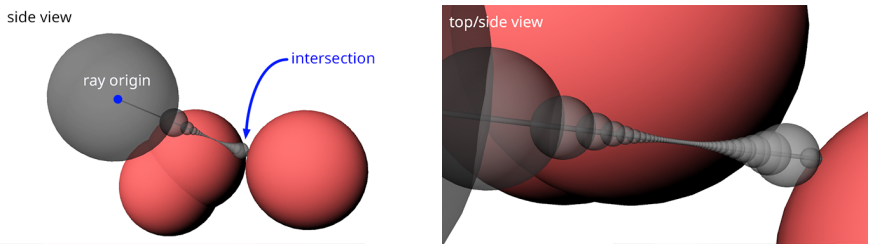
Université de Limoges
Faculté des Sciences & Techniques
Faculté des Sciences et Techniques de Limoges

ISIR - TP7 - Surface implicit et sphere tracing

Maxime MARIA

2022-2023

Dans ce TP, nous allons faire le rendu de surfaces implicites via un algorithme de *ray-marching* et plus particulièrement celui du *sphere-tracing* [Har95] (*cf.* Figure 1).



(a) (b)

Figure 1 – Illustration du *sphere-tracing* (Source : [STS]).

Comme vous allez le voir, l'architecture de base du projet n'est pas vraiment adaptée pour faire du rendu spécifiquement via cette méthode. Cependant, vous pourrez tout de même facilement implémenter le calcul d'intersection entre une surface implicite et un rayon. Par contre, pour aller plus loin (réflexion, réfraction, transformations, *etc.*) ou exploiter certains avantages du *sphere-tracing* (*eg.* l'anti-aliasing, le calcul d'ombres douces), il sera nécessaire de modifier l'architecture.

Dans ce TP, je vous guide beaucoup moins que d'habitude ! Utilisez le cours ainsi que les ressources données en fin de sujet pour vous aider dans le développement, notamment pour les fonctions de distance signées (SDF : *Signed Distance Function*). Vous devrez donc expliquer ce que vous avez compris dans le rapport de rendu final ! Ça vous entraînera pour la suite !

Les images de la figure 2 montrent quelques exemples de surfaces implicites rendues via *sphere-tracing*.

(a) (b) (c)

Figure 2 – Exemples de surfaces implicites rendues via *sphere-tracing* : (a) un blob, selon la SDF du cours 3 ; (b) le fameux Mandelbulb ; (c) plusieurs objets (SDF issues de [Qui]).

Pour commencer

Dans le dossier `objects`, vous trouverez une classe abstraite, `ImplicitSurface`, héritant de `BaseObject`. Elle possède quatre méthode :

- `intersect` et `intersectAny`, que vous connaissez bien désormais ;
- `_sdf` qui calcule la distance signée de la surface par rapport à un point (à implémenter dans les classes dérivées) ;
- `_evaluateNormal` qui évalue la normale à la surface en un point.

1 Le cœur de l'algorithme

Implémentez les méthodes `intersect` et `intersectAny` de la classe `ImplicitSurface`. Ces méthodes déterminent l'intersection entre un rayon et la surface. Il s'agit donc du cœur de l'algorithme de *sphere-tracing*.

2 Première surface

Créez une classe `ImplicitSphere` dérivant de la classe `ImplicitSurface`. Cette classe décrit donc une sphère de manière implicite, elle contient donc la position de son centre ainsi que son rayon. Implémentez la méthode `_sdf`. Testez votre programme, vous devriez voir une sphère, avec un mauvais shading.

3 Shading correct

Pour avoir un shading correct, il faut évaluer la normale de la surface au point d'intersection. Pour cela, implémentez la méthode `_evaluateNormal` de `ImplicitSurface` (oui, l'évaluation de la normale est la même quelque soit la surface !).

4 Plus d'implicites !

Ajouter au moins trois surfaces implicites différentes à votre programme ! Je vous laisse le choix !

Références

[Har95] John Hart. Sphere tracing : A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12, 06 1995.

[Qui] Distance functions. https://iquilezles.org/www/articles/distfunctions/distfunctions.htm. Accessed : 2022-03-24.

[STS] Rendering implicit surfaces and distance fields : Sphere tracing. https://www.scratchapixel.com/lessons/3d-basic-rendering/rendering-distance-fields/introduction. Accessed : 2022-03-24.