**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

в рамках ПМ.02 «Осуществление интеграции программных модулей»

СТУДЕНТ: Орлов Александр Анатольевич

Семестр 3

Специальность: 09.02.07 Информационные системы и программирование

РУКОВОДИТЕЛЬ: Шишерина Марина Александровна

Работа защищена с оценкой: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc123037381)

[**1.** **Техническое задание** 4](#_Toc123037382)

[**1.1.** **Назначение разработки и постановка задачи** 4](#_Toc123037383)

[**1.2.** **Требования к программному продукту** 4](#_Toc123037384)

[**1.3.** **Стадии и этапы разработки** 5](#_Toc123037385)

[**1.4.** **Критерии оценки готовности программного продукта** 5](#_Toc123037386)

[**2.** **Алгоритмы реализации** 6](#_Toc123037387)

[**2.1.** **Выбор типов данных** 6](#_Toc123037388)

[**2.2.** **Выбор элементов управления** 7](#_Toc123037389)

[**2.3.** **Алгоритмы рисования фигур** 11](#_Toc123037390)

[**3.** **Инструменты реализации** 16](#_Toc123037391)

[**4.** **Разработка программного продукта** 17](#_Toc123037392)

[**5.** **Тестирование программного продукта** 26](#_Toc123037393)

[**6.** **Выгрузка локального репозитория на удаленный сервер** 28](#_Toc123037394)

[**7.** **Руководство пользователя** 28](#_Toc123037395)

[**ЗАКЛЮЧЕНИЕ** 29](#_Toc123037396)

# **ВВЕДЕНИЕ**

Учебная практика является важной частью учебного процесса. Основной целью учебной-ознакомительной практики является сбор и аналитическая обработка материала, систематизация, закрепление знаний и навыков, полученных в период обучения.

**Цель учебной практики** по МП 06 «Сопровождение информационных систем» состоит в закреплении навыков работы со вспомогательными техническими продуктами для разработки и поддержки программного обеспечения, а также в закреплении понимания основных этапов жизненного цикла программного обеспечения.

Для достижения цели учебно-ознакомительной практики выделены следующие **задачи:**

* Изучить предметную область проекта;
* Исходя из требований составить техническое задание на разработку программного продукта;
* Выбрать инструментальные средства разработки программы;
* Сформировать критерии готовности программы;
* Выбрать подходящие алгоритмы реализации;
* Реализовать программу;
* Составить тест-кейс и осуществить проверку программного продукта на наличие критических ошибок;
* Реализовать работу с локальным и глобальным git-репозиторием;
* Разработать руководство пользователя;
* Написать отчетную документацию.

Данные задачи призваны систематизировать работу на практике, сделать ее более структурированной и поэтапной. В результате работы необходимо получить готовый программный продукт, тест-кейсы для проверки, опубликовать проект на платформе GitHub, составить итоговый отчет и заполнить дневник по практике.

# **Техническое задание**

## **Назначение разработки и постановка задачи**

Разрабатывается графический редактор. Данный программный продукт позволит пользователю рисовать из имеющихся инструментов, а также сохранять или загружать уже имеющиеся рисунки.

В рамках учебной практики требуется реализовать следующие задачи:

1. Составить требования к своему программному продукту;
2. Выбрать инструментальные средства разработки программы;
3. Разработать структуру редактора и его алгоритм работы;
4. Спроектировать простой и понятный пользовательский интерфейс;
5. Протестировать программный продукт;
6. Написать руководства по использованию программного продукта для пользователя;
7. Сформировать отчет о проделанной работе.

После постановки задач, можно приступать к непосредственному их выполнению.

## **Требования к программному продукту**

Функциональные требования:

* Открытие и сохранение изображения;
* Рисование линий при помощи мыши;
* Рисований многоугольников, где количество углов задает пользователь;
* Возможность выбор цвета;
* Наличие обводки фигуры определенным цветов;
* Наличие заливки фигур определенным цветом;
* Функция очистки экрана полностью или определенную его часть;

Данные требования обязаны быть реализованы в разрабатываемом программном продукте.

## **Стадии и этапы разработки**

Для лучшего понимания хода разработки программного продукта, следует обозначить стадии и этапы разработки.

1. Написание технического задания
   1. Анализ предметной области графического редактора
   2. Составление требований к собственному продукту
2. Проектирование графического редактора
   1. Спроектировать структуру графического редактора
   2. Спроектировать алгоритм работы графического редактора
   3. Спроектировать пользовательский интерфейс
3. Разработка графического редактора
   1. Разработка структуры редактора
   2. Разработка алгоритмов работы каждого элемента структуры
   3. Разработка дизайна пользовательского интерфейса
4. Тестирование графического редактора
   1. Создание тест-кейсов
   2. Проведение тест-кейсов
5. Сдача отчета

Таким образом я систематизировал свою деятельность по разработке программного продукта.

## **Критерии оценки готовности программного продукта**

Программный продукт считается готовым к использованию, если в нем:

* Реализованы все функциональные требование, обозначенные в техническом задании;
* Успешно пройдены тест-кейсы и не выявлены какие-либо критические ошибки.

После выполнения вышеперечисленных условий, разработка программного продукта считается успешно законченной и дальнейшими действиями является лишь ее поддержка.

# **Алгоритмы реализации**

## **Выбор типов данных**

Для начала необходимо определить какие переменные будут использоваться в программном продукте и какой у них тип данных. Для этого была составлена таблица с пояснением:

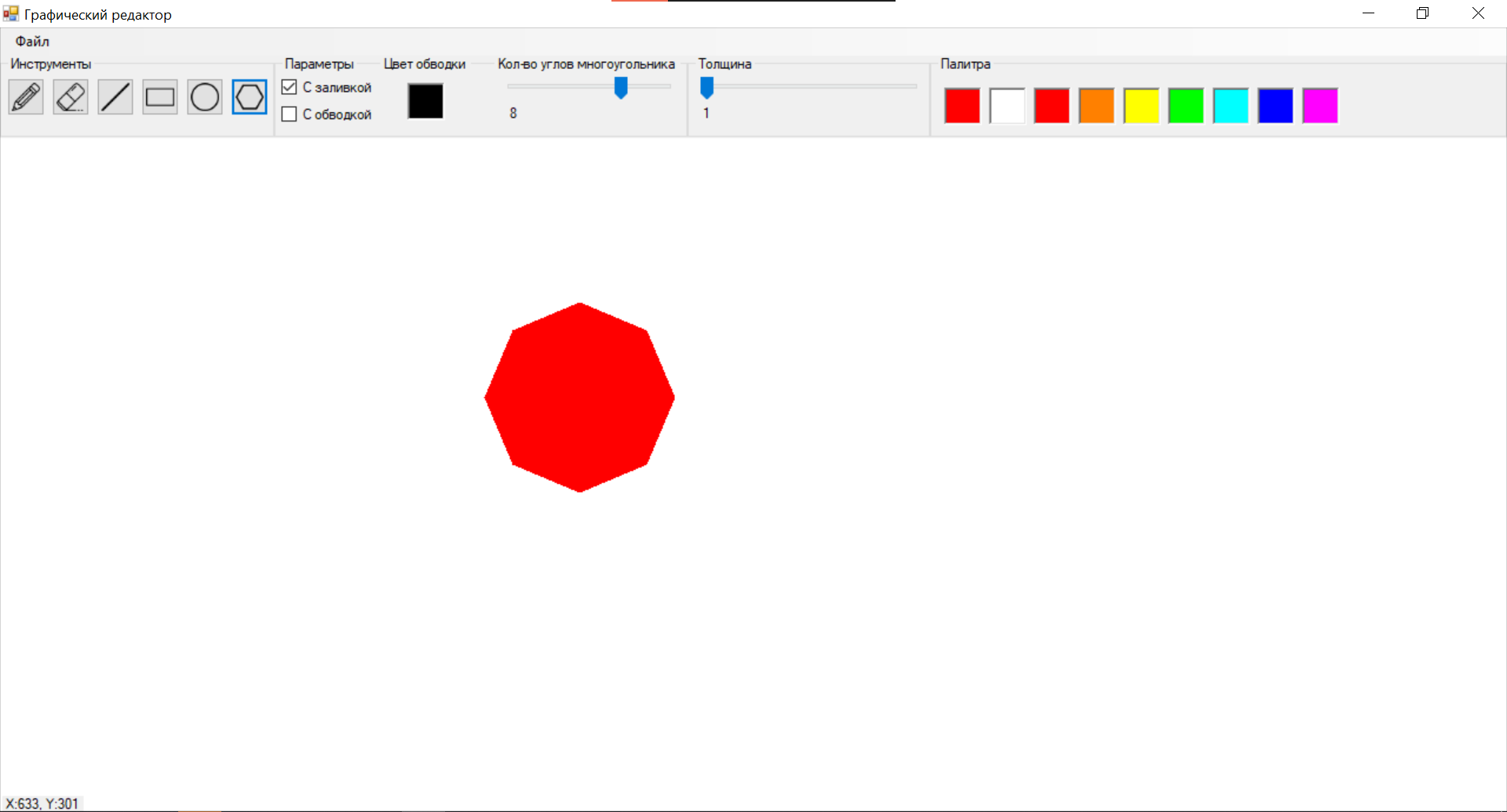
**Таблица 2.1.1 Переменные**

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Наименование переменной** | **Примечание** |
| Graphics | canvas | Поверхность для рисования |
| Bitmap | bitmap | Рисунок |
| Brush | brush | Кисть, необходимая для заливки фигур |
| Pen | pen | Карандаш, необходимый для отрисовки фигур |
| string | mode | Режим рисования, определяемый выбором инструмента |
| bool | isErasing | Отображает режим ластика |
| bool | fill | Отвечает за наличие заливки у фигуры |
| bool | border | Отвечает за наличие обводки у фигуры |
| int | x, y | Начальные координаты рисования |
| int | x1, y1 | Конечные координаты рисования |
| Points[] | p | Массив объектов класса Points, хранящие в себе координаты точек |

Таким образом мы разобрались с переменными и теперь можно приступать к выбору элементов управления

## **Выбор элементов управления**

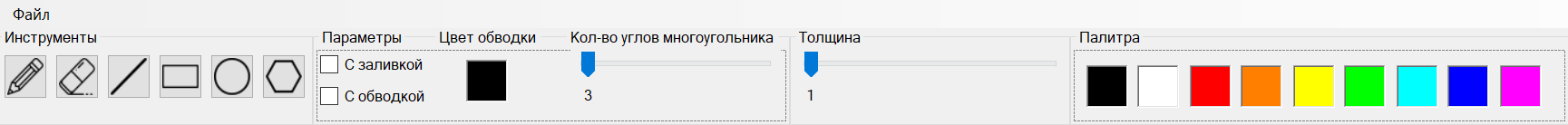
Интерфейс редактора разделен на панель рисования и панель инструментов. Под каждый из разделов был создан элемент управления Panel, позволяющий группировать другие элементы.



**Рис. 2.2.1 Интерфейс графического редактора**

Для реализации рисования в качестве холста был взять такой элемент управления как PictureBox. Он отображает заданное ему изображение и с ним довольно просто работать. Но для того, чтобы при изменении размеров окна PictureBox «ввел себя» корректно, в качестве родительского контейнера был взять ранее созданный Panel. Также в качестве параметра Dock в PictureBox взято значение Fill.

Для реализации полосы инструментов использовался GroupBox. Данный элемент необходим был для разделения частей полосы друг от друга. Отличием GroupBox от Panel является возможность включения заголовка и наличие обводки, что было необходимо для понятного отображения информации пользователю.



**Рис. 2.2.2 Полоса инструментов**

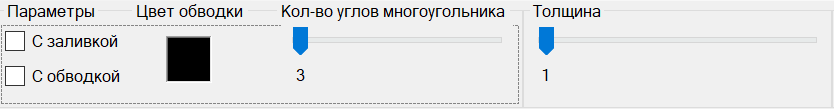
Панель с основными инструментами содержит в себе кнопки – элементы Button. Всего их 6 и каждая из них отвечает за рисование. Рассмотрим их по порядку:

1. «Карандаш» – позволяет пользователю рисовать от руки произвольным образом на холсте;
2. «Ластик» – позволяет пользователю стирать произвольную нарисованную область;
3. «Линия» – позволяет пользователю рисовать линию по двум заданным им точкам;
4. «Прямоугольник» – позволяет пользователю рисовать прямоугольник по двум заданным им точкам;
5. «Эллипс» – позволяет пользователю рисовать эллипс по двум заданным им точкам;
6. «Многоугольник – позволяет пользователю рисовать многоугольник по двум заданным им точкам. Количество углов определяется самим пользователем при помощи ползунка в панели «Параметры»



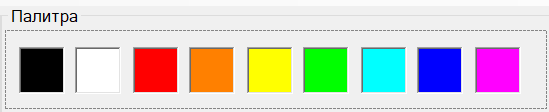
**Рис. 2.2.3 Панель «Инструменты»**

Следом идет панель «Параметры». Данная область отвечает за настройку карандаша и фигур. Первое что мы видим это элементы CheckBox, отвечающие за выбор пользователя рисовать с заливкой или обводкой. Далее идет выбор цвета обводки и для его реализации был взять PictureBox, к которому привязан ColorDialog. Данный компонент позволяет спрашивать у пользователя в виде диалогового окна о выборе цвета. Следом можно заметить два ползунка, один отвечает за количество углов многоугольника, а другой за толщины рисования линии. Оба ползунка были реализованы через TrackBar. Так как данный элемент не обладает числовым отображением значения, то пришлось использовать элемент Label и размещать его под ползунком. Также такие заголовки как «Цвет обводки» и «Кол-во углов многоугольника» были созданы через элемент управления Label, так как GroupBox не поддерживает больше одного заголовка.



**Рис. 2.2.4 Панель «Параметры»**

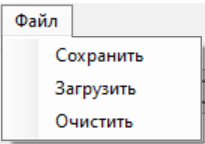
Последняя панель в полосе – это панель «Палитра». Она служит для выбора и отображения цвета. Все эти элементы созданы при помощи PictureBox. Первый PictureBox отображает цвет рисования и имеет диалоговое окно ColorDialog. Аналогично со следующим элементом, но здесь уже отображается цвет ластика. Последующие PictureBox не имеют диалоговых окон, а служат лишь ради выбора цвета, которым они закрашены.



**Рис. 2.2.5 Панель «Палитра»**

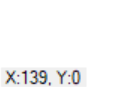
Остается рассмотреть меню и отображение координат. Начнем с меню. Оно создано через элемент управления MenuStrip. Каждый компонент здесь является ToolStripMenuItem, который может расскрываться и хранить в себе подобные себе элементы. В нашем случае в меню «Файл» содержится пункты:

* «Сохранить» – сохраняет рисунок в виде изображение у пользователя;
* «Загрузить» – загружает изображение, выбранное пользователем в редактор;
* «Очистить» – очищает текущую область рисунка.

****

**Рис. 2.2.6 Меню «Файл»**

Последнее что осталось это отображение координат. Создано оно через Label и привязано к нижнему левому углу.

****

**Рис. 2.2.7 Координаты**

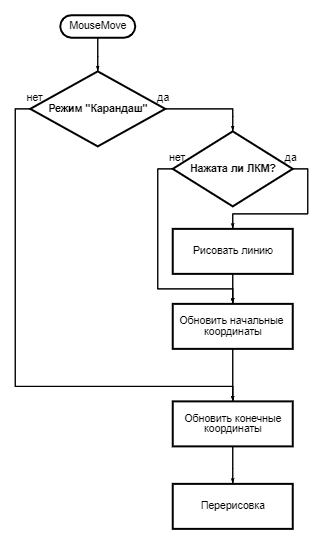
Таким образом мы рассмотрели все элементы управления необходимые для разработки графического редактора.

## **Алгоритмы рисования фигур**

Для того, чтобы что-то рисовать, необходимо для начала создать область рисования при помощи объекта класса Graphics и поместить в него объект класса Bitmap (т.е. сам рисунок, изображение). Далее на изображении что-то нарисовать и присвоить PictureBox свойству Image. Данный порядок действий одинаковый для рисования любым из предоставленных инструментов, но как конкретно рисует, тот или иной инструмент отличается, поэтому давайте рассмотрим это более детально.

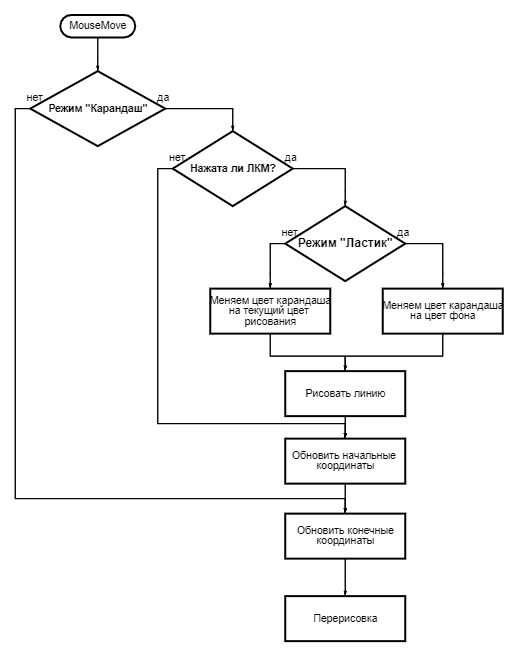
Начнем с того, что рисование само по себе происходит за счет реагирование формы на триггеры, т.е. какие-либо события, которые пользователь вызывает. Примерами таких событий может быть нажатие кнопки мыши, ее зажатие, передвижение и прочее. Как раз из таких событий и складывается алгоритм рисования фигур, о чем далее подробнее.

Карандаш – самый простой инструмент, который можно себе представить. Рисует он сразу, поэтому весь алгоритм будет находится в MouseMove (событие, отвечающие за перемещение мыши по области объекта, вызывающий данный метод). Чтобы рисовать при помощи объекта класса Graphics необходимо использовать один из существующих методов, начиная от рисования простых линий, прямоугольников, кругов, заканчивая кривыми Безье, полигонами и прочим. Самым простым вариантов будет использовать метод DrawLine(), а в качестве параметров мы будем использовать ранее созданный карандаш, координаты x, y первой точки и координаты x1,y1 второй точки. В качестве начальных координат мы записываем координаты нажатия мыши, а это отвечает событие MouseDown(). Таким образом мы получаем следующую блок-схему:



**Рис. 2.3.1 Блок-схема алгоритма работы события MouseMove**

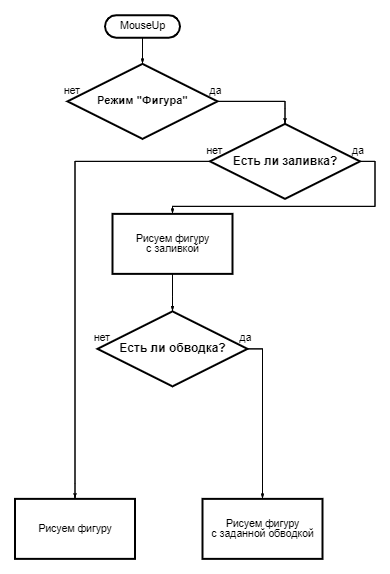
В качестве конечных координат в методе DrawLine() будет использоваться значение текущих координат мыши, ведь карандаш должен рисовать сразу вслед за мышью. Для получения текущих координат мыши используется параметр события – объект класса MouseEventArgs, содержащий в себе данные о событиях мыши. Данные, которые нам нужны, как раз и являются координаты мыши X и Y. По такому же принципу работает ластик. По факту это все тот же карандаш, но с цветом равным цвету фона. Поэтому для него алгоритм будет тот же, но с небольшим дополнением.



**Рис. 2.3.2 Дополненная блок-схема алгоритма работы события MouseMove**

С карандашом разобрались, далее движемся к фигурам. Они отличаются от обычного рисования карандашом, как минимум наличием формы, а значит и размеров. Получается, чтобы корректно нарисовать фигуру, необходимо точно определиться с ее конечными координатами (т.е. отпустить кнопку мыши). Для этого существует событие MouseUp().

Для начала нужно определить в каком режиме мы рисуем, т.е. какой тип фигуры. Не стоит забывать о наличии обводки у фигуры и ее заливки. После проверки всех условий можно рисовать фигуру. По итогу мы получаем следующие блок-схемы:



**Рис. 2.3.3 Обобщённая блок-схема алгоритма работы события MouseUp для фигур**

В качестве режима «Фигура» подразумевается любой из существующих в редакторе режим рисования фигур, за исключением линии. Для нее алгоритм выглядит еще проще:



**Рис. 2.3.4 Обобщённая блок-схема алгоритма работы события MouseUp для линии**

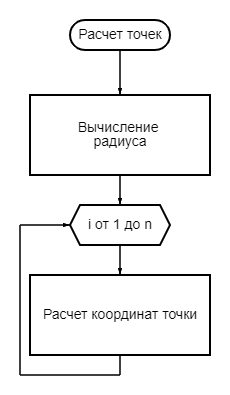
В качестве конечных координат используется координаты отпускания мыши. Их мы записали еще в MouseMove, когда обновляли конечные координаты.

Эллипс и прямоугольник мы рисуем по двум точкам, но рисование многоугольника происходит по другому алгоритму и о нем дальше. В качестве параметров функция использует все также ручку, но вместе координат двух точек используется массив точек. Поэтому в начале алгоритма необходимо инициализировать новый массив точек, размер которого будет равен количеству углов многоугольника, заранее заданного на ползунке пользователем. Далее зная количество углов при помощи функции массив заполняется точками.

Чтобы заполнить массив точками необходимо знать координаты середины описанной окружности многоугольника. Для этого я просто вычисляю центр фигуры по формуле, аналогично . Таким нехитрым способом я получаю координаты середины многоугольника. Далее вычисляется следующая точка следующим образом:

1. Вычисляется угол между вершинами равный , но так как расчет точек происходит в цикле, то данное значение домножается на текущую итерацию цикла;
2. Вершины многоугольника равноудалены от центра фигуры, поэтому радиус можно посчитать, проведя перпендикуляр к нижней границе фигуры, т.е. расстояние равное половине высоты фигуры, а это ;
3. Вычисляются координаты вершин по каждой из оси. Для OX: , OY: , где координаты центра, радиус, текущая итерация цикла.

Исходя из следующего алгоритма получается следующая блок-схема:



**Рис. 2.3.5 Блок-схема алгоритма Расчета точек многоугольника**

Таким образом мы рассмотрели все алгоритмы реализации программного продукта и теперь можно приступить к их программной реализации.

# **Инструменты реализации**

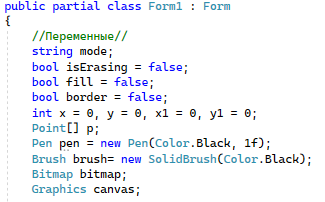
В качестве языка программирования для разработки программного продукта был выбран С# с использованием технологии Windows Forms (WinForms)

С# является одним из самых распространенных языков программирования и прост в освоение. Также стоит отметить, что язык является объектно-ориентированным, а это позволяет в свою очередь разрабатывать более сложные программы, как графический редактор.

Технология WinForms позволяет удобно разрабатывать графический интерфейс приложения, принимать и обрабатывать события, происходящие в приложении, что, собственно, необходимо в рамках данной учебной практики.

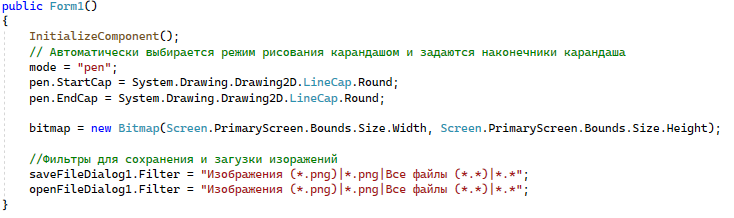
# **Разработка программного продукта**

Разработку программного продукта необходимо начать с объявления переменных.



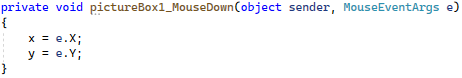
**Рис. 4.1 Объявление переменных**

Переменная mode – показывает какой режим рисования используется; isErasing – служит для обозначения режима «Ластик»; fill и border – указывают на наличие или отсутствие заливки и обводки; x,y,x1,y1 – координаты начальной и конечной точек; p – массив точек для отрисовки многоугольника; pen – карандаш для рисования с начальными параметрами, цвет – черный, толщина 1 пиксель; brush – кисть для заливки, в данном случае ее изначальный цвет черный; bitmap – изображение, с которым происходит работа; canvas – область рисования.



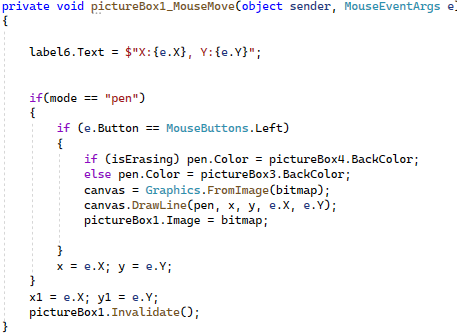
**Рис. 4.2 Конструктор формы**

Методом InitializeComponeent() – инициализируются все компоненты формы. Далее устанавливается режим рисования карандашом и наконечники карандаша. После инициализируется bitmap в соответствии с размерами окна пользователя. И также устанавливаются фильтры для диалоговых окон сохранения и загрузки.



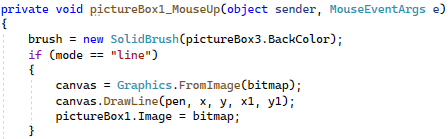
**Рис. 4.3 Метод MouseDown**

Здесь при нажатии мышки, текущие координаты мыши запоминаются, как начальные.



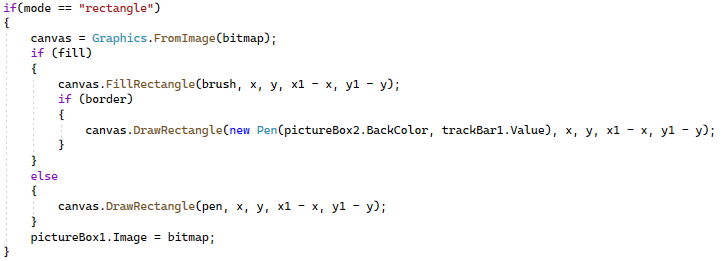
**Рис. 4.4 Метод MouseMove**

В label6 записываются текущие координаты мыши, которые после отображаются в нижнем левом углу. Далее идет условие на проверку режима «Карандаш», если режим соответствует, то проверяется была ли нажата ЛКМ, после проверяется включен ли режим ластика. В случае если включен, то цвет карандаша меняется на текущей цвет ластика, находящийся в pictureBox4, в ином случае цвет карандаша становится текущим цветом карандаша, выбранный в pictureBox3. В область рисование помещается ранее созданный bitmap и уже на нем через метод DrawLine() рисуется линия. После для отображения пользователю результатов его действий, в основной pictureBox1 вставляется нарисованное изображение. Если ЛКМ не нажата, то просто обновляются начальные координаты, а если режим «Карандаш» и вовсе неактивен, то обновляются конечные координаты. После вызывается метод Invalidate() для перерисовки.



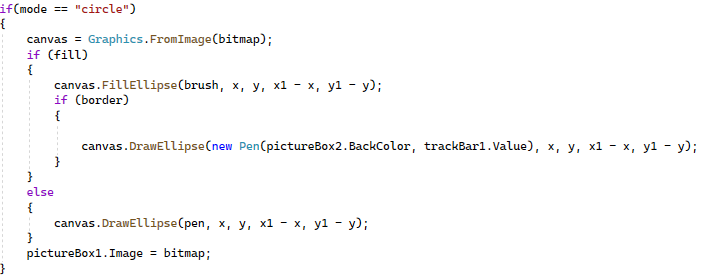
**Рис. 4.5 Метод MouseUp для режима «Линия»**

Инициализируется кисть для заливки последующих фигур. Цвет заливки берется из текущего цвета рисования в pictureBox3. Проверяется режим «Линия». Если режим совпадает, то в область рисования вставляется изображение. Далее на нем рисуется линия по начальным и конечным координатам точек и вставляется в pictureBox1.



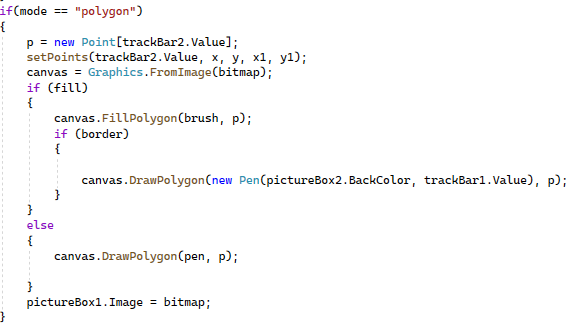
**Рис. 4.6 Метод MouseUp для режима «Прямоугольник»**

Проверяется условие на режим «Прямоугольник». Если режим совпадает, то в область рисования помещается изображение. После проверяется наличие заливки, если утверждение истинно, рисуется залитый прямоугольник. После проверяется наличие обводки, если утверждение истинно, то поверх залитого прямоугольника рисуется пустой прямоугольник в качестве обводки, цвет и толщина которого определяется цветом обводки в pictureBox2 и значением ползунка trackBar1. Если заливка не активна, то просто рисуется пустой прямоугольник. После в pictureBox1 переносится нарисованное изображение.



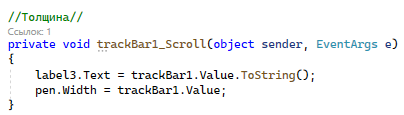
**Рис. 4.7 Метод MouseUp для режима «Эллипс»**

Аналогично для режима «Эллипс».



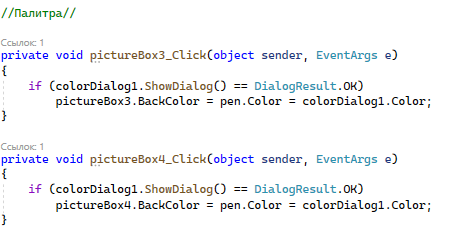
**Рис. 4.8 Метод MouseUp для режима «Многоугольник»**

Сначала инициализируется массив точек p с размером количество углов многоугольника, определяемым ползунком trackBar2. Далее вызывается метод установки точек. Далее действия аналогичные с предыдущими режимами.



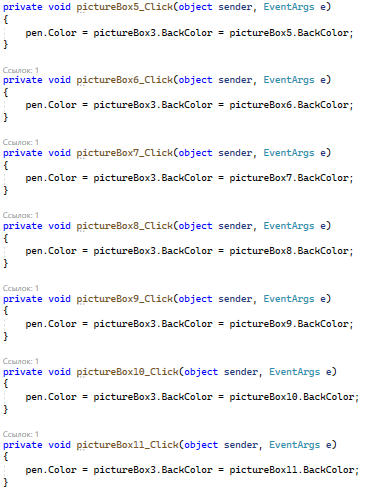
**Рис. 4.9 Настройка толщины карандаша**

При передвижении ползунка в сторону в label3 записывается значение самого ползунка trackBar1 и выводится под ним, также вместе с этим толщина карандаша устанавливается по текущему значению ползунка.



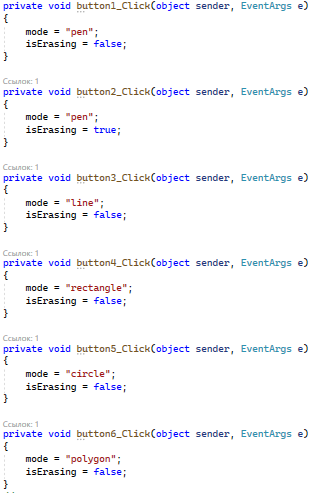
**Рис. 4.10 Настройка текущих цветов**

При помощи colorDialog1 вызывается окно диалога выбора цвета и если результат диалога является нажатая кнопка ОК, то текущий цвет меняется на выбранный в pictureBox3 или pictureBox4 в зависимости, где был вызван диалог. На этот же цвет меняется цвет рисования карандаша.



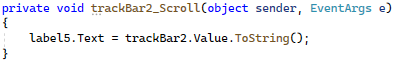
**Рис. 4.11 Выбор цвета из палитры**

Для каждого pictureBox палитры при нажатии меняет текущий цвет рисования на кликнутый цвет pictureBox.



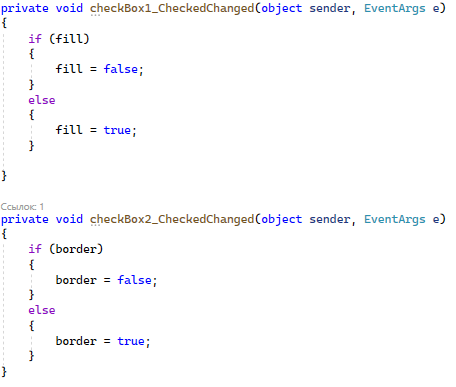
**Рис. 4.12 Описание нажатие кнопок инструментов**

При нажатии каждой из кнопок, режим меняется на соответствующий, и если это не режим «Ластик», то после нажатия кнопки режим «Ластик» отключается.



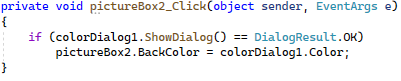
**Рис. 4.13 Ползунок выбора количества углов многоугольника**

Изменяет значение label5 на значение ползунка trackBar2, показывая его значение в числовом формате.



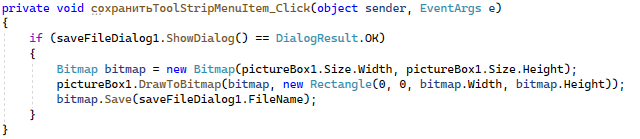
**Рис. 4.14 Чекбоксы заливки и обводки**

При изменении значения checkBox проверяется: если чекбокс уже отмечен, то значение переменный становится ложным, в ином случае истинным.



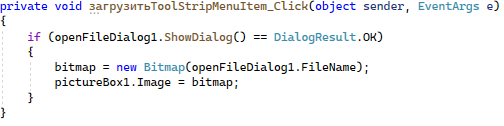
**Рис. 4.15 Выбор цвета обводки**

При нажатии по pictureBox2 вызывается colorDialog1. После выбора цвета и нажатия ОК. Цвет обводки pictureBox2 меняется на выбранный пользователем цвет.



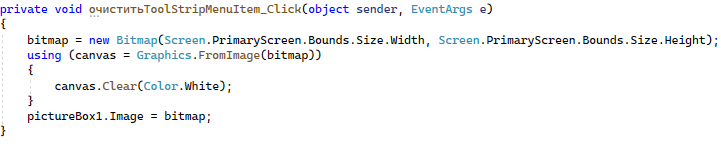
**Рис. 4.16 Кнопка меню «Сохранить»**

Вызывается диалог сохранения файла saveFileDialog1. При результате диалога ОК создается новое изображение bitmap по размерам основного pictureBox1. Далее с pictureBox1 переносится изображение в новый bitmap и сохраняется с именем указанным в saveFileDialog1.



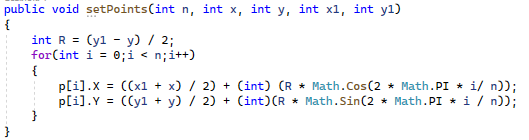
**Рис. 4.17 Кнопка меню «Загрузить»**

Вызывается диалог загрузки openFileDialog1. При результате диалога ОК создается новое изображение bitmap выбранное в диалоге и приравнивается к изображению на основном pictureBox1.



**Рис. 4.18 Кнопка меню «Очистить»**

Создается новое изображение bitmap в соответствии с размерами экрана пользователя. После используя область рисования, в которую занесено ранее созданное изображение, очищаем ее используя белый цвет и передает изображение в pictureBox1.



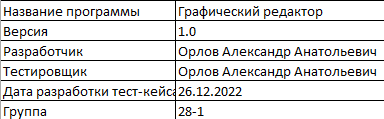
**Рис. 4.19 Функция расчёта координат точек многоугольника**

В качестве аргументов функция берет значение количества углов многоугольника – n и координаты начала и конца рисования. Внутри функции локально вычисляется радиус описанной окружности – R. Далее в цикле с счетчиком от 0 до количества углов в массив точек вычисляются и заносятся их значения.

# **Тестирование программного продукта**

Важны этапом в ходе разработки программного продукта является тестирование. В процессе тестирования был разработан тест-кейс.

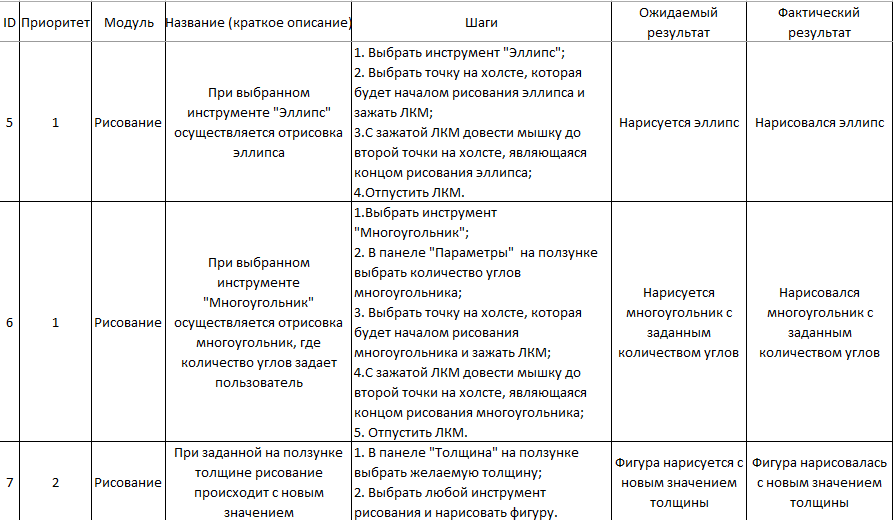
**Титульный лист**

****

**Рис 5.1 Титульный лист**

****

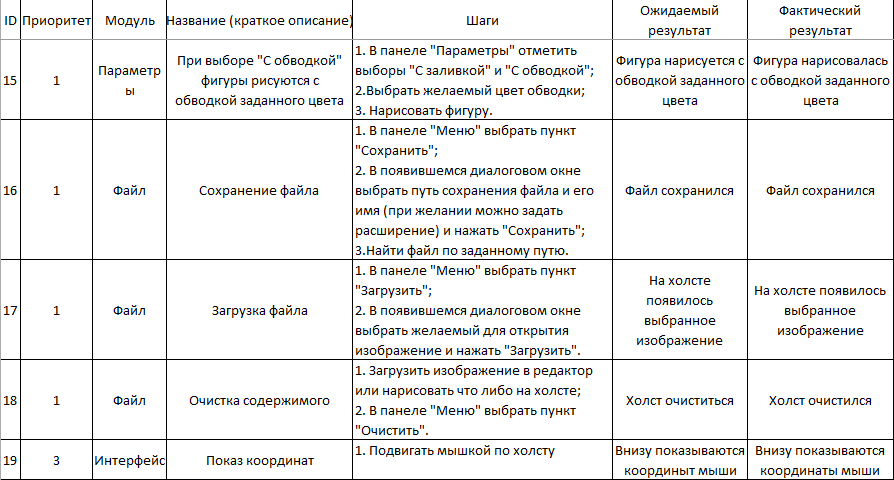
**Рисунок 5.1 Тест-кейс проверки графического редактора**

****

**Рисунок 5.2 Продолжение рис.5.1**

****

**Рисунок 5.3 Продолжение рис.5.1**

****

**Рисунок 5.4 Продолжение рис.5.1**

# **Выгрузка локального репозитория на удаленный сервер**

После написания кода программного продукта, мы выгружаем его на удаленный сервер GitHub.

GitHub — это служба размещения в Интернете репозиториев Git, которые используются для хранения содержимого docs.microsoft.com. В GitHub размещается основной репозиторий всех проектов.

Моя страница на GitHub’e: https://github.com/AlexEaglewhite6

# **Руководство пользователя**

Чтобы помочь людям разобраться в своем программном продукте, необходимо составить руководство пользователя.

Данная программа предназначена для рисования в ней при помощи мыши, но, помимо этого, есть и другие инструменты рисования и также их параметры. Также пользователь может сохранить свои рисунки или загрузить в редактор. Рассмотрим все поподробнее.

Панель «Инструменты»

* Режим «Карандаш» – рисует выбранным Вами цветом беспрерывную линию. Для рисования просто зажмите ЛКМ и введите мышь по холсту;
* Режим «Ластик» – стирает рисунок в области мыши в текущий цвет ластика. Для стирания зажмите ЛКМ и проведите мышью там, где необходимо стереть лишнее;
* Режим «Линия» – рисует линию по двум заданным точкам. Чтобы нарисовать линию выберете точку откуда Вы хотите ее начертить и с зажатой ЛКМ доведите мышь до того места, где Вы хотите закончить линию. После отпустите кнопку мыши;
* Режим «Прямоугольник» – рисует прямоугольник по двум заданным точкам. Чтобы нарисовать прямоугольник выберете точку откуда Вы хотите его начертить (верхняя левая точка) и с зажатой ЛКМ доведите мышь до того места, где Вы хотите закончить прямоугольник (нижняя правая точка). После отпустите кнопку мыши;
* Режим «Эллипс» – рисует эллипс по двум заданным точкам. Чтобы нарисовать эллипс выберете точку откуда Вы хотите его начертить и с зажатой ЛКМ доведите мышь до того места, где Вы хотите закончить эллипс. После отпустите кнопку мыши;
* Режим «Многоугольник» – рисует многоугольник по двум заданным точкам. Чтобы нарисовать многоугольник выберете точку откуда Вы хотите его начертить (верхняя левая точка) и с зажатой ЛКМ доведите мышь до того места, где Вы хотите закончить прямоугольник (нижняя правая точка). После отпустите кнопку мыши.

Панель «Параметры»

* «С заливкой» – означает создание будущих фигур полностью залитыми текущим цветом. Чтобы активировать данный параметр нажмите на квадратик;
* «С обводкой» – означает создание у залитой фигуры обводку в соответствии с цветом обводки, указанным в этой же панели. Чтобы активировать данный параметр нажмите на квадратик;
* «Цвет обводки» - означает какой цвет будет у будущей обводки. Чтобы выбрать цвет, нажмите по квадрату и в появившемся окне выберите цвет, после нажмите «ОК»;
* «Кол-во углов многоугольника» - означает какое количество углов будет у будущего многоугольника. Всего доступно значений от 3 до 10. Чтобы выбрать значение двигайте ползунок вправо или влево;
* «Толщина» – означает с какой толщиной Вы будете рисовать. Всего доступно значений от 1 до 25. Чтобы выбрать значение двигайте ползунок вправо или влево;

Панель «Палитра»

Первые два квадрата служат для обозначения текущего цвета рисования и ластика. Также рядом представлены другие цвета. Их Вы можете выбрать если захотите изменить цвет рисования. Но если Вы хотите другой цвет, которого нет на палитре, то кликните по квадрату текущего цвета и в появившемся диалоговом окне выберете понравившейся цвет. После нажмите «ОК».

Меню «Файл»

* «Сохранить» – позволяет сохранить рисунок. Для этого выберите пункт «Сохранить» и нажмите на него. В появившемся окне выберете путь сохранения изображение и его имя;
* «Загрузить» – позволяет загрузить изображение в редактор. Для этого выберете пункт «Загрузить» и нажмите на него. В появившемся окне выберете путь откуда загружать изображение и его имя;
* «Очистка» - позволяет пользователю очистить холст для рисования. Для этого выберете пункт «Очистить» и нажмите на него.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения учебной практики на тему «Устройство и функционирование информационных систем», «Графический редактор» были получены следующие результаты:

Дано описание предметной области

- рисование линий (по двум точкам)

- рисование многоугольников (кол-во углов задает пользователь)

- выбор цвета обводки

- возможность добавить заливку и ее цвет

- функцию сохранения и загрузки изображения.

Построена структура механизма графического редактора и диаграмма последовательности работы приложения;

Разработана структура приложения, обеспечивающая обработку данных от пользователя.

В ходе разработки графического редактора были определены средства, применяемые при разработке:

- Диаграммы;

-Windows forms, графический редактор с добавлением изменений от пользователя;

- Последовательности;

-Блок-схема.

Недостатки Windows forms:

-Неудобство при использовании на других устройствах;

- Процесс разработки требует точных и неизменяемых условий.

Путь улучшения графического редактора:

-Соблюдение единого Code Style;

-Добавление баз данных для лучшей обработки новых объектов на Windows Forms.