

Projet de lancer de rayon – Phase 1 (40 pts) faite durant les 2 premiers TPs

La base de code raytracer-template.zip se trouve dans le même dossier que ce document.

Le première tâche est de lire et comprendre ce qui se passe dans le code. Vous trouverez le détail des classes dans le document overview.

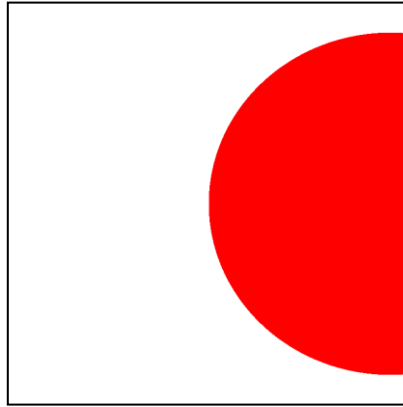
Pour la première partie de ce projet vous devez remplir les fonctions suivantes dans main.cpp et Object.cpp:

1. **(10 pts)** castRay()
 - 1.1. Cette fonction est appelée pour chaque rayon et devrait calculer l'intersection avec tous les objets de votre scène. Initialement, (cela changera par la suite) elle devra retourner la couleur de l'objet touché le plus proche. La couleur se trouve dans sceneObjects[i]->shadingValues.color pour l'objet i. Notez que vous pouvez utiliser (ce n'est pas le cas actuellement) _ID dans IntersectionValues pour enregistrer et retrouver l'objet touché correspond à chaque IntersectionValue.
2. **(5 pts)** raySphereIntersection()
 - 2.1. Calcule l'intersection d'un rayon avec une sphère
3. **(5 pts)** raySquareIntersection()
 - 3.1. Calcul l'intersection d'un rayon avec un carré
4. **(15 pts)** intersect()
 - 4.1. Remplir le résultat de IntersectionValues

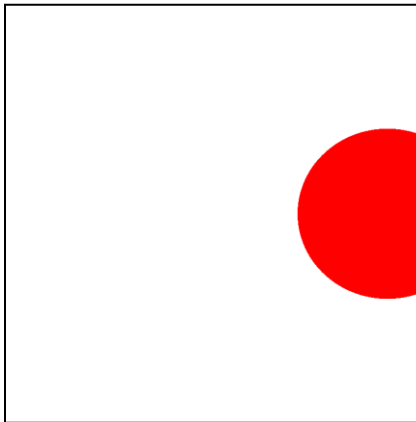
Voici quelques suggestions d'étapes à réaliser pour s'assurer du bon fonctionnement du projet.

Commencez par la scène _SPHERE. Dans ce cas, la fonction intersect() reste simple pour l'instant, vous pouvez donc vous concentrer sur les fonctions castRay() et raySphereIntersection(). Si elles fonctionnent correctement, vous devez obtenir un cercle rouge au centre de votre image.

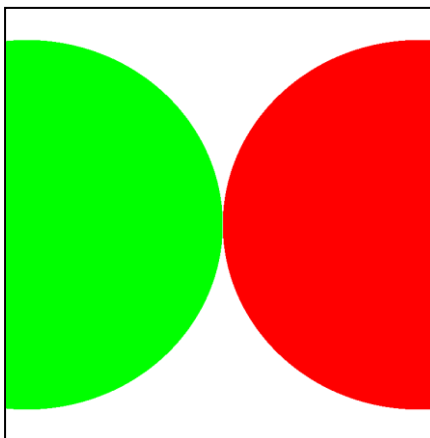
- Changer la position initiale de la sphère en changeant son centre de position à $\text{vec3}(1.0, 0.0, 0.0)$. Vous devez obtenir l'image suivante :



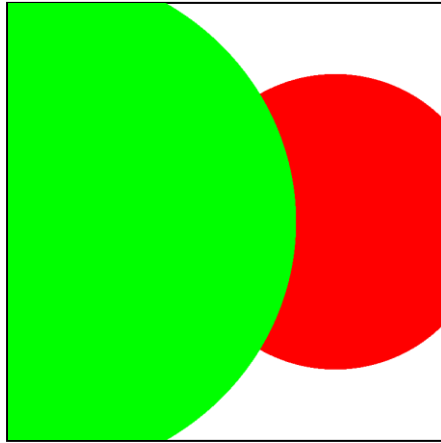
- Changée maintenant à $\text{vec3}(1.0, 0.0, 0.0)$ et radius 0.5. La sortie doit ressembler à l'image suivante :



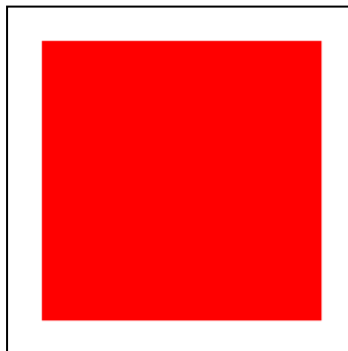
- Ajouter une second sphère (de couleur verte), appliquer une translation à l'une d'entre elles center $(1.0, 0.0, 0.0)$ et la deuxième center $(-1.0, 0.0, 0.0)$. Votre résultat doit ressembler à cela :



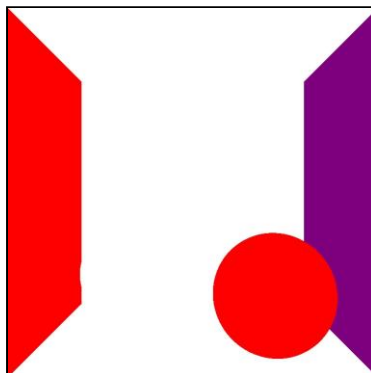
- Bougez, tournez :



- Passez à la scène `_SQUARE`. Maintenant vous devez écrire le code d'intersection d'un carré unité sur un plan XY-plane avec $Z=0$. C'est ce que `Square::raySquareIntersection` doit faire (Calculer l'intersection d'une droite et un plan ensuite limiter au carré unité). Maintenant mettre à jour `Square::intersect()` comme pour la sphère. Vous devriez maintenant avoir quelque chose comme l'image suivante (si vous changez la couleur du carré à rouge). Notez que les bords du carré ne vont pas jusqu'au bord de l'image.



- Bougez, tournez, comme la sphère, cela devrait fonctionner de la même façon.
- Maintenant passez à `_BOX` et VOILA une boîte de cornell apparaît.



En changeant les couleurs :

