Министерство общего и профессионального образования Ростовской области

государственное бюджетное профессиональное образовательное учреждение

Ростовской области

«Таганрогский авиационный колледж имени В.М. Петлякова»

**КУРСОВАЯ РАБОТА**

**на тему:**

**«Создание графического редактора»**

**по дисциплине: МДК.02.01 «Технология разработки программного обеспечения»**

**специальность 09.02.07 «Информационные системы и программирование»**

**Выполнил:**

**студент группы П-21**

**Путря Никита Витальевич**

**Проверил:**

**преподаватель: Швец Андрей Васильевич (Ф.И.О.)**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)**

**Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2024 г.**

Содержание

[Введение 3](#_Toc164931004)

[1. Общая часть 4](#_Toc164931005)

[1.1 Анализ предметной области 4](#_Toc164931006)

[1.2. Постановка задачи 13](#_Toc164931007)

[2. Технология разработки программного продукта (технологическая часть) 14](#_Toc164931008)

[2.1. Анализ входных и выходных данных 14](#_Toc164931009)

[2.2. Инфологическая модель 15](#_Toc164931010)

[2.3 Выбор и обоснование программных средств 16](#_Toc164931011)

[2.4 Выбор и обоснование аппаратных средств разработки 18](#_Toc164931012)

[3. Описание программных модулей 19](#_Toc164931013)

[3.1. Структура и алгоритм работы программного продукта 19](#_Toc164931014)

[3.2 Инструментарий разработки программного продукта 20](#_Toc164931015)

[3.3. Интерфейс программного продукта 21](#_Toc164931016)

[4. Экспериментальная часть 22](#_Toc164931017)

[4.1 Назначение испытаний 22](#_Toc164931018)

[Заключение 23](#_Toc164931019)

[Список использованных источников 24](#_Toc164931020)

[Листинг 26](#_Toc164931021)

[Программный код графического редактора. 26](#_Toc164931022)

# Введение

Высокоуровневый язык программирования — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта подобных языков — абстракция, то есть введение смысловых конструкций, которые кратко описывают такие структуры и операции над ними, описания которых на любом низкоуровневом языке программирования будут длинны и сложны для понимания.

**Python** — высокоуровневый язык программирования общего назначения. Язык ориентирован на повышение производительности разработчика и читаемости кода, путем некого «упрощения» процесса программирования.

**Целью курсовой работы** является разработка программы «Графический редактор», которая позволяет пользователю просматривать и редактировать графические файлы.

**Задачи курсовой работы:**

⦁ изучить методическую литературу по технологии разработки программных продуктов;

⦁ изучить теоретические аспекты среды программирования Pyton;

⦁ рассмотреть палитру свойств формы;

⦁ определиться со средством разработки приложения;

⦁ подготовить дизайн-проект приложения;

организовать удобную навигацию в программе;

**Объект исследования:** язык программирования, на котором будет разрабатываться графический редактор.

**Предмет исследования:** личные особенности разработки на языке Pyton.

**Гипотеза курсовой работы:** В основу разработки была взята программа «Графический редактор» на языке Pyton. Реализовать правильный алгоритм работы, правильную последовательность действий и полностью разработать методику создания.

# 1. Общая часть

# 1.1 Анализ предметной области

Под графической информацией мы понимаем всю совокупность информации, которая нанесена на самые различные носители — бумагу, пленку, кальку, картон, холст, оргалит, стекло, стену и т. д. В определенной степени графической информацией можно считать и объективную реальность, на которую направлен объектив фотоаппарата или цифровой камеры.

Компьютерная графика - область информатики, изучающая методы и свойства обработки изображений с помощью программно-аппаратных средств.

Под видами компьютерной графики подразумевается способ хранения изображения на плоскости монитора.

Представление данных на компьютере в графическом виде впервые было реализовано в середине 50-х годов.

Машинная графика в настоящее время уже вполне сформировалась как наука. Существует аппаратное и программное обеспечение для получения разнообразных изображений - от простых чертежей до реалистичных образов естественных объектов. Машинная графика используется почти во всех научных и инженерных дисциплинах для наглядности восприятия и передачи информации. Машинная графика властно вторгается в бизнес, медицину, рекламу, индустрию развлечений. Применение во время деловых совещаний демонстрационных слайдов, подготовленных методами машинной графики и другими средствам автоматизации конторского труда, считается нормой. В медицине становится обычным получение трехмерных изображений внутренних органов по данным компьютерных томографов. В наши дни телевидение и другие рекламные предприятия часто прибегают к услугам машинной графики и компьютерной мультипликации. Использование машинной графики в индустрии развлечений охватывает такие несхожие области как видеоигры и полнометражные художественные фильмы.

**В зависимости от способа формирования изображений компьютерную графику подразделяют:**

⦁ Растровая графика.

⦁ Векторная графика.

⦁ Трехмерная графика.

⦁ Фрактальная графика.

⦁ Символьная графика (устарела и на сегодняшний день практически не используется)

**Задачи обработки графической информации:**

⦁ Ввод и отображение графической информации;

⦁ Редактирование графической информации;

⦁ Сохранение графической информации;

⦁ Считывание (загрузка из файла) информации;

⦁ Вывод на печать.

Для построения, коррекции, сохранения и получения «бумажных» копий рисунков и других изображений используется специальная программа — графический редактор.

**Графический редактор** – это программа создания, редактирования и просмотра графических изображений.

Для создания изображений в графическом редакторе используются определенные инструменты линейка («отрезок»), прямоугольник, круг, эллипс и т. д. Такие инструменты, позволяющие изображать простые фигуры, называются «графическими примитивами». Это как бы простейшие элементы, из которых строится изображение. Чтобы воспользоваться инструментом, необходимо выбрать соответствующий «графический примитив» и установить курсор в ту точку экрана, где необходимо изобразить выбранную фигуру.

Функции всех графических редакторов приблизительно одинаковы (один из простейших графических редакторов для IBM-совместимых компьютеров — Paintbrush). Они позволяют пользователю:

⦁ создавать рисунки из графических примитивов;

⦁ применять для рисования различные цвета и «кисти» (т. е. использовать линии различной ширины и конфигурации);

⦁ «вырезать» рисунки или их части, временно хранить их в буфере («кармане») или запоминать на внешних носителях;

⦁ перемещать фрагмент рисунка по экрану;

⦁ «склеивать» один рисунок с другим;

⦁ увеличивать фрагмент рисунка для того, чтобы прорисовать мелкие детали;

⦁ добавлять к рисункам текст.

Многие графические редакторы позволяют также создавать компьютерную мультипликацию (анимацию), т. е. создавать на компьютере движущиеся изображения.

**«Среда» графического редактора состоит из трех основных частей.**

Инструментальная часть — набор пиктограмм, изображающих инструменты. Обычно, это — «кисть» для изображения линий произвольной конфигурации. Линейка предназначена для проведения отрезков, прямых, окружностей, прямоугольников, эллипсисов для создания соответствующих фигур, «ластик» для стирания изображений, «валик» для закраски фигур, «ножницы» для вырезания фрагментов изображений. Другая часть среды — палитра для выбора цвета изображений. Наконец, третья часть — меню команд редактора. Эти части среды обычно располагаются по краям экрана. Центральная часть экрана предназначена для рабочего поля (или, как говорят, «холста»), на котором создаются изображения.

Графический редактор, как правило, имеет следующие основные режимы работы: режим выбора и настройки инструмента, режим выбора цвета, режим работы с рисунком (рисование и редактирование), режим работы с внешними устройствами.

Работая с графическим редактором, пользователь применяет не только клавиатуру, но и (для большинства современных компьютеров и редакторов) манипулятор мышь. Создавая изображения на экране компьютера, можно не только рисовать их самому, но и использовать другие изображения, например фотографии, рисунки из книг и т. д. Для ввода такой дополнительной графической информации в компьютер используется специальное устройство — сканер.

Для работы с графическими изображениями существует большое количество различных программ и пакетов, называемых графическими редакторами.

Одной из таких программ является входящий в Windows графический редактор Paint. Используя его, можно считывать и сохранять изображение в широко известных растровых форматах. Он позволяет рисовать различные примитивы, закрашивать области цветами и т.п. Другим примером простого графического редактора служит программа Photo Plus. Ее интерфейс подобен редактору Paint, однако программа обладает существенно большими возможностями. В ней, кроме известных возможностей рисования графических примитивов, реализованы функции сканирования изображения, конвертации с различным количеством битов глубины цвета, изменения яркости, контрастности и корректировки цветовой гаммы.

Фирмой Corel разработан графический пакет, в котором интегрированы несколько программ для обработки различных изображений. В состав пакета входят мощный графический редактор CorelDraw!, позволяющий работать с векторными и растровыми изображениями, Corel Photo-Paint, ориентированный на обработку растровых изображений, в том числе высококачественных фотографий.

В настоящее время имеется множество программ для редактирования графических изображений. Эти программы в соответствии с делением графики на растровую и векторную можно условно разделить на два класса:

⦁ Программ для работы с растровой графикой.

⦁ Программ для работы с векторной графикой.

Также имеются программы, которые совмещают возможности программ этих двух классов. То есть позволяют создавать изображения состоящие из растровой и векторной графики. Среди программ первого можно выделить:

⦁ Графический редактор Paint - простой однооконный графический редактор, который позволяет создавать и редактировать достаточно сложные рисунки.

⦁ Photoshop фирмы Adobe многооконный графический редактор позволяет создавать и редактировать сложные рисунки, а также обрабатывать графические изображения (фотографии). Содержит множество фильтров для обработки фотографий (изменение яркости, контрастности и т.д.).

Среди программ второго можно выделить:

⦁ Программа Microsoft Draw - входящая в комплект MS Office. Эта программа служит для создания различных рисунков, схем. Обычно вызывается из MS Word.

⦁ Adobe Illustrator, Corel Draw - программы используются в издательском деле, позволяет создавать сложные векторные изображения. Как правило программы первого класса позволяют сохранять изображения в файлах стандартных форматов: bmp, pcx, gif, tif, а программы второго класса используют для этих целей свои форматы.

Изображения в графических редакторах хранятся по-разному.

Растровый формат характеризуется тем, что все изображение по вертикали и горизонта ли разбивается на достаточно мелкие прямоугольники - так называемые элементы изображения, или пиксели (от английского pixel - picture element).

В файле, содержащем растровую графику, хранится информация о цвете каждого пикселя данного изображения. Чем меньше прямоугольники, на которые разбивается изображение, тем больше разрешение (resolution), то есть, тем более мелкие детали можно закодировать в таком графическом файле.

При векторном формате рисунок представляется в виде комбинации простых геометрических фигур - точек, отрезков прямых и кривых, окружностей, прямоугольников и т.п. При этом для полного описания рисунка необходимо знать вид и базовые координаты каждой фигуры, например, координаты двух концов отрезка, координаты центра и диаметр окружности и т.д. Этот способ кодирования идеально подходит для рисунков, которые легко представить в виде комбинации простейших фигур, например, для технических чертежей.

Основным логическим элементом векторной графики является геометрический объект. В качестве объекта принимаются простые геометрические фигуры (так называемые примитивы - прямоугольник, окружность, эллипс, линия), составные фигуры или фигуры, построенные из примитивов, цветовые заливки, в том числе градиенты.

Преимущество векторной графики заключается в том, что форму, цвет и пространственное положение составляющих ее объектов можно описывать с помощью математических формул.

Важным объектом векторной графики является сплайн. Сплайн - это кривая, посредством которой описывается та или иная геометрическая фигура. На сплайнах построены современные шрифты TryeType и PostScript.

У векторной графики много достоинств. Она экономна в плане дискового пространства, необходимого для хранения изображений: это связано с тем, что сохраняется не само изображение, а только некоторые основные данные, используя которые, программа всякий раз воссоздает изображение заново. Кроме того, описание цветовых характеристик почти не увеличивает размер файла.

Объекты векторной графики легко трансформируются и модифицируются, что не оказывает практически никакого влияния на качество изображения. Масштабирование, поворот, искривление могут быть сведены к паре-тройке элементарных преобразований над векторами.

Объекты векторной графики легко трансформируются и модифицируются, что не оказывает практически никакого влияния на качество изображения. Масштабирование, поворот, искривление, могут быть сведены к паре-тройке элементарных преобразований над векторами.

В тех областях графики, где важное значение имеет сохранение ясных и четких контуров, например, в шрифтовых композициях, в создании логотипов и прочее, векторные программы незаменимы.

Векторная графика может включать в себя и фрагменты растровой графики: фрагмент становится таким же объектом, как и все остальные (правда, со значительными ограничениями в обработке).

Важным преимуществом программ векторной графики является развитые средства интеграции изображений и текста, единый подход к ним. Поэтому программы векторной графики незаменимы в области дизайна, технического рисования, для чертежно-графических и оформительских работ.

Однако, с другой стороны, векторная графика может показаться чрезмерно жесткой, "фанерной". Она действительно ограничена в чисто живописных средствах: в программах векторной графики практически невозможно создавать фотореалистические изображения. Кроме того, векторный принцип описания изображения не позволяет автоматизировать ввод графической информации, как это делает сканер для точечной графики. В последнее время все большее распространение получают программы 3-мерного моделирования, также имеющие векторную природу.

Обладая изощренными методами отрисовки (метод трассировки лучей, метод излучательности), эти программы позволяют создавать фотореалистичные растровые изображения с произвольным разрешением из векторных объектов при умеренных затратах сил и времени.

В любом случае, если вы работаете с графикой, то неизбежно будете иметь дело с обеими ее формами - векторной и растровой. Понимание их сильных и слабых сторон позволит вам выполнить свою работу максимально эффективно

**Система программирования** - совокупность языка программирования и системы разработки программ, обеспечивающей автоматизированную разработку и выполнение программ на данном языке, и подготовку соответствующей документации. Как правило, система программирования включает не эталонный вариант языка, а его некоторую версию, содержащую определенные упрощения или расширения. Некоторые системы программирования могут поддерживать разработку программ на нескольких языках. Наиболее известные системы программирования для персональных компьютеров: Visual Studio, разработанная фирмой Microsoft, поддерживающая языки программирования Basic, Java, C++; Delphi фирмы Inprise (Borland International Inc.), и др.

Компьютеризацию сегодня принято считать панацеей - только компьютер может повысить эффективность образования и промышленности, банковского дела и торговли, объединить через Интернет весь мир! И, очевидно, эта "пандемия" неостановима... Как всякий новый этап в развитии общества, компьютеризация несет с собой и новые проблемы. И одна из наиболее важных - экологическая. Много слов в печати и в других СМИ сказано о вредном влиянии компьютера на здоровье пользователей. Некоторые бойкие авторы даже грозят вымиранием человечеству, сидящему за дисплеями. Поэтому, необходимо объективно оценивать эти проблемы, так как для борьбы с любой опасностью, прежде всего, надо знать, что она собой представляет. Именно зная правила работы за ПК и соблюдая их, можно избежать всех негативных влияний на здоровье пользователей или свести их к минимуму.

Структурное программирование можно толковать как проектирование, написание и тестирование программы в соответствии с заранее определенной дисциплиной.

Основной целью структурного программирования является уменьшение трудностей тестирования и доказательства правильности программы. Это особенно важно при разработке больших программных систем. Опыт применения методов структурного программирования при разработке ряда сложных операционных систем показывает, что правильность логической структуры системы поддается доказательству, а сама программа допускает достаточно полное тестирование. В результате в готовой программе встречаются только тривиальные ошибки кодирования, которые легко исправляются.

Структурное программирование улучшает ясность и читабельность программ. Программы, которые написаны с использованием традиционных методов, особенно те, которые перегружены операторами GOTO, имеют хаотичную структуру. Структурированные программы имеют последовательную организацию, поэтому возможно читать такую программу сверху донизу без перерыва.

Наконец, структурное программирование призвано улучшить эффективность программ.

Всякая программа, в соответствии со структурным подходом к программированию, может быть построена только с использованием трех основных типов блоков:

**1. Функциональный блок, который на блок-схеме изображается в виде прямоугольников с одним входом и одним выходом.**

Функциональному блоку в языках программирования соответствуют операторы ввода и вывода или любой оператор присваивания. В виде функционального блока может быть изображена любая последовательность операторов, выполняющихся один за другим, имеющая один вход и один выход.

**2. Условная конструкция**

Этот блок включает проверку некоторого логического условия (P), в зависимости от которого выполняется либо один (S1), либо другой (S2) операторы.

**3. Блок обобщенного цикла**

Этот блок обеспечивает многократное повторение выполнения оператора S пока выполнено логическое условие P: цикл с параметром и цикл с предусловием.

При конструировании программы с использованием рассмотренных типов блоков эти блоки образуют линейную цепочку так, что выход одного блока подсоединяется к входу следующего. Таким образом, программа имеет линейную структуру, причем порядок следования блоков соответствует порядку, в котором они выполняются.

# 1.2. Постановка задачи

Требуется разработать приложение, состоящее из главной формы. Приложение должно демонстрировать пользователю работу базовых функций графического редактора.

**Функциональные требования:**

⦁ программа должна демонстрировать работу некоторых компонентов редактора;

⦁ наглядно раскрывать все компоненты изучаемой темы;

⦁ должна обладать наглядным интерфейсом, направленным на быстрый поиск необходимой информации.

**Требования к графическому интерфейсу пользователя:**

⦁ пользовательское меню;

⦁ элементы управления;

⦁ хорошо оформленный дизайн.

**Физические требования:**

⦁ наличие операционной системы Windows.

**Информация будет представлена в виде:**

⦁ графических изображений.

**Требования к интерфейсу программы:**

⦁ стандартный интерфейс приложения Windows.

**Требования к уровню диалога:**

⦁ доступный и понятный интерфейс, обеспечивающий лёгкое использование данной программы.

# 2. Технология разработки программного продукта (технологическая часть)

# 2.1. Анализ входных и выходных данных

**Входные данные** – это данные, которые пользователь добавляет в графическом редакторе.

Выбор цвета кисти для прорисовки линий, а так же цвета заливки для окрашивания фона

Управление ползунком для выбора цвета кисти или заливки в шестнадцатеричном формате.

Выбор толщины линий для более точного вывода изображения на экран,

Панель цветов для тех, кто не знаком с шестнадцатеричным форматом ввода цветов.

Построение изображения связано с редактированием и созданием рисунка.

**Выходные данные** – данные, которые выводятся пользователю автоматически и по его непосредственному запросу. К автоматическим данным относятся внешний вид, отображение добавленных ранее рисунков.

**Панель окон «Сохранить» «Открыть»**

Для сохранения информации в файл пользователю необходимо указать тип сохраняемого файла из предлагаемого списка, ввести его имя и нажать кнопку «сохранить». Процедура - как сохранения, так и открытия файла производится с использованием стандартных окон открытия/сохранения файлов характерных для большинства программных продуктов серии MS Office и другие. Подробно процедура сохранения различных типов файлов описана ниже в приложении пользователя.

# 2.2. Инфологическая модель

Реализованный графический редактор позволяет осуществлять работу с как уже имеющимися изображениями, так и создавать собственные изображения, с помощью различных возможностей таких как:

⦁ Возможно изменение цвета фигур, толщины линии их образа;

⦁ создание рисунков с помощью линий (карандаша), кисти различной толщины;

⦁ стирание любой нарисованной картинки;

⦁ сохранение картинки в формате \*.png и jpg.

Объектно-ориентированный анализ и проектирование нельзя определить ни как проектирование сверху вниз, ни как проектирование снизу вверх; его можно скорее назвать «возвратным проектированием», что подразумевает ступенчатый процесс разработки системы с постепенной модификацией различных, но, тем не менее, согласованных между собой логических и физических представлений о системе в целом.

# 2.3 Выбор и обоснование программных средств

При выборе среды реализации сравнивают программные продукты и пользуются различными средствами разработки приложений. Использование возможностей средств разработки приложений позволяет автоматизировать процесс разработки. Инструментальные средства позволяют:

⦁ создавать интерфейс, используя стандартные компоненты;

⦁ передавать управление процессам, в зависимости от состояния системы;

⦁ создавать оболочки для баз данных, как и сами базы данных;

⦁ разрабатывать более надежные программы путем обработки исключительных ситуаций возникающих при некорректной работе программы.

**Современные средства разработки характеризуются параметрами:**

⦁ поддержка объектно-ориентированного стиля программирования;

⦁ возможность использования CASE-технологий, как для проектирования разрабатываемой системы, так и для разработки моделей реляционных баз данных;

⦁ использование визуальных компонент для наглядного проектирования интерфейса;

⦁ поддержка БД.

Выше перечисленными свойствами обладают языки программирования:

⦁ SQL;

⦁ Pyton;

⦁ Visual C++.

Каждое из этих средств содержит весь спектр современного инструментария, который был перечислен ранее. Главное отличие состоит в области использования рассматриваемых средств.

Для реализации дипломного проекта была выбрана среда разработки Pyton. При решении поставленной задачи оптимально использовать для представления информационных материалов язык Pyton, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

Для реализации данной ИС была выбрана система программирования Pyton, так как она предоставляет наиболее широкие возможности для программирования приложений ОС Windows.

**Python** — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

**Преимущества Pyton по сравнению с аналогичными программными продуктами:**

* простоте синтаксиса и визуального восприятия кода;
* легкости изучения этого языка для новичков;
* изобилие различных библиотек и фреймворков;
* кроссплатформенность;
* поддержке ООП;
* динамической типизации.

Система программирования Pyton рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого.

К тому же, работодателей интересует, прежде всего, скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Pyton полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

# 2.4 Выбор и обоснование аппаратных средств разработки

Для разработки данного графического редактора использовалось аппаратное средство с приведенными ниже в таблицах 1 и 2 характеристиками.

Таблица 1. Свойства компьютера.

|  |  |
| --- | --- |
| Операционная система | Windows 11 Home Edition 64-bit |
| Язык интерфейса | русский |
| Процессор | Intel® Core i3 11400f |
| Оперативная память | 8 GB |

Данные характеристики были выбраны, так как они соответствовали характеристикам компьютера, в котором она была написана.

# 3. Описание программных модулей

# 3.1. Структура и алгоритм работы программного продукта

Созданная система в процессе деятельности может быть представлена рядом состояний, которые осуществляют те или иные действия. В системе можно выделить некоторое начальное состояние и конечное, завершающее работу состояние.

### Функциональность редактора:

- Масштабирование в %

- Черно-белый режим

- Розовый и голубой фильтры

- Обрезка до нужного размера (crop)

- Отражение по горизонтали

- Поворот вокруг оси

Файлы png после редактирования автоматически сохраняются как png, а jpg-файлы - соответственно, как jpg. При необходимости выполняется конвертация в RGB.

# 3.2 Инструментарий разработки программного продукта

Для создания программы в качестве средства разработки была выбрана среда программирования Pyton, так как Pyton - это один из самых популярных языков программирования.

Pyton– это комбинация нескольких важнейших технологий:

⦁ высокопроизводительный компилятор в машинный код;

⦁ объектно-ориентированная модель компонент;

⦁ визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;

⦁ средства для построения баз данных.

Компилятор, встроенный в Pyton, обеспечивает высокую производительность. Этот компилятор является одним из самых быстрых в мире. Он предлагает легкость разработки и быстрое время проверки готового программного блока и в то же время обеспечивает качество кода.

Кроме того, в Pyton компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это сказывается на фактическом быстродействии готового приложения.

Для разработки Главной Формы программного продукта были использованы следующие компоненты.

Форма (Form) – это важнейший визуальный компонент. Форма представляет собой видимое окно Windows и является основной частью практически любого приложения.

# 3.3. Интерфейс программного продукта

С помощью данного проекта можно создавать простейшие графические изображения (левой кнопкой мыши), закрашивать замкнутые области (правой кнопкой мыши), изменять цвет и толщину линии, сохранять рисунок в файле.

# Image Editor (Python, Pillow, Tkinter)

## Графический редактор на Python, Pillow и Tkinter

### Функциональность редактора:

- Масштабирование в %

- Черно-белый режим

- Розовый и голубой фильтры

- Обрезка до нужного размера (crop)

- Отражение по горизонтали

- Поворот вокруг оси

Файлы .png после редактирования автоматически сохраняются как .png, а jpg-файлы - соответственно, как jpg. При необходимости выполняется конвертация в RGB.

# 4. Экспериментальная часть

# 4.1 Назначение испытаний

Испытания производится, для того чтобы выявить какие либо недочеты в самом графическом редакторе и в последствии их исправить.

Будет проводиться на совместимость работы с другими ОС, а так же при их изменений системных требований.

Испытание программы проходило при её отладке, при испытании проверялись функциональные возможности подсистем и отдельных инструментов. Затем, осуществлялся сбор всех инструментов, производилось системное тестирование. С целью тестирования в данном редакторе создавались графические изображения при помощи многократного использования различных инструментов. Проверялась работа редактора при изменении цвета, толщины и вида линий, а также при изменении стиля заполнения различных фигур.

В редакторе можно произвести коррекцию уже созданных файлов, содержащих растровые изображения.

# Заключение

Целью моей курсовой работы была разработка программы «Графический редактор», которая позволяет пользователю просматривать и редактировать графические изображения.

В результате создана программа, которая позволяет осуществлять создание и редактирование графических изображений.

В процессе разработки можно выделить несколько основных этапов:

-исследование,

-проектирование,

-разработка,

-опытная эксплуатация,

-внедрение.

**На этапе исследования рассмотрены:**

-наиболее известные графические пакеты для создания иллюстраций;

-основные графические форматы файлов для хранения графической информации.

**А также проведен сравнительный анализ:**

-возможностей графических пакетов для создания иллюстраций;

Интерфейс созданной программы удобен, прост, наглядно отображает ее возможности. Главное меню редактора содержит команды работы с файлами, команды занесения рисунка в буфер и вывода рисунка из буфера.

Тестирование подтвердило, что программа корректно выполняет обработку данных и демонстрацию результатов. Всё это свидетельствует о работоспособности программы и позволяет сделать вывод о ее пригодности для создания и редактирования графических изображений.

# Список использованных источников

⦁ Васюков В.Н. Цифровая обработка сигналов и сигнальные процессоры в системах подвижной радиосвязи: Учебник для вузов Учебники НГТУ. 2003 - 235 с.

⦁ Вендров А.М. Проектирование программного обеспечения для экономических систем. М.: Финансы и статистика, 1999

⦁ Головин О. В., Чистяков Н. И., Шварц В., Хардон Агиляр И. Радиосвязь. Под ред. проф. О. В. Головина. М., Горячая линия - Телеком, 2001

⦁ Культин Н.Б. Программирование в Turbo Pascal и Delphi.- СПб.: БХВ - Санкт-Петербург, 2000

⦁ Симонович С., Евсеев Г., Алексеев А. Специальная информатика. Учебное пособие. - М.: АСТ - ПРЕСС: Инфорком - Пресс, 1998

⦁ Семакин И.Г., Шестаков А.П. Основы программирования: Учебник.- М.: Мастерство, 2002

⦁ Рудаков. Технология разработки программных продуктов. Учебное пособие, 2007

⦁ Процессы жизненного цикла программных средств. Информационная технология. ГОСТ Р ИСО/МЭК 12207-99, М.: Госстандарт России, 2000

⦁ Схемы алгоритмов и программ. Правила выполнения. ГОСТ 19.002-80. М.: Госстандарт России , 1980

⦁ Сибаров К.Г., Сколотнев Н.Н., Васин В.К., Начинаев В.Н. Охрана труда в вычислительных центрах: учебное пособие, М.: Машиностроение, 1985

⦁ ГОСТ 12.1.030-81. ССБТ. Электробезопасность. Защитное заземление, зануление

⦁ 12.1.003-83. ССБТ. Шум Общие требования безопасности

⦁ ГОСТ 12.1.005-88. ССБТ. Воздух рабочей зоны. Общие санитарно - гигиенические требования. М.: Изд-во стандартов, 1990

⦁ Фаронов В.В. Delphi 3. Учебный курс. М.: "Нолидж", 1998. 400 с.

⦁ Эндрю Возневич. Delphi. Освой самостоятельно: Пер. с англ. М.: Восточная Книжная Компания, 1996. 736 с.

⦁ <http://www.bibliofond.ru/view.aspx?id=540556> Программный продукт "Графический редактор учебно-методических материалов"

⦁ <http://nsportal.ru/shkola/informatika-i-ikt/library/graficheskaya-informatsiya> Графическая информация

⦁ Википедия свободная энциклопедия // <http://ru.wikipedia.org/wiki>

# Листинг

# Программный код графического редактора.

from tkinter import \*

from PIL import Image, ImageTk

class ImageEditor:

def \_\_init\_\_(self, window):

self.window = window

self.window.title("Графический редактор")

self.window.config(bg="#FFFFFF")

self.canvas = Canvas(self.window, width=400, height=400, bg="#F5F5F5")

self.canvas.pack(fill=BOTH, expand=True, padx=20, pady=20)

self.image\_file = None

self.image = None

self.image\_tk = None

self.undo\_stack = []

self.redo\_stack = []

# кнопки

open\_icon = ImageTk.PhotoImage(Image.open("buttons/open.png"))

self.open\_button = Button(self.window, image=open\_icon, command=self.open\_image, bd=0)

self.open\_button.image = open\_icon

self.open\_button.pack(side=LEFT, padx=10, pady=5)

undo\_icon = ImageTk.PhotoImage(Image.open("buttons/undo.png"))

self.undo\_button = Button(self.window, image=undo\_icon, command=self.undo, bd=0)

self.undo\_button.image = undo\_icon

self.undo\_button.pack(side=LEFT, padx=10, pady=5)

redo\_icon = ImageTk.PhotoImage(Image.open("buttons/redo.png"))

self.redo\_button = Button(self.window, image=redo\_icon, command=self.redo, bd=0)

self.redo\_button.image = redo\_icon

self.redo\_button.pack(side=LEFT, padx=10, pady=5)

save\_icon = ImageTk.PhotoImage(Image.open("buttons/save.png"))

self.save\_button = Button(self.window, image=save\_icon, command=self.save\_image, bd=0)

self.save\_button.image = save\_icon

self.save\_button.pack(side=LEFT, padx=10, pady=5)

resize\_icon = ImageTk.PhotoImage(Image.open("buttons/resize.png"))

self.resize\_button = Button(self.window, image=resize\_icon, command=self.resize\_image, bd=0)

self.resize\_button.image = resize\_icon

self.resize\_button.pack(side=LEFT, padx=10, pady=5)

rotate\_icon = ImageTk.PhotoImage(Image.open("buttons/rotate.png"))

self.rotate\_button = Button(self.window, image=rotate\_icon, command=self.rotate\_image, bd=0)

self.rotate\_button.image = rotate\_icon

self.rotate\_button.pack(side=LEFT, padx=10, pady=5)

flip\_icon = ImageTk.PhotoImage(Image.open("buttons/flip.png"))

self.flip\_button = Button(self.window, image=flip\_icon, command=self.flip\_image, bd=0)

self.flip\_button.image = flip\_icon

self.flip\_button.pack(side=LEFT, padx=10, pady=5)

black\_white\_icon = ImageTk.PhotoImage(Image.open("buttons/black\_white.png"))

self.black\_and\_white\_button = Button(self.window, image=black\_white\_icon, command=self.black\_and\_white\_image, bd=0)

self.black\_and\_white\_button.image = black\_white\_icon

self.black\_and\_white\_button.pack(side=LEFT, padx=10, pady=5)

crop\_icon = ImageTk.PhotoImage(Image.open("buttons/crop.png"))

self.crop\_button = Button(self.window, image=crop\_icon, command=self.crop\_image, bd=0)

self.crop\_button.image = crop\_icon

self.crop\_button.pack(side=LEFT, padx=10, pady=5)

add\_text\_icon = ImageTk.PhotoImage(Image.open("buttons/add\_text.png"))

self.add\_text\_button = Button(self.window, image=add\_text\_icon, command=self.add\_text, bd=0)

self.add\_text\_button.image = add\_text\_icon

self.add\_text\_button.pack(side=LEFT, padx=10, pady=5)

pink\_filter\_icon = ImageTk.PhotoImage(Image.open("buttons/pink\_filter.png"))

self.pink\_filter\_button = Button(self.window, image=pink\_filter\_icon, command=self.apply\_pink\_filter, bd=0)

self.pink\_filter\_button.image = pink\_filter\_icon

self.pink\_filter\_button.pack(side=LEFT, padx=10, pady=5)

blue\_filter\_icon = ImageTk.PhotoImage(Image.open("buttons/blue\_filter.png"))

self.blue\_filter\_button = Button(self.window, image=blue\_filter\_icon, command=self.apply\_blue\_filter, bd=0)

self.blue\_filter\_button.image = blue\_filter\_icon

self.blue\_filter\_button.pack(side=LEFT, padx=10, pady=5)

def open\_image(self):

# открытие изображения для редактирования

filename = filedialog.askopenfilename(title="Выберите файл", filetypes=(("jpeg файлы", "\*.jpg"), ("png файлы", "\*.png")))

if filename:

self.undo\_stack.append(self.image.copy() if self.image else None)

self.redo\_stack.clear()

self.image\_file = filename

self.image = Image.open(filename)

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def save\_image(self):

# конвертация и запись в файл

if self.image\_file and self.image:

defaultextension = ""

filetype = ""

selectedtype = self.image\_file.split('.')[-1]

if selectedtype.lower() == "jpg" or selectedtype.lower() == "jpeg":

defaultextension = ".jpg"

filetype = (("JPEG файлы", "\*.jpg"),)

elif selectedtype.lower() == "png":

defaultextension = ".png"

filetype = (("PNG файлы", "\*.png"),)

else:

defaultextension = ".jpg"

filetype = (("JPEG файлы", "\*.jpg"), ("PNG файлы", "\*.png"), ("Все файлы", "\*.\*"))

filename = filedialog.asksaveasfilename(initialdir="/", title="Сохранить файл как",defaultextension=defaultextension,

filetypes=filetype)

if filename:

if self.image.mode == 'RGBA':

self.image = self.image.convert('RGB')

elif self.image.mode == 'P':

self.image = self.image.convert('RGB')

if filename.endswith('.jpg') or filename.endswith('.jpeg'):

self.image.save(filename, 'JPEG', quality=90)

elif filename.endswith('.png'):

self.image.save(filename, 'PNG')

else:

messagebox.showerror("Ошибка!", "Неподдерживаемый формат")

def perform\_resize(self, width\_percent, height\_percent):

# масштабирование изображения

if self.image:

self.undo\_stack.append(self.image.copy())

width = int(self.image.width \* width\_percent / 100)

height = int(self.image.height \* height\_percent / 100)

self.image = self.image.resize((width, height))

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def rotate\_image(self):

# поворот изображения

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

self.image = self.image.rotate(90)

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def flip\_image(self):

# горизонтальное зеркальное отражение

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

self.image = self.image.transpose(Image.FLIP\_LEFT\_RIGHT)

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def black\_and\_white\_image(self):

# черно-белое изображение

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

self.image = self.image.convert('L')

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def crop\_image(self):

# обрезка до нужного размера

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

self.canvas.bind("<Button-1>", self.start\_crop)

def start\_crop(self, event):

# начало операции

self.crop\_coords = [event.x, event.y]

self.canvas.bind("<B1-Motion>", self.draw\_crop)

self.canvas.bind("<ButtonRelease-1>", self.end\_crop)

def draw\_crop(self, event):

# рисование красной рамки-границы

if hasattr(self, "crop\_rectangle"):

self.canvas.delete(self.crop\_rectangle)

x0, y0 = self.crop\_coords

x1, y1 = event.x, event.y

self.crop\_rectangle = self.canvas.create\_rectangle(x0, y0, x1, y1, outline="red")

def end\_crop(self, event):

# определение окончательного размера

x0, y0 = self.crop\_coords

x1, y1 = event.x, event.y

crop\_region = (x0, y0, x1, y1)

cropped\_image = self.image.crop(crop\_region)

self.image = cropped\_image

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

self.canvas.delete(self.crop\_rectangle)

self.canvas.unbind("<B1-Motion>")

self.canvas.unbind("<ButtonRelease-1>")

def add\_text(self):

# добавление надписей

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

self.canvas.bind("<Button-1>", self.add\_text\_callback)

def add\_text\_callback(self, event):

# определение позиции для вставки текста

text\_window = Toplevel()

text\_window.geometry("300x250")

text\_label = Label(text\_window, text="Введите текст надписи")

text\_label.pack()

text\_entry = Entry(text\_window)

text\_entry.pack()

# выбор цвета текста и размера шрифта

font\_color\_label = Label(text\_window, text="Цвет текста")

font\_color\_label.pack()

self.font\_color = StringVar()

self.font\_color.set("white")

font\_color\_menu = OptionMenu(text\_window, self.font\_color, "white", "black", "red", "green", "blue")

font\_color\_menu.pack()

font\_size\_label = Label(text\_window, text="Размер шрифта")

font\_size\_label.pack()

self.font\_size = StringVar()

self.font\_size.set("12")

font\_size\_menu = OptionMenu(text\_window, self.font\_size, "10", "12", "14", "16", "20", "24", "30")

font\_size\_menu.pack()

add\_button = Button(text\_window, text="Добавить надпись",

command=lambda: self.perform\_add\_text(text\_entry.get(), event))

add\_button.pack()

def perform\_add\_text(self, text, event):

# добавление надписи на поверхность изображения

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

x, y = event.x, event.y

font\_color = self.font\_color.get()

font\_size = int(self.font\_size.get())

self.canvas.create\_text(x, y, text=text, fill=font\_color, font=("Arial", font\_size))

def apply\_pink\_filter(self): # добавление слоя с розовой заливкой

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

if self.image.mode != "RGBA":

self.image = self.image.convert("RGBA")

pink\_color = (255, 192, 203, 128)

drawing\_layer = Image.new('RGBA', self.image.size, pink\_color)

self.image = Image.alpha\_composite(self.image, drawing\_layer)

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def apply\_blue\_filter(self): # голубая заливка

if self.image:

self.undo\_stack.append(self.image.copy())

self.redo\_stack.clear()

if self.image.mode != "RGBA":

self.image = self.image.convert("RGBA")

blue\_color = (173, 216, 230, 128)

drawing\_layer = Image.new('RGBA', self.image.size, blue\_color)

self.image = Image.alpha\_composite(self.image, drawing\_layer)

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

def undo(self): # отмена последнего действия

if self.undo\_stack:

self.redo\_stack.append(self.image.copy() if self.image else None)

self.image = self.undo\_stack.pop()

if self.image:

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

else:

self.canvas.delete("all")

def redo(self): # восстановление отмененного действия

if self.redo\_stack:

self.undo\_stack.append(self.image.copy() if self.image else None)

self.image = self.redo\_stack.pop()

if self.image:

self.image\_tk = ImageTk.PhotoImage(self.image)

self.canvas.create\_image(0, 0, anchor=NW, image=self.image\_tk)

else:

self.canvas.delete("all")

def resize\_image(self):

# масштабирование изображения

if self.image:

resize\_window = Toplevel()

resize\_window.geometry("300x250")

width\_label = Label(resize\_window, text="Нужная ширина в %")

width\_label.pack()

width\_entry = Entry(resize\_window)

width\_entry.pack()

height\_label = Label(resize\_window, text="Нужная высота в %")

height\_label.pack()

height\_entry = Entry(resize\_window)

height\_entry.pack()

resize\_button = Button(resize\_window, text="Масштабировать",

command=lambda: self.perform\_resize(int(width\_entry.get()), int(height\_entry.get())))

resize\_button.pack()

root = Tk()

editor = ImageEditor(root)

root.mainloop()

**UML-Диаграммы.**

