Benlamine
Mehdi
Lien GIT : https://github.com/Mehdi-Ben/MoteurDeJeu

Compte Rendu TP1 :

Question 1 :

MainWidget :

la classe contient les fonctions nécessaires à l'initialisation de l'interface, de la fenêtre openGL, des shaders et des textures. Elle détient aussi les fonctions de gestion des événements de la fenêtre, la fonction de gestion du rendu « paintGl » et la fonction « resizeGL » permet de redéfinir la taille de l'objet à afficher dans la fenêtre.

GeometryEngine :

la classe contient un tableau dans lequel sont définis tous les vertex nécessaire à l'affichage du cube ainsi qu'un tableau d'indices correspondant aux indices des vertex de chaque face rangés pour les dessinaient en triangle strip.
Ces informations sont stochées dans les buffers « «arrayBuff » et « indexBuff » pour être utiliser avec openGl dans la fonction « «drawCubeGeometry » pour dessiner le cube grâce à la fonction « glDrawElements ».

vshader.glsl :

Vertex Shader.
Récupère les coordonnées de l'objet sur la scène sur lesquelles appliquer la texture.

fshader.glsl :

Fragment Shader.
Défini la couleur à appliquer à chacune des coordonnées en fonction de la texture.

Question 2 :

void GeometryEngine::initCubeGeometry() :

Cette méthode contient les tableaux de vertex et d'indices nécessaires pour l'affichage du cube. Ces informations sont tranférées dans les buffers « «arrayBuff » et « indexBuff » pour être utiliser par openGl.

void GeometryEngine::drawCubeGeometry(QOpenGLShaderProgram *program) :

Cette méthode lie les tableaux de vertex et d'indices définis dans « initCubeGeometry() » au programme shader courant avec « .bind() ».
Elle spécifie où lire les données de position des vertex et celles de coordonnées des textures.
Elle dessine le cube grâce à la fonction « glDrawElements » dans laquelle on spécifie le type de dessin (GL_TRIANGLE_STRIP).

Compte Rendu TP2 :

Question 3 :

Le QbasicTimer utilisé permet la mise à jour à intervalles réguliers du terrain, cet intervalle est définie lors de l'appel à la méthode start . En écrivant ceci : timer.start(1000/fps,this); il possible de mettre le nombre de FPS voulu en argument.

Plus le taux de rafraîchissement de la fenêtre est bas, plus la rotation est lente.
La vitesse augmente donc plus rapidement dans les fenêtre au taux de rafraîchissement élevé.

Problèmes rencontrés :

J’ai eu quelques soucis lors de la création de la surface plane, je voulais tous d’abord la faire avec des triangles strips mais je n’ai pas réussi à m’en sortir avec les indices, donc je l’ai faite avec des triangles simples.
Je n’ai pour le moment pas essayer de mettre en place la lumière.
Par contre j’ai essayé de mettre en place une caméra freefly gérée par les flèches du clavier et la souris. Pour ce qui est des mouvement de translation (avancer,reculer,gauche,droite,haut,bas), je n’ai pas eu de problèmes mais j’en ai toujours un concernant la rotation à l’aide de la souris que j’espère régler bientôt.

Bonus

La coloration du terrain en fonction de l’altitude de ses sommets est en place.

Pour la lumière, qui n’est pas implémentée actuellement, il faut utiliser les shader en fonction du type de lumière voulue : diffuse pour une lumière de type soleil, ponctuelle pour une lumière localisée ou spéculaire pour les matériaux.