IMAC
INGÉNIEUR
ESIPE

Synthèse d'image II
Programmation C++
Projet
ImacDream/ImacNightmare

Au terme de vos tps de *Synthèse d'Image*, vous serez capable de faire un programme affichant des scènes 3D et avec vos nouvelles compétences en programmation C++, vous avez le bagage suffisant pour nous faire rêver ! Ce projet a pour but de faire un jeu contemplatif qui nous fait rêver ou tout au contraire nous plonge dans un cauchemar...

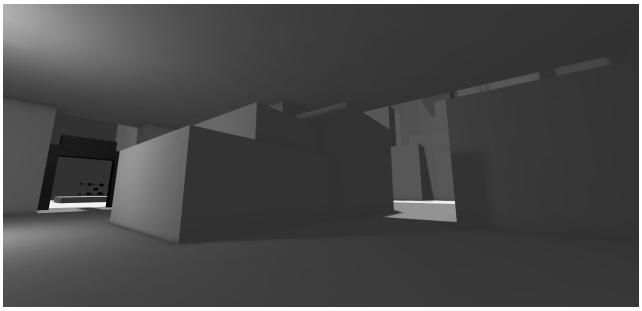

Figure 1 – Toon Garden - Unity Asset Store

Figure 2 – The Connection - Indie Game by BK-TN

Vous allez donc devoir réaliser un programme en C/C++ qui va permettre à un utilisateur d'explorer une scène 3D. Le but étant :

- De vous familiariser avec la programmation orientée objet.
- De vous faire utiliser les fonctions de dessin OpenGL 3+ (shaders, buffers...).
- De mettre en application des principes d'architecture logiciel.

Évaluateurs :
Venceslas Biri : biri@u-pem.fr
Vincent Nozick : vincent.nozick@univ-eiffel.fr
Sylvain Cherrier : sylvain.cherrier@univ-eiffel.fr
Steeve Vincent : steeve.v91@gmail.com

Modalités de rendu

- **Nombre de personnes par projet :** 2 ou 3
- **Livrable :**
  — Vous devrez utiliser le gestionnaire de version Git et héberger votre dépôt sur une plateforme en ligne, telle que GitHub, GitLab ou Bitbucket. Nous vous demanderons de nous envoyer le lien vers ce dépôt **rapidement**, afin que nous ayons accès à l'historique de vos commits.
  — Les sources C/C++ ainsi qu'un système de compilation permettant de compiler sur les machines de l'université (Linux avec les packages SDL (1 et/ou 2), SFML, OpenGL, GLEW, GLU, GLUT/FreeGLUT, SDL_image, SDL_ttf, SDL_mixer, Assimp disponibles ainsi que make, g++ et cmake bien sûr)
  — Un rapport au format PDF (maximum 12 pages) décrivant le projet (en particulier les aspects qui diffèrent du sujet initial, inutile de répéter le sujet), votre méthode de travail, des détails techniques si vous souhaitez détailler une fonctionnalité, les difficultés rencontrées et enfin les améliorations possibles.
- **Soutenance :** Si les conditions le permettent, vous pourrez défendre votre projet durant une soutenance en faisant une démonstration de votre projet et en reprenant les différents points du rapport. Votre présentation durera 10 minutes et sera suivi d'une série de questions du jury pour compléter votre soutenance.
- **Date de rendu (non définitive) :** Janvier 2021
- **Date de soutenance :** Janvier 2021

1 Jeu

Le programme devra, dans un premier temps, offrir une expérience contemplative, on explore une scène à la première personne dans le but de produire un sentiment, que ce soit de l'admiration, de la joie, de l'effroi... bref faite une oeuvre d'art en 3D. Puis dans un second temps, il faudra l'améliorer pour en faire un jeu, rajouter un objectif qui poussera l'utilisateur à fouiller l'ensemble de la scène.

1.1 Fonctionnalité

Milestone 0 (Charger une scène 3D)

- Charger l'ensemble des modèles 3D (positions, UVs, normals...) et positionner ces éléments dans la scène.
- Charger les matériaux (paramètres des interactions lumières-matières des modèles 3D : coef de diffusion, albedo, et textures...).
- Charger les lumières (au moins une lumière directionnelle).

Vous devrez utiliser un fichier de description de scène avec le format de votre choix. Ce fichier doit contenir la liste des modèles 3D, leur agencement dans la scène ainsi que les matériaux et le descriptif des lumières.

Milestone 1 (Explorer la scène)

- Explorer la scène en vue à la première personne : Se déplacer avec le clavier et regarder autour de soi avec la souris.

Milestone 2 (Interagir avec la scène)

- Animer un objet ou changer l'aspect de la scène en jouant avec les matériaux et les lumières en cliquant sur un objet ou en allant dans un endroit spécifique
- Utiliser des portails pour passer d'un monde à un autre / d'une scène à une autre

Milestone 3 (Jouer)

- Résoudre un(e) énigme/puzzle ou retrouver une série d'objets cachés
- Trouver une sortie ou une destination spécifique

Fonctionnalités additionnelles

- Une bonne expérience immersive possède une bonne ambiance sonore, utilisez *SDL_mixer* pour jouer une musique et créer des effets sonores
- Utiliser la librairie *Assimp* ou le format de fichier glTF pour charger des objets 3D
- Ajouter un narrateur (textuel ou auditif)
- Gérer les collisions

- Utiliser des shaders animés pour créer par exemple de la végétation ou de l'eau

2 Développement

Pour faire ce jeu vous devez respecter les contraintes décrites dans cette partie, tout écart devra être au préalable validé par vos évaluateurs et justifié dans le rapport et la soutenance.

2.1 Spécifications pour la Synthèse d'Image

- Utilisez des *Vertex Buffer Object*, des *Vertex Array Object* et le cas échéant des *Index Buffer Object* et bien sûr des shaders.
- Votre monde devra contenir a minima une lumière directionnelle.
- Vous proposerez à minima un modèle d'illumination de type Blinn-Phong avec texture. La texture représentera, pour chaque fragment, le coefficient de réflexion diffuse (on parle d'une texture d'albedo).
- Vous devrez charger un fichier texte qui décrit chaque meshes. Pour chaque mesh, il faut :
  — un chemin relatif vers le fichier du modèle 3D
  — des infos de transformations (translations, rotations, scales)
  — des informations de shaders (texture diffuse, albedo, ...)
- Vous devrez gérer les entrées utilisateurs

2.1.1 Conseils et extensions pour la Synthèse d'Image

Codez une surcouche d'OpenGL pour faciliter le développement. Elle pourrait contenir des classes simplifiant la gestion des ressources (VBO, textures, etc.) ou encapsulant des techniques utilisées régulièrement.

Pour allez plus loin, vous pouvez également :

- Utiliser des shaders plus complexes prenant des textures de normales voire d'autres paramètres du modèle d'illumination (specular notamment) (normal-mapping)
- Gérez des particules (OpenGL Tutorial - Particules)
- (Projection SI IMAC 3) Utilisez des FBO pour optimiser le rendu (OpenGL - Frame Buffer Object)
- (Projection SI IMAC 3) Utilisez des FBO pour gérer les ombres (OpenGL Tutorial - Light/Shadow Mapping)

2.2 Programmation C++

Ce projet est à réaliser en majeur partie en C++. Il est largement conseillé d'utiliser une version moderne du C++ (11, 14 ou 17).

2.2.1 Compilation

Vous devrez fournir les fichiers permettant de gérer la compilation avec **CMake** (au moins la version 3.13). Vous devrez également spécifier les options `-W`, `-Wall` et `-Werror` pour compiler votre projet.

2.2.2 Documentation

Nous attendons un code lisible, bien organisé et auto-documenté. Bien organisé signifie que vous répartirez judicieusement votre code sur plusieurs fichiers, et que les fonctions ou classes composants chaque fichier seront elles aussi bien réparties. C'est le moment d'utiliser les bonnes pratiques et des aides fournies par les Design Pattern. Gestion des responsabilités, nommage des Design Pattern utilisés/reconnus. La production d'un schéma UML (Diag de Classe, Diag de séquence système) serait une bonne façon de montrer ce qui est à l'oeuvre.

Par ailleurs, il est nécessaire de commenter votre code en gardant à l'esprit qu'il doit être compréhensible même sans ces commentaires : essayez de nommer vos variables et vos fonctions explicitement (utilisez '`objectCount`' au lieu de '`n`' par exemple) et n'hésitez pas découper de longs algorithmes en plus petites fonctions.

2.2.3 Côté technique

Voici ce qu'il serait bon de rencontrer dans votre projet :

- classes
- classes et fonctions template
- héritage
- polymorphisme
- usage massif d'outils de la STL
- exceptions pour traiter les erreurs systèmes (version d'opengl incompatibles, plus de ram, etc.)
- messages d'erreurs pour traiter les erreurs utilisateurs (fichiers inexistants, entrées invalides, etc.)
- asserts pour détecter et prévenir les erreurs de programmation (division par zéro, valeur nulle non attendue, etc.)
- espaces de nommage
- fonctions lambdas
- (bonus) des fonctions et des variables `constexpr`
- (bonus) fonctions variadics

Dans votre rapport, **vous fournirez un tableau** décrivant pour chacun de ces points (et d'autres le cas échéant) si vous les utilisez, et si oui, dans quels fichiers.

Voici ce qu'il serait regrettable de rencontrer dans votre projet :

- une mauvaise organisation de vos fichiers et de vos classes
- des variables, des fonctions ou des types mal nommés
- des références non constantes alors qu'elles devraient l'être
- des passages par copies non justifiés
- des erreurs de segmentation
- des fuites de mémoires

— des bugs
— du code de debug non retiré
— du code mort (= des portions de code non utilisés par le programme)
— du code dupliqué
— du code illisible d'un point de vue sémantique (des `for` dans des `for` dans des `if` dans des `for` dans des `else` par exemple)
— du code illisible d'un point de vue esthétique (indentation irrégulière, mélange camelCase / snake_case, mélange tabs / spaces)
— des mega fonctions de plus de 30 lignes
— des mega fichiers de plus de 500 lignes
— des variables globales non `constexpr`
— des crashs lorsque l'utilisateur effectue une opération non prévue

3 Notation

Le projet permettra une double évaluation en Synthèse d'images et en C++, et une vérification pour l'Architecture Logicielle. Vous obtiendrez donc deux notes à partir de votre travail, qui sera évalué d'une part sur la base d'une partie commune, et d'autre part spécifiquement aux deux matières. Une note d'Architecture Logicielle (coefficient 0.5) sera utilisée avec l'examen qui aura lieu sur papier.

Partie Commune (8)

Rapport ⋆

Soutenance ⋆⋆

Compilable par l'évaluateur ⋆⋆

Architecture logiciel et propreté du code ⋆⋆

Fonctionnalités du jeu ⋆ ⋆ ⋆

Originalité ⋆

Partie Synthèse d'Image (12)

Qualité du rendu graphique - modèles présents et agencés dans la scène 3D ⋆⋆

Qualité et consistance du modèle d'illumination ⋆ ⋆ ⋆

Experience Utilisateur et interaction avec la scène ⋆ ⋆ ⋆

Partie Programmation C++ (12)

Compilation et organisation des fichiers

Organisation du code, lisibilité

Usage adpté des outils C++ (template / héritage / polymorphisme / etc.)

Usage de la STL

Gestion des erreurs

Efficacité à runtime (si optimisation renseignée dans le rapport)

Bonus (2) (Soutenance++, Fonctionnalités ++, Corruption, ...)

4 Remarques

— Il est très important que vous réfléchissiez avant de commencer à coder aux principaux modules, algorithmes et aux principales structures de données que vous utiliserez pour votre application . Il faut également que vous vous répartissiez le travail et que vous déterminiez les tâches à réaliser en priorité.

— Vous pouvez utiliser un outil de répartition et suivi de tâches, par exemple Trello, Azendoo ou Notion

— Ne rédigez pas le rapport à la dernière minute sinon il sera bâclé et cela se sent

toujours.

— Le projet est à faire par binôme/trinome. Il est impératif que chacun d'entre vous travaille sur une partie et non pas tous "en même temps" (plusieurs qui regarde un travailler). sinon vous n'aurez pas le temps de tout faire. Néanmoins vous pouvez commencer ensemble pour définir la structure commune.

— N'oubliez pas de tester votre application à chaque spécification implémentée. Il est impensable de tout coder puis de tout vérifier après. Ne visez pas directement votre oeuvre finale, tester les implémentations basiques sur une scène simple.

— Vos chargés de TD et CM sont là pour vous aider. Si vous avez des doutes/des difficultés sur un point, n'attendez pas la soutenance pour nous en parler.