GMINT317 - Moteurs de jeux – TP4
Gestionnaire de ressources

Rémi Ronfard  *remi.ronfard@inria.fr*  *https://team.inria.fr/imagine/team/*

Objectifs

Le but de ce TP est de mettre en place un gestionnaire de ressources pour le chargement d'objets et de scènes complexes par le moteur de jeu.

Cela inclut la gestion uniforme d'un système de coordonnées pour votre scène 3D.

Bonus :

- Charger plusieurs autres types de modèles 3D
- Améliorer le gestionnaire de ressources

Gestionnaire de version

Au plus tard la semaine prochaine (minuit), vous devrez rendre un compte rendu de ce TP sur votre espace GIT, ainsi que votre code source.

Mettre en place un gestionnaire de ressources

Dans le TP3, nous avons chargé une ***height map*** (carte de profondeur). Pour ce faire, nous avons chargé une image.

Ici, nous allons mettre en place un gestionnaire de ressources qui servira à plusieurs choses :

- Sauvegarder la saison de nos différentes fenêtres (chaine de caractères),
- Permettre le chargement dynamique de terrain (Composé d'un nombre de sommets caractérisés par : X, Y, Z et une couleur) au lieu du chargement d'une image.
- Charger des objets 3D sur le terrain (Avec comme information : une liste de sommets, un nom d'objet et leurs coordonnées)

Question 1

Créer une classe FileManager chargée de lire et écrire les données de votre jeu dans un unique fichier binaire (« big file »).

Réalisez les différentes méthodes qui permettent de stocker les données de votre jeu :

- Terrain de chaque fenetre
- Saison de chaque fenêtre
- Etat de la caméra

En particulier, vous devez sauvegarder votre terrain sous la forme d'un tableau de sommets (X,Y,Z, Couleur) dans un unique fichier binaire (« big file »).

```
void TriangleWindow::saveCustomMap(QString localPath)
```

Réalisez les différentes méthodes qui permettent de lire les données stockées et de ré-initialiser la scène à partir d'un unique fichier (« big file »).

En particulier, vous devrez lire votre terrain et afficher le terrain à l'aide d'une fonction

```
void TriangleWindow::loadCustomMap(QString localPath)
```

similaire à la fonction loadMap des précédents TP.

Pensez à faire un commit et un push sur git pour chaque élément du gestionnaire de ressources.

Question 2

Réaliser un chargeur de données pour lire des modèles 3D d'arbres donnés avec le TP et les placer dans la scène. Attention, il faut pour cela prévoir une position 3D, une orientation 3D et une taille.

Attention : pour cette partie, vous allez devoir mettre en place un système uniforme de coordonnées pour le placement des objets.

Modifier les fonctions de votre gestionnaire de resource pour y inclure le nombre et le type d'arbres dans chaque fenêtre, leurs positions, leurs orientations et leur tailles.

NB – Le chargement des modèles 3D peut rester séparé du « big file ».

Compte rendu

Présentez toutes vos nouvelles fonctionnalités
Expliquez les points que vous n'êtes pas arrivé à réaliser et pourquoi.
Expliquez votre démarche de développement.
Présentez votre structure de données.
Expliquez comment vous vous y prendriez pour les parties bonus.

Bonus

Ajouter d'autres objets 3D à votre scène et mettre à jour le gestionnaire de resources.

Vous pouvez trouver de nombreux modèles gratuites sur le site BLEND SWAP

http://www.blendswap.com/

Ces modèles peuvent être importés dans Blender 3D puis exportés au format PLY.

Ajouter d'autres formats 3D à votre gestionnaire de ressources, par exemple : STL, OBJ, DAE, etc.

La spécification des principaux formats d'objets 3D est disponible à la même adresse :

- http://paulbourke.net/dataformats/