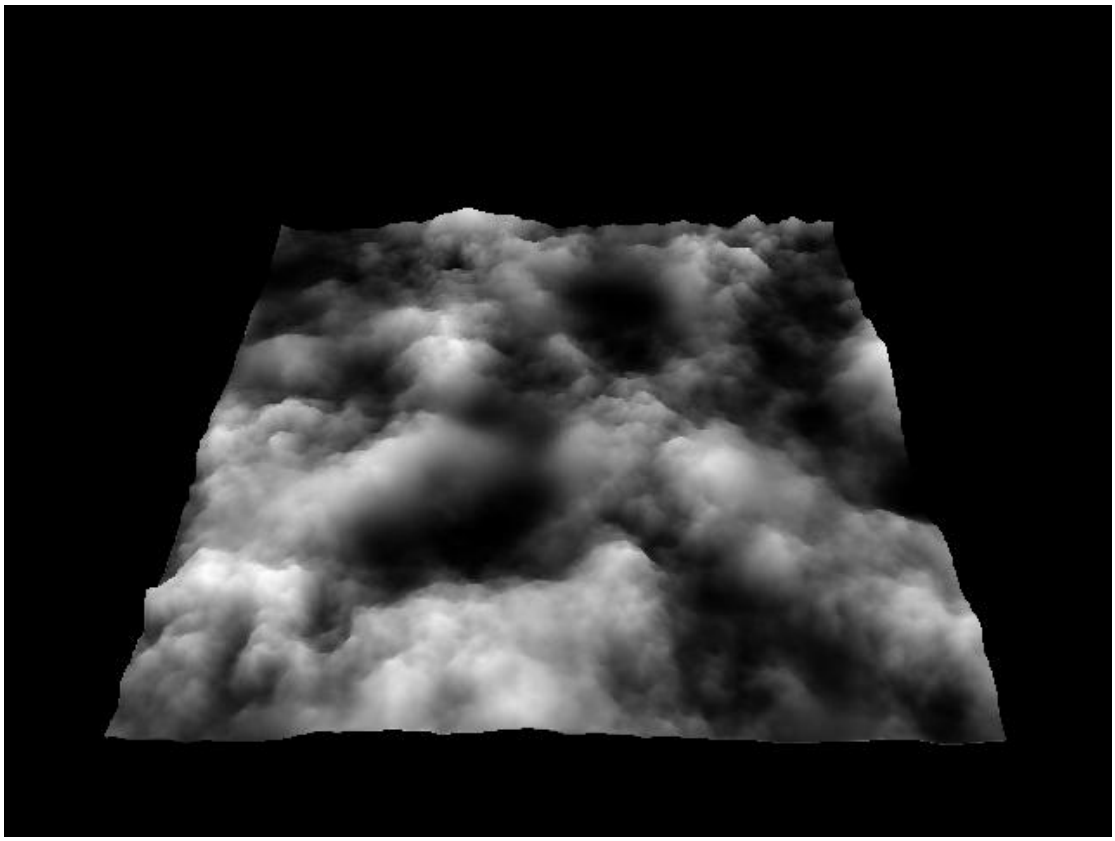
Compte rendu TP2 - Harbulot Xavier

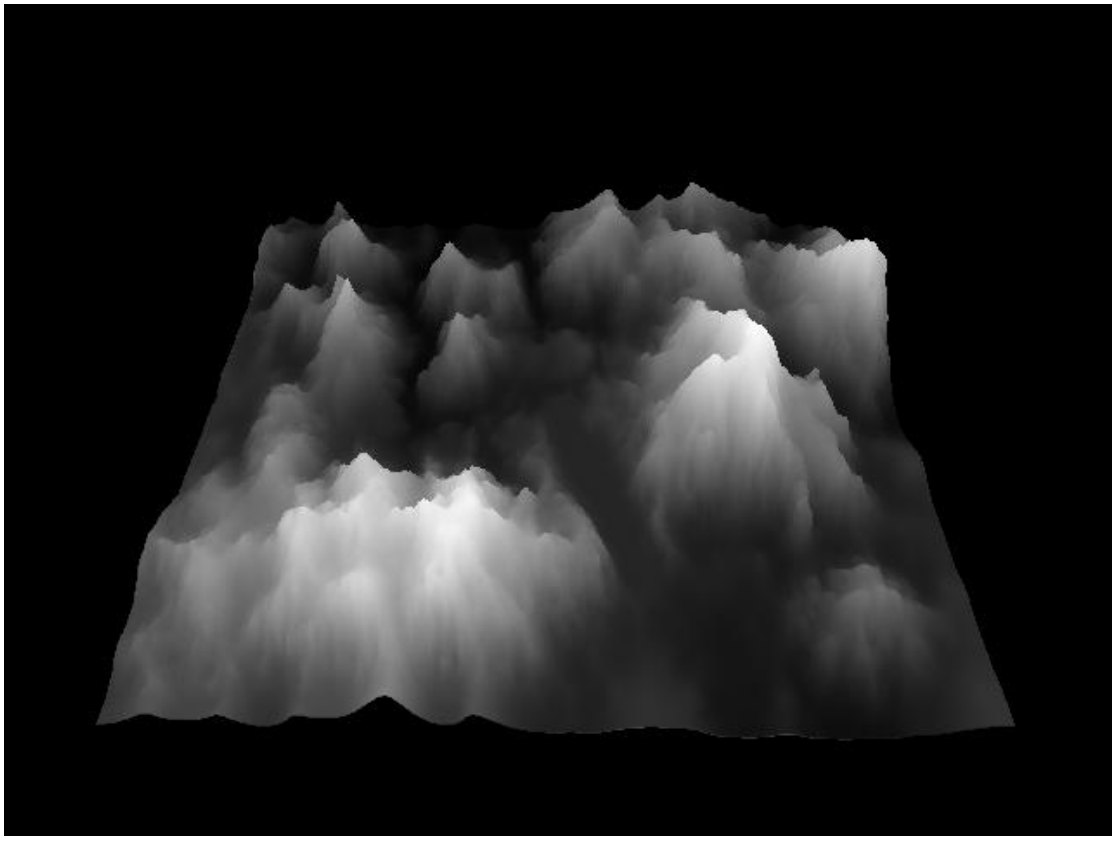
Question 1 - Affichage d'une heightmap :

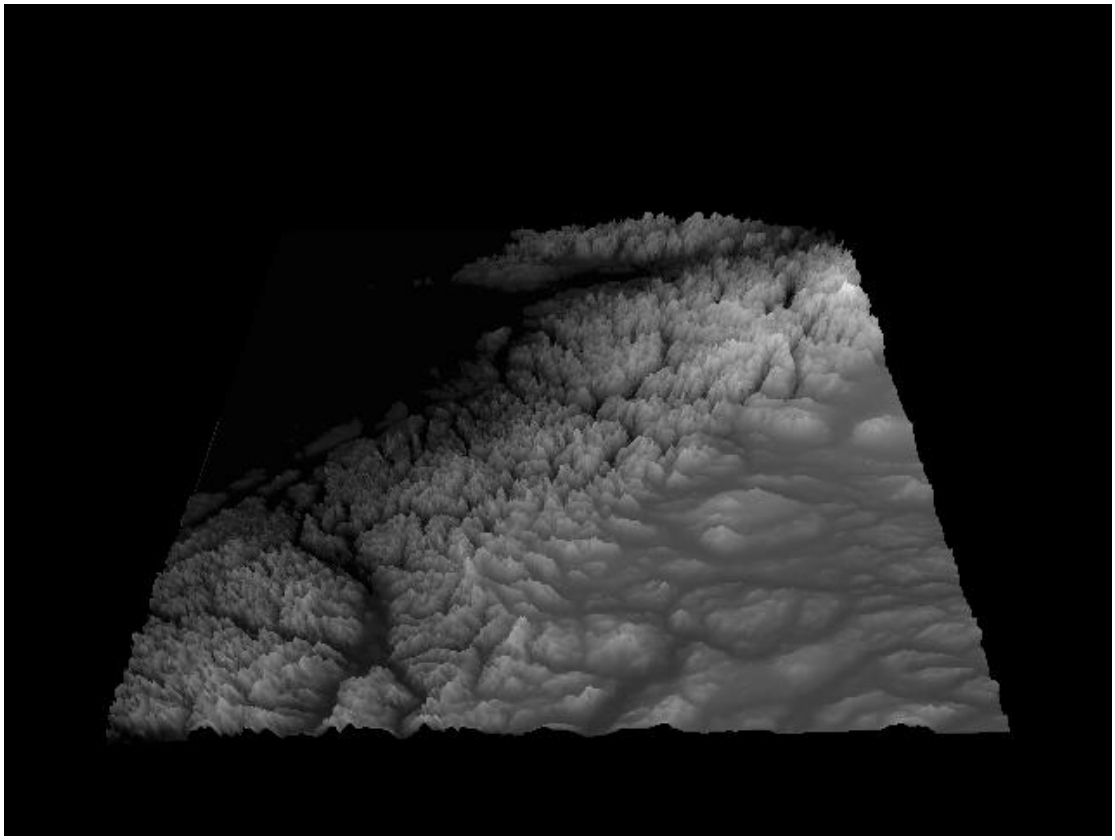
Je ne suis pas repartie du TP1 que j'avais réalisé, car je n'ai pas trouvé la solution que j'ai crée suffisamment simple pour pouvoir être réutilisée. J'ai donc téléchargé la version du TP2 sur github.

Voici le résultat obtenu pour chaque heightmap données :

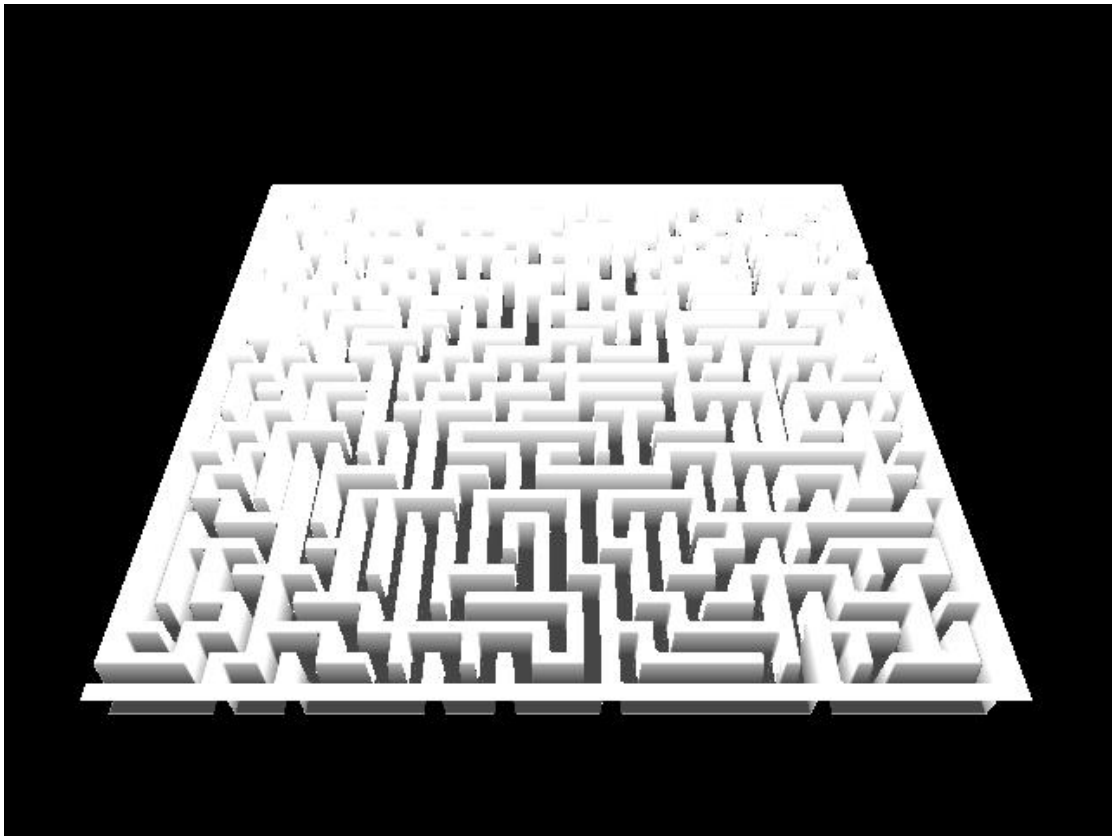


Les images des deux autres hightmaps ont étés réduites à 256x256 car le résultat n'était plus correct autrement.





Voici ensuite un labyrinthe que j'ai créé, pour le fun avec gimp :



J'ai rencontré une difficulté lorsque j'ai voulu augmenté la taille de mon tableau de points à plus de 256^2. Pour une raison que j'ignore, le résultat n'est affiché qu'en parti et n'est pas correct.

Question 2 :

Voici le code que j'ai utilisé pour pouvoir effectuer une rotation automatique de la carte :

```
    QTimer *timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(update()));
    timer->start();
```

J'ai ensuite créé fonction rotate, rajouté dans PaintGL :

```
void MainWidget::rotate() {
    QVector3D n = QVector3D(0.0,0.0,1.0).normalized();
    rotationAxis = (rotationAxis * 1 + n).normalized();
    rotation = QQuaternion::fromAxisAndAngle(rotationAxis,
speedRotation) * rotation;
}
```

Question 3 :

La mise à jour du terrain est effectué à chaque appel de la fonction update(), où qu'elle soit dans le code.
La fonction QTimer permet d'effectuer des fonctions à intervalles réguliers, ou une seule fois après X millisecondes.
On connecte la fonction timeout à l'un de nos fonctions slots. Après chaque intervalles de temps, le timeout envoie un signal qui permet de ré-utiliser la fonction souhaitée.

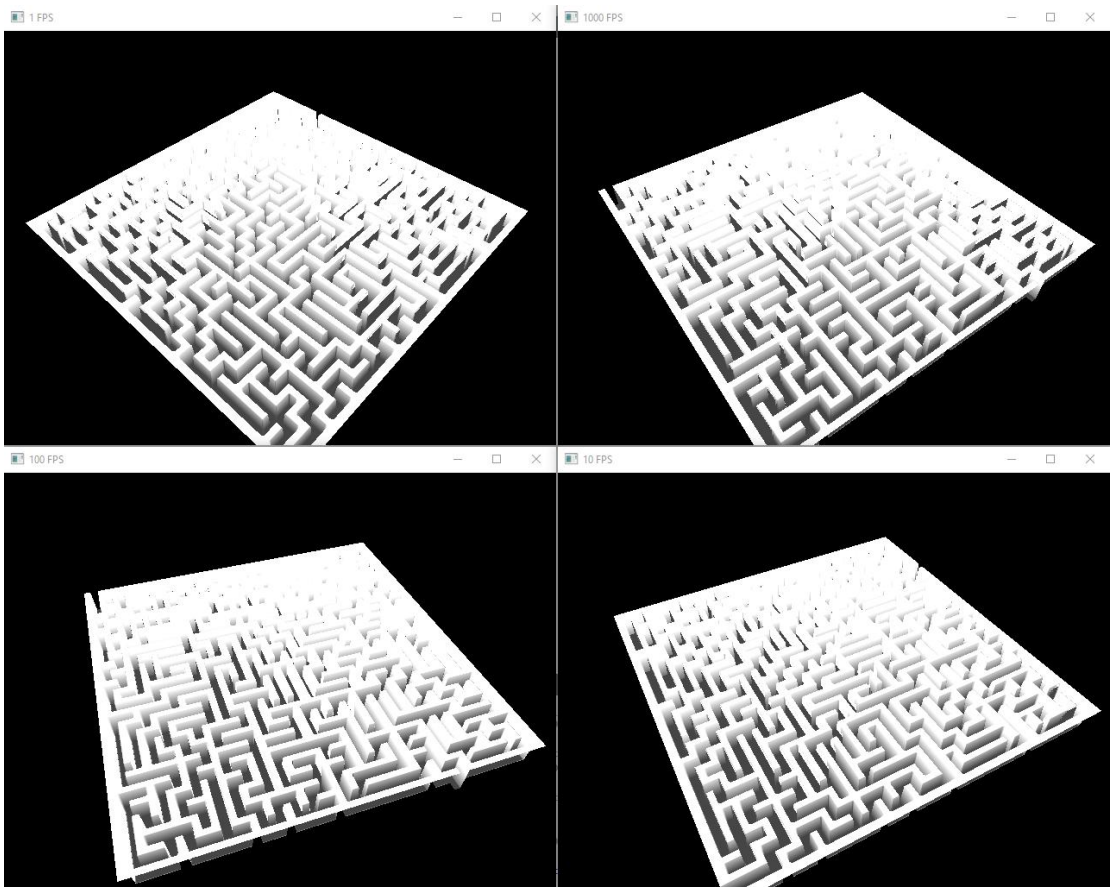
La fonction MainWidget modifié est la suivante :

```
MainWidget::MainWidget(int fps, QWidget *parent) :
(...)
{
    timer = new QTimer(parent);
    connect(timer, SIGNAL(timeout()), this, SLOT(update()));
    timer->start(fps);
}
```

Pour pouvoir afficher quatre fenêtre avec chacun un FPS différent, j'ai simplement utilisé ce code :

```
    // 1 FPS
    MainWidget widget2(1000);
    app.setApplicationName("1 FPS");
    widget2.show();

    // 10 FPS

    MainWidget widget3(100);
    app.setApplicationName("10 FPS");
    widget3.show();

    //100 FPS
    MainWidget widget4(10);
    app.setApplicationName("100 FPS");
    widget4.show();

    //1000 FPS
    MainWidget widget5(1);
    app.setApplicationName("1000 FPS");
    widget5.show();
```

Nous obtenons ce résultat :



Nous pouvons remarquer que la vitesse de rotation de chaque fenêtre diffère, allant de plus en plus vite avec la quantité de FPS utilisé. Cela est du au fait que l'on ne multiplie pas cette vitesse de rotation par un deltaTime, qui permet de garder cette vitesse synchronisé.

Pour la dernière partie, j'ai eu du mal à comprendre comment je pourrais avoir un paramètre global pour chaque fenêtre. Après beaucoup de recherche sur internet, j'ai trouvé une solution convenable : J'ai créé un fichier speed.h contenant uniquement :

```
#ifndef SPEED_H
#define SPEED_H

extern float speedRotation;

#endif // SPEED_H
```

Ainsi qu'un speed.cpp contenant :

```
#include "speed.h"

float speedRotation = 1;
```

J'inclue alors «speed.h» dans «mainwidget.cpp» et avec la fonction permettant de récupérer les inputs de l'utilisateur.

```
void MainWidget::keyPressEvent(QKeyEvent *event){
    switch (event->key()) {
    case Qt::Key_Up:
        speedRotation += 0.1;
        break;
    case Qt::Key_Down:
        speedRotation -= 0.1;
        break;
    default:
        break;
    }
}
```

On remarque alors que la vitesse de rotation augmente sur chaque fenêtre.