

COMMENT IMPLÉMENTER DE NOUVELLES SOURCES DE LUMIÈRE

Projet RayTracer : Marius Pain, Landry Gigant, Thomas Boué & Aubane Nourry

Introduction

L'intégration de nouvelles sources lumineuses dans le projet RayTracer d'Epitech nécessite une approche méthodique et structurée, étroitement alignée avec l'architecture existante, afin de garantir leur fonctionnement correct au sein du système de rendu. Dans cette section, nous explorerons les étapes essentielles pour implémenter de nouvelles lumières dans le cadre du projet RayTracer. Nous mettrons en évidence les différentes composantes nécessaires et les conventions établies pour assurer la compatibilité de ces sources lumineuses avec notre projet, tout en offrant une flexibilité permettant de créer divers types de lumières.

Créer un nouveau type de lumière

Dans notre RayTracer, une source de lumière prend en compte de multiples informations liées à la scène afin de permettre la génération point par point de l'image, colorée. Chaque scène peut contenir une multitude de sources de lumière ayant des comportements différents, définies dans un fichier de configuration.

(Voir le pdf sur les fichiers de configuration pour plus d'informations)

1. Les différents éléments

Afin d'implémenter votre classe vous devrez d'abord créer la méthode héritée via l'interface, puis vous verrez comment vous pouvez adapter l'analyse syntaxique des fichiers de configuration à votre source de lumière pour que votre bibliothèque puisse être chargée et utilisée correctement par notre raytracer.

a. Méthode héritée

En héritant de l'interface [ILight](#), vous héritez de la méthode suivante qu'il vous faudra implémenter :

- **Illuminate** : La méthode est appelée par le constructeur de l'image pour calculer l'intensité de la lumière à un point donné.


La méthode **Illuminate** prend en paramètres:

- `Math::Point3D point` : Correspondant au point duquel on essaye de calculer la couleur.
- `const std::shared_ptr<IMaterial> &material` : Correspondant au matériau de la primitive concernée.
- `const std::vector<std::shared_ptr<RayTracer:: IPrimitive>> &primitives` : Correspondant aux autres primitives contenues dans la scène.
- `Math::Vector3D normal` : Correspondant à la normale de la primitive passant par le point donné.

La fonction **Illuminate** retourne:

- `Math::Vector3D` : Correspondant aux valeurs R G B à ajouter à la couleur du point dû à l'effet de la lumière.

(Voir le pdf sur l'implémentation des matériaux et des primitives pour plus d'informations)

 **Astuce** : Pour éviter de réécrire plusieurs fois les méthodes de l'interface que vous utilisez, vous pouvez créer utiliser la classe abstraite [ALight](#) ou en créer d'autres.

b. Extern "C"

Dans votre classe principale, vous allez devoir rajouter une structure `"extern "C"` qui contient à l'intérieur obligatoirement au moins les fonctions :

- `"Light::*votre classe*> *loadLightInstance()"` : La fonction doit créer une nouvelle instance de votre lumière et la retourner en tant que pointeur.
- `"const std::string &get_name(void)"` : La fonction doit retourner une chaîne de caractère correspondant au nom de votre source lumineuse (ex : "DirectionalLight"...)
- `"__attribute__((constructor)) void initsharedlibrary()"` : Cette fonction sera appelée lorsque votre bibliothèque sera chargée par le programme, vous ne devez mettre aucune logique nécessaire à votre bibliothèque dedans, cependant vous pouvez faire en sorte que celle-ci écrive un message pour dire que celle-ci est chargée.
- `"__attribute__((destructor)) void destroysharedlibrary()"` : Cette fonction sera appelée lorsque votre bibliothèque sera déchargée par le core, vous ne devez mettre aucune logique nécessaire à votre bibliothèque dedans, cependant vous pouvez faire en sorte que celle-ci écrive un message pour dire que celle-ci est déchargée.

c. Adapter l'analyse syntaxique du fichier de configuration


Une fois que votre librairie dynamique est créée, vous arrivez à la dernière étape. Il ne reste plus qu'à modifier la partie analyse syntaxique du programme pour pouvoir renseigner votre nouveau type de lumière dans les fichiers de configuration. Lors de la création de la scène, chaque élément est chargé depuis le fichier de configuration par la classe [Scene](#) implémentée dans [Scene.cpp](#).

Il vous suffira d'ajouter du code ou une fonction dans la méthode déjà existante:

```
void RayTracer::Scene::createLights(libconfig::Setting &lights,
std::shared_ptr<core> core);
```

Afin de:

- Récupérer les attributs de l'objet qu'est votre lumière depuis le fichier en utilisant la libconfig (*Voir Comment Installer la libconfig*)
- Construire votre objet en appelant le ou un des constructeurs que vous avez implémenté pour votre classe
- Enregistrer votre lumière en utilisant son nom comme clé d'enregistrement dans la liste des lumières.

 **Attention :** Pour éviter tout problème à la compilation, n'oubliez pas d'inclure la définition de votre classe en haut du fichier de la scène, sinon le constructeur de la scène ne pourra appeler votre constructeur correctement.

2. Conclusion

Vous avez désormais toutes les clefs en main pour implémenter un nouveau système ou source de lumière pour notre raytracer, n'hésitez pas à faire une pull request pour ajouter votre lumière si vous souhaitez contribuer au projet. Nous vous remercions par avance de votre contribution.