Initiation API Graphique Moderne

Procédure du rendu

Début	Fin
Mardi 31/01/2023 – 9h	Lundi 17/02/2023 – 16h

Nombre d'étudiants par groupe : 1

Rendu :

- Sources : tag GOLD sur Git (sur la branche master)
- Sur Git (uniquement dans la version GOLD)
    - Exécutable
    - Libs pour qu'on puisse compiler votre projet
    - Screenshots: Au nombre de 5, dans un dossier Screens

4 points seront retirés si les consignes ne sont pas appliquées

Pénalités de retard:

- 5 minutes de retard = -1 points
- 15 minutes de retard = -2 points
- 30 minutes de retard = -4 points
- 1 heure de retard = -10 points
- Plus de retard = 0

Sujet

Le but du projet consiste à appréhender le fonctionnement des API graphiques modernes telles que Vulkan ou DirectX 12.

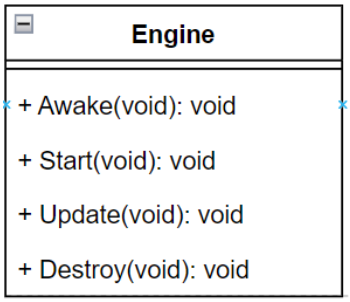
À la fin, vous serez capable de charger un modèle 3D sur la carte graphique, de l'afficher à l'écran, et de lui appliquer des effets visuels en utilisant une API graphique moderne.

Vous avez le choix d'utiliser **Vulkan ou DirectX 12** pour votre rendu.

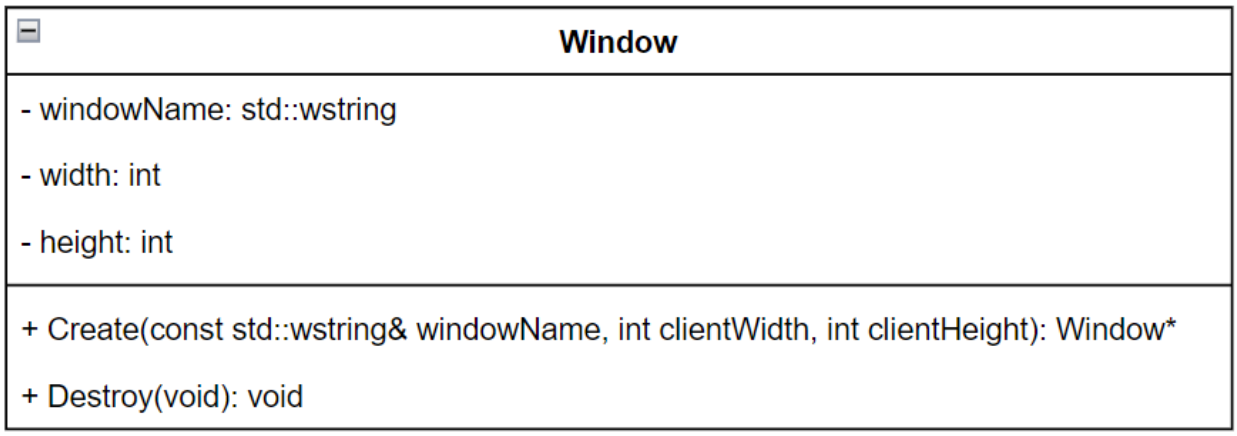
Voici une liste des étapes à suivre pour compléter le projet :

- Avoir une fenêtre avec une couleur d'arrière plan qui se ferme avec ESCAPE ou la croix en haut à droite de la fenêtre
- Dessiner un triangle
- Dessiner un cube avec de la perspective
- Appliquer un mouvement à ce cube toutes les x secondes
- Ajouter la possibilité de se déplacer dans la scène avec ZQSD
- Ajouter une texture sur le cube
- Ajouter des lumières
- Afficher plusieurs objets dans la scène 3D
- Bonus : coder un mini-jeu de type platformer (uniquement après fait un point avec votre responsable pédagogique pour voir ensemble ce qui pourrait être amélioré)

Voici les classes de base que vous allez devoir utiliser pour ce projet :

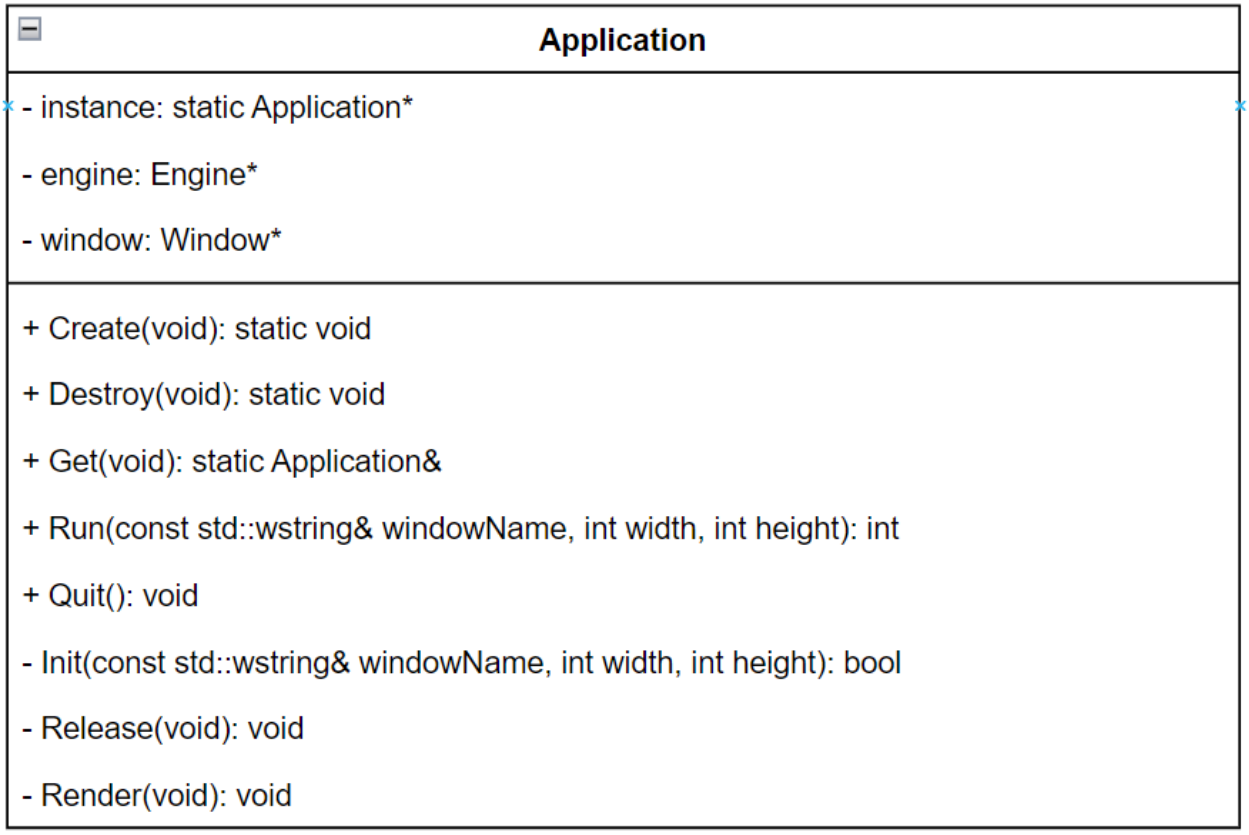


Engine : cette classe charge la scène 3D et la met à jour à chaque frame.



Window : cette classe initialise la fenêtre du jeu.
(elle utilise les fonctions de Windows/GLFW pour initialiser la fenêtre)

- **Create** renvoie un pointeur sur la fenêtre initialisée
- **Destroy** libère les ressources de la fenêtre



Application : cette classe met en place la boucle principale du jeu.

- **Create** initialise l'instance de l'application
- **Destroy** détruit l'instance de l'application
- **Get** permet de récupérer une référence sur l'instance de l'application
- **Run** lance la boucle principale du jeu
  - Appelle **Init** (quitte directement si l'initialisation s'est mal passée)
  - Boucle principale tant qu'on ne ferme pas l'application
    - Mets à jour les inputs
    - Appelle **Update** de **Engine**
    - Appelle la fonction d'affichage, **Render**
- **Quit** arrête l'application

- **Init** (appelée au début de **Run**) initialise l'application
    - Crée la fenêtre
    - Initialise l'API graphique
    - Retourne si l'initialisation s'est bien passée

- **Release** libère les ressources du jeu
    - Appelle **Destroy** de **Engine**
    - Libère les ressources de l'API graphique
    - Appelle **Destroy** de **Window**

- **Render** met à jour l'affichage de la fenêtre

Ces classes de bases sont obligatoires, mais vous pouvez en ajouter si vous le voulez.
Les API graphiques modernes ayant tendance à être très verbeuses, je vous invite à prendre le temps de bien architecturer votre code pour qu'il reste le plus lisible possible.

Compétences Évaluées

PROGRAMMATION GRAPHIQUE

Initiation à Vulkan : 100%

Liens utiles

Tuto Vulkan :
https://vulkan-tutorial.com/Introduction
Tuto DirectX 12 :
https://www.3dgep.com/learning-directx-12-1/