|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_«Информатика и системы управления» (ИУ)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_«Информационная безопасность» (ИУ8)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***Разработка простейшего графического***

***редактора для рисования***

Студент ИУ8-32 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П. Е. Быкова**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. В. Мацак**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

Оглавление

[ЦЕЛЬ 3](#_Toc59816054)

[ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ 3](#_Toc59816055)

[ВВЕДЕНИЕ 4](#_Toc59816056)

[ТРЕБОВАНИЯ К ПРОЕКТУ 4](#_Toc59816057)

[ПРОЕКТИРОВАНИЕ СИСТЕМЫ 4](#_Toc59816058)

[ВЫБОР ТЕХНОЛОГИЙ 7](#_Toc59816059)

[Выбор языка программирования 7](#_Toc59816060)

[Выбор используемых библиотек 11](#_Toc59816061)

[ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ 14](#_Toc59816062)

[ЗАКЛЮЧЕНИЕ 14](#_Toc59816063)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc59816064)

# ЦЕЛЬ

Создание простейшего графического редактора для рисования на алгоритмическом языке программирования Си++ при помощи знаний и навыков, полученных на 1 курсе по специальности «Информационная безопасность».

# ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

**Алгоритмический** **язык** **программирования** - формальный язык, используемый для записи, реализации и изучения алгоритмов. В отличие от большинства языков программирования, алгоритмический язык не привязан к архитектуре компьютера, не содержит деталей, связанных с устройством машины.

**Объектно-ориентированное** **программирование** (ООП) - методология программирования, основанная на представлении программы в виде совокупности [объектов,](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) каждый из которых является экземпляром определенного [класса,](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) а классы образуют иерархию наследования.

**Графический редактор** — [программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) (или пакет программ), позволяющая создавать, просматривать, обрабатывать и редактировать цифровые изображения (рисунки, картинки, фотографии) на [компьютере](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80).

**Qt** (произносится [[ˈkjuːt]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) (*кьют*) как «cute»[[9]](https://ru.wikipedia.org/wiki/Qt#cite_note-9) или неофициально Q-T (кью-ти)) — [кроссплатформенный](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Кроссплатформенное программное обеспечение) [IDE](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для разработки программного обеспечения на языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B). Есть также «привязки» ко многим другим языкам программирования: [Python](https://ru.wikipedia.org/wiki/Python) — [PyQt](https://ru.wikipedia.org/wiki/PyQt), [PySide](https://ru.wikipedia.org/wiki/PySide); [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby) — [QtRuby](https://ru.wikipedia.org/wiki/QtRuby" \o "QtRuby); [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) — [Qt Jambi](https://ru.wikipedia.org/wiki/Qt_Jambi" \o "Qt Jambi); [PHP](https://ru.wikipedia.org/wiki/PHP) — [PHP-Qt](https://ru.wikipedia.org/wiki/PHP-Qt" \o "PHP-Qt) и другие.

**Кроссплатформенность** (*межплатформенность*) — способность [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) работать с несколькими [аппаратными платформами](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0) или [операционными системами](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).

**Интегри́рованная среда́ разрабо́тки**, **ИСP** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Integrated development environment* *—****IDE***), также **единая среда разработки,** **ЕСР** — комплекс программных средств, используемый [программистами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) для разработки [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (ПО).

**Виджет** – это визуальный элемент интерфейса программы, помогает получить оперативный доступ к тому или иному стандартному действию.

# ВВЕДЕНИЕ

Графический редактор – программа, которая использует довольно расширенный способ отображения графических рисунков или картин виртуально, то есть на компьютере. Непосредственно всё что отображается в редакторе можно будет распечатать на принтере.

Конечно же, графические редакторы очень разнообразные. Их существует бессчетное множество, и каждый из них разработан под определённую цель. Данное приложение призвано дать начальные знания и умения по работе с графическими редакторами. Оно отлично подходит для тех, кто только начинает пробовать себя в цифровом рисунке.

# ТРЕБОВАНИЯ К ПРОЕКТУ

Данное приложение должно удовлетворять следующим требованиям:

* принимать и обрабатывать действия пользователя
* связывать действия пользователя с виртуальными инструментами для рисования (клавиатура, мышь,…)
* выполнение основных функций для рисования
* возможность сохранять получаемые изображения

# ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Система данного графического приложения состоит из 8 классов :

* Класс “окно” (mainwindow)
* Класс “лист” (Scene)
* Класс “линия” (Line)
* Класс “фигура” (Figure)
* Класс “ромб” (Romb)
* Класс “прямоугольник” (Square)
* Класс “эллипс” (Ellipse)
* Класс “треугольник” (Triangle)

Класс **QMainWindow** предоставляет главное окно приложения. Главное окно предоставляет структуру для создания пользовательского интерфейса приложения. Qt имеет класс QMainWindow и [связанные с ним классы](http://doc.crossplatform.ru/qt/4.4.3/application.html) для управления главным окном. QMainWindow имеет собственный компоновщик, в который можно добавлять такие объекты других классов, как  [QToolBar](http://doc.crossplatform.ru/qt/4.4.3/qtoolbar.html)'ы, [QDockWidget](http://doc.crossplatform.ru/qt/4.4.3/qdockwidget.html)'ы, [QMenuBar](http://doc.crossplatform.ru/qt/4.4.3/qmenubar.html), и [QStatusBar](http://doc.crossplatform.ru/qt/4.4.3/qstatusbar.html). Компоновщик имеет центральную область, которая может быть занята любым виджетом. В данном приложении класс **mainwindow** тесно связан с объектом класса **Scene**.

Рисунок 1 иллюстрирует главное окно данного приложения:



Рисунок 1 – Пользовательский интерфейс

Класс **Scene** отвечает за основной объект, на котором непосредственно будет производиться рисование всех изображений. В своей структуре он содержит методы/алгоритмы, позволяющие отображать отрисовку различных графических объектов, а также методы/алгоритмы, связывающие действия пользователя с виртуальными инструментами рисования (клавиатура, мышь,…):

* За отрисовку произвольной линии отвечает класс **Line,** в свою очередь класс **Scene** имеет указатель на объект класса **Line.** Также **Line** присутствует в методах класса **Scene** ***mousePressEvent(QMouseEvent \*event) и mouseMoveEvent(QMouseEvent \*event), mouseReleaseEvent(QMouseEvent \*event)*** обрабатывающих события нажатия и перемещения мыши.
* За отрисовку фигур отвечает класс **Figure** (создание фигуры при клике мыши и изменение её размера и формы при помощи перемещения курсора мыши), в свою очередь класс **Scene** имеет указатель на объект класса **Figure.** Также **Figure** присутствует в методах класса **Scene** ***mousePressEvent(QMouseEvent \*event) и mouseMoveEvent(QMouseEvent \*event),*** обрабатывающих события нажатия и перемещения мыши, связанных с фигурой.
* За изменение типов фигур отвечают методы класса **Scene** ***typeFigure****()* и ***setTypeFigure(const int type)***, а также созданные специально классы: **Romb, Square, Ellipse, Triangle,** осуществляющие отрисовку конкретных фигур.
* За перемещение фигур отвечает метод класса **Scene *keyPressEvent(QKeyEvent \*event)*,** который обрабатывает события клавиатуры.
* Ещё один метод класса **Scene** ***resizeEvent(QResizeEvent \*)*** отвечает за регулировку размера «листа»: сжатие, растяжение, полноэкранный режим.

Рисунок 2 иллюстрирует окно отрисовки данного приложения:

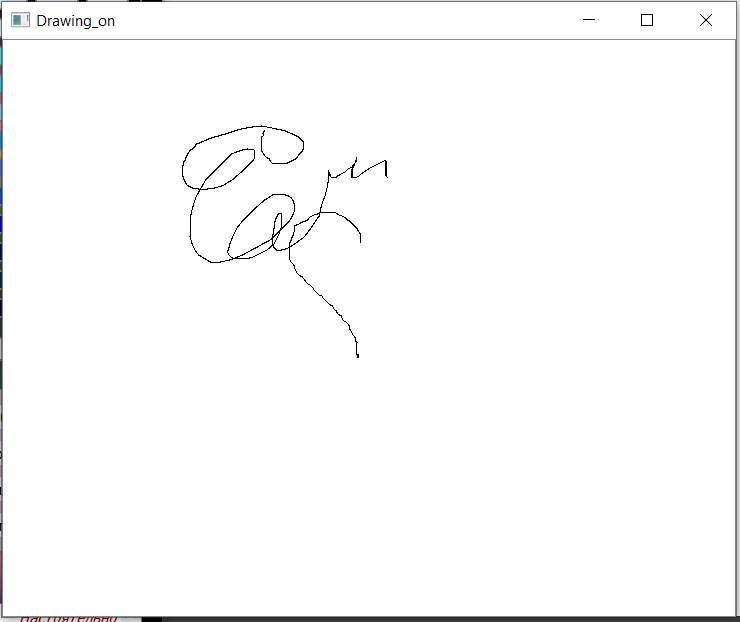
****

Рисунок 2 – Лист

# ВЫБОР ТЕХНОЛОГИЙ

## Выбор языка программирования

Выбор языка программирования очень важная часть в разработке любого проекта. Нужно подобрать именно тот язык, который будет самым

|  |  |
| --- | --- |
| оптимальным для конкретной разработки. Именно поэтому | чем больше |
| проект, тем больше стек технологий, который в нем используется. | |

Глобальные проекты состоят из огромного количества подпроектов. Для каждого из подпроектов выбирается тот язык, который будет наилучшем образом справляться с задачами, поставленными в нем.

Важными критериями при выборе технологий являются:

Размер и тип проекта Сложность проекта

Скорость разработки

Доступные инструменты разработки Наличие готовых решений

Гибкость решения

Наличие подробной документации Требования к нагрузкам

Требования к безопасности Кроссплатформенность

Возможность интеграции с другими решениями

Так как данная работа планировалась быть не очень объёмной, а время на разработку было ограничено, то стоял вопрос выбора одного языка, а не нескольких.

Всем известно, что языки программирования разделяются по сфере применения. Основными сферами являются веб-разработка, мобильная и игровая разработка. Самыми популярными языками в веб-разработке на данный момент являются: HTML, CSS, JavaScript, Java , Python и PHP . В разработке мобильных приложений на Android: Java, на iOS: Swift и Objective-C. И наконец в разработке игр: С#, С++, JavaScript, Java, Smalltalk. Данный проект относиться к сфере разработке игр, поэтому выбор был из последнего списка ЯП, технологии которого относятся к объектно-ориентированным языкам программирования.

Рассмотрев достоинства и недостатки каждого из перечисленных языков, я остановилась на выборе С++. Ниже приведено подробное доказательство принятого решения.

С++ - язык общего назначения и задуман для того, чтобы настоящие программистыполучилиудовольствие от самого процесса программирования. За исключением второстепенных деталей он содержит язык С как подмножество. Язык С расширяется введением гибких и эффективных средств, предназначенных для построения новых типов. Программист структурирует свою задачу, определив новые типы, которые точно соответствуют понятиям предметной области задачи. Такой метод построения программы обычно называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. С такимиобъектамиможно работать надежно ипросто даже в тех случаях, когда их тип нельзя установить на стадии трансляции. Программирование с использованием таких объектов обычно называют объектно-ориентированным. Если этот метод применяется правильно, то программы становятся короче и понятнее, а сопровождение их упрощается.

Достоинства С++:

|  |  |  |
| --- | --- | --- |
|  | Чрезвычайно мощный язык, содержащий с  программ практически любого назначения. | редства создания эффективных |
| Компилируемость со статической типизаци Сочетание высокоуровневых и низкоуровн | | ей.  евых средств. |

Реализация ООП.

Работает максимально быстро.

Предсказуемое выполнение программ, что является важным для построения систем реального времени.

Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов. Это упрощает (достаточно объявить переменную) и делает более надёжным освобождение ресурсов (память, файлы, семафоры и т. п.), а также позволяет гарантированно выполнять переходы состояний программы, не обязательно связанные с освобождением ресурсов (например, запись в журнал).

Пользовательские функции-операторы позволяют кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.

Язык поддерживает понятия физической (const) и логической (mutable) константности. Это делает программу надёжнее, так как позволяет компилятору, например, диагностировать ошибочные попытки изменения значения переменной. Объявление константности даёт программисту, читающемутекст программы дополнительное представление о правильном использовании классов и функций, а также может являться подсказкой для оптимизации. Перегрузка функций-членов по признаку константности позволяет определять изнутри объекта цели вызова метода (константный для чтения, неконстантный для изменения). Объявление mutable позволяет сохранять логическую константность при использовании кэшей и ленивых вычислений.

Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование, метапрограммирование (шаблоны, макросы).

Используя шаблоны, возможно создавать обобщённые контейнеры и алгоритмы для разных типов данных, а также специализировать и вычислять на этапе компиляции.

Возможность имитации расширения языка для поддержки парадигм, которые не поддерживаются компиляторами напрямую. Например, библиотека Boost.Bind позволяет связывать аргументы функций.

Возможность создания встроенных предметно-ориентированных языков программирования. Такой подход использует, например библиотека Boost.Spirit, позволяющая задавать EBNF-грамматику парсеров прямо в коде C++.

Используя шаблоны и множественное наследование можно имитировать классы-примеси и комбинаторную параметризацию библиотек. Такой подход применён в библиотеке Loki, класс SmartPrt которой позволяет, управляя всего несколькими параметрами времени компиляции, сгенерировать около 300 видов «умных указателей» для управления ресурсами.

Кроссплатформенность: стандарт языка накладывает минимальные требования на ЭВМ для запуска скомпилированных программ. Для определения реальных свойств системы выполнения в стандартной библиотеке присутствуют соответствующие возможности (например, std::numeric\_limits ). Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем.

Эффективность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования — при необходимости язык позволяет обеспечить максимальную эффективность программы.

Имеется возможность работы на низком уровне с памятью, адресами.

Высокая совместимость с языком Си, позволяющая использовать весь существующий Си-код (код на Си может быть с минимальными переделками скомпилирован компилятором C++; библиотеки, написанные на Си, обычно могут быть вызваны из C++ непосредственно без каких-либо дополнительных затрат, в том числе и на уровне функций обратного вызова, позволяя библиотекам, написанным на Си, вызывать код, написанный на С++).

В совокупности с вышеописанным, можно сделать вывод, что в данной разработки программного обеспечения C++ - оптимальный выбор.

## Выбор используемых библиотек

Qt (произносится как «кьют») – кросс-платформенный инструментарий для разработки программного обеспечения. Этот инструментарий создан компанией Trolltech и в данный момент принадлежит компании Nokia. Qt – это совокупность кросс-платформенной библиотеки классов, реализованной на языке C++, и ряда дополнительных инструментальных средств, включающих Meta Object Compiler (MOC) – объектный предкомпилятор, User Interface Compiler (UIC) – компилятор пользовательских интерфейсов, qmake – средство управления сборкой проектов.

Поддерживаются операционные системы MS Windows, Linux, MacOS, а также встраиваемые операционные системы Embedded Linux, Windows CE, Symbian. Наиболее известными примерами разработки на Qt являются: программа-коммуникатор Skype, медииа-плеер VLC, Google Earth (см. http://qt.nokia.com/qt-in-use ), графический интерфейс пользователя KDE, применяемый в ОС Linux. На сайте http://www.qt-apps.org/ приводится база OpenSource проектов, использующих Qt.

В состав Qt входят следующие группы классов:

- классы, обеспечивающие разработку оконного графического интерфейса пользователя, включая все основные управляющие примитивы;

- классы, реализующие работу с потоками, объектами синхронизации процессов/потоков;

- классы для работы с 2-х и 3-х мерной графикой, классы реализующие поддержку некоторых графических форматов хранения;

- реализация динамических массивов в виде шаблонов C++;

- классы для работы с XML;

- и пр.

В настоящее время существует две не полностью совместимые ветви версий Qt – 3.х и 4.х. При этом ветвь 3.х сохраняется для поддержки старых программ, а разработка новых рекомендована с использованием 4.х. Кроме того, существуют OpenSource версии для разработки программ, не предназначенных для коммерческого использования (доступны на сайте http :// qt.nokia. com), и коммерческие версии для разработки программ без ограничения целевого назначения.

Qt 3.x и 4.x поставляются в составе современных Linux-дистрибутивов, обеспечивая возможность разработки в интегрированных средах KDevelop, Eclipse и пр. Для ОС Windows имеются средства, позволяющие интегрировать Qt в среду разработки: uic, moc – компиляторы и QtDesigner. При этом возможна интеграция Qt 3.x в MS Visual Studio 2003 и Qt 4.x — в MS Visual Studio 2003/2005/2008. OpenSource Qt 4.x для Windows может быть интегрирована в IDE Eclipse с подключенным компилятором mingw-gcc, а также использоваться совместно с кроссплатформенной IDE QtCreator. Библиотеки для использования могут быть откомпилированы любым компилятором C++, имеющимся в ОС, например для Windows - MS Visual С++, Borland C++, mingw-gcc.

В библиотеке реализовано автоматическое удаление объектов, являющихся элементами графического интерфейса пользователя. Механизм реализован следующим образом: любой подобный объект Qt является потомком QObject, в состав которого входят средства хранения и позиционирования списка потомков, т.е. объектов, при создании которых этот объект указан как parent. Следовательно, при удалении корневого объекта возможно удаление всего дерева объектов-потомков.

При использовании Qt совершенно естественным является переопределение классов Qt средствами C++, что существенно упрощает код в приложениях, требующих однотипной реализации нестандартных элементов, например создание класса кнопки виртуальной клавиатуры с изменяемой надписью/рисунком на основе стандартного класса QButton.

В Qt может быть использовано кросс-платформенное средство управления сборкой проектов qmake, посредством которого из .pro-файлов генерируются файлы makefile для конкретной платформы с конкретными компиляторами и компоновщиками.

Формы с использованием классов Qt могут создаваться вручную или с использованием специального пакета QtDesigner. При создании форм вручную программист кодирует текст программы, включая по мере необходимости вызовы объектов классов Qt. При использовании QtDesigner программист графически компонует внешний вид и связи сигналов и слотов формы, а компилятор интерфейса UIC формирует из полученного описания формы код на языке C++, обеспечивающий создание этой формы.

Qt расширяет синтаксис описания классов C++ специальными средствами, обработка которых возложена на MOC. MOC обрабатывает исходный текст программы, подставляя вместо специфических конструкций реализацию заказанных свойств на С++. Соответственно на выходе MOC получается исходный код С++.

Компиляция и сборка программы осуществляется компилятором C++ и компоновщиком, доступными в рамках платформы, где осуществляется сборка.

К таким средствам в частности относятся сигналы и слоты, поддержка которых для конкретного диалога также генерируется автоматически в коде отдельной программы C++.

Ключевым механизмом взаимодействия объектов в Qt являются сигналы и слоты.

Каждый объект, интегрированный в систему управления Qt, т.е. описанный как Q\_OBJECT, может иметь типизированные слоты, обеспечивающие прием и обработку типизированных сигналов от других объектов, и собственные сигналы, прием которых могут осуществлять другие объекты. Связь между сигналами и слотами конкретных объектов устанавливается посредством функции connect(…).

Декларация сигналов и слотов осуществляется в теле класса с помощью ключевых слов signals и slots, воспринимаемых компилятором moc. Если необходимо предотвратить использование указанных ключевых слов, встречающихся в других библиотеках, то вместо них используют ключевые слова Q\_SIGNALS, Q\_SLOTS.

По правилам Qt один слот может принимать несколько сигналов, а один сигнал транслироваться на несколько слотов. Причем во взаимодействии участвуют не классы, а конкретные объекты, поэтому схема передачи сигналов к слотам может быть в любой момент динамически изменена.

Следует отметить, что механизм слотов не исключает возможности использования средств наследования и полиморфизма языка C++, так что любой класс Qt может быть переопределен.

При необходимости быстрого получения результата, проведения экспериментов по размещению объектов, общей оценки интерфейса возможно использование специального редактора интерфейсов QtDesigner. QtDesigner не накладывает никаких ограничений на средства разработки, поскольку интерфейс, созданный им, в конечном счете будет преобразован компилятором uic в код программы на языке C++, обеспечивающий создание именно этого интерфейса.

Это позволяет также использовать QtDesigner для обучения принципам программирования Qt и размещения элементов в форме, т.к. результирующий код является доступным и использует те же классы Qt, которые необходимы при ручной разработке.

Библиотека Qt снабжена системой документации QtAssistant, реализованной единообразно для всех платформ, на которых возможна разработка с использованием этой библиотеки.

QtAssistant предоставляет возможность навигации по разделам, по индексному указателю, а также нахождения необходимой фразы, в том числе включающей имена классов и/или методов, по контексту во всех статьях документации. Документация предоставляет несколько вариантов группировки классов и функций библиотеки, что позволяет быстро найти классы для работы, например с сигналами и слотами, с графикой, классы контейнеры и пр.

# ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ

В ходе написания курсового проекта возникали некоторые проблемы, поэтому для данного приложения была создана специальная документация по решению этих самых затруднительных моментов. Она находится под именем **«Справка»** в **Меню** приложения, которое располагается верхнем левом углу главного окна, там описаны все проблемы и пути их решения.

Для привлечения внимания к соответствующей документации на самом главном окне создан специальный виджет, выполняющий функцию уведомления пользователя об этом.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были получены навыки работы с графической библиотекой QT, освоены принципы разработки приложений. Был создан простейший графический редактор, позволяющий создавать пользователю собственные иллюстрации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация: Стандартные библиотеки С++ [Электронный курс]. URL: [www.cplusplus.com/reference/](http://www.cplusplus.com/reference/)
2. Документация: Официальная документация по языку программирования С++ [Электронный курс]. URL: <http://ru.cppreference.com/w/>
3. Документация: Официальная документация библиотеки QT. URL: <http://doc.crossplatform.ru/qt/4.4.3/>
4. Учебное пособие к лабораторным работам по курсу Алгоритмические языки и программирование МОСКВА 2009 год МГТУ им. Баумана Самарев Роман Станиславович «Программирование с использованиембиблиотеки Qt» URL:http://elearning.bmstu.ru/moodle/file.php/1/common\_files/library/ALP/Qt/bmstu\_iu6\_qt4.pdf