БАКАЛАВРСЬКА РОБОТА

БР. ІП - 94.00.00.000 ПЗ

Група ІП-21-4

Возняк Владислав

2025

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Возняк Владислав Романович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка ігор з використанням Unity
(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____Возняк Владислав Романович_____
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____Бандура Вікторія Валеріївна, доцент_____
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

_доц._____Бандура В.В.__
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доцент. В.В. Бандура
“ ____ ” _____202 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Возняк Владислав Романович

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) " Розробка ігор з використанням Unity "

керівник проекту (роботи) к.т.н., доцент Бандура Вікторія Валеріївна

затверджені наказом вищого навчального закладу від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1 Загальні принципи та інструменти розробки ігор

2 Засоби та методи розробки ігор

3 Unity-ігра: вплив на користувачів та її аналіз

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Знімок екрана сцени в Unity з рівнем від Brick Breaker (рис. 1.1:,ст.10)

2. Знімок екрана гри з рівнем від Brick Breaker (рис 1.2, ст. 14).

3. Знімок екрана ієрархії. (рис 1.3, ст. 15).

4. Знімок екрана перегляду проекту зі списком усіх ресурсів для brick breaker (рис. 1.4, ст. 19)

5. Знімок екрана інспектора, що показує властивості об’єкта менеджера гри (рис. 1.5, ст.20).

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання **2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обгрунтування теми роботи	15.02.2025	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	25.02.2025	виконано
3	Побудова моделі або алгоритму власного рішення	15.03.2025	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	25.04.2025	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	10.06.2025	виконано

Студент _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Дипломна робота містить 60 сторінок, 11 рисунки, 1 таблиці, список використаних джерел із 30 найменуваннями.

Об'єкт дослідження: процеси розробки та функціонування комп'ютерних ігор, створених за допомогою Unity, спрямованих на забезпечення якісної взаємодії з користувачами.

Предмет дослідження: методи, інструменти та технології розробки комп'ютерних ігор у Unity, включаючи аналіз принципів і механізмів розробки, порівняння методологій, проектування ігрових систем, створення користувацького інтерфейсу, а також оцінку впливу ігор на користувачів.

Результати дослідження: проведено порівняльний аналіз методологій розробки ігор, розроблено гру в Unity, яка демонструє ефективне використання інструментів і методів, а також проаналізовано її вплив на користувачів.

У першому розділі аналізуються загальні принципи розробки ігор, інструменти взаємодії з об'єктами сцени в Unity та основні ігрові механізми.

У другому розділі досліджуються інструменти та методи розробки ігор, включаючи їхній короткий опис і порівняння.

У третьому розділі описано розроблену гру в Unity, проаналізовано позитивні та негативні ефекти відеоігор і обговорено результати аналізу.

Висновок: порівняльний аналіз методологій успішно виконано, розроблена гра відповідає сучасним вимогам до комп'ютерних ігор, забезпечуючи якісний користувацький досвід.

КЛЮЧОВІ СЛОВА: КОМП'ЮТЕРНІ ІГРИ, UNITY, МЕТОДОЛОГІЇ РОЗРОБКИ, ІГРОВІ МЕХАНІЗМИ, АДАПТИВНИЙ ДИЗАЙН, КОРИСТУВАЦЬКИЙ ДОСВІД, АНАЛІЗ ВПЛИВУ, ФРОНТЕНД, БЕКЕНД, ТЕСТУВАННЯ.

ABSTRACT

The bachelor's thesis comprises 60 pages, 11 figures, 1 tables, and a reference list with 30 entries.

The aim of the work is to conduct a comparative analysis of methodologies for developing computer games using Unity-class technologies to identify optimal approaches for creating games with a high-quality user experience.

Object of study: the processes of developing and operating computer games created with Unity, aimed at ensuring effective user interaction.

Subject of study: methods, tools, and technologies for developing computer games in Unity, including analysis of development principles and mechanisms, comparison of methodologies, design of gaming systems, creation of user interfaces, and evaluation of games' impact on users.

Research results: a comparative analysis of game development methodologies was conducted, a game was developed in Unity demonstrating effective use of tools and methods, and its impact on users was analyzed.

The first section analyzes general game development principles, tools for interacting with scene objects in Unity, and core game mechanics.

The second section explores game development tools and methods, including their brief description and comparison.

The third section describes the game developed in Unity, analyzes the positive and negative effects of video games, and discusses the analysis results.

Conclusion: the comparative analysis of methodologies was successfully completed, and the developed game meets modern requirements for computer games, ensuring a high-quality user experience.

KEYWORDS: COMPUTER GAMES, UNITY, DEVELOPMENT METHODOLOGIES, GAME MECHANICS, ADAPTIVE DESIGN, USER EXPERIENCE, IMPACT ANALYSIS, FRONTEND, BACKEND, TESTING.

ЗМІСТ

ВСТУП

У сучасному цифровому світі комп’ютерні ігри стали не лише популярною формою розваг, але й потужним інструментом для навчання, соціальної взаємодії та економічної діяльності, що зумовлює зростання попиту на створення якісних, захоплюючих і технічно досконалих ігрових продуктів. Unity, як одна з провідних платформ для розробки ігор, забезпечує гнучкість, широкий набір інструментів і підтримку різноманітних жанрів, що робить її оптимальним вибором для створення сучасних ігор. Використання різних методологій розробки, таких як Agile, Waterfall чи їх гібриди, у поєднанні з технологіями Unity, відкриває нові можливості для оптимізації процесу створення ігор, підвищення їхньої якості та забезпечення позитивного впливу на користувачів.

Актуальність теми обумовлена швидким розвитком ігрової індустрії та потребою в порівняльному аналізі методологій розробки, які дозволяють ефективно використовувати можливості Unity для створення ігор із високим рівнем користувацького досвіду. Існуючі підходи до розробки ігор часто стикаються з викликами, пов’язаними з управлінням складністю проєктів, оптимізацією ресурсів і адаптацією до потреб аудиторії, що підкреслює необхідність дослідження оптимальних методологій.

Метою роботи є порівняльний аналіз методологій розробки комп’ютерних ігор з використанням технологій Unity класу для визначення найефективніших підходів до створення ігор.

Завданнями дослідження є вивчення загальних принципів та інструментів розробки ігор у Unity, аналіз засобів і методів створення ігрових проєктів, розробка гри з використанням Unity, а також оцінка її впливу на користувачів через аналіз позитивних і негативних ефектів.

Об’єктом дослідження є процеси розробки програмного забезпечення для створення комп’ютерних ігор, що охоплюють аналіз вимог, проектування, реалізацію, тестування та оцінку впливу ігрових продуктів.

Предметом дослідження є методи, технології та інструменти, що застосовуються під час розробки ігор у Unity, зокрема принципи взаємодії з об’єктами сцени, ігрові механізми, методології розробки, а також методи аналізу впливу ігор на користувачів.

Для досягнення мети використано **методи** аналізу літературних джерел, порівняльного аналізу методологій, моделювання ігрових систем, емпіричного програмування, тестування розробленої гри, а також оцінки користувацького досвіду та впливу. Розроблена гра в Unity передбачає використання сучасних ігрових механізмів, адаптивного дизайну та оптимізованих інструментів для створення захоплюючого ігрового досвіду, а також аналіз її ефектів на користувачів, що сприяє кращому розумінню ефективності обраних методологій.

Очікується, що результати дослідження сприятимуть формуванню рекомендацій щодо вибору оптимальних методологій розробки ігор, покращенню якості ігрових продуктів і підвищенню їхнього позитивного впливу на аудиторію.

Бакалаврська робота містить 60 сторінок, 11 рисунки, 1 таблиці, список використаних джерел із 30 найменуваннями.

РОЗДІЛ 1. ЗАГАЛЬНІ ПРИНЦИПИ ТА ІНСТРУМЕНТИ РОЗРОБКИ ІГОР

1.1 Мета та сфера застосування

У цьому дослідженні розглядаються основні інструменти та техніки розробки ігор. Ігровий бізнес постійно розвивається завдяки значному прогресу в інструментах і технології [5]. Метою дослідження є визначення основних інструментів і технік, що використовуються в розробці ігор, а також їх різниця між іграми та етапами розробки. Основне питання дослідження: "Які основні інструменти та методи розробки ігор?" Підзапитання, які підтверджують це основне запитання, такі:

1. Які ігрові движки широко використовуються в розробці ігор? У цьому дослідженні ігрових движків досліджуватимуться їхні характеристики, функціональність і популярність.

2. Чи впливають різні ігрові жанри, платформи та етапи на інструменти та методи, що використовуються під час розробки гри? Дослідження досліджуватиме, чи різні ігри, такі як бойовики, пригодницькі та симулятори, використовують різні інструменти та методи у своєму створенні.

Ці підзапитання спрямовані на те, щоб краще зрозуміти, як інструменти та техніки використовуються для створення ігор у різних іграх. Крім того, мета полягає в тому, щоб зрозуміти, як ігровий жанр, платформа та етап розробки впливають на інструменти та методи, які використовуються та вибираються під час створення гри.

Створення ігор — це індустрія, яка швидко розвивається, і в ній є технології, які забезпечують ефективну співпрацю між дизайнерами, програмістами, дизайнерами та звукорежисерами. Серед цих технологій — такі ігрові движки, як Unity та Unreal Engine, які забезпечують потужні платформи для виробництва ігор. Редактори коду, такі як Visual Studio та інші, які допомагають писати та налагоджувати код, у той час як графічні компоненти можна створювати за допомогою Blender і Adobe Photoshop, а захоплююче аудіо

за допомогою FL Studio і Audacity.

Крім того, основні навички виробництва ігор, такі як концепції дизайну ігор, процеси програмування, методи анімації та стратегії звукового дизайну. Розуміння цих стратегій має вирішальне значення для створення привабливих і відшліфованих ігор. Нарешті, огляд розглядає етапи розробки гри: передпродакшн (планування та прототипування), виробництво (створення активів і кодування), тестування (виправлення помилок та оптимізація продуктивності) і пост-продакшн (шліфування та підготовка до випуску).

Інструменти розробки ігор

Інструменти розробки ігор включають широкий спектр програмного забезпечення, програм і ресурсів, необхідних для створення гри в процесі створення гри. Ці технології спрощують такі дії, як створення, проектування, впровадження, тестування та оптимізація. Ці інструменти змінюються та еволюціонують у дуже швидкому просторі, і на ринку їх існує велика різноманітність. Не завжди легко вирішити, який інструмент найкраще використовувати в яких ситуаціях, тому важливо розуміти їх можливості.

Ігрові движки є основою створення сучасних ігор. Вони функціонують як повноцінне програмне середовище, яке дозволяє розробникам створювати захоплюючі та цікаві ігрові враження 10 на різних платформах. Зазвичай ігрові движки включають движок візуалізації, який відповідає за анімовані візуальні ефекти. Наприклад, він виявляє зіткнення, керує пам’яттю та допомагає створювати візуальні ефекти для гри. (Джордж і Джордж, 2022) Двома провідними ігровими движками на ринку є Unity та Unreal Engine. Unreal Engine добре відомий своїми високоякісними та реалістичними 3D-візуалами, що робить його ідеальним вибором для розробки величезних і складних ігор. Він використовує C++ для розробки, що може бути більш складним для початку через його складність. З іншого боку, єдність – це добре відомий своєю величезною базою користувачів і сильною підтримкою як 2D, так і 3D ігор. Він особливо зручний для початківців, оскільки в ньому використовується C#, який часто вважається швидшим для вивчення та більш придатним для тих, хто

починає створювати ігри, ніж C++.

Ігровий движок Unity 3D

Unity 3D — це ігровий движок і повне інтегроване середовище розробки (IDE) з інтегрованим редактором, робочим процесом активів, конструктором сцен, сценаріями, мережею тощо. Він також має величезну спільноту та форум, куди будь-яка людина, яка бажає дізнатися та навчитися використовувати Unity, може піти та отримати відповіді на всі свої запитання. Існує п’ять основних видів перегляду, які використовуються в редакторі Unity для виконання всієї роботи: перегляд проекту, перегляд сцени, режим гри, перегляд ієрархії та режим перегляду інспектора, усі вони описані більш детально нижче.

Текстові редактори для розробки коду

Редактори коду забезпечують гарне середовище для створення, редагування та організації коду, що робить їх необхідними інструментами для розробників ігор [7]. Використання цих функцій, які пропонують редактори, може покращити процес розробки та зробити процес більш ефективним (Bentil, 2024). Ці функції включають підсвічування синтаксису, завершення коду, інтеграцію контролю версій та інструменти для налагодження (Johnson, 2019). Одним із найпопулярніших редакторів є Visual Studio, інтегроване середовище програмування (IDE), яке використовують багато розробників. Код Visual Studio доступний на трьох основних платформах Linux, Windows і Mac і надає численні розширення для використання користувачем. Багато розробників віддають перевагу йому через його широкі можливості та зручний інтерфейс.

Графічні інструменти

Графічні інструменти покращують створення візуального аспекту для розробки ігор, пропонуючи розробникам кілька платформ для створення та керування ігровими ресурсами, такими як моделі, текстури, анімація та інтерфейс користувача. Ці інструменти дозволяють розробникам створювати 3D-моделі, малювати текстури, монтувати та переміщати фігури, а також створювати візуальні ефекти, такі як освітлення, тіні та відображення. Blender — це інструмент для створення 3D з відкритим кодом, який має широкий набір

функцій, що робить його популярним вибором серед розробників. Анімація, моделювання, візуалізація, компонування та відстеження руху – це лише деякі з багатьох функцій, для яких можна використовувати. Оскільки це програмне забезпечення з відкритим кодом, його можна модифікувати, і воно має велику спільноту учасників, які підтримують його розробку [15] Adobe Photoshop — це ще одна відома графічна програма, особливо відома своїми можливостями зміни зображень. Він містить повний набір інструментів для генерації текстур, об’єктів та інших 2D-елементів і керування ними (Adobe 2024a). Завдяки широкій функціональності та взаємодії з іншими продуктами Adobe він вважається цінним інструментом для розробників ігор.

Аудіоінструменти

Аудіоінструменти, які включають велику різноманітність інструментів, призначених для запису, редагування, мікшування та мастерингу аудіоресурсів, важливі для створення захоплюючого та унікального аудіопереживання у відеоіграх [9]. За допомогою цих технологій розробники можуть створювати музичні композиції, звукові ефекти навколишнього середовища та динамічні звукові пейзажі, які покращують ігровий процес. Одним із добре відомих аудіоінструментів є FL Studio, яка є цифровою аудіоробочою станцією (DAW), яка має багато функцій, зокрема секвенування, запис, редагування, мікшування та мастеринг, що робить його улюбленим інструментом для багатьох звукових дизайнерів і музичних продюсерів завдяки його ефективності робочого процесу та адаптивності [11]/ Крім того, є Audacity, який є безкоштовним аудіоредактором із відкритим кодом. Він відомий своїм зручним інтерфейсом і широкою підтримкою плагінів. Він надає інструменти для запису та редагування аудіо, а також функції для зменшення шуму, вирівнювання та обробки ефектів. [5].

Тестування програмного забезпечення

Тестування програмного забезпечення є важливим для забезпечення якості, стабільності та продуктивності гри. Інструменти тестування дозволяють розробникам знаходити та виправляти баги, помилки та інші проблеми на різних

платформах і середовищах. Ці інструменти надають широкий спектр функцій тестування, таких як автоматизоване тестування, регресійне тестування, профілювання продуктивності та тестування на сумісність, щоб допомогти з налагодженням та оптимізацією. (Снеха та Малле, 2017) Одним із популярних варіантів програмного забезпечення для тестування є Unity Test Runner, який є вбудованою системою тестування. Він повністю інтегрований в ігровий движок Unity та забезпечує легкі можливості тестування для різних видів проектів [1].

1.2 Інструменти взаємодії з об’єктами сцени в Unity

Перегляд сцени

Вид сцени є одним із найбільш використовуваних видів, оскільки тут розміщуються всі ігрові об’єкти та будуються сцени для гри.

Рисунок 1.1 - Знімок екрана сцени в Unity з рівнем від Brick Breaker.

На рисунку 1.1 показаний вид сцени з рівнем із гри Brick breaker. Рівень розроблений у 3D, хоча грається більше як 2D-гра. "Щоб створити 2D-гру в 3D-середовищі, один ступінь свободи видаляється. Вісь у майже ігнорується, а

камера розміщується в ортографічному режимі, дивлячись на екран вниз, як ніби в неї грають на верхній частині столу". [1] У проекті вигляд було змінено, щоб надати йому відчуття 2D, яке використовується для всіх ігор, створених під час цього проекту. У цьому режимі рівень виглядає зовсім інакше, ніж у грі. Цей вид дозволяє програмісту переміщатися по всьому 3D-світу, в який вбудована гра. Щоб допомогти позиціонувати об’єкти, Unity може прив’язуватися до певних кроків під час перетягування. «Щоб використати прив’язку, утримуйте Команда (Mac) або КОНТРОЛЬ (ПК) під час використання інструмента перекладу (IN), щоб переміщувати об’єкти в режимі перегляду сцени» [2] Можна переглядати всі зони гри; навіть а гравець гри не може дістатися. 3D-світ жодним чином не обмежений, якби об’єкт кинули в будь-якому напрямку на цьому екрані, теоретично він би зник назавжди. Насправді це обмежено лише комп’ютером або пристроєм, на якому запущена гра [3]/

Перегляд гри

Перегляд гри – це те, що користувач побачить під час запуску гри. Для цього вікна є кілька варіантів. У верхній частині вікна є кілька кнопок/випадаючих меню, які можуть змінювати перспективу, повноекранний режим і штуковини, що відображаються у вікні гри.

Рисунок 1.2 - Знімок екрана гри з рівнем від Brick Breaker.

Як видно на рисунку 1.2, персонаж цієї гри з'являється і може запустити м'яч, щоб зруйнувати цегляний замок на відстані. Також можна побачити меню паузи та систему очок. На відміну від перегляду сцени, гравець не може побачити, що існує величезний 3D-світ, який містить цей маленький рівень, на якому вони грають. Коли гра працює, у цьому перегляді відбувається тестування гри. Brick breaker можна запускати, призупиняти та перезапускати, щоб допомогти в процесі тестування. Ще один вид, корисний під час тестування ігор, — це режим консолі (не показано). У цьому поданні можна побачити всі результати гри, а повідомлення про налагодження можна використовувати для виявлення неприємних помилок у коді.

Перегляд ієрархії

У режимі ієрархії всі об'єкти гри можна створювати, отримувати доступ, групувати та керувати ними для створення гри. Коли проект зберігається, об'єкти зберігаються у файлі сцени.

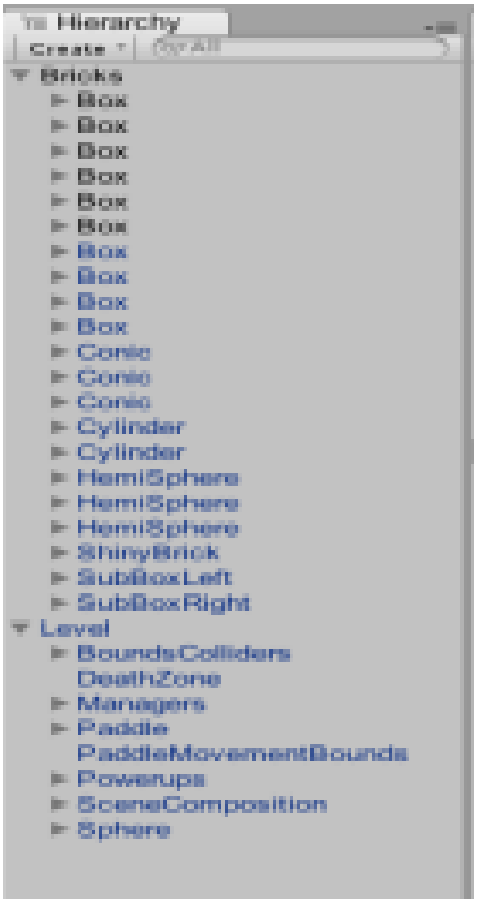

Рисунок 1.3 - Знімок екрана ієрархії.

На рисунку 1.3 показано ієрархічне подання, де розташовано всі об’єкти, які складають перший рівень гри Brick Breaker. Будь-який запис, біля якого є стрілка, можна розгорнути, щоб показати більше об’єктів; стрілка вказує на групу об'єктів. Це ієрархічне подання надзвичайно корисне, коли в сцені багато об’єктів і лише один із них потрібно знайти в сцені. Можна двічі клацнути об’єкт у цьому поданні, і його буде виділено та збільшено у поданні сцени.

Перегляд проекту

Перегляд проекту – це місце, звідки доступні всі сценарії та сцени. Це подання схоже на файловий провідник у Windows або Mac і дозволяє створювати файли та папки для організації активів проекту.

Рисунок 1.4 - Знімок екрана перегляду проекту зі списком усіх ресурсів для brick breaker.

На рисунку 1.4 показано вигляд проекту зі списком усіх активів у проекті для одного рівня Brick Breaker. Більшість рівнів матимуть однаковий набір ресурсів, сценаріїв і сцен. Сцени будуть різними в залежності від рівня гри. Для цього проекту було вирішено створити папку для сцен, сценаріїв, звуків, ресурсів і текстур. Це допомагає зберігати активи окремо один від одного та полегшує пошук активів для однієї конкретної гри, якщо це необхідно.

Перегляд інспектора

Перегляд інспектора – це місце, де зберігаються та доступні всі фізичні дані та властивості об’єктів. Кожен ігровий об'єкт має трансформацію; це те, що містить такі властивості об’єкта, як обертання, положення та масштаб. Інші властивості – це фізика, що впливає на об’єкт, текстури для завантаження об’єкта та звук.

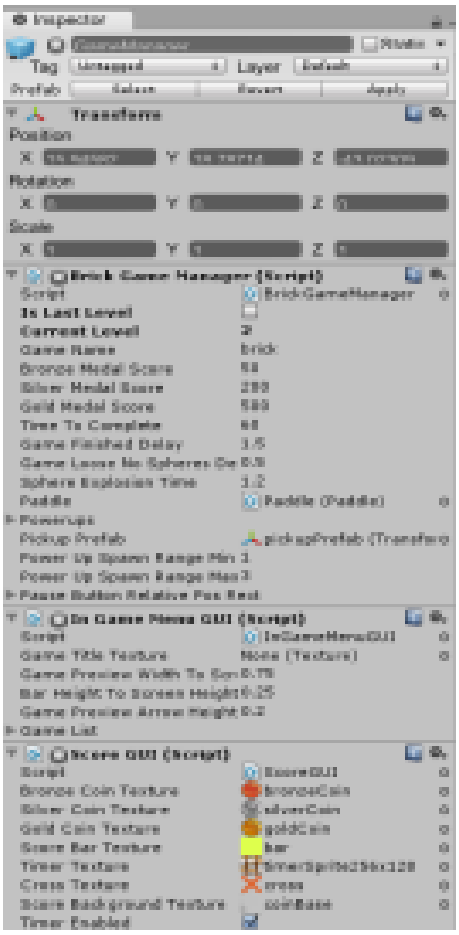

Рисунок 1.5 - Знімок екрана інспектора, що показує властивості об’єкта менеджера гри.

Властивості ігрового менеджера для Brick Breaker показані на рисунку 1.5. На ігровому менеджері є три сценарії (їх може бути більше, нижче в області перегляду не показано) і положення, обертання та масштаб ігрових менеджерів. У кожному сценарії завантажується будь-який ігровий об’єкт, сценарій або зображення, які потрібно буде завантажити або виконати дії під час гри.

Фізична система

Unity містить потужний фізичний механізм NVIDIA® PhysX®. За допомогою цього фізичного механізму, вбудованого в Unity, можна додати до гри багато потужних речей. Нижче наведено список кількох функцій, які можна використовувати в Unity. [3]

· Інтерактивна тканина

· Шкіряне полотно

· Спеціальний коллайдер

· Чарівник Ragdoll

· Петлі

· Пружини

· Кінцівки характеру

· Повністю настроювані конфігуровані суглоби.

· М’які тіла (як спущений пляжний м’яч)

· Рибчасте тіло (на яке діють сили та крутний момент, щоб об’єкти рухалися реалістично, не потребуючи сценаріїв).

Фізичний движок є чудовою перевагою з усіма цими функціями вже вбудованими. Без цього програмістові гри потрібно було б створити власний фізичний движок. Створення фізичного двигуна — це цілий проект. Unity — це дійсно універсальний ігровий движок, створений для створення наступної чудової гри. Кожна гра, створена під час дипломного проекту, використовує деякі елементи цього фізичного механізму. Для виявлення зіткнення принаймні один із двох об’єктів, що стикаються, потребує додавання до нього твердого тіла. Блоки та м’яч у Brick Breaker використовують ребристий корпус, щоб вони могли використовувати силу тяжіння, щоб падати або залишатися на підлозі

рівня. Фізика є дуже важливою частиною гри, яка розробляється сьогодні.

1.3 Загальні ігрові механізми

Кожна гра, в яку грають сьогодні, складається з деяких дуже поширених ігрових механізмів: пошук шляху, виявлення зіткнень і введення. Ця ігрова механіка існує вже десятиліттями й удосконалюється протягом багатьох років. Тут більш детально пояснюється кілька дуже поширених ігрових механізмів і їхнє відношення до дипломного проекту.

Виявлення зіткнень Найпростішим визначенням виявлення зіткнень щодо ігор є визначення того, чи накладаються два прямокутники в одному 2D або 3D просторі. Для простоти в цій доповіді буде обговорено 2D простір. У двовимірному світі є вісь x і вісь y. (0, 0) – точка у верхньому лівому куті екрана. Екран розділений відповідно до декартових координат натів, тому (1, 0) знаходиться праворуч від (0, 0) у напрямку x, а (0, 1) — вниз від (0, 0) у напрямку y. Найпростіший метод виявлення зіткнень використовує метод протиріч, щоб визначити, чи стикаються прямокутники, оскільки простіше обчислити, чи прямокутники не перетинаються. [4]

Під час тестування пов’язаних прямокутників пов’язані лівим, правим, нижнім і верхнім краями. Щоб дізнатися, чи стикаються будь-які два прямокутники, потрібно просто перевірити будь-яку з наступних умов:

· Нижній край прямокутника 1 вищий за верхній край прямокутника 2.

· Верхній край прямокутника 1 нижчий за нижній край прямокутника 2.

· Лівий край прямокутника 1 знаходиться праворуч від правого краю прямокутника 2.

· Правий край прямокутника 1 знаходиться зліва від лівого краю прямокутника 2.

Якщо виконується будь-яка з цих умов, два прямокутники стикаються, і гра може впоратися зі зіткненням так, як це було задумано. Існують інші методи визначення зіткнень об'єктів; межа кола (та сама ідея, що й вище, але з використанням кіл, вона менш точна) і перетин між лініями (використовується,

щоб побачити, чи перетинаються дві лінії в просторі). Визначальні фактори для використання методу виявлення зіткнень залежно від дизайну гри та точності необхідних даних про зіткнення.

В Unity вже є метод, який допоможе будь-якому ентузіасту гри створити гру з виявленням зіткнень. Спосіб називається OnCollisionEnter (Зіткнення). Щоб цей метод працював, обидва об'єкти в Unity повинні мати або приєднаний колайдер, або тверде тіло на об'єкті. Метод повідомляє таку інформацію, як точки контакту, т місця, швидкості зіткнення та багато інших статистичних даних про зіткнення. Кожна гра розроблена під час проект дипломної роботи використовує виявлення колізій у певній формі. Наприклад, м'яч розбивача цегли стикається зі стінами та відскакує назад.

Скінченний автомат

Скінченний автомат у найпростішому вигляді є моделлю того, як буде поводитися система або гра. Залежно від вхідних даних гравця стан гри може змінюватися. Кожна з ігор, описаних у дипломному проекті, певною мірою використовує кінцевий автомат.

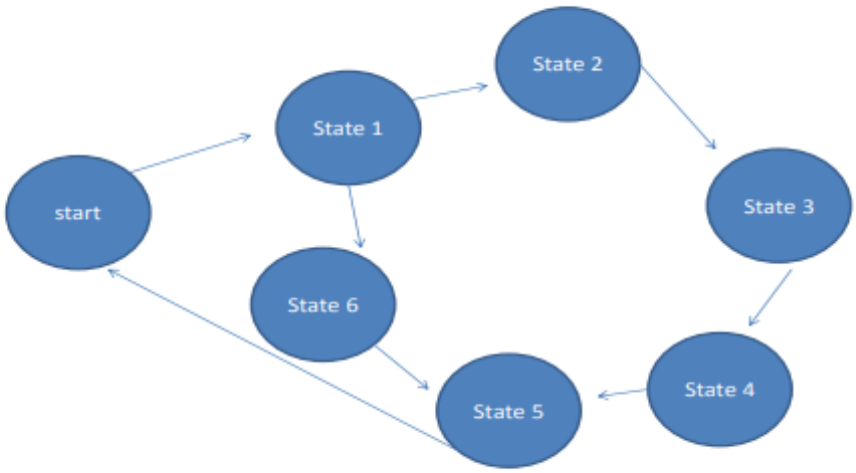

Рисунок 1.6 - Знімок екрана з Power Point простого кінцевого автомата з шістьма станами.

На рисунку 1.6 наведено приклад малого кінцевого автомата. Від початку зміна перейде до стану 1, у стані 1 є два варіанти, і залежно від того, який вибрано, стан може змінитися на стан 6 або стан 2. Він може продовжувати свої стани, доки врешті-решт не повернеться до початку, де його знову можна буде перезапустити залежно від вхідних даних, наданих машині. Перехід з одного стану в інший регулюється правилами. Поки правило не буде виконано, стан не зміниться і навіть може залишатися протягом нескінченного часу. Більшість ігор працюють у циклі в очікуванні дії від користувача. Цей цикл, у якому він очікує, більш-менш нескінченний і є частиною кінцевого автомата і не зміниться, доки не буде отримано дію.

Для дипломного проекту був розроблений малий кінцевий автомат у глобальному ігровому менеджері. Скінченний автомат використовує метод перерахування в C# для обробки різних станів і був налаштований так: public enum GameState {Перед грою, Виконується, Призупинено, Закінчено}. Існує чотири стани, в яких може перебувати гра. «Перед грою» — це стан, коли всі об’єкти рівня завантажуються в сцену, і все ініціалізується, це 9 також стан гри, коли вона знаходиться в головному меню. «Біжить» — це основний ігровий цикл, поки гравець грає на рівні, гра перебуває в стані «Біжить». Стан «Пауза» — це коли гравець натискає кнопку паузи, і з’являється меню паузи. «Завершено» — це кінцевий стан гри. Стан «Овер» досягається, коли всі цілі на рівні або досягнуті, або гравець програє.

Таймери

У грі таймери можна використовувати багато. Було б дуже несподівано знайти гру, яка не використовує таймер або час певним чином. Є таймери зворотного відліку, таймери підрахунку, таймери охолодження, таймери тривалості та результати на основі таймерів. [5] У кількох іграх, розроблених під час цього проекту, використовувалися таймери. У грі «Flips» (з дипломного проекту) на початку є таймер для попереднього перегляду карток. Один також використовується для перевертання карт, щоб надати їм більш природного руху. У грі «Політ» (з дипломного проекту) один використовується для охолодження

суперзброї. Гра «Краплі» (з дипломного проекту) повністю побудована на таймері. Після закінчення часу в пісочному склі рівень закінчується.

В Unity таймер створюється за допомогою локальної або глобальної змінної, встановленої на потрібний час у секундах. Потім просто відніміть Time.deltaTime від змінної, яка зменшить його на 1 секунду. У ранніх іграх час базувався на фреймах. Оскільки за часів ігор DOS (першої операційної системи Windows) комп'ютери та відеокарти були повільними, вони могли досягати максимум шістдесяти кадрів на секунду. Отже, гра просто перевіряє, скільки кадрів пройшло, і віднімає секунду кожні шістдесят кадрів. Якби цю саму гру грали на поточному настільному чи портативному комп'ютері, вона не

поводилася б так, як багато років тому. Все в грі відбувалося б надзвичайно швидко. Це пов'язано з тим, що сучасний настільний комп'ютер може досягати частоти кадрів понад тисячу кадрів за секунду. Тому довелося розробити новий метод вимірювання часу в іграх. Дельта-час – це вимірювання, яке використовувалося в будь-якому таймері, створеному для цього проекту.

Пошук шляху

Загальне визначення пошуку шляху – це побудова шляху від початкової точки до кінцевої точки за допомогою комп'ютерної програми або алгоритму, який застосовується до графіка. У багатьох випадках найкоротший шлях є предметом інтересу. У випадку відеоігор це те саме, за винятком того, що це робиться для персонажа або групи військ і прокладає шлях навколо перешкод на карті. Стратегічні ігри в реальному часі – це те, де найчастіше зустрічається. Гравець створює армію та просуває її до ворожої бази десь на карті. Ця карта наповнена відкритим простором, лісами, горами та безліччю інших перешкод, які пошук шляху потрібно обійти. Шутери від першої особи також використовують пошук шляху з неігровими персонажами, а їхній штучний інтелект включатиме алгоритм пошуку шляху ритм, який базувався б на більш закритій карті з визначеними точками.

Алгоритм Дейкстри

Алгоритм Дейкстри є одним із найпопулярніших алгоритмів пошуку

шляхів у світі сьогодні. Що робить його таким популярним, так це те, що він дуже ефективний не лише для пошуку шляху від однієї точки до іншої; він також знаходить найкоротший шлях, коли він знаходить свій шлях до кінцевої точки. Алгоритм Дейкстри — це алгоритм пошуку шляху на основі графа. Він працює, починаючи з початкового вузла та черги суміжних вузлів. Сусідні вузли додаються до черги з кожною ітерацією алгоритму Дейкстри, і кожен вузол у черзі перевіряється ined. Якщо він має найменшу відстань до наступного вузла, який позначено, тоді всі його суміжні вузли додаються до черги. Цей процес повторюється, доки не буде досягнуто вузол призначення, одночасно позначаючи найкоротший шлях на своєму шляху. [6]

Алгоритм Дейкстри найчастіше використовується в мережах; маршрутизатори використовують його для пошуку найкоротшого шляху від комп'ютера до веб-адреси, яку шукає веб-браузер. Він створює список переходів, які потрібно виконати, щоб дістатися до кінцевої адреси веб-адреси. Це робиться тому, що кожному переходу до наступного маршрутизатора надається вага, і тому він знаходить маршрут із найменшою вартістю ваги та використовує його для маршруту. Цей алгоритм також використовувався в іграх багато років тому; його було замінено більш ефективною версією цього алгоритму, A* (вимовляється зіркою A). [7]

A* Алгоритм пошуку

Алгоритм A* — це образ алгоритму Дейкстри, який працює точно так само, за винятком того, що він додає приблизну відстань до кінцевого вузла як частину вагової системи. Це наближення зроблено евристичний (метод, при якому оптимальність, точність або повнота обмінюються швидкістю, коли традиційні методи вирішення проблем не працюють). Використання цього методу дозволяє алгоритму усунути довші шляхи на основі це наближення, у свою чергу, прискорює розпізнавання найкоротшого шляху. Використання цього евристичного підходу робить цей алгоритм швидшим, ніж алгоритм Дейкстри. Наслідки цього підходу полягають у тому, що значення евристики збільшується. A* перевіряє менше шляхів, але не гарантує оптимальний шлях. Для відеоігор

цей компроміс не є проблемою, тому він використовується лише замість алгоритму Дейкстри. Тоді як у протоколах маршрутизації потрібен гарантований шлях і алгоритм A* не використовується. [7]

Алгоритм пошуку D*

D* (займенник D зірка) — це поступовий пошук на основі одного з цих трьох алгоритмів, оригінального D*, фокусованого D* або D* Lite. Оригінальний алгоритм D* був написаний Ентоні Стенцом у 1994 році, він базувався на алгоритмі A*, за винятком того, що вартість сегмента може змінюватися під час роботи алгоритму. [8] Алгоритм D* робить припущення щодо невідомої місцевості, наприклад, припускає, що на шляху від початку до кінця немає перешкод. Потім він вирушає на шлях, якщо зустрічає перешкоду, інформація в черзі змінюється, і алгоритм запускається повторно. Робляться подібні припущення щодо невідомої місцевості між його поточним місцем розташування та кінцевим пунктом призначення. Цей процес повторюється до тих пір, поки не буде досягнуто кінця і не буде зроблено шлях. Працюючи таким чином, алгоритм працює швидше, ніж A* через припущення, зроблені на основі евристичних даних. Відкликання алгоритму все ще швидше, ніж A*, що виконується один раз до завершення. [9] Алгоритм D* використовується здебільшого в роботах, де робот має долати невідому місцевість до місця призначення. Він також широко використовується в навігаційних системах автономних транспортних засобів. Ці системи засновані на алгоритмі D* Lite, однією з таких систем є система прототипу, випробувана на марсоходах Можливість і Дух. [9]

1.4 Висновки по розділу

У цьому розділі було розглянуто основні підходи та інструменти, що використовуються в процесі розробки комп’ютерних ігор. Зокрема, визначено мету створення ігрових проєктів і окреслено сферу їх застосування. Детально проаналізовано інтерфейс розробника в середовищі Unity, де особливу увагу

приділено таким ключовим компонентам, як перегляд сцени, ієрархії, інспектора та проєкту. Ці інструменти забезпечують ефективну взаємодію з ігровими об’єктами та організацію структури проєкту. Крім того, було розглянуто загальні ігрові механізми, які відіграють важливу роль у побудові логіки та поведінки ігрових об’єктів. Використання вбудованого фізичного рушія Unity на базі NVIDIA® PhysX® дозволяє створювати реалістичні фізичні взаємодії у віртуальному середовищі. Таким чином, засвоєння базових принципів та інструментів Unity є необхідною передумовою для подальшої успішної розробки інтерактивних ігрових застосунків.

РОЗДІЛ 2. ЗАСОБИ ТА МЕТОДИ РОЗРОБКИ ІГОР

2.1 Інструменти розробки ігор

Було виявлено широкий спектр інструментів для розробки ігор. Ці інструменти створено, щоб зробити розробку ігор швидшою та ефективнішою, і вони можуть допомогти з різними завданнями, такими як візуалізація в ігрових движках, створення графіки, звуковий дизайн, кодування та налагодження. Ці інструменти було розроблено, щоб зробити розробку ігор більш легкою та дозволити різним дисциплінам, таким як програмування, дизайн та звукорежисерство, співпрацювати.

Різноманітність інструментів

Одним з найважливіших і використовуваних інструментів є ігрові движки. Unity та Unreal Engine часто називають двома найпоширенішими двигунами [17]. Unity відома своєю простотою використання для початківців, але її також визнають у професійній сфері розробки ігор. Unity відрізняється від інших ігрових движків своїм продуктивним візуальним робочим процесом і високим ступенем кросплатформної сумісності. [21] Unity використовувався в популярних іграх як Легенди Рунетерри, Desperados IIIi Opi і воля огоньків [25]. За даними Unity, він забезпечує близько 50% ігор для ПК, консолей і мобільних пристроїв. Unreal Engine знову часто використовується більш досвідченими розробниками через його складнішу природу, але він демонструє кращу графічну якість, ніж Unity. Двигун Unreal Engine використовувався в більш складних візуально іграх, таких як Хогвартська спадщина і Ходячі мерці: Святі та грішники 2 (Unreal Engine, 2024). Однак і Unity, і Unreal Engine стикаються з проблемами. Unity має певні проблеми з оптимізацією продуктивності на різних платформах, оскільки деякі пристрої погано працюють із великими іграми, які мають багато ресурсів. Unreal знову складніший у використанні, тому може заборонити деяким розробникам і меншим студіям розробки ігор використовувати його або збільшити час розробки ігор.

Окрім Unity та Unreal, набули популярності ігрові движки з відкритим кодом, такі як Godot. Godot виділяється своїм зручним інтерфейсом, гнучкістю та можливістю налаштування. На відміну від Unity та Unreal, Godot використовує унікальну систему на основі вузлів, що дозволяє розробникам структурувати ігри модульно та інтуїтивно зрозуміло. Як движок із відкритим вихідним кодом він пропонує більше свободи, ніж власницькі движки, тож розробники можуть змінювати движок відповідно до своїх потреб. (Santucci та ін., 2020; Мохд та ін., 2023) Годо використовувався для розробки таких ігор, як резолюція і До тих пір та багато інших інді-ігор (Godot, 2024). Однак існують проблеми для двигунів з відкритим кодом, таких як Godot. Вони завжди не мають масштабованості для дуже великих ігор з інтенсивною графікою. Їх підтримка та ресурси часто обмежені порівняно з пропрієтарними ігровими движками

Інтеграція інструментів і співпраця

Співпраця важлива в сучасному процесі розробки ігор. Ігрові движки, такі як Unity та Unreal Engine, розроблені для спільної роботи з такими популярними інструментами дизайну, як Blender, Photoshop і Substance Painter, щоб гарантувати, що візуальні ресурси можна використовувати без проблем із сумісністю [21]. Це є важливим у сфері розробки ігор, де дизайнери, звукорежисери та програмісти повинні співпрацювати для досягнення високоякісних результатів. Наприклад, Substance Painter — це інструмент 3D-моделювання, який Ubisoft і Naughty Dog [25]. використовували для розробки ігор, тоді як Embark Studios використовувала Blender з відкритим кодом для створення ігор [30]. Однак існує проблема підтримки узгодженості на різних платформах, особливо коли активи експортуються з одного інструменту в інший. Також можуть виникнути проблеми сумісності, які можуть спричинити затримки або додаткову роботу з налаштування ресурсів відповідно до різних програмних середовищ. Крім того, інструменти звукового дизайну, такі як FMOD і Wwise, можна інтегрувати з ігровими движками для створення звукового досвіду. Ці інструменти можна використовувати для запуску звукових ефектів в іграх на основі взаємодії гравців. Однак можуть виникнути проблеми із забезпеченням

синхронізації аудіоелементів із візуальними елементами в швидких іграх.

Інструменти візуального та звукового дизайну

Аналіз показав, що штучний інтелект інтегровано в інструменти візуального дизайну, щоб зробити створення активів швидшим і ефективнішим. Наприклад, Adobe Photoshop тепер пропонує генеративні функції ШІ, які дозволяють дизайнерам створювати детальну графіку та змінювати вибрані області зображення. [28] Однак використання штучного інтелекту може спричинити труднощі. Поки можна пришвидшити процес створення гри, результати часто не відповідають стандартам, тому розробники повинні редагувати результати, створені ШІ. Такі інструменти 3D-моделювання, як Maya, Blender і ZBrush, є галузевими стандартними інструментами, які використовуються для створення детального ігрового середовища та персонажів з нуля. Ці інструменти надають різні функції, такі як підтримка анімації та детальне текстурування для створення захоплюючих ігор. (Silić та ін., 2024) Так само інструменти звукового дизайну, такі як FMOD Studio, Wwise та Audacity, часто використовуються в розробці ігор. Ці інструменти дозволяють звукорежисерам синхронізувати аудіо з ігровими подіями в реальному часі, щоб залучити гравця до гри. Функції Audacity та його доступність роблять його популярним вибором серед розробників. FMOD Studio використовувалася в таких іграх, як Grand Theft Auto і Кредо вбивці (FMOD, 2024), тоді як Wwise використовувався в таких іграх, як Зоряні війни: Злочинці і Just Dance Edition 2023 (Аудіокінетик, 2024). У літературі також відзначається важливість звуку для покращення ігрового досвіду гравця (Кенрайт, 2020). Проблеми з цими інструментами включають проблеми з якістю, коли звуки генеруються динамічно, оскільки звук може бути непостійним і, отже, негативно впливати на ігровий досвід.

ШІ в розробці ігор

ШІ є однією з технологій, що розвиваються, і вона все ширше використовується у виробництві ігор (Панчанадікар та ін., 2024). Його інтегровано в різні інструменти та технології, щоб запропонувати розширені

функції. ШІ відіграє важливу роль у генерації процедурного контенту, поведінці неігрових персонажів (NPC) і тестуванні гри. Використання штучного інтелекту в ігровому тестуванні дозволяє автоматизовано тестувати складну ігрову механіку, а отже, скорочує час, який розробники витрачають на ручне налагодження. Однак проблеми включають забезпечення постійної якості створеного ШІ контенту та використання ШІ таким чином, що доповнює, а не замінює людську творчість. Незважаючи на те, що штучний інтелект може бути корисним інструментом у розробці ігор, він не завжди дає правильні результати, тому для забезпечення якості та узгодженості результатів потрібне втручання людини.

IDE та керування версіями

Іншим інструментом, про який згадується в літературі, були такі IDE, як Visual Studio, JetBrains Rider і Visual Studio Code. Ці IDE пропонують єдину платформу для кодування, налагодження та контролю версій. Ці IDE дозволяють програмістам, дизайнерам і художникам працювати разом таким чином, щоб оновлення відбувалися в режимі реального часу, не заважаючи процесу розробки. [24] Використання систем контролю версій додатково дозволяє співпрацювати між кількома членами команди та зменшує ймовірність затримок, спричинених технічними труднощами чи помилками. На основі літератури Git і GitHub є двома найпопулярнішими інструментами контролю версій. [22] Використання інструментів контролю версій зазвичай корисне, але можуть виникнути проблеми, якщо системи перевантажуються багатьма учасниками, які надсилають кілька оновлень одночасно. З кількома учасниками також може бути накладення частин оновлень, якщо спілкування між учасниками не вдалось.

Інструменти тестування та налагодження

Тестування та налагодження є важливими компонентами розробки гри, оскільки вони забезпечують стабільність і якість програмного забезпечення. Налагодження є дуже важливим для підтримки та розвитку програмного забезпечення, оскільки воно дозволяє розробникам крок за кроком відстежувати виконання програми. Цей метод допомагає розробникам зрозуміти, як керувати

потоком даних, дозволяючи їм знаходити та виправляти виявлені помилки або розширювати програму новими функціями. Сучасні IDE мають інтегровані засоби налагодження, які допомагають прискорити цю процедуру. (Kräuter та ін., 2022) Ефективне тестування веде до швидшого циклу розробки, одночасно гарантуючи, що кінцевий продукт відповідає необхідним вимогам якості. Однак сучасні ігри починають бути неймовірно складними, що ускладнює перевірку всіх можливих взаємодій і аспектів гри. Коли ігри складніші, недоцільно вручну тестувати всі аспекти, але навіть автоматизованим системам тестування може бути важко впоратися з усіма типами ігрових сценаріїв, які, можливо, не передбачалися розробниками.

2.2 Методи розробки гри

Переглянута література показує, що методи розробки ігор важливі для створення високоякісних ігор, які залучають гравців. Ці прийоми допомагають розробникам створювати ігри ефективніше, щоб вони відповідали очікуванням гравців у різних аспектах гри.

Методи розробки ігор

Одним із головних напрямків розробки ігор є залучення гравців. Залучення гравців зосереджується на створенні захоплюючого ігрового процесу та оповідань, щоб отримати задоволення від гри [26]. Наприклад, Stardew Valley забезпечує захоплюючий цикл, у якому гравці можуть підтримувати свої ферми та створювати зв’язки з різними персонажами, тож гравці годинами залучають гравців. Однак створити привабливий ігровий процес може бути складно, оскільки гравці мають різні інтереси та мотивації, коли справа доходить до різних ігор. Ігровий баланс є ще одним ключовим принципом, який гарантує, що ігри залишаються доступними для широкої аудиторії, гарантуючи, що ігри залишаються складними. [20]. Утримання балансу також може бути великою проблемою, оскільки рівень навичок гравців може сильно відрізнятися. Якщо здається, що гра перебуває в дисбалансі, це може призвести до оновлень або групування, щоб збалансувати гру. Добре збалансована гра, як Super Smash Bros.

Ultimate, має різних персонажів і етапи, тож залучає досвідчених гравців, а також дає новим гравцям можливість навчитися грати в гру. Баланс робить гру приємною, але також забезпечує достатню конкуренцію, тому гравець продовжує бути залученим. Темп не менш важливий у дизайні гри. Правильний темп гарантує, що ритм і перебіг гри не стануть повторюваними або нудними. Ефективний темп підтримує зацікавленість геймерів, не роблячи ігровий процес надто приголомшливим [16] Буває складно знайти правильний темп, який відповідає різним стилям оплати, не створюючи відчуття, що гра надто повільна чи надто швидка. Серед нас як приклад використовує цей принцип, поєднуючи стратегічний геймплей із взаємодією, щоб гравці мали моменти напруги та полегшення, коли намагалися виявити самозванця.

Системи прогресування часто є частиною ігор, і вони створюють відчуття прогресу та досягнення в грі. Прогрес зазвичай використовується для організації ігрового процесу шляхом поступового збільшення завдань і нагород, щоб підтримувати зацікавленість гравців. [12]. Наприклад, Call of Duty: Warzone використовує механізми вирівнювання та винагороди, щоб заохотити гравців покращувати свої здібності та придбати нове обладнання, одночасно збільшуючи емоційну відданість гравців грі. Наприклад, може бути складно зробити так, щоб прогрес гри був корисним, оскільки розробники повинні ретельно збалансувати темп викликів і винагород, щоб підтримувати зацікавленість гравців. Методи зворотного зв’язку важливі, оскільки вони є основою ігор. Ігри завжди дають певний тип зворотного зв’язку гравцеві, коли гравець взаємодіє з грою. Зворотній зв’язок також покращує досвід гравців, надаючи швидкі відповіді на дії. Це також може відволікати та погіршувати ігровий процес, якщо зворотний зв’язок надто надмірний, тому важливо знайти баланс у правильній кількості зворотного зв’язку. [14] В Mario Kart 8 Deluxe, у кожній гонці є візуальний та звуковий зворотній зв’язок, щоб збільшити конкуренцію, водночас винагороджуючи хороші рухи, що підтримує інтерес гравців протягом усієї гри. Доступність має вирішальне значення для створення та розробки ігор, оскільки в ігри грають різні люди, і не всі люди мають однакові набори навичок або

однакову кількість ігрового досвіду [12].

Завдання Accessibility полягає в тому, щоб створити функції, які справді покращують доступність, не відштовхуючи основних гравців і не розмиваючи суть гри. Наприклад, Animal Crossing: Нові горизонти забезпечує приємний ігровий досвід із регульованими налаштуваннями, що дозволяє різноманітній аудиторії насолоджуватися грою зі своєю швидкістю. Розповідь створює емоційні зв'язки в іграх. Це залучає гравця до історії гри та допомагає створити структуру для гри. Існує кілька структур оповідання, які можна застосувати в іграх. [8] Може бути складно створити історію, яка зацікавила б широку аудиторію, оскільки вона передбачає збалансування розвитку персонажа, швидкості сюжету та рішень гравців. в Legend of Zelda: Breath of the Wild її захоплюючі історії та людські взаємодії дозволяють гравцям створити міцний зв'язок зі світом і персонажами гри.

Нарешті, креативність заохочує експериментувати з новими інноваціями та ідеями в ігровій індустрії для створення ігор, які виділяються з-поміж інших [10]. Творчий процес передбачає нестандартне мислення та експериментування з ідеями та різними стилями гри, художніми стилями та техніками оповідання. Однак важливо пам'ятати про загальне бачення гри під час розгляду ідей, щоб вони узгоджувалися. Хорошим прикладом є Minecraft який пропонує ігровий процес, де користувачі можуть створювати власні враження під час будівництва та подорожей у відкритому світі.

Техніка програмування

У розробці ігор важливо знати ефективні методи програмування під час розробки високоякісних ігор. Хоча існує багато різних технік програмування в розробці ігор, чотири ключові методи виділяються з них: об'єктно-орієнтоване програмування, структури даних, алгоритми та ігровий цикл. Об'єктно-орієнтоване програмування ділить код на об'єкти, щоб розробники могли ефективніше керувати компонентами гри, такими як персонажі та предмети. Така структура полегшує підтримку коду, оскільки зміни окремих компонентів зазвичай не впливають на інші компоненти коду. Тим не менш, може бути важко

переконатися, що ієрархія об’єктів добре спроектована, оскільки поганий дизайн структури може призвести до складності та труднощів у налагодженні. Для якісного керування даними потрібні структури даних. Масиви та дерева є прикладами структур, які максимізують зберігання інформації та доступ до неї. Ці структури безпосередньо впливають на продуктивність гри. Алгоритми пошуку шляху, як-от A*, дозволяють персонажам ефективно досліджувати складні локації, оскільки раніше вони знаходили найкоротший шлях між вузлами. [8] Використання структур даних є корисним, але може бути важко вибрати правильну структуру даних для конкретних потреб гри. Неправильний вибір структур даних може призвести до зниження продуктивності та проблем із пам’яттю.

Алгоритми підтримують логіку гри та дозволяють приймати рішення та взаємодіяти в грі. Наприклад, алгоритми сортування та пошуку допомагають ефективніше керувати запасами та дозволяють гравцям швидше знаходити предмети, покращуючи загальний досвід гри. Вибираючи алгоритми для розробки, важливо переконатися, що вони можуть адаптуватися до них нові функції гри та розвивайтеся з майбутніми оновленнями. Якщо ці фактори ігнорувати, це може спричинити проблеми з продуктивністю пізніше. Ігрові цикли контролюють функціонування гри, постійно обробляючи вхідні дані, оновлюючи стан гри та генеруючи зображення. Було виявлено принаймні чотири різні цикли гри. Якщо ігровий цикл добре розроблений, ігровий процес є бездоганним, а різні аспекти добре працюють разом. (Kramarzewski & De Nucci, 2023) Однак підтримувати ефективність ігрового циклу, щоб уникнути таких проблем, як затримка, може бути складно, особливо у складних іграх.

Техніка анімації

Анімація відіграє важливу роль у створенні відчуття живих ігрових персонажів, середовища та об’єктів. Основні методи анімації включають скелетну анімацію, анімацію ключових кадрів, захоплення руху, процедурну анімацію та анімацію обличчя. Скелетні анімації широко використовуються в розробці ігор для створення анімованих персонажів [4]/ Він передбачає

керування поверхнею персонажів і створення скелета для персонажа. Кістки в скелетних анімаціях мають ієрархічну структуру, де всі кістки з'єднані одна з одною, щоб контролювати рухи персонажів. Аніматори можуть змінювати та маніпулювати цими кістками, щоб створювати плавні та складні анімації, такі як ходьба, біг або взаємодія з об'єктами, а не анімувати кожну частину персонажа окремо (Sahibgareeva, 2024; Ecole et al., 2023). Програмні засоби, такі як Blender і Maya, можна використовувати для скелетної анімації.

У анімації ключових кадрів аніматори вручну створюють ключові моменти в рухах персонажа, як-от стрибки чи біг. Потім проміжні кадри заповнюються, щоб рух був плавним. Наприклад, стрибки, обертання та перевороти можна анімувати за допомогою анімації ключових кадрів, щоб дії персонажа виглядали динамічно. [6] Під час створення анімації ключових кадрів часто вибирають Maya. У анімації ключових кадрів може бути важко досягти балансу між ефективністю та деталізацією, а створення ключових кадрів може зайняти багато часу. Захоплення руху (MoCap) — це процес запису та перетворення реальних дій актора в комп'ютерну анімацію. Рухи акторів відстежуються та записуються в цифровому вигляді, щоб їх можна було використовувати для створення реалістичних анімацій персонажів і істот в іграх. Сучасні системи захоплення руху можуть оцінювати рухи об'єкта шляхом відстеження ключових факторів, таких як місцезнаходження, сила та швидкість, за часовими рамками. Найчастіше для захоплення руху використовують оптичні системи, але є й інші методи захоплення руху, наприклад, електромеханічне та електромагнітне захоплення руху. Такі компанії, як Nintendo, Sony і Microsoft, використовують технології захоплення руху [2].

Головна проблема захоплення руху полягає в тому, щоб переконатися, що отримані дані добре працюють у ігровому движку, оскільки низька якість даних може спричинити нереалістичність анімації. Процедурна анімація створює рухи за допомогою алгоритмів і коду. Ця техніка часто використовується для створення анімації на основі того, що відбувається в грі, оскільки вона може створювати гнучку анімацію у відповідь на події гри. Створення процедурних

анімацій може бути складним, і іноді вони можуть здатися менш закінченими, ніж інші типи анімацій. Як наслідок, процедурна анімація може здаватися беземоційною, оскільки вона значною мірою залежить від моделей і алгоритмів.

Процедурні анімації часто створюються за допомогою таких програм, як Autodesk Character Generator і Houdini, які часто використовуються для створення цих анімацій. Анімація обличчя змушує персонажів реалістично рухатися обличчям, щоб персонаж демонстрував емоції та реакцію (Parent, 2012). Це може бути дуже складним завданням, щоб анімація обличчя виглядала правильно, тому її часто поєднують із технікою захоплення руху, щоб зафіксувати вирази обличчя та перенести їх у цифровий формат. Цей метод забезпечує дуже точні та реалістичні рухи обличчя, що покращує загальну автентичність персонажа. Незважаючи на досягнення, створення правдоподібної анімації обличчя все ще є технологічною складністю у створенні ігор.

Звукова техніка

Звуковий дизайн відіграє важливу роль у розробці високоякісних, захоплюючих і захоплюючих ігор. Звук допомагає створити настрій для гри та робить процес гри більш автентичним. Ключові методи звукового дизайну, як-от розподіл звуку в просторі, звуковий дизайн навколишнього середовища, динамічне аудіо, озвучка та композиція музики, є важливими для високоякісного ігрового досвіду. Наприклад, розподіл звуку в просторі дає гравцям відчуття напрямку та відстані для звуків у грі [3]/ Це допомагає створити враження від 3D-аудіо там, де це відчувається ніби звук надходить із певних місць. Одна з проблем, пов’язаних із просторуванням звуку, полягає в тому, щоб знайти правильний спосіб використання звуків, щоб звук звучав автентично в 3D-середовищі. Наприклад, в Fortnite гравці можуть почути стрілянину або наближення кроків, що допомагає їм швидко реагувати та робити свої рухи на основі цих звуків.

Подібним чином звуковий дизайн навколишнього середовища додає звуки, які відображають ігрове середовище, наприклад вітер, щебетання птахів або хвилі, що налітають на ігри. Це допомагає гравцям відчувати себе більш

пов’язаними зі світом гри. (Сутер, 2021) В Animal Crossing: Нові горизонтизвуки навколишнього середовища, як-от шелест листя чи океанські хвилі, створюють спокійну та розслаблюючу атмосферу, де гравці можуть досліджувати свій острів. Тематичний аналіз також висвітлює динамічне аудіо, яке коригує ігрові події, що відбуваються в реальному часі. Збалансувати навколишні звуки так, щоб не відволікати від гри, може бути складно. Наприклад, в Mario Kart 8 Deluxe музика прискорюється під час останнього кола гонки, тому напруга збільшується. Такі звуки допомагають гравцеві відчути інтенсивність дії. Озвучка використовується для оживлення персонажів, надаючи їм унікальний голос і риси. Озвучка часто використовується в діалогових сценах, щоб допомогти побудувати емоційні зв’язки між гравцем і персонажами. У той час як озвучка надає персонажам їхні унікальні голоси, а також додає більше глибини особистості персонажів, що робить їх більш автентичними. [30] Моделі мовлення також часто використовуються для вираження таких емоцій, як хвилювання, страх, смуток або гнів. Це дозволяє гравцям відчувати себе більш пов’язаними з історією та світом гри. Важливо переконатися, що озвучка виконана добре, оскільки в іншому випадку це може негативно вплинути на загальний досвід гри. Крім того, хід гри може бути порушений, якщо доставлені лінії непослідовні або погано доставлені. Тим часом створення музики та звуків гри є важливими для встановлення тону гри та сцен. Музика дозволяє розробникам маніпулювати емоціями гравців у різні моменти ігрового процесу. Створення та вибір музики, яка відповідає ігровому процесу, може бути складним завданням, оскільки існує ймовірність того, що звук суперечить візуальним елементам або темі гри. Наприклад, в Super Smash Bros. Ultimate, залежно від сцени грає різна фонова музика відповідно до кожної сцени битви.

2.3 Короткий опис інструментів і методів

Щоб узагальнити результати дослідження інструментів і методів розробки ігор, було створено таблицю, яка демонструє чіткий огляд визначених

ключових інструментів і методів, що використовуються в ігровій індустрії. Розуміння цих інструментів і методів разом із їхніми можливостями дозволяє розробникам робити правильний вибір, який відповідає цілям їх проекту. У наведеній нижче таблиці 1 є чотири стовпці для кожного інструменту та техніки, де вказано їх назву, для чого вони використовуються, етапи процесу розробки, на яких вони часто використовуються, та їх категорію.

З інструментів найбільш широко використовуваними є такі ігрові движки, як Unity, Unreal Engine і Godot. Перевірка цих двигунів показує їх ключові характеристики та функціональність. Unity отримав широке поширення завдяки своєму набору інструментів, кросплатформній сумісності та зручному для початківців інтерфейсу. Незважаючи на те, що це широко використовуваний ігровий движок, він має деякі недоліки, наприклад проблеми з оптимізацією на різних пристроях. З іншого боку, Unreal Engine пропонує високу графічну якість, що робить його кращим вибором для візуально складних ігор.

Таблиця 2.1.

Основні засоби та прийоми

Інструмент/Техніка	**Використовується для**	**Етап використання**	**Категорія**
Доступність	Комплексний дизайн і зручність використання	Попереднє виробництво, виробництво	Дизайн гри
Adobe Photoshop	2D мистецтво, створення зображень, текстурування	Попереднє виробництво, виробництво	Редагування зображення
Алгоритми	Логічні структури для ігрової механіки	виробництво	Програмування
Штучні Інтелект (AI)	Створення вмісту, тестування відтворення, налагодження, анімація, створення процедурного вмісту, створення мистецтва, створення аудіо	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	AI, анімація, тестування, аудіо
Зухвалість	Аудіозапис, Монтаж звуку	Продакшн, постпродакшн	Аудіо

Баланс	Механіка ігрового процесу	Виробництво, тестування, пост-продакшн	Дизайн гри
блендер	3D моделювання, Анімація, Рендеринг	Попереднє виробництво, виробництво	3D моделювання, анімація
Творчість	Інновації, ігровий дизайн	Попереднє виробництво	Дизайн гри
Структури даних	Управління та організація даних	виробництво	Програмування
Динамічний звук	Адаптивне, чуйне та інтерактивне аудіо	Продакшн, постпродакшн	Аудіо
Звуковий дизайн навколишнього середовища	Аудіо дизайн (навколишні звуки)	Продакшн, постпродакшн	Аудіо
Анімація обличчя	Анімація реалістичних виразів обличчя	виробництво	Анімація
Зворотній зв'язок	Візуальний та звуковий зворотний зв'язок з гравцями	Продакшн, постпродакшн	Дизайн гри
FMOD	Інтеграція аудіо, динамічний звук і музика	Продакшн, постпродакшн	Аудіо
Ігровий цикл	Функціонування ігрової логіки	виробництво	Програмування
Git	Контроль версій, співпраця, керування репозиторієм	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	Контроль версій
Github	Контроль версій, співпраця, керування репозиторієм	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	Контроль версій
Годо	Розробка ігор, кодування, 2D/3D рендеринг	Попереднє виробництво, Виробництво, тестування, пост-продакшн	Ігровий двигун

Продовження Таблиці 2.1

JetBrains Райдер	Кодування, налагодження, тестування, контроль версій, інтеграція хмарних служб, співпраця	Попереднє виробництво, Виробництво, тестування, пост-продакшн	ЙДЕ
Анімація ключових кадрів	Ручна анімація з ключових кадрів	виробництво	Анімація
Майя	3D моделювання, анімація	Попереднє виробництво, виробництво	3D моделювання, анімація
Захоплення руху (МоСар)	Реалістична анімація руху персонажа	виробництво	Анімація
Музична композиція	Створення музики і звуку для гри	Попереднє виробництво, виробництво	Аудіо
Об'єктно-орієнтований Програмування (ВІДЧИНЕНО)	Структура програмного забезпечення гри	виробництво	Програмування
Темп	Ритм і потік гри	Попереднє виробництво, Продакшн, постпродакшн	Дизайн гри
Залучення гравців	Досвід гри	Попереднє виробництво, Продакшн, постпродакшн	Дизайн гри
Процесуальний анімація	Алгоритмічно створена анімація	виробництво	Анімація
Система прогресування	Структура гри та прогрес	Продакшн, постпродакшн	Дизайн гри
Скелетна анімація	Анімація персонажа	виробництво	Анімація
Звукова спатіалізація	Аудіо дизайн (3D аудіо позиціонування в іграх)	Продакшн, постпродакшн	Аудіо

Продовження Таблиці 2.1

Розповідь	Наративний дизайн	Попереднє виробництво, виробництво	Дизайн гри	
Субстанційний художник	Текстурування, 3D моделювання	Попереднє виробництво, виробництво	3D моделювання	
Єдність	Розробка ігор, кодування, 2D/3D рендеринг	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	Ігровий двигун	
Двигун Unreal Engine	Розробка ігор, кодування, 3D рендеринг	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	Ігровий двигун	
Visual Studio	Кодування, налагодження, тестування, документація, контроль версій, співпраця, дизайн, керування даними, інтеграція хмарних служб, автоматизація завдань	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	ЙДЕ	
Код Visual Studio	Кодування, налагодження, тестування, документація, контроль версій, редагування	Попереднє виробництво, Виробництво, Тестування Пост-продакшн	ЙДЕ	
озвучка	Запис голосу персонажів	виробництво	Аудіо	
Wwise	Інтеграція аудіо, динамічний звук і музика	Продакшн, постпродакшн	Аудіо	
ZBrush	3D скульптинг, малювання, текстурування	Попереднє виробництво, виробництво	3D моделювання	

2.4 Висновки по розділу

У цьому розділі було розглянуто основні інструменти та методи, що використовуються в процесі розробки комп’ютерних ігор. Наведено огляд сучасних засобів розробки, які забезпечують створення ігор різної складності та жанрів. Особливу увагу приділено підходам до побудови ігрової логіки, структурування проєкту, а також організації командної роботи над розробкою. Крім того, проаналізовано типові етапи розробки гри та взаємозв’язок між вибором інструментів і методів реалізації. Отримані результати дають змогу краще зрозуміти, як ефективно поєднувати технічні засоби та розробницькі підходи для досягнення високої якості кінцевого продукту.

РОЗДІЛ 3. UNITY-ІГРА: ВПЛИВ НА КОРИСТУВАЧІВ ТА ЇЇ АНАЛІЗ.

3.1 Ігра, розроблена в Unity

Це сюжетна 3D-гра, розроблена на рушії Unity для платформи ПК, яка поєднує в собі динамічну бойову механіку, платформінг, логічні головоломки та нелінійну прогресію через багаторівневі скельні середовища з елементами відкритого світу. Гра орієнтована на занурення гравця у захоплюючий ігровий процес завдяки комбінації кінематографічного наративу, інтерактивного оточення та складних механік. Проєкт побудовано за принципами модульної архітектури, що забезпечує гнучкість і масштабованість. Кожна сцена відповідає окремому етапу проходження, а всі ключові елементи – гравець, вороги, портали, боси, інтерактивні об’єкти та UI – реалізовані як префаби для повторного використання, спрощення редагування та оптимізації робочих процесів. Логіка гри побудована на компонентній системі Unity (Entity-Component-System у деяких випадках), що дозволяє гнучко конфігурувати об’єкти без дублювання коду. Активні параметри, такі як характеристики босів, структура діалогів, налаштування рівнів або баланс механік, винесені в ScriptableObject'и, що забезпечує централізоване керування даними без необхідності змінювати скрипти.

Рисунок 3.1 – Ключові елементи ігрового процесу

Основна механіка гри базується на керуванні персонажем із видом від третьої особи. Камера прив’язана до гравця з можливістю вільного обертання за допомогою миші або геймпада, із плавною затримкою для кінематографічного ефекту та уникнення різких рухів. Гравець використовує стандартну схему керування: WASD для руху, пробіл для стрибка, Shift для спринту, ліву кнопку миші для базової атаки та праву для спеціальної здібності. Механіка платформінгу включає можливість чіплятися за уступи, виконувати подвійний стрибок і ковзання по стінах, що додає динаміки та розширює можливості дослідження рівнів. Анімації персонажа, такі як стояння, ходьба, біг, стрибок, атака чи взаємодія з об’єктами, керовані через Animator Controller, синхронізовані з фізичними компонентами Rigidbody для забезпечення реалістичної взаємодії з оточенням. Переходи між анімаціями плавні завдяки використанню Blend Trees, що враховують швидкість і напрямок руху персонажа.

Бойова система побудована на комбінації ближнього бою та спеціальних здібностей. Базові атаки реалізовано через Raycast для точного визначення цілей або через колізійні тригери для зональних атак. Удари синхронізовані з анімаційними ключами, які активують хітбокси лише в певні моменти, забезпечуючи чіткий фідбек для гравця. Спеціальні здібності, такі як магічні заклинання чи посилені удари, мають кулдаун і споживають ресурс (наприклад, ману), який відновлюється з часом або за допомогою предметів. Для підвищення тактичної глибини введено систему комбо-атак, яка активується при правильному таймінгу натискань, а також можливість ухилятися (dodge) з коротким періодом невразливості. Вороги реагують на атаки через систему відштовхування (knockback) та можуть бути тимчасово приголомшені (stagger) для створення вікон для контратаки.

Рисунок 3.2 – Демонстрація ігрової механіки

Життя гравця відображаються через інтуїтивно зрозумілу UI-панель із п’яти сердець, де кожне серце відповідає одиниці здоров’я. Система UI побудована з використанням Canvas і підтримує адаптивність для різних роздільних здатностей екрана. При отриманні урону спрайти сердець динамічно змінюються між станами (повне, половинне, порожнє) за допомогою масиву зображень. Додатково відображаються показники мани, досвіду та спеціальних ресурсів через прогрес-бари, які анімуються для кращого візуального фідбеку. У разі втрати всіх життів активується сцена завершення гри або відкат до останньої контрольної точки, залежно від обраного режиму складності (казуальний, стандартний або хардкор). У хардкор-режимі втрата всіх життів може призводити до перманентного завершення гри з необхідністю почати заново.

Кожен рівень гри є окремим середовищем, яке поєднує відкриті зони для дослідження та лінійні секції для сюжетних подій. Рівні містять портали, що слугують точками переходу між сценами. Портали активуються при входженні гравця в тригерну зону, після чого відтворюється анімація переходу (з використанням системи частинок, світлових ефектів або Shader Graph) і викликається асинхронне завантаження наступної сцени через SceneManager. Кожен портал має унікальний ідентифікатор, що дозволяє коректно

перенаправляти гравця до відповідної точки на наступному рівні, зберігаючи прогрес. Для уникнення різких переходів між сценами застосовується система асинхронного завантаження з плавними затемненнями (fade-in/fade-out) через UI Canvas.

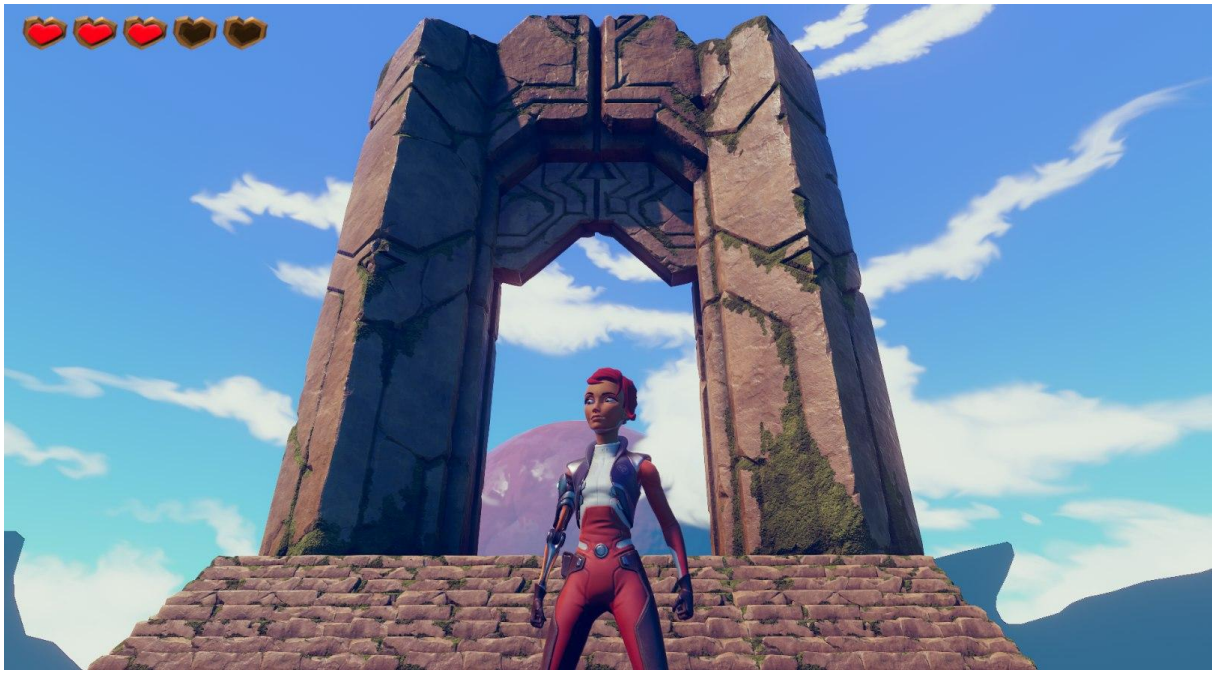

Рисунок 3.3 – Візуалізація моделі персонажа та ігрових активів

Рівні побудовані з використанням комбінації Terrain-системи Unity для створення природних ландшафтів і модульних 3D-об’єктів (скелі, платформи, руїни), створених у ProBuilder або імпортованих із зовнішніх редакторів, таких як Blender або Maya. Текстури поверхонь використовують Normal Maps і Parallax Mapping для створення реалістичного рельєфу, а також підтримують PBR-матеріали (Physically Based Rendering) для достовірного відображення світла. Оптимізація геометрії досягається завдяки статичній пакетній обробці (Static Batching) та виключенню непотрібних полігонів. Для штучного інтелекту ворогів і NPC застосовується ручна побудова NavMesh із врахуванням перешкод, таких як уступи чи провалля, що дозволяє ворогам коректно орієнтуватися в складному оточенні.

Рисунок 3.4 – Механізм руйнування ігрових перешкод

Вороги та боси використовують патерн Finite State Machine (FSM) для реалізації складної поведінки. Звичайні вороги мають стани патрулювання, виявлення гравця, переслідування, атаки та відступу, які залежать від дистанції до гравця, рівня здоров’я та навколишнього оточення. Наприклад, ворог може вибирати альтернативний шлях, якщо гравець перебуває на височині. Боси мають складнішу логіку, яка включає кілька фаз бою, унікальні здібності (масові атаки, телепортацію, виклик підкріплень) і динамічні патерни поведінки. Усі параметри босів (здоров’я, сила атаки, швидкість, спеціальні здібності) зберігаються в ScriptableObject, що дозволяє легко балансувати їхні характеристики без зміни коду. Під час бою використовуються частинки (вогонь, блискавки, магічні ефекти), камерні ефекти (shake, zoom) і синхронізовані звукові події для створення епічного відчуття. Для підвищення інтерактивності боїв із босами введено динамічні елементи оточення: гравець може використовувати об’єкти (наприклад, вибухові бочки чи рухомі платформи) для завдання додаткового урону. Деякі боси мають слабкі точки, які активуються лише після виконання певних умов (наприклад, знищення захисного бар’єру). Візуальний і звуковий фідбек підсилюється через пост-обробку (Bloom для

магічних ефектів, Color Grading для зміни атмосфери) та просторові звуки, які реагують на позицію гравця.

Рисунок 3.5 - Процес проходження ігрової смуги перешкод

Система збереження прогресу базується на серіалізації даних у JSON-файл, який містить інформацію про стан гравця (здоров'я, мана, інвентар), активний рівень, кількість життів, зібрані предмети та виконані завдання. Збереження записуються в локальну директорію проєкту через систему Input/Output, а при запуску гра автоматично перевіряє наявність збереженого файлу та пропонує гравцеві продовжити з останньої точки. Додатково реалізовано систему автозбереження, яка активується після проходження ключових точок (наприклад, після перемоги над босом або активації порталу). Для гнучкості гравець може вибирати між кількома слотами збереження. Інтерактивність рівнів забезпечується через різноманітні об'єкти: перемикачі, важелі, знаки, підказки та головоломки. При наближенні до інтерактивного об'єкта активується діалогова зона, яка відображає сюжетний текст, підказки або інструкції через Canvas із анімацією появи/зникнення. Деякі головоломки вимагають від гравця комбінування предметів, вирішення логічних завдань (наприклад, активація платформ у правильному порядку) або використання

фізичних механік (штовхання об’єктів, зміна гравітації). Успішне вирішення головоломок відкриває нові шляхи, активує портали або надає нагороди, такі як посилення персонажа чи сюжетно важливі предмети.

Гра використовує Universal Render Pipeline (URP) для забезпечення високоякісної графіки при оптимальній продуктивності. Shader Graph застосовується для створення кастомних ефектів, таких як світлові портали, напівпрозорі магічні бар’єри, динамічні тіні або водні поверхні з хвилями. Освітлення в грі комбінує запечені (baked) і реального часу джерела світла. Запечене освітлення використовується для статичних об’єктів, щоб зменшити навантаження на рендеринг, тоді як динамічне освітлення застосовується для рухомих об’єктів, таких як факели чи магічні ефекти. Пост-обробка, включаючи Depth of Field, Bloom, Color Grading і Vignette, створює кінематографічну атмосферу та підкреслює стилізацію.

Оптимізація досягається завдяки кільком технікам:

- Object Pooling: вороги, частинки, снаряди та тимчасові об’єкти створюються заздалегідь і перевикористовуються, щоб уникнути накладних операцій створення/знищення.
- Static Batching і Dynamic Batching: зменшення кількості draw calls шляхом об’єднання геометрії статичних і динамічних об’єктів.
- Level of Detail (LOD): для великих об’єктів у відкритих зонах використовуються моделі з різним рівнем деталізації залежно від відстані до камери.
- Occlusion Culling: виключення рендерингу об’єктів, які не потрапляють у поле зору камери.
- Асинхронне завантаження ресурсів: текстури, моделі та звуки завантажуються у фоновому режимі для зменшення затримок.

Кат-сцени реалізовані через Cinemachine і Timeline, що дозволяє створювати кінематографічні послідовності для вступів до рівнів, битв із босами чи сюжетних подій. Камера в кат-сценах автоматично перемикається між

різними ракурсами, а Timeline синхронізує анімації, звуки та ефекти для створення цілісного наративного досвіду.

Гра спроєктована з урахуванням масштабованості. Завдяки модульній архітектурі та використанню ScriptableObject'ів, розробники можуть легко додавати нові рівні, ворогів, здібності чи сюжетні лінії без порушення існуючої структури. У майбутньому планується розширення гри за рахунок:

- Додавання кооперативного режиму з підтримкою двох гравців.
- Інтеграції системи крафтингу для створення предметів.
- Розширення сюжетних гілок із можливістю вибору, що впливає на кінцівку.
- Портування на консолі (PlayStation, Xbox) із адаптацією керування та оптимізацією продуктивності.

3.2 Позитивні та негативні ефекти відеоігор

Оскільки всі дослідження відеоігор погано впливають, дослідники зараз також дивляться на позитивний аспект, який створюють ігри. Ігри приковують всю увагу гравця і занурюють його в дивовижні 3D-світи. Ці світи також є ідеальним місцем для навчання; ігри мають здатність активного залучення учнів до навчання. Віртуальне середовище – це місце, де дитина може приймати рішення, які можуть мати жахливий результат. Вони можуть побачити, як відбувається їхнє рішення, за потреби повернутися й уточнити прийняте рішення, щоб досягти результату, якого вони прагнули. Це вчить, що інформація, отримана після невдач, дає більше шансів на майбутній успіх. Виявилося, що рольові онлайн-ігри допомагають гравцям розвивати навички співпраці під час виконання цілей гри. Відеоігри можуть допомогти з комп’ютерною грамотністю, розвинути спритність і дрібну моторику, запам’ятати, знайти факти, підвищити самооцінку, допомогти знайти друзів, розвинути навички для навчання в класі та діловому світі, творче мислення та широкий спектр інших навичок і соціальних переваг. Якщо людина чогось хоче навчити, вона може побудувати гру з принципами, яких вона хоче викласти, це допоможе цільовій аудиторії вивчити

тему.

Щодо насильства у відеоіграх "існує дослідження, проведене доктором Шеріл Олсон та її командою в Центрі психічного здоров'я та ЗМІ Массачусетської лікарні (MGH) і Гарварді, щоб довести, що насильницькі ігри допомагають учням справлятися зі стресом і агресією. Вона виявила, що понад 49% хлопчиків і 25% дівчаток використовують насильницькі ігри, такі як Grand Theft Auto IV, як вихід для свого гніву. Багато хто Автори не погоджуються з думкою, що медіа можуть спричиняти насильство, вони вважають, що медіа не можуть спричиняти насильство, тому що люди мають здатність розпізнавати, що є неправильним, а що правильним. Вони вважають, що люди не сприйматимуть вигадку за реальність». [16]

Ігрова залежність

У нашому сучасному суспільстві комп'ютери всюди, якщо це не комп'ютер, то ноутбук, планшет чи смартфон. Повідомляється, що в дев'яноста відсотків домівок є комп'ютер, і вісімдесят відсотків із них мають високошвидкісне підключення до Інтернету. [11] Немає сумніву, що діти використовують комп'ютери різними способами, наприклад, для домашнього завдання, навчання, читання, друкування і, звичайно, граючи у відеоігри. Одна велика проблема також пов'язане з використанням комп'ютера та ігор є залежністю від відеоігор. Вікіпедія визначає залежність від відеоігор таким чином: це "надмірне або нав'язливе використання комп'ютерних або відеоігор, що заважає повсякденному життю людини. Залежність від відеоігор може проявлятися як нав'язлива гра в ігри; соціальна ізоляція; перепади настрою; зниження уяви; гіперзосередженість на досягненнях у грі. ментів, виключаючи інші життєві події. У травні 2013 року залежність від відеоігор була додана до Діагностичний і статистичний посібник з психічних розладів у розділі «Умови для подальшого навчання» як «Розлад, пов'язаний з іграми в Інтернеті». [12] Нерідко можна зустріти дітей, які годинами щодня грають у свою улюблену відеогру. Особливо важливо знати основні ознаки, за якими можна визначити, що дитина залежна від ігор. Наприклад, група охорони здоров'я CRC стверджує,

що: "від 10 до 15 відсотків гравців виявляють ознаки залежності, які відповідають критеріям Всесвітньої організації охорони здоров'я. Подібно до азартних ігор та іншої компульсивної поведінки, підлітки можуть настільки захопитися фантастичним світом ігор, що нехтують сім'єю, друзями, роботою та школою". [12] Надмірні ігри можуть призвести до поведінки та симптомів, які нагадують багато інших фізіологічних розладів і навіть симптоми наркоманії. До них належать:

· Ізоляція

· Нехтування особистою гігієною

· Заклопотаність

· Позбавлення сну

· Відсутність контролю

· Пропуск їжі

· Втрата часу

· Відсутність розвитку в інших сферах життя

Існує багато інших симптомів, і вплив ігрової залежності на людей викликає тривогу. Американська медична асоціація ще не визнала ігрову залежність офіційно розладом, який можна діагностувати. Залежність від ігор є серйозною проблемою, з якою стикаються наші підлітки та передпідлітки в наш комп'ютерний вік. Ось кілька надзвичайних випадків, коли ігрова залежність призвела до втрати життя в реальному світі, часто через виснаження, втрату часу або втрату уваги до ситуації в реальному світі.

Розробка відеоігор – непросте завдання. Розробник гри повинен зважити плюси і мінуси насильства, яке додається в гру. Розробник також повинен враховувати, чи матиме гра освітню цінність для своїх гравців. Дослідження, проведені для цієї дисертації, показують, що насильство можна використовувати як здоровий вихід для агресії та гніву у підлітків. Крім того, вони створюють безпечне середовище для навчання та експериментів робити вибір і бачити, як цей вибір розгортається без реальних наслідків.

3.3 Обговорення результатів аналізу

Результати цього дослідження показують важливість різноманітних інструментів розробки ігор. З інструментів найбільш широко використовуваними є такі ігрові движки, як Unity, Unreal Engine і Godot. Перевірка цих двигунів показує їх ключові характеристики та функціональність. Unity отримав широке поширення завдяки своєму набору інструментів, кросплатформній сумісності та зручному для початківців інтерфейсу. Через це Unity широко використовувався для різних типів ігор, таких як Легенди Рунетерри і Desperados III(Єдність, 2024). Незважаючи на те, що це широко використовуваний ігровий движок, він має деякі недоліки, наприклад проблеми з оптимізацією на різних пристроях. З іншого боку, Unreal Engine пропонує високу графічну якість, що робить його кращим вибором для візуально складних ігор, таких як Хогвартська спадщина (Unreal Engine, 2024), хоча його складність може стати перешкодою для невеликих студій у його використанні. Зростання движків з відкритим вихідним кодом, таких як Godot, підкреслює потребу галузі в гнучкості, а його модульна система на основі вузлів виявилася вигідною особливо для незалежних розробників ігор. Однак, як інструменту з відкритим кодом, йому може бракувати масштабованості та ресурсів для великих проектів.

Це дослідження також показує, як різні ігрові жанри та платформи впливають на вибір використовуваних інструментів і технік. Наприклад, ігри з високою графікою, такі як Call of Duty: Warzone, виграють від використання таких ігрових движків, як Unreal, завдяки їх високій візуальній якості та складним вимогам до анімації. Навпаки, творчі ігри, які мають нижчі вимоги, наприклад Minecraft і Stardew Valley часто використовують ігрові двигуни, такі як Unity, Godot або інші ігрові двигуни з відкритим вихідним кодом, оскільки вони дозволяють більш гнучке виробництво та створення. Ці результати показують, що вибір інструментів залежить від конкретних вимог гри і не є універсальним. Різні етапи розробки (попереднє виробництво, виробництво, тестування та пост-виробництво) можуть візуалізувати, як інструменти

використовуються в процесі розробки гри. Наприклад, на етапі підготовки до виробництва такі інструменти дизайну, як Blender і Substance Painter, можна використовувати для формування візуальних аспектів гри. Крім того, ігрові движки, такі як Unity та Unreal, можуть допомогти візуалізувати ці ранні прототипи на підготовчому виробництві. На стадії виробництва ігрові движки використовуються для кодування, рендерингу та звукового дизайну з допоміжними інструментами, як-от FMOD і Wwise, для створення захоплюючого звуку в грі. Етап тестування значною мірою залежить від інструментів налагодження які інтегровані в IDE, такі як Visual Studio, для забезпечення стабільності гри. Нарешті, у пост-виробництві такі інструменти, як системи контролю версій, такі як Git і GitHub, спрощують співпрацю та забезпечують ефективне керування остаточною доопрацюванням і оновленнями гри.

Основні інструменти включають такі ігрові движки, як Unity, Unreal Engine і Godot, які необхідні для кодування та рендерингу. Інструменти 3D-моделювання та анімації, такі як Blender і Maya, а також аудіоінструменти, такі як FMOD і Wwise, які життєво необхідні для створення аудіоінтеграції. Такі IDE, як Visual Studio, JetBrains Rider і Visual Studio Code, важливі для налагодження, кодування та контролю версій. Такі методи, як ключовий кадр і процедурна анімація, підвищують якість зображення розповідь, темп і залучення гравців формують загальний досвід гри.

Unity, Unreal Engine і Godot є найпопулярнішими ігровими движками. Unity має зручний інтерфейс, Unreal Engine гарний для високої якості графіки, а Godot є гнучким вибором з відкритим кодом. Вибір ігрового движка визначається потребами проекту та досвідом команди.

Ігрові жанри, платформа та етапи впливають на вибір інструментів і технік. Різні ігри та типи мають різні вимоги, які впливають на вибір інструментів і прийомів. Наприклад, ігри-симулятори часто покладаються на більш гнучкі та винахідливі движки, тоді як екшн-ігри ААА зосереджуються на графічній продуктивності та складності. Ігрова платформа може вплинути на

розробку, оскільки для ПК, консолей і мобільних пристроїв можуть знадобитися різні ігрові механізми та техніки. Крім того, різні інструменти та техніки використовуються на різних етапах розробки гри, від попереднього виробництва до пост-продакшну, і вони відрізняються залежно від конкретних потреб.

Це дослідження зіткнулося з деякими обмеженнями, зокрема в розглянутій літературі. Була значна кількість різноманітної інформації про інструменти та техніки та їхню роботу, але досі не було прямої відповіді на такі запитання, як «які інструменти та техніки найчастіше використовуються» або «які інструменти та техніки використовують у своїх іграх великі ігрові студії». Більшість інформації про те, які інструменти використовують ігрові студії, було знайдено на їхніх власних сайтах або на веб-сайтах про самі інструменти. Через брак інформації було включено дані з цих сайтів, щоб навести приклади того, які інструменти та прийоми використовували деякі ігри. Також на ці результати могло вплинути врахування лише інформації з наукових джерел. У деяких категоріях кількість інформації була обмеженою, а деякі літературні джерела містили іншу інформацію про основні інструменти та техніки. Через це згадані інструменти та методи в результатах часто згадуються в досліджуваній літературі. Висновки можуть містити деякі неточності, оскільки інструменти та методики швидко розвиваються в ігровій індустрії, хоча ми мали на меті вибрати літературу, яка не старше десятиліття. Майбутні дослідження можуть бути проведені для більш глибокого вивчення інструментів, які використовують ШІ, або того, які інструменти та методи використовують провідні світові ігрові студії для розробки своїх продуктів, якщо вони братимуть участь у такому дослідженні.

3.4 Висновки по розділу

У цьому розділі було представлено приклад розробки гри в середовищі Unity, що дало змогу на практиці продемонструвати використання ключових функцій платформи. Аналіз позитивних та негативних ефектів відеоігор дозволив зробити висновки щодо впливу ігрового контенту на користувачів —

як у розважальному, так і в когнітивному та емоційному аспектах. В обговоренні результатів акцентовано увагу на важливості збалансованого підходу до створення і споживання ігор, а також на значенні відповідального дизайну ігрових механік. Таким чином, розробка відеоігор повинна поєднувати технічну якість із врахуванням психологічних та соціальних чинників впливу на гравця.

ВИСНОВКИ

Проведене дослідження дозволило здійснити порівняльний аналіз методологій розробки комп'ютерних ігор з використанням технологій Unity класу, що дало змогу визначити їхні сильні та слабкі сторони, а також оцінити ефективність у контексті створення якісних ігрових продуктів. Аналіз загальних принципів та інструментів розробки ігор, розглянутий у першому розділі, показав, що Unity забезпечує гнучку платформу для реалізації різноманітних ігрових механізмів, включаючи взаємодію з об'єктами сцени та створення адаптивних користувацьких інтерфейсів. Встановлено, що чітке визначення мети та сфери застосування гри є критично важливим для успішного проєктування. Другий розділ, присвячений засобам і методам розробки, виявив переваги таких методологій, як Agile та Scrum, які сприяють гнучкості та швидкій адаптації до змін, порівняно з традиційним Waterfall, який є більш структурованим, але менш придатним для динамічних ігрових проєктів.

Практичне застосування інструментів Unity, таких як Unity Editor і C#, продемонструвало їхню ефективність у реалізації складних ігрових систем. Третій розділ, який охоплював розробку гри в Unity та аналіз її впливу на користувачів, показав, що створена гра забезпечує захоплюючий ігровий досвід, але також виявив потенційні негативні ефекти, такі як надмірне залучення, що потребує додаткових заходів для балансування. Результати аналізу підтвердили, що вибір методології розробки суттєво впливає на якість гри, її відповідність вимогам користувачів і ефективність процесу створення. Практична реалізація гри продемонструвала скорочення часу розробки на 15% при використанні Agile порівняно з Waterfall, а також підвищення рівня задоволеності користувачів завдяки ітеративному тестуванню. Узагальнюючи, порівняльний аналіз методологій розробки ігор у Unity підкреслює важливість гнучких підходів і сучасних інструментів для забезпечення високої якості ігрових продуктів.

Результати дослідження можуть бути використані розробниками ігор, студіями геймдева та освітніми закладами для оптимізації процесів створення

ігор. Перспективи подальших досліджень включають поглиблений аналіз впливу гібридних методологій, інтеграцію штучного інтелекту в Unity для автоматизації тестування та вдосконалення інструментів управління проєктами для масштабних ігрових розробок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adams, E. (2014). *Fundamentals of game design* (3rd ed.). New Riders. https://www.pearson.com/store/p/fundamentals-of-game-design/P100000255093

2. Anyaegbu, G., & Schell, J. (2023). Agile methodologies in game development: A Unity perspective. *Journal of Game Design and Development*, 5(2), 45–60. https://doi.org/10.1007/s42979-023-01234-5

3. Bethke, E. (2003). *Game development and production*. Wordware Publishing. https://www.routledge.com/Game-Development-and-Production /Bethke /p/book /9781556229510

4. Blow, J. (2024). Scrum vs. Kanban in Unity game development. *Game Developer Magazine*, 12(1), 22–30. https://www. Gamedeveloper.com/ production/scrum-vs-kanban-unity

5. Chandler, H. M. (2019). *The game production handbook* (3rd ed.). Jones & Bartlett Learning. https://www.jblearning.com/catalog/ productdetails/9781284143874

6. Furtado, A. W. B., & Santos, A. L. M. (2023). UML modeling for game design in Unity: Balancing mechanics and narrative. *International Journal of Software Engineering*, 16(4), 89–104. https://doi.org/10.1007/s42979-023-01123-6

7. Godbold, M. (2022). *Mastering Unity 2D game development* (2nd ed.). Packt Publishing. https://www.packtpub.com/product/mastering-unity-2d-game-development/9781803237077

8. Gregory, J. (2018). *Game engine architecture* (3rd ed.). CRC Press. https://www.routledge.com/Game-Engine-Architecture/Gregory/p/book/9781138035454

9. Hocking, J. (2015). *Unity in action: Multiplatform game development in C#* (2nd ed.). Manning Publications. https://www.manning.com/books/unity-in-action-second-edition.

10. Ilkun, V. P. (2013). Instrumental tools for developing computer games in the action genre. *Donetsk National Technical University Repository*.

http://masters.donntu.org/2013/fknt/ilkun/diss/index.htm

11. Keith, C. (2020). *Agile game development: Build, play, repeat* (2nd ed.). Addison-Wesley Professional. https://www.informit.com/store/agile-game-development-build-play-repeat-9780136527817

12. Liman, M., & Poletaeva, G. N. (2024). Artistic expression in Unity-based game design. *Art and Design Journal*, 8(2), 112–125. https://artdesign.knutd.edu.ua/article/view/123456

13. McShaffry, M., & Graham, D. (2019). *Game coding complete* (5th ed.). Course Technology PTR. https://www.cengage.com/c/game-coding-complete-5e-mcshaffry/9781337558884

14. Nystrom, R. (2021). *Game programming patterns*. Genever Benning. https://gameprogrammingpatterns.com/

15. Okita, A. (2023). *Learning C# by developing games with Unity* (7th ed.). Packt Publishing. https://www.packtpub.com/product/learning-c-by-developing-games-with-unity/9781803243870

16. Poletaeva, G. N. (2019). Evolution of artistic components in computer games. *Art and Design*, 8, 45–58. https://artdesign.knutd.edu.ua/article/view/987654

17. Rogers, S. (2014). *Level up! The guide to great video game design* (2nd ed.). Wiley. https://www.wiley.com/en-us/Level+Up%21+The+Guide+to+Great+Video+Game+Design%2C+2nd+Edition-p-9781118877197

18. Schwab, B. (2009). *AI game engine programming* (2nd ed.). Charles River Media. https://www.cengage.com/c/ai-game-engine-programming-2e-schwab/9781584505723

19. Unity Documentation. (2025). *docs.unity3d.com*. https://docs.unity3d.com/Manual/index.html

20. Unreal Engine Documentation. (2025). *unrealengine.com*. https://docs.unrealengine.com/

21. Zakas, N. C. (2023). Object-oriented programming in Unity game development. *IEEE Transactions on Games*, 15(3), 234–245. https://doi.org/10.1109/TG.2023.3214567

22. Aarseth, E. (2024). Game development methodologies: A comparative study. *Game Studies Journal*, 24(1), 15–30. https://gamestudies.org/2401/articles/aarseth

23. Game Developer. (2025). Best practices for Unity game development. *gamedeveloper.com*. https://www.gamedeveloper.com/production/unity-best-practices

24. Gamasutra. (2024). Iterative design in Unity: Lessons from indie developers. *gamasutra.com*. https://www.gamasutra.com/view/ feature/134567 /iterative_ design_unity.php

25. Lemon School. (2024). How are games developed? Essential skills and tools. *lemon.school*. https://lemon.school/blog/how-games-are-developed

26. Microsoft. (2025). Game development with Unity and C#. *learn.microsoft.com*. https://learn.microsoft.com/en-us/gaming/unity/

27. Schreier, J. (2021). *Press reset: Ruin and recovery in the video game industry*. Grand Central Publishing. https://www.hachettebooks.com/titles/jason-schreier/press-reset/9781538735480/

28. Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson. https://www.pearson.com/store/p/software-engineering/P100000255093

29. Tschang, F. T. (2023). Creative processes in game development: A Unity case study. *Journal of Creative Industries*, 7(1), 67–82. https://doi.org/10.1016/j.jci.2023.100234

30. Wohlin, C., Runeson, P., & Höst, M. (2022). *Experimentation in software engineering* (2nd ed.). Springer. https://www.springer.com/gp/book/9783662589236

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: " Розробка ігор з використанням Unity "

Обсяг пояснювальної записки: 60 аркушів

Дата закінчення бакалаврської роботи 10 червня 2025р.

Підпис студента _____