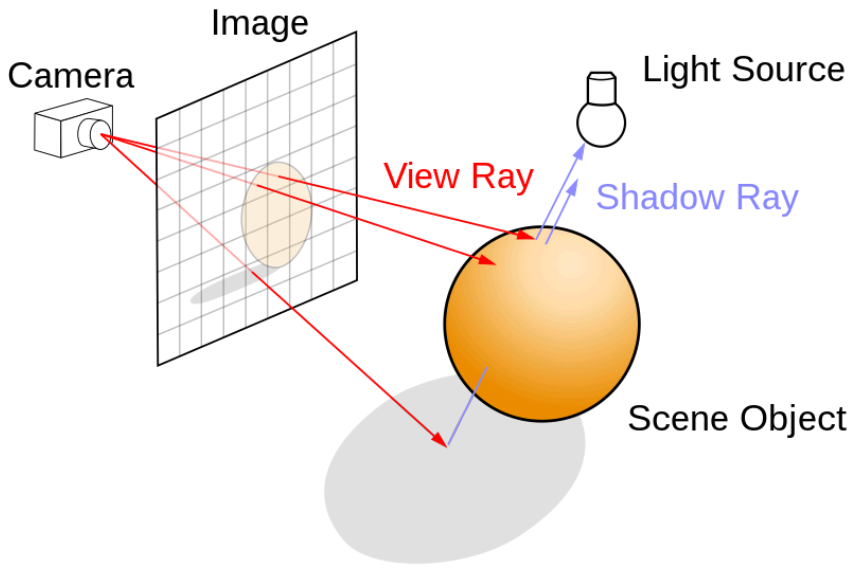
Documentation du Projet Raytracer



Introduction

Le projet Raytracer est une application de rendu graphique qui simule le comportement de la lumière pour créer des images 2D ou 3D. Il est largement utilisé dans les domaines de l'animation, des effets spéciaux, du design industriel et de la visualisation scientifique.

Le raytracing repose sur des principes physiques pour calculer comment la lumière se propage, se reflète et se réfracte à travers une scène virtuelle. En envoyant des rayons de lumière depuis une source (comme une caméra virtuelle) et en les suivant à mesure qu'ils interagissent avec les objets de la scène, le raytracing peut générer des images réalistes avec des effets de lumière, d'ombre et de réflexion complexes.

Utilisation du Projet Raytracer

```
Terminal
~/B-OOP-400> ./raytracer --help
USAGE: ./raytracer <SCENE_FILE>
  SCENE_FILE: scene configuration
```

Le projet Raytracer est conçu pour être utilisé comme une bibliothèque logicielle dans vos propres projets de rendu graphique. Voici comment l'utiliser :

1. **Intégration dans votre projet :** Incluez les fichiers sources du projet Raytracer dans votre projet existant ou créez un nouveau projet en utilisant la bibliothèque Raytracer.

2. **Définition de la scène :** Définissez les objets de la scène que vous souhaitez rendre, tels que des sphères, des cubes, des plans, etc. Spécifiez également les propriétés des matériaux, telles que la couleur, la réflectivité, la transparence, etc.

3. **Traitement de l'image :** Enregistrez l'image rendue dans le format PPM. Vous pouvez également appliquer des post-traitements tels que l'anti-crénelage, le flou, etc., selon vos besoins.

Configuration de la scène

Configuration de la caméra

La section `camera` comprend les paramètres suivants :
- `resolution` : La résolution de l'image à générer, spécifiée en largeur (`width`) et en hauteur (`height`).
- `position` : La position de la caméra dans l'espace 3D, spécifiée en coordonnées x, y et z.
- `rotation` : La rotation de la caméra autour de ses axes x, y et z.
- `fieldOfView` : Le champ de vision de la caméra, spécifié en degrés.

Primitives dans la scène

La section `primitives` contient une liste de primitives présentes dans la scène. Dans cet extrait, nous avons des sphères et des plans. Chaque primitive est définie par :
- Pour les sphères : les coordonnées du centre (`x`, `y`, `z`), le rayon (`r`) et la couleur (`r`, `g`, `b`).
- Pour les plans : l'axe de l'orientation (`axis`), la position le long de cet axe (`position`) et la couleur (`r`, `g`, `b`).

Configuration de l'éclairage

La section `lights` comprend les paramètres suivants :
- `ambient` : “YES” si elle doit être présente. “NO” pour le cas contraire
- `ambient_intensity` Le facteur multiplicatif de la lumière ambiante.
- `directional` : Une liste de lumières directionnelles, spécifiées par leurs coordonnées (`x`, `y`, `z`).

Ces informations définissent la scène dans laquelle le raytracer va générer une image en utilisant les paramètres de la caméra, les primitives définies et la configuration de l'éclairage spécifiée dans ce fichier de configuration.

Exemple de configFile :

```
camera : {
    resolution = { width = 600; height = 337; }; // Définit la résolution de la caméra
    position = { x = 0; y = 0; z = 0; }; // Définit la position initiale de la caméra
    rotation = { x = 0; y = 0; z = 0; }; // Définit la rotation initiale de la caméra
};

primitives : {
    spheres = ( // Définit les sphères dans la scène
        { x = 60; y = 5; z = 40; r = 2.0; color = { r = 255; g = 64; b = 64; }; }, //
Première sphère : position et couleur
        { x = -40; y = 20; z = -10; r = 1.0; color = { r = 64; g = 255; b = 64; }; }, //
Deuxième sphère : position et couleur
        { x = -5; y = 0; z = -4; r = 0.5; color = { r = 11; g = 0; b = 255; }; } //
Troisième sphère : position et couleur
    );

    planes = ( // Définit les plans dans la scène
        { axis = "Y"; position = -1.0; color = { r = 150; g = 150; b = 150; }; rotation =
"X"; rotation_degree = 1.0; } // Plan : axe, position, couleur et rotation
    );

    cylinders = ( // Définit les cylindres dans la scène
        { pos_x = 2; pos_y = 0; pos_z = -4; axis_x = 0; axis_y = 1; axis_z = 0; radius =
0.5; height = 1.0; color = { r = 255; g = 0; b = 0; } rotation = "X"; rotation_degree =
45.0; } // Cylindre : position, axe, rayon, hauteur, couleur et rotation
    );

    cones = ( // Définit les cônes dans la scène
        { pos_x = 0; pos_y = 1; pos_z = -3; axis_x = 0; axis_y = 1; axis_z = 0; angle = 4.0;
height = 1.0; color = { r = 255; g = 255; b = 0; }; rotation = "Z"; rotation_degree = 10.0;
} // Cône : position, axe, angle, hauteur, couleur et rotation
    );
};

lights : {
    ambient = "YES"; ambient_intensity = 0.2 //Entre 0 et 1 // Définit l'éclairage ambiant
et son intensité
    directional = ( // Définit les lumières directionnelles
        { x = -2.0; y = 0.0; z = 1.0;} // Lumières directionnelles : direction
    )
};
```

Qu'est-ce que le Raytracing ?

Le raytracing est une technique de rendu qui simule le comportement de la lumière dans une scène virtuelle en suivant la trajectoire des rayons lumineux à travers celle-ci. Voici quelques concepts clés du raytracing :

- **Rayon primaire :** Le rayon de lumière émis par la caméra virtuelle et envoyé dans la scène pour déterminer la couleur des pixels de l'image.

- **Intersections:** Les points où les rayons lumineux rencontrent les objets de la scène. À chaque intersection, des calculs sont effectués pour déterminer la couleur et l'intensité de la lumière réfléchie, réfractée ou diffusée.

- **Réflexion :** Le phénomène par lequel la lumière rebondit sur une surface, produisant des reflets de la lumière incidente.

- **Réfraction :** Le phénomène par lequel la lumière change de direction lorsqu'elle traverse un matériau transparent, tel que le verre ou l'eau.

- **Ombres :** Les zones de la scène qui ne reçoivent pas de lumière directe en raison de l'obstruction par d'autres objets.

Le raytracing permet de produire des images de haute qualité avec des effets visuels réalistes tels que les ombres douces, les réflexions et les réfractions, ce qui en fait une technique populaire dans les industries de la création graphique.

1. `RaytracerException` : Cette classe est une exception personnalisée qui hérite de `std::exception`. Elle est utilisée pour gérer les erreurs spécifiques à votre projet de ray tracing. Elle contient deux attributs privés : `_message` et `_type` qui stockent respectivement le message d'erreur et le type d'erreur.
1. `PrimitiveManager` : Cette classe est un gestionnaire pour les primitives dans votre projet. Il s'agit d'une classe template qui peut gérer différents types de primitives. Les détails spécifiques de son fonctionnement dépendent de l'implémentation dans le fichier source correspondant.
2. `check_and_parse` : Cette classe est responsable de la vérification et de l'analyse des entrées. Elle contient plusieurs méthodes pour obtenir des informations sur les primitives, la position de la caméra, la résolution de la caméra et la rotation de la caméra. Elle stocke ces informations dans des vecteurs privés.
3. `Raytracer` : Cette classe est le cœur de votre projet de ray tracing. Elle contient une méthode `ray_color` qui calcule la couleur d'un rayon, et une méthode `run` qui démarre le processus de ray tracing. Elle utilise une instance de `check_and_parse` pour obtenir les informations nécessaires pour le ray tracing.
4. `ILight` et `DirectionalLight` : Ces classes sont liées à la gestion de la lumière dans votre projet de ray tracing. `ILight` est une interface qui définit les méthodes que toutes les classes de lumière doivent implémenter. `DirectionalLight` est une classe qui implémente cette interface pour représenter une source de lumière directionnelle.