COMPTE-RENDU

HMIN317 – Moteur de Jeux

TP7 – 28 Novembre 2015

Contenu

– QuadTree

Sébastien Beugnon
M2 IMAGINA 2015-2016

1. Fonctionnalités

- Ajout d'un nouvel état à la Caméra permettant de voir le résultat du QuadTree.

- Ajout d'interactions (touche P) pour changer le mode du QuadTree (MinMax, Mean, StdDeviation)

- Ajout d'interactions (touche + / -) pour changer le seuil de tolérance de subdivision du QuadTree.

2. Démarche de développement

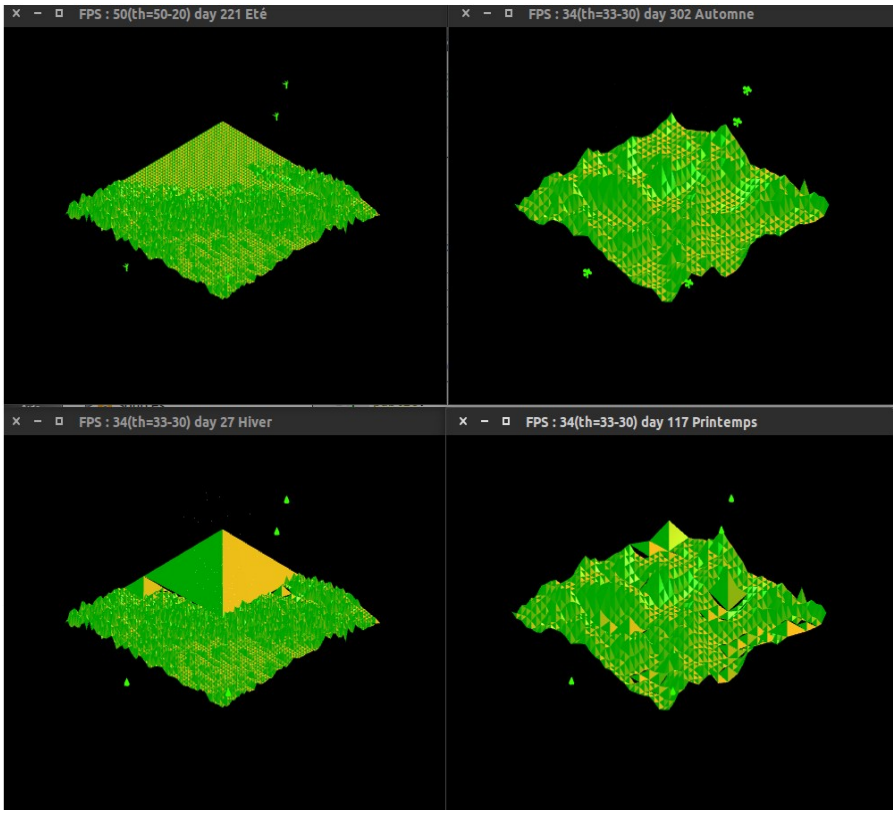
J'ai mis en place une classe *QuadTree* prenant en paramètre un pointeur de classe Terrain de mes précédents travaux pratiques. Ce *QuadTree* possède trois modes d'utilisation, lorsqu'on demande de modifier le QuadTree on utilise une tolérance, si le résultat de la résolution choisie (à l'aide du mode) est supérieur à la tolérance alors on subdivise le *QuadTree*. Trois modes ont été mis en place pour cette structure :

- MinMax, il réalise la différence entre le point le plus haut et le plus bas parmi les quatre points d'extrémité du carré.

- Mean, il calcule la moyenne de tout les points pour le carré courant.

- StdDeviation ou Standard Deviation, la structure réalise l'écart type avec tout les points à l'intérieur du carré courant.

Pour stocker et mettre à jour rapidement la structure, le *QuadSet* ne stocke uniquement que les coordonnées x_min, x_max, y_min, y_max du carré afin de n'avoir uniquement que les identifiants des quatre points déterminant un carré. Lors d'une mise-à-jour, si le carré courant n'a pas besoin d'être subdivisé et qu'il avait précédemment des *QuadSet* qui sont sa subdivision il les supprimer pour libérer de la mémoire.



Bonus

Pour tout ce qui est gestionnaire de scènes, j'ai implémenté ceci directement dans le projet de moteur de jeu que vous pouvez trouver sur le dépôt suivant :

https://github.com/TsubameDono/BreakingEngine

Le gestionnaire de scènes (ou hiérarchie d'objets) est basé sur les propriétés des *QObject* de Qt permettant de définir des objets enfants et un parent par objet.

Dans ce moteur, j'ai employé le patron de conception *Entity/Component* : la classe *GameObject* correspond à une entité de ma scène à laquelle je peux rattacher des composants comme le composant *Transform* qui garde en mémoire les coordonnées de l'objet, ses rotations et ses échelles en x, y et z.

De cette manière, il est plus de rattacher à chaque objet de la scène des informations supplémentaires qui lui sont propres (*Camera*, *Terrain*), des scripts (*Updatable*) ou des objets à dessiner. (*Renderer*)