Compte rendu : TP1 – MOTEUR DE JEUX

Vincent de Rousiers

Fonctionnalités :

Ayant passé plus de temps à installer Qt sur ma machine qu'à travailler mon TP, je n'ai pas donné à l'utilisateur le choix de lancer un exercice en particulier.
Ainsi, si on veut lancer l'exo1, il faut le décommenter (« **exo1()** ») dans le main et commenter « **window.heightMap();** »
Si l'on veut « La Montagne », il faut commenter « **exo1()** » et décommenter « **window.heightMap();** »

« **exo1()** » affichera un maillage de points dont les coordonnées Z sont randomisées et dont le nombre de points est paramétrable via une variable globale (« dimension » qui représente le nombre de point constituant un des côtés du maillage).
« **window.heightMap();** » quant à elle affiche la carte d'altitude correspondant à l'image fournie. L'altitude est représentée par un dégradé.

Concernant la caméra, le résultat est une adaptation aux besoins du TP d'un tutoriel et des TPs de l'année dernière.
Le mouvement se fait à l'aide des touches Z,Q,S,D et la vue est gérée par la souris.
Le plus gros problème réside dans la gestion de la souris, je n'ai pas encore réussi à produire un déplacement fluide et précis avec.

La lecture de la carte d'altitude fonctionne quant à elle parfaitement, l'altitude étant représenté par un dégradé allant du noir (profondeur) au blanc (sommet).

Démarche de développement :

Au commencement, un maillage de 16 points fut créé sans utiliser de classe.
Cela n'était pas bon.
Deux classes apparurent donc, **Face** et **Point**, ce qui permit de manipuler le maillage plus aisément.

Le maillage étant terminé, les efforts se portèrent sur la caméra.
Celle-ci est le fruit de mes précédents TPs de M1 sur OpenGl, quelque peu modifiés pour coller à ce TP.
Pour la vue, on récupère la différence entre deux mouvements de souris et on fait donc tourner la caméra dans le sens du mouvement.
Les mouvements sont gérés par les quatre touches habituelles à tout joueur de FPS (Z,Q,S,D) grâce à un listener dans la classe **Camera**.

La génération de la map consiste en la lecture de l'image, on crée un tableau de **Point** en récupérant les coordonnés (z étant calculé en fonction de l'intensité de gris du pixel).
On fabrique ensuite un tableau de **Face** qui va contenir les faces du maillage.
Pour calculer ces faces, on parcourt le tableau de points 4 par 4 (un carré, deux faces).
Par exemple, le premier carré sera composé des points 0,1, x, x+1 (avec x la dimension du maillage), le second sera 1,2,x+1,x+2 etc.
Chaque carré conduit à la création de deux faces, 0,1,x et 1,x+1,x pour le premier carré par

exemple.
Ex :

0	1	2	3	…	x-1
x	x+1	x+2	x+3	…	2x – 1
…	….	…	…	…	...

Structure de données:

Pour représenter les maillages, j'ai crée deux classes :

-**Point** : qui prend un tableau de 3 Glfloat pour représenter les coordonnées

-**Face** : qui prend un tableau de 3 **Points** (ou plus, ou moins, le nombre est paramétrique) pour représenter les points la composant

C'est deux structures allègent énormément le code et permettent une gestion plus aisée des problèmes. Par exemple, pour modifier la couleur en fonction de l'altitude, j'ai crée une fonction moyCoord(int) dans **Face** qui renvoie la moyenne des coordonnées en int (0 pour x, 1 pour y, 2 pour z) des points composant la face. On peut ainsi très rapidement connaître l'altitude d'une face par rapport à une autre.

Mes maillages sont stockés dans un tableau de **Face**, dont je calcule la taille en fonction du nombre de points initiaux. Je peux ensuite très facilement dessiner les faces en passant par le tableau.

La caméra possède sa propre classe qui gère les événements claviers et souris.

Bonus :

- Pour la collision, je pensais réutiliser l'altitude moyenne de la face la plus proche à la caméra pour empêcher celle-ci de la dépasser.
- Comme je l'ai déjà fait, je prend l'altitude maximum et minimum des faces puis je colorie chacune d'entre elles en fonction de son rapport avec ces maximum et minimum
- Je pense créer une « boule » émettrice de lumière (une « glLightfv ») que je ferais bouger à l'aide de touches (simuler un soleil). C'est une demande déjà exprimé dans l'UE de Modélisation 3D en M1.
- Pour localiser le brouillard dans les vallées, il doit être possible de prendre un pourcentage de face ayant une altitude inférieur à la moitié ou aux 3/4 des autres faces et leur associer le brouillard. Ainsi le brouillard ne sera présent que dans les creux du maillage.
- Pour créer un rendu infini, on pourrait n'afficher que les points du maillage présent dans le champ de vision de la caméra et ne calculer les points que dans un rayon restreint autour de la caméra.