

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут
математики та інформаційних технологій

Кафедра математики та інформатики

Макаренко Інна Сергіївна

Розробка 3D гри в середовищі Unity

кваліфікаційна робота

здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми « Комп’ютерні науки та інформаційні технології »
за спеціальністю 122 Комп’ютерні науки

Особистий підпис _____ 

Науковий керівник _____ Владислав КОЗУБ, доктор філософії

Завідувач кафедри _____ Микола СЕМЕНОВ, канд.пед. наук,
доцент

Полтава – 2025

Міністерство освіти і науки України
Державний заклад „Луганський національний університет
імені Тараса Шевченка”

Інститут математики та інформаційних технологій

Кафедра	Інформаційних технологій та систем
Освітній рівень	бакалавр
Напрямок підготовки (спеціальність)	122 “Комп’ютерні науки”
Галузь знань	12 “Інформаційні технології”

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТС

_____	_____
(підпис)	(ім’я, прізвище)
“ ”	202 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Макаренко Інна Сергіївна
(прізвище, ім’я, по батькові)

1. Тема Розробка 3D гри в середовищі Unity

Керівник кваліфікаційної
роботи

(прізвище, ініціали, науковий ступінь, вчене звання)

Затверджена наказом по
університету

від “ ” 202 року №

2. Строк подання здобувачем
вищої освіти проєкту

3. Вихідні дані до роботи
(проєкту)

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об’єкт розробки)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____ ” _____ 202_ року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	до 1 лютого	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	до 20 березня	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником	до 1 квітня	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	до 15 квітня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	до 30 квітня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	до 15 травня	
7.	Попередній захист роботи на кафедрі	до 30 травня	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	за 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	за 5 днів до державної атестації	

Здобувач вищої освіти



Інна Макаренко

підпис

(ім'я, прізвище)

Керівник проєкту (роботи)

підпис

(ім'я, прізвище)

АНОТАЦІЯ

Макаренко І.С.

Тема: Розробка 3D гри в середовищі Unity

Спеціальність: 122 „Комп’ютерні науки”

Установа: ДЗ ЛНУ імені Т.Шевченка, 2025 р.

Кваліфікаційна робота містить: 66 с., 19 рис., 1 табл., 15 джерел.

Об’єкт дослідження – процес розробки тривимірних комп’ютерних ігор.

Предмет дослідження – технології створення 3D ігор з використанням рушія Unity.

Мета роботи – створення повноцінної 3D гри з базовим геймплеєм, функціональними елементами та можливістю розширення.

Результати роботи. У процесі роботи було проаналізовано сучасні тенденції геймдизайну, особливості роботи середовища Unity, розроблено концепцію гри, реалізовано основні механіки руху гравця, перешкод, меню рівнів та систему звукового супроводу.

Висновки. У висновках підкреслено ефективність використання Unity для розробки тривимірних ігор початкового рівня та окреслено перспективи подальшого розвитку створеної гри.

Ключові слова. UNITY, 3D ГРА, ГЕЙМДИЗАЙН, ПРОГРАМУВАННЯ, СЦЕНАРІЙ ГРИ, ФУНКЦІОНАЛЬНІСТЬ.

ABSTRACT

Makarenko I.

Theme: Developing a 3D Game in Unity

Specialty: 122 "Computer Science"

Institution: Taras Shevchenko Luhansk National University, 2025

Qualification work contains: 66 pages, 19 figures, 1 table, 15 sources.

Object of research: the process of developing three-dimensional computer games.

Subject of research: technologies for creating 3D games using the Unity engine.

Purpose of the study: develop a complete 3D game with basic gameplay mechanics, functional elements, and possibilities for future expansion.

Results of research. During the research, modern trends in game design and the features of the Unity environment were analyzed; the concept of the game was developed; core mechanics for player movement, obstacles, level menus, and the sound system were implemented.

Conclusions. The conclusions highlight the effectiveness of using Unity for developing beginner-level 3D games and outline prospects for the further development of the created game.

Keywords. UNITY, 3D GAME, GAME DESIGN, PROGRAMMING, GAME SCENARIO, FUNCTIONALITY.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Історія розвитку ігрової індустрії.....	11
1.2 Основи геймдизайну	14
1.3 Огляд ігрових рушіїв (Unity, Unreal Engine тощо)	18
1.4 Висновки до розділу 1	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА 3D ГРИ В СЕРЕДОВИЩІ UNITY	30
2.1 Вибір інструментів і технологій.....	30
2.2 Розробка концепції гри.....	34
2.3 Структура гри та побудова сцени.....	36
2.4 Сценарій гри та програмна реалізація	47
2.5 Висновки до розділу 2	55
РОЗДІЛ 3. ТЕСТУВАННЯ, ОЦІНКА ТА ПЕРСПЕКТИВИ РОЗВИТКУ ГРИ	57
3.1 Тестування працездатності гри.....	57
3.2 Аналіз якості гри з точки зору користувача.....	61
3.3 Можливості покращення та подальшого розвитку	62
3.4 Висновки до розділу 3	64
ВИСНОВКИ	65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AAA-проєкти	–	Відеоігри найвищого рівня виробництва з великими бюджетами
ПК	–	Персональний комп'ютер
WebGL	–	Web Graphics Library (технологія для відображення 3D-графіки в браузері)
AI	–	Artificial Intelligence (штучний інтелект)
API	–	Application Programming Interface (інтерфейс прикладного програмування)
AR	–	Augmented Reality (доповнена реальність)
ARPG	–	Action Role-Playing Game (рольова гра з елементами екшену)
FPS	–	First Person Shooter (шутер від першої особи)
JRPG	–	Japanese Role-Playing Game (японська рольова гра)
MIT	–	Massachusetts Institute of Technology (Массачусетський технологічний інститут (ліцензія MIT))
RPG	–	Role-Playing Game (рольова гра)
UI	–	User Interface (користувацький інтерфейс)
UX	–	User Experience (користувацький досвід)
VR	–	Virtual Reality (віртуальна реальність)

ВСТУП

Актуальність роботи

Індустрія відеоігор стрімко розвивається, стаючи однією з найбільш прибуткових і технологічно насичених сфер сучасного інформаційного суспільства. Щороку виходить величезна кількість нових ігор, які пропонують користувачам все більш складний геймплей, реалістичну графіку та інтерактивність. У такій динамічній ситуації створення якісної 3D-гри в середовищі Unity має особливу актуальність, оскільки цей рушій надає потужні інструменти для розробки ігор різних жанрів і рівнів складності.

Кваліфікаційна робота присвячена розробці тривимірної гри на базі Unity, що є важливим практичним кроком у підготовці фахівців з інформаційних технологій.

Актуальність теми зумовлена високим попитом на кваліфікованих розробників ігор, здатних ефективно використовувати сучасні рушії для створення якісного програмного продукту. Unity є одним із найпопулярніших середовищ розробки, що використовується як інді-розробниками, так і великими ігровими студіями. Актуальність також підтверджується необхідністю глибокого розуміння процесів геймдизайну, програмування і 3D-моделювання для створення конкурентоспроможних продуктів.

Мета роботи

Мета бакалаврської роботи полягає у розробці повноцінної 3D-гри в середовищі Unity із функціональними елементами геймплею, системою управління, графічним оформленням та базовою механікою взаємодії з об'єктами ігрового світу.

Досягнення мети включає розв'язання таких **завдань**:

- 1) проаналізувати особливості розвитку ігрової індустрії та геймдизайну;

- 2) вивчити функціональні можливості середовища Unity;
- 3) спроектувати концепцію 3D-гри;
- 4) реалізувати базову механіку руху, взаємодії, переходів між рівнями;
- 5) забезпечити тестування працездатності гри;
- 6) проаналізувати результати тестування та запропонувати можливі шляхи вдосконалення гри.

Об'єктом дослідження є процес розробки тривимірної комп'ютерної гри у середовищі Unity.

Предметом дослідження є програмні засоби та технології, що забезпечують реалізацію основних ігрових механік, графічного оформлення та інтерфейсу користувача у 3D-іграх.

Практичне значення отриманих результатів

Створено діючий прототип 3D-гри, який може бути використаний як основа для подальшої розробки повноцінного продукту, а також у напрацюванні практичних навичок роботи з рушієм Unity, що є важливим для майбутньої професійної діяльності у сфері розробки ігор.

Структура і обсяг роботи

Робота складається з вступу, трьох розділів, висновків списку використаних джерел. Обсяг роботи становить 66 сторінок, обсяг використаної літератури – 15 джерел.

Перший розділ містить аналіз предметної області та постановку завдання, розглядає історію розвитку ігрової індустрії, основи геймдизайну та особливості середовища розробки Unity.

Другий розділ присвячений проектуванню гри, розробці сценарію, програмній реалізації основних механік, створенню інтерфейсу користувача та звукового супроводу.

Третій розділ включає тестування працездатності гри, аналіз якості з точки зору користувача та пропозиції щодо можливого подальшого розвитку проєкту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія розвитку ігрової індустрії

Ігрова індустрія – це одна з найдинамічніших галузей цифрових технологій, яка охоплює розробку, виробництво та продаж комп'ютерних ігор. Вона має глибоке історичне коріння, що бере початок ще з середини ХХ століття, і на сьогоднішній день є одним із головних сегментів індустрії розваг. У цьому підрозділі розглянемо основні етапи розвитку ігрової індустрії, її значення у сучасному світі, вплив на суспільство та роль технологій у формуванні нових форматів взаємодії з користувачем.

1.1.1 Початки та перші ігри (1950 - 1970 роки)

Розвиток відеоігор розпочався у 1950-х роках з появою перших експериментів у сфері обчислювальної техніки. Однією з найвідоміших перших ігор вважається "Tennis for Two" (1958), розроблена Вільямом Хігінботемом. Вона була відображена на осцилографі і дозволяла двом гравцям грати у спрощений варіант тенісу.

У 1962 році студент Массачусетського технологічного інституту Стів Рассел створив "Spacewar!", одну з перших комп'ютерних ігор, яка запускалася на PDP-1. Це стало передвісником майбутніх аркадних ігор.

1.1.2 Золота ера аркадних автоматів (1970 - 1985 роки)

У 1970-х роках відеоігри почали виходити на комерційний ринок. Компанія Atari у 1972 році випустила гру "Pong", яка стала надзвичайно популярною і започаткувала масову індустрію аркадних автоматів.

Протягом 1980-х років з'явилися легендарні ігри, як-от:

- "Pac-Man" (1980) – гра, яка вперше звернула увагу на широку аудиторію, включно з жінками;
- "Donkey Kong" (1981) – гра, що стала дебютом для персонажа Маріо від компанії Nintendo;
- "Tetris" (1984) – розроблений радянським програмістом Олексієм Пажитновим, він завоював світ і став символом ігрової культури.

Цей період також відзначився розвитком домашніх консолей: Atari 2600, ColecoVision та інших.

1.1.3 Криза 1983 року та відродження

У 1983 році індустрія відеоігор пережила глибоку кризу, пов'язану з перенасиченням ринку іграми низької якості, зокрема провалом гри "E.T. the Extra-Terrestrial". Багато компаній зазнали банкрутства, а довіра споживачів була підірвана.

Відродження настало завдяки японській компанії Nintendo, яка у 1985 році випустила Nintendo Entertainment System (NES) і гру "Super Mario Bros.", що започаткувала нову еру якості, брендування та ігрових франшиз.

1.1.4 Еволюція графіки та технологій (1990 - 2005 роки)

У 1990-х роках на перший план вийшла тривимірна графіка. З'явилися перші 3D-ігри, такі як:

- "Doom" (1993) – одна з перших FPS-ігор з псевдо-3D графікою;
- "Quake" (1996) – з повноцінною 3D-графікою;
- "Final Fantasy VII" (1997) – культова JRPG із кінематографічною подачею сюжету.

Поява Sony PlayStation, Sega Saturn, а пізніше PlayStation 2, Xbox, сприяла зміцненню ринку домашніх консолей. Графіка ставала дедалі реалістичнішою, з'являлися ігри з відкритим світом, як-от GTA III (2001).

1.1.5 Становлення індустрії як масової культури (2005 - 2015 роки)

Ігрова індустрія стала частиною глобальної культури. З'являються великі ігрові студії (Ubisoft, Electronic Arts, Bethesda), а бюджети на розробку блокбастерів сягають сотень мільйонів доларів.

Ігри на кшталт The Witcher 3, Skyrim, Assassin's Creed, Call of Duty визначали покоління геймерів. З розвитком онлайну з'являються масові мультиплеєрні проєкти (World of Warcraft, League of Legends, Dota 2).

1.1.6 Сучасний етап: незалежні розробники, VR, Unity та мобільний геймплей

Сьогоднішній стан індустрії позначений:

- a) розвитком VR та AR (Oculus, Meta Quest);
- b) зростанням ринку мобільних ігор (Clash of Clans, Candy Crush);
- c) популярністю інді-проєктів, які створюються невеликими командами або навіть однією людиною (Undertale, Stardew Valley);
- d) широким використанням ігрових рушіїв, таких як Unity та Unreal Engine.

Unity – один із найпоширеніших рушіїв, що дозволяє створювати 2D та 3D ігри для різних платформ. Саме завдяки Unity тисячі розробників, включаючи студентів, можуть створювати повноцінні продукти без гігантських бюджетів.

1.2 Основи геймдизайну

1.2.1 Поняття геймдизайну

Геймдизайн – це процес створення правил, логіки, механік та сценаріїв, які визначають поведінку гравця у віртуальному середовищі. Це одна з ключових складових розробки будь-якої гри, яка поєднує технічні, творчі та психологічні аспекти з метою створення цікавого та захоплюючого ігрового досвіду.

Геймдизайн включає в себе планування структури гри, визначення ігрового процесу (геймплею), розробку системи мотивації гравця, побудову світу гри, балансування рівнів складності та створення інтерфейсу взаємодії. Основне завдання геймдизайнера – забезпечити комфортне, інтуїтивне та цікаве середовище для користувача.

1.2.2 Основні компоненти геймдизайну

До основних елементів геймдизайну належать:

Ігрова механіка – набір правил і систем, що визначають, як працює гра. Наприклад, у платформерах механікою є стрибки, уникання перешкод, збирання бонусів.

Ігровий процес (геймплей) – це те, як гравець взаємодіє з грою. Він включає дії, що виконуються в грі, і реакцію системи на ці дії.

Прогресія гри – послідовність рівнів, ускладнення завдань, розвиток персонажів або відкриття нових можливостей. Важливо, щоб складність зростала поступово, не викликаючи фрустрації.

Баланс – це узгодження між складністю, нагородами, ворогами та ресурсами. Хороший баланс гарантує, що гравець відчуває виклик, але не розчарування.

Сюжет та наратив – історія, що лежить в основі гри. Навіть у простих іграх сюжет може допомогти втягнути гравця та зробити досвід емоційно насиченим.

1.2.3 Типи ігор та жанри

Розробка гри починається з вибору жанру, адже саме він визначає базову структуру геймплею.

Існує багато жанрів ігор, кожен з яких має свої особливості, механіки та типову аудиторію:

Екшн (Action) – ігри, що вимагають швидкої реакції, точності та спритності. Наприклад, шутери (FPS), платформери, файтинги. Популярні приклади: DOOM, Call of Duty, Super Mario.

Пригодницькі (Adventure) – акцент на сюжет, дослідження, вирішення головоломок. Часто мають повільніший темп. Наприклад, The Legend of Zelda, Life is Strange.

Рольові (RPG) – ігри з розвитком персонажа, виборами в діалогах, глибоким сюжетом. Поділяються на класичні, JRPG (японські), ARPG (екшен-RPG). Приклади: The Witcher 3, Final Fantasy.

Симулятори – моделюють реальні або вигадані процеси: польоти, фермерство, життя. Приклади: The Sims, Flight Simulator.

Стратегії – ігри, де потрібно планувати дії, управляти ресурсами, будувати армії. Є покрокові та в реальному часі. Приклади: Civilization, StarCraft.

Головоломки (Puzzle) – зосереджені на логіці, розв’язанні завдань. Наприклад, Tetris, Portal.

Спортивні – імітують реальні види спорту: футбол, баскетбол, гонки тощо. Приклади: FIFA, NBA 2K, Gran Turismo.

1.2.4 Геймдизайн у 3D-іграх

Геймдизайн у 3D-іграх має свої унікальні особливості, що відрізняють його від розробки 2D-ігор. Основна відмінність полягає у тривимірному просторі, який додає нові рівні складності, але й відкриває більше можливостей для створення захопливого ігрового процесу.

Просторова навігація та орієнтація: У 3D-просторі гравець має змогу рухатися у всіх трьох вимірах (вліво/вправо, вперед/назад, вгору/вниз). Це створює потребу у ретельному проєктуванні локацій та рівнів, які мають бути інтуїтивно зрозумілими та не викликати дезорієнтації у гравця. Важливу роль відіграє розміщення орієнтирів, таких як освітлення, архітектурні елементи чи кольори.

Камера та кут огляду: У 3D-іграх камера може бути фіксованою, від третьої чи від першої особи. Кожен з цих варіантів впливає на ігровий процес. Наприклад, камера від першої особи забезпечує глибше занурення у гру, тоді як камера від третьої особи дозволяє краще оцінювати навколишнє середовище та планувати дії.

Фізика і взаємодія з об’єктами: 3D-середовище дозволяє реалізувати складні фізичні взаємодії – падіння, стрибки, зіткнення, вагу предметів тощо. Геймдизайнер має передбачити, як гравець може маніпулювати об’єктами в просторі та які механіки це відкриває (наприклад, штовхання ящика, щоб залізти вище).

Естетика середовища та атмосфера: Візуальна складова у 3D-іграх особливо важлива. Ретельно змодельовані локації, текстури, освітлення й ефекти створюють атмосферу гри, яка значно впливає на сприйняття. Геймдизайнер разом із художниками визначає стиль гри – реалістичний, мультяшний, стилізований тощо.

Тестування простору: 3D-простір складніше тестувати на наявність помилок, таких як застрягання персонажа, порушення логіки переміщення чи “невидимі стіни”. Геймдизайнер має передбачати поведінку гравця в різних ситуаціях і уникати “мертвих зон” або неочікуваних сценаріїв.

1.2.5 Взаємодія геймдизайну з іншими елементами розробки

Геймдизайн – це серцевина ідеї гри, яка взаємодіє з усіма іншими компонентами розробки: програмуванням, графікою, звуком, наративом, тестуванням та навіть маркетингом. Лише завдяки тісній співпраці між цими сферами можна створити цілісний, логічний і привабливий ігровий продукт.

Геймдизайн та програмування: Геймдизайнер визначає, як повинні працювати механіки, а програмісти реалізують ці ідеї у коді. Наприклад, дизайнер описує поведінку ворогів, правила бою, систему інвентарю чи фізику об'єктів – і все це програміст має втілити у скриптах. Часто геймдизайнер створює документацію, у якій описує кожну функцію, що потім стає технічним завданням для розробника.

Геймдизайн і візуальне оформлення: Графіка і геймдизайн тісно пов'язані. Візуальний стиль гри має відповідати її жанру, настрою та геймплейним особливостям. Наприклад, якщо гравець має легко орієнтуватися у просторі, дизайнер співпрацює з 3D-модельєрами та художниками для створення впізнаваних орієнтирів, логічних маршрутів і візуально зрозумілих об'єктів. Крім того, кольори та освітлення можуть бути частиною ігрових механік.

Геймдизайн і звук: Звуковий супровід – важлива частина атмосфери гри. Геймдизайнер визначає, в яких ситуаціях повинні звучати ті чи інші звуки: кроки, удари, фонові мелодії, музика напруги тощо. Наприклад, у момент небезпеки може змінюватися музика, а звук відкривання скрині має бути приємним та задовольняти гравця.

Геймдизайн і сюжет (нарратив): У сюжетних 3D-іграх геймдизайн тісно пов'язаний з нарративом. Подача історії – це не лише діалоги чи катсцени, а й ігрові ситуації, в яких гравець переживає події. Наприклад, втеча з палаючого міста або боротьба з босом – це сюжетні моменти, які передаються через геймплей. Тому геймдизайнер співпрацює зі сценаристами, щоб логіка ігрового процесу не суперечила розвитку сюжету.

Геймдизайн і тестування: Геймдизайнер має враховувати зворотний зв'язок від тестувальників і гравців. Баланс, складність, баги – усе це виявляється на етапі тестування, після чого геймдизайнер аналізує інформацію і вносить зміни в ігрові механіки. Наприклад, якщо гравцям надто складно пройти рівень, його треба спростити або додати підказки.

Геймдизайн і маркетинг: Навіть маркетинг впливає на геймдизайн. Дизайнер має враховувати цільову аудиторію, популярні тренди і вимоги платформи. Наприклад, free-to-play модель вимагає додаткового балансу між безкоштовним контентом і платними елементами, що безпосередньо впливає на геймплей.

1.3 Огляд ігрових рушіїв (Unity, Unreal Engine тощо)

Ігровий рушій (англ. game engine) – це програмне середовище, яке дозволяє розробникам створювати ігри без необхідності писати весь код "з

нуля". Він надає набір інструментів для роботи з графікою, фізикою, звуком, штучним інтелектом, мережевими технологіями тощо. Існує чимало рушіїв, але найпопулярнішими є Unity, Unreal Engine, Godot та CryEngine. У цьому пункті ми розглянемо їхні особливості, переваги та недоліки.

1.3.1 Unity

Unity – це один із найпопулярніших ігрових рушіїв у світі, який був створений у 2005 році компанією Unity Technologies. Його ключова особливість – кросплатформність, що дозволяє розробникам створювати ігри для понад 25 різних платформ, включаючи Windows, macOS, Android, iOS, WebGL, PlayStation, Xbox, Nintendo Switch та інші.

Основні характеристики Unity:

- мова програмування: основною мовою, що використовується в Unity, є C#. Завдяки простому синтаксису ця мова є зручною навіть для початківців.
- редактор сцени: Unity має вбудований візуальний редактор, де розробник може розміщувати об'єкти, налаштовувати їх властивості, компоненти та взаємодію між ними.
- фізичний рушій: Unity підтримує фізику як для 2D, так і для 3D проєктів. Він забезпечує взаємодію об'єктів у просторі, гравітацію, колізії, сили тощо.
- підтримка віртуальної та доповненої реальності (VR/AR): Unity активно використовується для створення VR/AR-додатків, включаючи проєкти для шоломів Oculus Rift, HTC Vive, HoloLens та ін.
- модульність: Unity використовує компонентну архітектуру – кожен об'єкт складається з набору компонентів (скрипти, рендеринг, фізика тощо), які легко змінювати та комбінувати.

- Unity Asset Store: це вбудований магазин ресурсів, у якому можна знайти безкоштовні й платні моделі, текстури, анімації, скрипти, шейдери та цілі системи для використання у власному проєкті.
- підтримка URP та HDRP: Universal Render Pipeline (URP) і High Definition Render Pipeline (HDRP) – це рендерингові системи, які дозволяють адаптувати візуальні ефекти під потреби мобільних чи потужних платформ.

Переваги Unity:

- a) простота використання: Unity має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко розпочати роботу навіть без попереднього досвіду у розробці ігор.
- b) велика спільнота: завдяки популярності рушія, існує безліч навчальних матеріалів, курсів, форумів і відео-уроків. Це полегшує процес навчання і вирішення проблем.
- c) гнучкість: Unity можна використовувати не лише для створення ігор, а й для архітектурної візуалізації, анімації, симуляцій, навчальних програм, додатків в медицині, освіті тощо.
- d) кросплатформність: розроблену гру можна швидко експортувати на різні платформи без потреби писати окремий код для кожної з них.
- e) широкий спектр шаблонів: Unity підтримує як 2D-, так і 3D-шаблони, а також має підтримку для VR/AR-проєктів, шутерів, платформерів, головоломок тощо.

Недоліки Unity:

- a) продуктивність на великих сценах: при створенні складних 3D-сцен Unity може потребувати додаткової оптимізації, оскільки система управління пам'яттю не завжди ефективна при масштабних проєктах.

- b) комерційна ліцензія: безкоштовна версія має обмеження щодо доходу проєкту. Якщо розробник заробляє понад 100 000 доларів на рік, необхідно купувати професійну ліцензію.
- c) графіка нижчого рівня ніж Unreal Engine: хоча Unity має сильні візуальні можливості, за якістю графіки Unreal Engine часто перевершує Unity, особливо в AAA-проєктах.

Сфери застосування Unity:

окрім створення комп'ютерних і мобільних ігор, Unity використовується в таких сферах:

- освіта (інтерактивні симуляції, навчальні платформи)
- медицина (VR-сценарії для тренувань хірургів)
- архітектура та будівництво (візуалізація інтер'єрів та екстер'єрів)
- автомобільна промисловість (3D-моделювання інтерфейсів і технічних процесів)
- кінематографія (анімаційні фільми та візуальні ефекти)

Приклади ігор, створених на Unity:

- Hollow Knight – платформер з атмосферною графікою та бойовою механікою.
- Monument Valley – інноваційна головоломка з чудовим візуальним дизайном.
- Cuphead (початкові версії) – відома гра, що використовує стилістику мультфільмів 1930-х.
- Ori and the Blind Forest (частково Unity) – візуально приголомшлива гра.
- Pokemon Go – мобільна AR-гра, що працює частково на Unity.

1.3.2 Unreal Engine

Unreal Engine – це потужний ігровий рушій, розроблений компанією Epic Games. Він використовується у створенні як 2D, так і 3D ігор, зокрема проєктів з фотореалістичною графікою, віртуальною реальністю, симуляціями та архітектурними візуалізаціями.

Основні характеристики Unreal Engine:

- графіка високої якості: Unreal Engine забезпечує один із найкращих рівнів візуалізації серед усіх рушіїв. Завдяки використанню рушія Nanite (у Unreal Engine 5) та системі глобального освітлення Lumen, розробники можуть досягти майже фотореалістичних результатів.
- Blueprints – візуальне програмування: ця система дозволяє створювати ігрову логіку без глибоких знань програмування. Вона дуже зручна для дизайнерів і початківців.
- C++ для розробників: Unreal Engine підтримує повноцінне програмування на C++, що надає доступ до повної гнучкості рушія.
- інструменти для кінематографії: Unreal Engine має потужний набір інструментів для створення анімацій і кінематографічних сцен (наприклад, Sequencer).
- безкоштовне використання: Unreal Engine є безкоштовним до моменту, поки дохід від гри не перевищить \$1 млн. Після цього розробники сплачують роялті (звичайно 5% від прибутку).

Переваги Unreal Engine:

- a) підтримка великих, відкритих світів і великої кількості об'єктів на сцені.
- b) багатий набір інструментів прямо "з коробки" – майже не потрібно сторонніх плагінів.
- c) потужна система матеріалів і шейдерів.
- d) підтримка VR, AR, консольних та мобільних платформ.

Недоліки Unreal Engine:

- a) високі системні вимоги для розробки (потребує потужного комп'ютера).
- b) більш складний поріг входження для новачків у порівнянні з Unity.
- c) великий розмір проєктів та рушія.

Приклади ігор, створених на Unreal Engine:

- Fortnite (Epic Games) – багатокористувацький бойовик.
- Gears of War – серія шутерів від третьої особи.
- The Matrix Awakens – технодемка, яка демонструє можливості UE5.
- S.T.A.L.K.E.R. 2 (у розробці) – сучасний приклад використання рушія для великих відкритих світів.

1.3.3 Godot Engine

Godot Engine – це безкоштовний і відкритий ігровий рушій, що стрімко набирає популярності завдяки своїй простоті, гнучкості та активному розвитку спільноти. Godot дозволяє розробляти 2D та 3D ігри для різних платформ: Windows, Linux, macOS, Android, iOS, HTML5 тощо.

Основні переваги Godot:

- a) відкритий вихідний код: Godot є проєктом з відкритим вихідним кодом під ліцензією MIT, що дозволяє розробникам вільно змінювати рушій, адаптувати його під свої потреби без обмежень ліцензування.
- b) вбудований скриптовий редактор і власна мова GDScript: GDScript – мова програмування, схожа на Python, розроблена спеціально для Godot. Вона легка у вивченні та ідеально підходить для новачків.
- c) візуальний редактор сцени: Godot використовує підхід до побудови гри за допомогою ієрархії сцен (Scene Tree), де кожен об'єкт є вузлом

(Node). Це дозволяє зручно компонувати елементи ігрового світу та організовувати логіку.

- d) підтримка 2D і 3D: рушій однаково добре підтримує як 2D-графіку, так і 3D. Причому 2D-система не є просто додатком до 3D, а реалізована як повноцінний модуль.
- e) легка вага та швидкість запуску: Godot займає дуже мало місця, запускається швидко і не потребує встановлення – достатньо просто запустити файл рушія.
- f) кросплатформеність: готові ігри можна компілювати для великої кількості платформ – від мобільних до браузерних.

Недоліки Godot:

- a) менш потужний рендеринг для 3D у порівнянні з Unity або Unreal Engine (хоча у версії Godot 4.0 було значно покращено 3D-движок).
- b) відносно менша кількість навчальних ресурсів та документації, особливо для складних проєктів.
- c) обмежена підтримка сторонніх плагінів у порівнянні з Unity Asset Store або Unreal Marketplace.

Приклади ігор, створених на Godot Engine:

- "Kingdoms of the Dump" – піксельна RPG з унікальним світом, що розробляється з використанням Godot.
- "Rogue State Revolution" – стратегія, що ілюструє здатність рушія працювати з великою кількістю об'єктів і сценаріїв.
- "The Garden Path" – гра в жанрі life-sim, з атмосферою графікою та процедурною генерацією подій.
- "Ex Zodiac" – шутер у стилі Star Fox, який показує потужність рушія в аркадних 3D-проєктах.

1.3.4 CryEngine

CryEngine – це потужний ігровий рушій, розроблений німецькою компанією Crytek. Перший раз рушій був використаний у грі Far Cry (2004), після чого він отримав широке визнання завдяки приголомшливій графіці та високій продуктивності. CryEngine дозволяє створювати високоякісні ігри з реалістичним візуальним оформленням, особливо в жанрі шутерів і відкритих світів.

Ключові особливості CryEngine:

- фотореалістична графіка – рушій надає потужні інструменти для створення реалістичних освітлення, тіней, води, ландшафтів та ін.
- Advanced AI System – система штучного інтелекту дозволяє легко створювати складну поведінку персонажів.
- Sandbox Editor – CryEngine має візуальний редактор у режимі реального часу – один із найпотужніших у галузі.
- підтримка VR – рушій активно використовується для розробки проєктів у віртуальній реальності.
- вбудована фізика – потужна система фізики надає реалістичну взаємодію об'єктів.
- висока продуктивність – оптимізований для рендерингу великої кількості об'єктів на сцені.

Переваги CryEngine:

- a) вражаюча графіка – CryEngine здатний створювати високодеталізовані сцени з реалістичним освітленням, тінями, водою та природними ефектами. Це один із головних рушіїв у сфері фотореалізму.
- b) потужний візуальний редактор (Sandbox) – дозволяє змінювати об'єкти на сцені у реальному часі, з негайним попереднім переглядом результатів.

- с) високий рівень контролю над фізикою та анімаціями – рушій має просунуту систему фізики, включаючи руйнування об'єктів, кінематику та реалістичну поведінку матеріалів.
- d) підтримка VR та широких платформ – CryEngine підтримує розробку для ПК, консолей і VR-пристроїв.
- e) безкоштовне використання (з роялті) – Crytek надає рушій безкоштовно для багатьох користувачів, вимагаючи роялті лише після досягнення певного рівня доходу.

Недоліки CryEngine:

- a) CryEngine менш інтуїтивний, ніж Unity чи Unreal, особливо для новачків. Його складний інтерфейс вимагає більше часу для освоєння.
- b) у порівнянні з іншими рушіями, CryEngine має обмежену спільноту, менше відеоуроків і прикладів.
- c) для розробки та запуску проєктів потрібен потужний комп'ютер. Це може стати бар'єром для початківців.
- d) на відміну від Unity Asset Store або Unreal Marketplace, CryEngine має скромніший вибір додаткових ресурсів.
- e) менш популярний серед інді-розробників: через вищезгадані фактори CryEngine рідше використовується в інді-середовищі.

Приклади ігор, створених на CryEngine:

- Crysis (серія) – знаменита FPS-франшиза, яка стала еталоном графічної якості.
- Sniper: Ghost Warrior Contracts 2 – тактичний шутер з реалістичною фізикою і далекобійною стрільбою.
- Wolfen: Lords of Mayhem – ізометрична RPG з динамічним бойовим процесом.

1.3.5 Порівняльна характеристика рушіїв

Нижче в табл 1.1 можна побачити порівняння ключових параметрів рушіїв (Unity, Unreal, Godot, CryEngine).

Таблиця 1.1

Порівняльна характеристика ігрових рушіїв

Параметр	Unity	Unreal Engine	Godot	CryEngine
Мова програмування	C#, JavaScript (застарілий)	C++, Blueprint	GScript, C#, C++	C++, Lua
Графіка	Висока, налаштовувана	Дуже висока, фотореалістична	Середня, залежить від налаштувань	Дуже висока, особливо в природі
Інтерфейс	Зручний, простий у вивченні	Потужний, але складніший	Простий і легкий	Складний, але гнучкий
Складність вивчення	Легка для початківців	Середня / висока	Легка	Висока
Ліцензія / Вартість	Безкоштовна + підписка	Безкоштовна до \$1 млн доходу	Повністю безкоштовна (MIT)	Безкоштовна з роялті
Підтримка платформ	Багато (ПК, моб., VR, Web)	Багато (ПК, моб., VR, консолі)	ПК, Web, Android, iOS	ПК, консолі, VR
Підтримка спільноти	Дуже велика	Велика	Зростає	Обмежена
Маркетплейс / ресурси	Unity Asset Store	Unreal Marketplace	Існують, але обмежені	Мало ресурсів

Аналіз за напрямками застосування:

- a) інді-розробка: Unity і Godot – ідеальні варіанти. Unity має багату екосистему, Godot – повністю безкоштовний і легкий у вивченні.
- b) освітні проєкти / студенти: Godot – через простоту й відкритий код, або Unity, завдяки великій кількості курсів.
- c) AAA-проєкти: Unreal Engine і CryEngine. Вони мають вражаючі графічні можливості, підтримку великих команд і інструментів.
- d) кросплатформеність: Unity має найбільшу кількість підтримуваних платформ (включаючи WebGL та VR).

Кожен рушій має свої переваги, але для розробки освітнього 3D-проєкту з акцентом на легкість вивчення, доступність інструментів та документацію, Unity є найкращим вибором. Його зручний інтерфейс, величезна база знань і підтримка різних платформ дозволяють створювати якісні 3D-ігри навіть новачкам.

1.4 Висновки до розділу 1

У першому розділі було проведено всебічний аналіз предметної області, що дозволив глибше зрозуміти контекст, в якому розробляється 3D-гра, а також сформулювати чітке завдання проєкту.

У підпункті 1.1 було розглянуто історію та сучасний стан ігрової індустрії. Проаналізовано динаміку розвитку ігор з моменту їх появи до сучасності, акцентовано увагу на зростанні прибутковості та впливі 3D-технологій на геймдизайн.

Підпункт 1.2 був присвячений основам геймдизайну. Розглянуто базові поняття, ключові елементи ігрового процесу, жанри та типи ігор, а також

особливості дизайну саме у 3D-середовищі. Визначено важливість тісної взаємодії геймдизайну з іншими складовими гри – програмуванням, графікою, звуком тощо.

У підпункті 1.3 здійснено огляд найпопулярніших ігрових рушіїв: Unity, Unreal Engine, Godot та CryEngine. Наведено порівняльну характеристику рушіїв за ключовими параметрами. На основі аналізу зроблено висновок, що найбільш доцільним для розробки 3D-гри в умовах навчального проєкту є використання рушія Unity.

Таким чином, на основі проведеного аналізу було сформульовано чітке завдання: розробити 3D-гру з використанням рушія Unity, з урахуванням принципів геймдизайну, сучасних технологій і підходів до створення інтерактивного ігрового досвіду.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА 3D ГРИ В СЕРЕДОВИЩІ UNITY

2.1 Вибір інструментів і технологій

2.1.1 Обґрунтування вибору Unity

Для розробки 3D гри у межах цієї кваліфікаційної роботи було обрано ігровий рушій Unity, що є одним з найпопулярніших інструментів у світі геймдеву. Це рішення було прийняте з урахуванням низки технічних, функціональних та освітніх переваг, які надає Unity для початківців і професіоналів.

Основні причини вибору Unity:

- a) кросплатформеність – Unity підтримує розробку під більше ніж 25 платформ, включаючи Windows, macOS, Android, iOS, WebGL, PlayStation, Xbox тощо. Це дозволяє розширити аудиторію гри без необхідності масштабної переробки проєкту.
- b) дружній інтерфейс – Unity має інтуїтивно зрозуміле середовище розробки, що робить його зручним для новачків. Інтерфейс редактора дозволяє легко керувати сценами, об'єктами, компонентами та ресурсами.
- c) підтримка C# – Unity використовує мову програмування C#, яка є сучасною, об'єктно-орієнтованою і добре документованою. Це дає змогу легко організовувати код і масштабувати проєкт.
- d) розвинене ком'юніті – Існує величезна кількість документації, форумів, відеоуроків та безкоштовних курсів, які допомагають швидко знаходити рішення проблем та опановувати нові технології.

- e) Unity Asset Store – Це внутрішній магазин ресурсів, де розробники можуть знайти або придбати моделі, текстури, скрипти, анімації та інші елементи для гри. Це значно пришвидшує розробку та дозволяє зосередитися на унікальній логіці гри.
- f) гнучкість і масштабованість – Unity дозволяє реалізовувати як прості 2D-проекти, так і масштабні 3D-гри з високим рівнем деталізації. Підтримуються фізичні рушії, анімації, кінематографічні ефекти тощо.

Вибір Unity як основного рушія для реалізації проєкту зумовлений його доступністю, широким функціоналом, потужною підтримкою та відповідністю вимогам до 3D-ігор. Це оптимальне середовище для ефективної реалізації ігрового задуму в умовах навчального проєкту.

2.1.2 Додаткові інструменти (Visual Studio та Unity Asset Store)

У процесі розробки 3D гри за допомогою Unity важливу роль відіграють не лише сам рушій, але й інші допоміжні інструменти, які забезпечують зручне програмування, швидке створення контенту та підвищення ефективності розробки. У цьому проєкті основними додатковими інструментами були Visual Studio як середовище розробки та Unity Asset Store як джерело ресурсів.

Visual Studio є офіційно рекомендованим середовищем розробки для роботи з Unity. Його інтеграція з рушієм дозволяє ефективно писати, редагувати, налагоджувати та тестувати код на мові C#. Основні переваги Visual Studio:

- підсвічування синтаксису і автодоповнення: IDE допомагає уникати синтаксичних помилок і пришвидшує написання коду.
- інтеграція з Unity: Автоматичне підключення бібліотек Unity дозволяє зручно працювати з API рушія.

- налагодження (debugging): Visual Studio підтримує покрокове виконання коду, встановлення точок зупину та перегляд значень змінних.
- розширення і шаблони: Середовище підтримує встановлення плагінів, які оптимізують роботу над ігровими проєктами.

Завдяки глибокій інтеграції з Unity, Visual Studio дозволяє розробнику зосередитися безпосередньо на логіці гри, не відволікаючись на технічні складнощі.

Unity Asset Store – це офіційна платформа для обміну ресурсами між розробниками. Вона надає велику кількість готових 3D-моделей, шаблонів, текстур, звуків, анімацій, інструментів UI та скриптів, як платних, так і безкоштовних. Його використання має кілька переваг:

- економія часу: Замість створення всіх елементів з нуля, можна використовувати якісні ресурси з магазину.
- навчальні пакети: Багато ресурсів містять навчальні приклади або демонстраційні сцени, що допомагає краще зрозуміти їхню інтеграцію.
- висока якість: Ресурси, особливо платні, часто мають професійний рівень виконання, що дозволяє досягти кращої якості гри.
- широкий вибір: Asset Store охоплює майже всі аспекти розробки, включаючи інструменти для оптимізації, постобробки, навігації тощо.

В межах цієї роботи за допомогою Unity Asset Store були використані шаблони 3D-моделей, текстури навколишнього середовища та звукові ефекти, що дозволило зосередитися на розробці ігрової механіки та геймдизайну.

Використання Visual Studio і Unity Asset Store дозволяє значно підвищити продуктивність розробника, зменшити витрати часу та забезпечити

високу якість кінцевого продукту. Ці інструменти є важливою складовою сучасного процесу створення ігор у середовищі Unity.

2.1.3 Робоче середовище розробника

Робоче середовище розробника (англ. development environment) – це сукупність програмного забезпечення, інструментів та налаштувань, які забезпечують комфортну, продуктивну та ефективну розробку програмного продукту. У контексті створення 3D-гри в середовищі Unity, робоче середовище відіграє ключову роль, оскільки від нього залежить швидкість реалізації ідей, зручність налагодження коду та тестування функціоналу. Основні компоненти робочого середовища:

- a) операційна система – проєкт розроблявся в середовищі операційної системи Windows 11. Ця система забезпечує сумісність із більшістю інструментів для створення ігор, має зручний інтерфейс, а також підтримує стабільну роботу Unity Editor та Visual Studio.
- b) Unity Editor – основне середовище розробки, де відбувалось створення сцени, розміщення об'єктів, а також компонування логіки за допомогою скриптів. Unity забезпечує інтеграцію з Visual Studio, що спрощує написання коду та налагодження.
- c) Visual Studio – використовувалась як основний редактор коду. Інтеграція з Unity дозволяє зручно працювати зі скриптами на C#, використовуючи автодоповнення, підсвічування синтаксису та інструменти для налагодження.
- d) Unity Asset Store – для прискорення розробки частина візуальних і звукових ресурсів була взята з Unity Asset Store – офіційного магазину, що надає велику кількість готових моделей, текстур, звуків, анімацій та скриптів. Це дозволяє розробникам зосередитися на ігровій логіці, не витрачаючи надмірну кількість часу на створення базових елементів.

2.2 Розробка концепції гри

Розробка концепції – один із найважливіших етапів у створенні гри. На цьому етапі формується загальне бачення гри, визначаються сюжет, механіки та орієнтація на цільову аудиторію. Саме концепція визначає, якою буде гра, на кого вона орієнтована, які емоції викликатиме у гравця та яким буде її ігровий процес. Концепція гри, створена в межах цього проєкту, базується на принципах простоти, захоплюючого геймплею та поступового ускладнення.

2.2.1 Ідея та сюжет гри

Ідея гри полягає в створенні аркадного платформера з елементами виживання та реакції. Гравець керує персонажем, який рухається по рівню з автоматично прокручуваним фоном, уникаючи перешкод і намагаючись дістатися фінішу. Основна мета – вижити якомога довше та пройти всі рівні, які з кожним разом стають складнішими.

Сюжет у класичному розумінні не є головним елементом гри, однак присутній легкий наратив: головний герой – мандрівник, який досліджує різні віртуальні світи, кожен з яких має свої особливості. Кожен рівень – це нова "локація", яку потрібно подолати, щоб перейти до наступної.

Сюжетна лінія виконує допоміжну функцію, слугуючи мотивацією до проходження рівнів. Основний акцент зосереджено на динамічному геймплеї.

2.2.2 Основні механіки

Механіки гри є ключовими елементами, які визначають взаємодію гравця з ігровим світом. У грі реалізовані такі основні механіки:

- рух персонажа: Персонаж автоматично рухається вперед, а гравець може керувати стрибками. Це дозволяє долати перешкоди різної висоти.
- перешкоди: На шляху героя з'являються динамічні та статичні об'єкти, зіткнення з якими призводить до завершення рівня.
- анімація та ефекти: Стрибки та зіткнення супроводжуються звуковими ефектами, що робить гру більш динамічною.
- прогресія рівнів: З кожним наступним рівнем швидкість перешкод зростає, додаються нові типи небезпек.
- система початку та завершення гри: Перед початком гри з'являється екран із кнопками "Почати гру" та "Вибрати рівень". Після завершення рівня відображається меню з опціями "Пройти рівень повторно", "Наступний рівень" або "Вийти в меню".
- звуковий супровід: У грі реалізовано базовий звуковий супровід, який включає ефекти натискання кнопок, стрибків, зіткнень тощо.

2.2.3 Цільова аудиторія

Цільова аудиторія гри – користувачі віком від 10 до 30 років, які шукають легку, динамічну гру з простим керуванням. Гра підійде для:

- казуальних гравців, які хочуть пограти кілька хвилин у перервах;
- початківців, які ще не мають досвіду у складних геймплейних проєктах;
- мобільних гравців, оскільки гра потенційно може бути портована на мобільні платформи завдяки простому управлінню.

Завдяки простоті, приємному візуальному стилю та поступовому ускладненню геймплею, гра здатна утримати увагу користувача на кілька сеансів гри, що і є метою для ігор цього типу.

2.3 Структура гри та побудова сцени

Процес розробки 3D-гри передбачає не лише створення механік і візуального оформлення, але й логічну організацію структури проєкту, яка включає побудову рівнів, сцен, взаємодію об'єктів і переходи між ігровими станами. Кожен рівень – це окрема сцена, яка має власну логіку, налаштування й елементи середовища.

2.3.1 Рівні гри та їх логіка

Гра розроблена за принципом поетапного проходження рівнів, де кожен наступний рівень має вищу складність та нові виклики для гравця.

Загальна структура рівнів:

- кожен рівень реалізовано як окрему сцену в Unity.
- на рівні розташовані об'єкти середовища, фонові елементи, платформи, перешкоди та анімаційні елементи.
- зіткнення з перешкодами персонажа призводить до поразки і викликає появу меню з варіантами: "Play again" (Повторити рівень) або "Exit to menu" (Вийти в меню).

Логіка переходів:

- a) гра починається з головного меню, де користувач може вибрати: "Start Game" (Почати гру) або "Level selection" (Вибір рівня).
- b) при натисканні "Start Game" запускається перший рівень.
- c) після завершення кожного рівня (досягнення кінцевої точки) гравцеві показується екран успіху, де можна обрати: "Next Level" (Наступний рівень), "Level again" (Пройти рівень повторно) або "Exit to menu" (Вийти в меню).

- d) якщо натиснути "Next Level", запускається нова сцена з наступним рівнем.
- e) при використанні кнопки "Level selection" з головного меню, відкривається панель із кнопками, що дозволяють відразу перейти до будь-якого відкритого рівня.

Ускладнення з кожним рівнем:

- рівень 1 – ознайомчий, з мінімальною кількістю перешкод.
- рівень 2 – швидкість руху перешкод зростає, як і їх кількість, з'являються нові типи об'єктів.
- рівень 3 – підвищується кількість і швидкість перешкод, з'являються нові типи об'єктів.

Логіка взаємодії з гравцем:

- для початку гри на кожному рівні необхідно натиснути клавішу Enter, після чого персонаж починає рух.
- якщо гравець зазнає поразки, автоматично завантажується екран поразки з кнопками "Play again" або "Exit to menu".

Програмна реалізація:

- для організації логіки переходів і рівнів використовуються скрипти на мові C#.
- кожен рівень має унікальний індекс, що дозволяє динамічно завантажувати сцени за допомогою `SceneManager.LoadScene(index)`.
- всі глобальні змінні гри (номер рівня, стан гри тощо) зберігаються в окремому `GameManager`.

2.3.2 Створення ігрових об'єктів

У процесі побудови сцени гри важливу роль відіграє правильна організація ігрових об'єктів, їх розміщення, типи та поведінка під час гри. У даному проєкті реалізовано нескінченний "раннер", у якому створюється ілюзія руху завдяки анімації заднього фону та появи перешкод, що рухаються вліво. Сам персонаж при цьому залишається нерухомим у центрі екрана.

Основні типи об'єктів у проєкті:

а) гравець (Player):

- 1) центральний об'єкт гри, який стоїть нерухомо на платформі;
- 2) має компонент BoxCollider, що дозволяє виявляти зіткнення з іншими об'єктами, зокрема з перешкодами;
- 3) обладнаний анімацією стрибка, яка активується натисканням клавіші;
- 4) поведінка гравця контролюється через скрипт, який обробляє натискання клавіш і взаємодію з колайдерами.

б) фон (Background):

- 1) це одне зображення місцевості, яке рухається вліво для створення ефекту пересування;
- 2) переміщується за допомогою скрипта, який зміщує об'єкт уздовж осі X.
- 3) Скрипт bg.cs (рис. 2.1) забезпечує нескінченне повторення зображення фону, створюючи ефект безперервного руху. У методі Start() зберігається початкова позиція фону та визначається його ширина на основі компонента BoxCollider. У методі Update() виконується перевірка, чи зсунувся фон ліворуч поза межі своєї ширини. Якщо так – позиція фону скидається до початкової, завдяки чому він безперервно повторюється.

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class bg : MonoBehaviour
{
    private Vector3 startPosition;
    private float repeatWidth;

    void Start()
    {
        startPosition = transform.position;
        repeatWidth = GetComponent<BoxCollider>().size.x / 2;
    }

    void Update()
    {
        if (transform.position.x < startPosition.x - repeatWidth)
        {
            transform.position = startPosition;
        }
    }
}

```

Рис. 2.1. Скрипт bg.cs

с) платформа (Ground):

- 1) є нерухомою основою, на якій стоїть гравець;
- 2) має компонент BoxCollider для взаємодії з об'єктом гравця (зокрема, запобігання провалюванню);
- 3) залишається статичною під час гри, не переміщується разом із фоном.

d) перешкоди (Obstacles):

- 1) генеруються динамічно під час гри через певні інтервали;
- 2) рухаються ліворуч разом із фоном, створюючи виклик для гравця;
- 3) оснащені компонентом BoxCollider для перевірки зіткнень з гравцем;
- 4) можуть знищуватися після виходу за межі екрана для оптимізації продуктивності.

- 5) скрипт `Obstacle.cs` (рис. 2.2) відповідає за переміщення перешкод. Після досягнення певної координати об'єкти автоматично знищуються, звільняючи ресурси.

```

void Update()
{
    if (gm.gameStarted && !playerScript.gameOver)
    {
        transform.Translate(Vector3.left * speed * Time.deltaTime);
    }

    if (transform.position.x < leftBound && gameObject.CompareTag("obstacle"))
    {
        Destroy(gameObject);
    }
}

```

Рис. 2.2. Частина скрипта `Obstacle.cs`

Принцип побудови:

- створення об'єкта у Unity (`GameObject`);
- призначення спрайта або моделі;
- додавання фізичних компонентів (у даному випадку `BoxCollider`);
- призначення скриптів для поведінки об'єкта (рух, взаємодія);
- розміщення на сцені з урахуванням логіки геймплею.

Оптимізація за допомогою Prefab-ів:

Усі перешкоди створено як Prefab-и, які можна багаторазово використовувати в грі. Це дозволяє ефективно реалізувати генерацію нових елементів за допомогою скриптів без необхідності дублювати об'єкти вручну.

2.3.3 Камера, освітлення, UI

У процесі створення гри важливу роль відіграють не лише ігрові об'єкти та рівні, а й технічні аспекти візуалізації – такі як налаштування камери, освітлення та інтерфейс користувача (UI). Ці елементи мають безпосередній вплив на ігрове сприйняття, комфорт гравця та загальну естетику проєкту.

Камера. У проєкті використовується стандартна Main Camera (рис. 2.3), розміщена у тривимірному просторі на координатах (X: 9, Y: 4, Z: -15) з орієнтацією по осі Z. Камера має перспективну проєкцію з вертикальним полем зору 40 одиниць, що дозволяє створити ефект глибини сцени.

Параметри камери:

- Projection: Perspective;
- Field of View: 40;
- Clipping Planes: Near – 0.3, Far – 1000;
- Audio Listener: активований для обробки звукового супроводу;

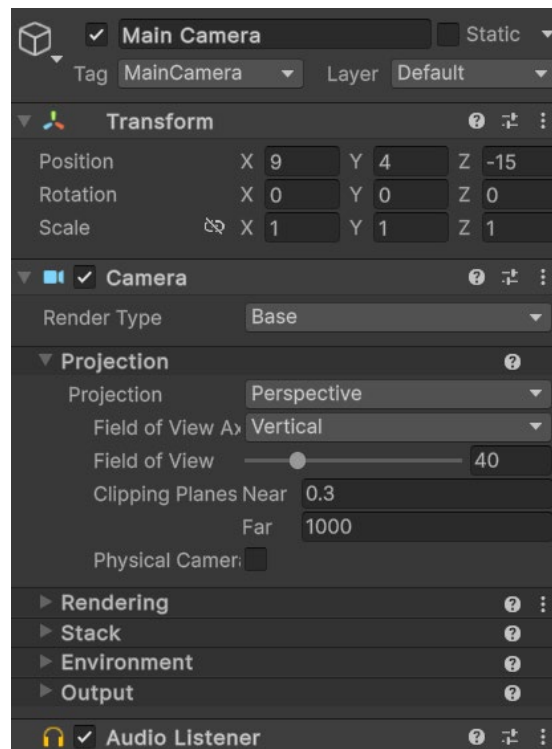


Рис. 2.3. Параметри камери

Освітлення. Для освітлення сцени використовується джерело світла типу Directional Light (рис. 2.4), що імітує сонячне світло. Джерело розташовано на висоті (Y: 20) з поворотом по осях (X: 50, Y: -30), що дозволяє створити м'яке природне освітлення з реалістичними тінями.

Основні налаштування:

- Type: Directional
- Mode: Mixed
- Intensity: 1
- Shadow Type: Soft Shadows

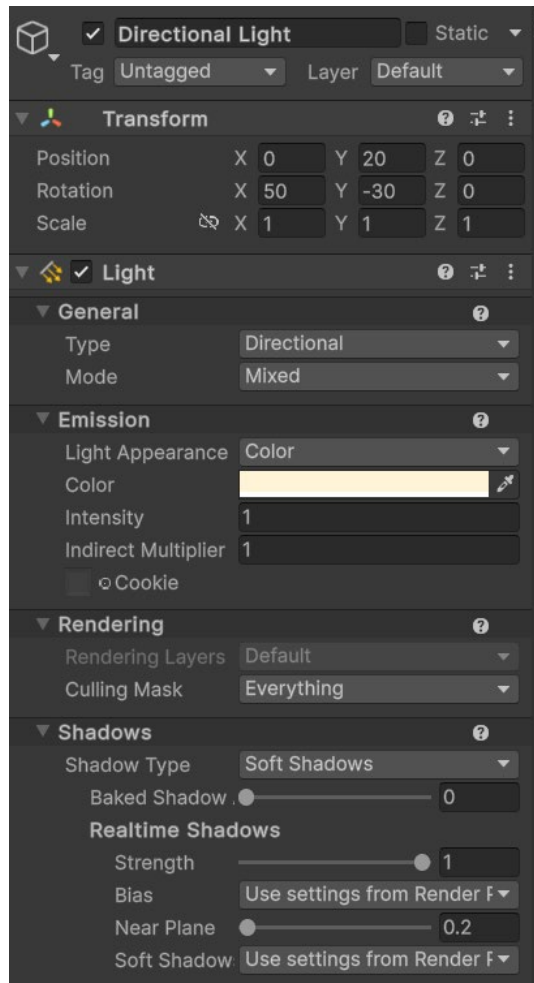


Рис. 2.4. Параметри освітлення

UI (Користувацький інтерфейс). Інтерфейс користувача реалізований у вигляді ієрархії об'єктів у компоненті Canvas. Завдяки використанню панелей, кнопок та текстових полів, UI забезпечує інтуїтивне керування грою та взаємодію користувача з нею. Нижче наведено основні складові інтерфейсу:

- а) RestartPanel – панель, яка з'являється після завершення гри при зіткненні гравця з перешкодами (рис. 2.5):
 - 1) RestartText – текст із відповідним повідомленням;
 - 2) RestartButton – кнопка для перезапуску рівня;
 - 3) ExitToMenuButton1 – кнопка виходу в головне меню.



Рис. 2.5. Панель програшу

b) `LevelCompletePanel` – панель, що з'являється після успішного завершення рівня (рис. 2.6):

- 1) `LevelCompleteText` – повідомлення про завершення;
- 2) `RestartButton2` – ще одна кнопка для повторного проходження;
- 3) `NextLevelButton` – перехід до наступного рівня;
- 4) `ExitToMenuButton` – вихід до меню.



Рис. 2.6. Панель перемоги

- c) StartPanel – головна панель запуску гри (рис. 2.7), містить:
- 1) StartText – назва гри;
 - 2) StartButton – кнопка старту гри;
 - 3) LevelSelectButton – кнопка переходу до вибору рівнів.



Рис. 2.7. Головна панель

- d) LevelSelectPanel – панель вибору рівнів (рис. 2.8), включає:
- 1) LevelSelectText – заголовок панелі;
 - 2) Level1Button, Level2Button, Level3Button – кнопки вибору рівня.

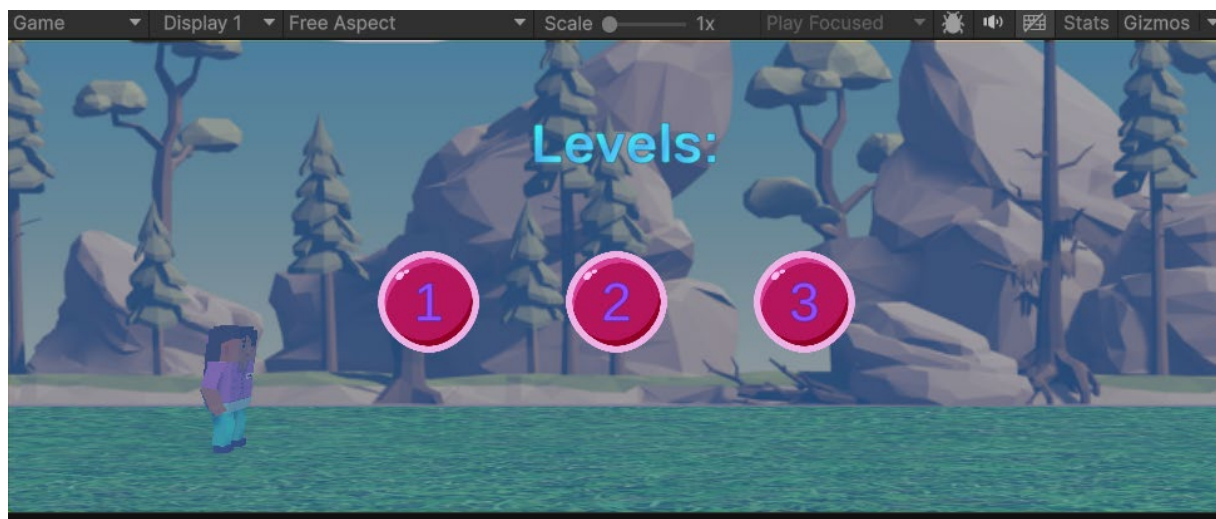


Рис. 2.8. Панель вибору рівнів

- е) `PressEnterText` – текстове поле з підказкою натискання клавіші для початку гри.
- ф) `LevelText` – індикатор поточного рівня.
- г) `PauseButton` – кнопка паузи.
- h) `PauseMenuPanel` – панель, яка з'являється під час паузи:
 - 1) `ResumeButton` – кнопка для продовження гри.

Весь інтерфейс реалізовано з використанням стандартних елементів UI Unity, що дозволяє легко налаштовувати стиль, поведінку та анімації елементів.

2.4 Сценарій гри та програмна реалізація

2.4.1 Основні скрипти (рух гравця, спавн перешкод, фінішу тощо)

У рамках реалізації основного функціоналу гри було створено кілька ключових скриптів, що забезпечують базову логіку геймплею: контроль руху персонажа, генерацію перешкод, виявлення колізій, відображення фінішної точки тощо. Основні компоненти реалізовано на мові програмування C# у середовищі Unity.

Скрипт управління персонажем (Player). Основний скрипт гравця (рис. 2.9) відповідає за стрибок, перевірку стану гри, взаємодію з фінішом, а також відтворення звукових ефектів.

У методі Start() виконується ініціалізація компонентів Rigidbody та AudioSource, а також модифікація сили гравітації для забезпечення більш реалістичної фізики.

У методі Update() реалізовано логіку стрибка, яка активується після натискання пробілу (Space) лише за умови, що гра почалась (`gameStarted == true`) і персонаж знаходиться на землі (`isOnGr == true`). Після стрибка програвється звуковий ефект.

При вході у тригер з тегом Finish, тобто при досягненні фінішу, гра завершується, запускається переможний звук, зупиняється фонове звучання та з деякою затримкою викликається метод `TriggerVictory()`.

```

18  ✓ void Start()
19  | {
20  |     rb = GetComponent<Rigidbody>();
21  |     audioSource = GetComponent<AudioSource>();
22  |     Physics.gravity = Physics.gravity * gravityMod;
23  | }
24  |
25  ✓ void Update()
26  | {
27  |     if(!gm.gameStarted || gameOver)
28  |     {
29  |         return;
30  |     }
31  |     if (Input.GetKeyDown(KeyCode.Space) && isOnGr == true)
32  |     {
33  |         rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
34  |         isOnGr = false;
35  |         if (audioSource != null && jumpSound != null)
36  |         {
37  |             audioSource.PlayOneShot(jumpSound);
38  |         }
39  |     }
40  | }
41  |
42  ✓ void OnTriggerEnter(Collider other)
43  | {
44  |     if (other.CompareTag("Finish"))
45  |     {
46  |         if (audioSource != null && victorySound != null)
47  |         {
48  |             audioSource.PlayOneShot(victorySound);
49  |         }
50  |         gameOver = true;
51  |         if (AudioManager.Instance != null)
52  |         {
53  |             AudioManager.Instance.StopMusic();
54  |         }
55  |         Invoke(nameof(TriggerVictory), 0.7f);
56  |     }
57  | }

```

Рис. 2.9. Частина скрипта Player.cs

Генерація перешкод. Ключову роль у реалізації механіки гри відіграє скрипт `Spawn.cs`, який відповідає за генерацію перешкод та фінішної лінії. Його логіка побудована на автоматичному створенні об'єктів з певним інтервалом після старту рівня. Після того, як гра починається (`gameStarted == true`), запускається метод `SpawnObstacle()` (рис. 2.10) з інтервалом у 2 секунди.


```

void SpawnObstacle()
{
    if (isGameOver) return;
    GameObject[] currentObstacleArray = level1Obstacles;
    int sceneIndex = SceneManager.GetActiveScene().buildIndex;
    if (sceneIndex == 1)
    {
        currentObstacleArray = level2Obstacles;
    }
    else if (sceneIndex == 2)
    {
        currentObstacleArray = level3Obstacles;
    }
    int randomIndex = Random.Range(0, currentObstacleArray.Length);
    GameObject selectedObstacle = currentObstacleArray[randomIndex];
    GameObject spawnedObstacle = Instantiate(selectedObstacle, spawnPosition, selectedObstacle.transform.rotation);
    lastSpawnedObstaclePos = spawnedObstacle.transform.position;
    obstacleCount++;
}

```

Рис. 2.10. Метод SpawnObstacle() скрипта Spawn.cs

У методі SetSpawnPositionByLevel() (рис. 2.11) враховується поточний рівень (визначається через індекс сцени), і встановлюється відповідна координата для появи перешкод. Це дозволяє адаптувати геймплей до змін сцени.

```

void SetSpawnPositionByLevel()
{
    int sceneIndex = SceneManager.GetActiveScene().buildIndex;
    if (sceneIndex == 0)
    {
        spawnPosition = new Vector3(17, 0, -0.4f);
    }
    else if (sceneIndex == 1)
    {
        spawnPosition = new Vector3(19, 0, -0.4f);
    }
    else if (sceneIndex == 2)
    {
        spawnPosition = new Vector3(21, 0.9f, -4f);
    }
}

```

Рис. 2.11. Метод SetSpawnPositionByLevel() скрипта Spawn.cs

Метод SpawnObstacle() вибирає випадкову перешкоду з масиву, що відповідає поточному рівню, і створює її на сцені в зазначеній позиції. Кожна

Таким чином, при завершенні поточного рівня гравець може натиснути кнопку Next Level, і відбудеться завантаження наступної сцени (наступного рівня) за її індексом у Build Settings. Для кожного рівня цей індекс визначається під час розробки:

- рівень 1 – індекс 0
- рівень 2 – індекс 1
- рівень 3 – індекс 2

Крім того, у меню вибору рівня доступні кнопки Level1Button, Level2Button, Level3Button, які безпосередньо викликають методи LoadLevel1(), LoadLevel2() та LoadLevel3() відповідно (рис. 2.14). Це забезпечує можливість ручного вибору рівня.

```

public void LoadLevel1()
{
    SceneManager.LoadScene(0);
    gameStarted = false;
    isRestarted = true;
}

public void LoadLevel2()
{
    SceneManager.LoadScene(1);
    gameStarted = false;
    isRestarted = true;
}

public void LoadLevel3()
{
    SceneManager.LoadScene(2);
    gameStarted = false;
    isRestarted = true;
}

```

Рис. 2.14. Методи виклику рівнів скрипта gm.cs

Ці переходи доповнюються візуальними ефектами на початку рівня методом ShowLevelText(), який показує номер поточного рівня, плавно змінюючи прозорість тексту (рис. 2.15).

```

public IEnumerator ShowLevelText()
{
    int currentLevel = SceneManager.GetActiveScene().buildIndex + 1;
    levelText.text = "Level " + currentLevel;
    levelText.gameObject.SetActive(true);
    Color color = levelText.color;
    color.a = 0;
    levelText.color = color;
    float duration = 1f;
    float t = 0;
    while (t < duration)
    {
        t += Time.deltaTime;
        color.a = Mathf.Lerp(0, 1, t / duration);
        levelText.color = color;
        yield return null;
    }
    yield return new WaitForSeconds(3f);
    t = 0;
    while (t < duration)
    {
        t += Time.deltaTime;
        color.a = Mathf.Lerp(1, 0, t / duration);
        levelText.color = color;
        yield return null;
    }
    levelText.gameObject.SetActive(false);
}

```

Рис. 2.15. Метод ShowLevelText() скрипта gm.cs

2.4.3 Звукове оформлення

Звукове оформлення є важливою складовою геймплею, що створює емоційну атмосферу гри, забезпечує зворотний зв'язок для гравця та підсилює ефект занурення у віртуальний світ. У розробленій грі реалізовано кілька типів звукових ефектів і фоновой музики, які активуються у відповідні моменти геймплею.

Відтворення кнопкових звуків. Всі інтерфейсні кнопки супроводжуються звуковим кліком. Це реалізовано через метод PlayButtonSound() у скрипті gm.cs, який відтворює аудіокліп buttonClickSound через компонент AudioSource.

Для уніфікованого підходу використовується обгортка PlayButtonSoundAndInvoke(), яка спочатку запускає звук кнопки, а потім – викликає потрібний метод із затримкою (рис. 2.16).

```

public void PlayButtonSound()
{
    if (audioSource != null && buttonClickSound != null)
    {
        audioSource.PlayOneShot(buttonClickSound);
    }
}

void PlayButtonSoundAndInvoke(string methodName)
{
    PlayButtonSound();
    Invoke(methodName, 0.3f);
}

```

Рис. 2.16. Частина скрипта gm.cs

Звук початку гри. Після натискання клавіші Enter для початку гри відтворюється звук enterSound. Це реалізовано методом PlayEnterSoundThenStart(), який чекає завершення звуку перед початком ігрового процесу (рис. 2.17).

```

private void PlayEnterSoundThenStart()
{
    if (enterSound != null && audioSource != null)
    {
        audioSource.PlayOneShot(enterSound);
        float delay = enterSound.length;
        Invoke(nameof(StartGame), delay);
    }
    else
    {
        StartGame();
    }
}

```

Рис. 2.17. Метод PlayEnterSoundThenStart() скрипта gm.cs

Звук стрибка. Звук стрибка виконує важливу роль у відчутті фізики руху. Його реалізовано в скрипті гравця Player.cs. При натисканні клавіші стрибка "Space" запускається звук jumpSound.

Звук зіткнення з перешкодами. У грі реалізовано звуковий ефект, який супроводжує момент зіткнення гравця з перешкодами. Коли гравець

стикається з об'єктом з тегом "obstacle", виконується програвання звуку зіткнення (рис. 2.18).

```

else if (collision.gameObject.CompareTag("obstacle"))
{
    if (audioSource != null && collisionSound != null)
    {
        audioSource.PlayOneShot(collisionSound);
    }
    gameOver = true;
    if (AudioManager.Instance != null)
    {
        AudioManager.Instance.StopMusic();
    }
    FindAnyObjectByType<spawn>().StopSpawning();
    Invoke(nameof(TriggerGameOver), 0.4f);
}

```

Рис. 2.18. Частина скрипта Player.cs

Фонова музика рівнів. Для кожного рівня передбачено окрему музичну композицію, яка автоматично відтворюється при завантаженні сцени. Це реалізовано у класі AudioManager.cs (рис. 2.19), який є синглтоном і зберігається між сценами завдяки методу DontDestroyOnLoad. Музика автоматично вмикається при запуску гри методом PlayMusicForLevel, що викликається з коду gm.cs. У разі завершення гри або зіткнення з перешкодою музика припиняється через StopMusic().

```

void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
        audioSource = GetComponent();
    }
    else
    {
        Destroy(gameObject);
        return;
    }
    audioSource = GetComponent();
}

public void PlayMusicForLevel(int levelIndex)
{
    if (audioSource != null && levelMusic != null && levelIndex < levelMusic.Length)
    {
        audioSource.clip = levelMusic[levelIndex];
        audioSource.Play();
    }
}

public void StopMusic()
{
    if (audioSource != null)
        audioSource.Stop();
}

```

Рис 2.19. Частина скрипта AudioManager.cs

2.5 Висновки до розділу 2

У другому розділі було здійснено всебічне проєктування 3D гри на основі середовища Unity. Було визначено основну ідею гри, описано механіку, структуру рівнів та інтерфейс користувача. Розроблено сценарій гри з реалізацією ключових скриптів, включно з рухом гравця, обробкою колізій, взаємодією з перешкодами та системою перемоги.

Особливу увагу приділено елементам взаємодії гравця з ігровим середовищем, а також створенню зручного меню та логіки переходу між рівнями. Також реалізовано повноцінне звукове оформлення, що включає фонову музику, звуки дій і подій, що підсилюють атмосферу гри.

У результаті розроблено цілісну ігрову логіку, яка може бути масштабована та вдосконалена на наступних етапах реалізації проєкту.

РОЗДІЛ 3. ТЕСТУВАННЯ, ОЦІНКА ТА ПЕРСПЕКТИВИ РОЗВИТКУ ГРИ

3.1 Тестування працездатності гри

3.1.1 Методика проведення тестування

Тестування гри є важливим етапом завершення розробки, що дозволяє переконатися у її стабільній роботі та виявити можливі помилки до її релізу. У процесі створення 3D-гри в середовищі Unity було застосовано ручне тестування, яке здійснювалося на різних етапах проєктування і програмування.

Основна мета тестування полягала у перевірці працездатності основних компонентів гри: управління гравцем, взаємодії з об'єктами, переходів між рівнями, роботи UI-елементів (меню, кнопки, панелі), звукового супроводу та обробки ситуацій завершення гри.

Тестування проводилося на ПК із середніми технічними характеристиками, що відповідають вимогам більшості користувачів. Було перевірено поведінку гри під час різних сценаріїв:

- запуску та завершення гри;
- перемоги або поразки на рівні;
- переходу до наступного рівня;
- перезапуску гри;
- обробки клавіш керування та натискання кнопок UI;
- взаємодії з перешкодами та фінішними об'єктами.

Також проводилося тестування звукових ефектів: відтворення звуку стрибка, зіткнення, досягнення фінішу та фонові музики. Особлива увага

приділялася перевірці логіки вмикання і вимикання звуків та їхньої відповідності подіям у грі.

Методика включала тестування кожного рівня окремо, а також поведінку гри після її перезапуску або завершення. Усі зафіксовані помилки та неточності були задокументовані та виправлені на етапі завершення проєкту.

3.1.2 Випробування функціональних елементів

У межах тестування функціональності було здійснено перевірку ключових компонентів гри, які забезпечують її ігровий процес. Основна увага приділялася елементам, що безпосередньо впливають на взаємодію гравця із середовищем, логіку ігрових подій та реагування на дії користувача.

Рух гравця. Було протестовано рух гравця вперед, стрибок та приземлення. Перевірялася стабільність роботи скриптів, що забезпечують плавність руху та обмеження подвійного стрибка. Також було перевірено, чи реагує гравець на натискання клавіш лише тоді, коли він перебуває на землі (тобто працює логіка `isOnGround`).

Колізії з перешкодами. Функціонально перевірено, що при зіткненні з об'єктами, які мають тег `obstacle`, спрацьовує відповідний звуковий ефект, гра зупиняється, запускається функція `TriggerGameOver`, а генерація нових перешкод припиняється.

Досягнення фінішу. Було протестовано логіку досягнення гравцем фінішного об'єкта. Перевірено:

- появу панелі з написом "Victory";
- відтворення відповідного звуку;
- доступність кнопок "Level again", "Next Level" і "Exit to menu";
- правильність переходу до наступного рівня.

Система меню. Було протестовано головне меню гри, кнопки початку гри, вибору рівня, перезапуску, повернення до меню. Всі кнопки спрацьовують коректно, викликаючи відповідні дії без помилок. Також перевірено, що після повернення до меню фонову музику можна перезапустити без збоїв.

Звукове оформлення. Окремо перевірялися звуки:

- натискання кнопок;
- стрибка;
- зіткнення з перешкодою;
- досягнення фінішу;
- фонової музики для кожного рівня.

Звуки відтворювалися відповідно до подій. Була також протестована система керування музикою за допомогою скрипта AudioManager, який гарантує правильне відтворення треків згідно з поточним рівнем гри.

Генерація перешкод. Під час гри перевірено механізм поступової генерації перешкод та їхню швидкість відповідно до рівня. На кожному новому рівні перешкоди рухаються швидше, що свідчить про правильну реалізацію складності.

Усі елементи гри були перевірені на працездатність, і виявлені незначні помилки були усунуті під час процесу налагодження.

3.1.3 Виявлені помилки та їх усунення

Під час процесу тестування гри були виявлені кілька функціональних та логічних помилок, які могли впливати на якість ігрового досвіду.

а) подвійний стрибок при натисканні клавіші в повітрі:

- 1) Опис: гравець міг виконати повторний стрибок, навіть перебуваючи в повітрі.

- 2) Усунення: було додано перевірку логічної змінної `isOnGround`, яка дозволяє стрибок лише тоді, коли гравець перебуває на землі.
- b) відсутність звуку при зіткненні з перешкодою:
- 1) Опис: звуковий ефект зіткнення не відтворювався.
 - 2) Усунення: перевірено, що компонент `AudioSource` прив'язано до об'єкта гравця, а також що в полі `collisionSound` призначено відповідний звуковий файл. Додано перевірку `if (audioSource != null && collisionSound != null)` у метод `OnCollisionEnter`.
- c) музика рівня не змінювалася при переході до нового рівня:
- 1) Опис: після проходження рівня та переходу до наступного відтворювалася стара музика.
 - 2) Усунення: було оновлено логіку у скрипті `AudioManager`, щоб при завантаженні нового рівня викликався метод `PlayMusicForLevel(levelIndex)`, де `levelIndex` відповідає новому рівню.
- d) фінальний екран не з'являвся після досягнення фінішу:
- 1) Опис: у деяких випадках не відображалось вікно з написом "Victory".
 - 2) Усунення: було виправлено логіку перевірки тегу об'єкта `Finish`, а також переконанося, що UI-панель активується через `SetActive(true)` після події.
- e) повільне оновлення рівня після натискання кнопки "Next Level":
- 1) Опис: після переходу до наступного рівня гра одразу починалася без очікування натискання клавіші `Enter`.
 - 2) Усунення: у скрипті, відповідальному за запуск гри, було додано перевірку `isGameStarted`, яка активується лише після натискання `Enter`, незалежно від номеру рівня.

Завдяки системному тестуванню та оперативному виправленню помилок вдалося забезпечити стабільну роботу гри на всіх етапах проходження.

3.2 Аналіз якості гри з точки зору користувача

Якість гри є ключовим чинником, який впливає на успішність проекту серед гравців. Відповідно до основних критеріїв оцінки, до уваги беруться такі аспекти: зручність управління, стабільність роботи, візуальна привабливість, звукове оформлення та загальне задоволення від ігрового процесу.

Попри відсутність зовнішнього тестування реальними користувачами, гру було проаналізовано з урахуванням основних принципів юзабіліті, геймдизайну та загальних очікувань середньостатистичного гравця. Такий самостійний аналіз дозволяє оцінити потенційну якість гри з точки зору майбутніх користувачів.

Зручність користування та інтерфейс. Гра реалізована з фокусом на простоту керування та зрозумілий інтерфейс. Всі основні дії – стрибок, уникання перешкод, перехід між рівнями – виконуються інтуїтивно зрозумілими клавішами. Застосування таких звичних елементів, як меню з кнопками "Start Game" (Почати гру), "Next Level" (Наступний рівень), "Play again" (Грати знову), сприяє зручності користування навіть без інструкцій.

Крім того, перед початком гри гравцеві пропонується натиснути клавішу Enter, що дозволяє підготуватися до старту. Це свідчить про продуманий UX (користувацький досвід). Інтерфейс не перевантажений зайвими елементами, що сприяє зосередженню на геймплеї.

Візуальний стиль, звуковий супровід та ігровий процес. Візуальне оформлення гри просте, але достатньо виразне: всі елементи чітко

відокремлені один від одного, що полегшує орієнтацію у просторі. Анімації, як-от рух гравця, з'явлення перешкод, зіткнення або досягнення фінішу, додають динаміки й ефектності.

Особливу роль відіграє звуковий супровід. Завдяки вбудованим аудіоефектам, гравець отримує миттєвий зворотний зв'язок на свої дії: звук стрибка, зіткнення з перешкодою, досягнення фінішу – все це створює відчуття занурення у гру. Окрема фоновіа музика для кожного рівня не лише додає емоційного фону, а й підтримує загальний темп гри.

Також слід відзначити збалансованість ігрового процесу. Рівні поступово ускладнюються, зокрема за рахунок збільшення швидкості перешкод. Це стимулює гравця до вдосконалення своїх навичок і не дає грі стати одноманітною.

Планування користувацького тестування. У майбутньому планується організувати тестування гри серед потенційних користувачів. Це дозволить зібрати зворотний зв'язок стосовно якості геймплею, привабливості дизайну, зручності інтерфейсу та загального враження від гри. За результатами тестування можна буде сформуувати список покращень для наступних версій проєкту.

Попередні висновки щодо якості. На основі самостійного аналізу можна стверджувати, що гра функціонує стабільно, має зрозуміле управління та відповідає базовим вимогам до сучасного 3D-проєкту. Утім, остаточна оцінка якості потребує залучення ширшого кола гравців і врахування їхніх вражень.

3.3 Можливості покращення та подальшого розвитку

Хоча базова версія гри вже функціональна і містить основні механіки, існує чимало напрямів для подальшого вдосконалення як у технічному, так і в

ігровому плані. Покращення можуть підвищити привабливість гри, розширити її функціонал і зробити досвід гравця ще цікавішим.

Одним з основних напрямів розвитку є ускладнення геймплею. До гри можна додати нові типи перешкод із різною поведінкою (наприклад, анімовані). Також можна ввести нові можливості для гравця: подвійний стрибок, тимчасову невразливість, підбір бонусів або впровадити систему життів. Це зробить гру більш захопливою та різноманітною.

Ще одним варіантом розвитку є створення додаткових рівнів із різними візуальними стилями, новими фонами та музичними темами. Це збагатить гру та зробить її більш динамічною і цікавою для гравця.

Іншою перспективною можливістю є реалізація системи досягнень. Додавання таблиці рекордів, системи балів або досягнень може суттєво підвищити мотивацію гравця. Наприклад, гравці можуть змагатися за найкращий результат або відкривати нові можливості за виконання певних умов (кількість рівнів без поразки, збирання бонусів тощо). Крім того, можна реалізувати систему збереження прогресу.

Одним із важливих аспектів, що може покращити користувацький досвід, є додавання можливості вибору шкінів персонажа. Гравець зможе обирати зовнішній вигляд героя відповідно до своїх уподобань. Це можна реалізувати через внутрішню галерею або магазин, де шкіни відкриваються за досягнення або накопичені бали. Така функція не тільки урізноманітнить візуальний стиль гри, а й сприятиме підвищенню мотивації користувача до подальшого проходження.

Значну увагу можна приділити покращенню графіки, візуальних ефектів і анімації персонажа. Застосування більш деталізованих 3D-моделей, тіней і освітлення зробить візуальну частину гри сучаснішою та привабливішою.

У перспективі гра може бути адаптована для мобільних пристроїв або вебплатформ. Це вимагатиме адаптації керування під сенсорні екрани або браузерне середовище, але значно розширить аудиторію користувачів. Також можна розглянути багатомовну локалізацію гри, щоб зробити її доступною для гравців із різних країн.

Таким чином, можливості для розвитку проєкту залишаються відкритими, і в разі подальшої роботи над грою вона має всі шанси стати більш повноцінним, конкурентоспроможним ігровим продуктом.

3.4 Висновки до розділу 3

У третьому розділі було проведено тестування та аналіз працездатності реалізованої гри. Визначено, що всі основні функціональні елементи працюють коректно: гравець може переміщуватись, уникати перешкод, проходити рівні та взаємодіяти з ігровим середовищем згідно із закладеною логікою. Також перевірено коректність роботи звукового оформлення, анімацій та системи переходів між рівнями.

Попри те, що користувацьке тестування у широкому масштабі не проводилось, було здійснено оцінку якості гри на основі загальноприйнятих критеріїв, таких як зручність керування, чіткість візуального оформлення, відповідність жанру та загальна ігрова динаміка. Виявлено потенціал для подальшого вдосконалення проєкту.

У підпункті 3.3 окреслено можливості майбутнього розвитку гри, зокрема додавання нових рівнів, системи досягнень, варіантів зовнішнього вигляду персонажа (скінів), поліпшення графіки, адаптації до інших платформ та багатомовної підтримки. Таким чином, розроблений проєкт має перспективу для подальшого розширення та вдосконалення з урахуванням потреб і побажань потенційних користувачів.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було реалізовано повний цикл розробки тривимірної гри в середовищі Unity. Спочатку проведено аналіз предметної області, досліджено історію розвитку ігрової індустрії, розглянуто основи геймдизайну, особливості створення 3D-ігор, а також проаналізовано можливості рушія Unity для реалізації поставленого завдання.

На етапі проєктування було визначено концепцію гри, розроблено її основні механіки, запропоновано сценарій та здійснено програмну реалізацію. Основну увагу приділено розробці системи руху гравця, взаємодії з перешкодами, обробці подій зіткнення, переходам між рівнями, а також звуковому оформленню, що сприяє створенню більшої залученості гравця у процес.

У ході тестування було підтверджено працездатність основних ігрових механік і визначено напрями можливого вдосконалення проєкту. Зокрема, запропоновано додати більше варіацій рівнів, розширити функціонал через систему досягнень, реалізувати вибір шкінів для персонажа, покращити графічне оформлення та адаптувати гру для різних платформ.

Таким чином, поставлені у роботі завдання були виконані повністю, а розроблена 3D-гра є функціональним і перспективним проєктом, який може стати основою для подальшого розвитку й удосконалення відповідно до вимог сучасного ігрового ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unity Technologies. Unity User Manual [Електронний ресурс]. – Режим доступу: <https://docs.unity3d.com/Manual/index.html>
2. Unity Technologies. Scripting API [Електронний ресурс]. – Режим доступу: <https://docs.unity3d.com/ScriptReference/>
3. Rollings A., Adams E. Andrew Rollings and Ernest Adams on Game Design. – New Riders, 2003. – 650 p.
4. Schell J. The Art of Game Design: A Book of Lenses. – 3rd ed. – CRC Press, 2019. – 600 p.
5. Unreal Engine Documentation [Електронний ресурс]. – Режим доступу: <https://docs.unrealengine.com/>
6. Freeman A. Pro Unity Game Development with C#. – Apress, 2021. – 733 p.
7. Бондаренко О.М., Бондаренко С.О. Основи програмування на мові C# у середовищі Unity. – Київ: Кондор, 2021. – 224 с.
8. Калашніков С. Геймдизайн: Основи створення ігор. – Харків: Фоліо, 2020. – 288 с.
9. Ігрові рушії: порівняльний аналіз [Електронний ресурс]. – Режим доступу: <https://gamedevelopment.tutsplus.com/articles/the-top-game-engines-gamedev-61>
10. Bishop J. Unity 2021 By Example: Learn about game and virtual reality development by creating five engaging projects. – Packt Publishing, 2021. – 798 p.
11. Мельник В.В. Програмування ігор: теорія і практика. – Львів: ЛНУ ім. Івана Франка, 2020. – 312 с.
12. Vernick M. Learn Unity Programming with C#. – Independently Published, 2020. – 450 p.
13. Офіційний форум Unity [Електронний ресурс]. – Режим доступу: <https://forum.unity.com/>

14. Rabin S. Game AI Pro 3: Collected Wisdom of Game AI Professionals. – CRC Press, 2017. – 617 p.
15. Zyda M. From Visual Simulation to Virtual Reality to Games. – Computer, Vol. 38, No. 9, IEEE Computer Society, 2005. – P. 25-32.