COMPTE-RENDU TP2

HMIN317 – Moteur de jeux

Résumé

Ajout d'un timer gérant le rafraîchissement d'une fenêtre OpenGL.

David Lonni

Master 2 IMAGINA

1) Questions

<u>Question n°1</u> :

Interactions avec le programme.

Z – S : Avancer/Reculer

Q – D : Tourner le terrain à gauche/à droite

A – E : Incliner vers le haut/bas

W : Changer la représentation du terrain (points, triangles, triangles avec couleur alternée, lignes, triangles colorés en fonction de l'altitude, lignes et triangles colorés en fonction de l'altitude)

X : Changer de heightmap pour la génération du terrain

Le paramètre *etat* définit les différents types d'affichage du terrain (cités ci-dessus). Les variables *ss*, *rotX* et *rotY* définissent respectivement l'échelle du terrain en xyz (scale) ainsi que la rotation en X et Y de celui-ci.

<u>Question n°5</u> :

Que se passe-t-il lorsque la fréquence de rafraîchissement est trop basse/haute ?

Lorsque la fréquence de rafraîchissement est trop basse, on a une faible fréquence de mise à jour de l'affichage de la fenêtre, ce qui a pour effet de faire « saccader » le terrain lors de la rotation. Au contraire, plus celle-ci est élevée, plus la rotation et les actions effectuées sont mises à jour régulièrement à l'écran et l'affichage devient fluide.

2) Démarche de développement

Afin de rafraîchir l'affichage de la fenêtre uniquement en fonction du taux de rafraîchissement passé en paramètre (et donc grâce au **QTimer)**, j'ai retiré l'appel de *renderNow()* à la fin de la fonction *keyPressEvent()* qui permettait la mise à jour de l'affichage à chaque interaction avec le clavier. Ainsi le timer est le seul à pouvoir demander une mise à jour de l'affichage.

Pour que les 4 fenêtres du programme partagent la même caméra, j'ai créé une instance de la classe **Camera** dans le *main*. J'ai pu alors faire passer le pointeur de cette instance à tous les constructeurs des **GameWindow** afin qu'ils pointent vers la même caméra et ainsi avoir le même point de vue sur le terrain en fonction des entrées clavier de l'utilisateur.

Et enfin, pour pouvoir animer tous les terrains des différentes fenêtres et pouvoir les arrêter en même temps en appuyant sur la touche « C », j'ai créé une variable « *static bool animate* » qui est commune à toutes les instances de ma classe **GameWindow**. Contrairement au taux de rafraîchissement qui est une variable « *int m_refresh_rate* » de la classe **GameWindow** et qui peut donc être défini séparément d'une fenêtre à l'autre.

3) Bonus

Pourquoi la classe GameWindow hérite-t-elle de OpenGLWindow ?

La classe **GameWindow** hérite de la classe **OpenGLWindow** afin de pouvoir posséder les méthodes et attributs permettant la création d'une fenêtre OpenGL et ainsi gérer l'affichage d'un environnement 3D.

Quels sont les avantages et inconvénients de cet héritage ?

Avec cet héritage, on peut directement accéder aux méthodes permettant de gérer l'affichage et le rendu de la fenêtre. Cependant, la classe **OpenGLWindow** contient des méthodes que nous devons redéfinir afin d'obtenir le rendu que nous souhaitons ainsi qu'initialiser les paramètres dont nous avons besoin.

Comment serait-il possible d'éviter cet héritage ?

Il serait possible d'éviter cet héritage en ajoutant un attribut dans la classe **GameWindow.** Un pointeur sur une **OpenGLWindow** nous permettrait d'accéder aux méthodes et attributs de cette classe sans pour autant en hériter. Cependant, il nous serait alors impossible de redéfinir les méthodes de rendu et d'initialisation de la classe **OpenGLWindow**.