Prise en main de Qt Creator, Git, et OpenGL ES 2.0

Question 1 :

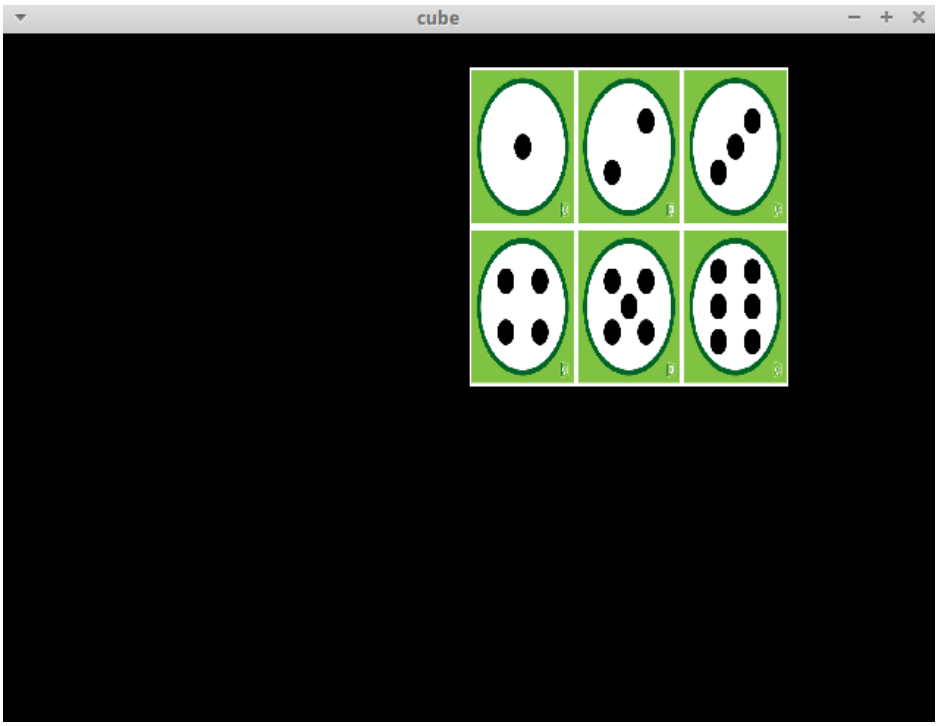
La classe MainWidget permet de configurer la fenêtre principale, de gérer les évènements souris, et d'initialiser les propriétés d'OpenGL (le repère, les textures et les shaders)
La classe GeometryEngine permet de dessiner des formes géométriques dans le programme passé en paramètre.

Le fichier vshader.glsl calcule la position dans l'écran et transfert les coordonnées au fshader.glsl qui va afficher la texture.

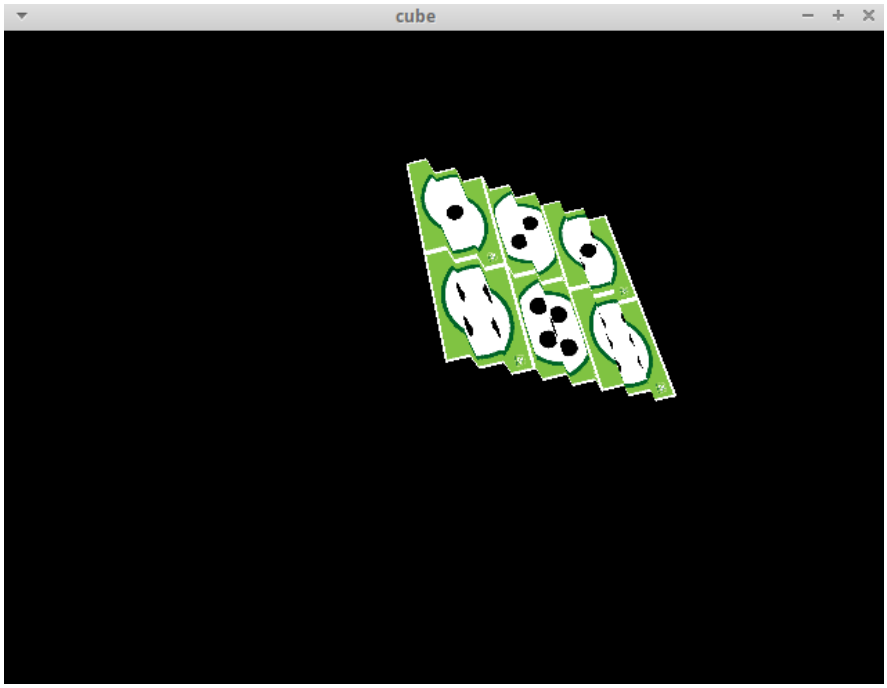
Question 2 :

initCubeGeometry() : initialise le tableau de points du cube ainsi que le tableau d'indices des faces puis les lie au VBO.
drawCubeGeomtry() : lit les deux VBO et les relie aux pipelines d'OpenGL, puis appelle glDrawElements() qui va dessiner les points et les faces.

Question 3 :



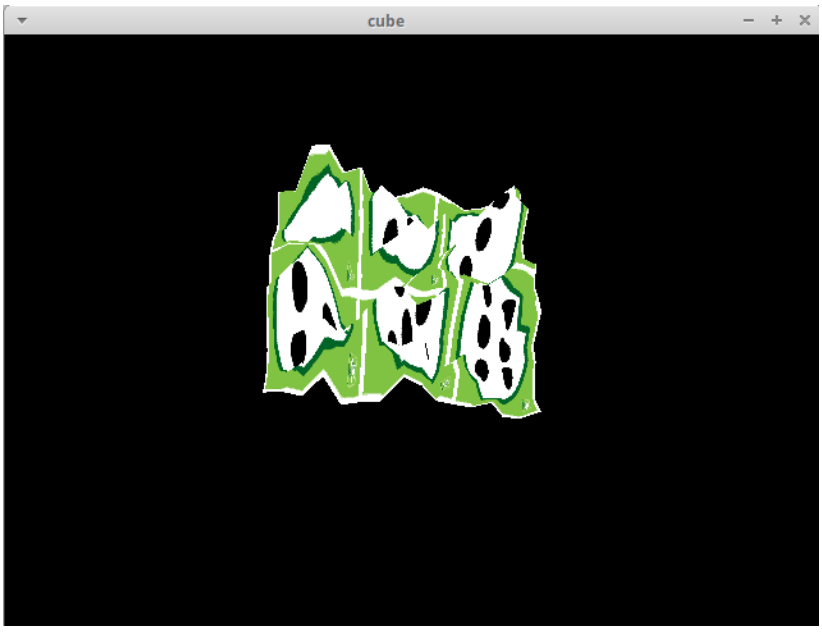
Question 4 :

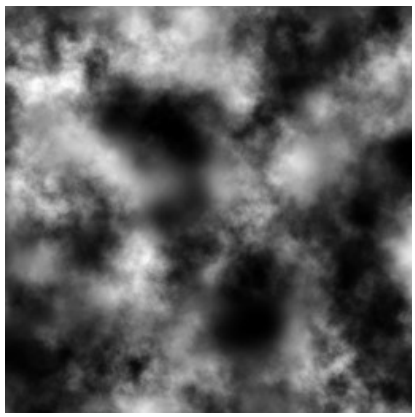


Une ligne sur deux a un z=0.1, l'autre z=0

TP2

Question 1 :Le niveau de gris de la heightmap donne l'altitude du terrain. Plus le point est blanc, plus il sera haut.





Question 3 : Le terrain se met à jour dans la méthode « timerEvent » qui est appelée lors d'un évènement de timer. La classe QTimer gère la vitesse de mise à jour de l'affichage dans cet exemple. En effet, en indiquant une valeur à la méthode start, on peut influer sur la vitesse d'appel de timerEvent.

Problèmes et difficultés rencontrées :

J'ai pris beaucoup de temps à faire un algorithme qui me convenait pour le découpage de la surface en n sommets. J'ai encore des soucis de compréhension avec les méthodes openGL, notamment pour la gestion de la caméra.

Idées de résolution des bonus :

Pour la lumière, openGL permet de faire des sources lumineuses. Il aurait fallu indiquer la réflexion des lumières sur la texture.
Pour les couleurs, plutôt que d'attribuer une nuance de gris en fonction de la heightmap, j'aurais pu choisir ces valeurs dans l'espace RGB.