Ray Marching

Burgalat - Garcia

Table des matières

1 Tout commence par le commencement **2**
1.1 Le principe du Ray Marching . . . 2
1.2 Affichage . . . 2
1.3 SdF . . . 2
1.4 Affichage des lumières et ombres . . . 3

2 Opérations **3**
2.1 Opérations de base : . . . 3
2.1.1 Union . . . 3
2.1.2 Intersection . . . 3
2.1.3 Soustarction . . . 3
2.1.4 Rotations . . . 4
2.2 Amélioration des effets . . . 4
2.2.1 Fonctions de lissage - Smooth . . . 4

3 Optimisations **4**
3.1 Bounding Volumes Hierarchy (BVH-Trees) . . . 4
3.2 Multi-threading . . . 4
3.3 Chiffres . . . 4
3.4 Méthode naive . . . 4

1 Tout commence par le commencement

1.1 Le principe du Ray Marching

Le Ray marching consiste a faire du rendu 3D grace a des Signed Distances Functions (SDF). Qui sont des fonctions qui calculent la distance entre un point et un objet, avec une distance négative des que le point se trouve a l'interieur de l'objet.

Pour cela, pour chaque pixel de la camera on va envoyer un rayon depuis la camera passant par ce pixel. Pour savoir quel est la distance maximal que ce rayon peut parcourir, on calcule la distance minimal a tous les objets de la scene. On peut donc avancer de cette distance et on repette cela jusqu'a atteindre un objet ou dépasser une distance minimal. On peut donc colorier le pixel en fonction du resultat.

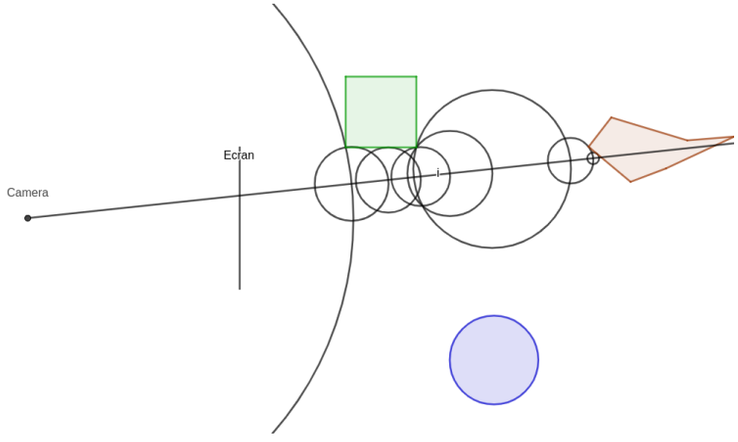

Figure 1 – Schéma rayon Ray Marching en 2D

1.2 Affichage

1.3 SdF

Le principe de la Signed Distance Function (SdF) est de calculer la distance entre un point et un objet, avec une distance négative des que le point se trouve a l'interieur de l'objet.

On peut donc donner à chaque forme de base une fonction distance afin de permettre, lors du lancer de chaque rayon, de savoir si le rayon atteint l'objet (la SdF renverra une valeur négative lorsqu'il arrivera à un point dans l'objet), où s'il ne touche pas (à partir d'une certaine distance atteinte par le rayon, on considère qu'aucun objet est atteint et alors on renvoie un fond uniforme).

Exemple de la sphère :

```
float SDF_sphere(coord p, coord centre, float rayon){
  float dist =
    sqrt((p.x-centre.x)*(p.x-centre.x)+(p.y-centre.y)*(p.y-centre.y)+(p.z-centre.z)*(p.z-centre.z));
  return dist - rayon;
}
```

Les formes de base implémentées sont :
— Plan
— Sphère
— Tor
— Boite
— Pyramide
— Cylindre

Exemple du triangle :

Le triangle est l'une des forme les plus importante en modélisation 3D car tous objet peut etre réprensenté facilement (avec plus ou moins de précision) grace a seulement des triangles.

1.4 Affichage des lumières et ombres

2 Opérations

2.1 Opérations de base :

2.1.1 Union

Afin de procéder l'union entre 2 objets/formes, il suffit de considérer à chaque fois la plus faible valeur entre la sdf d'un objet et celle de l'autre, ce qui nous permettra, pour chaque rayon de lancé, de capter la présence de la première forme présente sur son chemin.

```
float UnionSDF (float d1, float d2){
    return MIN(d1,d2);
}
```

2.1.2 Intersection

Au contraire pour l'Intersection, il suffit de considérer la distance maximale : Si un des 2 objets n'est pas présent, la fonction intersection renverra toujours le maximum des 2 sdf, soit une distance positive et ainsi aucun objet ne sera atteint. Pour les points sur lesquels les 2 objets sont présents, la fonction renverra bien une distance négative (car les 2 le sont) à l'arrivée du rayon en ce point.

```
float IntersectSDF (float d1, float d2){
    return MAX(d1,d2);
}
```

2.1.3 Soustarction

La soustraction de F1 par F2 se traduit à $F1_{\setminus F2}$
Il suffit donc de considérer les points où la sdf de F1 est positive, et celle de F2 négative.

```
float SubstractSDF (float d1, float d2){
    return MAX(d1, -d2);
}
```

2.1.4 Rotations

2.2 Amélioration des effets

2.2.1 Fonctions de lissage - Smooth

3 Optimisations

3.1 Bounding Volumes Hierarchy (BVH-Trees)

Arbres K-dimensionnels de priorisation des SdF

3.2 Multi-threading

3.3 Chiffres

3.4 Méthode naive

Nous avons sélectionner plusieurs scenes de references pour visualiser les performances de notre programme.

[insert scene]

Premiere implementation (sans reel opti) :

[mettre des graphiques en camenbert et tout cas]