Compte rendu TP 2

Question 1 :

Pour ouvrir et lire une image d'altitude, j'utilise un Qimage avec le lien de l'image à utiliser :

- Dans geometryengine.h, j'ai ajouté (private) :

```
QImage *heightmap;
```

- Dans geometryengine.cpp, dans le constructeur :

```
// Highmap
heightmap = new QImage(":/heightmap-1.png");
initPlaneGeometryHeightmap(heightmap);
```

J'ai ensuite créé une méthode `void GeometryEngine::initPlaneGeometryHeightmap(QImage *heightmap)` et qui va, pour chaque point du plan, indiquer une position sur l'axe Z par rapport à l'intensité des pixels de l'image précédemment chargée :

```
// Create array of 16 x 16 vertices facing the camera  (z=cte)
VertexData vertices[16*16];
int mapJumpX = heightmap->width()  / 16;
int mapJumpY = heightmap->height() / 16;

for (int i=0;i<16;i++)
    for (int j=0;j<16;j++)
    {
        // Vertex data for face 0
        double z = QColor::fromRgb(heightmap->pixel(i * mapJumpX, j *
mapJumpY)).red() / 512.0 + 2.0;

        vertices[16*i+j] = { QVector3D(0.1*(i-8),0.1*(j-8), z),
QVector2D(0.33*i/16.0,0.5*j/16.0)};
        // add height field eg (i-8)*(j-8)/256.0
    }
```

Il suffit d'appeler maintenant cette méthode pour avoir un plan déformé par la carte d'altitude :

Question 2 :

Pour avoir une vue d'angle de 45 degrés, on utilise les méthodes suivantes :

```
QQuaternion framing = QQuaternion::fromAxisAndAngle(QVector3D(1,0,0),-45.0);
matrix.rotate(framing);
```

Pour faire tourner notre plan autour de son origine, on peut utiliser les même fonctions que lors du clic et déplacement de la souris, mais au lieu de l'appliqué avec un événement, on l'applique à chaque update :

- Dans le constructeur de MainWidget, on indique une vitesse de rotation :

  ```
  angularSpeed(0.5)
  ```

- On change la méthode timerEvent comme suit :

```
void MainWidget::timerEvent(QTimerEvent *)
{
    // Update rotation
    rotation = QQuaternion::fromAxisAndAngle(rotationAxis, angularSpeed) *
rotation;
    // Request an update
    update();
}
```

Question 3 :

La mise à jour du terrain est contrôler par un évent de type « Timer ». Cela signifie que à chaque fois que le timer atteint 0, un signal est envoyé au widget MainWidget, qui fait appelle à la méthode (slot) eventTimer.

```
MainWidget::MainWidget(QWidget *parent, int fps)
```

Pour afficher les différentes fenêtres :

```
MainWidget widget(nullptr, 1);
MainWidget widget1(nullptr, 10);
MainWidget widget2(nullptr, 100);
MainWidget widget3(nullptr, 1000);
widget.show();
widget1.show();
widget2.show();
widget3.show();
```

Pour être sûr que chaque plan tourne à la même vitesse, je me base sur une vitesse à 60 fps (0,5), puis je fais le calcul suivant :

```
// Une vitesse de 0.5 pour 60 fps, donc
angularSpeed = 0.5 * (60.0 / static_cast<double>(m_fps));
```

Ainsi, pour chaque fenêtre, je conserve le même état de rotation. Si on ne fait pas cela, nous obtiendrons des vitesses différentes (faible FPS = faible vitesse).

Je rencontre cependant un problème sur ma machine, celle-ci semble limiter le nombre de FPS à ≤ 60. Je ne peux donc pas tester le résultat avec 100 et 1000 FPS.