Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

РАЗРАБОТКА ГРАФИЧЕСКОГО РЕДАКТОРА

ОТЧЕТ
ПО РЕЗУЛЬТАТАМ
Учебной практики:
(вид практики)
Получение первичных навыков научно-исследовательской работы (рассред.)
(тип практики)

Обучающийся гр. 430-4

_____ В. С. Завзятов
(подпись) (И.О. Фамилия)

«___» _____ 2021 г.
(дата)

Руководитель практики
от Университета:
Старший преподаватель каф. АСУ,
(должность, ученая степень, звание)

_____ _____ А. Е. Косова
(оценка) (подпись) (И.О. Фамилия)

«___» _____ 2021 г.
(дата)

Томск 2021

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ
Зав. кафедрой АСУ
канд. техн. наук, доц.
В. В. Романенко

(подпись)
«___» _____ 2021 г.
(дата)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на Учебную практику:
(вид практики)

Получение первичных навыков научно-исследовательской работы (рассред.)
(тип практики)

студенту гр. 430-4 факультета Систем управления

Завзятову Владиславу Сергеевичу
(Ф.И.О. студента)

1. Тема практики: Разработка Графического редактора

2. Цель практики: Получить навыки разработки приложений, получить навыки командной разработки

3. Задачи практики: Изучить Фреймворк Qt, Реализовать пользовательский интерфейс, изучить основы HTML, реализовать мануал, реализовать систему обратной связи

4. Сроки практики: с «___» _____ 2021 г. по «___» _____ 2021 г.

Совместный рабочий график (план) проведения практики

№ п/п	Перечень заданий	Сроки выполнения
1	Выбор языка программирования и фреймворка	15.09
2	Определение функционала приложения	20.09
3	Реализация класса MainWindow	4.10
4	Реализация класса Manual	1.11
5	Реализация класса Feedback	15.11
6	Завершение привязки всех сигналов к слотам	19.11
7	Комплексное тестирование приложения	29.11

Дата выдачи задания: «___»_____ 2021 г.

Руководитель практики от Университета

Старший преподаватель каф. АСУ _____ А. Е. Косова
(должность, ученая степень, звание) (подпись) (Ф.И.О.)

Задание принял к исполнению: «___» _____ 2021 г.

Студент гр. 430-4 _____ В. С. Завзятов
(подпись) (Ф.И.О.)

Оглавление

Введение

Без компьютерной графики невозможно представить себе не только компьютерный, но и обычный, вполне материальный мир. На сегодняшний день компьютеры и компьютерная графика неотъемлемая часть жизни современного общества. Для примера назовём медицину (компьютерная томография), научные исследования (визуализация строения вещества, векторных полей и других данных), моделирование тканей и одежды, опытно-конструкторские разработки, рекламные щиты, цветные журналы, спецэффекты в фильмах - всё это в той или иной мере имеет отношение к компьютерной графике. Поэтому созданы программы для создания и редактирования изображений, то есть графические редакторы.

Компьютерной графикой в последнее время занимаются многие, что обусловлено высокими темпами развития вычислительной техники. Более 90% информации здоровый человек получает через зрение или ассоциирует с геометрическими пространственными представлениями. Компьютерная графика имеет огромный потенциал для облегчения процесса познания и творчества.

1 Описание предметной области

Графический редактор — программа (или пакет программ), позволяющая создавать, просматривать, обрабатывать и редактировать цифровые изображения (рисунки, картинки, фотографии) на компьютере.

Типы графических редакторов:

- Векторный графический редактор
- Растровый графический редактор

Растровый графический редактор — специализированная программа, предназначенная для создания и обработки растровых изображений, то есть графики, которая в память компьютера записывается как набор точек, а не как совокупность формул геометрических фигур. Подобные программные продукты нашли широкое применение в обработке цифровых фотографий и применяются в работе художников-иллюстраторов, при подготовке изображений к печати типографским способом или на фотобумаге, публикации в интернете.

Векторный графический редактор представляет собой приложение, позволяющее конструировать иллюстрации из различных геометрических объектов, которые называются графическими примитивами. Каждый редактор предлагает свой набор графических примитивов.

К основным функциям графических редакторов можно отнести:

- Создание рисунка. Изображение в редакторе может создаваться как вручную, так и с использованием особых инструментов (штампов, кривых и т. д.).
- Преобразование уже готового изображения. Фотографии и картинки можно перемещать, поворачивать и масштабировать. Также такие программы предоставляют возможность работы с отдельными частями изображения. К примеру, обычно бывает доступной такая функция, как удаление фрагмента изображения. Картинки

также можно копировать как полностью, так и частями, а еще склеивать и раскрашивать.

- Ввод текста в картинку. Пользоваться при этом обычно можно самыми разными шрифтами — как современными, так и стилизованными «под старину».
- Работа с внешними устройствами. Нарисованное или отредактированное изображение при желании можно распечатать на принтере, не выходя из программы. Разумеется, файл можно сохранить в любую папку на жестком или внешнем диске.

2 Обзор программного обеспечения

2.1 Язык программирования C++

C++ — компилируемый, статически типизированный язык программирования общего назначения, на котором можно создавать программы любого уровня сложности. C++ – это мультипарадигмальный язык (от слова парадигма – стиль написания компьютерных программ), включающий широкий спектр различных стилей и технологий программирования. Часто его причисляют к объектно-ориентированным языкам, но, строго говоря, это не так. В процессе работы разработчик получает абсолютную свободу в выборе инструментов для того, чтобы задача, решаемая с помощью того или иного подхода, была решена максимально эффективно. Иными словами, C++ не понуждает программиста придерживаться только одного стиля разработки программы (например, объектно-ориентированного).

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C. Со временем, практическая совместимость между языками C и C++ постепенно будет утрачиваться, так как языки разрабатывают разные группы по стандартизации, не взаимодействующие друг с другом.

C++ повлиял на многие языки программирования, в их числе: Java, C#, PHP, Perl, D, Lua, Rust.

C++ имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

2.2 Фреймворк Qt

2.2.1 Общая информация о фреймворке

Qt - это кроссплатформенный фреймворк для разработки ПО на языке программирования C++(и не только). Также имеется и для Ruby — QtRuby, для Python — PyQt, PHP — PHP-Qt и других языков программирования. Разрабатывается компанией Trolltech с 1996 года.

С использованием этого фреймворка написано множество популярных программ: 2ГИС для Android, Kaspersky Internet Security, Virtual Box, Skype, VLC Media Player, Opera и другие. KDE — это одно из окружений рабочего стола со множеством программ для Linux написано с использованием фреймворка Qt.

Qt полностью объектно-ориентированная, кросс-платформенная. Дает возможность разрабатывать платформо-независимое ПО, написанный код можно компилировать для Linux, Windows, Mac OS X и других операционных систем. Включает в себя множество классов для работы с сетью, базами данных, классы-контейнеры, а также для создания графического интерфейса и множество других(чуть ниже).

Qt использует MOC (Meta Object Compiler) для предварительной компиляции программ. Исходный текст программы обрабатывается MOC, который ищет в классах программы макрос Q_OBJECT и переводит исходный код в мета-объектный код, после чего мета-объектный код компилируется компилятором C++. MOC расширяет функциональность фреймворка, благодаря ему добавляются такие понятия, как слоты и сигналы.

В Qt имеется огромный набор виджетов (Widget), таких как: кнопки, прогресс бары, переключатели, checkbox, и другие — они обеспечивают стандартную функциональность GUI (графический интерфейс пользователя). Позволяет использовать весь функционал пользовательского интерфейса — меню, контекстные меню, drag&drop.

2.2.2 Описание используемых классов фреймворка

2.2.2.1 QMainWindow

Главное окно обеспечивает основу для создания пользовательского интерфейса приложения. Qt имеет QMainWindow и связанные с ним классы для управления главным окном. QMainWindow имеет свой собственный макет, в который вы можете добавить QToolBars, QDockWidgets, QMenuBar и QStatusBar. В макете есть центральная область, которую можно занять любым виджетом.

2.2.2.2 QWidget

Виджет - это элемент пользовательского интерфейса: он получает события от мыши, клавиатуры и другие события от оконной системы и рисует свое представление на экране. Каждый виджет прямоугольный, и они отсортированы в Z-порядке. Виджет обрезается своим родителем и виджетами перед ним.

Виджет, который не встроен в родительский виджет, называется окном. Обычно окна имеют рамку и строку заголовка, хотя также можно создавать окна без такого оформления, используя подходящие флаги окна. В Qt QMainWindow и различные подклассы QDialog являются наиболее распространенными типами окон.

Конструктор каждого виджета принимает один или два стандартных аргумента:

QWidget * parent = nullptr является родителем нового виджета. Если это nullptr (по умолчанию), новый виджет будет окном. В противном случае он будет дочерним по отношению к родительскому объекту и будет ограничен геометрией родительского объекта (если вы не укажете Qt :: Window как флаг окна).

Qt :: WindowFlags f = {} (где доступно) устанавливает флаги окна; значение по умолчанию подходит почти для всех виджетов, но чтобы получить,

например, окно без рамки оконной системы, вы должны использовать специальные флаги.

QWidget имеет много функций-членов, но у некоторых из них мало прямых функций; например, QWidget имеет свойство шрифта, но никогда не использует его. Есть много подклассов, которые обеспечивают реальную функциональность, например QLabel, QPushButton, QListWidget и QTabWidget.

2.2.2.3 QTextBrowser

Этот класс расширяет QTextEdit (в режиме только для чтения), добавляя некоторые функции навигации, чтобы пользователи могли переходить по ссылкам в гипертекстовых документах.

Если вы хотите предоставить своим пользователям редактируемый редактор форматированного текста, используйте QTextEdit. Если вам нужен текстовый браузер без гипертекстовой навигации, используйте QTextEdit и используйте QTextEdit :: setReadOnly () для отключения редактирования. Если вам просто нужно отобразить небольшой фрагмент форматированного текста, используйте QLabel.

Содержимое QTextEdit устанавливается с помощью setHtml () или setPlainText (), но QTextBrowser также реализует функцию setSource (), что позволяет использовать именованный документ в качестве исходного текста. Имя ищется в списке путей поиска и в каталоге текущей фабрики документов.

Если имя документа заканчивается привязкой (например, «#anchor»), текстовый браузер автоматически прокручивается до этой позиции (с помощью scrollToAnchor ()). Когда пользователь щелкает гиперссылку, браузер сам вызывает setSource () со значением href ссылки в качестве аргумента. Вы можете отслеживать текущий источник, подключившись к сигналу sourceChanged ().

2.2.2.4 QFileDialog

Класс QFileDialog позволяет пользователю перемещаться по файловой системе, чтобы выбрать один или несколько файлов или каталог.

Самый простой способ создать QFileDialog - использовать статические функции. Если вы хотите использовать несколько фильтров, разделите каждый из них двумя точками с запятой.

Свойство fileMode содержит режим работы диалога; это указывает, какие типы объектов предполагается выбрать пользователем. Используйте setNameFilter (), чтобы установить фильтр файлов диалогового окна.

Диалоговое окно файла имеет два режима просмотра: Список и Подробности. Список представляет содержимое текущего каталога в виде списка имен файлов и каталогов. Подробно также отображает список имен файлов и каталогов, но предоставляет дополнительную информацию рядом с каждым именем, например размер файла и дату изменения.

2.2.2.5 QMenuBar

Строка меню состоит из списка пунктов раскрывающегося меню. Вы добавляете пункты меню с помощью addMenu (). Нет необходимости выкладывать строку меню. Он автоматически устанавливает свою собственную геометрию в верхнюю часть родительского виджета и соответствующим образом изменяет ее всякий раз, когда изменяется размер родительского виджета.

В большинстве приложений в стиле главного окна вы должны использовать функцию menuBar (), предоставленную в QMainWindow, добавляя QMenus в строку меню и добавляя QActions во всплывающие меню.

Виджеты можно добавлять в меню, используя экземпляры класса QWidgetAction для их хранения. Затем эти действия можно вставить в меню обычным способом.

На разных платформах разные требования к внешнему виду панелей меню и их поведению при взаимодействии с ними пользователя. Например,

системы Windows часто настраиваются таким образом, что мнемоника подчеркнутых символов, обозначающая сочетания клавиш для элементов в строке меню, отображается только при нажатии клавиши Alt.

2.2.2.6 QMenu

Виджет меню - это меню выбора. Это может быть раскрывающееся меню в строке меню или отдельное контекстное меню. Выпадающие меню отображаются в строке меню, когда пользователь щелкает соответствующий элемент или нажимает указанную комбинацию клавиш. Используйте QMenuBar :: addMenu (), чтобы вставить меню в строку меню. Контекстные меню обычно вызываются специальной клавишей клавиатуры или щелчком правой кнопки мыши. Они могут выполняться либо асинхронно с popup (), либо синхронно с exec (). Меню также можно вызывать в ответ на нажатие кнопок; они похожи на контекстные меню, за исключением того, как они вызываются.

Меню состоит из списка действий. Действия добавляются с помощью функций addAction (), addActions () и insertAction (). Действие отображается вертикально и отображается с помощью QStyle. Кроме того, действия могут иметь текстовую метку, необязательный значок, нарисованный в самом левом углу, и последовательность горячих клавиш, например «Ctrl + X».

Существует четыре типа элементов действий: разделители, действия, отображающие подменю, виджеты и действия, выполняющие действие. Разделители вставляются с помощью addSeparator (), подменю - с помощью addMenu (), а все остальные элементы считаются элементами действий.

При вставке элементов действий вы обычно указываете получателя и слот. Получатель будет уведомлен всякий раз, когда элемент будет запущен (). Кроме того, QMenu предоставляет два сигнала, triggered () и hovered (), которые сигнализируют о QAction, запущенном из меню. Вы очищаете меню с

помощью clear () и удаляете отдельные элементы действий с помощью removeAction ().

QMenu также может предоставлять отрывное меню. Отрывное меню - это окно верхнего уровня, которое содержит копию меню. Это дает возможность пользователю «отрывать» часто используемые меню и размещать их в удобном месте на экране. Если вам нужна эта функция для определенного меню, вставьте дескриптор отрыва с помощью setTearOffEnabled (). При использовании отрывных меню имейте в виду, что эта концепция обычно не используется в Microsoft Windows, поэтому некоторые пользователи могут быть с ней не знакомы. Вместо этого рассмотрите возможность использования QToolBar.

Виджеты можно вставлять в меню с помощью класса QWidgetAction. Экземпляры этого класса используются для хранения виджетов и вставляются в меню с помощью перегрузки addAction (), которая принимает QAction. Если QWidgetAction запускает сигнал triggered (), меню закроется.

2.2.3 Слоты и сигналы

В программировании графического интерфейса, когда мы меняем один виджет, мы часто хотим что бы другой виджет получил об этом уведомление. В общем случае, мы хотим что бы объекты любого типа могла общаться с другими.

В Qt используется техника — сигналы и слоты. Сигнал вырабатывается когда происходит определенное событие. Слот это функция, которая вызывается в ответ на определенный сигнал. Виджеты Qt имеют много предопределенных сигналов и слотов, но мы всегда можем сделать дочерний класс и добавить наши сигналы и слоты в нем.

Механизм сигналов и слотов типобезопасен. Сигнатура сигнала должна совпадать с сигнатурой слота-получателя. Так как сигнатуры сравнимы, компилятор может помочь нам обнаружить несовпадение типов. Сигналы и слоты

слабо связаны. Класс, который вырабатывает сигнал не знает и не заботится о том, какие слоты его получат. Механизм сигналов и слотов Qt гарантирует, что если мы подключим сигнал к слоту, слот будет вызван с параметрами сигнала в нужное время. Сигналы и слоты могут принимать любое число аргументов любого типа.

Все классы, наследуемые от QObject или его дочерних классов (например, QWidget) могут содержать сигналы и слоты. Сигналы вырабатываются объектами когда они изменяют свое состояние так, что это может заинтересовать другие объекты. При этом он на знает и не заботится о том что у его сигнала может не быть получателя.

Слоты могут быть использованы для получения сигналов, но они так же нормальные функции-члены. Так же как объект не знает ничего о получателях своих сигналов, слот ничего не знает о сигналах, которые к нему подключены. Это гарантирует что полностью независимые компоненты могут быть созданы с помощью Qt.

Мы можем подключать к одному слоту столько сигналов, сколько захотим, также один сигнал может быть подключен к стольким слотам, сколько необходимо. Так же возможно подключать сигнал к другому сигналу.

2.3 Qt Creator

Qt Creator (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная IDE для разработки на C, C++, JavaScript и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML. Поддерживаемые компиляторы: GCC, Clang, MinGW, MSVC, Linux ICC, GCCE, RVCT, WINSCW.

Основная задача Qt Creator — упростить разработку приложения с помощью фреймворка Qt на разных платформах. Поэтому среди возможностей, присущих любой среде разработки, есть и специфичные, такие как отладка

приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов: как на QML, так и на QtWidgets.

Преимущества и недостатки Qt Creator:

Преимущества:

- Редактор кода. Еще одна особенность — редактор исходного кода. Эта функция поставляется с возможностью автозаполнения и подсветки синтаксиса. Кроме того, Qt Creator поддерживает декларативную разработку пользовательского интерфейса (UI) через свой модуль Qt Quick.
- Совместимость. Одним из приятных аспектов Qt Creator является его кроссплатформенная совместимость. Пользователи могут просто создать приложение в Windows и переместить его в Linux или Mac, открыть его в локальном Qt Creator, и оно будет соответствовать. Это позволяет извлечь максимальную пользу и заставить работать так, как это могут себе представить как начинающие, так и опытные разработчики.
- Разработка и внедрение. ПО предлагает комплексные и интуитивно понятные API. В результате, поддержка кросс-компиляции обеспечивает быструю разработку, создание прототипов и реализацию проектов
- Надежная и стабильная платформа. Qt Creator предлагает мгновенную и беспроблемную аппаратную интеграцию с полной оптимизацией и без издержек. К тому же вы имеете абсолютный контроль над своим кодом.
- Интуитивно понятный пользовательский интерфейс. Дизайн этого ПО, разработка и программирование сценариев позволяют создавать гибкие пользовательские интерфейсы. Кроме того, есть возможность выбрать лучший подход к проектированию.

- В результате, пользовательский интерфейс обеспечивает ощущение и внешний вид рабочего стола. К тому же, стиль существующего пользовательского интерфейса элементарно изменяется и настраивается.
- Настраиваемый UX. Это ПО позволяет настраивать и создавать современный интерфейс, чтобы предоставлять конечным пользователям естественный пользовательский опыт во всех средах. Qt Creator предлагает создание интерфейсов, масштабируемых для экранов разных размеров.
- Масштабирование. ПО обеспечивает эффективную оптимизацию производительности аппаратных ресурсов с помощью C++, QML или других языков программирования. Кроме того, можно увеличивать масштаб до уровня с несколькими экранами и уменьшать его чтобы сосредоточиться на небольших устройствах.
- Легкость. Это ПО является гибким и простым в использовании, а его инструменты и функции доступны для упрощения процесса разработки. К тому же, кроссплатформенное ПО предлагает библиотеки и API. Qt Creator ориентирован на будущее, так как позволяет изменять требования.

Недостатки:

- Отключение. Пользователи жалуются на неожиданное завершение работы. Таким образом, данное явление негативно влияет на рабочий процесс и производительность.
- Расположение поисковой системы. Пользователь тратит много времени на поиск данного инструмента.
- Библиотека редко обновляется, поэтому пользователи ограничены уже имеющимися материалами и ресурсами. Чтобы улучшить рабочий процесс пользователей, Qt Creator необходимо регулярно обновлять библиотеку.

3 Обзор нормативной документации

3.1 ГОСТ

Отчёт оформлен в соответствии с ОС ТУСУР 01-2013.

4 Описание структуры разрабатываемой программы

4.1 Функциональность приложения

Приложение предназначено для создания и редактирования графических изображений. Оно включает в себя работу с файлами, такую как сохранение, открытие и создание изображения. Также различные способы рисования, настройку инструментов и очистку холста. В приложении предусмотрена система обратной связи и руководство пользователя.

В приложении мы реализовали инструменты для создания и редактирования изображения, такие как рисование кистью, рисование геометрических примитивов, заливку холста и примитивов, а также разлиновку.

У инструментов есть следующие настраиваемые свойства: размер кисти, а также её цвет, стиль штриха, вид заливки и дублирование.

В качестве обратной связи, у нас выступает генерирование файла с данными, который пользователь может отправить на почту разработчикам, указанную в мануале.

Для каждой функции в нашем приложении реализована горячая клавиша.

4.2 Описание функции main

Функция main является точкой входа в приложение. Здесь идёт установка языка системы. Происходит создание и первичная настройка главного окна.

4.3 Описание класса MainWindow

Класс MainWindow представляет из себя основное окно приложения. Через него происходит взаимодействие с классами Canvas, Manual, Feedback.

Класс Canvas отвечает за холст и рисование на нём, но для работы с ним нам необходимо отобразить его в главном окне. Мы визуально отображаем

холст с помощью класса QGraphicsView. Здесь происходит настройка размера и установка позиции обозревателя холста.

Все кнопки являются переменными-членами класса MainWindow, но они должны посылать информацию других классам. Например, кнопки установки текущего инструмента должны посылать информацию в класс Canvas. Для того чтобы осуществить такое взаимодействие в Qt мы используем систему сигналов и слотов.

При нажатии на кнопку задействуется сигнал, вызывающий слот, который в свою очередь вызывает необходимые методы из других классов.

4.4 Описание класса Canvas

Класс Canvas отвечает рисование на холсте, его очистку, а также хранит информацию о всех объектах, которые находятся на холсте.

Canvas содержит методы, которые подразумевают в себе настройки размера, цвета и штриха кисти; установки выбранного инструмента в данный момент.

В классе содержатся обработчики событий мыши, которые отслеживают нажатие и отпускание кнопки мыши, а также следят за её положением.

4.5 Описание класса Figure

Класс Figure предназначен для создания геометрических примитивов. В будущем использование отдельного класса для них позволит производить над уже созданными фигурами такие действия, как: перемещение, поворот, изменение размера и удаление фигуры.

4.6 Описание класса Feedback

Класс Feedback предназначен для предоставления пользователям возможности оставлять обратную связь авторам, которая позволит в будущем улучшить продукт. Пользователи могут написать о найденных ошибках и

недочётах, а также написать свои идеи и предложения, связанные с редактором.

Окно представляет из себя три текстовых поля. Первое предназначено для имени пользователя. Второе – для адреса электронной почты, чтобы разработчики могли отправить свой ответ. Третье поле предназначено для текста отзыва.

Класс содержит в себе два метода. Первый метод срабатывает при нажатии на кнопку “Отмена” и скрывает окно. Второй же метод срабатывает при нажатии на кнопку “Ок”. В этот момент открывается текстовый файл в директории приложения, в который происходит запись всей информации окна. В будущем необходимо доработать данный класс, изменив функционал таким образом, чтобы данные не записывались в файл на компьютере пользователя, а сразу отправлялись на почту разработчикам. Для этого вероятнее всего использоваться Simple Mail Transfer Protocol (SMTP), предназначенный для передачи электронной почты в сетях TCP/IP.

4.7 Описание класса Manual

Класс Manual представляет из себя окно с руководством пользователя. Отображение текста происходит с помощью класса QTextBrowser. Данный класс отображает данные в формате html, поэтому руководство описано в гипертекстовом документе.

Руководство содержит в себе описание функциональной составляющей редактора и контакты разработчиков.

4.8 Создание инсталлятора

В качестве приложения для создания инсталлятора выбор пал на Inno Setup. Inno Setup — опенсорсный проект, код которого доступен на гитхабе. В силу бесплатности и низкого порога вхождения мой выбор остановился именно на нем, как инструменте позволившем выполнить работу быстро и качественно.

Перед описанием непосредственной секций установщика. Определим некоторые константы с помощью директивы #define:

- Константа имени приложения Name — "XPainter";
- Константа версии приложения Version — "0.0.1";
- Константа фирмы-разработчик Publisher — " ZKL";
- Константа сайта фирмы разработчика URL — " https://github.com/Luzinsan/Acquiring-primary-research-skills-dispersed";
- Константа имени исполняемого модуля ExeName — " XPainter.exe";

Тело скрипта разделяется на секции, каждая из которых несет свое функциональное назначение. Обязательная секция [Setup] задает глобальные параметры работы инсталлятора и деинсталатора. Сгенерируем уникальный идентификатор приложения (GUID), используемый для регистрации приложения в реестре Windows с помощью инструмента, расположенного в меню Tools -> Generate GUID (или используя горячую клавишу Shift + Ctrl + G) и вставим в параметр AppId. Далее указываем имя приложения, под которым оно будет установлено в системе, его версию, данные фирмы разработчика, адреса сайтов разработчика, технической поддержки и обновления. Таким образом определим такие параметры:

- AppName константой Name;
- AppVersion константой Version;
- AppPublisher константой Publisher;
- AppPublisherURL константой URL;
- AppSupportURL константой URL;
- AppUpdatesURL константой URL.

Путь, по умолчанию предлагаемый инсталлятором для установки определяем опцией DefaultDirName. При этом переменная {pf} — это путь в

каталог Program Files соответствующей разрядности. Опция DefaultGroupName определяет имя группы программы в меню «Пуск». Для указания имени приложения мы используем данное нами выше макроопределение Name, обрамляя его фигурными скобками и решеткой.

Пара опций OutputDir и OutputBaseFileName задают каталог, куда будет записан скомпилированный «setup» и его имя (без расширения). Кроме этого, указываем, где взять иконку для XPainter-setup.exe опцией SetupIconFile.

Последние опции в этой секции определяют алгоритм сжатия (LZMA) и указывают, что все файлы сжимаются одновременно, а не по отдельности (SolidCompression) что ускоряет процесс распаковки при большом количестве однотипных файлов.

В хорошем установщике должна быть поддержка нескольких языков. Включаем её в наш «setup», используя опциональную секцию [Languages]. При отсутствии данной секции будет использоваться английский язык. Каждая строка в данной секции задает один из используемых при установке языков. Синтаксис строки таков:

<имя параметра>: <значение параметра>

В качестве разделителя параметров используется точка с запятой. Параметр Name говорит за «имя» языка. Допускаются общепринятые двухбуквенные сокращения («en», «ru», «de» и так далее). Параметр MessagesFile сообщает компилятору в каком месте взять шаблон сообщений, выводимых при инсталляции. Эти шаблоны берем в каталоге компилятора Inno Setup, о чем мы сообщаем директивой compiler. Для русского языка годится Languages\Russian.isl

Параметр LicenseFile задает путь к файлу с текстом лицензии на соответствующем языке. В нашем случае параметр LicenseFile задаёт путь к файлу, содержащим условное соглашение о том, что данное приложение является приложением народного достояния, код которого может свободно распространяться в сети Интернет. Такое приложение никто не имеет права присваивать

себе в исключительную собственность, то есть ограничивать права других на использование этого приложения.

Обычно установщик предлагает нам, например, определится, хотим мы или не хотим создать ярлык на рабочем столе. Такие опции установки определяются необязательной секцией [Tasks]. Параметры:

- Name - задает имя операции — «desktopicom» — создание иконки на рабочем столе;
- Description — описание флажка с опцией, которую увидит пользователь. Конструкция {cm:<имя сообщения>} задает стандартный текст сообщения, соответствующий выбранному в начале инсталляции языку;
- GroupDescription — заголовок группы флагов с опциями;
- Flags — задает определенные действия и состояния элементов управления, в данном случае указывая, что галочка «создать ярлык на рабочем столе» должна быть снята.

Теперь укажем, какие файлы надо включить в дистрибутив и где их надо поместить при установке. Для этого используется обязательная секция [Files], где мы определяем путь исполняемого файла и прилагающиеся ресурсы с параметрами:

- Source — путь к файлу-источнику. Всё необходимое программе для работы лежит в каталоге release проекта Qt Creator;
- DestDir — каталог установки. Переменная {app} содержит путь, выбранный пользователем в окне установщика;
- Flags — разнообразные флаги. Для исполняемого файла нашего приложения: игнорирование версии программы при перезаписи исполняемого модуля, если он уже существует в системе (ignorevarsion); для остальных файлов и каталогов так же игнорируем версию, рекурсивно включаем все подкаталоги и файлы источника (recursesubdirs) и создаем подкаталоги, если их нет (createallsubdirs)

Наконец опционально укажем компилятору, где брать иконки для размещения в меню программ и на рабочем столе в секции [Icons] ключами:

- {group}, следующим параметром выступает Filename с путём {app}\{#ExeName}";
- {userdesktop} с путём Filename: {app}\{#ExeName}", где в качестве рабочего каталога выступает WorkingDir: {app}, а непосредственно путь к расположению файла-иконки укажем как IconFilename: {app}\Icons\icons8_impossible_shapes_64_8PK_icon.ico; в параметр Tasks устанавливается ключ desktopicon;

Далее жмем Ctrl + F9 и собираем инсталлятор. Начинается процесс сборки, после которого в указанном для установки пути мы получаем файл установщика.

5 Описание команды разработчиков и их функций в проекте

Наша команда состояла из трех человек, которые занимались отдельными частями нашего приложения.

Куминов Павел Алексеевич, студент каф. АСУ, гр. 430-4. Занимался проектированием приложения, разработкой класса Figure, реализацией функций манипулирования с файлами.

Завзятов Владислав Сергеевич, студент каф. АСУ, гр. 430-4. Занимался разработкой классов MainWindow, Manual, Feedback.

Лузинсан Анастасия Александровна, студентка каф. АСУ, гр. 430-2. Занималась разработкой класса Canvas, проводила маркетинговые исследования.

6 Тестирование

6.1 Тестирование кисти

Цель теста: убедиться в работоспособности кисти, проверка с разными размерами, цветами.

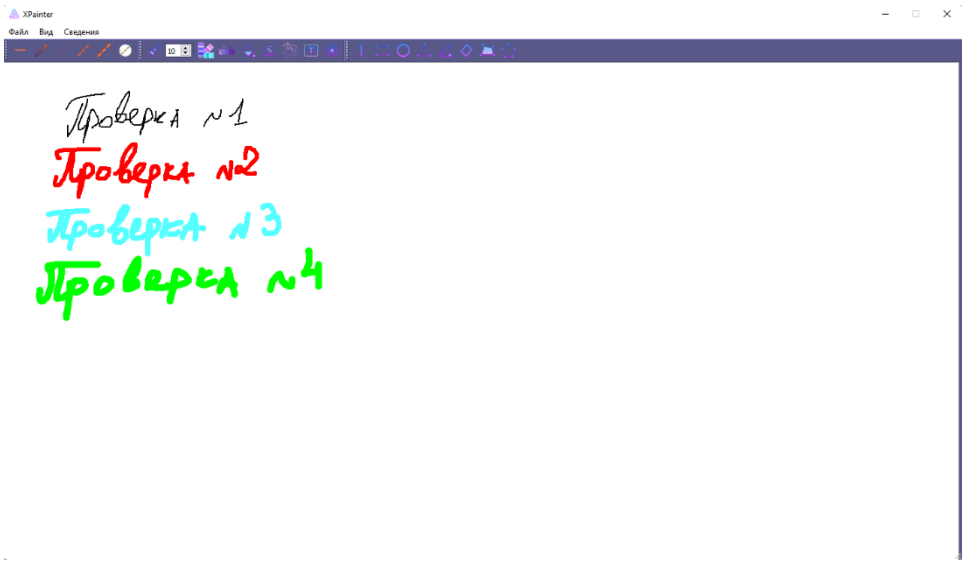

Рисунок 6.1 – тестирование инструмента “Кисть”

6.2 Тестирование стилей штриха

Цель теста: убедиться в работоспособности инструмента “Стили штриха”.

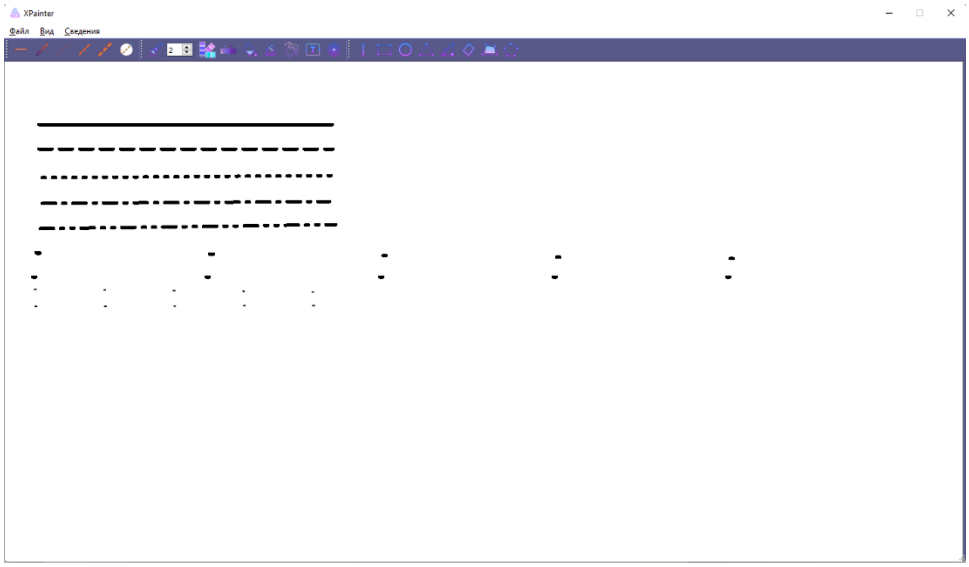

Рисунок 6.2 – тестирование инструмента “Стили штриха”

6.3 Тестирование градиента

Цель теста: убедиться в работоспособности градиента.

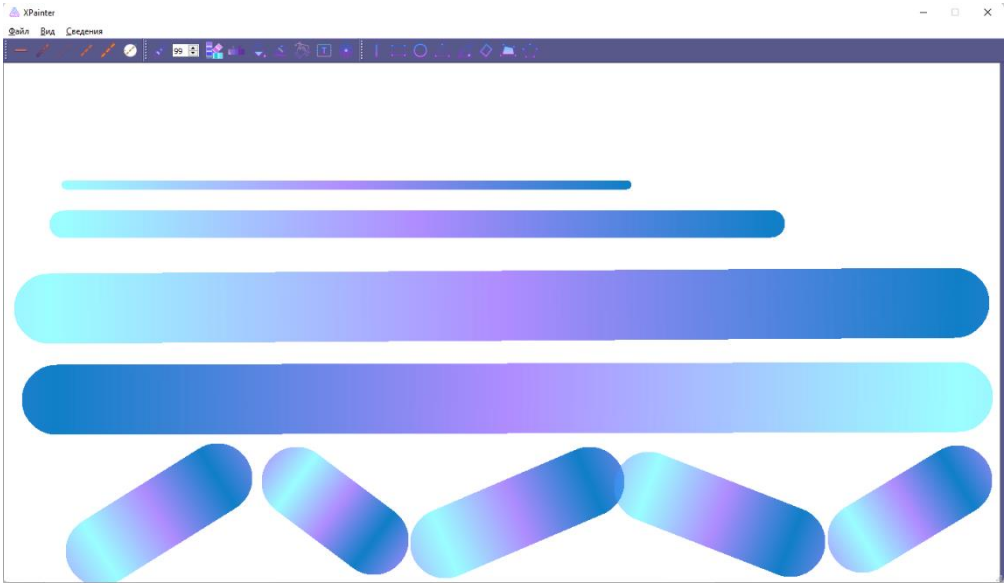

Рисунок 6.3 – тестирование градиента.

6.4 Тестирование инструмента “Текст”

Цель теста: убедиться в работоспособности инструмента “Текст”

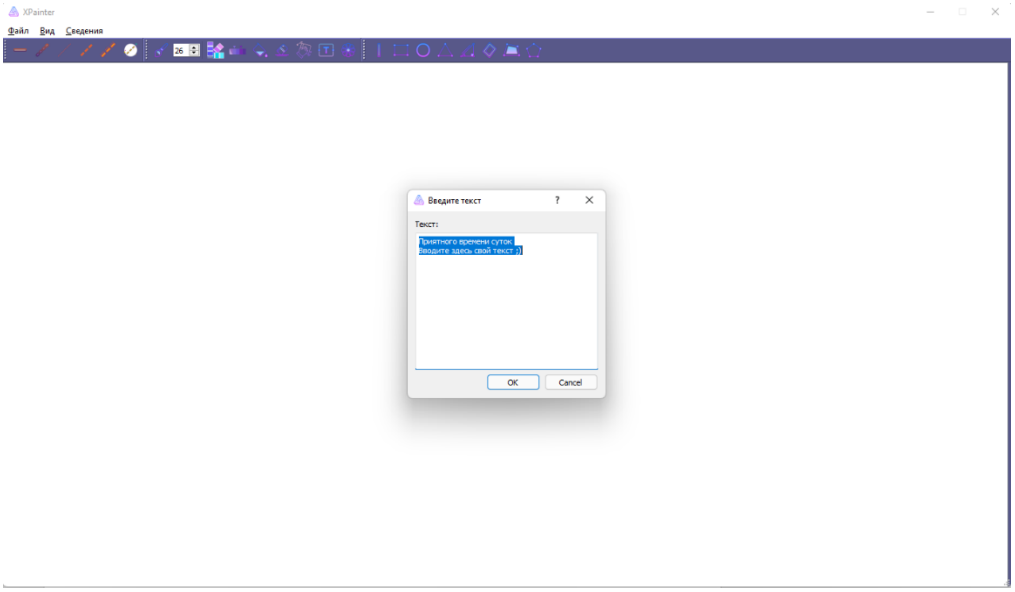

Рисунок 6.4 – окно для ввода желаемого текста

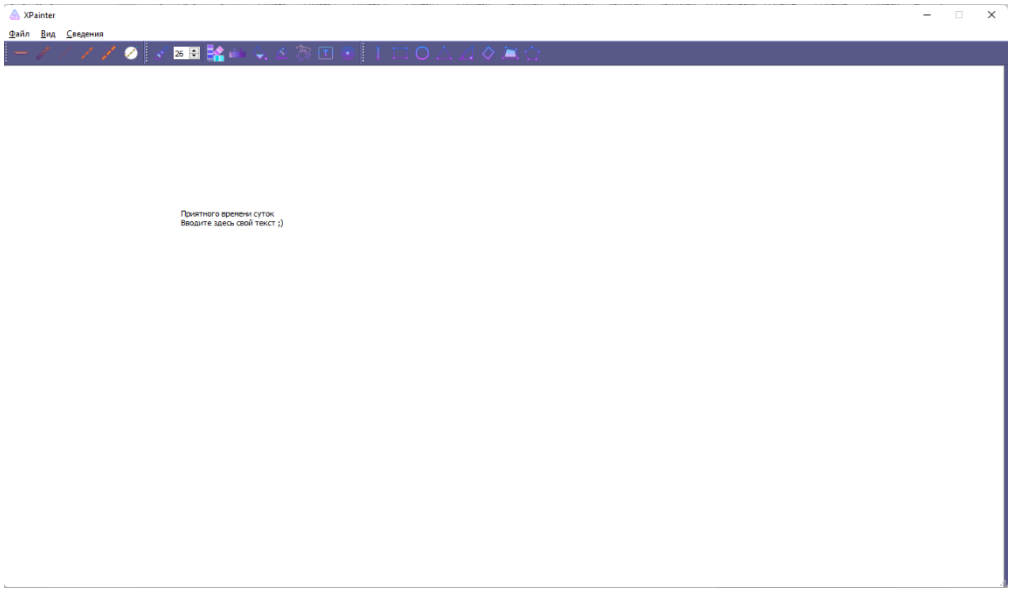

Рисунок 6.5 – окно редактора с введенным текстом

6.5 Тестирование инструмента “Дублирование”

Цель теста: убедиться в работоспособности инструмента “Дублирование”

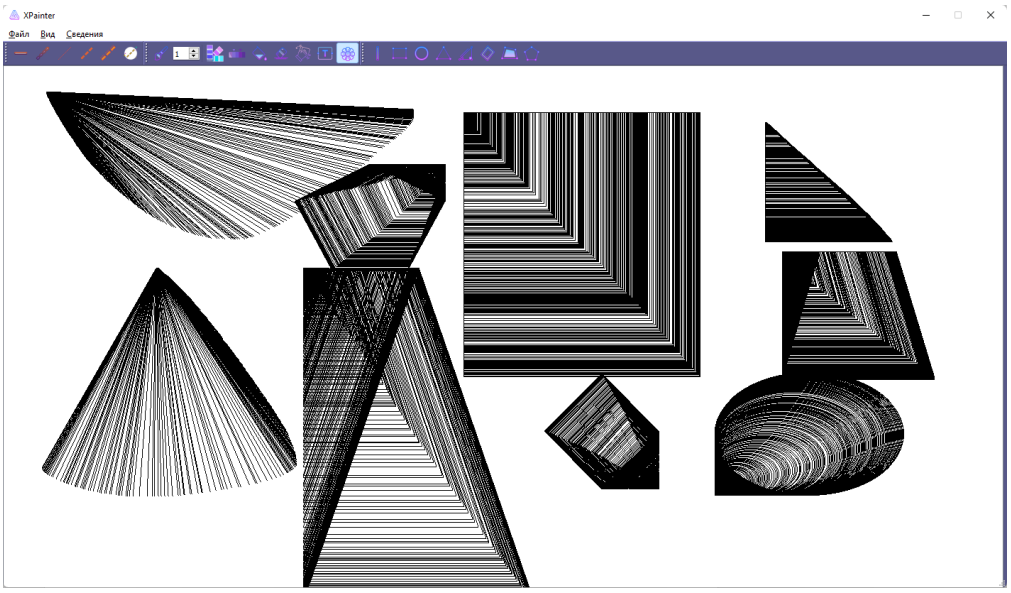

Рисунок 6.6 – тестирование инструмента “Дублирование”

6.6 Тестирование инструмента “Разлиновка”

Цель теста: убедиться в работоспособности инструмента “Разлиновка”

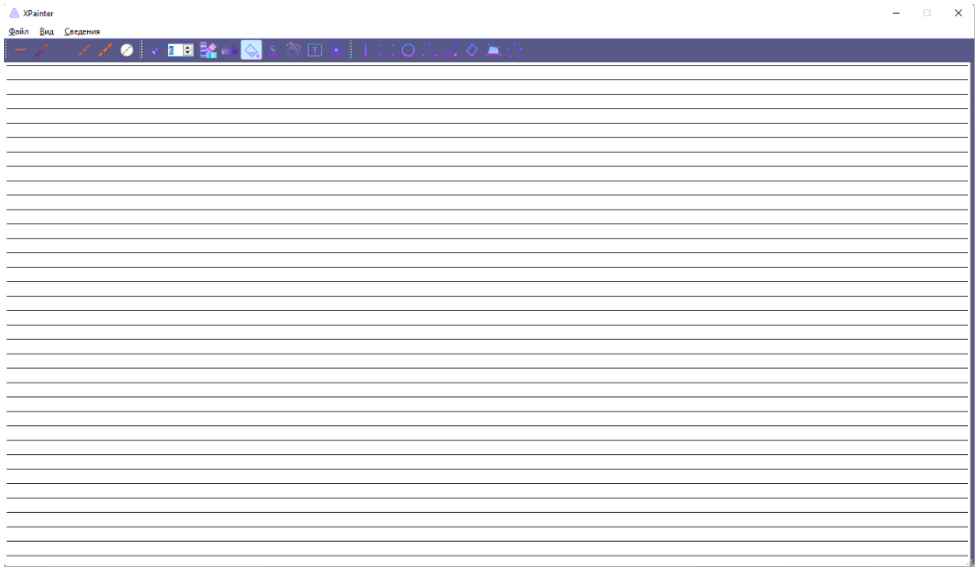

Рисунок 6.7 – тестирование инструмента “Разлиновка”

6.7 Тестирование системы обратной связи

Цель теста: убедиться в работоспособности системы обратной связи

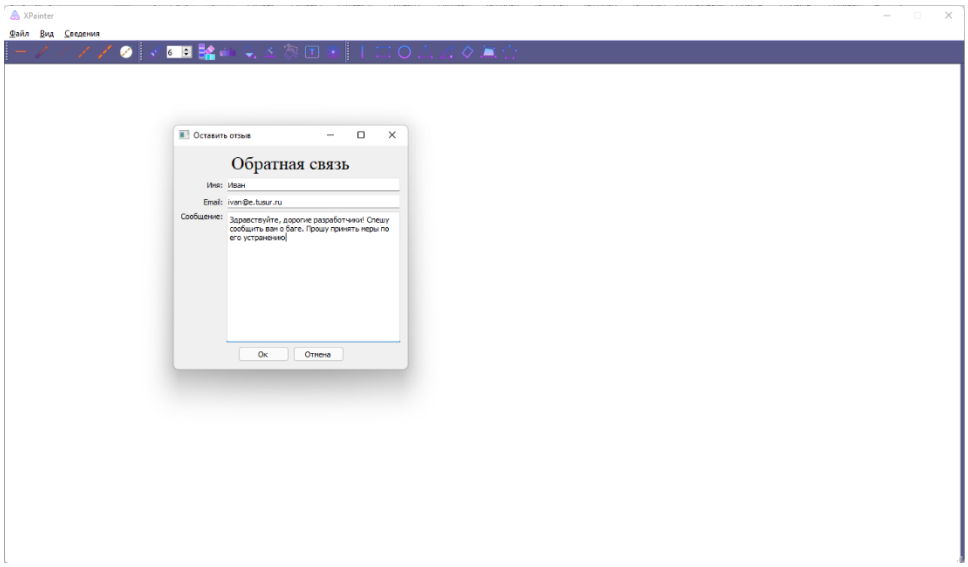

Рисунок 6.8 – окно с вводом обратной связи

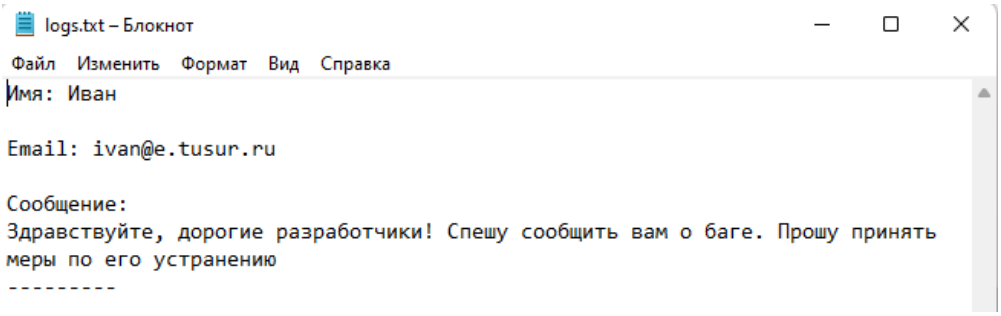

Рисунок 6.9 – сообщение в логах

Заключение

В ходе прохождения учебной практики, я получил навыки разработки приложений, а также навыки командной разработки. Изучил Фреймворк Qt, реализовал пользовательский интерфейс, изучил основы HTML, реализовал мануал и систему обратной связи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Графический редактор - Википедия https://ru.wikipedia.org/wiki/Графический_редактор
2. Векторный графический редактор кратко https://obrazovaka.ru/informatika/vektornyy-graficheskiy-redaktor.html
3. Графический редактор это что такое? Растровые и векторные графические редакторы и их инструменты https://o-dns.ru/programmy/programmy-dlya-raboty-s-grafikoj-graficheskie-redaktory-dlya-kompyutera
4. Общие сведения о языке программирования C++ http://inf-w.ru/?page_id=8655
5. Краткий обзор кроссплатформенного фреймворка Qt https://nicknixer.ru/programmirovanie/kratkij-obzor-krossplatformennogo-frejmvorka-qt/
6. Подробный обзор, настройка и установка Qt Creator https://linuxvsem.ru/programs/qt-creator
7. Сигналы и слоты в Qt https://habr.com/ru/post/50812/
8. GNU General Public License – Википедия https://ru.wikipedia.org/wiki/GNU_General_Public_License
9. Inno Setup: создание инсталлятора на примере развертывания C# приложения https://habr.com/ru/post/255807/
10. Основы HTML https://html5book.ru/osnovy-html/

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг файла с реализацией main

```
#include "mainwindow.h"
#include <QApplication>
#include <QLocale>
#include <QTranslator>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "XPainter_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            app.installTranslator(&translator);
            break;
        }
    }
    MainWindow window;
    window.setPalette(QPalette(QColor(90,90,140,250)));
    window.setWindowIcon(QIcon("Icons/icons8-impossible-shapes-64.png"));
    window.resize(1420, 800);
    window.setFixedSize(1415, 790);
    window.setWindowTitle("XPainter");
    window.show();
    return app.exec();
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг заголовочного файла класса MainWindow

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include "canvas.h"
#include "manual.h"
#include "feedback.h"
#include <QMainWindow>
#include <QGraphicsView>
#include <QComboBox>
#include <QAction>
class MainWindow : public QMainWindow
{
   Q_OBJECT
public:
   MainWindow(QWidget *parent = nullptr);
   QGraphicsView *viewer;
   Canvas *canvas;  // Объявляем кастомную графическую сцену
   QComboBox *combo;
   Manual* manualWindow = new Manual();
   Feedback* feedbackWindow = new Feedback();
public slots:
   void toggleStatusbar();
   void setColor();
   void setFigureLine();
   void setFigureRectangle();
   void setFigureCircle();
   void setFigureTriangleRectangular();
   void setFigureTriangle();
   void setFigureRhomb();
   void setFigureTrapezoid();
   void setFigurePentagon();
```

```
    void openFile();
    void saveFile();
    void createFile();
    void openManualWindow();
    void openFeedbackWindow();
private:
    QAction *viewst;
};
#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг файла с реализацией класса MainWindow

```
#include "mainwindow.h"
#include <QApplication>
#include <QMenu>
#include <QToolBar>
#include <QStatusBar>
#include <QMenuBar>
#include <QSpinBox>
#include <QColorDialog>
#include <QFileDialog>
MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent),
     canvas(new Canvas(this)) // Инициализируем графическую сцену
{
   qApp->setAttribute(Qt::AA_DontShowIconsInMenus, false);
   /*****************МЕНЮ -> ФАЙЛ************/
   QMenu *file;
   file = menuBar()->addMenu("&Файл");
   /***СОЗДАТЬ**********************/
   QPixmap newpix("Icons/icons8-new-64.png");
   QAction *newa = new QAction(newpix, "&Создать", this);
   newa->setShortcut(tr("Ctrl+N"));
   file->addAction(newa);
   connect(newa, &QAction::triggered, this, &MainWindow::createFile);
   /*************ОТКРЫТЬ***************/
   QPixmap openpix("Icons/icons8-image-file-64.png");
   QAction *open = new QAction(openpix, "&Открыть", this);
   open->setShortcut(tr("Ctrl+O"));
   file->addAction(open);
   connect(open, &QAction::triggered, this, &MainWindow::openFile);
   /*************СОХРАНИТЬ*****************/
```

```
QPixmap savepix("Icons/icons8-save-64.png");
QAction *save = new QAction(savepix, "&Сохранить", this);
save->setShortcut(tr("Ctrl+S"));
file->addAction(save);
connect(save, &QAction::triggered, this, &MainWindow::saveFile);
file->addSeparator();
/********************ВЫХОД***/
QPixmap quitpix("Icons/icons8-shutdown-64.png");
QAction *quit = new QAction(quitpix, "&Выход", this);
quit->setShortcut(tr("Ctrl+Q"));
file->addAction(quit);
connect(quit, &QAction::triggered, this, &QApplication::quit);
/********************МЕНЮ -> ВИД***************/
QMenu *sideb;
sideb = menuBar()->addMenu("&Вид");
/****ЛИНЕЙКИ****************/
QPixmap backgroundpix("Icons/icons8-paint-roller-64.png");
QAction *background = new QAction(backgroundpix, "&Задний фон", this);
background->setShortcut(tr("Ctrl+B"));
background->setCheckable(true);
sideb->addAction(background);
connect(background, &QAction::triggered, canvas, &Canvas::setBackground);
/************ЛИНИИ СЕТКИ**********/
QPixmap gridLinespix("Icons/icons8-line-width-64.png");
QAction *gridLines = new QAction(gridLinespix, "&Разлиновка", this);
gridLines->setShortcut(tr("Ctrl+G"));
gridLines->setCheckable(true);
sideb->addAction(gridLines);
connect(gridLines, &QAction::triggered, canvas, &Canvas::drawGridLines);
/*****************МЕНЮ -> СВЕДЕНИЯ***************/
QMenu *info;
info = menuBar()->addMenu("&Сведения");
/********СВОЙСТВА**********************/
QPixmap propertiespix("Icons/icons8-user-manual-64.png");
```

```
        QAction *manualp = new QAction(propertiespix, "&Мануал", this);
        manualp->setShortcut(tr("Ctrl+M"));
        info->addAction(manualp);
        connect(manualp, &QAction::triggered, this, &MainWindow::openManualWindow);
        /***************FEEDBACK*************/
        QPixmap feedbackpix("Icons/icons8-feedback-64.png");
        QAction *feedbackp = new QAction(feedbackpix, "&Оставить отзыв", this);
        feedbackp->setShortcut(tr("Ctrl+F"));
        info->addAction(feedbackp);
        connect(feedbackp, &QAction::triggered, this, &MainWindow::openFeedbackWin-
dow);
        /**********ГРАФИЧЕСКАЯ СЦЕНА************/
        viewer = new QGraphicsView(this);
        canvas->setSceneRect(0,0,1400,780);
        viewer->setFixedSize(1410,790);
        viewer->setScene(canvas);
        /**************ПАНЕЛЬ ИНСТРУМЕНТОВ -> СТИЛЬ ШТРИХА*******/
        QToolBar *dashoffset = addToolBar("&Стиль штриха");
        /*ПРЯМАЯ ЛИНИЯ*************************************/
        QPixmap solidlinepix("Icons/icons8-horizontal-line-64.png");
        QAction *solidline = dashoffset->addAction(QIcon(solidlinepix), "&Прямая линия");
        solidline->setCheckable(false);
        connect(solidline, &QAction::triggered, canvas, &Canvas::setSolidLine);
        /*************ПУНКТИРНАЯ ЛИНИЯ**********************/
        QPixmap dashlinepix("Icons/icons8-dashed-line-64.png");
        QAction *dashline = dashoffset->addAction(QIcon(dashlinepix), "&Пунктирная ли-
ния");
        dashline->setCheckable(false);
        connect(dashline, &QAction::triggered, canvas, &Canvas::setDashLine);
        /******************ТОЧЕЧНАЯ ЛИНИЯ*****************/
        QPixmap dotlinepix("Icons/icons8-dashed-line-80.png");
        QAction *dotline = dashoffset->addAction(QIcon(dotlinepix), "&Точечная линия");
        dotline->setCheckable(false);
        connect(dotline, &QAction::triggered, canvas, &Canvas::setDotLine);
```

```
        /*****************ЛИНИЯ ТИРЕ-ТОЧКА***********/
        QPixmap dashdotlinepix("Icons/icons8-dashed-line-100.png");
        QAction *dashdotline = dashoffset->addAction(QIcon(dashdotlinepix), "&Линия тире-точка");
        dashdotline->setCheckable(false);
        connect(dashdotline, &QAction::triggered, canvas, &Canvas::setDashDotLine);
        /***************ЛИНИЯ ТИРЕ-ТОЧКА-ТОЧКА**********************/
        QPixmap dashdotdotlinepix("Icons/icons8-dashed-line-65.png");
        QAction *dashdotdotline = dashoffset->addAction(QIcon(dashdotdotlinepix), "&Линия тире-точка-точка");
        dashdotdotline->setCheckable(false);
        connect(dashdotdotline, &QAction::triggered, canvas, &Canvas::setDashDotDotLine);
        /*****************ШИРОКИЙ ПУНКТИР*/
        QPixmap cusromlinepix("Icons/icons8-dashed-line-50.png");
        QAction *cusromline = dashoffset->addAction(QIcon(cusromlinepix), "&Широкий пунктир");
        cusromline->setCheckable(false);
        connect(cusromline, &QAction::triggered, canvas, &Canvas::setCustomLine);
        /**************ПАНЕЛЬ ИНСТРУМЕНТОВ -> ОСНОВНАЯ**********/
        QToolBar *toolbar = addToolBar("&Панель инструментов");
        /*КИСТЬ*************************/
        QPixmap brushpix("Icons/icons8-paint-64.png");
        QAction *brush = toolbar->addAction(QIcon(brushpix), "&Кисть");
        brush->setShortcut(tr("Ctrl+P"));
        brush->setCheckable(false);
        connect(brush, &QAction::triggered, canvas, &Canvas::setToolLine);
        /*****ШИРИНА ЛИНИИ**********************/
        QSpinBox *spinbox = new QSpinBox(this);
        toolbar->addWidget(spinbox);
        connect(spinbox, static_cast<void (QSpinBox::*)(int)>(&QSpinBox::valueChanged),
                canvas, static_cast<void (Canvas::*)(int)>(&Canvas::setSize));
        toolbar->addSeparator();
        /*****************ПАЛИТРА******************/
        QPixmap palettepix("Icons/icons8-color-palette-64.png");
```

```
QAction *palette = toolbar->addAction(QIcon(palettepix), "&Палитра");
palette->setShortcut(tr("Ctrl+C"));
palette->setCheckable(false);
connect(palette, &QAction::triggered, this, &MainWindow::setColor);
/*****************ГРАДИЕНТ*****************/
QPixmap gradientpix("Icons/icons8-gradient-64.png");
QAction *gradient = toolbar->addAction(QIcon(gradientpix), "&Градиент");
gradient->setShortcut(tr("Ctrl+>"));
gradient->setCheckable(true);
connect(gradient, &QAction::triggered, canvas, &Canvas::setGradient);
/*****************ЗАПОЛНЕНИЕ ФИГУР*************************/
QPixmap fillpix("Icons/icons8-fill-color-64.png");
QAction *fill = toolbar->addAction(QIcon(fillpix), "&Заполнение");
fill->setShortcut(tr("Ctrl+<"));
fill->setCheckable(true);
connect(fill, &QAction::triggered, canvas, &Canvas::setFillFigure);
/**********************ЛАСТИК*******************/
QPixmap eraserpix("Icons/icons8-eraser-64.png");
QAction *eraser = toolbar->addAction(QIcon(eraserpix), "&Ластик");
eraser->setShortcut(tr("Ctrl+E"));
eraser->setCheckable(false);
connect(eraser, &QAction::triggered, canvas, &Canvas::setToolEraser);
/******************УДАЛИТЬ ВСЁ******************/
QPixmap deleteallpix("Icons/icons8-clean-64.png");
QAction *deleteall = toolbar->addAction(QIcon(deleteallpix), "&Удалить всё");
deleteall->setShortcut(tr("Ctrl+D"));
deleteall->setCheckable(false);
connect(deleteall, &QAction::triggered, canvas, &Canvas::deleteAll);
/*******************ТЕКСТ**********************/
QPixmap textpix("Icons/icons8-text-box-64.png");
QAction *text = toolbar->addAction(QIcon(textpix), "&Текст");
text->setShortcut(tr("Ctrl+T"));
text->setCheckable(false);
connect(text, &QAction::triggered, canvas, &Canvas::setToolText);
```

```
        /*******************ДУБЛИРОВАНИЕ*******************/
        QPixmap duplicationpix("Icons/icons8-geometric-flowers-64.png");
        QAction *duplication = toolbar->addAction(QIcon(duplicationpix),
"&Дублирование");
        duplication->setShortcut(tr("Ctrl+W"));
        duplication->setCheckable(true);
        connect(duplication, &QAction::triggered, canvas, &Canvas::setToolDuplication);
        /*******************ПАНЕЛЬ ИНСТРУМЕНТОВ -> ФИГУРЫ*********/
        QToolBar *figures = addToolBar("Фигуры");
        /*ПРЯМАЯ*************************/
        QPixmap linepix("Icons/icons8-vertical-line-64.png");
        QAction *line = figures->addAction(QIcon(linepix), "&Прямая");
        line->setShortcut(tr("Ctrl+L"));
        line->setCheckable(false);
        connect(line, &QAction::triggered, this, &MainWindow::setFigureLine);
        /*******КВАДРАТ************************/
        QPixmap rectpix("Icons/icons8-rectangular-64.png");
        QAction *rect = figures->addAction(QIcon(rectpix), "&Квадрат");
        rect->setShortcut(tr("Ctrl+R"));
        rect->setCheckable(false);
        connect(rect, &QAction::triggered, this, &MainWindow::setFigureRectangle);
        /**************ОКРУЖНОСТЬ******************/
        QPixmap circlepix("Icons/icons8-circle-64.png");
        QAction *circle = figures->addAction( QIcon(circlepix), "&Окружность");
        circle->setShortcut(tr("Ctrl+O"));
        circle->setCheckable(false);
        connect(circle, &QAction::triggered, this, &MainWindow::setFigureCircle);
        /************ПРОИЗВОЛЬНЫЙ ТРЕУГОЛЬНИК************/
        QPixmap trianglepix("Icons/icons8-triangle-64.png");
        QAction *triangle = figures->addAction( QIcon(trianglepix),"&Треугольник");
        triangle->setShortcut(tr("Ctrl+T"));
        triangle->setCheckable(false);
        connect(triangle, &QAction::triggered, this, &MainWindow::setFigureTriangle);
        /*************ПРЯМОУГОЛЬНЫЙ ТРЕУГОЛЬНИК***********/
```

```
        QPixmap trianglerectpix("Icons/icons8-trigonometry-64.png");
        QAction *trianglerect = figures->addAction( QIcon(trian-
glerectpix),"&Прямоугольный треугольник");
        trianglerect->setShortcut(tr("Ctrl+T+R"));
        trianglerect->setCheckable(false);
        connect(trianglerect, &QAction::triggered, this, &MainWindow::setFigureTrian-
gleRectangular);
        /************************РОМБ**********************/
        QPixmap rhombpix("Icons/icons8-rhomboid-shape-64.png");
        QAction *rhomb = figures->addAction(QIcon(rhombpix), "&Ромб");
        rhomb->setShortcut(tr("Ctrl+Shift+R"));
        rhomb->setCheckable(false);
        connect(rhomb, &QAction::triggered, this, &MainWindow::setFigureRhomb);
        /**********************ТРАПЕЦИЯ**************/
        QPixmap trapezoidpix("Icons/icons8-irregular-quadrilateral-64.png");
        QAction *trapezoid = figures->addAction(QIcon(trapezoidpix),  "&Трапеция");
        trapezoid->setShortcut(tr("Ctrl+Shift+T"));
        trapezoid->setCheckable(false);
        connect(trapezoid, &QAction::triggered, this, &MainWindow::setFigureTrapezoid);
        /*************ПЯТИУГОЛЬНИК**/
        QPixmap pentagonpix("Icons/icons8-pentagon-64.png"); // Тоже найти иконку
        QAction *pentagon = figures->addAction(QIcon(pentagonpix),  "&Пятиугольник");
        pentagon->setShortcut(tr("Ctrl+Shift+T"));
        pentagon->setCheckable(false);
        connect(pentagon, &QAction::triggered, this, &MainWindow::setFigurePentagon);
        /**********НИЖНЯЯ ПАНЕЛЬ**********/
        setCentralWidget(viewer);
        statusBar()->showMessage("Готово");
    }
    void MainWindow::toggleStatusbar()
    {
        if (viewst->isChecked())
            statusBar()->show();
        else statusBar()->hide();
```

```
}
void MainWindow::setColor()
{
   QColor color = QColorDialog::getColor(Qt::black, this, "Палитра");
   if(color.isValid())
   {
      canvas->pen.setBrush(color);
      canvas->currentColor = color;
      canvas->gradient = false;
   }
   else canvas->pen.setBrush(canvas->currentColor);
}

void MainWindow::setFigureLine()
{
   canvas->setToolFigure();
   canvas->setLine();
}
void MainWindow::setFigureRectangle()
{
   canvas->setToolFigure();
   canvas->setRectangle();
}
void MainWindow::setFigureCircle()
{
   canvas->setToolFigure();
   canvas->setCircle();
}
void MainWindow::setFigureTriangle()
{
   canvas->setToolFigure();
   canvas->setTriangle();
}
void MainWindow::setFigureTriangleRectangular()
```

```
{
   canvas->setToolFigure();
   canvas->setTriangleRectangular();
}
void MainWindow::setFigureRhomb()
{
   canvas->setToolFigure();
   canvas->setRhomb();
}
void MainWindow::setFigureTrapezoid()
{
   canvas->setToolFigure();
   canvas->setTrapezoid();
}
void MainWindow::setFigurePentagon()
{
   canvas->setToolFigure();
   canvas->setPentagon();
}
void MainWindow::openFile()
{
   QString fileName = QFileDialog::getOpenFileName(this,
                                    tr("Открыть файл"),
                                    "C://",
                                    "Images (*.png, *.jpg)");
   canvas->loadPicture(fileName);
}
void MainWindow::saveFile()
{
   QString fileName = QFileDialog::getSaveFileName(this,
                                    tr("Save file"),
                                    QDir::homePath(),
                                    "Images (*.png, *.jpg)");
   canvas->savePicture(fileName);
```

```
}
void MainWindow::createFile()
{
   saveFile();
   canvas->deleteAll();
}
void MainWindow::openManualWindow()
{
   manualWindow->show();
}

void MainWindow::openFeedbackWindow()
{
   feedbackWindow->show();
}
```

ПРИЛОЖЕНИЕ Г

(обязательное)

Листинг заголовочного файла класса Manual

```
#ifndef MANUAL_H
#define MANUAL_H
#include <QWidget>
namespace Ui {class Manual;}
class Manual : public QWidget
{
   Q_OBJECT
public:
   explicit Manual(QWidget *parent = nullptr);
   ~Manual();
private:
   Ui::Manual *ui;
};
#endif // MANUAL_H
```

ПРИЛОЖЕНИЕ Д

(обязательное)

Листинг файла с реализацией класса Manual

```
#include "ui_manual.h"
#include "manual.h"
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QTextBrowser>
#include <QIcon>
Manual::Manual(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Manual)
{
    ui->setupUi(this);
    this->setWindowIcon(QIcon("Icons/icons8-show-property-64.png"));
    this->setWindowTitle("Мануал");
    this->setFixedSize(860, 640);

    QGridLayout *grid = new QGridLayout();
    grid->setVerticalSpacing(5);
    grid->setHorizontalSpacing(5);

    QTextBrowser* textBrowser = new QTextBrowser(this);
    grid->addWidget(textBrowser, 0, 0, 0, 0);
    const QString &page = "Manual/About.html";
    textBrowser->resize(840, 620);
    textBrowser->setSource(page);
}
Manual::~Manual()
{ delete ui; }
```

ПРИЛОЖЕНИЕ Е

(обязательное)

Листинг заголовочного файла класса Feedback

```
#ifndef FEEDBACK_H
#define FEEDBACK_H
#include <QWidget>
#include <QLabel>
#include <QLineEdit>
#include <QTextEdit>
class Feedback: public QWidget
{
public:
   Feedback(QWidget *parent = 0);
private:
   QLabel *name, *email, *review;
   QLineEdit *nameline, *emailline;
   QTextEdit *reviewline;
private slots:
   void close();
   void logging();
};
#endif // FEEDBACK_H
```

ПРИЛОЖЕНИЕ Ж

(обязательное)

Листинг файла с реализацией класса Feedback

```
#include "feedback.h"
#include <QApplication>
#include <QGridLayout>
#include <QPushButton>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QFile>
#include <QTextStream>
Feedback::Feedback(QWidget *parent): QWidget(parent)
{
   this->setWindowIcon(QIcon("Icons/icons8-layers-64.png"));
   this->setWindowTitle("Оставить отзыв");
   QVBoxLayout *vbox = new QVBoxLayout(this);
   QGridLayout *grid = new QGridLayout();
   grid->setVerticalSpacing(5);
   grid->setHorizontalSpacing(5);
   QLabel *title = new QLabel("Обратная связь", this);
   title->setFont(QFont("Times New Roman", 20));
   title->setAlignment(Qt::AlignCenter | Qt::AlignTop);
   grid->addWidget(title, 0, 0, 1, 2);
   name = new QLabel("Имя:", this);
   name->setAlignment(Qt::AlignRight | Qt::AlignVCenter);
   grid->addWidget(name, 1, 0, 1, 1);
   nameline = new QLineEdit(this);
   grid->addWidget(nameline, 1, 1, 1, 1);
   email = new QLabel("Email:", this);
   email->setAlignment(Qt::AlignRight | Qt::AlignVCenter);
   grid->addWidget(email, 2, 0, 1, 1);
   emailline = new QLineEdit(this);
   grid->addWidget(emailline, 2, 1, 1, 1);
```

```
    review = new QLabel("Сообщение:", this);
    review->setAlignment(Qt::AlignRight | Qt::AlignTop);
    grid->addWidget(review, 3, 0, 1, 1);
    reviewline = new QTextEdit(this);
    grid->addWidget(reviewline, 3, 1, 3, 1);
    vbox->addLayout(grid);
    vbox->addStretch(1);
    // Кнопки Ок и Отмена
    QHBoxLayout *hbox = new QHBoxLayout();
    QPushButton *okbtn = new QPushButton("Ок", this);
    connect(okbtn, &QPushButton::clicked, this, &Feedback::logging);
    hbox->addWidget(okbtn, 0, Qt::AlignRight);
    QPushButton *quitbtn = new QPushButton("Отмена", this);
    connect(quitbtn, &QPushButton::clicked, this, &Feedback::close);
    hbox->addWidget(quitbtn, 0, Qt::AlignLeft);
    vbox->addLayout(hbox);
    setLayout(vbox);
}
void Feedback::close()
{ this->hide(); }
void Feedback::logging()
{
    QFile file("logs.txt");
    if(file.open(QIODevice::Append))
    {
        QTextStream stream(&file);
        stream << name->text() << " " << nameline->text() << "\n\n" <<
                email->text() << " " << emailline->text() << "\n\n" <<
                review->text() << "\n" << reviewline->toPlainText() <<
                "\n---------\n";
        file.close();
    }
    close();
}
```

ПРИЛОЖЕНИЕ И

(обязательное)

Листинг HTML файла с руководством пользователя

```
<!DOCTYPE html>
  <head>
  </head>
  <body>
    <div style="display: grid; column-gap: 16px; grid-template-columns: repeat(9, minmax(0, 1fr));" >
      <div style="grid-column-start: 2; align-self: center; margin-left: 50%;">
        <a href = "https://github.com/Luzinsan/Acquiring-primary-research-skills-dispersed" >
          <img src="Icons/icons8-impossible-shapes-64.png"
            style="width: 45px; height: 45px;" />
        </a>
      </div>
      <div style="grid-column-start: 3; grid-column-end: 9;">
          <h1 style="font-family: 'Times New Roman', Times, serif">
            Вас приветствует мануал по "XPaint".
          </h1>
      </div>
      <div style="grid-column-start: 2; grid-column-end: 8;">
        <p style="font-weight: 600;">
          Меню «Вид» включает в себя:
        </p>
      </div>
      <div style="grid-column-start: 3; grid-column-end: 8;">
        <ul type="disc">
          <li>«Задний фон»</li>
          <div style="text-align:justify; margin-left: 20px;">
            <p>
              Позволяет залить холст градиентом.
            </p>
```

```
            </div>
            <li>«Разлиновка»</li>
            <div style="text-align:justify; margin-left: 20px;">
              <p>
                Позволяет задать разлиновку холста. <br>
                Расстояние между линииями зависит от размера кисти.
              </p>
            </div>
          </ul>
        </div>
        <div style="grid-column-start: 2; grid-column-end: 8;">
          <p style="font-weight: 600;">
            Панель инструментов «Стили штриха» включает в себя:
          </p>
        </div>
        <div style="grid-column-start: 3; grid-column-end: 10;">
          <p>
            Данные стили применяются к инструментам «Кисть», «Фигуры», «Разлиновка».
          </p>
        </div>
        <div style="grid-column-start: 3; grid-column-end: 8;">
          <ul type="disc">
            <li>Прямую линию</li>
            <li>Пунктирную линию</li>
            <li>Точечную линию</li>
            <li>Линию тире-точка</li>
            <li>Линию тире-точка-точка</li>
            <li>Широкий пунктир</li>
          </ul>
        </div>
        <div style="grid-column-start: 2; grid-column-end: 8;">
          <p style="font-weight: 600;">
            Панель инструментов «Главная» включает в себя:
```

```
            </p>
        </div>
        <div style="grid-column-start: 3; grid-column-end: 9;">
            <ul type="disc">
                <li>Инструмент «Кисть»</li>
                <div style="text-align:justify; margin-left: 20px;">
                    <p>
                        «Кисть» является основным инструментом для рисования. <br>
                        К нему можно применить «Стили штриха», изменить размер в окне задания размера кисти,
                        изменить цвет через инструмент «Палитра». <br>
                        После применения инструмента «Дублирование» инструмент «Кисть»  рисует только прямые,
                        начала которых находятся в одной точке.
                    </p>
                </div>
                <li>Окно задания размера кисти</li>
                <div style="text-align:justify; margin-left: 20px;">
                    <p>
                        Позволяет задать размер кисти.
                    </p>
                </div>
                <li>Инструмент «Палитра»</li>
                <div style="text-align:justify; margin-left: 20px;">
                    <p>
                        Позволяет выбрать один из стандартных цветов или задать свой через цветовую панель или HTML-код.
                    </p>
                </div>
                <li>Инструмент «Ластик»</li>
                <div style="text-align:justify; margin-left: 20px;">
                    <p>
                        Позволяет стирать ваш "рисунок", используя кисть.
                    </p>
```

```
            </div>
            <li>Инструмент очистки всего холста</li>
            <div style="text-align:justify; margin-left: 20px;">
              <p>
                Позволяет полностью очистить холст от всего.
              </p>
            </div>
            <li>Инструмент «Текст»</li>
            <div style="text-align:justify; margin-left: 20px;">
              <p>
                Инструмент «Текст» позволяет выводить текст на холст. Цвет текста зависит от цвета выбранного на «Палитре».
              </p>
            </div>
            <li>Инструмент «Дублирование».</li>
            <div style="text-align:justify; margin-left: 20px;">
              <p>
                Инструмент «Дублирование» дублирует фигуры при рисовании.
              </p>
            </div>
          </ul>
        </div>
        <div style="grid-column-start: 2; grid-column-end: 8;">
          <p style="font-weight: 600;">
            Панель инструментов «Фигуры» включает в себя:
          </p>
        </div>
        <div style="grid-column-start: 3; grid-column-end: 8;">
          <ul type="disc">
            <li>Инструмент «Прямая»</li>
            <li>Инструмент «Четырехугольник»</li>
            <li>Инструмент «Окружность»</li>
            <li>Инструмент «Треугольник»</li>
            <li>Инструмент «Прямоугольный треугольник»</li>
```

```
                    <li>Инструмент «Ромб»</li>
                    <li>Инструмент «Трапеция»</li>
                    <li>Инструмент «Пятиугольник»</li>
                  </ul>
                </div>
                <div style="grid-column-start: 6; grid-column-end: 9;">
                  <p style="font-family: 'Times New Roman', Times, serif;">
                    Спасибо, что используете наш продукт.<br>
                    Наша команда разработчиков:
                  </p>
                </div>
                <div style="grid-column-start: 6; grid-column-end: 9;">
                  <p style="font-family: 'Times New Roman', Times, serif;">
                    Лузинсан Анастасия Александровна, ТУСУР, АСУ, гр. 430-2.
                  </p>
                  <p style="font-family: 'Times New Roman', Times, serif;">
                    Куминов Павел Алексеевич, ТУСУР, АСУ, гр. 430-4.
                  </p>
                  <p style="font-family: 'Times New Roman', Times, serif;">
                    Завзятов Владислав Сергеевич, ТУСУР, АСУ, гр. 430-4.
                  </p>
                                    <p style="font-family: 'Times New Roman', Times, serif;"
align = "center">
                    2021 год.
                  </p>
                </div>
              </div>
            </body>
          </html>
```

ПРИЛОЖЕНИЕ К

(обязательное)

Листинг скрипта установщика

```
; Имя приложения
#define   Name      "XPainter"
; Версия приложения
#define   Version    "0.0.1"
; Фирма-разработчик
#define   Publisher  "ZKL"
; Сайт фирмы разработчика
#define    URL          "https://github.com/Luzinsan/Acquiring-primary-research-skills-dis-
persed"
; Имя исполняемого модуля
#define   ExeName    "XPainter.exe"

[Setup]
; Уникальный идентификатор приложения,
;сгенерированный через Tools -> Generate GUID
AppId={{D8527779-796C-4391-8CE1-44DD996934AD}
; Прочая информация, отображаемая при установке
AppName={#Name}
AppVersion={#Version}
AppPublisher={#Publisher}
AppPublisherURL={#URL}
AppSupportURL={#URL}
AppUpdatesURL={#URL}
; Путь установки по-умолчанию
DefaultDirName={pf}\{#Name}
; Имя группы в меню "Пуск"
DefaultGroupName={#Name}
; Каталог, куда будет записан собранный setup и имя исполняемого файла
OutputDir=U:\XPainter-setup
OutputBaseFileName=XPainter-setup
```

```
; Файл иконки
SetupIconFile=U:\ProjectQt\for_build\icons8_impossible_shapes_64_8PK_icon.ico
; Параметры сжатия
Compression=lzma
SolidCompression=yes

[Languages]
Name: "russian"; MessagesFile: "compiler:Languages\Russian.isl"; LicenseFile: "U:\ProjectQt\build-ZKL-Desktop_Qt_5_13_2_MinGW_32_bit-Release\release\Manual\About.txt"

[Tasks]
; Создание иконки на рабочем столе
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]
; Исполняемый файл
Source: "U:\ProjectQt\build-ZKL-Desktop_Qt_5_13_2_MinGW_32_bit-Release\release\XPainter.exe"; DestDir: "{app}"; Flags: ignoreversion
; Прилагающиеся ресурсы
Source: "U:\ProjectQt\build-ZKL-Desktop_Qt_5_13_2_MinGW_32_bit-Release\release\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs

[Icons]
Name: "{group}\{#Name}"; Filename: "{app}\{#ExeName}"
Name: "{userdesktop}\{#Name}"; Filename: "{app}\{#ExeName}"; WorkingDir: "{app}";IconFilename: "{app}\Icons\icons8_impossible_shapes_64_8PK_icon.ico"; Tasks: desktopicon;
```