МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра комп’ютерних наук

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: “Об’єктно-орієнтоване програмування”

на тему:

**“ Графічний редактор”**

студента І курсу групи КН-19-2

спеціальності 122 “Комп’ютерні науки”

Копилова Владислава Руслановича

(прізвище, ім’я та по-батькові)

Керівник: ст. викладач кафедри ІПЗ Чижмотря О.В.

Дата захисту: “\_\_\_” 2020р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Марчук

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський

(підпис) (прізвище та ініціали)

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра комп’ютерних наук

Освітній рівень: бакалавр

Спеціальність 122 «Комп’ютерні науки»

“ЗАТВЕРДЖУЮ”

Зав. кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ І.В. Пулеко

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 20\_\_р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Копилову Владиславу Руслановичу

1. Тема роботи: Розробка графічного редактору,

керівник роботи: ст. викладач кафедри ІПЗ Чижмотря О.В.

2. Строк подання студентом: “09” червня 2020р.

3. Вихідні дані до роботи: Розробити графічний редактор.

4. Зміст розрахунково-пояснювальної записки(перелік питань, які підлягають розробці)

* + - 1. Аналіз технічного завдання
      2. Пошук та дослідження аналогічних розробок
      3. Алгоритми роботи майбутньої програми

4. Опис роботи програми

5. Програмне дослідження

5. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КП.

2. Посилання на репозиторій: <https://gitlab.com/Tamskray/graphics-editor.git>

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Чижмотря О.В., ст. викладач каф. ІПЗ |  |  |

1. Дата видачі завдання “17” лютого 2020 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 20.05.20 -22.05.20 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 22.05.20 -24.05.20 |  |
| 3 | Формулювання технічного завдання | 24.05.20 – 25.05.20 |  |
| 4 | Опрацювання літературних джерел | 25.05.20 – 02.06.20 |  |
| 5 | Проектування структури | 02.06.20 – 03.06.20 |  |
| 6 | Написання програмного коду | 03.06.20 – 06.06.20 |  |
| 7 | Налагодження | 06.06.20 – 08.06.20 |  |
| 8 | Написання пояснювальної записки | 08.06.20 –  09.06.20 |  |
| 9 | Захист | .06.20 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Р. Копилов

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему “Графічний редактор” складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на **52** сторінках друкованого тексту.

Пояснювальна записка має **51** сторінки додатків. Список використаних джерел містить **10** найменувань і займає **1** сторінку. В роботі наведено **36** рисунків. Загальний обсяг роботи – **70** сторінок.

У першому розділі було обґрунтовано створення програми для малювання, редагування, збереження зображень.

У другому розділі проведено проектування і розробка програмного продукту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи створення програми для малювання, редагування, збереження зображень.

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: ООП, ІНСТРУМЕНТИ, РЕДАГУВАННЯ, ЗБЕРЕЖЕННЯ.

**ЗМІСТ**

[**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ** 6](#_Toc42604284)

[**ВСТУП** 7](#_Toc42604285)

[**РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ** 11](#_Toc42604286)

[1.1 Аналіз задачі, засобів та методів її вирішення 11](#_Toc42604287)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи. 12](#_Toc42604288)

[1.3 Технічне завдання на курсову роботу 16](#_Toc42604289)

[Висновки до першого розділу 17](#_Toc42604290)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 18](#_Toc42604291)

[2.1 Проектування загального алгоритму роботи програми 18](#_Toc42604292)

[2.2 Розробка функціональних алгоритмів роботи програми 19](#_Toc42604293)

[2.3 Розробка програмного забезпечення 19](#_Toc42604294)

[Висновки до другого розділу 36](#_Toc42604295)

[**РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ** 37](#_Toc42604296)

[3.1 Опис роботи додатку та його тестування 37](#_Toc42604297)

[Висновки до третього розділу 50](#_Toc42604298)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 51](#_Toc42604299)

[**ДОДАТОК** 52](#_Toc42604300)

# **ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

VS – Visual Studio

ПЗ – Програмне забезпечення

# **ВСТУП**

У цій курсовій роботі буде наведено процес створення графічного редактору.

Графічний редактор - прикладна програма (або пакет програм), що дозволяє її користувачеві створювати і редагувати зображення на екрані комп'ютера і зберігати їх в графічних форматах файлів, наприклад, JPEG, PNG, GIF, TIFF.

Деякі графічні редактори призначені для редагування фотографій, тоді як інші — переважно для створення та редагування малюнків.

Існує три типи графічних редакторів:

1) Растрові графічні редактори;

2) Векторні графічні редактори;

3) Гібридні графічні редактори.

1) Растровий графічний редактор — спеціалізована програма для створення і обробки растрових зображень. Ці програмні продукти знайшли широке застосування в роботі художників-ілюстраторів, при підготовці зображень до друку або на фотопапері, публікації в інтернеті.

Растрові графічні редактори дозволяють користувачеві створювати і редагувати зображення на екрані комп'ютера (серед звичних інструментів — декілька типів ліній, стирання, копіювання об'єктів, додавання тексту, заповнення кольору фону…), а також зберігати їх в різних растрових форматах. Формати збереження зображень поділяються на такі, що дозволяють зберігати растрову графіку з незначним зниженням якості за рахунок використання алгоритмів стиснення з втратами (JPEG, PNG, GIF і TIFF), та такі, що також підтримують стиснення (RLE), але загалом є «попіксельним» описом зображення (BMP).

На противагу векторним редакторам, растрові використовують для утворення зображень матрицю точок (bitmap). При цьому, більшість сучасних растрових редакторів містять векторні інструменти редагування як допоміжні.

Щоб охарактеризувати растрові графічні редактори, можна порівняти їх з звичайним малюванням, тобто з малюванням олівцем на папері. Основна відмінність між малюнками на папері і на екрані монітора полягає в тому, що на моніторі будь-яке зображення дискретно, тобто воно складається з кінцевого числа прямокутних точок, кожна з яких може бути тільки одного кольору з фіксованою палітри. У малюнка на папері таку мінімальну одиницю, як точка, знайти не можна. На папері неможливо намалювати два абсолютно однакових малюнка. За допомогою графічного редактора під можна створити саме такий дискретний малюнок, що складається з фіксованого числа точок, після чого можна змінювати колір кожної точки окремо. Саме тому будь-який редактор такого типу перед створенням малюнка просить вказати його точні розміри і іноді палітру кольорів. Такі редактори дозволяють будувати лінії і графічні примітиви, заливати області певним кольором, забивати текст, малювати різними інструментами - олівцями, пензлями, розпилювачами. Залежно від обраного інструменту будуються лінії з різними властивостями - напівпрозорі, з розмитими краями, заповнені текстурою. Завжди є можливість збільшувати масштаб, для того щоб можна було працювати з окремими точками.

2) Векторні графічні редактори дозволяють користувачеві створювати і редагувати векторні зображення безпосередньо на екрані комп'ютера, а також зберігати їх в різних векторних форматах, наприклад, CDR, AI, EPS, WMF або SVG.

Перевага векторної графіки полягає в тому, що форму, колір і просторове положення складових її об'єктів можна описувати за допомогою математичних формул, а не описувати кожен піксель окремо.

Важливим об'єктом векторної графіки є сплайн. Сплайн - це крива, за допомогою якої описується та чи інша геометрична фігура. На сплайнах побудовані сучасні шрифти TryeType і PostScript.

У векторної графіки багато достоїнств. Вона економна в плані дискового простору, необхідного для зберігання зображень: це пов'язано з тим, що зберігається не саме зображення, а тільки деякі основні дані, використовуючи які, програма щоразу відтворює зображення заново. Крім того, опис колірних характеристик майже не збільшує розмір файлу.

Об'єкти векторної графіки легко трансформуються і модифікуються, що не має практично ніякого впливу на якість зображення. Масштабування, поворот, викривлення можуть бути зведені до парі-трійці елементарних перетворень над векторами.

У тих областях графіки, де важливе значення має збереження ясних і чітких контурів, наприклад, в шрифтових композиціях, у створенні логотипів і інше, векторні програми незамінні. Векторна графіка може включати в себе і фрагменти растрової графіки: фрагмент стає таким же об'єктом, як і всі інші (правда, зі значними обмеженнями в обробці).

Векторна графіка описує зображення з використанням прямих і вигнутих ліній, які називаються векторами, а також параметрів, що описують кольору та розташування. Наприклад, зображення деревного листа описується точками, через які проходить лінія, створюючи тим самим контур листа. Колір листа задається кольором контура та області всередині цього контуру. У растровому редакторі лист дерева описується конкретним розташуванням і кольором кожної точки сітки.

При редагуванні елементів векторної графіки Ви змінюєте параметри прямих і вигнутих ліній, що описують форму цих елементів. Ви можете переносити елементи, міняти їх розмір, форму і колір, але це не відіб'ється на якості і1х візуального представлення. Векторна графіка не залежить від дозволу, тобто може бути показана в різноманітних вихідних пристроях з різним дозволом без втрати якості.

3) Гібридний графічний редактор — растрово-векторний редактор, програма, забезпечує інтерактивну роботу з векторною і растровою графікою одночасно за допомогою набору спільних інструментів для обох форм представлення графіки, з можливістю їх взаємного перетворення.

- дозволяє працювати з растровою і векторною графікою одночасно і перетворювати одну в іншу.

- обробкою електронних креслень, отриманих за допомогою сканування, є програма, яка дозволяє швидко вносити зміни безпосередньо в растрову форму подання вихідних креслень

Векторизація і растеризация — специфічні парні функції гібридного редактора, що забезпечують двонаправлене перетворення графічних об'єктів. Крім того, такі функції здійснюються окремо за допомогою програм-векторизатор (векторизація) і в растрових редакторах при імпорті векторних файлів (растеризация).

Реалізація процесу перетворення растрової графіки у векторну в гібридному редакторі може мати деякі особливості. Так, вона може відбуватися не відразу, а поступово, під час усього життєвого циклу відсканованого креслення, при внесенні до нього змін і додавань у векторній формі. Поступово частка векторної графіки буде зростати, поки весь креслення не перетвориться на векторний.

Актуальність роботи полягає в тому, що графічний редактор є інструментом для роботи з зображеннями, що дозволяє користувачам практично будь-якого рівня підготовки вирішувати поставлені перед ними завдання. Відредагувати зображення, створити свій малюнок і тд.

Мета роботи полягає в розробці програмного продукту - розробка програмного комплексу з використанням об'єктно-орієнтованого підходу та закріплення знань з програмування на мові C# і набуття досвіду створення програмного додатка.

Об'єкт дослідження курсової роботи - процес малювання, обробки, редагування ,зберігання та відображення зображень та створених малюнків.

Предмет дослідження – програма для малювання, обробки, зберігання та редагування зображень.

# **РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ**

## Аналіз задачі, засобів та методів її вирішення

Задача полягає в тому, що потрібно:

1. Визначити основні функції програмного додатку.
2. Визначити функціонал програмного продукти, засоби роботи.
3. Створити простий інтерфейс програми для комфортного користування.

В процесі створення графічного редактору брався будь-який аспект проекту, розроблялася його найлегша версія, а після того вдосконалювалася, чи повністю перероблювалася.

На основі поставлених цілей і вимог до створюваного продукту найкращим середовищем розробки графічного редактору буде Visual Studio, розробка буде виконуватись за допомогою додатку WinForms, адже він має широкі можливості в створенні функціоналу програми.

Програма включає в себе декілька шляхів користування:

1) Малювання різними інструментами на холсті, сворення різних геометричних фігур, створення тексту на холсті та його обробка, зміна фону, відкривання зображення та його малювання на ньому, збереження зображення

2) Обробка зображення різними фільтрами, редагування зображення та його збереження.

## Аналіз існуючого програмного забезпечення за тематикою курсової роботи.

Для аналізу існуючого програмного забезпечення за тематикою курсової роботи було вибрано деякі растрові графічні редактори, адже наш програмний додаток буде саме растровим графічним редактором і потрібно проаналізувати аналоги, щоб розуміти з чим маєш справу.

1) Один з найпопулярніших це, звичайно, Adobe Photoshop - графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень і найвідомішим продуктом фірми Adobe. Часто цю програму називають просто Photoshop (Фотошоп).



Рис.1.1.Логотип графічного редактору Фотошоп

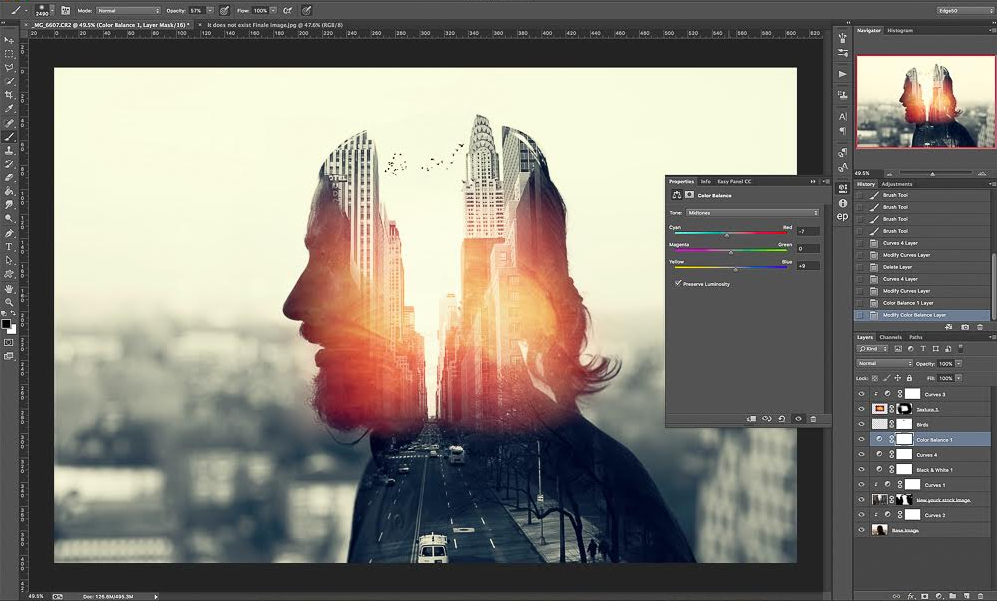


Рис.1.2. Вигляд інтерфейсу графічного редактору Фотошоп

Функції програми Фотошоп дуже різноманітні. Фотошоп є одночасно потужною фотолабораторією і засобом створення цифрових зображень. За допомогою набору основних інструментів, таких як Кисть (Paintbrush), Аерограф (Airbrush), Перо (Pen) і Олівець (Pencil), можна малювати і розфарбовувати зображення. Всі перераховані інструменти дають можливість настройки цілого ряду параметрів: ширини мазка, ступеня розмивання крайок, жорсткості і м'якості. При роботі з зображеннями Фотошоп дає можливість роботи з слоями. Можна створювати монтажі і колажі з декількох зображень, редагуючи кожне як окремий, незалежний шар.

Коротко про інтерфейс:

Панель управління забезпечує доступ до команд головного меню і управління інтерфейсом. Панель інструментів містить всі інструменти, які можна застосовувати для редагування зображення. Панель параметрів відображає, який інструмент обраний для роботи в даний момент. В область палітр представлені всі інструменти для маніпуляцій із зображенням. Тим, хто хоче орієнтуватися в програмі всерйоз, радимо взяти підручник і поступово вивчити, для чого потрібна та чи інша команда.

Мінуси графічного редактору:

-складність його освоєння для початківців

-високу вартість продукту.

-не підтримує ОС Linux.

Такі негативні моменти дуже відносні, так що не дивлячись на це Фотошоп залишається лідером на ринку графічних редакторів.

2) GIMP - растровий графічний редактор, із деякою підтримкою векторної графіки. Проєкт розпочали 1995 року Spencer Kimball і Peter Mattis як навчальний проєкт в Берклі. В 1997, після закінчення ними університету GIMP став частиною проєкту GNU. Програма підтримується та розвивається товариством добровольців, ліцензована за умовами GNU General Public License версії 3+, починаючи з релізу 2.8. Символом GIMP є койот Вілбер (Wilber). Програма працює на системах Microsoft Windows, Gnu/Linux, FreeBSD (або OpenBSD), MacOS X, OpenSolaris.



Рис.1.3.Логотип графічного редактору GIMP



Рис.1.4.Інтерфейс графічного редактору GIMP

Типові завдання, які можна вирішувати за допомогою GIMP, включають в себе створення графіки і логотипів, масштабування і кадрування фотографій, розфарбовування, комбінування зображень з використанням слоїв, ретушування і перетворення зображень в різні формати.

Традиційно GIMP вважається вільним аналогом ряду редакторів (найчастіше називається Adobe Photoshop), хоча самі розробники часто заперечують проти такого формулювання.

В GIMP присутній набір інструментів корекції: криві, рівні, мікшер каналів, постеризація, тон-насиченість, баланс кольорів, яскравість-контраст, знебарвлення.

За допомогою фільтрів, інструментів, масок і шарів з різними типами накладення можна: вирівнювати завалений горизонт, прибирати спотворення, що вносяться оптикою, коригувати перспективу, виконувати клонування об'єктів з урахуванням перспективи, кадрувати фотографії, видаляти дефекти на кшталт пилу на матриці (штамп, лікувальна кисть), імітувати використання різних цветофільтров, «витягувати» втрачену деталізацію в тінях і тд.

Малювання: кілька змальовують інструментів, вільно масштабовані кисті, підтримка кистьовий динаміки, підтримка графічних планшетів.

Мінуси:

В даний час використання стабільної версії GIMP в комерційному дизайні, поліграфії і фотографії пов'язане з рядом складнощів, а в багатьох випадках і зовсім неможливо:

• немає підтримки плашкових кольорів (і палітри Pantone);

• немає повноцінної підтримки колірних моделей, CIELAB і CIE XYZ;

• немає підтримки HDRi і операторів відображення тонів;

• немає процедурних (корегувальних) шарів і ефектів (стилів) шарів;

1.3 Технічне завдання на курсову роботу

**Підстава для розробки:**

Підставою для розробки курсової роботи є завдання на курсову роботу.

**Призначення розробки:**

Програма призначена для малювання, обробки, редагування, збереження зображень.

**Вимоги до програмного засобу:**

Графічний редактор повинен давати можливість користувачу малювати на холсті, створювати різні геометричні фігури, текст, обробка та збереження зображень.

Програма повинна виконувати наступні функції:

1. Давати можливість користувачу малювати різними іструментами, які мають різні стилі та налаштування.
2. Створювати текст, редагувати його та налаштовувати.
3. Створювати різні геометричні фігури.
4. Змінювати розмір холста для малювання.
5. Відкривання зображення та малювання на ньому, створювання фігур, тексту і тд.
6. Зміна фонового кольору.
7. Такі інструменти, як ластик та піпетка.
8. Редагування зображення на RGB каналах.
9. Редагування зображення різними готовими фільтрами.

Потрібно дотримуватись максимальної простоти і зручності в інтерфейсі, які будуть інтуїтивно зрозумілі для кожного користувача.

У разі виконання некоректних дій передбачено підказки та приклади, максимальне роз’яснення причини, через яку певна дія не вдалась чи вдалась некоректно.

**Вимоги до надійності:**

Передбачити ситуації при виконанні некоректних дій зі сторони користувача програмного продукту.

**Вимоги до складу і параметрів технічних засобів:**

Програма повинна працювати на комп’ютерах.

**Вимоги до інтерфейсу:**

Інтерфейс має бути простим, зручним та інтуїтивно зрозумілим для кожного користувача.

**Вимоги до інформаційної і програмної сумісності:**

Повинна бути встановлена Visual Studio 2013 і вище.

**Спеціальні вимоги**

Немає.

## Висновки до першого розділу

У ході виконання першого розділу була проаналізована основна задача, засіб її вирішення. Були проаналізовані аналогічні графічні редактори. Були визначені основні технічні завдання, яких необхідно дотримуватись при створенні програмного продукту, для опимальної та зручної роботи, та реалізаціх обширного функціоналу графічного редактору.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## 2.1 Проектування загального алгоритму роботи програми

На даному етапі розробки розглянемо загальний алгоритм роботи графічного редактору.

При відкритті програми користувач одразу бачить її інтерфейс. На початку роботи у користувача буде вибрано інструмент “none”, що не дозволить малювати на холсті. Користувач може вибирати різні інструменти для малювання, створення геометричних фігур , тексту, змінювати іх товщину та стилі, вибір кольору, та очистка холста для малювання.

Також користувача зустрічає меню, в якому він може створити новий файл, відкрити зображення, зберегти його, вийти с програми.

У меню є пункт “Filtres”, після натискання якого, користувач переходить на нову форму, де він може відкривати та зберігати зображення. Основне призначення цієї форми, це редагування відкритого зображення за допомогою фільтрів та редагування на RGB каналах.



Рис.2.1. Блок-схема програми.

## 2.2 Розробка функціональних алгоритмів роботи програми

Алгоритми роботи програми:

1) Перша форма: вибір лише одного інструменту для малювання, стоверння фігур, тексту і тд. Відкриття та збереження зображень.

2) Друга форма: відкриття та збереження зображень, основні фільтри для редагування, редагування зображення на RGB каналах.

## 2.3 Розробка програмного забезпечення

***Розробка інтерфейсу графічного редактора:***

Для розробки інтерфейсу було використано Visual Studio 2019 та створено додаток WinForms. Оскільки він дозволяє реалізувати візуальні можливості зрозумілі для користувача.

Спочатку, спираючись на вимоги до продукту, створимо зручний інтерфейс програми, додамо іконки, щоб було зручно і зрозуміло на інтуїтивному рівні

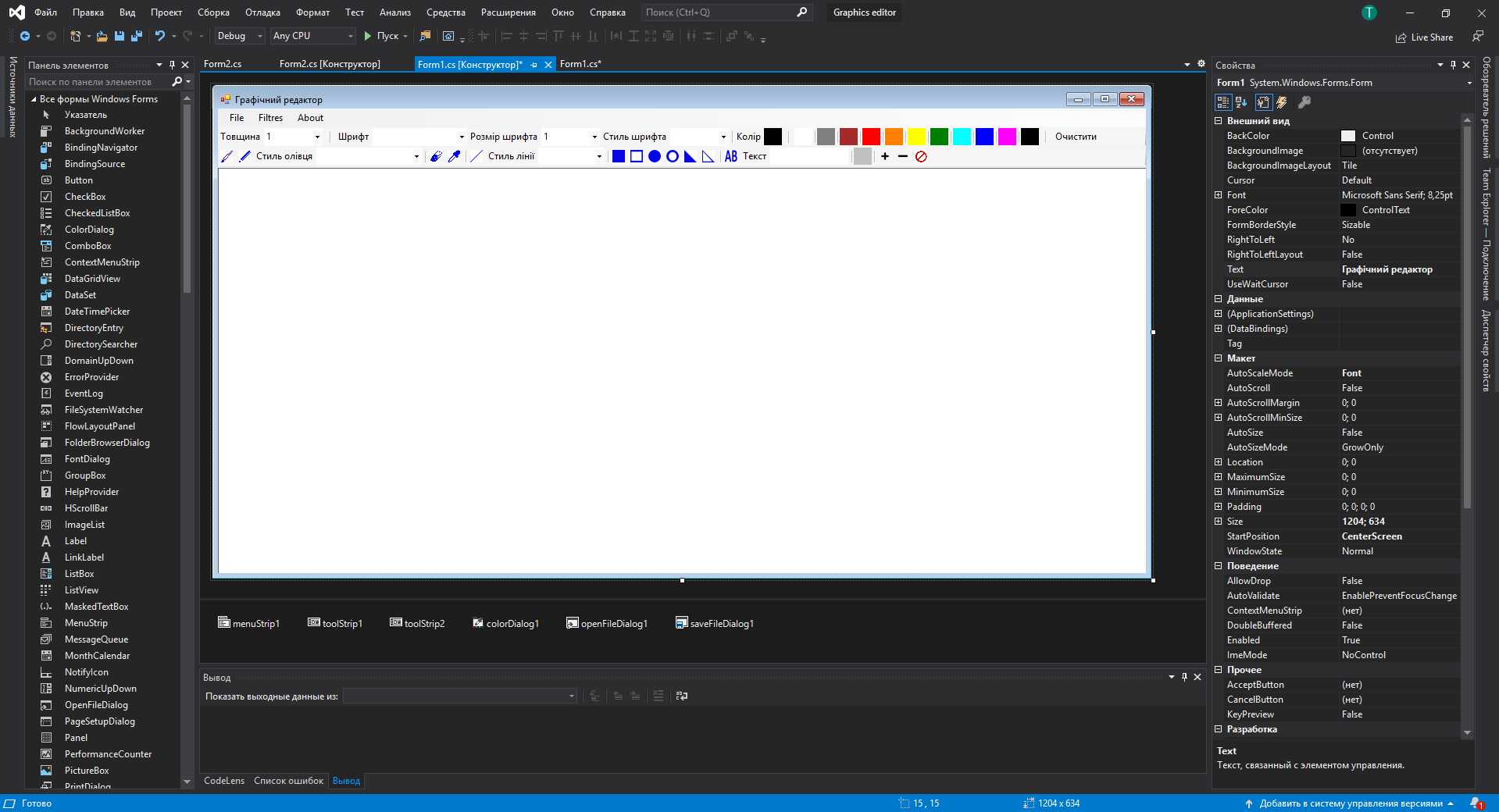


Рис.2.2.Інтерфейс програми

Програма складається з основного модуля Form1, що здійснює синхронізацію процесу обробки даних і взаємодії конкретних екземплярів конкретних об'єктів.

На даній формі розташуємо такі компоненти:

1) pictureBox1 - полотно, на якому буде основне зображення.

2) menuStrip1 – меню з трьома пунктами

Перший пункт «File», він має такі підпункти, як

«New» - очищює полотно;

«Open» - відкриває зображення;

«Save» - збережує зображення;

«Exit» - вихід із програми.

Другий пукнт «Filtres», при натисканні переходить на другу форму.

Третій пункт «About» про автора програмного продукту.

3) toolStrip1 – на ньому знаходяться деякі інструменти для роботи:

- Напис «Товщина» і список, що випадає, який регулює товщину кисті;

- Напис «Шрифт» і список, що випадає з назвами шрифтів;

- Напис «Розмір шрифта» і список, який регулює розмір шрифта.

- Напис «Стиль шрифта» і список, що випадає зі стилями шрифта.

- Напис «Колір кисті», кнопка налаштування кольору та відображення поточного кольору і кнопки швидкої зміни на зазначений колір;

- Кнопка очищення полотна.

4) toolStrip2 - меню інструментів. У ньому знаходяться:

- Кисть (brush);

- Олівець(pensil);

- Напис «Стиль олівця» і спсок, у якомі знаходяться усі додані стилі

- Ластик (eraser);

- Піпетка(pip);

- Лінія (line);

- Написа «Стиль лінії» і список, у якому знаходяться усі додані стилі лінії

- Прямокутник (rectangle);

- Форма прямокутника(spacerectangle);

- Еліпс (ellipse);

- Форма еліпса(spaceellipse);

- Трикутник(triangle);

- Форма трикутника(triangle1);

- Текст(text);

- Напис «Текст» і textBox, у якому потрібно писати заданий текст

- Кнопка зміни кольору фону полотна(ButtonBackgroundColor);

- Кнопка, яка збільшує розмір полотна(plusSize);

- Кнопка, яка зменшує розмір полотна(minusSize);

- None (none), активна при запуску програми.

Інтерфейс другої форми:

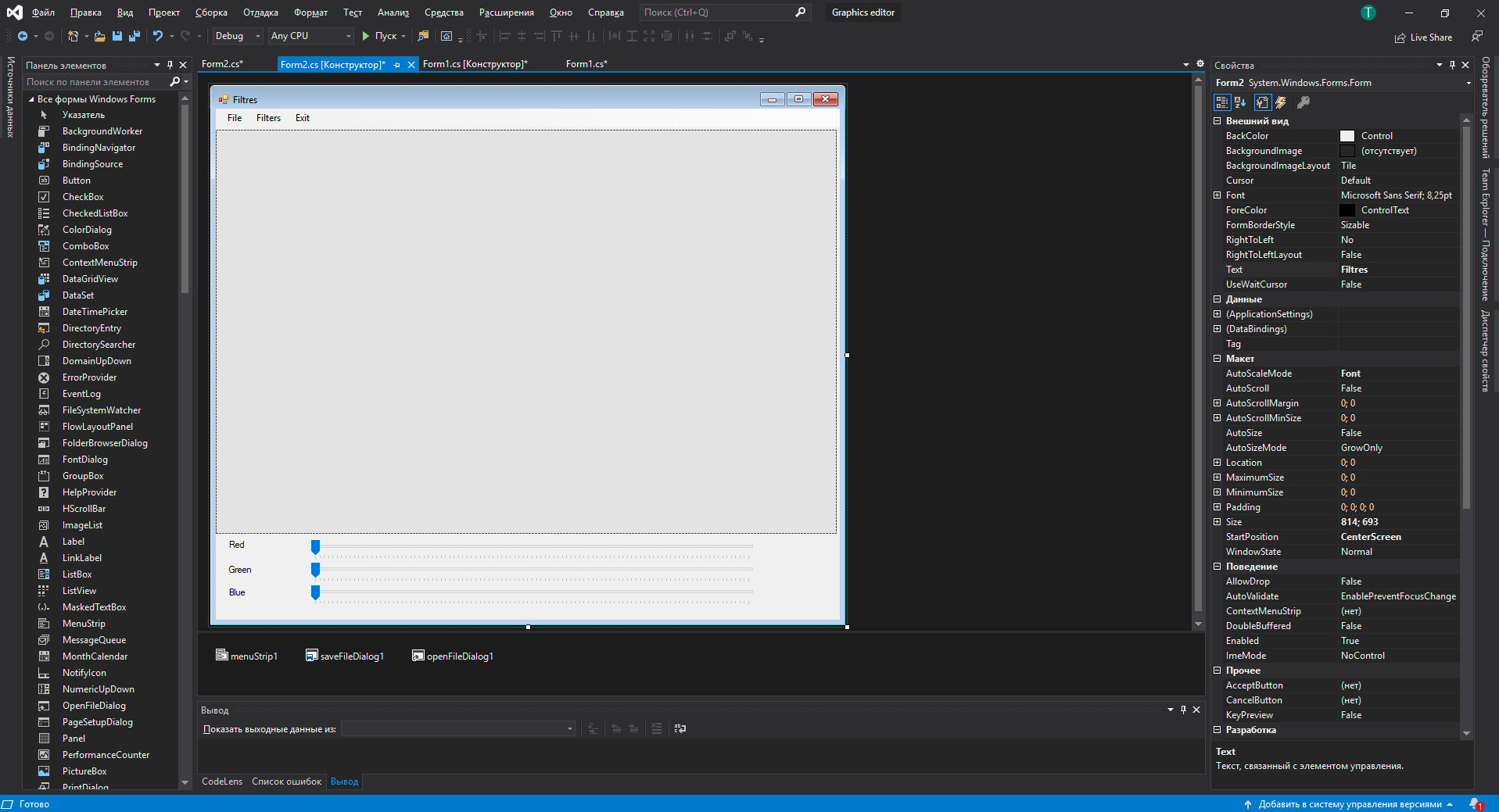


Рис.2.3.Інтерфейс другої форми

На даній формі розташуємо такі компоненти:

1) pictureBox1 - полотно, на якому відкрите зображення.

2) menuStrip1 – меню з трьома пунктами

Перший пункт «File», він має такі підпункти, як

«Open» - відкриває зображення;

«Save» - збережує зображення;

Другий пукнт «Filtres», він має такі підпункти, які редагують зображення(None, Sepia, Artistic, Gray, Spike, Flash, Frozen, Suju, Dramatic, Kakao).

Третій пункт «Exit» - вихід з програми.

3) Напис «Red» та trackBar для редагування доданого зображення на червоному каналі.

4) Напис «Green» та trackBar для редагування доданого зображення на червоному каналі.

5) Напис «Blue» та trackBar для редагування доданого зображення на червоному каналі.

***Глобальні змінні:***

Для того щоб програма малювала тільки при натисканні миші, треба завести змінну, яка буде відслідковувати це натискання:

bool isPressed;

Також знадобиться відстежувати початкові і кінцеві координати миші, для малювання простих фігур (квадрат, прямокутник, коло, еліпс, лінія, трикутник), для цього відповідно заведемо 4 змінні:

int x1, y1, x2, y2;

Щоб при малюванні до віджимання кнопки миші фігура змінювала свій розмір в залежності від позиції миші, треба звернутися до Bitmap:

public Bitmap snapshot, tempDraw;

Також треба регулювати колір, товщину кисті та розмір тексту:

Color foreColor;

int lineWidth = 2;

int FontSize = 2;

Вибирати інструмент ми будемо через рядок selectedTool та одразу вкажемо вибраний інструмент:

string selectedTool = "none";

Щоб можна було малювати, потрібно створити основну кисть:

Pen pen;

***Ініціалізація компонентів:***

Розмір Bitmap дорівнює розміру PictureBox1; Визначаємо колір кисті і створюємо її з lineWidth; Задаємо початкові значення при щапуску програми до деяких comboBox.

InitializeComponent();

snapshot = new Bitmap(pictureBox1.ClientRectangle.Width,pictureBox1.ClientRectangle.Height);

tempDraw = (Bitmap)snapshot.Clone();

foreColor = Color.Black;

pen = new Pen(foreColor, lineWidth);

none.Checked = true;

sizeCB.SelectedIndex = 9;

choseLine.SelectedIndex = 0;

pensilStyle.SelectedIndex = 0;

fontSize.SelectedIndex = 13;

fontStyle.SelectedIndex = 0;

Заповнюємо comboBox назвами різних шрифтів;

Змінюємо властивість AlowDrop на true.

private void Form1\_Load(object sender, EventArgs e)

{

FontFamily[] family = FontFamily.Families;

foreach (FontFamily font in family)

{

fontName.Items.Add(font.GetName(1).ToString());

}

pictureBox1.AllowDrop = true;

}

***Додаткові методи:***

Метод clear() очищює полотно:

public void clear()

{

Graphics g = pictureBox1.CreateGraphics();

g.Clear(Color.White);

pictureBox1.Image = null;

snapshot = new Bitmap(snapshot.Width, snapshot.Height);

tempDraw = (Bitmap)snapshot.Clone();

g.Dispose();

}

Метод змінює колір кисті та кнопки відображаємого кольору:

public void ReloadColor(Color a)

{

foreColor = a;

colorB.BackColor = foreColor;

pen.Color = foreColor;

}

***Події для pictureBox1:***

Коли кнопка миші клацне, активується подія pictureBox1\_MouseDown, яке переведе isPressed в true, збереже початкові координати (x1, y1) і скопіює snapshot в tempDraw:

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

isPressed = true;

x1 = e.X;

y1 = e.Y;

tempDraw = (Bitmap)snapshot.Clone();

}

Коли курсор миші буде рухатися, активується події pictureBox1\_MouseMove, яке спочатку перевірить, чи натиснута кнопка миші, якщо так, то зберігає координати миші (x2, y2) і викликає перерисовку pictureBox1:

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (isPressed)

{

x2 = e.X;

y2 = e.Y;

pictureBox1.Invalidate();

pictureBox1.Update();

}

}

Коли кнопку миші віджали, активується подія pictureBox1\_MouseUp, яке isPressed переводить на false, а tempdraw копіює в snapshot:

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

isPressed = false;

snapshot = (Bitmap)tempDraw.Clone();

}

Подія pictureBox1\_Paint спрацьовує при необхідності перемальовування елемента управління і є якраз тією подією, якою ми все малюємо. Залежно від обраного інструменту буде малюватися потрібна фігура.

Якщо значення selectedTool не дорівнює кисті і не дорівнює ластику, то копіюємо snapshot в tempDraw. Це потрібно для візуального зручності при малюванні;

Створюєм новий Graphics із зазначеного tempDraw

if (selectedTool != "brush" && selectedTool != "eraser" && selectedTool != "pensil")

tempDraw = (Bitmap)snapshot.Clone();

Graphics g = Graphics.FromImage(tempDraw);

Дивимось, який інструмент обраний, якщо обрана лінія, малюємо її в заданих координатах. В залежності від вибарного індексу в comboBox(choseLine), ми малюємо лінію різним стилем:

case "line":

if (tempDraw != null)

{

if (choseLine.SelectedIndex == 0)

{

g.DrawLine(pen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 1)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.Dash;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 2)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.DashDot;

g.DrawLine(myPen, x1, y1, x2, y2);

}

…..

else if (choseLine.SelectedIndex == 8)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.StartCap = LineCap.Triangle;

myPen.EndCap = LineCap.Flat;

g.DrawLine(myPen, x1, y1, x2, y2);

}

}

break;

Якщо обраний прямокутник, малює його в заданих координатах (для чотирьох сторін):

case "rectangle":

if (tempDraw != null)

{

if (x2 < x1 && y2 < y1)

{

g.FillRectangle(new SolidBrush(foreColor), x2, y2, Math.Abs(x2 - x1), Math.Abs(y2 - y1));

}

else if (x2 < x1 && y2 > y1)

{

g.FillRectangle(new SolidBrush(foreColor), x2, y1, Math.Abs(x2 - x1), y2 - y1);

}

else if (x2 > x1 && y2 < y1)

{

g.FillRectangle(new SolidBrush(foreColor), x1, y2, x2 - x1, Math.Abs(y2 - y1));

}

else

g.FillRectangle(new SolidBrush(foreColor), x1, y1, x2 - x1, y2 - y1);

}

break;

Аналогічно працює форма прямокутника, тількі замість FillRectangle буде DrawRectangle

Якщо обрана кисть, малює її в заданих координатах:

case "brush":

if (tempDraw != null)

g.DrawLine(pen, x1, y1, x2, y2);

x1 = x2;

y1 = y2;

break;

У разі вибору ластик, малює його в заданих координатах:

case "eraser":

if (tempDraw != null)

g.FillEllipse(new SolidBrush(pictureBox1.BackColor), x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

break;

У разі вибору еліпсу, малює його в заданих координатах:

case "ellipse":

g.FillEllipse(new SolidBrush(foreColor), x1, y1, x2 - x1, y2 - y1);

break;

Аналогічно працює форма еліпсу, тількі замість FillEllipse буде DrawEllipse

У разі вибору олівця, малює його в заданих координатах. В залежності від вибарного індексу в comboBox(pensilStyle), ми малюємо лінію різним стилем; Також додаємо візуалізацію зі згладжуванням:

case "pensil":

g.SmoothingMode = SmoothingMode.AntiAlias;

if (pensilStyle.SelectedIndex == 0)

{

g.FillEllipse(new SolidBrush(foreColor), x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 1)

{

HatchBrush hb = new HatchBrush(HatchStyle.Cross, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

…..

else if (pensilStyle.SelectedIndex == 26)

{

HatchBrush hb = new HatchBrush(HatchStyle.ZigZag, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

break;

У разі вибору тексту, ми свторюємо текст. Беремо назву шрифта з comboBox (fontName), якщо шрифт не обраний, то програма буде малювати текст автоматично вибраним VS, беремо розмір шрифта з comboBox (fontSize). В залежності від індексу, малюємо текст обраним стилем. Сам текст для промальовування ми беремо з textbox(textToDraw), який було введено користувачем. Малюємо текст обраним кольором у заданих координатах.

case "text":

if (fontStyle.SelectedIndex == 0)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Regular);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 1)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Bold);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 2)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Italic);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 3)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Underline);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

break;

У разі вибору трикутника, малює його в заданих координатах:

case "triangle":

if (tempDraw != null)

{

Point[] myPoints =

{

new Point(x1,y1), new Point(x2,y2), new Point(x1,y2)

};

g.FillPolygon(new SolidBrush(foreColor), myPoints);

}

break;

Аналогічно працює форма трикутника, тільки FillPolygon змінюємо на DrawPolygon

Подія pictureBox1\_MouseClick активується, коли клацають кнопкою миші. Воно призначене для піпетки і працює за тим же принципом, тобто, якщо обраний інструмент піпетка, то він повертає піксель, де була натиснута кнопка і змінює колір через ReloadColor:

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

if (selectedTool == "pip")

{

Bitmap copy = new Bitmap(snapshot);

ReloadColor(copy.GetPixel(e.X, e.Y));

}

}

Для того, щоб можна було перекидати зображення за допомогою миші, своренмо такі події:

private void pictureBox1\_DragDrop(object sender, DragEventArgs e)

{

var data = e.Data.GetData(DataFormats.FileDrop);

if(data != null)

{

var fileNames = data as string[];

if (fileNames.Length > 0)

pictureBox1.Image = Image.FromFile(fileNames[0]);

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

private void pictureBox1\_DragEnter(object sender, DragEventArgs e)

{

e.Effect = DragDropEffects.Copy;

}

***Події меню:***

Для першого підпункту пункта меню «File». Для "New" скористаємося методом clear:

private void newPicture\_Click(object sender, EventArgs e)

{

clear();

}

При натисканні "Open" відкриється діалогове вікно, в ньому вибирається файл. Після натискання "ОК" вікно закриється, очистить все зображення методом clear, потім відкриє вибране зображення в pictureBox1 і звідти скопіює його в snapshot і tempDraw, після цього перейменує нашу програму на шлях до зображення:

private void openPicture\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

clear();

pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Refresh();

pictureBox1.Update();

snapshot = new Bitmap(pictureBox1.Image);

tempDraw = new Bitmap(snapshot);

this.Text = openFileDialog1.FileName;

}

}

При натисканні "Save" відкриється діалогове вікно, в якому треба вибрати шлях, назва файлу і формат, для того щоб зберегти його в зазначеному місці, вказаним ім'ям і обраним форматом, а після призначає назва програми назвою записаного файлу:

private void savePicture\_Click(object sender, EventArgs e)

{

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string fileName = saveFileDialog1.FileName;

string strFilExtn = fileName.Remove(0, fileName.Length - 3);

this.Text = fileName;

switch (strFilExtn)

{

case "bmp": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Bmp); break;

case "jpg": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Jpeg); break;

case "png": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Png); break;

default: break;

}

}

}

При натисканні "Exit":

private void exit\_Click(object sender, EventArgs e)

{

Close();

}

Другий пункт меню «Filtres» відкриває другу форму:

private void filtresToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

f2.Show();

}

Третій пункт меню «About» відкриває MessageBox в якому є інформація про автора:

private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Виконав: Копилов В.Р., КН-19-2");

}

***Події для інструментів:***

Щоб можна було змінювати товщину кисті через sizeCB, є подія sizeCB\_TextChanged. Коли текст в sizeCB змінюється, товщина стає дорівнює цьому значенню. Щоб не виникало помилок при випадковому (або навмисному) написанні тексту в sizeCB, скористаємося try і catch:

private void sizeCB\_TextChanged(object sender, EventArgs e)

{

try

{

lineWidth = int.Parse(sizeCB.Text);

pen = new Pen(foreColor, lineWidth);

}

catch { }

}

При натисканні на кнопку кольору кисті, вилазить вікно з налаштуванням і вибором кольору, при натисканні на який колір змінюється за допомогою ReloadColor:

private void colorB\_Click(object sender, EventArgs e)

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

ReloadColor(colorDialog1.Color);

}

}

Кнопки швидкого доступу всі однакові, тільки різний колір:

private void whiteC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.White);

}

Щоб можна було змінювати товщину шрифта. Аналогічний до зміни товщини кисті:

private void fontSize\_TextChanged(object sender, EventArgs e)

{

try

{

FontSize = int.Parse(fontSize.Text);

}

catch { }

}

Для зміни фонового кольору полотна:

private void ButtonBackgroundColor\_Click(object sender, EventArgs e)

{

ColorDialog c = new ColorDialog();

if (c.ShowDialog() == DialogResult.OK)

{

pictureBox1.BackColor = c.Color;

ButtonBackgroundColor.BackColor = c.Color;

}

}

Для зменшення та збільшення розмірів полотна:

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (pictureBox1.Width < 2000 || pictureBox1.Height < 2000 && pictureBox1.Width > 100 || pictureBox1.Height > 100)

{

pictureBox1.Width += 100;

pictureBox1.Height += 100;

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

if (pictureBox1.Width < 2000 || pictureBox1.Height < 2000 && pictureBox1.Width > 100 || pictureBox1.Height > 100)

{

pictureBox1.Width -= 100;

pictureBox1.Height -= 100;

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

На всі кнопки при кліці діє одна подія tool\_click. Спочатку всі кнопки віджимаються, потім обрана стає такою, що натискує, а selectedTool змінюється на ім'я обраної кнопки:

private void tool\_click(object sender, EventArgs e)

{

brush.Checked = false;

line.Checked = false;

rectangle.Checked = false;

eraser.Checked = false;

ellipse.Checked = false;

none.Checked = false;

pip.Checked = false;

pensil.Checked = false;

spacerectangle.Checked = false;

spaceellipse.Checked = false;

text.Checked = false;

triangle1.Checked = false;

triangle.Checked = false;

ToolStripButton selected = sender as ToolStripButton;

selected.Checked = true;

selectedTool = selected.Name;

}

***Form2:***

Глобальні змінні:

Image file;

bool opened = false;

Метод reload() для очищення від застосованих фільтрів:

void reload()

{

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

if (opened)

{

file = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Image = file;

opened = true;

}

}

}

У програмі зроблено 9 фільтрів і всі вони працюють однаково, тільки змінюються значення, які відповідають значеням на RGB каналах та яскравості.

Наприклад, візьмемо фільтр Sepia.

Застосовуємо метод reload() для того, щоб фільтри не накладались один на одного. Далі перевіряємо , чи є на pictureBox зображення. Далі іде зберігання зображення у змінну img типу зображення з picturebox1. Потім іде створення Bitmap висоти імпортованого зображення у вікні зображення, що складається з піксельних даних для графічного зображення та його атрибутів. Створивши об’єкт зображення, розподілюємо ia, щоб змінити атрибут зображень. Тепер створимо об'єкт кольорової матриці для зміни кольорів або застосування фільтру до зображення. Далі задаємо значення… Передаємо кольорову матрицю об’єкту атрибута зображення ia. Створюємо новий об’єкт графіки з назвою g; Створюємо графічний об’єкт для зміни. Графіка newGraphics = Graphics.FromImage (imageFile); Далі розташовуємо картинку.

private void sepiaToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{.393f, .349f, .272f, 0, 0},

new float[]{.769f, .686f, .534f, 0, 0},

new float[]{.189f, .168f, .131f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

Події trackBar викорисовуєть метод rgbChanges();

private void redbar\_ValueChanged(object sender, EventArgs e)

{

rgbChanges();

}

Метод аналогічний до методів фільрів, за виключенням значень, які беруться з trackbar та оголошених змінних, для того, щоб відображати значення у лейблах.

void rgbChanges()

{

float changered = redbar.Value \* 0.1f;

float changegreen = greenbar.Value \* 0.1f;

float changeblue = bluebar.Value \* 0.1f;

redvalue.Text = changered.ToString();

greenvalue.Text = changegreen.ToString();

bluevalue.Text = changeblue.ToString();

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1+changered, 0, 0, 0, 0},

new float[]{0, 1+changegreen, 0, 0, 0},

new float[]{0, 0, 1+changeblue, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

Відкриття за збереження зображень відбувається так:

private void openToolStripMenuItem\_Click(object sender, EventArgs e)

{

DialogResult dr = openFileDialog1.ShowDialog();

if (dr == DialogResult.OK)

{

file = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Image = file;

opened = true;

}

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (opened)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Images|\*.png;\*.bmp;\*.jpg";

ImageFormat format = ImageFormat.Png;

if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string ext = Path.GetExtension(sfd.FileName);

switch (ext)

{

case ".jpg":

format = ImageFormat.Jpeg;

break;

case ".bmp":

format = ImageFormat.Bmp;

break;

}

pictureBox1.Image.Save(sfd.FileName, format);

}

}

else

{

MessageBox.Show("Зображення не завантажено, спочатку завантажте зображення");

}

}

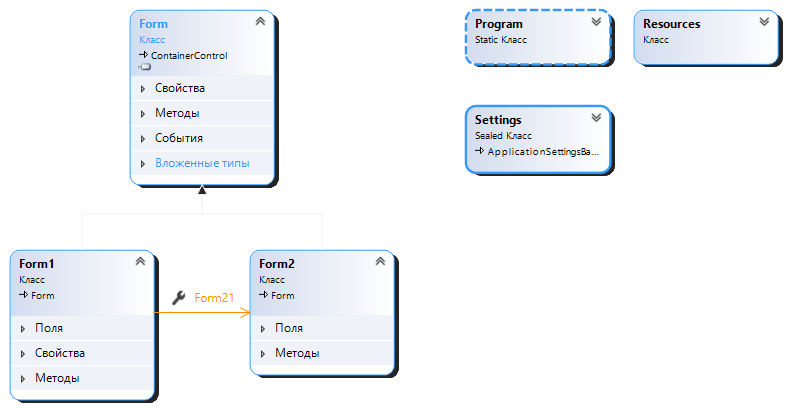


Рис.2.4. Діаграма класів.

## Висновки до другого розділу

У ході виконання другого розділу було спроектовано загальний функціонал програмного додатку та блок схема, яка її описує, було реалізовано графічний інтерфейс програми, були створені основні методи та події для обширного функціоналу програми, створена діаграма класів.

# **РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

## 3.1 Опис роботи додатку та його тестування

На даному етапі курсової роботи буде продемонстрована робота вже реалізованого графічного редактору. Буде протестовано можливості програмного продукту, весь його функціонал та можливості.

При запуску програми ми одразу бачимо її інтерфейс, одразу обрний інструмент «none»

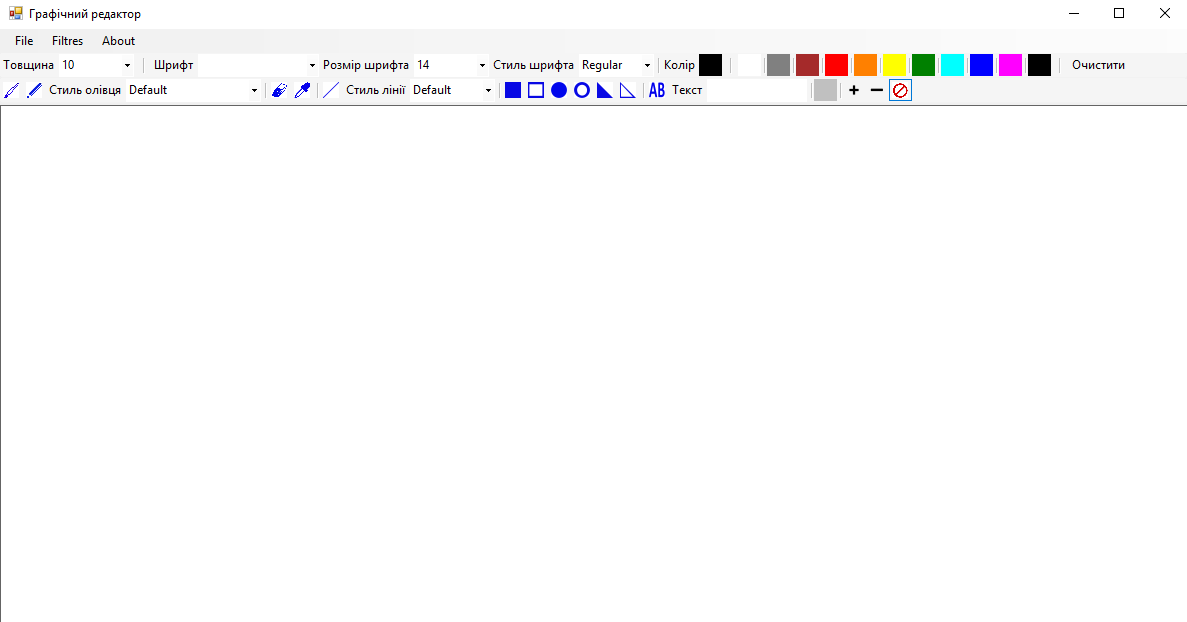


Рис.3.1.Інтерфейс програми

Протестуємо перший інструмент «Кисть» з налаштуванням кольору та товщини. Ми можемо налаштовувати її розмір:

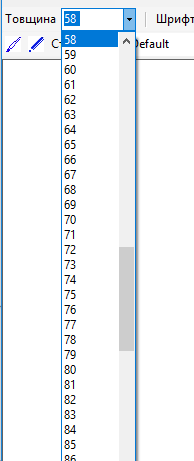


Рис.3.2.Вибір товщини

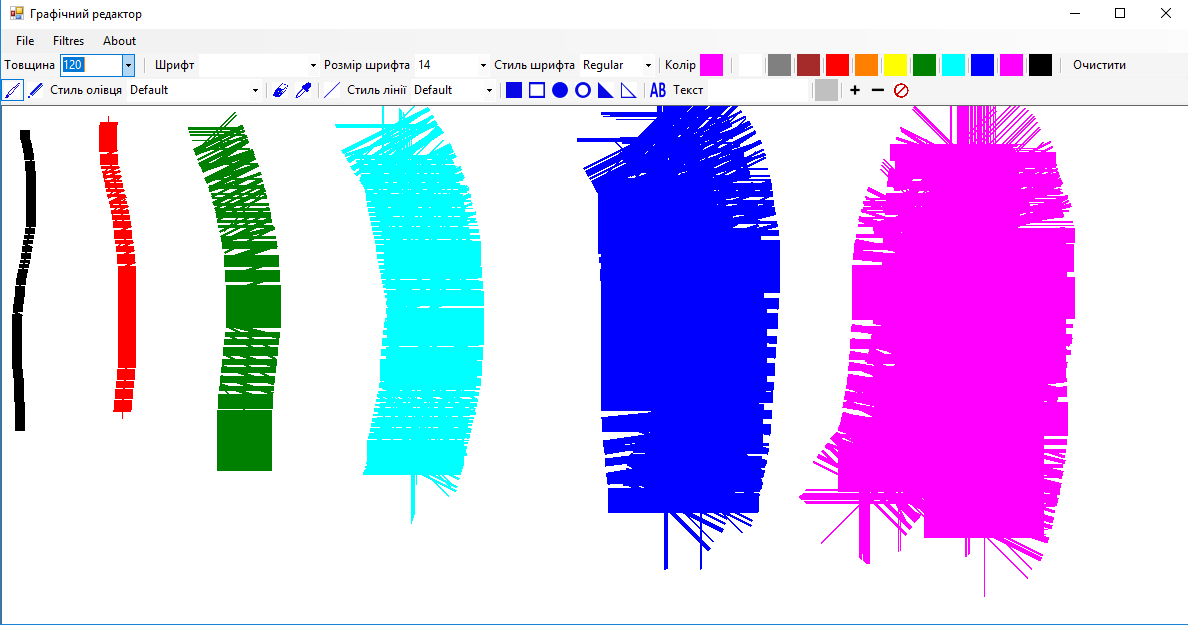


Рис.3.3.Тестування кисті

Другий інструмент має не один стиль, ми маємо можливість вибирати стиль олівця:

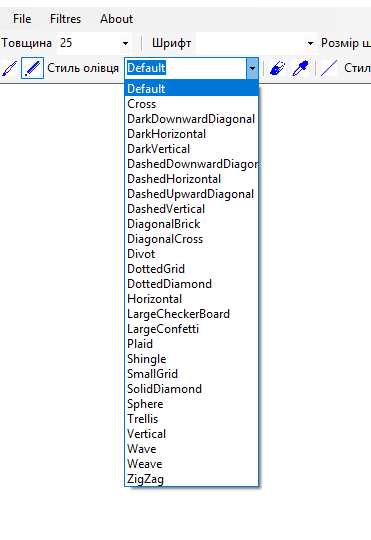


Рис.3.4.Стилі олівця

Протестуємо інструмент «Олівець» з налаштуванням кольору, товщини та різних стилів.

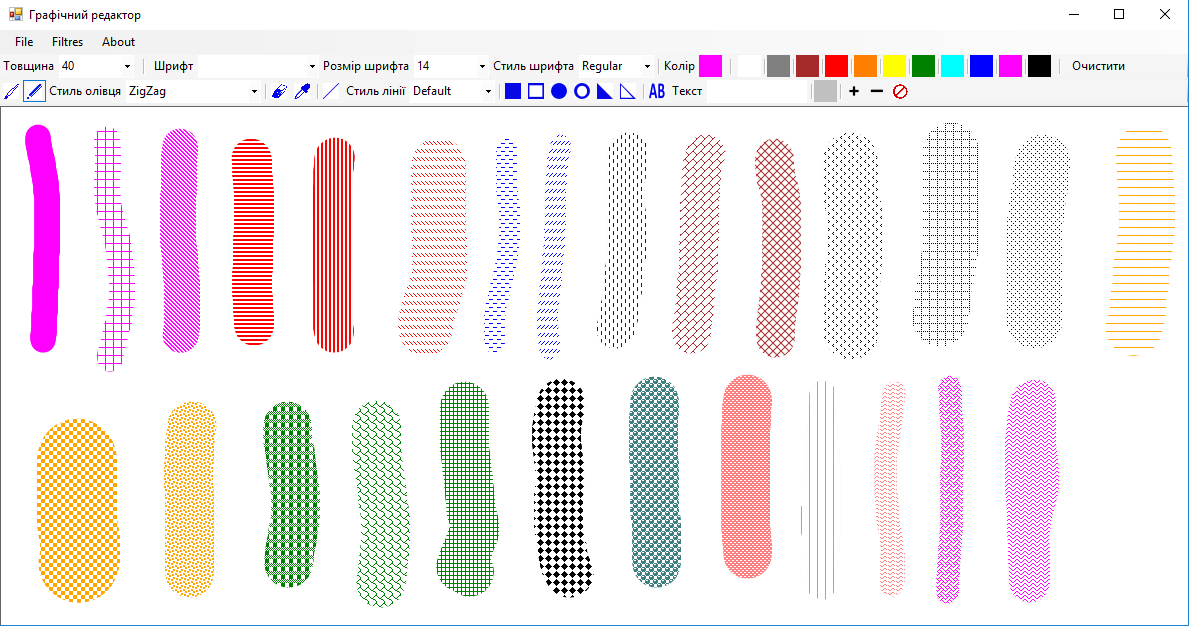


Рис.3.5.Тестування олівця

Далі іде інструмент «Ластик», давайте його протестуємо:

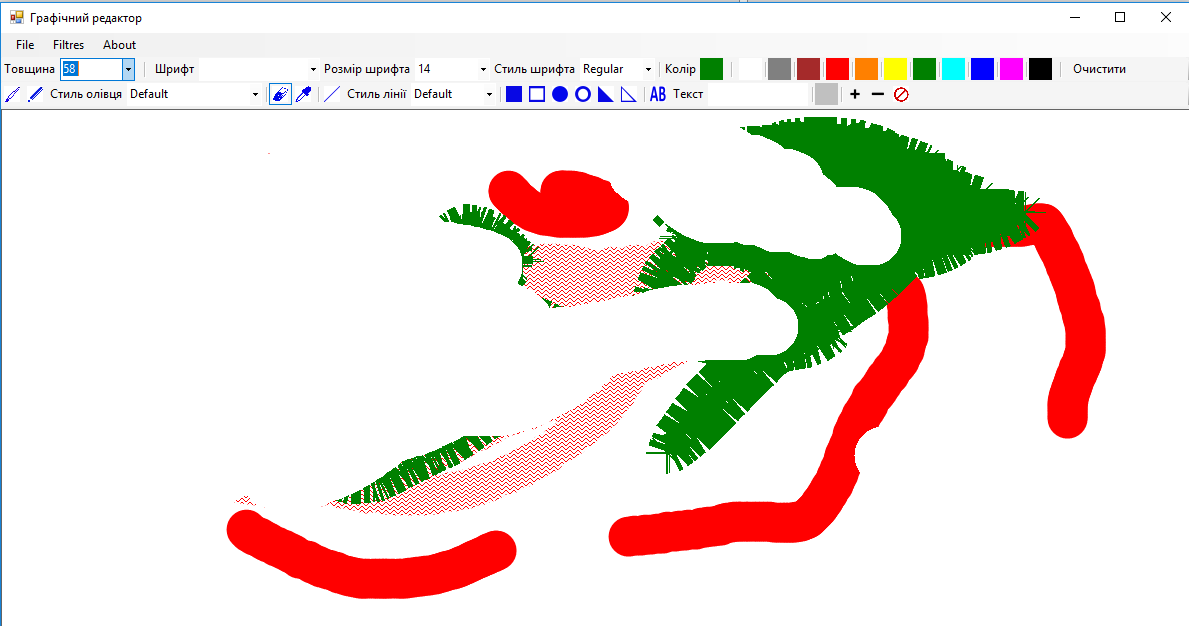


Рис.3.6.Тестування ластику

Інструмент «Піпетка» визначає колір:

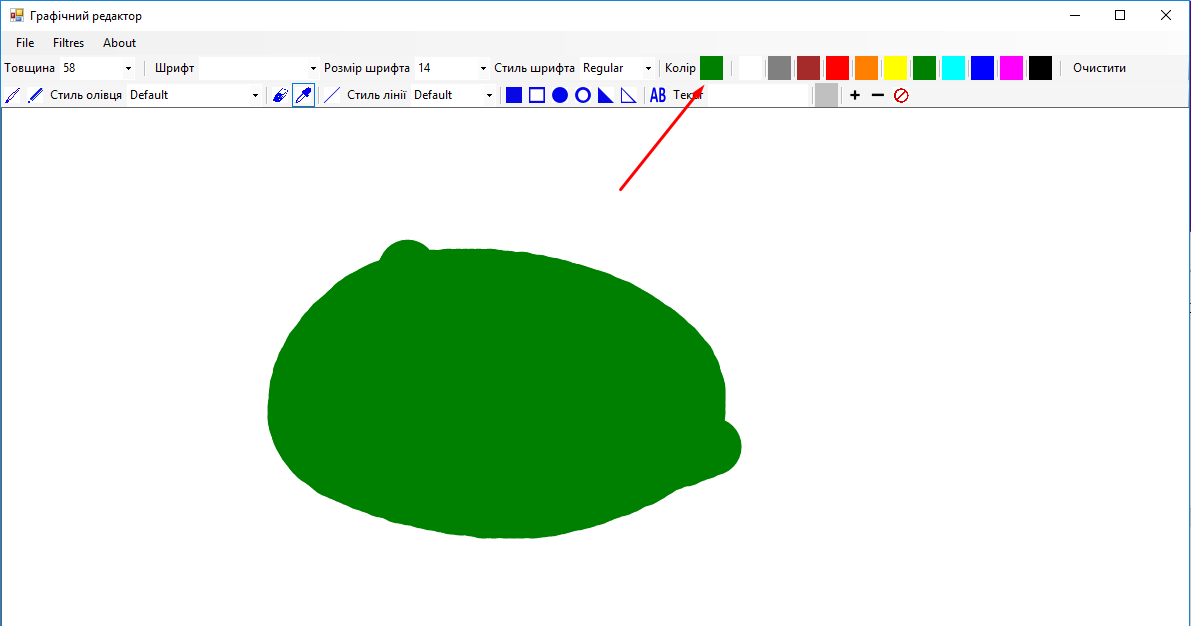


Рис.3.7.Тестування інструменту піпетка

Далі інструмент «Лінія». Малює пряму лінію. У цього інструменту також є різні стилі:

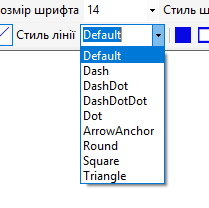


Рис.3.8.Стилі лінії

Протестуємо лінію різними кольорами, товщиною та стилями:



Рис.3.9.Тестування інструменту лінія

Далі іде ряд інструментів, які малюють прості геометричні фігури:

Розмір фігур змінюється разом із положенням мишки:

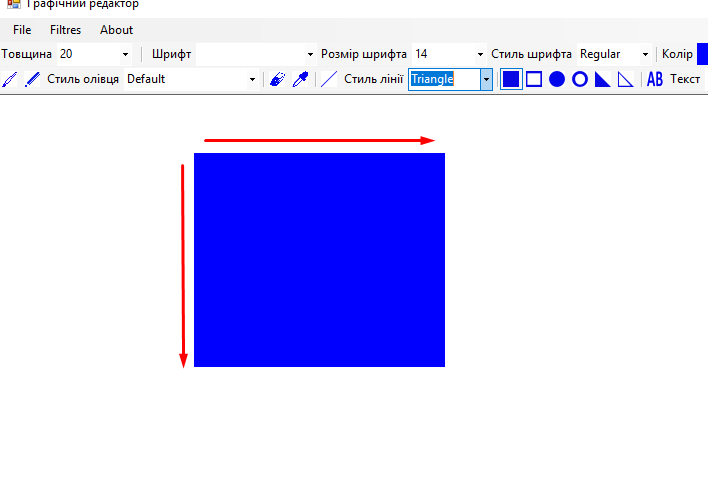


Рис.3.10.Тестування прямокутника

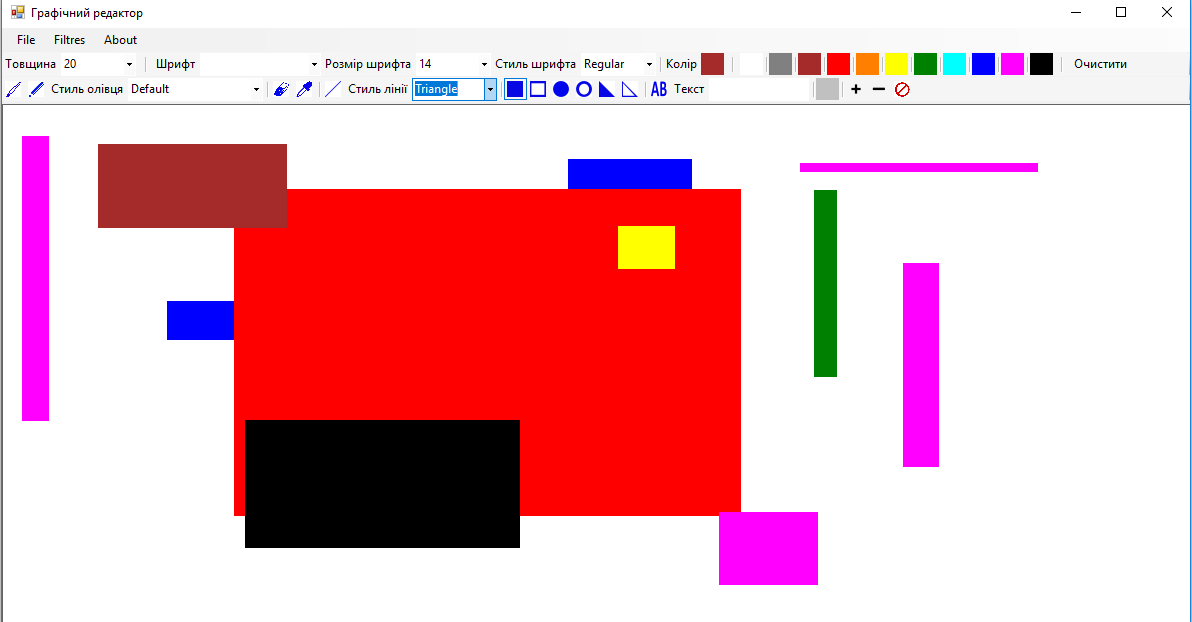


Рис.3.11.Тестування прямокутника різних форм

Форму прямокутника пряцює так само, також ми можемо змінювати її товщину:



Рис.3.12.Тестування форми прямокутника

Аналогічним чином протестуємо інструменти еліпса та трикутника:

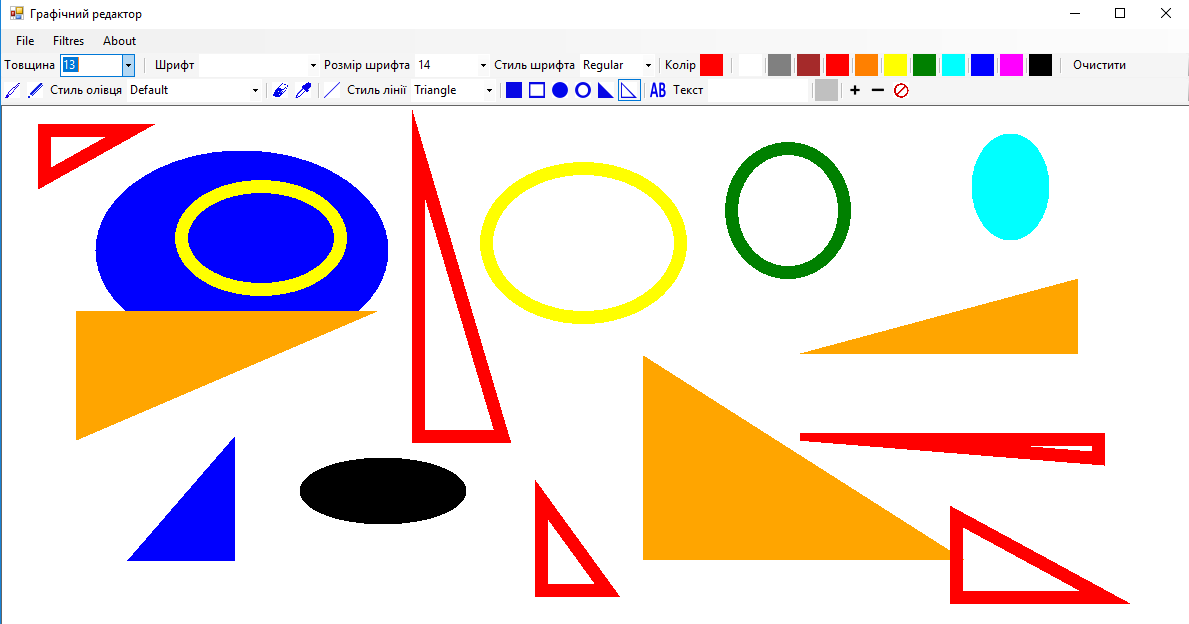


Рис.3.13.Тестування еліпсу та трикутника

Далі іде інструмент «Текст». Ми можемо змінювати шрифт тексту, його розмір, стиль та його колір. Для того, щоб промалювати текст, спочатку його треба записати у textBox:

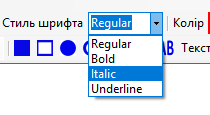
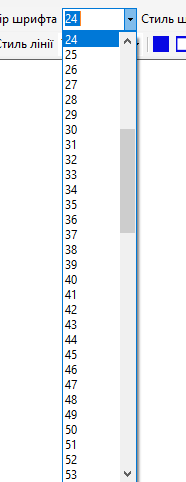
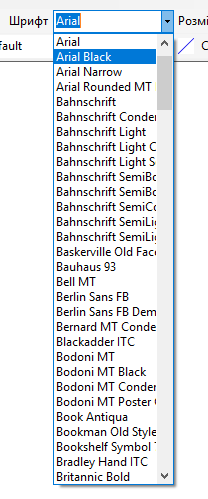


Рис.3.14.1. Вибір шрифут, товщини та стилю

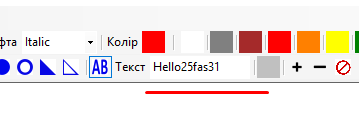


Рис.3.14.2.Написання тексту



Рис.3.14.3.Промальовування тексту на полотні

Далі іде колір фону, він змінює колір фоновий полотна:

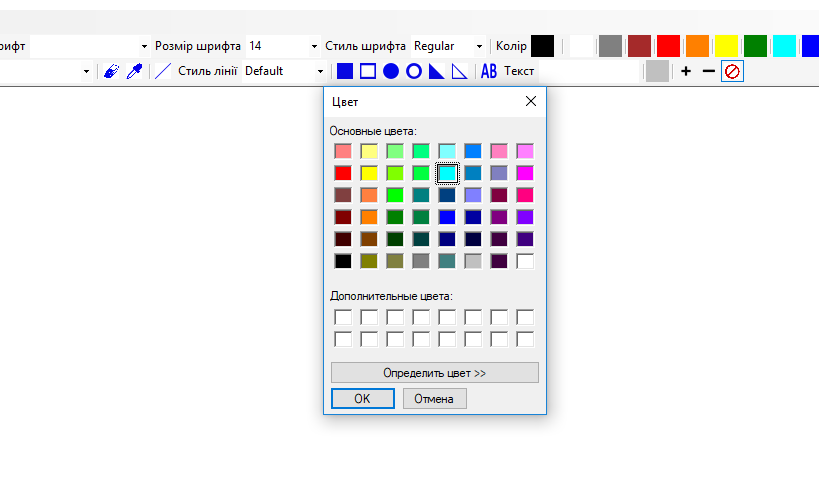


Рис.3.15.1.Вибір кольору фона



Рис.3.15.2.Зміна кольору фону та коліру кнопки до відповідного

Далі іде інструменти збільшення та зменшення розміру полотна:

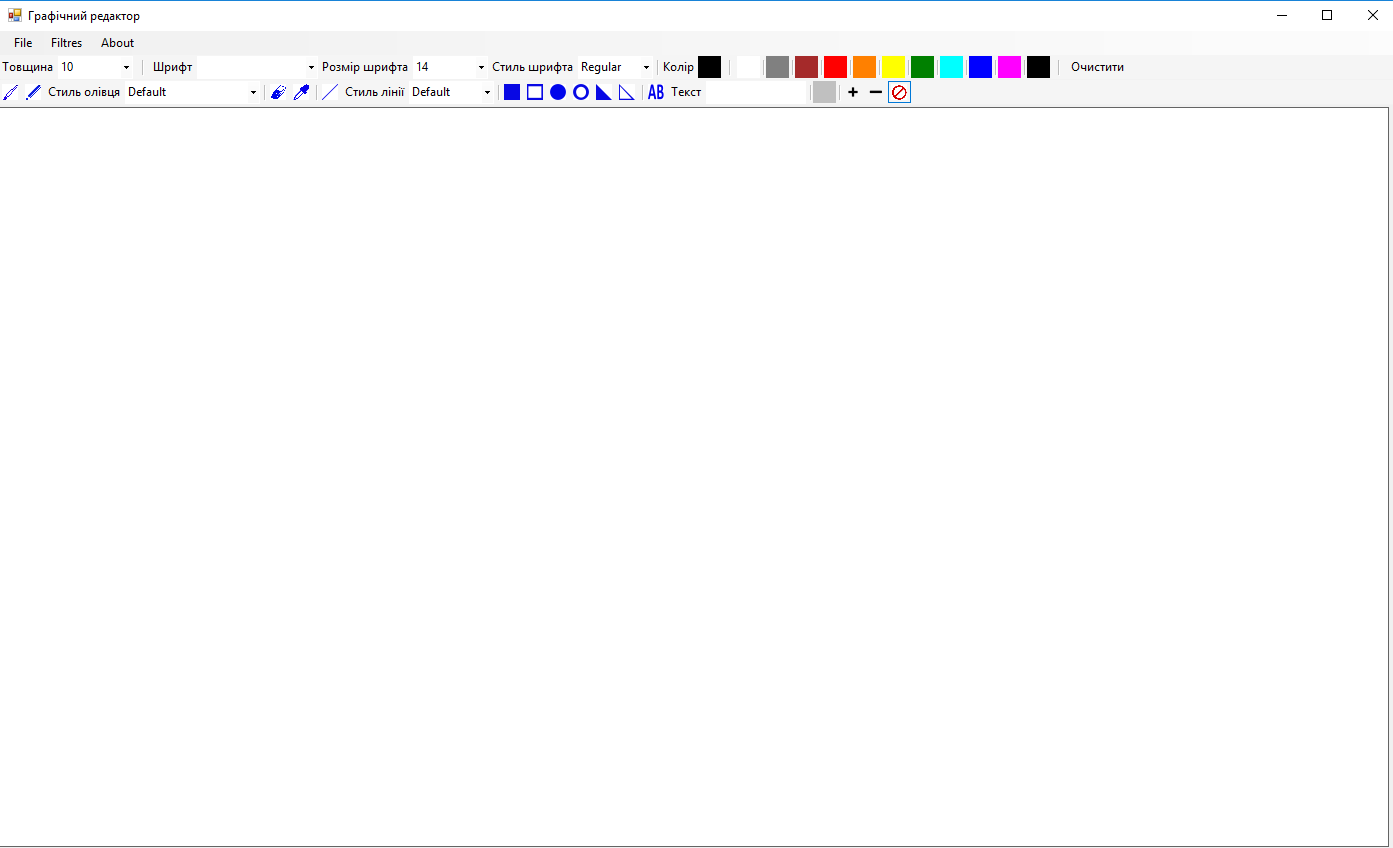


Рис.3.16.1.Збільшення розміру полотна

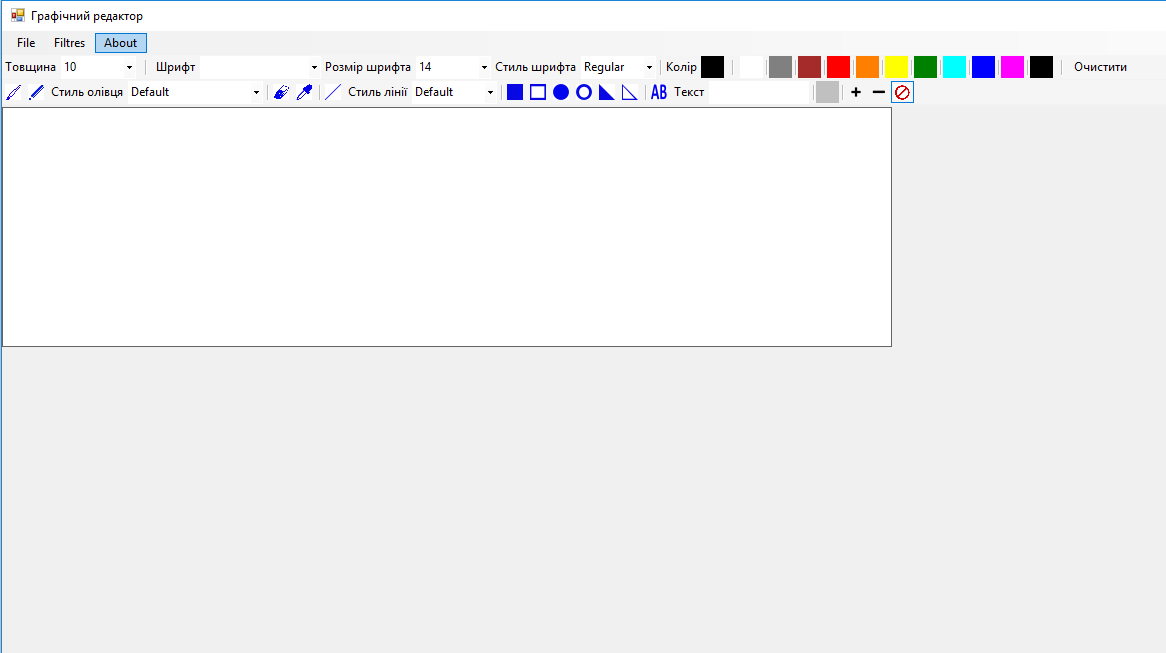


Рис.3.16.2.Зменшення розміру полотна

Також ми можемо відкрити зображення та робити усі вище вказані маніпуляції на зображенні:

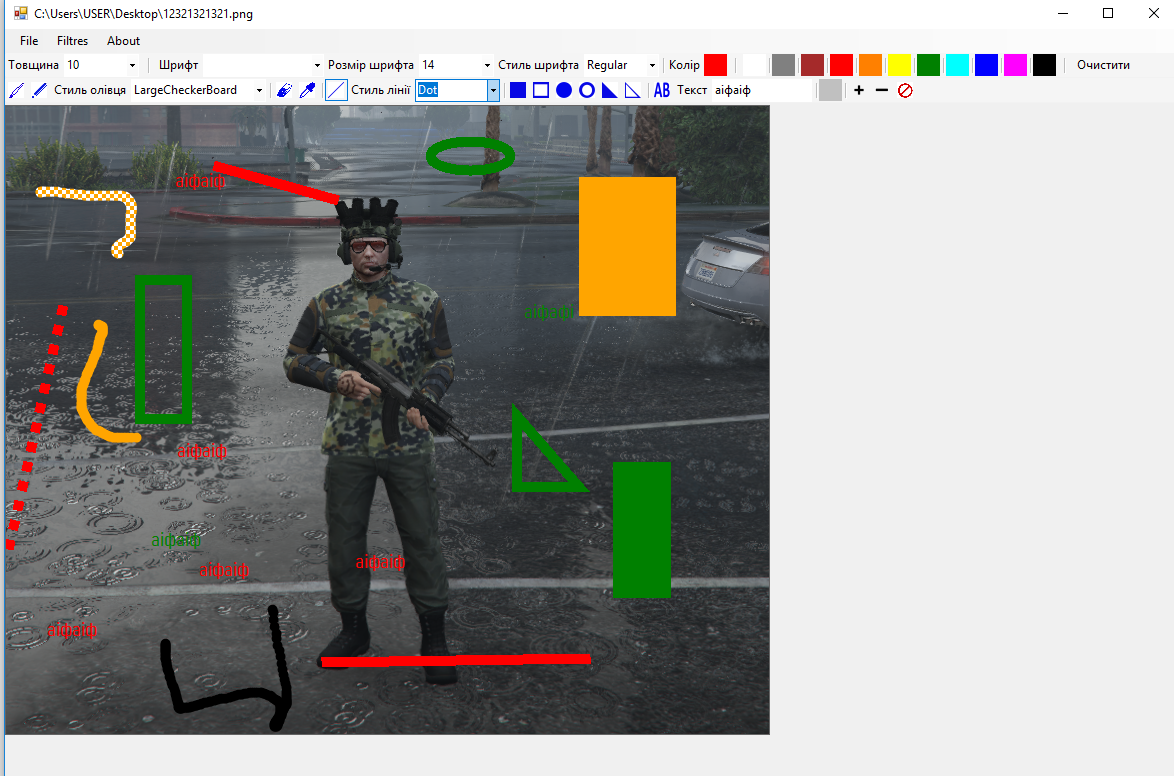


Рис.3.17.Малювання на зображені

Можливості меню:

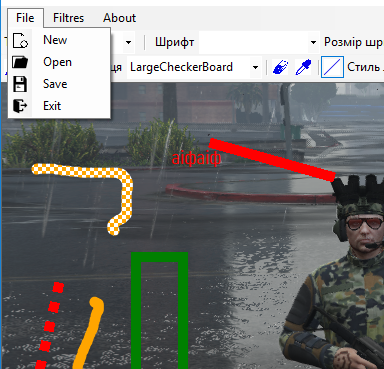


Рис.3.18.Можливості меню

У другій формі, на яку ми переходимо за допомогою меню ми можемо редагувати зображення за фільрами та на RGB каналах:

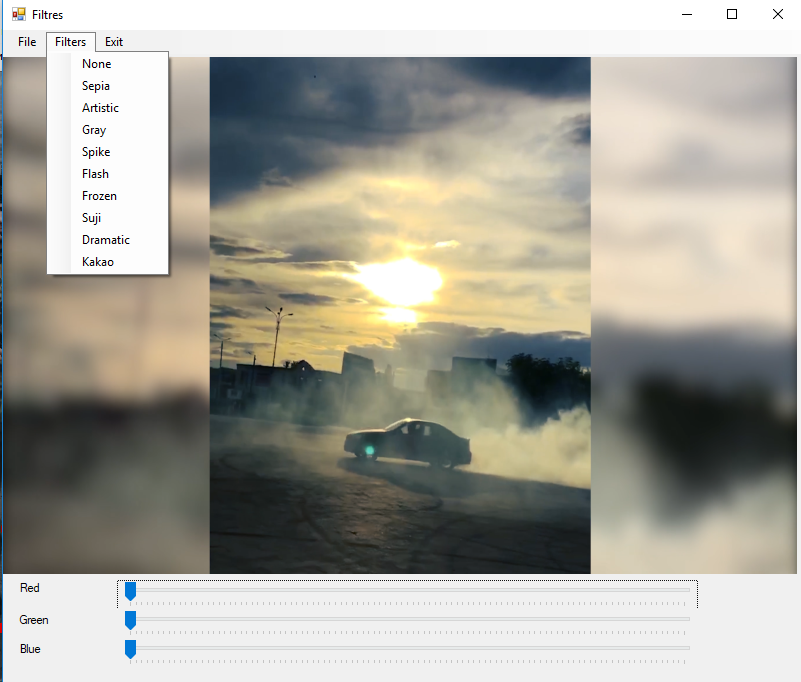


Рис.3.19.Можливості форми для редагування зображень

Протестуємо декілька фільтрів та редагування зображення на каналах

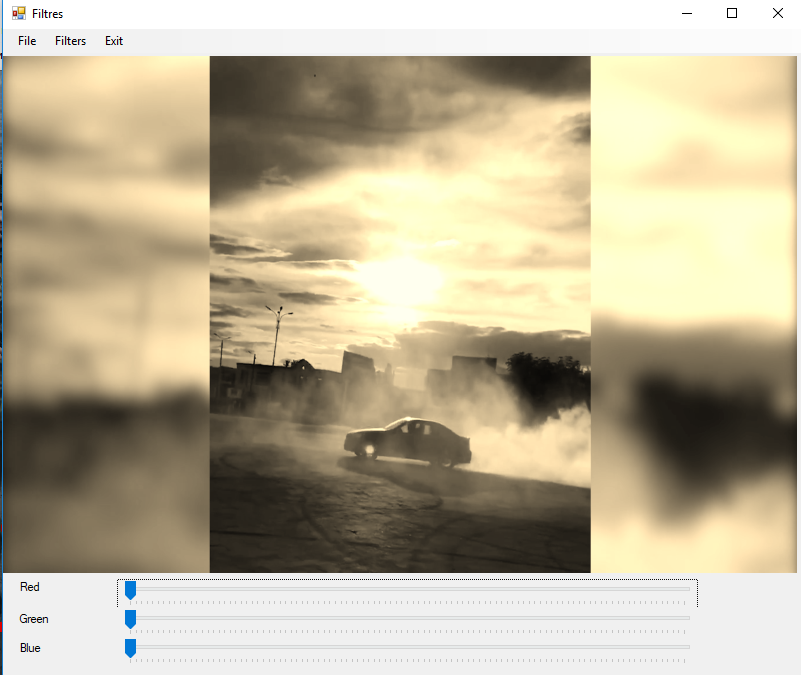


Рис.3.20.1.Фільтр Sepia

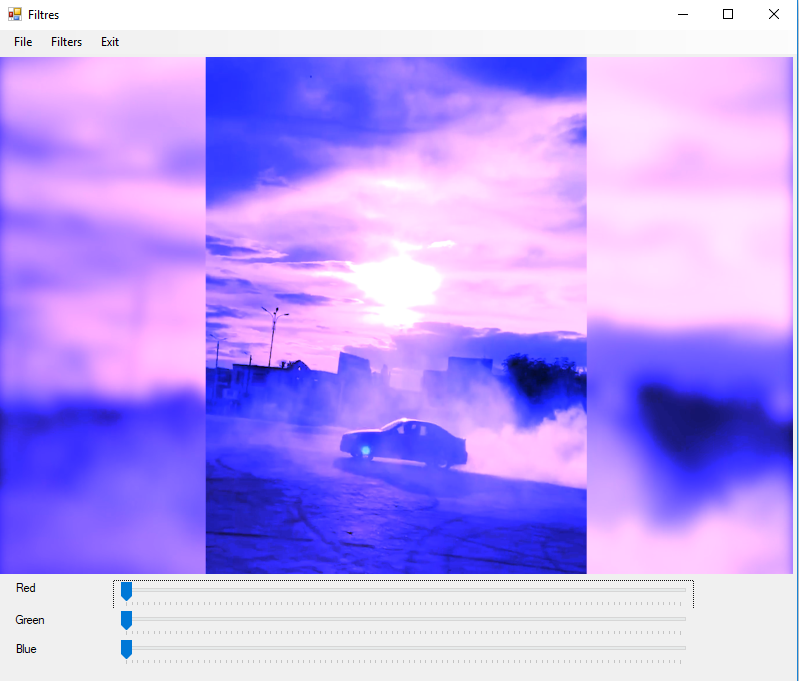


Рис.3.20.2.Фільтр Frozen

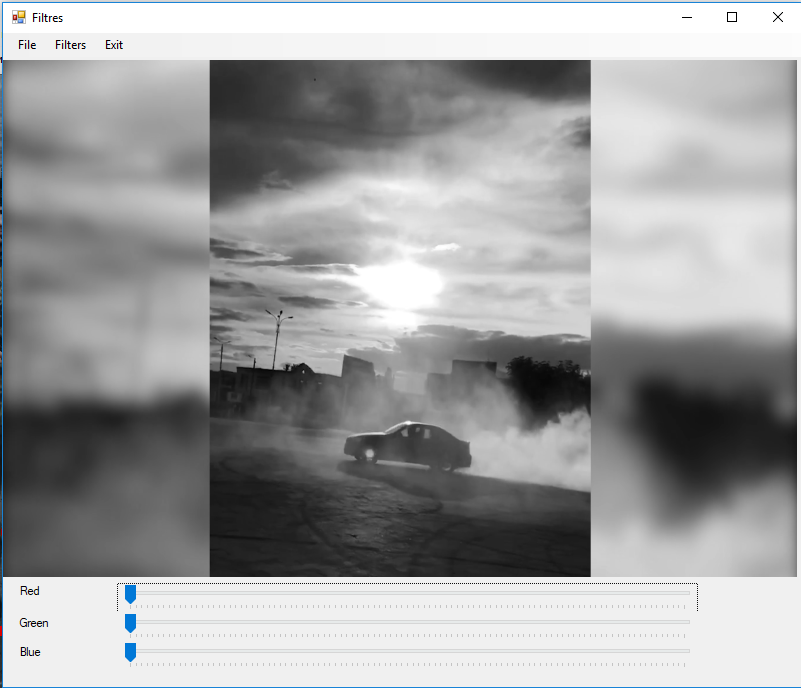


Рис.3.20.3.Фільр Gray

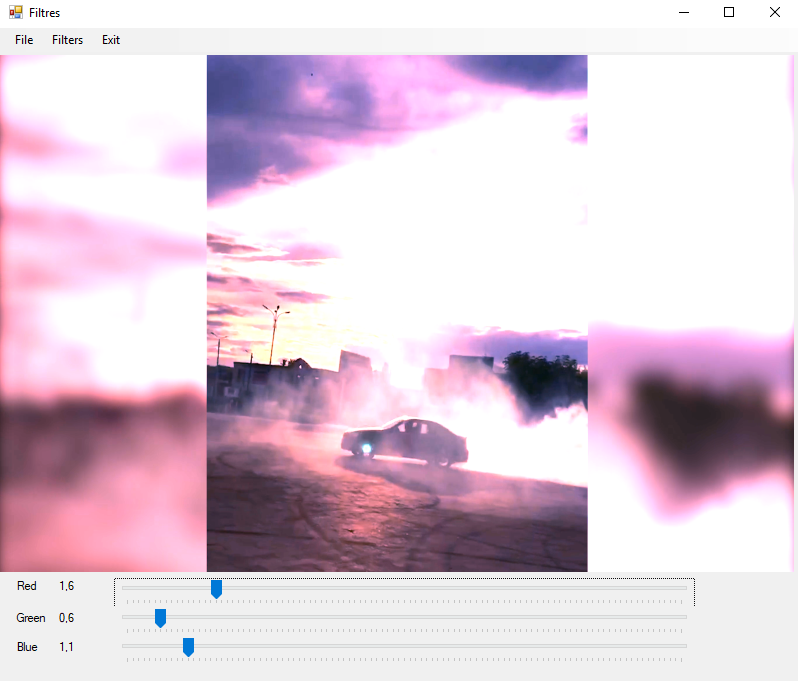


Рис.3.20.4.Редагування зображення

Можливості меню другої форми:

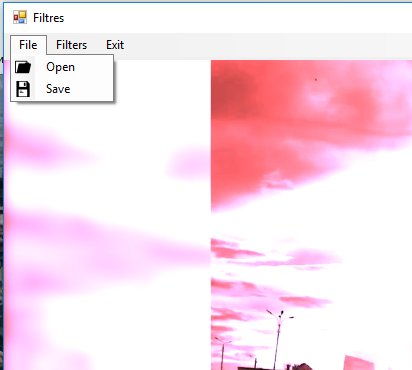


Рис.3.21.Можливості меню другої форми

## Висновки до третього розділу

В ході виконання третього розділу було протестовано графічний редактор. Були протестовані всі можливості програмного продукту, увесь його функціонал. Весь функціонал був розписаний та продемонстрований на відповідних рисунках. Під час роботи різного роду проблем не виникало.

**ВИСНОВКИ**

В першому розділі курсового проекту був спроектований графічний редактор, які можливості та який функіонал буде мати прогамний продукт.

В другому розділі курсового проекту було проведено розробку графічного редактору. При розробці функціонального алгоритму було проведено дослідження методів та подій розробки програмного коду додатку. На етапі був розроблений код графічного редактору. В результаті було отримано програмний код, який повністю відповідає спроектованій в першому розділі програмі та який справно працює і виконує все необхідне.

В третьому розділі курсового проекту було описані функціїї програми. Було продемонстровано та рисункаї увесь функціонал графічного редактору. В ході тестування проблем не виникало.

В результаті виконання курсового проекту отримано графічний редактор, який повністю відповідає запланованому функціоналу на початку проектування. Інтерфейс програми вийшов повністю комфортним та інтуїтивно зрозумілим, а функціонал досить общирним, тому програма є зручною в користуванні.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. [Герберт](https://vk.com/wall-54530371_2781) Шілд – C# 4.0 : повне керівництво (2011)
2. [Документація](https://vk.com/wall-54530371_100580) Microsoft на мові C#. Практичний посібник. Створення об'єктів Graphics для малювання. Режим доступу: <https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/advanced/how-to-create-graphics-objects-for-drawing>
3. YouTube канал про програмуванні на мові С#. Режим доступу: <https://www.youtube.com/user/admshwan/videos>
4. [Вікіпедія про графічні редактоки](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80).
5. Сімоновіч С.В. Комп’ютерна графіка.(2005)
6. Веб-сторінка про графічний інтерфейс на мові C#. Режим доступу: <http://www.frolov-lib.ru/books/msnet/c_sharp2/ch10.html>
7. YouTube – канал miniprogrammers. Режим доступу: <https://www.youtube.com/channel/UCMW-UzXq6qzEYCCaEVbI7_Q>
8. Сайт про програмування. Режим доступу: <https://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php>
9. Сайт про програмування. Режим доступу: <https://metanit.com/>
10. YouTube – канал Dawisko1. Режим доступу: <https://www.youtube.com/user/Dawisko1/videos>

# **ДОДАТОК**

Додаток

Програмний код

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

namespace Graphics\_editor

{

public partial class Form1 : Form

{

bool isPressed;

int x1, y1, x2, y2;

public Bitmap snapshot, tempDraw;

Color foreColor;

int lineWidth = 2;

int FontSize = 2;

string selectedTool = "none";

Pen pen;

public Form2 Form21

{

get => default;

set

{

}

}

public void clear()

{

Graphics g = pictureBox1.CreateGraphics();

g.Clear(Color.White);

pictureBox1.Image = null;

snapshot = new Bitmap(snapshot.Width, snapshot.Height);

tempDraw = (Bitmap)snapshot.Clone();

g.Dispose();

}

public void ReloadColor(Color a)

{

foreColor = a;

colorB.BackColor = foreColor;

pen.Color = foreColor;

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

if (selectedTool != "brush" && selectedTool != "eraser" && selectedTool != "pensil")

tempDraw = (Bitmap)snapshot.Clone();

Graphics g = Graphics.FromImage(tempDraw);

switch (selectedTool)

{

case "line":

if (tempDraw != null)

{

if (choseLine.SelectedIndex == 0)

{

g.DrawLine(pen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 1)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.Dash;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 2)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.DashDot;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 3)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.DashDotDot;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 4)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.DashStyle = DashStyle.Dot;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 5)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.StartCap = LineCap.ArrowAnchor;

myPen.EndCap = LineCap.DiamondAnchor;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 6)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.StartCap = LineCap.Round;

myPen.EndCap = LineCap.RoundAnchor;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 7)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.StartCap = LineCap.Square;

myPen.EndCap = LineCap.SquareAnchor;

g.DrawLine(myPen, x1, y1, x2, y2);

}

else if (choseLine.SelectedIndex == 8)

{

Pen myPen = new Pen(foreColor, lineWidth);

myPen.StartCap = LineCap.Triangle;

myPen.EndCap = LineCap.Flat;

g.DrawLine(myPen, x1, y1, x2, y2);

}

}

break;

case "rectangle":

if (tempDraw != null)

{

if (x2 < x1 && y2 < y1)

{

g.FillRectangle(new SolidBrush(foreColor), x2, y2, Math.Abs(x2 - x1), Math.Abs(y2 - y1));

}

else if (x2 < x1 && y2 > y1)

{

g.FillRectangle(new SolidBrush(foreColor), x2, y1, Math.Abs(x2 - x1), y2 - y1);

}

else if (x2 > x1 && y2 < y1)

{

g.FillRectangle(new SolidBrush(foreColor), x1, y2, x2 - x1, Math.Abs(y2 - y1));

}

else

g.FillRectangle(new SolidBrush(foreColor), x1, y1, x2 - x1, y2 - y1);

}

break;

case "brush":

if (tempDraw != null)

g.DrawLine(pen, x1, y1, x2, y2);

x1 = x2;

y1 = y2;

break;

case "eraser":

if (tempDraw != null)

g.FillEllipse(new SolidBrush(pictureBox1.BackColor), x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

break;

case "ellipse":

g.FillEllipse(new SolidBrush(foreColor), x1, y1, x2 - x1, y2 - y1);

break;

case "pensil":

g.SmoothingMode = SmoothingMode.AntiAlias;

if (pensilStyle.SelectedIndex == 0)

{

g.FillEllipse(new SolidBrush(foreColor), x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 1)

{

HatchBrush hb = new HatchBrush(HatchStyle.Cross, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 2)

{

HatchBrush hb = new HatchBrush(HatchStyle.DarkDownwardDiagonal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 3)

{

HatchBrush hb = new HatchBrush(HatchStyle.DarkHorizontal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 4)

{

HatchBrush hb = new HatchBrush(HatchStyle.DarkVertical, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 5)

{

HatchBrush hb = new HatchBrush(HatchStyle.DashedDownwardDiagonal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 6)

{

HatchBrush hb = new HatchBrush(HatchStyle.DashedHorizontal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 7)

{

HatchBrush hb = new HatchBrush(HatchStyle.DashedUpwardDiagonal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 8)

{

HatchBrush hb = new HatchBrush(HatchStyle.DashedVertical, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 9)

{

HatchBrush hb = new HatchBrush(HatchStyle.DiagonalBrick, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 10)

{

HatchBrush hb = new HatchBrush(HatchStyle.DiagonalCross, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 11)

{

HatchBrush hb = new HatchBrush(HatchStyle.Divot, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 12)

{

HatchBrush hb = new HatchBrush(HatchStyle.DottedGrid, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 13)

{

HatchBrush hb = new HatchBrush(HatchStyle.DottedDiamond, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 14)

{

HatchBrush hb = new HatchBrush(HatchStyle.Horizontal, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 15)

{

HatchBrush hb = new HatchBrush(HatchStyle.LargeCheckerBoard, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 16)

{

HatchBrush hb = new HatchBrush(HatchStyle.LargeConfetti, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 17)

{

HatchBrush hb = new HatchBrush(HatchStyle.Plaid, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 18)

{

HatchBrush hb = new HatchBrush(HatchStyle.Shingle, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 19)

{

HatchBrush hb = new HatchBrush(HatchStyle.SmallGrid, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 20)

{

HatchBrush hb = new HatchBrush(HatchStyle.SolidDiamond, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 21)

{

HatchBrush hb = new HatchBrush(HatchStyle.Sphere, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 22)

{

HatchBrush hb = new HatchBrush(HatchStyle.Trellis, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 23)

{

HatchBrush hb = new HatchBrush(HatchStyle.Vertical, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 24)

{

HatchBrush hb = new HatchBrush(HatchStyle.Wave, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 25)

{

HatchBrush hb = new HatchBrush(HatchStyle.Weave, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

else if (pensilStyle.SelectedIndex == 26)

{

HatchBrush hb = new HatchBrush(HatchStyle.ZigZag, foreColor, Color.White);

g.FillEllipse(hb, x1, y1, lineWidth, lineWidth);

x1 = x2;

y1 = y2;

}

break;

case "spacerectangle":

if (tempDraw != null)

{

if (x2 < x1 && y2 < y1)

{

g.DrawRectangle(pen, x2, y2, Math.Abs(x2 - x1), Math.Abs(y2 - y1));

}

else if (x2 < x1 && y2 > y1)

{

g.DrawRectangle(pen, x2, y1, Math.Abs(x2 - x1), y2 - y1);

}

else if (x2 > x1 && y2 < y1)

{

g.DrawRectangle(pen, x1, y2, x2 - x1, Math.Abs(y2 - y1));

}

else

g.DrawRectangle(pen, x1, y1, x2 - x1, y2 - y1);

}

break;

case "spaceellipse":

g.DrawEllipse(pen, x1, y1, x2 - x1, y2 - y1);

break;

case "text":

if (fontStyle.SelectedIndex == 0)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Regular);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 1)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Bold);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 2)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Italic);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

else if (fontStyle.SelectedIndex == 3)

{

Font f1 = new Font(fontName.Text, FontSize, FontStyle.Underline);

g.DrawString(textToDraw.Text, f1, new SolidBrush(foreColor), x1, y1);

}

break;

case "triangle1":

if (tempDraw != null)

{

Point[] myPoints =

{

new Point(x1,y1), new Point(x2,y2), new Point(x1,y2)

};

g.DrawPolygon(pen, myPoints);

}

break;

case "triangle":

if (tempDraw != null)

{

Point[] myPoints =

{

new Point(x1,y1), new Point(x2,y2), new Point(x1,y2)

};

g.FillPolygon(new SolidBrush(foreColor), myPoints);

}

break;

}

g.Dispose();

e.Graphics.DrawImageUnscaled(tempDraw, 0, 0);

}

private void tool\_click(object sender, EventArgs e)

{

brush.Checked = false;

line.Checked = false;

rectangle.Checked = false;

eraser.Checked = false;

ellipse.Checked = false;

none.Checked = false;

pip.Checked = false;

pensil.Checked = false;

spacerectangle.Checked = false;

spaceellipse.Checked = false;

text.Checked = false;

triangle1.Checked = false;

triangle.Checked = false;

ToolStripButton selected = sender as ToolStripButton;

selected.Checked = true;

selectedTool = selected.Name;

}

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

isPressed = true;

x1 = e.X;

y1 = e.Y;

tempDraw = (Bitmap)snapshot.Clone();

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (isPressed)

{

x2 = e.X;

y2 = e.Y;

pictureBox1.Invalidate();

pictureBox1.Update();

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

isPressed = false;

snapshot = (Bitmap)tempDraw.Clone();

}

private void sizeCB\_TextChanged(object sender, EventArgs e)

{

try

{

lineWidth = int.Parse(sizeCB.Text);

pen = new Pen(foreColor, lineWidth);

}

catch { }

}

private void newPicture\_Click(object sender, EventArgs e)

{

clear();

}

private void openPicture\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

clear();

pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Refresh();

pictureBox1.Update();

snapshot = new Bitmap(pictureBox1.Image);

tempDraw = new Bitmap(snapshot);

this.Text = openFileDialog1.FileName;

}

}

private void savePicture\_Click(object sender, EventArgs e)

{

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string fileName = saveFileDialog1.FileName;

string strFilExtn = fileName.Remove(0, fileName.Length - 3);

this.Text = fileName;

switch (strFilExtn)

{

case "bmp": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Bmp); break;

case "jpg": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Jpeg); break;

case "png": snapshot.Save(fileName, System.Drawing.Imaging.ImageFormat.Png); break;

default: break;

}

}

}

private void colorB\_Click(object sender, EventArgs e)

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

ReloadColor(colorDialog1.Color);

}

}

private void clearPaint\_Click(object sender, EventArgs e)

{

clear();

}

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

if (selectedTool == "pip")

{

Bitmap copy = new Bitmap(snapshot);

ReloadColor(copy.GetPixel(e.X, e.Y));

}

}

private void whiteC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.White);

}

private void grayC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Gray);

}

private void brownC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Brown);

}

private void redC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Red);

}

private void orangeC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Orange);

}

private void yellowC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Yellow);

}

private void greenC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Green);

}

private void aquaC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Aqua);

}

private void blackC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Black);

}

private void Form1\_Load(object sender, EventArgs e)

{

FontFamily[] family = FontFamily.Families;

foreach (FontFamily font in family)

{

fontName.Items.Add(font.GetName(1).ToString());

}

pictureBox1.AllowDrop = true;

}

private void fontSize\_TextChanged(object sender, EventArgs e)

{

try

{

FontSize = int.Parse(fontSize.Text);

}

catch { }

}

private void ButtonBackgroundColor\_Click(object sender, EventArgs e)

{

ColorDialog c = new ColorDialog();

if (c.ShowDialog() == DialogResult.OK)

{

pictureBox1.BackColor = c.Color;

ButtonBackgroundColor.BackColor = c.Color;

}

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (pictureBox1.Width < 2000 || pictureBox1.Height < 2000 && pictureBox1.Width > 100 || pictureBox1.Height > 100)

{

pictureBox1.Width += 100;

pictureBox1.Height += 100;

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

if (pictureBox1.Width < 2000 || pictureBox1.Height < 2000 && pictureBox1.Width > 100 || pictureBox1.Height > 100)

{

pictureBox1.Width -= 100;

pictureBox1.Height -= 100;

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

private void pictureBox1\_DragDrop(object sender, DragEventArgs e)

{

var data = e.Data.GetData(DataFormats.FileDrop);

if(data != null)

{

var fileNames = data as string[];

if (fileNames.Length > 0)

pictureBox1.Image = Image.FromFile(fileNames[0]);

}

snapshot = new Bitmap(pictureBox1.Width, pictureBox1.Height);

tempDraw = (Bitmap)snapshot.Clone();

}

private void pictureBox1\_DragEnter(object sender, DragEventArgs e)

{

e.Effect = DragDropEffects.Copy;

}

private void filtresToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

f2.Show();

}

private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Виконав: Копилов В.Р., КН-19-2");

}

private void exit\_Click(object sender, EventArgs e)

{

Close();

}

private void blueC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Blue);

}

private void fuchsiaC\_Click(object sender, EventArgs e)

{

ReloadColor(Color.Fuchsia);

}

public Form1()

{

InitializeComponent();

snapshot = new Bitmap(pictureBox1.ClientRectangle.Width, pictureBox1.ClientRectangle.Height);

tempDraw = (Bitmap)snapshot.Clone();

foreColor = Color.Black;

pen = new Pen(foreColor, lineWidth);

none.Checked = true;

sizeCB.SelectedIndex = 9;

choseLine.SelectedIndex = 0;

pensilStyle.SelectedIndex = 0;

fontSize.SelectedIndex = 13;

fontStyle.SelectedIndex = 0;

}

}

}

**Form2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Imaging;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Graphics\_editor

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

Image file;

bool opened = false;

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

void reload()

{

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

if (opened)

{

file = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Image = file;

opened = true;

}

}

}

void rgbChanges()

{

float changered = redbar.Value \* 0.1f;

float changegreen = greenbar.Value \* 0.1f;

float changeblue = bluebar.Value \* 0.1f;

redvalue.Text = changered.ToString();

greenvalue.Text = changegreen.ToString();

bluevalue.Text = changeblue.ToString();

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1+changered, 0, 0, 0, 0},

new float[]{0, 1+changegreen, 0, 0, 0},

new float[]{0, 0, 1+changeblue, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void noneToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

}

private void sepiaToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{.393f, .349f, .272f, 0, 0},

new float[]{.769f, .686f, .534f, 0, 0},

new float[]{.189f, .168f, .131f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void artisticToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1,0,0,0,0},

new float[]{0,1,0,0,0},

new float[]{0,0,1,0,0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 1, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void grayToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{0.299f, 0.299f, 0.299f, 0, 0},

new float[]{0.587f, 0.587f, 0.587f, 0, 0},

new float[]{0.114f, 0.114f, 0.114f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 0}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void spikeToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1+0.3f, 0, 0, 0, 0},

new float[]{0, 1+0.7f, 0, 0, 0},

new float[]{0, 0, 1+1.3f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void flashToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1+0.9f, 0, 0, 0, 0},

new float[]{0, 1+1.5f, 0, 0, 0},

new float[]{0, 0, 1+1.3f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void frozenToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{1+0.3f, 0, 0, 0, 0},

new float[]{0, 1+0f, 0, 0, 0},

new float[]{0, 0, 1+5f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void sujiToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{.393f, .349f+0.5f, .272f, 0, 0},

new float[]{.769f+0.3f, .686f, .534f, 0, 0},

new float[]{.189f, .168f, .131f+0.5f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void dramaticToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{.393f+0.3f, .349f, .272f, 0, 0},

new float[]{.769f, .686f+0.2f, .534f, 0, 0},

new float[]{.189f, .168f, .131f+0.9f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void kakaoToolStripMenuItem\_Click(object sender, EventArgs e)

{

reload();

if (!opened)

{

MessageBox.Show("Відкрийте зображення, а потім застосуйте зміни");

}

else

{

Image img = pictureBox1.Image;

Bitmap bmpInverted = new Bitmap(img.Width, img.Height);

ImageAttributes ia = new ImageAttributes();

ColorMatrix cmPicture = new ColorMatrix(new float[][]

{

new float[]{.393f, .349f, .272f+1.3f, 0, 0},

new float[]{.769f, .686f+0.5f, .534f, 0, 0},

new float[]{.189f+2.3f, .168f, .131f, 0, 0},

new float[]{0, 0, 0, 1, 0},

new float[]{0, 0, 0, 0, 1}

});

ia.SetColorMatrix(cmPicture);

Graphics g = Graphics.FromImage(bmpInverted);

g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);

g.Dispose();

pictureBox1.Image = bmpInverted;

}

}

private void redbar\_ValueChanged(object sender, EventArgs e)

{

rgbChanges();

}

private void greenbar\_ValueChanged(object sender, EventArgs e)

{

rgbChanges();

}

private void bluebar\_ValueChanged(object sender, EventArgs e)

{

rgbChanges();

}

private void openToolStripMenuItem\_Click(object sender, EventArgs e)

{

DialogResult dr = openFileDialog1.ShowDialog();

if (dr == DialogResult.OK)

{

file = Image.FromFile(openFileDialog1.FileName);

pictureBox1.Image = file;

opened = true;

}

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (opened)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Images|\*.png;\*.bmp;\*.jpg";

ImageFormat format = ImageFormat.Png;

if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string ext = Path.GetExtension(sfd.FileName);

switch (ext)

{

case ".jpg":

format = ImageFormat.Jpeg;

break;

case ".bmp":

format = ImageFormat.Bmp;

break;

}

pictureBox1.Image.Save(sfd.FileName, format);

}

}

else

{

MessageBox.Show("Зображення не завантажено, спочатку завантажте зображення");

}

}

}

}