МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного додатку для обробки графічної інформації

Виконав студент 4 курсу групи К-25
Спеціальність 122 комп'ютерні науки
Дерев'янко Михайло Валерійович
Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.т.н., доцент
Великодний Станіслав Сергійович

Рецензент к.ф.-м.н., доц.
Ткач Тетяна Борисівна

Одеса 2020

ЗМІСТ

Скорочення та умовні познаки 5
Вступ 7
1 Аналіз предметної області 9
1.1 Опис предметної області 9
1.2 Характеристика об’єкту розробки 10
1.3 Аналіз існуючих аналогів 10
1.4 Інструментальні засоби розробки 15
1.5 Вимоги до об’єкту розробки 19
2 Проєктування об’єкту розробки 20
2.1 Кольорові моделі для представлення зображення 20
2.2 Корекція яскравості і контрастності зображень 23
2.3 Метод Гауса для фільтрації зображень 24
2.4 Проектування системи UML-засобами 27
3 Реалізація об’єкту розробки 30
3.1 Опис інтерфейсу користувача 30
3.2 Опис основних методів в програмному коді 33
3.2.1 Налаштування рівня різкості зображення 33
3.2.2 Керування рівнем «Яскравість» 34
3.2.3 Клас FCColor з основними методами роботи над зображенням 35
3.2.4 Кольоровий тон «Color Tone» 43
3.2.5 Реалізація фільтру «Black&White» 45
3.2.6 Реалізація фільтру «Розмивання Гауса» 45
Висновки 52
Перелік джерел посилання 53

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ООП – об’єктно-орієнтоване програмування

BMP – Bitmap Picture

CMYK – Cyan Magenta Yellow Key

GIF – Graphics Interchange Forma

HLS – Hue Lightness Saturation

HSB – Hue Saturation Brightness

HSV – Hue Saturation Value

JPEG – Joint Photographic Experts Group

PNG – Portable Network Graphics

RGB – Red Green Blue

SVG – Scalable Vector Graphics

TIFF – Tagged Image File Format

UML – Unified Modeling Language

Гаджет – невеликий пристрій, призначене для полегшення і вдосконалення життя людини

Емодзі – особлива мова ідеограм і смайлів

Actor – актор на діаграмі варіантів використання

Additive – адитивна кольорова модель

Adobe Photoshop – графічний редактор

C – мова програмування

C++ – мова програмування

JPG – формат графічного зображення

Microsoft Office – офісний пакет

Movavi Photo Editor – графічний редактор

PicsArt – мобільний додаток

Precompiled header – механізм прекомпіляції заголовних файлів

The GIMP – графічний редактор

TGA – растровий графічний формат

True Color – спосіб кодування кольору – 24-розрядний

Use Case – варіант використання на UML-діаграмі

Visual Studio 2017 – середовище візуальної розробки

#define – директива C/C++ для об’яви констант

#include – директива C/C++ для підключення файлів

ВСТУП

Комп'ютерна графіка охоплює всі види і форми представлення зображень, доступних для сприйняття людиною або на екрані монітора, або у вигляді копії на зовнішньому носії (папір, кіноплівка, тканина і інше).

Без комп'ютерної графіки неможливо уявити собі не тільки комп'ютерний, але і звичайний, цілком матеріальний світ. На сьогоднішній день комп'ютери і комп'ютерна графіка невід'ємна частина життя сучасного суспільства. Рекламні щити, кольорові журнали, спец ефекти у фільмах – все це в тій чи іншій мірі має ставлення до комп'ютерної графіки. Тому створені програми для створення і редагування зображень, тобто графічні редактори.

На даний момент, існує досить велика кількість графічних редакторів, як досить відомих, так і не дуже поширених. Впродовж останнього десятиріччя, в століття смартфонів і інших гаджетів, фотографія є невід'ємною частиною користувачів цих пристроїв. Кожен користувач прагне зробити фото унікальним, і тому особливу увагу заслуговує особливий вид графічний редакторів – фоторедактори.

Фоторедакторами можуть бути як графічні редактори, так і спеціалізовані додатки, наприклад для видалення ефекту червоних очей, або коригування контрасту. Головною особливістю спеціалізованих додатків є те, що для роботи з ними не треба бути професійним фотографом або дизайнером, всього в кілька простих дій користувачі можуть створити унікальний шедевр з будь – якого зображення.

Темою дипломного проекту є розробка програмного додатку для обробки графічної інформації. Цей редактор повинен забезпечити можливість обробки зображення більш ніж 15 різними ефектами, на вибір користувача, з різним рівнем накладення, а також можливістю поєднувати результати роботи декількох фільтрів для одного зображення.

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 53;

- кількість рисунків – 14;
- кількість таблиць – 0;
- кількість посилань – 17.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Комп'ютерна графіка являє собою одну з сучасних технологій створення різних зображень за допомогою апаратних і програмних засобів комп'ютера, відображення їх на екрані монітора і потім збереження в файлі або друку на принтері [1][1)].

Без комп'ютерної графіки неможливо уявити собі не тільки комп'ютерний, але і матеріальний світ. Візуалізація даних знаходить застосування в самих різних сферах людської діяльності. Наприклад: медицина (комп'ютерна томографія), наукові дослідження (візуалізація будови речовини, векторних полів), моделювання тканин та одягу, дослідно – конструкторські розробки.

Поняття комп'ютерної графіки включає в себе наступні основні поняття:

- роздільна здатність екрану;
- дозвіл принтера;
- дозвіл зображення;
- фізичний розмір зображення може вимірюватися як в пікселях, так і в одиницях довжини. Він створюється при створенні зображення і зберігається разом з файлом;
- кольорове вирішення – визначає метод кодування колірної і інформації, і від нього залежить те, скільки квітів на екрані може відображатися одночасно;
- колірна модель – це спосіб поділу колірного відтінку на складові компоненти. Існує багато різних типів колірних моделей, але в комп'ютерній графіці, як правило, застосовується не більше трьох (RGB, CMYK, HSB);

[1)] [1] Комп'ютерна графіка та її види. URL: https://sites.google.com/site/informatika324/komp-uterna-grafika-ta-ieie-vidi. (дата звернення 10.03.2020).

– колірна палітра – таблиця даних, в якій зберігається інформація про те, яким кодом закодований той чи інший колір. Найзручніший для комп'ютера спосіб кодування кольору – 24- розрядний, True Color.

Додатки комп'ютерної графіки дуже різноманітні. Для кожного напряму створюється спеціальне програмне забезпечення, яке називається графічними програмами, або графічним пакетом [2][1)].

1.2 Характеристика об’єкту розробки

Об’єктом розробки є програмний додаток для обробки графічної інформації. Для вхідних даних користувачу буде представлено більш десятка різноманітних авторських фільтрів для отримання більш оригінальних та привабливих фотографій. Додаток можна буде використовувати як на комп’ютері, так і через смартфон.

Програмний додаток буде підтримувати такі формати даних, як:

- Jpg;
- Jpeg;
- Png;
- Gif;
- Bmp.

1.3 Аналіз існуючих аналогів

Для постановки задачі на першому етапі роботи необхідно розглянути декілька аналогів для аналізу їх від’ємностей і недоліків. Це допоможе більш чітко сформувати основні вимоги до об’єкту розробки.

[1)] [2] Комп’ютерна графіка. URL: http://pz.ptngu.com/lectures/lectures/lecture_13.html. (дата звернення 10.03.2020).

На етапі огляду програмних засобів обробки растрових зображень і фотографій слід розглянути Adobe Photoshop (рис. 1):

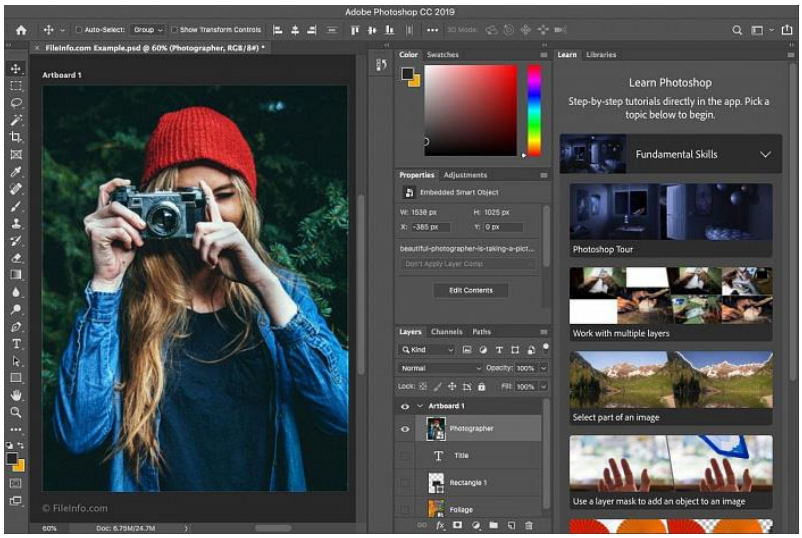

Рисунок 1 – Скріншот робочого вікна Adobe Photoshop

Він є одним з найпопулярніших і просунутих засобів обробки зображень і фотографій на сьогоднішній день. Програмне забезпечення здатне задовольнити як запити професійних фотографів, так і потреби любителів. Для вас надано масштабний спектр палітр, текстур і спецефектів. Обробка фотографії високої якості гарантована [3][1)].

Ключові можливості Photoshop:

– створення та редагування ексклюзивного контенту;
– усунення ефекту червоних очей і фонових похибок;
– графічні композиції можна формувати з використанням шарів;
– додавання шумів, розмиття, текстової інформації і так далі;
– функція колірної автокорекції;
– режими градації сірого, RGB і CMYK каналів;

[1)] [3] Лучшие графические редакторы: Топ-20. URL: https://www.canva.com/ru_ru/obuchenie/graficheskij-redaktor-20/. (дата звернення 12.03.2020).

– відправлення контенту на друк;

– створення растрових зображень;

– збереження графічних файлів в будь – якому необхідному форматі.

До недоліків слід віднести об’єм ресурсів комп’ютера для його розміщення та труднощі у роботи користувача без стартових навичок роботи з цим програмним забезпеченням.

Наступний аналог – Movavi Photo Editor (рис. 2):

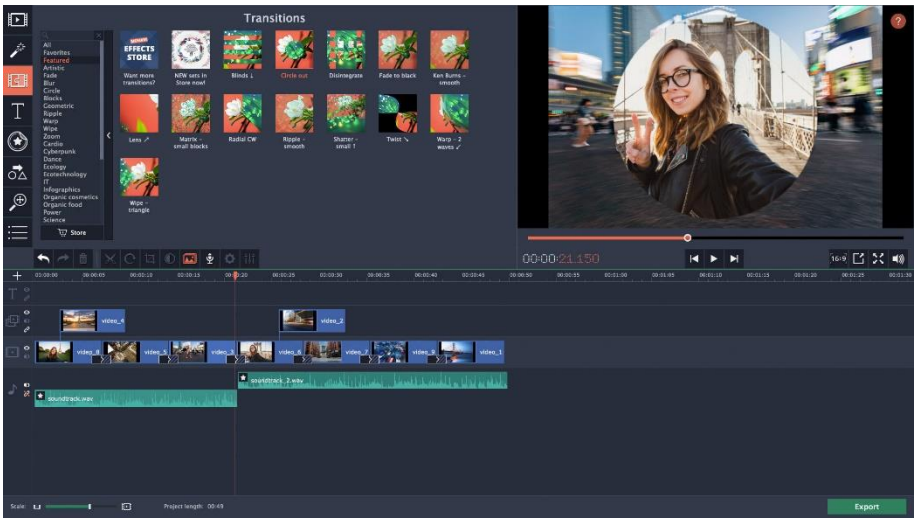

Рисунок 2 – Скріншот робочого вікна Movavi Photo Editor

Незважаючи на всі об'єктивні переваги Photoshop перед аналогами, він залишається суто професійним інструментом із завидним функціоналом, але складним управлінням. Movavi Photo Editor не поступається дітищу Adobe за можливостями, однак має простий, дружній інтерфейс, адаптований для непрофесіоналів. Складні дії (видалення об'єктів, заміна фону та ін.). Колірокорекція, обрізка, накладення ефектів і інші тонкі настройки проводяться буквально в кілька кліків [3][1)].

[1)] [3] Лучшие графические редакторы: Топ-20. URL: https://www.canva.com/ru_ru/obuchenie/graficheskij-redaktor-20/. (дата звернення 10.03.2020).

Головні переваги Movavi Photo Editor:

– всі функції професійного редактора в доступному виконанні;
– просунута робота з колірними профілями;
– проста реалізація складних операцій (видалення елементів, фону);
– величезна кількість фільтрів і ефектів;
– відмінний інструмент ретуші для обличчя і тіла на фото;
– висока швидкість, низькі вимоги до системи;
– можливість повністю оцінити програму завдяки грамотній системі апробації.

Наступним обєктом аналізу є The GIMP (рис. 3). Безкоштовний графічний редактор The GIMP вважається аналогом більш популярного Photoshop, утиліта містить всі необхідні функції обробки цифрових зображень і має російськомовний інтерфейс, який можна налаштовувати з урахуванням ваших уподобань [4][1)].

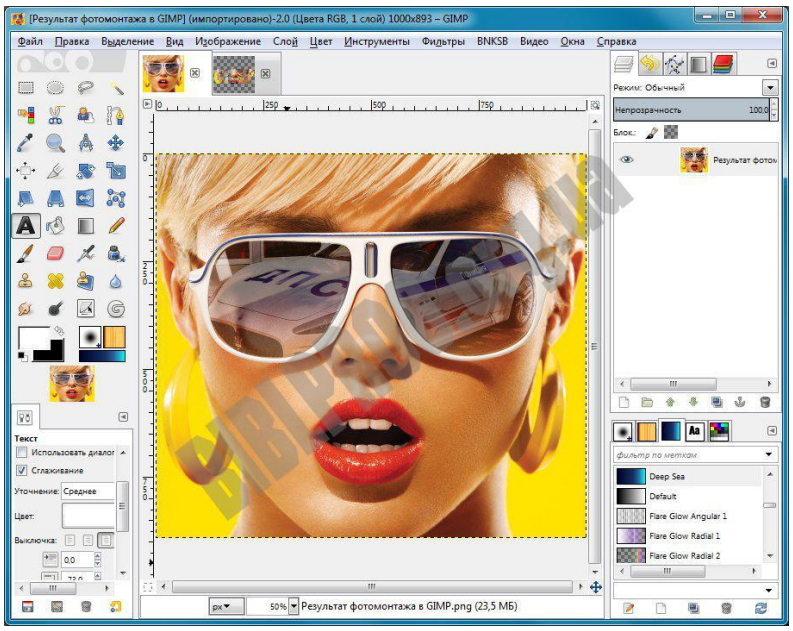

Рисунок 3 – Скріншот робочого вікна The GIMP

[1)] [4] Бесплатные графические редакторы. URL: https://remontka.pro/besplatnye-graficheskie-redaktory/. (дата звернення 14.03.2020).

Поширений в області ретушування контенту і при формуванні різноманітних логотипів, а також дизайнерських елементів аматорського рівня для web-сайтів.

Ключові особливості The GIMP:

– налаштування параметрів яскравості, контрастності, прозорості, стилю і кольору використовуваних кистей і олівців;
– підтримка графічних планшетів від різних виробників;
– формування лого і дизайну сайтів;
– змінні RGB – канали, робота з шарами;
– відкриває і конвертує файли в форматах JPEG, GIF, BMP, PNG, TIFF, TGA, SVG і так далі;
– відрізняється великою кількістю спеціальних ефектів, інструментів і фільтрів;
– допускається відкриття вікон інтерфейсу в окремих вкладках.

Останнім аналогом розглянемо популярний мобільний додаток PicsArt (рис. 4):

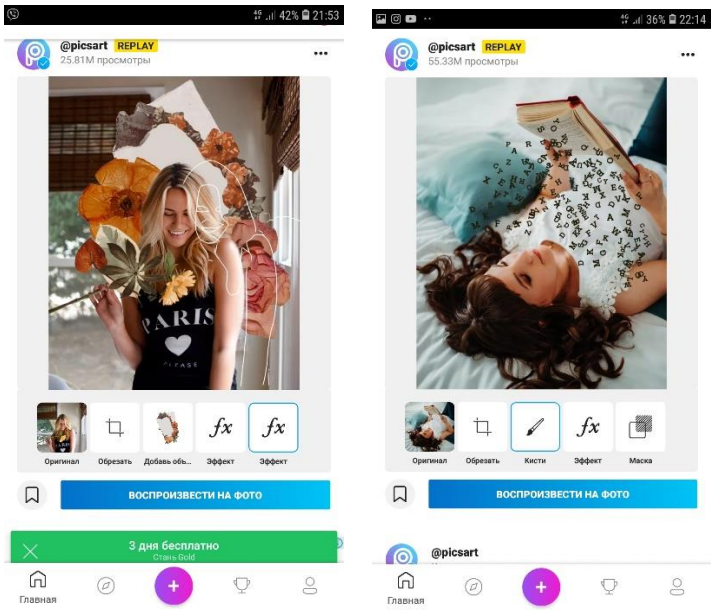

Рисунок 4 – Скріншот робочого вікна PicsArt

PicsArt – це безкоштовний мобільний фото редактор №1, легкий у використанні, більше, ніж просто накладення фільтрів. Після встановлення користувач отримує:

- безліч інструментів, ефектів, безкоштовних стікерів, а також мільйонів стікерів від інших користувачів, інструментів для малювання, створення колажу і селфі;
- також у розпорядженні користувача будуть маски, рамки, шпалери, емодзі. PicsArt допомогає у створенні приголомшливих зображень, у відкритті нових технік.

У креативний набір PicsArt входять творець для колажу, інструменти для малювання, фото редактор, камера і багато іншого [5][1)].

Програма включає інструменти для вирізання фігур, кадрування, розтягування, клонування, додавання тексту і роботи з кривими. Також воно має величезну бібліотеку з художніми фото фільтрами, включаючи HDR, рамками, шпалерами, виносками і не тільки. Всі інструменти мають режим настройки кистей і їх товщини для вибіркового застосування в бажаної області зображення [6][2)].

1.4 Інструментальні засоби розробки

1.4.1 Мова програмування

Основною мовою для розробки було обрано мову C++. Це компільована статично типізована мова програмування загального призначення. Підтримує різні парадигми програмування, але, в порівнянні з його попередником – мовою C, – найбільшу увагу приділено підтримці об'єктно – орієнтованого і узагальненого програмування.

[1)] [5] PicsArt – описание. URL: https://aminoapps.com/c/myphonerus/page/item/picsart-opisanie/ER8z_6BliLID5DN85gj53Lbx8l26LlGaG. (дата звернення 14.03.2020).

[2)] [6] PicsArt. URL: https://viberfun.ru/multimedia/113-picsart.html. (дата звернення 11.03.2020).

Слід розглянути переваги і недоліки мови, щоб зрозуміти, що робить його настільки потужним і універсальним інструментом в руках програміста. Перш за все, необхідно підкреслити, що оцінювати від'ємності і, особливо, недоліки C++ необхідно в контексті тих принципів, на яких будувалася мова, і вимог, які до неї спочатку висувалися.

C++ – надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт і драйверів до складних програмних комплексів самого різного призначення. Зокрема, підтримуються різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси) [7][1)].

Передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Весь код, неявно генерується компілятором для реалізації мовних можливостей. Макроси (#define) є потужним, але небезпечним засобом. Вони збережені в C++ незважаючи на те, що необхідність в них, завдяки шаблонам і вбудованим функціям, не так вже й велика. У успадкованих стандартних Сі-бібліотеках багато потенційно небезпечних макросів [8][2)].

Препроцесор, успадкований від Сі, дуже примітивний. Це призводить з одного боку до того, що з його допомогою можна (або важко) здійснювати деякі завдання метапрограмування, а з іншого, внаслідок своєї примітивності, він часто приводить до помилок і вимагає багато дій з обходу потенційних проблем.

Погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовки (#include) серйозно уповільнює компіляцію при підключенні великої кількості модулів (бо результуючий файл, який обробляється компілятором, виявляється дуже великий). Ця схема

[1)] [7] Почему C++ крут, актуален и бессмертен. URL: https://vc.ru/hr/50161-pochemu-c-krut-aktualen-i-bessmerten. (дата звернення 11.03.2020).

[2)] [8] Рейтинг востребованности языков программирования. URL: http://www.tadviser.ru/index.php/Статья:Рейтинг_востребованности_языков_программирования. (дата звернення 11.03.2020).

без змін скопійована в C++. Для усунення цього недоліку, багато компілятори реалізують механізм прекомпіляції заголовних файлів (англ. Precompiled header) [8][1)].

До власних недоліків C++ можна віднести складність і надмірність, через які C++ важко вивчати, а побудова компілятора пов'язане з великою кількістю проблем. Деякі вважають недоліком мови C++ відсутність вбудованої системи збірки сміття.

Переваги мови C ++:

- масштабованість. Мовою C++ розробляють програми для найрізноманітніших платформ і систем;
- можливість роботи на низькому рівні з пам'яттю, адресами, портами. Що, при необережному використанні, може легко перетворитися на недолік;
- можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація, і обчислення на етапі компіляції, використовуючи шаблони.

1.4.2 Вибір середи розробки

При розробці і реалізації програмного продукту було використано середовище візуальної розробки Visual Studio 2017 (рис. 5):

Архітектура операційної системи захищає програми від ушкодження одна одної і самою операційною системою. При цьому використовується відмовостійка структурована обробка особливих ситуацій на всіх архітектурних рівнях, включаючи відновлювану файлову систему NTFS і забезпечує захист за допомогою вбудованої системи безпеки і вдосконалених методів управління пам'яттю.

[1)] [8] Рейтинг востребованности языков программирования. URL: http://www.tadviser.ru/index.php/Статья:Рейтинг_востребованности_языков_программирования. (дата звернення 11.03.2020).

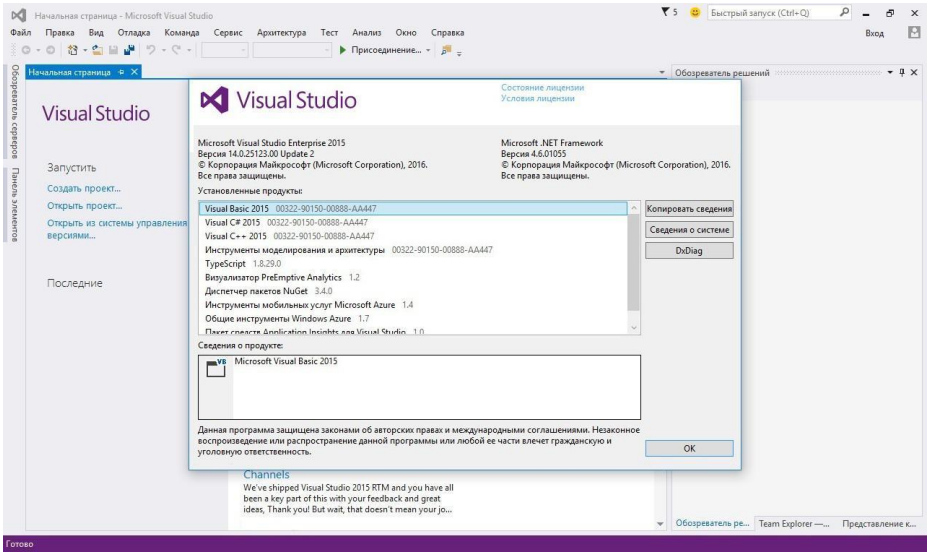

Рисунок 5 – Середа розробки Microsoft Visual Studio

Система надає можливості для роботи в багатьох країнах світу на національних мовах, досягається за допомогою застосування стандарту ISO Unicode. Для розробки програмного продукту було вибрано програмне середовище Visual Studio 2017 з багатьох причин.

По-перше, Visual Studio 2017 є інтегрованим середовищем швидкої розробки програмного забезпечення для роботи під Microsoft Windows, на даний момент найпоширенішою операційною системою [9][1)].

Середовище програмування Visual Studio 2017 року було обрано завдяки можливості безпроблемного зв'язку з офісним пакетом Microsoft Office.

По-третє, Visual Studio 2017 року було обрано через доступності цієї програми будь-якому користувачеві. Досить проста установка зв'язку з цим пояснюється тим, як уже зазначалося вище, що навколишнє середовище Visual Studio 2017 – це один з найвідоміших засобів створення Windows-додатків і зв'язку з продукцією Microsoft встановлюються досить легко, Visual Studio

1) [9] Вышла Visual Studio 2017: рассказываем о новых возможностях инструментов от Microsoft. URL: https://tproger.ru/news/vs-2017/. (дата звернення 16.03.2020).

2017 відзначається своєю стабільністю, швидкістю і низькими вимогами до апаратного забезпечення комп'ютера [10][1)].

1.5 Вимоги до об'єкту розробки

Об'єктом розробки є програмний додаток для обробки графічної інформації. Цей редактор повинен забезпечити можливість обробки зображення більш ніж 15 різними ефектами, на вибір користувача, з різним рівнем накладенням.

До основних фільтрів обов'язково слід включити:

- методи конвертації кольорових моделей;
- класичні методи зображення градації сірого відтінку;
- переходи щодо рівня яскравості і т.п.;
- додаткові методи обробки зображень.

[1)] [10] Знакомство с Visual Studio. URL: https://habr.com/ru/sandbox/79099/ . (дата звернення 16.03.2020).

2 ПРОЄКТУВАННЯ ОБ'ЄКТУ РОЗРОБКИ

2.1 Кольорові моделі для представлення зображення

Колірна модель RGB. В основі однієї з найбільш поширених колірних моделей, званої RGB моделлю, лежить відтворення будь-якого кольору шляхом додавання трьох основних кольорів: червоного (Red), зеленого (Green) і синього (Blue). Кожен канал – R, G або B є свій окремий параметр, який вказує на кількість відповідної компоненти в кінцевому кольорі. На приклад: (255, 64, 23) – колір, що містить сильний червоний компонент, трохи зеленого і зовсім небагато синього.

Природно, що цей режим найбільш підходить для передачі багатства фарб навколишньої природи. Але він вимагає і великих витрат, так як глибина кольору тут найбільша – 3 канали по 8 біт на кожен, що дає в цілому 24 біта.

Оскільки в RGB моделі відбувається складання квітів, то вона називається адитивною (additive). Саме на такій моделі побудовано відтворення кольору сучасними моніторами [11][1)].

Колірним простором RGB моделі є одиничний куб (рис. 6).

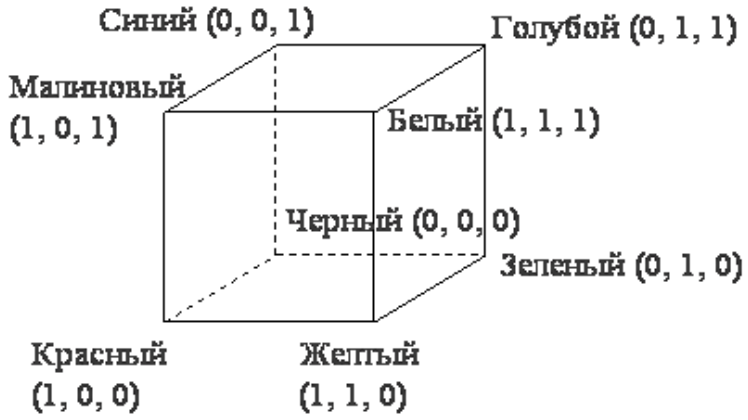

Рисунок 6 – Кольоровий простір RGB-моделі

[1)] [11] Цветовые модели. URL: http://compgraph.tpu.ru/Colors_models.htm. (дата звернення 18.03.2020).

Кольорові моделі HSV і HLS орієнтовані на роботу з кольоро- передаючою апаратурою і для деяких людей незручні. Вони спираються на інтуїтивні поняття тону насиченості і яскравості.

У колірному просторі моделі HSV (Hue, Saturation, Value), іноді званої HSB (Hue, Saturation, Brightness), використовується циліндрична система координат, а безліч допустимих кольорів є шестигранний конус, поставлений на вершину (рис. 7).

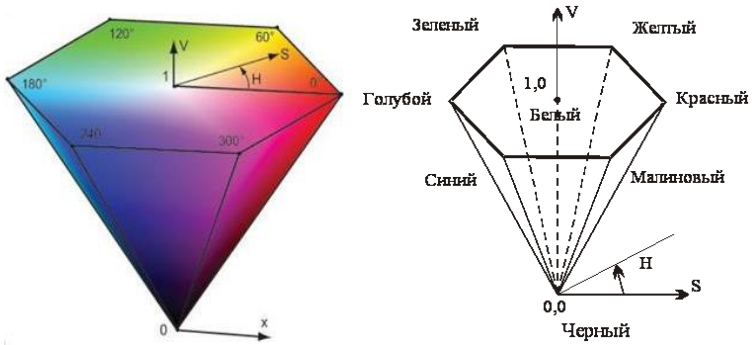

Рисунок 7 – Кольоровий мпростіп HSV моделі

Підстава конуса являє яскраві кольори і відповідає V = 1. Однак кольору підстави V = 1 не мають однакової сприймається інтенсивності. Тон (H) вимірюється кутом, відлічуваним навколо вертикальної осі OV. При цьому червоному кольору відповідає кут 0°, зеленому – кут 120° і т. Д. Кольори, взаємно доповнюють один одного до білого, знаходяться навпроти один одного, т. Е. Їх тони відрізняються на 180°. Величина S змінюється від 0 на осі OV до 1 на гранях конуса.

Конус має одиничну висоту (V = 1) і підстава, розташоване на початку координат. В основі конуса величини H і S сенсу не мають. Білому кольору відповідає пара S = 1, V = 1. Вісь OV (S = 0) відповідає ахроматичних квітів (сірих тонів) [11][1)].

[1)] [11] Цветовые модели. URL: http://compgraph.tpu.ru/Colors_models.htm. (дата звернення 18.03.2020).

Процес додавання білого кольору до заданого можна представити як зменшення насиченості S, а процес додавання чорного кольору – як зменшення яскравості V. Заснуванню шестигранного конуса відповідає проекція RGB куба вздовж його головної діагоналі.

Ще одним прикладом системи, побудованої на інтуїтивних поняттях тони насиченості і яскравості, є система HLS (Hue, Lightness, Saturation). Тут безліч всіх кольорів є два шестигранні конуса (рис. 8), поставлених один на одного (підстава до основи).

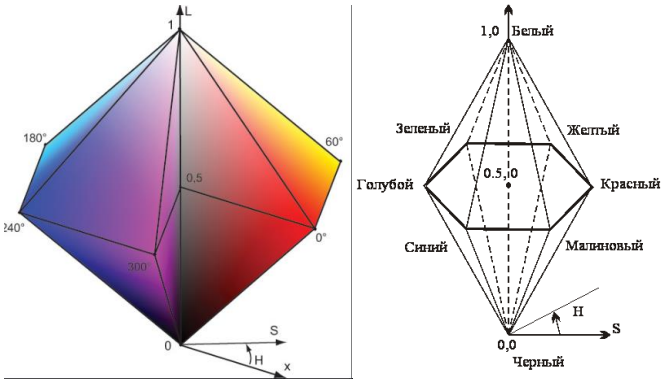

Рисунок 8 – Кольоровий мпростіп HLS моделі

Порівняння моделей HSL і HSV:

- насиченість в моделі HSL завжди змінюється від повністю насиченого кольору до еквівалентного сірому кольору, в той час як в моделі HSV при V = 1 повністю насичений колір переходить до білого;
- освітлення в моделі HSL змінюється від чорного через вибране значення кольоровості – до білого, а в моделі HSV – проходить лише половину шляху – від чорного до вибраного кольоровому [12][1)].

[1)] [12] Цветовые модели HSV и HLS. URL: https://www.intuit.ru/studies/courses/70/70/lecture/2094?page=4. (дата звернення 20.03.2020).

2.2 Корекція яскравості і контрастності зображень

Зображення часто є мало контрастними. Слабкий контраст, як правило, обумовлений широким діапазоном відтворюваних яскравостей, нерідко поєднується з не лінійністю характеристики передачі рівнів. Характер залежності зміни яскравості палітри пікселів від мінімального значення до максимального також впливає на якість зображення. Оптимальною є лінійна функція зміни інтенсивності пікселів.

При увігнутою характеристиці зображення буде більш темним, при опуклою – більш світлим. І в тому, і в іншому випадку ознаки об'єктів можуть бути перекручені і недостатньо добре ідентифіковані. Корекція (лінеаризація) яскравості палітри істотно покращує якість зображення.

Мала контрастність може бути обумовлена і тим, що варіації функції яскравості пікселів на зображенні набагато менше допустимого діапазону шкали яркостей. В цьому випадку контрастність зображення підвищується шляхом "розтягування" реального динамічного діапазону яркостей на всю шкалу за допомогою лінійного по елементного перетворення.

Інший спосіб корекції яскравості палітри пов'язаний з інверсією вхідного зображення. Оскільки розрізняти слабкі сигнали на темному тлі досить складно, то інверсна форма подання таких зображень має іншу гістограму яскравості, більш прийнятну для спостереження і візуальної ідентифікації.

Деякі завдання обробки зображення пов'язані з перетворенням півтонування (багато градацій яскравості) в бінарне (дві градації). Перетворення здійснюється для того, щоб скоротити інформаційну надмірність зображення, залишити в ньому тільки інформацію, яка потрібна для вирішення конкретного завдання [13][1)].

[1)] [13] Корекція фотографій у Photoshop. Тонова корекція фотографій. URL: http://www.ukr.vkadre.kiev.ua/photographers/foto-text/tonovaya-korrekcia.html. (дата звернення 23.03.2020).

У бінарному зображенні повинні бути збережені певні деталі (наприклад, обриси зображених об'єктів) і виключені несуттєві особливості (фон). Порогова обробка півтонування полягає в поділі всіх елементів зображення на два класи A1 і A2 за ознакою яскравості з кордоном Agr, I у виконанні відповідної порогової фільтрації з заміною пікселів зображення на встановлену яскравість класів.

Вибір кордону визначається видом гістограми яскравості вихідного зображення. Для найпростіших зображень типу креслень, машинописного тексту і т.п., що мають бімодальне розподіл, межа встановлюється по мінімуму між модами розподілу.

У загальному випадку зображення може бути многомодальним, і якщо встановлюється досить надійне відповідність між об'єктами і відповідними модами їх яскравості, то порогова фільтрація також може передбачати кілька класів яскравості пікселів [13][1)].

2.3 Метод Гауса для фільтрації зображень

«Розмивання Гауса» – це метод фільтрації зображення за допомогою функції Гауса, який призводить до розмивання графічної інформації. Даний ефект широко використовується в графічних програмах, як правило, для зменшення зашумленості зображенні та зниження деталізації. Візуальний ефект цієї фільтрації розмивання аналогічний погляду на зображення крізь напівпрозорий екран [14][2)].

«Розмивання Гауса» – це тип фільтру розмивання зображення, що використовує функцію Гауса (яка також зустрічається у нормальному розподілі в

[1)] [13] Корекція фотографій у Photoshop. Тонова корекція фотографій. URL: http://www.ukr.vkadre.kiev.ua/photographers/foto-text/tonovaya-korrekcia.html. (дата звернення 23.03.2020).

[2)] [14] Коригування різкості та розмиття зображення. URL: https://helpx.adobe.com/ua/photoshop/using/adjusting-image-sharpness-blur.html. (дата звернення 23.03.2020).

області статистики) для розрахунку трансформації кожного пікселя у зображенні.

Рівняння функції Гауса в одному вимірі:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

Для двовимірного випадку, вираз складається з двох таких функцій, по одній для кожної осі виміру:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

де x – це відстань від початку координат в осі абсцис, y – це відстань від початку координат у осі ординат, а σ є стандартне відхилення розподілу Гауса. Коли метод застосовується у двох вимірах, отримується поверхня, контури якої є концентричні кола розподілу Гауса з центральної точки. Значення з цього розподілу використовуються для створення матриці згортки. Для кожного нового значення пікселя визначається середнє зважене в околі пікселя.

Значення поточного оригінального пікселя має більшу вагу (найвище значення розподілу Гауса), а сусідні пікселі отримують все меншу вагу в залежності від того наскільки далеко вони знаходяться від поточного оригінального пікселя. Це надає ефект розмитості, яка зберігає кордони та краї краще, ніж інші, аналогічні фільтри розмиття [14][1)].

Теоретично, в кожній точці зображення буде відмінним від нуля результат функції Гауса, це означає, що для розрахунку для кожного пікселя необхідно брати значення усіх пікселів у зображенні. На практиці, при обчисленні дискретного спрощення функції Гауса, вага пікселів на відстані більш ніж 3σ

[1)] [14] Коригування різкості та розмиття зображення. URL: https://helpx.adobe.com/ua/photoshop/using/adjusting-image-sharpness-blur.html. (дата звернення 23.03.2020).

достатньо мала, щоб мати якісний вплив на середнє зважене значення і вважається нулем. Значення пікселів, що розташовані за межами цього діапазону можуть бути проігноровані.

Розмивання Гауса може застосовуватися для двовимірних зображень у вигляді двох незалежних одновимірних, так званих лінійно розділених розрахунків.

Тобто, ефект застосування двовимірної матриці може бути досягнуто за рахунок застосування ряду одновимірної матриці Гауса в горизонтальному напрямку, а потім повторно у вертикальному напрямку.

Застосування розмивання Гауса призводить до розмивання на зображення і має ті ж наслідки, що застосування єдиного розмивання Гауса, радіус якого рівний квадратному кореню з суми квадратів радіусу розмиття, що застосовується.

Для скорочення часу обчислень можна скористатися лінійно відокремленим розмиванням Гауса, розділивши цей процес на два етапи. У першому проходженні одновимірна матриця використовується для розмиття зображення тільки в горизонтальному або вертикальному напрямку.

На другому проході матриця використовується для розмиття в іншому напрямку. Отриманий результат відповідає результату з використанням двовимірних матриць, але вимагає меншої кількості обчислень.

Дискретизація, як правило, досягається шляхом відбору для фільтра гаусівських дискретних точок на відповідних позиціях кожного центрального пікселя. Це дозволяє зменшити обчислювальні витрати, але при дуже малих значеннях точки відбору гауссової функції невелика кількість зразків призводить до великої помилки. У цих випадках точність підтримується (при невеликій обчислювальної вартості) шляхом інтеграції гауссових функцій по площі кожного пікселя [13][1)].

[1)] [13] Корекція фотографій у Photoshop. Тонова корекція фотографій. URL: http://www.ukr.vkadre.kiev.ua/photographers/foto-text/tonovaya-korrekcia.html. (дата звернення 23.03.2020).

2.4 Проектування системи UML-засобами

UML (Unified Modeling Language) предназначен для спрощення спілкування і взаімодействія участніков проекта, сокращенія часу на об'ясненіе і усвоеніе інформациі, полегшення документірованія.

UML – графіческая нотація, предназначенная для опісанія і моделірованія процесс, протекающіх в ході разработкі. Діаграмми данной нотациі разлічаются по типами і опісивают разлічние аспекти разработкі. Розрізняють 2 основні типа UML-діаграмм: структурні і поведенческіе.

Структурні діаграми відображають елементи, з яких складається система. Поведенческі моделі описують процес, який існує у системі [15][1)].

Не можна починати проектування системи відразу з визначення класів і побудови таких діаграм. Щоб зрозуміти і правильно спроектувати майбутню систему, потрібно перш за все визначити, що вона буде робити, тобто, необхідно створення проекції, званої функціональної моделю. Першим кроком при описі функціональності системи є моделювання вимог до неї [16][2)].

Діаграми варіантів використання допомагають нам при визначенні користувальницьких сценаріїв. Вони дозволяють не тільки не пропустити жодного існуючого сценарію, але також позначити взаємодія різних користувачів з системою і виявити сценарії, які є включення і розширенням існуючих.

Таким чином, отримується найбільш повна картину сценаріїв і не пропускається жодної необхідної функції системи.

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання.

---

1) [15] UML-моделирование. URL: https://fingers.by/blog/uml. (дата звернення 30.03.2020).

2) [16] Диаграммы вариантов использования. URL: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4. (дата звернення 30.03.2020).

При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник.

У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою [15][1)].

Мета варіанту використання полягає в тому, щоб визначити закінчений аспект або фрагмент поведінки деякої сутності без розкриття внутрішньої структури цієї сутності. В якості такої сутності може виступати вихідна система або будь-який інший елемент моделі, який володіє власною поведінкою, подібно підсистемі або класу в моделі системи.

Кожен варіант використання відповідає окремому сервісу, який надає моделюючу сутність або систему по запиту користувача (актора), т. Е. Визначає спосіб застосування цієї сутності.

Сервіс, який ініціалізується за запитом користувача, є закінченою послідовність дій. Це означає, що після того як система закінчить обробку запиту користувача, вона повинна повернутися в початковий стан, в якому готова до виконання наступних запитів [17][2)].

Між елементами діаграми Use-Case існують такі відношення, як: залежність, розширення, включення, та асоціація (найчастіше відношення на діаграмі варіантів використання).

Так, для об’єкту розробки слід побудувати діаграму Use-Case, для якої актором буде виступати сам програмний додаток (рис. 9):

[1)] [15] UML-моделирование. URL: https://fingers.by/blog/uml. (дата звернення 18.03.2020).

[2)] [17] UML – універсальна мова моделювання. URL: http://sites.znu.edu.ua/webprog/lect/1238.ukr.html. (дата звернення 19.03.2020).

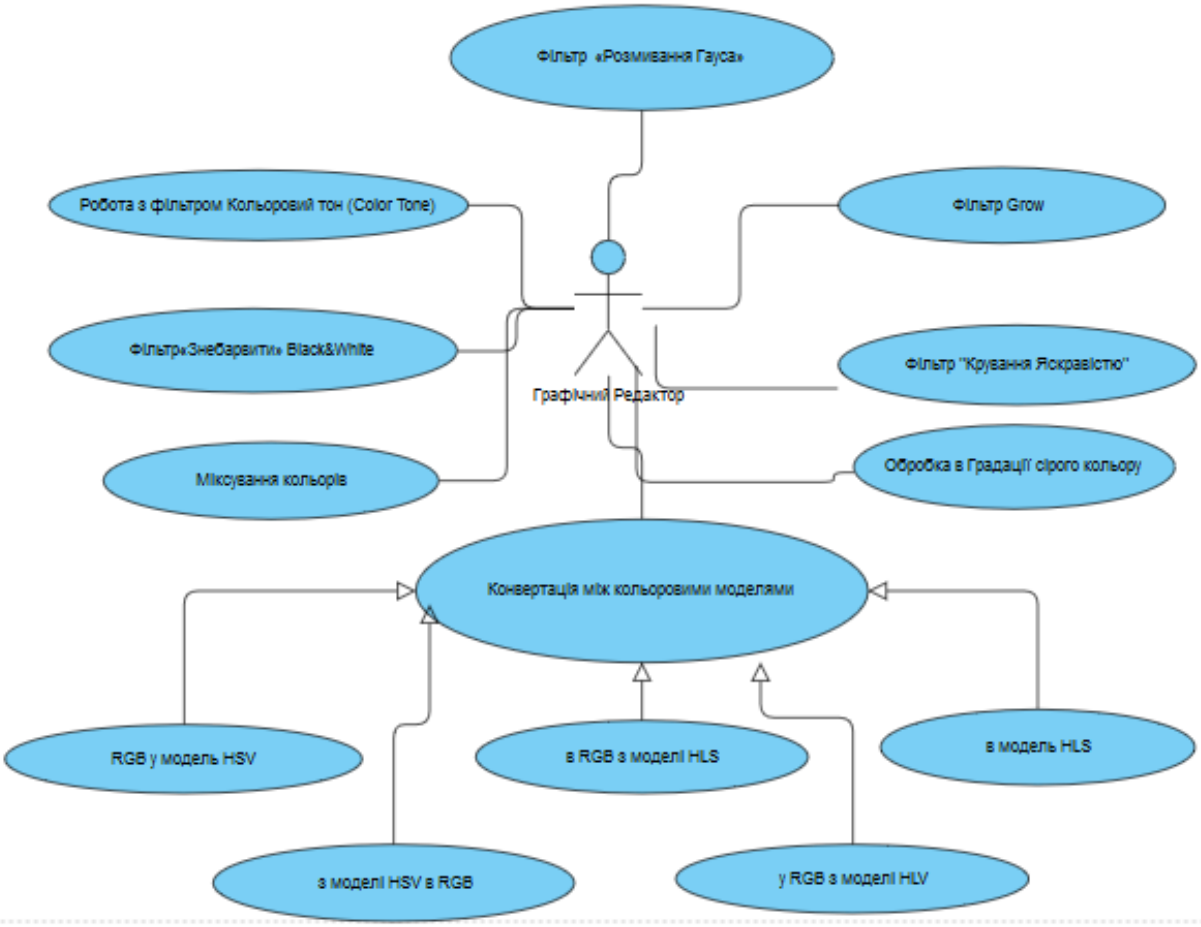

Рисунок 9 – Діаграма Use-Case

3 РЕАЛІЗАЦІЯ ОБ’ЄКТУ РОЗРОБКИ

3.1 Опис інтерфейсу користувача

Після запуску програми на екрані з’являється головне вікно (рис. 10), у верхній частині якого знаходиться головне меню, робоча область зафарбована сірим фоном, а справа від неї знаходиться панель фільтрів.

Внизу головного вікна розміщена панель стану, яка відображає процес виконання операцій.

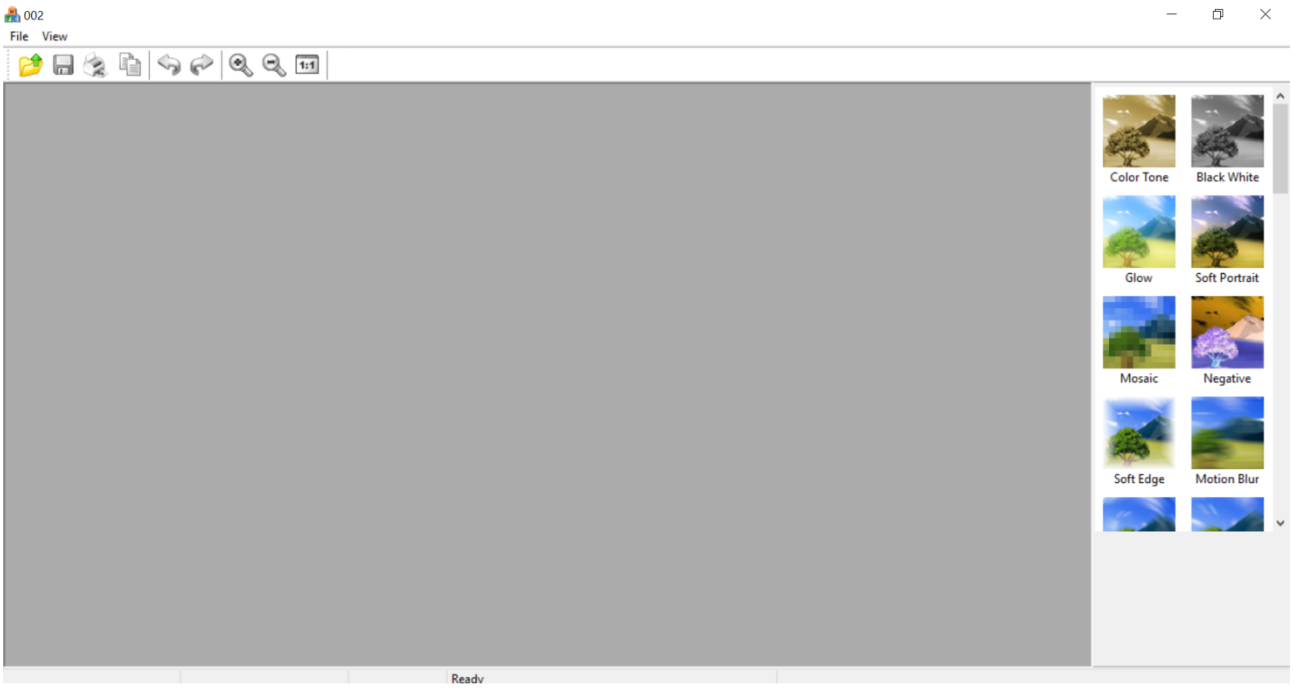

Рисунок 10 – Головне вікно програми

Для того, щоб почати роботу с зображенням його необхідно відкрити, обравши у головному меню пункт «Відкрити» (рис. 11).

Відкрите зображення відобразитися у новому створеному вінку у робочій області. Програма підтримує декілька типів графічних файлів. Тепер для зображення можна обрати фільтр і рівень застосування (рис. 12).

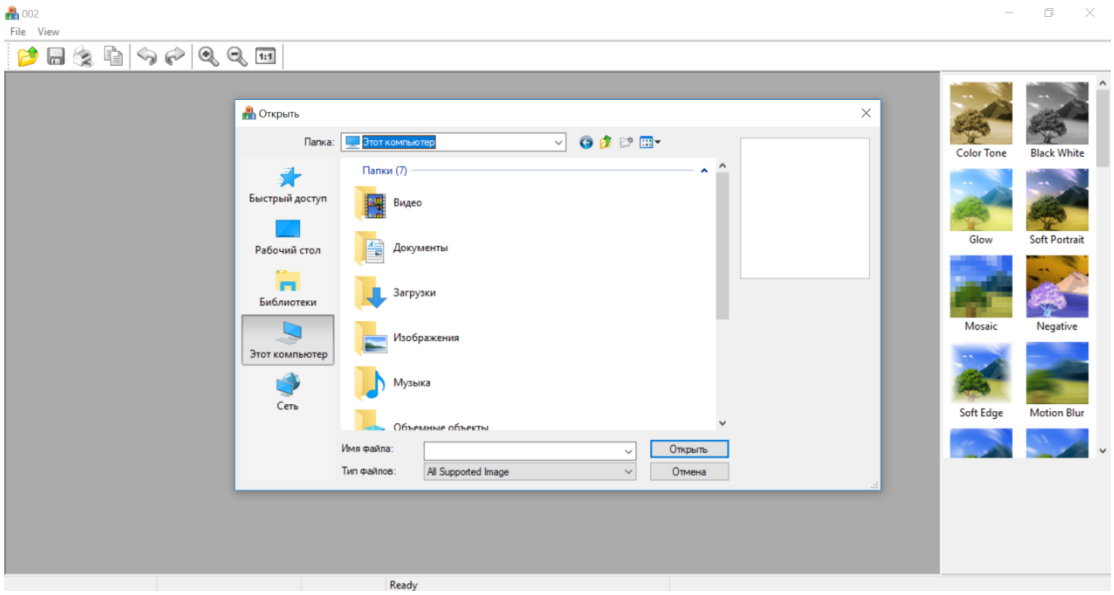

Рисунок 11 – Вікно вибору графічного файлу

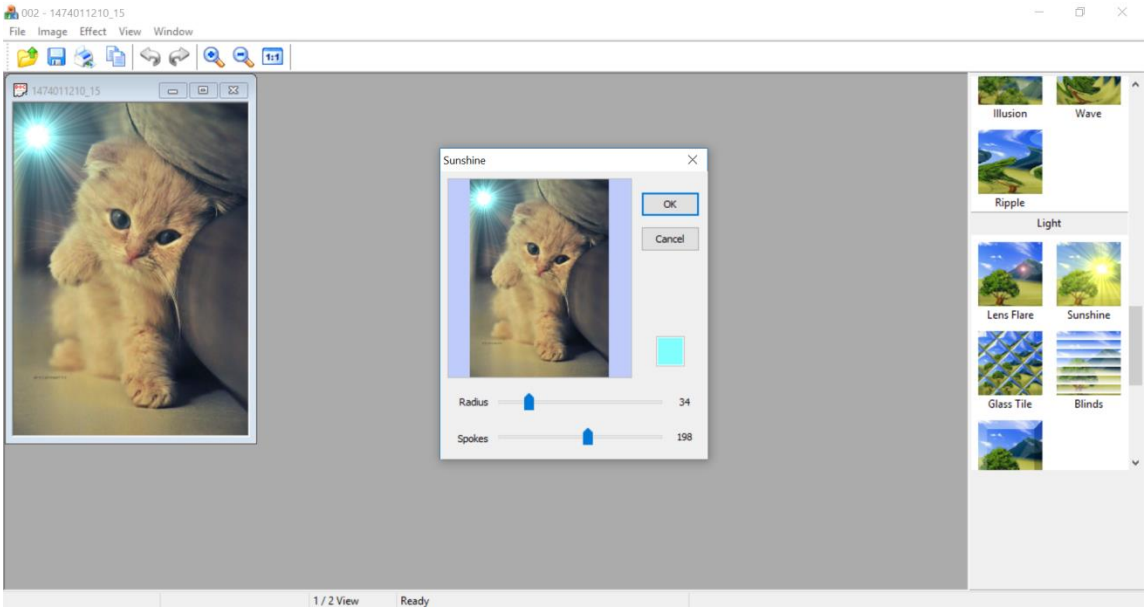

Рисунок 12 – Застосування фільтру до зображення

Розроблена програма дозволяє працювати додатку у багато віконному режимі для зручності обробки зображень (рис. 13). Цей підхід допомогає при створенні декількох зображень з однотипними фільтрами.

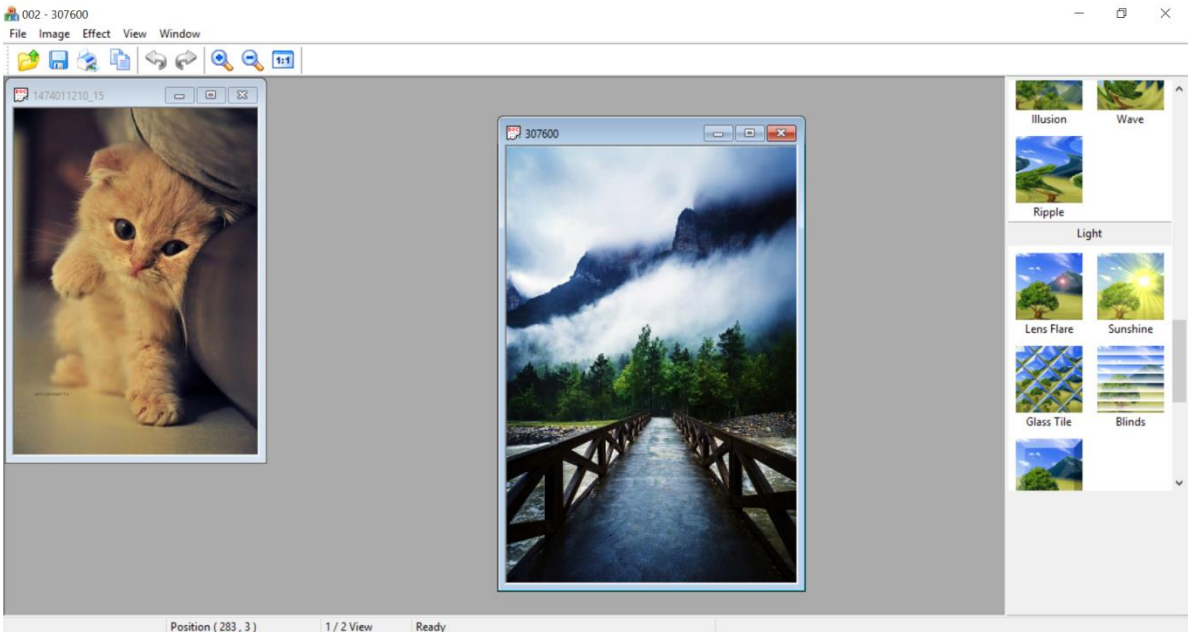

Рисунок 13 – Демонстрація багатовіконной обробки зображень

Оброблене зображення можна зберегти, обравши пункт меню «Зберегти» (рис. 14).

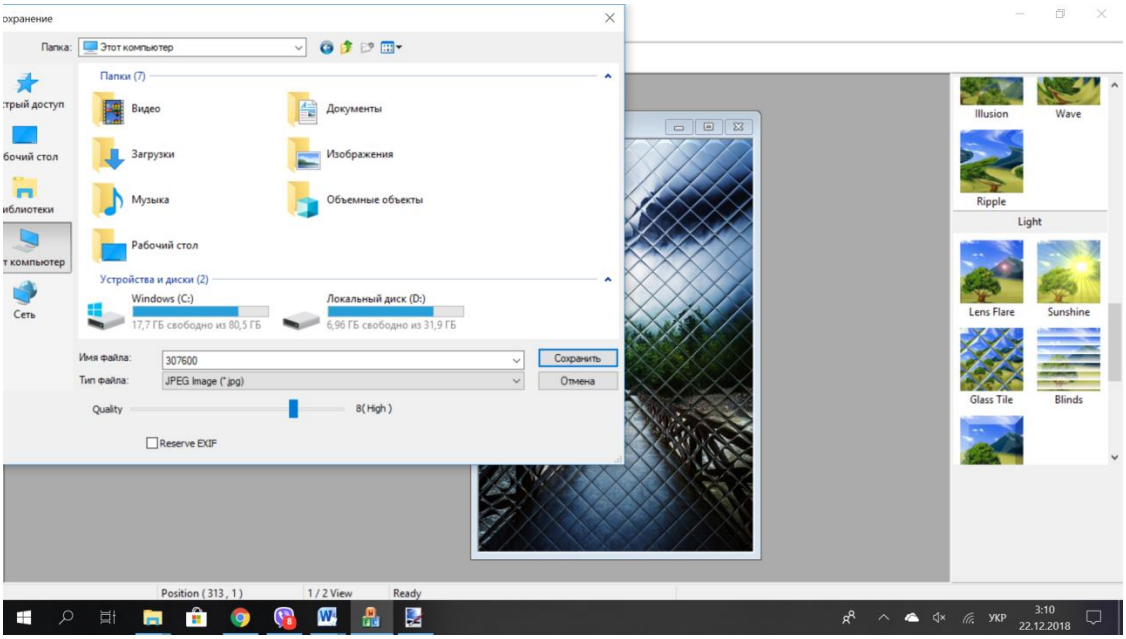

Рисунок 14 – Збереження файлу зображення

3.2 Опис основних методів в програмному коді

3.2.1 Налаштування рівня різкості зображення

Різкість описує розрізнення деталей на фотографії, і вона може використовуватися як важливий творчий інструмент для виділення текстури. Відповідна техніка фотографії та пост-обробки може значно поліпшити різкість, хоча вона безумовно обмежена можливостями вашої камери, збільшенням зображення і дистанцією перегляду. На сприйняту різкість зображення впливають два фундаментальні чинники: дозвіл і чіткість.

Керування різкістю, яскравістю і контрастом можна виконати у фільтрі Glow. Для реалізації фільтру Glow створено клас FCEffectSoftGlow:

```
class FCEffectSoftGlow : public FCEffectBlur_Gauss
{
    FCObjImage   m_bak ;
    std::auto_ptr<FCImageEffect>   m_bc ;
public:
```

Конструктор класу може приймати: nRadius >= 1, -100 <= nBrightness <= 100, -100 <= nContrast <= 100, використовуючи конструктор батьківського класу Гаусового розмиття для керування різкістю:

```
 FCEffectSoftGlow (int nRadius, int nBrightness, int nContrast) :
FCEffectBlur_Gauss(nRadius, true)
    {
        m_bc.reset (new FCEffectBrightnessContrast(nBrightness,
nContrast)) ;
    }
```

Методи застосування змін різкості, яскравості і контрасту для зображення реалізовани наступним чином:

```
private:
    virtual void OnBeforeProcess (FCObjImage& img)
    {
        m_bak = img ;
        FCEffectBlur_Gauss::OnBeforeProcess(img) ;
        m_bc->OnBeforeProcess(img) ;
    }
```

```
    virtual void OnFinishPixel (FCObjImage& img, int x, int y,
BYTE* pPixel)
    {
        m_bc->ProcessPixel (img, x, y, pPixel) ;

        BYTE   * pOld = m_bak.GetBits(x,y) ;
        for (int i=0 ; i < 3 ; i++)
        {
            pPixel[i] = 255 - (255 - pOld[i])*(255 -
pPixel[i])/255 ;
        }
        if (img.ColorBits() == 32)
        {
            PCL_A(pPixel) = PCL_A(pOld) ;
        }
    }
};
```

3.2.2 Керування рівнем «Яскравість»

Діапазон яскравості зображення в комп'ютері може мати відмінності від діапазону яркостей вихідного, наприклад, в силу недостатньої експозиції. Існує два можливих способу корекції яскравості. Відповідно до першого способу зображення лінійно відображається в діапазоні яркостей вихідного.

Другий спосіб передбачає обмеження яскравості пікселів в обробленому зображенні максимальним і мінімальним граничними рівнями, і має більш широке застосування.

Присутність в зображенні найсвітліших і найтемніших тонів створює враження гарної контрастності, однак зайва контрастність призводить до того, що максимальні градації впливають на середні тони, а більшість деталей зображення пофарбовані саме в середніх тонах і зайва контрастність може призводить до втрати цих деталей або ускладнити їх виділення.

Для керування яркістю і контрастністю зображення реалізовано клас CEffectBrightnessContrast:

```
class FCEffectBrightnessContrast : public FCEffectLUT
{
    int   m_brightness ;
```

```
    int   m_contrast ;
public:
```

До класу включено конструктор, який допомагає задавати рівень ефекту яскравості зображення:

```
/**
        Constructor \n
        param -100 <= nBrightness <= 100, 0 means not change \n
        param -100 <= nContrast <= 100, 0 means not change
    */
    FCEffectBrightnessContrast (int nBrightness, int nContrast)
    {
        m_brightness = FClamp(nBrightness, -100, 100) ;
        m_contrast = FClamp(nContrast, -100, 100) ;
    }
private:
    virtual int InitLUTtable (int nLUTIndex)
    {
        double  d = (100 + m_contrast) / 100.0 ;
        int      n = (int)((nLUTIndex-128) * d + (m_brightness +
128) + 0.5) ;
        return FClamp0255(n) ;
    }
};
```

3.2.3 Клас FCColor з основними методами роботи над зображенням

Клас FCColor є потомком класу RGBQUAD:

```
Class FCColor : public RGBQUAD
{
    int   m_doxygen ;
   public:
```

Конструктор, що приймає колір зображення у RGB:

```
  FCColor (int r, int g, int b, int a=0xFF)
    {
        rgbRed = (BYTE)r ;
        rgbGreen = (BYTE)g ;
        rgbBlue = (BYTE)b ;
        rgbReserved = (BYTE)a ;
    }
```

Конструктор, який приймає колір в числовому форматі COLORREF - стандартний тип для представлення кольорів в Win32:

```
FCColor (COLORREF c)
  {
      rgbRed = GetRValue(c) ;
      rgbGreen = GetGValue(c) ;
      rgbBlue = GetBValue(c) ;
      rgbReserved = 0xFF ;
  }
```

Крім цього до класу FCColor включено метод CombinePixel() для комбінування двох 32bpp (bits per pixel – кількість біт на піксель) пікселів (де: c1 – задній план, c2 – передній план):

```
  static RGBQUAD CombinePixel (RGBQUAD c1, RGBQUAD c2)
  {
      if (PCL_A(&c1) || PCL_A(&c2))
      {
          if (PCL_A(&c2) == 0)
          {
              return c1 ;
          }
          if ((PCL_A(&c1) == 0) || (PCL_A(&c2) == 0xFF))
          {
              return c2 ;
          }

          RGBQUAD   ret ;
```

Якщо перевірка показчиків заднього та переднього планів фото задовольняє умовам то йде перетворення зображення згідно алгоритму:

```
          int   t1 = 0xFF * PCL_A(&c1),
                t2 = 0xFF * PCL_A(&c2),
                k = t1 - PCL_A(&c1) * PCL_A(&c2), // >= 0
                nTemp = t2 + k ; // ==0 only alpha1=0 & alpha2=0
           PCL_B(&ret) = (BYTE)((t2*PCL_B(&c2) + k*PCL_B(&c1)) /
nTemp) ;
           PCL_G(&ret) = (BYTE)((t2*PCL_G(&c2) + k*PCL_G(&c1)) /
nTemp) ;
```

```
            PCL_R(&ret) = (BYTE)((t2*PCL_R(&c2) + k*PCL_R(&c1)) /
nTemp) ;
            PCL_A(&ret) = (BYTE)(nTemp / 0xFF) ;
```

Для спрощення читання неоптимізованого зображення використовується наступні дії:

```
            int     nTemp = 0xFF * (PCL_A(&c1) + PCL_A(&c2)) -
PCL_A(&c1)*PCL_A(&c2) ;
            PCL_B(&ret) = (0xFF*PCL_B(c2)*PCL_A(&c2) + (0xFF -
PCL_A(&c2))*PCL_B(&c1)*PCL_A(&c1)) / nTemp ;
            PCL_G(&ret) = (0xFF*PCL_G(c2)*PCL_A(&c2) + (0xFF -
PCL_A(&c2))*PCL_G(&c1)*PCL_A(&c1)) / nTemp ;
            PCL_R(&ret) = (0xFF*PCL_R(c2)*PCL_A(&c2) + (0xFF -
PCL_A(&c2))*PCL_R(&c1)*PCL_A(&c1)) / nTemp ;
            PCL_A(&ret) = nTemp / 0xFF ;
            */
            return ret ;
        }
        return FCColor(0xFF,0xFF,0xFF,0) ;
    }
```

Метод для змішування кольорів AlphaBlend cr на pDest, результат зберігається у pDest. pDest – адреса 24bpp або 32bpp пікселю, cr – передній план.

```
     static void AlphaBlendPixel (void* pDest, RGBQUAD cr)
    {
        int   a = PCL_A(&cr) ;
        if (a == 0xFF)
        {
            PCL_B(pDest) = PCL_B(&cr) ;
            PCL_G(pDest) = PCL_G(&cr) ;
            PCL_R(pDest) = PCL_R(&cr) ;
            return ;
        }
        if (a == 0)
            return ;

        int   t = 0xFF - a ;
        PCL_B(pDest) = (BYTE)((PCL_B(&cr) * a + PCL_B(pDest) * t)
/ 0xFF) ;
        PCL_G(pDest) = (BYTE)((PCL_G(&cr) * a + PCL_G(pDest) * t)
/ 0xFF) ;
        PCL_R(pDest) = (BYTE)((PCL_R(&cr) * a + PCL_R(pDest) * t)
/ 0xFF) ;
    }
```

Метод розрахунку значення відтінків сірого в пікселях. prgb – адреса 24bpp або 32bpp пікселю.

```
     static BYTE GetGrayscale (const void* prgb)
    {
        return (BYTE)((30*PCL_R(prgb) + 59*PCL_G(prgb) +
11*PCL_B(prgb)) / 100) ;
    }
```

Метод швидкого копіювання пікселів. nBPP - 8, 16, 24, 32:

```
static void CopyPixelBPP (void* pDest, const void* pSrc, int
nBPP)
    {
        if (nBPP == 32)
            *(DWORD*)pDest = *(DWORD*)pSrc ;
        else if (nBPP == 24)
        {
            *(WORD*)pDest = *(WORD*)pSrc ;
            ((BYTE*)pDest)[2] = ((BYTE*)pSrc)[2] ;
        }
        else if (nBPP == 8)
            *(BYTE*)pDest = *(BYTE*)pSrc ;
        else if (nBPP == 16)
            *(WORD*)pDest = *(WORD*)pSrc ;
        else
        {
            assert(false) ;
        }
    }
```

Метод конвертації в кольорову модель HLS (hue, saturation, lightness) з RGB. prgb – адреса 24bpp або 32bpp пікселю.

```
 static void RGBtoHLS (const void* prgb, double& H, double& L,
double& S)
    {
        int   buf[] = {PCL_R(prgb), PCL_G(prgb), PCL_B(prgb)} ;
        int   n_cmax = *std::max_element(buf, buf+3) ;
        int   n_cmin = *std::min_element(buf, buf+3) ;
```

Покажчик світлоти зображення розраховується по наступній формулі:

```
L = (n_cmax + n_cmin) / 2.0 / 255.0 ;

        if (n_cmax == n_cmin)
        {
            S = H = 0.0 ;
            return ;
        }
```

Перетворення з RGB моделі виконуються наступним чином:

```
        double   r = PCL_R(prgb) / 255.0,
                 g = PCL_G(prgb) / 255.0,
                 b = PCL_B(prgb) / 255.0,
                 cmax = n_cmax / 255.0,
                 cmin = n_cmin / 255.0,
                 delta = cmax - cmin ;
```

Враховується рівень світлоти зображення:

```
        if (L < 0.5)
            S = delta / (cmax + cmin) ;
        else
            S = delta / (2.0 - cmax - cmin) ;

        if (PCL_R(prgb) == n_cmax)
            H = (g-b) / delta ;
        else if (PCL_G(prgb) == n_cmax)
            H = 2.0 + (b-r) / delta ;
        else
            H = 4.0 + (r-g) / delta ;
        H /= 6.0 ;
        if (H < 0.0)
            H += 1.0 ;
    }
```

Метод конвертації в кольорову модель RGB з моделі HLS:

```
    static RGBQUAD HLStoRGB (double H, double L, double S)
    {
        if ((!(S > 0)) && (!(S < 0))) // == 0
            return DoubleRGB_to_RGB (L, L, L) ;
```

тут перевіряється рівень світлоти і виконуються відповідні перетворення для зображення:

```
        double   m1, m2 ;
        if (L > 0.5)
            m2 = L + S - L*S ;
        else
            m2 = L * (1.0 + S) ;
        m1 = 2.0*L - m2 ;

        double   r = HLS_Value (m1, m2, H*6.0 + 2.0) ;
        double   g = HLS_Value (m1, m2, H*6.0) ;
        double   b = HLS_Value (m1, m2, H*6.0 - 2.0) ;
        return DoubleRGB_to_RGB (r,g,b) ;
    }
```

Метод конвертації в кольорову модель HSV (Hue, Saturation, Value) з RGB. prgb – адреса 24bpp або 32bpp пікселю.

```
static void RGBtoHSV (const void* prgb, double& H, double& S,
double& V)
    {
        int   buf[] = {PCL_R(prgb), PCL_G(prgb), PCL_B(prgb)} ;
        int   n_cmax = *std::max_element(buf, buf+3) ;
        int   n_cmin = *std::min_element(buf, buf+3) ;
```

Кожен колір моделі переводимо до відповідних значень:

```
        double   r = PCL_R(prgb) / 255.0,
                 g = PCL_G(prgb) / 255.0,
                 b = PCL_B(prgb) / 255.0,
                 delta = (n_cmax - n_cmin) / 255.0 ;

        V = n_cmax / 255.0 ;

        if (n_cmax == n_cmin)
        {
            S = H = 0.0 ;
            return ;
        }
```

У випадку виконання умови насиченість зображення розраховується наступним чином:

```
        S = (n_cmax - n_cmin) / (double)n_cmax ;
        if (PCL_R(prgb) == n_cmax)
            H = (g - b) / delta ;
```

```
        else if (PCL_G(prgb) == n_cmax)
            H = 2.0 + (b - r) / delta ;
        else if (PCL_B(prgb) == n_cmax)
            H = 4.0 + (r - g) / delta ;
        H /= 6.0 ;
        if (H < 0.0)
            H += 1.0 ;
        else if (H > 1.0)
            H -= 1.0 ;
    }
```

Метод конвертації в кольорову модель RGB з моделі HLV:

```
    static RGBQUAD HSVtoRGB (double h, double s, double v)
    {
        if ((!(s > 0)) && (!(s < 0))) // == 0
            return DoubleRGB_to_RGB (v, v, v) ;

        if (!(h < 1)) // >= 1
            h = 0.0 ;
        h *= 6.0 ;

        double  f = h - (int)h,
                p = v * (1.0 - s),
                q = v * (1.0 - s * f),
                t = v * (1.0 - s * (1.0 - f)) ;
        switch ((int)h)
        {
            case 0 : return DoubleRGB_to_RGB (v, t, p) ;
            case 1 : return DoubleRGB_to_RGB (q, v, p) ;
            case 2 : return DoubleRGB_to_RGB (p, v, t) ;
            case 3 : return DoubleRGB_to_RGB (p, q, v) ;
            case 4 : return DoubleRGB_to_RGB (t, p, v) ;
            case 5 : return DoubleRGB_to_RGB (v, p, q) ;
        }
        return DoubleRGB_to_RGB (0, 0, 0) ;
    }
```

Метод розрахунку білінійної інтерроляції (0 <= x,y < 1, дистанція до crPixel[0,0], nBpp – може бути 24 or 32, crPixel – може мати наступні значення: [0,0], [1,0], [0,1], [1,1]):

```
static RGBQUAD GetBilinear (double x, double y, int nBpp, BYTE*
crPixel[4])
    {
        const BYTE  * px0 = crPixel[0], * px1 = crPixel[1],
```

```
                    * px2 = crPixel[2], * px3 = crPixel[3] ;
        RGBQUAD     crRet = {0xFF, 0xFF, 0xFF, 0xFF} ;

        if ((nBpp == 24) || ((PCL_A(px0) & PCL_A(px1) & PCL_A(px2)
& PCL_A(px3)) == 0xFF))
        {
```

Якщо діапозон 24біт, то:

```
            for (int i=0 ; i < 3 ; i++)
            {
                double   m0 = px0[i] + x * (px1[i] - px0[i]),
                         m1 = px2[i] + x * (px3[i] - px2[i]),
                         my = m0 + y * (m1 - m0) ;
                ((BYTE*)&crRet)[i] = FClamp0255(my) ;
            }
        }
        else
        {
```

Якщо діапозон 32біт, то:

```
 // is 32bit with alpha, calculate destinate alpha value
            double     m0  =  PCL_A(px0)  +  x  *  (PCL_A(px1)  -
PCL_A(px0)),
                       m1  =  PCL_A(px2)  +  x  *  (PCL_A(px3)  -
PCL_A(px2)),
                       my = m0 + y * (m1 - m0),
                       dAlpha = my ;
            PCL_A(&crRet) = FClamp0255(my) ;
            if (PCL_A(&crRet)) // has alpha
            {
                for (int i=0 ; i < 3 ; i++)
                {
                  double   nSum0 = PCL_A(px0) * px0[i],
                           nSum2 = PCL_A(px2) * px2[i] ;

                 m0 = nSum0 + x * (px1[i] * PCL_A(px1) - nSum0) ;
                 m1 = nSum2 + x * (px3[i] * PCL_A(px3) - nSum2) ;
                 my = m0 + y * (m1 - m0) ;
                 ((BYTE*)&crRet)[i] = FClamp0255(my / dAlpha) ;
                }
            }
        }
        return crRet ;    }
```

Робота методу HLS_Value ():

```
private:
    static double HLS_Value (double n1, double n2, double h)
    {
        if (h > 6.0)
            h -= 6.0 ;
        else if (h < 0.0)
            h += 6.0 ;

        if (h < 1.0)
            return n1 + (n2 - n1) * h ;
        else if (h < 3.0)
            return n2 ;
        else if (h < 4.0)
            return n1 + (n2 - n1) * (4.0 - h) ;
        return n1 ;
    }
```

Метод округлення значення RGB:

```
    static RGBQUAD DoubleRGB_to_RGB (double r, double g, double b)
    {
        return  FCColor  (FClamp0255(r*255),  FClamp0255(g*255),
FClamp0255(b*255)) ;
    }
};
```

3.2.4 Кольоровий тон «Color Tone»

Фільтр «Тон» управляє колірним тоном (кольором) і насиченістю (чистотою) всього зображення або окремих колірних складових зображення.

Фільтр «Тон» можна використовувати для створення спеціальних ефектів, фарбування чорно-білої фотографії (наприклад, в тон «сепія») або зміни колірного діапазону на зображенні.

Для застосування фільтра повзунком «Колір» необхідно вибрати колір, а повзунком «Насиченість» змінити рівень насиченості.

Для реалізації фільтру Color Tone створено класс:

```
class FCEffectColorTone : public FCImageEffect
{
    double   m_hue ;
```

```
    double   m_saturation ;
    double   m_lum_tab[256] ;
public:
```

Конструктор, що приймає колір тону та значення насиченості:

```
    FCEffectColorTone (RGBQUAD crTone, int nSaturation)
    {
        double   l ;
        FCColor::RGBtoHLS (&crTone, m_hue, l, m_saturation) ;
```

m_saturation = m_saturation * (nSaturation/255.0) * (nSaturation/255.0) ;

m_saturation = ((m_saturation < 1) ? m_saturation : 1) ;

```
    for (int i=0 ; i < 256 ; i++)
        {
            RGBQUAD   cr = FCColor(i,i,i) ;
            double    h, l, s ;
            FCColor::RGBtoHLS (&cr, h, l, s) ;

            l = l * (1 + (128-abs(nSaturation-128)) / 128.0 / 9.0)
;
            m_lum_tab[i] = ((l < 1) ? l : 1) ;
        }
    }
```

Метод для застосування фільтру виконується у наступній послідовності:

- визначаємо насиченість тону;
- отримуємо колір переднього плану;
- викликаємо метод CopyPixel для швидкого застосування фільтру на всі пікселі зображення, що змішує колір основного зображення та переднього плану.

```
virtual void ProcessPixel (FCObjImage& img, int x, int y, BYTE*
pPixel)
    {
        double    l = m_lum_tab[FCColor::GetGrayscale(pPixel)];
```

```
        RGBQUAD   cr = FCColor::HLStoRGB (m_hue, l,
m_saturation);
        FCColor::CopyPixelBPP (pPixel, &cr, 24);
    }
};
```

3.2.5 Реалізація фільтру «Black&White»

Фільтр «Знебарвити» виконує перетворення кольорового зображення в чорно-біле, привласнюючи кожному пікселю зображення в режимі RGB пропорційне значення червоного, зеленого і синього. Загальна яскравість кожного пікселя залишається незмінною. Дана команда приводить до аналогічного результату, який досягається установкою значення насиченості -100 в діалоговому вікні «Кольоровий тон».

Для реалізації фільтру створено клас FCEffectGrayscale:

```
class FCEffectGrayscale : public FCImageEffect
{
    virtual void ProcessPixel (FCObjImage& img, int x, int y, BYTE*
pPixel)
    {
        PCL_R(pPixel)   =   PCL_G(pPixel)   =   PCL_B(pPixel)   =
FCColor::GetGrayscale(pPixel) ;
    }
public:
    /// Constructor.
    FCEffectGrayscale() {}
};
```

3.2.6 Реалізація фільтру «Розмивання Гауса»

При конвертації гаусівських безперервних величин у дискретні значення, необхідного для ядра, сума цих значень буде відрізнятися від 1. Це може викликати потемніння або яскравість зображення. Для виправлення цього значення може бути нормалізована шляхом ділення кожного терміна в ядрі за сумою всіх умов в ядрі.

Для реалізації розмивання Гауса, яке використовується в більшості фільтрів, створено клас FCEffectBlur_Gauss:

```
class FCEffectBlur_Gauss : public FCImageEffect
{
public:
    enum BLUR_COMPONENT
    {
        BLUR_COMPONENT_ALL = 1,
        BLUR_COMPONENT_ONLY_ALPHA = 2,
    };
```

Конструктор:

```
  FCEffectBlur_Gauss (int nSize, bool bCopyEdge,
BLUR_COMPONENT blur_component=BLUR_COMPONENT_ALL)
    {
        m_copy_edge = (bCopyEdge ? 1 : 0) ;
        m_component = blur_component ;
        MakeKernel (((nSize >= 1) ? nSize : 1) / 2.0) ;
        m_padding = (int)m_kernel.size() / 2 ;
    }
```

Якщо nSize >= 1, то знаення bCopyEdge – встановилюється як false для того, щоб розмити границю з alpha==0, blur_component – може бути двох значень: FCEffectBlur_Gauss::BLUR_COMPONENT_ALL або FCEffectBlur_Gauss::BLUR_COMPONENT_ONLY_ALPHA.

Також, для классу FCEffectBlur_Gauss реалізовано декілька методів для коректної роботи алгоритму:

- метод OnFinishPixe();
- метод виконання розмиття зображення ProcessWholeImage():
- метод горизонтального розмиття BlurHorizon();
- метод вертикального розмиття BlurVertica();
- метод отримання вертикальної лінії VerticalGetLine();
- метод отримання горизонтальної лінії HorizonGetLine();
- метод заповнення FillPadding();
- метод отримання структури RGBQUAD;

- метод розмиття піклелю BlurOnePixel();
- метод косвеної рекурсії для створення ядра розмиття MakeKernel();
- метод розділення ваги DivideWeight().

Метод OnFinishPixe(), що може використовуватися похідним класом для модифікації pPixel, що був розмитий:

```
protected:
    virtual void OnFinishPixel (FCObjImage& img, int x, int y,
BYTE* pPixel) {}
```

Тут використовуються значення, необхідні для розрахунку розмиття:

```
private:
    int      m_copy_edge ;
    BLUR_COMPONENT   m_component ;
    std::vector<double>   m_kernel ;
    int      m_padding ;

    int      m_bpp ; // 24 or 32
    RGBQUAD  * m_line_buf ;

private:
    enum
    {
        MAX_RADIUS = 100,
    };
```

Другий метод – метод розділення ваги DivideWeight():

```
    static void DivideWeight (std::vector<double>& ls)
    {
        double   t = 0 ;
        for (size_t i=0 ; i < ls.size() ; i++)
            t += ls[i] ;

        for (size_t i=0 ; i < ls.size() ; i++)
            ls[i] /= t ;
    }
```

Метод косвеної рекурсії для створення ядра розмиття MakeKernel():

```
void MakeKernel (double fRadius)
```

```
    {
        m_kernel.reserve (2*MAX_RADIUS + 1) ;
        for (int i = -MAX_RADIUS ; i <= MAX_RADIUS ; i++)
        {
            double   t = i / fRadius ;
            m_kernel.push_back (exp (-t*t/2)) ;
        }
```

Необхідно розділити вагу с урахуванням дуже малих значень:

```
        DivideWeight(m_kernel) ;
        double   max_precision = 1 / (2.0 * 255) ;
        double   fTotal = 0 ;
        for (int i=0 ; i < MAX_RADIUS ; i++)
        {
            fTotal += (2 * m_kernel.front()) ;

            if (fTotal < max_precision)
            {
                m_kernel.erase (m_kernel.begin()) ;
                m_kernel.pop_back() ;
            }
            else
            {
                break ;
            }
        }
        DivideWeight(m_kernel) ;
```

Метод отримання структури RGBQUAD, яка описує колір, що складається з відносних інтенсивностей червоного, зеленого і синього кольору:

```
RGBQUAD GetRGBQUAD (const void* pPixel)
    {
        RGBQUAD   c = FCColor(0,0,0,0xFF);//default alpha is 255
        FCColor::CopyPixelBPP (&c, pPixel, m_bpp) ;
        return c ;
    }
```

Метод заповнення FillPadding():

```
  void FillPadding (RGBQUAD*& d, const void* pPixel)
    {
        for (int i=0 ; i < m_padding ; i++, d++)
        {
```

```
            *d = (m_copy_edge ? GetRGBQUAD(pPixel) :
FCColor(0xFF,0xFF,0xFF,0)) ;
        }
    }
```

Метод отримання горизонтальної лінії HorizonGetLine():

```
  void HorizonGetLine (FCObjImage& img, int y, RGBQUAD* d)
    {
        FillPadding (d, img.GetBits(0,y)) ;
        for (int x=0 ; x < img.Width() ; x++, d++)
        {
            *d = GetRGBQUAD(img.GetBits(x,y)) ;
        }
        FillPadding (d, img.GetBits(img.Width()-1,y)) ;
    }
```

Метод отримання вертикальної лінії VerticalGetLine():

```
    void VerticalGetLine (FCObjImage& img, int x, RGBQUAD* d)
    {
        FillPadding (d, img.GetBits(x,0)) ;
        for (int y=0 ; y < img.Height() ; y++, d++)
        {
            *d = GetRGBQUAD(img.GetBits(x,y)) ;
        }
        FillPadding (d, img.GetBits(x,img.Height()-1)) ;
    }
```

Метод вертикального розмиття BlurVertica():

```
  void BlurVertical (FCObjImage& img, FCProgressObserver*
pProgress)
    {
        for (int x=0 ; x < img.Width() ; x++)
        {
            VerticalGetLine (img, x, m_line_buf) ;

            RGBQUAD   * s = m_line_buf ;
            for (int y=0 ; y < img.Height() ; y++, s++)
            {
                BlurOnePixel (img.GetBits(x,y), s) ;
            }
            if (pProgress)
            {
```

```
                if (!pProgress->UpdateProgress (0)) // need call
it for cancel process
                    break ;
            }
        }
    }
```

Метод горизонтального розмиття BlurHorizon () виконується аналогічно попередньому:

```
 void BlurHorizon (FCObjImage& img, FCProgressObserver*
pProgress)
    {
        for (int y=0 ; y < img.Height() ; y++)
        {
            HorizonGetLine (img, y, m_line_buf) ;

            RGBQUAD   * s = m_line_buf ;
            for (int x=0 ; x < img.Width() ; x++, s++)
            {
                BYTE   * pPixel = img.GetBits(x,y) ;
                BlurOnePixel (pPixel, s) ;
                OnFinishPixel (img, x, y, pPixel) ;
            }

            if (pProgress)
            {
                if (!pProgress->UpdateProgress (100 * (y+1) /
img.Height()))
                    break ;
            }
        }
    }
```

Метод виконання розмиття зображення ProcessWholeImage():

```
 virtual void ProcessWholeImage (FCObjImage& img,
FCProgressObserver* pProgress)
    {
        if (img.IsValidImage())
        {
            // prepare param
            m_bpp = img.ColorBits() ;
            int   w = img.Width() ;
            int   h = img.Height() ;
            int   nCount = ((w > h) ? w : h) + 2*m_padding + 2 ;
// +2 is not necessary
```

```
            m_line_buf = new RGBQUAD[nCount] ;
            BlurVertical (img, pProgress) ;
            BlurHorizon (img, pProgress) ;
            delete[] m_line_buf ;
        }
    }
};
```

ВИСНОВКИ

Графічна, інформація, відео фрагменти, фотографії все більше і більше стають основним матеріалом обробки на комп'ютері.

Під час аналізу предметної області були розглянуті аналоги програмного додатку, а також обрані програмні засоби реалізації дипломного проекту.

На етапі проектування побудована діаграма варіантів використання, а також розглянуті методи фільтрації зображень для подальшого їх програмування.

В ході виконання дипломної роботи було отримано повно – функціональне програмне забезпечення, повністю готове до застосування. Дана програма орієнтована на користувачів, що полюбляють оброблювати фотографії і зображення, а також може використовуватися в якості наочності в навчальному процесі.

Інтерфейс створеної програми зручний, простий, наочно відображає її можливості. Головне меню редактора містить команди роботи з файлами, а також команди скасованих і повторення змін.

В панелі фільтрів реалізовано понад 15 різних фото фільтрів, що дозволяють обробляти зображення з можливістю різного рівня накладення.

Розроблене програмне забезпечення задовольняє всім вимогам, поставленим на етапі постановки завдання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Комп'ютерна графіка та її види. URL: https://sites.google.com/site/informatika324/komp-uterna-grafika-ta-ieie-vidi. (дата звернення 10.03.2020).
2. Комп’ютерна графіка. URL: http://pz.ptngu.com/lectures/lectures/lecture_13.html. (дата звернення 10.03.2020).
3. Лучшие графические редакторы: Топ-20. URL: https://www.canva.com/ru_ru/obuchenie/graficheskij-redaktor-20/. (дата звернення 12.03.2020).
4. Бесплатные графические редакторы. URL: https://remontka.pro/besplatnye-graficheskie-redaktory/. (дата звернення 14.03.2020).
5. PicsArt – описание. URL: https://aminoapps.com/c/myphonerus/page/item/picsart-opisanie/ER8z_6BliLID5DN85gj53Lbx8l26LlGaG. (дата звернення 14.03.2020).
6. PicsArt. URL: https://viberfun.ru/multimedia/113-picsart.html. (дата звернення 11.03.2020).
7. Почему C++ крут, актуален и бессмертен. URL: https://vc.ru/hr/50161-pochemu-c-krut-aktualen-i-bessmerten. (дата звернення 11.03.2020).
8. Рейтинг востребованности языков программирования. URL: http://www.tadviser.ru/index.php/Статья:Рейтинг_востребованности_языков_программирования. (дата звернення 11.03.2020).
9. Вышла Visual Studio 2017: рассказываем о новых возможностях инструментов от Microsoft. URL: https://tproger.ru/news/vs-2017/. (дата звернення 16.03.2020).м

10. Знакомство с Visual Studio. URL: https://habr.com/ru/sandbox/79099/ . (дата звернення 16.03.2020).
11. Цветовые модели. URL: http://compgraph.tpu.ru/Colors_models.htm. (дата звернення 18.03.2020).
12. Цветовые модели HSV и HLS. URL: https://www.intuit.ru/studies/courses/70/70/lecture/2094?page=4. (дата звернення 20.03.2020).
13. Корекція фотографій у Photoshop. Тонова корекція фотографій. URL: http://www.ukr.vkadre.kiev.ua/photographers/foto-text/tonovaya-korrekcia.html. (дата звернення 23.03.2020).
14. Коригування різкості та розмиття зображення. URL: https://helpx.adobe.com/ua/photoshop/using/adjusting-image-sharpness-blur.html. (дата звернення 23.03.2020).
15. UML-моделирование. URL: https://fingers.by/blog/uml. (дата звернення 30.03.2020).
16. Диаграммы вариантов использования. URL: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4. (дата звернення 30.03.2020).
17. UML – універсальна мова моделювання. URL: http://sites.znu.edu.ua/webprog/lect/1238.ukr.html. (дата звернення 19.03.2020).