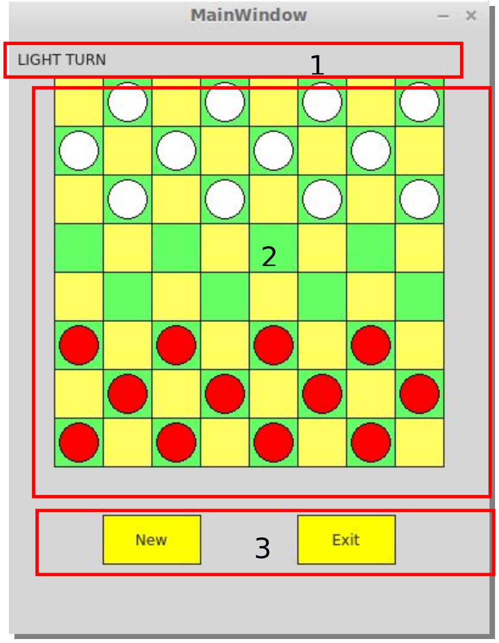
Project Checker

In this project, you are required to make a simple checkers game(http://en.wikipedia.org/wiki/Draughts).

A checker game is played by two opponents, on opposite sides of the 8*8 game board as the picture shows. One player has the dark pieces; the other has the light pieces. Players alternate turns. A player may not move an opponent's piece. The player with the light pieces moves first unless stated otherwise. A move consists of moving a piece diagonally to an adjacent unoccupied square. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured (and removed from the game) by jumping over it.



LIGHT TURN

And to make it simple, if one side move one piece to the bottom of the opponent, or the opponent has no pieces remaining on the board, he wins the game.

Requirements

The program should has three parts.

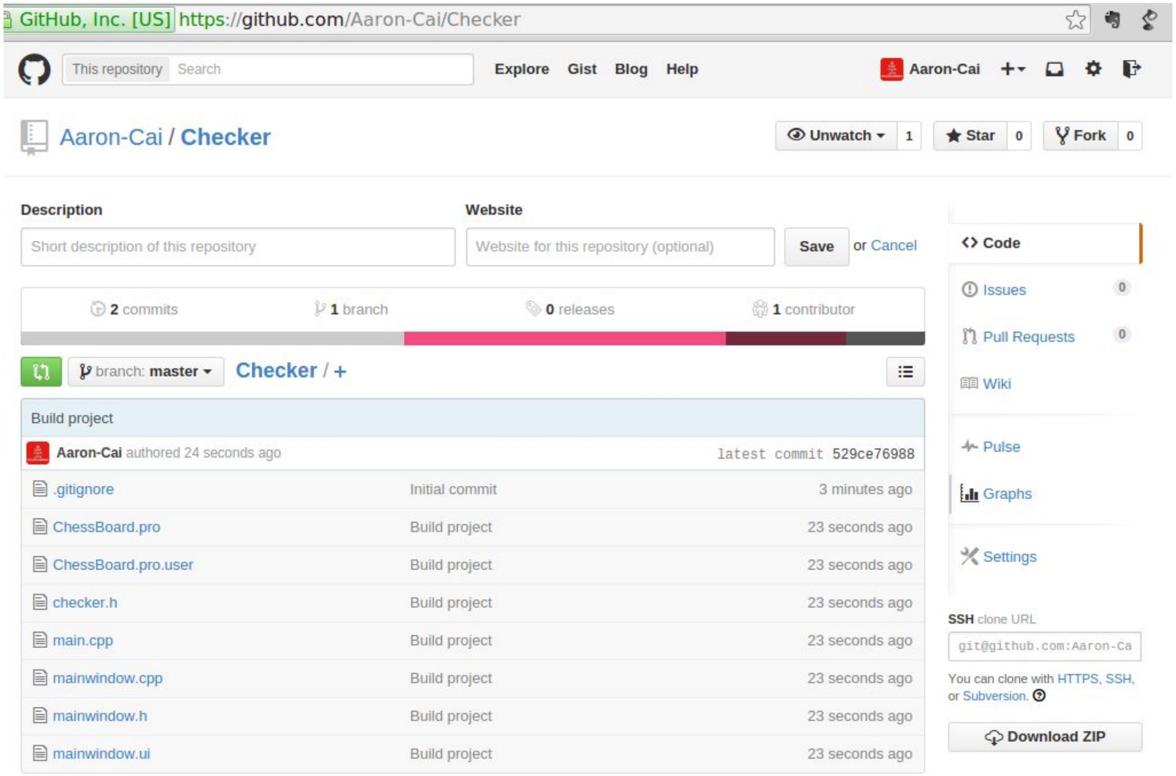
1. Region 1 is for game message showing, such as light side's turn or dark side's, light side wins or dark side wins
2. Region 2 is a 8*8 board with light and dark pieces, players could click the piece to move it.
3. Region 3 consists of two button, one is for a new game and the other is for exiting the program.

You are required to use the provided code to implement a program like the picture show. It should support **two players** and allow them **move their pieces one by one** and be able to show **who wins and who loses**.

First you need to get the code from the Internet.

How to get the code?

Step1: download the code from https://github.com/Aaron-Cai/Checker



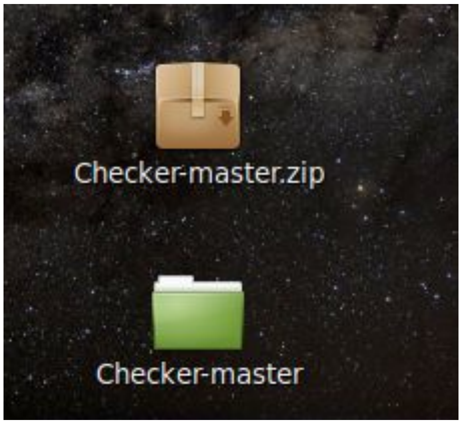
Build project

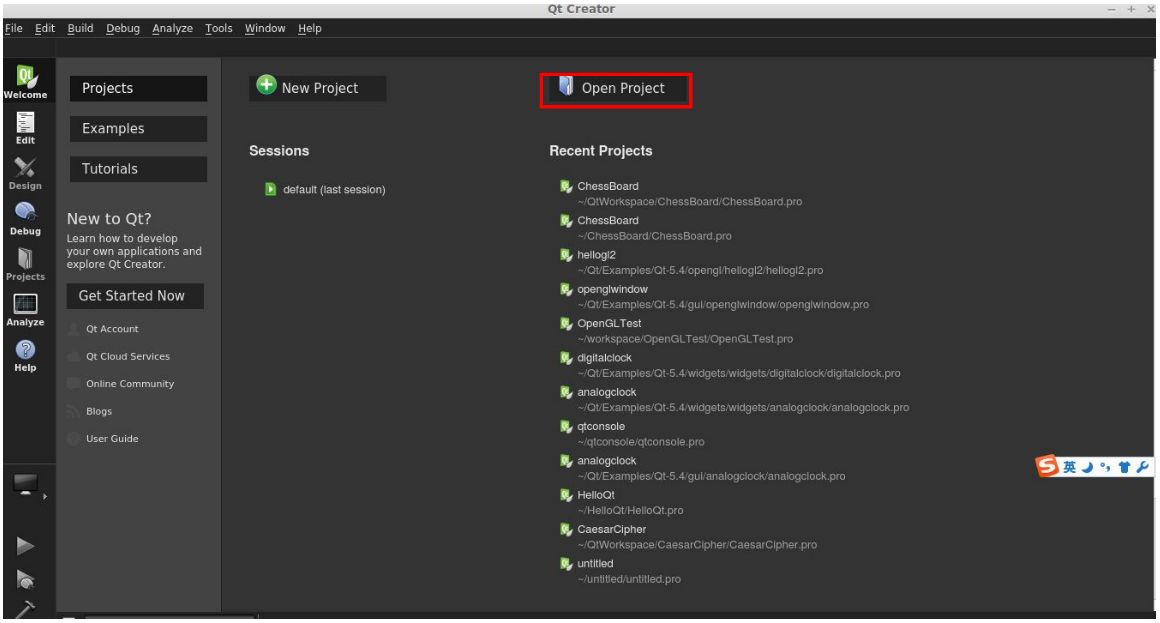
Aaron-Cai authored 24 seconds ago — latest commit 529ce76988

.gitignore	Initial commit	3 minutes ago
ChessBoard.pro	Build project	23 seconds ago
ChessBoard.pro.user	Build project	23 seconds ago
checker.h	Build project	23 seconds ago
main.cpp	Build project	23 seconds ago
mainwindow.cpp	Build project	23 seconds ago
mainwindow.h	Build project	23 seconds ago
mainwindow.ui	Build project	23 seconds ago

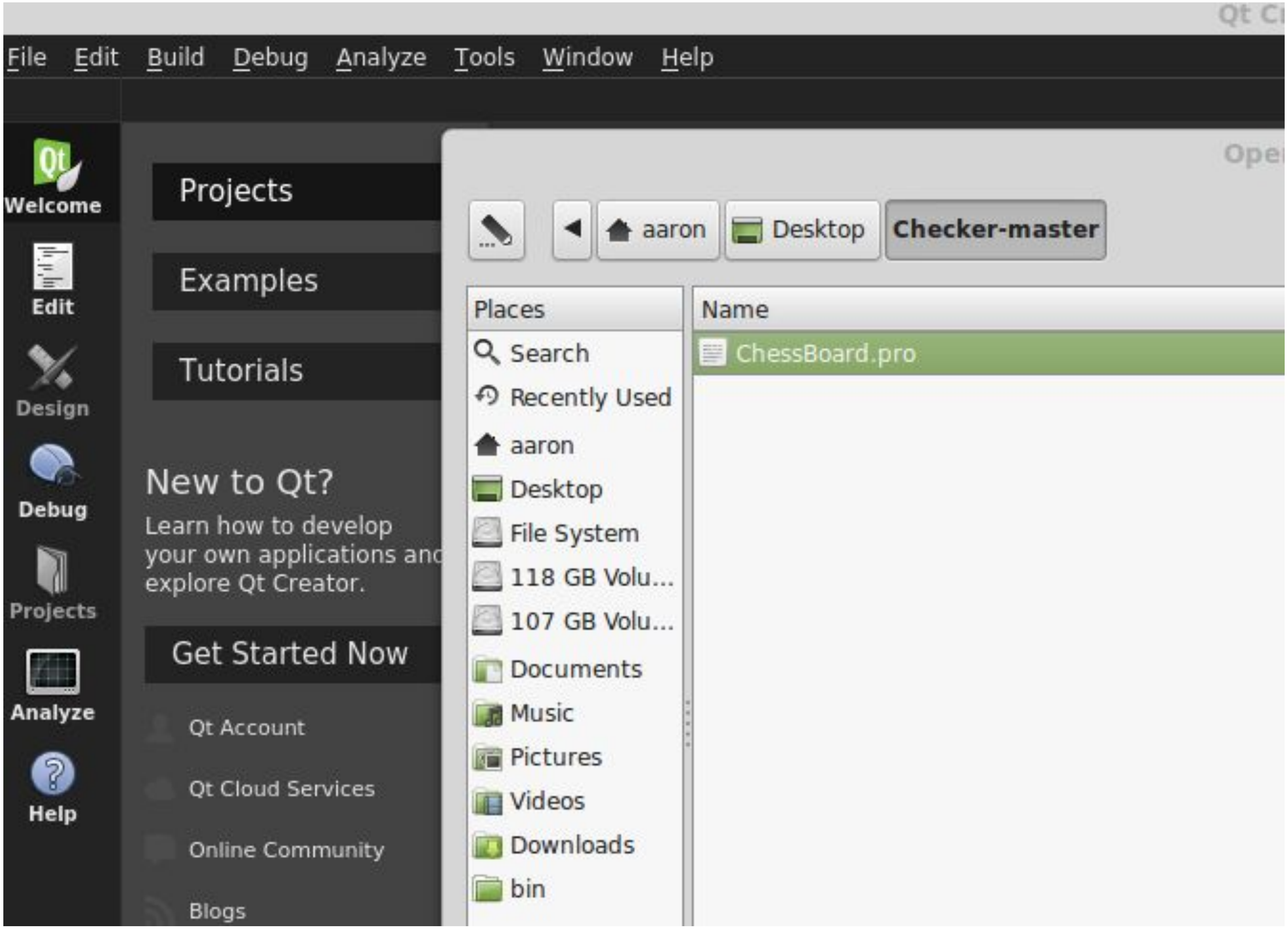
The webpage where to download the code

Step 2: unzip the code and open it with Qt creator





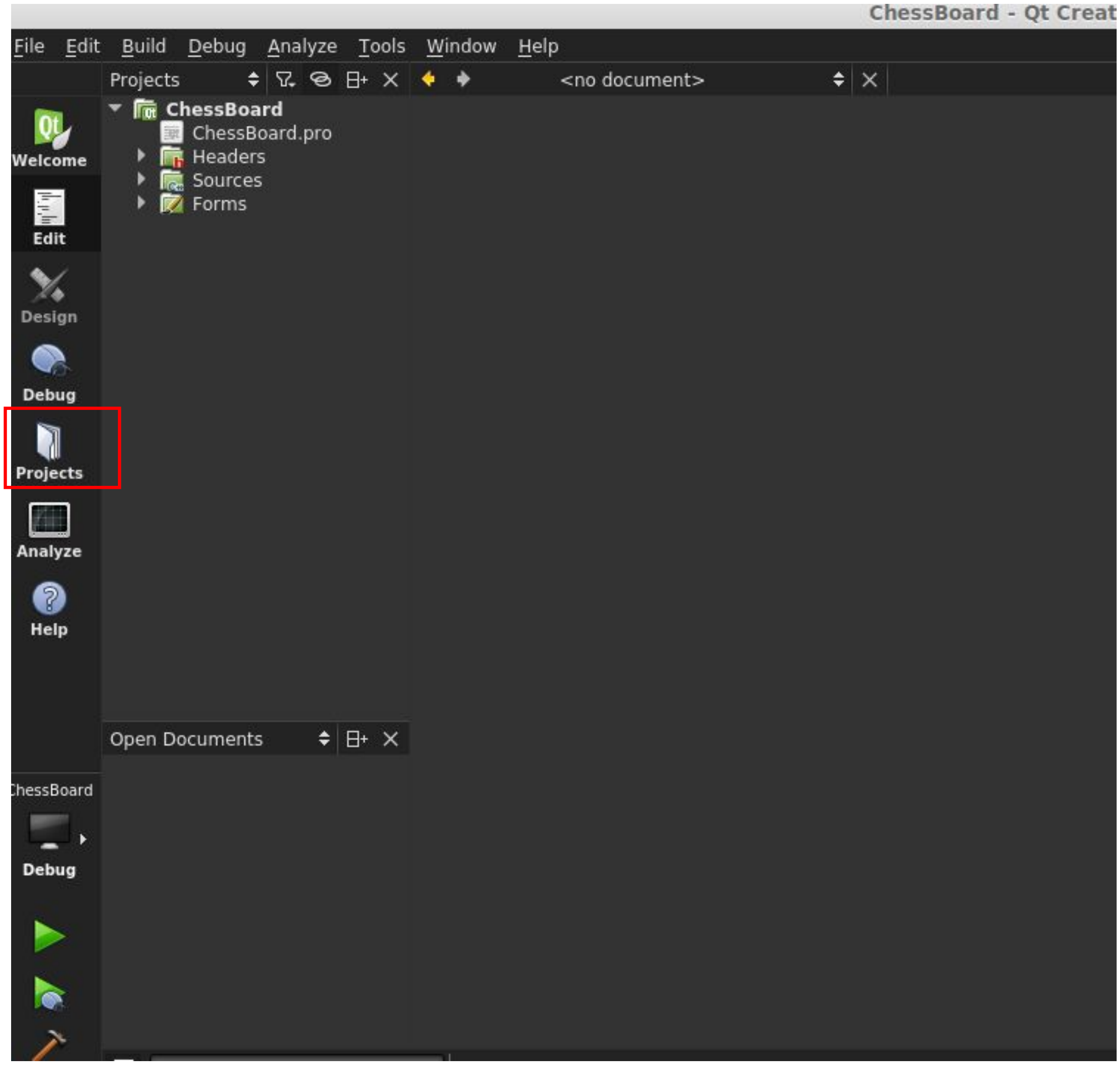
ChessBoard
~/QtWorkspace/ChessBoard/ChessBoard.pro
ChessBoard
~/ChessBoard/ChessBoard.pro
hellogl2
~/Qt/Examples/Qt-5.4/opengl/hellogl2/hellogl2.pro
openglwindow
~/Qt/Examples/Qt-5.4/gui/openglwindow/openglwindow.pro
OpenGLTest
~/workspace/OpenGLTest/OpenGLTest.pro
digitalclock
~/Qt/Examples/Qt-5.4/widgets/widgets/digitalclock/digitalclock.pro
analogclock
~/Qt/Examples/Qt-5.4/widgets/widgets/analogclock/analogclock.pro
qtconsole
~/qtconsole/qtconsole.pro
analogclock
~/Qt/Examples/Qt-5.4/gui/analogclock/analogclock.pro
HelloQt
~/HelloQt/HelloQt.pro
CaesarCipher
~/QtWorkspace/CaesarCipher/CaesarCipher.pro
untitled
~/untitled/untitled.pro



ChessBoard.pro

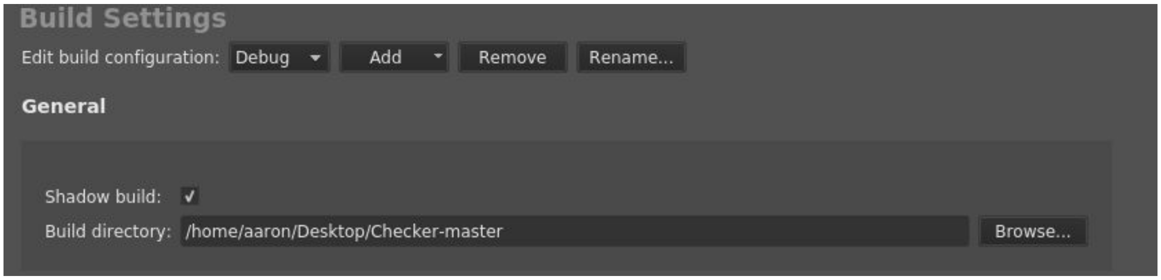
Step 3: Reset the build directory

Click Projects on the left



Choose build directory

Highly recommend to choose the directory just the same folder of the code, which could avoid some strange errors


Build Settings

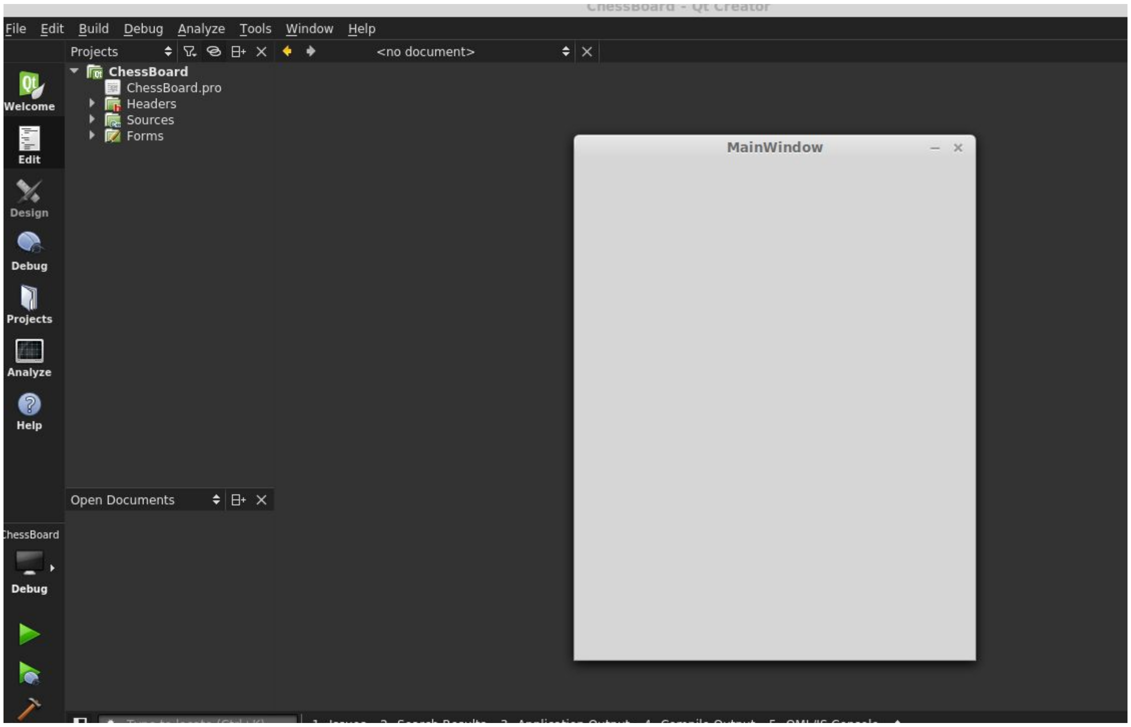
Edit build configuration: Debug

General

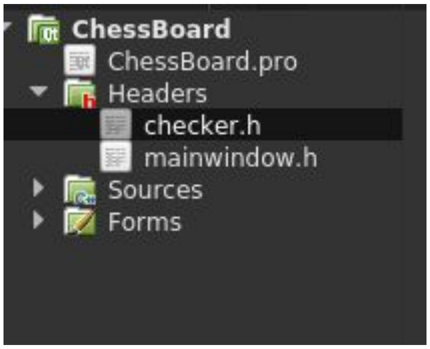
Shadow build: ☑

Build directory: /home/aaron/Desktop/Checker-master

Now you can click the run button to check the settings and you should get a blank window like this.



Then find the checker.h file as the right picture shows. All your code should be written in this file.



If all things goes well, it’s time for you to make your own checkers game program. In the next section I will give you some hints useful for the program.

Some useful hints

In checker.h file, you will find three empty functions for you to finish.

void iniGame();

This function is for game initialization. Any thing that needed to be done before the game start should be done in this function.

void Draw(QPaintDevice *device);

This functino is responsible for drawing all elements of the checkers game including the board(consists of light and dark squares) and the pieces(light and dark circle).

void GameLogic(QMouseEvent *ev, QWidget *window);

This function will keep listening and responds when user click mouse inside the window. You could get the position where user clicked the mouse by the next code:

```
    ev->x();
    ev->y();
```

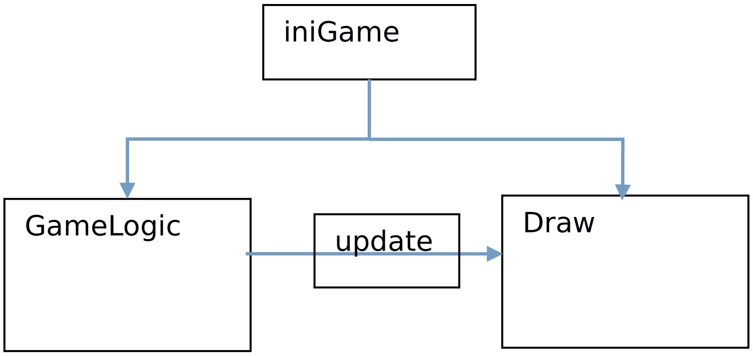
And you will get the x and y coordinate. The origin of the coordinate is the left top of the window.

function.Remember that the draw function runs only once a time and When user move the piece, you need to re-draw the window, so You should call update() as the following code **at the end of this GameLogic function**.

```
    window->update();
```

This is important. Without this anything you changed will not be displayed on the window.

All three functions will be called automatically as the following flow chart shows.

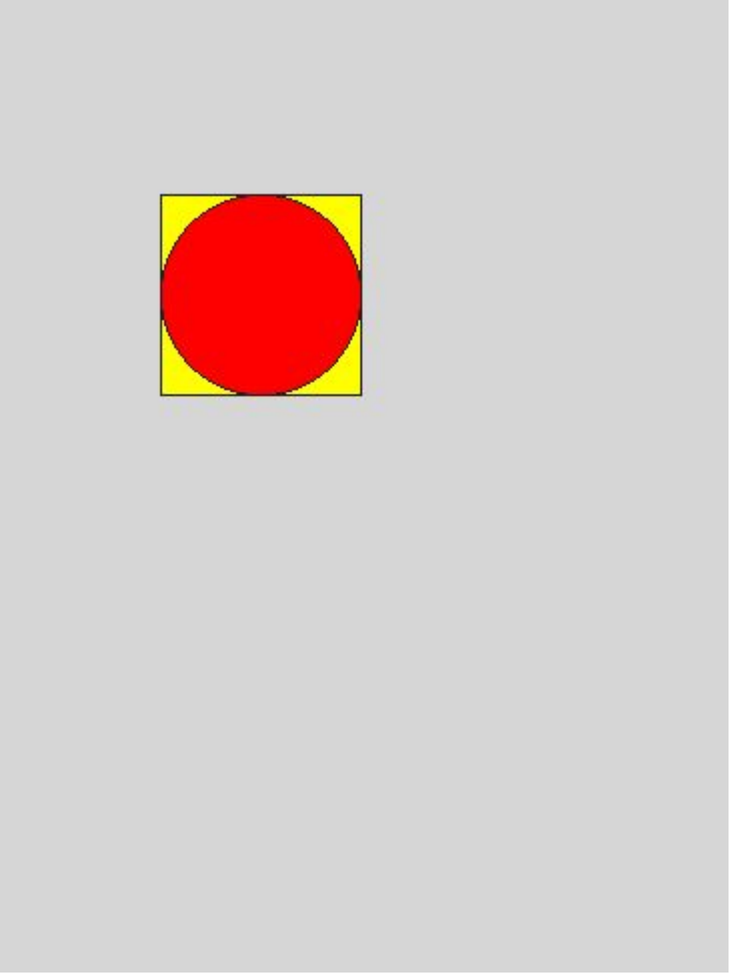


How to paint?

To paint the shape you needed, you need to use a type of QPainter as following codes.

```
  QPainter painter(device);
 painter.setBrush(Qt::yellow);
 painter.drawRect(100,100,100,100);
 painter.setBrush(Qt::red);
 painter.drawEllipse(100,100,100,100);
```

Copy and paste in the function draw and click run, you will get this picture:



‘painter.setBrush(Qt::yellow)’ means the the painter will draw the next shape by fill it with the color of yellow. ‘painter.drawRect(100,100,100,100)’ means the painter will draw a rectangle in the **position** of (100,100)(1st and 2nd parameter), and the **width** of rectangle is 100(3rd parameter) pixel and the **height** is 100(4th parameter) pixel.

‘painter.drawEllipse(100,100,100,100)’ means draw a ellipse just inside the rectangle in the position of (100,100)(1st and 2nd parameter) with width of 100(3rd parameter) and height of 100(4th parameter).

In this project I provide you a fixed size window with 400 width and 500 height. Just everything needed in the game on this window.

The logic of the game

To implement the game, you should be clear about the procedure. Here I give you some hints.

1 Use a 8*8 int array to represents the pieces on the board, you could use 0 to represent no piece and 1 to represent a light piece and 2 to represent a dark piece.

And the whole game process could be divided into at least 6 status follows:

DARK_TURN

DARK_MOVE

LIGHT_TURN

LIGHT_MOVE

DARK_WIN

LIGHT_WIN

You should write different responds for user's mouse click according to different current status.

The project could be a little challenge for you but try your best to make it. I'm looking forward to playing the checker game from you.