Rapport Tp7 ISIR

Jérémy Vacher

Avril 2024

Table des matières

1 Introduction

L'objectif de ce TP, est de réaliser de nouvelles surfaces que l'on appelle surface implicite. En calculant la sdf de chaque surface, nous permettant de savoir si le point est a l'intérieur, à l'extérieur de la surface ou sur la surface. Cela nous permettra alors de faire des objets complexes sans utiliser un grand maillage.

2 Le coeur de l'algorithme

On commence par implémenter la méthode **intersect()**, nous permettant de vérifier l'existence d'une intersection entre un **rayon** et une **surface**. Pour cela, on va utiliser un algorithme de John C. Hart permettant l'intersection entre une surface implicite et un rayon.

On va donc parcourir le rayon de p_Tmin à p_tMax et calculer la **SDF**(signed distance function). La **SDF**, va nous permettre de calculer la **distance signée** entre le point du rayon et la surface implicite. On pourra alors savoir ou le point se situe par rapport à la surface implicite. Ensuite, on regarde si notre distance est inférieure à une certaine valeur pour vérifier l'intersection et on met a jour les informations de **pHitRecord**.

Pour la méthode **intersectAny()**, on va juste vérifier s'il existe une intersection entre notre surface et le point en utilisant la méthode vue précédemment sans mettre a jour pHitRecord.

3 Première surface

Dans cette partie, on va créer une classe décrivant une sphère de manière implicite dérivant de la classe ImplicitSurface. Pour calculer la SDF on va calculer la distance entre le point et le centre de la sphère en utilisant **glm : :length()** auquel on soustrait le rayon. Si on est sur la surface alors la SDF sera égal à zéro, si la distance est inférieure au rayon alors on est a l'intérieur de la sphère et à l'extérieur sinon.

On obtient alors la figure ci-dessous :

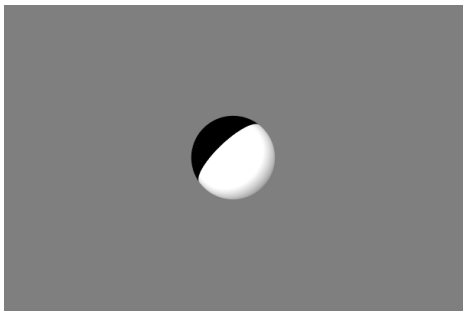

FIGURE 1 – Résultat question 2

4 Shading correct

Nous allons implémenter la fonction **evaluateNormal()** pour évaluer la normale de la surface au point d'intersection. Pour cela, j'ai utilisé la différence finie avec les dérivées partiels.

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

avec h notre minDistance et f(x) la fonction SDF

On va donc calculer les dérivées partiels de la SDF avec chaque composante et on retournera un vecteur composé de chaque composante normalisé. On obtient alors la figure ci-dessous :

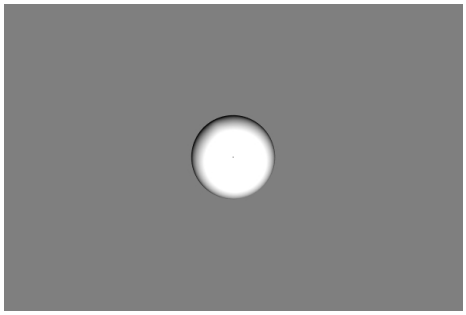

FIGURE 2 – Résultat question 3

5 Plus d'implicites

On va rajouter des surfaces implicites, pour ce faire on va utiliser la même méthode que pour la sphère. On va créer de nouvelle classe dérivant de la classe **ImplicitSurface** et calculer la **SDF** de la surface implicite voulue.

Pour créer de nouvelle surface implicite, nous allons utiliser le site donné dans le TP nous donnant les SDF de différentes surfaces implicites.

Nous avons alors commencé par créer la classe **roundBox**, représentant une boite avec des bords arrondis, elle prendra trois paramètres le **centre** de la boite, un vecteur représentant les **dimensions** la largeur,longueur et hauteur et le **rayon** de l'arrondi des bords. Pour calculer la SDF, on commence par calculer un vecteur (q) qui sera la distance entre le centre et les bords de la boites en prenant en compte les bords arrondis. Enfin, on fait **glm : :max(q,0.f)** pour ne garder que les points a l'extérieur de la boite et **glm : :min(glm : :max(q.x, glm : :max(q.y, q.z)), 0.0f) - rayon** va nous permettre de vérifier que l'on est sur la surface et on soustrait rayon pour avoir les bords arrondis. On obtient alors la figure ci-dessous :

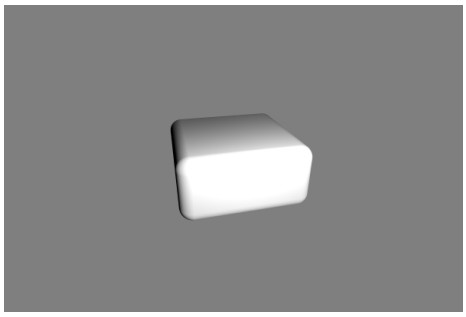

FIGURE 3 – Round Box

Ensuite, nous avons réaliser une classe créant une surface de **prisme triangulaire** prenant en paramètre le **centre** et un vecteur avec la **demi largeur** et la **demi hauteur** du prisme. Pour calculer la **SDF**, on va récupérer le max entre la distance entre la composante du point en z et la hauteur du prisme et la partie diagonale du prisme ce qui nous donne le résultat suivant.

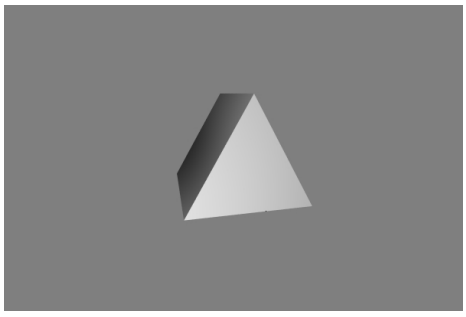

FIGURE 4 – Prisme triangulaire

Enfin, une classe **implicitCapedTous** représentant une sorte de demi anneau, avec en paramètre le **centre** de la surface, les **dimensions** de l'ellipse, le **rayon** de l'**anneau** et le **rayon** de l'**extrémité**. Pour calculer la SDF, on vérifie si le point est à l'intérieur ou a l'extérieur de la surface en comparant le produit entre la hauteur et la coordonnée x du point et le produit entre la largeur et la coordonnée y du point. Enfin, on calcule la distance du point situé sur la surface et les extrémité et on obtient la figure ci-dessous.

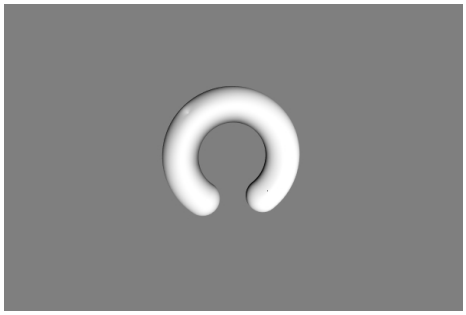

FIGURE 5 – Caped Torus