

TP « Découverte d'un IDE : QT »

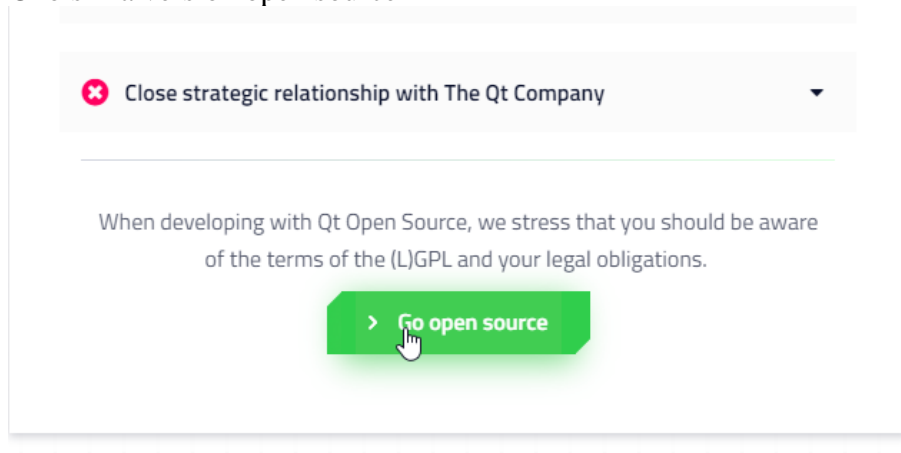
Le but de ce TP est la manipulation d'un outil de développement intégré (IDE) qui va vous permettre d'écrire, compiler, débbugger du code facilement.
J'ai choisi l'environnement QT (qui se dit « *cute* » en anglais).

1 - Installation

1.1 Download

Allez sur le site <https://www.qt.io/download>

Choisir la version open source



Accepter les obligations open source

Your Open Source Obligations

(L)GPL has been chosen as the primary open-source license for Qt, because it provides your end users' the following Free Software "four degrees of freedom":

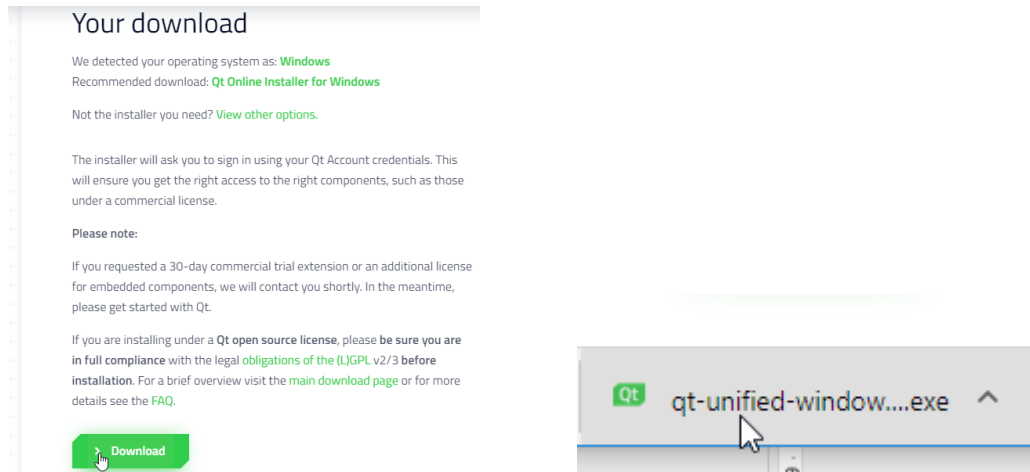
- To redistribute copies of your or their version of your program, so they can help their "neighbor".
- To distribute copies of their modified versions of your program to others. By doing this they can give the whole community a chance to benefit from their changes. *Access to the source code is a precondition for this.*
- To run the program as they wish, for any purpose.
- To study how the program works, and change it so it does their computing as they wish. *Access to the source code is a precondition for this.*

All users of Qt under (L)GPL must adhere to this and fully meet all the requirements set by the (L)GPL. It is recommended that a thorough legal analysis is conducted when choosing to use any open-source licenses. In many cases, the (L)GPL is a viable solution to use, but it is important that the freedoms of the (L)GPL are not restricted from the user of an application or device using an (L)GPL library such as Qt, which may be difficult to achieve in some use cases.

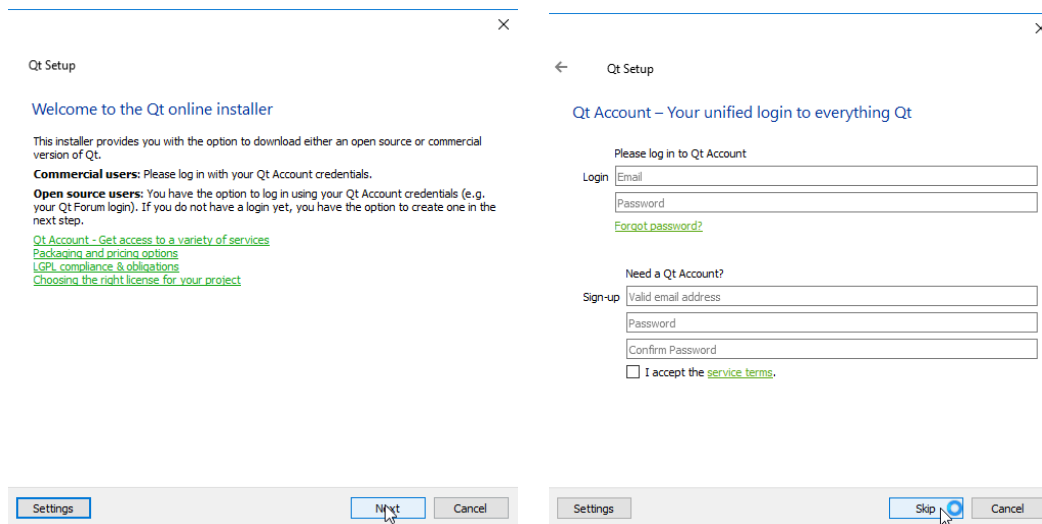
I have read and understand my obligations when developing under the open source license of Qt. I confirm that my project complies with these terms.



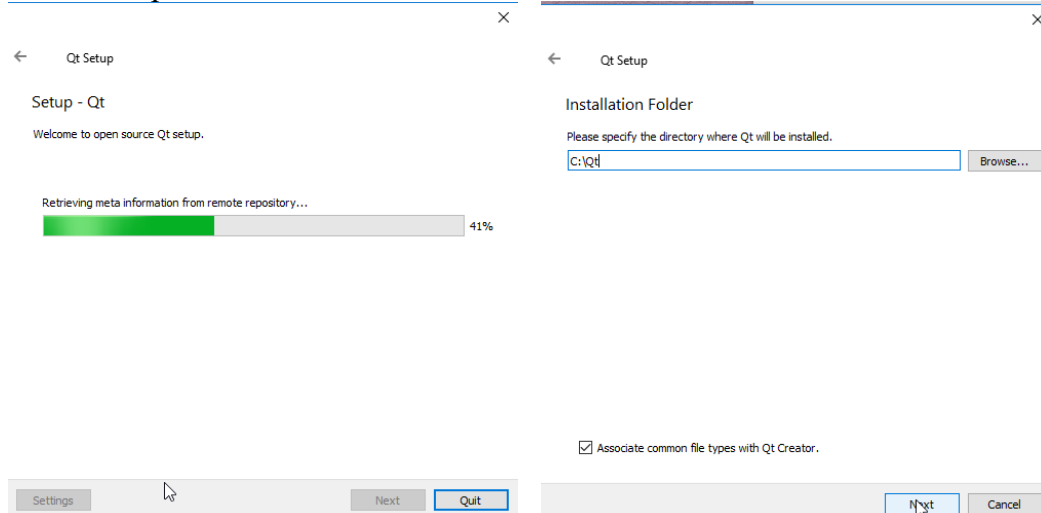
Pour Windows, cliquer sur download, puis sur l'installer



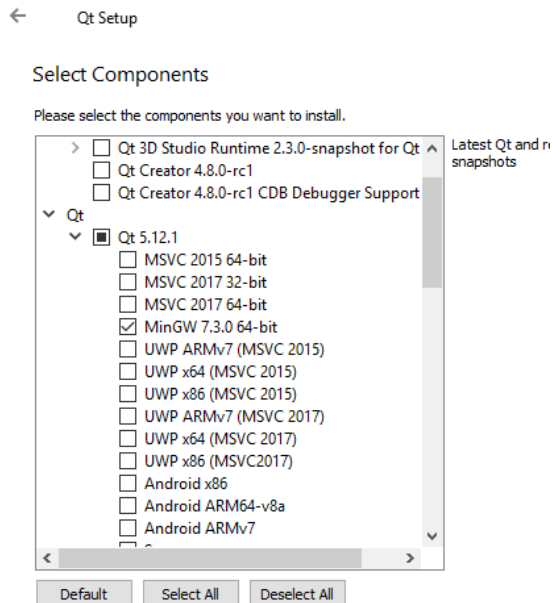
Démarrer l'installateur, cliquer sur skip pour la phase de « Qt Account »



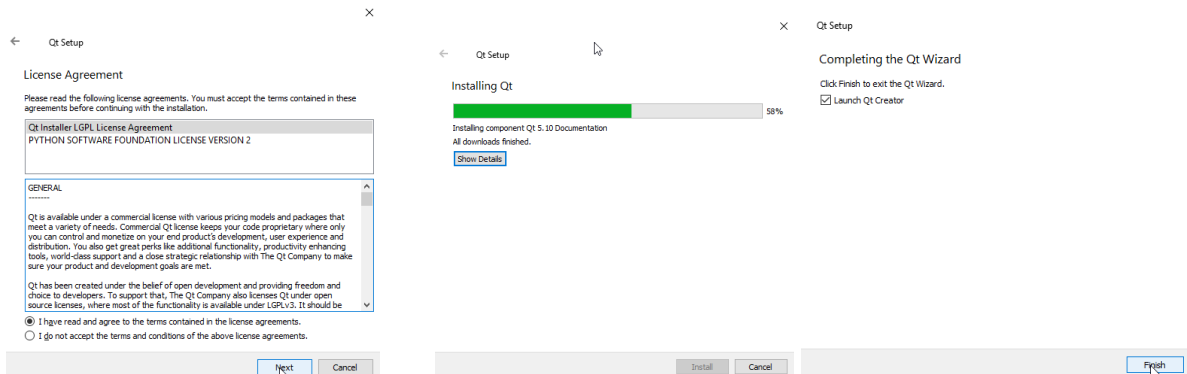
L'installateur démarre, puis cliquer sur Next. Sur le panel « installation Folder » cliquer sur Next
Attention : il est préférable de ne pas changer le chemin par défaut C:\Qt et surtout de pas mettre d'espace dans le nom du chemin.



IMPORTANT : bien choisir la version MinGW 7.3 64-bit



Accepter les « License Agreement », L'installer continue et en fonction du débit réseaux, il faut compter jusqu'à 10 min. pour terminer l'installation de Qt.



Vous êtes enfin prêt à lancer Qt Creator

1.2 Configurer QtCreator.

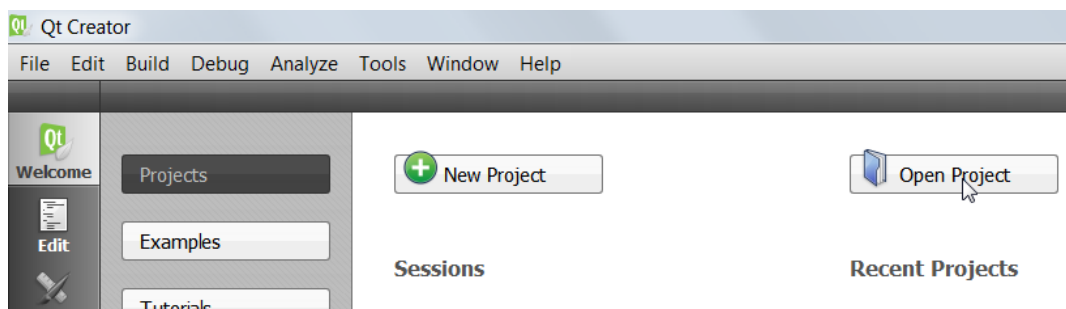
1.3 Récupérer un projet

En utilisant votre shell MSYS2, avec git.

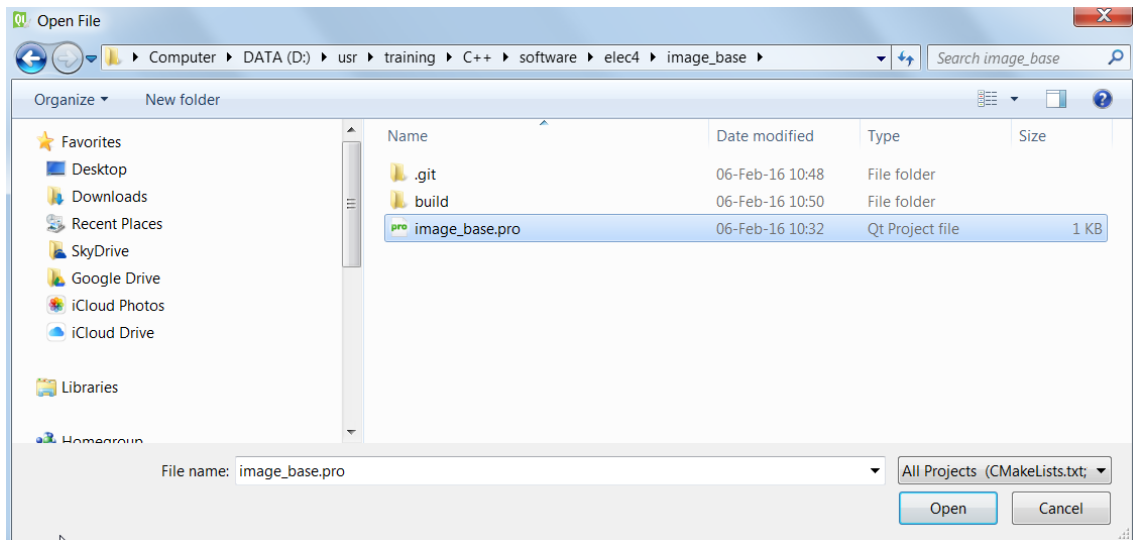
```
shell> git clone https://github.com/elec4/image_base.git
Cloning into 'image_base'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
```

1.4 Lancer un projet dans QT

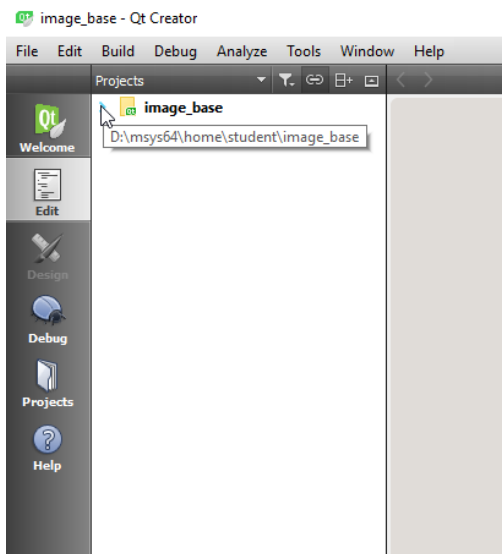
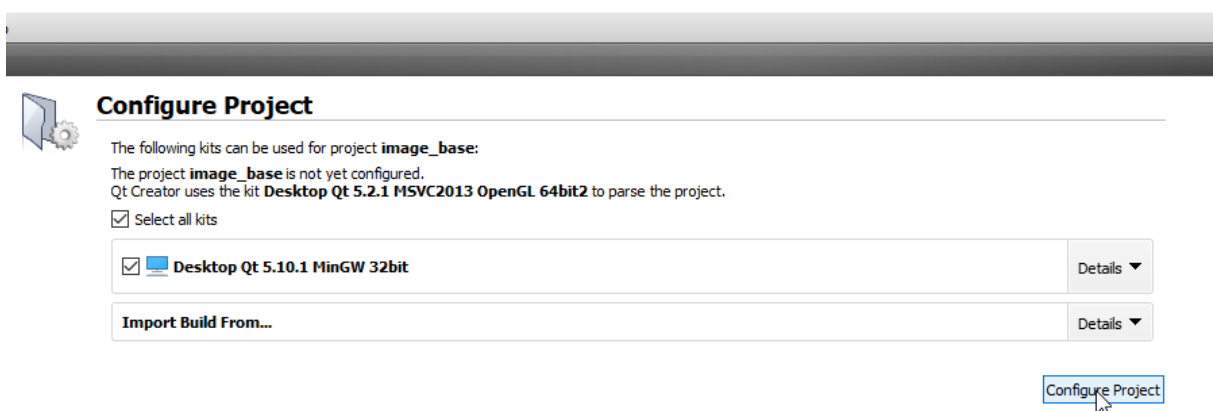
Dans Qt Creator, vous cliquez sur « *Open Project* ».



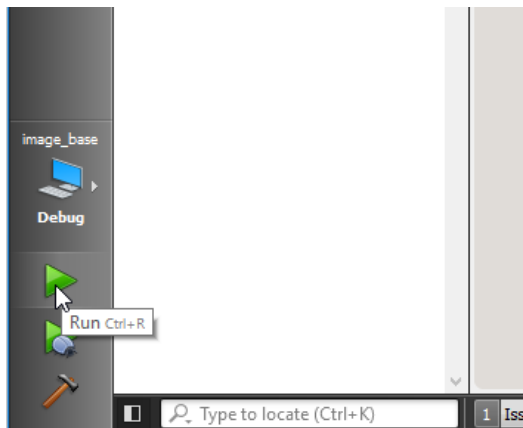
Vous devez ouvrir le fichier `image_base.pro`. Attention : le chemin d'accès est le chemin Windows correspondant au dossier `image_base` créé par la command git ci-dessus.
Chez moi : `D:/usr/training/C++/software/elec/image_base`



Puis cliquer sur Configure Project

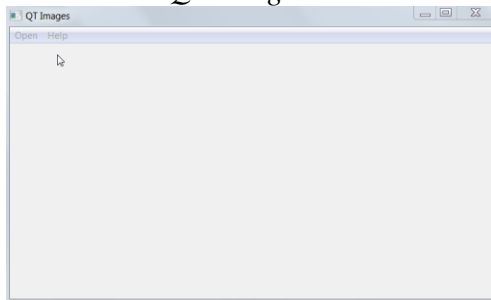


Puis cliquer sur Run (compilation et lancement de l'application)

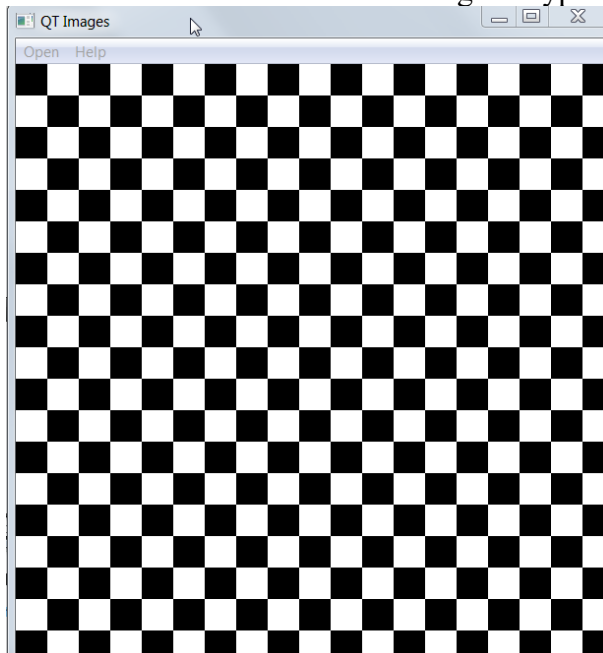


Vous cliquez ensuite sur le marteau (« build ») puis sur la flèche verte (« run »)

La fenêtre « *QT Images* » s'affiche.

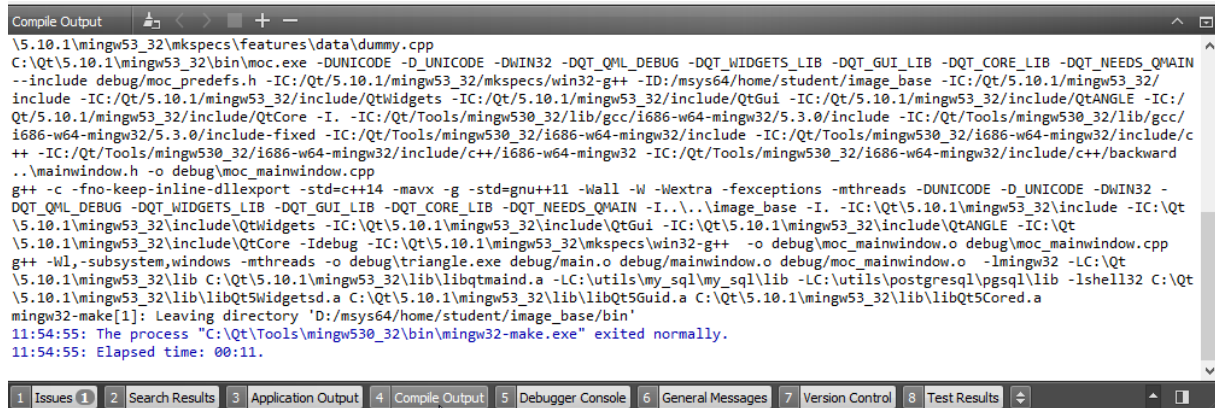


Sélectionner alors « *Open -> CheckBoard Image* »
Vous devez voir s'afficher une image de type damier.



1.5 Explorer les fonctionnalités de « Qt Developer »

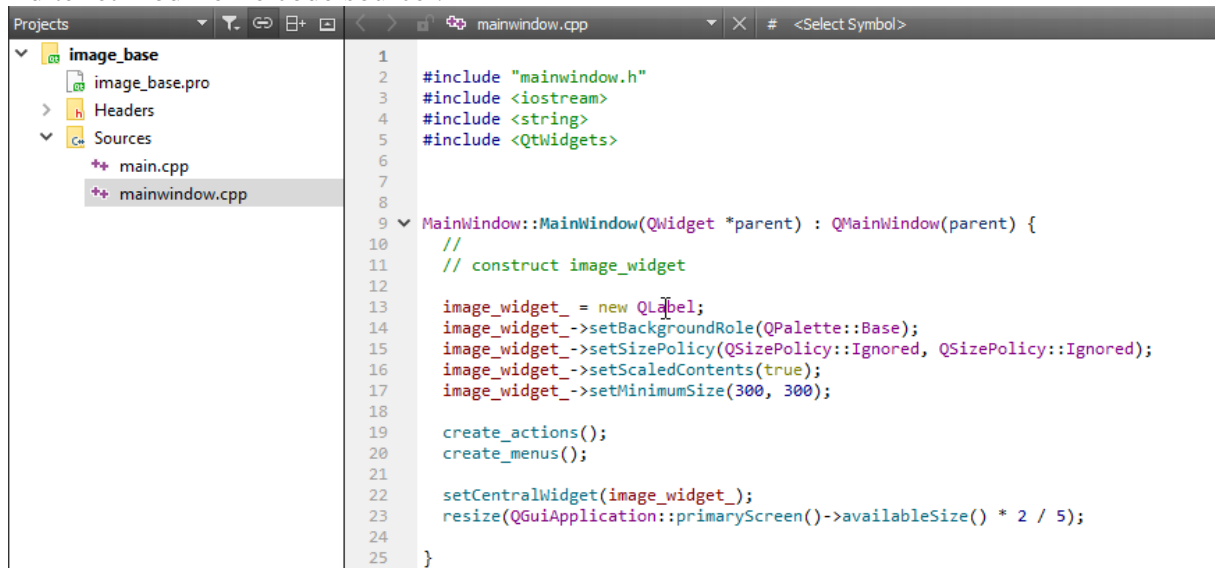
Voir les résultats de compilation :



```

Compile Output
\5.10.1\mingw53_32\mspecs\features\data\dummy.cpp
C:\Qt\5.10.1\mingw53_32\bin\moc.exe -DUNICODE -D_UNICODE -DWIN32 -DQT_QML_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -DQT_NEEDS_QMAIN
--include debug/moc_predefs.h -IC:/Qt/5.10.1\mingw53_32\mspecs\win32-g++ -ID:/msys64/home/student/image_base -IC:/Qt/5.10.1\mingw53_32/
include -IC:/Qt/5.10.1\mingw53_32/include/QtWidgets -IC:/Qt/5.10.1\mingw53_32/include/QtGui -IC:/Qt/5.10.1\mingw53_32/include/QtANGLE -IC:/
Qt/5.10.1\mingw53_32/include/QtCore -I. -IC:/Qt/Tools/mingw530_32/lib/gcc/i686-w64-mingw32/5.3.0/include -IC:/Qt/Tools/mingw530_32/lib/gcc/
i686-w64-mingw32/5.3.0/include-fixed -IC:/Qt/Tools/mingw530_32/i686-w64-mingw32/include -IC:/Qt/Tools/mingw530_32/i686-w64-mingw32/include/c
++ -IC:/Qt/Tools/mingw530_32/i686-w64-mingw32/include/c++/i686-w64-mingw32 -IC:/Qt/Tools/mingw530_32/i686-w64-mingw32/include/c++/backward
..\mainwindow.h -o debug/moc_mainwindow.cpp
g++ -c -fno-keep-inline-dllexport -std=c++14 -mavx -g -std=gnu++11 -Wall -W -Wextra -fexceptions -mthreads -DUNICODE -D_UNICODE -DWIN32 -
DQT_QML_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -DQT_NEEDS_QMAIN -I. -I. -Iimage_base -I. -IC:/Qt/5.10.1\mingw53_32/include -IC:/Qt
/5.10.1\mingw53_32/include/QtWidgets -IC:/Qt/5.10.1\mingw53_32/include/QtGui -IC:/Qt/5.10.1\mingw53_32/include/QtANGLE -IC:/Qt
/5.10.1\mingw53_32/include/QtCore -Idebug -IC:/Qt/5.10.1\mingw53_32\mspecs\win32-g++ -o debug/moc_mainwindow.o debug/moc_mainwindow.cpp
g++ -Wl,-subsystem,windows -mthreads -o debug/triangle.exe debug/main.o debug/mainwindow.o debug/moc_mainwindow.o -lmingw32 -LC:/Qt
/5.10.1\mingw53_32\lib C:/Qt/5.10.1\mingw53_32\lib\libqtmaind.a -LC:/utils/my_sql/my_sql\lib -LC:/utils/postgresql\pgsql\lib -lshell32 C:/Qt
/5.10.1\mingw53_32\lib\libQt5Widgets.a C:/Qt/5.10.1\mingw53_32\lib\libQt5Gui.a C:/Qt/5.10.1\mingw53_32\lib\libQt5Core.a
mingw32-make[1]: Leaving directory 'D:/msys64/home/student/image_base/bin'
11:54:55: The process "C:/Qt/Tools/mingw530_32/bin/mingw32-make.exe" exited normally.
11:54:55: Elapsed time: 00:11.
  
```

Editer et modifier le code source :



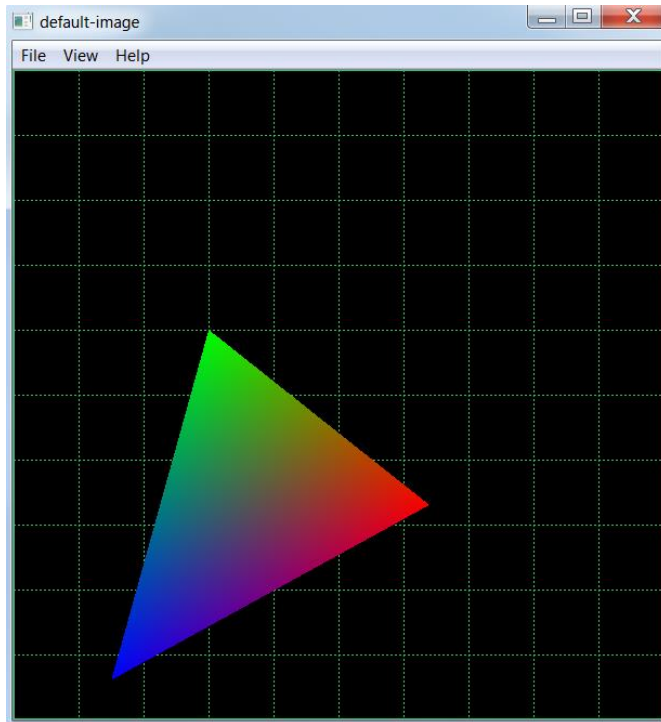
```

Projects
  image_base
    image_base.pro
    Headers
    Sources
      main.cpp
      mainwindow.cpp

mainwindow.cpp
1
2 #include "mainwindow.h"
3 #include <iostream>
4 #include <string>
5 #include <QtWidgets>
6
7
8
9 MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) {
10 //
11 // construct image_widget
12
13 image_widget_ = new QLabel;
14 image_widget_ -> setBackgroundRole(QPalette::Base);
15 image_widget_ -> setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
16 image_widget_ -> setScaledContents(true);
17 image_widget_ -> setMinimumSize(300, 300);
18
19 create_actions();
20 create_menus();
21
22 setCentralWidget(image_widget_);
23 resize(QGuiApplication::primaryScreen()->availableSize() * 2 / 5);
24
25 }
  
```

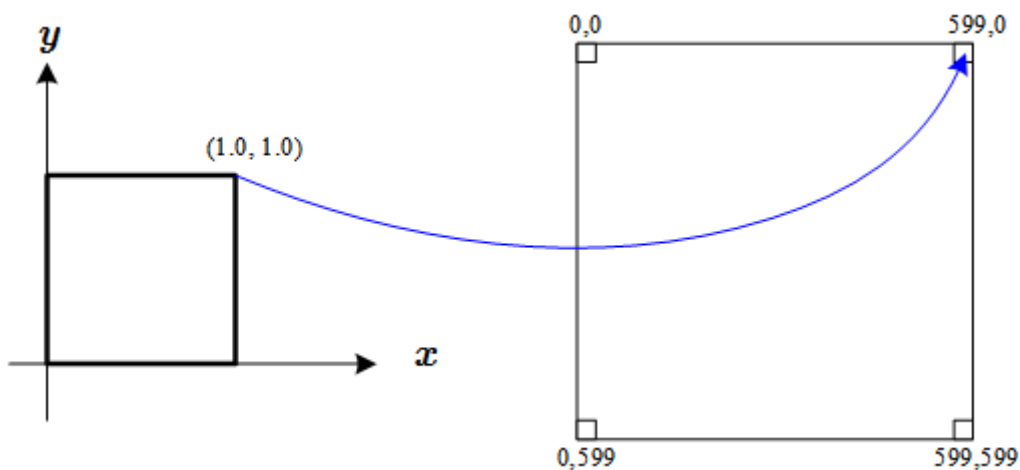
2 - Génération d'une nouvelle image

Vous devez maintenant modifier le programme `image_base`, pour générer une nouvelle image (triangle) de taille 600x600 qui devra ressembler à l'image suivante :

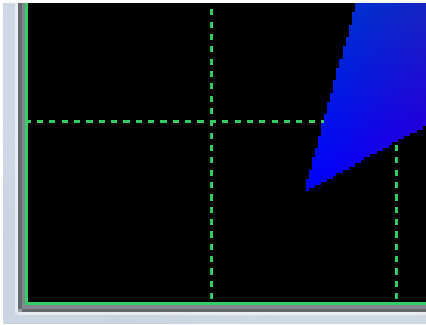


2.1 Dessin cadre et grille

On utilisera comme référence un repère orthonormé standard avec un carré de coté 1.0 dont le bord bas-gauche est en $(0,0)$. Vous devrez faire les transformations nécessaires pour convertir les coordonnées pixels vers/depuis les coordonnées standards.



Dans ce carré unité de couleur noir, vous dessinerez un cadre et une grille en pointillé au pas $1/10^{\text{ème}}$ de couleur verdâtre RGB=(51, 204,102).



2.2 Dessin du triangle

Dans ce carré unité, les coordonnées des 3 couleurs primaires (RGB) sont :

Rouge	0.64	0.33
Vert	0.3	0.6
Bleu	0.15	0.06

La couleur de chaque point à l'intérieur du triangle est calculée en utilisant une interpolation barycentrique (voir référence ci-dessous).

Conseil :

- Faire en premier un triangle d'une seule couleur.
-

2.3 Reference

Introduction sur QT : Partie 3 du document pdf « programmez avec le langage c++ »

Documentation sur QT : <http://doc.qt.io/qt-5/index.html>

Interpolation barycentrique :

http://en.wikipedia.org/wiki/Barycentric_coordinate_system

<https://classes.soe.ucsc.edu/cms160/Fall10/resources/barycentricInterpolation.pdf>

Quelques fonctions QT intéressantes

```
QRgb qRgb(int r, int g, int b);
void QPainter::drawLine(const QPoint &p1, const QPoint &p2);
void QPainter::setPen(const QPen & pen)
void QPen::setDashPattern(const QVector<qreal> & pattern)
void QImage::setPixel(int x, int y, uint index_or_rgb)
```

3 - Amélioration de l'image

Faire une recherche image avec les mots clés suivants « CIE 1931 » sur GOOGLE ou BING.
Une rapide comparaison entre ces images et l'image générée par votre programme montre qu'il manque une couleur, laquelle ?
Expliquer pourquoi cette couleur ne peut pas être atteinte par votre programme.

Cette couleur est située aux coordonnées suivantes : 0.3127 0.32903

Modifiez votre programme en conséquence pour que votre triangle coloré ressemble plus aux images trouvées par GOOGLE ou BING.

3.1 Livrables pour ce TP

Aucun : ce TP n'est pas noté.

Attention : le code de ce TP sera utilisé dans un prochain TP donc bien faire le boulot SVP.