**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ПМ.03 Участие в интеграции программных модулей**

**Группа: 4ПКС-114**

**УТВЕРЖДАЮ**

**Председатель цикловой комиссии**

**программирования и баз данных**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Пестов А.И./**

**\_\_\_\_.\_\_\_\_. 2017**

**ПРОЕКТ КУРСОВОЙ**

**На тему: Программа для просмотра изображений c графическим интерфейсом на языке C#**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Руководитель курсового проекта**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Морозова М.В./**

**Исполнитель курсового проекта**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Тимченко А.А./**

**Оценка за проект:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_.\_\_\_\_.2017**

ОТЗЫВ РУКОВОДИТЕЛЯ КУРСОВОГО ПРОЕКТА

– соответствие сроков этапов выполнения курсового проекта срокам, указанным в графике выполнения;

– актуальность темы проекта;

– содержание работы (краткая характеристика);

– положительные стороны работы;

– практическая направленность проекта;

– отрицательные стороны проекта: ошибки, опечатки, неточности и т.д.;

– неполнота описания объектов и методов.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc499713514)

[1 ОБЩИЙ РАЗДЕЛ 5](#_Toc499713515)

[1.1 Системные требования 5](#_Toc499713516)

[1.2 Характеристика системы программирования 5](#_Toc499713517)

[2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 8](#_Toc499713518)

[2.1 Предпроектное исследование предметной области 8](#_Toc499713519)

[2.2 Анализ требований и определение спецификация программного обеспечения 10](#_Toc499713520)

[2.3 Проектирование программного обеспечения 12](#_Toc499713521)

[2.4 Разработка пользовательских интерфейсов программного обеспечения 13](#_Toc499713522)

[2.5 Тестирование и отладка программного обеспечения 14](#_Toc499713523)

[3 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ 17](#_Toc499713524)

[3.1 Руководство программиста 17](#_Toc499713525)

[3.2 Руководство пользователя 17](#_Toc499713526)

[ЗАКЛЮЧЕНИЕ 20](#_Toc499713527)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc499713528)

[ПРИЛОЖЕНИЕ А 22](#_Toc499713529)

[ПРИЛОЖЕНИЕ Б 37](#_Toc499713530)

[ПРИЛОЖЕНИЕ В 40](#_Toc499713531)

# ВВЕДЕНИЕ

Абсолютное большинство пользователей персональных компьютеров, так или иначе, сталкивалось с задачей просмотра графических изображений. Не имеет значения, с какой именно целью требовалось просмотреть то или иное изображение, так как конечным итогом этой задачи является открытие самого изображения, его удобный просмотр и, возможно, демонстрация этого изображения другому человеку с указанием на какую-либо важную деталь просматриваемого изображения.

На сегодняшний день существует огромное количество разнообразных программ для просмотра и редактирования графических изображений, все они предоставляют пользователю персонального компьютера возможность просмотра графических изображений, хоть и похожими, но абсолютно разными способами.

Большая часть приложений позволяющих просматривать графические изображения имеют сложный интерфейс с огромным количеством дополнительных функций, многие из которых не используются пользователями практически никогда.

Сложный интерфейс и огромное количество дополнительных функций пагубно влияют на скорость запуска программы при необходимости просмотреть графическое изображение. В некоторых случаях промежуток времени от момента запуска приложения до момента отображения графического изображения занимает более 5 секунд. Столь длительное ожидание просто неприемлемо. Однако, после длительного ожидания, помимо необходимого изображения пользователь увидит ещё и целое окно, массу кнопок и контекстных меню.

Целью данного курсового проекта является создание приложения, позволяющего удобно, быстро и без лишних элементов просматривать разнообразные графические изображения.

Столь строгие требования к приложению не делают неприемлемым использование классического графического интерфейса с основным окном, зоной просмотра и дополнительными кнопками действий. Интерфейсом разрабатываемого приложения станет само графическое изображение, которое пользователь захочет просмотреть.

Такой подход позволяет избавиться от тяжелого бремени окон и дополнительных кнопок действий, но так же накладывает ограничения на функционал находящийся в быстром доступе. Простой пользователь не должен запоминать несколько десятков горячих клавиш для работы с программой. Привычные действия должны происходить интуитивно и просто, иначе потеряется смысл ухода от оконного приложения.

Переместить открытое изображение можно с помощью зажатия левой кнопки мыши и перетаскивания. Закрытие происходит при нажатии по изображению правой кнопкой мыши. Приближение и отдаление происходит с помощью прокрутки колеса мыши вверх или вниз. Режим выделения и переход к окну с настройками программы будут происходить по нажатию определённых клавиш на клавиатуре.

# 1 ОБЩИЙ РАЗДЕЛ

## 1.1 Системные требования

Для запуска данного приложения персональный компьютер должен обладать, как минимум, следующими характеристиками:

– операционная система Windows 7;

– оперативная память объемом 1ГБ;

– видеокарта с поддержкой DirectX 9;

– монитор разрешением 1280 на 720 точек;

– 2МБ свободного дискового пространства;

– .NET Framework 4.5.

## 1.2 Характеристика системы программирования

Для работы над курсовым проектом было принято решение использовать среду программирования “Microsoft Visual Studio 2017”, так как языком, на котором разрабатывается приложение, является Visual C# в связке с платформой .NET Framework, так же разработанный Microsoft. Основой графического интерфейса выбрана система Windows Presentation Foundation, так как она обладает огромным количеством преимуществ относительно Windows Forms. WPF позволяет более гибко настраивать и использовать графический интерфейс, имеет расширенные графические функции.

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 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 процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Windows Presentation Foundation (WPF) – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML. В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. Технология WPF (Windows Presentation Foundation) является частью платформы .NET и представляет собой подсистему для построения графических интерфейсов. Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложится на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики. Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C# и VB.NET, либо совмещать и то, и другое.

.NET Framework – программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду. Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems (ныне принадлежит Oracle). Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Microsoft Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET на некоторых других операционных системах. В настоящее время .NET Framework получает развитие в виде .NET Core, изначально предполагающей кросcплатформенную разработку и эксплуатацию.

C# – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998-2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде Visual Studio.

# 2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 2.1 Предпроектное исследование предметной области

2.1.1 Описательная модель предметной области

На данный момент имеется большое число программ, реализующих просмотр графических изображений, в среде операционной системы Microsoft Windows.

Все они достаточно хороши, имеют круг лояльных пользователей и справляются со своей основной задачей – просмотром графических изображений, но практически все они имеют два больших минуса:

– перегруженный интерфейс;

– длительное время запуска.

Так же ни один продукт на рынке не обладает возможностью быстрого выделения части изображения для точного указания некоторых важных деталей, не прибегая к “тыканью пальцем” в монитор.

Создаваемое программное решение позволит решить обе эти проблемы. Оно избавит пользователя от ожидания загрузки программы, открытия окна и, только затем отображения изображения. Простой, интуитивно понятный интерфейс, на основе самого открываемого изображения, позволит использовать программу быстро и легко. Отсутствие гигантского количества дополнительных функций позволит пользователю сосредоточиться именно на изображении, а не на окне и кнопках вокруг изображения.

2.1.2 Описание входной информации

Входной информацией разрабатываемого приложения являются самые популярные и распространенные форматы графических изображений:

– JPEG – Joint Photographic Experts Group;

– JPG – Раннее название формата JPEG;

– PNG – Portable Network Graphics;

– TIFF – Tagged Image File Format;

– BMP – Bitmap Image File;

– GIF – Graphics Interchange Format.

Файлы изображений загружаются в программу после передачи пользователем пути, по которому они располагаются.

Дополнительно приложение получает от персонального компьютера следующую информацию:

– нажимаемые клавиши клавиатуры;

– нажимаемые кнопки мыши;

– действие прокрутки колеса мыши;

– положение и перемещения указателя мыши на экране;

– высоту и ширину экрана в пикселях.

2.1.3 Описание выходной информации

Выходной информацией разрабатываемого приложения является загруженное графическое изображение, линии нарисованные пользователем на этом же изображении и графический интерфейс окна настроек.

В определенных случаях приложение производит редактирование системного реестра, следовательно, выходной информацией так же являются ключи системного реестра.

2.1.4 Требования к программному обеспечению

2.1.4.1 Функциональные требования

Приложение, при запуске на любом из поддерживаемых устройств должно соответствовать следующим требованиям:

– быстрый запуск;

– работа с самыми популярными и распространенными форматами графических изображений: JPEG, JPG, PNG, BMP, GIF;

– увеличение и уменьшение изображения;

– предоставление пользователю возможности сделать приложение программой для открытия изображений по умолчанию нажатием одной кнопки;

– возможность моментально выделить какой-либо элемент или часть изображения;

– возможность настройки некоторых параметров приложения через отдельно окно;

– сквозная прозрачность в случае наличия у изображения прозрачных участков.

2.1.4.2 Нефункциональные требования

Основной интерфейс программы должен основываться на открываемом графическом изображении. На нем не должно располагаться ничего, кроме самого отображаемого изображения.

Открытие и закрытие приложения, после окончания просмотра изображения, должны сопровождаться плавными, но быстрыми анимациями появления и исчезновения.

Управление должно происходить через интуитивно понятные действия.

## 2.2 Анализ требований и определение спецификация программного обеспечения

На рисунке 1 представлена диаграмма вариантов использования, на ней отображены действия, совершаемые пользователем при использовании программы.

Пользователь

Рисунок 1 – Диаграмма вариантов использования

На рисунке 2 представлена контекстная диаграмма потоков данных. Она отображает потоки данных между пользователем и приложением.

На данной диаграмме можно заметить, что количество данных передаваемое пользователем программе, крайне невелико. Это обусловлено отсутствием привычного интерфейса с основным окном и кнопками.

1

Пользователь

Приложение

Нажатия клавиатуры

Действия мыши

Изменение настроек

Вывод изображения

Вывод формы настроек

1

Рисунок 2 – Контекстная диаграмма потоков данных

На рисунке 3 находится детализированная диаграмма потоков данных, она является подробной версией контекстной диаграммы потоков данных, и детальнее описывает потоки данных, проходящие между пользователем и приложением.

Загрузка изображения, получение параметров экрана и положения курсора

Работа режима выделения

Вывод изображения

1.1

Вывод окна настроек

Нажатие горячей клавиши

Вывод выделения

Сохранение настроек

1.2

1.3

Рисунок 3 – Детализированная диаграмма потоков данных

На рисунке 4 изображена контекстная функциональная диаграмма. Данная диаграмма отображает взаимосвязи функций приложения.

Приложение

Файл изображения

Параметры экрана

Изображение

на экране

Пользователь

Настройки

Алгоритмы расчета

размеров

Действия пользователя

Рисунок 4 – Контекстная функциональная диаграмма

## 2.3 Проектирование программного обеспечения

На рисунке 5 показана структурная схема разрабатываемого приложения. Структурная схема определяет основные функциональные части программы, их назначение и взаимосвязи между ними. Схема отображает принцип действия программы в самом общем виде.

Вывод изображения

Отображение окна настроек

Режим выделения

Нажатие  
«С»

Нажатие  
«S»

Нажатие  
«ПКМ»

нажатие  
«V»

Нажатие  
«ПКМ»

Нажатие  
«Save»

Нажатие  
«Exit»

Нажатие  
«Set as default»

Сохранение  
настроек

Редактирование реестра

Рисунок 5 – Структурная схема

## 2.4 Разработка пользовательских интерфейсов программного обеспечения

Главное окно программы изначально является полностью прозрачным прямоугольником без рамок, макет можно посмотреть на рисунке 6. После запуска программы оно трансформируется в открываемое изображение.



Рисунок 6 – Главное окно программы

Так же у программы имеется окно настроек. Внешний вид окна настроек показан на рисунке 7.

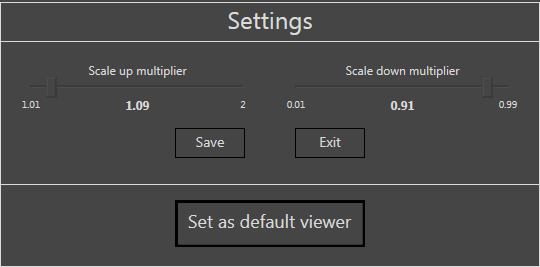


Рисунок 7 – Окно настроек программы

## 2.5 Тестирование и отладка программного обеспечения

Одним из основных требований к приложению является возможность работы с популярными форматами. Для проверки совместимости был составлен тестовый набор одинаковых изображений размером 400 на 400 пикселей в разных форматах. Результаты тестирования отображены в таблице 1.

|  |  |
| --- | --- |
| Формат изображения | Результат запуска |
| .JPEG | Работает корректно |
| .JPG | Работает корректно |
| .PNG | Работает корректно |
| .BMP | Работает корректно |
| .GIF | Работает корректно |
| .TIFF | Работает корректно |

Таблица 1 – Тестирование различных форматов

Для проверки корректности работы алгоритмов расчета размеров использовался тестовый набор, представленный в таблице 2.

|  |  |  |
| --- | --- | --- |
| Ширина(пикс.) | Высота(пикс.) | Результат запуска |
| 8400 | 2400 | Работает корректно |
| 234 | 349 | Работает корректно |
| 8700 | 1080 | Работает корректно |
| 2 | 2 | Работает корректно |
| 2083 | 2499 | Работает корректно |
| 2500 | 1200 | Работает корректно |
| 297 | 194 | Работает корректно |
| 1100 | 703 | Работает корректно |
| 1 | 1 | Работает корректно |

Таблица 2 – Тестирование различных размеров

Для проверки работы сквозной прозрачности использовались форматы .PNG и .TIFF, так как из поддерживаемых форматов только они имеют альфа канал. Тестирования показало корректную работу программы. Результат можно видеть на рисунке 8.

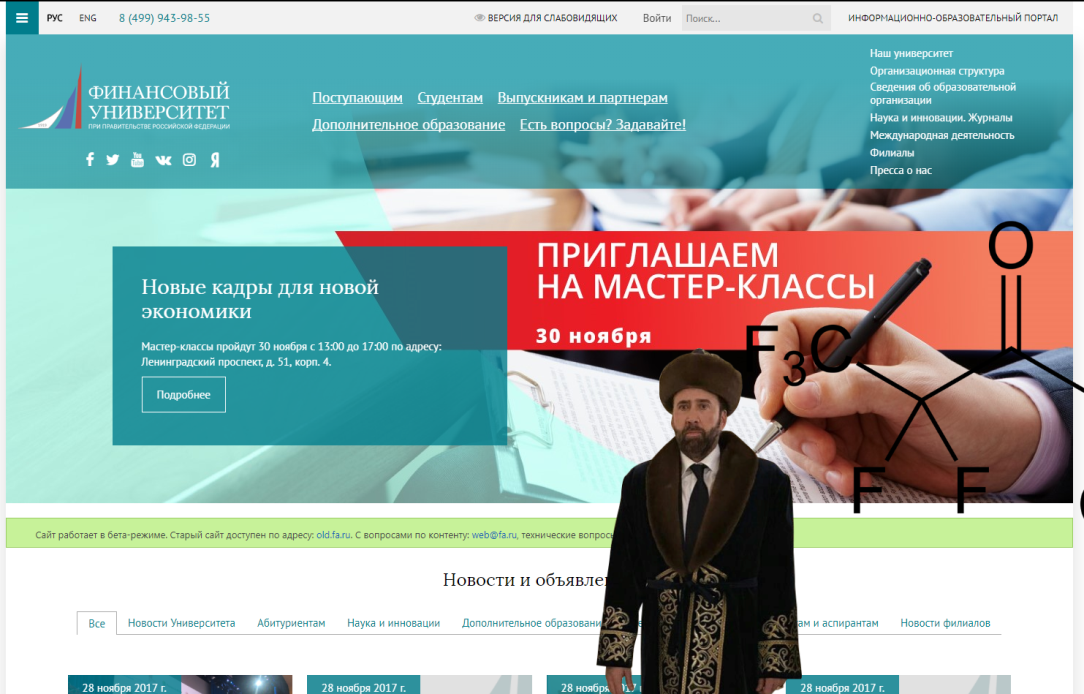


Рисунок 8 – Тестирование сквозной прозрачности

Для проверки режима выделения были выбраны все изображения из набора тестирования различных размеров. Во всех случаях режим выделения работал правильно. Один из тестов можно видеть на рисунке 10.



Рисунок 9 – Тестирование режима выделения

Для тестирования работы окна настроек производилось многократное изменение и сохранение параметров. Во всех случаях сохранённые параметры вступали в силу после перезапуска приложения.

При попытке запуска приложения без передачи изображения, приложение завершает работу без ошибок.

Алгоритмы расчета размеров изображения при изменении размеров работают без ошибок во всех протестированных случаях. Изменение размера происходит в соответствии с заданным в настройках коэффициентом. Пропорции исходного изображения не теряются.

Различные тесты показали соответствие приложения требованиям, поставленным перед разработкой.

# 3 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

## 3.1 Руководство программиста

3.1.1 Назначение и условия применения программы

Программа предназначена для удобного и быстрого просмотра графических изображений популярных форматов, а так же для выделения какого-либо участка изображения во время просмотра.

Для применения программы необходимо иметь компьютер с такими характеристиками или выше:

– операционная система Windows 7;

– оперативная память объемом 1ГБ;

– видеокарта с поддержкой DirectX 9;

– монитор разрешением 1280 на 720 точек;

– 2МБ свободного дискового пространства;

– .NET Framework 4.5.

3.1.2 Характеристики программы

Программа имеет несколько ключевых характеристик, отличающих её от других программ подобного вида:

– интуитивный интерфейс на основе простых действий;

– быстрый запуск;

– режим быстрого выделения части изображения;

– сквозная прозрачность, если таковая имеется у изображения.

## 3.2 Руководство пользователя

3.2.1 Назначение программы

Программа предназначена для просмотра изображений форматов .JPEG, .JPG, .PNG, .GIF, .TIFF и .BMP. Так же программа имеет инструмент быстрого выделения области изображения.

3.2.2 Условия выполнения программы

Для выполнения программы должны быть созданы подходящие условия, а именно, соответствие компьютера следующим требованиям:

– операционная система Windows 7 или новее;

– оперативная память объемом 1ГБ или более;

– видеокарта с поддержкой DirectX 9 или новее;

– монитор разрешением 1280 на 720 точек или более;

– 2МБ свободного дискового пространства или более;

– .NET Framework 4.5 или новее.

3.2.3 Выполнение программы

Для выполнения программы необходимо перетащить одно или несколько изображений на файл программы (PicViewer2.0.exe).

Управление программой происходит следующим образом:

– перемещение по экрану производится путём зажатия ЛКМ на любой части изображения и перетаскивания в нужную точку;

– увеличение изображения производится прокручиванием колёсика мыши в направлении от себя или нажатием стрелки вверх;

– уменьшение изображения производится прокручиванием колёсика мыши в направлении на себя или нажатием стрелки вниз;

– вход в режим выделения производится путём нажатия клавиши “C” на клавиатуре;

– выход из режима выделения производится путём нажатия клавиши “V” на клавиатуре;

– открытие окна с настройками производится путём нажатия клавиши “S” на клавиатуре;

– закрытие приложения производится нажатием ПКМ по любой части изображения.

При необходимости приложение можно сделать средством просмотра изображений по умолчанию, для этого на окне настроек необходимо нажать соответствующую кнопку.

# ЗАКЛЮЧЕНИЕ

Целью курсового проекта являлась разработка графического приложения для операционных систем семейства Windows, реализовывающего функционал просмотра изображений на языке C#. Максимальный упор делался на быстроту и простоту использования приложения. Для реализации проекта отвечающего этим требованиям пришлось использовать систему Windows Presentation Foundation совместно с платформой .NET Framework. Это позволило создать удобное и быстрое приложение со всем необходимым функционалом, однако не обошлось и без проблем. Все возникшие проблемы были связаны с относительной новизной Windows Presentation Foundation, и слишком обширными возможностями .NET Framework, среди которых было крайне легко запутаться.

Несмотря на проблемы, возникавшие во время разработки, конечный продукт получился высокого качества и его уже использует несколько десятков человек в повседневной жизни. С дальнейшим развитием приложения число довольных людей использующих его будет только расти.

Отдельно стоит сказать о реализации в программе изменяемых настроек, это позволяет сделать его удобным для каждого пользователя. На данном этапе развития программы количество настроек не столь велико, но при дальнейшей разработке будет легко реализовать настройки других параметров, ведь программный код уже для этого приспособлен.

Тестирование программы показало, что она справляется со всеми поставленными перед ней задачами.

Подводя итог можно сказать, что все поставленные цели успешно выполнены и программа представляет собой готовый продукт, однако в ней всё равно остались пути для улучшения, ведь улучшение чего угодно это бесконечный процесс.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Стандарты

1. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
2. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. ГОСТ 7.82 – 2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
4. Единая система программной документации – М.: Стандартинформ, 2005. – 128 с.

Монографии, учебники, учебные пособия

1. Гуриков С.Р. Введение в программирование на языке Visual C#: учебное пособие. – М.: ФОРУМ: ИНФРА-М, 2013.
2. Мэтью Мак-Дональд WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов – М.: apress, 2013.

Интернет ресурсы

1. Система вопросов и ответов о программировании – [Электронный ресурс]. – Режим доступа: http:/stackoverflow.com/
2. Форум начинающих и профессиональных программистов – [Электронный ресурс]. – Режим доступа: http://cyberforum.ru/
3. Полная документация .NET Framework – [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/en-us/library/ff361664(v=vs.110).aspx

# ПРИЛОЖЕНИЕ А

MainWindow.xaml.cs

using System;

using System.Drawing;

using System.Drawing.Imaging;

using System.IO;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace PicViewer2.\_0

{

/// <summary>

/// MainWindow

/// </summary>

public partial class MainWindow : Window

{

static uint MaxStartSize = Properties.Settings.Default.MaxStartSize;

static uint MinimumSize = Properties.Settings.Default.MinimumSize;

static double ScaleMultiplierUp = Properties.Settings.Default.ScaleMultiplierUp;

static double ScaleMultiplierDown = Properties.Settings.Default.ScaleMultiplierDown;

static int ScreenWidth = Convert.ToInt32(SystemParameters.PrimaryScreenWidth);

static int ScreenHeight = Convert.ToInt32(SystemParameters.PrimaryScreenHeight);

const int SmoothResizeSteps = 4;

bool Oversize = false;

bool Undersize = false;

bool Resizing = false;

double WidthDouble, HeightDouble;

Bitmap LoadedImage, MaxNativeSizeImage;

bool IsCanvasShown = false;

System.Windows.Point CurrentPoint;

/// <summary>

/// Реализует анимацию изменения размеров окна

/// </summary>

private void SmoothResize(double startX, double startY, double targetX, double targetY,

double startWidth, double startHeight, double targetWidth, double targetHeight)

{

if(Resizing == false)

{

Resizing = true;

SetWindowSizeAndPositionBySteps((targetHeight - startHeight) / SmoothResizeSteps,

(targetWidth - startWidth) / SmoothResizeSteps,

(targetX - startX) / SmoothResizeSteps,

(targetY - startY) / SmoothResizeSteps,

targetX, targetY, targetWidth, targetHeight);

Resizing = false;

}

}

/// <summary>

/// Реализует анимацию появления окна

/// </summary>

private void FadeInAnimation()

{

DoubleAnimation \_FadeIn = new DoubleAnimation(1, new TimeSpan(0, 0, 0, 0, 300));

this.BeginAnimation(OpacityProperty, \_FadeIn);

}

/// <summary>

/// Реализует анимацию закрытия окна

/// </summary>

private void FadeOutAnimation()

{

DoubleAnimation \_FadeOut = new DoubleAnimation(0, new TimeSpan(0, 0, 0, 0, 100));

\_FadeOut.Completed += FadeOutAnimation\_Completed;

this.BeginAnimation(OpacityProperty, \_FadeOut);

}

/// <summary>

/// Конвертирует Bitmap в BitmapImage

/// </summary>

private BitmapImage ConvertToBitmapImage(Bitmap picture)

{

MemoryStream \_MemStr = new MemoryStream();

picture.Save(\_MemStr, ImageFormat.Png);

\_MemStr.Position = 0;

BitmapImage \_Out = new BitmapImage();

\_Out.BeginInit();

\_Out.StreamSource = \_MemStr;

\_Out.EndInit();

return \_Out;

}

/// <summary>

/// Загружает изображение из файла в LoadedImage

/// </summary>

private void LoadImageFromFile(string path)

{

LoadedImage = new Bitmap(path);

}

/// <summary>

/// Устанавливает фоном окна изображение

/// </summary>

private void SetBackBitmapImage(BitmapImage image)

{

this.Background = new ImageBrush(image);

}

/// <summary>

/// Перемещает окно на заданную позицию

/// </summary>

private void SetWindowPosition(double x, double y)

{

this.Top = y;

this.Left = x;

}

/// <summary>

/// Изменяет размер окна на заданный

/// </summary>

private void SetWindowSize(double height, double width)

{

this.Height = height;

this.Width = width;

}

/// <summary>

/// Плавно меняет параметры окна по шагам

/// </summary>

private void SetWindowSizeAndPositionBySteps(double heightStep, double widthStep, double xStep, double yStep,

double targetX, double targetY, double targetWidth, double targetHeight)

{

for (int i = 0; i < SmoothResizeSteps - 1; i++)

{

this.Top += yStep;

this.Left += xStep;

this.Height += heightStep;

this.Width += widthStep;

}

this.Top = targetY;

this.Left = targetX;

this.Height = targetHeight;

this.Width = targetWidth;

}

/// <summary>

/// Передаёт изображение максимального нативного разрешения в MaxNativeSizeImage

/// </summary>

private void CreateMaxNativeSizeImage()

{

if ((LoadedImage.Height > ScreenHeight) && (LoadedImage.Height >= LoadedImage.Width))

{

MaxNativeSizeImage = new Bitmap(LoadedImage, Convert.ToInt32((Convert.ToDouble(ScreenHeight) / Convert.ToDouble(LoadedImage.Height)) \* LoadedImage.Width), ScreenHeight);

return;

}

if ((LoadedImage.Width > ScreenWidth) && (LoadedImage.Width >= LoadedImage.Height))

{

MaxNativeSizeImage = new Bitmap(LoadedImage, ScreenWidth, Convert.ToInt32((Convert.ToDouble(ScreenWidth) / Convert.ToDouble(LoadedImage.Width)) \* LoadedImage.Height));

return;

}

MaxNativeSizeImage = LoadedImage;

}

/// <summary>

/// Запускает анимацию исчезновения окна

/// </summary>

private void Window\_MouseRightButtonDown(object sender, MouseButtonEventArgs e)

{

FadeOutAnimation();

}

/// <summary>

/// Закрывает приложение после окончания анимации исчезновения

/// </summary>

private void FadeOutAnimation\_Completed(object sender, EventArgs e)

{

this.Hide();

Environment.Exit(0);

}

/// <summary>

/// Перемещает главное окно при зажатии на нём ЛКМ

/// </summary>

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

if (IsCanvasShown == false)

{

this.DragMove();

}

}

/// <summary>

/// Загрузка изображения и запуск инициализации

/// </summary>

public MainWindow()

{

string[] args = Environment.GetCommandLineArgs();

try

{

LoadImageFromFile(args[1]);

}

catch

{

Environment.Exit(0);

}

InitializeComponent();

this.Title = "PicViewer2.0 - " + System.IO.Path.GetFileName(args[1]);

}

/// <summary>

/// Рассчитывает первоначальные размеры изображения, устанавливает фон и перемещает на место курсора

/// </summary>

private void Window\_Initialized(object sender, EventArgs e)

{

double \_LoadedImageHeight = LoadedImage.Height;

double \_LoadedImageWidth = LoadedImage.Width;

CreateMaxNativeSizeImage();

if (\_LoadedImageWidth > \_LoadedImageHeight)

{

if (\_LoadedImageWidth > MaxStartSize)

{

WidthDouble = MaxStartSize;

}

else

{

WidthDouble = LoadedImage.Width;

}

HeightDouble = \_LoadedImageHeight / (\_LoadedImageWidth / WidthDouble);

}

else

{

if (\_LoadedImageHeight > MaxStartSize)

{

HeightDouble = MaxStartSize;

}

else

{

HeightDouble = LoadedImage.Height;

}

WidthDouble = \_LoadedImageWidth / (\_LoadedImageHeight / HeightDouble);

}

System.Windows.Point \_Mouse = NativeMethods.GetMousePosition();

SetWindowPosition(\_Mouse.X - (WidthDouble / 2d), \_Mouse.Y - (HeightDouble / 2d));

SetWindowSize(HeightDouble, WidthDouble);

SetBackBitmapImage(ConvertToBitmapImage(MaxNativeSizeImage));

}

/// <summary>

/// Запускает анимацию появления

/// </summary>

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

FadeInAnimation();

}

/// <summary>

/// Расчитывает размер окна при увеличении

/// </summary>

private void ScaleUpWindowCalculation(double width, double height)

{

//Оверсайз по ширине

if (((WidthDouble \* ScaleMultiplierUp) > ScreenWidth) || Oversize == true)

{

HeightDouble += (HeightDouble / WidthDouble) \* (ScreenWidth - WidthDouble);

WidthDouble = ScreenWidth;

Oversize = true;

return;

}

else

{

WidthDouble = width \* ScaleMultiplierUp;

HeightDouble = height \* ScaleMultiplierUp;

}

//Оверсайз по высоте

if (((HeightDouble \* ScaleMultiplierUp) > ScreenHeight) || Oversize == true)

{

WidthDouble += (WidthDouble / HeightDouble) \* (ScreenHeight - HeightDouble);

HeightDouble = ScreenHeight;

Oversize = true;

return;

}

else

{

WidthDouble = width \* ScaleMultiplierUp;

HeightDouble = height \* ScaleMultiplierUp;

}

}

/// <summary>

/// Изменяет размер окна при уменьшении

/// </summary>

private void ScaleUpWindow()

{

if (Oversize == false && IsCanvasShown == false)

{

ScaleUpWindowCalculation(WidthDouble, HeightDouble);

SmoothResize(this.Left, this.Top, this.Left - ((WidthDouble - this.Width) / 2d), this.Top - ((HeightDouble - this.Height) / 2d),

this.Width, this.Height, WidthDouble, HeightDouble);

if (Undersize == true)

{

Undersize = false;

}

}

/// <summary>

/// Расчитывает размер окна при уменьшении

/// </summary>

private void ScaleDownWindowCalculation(double width, double height)

{

//Андерсайз по ширине

if (((WidthDouble \* ScaleMultiplierDown) < MinimumSize) || Undersize == true)

{

Undersize = true;

return;

}

else

{

WidthDouble = width \* ScaleMultiplierDown;

HeightDouble = height \* ScaleMultiplierDown;

}

//Андерсайз по высоте

if (((HeightDouble \* ScaleMultiplierDown) < MinimumSize) || Undersize == true)

{

Undersize = true;

return;

}

else

{

WidthDouble = width \* ScaleMultiplierDown;

HeightDouble = height \* ScaleMultiplierDown;

}

}

/// <summary>

/// Изменяет размер окна при уменьшении

/// </summary>

private void ScaleDownWindow()

{

if (Undersize == false && IsCanvasShown == false)

{

ScaleDownWindowCalculation(WidthDouble, HeightDouble);

SmoothResize(this.Left, this.Top, this.Left + ((this.Width - WidthDouble) / 2d), this.Top + ((this.Height - HeightDouble) / 2d),

this.Width, this.Height, WidthDouble, HeightDouble);

if (Oversize == true)

{

Oversize = false;

}

}

}

/// <summary>

/// Обрабатывает прокрутку колеса мыши

/// </summary>

private void Window\_MouseWheel(object sender, MouseWheelEventArgs e)

{

if (e.Delta > 0)

{

ScaleUpWindow();

}

else

{

ScaleDownWindow();

}

}

/// <summary>

/// Обрабатывает нажатия кнопок мыши

/// </summary>

private void Window\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.Key)

{

case Key.Up:

ScaleUpWindow();

break;

case Key.Down:

ScaleDownWindow();

break;

case Key.S:

this.Hide();

SettingsWindow \_SettingsWindow = new SettingsWindow();

System.Windows.Point \_Mouse = NativeMethods.GetMousePosition();

\_SettingsWindow.Show();

\_SettingsWindow.Top = \_Mouse.Y - (\_SettingsWindow.Height / 2d);

\_SettingsWindow.Left = \_Mouse.X - (\_SettingsWindow.Width / 2d);

break;

case Key.C:

ShowCanvas();

break;

case Key.V:

HideCanvas();

break;

default:

break;

}

}

/// <summary>

/// Рисует линию на canvas

/// </summary>

private void DrawingCanvas\_MouseMove(object sender, MouseEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

{

Line line = new Line

{

Stroke = new SolidColorBrush(System.Windows.Media.Color.FromRgb(255, 0, 0)),

StrokeThickness = 6,

X1 = CurrentPoint.X,

Y1 = CurrentPoint.Y,

X2 = e.GetPosition(this).X,

Y2 = e.GetPosition(this).Y,

StrokeStartLineCap = PenLineCap.Round,

StrokeEndLineCap = PenLineCap.Round

};

CurrentPoint = e.GetPosition(this);

DrawingCanvas.Children.Add(line);

}

}

/// <summary>

/// Получает координаты курсора при нажатии ЛКМ по canvas

/// </summary>

private void DrawingCanvas\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.ButtonState == MouseButtonState.Pressed)

{

CurrentPoint = e.GetPosition(this);

}

}

/// <summary>

/// Устанавливает фон Canvas, удаляет фон формы и разворачивает Canvas

/// </summary>

private void ShowCanvas()

{

if (IsCanvasShown == false)

{

DrawingCanvas.Background = this.Background;

this.Background = new SolidColorBrush(System.Windows.Media.Color.FromArgb(0, 0, 0, 0));

DrawingCanvas.Height = this.Height;

DrawingCanvas.Width = this.Width;

IsCanvasShown = true;

}

}

/// <summary>

/// Скрывает и очищает Canvas, возвращает фон формы

/// </summary>

private void HideCanvas()

{

if (IsCanvasShown == true)

{

SetBackBitmapImage(ConvertToBitmapImage(MaxNativeSizeImage));

DrawingCanvas.Children.Clear();

DrawingCanvas.Height = 0;

DrawingCanvas.Width = 0;

IsCanvasShown = false;

}

}

}

}

NativeMethods.cs

using System;

using System.Runtime.InteropServices;

namespace PicViewer2.\_0

{

class NativeMethods

{

[DllImport("user32.dll")]

[return: MarshalAs(UnmanagedType.Bool)]

internal static extern bool GetCursorPos(ref System.Drawing.Point pt);

/// <summary>

/// Возвращает позицию курсора

/// </summary>

public static System.Windows.Point GetMousePosition()

{

System.Drawing.Point pt = new System.Drawing.Point();

GetCursorPos(ref pt);

return new System.Windows.Point(pt.X, pt.Y);

}

[DllImport("shell32.dll", CharSet = CharSet.Auto, SetLastError = true)]

/// <summary>

/// Уведомляет систему об изменениях в оболочке

/// </summary>

public static extern void SHChangeNotify(uint wEventId, uint uFlags, IntPtr dwItem1, IntPtr dwItem2);

}

}

SettingsWindow.xaml.cs

using Microsoft.Win32;

using System;

using System.Reflection;

using System.Windows;

using System.Windows.Input;

namespace PicViewer2.\_0

{

/// <summary>

/// Логика взаимодействия для SettingsWindow.xaml

/// </summary>

public partial class SettingsWindow : Window

{

/// <summary>

/// Инициализирует окно настроек

/// </summary>

public SettingsWindow()

{

InitializeComponent();

this.ScaleUpSlider.Value = Properties.Settings.Default.ScaleMultiplierUp;

this.ScaleUpLabel.Content = ScaleUpSlider.Value;

this.ScaleDownSlider.Value = Properties.Settings.Default.ScaleMultiplierDown;

this.ScaleDownLabel.Content = ScaleDownSlider.Value;

}

/// <summary>

/// Реализует перемещение окна настроек

/// </summary>

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

/// <summary>

/// Скрывает окно настроек и завершает программу

/// </summary>

private void ExitLabel\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

this.Hide();

Environment.Exit(0);

}

/// <summary>

/// Сохраняет настройки программы

/// </summary>

private void SaveLabel\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

Properties.Settings.Default.ScaleMultiplierUp = Convert.ToDouble(ScaleUpLabel.Content);

Properties.Settings.Default.ScaleMultiplierDown = Convert.ToDouble(ScaleDownLabel.Content);

Properties.Settings.Default.Save();

}

/// <summary>

/// Отображает текущее значение слайдера ScaleUpSlider

/// </summary>

private void ScaleUpSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

this.ScaleUpLabel.Content = Math.Round(ScaleUpSlider.Value, 2);

}

/// <summary>

/// Отображает текущее значение слайдера ScaleDownSlider

/// </summary>

private void ScaleDownSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

this.ScaleDownLabel.Content = Math.Round(ScaleDownSlider.Value, 2);

}

/// <summary>

/// Привязывает поддерживаемые расширения к приложению

/// </summary>

private void SetAsDefaultButton\_Click(object sender, RoutedEventArgs e)

{

SetFileAssociation(".jpg");

SetFileAssociation(".jpeg");

SetFileAssociation(".png");

SetFileAssociation(".gif");

SetFileAssociation(".tiff");

SetFileAssociation(".bmp");

}

/// <summary>

/// Редактирует реестр для привязки расширения к приложению

/// </summary>

public static void SetFileAssociation(string extension)

{

string \_KeyName = "PicViewer2.0";

string \_OpenWith = Assembly.GetEntryAssembly().Location;

RegistryKey \_BaseKey;

RegistryKey \_OpenMethod;

RegistryKey \_Shell;

RegistryKey \_CurrentUser;

\_BaseKey = Registry.ClassesRoot.CreateSubKey(extension);

\_BaseKey.SetValue("", \_KeyName);

\_OpenMethod = Registry.ClassesRoot.CreateSubKey(\_KeyName);

\_OpenMethod.CreateSubKey("DefaultIcon").SetValue("", "\"" + \_OpenWith + "\",0");

\_Shell = \_OpenMethod.CreateSubKey("Shell");

\_Shell.CreateSubKey("open").CreateSubKey("command").SetValue("", "\"" + \_OpenWith + "\"" + " \"%1\"");

\_BaseKey.Close();

\_OpenMethod.Close();

\_Shell.Close();

\_CurrentUser = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\FileExts\\" + extension, true);

\_CurrentUser.DeleteSubKey("UserChoice", false);

\_CurrentUser.Close();

NativeMethods.SHChangeNotify(0x08000000, 0x0000, IntPtr.Zero, IntPtr.Zero);

}

}

}

# ПРИЛОЖЕНИЕ Б

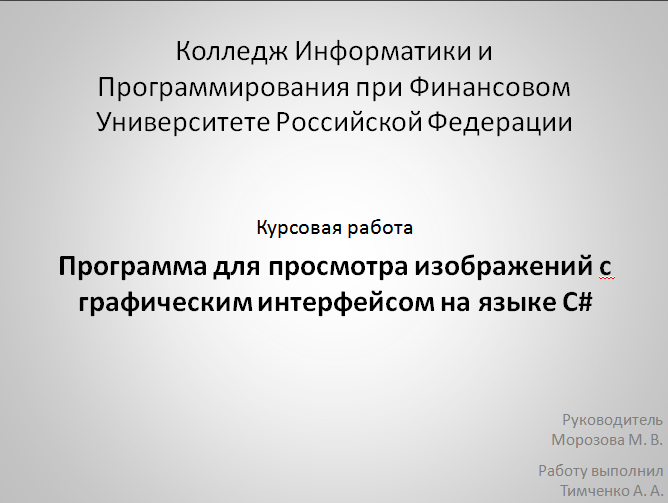


Рисунок Б.1 – Слайд презентации №1



Рисунок Б.2 – Слайд презентации №2

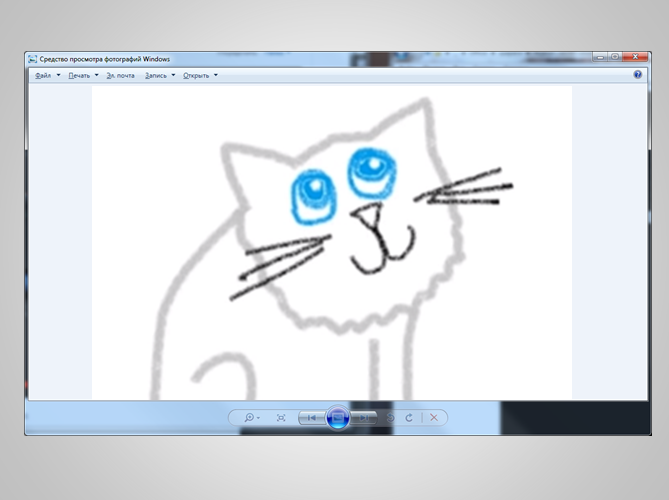


Рисунок Б.3 – Слайд презентации №3



Рисунок Б.4 – Слайд презентации №4

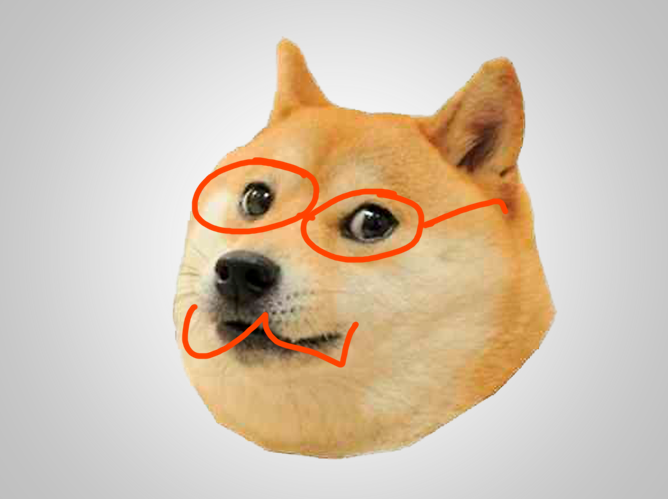


Рисунок Б.5 – Слайд презентации №5

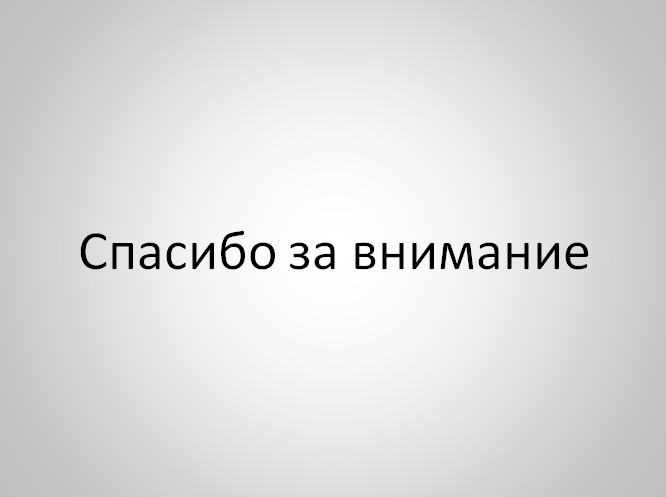


Рисунок Б.6 – Слайд презентации №6

# ПРИЛОЖЕНИЕ В

Текст слайда 1:

Давайте начнем?

\*клик\*

Текст слайда 2:

Мы смотрим много изображений за день, многие из них на ПК. Я ведь прав? И не пытайтесь отрицать.

Как мы обычно это делаем?

\*клик\*

Текст слайда 3:

Мы два раза кликаем по картинке и ждем пока откроется окно программы.

Потом ещё ждём…. Долго, наверное, ждём…. О! Дождались!

И что мы видим? Мы хотели изображение а у нас тут ещё какое-то окно и гора кнопкок. Мы этого не просили!

А теперь смотрите что я придумал…

\*клик\*

Текст слайда 4:

Мы открываем файл…

И вжух. У нас есть изображение. Сразу. Ну только немного анимации было, но я думаю так красивее.

А что если я захочу изменить размер?

\*клик\*

Я просто покручу колёсико мышки: вверх или вниз, смотря что мне нужно.

А как двигать?

\*клик\*  
Да просто беру и двигаю. За любую часть.

Но что будет если у нас есть что-то прозрачное на изображении?  
\*клик\*  
А ничего. Ничего там не будет. Прозрачное же.

\*клик\*

Что вы на это скажете?

Ах, да, забыл, закрывается на ПКМ.

Но это ещё не всё. Мы часто показываем изображения друг-другу и тыкаем пальцем в экран, ну или, в лучшем случае водим курсором.  
\*клик\*

Текст слайда 5:

Но курсор не всегда бывает видно, поэтому тут можно чуть-чуть пообводить…  
Ну или просто порисовать каракули.

Текст слайда 6:

Спасибо за внимание!