Raytracing - TD 02

Premier raytracer

L'objectif de ce TD est d'implémenter des fonctions testant l'intersection entre un rayon et des formes géométriques, afin de pouvoir afficher cette géométrie à l'écran.

Exercice 01 – Structures pour un raytracer

Vous devez commencer par créer des structures pour représenter des objets et concepts à la base d'un algorithme de lancer de rayons.

A faire :

01. Créez le fichier */include/raytracer.h* , dans lequel vous définirez une structure `Ray` , qui servira a représenter et stocker un rayon.
Un rayon est défini par une position d'origine et une direction.
Pensez à ajouter les lignes nécessaires pour que ce fichier ne soit importé qu'une seule fois.

02. Dans le fichier */include/raytracer.h* , créez une structure `Intersection` .
Une intersection se définira par sa position et la couleur de la surface à l'intersection.

03. Créez une un fichier */include/shape.h* , et créez dedans la structure `Sphere` .
Une sphère est définie par un centre un rayon, et une couleur.

Optionel :

04. Ajoutez les fonctions suivantes, pour créer plus facilement ces objets :
- createRay(…)
- createSphere(…)

(Je vous laisse le soin de trouver les signatures de ces fonctions)

Exercice 02 – Implémentation des fonctions d'intersection

Il est maintenant temps d'implémenter et de tester les fonctions d'intersection.

A faire :

01. Dans le fichier */include/raytracer.h* , ajoutez la signature de la fonction suivante :

```
int intersectsSphere(Ray r, Sphere s,
    Intersection* intersection, float* t);
```

Cette fonction teste si le rayon intersecte une sphère, et remplira la structure le cas écheant.
Si c'est le cas, elle devra retourner 1, après avoir calculé la valeur du paramètre `t` .
Sinon, elle devra retourner 0.

02. Dans le fichier /src/raytracer.c, implémentez cette fonction.
Repartez des équations d'intersection que vous avez trouvé dans l'exercice 04 du TD 01.

Optionel :

03. Dans votre fichier main.c , testez vos intersections en créant plusieurs sphères et rayons avec des paramètres différents, et en appelant ces fonctions avec ces objets.

Exercice 03 – Structure de la scène

On appelle scène l'ensemble des objets qu'on désire potentiellement voir apparaître à l'écran, en fonction de la position d'une caméra.

A faire :

01. Dans *raytracer.h* , créez une structure `Scene` , qui contiendra un tableau de `Sphere`. (Un tableau de taille fixe avec un compteur conviendra pour l'exercice)

02. Ajoutez la fonction `Scene createScene()`.
Cette fonction créera une `Scene` , et initialisera les champs de la structure nouvellement crée.

03. Ajoutez les fonctions :
```
void addSphereToScene(Scene* scene, Sphere s);
```

Vous devrez, comme vu précédemment, définir la signature de la fonction dans le header, et en donner l'implémentation dans le fichier *.c* correspondant.

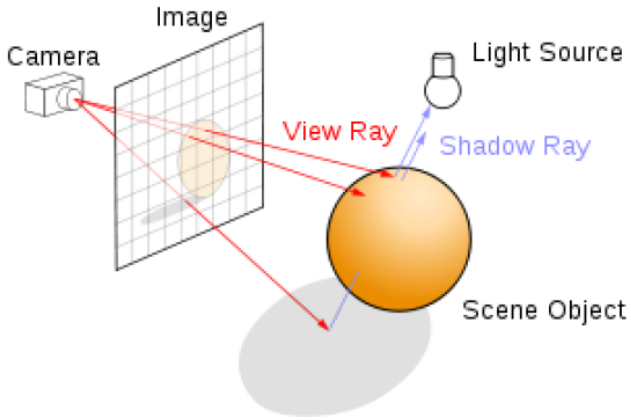
04. Écrivez la fonction :
```
int throwRayThroughScene(
    Ray r, Scene scene, Intersection* intersectionNearest)
```

Cette fonction boucle sur les objets d'une scène afin de trouver l'intersection la plus proche pour le rayon donnée en paramètre. La fonction renvoie 1 si il y a intersection, 0 sinon.

Exercice 04 – Mise en place du raytacer

Vous avez maintenant toutes les briques en place pour effectuer votre premier raytracing.

Il reste toutefois une dernière question à résoudre :
Comment définir la position et la directions de rayons que vous allez lancer ?



Il faut considérer que l'image affichée à l'écran est représentée par un quad placé juste devant la caméra, et que cette dernière va lancer des rayons a travers chaque pixel de l'image.
Ces rayon détecteront chacun si ils entrent en collision avec un objet de la scène.

Pour ce premier raytracer, nous utiliserons les valeurs suivantes :
- position de la caméra : (0, 0, 0)
- direction de la caméra : (0, 0, -1) (aussi appelée forward vector)
- image positionnée sur le plan Z = -1, dans un carré de taille 2 centré en (0, 0, -1).

Cela implique que les cordonnées des coins du carré seront :
(-1, -1, -1), (1, -1, -1), (1, 1, -1) et (-1, 1, -1)

Votre **framebuffer** (tableau contenant les couleurs des pixels) sera donc représenté virtuellement par ce quad. Chaque pixel possède donc une position 3D située dans ce quad virtuel, et les rayons seront lancés à travers ces pixels, depuis la caméra.

Pour stocker et afficher une image à l'écran, vous utiliserez à nouveau la librairie SDL, pour stocker et afficher une image à l'écran. Un framebuffer se résume à une `SDL_Surface`.

Pour chaque pixel (i, j) du framebuffer, il sera aisé d'obtenir
- sa position x dans le repère de la scène : x = -1 + 2 * (i / imageWidth)
- sa position y dans le repère de la scène : y = -(1 + 2 * (j / imageHeight))
- sa position z dans le repère de la scène : z = -1

- le rayon qui le traverse, ayant pour origine (0, 0, 0) et passant par le point (x, y, z) :
(x, y, z) – (0, 0, 0) = (x, y, z)

A faire :

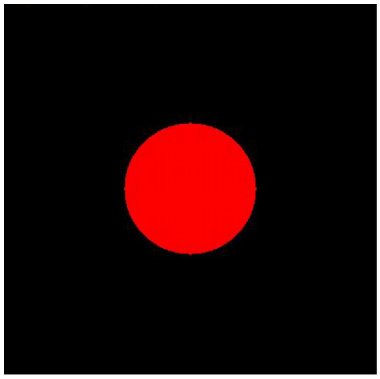
01. Copiez les fichiers *doc/sdl_tools.h* et *doc/sdl_tools.c* dans *include/* et *src/* .
Ils fournissent une fonction `putPixel()` qui remplira un objet de type `SDL_Surface`, avec une couleur donnée, à une position de pixel donnée.

02. Dans les fichiers *raytracer.h* et *raytracer.c* , créez la fonction :
`void simpleRaytracing(const Scene* scene, SDL_Surface* framebuffer)`

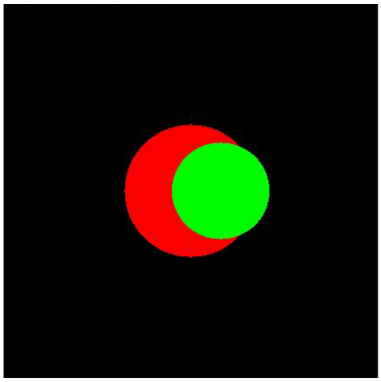
Cette fonction itère sur chacun des pixels du framebuffer :
Pour chaque pixel (i, j) elle calcule le rayon R(i, j) selon les formules donnée précédemment, puis envoie ce rayon dans la scène.
Si une intersection est trouvée, elle place la couleur du point d'intersection à la position (i, j) du framebuffer (avec la fonction `putPixel` de *sdl_tools.h*). Sinon, on touche pas a ce pixel.

03. Remplacez le contenu de votre fichier *main.c* par celui du fichier *doc/main_raytracer.c* .
Ce nouveau template initialise la SDL et crée une fenêtre, un framebuffer, et se charge de l'afficher dans une mini boucle de rendu.

04. Ajoutez dans votre scène une sphère rouge de rayon 1 à la position (0, 0 , -3).
04b. Appelez la fonction `simpleRaytracing` sur votre scène, et lancez votre programme.
Vous devriez voir un cercle rouge s'afficher à l'écran :



05. Ajoutez dans votre scène une sphère verte de rayon 0.5, à la position (0.25, 0, -2).
Vous devriez voir la deuxième sphère s'afficher juste devant la première :



Optionel :

06. Inversez les deux ajouts de sphères, et constatez que le résultat est parfaitement identique, confirmant ainsi que les intersections détectées sont bien les plus proches de la caméra.

07a. Ajoutez d'avantage de sphères dans votre scène
07b. Vérifiez que tout s'affiche correctement.

Félicitations, vous venez de développer votre tout premier raytracer.
Ce dernier est encore très rudimentaire (pas d éclairage où de réflexions, caméra fixe), mais la suite de cette série de TD vous montrera comment l'améliorer.