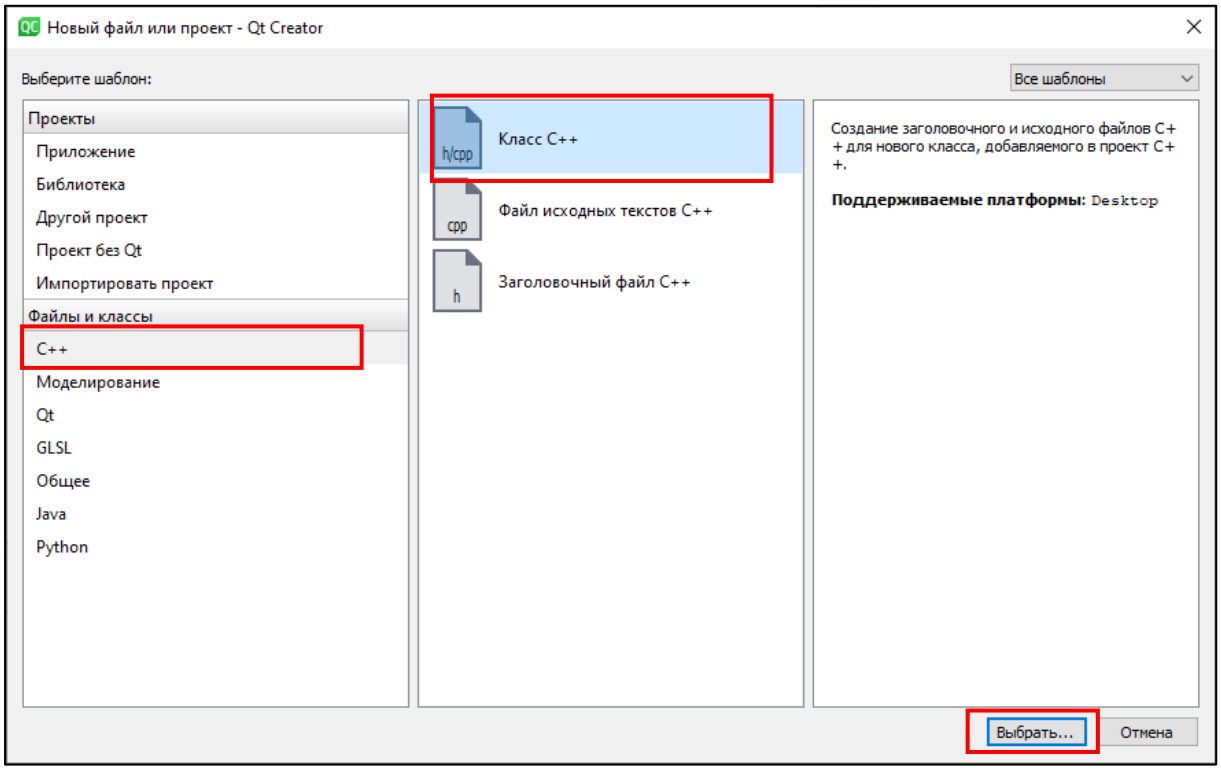
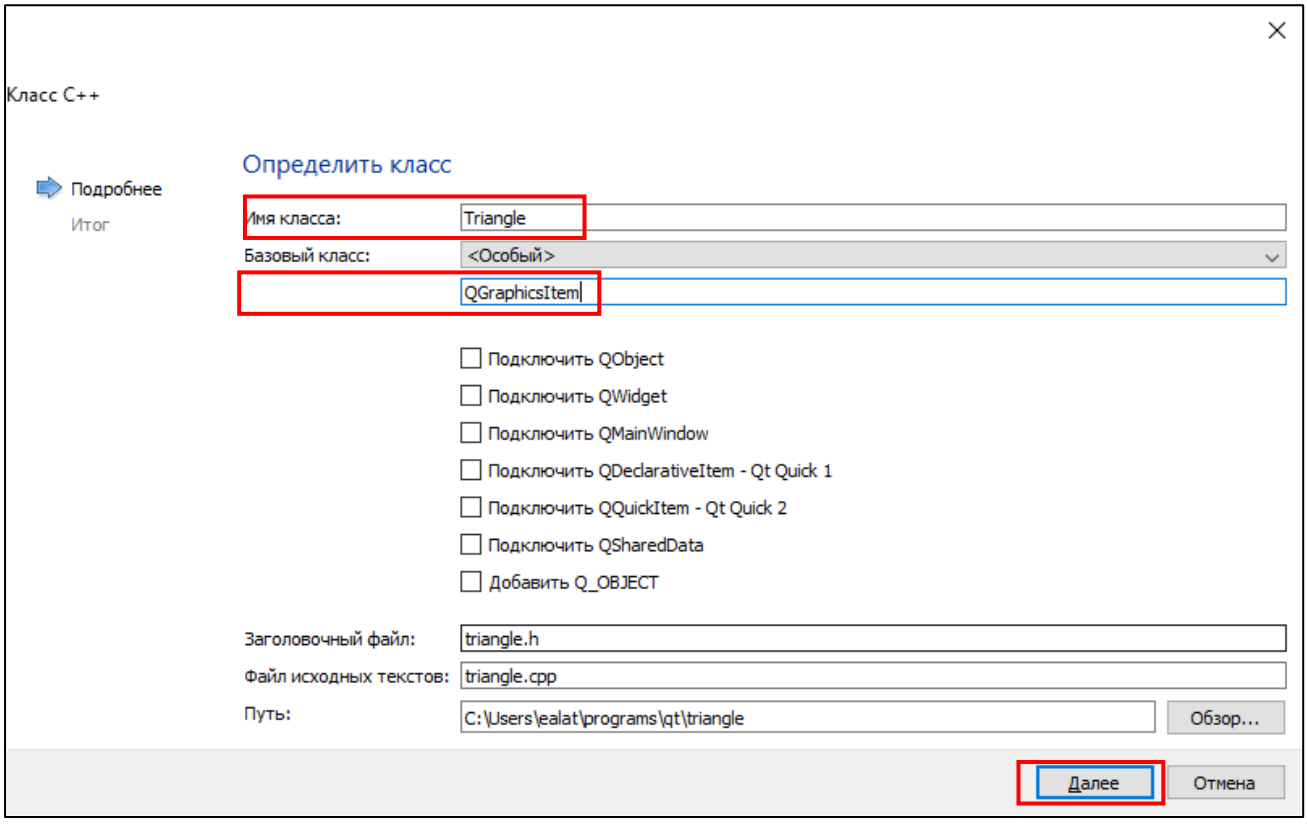
Рисование в Qt

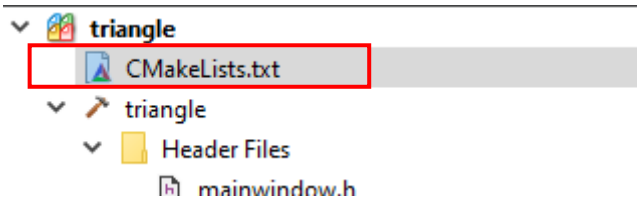
1. Создайте новый проект в Qt Creator.
2. Добавьте на форму компонент Graphics View и установите для него значение ширины 580 и высоты 580. Форму при необходимости можно тоже увеличить. Также можно удалить statusBar с формы.
3. Добавьте в проект новый файл. Это можно сделать через меню Файл «Создать файл или проект…», в появившемся окне выбрать опции как показано на рисунке и затем нажать кнопку «Выбрать…».



4. В появившемся окне введите имя класса Triangle, базовый класс оставьте без изменения, а в следующей строчке напишите QGraphicsItem. Нажмите «Далее».



5. В появившемся окне нажмите «Завершить».
6. Если файлы не появились слева в списке всех файлов, необходимо открыть файл CMakeLists.txt



7. Затем добавить в него следующее (скорее всего, это потребуется сделать только в более свежих версиях, например, если вы захотите установить Qt на домашнем компьютере):

```
if(ANDROID)
  add_library(triangle SHARED
    main.cpp
    mainwindow.cpp
    mainwindow.h
    mainwindow.ui
  )
else()
  add_executable(triangle
    main.cpp
    mainwindow.cpp
    mainwindow.h
    mainwindow.ui
    triangle.cpp
    triangle.h
  )
endif()
```

8. Теперь нужно добавить программный код. В файл **MainWindow.h** необходимо дописать в верхней части

```
#include <QGraphicsScene>
#include <triangle.h>
```

Также необходимо объявить графическую сцену и треугольник как приватные члены класса:

```
QGraphicsScene  *scene;     // Объявляем графическую сцену
Triangle        *triangle;  // и треугольник
```

В итоге файл должен выглядеть примерно так (у вас может быть немного по-другому, главное, не удаляйте никаких строчек из кода и не пишите лишнего).

```h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>

#include <triangle.h>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private:
    Ui::MainWindow *ui;
    QGraphicsScene  *scene;     // Объявляем графическую сцену
    Triangle        *triangle;  // и треугольник
};
#endif // MAINWINDOW_H
```

9. Перейдем к файлу **triangle.h**. Далее нужно добавить несколько строчек, чтобы получился такой результат.

```h
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <QGraphicsItem>
#include <QPainter>

class Triangle : public QGraphicsItem
{
public:
    Triangle();
    ~Triangle();
protected:
    /* Определяем метод, возвращающий область, в которой находится треугольник */
    QRectF boundingRect() const;
    /* Определяем метод для отрисовки треугольника */
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);
};

#endif // TRIANGLE_H
```

10. В файле **MainWindow.cpp** в конструкторе дописываем ряд строчек.

```
this->resize(600,600);              // Задаем размеры окна
this->setFixedSize(600,600);        // Фиксируем размеры окна
scene = new QGraphicsScene();       // Инициализируем граф. сцену
triangle = new Triangle();          // Инициализируем треугольник
ui->graphicsView->setScene(scene);
ui->graphicsView->setRenderHint(QPainter::Antialiasing);
ui->graphicsView->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
ui->graphicsView->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
scene->setSceneRect(-250,-250,500,500);
scene->addLine(-250,0,250,0,QPen(Qt::black));
scene->addLine(0,-250,0,250,QPen(Qt::black));
scene->addItem(triangle);
triangle->setPos(0,0);
```

Результат должен выглядеть примерно так:

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

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->resize(600,600);          // Задаем размеры виджета, то есть окна
    this->setFixedSize(600,600);    // Фиксируем размеры виджета

    scene = new QGraphicsScene();   // Инициализируем графическую сцену
    triangle = new Triangle();      // Инициализируем треугольник

    ui->graphicsView->setScene(scene);  // Устанавливаем графическую сцену в graphicsView
    ui->graphicsView->setRenderHint(QPainter::Antialiasing);    // Устанавливаем сглаживание
    ui->graphicsView->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff); // Отключаем скроллбар по вертика
    ui->graphicsView->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff); // Отключаем скроллбар по гориз

    scene->setSceneRect(-250,-250,500,500); // Устанавливаем область графической сцены

    scene->addLine(-250,0,250,0,QPen(Qt::black));   // Добавляем горизонтальную линию через центр
    scene->addLine(0,-250,0,250,QPen(Qt::black));   // Добавляем вертикальную линию через центр

    scene->addItem(triangle);   // Добавляем на сцену треугольник
    triangle->setPos(0,0);      // Устанавливаем треугольник в центр сцены
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

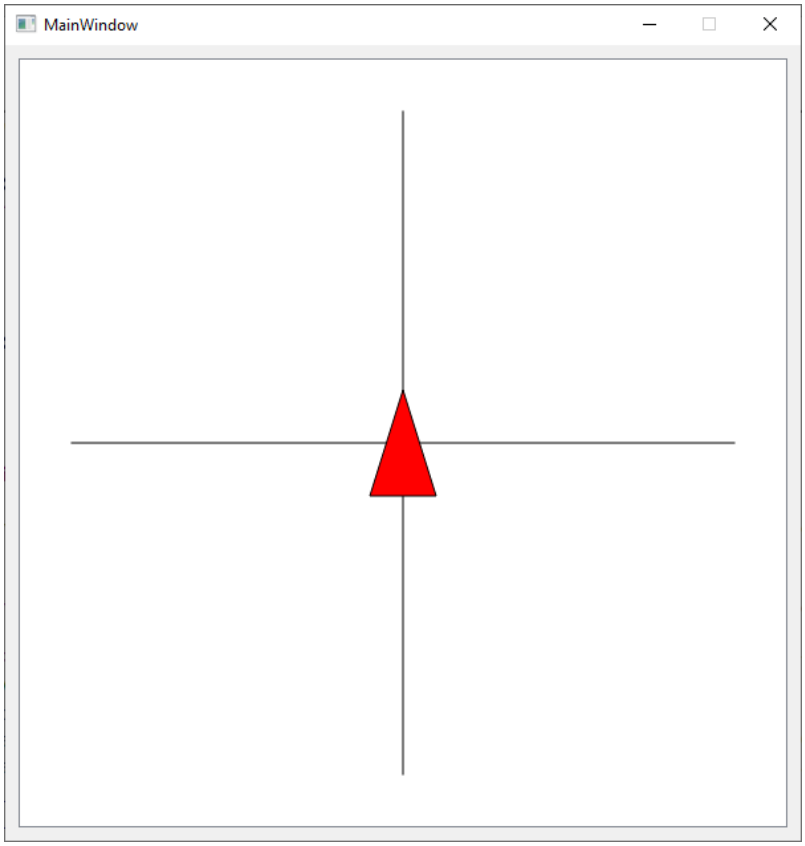
11. Осталось дописать нужный код в **triangle.cpp**.

```
Triangle::Triangle() :
    QGraphicsItem()
{
}
Triangle::~Triangle()
{
}
QRectF Triangle::boundingRect() const
{
    return QRectF(-25,-40,50,80);
}
void Triangle::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
        QPolygon polygon;
        polygon << QPoint(0,-40) << QPoint(25,40) << QPoint(-25,40);
        painter->setBrush(Qt::red);
        painter->drawPolygon(polygon);
        Q_UNUSED(option);
        Q_UNUSED(widget);
}
```

Файл должен выглядеть так

```cpp
#include "triangle.h"

Triangle::Triangle() :
    QGraphicsItem()
{

}

Triangle::~Triangle()
{

}

QRectF Triangle::boundingRect() const
{
    return QRectF(-25,-40,50,80);   // Ограничиваем область, в которой лежит треугольник
}

void Triangle::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
        QPolygon polygon;   // Используем класс полигона, чтобы отрисовать треугольник
        // Помещаем координаты точек в полигональную модель
        polygon << QPoint(0,-40) << QPoint(25,40) << QPoint(-25,40);
        painter->setBrush(Qt::red);     // Устанавливаем кисть, которой будем отрисовывать объект
        painter->drawPolygon(polygon);  // Рисуем треугольник по полигональной модели
        Q_UNUSED(option);
        Q_UNUSED(widget);
}
```

12. Запускаем проект.



13. Добавим возможности управления треугольником. Начнем с файла **MainWindow.h**, в который необходимо добавить таймер и возможности управления с клавиатуры. Для этого в верхней части добавляем

```
#include <QShortcut>
#include <QTimer>
```

В состав приватных членов класса дописываем

```
QTimer          *timer;
```

```cpp
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>
#include <QShortcut>
#include <QTimer>

#include <triangle.h>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private:
    Ui::MainWindow *ui;
    QGraphicsScene  *scene;     // Объявляем графическую сцену
    Triangle        *triangle;  // и треугольник
    QTimer          *timer;
};
#endif // MAINWINDOW_H
```

14. Перейдем к файлу **MainWindow.cpp** и допишем в конце конструктора следующие строчки

```cpp
timer = new QTimer();
connect(timer, &QTimer::timeout, triangle, &Triangle::slotGameTimer);
timer->start(1000 / 50);
scene->addLine(-250,-250, 250,-250, QPen(Qt::black));
scene->addLine(-250, 250, 250, 250, QPen(Qt::black));
scene->addLine(-250,-250,-250, 250, QPen(Qt::black));
scene->addLine( 250,-250, 250, 250, QPen(Qt::black));
```

```cpp
    /* Инициализируем таймер и вызываем слот обработки сигнала таймера
     * у Треугольника 20 раз в секунду.
     * Управляя скоростью отсчётов, соответственно управляем скоростью
     * изменения состояния графической сцены
     * */
    timer = new QTimer();
    connect(timer, &QTimer::timeout, triangle, &Triangle::slotGameTimer);
    timer->start(1000 / 50);
    /* Дополнительно нарисуем органичение территории в графической сцене */
    scene->addLine(-250,-250, 250,-250, QPen(Qt::black));
    scene->addLine(-250, 250, 250, 250, QPen(Qt::black));
    scene->addLine(-250,-250,-250, 250, QPen(Qt::black));
    scene->addLine( 250,-250, 250, 250, QPen(Qt::black));
```

15. Изменим файл **triangle.h** следующим образом

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsScene>

#include <windows.h>

class Triangle : public QObject, public QGraphicsItem
{
    Q_OBJECT
public:
    explicit Triangle(QObject *parent = 0);
    ~Triangle();
public slots:
    void slotGameTimer(); // Слот, который отвечает за обработку перемещения треугольника

protected:
    /* Определяем метод, возвращающий область, в которой находится треугольник */
    QRectF boundingRect() const;
    /* Определяем метод для отрисовки треугольника */
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);

private:
    qreal angle;    // Угол поворота графического объекта
};

#endif // TRIANGLE_H
```

16. Файл **triangle.cpp** нужно будет изменить и дополнить. Необходимо изменить конструктор

```
Triangle::Triangle(QObject *parent) :
    QObject(parent), QGraphicsItem()
{
    angle = 0;     // Задаём угол поворота графического объекта
    setRotation(angle);     // Устанавилваем угол поворота графического объекта
}
```

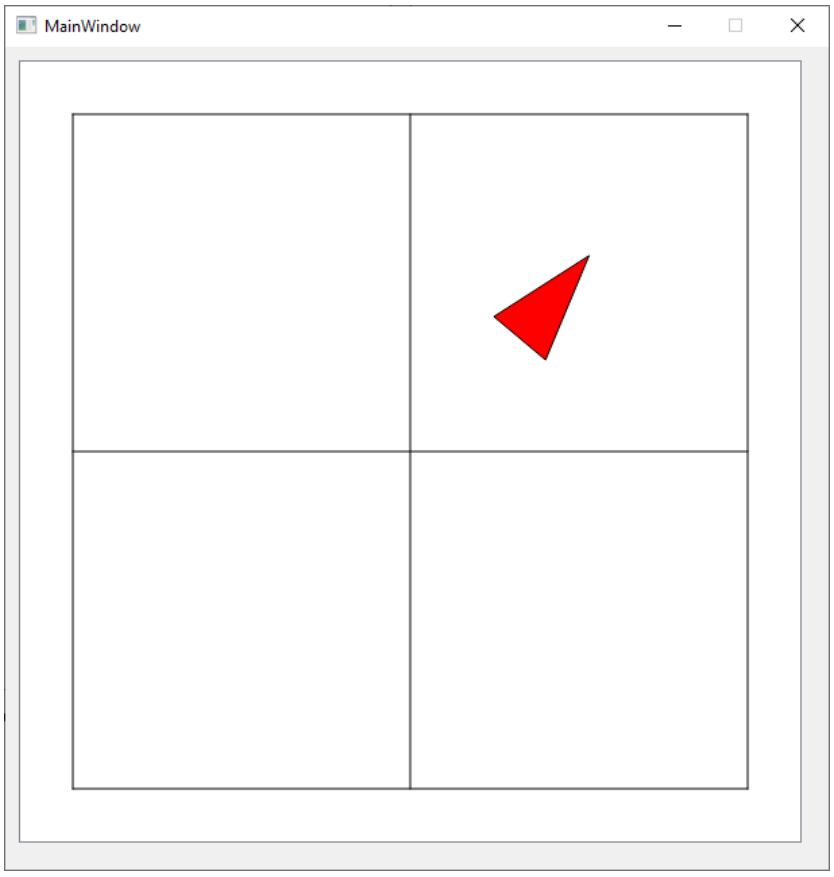
А также добавить метод для управления треугольником slotGameTimer

```
void Triangle::slotGameTimer()
{
    if(GetAsyncKeyState(VK_LEFT)){
        angle -= 10;        // Задаём поворот на 10 градусов влево
        setRotation(angle); // Поворачиваем объект
    }
    if(GetAsyncKeyState(VK_RIGHT)){
        angle += 10;        // Задаём поворот на 10 градусов вправо
        setRotation(angle); // Поворачиваем объект
    }
    if(GetAsyncKeyState(VK_UP)){
        setPos(mapToParent(0, -5));
    }
    if(GetAsyncKeyState(VK_DOWN)){
        setPos(mapToParent(0, 5));
    }
    if(this->x() - 10 < -250){
        this->setX(-240);       // слева
    }
    if(this->x() + 10 > 250){
        this->setX(240);        // справа
    }
    if(this->y() - 10 < -250){
        this->setY(-240);       // сверху
```

```
    }
    if(this->y() + 10 > 250){
        this->setY(240);         // снизу
    }
}
```

```
32  void Triangle::slotGameTimer()
33  {
34      /* Поочерёдно выполняем проверку на нажатие клавиш  */
35      if(GetAsyncKeyState(VK_LEFT)){
36          angle -= 10;         // Задаём поворот на 10 градусов влево
37          setRotation(angle); // Поворачиваем объект
38      }
39      if(GetAsyncKeyState(VK_RIGHT)){
40          angle += 10;         // Задаём поворот на 10 градусов вправо
41          setRotation(angle); // Поворачиваем объект
42      }
43      /* Продвигаем объект на 5 пикселей вперёд, переведя их в координатную систему  графической сцены */
44      if(GetAsyncKeyState(VK_UP)){
45          setPos(mapToParent(0, -5));
46      }
47     /* Продвигаем объект на 5 пикселей назад, переведя их в координатную систему  графической сцены */
48      if(GetAsyncKeyState(VK_DOWN)){
49          setPos(mapToParent(0, 5));
50      }
51      /* Проверка выхода за границы поля. Если объект выходит за заданные границы, возвращаем его назад */
52      if(this->x() - 10 < -250){
53          this->setX(-240);        // слева
54      }
55      if(this->x() + 10 > 250){
56          this->setX(240);         // справа
57      }
58      if(this->y() - 10 < -250){
59          this->setY(-240);        // сверху
60      }
61      if(this->y() + 10 > 250){
62          this->setY(240);         // снизу
63      }
64  }
```

17. Запустите приложение и попробуйте управлять треугольником



18. Попробуйте самостоятельно изменить поле и внешний вид треугольника