeur d'albedo : | 0,9 |
| Créateur des données numériques : | Jean-Philippe Hardy |
| Créateur de l'environnement : | Étienne Lefort |

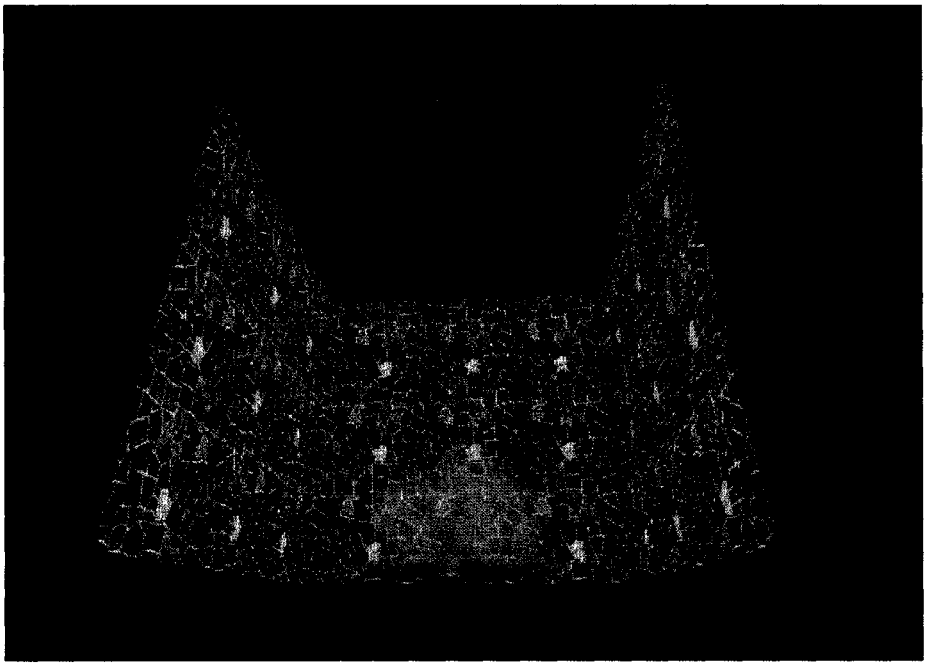


**Figure 4.10 Début d'explosion**

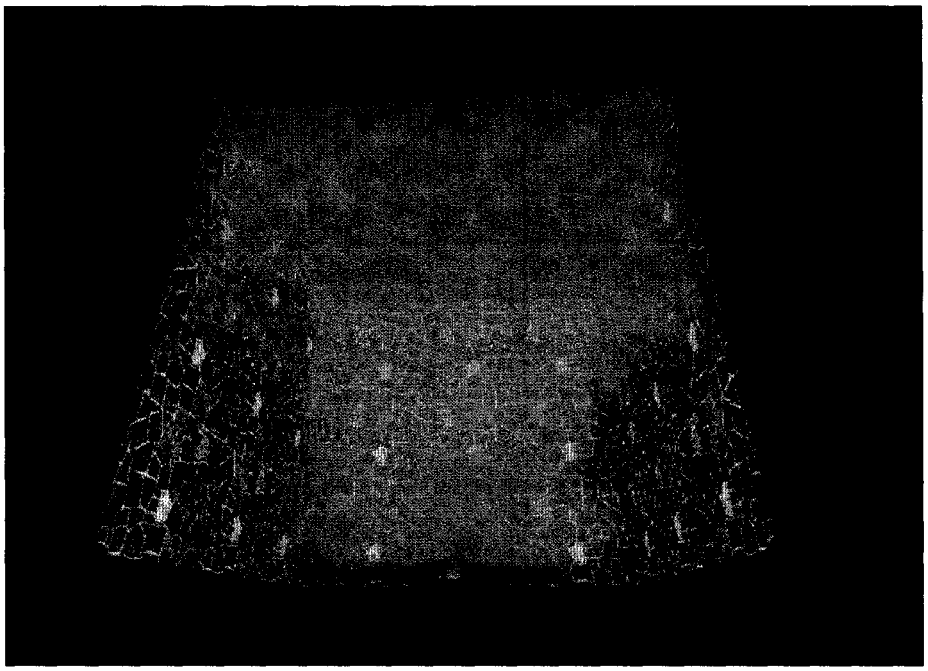


**Figure 4.11 Fin de l'explosion**

4.1.3 Modélisation de nuages

La Figure 4.12, la Figure 4.13 ainsi que la Figure 4.14 illustrent différentes modélisations de nuages. Comme l'indique le Tableau 4.6, la constante d'albedo utilisée est plus élevée que pour l'ensemble des simulations de fumée précédentes, ce qui lui attribue une intensité plus élevée.

**Tableau 4.6 Paramètres de simulation de nuages**

| Simulation : | Nuages |
|---|---|
| Seuil de densité minimale : | 1,19848 |
| Seuil de densité maximale : | 4,3605 |
| Décalage sur la courbe d'opacité : | +0,7 |
| Type de relation d'opacité : | Exponentielle |
| Nombre de plans de projection : | 200 |
| Dimensions de la texture : | 65x65x65 |
| Valeur d'albedo : | 0,9 |
| Créateur des données numériques : | Jean-Philippe Hardy |

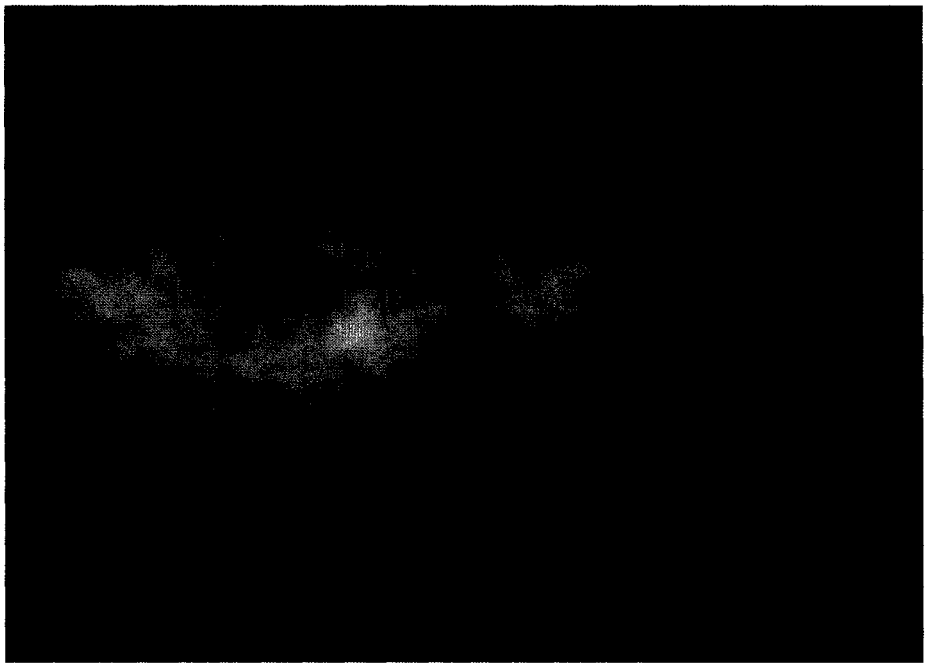


**Figure 4.12 Modélisation de nuages**

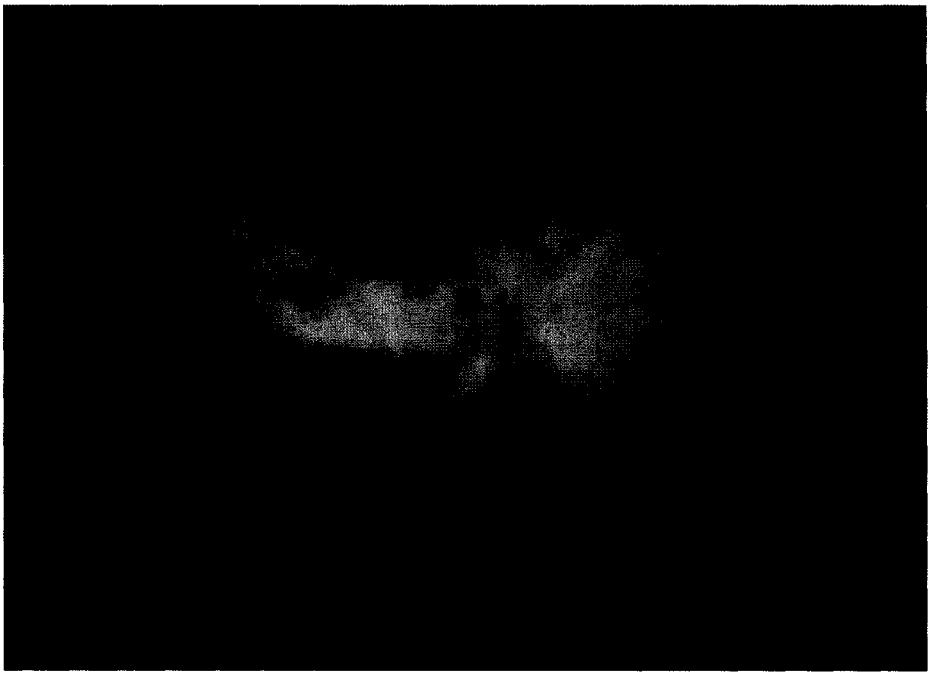


**Figure 4.13 Modélisation de nuages légèrement ombragés**

**Figure 4.14 Modélisation de nuages**

## 4.2 Avantages et limites de la modélisation proposée

À présent, nous pouvons reprendre les trois critères déterminés initialement, soit le réalisme du rendu, l'exactitude des résultats et l'interactivité, afin d'évaluer notre modélisation.

Au niveau du réalisme du rendu, certains compromis ont été effectués pour obtenir une bonne performance. Ceux-ci ont été appliqués au niveau du calcul de l'illumination. Premièrement, l'illumination est calculée en tenant compte d'une seule source de lumière, considérée comme étant la source d'éclairage principale. Cette simplification peut être adéquate dans le cas de rendu de nuages où la source de lumière dominante est sans contredit le soleil. Par contre, pour des milieux de dimensions plus restreintes, comme des pièces où des phénomènes de vapeur ou de fumée ont lieu, l'illumination pourrait s'avérer plus réaliste si plusieurs sources d'éclairage étaient prises en compte. Au niveau de l'implémentation, cette modification engendrerait peu de répercussions, car la première phase de calcul concernant la propagation de la lumière n'aurait qu'à être effectuée à plusieurs reprises. Évidemment, le coût en calcul supplémentaire serait proportionnel au nombre de sources lumineuses. Si le niveau d'interactivité n'était pas à prendre en considération, ces calculs additionnels vaudraient la peine d'être effectués, car le rendu s'en trouverait grandement amélioré. En effet, en utilisant une seule source d'illumination, une partie du gaz se retrouve illuminée, laissant l'autre partie inévitablement ombragée. L'utilisation de plusieurs sources d'illumination comblerait cette lacune et créerait des effets d'illumination plus convaincants.

Une autre simplification concerne également le calcul de la propagation de la lumière. Notre modélisation réduit le calcul de la diffusion multiple en considérant uniquement deux directions de propagation de la lumière : celle de la direction des rayons lumineux et celle vers l'observateur. Il ne s'agit donc pas d'une véritable diffusion multiple. Pour obtenir un rendu plus réaliste, il faudrait effectuer plusieurs itérations de diffusion. Cela

permettrait d'obtenir une meilleure répartition de la lumière dans le cas de gaz disposant d'une valeur d'albedo élevée. Toutefois, l'effet serait beaucoup moins notable pour des albedos faibles. Quoiqu'il en soit, notre approximation de la diffusion multiple permet quand même d'obtenir un rendu qui se rapproche de la réalité et un observateur moyen ne devrait pas être dérangé par cette simplification. D'ailleurs, cette dernière était essentielle pour pouvoir conserver une interactivité fluide, car les coûts en calcul pour augmenter le nombre d'itérations de diffusion sont très élevés.

Au niveau de l'exactitude des données, notre modélisation s'avère représentative des résultats numériques. Aucune variable aléatoire n'intervient dans le calcul de rendu, tout reposant sur les données de densité et de composition chimique. Le réalisme de la visualisation dépend donc du raffinement de l'échantillonnage de l'espace. Un échantillonnage trop grossier engendrera une image de gaz dont la structure semblera trop délimitée.

Sur le plan de l'interactivité, notre modélisation permet une performance suffisante pour qu'un utilisateur puisse visualiser le phénomène gazeux de façon intuitive. Les cartes graphiques sont optimisées pour les calculs de textures, de sorte qu'une texture 3D raisonnablement volumineuse peut facilement être supportée par l'application. L'observateur peut effectuer des déplacements et des rotations à la scène sans que des délais viennent troubler la visualisation. Finalement, les approximations apportées au calcul de la luminosité permettent au calcul de rendu d'être suffisamment rapide pour supporter l'animation d'une simulation à travers le temps.

De façon générale, nous pouvons conclure que nos objectifs initiaux ont été atteints. Évidemment, cette évaluation ne peut s'effectuer que par des jugements subjectifs. Le Tableau 4.7 illustre les cotes d'atteinte des objectifs au niveau du réalisme, d'exactitude et d'interactivité que nous nous sommes attribuées.

**Tableau 4.7 Évaluation de l'atteinte des objectifs**

| Réalisme | Exactitude | Interactivité |
|---|---|---|
| Bon | Parfaite | Parfaite |

## 4.3 Comparaison avec d'autres approches

Si beaucoup de recherches ont été effectuées ces dernières années au niveau de la modélisation des phénomènes gazeux, les applications permettant leur visualisation sont peu répandues. Nous pouvons tout de même en relever deux. La première, VU, est fort utilisée dans le domaine de la recherche pour le domaine de la visualisation en général. La seconde, Smokeview, a été développée spécialement pour la visualisation de fumée. Dans cette section, nous allons comparer ces deux logiciels avec l'application que nous avons développée.

### 4.3.1 Comparaison avec VU

VU est un programme conçu pour la visualisation de solutions numériques. Toutefois, VU n'est pas spécialisé pour la visualisation de données portant sur des gaz. La visualisation de ces derniers ne peut être effectuée que par la création d'iso-surfaces ou par des plans semi-transparents représentant les champs de densité. La Figure 4.15 illustre un exemple de fumée obtenue par VU.

**Figure 4.15 Fumée formée de plans semi-transparents créée par VU**

La modélisation utilisée par VU nécessite peu de temps de calcul et offre des informations basées sur les résultats de densité. Par contre, elle offre peu de réalisme de rendu, aucun calcul d'illumination n'étant effectué.

### 4.3.2 Comparaison avec Smokeview

Smokeview est une application développée pour visualiser les résultats de calculs numériques générés par FDS [40]. Si elle vise spécifiquement la modélisation de fumée, elle utilise néanmoins des méthodes semblables à celles de VU. Trois méthodes principales sont suggérées à l'usager. La première compresse les résultats de densité en une série de plans semi-transparents. Un exemple est donné à la Figure 4.16.

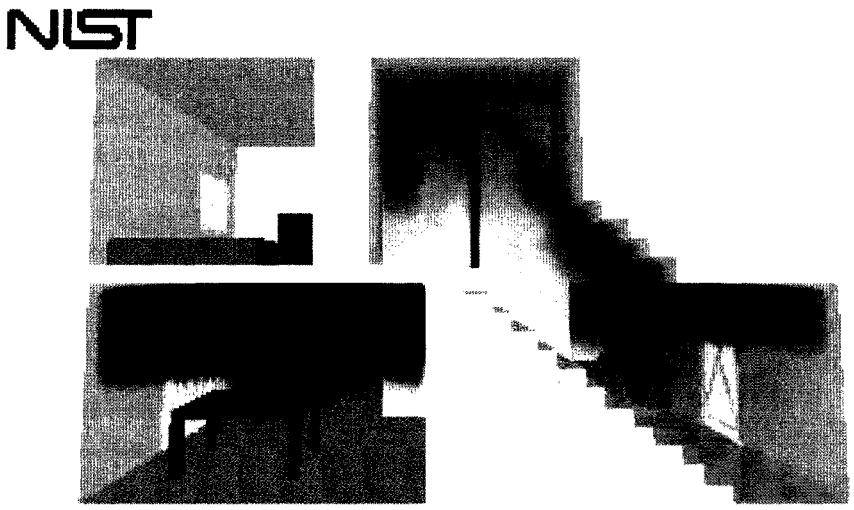



**Figure 4.16 Fumée formée de plans semi-transparents créée par Smokeview**

La deuxième méthode utilisée par Smokeview consiste à représenter la fumée par un ensemble de particules pouvant être animées dans le temps. La Figure 4.17 en présente un résultat.

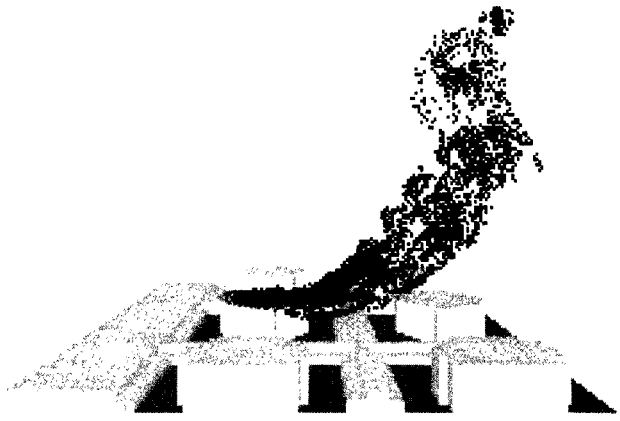


**Figure 4.17 Fumée formée de particules créée par Smokeview**

Finalement, tout comme VU, Smokeview permet de représenter la fumée par des iso-surfaces. La Figure 4.18 en donne un exemple.

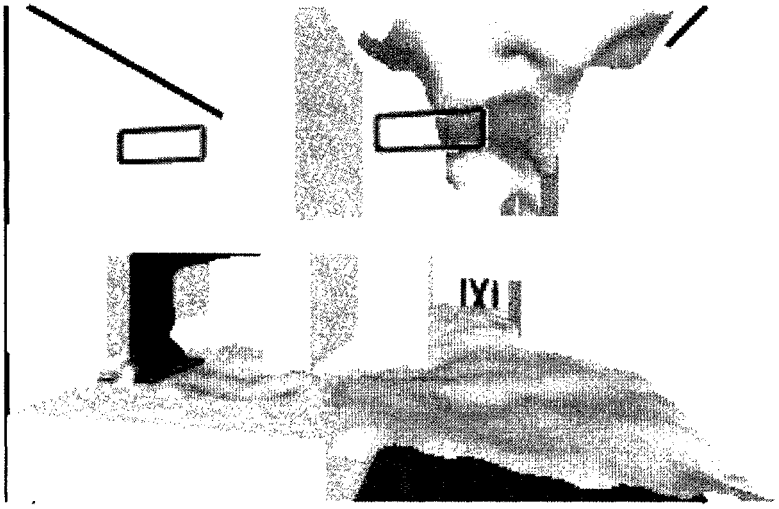


**Figure 4.18 Fumée formée d'iso-surfaces créée par Smokeview**

Comme nous pouvons le constater, Smokeview, à l'instar de VU, permet d'explorer les résultats numériques, mais reste limité au point de vue du réalisme visuel. L'application permet de comprendre le comportement de la fumée, mais ne représente pas sa véritable apparence. Notre application permet d'effectuer un pas de plus en tenant compte de l'illumination à un coût en ressources très acceptable.

Le tableau Tableau 4.8 reprend les types de modélisations suggérées par la littérature ainsi que celles supportées par VU et Smokeview à des fins comparatives. Les valeurs sur fond rouge ont trait à la modélisation que nous avons développée. Nous pouvons nous apercevoir que cette dernière répond plus adéquatement à nos trois critères d'évaluation que les autres méthodes proposées. Il est à noter toutefois que la méthode des nuées, si elle était implémentée autrement que par lancer de rayon, pourrait probablement engendrer des résultats comparables à ceux que nous avons obtenus.

**Tableau 4.8 Comparaison avec les autres approches**

| Type de modélisation | Implémentation | Réalisme | Exactitude | Interactivité |
|---|---|---|---|---|
| **Champs vectoriels** | Littérature | Faible | Bonne | Bonne |
| **Surfaces iso-densité** | Littérature | Moyen | Faible | Bonne |
| | VU | Faible | Faible | Bonne |
| | Smokeview | Faible | Faible | Bonne |
| **Méthode des nuées** | Littérature | Bon | Bonne | Moyenne* |
| **Lancer de rayon** | Littérature | Bon | Bonne | Faible |
| **Système de particules** | Littérature | Moyen | Bonne | Moyenne |
| | Smokeview | Faible | Bonne | Bonne |
| **Hypertextures** | Littérature | Bon | Moyenne | Bonne |
| | [illegible] | [illegible] | [illegible] | [illegible] |
| **Méthode des ellipsoïdes** | Littérature | Moyen | Faible | Moyenne |
| **Méthodes de projection** | Littérature | Bon | Moyenne | Moyenne |
| | VU | Moyen | Bonne | Bonne |
| | Smokeview | Moyen | Bonne | Bonne |

* Dépend de la méthode de rendu utilisée

# CONCLUSION ET TRAVAUX FUTURS

L'objectif de notre projet consistait à déterminer une méthode de modélisation pour les phénomènes gazeux permettant d'effectuer un bon compromis entre le réalisme de rendu, l'exactitude des données et la performance. Or, nos résultats indiquent que la méthode utilisant les textures 3D pouvait allier ces trois exigences. L'approximation de la diffusion multiple par la diffusion vers deux directions permet d'obtenir un rendu supérieur à la diffusion simple. Cet algorithme donne des résultats intéressants surtout au niveau de la fumée. Toutefois, l'approximation devient plus grossière dans le cas où l'albedo est plus élevé, comme c'est souvent le cas avec les nuages. Pour ce type de gaz, notre modélisation pourrait ne pas combler les besoins d'une visualisation où les attentes au niveau du rendu seraient particulièrement élevées.

De façon générale, nous avons vérifié l'hypothèse avançant que la visualisation des différents phénomènes gazeux pouvait être soumise à une même procédure. En utilisant uniquement deux variables, le décalage sur la courbe de transparence et l'indice d'albedo, un utilisateur peut obtenir différents types de rendu pour un même fichier de données.

Évidemment, puisque notre modélisation diminue le niveau de réalisme au profit de la performance, le développement constant du matériel informatique laisse présager de nouvelles ouvertures à la visualisation. Il est probable que d'ici peu, il sera envisageable d'effectuer plus d'une itération pour le calcul de la propagation de la lumière tout en conservant la même fluidité d'interactivité. De même, des dimensions de textures 3D de plus en plus volumineuses pourront être utilisées. Toutefois, les algorithmes de base que nous avons décrits pourront continuer à être utilisés.

Suite aux résultats que nous avons obtenus, nous avons décidé d'implémenter notre modélisation pour un environnement d'immersion. À cette fin, la CAVE (« Cave

Automatic Virtual Environmnent ») a été utilisée et nous avons obtenu des résultats très intéressants. Hormis une certaine latence causée par les calculs onéreux au niveau de la transparence, la même qualité d'image était obtenue que sur une station Linux conventionnelle. Il s'agit donc d'une option intéressante pour des fins de visualisation.

Nous avons mentionné qu'en raison de considérations de performance, nous avons dû effectuer certains compromis au niveau du réalisme. Puisque la majorité du temps de calcul est consacré à parcourir une grille 3D et à effectuer des traitements itératifs, il pourrait être envisagé de paralléliser notre application et de faire appel à plus d'un processeur. Ces gains en performance pourraient éventuellement permettre de calculer l'illumination à partir de plusieurs sources de lumière et d'accroître ainsi le niveau de réalisme du rendu.

# BIBLIOGRAPHIE

[1] Gardner G. Y.: "Visual Simulation of Clouds", Computer Graphics, Volume 19, Number 3, pp. 297-303, 1985.

[2] Elinas P., Sturzlinger W.: "Real-time Rendering of 3D Clouds", Journal of Graphics Tools, Volume 5, Number 4, pp. 33-45, 2000.

[3] Max N., Crawfis R.: "Advances in Scientific Visualization", IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, Vol 2410, pp. 340-345, février 1995.

[4] Taxen G.: "Cloud Modeling for Computer Graphics", Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 1999.

[5] Dobashi Y., Kaneda K., Yamashita H., Okita T., Nishita T.: "A Simple, Efficient Method for Realistic Animation of Clouds", ACM SIGGRAPH, pp. 18-28, 2000.

[6] Heinzlreiter P., Kurka G., Volkert J., "Real-time Visualization of Clouds", Proceedings WSCG'2002 - the 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002, Plzen, Czech Republic, February 2002.

[7] King S. A., Crawfis R. A., Reid W., "Fast Animation of Amorphous and Gaseous Phenomena", Volume Graphics '99, Swansea, Wales, pp 333-346, March 1999.

[8] Heidrich W., Westermann R., Seidel H., Ertl T. « Applications of Pixel Textures in Visualization and Realistic Image Synthesis », Proceedings of the 1999 Symposium on Interacive 3D Graphics, 1999.

[9] Harris M. J., Lastra A. "Real-Time Cloud Rendering," Proceedings of Eurographics 2001, Vol. 20, No. 3, pp. 76--84, September 2001.

[10] Stam, J., Fiume E. « Turbulent Wind Fields for Gaseous Phenomena", Proceedings of SIGGRAPH '93, pp. 369-376, 1993.

[11] Stam J., Fiume E. "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes". Proceedings of SIGGRAPH '95, pp. 129-136, 1995.

[12] Stam J. "Interacting with smoke and fire in real time", Communications of the ACM, Volume 43, Issue 7, pp. 76-83, 2000.

[13] Stam J. "A General Animation Framework for Gaseous Phenomena", ERCIM Research Report R047, January 1997.

[14] Stam J. « Multi-Scale Stochastic Modelling of Complex Natural Phenomena », PhD Thesis, Dept. of Computer Science, University of Toronto, 1995.

[15] Fedkiw R., Stam J. et Jensen H. W. "Visual Simulation of Smoke", SIGGRAPH 2001 Conference Proceedings, Annual Conference Series, pp. 15-22, August 2001.

[16] Stam J. "Stochastic Rendering of Density Fields", Proceedings of Graphics Interface '94, pp. 51-58, May 1994.

[17] Sillion F. « Clustering and Volume Scattering for Hierarchical Radiosity Calculations », Fifth Eurographics Workshop on Rendering, pp. 105-117, Juin 1994.

[18] Rasmussen, N., Nguyen, D., Geiger, W. and Fedkiw, R. "Smoke Simulation for Large Scale Phenomena", SIGGRAPH 2003, ACM TOG 22, pp. 703-707, 2003.

[19] Nishita T., Dobashi Y., Nakamae E. "Display of Clouds taking into Account Multiple Anisotropic Scatterring and Sky Light", ACM SIGGRAPH, pp. 379-386, 1996.

[20] Max, Nelson. « Optical Models for Direct Volume Rendering », IEEE Transactions on Visualization and Computer Graphics, vol. 1 no. 2, June 1995.

[21] Pattanaik, S. N., Murdur, S. P. "Computation of Global Illumination in a Participating Medium by Monte Carlo Simulation", The Journal of Visualisation and Computer Animation, Vol. 4(3), pp. 133-152, 1993.

[22] Ebert, D. "Procedural Modeling, Animation, and Rendering of Gases, Fluids, and Textures," *SIGGRAPH 95 Course 33 Notes,* Chapter 3, August 1995.

[23] Ebert, D. « Volumetric Modeling with Implicit Functions : A Cloud is Born », Visual Proc. of SIGGRAPH'97, p. 147, 1997.

[24] Kajiya J. T., Herzen B. V. "Ray tracing Volume Densities," Computer Graphics, Vol. 18, No. 3, pp. 165-174, 1984.

[25] Da Dalto L., Jessel J.-P. « Fast Rendering of Participating Media in a Global Illumination Application », International Conference on Visualization and Modelling, décembre 1995.

[26] Da Dalto L. et al. « A New Approach for Multiple Scattering Modelling in Participating Media", Fifth International Conference in Central Europe on Computer Graphics and Visualization, Czech Republic, 1997.

[27] Da Dalto L. « Modèles pour la simulation de phénomènes naturels en images de synthèse », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1997.

[28] Barrero D. "Simulation et visualisation de phénomènes naturels pour la synthèse d'images", Thèse de Doctorat (PhD), IRIT - Université Paul Sabatier, Toulouse, Janvier 2001.

[29] Inakage M. "Volume Tracing of Atmospheric Environments". The Visual Computer, No. 7, pp. 104-113, 1991.

[30] Rockwell G., Bradley J. « Visualisation scientifique et analyse de texte », Littérature, informatique, lecture, Presses universitaires de Limoges, 1999.

[31] Lefer W. « La visualisation scientifique », Le Bulletin de l'AFIG, No. 8, 1998.

[32] Blythe D. « Advanced Graphics Programming Techniques Using OpenGL", SIGGRAPH '99 Course, Chapitre 16, 1999.

[33] Woo M., Neider J., Davis T. and Shreiner D. « OpenGL Programming Guide », Addison-Wesley, third edition, 1999.

[34] Shreiner D. « OpenGL Reference Manual », Addison-Wesley, third edition, 2000.

[35] Fernando R., Kilgard M. J. « The Cg Tutorial : The Definitive Guide to Programmable Real-Time Graphics », Addison-Wesley, 2003.

[36] Post F. H. « Visualization techniques for vector fields », Computer Graphics and CAD/CAM Group, Faculty ITS, 2000.

[37] Hardy J.-P. « Simulations numériques et visualisation d'incendies sous ventilés avec FDS v3.10 », Mémoire de maîtrise, École Polytechnique de Montréal, Montréal, 2004.

[38] McGrattan, K.B., et al. « Fire Dynamics Simulator (Version 3) – User's Guide", NIST, 81p., NISTIR 6784, 2002.

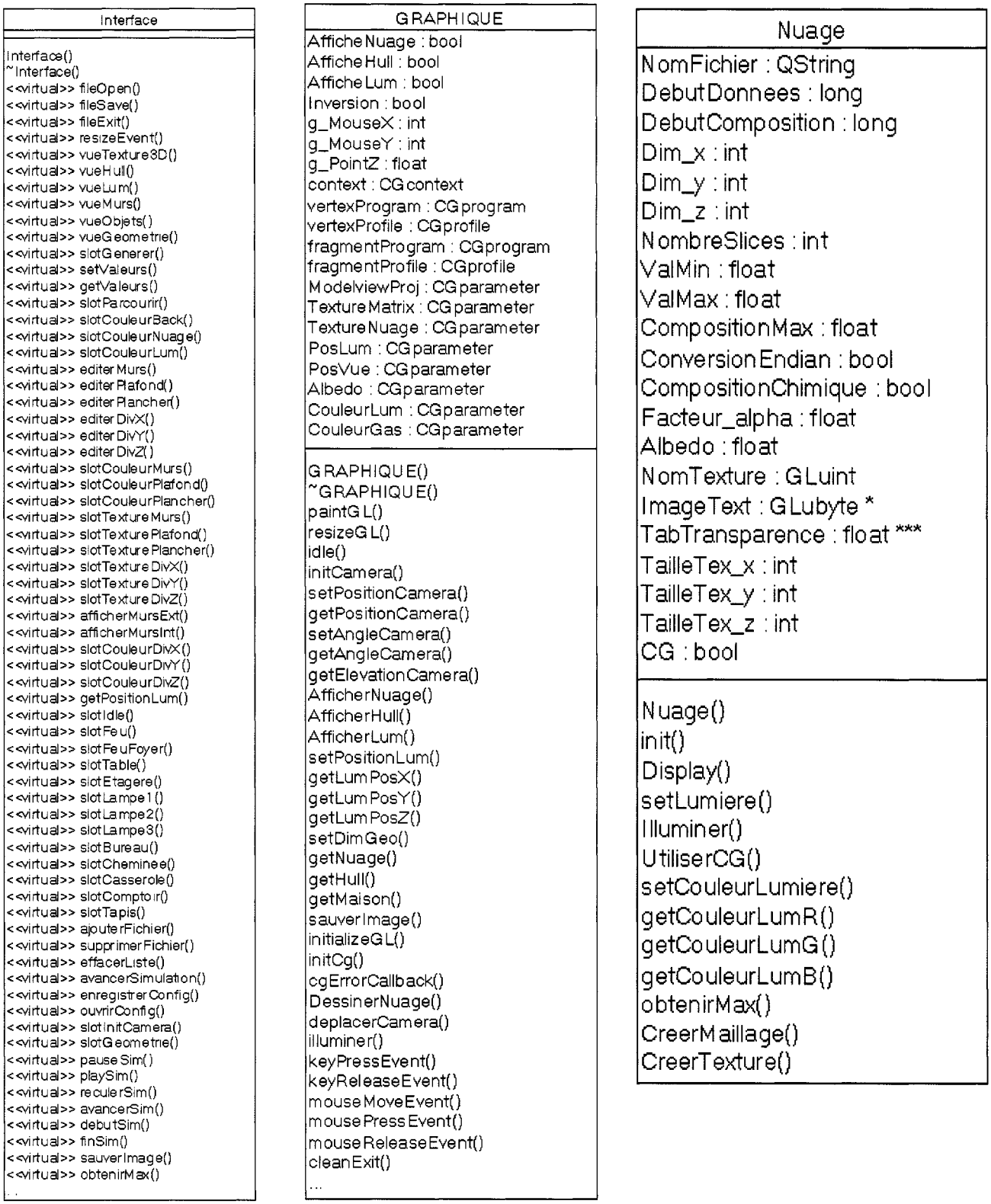
[39] McGrattan, K.B., et al. « Fire Dynamics Simulator (Version 3) – Technical Reference Guide", NIST, 51p. NISTIR 6783, 2002.

[40] Forney G. P., McGrattan K. B. « User's Guide for Smokeview Version 3.1 – A Tool for Vizualising Fire Dynamics Simulation Data », NIST, 60 p. NISTIR 6980, 2002.

[41] Ozell, B., "VU, un programme de visualisation configurable", Rapport interne du CERCA, mai 1995.

# ANNEXE

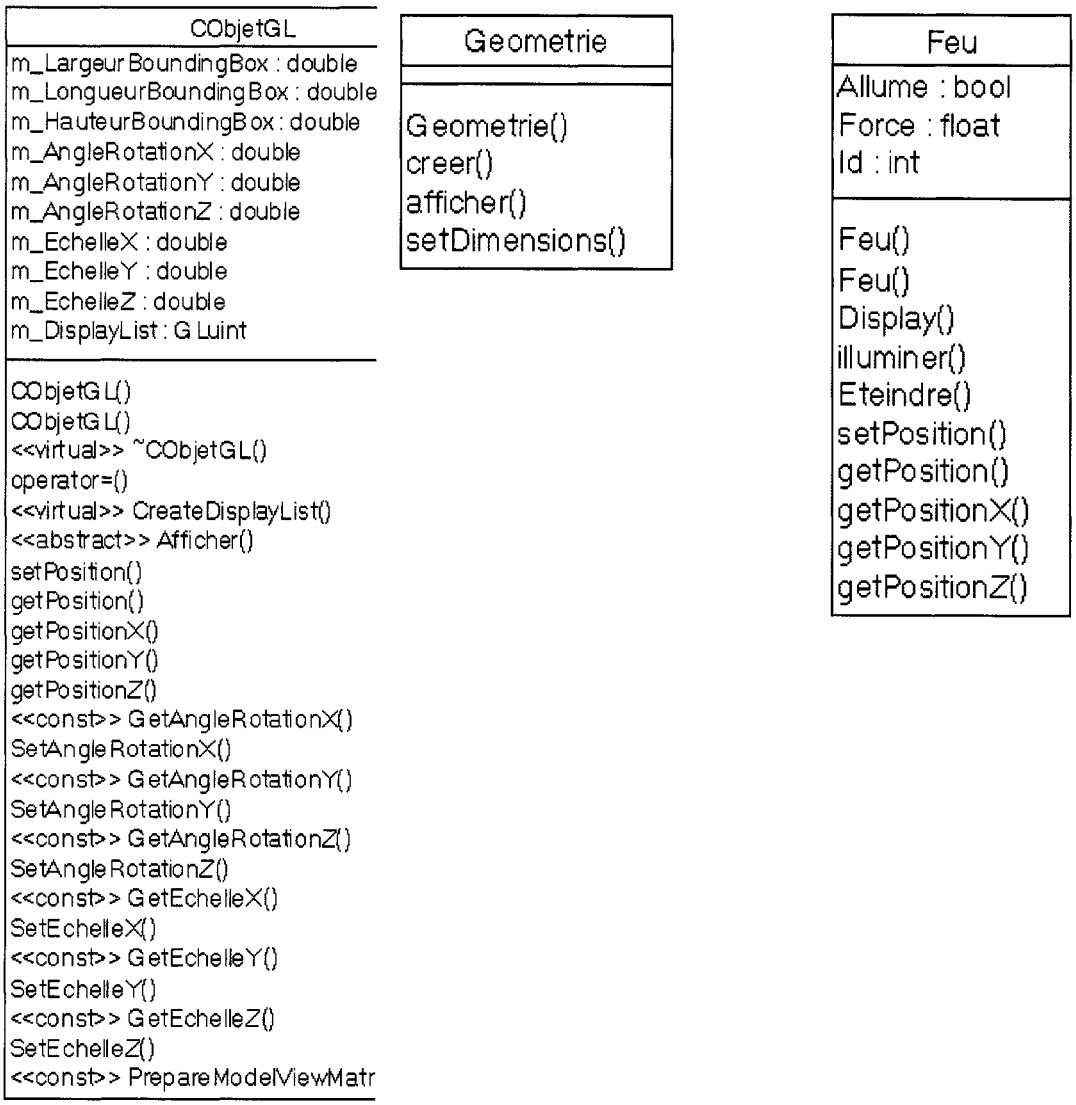
## LISTE DES CLASSES PRINCIPALES


Interface
Interface()
~Interface()
<<virtual>> fileOpen()
<<virtual>> fileSave()
<<virtual>> fileExit()
<<virtual>> resizeEvent()
<<virtual>> vueTexture3D()
<<virtual>> vueHull()
<<virtual>> vueLum()
<<virtual>> vueMurs()
<<virtual>> vueObjets()
<<virtual>> vueGeometrie()
<<virtual>> slotGenerer()
<<virtual>> setValeurs()
<<virtual>> getValeurs()
<<virtual>> slotParcourir()
<<virtual>> slotCouleurBack()
<<virtual>> slotCouleurNuage()
<<virtual>> slotCouleurLum()
<<virtual>> editerMurs()
<<virtual>> editerPlafond()
<<virtual>> editerPlancher()
<<virtual>> editerDivX()
<<virtual>> editerDivY()
<<virtual>> editerDivZ()
<<virtual>> slotCouleurMurs()
<<virtual>> slotCouleurPlafond()
<<virtual>> slotCouleurPlancher()
<<virtual>> slotTextureMurs()
<<virtual>> slotTexturePlafond()
<<virtual>> slotTexturePlancher()
<<virtual>> slotTextureDivX()
<<virtual>> slotTextureDivY()
<<virtual>> slotTextureDivZ()
<<virtual>> afficherMursExt()
<<virtual>> afficherMursInt()
<<virtual>> slotCouleurDivX()
<<virtual>> slotCouleurDivY()
<<virtual>> slotCouleurDivZ()
<<virtual>> getPositionLum()
<<virtual>> slotIdle()
<<virtual>> slotFeu()
<<virtual>> slotFeuFoyer()
<<virtual>> slotTable()
<<virtual>> slotEtagere()
<<virtual>> slotLampe1()
<<virtual>> slotLampe2()
<<virtual>> slotLampe3()
<<virtual>> slotBureau()
<<virtual>> slotCheminee()
<<virtual>> slotCasserole()
<<virtual>> slotComptoir()
<<virtual>> slotTapis()
<<virtual>> ajouterFichier()
<<virtual>> supprimerFichier()
<<virtual>> effacerListe()
<<virtual>> avancerSimulation()
<<virtual>> enregistrerConfig()
<<virtual>> ouvrirConfig()
<<virtual>> slotInitCamera()
<<virtual>> slotGeometrie()
<<virtual>> pauseSim()
<<virtual>> playSim()
<<virtual>> reculerSim()
<<virtual>> avancerSim()
<<virtual>> debutSim()
<<virtual>> finSim()
<<virtual>> sauverImage()
<<virtual>> obtenirMax()
...
GRAPHIQUE
AfficheNuage : bool
AfficheHull : bool
AfficheLum : bool
Inversion : bool
g_MouseX : int
g_MouseY : int
g_PointZ : float
context : CGcontext
vertexProgram : CGprogram
vertexProfile : CGprofile
fragmentProgram : CGprogram
fragmentProfile : CGprofile
ModelviewProj : CGparameter
TextureMatrix : CGparameter
TextureNuage : CGparameter
PosLum : CGparameter
PosVue : CGparameter
Albedo : CGparameter
CouleurLum : CGparameter
CouleurGas : CGparameter
GRAPHIQUE()
~GRAPHIQUE()
paintGL()
resizeGL()
idle()
initCamera()
setPositionCamera()
getPositionCamera()
setAngleCamera()
getAngleCamera()
getElevationCamera()
AfficherNuage()
AfficherHull()
AfficherLum()
setPositionLum()
getLumPosX()
getLumPosY()
getLumPosZ()
setDimGeo()
getNuage()
getHull()
getMaison()
sauverImage()
initializeGL()
initCg()
cgErrorCallback()
DessinerNuage()
deplacerCamera()
illuminer()
keyPressEvent()
keyReleaseEvent()
mouseMoveEvent()
mousePressEvent()
mouseReleaseEvent()
cleanExit()
...
Nuage
NomFichier : QString
DebutDonnees : long
DebutComposition : long
Dim_x : int
Dim_y : int
Dim_z : int
NombreSlices : int
ValMin : float
ValMax : float
CompositionMax : float
ConversionEndian : bool
CompositionChimique : bool
Facteur_alpha : float
Albedo : float
NomTexture : GLuint
ImageText : GLubyte *
TabTransparence : float ***
TailleTex_x : int
TailleTex_y : int
TailleTex_z : int
CG : bool
Nuage()
init()
Display()
setLumiere()
Illuminer()
UtiliserCG()
setCouleurLumiere()
getCouleurLumR()
getCouleurLumG()
getCouleurLumB()
obtenirMax()
CreerMaillage()
CreerTexture()

| Observateur |
| --- |
| Angle : float<br>Elevation : float<br>VitesseTranslation : float<br>VitesseRotation : float |
| calculerCible()<br>Observateur()<br>setPosition()<br>setPosition()<br>getPosition()<br>getPositionX()<br>getPositionY()<br>getPositionZ()<br>translater()<br>translater()<br>avancer()<br>deplacer()<br>monter()<br>rotaterPositif()<br>rotaterNegatif()<br>Elever()<br>Pencher()<br>setAngle()<br>getAngle()<br>getElevation()<br>getCible()<br>getDirection() |

| Soleil |
| --- |
| Modif : bool |
| Soleil()<br>Display()<br>setPosition()<br>getPosition()<br>getPositionX()<br>getPositionY()<br>getPositionZ()<br>getCouleurR()<br>getCouleurG()<br>getCouleurB()<br>setModification()<br>getModification()<br>setCouleur() |

| Maison |
| --- |
| afficherExt()<br>afficherDivisions()<br>Maison()<br>init()<br>Display()<br>setDimensions()<br>setFacteurEchelle()<br>creerGeometrie()<br>afficherMursExt()<br>afficherMursInt()<br>afficherFeuComptoir()<br>afficherFeuFoyer()<br>afficherTable()<br>afficherEtagere()<br>afficherLampeSalon()<br>afficherLampeCuisine()<br>afficherLampadaire()<br>afficherBureau()<br>afficherCheminee()<br>afficherCasserole()<br>afficherComptoir()<br>afficherTapis()<br>afficherMurs()<br>afficherObjets()<br>afficherGeometrie()<br>setCouleurMurs()<br>setCouleurPlancher()<br>setCouleurPlafond()<br>setCouleurDivX()<br>setCouleurDivY()<br>setCouleurDivZ()<br>setTextureMurs()<br>setTexturePlancher()<br>setTexturePlafond()<br>setTextureDivX()<br>setTextureDivY()<br>setTextureDivZ() |


CObjetGL
m_LargeurBoundingBox : double
m_LongueurBoundingBox : double
m_HauteurBoundingBox : double
m_AngleRotationX : double
m_AngleRotationY : double
m_AngleRotationZ : double
m_EchelleX : double
m_EchelleY : double
m_EchelleZ : double
m_DisplayList : GLuint
CObjetGL()
CObjetGL()
<<virtual>> ~CObjetGL()
operator=()
<<virtual>> CreateDisplayList()
<<abstract>> Afficher()
setPosition()
getPosition()
getPositionX()
getPositionY()
getPositionZ()
<<const>> GetAngleRotationX()
SetAngleRotationX()
<<const>> GetAngleRotationY()
SetAngleRotationY()
<<const>> GetAngleRotationZ()
SetAngleRotationZ()
<<const>> GetEchelleX()
SetEchelleX()
<<const>> GetEchelleY()
SetEchelleY()
<<const>> GetEchelleZ()
SetEchelleZ()
<<const>> PrepareModelViewMatr
Geometrie
Geometrie()
creer()
afficher()
setDimensions()
Feu
Allume : bool
Force : float
Id : int
Feu()
Feu()
Display()
illuminer()
Eteindre()
setPosition()
getPosition()
getPositionX()
getPositionY()
getPositionZ()