Rapport Tp2 ISIR

jeremy Vacher

February 2024

Table des matières

1 Introduction

Dans ce TP, nous alons voir une méthode permettant de générer des objets plus complexes en utilisant un maillage triangulaire. Nous verrons aussi deux méthodes permettant de réduire le temps de calcul et d'éviter les calculs inutiles en utilisant la méthode de la boite englobante et des hiérarchies de volumes englobants.

2 Premiers maillages

Dans cette partie, nous allons implémenter une méthode permettant de réaliser un maillage triangulaire. Pour ce faire, il a fallut récupérer l'algorithme de Möller et Trumbore pour vérifier l'intersection entre un rayon et un triangle, ici notre maillage triangulaire.

Cette méthode va permettre de vérifier si notre rayon est parallèle à notre triangle ou si nous avons effectivement une intersection. En testant le programme avec l'algorithme d'intersection et un objet sphère, nous obtenons cette image.

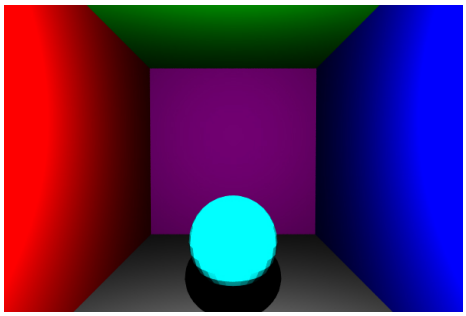

FIGURE 1 – Résultat question 1 a

Ensuite, pour régler le problème des facettes de la sphères on rajoute une fonction **getNormal()** qui va utiliser les coordonnées barycentriques du point d'intersection. Aprés avoir réaliser la fonction nous allons changer dans la classe **triangleMesh** la condition lorsque nous trouvons une intersection en remplaçant la fonction **getFaceNormal()** par notre nouvelle fonction. Nous obtenons alors cette image :

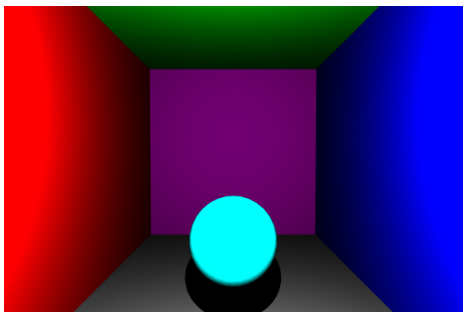

FIGURE 2 – Résultat question 1 b

Maintenant en chargeant l'objet lapin nous obtenons cette image.

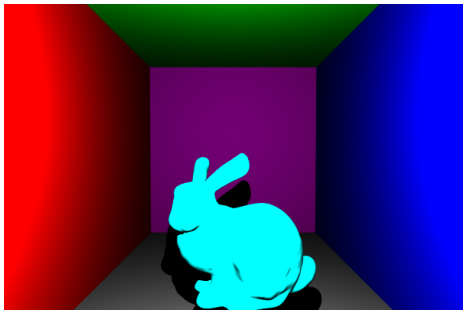

FIGURE 3 – Résultat question 1 c

3 Boites englobantes

Pour pallier le problème de lenteur de notre programme en utilisant l'objet lapin, nous allons utiliser des **boites englobantes** qui vont nous permettre de vérifier si nous avons une intersection entre le rayon et la boite avant de vérifier une intersection avec les triangles.

On va commencer par implémenter nos deux fonctions **extend** de la classe **aabb**. Pour cela dans la fonction ayant en attributs un points, on regarde le minimum entre le sommet min de notre boite englobante et notre point permettant de **mettre a jour** le sommet min de notre boites. On utilise le même fonctionnement pour le sommet max ainsi que pour la fonction **extend** ayant en attribut un **aabb** puis on utilise la fonction dans la méthode **addVertex**.

Enfin on modifie la fonction intersect de **triangleMesh** en rajoutant une condition qui retournera **false** si nous n'avons pas d'intersection avec la boite.

4 Hiérarchie de volumes englobants

L'objectif de cette partie est de réduire le temps de calcul de notre image en utilisant une structure BVH. Pour cela, nous allons faire des partitions en divisant en deux parties le plus grand axe de l'AABB. Nous allons alors implémenter les différentes méthodes de la classe BVH.

Nous allons donc commencer par réaliser les fonctions **intersectRect** et **intersectAnyRect** qui vont nous permettre de savoir si nous avons une intersection avec le rayon. Dans ces fonctions on regardera si nous sommes une feuille, si nous avons une intersection avec les triangles alors on mettra a jour certains paramètres et si nous avons un enfant droit ou gauche alors on vérifiera si nous avons une intersection et on retournera true.

Malheureusement je n'ai pas réussis a finir l'implémentassions de la classe BVH donc mon moteur sera moins efficace.