République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER DE BISKRA

N° d'ordre : IVA3/M2/2018

Faculté des sciences exactes et sciences de la nature et de la vie

Département d'informatique

Mémoire

En vue de l'obtention du diplôme de

Master académique en Informatique

(Option : Image et vie artificielle)

Thème

Génération de feu par les systèmes de particules

Présenté par :

HAFIDI Mohcen

Soutenu le 24 juin 2018, devant le jury composé de :

Présidente :	Moufida BENCHABANE	MAA	Université de Biskra
Rapporteur :	Cherif FOUDIL	PROF	Université de Biskra
Examinateur :	Mebarek BOUCETTA	MAA	Université de Biskra

Dédicaces

Je dédie ce modeste travail :

À ceux que j'ai de plus cher au monde : mes parents.

À mon frère et mes sœurs

À tous mes amis et collègues.
À tous ceux qui m'ont aimé et me souhaitent le bonheur et la réussite.

Remerciements

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

Je tiens à remercier vivement mon encadrant, Le Pr.Cherif Foudil qui, par son expérience et sa tolérance, a guidé mon travail durant toute la période de la préparation de mon mémoire. Nous voudrions également lui témoigner notre gratitude pour sa patience et son soutien qui nous a été précieux afin de mener notre travail à bon port.

J'adresse également mes remerciements à l'ensemble des enseignants qui m'ont enseigné durant mon parcoure d'études, notamment au enseignants qui ne m'ont pas enseigné.

Je tiens à exprimer notre gratitude une double fois à l'égard du Président du Jury.

nous remercions également les membres du jury pour leurs assistance malgré leurs charges pédagogiques et scientifiques.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Resumé

Ce mémoire traitera la question suivante : Comment simuler un feu 3D qui doit être proche de la réalité, interactif, éditable tout en restant en temps réel ? Le résultat qui représente la réponse à la question du sujet va être un simulateur de feu qui peut créer des animations de feu 3D proches de la réalité, interactive, éditable et optimale à l'aide d'une approche mixte à base d'un système de particules. L'interactivité du feu avec les objets de la scène implémentés dans notre système va être basée sur un algorithme de détection d'évitement de collisions que nous avons créé. L'optimisation sera dans le coté rendu et non pas dans le coté calcul et ceci a été réalisé en utilisant le rendu par instance qui fonctionne avec le parallélisme de la carte graphique.

Abstract

This brief will address the following question :How to simulate a 3D fire that has to be close to reality, interactive, editable and in real time ? The resulting work that represents the answer to the question of the subject is going to be a fire Simulator that can create a 3D fire animation which is close to reality, interactive, editable and optimal using a particles system. The interactivity of the fire with the objects of the scene implemented in our system is going to be based on an algorithm of collision detection and collision avoidance that has been created by us. The optimization will be in the rendering side and not in the calculations and this is had been achieved using the Instanced rendering that works with the graphic card's parallelism.

ملخص

ستتناول هذه الاطروحة السؤال التالي : كيف يمكن محاكاة نار ثلاثية الأبعاد و التي يجب ان تكون قريبة من الواقع ، وقابلة للتفاعل مع الاجسام الأخرى في حين يجب ان تبقي في الوقت الحقيقي ؟ النتيجة التي تمثل الاجابه على سؤال الموضوع ستكون على شكل محاكاة نار قريبة من الواقع ، تفاعلية و الأمثل, باستخدام نظام الجسيمات. التفاعل الذي يشمل النار و أجسام المشهد ستكون علي أساس خوارزم الكشف و تجنب الاصطدام الذي أنشئ من طرفنا. لقد تم التحسين من الجانب البصري و ليس من الجانب الحسابي و هذا تم القيام به باستخدام ال Instanced rendering

Table des matières

0.1 Introduction 11
0.2 Définition du problème 12
0.3 Objectifs 12
0.4 Contributions 13
0.5 Structure du mémoire 13

1 CHAPITRE I : Les Feux **15**
1.1 Introduction 16
1.2 Définition d'un feu 16
1.3 Le Triangle de feu 17
1.4 Classification des feux 18
1.5 La simulation d'un feu 23
1.5.1 Introduction 23
1.5.2 Les méthodes de simulation de feu 23
1.5.2.1 Méthode à base de cartographie de texture 24
1.5.2.2 Méthode à base physiques et mathématiques 25
1.5.2.3 Méthode à base automate cellulaire 25
1.5.2.4 Méthode à base reconstruction d'images tomographiques 25
1.5.3 Comparaison entre les méthodes de simulation d'un feu 27
1.5.4 Discutions de l'étude comparative 28
1.6 Conclusion 29

2 CHAPITRE II : Les systèmes de particules **30**
2.1 Introduction 31

2.2 Apparition et Découverte des systèmes de particules 31
2.3 Définition d'un SP 31
2.4 Une particule dans un SP 32
2.4.1 Définition 32
2.4.2 Les paramètres d'une particule 32
2.5 Algorithme de base d'un SP 33
2.5.1 Introduction 33
2.5.2 Algorithme 33
2.6 Les différents travaux de simulation de feu en utilisant les systèmes de particules 34
2.7 Conclusion 40

3 CHAPITRE III : L'interactivité et les systèmes de particules 41
3.1 Introduction 42
3.2 Interactivité avec l'environnement 43
3.3 Évitement d'une collision 44
3.4 Collision avec une sphère 45
3.5 Collision avec des objets complexes 45
3.6 Conclusion 46

4 CHAPITRE IV : Conception du système 47
4.1 Introduction 48
4.1.1 Pourquoi une approche qui se base sur les systèmes de particules ? 48
4.1.2 Quelles sont les classes de feux que nous avons simulé ? 48
4.2 Objectifs 50
4.2.1 Schémas des objectifs 50
4.2.1.1 Le réalisme 50
4.2.1.2 L'édition 51
4.2.1.3 L'Interactivité 51
4.2.1.4 L'optimalité 51

4.3 Notre modèle 52
4.3.1 Schémas de notre système 52
4.3.1.1 Réalisme 53
4.3.1.2 Édition 57
4.3.1.3 Interactivité 57
4.3.1.4 Optimalité 59
4.3.2 Notre algorithme 60
4.3.2.1 Explications et détails de notre algorithme 61
4.3.3 Comparaison avec l'algorithme de Reeves 71
4.4 Contributions 72
4.5 Conclusion 73

5 CHAPITRE V : Étude de cas et implémentations 74

5.1 Introduction 75
5.2 Outils utilisés 76
5.2.1 Langages de programmation 76
5.2.2 Bibliothèques 77
5.2.3 Environnement de programmation 77
5.3 Logiciels et matériels utilisés 78
5.4 Présentation de l'interface de l'application 79
5.4.1 Interface d'accueil 80
5.4.2 Interface principale 81
5.4.2.1 Composantes de l'interface principal 82
5.5 Tests et comparaisons 85
5.5.1 Tests concernant le réalisme 85
5.5.2 Tests concernant l'interactivité 91
5.5.3 Autres tests 95
5.5.4 Étude comparative 97
5.6 Conclusion 102

Table des figures

1-1 Le triangle de feu. 17
1-2 Un Feu de camp. 18
1-3 Les cracheurs de feu. 19
1-4 Un Feu d'un gaz. 20
1-5 Un Feu d'un métal. 21
1-6 L'huile de cuisson. 22

2-1 Le mouvement des particules en évitant un obstacle [16]. 35
2-2 Résultat d'une simulation de feu avec différentes hauteurs en utilisant un *SP* [14]. 36
2-3 Résultat d'une simulation de feu d'un bougie en utilisant un *SP* [9]. . 37
2-4 Un *SP* avec 16384 particules (à gauche) et avec 256 particules (à droite) : [12]. 38

4-1 Les objectifs de notre travail. 50
4-2 Le schémas de notre système 52
4-3 La gravité inversée 53
4-4 Avec et Sans Le mélange additif 54
4-5 Un exemple d'une de nos textures UHD 55
4-6 Les Flocons de feu dans La réalité 56
4-7 Les Flocons de feu dans une de nos textures 56
4-8 La loi de l'attraction universelle de Newton. 58
4-9 La loi d'attraction universelle de Newton multipliée par (-1) 58
4-10 Les formes des émetteurs 62

4-11 Un cône avec des vecteurs de vélocités générés aléatoirement 63
4-12 La distance avec les rayons de la sphère et de la particule. 64
4-13 Le cas d'une collision. 65
4-14 La loi de Newton avec $M_i = 1$ 67
4-15 Le cas d'une collision entre la particule p et deux objets interactifs. . 67
4-16 Le volume englobant 69

5-1 L'interface d'accueil. 80
5-2 L'interface principal. 81
5-3 La barre d'outils. 82
5-4 Feu de camp, vue rapprochée. 85
5-5 Feu de camp, vue de loin. 86
5-6 Feu de camp, vue de face. 86
5-7 Feu d'une cuisinière à gaz, vue rapprochée. 88
5-8 Feu de camp, vue de loin. 89
5-9 Test d'interactivité avec un mur de feu. 91
5-10 Test d'interactivité avec un feu de camp. 93
5-11 Autres variétés de tests 95
5-12 Comparaison avec un feu réaliste. 97
5-13 Comparaison d'un feu d'une cuisinière de notre simulation avec un feu réaliste. 99
5-14 Comparaison entre notre simulation de feu et entre les travaux précédents et des images de la réalité. 100

Liste des tableaux

1.1 Une étude comparative entre les méthodes de simulation d'un feu . . 27

4.1 Comparaison de notre algorithme avec celui de Reeves 71

5.1 Notre configuration logicielle et matérielle. 78
5.2 La configuration pour un feu de camp 87
5.3 La configuration pour un feu d'une cuisinière à gaz (test de Réalisme) 90
5.4 La configuration pour un mur de feu (test d'interactivité). 92
5.5 La configuration pour un feu de camp (test d'interactivité). 94
5.6 Tableau comparative entre notre travail et les différents travaux similaires. 101

Liste des codes sources

4.1 L'utilisation de l'invisibilité 59
4.2 Détection de collisions. 65
4.3 Calcule de distance entre deux points dans un espace 3D. 66
4.4 Incrémentation de la Vélocité. 66
4.5 changement d'échelle pour sommeDesForces. 68

Abréviations

SP **S**ystème de **P**articules

RVB **R**ouge **V**ert **B**leu

PPP **P**oint **P**ar **P**ouce

UHD **U**ltra **H**aute **D**éfinition

IDE **I**ntegrated **D**evelopment **E**nvironment

3DFS **3** **D**imensional **F**ire **S**imulator

Unités

px pixel

cm centimètre

Constantes de la physiques

Constante gravitationnelle $G = 6.67408 \times 10^{-11}\ m^3 kg^{-1} s^{-2}$

Introduction générale

0.1 Introduction

Le feu est un phénomène visuel complexe et intéressant mais ne peut pas être étudier au niveau microscopique à cause de sa nature se qui le rend ue phénomène moins compréhensible et dangereux, mais malgré sa nature qui le rend intouchable et irréprochable, l'informatique a pu examiner son apparence et son comportement à travers des simulations de feu.

Les systèmes de particules sont une des méthodes les plus utilisées pour simuler un feu, et ceci est à l'aide de ces particules qui sont en générale des petits points qui prennent ensemble une forme ressemblant à une flamme.

Le noyau utilisé pour presque tous les systèmes de particules est l'algorithme de WILLIAM T. REEVES [11] qui a été créé en 1983 dans un article surnommé "Particle Systems A Technique for Modeling a Class of Fuzzy Objects", son travail a été utilisé dans le film Star Trek II : The Wrath of Kahn par la société de production de cinéma "Lucasfilm".

0.2 Définition du problème

L'objectif principal de tout simulation de feu c'est que cette dernière soit proche à la réalité par rapport à une vue humaine. Avoir une simulation de feu réel exige beaucoup de ressources matérielles informatiques ce qui rend la plupart des simulations de feu proche à la réalité mais qui ne s'exécutent pas en temps réel, et donc leurs utilisation se limite dans des films ou l'interactivité avec le feu est complètement absente.

Nous nous intéressons à une simulation d'un feu **réaliste, interactive et qui s'exécute en temps réel**, et donc la question qui se pose est comment peut-on avoir une simulation pareille avec **moins de ressources matérielle informatique**.

0.3 Objectifs

Pour l'avancement de l'étude et pour éclaircir les choses il été inévitable de fixer des objectifs de travail qui ont été comme un guide tout au long de notre progression.

les objectifs fixés sont :

- Avoir une simulation d'un feu qui est proche à la réalité.
- Avoir une simulation d'un feu qui s'exécute en temps réel et donc, une simulation optimisée en termes de temps de calcule et d'utilisation de ressources matérielles informatiques.
- Avoir une simulation d'un feu qui peut interagir avec les objets physique de son environnement, tout en restant en temps réel.
- Avoir une simulation d'un feu qui peut être modifiée par l'utilisateur, tout en restant en temps réel.
- Avoir une simulation d'un feu qui se base sur un système de particules.

0.4 Contributions

Rappelons nous du problème, en générale le problème rencontré au cours de notre étude des travaux précédents tournaient sur le coté réalisme et le coté interactivité dans les simulations de feu. Les contributions qui vont être proposées vont s'assurer d'avoir des meilleurs résultats visuels et une interactivité avec les objets de la scène tout en restant en temps réel.

Les contributions se résument en :

- Contribution 1 : notre première contribution concerne le réalisme, pour avoir un haut niveau de réalisme nous allons utiliser : des techniques tel que (**le mélange additif** et **les flocons de feu**) et des **textures ultra haute définition** que nous allons créer nous même.

- Contribution 2 : notre deuxième contribution concerne l'**interactivité** du système de particules avec les objets de la scène tout en restant en temps réel, pour cela nous allons créer des algorithmes qui consistent en la détection et l'évitement des collisions entre les particules et les objets de la scène.

0.5 Structure du mémoire

Ce manuscrit est organisé comme suit :

- **partie 1 - Généralités et état de l'art :** cette partie enveloppe les trois chapitres :

  Le chapitre 2 les Feux Le chapitre 2 présente un état de l'art sur les feux, il commence par des généralités sur les feux, après il passe à la classification des feux ensuite il introduit les méthodes de simulations de feu et pour finir il va être clôturé par une étude comparative entre les méthodes de simulation de feu.

Le chapitre 3- les systèmes de particules Quand à lui, il commence par des définitions concernant les systèmes de particules, après il passe à l'algorithme de base d'un système de particules qui est l'algorithme de Reeves et se clôture par les différents travaux utilisés pour les simulations de feu.

Le chapitre 4- l'interactivité et les systèmes de particules Le chapitre 4 présente les informations de base nécessaires à la compréhension du terme "interactivité" et sa relation avec notre travail.

- **partie 2 - Conception** cette partie présentera nos contributions et nos travaux pratiques :

  Le chapitre 5- Conception du système Le chapitre 5 présente la conception de notre système de particules, il commence par les objectifs de notre travail, après il passe à la présentation de notre modèle et termine par la présentation de nos contributions.

  Le chapitre 6- Étude de cas et implémentations Ce chapitre aborde les outils et les plateformes que nous avons utilisé durant notre travail, il introduit par la suite une description détaillée de la configuration matérielle et logicielle utilisée pour la création et pour les tests de notre outil intitulé : 3DFS. Le chapitre est terminé par la présentation de l'interface graphique de 3DFS, des tests et des comparaisons des résultats que nous avons eu avec les travaux précédents et des feux réels.

Chapitre 1

CHAPITRE I : Les Feux

1.1 Introduction

Un Feu, est en gros un phénomène qui a la faculté de produire de la lumière et de la chaleur et précisément, c'est la dégradation visible d'un corps par une réaction chimique exothermique d'oxydation appelée combustion.

Ce chapitre va détailler tout ce qui est en relation avec les feux dans la vie réelle : types, formes, couleurs, etc. ensuite, il va détailler les méthodes qui existent actuellement pour simuler des flammes proches à la réalité.

1.2 Définition d'un feu

Scientifiquement, une flamme est une réaction chimique qui produise une température d'au moins de 1500 k et généralement environ 2500 k, le feu est un ensemble de flammes. Une flamme peut avoir une épaisseur d'un ordre de $10^{-3}cm$ et un taux de production exothermique d'énergie par unité de volume d'environs $10^{8}w \cdot cm^{-3}$ [10].

1.3 Le Triangle de feu

Le feu peut être produit si on a les trois éléments essentiels suivant :

- L'énergie initiale.
- Le comburant : est une substance(composé ou matière) chimique qui a pour propriété de permettre la combustion d'un combustible.
- Le combustible : est une substance(composé ou matière) chimique.

Avec le tringle de feu schématisé dans la figure (Voir la figure 1-1), on peut voir des exemples sur les sources des trois éléments vue précédemment [10].

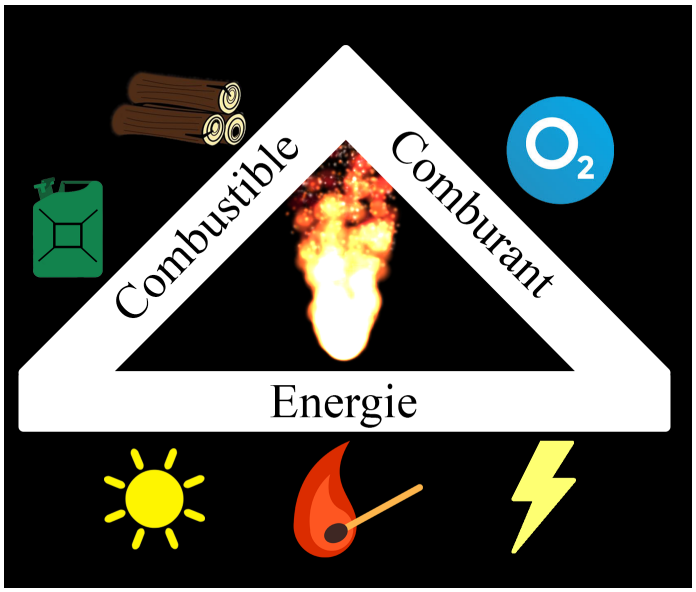


FIGURE 1-1 – Le triangle de feu.

1.4 Classification des feux

Il existe différentes classes de feu reconnues dans le monde, selon le degré du danger [6].

- Classe A : du combustible ordinaire (figure : 1-2) comme du bois, du papier, des vêtements ou du caoutchouc.

FIGURE 1-2 – Un Feu de camp.

- Classe B : comme les liquides inflammables (figure : 1-3) tel-que : l'essence, les goudrons, les graisses de pétrole, les huiles, les peintures à l'huile, l'alcool ou les solvants.

FIGURE 1-3 – Les cracheurs de feu.

- Classe C : les gaz inflammables (figure : 1-4) par exemple : le propane et le butane.

FIGURE 1-4 – Un Feu d'un gaz.

- Classe D : des Métaux combustibles (figure : 1-5) comme le magnésium, le titane, le zirconium, le sodium, le lithium ou le potassium.

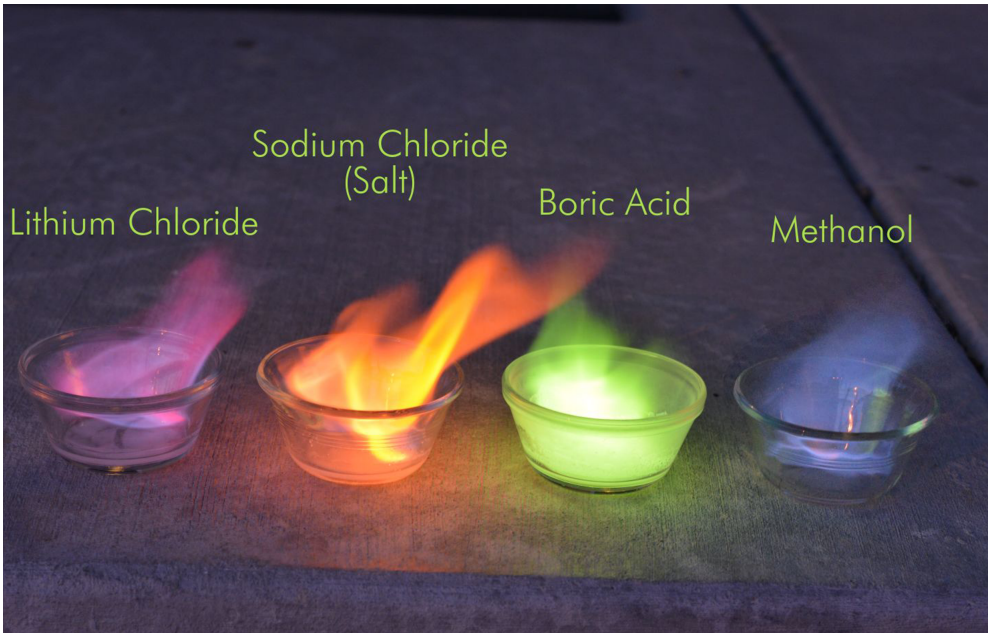


FIGURE 1-5 – Un Feu d'un métal.

- Classe E : un équipement électrique sous tension comme un ordinateur, moteurs, transformateurs.

- Classe F : des feux qui peuvent se générer des huiles de cuisson (figure : 1-6) et les graisses des animaux.

FIGURE 1-6 – L'huile de cuisson.

1.5 La simulation d'un feu

1.5.1 Introduction

Le feu a des propriétés physiques complexes et un mouvement aléatoire. Ceci est un grand défi qui vise les gens qui travaillent sur la simulation d'un feu réaliste comme dans notre cas.

Dans cette section nous allons voir les méthodes de simulation d'un feu, la différence entre elles et la méthode qui nous intéresse pour notre travail.

1.5.2 Les méthodes de simulation de feu

Pour simuler un feu réaliste, il existe plusieurs méthodes Wu ZhaoHui, Zhou Zhong et Wu Wei [15]. ont classifié les méthodes qui peuvent être la base d'une simulation d'un feu dans leurs 'Survey' surnommé 'Realistic Fire Simulation'. Les classes des méthodes de simulation de feu peuvent être :

- Méthode à base de Cartographie de texture ou 'Texture mapping'.
- Méthode à base de système de particules.
- Méthode à base physiques et mathématiques.
- Méthode à base automate cellulaire.
- Méthode à base de reconstruction d'image tomographique.

1.5.2.1 Méthode à base de cartographie de texture

Cette technique utilise des cartographies de texture pour modéliser un effet particulier. Au début des études de cette technique, de simples textures ont été prises d'un feu réel et attaché après à une géométrie d'un feu. Par la suite autres méthodes comme les BillBoard, BillBoard croisés, volume texture et texture vidéo ont fait leurs apparitions. La Méthode à base de Cartographie de texture à l'avantage dans la vitesse de calcul, elle peut modéliser le feu avec une mineur consommation de temps de calcul. Par contre, il est extrêmement difficile d'avoir un feu avec un mouvement réaliste. Ce qui rend cette méthode préférable juste pour les simulations de feu qui ne réclame pas un haut niveau de réalité.

1.5.2.2 Méthode à base physiques et mathématiques

Cette méthode utilise la mécanique des fluides numérique ou en anglais : the models of computational fluid dynamics (CFD), pour la simulation de feu. Une des principales idées de cette méthode est de prendre et/ou considéré le feu comme étant un fluide (liquide). Donc il est possible par la suite de modéliser et d'animer le feu avec la résolution des équations de Navier-Stokes (NS) qu'on peut trouver en mécanique des fluides. FedKiw [5] a fait un bon résumé pour les méthodes à base physiques pour la simulation de feu. La Méthode à base physiques et mathématiques peut être extrêmement exacte pour la description d'un mouvement d'un feu, mais la résolution des équations de Navier-Stokes (NS) est une opération très complexe et qui demande beaucoup de temps de calcul ce qui nous donne l'handicape de la très grande difficulté pour une réalisation d'une simulation d'un feu en temps réel.

1.5.2.3 Méthode à base automate cellulaire

Les automates cellulaires ont été introduits premièrement par Von Neumann en 1950. Les automates cellulaires consistent en une grille régulière de cellules chacune d'eux contient un état choisi parmi un ensemble fini et qui peut évoluer au cours du temps.

L'état d'une cellule au temps t+1 est dépendante de l'état au temps t d'un nombre fini de cellules appelé son voisinage. Les voisinages se varient : deux, quatre, huit connexités ou autres. Cette méthode est mieux adaptée pour les simples simulations à deux dimensions.

1.5.2.4 Méthode à base reconstruction d'images tomographiques

La Méthode à base reconstruction d'images tomographiques peut être considérée comme un processus ou la première étape à faire est de prendre des images simultanées d'un feu réel à plusieurs vues en d'autres termes donner de différentes positions au capteur. Subséquemment, il faut appliquer la technique de la reconstruction tomographique à partir des images déjà prises, afin d'avoir une simulation d'un feu

réel. Cette méthode a fait beaucoup de progrès durant ces dernières années, elle a l'avantage d'avoir des résultats réalistes comme elle prend ces données directement du monde réel afin de faire la reconstruction. Mais elle a l'inconvénient où le processus d'acquisition s'avère complexe et elle a un niveau de contrôle et d'interactivité très limité en d'autres termes on ne peut pas appliquer des modifications ou interagir avec la feu durant son animation.

1.5.3 Comparaison entre les méthodes de simulation d'un feu

Choisir la méthode la plus adaptée à notre travail a nécessité une étude comparative entre les méthodes de simulation de feu (Voir le tableau 1.1).

	Complexité temporelle	**Réalisme**	**Édition**	**Interactivité**
Cartographie de texture	Médiocre	Bas	Non	Aucune
Système de particules	Proportionnelle au nombre de particules	Moyen	Haute	moyenne
À base physiques	Grande	Haut	Moyenne	Haute
Automate cellulaire	Dépend des cellules	Moyen	Moyenne	Limité
Reconstruction tomographique	Ne peut être en temps réel	Très haut	Non	Non

TABLE 1.1 – Une étude comparative entre les méthodes de simulation d'un feu

1.5.4 Discutions de l'étude comparative

On déduit qu'il n'y a pas une méthode absolue pour une simulation d'un feu, car :

- La méthode qui ne consomme pas beaucoup de temps de calcul est la méthode : cartographie des texture.
- La méthode qui donne un très haut niveau de réalisme c'est la méthode : Reconstruction tomographique.
- La méthode qu'on peut utiliser pour éditer à tout moment n'importe quel paramètre au cours d'une animation d'un feu c'est les système de particules.
- La méthode qu'on peut utiliser pour avoir des interactivité entre le feu et les autres objets c'est la méthode : À base physiques et mathématiques.

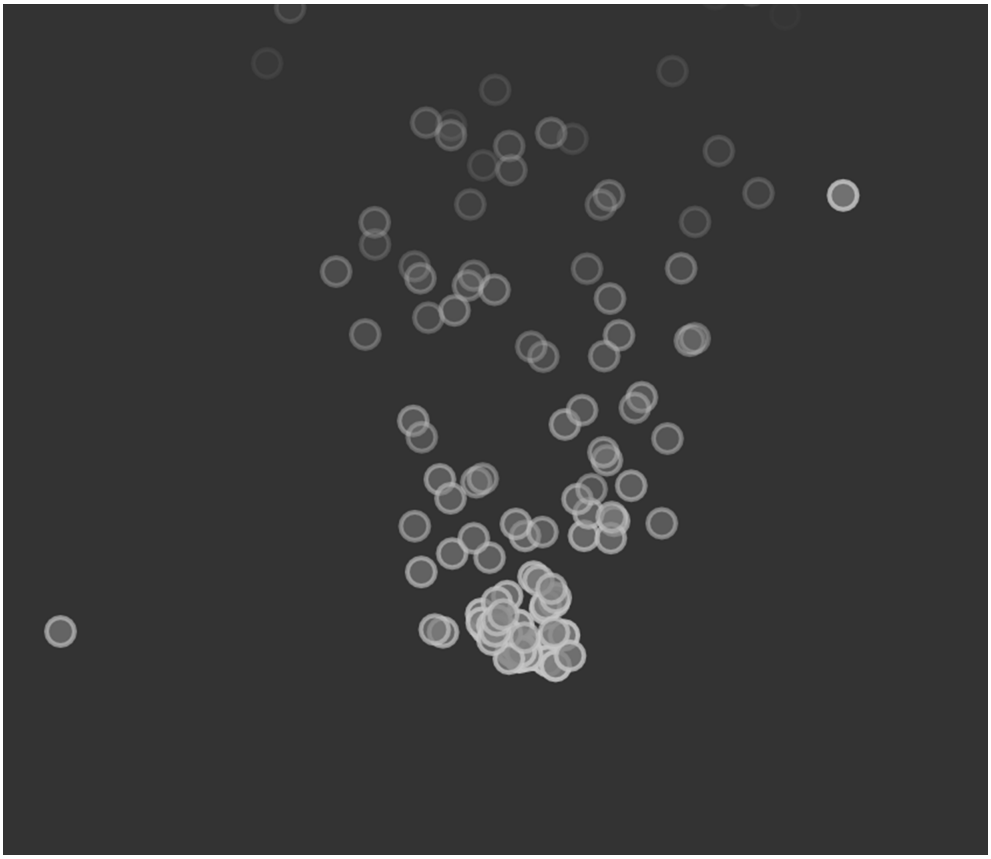
Donc pour avoir une méthode adaptée au maximum à notre travail il, a fallu utiliser une méthode combinée, dans la quelle on va utiliser des Cartographies de textures, un système de particule et des lois physiques et mathématiques.

1.6 Conclusion

Dans ce chapitre, on a pu voir tout ce qui est primordial concernant les simulations des feux, on a pu voir les classes de feu selon leurs degrés de danger, les différentes méthodes utilisées pour la simulation d'un feu et une petite comparaison entre ces méthodes. Une de ces méthodes n'a pas été détaillée volontairement, car nous avons spécifié tout un chapitre pour celle-ci, cette dernière va être la base de notre travail qui est un mélanges de plusieurs méthodes. La méthode visée est celle d'un *SP* qui va être entamé dans le chapitre qui suit.

Chapitre 2

CHAPITRE II : Les systèmes de particules



2.1 Introduction

Les particules sont parmi les objets les plus simples et les plus utilisées dans les simulations dynamiques. Par exemple les systèmes de simulation pour la génération d'une fumée, d'un feu, des nuages ou des liquides, même les tissus et les cheveux peuvent être simulés à l'aide d'un *SP* : système particules, la principale différence entre les divers systèmes de particules utilisés de nos jours réside dans les méthodes utilisées pour résoudre leurs équations de mouvement.

Ce chapitre va parler des *SP* et tout ce qui est en relation avec eux afin d'avoir une vue profonde sur la méthode utilisée dans notre travail.

2.2 Apparition et Découverte des systèmes de particules

Les SP ont fait leurs apparitions en 1961, dans l'explosion de vaisseau dans le jeu Spacewar qui s'exécutait sur l'ordinateur PDP-1. 17 ans plus tard en 1978 il y avait le jeu "Asteroids" dans lequel il y'avais l'explosion des vaisseau en plusieurs arrêts. 5 ans par la suite en 1983 WILLIAM T. REEVES [11] a publié un article surnommé "Particle Systems A Technique for Modeling a Class of Fuzzy Objects", son travail a été utilisé dans le film Star Trek II : The Wrath of Kahn par la société de production de cinéma "Lucasfilm".

2.3 Définition d'un SP

Un SP est une collection de nombreuses particules minuscules qui, ensemble, représentent un objet d'une forme irrégulière tel que le feu, l'eau ou la neige. Dans un SP, les particules sont générées, se déplacent et changent, et meurent en cours du temps. Le SP est en charge de la suppression, l'ajouter des particules et de la mise à jour des attributs qui appartiennent à chacune de ces dernières.

2.4 Une particule dans un SP

2.4.1 Définition

Une particule est un petit point, cercle, carré, cube ou une sphère minuscule qui est générée par un SP, qui évolue et qui disparaît selon sa durée de vie. Toutes les particules d'un SP agissent d'une manière cohérente et en masse, elles se déplacent et agissent en groupe et non pas individuellement, dans le but de simuler le mouvement d'un objet d'une forme irrégulière tel qu'un gaz ou un liquide.

2.4.2 Les paramètres d'une particule

Toutes les particules d'un système de particule ont les mêmes paramètres sauf qu'elles se diffèrent au niveau des valeurs propres à chacune, ces paramètres peuvent être :

- Une position : le centre x, y et z et l'orientation.
- Un mouvement : rotation, rapidité ou vitesse, accélération, etc.
- Une vélocité : c'est un vecteur qui représente une vitesse et une direction.
- Des forces : la gravité.
- Une couleur (RGBA) : qui peut aussi être une texture.
- Une forme(la géométrie) : un point, une arête, une sphère, un cube, un rectangle, etc.
- Une taille : petite, grande.
- Un volume : une densité, une masse.
- Une durée de vie.

2.5 Algorithme de base d'un SP

2.5.1 Introduction

Ils existent de nombreux algorithmes des SP, des algorithmes avec de différents paramètres, avec la prise en compte des collisions, avec la prise en compte des forces physiques ou ceux qui prennent en compte tous ça. Les algorithmes des SP se diffèrent selon les besoins. Dans cette section, on va présenter l'algorithme de base inventé par WILLIAM T. REEVES [11], cet algorithme est utilisé presque par tous les SP.

2.5.2 Algorithme

Algorithme 1 : Algorithme de base d'un SP

```
1  début
2      pour chaque Image de l'animation faire
3          De nouvelles particules sont générées et initialisées dans le système;
4          pour chaque Particule p faire
5              Mettre à jour les valeurs des attribues de p;
6              si La durée de vie de p > vie prescrite à p alors
7                  Supprimer p du système;
8              fin
9          fin
10         Faire le Rendu d'une image avec toutes les particules qui sont encore
           en vie;
11     fin
12 fin
```

2.6 Les différents travaux de simulation de feu en utilisant les systèmes de particules

1. Le travail de WILLIAM T. REEVES [11] en 1983 : une simulation d'une explosion basique crée pour le film Star Trek 2 : The Wrath of Khan, sont travail été référencé dans presque tous les travaux récents, sont travail est la base de tous les systèmes de particules de nos jours.

2. Le travail de Satoru Yoshida Et Tomoyulu Nishita [16] en 2000 : ils ont utilisé la technique des "metaballs", c'est un concept définie par Jim blinn [2] où chaque "ball" ou balle représente une fonction mathématique. Cette technique est utilisée dans le but de simuler la distribution de la densité de la fumée, en considérant la fumée comme une combinaison de "metaballs". Ils ont aussi pris en considération Les flux autour des obstacles. Le problème principale de cette technique c'est qu'elle est extrêmement couteuse en terme de complexité temporelle car chaque balle représente une fonction mathématiques.

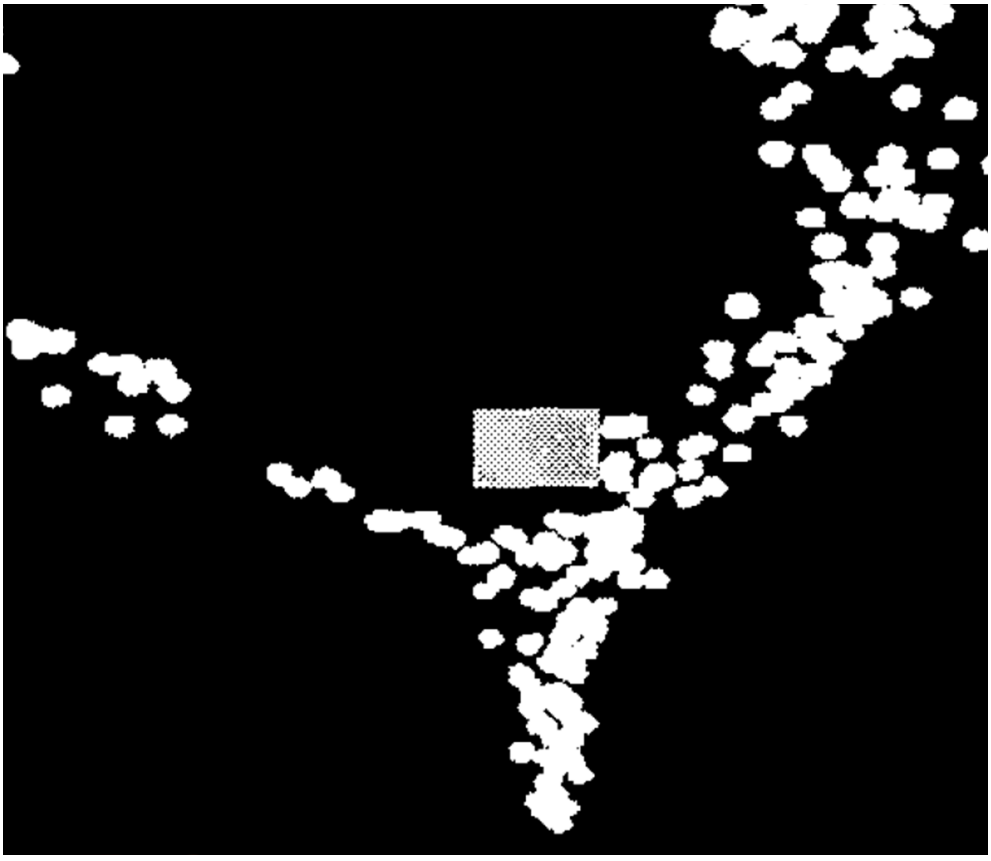


FIGURE 2-1 – Le mouvement des particules en évitant un obstacle [16].

3. Le travail de Sanandanan Somasekaran [14] en 2005 : il a fait une implémentation d'une simulation d'un feu plus ou moins proche à la réalité en utilisant un système de particules.

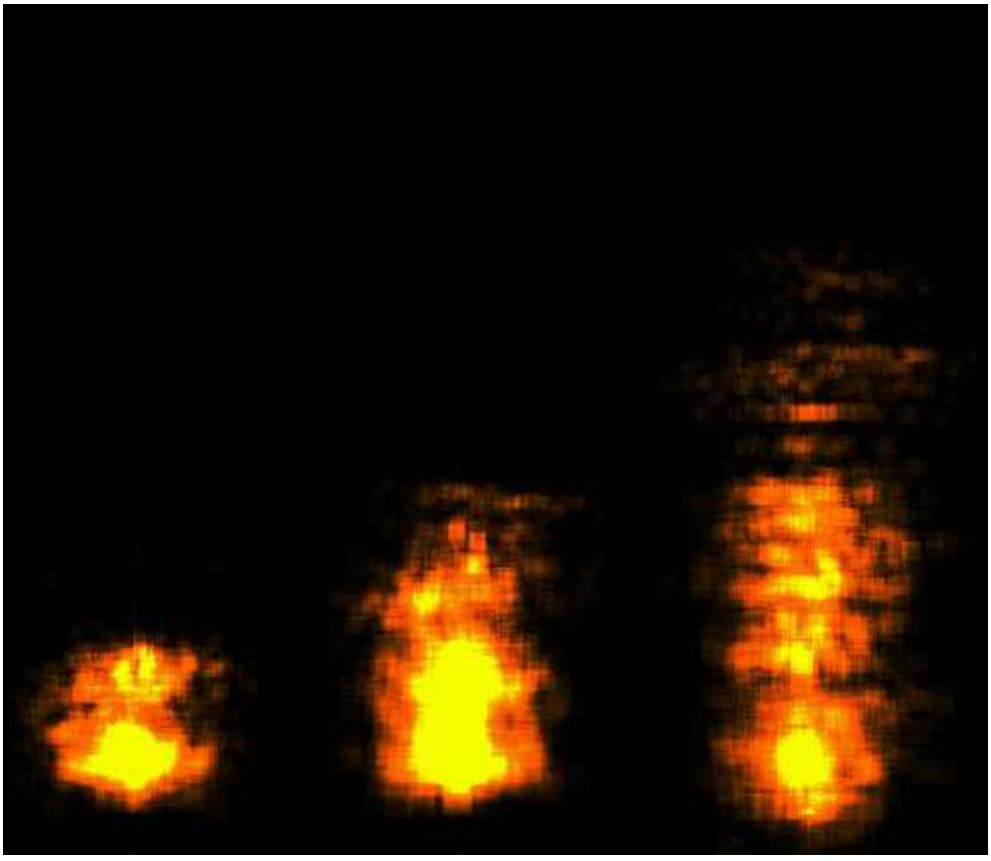


FIGURE 2-2 – Résultat d'une simulation de feu avec différentes hauteurs en utilisant un *SP* [14].

4. Le travail de C.H.Perry et de R.W.Picard [9] en 1994 : Ils ont présenté une méthode dans laquelle ils ont combiner un système de particules pour un feu avec un modèle de propagation de feu. cette méthode fournit le contrôle de la gravité et le contrôle du vent en parallèle avec la prise en charge de la déformation de la géométrie d'un objet après avoir subi une brûlure. Ceci augmente de plus en plus les interactions naturelles du feu avec l'environnement.

FIGURE 2-3 – Résultat d'une simulation de feu d'un bougie en utilisant un *SP* [9].

5. Le travail de Knut Erik Samuel Rødal et de Geir Storli [12] en 2006, ils ont réalisé une simulation et une visualisation d'un feu à base physique à temps réel en utilisant les GPUs, et ils se sont basés sur un système de particules.

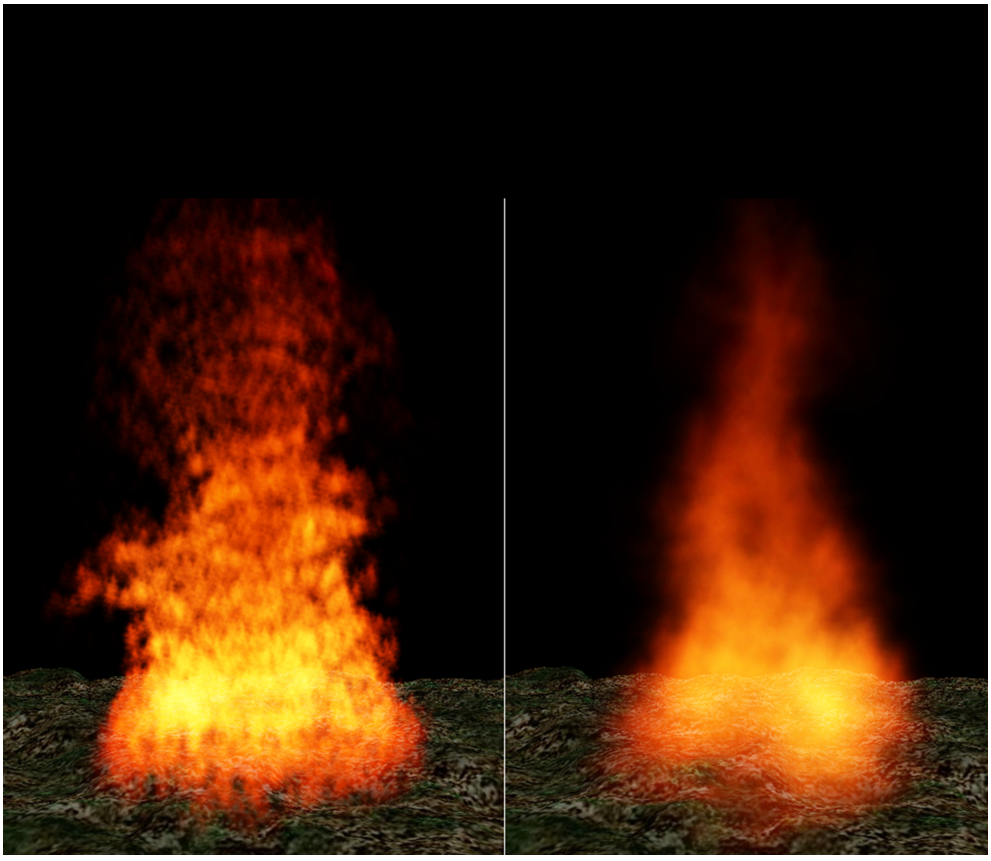


FIGURE 2-4 – Un *SP* avec 16384 particules (à gauche) et avec 256 particules (à droite) : [12].

6. Encore plus de travaux ont été réaliser et qui visèrent les simulations réalistes du feu mais qui été destinés à être utiliser dans de différents domaines tel que les films ou les applications de la réalité virtuelle, prenant l'exemple de ZHOU et al [17] en 2006 qui se basèrent sur les travaux de [8],[7] et [1], ZHOU et al ont introduit un modèle de simulation d'incendie basé sur un SP . Le système de particules de feu été implémenter par VRML en anglais "Virtual Reality Modeling Language" qui est un langage de description d'univers virtuels en 3 dimensions.

   Ou encore l'exemple de Horvath C. et al [4] en 2009 qui se basèrent sur les travaux [3], [5] et [13], Horvath C. et al ont proposer une nouvelle combinaison de simulation de grille de particules avec des simulations de raffinement très fines et sous GPU. ils ont proposé également un rendu volumique simple. leurs

résultats peuvent être intégrés dans des films de haute résolution. Le problème avec leur travail c'est que le feu n'est pas interactive, et la simulation n'est pas en temps réel et donc leur travail est destiner à être utiliser juste dans les filmes.

2.7 Conclusion

Dans ce chapitre, on a vu comment chercheurs ont procédés pour simuler les feux. Un de nos objectifs est d'avoir un feu réaliste qui doit être interactif, on doit avoir non seulement un système de particule classique, mais un *SP* avec des interactions de type particules-objets, c'est-à-dire des interactions entre lui et entre l'environnement. Voilà pourquoi on va voir l'interactivité et ça relation avec les systèmes de particules dans le chapitre qui suit.

Chapitre 3

CHAPITRE III : L'interactivité et les systèmes de particules

3.1 Introduction

Dans la réalité, quand on place un objet prés d'un feu, ce dernier va se transformer, être brûlé ou il peut rester comme il est, cela tout dépend de son type de matériel : bois, fer, plastique, papier, etc. Ce qui nous intéresse dans notre travail, c'est le comportement de notre système de particules ou bien notre feu par rapport aux autres objets, c'est-à-dire répondre à la question de comment est ce que nos particules peuvent elle éviter un objet dans la scène et pour cela, on va prendre les travaux relatifs les plus intéressants, de les résumer et d'essayer de les améliorer en créant une méthode qui répond à nos besoins.

3.2 Interactivité avec l'environnement

Les systèmes de particules doivent pouvoir interagir avec leurs environnements, principalement composés d'objets solides. Quand une particule entre en collision avec un objet, il devrait arrêter son comportement actuel et réagir à la surface de l'objet. Quand une collision se produit, la particule doit prendre une des taches suivantes :

- La particule change ses propriétés ou lance un nouveau système de particules, comme le cas d'une simulation d'un feu.
- La particule se colle à la surface de l'objet, comme dans le cas d'une simulation de la neige.
- La particule rebondit à la surface de l'objet, comme dans le cas d'une simulation de l'eau.
- La particule meurt.

La chose commune dans tous ces cas est la détection d'une collision. Une fois cette dernière est faite, les différentes réactions dépendent de ce qu'on veut avoir comme simulation (neige, pluie, feu, etc.). La Complexité de la détection des collisions dépend de la complexité de l'environnement et plus précisément des objets.

3.3 Évitement d'une collision

La détection d'une collision n'est pas une tâche évidente car nous ne savons pas précisément où la particule sera au prochain pas de temps, et donc nous ne savons pas si cela atteindra la surface de l'objet. La raison pour laquelle nous ne savons pas où la particule sera au prochain pas de temps est que la mise à jour des mouvements particules sont basées sur l'aléatoire. Et donc la solution est d'approximer la position future de la particule à partir des données que nous avons déjà dans le système de particule :

- la position actuelle de la particule.
- la vélocité actuelle de la particule.

Après avoir la position actuelle et la future position approximée, l'évitement d'une collision est définit par rapport au système, dans notre cas, c'est un feu et donc on peut par exemple faire glisser les particules sur la surface de l'objet (la manipulation est faite selon le type de la simulation).

3.4 Collision avec une sphère

Une sphère est un volume fermé. La condition d'appartenance à une sphère est juste définie par la distance entre un point et le centre de celle-ci, si cette distance est inférieure au rayon c'est-à-dire que le point appartient à la sphère, si elle est supérieure le point est en dehors de la sphère . Étant donné que la position actuelle d'une particule ne peut jamais être à l'intérieur de la sphère, il suffit de tester la position future de la particule et déterminer si elle est à l'intérieur. Cependant, il peut arriver que la prédiction de la position future n'est pas assez précise, et que la particule est piégée à l'intérieur de la sphère. Si la particule meurt au contact de l'objet, ce n'est pas un gros problème, mais s'il est supposé rebondir ou glisser tout autour de la circonférence de la sphère, la situation devient compliquée. La solution est de tester en plus si la position actuelle de la particule est à l'intérieur de la sphère, et si c'est le cas, il est possible de forcer la vélocité de cette particule afin de sortir à l'extérieur de la sphère et revenir aux tests de collisions par la suite.

3.5 Collision avec des objets complexes

Certains objets ne sont pas définis comme des primitives géométriques, et donc il n'ont pas une fonction mathématique qui leurs définies, ce sont plutôt des ensembles de facettes à quatre ou à trois sommets. Pour détecter une collision avec ce genre d'objet, nous devons tester individuellement chacune de ces facettes qui forment l'objet : cela peut être très coûteux en terme de complexité temporelle.

3.6 Conclusion

Avec ce chapitre, nous avons pu résumer l'interactivité et ça relation avec les systèmes de particules, on a eu une vue globale sur les collisions et sur comment se comportent les systèmes de particules pour éviter les collisions.

La partie état de l'art se termine avec cette conclusion et nous allons à présent passer à notre travail personnel qui sera entamer dans les chapitres qui suivent.

Chapitre 4

CHAPITRE IV : Conception du système

4.1 Introduction

L'objectif de ce projet est de créer un feu réaliste en temps réel. Ceci implique la mise en place d'un programme de simulation informatique qui va être utilisé pour étudier et modéliser la dynamique et l'esthétique du feu pour atteindre des résultats qui paraissent réalistes et qui peuvent se dérouler en temps réel et pour cela, on a eu recours aux systèmes de particules.

4.1.1 Pourquoi une approche qui se base sur les systèmes de particules ?

La modélisation de la dynamique du feu à l'aide de systèmes de particules nous permet de manipuler tous les facteurs qui concernent un feu : la forme, la luminescence, l'opacité, la couleur et surtout le comportement.

4.1.2 Quelles sont les classes de feux que nous avons simulé ?

D'abord rappelons nous des classes de feu connues vues dans 1.4 au chapitre des feux, on a :

- La classe A : on peut avoir un feu avec du combustible ordinaire comme le bois.
- La classe B : on peut avoir un feu avec des liquides inflammables comme l'alcool.
- La classe C : on peut avoir un feu avec du Gaz inflammable comme le propane.
- La classe D : on peut avoir un feu avec un métal inflammable comme le lithium.
- La classe E : on peut avoir un feu à cause d'un équipement électrique sous tension comme un moteurs.
- La classe F : on peut avoir un feu avec les huiles de cuisson, et ce sont les plus dangereux.

On s'est intéressé aux deux classes A et C, car elles se différent en plusieurs facteurs comme la forme, la couleur et le comportement des flammes. On a visé les classes qui

ont de différents facteurs afin de pouvoir mettre on œuvre tous ce que notre système peut et ne peut pas faire concernant les simulations des feux.

4.2 Objectifs

Pour s'assurer qu'on va obtenir de bons résultats en ce qui concerne une simulation d'un feu, on a fixé des objectifs ou des taches qui ont été comme un guide dans notre travail.

4.2.1 Schémas des objectifs

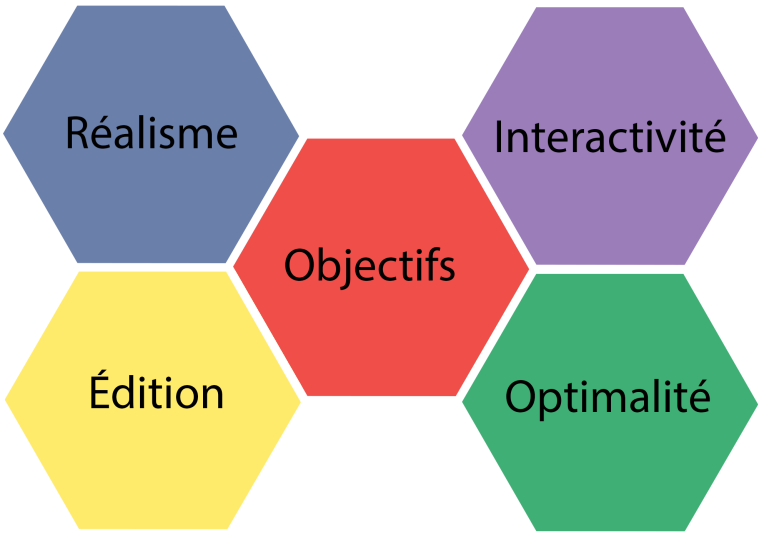


FIGURE 4-1 – Les objectifs de notre travail.

4.2.1.1 Le réalisme

C'est le fait d'avoir une simulation d'un feu proche à la réalité par rapport au niveau visuel d'un être humain.

4.2.1.2 L'édition

C'est le fait de pouvoir manipuler le système de particules (Le feu) en temps réel en d'autres termes, c'est de pouvoir modifier et/ou supprimer toute sorte de paramètres (Taille, couleur, durée de vie, etc.) en temps réel.

4.2.1.3 L'Interactivité

Il y a plusieurs types d'interactivité (Objet-objet, humain-système, etc.) ce qui nous intéresse dans notre simulation, c'est l'interactivité du système de particules avec les objets de l'environnement ou de la scène 3D, car en réalité les flammes interagissent avec tout son environnement.

4.2.1.4 L'optimalité

C'est tout simplement le système de particules le plus optimal possible, en ce qui concerne la complexité temporelle et l'utilisation des ressources tels que les GPUs.

4.3 Notre modèle

Notre modèle est basé sur l'algorithme de Reeves, avec l'ajout de plusieurs autres techniques et fonctionnalités qui ont fait de notre simulation, une simulation encore plus proche à la réalité.

Dans cette section, on va détailler les parties essentielles de notre travail qui vont participer à l'obtention d'une bonne simulation d'un feu.

4.3.1 Schémas de notre système

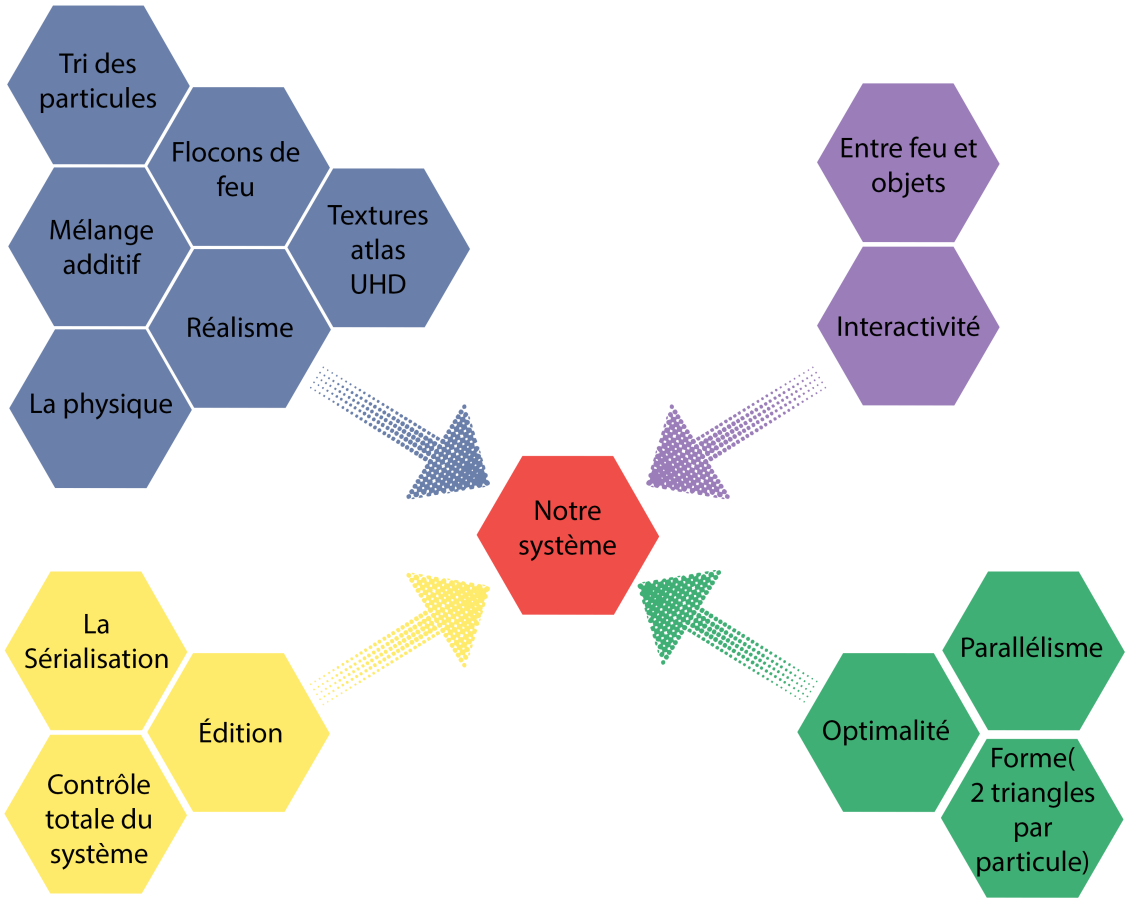


FIGURE 4-2 – Le schémas de notre système

4.3.1.1 Réalisme

1. La physique :

   (a) On a eu recours à la physique pour le calcul des forces qui sont appliquées sur nos particules, une de ces forces est la gravité qui peut être représentée par un vecteur avec une direction qui commence du haut et qui se dirige vers le bas, sauf que dans notre cas, la gravité va au sens inverse en d'autres termes du bas vers le haut (Voir la figure 4-3).

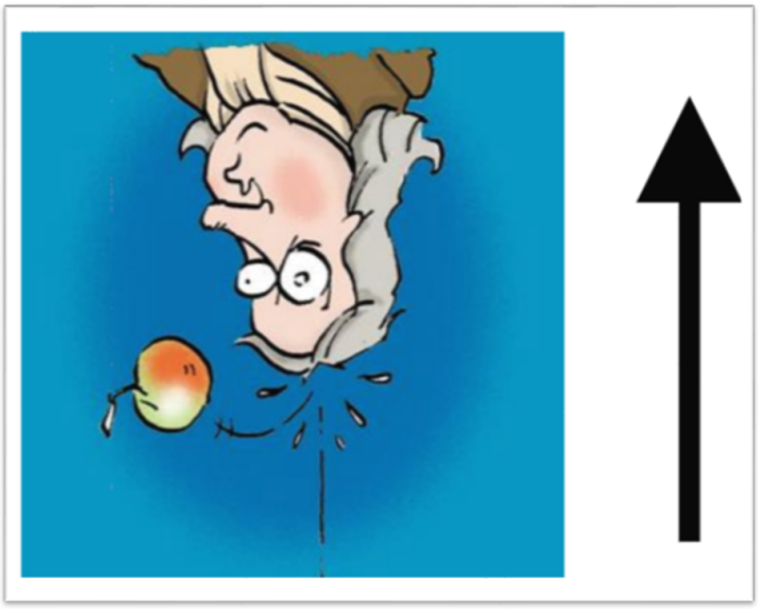


FIGURE 4-3 – La gravité inversée

   (b) On a aussi eu recours à la physique pour le calcul des forces répulsives et des forces attractives pour la détection et l'évitement des collisions entre le feu et les objets de la scène.

2. Le Mélange additif :

   (a) Le mélange additif ou bien "The additive blending" est une technique qui consiste à additionner toutes les parties des particules qui sont positionnées en parallèle par rapport à l'observateur (La caméra), et donc de plus il y a des particules positionnées en parallèle de plus le pixel à calculer converge vers le blanc (Voir la figure 4-4).

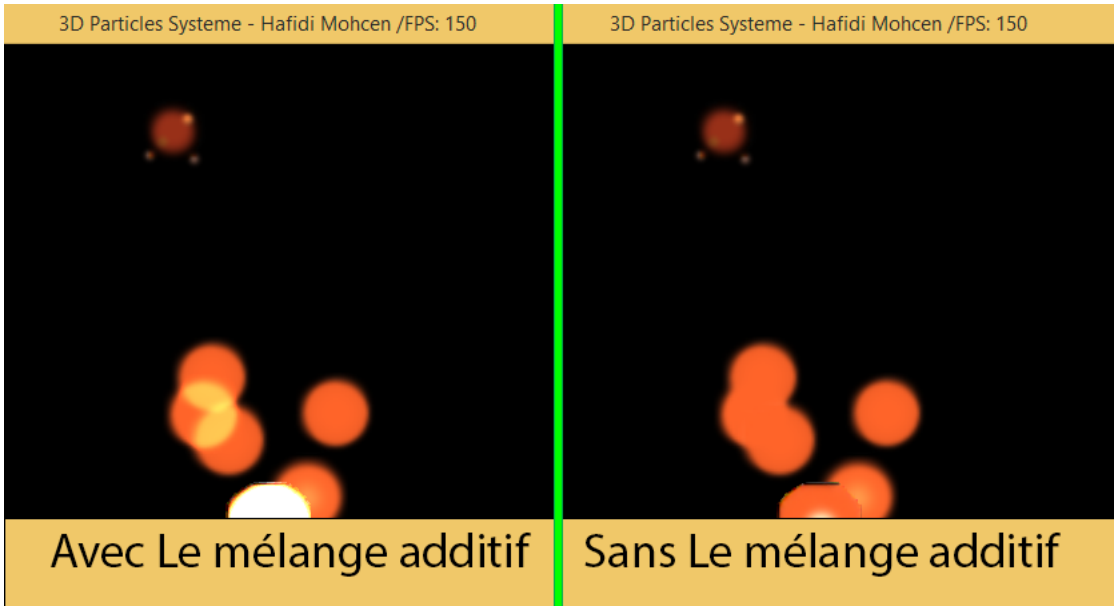


FIGURE 4-4 – Avec et Sans Le mélange additif

3. Textures atlas UHD :

   (a) On peut définir une texture atlas comme une liste qui contient plusieurs textures, On a créer des textures atlas pour éviter le chargement de plusieurs textures dans les "shaders", car ceci va ralentir notre animation et va diminuer le FPS.

(b) Les textures atlas Ultra Haute Définition dans notre cas sont des textures de définition : 8 192 x 8 192 et de 16 384 x 16 384 et de résolution égale à : 300 ppp, ces textures sont crées pour stocker les différents états et formes de nos particules (Voir la figure 4-5).

FIGURE 4-5 – Un exemple d'une de nos textures UHD, qui est utilisée pour un feu et sa fumée en même temps

4. Les Flocons de feu :

   (a) Les flocons de feu ou bien "flakes of fire" sont les petites particules que relâche un feu de classe A, D ou E (Voir la figure 4-6).

FIGURE 4-6 – Les Flocons de feu dans La réalité

L'implémentation des flocons de feu a été réalisée en spécifiant une zone spéciale dans quelques unes de nos textures.(Voir la figure 4-7).

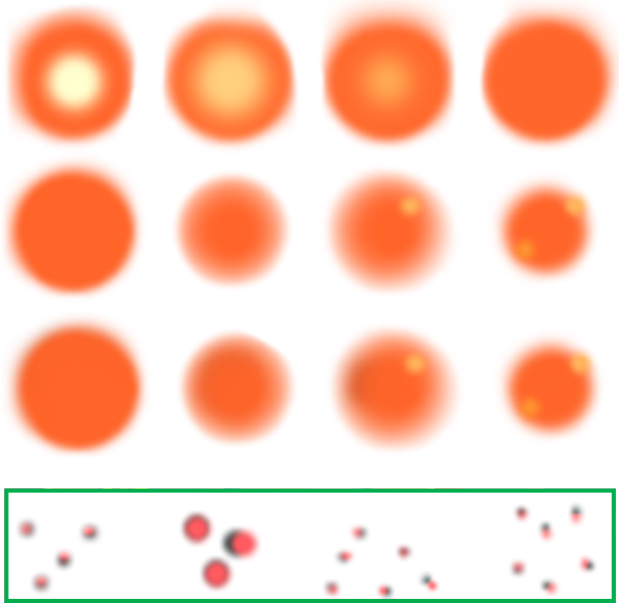


FIGURE 4-7 – Les Flocons de feu dans une de nos textures

5. Tri des particules :

   (a) Le tri des particules, c'est de faire en sorte que la particule la plus proche de la caméra est celle qui apparaît au premier lieu. Ceci est la solution du problème suivant : imaginons qu'on a plusieurs émetteurs de particules et que ces derniers sont positionnés en parallèle par rapport à la caméra dans ce cas le système peut afficher les particules qui sont en arrière au premier lieu et bien sur on veut pas que cela arrive, du coup la solution été tous simplement de trier les particules en fonction de leurs positions et de leurs distances avec la caméra.

4.3.1.2 Édition

1. Le contrôle totale du système : Dans notre système on a pu créer une interface graphique qui aide à manipuler le système de particules et/ou de créer des système de particules au choix. Ceci implique le changement des paramètres des systèmes de particules (Texture, taille, forme des particules, forme des émetteurs, durée de vie des particules, volume englobant, etc.), de l'environnement, des modèles d'éclairements, etc. et tout ça est bien sur peut être fait en temps réel.

2. La sérialisation : On utilisant la sérialisation, on a pu arriver à sauvegarder les états de nos objets dans des fichiers d'une extension de ".ser", afin de les charger par la suite après avoir relancer l'application. Cela nous a permis de sauvegarder l'état de l'interface graphique ainsi que la configuration de la scène 3D (Positions, orientations et tailles des objets).

4.3.1.3 Interactivité

1. Entre Feu et objets : D'abord, pour être dans la bonne piste, l'interactivité est un mot assez large, il y a plusieurs types d'interactivité comme on a vu dans 4.2.1.3. Ce qui nous intéresse dans notre simulation, c'est l'interactivité du système de particules avec les objets de l'environnement.

Pour cela, il a fallu que nos particules évitent les objets qui sont à leurs portés, et ceci exige la détection et l'évitement des collisions.

(a) **la détection de collisions :** pour pouvoir détecter si un objet est dans un champ de notre système de particules, ou plus précisément si un objet est a la porté d'une particule, on va calculer la distance entre cet objet et toutes les particules.

(b) **l'évitement de collisions :**

- Après avoir su qu'un objet est en collision avec une particule, cette dernière doit donc l'éviter dans la prochaine image de l'animation. Pour cela, on a utilisé les forces qui se définissent par la loi de l'attraction universelle de Newton (Voir la figure 4-8).

$$F_{A/B} = G\frac{M_A M_B}{d^2}$$

FIGURE 4-8 – La loi de l'attraction universelle de Newton,

A et B sont deux corps de masses respectives M_A et M_B, d est la distance qui les sépare, $F_{A/B}$ est La force exercée sur le corps B par le corps A, G est la constante gravitationnelle.

- Dans notre travail nous avons eu besoin d'un évitement d'une collision, en d'autres termes nous avons eu besoin d'une force répulsive et non pas d'une force attractive et donc pour avoir cela on va tout simplement multiplier la formule de Newton par (-1) afin d'inverser les directions des vecteurs des forces d'attraction pour avoir à la fin des forces de répulsion (Voir la figure 4-9).

$$F_{A/B} = G\frac{M_A M_B}{d^2}(-1)$$

FIGURE 4-9 – La loi d'attraction universelle de Newton multipliée par (-1)

Remarque : On va voir ceci en détailles à 4.3.2.

4.3.1.4 Optimalité

1. Parallélisme : Afin d'optimiser notre modèle on a utilisé la puissance des GPUs. Avec l'utilisation de la technique qui s'appelle « Instanced Rendering » on a pu envoyer une liste de particules à la carte graphique au lieu d'une seule et cela nous a permis d'arriver à générer 10 000 particules par seconde avec 25 images par seconde.

2. Forme(2 triangles par particule) :

   (a) Pour optimiser notre modèle on choisi d'utiliser les forme carrées pour la géométrie d'une particule, on précise que cette forme est seulement relative à la géométrie de l'objet "particule", Ceci nous a permit d'utiliser seulement deux triangles par particule et donc notre rendu va être seulement avec : 6 vertex, 2 normales et 12 paramètres de textures pour chaque particule.

   (b) Si les particules ont des formes carrées, comment peut-on par exemple avoir une forme circulaire dans l'animation ? Et bien, la réponse est la transparence. On peut définir n'importe quelle forme dans un carré et mettre les pixels qui ne sont pas inclues dans notre forme comme pixel transparent(Qui a un canal alpha égale à 0). Après dans le shader on va ignorer le pixel de la particule qui a un niveau de transparence égale à 0 en utilisant les lignes de codes suivants (Voir le code 4.1).

```
1 if(finalColor.a == 0){
2     discard;
3
4 }
```

Code 4.1 – L'utilisation de l'invisibilité

4.3.2 Notre algorithme

Algorithme 2 : Notre algorithme

Données : encore_en_vie : Variable booléenne

```
1  début
2      Créer et initialiser une liste liste_Ps de N Systèmes de particules();
3      Créer et initialiser une liste liste_OI de K objets de type objet interactif();
4      pour chaque Image de l'animation faire
5          pour chaque Système de particules Ps dans liste_Ps faire
6              De nouvelles particules sont générées et initialisées();
7              pour chaque Particule p dans Ps faire
8                  si Détecter les collisions(p, liste_OI) = faux alors
9                      encore_en_vie ← mettre à jour p();
10                 sinon
11                     encore_en_vie ← Éviter les collisions et mettre à jour p(p,
                        liste_OI);
12                 fin
13                 si P n'est pas dans le volume englobant de Ps() alors
14                     encore_en_vie ← faux;
15                 fin
16                 si encore_en_vie = faux alors
17                     Supprimer p de Ps;
18                 fin
19             fin
20         fin
21         Faire le Rendu d'une image avec toutes les particules
            vivantes(liste_Ps);
22     fin
23 fin
```

4.3.2.1 Explications et détails de notre algorithme

- Ligne 2 :

  Créer et initialiser une liste_Ps de N Systèmes de particules() ;

  On va créer une liste d'éléments qui sont de type "Système de particules", qui va contenir un ou plusieurs systèmes de particules, on va aussi les initialiser, nous précisant que l'initialisation concerne les systèmes de particules et non pas les particules. Les paramètres qui vont être initialisés pour chaque système de particules sont :

  - **Texture :** on a utilisé ce qu'on appelle les "Texture atlas" c'est une simple image qui contient plusieurs textures, avec cette "texture atlas" on définit la forme et les couleurs d'une particule à un temps donné.
  - **Position :** c'est un vecteur de trois nombres réels, avec lequel on définit la position de l'émetteur dans la scène 3D.
  - **Nombre de particules émises par seconde :** c'est le nombre de particules qu'on génère chaque seconde.
  - **Vitesse :** la vitesse avec laquelle les particules bougent.
  - **Conforme de la gravité :** qui joue le rôle d'un modificateur de gravité.
  - **Durée de vie :** représente la durée de vie maximale que peut prendre une particule.
  - **Taille :** représente la taille maximale que peut prendre une particule.
  - **Volume englobant :** c'est le cube qui limite le système de particules.
  - **Forme :** représente la forme du système de particule à sa naissance et non pas les particules elles-mêmes les formes possibles sont : ligne, cercle ou torus, Disque, rectangle ou parallélépipède, un seule point d'émission, et pour finir la forme qui a été utilisée pour simuler la classe de feu(gaz), notre systeme reçoit N nombre de sommets et retourne N positions autour d'un cercle et bien sur ces positions sont utilisées pour l'émission des particules (Voir la figure 4-10).

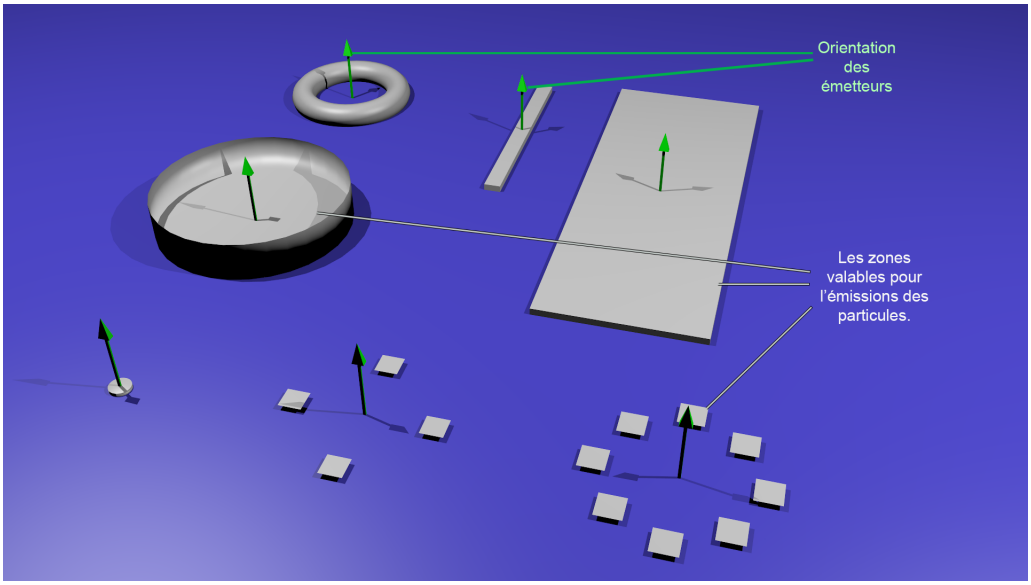


FIGURE 4-10 – Les formes des émetteurs (On a utilisé Blender pour la modélisation de ce modèle et le lancer de rayons qui existe en Blender pour faire le rendu de cette image).

- Ligne 3 :

  Créer et initialiser une liste liste_OI de K objets de type objet interactif() ;

  Le type "objet interactif" est le type des objets qui vont interagir avec les systèmes de particules (Les feux). On va créer une liste d'éléments qui sont de type "objet interactif" et initialiser leurs : positions, tailles et orientations.

- Ligne 6 :

  De nouvelles particules sont générées et initialisées() ;

  Il ne faut surtout pas confondre "initialisation des systèmes de particules" et "initialisation des particules". Le rôle de cette fonction est la création et l'initialisation des particules d'un système de particules donné.

  - **Initialisation et génération d'une particule p pour un système de particule Ps :**

    les paramètres qui vont être initialisés pour chaque particule sont : position, vélocité, gravité, durée de vie, rotation, taille, texture.

- Position : va prendre une position (x, y et z) aléatoire et il faut qu'elle reste dans la zone valable qui est définie par le paramètre : "forme" propre à l'émetteur du système de particule Ps (Déjà initialisée à la "Ligne 2").
- vélocité : c'est un vecteur qui est généré d'une manière aléatoire ET limité dans un cône (Voir la figure 4-11)

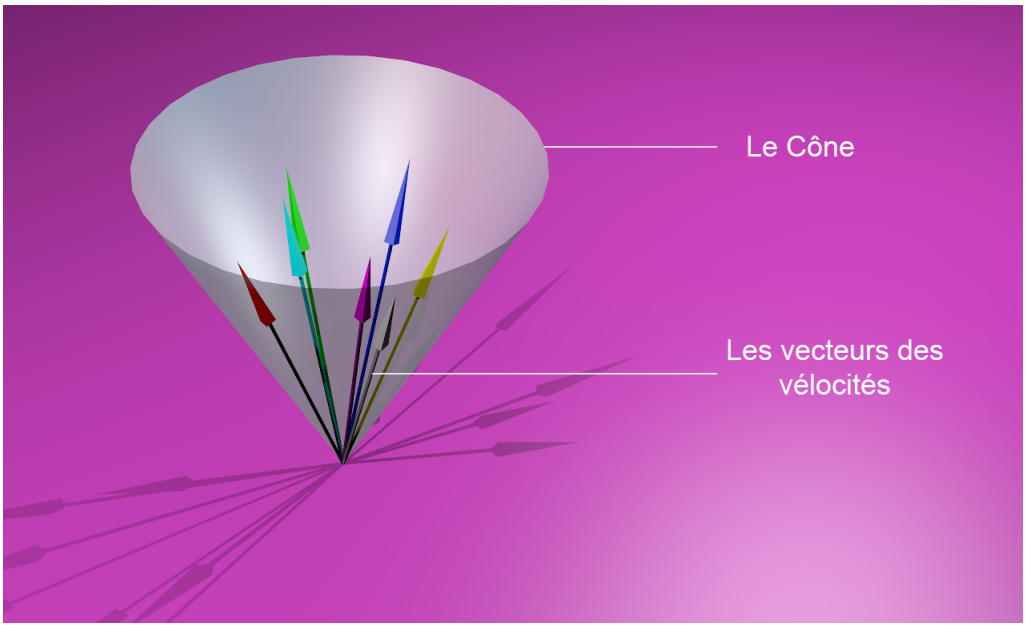


FIGURE 4-11 – Un cône avec des vecteurs de vélocités générés aléatoirement (On a utilisé Blender pour la modélisation de ce modèle et le lancer de rayons qui existe en Blender pour faire le rendue de cette image).

- gravité : prend la valeur du paramètre "Conforme de la gravité" de Ps.
- durée de vie : prend une valeur aléatoire limité entre le paramètre "Durée de vie" qui appartient à Ps et la somme ("Durée de vie" + duréeErreur), duréeErreur est une constante.
- rotation : ce paramètre est un peux spécial dans cette fonction, il va prendre 0, mais après il va être calculé de telle sorte que la particule va toujours être orientée vers la caméra.
- taille : prend une valeur aléatoire limitée entre le paramètre "Taille"

qui appartient à Ps et la somme (”Taille” + tailleErreur), tailleErreur est une constante.

- Ligne 8 :

  Détecter les collisions(p, liste_OI) ;

  – Cette fonction prend en paramètres une particule P et une liste d'objets interactifs liste_OI, elle va voir si P est en collision avec un ou plusieurs de ces objets, pour cela elle va tester si la ”distance avec les rayons” (Voir la figure 4-12) est inférieure à la somme des deux rayons (Rayon de la sphère et celui de la particule), si elle est inférieure, c'est qu'il y a une collision (Voir la figure 4-13) et donc la fonction marque cet objet et continue de tester les autres objets et à la fin elle retourne ”vrai”, car il y a au moins une collision avec la particule P.

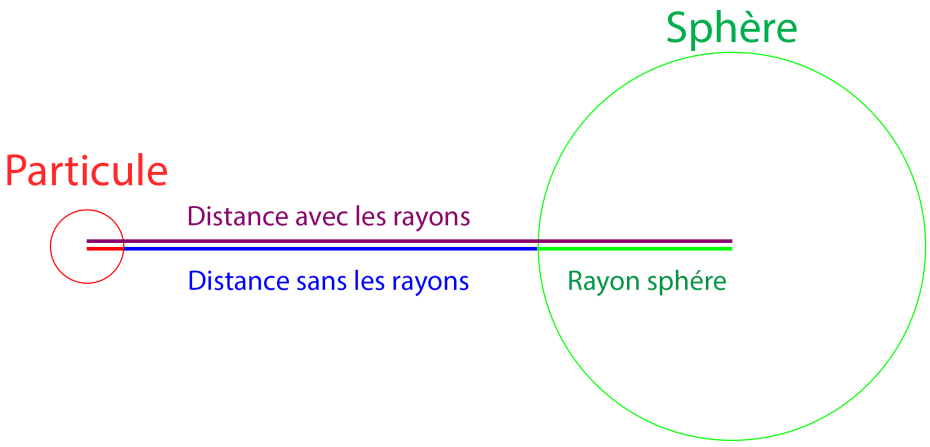


FIGURE 4-12 – La distance avec les rayons de la sphère et de la particule.

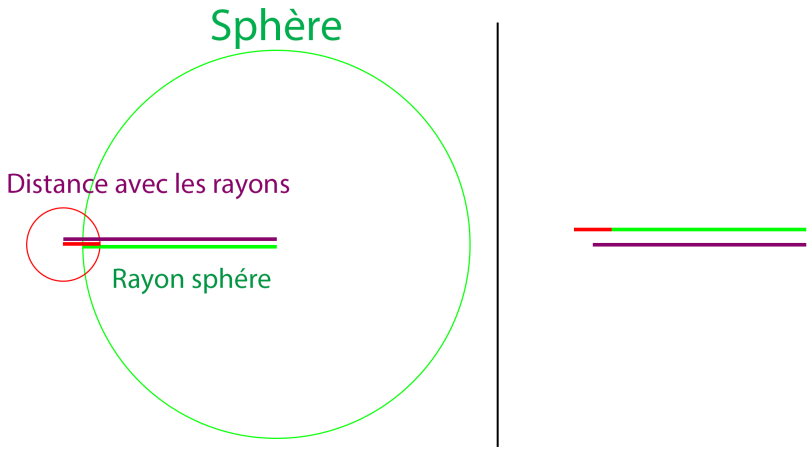


FIGURE 4-13 – Le cas d'une collision.

– Le code source de la fonction qui détecte les collisions : (Voir le Code 4.2)

```
1 boolean collisionDetection( Vector3f pPosition, List
    interactiveObjects) {
2   boolean result = false;
3   float distanceP_Object = 0;
4   float sommeRadiuses    = 0;
5   for(int i =0; i< interactiveObjects.size(); i++) {
6     distanceP_Object = distance3D_Points(pPosition,
    interactiveObjects(i).getPosition);
7     sommeRadiuses =(ParticulesRadius + interactiveObjects(i).
    getObjectRadius);
8     if(distanceP_Object - sommeRadiuses < 0)
9     {
10      interactiveObjects(i).setInFireRange(true);
11      result = true;
12    }
13  }
14  return result;
15 }
```

Code 4.2 – Détection de collisions.

– la fonction qui calcule la distance entre deux points dans un espace 3d est définit par le code source suivant : voir **??**.

```
1 float distance3D_Points(Vector3f p1, Vector3f p2){
2   return(float) Math.sqrt(
3                 Math.pow(p1.x - p2.x, 2) +
4                 Math.pow(p1.y - p2.y, 2) +
5                 Math.pow(p1.z - p2.z, 2)
6                 );
7 }
```

Code 4.3 – Calcule de distance entre deux points dans un espace 3D.

- Ligne 9 :

Mettre à jour $p()$;

– Cette fonction fait la mise à jour des paramètres de p sans se préoccuper des objets interactifs car il n'existe aucune collision avec eux, et donc la particule va subir seulement à la force de la gravité.

– Comment on fait la mise à jour des paramètres d'une particule ? Pour répondre à cette question, on va commencer par incrémenter la Vélocité et l'additionnée avec la position de la particule, après on va faire la mise à jour des coordonnées de texture selon la durée de vie de la particule ensuite, on va décrémenter la durée de vie de cette dernière.

– Calcul de la Vélocité : la vélocité est un vecteur de 3 réels, pour l'incrémenter, on va utiliser la seule force qui affecte la particule et c'est la gravité, et donc on va faire : (Voir le code 4.4)

```
1 velocity.y += GRAVITE_constante * modificateurDeGravite *
    temps_pris_pour_generer_une_image();
```

Code 4.4 – Incrémentation de la Vélocité.

- Ligne 11 :

  Éviter les collisions et mettre à jour p(p, liste_OI) ;

  – Cette fonction prend en paramètres une particule P et une liste d'objets interactifs liste_OI, va passer par la procédure d'évitement de collision et va mettre à jour p.

  – Une fois une ou plusieurs collisions sont détectées, on va devoir les évitées.

  – Dans la fonction "Détecter les collisions()" nous avant marquer tous les objets qui sont en collisions avec la particule P, et donc on sait qui sont les objets que la particule p devra évitée.

  – Pour éviter les objets interactifs marqués nous avons utiliser la loi de Newton (Voir 4.3.1.3) en apportant une petite modification qui est le remplacement de toute masse d'objet par 1 et donc on aura la formule : (Voir la figure 4-14)

$$F_{A/B} = G\frac{1}{d^2}(-1)$$

FIGURE 4-14 – La loi de Newton avec $M_i = 1$, i est l'indice d'un objet interactif ou d'une particule

Nous avons utiliser cette formule pour calculer la somme des forces répulsives qui proviennent des objets interactifs marqués, (Voir la figure 4-15).

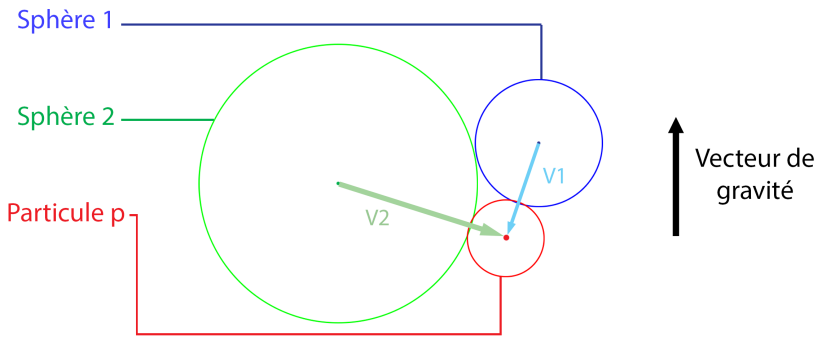


FIGURE 4-15 – Le cas d'une collision entre la particule p et deux objets interactifs.

– Comment calculer la somme des forces répulsives ? une force est tout simplement un vecteur de trois nombres réels.

  Supposant qu'on a une particule qui est en collision avec deux sphères (voir l'image 4-15), Dans ce cas il faut calculer pour chaque objet interactif un vecteur qui part du centre de ce dernier et qui se dirige vers le centre de la particule en utilisant la formule vue précédemment (Voir la figure 4-14), pour notre exemple ceci revient à calculer V1 et V2, ensuite on va faire la somme de tous ces vecteurs (V1 + V2 pour notre exemple), pour obtenir à la fin un seul vecteur sommeDesForces qui va représenter les forces qui doivent affecter la particule p.

– maintenant qu'on a le vecteur sommeDesForces, il nous reste le calcul de la distance carré d^2, cela va être effectuer en calculant la (taille du vecteur sommeDesForces)^2.

– Nous avons d^2 et nous avons le vecteur sommeDesForces, maintenant nous allons changer l'échelle du vecteur sommeDesForces en utilisant d^2 et le vecteur sommeDesForces pour personnaliser le vecteur de force finale qui va affecter la particule p, le changement d'échelle va être effectué comme suit : (Voir le code 4.5)

```
1 //La constante gravitationnelle.
2 float G = 6.67408; // ou autre
3 //Loi universelle de la gravitation
4 float F = G / d_carre;
5 //changement d'echelle
6 sommeDesForces.x = sommeDesForces.x * F;
7 sommeDesForces.y = sommeDesForces.y * F;
8 sommeDesForces.z = sommeDesForces.z * F;
```

Code 4.5 – changement d'échelle pour sommeDesForces.

– Et pour finir nous allons ajouter le vecteur final des forces au vecteur de vélocité et refaire exactement ce qu'on a fait dans la Ligne 9 de la fonction "Mettre à jour $p()$;".

- Ligne 13 :

  P n'est pas dans le volume englobant de Ps() ;

  – Pour cela, il faut seulement tester si une particule appartient à un parallélépipède personnalisé qui va être le volume englobant pour un système de particule donné (Voir la figure 4-16).

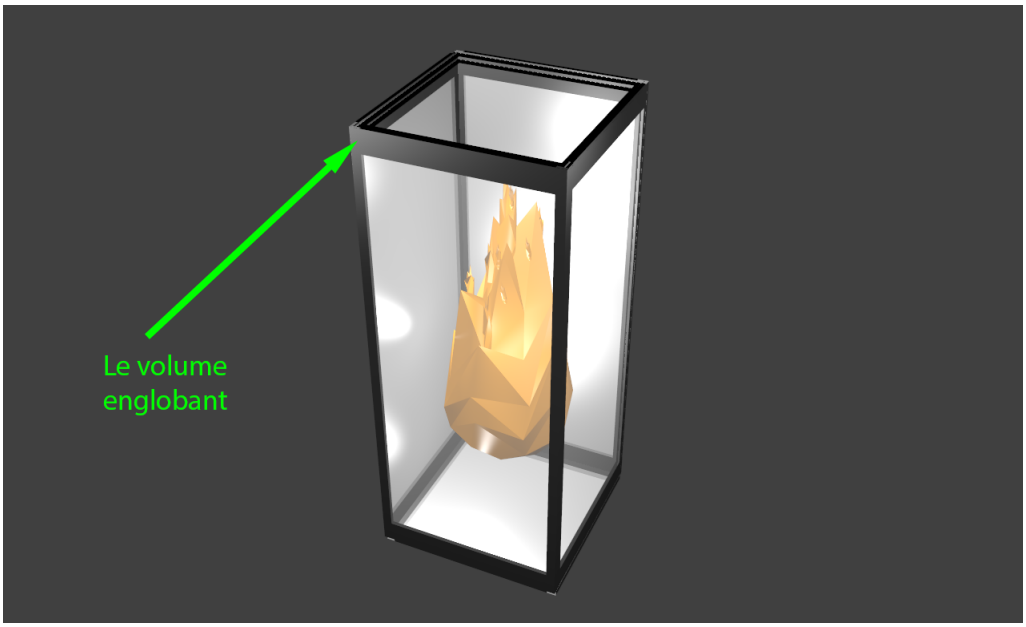


FIGURE 4-16 – Le volume englobant (On a utilisé Blender pour la modélisation de ce modèle et le lancer de rayons qui existe en Blender pour faire le rendue de cette image).

  – Cette fonction retourne vrai si la particule est dans le parallélépipède et faux si non.

- Ligne 21 :

  Faire le Rendu d'une image avec toutes les particules vivantes(liste_Ps) ;

  - La façon la plus moderne pour faire le rendu des modèles en utilisant OpenGL est avec les VAO (En anglais : Vertex Array Object) et les VBO (En anglais : Vertex Buffer Object). Nous avons utiliser la technique qui s'appelle "Instanced Rendering" avec les VAO et les VBO afin d'envoyer plusieurs particules à la carte graphique au lieu d'une seule.
  - VBO : Est un tableau ou on peut stocker des valeurs réelles.
  - VAO : Est une liste de VBO, et donc une liste qui contient plusieurs tableaux de valeurs réelles.
  - Nous avant utiliser les VAO et les VBO pour stocker les informations relatives aux particules : positions, coordonnées de textures, normales, etc.

4.3.3 Comparaison avec l'algorithme de Reeves

Pour comparer les deux algorithmes, on va utiliser un simple tableau, (Voir le tableau 4.1), où :

- "sp1" est un système de particules.
- "p" est une particule qui existe chez "sp1".
- "em" est un émetteur qui appartient à "sp1".

	L'Algorithme de Reeves	**Notre algorithme**
Forme de *p*	Spécifier par le *SP*	spécifier par le *SP*
Couleur(s) de *p*	Une seule couleur *RGB*	Texture UHD
Taille de *p*	Spécifier par le *SP*	spécifier par le *SP*
durée de vie de *p*	Spécifier par le *SP*	spécifier par le *SP*
Forme de *em*	Spécifier par le *SP*	spécifier par le *SP*
Interactivité (objets-particules)	non implémentée	implémentée
Mélange additif	implémenté	implémenté
Flocons de Feu	non implémenté	implémenté
Édition (en temps réel)	non	oui
Instanced rendering	non implémenté	implémenté

TABLE 4.1 – Comparaison de notre algorithme avec celui de Reeves

4.4 Contributions

Rappelons nous d'abord du problème, en générale le problème rencontré au cours de notre étude des travaux précédents tournaient sur le coté réalisme et le coté interactivité dans les simulations de feu. Le but des contributions qui ont été proposées est de s'assurer d'avoir des meilleurs résultats visuels et une interactivité du SP avec les objets de la scène, tout en restant en temps réel.

Nos contributions à propos des problèmes rencontrés se résument comme suit :

- Contribution 1 : notre première contribution concerne le réalisme, pour avoir un plus haut niveau de réalisme nous avons utiliser : des techniques tel que (le mélange additif et les flocons de feu) et des textures de ultra haute définition que nous avons créer (Voir les section précédentes de ce chapitres).

- Contribution 2 : notre deuxième contribution concerne l'interactivité du système de particules avec les objets de la scène tout en restant en temps réel, pour cela nous avons créer un algorithme qui consiste en la détection et l'évitement des collisions entre les particules et les objets de la scène (Voir les section précédentes de ce chapitres).

Les sections qui précédent ont détaillé au maximum les solutions que nous avons pu avoir pour la résolution des problèmes de réalisme et d'interactivité dans notre simulation de feu.

4.5 Conclusion

Ce chapitre a pu résumer la plupart des points essentiels de notre travail d'abord, nous avons vu les deux classes de feu les plus intéressantes et qui ont été choisies parmi les classes vues précédemment et qui étaient : classe A avec du combustible ordinaire comme le bois et classe C avec du Gaz inflammable comme le propane, ensuite nous passer par les objectifs de notre travail qui tournaient sur : le réalisme, l'édition, l'Interactivité et l'optimalité, par la suite nous avons résumé notre travail en utilisant le schéma du système et un algorithme, et pour finir nous avons fait une comparaison entre les deux algorithmes : le nôtre et celui de Reeves.

Dans le chapitre suivant nous allons sortir du théorique et nous allons entrer dans la pratique avec l'étude de cas et l'implémentions de notre travail, en d'autres termes nous allons voir tout ce qui a une relation avec l'application qui se base sur les informations que nous avons vu dans ce chapitre.

Chapitre 5

CHAPITRE V : Étude de cas et implémentations

5.1 Introduction

Après avoir décrit notre modèle dans le chapitre qui précède, nous allons à présent conclure le mémoire avec ce dernier chapitre.

Nous allons tout d'abord voir l'implémentation de notre modèle ensuite présenter l'interface de notre application et finir avec une exposition des tests obtenus par 3DFS.

5.2 Outils utilisés

Cette section va présenter les outils et les plates-formes qui ont été utilisés pour la création de 3DFS.

5.2.1 Langages de programmation

- **Java :** Nous avons choisi d'utiliser le langage de programmation Java comme un langage principal pour notre application, car :
  - C'est un langage qui assure une portabilité excellente d'où le slogan de java : "Écrivez une fois, exécutez partout", on écrit nos codes une fois et on les exécutes dans n'importe quel système d'exploitation.
  - C'est un langage de programmation de haut niveau.
  - C'est un langage qui se repose sur les concepts de l'orienté objets.
  - C'est un langage qui permet l'accès direct au GPU, possible grâce à la bibliothèque graphique OpenGL.
  - C'est un langage qui a de nombreuses bibliothèques tierces.
- **GLSL :** Est l'abréviation de "OpenGL Shading Language", c'est un langage de programmation de shaders. Les shaders sont des programmes qui s'exécutent au niveau du GPU ce qui donne leur puissance et leur flexibilité dans le rendu 3D.

5.2.2 Bibliothèques

- **LWJGL :** LWJGL est une bibliothèque Java qui permet l'accès aux API natives utiles au développement des : applications graphiques (avec OpenGL), audio (avec OpenAL) et de calcul parallèle (ave OpenCL). Cet accès est direct et performant.

- **OpenGL :** OpenGL est une bibliothèque avec la quelle on exploites la carte graphique. Nous avons utiliser OpenGL, car :

  – OpenGl est compatible avec JAVA et peut être lier avec lui à travers la bibliothèque LWJGL.

5.2.3 Environnement de programmation

Integrated Development Environment (IDE) est un programme combinant un éditeur de texte, un compilateur, des outils de création automatique et souvent un débogueur. En Java, il existe plusieurs IDE tels que NetBeans (Sun), JCreator ou Eclipse (IBM). L'IDE que nous avons utilisé est Éclipse car c'est un IDE libre et qui parce-que nous l'avons utiliser avons.

Eclipse est un IDE et un projet écrit en Java, décliné et organisé en un ensemble de sous-projets de développements logiciels, Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation. L'avantage d'éclipse c'est qu'on peut ajouter des "plugins", ce qui n'est pas le cas pour d'autres IDE.

5.3 Logiciels et matériels utilisés

Dans cette section nous allons voir quelle été la configuration logicielle et matérielle utilisée pour la création de 3DFS et pour les tests (Voir le tableau 5.1).

Configuration logicielle et matérielle	
Système d'exploitation	Windows 8.1 Professionnel 64-bit
Processeur	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz
Processeur\ Nombre de cœurs	8
Processeur\ Génération	3éme
RAM	8 Go
Fabricant de l'ordinateur	ASUSTeK COMPUTER INC.
Carte graphique 1	Intel(R) HD Graphics 4000 (non utilisé pour le rendue)
Carte graphique 2	NVIDIA GeForce GT 740M (utilisé pour le rendue)
Carte graphique 2\ Vram	2047MB DDR3
Carte graphique 2\ largeur des bus	128 bit
Carte graphique 2\ OpenGL supporté	OpenGL 4.6
Carte graphique 2\ OpenCL supporté	OpenCL 1.2
Carte graphique 2\ Cuda supporté	Cuda 3.0
Carte graphique 2\ Cuda \ thread par bloque	1024
Carte graphique 2\ Taille max d'une image 2D supportée	16384px x 16384px

TABLE 5.1 – Notre configuration logicielle et matérielle.

5.4 Présentation de l'interface de l'application

L'application commence par une petite animation qui présente le logo et le nom de l'outil, ensuite une interface d'accueil va apparaitre, et enfin l'interface principal ou il y aura tous les traitement et toutes les animations pour notre systeme de particules.

5.4.1 Interface d'accueil

L'interface d'accueil (Voir la figure 5-1)) est une interface ou l'utilisateur peut choisir le mode de traitement, il y aura trois modes :

FIGURE 5-1 – L'interface d'accueil. (Cette figure est une capture d'écran prise de 3DFS)

- **Le mode "Zone libre" :** est un mode ou l'utilisateur peut créer des animations selon son choix et modifier tout sorte de paramètres. L'utilisateur peut par exemple créer de nouvelles textures par exemple une texture de pluie ou une texture de la neige, les importer dans l'outil et avoir par la suite d'autres formes et d'autres types de système de particules.

- **Les modes "Zone avec un feu de bois" et "Zone avec un feu de gaz" :** ces modes sont crées pour illustrer une simulation d'un feu de combustible de type .rsp bois de La classe A et gaz de La classe C vue dans (le chapitre 4). Au contraire du mode précédant ce mode n'autorise pas la modification de quelques paramètres.

5.4.2 Interface principale

L'interface principale (Voir la figure 5-2) ou l'interface des tests apparait juste après avoir choisi "le mode" dans l'interface d'accueil, L'interface principale est l'interface ou l'utilisateur peut **créer** et/ou **éditer** un système de particules dans le cas ou il aura choisi le mode "Zone libre" ou alors consulter et interagir avec des simulations déjà présentes dans le cas ou il aura choisi un des deux autres modes.

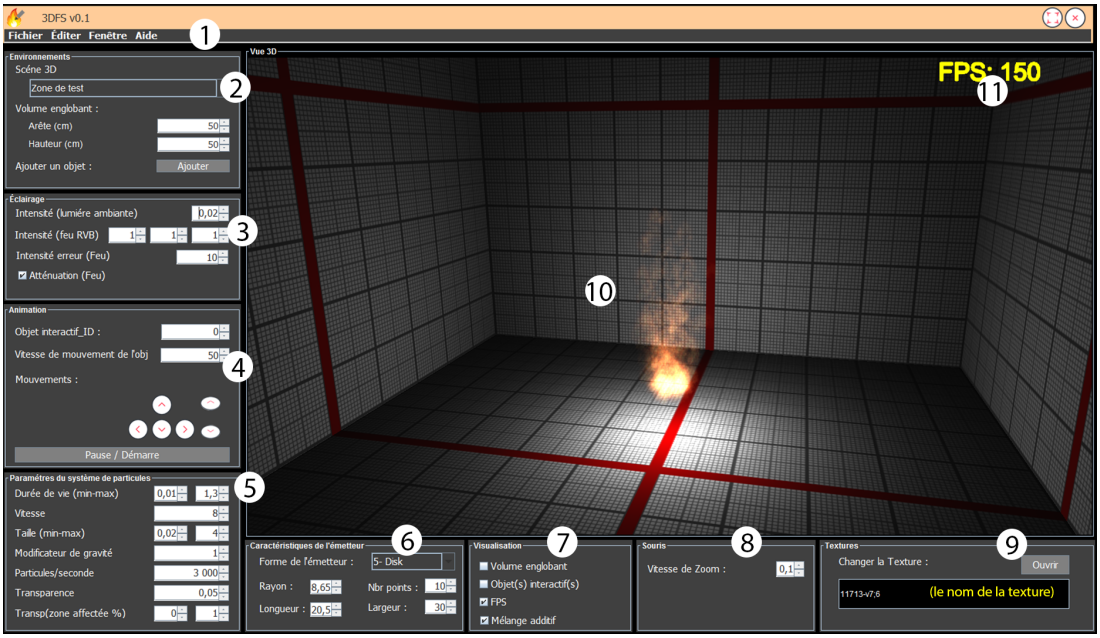


FIGURE 5-2 – L'interface principal. (Cette figure est une capture d'écran prise de 3DFS)

5.4.2.1 Composantes de l'interface principal

- **(1) La barre d'outils :** la barre d'outils (Voir la figure 5-3)donne l'accès à plusieurs fonctionnalités mais nous allons citer que les fonctionnalités les plus intéressantes :
  - Dans la barre "Fichier" \"Ouvrir" ou "Sauvegarder" : nous avons pu implémenter ces deux fonctionnalités qui servent à importer et à écrire des données depuis et dans des fichiers.ser en utilisant la sérialisation (vue en 4.3.1).
  - Dans la barre "Aide" \"à propos" : une fenêtre va apparaitre, cette dernière va contenir toutes les informations concernant 3DFS et son développeur.

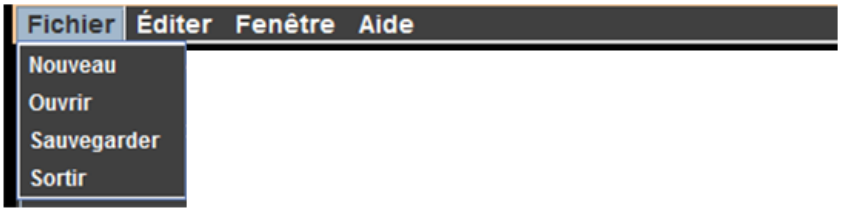


FIGURE 5-3 – La barre d'outils. (Cette figure est une capture d'écran prise de 3DFS.)

- **(2) Le panneau "Environnement" :** ce panneau contient les paramètres qui ont une relation avec l'environnement :
  - La Scéne 3D : peut être une boite, une forêt, une cuisine ou autre.
  - Le volume englobant : le parallélépipède qui limite le feu.
  - Ajouter un objet : sert à ajouter un nouveau objet interactif, qui peut évidement interagir avec le feu.

- **(3) Le panneau "Éclairage" :** ce panneau contient les paramètres qui ont une relation avec l'éclairement du système de particules et l'éclairement de la lumière ambiante :
  - Intensité (lumière ambiante) : peut être une boite, une foret, une cuisine ou autre.
  - Intensité erreur (Feu) : normalement l'intensité du feu est déjà calculée par le système mais nous avons ajouter ceci juste pour donner un contrôle totale du système à l'utilisateur. ce paramètre sert à ajouter une valeur à l'intensité actuelle du feu et mis à 0 si l'utilisateur est satisfait avec l'intensité actuelle du feu.
  - Atténuation (Feu) : se paramètre sert à activer ou a désactiver l'atténuation du feu.
- **(4) Le panneau "Animation" :** ce panneau contient les paramètres qui ont une relation avec l'animation dans la scéne :
  - Objet ID : est pour choisir quel objet à faire bouger.
  - Mouvements : des boutons qui servent à bouger un objet interactif.
  - Pause/Démarre : un bouton qui sert à mettre en pause le système de particules.
- **(5) Le panneau "Paramètres du système de particules" :** ce panneau contient les paramètres du système de particules vus dans le chapitre 4.
- **(6) Le panneau "Caractéristiques de l'émetteur" :** ce panneau contient les paramètres qui ont une relation avec l'émetteur du système de particules vue en 4.3.2.1.

- **(7) Le panneau "Visualisation" :** sert à visualiser ou à rendre invisible les objets qui sont dans la scéne.
- **(8) Le panneau "Souris" :** dans le quel on trouve les paramètres de la souris.
- **(9) Le panneau "Textures" :** dans le quel on peut importer des textures pour notre système de particules.
- **(10) Le panneau "Vue 3D" :** dans le quel se visualise la scéne 3d.
- **(11) Le FPS :** Affiche le nombre des images rendues par seconde.

5.5 Tests et comparaisons

Cette section va présenter les résultats de notre travail en utilisant des tests faites avec 3DFS. Pour l'illustration des tests on va utiliser des captures d'écrans et les comparer avec des images prisent de la réalité.

Dans le chapitre 5 (Voir 4) nous avons parlé des deux classes : la classe A avec du combustible ordinaire comme le bois et la classe C avec du Gaz inflammable comme le propane, dans cette section nos tests principaux vont être focalisés sur ces deux classes de feu.

5.5.1 Tests concernant le réalisme

- **Test 1 : Feu de camp**

  1. **Résultats :**

     – Vue rapprochée :

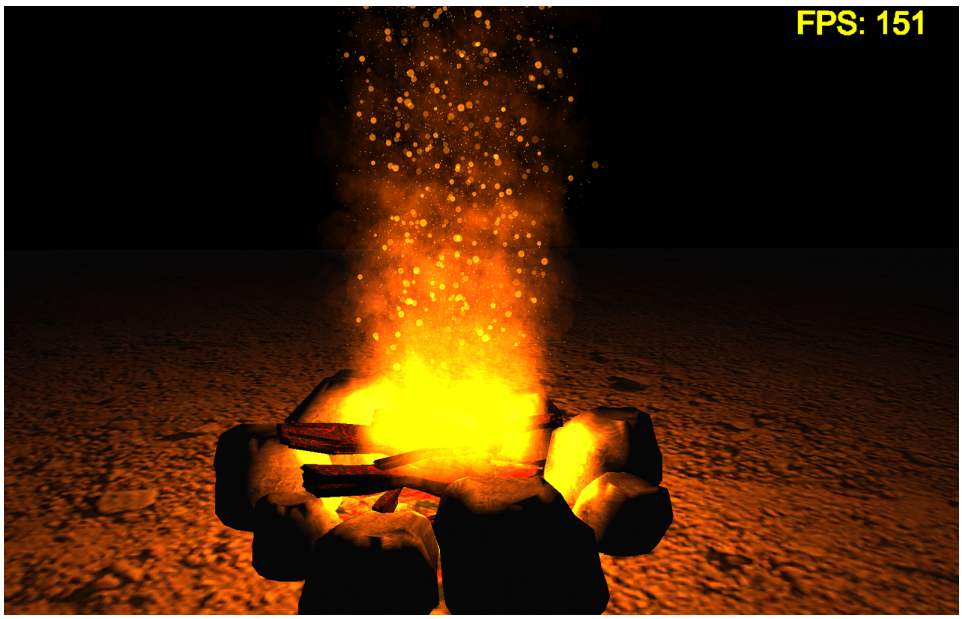


FIGURE 5-4 – Feu de camp, vue rapprochée. (Cette figure est une capture d'écran prise de 3DFS)

– Vue de loin :

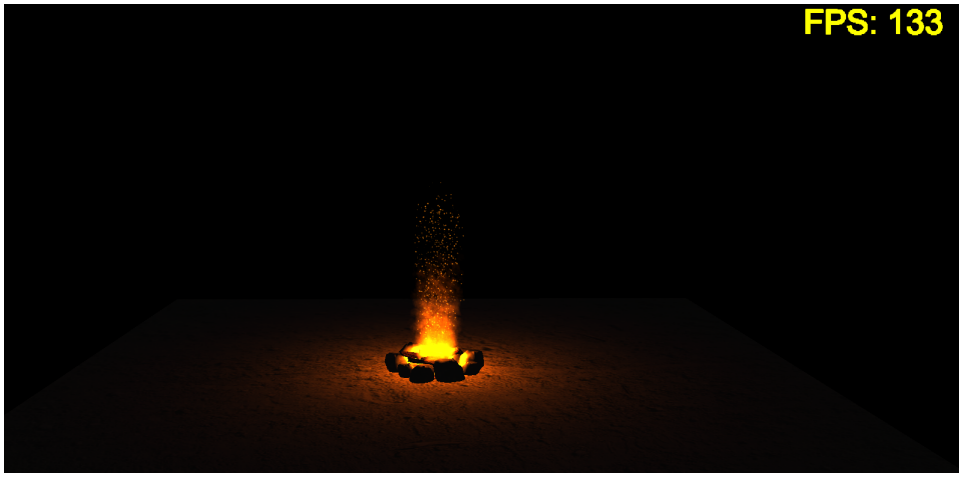


FIGURE 5-5 – Feu de camp, vue de loin. (Cette figure est une capture d'écran prise de 3DFS)

– Vue de face :

FIGURE 5-6 – Feu de camp, vue de face. (Cette figure est une capture d'écran prise de 3DFS)

2. **Description :** Nous avons pu obtenir les images ci-dessus (Voir les figures 5-4, 5-5 et 5-6) avec 3DFS et en utilisant la configuration du tableau (Voir le tableau 5.2) où :
   - "sp1" est un système de particules.
   - "p" est une particule qui existe chez "sp1".
   - "Ps" est l'ensemble de toutes les particules qui appartiennent à "sp1".
   - "em" est un émetteur qui appartient à "sp1".

	Feu de camp(test de réalisme)
FPS	114 de loin et 78 de prés
Formes des *Ps*	Pour chaque proportion de temps qui passe de nouvelles formes complexes sont affectées au *Ps*.
Couleurs des *Ps*	Gamme de couleurs entre rouge et jaune définis par une texture
Définition de la texture	8 192px × 8 192px
Résolution de la texture	300 ppp
Transparence des *Ps*	0,036 sur 1
Transparence (zone affectée) *Ps*	y = 0.75 sur 1 , x = 1 sur 1
Taille des *Ps*	Aléatoire entre : $[1; 2]$
Durée de vie des *Ps*	Aléatoire entre : $[0.5; 1.4]$
Forme de *em*	Disque de rayon 9,35cm
Vitesse	8.0
Modificateur de gravité	1.0
Nombre de "Ps" émisses par seconde	8 000
Interactivité (objets-particules)	oui
Mélange additif	Actif
Flocons de Feu	Présents
Édition (en temps réel)	Oui
Rendu par instance	Oui
La Fumée avec le feu	Non présente

TABLE 5.2 – La configuration pour un feu de camp

- **Test 2 : Feu d'une cuisinière à gaz**

  1. **Résultats :**

     – Vue de face :

FIGURE 5-7 – Feu de camp, vue rapprochée , la figure en haut représente un feu doux et celle du bas un feu fort (Cette figure est une capture d'écran prise de 3DFS)

– Vue de haut :

FIGURE 5-8 – Feu de camp, vue de loin , la figure en haut représente un feu doux et celle du bas un feu fort (Cette figure est une capture d'écran prise de 3DFS)

2. **Description :** Nous avons pu obtenir les images ci-dessus (Voir les figures 5-7 et 5-8) avec 3DFS et en utilisant la configuration du tableau (Voir le tableau 5.3) où :
   - "sp1" est un système de particules.
   - "p" est une particule qui existe chez "sp1".
   - "Ps" est l'ensemble de toutes les particules qui appartiennent à "sp1".
   - "em" est un émetteur qui appartient à "sp1".
   - "même" veut dire que c'est la même configuration pour un feu-doux.

	Feu d'une cuisinière à gaz (test de Réalisme)
FPS	127 de loin et 82 de prés
Formes des *Ps*	des Disques texturées
Couleurs des *Ps*	Gamme de couleurs entre blanc et bleu définis par une texture
Définition de la texture	8 192px × 8 192px
Résolution de la texture	300 ppp
Transparence des *Ps*	0,041 sur 1
Transparence (zone affectée)	y = 1 sur 1 , x = 1 sur 1
Taille des *Ps*	Fixe : 2.1
Durée de vie des *Ps*	Aléatoire entre : $[0.5; 0.8]$
Forme de *em*	N points dans un cercle de rayon = 8.85 et N = 40
Vitesse	8.0
Modificateur de gravité	0.4
Nombre de "Ps" émisses par seconde	10 000
Interactivité (objets-particules)	Un seul objet
Mélange additif	Actif
Flocons de Feu	Non présentes
Édition (en temps réel)	Oui
Rendu par instance	Oui
La Fumée avec le feu	non présente

TABLE 5.3 – La configuration pour un feu d'une cuisinière à gaz (test de Réalisme)

5.5.2 Tests concernant l'interactivité

- **Test 3 : test d'interactivité avec un mur de feu :**

  1. **Résultats :**

     – Vue de face :

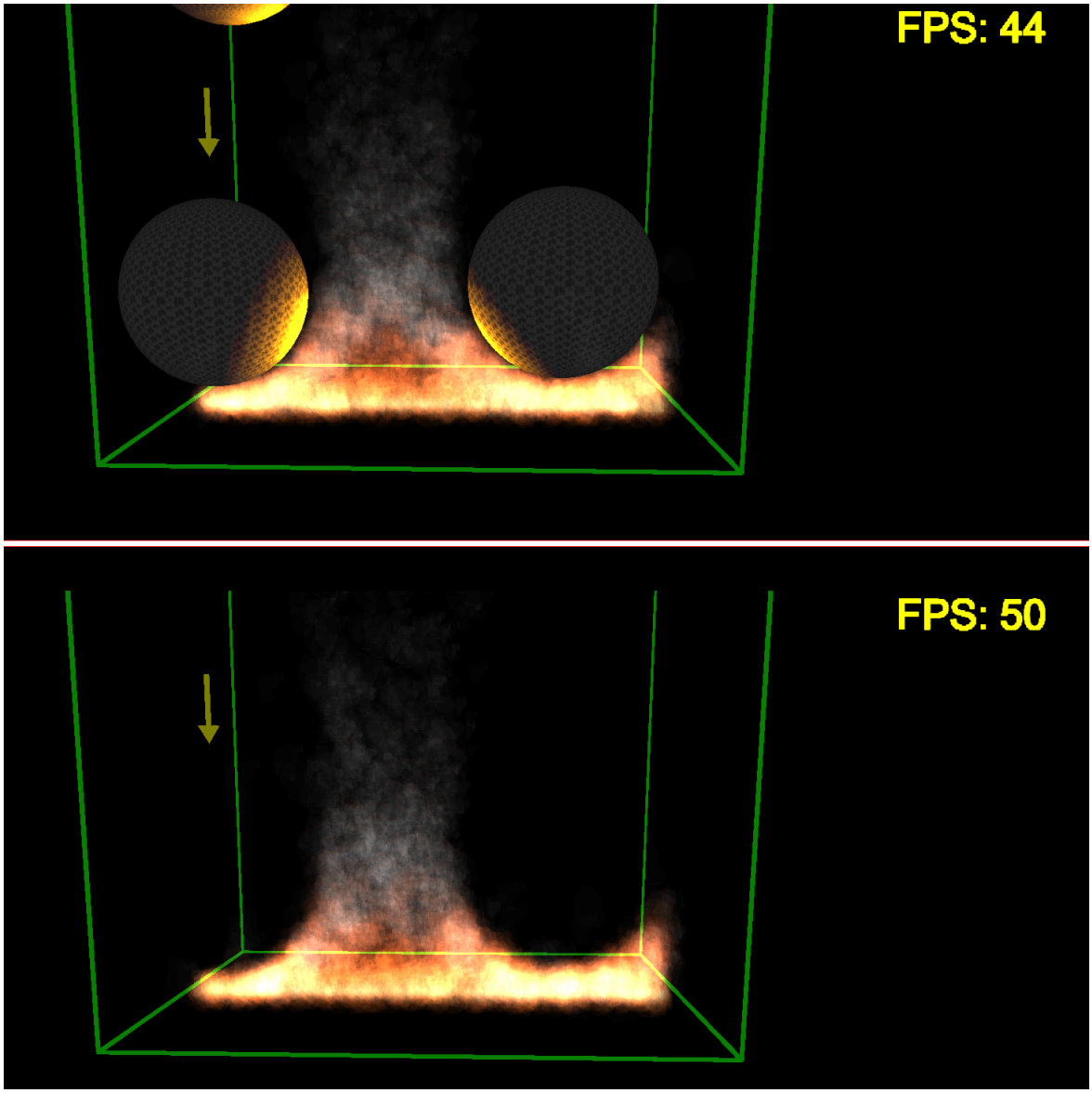


FIGURE 5-9 – Test d'interactivité avec un mur de feu, les objets interactifs sont visibles dans la figure en haut et non visibles dans celle du bas mais il restent présents dans les deux cas. (Ces figures sont des captures d'écran prises de 3DFS)

2. **Description :**

   Nous avons pu obtenir les images ci-dessus (Voir la figure 5-9) avec 3DFS en utilisant la configuration du tableau (Voir le tableau 5.4) où :

   – "sp1" est un système de particules.

   – "p" est une particule qui existe chez "sp1".

   – "Ps" est l'ensemble de toutes les particules qui appartiennent à "sp1".

   – "em" est un émetteur qui appartient à "sp1".

   – "même" veut dire que c'est la même configuration pour un feu-doux.

	Mur de feu (test d'interactivité)
FPS	Entre 44 et 50
Formes des *Ps*	Pour chaque proportion de temps qui passe de nouvelles formes complexes sont affectées au *Ps*.
Couleurs des *Ps*	Gamme de couleurs entre rouge et jaune définis par une texture
Définition de la texture	8 192px × 8 192px
Résolution de la texture	300 ppp
Transparence des *Ps*	0,038 sur 1
Transparence (zone affectée) *Ps*	y = 1 sur 1 , x = 1 sur 1
Taille des *Ps*	Fixe : 3.4
Durée de vie des *Ps*	Aléatoire entre : $[0.3; 1.3]$
Forme de *em*	Ligne de longueur 50cm
Vitesse	8.0
Modificateur de gravité	1.0
Nombre de "Ps" émisses par seconde	8 000
Interactivité (objets-particules)	Deux objets interactifs
Mélange additif	Actif
Flocons de Feu	Non présentes
Édition (en temps réel)	Oui
Rendu par instance	Oui
La Fumée avec le feu	Présente

TABLE 5.4 – La configuration pour un mur de feu (test d'interactivité).

- Test 4 : test d'interactivité avec un feu de camp :

  1. **Résultats :**

     – Vue de face :

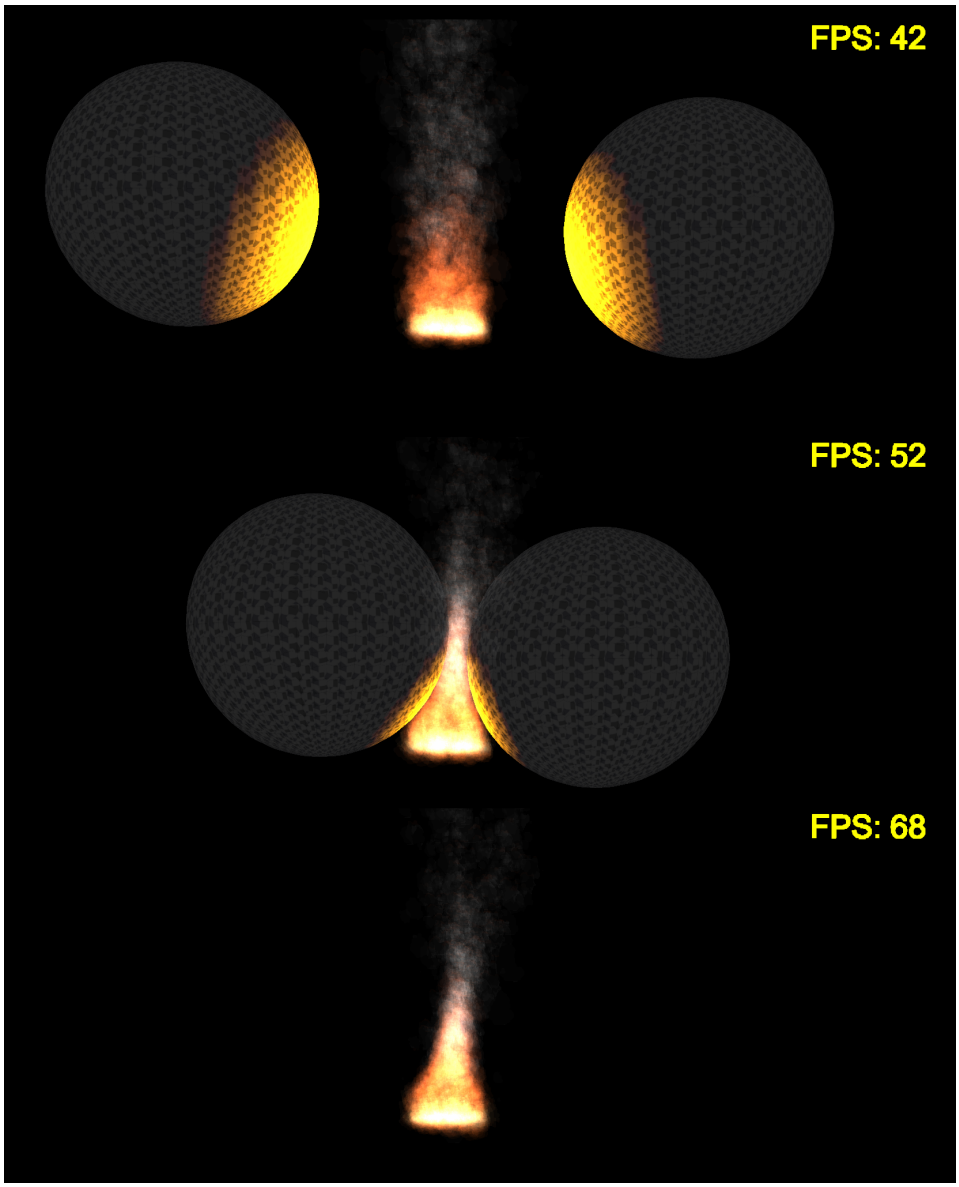


FIGURE 5-10 – Test d'interactivité avec un feu de camp. Dans la figure en bas les objets interactifs sont invisibles mais présents(Ces figures sont des captures d'écran prises de 3DFS)

2. **Description :**

   Nous avons pu obtenir les images ci-dessus (Voir la figure 5-10) avec 3DFS en utilisant la configuration du tableau (Voir le tableau 5.5) où :

   – "sp1" est un système de particules.

   – "p" est une particule qui existe chez "sp1".

   – "Ps" est l'ensemble de toutes les particules qui appartiennent à "sp1".

   – "em" est un émetteur qui appartient à "sp1".

   – "même" veut dire que c'est la même configuration pour un feu-doux.

	Feu de camp (test d'interactivité)
FPS	Entre 52 et 68
Formes des *Ps*	Pour chaque proportion de temps qui passe de nouvelles formes complexes sont affectées au *Ps*.
Couleurs des *Ps*	Gamme de couleurs entre rouge et jaune définis par une texture
Définition de la texture	8 192px × 8 192px
Résolution de la texture	300 ppp
Transparence des *Ps*	0,036 sur 1
Transparence (zone affectée) *Ps*	y = 1 sur 1 , x = 1 sur 1
Taille des *Ps*	Aléatoire entre : $[1; 1.9]$
Durée de vie des *Ps*	Aléatoire entre : $[0.5; 1.4]$
Forme de *em*	Disque de rayon = 9.35cm
Vitesse	8.0
Modificateur de gravité	1.0
Nombre de "Ps" émisses par seconde	3 000
Interactivité (objets-particules)	Deux objets interactifs
Mélange additif	Actif
Flocons de Feu	Non présentes
Édition (en temps réel)	Oui
Rendu par instance	Oui
La Fumée avec le feu	Présente

TABLE 5.5 – La configuration pour un feu de camp (test d'interactivité).

5.5.3 Autres tests

1. **Résultats :** l'image (Voir la figure 5-11) présente les autres variétés de tests qu'on peut obtenir en utilisant 3DFS avec des différentes configurations de paramètres.

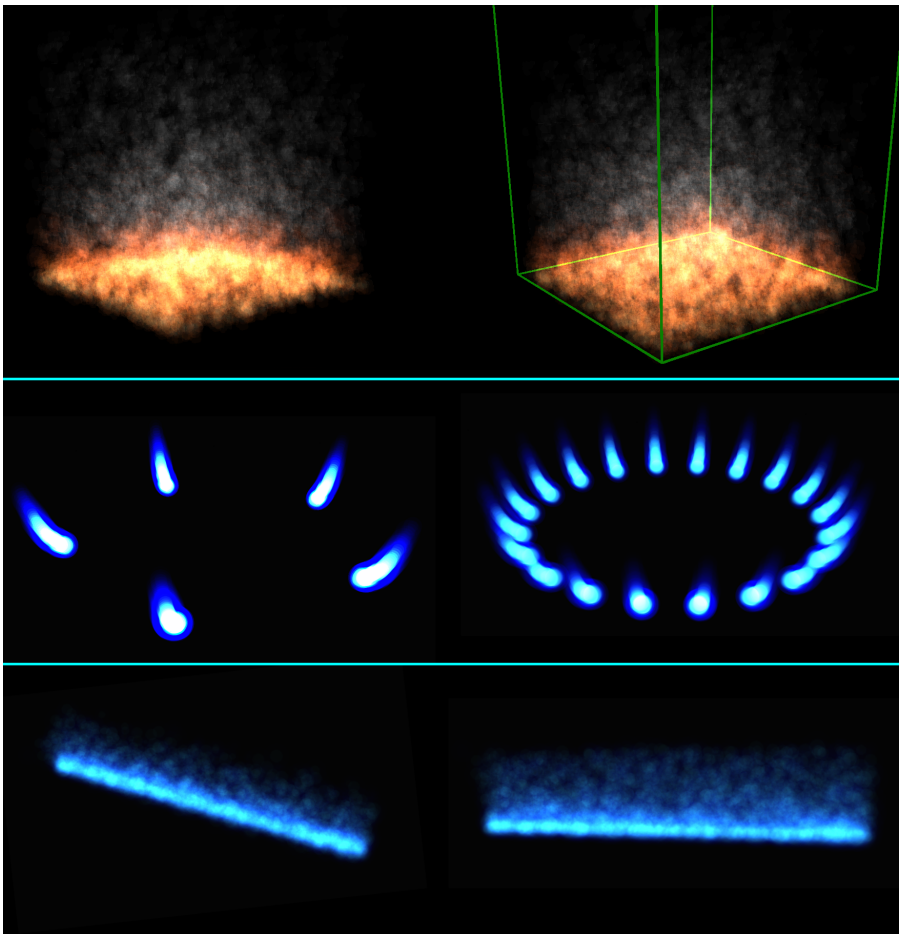


FIGURE 5-11 – Autres variétés de tests (Ces figures sont des captures d'écran prises de 3DFS)

2. **Description :** Ces résultats (Voir la figure 5-11) ont pu être obtenues en changeant les caractéristiques de l'émetteur :

   - Pour les deux premiers résultats de la figure (Voir la figure 5-11) nous avons mis la forme de l'émetteur comme "Rectangle", la longueur à 50cm et la largeur à 50cm.
   - Pour les deux résultats qui sont au milieu nous avons mis la forme de l'émetteur à "N points dans un cercle", le rayon à 9cm le Nombre de points N varie pour chaque résultat.
   - pour les deux résultats qui restent nous avons mis la forme de l'émetteur comme "Ligne" et la longueur à 50cm.

5.5.4 Étude comparative

Dans cette sous-section nous allons comparer nos travaux avec des images prises de la réalité et des images des travaux précédents.

1. **Comparaison d'un feu de camp avec des images d'un feu réel :**

   - La figure (Voir 5-12) présente nos résultats de la simulation d'un feu de camp à coté d'un feu réaliste.

FIGURE 5-12 – Comparaison avec un feu réaliste. (Les figures qui sont à gauche sont des captures d'écran prises de 3DFS)

- Les similarités trouvées au niveau visuel du feu entre les images de notre simulation et les images du feu réel (Voir la figure 5-12) sont :
  - Les flocons de feu.
  - La dérivation à droite du feu à la présence d'un souffle de vent (voire les deux premières images en haut de la figure 5-12).
  - La petite proportion de lumière orange qui se dégage du feu.
  - La même gamme de couleur du feu qui est entre (le jaune et le rouge).

2. **Comparaison d'un feu d'une cuisinière de notre simulation avec un feu réaliste :**

   - La figure (Voir 5-13) présente notre résultat de la simulation d'un feu d'une cuisinière à gaz à coté d'un feu réaliste.

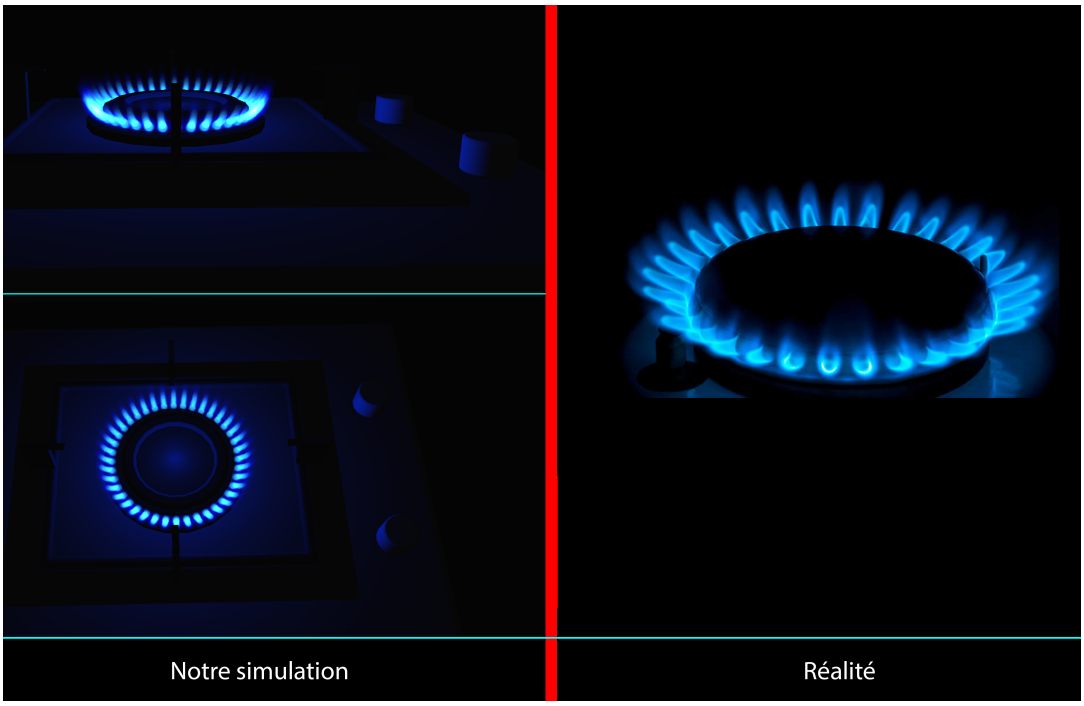


FIGURE 5-13 – Comparaison d'un feu d'une cuisinière avec un feu réaliste. (les figures qui sont à gauche sont des captures d'écran prises de 3DFS)

   - Les similarités trouvées au niveau visuel du feu entre les images de notre simulation et l'image du feu réel (Voir la figure 5-13) sont :
     - Le même nombre de source de feu (d'émetteurs) trouvés dans une cuisinière à feu.
     - La même direction des flammes et qui n'est pas du bas vers le haut comme dans un feu de camp mais sous forme d'un arc.
     - La petite proportion de lumière bleu qui se dégage du feu de la cuisinière.
     - La même gamme de couleur du feu qui est entre (le blanc et le bleu).

3. **Comparaison entre notre simulation de feu et entre les travaux précédents :**

   - La figure (Voir 5-14) présente notre résultat de la simulation du feu de camp à coté des autres simulations des travaux similaires.

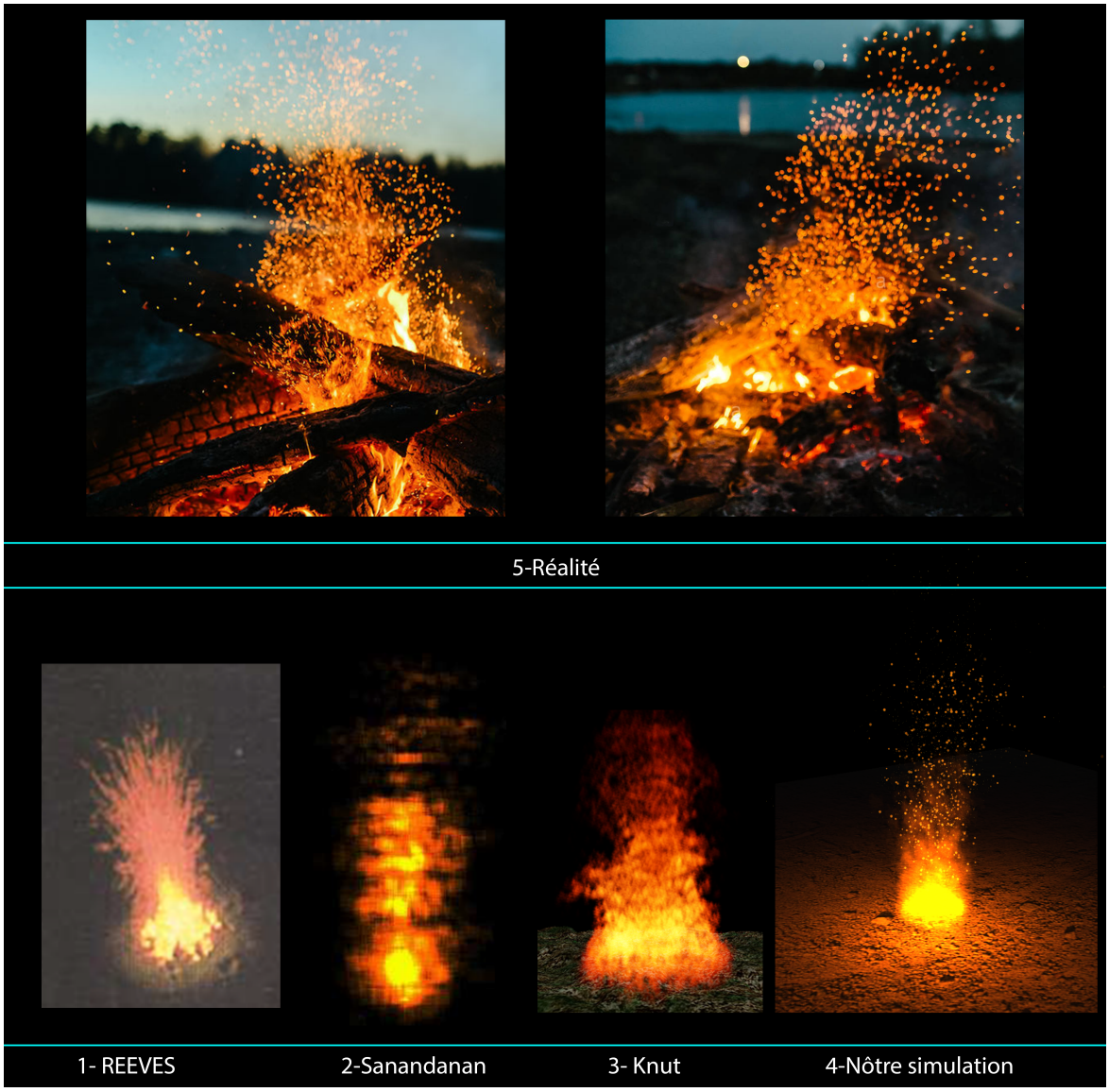


FIGURE 5-14 – Comparaison entre notre simulation de feu et entre les travaux précédents et des images de la réalité. (la figure "4-notre simulation" est une capture d'écran prise de 3DFS)

- Le tableau suivant (Voir le tableau 5.6) présente une comparaison détaillée entre notre travail et les différents travaux similaires.

	4-Notre simulation	1-Reeves	2-Sanandanan	3-Knut
Flocons de Feu	Implémentés	Non implémentés	Non implémentés	Non implémentés
Interactivité (objets-particules)	Implémentée	Non implémentée	Non implémentée	Non implémentée
Rendu par instance	Oui	Non	Non	Non
La proportion de la lumière oronge que relâche le feu	présente	Non présente	Non présente	Non présente
La Fumée avec le feu	Non présente mais implémentée	Non présente	Non présente	Non présente
Nombre de "Ps" émisses par seconde	8 000	25 000	...	16 384
Édition (en temps réel)	Oui	Non	Oui	Oui
Couleurs des *Ps*	Gamme de couleurs entre rouge et jaune définis par une texture	Gamme de couleurs entre rouge et jaune	Gamme de couleurs entre rouge et jaune	Gamme de couleurs entre rouge et jaune
Mélange additif	Implémenté	Implémenté	Implémenté	Implémenté
Forme de *em*	Disque	point	Disque	Disque

TABLE 5.6 – Tableau comparative entre notre travail et les différents travaux similaires.

5.6 Conclusion

Dans ce chapitre nous avons pu résumer la plupart de nos travaux pratiques, d'abord nous avons abordé les outils et les plateformes que nous avons utilisé durant notre travail pratique, par la suite nous sommes passé à la description détaillée de la configuration matérielle et logicielle qui a été utilisée pour la création et pour les tests de notre outil qui s'intitule : 3DFS. Ensuite nous avons passer à la présentation de l'interface graphique de 3DFS, et pour finir nous avons fait des tests et des comparaisons des résultats que nous avons pu avoir avec 3DFS.

Conclusion générale

Nous rappelons que dans le cadre de notre analyse, le type de simulation de feu visée et qui représente notre objectif été le type d'une simulation de feu qui se base sur un système de particules, qui est proche à la réalité et qui doit intégrer un certain niveau d'interactivité avec l'environnement tout en restant en temps réel.

Nous avons dans un premier temps exposé les solutions apportées précédemment pour réaliser une simulation d'un feu. les travaux vus avant se sont révélés loin d'être proche à la réalité et même s'il y a eu ceux qui sont proches à la réalité c'étaient des travaux qui ont été réalisés sans se préoccuper de l'interactivité du feu avec son environnement.

Afin de réaliser une simulation qui assure un bon niveau de réalisme comparé aux autres travaux, nous avons utilisé une mixture des méthodes vues en 1.5.2 afin de bénéficier du meilleur de chaque méthode.

Face à l'absence du réalisme et de l'interactivité dans les simulations de feu, nous nous sommes parer de :

- L'absence du réalisme, en utilisant des techniques tell que (**le mélange additif** et **les flocons de feu**) et des **textures ultra haute définition** que nous avons créées.

- L'absence de l'interactivité, en utilisant les principes de la détection et de l'évitement des collisions tout en créant un algorithme qui couvre tous ces aspects.

- L'absence de l'optimalité, en utilisant le rendu par instance qui fonctionne avec le parallélisme de la carte graphique et qui sert à optimiser dans le coté affichage et non pas dans le coté calcul.

Notre travail consisté en la simulation d'un feu réel mais nous nous sommes permis d'aller au-delà d'une simulation d'un feu en créant un outil avec le quel on peut créer des simulation de feu, de pluie, de la neige etc...), et ceci est permis grâce au contrôle total des simulations fournis par notre système qui s'intitule 3DFS.

Pour finir, nous pensons à poursuivre notre travail afin de combler de nouveaux objectifs qui servirons à avoir des simulations encore plus proche à la réalité comme :

- L'optimisation des calculs en utilisant les GPU avec CUDA ou OpenCL afin de pouvoir générer plus de particules dans la scéne et de rester en temps réel.
- Le changement de la géométrie des objets au contact du feu selon leurs types de matériaux.

Bibliographie

[1] Plume Jim Barton, Jack. Public participation in a spatial decision support system for public housing. *Computers, Environment and Urban Systems*, 9(6) :630–652, 2005. Article.

[2] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3) :235–256, July 1982. Article.

[3] HOURIHAN J. BRIDSON, R. and M. NORDENSTAM. Curl-noise for procedural fluid flow. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 26(1), 2007. Article.

[4] Horvath C. and Geiger W. Directable, high-resolution simulation of fire on the gpu. *ACM Transactions on Graphics (TOG)*, 28(3), August 2009. Article.

[5] SHINAR T. HONG, J.-M. and R. FEDKIW. Wrinkled flames and cellular patterns. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 26(3) :1188–1198, 2007. Article.

[6] A. Maurice Jones. *Fire Protection systems*. Delmar, 2009. Book.

[7] Holobar A Koročsec, D. Building interactive virtual environments for simulated training in medicine using vrml and java/javascript. *Computer Methods and Programs in Biomedicine*, 80(4) :61–70, 2005. Article.

[8] Branko MaroviÄÄ. Web-based grid-enabled interaction with 3d medical data. *Future Generation Computer Systems*, 22 :385–392, 2006. Article.

[9] C.H. Perry and R.W. Picard. Synthesizing flames and their spreading. *Proceedings of the Fifth Eurographics Workshop on Animation and Simulation Journal*, pages 1–14, 1994. Article.

[10] James G. Quintiere. *Fundamentals of Fire Phenomena*. Wiley, 2008. Book.

[11] William Thomas Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics Journal*, 2(2) :91–108, April 1983. Article.

[12] K. E. S. Rodal and G. Storli. Physically based simulation and visualization of fire in real-time using the gpu. *Master of Science in Computer Science, Norwegian University of Science and Technology*, January 2006. Masters Thesis.

[13] RASMUSSEN N. SELLE, A. and R. FEDKIW. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3) :910–914, 2005. Article.

[14] Sanandanan Somasekaran. Using particle systems to simulate real-time fire. 2005. Mémoire.

[15] Zhou Zhong Wu ZhaoHui and Wu Wei. Realistic fire simulation : A survey. *Computer-Aided Design and Computer Graphics Journal*, pages 333–340, 2011. Survey.

[16] S. Yoshida and T. Nishita. Modelling of smoke flow taking obstacles into account. *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, pages 135–443, 2000. Article.

[17] LU L. ZHOU S., SUN Y. and CHEN Z. Fire simulation model based on particle system and its application in virtual reality. *16th International Conference on Artificial Reality and Telexistence–Workshops (ICAT'06)*, November 2006. Article.