Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту
Завідувач кафедри комп'ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01 » червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
спеціальність 122 Комп'ютерні науки
освітня програма 122.00.01 Інформатика

Тема: ВІЗУАЛІЗАЦІЯ ГРАФІЧНИХ ЗОБРАЖЕНЬ У ФОРМАТІ ПІКСЕЛЬ-АРТУ НА МОБІЛЬНИХ ПРИСТРОЯХ

Виконала
студентка групи ІНб-2-22-4.0д
Молодецька Юлія Георгіївна
_____
(підпис)
Науковий керівник
кандидат педагогічних наук,
доцент
Глушак О.М.
_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
31 жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА
«Візуалізація графічних зображень у форматі піксель-арту на мобільних пристроях»

Виконавець – студентка групи ІНб-2-22-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Молодецька Юлія Георгіївна

1. Вихідні дані: науково-технічна література, публікації та дослідження у сфері комп'ютерної графіки, зокрема растрової графіки та піксель-арту; матеріали щодо методів обробки та візуалізації зображень; документація та ресурси, що описують технології розробки мобільних додатків (Android, графічні бібліотеки, ігрові рушії); сучасні дослідження у сфері оптимізації графіки на мобільних пристроях.
2. Основні завдання: проаналізувати сучасні підходи до візуалізації піксель-арту; дослідити особливості піксель-арту та формати графічних зображень; дослідити методи масштабування та візуалізації графіки; проаналізувати особливості відображення графіки на мобільних пристроях; розробити мобільний додаток і провести аналіз його роботи.
3. Пояснювальна записка:

Обсяг – до 54 сторінок формату А4 комп'ютерного набору з дотриманням вимог стандарту та методичних рекомендацій кафедри.

4. Графічні матеріали: не передбачено
5. Додатки: блок-схема алгоритму роботи програмного забезпечення
6. Строк подання роботи на кафедру: 01 червня 2026 р.

Науковий керівник:
кандидат педагогічних наук,
доцент
Глушак Оксана Михайлівна
31 жовтня 2025р.

Виконавець:
Молодецька Юлія Георгіївна
31 жовтня 2025р.

Календарний план

№	Етапи виконання роботи	Термін виконання	Примітка
1.	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2.	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 - 11.01.26	Виконано
3.	Аналіз технологій та інструментів розробки мобільних графічних додатків на платформі Android.	26.01.26 - 15.03.26	Виконано
4.	Дослідження методів обробки, масштабування та відображення піксельної графіки на мобільних пристроях.	16.03.26 - 31.03.26	Виконано
5.	Розробка архітектури та програмна реалізація мобільного додатку для візуалізації піксель-арту.	01.04.26 - 15.04.26	Виконано
6.	Тестування програмного забезпечення та аналіз результатів його роботи.	16.04.26 - 30.04.26	Виконано
7.	Впровадження механізмів редагування, роботи з шарами, палітрами кольорів та експорту зображень.	01.05.26 - 15.05.26	Виконано
8.	Підготовка пояснювальної записки, графічних матеріалів, додатків.	25.05.25 - 12.06.26	Виконано
9.	Захист кваліфікаційної роботи.	17.06.26	Виконано

«Затверджую»
Науковий керівник
Кандидат педагогічних наук,
доцент
Глушак Оксана Михайлівна

Індивідуальний план склала
Молодецька Ю.Г.

РЕФЕРАТ

Кваліфікаційна робота: 54 с., 12 рис., 32 посилань.

Актуальність: дослідження обумовлене широким використанням піксель-арту в мобільних додатках та іграх, а також необхідністю забезпечення якісної візуалізації зображень на пристроях з різними характеристиками екранів. Існуючі методи не завжди забезпечують чіткість та продуктивність, що потребує вдосконалення підходів до відображення графіки.

Об'єкт дослідження: процес візуалізації графічних зображень на мобільних пристроях.

Предмет дослідження: методи та засоби відображення піксель-арту з урахуванням масштабування та особливостей мобільних екранів.

Мета роботи: розробка ефективного методу візуалізації піксель-арту на мобільних пристроях із забезпеченням високої якості відображення.

Завдання:

1. проаналізувати сучасні підходи до візуалізації піксель-арту;
2. дослідити особливості піксель-арту та формати графічних зображень;
3. дослідити методи масштабування та візуалізації графіки;
4. проаналізувати особливості відображення графіки на мобільних пристроях;
5. розробити мобільний додаток і провести аналіз його роботи.

Основні результати: у роботі проведено аналіз методів візуалізації піксель-арту, визначено основні проблеми відображення на мобільних пристроях, обґрунтовано вибір методу масштабування без втрати якості, розроблено мобільний додаток для відображення піксельної графіки та проведено аналіз його роботи.

Практичне значення дослідження: полягає у можливості використання розробленого підходу та програмного рішення при створенні мобільних додатків для забезпечення якісної візуалізації піксель-арту.

Ключові слова: піксель-арт, растрова графіка, візуалізація, мобільні пристрої, масштабування, графічні зображення, обробка зображень, Android, рендеринг, оптимізація

ЗМІСТ

ВСТУП

Сучасний розвиток мобільних технологій супроводжується активним використанням графічних зображень у мобільних додатках та іграх. Одним із популярних напрямків цифрової графіки є піксель-арт, який характеризується чіткою структурою пікселів та стилізованим виглядом. Незважаючи на простоту, такий тип графіки широко застосовується завдяки своїй естетичній привабливості та низьким вимогам до обчислювальних ресурсів.

Разом із тим, відображення піксель-арту на сучасних мобільних пристроях супроводжується рядом проблем. Різноманітність характеристик екранів, таких як роздільна здатність та щільність пікселів, призводить до необхідності масштабування зображень. Неправильний вибір методів масштабування може викликати розмиття, втрату чіткості та спотворення графіки, що негативно впливає на якість візуалізації.

Аналіз сучасних досліджень та програмних рішень показує, що існуючі підходи до візуалізації піксель-арту не завжди забезпечують оптимальний баланс між якістю зображення та продуктивністю. Це обумовлює актуальність дослідження методів відображення піксельної графіки на мобільних пристроях та розробки ефективних рішень у цій сфері.

Метою роботи є розробка ефективного методу візуалізації графічних зображень у форматі піксель-арту на мобільних пристроях із забезпеченням високої якості відображення.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. проаналізувати наукові джерела та сучасні підходи до візуалізації піксель-арту;
2. дослідити особливості піксель-арту та формати графічних зображень;
3. дослідити методи масштабування та візуалізації графіки;
4. проаналізувати особливості відображення графіки на мобільних пристроях;

5. розробити та реалізувати мобільний додаток і провести аналіз його роботи.

Об’єктом дослідження є процес візуалізації графічних зображень на мобільних пристроях.

Предметом дослідження є методи та засоби відображення піксель-арту з урахуванням масштабування та особливостей мобільних екранів.

У роботі використано такі методи дослідження: аналіз наукової літератури, порівняльний аналіз існуючих методів, моделювання процесів візуалізації, програмна реалізація та експериментальна перевірка отриманих результатів.

Практичне значення одержаних результатів полягає у можливості застосування розробленого методу та програмного рішення при створенні мобільних додатків та ігор для забезпечення якісного відображення піксельної графіки.

Галузь застосування – розробка мобільних додатків, комп’ютерні ігри, системи відображення графічної інформації.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ВІЗУАЛІЗАЦІЇ ПІКСЕЛЬ-АРТУ ТА ОСОБЛИВОСТЕЙ ЇХ ЗАСТОСУВАННЯ

1.1 Піксель-арт як напрям цифрової графіки

Піксель-арт є особливим видом цифрового мистецтва, у якому зображення створюється на рівні окремих пікселів, що виступають базовими елементами композиції. У наукових дослідженнях зазначається, що піксель-арт - це вид цифрового мистецтва, у якому зображення створюються та редагуються на рівні найменших елементів - пікселів [2].

Виникнення піксель-арту безпосередньо пов’язане з технічними обмеженнями ранніх комп’ютерних систем і ігрових платформ. Зокрема, підкреслюється, що піксель-арт бере свій початок переважно з цифрових ігор [2]. Також зазначається, що він виник через обмеження апаратного забезпечення перших поколінь цифрових ігор [5].

Згідно з навчальними матеріалами та практичними посібниками, піксель-арт характеризується необхідністю точного контролю кожного пікселя, де художник працює з обмеженою палітрою та низькою роздільною здатністю зображення [1]. Це підтверджує, що кожен елемент зображення має значення для кінцевого результату.

Основною особливістю піксель-арту є його дискретна структура. У дослідженнях зазначено, що цей стиль характеризується обмеженою палітрою кольорів та чітко видимими блоками пікселів [6]. Це означає, що зображення не згладжуються, а навпаки підкреслюють свою піксельну природу.

Важливою характеристикою є також ручна точність створення зображень. У піксель-арті розміщення кожного пікселя є продуманим [5]. Це свідчить про високий рівень деталізації та контролю.

З розвитком технологій піксель-арт не зник, а трансформувався у самостійний художній стиль. У дослідженнях підкреслюється, що піксель-арт перетворився з вимушеного обмеження на свідомий художній вибір [5].

Сучасні дослідження також підтверджують його значний художній потенціал. Зокрема, зазначається, що піксель-арт є потужним інструментом художнього вираження [2], що дозволяє створювати унікальні візуальні образи.

Разом із тим, піксель-арт має специфічні технічні вимоги до обробки. Багато стандартних алгоритмів не підходять для цього стилю. Наприклад, зазначено, що жоден із методів не працює однаково добре як для фотографій, так і для малюнків чи піксель-арту [4]. Це пояснює необхідність використання спеціалізованих підходів.

Крім того, сучасні методи машинного навчання також мають обмеження при роботі з таким типом графіки. Як зазначається у дослідженні, багато традиційних моделей машинного навчання не працюють ефективно з піксель-артом, де важливий кожен окремий піксель [6].

Проблеми виникають і при масштабуванні зображень. Зокрема, зазначається, що стандартні методи дають низьку якість результату для зображень із низькою квантизацією або піксель-арту [4]. Це є важливим аргументом для розробки спеціалізованих алгоритмів візуалізації.

Таким чином, піксель-арт є не лише історичним явищем, але й сучасним напрямом цифрової графіки, що поєднує художні та технічні аспекти. Його особливості обумовлюють необхідність розробки спеціалізованих методів обробки та візуалізації, що є актуальним завданням у контексті мобільних додатків.

1.2 Методи масштабування та обробки піксель-арту

Однією з ключових проблем при роботі з піксель-артом є масштабування зображень. Оскільки цей тип графіки базується на дискретній структурі пікселів, стандартні методи обробки зображень часто призводять до втрати якості та спотворення візуального стилю.

Традиційні підходи до масштабування растрових зображень включають використання інтерполяційних методів, таких як nearest neighbor, bilinear та

bicubic. Як зазначається у наукових дослідженнях, класичний підхід до масштабування полягає у використанні лінійних фільтрів, таких як nearest neighbor та bicubic [7].

Однак такі методи розроблялися переважно для фотографічних зображень і не враховують специфіку піксель-арту. У результаті їх застосування часто призводить до розмиття зображення. Зокрема, підкреслюється, що зображення, збільшені таким способом, зазвичай страждають від розмиття чітких контурів [7].

Це пояснюється тим, що інтерполяційні алгоритми намагаються створити плавні переходи між пікселями, тоді як у піксель-арті важливо зберегти чітку межу між ними. У зв’язку з цим метод nearest neighbor є більш придатним, оскільки він не змінює значення пікселів, а просто дублює їх.

Разом із тим, навіть спеціалізовані методи масштабування мають обмеження. У дослідженнях зазначається, що спеціалізовані алгоритми масштабування піксель-арту збільшують зображення лише у фіксовану кількість разів [7]. Це обмежує їх застосування в сучасних системах, де потрібна гнучкість масштабування.

Крім того, такі алгоритми часто працюють локально, що призводить до артефактів. Зокрема, підкреслюється, що вони не здатні коректно обробляти неоднозначні структури та створюють ефект “сходинок” [7].

Окрему категорію становлять методи векторизації зображень. Зазначається, що існують підходи до перетворення піксель-арту у векторне представлення, що дозволяє масштабувати зображення без втрати якості. Зокрема, описано алгоритм створення векторного представлення, незалежного від роздільної здатності [7].

Проте такі методи також мають обмеження. Вони можуть змінювати оригінальний стиль зображення, оскільки згладжують контури та створюють нові форми. Це суперечить основній ідеї піксель-арту, де важлива точність кожного пікселя.

У сучасних дослідженнях також розглядаються методи машинного навчання для обробки зображень. Однак вони не завжди підходять для піксель-арту. Зокрема, зазначається, що багато моделей машинного навчання не працюють ефективно з піксель-артом [6].

Причина цього полягає у тому, що такі моделі орієнтовані на аналіз груп пікселів, а не окремих елементів. Це підтверджується тим, що нейронні мережі обробляють групи сусідніх пікселів разом [6], що ускладнює точну обробку піксельної графіки.

Також важливо враховувати роль кольорової палітри. У дослідженнях підкреслюється, що піксель-арт базується на обмеженій кількості кольорів, що впливає на методи обробки та відображення [3]. Обмеження палітри вимагає використання спеціальних технік, таких як dithering, які дозволяють імітувати додаткові відтінки.

У практичних посібниках також зазначається, що ефективна робота з піксель-артом передбачає контроль над масштабуванням та відключення згладжування [1]. Це є важливим аспектом при розробці програмних рішень. Таким чином, аналіз існуючих методів показує, що універсального підходу до масштабування піксель-арту не існує.

Це обумовлює необхідність розробки власного підходу до візуалізації піксель-арту, який забезпечить збереження чіткості зображення, гнучкість масштабування та ефективність використання ресурсів, що є особливо важливим для мобільних пристроїв.

Важливим аспектом масштабування піксель-арту є також проблема збереження топології зображення. У піксельній графіці навіть один піксель може представляти окремий елемент або межу об’єкта. У зв’язку з цим стандартні методи інтерполяції можуть порушувати структуру зображення. Як зазначається у дослідженнях, у піксель-арті кожен окремий піксель може бути самостійною ознакою або нести важливе значення [7].

Це означає, що будь-яке згладжування або змішування кольорів призводить до втрати інформації, яка є критичною для сприйняття

зображення. Саме тому класичні алгоритми, які працюють добре для фотографій, є непридатними для піксель-арту.

Крім того, особливу складність становить обробка діагональних з’єднань пікселів. Відповідно до досліджень пікселі з’єднані лише в одній точці [7]. Це створює неоднозначності при масштабуванні та може призводити до розривів у зображенні.

Ще однією проблемою є так званий ефект “staircasing” (ефект сходинок), який виникає при збільшенні зображення. Як зазначається у роботах, багато алгоритмів страждають від появи артефактів у вигляді сходинок [7]. Це значно погіршує візуальне сприйняття графіки.

Окремо слід розглянути питання обмеженої палітри кольорів, яка є характерною для піксель-арту. Як зазначається у дослідженнях, використання обмеженої кількості кольорів є не лише технічним обмеженням, але й важливим художнім прийомом [3]. Це означає, що алгоритми обробки повинні зберігати початкову палітру та не створювати нових відтінків.

У зв’язку з цим застосовуються спеціальні техніки, зокрема dithering, які дозволяють створювати ілюзію більшої кількості кольорів. Однак при масштабуванні такі ефекти можуть спотворюватися, що потребує додаткового врахування при розробці алгоритмів.

З точки зору програмної реалізації, важливим є також питання продуктивності. Багато сучасних методів обробки зображень є обчислювально складними та не підходять для використання у мобільних додатках. Це особливо актуально для алгоритмів, заснованих на машинному навчанні.

Як зазначається у дослідженнях, такі методи орієнтовані на складні статистичні моделі та великі обсяги даних, що обмежує їх використання в реальному часі [6]. Це підтверджує необхідність використання більш простих, але ефективних алгоритмів.

Таким чином, аналіз методів масштабування піксель-арту показує, що ключовими вимогами до них є:

- збереження структури пікселів;

- відсутність згладжування;
- підтримка довільного масштабу;
- ефективність з точки зору обчислювальних ресурсів.

Урахування цих вимог є необхідною умовою для створення якісного програмного рішення, особливо в контексті мобільних платформ, де ресурси є обмеженими.

1.3 Особливості візуалізації піксель-арту на мобільних пристроях

Візуалізація графічних зображень на мобільних пристроях має ряд специфічних особливостей, які суттєво впливають на якість відображення піксель-арту. На відміну від стаціонарних систем, мобільні пристрої характеризуються різноманітністю апаратних параметрів, обмеженими обчислювальними ресурсами та використанням адаптивних механізмів масштабування.

Однією з ключових характеристик мобільних пристроїв є щільність пікселів (DPI), яка визначає кількість пікселів на одиницю площі екрана. Високі значення DPI призводять до того, що піксель-арт, створений для низької роздільної здатності, може виглядати занадто дрібним або втрачати свою виразність. Це пов’язано з тим, що створення піксель-арту передбачає точний контроль над розташуванням пікселів [1].

У зв’язку з цим виникає необхідність масштабування зображення під конкретний пристрій. Однак стандартні механізми масштабування, що використовуються в мобільних операційних системах, часто застосовують згладжування, яке спотворює структуру піксель-арту. Це узгоджується з результатами досліджень, де зазначається, що традиційні методи обробки призводять до розмиття чітких меж [7].

Ще однією важливою проблемою є автоматичне масштабування інтерфейсу, яке виконується системою. У багатьох мобільних середовищах зображення масштабуються відповідно до параметрів екрана без урахування

їх типу. Це означає, що піксель-арт може бути оброблений як звичайне зображення, що призводить до втрати його характерних властивостей.

Крім того, мобільні пристрої мають обмежені обчислювальні ресурси, що впливає на вибір алгоритмів візуалізації. Як зазначається у дослідженнях, складні алгоритми обробки зображень, зокрема засновані на машинному навчанні, можуть бути неефективними в умовах обмежених ресурсів [6]. Це означає, що при розробці мобільних додатків необхідно враховувати баланс між якістю та продуктивністю.

Значну роль також відіграє тип графічного API, що використовується у мобільному додатку. Наприклад, при використанні графічних бібліотек або рушіїв часто за замовчуванням застосовується згладжування текстур. Це суперечить принципам піксель-арту, де важливо зберегти чіткість пікселів.

У дослідженнях також підкреслюється, що піксель-арт має специфічні вимоги до відображення через свою структуру. Враховуючи важливість кожного пікселя, навіть незначні зміни при рендерингу можуть суттєво вплинути на сприйняття зображення.

Окрему увагу слід приділити питанню адаптації піксель-арту до різних розмірів екранів. У сучасних мобільних пристроях існує велика кількість варіантів роздільної здатності, що ускладнює забезпечення однакової якості відображення. У зв’язку з цим виникає необхідність використання адаптивних підходів до масштабування.

З точки зору дизайну інтерфейсів, піксель-арт має ряд переваг. Як зазначається у дослідженнях, він є ефективним засобом створення візуального стилю, який легко впізнається та не потребує значних ресурсів [2]. Це робить його привабливим для використання у мобільних додатках та іграх.

Разом із тим, важливо враховувати, що піксель-арт створювався для низької роздільної здатності, і його пряме використання на сучасних пристроях може бути неефективним. Як зазначається у дослідженнях, сучасні технології можуть змінювати візуальні характеристики піксель-арту, що вимагає адаптації методів його відображення [5].

Ще одним аспектом є використання текстур у графічних системах. При відображенні піксель-арту як текстури можуть виникати артефакти, пов’язані з фільтрацією та інтерполяцією. Це вимагає налаштування параметрів рендерингу, зокрема вимкнення згладжування та використання точкового масштабування.

Таким чином, можна виділити основні фактори, що впливають на візуалізацію піксель-арту на мобільних пристроях:

- щільність пікселів екрану;
- автоматичне масштабування системою;
- використання згладжування;
- обмеження продуктивності;
- різноманітність роздільних здатностей.

Урахування цих факторів є необхідною умовою для забезпечення якісного відображення піксель-арту. Саме тому розробка спеціалізованих методів візуалізації є актуальним завданням, особливо у контексті створення мобільних додатків.

Додатково слід враховувати особливості апаратного прискорення графіки на мобільних пристроях. Сучасні мобільні системи активно використовують графічні процесори (GPU) для рендерингу зображень. Однак ці системи зазвичай оптимізовані для обробки текстур із використанням інтерполяції, що може негативно впливати на якість піксель-арту.

При використанні стандартних механізмів рендерингу текстур відбувається автоматичне згладжування, яке змінює початкову структуру зображення. Це особливо критично для піксель-арту, оскільки, як зазначається у дослідженнях, кожен піксель може бути окремою значущою частиною зображення [7]. Таким чином, навіть незначне згладжування призводить до втрати інформації.

Крім того, у процесі рендерингу важливу роль відіграє спосіб представлення зображення у вигляді текстури. При масштабуванні текстур можуть застосовуватися різні фільтри, зокрема лінійна або анізотропна

фільтрація. Ці методи добре підходять для фотографічних зображень, проте для піксель-арту вони є небажаними, оскільки змінюють чіткість контурів.

У дослідженнях зазначається, що класичні підходи до обробки зображень не враховують особливості піксельної графіки, оскільки вони не враховують специфіку вихідних даних [7]. Це означає, що вони не здатні зберігати структуру пікселів без додаткових налаштувань.

Ще одним важливим аспектом є масштабування інтерфейсу користувача (UI scaling). У мобільних операційних системах застосовуються механізми автоматичного масштабування інтерфейсних елементів, що забезпечують адаптацію до різних розмірів екранів. Проте ці механізми не враховують специфіку піксель-арту, що призводить до його спотворення.

Зокрема, при масштабуванні UI-елементів система може змінювати їх розміри з використанням інтерполяції, що викликає розмиття. Це особливо критично для елементів інтерфейсу, створених у стилі піксель-арту, таких як іконки, кнопки та шрифти.

Крім того, важливим є питання синхронізації логічних та фізичних пікселів. У сучасних мобільних пристроях використовується поняття незалежних від щільності пікселів (dp), що дозволяє адаптувати інтерфейс до різних екранів. Проте при роботі з піксель-артом це може призводити до невідповідності між логічною та фізичною сіткою пікселів.

Це, у свою чергу, може викликати артефакти при відображенні, зокрема розмиття або нерівномірне масштабування. Для уникнення таких проблем необхідно забезпечити кратність масштабування, що відповідає цілочисельним значенням.

Важливим аспектом є енергоспоживання мобільних пристроїв. Використання складних алгоритмів обробки зображень може призводити до підвищеного навантаження на процесор і графічний модуль, що негативно впливає на автономність пристрою. Це обмежує можливість застосування ресурсомістких методів.

У зв’язку з цим доцільним є використання оптимізованих алгоритмів, які забезпечують достатню якість при мінімальних витратах ресурсів. Це відповідає загальним вимогам до мобільних додатків, де важливими є продуктивність та енергоефективність.

Також слід враховувати, що піксель-арт часто використовується у динамічних сценах, зокрема в іграх. Це означає, що візуалізація повинна відбуватися в реальному часі. У таких умовах навіть незначні затримки або додаткові обчислення можуть впливати на плавність анімації.

У дослідженнях також підкреслюється, що традиційні алгоритми масштабування можуть не забезпечувати стабільної якості при зміні масштабу. Зокрема, вони можуть давати низьку якість результату для піксель-арту [4].

Таким чином, ефективна візуалізація піксель-арту на мобільних пристроях потребує врахування не лише графічних, але й апаратних, програмних та енергетичних факторів. Це значно ускладнює задачу та обґрунтовує необхідність розробки спеціалізованих рішень.

1.4 Аналіз існуючих підходів та постановка задачі дослідження

У попередніх підрозділах було розглянуто основні характеристики піксель-арту, методи його обробки та особливості відображення на мобільних пристроях. Проведений аналіз показав, що, незважаючи на широке використання піксель-арту в сучасних цифрових продуктах, існує ряд невирішених проблем, пов’язаних із його візуалізацією.

Однією з ключових проблем є використання універсальних методів обробки зображень, які не враховують специфіку піксель-арту. Як зазначається у дослідженнях, класичні алгоритми масштабування призводять до розмиття чітких меж [7], що є неприйнятним для піксельної графіки.

Крім того, сучасні алгоритми обробки зображень орієнтовані переважно на фотографічні дані. У зв’язку з цим підкреслюється, що жоден із методів не працює однаково добре як для фотографій, так і для піксель-арту [4]. Це

свідчить про відсутність універсального підходу до обробки різних типів графіки.

Окрему групу становлять спеціалізовані алгоритми масштабування піксель-арту. Хоча вони дозволяють покращити якість зображення, їх можливості є обмеженими. Зокрема, зазначається, що такі алгоритми збільшують зображення лише у фіксовану кількість разів [7], що не відповідає вимогам сучасних мобільних систем.

Також важливою проблемою є обробка структурних особливостей піксель-арту. У дослідженнях підкреслюється, що кожен піксель може бути окремою значущою одиницею [6]. Це означає, що навіть незначні зміни можуть призвести до втрати змісту зображення.

Методи векторизації пропонують альтернативний підхід до вирішення проблеми масштабування. Зокрема, у дослідженні [7] описано алгоритм, що дозволяє отримати представлення, незалежне від роздільної здатності. Проте такі підходи змінюють вихідний стиль зображення та не завжди зберігають його автентичність.

Сучасні методи на основі машинного навчання також мають обмеження. Згідно з дослідженням, багато моделей не працюють ефективно з піксель-артом, оскільки важливим є кожен окремий піксель [6]. Це пояснюється тим, що такі методи орієнтовані на узагальнення, тоді як піксель-арт потребує точності.

Крім того, значну роль відіграють особливості мобільних пристроїв. Як було показано раніше, автоматичне масштабування, використання згладжування та різноманітність апаратних характеристик призводять до додаткових труднощів при відображенні піксель-арту.

Узагальнюючи результати аналізу, можна виділити основні недоліки існуючих підходів:

- використання алгоритмів, не пристосованих до піксель-арту;
- втрата чіткості зображення при масштабуванні;
- обмеженість спеціалізованих алгоритмів;

- невідповідність методів обробки умовам мобільних пристроїв;
- висока обчислювальна складність сучасних методів.

З іншого боку, піксель-арт залишається актуальним і широко використовується у сучасних цифрових продуктах. Як зазначається у дослідженнях, він є потужним інструментом художнього вираження [2], що обумовлює необхідність його подальшого дослідження.

Таким чином, існує суперечність між популярністю піксель-арту та недостатнім рівнем розвитку методів його візуалізації на мобільних пристроях.

Виходячи з проведеного аналізу, можна сформулювати основну задачу дослідження: розробка ефективного підходу до візуалізації піксель-арту на мобільних пристроях, який забезпечить збереження чіткості зображення, гнучкість масштабування та високу продуктивність.

У рамках даної роботи передбачається розробка мобільного додатку, який реалізує обрані методи візуалізації. Основними вимогами до розроблюваного рішення є:

- збереження структури пікселів без згладжування;
- підтримка масштабування без втрати якості;
- адаптація до різних параметрів екранів;
- ефективне використання ресурсів мобільного пристрою.

Реалізація такого підходу дозволить підвищити якість відображення піксель-арту та забезпечити його ефективне використання у мобільних додатках.

Додатково слід розглянути питання узгодженості між різними етапами обробки та відображення піксель-арту. У сучасних програмних системах процес візуалізації включає кілька рівнів: зберігання зображення, його обробку, передачу в графічний конвеєр та фінальне відображення на екрані. На кожному з цих етапів можуть виникати втрати якості, якщо не враховуються особливості піксельної графіки.

Зокрема, при передачі зображення в графічний конвеєр може відбуватися автоматичне перетворення форматів або застосування фільтрів, що змінюють початкові дані. Це суперечить принципу піксель-арту, оскільки важливим є кожен окремий піксель [6], що підкреслює необхідність збереження точності на всіх етапах обробки.

Ще одним аспектом є проблема узгодження різних масштабів у межах одного додатку. У мобільних системах можуть одночасно використовуватися різні елементи інтерфейсу, що мають різну роздільну здатність. Це створює складність при інтеграції піксель-арту у загальну систему відображення.

Також важливо враховувати особливості форматів зображень. Наприклад, використання форматів із втратами може призводити до спотворення кольорів і появи артефактів. У той же час формати без втрат забезпечують кращу якість, але можуть мати більший обсяг даних. Це створює необхідність вибору оптимального компромісу між якістю та продуктивністю.

З точки зору архітектури програмного рішення, доцільним є вибір підходу до організації рендерингу. Існують два основні підходи: використання високорівневих графічних бібліотек або пряме використання низькорівневих API. Перший варіант є простішим у реалізації, але може не забезпечувати необхідного контролю над процесом відображення. Другий варіант дозволяє більш точно керувати параметрами рендерингу, але потребує більш складної реалізації.

Крім того, необхідно враховувати питання сумісності між різними пристроями. Різні моделі мобільних пристроїв можуть мати відмінності у реалізації графічних підсистем, що впливає на результат відображення. Це означає, що розроблюване рішення повинно бути універсальним та адаптивним.

У контексті даного дослідження важливим є також питання визначення критеріїв оцінки якості візуалізації. До таких критеріїв можна віднести:

- чіткість відображення пікселів;
- відсутність артефактів;

- відповідність оригінальному зображенню;
- продуктивність роботи додатку.

Ці критерії дозволяють об’єктивно оцінити ефективність запропонованого рішення та порівняти його з існуючими підходами.

Таким чином, проведений аналіз показує, що проблема візуалізації піксель-арту є комплексною і включає не лише алгоритмічні, але й апаратні та програмні аспекти. Існуючі підходи не забезпечують повного вирішення цієї задачі, що обумовлює необхідність розробки нового підходу.

У результаті можна уточнити постановку задачі дослідження: необхідно розробити мобільний додаток, який забезпечить коректне відображення піксель-арту з урахуванням особливостей мобільних пристроїв, збереженням структури пікселів та оптимізацією продуктивності.

Розв’язання цієї задачі дозволить підвищити якість графічного відображення у мобільних додатках і розширити можливості використання піксель-арту в сучасних цифрових продуктах.

Висновки до розділу 1

У першому розділі було проаналізовано особливості піксель-арту як напряму цифрової графіки та розглянуто основні методи його обробки і масштабування. Встановлено, що використання традиційних алгоритмів масштабування не завжди забезпечує збереження структури пікселів і може призводити до погіршення якості зображення.

Також досліджено особливості відображення піксельної графіки на мобільних пристроях та визначено основні фактори, що впливають на якість візуалізації.

На основі проведеного аналізу сформульовано основні вимоги до програмного рішення для візуалізації піксель-арту та обґрунтовано необхідність розробки мобільного додатку, який забезпечуватиме коректне масштабування, збереження структури пікселів і ефективне використання ресурсів пристрою.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ ВІЗУАЛІЗАЦІЇ ПІКСЕЛЬ-АРТУ

2.1 Вибір методів масштабування та візуалізації піксель-арту

У процесі розробки програмного рішення для візуалізації піксель-арту на мобільних пристроях одним із ключових етапів є вибір методів масштабування та відображення графічних зображень. Як було встановлено у першому розділі, піксель-арт має специфічні властивості, пов’язані з дискретною структурою зображення та значущістю кожного окремого пікселя, що обумовлює необхідність застосування спеціалізованих підходів до його обробки.

Класичні методи масштабування растрових зображень базуються на використанні інтерполяційних алгоритмів, які широко описані у працях з цифрової обробки зображень [9; 10]. До таких методів належать найближчий сусід (nearest neighbor), білінійна та бікубічна інтерполяція. Вони забезпечують плавне збільшення зображень, однак не враховують особливості піксель-арту, що призводить до розмиття контурів та втрати характерної стилістики.

Згідно з фундаментальними дослідженнями у сфері комп’ютерної графіки, застосування реконструкційних фільтрів при масштабуванні призводить до згладжування зображення та зміни його початкових характеристик [13]. Це є допустимим для фотографічних зображень, однак у випадку піксель-арту така обробка є небажаною, оскільки порушує чіткість піксельної структури.

Альтернативою є використання методу найближчого сусіда, який не виконує інтерполяцію значень пікселів, а просто дублює їх. Такий підхід дозволяє зберегти початкову структуру зображення, що є критично важливим для піксель-арту. Водночас він має обмеження, зокрема можливість появи “сходинкових” артефактів при збільшенні масштабу.

У сучасних дослідженнях також розглядаються підходи, засновані на використанні нейронних мереж для обробки зображень [11]. Проте такі методи орієнтовані на покращення якості фотографічних даних та не забезпечують збереження дискретної структури піксель-арту. Крім того, їх застосування у мобільних додатках обмежене високими обчислювальними витратами.

Важливим аспектом є також спосіб представлення графічних зображень у програмному середовищі. У мобільних додатках, зокрема на платформі Android, для відображення графіки широко використовуються механізми Canvas та Drawable [14]. Вони забезпечують гнучкі можливості для роботи з 2D-графікою, дозволяючи здійснювати масштабування, трансформацію та відображення зображень у реальному часі.

Згідно з офіційною документацією Android, використання Canvas дозволяє безпосередньо керувати процесом рендерингу та відключати згладжування, що є критично важливим для коректного відображення піксель-арту [14]. Це робить даний підхід одним із найбільш доцільних для реалізації поставленої задачі.

Крім того, архітектура графічної підсистеми Android передбачає використання апаратного прискорення, що описано у відповідних документах [12]. Це дозволяє підвищити продуктивність відображення графіки, однак вимагає врахування особливостей роботи графічного конвеєра, зокрема автоматичного застосування фільтрації текстур.

У зв’язку з цим важливим є контроль параметрів відображення, зокрема відключення згладжування та використання цілочисельного масштабування. Як показано у роботах з комп’ютерної графіки, правильний вибір методу масштабування суттєво впливає на якість кінцевого зображення [8].

Також слід враховувати питання продуктивності мобільних пристроїв. Згідно з рекомендаціями Google, ефективна робота графічних додатків повинна забезпечувати баланс між якістю відображення та використанням

ресурсів [16]. Це означає, що вибрані методи повинні бути не лише якісними, але й оптимізованими з точки зору швидкодії.

Таким чином, проведений аналіз дозволяє зробити висновок, що для задачі візуалізації піксель-арту на мобільних пристроях найбільш доцільними є прості алгоритми масштабування у поєднанні з контролем параметрів рендерингу. Обраний підхід забезпечує збереження якості зображення та ефективність роботи програмного рішення.

2.2 Обґрунтування вибору технологій та інструментів розробки мобільного додатку.

У процесі розробки програмного рішення для візуалізації піксель-арту важливим етапом є вибір технологій та інструментів реалізації. Від правильності цього вибору залежить не лише функціональність додатку, але й продуктивність, зручність використання та якість відображення графічної інформації.

Для реалізації поставленої задачі було обрано платформу Android, що обумовлено її широким розповсюдженням та наявністю розвинених засобів для роботи з графікою. Згідно з офіційною документацією, Android надає набір інструментів для обробки та відображення графічних зображень, зокрема через використання графічного конвеєра та апаратного прискорення [12].

У якості основного інструменту для роботи з графікою було використано механізм Canvas, який забезпечує можливість створення та редагування двовимірних зображень у реальному часі. Відповідно до документації Android Developers, Canvas дозволяє виконувати операції малювання, масштабування та трансформації зображень, що є необхідним для реалізації функціоналу піксель-арту [14].

Вибір Canvas як основного інструменту обумовлений тим, що він забезпечує достатній рівень контролю над процесом відображення, зокрема дозволяє відключати згладжування та працювати з окремими пікселями. Це є критично важливим для збереження характерної структури піксель-арту.

Альтернативою використанню Canvas є застосування графічних API нижчого рівня, таких як OpenGL. Однак їх використання у даному випадку є недоцільним, оскільки вони потребують більш складної реалізації та не дають суттєвих переваг для задач двовимірної графіки. Таким чином, обраний підхід дозволяє досягти оптимального балансу між складністю реалізації та функціональними можливостями.

Для реалізації імпорту зображень було використано стандартні механізми Android для вибору файлів із галереї. Це дозволяє користувачу швидко завантажити необхідне зображення без використання сторонніх бібліотек.

2.3 Архітектура та функціональна організація мобільного додатку

У процесі розробки мобільного додатку для візуалізації піксель-арту було визначено загальну архітектуру програмного рішення, яка забезпечує зручність використання, модульність та ефективність роботи. Архітектура додатку побудована за принципом поділу функціональності на окремі логічні компоненти, що спрощує розробку, тестування та подальше розширення системи.

Загалом додаток можна представити як сукупність взаємопов’язаних модулів, кожен з яких відповідає за виконання певної функції. Основними компонентами системи є:

- модуль імпорту зображень;
- модуль обробки та масштабування;
- модуль відображення графіки;
- модуль редагування;
- модуль управління шарами;
- модуль збереження та експорту.

Структурну модель програмного забезпечення мобільного додатку наведено на рис. 2.1.

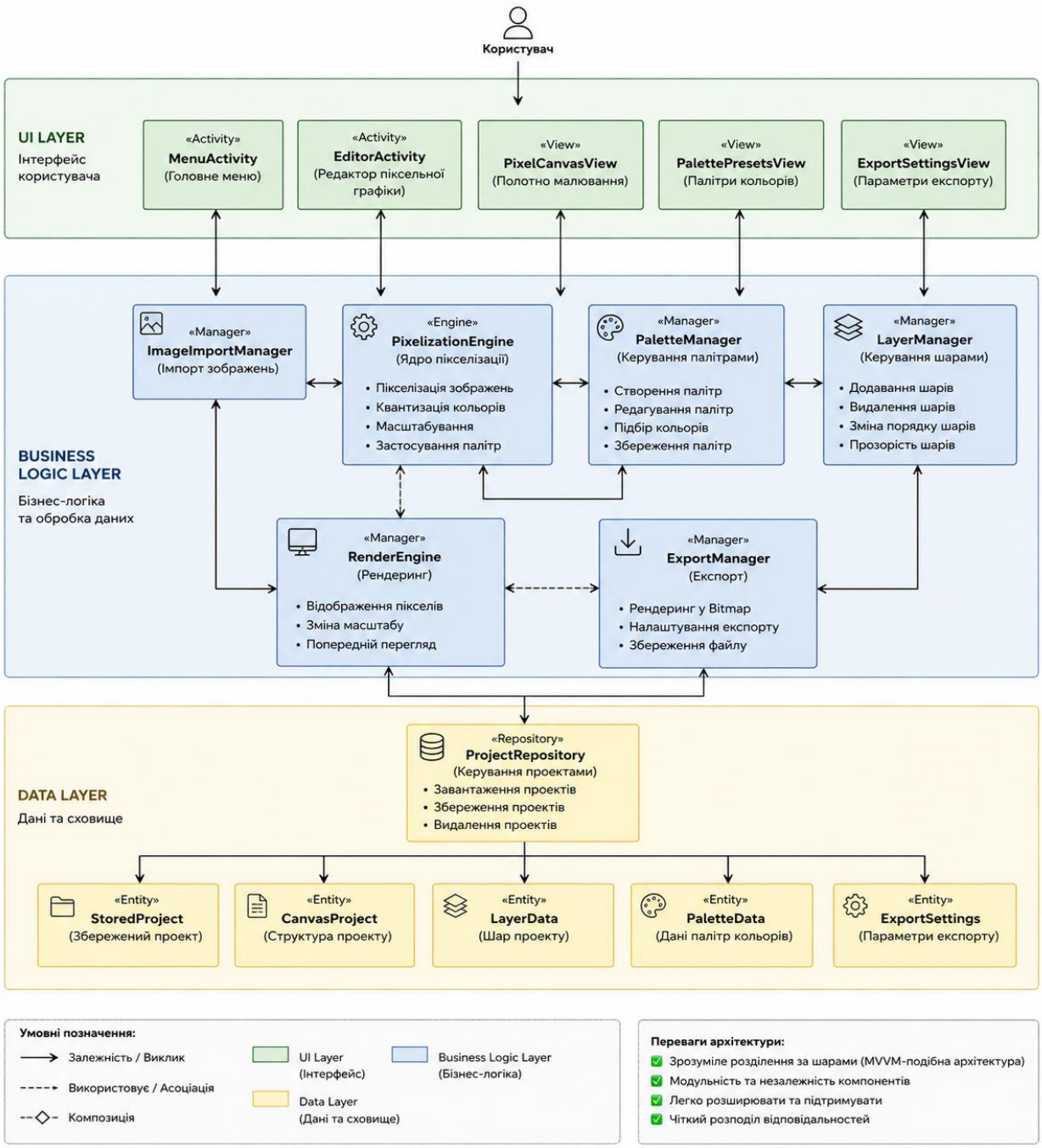


Рисунок 2.1. Модель структури програмного забезпечення

Структурна модель програмного забезпечення мобільного додатку для візуалізації графічних зображень у форматі піксель-арту побудована за принципом багатошарової архітектури, що забезпечує чіткий розподіл функцій між компонентами системи. Такий підхід дозволяє відокремити інтерфейс користувача від бізнес-логіки та механізмів збереження даних, що позитивно впливає на масштабованість, підтримку та модифікацію

програмного забезпечення. Архітектура системи складається з трьох основних рівнів: UI Layer, Business Logic Layer та Data Layer.

Рівень UI Layer відповідає за взаємодію користувача із мобільним застосунком. Саме цей рівень містить усі візуальні компоненти та екрани, через які користувач виконує роботу із піксельною графікою. Центральним елементом є MenuActivity, що виконує функції головного меню та забезпечує навігацію між основними модулями системи. Через нього користувач отримує доступ до редактора, налаштувань експорту та керування проєктами.

Основним робочим середовищем системи виступає EditorActivity, який реалізує функціонал редактора піксельної графіки. Саме тут відбувається взаємодія користувача із механізмами пікселізації, палітрами кольорів, шарами та інструментами редагування. Даний компонент виступає проміжною ланкою між інтерфейсом користувача та бізнес-логікою системи.

Компонент PixelCanvasView реалізує полотно малювання та забезпечує відображення піксельної структури зображення. Через нього виконуються масштабування, переміщення полотна, попередній перегляд та інтерактивне редагування графіки. Для роботи з кольорами використовується PalettePresetsView, який дозволяє обирати готові палітри, створювати власні набори кольорів та редагувати існуючі палітри. Компонент ExportSettingsView відповідає за налаштування параметрів експорту, зокрема вибір формату файлу, параметрів якості та роздільної здатності.

Другий рівень архітектури – Business Logic Layer – є ядром системи, у якому реалізовано основні алгоритми обробки даних. Центральним компонентом цього рівня виступає PixelizationEngine, що забезпечує виконання алгоритмів перетворення звичайних растрових зображень у формат піксель-арту. Даний модуль виконує пікселізацію зображень, квантізацію кольорів, масштабування та адаптацію палітр. Саме PixelizationEngine реалізує ключову функціональність програмного забезпечення, тому він має найбільшу кількість взаємозв’язків з іншими компонентами системи.

Модуль ImageImportManager відповідає за імпорт графічних файлів у систему. Він забезпечує завантаження зображень, підтримку популярних графічних форматів та передачу даних до ядра пікселізації. Це дозволяє ізолювати процеси роботи з файлами від інших компонентів архітектури.

Для роботи з кольоровими палітрами використовується PaletteManager. Його функціональність включає створення, редагування, збереження та підбір палітр кольорів. Наявність окремого менеджера палітр є важливою для систем піксельної графіки, оскільки палітра визначає стилістику та художню цілісність кінцевого результату.

Компонент LayerManager реалізує механізм роботи із шарами. Він дозволяє додавати нові шари, змінювати порядок їх відображення, керувати прозорістю та видаляти непотрібні елементи. Підтримка багатошарової структури свідчить про наближеність функціоналу застосунку до професійних графічних редакторів.

RenderEngine відповідає за рендеринг графічного вмісту та відображення пікселів на екрані мобільного пристрою. Через цей модуль реалізуються оновлення полотна, попередній перегляд та масштабування сцени. Він виконує роль проміжного механізму між алгоритмами обробки даних та інтерфейсом користувача.

Окремим компонентом бізнес-логіки є ExportManager, який забезпечує формування фінального результату та його збереження. Модуль відповідає за рендеринг у Bitmap, налаштування параметрів експорту та запис готових файлів у пам’ять пристрою.

Третій рівень архітектури – Data Layer – відповідає за збереження та організацію даних. Центральним елементом цього рівня є ProjectRepository, який реалізує механізм роботи з проєктами. Через нього виконуються операції збереження, завантаження та видалення проєктів. Використання репозиторію дозволяє відокремити логіку доступу до даних від бізнес-логіки системи та забезпечує гнучкість при зміні способу зберігання інформації.

У Data Layer також представлено набір сутностей, що описують структуру даних системи. StoredProject містить інформацію про збережені проєкти користувача. CanvasProject описує параметри полотна та структуру редактора. LayerData зберігає інформацію про окремі шари, їхній порядок та прозорість. PaletteData містить дані палітр кольорів, а ExportSettings описує параметри експорту готових зображень.

Важливою особливістю представленої структурної моделі є чітке розмежування відповідальностей між компонентами. Кожен модуль виконує окрему функцію та взаємодіє з іншими компонентами лише через визначені зв’язки. Такий підхід забезпечує слабку зв’язаність системи та високу когезію окремих модулів. Це значно спрощує підтримку програмного забезпечення, тестування компонентів та подальше розширення функціоналу.

Архітектура моделі відповідає сучасним принципам проєктування мобільних застосунків, зокрема концепціям Clean Architecture, Repository Pattern та Separation of Concerns. Завдяки цьому система є гнучкою, модульною та придатною до масштабування. Представлена структурна модель може ефективно використовуватися як основа для розробки сучасного мобільного редактора піксельної графіки та є достатньо обґрунтованою для використання у дипломному або науковому проєкті.

Після аналізу структурної моделі програмного забезпечення доцільно перейти до розгляду функціональної структури мобільного додатку, оскільки саме вона відображає набір основних функцій системи та взаємозв’язки між окремими підсистемами з точки зору виконуваних користувачем дій. Якщо структурна модель демонструє внутрішню організацію програмного забезпечення та взаємодію його компонентів, то функціональна структура описує прикладні можливості мобільного застосунку та логіку реалізації основних процесів обробки піксельної графіки.

Модуль імпорту відповідає за отримання зображення з галереї пристрою. У процесі роботи використовується стандартний механізм вибору файлів, що дозволяє користувачу завантажити зображення у форматах JPG або

PNG. Після вибору зображення воно перетворюється у внутрішній формат представлення (Bitmap), який використовується для подальшої обробки.

Модуль обробки та масштабування

Після імпорту зображення передається до модуля обробки, де виконується його масштабування та підготовка до роботи у форматі піксель-арту. У цьому модулі реалізовано алгоритми зміни розміру зображення з урахуванням збереження піксельної структури.

Крім того, тут виконується приведення кольорів до обраної палітри. Це дозволяє зменшити кількість кольорів та забезпечити відповідність стилю піксель-арту.

Модуль відображення графіки

Модуль відображення реалізований із використанням Canvas та відповідає за виведення зображення на екран. Основною задачею цього модуля є забезпечення коректного відображення пікселів без згладжування.

У цьому модулі також реалізовано:

- масштабування зображення для відображення на екрані;
- відображення сітки пікселів;
- оновлення зображення в реальному часі при внесенні змін.

Модуль редагування

Модуль редагування забезпечує можливість внесення змін до зображення. До його функцій належать:

- інструмент заливки;
- інструмент гумки;
- малювання окремих пікселів;
- скасування попередніх дій (undo).

Реалізація undo базується на збереженні історії змін, що дозволяє повертатися до попередніх станів зображення.

Модуль управління шарами

Для розширення можливостей редагування було реалізовано систему шарів. Кожен шар містить окреме зображення, яке може бути незалежно змінене. Користувач може:

- створювати нові шари;
- змінювати їх порядок;
- вмикати або вимикати видимість;
- видаляти шари.

Такий підхід дозволяє виконувати складніші операції редагування без ризику пошкодження основного зображення.

Модуль збереження та експорту

Модуль збереження відповідає за зберігання проєктів у внутрішній пам’яті додатку. Це дозволяє користувачу повертатися до редагування у будь-який момент.

Модуль експорту забезпечує збереження готового зображення у форматі PNG без втрат якості. При цьому користувач може задати параметри експорту, зокрема розмір та інші характеристики зображення.

Функціональна структура мобільного додатку відображає основні можливості системи та взаємозв’язок між окремими функціональними підсистемами. Вона наведена на рис. 2.2

Рисунок 2.2. Функціональна структура мобільного додатку

Функціональна структура мобільного додатку для візуалізації графічних зображень у форматі піксель-арту побудована за модульним принципом і охоплює основні напрями роботи системи: керування проєктами, пікселізацію зображень, редагування графіки, роботу із шарами, керування палітрами, візуалізацію полотна та додаткові сервісні функції. Такий підхід дозволяє систематизувати функціональні можливості застосунку та забезпечити зручність взаємодії користувача з програмою.

Перший функціональний модуль відповідає за роботу з проєктами. У межах цього модуля реалізуються створення нового проєкту, відкриття існуючих файлів, збереження результатів роботи та механізм автозбереження. Також передбачено імпорт графічних зображень і можливість експорту готового результату у формат PNG. Наявність даного функціонального блоку забезпечує повний цикл роботи користувача із проєктом – від створення до збереження фінального зображення.

Другий модуль реалізує функції пікселізації зображення та є ключовим у контексті роботи застосунку. Спочатку відбувається імпорт вихідного растрового зображення, після чого система виконує зменшення роздільної здатності для формування піксельної структури. Далі застосовується квантізація кольорів, яка дозволяє обмежити кількість кольорів відповідно до обраної палітри. Після цього здійснюється застосування палітри та масштабування пікселів, у результаті чого формується кінцеве зображення у стилі піксель-арту. Даний модуль реалізує основну алгоритмічну частину програмного забезпечення та забезпечує автоматизоване перетворення графіки.

Третій функціональний блок призначений для редагування зображення. У його складі передбачено інструменти пензля пікселів, гумки, заливки області та переміщення елементів. Також система підтримує функцію скасування дій, що підвищує зручність редагування та дозволяє користувачу повертатися до попередніх станів проєкту. Цей модуль забезпечує

інтерактивну взаємодію користувача із піксельною графікою та реалізує базові можливості графічного редактора.

Окремий модуль відповідає за роботу із шарами. У ньому реалізовано створення нових шарів, їх видалення, переміщення та зміну порядку відображення. Крім того, підтримується функція блокування шарів, що дозволяє уникнути випадкового редагування окремих елементів композиції. Наявність багатошарової структури значно розширює можливості редагування та наближує функціонал застосунку до професійних графічних редакторів.

Модуль керування палітрами забезпечує роботу з кольоровими схемами піксель-арту. Користувач може обирати готові палітри, створювати власні набори кольорів, редагувати палітри та виконувати їх імпорт або експорт. Використання окремого функціонального блоку для палітр є особливо важливим для стилізації піксельної графіки, оскільки палітра визначає художній стиль та візуальну цілісність зображення.

Функціональний блок візуалізації та масштабування відповідає за відображення графічного полотна на екрані мобільного пристрою. У його межах реалізовано масштабування полотна, переміщення робочої області та відображення сітки пікселів. Це забезпечує точність редагування та покращує зручність роботи з деталізованими піксельними зображеннями.

Останній модуль містить додаткові функції системи. До нього входять управління пам’яттю проєктів, оптимізація продуктивності та підтримка різних співвідношень сторін. Ці функції спрямовані на підвищення стабільності роботи мобільного застосунку, ефективне використання ресурсів мобільного пристрою та адаптацію системи до різних форматів екранів.

Таким чином, функціональна структура мобільного додатку охоплює всі основні процеси роботи із піксельною графікою та забезпечує повний цикл створення, редагування, візуалізації та збереження піксель-арту. Побудова системи за модульним принципом забезпечує логічне групування функцій,

спрощує розширення програмного забезпечення та підвищує ефективність його подальшого супроводу й модернізації.

Загальна схема роботи додатку

Взаємодію модулів можна описати наступним чином:

- користувач імпортує зображення;
- зображення обробляється та масштабується;
- відображається на екрані;
- користувач виконує редагування;
- результат зберігається або експортується.

Алгоритм роботи програмного забезпечення наведено у Додатку А

Особливості реалізації

При розробці додатку було враховано ряд важливих аспектів:

- використання цілочисельного масштабування для збереження структури пікселів;
- відключення згладжування при відображенні;
- оптимізація роботи з пам’яттю;
- забезпечення швидкого оновлення інтерфейсу

Ці особливості дозволяють досягти високої якості візуалізації та стабільної роботи додатку.

Програмна реалізація мобільного додатку базується на об’єктно-орієнтованому підході, що забезпечує модульність, повторне використання компонентів та спрощує підтримку програмного забезпечення. Основні функції системи реалізовані у вигляді окремих класів, кожен з яких відповідає за певну частину функціональності додатку.

Для наочного представлення взаємозв’язків між основними класами програмного забезпечення було побудовано UML-діаграму класів, наведену на рис. 2.3.

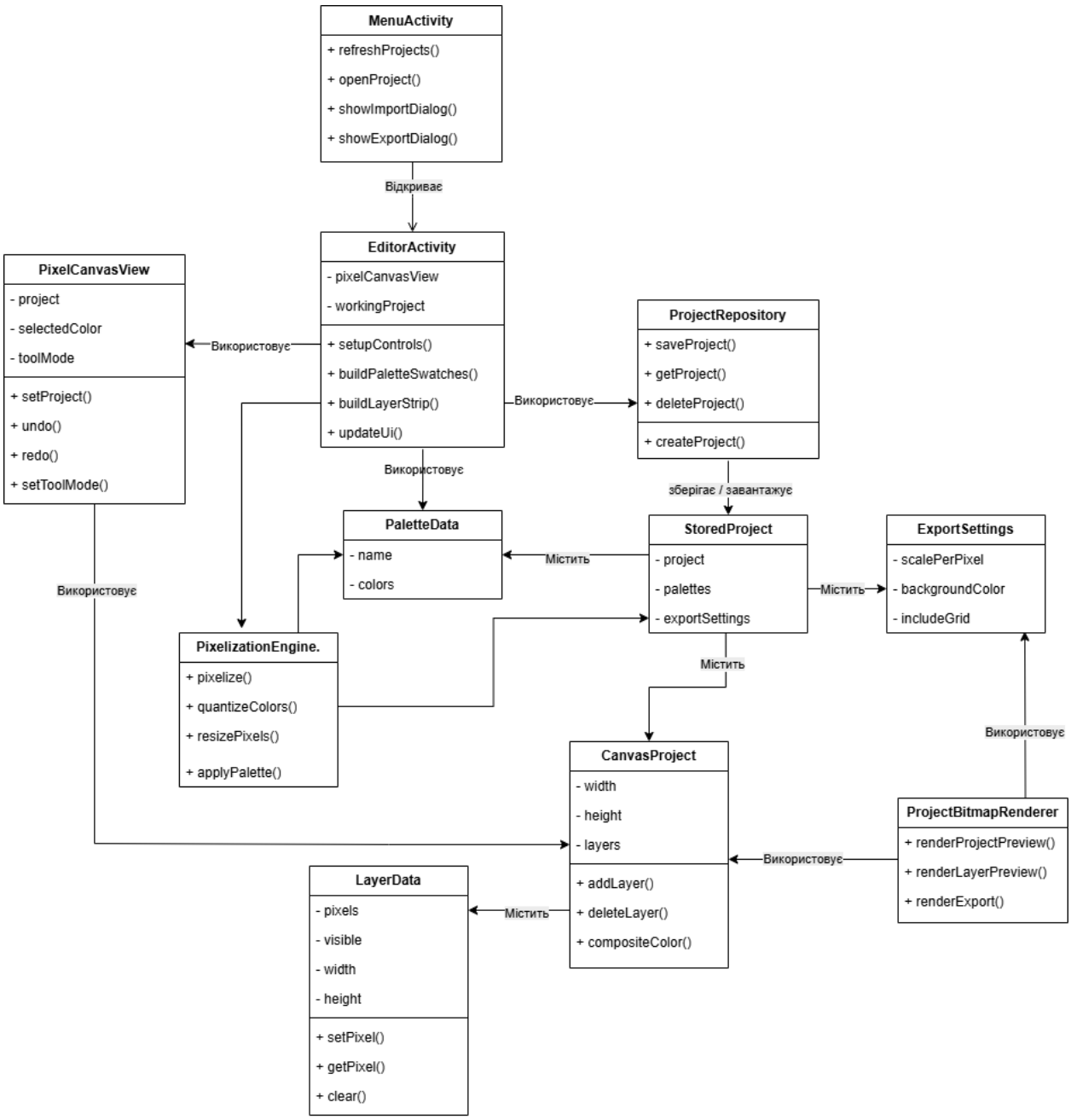


Рисунок 2.3. Модель класів мобільного додатку

На рисунку 2.3 представлено UML-діаграму класів мобільного додатку для візуалізації піксель-арту. Центральним класом системи є EditorActivity, який забезпечує взаємодію користувача з редактором та координує роботу основних компонентів додатку.

Для роботи з полотном малювання використовується клас PixelCanvasView, який взаємодіє зі структурою проєкту CanvasProject.

Зберігання та завантаження проєктів реалізовано у класі ProjectRepository, що працює з контейнером даних StoredProject.

Окремі класи відповідають за керування палітрами (PaletteData), шарами (LayerData), параметрами експорту (ExportSettings) та рендерингом зображення (ProjectBitmapRenderer). Для реалізації алгоритмів пікселізації використовується клас PixelizationEngine, який забезпечує масштабування, квантизацію кольорів та застосування палітр.

Для забезпечення модульності та спрощення підтримки програмного забезпечення архітектуру мобільного додатку було організовано у вигляді окремих взаємопов’язаних компонентів.

UML-діаграму компонентів програмного забезпечення наведено на рис. 2.4.

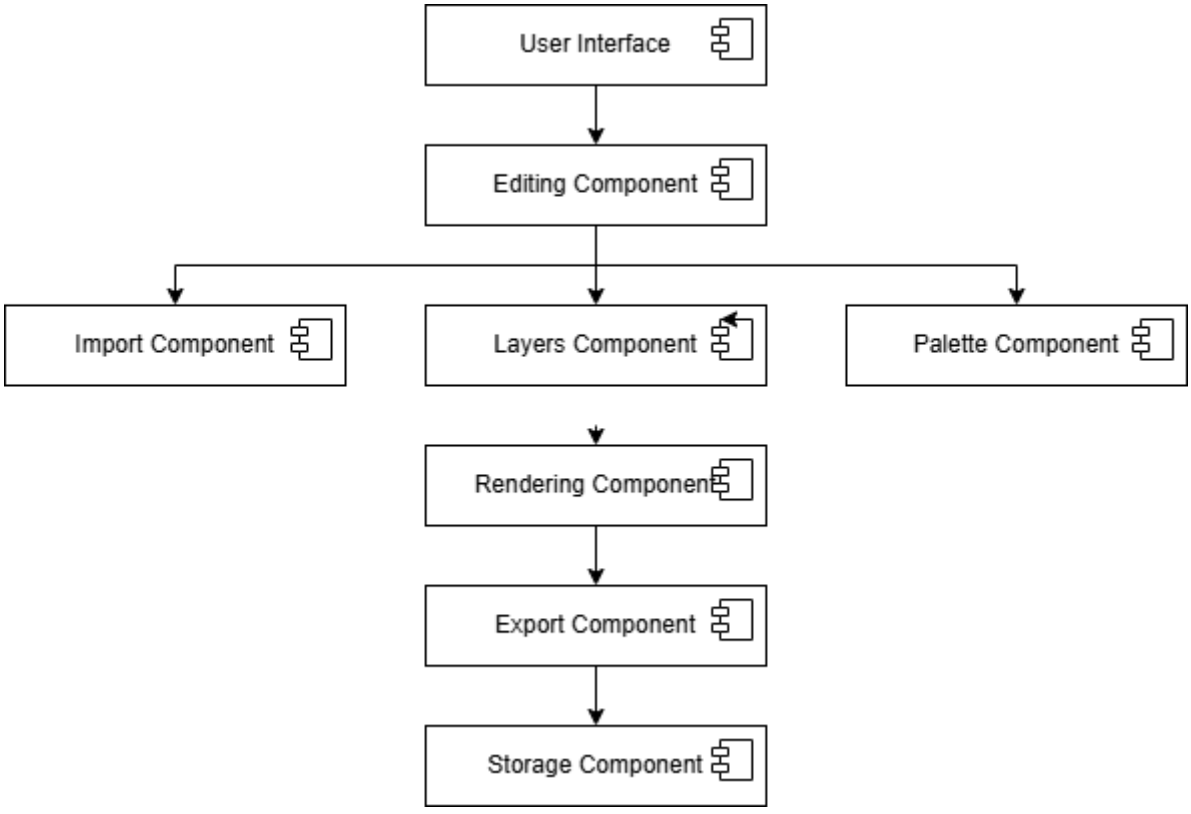


Рисунок 2.4. Компоненти програмного забезпечення

На рисунку 2.4 представлено UML-діаграму компонентів програмного забезпечення мобільного додатку для візуалізації піксель-арту. Архітектура системи побудована за модульним принципом і складається з окремих

компонентів, що відповідають за реалізацію різних функцій програмного забезпечення.

Компонент User Interface забезпечує взаємодію користувача з мобільним додатком. Основна логіка редагування реалізована у компоненті Editing Component, який взаємодіє з компонентами імпорту зображень, керування шарами та палітрами кольорів.

Відображення результату роботи виконується у компоненті Rendering Component, а експорт готового зображення — у компоненті Export Component. Для збереження даних проєкту використовується Storage Component. Такий підхід забезпечує незалежність окремих підсистем, спрощує підтримку програмного забезпечення та дозволяє розширювати функціональність мобільного додатку.

Таким чином, розроблена архітектура мобільного додатку забезпечує модульність, гнучкість та зручність подальшого розширення програмного забезпечення. Використання багаторівневої структури, поділу функціональності на окремі компоненти та об’єктно-орієнтованого підходу дозволило реалізувати ефективний мобільний редактор для роботи з піксельною графікою.

Висновки до розділу 2

У другому розділі обґрунтовано вибір методів масштабування та технологій реалізації мобільного додатку для роботи з піксельною графікою. Визначено доцільність використання платформи Android та механізмів Canvas і Bitmap для забезпечення коректного відображення пікселів без згладжування.

Крім того, розроблено структурну модель програмного забезпечення, функціональну структуру додатку, UML-діаграми класів і компонентів, а також алгоритм роботи системи. Це дозволило сформувати цілісне уявлення про архітектуру програмного забезпечення та підготувати основу для його практичної реалізації.

РОЗДІЛ 3.
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВІЗУАЛІЗАЦІЇ ГРАФІЧНИХ ЗОБРАЖЕНЬУ ФОРМАТІ ПІКСЕЛЬ-АРТУ

3.1Реалізація основних функцій мобільного додатку

У межах виконання дипломної роботи було реалізовано мобільний додаток для візуалізації та редагування піксель-арту на платформі Android. Програмне забезпечення реалізує функції імпорту зображень, пікселізації, масштабування, роботи з палітрами кольорів, шарами та експорту готових результатів.

Під час розробки було використано механізми Canvas та Bitmap, що забезпечують можливість ефективної роботи з растровою графікою та відображення пікселів без згладжування [14; 19]. Для реалізації масштабування було використано підходи, орієнтовані на збереження структури піксель-арту та уникнення спотворення контурів [13; 20].

У даному розділі представлено результати практичної реалізації мобільного додатку, особливості програмної реалізації окремих функцій, а також аналіз отриманих результатів.

Інтерфейс мобільного додатку було реалізовано з урахуванням особливостей роботи з піксельною графікою на мобільних пристроях. Під час розробки основну увагу було приділено забезпеченню зручності взаємодії користувача з полотном редагування, швидкому доступу до інструментів та коректному відображенню графічних елементів при різних рівнях масштабування. Реалізований інтерфейс дозволяє виконувати основні операції редагування у реальному часі без помітних затримок.

Робота з мобільним додатком починається зі створення нового проєкту або імпорту зображення з галереї пристрою. Користувач має можливість обрати зображення для подальшої пікселізації та редагування.

Приклад процесу вибору зображення наведено на рисунку 3.1.

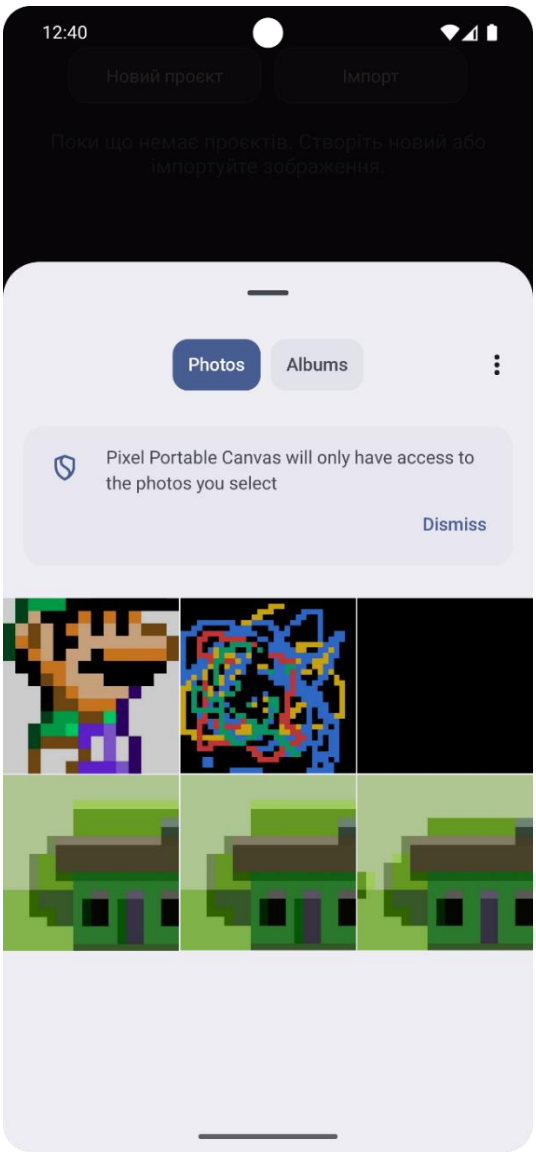


Рисунок 3.1. Вибір зображення з галереї

Після імпорту зображення користувачу надається можливість задати параметри обробки, зокрема розміри масштабування та палітру кольорів. Це реалізовано за допомогою відповідного інтерфейсу налаштувань, що дозволяє адаптувати зображення до вимог піксель-арту (рис. 3.2).

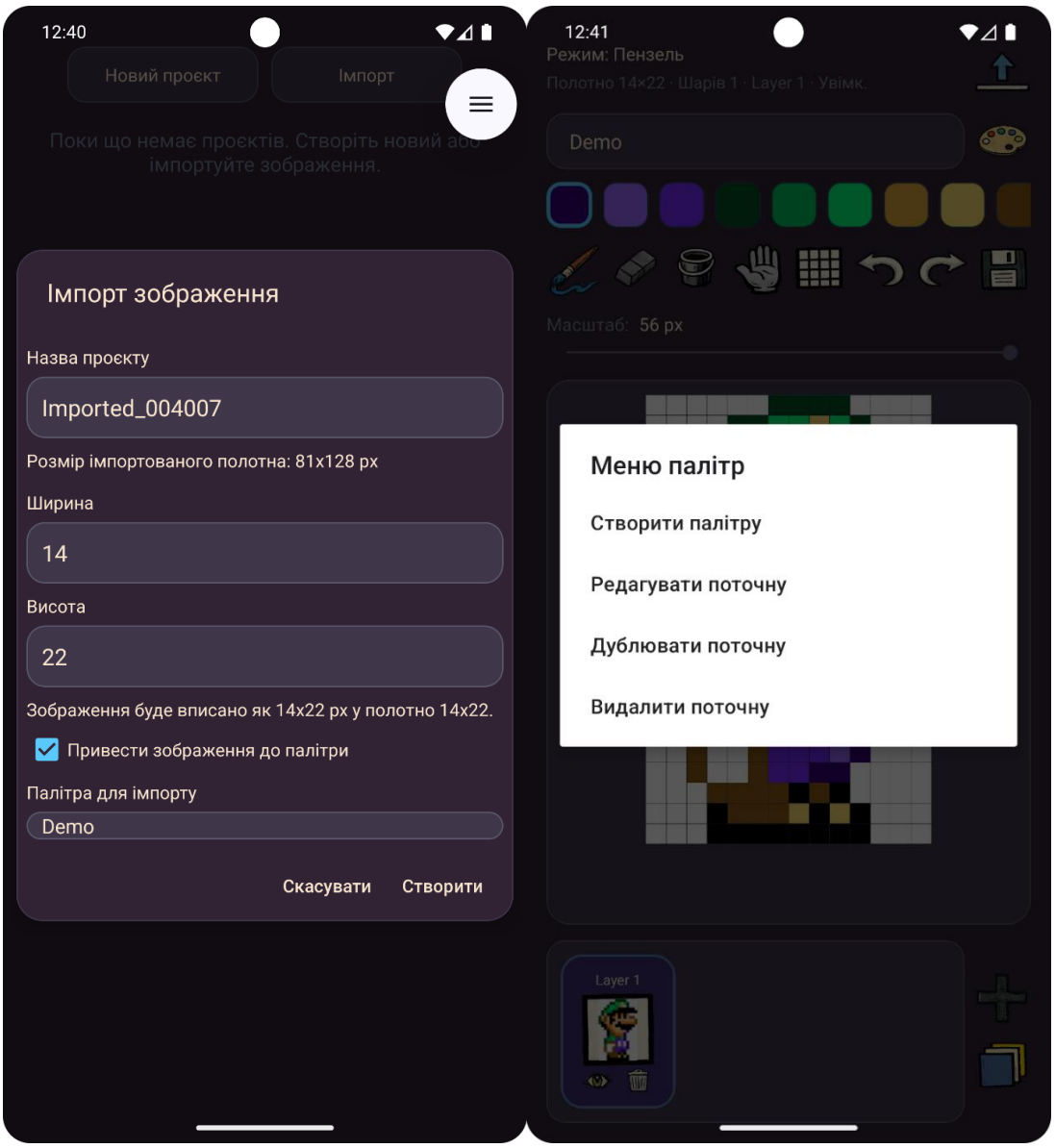


Рисунок 3.2. Меню імпорту та налаштування палітри

Для забезпечення зручності редагування було реалізовано відображення сітки, яка дозволяє візуально розділити зображення на окремі пікселі. Це особливо важливо при роботі з піксель-артом, оскільки дозволяє точно контролювати внесення змін (рис. 3.3).

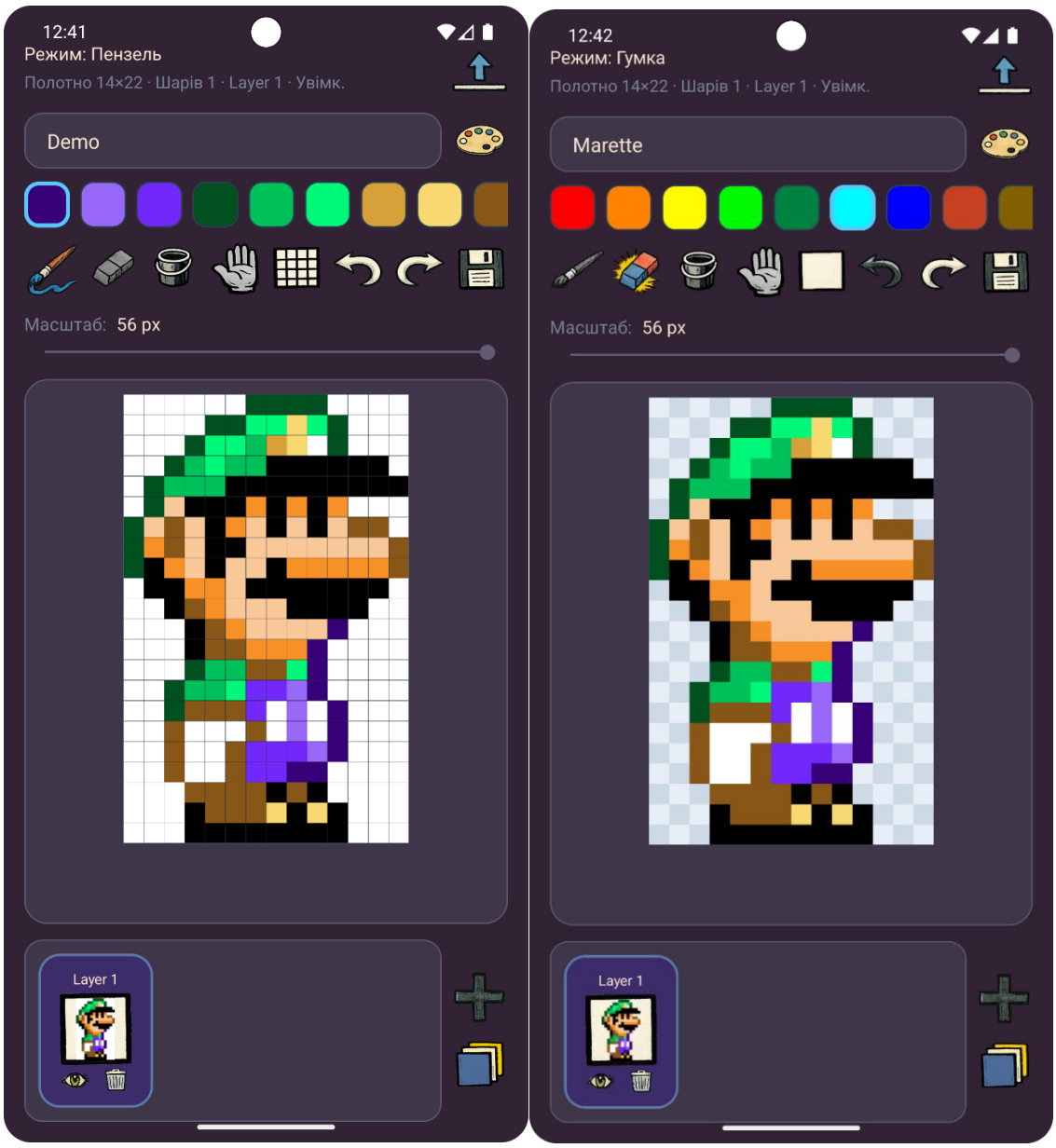


Рисунок 3.3. Відображення сітки та робота інструментів редагування

З метою розширення функціональних можливостей додатку було реалізовано систему шарів. Користувач може створювати нові шари, керувати їх видимістю та редагувати окремо один від одного. Такий підхід дозволяє експериментувати з графікою без ризику пошкодження основного зображення (рис. 3.4).

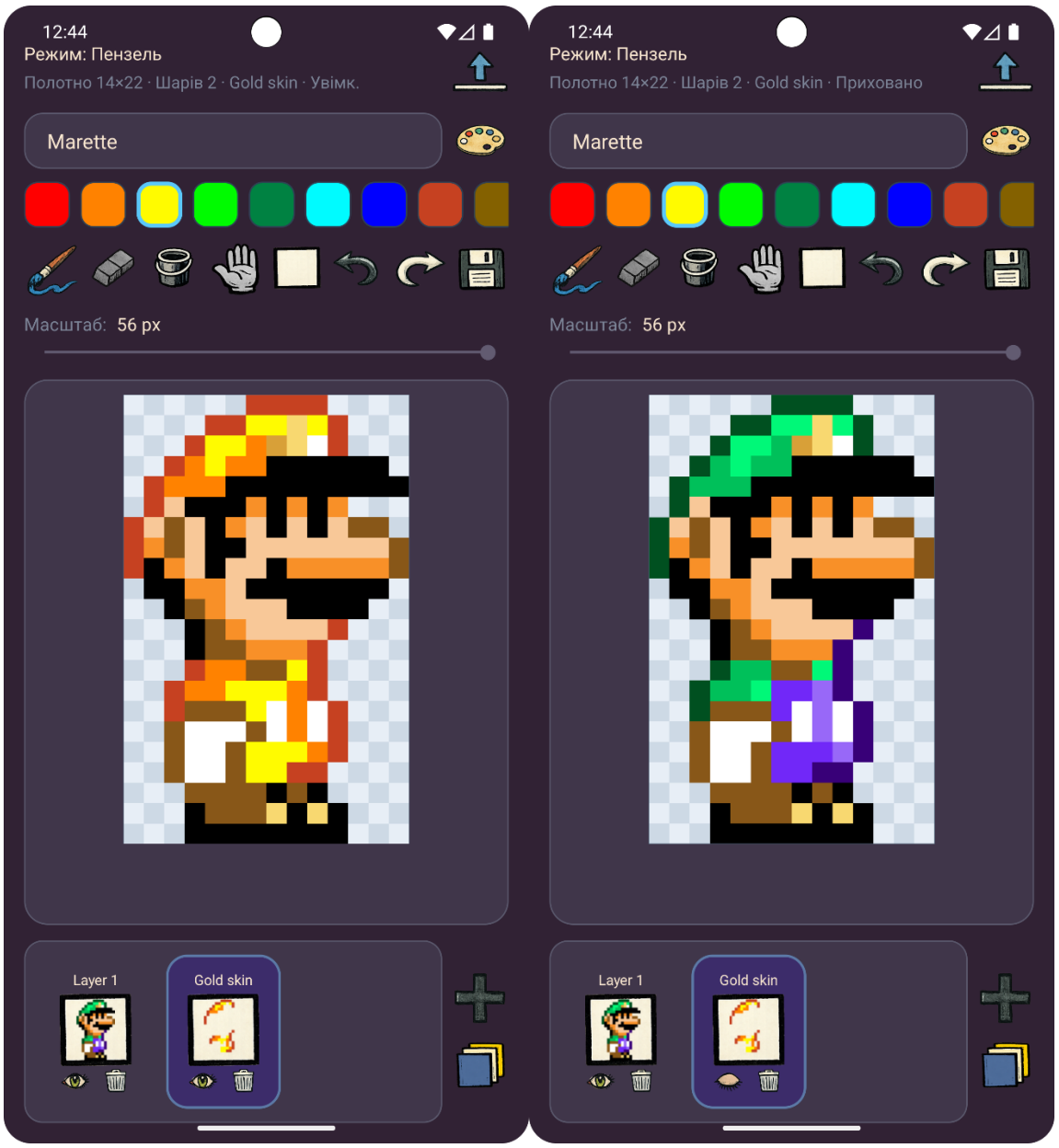


Рисунок 3.4. Робота з шарами

Для редагування зображення реалізовано базові інструменти, такі як заливка, гумка та можливість скасування дій (undo). Це забезпечує зручність роботи та підвищує ефективність редагування (рис. 3.5).

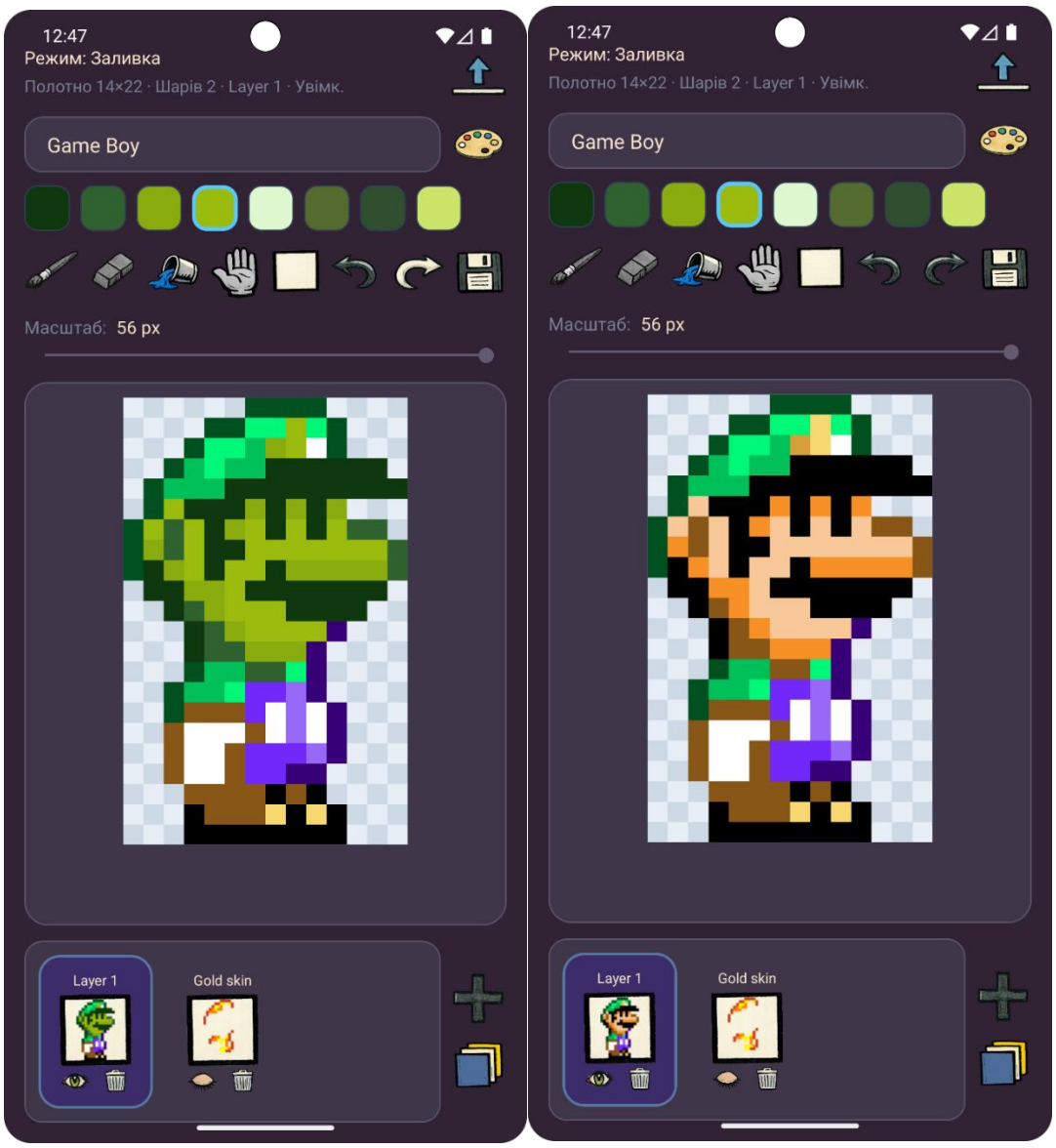


Рисунок 3.5. Використання інструментів редагування

Окрему увагу було приділено управлінню проєктами. Користувач має можливість перейменовувати, дублювати та видаляти проєкти, що значно спрощує організацію роботи (рис. 3.6).

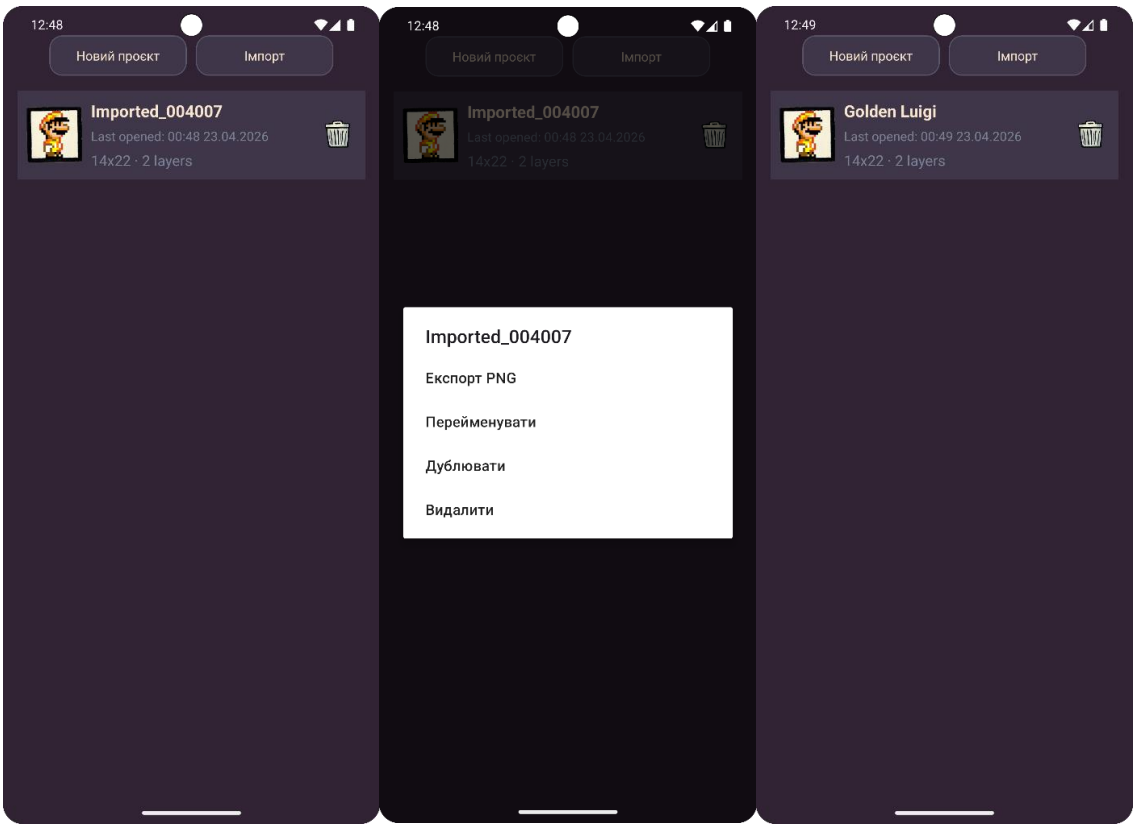


Рисунок 3.6. Управління проєктами

Завершальним етапом роботи є експорт зображення. У додатку реалізовано збереження результату у форматі PNG без втрат якості, що відповідає вимогам до піксель-арту. Під час експорту користувач може налаштувати основні параметри зображення, зокрема масштаб, колір фона та колір сітки готового результату. Реалізація експорту забезпечує коректне збереження структури окремих пікселів без появи розмиття та спотворення контурів. Процес експорту представлений на рисунку 3.7.

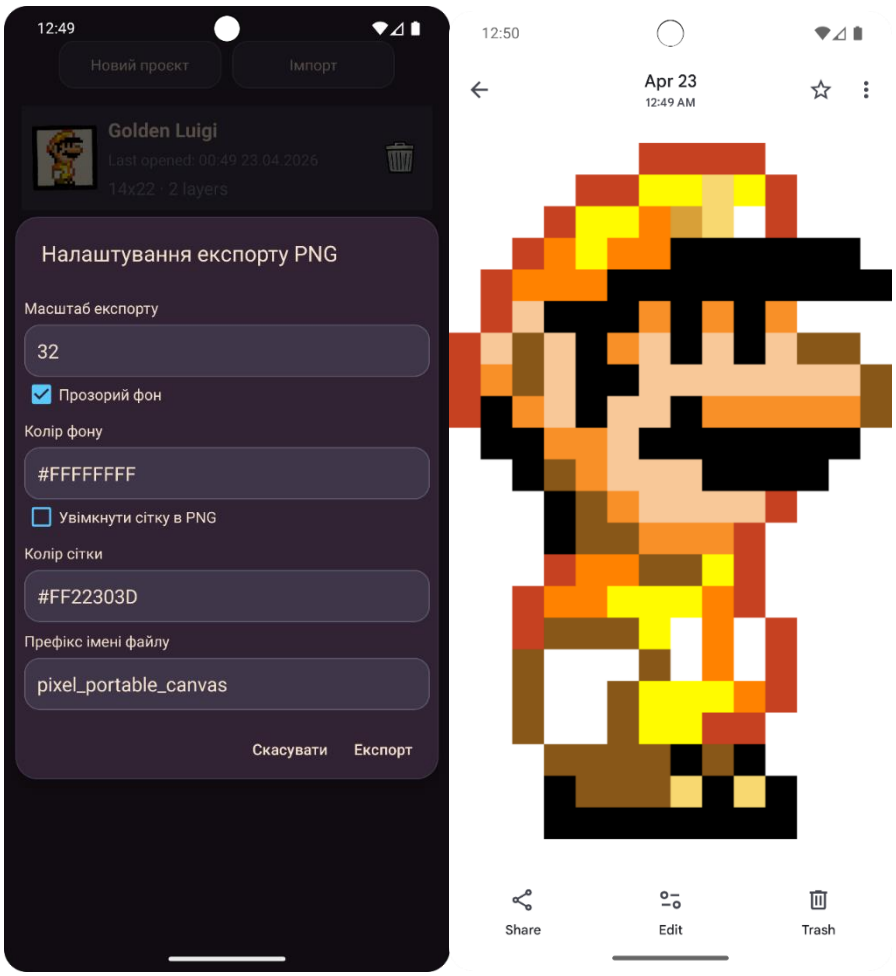


Рисунок 3.7. Експорт зображення

Крім того, при розробці було враховано рекомендації щодо оптимізації продуктивності мобільних додатків, зокрема ефективне використання пам’яті та мінімізація обчислювальних витрат [16]. Це дозволяє забезпечити стабільну роботу додатку навіть на пристроях із обмеженими ресурсами.

Таким чином, обрані технології та інструменти реалізації дозволяють забезпечити необхідну функціональність додатку, високу якість відображення піксель-арту та ефективність використання ресурсів мобільних пристроїв.

3.2 Програмна реалізація основних функцій мобільного додатку

У даному підрозділі наведено ключові фрагменти програмного коду, що реалізують основні функції мобільного додатку для візуалізації піксель-арту. Згідно з методичними рекомендаціями, у роботі наведено лише найбільш важливі частини програмної реалізації.

Лістинг 3.1 Імпорт зображення

Імпорт зображення реалізовано за допомогою стандартного механізму вибору файлів у системі Android.

```
    private final ActivityResultLauncher<String> importImageLauncher =
            registerForActivityResult(new
ActivityResultContracts.GetContent(), uri -> {
                if (uri != null) {
                    showImportDialog(uri);
                }
            });
```

Даний фрагмент коду забезпечує отримання зображення з галереї пристрою. Після вибору файлу викликається метод showImportDialog(), який відповідає за подальшу обробку та створення нового проєкту.

Лістинг 3.2 Масштабування зображення

Для забезпечення відповідності вимогам піксель-арту застосовується масштабування без згладжування.

```
    Bitmap.createScaledBitmap(decoded, scaledWidth, scaledHeight, false);
```

Параметр false вимикає згладжування, що дозволяє зберегти чіткість пікселів. Це є критично важливим для коректного відображення піксель-арту.

Лістинг 3.3 Приведення до палітри

Для обмеження кількості кольорів використовується алгоритм вибору найближчого кольору з палітри.

```
    int color = fittedBitmap.getPixel(x, y);
    if (palette != null) {
        color = quantizeToPalette(color, palette);
    }
    baseLayer.setPixel(x + offsetX, y + offsetY, color);
```

Сам алгоритм пошуку найближчого кольору базується на порівнянні RGB-компонент пікселя з кожним кольором палітри. Для цього обчислюється відстань між кольорами, після чого вибирається варіант із найменшим значенням.

```
    for (int candidate : palette.getColors()) {
        int dr = red - Color.red(candidate);
        int dg = green - Color.green(candidate);
        int db = blue - Color.blue(candidate);
        long distance = (long) dr * dr + (long) dg * dg + (long) db * db;
```

```
        if (distance < bestDistance) {
            bestDistance = distance;
            nearestColor = candidate;
        }
    }
```

Отже, приведення до палітри реалізовано під час імпорту зображення: кожен піксель перетворюється на найближчий доступний колір із вибраної палітри, що дозволяє отримати більш характерний для піксель-арту результат.

Лістинг 3.4 Відображення полотна

Відображення піксельного полотна реалізовано у методі onDraw().

```
int color = project.compositeColorAt(x, y);
if (color != Color.TRANSPARENT) {
    fillPaint.setColor(color);
    canvas.drawRect(left, top, right, bottom, fillPaint);
}
```

Цей код відповідає за відображення кожного окремого пікселя на екрані. Важливо, що згладжування відключене, що дозволяє зберегти характерну структуру зображення.

Лістинг 3.5 Інструмент малювання

Для безперервного малювання використовується алгоритм побудови лінії.

```
while (true) {
      applyPixel(x0, y0, color);
      if (x0 == x1 && y0 == y1) {
            break;
      }
      int e2 = 2 * err;
      if (e2 >= dy) {
            err += dy;
            x0 += sx;
      }
```

Цей алгоритм забезпечує коректне заповнення всіх проміжних пікселів між точками дотику, що запобігає появі розривів при малюванні.

Лістинг 3.6 Інструмент заливки

Заливка реалізована за допомогою алгоритму flood fill.

```
while (head < tail) {
    int currentX = queueX[head];
```

```
            int currentY = queueY[head];
            head++;

            if (activeLayer.getPixel(currentX, currentY) != targetColor)
continue;

            activeLayer.setPixel(currentX, currentY, replacementColor);
        }
```

Цей підхід дозволяє ефективно заповнювати області без використання рекурсії, що є важливим для стабільної роботи на мобільних пристроях.

Лістинг 3.7 Робота з шарами

Додавання та керування шарами реалізовано наступним чином:

```
public void addLayerAboveActive(String name) {
    int insertIndex = Math.min(activeLayerIndex + 1, layers.size());
    layers.add(insertIndex, new LayerData(name, width, height));
    activeLayerIndex = insertIndex;
}
```

Цей фрагмент демонструє динамічне створення нового шару та зміну активного шару.

Лістинг 3.8 Збереження проєктів

Збереження виконується у внутрішню пам’ять додатку.

```
ObjectOutputStream outputStream =
        new ObjectOutputStream(new FileOutputStream(temp));
outputStream.writeObject(project);
outputStream.flush();
```

Застосування серіалізації дозволяє зберігати повний стан проєкту, включаючи шари, палітри та налаштування.

Лістинг 3.9 Експорт зображення

Формування фінального зображення виконується шляхом масштабування кожного пікселя.

```
int scale = Math.max(1, settings.scalePerPixel);
int width = project.getWidth() * scale;
int height = project.getHeight() * scale;
Bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
```

Такий підхід забезпечує експорт зображення без втрати якості та збереження структури пікселів.

3.3 Аналіз результатів роботи програмного забезпечення

У результаті виконання дипломної роботи було реалізовано мобільний додаток для візуалізації та редагування піксель-арту на платформі Android. Програмне забезпечення забезпечує імпорт графічних зображень, їх пікселізацію, редагування, масштабування, роботу з шарами та палітрами кольорів, а також експорт готових результатів у форматі PNG.

Під час реалізації функцій масштабування було враховано особливості роботи з піксельною графікою. Використання методів, орієнтованих на збереження структури пікселів, дозволило уникнути розмиття контурів та появи небажаних артефактів [19; 20]. Аналогічні підходи до обробки pixel art-зображень розглядаються у сучасних дослідженнях масштабування та пікселізації графіки [24; 29].

Для реалізації роботи з графічними даними було використано механізми Canvas та Bitmap, які забезпечують ефективне відображення растрової графіки у мобільному середовищі [30]. Використання Canvas дозволило реалізувати малювання окремих пікселів, масштабування полотна та оновлення графіки у реальному часі без застосування згладжування [14].

Реалізований механізм роботи з палітрами кольорів забезпечує обмеження кількості кольорів та підтримку стилістики піксель-арту. Підходи до color quantization та генерації палітр, використані при реалізації програмного забезпечення, відповідають сучасним дослідженням у сфері цифрової обробки зображень [26; 27; 28]. Крім того, використання фіксованих палітр дозволяє досягти більш однорідного візуального стилю готових зображень [25].

Важливим елементом програмного забезпечення є підтримка шарів, що дозволяє незалежно редагувати окремі частини зображення. Такий підхід спрощує створення складних pixel art-композицій та відповідає сучасним принципам побудови графічних редакторів [18].

Під час реалізації програмного забезпечення також було враховано рекомендації щодо оптимізації графічних мобільних додатків [17]. Зокрема,

було реалізовано оптимізацію роботи з пам’яттю та швидке оновлення інтерфейсу користувача. Це дозволило забезпечити стабільну роботу додатку навіть при обробці великих растрових зображень.

У процесі дослідження також було проаналізовано сучасні підходи до використання нейронних мереж та методів суперрезолюції для обробки графіки [21; 22; 23]. Проте такі методи орієнтовані переважно на фотографічні зображення та потребують значних обчислювальних ресурсів, що обмежує доцільність їх використання у мобільних редакторах піксель-арту.

Під час тестування програмного забезпечення було перевірено коректність роботи основних функцій мобільного додатку, зокрема імпорту графічних зображень, масштабування, редагування пікселів, роботи з шарами, палітрами кольорів та експорту результатів. Тестування проводилося на мобільних пристроях з різними параметрами екрана та роздільною здатністю. У процесі перевірки оцінювалася стабільність роботи додатку, швидкість оновлення інтерфейсу та коректність відображення pixel art-графіки при різних рівнях масштабування. Отримані результати показали коректну роботу реалізованих функцій та відсутність критичних помилок під час роботи додатку.

Проведене тестування підтвердило, що розроблений мобільний додаток коректно виконує основні функції імпорту, редагування, масштабування та експорту графічних зображень. Отримані результати свідчать про доцільність використання обраних методів та технологій для реалізації мобільних засобів роботи з піксельною графікою.

Висновки до розділу 3

У третьому розділі виконано практичну реалізацію мобільного додатку для візуалізації та редагування піксель-арту на платформі Android. Реалізовано основні функції програмного забезпечення, зокрема імпорт зображень, пікселізацію, масштабування, роботу з палітрами кольорів і шарами, а також експорт результатів у форматі PNG.

Проведене тестування показало коректну роботу основних функціональних модулів додатку. Під час перевірки було підтверджено правильність роботи механізмів імпорту та експорту зображень, інструментів редагування, системи шарів і палітр кольорів. Також встановлено, що використання Canvas та Bitmap забезпечує якісне відображення піксельної графіки без розмиття та спотворення контурів.

Отримані результати свідчать про можливість використання розробленого програмного забезпечення як інструменту для створення та редагування піксель-арту на мобільних пристроях. Реалізовані архітектурні та програмні рішення забезпечують подальше розширення функціональності додатку та його адаптацію до нових вимог користувачів.

ВИСНОВКИ

1. У ході виконання дипломної роботи було проаналізовано сучасні підходи до візуалізації піксель-арту, зокрема методи масштабування растрових зображень та способи збереження чіткості піксельної структури. Проведений аналіз показав, що для коректного відображення pixel art-графіки найбільш доцільним є використання цілочисельного масштабування та відключення згладжування.

2. Досліджено особливості піксель-арту, а також розглянуто основні формати графічних зображень, що використовуються у мобільних додатках. Визначено особливості представлення піксельної графіки та вимоги до її обробки у програмному середовищі.

3. Проаналізовано методи масштабування та візуалізації графіки. Розглянуто класичні алгоритми інтерполяції та встановлено, що їх використання для піксель-арту може призводити до розмиття контурів і втрати характерної стилістики зображення. На основі проведеного аналізу було обґрунтовано вибір підходів, орієнтованих на збереження структури окремих пікселів.

4. Проаналізовано особливості відображення графіки на мобільних пристроях та досліджено можливості платформи Android для роботи з растровими зображеннями. Для реалізації графічної підсистеми було використано механізми Canvas і Bitmap, що забезпечують ефективну роботу з двовимірною графікою та коректне відображення pixel art-зображень.

5. Було розроблено мобільний додаток для візуалізації та редагування піксель-арту на платформі Android, який підтримує імпорт графічних зображень, пікселізацію, роботу з палітрами кольорів, шарами, масштабування полотна та експорт готових результатів у форматі PNG. Проведений аналіз результатів роботи програмного забезпечення показав коректність реалізації основних функцій та можливість подальшого розширення функціональності додатку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pixel Art: Complete Tutorial for Beginners and Professionals. URL: https://generalistprogrammer.com/tutorials/pixel-art-complete-tutorial-beginner-to-pro?utm_source (дата звернення: 08.04.2026).

2. The Pixel Art as Computer Graphics Artistic Expression in Digital Games. URL: https://www.researchgate.net/publication/384557883_The_Pixel_Art_as_Computer_Graphics_Artistic_Expression_in_Digital_Games (дата звернення: 08.04.2026).

3. Research on the Application of Pixel Art in Game Character Design. URL: https://www.researchgate.net/publication/362831872_Research_on_the_Application_of_Pixel_Art_in_Game_Character_Design (дата звернення: 08.04.2026).

4. Image Scaling Techniques and Their Impact on Pixel-Based Graphics. URL: https://arxiv.org/pdf/2007.15933 (дата звернення: 08.04.2026).

5. Advances in Image Processing and Pixel-Based Rendering Methods. URL: https://arxiv.org/pdf/2212.09692 (дата звернення: 08.04.2026).

6. Efficient Rendering Techniques for Mobile Graphics Systems. URL: https://arxiv.org/pdf/2203.12130 (дата звернення: 08.04.2026).

7. ACM Digital Library. Image Processing and Graphics Rendering Methods. DOI: 10.1145/2010324.1964994 (дата звернення: 08.04.2026).

8. Heckbert P. Fundamentals of Texture Mapping and Image Warping. URL: https://www.cs.cmu.edu/~ph/texfund/texfund.pdf (дата звернення: 08.04.2026).

9. Gonzalez R. C., Woods R. E. Digital Image Processing. - 4th ed. - Pearson, 2018. - 1168 p.

10. Szeliski R. Computer Vision: Algorithms and Applications. URL: https://szeliski.org/Book/ (дата звернення: 08.04.2026).

11. Zhang K., Zuo W., Chen Y., Meng D., Zhang L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. URL: https://arxiv.org/pdf/1608.03981 (дата звернення: 08.04.2026).

12. Android Open Source Project. Android Graphics Pipeline. URL: https://source.android.com/docs/core/graphics (дата звернення: 08.04.2026).

13. Mitchell D. P., Netravali A. N. Reconstruction Filters in Computer Graphics. *Computer Graphics (SIGGRAPH)*. 1988. Vol. 22, No. 4. P. 221-228. - DOI: 10.1145/378456.378514.

14. Android Developers. Canvas and Drawables. URL: https://developer.android.com/guide/topics/graphics/2d-graphics (дата звернення: 08.04.2026).

15. Android Developers. RenderScript. GPU Rendering. URL: https://developer.android.com/guide/topics/renderscript (дата звернення: 08.04.2026).

16. Google. Android Performance Best Practices. URL: https://developer.android.com/topic/performance (дата звернення: 08.04.2026).

17. McGuire M. The Graphics Codex. URL: https://graphicscodex.com/ (дата звернення: 08.04.2026).

18. OpenCV Documentation. URL: https://docs.opencv.org/ (дата звернення: 08.04.2026).

19. Kopf J., Lischinski D. Depixelizing Pixel Art. ACM Transactions onGraphics. 2011. Vol. 30, № 4. DOI: 10.1145/2010324.1964994.

20. Wu Z., Zhang Y., Thomas G. Aliasing-Aware and Cell-Controllable Pixelization. Cardiff University. 2022. URL: https://orca.cardiff.ac.uk/id/eprint/152816/14/pixelart.pdf (дата звернення: 08.04.2026).

21. Seo J. W., Kim H., Lee J. Structure-Aware Pixel Art Scaling via Block Size Detection. *Applied Sciences*. 2026. Vol. 16, № 5. URL: https://www.mdpi.com/2076-3417/16/5/2314 (дата звернення: 08.04.2026).

22. Trusov A., Limonova E. The Analysis of Projective Transformation Algorithms for Image Recognition on Mobile Devices. arXiv. 2019. URL: https://arxiv.org/abs/1912.01401 (дата звернення: 08.04.2026).

23. Zhao H., Shi J., Qi X. Efficient Image Super-Resolution Using Pixel Attention. arXiv. 2020. URL: https://arxiv.org/abs/2010.01073 (дата звернення: 08.04.2026).

24. Furuta R., Inoue N., Yamasaki T. PixelRL: Fully Convolutional Network with Reinforcement Learning for Image Processing. arXiv. 2019. URL: https://arxiv.org/abs/1912.07190 (дата звернення: 08.04.2026).

25. Bansal A., Sheikh Y., Ramanan D. PixelNN: Example-based Image Synthesis. arXiv. 2017. URL: https://arxiv.org/abs/1708.05349 (дата звернення: 08.04.2026).

26. Frackiewicz M. Efficient Color Quantization Using Superpixels. *Sensors*. 2022. Vol. 22, № 16. URL:https://pmc.ncbi.nlm.nih.gov/articles/PMC9416436/ (дата звернення: 08.04.2026).

27. Huang S. C. An Efficient Palette Generation Method for Color Image Quantization. *Applied Sciences.* 2021. Vol. 11, № 3. URL: https://www.mdpi.com/2076-3417/11/3/1043 (дата звернення: 08.04.2026).

28. Pérez-Delgado M. L. A Comparative Study of Color Quantization Methods Using Image Quality Assessment Indices. *Multimedia Tools and Applications.* 2024. URL:https://link.springer.com/article/10.1007/s00530-023-01206-7 (дата звернення: 08.04.2026).

29. Gerstner P., DeCarlo D., Alexa M., Finkelstein A. Pixelated Image Abstraction. *ACM Transactions on Graphics.* 2012. Vol. 31, № 4. URL: https://gfx.cs.princeton.edu/pubs/Gerstner_2012_PIA/index.php (дата звернення: 08.04.2026).

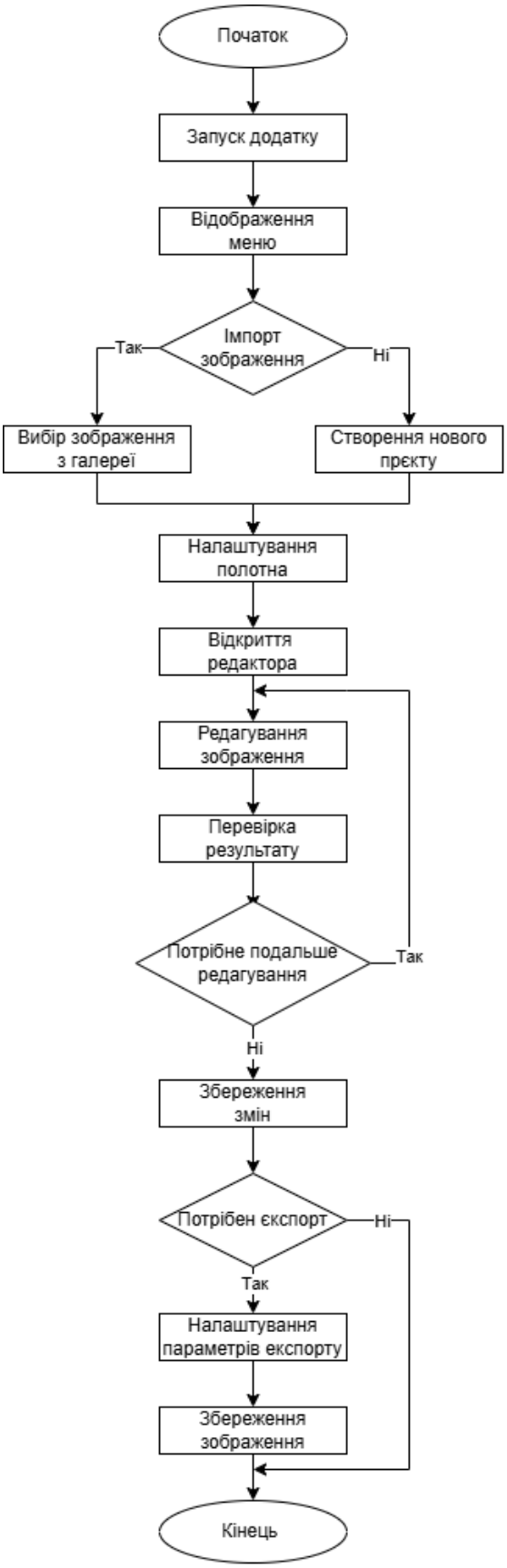
30. Bondarchuk, A., Dibrivniy, O., Grebenyk, V., & Onyshchenko, V. (2021, October). Motion Vector Search Algorithm for Motion Compensation in Video Encoding. In 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T) (pp. 345-348). IEEE.

31. Android Developers. Canvas API Reference. URL: https://developer.android.com/reference/android/graphics/Canvas (дата звернення: 08.04.2026).

32. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

Додаток А

Рисунок А.1



Алгоритм роботи програмного забезпечення.

На рисунку А.1 представлено алгоритм роботи програмного забезпечення мобільного додатку для візуалізації піксель-арту. Робота системи починається із запуску додатку та відображення головного меню. Далі користувач може створити новий проєкт або імпортувати зображення з галереї пристрою.

Після налаштування параметрів полотна відкривається редактор, у якому здійснюється редагування зображення. Користувач може виконувати повторне редагування до отримання необхідного результату. Після завершення роботи зміни зберігаються, а готове зображення за потреби експортується у формат PNG.