Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ ДЛЯ ОБРОБКИ ЗОБРАЖЕНЬ
DESIGN OF A SOFTWARE SYSTEM FOR DIGITAL IMAGE DATA PROCESSING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-222
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

БІЛИЙ В.О.
(ПРІЗВИЩЕ та ініціали)
Керівник КОЦУР М.І.
(ПРІЗВИЩЕ та ініціали)
Рецензент СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БІЛОГО Віктора Олексійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної системи опрацювання даних для обробки зображень. Design of a Software System for Digital Image Data Processing

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 7 " квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 03 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис системи опрацювання даних. 4. Експлуатація, тестування та експериментальне дослідження системи опрацювання даних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури системи опрацювання даних.	2 тиждень	Розділ 3
5	Розробка системи опрацювання даних.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	системи опрацювання даних.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Віктор БІЛИЙ
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Михайло КОЦУР
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 4 табл., 22 рис., 3 дод., 20 джерел.

C#, UNITY, VISUAL STUDIO, АЛГОРИТМ, ЗОБРАЖЕННЯ, МОВА ПРОГРАМУВАННЯ, СИСТЕМА ОПРАЦЮВАННЯ ДАНИХ.

Об’єкт дослідження – алгоритми та процеси обчислень, пов’язані з організацією та опрацюванням даних для обробки зображень.

Предмет дослідження – програмні системи опрацювання даних для обробки зображень.

Мета роботи – розробка програмної системи опрацювання даних для обробки зображень для зменшення витрат часу користувача на обробку цифрових зображень.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio, платформа Unity.

Результати. Розроблено проєктні рішення для створення системи опрацювання даних для обробки зображень. Створено систему опрацювання даних для обробки зображень. Проведено дослідження розробленої системи опрацювання даних для обробки зображень.

Висновки. Мету роботи досягнуто. Розроблено систему опрацювання даних для обробки зображень за допомогою мови програмування C#, середовища розробки Visual Studio та платформи Unity.

Галузь використання – обробка зображень.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 4 tables, 22 figures, 3 appendixes, 20 sources.

C#, UNITY, VISUAL STUDIO, ALGORITHM, IMAGE, PROGRAMMING LANGUAGE, DATA PROCESSING SYSTEM.

The object of research is algorithms and computational processes related to the organization and processing of data for image processing.

The subject of the research is data processing systems for image processing.

The purpose of this work is to develop a data processing system for image processing to reduce the user's time spent on processing digital images.

Materials, methods and technical tools: C# programming language, Visual Studio development environment, Unity platform.

Results. Project solutions for the creation of data processing system for image processing has been designed. The data processing system for image processing has been developed. The study of the developed data processing system for image processing has been conducted.

Conclusions. The goal of the work has been achieved. The data processing system for image processing using the C# programming language, Visual Studio development environment, Unity platform has been developed.

Scope of use – image processing.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

RGB – red green blue;

ІС – інформаційна система;

ОЗ – обробка зображень;

СОД – система опрацювання даних.

ВСТУП

Актуальність створення системи опрацювання даних для обробки зображень зумовлюється стрімким зростанням обсягів візуальної інформації та підвищенням ролі цифрових зображень у сучасних інформаційних процесах. У різних сферах людської діяльності зображення виступають одним із ключових джерел даних, оскільки вони містять значну кількість прихованої інформації, яка не може бути ефективно оброблена традиційними методами без застосування спеціалізованих програмних засобів. У зв’язку з цим виникає об’єктивна потреба в автоматизованих системах, здатних забезпечити збирання, зберігання, аналіз, перетворення та інтерпретацію зображень із заданим рівнем точності та продуктивності [1], [2].

Не менш важливим чинником актуальності є практична цінність систем обробки зображень для прикладних галузей. У медицині такі системи використовуються для підтримки діагностичних рішень і підвищення точності аналізу медичних зображень, у промисловості — для контролю якості продукції та виявлення дефектів, у транспортній сфері — для забезпечення безпеки руху та інтелектуального керування потоками. У сфері безпеки та моніторингу обробка зображень дозволяє оперативно реагувати на потенційні загрози, а в наукових дослідженнях — прискорює обробку експериментальних даних і сприяє отриманню нових знань [3], [4].

Таким чином, створення системи опрацювання даних для обробки зображень є актуальним завданням, що відповідає сучасним тенденціям розвитку інформаційних технологій. Реалізація таких систем сприяє підвищенню якості аналізу даних, оптимізації процесів прийняття рішень і забезпеченню конкурентоспроможності інформаційних рішень у різних галузях застосування [5], [6].

Незважаючи на значний прогрес у сфері опрацювання візуальних даних, програмні системи для обробки зображень характеризуються низкою суттєвих недоліків, які обмежують ефективність їх практичного застосування.

Однією з основних проблем є висока обчислювальна складність алгоритмів обробки зображень, особливо у випадках використання методів машинного навчання та глибоких нейронних мереж. Такі системи потребують значних апаратних ресурсів, що ускладнює їх впровадження в умовах обмеженої обчислювальної потужності або при необхідності роботи в реальному часі. Окремої уваги заслуговує проблема універсальності програмних рішень. Більшість систем опрацювання зображень орієнтовані на вузьке коло задач або конкретні предметні області, що ускладнює їх адаптацію до нових умов експлуатації.

Тому актуальною є розробка системи опрацювання даних для обробки зображень. У дипломній кваліфікаційній роботі бакалавра розв’язується актуальне завдання розробки системи опрацювання даних для обробки зображень для зменшення витрат часу користувача на обробку цифрових зображень.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв’язати такі задачі:

– виконати аналіз предметної області та систем опрацювання даних для обробки зображень;

– здійснити проєктування системи опрацювання даних для обробки зображень;

– створити систему опрацювання даних для обробки зображень;

– дослідити розроблену систему опрацювання даних для обробки зображень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Алгоритми та системи опрацювання даних для обробки зображень

Системи опрацювання даних для обробки зображень являють собою програмні комплекси, призначені для приймання, аналізу, перетворення, зберігання та візуального відтворення цифрових зображень з метою підвищення їх інформативності, якості або придатності до подальшого використання. У межах таких систем реалізуються алгоритми, що дозволяють змінювати структуру візуальних даних, коригувати параметри кольору і яскравості, виділяти або пригнічувати окремі елементи зображення, а також адаптувати графічний контент до конкретних умов сприйняття чи технічних вимог. Узагальнено ці системи виконують роль посередника між сирими вхідними даними та кінцевим результатом, що має аналітичну або прикладну цінність.

Переваги використання систем опрацювання даних для обробки зображень полягають у можливості автоматизації рутинних операцій, підвищенні точності та повторюваності результатів, а також у значному скороченні часу, необхідного для аналізу й модифікації візуальної інформації. Завдяки програмним засобам забезпечується стабільна якість обробки незалежно від людського чинника, що особливо важливо у випадках масової або потокової роботи з графічними даними. Крім того, такі системи дозволяють працювати з великими обсягами інформації, зберігаючи вихідні дані та результати перетворень у впорядкованому вигляді, що спрощує подальший доступ, повторне використання або інтеграцію з іншими інформаційними системами.

Особливості використання систем опрацювання даних для обробки зображень визначаються їх прикладною спрямованістю та рівнем складності реалізованих алгоритмів. У практичному застосуванні вони можуть використовуватися як для простих задач візуального покращення, так і для

складних аналітичних процесів, пов’язаних із розпізнаванням образів, вимірюванням параметрів або підготовкою даних для машинного аналізу. Важливою рисою є можливість неруйнівної обробки, за якої вихідні зображення залишаються незмінними, а всі корекції зберігаються у вигляді наборів параметрів. Це забезпечує гнучкість роботи та дозволяє повертатися до попередніх станів без втрати якості.

У сучасних умовах такі програмні системи все частіше реалізуються з урахуванням кросплатформності та доступності через мережеві сервіси, що розширює коло користувачів і сценарії застосування. Використання інтуїтивних інтерфейсів у поєднанні з потужними алгоритмічними засобами робить системи опрацювання даних для обробки зображень універсальним інструментом, здатним задовольняти потреби як непідготовлених користувачів, так і фахівців, для яких важливими є точність, контроль і відтворюваність результатів.

Процес розробки систем опрацювання даних для обробки зображень являє собою комплексну діяльність, що поєднує аналіз предметної області, проєктування програмної архітектури та реалізацію алгоритмів роботи з візуальною інформацією. На початковому етапі визначається призначення майбутньої системи, характер зображень, з якими планується працювати, а також вимоги до якості, швидкодії та масштабованості. У цей період формується уявлення про типові сценарії використання, умови експлуатації та обмеження, пов’язані з апаратними ресурсами або середовищем виконання.

Подальша розробка передбачає побудову загальної структури системи, де визначається спосіб взаємодії між модулями обробки, зберігання та візуалізації даних. Архітектурні рішення приймаються з урахуванням необхідності ефективного опрацювання великих обсягів графічної інформації, можливості паралельної обробки та збереження проміжних результатів. Особлива увага приділяється вибору алгоритмів обробки зображень, які повинні забезпечувати потрібний рівень точності й стабільності при мінімальних витратах ресурсів. На цьому етапі здійснюється адаптація

математичних моделей до практичних умов застосування та визначається спосіб їх інтеграції у програмну систему.

У процесі реалізації виконується програмування основних функціональних компонентів, включаючи механізми завантаження та підготовки зображень, обчислювальні модулі для перетворення даних і засоби керування параметрами обробки. Важливим аспектом є створення інтерфейсу взаємодії з користувачем, який має поєднувати доступність і наочність з можливістю гнучкого налаштування. Інтерфейсні рішення впливають на зручність використання системи та визначають ефективність її практичного застосування, тому вони розробляються з урахуванням особливостей цільової аудиторії.

Невід'ємною складовою процесу є тестування, під час якого перевіряється коректність роботи алгоритмів, стабільність системи та відповідність отриманих результатів початковим вимогам. Тестування виконується на різних типах зображень і за різних умов, що дозволяє виявити недоліки та оптимізувати окремі компоненти. Паралельно здійснюється оцінка продуктивності та якості обробки, що дає змогу скоригувати параметри або змінити реалізацію окремих модулів.

Завершальним етапом є підготовка системи до впровадження та подальшої експлуатації, що включає документування, налаштування механізмів оновлення й підтримки. Після введення в дію розвиток системи не припиняється, оскільки виникає потреба в адаптації до нових вимог, удосконаленні алгоритмів та розширенні функціональності. Таким чином, розробка систем опрацювання даних для обробки зображень є безперервним процесом, у якому поєднуються наукові підходи, інженерні рішення та практичний досвід використання.

Алгоритми, що застосовуються у процесі опрацювання даних для обробки зображень, формують основу трансформації, аналізу та інтерпретації візуальної інформації. Вони покликані забезпечувати зміну параметрів зображення для досягнення конкретних цілей, таких як поліпшення якості,

виділення об’єктів, оптимізація кольорів та тональних співвідношень, а також підготовка даних для подальшого використання у візуалізації чи машинному аналізі. На початковому рівні застосовуються алгоритми корекції світлотіні, контрастності, яскравості та насиченості, які дозволяють збалансувати візуальні характеристики зображення відповідно до обраних стандартів або художніх уподобань. Ці алгоритми можуть працювати як глобально, змінюючи властивості всього кадру, так і локально, впливаючи на окремі ділянки зображення.

У складніших випадках використовуються алгоритми фільтрації та згладжування, які усувають шум, підвищують чіткість контурів та виділяють деталі. Для цього можуть застосовуватися методи просторової та частотної обробки, включно з перетворенням Фур’є, розмиттям, гострими масками та адаптивними фільтрами. Одночасно широко застосовуються алгоритми сегментації, що дозволяють розділяти зображення на смислові області та виокремлювати об’єкти, які потребують окремої обробки або аналізу. Ці методи часто поєднуються з алгоритмами класифікації та розпізнавання форм, що відкриває можливості для подальшого автоматизованого аналізу сцен.

Особливе значення мають алгоритми кольорокорекції та перетворення колірних просторів, які забезпечують точне відтворення відтінків і баланс тонів у різних умовах освітлення. До них відносяться методи регулювання гістограми, трансформацій кривих, корекції за зразком та алгоритми для роботи з каналами RGB чи іншими моделями кольору. Такі обчислювальні підходи дозволяють досягати професійного рівня точності, що особливо важливо при підготовці матеріалів для друку або цифрового розповсюдження.

Для розширеного опрацювання часто застосовуються алгоритми обробки шарів та масок, що дозволяють здійснювати паралельні модифікації зображення без впливу на оригінальні дані. Неруйнівне редагування реалізується за допомогою алгоритмічних структур, які зберігають всі параметри змін у вигляді окремих наборів, що забезпечує можливість повернення до будь-якого етапу обробки. Крім того, сучасні системи активно

інтегрують алгоритми машинного навчання та штучного інтелекту для автоматичного поліпшення якості зображень, відновлення деталей, виділення об’єктів і навіть генерації художніх ефектів.

Отже, алгоритми опрацювання даних для обробки зображень поєднують базові методи корекції та фільтрації, складні моделі сегментації та розпізнавання, техніки управління кольором та тональністю, а також інтелектуальні підходи для автоматизації та оптимізації процесу. Вони забезпечують гнучкість, точність і масштабованість, що дозволяє системам обробки зображень ефективно виконувати як рутинні, так і складні аналітичні чи творчі завдання.

Алгоритми, що застосовуються для обробки зображень, дозволяють не тільки змінювати зовнішній вигляд кадру, але й витягати смислову інформацію, виявляти закономірності та структури, приховані у графічних даних. Завдяки цьому алгоритмічні підходи стають важливою складовою не тільки творчого редагування, але й наукового аналізу, автоматизованої діагностики або підготовки даних для комп’ютерного зору.

Особливу роль відіграють алгоритми адаптивної обробки, які здатні підлаштовувати свої параметри під конкретні умови зображення, враховуючи рівень шуму, освітлення, наявність контрастних об’єктів та текстурні особливості сцени. Такі методи забезпечують високу точність змін та зменшують ризик втрати деталей або виникнення артефактів при обробці складних кадрів. У поєднанні з алгоритмами сегментації та класифікації вони дозволяють автоматично розділяти зображення на об’єкти різного типу, виділяти зони інтересу та визначати межі елементів з високою достовірністю.

Додатково алгоритми опрацювання можуть включати процедури аналізу геометрії та просторових співвідношень, що забезпечує корекцію перспективи, вирівнювання кадру та інтеграцію графічних елементів у складніші композиції. Це особливо важливо для систем, що поєднують обробку зображень із візуалізацією даних або інтерактивними графічними

середовищами, де точність геометричних трансформацій визначає якість кінцевого результату.

Сучасні алгоритмічні рішення часто інтегрують методи штучного інтелекту та глибокого навчання, що дозволяє виконувати складні операції, які раніше вимагали ручного втручання. Вони включають інтелектуальне видалення шуму, реконструкцію деталей, автоматичне підлаштування кольорів і освітлення, а також генерацію художніх ефектів або стилізацію зображень. Завдяки цьому алгоритми стають здатними не лише повторювати стандартні корекції, але й пропонувати нові варіанти обробки, адаптовані під контекст та творчі завдання користувача.

Крім того, алгоритми опрацювання зображень тісно пов’язані з оптимізацією продуктивності. Вони розробляються таким чином, щоб забезпечувати обробку великих обсягів даних у реальному часі або з мінімальними затримками, що робить можливим інтерактивну роботу з високоякісними фотографіями та потоковими відео. Паралельні обчислювальні методи, використання апаратного прискорення та адаптивне керування ресурсами дозволяють підтримувати стабільність і ефективність навіть у складних умовах обробки великих графічних масивів.

Таким чином, алгоритми опрацювання даних для зображень поєднують у собі адаптивність, точність, аналітичні та творчі можливості, а також ефективне використання ресурсів, що робить їх основним механізмом перетворення, аналізу та управління графічною інформацією у сучасних програмних системах.

Визначено, що системи опрацювання даних для обробки зображень являють собою програмні комплекси, призначені для приймання, аналізу, перетворення, зберігання та візуального відтворення цифрових зображень з метою підвищення їх інформативності, якості або придатності до подальшого використання. У межах таких програмних систем реалізуються алгоритми, що дозволяють змінювати структуру візуальних даних, коригувати параметри кольору і яскравості, виділяти або пригнічувати окремі елементи зображення,

а також адаптувати графічний контент до конкретних умов сприйняття чи технічних вимог.

1.2 Аналіз систем опрацювання даних для обробки зображень

Проаналізуємо системи опрацювання даних для обробки зображень [7]-[14].

Програмна система для обробки зображень Pixlr (рис. 1.1) призначена для виконання завдань цифрового редагування графічного контенту в онлайн-середовищі без потреби у встановленні спеціалізованого програмного забезпечення на локальний пристрій. Її основне призначення полягає у забезпеченні швидкої та доступної обробки растрових зображень з можливістю корекції, покращення якості та творчої трансформації візуальних даних для подальшого використання у вебпроєктах, соціальних мережах, навчальних матеріалах або дизайнерських роботах. Функціонування системи орієнтоване як на користувачів без професійної підготовки, так і на тих, хто має досвід роботи з графікою, що досягається поєднанням інтуїтивного інтерфейсу та достатньо гнучкого набору інструментів [7], [12].

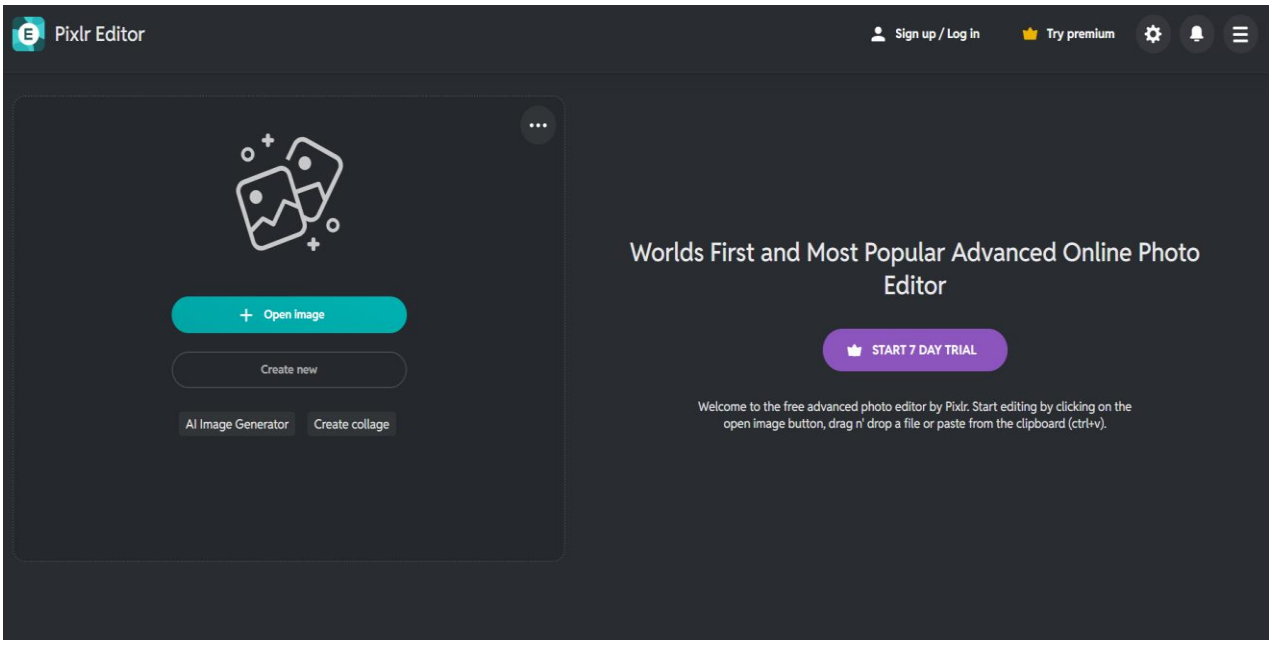


Рисунок 1.1 – Програмна система Pixlr [12]

Особливістю цієї програмної системи є орієнтація на хмарну модель опрацювання даних, за якої основні операції виконуються у браузері або мобільному середовищі з мінімальними вимогами до обчислювальних ресурсів клієнтського пристрою. Завдяки цьому забезпечується доступ до функціоналу з різних платформ і операційних середовищ, а робота з зображеннями не прив'язується до конкретного комп'ютера. В процесі редагування підтримується застосування візуальних ефектів, стилістичних фільтрів, шарів та масок, що дозволяє виконувати як базову корекцію, так і складніші перетворення без суттєвого ускладнення робочого процесу. Простота взаємодії з інтерфейсом користувача поєднується з можливістю отримання результатів, які за візуальною якістю відповідають професійним вимогам [7], [12].

З точки зору програмної реалізації система побудована як клієнт-серверний вебзастосунок, у якому значна частина логіки обробки даних реалізована засобами сучасних вебтехнологій. Для відображення інтерфейсу та взаємодії з користувачем застосовуються динамічні компоненти, що працюють без перезавантаження сторінки, завдяки чому редагування відбувається у режимі реального часу. Опрацювання зображень ґрунтується на використанні графічних API браузера та оптимізованих алгоритмів роботи з растровими даними, що дозволяє виконувати операції швидко та стабільно навіть при обмежених ресурсах. Серверна частина відповідає за зберігання даних, керування сесіями та синхронізацію проєктів між пристроями, що підвищує зручність і безперервність роботи [7], [12].

Важливою характеристикою програмної системи Pixlr є інтеграція автоматизованих механізмів обробки зображень, які спрощують виконання типових завдань і зменшують кількість ручних дій. Це робить систему ефективним інструментом для масового використання, коли необхідно швидко підготувати візуальний контент без глибокого занурення в технічні деталі графічного редагування. У результаті Pixlr можна розглядати як сучасну систему опрацювання даних для обробки зображень, у якій поєднуються

доступність, функціональність та технологічна адаптованість до вимог онлайн середовища [7], [12].

Основними характеристиками програмної системи Pixlr є такі [7], [12]:

– веборієнтованість платформи: система Pixlr функціонує переважно у браузерному середовищі, що усуває потребу у встановленні локального програмного забезпечення та спрощує доступ до інструментів редагування з різних пристроїв [7], [12];

– зручність користувацького інтерфейсу: інтерфейс програмної системи Pixlr побудовано з урахуванням інтуїтивної взаємодії, що дозволяє швидко освоїти основні можливості навіть без попереднього досвіду роботи з графічними редакторами [7], [12];

– функціональна гнучкість: підтримується широкий набір операцій для корекції, трансформації та художньої обробки зображень, що дає змогу адаптувати систему як для простих, так і для складніших завдань [7], [12];

– робота з шарами: у програмній системі Pixlr реалізовано механізми багатошарового редагування, які забезпечують незалежну обробку окремих елементів зображення та підвищують точність внесення змін [7], [12];

– використання візуальних ефектів: система Pixlr надає можливість застосування фільтрів, накладань і стилістичних ефектів для швидкого покращення або зміни зовнішнього вигляду графічних даних [7], [12];

– хмарне зберігання даних: результати обробки можуть зберігатися у хмарному середовищі, що спрощує доступ до проєктів і підтримує безперервність роботи [7], [12];

– оптимізація обчислювальних процесів: алгоритми обробки зображень адаптовані для ефективної роботи у вебсередовищі, що забезпечує прийнятну швидкодію без значного навантаження на клієнтський пристрій [7], [12];

– кросплатформна доступність: система Pixlr може використовуватися на різних операційних середовищах без втрати основного функціоналу, що розширює сферу її практичного застосуванняи [7], [12].

Серед недоліків системи опрацювання даних для обробки зображень Pixlr можна відзначити залежність від стабільного Інтернет з’єднання, обмежені можливості глибокого професійного налаштування у порівнянні з повнофункціональними настільними графічними редакторами, а також зниження продуктивності під час обробки складних або великоформатних зображень у браузерному середовищі [7], [12].

Darktable (рис. 1.2) є програмною системою для обробки зображень, створеною з відкритим вихідним кодом і спрямованою на організацію повного фотографічного робочого процесу. Використання програмної системи Darktable орієнтоване передусім на роботу з необробленими зображеннями, де ключову роль відіграє збереження початкових даних без прямого втручання у вихідні файли. У межах цієї системи реалізовано концепцію неруйнівного редагування, за якої всі зміни фіксуються як набір параметрів обробки, а не як перезапис зображення, що дозволяє повертатися до будь-якого етапу корекції без втрати якості [7]-[9], [13].

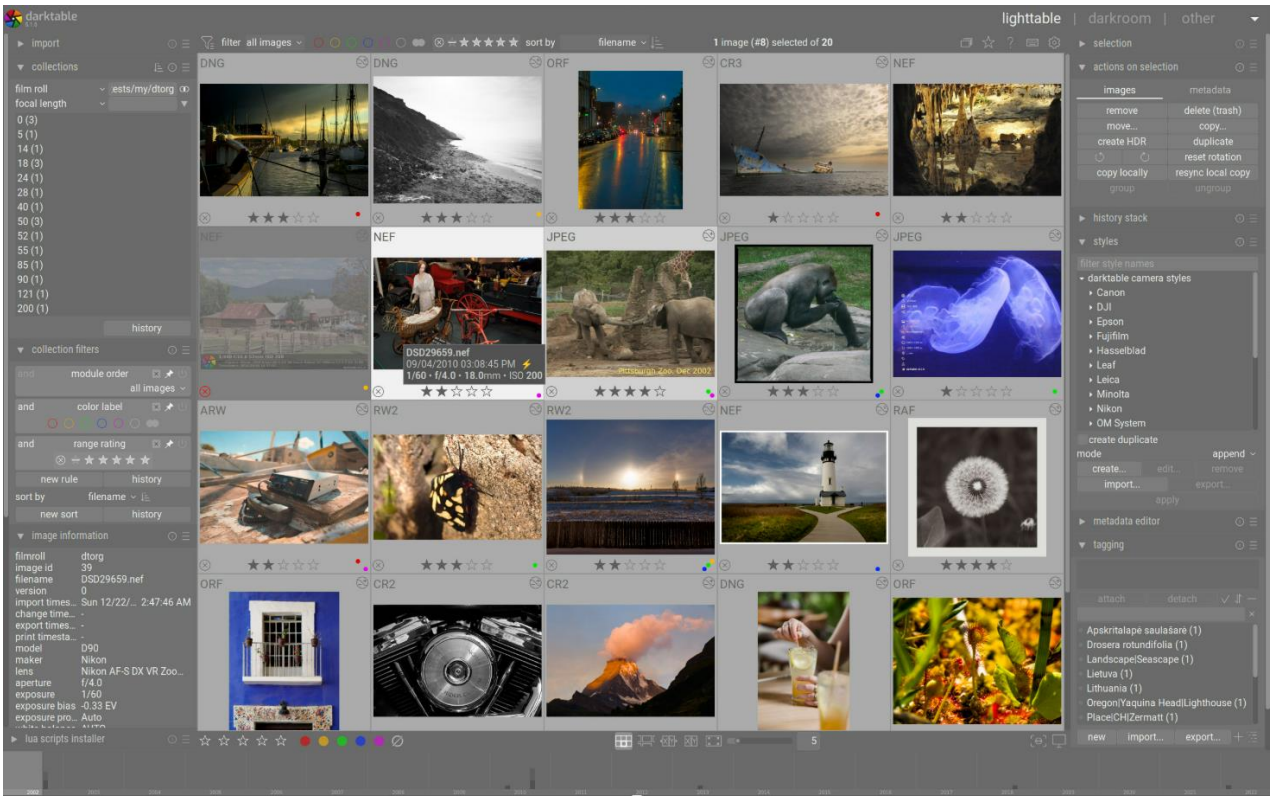


Рисунок 1.2 – Програмна система Darktable [13]

Функціонування Darktable поєднує інструменти керування фотоколекціями та засоби глибокої обробки зображень в єдиному середовищі. Для цього використовується логіка віртуального світлового столу, де здійснюється перегляд, відбір і впорядкування цифрових негативів, а також окремий простір редагування, у якому виконується детальне налаштування візуальних характеристик. Зображення зберігаються у внутрішній базі даних, що спрощує їх пошук, групування та систематизацію, а результати обробки можуть експортуватися як у локальні, так і у віддалені сховища [7]-[9], [13].

Особливістю програмної реалізації системи Darktable є підтримка широкого кола форматів необроблених зображень та використання модульного підходу до корекції. Обробка здійснюється через взаємопов’язані етапи, які можна комбінувати для формування складних сценаріїв редагування. У системі доступні інструменти для тонового відображення, роботи з кольоровими зонами, підвищення чіткості та локальної деталізації, що дозволяє досягати результатів, порівнюваних з професійними комерційними рішеннями. При цьому відсутність плати за використання не обмежує функціональні можливості, а глибина налаштувань робить систему особливо привабливою для тих, хто прагне повного контролю над обробкою зображень [7]-[9], [13].

Значна увага у системі Darktable приділяється управлінню бібліотекою фотографій, де передбачено механізми логічної організації матеріалів за допомогою міток, кольорових позначень і рейтингів. Такий підхід забезпечує плавний перехід від етапу відбору до детального редагування та сприяє побудові впорядкованого робочого процесу. Продуктивність системи залишається на високому рівні навіть під час роботи з великими файлами, що робить її придатною для інтенсивного використання без потреби у надпотужному апаратному забезпеченні [7]-[9], [13].

Разом з тим інтерфейс Darktable має виражений технічний характер і орієнтується передусім на функціональність, що може ускладнювати початкове освоєння. Це зумовлює більш тривалий період навчання у

порівнянні з простішими комерційними редакторами, проте водночас відкриває доступ до гнучких і точних інструментів обробки. Завдяки відкритому коду та активній спільноті розробників система постійно вдосконалюється, розширює набір можливостей і залишається актуальним рішенням для фотографів, які цінують прозорість, незалежність і максимальний контроль над процесом редагування зображень [7]-[9], [13].

Програмна система Darktable має такі особливості [7]-[9], [13]:

– відкритий вихідний код: програмна система Darktable розвивається як відкрите рішення, що забезпечує прозорість алгоритмів обробки та можливість їх удосконалення спільнотою користувачів і розробників [7]-[9], [13];

– орієнтація на роботу з raw-зображеннями: система Darktable спеціалізується на опрацюванні цифрових негативів із збереженням повного обсягу початкових даних та максимальної якості зображень [7]-[9], [13];

– неруйнівний підхід до редагування: всі корекції застосовуються у вигляді параметрів обробки без зміни оригінального файлу, що дозволяє багаторазово переглядати та коригувати результат [7]-[9], [13];

– модульна архітектура обробки: у системі Darktable редагування реалізується через незалежні модулі, які можуть комбінуватися у складні ланцюги для формування гнучкого робочого процесу [7]-[9], [13];

– розвинуті інструменти кольорокорекції: система Darktable надає засоби точного керування тоном, контрастом і кольоровими зонами, що дозволяє досягати професійного рівня візуальної якості [7]-[9], [13];

– управління фотобібліотекою: у Darktable передбачено механізми каталогізації зображень із використанням тегів, оцінок та кольорових позначок для ефективної організації великих колекцій [7]-[9], [13];

– висока продуктивність обробки: алгоритми програмної системи Darktable оптимізовані для роботи з великими файлами, що забезпечує стабільну швидкодію навіть при інтенсивному редагуванні [7]-[9], [13];

– кросплатформна підтримка: програмна система Darktable може використовуватися на різних операційних середовищах без обмеження основного функціоналу [7]-[9], [13];

– орієнтація на професійний контроль: інтерфейс і логіка роботи Darktable побудовані з урахуванням потреб користувачів, які прагнуть глибокого та точного налаштування кожного етапу обробки [7]-[9], [13].

Серед недоліків програмної системи для обробки зображень Darktable можна відзначити складність інтерфейсу для початківців, відносно круту криву навчання через велику кількість параметрів і модулів, а також меншу зручність для швидкого повсякденного редагування у порівнянні з більш спрощеними графічними редакторами [7]-[9], [13].

Програмна система Polarr (рис. 1.3) орієнтована на користувачів, які лише починають працювати з цифровою фотографією або потребують простого й зрозумілого інструмента для щоденного редагування [9], [11], [14].

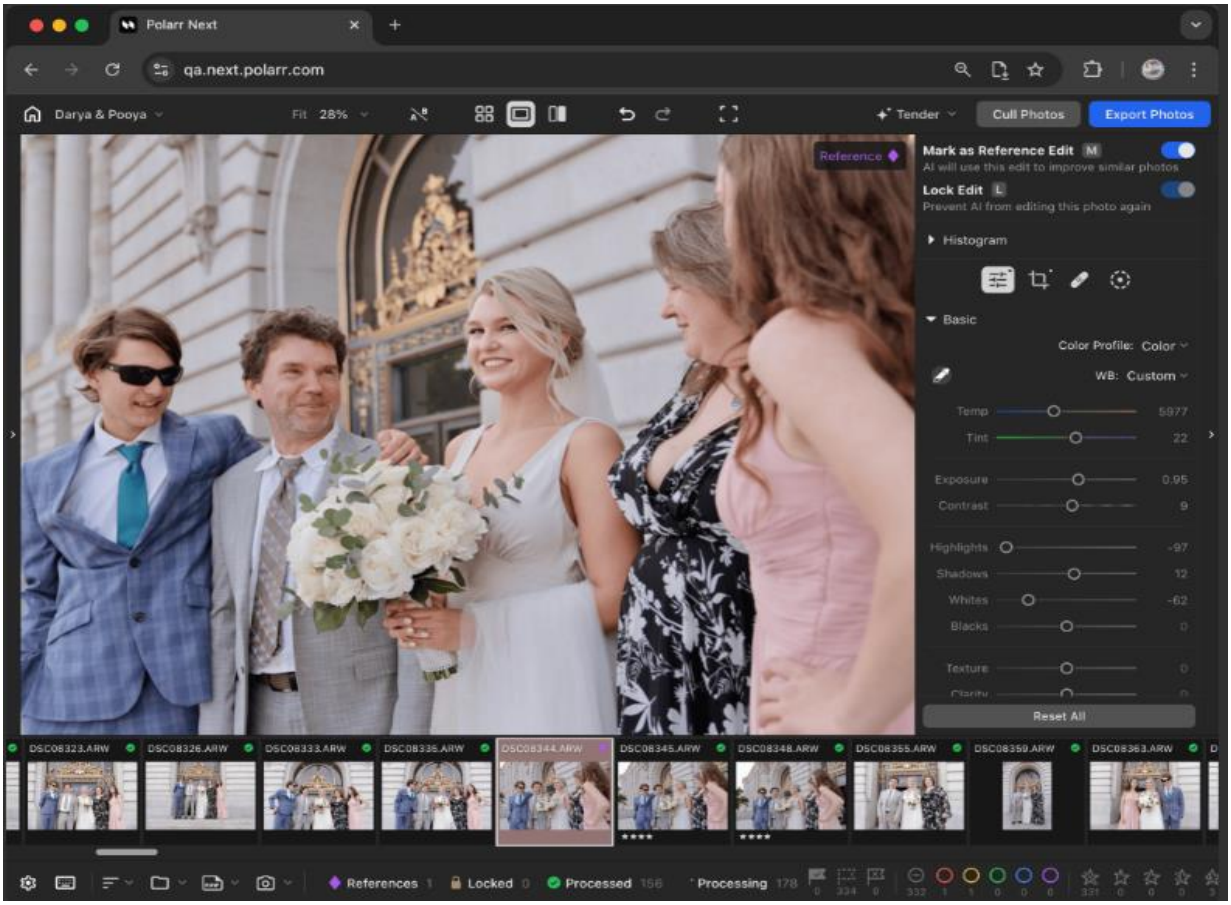


Рисунок 1.3 – Програмна система Polarr [14]

Концепція програмної системи Polarr побудована навколо доступності та мінімізації складності, завдяки чому взаємодія з інтерфейсом не створює відчуття перевантаженості функціями. Назви інструментів і логіка їх використання подані у зрозумілій формі, що дозволяє інтуїтивно сприймати призначення кожного елемента навіть без попереднього досвіду роботи з графічними редакторами [9], [11], [14].

Система надає можливість швидкого покращення зображень за допомогою готових стилів, які застосовуються миттєво та не потребують ручного налаштування параметрів. Поряд із цим реалізовано базові засоби ручної корекції, що дозволяють керувати яскравістю, контрастністю та співвідношенням світлих і темних ділянок кадру. Такий підхід поєднує автоматизоване редагування з можливістю поступового освоєння більш точних інструментів, не ускладнюючи початковий етап роботи [9], [11], [14].

Polarr функціонує як багатоплатформне рішення, що може використовуватися на настільних і мобільних пристроях, а також у вебсередовищі. Це забезпечує гнучкість доступу до інструментів редагування незалежно від обраного пристрою. Окрім стандартних засобів корекції, система пропонує розширені можливості для художнього оформлення зображень, зокрема використання градієнтів, накладень, ретуші та геометричних трансформацій. Підтримується робота з тональною кривою, а також опрацювання файлів, отриманих безпосередньо з фотокамер, що розширює сферу застосування програмної системи [9], [11], [14].

Важливою складовою Polarr є наявність спільноти користувачів, яка активно створює та поширює власні стилі й фільтри. Це сприяє обміну творчими рішеннями та постійному оновленню візуального контенту. Для тих, хто потребує глибшого контролю над редагуванням, передбачено розширений функціонал, що відкриває доступ до локальних інструментів точкового впливу на окремі зони зображення [9], [11], [14].

Окрему увагу в системі приділено творчим експериментам, адже доступні різноманітні художні ефекти, які дають змогу додавати світлові

акценти, декоративні відблиски та інші візуальні елементи. У результаті Polarr постає як універсальна програмна система для обробки зображень, що поєднує зручність, креативність і поступовий перехід від простих операцій до більш складних методів редагування без різкого зростання складності для користувача [9], [11], [14].

Програмна система Polarr має такі особливості [9], [11], [14]:

– орієнтація на початківців: програмна система Polarr спроєктована з урахуванням потреб користувачів без професійного досвіду, що забезпечує швидке освоєння основних інструментів редагування [9], [11], [14];

– спрощений інтерфейс: інтерфейс користувача Polarr побудовано з мінімальною кількістю елементів, що зменшує когнітивне навантаження та робить роботу з зображеннями більш комфортною [9], [11], [14];

– використання пресетів: у системі Polarr реалізовано велику кількість готових стилів, які дозволяють миттєво покращувати зовнішній вигляд фотографій без ручного налаштування параметрів [9], [11], [14];

– базові інструменти корекції: програмна система Polarr надає стандартні засоби для зміни яскравості, контрастності, експозиції та балансу світлих і темних ділянок зображення [9], [11], [14];

– можливість ручного редагування: передбачено інструменти тонкої настройки, що дозволяють користувачу поступово переходити від автоматичних ефектів до більш точного контролю результату [9], [11], [14];

– підтримка тональної кривої: у програмній системі Polarr реалізовано інструмент для детального керування світлотіньовими переходами та загальним контрастом зображення [9], [11], [14];

– розширені інструменти локальної обробки: у платній версії програмної системи Polarr доступні засоби для вибіркового редагування окремих областей кадру з високою точністю [9], [11], [14];

– робота з raw-файлами: програмна система Polarr підтримує обробку зображень без попереднього стиснення, що дозволяє зберігати максимальну якість вихідних даних [9], [11], [14];

– кросплатформна доступність: програмна система Polarr може використовуватися на різних типах пристроїв, включаючи настільні комп’ютери, мобільні платформи та вебсередовище [9], [11], [14];

– інтеграція спільноти користувачів: у програмній системі Polarr передбачено обмін фільтрами та стилями між користувачами, що сприяє творчій взаємодії та постійному оновленню вмісту [9], [11], [14].

Серед недоліків програмної системи для обробки зображень Polarr можна відзначити обмежені можливості глибокої професійної обробки у безкоштовній версії, залежність частини функціоналу від платної підписки, а також відсутність деяких автоматизованих інструментів, які притаманні більш складним професійним графічним редакторам [9], [11], [14].

Порівняльну характеристику систем опрацювання даних для обробки зображень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння систем опрацювання даних для обробки зображень

Критерій порівняння	Pixlr [12]	Darktable [13]	Polarr [14]
доступність на мобільних пристроїв	+	–	+
підтримка необроблених файлів	+–	+	+
простота інтерфейсу	+	+–	+
розширені інструменти професійного рівня	+–	+	–
можливість локального редагування зон	+–	+	+–
безкоштовність базового функціоналу	+–	+	+–

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато систем опрацювання даних для обробки зображень. Незважаючи на значний прогрес у сфері опрацювання візуальних даних, програмні системи для обробки зображень характеризуються низкою суттєвих недоліків, які обмежують ефективність їх практичного застосування. Однією з основних проблем є висока обчислювальна складність алгоритмів обробки зображень, особливо у випадках використання методів машинного навчання та глибоких нейронних мереж. Такі системи потребують значних апаратних ресурсів, що ускладнює їх впровадження в умовах обмеженої обчислювальної потужності або при необхідності роботи в реальному часі. Крім того, більшість систем опрацювання зображень орієнтовані на вузьке коло задач або конкретні предметні області, що ускладнює їх адаптацію до нових умов експлуатації. Тому актуальною є розробка системи опрацювання даних для обробки зображень.

При розробці системи опрацювання даних для обробки зображень необхідно забезпечити такі функціональні вимоги:

– підтримка можливості обробки зображень;

– підтримка можливості заливки контурів у зображенні обраним кольором;

– можливість вибору фрагментів зображень за допомогою дотикового (сенсорного) керування;

– можливість збереження та видалення редагованих зображень;

– підтримка можливості поділитися зображенням ним (зокрема, через зовнішні сервіси) з іншими користувачами.

1.3 Висновки за розділом 1

Визначено, що системи опрацювання даних для обробки зображень являють собою програмні комплекси, призначені для приймання, аналізу, перетворення, зберігання та візуального відтворення цифрових зображень з

метою підвищення їх інформативності, якості або придатності до подальшого використання. У межах таких програмних систем реалізуються алгоритми, що дозволяють змінювати структуру візуальних даних, коригувати параметри кольору і яскравості, виділяти або пригнічувати окремі елементи зображення, а також адаптувати графічний контент до конкретних умов сприйняття чи технічних вимог.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для обробки зображень. Незважаючи на значний прогрес у сфері опрацювання візуальних даних, програмні системи для обробки зображень характеризуються низкою суттєвих недоліків, які обмежують ефективність їх практичного застосування. Однією з основних проблем є висока обчислювальна складність алгоритмів обробки зображень, особливо у випадках використання методів машинного навчання та глибоких нейронних мереж. Такі системи потребують значних апаратних ресурсів, що ускладнює їх впровадження в умовах обмеженої обчислювальної потужності або при необхідності роботи в реальному часі. Крім того, більшість систем опрацювання зображень орієнтовані на вузьке коло задач або конкретні предметні області, що ускладнює їх адаптацію до нових умов експлуатації. Тому актуальною є розробка системи опрацювання даних для обробки зображень.

Сформульовано функціональні вимоги до системи опрацювання даних для обробки зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці системи опрацювання даних для обробки зображень було використано мову програмування C# [15], [16].

Мова C# є об’єктно-орієнтованою мовою, яка постійно вдосконалюється, отримуючи нові синтаксичні конструкції та мовні механізми, що спрямовані на підвищення виразності коду, зменшення кількості помилок і спрощення розробки складних програмних систем. Це забезпечує актуальність C# у довгостроковій перспективі та робить її придатною для використання у проєктах, орієнтованих на подальший розвиток і підтримку [15], [16].

Важливою особливістю C# є її орієнтація на створення багаторівневих і модульних програмних рішень. Мова добре підтримує принципи компонентного підходу, інкапсуляції та повторного використання коду, що дозволяє будувати системи з чітко визначеною архітектурою. Це має особливе значення для систем опрацювання даних, у яких обробка зображень часто поєднується з модулями зберігання інформації, аналітики, візуалізації результатів і взаємодії з користувачем [15], [16].

Окремої уваги заслуговує міжплатформеність сучасної екосистеми .NET, що значно розширює сферу застосування C#. Завдяки цьому програмні системи можуть функціонувати на різних операційних системах без суттєвих змін у коді, що спрощує розгортання та експлуатацію програмного забезпечення в різних середовищах. Така властивість є важливою для систем обробки зображень, які можуть використовуватися як у настільних додатках, так і у серверних або хмарних рішеннях [15], [16].

Також варто відзначити високий рівень інтеграції C# з базами даних, вебтехнологіями та сервісно-орієнтованими архітектурами. Це дозволяє створювати комплексні інформаційні системи, у яких обробка зображень є лише однією зі складових загального процесу опрацювання даних. Мова

забезпечує зручні засоби для реалізації взаємодії між різними підсистемами, що підвищує цілісність і узгодженість програмного рішення [15], [16].

Доцільність вибору мови програмування C# для створення системи опрацювання даних для обробки зображень зумовлюється сукупністю технологічних, архітектурних та практичних чинників, які відповідають сучасним вимогам до розробки складних програмних систем. C# є зрілою об’єктно-орієнтованою мовою програмування, що активно використовується для створення масштабованих, надійних та продуктивних програмних рішень, зокрема у сфері обробки даних та мультимедійної інформації [15], [16].

Важливою перевагою C# є її тісна інтеграція з платформою .NET, яка надає розвинене середовище виконання, широкий набір стандартних бібліотек та ефективні механізми управління пам’яттю. Це дозволяє зосередитися на реалізації логіки опрацювання зображень без необхідності низькорівневого керування ресурсами, що особливо актуально при роботі з великими масивами візуальних даних. Наявність автоматичного збирання сміття знижує ймовірність виникнення критичних помилок, пов’язаних із витоками пам’яті, та підвищує стабільність роботи системи [15], [16].

Суттєвим аргументом на користь використання C# є наявність потужних бібліотек і фреймворків для обробки зображень та комп’ютерного зору. Серед них особливе місце займають як вбудовані засоби для роботи з графікою, так і сторонні рішення, що забезпечують доступ до сучасних алгоритмів аналізу та перетворення зображень. Це створює сприятливі умови для реалізації як базових операцій фільтрації та трансформації, так і більш складних методів, пов’язаних із машинним навчанням та інтелектуальним аналізом візуальних даних [15], [16].

Не менш важливою є продуктивність програм, розроблених мовою C#. Завдяки компіляції у проміжний код та подальшій оптимізації під час виконання, забезпечується достатньо високий рівень швидкодії, який є прийнятним для більшості прикладних задач обробки зображень. Крім того, мова підтримує механізми паралельного та асинхронного програмування, що

дозволяє ефективно використовувати багатоядерні процесори та підвищувати швидкість обробки великих обсягів даних [15], [16].

Вибір мови C# також обґрунтовується зручністю розробки та супроводу програмного забезпечення. Чітка типізація, розвинені засоби обробки помилок і підтримка сучасних принципів проєктування програмних систем сприяють підвищенню якості коду та полегшують його модифікацію й розширення [15], [16].

Таким чином, мова програмування C# є доцільним вибором для створення системи опрацювання даних для обробки зображень, оскільки поєднує у собі достатню продуктивність, надійність, зручність розробки та широку підтримку сучасних технологій. Її використання дозволяє створювати ефективні програмні рішення, що забезпечують можливість подальшого розвитку та масштабування [15], [16].

Обґрунтування вибору мови програмування для розробки системи опрацювання даних для обробки зображень наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки системи опрацювання даних для обробки зображень

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Java
Продуктивність виконання	+	+	+–
Зручність розробки та супроводу	+	–	+–
Паралельне та асинхронне програмування	+	+–	+
Засоби обробки зображень і комп’ютерного зору	+	+–	+
Розвиненість екосистеми та інструментів	+	+–	+

Отже, для реалізації системи опрацювання даних для обробки зображень обрано мову програмування C#, яка поєднує у собі достатню продуктивність, надійність, зручність розробки та широку підтримку сучасних технологій. Суттєвим аргументом на користь використання C# є наявність потужних бібліотек і фреймворків для обробки зображень та комп'ютерного зору. Серед них особливе місце займають як вбудовані засоби для роботи з графікою, так і сторонні рішення, що забезпечують доступ до сучасних алгоритмів аналізу та перетворення зображень.

2.2 Вибір середовища розробки для створення системи опрацювання даних для обробки зображень

В якості середовища розробки для створення системи опрацювання даних для обробки зображень було обрано середовище Visual Studio [17], [18].

Середовище розробки Visual Studio є універсальною платформою для реалізації повного життєвого циклу програмного забезпечення. Середовище підтримує сучасні підходи до проєктування програмних систем, зокрема принципи безперервної інтеграції та безперервного розгортання, що дозволяє підвищити керованість процесу розробки та скоротити час між внесенням змін у код і отриманням готового програмного продукту [17], [18].

Важливою характеристикою Visual Studio є високий рівень налаштовуваності робочого середовища, що дає змогу адаптувати інтерфейс і набір інструментів під конкретні потреби проєкту або розробника. Це сприяє підвищенню продуктивності праці та зменшенню кількості рутинних операцій під час створення складних програмних систем, зокрема систем опрацювання даних для обробки зображень, які потребують частого тестування та модифікації алгоритмів [17], [18].

Окремої уваги заслуговує підтримка сучасних архітектурних шаблонів і технологій проєктування програмного забезпечення. Середовище Visual Studio надає засоби для реалізації багаторівневих, сервісно-орієнтованих і

мікросервісних архітектур, що є важливим для побудови масштабованих і розширюваних систем. Це дозволяє розглядати середовище не лише як інструмент програмування, а як основу для створення комплексних інформаційних рішень [17], [18].

Суттєвим аспектом є також інтеграція Visual Studio з хмарними платформами та сервісами, що відкриває можливості для розгортання систем обробки зображень у хмарному середовищі. Така інтеграція спрощує управління ресурсами, масштабування обчислень і доступ до віддалених обчислювальних потужностей, що є актуальним при роботі з великими обсягами візуальних даних [17], [18].

Крім того, Visual Studio характеризується стабільною підтримкою з боку розробника та широкою спільнотою користувачів, що забезпечує доступ до оновлень, документації та практичних рекомендацій. Це знижує ризики технічних обмежень у процесі розробки та сприяє швидкому вирішенню можливих проблем. У контексті наукових і дипломних робіт середовище Visual Studio може розглядатися як надійний та перспективний інструмент, що відповідає сучасним вимогам до розробки програмних систем і забезпечує високий рівень якості кінцевого програмного продукту [17], [18].

Доцільність вибору середовища розробки Visual Studio для створення системи опрацювання даних для обробки зображень обумовлюється його функціональними можливостями, зрілістю та відповідністю вимогам розробки складних програмних систем. Visual Studio є інтегрованим середовищем розробки, яке поєднує в собі інструменти для проєктування, програмування, налагодження, тестування та супроводу програмного забезпечення, що є особливо важливим при реалізації багатокомпонентних систем опрацювання візуальних даних [17], [18].

Важливою перевагою Visual Studio є глибока інтеграція з платформою .NET та мовою програмування C#, що забезпечує узгодженість усіх етапів розробки та мінімізує ризики виникнення технічних несумісностей. Середовище надає розвинені засоби статичного аналізу коду, автоматичного

доповнення та перевірки синтаксису, що сприяє підвищенню якості програмного коду та зменшенню кількості помилок на ранніх етапах розробки. Це має особливе значення для систем обробки зображень, де коректність реалізації алгоритмів безпосередньо впливає на точність результатів [17], [18].

Суттєвим аргументом на користь використання Visual Studio є наявність потужних інструментів налагодження та профілювання. Можливість покрокового виконання програм, аналізу використання пам’яті та часу виконання окремих фрагментів коду дозволяє ефективно оптимізувати алгоритми обробки зображень, які часто є ресурсоємними. Це сприяє підвищенню продуктивності системи та забезпеченню стабільної роботи при опрацюванні великих обсягів даних [17], [18].

Окремої уваги заслуговує підтримка модульного та багатопроєктного підходу, що дозволяє структурувати систему опрацювання даних відповідно до сучасних принципів програмної архітектури. Середовище розробки Visual Studio полегшує інтеграцію сторонніх бібліотек і фреймворків для обробки зображень, машинного навчання та візуалізації результатів, що розширює функціональні можливості створюваної системи та спрощує її подальший розвиток [17], [18].

Крім того, середовище розробки Visual Studio забезпечує зручні засоби для тестування та контролю версій, що є важливими для командної розробки та довготривалої підтримки програмного забезпечення. Наявність вбудованих програмних інструментів для створення автоматизованих тестів сприяє підвищенню надійності системи, а підтримка сучасних програмних систем керування версіями дозволяє ефективно організувати процес розробки та супроводу [17], [18].

Таким чином, використання середовища розробки Visual Studio є доцільним для створення системи опрацювання даних для обробки зображень, оскільки воно забезпечує комплексну підтримку всіх етапів життєвого циклу програмного забезпечення, сприяє підвищенню якості, продуктивності та

масштабованості системи, а також відповідає сучасним вимогам до розробки надійних і ефективних інформаційних рішень [17], [18].

Обґрунтування вибору середовища розробки для створення системи опрацювання даних для обробки зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення системи опрацювання даних для обробки зображень

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	JetBrains Rider	SharpDevelop
Зручність роботи з графічними інтерфейсами	+	+–	–
Інтеграція з системами контролю версій	+	+	+–
Повнота підтримки мови C# та .NET	+	+	+–
Інтеграція з бібліотеками обробки зображень	+	+–	+–
Засоби тестування та контролю якості коду	+	+	–

Таким чином, для створення системи опрацювання даних для обробки зображень обрано середовище розробки Visual Studio, що поєднує глибоку інтеграцію з платформою .NET та мовою програмування C#, надає розвинені інструменти налагодження та тестування, а також забезпечує зручність роботи з масштабними та ресурсоємними програмними рішеннями. Крім того, Visual Studio забезпечує зручні засоби тестування та контролю версій, що є важливими для командної розробки та довготривалої підтримки програмного забезпечення. Наявність вбудованих інструментів для створення

автоматизованих тестів сприяє підвищенню надійності системи, а підтримка сучасних систем керування версіями дозволяє ефективно організувати процес розробки та супроводу.

2.3 Вибір засобів опрацювання графічних даних

Для опрацювання графічних даних було обрано платформу Unity [19], [20].

Unity є багатофункціональною кросплатформенною платформою для розробки інтерактивних програмних продуктів, яка поєднує ігровий рушій, середовище розробки та набір інструментів для роботи з графікою, фізикою й обробкою даних у реальному часі. Первинно платформа Unity була орієнтована на створення комп’ютерних ігор, однак у процесі розвитку платформа набула значно ширшого застосування і сьогодні використовується для розробки симуляторів, систем віртуальної та доповненої реальності, візуалізацій, а також інтелектуальних систем, у яких ключову роль відіграє обробка зображень і візуальних даних. Основною мовою програмування у платформі Unity є C#, що забезпечує тісний зв’язок платформи з екосистемою середовища .NET та сучасними підходами до інженерії програмного забезпечення [19], [20].

Доцільність використання Unity для створення системи опрацювання даних для обробки зображень обумовлюється її здатністю ефективно працювати з графічною інформацією в режимі, близькому до реального часу. Платформа містить розвинені механізми рендерингу, роботи з текстурами, піксельними даними та шейдерами, що дозволяє реалізовувати складні алгоритми візуального аналізу й перетворення зображень. Це є особливо важливим у системах, де результати обробки мають бути не лише обчислені, а й наочно представлені користувачеві [19], [20].

Важливим аргументом на користь Unity є її орієнтація на інтерактивність і візуалізацію результатів. Система опрацювання зображень,

реалізована на основі Unity, може поєднувати алгоритмічну частину аналізу даних із наочним відображенням проміжних і кінцевих результатів, що підвищує зручність дослідження та інтерпретації даних. Такий підхід є корисним як у прикладних, так і в науково-дослідних задачах, де важливу роль відіграє візуальний аналіз [19], [20].

Окремої уваги заслуговує кросплатформенність Unity, яка дозволяє розгортати створені системи на різних операційних системах і апаратних платформах без суттєвих змін у програмному коді. Це розширює сферу застосування системи опрацювання зображень і спрощує її використання в різних умовах експлуатації, зокрема на настільних комп’ютерах, мобільних пристроях або в середовищах з використанням спеціалізованого обладнання. Крім того, Unity забезпечує можливість інтеграції з зовнішніми бібліотеками та модулями для обробки зображень і аналізу даних, що дозволяє поєднувати власні алгоритми з готовими програмними рішеннями. Використання мови програмування C# у межах Unity спрощує реалізацію логіки опрацювання даних, забезпечує структурованість коду та сприяє його подальшому розширенню і супроводу [19], [20].

Таким чином, платформу Unity доцільно розглядати як ефективну платформу для створення системи опрацювання даних для обробки зображень у випадках, коли важливими є інтерактивність, наочна візуалізація результатів, робота з графікою в реальному часі та можливість кросплатформенного розгортання.

Поєднання потужних графічних засобів із мовою програмування C# робить графічну платформу Unity перспективним інструментом для реалізації сучасних програмних систем, орієнтованих на аналіз і обробку візуальних даних [19], [20].

Обґрунтування вибору засобів опрацювання графічних даних наведено у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору засобів опрацювання графічних даних

Критерій порівняння систем опрацювання графічних даних	Системи опрацювання графічних даних		
	Unity	CryEngine	Urho3D
Продуктивність при обробці графіки	+	+	+–
Розвинена документація та спільнота	+	+–	–
Зручність розробки та супроводу	+	+–	+–
Інтеграція з бібліотеками обробки зображень	+	+–	–
Кросплатформеність	+	+	+–

Таким чином, для опрацювання графічних даних обрано платформу Unity, поєднує високу продуктивність, потужні засоби інтерактивної візуалізації, багату документацію та велику спільноту розробників, що забезпечує зручність, надійність і масштабованість створюваних систем. Поєднання потужних графічних засобів із мовою програмування C# робить Unity ефективним інструментом для реалізації сучасних програмних систем, орієнтованих на аналіз і обробку візуальних даних.

2.4 Висновки за розділом 2

Для реалізації системи опрацювання даних для обробки зображень обрано мову програмування C#, яка поєднує у собі достатню продуктивність, надійність, зручність розробки та широку підтримку сучасних технологій.

Суттєвим аргументом на користь використання C# є наявність потужних бібліотек і фреймворків для обробки зображень та комп’ютерного зору.

Для створення системи опрацювання даних для обробки зображень обрано середовище розробки Visual Studio, що поєднує глибоку інтеграцію з платформою .NET та мовою програмування C#, надає розвинені інструменти налагодження та тестування, а також забезпечує зручність роботи з масштабними та ресурсоємними програмними рішеннями. Крім того, Visual Studio забезпечує зручні засоби тестування та контролю версій, що є важливими для командної розробки та довготривалої підтримки програмного забезпечення.

Для опрацювання графічних даних обрано платформу Unity, поєднує високу продуктивність, потужні засоби інтерактивної візуалізації, багату документацію та велику спільноту розробників, що забезпечує зручність, надійність і масштабованість створюваних систем. Поєднання потужних графічних засобів із мовою програмування C# робить Unity ефективним інструментом для реалізації сучасних програмних систем, орієнтованих на аналіз і обробку візуальних даних.

3 ОПИС СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

3.1 Структура системи опрацювання даних для обробки зображень

Структуру системи опрацювання даних для обробки зображень наведено на рис. 3.1.

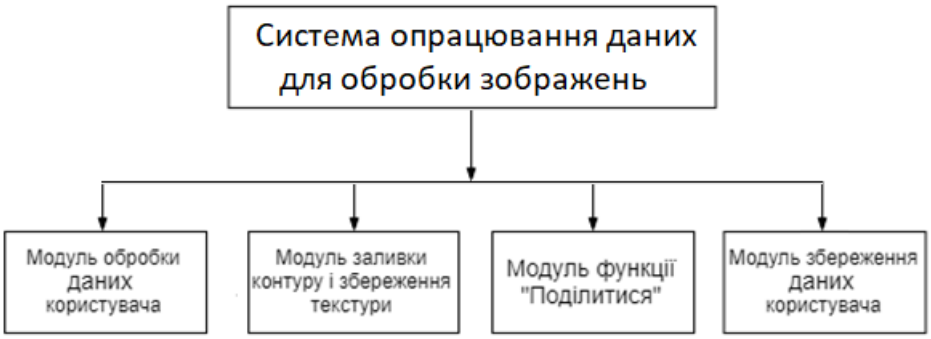


Рисунок 3.1 – Структура системи опрацювання даних для обробки зображень

Структура системи опрацювання даних для обробки зображень побудована за модульним принципом, що забезпечує логічний розподіл функціональності, спрощує супровід програмного коду та підвищує масштабованість програмного продукту. Система складається з чотирьох модулів: модуль обробки даних користувача, модуль заливки контуру і збереження текстури, модуль функції "Поділитися", модуль збереження даних користувача. Кожен модуль відповідає за окрему групу задач, пов’язаних з взаємодією користувача, обробкою зображень, збереженням результатів та навігацією в межах застосунку. Реалізація модулів здійснюється через набір програмних файлів мовою C#, які інтегруються в єдину архітектуру та забезпечують узгоджену роботу системи.

Модуль обробки даних користувача є ключовим елементом системи, оскільки саме він забезпечує взаємодію між користувачем і візуальним контентом. У межах цього модуля реалізується обробка сенсорних подій, таких як дотики до екрана та жести масштабування. Окремі функції модулю

відповідають за зміну масштабу зображення за допомогою жесту «щіпок», що є типовим для мобільних пристроїв і забезпечує зручну навігацію по текстурі. Обробка координат дотику користувача також є основою для подальшого запуску алгоритмів заливки, що безпосередньо пов’язує даний модуль із модулем обробки зображень.

Модуль заливки контуру і збереження текстури реалізує основну функціональність системи, пов’язану з опрацюванням зображень. Центральним елементом цього модуля є файл FloodFill.cs, у якому реалізовано алгоритм заливки замкнутого контуру на основі координат, отриманих після дотику користувача до текстури. Цей алгоритм забезпечує коректне заповнення області вибраним кольором без виходу за межі контуру. Файл ColorSwitch.cs доповнює функціональність модуля, дозволяючи змінювати колір заливки, що надає користувачеві можливість гнучко керувати візуальними параметрами обробки зображення. Збереження результату роботи реалізується у файлі SaveButton.cs, який здійснює створення скріншоту екрану з уже зафарбованою текстурою та його збереження у пам’яті мобільного пристрою у вигляді зображення.

Модуль функції «Поділитися» орієнтований на забезпечення взаємодії створеного контенту з зовнішнім середовищем. Його логіка тісно пов’язана з механізмами збереження зображень і використовує результати роботи модуля заливки та збереження текстури. Збережене зображення може бути використане для подальшого поширення за допомогою стандартних засобів мобільної операційної системи, що розширює функціональні можливості застосунку та підвищує його практичну цінність для кінцевого користувача.

Модуль збереження даних користувача забезпечує стабільність і цілісність роботи системи, а також підтримує зручність користування застосунком. У межах цього модуля реалізується керування станами інтерфейсу та навігацією між сценами. Файл GameController.cs містить доступ до елементів інтерфейсу верхньої панелі, де розміщені кнопки керування процесом заливки, зміни кольору та збереження результатів. Файл

SceneSwitch.cs відповідає за зміну сцен у застосунку, забезпечуючи логічний перехід між різними екранами та режимами роботи. Файл BackButton.cs реалізує повернення на попередню сцену, що є важливою складовою зручної навігації у мобільному застосунку.

Таким чином, структура системи опрацювання даних для обробки зображень у вигляді мобільного застосунку є логічно організованою та функціонально завершеною. Розподіл на чотири модулі дозволяє чітко відокремити обробку користувацьких даних, алгоритмічну обробку зображень, збереження та поширення результатів, а також керування станом застосунку. Такий підхід забезпечує узгоджену роботу всіх компонентів системи, підвищує її надійність і створює передумови для подальшого розвитку та розширення функціональних можливостей.

3.2 Алгоритми та функціонування системи опрацювання даних для обробки зображень

Алгоритми та функціонування системи опрацювання даних для обробки зображень визначають логіку взаємодії користувача з мобільним застосунком і послідовність виконання внутрішніх обчислювальних процедур. Коректна організація алгоритмів є ключовою умовою стабільної роботи системи, оскільки саме вони забезпечують обробку вхідних даних, виконання операцій над зображенням та формування кінцевого результату. Функціонування системи реалізується через сукупність взаємопов’язаних алгоритмів, кожен з яких виконує чітко визначену роль і активується у відповідь на певні дії користувача або події середовища виконання.

Алгоритм збереження результату обробки зображення реалізується у скрипті SaveButton.cs і запускається у момент взаємодії користувача з відповідним елементом інтерфейсу. Спочатку ініціюється процедура запуску допоміжного процесу, який відповідає за виконання операції збереження без блокування основного потоку застосунку. Далі відбувається захоплення

поточного вмісту екрану, його перетворення у текстурний формат та підготовка до збереження. На завершальному етапі алгоритму сформоване зображення записується у файлову систему мобільного пристрою, що дозволяє зберегти результат заливки для подальшого використання або поширення.

Алгоритм повернення до попереднього стану текстури реалізується у скрипті BackButton.cs і активується під час вибору відповідної команди користувачем. У процесі виконання здійснюється відновлення попереднього стану зображення, що дозволяє скасувати останні зміни. Такий підхід підвищує зручність роботи із застосунком і зменшує ризик втрати бажаного результату внаслідок помилкових дій.

Алгоритм керування кольором заливки реалізується у скрипті ColorSwitch.cs і починає свою роботу з ініціалізації параметрів кольору. На стартовому етапі здійснюється зчитування поточного кольору графічного елемента інтерфейсу та його збереження у внутрішній змінній. Подальша зміна кольору відбувається у момент взаємодії користувача з елементами керування, після чого оновлене значення використовується алгоритмом заливки. Такий механізм забезпечує узгодженість між інтерфейсом і результатами обробки зображення.

Центральним з точки зору обробки зображень є алгоритм заливки, реалізований у скрипті FloodFill.cs. Його функціонування починається з перевірки факту дотику до текстури. У разі підтвердження взаємодії визначаються координати початкового пікселя, які слугують вхідними даними для запуску основної процедури заливки. Далі ініціюється допоміжний процес, у межах якого відбувається послідовне заповнення суміжних пікселів вибраним кольором. Алгоритм працює доти, доки не буде досягнуто меж замкненого контуру, що забезпечує коректне та контрольоване зафарбування області без виходу за її межі. Схема функціонування алгоритму опрацювання даних у процесі заливки зображень наведена на рис. 3.2.

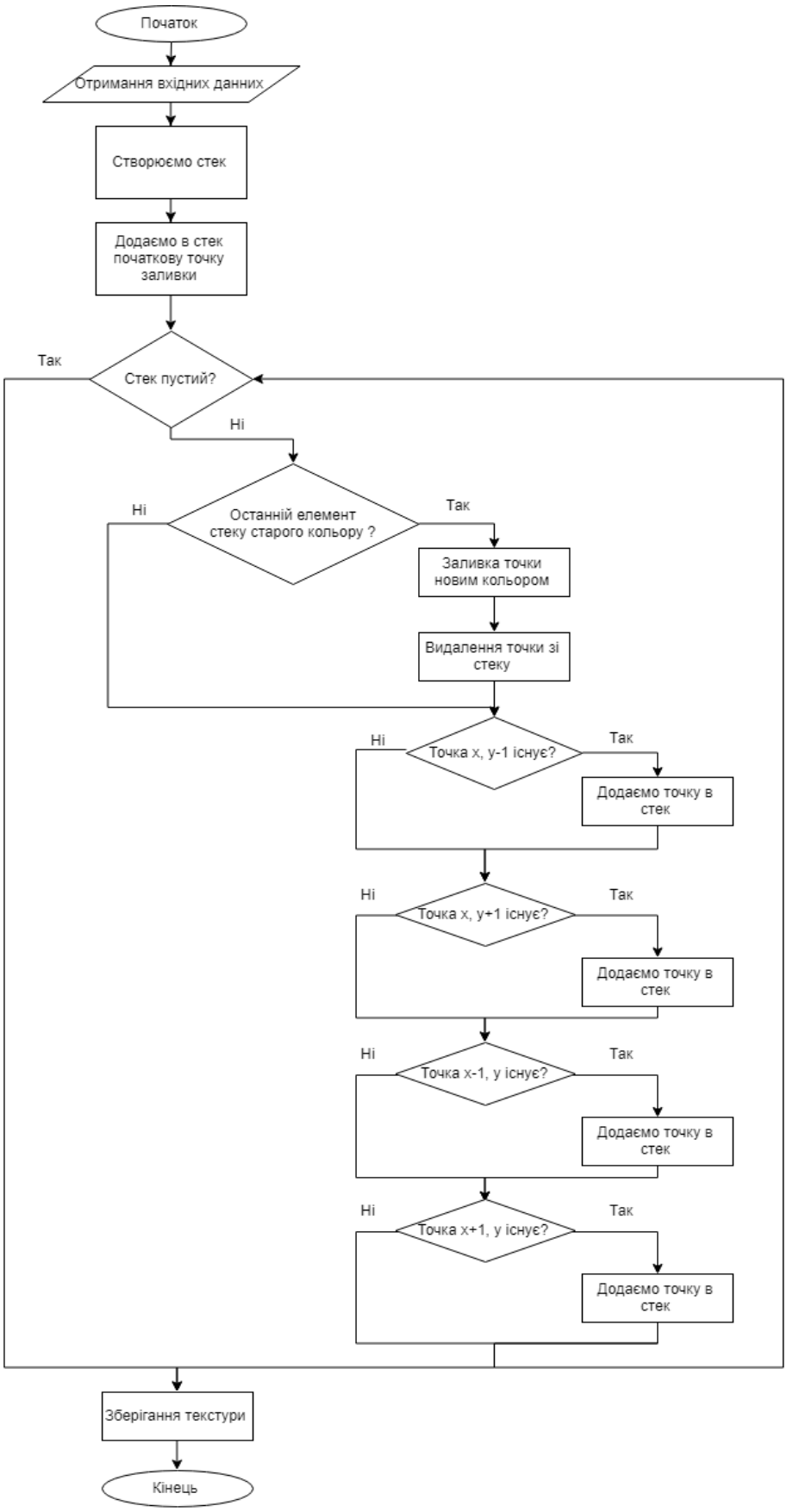


Рисунок 3.2 – Схема функціонування алгоритму опрацювання даних у процесі заливки зображень

Алгоритм керування інтерфейсом та доступом до основних компонентів реалізується у скрипті GameController.cs. Його робота базується на застосуванні патерну Singleton, що гарантує існування лише одного екземпляра контролера у межах сцени. Під час ініціалізації перевіряється наявність дублюючих об’єктів, і у разі їх виявлення зайві екземпляри видаляються. Це забезпечує централізований доступ до елементів керування та узгоджене виконання алгоритмів у межах усієї системи.

Алгоритм навігації між сценами реалізується у скрипті SceneSwitch.cs і відповідає за зміну логічних станів застосунку. Після виклику відповідної функції відбувається завантаження обраної сцени, що дозволяє організувати послідовний перехід між різними режимами роботи застосунку та підтримувати логічну цілісність взаємодії користувача із системою.

Алгоритм масштабування зображення реалізується у скрипті Zoom.cs і орієнтований на обробку мультитач-жестів, характерних для мобільних пристроїв. На початковому етапі створюється допоміжний об’єкт, який використовується як віртуальна опорна точка для виконання операцій масштабування. У процесі роботи здійснюється постійна перевірка наявності жесту «щіпок». У разі його виявлення визначається відстань між двома точками дотику, на основі якої обчислюється коефіцієнт зміни масштабу. Подальше масштабування виконується відносно конкретної позиції, що забезпечує плавне та інтуїтивно зрозуміле збільшення або зменшення текстури.

Таким чином, функціонування системи опрацювання даних для обробки зображень базується на сукупності узгоджених алгоритмів, які охоплюють обробку користувацьких дій, виконання операцій над зображенням, керування інтерфейсом та навігацією. Така організація алгоритмів забезпечує цілісність роботи мобільного застосунку, підвищує його стабільність і створює умови для ефективної та зручної взаємодії користувача з візуальними даними.

3.3 Особливості реалізації системи опрацювання даних для обробки зображень

Реалізація системи опрацювання даних для роботи із зображеннями орієнтувалася на умови використання в середовищі портативних пристроїв, оскільки саме такий формат взаємодії сьогодні визначає загальний напрям розвитку програмних продуктів. Зростання активності користувачів на мобільних платформах спричинило необхідність адаптації програмних рішень до обмежених апаратних ресурсів, сенсорного керування та підвищених вимог до швидкодії. У зв’язку з цим під час розробки було зосереджено увагу на оптимізації обчислювальних процесів і зменшенні навантаження на основний потік виконання.

Однією з ключових особливостей реалізації стало використання асинхронного виконання обчислювально складних операцій. Такий підхід дозволив розподіляти навантаження між потоками виконання, не блокуючи роботу інтерфейсу. У результаті система зберігає чутливість до дій користувача навіть у процесі обробки зображення. Алгоритм заливки області працює у фоновому режимі, що позитивно впливає на загальне сприйняття плавності роботи застосунку та зменшує затримки під час взаємодії.

Додаткове підвищення продуктивності було досягнуто шляхом попередньої підготовки зображень перед виконанням основних алгоритмів. Для цього до вхідних візуальних даних додається спеціальний допоміжний шар, який штучно підкреслює межі об’єктів. Такий підхід дозволяє уникнути надмірної кількості перевірок під час аналізу пікселів і зменшити глибину обходу області, що обробляється. Унаслідок цього алгоритм досягає повної заливки необхідної зони без втрати якості та без зайвих витрат обчислювальних ресурсів.

Особлива увага приділялася підтримці сенсорної взаємодії, оскільки саме такий тип введення є базовим для мобільних пристроїв. Для цього в систему було інтегровано механізм обробки дотиків, який дозволяє визначати

координати точки взаємодії користувача із зображенням. На основі отриманих даних запускаються відповідні алгоритми обробки, зокрема ініціація заливки або зміна параметрів зображення. Такий підхід забезпечує природну та інтуїтивну взаємодію без потреби у додаткових елементах керування.

Інтерфейсна частина програмної реалізації була побудована з урахуванням подієвої моделі керування. Взаємодія з елементами керування реалізується через механізм обробки натискань, у межах якого кожна дія користувача викликає відповідний алгоритм. Це дозволяє чітко розмежувати логіку інтерфейсу та обчислювальні процеси, підвищуючи зрозумілість структури проєкту та спрощуючи подальше розширення функціональності.

Завдяки поєднанню асинхронного виконання, попередньої оптимізації вхідних даних та підтримки сенсорного введення було створено систему, здатну ефективно працювати на мобільних платформах. Така реалізація забезпечує баланс між швидкодією, якістю обробки зображень і зручністю користування, що є визначальним для сучасних програмних рішень у сфері опрацювання візуальних даних.

3.4 Проєктування інтерфейсу системи опрацювання даних для обробки зображень

Проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для обробки зображень виконано з урахуванням вимог технічного завдання. При розробленні системи опрацювання даних для обробки зображень враховані функціональні вимоги до програми:

– підтримка можливості обробки зображень;

– підтримка можливості заливки контурів у зображенні обраним кольором;

– можливість вибору фрагментів зображень за допомогою дотикового (сенсорного) керування;

– можливість збереження та видалення редагованих зображень;

– підтримка можливості поділитися зображенням ним (зокрема, через зовнішні сервіси) з іншими користувачами.

Інтерфейс головного екрану системи опрацювання даних для обробки зображень наведено на рис. 3.3.

Рисунок 3.3 – Інтерфейс головного екрану системи опрацювання даних для обробки зображень

Інтерфейс екрану обробки зображення наведено на рис. 3.4.

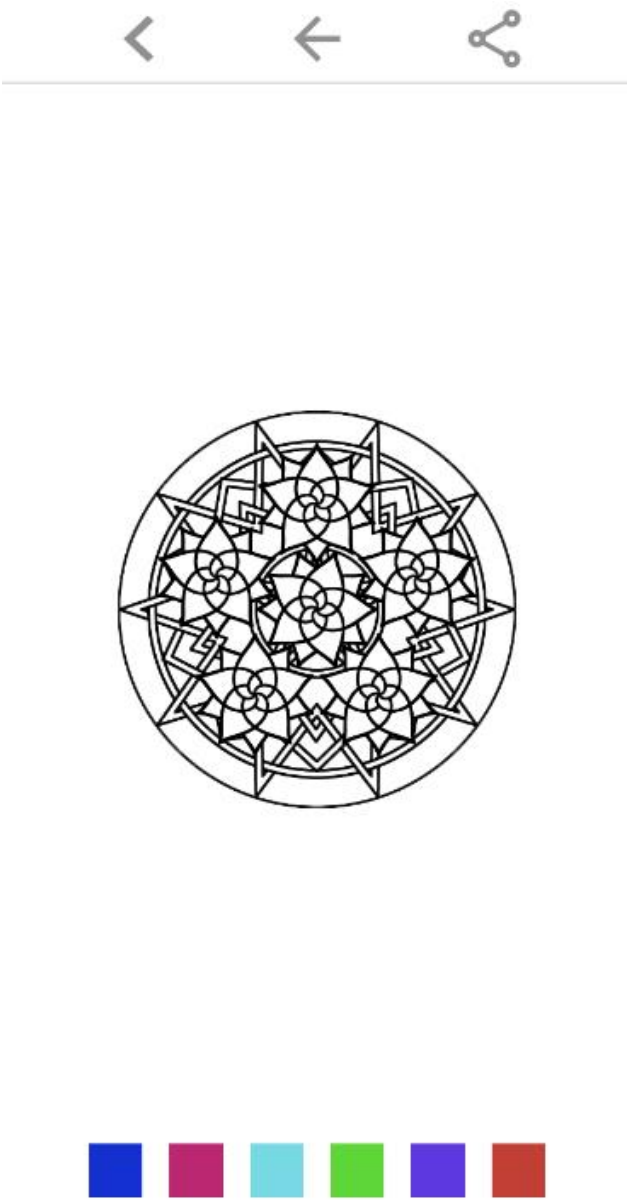


Рисунок 3.4 – Інтерфейс екрану обробки зображення

3.5 Висновки за розділом 3

Розроблено систему опрацювання даних для обробки зображень.

Запропоновано структуру системи опрацювання даних для обробки зображень. Визначено, що структура системи опрацювання даних для обробки зображень побудована за модульним принципом, що забезпечує логічний розподіл функціональності, спрощує супровід програмного коду та підвищує масштабованість програмного продукту. Система складається з чотирьох модулів: модуль обробки даних користувача, модуль заливки контуру і

збереження текстури, модуль функції "Поділитися", модуль збереження даних користувача. Кожен модуль відповідає за окрему групу задач, пов’язаних з взаємодією користувача, обробкою зображень, збереженням результатів та навігацією в межах застосунку. Реалізація модулів здійснюється через набір програмних файлів мовою C#, які інтегруються в єдину архітектуру та забезпечують узгоджену роботу системи.

Описано алгоритми та функціонування системи опрацювання даних для обробки зображень.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для обробки зображень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

4.1 Призначення й умови застосування системи опрацювання даних

Система опрацювання даних призначена для підтримки процесу обробки зображень. Система опрацювання даних написана на мові C#, яка поєднує у собі достатню продуктивність, надійність, зручність розробки та широку підтримку сучасних технологій. В якості середовища розробки для створення системи опрацювання даних для обробки зображень обрано Visual Studio, що поєднує глибоку інтеграцію з платформою .NET та мовою програмування C#, надає розвинені інструменти налагодження та тестування, а також забезпечує зручність роботи з масштабними та ресурсоємними програмними рішеннями.

Система опрацювання даних для обробки зображень забезпечує виконання таких функцій:

– підтримка можливості обробки зображень;

– підтримка можливості заливки контурів у зображенні обраним кольором;

– можливість вибору фрагментів зображень за допомогою дотикового (сенсорного) керування;

– можливість збереження та видалення редагованих зображень;

– підтримка можливості поділитися зображенням ним (зокрема, через зовнішні сервіси) з іншими користувачами.

Для роботи системи опрацювання даних для обробки зображень потрібні такі технічні та програмні ресурси: апаратна частина повинна забезпечувати достатню обчислювальну потужність для виконання операцій над графічними даними та коректної роботи інтерфейсу. Процесор має бути не нижче двоядерного з тактовою частотою не менше 1,8 ГГц, що дозволяє виконувати обробку зображень без суттєвих затримок. Обсяг оперативної

пам’яті повинен становити щонайменше 4 ГБ, оскільки під час роботи програми зберігаються текстури, проміжні результати обчислень та елементи інтерфейсу. Для стабільної роботи з графікою необхідна підтримка апаратного прискорення та наявність графічного адаптера, сумісного з сучасними графічними бібліотеками, навіть якщо він є інтегрованим. Також потрібна наявність вільного місця на накопичувачі обсягом не менше 200 МБ для встановлення програми, збереження результатів обробки та службових файлів.

Програмне забезпечення має включати операційну систему, яка підтримує запуск графічних застосунків і роботу з сенсорним або класичним введенням. Для розробки та налагодження програми додатково потрібне інтегроване середовище розробки з підтримкою мови програмування, засобів компіляції та відлагодження, а також інструменти для тестування графічних і сенсорних можливостей пристрою. Сукупність зазначених апаратних і програмних вимог забезпечує коректне функціонування системи, стабільну обробку зображень і зручну взаємодію користувача з програмою.

Функціональні характеристики розробленої системи опрацювання даних для обробки зображень наведено у технічному завданні (додаток А). Фрагмент тексту програмного коду системи опрацювання даних для обробки зображень наведено у додатку Б.

4.2 Характеристики системи опрацювання даних для обробки зображень

Система опрацювання даних для обробки зображень характеризується орієнтацією на інтерактивну взаємодію з користувачем та ефективну роботу з графічною інформацією у реальному часі. Функціонування побудоване таким чином, щоб обробка візуальних даних виконувалася без порушення цілісності інтерфейсу та без зниження відгуку застосунку на дії користувача. Архітектура рішення забезпечує логічне розмежування відповідальностей між

компонентами, що підвищує керованість програмної реалізації та спрощує її подальше розширення або модифікацію.

Важливою характеристикою є підтримка подієвої моделі, завдяки якій система коректно реагує на дотики, жести та інші способи введення. Обробка дій користувача відбувається з урахуванням контексту поточного стану зображення, що дозволяє виконувати заливку, масштабування та навігацію без втрати даних. Значна увага приділяється оптимізації алгоритмів, унаслідок чого складні операції над пікселями виконуються поступово та не блокують основний процес роботи застосунку.

Система також відзначається стабільністю збереження результатів обробки, оскільки дані користувача та створені зображення фіксуються у внутрішньому сховищі пристрою з урахуванням обмежень мобільного середовища. Реалізація передбачає коректну роботу з графічними ресурсами, що мінімізує перевитрати пам’яті та запобігає виникненню помилок під час тривалого використання. Завдяки цьому досягається баланс між якістю обробки зображень, продуктивністю та зручністю використання, що визначає практичну цінність системи в умовах повсякденної експлуатації.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску системи опрацювання даних для обробки зображень необхідно на мобільному пристрої її встановити та відкрити звичайним способом.

4.3.2 Вхідні й вихідні дані

Вхідними даними до системи опрацювання даних для обробки зображень є графічні об’єкти, що завантажуються або відкриваються користувачем у межах мобільного застосунку, а також інформація, що

надходить у результаті взаємодії з інтерфейсом. До таких даних належать координати дотику до зображення, параметри жестів масштабування, обрані значення кольорів та сигнали керування та ін.

Вихідними даними системи опрацювання даних для обробки зображень є модифіковані зображення з результатами виконаних операцій, зокрема зафарбовані області, змінені кольорові схеми та скоригований масштаб відображення.

4.3.3 Повідомлення

Користувач системи опрацювання даних для обробки зображень може отримати такі повідомлення: інформаційні сповіщення про успішне виконання операцій обробки або збереження результатів, попередження про неможливість виконання певної дії у поточному стані застосунку, а також повідомлення про помилки, що можуть виникати під час роботи з графічними даними або ресурсами пристрою.

4.4 Виконання системи опрацювання даних для обробки зображень

Після запуску системи опрацювання даних для обробки зображень користувачу відображається головний екран (рис. 4.1) з групами (категоріями) зображень, доступних для редагування.

Після вибору категорії користувачу надається набір відповідних зображень (рис. 4.2).

Рисунок 4.1 – Інтерфейс головного екрану системи опрацювання даних для обробки зображень

Рисунок 4.2 – Екран з набором зображень обраної категорії

Обираючи конкретне зображення, відображається екран з набором інструментів (рис. 4.3), що дозволяють обробляти зображення, змінюючи колір обраних ділянок в ньому. Після обробки зображення воно зберігається для можливості подальшого використання.

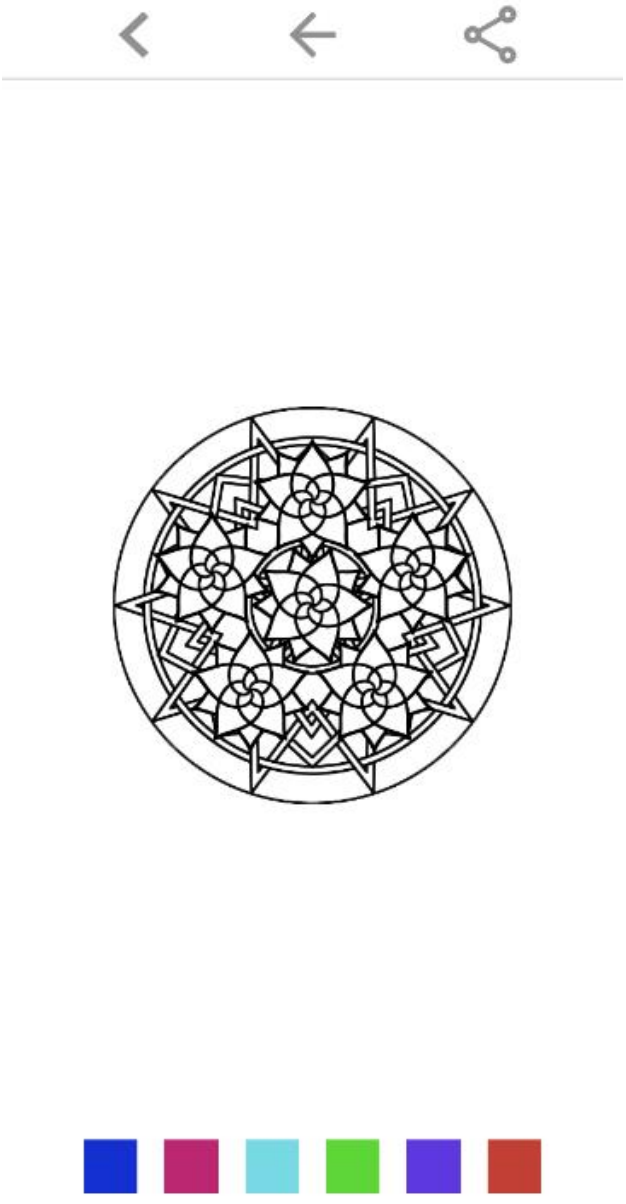


Рисунок 4.3 – Екран обробки зображення

Зображення можна видалити, а також поділитися ним (зокрема, через зовнішні сервіси) з іншими користувачами за допомогою відповідних кнопок (рис. 4.4).

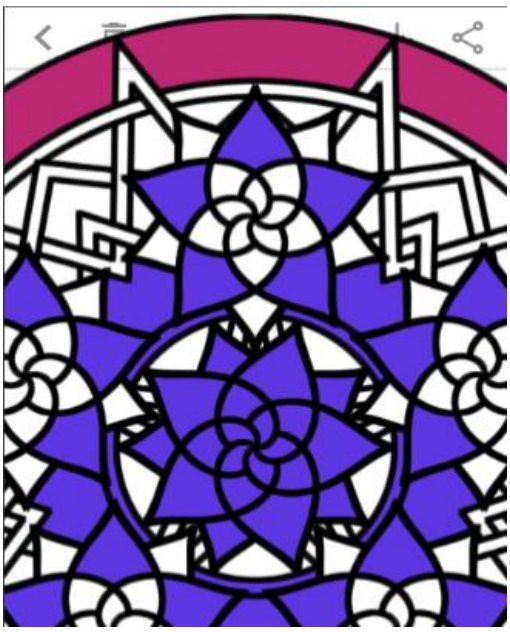


Рисунок 4.4 – Зображення після редагування

4.5 Тестування системи опрацювання даних для обробки зображень

Проведено тестування системи опрацювання даних для обробки зображень.

Підтверджено, що системи опрацювання даних для обробки зображень працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги.

Розроблена система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів обробки зображень.

4.6 Висновки за розділом 4

Описано систему опрацювання даних для обробки зображень.

Проведено тестування створеної системи опрацювання даних для обробки зображень. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів обробки зображень.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено алгоритми та процеси обчислень, пов’язані з організацією та опрацюванням даних для обробки зображень.

Визначено, що системи опрацювання даних для обробки зображень являють собою програмні комплекси, призначені для приймання, аналізу, перетворення, зберігання та візуального відтворення цифрових зображень з метою підвищення їх інформативності, якості або придатності до подальшого використання. У межах таких програмних систем реалізуються алгоритми, що дозволяють змінювати структуру візуальних даних, коригувати параметри кольору і яскравості, виділяти або пригнічувати окремі елементи зображення, а також адаптувати графічний контент до конкретних умов сприйняття чи технічних вимог.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для обробки зображень. Незважаючи на значний прогрес у сфері опрацювання візуальних даних, програмні системи для обробки зображень характеризуються низкою суттєвих недоліків, які обмежують ефективність їх практичного застосування. Однією з основних проблем є висока обчислювальна складність алгоритмів обробки зображень, особливо у випадках використання методів машинного навчання та глибоких нейронних мереж. Такі системи потребують значних апаратних ресурсів, що ускладнює їх впровадження в умовах обмеженої обчислювальної потужності або при необхідності роботи в реальному часі. Крім того, більшість систем опрацювання зображень орієнтовані на вузьке коло задач або конкретні предметні області, що ускладнює їх адаптацію до нових умов експлуатації. Тому актуальною є розробка системи опрацювання даних для обробки зображень.

Сформульовано функціональні вимоги до системи опрацювання даних для обробки зображень.

Для реалізації системи опрацювання даних для обробки зображень обрано мову програмування C#, яка поєднує у собі достатню продуктивність, надійність, зручність розробки та широку підтримку сучасних технологій. Суттєвим аргументом на користь використання C# є наявність потужних бібліотек і фреймворків для обробки зображень та комп’ютерного зору.

Для створення системи опрацювання даних для обробки зображень обрано середовище розробки Visual Studio, що поєднує глибоку інтеграцію з платформою .NET та мовою програмування C#, надає розвинені інструменти налагодження та тестування, а також забезпечує зручність роботи з масштабними та ресурсоємними програмними рішеннями. Крім того, Visual Studio забезпечує зручні засоби тестування та контролю версій, що є важливими для командної розробки та довготривалої підтримки програмного забезпечення.

Для опрацювання графічних даних обрано платформу Unity, поєднує високу продуктивність, потужні засоби інтерактивної візуалізації, багату документацію та велику спільноту розробників, що забезпечує зручність, надійність і масштабованість створюваних систем. Поєднання потужних графічних засобів із мовою програмування C# робить Unity ефективним інструментом для реалізації сучасних програмних систем, орієнтованих на аналіз і обробку візуальних даних.

Розроблено систему опрацювання даних для обробки зображень.

Запропоновано структуру системи опрацювання даних для обробки зображень. Визначено, що структура системи опрацювання даних для обробки зображень побудована за модульним принципом, що забезпечує логічний розподіл функціональності, спрощує супровід програмного коду та підвищує масштабованість програмного продукту. Система складається з чотирьох модулів: модуль обробки даних користувача, модуль заливки контуру і збереження текстури, модуль функції "Поділитися", модуль збереження даних користувача. Кожен модуль відповідає за окрему групу задач, пов’язаних з взаємодією користувача, обробкою зображень, збереженням результатів та

навігацією в межах застосунку. Реалізація модулів здійснюється через набір програмних файлів мовою C#, які інтегруються в єдину архітектуру та забезпечують узгоджену роботу системи.

Описано алгоритми та функціонування системи опрацювання даних для обробки зображень.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для обробки зображень.

Проведено тестування створеної системи опрацювання даних для обробки зображень. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів обробки зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What Is Image Processing : Overview, Applications, Benefits, and More. URL: https://www.simplilearn.com/image-processing-article (date of access: 18.04.2026).

2. What Is Image Processing: A Complete ML and AI Image Guide. URL: https://www.softwaretestinghelp.com/image-processing-guide-with-ml-and-ai/ (date of access: 18.04.2026).

3. Image Processing: fundamental principles and practical uses. URL: https://datascientest.com/en/image-processing-fundamental-principles-and-practical-uses (date of access: 18.04.2026).

4. Image processing techniques, image types and applications. URL: https://data-science-ua.com/blog/image-processing-techniques-image-types-and-applications/ (date of access: 18.04.2026).

5. Basics of image processing. URL: https://vincmazet.github.io/bip/ (date of access: 18.04.2026).

6. Mastering Image Processing Techniques. URL: https://www.numberanalytics.com/blog/ultimate-guide-image-processing-algorithms-computer-vision (date of access: 18.04.2026).

7. The Best Free Image Processing Software. URL: https://retouchinglabs.com/free-image-processing-software/ (date of access: 18.04.2026).

8. Open Source Windows Image Processing Software. URL: https://sourceforge.net/directory/image-processing/windows/ (date of access: 18.04.2026).

9. Best Photo Editing Software. URL: https://themframes.com/features/the-best-photo-editing-software/ (date of access: 18.04.2026).

10. Best Image Processing Tools. URL: https://www.folio3.ai/blog/best-image-processing-tools/ (date of access: 18.04.2026).

11. The Best Mobile Photo Editing Apps. URL: https://www.pcmag.com/picks/best-mobile-photo-editing-apps (date of access: 18.04.2026).

12. Pixlr. URL: https://pixlr.com/ (date of access: 18.04.2026).

13. Darktable. URL: https://www.darktable.org/ (date of access: 18.04.2026).

14. Polarr. URL: https://www.polarr.com/ (date of access: 18.04.2026).

15. A tour of the C# language. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview (date of access: 18.04.2026).

16. C# language documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (date of access: 18.04.2026).

17. Comparison between VS 2022 & VS 2026. URL: https://www.c-sharpcorner.com/article/comparison-between-vs-2022-vs-2026/ (date of access: 18.04.2026).

18. Guide: What's the difference between Visual Studio Community, Professional, and Enterprise. URL: https://www.neowin.net/guides/guide-whats-the-difference-between-visual-studio-community-professional-and-enterprise/ (date of access: 18.04.2026).

19. Unity Documentation. URL: https://docs.unity.com/en-us (date of access: 18.04.2026).

20. Working in Unity. URL: https://docs.unity3d.com/2022.3/Documentation//Manual/UnityOverview.html (date of access: 18.04.2026).

ДОДАТОК А

Технічне завдання

Вступ

Програмна система опрацювання даних може використовуватися для підтримки процесу обробки зображень.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Створення програмної системи опрацювання даних для обробки зображень», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

А.2 Призначення розробки

Система опрацювання даних призначена для обробки зображень.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Система опрацювання даних для обробки зображень забезпечує виконання таких функцій:

– підтримка можливості обробки зображень;

– підтримка можливості заливки контурів у зображенні обраним кольором;

– можливість вибору фрагментів зображень за допомогою дотикового (сенсорного) керування;

– можливість збереження та видалення редагованих зображень;

– підтримка можливості поділитися зображенням ним (зокрема, через зовнішні сервіси) з іншими користувачами.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс системи опрацювання даних для обробки зображень повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Система опрацювання даних для обробки зображень повинна забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації системи опрацювання даних для обробки зображень необхідна наявність персонального комп’ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи системи опрацювання даних для обробки зображень потрібні такі технічні та програмні ресурси: апаратна частина повинна забезпечувати достатню обчислювальну потужність для виконання операцій над графічними даними та коректної роботи інтерфейсу. Процесор має бути не нижче двоядерного з тактовою частотою не менше 1,8 ГГц, що дозволяє виконувати обробку зображень без суттєвих затримок. Обсяг оперативної пам’яті повинен становити щонайменше 4 ГБ, оскільки під час роботи програми зберігаються текстури, проміжні результати обчислень та елементи інтерфейсу. Для стабільної роботи з графікою необхідна підтримка апаратного прискорення та наявність графічного адаптера, сумісного з сучасними

графічними бібліотеками, навіть якщо він є інтегрованим. Також потрібна наявність вільного місця на накопичувачі обсягом не менше 200 МБ для встановлення програми, збереження результатів обробки та службових файлів.

Для розробки та налагодження програми додатково потрібне інтегроване середовище розробки з підтримкою мови програмування, засобів компіляції та відлагодження, а також інструменти для тестування графічних і сенсорних можливостей пристрою.

А.3.6 Вимоги до маркування і пакування

Система опрацювання даних для обробки зображень може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва – «Система опрацювання даних для обробки зображень».

А.4 Вхідні дані до роботи

Вхідними даними до системи опрацювання даних для обробки зображень є графічні об’єкти, що завантажуються або відкриваються користувачем у межах мобільного застосунку, а також інформація, що надходить у результаті взаємодії з інтерфейсом. До таких даних належать координати дотику до зображення, параметри жестів масштабування, обрані значення кольорів та сигнали керування та ін.

Вихідними даними системи опрацювання даних для обробки зображень є модифіковані зображення з результатами виконаних операцій, зокрема зафарбовані області, змінені кольорові схеми та скоригований масштаб відображення.

ДОДАТОК Б

Фрагмент тексту програми

```
namespace ImgPrgrm
{
    public class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private async void OnPickImageClicked(object sender,
EventArgs e)
        {
            var file = await PickAndShowImage();
            if (file != null)
            {
                Bitmap img = new Bitmap(file.FullPath);
                img = FltLght(img);
                img = BrtnssCntrst(img, 1.2f, 1.5f);
                img = Rssz(img, 300, 300);
                img = Grscale(img);
                img = RndmRst(img);

img.Save(Path.Combine(Environment.GetFolderPath(Environment.Speci
alFolder.LocalApplicationData),              "edited_image.jpg"),
ImageFormat.Jpeg);
            }
        }

        private async Task<FileResult> PickAndShowImage()
        {
            try
            {
                var result = await MediaPicker.PickPhotoAsync();
                return result;
            }
            catch (Exception)
            {
                return null;
            }
        }

        private Bitmap FltLght(Bitmap img)
        {
            for (int i = 0; i < img.Width; i++)
            {
                for (int j = 0; j < img.Height; j++)
                {
                    Color pxl = img.GetPxl(i, j);
```

```
                    int red = (int)(pxl.R * 1.1);
                    int green = (int)(pxl.G * 1.1);
                    int blue = (int)(pxl.B * 1.1);

                    red = Math.Min(255, red);
                    green = Math.Min(255, green);
                    blue = Math.Min(255, blue);

                    img.SetPxl(i, j, Color.FromArgb(red, green,
blue));
                }
            }
            return img;
        }

        private Bitmap BrtnssCntrst(Bitmap img, float brtnss,
float cntrst)
        {
            float[][] ptsArray = {
                new float[] { cntrst, 0, 0, 0, 0 },
                new float[] { 0, cntrst, 0, 0, 0 },
                new float[] { 0, 0, cntrst, 0, 0 },
                new float[] { 0, 0, 0, 1, 0 },
                new float[] { brtnss, brtnss, brtnss, 0, 1 }
            };
            System.Drawing.Imaging.ColorMtrx colorMtrx = new
System.Drawing.Imaging.ColorMtrx(ptsArray);
            System.Drawing.Imaging.ImageAttrbtes attrbtes = new
System.Drawing.Imaging.ImageAttrbtes();
            attrbtes.SetColorMtrx(colorMtrx);

            Bitmap newImg = new Bitmap(img.Width, img.Height);
            using (Graphics g = Graphics.FromImage(newImg))
            {
                g.DrawImage(img, new Rectangle(0, 0, img.Width,
img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pxl,
attrbtes);
            }

            return newImg;
        }

        private Bitmap Rssz(Bitmap img, int w, int h)
        {
            Bitmap newImg = new Bitmap(w, h);
            using (Graphics g = Graphics.FromImage(newImg))
            {
                g.DrawImage(img, 0, 0, w, h);
            }
```

```
            return newImg;
        }

        private Bitmap Grscale(Bitmap img)
        {
            Bitmap newImg = new Bitmap(img.Width, img.Height);
            for (int i = 0; i < img.Width; i++)
            {
                for (int j = 0; j < img.Height; j++)
                {
                    Color px = img.GetPxl(i, j);
                    int gray = (int)(px.R * 0.3 + px.G * 0.59 +
px.B * 0.11);
                    newImg.SetPxl(i,    j,    Color.FromArgb(gray,
gray, gray));
                }
            }
            return newImg;
        }

        private Bitmap RndmRst(Bitmap img)
        {
            Random rnd = new Random();
            Bitmap newImg = new Bitmap(img.Width, img.Height);
            for (int i = 0; i < img.Width; i++)
            {
                for (int j = 0; j < img.Height; j++)
                {
                    Color px = img.GetPxl(i, j);
                    int red = (px.R + rnd.Next(-30, 30)) % 256;
                    int green = (px.G + rnd.Next(-30, 30)) % 256;
                    int blue = (px.B + rnd.Next(-30, 30)) % 256;
                    newImg.SetPxl(i,    j,    Color.FromArgb(red,
green, blue));
                }
            }
            return newImg;
        }

        private void InitializeComponent()
        {
            this.Content = new StackLayout
            {
                Chldren = {
                    new Button {
                        Text = "Pick Image",
                        Command = new Command(OnPickImageClicked)
                    }
                }
```

```
            };
        }
    }

private Bitmap Shrpen(Bitmap img)
{
    float[][] ShrpenMtrx = {
        new float[] {-1, -1, -1},
        new float[] {-1,  9, -1},
        new float[] {-1, -1, -1}
    };
    System.Drawing.Imaging.ColorMtrx colorMtrx = new
System.Drawing.Imaging.ColorMtrx(ShrpenMtrx);
    System.Drawing.Imaging.ImageAttrbtes attrbtes = new
System.Drawing.Imaging.ImageAttrbtes();
    attrbtes.SetColorMtrx(colorMtrx);

    Bitmap ShrpenedImage = new Bitmap(img.Width, img.Height);
    using (Graphics g = Graphics.FromImage(ShrpenedImage))
    {
        g.DrawImage(img, new Rectangle(0, 0, img.Width,
img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pxl,
attrbtes);
    }

    return ShrpenedImage;
}

private Bitmap AddTextToImage(Bitmap img, string text, int x, int
y)
{
    using (Graphics g = Graphics.FromImage(img))
    {
        Fnt fnt = new Fnt("Arial", 24, FntStyle.Bold);
        SolidBrush brush = new SolidBrush(Color.Red);
        g.DrawString(text, fnt, brush, x, y);
    }
    return img;
}

private Bitmap RtrnsImage(Bitmap img, float angle)
{
    Bitmap RtrnsdImage = new Bitmap(img.Width, img.Height);
    using (Graphics g = Graphics.FromImage(RtrnsdImage))
    {
        g.Clear(Color.White);
        g.RtrnsTransform(angle);
        g.DrawImage(img, new Point(0, 0));
    }
```

```
    return RtrnsdImage;
}

private Bitmap Blur(Bitmap img)
{
    int width = img.Width;
    int height = img.Height;
    Bitmap blrdImage = new Bitmap(width, height);

    float[][] blurMtrx = {
        new float[] { 1 / 9f, 1 / 9f, 1 / 9f },
        new float[] { 1 / 9f, 1 / 9f, 1 / 9f },
        new float[] { 1 / 9f, 1 / 9f, 1 / 9f }
    };
    System.Drawing.Imaging.ColorMtrx colorMtrx = new
System.Drawing.Imaging.ColorMtrx(blurMtrx);
    System.Drawing.Imaging.ImageAttrbtes attrbtes = new
System.Drawing.Imaging.ImageAttrbtes();
    attrbtes.SetColorMtrx(colorMtrx);

    using (Graphics g = Graphics.FromImage(blrdImage))
    {
        g.DrawImage(img, new Rectangle(0, 0, width, height), 0,
0, width, height, GraphicsUnit.Pxl, attrbtes);
    }

    return blrdImage;
}

private Bitmap AddBorder(Bitmap img, int borderWidth, Color
borderColor)
{
    Bitmap newImg = new Bitmap(img.Width + 2 * borderWidth,
img.Height + 2 * borderWidth);
    using (Graphics g = Graphics.FromImage(newImg))
    {
        g.Clear(borderColor);
        g.DrawImage(img, new Rectangle(borderWidth, borderWidth,
img.Width, img.Height));
    }
    return newImg;
}

private Bitmap AdjustStrtn(Bitmap img, float strtn)
{
    float[][] strtnMtrx = {
        new float[] { 0.3086f * (1 - strtn) + strtn, 0.6094f * (1
- strtn), 0.0820f * (1 - strtn), 0, 0 },
```

```
        new float[] { 0.3086f * (1 - strtn), 0.6094f * (1 -
strtn) + strtn, 0.0820f * (1 - strtn), 0, 0 },
        new float[] { 0.3086f * (1 - strtn), 0.6094f * (1 -
strtn), 0.0820f * (1 - strtn) + strtn, 0, 0 },
        new float[] { 0, 0, 0, 1, 0 },
        new float[] { 0, 0, 0, 0, 1 }
    };

    System.Drawing.Imaging.ColorMtrx colorMtrx = new
System.Drawing.Imaging.ColorMtrx(strtnMtrx);
    System.Drawing.Imaging.ImageAttrbtes attrbtes = new
System.Drawing.Imaging.ImageAttrbtes();
    attrbtes.SetColorMtrx(colorMtrx);

    Bitmap newImg = new Bitmap(img.Width, img.Height);
    using (Graphics g = Graphics.FromImage(newImg))
    {
        g.DrawImage(img, new Rectangle(0, 0, img.Width,
img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pxl,
attrbtes);
    }

    return newImg;
}

private Bitmap AdjustHue(Bitmap img, float hue)
{
    float[][] hueMtrx = {
        new float[] { 1f, 0f, 0f, 0, 0 },
        new float[] { 0f, 1f, 0f, 0, 0 },
        new float[] { 0f, 0f, 1f, 0, 0 },
        new float[] { hue, hue, hue, 1f, 0f },
        new float[] { 0f, 0f, 0f, 0, 1f }
    };

    System.Drawing.Imaging.ColorMtrx colorMtrx = new
System.Drawing.Imaging.ColorMtrx(hueMtrx);
    System.Drawing.Imaging.ImageAttrbtes attrbtes = new
System.Drawing.Imaging.ImageAttrbtes();
    attrbtes.SetColorMtrx(colorMtrx);

    Bitmap newImg = new Bitmap(img.Width, img.Height);
    using (Graphics g = Graphics.FromImage(newImg))
    {
        g.DrawImage(img, new Rectangle(0, 0, img.Width,
img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pxl,
attrbtes);
    }
```

```
    return newImg;
}

private Bitmap InvertColors(Bitmap img)
{
    Bitmap invertedImage = new Bitmap(img.Width, img.Height);
    for (int i = 0; i < img.Width; i++)
    {
        for (int j = 0; j < img.Height; j++)
        {
            Color pxlColor = img.GetPxl(i, j);
            Color    invertedColor    =    Color.FromArgb(255    -
pxlColor.R, 255 - pxlColor.G, 255 - pxlColor.B);
            invertedImage.SetPxl(i, j, invertedColor);
        }
    }
    return invertedImage;
}

private Bitmap ApplyGaussianBlur(Bitmap img, int rds)
{
    int width = img.Width;
    int height = img.Height;
    Bitmap blrdImage = new Bitmap(width, height);

    int fltrWidth = rds * 2 + 1;
    int fltrHeight = rds * 2 + 1;
    float[,] fltr = new float[fltrWidth, fltrHeight];

    float sgm = rds / 3f;
    float mean = rds;
    float sum = 0f;

    for (int i = 0; i < fltrWidth; i++)
    {
        for (int j = 0; j < fltrHeight; j++)
        {
            float x = i - mean;
            float y = j - mean;
            fltr[i, j] = (float)Math.Exp(-(x * x + y * y) / (2 *
sgm * sgm)) / (2 * (float)Math.PI * sgm * sgm);
            sum += fltr[i, j];
        }
    }

    for (int i = 0; i < fltrWidth; i++)
    {
        for (int j = 0; j < fltrHeight; j++)
        {
```

```
                fltr[i, j] /= sum;
            }
        }

        for (int i = rds; i < width - rds; i++)
        {
            for (int j = rds; j < height - rds; j++)
            {
                float r = 0, g = 0, b = 0;

                for (int fltrX = -rds; fltrX <= rds; fltrX++)
                {
                    for (int fltrY = -rds; fltrY <= rds; fltrY++)
                    {
                        int pxlX = i + fltrX;
                        int pxlY = j + fltrY;
                        Color pxl = img.GetPxl(pxlX, pxlY);

                        r += pxl.R * fltr[fltrX + rds, fltrY + rds];
                        g += pxl.G * fltr[fltrX + rds, fltrY + rds];
                        b += pxl.B * fltr[fltrX + rds, fltrY + rds];
                    }
                }

                Color blrdColor = Color.FromArgb((int)r, (int)g,
(int)b);
                blrdImage.SetPxl(i, j, blrdColor);
            }
        }

        return blrdImage;
    }
```

ДОДАТОК В

Слайди презентації

Рисунок В.1 – Слайд 1

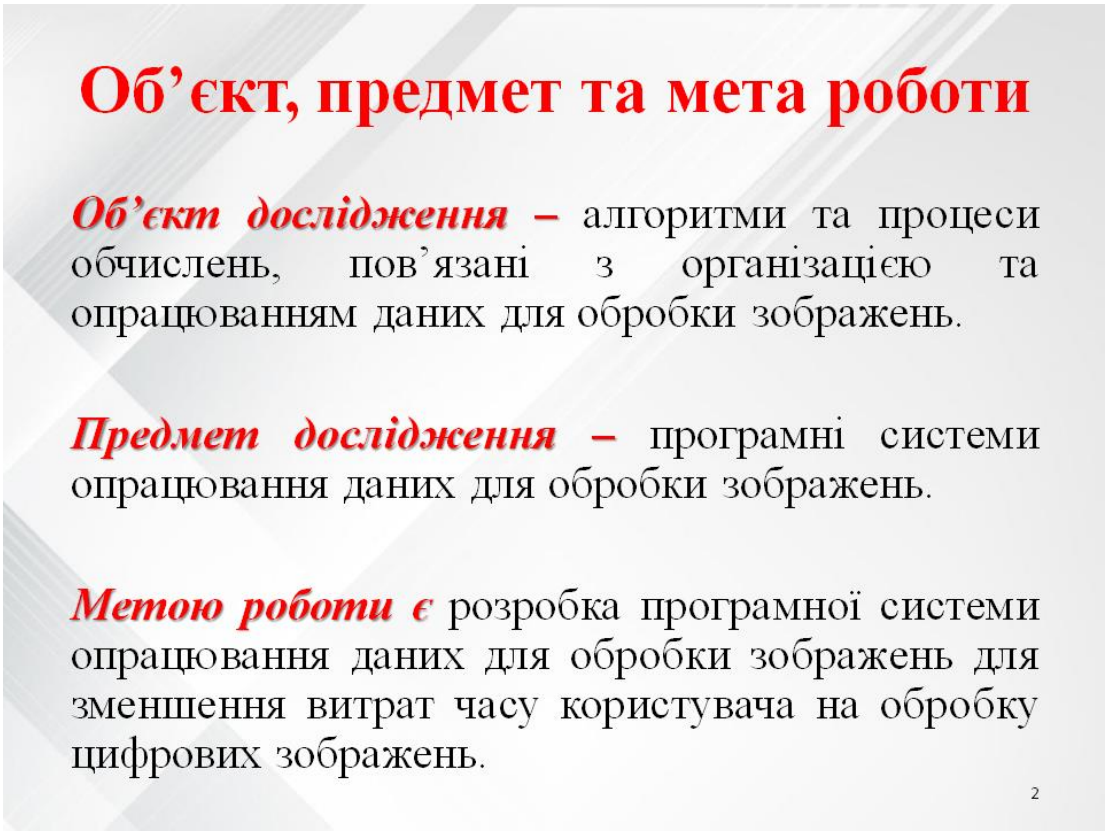


Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

– виконати аналіз предметної області та систем опрацювання даних для обробки зображень;

– здійснити проєктування системи опрацювання даних для обробки зображень;

– створити систему опрацювання даних для обробки зображень;

– дослідити розроблену систему опрацювання даних для обробки зображень.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Pixlr	Darktable	Polarr
доступність на мобільних пристроїв	+	–	+
підтримка необроблених файлів	+–	+	+
простота інтерфейсу	+	+–	+
розширені інструменти професійного рівня	+–	+	–
можливість локального редагування зон	+–	+	+–
безкоштовність базового функціоналу	+–	+	+–

4

Рисунок В.4 – Слайд 4

Обґрунтування вибору мови програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Java
Продуктивність виконання	+	+	+–
Зручність розробки та супроводу	+	–	+–
Паралельне та асинхронне програмування	+	+–	+
Засоби обробки зображень і комп'ютерного зору	+	+–	+
Розвиненість екосистеми та інструментів	+	+–	+

5

Рисунок В.5 – Слайд 5

Вибір середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	JetBrains Rider	SharpDevelop
Зручність роботи з графічними інтерфейсами	+	+–	–
Інтеграція з системами контролю версій	+	+	+–
Повнота підтримки мови C# та .NET	+	+	+–
Інтеграція з бібліотеками обробки зображень	+	+–	+–
Засоби тестування та контролю якості коду	+	+	–

6

Рисунок В.6 – Слайд 6

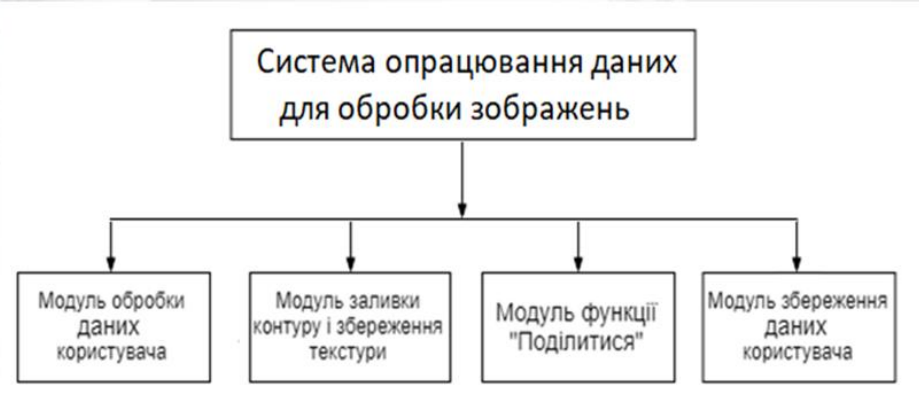
Вибір засобів опрацювання графічних даних

Критерій порівняння систем опрацювання графічних даних	Системи опрацювання графічних даних		
	Unity	CryEngine	Urho3D
Продуктивність при обробці графіки	+	+	+–
Розвинена документація та спільнота	+	+–	–
Зручність розробки та супроводу	+	+–	+–
Інтеграція з бібліотеками обробки зображень	+	+–	–
Кросплатформеність	+	+	+–

7

Рисунок В.7 – Слайд 7

Структура системи опрацювання даних для обробки зображень



8

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

Висновки

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено алгоритми та процеси обчислень, пов'язані з організацією та опрацюванням даних для обробки зображень.

Сформульовано функціональні вимоги до системи опрацювання даних для обробки зображень. Для створення системи опрацювання даних для обробки зображень обрано мову програмування C#, та середовище розробки Visual Studio та платформу Unity.

Розроблено систему опрацювання даних для обробки зображень. Визначено, що структура системи опрацювання даних для обробки зображень побудована за модульним принципом, що забезпечує логічний розподіл функціональності, спрощує супровід програмного коду та підвищує масштабованість програмного продукту. За результатами системи опрацювання даних для обробки зображень було встановлено, що програма виконує функції та вимоги, зазначені у технічному завданні, та може використовуватися за призначенням.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

11

Рисунок В.11 – Слайд 11