

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 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗВАЖАЛЬНОГО
МОБІЛЬНОГО ЗАСТОСУНКУ
SOFTWARE FOR AN ENTERTAINMENT
MOBILE APPLICATION

Виконав: студент 4 курсу, групи КНТ-120сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ШЕМЕТЬКО К.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШЕМЕТЬКА Кирила Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення розважального мобільного застосунку. Software for an Entertainment Mobile Application

керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від "04" квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження технологій розробки ігрових застосунків. 3. Проєктування ПЗ. 4. Конструювання проєкту. 5. Керівництво програміста. 5. Керівництво користувача.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент

_____ Кирило ШЕМЕТЬКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник роботи

_____ Анжеліка ПАРХОМЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 114 с., 6 табл., 39 рис., 2 дод., 59 джерел.

ГРА, UNITY3D, МОБІЛЬНИЙ ЗАСТОСУНОК, ОРГАНІЗАЦІЯ ДОЗВІЛЛЯ, ANDROID.

Об'єкт дослідження – процес створення мобільного застосунку.

Предмет дослідження – методи та засоби розробки розважального мобільного застосунку.

Мета роботи – розробка програмного забезпечення розважального мобільного застосунку для організації дозвілля дітей та дорослих.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, технологія та ігровий двигун Unity, мови програмування C#, персональний комп'ютер з процесором AMD RYZEN 5 під управлінням операційної системи Microsoft Windows 11, або ноутбук з процесором Intel Core i7 під управлінням операційної системи macOS Monterey.

Результати. Створено програмний Android застосунок, який призначено для використання дітьми та дорослими з метою розваг та цікавого проведення вільного часу.

Висновки. Практична цінність розробки полягає в тому, що створений розважальний мобільний застосунок має привабливий інтерфейс, гарантує низьку витрату заряду смартфона та може бути опублікований до Google Play Market.

Галузь використання – ігри для смартфонів.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 114 pages, 6 tables, 39 figures, 2 appendixes, 59 sources.

GAME, UNITY3D, MOBILE APPLICATION, ORGANIZING OF LEISURE TIME, ANDROID.

Object of study is the process of creating a mobile application.

Subject of study are methods and tools of developing entertaining mobile applications.

The goal of the work is to develop software for an entertaining mobile application for organizing leisure activities for children and adults.

Materials, methods, and tools: structural and object-oriented programming, Unity technology and game engine, C# programming languages, a personal computer with an AMD RYZEN 5 processor running the Microsoft Windows 11 operating system, or a laptop with an Intel Core processor i7 running the macOS Monetary operating system.

Results. The Android application has been created that is intended for use by children and adults for entertainment and interesting leisure activities.

Conclusions. The practical value of the development lies in the fact that the created entertainment mobile application has an attractive interface, guarantees low smartphone battery consumption and can be published on the Google Play Market.

The field of application is games for smartphones.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Огляд існуючих типів ігор	10
1.2 Геймінг на основі віртуальної реальності	16
1.3 Портативний геймінг	19
1.4 Мобільні ігри	25
1.5 Висновки до розділу 1	29
2 Дослідження технологій розробки ігрових застосунків	30
2.1 Технології розробки розважальних застосунків	30
2.2 Захоплення рухів в іграх.....	333
2.3 Програмне забезпечення для візуалізації	36
2.4 Постановка завдань дипломної роботи.....	44
3 Проєктування програмного забезпечення	46
3.1 Аналіз вимог	46
3.2 Методика розробки мобільної гри	48
3.3 Архітектура системи	49
3.4 Огляд та вибір інструментарію розробки	50
3.5 Висновки до розділу 3	60
4 Конструювання проєкту	62
4.1 Схема алгоритму функціонування застосунку	62
4.2 Створення компонентів	63
4.3 Компоновка об'єктів на сцені	68
4.4 Розробка скриптів і нанесення їх на об'єкти	71
4.5 Програмна реалізація	76
4.6 Висновки до розділу 4	78
5 Керівництво програміста.....	79
5.1 Розгортання проєкту	79

5.2 Структура проєкта.....	79
5.3 Додавання персонажа	80
5.4 Додавання платформи.....	82
5.5 Збірка проєкту	82
5.6 Висновки до розділу 5	83
6 Керівництво користувача	84
6.1 Призначення мобільного застосунку	84
6.2 Запуск гри	84
6.3 Висновки до розділу 6	88
Висновки	89
Прелік джерел посилань	90
Додаток А Текст програми.....	96
Додаток Б Слайди презентації	109

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AJAX	– Asynchronous JavaScript And XML;
APK	– Android Package;
AR	– Augmented Reality;
CGI	– Common Gateway Interface;
CSS	– Cascading Style Sheets;
HTML	– HyperText Markup Language;
IDE	– Integrated Development Environment;
MVC	– Model View Controller;
PS	– PlayStation;
PSP	– PlayStation Portable;
RPG	– Role-Playing Games;
SDK	– Software Development Kit;
VR	– Virtual Reality;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер.

ВСТУП

На сьогоднішній день створено багато різноманітних ігрових застосунків для мобільних пристроїв, десктопних застосувань, гральних консолей та обладнання віртуальної реальності: 3D шутери, платформери, ігри з віртуальною та доповненою реальністю тощо. Особливо актуальною є розробка мобільних і кросплатформних ігор, не прив'язаних до конкретного типу пристроїв. Мобільні відеоігри зараз на першому місці за кількістю користувачів, тому що майже кожна людина має при собі смартфон, і все що потрібно – лише обрати та завантажити через Steam, Play Market або Apple Store певний ігровий застосунок.

Дослідження показали, що для створення ігрових застосунків розроблено різні ігрові движки та бібліотеки. Основними двигунами є Unity від Unity Technologies, CryEngine від Crytek і Unreal Engine від Epic Games.

Сьогодні Unity є одним із найпопулярніших ігрових движків і основним інструментом для створення ігор для індивідуальних розробників не лише завдяки доступній бізнес-моделі, а й завдяки надійності редактора проєктів, щорічним вдосконаленням і легкості застосування. Редактор Unity містить багато інструментів для прискорення та повторення циклів розробки, включаючи ігровий режим для швидкої перевірки продуктивності в реальному часі. Тому, саме цей інструмент було обрано для створення мобільного розважального застосунку.

Розроблюваний мобільний застосунок відноситься до жанра нескінченний ранер. Популярність таких застосунків пояснюється простим геймплеєм, який добре працює на сенсорних пристроях. Цей жанр є актуальним, про що свідчить зростаюча популярність ранера Subway Surfers, кількість завантажень якого у Google Play і Apple Store перевищує 16,51 мільйонів. Тому, тема роботи є актуальною.

Мета роботи – розробка програмного забезпечення розважального мобільного застосунку для організації дозвілля дітей та дорослих.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих типів ігор

На сьогоднішній день ігри є одним з найпопулярнішими видами розваг як для молодого так і дорослого покоління людей. За рахунок різноманітності жанрів кожен знаходить для себе цікавий продукт, який можна купити на будь-якому ліцензійному цифровому майданчику [1].

Для кожної платформи існують свої, наприклад, якщо брати Sony з її легендарною консоллю Playstation ми має PS Store де власник приставки може купити будь-яку гру за певну ціну в його регіоні, так само і Microsoft з Xbox, Nintendo з Nintendo Switch тощо [1].

Існує безліч платформ для ігор, такі як ПК, консолі (Playstation і Xbox), портативні консолі (Nintendo і Steam Deck), і звичайно ж смартфони, більш детально зображено на рис. 1.1.

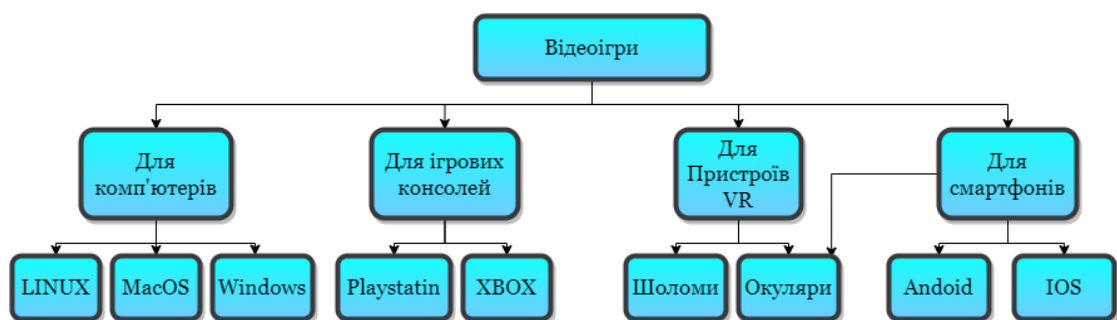


Рисунок 1.1 – Класифікація типів відеоігор

На багатьох з них адаптовані одні і ті ж ігри, але існує так звані ексклюзиви. Наприклад Sony для своїх платформ PS роблять ігри які в які не можна пограти на інших пристроях: Bloodborn, Gran Turismo і т.д. До недавніх пір було безліч і інших проєктів, які були справді унікальні, проте жага вигоди змушує Sony робити порти своїх ексклюзивів і на ПК, таким чином світ побачив легендарного Sony Spider-Man, серію ігор God Of War і

Last of Us, який вийшов найпроблемнішим портом із усіх, оскільки Sony доручила цю роботу дочірній компанії [1].

Типи та жанри ігор на прикладі найпопулярніших для геймерів і платформ таких як ПК або консолі: екшен ігри, ігри пригоди, RPG-ігри, шутери, стратегії, ігри виживання та гонки [1].

Відеоігри в жанрі екшн – це відеоігри, що визначаються динамічним геймплеєм з упором, як правило, на рух, бій, моторно-зорову координацію і час реакції. Як жанр, це дуже широкий термін, і зазвичай він використовується для визначення ігор, які можуть не відповідати конкретнішим жанрам, таким як головоломки, платформери, гонки, FPS або RPG, хоча шутери та відеоігри-платформери також можуть підпадати під це визначення жанру екшн. Нерідко елементи з інших типів ігор включені до відеоігор-бойовиків; в результаті цього багато хто використовує слово «екшн / пригода» для позначення ігор у жанрі екшн з підвищеною увагою до елементів пригод, таких як, наприклад, дослідження та/або збір. Багато інших основних жанрів дійсно включають рівень контролю, яким відомий жанр бойовиків; в той час як шутери від першої особи засновані на русі та стрільбі, вони класифікуються відповідно до унікальної перспективи гри. Так само платформні відеоігри, як правило, являють собою екшн-ігри, в яких велика увага приділяється головоломкам зі стрибками [2].

Пригодницькі відеоігри розповідають історію свого персонажа чи персонажів у буквальній пригоді. Пригодницька відеогра зазвичай заснована на сюжеті в тому сенсі, що найбільш відмінною рисою пригодницької відеоігри є те, що, можливо, за винятком міні-ігор, вся гра слідує єдиній єдиній темі і має мало або взагалі немає змін екрана/перспективи. на відміну від деяких рольових ігор). Це служить для того, щоб зосередити гравця на єдиній задачі його або її пригоди. Найпопулярніші ігри з цього жанру це Вільямс, Зельда, Скайрім [3].

Рольові відеоігри (RPG) – це жанр ігор, який зазвичай дає вам прямий контроль над персонажем або групою персонажів, здібностями, предметами,

навичками та іншими атрибутами. Рольова гра сумно відома історіями епічного масштабу та великої довжини.

Найпопулярніша гра жанру RPG – World of Warcraft, яка є відгалуженням основою всесвіту Warcraft [4].

Відеоігри шутери – це поджанр відеоігор у жанрі екшн, які часто перевіряють швидкість та час реакції гравця. Він включає безліч поджанрів, які мають спільну особливість зосередження уваги «на діях аватара з використанням будь-якої зброї».

Зазвичай ця зброя є пістолетом або будь-якою іншою зброєю дальньої дії». Найпоширенішим ресурсом у багатьох шутерах є боєприпаси. Найчастіше метою шутера є стрілянина по противникам та виконання місій без смерті персонажа гравця [5].

Стратегії це ігри, які змушують вас думати. Вони змушують тебе використовувати свій мозок, і це спричиняє біль. Існують різні типи стратегічних ігор, поджанри, такі як стратегія у реальному часі та покрокова стратегія. Стратегія – це елемент, який можна знайти в багатьох нестратегічних іграх, що стосується будь-якого моменту, коли вам потрібно подумати про тактику. Наприклад, в екшн-грі може бути певна стратегія, що робить атаку ефективнішою. Стратегічні відеоігри беруть свій початок у стратегічних настільних іграх, включаючи шахи, го, шашки та Wargaming, які також надихнули на створення рольових відеоігор [6].

Ігри на виживання – це поджанр екшн-відеоігор, дія яких відбувається у ворожому, інтенсивному середовищі відкритого світу, де гравці зазвичай починають з мінімальним спорядженням і повинні збирати ресурси, створювати інструменти, зброю та укриття, а також виживати якнайдовше. Багато ігор на виживання засновані на випадковому або процедурно згенерованому постійному оточенні; у нещодавно створені ігри часто можна грати онлайн, що дозволяє кільком гравцям взаємодіяти в єдиному постійному світі [7].

Ігри на виживання, як правило, відкриті, без поставленої мети і часто тісно пов'язані з жанром жахів на виживання, в якому гравець повинен вижити в надприродній обстановці, такий як зомбі-апокаліпсис [7].

Гоночні відеоігри відносяться до відеоігор, які зазвичай складаються з будь-якого транспортного засобу, що конкурує з іншими транспортними засобами на серії різних трас або рівнів, де мета полягає в тому, щоб дістатися до кінця раніше від інших. Гоночні відеоігри не завжди повинні містити транспортні засоби; у деяких іграх, таких як Sonic R, просто бігають персонажі [8].

Також є екшн гри змішані з гонками, такі як GTA, Watch Dogs і т.д.

У сьогоdnішній час існує безліч ігор різних жанрів, що пропонують гравцям різноманітні враження та можливості. Один з цих жанрів - ігри-симулятори, які пропонують унікальний досвід імітації реальних дій або процесів. У таких іграх гравці можуть відчувати себе в ролі рибалки, фермера, пілота, підприємця та багатьох інших професій або ситуацій.

Ігри-симулятори дозволяють гравцям погрузитися в віртуальний світ, де вони можуть виконувати реалістичні завдання, розвивати навички та досліджувати нові можливості. Наприклад, симулятор рибалки може навчити гравця правильно використовувати риболовні снасті, розуміти поведінку риби та вдосконалювати свої навички лову. У той же час, симулятор бізнесу може дати гравцям можливість керувати власним підприємством, приймати стратегічні рішення, вдосконалювати процеси та досягати успіху [9].

Ці ігри-симулятори не лише розважають, але й надають навчальну цінність, допомагаючи гравцям розширити свої знання та навички в певній сфері. Вони також дають можливість зануритися у фантастичні світи або відтворити реальні сценарії з життя. Ігри-симулятори є популярним жанром серед гравців, які цікавляться реалістичними досвідами та новими викликами. Приклади ігор представлені в таблиці 1.1.

Таблиця 1.1 – Приклади ігор

Приклад інтерфейсу	Назва гри
1	2
	Приклад ігри пригоди – Легендарний Шрек 2 [10]
	Приклад ігри пригоди – WOW [5211]
	Приклад ігри шутера – Valorant [4612]
	Приклад ігри стратегії – Warcraft 3 [5013]
	Приклад ігри виживання – DayZ [14]

Продовження таблиці 1.1

1	2
	Приклад ігри симулятора – Bread Simulator [4515]
	Пристрої для геймерів – ПК/PS/XBOX [1653]

В сучасному світі відеоігор, інді-ігри займають важливе місце в широкому спектрі ігрових жанрів. За останні роки інді-ігри зазнали значного зростання популярності, завдяки новаторським ідеям розробників. Назва "Незалежні ігри" або "Indie Games" виникла у зв'язку з їхньою незалежністю від спонсорів або видавців, що фінансують розробку ігор. Це, в свою чергу, часто спричиняє доступніші ціни для гравців, які зазвичай коливаються від 5 до 25 євро.

Отже, інді-ігри в основному розробляються з обмеженими бюджетами, що часто називається "малобюджетними іграми", і команди розробників часто складаються з невеликої кількості або навіть окремих осіб. Інді-ігри відрізняються від інших жанрів не лише своїми фінансовими особливостями, але й за наявністю нетрадиційного графічного стилю та інноваційних геймплейних елементів. Ці ігри виступають як платформа для реалізації свіжих та креативних ідей, набагато більше, ніж забезпеченням постійних інкрементальних змін для вже встановлених брендів.

Серед видатних представників інді-ігр можна згадати такі проєкти, як Minecraft, Terraria та Slender: The Eight Pages. Кожна з цих ігор відображає унікальний підхід до геймплею, зокрема, Minecraft став світовим феноменом, завдяки свободі та безмежним можливостям будівництва, а Terraria (рис.1.2) представляє підземний світ зі своїми власними викликами та пригодами [17].

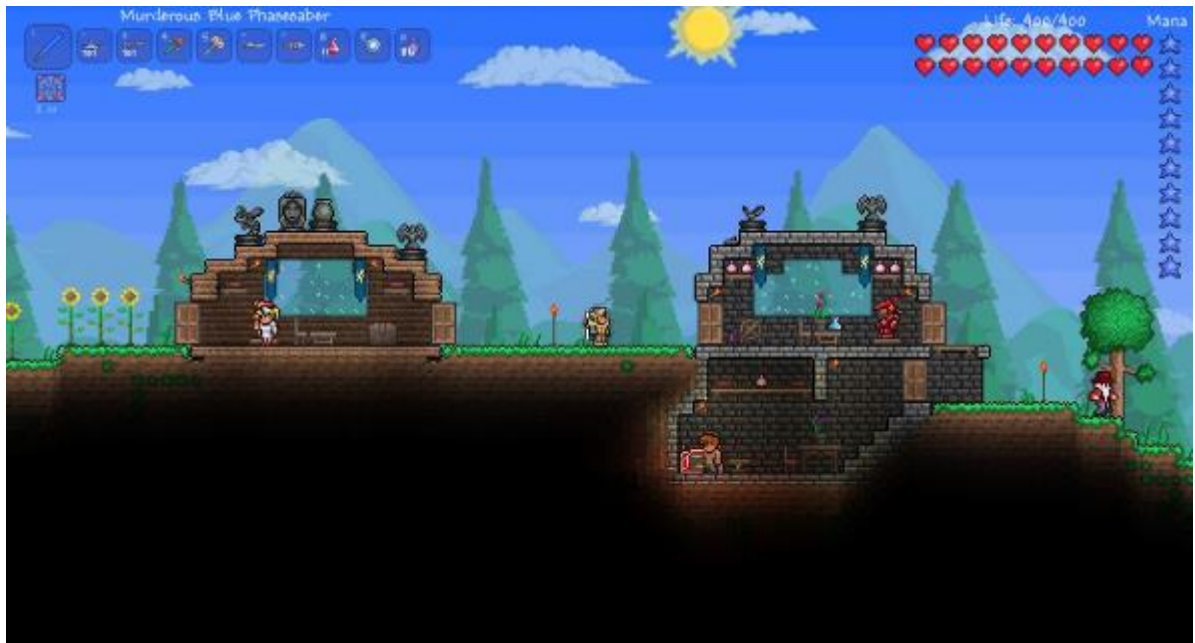


Рисунок 1.2 – Terraria – інді гра [18]

1.2 Геймінг на основі віртуальної реальності

Поняття геймінгу віртуальної реальності (Virtual Reality, VR) з'явилося з можливістю занурити гравця у так званий цифровий світ через спеціальний шолом та двох контролерів, які забезпечують гравцю управління. Ігри віртуальної реальності – це те, де людина може відчувати себе у тривимірному середовищі та взаємодіяти з цим середовищем під час гри.

Гарнітура віртуальної реальності (або VR-гарнітура) – це пристрій, що носить на голові, що забезпечує користувачеві віртуальну реальність [19]. Зображення VR – гарнітури представлено на рис. 1.3.



Рисунок 1.3 – VR шолом [19]

Одним із способів виявлення присутності людини у грі є біосенсор. Це невеликі датчики, які кріпляться до рукавички, костюма або навіть тіла і реєструють рухи, які здійснюють ця людина в тривимірному просторі. Ці рухи інтерпретуються комп'ютером і викликають різні реакції у цьому просторі. Наприклад: грає носить рукавичку передачі даних, до якої прикріплені датчики. Людина носить це як частина гри у ведення. Ці датчики фіксують рухи вашої руки в рамках цієї гри, наприклад, коли гравець повертає кермо у певному напрямку. Ці рухи передаються назад на комп'ютер, який потім аналізує дані та використовує їх для перетворення ваших дій у відповідні реакції на екрані. Це звучить трохи технічно, але це розумний спосіб виявити рухи гравця під час гри та використовувати їх, щоб впливати на те, що відбувається в цій грі. Це означає, що ви, гравець, стаєте важливою частиною гри. Це відомо як «імерсивний досвід» [2033].

На жаль, управління на сьогоднішній день не ідеальне, тому що воно дуже сильно впливає на вестибулярний апарат і гравець може відчувати сильний дискомфорт, запаморочення і нудоту при пересуванні. Для цього в такі ігри обов'язково пишуть попередження і створюють окремий тип пересування – телепорт по точках, що є більш безпечним для гравця. Більшість VR ігор це просто звичайні розваги та інтерактив, однак у 2020 році компанія Valve випускає найпроривніший сюжетний проєкт легендарної серії ігор Half Life – Half Life Alyx. Це повноцінний VR шутер із чудовим

сюжетом. Гра отримала божевільні оцінки як критиків так і фанатів, які майже 20 років чекали хоч якесь світло від цієї серії, але найголовніше, що ця гра дала стимул для інших розробників переходити до VR

Існують ще AR окуляри, які не відносяться до геймерських гаджетів і не варто їх порівнювати з окулярами/шоломом VR. Розумні окуляри доповненої реальності – це окуляри, що носяться з комп'ютерними можливостями, які надають додаткову інформацію, в ідеалі 3D-зображення та інформацію, таку як анімація і відео, до сцен реального світу користувача шляхом накладання згенерованої комп'ютером або цифрової інформації на реальний світ користувача.

На відміну від VR пристроїв, AR не мають додаткових контролерів для використання у грі, бо це просто пристрої для спрощення завдань у повсякденному житті. Зображення AR окулярів представлено на рисунку 1.4.



Рисунок 1.4 – AR [21]

Ще один шолом, який теж не варто прирівнювати до геймерських, був придуманий спочатку компанією Samsung для своїх пристроїв – Samsung Gear VR. Це була спроба компанії залучити сторонніх розробників створювати VR продукти для платформ Android на базі своїх телефонів. Пристрій працював так, що в спеціальний шолом вставляється смартфон від компанії Samsung і користувач "отримував" ефект VR. Однак лінзи не можуть передати повний ефект VR через сторонній дисплей і погано реалізовані контролери, ну і звичайно ж відсутність повноцінних VR ігор на смартфон. Як результат продукт був провальним і більше ніж на перегляд фільмів у VR шолом не

годився і був швидко знятий з продажу. Зображення Samsung Gear VR представлено на рис. 1.5.



Рисунок 1.5 – Samsung Gear VR [22]

1.3 Портативний геймінг

Портативні ігрові консолі менші за домашні ігрові консолі та містять консоль, екран, динаміки та елементи керування в одному блоці, що дозволяє людям носити їх і грати в них у будь-який час і в будь-якому місці.

Не дивлячись на свої розміри не варто порівнювати їх з мобільним гейміном, це абсолютно різні поняття. Ігрові портативні консолі існували з кінця 80-х років, хоча історія створення починається ще з кінця 70-х, але через одноманітність ігор (а їх було менше 5 штук) і незатребуваності на той момент консоль не набула популярності і мало хто про неї досі знає. Зображення першої портативної консолі представлено на рис. 1.6.



Рисунок 1.6 – Milton-Bradley-Microvision – перша портативна консоль [23]

Історія цих консолей почалася наприкінці 90-х і отримала неофіційну назву – консольні війни. Дві японські компанії, які мали вже на той момент свої легендарні консолі Nintendo і Sega, вирішили створити проривні на той момент перші портативні консолі.

За сьогоднішніми мірками Game Boy – потворна сіра цегла. Але цегла, яка була основою стільки дитячих років, тому його суворі кути, фіолетові кнопки і зелено-жовтий дисплей завжди здаватимуться чарівними. Крім того, він був міцним і поміщався в задню кишеню [24].

Якщо Game Boy був цеглою, то Game Gear був шлакоблоком, що майже перетворювало слово «портативний» на глузування. Штани, можливо, були мішкуватими на початку 1990-х, але навіть MC Hammer не пошкодував роботи, щоб сховати цього звіра розміром 209 x 111 x 37 мм у свої штани [24].

На відміну від Game Boy, система Sega була вирівняна по горизонталі, з хрестовиною зліва та кнопками праворуч, що дозволяло запускати ігри у ландшафтному режимі. Вона виглядала безперечно крутіше і технологічніше за свого конкурента від Nintendo, але тягати її з собою було зовсім не весело [24].

На папері Game Gear тут очевидний переможець. Бо похмурий монохроматичний Game Boy не міг конкурувати з повнокольоровою потужністю екрану системи Sega з роздільною здатністю 160x144 пікселів. що світився, як маяк у ночі. Game Gear виконала цю обіцянку та забезпечила ігри консольного рівня на ходу, повністю емулюючи Sega Master System.

Схоже, що Sega перемогла Nintendo у гонці озброєнь, випустив пристрій, який не тільки перевершив Game Boy у технологічному відношенні, але й виглядав як науково-фантастичне майбутнє, на яке всі чекали. Але була лише одна проблема: його батарейний відсік виявився еквівалентом термального вихлопного отвору Зірки смерті.

Повнокольоровий дисплей з підсвічуванням коштував дорого, тому що Game Gear їла батарейки типу AA на завтрак, поглинаючи їх жадібними

ковтками по шість штук за раз. Це принесло від двох до трьох годин ігрового часу в русі – навіть Sonic the Hedgehog не міг так швидко проходити рівні.

Game Boy, з іншого боку, був Duracell Bunny серед 8-бітних портативних пристроїв, здатних працювати від 10 до 14 години на чотирьох батареях AA, саме те, що вам потрібно, граючи в дорозі [24].

Game Gear справила відзначене нагородами враження від лінійки Master System: класичні ігри Sega, такі як Sonic, Street of Rage, Space Harrier та Ecco the Dolphin без жодних поступок з'явилися в системі.

Порти, випущені не для Sega, показали себе однаково добре – погляньте на Prince of Persia і криваву класичну Mortal Kombat від Midway, але оригінальність не була їх сильною стороною. Пристойні платформні ексклюзиви, такі як X-Men 2: Game Master's Legacy to Tails Adventure, остронь, переважна більшість бібліотеки Game Gear складалася з портів Master System, а стороння підтримка була відсутня.

Тим часом Game Boy був унікальним пристроєм з оригінальними іграми, підкріпленим програмною бібліотекою, яка затьмарила свого конкурента з чудовими характеристиками, і Покемон миттєво став глобальним явищем, яке започаткувало розраховані на багато користувачів гру і обміну культурою в іграх.

Game Boy був далеко не таким гарним консольним портом – Street Fighter 2 був хитким безладом, а FIFA International Soccer досягла таких же результатів, як і перебування Стіва Макларена в Ньюкасл Юнайтед, але стороння підтримка була великою [24].

Історія протистояння Game Boy і Game Gear – це історія того, хто програв, щоб змагатися з Роккі: система Sega була Голіафом, який перевершив свого конкурента в усіх відношеннях, за винятком того, де це дійсно мало значення. На той час появи Game Boy ціна на Game Boy перебувала у вільному падінні. І вирішальні чинники, зроблені Game Boy безперечним королем свого покоління. Зображення Game Boy та Game Gear представлено на рис. 1.7.



Рисунок 1.7 – Game Boy vs Game Gear [265]

За підсумками нічого більше від компанії Sega не виходило, а Nintendo продовжувало розвивати цей напрямок аж до конкуренції з Sony PSP і Nintendo 3DS. Примітно, що консоль від Sony була цікавіша, потужніша і більш функціональна ніж консоль від Nintendo, проте їх продажі на ринку мають плюс мінус однакові цифри [26].

У 2011 році Sony вирішує випустити наступне покоління консолей – PS Vita. На папері все було чудово, консоль мала сенсорний дисплей, другий стик, камеру, покращений Wi-Fi і Bluetooth і так далі [26].

Але насправді користувачі отримали малофункціональну цеглу, яка мала неадекватні ціни на ігри, погане ПЗ і найнеприємніше – карта пам'яті. Sony боячись конкуренції зробили свою картку пам'яті для PS Vita – PS Vita Memory Card. Проблема цієї картки на той момент її ціна за такий маленький обсяг пам'яті: 16 ГБ за 70 доларів США на 2011 рік, коли ціна консолі була 250 доларів США. Як результат консоль продася жакливим тиражем за мірками консолі, швидко втратила свою популярність і затребуваність, і Sony просто пішла з ринку портативних консолей [26].

У 2017 році виходить Nintendo Switch – наступне покоління портативних консолей Nintendo, що здобуло величезну популярність, скажені продажі як в Японії так і у світі незважаючи на свої мінуси, а їх дуже багато.

Закрита система – перша і найбільша проблема цієї консолі, крім ігор на ній не можна робити – нічого, відсутність YouTube, браузера, можливість дивитися відео, фото і так далі. Самі Nintendo заявляють, що це один із способів боротьби з піратством, однак це безглуздо і позбавляти гравців можливості серфити в інтернеті на консолі, яка має підключення до Wi-Fi – дивний хід.

Наступна проблема – доступ до онлайн ігор та комунікація з іншими гравцями. Що б пограти в ігри разом із друзями треба заплатити за передплату та аналогічно з комунікацією. Для зв'язку з користувачами потрібно також купити підписку, проте зв'язок йде не через саму консоль, а через смартфон [27].

І остання проблема – занадто високі ціни на ігри, які не мають прив'язку до регіонів. Незважаючи на все це користувачі просто змирилися з цим і консоль досі користується величезною популярністю і відмінно продається [27].

Нещодавно в "гру" вступили Valve з їхньою консоллю – Steam Deck.

Неймовірний пристрій, який поступово заповнює ринок і має купу плюсів. По-перше, своя система – SteamOs. Це Linux система мають всю свободу дій для користувача, за наявності адаптера консоль перетворюється на повноцінний ПК. За бажання можна поставити туди Windows, проте відсутність нормальних драйверів не дозволить повністю насолодитися іграми. Найприємніше, що не потрібно платити не за якісь підписки, як це зробили Nintendo. Ну і найголовніше – це повноцінний доступ до Steam та ігор, які вже є у користувача на обліковому записі з можливістю купувати ігри за регіональними цінами. Демонстрація видів та методів використання портативних пристроїв (Steam Deck та Nintendo Switch) наведено в таблиці 1.2.

Таблиця 1.2 – Steam Deck та Nintendo Switch

Вид пристрою	Опис
	Nintendo Switch [28]
	Перехідники для консолі, щоб вивести зображення на телевізор [29]
	Steam Deck [30]
	Steam Deck як ПК [31] Помилка! Джерело посилання не знайдено.
	Перехідник для Steam Deck [3210]

1.4 Мобільні ігри

Мобільні відеоігри, крім відеоігор, які просто доступні для мобільних пристроїв, є типом відеоігор, явно створених з урахуванням декількох варіантів дизайну, щоб змусити або заохочувати певну поведінку серед своїх гравців. Багато мобільних відеоігор не призначені для безперервної гри; у них часто буде механіка збору ресурсів, яка обмежена за часом. Це часто означає, що ігровий процес розділений на безліч слотів, кожен з яких триває лише кілька хвилин [3314].

Величезна проблема всіх мобільних ігор є постійна нав'язлива монетизація всіх процесів геймплею, шляхом штучного ускладнення гри.

Існує уповільнення прогресу шляхом величезного часу апгрейду, якщо на початку гри користувач отримує багато бустів і будівлі прокачуються за хвилини, то з кожним новим рівнем час апгрейду може збільшуватися до днів або місяців очікування. На це є тільки одне рішення, а саме донатні послуги за солідні гроші які пропонує гра, представлено на рис. 1.8.

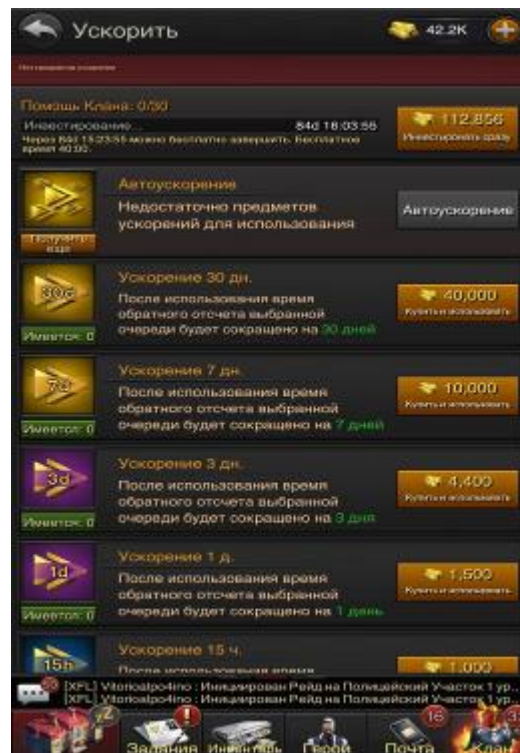


Рисунок 1.8 – Скріншот з гри Game of Mafia [34]

Але так само є і багато ігор як платних, так і безкоштовних, які мають інші особливості та різноманітність геймплею. Наприклад гри де відсутня система апгрейду можуть містити рекламу яка спливатиме постійно і для вирішення цієї проблеми або "купіть версію гри без реклами", або можна вимкнути інтернет на пристрої, але тим самим користувач стає повністю недоступним для мережі і втрачає комунікацію із зовнішнім світом.

Технічні можливості смартфонів дійшли до моменту, коли телефон може стабільно запускати та відтворювати повноцінні порти ігор з інших платформ, природно з адаптованою графікою під пристрій. Приклад такого порту це Fortnite, який давав можливість геймеру зі смартфона грати з користувачами інших платформ таких як ПК та консолі – разом. Зображення порівняння Консольної та мобільної версії Fortnite представлено на рис. 1.9.



Рисунок 1.9 – Порівняння консольної та мобільної версії Fortnite [35]

На жаль через фінансовий скандал гри була прибрана з цифрових платформ, але є багато інших ігор адаптованих під крос-платформи.

Крос-платформна гра або crossplay – це комп'ютерна гра в онлайн режимі з використанням різних платформ для одноразового підключення. [36]

До таких ігор можна так само віднести Wild Rift, Genshin Impact і багато інших.

Але є також ігри які спочатку робилися під усі пристрої. Той самий Genshin Impact приклад такої гри, щоб користувачі могли зайти в гру з будь-якої точки знаходження.

Як повноцінних жанрів на мобільні пристрої відсутні. Все впирається в ігри клікери або умовні перегони типу серії ігор Asphalt.

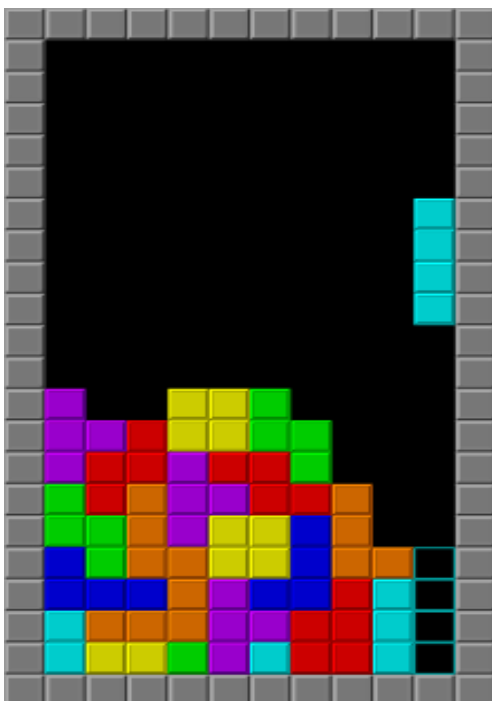
Інкрементні ігри (також відомі як неактивні ігри, ігри з клацаннями або ігри з клацаннями) – це відеоігри, ігровий процес яких полягає в тому, що гравець виконує прості дії (наприклад, натискання на екрані) кілька разів, щоб отримати валюту. Це можна використовувати для отримання предметів або здібностей, які збільшують швидкість накопичення валюти [37].

Однак на телефонах користуються популярністю "класичні" ігри такі як: Шахи, шашки, карти, тетріс і так далі. Креативні розробники роблять все можливе, що ти ці ігри виглядали по-новому, додаючи ефекти, 3D графіку, фарби тощо. Для порівняння – класичний тетріс та тетріс для смартфонів. Зображення Tetris Classic та Tetris Classic представлено на рис. 1.10.

Останніми роками індустрія відеоігор невинно розвивається з численними ігровими можливостями. Зі збільшенням продажів цифрових копій ігор, мобільних і кросплатформних ігор, не прив'язаних до однієї конкретної консолі, а також зростанням популярності сервісів потокового передавання ігор, таких як Game Pass від Microsoft, і сервісів онлайн-розповсюдження, таких як Steam, з'явилися нові можливості. значно розширився.

Найпопулярніші ігри з часів появи телефонів на базі Android були ігри раннери. Першою такою грою була – Temple Run.

Шукач скарбів вибігав з печери і намагався втекти від величезної кулі, яка хотіла розчавити персонажа.



Tetris Classic [38]



Tetris Classic [39]

Рисунок 1.10 – Візуальне порівняння гри тетріс

Через популярність гри з'явилося безліч клонів, але найуспішніший з них – Subway Surfer. Гра залишається найпопулярнішим ранером і на сьогоднішній день для користувачів, оскільки гра проста і добре вбиває час [40].

Хоча Temple Run і Subway Surfers є іграми з безкінечним бігом, між ними є значні відмінності. Геймплей Subway Surfers базується на швидкості і спритності, а художнє оформлення в ньому більш яскраве і мультяшне. У той час як Temple Run більше зосереджений на виживанні та стратегії, в ньому присутні більш реалістичний і таємничий вигляд. Обидві гри мають лідерські таблиці і щотижневі виклики для соціальної взаємодії, а Temple Run також має можливість багатокористувацької гри. Temple Run вимагає від гравців проводити пальцем у різних напрямках, щоб уникати перешкод, тоді як Subway Surfers має більш просту систему керування, засновану на дотику. Обидві гри також мають трохи різні схеми управління [4021].

Порівняння ранерів представлено в таблиці 1.6.

Таблиця 1.6 – Порівняння ранерів

Характеристика	Subway Surfer	Temple Run
Стиль	Мультишний	Реалістичний
Фокус гри	Швидкість та ловкість	Виживання та стратегія
Соціальні функції	Щотижневі завдання, таблиці лідерів	Щотижневі завдання, таблиці лідерів, мультиплеєр
Система контролю	Натискання на екран	Проведення по екрану
Спеціальні можливості	Реактивні ранці, ховерборди, прискорення	Щити, прискорення, можливість відродження
Середовище	Міське середовище	Джунглевє середовище

1.5 Висновки до розділу 1

В ході аналізу предметної області було проведено огляд існуючих типів ігор. Було виявлено, що ігри на сьогоднішній день насичені різноманітністю та розмаїттям. Розглянуті були різні жанри, починаючи від екшену та пригод до стратегій та симуляторів. Кожен жанр має свої особливості та цільову аудиторію.

Під час огляду було виявлено, що одним з популярних типів ігор є нескінечні раннери. Цей жанр характеризується неперервним рухом головного персонажу через серію перешкод, де гравець повинен уникати зіткнень та досягати максимальної дистанції.

Детальне вивчення існуючих типів ігор дало цінну інформацію, яка послужить основою для подальшого проектування та розробки розважального мобільного застосунку.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ

2.1 Технології розробки розважальних застосунків

При розробці комп'ютерних ігор використовуються різні технології, включаючи Adobe Flash un ActionScript, HTML, CSS, AJAX , Java, C# , Unity3D , Adobe (Macromedia) Director, C++, PHP, ASP.NET тощо.

Найбільш популярними Flash і Java, так як ці технології просто в створенні. інтерактивні ігри анімації. Однак Adobe Flash є більш Flash-додатки можуть працювати на 99% комп'ютерів, підключених до Інтернету, в той час як Java-додатки можуть працювати тільки на 73% комп'ютерів [41].

Технологія Adobe Flash призначена для виробництва інтерактивного відео. Multimedia-потужний редактор векторної графіки, який може Потужний редактор векторної графіки для створення двовимірної векторної графіки. Вони використовує векторну графіку, анімацію і графічні зображення, а також включає звук і відео. Для розробки Flash-додатків використовуйте такі продукти, як Adobe Flash. Professional, Adobe Flash Builder, Adobe Flex SDK, FlashDevelop і об'єктно-орієнтована мова ActionScript, а також Flash Player для виконання. Остання версія Flash Player 11, випущена в 2011 році, забезпечує підтримку тривимірної графіки і значне збільшення продуктивності Flash Player [41].

Flash-програми можуть використовувати нову версію FlashPlayer 11, випущену в 2011 році і підтримує 3D-графіку. На жаль, засоби розробки для цієї версії поки не Засоби розробки для цієї версії поки недоступні. В даний час розробка Flash-додатків зазвичай ведеться з використанням Flash Player 10 та середовища розробки Adobe Flash Professional CS5.5.

Розробка Flash-додатків ведеться за допомогою Adobe Flash Professional CS5.5. комп'ютерних ігор може бути виконаний трьома способами:

- використовуючи середовище розробки Adobe Flash Professional. Flash-додатки можуть бути створені без знання програмування;
- використовуючи Adobe FlashBuilder, продукт для програмістів, в якому можна використовувати бібліотеки та об'єкти. Однак їх використання може збільшити розмір програми;
- використання бібліотек Adobe Flex SDK. У цьому випадку для розробки розробка і компіляція файлів ActionScript в цьому випадку. FlashDevelop. Примітно, що на відміну від попередніх продуктів Adobe Flex SDK і FlashDevelop продукти надаються безкоштовно.

Враховуючи, що за продукти Adobe Flash Professional і Adobe Flash Builder доводиться платити, Кращий спосіб легко і швидко створювати Flash-додатки – використовувати Adobe Flash Professional. Можна припустити, що Adobe Flash Professional можна використовувати для швидкого та простого створення програм та підключення спецефектів, Рішення полягає у використанні FlashDevelop, середовища розробки для швидкого і простого створення Flash-додатків та підключення спецефектів, що потребують програмування (ActionScript), з Adobe Flash Professional [41].

Java багато в чому схожа на Flash, оскільки спрямована на створення Інтернет-додатків. Спеціально для цього була розроблена технологія під назвою JavaFx, більше схожа на Flex (попередник Flash), ніж Flash. І Flex, і JavaFx вперше черга націлена на розробку RIA-додатків (rich Інтернет-додатків). JavaFx дозволяє використовувати всю функціональність віртуальної машини Java Virtual Machine JavaFx дозволяє використовувати всі можливості віртуальної машини Java і пропонує особливі переваги в кількох областях, наприклад, при керуванні ланцюжками з кількох завдань, що виконуються. В час розробки додатків трохи простіше розробляти у Flex, оскільки деякі операції можуть бути виконані у Flex швидше, наприклад, відображення відео Java в даний час широко використовується в смартфонах: вона заснована на операційній системі Android; Flash – додатки також можуть також працювати на Android, але нині менш популярні [42].

У таблиці 2.1 наведено порівняння перелічених технологій.

Таблиця 2.1 – Порівняння технологій розробки ігор

Показник	Flash	HTML, CSS	AJAX	Silverlight	Java	Shockwave	Unity
Сумісність браузерів	Немає	Є	Є	Немає	Немає	Немає	Немає
Індексація змісту	Немає	Є	Є	Є	Є	Немає	Немає
Обробка зображень	Є	Немає	Немає	Є	Є	Є	Є
Векторна підтримка графіки	Є	Є	Є	Є	Є	Є	Є
Постійне з'єднання із сервером	Є	Немає	Немає	Є	Є	Є	Є
Підтримка мультимедіа – аудіо, відео	Є	HTML5	HTML5	Є	Є	Є	Є
Середовище функціонування	Windows, MacOS, Linux, Android	Всі	Всі	Windows, MacOS, Linux, Windows Mobile	Windows, MacOS, Linux, мобільні пристрої (без iOS)	Windows, MacOS	Windows, MacOS, Android, iOS, tvOS, Playstation, Xbox
Необхідний plug-in /інсталяції	Так, 3 MB	Немає	Немає	Так, 6 MB	Так, 16.1 MB	Так, 6.4 MB	Так, 600 KB
Індексація змісту	Немає	Є	Немає	Є	Немає	Немає	Немає
Операційні системи для розробки	Windows, MacOS	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows	Windows, MacOS, Linux	Windows, MacOS	Windows, MacOS
Підтримка 3D	Є	Немає	Немає	Є	Є	Є	Є
Доступність технології	99 %	100 %	100 %	~ 60 %	77 %	52 %	< 1%

Для порівняльного аналізу технологій, що використовуються при розробці комп'ютерних ігор, були обрані Flash і Java, які є найчастіше

використовуваними; оскільки HTML, CSS та AJAX використовуються разом для створення рішення; Silverlight, як продукт Microsoft, багато в чому схожий на Flash;

Shockwave – по функціональності схожий на Flash, більш того, у своїх додатках схожий на функціональність з Flash, але на практиці часто додатково використовує Flash [4315].

Unity – технологія для створення мережевих додатків, зокрема, ігр.

Як видно з таблиці 2.1, між розглянутими технологіями існує багато спільного. Основною відмінністю є доступність (скільки комп'ютерів встановлено). Існують також відмінності між технологіями в особливостях розробки додатків але ці питання не розглядаються. Тому перерахувати кращі технології для розробки гри не є можливим. Технологія для створення комп'ютерних ігор-її слід вибирати з наступних причин – слід вибирати в залежності від конкретної задачі наявності ресурсів. Наприклад, при розробці гри, що включає велику кількість рухомих об'єктів, найбільш підходящою є Flash. Якщо. програма, розроблена для мобільних пристроїв.

2.2 Захоплення рухів в іграх

Захоплення руху, або скорочено мокап – це процес запису руху та його перетворення на цифрову анімацію. Він використовується ігровою індустрією з 90-х років, коли Mortal Kombat стала першою відеогрою, в якій рухи актора копіювалися на цифровому персонажі. За останні десятиліття технологія захоплення руху для відеоігор пройшла довгий шлях і тепер може реалістично відтворювати рухи двоногих (гуманоїдів) у 3D.

Для незалежних розробників та невеликих ігрових студій захоплення рухів – це доступний та швидкий спосіб створення анімації та прискорення виробничого процесу. Записати анімацію набагато швидше, ніж створювати ключові кадри вручну традиційним методом [4440].

У розробці ігор є два підходи до переміщення анімованого персонажа:

– рух, керований анімацією – відноситься до руху, який слідує за основною анімацією. Він виглядає більш плавним та кінематографічним, але гравець не може так швидко керувати діями;

– рух, керований сценарієм – це кореневий рух, керований кодом. Ігрові гравці можуть мати мілісекундний час реакції, але рухи будуть уривчастими і не такими плавними. Цей стиль часто використовується у файтингах від першої особи.

Використання захоплення руху підходить для всіх цифрових анімацій персонажів, але знання того, який стиль руху можна використовувати має вирішальне значення для ваших записів. Наприклад, рух, керований анімацією, може дозволити вашому мокап-актору бути більш плавним або екстравагантним у своїй діяльності. З іншого боку, під час запису руху для системи, керованої сценаріями, зведіть до мінімуму всі непотрібні дії, щоб анімація залишалася чистою та чіткою [4440].

У той час як відчуття відноситься до того, наскільки гравець відчуває контроль над персонажем, плавність більше пов'язана з тим, наскільки добре ваші анімації змішуються при переході між анімаціями. Можна значно вплинути на ці елементи в програмі, але рухи актора захоплення руху також відіграють велику роль у тому, як гравець відчуває конкретний рух. Наприклад, намагаючись покращити відчуття «свободи» гравця при переміщенні у цифровому світі, треба поставити такі питання.

Чи додає ця дія гравцеві почуття контролю?

Чи повідомляє цей рух про той вплив, який він має чинити?

Чи додає ця дія фантазії гравця про те, що він персонаж?

Ця анімація сприяє розвитку персонажа?

Плинність, з іншого боку, є систематичним процесом. В ідеальному світі треба, щоб ваші анімації плавно поєднувалися з небажаною деформацією. Для цього ми маємо побудувати «машину станів», здатну перемикає та змішувати зациклені та одноразові анімації. Кінцевий автомат повинен реагувати на введення гравця та швидко комбінувати анімації та

циклічно виконувати дії. Наприклад, персонаж може перемикатися між ходьбою бігом спринтом ходьбою стрибком, не відриваючи кінцівок і не відводячи погляду [4440].

Захоплення руху буває двох різних форм; оптичні та інерційні. Оптичне захоплення руху це тип відстеження, в якому використовуються камери і яскраві маркери для перетворення руху в дані. Інерційне відстеження зазвичай є костюмом мокапа і використовує крихітні гіроскопи, розміщені в ключових точках руху, для запису руху в дані анімації. Для захоплення руху за інерцією не потрібні спеціальні камери, спеціальний оператор та актор чи студійний простір. В інді-мокапі розробник може одягнути костюм, грати та записувати рухи самотійно [4440].

Типова установка захоплення руху буде використовувати інерційне відстеження руху для тіла та рук та оптичне захоплення руху для людського обличчя. Мокап обличчя можна зробити за допомогою чогось простого, наприклад, iPhone, або зняти у надвисокій роздільній здатності, використовуючи складну установку з кількома камерами. Ваш метод залежить від ступеня реалізму, якого можна досягти, та вашого бюджету. Припускаючи, що розробник шукає недорогої установки для всього тіла, повнофункціональний комплект захоплення Rokoko пропонує відмінне співвідношення ціни та якості для невеликої студії та є відмінним стартом. Він містить таке:

- Rokoko Smartsuit Pro: костюм для захоплення всього тіла;
- Rokoko Smartgloves: захоплення рухів рук;
- Rokoko Remote: програмне забезпечення для захоплення виразів обличчя (потрібен iPhone X або вище);
- Rokoko Studio Plus: програмне забезпечення для запису даних захоплення руху;
- додатково: безкоштовний плагін ретаргетинг для вашого ігрового движка (Unreal Engine / Unity) для запису даних анімації прямо на вашого персонажа в режимі реального часу.

2.3 Програмне забезпечення для візуалізації

Крім ігрових двигунів та інших технологій у розробці гра так само важлива візуальні об'єкти та інтерфейс. Для цього у великих компаніях є цілі групи людей – геймдизайнери. Вони ж діляться на групи людей, які займаються дизайном об'єктом, карти, персонажів і так далі.

Для роботи на такій посаді необхідні високі пізнання не тільки з ПЗ, але і мати уявлення в анатомії, архітектурі в залежності від поставлених завдань.

Blender – це абсолютно безкоштовний програмний інтерфейс 3D-комп'ютерної графіки з відкритим вихідним кодом, який використовується для створення 3D-моделей, анімації, візуальних ефектів, оснастки та багато іншого. Це галузевий стандарт для створення в 3D-цифровому просторі, і, поєднавши програмне забезпечення для друку Cuga з 3D-принтером, де можна взяти свою ідею, змоделювати її в 3D і надрукувати [4655].

Створюючи свій перший проєкт, треба ознайомитися з інтерфейсом Blender, який включає:

Налаштування сцени: це перехід з режиму об'єкта в режим редагування де можна знайти різні інструменти та функції для додавання до сцени.

Основні інструменти: у режимі об'єкта є основні інструменти для взаємодії з об'єктом. Починаючи згори, можна бачити курсор, походження світу, переміщати, обертати, масштабувати та трансформувати.

Вікно перегляду: вікно перегляду – це головне вікно або вид, який використовуватиметься для огляду об'єктів у 3D-сцені.

Колекція сцен: містить усі об'єкти сцени. Blender автоматично створює порожню сцену з кубом, світлом і камерою. Оскільки до сцени додається більше форм і об'єктів, творці захочуть, щоб ця область була організована та названа належним чином [4555].

Редактор властивостей: показує та дозволяє редагувати всі активні дані, включаючи активну сцену та об'єкт у сцені. Зображення прикладу роботи в Blender представлено на рис. 2.1.

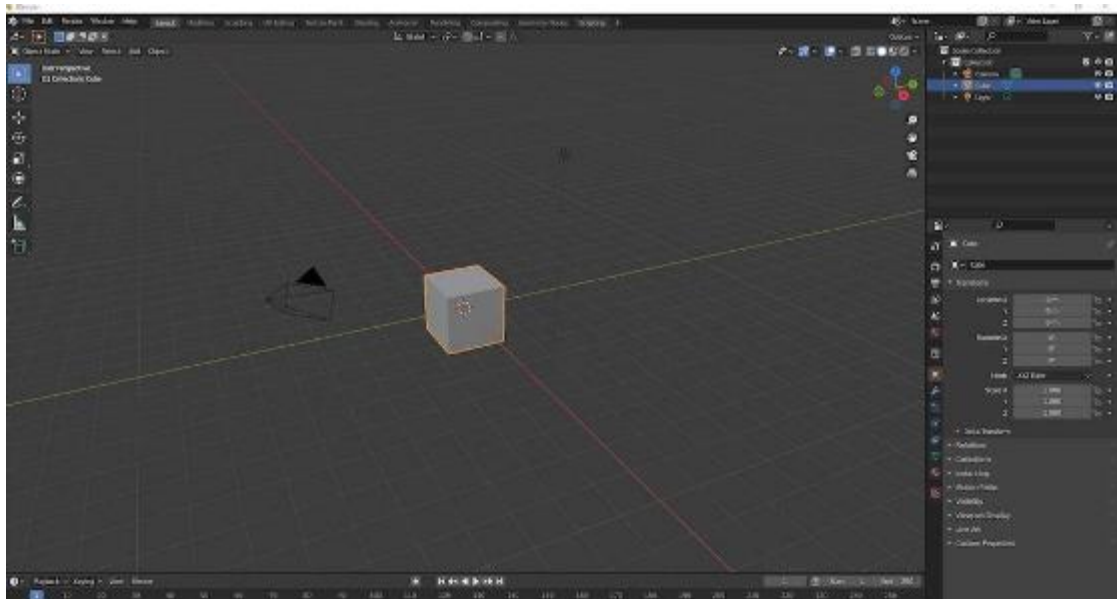


Рисунок 2.1 – Blender [48]

Примітиви – це попередньо визначені геометричні фігури в Blender. Це можуть бути куби, сфери, циліндри – прості фігури, які можна використовувати як відправні точки для більш складних об'єктів. Зображення примітивів багатокутників представлено на рис.2.2.

Затінення вікна перегляду. Існує чотири основні режими відображення, які змінюють вигляд 3D-об'єктів на екрані.

Каркас, починаючи зліва: з'явиться лише рамка об'єкта, де можна бачити крізь свій об'єкт.

Суцільний режим: об'єкт є суцільним, і можуть бути застосовані тіні.

Режим попереднього перегляду матеріалу: об'єкт буде затінено різними кольорами та зображеннями [48].

Відображено: об'єкт відображатиметься як повністю текстурований і відрендерений об'єкт (рис.2.3).

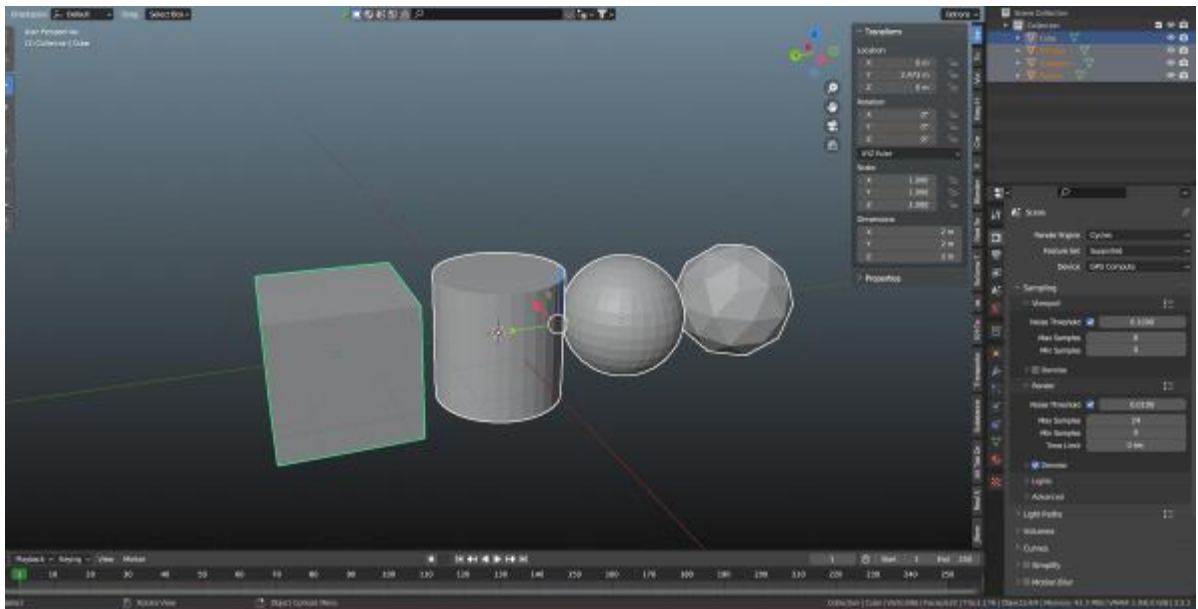
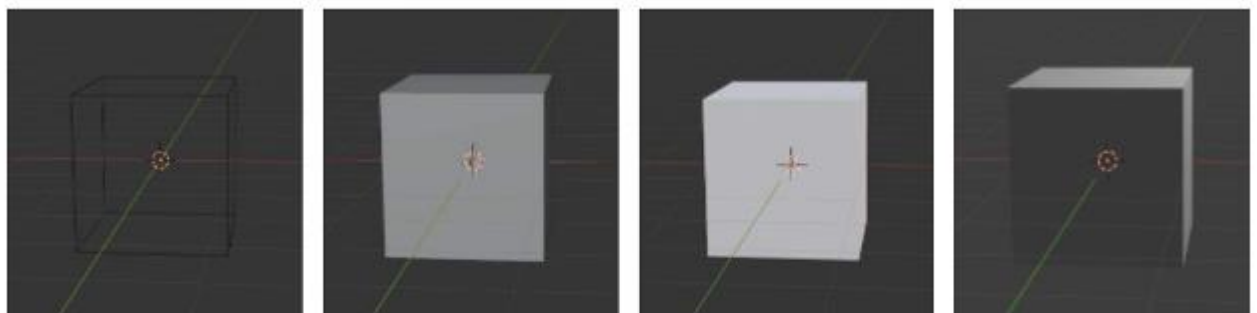


Рисунок 2.2 – Фігури [4725]



Wireframe

Solid

Textured (You won't see this pattern on your cube without adding materials to it)

Rendered (Rendered without a texture added is grey)

а

б

в

г

а – сітка; б – твердий; в – текстурований; г – відтворення

Рисунок 2.3 – Приклад типів відображення в Blender [46]

При розробці використовується частіше всього для створення сцени та окремих моделей середовища. Але краще для розробки сцени буде 3ds Max.

Раніше відома як «3D Studio», 3ds Max – це програма для 3D-моделювання, анімації та візуалізації від підрозділу медіа та розваг. Програмне забезпечення, яке широко використовується в інтерактивних іграх, візуальних ефектах для фільмів і моделях промислового дизайну, дозволяє створювати 2D-фігури, які стають перерізами 3D-моделей. Програма містить модуль анімації, який використовує інверсну кінематику,

яка пов'язує компоненти таким чином, щоб вони рухалися разом, додаючи до ефекту оживлення персонажа [48].

3ds Max – це програма для 3D-моделювання, яка використовує багатокутне моделювання. Моделювання багатокутників дає користувачам певний контроль над окремими сітками через координати X, Y і Z, які відображаються як група вершин, ребер і граней [48].

Користувач часто починає з примітивної форми, як простий екструдований об'єкт. Потім форму додатково визначають шляхом додавання додаткових деталей, наприклад маніпулювання наявними поверхнями.

Кожній частині 3D-моделі може бути призначена текстура. Використовуючи планарне відображення та інші робочі процеси з текстурами, можна створювати необмежену кількість текстур. Тривимірні моделі з деталями та текстурами створюють чудові візуалізації [48].

Іншою основною функцією 3ds Max є його здатність відтворювати 3D-моделі, що призводить до нерухомого зображення. Студія персонажів 3ds Max або плагін Character Animation Toolkit (CAT) є варіантами для анімації ресурсів на основі персонажів. Студія персонажів дозволяє анімувати двоногих персонажів, а за допомогою плагіна CAT можна створювати нелінійну анімацію, моделювати м'язи та інші функції фальсифікації персонажів [48].

3ds Max пропонує універсальний інтерфейс, який виділяється серед конкуруючих продуктів, таких як Maya та Blender. (Детальніше про це пізніше.) 3ds Max дозволяє користувачам активувати більшість функцій програми з кількох елементів інтерфейсу, що забезпечує високоефективний робочий процес [48].

Користувачі також можуть налаштувати свій інтерфейс користувача кількома різними способами. Наприклад, панелі інструментів і панелі команд можна переміщувати для більш плавного робочого процесу. Користувацькі комбінації клавіш, нові панелі інструментів і кнопки, а також спеціальні

сценарії запису можуть покращити інтерфейс. Зображення прикладу інтерфейсу 3ds Max представлено на рис. 2.4.

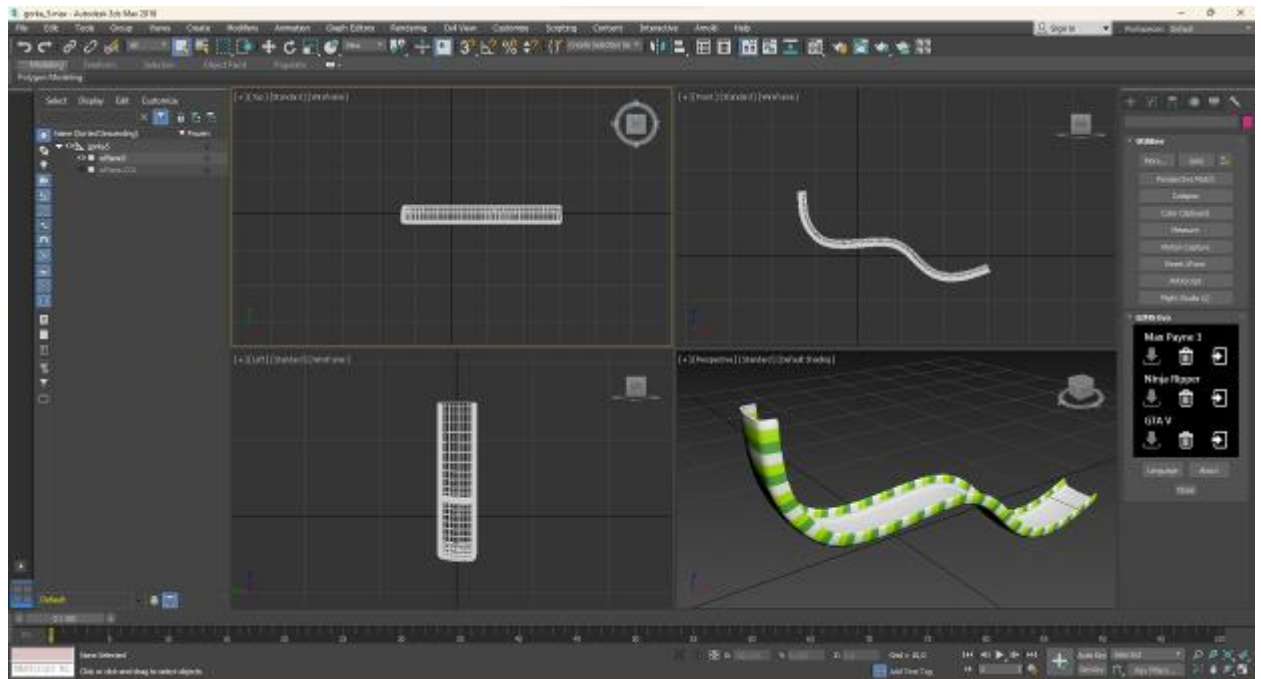


Рисунок 2.4 – Приклад інтерфейсу 3ds Max [49]

У 3ds Max є все необхідне для роботи на професійному рівні, але він також чудово підходить для студентів та ентузіастів.

Ігрові студії зазвичай використовують 3ds Max для створення персонажів та інших основних ресурсів у розробці ігор. Потім ці ресурси часто імпортуються в інші програми, такі як Mudbox, щоб текстурувати їх ще більше [4820].

Архітектори та дизайнерські фірми часто використовують 3ds Max для розробки концептуального мистецтва. Це дозволяє клієнтам побачити концептуальні візуалізації перед завершенням деталі проєкту.

Професійні аніматори – це ще одна значна група, яка використовує 3ds Max, оскільки це програмне забезпечення чудово підходить для роботи над фільмами, включаючи високобюджетні та інді-фільми. Наприклад, популярний фільм 2012 року «Аватар» використовував 3ds Max для CGI – анімації разом із акторами, які грали в живих. Рекламні ролики також часто

використовують анімацію, створену в 3ds Max, щоб додати спеціальні ефекти до живих кадрів.[4823]

Maya – це популярна програма комп’ютерної тривимірної графіки, якою користуються як професіонали, так і любителі. Він здатний до анімації, моделювання, ліплення, симуляції та відтворення реалістичних об’єктів і персонажів, інтерактивних сцен і захоплюючих ефектів. Maya може і використовується практично всіма: від дизайнерів ігор і 3D-моделувальників до аніматорів і художників VFX у таких франшизах, як «Володар кілець», «Гаррі Поттер» і «Зоряні війни» [50].

Оскільки Maya так багато може зробити, ми розбили його на частини, починаючи з інсталяції та просуваючись до корисних інструментів, закінчуючи колекцією ресурсів і посібників, якщо зануритися ще глибше. Загалом 3D-моделювання використовує програмне забезпечення для створення тривимірних об’єктів, таких як будівлі, деталі машин або персонажі для проєктування, ігор чи анімації. Зазвичай будь-яка програма 3D-моделювання може бути орієнтована лише на одну з цих професій, але, як уже згадувалося, Maya здатна робити все це [50].

Є кілька інших програм, схожих на Maya, наприклад Blender, 3ds Max і ZBrush. Blender і Maya досить схожі, але Maya нібито більш зручна для користувача. Деякі функції Maya, як-от візуалізація та VFX, є більш професійними. Доступ до самих програм також є різницею: Blender безкоштовний, а Maya має ціну. Zbrush – це одне з, якщо не найпопулярніше програмне забезпечення з високою оцінкою, яке використовується для цифрового моделювання та ліплення. Безстрокова ліцензія на ZBrush коштує майже 1000 доларів США, а місячна вартість становить приблизно 30 доларів США, що приблизно в сім разів менше, ніж щомісячна плата Maya [51]. Зображення інтерфейсу програми Maya 3D представлено на рис. 2.5.



Рисунок 2.5 – Maya 3D [52]

ZBrush – найдосконаліша програма для 3D-скульптури. Що відрізняє його від інших 3D-інструментів, так це те, що ZBrush імітує традиційні техніки ліплення, які виконуються в цифровому вигляді на комп'ютері.

Ліплення в ZBrush схоже на роботу з цифровою глиняною кулькою, надаючи їй форму, ніби воно робить вручну.

Інструменти для ліплення від ZBrush надають широкий ступінь творчої свободи. Художники не тільки можуть створювати більш органічні та детальні моделі за допомогою ZBrush, вони часто можуть отримати готовий продукт набагато швидше, ніж за допомогою інших програм, таких як Maya або 3ds Max.

Розроблений Pixologic, Inc., ZBrush був вперше представлений у 1999 році та офіційно випущений для ПК та Mac у 2009 році.

Сьогодні він визнаний найдосконалішим програмним забезпеченням для 3D-скульптури на ринку. Завдяки великому набору інструментів і широкому спектру програм легко зрозуміти, чому ZBrush став таким успішним.

Найперші версії ZBrush містили 30 інструментів пензля, які дозволяли виконувати різноманітні процеси та більше контролю. Використовуючи ці пензлі, художник може виліпити цифрову версію майже всього, що забажає. Зображення інтерфейсу програми Zbrush представлено на рис. 2.6. Оскільки ZBrush використовується для створення 3D-моделей із високою роздільною здатністю для фільмів, відеоігор і спеціальних ефектів, є дуже висока ймовірність того, побачит щось, що містить мистецтво, створене в програмі [5319].

ZBrush – поширений інструмент для професійних кіностудій та студій відеоігор у всьому світі [5319].

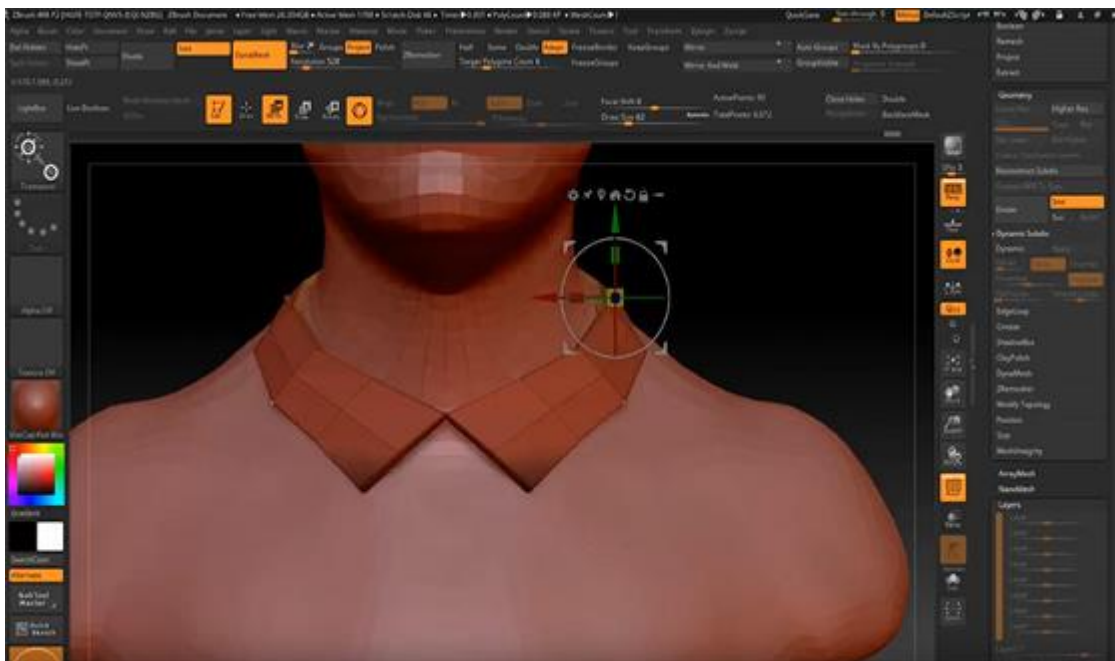


Рисунок 2.6 – Zbrush [54]

Художники ZBrush також працюють у вищих навчальних закладах, маркетингу та скрізь, де є попит на високоякісну 3D-анімацію та ефекти.

Більшість ігор AAA на ПК та консолях наступного покоління використовують ZBrush у своєму конвеєрі анімації. Використовуючи ZBrush, сучасні ігрові студії можуть деталізувати другорядних персонажів і створювати складні світи, не додаючи часу на розробку [53].

Такі ігри, як «God of War» і «Assasins Creed», скористалися цими функціями ZBrush.

Одна з найкращих шкіл концепт-арту у світі FZD School of Design має лише 2 обов'язкові програми для вивчення концепт-арту: Photoshop і ZBrush. Можна побачити це в їхньому розділі реєстрації в розділі «плата за обладнання» [53].

Ті, хто вивчає концепт-арт і закінчив FZD зі ступенем мистецтва розваг, використовують лише ці дві програми. Це дійсно багато говорить про вплив цього програмного забезпечення на ігровий світ. Але ZBrush також популярний серед кіностудій і навіть отримав нагороду Академії за революцію в спецефектах [53].

2.4 Постановка завдань дипломної роботи

Проведений аналіз дозволив виявити особливості технологій, які використовуються для розробки розважальних ігрових застосунків.

У ході дослідження показано, що на сьогоднішній день існує розмаїття інструментаріїв та технологій, що дозволяють створювати ігри різних жанрів і на різних платформах. Вибір технології для розробки ігрового застосунку залежить від ряду факторів, таких як розмір команди розробників, вимоги до графіки, фізики та штучного інтелекту, підтримування платформи та бюджет проекту.

Проаналізувавши технології та методи розробки ігор, вирішено створювати мобільну гру враховуючи такі переваги :

- популярність мобільних ігор;
- доступність;
- можливість постійного апдейту;
- час розробки;
- можливість публікації на платформи (Play Market та AppStore).

Статистика показує, що найпопулярнішими іграми є клікери, зокрема раннери. Цей жанр був обраний через його популярність на мобільних пристроях та можливості гнучкої розробки.

Метою роботи є – розробка програмного забезпечення розважального мобільного застосунку для організації дозвілля дітей та дорослих.

Завдання роботи:

- дослідити методику розробки мобільної гри;
- виконати аналіз вимог та розробити архітектуру системи;
- проаналізувати та обрати інструментарій розробки;
- виконати програмну реалізацію та тестування програми.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог

Аналіз вимог до мобільної гри-раннера включає визначення функціональних та нефункціональних вимог до проєкту. Основна мета аналізу вимог полягає в розумінні очікувань користувачів та створенні основи для розробки гри. Нижче наведено загальний огляд ключових елементів аналізу вимог до мобільної гри-раннера:

Платформні вимоги:

- на якій версії операційної системи мобільного пристрою гра повинна працювати;
- встановити мінімальні вимоги щодо роздільної здатності екрану для забезпечення належного відображення графіки;
- розгляньте можливість додавання поширення досягнень гравців на соціальних медіа, наприклад, публікацію результатів на Facebook або Twitter.

Системні вимоги та розмір гри:

- встановити максимальний допустимий розмір гри для забезпечення ефективного завантаження та використання пам'яті пристрою;
- можливість забезпечення сумісності з різними моделями мобільних пристроїв та їх характеристиками.

Локалізація: визначити, на які мови повинна бути перекладена гра, залежно від цільової аудиторії. Найчастіше це англійська мова.

Вимоги до тестування та налагодження: інструменти та процедури для виявлення та усунення помилок та дефектів гри.

Тестування сумісності: забезпечити відповідне тестування гри на різних моделях та версіях мобільних пристроїв.

Аналіз вимог дозволить чітко визначити функціональність та характеристики, які повинні бути включені в мобільну гру-раннер, і встановити основу для успішного розроблення проєкту.

Функціональні вимоги:

- керування персонажем – гравець повинен мати змогу керувати персонажем гри, здійснюючи різні дії, такі як стрибок, прискорення, зміна полоси руху тощо;

- геймплей – розробка захопливого та викликаючого інтерес геймплею, включаючи перешкоди, збирання бонусів або монет, покращення характеристик персонажа та рівнів складності.

- графіка та анімація – створення привабливого візуального середовища, яке включає як персонажа, так і оточуючі об'єкти, з використанням якісної графіки та плавної анімації;

- звуковий супровід – додавання аудіоефектів та музики, щоб підсилити враження гри та створити належну атмосферу.

- Нефункціональні вимоги:

- сумісність з платформою – гра повинна бути розроблена для конкретної мобільної платформи, такої як Android або iOS, з дотриманням вимог технічних специфікацій та рекомендацій платформи;

- продуктивність – гра повинна працювати ефективно, забезпечуючи плавний рух персонажа та відповідну швидкість реакції на керування гравця;

- збереження даних - можливість збереження прогресу гравця та інших налаштувань в грі, щоб користувач міг продовжити гру з того ж місця або на іншому пристрої;

- безпека – захист гри від шкідливих програм або несанкціонованого доступу до внутрішніх файлів чи функцій.

3.2 Методика розробки мобільної гри

Схему алгоритму методики наведено на рис. 3.1.



Рисунок 3.1 – Схема алгоритму розробки гри

- Крок 1. Аналіз вимог та розробка концепції гри.
- Крок 2. Розробка архітектури проєкту.
- Крок 3. Аналіз та вибір ігрового двигуна та мови програмування.
- Крок 4. Аналіз та вибір графічного редактора.
- Крок 5. Створення компонентів.
- Крок 6. Компоновка об'єктів на сцені.
- Крок 7. Розробка скриптів і нанесення їх на об'єкти.
- Крок 8. Програмна реалізація.
- Крок 9. Тестування проєкту на мобільному пристрої.

3.3 Архітектура системи

Архітектура програми для розробки гри може варіюватися залежно від розміру, складності та типу гри, але основні принципи і концепції залишаються схожими. Архітектуру системи наведено на рис. 3.2.

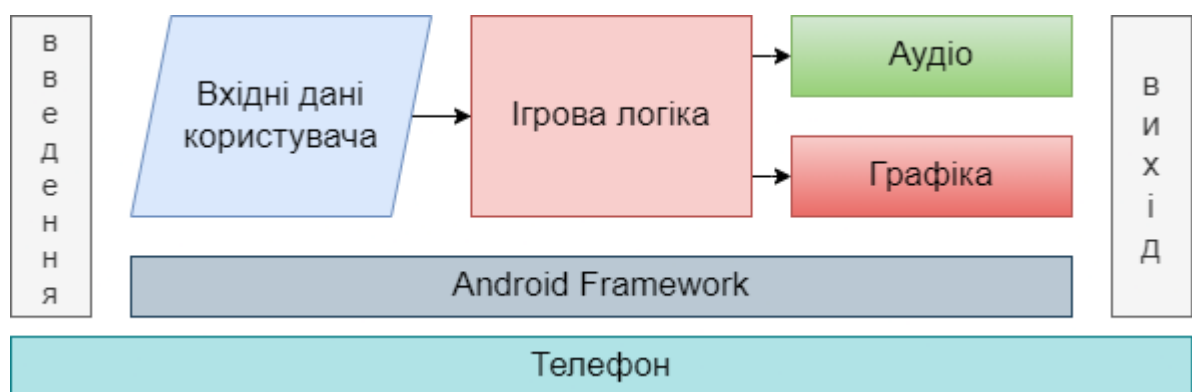


Рисунок 3.2 – Архітектура системи

Всі із загальних аспектів архітектури програми, які будуть використовуватися для розробки гри:

- Модель-Вид-Контролер (Model-View-Controller, MVC) – Ця архітектурна парадигма використовується для відокремлення логіки програми від її представлення та взаємодії з користувачем. Модель представляє дані та бізнес-логіку, Вид відображає графічний інтерфейс

користувача, а Контролер обробляє взаємодію користувача з грою та взаємодіє з Моделлю та Видом;

- компонентно-орієнтована архітектура – гра може бути розбита на набір компонентів, кожен з яких відповідає за певну функціональність або елемент гри. Компоненти можуть бути незалежними, перевикористовуваними та замінюваними, що спрощує розробку та розширення гри;

- двигун гри – гра буде розроблена на двигуні Unity. Двигун вже надає скелет архітектури та набір інструментів для створення гри, включаючи роботу з графікою, фізикою, аудіо, введенням користувача та іншими аспектами гри;

- станова машина (State Machine) – станова машина використовується для керування різними станами гри, такими як головне меню, рівні, діалоги тощо. Станова машина дозволяє зручно керувати поведінкою гри залежно від поточного стану;

- модульність – розробка гри може бути поділена на модулі для кращого організації та керування кодом. Модулі можуть включати системи фізики, штучного інтелекту, управління персонажем, аудіо тощо. Це дозволяє розробникам працювати над окремими частинами гри паралельно та зручно об'єднувати їх в кінцевий продукт.

Важливо планувати та проєктувати архітектуру перед початком розробки, щоб забезпечити легкість розширення, підтримку та покращення гри у майбутньому.

3.4 Огляд та вибір інструментарію розробки

3.4.1 Порівняння та вибір двигунів

При розгляді процесу вибору інструментарію для розробки ігрового проєкту, доцільно враховувати ряд факторів, що мають вирішальне значення:

- проаналізувати вимоги проєкту, зокрема його розмір, тип гри, графічні вимоги та цільову платформу. Врахування цих аспектів дозволить вибрати інструментарій, який найбільш відповідає потребам проєкту;
- досвід розробки – розглянути власний досвід у використанні конкретних інструментів розробки ігрових проєктів. Оцінка власних знань та навичок допоможе визначити, наскільки комфортно й ефективно можна працювати з певним інструментарієм;
- ресурси та підтримка – провести аналіз наявності документації, ресурсів та рівня підтримки для кожного потенційного інструментарію розробки. Наявність доступних матеріалів та наявність підтримки впливають на швидкість навчання та рівень продуктивності при використанні інструментарію;
- спільнота розробників – розмір та активність спільноти розробників, пов'язаної з кожним потенційним інструментарієм. Активна та дружня спільнота розробників надає можливість отримувати підтримку, допомогу та спілкуватися з колегами, що сприяє взаємному збагаченню знань та досвіду.

У таблиці 3.1 наведено порівняння 7 найпопулярніших ігрових движків на 2023 рік, виходячи з опитувань і статистики ігор на яких вони були створені [55].

Unity є одним із найпопулярніших ігрових движків і основним інструментом для створення ігор для індивідуальних розробників не лише завдяки доступній бізнес-моделі, а й завдяки надійності редактора проєктів, щорічним вдосконаленням і, що найголовніше, його легкості. використання та зростаючої спільноти розробників по всьому світу. Оскільки сама Unity керує складними закулісними процесами (дисплей, фізичні процеси, інтеграція та кросплатформне розгортання) – це лише деякі з них, розробники можуть зосередитися на створенні систем, які оживляють ігри та забезпечують інтерактивну взаємодію в реальному часі [55].

Таблиця 3.1 – Порівняльна характеристика двигунів [55]

Двигун	Реліз	Найпопулярніші ігри	Ціна	ОС
Unity	2005	Genshin Impact	Безкоштовно	Win 7+, MacOS
UE4	1998	Fortnite	Безкоштовно	Win 7+, MacOS
Godot	2014	Cruetly squad	Безкоштовно	Win 7+, MacOS
GameMaker Studio	2013	Bayou island	Безкоштовно	Win 7+
Armory	2020	Armory 3D	Безкоштовно	Win 7+
CryEngine	2002	Crysis	Безкоштовно	Win 7+, MacOS
Defold	2002	Bring me cackes	Безкоштовно	Win 7+, MacOS

Редактор Unity містить багато інструментів для прискорення та повторення циклів розробки, включаючи ігровий режим для швидкої перевірки продуктивності в реальному часі:

- універсальний редактор: доступний для Windows, Mac і Linux, він містить інструменти як для художників, які створюють цікаві світи, так і для розробників, які впроваджують логіку гри та ідеальний геймплей;
- 2D і 3D: Unity підходить як для 2D, так і для 3D, надаючи всі необхідні функції для будь-яких потреб у будь-якому жанрі;
- інструменти пошуку: Unity має систему навігації, яка дозволяє неігровим персонажам вільно орієнтуватися в ігровому світі;
- навігаційні сітки створюються автоматично на основі ландшафтних даних і навіть враховують динамічні перешкоди, завдяки яким курс персонажа змінюється під час дії;
- інтерфейс користувача: вбудована система, яка дозволяє швидко та легко створювати інтерфейс користувача;
- фізичні механізми: скористайтеся всіма перевагами Box2D, нової фізичної системи DOTS і NVIDIA PhysX для створення реалістичних і ефективних ігор;

- спеціальні та сторонні інструменти: Unity Editor підтримує встановлення розширень, адаптованих до потреб вашої студії або завантажених із Asset Store, магазину, де можна знайти всі ресурси, інструменти та розширення для прискорення ваших проєктів;
- покращена командна робота: дивіться, над чим працюють інші, прямо у вікні редактора Unity, де ми проводимо більшу частину свого часу.

Плюси Unity:

- доступний і зрозумілий інтерфейс;
- скрипти на C#, JavaScript і Boo;
- повна інтеграція ігрового движка з середовищем розробки;
- підтримка імпорту великої кількості форматів;
- підтримка Tissue Physics (PhysX Cloth);
- можливість додавання функціональності;

GameMaker Studio. Популярний ігровий движок GameMaker Studio, створений у 2017 році, є найновішою версією продукту, який існує з 1999 року під багатьма назвами та ітераціями. Ігри, розроблені за допомогою ігрового движка GameMaker, підтримуються на багатьох платформах, включаючи Nintendo Switch [55].

Окрім візуального редактора, GameMaker Language можна використовувати для програмування настроюваної поведінки, яка виходить за рамки, доступні за допомогою візуального програмування. Як простий у використанні ігровий механізм, кожен може легко розробляти ігри з ним.

Як власний ігровий движок і фреймворк GameMaker Studio 2 не ідеальний для тих, хто шукає бюджетне рішення. Крім того, він також спрямований на інші двигуни, такі як Phasers. Розроблений ігровий движок дійсно має деякі обмеження щодо можливостей 3D, але він не може зрівнятися з візуальним редактором ігрових движків Unity, Unreal або Godot [55].

Переваги:

- простота використання – GameMaker Studio має простий та зрозумілий інтерфейс, що робить його доступним навіть для початківців у галузі розробки ігор;
- легка навігація – GameMaker Studio має інтуїтивно зрозумілу систему навігації та редагування, що полегшує роботу з інструментом;
- вбудований спрайт-редактор – GameMaker Studio має вбудований редактор спрайтів, який дозволяє створювати та редагувати графіку без необхідності використання сторонніх програм.

Недоліки:

- обмежені можливості – порівняно з іншими двигунами, GameMaker Studio може бути обмеженим у функціоналі та можливостях розробки складніших ігор.
- доступна тільки платна версія;
- мала спільнота розробників.

Armory. З Armory розробникам більше не потрібно стрибати між програмами для розробки ігор, оскільки Blender є популярним вибором серед незалежних розробників.

Armory найкраще вивчати, ознайомившись із Blender та його широким набором інструментів. Armory – чудовий ігровий движок із відкритим вихідним кодом, якщо вивчити Blender [55].

Переваги:

- інтеграція з Blender – Armory інтегрований з популярним 3D-пакетом Blender, що дозволяє розробляти ігри безпосередньо в Blender;
- відкрите програмне забезпечення: Armory є вільним та відкритим двигуном, що надає доступ до вихідного коду та можливість модифікувати функціонал;
- підтримка веб-розробки – Armory дозволяє створювати веб-ігри з використанням HTML5 та WebGL.

Недоліки – має меншу спільноту розробників порівняно з більш відомими двигунами, що може вплинути на доступ до документації та підтримки.

CryEngine – ігровий движок, створений німецькою компанією Crytek у 2002 році та спочатку використовувався в шутері від першої особи Far Cry [55].

Багато різних ігор було створено на CryEngine різними студіями, які ліцензували двигун: Far Cry, Crysis, Entropia Universe, Blue Mars, Warface, Homefront: The Revolution, Sniper: Ghost Warrior, Armored Warfare, Evolve та багато інших. У березні 2016 року Crytek оголосила про випуск свого нового CryEngine V і незабаром опублікувала вихідний код на Github [55].

Плюси CryEngine:

- можливість прикрасити гру чудовою графікою;
 - створить потужний саундтрек;
 - початківець програміст зможе з легкістю створити інтерфейс користувача;
 - розробка ігор для ПК, Xbox, PlayStation з підтримкою DirectX
- Open source game engine in C++.

Unreal Engine – це сучасний ігровий движок на основі C++, розроблений і підтримуваний Epic Games. Можливості EU дозволяють створювати та редагувати елементи тривимірної анімації, спецефекти для фільмів та ігор, розробляти різноманітні навчальні програми. Програмний код на цьому движку працює на більшості сучасних платформ і операційних систем (Android, iOS, Linux, Mac OS, Windows, PlayStation 4, PSP, Xbox One, PS Vita). Отже, однією з особливостей Unreal Engine є його універсальність.

Переваги:

- потужні графічні можливості – Unreal Engine 4 володіє потужним і вражаючим графічним рушієм, який дозволяє створювати високоякісну, кіномотграфічну графіку;

- велика спільнота розробників, що забезпечує доступ до багатой документації та ресурсів;
- мова програмування C++, що дає розробникам більшу гнучкість та контроль над кодом гри;
- готові шаблони для різних жанрів ігор.

Недоліки:

- вимоги до апаратного забезпечення – Unreal Engine може бути вимогливим до апаратного забезпечення, що може вплинути на продуктивність розробки;
- навчання – через використання мови програмування C++ та складнішої архітектури, Unreal Engine може бути складнішим для вивчення для новачків.

Godot Незважаючи на те, що Godot займається розробкою ігор з 2014 року, движок останнім часом стає все більш популярним. Якщо шукати щось менш складне та з відкритим кодом, можна використовувати ігровий движок Godot і розробляти свої ігри так, як вам подобається [55].

На додаток до підтримки як 2D, так і 3D-графіки, двигун добре підходить для створення будь-яких ігор.

Крім того, у розробці ігор Godot використовується унікальний підхід до представлення конкретних функцій гри за допомогою архітектури вузлів і сцен, що робить її більш зручною для користувачів серед конкурентів. Можна отримати справжнього переможця, якщо об'єднати його з компанією з розробки відеоігор [55].

Мова сценаріїв, яку використовує Годо, називається GDScript, і вона має деякі недоліки. Оскільки мова була розроблена спеціально для розробки ігор Godot, вона схожа на Python. Тим не менш, досвідченим розробникам ігор може бути нудно вивчати ще одну мову. Оскільки Godot менш відомий, як Unity або Unreal [55].

Усі необхідні інструменти розробки є в Defold. Однак веб-розробники та розробники мобільних ігор можуть використовувати його для створення

ігор. Це твердження підтверджується вбудованими можливостями керування командою та контролю версій. Крім того, двигун пропонує спрощене рішення з інструментами, спеціально розробленими для 2D-ігор [55].

Механізм відеоігор Defold використовує мову програмування Lua для сценаріїв поведінки. Щоб забезпечити легке та безкоштовне створення мобільних ігор [55].

Всі ці двигуни об'єднує одне - вони безкоштовні і доступні для загального користування, але є нюанс. У безкоштовних версіях якщо розробник зі свого продукту отримує дохід вище 100 000 доларів США на рік, то він повинен купити Про версію, або корпоративну.

Безліч двигунів інших компаній залишаються закритими для сторонніх розробників, тому доступ до них мають самі працівники цих компаній. Наприклад це Rockstar Games з їх двигунів Rage у Sony з PhyreEngine.

Такі компанії через свій статус на ринку беруть малу кількість програмістів, з якими є угоди про нерозголошення та нерозповсюдження свого ПЗ стороннім особам. У разі порушень цих угод передбачаються покарання, починаючи від штрафів, закінчуючи кримінальною відповідальністю.

Сучасний ринок програмного забезпечення дозволяє купувати готові активи, не створюючи їх з нуля. Одним із найпопулярніших постачальників послуг є AssetStore від UnityTechnologies, та UnrealEngine Marketplace від Epic Games [55].

3.4.2 Порівняння та вибір мови програмування

Для програмування комп'ютерних ігор використовуються Java, C# та C++ – мови програмування. У таблиці 3.2 наведено детальне порівняння мов програмування для розробки розважальних застосунків.

C++ – мова програмування загального призначення. Плюси та мінуси мови програмування:

– він відрізняється різноманітністю і функціональністю, дозволяє виконувати операції з об'єктно-орієнтованого, процедурного та узагальненого програмування;

Таблиця 3.2 – Порівняння мов програмування

Назва	Java	C#	C++
Об'єктна орієнтація	Гібрид	Гібрид	Гібридна, Мулті
Спадкування	Один клас, декілька інтерфейсів	Один клас, декілька інтерфейсів	Декілька
Оператор перевантаження	Так	Так	Так
Методи перевантаження	Ні	Так	Так
Сміттезбірник	Позначити і змінити або Генераційний	Позначити і змінити або Генераційний	Немає
Класи, змінні, методи	Так	Так	Так
Рефлексія	Так	Так	Ні
Типи контроль доступу	public protected, “package” private	public protected, private internal, protected internal	public protected, private “ friends”
Багатопотоковість	Так	Так	Ні
Регулярні вирази	Стандартна бібліотека	Стандартна бібліотека	Ні
Вказівник арифметики	Ні	Так	Так
Мовний інтерфейс	C, C++	Усі .Net мови	C, Assembler

- основною перевагою C++ є велика стандартна бібліотека, яка включає такі функції, як введення/виведення та багатопотоковість, а також забезпечує можливість зручної алгоритмізації та контейнеризації;
- c++ однаково ефективний на різних платформах і успішно поєднується з іншими ігровими інструментами;
- вивчення C++ є складним процесом і вимагає від початківців програмістів великих зусиль [5647];

C# – це сучасна об'єктно-орієнтована мова програмування загального призначення, що вимовляється як "C sharp". Розроблена компанією Microsoft під керівництвом Андерса Хейлсберга (Anders Hejlsberg) та його команди в рамках ініціативи .Net і схвалена Європейською асоціацією виробників комп'ютерів (ECMA) та Міжнародною організацією зі стандартизації (ISO) [5759]. Особливості:

- мова популярна в геймплеї завдяки повній об'єктній орієнтованості;
- використовується для розробки ігрових продуктів на комп'ютері (включаючи .NET Framework);
- добре працює з механізмами та програмним забезпеченням графічного та звукового дизайну;
- він забезпечує підтримку поліморфізму, перезапуск оператора і дозволяє розробити всі необхідні елементи, такі як архітектура та логіка, для створення повноцінної гри [5859].

Java – це високорівнева, об'єктно-орієнтована мова програмування на основі класів, яка розроблена таким чином, щоб мати якомога менше залежностей при реалізації. Це універсальна мова програмування ["Write once, run anywhere?". Computer Weekly. May 2, 2002. Archived from the original on August 13, 2021. Retrieved July 27, 2009.]. Тобто скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності перекомпіляції.["1.2 Design Goals of the Java Programming Language". Oracle. January 1, 1999. Archived from the original on January 23, 2013. Retrieved

January 14, 2013.] Програми на Java зазвичай компілюються в байт-код, який може працювати на будь-якій віртуальній машині незалежно від базової архітектури комп'ютера. Синтаксис Java подібний до C та C++, але має менше низькорівневих засобів, ніж будь-яка з них [5858]. Плюси та мінуси :

- мова використовується для вивчення логіки та механіки мобільних ігор, особливо для продуктів Android;
- популярність цієї мови пояснюється його багатопотоковістю і можливістю безпроблемної взаємодії з пам'яттю мобільних пристроїв;
- він добре взаємодіє з графічними та аудіо двигунами та програмами;
- java часто реалізує серверну структуру для багатокористувацьких ігор.

3.5 Висновки до розділу 3

Популярність і спільнота: Unity є одним з найпопулярніших двигунів гри, має велику активну спільноту розробників, форуми, документацію та різноманітні посібники. Це означає, що можна легко знайти допомогу, питання і ресурси для розв'язання проблем та вивчення нових можливостей.

Мультиплатформеність: Unity дозволяє розробляти ігри для різних платформ, включаючи мобільні пристрої (iOS, Android), комп'ютери (Windows, macOS), ігрові консолі (PlayStation, Xbox) та веб-браузери. Це дає вам можливість досягати широкої аудиторії користувачів.

Графіка та візуалізація: Unity надає потужні можливості для розробки графіки в 2D і 3D. Він має вбудовані інструменти для створення ефектів, анімацій, освітлення, шейдерів та інших візуальних ефектів. Також можна використовувати сторонні інструменти, такі як Blender або Maya, для створення моделей і анімацій, і легко імпортувати їх у Unity.

Мова програмування: Unity підтримує C# як основну мову програмування. C# є потужною та ефективною мовою з великою кількістю

ресурсів та документації. Використання C# дозволяє створювати складну логіку гри та забезпечувати високу продуктивність.

Unity має великий маркетплейс Asset Store, можна знайти готові ресурси, які полегшують розробку, такі як моделі, текстури, аудіо, плагіни та інші. Це дозволяє прискорити процес розробки та знизити витрати на створення контенту.

Проаналізувавши ігрові движки, для проєкту було обране Unity Engine так як він є більш функціональним та підходить найкраще для створення мобільних ігор.

4 КОНСТРУЮВАННЯ ПРОЄКТУ

4.1 Схеми алгоритму функціонування застосунку

Схему алгоритму функціонування застосунку наведено на рис. 4.1.

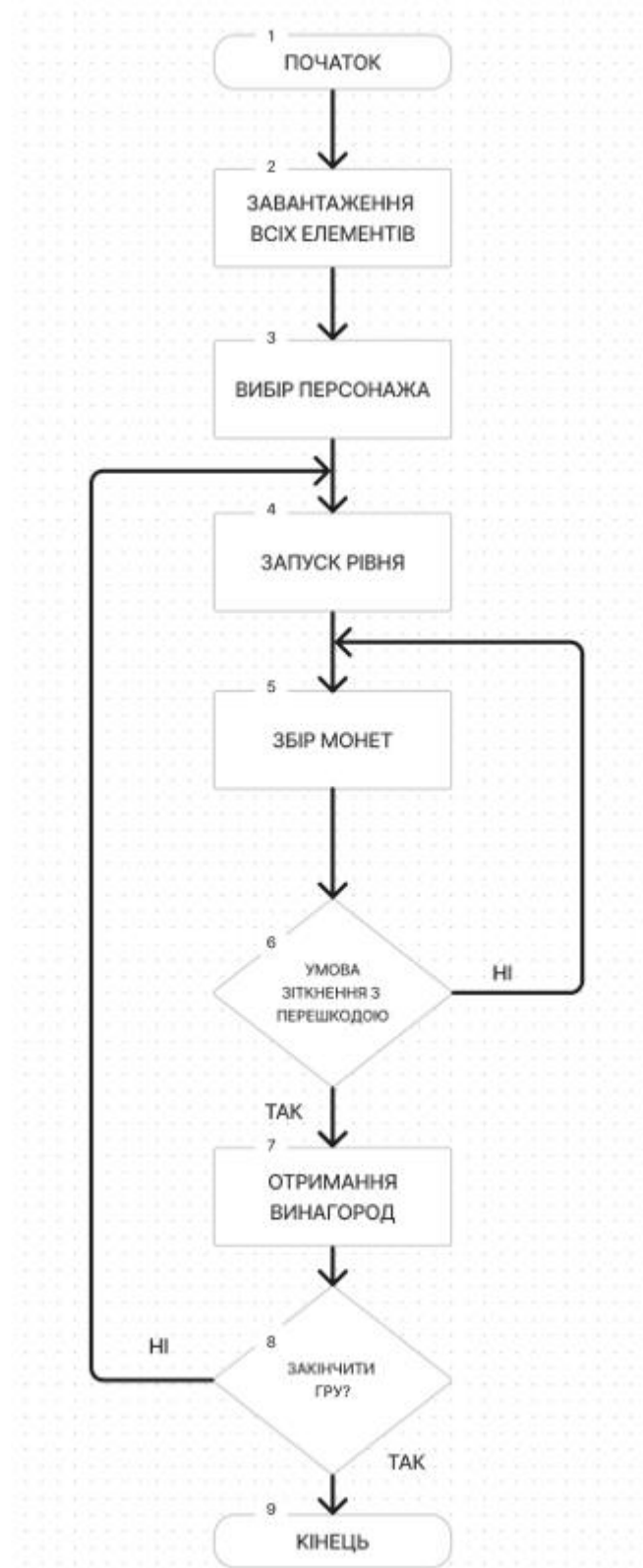


Рисунок 4.1 – Схеми алгоритму

Блок 1. Початок роботи застосунку.

Блок 2. Загрузка всіх візуальних елементів, скриптів, сцен тощо.

Блок 3. Вікно вибору вигляду головного персонажу.

Блок 4. Запуск сцени з основним геймплеєм.

Блок 5. Головний герой біжить по дорозі, та збирає монети водночас усуваючи різні перешкоди.

Блок 6. Умова, якщо герой зіткнувся з перешкодою.

Блок 7. Якщо персонаж зіткнувся з перешкодою, то з'являється меню з отриманням винагород та відображення рекодру.

Блок 8. За умовою, користувач обирає, закінчити гру, або продовжити.

Блок 9. Вихід з гри.

4.2 Створення компонентів

4.2.1 Візуальні 3D

Усі графічні 3D об'єкти імпортовані у форматі .fbx та .obj, створені у графічному редакторі Blender.

FBX і OBJ – це формати файлів, які використовуються для обміну та зберігання 3D-моделей.

3D моделі повинні бути оптимізовані і використовувати мінімальну кількість полігонів ~ 150000 одиниць на всі об'єкти, включаючи сцену для коректної роботи програми.

LOD (Level of Detail) – це техніка, що використовується в комп'ютерних іграх для оптимізації відображення об'єктів залежно від їхньої відстані до камери. Ідея полягає в тому, щоб замінити деталізовану версію об'єкта на менш деталізовану версію, коли об'єкт знаходиться далеко від камери [5942].

Принцип роботи LOD полягає в створенні кількох версій моделі об'єкта з різними рівнями деталізації (LOD-рівнями). LOD дозволяє покращити продуктивність гри, зменшити навантаження на обчислювальні

ресурси і підвищити швидкість відображення. Віддалені об'єкти, які мають меншу важливість для гравця, можуть мати значно меншу кількість полігонів або використовувати спрощені текстури, що дозволяє заощадити ресурси [5942].

Для проєкту було створено 3D об'єкти з різним рівнем LOD для гри, що представлено на рис. 4.2.

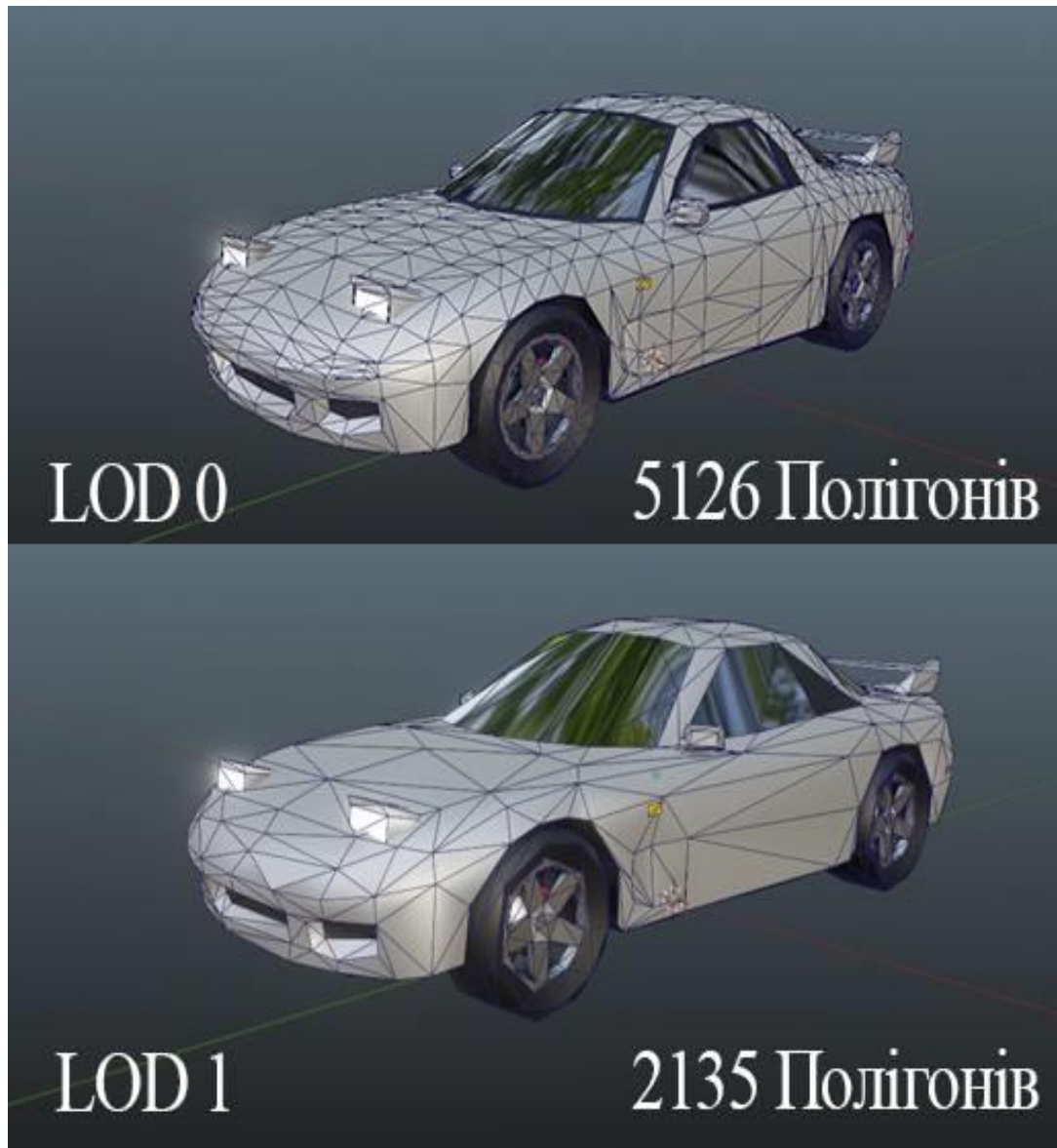


Рисунок 4.2 – Приклад LOD в проєкті

Для гри була створена основна сцена (локація), на яких розміщені динамічні та статичні об'єкти.

До динамічних відносяться: персонаж (Головний Герой), перешкоди (машини), основна валюта як бонуси на карті. Моделі персонажу представленої гри потребується від 10000 до 20000 полігонів.

До статичних відносяться сама сцена без об'єктів, які з'являються по скрипту. Зображення прикладу платформи представлені на рис. 4.3, зображення розмірів 3D об'єктів представлено на рис. 4.4.



Рисунок 4.3 – Приклад платформи



Рисунок 4.4 – Розміри 3D об'єктів

4.2.2 Створення візуальних компонентів інтерфейсу

У цьому розділі дипломної роботи розглядається розробка інтерфейсу для мобільної гри, з особливим акцентом на важливість UI (користувацького інтерфейсу).

Всі зображення імпортовані у форматі .png. Розміри зображень повинні мати один за таких розмірів: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 пікселів.

Аналіз матеріалу: Для створення якісного інтерфейсу було проведено детальний аналіз релевантного матеріалу, включаючи стандартні шаблони та рекомендації для UI-дизайну. Було вивчено і надихнутося різними джерелами для розробки естетично привабливого та зручного інтерфейсу.

Імпорт з Asset Store: Для полегшення процесу розробки та отримання готових ресурсів, було використано Asset Store – маркетплейс Unity, де доступні готові елементи UI, шрифти, іконки та інші ресурси. Завантаженням та імпортом цих ресурсів з Asset Store ми забезпечили швидку та зручну роботу з елементами інтерфейсу.

Розташування елементів: Відповідно до розглянутих шаблонів та вимог проєкту, всі елементи інтерфейсу були розміщені в середовищі Unity. Застосовано принципи та правила UI-дизайну, такі як зручність розташування, читабельність тексту, наочність кнопок та інтерактивних елементів.

Кодування кнопок та анімацій: Для реалізації функціональності кнопок та анімацій в інтерфейсі був написаний відповідний код. Цей код був наведений в Додатку А.

Зображення інтерфейсу головного меню з різними розширеннями екрану представлено на рис. 4.5.

На головному меню є 5 кнопок:

- кнопка запуску гри (зелена кнопка);



Рисунок 4.5 – Інтерфейс головного меню

- кнопка щоденних нагород (блакитна кнопка з іконкою подарунок);
- кнопка кастомізації персонажа (блакитна кнопка з іконкою профілю);
- кнопка щоденних завдань (блакитна кнопка з іконкою календаря);
- кнопка магазину (блакитна кнопка з іконкою прилавка).

4.3 Компоновка об'єктів на сцені

У проєкті реалізована ігрова ієрархія (послідовність) сцен, а саме: вступ з анімацією – головне меню – ігрова сцена. Усі ці сцени мають переходи шляхом завантаження. Зображення ієрархії сцен представлено на рис. 4.6.

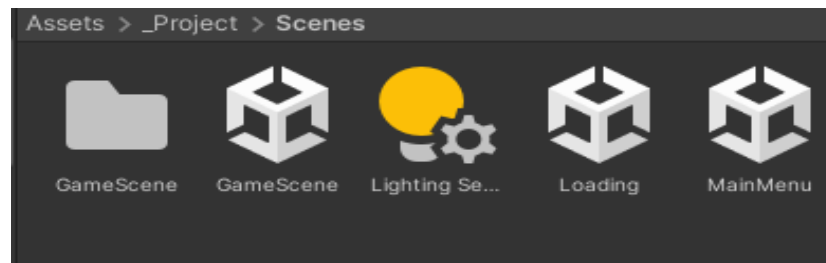


Рисунок 4.6 – Ієрархія сцен

Сцени загрузки є важливою складовою багатьох ігор і виконують багато функцій. Ось декілька причин, чому сцени загрузки потрібні в іграх:

Завантаження ресурсів: Під час переходу між різними рівнями або сценами гри потрібно завантажити нові ресурси, такі як моделі персонажів, текстури, звукові ефекти тощо. Сцена загрузки дає можливість виконати цей процес без переривання гри, дозволяючи гравцеві зосередитися на екрані завантаження.

Візуальна показівка: Сцена загрузки може слугувати візуальним елементом, який тримає гравця зайнятим під час завантаження. Вона може включати анімації, ілюстрації, прогрес-бари або інші ефекти, які роблять процес загрузки більш захоплюючим і відволікають увагу від очікування.

Завантаження гри: сцена загрузки також може служити для ініціалізації гри, налаштуванні, завантаження збережених даних гравця або проведення будь-яких інших процесів, необхідних для правильного початку гри. Це дозволяє зберігати ресурси і зменшувати час завантаження, дозволяючи гравцеві плавно переміщатися між різними областями гри.

В цілому, сцени загрузки важливі в іграх, оскільки вони забезпечують плавний перехід між різними частинами гри, показують прогрес

завантаження, створюють настрій і забезпечують оптимальне завантаження ресурсів для безперебійного ігрового досвіду. Зображення ієрархії сцени загрузки представлено на рис 4.7.

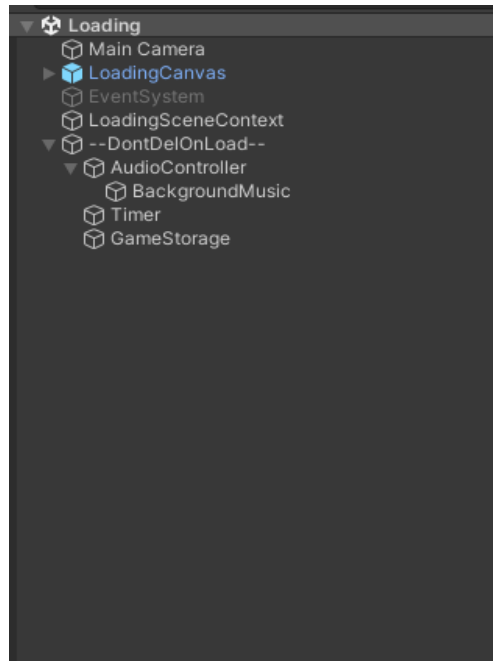


Рисунок 4.7 – Ієрархія сцени завантаження

Сцени загрузки забезпечують плавний перехід між різними частинами гри, показують прогрес завантаження, створюють настрій і забезпечують оптимальне завантаження ресурсів для безперебійного ігрового досвіду.

Головна камера (Main Camera): Створення та налаштування основної камери, яка використовуватиметься на сцені.

Канвас завантаження (LoadingCanvas): Створення канвасу, який міститиме візуальний інтерфейс завантаження, такий як прогрес-бари, текстові індикатори і т.д.

LoadingSceneContex: Створення завантажувальної сцени, де буде реалізовано візуальний інтерфейс завантаження та весь контекст, пов'язаний із процесом завантаження.

Скрипт, що реалізує логіку завантажувальної сцени: Створення та програмування скрипта, який керуватиме всією логікою та прогресом

завантаження. Цей скрипт може містити функції завантаження ресурсів, оновлення прогресу та перехід до наступної сцени після завершення завантаження.

DontDelOnLoad: Створення об'єктів, які залишаються активними протягом усіх сцен. Ці об'єкти переносяться на інші сцени після завершення завантаження.

Переходи між сценами відбуваються завдяки скрипту **SceneController.cs**

Зображення ієрахії сцени головного меню представлено на рис. 4.8.

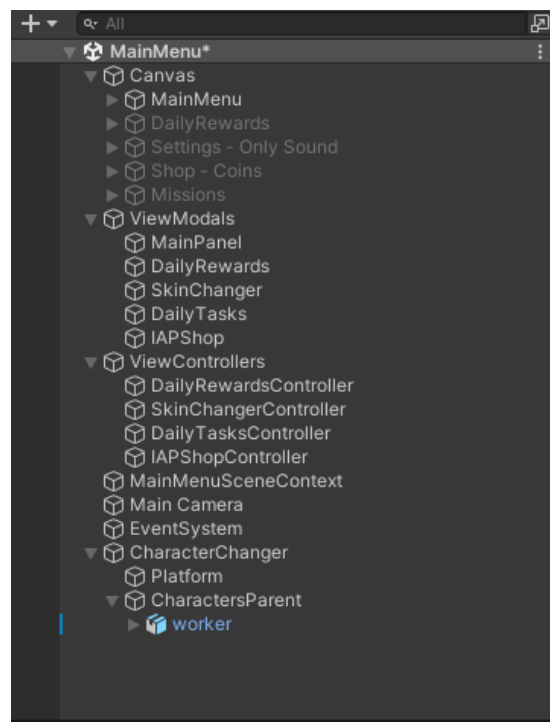


Рисунок 4.8 – Ієрахія сцени головного меню

На головній сцені розташовані всі об'єкти які тільки керують логікою головного меню та не виходять за межі (приклад зображений на рис. 4.8).

Вся логіка головного меню є у відображенні візуалу та надання кнопкам потрібних дій, наприклад як перехід на ігрову сцену та відкриття вікон для кастомізації та отримання різних Нагород.

На ігровій сцені відбуватиметься весь основний геймплей, де головний герой біжить та перестрибує крізь перешкоди, збираючи монети.

Якщо персонаж зіштовхнеться з перешкодою, то гра буде закінчена. На цій сцені знаходяться лише скрипти, які відповідають за всю ігрову логіку.

- генерація платформ;
- генерація і запуск трафіка;
- генерація монет;
- ініціалізація обраного користувачем героя.

4.4 Розробка скриптів і нанесення їх на об'єкти

4.4.1 Генерація локації

На сцені ініціалізуються і додаються до списку всі різновиди платформ, у той кількості яку вказує розробник, щоб об'єкти переставляти, а не видаляти і заново створювати їх на сцені. Це зроблено для оптимізації, оскільки проєкт створений для мобільних пристроїв. На рис. 4.9 зображено сцену гри у редакторі, де можна розглянути систему генерування рівня. Генерування всіх платформ реалізує скрипт «MapSpawner.cs», який наведено в Додатку А.

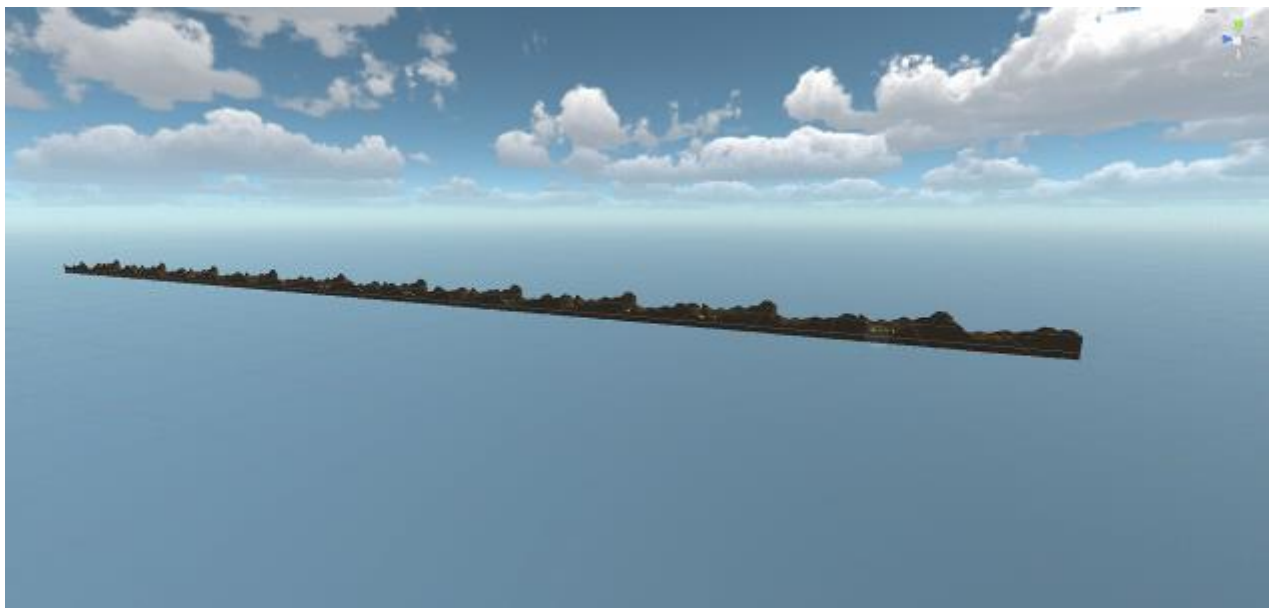


Рисунок 4.9 – Вигляд згенерованих платформ

4.4.2 Генерація трафіка (перешкод)

На сцені відбувається ініціалізація кількох автомобілів, які потім додаються до пулу. Це дозволяє оптимізувати проєкт та ефективно керувати трафіком. Автомобілі рухаються вперед дорогою та між смугами. Кожному типу машини можна вказати частоту появи.

Основним елементом керування трафіком є скрипт "HWTrafficSpawner.cs". У цьому скрипті зберігаються інформація про смуги, автомобілі та потрібні параметри. Кожен автомобіль рухається зі своєю швидкістю і може переходити з однієї смуги на іншу без пригод, оскільки автомобіль кожен кадр визначає, що довкола нього.

Крім того, автомобілі оснащені лампами, які відображають необхідні сигнали на дорозі, такі як сигнали поворотів або зупинки.

Все вищеприписане ілюстровано рис. 4.10, який представляє схематичне зображення системи трафіку у проєкті.

Автомобіль вважається перешкодою, якщо з ним зіткнеться персонаж, то гра буде окінчена.

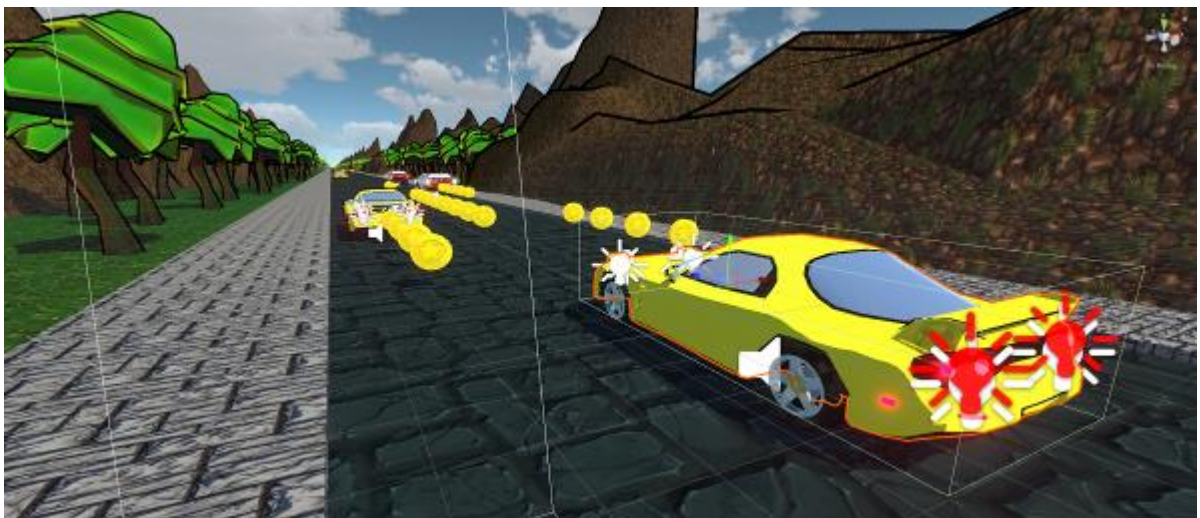


Рисунок 4.10 – Схематичне зображення системи трафіку у проєкті

4.4.3 Створення персонажа в Unity

Для того щоб можливо було керувати персонажем в грі, було розроблено декілька скриптів, та нанесення найважливіших компонентів на об'єкт, тобто:

- прикріплення компонента "Transform" до персонажа, який дозволяє керувати позицією, поворотом та масштабом персонажа;
- прикріплення компонента "Rigidbody" для фізики, який дозволяє реалізувати фізичну взаємодію з навколишнім середовищем, таке як гравітація, колізії та сили;
- створення та прикріплення скрипта "PlayerMovement" для керування персонажем, який містить логіку керування персонажем, таку як пересування, стрибки, взаємодія з об'єктами, виклик анімацій тощо;
- додавання коллайдера, який визначає його фізичну форму та виявлення зіткнень з іншими об'єктами у сцені;
- прикріплення компонента "CharacterController" для зручного управління персонажем, логіка управління зі скрипта "PlayerMovement" до скрипта "CharacterController", щоб забезпечити зручніше та модульніше управління персонажем;
- прикріплення компонента "Animator Contorller", для повного керування анімаціями.

Зображення списку компонентів персонажу представлено на рис. 4.11.

На 3D об'єкті персонажа прикріплений аніматор, який керує кістками об'єкта та переміщає їх відповідно до анімації, що викликається в коді.

Коли персонаж готовий, а введення налаштоване, потрібно настроїти програвання анімації. Animation Controller – це інструмент Unity, у якому описуються визначені стани персонажа. В даному випадку бігти, стрибати чи померти.

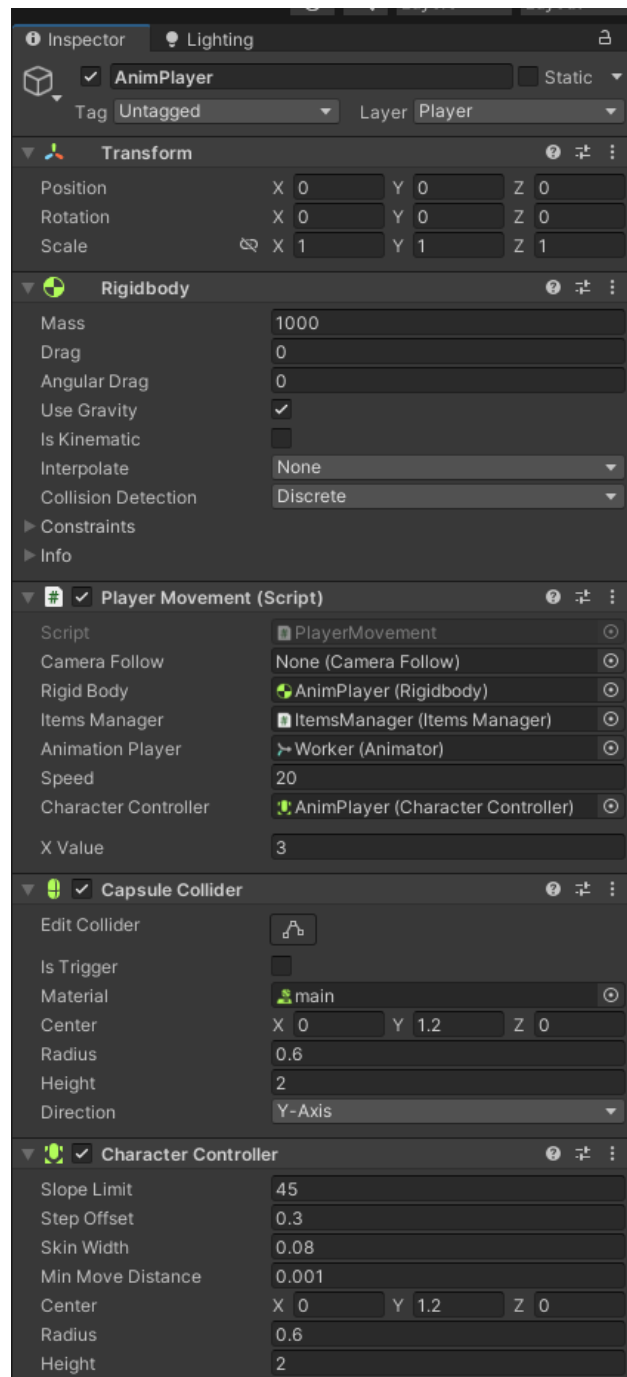


Рисунок 4.11 – Список компонентів персонажу

Кожному блоку відповідає свій анімаційний кліп та налаштовуються графі переходів, що зображені на рис. 4.12.

Коли персонаж повністю створений і введення налаштовано, необхідно налаштувати відтворення анімації. У Unity використовується Animation Controller, який є інструментом визначення різних станів персонажа. В даному випадку стану можуть включати біг, стрибок, поворот

наліво і направо. Кожному стану відповідає свій анімаційний кліп, який визначає конкретну анімацію цього стану.

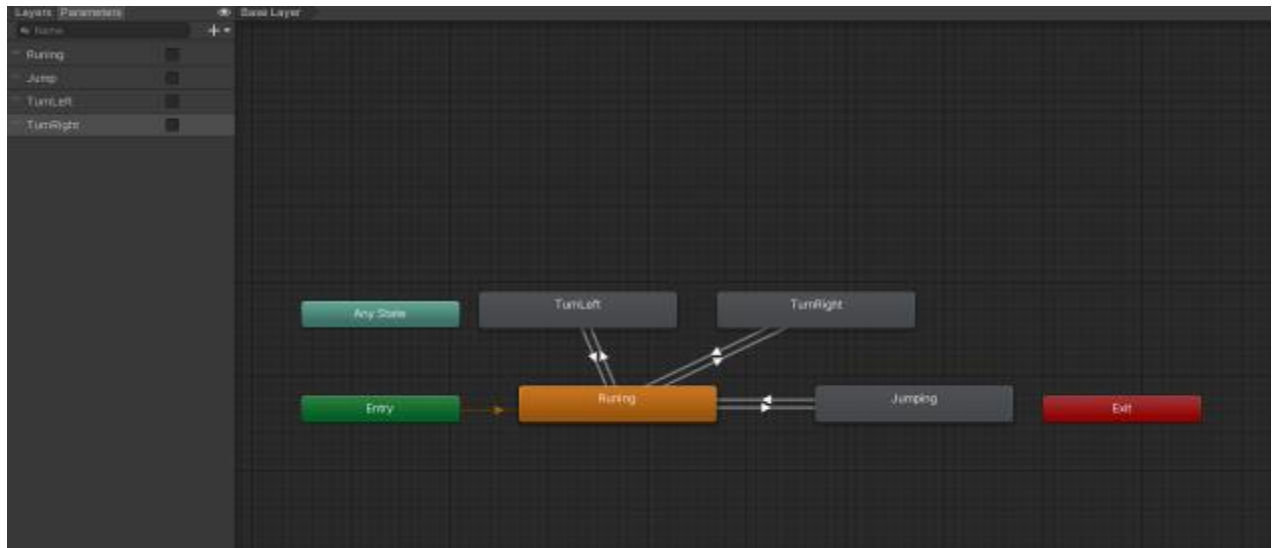


Рисунок 4.12 – Вікно animation controller

Animation Controller надає графічний інтерфейс, де можна настроїти графи переходів між різними станами. Граф переходів (як показано на рис. 4.12) дозволяє визначити, які умови мають бути виконані, щоб персонаж перейшов із одного стану до іншого. При виконанні умови бігу персонаж перейде в стан "Runing", і відповідна анімація програватиметься.

4.4.4 Створення механіки управління

Основною механікою управління персонажем для гравця є – свайпи, тобто проведення пальцем у напрямку руху персонажа: вгору, вниз, вправо, вліво. За рух персонажа відповідає скрипт "SwipeController.cs". У скрипті отримуємо торкання по екрану по осях X і Y та за потрібною умовою вже визначаємо напрямок свайпа.

Основною механікою, що використовується гравцем, є свайпи-проведення пальцем у певному напрямку для керування рухом персонажа. Для обробки таких свайпів використовується скрипт "SwipeController.cs". У

цьому скрипті ми отримуємо дані про торкання екрана вздовж осей X та Y, і на основі певних умов визначаємо напрямок свайпа. Наприклад, якщо користувач проводить палець, скрипт розпізнає це як свайп вгору. Аналогічно, свайпи вниз, праворуч або ліворуч також обробляються відповідним чином.

Це дозволяє гравцеві контролювати рух персонажа шляхом проведення свайпів у потрібному напрямку, що забезпечує інтерактивне та інтуїтивно зрозуміле управління у грі.

4.5 Програмна реалізація

В рамках розробки проєктів у середовищі Unity, процес створення APK-файлу це етап побудови проєкту. Для здійснення цієї процедури необхідно виконати наступні кроки.

Налаштування проєкту перед здійсненням побудови. Це включає в себе вибір платформи Android, встановлення належних параметрів, версії Android SDK, а також забезпечення належної конфігурації відповідних налаштувань, як зображено на рис. 4.13.

Побудова проєкту, після налаштування проєкту треба перейти до фази побудови (рис. 4.14). В цьому кроці Unity здійснює компіляцію проєкту та створення APK-файлу. Процес побудови включає компіляцію програмного коду, включення ресурсів (таких як текстури, звуки, моделі) і пакування їх у формат APK.

Після успішного завершення побудови APK-файлу, треба протестувати на мобільному пристрої. Це дозволяє виявити можливі проблеми та помилки та внести необхідні зміни для досягнення оптимальної функціональності та продуктивності додатку.

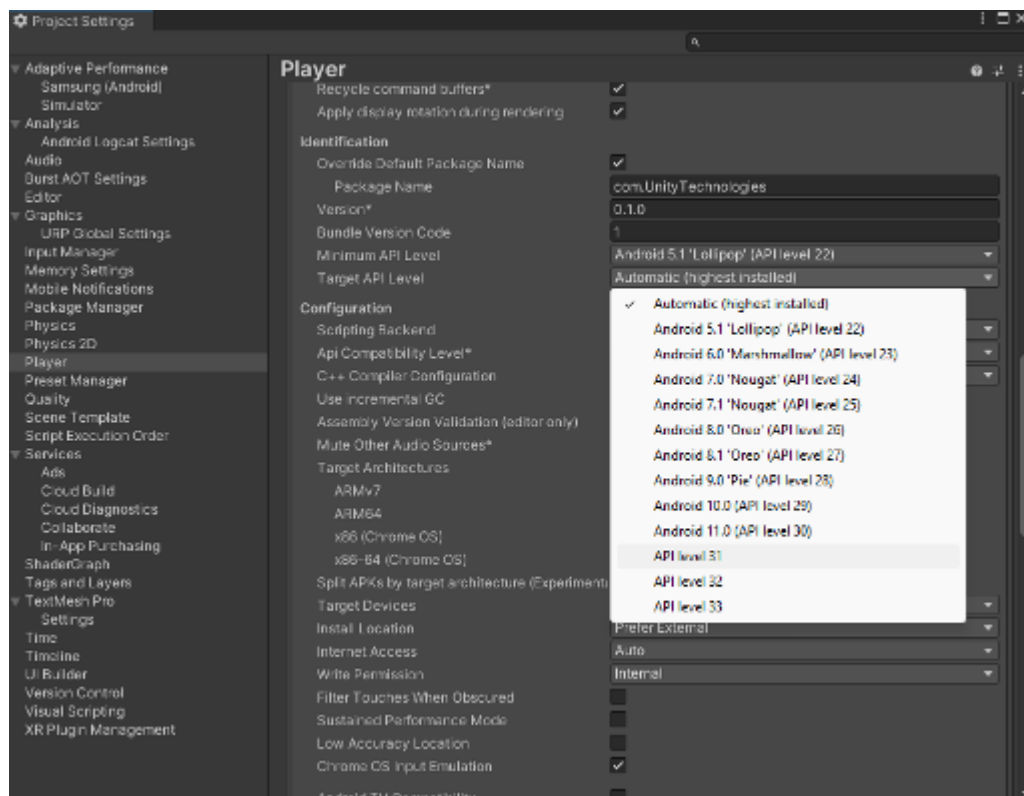


Рисунок 4.13 – Вікно налаштувань проєкту

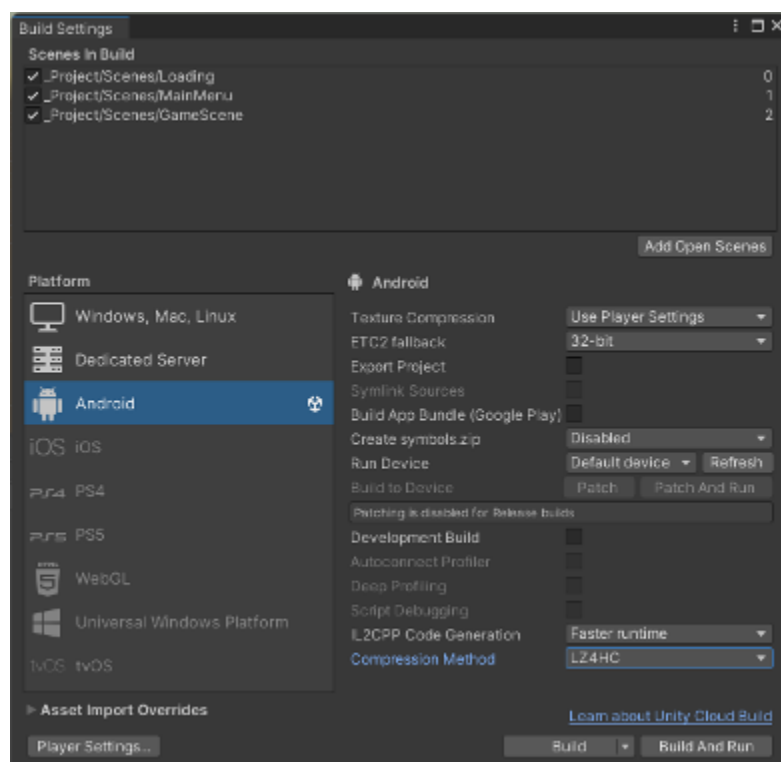


Рисунок 4.14 – Вікно побудови проєкту

4.6 Висновки до розділу 4

В цьому розділі була надана детальна інформація про створення візуальних компонентів. Описано процес додавання та компоновка їх на сцені, розробка скриптів.

Здійснення процесу створення білда, який буде придатний для використання на мобільних пристроях. Це означає збірку проєкту в .apk форматі, який можна встановити на мобільний пристрій з операційною системою Android.

5 КЕРІВНИЦТВО ПРОГРАМІСТА

5.1 Розгортання проєкту

Проект Unity знаходиться в архівному файлі під назвою «Highway_Runner_Project.zip».

Файл можна надати за допомогою флешки, або скачати з GitLab.

Щоб запустити проєкт, потрібно скопіювати на ПК вміст архіву та додати путь проєкта в Unity Hub.

Проект Unity знаходиться у стислому архіві під назвою "Highway_Runner_Project.zip". Щоб отримати доступ до файлів проєкту, можна скопіювати їх з флешки або завантажити з GitLab.

Для запуску проєкту потрібно розпакувати вміст архіву на комп'ютері. Після цього відкрити Unity Hub і додати шлях до розпакованих файлів проєкту. Unity Hub автоматично виявить проєкт і дозволить відкрити його для редагування та запуску.

Для успішного запуску проєкту можуть знадобитися додаткові ресурси або плагіни, які використовуються у проєкті.

5.2 Структура проєкта

Структура проєкту, що включає директорії та файли, представлена на рисунку 5.1, що зображує структуру директорій проєкту. Основною директорією, де є всі файли проєкту, є Assets. У цій директорії є пакети, які можуть бути встановлені через Package Manager або вручну, скрипти, сцени, префаби, зображення, 3D-моделі, плагіни та інші файли, необхідні для проєкту.

Директорія _Project є однією з основних у проєкті. Усі файли, специфічні для даного проєкту, мають бути розміщені у цій директорії. Скрипти, що знаходяться в директорії GameRunner, не можуть викликатися іншими скриптами, розташованими поза цією папкою. Взаємодія може

здійснюватись лише у напрямку від GameRunner до MainScripts, а не навпаки. Зображення діаграми дерикторій проєкта представлено на рис. 5.1.

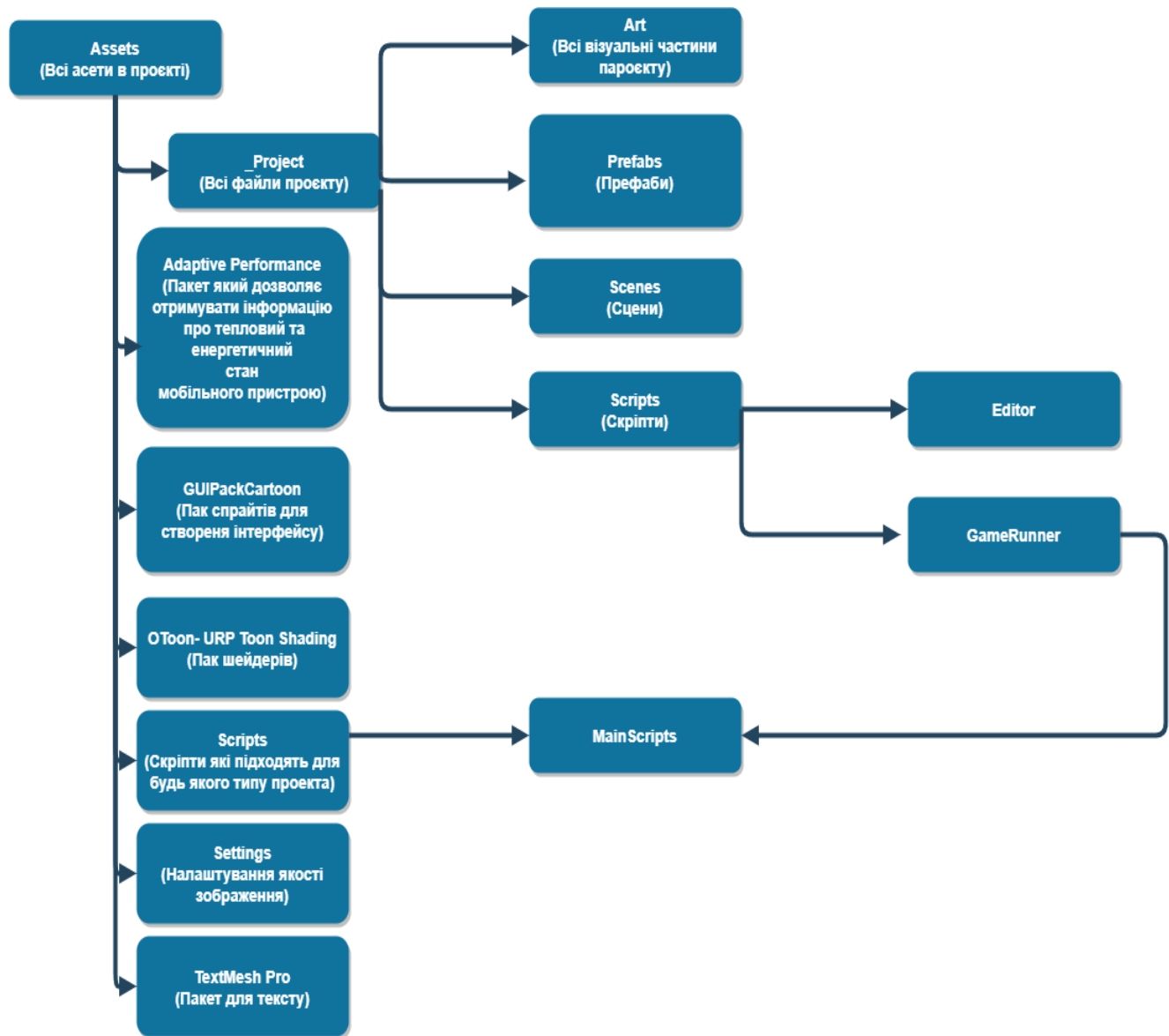


Рисунок 5.1 – Діаграма директорій проєкта

5.3 Додавання персонажа

Для включення додаткової моделі персонажу в існуючий проєкт Unity, необхідно здійснити процес імпорту 3D моделі у форматі .fbx до відповідної директорії "Assets/_Project/Art/Models/Peds/Назва_нового_персонажу".

Важливо забезпечити, щоб модель мала такий самий скелет (тобто ієрархічну структуру кісток) як і основна модель, зразок цього наведено на рисунку 5.2.

Після успішного імпортування моделі, необхідно відкрити префаб героя, який використовується в проєкті. У цьому префабі потрібно додати нову 3D модель, розмістити її в потрібному положенні та, при необхідності, змінити масштаб для досягнення бажаного ефекту. Крім цього, необхідно призначити Animation Controller для відповідного керування анімаціями нової моделі.

Animation Controller є важливим компонентом, що відповідає за керування анімаціями персонажу в грі. Його завдання полягає у визначенні правильного відтворення анімацій залежно від дій гравця або інших важливих факторів в грі. Тому необхідно належним чином налаштувати Animation Controller для нової моделі персонажу, щоб забезпечити необхідні анімаційні переходи та відтворення.

Після внесення всіх необхідних змін до префабу та моделі персонажу, важливо зберегти всі зміни, щоб вони були належним чином застосовані у проєкті Unity. Зображення скелету пересонажа представлено на рис. 5.2.



Рисунок 5.2 – Скелет персонажа

5.4 Додавання платформи

Для того, щоб додати додаткову модель платформи до проєкту, необхідно виконати кілька кроків. Початок з імпорту моделі в проєкт. Треба імпортувати модель у форматі .obj або .fbx в директорію «Assets/_Project/Art/Models/MAPS».

Після імпортування моделі, можна перенести об'єкт платформи на сцену вашої гри. Для того, щоб забезпечити фізичну взаємодію з іншими об'єктами на сцені, необхідно створити коллайдер. Залежно від форми вашої моделі, де можна використовувати Box Collider або Mesh Collider.

Після додавання коллайдера до моделі платформи, можна зберегти цей об'єкт як префаб. Префаб – це готовий шаблон об'єкта, який можна повторно використовувати в інших частинах вашої гри.

Останній крок – забезпечити спавн цього префабу на ігровій сцені. Для цього потрібно внести зміни в скрипт «MapSpawner.cs». Треба додати посилання на префаб платформи в цей скрипт, щоб він міг створювати нові примірники цього об'єкта під час гри.

5.5 Збірка проєкту

Після внесення всіх змін у проєкт, необхідно здійснити процес збирання в .apk файл та протестувати його на мобільному пристрої з операційною системою Android. Треба, щоб у процесі збирання не виникло жодних помилок, а всі функції гри працювали належним чином.

Після успішного протестування та перевірки, можна розглянути можливість публікації гри в магазині Play Market. Це дозволить поділитися проєктом зі світом та надати доступ до нього для користувачів Android-пристроїв. Для публікації в магазині Play Market також знадобиться створити обліковий запис розробника та відповідати всім вимогам та політикам, встановленим Google.

Збірка гри в .apk файл, успішне тестування та подальша публікація в магазині Play Market дозволять досягти більшої аудиторії та поділитися творчістю з іншими користувачами мобільних пристроїв.

5.6 Висновки до розділу 5

В даному розділі була надана детальна інформація про архітектуру проєкту та умови його експлуатації. Описано процес додавання нових моделей до проєкту, а також налаштування змін у проєкті. Крім того, розглянуто процедуру створення білда, який придатний для використання на мобільних пристроях.

Після внесення всіх необхідних змін у проєкт, важливо здійснити процес створення білда, який буде придатний для використання на мобільних пристроях. Це означає збірку проєкту в .apk форматі, який можна встановити на мобільний пристрій з операційною системою Android. Після створення білда рекомендується провести тестування на мобільному пристрої, щоб переконатися, що всі функції працюють належним чином та відсутні будь-які помилки.

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Призначення мобільного застосунку

Завданням розважального мобільного застосунку є створення забавного та захоплюючого досвіду для користувача, шляхом керування персонажем, який уникатиме перешкод, збираючи бонуси та виконуючи різноманітні завдання. Гра пропонує широкий спектр рівнів, локацій та персонажів, що дозволяє користувачам насолоджуватись безліччю ігрових можливостей.

Для ефективного використання мобільної гри необхідний смартфон або планшет з операційною системою Android. Також вимагається стабільне з'єднання з мережею Інтернет, яке забезпечує доступ до оновлень гри, та онлайн-функцій.

Основною програмною складовою гри є сама мобільна програма, яка включає в себе графічний інтерфейс користувача, анімацію, звукові ефекти та механіку геймплею. Гра підтримується інтегрованими бібліотеками та інструментами розробки, що забезпечують її оптимальну функціональність та сумісність з різними пристроями.

Загалом, мобільна гра призначена для забезпечення розважального досвіду користувачів, шляхом надання їм можливості керувати персонажем у віртуальному середовищі.

6.2 Запуск гри

Для запуску гри, потрібно її інсталиувати на мобільний пристрій.

Далі після того, як був здійснений запуск гри, користувач потрапляє на головне меню гри, де розташовані кнопки, серед яких користувачу потрібно обрати саме таку, яка йому потрібна. Головне меню гри продемонстровано на рис. 6.1.



Рисунок 6.1 – Головне меню гри

На головному мені присутньо 5 кнопок:

- кнопка під номером 1 відціне ігрову сцену, та почнеться головний геймплей;
- кнопка під номером 2 відціняє панель щоденних винагород (рис. 6.2);
- кнопка під номером 3 перенесе на панель зміни вигляду персонажа;
- кнопка під номером 4 відціняє панель щоденних завдань;
- кнопка під номером 5 відціняє панель магазину монет, де можна за реальні гроші купити монети.



Рисунок 6.2 – Панель щоденних винагород

Ігрова сцена, є ключовим елементом геймплею, який надає користувачеві можливість інтерактивного досвіду та взаємодії з віртуальним середовищем, приклад зображений на рис. 6.3.

На цій ігровій сцені користувач бере участь у захоплюючому перегоні по шосе, що характеризуються жвавим рухом, різнобарвними графічними елементами та великою кількістю деталей. Пейзажі представлені у формі тривимірних об'єктів, які включають гори, тунелі, дорогу та дерева.

Графічне візуальне оформлення сцени відрізняється яскравими кольорами та використанням високоякісних текстур, що створює реалістичне відчуття присутності у віртуальному світі. Крім того, анімація персонажів та

навколишнього середовища додає динаміки та руху до сцени, створюючи враження швидкості та напруженості.

Геймплей на ігровій сцені вимагає активної взаємодії користувача з пристроєм управління, таким як сенсорний екран смартфона чи планшета. Користувач має можливість керувати персонажем, переміщуючи його вправо, вліво, та вгору, з метою уникнення перешкод, збирання бонусів та досягнення як найбільшої відстані.



Рисунок 6.3 – Приклад ігрової сцени

6.3 Висновки до розділу 6

В даному розділі було розглянуто процес взаємодії користувача з мобільною грою. Були описані способи інсталювання.

Інтерфейс розробленого мобільного додатку відповідає типовим вимогам та очікуванням користувача. Він був створений з урахуванням зручності та легкості використання, максимізуючи його основну функціональність.

Загалом, інтерфейс розробленого мобільного додатку для гри є інтуїтивно зрозумілим для будь-якого користувача. Він надає можливість легко та ефективно користуватись усіма функціями гри, забезпечуючи задоволення та комфорт користувачів.

ВИСНОВКИ

Проведений глибокий аналіз індустрії ігор та її напрямів підтверджує, що мобільні ігри будуть продовжувати займати важливу позицію на ринку індустрії розваг. Враховуючи зростання кількості користувачів мобільних пристроїв та постійне удосконалення технологій, такі ігри стають все більш популярними.

Процес розробки гри був детально розглянутий у даній дипломній роботі, зокрема створення інтерфейсу, фізики об'єктів та анімації графічних елементів. Особливу увагу було приділено технології ігрових двигунів, яка надає значні покращення якості та вдосконалення, сприяє створенню нових захоплюючих ігрових світів з високоякісною графікою та фотореалістичними ефектами.

Для розробки розважального мобільного застосунка було обрано мову C# та ігровий рушій Unity 3D, а для створення 3D об'єктів - графічний редактор Blender. Використання середовища розробки Unity 3D виявилось надійним та зручним засобом реалізації мобільного застосунку завдяки його функціоналу.

Розроблений мобільний застосунок-гра дозволяє відображати інформацію про кількість зібраних монет та найкращий результат користувача.

Мобільний застосунок працює стабільно, без необхідності внесення змін в проєкт. Він має зручний та зрозумілий інтерфейс, а також забезпечує коректність даних при вводі та виводі.

Архітектура проєкта була розроблена з можливістю для майбутніх змін, а саме додавання нових персонажів, карт, бонусів, тощо. Ці змінні збільшать кількість контенту, що у результаті привабить нових користувачів.

Результати роботи були представлені в тезах доповіді НПК «Тиждень науки-2023».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Farnsworth, B. Science of Video Games [Electronic resource] / B. Farnsworth – Access mode: <https://imotions.com/blog/insights/research-insights/science-video-games/>.
2. Fandom action video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Action_video_games.
3. Fandom adventure video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Adventure_video_games.
4. Fandom role-playing video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Role-playing_video_games.
5. Fandom shooter video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Shooter_video_games.
6. Fandom strategy video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Strategy_video_games.
7. Wikipedia contributors. Survival game. [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Survival_game/.
8. Fandom Racing video games [Electronic resource]. – Access mode: https://gamicus.fandom.com/wiki/Racing_video_games.
9. Wikipedia contributors. Simulation Video Game [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Simulation_video_game.
10. Old Games Download. Shrek 2 [Electronic resource]. – Access mode: <https://oldgamesdownload.com/shrek-2>.
11. Blizzard News. Ende der Aktion: WoW-Beute für Mitglieder von Prime Gaming [Electronic resource]. – Access mode: <https://news.blizzard.com/de-de/world-of-warcraft/23922954/ende-der-aktion-wow-beute-fur-mitglieder-von-prime-gaming-holt-euch-den-grossen-kampfbaren>.
12. Polygon Valorant update nerfs Bucky and Guardian, buffs Viper and Breach [Electronic resource]. – Access mode:

<https://www.polygon.com/2020/3/3/21162047/valorant-riot-updates-balance-patches-new-characters-maps>.

13. PCMag. Warcraft III: Reforged (for PC) - Review [Electronic resource]. – Access mode: <https://www.pcmag.com/reviews/warcraft-iii-reforged-for-pc>.

14. PC Gamer. DayZ review. [Electronic resource]. – Access mode: <https://www.pcgamer.com/dayz-review>.

15. Steam. I Am Bread [Electronic resource]. – Access mode: https://store.steampowered.com/app/327890/I_Am_Bread/?l=german.

16. Laptop Mag. PS5 vs. Nintendo Switch vs. Xbox Series X vs. PC: Which Gives You the Best Bang for Your Buck? [Electronic resource]. – Access mode: <https://www.laptopmag.com/features/ps5-vs-nintendo-switch-vs-xbox-series-x-vs-pc-which-gives-you-the-best-bang-for-your-buck>.

17. Indie Game Culture. Best Indie Survival Games [Electronic resource]. – Access mode: <https://indiegameculture.com/listicles/best-indie-survival-games>.

18. Polygon. Terraria House Building Guide: Stairs, Walls, Blocks, Doors Tutorial [Electronic resource]. – Access mode: <https://www.polygon.com/terraria-guide/22360700/house-build-stairs-walls-blocks-doors-tutorial>.

19. Wikipedia contributors. Virtual Reality Headset: [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Virtual_reality_headset#History.

20. Virtual Reality Society. What is VR Gaming? [Electronic resource]. – Access mode: <https://www.vrs.org.uk/virtual-reality-games/what-is-vr-gaming.html>.

21. Queppelin. AR Glasses for Navigation [Electronic resource]. – Access mode: <https://www.queppelin.com/ar-glasses-for-navigation>

22. Mashable. Samsung Gear VR Headset [Electronic resource]. – Access mode: <https://mashable.com/deals/samsung-gear-vr-headset>.

23. Wikipedia contributors. Microvision [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Microvision>.

24. EasyTechJunkie. What is a Game Boy? [Electronic resource]. – Access mode: <https://www.easytechjunkie.com/what-is-a-game-boy.htm>.
25. Twitter. RetroTechInfo: Microvision Handheld Photo [Electronic resource]. – Access mode: <https://twitter.com/retrotechinfo/status/861943109332107268/photo/1>.
26. History of Computer. The PlayStation Vita: History, Launch, and Ultimate Failure Explained [Electronic resource]. – Access mode: <https://history-computer.com/the-playstation-vita-history-launch-and-ultimate-failure-explained/>.
27. Towell, J Console features no one uses (But Should) [Electronic resource] /J. Towell. – Access mode: <https://www.gamesradar.com/console-features-no-one-uses>.
28. Nintendo. Nintendo Switch [Electronic resource]. – Access mode: <https://www.nintendo.de/Hardware/Nintendo-Switch-Familie/Nintendo-Switch/Nintendo-Switch-1148779.html>.
29. Nintendo Store UK [Electronic resource]. – Access mode: <https://nintendostoreuk.queue-it.net/>.
30. Johnes,R. Steam Deck review [Electronic resource] / R. Johnes. – Access mode: <https://www.trustedreviews.com/reviews/steam-deck>.
31. Evangelho, J. Steam Deck desktop mode challenge: day 1 [Electronic resource] / J. Evangelho. – Access mode: <https://www.forbes.com/sites/jasonevangelho/2022/03/04/steam-deck-desktop-mode-challenge-day-1-get-a-hub-and-get-docked/>.
32. PC Games Hardware. Docking Station mit Upgrade-Anschluss vor Release [Electronic resource]. – Access mode: <https://www.pcgameshardware.de/Steam-Deck-Hardware-277668/News/Dockingstation-mit-Upgrade-nch-vor-Release-1393601>.
33. Adir, C. Positive effects of video games [Electronic resource] /C. Adir. – Access mode: <https://gamequitters.com/positive-effects-of-video-games>.

34. Google Play Store game of mafia [Electronic resource]. – Access mode:

<https://play.google.com/store/apps/details?id=com.yottagames.gameofmafia>.

35. PhoneArena Fortnite: iOS, Android, Console (PS4, Xbox), PC differences comparison [Electronic resource]. – Access mode: https://www.phonearena.com/news/fortnite-ios-android-console-ps4-xbox-pc-differences-comparison_id103461.

36. Wikipedia contributors. Cross-platform play [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Cross-platform_play.

37. Wikipedia contributors. Incremental games [Electronic resource]. – Access mode: https://en.m.wikipedia.org/wiki/Incremental_game

38. Wikipedia contributors. Tetris [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Tetris>.

39. Bleier, T. Tetris: the highest scsoring game in classic tetris world championship [Electronic resource] / T. Bleier. – Access mode: <https://gamerant.com/tetris-world-championship-highest-scoring-game-classic-tetris>.

40. Audrey. Subway Surfers vs Temple Run [Electronic resource] / Audey. – Access mode: <https://subwayapk.com/subway-surfers-vs-temple-run-how-they-differ>.

41. Adobe: Flash Player Statistics [Electronic resource]. – Access mode: http://www.adobe.com/products/player_census/flashplayer/.

42. Education-wiki. Що таке JavaFX [Електрон. ресурс] / Education Wiki. – Режим доступу: <https://uk.education-wiki.com/9288574-what-is-javafx>.

43. 3Ddroid програма для створення флеш-ігор [Електрон. ресурс]. – Режим доступу: <https://3ddroid.ru/browsers/the-program-for-creating-flash-games-create-flash-animations/>.

44. Rokoko Insights motion capture in video games [Electronic resource]. – Access mode: <https://www.rokoko.com/insights/motion-capture-in-video-games>.

45. CGI School. Програми для створення 3D-графіки [Електрон. ресурс] / CGI School. – Режим доступу: <https://cgischool.ua/programy-dlia-stvorennia-3d-grafiky>.
46. All3DP – Blender 3D Printing Tutorial [Electronic resource]. – Access mode: <https://all3dp.com/2/blender-3d-printing-tutorial>.
47. Blender [Electronic resource]. – Access mode: <https://www.blender.org/>.
48. PCMag Encyclopedia. 3ds Max [Electronic resource] / PCMag Encyclopedia. – Access mode: <https://www.pcmag.com/encyclopedia/term/3ds-max>.
49. Autodesk 3ds Max [Electronic resource]. – Access mode: <https://www.autodesk.com/products/3ds-max/>.
50. All3DP. What is 3ds Max? Simply Explained [Electronic resource] / All3DP. – Access mode: <https://all3dp.com/2/what-is-3ds-max-simply-explained>.
51. Definitions.net. Autodesk Maya [Electronic resource]. – Access mode: <https://www.definitions.net/definition/autodesk+maya>.
52. Alienbrain. Version Control for Maya [Electronic resource]. – Access mode: <https://www.alienbrain.com/version-control-for-maya.php>.
53. Concept Art Empire. - What is ZBrush? [Electronic resource] / Concept Art Empire. – Access mode: <https://conceptartempire.com/what-is-zbrush>.
54. Documentation. Pixologic ZBrushCore. Interface and navigation [Electronic resource]. – Access mode: <http://docs.pixologic.com/zbrushcore/interface-navigation>.
55. Game Engines [Electronic resource]. – Access mode: <https://theninehertz.com/blog/game-engines>.
56. Klein, O. Modern C++ Design: Generic Programming and Design Patterns Applied / O. Klein. – Heidelberg, 2020. – 28 p.
57. GeeksforGeeks - Introduction to C# [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-c-sharp>.

58. Wikipedia contributors. Java programming language [Electronic resource] / Wikipedia contributors. – Access mode: [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)).

59. CGspectrum. What is Level of Detail (LOD) in 3D Modeling? [Electronic resource] / CGspectrum. – Access mode: <https://www.cgspectrum.com/blog/what-is-level-of-detail-lod-3d-modeling>.

ДОДАТОК А
Текст програми

A.1 ScenesController.cs – скрипт переходу між сценами

```
namespace GameRunner.Scenes
{
    public class ScenesController : MonoBehaviour
    {
        [SerializeField] private Canvas _canvas;
        [SerializeField] private LoadingFade _loadingFade;

        public void LoadScene(EScene scene)
        {
            _canvas.enabled = true;
            StartCoroutine(LoadSceneAsyncSingle(scene));
        }

        private IEnumerator LoadSceneAsyncSingle(EScene scene)
        {
            var operation = SceneManager.LoadSceneAsync(scene.ToString(),
LoadSceneMode.Single);

            while (operation.progress < 0.95)
            {
                yield return null;

                _loadingFade.DisableFadeAndDestroy();
                StopAllCoroutines();
            }

            while (!operation.isDone)
            {
                yield return null;
            }
        }
    }
}
```

A.2 MapSpawner.cs – скрипт генерації рівня

```
namespace GameRunner.Pooling
{
    public class MapSpawner : MonoBehaviour
    {
        [System.Serializable]
        public class RoadObjects
        {
            public GameObject roadObject;
        }

        [SerializeField] private bool _automaticRoadLength = true;
        [Header("Use This Layer On Road For Calculating Road Length")]
```

```

[SerializeField] private LayerMask _asphaltLayer;
[Header("Pooling Road Objects. Select Them While They Are On Your Scene")]
[SerializeField] private RoadObjects[] _roadObjects;

[SerializeField] private float _manualRoadLength = 60f;

[SerializeField] private float _roadWidth = 13.5f;

private Transform _reference;
private bool _isAnimateNow = true;
private int _index = 0;
private float[] _roadLength;

private int _roadAmountInPool = 10;

internal List<GameObject> Roads = new List<GameObject>();

private void Awake()
{
    _reference = Camera.main.transform;
    _roadLength = new float[_roadObjects.Length];

    for (int i = 0; i < _roadObjects.Length; i++)
    {
        if (_automaticRoadLength)
            _roadLength[i] = GetRoadLength(i);
        else
            _roadLength[i] = _manualRoadLength;
    }

    CreateRoads();
}

protected float GetRoadLength(int road_index)
{
    GameObject roadReference = Instantiate(_roadObjects[road_index].roadObject,
    Vector3.zero, Quaternion.identity);
    Bounds combinedBounds =
roadReference.GetComponentInChildren<Renderer>().bounds;
    Renderer[] renderers =
roadReference.GetComponentInChildren<Renderer>();

    foreach (Renderer render in renderers)
    {
        if (render != roadReference.GetComponent<Renderer>() && 1 <<
render.gameObject.layer == _asphaltLayer)
            combinedBounds.Encapsulate(render.bounds);
    }

    Destroy(roadReference);
    return combinedBounds.size.z;
}

```

```

    }

    private void CreateRoads()
    {
        GameObject allRoads = new GameObject("All Roads");
        allRoads.transform.position = Vector3.zero;
        allRoads.transform.rotation = Quaternion.identity;

        for (int i = 0; i < _roadAmountInPool; i++)
        {
            for (int k = 0; k < _roadObjects.Length; k++)
            {
                GameObject go = Instantiate(_roadObjects[k].roadObject,
                    _roadObjects[k].roadObject.transform.position,
                    _roadObjects[k].roadObject.transform.rotation);

                go.isStatic = true;
                Roads.Add(go);

                go.transform.SetParent(allRoads.transform);
            }
        }

        for (int i = 0; i < Roads.Count; i++)
        {
            if (i != 0)
                Roads[i].transform.position =
                    new Vector3(0f, Roads[i].transform.position.y, Roads[i]
1].transform.position.z + _roadLength[_index <= 0]
                    ? _roadObjects.Length - 1 : _index - 1]);

            _index++;

            if (_index >= _roadObjects.Length)
                _index = 0;
        }
        _index = 0;
    }

    private void Update()
    {
        if (_isAnimateNow)
            AnimateRoad();
    }

    private void AnimateRoad()
    {
        for (int i = 0; i < Roads.Count; i++)
        {
            if (_reference.transform.position.z > (Roads[i].transform.position.z +
(_roadLength[_index] * 2f)))

```

```

        {
            Roads[i].transform.position = new Vector3(0f,
Roads[i].transform.position.y, (Roads[i].transform.position.z + (_roadLength[_index] *
Roads.Count)));
        }

        _index++;

        if (_index >= _roadObjects.Length)
            _index = 0;
    }
}

#if UNITY_EDITOR
private void OnDrawGizmos()
{
    Gizmos.color = new Color(0f, 1f, 0f, .75f);
    Gizmos.DrawCube(Vector3.zero, new Vector3(_roadWidth * 3f, 1f, 10f));

    Gizmos.DrawCube(Vector3.zero, new Vector3(1f, 1f, _manualRoadLength *
3f));
}
#endif
}
}

```

A.3 SwipeController.cs – скрипт свайпів

```

namespace GameRunner.Swipe
{
    public class SwipeController : MonoBehaviour
    {
        //Constants
        private const float fSensitivity = 15f;

        //VARIABLES
        //distance calculation
        private float fInitialX;
        private float fInitialY;
        private float fFinalX;
        private float fFinalY;

        private float inputX;
        private float inputY; //y-coordinate
        private float slope; //slope (m) of the the
        private float fDistance; //magnitude of distance between two positions
        private int iTouchStateFlag; //flag to check
        public SwipeDirection sSwipeDirection; //string to receive swipe output

        void Start()
        {
            fInitialX = 0.0f;

```

```

fInitialY = 0.0f;
fFinalX = 0.0f;
fFinalY = 0.0f;
inputX = 0.0f;
inputY = 0.0f;
iTouchStateFlag = 0;
sSwipeDirection = SwipeDirection.Null;
}

void Update()
{
    if (PlayerMovement.isplay)
    {
        if (PlayerMovement.SpeedMoving > 10)
        {
            if (Input.mousePosition.y < Screen.height - Screen.height / 8 &&
                Input.mousePosition.y > Screen.height - Screen.height)
            {
                if (iTouchStateFlag == 0 && Input.GetMouseButtonDown(0))
                {
                    fInitialX = Input.mousePosition.x;
                    fInitialY = Input.mousePosition.y;

                    sSwipeDirection = SwipeDirection.Null;
                    iTouchStateFlag = 1;
                }
                if (iTouchStateFlag == 1)
                {
                    fFinalX = Input.mousePosition.x;
                    fFinalY = Input.mousePosition.y;

                    sSwipeDirection = swipeDirection();
                    if (sSwipeDirection != SwipeDirection.Null)
                        iTouchStateFlag = 2;
                }
                if (iTouchStateFlag == 2 || Input.GetMouseButtonUp(0))
                {
                    iTouchStateFlag = 0;
                }
            }
        }

        switch (getSwipeDirection())
        {
            case SwipeDirection.Null:
                break;
            case SwipeDirection.Jump:
                PlayerMovement.player.Jump();
                break;
            case SwipeDirection.Right:
                PlayerMovement.player.TurnRight();
                break;
        }
    }
}

```

```

        case SwipeDirection.Left:
            PlayerMovement.player.TurnLeft();
            break;
        default:
            break;
    }
}
}
}
/*
 *      FUNCTION: Return swipe direction.
 *      RETURNS: Returns NULL if no swipes are detected.
 *              Returns SwipeDirection if a swipe is detected
 */
public SwipeDirection getSwipeDirection()
{
    if (sSwipeDirection != SwipeDirection.Null)//if a swipe is detected
    {
        var etempSwipeDirection = sSwipeDirection;
        sSwipeDirection = SwipeDirection.Null;

        return etempSwipeDirection;
    }
    else
        return SwipeDirection.Null;//if no swipe was detected
}

/*
 *      FUNCTION: Calculate the swipe direction
 */
private SwipeDirection swipeDirection()
{
    //calculate the slope of the swipe
    inputX = fFinalX - fInitialX;
    inputY = fFinalY - fInitialY;
    slope = inputY / inputX;
    //calculate the distance of tap start and end
    fDistance = Mathf.Sqrt(Mathf.Pow((fFinalY - fInitialY), 2) +
    Mathf.Pow((fFinalX - fInitialX), 2));

    //print(inputX);
    //print(inputY);

    if (fDistance <= (Screen.width / fSensitivity))//higher the dividing factor higher
the sensitivity
        return SwipeDirection.Null;

    if (inputX >= 0 && inputY > 0 && slope > 1)//first octant JUMP
    {
        Debug.Log("Jump");

        return SwipeDirection.Jump;
    }
}

```

```

    }
    else if (inputX <= 0 && inputY > 0 && slope < -1)//eighth octant JUMP
    {
        Debug.Log("Jump");
        return SwipeDirection.Jump;
    }
    else if (inputX > 0 && inputY >= 0 && slope < 1 && slope >= 0)//second
octant RIGHT
    {
        Debug.Log("Right");
        return SwipeDirection.Right;
    }
    else if (inputX > 0 && inputY <= 0 && slope > -1 && slope <= 0)//third octant
RIGHT
    {
        Debug.Log("Right");
        return SwipeDirection.Right;
    }
    else if (inputX < 0 && inputY >= 0 && slope > -1 && slope <= 0)//seventh
octant LEFT
    {
        Debug.Log("Left");
        return SwipeDirection.Left;
    }
    else if (inputX < 0 && inputY <= 0 && slope >= 0 && slope < 1)//sixth octant
LEFT
    {
        Debug.Log("Left");
        return SwipeDirection.Left;
    }

    return SwipeDirection.Null;
}
}
}

```

A.4 HWTrafficSpawner.cs – скрипт генерації трафіку

```

namespace GameRunner.Pooling
{
    public class HWTrafficSpawner : MonoBehaviour
    {
        private Transform _reference;

        public Transform[] Lines;

        public LayerMask TrafficCarsLayer;

        public TrafficCars[] TrafficCars;

        private List<HWTrafficCar> _trafficCars = new List<HWTrafficCar>();
    }
}

```

```

private bool animateNow
{
    get
    {
        return true;
    }
}

private void Start()
{
    _reference = Camera.main.transform;
    CreateTraffic();
}

private void Update()
{
    if (animateNow)
        AnimateTraffic();
}

private void CreateTraffic()
{
    for (int i = 0; i < TrafficCars.Length; i++)
    {
        for (int k = 0; k < TrafficCars[i].frequence; k++)
        {
            var car = Instantiate(TrafficCars[i].trafficCar, Vector3.zero,
Quaternion.identity);
            car.Init(TrafficCarsLayer, this);
            _trafficCars.Add(car);
            car.gameObject.SetActive(false);
        }
    }
}

private void AnimateTraffic()
{
    for (int i = 0; i < _trafficCars.Count; i++)
    {
        if (_reference.transform.position.z > (_trafficCars[i].transform.position.z +
15) || _reference.transform.position.z < (_trafficCars[i].transform.position.z - (325)))
            ReAlignTraffic(_trafficCars[i]);
    }
}

private void ReAlignTraffic(HWTrafficCar realignableObject)
{
    if (!realignableObject.gameObject.activeSelf)

```

```

        realignableObject.gameObject.SetActive(true);

        int randomLine = Random.Range(0, Lines.Length);

        realignableObject.currentLine = randomLine;
        realignableObject.transform.position = new
Vector3(Lines[randomLine].position.x, Lines[randomLine].position.y,
(_reference.transform.position.z + (Random.Range(100, 300))));

        realignableObject.SendMessage("OnReAligned");

        if (CheckIfClipping(realignableObject.triggerCollider))
            realignableObject.gameObject.SetActive(false);
    }

    bool CheckIfClipping(BoxCollider trafficCarBound)
    {
        for (int i = 0; i < _trafficCars.Count; i++)
        {
            if (!trafficCarBound.transform.IsChildOf(_trafficCars[i].transform) &&
                _trafficCars[i].gameObject.activeSelf)
            {
                if (BoundsExtension.ContainBounds(trafficCarBound.transform,
                trafficCarBound.bounds, _trafficCars[i].triggerCollider.bounds))
                    return true;
            }
        }

        return false;
    }
}

```

A.5 PlayerMovement.cs – скрипт руху персонажа

```

namespace GameRunner.Player.Movement
{
    public class PlayerMovement : MonoBehaviour
    {

        public static PlayerMovement player;

        [SerializeField] private CameraFollow _cameraFollow; // OLD

        [SerializeField] private Rigidbody _rigidBody;
        [SerializeField] protected ItemsManager _itemsManager;
        [SerializeField] private Animator _animationPlayer;
        [SerializeField] private float _speed = 15;
        [SerializeField] private CharacterController _characterController;
    }
}

```

```

[Space]
[SerializeField] private float _xValue = 3f;
private CapsuleCollider _capsuleCollider;
private bool _alowjump; // turn to release 1 shot

private ESide _side = ESide.Mid;

private float _NexXPos;

private bool _swipeLeft;
private bool _swipeRight;

private float _score;
private float _boostScore = 1;

public static float SpeedMoving => player._speed;

public static bool isplay = true;

public static uint Score => (uint)player._score;
public static int Coins;

private void OnValidate()
{
    _cameraFollow = FindObjectOfType<CameraFollow>();
    _itemsManager = FindObjectOfType<ItemsManager>();

    _rigidBody = GetComponent<Rigidbody>();
    _capsuleCollider = GetComponent<CapsuleCollider>();

    _characterController = GetComponent<CharacterController>();
}

private void Start()
{
    _side = ESide.Mid;
    transform.position = Vector3.zero;

    _animationPlayer.Play("Runing");
}

private void Update()
{
    if (!isplay) return;

    _characterController.Move((_speed * Time.deltaTime) * Vector3.forward);

    _swipeLeft = Input.GetKeyDown(KeyCode.A) ||
Input.GetKeyDown(KeyCode.LeftArrow);

```

```

        _swipeRight = Input.GetKeyDown(KeyCode.D) ||
Input.GetKeyDown(KeyCode.RightArrow);

        if (Input.GetKeyDown(KeyCode.Space) ||
Input.GetKeyDown(KeyCode.UpArrow)) Jump();

        if (_swipeLeft)
        {
            TurnLeft();
        }
        else if (_swipeRight)
        {
            TurnRight();
        }

        _characterController.Move((_NexXPos - transform.position.x) * Vector3.right);

        _score += Time.deltaTime * _boostScore;
    }

    public void TurnLeft()
    {
        if (_side == ESide.Mid)
        {
            _animationPlayer.SetBool("TurnLeft", true);

            StartCoroutine(LerpLeftRight(ESide.Left, transform.position.x, -_xValue));
        }
        else if (_side == ESide.Right)
        {
            TurnMid();
        }
    }

    public void TurnRight()
    {
        if (_side == ESide.Mid)
        {
            _animationPlayer.SetBool("TurnRight", true);

            StartCoroutine(LerpLeftRight(ESide.Right, transform.position.x, _xValue));
        }
        else if (_side == ESide.Left)
        {
            TurnMid();
        }
    }

    private float _lerpDuration = 0.3f;
    private float _valueToLerp;

```

```

private IEnumerator LerpLeftRight(ESide goToSide, float startValue, float
goToPos)
{
    float timeElapsed = 0;
    while (timeElapsed < _lerpDuration)
    {
        _valueToLerp = Mathf.Lerp(startValue, goToPos, timeElapsed /
_lerpDuration);
        timeElapsed += Time.deltaTime;

        _NexXPos = _valueToLerp;//goToSide == ESide.Left ? _valueToLerp * -1 :
_valueToLerp;

        yield return null;
    }

    _valueToLerp = goToPos;
    _side = goToSide;
}

public void TurnMid()
{
    StartCoroutine(LerpLeftRight(ESide.Mid, transform.position.x, 0));
}

public void Jump()
{
    _animationPlayer.SetBool("Jump", true);
}

private void Awake()
{
    Coins = 55;

    player = this;
}
}
}

```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Титульний слайд

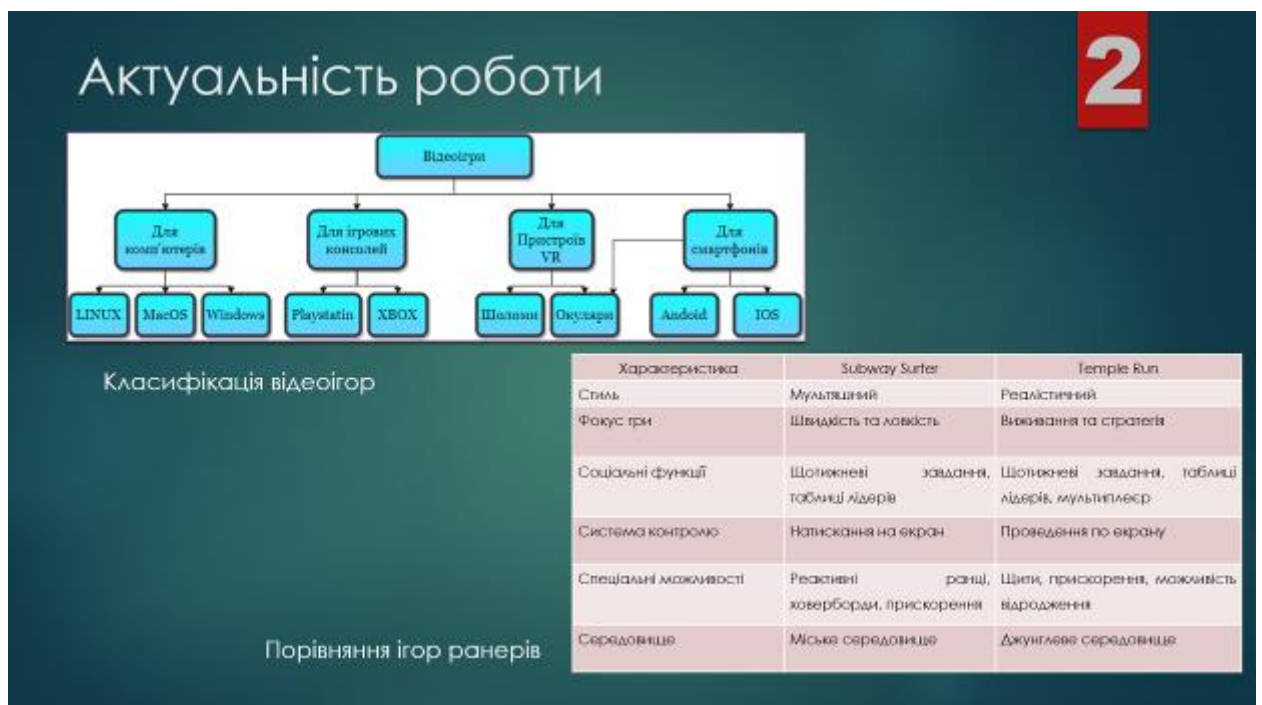


Рисунок Б.2 – Актуальність роботи

3 Мета та завдання роботи

Мета роботи – розробка програмного забезпечення розважального мобільного застосунку для організації дозвілля дітей та дорослих.

Завдання роботи:

- дослідити методику розробки мобільної гри;
- виконати аналіз вимог та розробити архітектуру системи;
- проаналізувати та обрати інструментарій розробки;
- виконати програмну реалізацію та тестування програми.

Рисунок Б.3 – Мета та завдання роботи



Рисунок Б.4 – Проєктування ПЗ

5

Проектування ПЗ

Таблиця порівняння ігрових рушіїв

Двигун	Реліз	Найпопулярніші ігри	Ціна	ОС
Unity	2005	Genshin Impact	Безкоштовно	Win 7+, MacOS
UE4	1998	Fortnite	Безкоштовно	Win 7+, MacOS
Godot	2014	Cruelty squad	Безкоштовно	Win 7+, MacOS
GameMaker Studio	2013	Bayou Island	Безкоштовно	Win 7+
Armory	2020	Armory 3D	Безкоштовно	Win 7+
CryEngine	2002	Crysis	Безкоштовно	Win 7+, MacOS
Defold	2002	Bring me cakes	Безкоштовно	Win 7+, MacOS

Таблиця порівняння мов програмування

мова	Java	C#	C++
Об'єктна орієнтація	Гібрид	Гібрид	Гібрид, Мульти
Спадування	Один клас, декілька інтерфейсів	Один клас, декілька інтерфейсів	Декілька
Оператор перевантаження	Так	Так	Так
Методи перевантаження	Ні	Так	Так
Смисловий	Позначити і зміняти або Генерацийний	Позначити і зміняти або Генерацийний	Немає
Класи, методи	Так	Так	Так
Рефлексив	Так	Так	Ні
Типи контролю доступу	public, protected, "package", private	public, protected, private, internal, protected internal	public, protected, private, "friends"
Багатопотоковість	Так	Так	Ні
Регулярні вирази	Стандартна бібліотека	Стандартна бібліотека	Ні
Ікодинг арифметики	Ні	Так	Так
Мовний інтерфейс	C, C++	Ucl, .Net, more	C, Assembler

Рисунок Б.5 – Проектування ПЗ



Рисунок Б.6 – Конструювання ПЗ



Рисунок Б.7 – Результат розробки

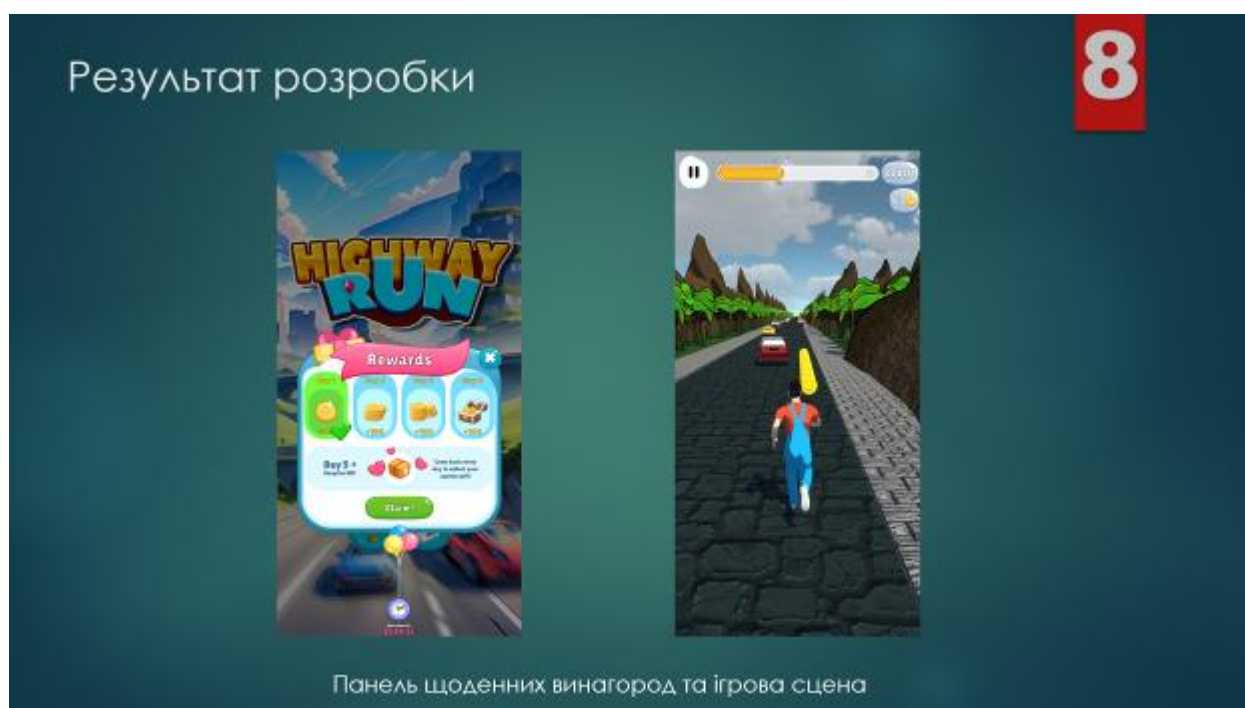


Рисунок Б.8 – Результат розробки

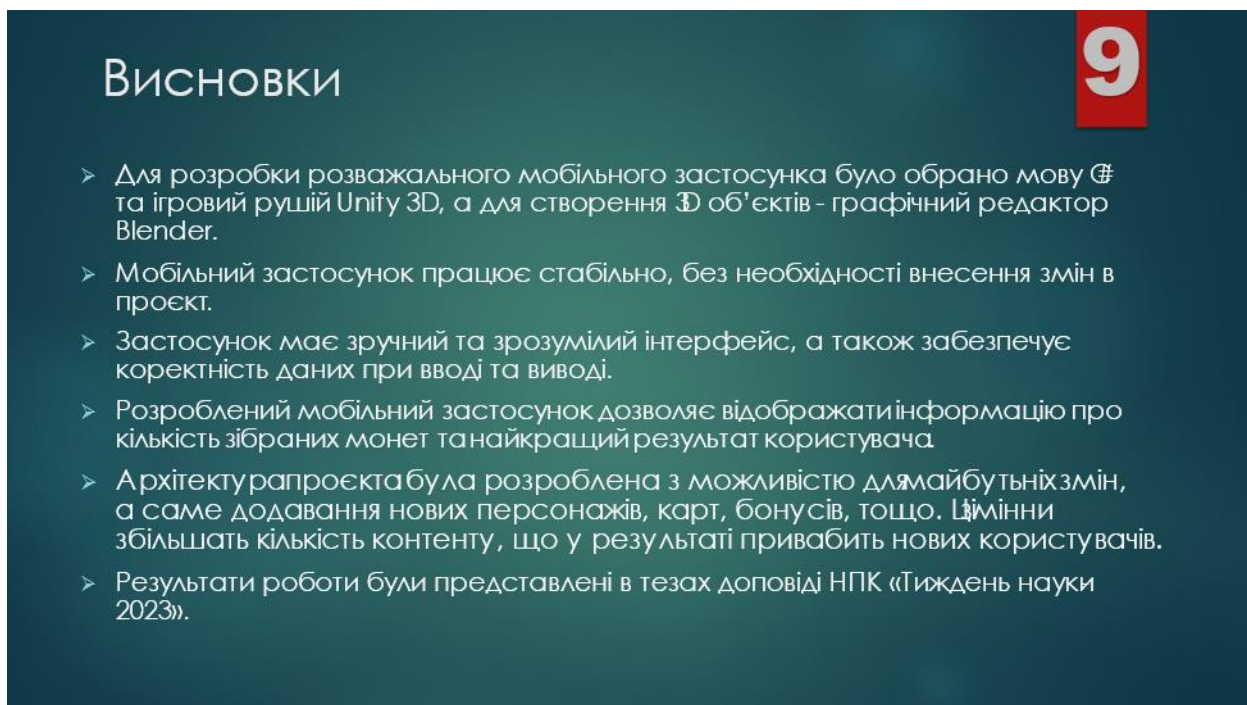


Рисунок Б.9 – Висновки

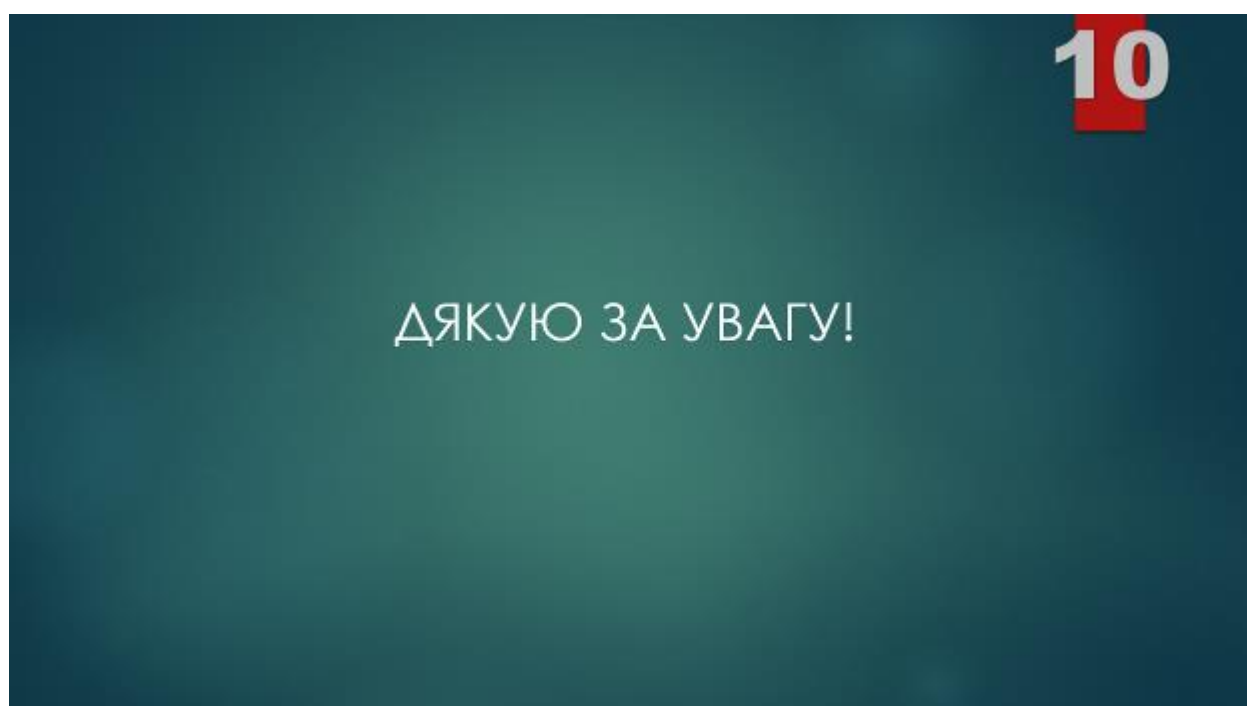


Рисунок Б.10 – Подяка