



**Code less.  
Create more.  
Deploy everywhere.**

## A Simple Floorplan Painter Using Qt-C++

---

# Outline

- What is Qt?
- Downloading Qt
- Qt Installation
- New Qt Project
- Paint

# What is Qt?

- Qt (官方發音同cute)，是一個跨平台的 C++ 應用程式開發框架，廣泛用於開發 GUI 程式
- Qt 擁有完善的 C++ 圖形函式庫
- 支援多種平台。會自動依平台的不同，表現平台特有的圖形介面風格
- 目前最新版本為 5.8.0
- Windows/Linux/MacOS 皆可用

source: Wikipedia

# Downloading Qt (Option I)

- URL: <http://www.qt.io/download-open-source/>
- Online installer (smaller size)

**Your download**

We detected your operating system as: macOS  
Recommended download: Qt Online Installer for macOS

Before you begin your download, please make sure you:

- › learn about the [obligations of the LGPL](#).
- › read the [FAQ](#) about developing with the LGPL.

[Download Now](#)

Qt online installer is a small executable which downloads content over internet based on your selections. It provides all Qt 5.x binary & source packages and latest Qt Creator.

For more information visit our [Developers page](#).  
Not the download package you need? [Hide All Downloads](#)

# Downloading Qt (Option II)

- URL: <http://www.qt.io/download-open-source/>
- Full offline installer (larger size)
- Use this one to demo

## Your download

We detected your operating system as: macOS  
Recommended download: Qt Online Installer for macOS

Before you begin your download, please make sure you:

- › learn about the [obligations of the LGPL](#).
- › read the [FAQ](#) about developing with the LGPL.

[Download Now](#)

Qt online installer is a small executable which downloads content over internet based on your selections. It provides all Qt 5.x binary & source packages and latest Qt Creator.

For more information visit our [Developers page](#).

Not the download package you need? [View All Downloads](#)

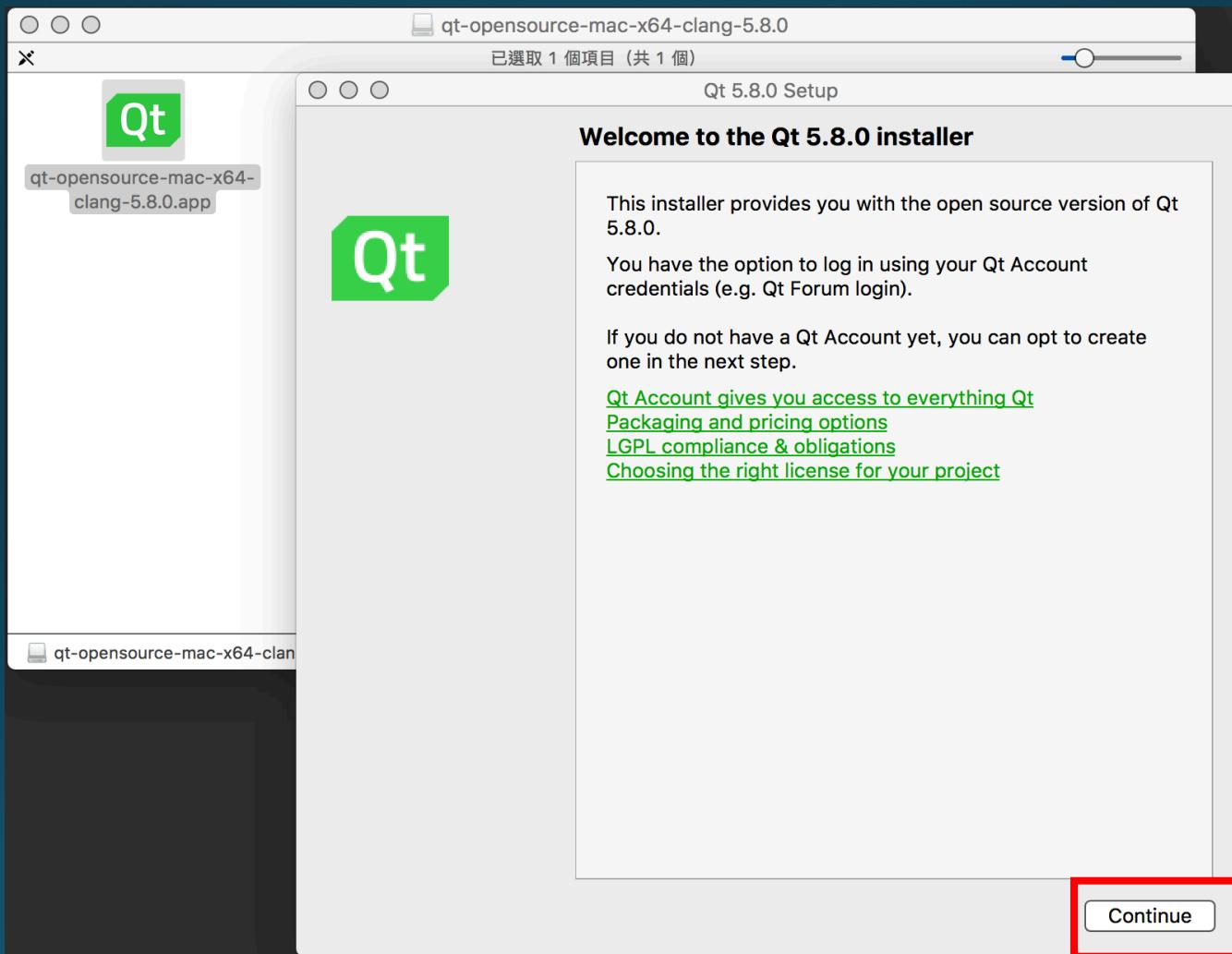
### macOS Host

- › Qt 5.8.0 for macOS (1.2 GB) [\(info\)](#)
- › Qt 5.8.0 for Android (armv7, x86) (1.4 GB) [\(info\)](#)
- › Qt 5.8.0 for Android (armv7, x86) and iOS (3.4 GB) [\(info\)](#)

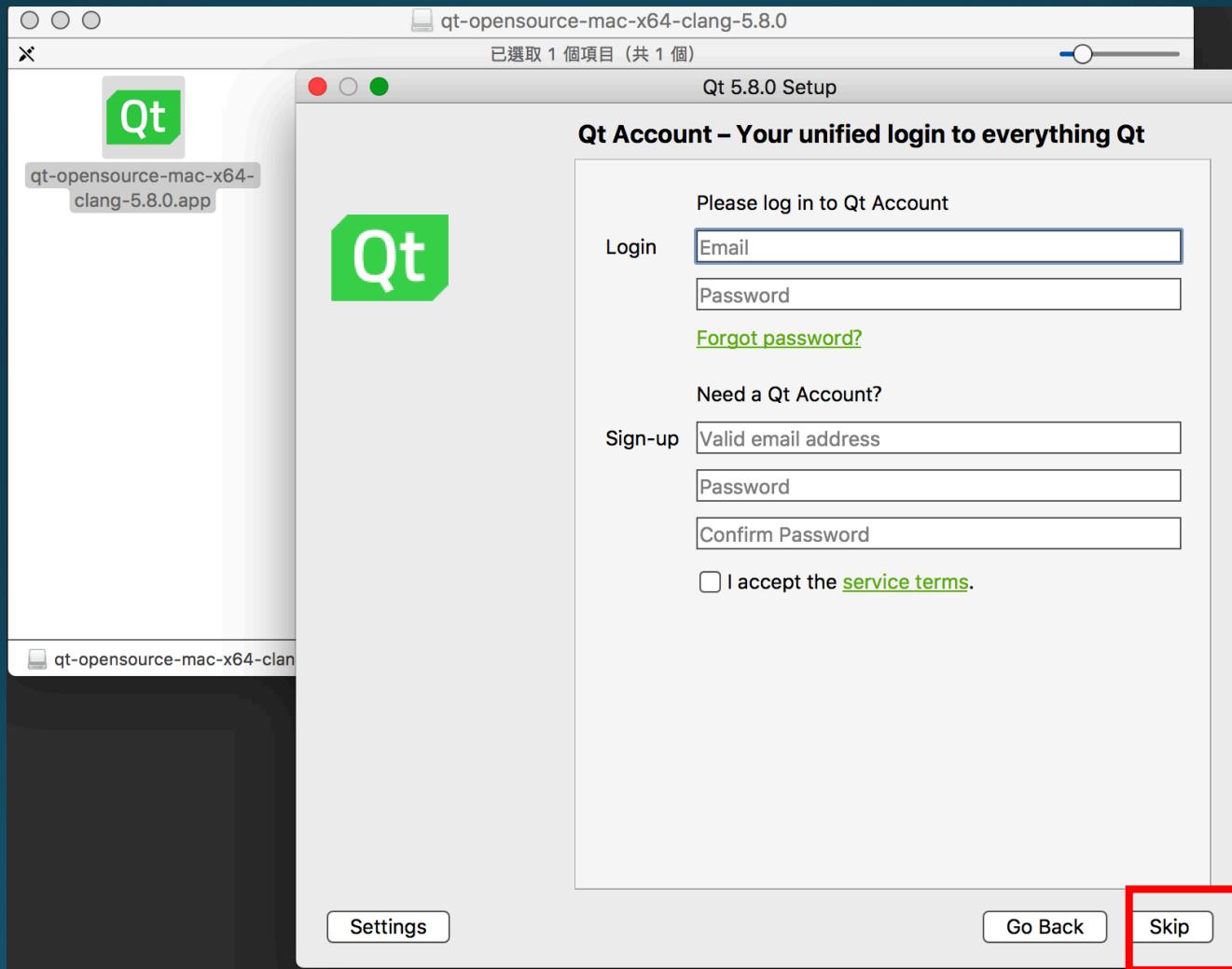
### Windows Host

- › Qt 5.8.0 for Windows 64-bit (VS 2015, 1.0 GB) [\(info\)](#)
- › Qt 5.8.0 for Windows 32-bit (VS 2015, 1.0 GB) [\(info\)](#)
- › Qt 5.8.0 for Windows 64-bit (VS 2013, 958 MB) [\(info\)](#)
- › Qt 5.8.0 for Windows 32-bit (VS 2013, 947 MB) [\(info\)](#)
- › Qt 5.8.0 for Windows 32-bit (MinGW 5.3.0, 1.2 GB) [\(info\)](#)
- › Qt 5.8.0 for Android (Windows 32-bit, 1.3 GB) [\(info\)](#)
- › Qt 5.8.0 for WinRT 32-bit (VS 2013, 1.2 GB) [\(info\)](#)
- › Qt 5.8.0 for WinRT 32-bit (VS 2015, 1.2 GB) [\(info\)](#)

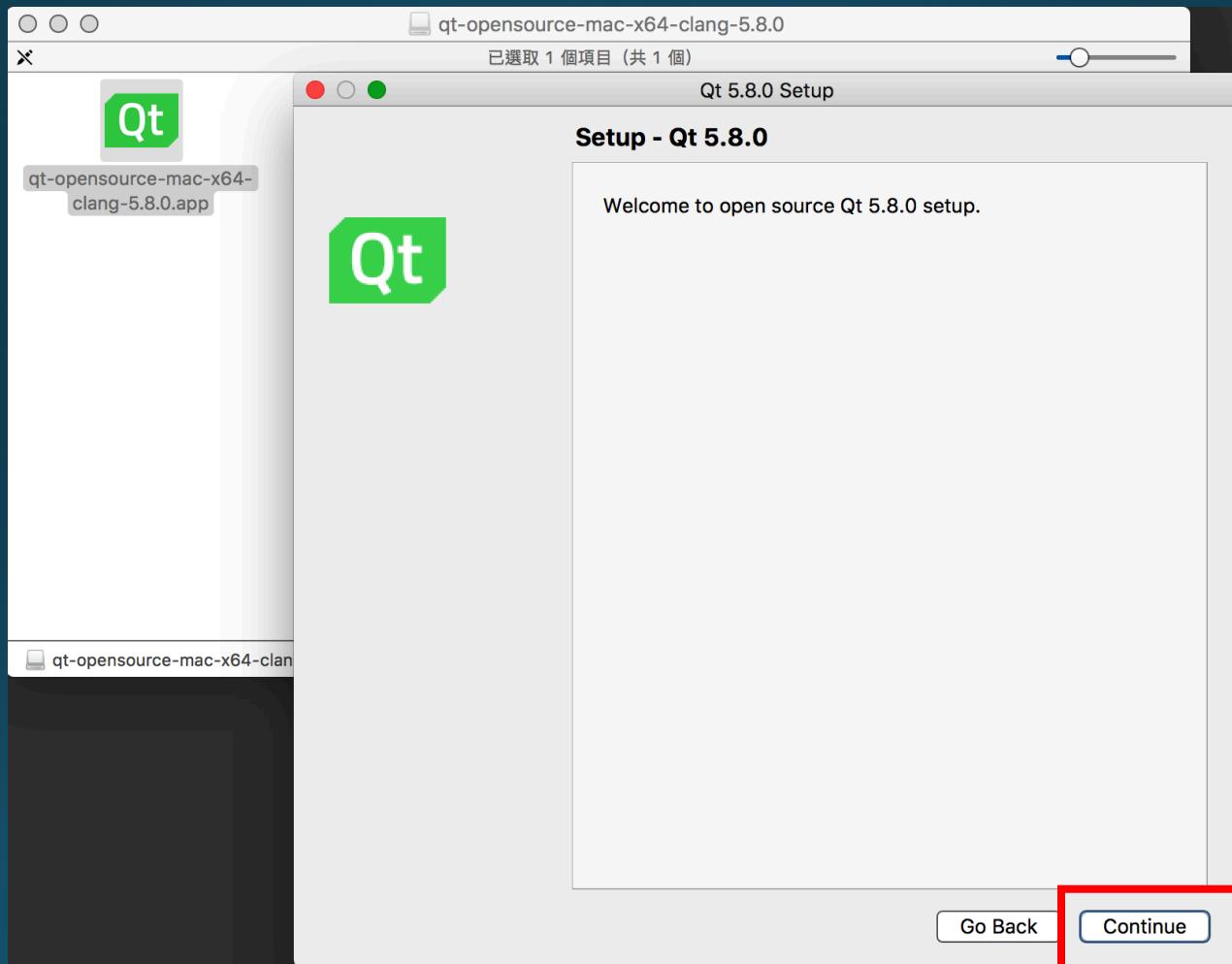
# Qt Installation (1/9)



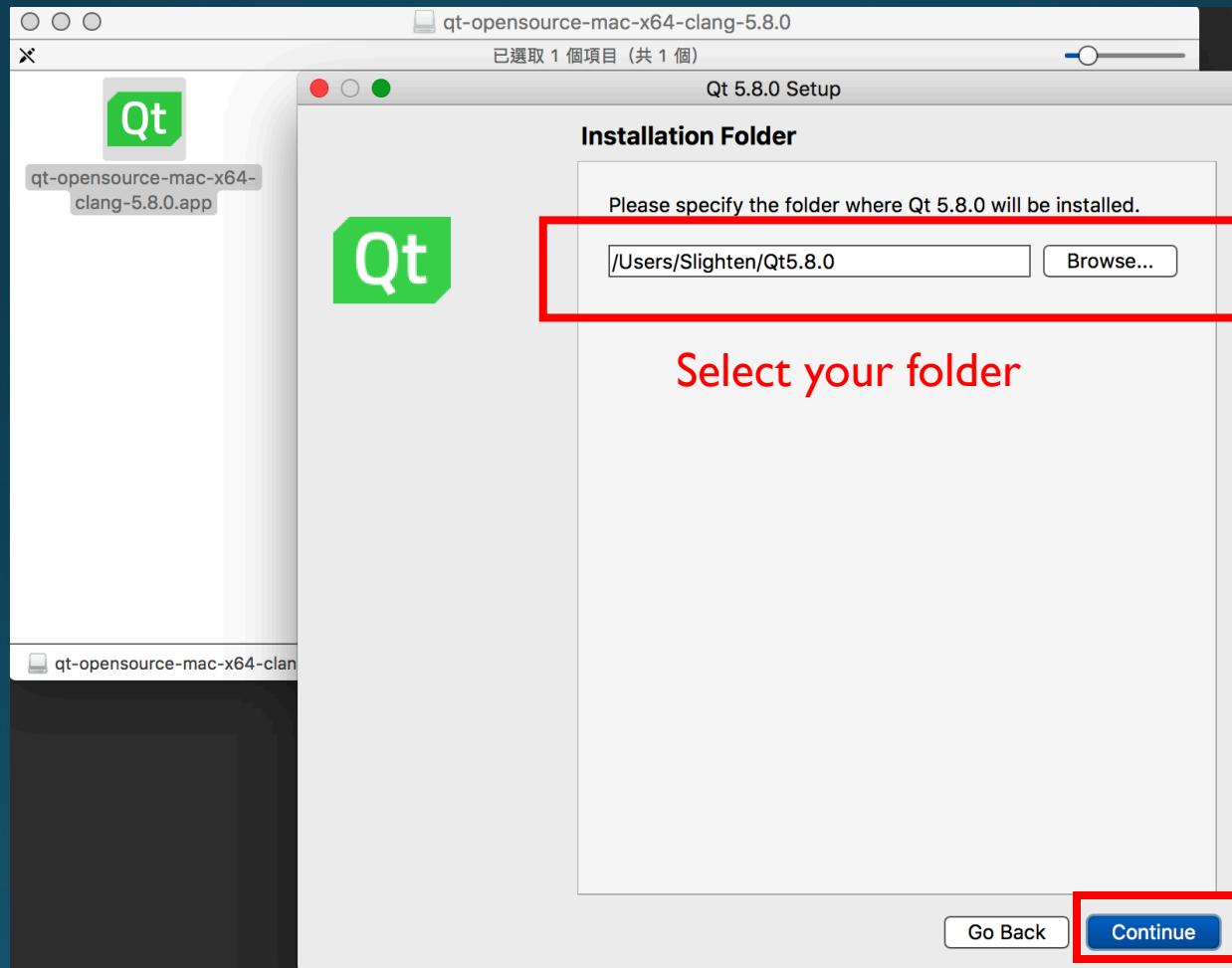
# Qt Installation (2/9)



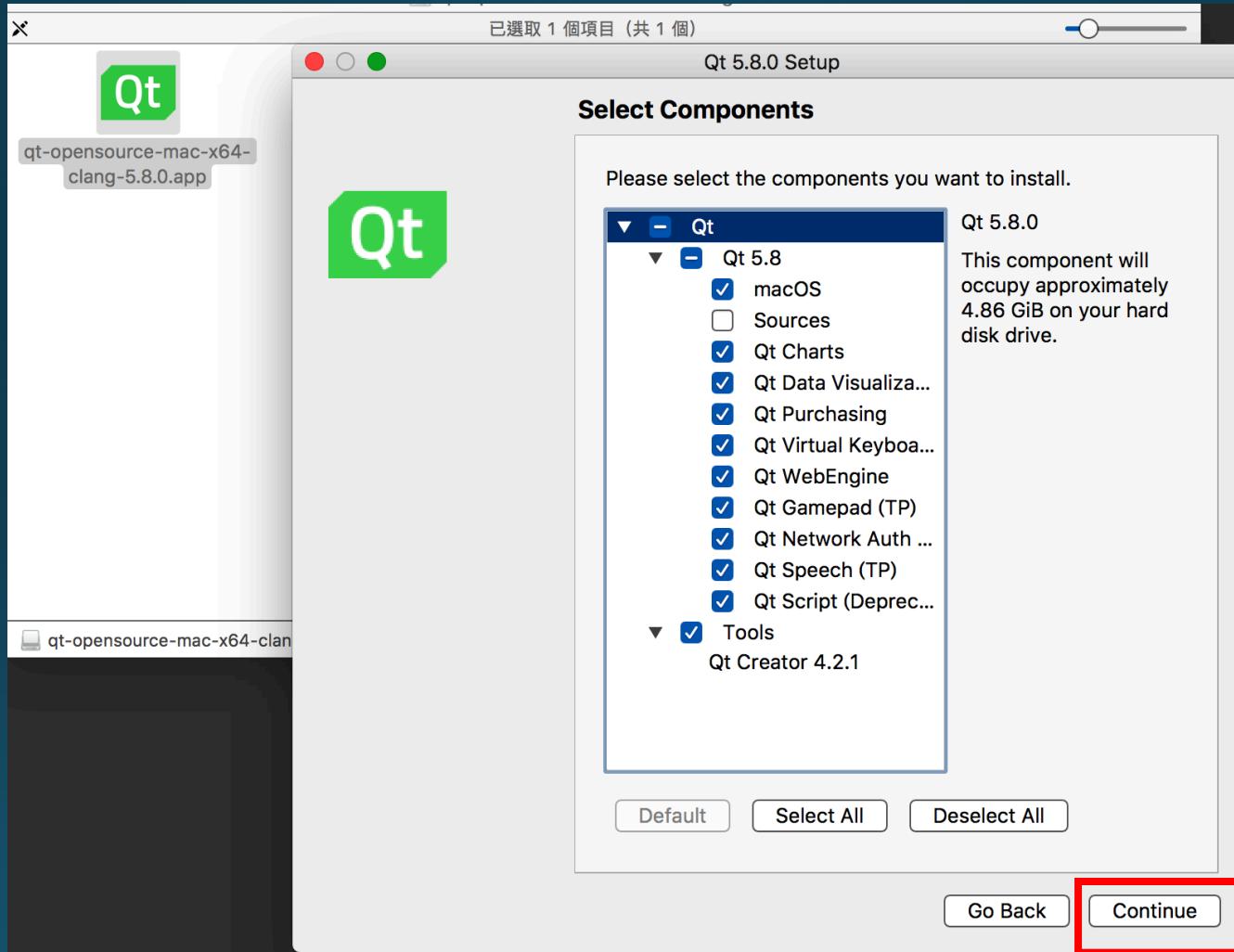
# Qt Installation (3/9)



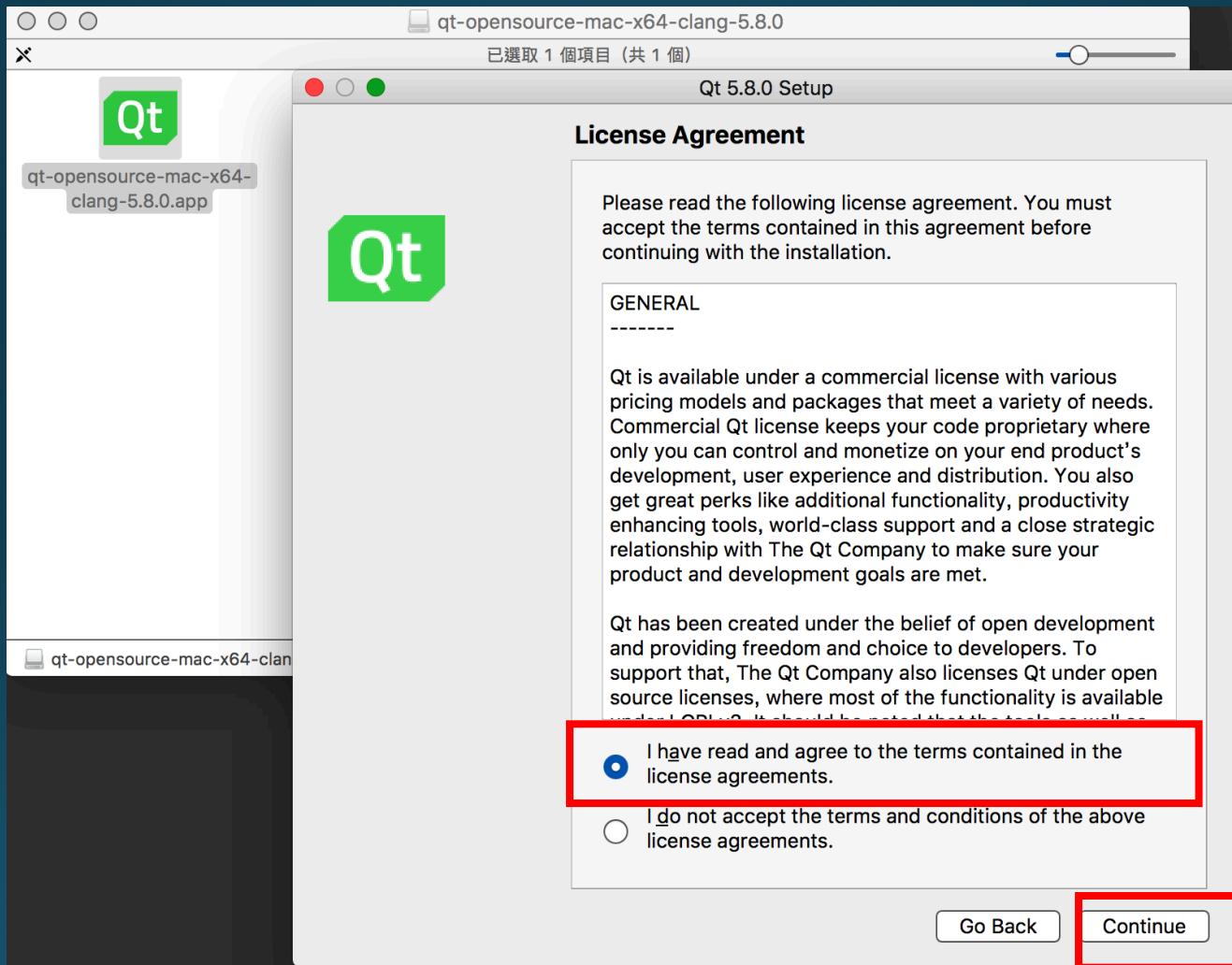
# Qt Installation (4/9)



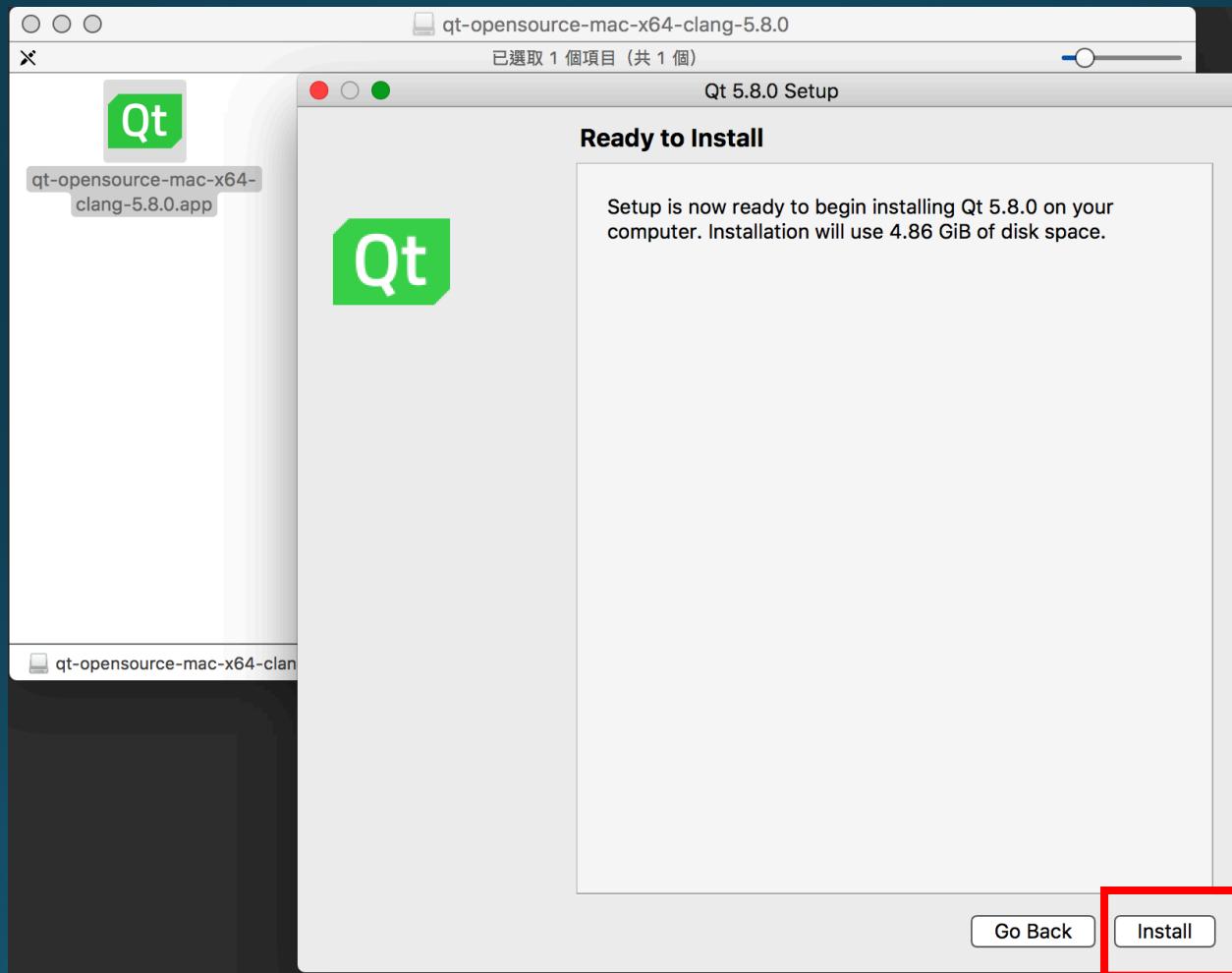
# Qt Installation (5/9)



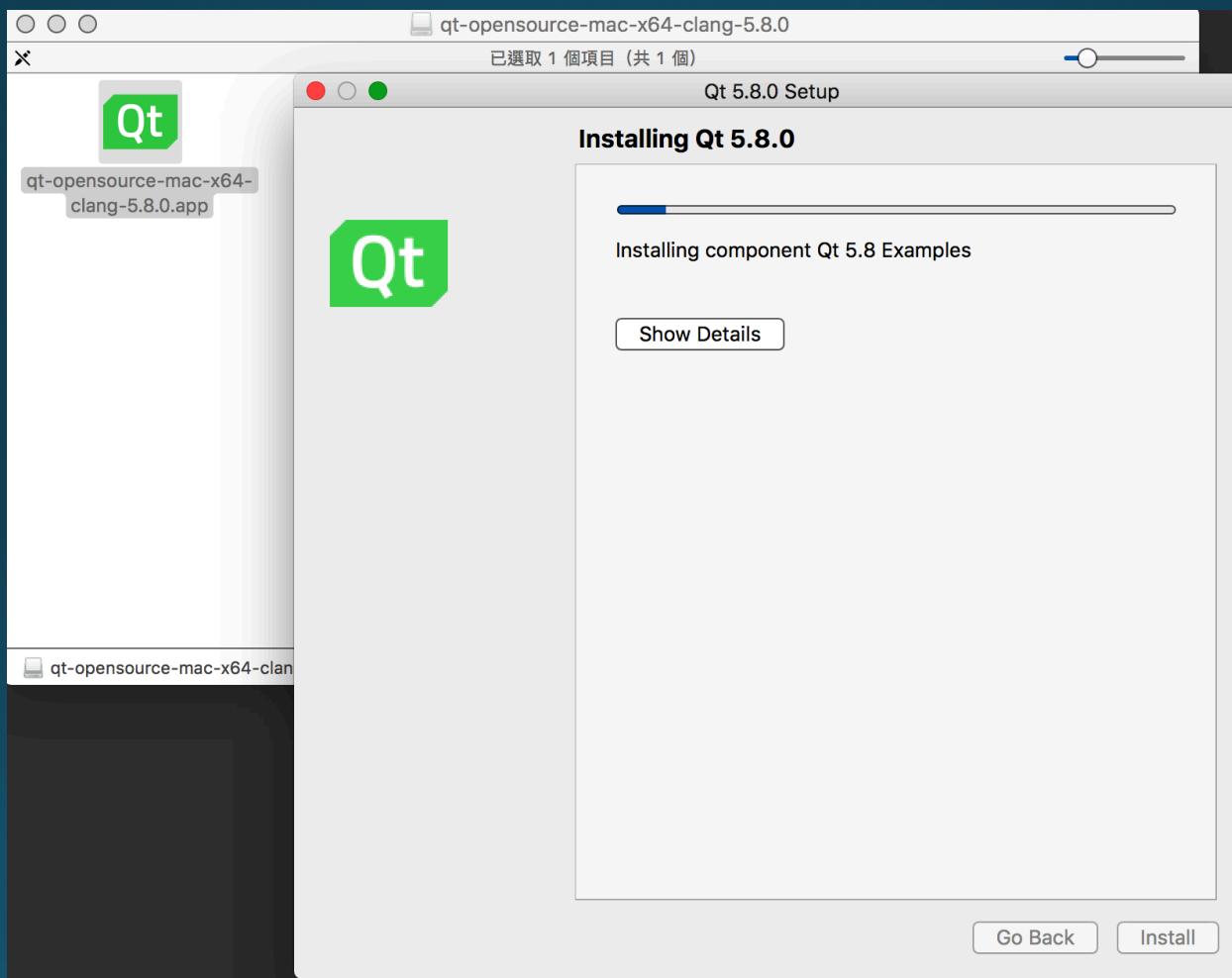
# Qt Installation (6/9)



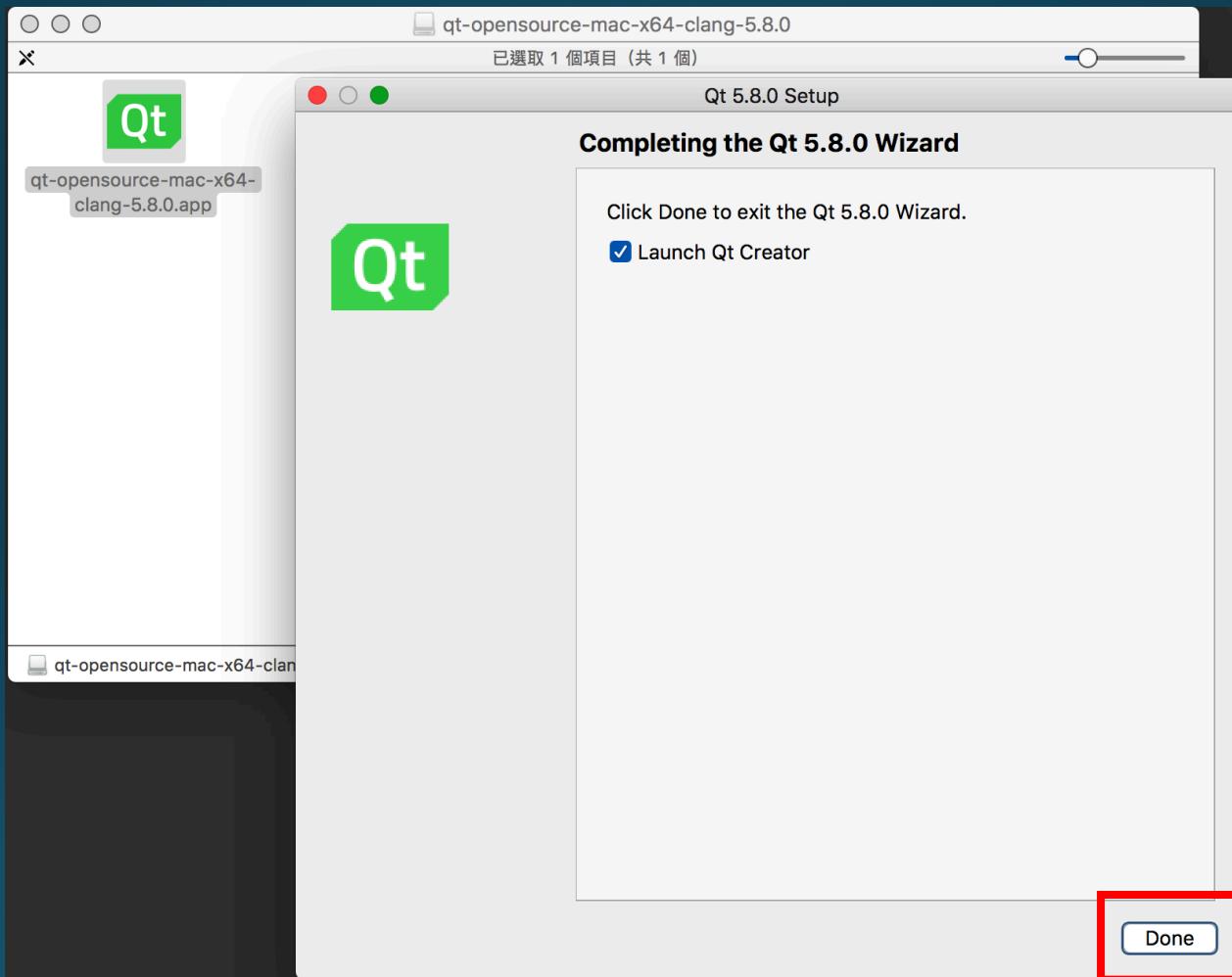
# Qt Installation (7/9)



# Qt Installation (8/9)

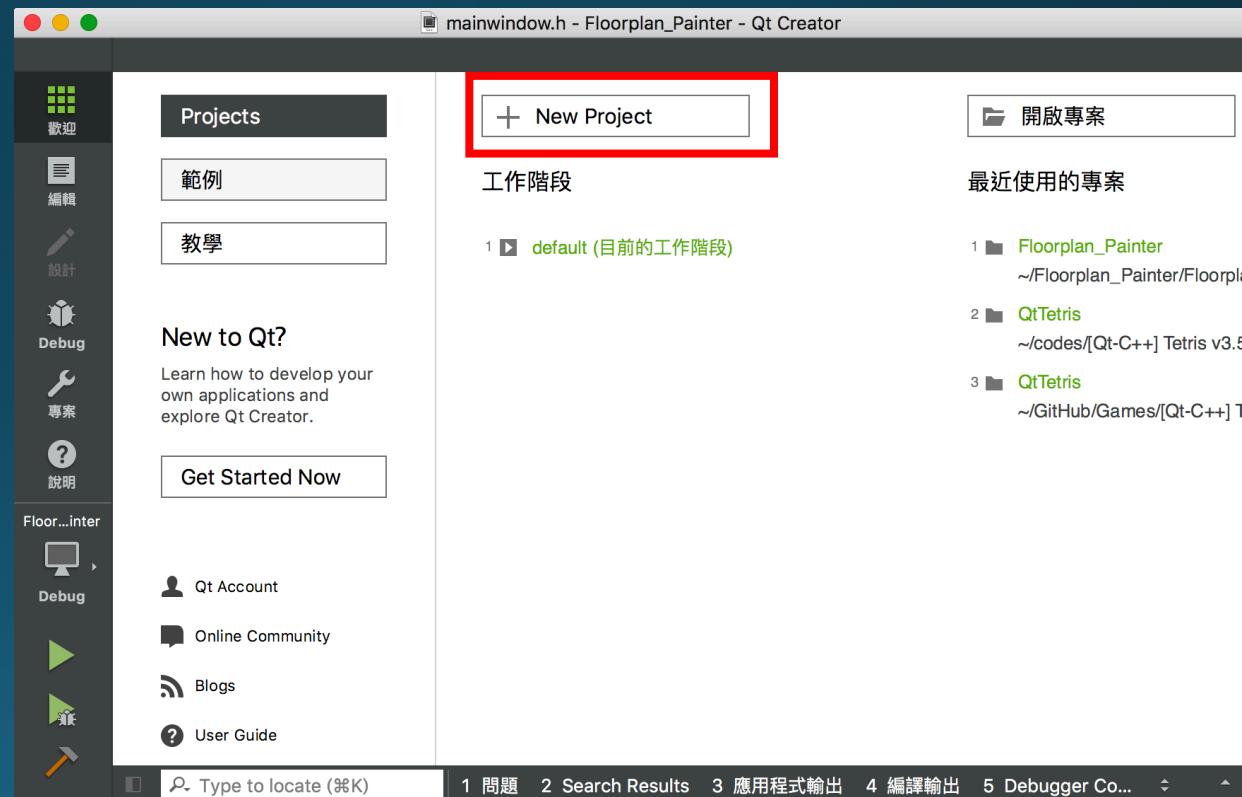


# Qt Installation (q/q)



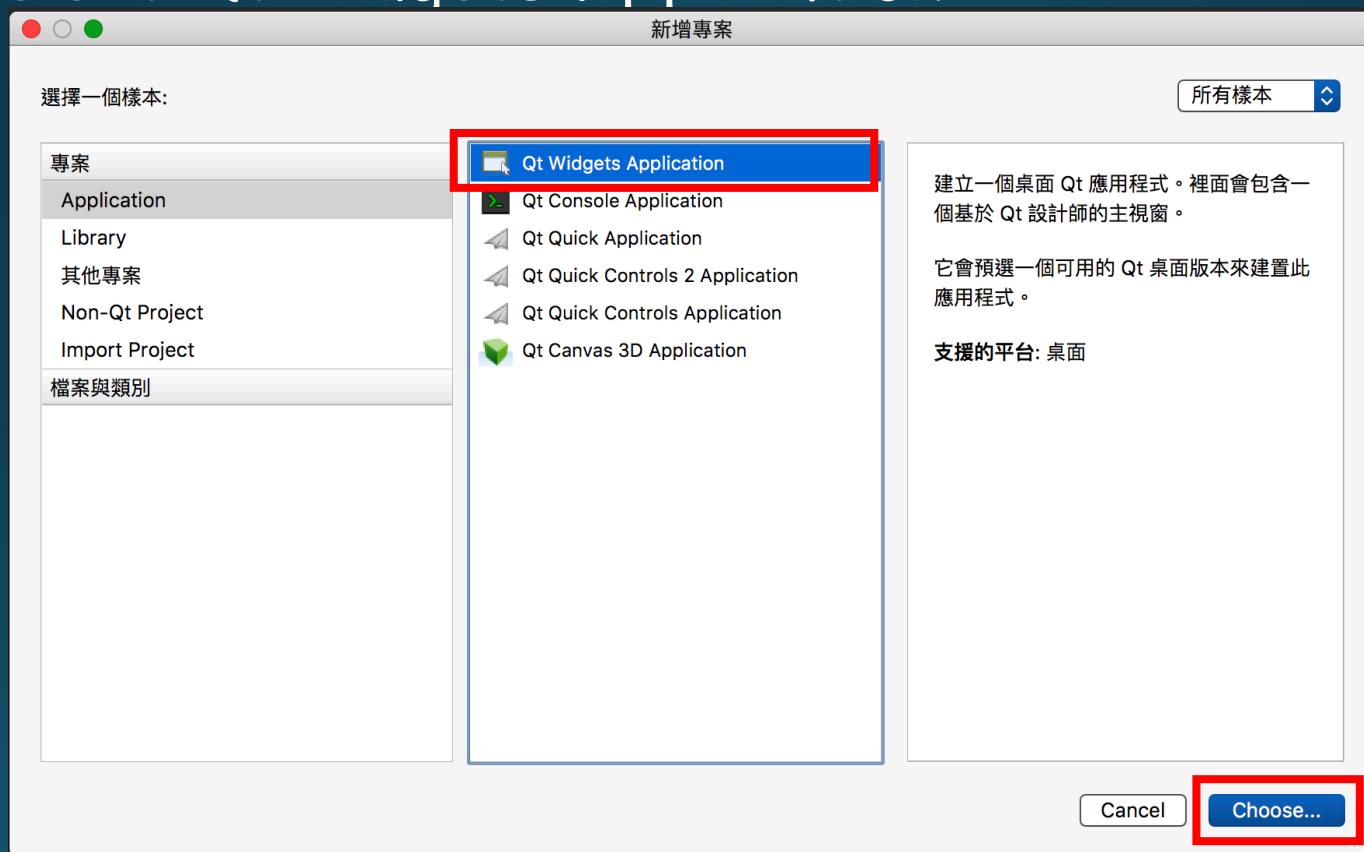
# New Qt Project

- Open Qt Creator
- New Project



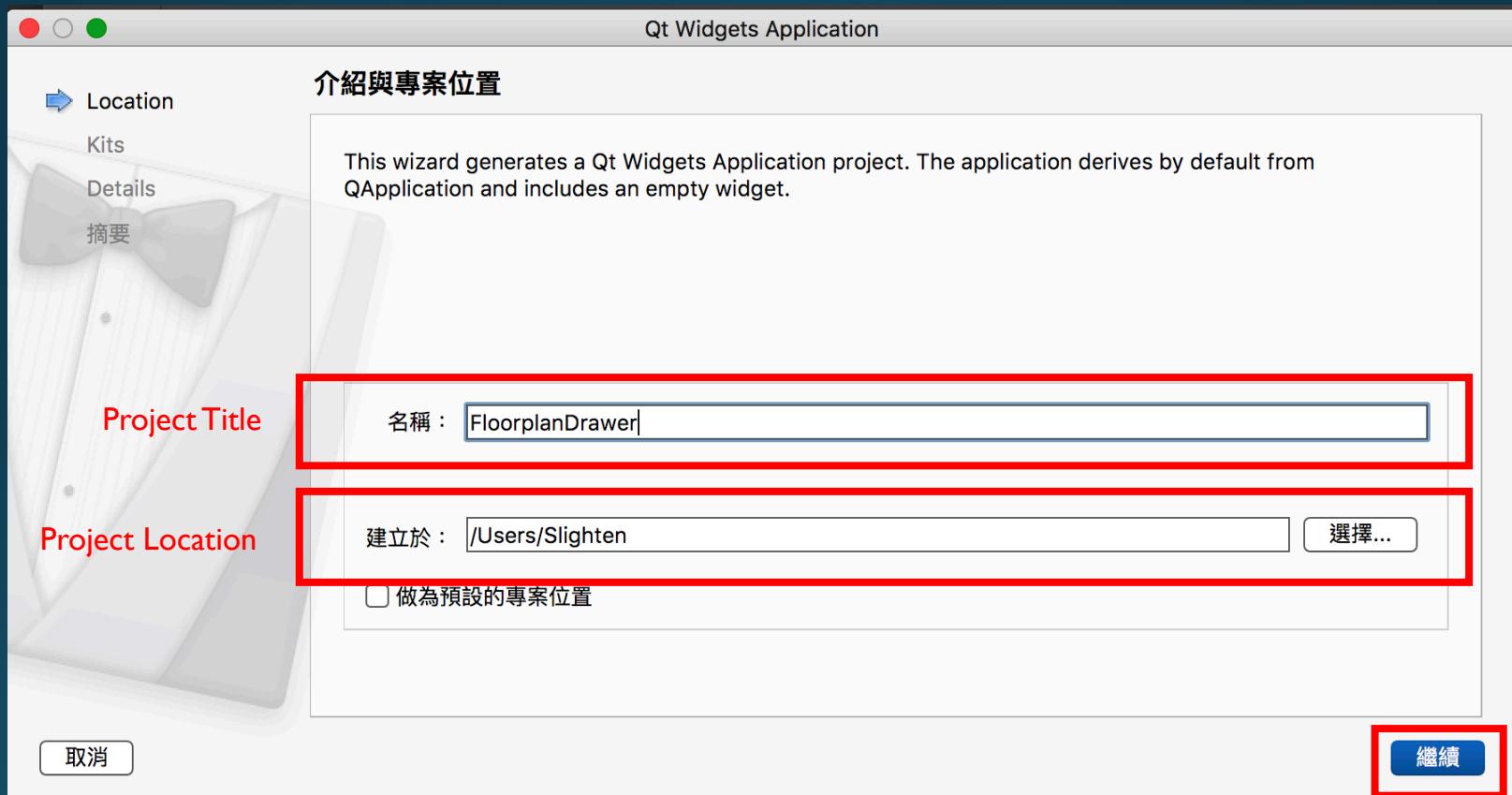
# New Qt Project

- Select Qt Widgets Application



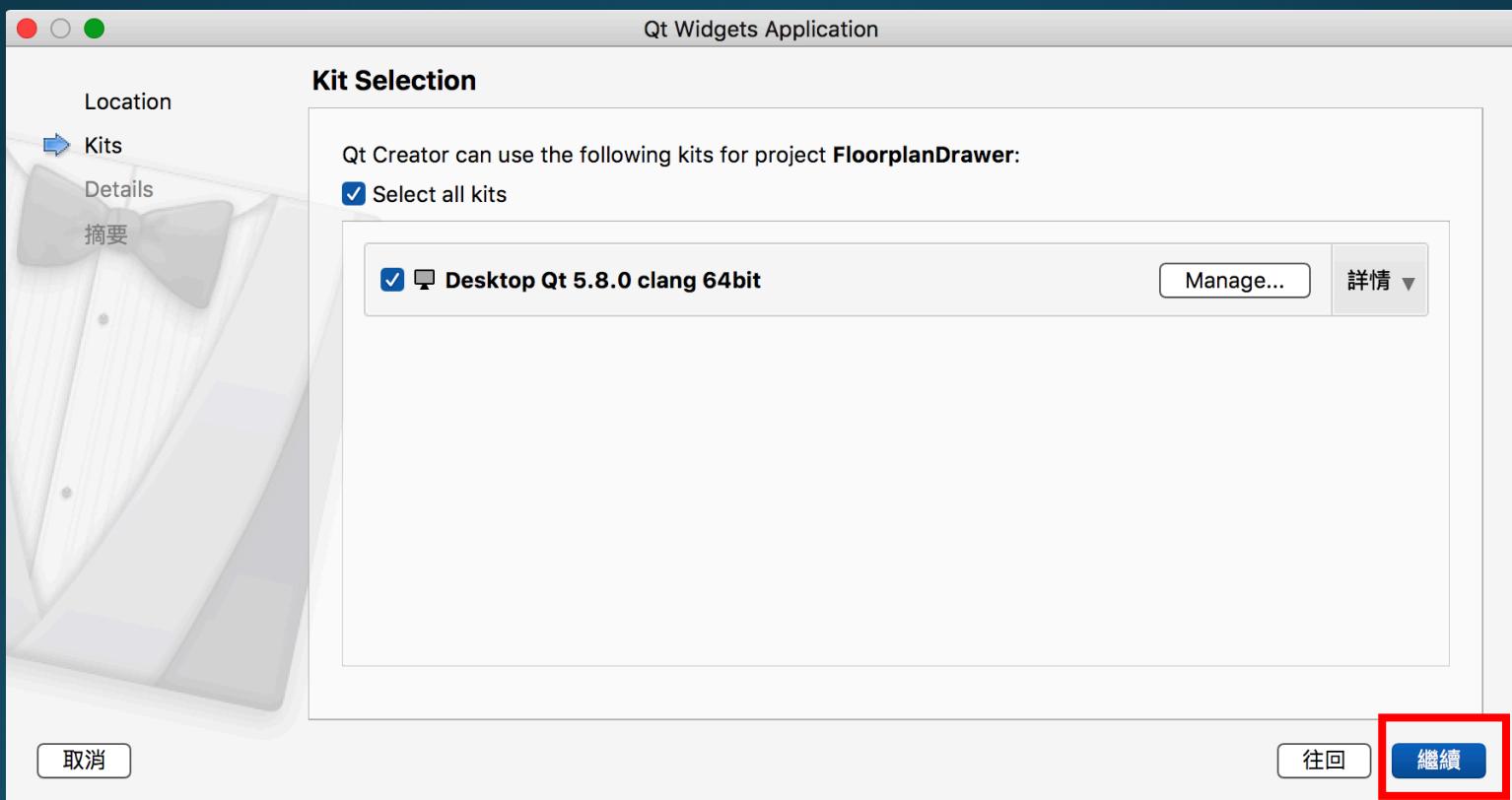
# New Qt Project

- No Chinese name and no whitespace is allowed



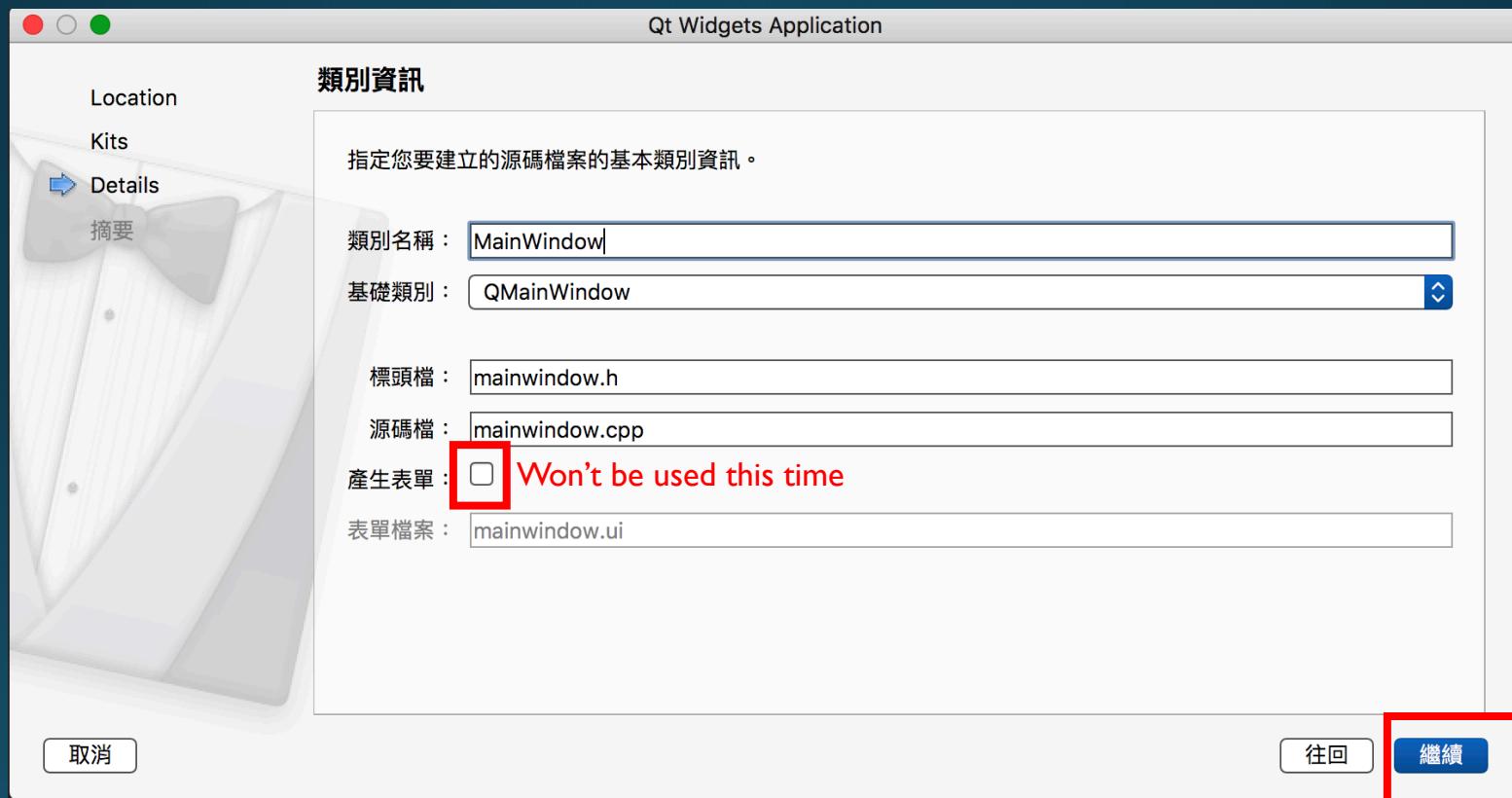
# New Qt Project

- Selecting kits



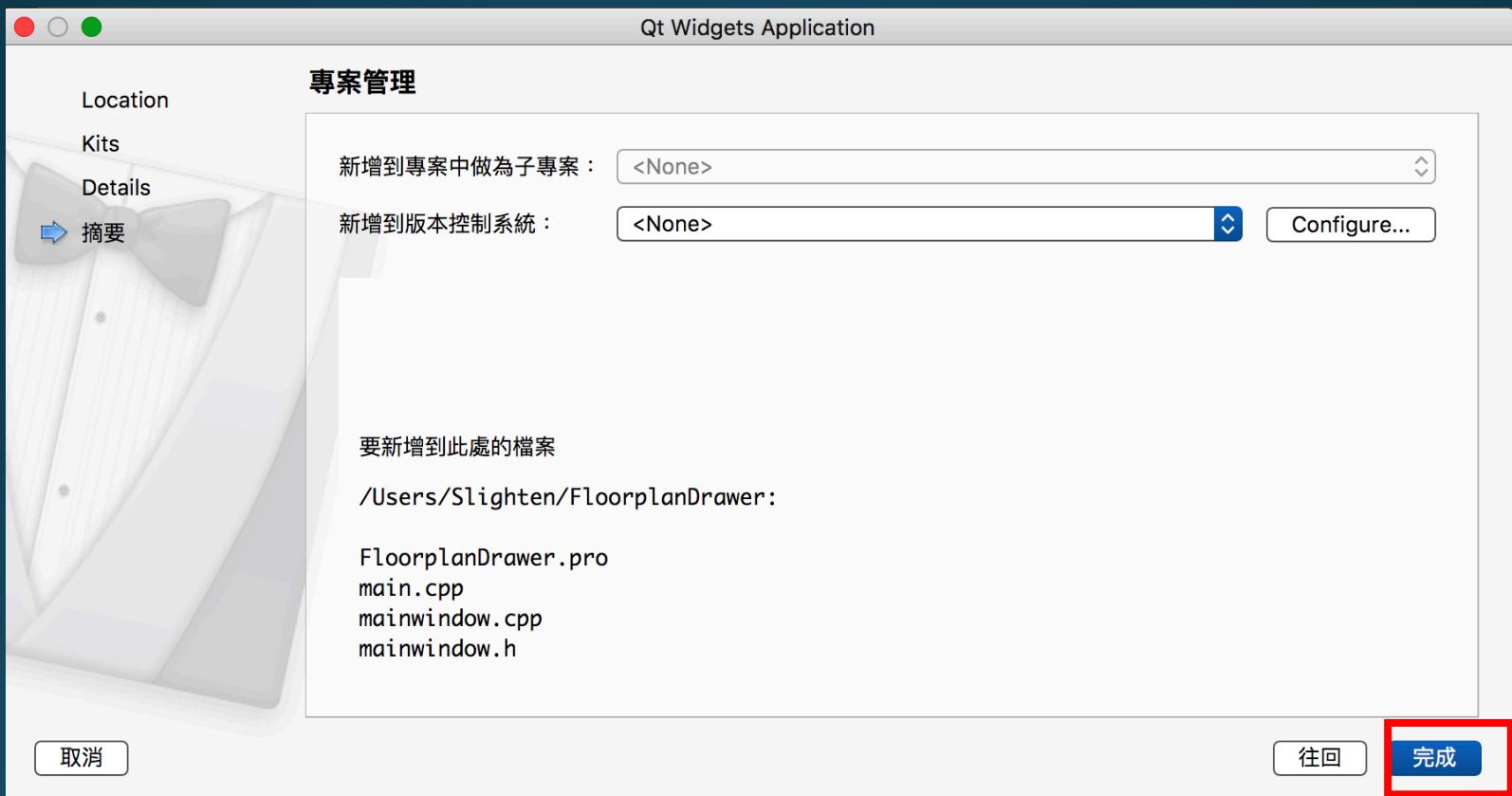
# New Qt Project

- Uncheck 產生表單



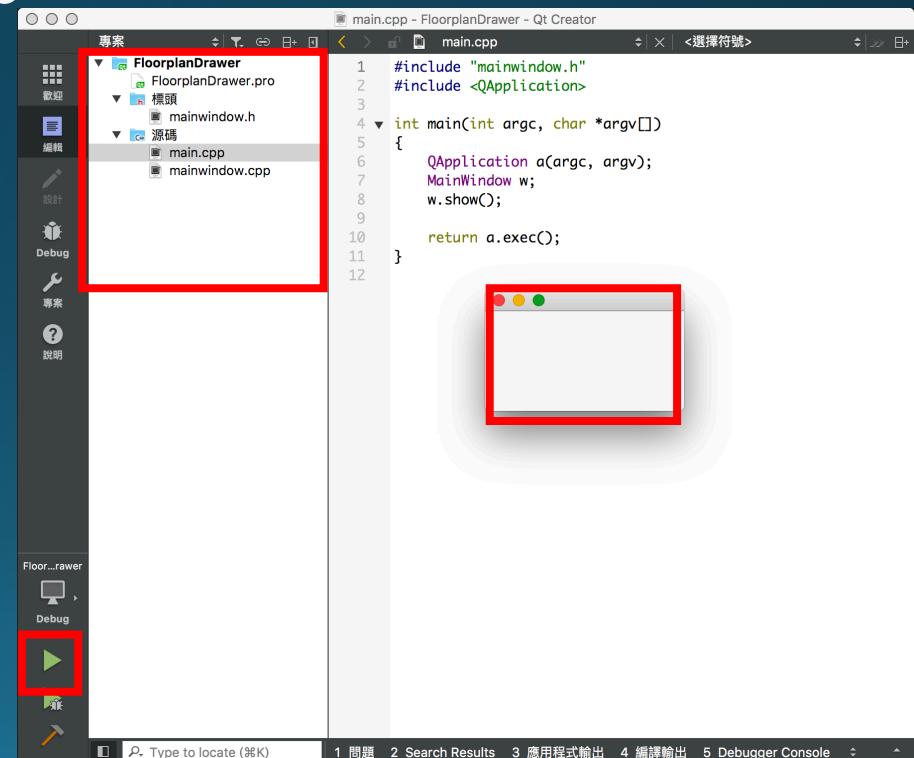
# New Qt Project

- Done



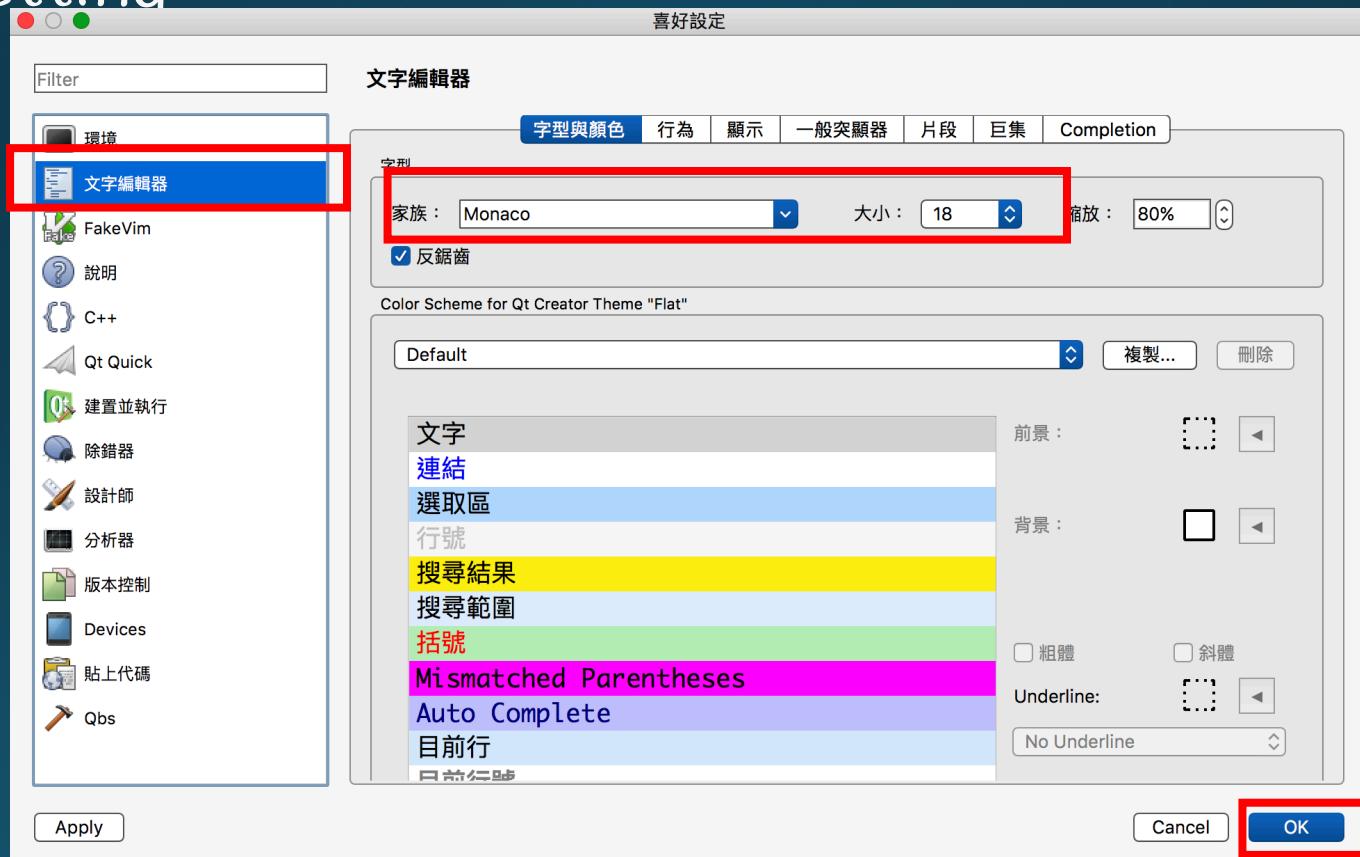
# New Qt Project

- See the default template
  - FloorplanDrawer.pro
  - mainwindow.h
  - main.cpp
  - mainwindow.cpp
- Click 執行
  - An empty window will be shown



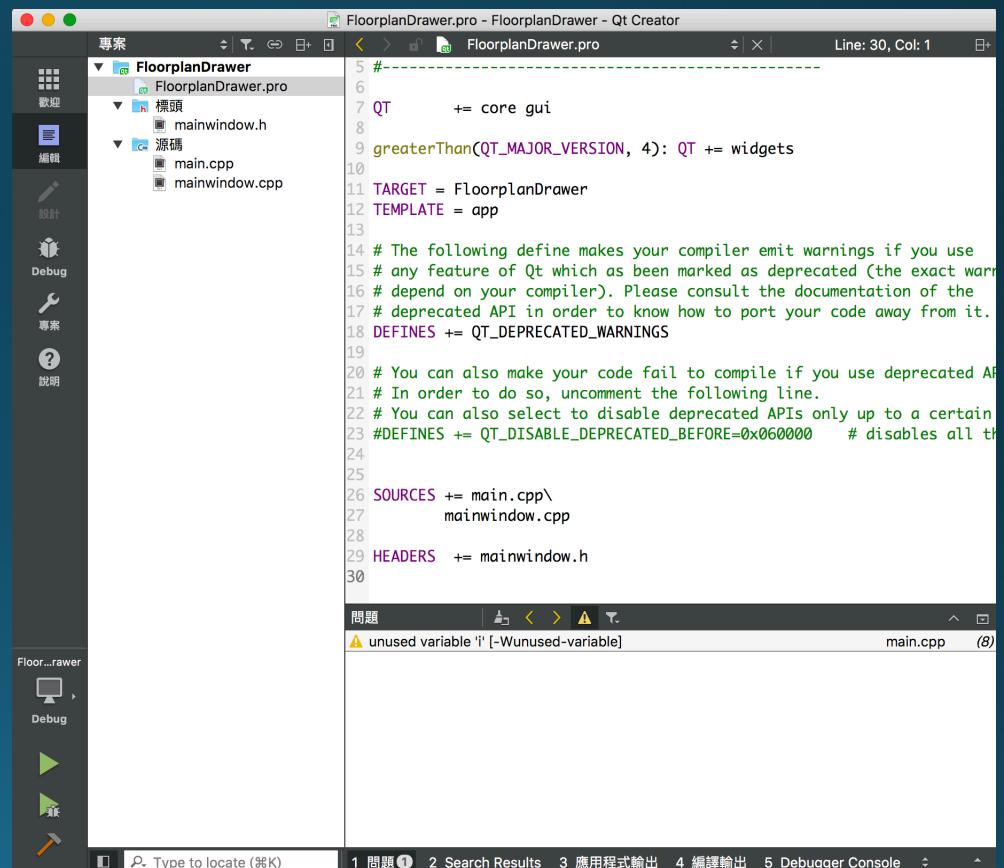
# Font Size Configuration

- Setting



# Qt Project File

- It's a Makefile for Qt



The screenshot shows the Qt Creator interface with a project named "FloorplanDrawer". The left sidebar displays project files: "FloorplanDrawer.pro", "mainwindow.h", "main.cpp", and "mainwindow.cpp". The main editor window shows the content of "FloorplanDrawer.pro":

```
FloorplanDrawer.pro - FloorplanDrawer - Qt Creator
Line: 30, Col: 1

#-
QT      += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = FloorplanDrawer
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warn
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated API
# In order to do so, uncomment the following line.
# # You can also select to disable deprecated APIs only up to a certain
# #DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the
# 

SOURCES += main.cpp\
           mainwindow.cpp

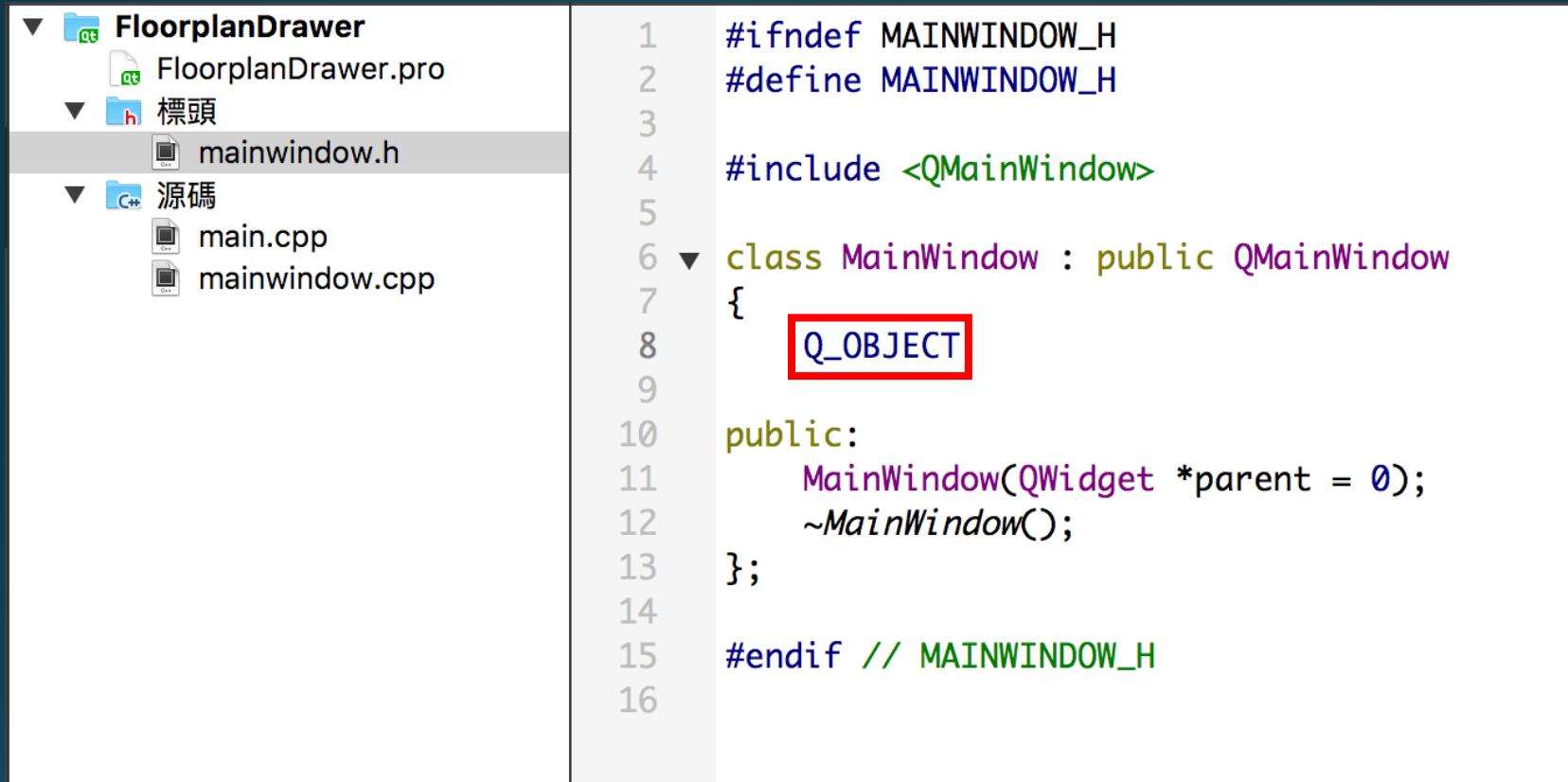
HEADERS += mainwindow.h

```

The bottom right corner of the editor shows a warning: "unused variable 'l'" [-Wunused-variable] in "main.cpp" at line 8.

The bottom navigation bar includes tabs for "問題 1", "Search Results 2", "應用程式輸出 3", "編譯輸出 4", and "Debugger Console 5".

# Overview to mainwindow.h



The image shows the Qt Creator interface. On the left, the project tree displays a folder named 'FloorplanDrawer' containing a 'FloorplanDrawer.pro' file, a 'mainwindow.h' file highlighted in grey, and a '源碼' (Source Code) folder with 'main.cpp' and 'mainwindow.cpp' files. On the right, the code editor shows the contents of 'mainwindow.h'. The code is as follows:

```
1 ifndef MAINWINDOW_H
2 define MAINWINDOW_H
3
4 include <QMainWindow>
5
6 class MainWindow : public QMainWindow
7 {
8     Q_OBJECT
9
10 public:
11     MainWindow(QWidget *parent = 0);
12     ~MainWindow();
13 }
14
15 endif // MAINWINDOW_H
16
```

The 'Q\_OBJECT' macro at line 8 is highlighted with a red box.

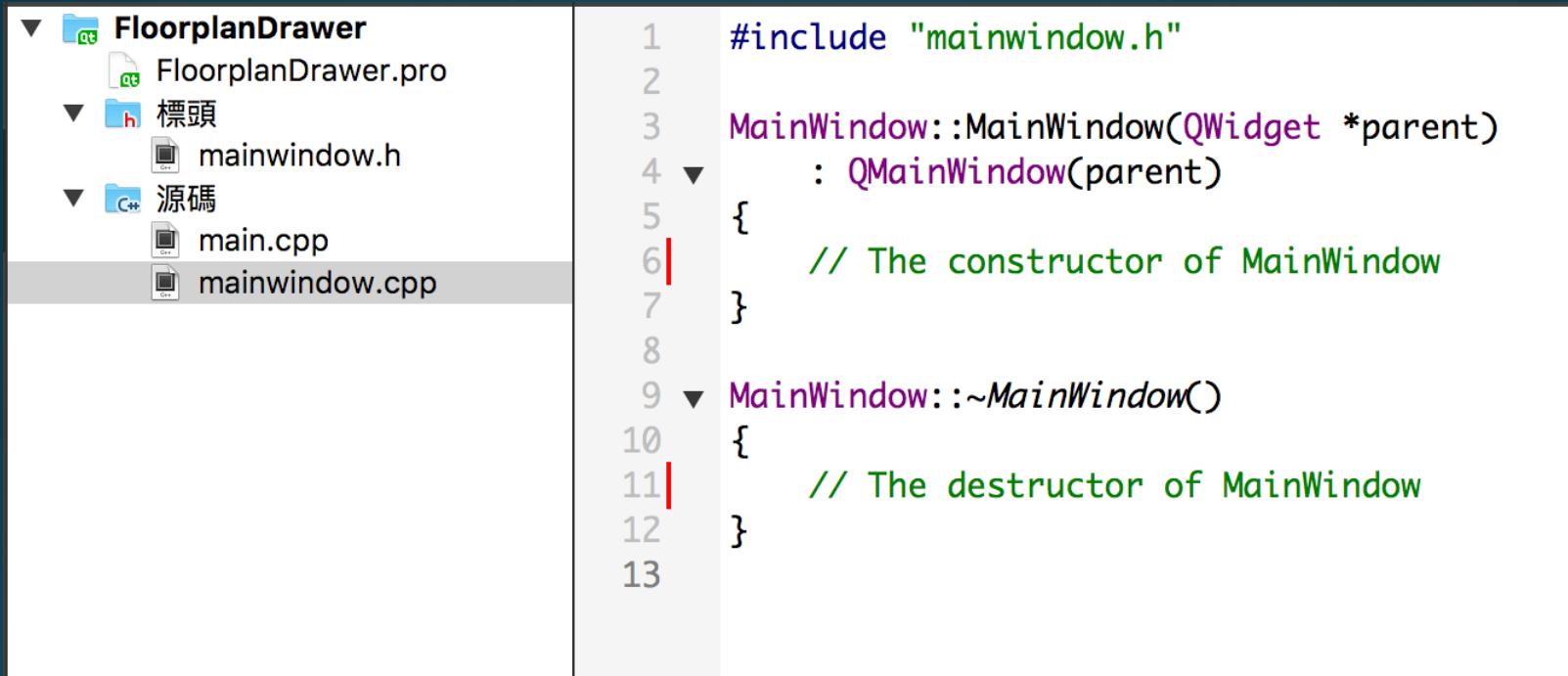
# What is Q\_OBJECT?

- <http://woboq.com/blog/how-qt-signals-slots-work.html>
- Control/Command + Left Click to check it thoroughly

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5
6 class MainWindow : public QMainWindow
7 {
8     Q_OBJECT // Line 8 highlighted with a red box
9
10 public:
11     MainWindow(QWidget *parent = 0);
12     ~MainWindow();
13 }
14
15 #endif // MAINWINDOW_H
16
```

```
194 #if defined(Q_CC_GNU) && !defined(Q_CC_INTEL) && Q_CC_GNU >= 600
195 # define Q_OBJECT_NO_ATTRIBUTES_WARNING QT_WARNING_DISABLE_G
196 #else
197 # define Q_OBJECT_NO_ATTRIBUTES_WARNING
198 #endif
199
200 /* qmake ignore Q_OBJECT */
201 #define Q_OBJECT \
202 public: \
203     Q_OBJECT_CHECK \
204     QT_WARNING_PUSH \
205     Q_OBJECT_NO_OVERRIDE_WARNING \
206     static const QMetaObject staticMetaObject; \
207     virtual const QMetaObject *metaObject() const; \
208     virtual void *qt_metacast(const char *); \
209     virtual int qt_metacall(QMetaObject::Call, int, void **); \
210     QT_TR_FUNCTIONS \
211 private: \
212     Q_OBJECT_NO_ATTRIBUTES_WARNING \
213     Q_DECL_HIDDEN_STATIC_METACALL static void qt_static_metacall \
214     QT_WARNING_POP \
215     struct QPrivateSignal {}; \
216     QT_ANNOTATE_CLASS(qt_qobject, "") \
217
218 /* qmake ignore Q_OBJECT */
219 #define Q_OBJECT_FAKE Q_OBJECT QT_ANNOTATE_CLASS(qt_fake, "")
```

# Overview to mainwindow.cpp



The screenshot shows the Qt Creator interface. On the left, the project tree for 'FloorplanDrawer' is visible, containing 'FloorplanDrawer.pro', a 'mainwindow.h' file under 'header', and two files ('main.cpp' and 'mainwindow.cpp') under 'source code'. The 'mainwindow.cpp' file is currently selected and shown in the main editor area. The code in 'mainwindow.cpp' is as follows:

```
1 #include "mainwindow.h"
2
3 MainWindow::MainWindow(QWidget *parent)
4     : QMainWindow(parent)
5 {
6     // The constructor of MainWindow
7 }
8
9 MainWindow::~MainWindow()
10 {
11     // The destructor of MainWindow
12 }
13
```

# Overview to main.cpp

The screenshot shows a Qt-based IDE interface. On the left, the project tree for "FloorplanDrawer" is displayed, containing "FloorplanDrawer.pro", "mainwindow.h", "main.cpp", and "mainwindow.cpp". The "main.cpp" file is selected. On the right, the code editor shows the following C++ code:

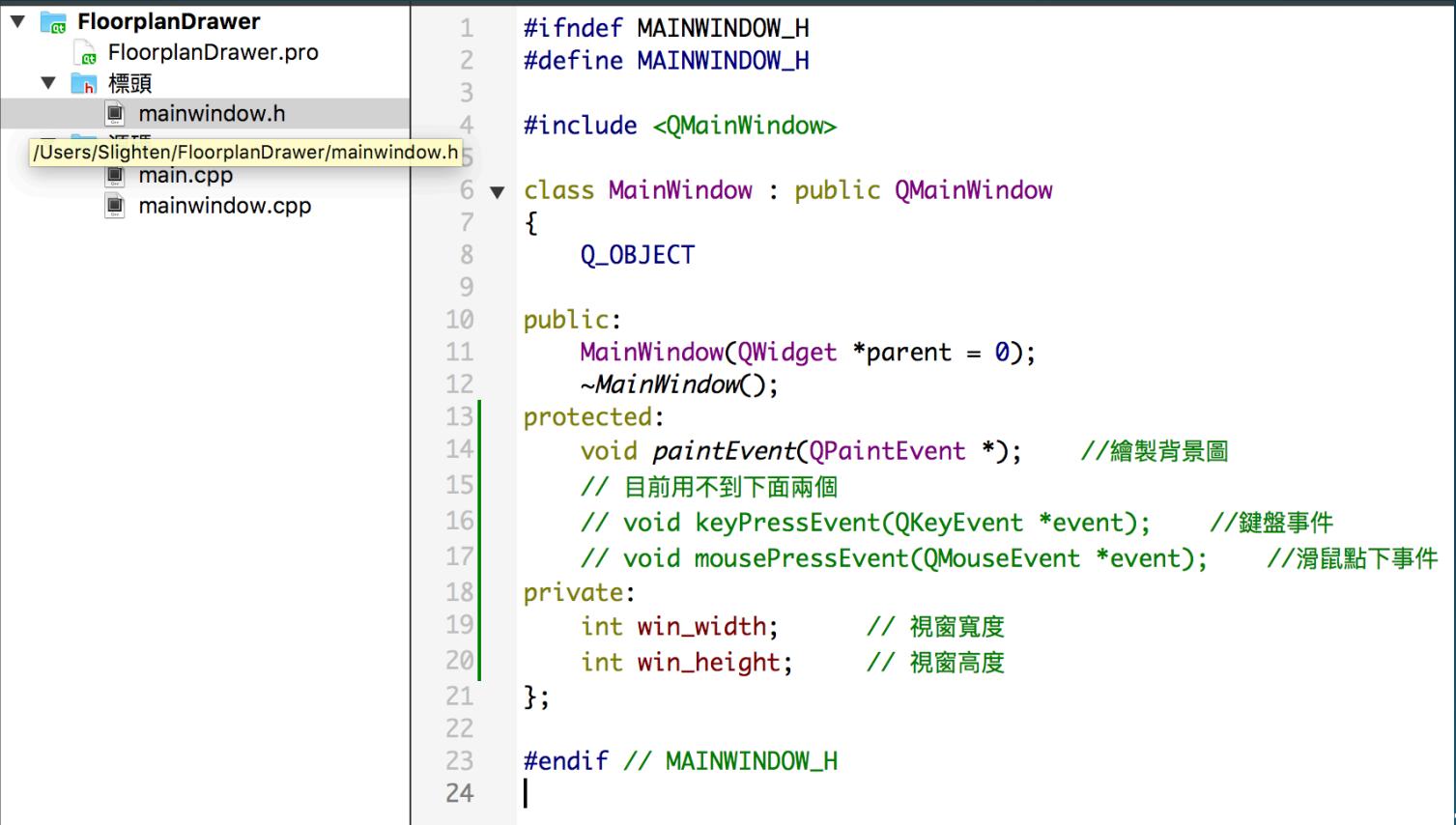
```
1 #include "mainwindow.h"
2 #include < QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     // Create a QApplication whose constructor requires
7     // argc and argv to be parameters
8     QApplication a(argc, argv);
9     // MainWindow Object Instantiation
10    MainWindow w;
11    // Show the main window
12    w.show();
13
14    // Enter an event loop and wait for users' action
15    return a.exec();
16
17 }
```

A red arrow points from the "return a.exec();" line in the code to a small screenshot of a Mac OS X-style window frame with red, yellow, and green buttons at the top center.

# A Simple Floorplan Painter

# paintEvent() member function

- Create a member function called paintEvent();



The screenshot shows a Qt-based IDE interface. On the left, the project tree displays a folder named 'FloorplanDrawer' containing a 'FloorplanDrawer.pro' file and a 'mainwindow.h' file under a '標頭' (Headers) folder. Below the tree, a file browser shows the contents of 'mainwindow.h'. The code in 'mainwindow.h' is as follows:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();
protected:
    void paintEvent(QPaintEvent *); // 繪製背景圖
    // 目前用不到下面兩個
    // void keyPressEvent(QKeyEvent *event); // 鍵盤事件
    // void mousePressEvent(QMouseEvent *event); // 滑鼠點下事件
private:
    int win_width; // 視窗寬度
    int win_height; // 視窗高度
};

#endif // MAINWINDOW_H
```

# Set Main Window Size in Constructor

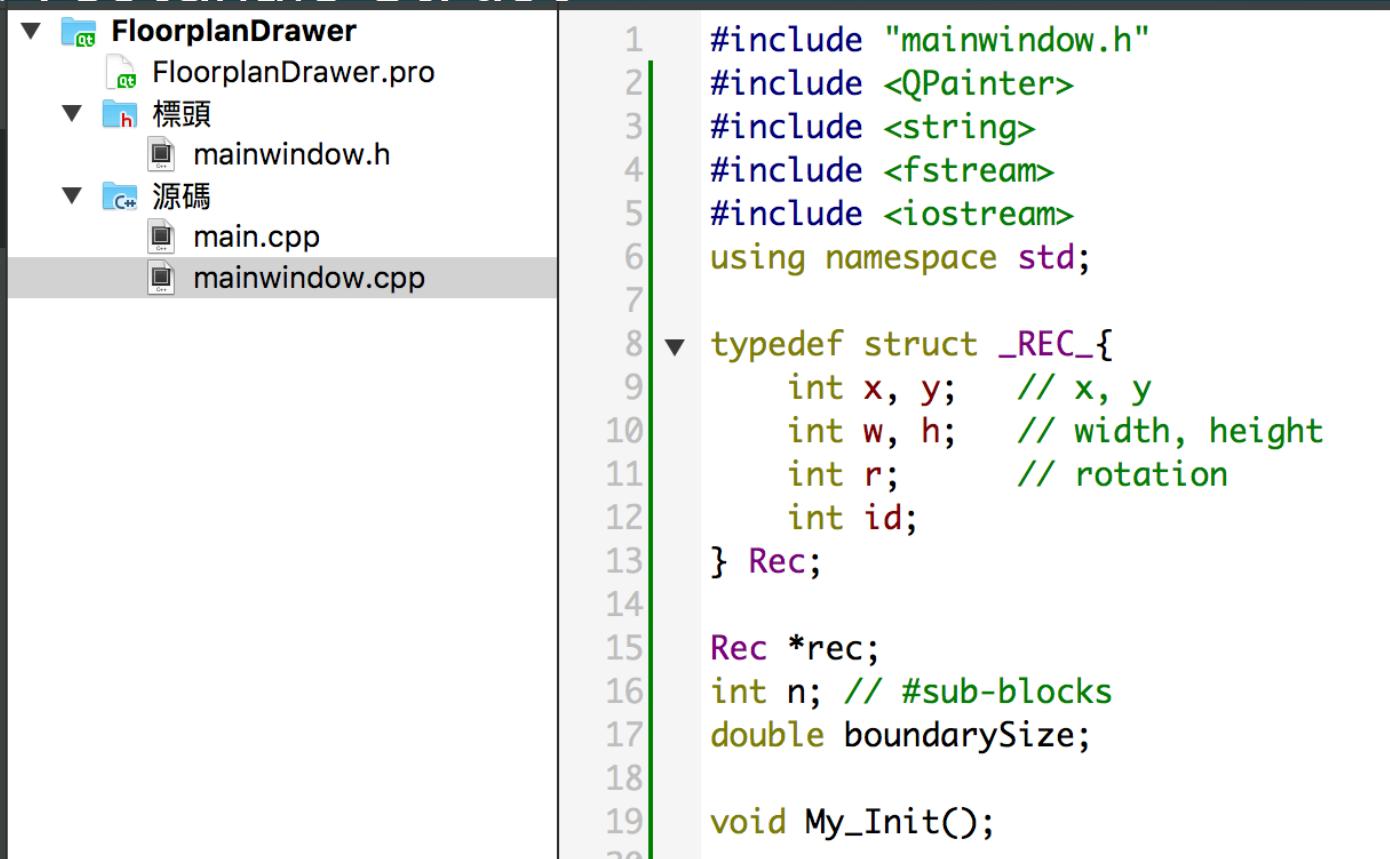
The screenshot shows the Qt Creator IDE interface. On the left, the project tree displays a .pro file, a header file 'mainwindow.h', and two source files, 'main.cpp' and 'mainwindow.cpp'. The 'mainwindow.cpp' file is currently selected and shown in the main editor area. The code in the editor is as follows:

```
21 MainWindow::MainWindow(QWidget *parent)
22     : QMainWindow(parent)
23 {
24     // The constructor of MainWindow
25     win_width = 500;
26     win_height = 500;
27     this->setFixedSize(win_width, win_height); //視窗大小
28     My_Init();
29 }
30 }
```

The code defines the constructor for the MainWindow class, which inherits from QMainWindow. It sets the window's width to 500 pixels and height to 500 pixels using the setFixedSize() method. It also calls the My\_Init() function.

# A Simple Rectangle Structure

- A rectangle struct



The image shows a screenshot of a Qt-based IDE. On the left, the project tree for "FloorplanDrawer" is visible, containing a .pro file, a "mainwindow.h" header file, a "main.cpp" source file, and a "mainwindow.cpp" source file. The "mainwindow.cpp" file is currently selected and shown in the main code editor window on the right. The code editor displays the following C++ code:

```
1 #include "mainwindow.h"
2 #include <QPainter>
3 #include <string>
4 #include <fstream>
5 #include <iostream>
6 using namespace std;
7
8 ▶ typedef struct _REC_{
9     int x, y;    // x, y
10    int w, h;    // width, height
11    int r;        // rotation
12    int id;
13 } Rec;
14
15 Rec *rec;
16 int n; // #sub-blocks
17 double boundarySize;
18
19 void My_Init();
20
```

# My\_Init()

- 把方塊資訊放進 struct

The screenshot shows a Qt IDE interface. On the left, the project structure is displayed under 'FloorplanDrawer':

- FloorplanDrawer (Project)
- FloorplanDrawer.pro
- Headers (Folder)
  - mainwindow.h
- Sources (Folder)
  - main.cpp
  - mainwindow.cpp

The 'main.cpp' file is currently selected and shown in the main editor area. The code implements the `My_Init()` function:

```
// 讀檔
void My_Init(){
    ifstream fin;
    int x, y, w, h, r, id;
    fin.open("/Users/Slighten/result.rpt" ios::in);
    if (!fin) { cerr << "read fail\n"; exit(0); }
    fin >> n >> boundarySize;
    rec = new Rec[n];
    for (int i = 0; i < n; i++){
        fin >> id >> x >> y >> w >> h >> r;
        rec[i].x = x;
        rec[i].y = y;
        rec[i].id = id;
        rec[i].r = r;
        if (r){
            rec[i].w = h;
            rec[i].h = w;
        }
        else {
            rec[i].w = w;
            rec[i].h = h;
        }
    }
    fin.close();
}
```

A red rectangular highlight is placed over the line `fin.open("/Users/Slighten/result.rpt" ios::in);`, indicating it is the current line of interest.

# My result.rpt

- 格式

#blocks(=n) boundarySize

$id_1 \ x_1 \ y_1 \ w_1 \ h_1 \ r_1$

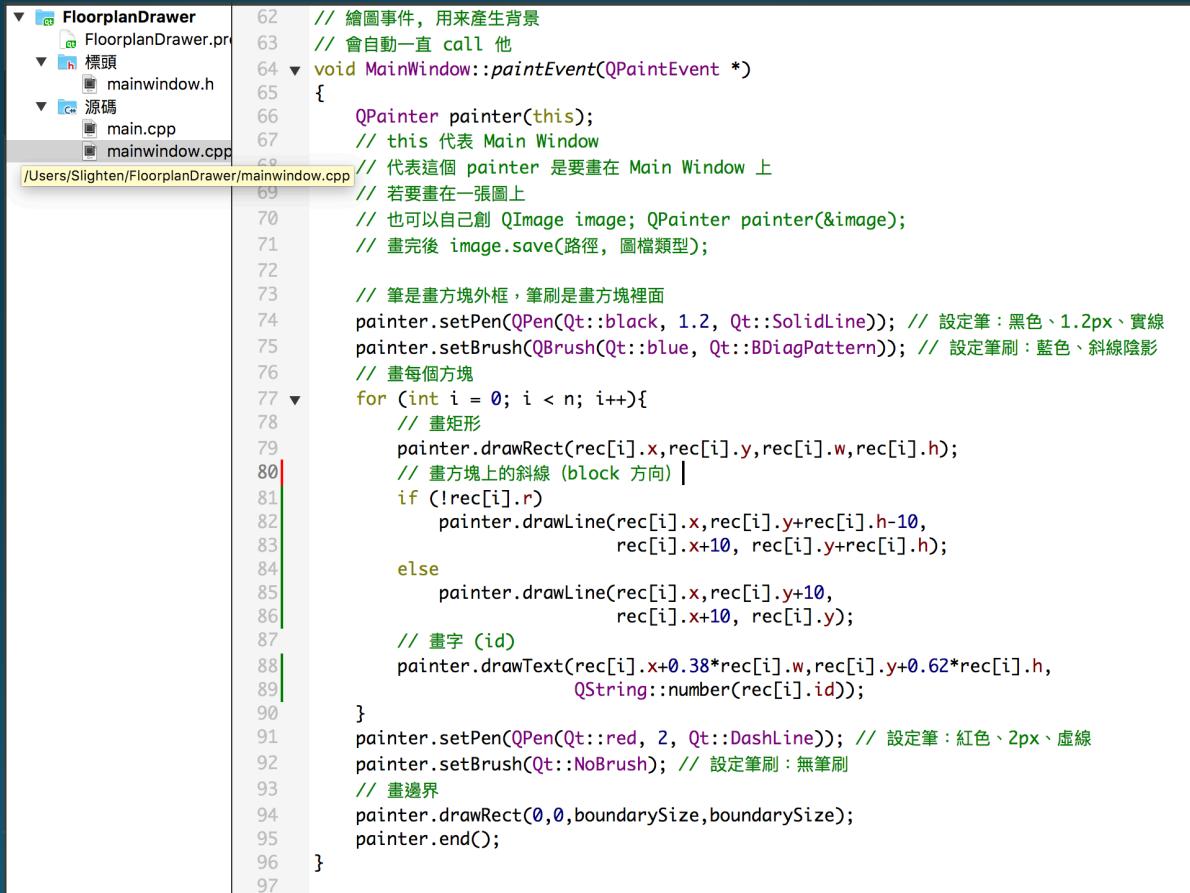
$id_2 \ x_2 \ y_2 \ w_2 \ h_2 \ r_2$

...

$id_n \ x_n \ y_n \ w_n \ h_n \ r_n$

result.rpt						
100	454	341446				
3	179	67	37	67	0	
36	242	223	39	67	0	
73	201	0	61	67	0	
97	385	308	17	67	1	
17	198	271	44	66	0	
96	54	256	40	66	0	
99	111	0	49	66	0	
27	361	109	25	65	0	
49	382	383	47	65	1	
56	319	420	17	63	1	
68	329	57	52	63	1	
8	128	236	50	62	1	
29	104	229	24	62	0	
98	50	158	22	62	0	
54	430	0	18	61	0	
66	268	73	67	61	1	
72	393	219	61	61	0	
37	312	327	20	60	0	
11	100	88	39	59	1	

# Implement paintEvent()

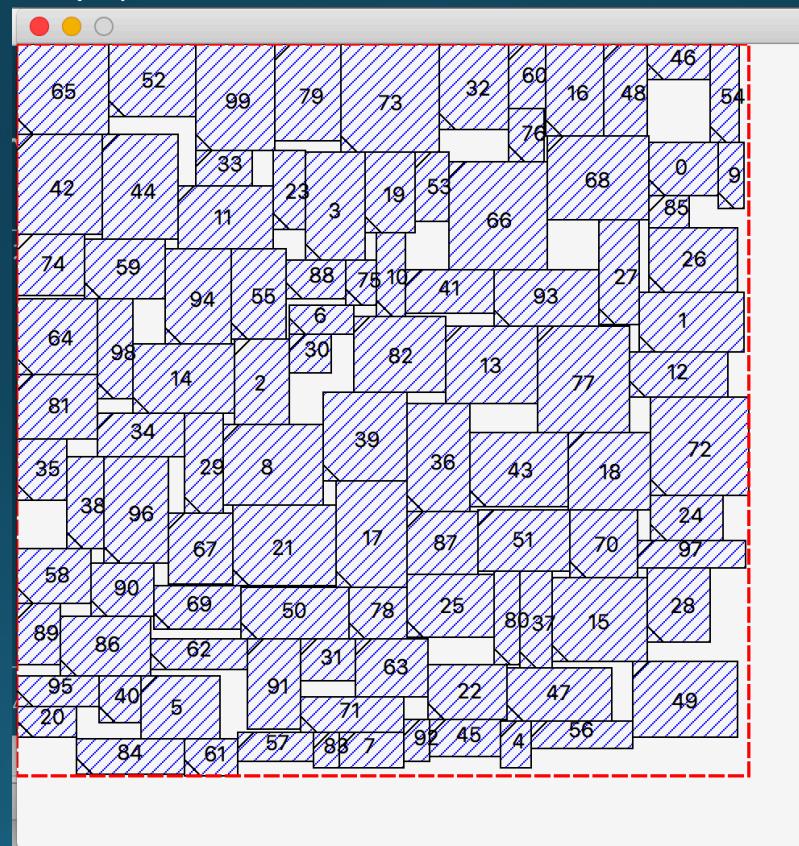


The screenshot shows a Qt-based IDE interface with a file tree on the left and a code editor on the right. The file tree for 'FloorplanDrawer' includes 'FloorplanDrawer.pro', 'mainwindow.h', 'main.cpp', and 'mainwindow.cpp'. The code editor displays the implementation of the 'paintEvent' method in 'mainwindow.cpp'. The code uses a QPainter to draw rectangles and diagonal lines on a Main Window, setting pen colors and styles, and drawing text labels for each rectangle.

```
62 // 繪圖事件，用來產生背景
63 // 會自動一直 call 他
64 void MainWindow::paintEvent(QPaintEvent *)
65 {
66     QPainter painter(this);
67     // this 代表 Main Window
68     // 代表這個 painter 是要畫在 Main Window 上
69     // 若要畫在一張圖上
70     // 也可以自己創 QImage image; QPainter painter(&image);
71     // 畫完後 image.save(路徑, 圖檔類型);
72
73     // 筆是畫方塊外框，筆刷是畫方塊裡面
74     painter.setPen(QPen(Qt::black, 1.2, Qt::SolidLine)); // 設定筆：黑色、1.2px、實線
75     painter.setBrush(QBrush(Qt::blue, Qt::BDiagPattern)); // 設定筆刷：藍色、斜線陰影
76     // 畫每個方塊
77     for (int i = 0; i < n; i++){
78         // 畫矩形
79         painter.drawRect(rec[i].x, rec[i].y, rec[i].w, rec[i].h);
80         // 畫方塊上的斜線 (block 方向) |
81         if (!rec[i].r)
82             painter.drawLine(rec[i].x, rec[i].y+rec[i].h-10,
83                             rec[i].x+10, rec[i].y+rec[i].h);
84         else
85             painter.drawLine(rec[i].x, rec[i].y+10,
86                             rec[i].x+10, rec[i].y);
87         // 畫字 (id)
88         painter.drawText(rec[i].x+0.38*rec[i].w, rec[i].y+0.62*rec[i].h,
89                         QString::number(rec[i].id));
90     }
91     painter.setPen(QPen(Qt::red, 2, Qt::DashLine)); // 設定筆：紅色、2px、虛線
92     painter.setBrush(Qt::NoBrush); // 設定筆刷：無筆刷
93     // 畫邊界
94     painter.drawRect(0, 0, boundarySize, boundarySize);
95     painter.end();
96 }
97 }
```

# Click 執行

- (0,0) 在左上角



# Code Template

- Please refer to: <https://github.com/Slighten/-Qt-C-FloorplanDrawer>
- For more details please refer to reference/  
Qt\_mini\_project/QT\_Introduction.pptx