COMPTE-RENDU TP7

HMIN317 – Moteur de jeux

Résumé

Ajout d'un Quadtree sur le Terrain et d'un LOD pour les arbres.

David Lonni

Master 2 IMAGINA

1) Fonctionnalités

Terrain :

- Se déplacer avec ZQSD et Up, Down pour monter ou descendre selon l'axe Y,
- Lorsque le clic gauche est enfoncé, cela permet de manipuler la caméra comme dans un jeu de type FPS,
- La touche « W » permet d'afficher le terrain en mode wireframe,
- La touche « X » permet de changer la carte du terrain,
- Les touches « 4 » et « 6 » du pavé numérique permettent de faire varier la direction de la lumière directionnelle selon l'axe X,
- Les touches « 8 » et « 5 » du pavé numérique permettent de faire varier la direction de la lumière directionnelle selon l'axe Z,
- Ces 2 dernières fonctionnalités font permettre de changer le rendu du « toon shading » appliqué sur le terrain,
- La touche « V » permet de passer du Terrain décimé selon le Quadtree au Terrain non décimé et vice-versa,
- La touche « 1 » permet d'afficher/cacher le Terrain,
- La touche « 2 » permet d'afficher/cacher le Quadtree,
- La touche « 3 » permet d'afficher/cacher le résultat de la décimation,
- La flèche directionnelle droite permet d'augmenter le paramètre delta de mon Quadtree,
- La flèche directionnelle gauche permet de diminuer le paramètre delta de mon Quadtree,
- La touche « + » du pavé numérique permet d'augmenter la résolution minimum que peuvent avoir les voxels du Quadtree,
- La touche « - » du pavé numérique permet de diminuer la résolution minimum que peuvent avoir les voxels du Quadtree.

2) Démarche de développement

Quadtree static

J'ai commencé par développer un Quadtree Static qui permet d'afficher une grille sur le Terrain de manière fixe.

Pour cela j'ai développé une classe Voxel qui me permet d'afficher des cubes en fil de fer à des positions et tailles différentes sur le Terrain.

J'ai ensuite développé une classe Quadtree qui permet de définir le Quadtree à afficher sur le Terrain. Celui-ci va être créé grâce à la méthode récursive :

```
void Quadtree::QuadtreeStatic(Voxel* v, int offsetX, int offsetY)
```

Cette méthode va créer le Quadtree à partir d'un voxel parent qui englobe tout le Terrain et part des offset X et Y en 0.

Elle va subdiviser les voxels jusqu'à parvenir à une résolution minimum qui correspond à la taille minimale que les voxels peuvent avoir où bien après avoir atteint la valeur de delta voulu dans le voxel courant.

Pour calculer ce delta, on va parcourir les vertex qui sont compris dans le voxel courant puis calculer l'écart-type de la hauteur de tous ces vertex.

Si l'écart-type de la position en Y des points est plus petit que le delta voulu, on arrête de subdiviser ce voxel et on continue à former le Quadtree.

À la suite de ces opérations, on peut afficher le Quadtree sur le Terrain sous la forme d'une grille que l'on peut mettre à jour à chaque changement du Terrain.

Remeshing

Suite à cela, j'ai mis à jour mon tableau qui contient les indices des vertex du Terrain afin d'afficher le Terrain qui correspond à la subdivision imposée par le Quadtree. Je crée alors des triangles en donnant à mon tableau d'indices les identifiants de mes vertex correspondant aux quatre coins de mes voxels formant mon Quadtree.

Quadtree dynamique

Afin de rendre dynamique mon Quadtree, j'ai lié le paramètre delta de mon Quadtree avec la distance de la caméra par rapport à mon Terrain. Plus ma caméra s'éloigne, et plus mon paramètre delta augmente et inversement. Je mets donc à jour mon Quadtree à chaque déplacement de ma caméra et donc également la topologie de mon Terrain en fonction de celui-ci.

LOD Tree

J'ai fini par ajouter un arbre sur mon Terrain que j'ai défini avec 3 LOD différents. J'affiche donc le bon niveau de détail de cet arbre en fonction de la distance avec la caméra.