Infographie

Création d'un logiciel de lancer de rayon

Pénombre

Facettes

Jean-Guillaume Delculry

3ème année — Option IHM

E.I.S.T.I.

Table des matières

Introduction **5**

1 Travail effectué **5**
1.1 Ce qui a été fait 5
1.2 Ce qui est à améliorer 5

2 La pénombre **6**

3 Facettes **7**

Conclusion **8**

Introduction

Il est impressionnant de constater que par le jeu d'opérations mathématiques, il est possible d'afficher de si belles choses.

Ce cours d'infographie m'a permis de compléter mes connaissances, je veux parler de celles que j'avais — et qui s'avèrent plutôt succintes — concernant le raytracing. En effet, obtenir de telles images me replonge à l'époque où je bricolais des images à l'aide du très célèbre logiciel `Persistance Of Vision`, POV pour les intimes.

1 Travail effectué

1.1 Ce qui a été fait

Le logiciel de lancer de rayon a été pogrammé en C++. Les fonctionnalités suivantes ont été implémentées et fonctionnent :

- multisources
- réflexion spéculaire
- ombre
- effet miroir
- pénombre
- plans bifaces
- sphères
- facettes bifaces
- récursivité

Pour garder mon code plus clair, je l'ai commenté un minimum. Je pense que ce gain de lisibilité permet une bonne compréhension, d'autant plus que l'implémentation parle d'elle même.

1.2 Ce qui est à améliorer

Il y a des parties de code à améliorer. En effet, les variables suivantes sont fixées dans le corps du programme :

- la constante d'éclairage `ECLAIRAGE`
- la rugosité `N`
- l'intensité de la source lumineuse `Ip`
- le coefficient de réflexion spéculaire K_S
- le coefficient de reflexion diffuse K_D
- l'attenuation de la source lumineuse en fonction de la distance f_a

De même, le code n'a pas été totalement optimisé pour gagner du temps. Il est didactique, et c'est tout ce que je lui demande.

Il faudrait aussi ajouter des fonctionnalités en ligne de commande.

2 La pénombre

La pénombre est un effet extrêmement simple : il s'agit e créer des sources virtuelles décallées d'une petite quantité Δ sur les axes x,y et z qui contribuent à l'éclairage de la scène dans des proportions modeste (il faut en effet moyenner l'intensité).

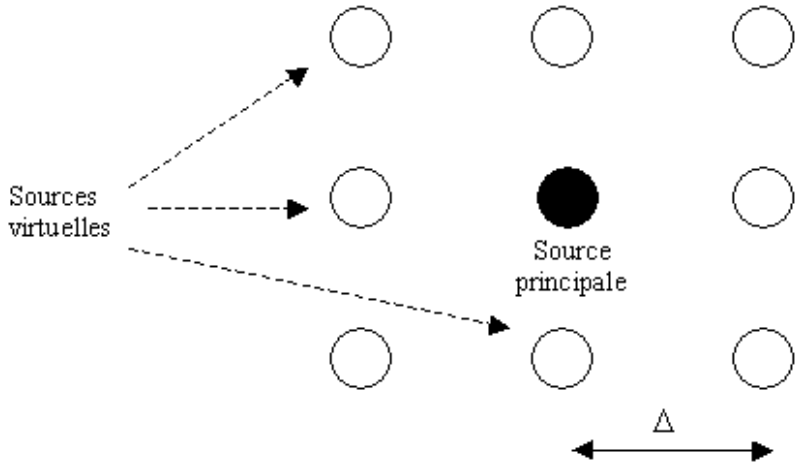

FIG. 1 – Principe des sources virtuelles

En faisant varier Δ, on peut obtenir de nombreux effets de pénombre. Quelques exemples sont rassemblés sur la page suivante.

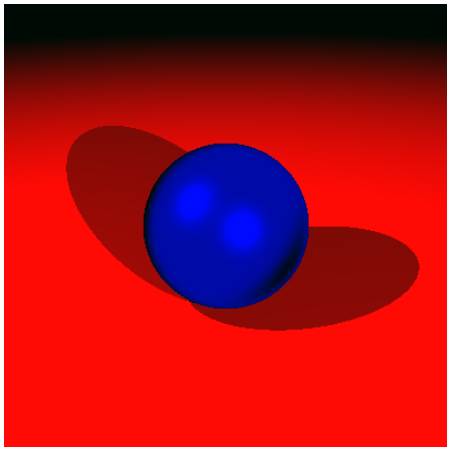

FIG. 2 – sans pénombre

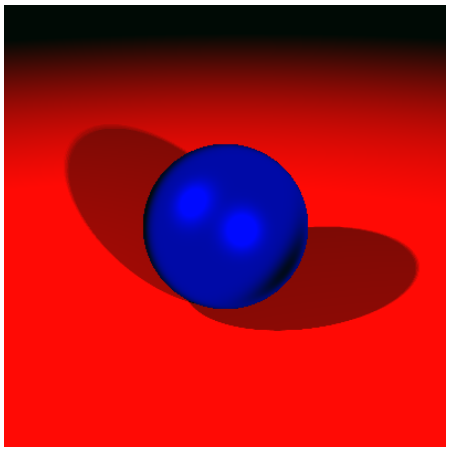

FIG. 3 – avec $\Delta = 0.1$

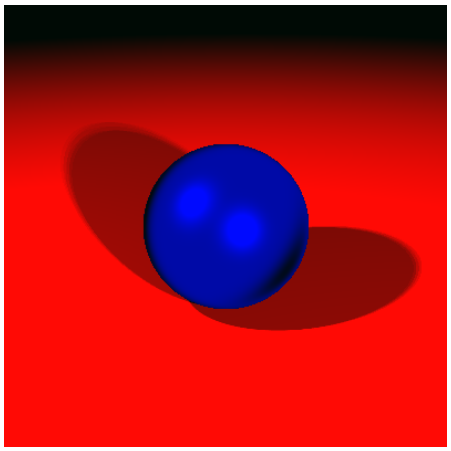

FIG. 4 – avec $\Delta = 0.2$

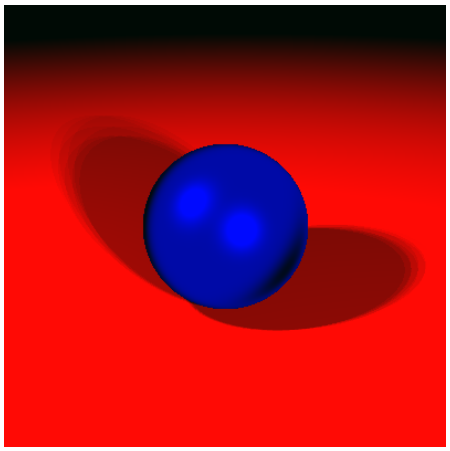

FIG. 5 – avec $\Delta = 0.5$

Le code relatif à la pénombre se trouve dans la fonction `Luminosite raytrace(Rayon Ray,int cont)` qui reçoit le rayon calculé, ainsi que le nombre de réplexions successives.

Personnellement, j'ai choisi de fixer Δ à 0.1.

3 Facettes

Les facettes ont les mêmes propriétés que les plans, sauf que l'on définit un contour de 3 points délimitant la surface à afficher.

Les test sont les suivants :

1. On calcule l'équation plan contenant la facette
2. On regarde si le rayon envoyé intersecte ce plan

3. si oui, on regarde si le point d'intersection appartient au contour (A, B, C)
4. si oui, alors on intersecte la facette. Sinon, il n'y a pas d'objet empêchant la popagation du rayon (on ne considère pas l'intersection avec le plan).

Pour vérifier si le point d'intesection I appartient à la facette, on effectue les opérations suivantes :

1. Si le point d'intersection appartient au plan contenant la facette, on vérifie s'il apartient à la surface bleue :
$$s1 = (\overrightarrow{IA} \wedge \overrightarrow{IB}) \cdot (\overrightarrow{IA} \wedge \overrightarrow{IC}) < 0$$
2. puis on vérifie que le point d'intersection appartient à la surface verte :
$$s2 = (\overrightarrow{IB} \wedge \overrightarrow{IA}) \cdot (\overrightarrow{IB} \wedge \overrightarrow{IC}) < 0$$
3. Si le point d'intersection satisfait à ces deux conditions alors, il est dans la facette (partie jaune).

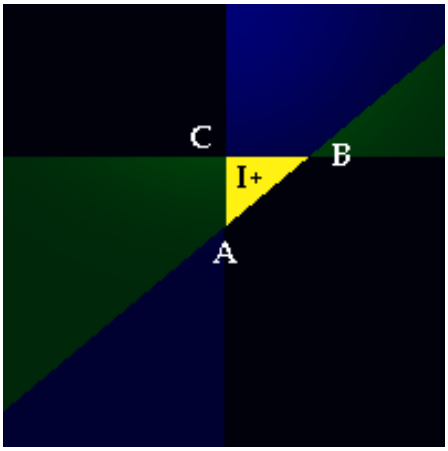

Fig. 6 – tests d'appartenance au contour

Conclusion

Travailler d'arrache-pied sur ce projet m'a permis de découvrir l'envers du décor. J'ai maintenant bien compris comment un raytraceur permettait la génération d'images de synthèse, ce qui me permet par ailleurs d'envisager d'autres applications.

Pour un projet de réalité virtuelle, j'ai réutilisé ce moteur, un peu plus allégé, et j'y ai ajouté une fonction générant du bruit de Perlin pseudo-aléatoire. Je dispose ainsi d'une base me permettant d'effectuer des tests su ces bruits.