Modélisation et représentation d'une scène dans l'espace

Sommaire

Introduction

L'infographie, autrement dit la création d'image assistée par ordinateur, est devenue au cours de ces dernières années un des domaines les plus importants, utilitairement parlant. En effet, de la création de personnages dans les jeux vidéos ou d'images de synthèse en cinématographie à la représentation de simulations en physique ou en mathématiques, l'infographie démontre sa puissance en tant qu'outil permettant de générer informatiquement des illustrations selon les désirs de l'utilisateur.

Cet exposé s'intéressera à retracer le début de l'évolution de l'infographie en passant par l'élaboration d'un algorithme permettant d'afficher une scène tridimensionnelle, et la résolution de problèmes que posent la construction d'un tel algorithme afin d'aboutir à une scène la plus réaliste possible.

Partie I - Méthodes d'affichage

Le premier problème de l'affichage est, bien sûr, de faire en sorte que les parties cachés n'apparaissent pas sur l'image finale. Pour ce faire, il existe deux méthodes principales : celle du Z-buffer, et celle du lancer de rayon.

Pour illustrer leurs principes, on s'intéresse ici à une scène représentée dans la figure 1, composée d'un cube et d'un carré, que l'on observe depuis un point de vue à travers un écran virtuel composé de pixels.

A) Z-Buffer

Cette méthode est basée sur un pré-calcul de la scène, consistant à discrétiser les objets en un certain nombre de points. Puis, pour chaque point obtenu, on trace la droite passant par ce point et le point de vue, et on détermine le pixel intercepté, s'il existe, par cette droite.

Dès lors, soit le pixel est vide, auquel cas, on le remplit de la couleur du point, soit il y a déjà une couleur associée à un point qui remplit le pixel. Dans ce dernier cas, on compare les distances entre le pixel et chacun des points, puis, on prend la couleur du plus proche, comme illustré dans la figure 1.1.

Le nom de la méthode provient du cas où la projection sur l'écran est orthogonale, auquel cas il suffit, en plaçant un axe "z" perpendiculaire à l'écran, de comparer les composantes selon cet axe.

L'avantage de cette méthode est d'être, en théorie, rapide, ne nécessitant que peu de calculs une fois la discrétisation effectuée.

Cependant, elle pose aussi un certain nombre de contraintes. La discrétisation doit être faite suffisamment de points pour être sûr d'avoir une résolution suffisante, mais auquel cas, le nombre de points calculés et conservés en mémoire peut rapidement exploser (proportionnel à la surface des polygones)

B) Lancer de rayon

Cette méthode adopte le point de vue inverse de la précédente. On part cette fois-ci du point de vue, et on « lance » un rayon vers un pixel pour récupérer le point qui sera affiché (cf figure 1.2).

Nous perdons, par rapport à l'autre méthode, de la rapidité puisqu'il faut calculer l'intersection de chaque rayon avec la scène, mais on gagne en mémoire.

D'autre part, la méthode du lancer de rayon a aussi l'avantage de permettre d'implémenter facilement l'éclairage par des sources lumineuses, ainsi que la réflexion par des miroirs, comme nous le verrons dans la seconde partie.

Partie II – Problèmes de géométrie

A) Objets utilisables

Nous l'avons vu dans le détail de la méthode, le lancer de rayon nécessite de calculer l'intersection d'une droite avec un des solides de la scène. Mais cela n'est pas toujours possible de manière simple.

En effet, dans le cas où l'équation de la surface intersectée est simple, c'est-à-dire au plus un système polynomial de degré 2 en les coordonnées, il est facile, en paramétrant le rayon, de calculer les points d'intersection de manière formelle. C'est le cas par exemple avec les sphères, les cubes les cylindres.

Pour d'autres surfaces, le principe est le même, mais le calcul des points d'intersection doit passer par une résolution numérique.

L'implémentation d'une résolution formelle pour les quelques formes simples cités donne, par exemple, la scène représentée dans la figure 2.1.

B) Éclairage et miroirs

C'est dans cette partie que l'on peut constater un des grands avantages du lancer de rayon, que ce soit pour l'éclairage ou la réflexion.

Concernant l'éclairage, pour savoir quelle est la luminosité en un point, il suffit de considérer le rayon partant du point, en direction d'une source lumineuse, et de voir si ce rayon intersecte bien ladite source, ou est intercepté avant. L'algorithme utilisé ici présente l'avantage d'être le même que celui utilisé pour les rayons partant de la source. L'implémentation de sources lumineuses aboutit à la scène représentée dans la figure 2.2.

On constatera aussi que le problème pour calculer les réflexions est bien plus simple avec le lancer de rayon. Il suffit dans notre cas d'appliquer simplement l'algorithme sur le rayon réfléchi sur un miroir, ce qui donne la scène représentée dans la figure 2.3.

Le jeu d'ombres et de lumière ainsi implémenté contribue à renforcer le réalisme de la scène. Cependant, un dernier point reste à régler, qui est celui des textures.

Partie III – Génération de textures

A) Texture aléatoire

De manière générale, lorsqu'on parle « d'appliquer » une texture à une surface, cela revient à « coller » une image, la texture, sur ladite surface. Cependant, pour certaines surfaces, cette opération se révèle difficile, comme dans le cas des sphères. Les cartographes connaissent un problème similaire lors de la réalisation de la carte de la Terre, comment le faire en ayant le moins de déformations possibles.

Dans notre cas, nous ne nous intéresserons uniquement aux surfaces sur lesquelles appliquer une texture ne pose pas de problème, à savoir les surfaces planes et les cylindres.

Dans les scènes précédentes, c'est le côté uni de la couleur des surfaces qui fait qu'elles ont l'air peu réelles.

Une idée naïve pour résoudre le problème est donc de perturber la texture en y ajoutant un « bruit », autrement dit de créer de petites variations de couleurs de sorte à ne plus avoir de texture unie. L'application de cette idée donne la scène représentée dans la figure 3.1.

Le rendu est différent ; les surfaces ne sont plus unies, mais le fait d'avoir perturbé aléatoirement crée de fortes discontinuités, qui, au final, n'aident pas au réalisme de la scène. D'où la nécessité de générer un bruit cohérent.

B) « Bruit » de Perlin

1) Génération d'un bruit cohérent

Le problème est donc de générer un bruit, possédant un caractère aléatoire, mais aussi continu. L'idée de l'algorithme de Perlin est donc d'interpoler un bruit aléatoire afin de le lisser.

Le bruit de départ est donc interpolé avec un certain nombre de points. Plus le nombre de points est grand, moins la cohérence de la figure interpolée est globale et plus les détails sont visibles. Ainsi, en choisissant différents échantillonnages, on obtient plusieurs figures d'interpolation, plus ou moins détaillées.

La dernière étape consiste à superposer les différentes figures, de sorte à obtenir un bruit possédant une certaine cohérence globale, mais avec des détails. En pondérant les figures lors de la superposition, on peut ainsi mettre l'accent sur les détails ou non.

2) Applications

Avec le bruit obtenu avec l'algorithme de Perlin, on peut ainsi perturber des textures tout en les laissant continues. La figure 3.2 montre quelques exemples de textures obtenues à l'aide de cet algorithme.

En allant plus loin, on peut, en modifiant l'algorithme d'interpolation, modifier les bruits obtenus, en donnant, par exemple, plus d'importance à un axe donné sur la texture. Les résultats de cette interpolation modifiée sont présentés dans la figure 3.3.

Finalement, on aboutit à la scène présentée dans la figure 3.4, bilan actuel des résultats de cet exposé.

Bibliographie :

JAUBER Benoît, Outils pour les jeux sur ordinateur : Prise de connaissance de scènes 3D, thèse de doctorat (Université de Limoges), 2008

DACHSBACHER Casten, Interactive Terrain Rendering : Towards Realism with Procedural Models and Graphics Hardware, thèse de doctorat (Université de Stuttgart), 2006

LASRAM Annas, Exploration et rendu de texture synthétisées, thèse de doctorat (Université de Lorraine), 2012

Bruits et nombres aléatoires cohérents
Cours OpenClassroom (http://fr.openclassrooms.com/)

Annexes

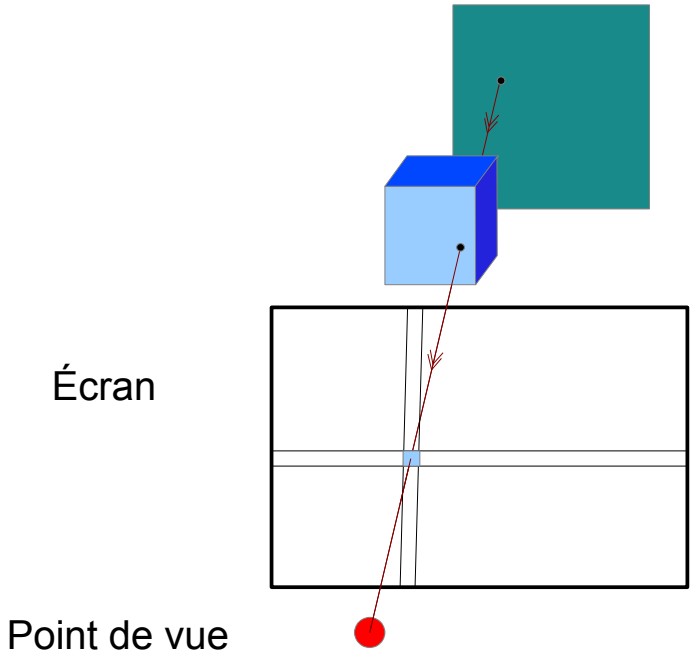

Figure 1.1 : Z-buffer

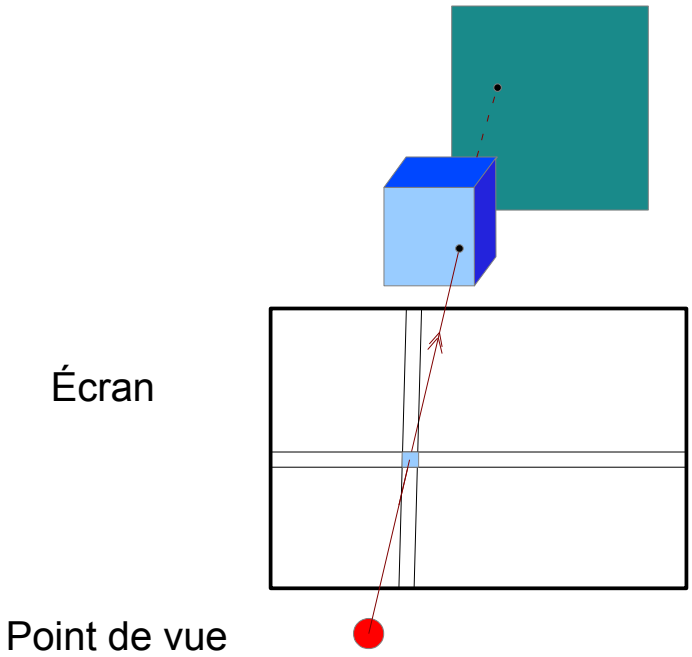

Figure 1.2 : Lancer de rayon

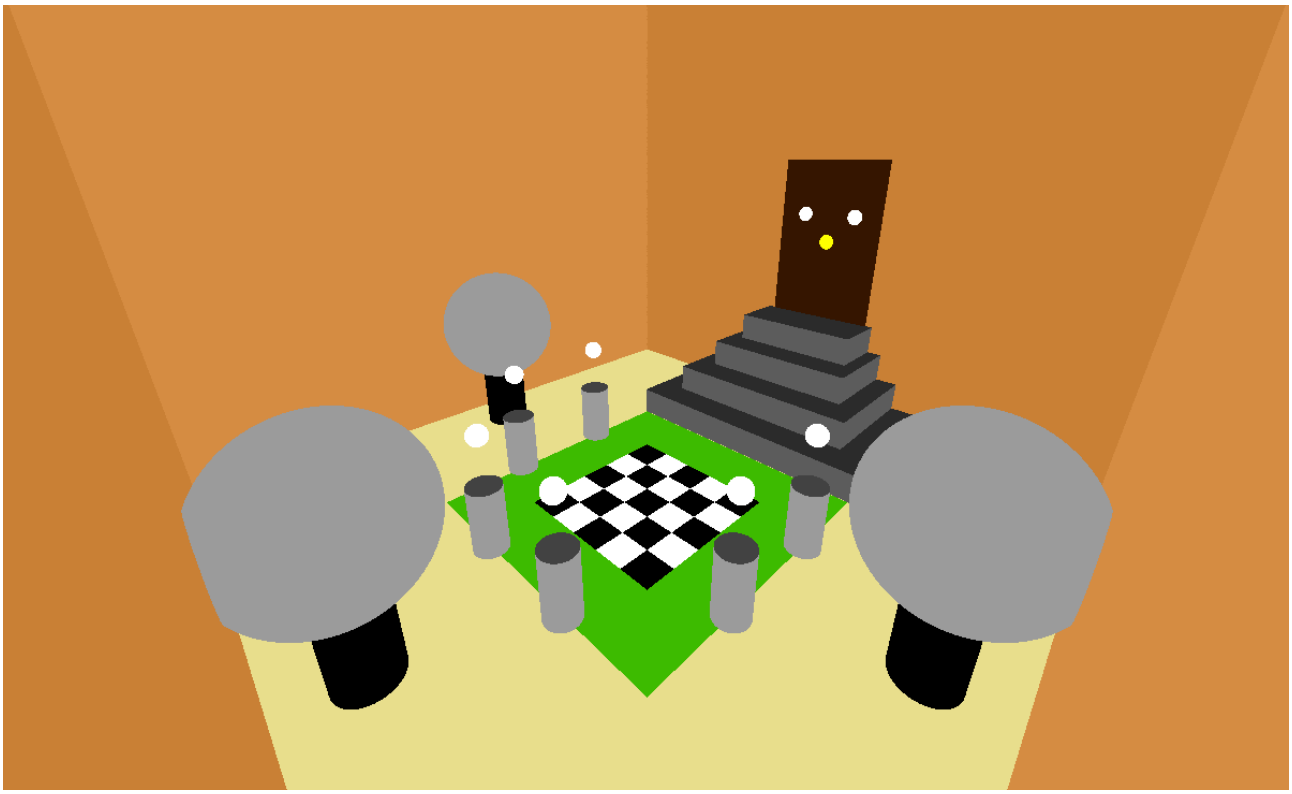

Figure 2.1 : Première scène

Figure 2.2 : Sources lumineuses

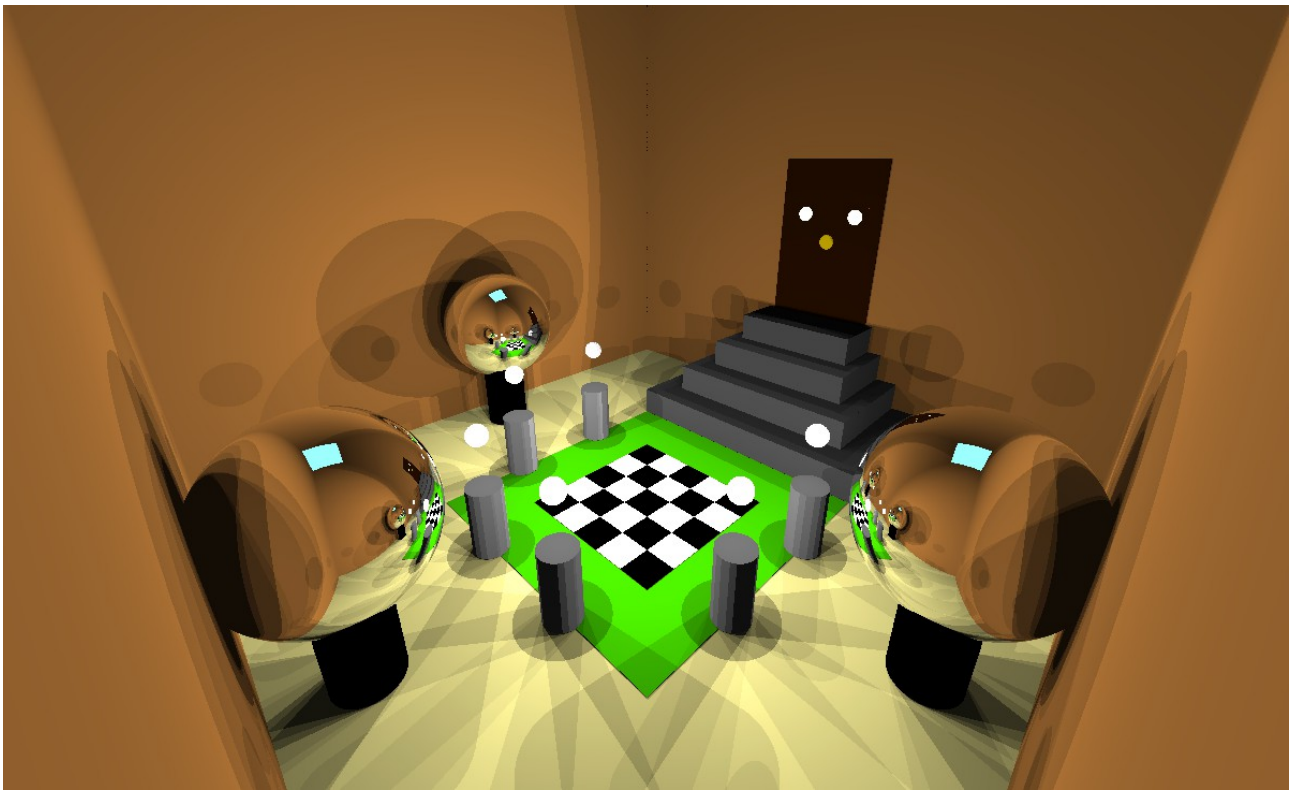

Figure 2.3 : Ajout de miroirs

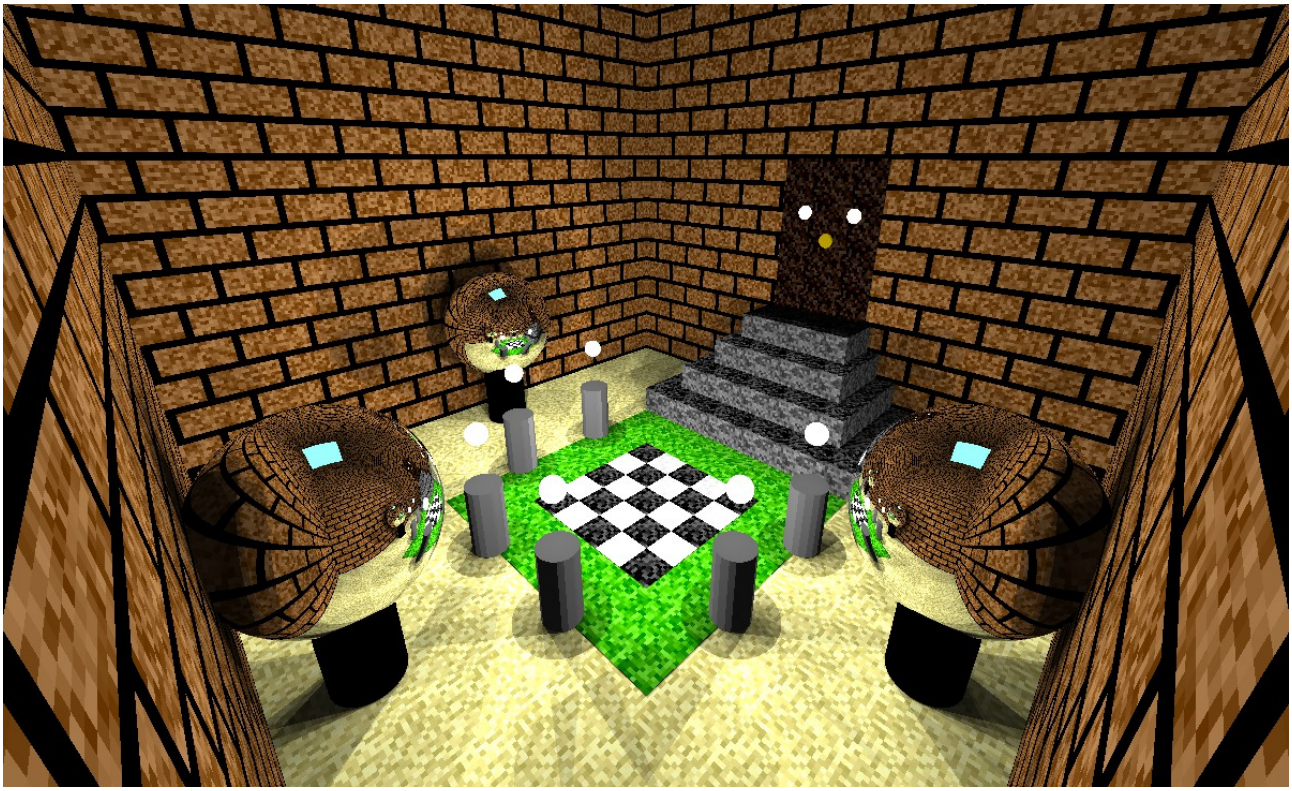

Figure 3.1 : Textures aléatoires

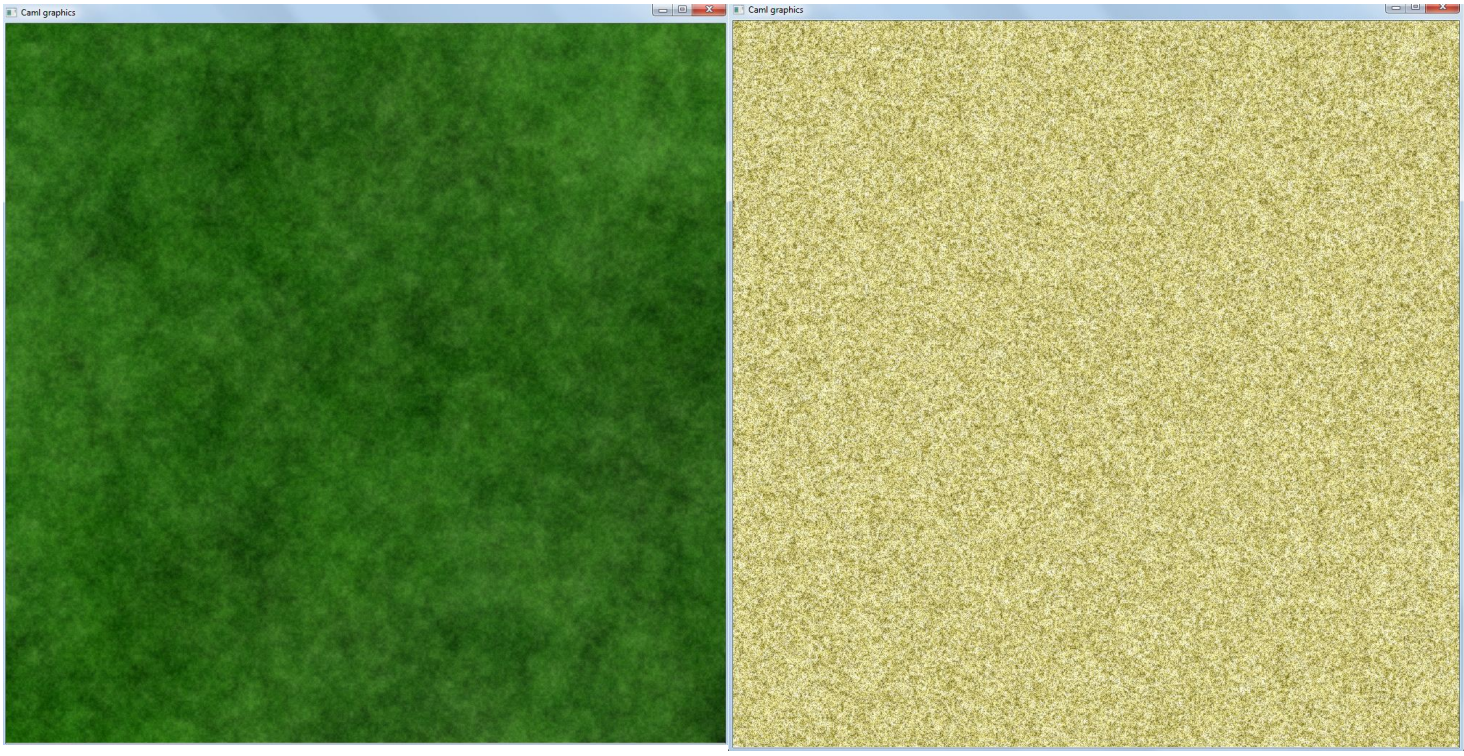

Figure 3.2 : Textures de Perlin

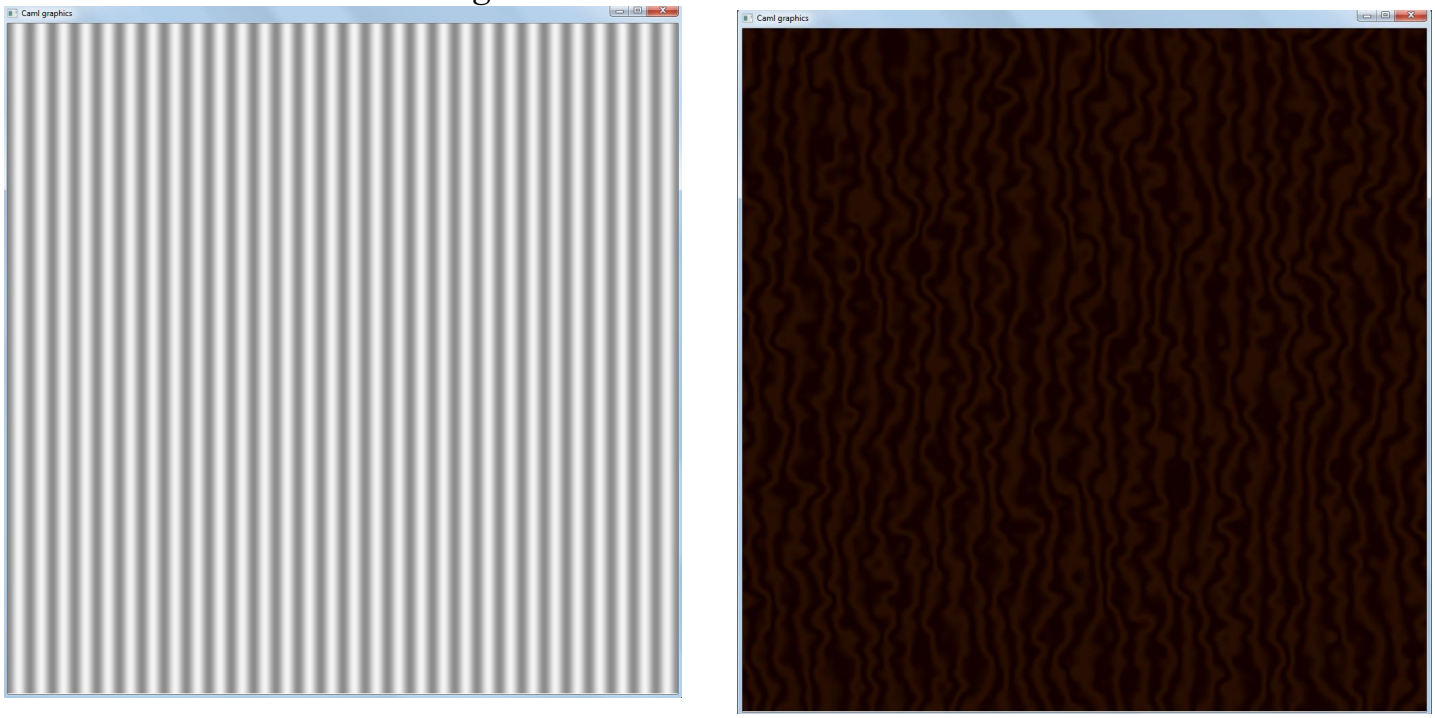

Figure 3.3 : Interpolation modifiée

Figure 3.4 : Scène finale