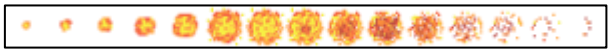
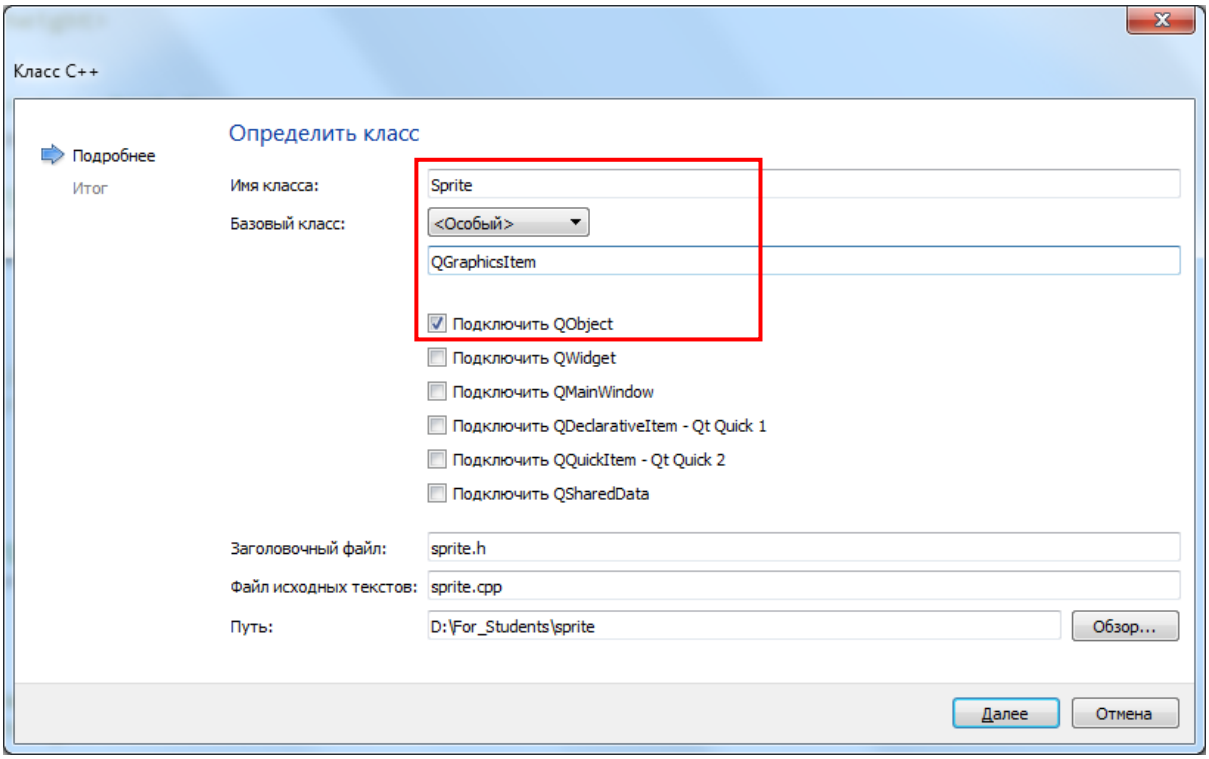
Спрайтовая графика в Qt

Спрайты — небольшие рисунки с изображениями персонажей и других объектов, используемые для создания анимации в приложениях. Анимация осуществляется с помощью перебора изображений из атласа спрайтов. Атлас спрайтов представляет собой набор спрайтов, сохраненных в одном графическом файле.
Небольшие спрайты, как правило, расположены подряд горизонтально.



В атласах спрайтов большего размера изображения часто располагаются в виде таблицы.

1. Создайте новый проект в Qt Creator.
2. Добавьте на форму компонент Graphics View и растяните его по всей форме. Также можно удалить statusBar с формы.
3. Добавьте в проект новый класс (см. предыдущее занятие). Имя класса Sprite, он наследуется от QGraphicsItem. Также нужно подключить QObject.



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

```
#include <QGraphicsScene>
#include <QTimer>
#include <QList>
#include <QPixmap>

#include "sprite.h"
```

Также необходимо объявить графическую сцену как приватный член класса:

```
QGraphicsScene  *scene;     // Объявляем графическую сцену
```

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 <QTimer>
#include <QList>
#include <QPixmap>

#include "sprite.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
    QGraphicsScene *scene;  // Объявляем графическую сцену
};

#endif // MAINWINDOW_H
```

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

```
scene = new QGraphicsScene();
ui->graphicsView->setScene(scene);
scene->addItem(new Sprite());
```

В верхней части файла изменим вторую строчку:

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

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);

    scene = new QGraphicsScene();       // Инициализируем графическую сцену
    ui->graphicsView->setScene(scene);  // Устанавливаем графическую сцену в graphicsView
    scene->addItem(new Sprite());       // Помещаем на сцену новый объект спрайта
}

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

```

6. Изменим содержимое файла **sprite.h**. Сначала подключим дополнительные файлы:

```
#include <QGraphicsItem>
#include <QTimer>
#include <QPixmap>
#include <QPainter>
```

Затем изменим класс, добавив конструктор, деструктор, слот, поля и методы.

```
class Sprite : public QObject, public QGraphicsItem
{
    Q_OBJECT
public:
    explicit Sprite(QObject *parent = 0);
    virtual ~Sprite(){};

private slots:
    void nextFrame();

private:
    void paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget);
    QRectF boundingRect() const;

private:
    QTimer *timer;
    QPixmap *spriteImage;
    int currentFrame;
};
```

```
#ifndef SPRITE_H
#define SPRITE_H

#include <QObject>
#include <QGraphicsItem>
#include <QTimer>
#include <QPixmap>
#include <QPainter>

class Sprite : public QObject, public QGraphicsItem
{
    Q_OBJECT
public:
    explicit Sprite(QObject *parent = 0);
    virtual ~Sprite(){};

private slots:
    void nextFrame();   // Слот для пролистывания изображения в QPixmap

private:
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);
    QRectF boundingRect() const;

private:
    QTimer *timer;      // Таймер для пролистывания изображения в QPixmap
    QPixmap *spriteImage;   // В данный объект QPixamp будет помещён спрайт
    int currentFrame;   // Координата X, с которой начинает очередной кадр спрайта
};

#endif // SPRITE_H
```

7. Осталось дописать нужный код в **sprite.cpp**. Сначала допишем конструктор

```
Sprite::Sprite(QObject *parent) :
    QObject(parent), QGraphicsItem()
{
    currentFrame = 0;
    spriteImage = new QPixmap(":sprite.png");
    timer = new QTimer();
```

```
    connect(timer, &QTimer::timeout, this, &Sprite::nextFrame);
    timer->start(25);
}
```

Отрисовка прямоугольника (вспомогательный метод)

```
QRectF Sprite::boundingRect() const
{
    return QRectF(-10,-10,20,20);
}
```

Отрисовка спрайта. Здесь мы передаем в функцию:

- Координаты X и Y, куда помещается QPixmap (изображение).
- Указатель на QPixmap.
- Координаты в изображении QPixmap, откуда будет отображаться изображение.
- Задавая координату X с помощью переменной currentFrame, мы будем как бы передвигать камеру по спрайту.
- Ширина и высота отображаемого кадра.

```
void Sprite::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    painter->drawPixmap(-10,-10, *spriteImage, currentFrame, 0, 20,20);
    Q_UNUSED(option);
    Q_UNUSED(widget);
}
```

Переход к следующему кадру по сигналу таймера

```
void Sprite::nextFrame()
{
    currentFrame += 20;
    if (currentFrame >= 300 ) currentFrame = 0;
    this->update(-10,-10,20,20);
}
```

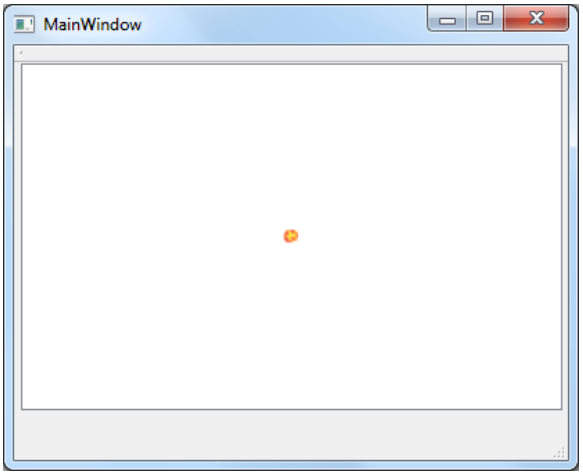
Файл должен выглядеть так

```
#include "sprite.h"

Sprite::Sprite(QObject *parent) :
    QObject(parent), QGraphicsItem()
{
    currentFrame = 0;   // Устанавливаем координату текущего кадра спрайта
    spriteImage = new QPixmap(":sprite.png"); // Загружаем изображение спрайта в QPixmap
    timer = new QTimer();   // Создаём таймер для анимации спрайта
    // Подключаем сигнал от таймера к слоту перелистывания кадров спрайта
    connect(timer, &QTimer::timeout, this, &Sprite::nextFrame);
    timer->start(25);   // Запускаем спрайт на генерацию сигнала с периодичность 25 мс
}

QRectF Sprite::boundingRect() const
{
    return QRectF(-10,-10,20,20);
}

void Sprite::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    /* В отрисовщике графического объекта отрисовываем спрайт*/
    painter->drawPixmap(-10,-10, *spriteImage, currentFrame, 0, 20,20);
    Q_UNUSED(option);
    Q_UNUSED(widget);
}

void Sprite::nextFrame()
{
    /* По сигналу от таймера передвигаем на 20 пикселей точку отрисовки
     * Если currentFrame = 300 то обнуляем его, поскольку размер sprite sheet 300 пикселей на 20
     * */
    currentFrame += 20;
    if (currentFrame >= 300 ) currentFrame = 0;
    this->update(-10,-10,20,20); // и перерисовываем графический объект с новым кадром спрайта
}
```

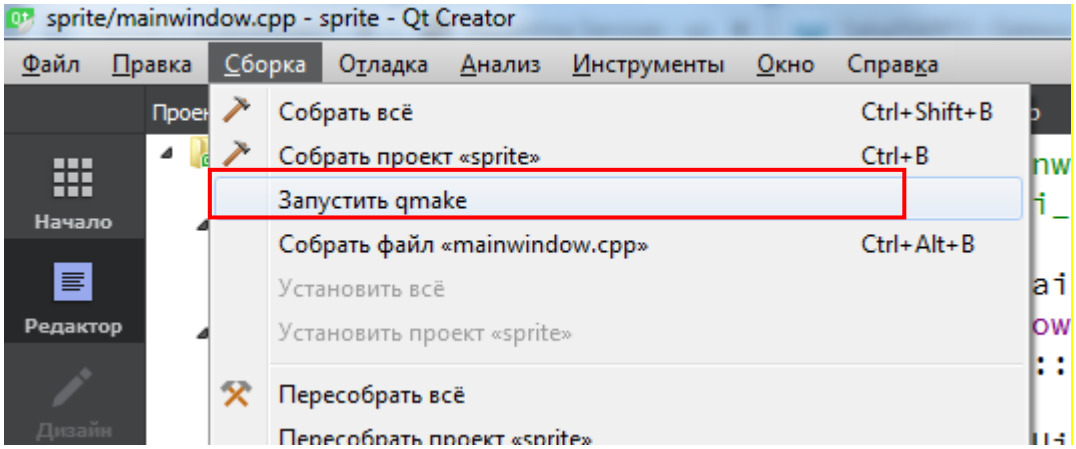
8. Прежде чем запускать проект, нужно добавить изображение в проект. Для этого находим скачанный ранее файл sprite.png и помещаем его в папку с проектом. Далее необходимо подключить его к ресурсам проекта. Для этого нужно открыть файл sprite.pro и дописать в конце
   RESOURCES += sprite.png
   Результат должен выглядеть примерно так

```
26 SOURCES += main.cpp\
27         mainwindow.cpp \
28     sprite.cpp
29
30 HEADERS  += mainwindow.h \
31     sprite.h
32
33 FORMS    += mainwindow.ui
34
35 RESOURCES += sprite.png
```

9. Запускаем проект.



10. Если запустить проект не удалось из-за ошибки **undefined reference to 'vtable for Sprite'**, необходимо вручную перезапустить утилиту qmake. Сделать это можно через меню Сборка



11. Аналогичным образом создайте новый проект и добавьте в него спрайтовую анимацию для другого спрайта. Например, можно использовать прилагаемый файл boy_sprite.png или создать свой. Добавьте к спрайту управление с помощью клавиатуры (см. предыдущее занятие).