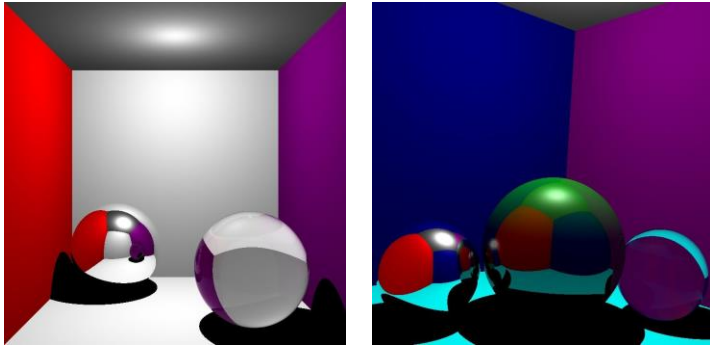


Projet de lancer de rayon



Base de code : Toutes les fonctions utilisées pour construire le lanceur de rayon sont inclus, les parties de code que vous devez compléter sont marquées par des TODO dans les commentaires.

Décompresser le fichier, dans le dossier résultant :

```
mkdir build
cd build
cmake ..
make -j
./raytracer
```

Voici une documentation des classes présentes.

- **ObjMesh** est une classe qui permet de charger un maillage mais qui contient pour l'instant une sphere de subdivision comme maillage potentiel.
- **Object** est une super classe pour tous les objets de la scène. Elle contient :
 - Le maillage nécessaire pour le rendu OpenGL
 - Les attributs nécessaire au calcul d'ombrage local
 - Les matrices comme C (modelview), C^{-1} , $(C^*)^{-1}$, et $(C^{-1})^T$. Celles-ci sont calculées par la fonction `setModelView`
- **Sphere** et **Square** sont des enfants de la classe **Object**.
 - Chacune contient une fonction `intersection` que vous allez implémenter.
 - Chacune contient une fonction plus générale de routine d'intersection `ray/sphere` et `ray/square` utilisées par la fonction `intersect`. Vous devrez compléter ces fonctions.
- Les fonctions principales se trouvent dans `main.cpp` :

- `rayTraceReceptor` et `write_image` sont des fonctions créant l'image de sortie `output.png` et peuvent être ignorées.
- `initGL` et `drawObject` gèrent la fenêtre OpenGL et peuvent être ignorées.
- `findRay` et `rayTrace` vous sont fournies. `findRay` fournit le rayon par l'intermédiaire de son point de départ et sa direction pour un pixel le résultat est un vecteur de dimension 2 (`ray_start`, `ray_dir`). `rayTrace` appelle `findRay` pour chaque pixel de votre image calculée, appelle `castRay`, pour chaque pixel et génère un png en sortie.
- Comme vous pouvez le constater, `castRay` est la fonction réursive au cœur de votre projet. Ses variables d'entrée sont le point de départ du rayon, la direction du rayon, le dernier objet intersecté par le rayon (demandez-vous pourquoi cela est primordial), et la profondeur de récursion.
- `shadowFeeler` envoie un rayon d'ombrage vers une lumière à partir d'un point. Elle prend également en entrée l'objet envoyant le rayon d'ombrage (demandez-vous pourquoi cela pourrait être utile).
- Une fonction `castRayDebug` est incluse pour lancer un seul rayon et observer le résultat. Cette fonction pourrait vous être utile pour créer quelque chose de similaire (code non fourni).

Scènes fournies : Vous trouverez 3 scènes une sphère, un cube, et une boîte de cornell : 6 cotés, 2 sphères (une miroir et une en verre), et une lumière ponctuelle unique au centre du plafond. ***Vous serez noté sur la scène de la boîte de cornell.***

Fonctionnement du programme : Votre programme vous fournit une fenêtre OpenGL avec votre scène. **Pour rendre votre vue courante en utilisant votre lanceur de rayon appuyez sur « r ».** '1', '2', '3' permet de changer de scène.