

# 林林.台灣 | Linwebs - 課程

[首頁](#) / [課程](#) / [Qt](#) / 跨平台圖形化程式開發 - (3)賽車遊戲開發-程式碼實作

## 跨平台圖形化程式開發 - (3)賽車遊戲開發-程式碼實作

本文章為嘉大資工讀書會第二期課程內容

場次 4-1 【跨平台圖形化程式設計開發(二)】跨平台圖形化程式開發

本文章接續前一篇文章【[跨平台圖形化程式開發 - \(2\)賽車遊戲開發-介面編排設計](#)】內容

本文章為步驟截圖，詳細說明請參照課程影片。

此程式以 game\_status 變數作為遊戲狀態的 flag，狀態如下：

0 => init 遊戲被開啟時

1 => playing 遊戲遊玩時

2 => pause 遊戲暫停時

3 => timeout 時間到，遊戲結束時

4 => die 碰到障礙物，遊戲結束時

PS: 本課程圖片較多，此頁面圖片有經過壓縮處理，若圖片模糊不清，可點選圖片開啟圖片原始檔

1. 首先，先讓背景可動態移動，隨著車子的前進而改變。

開啟【mainwindow.h】的標頭檔撰寫以下程式碼。

```

MainWindow.h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void update_object(); // 遊戲物體移動更新

private:
    Ui::MainWindow *ui;
    int game_status; // 遊戲狀態
    int bgm_pos; // 背景的位置(水平)
    QTimer *object_timer; // 物體移動計時器
};

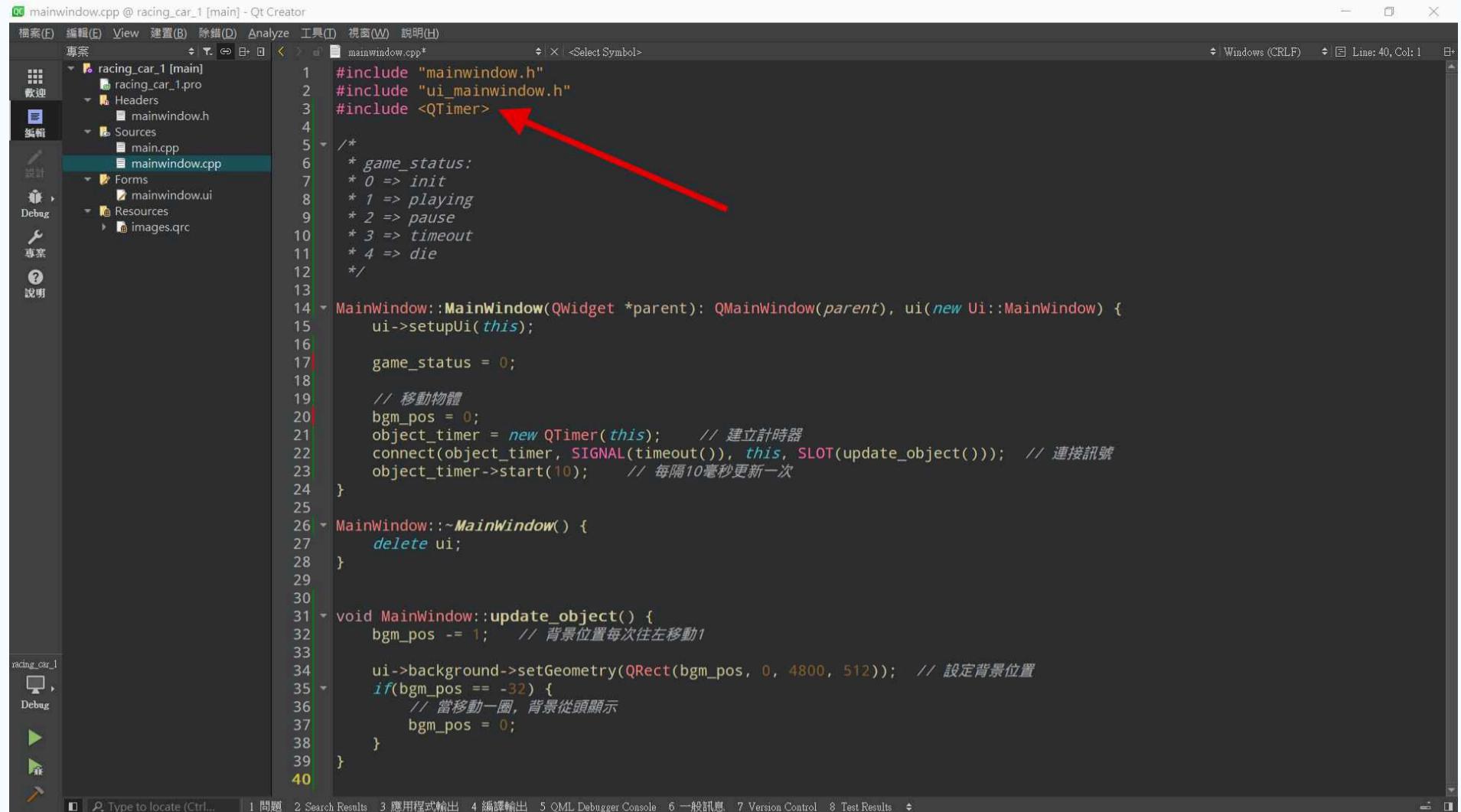
#endif // MAINWINDOW_H

```

2. 在此使用計時器，每當一段時間就呼叫函式，執行移動背景的程式碼片段。

首先，引入 QTimer 檔案。

接下來，在建構子函式中加入以下程式碼，並撰寫 update\_object 的函式。



```

mainwindow.cpp @ racing_car_1 [main] - Qt Creator
檔案(F) 編輯(E) View 建置(B) 除錯(D) Analyze 工具(I) 視窗(W) 說明(H)
專案 racing_car_1 [main]
    欢迎
    編輯
        mainwindow.h
        main.cpp
    設計
        mainwindow.ui
    Resources
        images.qrc
    Debug
    專案
    說明

mainwindow.cpp
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QTimer> ↑
4
5 /*
6 * game_status:
7 * 0 => init
8 * 1 => playing
9 * 2 => pause
10 * 3 => timeout
11 * 4 => die
12 */
13
14 MainWindow::MainWindow(QWidget *parent): QMainWindow(parent), ui(new Ui::MainWindow) {
15     ui->setupUi(this);
16
17     game_status = 0;
18
19     // 移動物體
20     bgm_pos = 0;
21     object_timer = new QTimer(this); // 建立計時器
22     connect(object_timer, SIGNAL(timeout()), this, SLOT(update_object())); // 連接訊號
23     object_timer->start(10); // 每隔10毫秒更新一次
24 }
25
26 MainWindow::~MainWindow() {
27     delete ui;
28 }
29
30
31 void MainWindow::update_object() {
32     bgm_pos -= 1; // 背景位置每次往左移動1
33
34     ui->background->setGeometry(QRect(bgm_pos, 0, 4800, 512)); // 設定背景位置
35     if(bgm_pos == -32) {
36         // 當移動一圈，背景從頭顯示
37         bgm_pos = 0;
38     }
39 }
40

```

3. 建置並執行，可看到背景已可動態的移動，彷彿車子不斷在前進。



4. 接下來，加入遊戲時間計時的部分，在標頭檔中加入以下程式碼。

```

mainwindow.h @ racing_car_1 [main] - Qt Creator
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include < QMainWindow>
5
6 QT_BEGIN_NAMESPACE
7 namespace Ui { class MainWindow; }
8 QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14     public:
15         MainWindow(QWidget *parent = nullptr);
16         ~MainWindow();
17
18     private slots:
19         void update_time(); // 遊戲時間更新
20         void update_object(); // 遊戲物體移動更新
21
22     private:
23         Ui::MainWindow *ui; // 遊戲時間
24         int time; // 遊戲狀態
25         int game_status; // 遊戲物體移動
26         int bgm_pos; // 背景的位置(水平)
27
28         QTimer *clock_timer; // 遊戲時間計時器
29         QTimer *object_timer; // 物體移動計時器
30
31         void game_start();
32         void game_pause();
33         void game_stop();
34 };
35
36 #endif // MAINWINDOW_H

```

5. 撰寫以下遊戲時間計時器及遊戲停止的程式碼。

```

mainwindow.cpp @ racing_car_1 [main] - Qt Creator
13
14 MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow) {
15     ui->setupUi(this);
16
17     game_status = 0;
18
19     // 移動物體
20     bgm_pos = 0;
21     object_timer = new QTimer(this); // 建立計時器
22     connect(object_timer, SIGNAL(timeout()), this, SLOT(update_object())); // 連接訊號
23     object_timer->start(10); // 每隔10毫秒更新一次
24
25     // 更新時間
26     time = 30;
27     clock_timer = new QTimer(this); // 建立計時器
28     connect(clock_timer, SIGNAL(timeout()), this, SLOT(update_time())); // 連接訊號
29     //clock_timer->start(1000); // 每隔1000毫秒更新一次
30 }
31
32 MainWindow::~MainWindow() {
33     delete ui;
34 }
35
36 void MainWindow::update_time() {
37     time -= 1; // 時間每次減少1
38
39     ui->lcd_clock->display(time);
40
41     if(time == 0) {
42         game_status = 3; // 設為 timeout 的狀態
43         game_stop();
44     }
45 }
46
47 void MainWindow::game_stop() {
48     clock_timer->stop();
49     object_timer->stop();
50 }
51
52 void MainWindow::update_object() {
53     bgm_pos -= 1; // 背景位置每次往左移動1
54 }

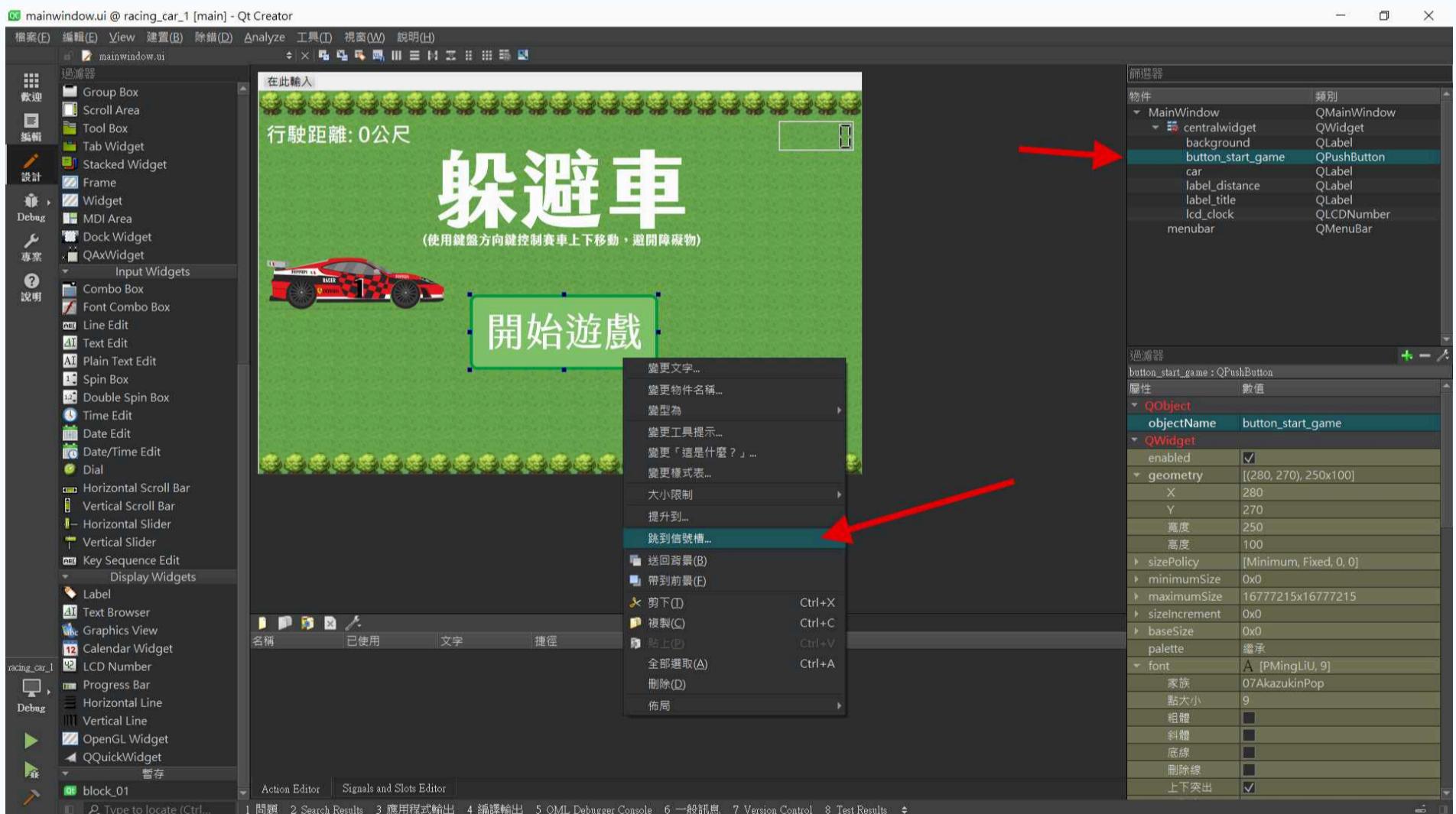
```

6. 建置並執行，可看到遊戲時間計時器已可運作。

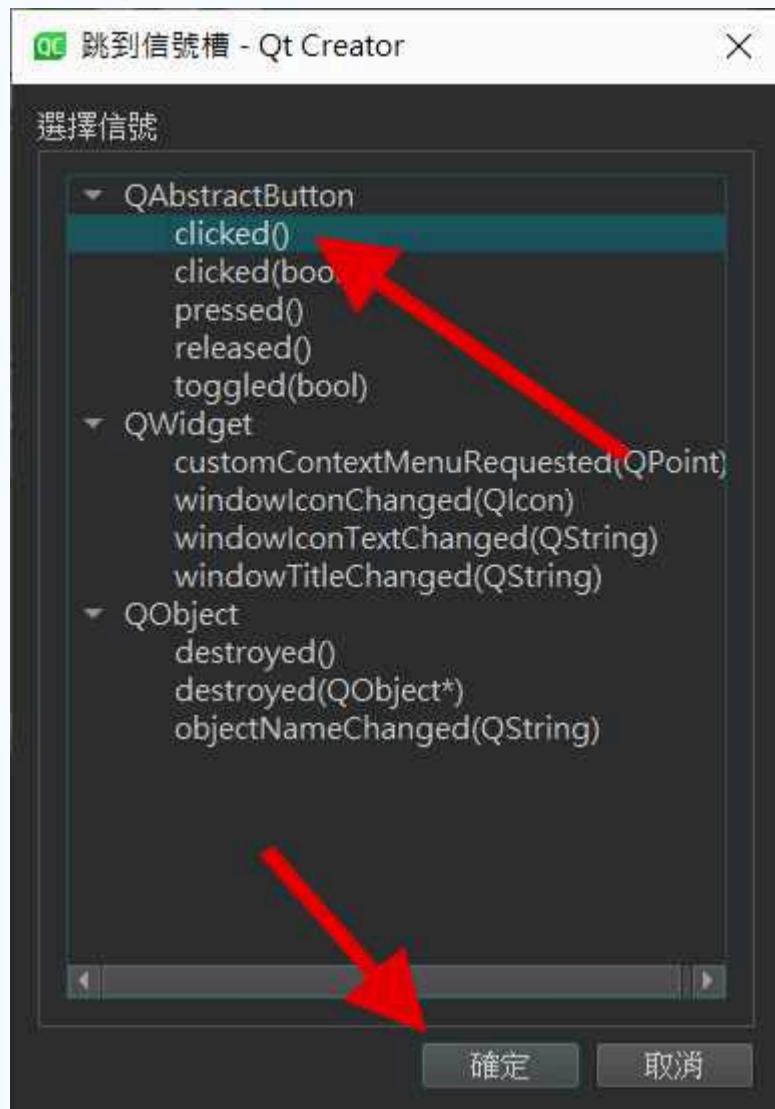


7. 切換到 ui 設計的部分，點選開始遊戲的按鈕，加入按鈕的觸發事件。

在按鈕上點選滑鼠右鍵，選擇【跳到信號槽...】。



## 8. 選擇【QAbstractButton】中的【clicked()】訊號。



9. 可看到 Qt Creator 自動產生觸發此按鈕時，會執行的函式的程式碼。

```

mainwindow.cpp @ racing_car_1 [main] - Qt Creator
mainwindow.cpp
MainWindow::MainWindow()
{
    clock_timer = new QTimer(this); // 建立計時器
    connect(clock_timer, SIGNAL(timeout()), this, SLOT(update_time())); // 連接訊號
    clock_timer->start(1000); // 每隔1000毫秒更新一次
}
MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::update_time()
{
    time -= 1; // 時間每次減少1
    ui->lcd_clock->display(time);
    if(time == 0) {
        game_status = 3;
        game_stop();
    }
}

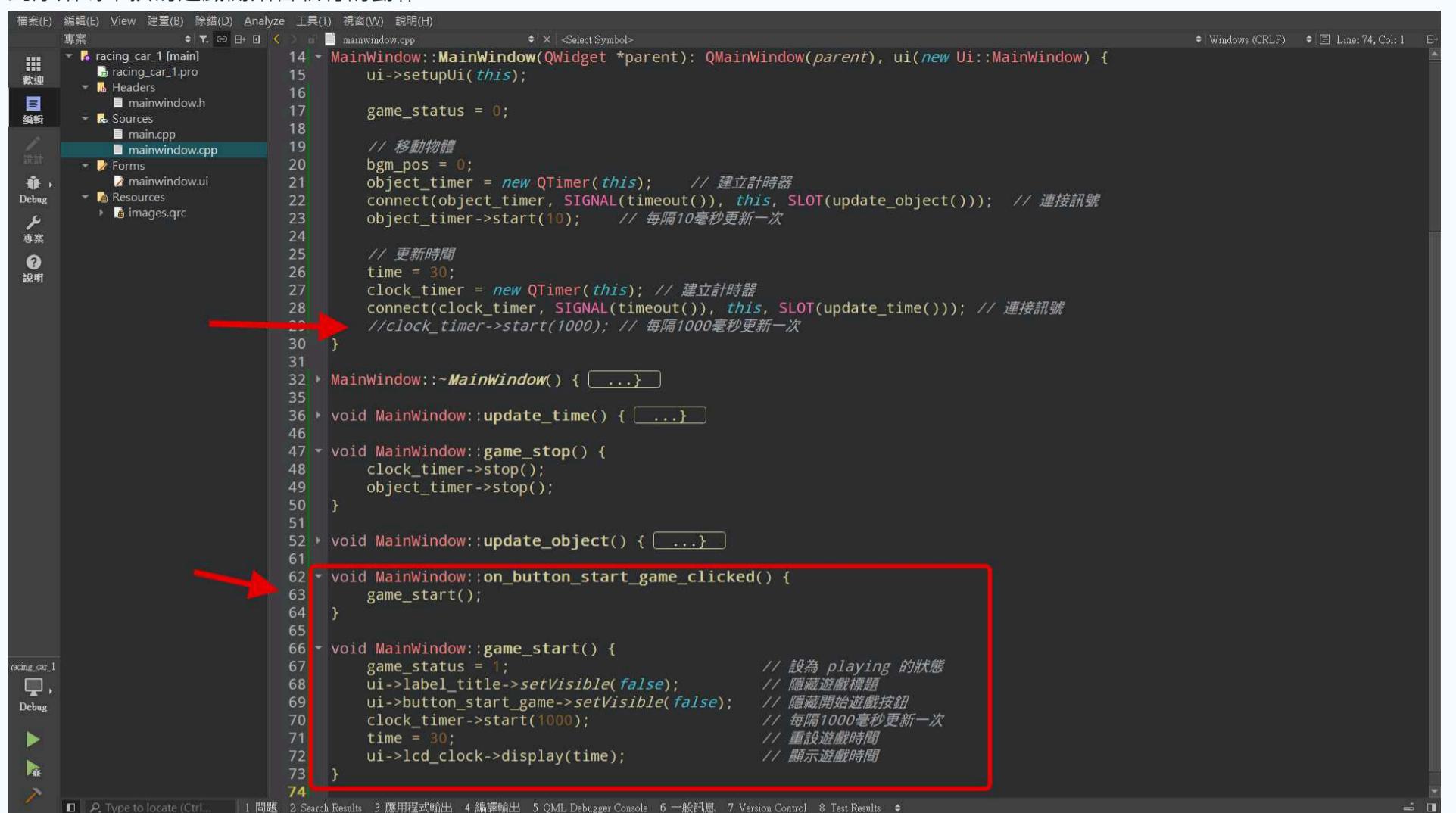
void MainWindow::game_stop()
{
    clock_timer->stop();
    object_timer->stop();
}

void MainWindow::update_object()
{
    bgm_pos -= 1; // 背景位置每次往左移動1
    ui->background->setGeometry(QRect(bgm_pos, 0, 4800, 512)); // 設定背景位置
    if(bgm_pos == -32) {
        // 當移動一圈，背景從頭顯示
        bgm_pos = 0;
    }
}

void MainWindow::on_button_start_game_clicked()
{
}

```

## 10. 到原始碼中撰寫遊戲開始會執行的動作。



```

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow) {
    ui->setupUi(this);

    game_status = 0;

    // 移動物體
    bgm_pos = 0;
    object_timer = new QTimer(this); // 建立計時器
    connect(object_timer, SIGNAL(timeout()), this, SLOT(update_object())); // 連接訊號
    object_timer->start(10); // 每隔1毫秒更新一次

    // 更新時間
    time = 30;
    clock_timer = new QTimer(this); // 建立計時器
    connect(clock_timer, SIGNAL(timeout()), this, SLOT(update_time())); // 連接訊號
    //clock_timer->start(1000); // 每隔1000毫秒更新一次

}

MainWindow::~MainWindow() { ... }

void MainWindow::update_time() { ... }

void MainWindow::game_stop() {
    clock_timer->stop();
    object_timer->stop();
}

void MainWindow::update_object() { ... }

void MainWindow::on_button_start_game_clicked() {
    game_start();
}

void MainWindow::game_start() {
    game_status = 1; // 設為 playing 的狀態
    ui->label_title->setVisible(false); // 隱藏遊戲標題
    ui->button_start_game->setVisible(false); // 隱藏開始遊戲按鈕
    clock_timer->start(1000); // 每隔1000毫秒更新一次
    time = 30; // 重設遊戲時間
    ui->lcd_clock->display(time); // 顯示遊戲時間
}

```

11. 建置並執行，可看到當點選開始按鈕時，遊戲標題及開始按鈕會隱藏，計時器會開始計時。



12. 當按下鍵盤方向鍵時，移動車輛的位置，到標頭檔中撰寫以下程式碼，偵測鍵盤按壓的事件。

```

#include <QMainWindow>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

protected:
    void keyPressEvent(QKeyEvent *);

private slots:
    void update_time(); // 遊戲時間更新
    void update_object(); // 遊戲物體移動更新

    void on_button_start_game_clicked();

private:
    Ui::MainWindow *ui;
    int time; // 遊戲時間
    int game_status; // 遊戲狀態
    int bgm_pos; // 背景的位置(水平)
    int car_pos; // 車子的位置(垂直)
    int car_direction; // 車子的方向(垂直)
    float car_distance; // 車子移動的距離

    QTimer *clock_timer; // 遊戲時間計時器
    QTimer *object_timer; // 物體移動計時器

    void game_start();
    void game_pause();
    void game_stop();
};

#endif // MAINWINDOW_H

```

13. 到原始碼中引入 QKeyEvent 和 qDebug 檔案。

QDebug 類似 C++ 的 cout 功能，可在 Qt 程式測試時輸出 Debug 訊息在 Qt Creator 上，方便程式的偵錯。

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QTimer>
#include <QKeyEvent>

/*
 * game_status:
 * 0 => init
 * 1 => playing
 * 2 => pause
 * 3 => timeout
 * 4 => die
 */

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow) { ... }

MainWindow::~MainWindow() { ... }

void MainWindow::update_time() { ... }

void MainWindow::game_stop() { ... }

void MainWindow::update_object() { ... }

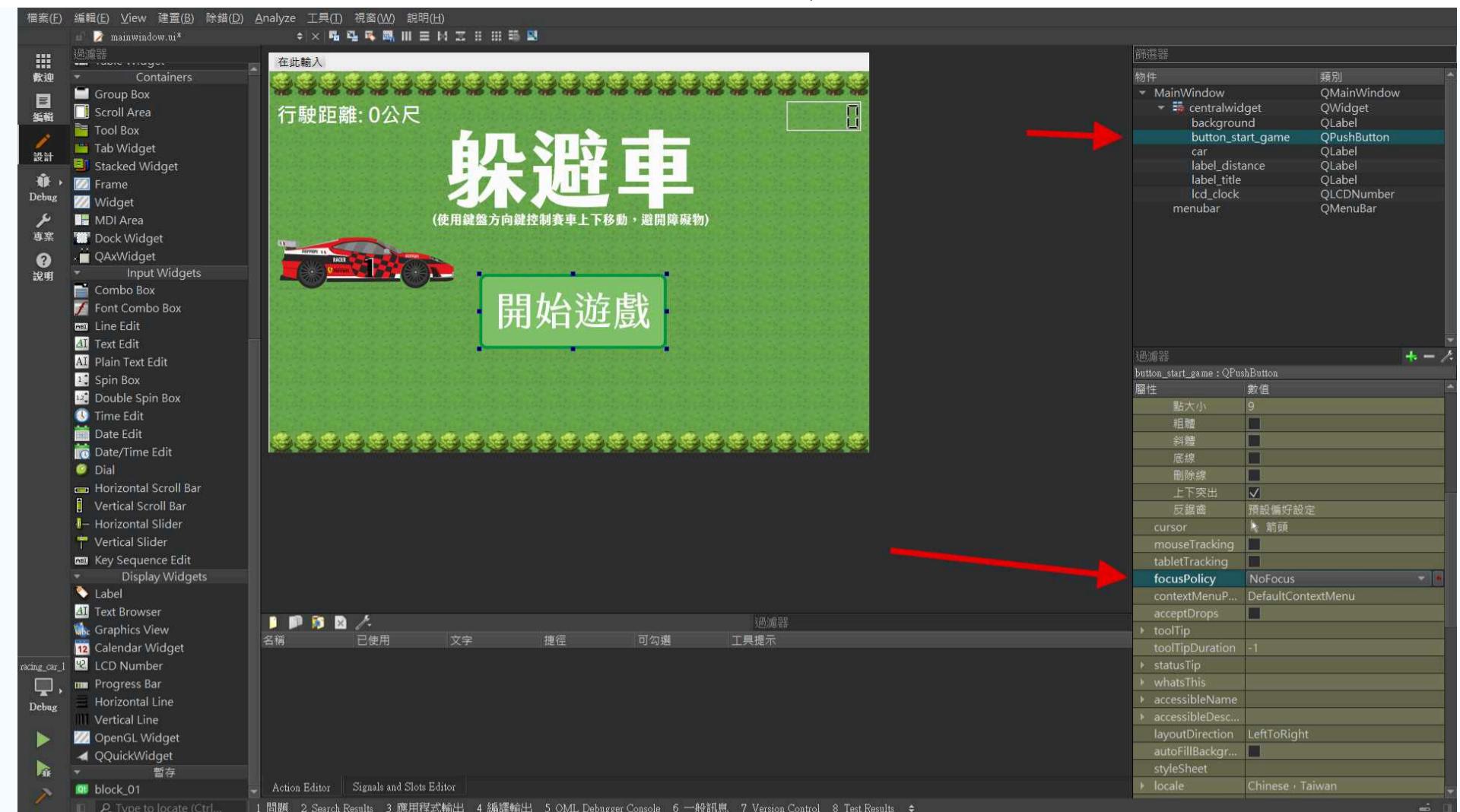
void MainWindow::on_button_start_game_clicked() { ... }

void MainWindow::game_start() { ... }

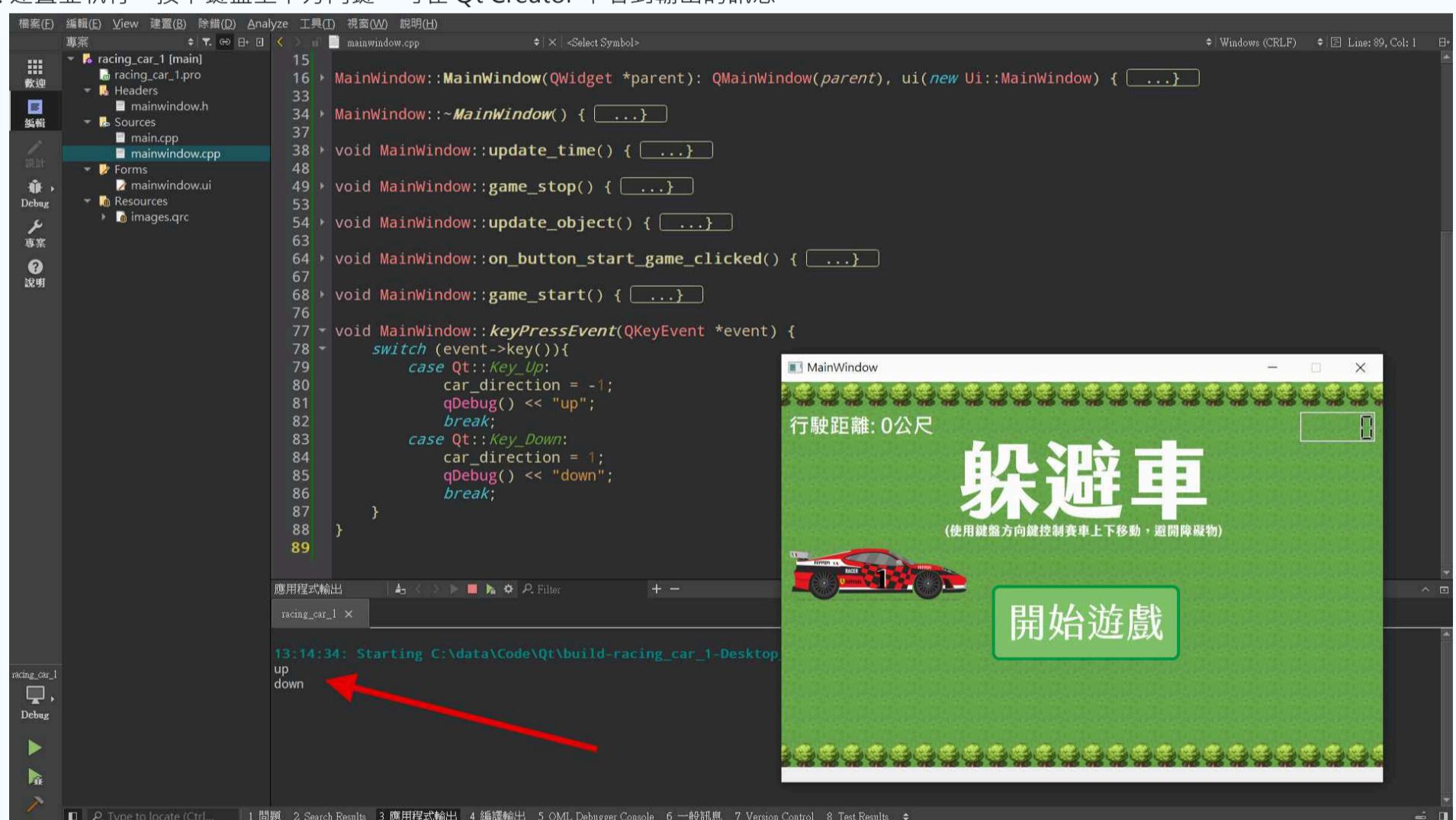
void MainWindow::keyPressEvent(QKeyEvent *event) {
    switch (event->key()) {
        case Qt::Key_Up:
            car_direction = -1;
            qDebug() << "up";
            break;
        case Qt::Key_Down:
            car_direction = 1;
            qDebug() << "down";
            break;
    }
}

```

14. 由於開始遊戲的按鈕會優先搶奪 ui 的 focus，必須至 ui 設計中，把按鈕的 focusPolicy 設為 NoFocus，才不會導致按下按鈕時，讓程式誤以為要對按鈕進行操作。



15. 建置並執行，按下鍵盤上下方向鍵，可在 Qt Creator 中看到輸出的訊息。



## 16. 接下來撰寫移動車子的程式，切換到原始碼，撰寫以下程式。

```

MainWindow::update_object() {
    bgm_pos -= 1; // 背景位置每次往左移動1
    ui->background->setGeometry(QRect(bgm_pos, 0, 4800, 512)); // 設定背景位置
    if(bgm_pos == -32) {
        // 當移動一圈，背景從頭顯示
        bgm_pos = 0;
    }
    if(game_status == 1) {
        // playing
        car_distance += 1; // 增加移動距離
        ui->label_distance->setText("行駛距離: " + QString::number(car_distance, 'f', 0) + "公尺"); // 顯示移動距離
        move_car(); // 呼叫 move_car()
    }
}

void MainWindow::move_car() {
    int car_new_pos = car_pos + car_direction; // 將接下來要移動到的位置暫存起來
    if((car_new_pos) >= 30 && (car_new_pos) <= 410) { // 判斷接下來要移動的位置是否超過跑道上下邊緣
        // 未超過邊緣
        car_pos = car_new_pos; // 變更車子的位置
        ui->car->setGeometry(QRect(10, car_pos, 237, 71)); // 設定車子的垂直位置
    } else {
        // 超過邊緣
        game_status = 4; // 設為 die 的狀態
        game_stop(); // 遊戲結束
    }
}

```

## 17. 撰寫以下程式。

```

void MainWindow::game_start() {
    game_status = 1; // 設為 playing 的狀態
    ui->label_title->setVisible(false); // 隱藏遊戲標題
    ui->button_start_game->setVisible(false); // 隱藏開始遊戲按鈕
    clock_timer->start(1000); // 每隔1000毫秒更新一次
    time = 30; // 重設遊戲時間
    ui->lcd_clock->display(time); // 顯示遊戲時間
    car_pos = 220; // 重設車子的垂直位置(位於中央)
    car_distance = 0; // 重設移動距離
}

void MainWindow::keyPressEvent(QKeyEvent *event) {
    switch (event->key()) {
        case Qt::Key_Up:
            car_direction = -1;
            qDebug() << "up";
            break;
        case Qt::Key_Down:
            car_direction = 1;
            qDebug() << "down";
            break;
    }
}

```

### 18. 建置並執行，按下鍵盤上下方向鍵，可看到車子可上下移動。

The screenshot shows the Qt Creator interface with the project 'racing\_car\_1' open. The code editor displays mainwindow.cpp with the following snippet:

```

37 void MainWindow::update_time() { ... }
38 void MainWindow::game_stop() { ... }
39 void MainWindow::update_object() { ... }
40
41 void MainWindow::keyPressEvent(QKeyEvent *event) {
42     if (event->key() == Qt::Key_Down) {
43         car_pos = car_new_pos - 10;
44         ui->car->setGeometry(QRect(10, car_pos, 237, 71));
45     } else if (event->key() == Qt::Key_Up) {
46         car_pos = car_new_pos + 10;
47         ui->car->setGeometry(QRect(10, car_pos, 237, 71));
48     }
49 }

```

The application window shows a red racing car on a green track with a progress bar at the top right. The status bar indicates '行駛距離: 195公尺'. The application output terminal shows the following log:

```

13:25:22: C:\data\Code\Qt\build-racing_car_1-Desktop_Qt_5_15_2_MinGW_32_bit-Debug\debug\racing_car_1.exe exited with code 0
13:25:24: Starting C:\data\Code\Qt\build-racing_car_1-Desktop_Qt_5_15_2_MinGW_32_bit-Debug\debug\racing_car_1.exe ...
down
up
down
up
down

```

### 19. 當車子碰到跑到邊緣時，遊戲會停止。

The screenshot shows the Qt Creator interface with the project 'racing\_car\_1' open. The code editor displays mainwindow.cpp with the following snippet:

```

76     car_pos = car_new_pos; // 變更車子的位置
77     ui->car->setGeometry(QRect(10, car_pos, 237, 71)); // 設定車子的垂直位置
78 } else {
79     // 超過邊緣
80     game_status = 4; // 設為 die 的狀態
81     game_stop(); // 遊戲結束
82 }
83
84 void MainWindow::on_button_start_game_clicked() {
85     game_start();
86 }
87
88 void MainWindow::game_start() {
89     game_status = 1; // 設為 playing 的狀態
90     ui->label_title->setVisible(false);
91     ui->button_start_game->setVisible(false);
92     clock_timer->start(1000);
93     time = 30;
94     ui->lcd_clock->display(time);
95     car_pos = 220;
96     car_distance = 0;
97 }
98
99 void MainWindow::keyPressEvent(QKeyEvent *event) {
100 }
101
102 }
103
104 void MainWindow::keyPressEvent(QKeyEvent *event) {
105 }

```

The application window shows the red racing car at the edge of the track. The status bar indicates '行駛距離: 834公尺'. The application output terminal shows the following log:

```

13:18:30: Starting C:\data\Code\Qt\build-racing_car_1-Desktop_Qt_5_15_2_MinGW_32_bit-Debug\debug\racing_car_1.exe
down

```

### 20. 接下來，讓遊戲增加難度，新增一些障礙物在遊戲中。

在 ui 設計中加入一些 Label。

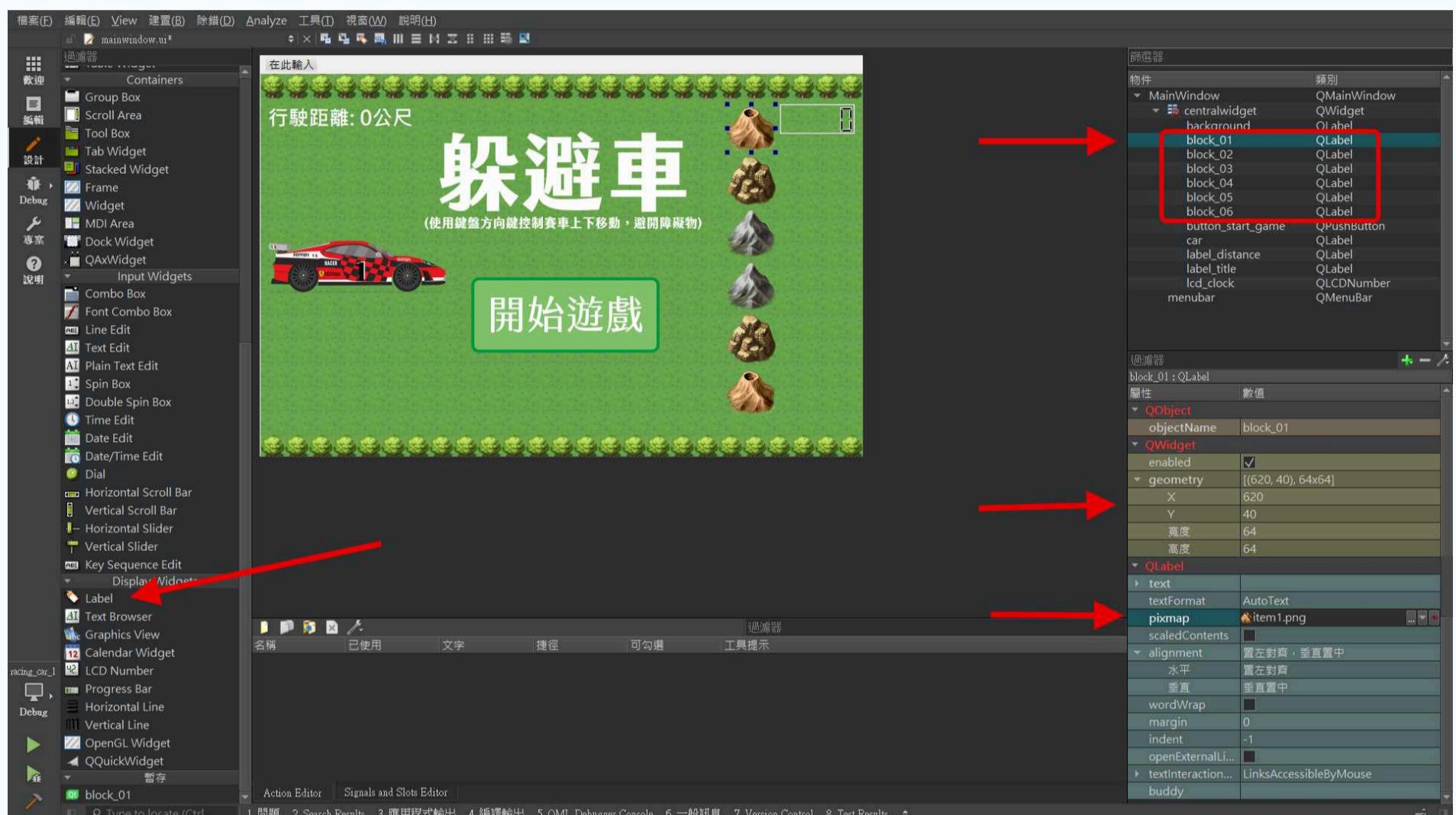
```

block_01 = {
    objectName: block_01,
    geometry: [x: 620, y: 40, 寬度: 64, 高度: 64],
    pixmap: item1.png
}

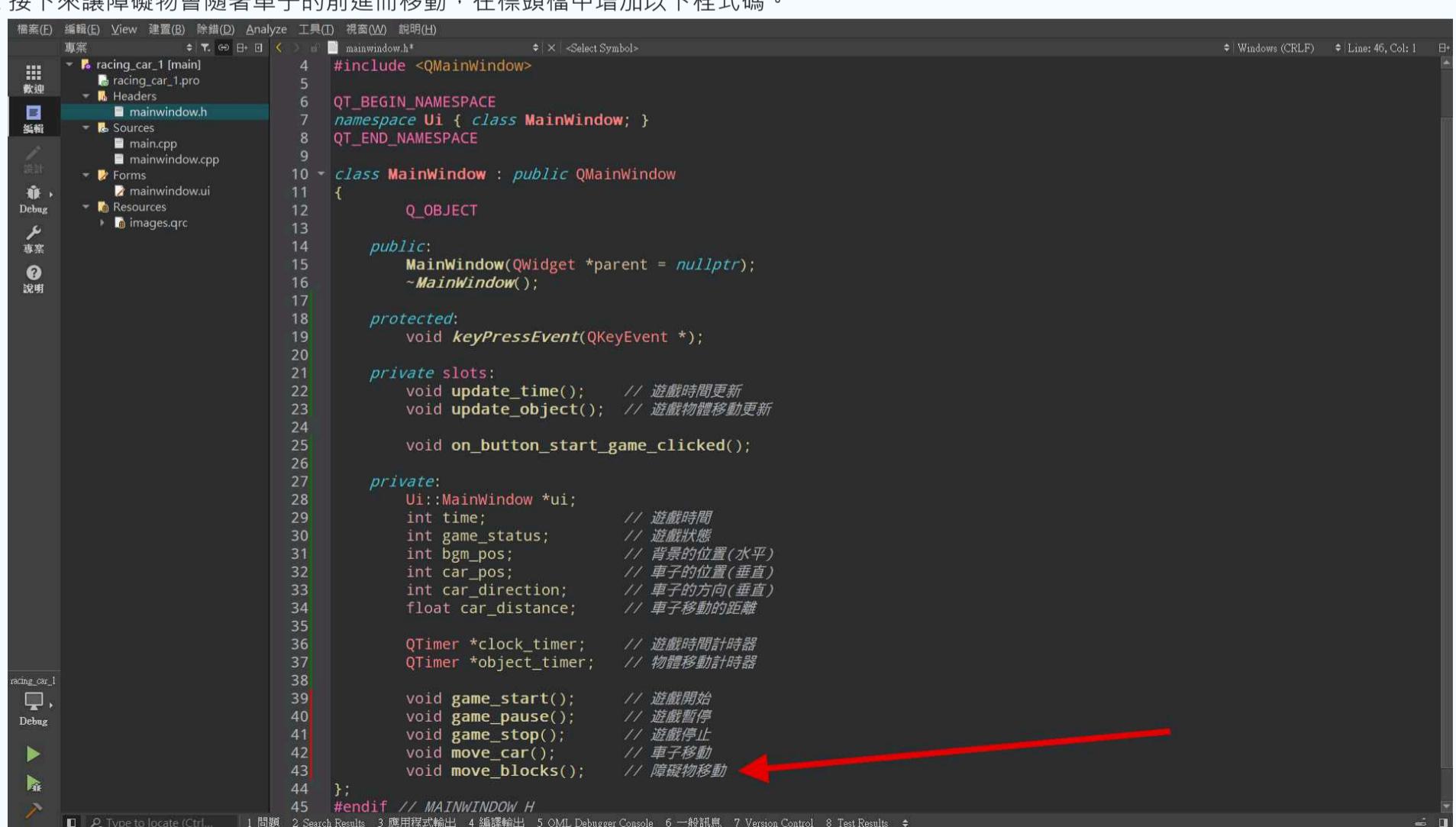
block_02 = {
    objectName: block_02,
    geometry: [x: 620, y: 110, 寬度: 64, 高度: 64],
    pixmap: item2.png
}

block_03 = {
    objectName: block_03,
    geometry: [x: 620, y: 180, 寬度: 64, 高度: 64],
    pixmap: item3.png
}

```



## 21. 接下來讓障礙物會隨著車子的前進而移動，在標頭檔中增加以下程式碼。



## 22. 首先，在程式被開啟時，將障礙物放在畫面看不到的地方。

The screenshot shows the Qt Creator IDE with the mainwindow.cpp file open. A red arrow points to line 23, which sets the geometry of six blocks to (-100, 0, 64, 64). The right side of the screen shows a game window titled 'MainWindow' with a red racing car on a track. The text '躲避車' (Avoidance) is displayed prominently. A green button labeled '開始遊戲' (Start Game) is visible. The status bar at the bottom indicates the game distance is 0 meters.

```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4 #include <QTimer>
5 #include <QKeyEvent>
6 /*
7 * game_status:
8 * 0 => init
9 * 1 => playing
10 * 2 => pause
11 * 3 => timeout
12 * 4 => die
13 */
14
15 MainWindow::MainWindow(QWidget *parent): QMainWindow(parent)
16     ui->setupUi(this);
17
18     game_status = 0;
19
20     // hide block
21     ui->block_01->setGeometry(QRect(-100, 0, 64, 64));
22     ui->block_02->setGeometry(QRect(-100, 0, 64, 64));
23     ui->block_03->setGeometry(QRect(-100, 0, 64, 64));
24     ui->block_04->setGeometry(QRect(-100, 0, 64, 64));
25     ui->block_05->setGeometry(QRect(-100, 0, 64, 64));
26     ui->block_06->setGeometry(QRect(-100, 0, 64, 64));
27
28     // 移動物體
29     bgm_pos = 0;
30     object_timer = new QTimer(this); // 建立計時器
31     connect(object_timer, SIGNAL(timeout()), this, SLOT(update_object()));
32     object_timer->start(10); // 每隔10毫秒更新一次
33
34     // 更新時間
35     time = 30;
36     clock_timer = new QTimer(this); // 建立計時器
37     connect(clock_timer, SIGNAL(timeout()), this, SLOT(update_time()));
38     //clock_timer->start(1000); // 每隔1000毫秒更新一次
39 }
40
41 MainWindow::~MainWindow() { ... }
42

```

## 23. 使用 define 定義障礙物的位置(也可使用陣列、結構等方式儲存)。

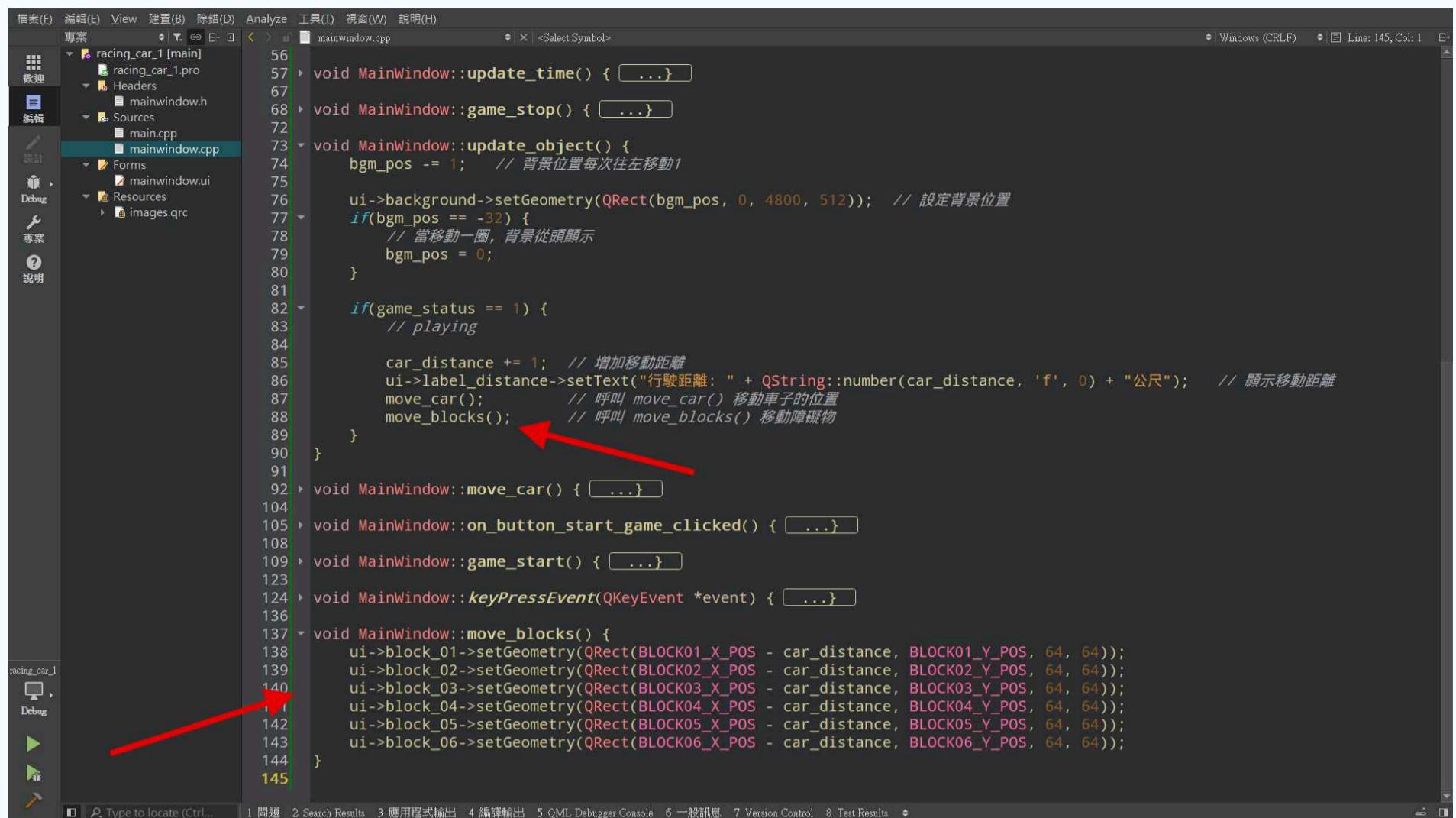
The screenshot shows the Qt Creator IDE with the mainwindow.cpp file open. A red arrow points to the define statements starting from line 6, which define the x and y positions for six blocks. The right side of the screen shows a game window with a red racing car and several green bushes representing obstacles. The status bar at the bottom indicates the game distance is 0 meters.

```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4 #include <QTimer>
5 #include <QKeyEvent>
6 #define BLOCK01_X_POS 300
7 #define BLOCK01_Y_POS 40
8 #define BLOCK02_X_POS 2300
9 #define BLOCK02_Y_POS 110
10 #define BLOCK03_X_POS 900
11 #define BLOCK03_Y_POS 190
12 #define BLOCK04_X_POS 2800
13 #define BLOCK04_Y_POS 260
14 #define BLOCK05_X_POS 1700
15 #define BLOCK05_Y_POS 310
16 #define BLOCK06_X_POS 1500
17 #define BLOCK06_Y_POS 340
18 /*
19 * game_status:
20 * 0 => init
21 * 1 => playing
22 * 2 => pause
23 * 3 => timeout
24 * 4 => die
25 */
26
27 MainWindow::MainWindow(QWidget *parent): QMainWindow(parent), ui(new Ui::MainWindow) { ... }
28
29 MainWindow::~MainWindow() { ... }
30
31 void MainWindow::update_time() { ... }
32
33 void MainWindow::game_stop() { ... }
34
35 void MainWindow::update_object() { ... }
36
37 void MainWindow::move_car() { ... }
38
39 void MainWindow::on_button_start_game_clicked() { ... }
40
41 void MainWindow::game_start() { ... }
42

```

## 24. 撰寫以下程式碼，當背景移動時，也移動障礙物的位置。

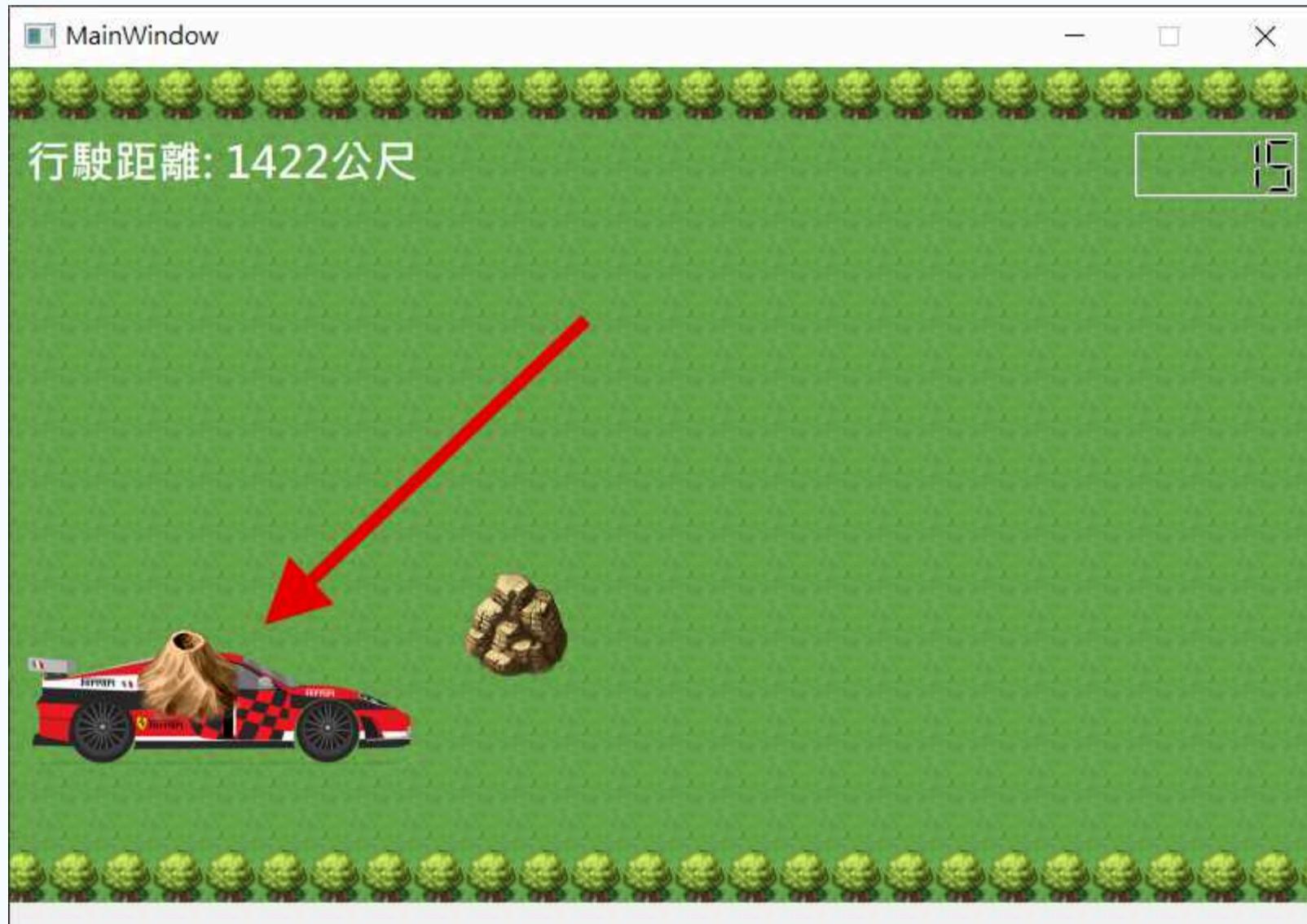


```

56 void MainWindow::update_time() { ... }
57 void MainWindow::game_stop() { ... }
58 void MainWindow::update_object() {
59     bgm_pos -= 1; // 背景位置每次往左移動1
60
61     ui->background->setGeometry(QRect(bgm_pos, 0, 4800, 512)); // 設定背景位置
62     if(bgm_pos == -32) {
63         // 當移動一圈，背景從頭顯示
64         bgm_pos = 0;
65     }
66
67     if(game_status == 1) {
68         // playing
69
70         car_distance += 1; // 增加移動距離
71         ui->label_distance->setText("行駛距離: " + QString::number(car_distance, 'f', 0) + "公尺"); // 顯示移動距離
72         move_car(); // 呼叫 move_car() 移動車子的位置
73         move_blocks(); // 呼叫 move_blocks() 移動障礙物
74     }
75 }
76
77 void MainWindow::move_car() { ... }
78
79 void MainWindow::on_button_start_game_clicked() { ... }
80
81 void MainWindow::game_start() { ... }
82
83 void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
84
85 void MainWindow::move_blocks() {
86     ui->block_01->setGeometry(QRect(BLOCK01_X_POS - car_distance, BLOCK01_Y_POS, 64, 64));
87     ui->block_02->setGeometry(QRect(BLOCK02_X_POS - car_distance, BLOCK02_Y_POS, 64, 64));
88     ui->block_03->setGeometry(QRect(BLOCK03_X_POS - car_distance, BLOCK03_Y_POS, 64, 64));
89     ui->block_04->setGeometry(QRect(BLOCK04_X_POS - car_distance, BLOCK04_Y_POS, 64, 64));
90     ui->block_05->setGeometry(QRect(BLOCK05_X_POS - car_distance, BLOCK05_Y_POS, 64, 64));
91     ui->block_06->setGeometry(QRect(BLOCK06_X_POS - car_distance, BLOCK06_Y_POS, 64, 64));
92 }
93
94
95

```

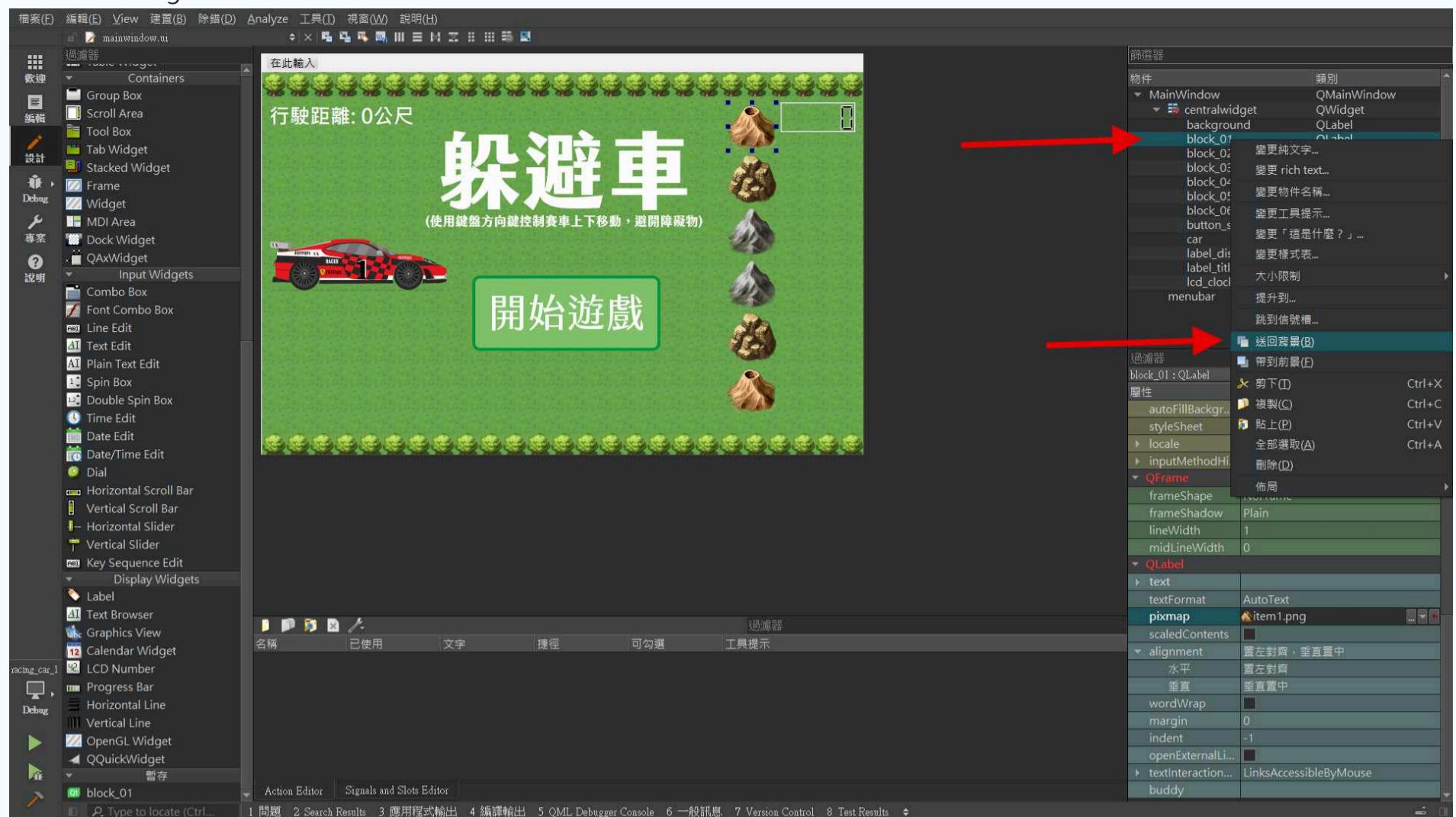
## 25. 建置並執行，可看到障礙物已可移動，但發現會有障礙物可壓在車子上方的情況。



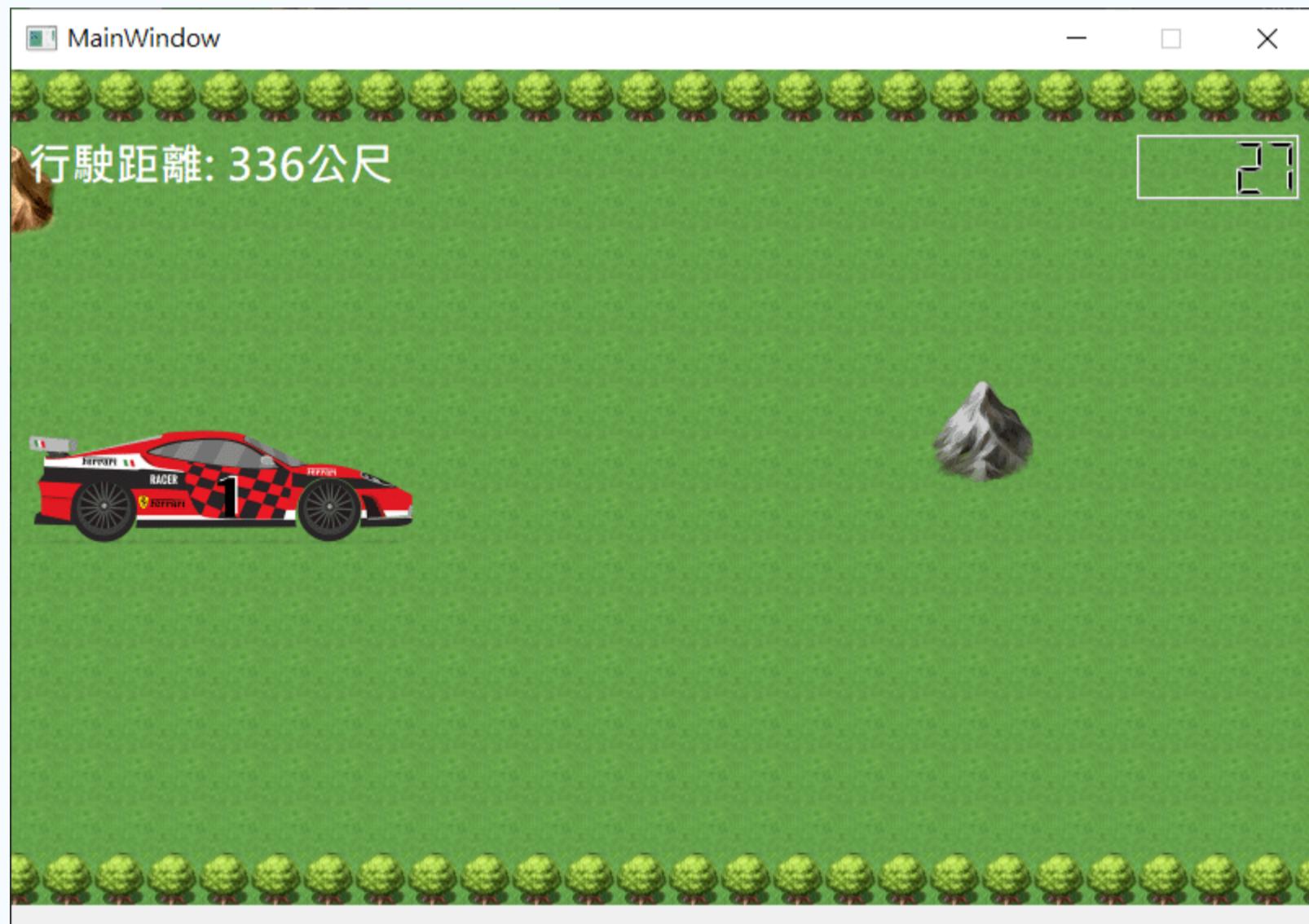
## 26. 原因是因為圖層的問題，障礙物位於比車子高的圖層。

到 ui 設計的地方，在障礙物上按滑鼠右鍵，點選【送回背景】，將所有障礙物移至圖層的最下層。

最後，將 background 的背景，點選【送回背景】，將背景回歸至最下層的狀態。



27. 建置並執行，可看到車子已不會被障礙物所覆蓋。



## 28. 接下來，偵測車子是否碰撞到障礙物，當碰撞到障礙物時，則遊戲結束。

```

namespace Ui { class MainWindow; }

QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

protected:
    void keyPressEvent(QKeyEvent *);

private slots:
    void update_time(); // 遊戲時間更新
    void update_object(); // 遊戲物體移動更新

    void on_button_start_game_clicked();

private:
    Ui::MainWindow *ui;
    int time; // 遊戲時間
    int game_status; // 遊戲狀態
    int bgm_pos; // 背景的位置(水平)
    int car_pos; // 車子的位置(垂直)
    int car_direction; // 車子的方向(垂直)
    float car_distance; // 車子移動的距離

    QTimer *clock_timer; // 遊戲時間計時器
    QTimer *object_timer; // 物體移動計時器

    void game_start(); // 遊戲開始
    void game_pause(); // 遊戲暫停
    void game_stop(); // 遊戲停止
    void move_car(); // 車子移動
    void move_blocks(); // 障礙物移動
    void detect_blocks(); // 偵測障礙物位置
    bool is_collision(int x, int y); // 判斷是否碰撞到障礙物
};

#endif // MAINWINDOW_H

```

## 29. 切換至原始檔，針對每個障礙物撰寫偵測障礙物的程式碼。

```

void MainWindow::detect_blocks() {
    int tmp_padding_x, tmp_padding_y; // 障礙物與車子的距離

    tmp_padding_x = BLOCK01_X_POS - car_distance;
    tmp_padding_y = BLOCK01_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }

    tmp_padding_x = BLOCK02_X_POS - car_distance;
    tmp_padding_y = BLOCK02_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }

    tmp_padding_x = BLOCK03_X_POS - car_distance;
    tmp_padding_y = BLOCK03_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }

    tmp_padding_x = BLOCK04_X_POS - car_distance;
    tmp_padding_y = BLOCK04_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }

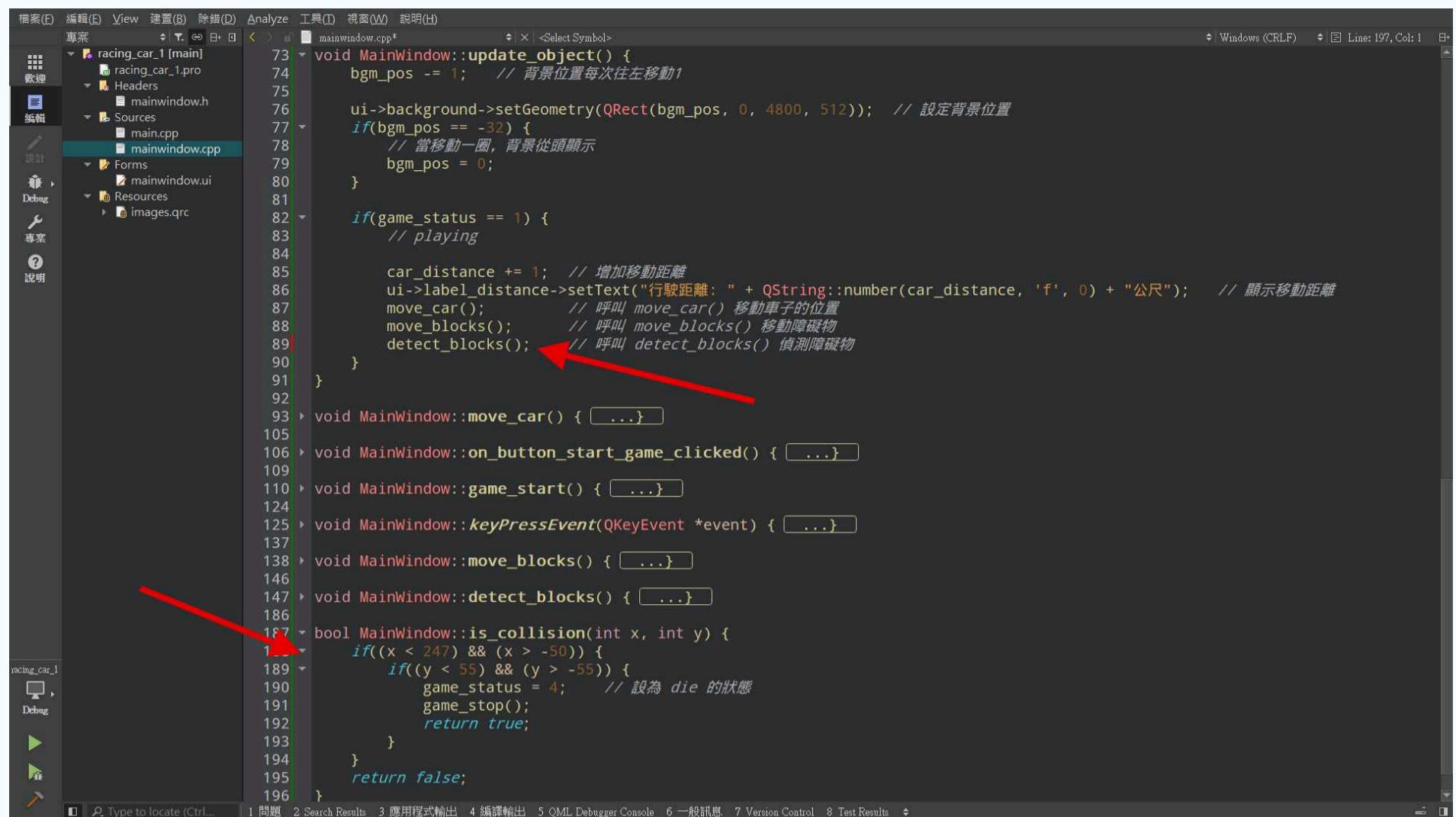
    tmp_padding_x = BLOCK05_X_POS - car_distance;
    tmp_padding_y = BLOCK05_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }

    tmp_padding_x = BLOCK06_X_POS - car_distance;
    tmp_padding_y = BLOCK06_Y_POS - ui->car->y();
    if(is_collision(tmp_padding_x, tmp_padding_y)) {
        return;
    }
}

bool MainWindow::is_collision(int x, int y) { ... }

```

30. 撰寫以下程式碼。並讓障礙物每次移動時，偵測是否被車子碰撞。



```

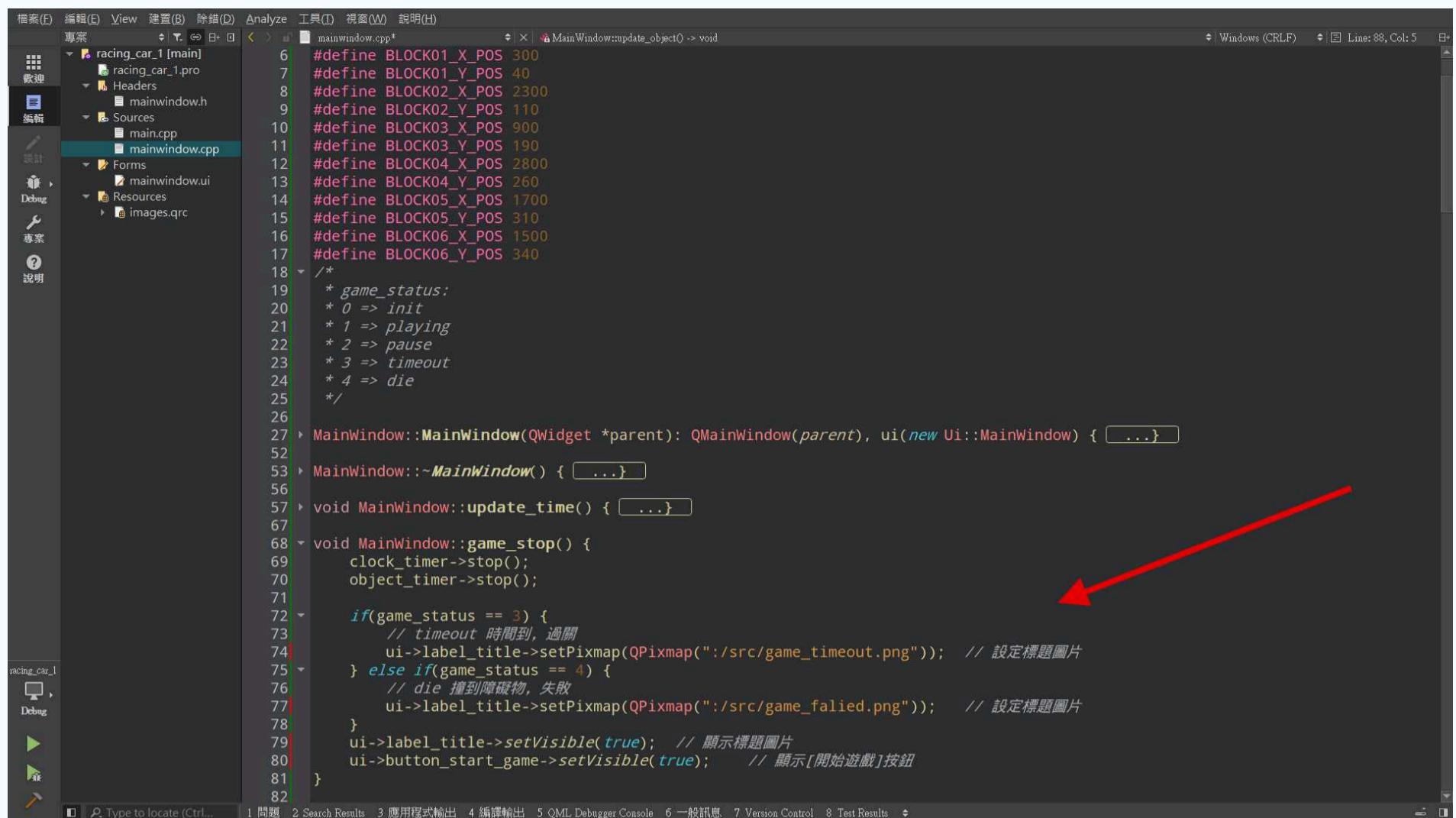
MainWindow.cpp
1 void MainWindow::update_object() {
2     bgm_pos -= 1; // 背景位置每次往左移動1
3
4     ui->background->setGeometry(QRect(bgm_pos, 0, 4800, 512)); // 設定背景位置
5
6     if(bgm_pos == -32) {
7         // 當移動一圈，背景從頭顯示
8         bgm_pos = 0;
9     }
10
11     if(game_status == 1) {
12         // playing
13
14         car_distance += 1; // 增加移動距離
15         ui->label_distance->setText("行駛距離: " + QString::number(car_distance, 'f', 0) + "公尺"); // 顯示移動距離
16         move_car(); // 呼叫 move_car() 移動車子的位置
17         move_blocks(); // 呼叫 move_blocks() 移動障礙物
18         detect_blocks(); // 呼叫 detect_blocks() 偵測障礙物
19     }
20
21 }
22
23 void MainWindow::move_car() { ... }
24
25 void MainWindow::on_button_start_game_clicked() { ... }
26
27 void MainWindow::game_start() { ... }
28
29 void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
30
31 void MainWindow::move_blocks() { ... }
32
33 void MainWindow::detect_blocks() { ... }
34
35 bool MainWindow::is_collision(int x, int y) {
36     if((x < 247) && (x > -50)) {
37         if((y < 55) && (y > -55)) {
38             game_status = 4; // 設為 die 的狀態
39             game_stop();
40             return true;
41         }
42     }
43     return false;
44 }

```

31. 建置並執行，可看到車子碰撞到障礙物時，遊戲結束。



32. 當遊戲結束時顯示【遊戲結束】，並顯示【開始遊戲】的按鈕。



```

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow) { ... }

MainWindow::~MainWindow() { ... }

void MainWindow::update_time() { ... }

void MainWindow::game_stop() {
    clock_timer->stop();
    object_timer->stop();

    if(game_status == 3) {
        // timeout 時間到, 過關
        ui->label_title->setPixmap(QPixmap(":/src/game_timeout.png")); // 設定標題圖片
    } else if(game_status == 4) {
        // die 撞到障礙物, 失敗
        ui->label_title->setPixmap(QPixmap(":/src/game_failed.png")); // 設定標題圖片
    }
    ui->label_title->setVisible(true); // 顯示標題圖片
    ui->button_start_game->setVisible(true); // 顯示[開始遊戲]按鈕
}

```

33. 建置並執行，可看到當遊戲結束後，會顯示此畫面。



34. 當遊戲再次開始時，啟動物體移動計時器，但由於當成是一開啟時，物體移動計時器就已經被開啟，再次開啟會導致發生錯誤，因此用以下程式碼判斷，當物體移動計時器未被開啟，則將它開啟。



```

/mainwindow.cpp*
53 > MainWindow::~MainWindow() { ... }
56
57 > void MainWindow::update_time() { ... }
67
68 > void MainWindow::game_stop() { ... }
82
83 > void MainWindow::update_object() { ... }
102
103 > void MainWindow::move_car() { ... }
115
116 > void MainWindow::on_button_start_game_clicked() { ... }
119
120 > void MainWindow::game_start() {
    game_status = 1; // 設為 playing 的狀態
121
123     ui->label_title->setVisible(false); // 隱藏遊戲標題
124     ui->button_start_game->setVisible(false); // 隱藏開始遊戲按鈕
125
126     clock_timer->start(1000); // 每隔1000毫秒更新一次
127
128     time = 30; // 重設遊戲時間
129     ui->lcd_clock->display(time); // 顯示遊戲時間
130
131     car_pos = 220; // 重設車子的垂直位置(位於中央)
132     car_distance = 0; // 重設移動距離
133     car_direction = 0; // 重設車子移動方向(垂直)
134
135     bgm_pos = 0; // 重設背景位置
136
137     if(!object_timer->isActive()) {
138         // 若物體移動計時器未啟動，則將它啟動
139         object_timer->start(10);
140     }
141 }
142
143 > void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
155
156 > void MainWindow::move_blocks() { ... }
164
165 > void MainWindow::detect_blocks() { ... }
204
205 > bool MainWindow::is_collision(int x, int y) {

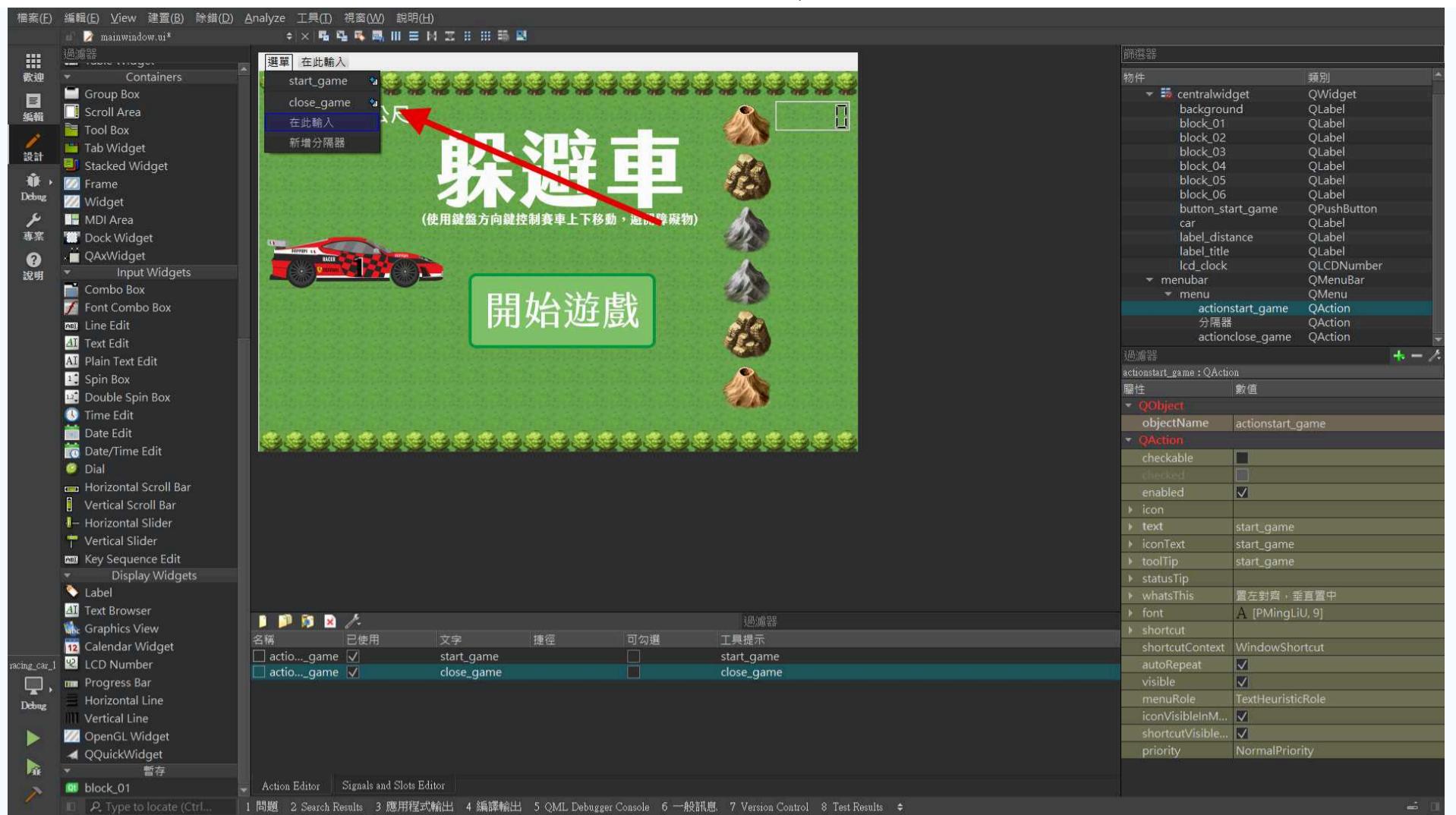
```

35. 接下來，加入選單的部分，切換到 ui 設計，點選上方的選單。

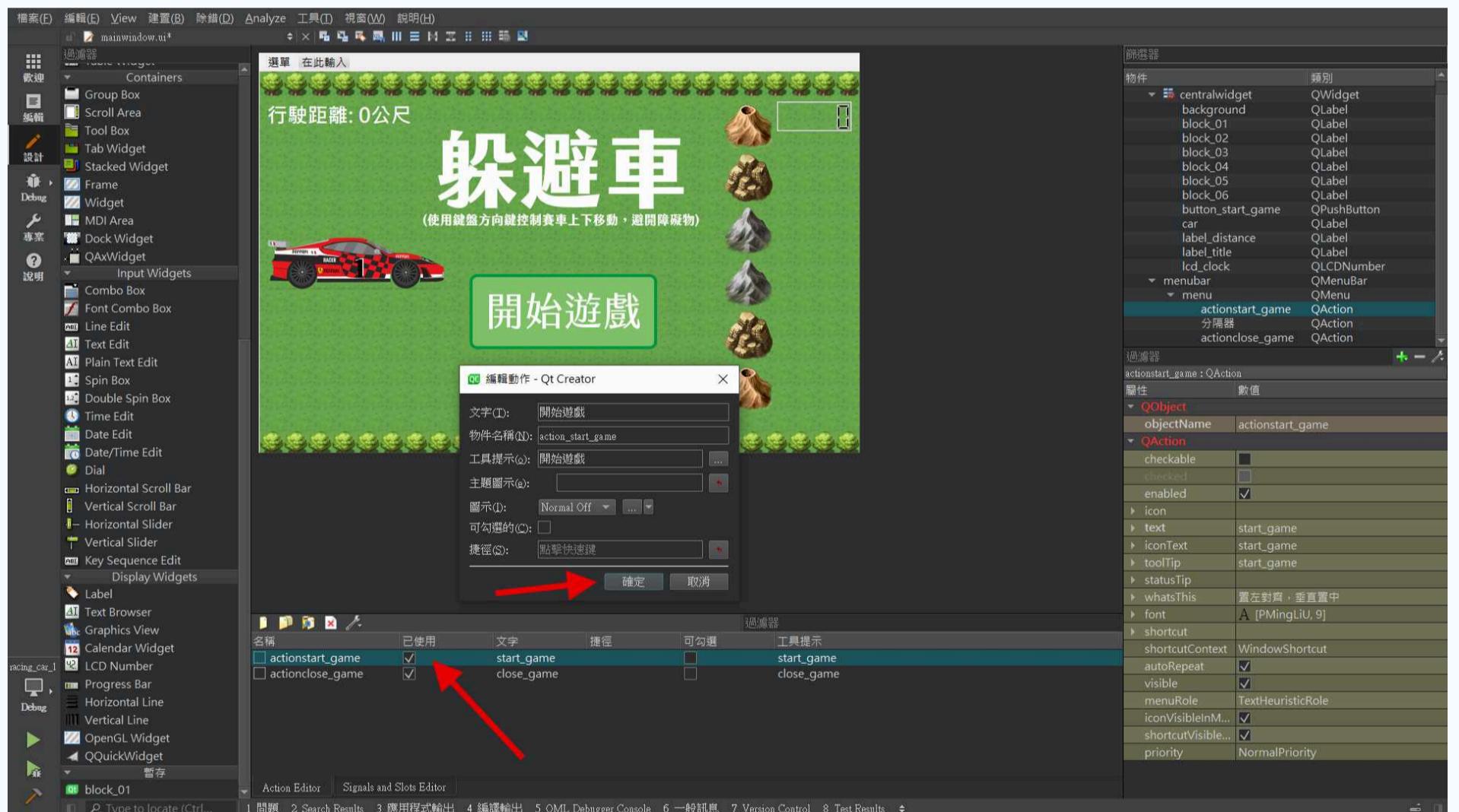


36. 在此處輸入選單的選項，在此請先以英文做輸入，此輸入的值會作為函式的名稱(下一步驟再將其修改為中文)。

(在此示範加入兩個選單，且中間加入一個分隔器)



37. 點選下方的選單，調整選單的設定。



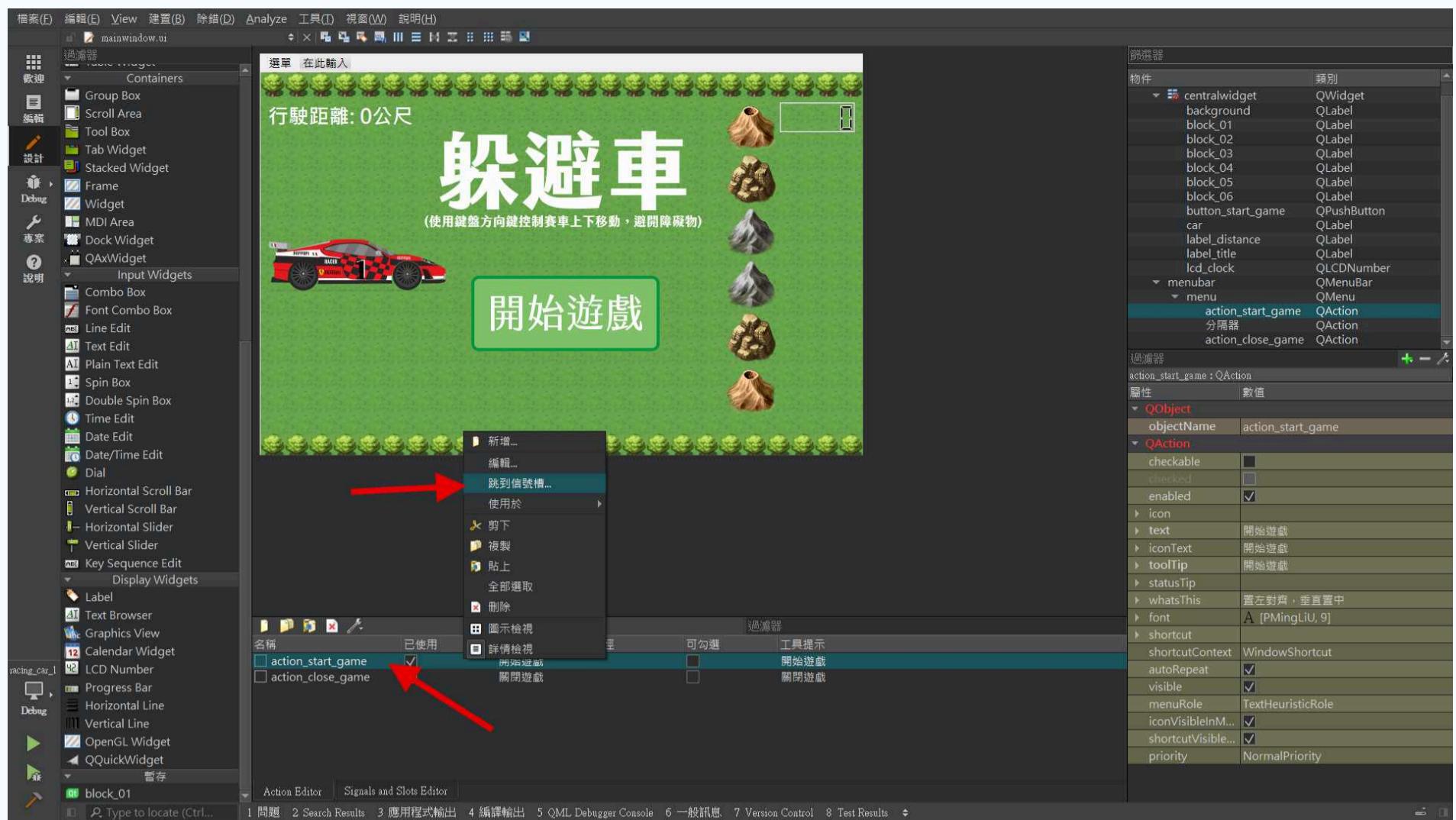
## 38. 選單的設定值如下圖所示。



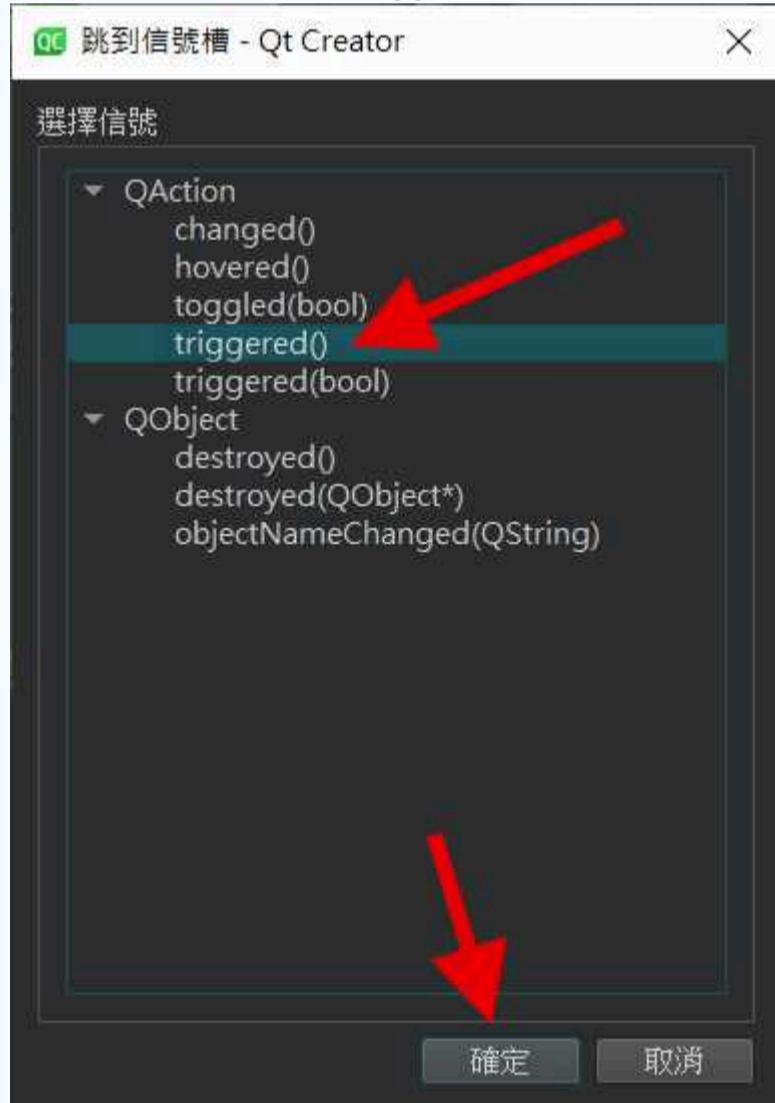
## 39. 建置並執行，可看到選單已顯示在畫面上。



40. 接下來撰寫選單被點觸發時所要執行的事件，在選單上點選滑鼠右鍵，點選【跳到信號槽...】。



41. 選擇【QAction】中的【triggered()】訊號。



## 42. 撰寫當選單被點選時，所要執行的程式碼。

```

52 > MainWindow::~MainWindow() { ... }
53 > void MainWindow::update_time() { ... }
54 > void MainWindow::game_stop() { ... }
55 > void MainWindow::update_object() { ... }
56 > void MainWindow::move_car() { ... }
57 > void MainWindow::on_button_start_game_clicked() { ... }
58 > void MainWindow::game_start() { ... }
59 > void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
60 > void MainWindow::move_blocks() { ... }
61 > void MainWindow::detect_blocks() { ... }
62 > bool MainWindow::is_collision(int x, int y) { ... }
63 > void MainWindow::on_action_start_game_triggered() {
64     switch (game_status) {
65         case 0: // init
66         case 3: // timeout
67         case 4: // die
68             game_start();
69             break;
70         case 1: // playing
71         case 2: // pause
72             game_pause();
73             break;
74     }
75 }
76 > void MainWindow::on_action_close_game_triggered() {
77     close();
78 }
79

```

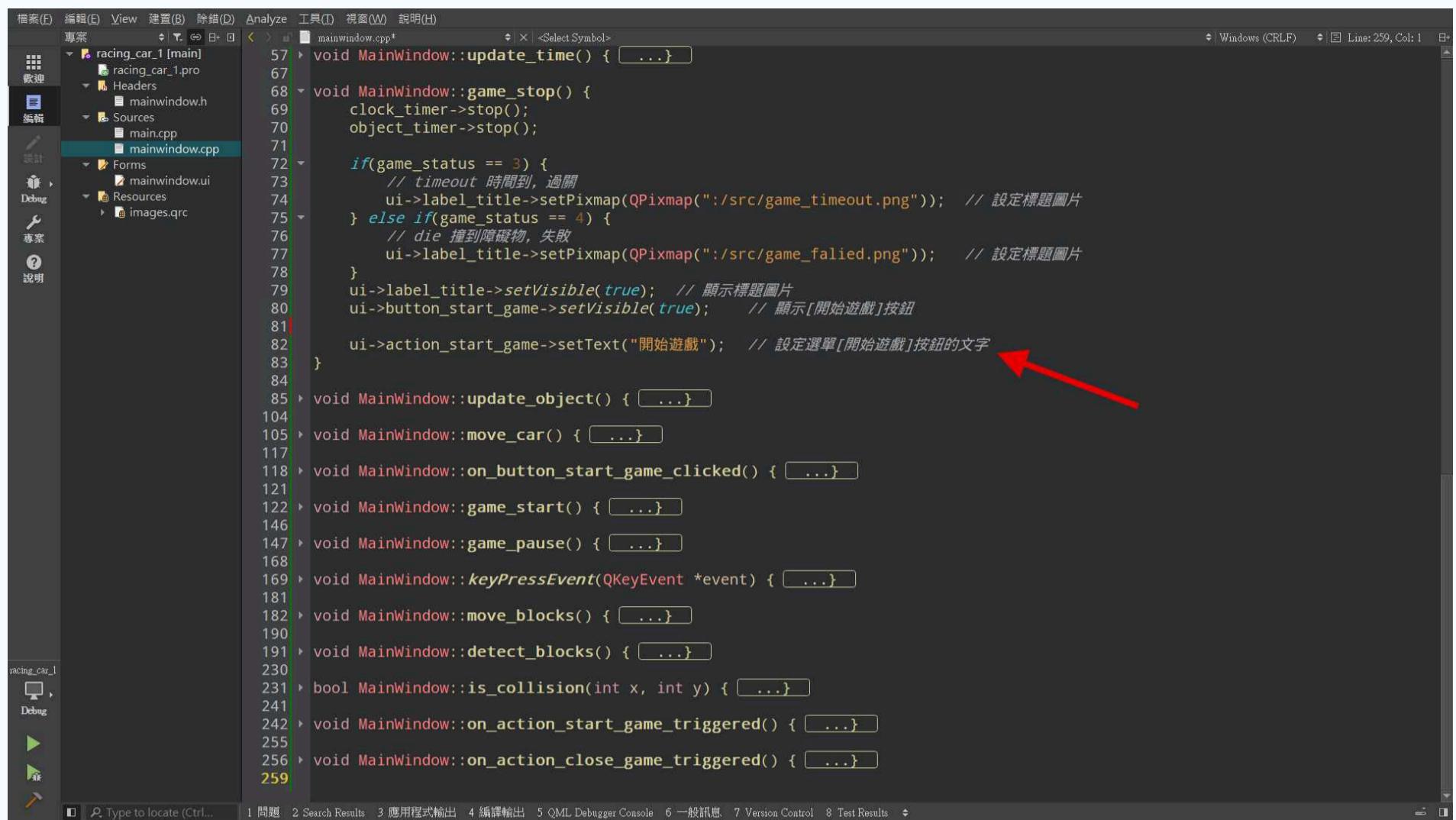
## 43. 遊戲暫停的函式有兩個狀態，在變更遊戲狀態時，要調整選單的文字。

```

104 > void MainWindow::move_car() { ... }
105 > void MainWindow::on_button_start_game_clicked() { ... }
106 > void MainWindow::game_start() { ... }
107 > void MainWindow::game_pause() {
108     if(game_status == 1) {
109         // playing 變為 pause
110         game_status = 2; // 設為 pause 的狀態
111
112         clock_timer->stop(); // 停止時間計時器
113         object_timer->stop(); // 停止物體移動計時器
114         ui->label_title->setPixmap(QPixmap(":/src/game_pause.png")); // 設定標題圖片
115         ui->label_title->setVisible(true); // 顯示標題圖片
116         ui->action_start_game->setText("開始遊戲"); // 設定選單[開始遊戲]按鈕的文字
117
118     } else if(game_status == 2) {
119         // pause 變為 playing
120         game_status = 1; // 設為 playing 的狀態
121
122         clock_timer->start(1000); // 開始時間計時器
123         object_timer->start(10); // 開始物體移動計時器
124         ui->action_start_game->setText("暫停遊戲"); // 設定選單[開始遊戲]按鈕的文字
125         ui->label_title->setVisible(false); // 顯示標題圖片
126     }
127 }
128 > void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
129 > void MainWindow::move_blocks() { ... }
130 > void MainWindow::detect_blocks() { ... }
131 > bool MainWindow::is_collision(int x, int y) { ... }
132 > void MainWindow::on_action_start_game_triggered() { ... }
133 > void MainWindow::on_action_close_game_triggered() { ... }
134

```

#### 44. 在遊戲停止時也是要變更選單的文字。

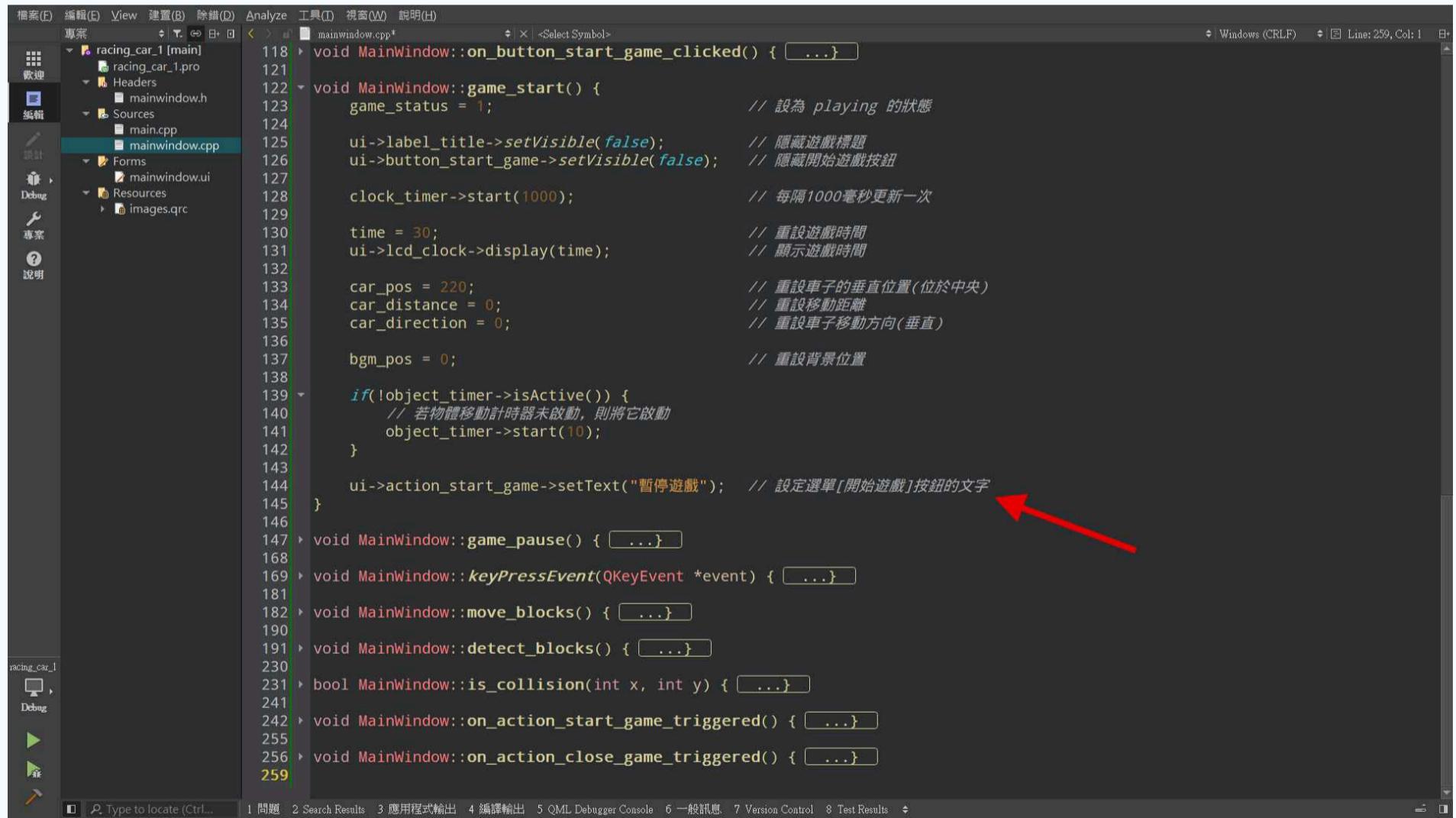


```

mainwindow.cpp*
57 void MainWindow::update_time() { ... }
58
59 void MainWindow::game_stop() {
60     clock_timer->stop();
61     object_timer->stop();
62
63     if(game_status == 3) {
64         // timeout 時間到，過關
65         ui->label_title->setPixmap(QPixmap(":/src/game_timeout.png")); // 設定標題圖片
66     } else if(game_status == 4) {
67         // die 撞到障礙物，失敗
68         ui->label_title->setPixmap(QPixmap(":/src/game_failed.png")); // 設定標題圖片
69     }
70     ui->label_title->setVisible(true); // 顯示標題圖片
71     ui->button_start_game->setVisible(true); // 顯示[開始遊戲]按鈕
72
73     ui->action_start_game->setText("開始遊戲"); // 設定選單[開始遊戲]按鈕的文字
74 }
75
76 void MainWindow::update_object() { ... }
77
78 void MainWindow::move_car() { ... }
79
80 void MainWindow::on_button_start_game_clicked() { ... }
81
82 void MainWindow::game_start() { ... }
83
84 void MainWindow::game_pause() { ... }
85
86 void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
87
88 void MainWindow::move_blocks() { ... }
89
90 void MainWindow::detect_blocks() { ... }
91
92 bool MainWindow::is_collision(int x, int y) { ... }
93
94 void MainWindow::on_action_start_game_triggered() { ... }
95
96 void MainWindow::on_action_close_game_triggered() { ... }
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259

```

#### 45. 遊戲開始時也是如此。



```

mainwindow.cpp*
118 void MainWindow::on_button_start_game_clicked() { ... }
119
120 void MainWindow::game_start() {
121     game_status = 1; // 設為 playing 的狀態
122
123     ui->label_title->setVisible(false); // 隱藏遊戲標題
124     ui->button_start_game->setVisible(false); // 隱藏開始遊戲按鈕
125
126     clock_timer->start(1000); // 每隔1000毫秒更新一次
127
128     time = 30; // 重設遊戲時間
129     ui->lcd_clock->display(time); // 顯示遊戲時間
130
131     car_pos = 220; // 重設車子的垂直位置(位於中央)
132     car_distance = 0; // 重設移動距離
133     car_direction = 0; // 重設車子移動方向(垂直)
134
135     bgm_pos = 0; // 重設背景位置
136
137     if(!object_timer->isActive()) {
138         // 若物體移動計時器未啟動，則將它啟動
139         object_timer->start(10);
140     }
141
142     ui->action_start_game->setText("暫停遊戲"); // 設定選單[開始遊戲]按鈕的文字
143 }
144
145 void MainWindow::game_pause() { ... }
146
147 void MainWindow::keyPressEvent(QKeyEvent *event) { ... }
148
149 void MainWindow::move_blocks() { ... }
150
151 void MainWindow::detect_blocks() { ... }
152
153 bool MainWindow::is_collision(int x, int y) { ... }
154
155 void MainWindow::on_action_start_game_triggered() { ... }
156
157 void MainWindow::on_action_close_game_triggered() { ... }
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259

```

## 46. 建置並執行，點選選單中的選項，測試選單的運作。



## 47. 接下來，加入按下鍵盤空白鍵可讓遊戲暫停的程式碼。

Screenshot of Qt Creator showing the code editor with the file `mainwindow.cpp` open. A red arrow points to the line of code where the key press event for the spacebar is handled.

```

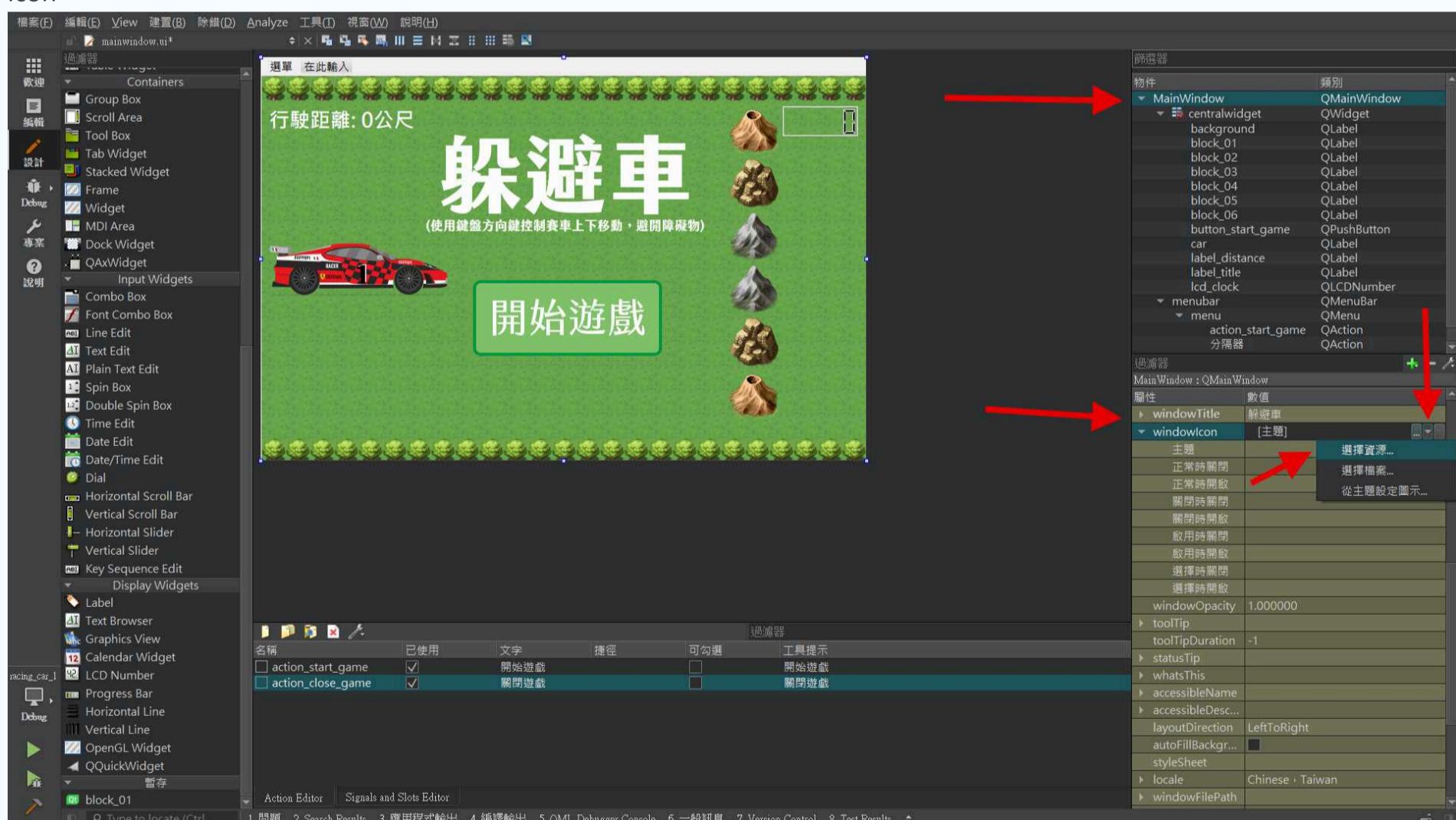
void MainWindow::keyPressEvent(QKeyEvent *event) {
    switch (event->key()) {
        case Qt::Key_Up:
            car_direction = -1;
            // qDebug() << "up";
            break;
        case Qt::Key_Down:
            car_direction = 1;
            // qDebug() << "down";
            break;
        case Qt::Key_Space:
            game_pause();
            // qDebug() << "space";
            break;
    }
}

```

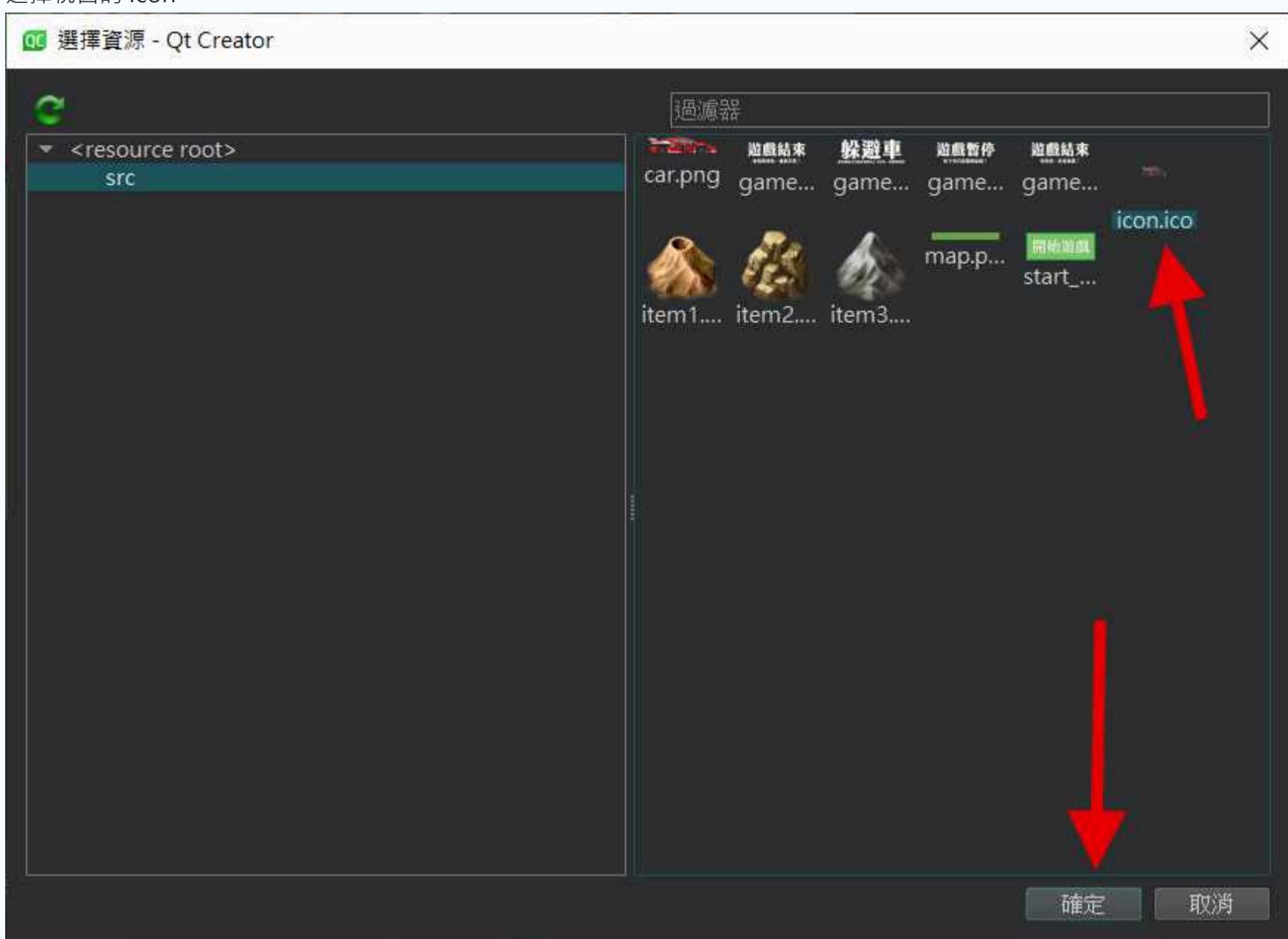
## 48. 最後，修改視窗的標題文字。



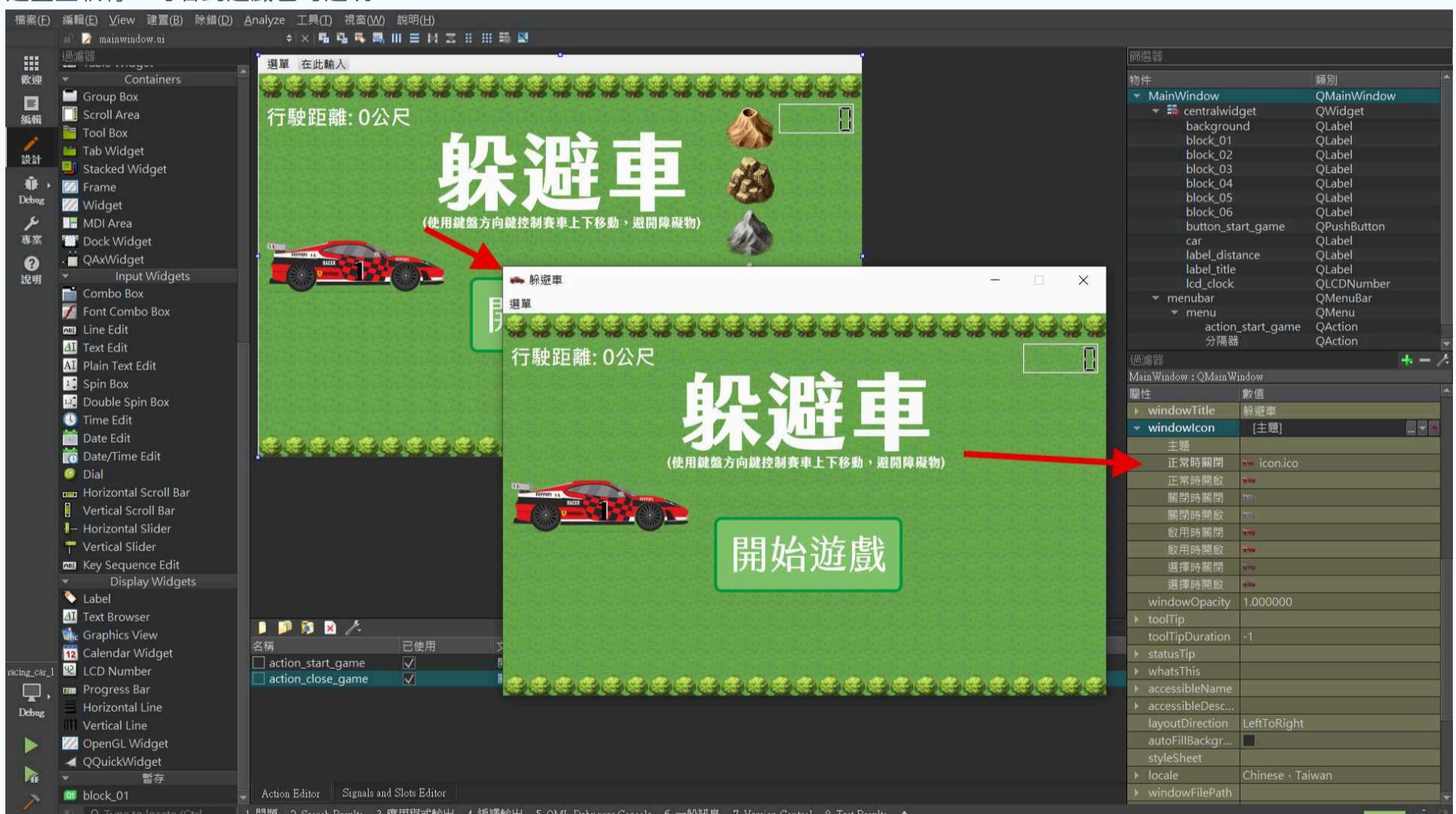
49. 到 ui 設計中選擇【MainWindow】，調整【windowTitle】可變更程式視窗的標題文字，調整【windowIcon】可調整程式視窗的 icon。



## 50. 選擇視窗的 icon。



## 51. 建置並執行，可看到遊戲已可遊玩。



52. 恭喜您已遊戲的設計，接下來請閱讀下一篇文章【[跨平台圖形化程式開發 - \(4\) 賽車遊戲開發-發布 Windows 程式](#)】將程式發布。

建立時間：2021/5/26 PM 2:11

修改時間：2021/5/28 AM 10:52

作者: Linwebs

標籤