Compte-rendu TP2
Game loops et timers

HMIN317 - Moteur de jeux

BOYER Benoît

M2 IMAGINA - Septembre 2017

Table des matières

0.1 Question 1

0.1.1 Modifier votre TP précédent pour lire une height map

Dans un premier temps, on va devoir charger l'image qui va nous servir de height map dans la fonction `GeometryEngine::initPlaneGeometry(QWidget *qw)`, auquel on aura rajouté le paramètre `QWidget *qw` pour avoir des fenêtres.

```
132     //Dans un premier temps, une boite de dialogue pour choisir
        la height map
133     QString fichier = QFileDialog::getOpenFileName(qw, "Ouvrir
        Heightmap","../","*.png");
134
135     if (fichier.isEmpty() or fichier.isNull())
136     {
137         std::cout << "Aucun fichier choisi" << std::endl;
138         std::exit(0);
139     }
```

Listing 1 – Selection de la height map

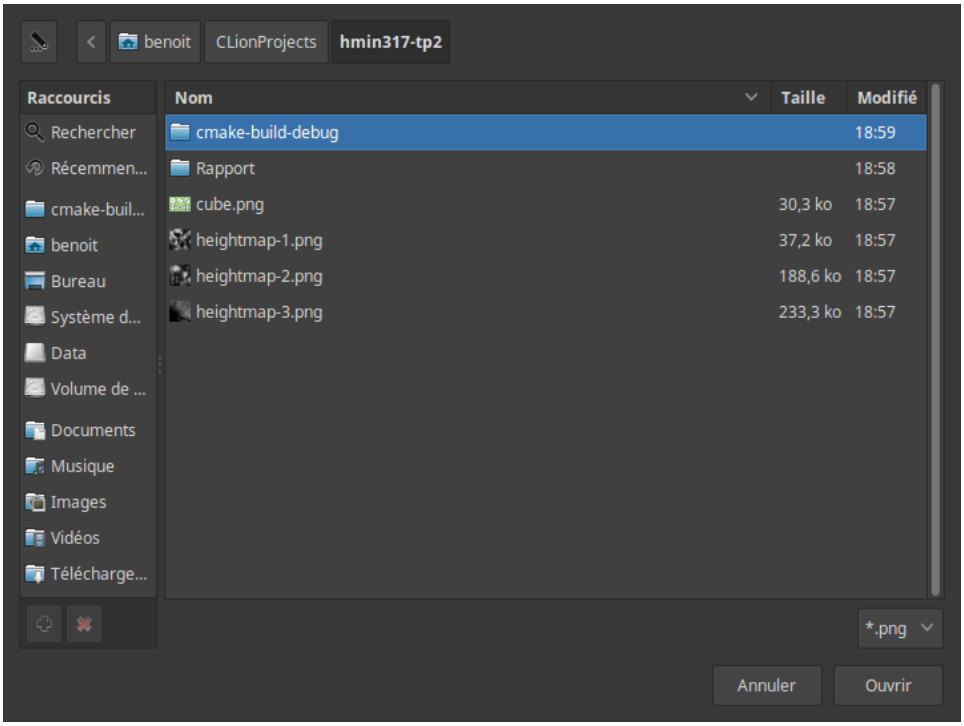

FIGURE 1 – La fenêtre de dialogue obtenue

Ensuite, on utilise une `QImage` qui servira à parcourir l'image pour récupérer la valeur des pixels pour les assigner à une hauteur.

0.1.2 Afficher un terrain à partir d'une height map

Pour générer le terrain à partir de la height map, il suffit d'appliquer la valeur de la heightmap lors de la génération des vertices :

```
146     for (int i=0;i<heightMap.height();i++)
147         for (int j=0;j<heightMap.width();j++)
148         {
149             // Vertex data for face 0
150             vertices[heightMap.width()*i+j] = {QVector3D(0.1*(i
    -8),0.1*(j-8), qGray(heightMap.pixel(i, j))/50.0), QVector2D
    (0.33*i/16.0,0.5*j/16.0)};
151         }
```

Listing 2 – Selection de la height map

L'élément important étant la valeur y, caractérisée par la fonction `qGray()`, qui va récupérer un pixel d'une `QImage`, dans notre cas qui s'appelle heightMap, et qu'on utilise la fonction `QImage::pixel(x, y)` pour récupérer la valeur d'un pixel en RGB aux coordonnées fournies.
La fonction `qGray()` va convertir la valeur en niveaux de gris, et par conséquent donner la hauteur (qui est ici divisée par 50 pour éviter d'avoir des valeurs trop élevées).

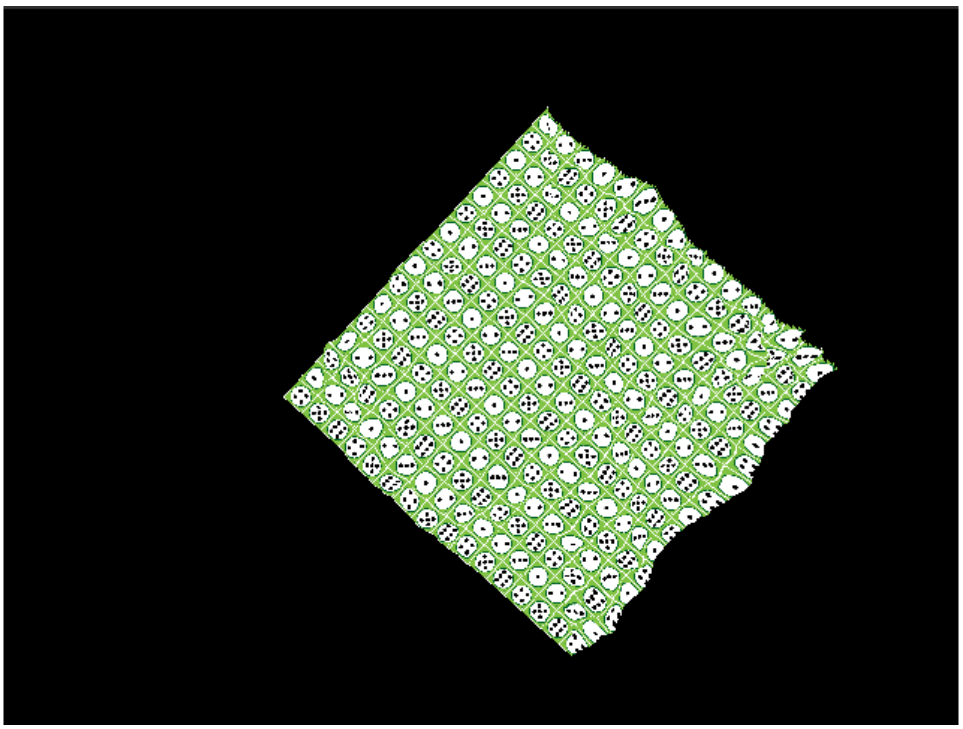

FIGURE 2 – Résultat d'un terrain qui basé sur une heightmap

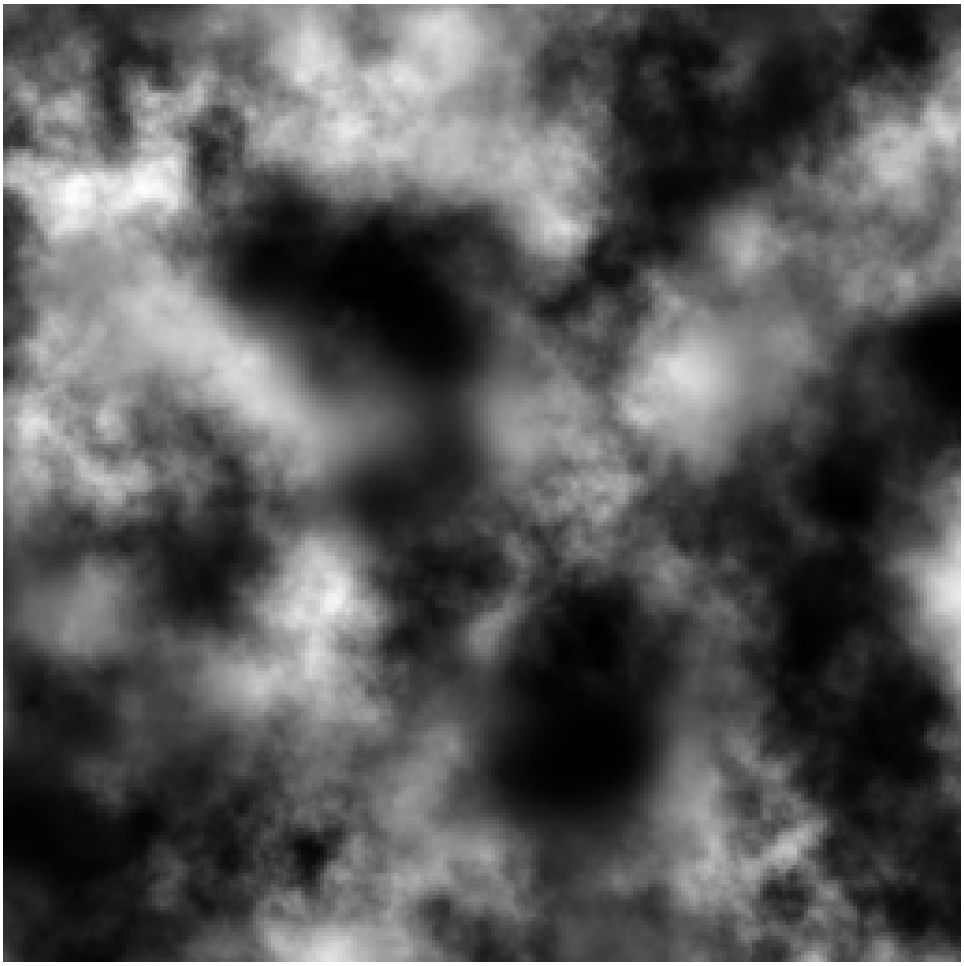

FIGURE 3 – Heightmap appliquée

0.2 Question 2

0.2.1 Modifiez votre fonction d'affichage pour regarder le terrain sous un angle de 45 degrés.

Pour regarder le terrain sous un certain angle, nous avons plusieurs possibilités :

— Incliner le terrain
— Incliner la caméra
— Incliner les deux

Dans notre cas, nous allons uniquement incliner le terrain avec une matrice de rotation, soit un Quaternion :

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

Listing 3 – Rotation du terrain

FIGURE 4 – Avant l'inclinaison

FIGURE 5 – Après l'inclinaison

0.2.2 Faire tourner le terrain autour de son origine avec une vitesse constante.

Lorsqu'on fait glisser le terrain avec la souris, on arrive à faire tourner le terrain, avec une décélération progressive, qui est lié à la friction, pour cela, on constate que tout se passe dans *mainwidget.cpp*, dans les fonctions `paintGL()` pour appliquer le rendu et `timerEvent(QTimerEvent*)` qui va appliquer la décélération, et donc une action sur un temps constant.
Cependant, il faut appliquer un mouvement de base, du coup, lors de la mise à jour, on le forcera tout le temps à tourner autour de l'axe Z, avec une vitesse qu'on réglera avec la variable `angularSpeed`.

0.3 Question 3

0.3.1 Comment est contrôlée la màj du terrain dans MainWidget ?

La mise à jour du terrain s'effectue dans la fonction héritée de QObject : *timerEvent(QTimerEvent *)*.

0.3.2 À quoi sert la classe QTimer ? Comment fonctionne-t-elle ?

D'après la documentation du site de Qt [1] :
"*The QTimer class provides repetitive and single-shot timers.*"
QTimer fournit donc des timers dont on fournit la durée, avec un signal à produire à la fin du timer. On peut donc s'en servir par exemple pour contrôler la fréquence de rafraîchissement de la fenêtre, ou bien scripter une scène (par exemple, on a 30 secondes pour atteindre la sortie sinon la porte sera verrouillée).

0.3.3 Modifier le constructeur de la classe MainWidget pour qu'il prenne en paramètre la fréquence de mise à jour.

Lors du constructeur, j'ai rajouté un paramètre `int _fps` avec une variable en plus pour la stocker, où on assignera le paramètre au timer. Ensuite, il suffit d'appliquer le framerate dans `MainWidget::initializeGL()` et plus précisément sur `timer.start()`.

0.3.4 Modifier votre programme principal pour afficher votre terrain dans quatre fenêtres différentes.

Dans *main.cpp*, il suffit de déclarer 4 MainWidget différents puis de les afficher :

```
70      MainWidget widget1(nullptr, 1);
71      MainWidget widget10(nullptr, 10);
72      MainWidget widget100(nullptr, 100);
73      MainWidget widget1000(nullptr, 1000);
74
75      widget1.show();
76      widget10.show();
77      widget100.show();
78      widget1000.show();
```

Listing 4 – Déclaration et affichage des fenêtres

On constate que les 4 fenêtres n'effectuent pas une rotation en même temps, où à 1 fps elle semble "laguer" et pour 100 et 1000fps elle semble fluide.

1. `http://doc.qt.io/qt-5/qtimer.html`

Cependant, à partir d'un framerate au-dessus de 60, il semblerait que les vitesses supérieures soient bloqués à un framerate lié à l'écran (qui est de 60Hz chez moi), cela peut être dû à la synchro verticale que les paramètres de l'ordinateur forcent.

0.3.5 Utiliser les flèches pour modifier les vitesses de rotations de votre terrain

Dans la fonction `keyPressEvent(QKeyEvent *e)`, il suffit de rajouter la détection de vitesse (dans mon cas, on utilisera + et - pour augmenter et réduire la vitesse) :

```
262     vitesse += e->key() == Qt::Key_Plus ? 1 : 0;
263     vitesse -= e->key() == Qt::Key_Minus ? 1 : 0;
```

Listing 5 – Gestion de la vitesse

On constate que chaque événement va récupérer la nouvelle information selon la fenêtre sélectionnée.

0.4 Questions bonus

0.4.1 Texturer le terrain en utilisant des couleurs

Comme pour le précédent TP, on sait que les valeurs iront de 0 à 255, donc on appliquera la couleur de la texture selon sa hauteur. Pour cela, je regarderais quelle est la valeur de la plus haute, puis j'appliquerais un pourcentage sur cette plus haute valeur, qui sera le pourcentage de blanc.

0.4.2 Jouer avec la lumière

La lumière de type soleil serait une directionnal light. Pour les matériaux et leurs effets, il faudra jouer sur ses types de rendus (voir CR_TP1).