Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 03 «информационные системы и технологии»

Специализация «издательско-полиграфический комплекс»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Объектно-ориентированное программирование»

Тема «Графический редактор»

Исполнитель

студент 2 курса группы 2 \_\_\_\_\_\_\_\_\_\_ П. С. Карпович

подпись, дата

Руководитель

доцент, к.т.н. \_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись, дата

Курсовая работа защищена с оценкой

Руководитель Н.В. Пацей

подпись

Минск 2015

Оглавление

[Введение 3](#_Toc422040896)

[1.Информационные разделы и технологии 4](#_Toc422040897)

[1.1.Язык программирования C# и платформа .NET 4](#_Toc422040898)

[1.2.Технология WPF 5](#_Toc422040899)

[1.3.Microsoft Visual Studio 7](#_Toc422040900)

[2.Архитектура. Проектирование и реализация 9](#_Toc422040901)

[2.1.Описание цели и постановка задачи 9](#_Toc422040902)

[2.2.Обзор аналогов 10](#_Toc422040903)

[2.3.Проектирование 12](#_Toc422040904)

[2.4.Структура проекта 13](#_Toc422040905)

[2.4.Пользовательские классы 14](#_Toc422040906)

[3.Руководство пользователя 22](#_Toc422040907)

[4.Тестирование программы 24](#_Toc422040908)

[Заключение 25](#_Toc422040909)

[Список используемой литературы и источников 26](#_Toc422040910)

# Введение

Представление данных на мониторе компьютера в графическом виде впервые было реализовано в середине 50-х годов для больших ЭВМ, применявшихся в научных и военных исследованиях. С тех пор графический способ отображения данных стал важнейшей частью подавляющего числа компьютерных систем, в особенности персональных, и получил имя компьютерной графики.

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 и создается множество программ для создания и редактирования изображений, то есть графические редакторы, о создании которого здесь и пойдет речь.

# 1.Информационные разделы и технологии

При создании приложения были задействован перечень следующих инструментов:

• Microsoft Visual Studio;

• Adobe Photoshop CS6;

Microsoft Visual Studio 2010 использовалась в качестве исполняемой платформы, в которой происходило непосредственное создание приложения. Abode Photoshop служил инструментом для формирования иконок и различного рода изображений, используемых а качестве ресурсов программы.

# 1.1.Язык программирования C# и платформа .NET

C# - строго типизированный и объектно-ориентированный язык, чей код внешне похож на C++ (и Java). Это решение проектировщиков языка C# позволяет разработчикам C-подобных языков легко освоить C#.

Исходный код C# компилируется в так называемый управляемый код (managed code) на промежуточном языке IL (Intermediate Language), который находится посередине межу высокоуровневым языком (C#) и языком более низкого уровня (ассемблером/машинным кодом). Во время выполнения среда CLR (Common Language Runtime – общеязыковая исполняющая среда) на лету компилирует код IL, применяя для этого оперативную компиляцию (Just in Time - JIT). Очевидным недостатком этой техники может показаться неэффективность компиляции кода во время выполнения. Этот процесс отличается от интерпретации. Компилятор JIT не компилирует функцию или метод при каждом вызове; он делает это только первый раз, при этом продуцируя машинный код, родной по отношению к платформе, на которой он выполняется. Очевидное достоинство компиляции JIT – сокращение рабочего множества приложения. Во время выполнения приложения компилируется лишь тот код, который необходим. CLR может также оптимизировать выполнение программы на лету. Таким образом, для большинства приложений достоинств у такого подхода больше.

В каркасе среды разработки .NET Framework можно выделить два основных компонента: статический – FCL (Framework Class Library) или библиотека классов каркаса; динамический – CLR.

Число классов библиотеки FCL велико (несколько тысяч). Поэтому логически классы с близкой функциональностью объединяются в группы, называемые пространством имен (namespace). Основным пространством имен библиотеки FCL является пространство System, содержащее как классы, так и другие вложенные пространства имен.

Для динамического компонента CLR физической единицей, объединяющей классы и другие ресурсы, является сборка (assembly). Сборка представляет собой один или несколько файлов, содержащих все необходимые сведения о развертывании программы и ее версии. Сборки составляют основу среды .NET. Они предоставляют механизмы для надежного взаимодействия компонентов, межязыковой возможности взаимодействия и управления версиями. Кроме того, сборки определяют область действия программного кода.

Сборка состоит из четырех разделов. Первый раздел представляет собой декларацию сборки. Декларация содержит сведения о самой сборке. К этой информации относится имя сборки, номер ее версии, культурной среды (язык и региональные стандарты). Второй раздел сборки содержит метаданные типов, т.е. сведения о типах данных, используемых в программе. Среди прочих преимуществ метаданные типов способствуют межъязыковой возможности взаимодействия. Третий раздел сборки содержит программный код в формате MSIL (Microsoft Intermediate Language – промежуточный язык корпорации Microsoft). Четвертый раздел сборки содержит ресурсы, используемые программой.

Исполняемый файл, создаваемый во время компиляции программы на C#, на самом деле представляет собой сборку, содержащую исполняемый код этой программы, а также другие виды информации. Таким образом, когда компилируется программа на C#, сборка получается автоматически.

# 1.2.Технология WPF

Windows Presentation Foundation (WPF) — это графическая система отображения для Windows. Платформа WPF спроектирована для .NET под влиянием таких современных технологий отображения, как HTML и Flash, и использует аппаратное ускорение. Она также является наиболее радикальным изменением в пользовательском интерфейсе Windows со времен Windows 95.

Лежащая в основе WPF графическая технология — это не GDI/GDI+. Теперь это DirectX. Примечательно, что приложения WPF используют DirectX независимо от создаваемого типа пользовательского интерфейса. Это значит, что создается ли сложная трехмерная графика (DirectX's forte), или просто рисуются кнопки и простой текст — вся работа по рисованию проходит через конвейер DirectX. В результате даже самые заурядные бизнес-приложения могут использовать богатые эффекты вроде прозрачности и сглаживания. Также получается выигрыш от аппаратного ускорения, и это означает, что DirectX передает как можно больше работы узлу обработки графики (graphics processing unit — GPU), который представляет собой отдельный процессор на видеокарте.

Даже если бы единственным достоинством WPF было аппаратное ускорение через DirectX, это уже стало бы значительным усовершенствованием, хоть и не революционным. Однако WPF на самом деле включает целый набор высокоуровневых служб, ориентированных на прикладных программистов.

Ниже приведен список некоторых наиболее существенных изменений, которые привнесла с собой технология WPF в мир программирования Windows-приложений.

• Веб-подобная модель компоновки. Вместо того чтобы фиксировать элементы управления на месте с определенными координатами, WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высоко динамичного содержимого или к разным языкам.

• Богатая модель рисования. Вместо рисования пикселей в WPF вы имеете дело с примитивами — базовыми фигурами, блоками текста и прочими графическими ингредиентами. Кроме того, доступны такие новые средства, как действительно прозрачные элементы управления, возможность укладывания друг на друга множества уровней с разной степенью прозрачности, а также встроенная поддержка трехмерной графики.

• Развитая текстовая модель. После многих лет нестандартной обработки текстов WPF наконец-то предоставляет Windows-приложениям возможность отображения расширенного стилизованного текста в любом месте пользовательского интерфейса. И если нужно отображать значительные объемы текста, для повышения читабельности можно воспользоваться развитыми средствами отображения документов, такими как переносы, разбиение на колонки и выравнивание.

• Анимация как первоклассная программная концепция. В WPF нет необходимости использовать таймер для того, чтобы заставить форму перерисовать себя. Вместо этого доступна анимация — неотъемлемая часть платформы. Анимация определяется декларативными дескрипторами, и WPF запускает ее в действие автоматически.

• Поддержка аудио и видео. Прежние инструментальные наборы для построения пользовательских интерфейсов, такие как Windows Forms, были весьма ограничены в работе с мультимедиа. Однако WPF включает поддержку воспроизведения любого аудио- или видеофайла, поддерживаемого проигрывателем Windows Media, позволяя воспроизводить более одного медиафайла одновременно. Что еще больше впечатляет — WPF предоставляет в ваше распоряжение инструменты для интеграции видеосодержимого в остальную часть пользовательского интерфейса, позволяя выполнять такие экзотические трюки, как размещение видеоокна на поверхности вращающегося трехмерного куба.

• Стили и шаблоны. Стили позволяют стандартизировать форматирование и многократно использовать его по всему приложению. Шаблоны дают возможность изменить способ отображения элементов, даже таких основополагающих, как кнопки. Построение интерфейса с обложками еще никогда не было таким простым.

• Команды. Большинству пользователей известно, что не имеет значения, откуда они инициируют команду открытия (Open) — через меню или панель инструментов; конечный результат один и тот же. Теперь эта абстракция доступна коду — можно определять команды приложения в одном месте и привязывать их к множеству элементов управления.

• Декларативный пользовательский интерфейс. Хотя можно конструировать окно WPF в коде, в Visual Studio используется другой подход. Содержимое каждого окна сериализуется в виде XML-дескрипторов в документе XAML. Преимущество состоит в том, что пользовательский интерфейс полностью отделяется от кода, и дизайнеры графики могут использовать профессиональные инструменты для редактирования файлы XAML, улучшая внешний вид всего приложения. (XAML — это сокращение от Extensible Application Markup Language.

• Приложения на основе страниц. Используя WPF, можно строить браузер-подобные приложения, которые позволяют перемещаться по коллекции страниц, оснащенной кнопками навигации "вперед" и "назад". WPF автоматически обрабатывает все сложные детали, такие как хронология посещения страниц. Проект можно даже развернуть в виде браузерного приложения, которое выполняется внутри Internet Explorer.

Традиционные Windows-приложения связаны определенными предположениями относительно разрешения экрана. Обычно разработчики рассчитывают на стандартное разрешение монитора (вроде 1024x768 пикселей) и проектируют свои окна с учетом этого, стараясь обеспечить разумное поведение при изменении размеров в большую и меньшую сторону.

Проблема в том, что пользовательский интерфейс в традиционных Windows-приложениях не является масштабируемым. В результате, если вы используете монитор с высоким разрешением, который располагает пиксели более плотно, окно приложения становится меньше и читать текст в нем труднее. Эта проблема особенно актуальна для новых мониторов, которые имеют высокую плотность пикселей и соответственно работают с более высоким разрешением.

WPF же не страдает от этой проблемы, потому что самостоятельно визуализирует все элементы пользовательского интерфейса — от простых фигур до таких распространенных элементов управления, как кнопки. В результате если вы создаете кнопку шириной в 1 дюйм на обычном мониторе, она останется шириной в 1 дюйм и на мониторе с высоким разрешением. WPF просто визуализирует ее более детализировано, с большим количеством пикселей.

# 1.3.Microsoft Visual Studio

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WPF, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight. 10 Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно- ориентированных языках программирования или инструментов для прочих аспектов цикла разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server). Новая возможность IntelliTrace позволяет оставить проблему невозможности воспроизведения ошибок в прошлом. Тестеры могут подробно и эффективно описывать ошибки и связанные системные сведения и даже включать в отчет моментальные снимки среды. Таким образом разработчики всегда смогут воспроизвести их в том состоянии, в котором их обнаружили. Team Foundation Server (TFS) — это платформа совместной работы на основе решения Майкрософт по управлению жизненным циклом приложений. TFS автоматизирует и упрощает разработку программного обеспечения, а также обеспечивает возможность отслеживать все элементы проекта и видеть в реальном времени его состояние для всех участников проекта. Платформа также предоставляет мощные средства панели мониторинга и создания отчетов.

# 2.Архитектура. Проектирование и реализация

# 2.1.Описание цели и постановка задачи

В данном курсовом проекте необходимо разработать примитивный растровый графический редактор.

Поскольку приложение касается области творческой деятельности, требуется наивысшее внимание уделить дизайну пользовательского интерфейса и его юзабилити. Приложение должно поддерживать функционал работы с внешними графическими изображениями, т.е. необходимо предусмотреть возможности создания, открытия и сохранения редактируемых изображений на жесткий диск. В программе должна быть реализована возможность рисования, как стандартными средствами, так и с помощью предопределенного набора графических примитивов. Также, поскольку редактор разрабатывается на платформе WPF, описанной выше, в должна поддерживаться грамотная система компоновки элементов управления, обеспечивающая хорошую масштабируемость окна приложения.

Функции, которые должно выполнять программное средство:

• Работа с внешними графическими файлами.

• Возможность сохранения рисунка в различных форматах.

• Поддержка рисования графических примитивов.

• Использование механики слоев изображения.

# 2.2.Обзор аналогов

Существует огромное количество всевозможных программных продуктов для работы с графическими элементами.

Среди графических редакторов, есть мощные профессиональные программы (Adobe Photoshop, Adobe Illustrator, 3D Studio, CorelDraw) с массой вспомогательных плагинов и всяческих дополнительных возможностей. Есть более простые и более компактные - Paintshop Pro, Photofinish или Paint из состава Windows. Графические редакторы включают в себя два типа: растровые (Adobe Photoshop,Paintshop,Paint), рисующие изображение по точкам, для каждой из которых отдельно заданы её цвет и яркость; векторные, рисующие сразу целую линию - дугу, отрезок прямой, а сложные линии представляют как совокупность таких дуг и отрезков. Векторные графические редакторы (CorelDraw,Adobe Illustrator) позволяют проделывать очень сложные трансформации формы рисунка, сжатия и растяжения, любые изменения размера, преобразования контуров.

Растровые программы используют, когда надо обрабатывать сканированные изображения-картины, рисунки, фотографии. Основной упор делается на ретуширование изображений, коррекцию цветов, подбор цветов, подбор оптимального контраста, яркости, чёткости, на разного рода размывки и затуманивания, игры со светотенью, составление коллажей. Но с формой объектов они работают плохо.

Рассмотрим нескольких распространенных редакторов:

• Для начала, один из самых примитивных программ обработки графики: стандартное приложение операционной системы Windows– MSPaint.

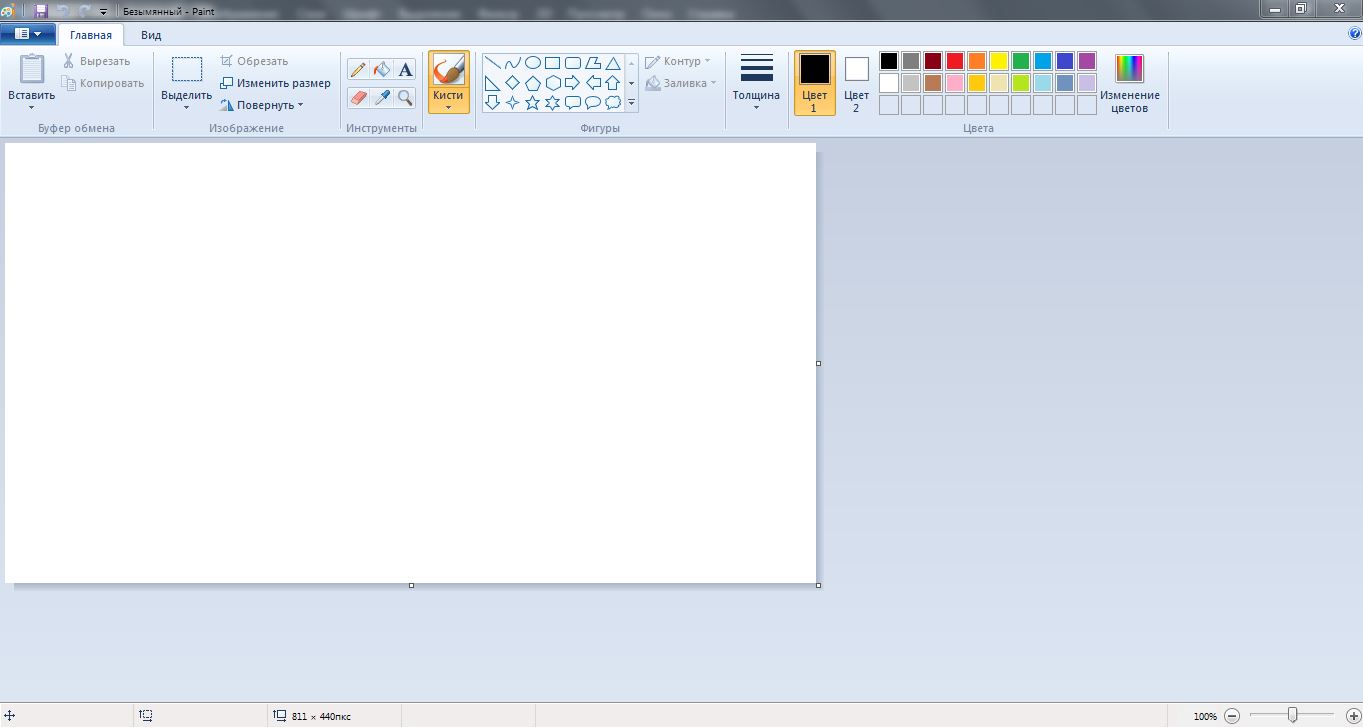


Рисунок 1. MS Paint

Данный графический редактор обладает всеми основными функциями, необходимыми для того, чтобы графический редактор был графическим редактором, а именно рисование простыми линиями, рисование примитивов, настройка пера и выбор цвета.

При создании курсового проекта, в плане реализуемых функций, в качестве ориентира выступал именно этот редактор.

• Paint.NET.

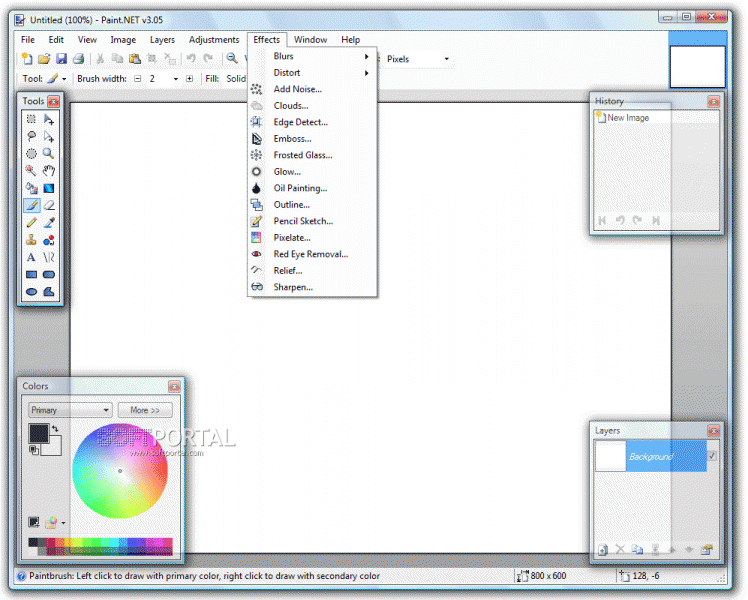


Рисунок 2. Paint.NET

Данный продукт интересен прежде всего тем, что написан на той же платформе .NET Framework с использованием технологии WPF, на которой написано и создаваемое программное средство.

Paint.NET использует все возможности WPF, включая аппаратное ускорение, что определенно сказывается на производительности, которая зачастую бывает крайне необходима, при обработке сложных или объемных изображений.

Количество доступных функций здесь гораздо больше, чем в предыдущем аналоге и данный редактор может предоставить профессиональным пользователям необходимый базовый инструментарий по всем важнейшим категориям.

# 2.3.Проектирование

Приложение должно содержать одно главное окно с рабочей областью и панелями инструментов, которое должно выступать основой для всех дополнительных окон. Верхняя часть окна отведена для нескольких ярусов инструментальных панелей, а также пунктов меню. Все это требуется “упаковать” в различные контейнеры для достижения наилучшего эффекта компоновки интерфейса.

В левой части экрана будет располагаться главная панель инструментов, содержащая наиболее важные и используемые средства рисования, такие как ластик или кисть.

При запуске приложения, во время загрузки его в оперативную память, должно появляться вступительное окно и информацией о продукте.

# 2.4.Структура проекта

Обобщенная структура проекта на основании классов представлена на следующем рисунке:

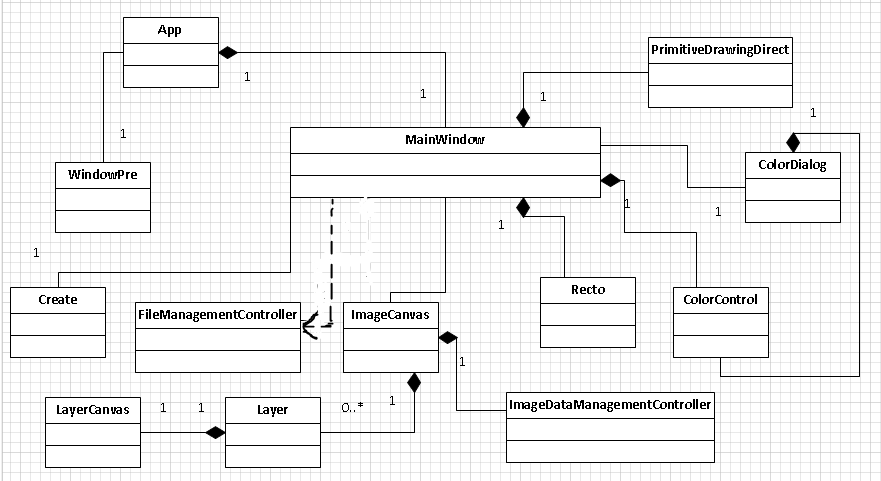


Рисунок 3.Структура классов приложения

В следующем разделе будут подробнее описана структура самих классов.

Как можно увидеть из рисунка, основная масса классов взаимодействует с главным окном (MainWindow). Главное окно связывается с двумя дочерними окнами – создания изображения (Create) и диалога настройки цвета (ColorDialog). Также главное окно включает в себя некоторые пользовательские элементы управления, а также обращается к статическим методам работы с файлами изображений класса FileManagementController.

Непосредственно само изображение представляется в приложении в виде класса ImageCanvas, включающего в себя переменное количество слоев (Layer-LayerCanvas) и класс управления стеком возврата действий – ImageDataManagementController.

В редакторе имеется возможность работы со слоями изображения. Что реализовано в специальном диалоговом окне LayerControl.

Методы рисования располагаются в классе PrimitiveDrawingDirect.

# 2.4.Пользовательские классы

Рассмотрим структурную схему и назначение каждого класса:

• App – класс приложения, наследуемый от Application – содержит точку входа программного модуля и производит загрузку вступительного и главного окна.

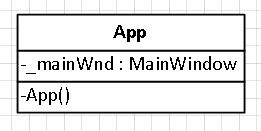


Рисунок 4. App

• WindowPre – окно предварительной загрузки, ничего стоящего внимания.

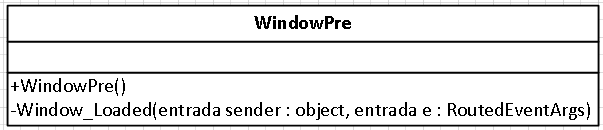


Рисунок 5.WindowPre

• FileManagementController – класс, содержащий статические методы работы с изображениями.

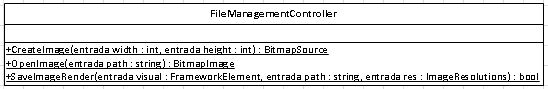


Рисунок 6. FileManagementController

• MainWindow – главное окно, управляющий класс приложения. Содержит уйму всевозможных методов и обработчиков событий. Во время выполнения многих операций обращается к методам других пользовательских классов.



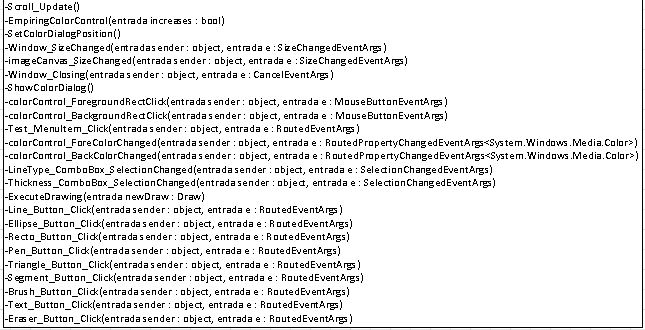


Рисунок 7. MainWindow

• Create – окно создания нового изображения. Модальное дочернее окно предыдущего, обращается к методам класса FileManagementController.

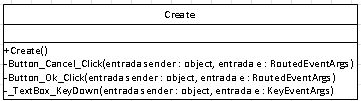


Рисунок 8. Create

• ColorDialog – вспомогательное окно настройки цвета по координатам (aRGB), или выбора из готовой палитры. Позволяет настраивать цвет основной и дополнительной кисти.

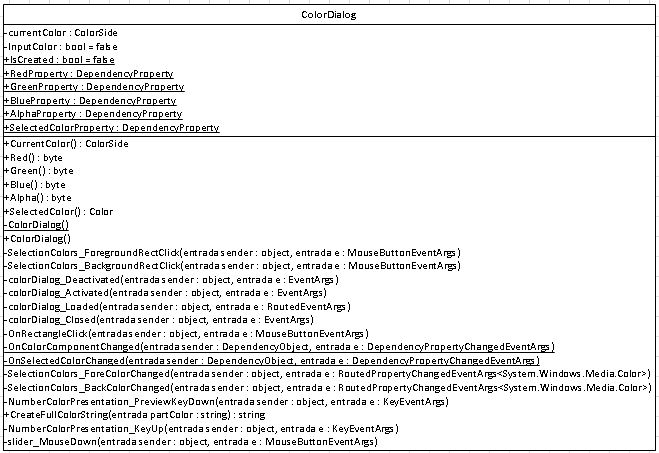


Рисунок 9. ColorDialog

• ColorControl - пользовательский элемент управления для выбора цвета (основной-дополнительный). Содержится как в главном окне, так и в диалоге выбора цвета.

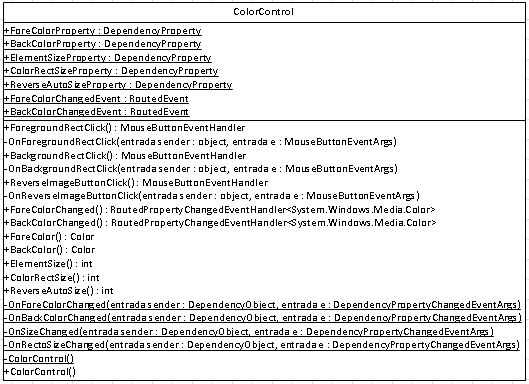


Рисунок 10. ColorControl

• Recto – пользовательский элемент управления, область для миниатюрного отображения рабочей поверхности открытой картинки.

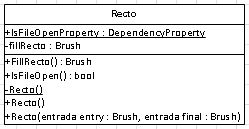


Рисунок 11. Recto

• ImageCanvas – холст-оболочка, представляющий собой контейнер для изображения, открытым в редакторе на текущий момент. Содержит список слоев рисунка, а также объект класса, конфигурирующего стек предыдущих действий ImageDataManagementController.

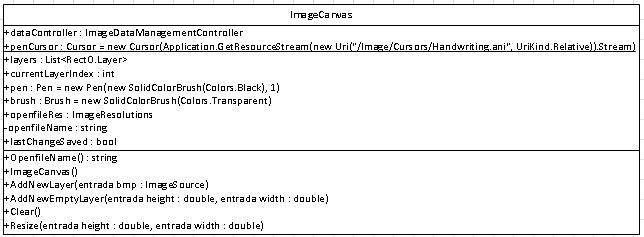


Рисунок 12. ImageCanvas

• ImageDataManagementController – класс, содержащий стек предыдущих состояний изображения (стандартные команды Undo и Redo), а также методы управления откатом к предыдущим и последующим шагам.

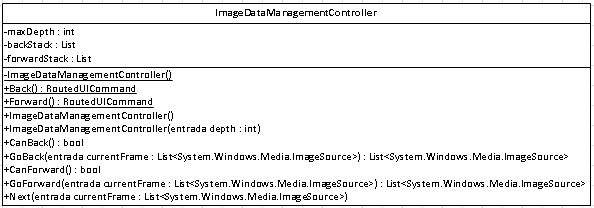


Рисунок 13. ImageDataManagementController

• Layer – один слой изображения. Вернее, снова контейнер для самого изображения, по причине технических решений. Может содержаться в любом количестве в экземпляре класса ImageCanvas.



Рисунок 14. Layer

• LayerCanvas – Внутреннее содержимое каждого слоя, некое конкретное изображение, на котором и происходит рисование. Класс наследуется от Canvas-а и переопределяет список визуальных объектов вместе с сопутствующими методами. Это было сделано для того, чтобы рисовать не на таких высоких уровнях наследственности как FrameworkElement, UIElement или Panel, а на самом низкоуровневом (из возможных в данном контексте) - визуальном уровне. Т.е. при рисовании пользователь работает с объектами класса Visual, что значительно облегчает “вес” каждого графического объекта.

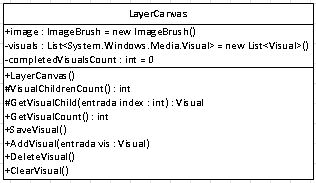


Рисунок 15. LayerCanvas

• PrimitiveDrawingDirect – данный класс является посредником между конкретными методами рисования и областью отображения графики. Он инкапсулирует в себе множество подклассов рисования, а также абстрактный класс Draw, от которого эти классы и наследуются.

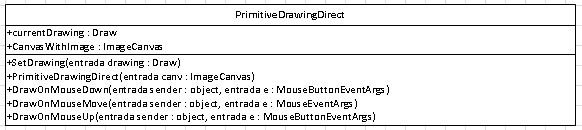


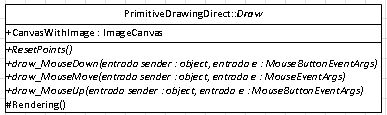
Рисунок 16. PrimitiveDrawingDirect

Рисунок 17. Draw

Классы, наследуемые от Draw, содержат методы-обработчики событий, перенаправляемые классом PrimitiveDrawingDirect на открытое изображение. Также в классе Draw имеется метод Rendering(), при помощи которого после каждого действия новая графика растрируется на исходное изображение.

Иерархия наследования класса Draw выглядит примерно так:

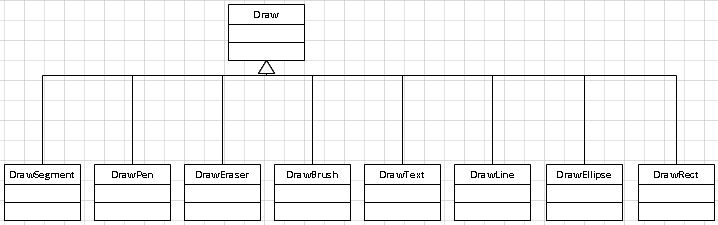


Рисунок 18. Иерархия наследования Draw

• SmallLayer – пользовательский элемент управления, представляющий собой краткое описание слоя изображения. Включает в себя динамическую визуальную пиктограмму слоя и его имя. Используется в контексте окна управления слоями.

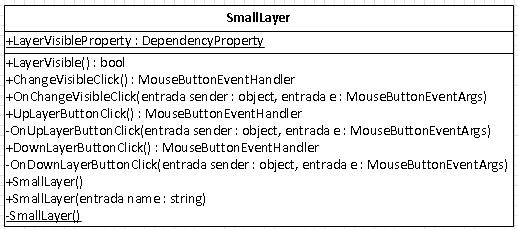


Рисунок . SmallLayer

• LayerControl – окно управления слоями текущего изображения. Содержит методы, позволяющие создавать, удалять, а также перемещать слои и скрывать их содержимое. Инкапсулирует в себе коллекцию элементов управления SmallLayer.

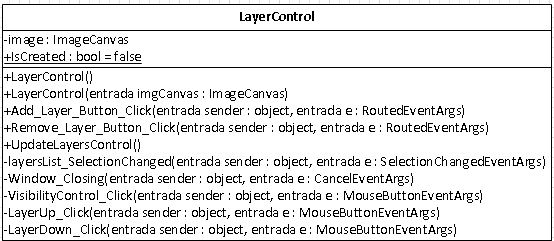


Рисунок . LayerControl

# 3.Руководство пользователя

При запуске приложения необходимо подождать некоторое время, пока оно загрузится. Пока это происходит, на экране будет отображаться вступительное окно:



Рисунок 21. Окно предзагрузки

После чего запуститься главное окно редактора. Если у вас установлена trial-версия программного средства, то система известит вас о том, что в редакторе будет заблокирован некоторый функционал:

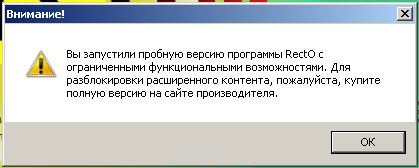


Рисунок 22. Уведомление при неполной версии

Далее у пользователя появляется полная свобода действия:

• Например, для того, чтобы загрузить на рабочую область какое-то изображение нужно выбрать пункт “Открыть” в меню “Файл”, нажать на кнопку панели инструментов, либо выполнить нажатие комбинации горячих клавиш – Ctrl + O. После чего откроется диалог открытия файла, в проводнике которого следует выбрать интересующее изображение.

• Для создания нового пустого изображения нужно нажать в меню “Файл” пункт “Создать”, либо одноименную кнопку на верхней панели инструментов, либо горячие клавиши Ctrl + O. После этого откроется специальное окно:

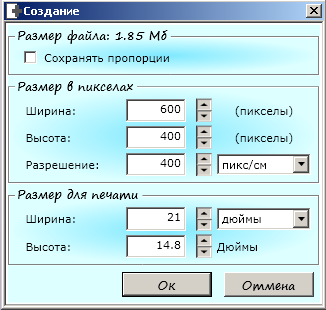


Рисунок 23. Диалог создания изображения

В диалоге создания файла можно задать такие параметры, как ширина и высота создаваемого изображения. После нажатия на кнопку “Ок” созданное изображение появится на главном окне.

• При желании отменить последнее действие нужно нажать сочетание клавиш Ctrl + Z, либо выполнить это действие через пункты меню и верхней панели инструментов. Откат произойдет мгновенно. При желании вернуть впередиидущий шаг, необходимо выполнить обратную операцию – отмену отмены последнего действия (Ctrl + Y).

• Для изменения размера изображения существует легкий и удобный способ: Нужно ухватиться мышкой за нижнюю, либо правую границу изображения, и просто передвинуть до нужного размера.

• Чтобы нарисовать что-либо, нужно выделить соответствующий инструмент. В ознакомительной бесплатной версии доступно 9 инструментов: Карандаш, кисть, ластик и текст на боковой панели инструментов, и Линия, Эллипс, Прямоугольник, Треугольник, Сектор на панели Фигур сверху. После выбора нужного способа рисования, достаточно нажать левой кнопкой мыши на поверхности изображения, чтобы начать рисовать.

• Для работы со слоями необходимо открыть окно Слоев, что можно совершить по нажатию пункта меню Окно слоев в разделе Окно. Или же, если требуются примитивные операции удаления-вставки, есть возможность выполнить данные действия без открытия соответствующего окна, а посредством кнопок на панели инструментов, либо пунктов меню Слои.

# 4.Тестирование программы

Программный продукт был хорошо оптимизирован, как в плане программных алгоритмов, так и в плане адаптивности интерфейса пользователя.

Для того, чтобы увидеть все прелести компоновки WPF, в разделе меню “Окно” был сделан специальный пункт “Тестирование компоновки”, наглядно показывающий, как будет отображаться окно при разном размере монитора у пользователей.

Как было замечено ранее, программный модуль был разработан таким образом, чтобы исключить возможности появления ошибок и исключений, в принципе.

Однако, существует несколько типов исключений, которые все-таки можно наблюдать:

При попытке открытия поврежденного изображения или файла неизвестного формата, в процессе декодирования может возникнуть следующее исключение:

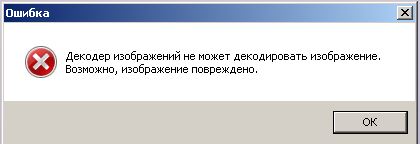


Рисунок 24. Ошибка графического декодера

Вторая же ошибка может возникнуть при работе с изображениями огромного размера. Если на компьютере не хватает оперативной памяти, пользователь получит следующее сообщение об ошибке:

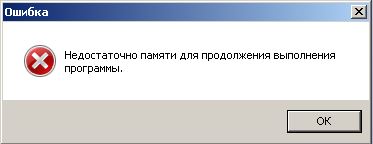


Рисунок 25. Недостаточно памяти

# Заключение

Итогом курсовой работы стало создание неполноценного графического редактора, имеющего приятный для глаза интерфейс, не сложную систему команд, и базовый набор функций, которые обязано выполнять любое программное средство обработки графической информации, а именно: открытие, закрытие и сохранения файлов графических данных, работа с функциями возврата назад и вперед, рисование разнообразных примитивов различными цветами и видами линий.

Приложение было разработано на платформе Windows Presentation Foundation, поэтому значительное внимание также было уделено компоновке элементов управления в неразделимый и цельный объект, полностью адаптивный к изменению типа и качества устройства отображения, например, монитора.

# Список используемой литературы и источников

1. Мак-Дональд М. WPF- Windows Presentation Foundation в .NET 4.0 с примерами на C#

2. Пацей Н. В. Объектно-ориентированное программирование на C++/C# В 2-ч частях, Часть 1