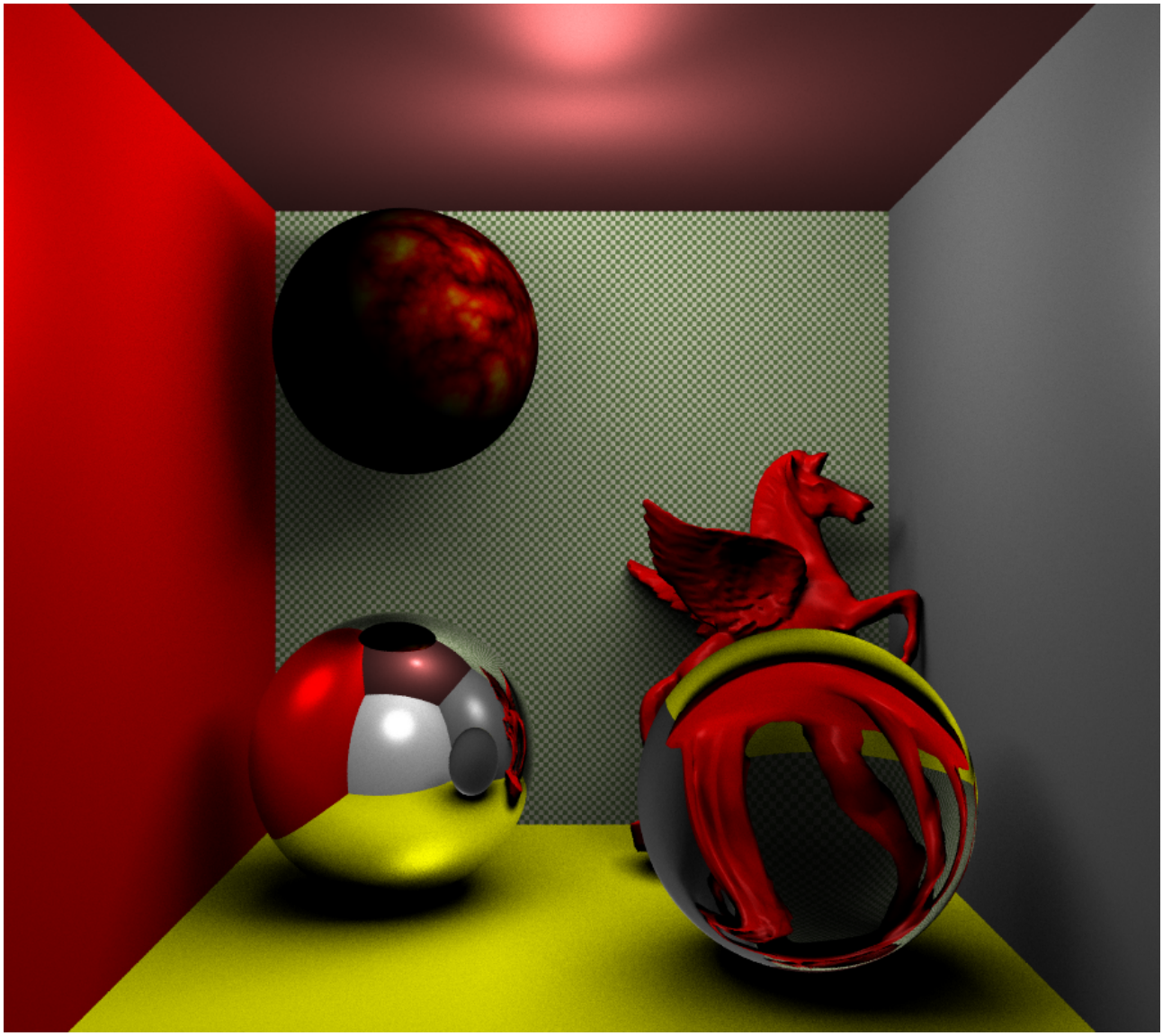
HAI719I – Programmation 3D

Rendu projet phase 4

Léo Hafdane – e22202516

16/01/2026



M1 Informatique – IMAGINE
Faculté des Sciences
Université de Montpellier.

1 Settings et présets

Dans la phase 3, j'ai introduit un système de Paramètres globaux (dans `Settings`) et de presets qu'on peut changer. Cette partie introduits plusieurs paramètres : `ENABLE_GLASS` , `AIR_INDEX_MEDIUM` et `MAX_LEAF_SIZE` ainsi que les nouveaux presets `PHASE_4_REFRACTION` et `PHASE_4_KDTREE` on peut toujours changer de preset avec p et P.

2 Réfraction

Pour la réfraction, je me suis aidé de cette partie de page wikipédia (en plus de celle-ci) pour comprendre le phénomène et de ce site pour m'aider à implémenter la réfraction en elle-même. Tout ceci se retrouve dans ma fonction `Scene::computeRefractionRay`.

Puisque ce calcul dépends de la densité du matériau que le rayon quitte et de celle dans lequel il entre, il faut que les rayons gardent en mémoire les objets par lequels ils sont passés ainsi que leur densité. La structure `Ray` contiens maintenant une liste initialisée avec un `index_medium` à 1, celui de l'air. Je suis parti du principe que la caméra ne se trouve jamais dans un solide.

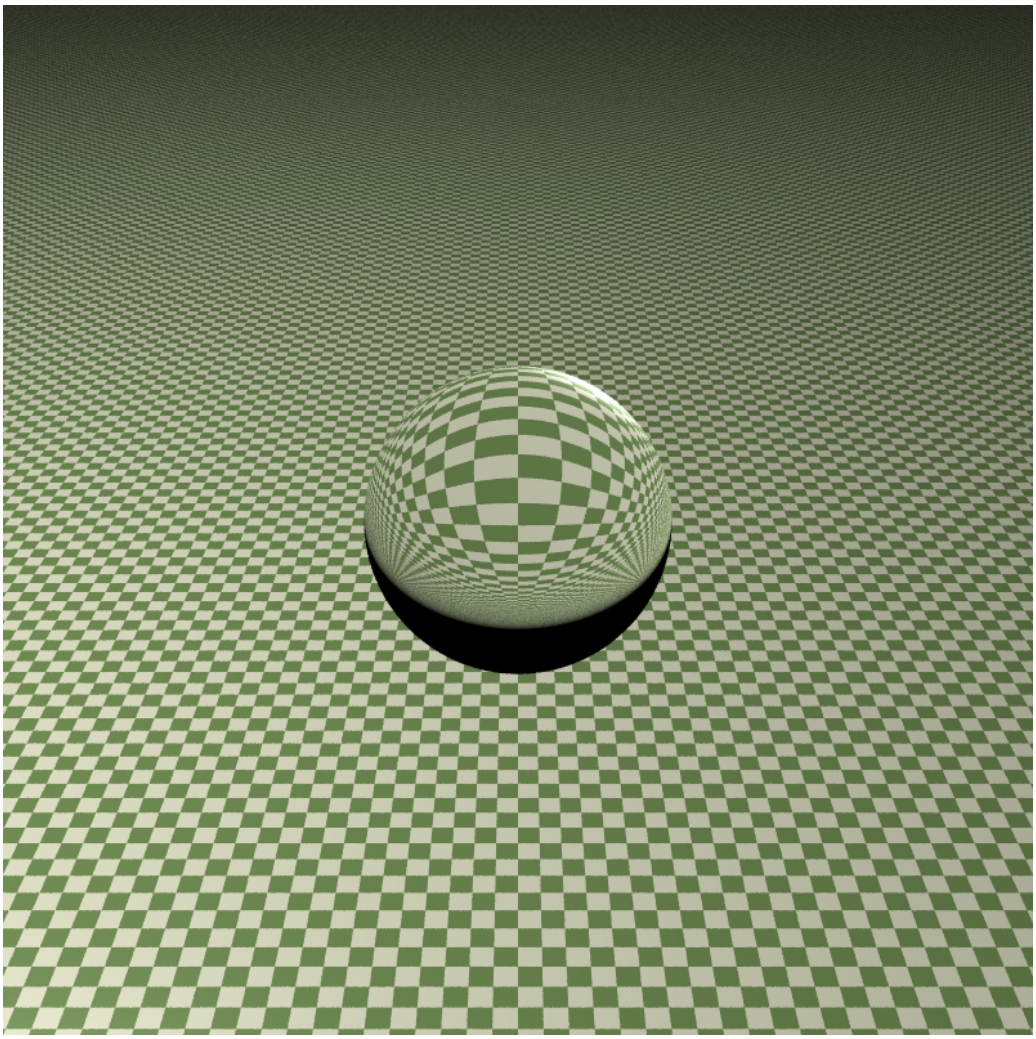


FIGURE 1 – Cliquez sur l'image pour voir une démonstration vidéo (boucle).
Sinon elle est ici

3 KdTree

3.1 Bounding box

Avant de faire mon KdTree, j'ai eu besoin de coder une axis-aligned bounding box (AABB). En effet, lors d'une intersection avec le KdTree, on vérifie dans un premier temps si le rayon touche la AABB. Cela nous permettra de savoir dans quelle branche aller pendant le calcul d'intersection. De plus, cela permets également d'éviter de faire des calculs d'intersection sur une mesh qu'on est sûr de ne pas toucher, accélérant ainsi tous les autres rayons.

Pour construire une AABB, il suffit d'itérer sur tous les triangles et de garder en mémoire les composants minimum et maximum de chaque axe. La AABB finale étant le cube formé par $min_{x,y,z}$ et $max_{x,y,z}$.

3.2 Construction

Un KdTree est défini par son ensemble de triangles T où $|T| = n$, une AABB, un paramètre α décrivant le nombre maximal de triangles dans une feuille, s son axe de coupe ainsi que deux enfants k_1 et k_2. L'idée du KdTree va de pair avec sa construction :

- Si $n < \alpha$ Alors le KdTree est une feuille, il n'a pas d'enfants mais garde son ensemble de triangles.
- Sinon, le KdTree doit avoir deux enfants, avec deux nouveaux ensemble de triangles et deux nouvelles AABB :
    1. Trier T et calculer C le centroïde médian sur l'axe s.
    2. Séparer T et deux tableaux T_1 et T_2 tq T_1 contiens tous les triangles qui ont au moins un point à gauche de C sur l'axe s (resp. T_2 à droite).
    3. Calculer les deux nouvelles AABB.
    4. Construire les deux enfants de la même manière mais avec $s = (s + 1)\%3$

3.3 Intersection

L'intersection consiste en un simple parcours d'arbre, Si le KdTree actuel est une feuille alors il teste l'intersection à la bounding box des deux enfants. Et les parcours en premier la AABB ayant le plus petit t et l'autre ensuite.

3.4 Benchmarks

Pour tester l'efficacité de mon KdTree, j'ai fais des benchmarks qui teste pour chaque mesh disponible une taille maximale de feuille de 0 et les premières puissance de 2. Cela m'a donné un gros fichier csv : *raw_mesh_benchmark.csv* avec 3 colonnes : `Mesh;MaxLeafSize;ElapsedMs`. En le lisant, on peut voir que les performances augmentent énormément entre une taille de feuille de 0 (pas de KdTree) et un taille de feuille de 1. Que ensuite, les performances stagnent relativement jusqu'a 8 triangles par feuille pour finalement augmenter jusqu'à atteindre les mêmes performances que sans KdTree.

3.4.1 Exemples Génération

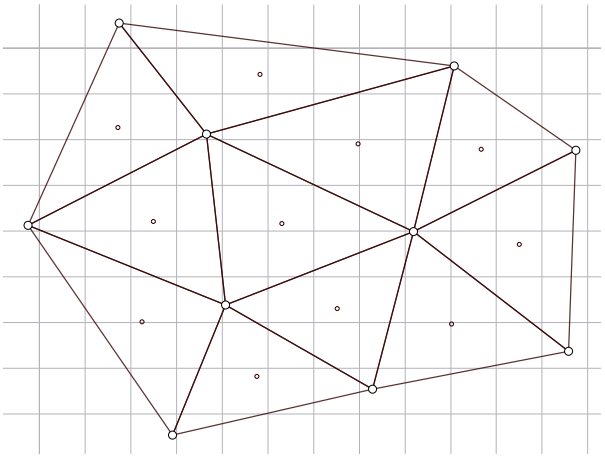


FIGURE 2 – Étape 0 de construction d'un Kd-Tree, on a 11 Triangles

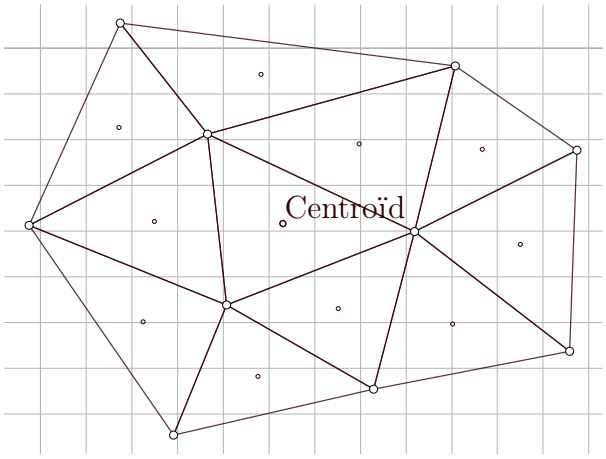


FIGURE 3 – Étape 1 de construction d'un Kd-Tree, le centroïde médian est marqué

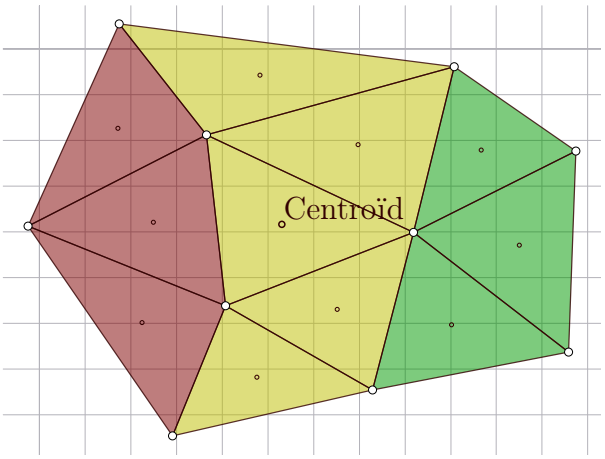


FIGURE 4 – Étape 2 de construction d'un Kd-Tree, le centroïde médian est marqué

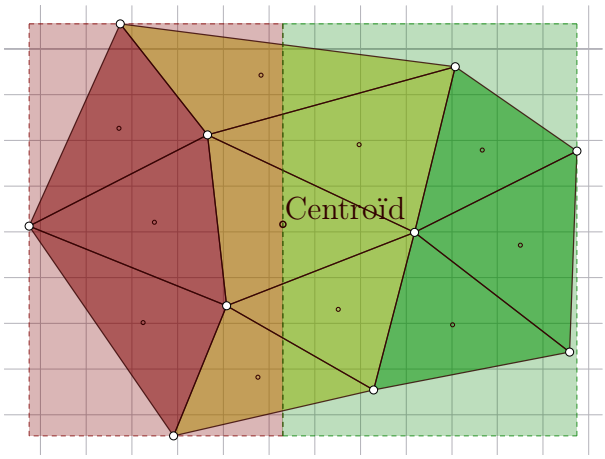


FIGURE 5 – Étape 3 de construction d'un Kd-Tree, le centroïde médian est marqué

On a maintenant une profondeur de KdTree, la racine étant la grande boite qui englobe tout les tirangles, T_0 Contiens les triangles colorés en rouge et jaune tandis que T_1 contiens ceux en jaune et vert. On peut également voir leur propre AABB réprésentés en rouge et vert.