Calcul et Modélisation Géométrique pour l'Informatique Graphique
Ecole Centrale – Raphaëlle Chaine

TP1 – Prise en main de QT Creator, maillage et structure de données

Qt est un ensemble de bibliothèques multiplateforme. Ce type de bibliothèque répond à un objectif W.O.R.A. (Write Once, Run Anywhere). Cela signifie qu'à part quelques petites modifications, vous devriez pouvoir compiler et exécuter votre code dans les principaux systèmes d'exploitation (desktop et mobile).

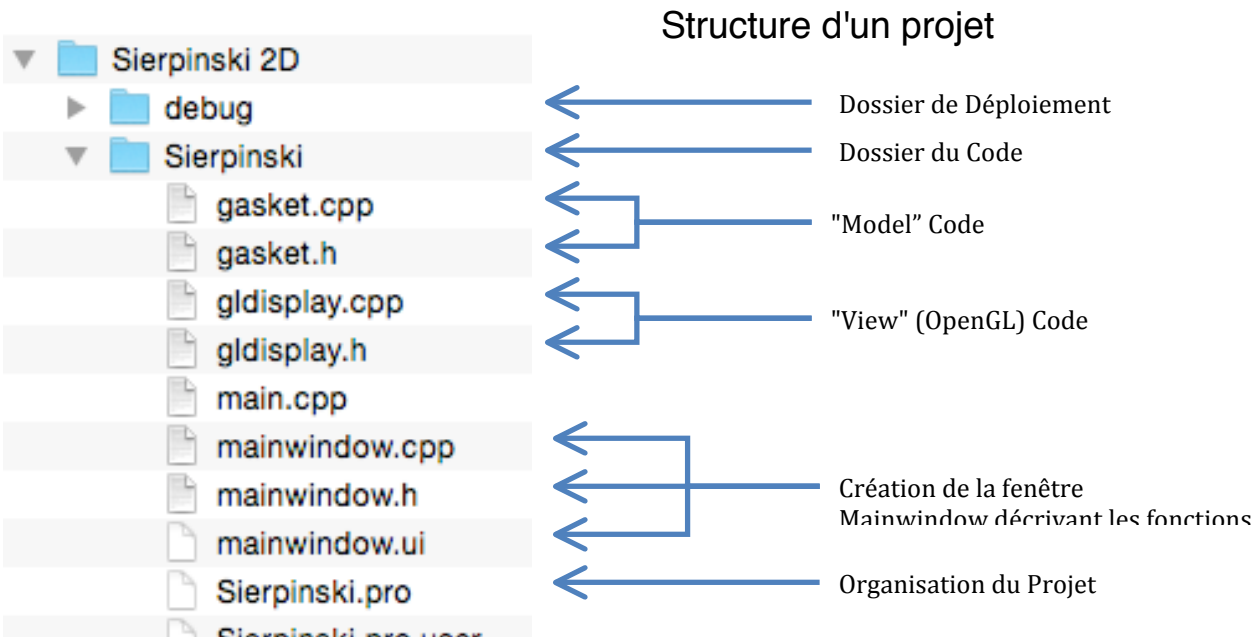
Bien que Qt puisse être utilisé dans une vaste gamme d'applications, il est surtout utilisé pour la programmation des logiciels dotés de GUI (Graphical User Interface), en raison de la simplicité avec laquelle son IDE, Qt Creator, permet la création de fenêtres et d'autres éléments graphiques.

Installation (Qt + Qt Creator) - Linux, Windows, Mac (Open Source)
http://www.qt.io/download-open-source/
Compléments :
Installation de qt et qt-creator sous linux :

- apt-get install qt5-default
- apt-get install libqt5opengl5-dev
- apt-get install qt-creator
- Si erreur à la compilation des programmes de l'archive fournie : remplacer #include <gl.h> remplacer par #include <GL/gl.h>

Installation de qt et qt-creator sous windows (avec MinGW ou avec Visual Studio): tutoriel.

Récupérez l'archive d'exemples à l'adresse suivante :
https://dl.univ-lyon1.fr/84xhtaass



Mainwindow : description des évènements qui peuvent se produire dans la fenêtre, éventuellement associés à des signaux reçus par des widgets.

ui : fichier xml décrivant les Qwidgets. La modification de ce fichier via l'utilisation de Qt designer dans Qt creator entraine l'enrichissement du module Mainwindow, qu'il convient ensuite d'éditer à la main.

Lors de la compilation Qt transforme le code C++ en un autre code source (présent dans Built), dans lequel les macros auront été traitées. Ainsi tous les widgets qui auront été définis comme descendant de la fenêtre seront également disponibles via le pointeur ui correspondant à un attribut de la fenêtre. Connect est une fonction statique de la fenêtre.

Ouvrir et exécuter un projet

1) Ouvrir Sierpinski.pro.

2) Donner les informations spécifiques à la machine et au compilateur.
(N.B.: Il ne faut pas placer le Dossier de Déploiement dans le Dossier du code)

3) Compiler et exécuter le projet.

Où faire les modifications ?

1) En ce que concerne OpenGL: **gldisplay.h** et **gldisplay.cpp**

2) Définitions des données géométriques et des données graphiques: **gasket.h** et **gasket.cpp**.

Remarque : la classe GLDisplay offerte par le module gldisplay contient une donnée membre de type Gasket et la fonction GLDisplay::paintGL invoque la fonction draw de ce Gasket. Dans les versions 3D, la zone délimitant le scène à afficher se trouve dans GLDisplay::resizeGL.

3) Interface: **mainwindow.ui**, **mainwindow.h**, **mainwindow.cpp**
Il faut savoir que dans Qt, les objets sont non seulement dotés d'attributs et de fonctions membres ou de méthodes, mais aussi de signaux et de slots. Les slots sont des méthodes connectées à des signaux. Ainsi il sera possible de connecter un slot d'un objet au signal d'un autre.

Exemple d'ajout d'un bouton

1) Ouvrir **mainwindow.ui** dans qt-creator.

2) Placer un PushButton dans la zone grisée. Attention : dans l'arborescence, le pushButton ne doit pas être sur un niveau inférieur au widget GLDisplay ; ils doivent être sur le même plan. Pour cela, créer le PushButton en le glissant au dessus.

3) Retourner dans le mode Editer

Ajouter un **public slot** à un objet Qt (ici dans la classe mainWindow)

```
// mainwindow.h
...
public slots:
    void onButton();
...

// mainwindow.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

void MainWindow::onButton()
{
    ... code de l'action à effectuer
    par exemple ui->pushButton->setText("Released");
 }
```

4) Connecter le **signal** au **slot**.

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui->pushButton, SIGNAL(released()), this,
            SLOT(onButton()));
}

//ui est un objet qui contient tous les widgets du formulaire
```

Raccourcis claviers

- Ctrl + r : Run
- Ctrl + b : Build
- Ctrl + espace : pour compléter une variable ou un nom de fonction
- F4 : Passage du .hpp au .cpp
- Ctrl + / : Pour commenter ou décommenter une partie du code
- Ctrl + e, puis 2 : Séparation de l'écran en 2.

Chargement d'un maillage

1) Mettre en place une structure de données maillage triangulé.
    - On commencera tout d'abord par modéliser un maillage par un vector de ses sommets et un vector de ses faces. Dans ce premier modèle, les faces ne sont pas attachées entre elles. Elles sont simplement représentées par les indices de leurs 3 sommets.
    - Mettre ensuite en place la structure de données correspondant au modèle topologique basé sommets et faces vu en cours. Dans ce modèle, les faces sont attachées entre elles.

2) Chargement et affichage de maillages rudimentaires pour tester votre structure de données.
- Un tétraèdre
- Un cube (trianguler ses 6 faces)
- Une pyramide à base carrée
- Une boîte englobante 2D (composée de 2 triangles) dont les bords sont reliés à un sommet infini à l'arrière.

3) Ecrire une routine de chargement, dans votre structure de donnée, d'un maillage triangulé lu dans un fichier au format OFF (vous pouvez télécharger à l'adresse https://dl.univ-lyon1.fr/tnf9e0ou un joli exemple de maillage réalisé par un artiste).
Format OFF :

Nombre de sommets s
Nombre de faces c
Description des s faces (séquence d'indices des sommets de la face, précédée de son nombre de sommets).

4) Utiliser Qt Creator pour écrire une petite application d'affichage de votre maillage (vous pouvez partir de l'exemple Siperpinski 2D ou 3D, suivant qu'il s'agit d'un maillage 2D ou 3D). On pourra également mettre en place quelques boutons ou menus pour changer de mode d'affichage (affichage en fil de fer, affichage plein des faces).

Veillez à mettre en œuvre une approche logicielle soignée.

Remarque : Les instructions OpenGL utilisées dans l'archive correspondent à une version désormais caduque d'OpenGL depuis l'essor de la programmation par shader. L'exemple possède néanmoins l'avantage d'être simple à comprendre, et vous pourrez le faire évoluer au fur et à mesure de vos apprentissages en synthèse d'Image.
Dans Qt4 et Qt5, le module QtOpenGL fournit la classe QGLWidget et ses dérivées (toutes sont préfixées par QGL). Néanmoins, il est possible dans Qt5 d'utiliser le module QtGui, contenant des classes préfixées par QOpenGL.