Лабораторная работа №1

Цель работы

Изучить построение графики при создании ПО.

Задание

Построить произвольное изображение с использованием примитивов. В построении использовать различные цвета, толщину линий и заливку. Для отображения использовать как прямой вывод на виджет, так и класс `QGraphicsView`. Допускается написание отдельных программ для каждого способа вывода.

Изображение должно обладать уникальностью.

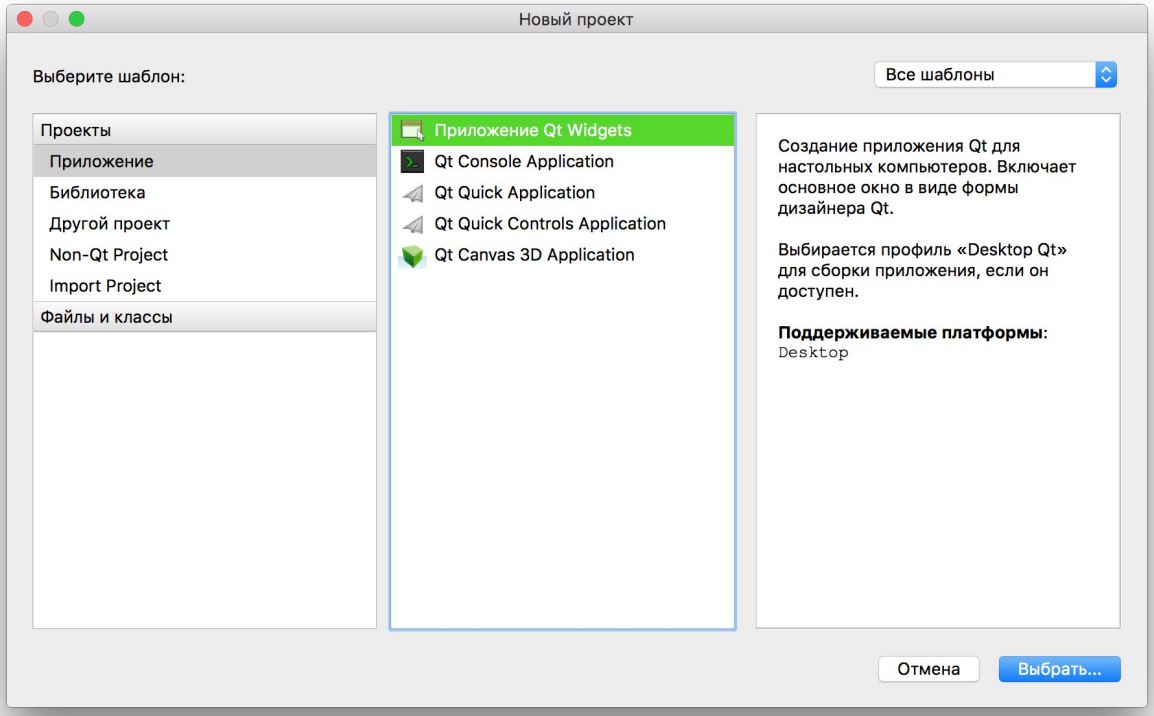
Дополнительные задания

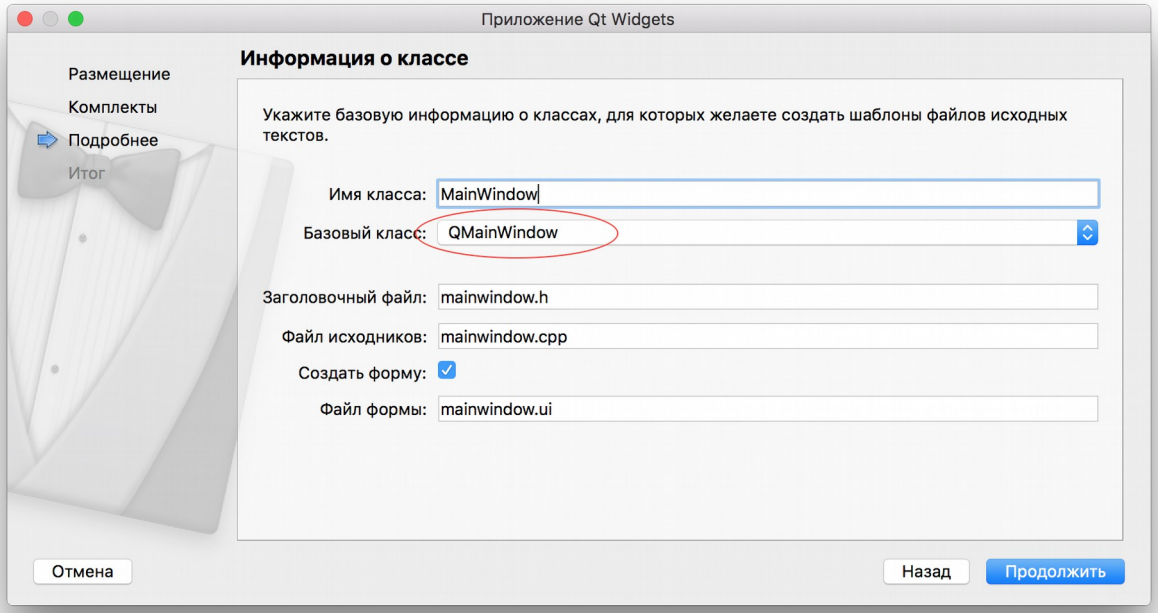
Не являются обязательными к выполнению.

1. Создать свой виджет, который будет отображать изображение.
2. Добавить анимацию изображения.
3. Создать свой элемент для `QGraphicsView.`

Как делать

1. Создаем проект для оконных приложений на базе Qt Widgets. Следуем всем этапам, как обычно.

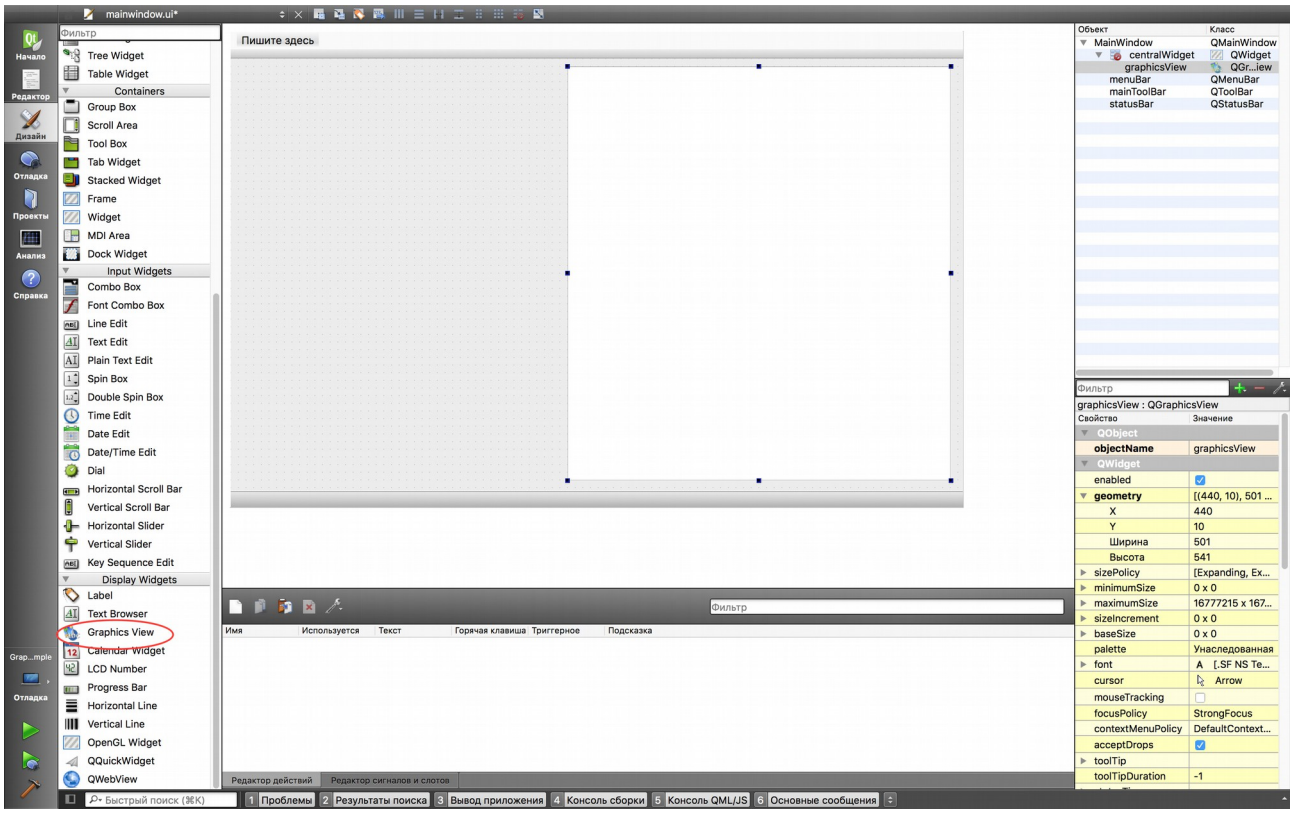




Выбираем базовый класс QMainWindow или QWidget.

Для QGraphicsView

1. Надо добавить компонент `QgraphicsView на форму окна.`



Теперь можно построить сложное изображение из готовых примитивов. Код для создания изображения можно разместить в обработчике нажатия кнопки, пункта меню или в конструк - торе класса окна.

2. Написание кода для построения изображения.

Первым делом необходимо подключить заголовочные файлы примитивов. Примитивы можно использовать готовые, например эллипс, прямоугольник, многоугольник. Или же создать свой примитив. Информацию о готовых примитивах и их создании можно найти в документации. В разделе Graphics View Framework.

Для работы с QgraphicsView надо создать QgraphicsScene:

```
QGraphicsScene *scene = new QgraphicsScene;
```

Qtransform задает геометрические преобразования.

```
QTransform tr;
```

QgraphicsEllipseItem – приметив эллипса. Задается прямоугольником, в который вписывается эллипс.

```
QGraphicsEllipseItem *el =
        new QGraphicsEllipseItem(0,0,1,5);
```

Задаем перо (для линий и контуров) и кисть (для заливки). Перо имеет тип линии, цвет и толщину. Кисть имеет опять же цвет и способ заливки: пустая, сплошная, разные виды штрихо-вок.

```
el->setPen(QPen(Qt::green));
el→setBrush(QBrush(Qt::blue,Qt::SolidPattern));
```

Поворот и масштабирование изображения. Угол измеряется в градусах. Масштаб задаётся вещественными числами отдельно для каждой оси.

```
tr.rotate(30);
tr.scale(2,2);
```

Добавление преобразований к эллипсу.

```
el→setTransform(tr);
```

Добавляем созданный эллипс к сцене.

```
scene→addItem(el);
```

И создаем элипс из параметров по умолчанию.

```
scene→addEllipse(0,0,1,5);
```

Добавляем сцену к QgraphicsView, расположенному на форме.

```
ui→graphicsView→setScene(scene);
```

Задаем масштаб всего изображения.

```
ui->graphicsView->scale(20,20);
```

Для прямого вывода

Для прямого вывода надо перегрузить метод void paintEvent(QPaintEvent *e);

В его реализации можно рисовать в области виджета для которого перегружен метод.

Необходимо подключить заголовочный файл Qpainter, поскольку через этот клас будет происходить работа построения изображения.

Создаем объект QPainter. Он и будет строить изображение. Построение изображения будет происходить всякий раз, когда надо будет обновить изображение окна.

```
QPainter painter(this);
```

Настройка кисти и пера. Ничего нового.

```
QPen pen;
pen.setWidth(3);
```

```
pen.setColor(Qt::blue);
pen.setStyle(Qt::DashDotLine);

QBrush brush;
brush.setColor(Qt::green);
brush.setStyle(Qt::SolidPattern);
```

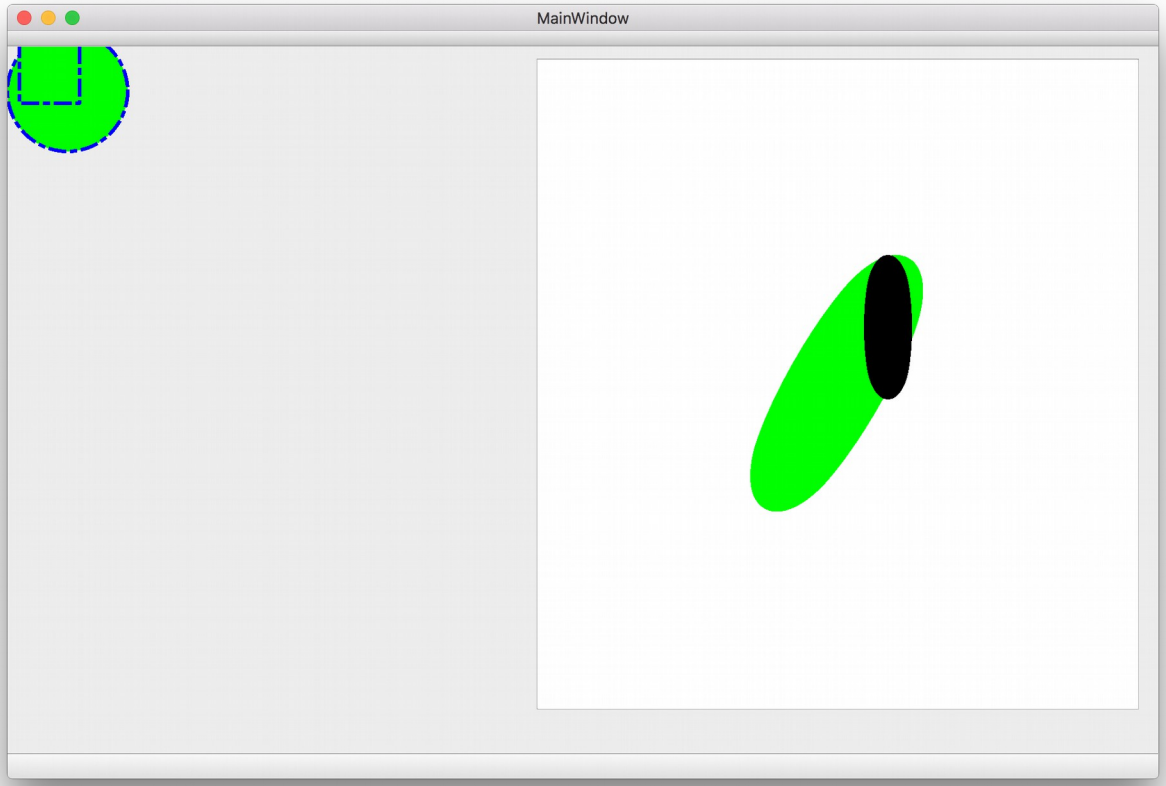
Задаем перо и цвет Qpainter. Они будут использоваться до следующей смены.

```
painter.setPen(pen);
painter.setBrush(brush);
```

Используем методы Qpainter для построения изображения. Все доступные возможности смотрим в документации.

```
painter.drawEllipse(0,0,100,100);
painter.drawRect(10,10,50,50);
```

Итог работы



Да, при построении на виджете надо учитывать перекрытие области видимости другими объектами.

У двух способов различаются системы координат. Для Qpainter отсчет идет от левого верхне - го угла. Ось X увеличивается с лева на право. Ось У – сверху вниз.