МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент
_____ Є. О. Давиденко
«___»_____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Ігровий застосунок в жанрі Action RPG на основі рушія Unreal Engine

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ – 409.1910907

Студент _____В. М. Гільщанський
«___»_____ 2023р.

Керівник PhD, ст. викладач кафедри _____І. О. Кандиба
«___»_____ 2023р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
«___»_____ 2023р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії програмного
забезпечення, доцент, канд. тех. наук.
_____Є. О. Давиденко
«_____»_____20__р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Гільщанському Владиславу Миколайовичу_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
____«Ігровий застосунок в жанрі Action RPG на основі рушія Unreal Engine»____
Затверджена наказом по ЧНУ від «17»__березня_____2023р. № ____60_____

2. Строк представлення кваліфікаційної роботи «___»_____20__р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
Розроблено ігровий застосунок в жанрі Action RPG на основі рушія Unreal Engine

4. Перелік питань, що підлягають розробці:
___Дослідити основні принципи Action RPG жанру та визначити його особливості, аналіз конкурентів на ринку ігор в жанрі Action RPG, ознайомитися з можливостями та обмеженнями рушія Unreal Engine, створити графічний дизайн гри, включаючи стилістику та інтерфейс користувача, розробка механік гри, таких як бій, систему навичок, тощо, розробити логіку гри, провести тестування гри та налагодження, виправляючи помилки та удосконалюючи гру_____

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Канд. Техн. наук, доцент Алєксєєва Анна Олександрівна	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи PhD, ст. викладач кафедри ІПЗ Кандиба Ігор Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Гільщанський Владислав Миколайович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____»_____20____р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема:«Ігровий застосунок в жанрі Action RPG на основі рушія Unreal Engine»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	10.03.2023р.	17.03.2023р.	виконано
2.	Огляд літератури за темою роботи	18.03.2023р.	1.05.2023р.	виконано
3.	Аналіз предметної галузі	04.05.2023р.	06.05.2023р.	виконано
4.	Розробка проєктних рішень	07.05.2023р.	15.05.2023р.	виконано
5.	Моделювання та конструювання ПЗ	07.05.2023р.	15.05.2023р.	виконано
6.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	07.05.2023р.	18.05.2023р.	виконано
7.	Розробка спеціальної частини з охорони праці	18.05.2023р	21.05.2023р	виконано
8.	Відгук керівника КРБ	15.06.2023р	16.06.2023р	виконано
9.	Оформлення КРБ та презентації	07.05.2023	18.05.2023р	виконано
10.	Попередній захист	28.05.2023р	29.05.2023р	виконано
11.	Рецензування	16.06.2023р	17.06.2023р	виконано
12.	Завершення оформлення КРБ та презентації	17.06.2023р	19.06.2023р	виконано
13.	Захист кваліфікаційної роботи	26.06.2023р	28.06.23р	виконано

Розробив студент _____Гільщанський В. М._____ _____
(прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

Керівник роботи __PhD, ст. викладач кафедри ІПЗ Кандиба І.О. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок в жанрі Action RPG на основі рушія Unreal Engine»

Студент 409 гр.: Гільщанський Владислав Миколайович

Керівник: PhD, ст. викладач кафедри Кандиба Ігор Олександрович

Об’єктом кваліфікаційної роботи є процес розробки ігрового застосунку в жанрі Action RPG.

Предметом кваліфікаційної роботи є інструментарій рушія Unreal Engine 5 для розробки гри у жанрі Action RPG.

Метою кваліфікаційної роботи є вдосконалення ігрового процесу гри в жанрі Action RPG шляхом розробки застосунку з використанням інструментарію Unreal Engine 5.

Для досягнення цієї мети необхідно вирішити такі завдання:

1. проведення аналізу існуючих ігор в жанрі Action RPG;
2. дослідження функціональних можливостей рушія Unreal Engine 5 та його інструментарію для розробки ігрового застосунку в жанрі Action RPG;
3. розробка концепту ігрового світу та персонажів на основі жанру Action RPG;
4. розробка геймплею, включаючи бої, завдання, локації та інші ігрові елементи;
5. розробка графіки, звукового супроводу та інших аудіовізуальних ефектів;
6. проведення тестування розробленого прототипу.

Робота спрямована на розробку високоякісного ігрового застосунку в жанрі Action RPG на основі рушія Unreal Engine.

Графічна складова гри високої якості, наявні різноманітні геймплейні можливості, використано сучасні технологій розробки ігрових застосунків.

У вступі визначається актуальність теми, предмет та об’єкт роботи та проводиться короткий огляд поставленої задачі.

У першому розділі проводиться аналіз предметної області, розглядаються основні характеристики жанру Action RPG, аналізуються наявні аналоги системи з’ясовуються етапи розробки.

У другому розділі проводиться аналіз вимог до системи та етапів її створення. Надається огляд на історію появи UML, розглядаються основні поняття та типи діаграм в UML та створюються власні.

У третьому розділі представляється потужний стек технологій для розробки ігрових застосунків, описується виконана робота з моделювання.

У четвертому розділі розглядається процес створення меню та інтерфейсу користувача. Реалізовується керуванням персонажем та інші компоненти гри.

У висновках проводиться аналіз проведеної роботи та отриманих результатів.

У спеціальній частині з охорони праці та безпеки в надзвичайних ситуаціях йдеться про техніку безпеки при роботі в офісних приміщеннях із комп’ютерним обладнанням.

КРБ викладена на 66 сторінки, вона містить 4 розділи, 54 ілюстрацій, 2 таблиці, 11 джерел в переліку посилань

Ключові слова: ігровий застосунок, Action RPG, Unreal Engine, графічна складова, геймплей, тестування, ігрові механіки.

ABSTRACT

of the Bachelor's Thesis

«Action RPG game application based on the Unreal Engine»

Student of group 409: Hilshchanskyi Vladyslav Mykolayovych

Supervisor: PhD, Senior lecturer Kandyba Igor Oleksandrovych

The object of qualification work is the process of developing a game application in the genre of Action RPG.

The subject of the qualification work is the tools of the Unreal Engine 5 engine for developing a game in the genre of Action RPG.

The purpose of the qualification work is to improve the gameplay of the game in the genre of Action RPG by developing an application using the Unreal Engine 5 tools.

To achieve this goal, it is necessary to solve the following tasks:

1. analyzing existing games in the genre of Action RPG;

2. studying the functionality of the Unreal Engine 5 engine and its tools for developing an Action RPG game application;

3. development of the concept of the game world and characters based on the Action RPG genre;

4. development of gameplay, including battles, tasks, locations and other game elements;

5. development of graphics, sound and other audiovisual effects;

6. testing of the developed prototype.

The work is aimed at developing a high-quality gaming application in the Action RPG genre based on the Unreal Engine.

The graphic component of the game is of high quality, there are various gameplay features, and modern technologies for developing game applications are used.

The introduction defines the relevance of the topic, the subject and object of the work, and provides a brief overview of the task.

The first section analyzes the subject area, considers the main characteristics of the Action RPG genre, analyzes the existing analogs of the system, and clarifies the stages of development.

The second section analyzes the requirements for the system and the stages of its creation. An overview of the history of UML is provided, the basic concepts and types of diagrams in UML are considered, and your own diagrams are created.

The third section presents a powerful technology stack for the development of gaming applications and describes the modeling work performed.

The fourth section describes the process of creating menus and user interfaces. It implements character controls and other game components.

The conclusion analyzes the work done and the results obtained.

The special section on labor protection and safety in emergency situations deals with safety precautions when working in office premises with computer equipment.

The Bachelor's project is set out on 66 pages, it contains 4 sections, 54 illustrations, 2 tables, 11 sources in the list of references

Keywords: game application, Action RPG, Unreal Engine, graphic component, gameplay, testing, game mechanics.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

NPC – «Non-Player Character» або «Персонаж, керований комп'ютером»

UML – Unified Modeling Language

UE5 – Unreal Engine 5

3D – Тривимірна модель

UDIM – U DIMension – «розмірність по U»

TD – Textel Density

UI – User Interface

ВСТУП

Стрімкий розвиток технологій проклав шлях для розробки численних застосунків і програмних рішень у різних галузях. Ігрові застосунки жанру Action RPG поєднують в собі елементи рольової гри та бойовика. Вони набули величезної популярності серед гравців різного віку.

Ці ігри дозволяють користувачам вирушати в захопливі пригоди, брати участь у битвах і досліджувати величезні віртуальні світи. Оскільки попит на цікаві та візуально захоплюючі ігри продовжує зростати, технології їх створення також стрімко розвиваються та еволюціонують. Саме тому ця робота буде зосереджена на розробці ігрового застосунку жанру Action RPG на основі рушія Unreal Engine 5, що є найбільш сучасним та актуальним ігровим рушієм.

Необхідність нової розробки і удосконалення (модернізації) існуючих ігрових застосунків обґрунтовується результатом аналізу сучасного стану гейміндустрії. На сьогоднішній день існують великі можливості для розробки ігор в жанрі Action RPG, але важливо постійно вдосконалюватись і застосовувати сучасні технології, щоб забезпечити гравцям неперевершений геймплей та емоційний досвід.

Основні проєктні рішення в цій роботі будуть спрямовані на оптимізацію геймплею, покращення візуальної атмосфери гри.

Рушій Unreal Engine забезпечує високу якість графіки, анімації та фізики, що дозволяє розробникам створювати реалістичні та захоплюючі ігрові світи. Також створення ігрових застосунків є актуальною задачею у сфері розваг та розвитку комп'ютерних технологій.

Об’єктом кваліфікаційної роботи є процес розробки ігрового застосунку в жанрі Action RPG.

Предметом кваліфікаційної роботи є інструментарій рушія Unreal Engine 5 для розробки гри у жанрі Action RPG.

Метою кваліфікаційної роботи є вдосконалення ігрового процесу гри в жанрі Action RPG шляхом розробки застосунку з використанням інструментарію Unreal Engine 5.

Для досягнення цієї мети необхідно вирішити такі завдання:

1. проведення аналізу існуючих ігор в жанрі Action RPG;
2. дослідження функціональних можливостей рушія Unreal Engine 5 та його інструментарію для розробки ігрового застосунку в жанрі Action RPG;
3. розробка концепту ігрового світу та персонажів на основі жанру Action RPG;
4. розробка геймплею, включаючи бої, завдання, локації та інші ігрові елементи;
5. розробка графіки, звукового супроводу та інших аудіовізуальних ефектів;
6. проведення тестування розробленого прототипу.

Об’єктом дослідження є процес розробки ігрового застосунку в жанрі Action RPG.

Результати цієї роботи можуть бути застосовані в галузі розробки ігор, особливо в жанрі Action RPG. Розроблений ігровий застосунок може бути використаний як продукт для комерційного випуску або як демонстраційний матеріал для презентації можливостей Unreal Engine у галузі ігрової розробки.

1 АНАЛІЗ ІГРОВИХ ЗАСТОСУНКІВ В ЖАНРІ ACTION RPG

1.1 Дослідження особливостей ігрового жанру Action RPG

Action RPG (або Action Role-Playing Game) – це жанр комп'ютерних ігор, де гравці можуть зануритися у фантастичні світи, воювати зі злом, розблоковувати нові навички та знаряддя бою та досліджувати світ, виконуючи різні завдання.

Жанр ARPG бере свій початок у східній традиції, з таких ігор, як Panorama Toh та Bokosuka Wars (рис. 1.1) 1983 року, The Tower of Druaga, Dragon Slayer, Hydlide та Courageous Perseus, випущені у 1984 році, відіграли важливу роль у створенні основних правил та механік жанру. Ці ігри надихнули на створення The Legend of Zelda (1986), впливової пригодницької гри в жанрі Action-Adventure. У відповідь, конкуруючі серії, такі як Dragon Quest та Final Fantasy, закріпили механіку покрокових боїв, хоча Final Fantasy пізніше запровадила власну систему активних боїв, яка додала більше елементів екшену. З часом різниця між покроковими рольовими іграми та бойовиками на Сході стала більш розмитою, а сучасні ігри поєднують елементи обох стилів [1].

Рисунок 1.1 – Приклад ігрового процесу гри Bokosuka Wars

В1996 році вийшла гра Diablo, яка стала однією з найвідоміших Action RPG ігор у світі та принесла розробникам чималий прибуток. Гра мала швидкий темп, багато ворогів та велику кількість зброї та спеціальних навичок, що робило гру дуже захоплюючою.

Іншим відомим прикладом Action RPG є серія The Elder Scrolls, яка почалася з гри Arena (1994) та продовжилася у Daggerfall (1996), Morrowind (2002), Oblivion (2006) та Skyrim (2011). Ці ігри поєднували в собі відкритий світ, детально пророблені персонажі та діалоги з великою кількістю активного бою.

Рисунок 1.2 – Приклад ігрового процесу гри The Elder Scrolls: Arena

З часом жанр Action RPG продовжує розвиватися та отримує все більше популярності. Сучасні Action RPG ігри, такі як Dark Souls (2011), Bloodborne (2015), Nioh (2017), Monster Hunter: World (2018) та багато інших, надають гравцям ще більші можливості для насолоди динамічним геймплеєм та пошуків інноваційних способів розвитку своїх персонажів [2].

1.2 Аналіз особливостей ігрового застосунка в жанрі Action

Процес розробки ігрового застосунку в жанрі Action RPG є складним та багатоетапним. Він містить в собі наступні основні етапи:

1. *Концептуалізація*: це етап визначення концепції гри, включаючи жанр, геймплей, механіки та стиль. Також на цьому етапі зазвичай визначається аудиторія гри та її цільова аудиторія.

2. *Проєктування*: це етап створення концепції гри, її головних механік, інтерфейсу користувача, а також інших важливих елементів. На цьому етапі зазвичай створюються сценарії, концептуальні малюнки та перші ігрові прототипи.

3. *Розробка*: це етап створення фактичної гри. На цьому етапі розробники створюють графічні елементи, програмний код, звукову та музичну складові, анімацію та інші елементи гри.

4. *Тестування*: це етап перевірки різних аспектів гри, включаючи геймплей, функціональність та взаємодію з користувачем. На цьому етапі зазвичай використовуються тестові групи користувачів, щоб отримати фідбек щодо вдосконалення гри.

5. *Реліз*: це етап запуску гри. На цьому етапі розробники завантажують гру на різні платформи та дозволяють користувачам придбати її.

Після релізу розробники можуть продовжувати підтримувати гру, випускаючи оновлення та патчі, що виправляють помилки та додавання нового контенту [3]. Також можуть проводитись різні маркетингові заходи, щоб привернути увагу до гри та збільшити її продажі. Крім того, можуть початися роботи над наступною грою або проєктом.

Функціональні особливості процесу розробки ігрового застосунку в жанрі Action RPG пов'язані з особливостями геймплею та механіками, що характерні для цього жанру.

1. *Розробка механіки бою та інтерактивних елементів*: для гри у жанрі Action RPG важливо мати добре пророблену систему бою, яка повинна бути динамічною та дозволяти гравцю використовувати різноманітні прийоми та вміння. Крім того, інтерактивні елементи, такі як розбиття предметів та інші ефекти, також можуть бути також важливими наприклад у битві з босами.

1. *Система розвитку персонажа*: у іграх жанру Action RPG велику роль відіграє система розвитку персонажа, що дозволяє гравцеві покращувати навички та характеристики свого героя, що робить його більш потужним у бою.

2. *Генерація випадкових елементів гри*: для збагачення геймплею гри у жанрі Action RPG може бути використана система генерації випадкових подій, ворогів, предметів та інших елементів гри.

3. *Система квестів та місій*: наявність квестів та місії, які гравець повинен виконувати, щоб розвивати історію гри, при цьому отримуючи різноманітні винагороди за їх виконання.

4. *Система поведінки ворогів та NPC*: розробка системи поведінки повинна бути добре проробленою, щоб вороги та NPC виконували свої завдання, реагували на дії гравця та взаємодіяли з іншими елементами гри.

5. *Система анімації*: важливою функціональною особливістю є створення анімацій для кожної існуючої дії гравця, що додає імерсивності та привабливості грі.

6. *Система збереження та завантаження гри*: для зручності гравців у іграх жанрі Action RPG важливо мати систему збереження та завантаження гри, що дозволяє їм продовжувати гру з того місця, на якому вони зупинилися.

7. *Розробка ігрового світу та його різноманітних локацій*: у жанрі Action RPG важливо мати великий та детально розроблений ігровий світ з різноманітними локаціями.

1.2 Аналіз існуючих аналогів в жанрі Action RPG

Mortal Shell – це Action RPG гра, яка випробовує ваш здоровий глузд і стійкість у зруйнованому світі. Ваші Вороги дуже швидкі та спритні, а виживання вимагає неперевершеної обізнаності, точності та сконцентрованості. Гравець має повертати загублених воїнів, обстежувати потаємні святилища і перемагати грізних ворогів.

Розробник: Cold Symmetry (випущено Playstack)

Архітектура: Desktop application (гра)

Мова реалізації: C++ (рушій Unreal Engine 4)

Рисунок 1.3 – Ігровий процес Mortal Shell

Перелік функцій/характеристик:

1. гравець керує безтілесним духом, який може пересісти в оболонки мерців та використовувати їх вміння та зброю;
2. система бою, яка базується на ухилі, блокуванні та контратаках;

3. гра відрізняється високим рівнем складності та вимагає від гравця стратегічного мислення та вивчення ворожих атак;

4. містична атмосфера гри, яка приваблює гравців своїм химерним світом, розповідями та графікою;

5. гра доступна на платформах PlayStation 4, Xbox One, Microsoft Windows.

Переваги:

1. унікальна система гри, що відрізняється від інших відеоігор жанру "Dark Souls";

2. графіка та атмосфера гри дозволяють гравцеві підірвати свої емоції та зануритися у світ гри;

3. високий рівень складності та стратегічного мислення роблять гру викликом для гравців, що цікавляться жанром;

4. пересідання у різних оболонках мерців дає гравцеві широкий вибір стилів гри та інші вміння.

Недоліки:

1. гра складна та може викликати відчуття невдачі, що може відштовхувати деяких гравців;

2. гра має деякі технічні проблеми та баги, які необхідно виправити;

3. гра не підтримує мультиплеєр.

4. деякі гравці вважають, що грі бракує різноманітності в плані типів ворогів, оточення та загальної механіки гри. Вони можуть стверджувати, що через деякий час гра стає одноманітною, з обмеженою кількістю сюрпризів або нових викликів.

Джерело інформації: https://www.mortalshell.com/

Tales of Arise – японська рольова гра, події якої відбуваються у процвітаючому світі. Сюжет починається з двох народжених у різних світах людей, які хочуть змінити свою долю та створити нове майбутнє.

Розробник: Bandai Namco Entertainment

Архітектура: Desktop application (гра)

Мова реалізації: C++ (рушій Unreal Engine 4)

Рисунок 1.4 – Ігровий процес Tales of Arise

Перелік функцій/характеристик:

1. гравці можуть вибрати одного з декількох головних героїв та відправитися в епічну подорож, щоб збирати інформацію про світ та боротися з ворогами;
2. у грі є система бойових мистецтв, яка дозволяє гравцям виконувати різноманітні атаки та комбо;
3. гра має вражаючу графіку та музику, що допомагають поглибити імерсію в світ гри;

4. у грі є можливість розвивати навички та збирати предмети для поліпшення героїв та їхнього спорядження;

5. гра доступна на платформах PlayStation 4, Xbox One, Microsoft Windows.

Переваги:

1. ігровий світ гри детально розроблений та цікавий;

2. система бойових мистецтв дає гравцям велику свободу дій та можливість експериментувати з різноманітними атаками;

3. гра має гарну графіку та звуковий супровід;

4. велика кількість предметів та навичок дозволяє гравцям розвивати своїх героїв за власними уподобаннями;

5. гра має цікавий сюжет та гарну характеризацію персонажів.

Недоліки:

1. гра може виявитись складною для новачків у жанрі рольових ігор;

2. у грі можуть виникнути проблеми з оптимізацією на старіших комп'ютерах;

3. деякі гравці можуть вважати, що гра має деякі стандартні елементи жанру та не пропонує нічого нового або унікального.

Джерело інформації: https://www.bandainamcoent.com/games/tales-of-arise

Monster Hunter Rise – це Action RPG гра, шоста основна частина серії Monster Hunter, де гравець стає мисливцем, досліджуватимете абсолютно нові світи та використовуватимете різноманітну зброю, щоб знищити жахливих монстрів у рамках основної сюжетної лінії.

Розробник: Capcom

Архітектура: Desktop application (гра)

Мова реалізації: невідомо (рушій RE Engine)

Рисунок 1.5 – Ігровий процес Monster Hunter Rise

Перелік функцій/характеристик:

1. гравець грає за мисливця на монстрів, який виконує завдання та полює на монстрів, використовуючи різні зброї та інструменти;
2. гра має велику кількість монстрів та локацій, які можуть бути досліджені гравцем;
3. гра має розвинутий розділ крафту, де гравець може створювати нову зброю та екіпірування з матеріалів, зібраних від монстрів;
4. у грі є розвинута система боїв та уникнення атак монстрів;
5. гра має режим мультиплеєра, де гравці можуть допомагати один одному в полюванні на монстрів.

Переваги:

1. чудова графіка та анімація, що створюють реалістичну атмосферу гри;
2. багато монстрів та локацій, які дозволяють гравцеві досліджувати світ гри та збирати різні матеріали;
3. розвинута система крафту та екіпірування, що дає гравцю можливість створювати унікальні зброї та екіпірування для максимальної ефективності в полюванні на монстрів;
4. режим мультиплеєра дозволяє гравцям допомагати один одному в полюванні на монстрів, що робить гру більш соціальною та цікавою.

Недоліки:

1. для деяких гравців гра може виявитись складною та вимагати великої кількості часу та зусиль для досягнення успіху в грі;
2. деякі аспекти гри можуть бути важкими для новачків, тому їм може знадобитись додатковий час для ознайомлення з механікою гри та її правилами.

Джерело інформації: https://www.monsterhunter.com/rise/

1.3 Специфікації вимог до ігрового застосунку

Призначення системи: надання можливості гравцям брати участь у захоплюючих боях та існувати в унікальному віртуальному світі.

Сфера застосування: сфера розваг, реклама та маркетинг, розвиток ігрової індустрії.

Характеристики користувачів: користувачі віком від 18 років, повинні мати ПК та доступ до мережі Інтернет.

Загальна структура і склад системи:

Функції системи:

1. система управління персонажем такими як рух, удари, тощо;
2. система бойових механік включаючи різні види зброї, бойові прийоми, ефекти тощо;
3. система розвитку персонажа, яка дозволяє гравцю підвищувати рівень свого персонажа;

4. система збору ресурсів;
5. система інвентарю;
6. система поведінки для противників;
7. система звуку та музики;
8. система розподілу очок навичок та характеристик;
9. система здоров'я та енергії, що відновлюються під час гри;
10. система збереження гри, що дозволяє гравцеві продовжувати гру з місця, де він завершив попередній сеанс;

Вимоги до технічного забезпечення:

- операційна система: Windows 10 або вище;
- процесор: Intel Core i5-6600K або AMD Ryzen 5 1600X;
- відеокарта: NVIDIA GeForce GTX 970 або AMD Radeon R9 390X;
- оперативна пам'ять: не менше 8 ГБ;
- місце на жорсткому диску: не менше 50 ГБ.

Архітектура програмної системи: складається з клієнтської частини.

Системне програмне забезпечення:

Для розробки ігрового застосунку на UE5 необхідно встановити такі системні програмні засоби, як Windows 10 або вище, DirectX 12 або вище, Visual Studio 2019 або вище, а також оновлення для Unreal Engine 5.

Мова і технологія розробки ПЗ: Рекомендовано використовувати мову програмування C++ або систему Blueprints.

Інтерфейс користувача: Інтерфейс користувача має бути інтуїтивно зрозумілим та зручним для використання. Необхідно створити добре продуманий інтерфейс, який дозволить користувачам легко керувати персонажем, взаємодіяти з ігровим світом та керувати настройками гри.

Висновки до розділу 1

У розділі було наведено огляд жанру Action RPG (або Action Role-Playing Game), що є популярним жанром комп'ютерних ігор. Він дає можливість гравцям зануритись у фантастичні світи, боротись зі злом, розблоковувати нові навички і знаряддя бою та досліджувати світ, виконуючи різні завдання. Історія появи жанру пов'язана з розвитком комп'ютерних ігор загалом, починаючи з текстових пригод, а потім з'явилися графічні ігри з елементами ролевих ігор.

Описано також кілька Action RPG ігор, таких як Diablo та серія The Elder Scrolls, які стали відомими та успішними у своєму жанрі. Розглянуто процес розробки ігрового застосунку в жанрі Action RPG, який включає етапи концептуалізації, проєктування, розробки, тестування та релізу. Після релізу гра може продовжувати підтримуватися оновленнями та маркетинговими заходами.

Досліджено функціональні особливості процесу розробки Action RPG, зокрема розробку механіки бою та інтерактивних елементів, систему розвитку персонажа, генерацію випадкових елементів гри, систему квестів та місій, систему поведінки ворогів та NPC, систему анімації, систему збереження та завантаження гри, розробку ігрового світу та локацій.

Проаналізовано відібрані три аналоги ігрових застосунків такі як: Mortal Shell, Tales of Arise та Monster Hunter Rise. Досліджено їх перелік функцій та характеристик, а також виокремлено основні переваги та недоліки.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ІГРОВОГО ЗАСТОСУНКУ

2.1 Загальні засади моделювання ПЗ

У 1990-х роках об'єднання трьох авторів, Грейді Буча (Grady Booch), Івара Якобсона (Ivar Jacobson) та Джеймса Рамбо (James Rumbaugh), вийшли на передній план у розробці об'єктно-орієнтованих методів проєктування. Кожен з цих авторів мав свою власну методологію: Буч - Object-Oriented Design (OOD), Якобсон - Object-Oriented Analysis (OOA), а Рамбо - Object Modeling Technique (OMT).

У 1994 році ці три автори почали працювати разом над спільною методологією проєктування. Це призвело до створення UML (Unified Modeling Language) - мови моделювання, яка поєднала в собі найкращі аспекти методологій Буча, Якобсона та Рамбо.

Отже, UML – це мова, яка дозволяє розробникам програмного забезпечення описувати та проєктувати складні системи за допомогою графічних зображень та текстових описів, вона є стандартом для візуалізації, специфікації, побудови та документування програмних систем [4].

Основні зміни, які були внесені в різні версії UML, можна описати таким чином:

UML 1.x:

1. перша версія UML була випущена в 1997 році і містила 9 видів діаграм;
2. UML 1.x була спрямована на моделювання статичної структури системи, зокрема класів, об'єктів, пакетів та компонентів;
3. в UML 1.x також вводилися поняття асоціації, агрегації та композиції, які дозволяли описувати відносини між об'єктами.

UML 2.x:

1. UML 2.0 була випущена в 2005 році і містила більше 20 видів діаграм.
2. UML 2.x розширювала функціональність мови, дозволяючи моделювати більш складні аспекти систем, такі як поведінка та протоколи.
3. в UML 2.x було введено поняття профілю, яке дозволяло розширювати мову для моделювання конкретних типів систем.

4. UML 2.x також включала підтримку для моделювання відносин між різними системами та компонентами.

Основні зміни в UML 2.x включають такі нововведення, як State Machine Diagrams, Composite Structures, Timing Diagrams, Interaction Overview Diagrams, Model Driven Architecture (MDA) та багато іншого.

Загалом, UML 2.x стала потужним засобом для моделювання складних систем та допомагає розробникам програмного забезпечення краще розуміти та взаємодіяти зі своїми системами.

2.2 Діаграма варіантів використання

Діаграма варіантів використання є важливим інструментом аналізу вимог для проєктування системи. Вона використовується для ідентифікації потенційних дійових осіб та їх взаємодії з системою, а також для опису функціональної поведінки системи. Діаграма варіантів використання допомагає проєктній команді зрозуміти, як користувачі будуть взаємодіяти з системою, та допомагає визначити, які функції повинні бути реалізовані для досягнення мети користувачів.

Отже, діаграма варіантів використання (Use Case diagram) – це графічне зображення, яке використовується для опису функціональної поведінки системи з точки зору користувачів. Вона включає в себе дієві особи (actors), варіанти використання (use cases) та їх взаємозв'язки.

Дієві особи включають користувачів, які взаємодіють з системою, а також будь-які зовнішні системи, які можуть взаємодіяти з нашою системою. Варіанти використання повинні відображати всі можливі функції, які система може здійснити для користувача. Кожен варіант використання має бути названий таким чином, щоб він максимально точно відображав його функцію.

Варіанти використання (use cases) – це описи функціональної поведінки системи з точки зору користувачів. Вони описують те, що система може робити для користувача, який викликає діяльність, що виконується в системі. Кожен варіант використання зображується у діаграмі варіантів використання у вигляді овалу з назвою [4].

Сценарій - це конкретний варіант використання, який описує послідовність дій, які користувач виконує для досягнення певної мети з використанням системи.

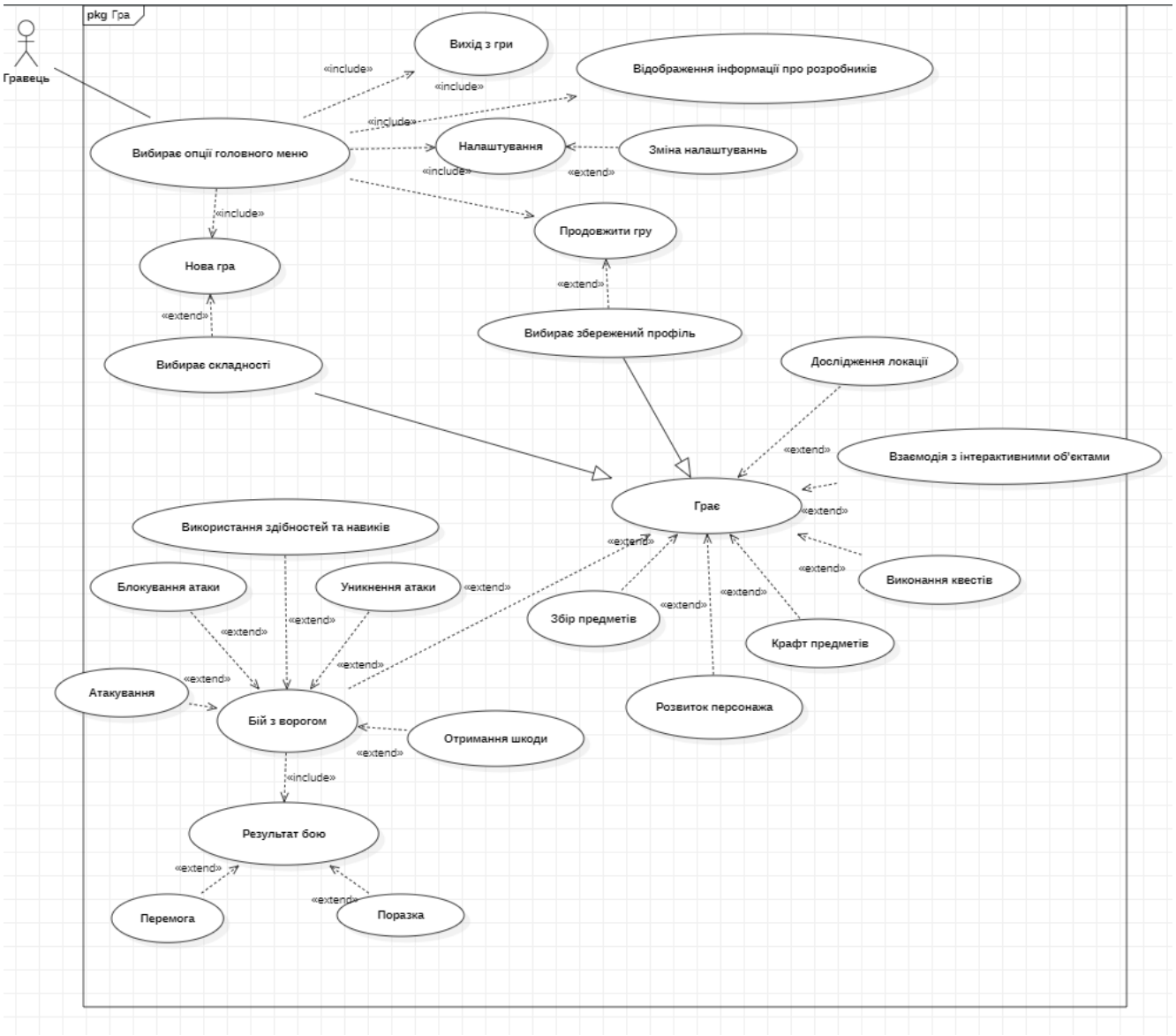

Рисунок 2.1 – Діаграма варіантів використання

Короткий usercase:

Користувач запускає ігровий застосунок на своєму пристрої, після чого він відкривається головне меню гри (стартовий екран). Він може обрати різні опції, такі як створення нової гри або продовження збереженої та інші доступні опції. Він обирає нову гру і вибирає режим складності, після чого починається сама гра, де гравець з’являється на стартовій локації та починає взаємодіяти з ігровим світом.

Поверхневий usercase:

Головний сценарій (успішний)

Гравець запускає ігровий застосунок на своєму пристрої. На екрані з'являється головне меню гри, де гравець обирає опцію "Продовжити гру". Гравець обирає свій збережений профіль, в якому зберігаються всі його досягнення та прогрес. Гравець знаходить себе в ігровому світі, де може розглядати своє обладнання та інвентар, змінювати налаштування гри та виконувати різні завдання. Гравець вибирає своє завдання або місію, використовуючи інтерфейс гри. Гравець виконує різні завдання, включаючи боротьбу зі монстрами, збирання ресурсів, дослідження світу гри тощо. Гравець збирає різні предмети, які можуть бути використані для поліпшення його персонажа. Гравець здійснює крафт, щоб отримати необхідні предмети. Гравець здійснює розвиток свого персонажа, поліпшуючи його характеристики та вміння. Гравець продовжує гру, виконуючи різні завдання та досліджуючи світ гри, намагаючись досягти максимального рівня досягнень та прогресу.

Альтернативні сценарії:

1. гравець обирає опцію "Нова гра", обирає складність та починає грати з початку. Він також може зберегти свій прогрес на новий слот профілю, щоб продовжити гру з поточного місця. Після чого він буде мати два профіля з різним прогресом гри;
2. гравець обирає опцію опцію "Налаштування", щоб змінити графічні налаштування, звукові ефекти, чутливість контролера тощо;
3. гравець вибирає опцію "Вийти з гри", коли закінчує грати, щоб зберегти свій прогрес та вийти з гри;
4. гравець обирає опцію "Credits", щоб переглянути список розробників та інших працівників, які працювали над грою;
5. гравець обирає опцію "Продовжити гру", але з’являється вікно з помилкою.
6. гравець може спробувати перезавантажити гру, щоб вирішити проблему або звернутися в технічну підтримку.

Повний usercase

Use Case Name: Гравець розпочинає нову гру в жанрі Action RPG

Scope: System

Level: User-goal

Primary Actor: Гравець

Stakeholders and interests:

- гравець: бажає грати в нову гру в жанрі Action RPG;
- розробники: бажають, щоб гравець зміг успішно почати гру і насолоджуватись її геймплеєм.

Preconditions:

- користувач має доступ до комп'ютера або мобільного пристрою;
- на пристрої встановлено застосунок на основі рушія Unreal Engine;
- гравець має створений обліковий запис в грі.

Success Guarantee:

- гравець успішно запустить нову гру в жанрі Action RPG на пристрої;
- гравець буде знайомий з основними елементами гри і зможе продовжити її геймплей.

Main Success Scenario:

- гравець запускає застосунок на пристрої;
- гравець обирає опцію "Нова гра" в меню гри;
- система пропонує гравцю вибрати рівень складності гри;
- гравець обирає бажаний рівень складності;
- система генерує світ для гравця;
- гравець потрапляє в ігровий світ і отримує контроль над головним героєм;
- гравець знайомиться зі стартовою локацією і інтерфейсом гри;
- гравець розпочинає дослідження ігрового світу та боротьбу зі ворогами;
- гравець виконує поставлені завдання та отримує додаткові нагороди;
- гравець може зберегти гру та продовжити геймплей пізніше.

Extensions:

1а. Гравець відмовляється вибирати рівень складності і повертається до меню гри.

2а. Гравець не може обрати бажаний рівень складності:

- система пропонує гравцю обрати менш високий рівень складності;
- гравець обирає менш високий рівень складності;
- гравець повертається до кроку 5 головного сценарію;

3а. Гравець не успішно розпочинає гру через технічні проблеми:

- система виводить повідомлення про проблему та пропонує спробувати ще раз;
- гравець повторює спробу ще раз;
- якщо проблема не вирішена, гравець звертається до технічної підтримки.

Special Requirements:

- гра повинна мати різні рівні складності, які будуть відповідати різним рівням ігрового досвіду гравців;
- гра повинна мати систему автоматичного збереження прогресу гравця;
- гра повинна мати систему навчання, яка допоможе гравцеві ознайомитись з основними елементами гри.

Technology and Data Variations List:

Рекомендується використовувати Unreal Engine 5 для розробки гри.

Frequency of Occurrence:

Частота виконання usecase при користуванні системою залежить від кількості гравців, які користуються

Miscellaneous:

Можна додати мультиплеєрний режим для гри з друзями.

2.3 Діаграми взаємодії

Аналіз системи – це процес розуміння і вивчення функціональних вимог і поведінки системи. Під час аналізу системи часто використовуються діаграми взаємодії між об'єктами, які можуть бути послідовністю або кооперативними.

Діаграми взаємодії між об'єктами – це тип діаграм, що використовується для моделювання взаємодії між об'єктами системи. Ці діаграми показують, які об'єкти беруть участь у виконанні конкретного варіанту використання, які повідомлення передаються між об'єктами та які послідовності виконання дій.

Діаграми взаємодії між об'єктами можуть бути послідовності або кооперативними. Діаграми послідовності відображають послідовність повідомлень, які передаються між об'єктами для виконання конкретного варіанту використання.

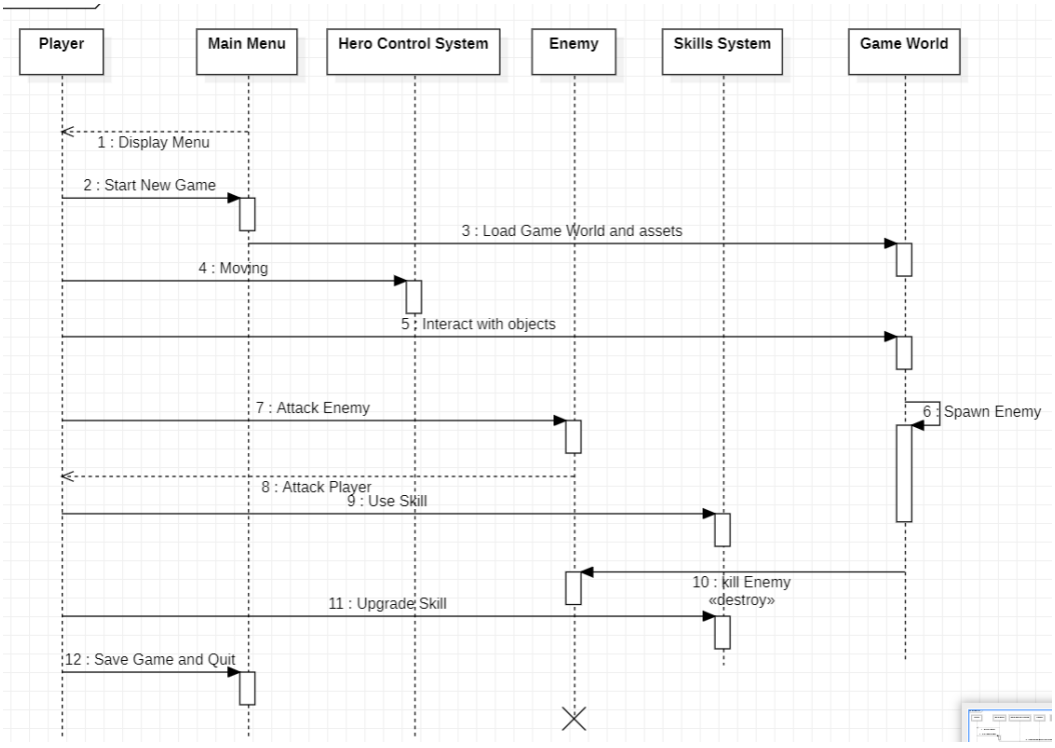

Рисунок 2.2 – Sequence діаграма «Gameplay»

Від «Main Menu» до «Player» - Reply Message: Display Menu, гравець запустив гру. Гра відобразила головне меню.

Від «Player» до «Main Menu» - Message: Start New Game, гравець обирає потрібний пункт меню.

Від «Main menu» до «Game World» - Message: Load Game World and Assets, завантаження рівня та його асетів.

Від «Player» до «Hero Control System» - Message: Moving, керує героєм.

Від «Player» до «Game World» - Message:Interact with objects, взаємодія з ігровим світом.

Від «Game World» до «Game World» - Self Message:Spawn Enemy, генерація ворогів.

Від «Player» до «Enemy» - Message:Attack Enemy, бій з ворогом.

Від «Enemy» до «Player» - Reply Message:Attack Enemy, бій з героєм.

Від «Player» до «Skill System» - Message:Use Skill, використати навичку.

Від «Game World» до «Enemy» - Message:Kill Enemy, прибирання Enemy з рівня після його вбивства

Від «Player» до «Skill System» - Message: Upgrade Skill, покращити навичку.

Від «Player» до «Main Menu» - Message: Save Game and Quit, гравець виходить з гри.

Кооперативні діаграми показують взаємодію між об'єктами, яка може включати паралельну обробку повідомлень або синхронізацію між об'єктами [4].

2.4 Діаграма станів

Діаграма станів – це тип діаграми, що дозволяє описувати поведінку об'єкта чи системи та переходи між різними станами. Вона може бути використана для опису процесу функціонування, коли необхідно відслідковувати зміну стану системи від початку до кінця процесу [4].

Для створення діаграми станів необхідно:

1. Визначити всі можливі стани, в які може переходити система. Ці стани можуть бути визначені як інтервали часу, коли система знаходиться в певному стані або як окремі стани, що визначаються певними умовами.

2. Визначити всі можливі переходи між станами. Ці переходи можуть бути залежні від різних умов, таких як вхідні дані, рішення користувача або внутрішні стани системи.

3. Надати кожному стану та переходу назву, щоб їх можна було ідентифікувати.

Діаграма станів може бути використана для опису процесу функціонування, наприклад, процесу замовлення товару в інтернет-магазині.

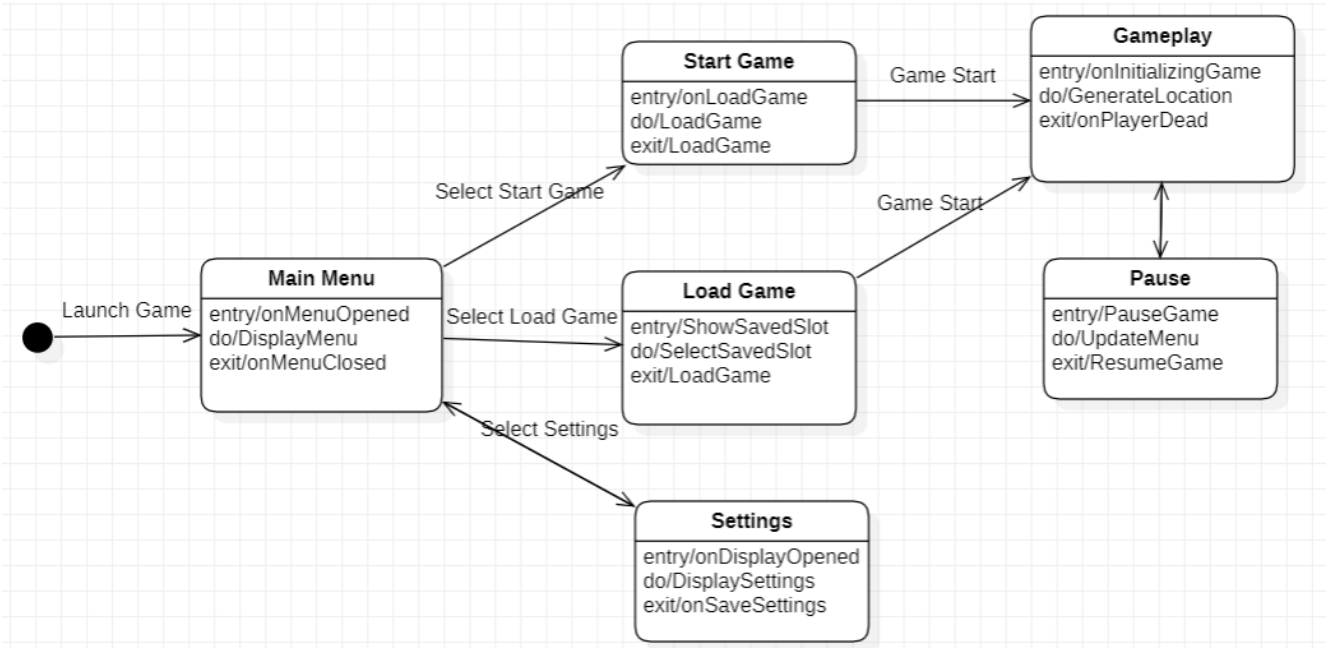

Рисунок 2.3 – Діаграма стану в цілому

В цій діаграмі (рис. 2.3) є декілька станів, стан головного меню, в якому перебуває користувач після запуску гри, в свою чергу меню може мати декілька станів, які змінюються в залежності від вибраної опції.

Стани Start/Load Game запускають сам ігровий процес, в якому гравець може викликати такий стан як Pause.

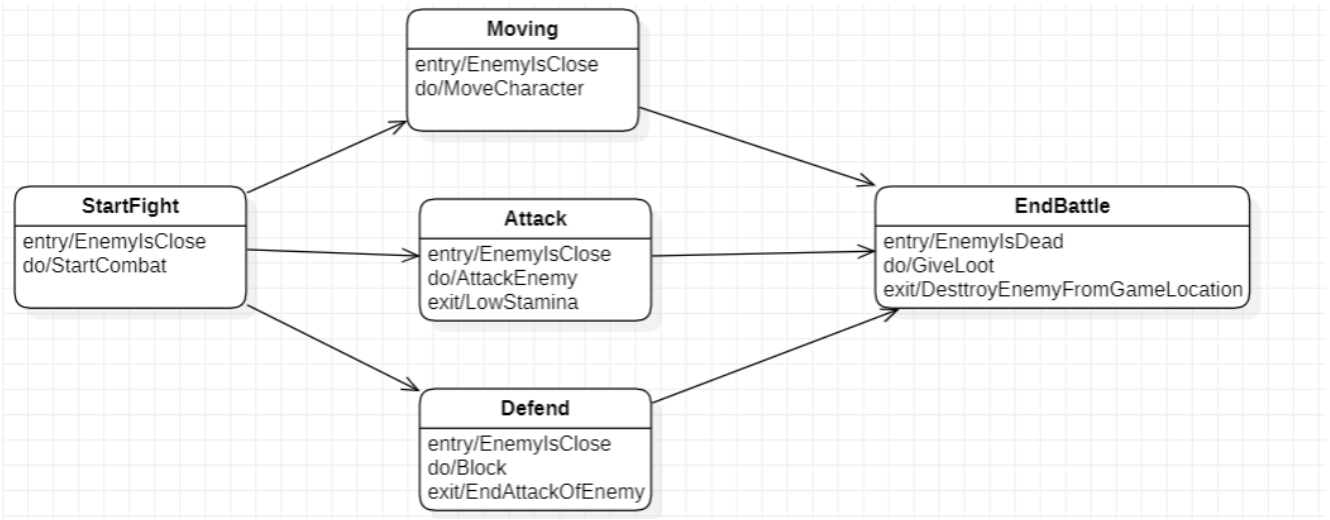

Рисунок 2.4 – Діаграма стану «Бій з ворогом»

В цій діаграмі (рис. 2.4) є декілька станів бою з ворогом, головним тригером є те що ворог дуже близько з героєм. Закінчення стану бій з ворогом є смерть ворога.

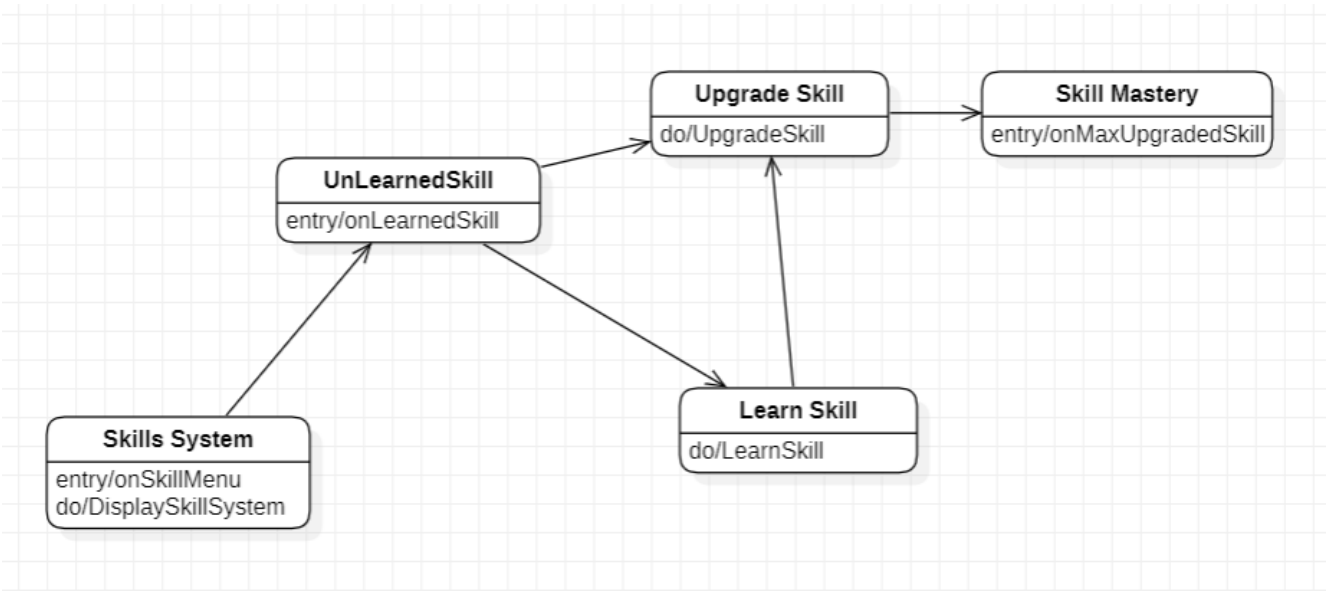

Рисунок 2.5 – Діаграма стану «Skill System»

В цій діаграмі (рис. 2.5) є декілька станів Skill System.

1. Стан "Skills System": це початковий стан, в якому гравець може переглянути всі доступні навички та їх властивості.

2. Стан "Unlearned Skill": це стан, в якому гравець може вибрати навичку, яку він хоче вивчити, але поки що не має можливостей.

3. Стан "Learn Skill": це стан, в якому гравець може вивчити нову навичку.

4. Стан "Upgrade Skill": це стан, в якому гравець може покращити свої наявні навички.

5. Стан "Skill Mastery": це стан, в якому гравець покращив вивчену навичку до максимального рівня, отримуючи додаткові бонуси та можливості в грі.

2.5 Діаграма діяльності

Діаграми діяльності є одним із видів UML діаграм та використовуються для моделювання поведінки системи або її частини, зокрема для опису послідовності дій, які виконує система, або опису процесу прийняття рішень.

На діаграмах діяльності представлено виконання операцій у вигляді потоку керування, де операції зображуються у вигляді прямокутників зокрема з урахуванням їхньої назви та можливих вхідних параметрів. Стрілки, що з'єднують ці операції, відображають порядок виконання. Крім того, на діаграмах діяльності можуть бути відображені різноманітні елементи, такі як різні види вітвлень, злиття та інші.

Також часто виникає питання, у чому різниця між діаграмою діяльності та послідовності. Головна мета діаграми послідовності – показати порядок виконання, або послідовність дій. Водночас діаграма діяльності потрібна для опису роботи всієї системи, вона показує перехід від однієї дії до іншої.

Ці дії можуть виконуватися людьми, програмними компонентами або комп'ютерами. Потік керування (порядок виконання) на діаграмі діяльності переходить від однієї операції до іншої.

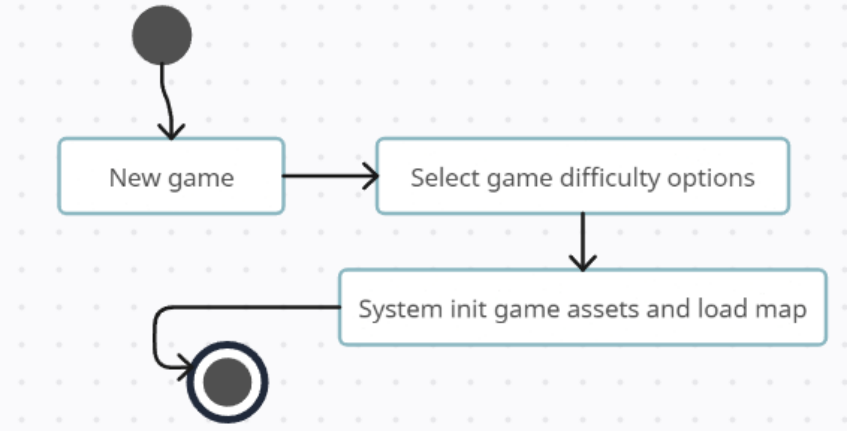

Рисунок 2.6 – Діаграма діяльності «Start a New Game»

Пояснення:

1. користувач обирає опцію "Нова гра" з головного меню;
2. система відображає список варіантів складності гри;
3. користувач обирає складність гри;
4. система ініціалізує ігрові ресурси, такі як персонаж гравця та вороги;
5. система завантажує ігрову карту для нової ігрової сесії;

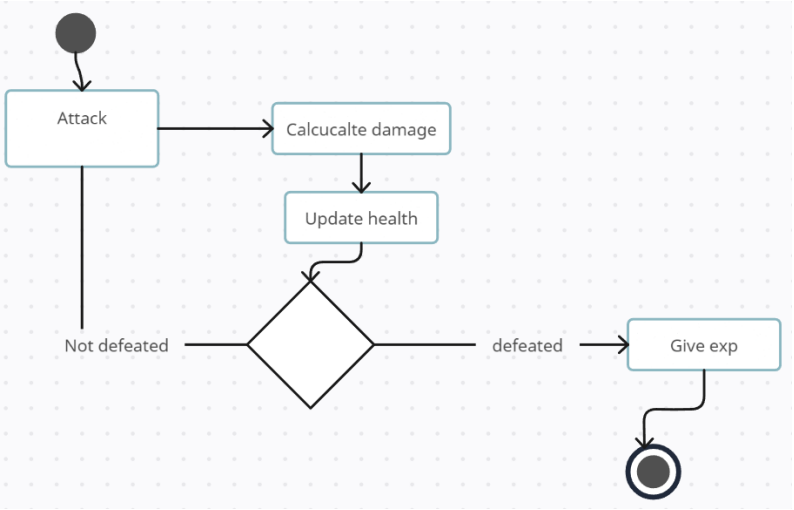

Рисунок 2.7 – Діаграма діяльності «Attack an Enemy»

Пояснення:

1. користувач обирає опцію "Атакувати" під час бою;

2. система розраховує шкоду, яку персонаж гравця завдає ворогу, виходячи з різних факторів;

3. система оновлює здоров'я ворога на основі завданої шкоди;

4. система перевіряє, чи був ворог переможений;

5. якщо ворога переможено, система нагороджує очками досвіду;

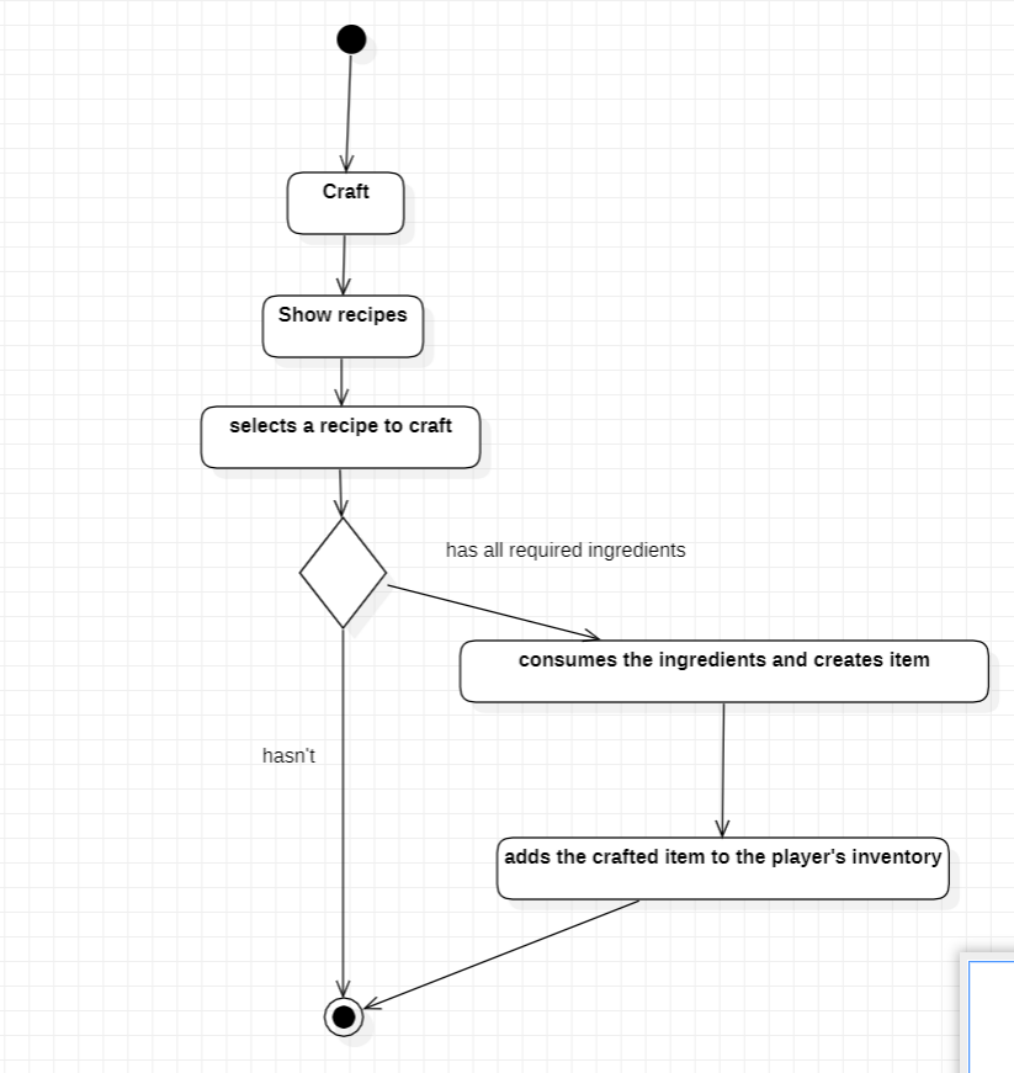

Рисунок 2.8 – Діаграма діяльності «Craft Item»

Пояснення:

1. користувач обирає опцію "Ремесло" в меню;

2. система показує список доступних рецептів крафту;

3. користувач обирає рецепт для крафтингу;

4. система перевіряє, чи є у персонажа гравця всі необхідні інгредієнти для обраного рецепта;

5. якщо у персонажа гравця є всі необхідні інгредієнти, система споживає інгредієнти і створює крафтовий предмет;

6. система додає створену річ до інвентарю персонажа.

Висновки до розділу 2

У розділі було представлено огляд історії розвитку мови моделювання UML (Unified Modeling Language) та її основних версій - UML 1.x і UML 2.x. Показано, що UML є потужним інструментом для моделювання складних систем і допомагає розробникам програмного забезпечення краще розуміти та взаємодіяти зі своїми системами.

Описано, що UML 1.x була спрямована на моделювання статичної структури системи, включаючи класи, об'єкти, пакети та компоненти, а також введено поняття асоціації, агрегації та композиції. УML 2.x, у свою чергу, розширила функціональність мови, дозволяючи моделювати більш складні аспекти систем, такі як поведінка та протоколи. Також введено поняття профілю, яке дозволяє розширювати мову для моделювання конкретних типів систем, і підтримку для моделювання відносин між різними системами та компонентами.

Викладено основні поняття діаграм в UML, зосереджуючись на діаграмі варіантів використання, діаграмах взаємодії між об'єктами та діаграмах станів. Діаграма варіантів використання використовується для ідентифікації потенційних дійових осіб та їх взаємодії з системою, а діаграми взаємодії між об'єктами дозволяють моделювати взаємодію між об'єктами системи за допомогою послідовностей або кооперативних діаграм. Діаграма станів дозволяє описати поведінку об'єкта чи системи та переходи між різними станами.

Також було описано сценарії використання у різних формах (коротка, поверхнева та повна).

3 МОДЕЛЮВАННЯ ЕЛЕМЕНТІВ ІГРОВОГО ЗАСТОСУНКУ

3.1 Стек технологій

Для створення ігрового застосунку необхідно використовувати потужний стек технологій, що включає програми для моделювання, текстурінгу та реалістичного візуалізації. Для цього розділу було обрано такі програми, як Blender, Substance 3D Painter та UE5.0, через їхні високі можливості та популярність у галузі розробки ігор.

Blender є повнофункціональним програмним пакетом для 3D-моделювання, анімації, рендерингу та композитингу. Він надає широкий набір інструментів, які дозволяють моделювати складні об'єкти рівня та ігрових персонажів. Blender має інтуїтивний інтерфейс, швидку обробку геометрії та розширені можливості для текстурінгу та створення матеріалів [5]. Вибір Blender обґрунтовується його безкоштовністю та активною спільнотою розробників, що забезпечує швидку підтримку та оновлення програми.

Таблиця 3.1 – Порівняльна таблиця Blender та його аналогів

Функції	**Blender**	**Autodesk Maya**	**Cinema 4D**	**3ds Max**	**SketchUp**
Вартість	Безкоштовний	Платний	Платний	Платний	Безкоштовний/ Платний
Користувацький інтерфейс	Дружній та кастомізований	Складний та потужний	Інтуїтивний	Розширений, але вимагає навчання	Простий та інтуїтивний
Функціонал	Широкий спектр інструментів та функцій, включаючи моделювання, анімацію, рендеринг, VFX та багато іншого	Повний спектр інструментів для професійних анімаційних та VFX проєктів	Високоякісний рендеринг, фізична симуляція, динаміка та VFX	Широкі можливості для моделювання, анімації, симуляції та рендерингу	Простий інструмент для створення 3D моделей та візуалізації
Підтримка платформ	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS	Windows	Windows, macOS
Рендеринг	Внутрішній рендерер, Cycles, Eevee та підтримка інших рендерерів через плагіни	Внутрішній рендерер, Arnold, Mental Ray, V-Ray та багато інших	Внутрішній рендерер, Arnold, Physical, ProRender та інші	Внутрішній рендерер, Arnold, V-Ray, Corona та багато інших	Внутрішній рендерер, Podium, Thea, VRay та інші

Кінець таблиці 3.1

Анімація	Повна підтримка анімації з ключовими кадрами, ієрархічними схемами, системами частинок та фізикою	Широкий спектр інструментів для анімації об'єктів, персонажів та VFX	Професійний інструмент для створення складних анімаційних сцен та VFX	Широкі можливості для анімації об'єктів, персонажів та симуляцій	Обмежена функціональність анімації
Моделювання	Повна підтримка для моделювання об'єктів, скульптури, NURBS, ретопології та багато іншого	Розширені інструменти для моделювання, включаючи NURBS, скульптуру, ретопологію та багато іншого	Широкі можливості для моделювання об'єктів, скульптури, NURBS та іншого	Розширені інструменти для моделювання об'єктів, скульптури, NURBS та іншого	Простий інструмент для базового моделювання об'єктів
Підтримка форматів файлів	Широкий спектр підтримуваних форматів, включаючи OBJ, FBX, STL, PLY та багато інших	Широкий спектр підтримуваних форматів, включаючи OBJ, FBX, STL, Alembic та багато інших	Широкий спектр підтримуваних форматів, включаючи OBJ, FBX, Alembic та багато інших	Широкий спектр підтримуваних форматів, включаючи OBJ, FBX, Alembic та багато інших	Широкий спектр підтримуваних форматів, включаючи SKP, OBJ, STL, DAE та багато інших

Substance 3D Painter є потужним інструментом для текстурінгу 3D-моделей. Ця програма дозволяє легко створювати реалістичні матеріали, застосовувати текстури, малювати деталізацію та створювати ефекти зношування. Substance Painter має широку бібліотеку матеріалів та текстур, що спрощує процес створення візуально привабливих об'єктів [6]. Вибір Substance Painter обґрунтовується його інтуїтивним інтерфейсом та високою якістю генерації текстур.

Таблиця 3.2 – Порівняльна таблиця Substance 3D Painter та його аналогів

Функція/Характеристика	**Substance 3D Painter**	**Quixel Mixer**	**3D-Coat**	**Mari**
Вартість	Платний	Безкоштовний	Платний	Платний
Інтерфейс	Сучасний та інтуїтивний	Зручний	Складний	Складний
Підтримка форматів	Широкий спектр форматів, включаючи .sbsar, .obj, .fbx та інші	Обмежений	Широкий спектр форматів, включаючи .obj, .fbx, .psd та інші	Обмежений

Кінець таблиці 3.2

Технології шарування	Продуктивне шарування та змішування матеріалів з високою роздільною здатністю	Змішування шарів з підтримкою штучного інтелекту	Шарування матеріалів з можливістю редагування низького полігону	Шарування матеріалів з інтуїтивним інтерфейсом
Інструменти пензлика	Розширений набір інструментів пензлика з великою кількістю параметрів	Різноманітні інструменти пензлика для штучного інтелекту	Розширений набір інструментів пензлика з функціями розтягування, різання та інших	Розширений набір інструментів пензлика з великою кількістю параметрів
Реалістичність	Реалістичне моделювання матеріалів з високою різкістю текстур	Висока якість рендерингу за допомогою штучного інтелекту	Деталізація текстур та матеріалів з низькими полігонами	Висока реалістичність текстур та матеріалів
Робота з UV-розгорткою	Удосконалені інструменти розгортки UV-карти з автоматичною розгорткою та редагуванням	Простота розгортки UV-карти та можливість автоматичного редагування	Розгортка UV-карти з функціями автоматичної розгортки та редагуванням	Розгортка UV-карти зі вбудованими інструментами редагування
Імпорт/експорт	Підтримка імпорту та експорту в різні формати для інтеграції з іншими програмами	Підтримка імпорту та експорту в обмеженому форматі	Широкий спектр імпорту та експорту форматів для інтеграції з іншими програмами	Широкий спектр імпорту та експорту форматів для інтеграції з іншими програмами

UE5.0 (Unreal Engine 5.0) є потужним ігровим двигуном, який надає великі можливості для розробки ігрових застосунків. UE5.0 має вбудовану підтримку реалістичного освітлення, динамічних ефектів та фотореалістичного рендерингу.

Цей двигун забезпечує велику швидкість розробки, масштабність та можливість використання готових рішень для фізики, штучного інтелекту та інших складових гри [7]. Вибір UE5.0 обґрунтовується його потужними можливостями та активною спільнотою розробників.

Функція/Характеристика	Unreal Engine 5 (UE5)	Unity	CryEngine	Frostbite
Рендеринг	Nanite (Virtualized Micro Polygon Geometry) and Lumen (Real-time Global Illumination)	HDRP (High Definition Render Pipeline) and URP (Universal Render Pipeline)	CryEngine Renderer	Frostbite Renderer
Освітлення	Real-time Global Illumination (Lumen)	Real-time Global Illumination (Unity 2020+ with HDRP)	Real-time Global Illumination	Real-time Global Illumination
Фізика	PhysX	Unity Physics	CryPhysics	Frostbite Physics
Мови програмування	C++, Blueprints	C#, UnityScript	C++, Lua	C++
Підтримка різних платформ	PC, Console, Mobile	PC, Console, Mobile	PC, Console	PC, Console
Документація	Обширна та детальна	Обширна та детальна	Обширна та детальна	Обширна та детальна
Спільнота	Велика та активна	Велика та активна	Менша, але активна	Велика та активна

Комбінація Blender, Substance Painter та UE5.0 дозволяє створити повноцінний ігровий застосунок зі складними об'єктами рівня, деталізованими ігровими персонажами та реалістичними матеріалами. Цей стек технологій дозволить розробникам досягти високої якості візуальної привабливості та геймплею у своєму проєкті.

3.2 Моделювання об’єктів рівня

Об’єкти рівня будуть створенні за допомогою полігонального моделювання. Полігональне моделювання є одним з найпоширеніших методів створення 3D моделей у сучасній комп'ютерній графіці. Воно базується на використанні полігонів - геометричних фігур з визначеними вершинами, ребрами та поверхнями.

Полігональні моделі складаються з масиву таких полігонів, як трикутники, чотирикутники або багатокутники, які відображають форму та структуру об'єкта.

Створення модульних 3D моделей для рівня гри - це процес створення 3D об'єктів, які можуть бути комбіновані та використовуватись для побудови рівнів у відеоіграх. Модульність означає, що ці моделі складаються з окремих частин або модулів, які можуть бути з'єднані разом для створення більш складних структур [8].

Одна з головних переваг модульних 3D моделей полягає в тому, що вони дозволяють геймдизайнерам швидко створювати та змінювати рівні, не

витрачаючи багато часу на створення кожного елемента окремо. Замість цього, вони можуть використовувати набір попередньо створених модулів і збирати їх у різних комбінаціях, щоб отримати різноманітність у грі.

Хоча створення модульних 3D моделей для рівня гри має свої переваги, також існують деякі недоліки, які варто врахувати:

1. *Обмежена унікальність*: Використання модульних моделей може призводити до того, що рівні гри стають схожими або навіть однотипними. Це може знизити ступінь оригінальності та унікальності гри. Якщо всі рівні будуються з одних і тих самих модулів, гра може стати передбачуваною і втратити свою привабливість для гравців;

2. *Обмежені можливості дизайну*: Використання модульних моделей може обмежувати можливості геймдизайнера у створенні унікальних та нетрадиційних рівнів. Деякі складні або незвичні форми та структури можуть бути важко втілити за допомогою заздалегідь підготовлених модулів.

Використовуючи функціонал Blender'у було створено 11 модульних об'єктів, які можна комбінувати (рис. 3.1).

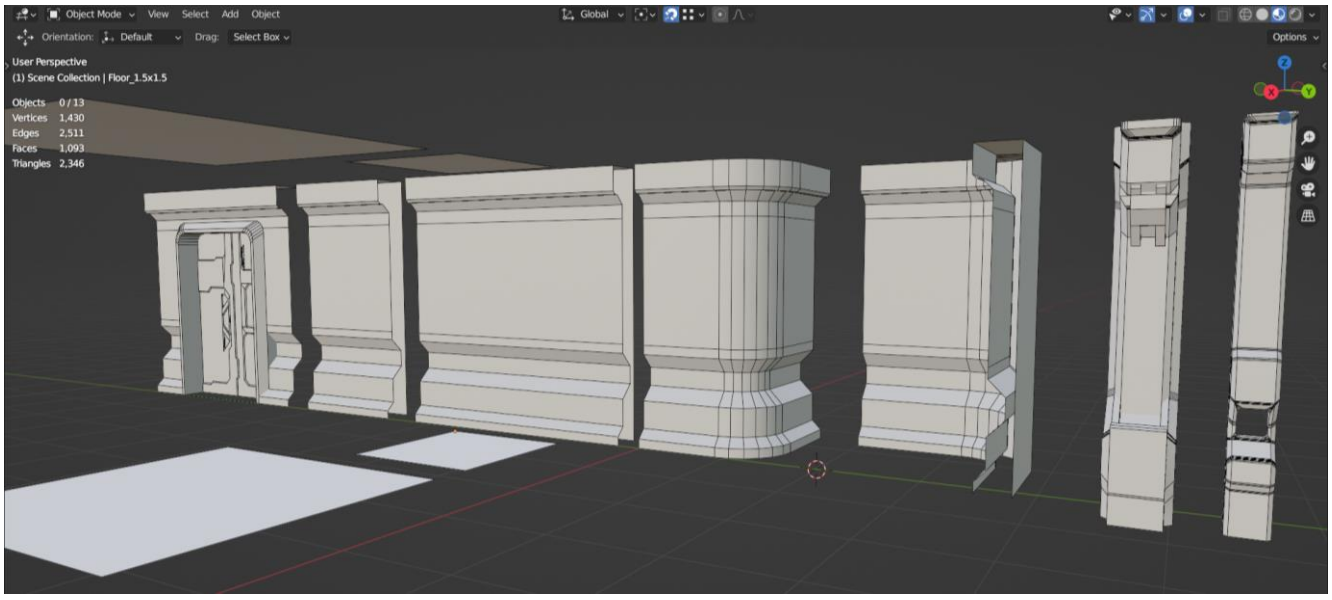

Рисунок 3.1 – 3D об'єкти рівня

В склад об'єктів рівнів входять 5 типів стін, 2 колони та 2 типи підлоги зі стелею.

3.3 Моделювання ігрового персонажу

Моделювання ігрового персонажу в Blender з використанням скульптингу є цікавим та творчим процесом. Ось кілька етапів та особливостей, які варто врахувати при моделюванні ігрового персонажу:

1. *Планування*: Перш за все, перед початком моделювання потрібно визначити концепцію вашого персонажа. Ви можете створити скетч або зображення, яке ви будете використовувати як основу для моделювання. Це допоможе вам мати чітке уявлення про те, яким повинен бути ваш персонаж.

2. *Базова геометрія*: Почніть зі створення базової форми вашого персонажа за допомогою полігонів або скульптингу базової геометрії. Ви можете використовувати готові базові моделі людського тіла або створити їх самостійно.

3. *Скульптинг деталей*: Після створення базової форми ви можете перейти до додавання деталей за допомогою інструментів скульптингу в Blender.

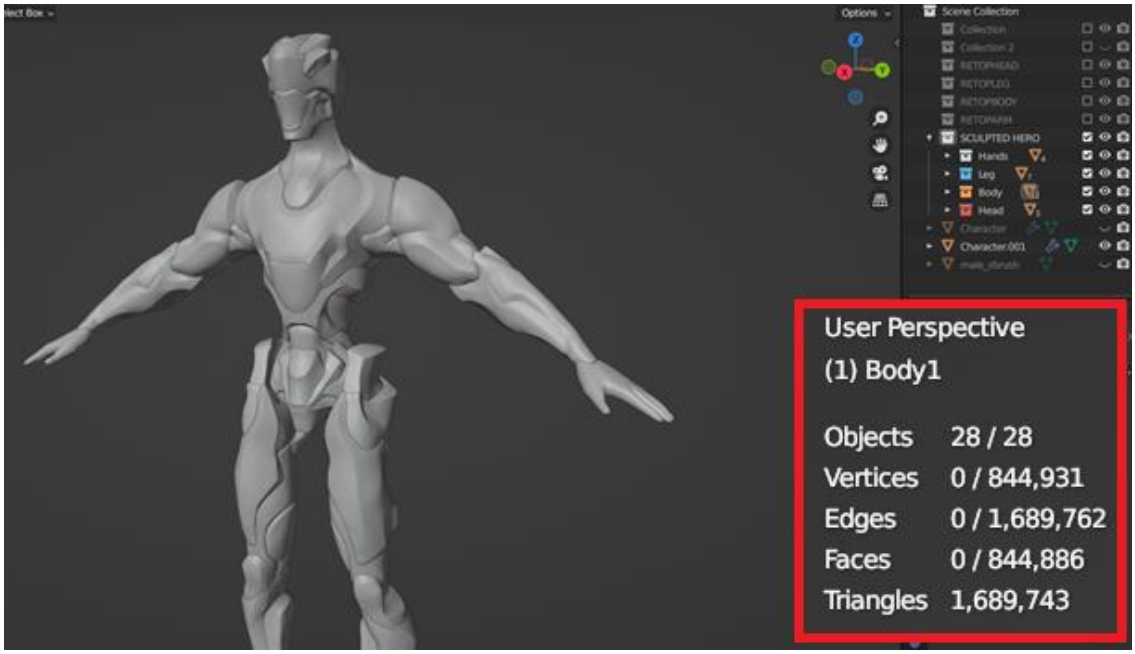

Рисунок 3.2 – Створена модель головного ігрового персонажу

Ігровий персонаж після закінчення скульптингу має дуже щільну сітку полігонів, це можна побачити по кількості полігонів, а саме *Triangles* на рис 3.2 у червоній рамці. Така кількість автоматично робить персонажа непридатним до

використовування його в ігрі, тому наступним етапом завершення створення ігрового персонажу є ретопологія.

Ретопологія - це процес перебудови геометрії моделі, зазвичай після створення її базової форми або після скульптингу, з метою отримання оптимізованої топології для покращеної анімації, деформації та ефективного використання ресурсів у реальному часі [9].

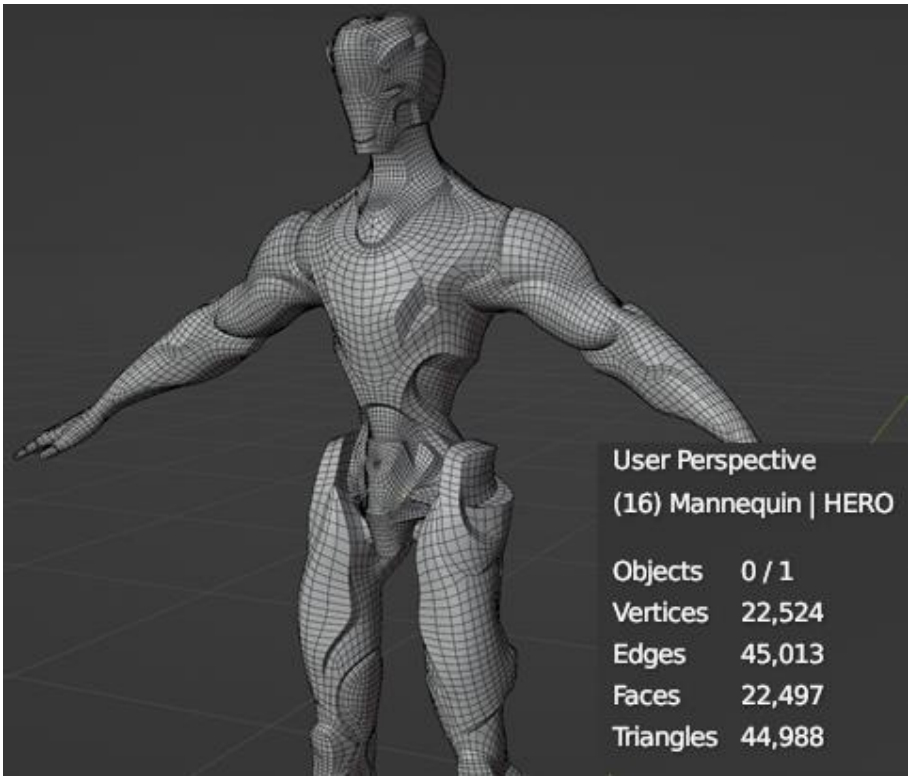

Рисунок 3.3 – Щільність сітки та кількість полігонів після ретопології

Після ретопології, персонаж має набагато оптимізованішу сітку і робить його придатним до подальших маніпуляцій.

3.4 Створення розгортки та текстур

Розгортка 3D моделей, відома також як UV-розгортка або UV-розмітка, є процесом присвоєння текстурним поверхням 3D об'єкта 2D координат, які використовуються для нанесення текстур, кольорів та інших деталей на модель.

При відсутності цієї розгортки, створення матеріалів чи текстури неможливе.

При створені розгортки, Textel density (текстурна щільність) відноситься до кількості текстурних пікселів (текселів), які припадають на одиницю поверхні моделі. Це показник, який визначає, наскільки деталізовано текстури будуть відображатися на поверхні об'єкта [10].

Вибір оптимальної текстурної щільності залежить від кількох факторів, включаючи розмір текстур, розмір та деталізацію моделі, обсяг пам'яті, доступний для використання і технічні обмеження використовуваного двигуна або платформи.

Збільшення текстурної щільності означає, що на кожен піксель текстури припадатиме більше полігонів або вершин, що утворюють поверхню моделі. Це забезпечує більшу деталізацію та роздільну здатність текстур на поверхні. З іншого боку, зменшення текстурної щільності призводить до меншої кількості полігонів або вершин, які використовуються для відображення текстур.

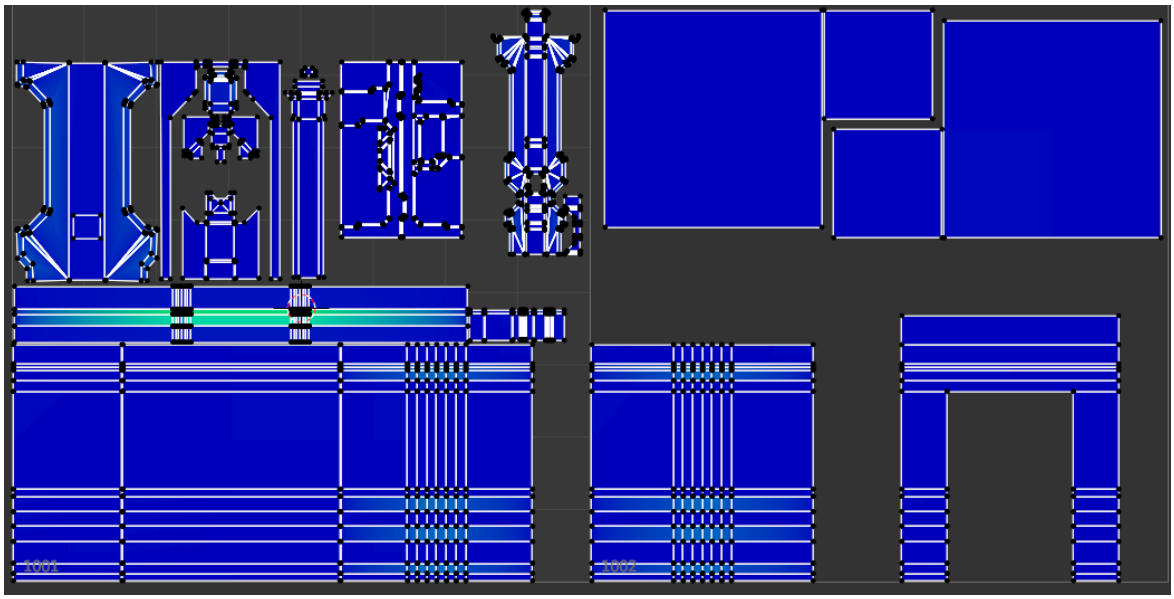

Рисунок 3.4 – Приклад розгортки об’єктів рівня на UDIM карті

Використання UV-карт може мати один недолік - вони складаються з однієї текстури для всієї сіті. Здебільшого цього достатньо, але недоліком є те, що текстура має одну роздільну здатність для всієї сіті.

Це спричиняє проблеми, якщо у вас дуже велика сіть з геометрією різної важливості. Тому для всіх створених моделей використовувались UDIM карти.

UDIM (U Dimension) є стандартом, що використовується в комп'ютерній графіці та моделюванні для розміщення та ідентифікації текстур на поверхнях 3D-моделей. UDIM дозволяє розширити традиційне обмеження однієї текстури на поверхню моделі, дозволяючи використовувати набір текстур, розташованих на різних координатних просторах.

Замість одного текстурного квадрату, який використовується в класичній текстуруванні, UDIM дозволяє використовувати багато текстурних квадратів, розташованих поруч один з одним на площині UDIM. Кожен квадрат представляє окрему текстуру, яку можна малювати, редагувати та накладати на відповідну частину моделі.

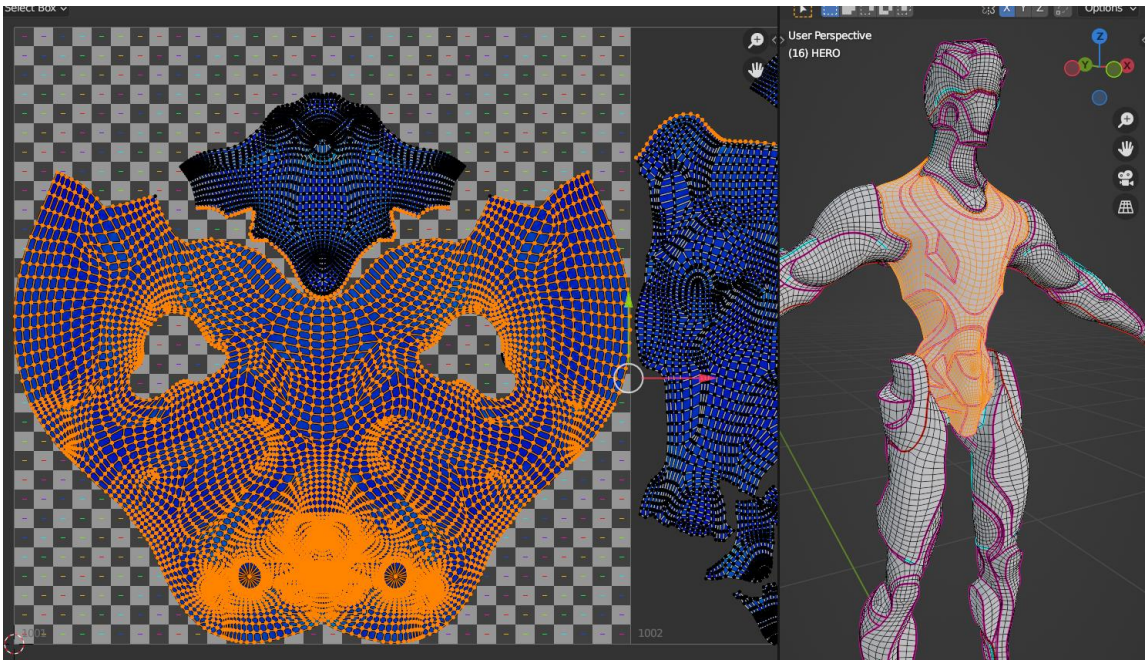

Рисунок 3.5 – Приклад розгортки тіла ігрового персонажу

Перевагою UDIM полягає у збільшенні простору для створення текстур та використанні вищої роздільної здатності для деталей. Це особливо корисно для моделей з великою кількістю деталей, таких як персонажі, де більш висока деталізація текстур є важливою.

Після того як розгортка готова, можна перейти до створення текстурування створених 3D моделей. За допомогою програми Substance Painter 3D та його

функціоналу створюємо текстури. Текстурування є творчим процесом для інді-розробника, тому в Substance Painter ви маєте безліч можливостей для виявлення свого художнього потенціалу.

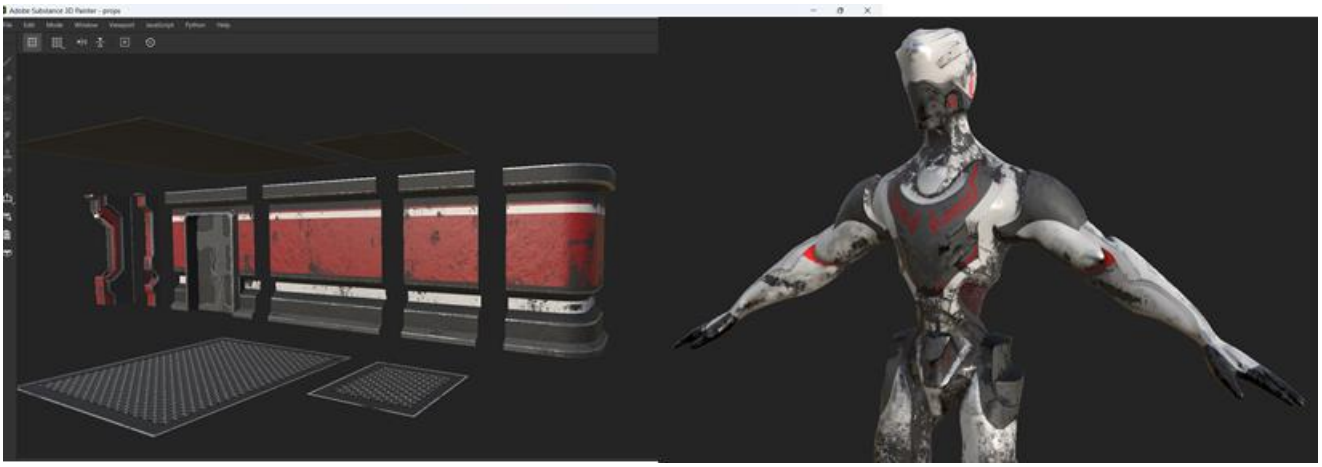

Рисунок 3.6 – Створені текстури для 3D моделей

Ви можете створювати унікальні матеріали, відтворювати реалістичні текстури зі шрамами, брудом, ржавчиною тощо, а також експериментувати з різними кольорами, освітленням та ефектами, щоб створити вражаючі візуальні елементи для своїх ігрових проєктів.

Substance Painter надає потужні інструменти, такі як Smart Materials, Particle Brushes та Procedural Masks, що дозволяють вам створювати складні та деталізовані текстури, які підкреслюють унікальність вашого власного стилю та візії в грального світі.

3.5 Створення рівня

Після створення 3D моделей та текстур до них, можна імпортувати все в рушій. Для цього потрібно створити проєкт з шаблоном пустого рівня, обравши відповідний шаблон з меню швидкого доступу створення проєкту.

Бажано попередньо вказати тип програмування на Blueprints та пресет якості на Scalable (рис. 3.7).

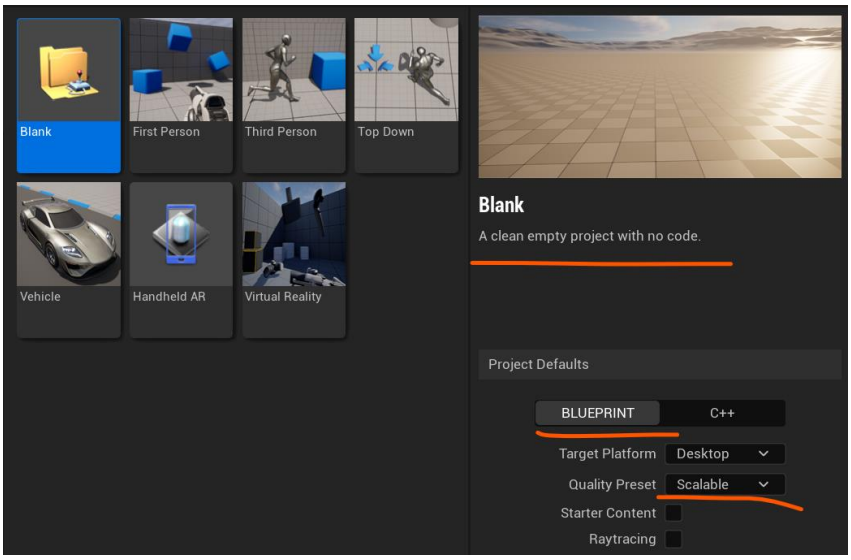

Рисунок 3.7 – Створення проєкту за пустим шаблоном

Після цього можна імпортувати всі створенні об’єкти в рушій. Для створення рівня, нам знадобиться переміщати створенні об’єкти рівня на сцену, виконувати це по одному об’єкту займе дуже багато часу, тому доречно створити пресети. Для цього створюємо «blueprint» та вказуємо тип «actor». Далі додаємо потрібну кількість компонентів «static mesh» вказавши потрібний 3D об’єкт (рис. 3.8).

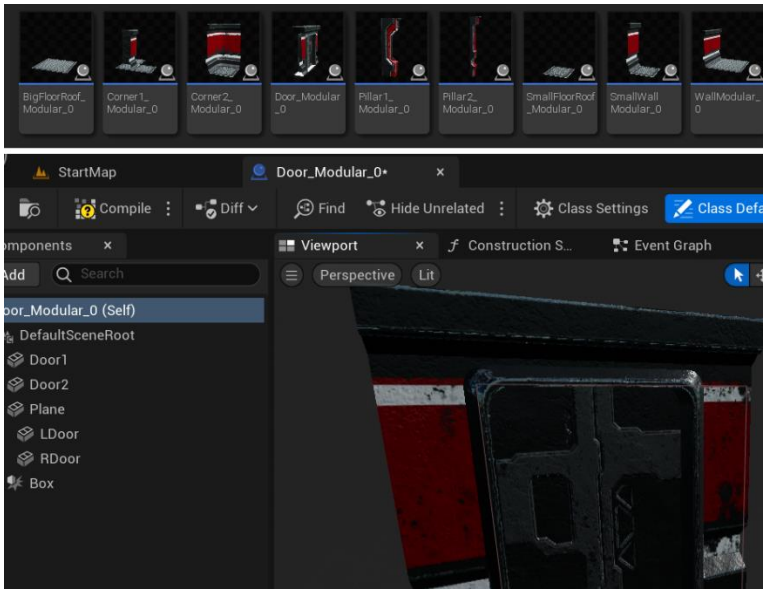

Рисунок 3.8 – Створення пресетів для об’єктів рівня

Тепер швидкість ручного створення рівня пришвидшиться в декілька разів.

Висновки до розділу 3

У даному розділі описано стек технологій, який був використаний для розробки ігрового застосунку. Обрано програми Blender, Substance 3D Painter та UE5.0, оскільки вони є популярними та мають широкий спектр можливостей в галузі розробки ігор.

Blender є повнофункціональним програмним пакетом для 3D-моделювання, анімації, рендерингу та композитингу. Substance 3D Painter є потужним інструментом для текстурінгу 3D-моделей. UE5.0 (Unreal Engine 5.0) є потужним ігровим двигуном, який надає широкі можливості для розробки ігрових застосунків. Комбінація Blender, Substance Painter та UE5.0 дозволяє створити повноцінний ігровий застосунок зі складними об'єктами рівня, деталізованими ігровими персонажами та реалістичними матеріалами. Цей стек технологій дозволяє розробникам досягти високої якості візуальної привабливості та геймплею у своєму проєкті.

Також було описано процес створення 3D об'єктів для рівнів гри за допомогою полігонального моделювання та модульного підходу та процес моделювання ігрового персонажу. Виконати ретопологію моделі для отримання оптимізованої топології, яка підходить для анімації та використання в реальному часі. Було розглянуто процес розгортки 3D моделей.

Завершальним етапом є імпортування створених 3D моделей та текстур до ігрового рушія.

4 СТВОРЕННЯ ІГРОВОГО ЗАСТОСУНКУ

4.1 Меню гри

Почнімо зі створення меню для ігрового застосунку. Для цього створимо папку Maps, та створимо пустий рівень за допомогою «Ctrl+N» та обравши відповідний шаблон (рис. 4.1).

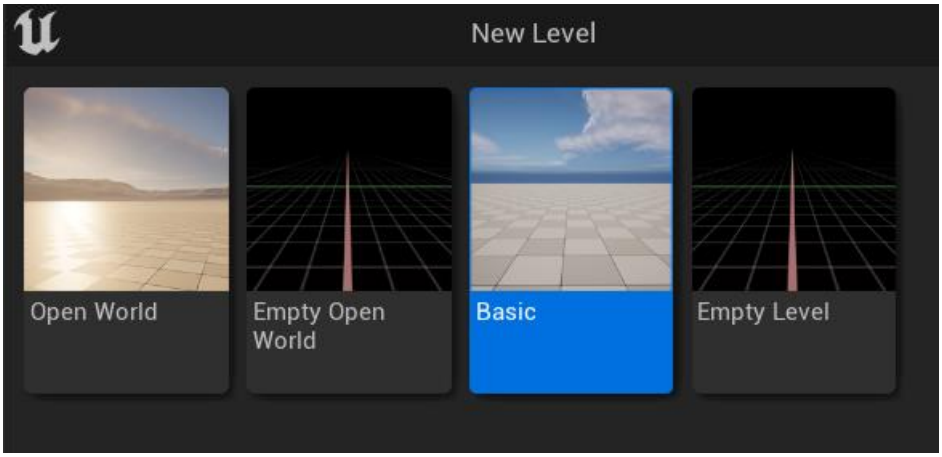

Рисунок 4.1 – Створення пустого рівня

Далі створюємо папку де будуть розміщенні всі віджети (UI), далі через контекстне меню «Content Drawer» створюємо перший Widget Blueprint (рис. 4.2). Перейменовуємо його та відкриваємо.

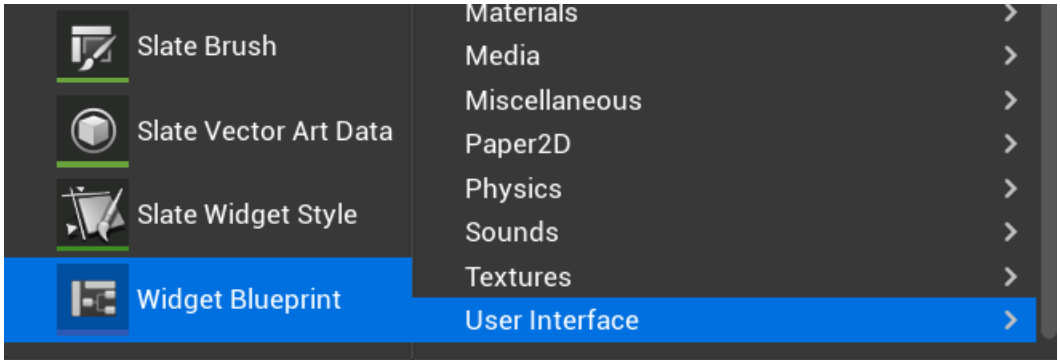

Рисунок 4.2 – Створення першого віджету

Після відкриття в панелі «Palette» перетягуємо «Canvas Panel», який працює як контейнер, що може містити інші елементи, такі як текстові блоки, зображення, кнопки тощо. Він також надає можливість гнучко налаштовувати за допомогою властивостей і налаштувань.

Після розміщення «Canvas Panel», можна приступити до розміщення інших контейнерів, кнопок, текстових блоків тощо.

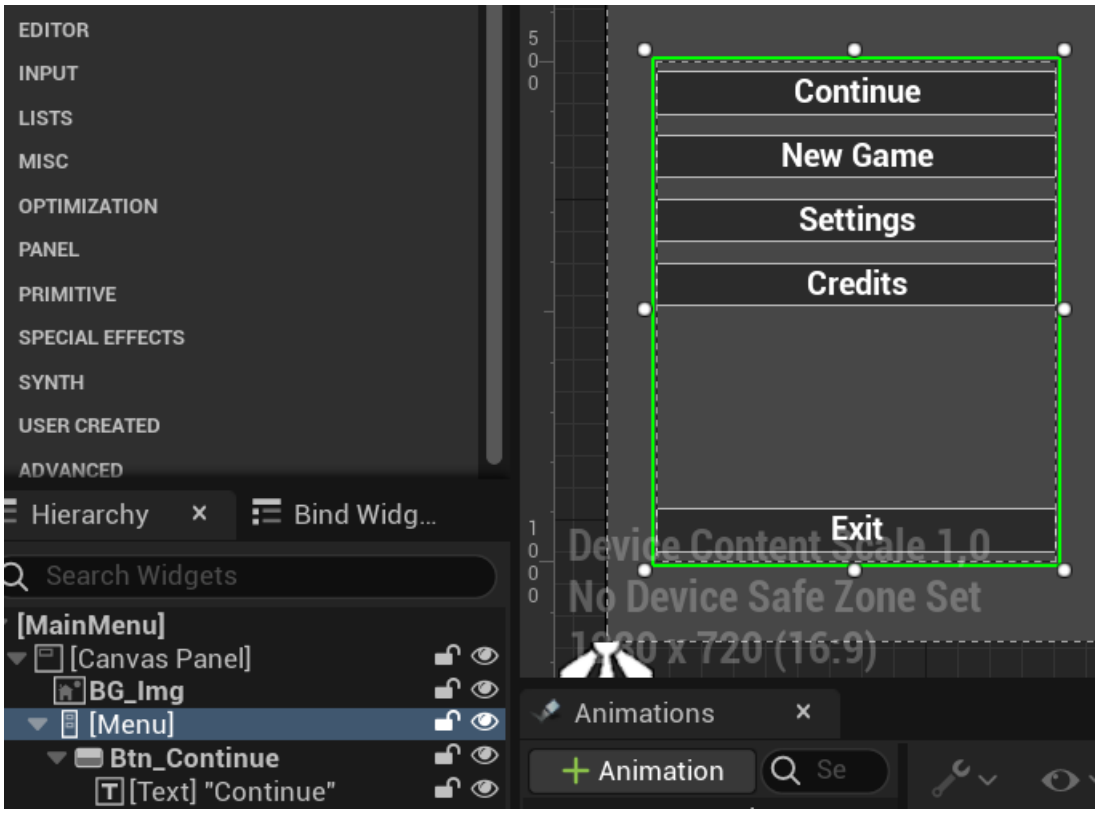

Рисунок 4.3 – Створений блок меню, що має кнопки на текст

Для кожного елемента меню, можна створити окремий новий віджет, або використовувати цей же. Тому для деяких можемо продублювати попередній блок меню, змінивши потрібні параметри.

Щоб при натисканні певної кнопки з’являлась потрібна частина меню, можна створити прості анімації, основані на зміщенні по осях X чи Y. Для цього натискаємо на відповідно кнопку в панелі «Animation». Обираємо потрібний елемент меню після натискання кнопки «Track» та додаємо потрібні параметри що будуть змінюватись.

Далі встановлюймо ключові кадри разом з потрібним значенням параметрів та змінюємо їх на потрібному відрізку часу (рис. 4.4).

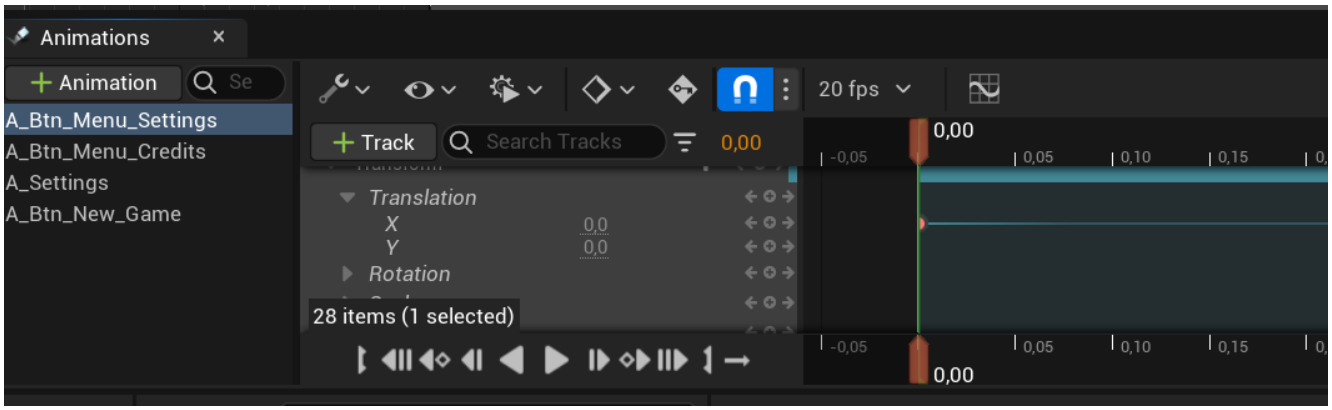

Рисунок 4.4 – Створені анімації для кнопок

Наступним кроком буде реалізація функціоналу кнопок. Для цього, для кожної кнопки що є в меню створюємо «onClicked» подію.

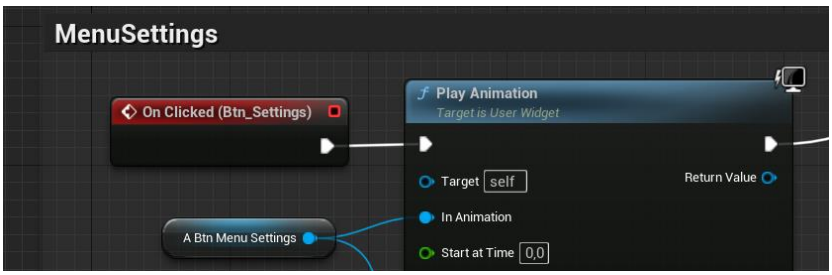

Рисунок 4.5 – Виклик анімації для кнопки після кліку

Для реалізації меню налаштувань створюємо новий віджет, та розміщуємо потрібні елементи. Далі переходимо в робочій простір «Graph», та створюємо потрібні функції, змінні (рис. 4.5).

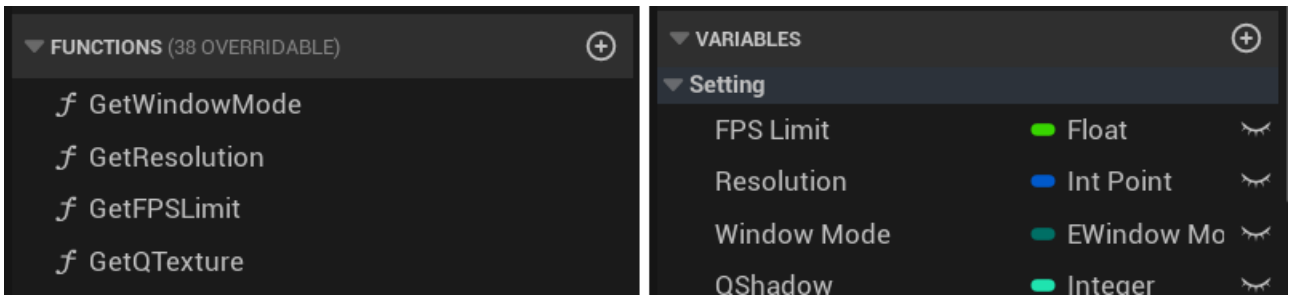

Рисунок 4.6 – Виклик анімації для кнопки після кліку

Всі змінні, що стосуються якості повинні бути типу «Integer».

Перейдемо до реалізації функцій, вони всі будуть відповідати за відображення назв для відповідних налаштувань, тому в кожній відповідній функції встановлюємо значення відповідних змінних. Приклад реалізації можна побачити на рис. 4.7, зліва зверху встановлення текстового значення для режиму вікна, справа зверху встановлення текстового значення роздільної здатності та знизу приклад який є однаковим для всіх графічних характеристик.

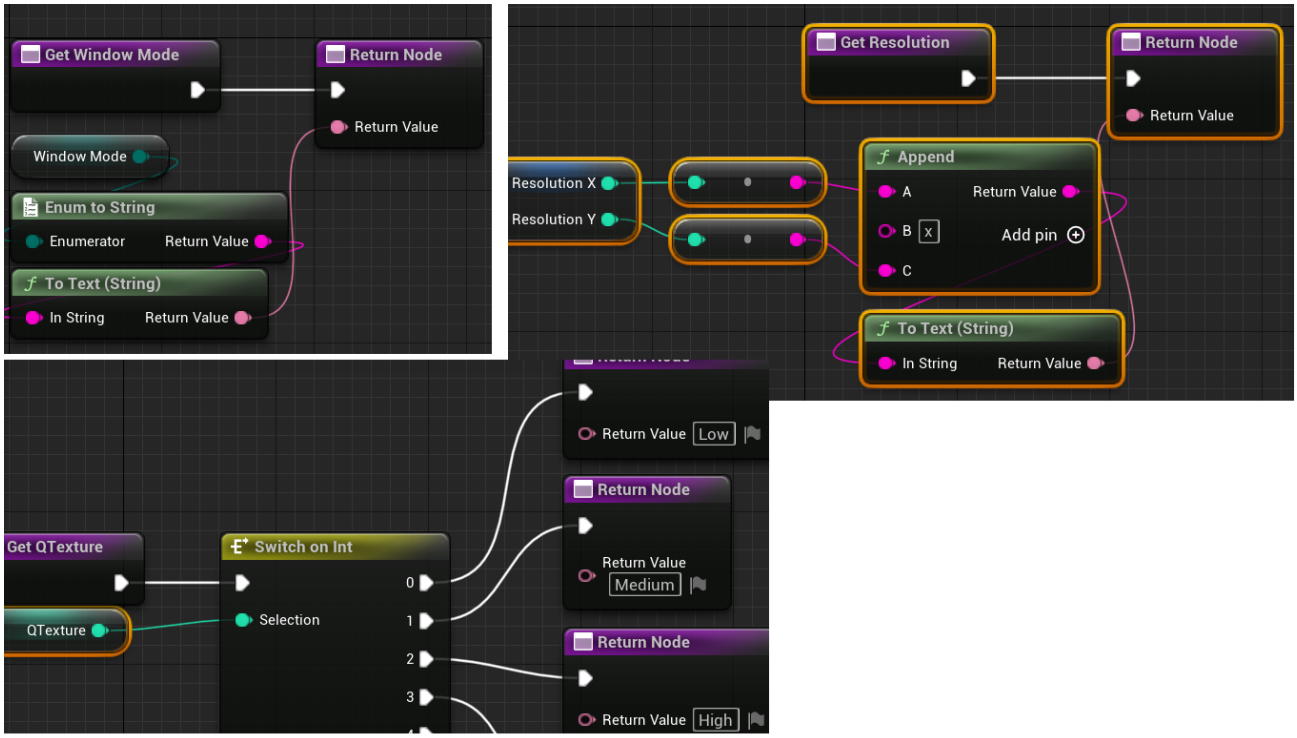

Рисунок 4.7 – Приклад реалізацій функцій

Після завершення реалізацій встановлюємо ці функції в текстові блоки налаштувань рис. 4.8.

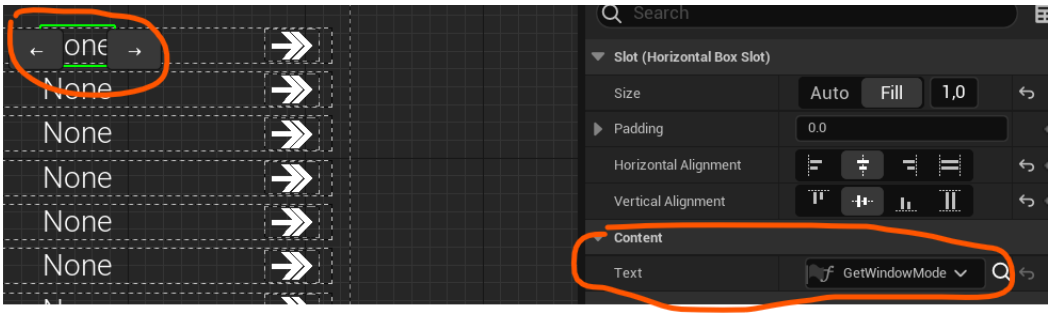

Рисунок 4.8 – Встановлення функцій

Це дає змогу програмно змінювати значення в залежності від параметрів.

Далі створюємо «onClicked» події для всіх кнопок, що змінюють значення налаштувань, та додаємо реалізацію.

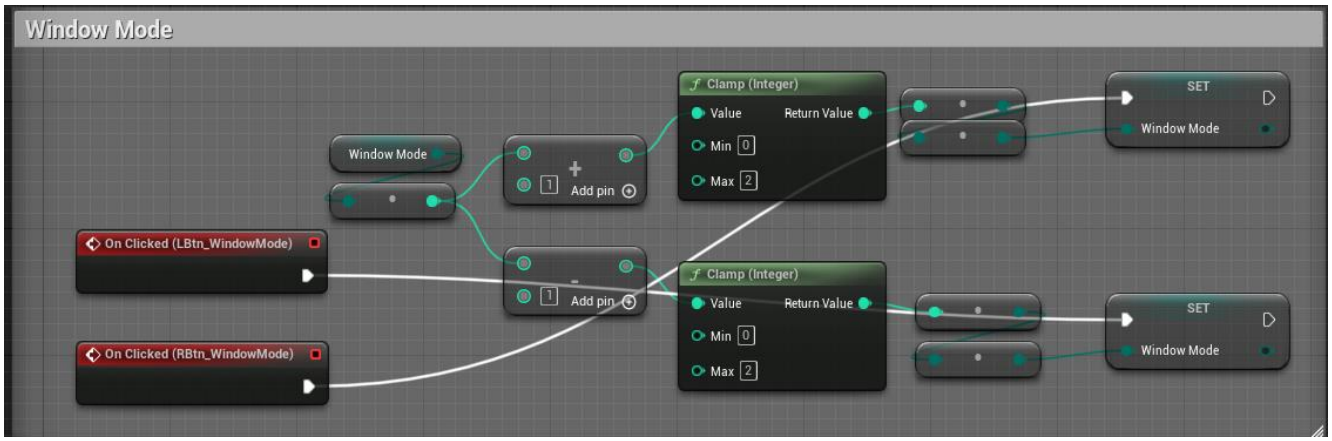

Рисунок 4.9 – Реалізація зміни значень

На рис. 4.9 показана реалізація, зміни режиму вікна. За допомогою створених подій встановлюємо значення змінної «Window Mode», яка має тип Enum (EWindowMode) в залежності від нажатої кнопки (збільшуємо або зменшуємо) в обмеженні від 0 до 2. Всі інші реалізації для зміни значень, відрізняються лише обмеженням.

Для того, щоб застосувати обрані налаштування, викликаємо функцію «Get Game User Setting» та знаходимо функції які відповідають за встановлення значень для конкретних налаштувань, що в нас присутні (рис. 4.10).

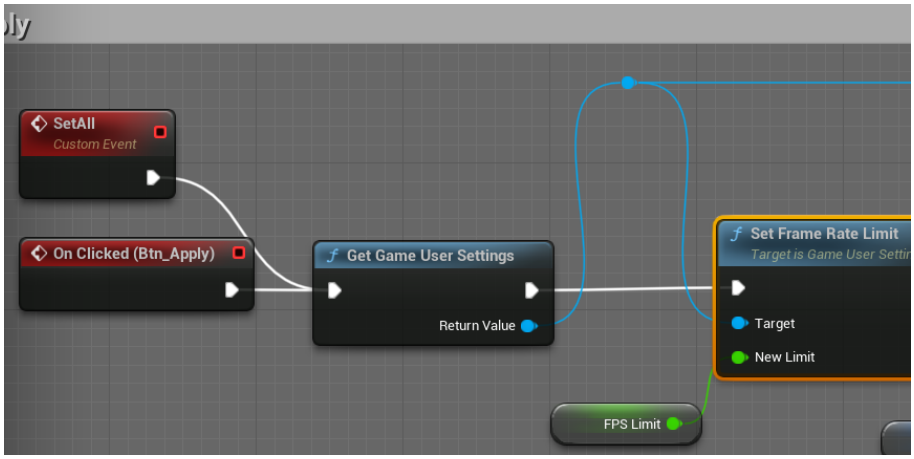

Рисунок 4.10 – Реалізація зміни значень

Також встановлюємо значення з наших змінних до цих функцій

Для перевірки роботи меню налаштувань, запускаємо на виконання в окремому вікні та змінюємо параметри (рис. 4.11).

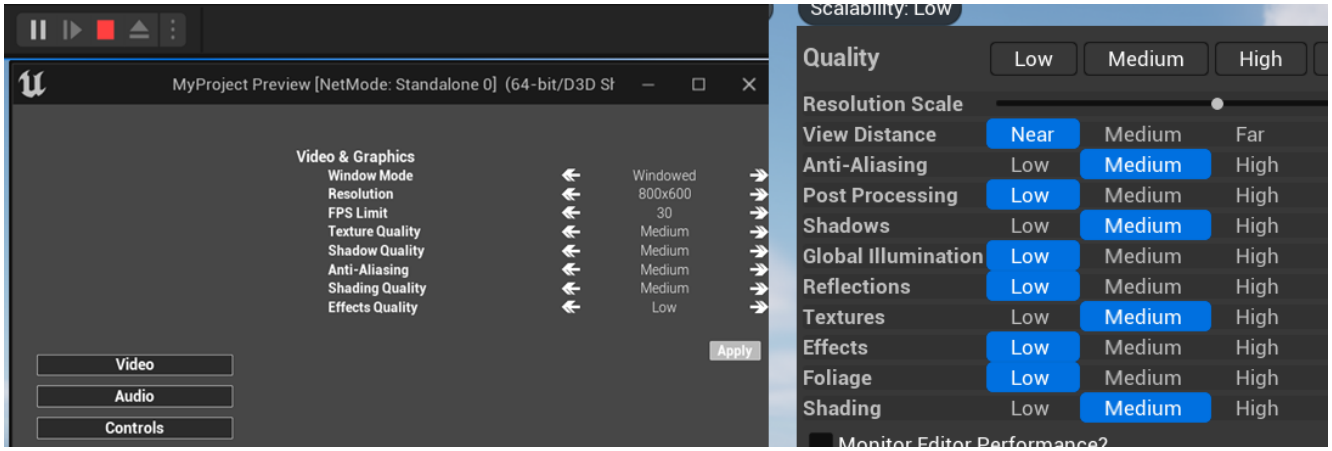

Рисунок 4.11 – Перевірка меню налаштувань (зміна розміру вікна зліва, та встановлення графічних налаштувань справа)

В результаті всі обрані графічні налаштування змінились. Було встановлено розмір вікна та інші графічні налаштування.

4.2 Реалізація управління персонажем

Почнімо зі створення «InputAction». Їх кількість залежить від кількості подій які ми плануємо створити. В нашому випадку, це буде декілька базових для переміщення персонажу (вліво/вправо та вперед/назад), для керування камерою, для лутингу валюти з предметів рівня, для ввімкнення системи прокачка навичок, та для атаки персонажа.

Після створення «InputAction», треба створити «Input Action Context». Він використовується для визначення контексту, в якому може виконуватись певна дія (input action). Контекст включає в себе різні умови або стани, які повинні бути виконані, щоб дія була активована.

Наприклад, якщо ми маємо дію «Атакувати», ми можете встановити «Input Action Context» таким чином, щоб ця дія була доступною тільки в певних ситуаціях, наприклад, коли гравець перебуває в режимі бою або має зброю в руках.

Таким чином, «Input Action Context» допомагає керувати діями.

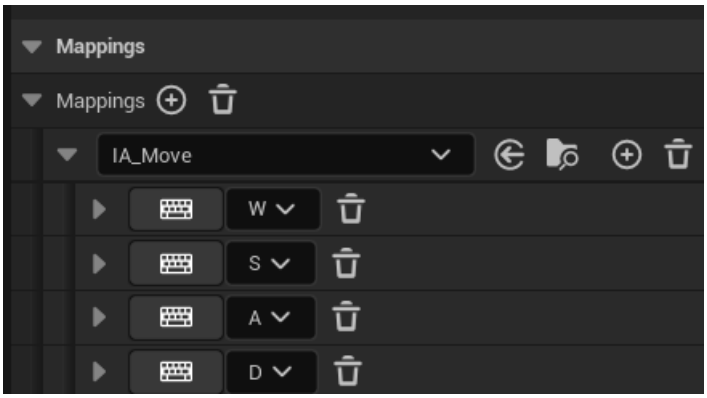

Рисунок 4.12 – Приклад мапінгу для дії «IA_Move»

Наступним кроком буде створення Blueprint'у для нашого персонажу, для цього створюємо новий Blueprint і вказуємо тип «Character Blueprint» в меню створення, та відкриваємо його. В панелі компонентів будуть знаходиться 4 елементи: «Capsule Component» та «Mesh», «Arrow Component» що є дочірніми елементами попереднього, а також додатковий компонент «CharacterMovements» що відповідає за різні налаштування для переміщення персонажу по рівню.

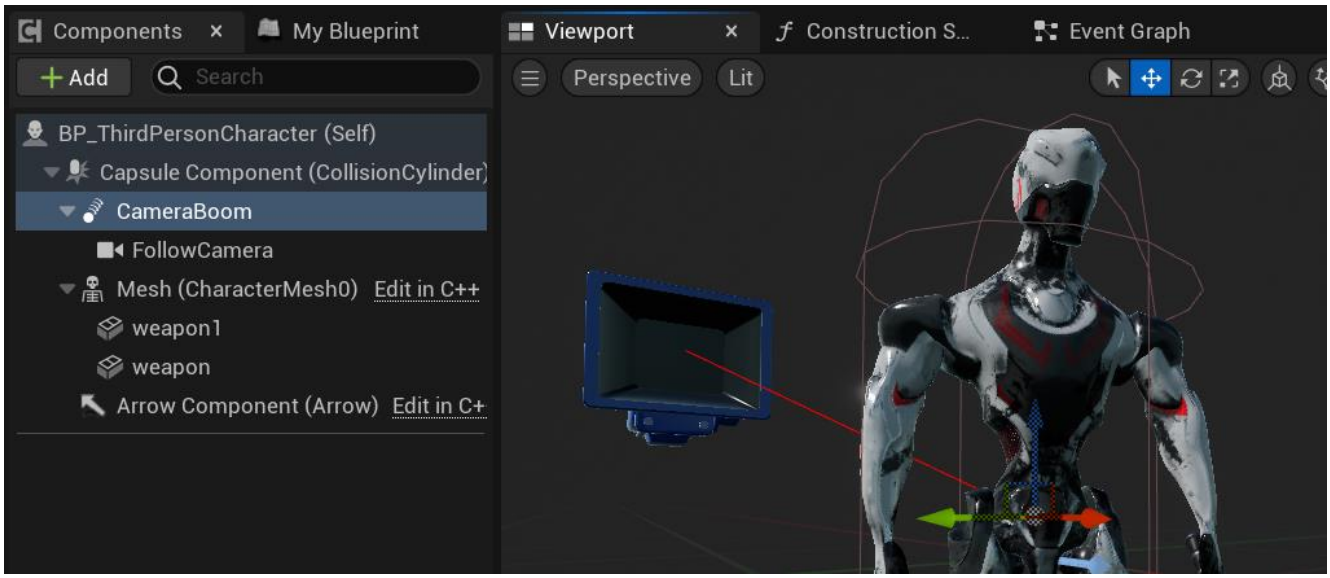

Рисунок 4.13 – Компоненти «Character Blueprint» з деякими налаштуваннями

«Capsule Component» використовується для простої колізії, «Mesh» дає можливість встановити саму модель нашого персонажу та інші його параметри. «Arrow Component» використовується лише з однією метою – вказання в яку сторону повинен «дивитись» елементи «Mesh» з встановленим персонажем.

Реалізація переміщення персонажа по рівню вліво/вправо та вперед/назад відображене має не складну логіку. Все що треба зробити, це викликати потрібні, створені, «InputAction» то додати декілька функції. «Get Control Rotation» – дозволяє отримати інформацію про поточний поворот або напрямок, в якому спрямовано управління або камеру в грі. Та дві функції для отримання вектору і з'єднати це все у відповідні піни двох методів «AddMovementInput».

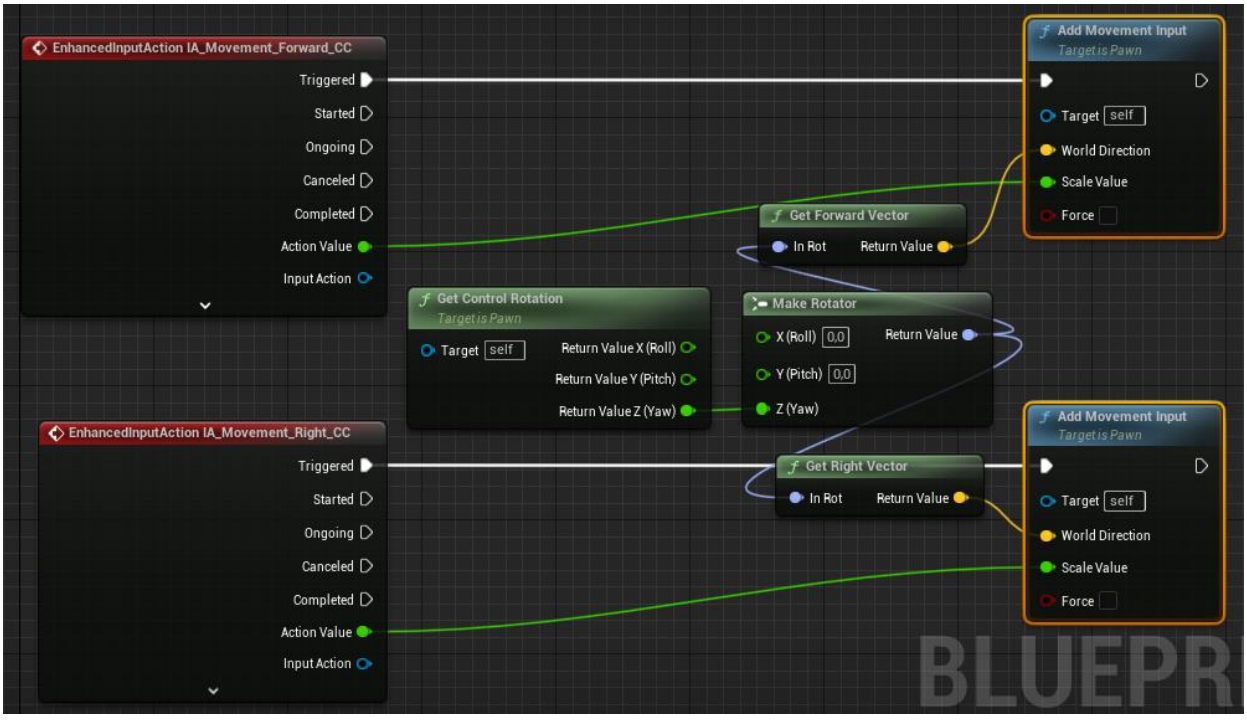

Рисунок 4.14 – Blueprint з логікою переміщення персонажа

Реалізуємо можливість бігу персонажу. Для цього викликаємо відповідну дію, та встановлюємо значення «Max Walk Speed» (рис .4.15)

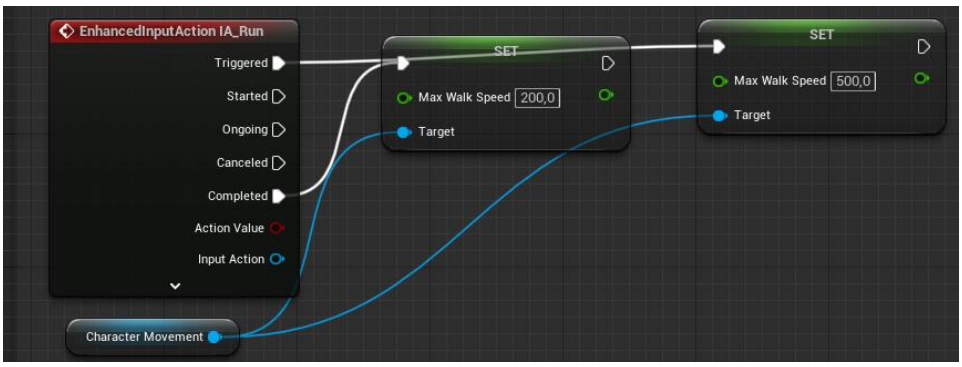

Рисунок 4.15 – Blueprint з логікою переміщення персонажа (Біг)

Значення встановлюють тоді коли ми натиснули кнопку і відпустили.

Щоб всі реалізовані «ActionInputs» працювали, треба додати наступні фрагменти:

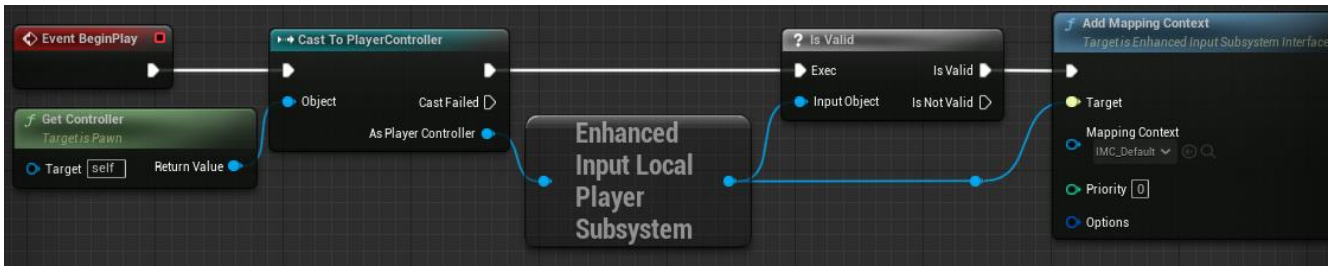

Рисунок 4.16 – Головна логіка для виконання всіх «ActionInputs»

1. «Get Controller»: Це метод, який викликається на об'єкті гравця або персонажа в Unreal Engine. Він повертає об'єкт контролера, який управляє даним гравцем або персонажем. Цей контролер відповідає за обробку введення (input) та керування діями гравця.

2. «Cast to PlayerController»: Цей метод виконує перетворення (casting) об'єкта контролера на тип "PlayerController". Він переконується, що отриманий об'єкт контролера є саме ігровим контролером гравця, щоб мати доступ до специфічних функцій і методів, які присутні в PlayerController.

3. «Enhanced Input Local Player Subsystem»: Це посилене введення локальної підсистеми гравця, що використовується в Unreal Engine. Вона надає розширені можливості для обробки введення в грі. Цей метод викликає підсистему введення для конкретного гравця, до якого належить контролер, і дозволяє виконати додаткові операції з введенням.

4. «Is Valid»: Цей метод перевіряє, чи є об'єкт підсистеми введення дійсним (valid). Це допомагає уникнути помилок, якщо підсистема не існує або є недійсною. Якщо підсистема валідна, метод продовжить виконання коду.

5. «Add Mapping Context»: Цей метод додає контексти мапування введення. Контексти мапування використовуються для забезпечення додаткової інформації про поточний контекст гри або ситуацію. Вони допомагають управляти активацією або деактивацією конкретних мапувань введення на основі цих контекстів.

Разом ці методи використовуються для отримання контролера гравця, перетворення його на PlayerController, взаємодії з підсистемою введення та додавання контексту мапування. Це дозволяє керувати та обробляти введення гравця в грі з додатковими можливостями та контролем.

4.3 Реалізація додаткових компонентів персонажу

Наступим кроком буде створення UI для персонажу. Для цього створюємо віджет да додаємо декілька текстових блоків для відображення рівню, кількість необхідного та здобутого досвіду та «ProgressBar» який буде відповідати за шкалу здоров'я головного героя.

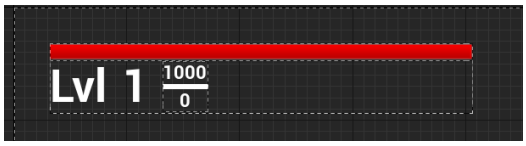

Рисунок 4.17 – UI головного героя

Для реалізації функціоналу цього UI, нам потрібно змінні які будуть зберігати максимальне та поточне здоров'я, рівень а також необхідний досвід і здобутий досвід. Отже, для кращого управління проєктом, створимо окремий Blueprint з типом «Actor Component», та створимо в ньому необхідні змінні та функції (рис. 4.18).

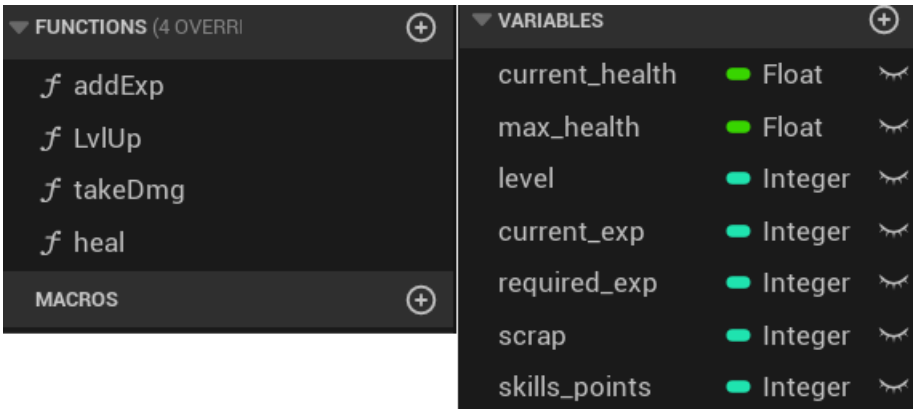

Рисунок 4.18 – Функції та змінні

Реалізація функцій буде простою, так як нам потрібно в них лише встановлювати значення створених змінних.

Наприклад функція нарахування досвіду має такий вигляд:

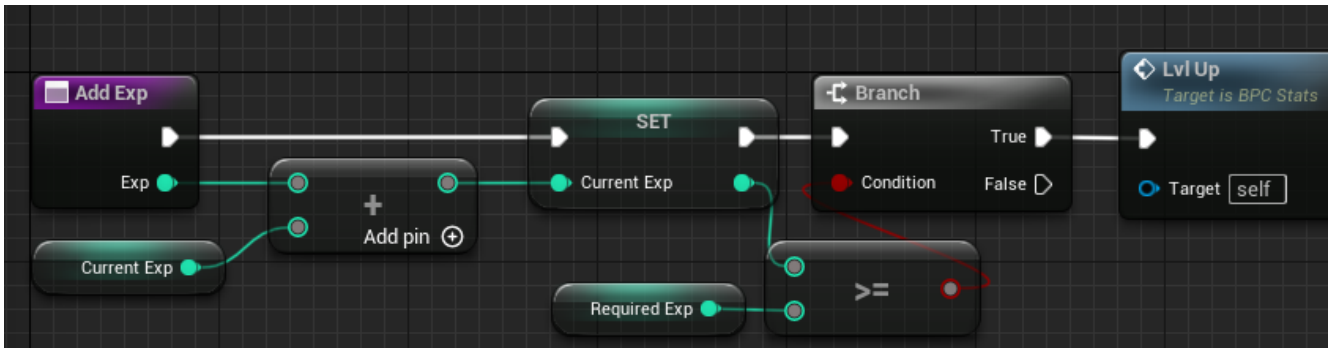

Рисунок 4.19 – Реалізація функції «AddExp»

В ній є вхідний параметр «Exp» – кількість досвіду та нескладна обробка. Ми додаємо значення вхідного параметру до змінної поточної кількості досвіду, та встановлюємо значення в змінну.

Це значення ми перевіряємо чи воно є більшим або рівним значенню змінної потрібного досвіду. Якщо так, викликаємо іншу функцію – «Lvl Up».

В свою чергу, «Lvl Up» має наступну реалізацію:

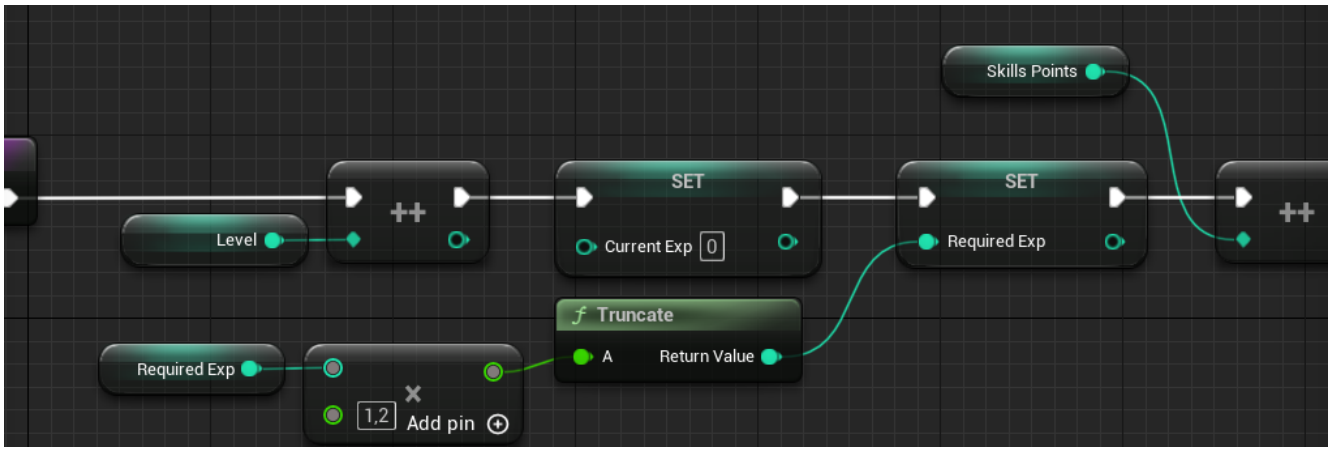

Рисунок 4.20 – Реалізація функції «Lvl Up»

Як тільки метод викликається, значення змінних «Level» та «Skill Points» збільшується на один, значення поточного досвіду стає рівним нулю, а значення необхідного досвіду збільшується в 1.2 рази.

Повернемося до UI для головного героя, та створимо декілька функцій, які будуть встановлювати значення для відповідних текстових блоків.

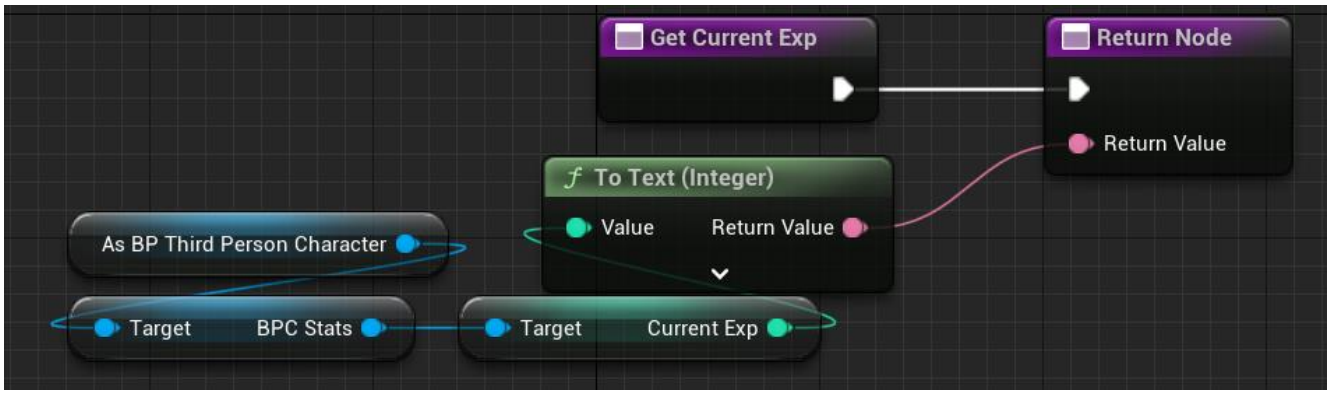

Рисунок 4.21 – Приклад функції для встановлення значення поточного досвіду в текстовий блок

В ньому, ми витягуємо значення змінної поточного досвіду, конвертуємо це значення в тип «Text» та повертаємо як значення. Далі потрібно просто додати прив’язку до текстового блоку. Всі інші функції мають схожу реалізацію.

Для перевірки відпрацювання створимо «KeyEvent» на будь-яку кнопку яка буде виконувати заданні дії при натисканні кнопки. В нашому випадку ми будемо викликати функцію «AddExp».

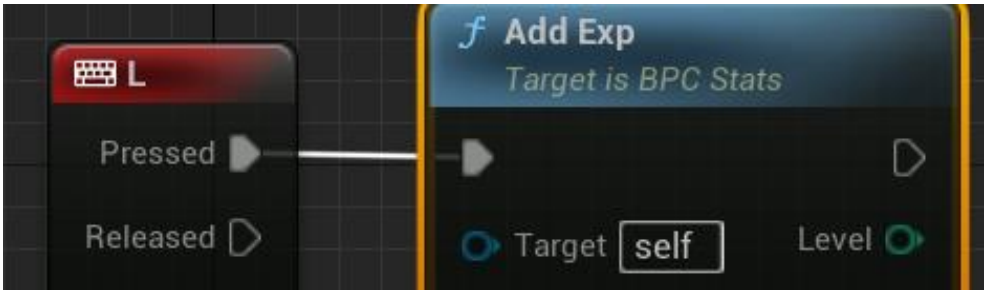

Рисунок 4.22 – «KeyEvent» для перевірки функції

Для симуляції ігрового процесу, значення яке буде додаватись буде випадковим числом в певному діапазоні.

Запустимо поточний рівень у Viewport та перевіримо виконання запрограмованих дії.

Рисунок 4.23 – Демонстрація виконання функції

На рисунку що вище, є три скріншоти, перший це після запуску, значення за замовчуванням успішно встановились. Другий скріншот – після натискання кнопки «L» клавіатури. Третій скріншот – також після натискання, але певну кількість разів, для тригеру функції «Lvl Up». Отже, всі функції працюють правильно.

Реалізуємо функціонал «Skill» системи. Для цього створимо віджет з необхідними компонентами та перейдемо до реалізації перемикання.

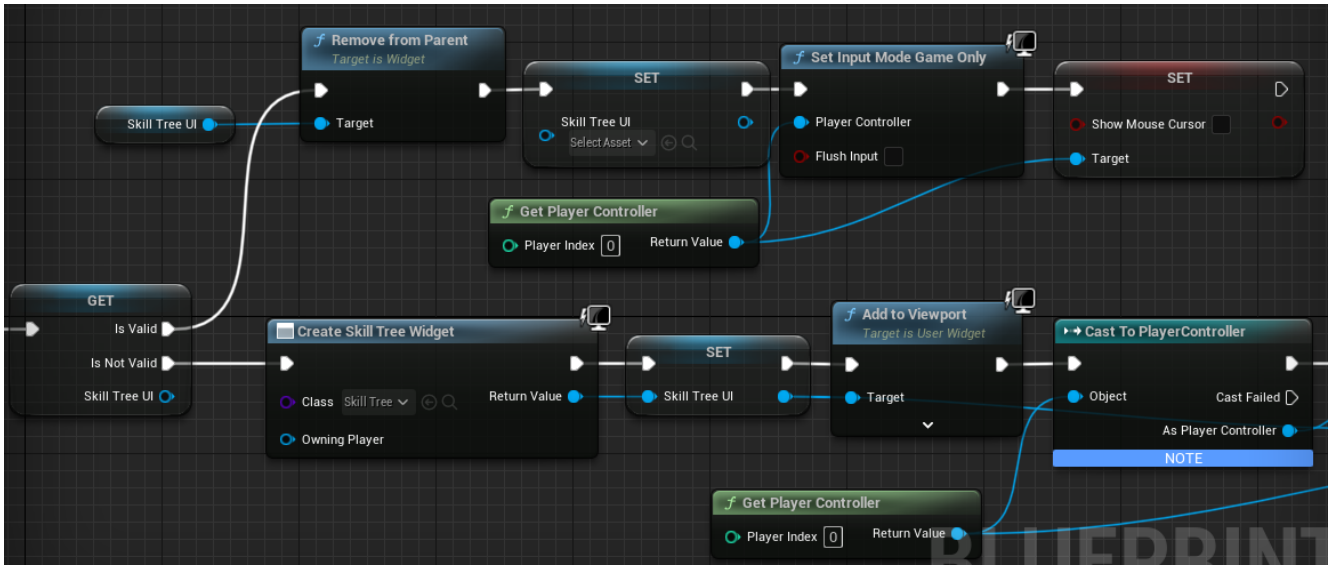

Рисунок 4.24 – Демонстрація імплементація функції

При виклику функції перевіряються валідність даних, в результаті чого або створюється віджет та додається до Viewport, або видаляється.

Віджет з цією системою має наступний вигляд:

Рисунок 4.25 – Демонстрація інтерфейсу системи навичок

Для продовження роботи створюємо функцію яка буде передавати значення «Skill Points» в текстовий блок. Наступним кроком буде створення змінних і функцій, які будуть зберігати значення навички такі як: назва, короткий опис, кількість необхідних поінтів та інші необхідні дані а також створимо декілька функцій.

Почнімо з функції яка буде оновлювати навички, та задавати фон рамки, для кращого розуміння чи ця навичка вивчена, недоступна або можна вивчити.

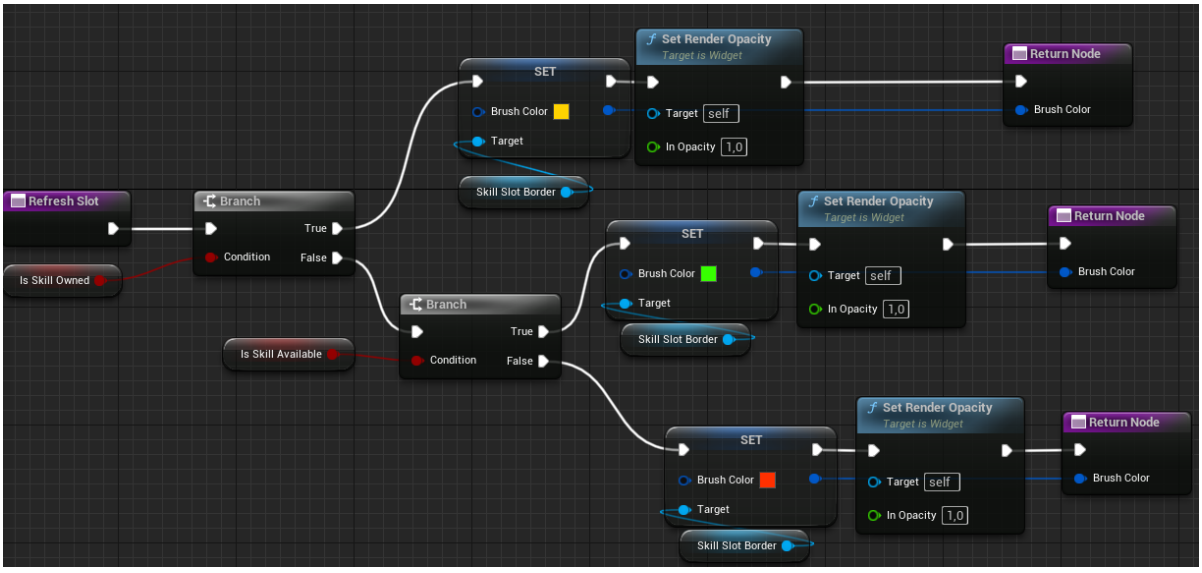

Рисунок 4.26 – Імплементація функції оновлення оформлення навичок

Починається функція з перевірки чи змінна «isSkillOwned» є істиною.

В залежності від результату, в першому випадку встановлюється колір рамки на жовтий, що буде символізувати, що навичка вже вивчена. В іншому випадку буде ще одна перевірка, але для змінної «is Skill Available». В залежності від результату, також буде встановлюватись колір рамки, але на інший. Цей метод повертає значення кольору рамки.

Також можемо створити функцію яка буде змінювати колір текстового блоку, який відповідає за зберігання кількості необхідних «skill points». Приклад реалізації:

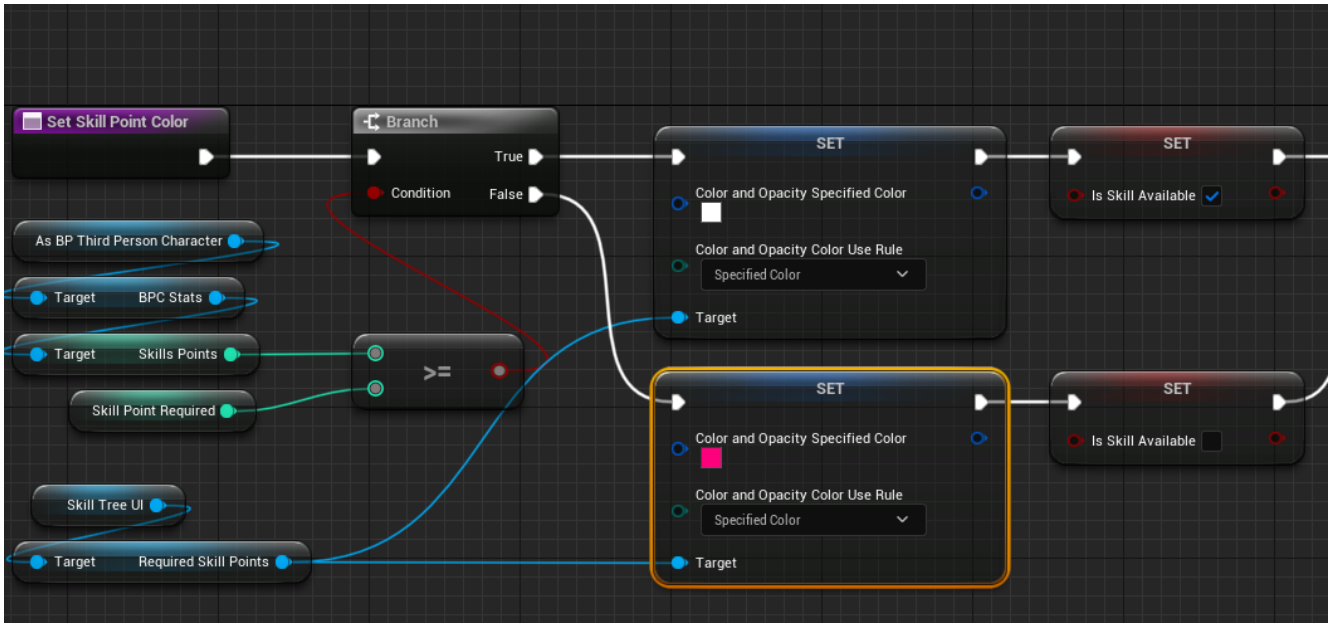

Рисунок 4.27 – Імплементація функції оновлення текстового блоку

Починається ця функція з перевірки кількості наявних та необхідних поінтів навичок, в залежності від результату встановлюється певний колір тексту, а також встановлюється значення булевої змінної, після цього йде додаткова перевірка, та встановлюється текстове значення доступності навичок.

Далі потрібно реалізувати функцію яка буде викликатись при натисканні на навичку.

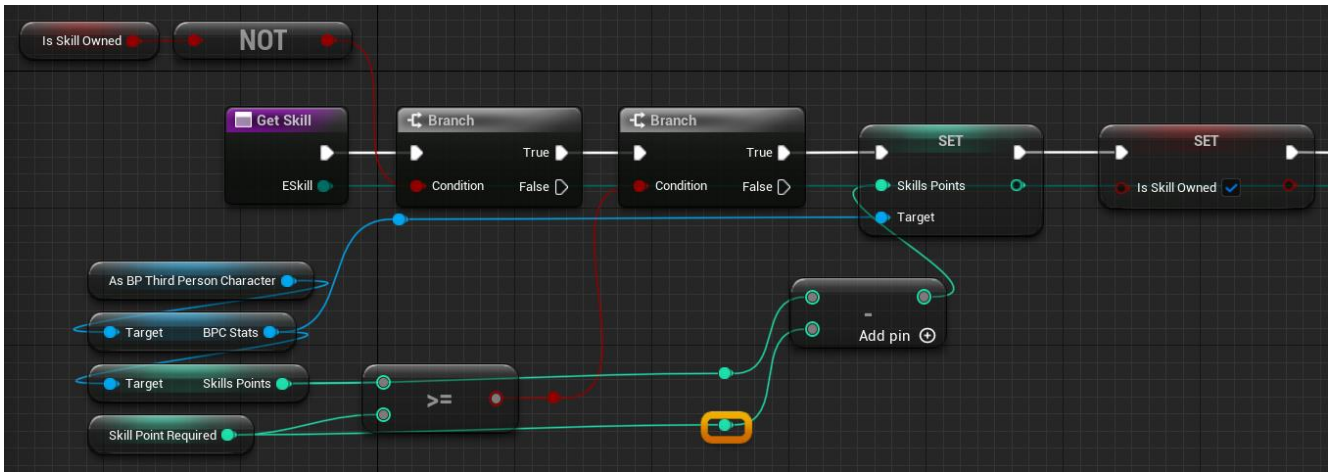

Рисунок 4.28 – Імплементація функції «вивчення навички»

Ця функція також починається з перевірок, якщо в них буде дві істини, то наявна кількість «skill points» буде відніматись від необхідної кількості.

Рисунок 4.29 – Перевірка відпрацювання функцій

На рисунку 4.29 перші два скріншоти відображають початковий стан, коли в нас нема жодного поінту навичок. Наступні два скріншоти відображають коли в нас є один доступний поінт. І останні відображають вивчену навичку та оновлення коли маємо більше одного поінту, а також вивчення двох навичок.

4.4 Реалізація поведінки противника

Створення поведінки для персонажу можна здійснити кількома різними способами. Ми можемо використовувати візуальний скриптінг Blueprint, щоб навчити персонажа "щось робити", наприклад, відтворювати анімацію, рухатись до певної локації, реагувати на удари і тощо.

Але коли вам потрібно, щоб персонажі «самостійно» думали та приймали рішення, тут можуть допомогти дерева поведінки (Behavior Trees). Для створюємо таке дерево та дошку (Blackboard), вона потрібна для зберігання змінних для Behavior Trees.

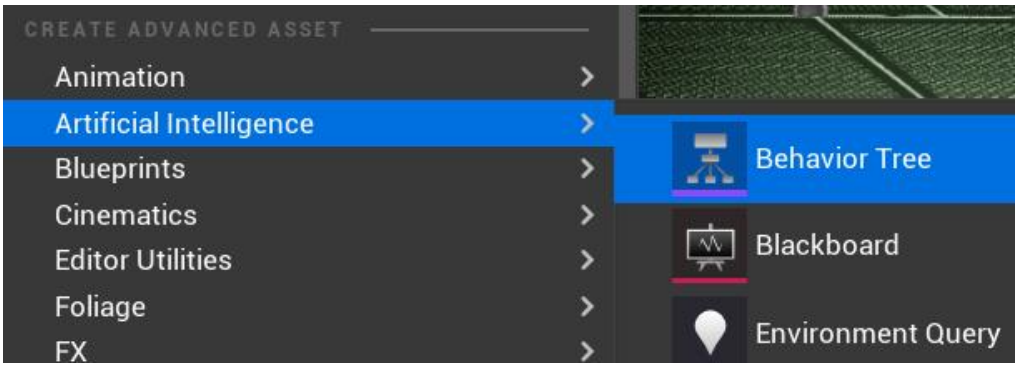

Рисунок 4.30 – Створення дерева поведінки

Нам потрібно буде для початку лише 3 поведінки: патрулювання, переслідування та атакування героя. Після створення дерева поведінки та його відкриття, нас зустрічає «ROOT» нода, яка відповідає за все дерево поведінки. Від нього проводимо лінію вниз та створюємо ноду «Selector», буде виконувати дочірні елементи зліва направо, і зупинятись, коли хоча б один з дочірніх елементів досягне успіху. Від цієї ноди проводимо лінію вниз та створюємо «Sequence», оскільки плануємо вказати послідовність виконуваних дій. Далі створюємо «Tasks», які уже будуть виконувати конкретну потрібну дію.

Створене дерево поведінки можна побачити на наступному рисунку.

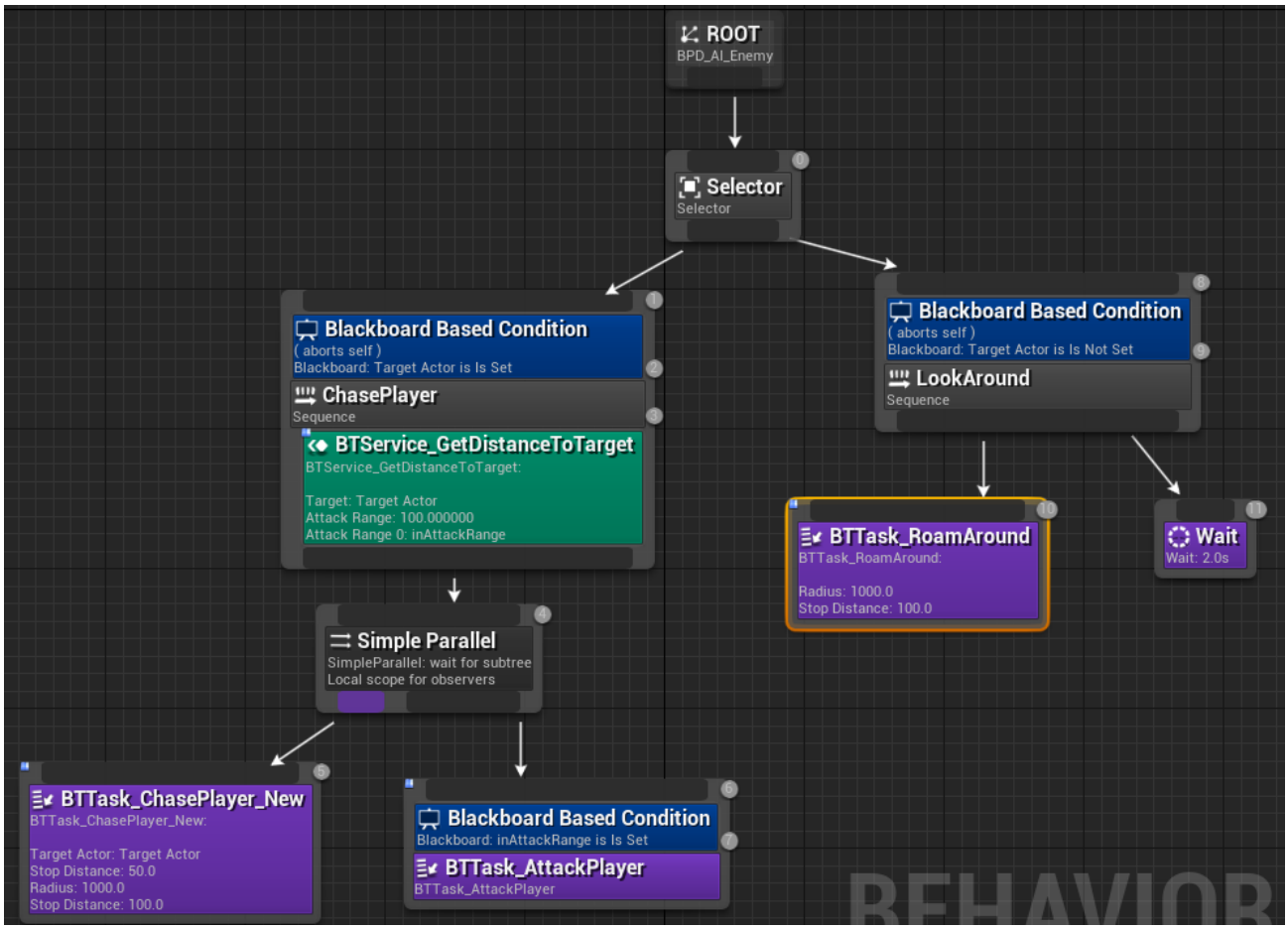

Рисунок 4.31 – Дерево поведінки

Це дерево має чотири Tasks, які виконують певні дії, щоб патрулювати вказану зону, переслідувати та атакувати головного героя.

Task на переслідування має наступний вигляд:

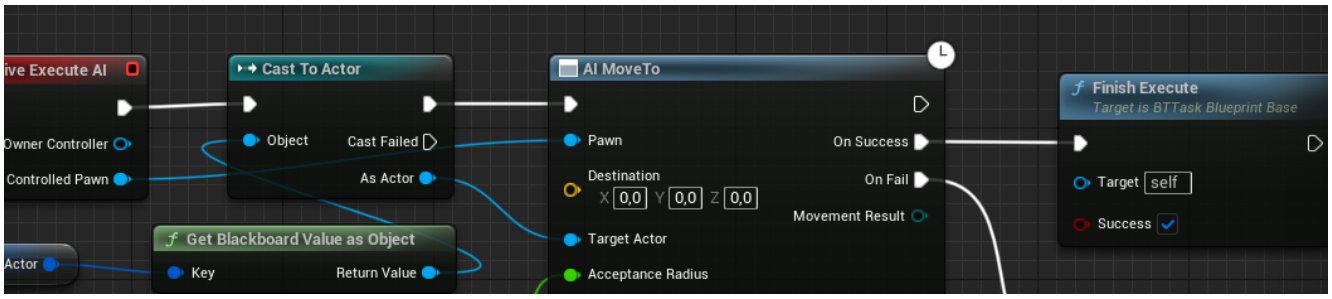

Рисунок 4.32 – Task на переслідування

В ньому витягується змінна з дошки і перетворюється в тип змінної Actor. Далі виклається вбудований метод «AI MoveTo», персонаж буде виконувати цю

задачу, прокладаючи шлях до цільової локації і рухаючись в напрямку цілі з вказаною швидкістю.

Під час виконання задачі «AIMoveTo» персонаж буде автоматично визначати оптимальний маршрут до цілі, обходячи перешкоди, якщо вони є в його шляху. Він також буде орієнтуватися на правильну орієнтацію, щоб рухатися в напрямку цілі.

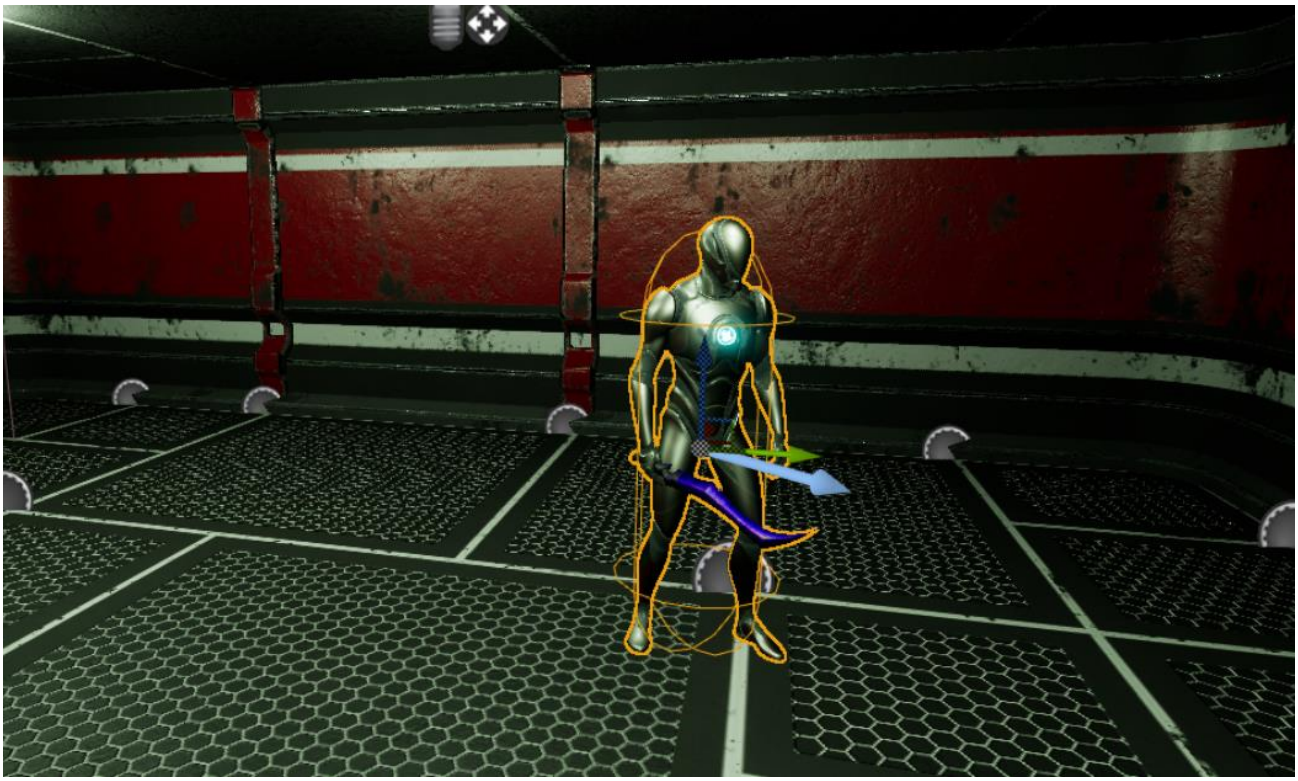

Рисунок 4.33 – Противник на ігровому рівню

Task на патрулювання має майже ту ж саму реалізацію з незначними змінами.

Висновки до розділу 4

Цей розділ починається з опису процес створення меню для ігрового застосунку. Розглянуто використання контейнерів, кнопок, текстових блоків та анімацій для створення елементів меню та їх взаємодії. Також описано виклик функції для отримання поточних налаштувань гравця та встановлення нових значень відповідно до обраних налаштувань.

Також описано реалізацію управління персонажем в ігровому застосунку. Починаючи зі створення «InputAction» для різних дій, таких як переміщення персонажа, керування камерою, лутинг, прокачка навичок та атака персонажа. Було

створено окремий «Blueprint» з типом «Actor Component» для зберігання змінних і функцій, що відповідають за значення рівня, досвіду та іншої інформації про персонажа. Також було реалізовано «Skill» системи. Колір оформлення навичок змінюється залежно від того, чи навичка вивчена, доступна або є недостатньо поінтів навичок.

У цьому розділі було розглянуто реалізацію поведінки противника за допомогою дерев поведінки (Behavior Trees). Використовуючи цей підхід, можна створити складні та гнучкі системи поведінки для персонажів, які дозволяють їм самостійно думати та приймати рішення на основі заданих правил.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було проведено аналіз та огляд Action RPG жанру. Розглянуто його основні характеристики, включаючи можливість гравців зануритися у фантастичні світи, боротися зі злом, розвивати персонажів та досліджувати великі відкриті світи. Також були висвітлені деякі видатні приклади Action RPG ігор, такі як Diablo та серія The Elder Scrolls, що сприяли популярності жанру.

Детально розглянутий процес розробки ігрових застосунків у жанрі Action RPG, який включає концептуалізацію, проєктування, розробку, тестування та реліз. Було проаналізовано різні аспекти розробки, такі як механіка бою, система розвитку персонажа, генерація випадкових елементів гри, система квестів та місій, поведінка ворогів та NPC, анімація, система збереження та завантаження гри, а також розробка ігрового світу та локацій.

Також були проаналізовані три вибрані ігрові застосунки - Mortal Shell, Tales of Arise та Monster Hunter Rise. Були визначені їх основні переваги та недоліки, а також висвітлені їх функціональні особливості.

У другому розділі було представлено огляд історії та основних версій мови моделювання UML. Розглянуто основні поняття та типи діаграм в UML, такі як діаграма варіантів використання, діаграми взаємодії між об'єктами та діаграми станів. Було створено такі UML діаграми: вараіантів використання, взаємодії, станів, діяльності.

У третьому розділі було представлено потужний стек технологій для розробки ігрових застосунків, який включає програми Blender, Substance 3D Painter та Unreal Engine 5.0. Цей стек технологій надає розробникам високі можливості у створенні 3D-моделей, текстурингу та розробки ігрових застосунків з високою якістю візуальної привабливості та геймплею.

Було змодельовано елементи рівня, такі як: 5 типів стін, 2 колони та 2 типи підлоги зі стелею, а також створений ігровий персонаж. Для всіх об’єктів було створено UV – розгортки та створені матеріали з текстури. Окремо для ігрового персонажу була проведене оптимізація сітки.

У четвертому розділі розглянуто процес створення меню для ігрового застосунку, включаючи використання контейнерів, кнопок, текстових блоків та анімацій для створення елементів меню та їх взаємодії. Описано також виклик функції для отримання поточних налаштувань гравця та встановлення нових значень відповідно до обраних налаштувань. Також було створено графічний інтерфейс для гравця.

Наведено реалізацію управління персонажем в ігровому застосунку, починаючи зі створення "InputAction" для різних дій, таких як переміщення персонажа, керування камерою, лутинг, прокачка навичок та атака персонажа.

Також розглянуто реалізацію поведінки противника з використанням дерев поведінки (Behavior Trees).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Action RPG - TV Tropes. URL: http://surl.li/hsefv (дата звернення: 17.05.2023);

2. 52 Best Action RPGs To Play In 2023 - Gameranx. URL: http://surl.li/hsegc (дата звернення: 17.05.2023);

3. The 7 Stages of Game Development. URL: https://www.g2.com/articles/stages-of-game-development (дата звернення: 17.05.2023);

4. Підручник з Umbrello UML Modeller. URL: https://docs.kde.org/stable5/uk/umbrello/umbrello/index.html (дата звернення: 18.05.2023);

5. Introduction – Blender Manual. URL: https://docs.blender.org/manual/en/latest/getting_started/about/introduction.html#who-uses-blender (дата звернення: 04.06.2023);

6. Substance 3D Painter | Substance 3D Painter. URL: https://helpx.adobe.com/substance-3d-painter.html (дата звернення: 04.06.2023);

7. Unreal Engine 5.0 Release Notes | Unreal Engine 5.0 Documentation. URL: https://docs.unrealengine.com/5.0/en-US/unreal-engine-5.0-release-notes/ (дата звернення: 04.06.2023);

8. Creating Modular Game Art For Fast Level Design. URL: https://www.gamedeveloper.com/production/creating-modular-game-art-for-fast-level-design (дата звернення: 04.06.2023);

9. Retopology – Blender Manual. URL: https://docs.blender.org/manual/en/latest/modeling/meshes/retopology.html (дата звернення: 04.06.2023);

10. Texel Density. URL: https://www.beyondextent.com/deep-dives/deepdive-texeldensity (дата звернення: 04.06.2023);

11. Whittle S., Doran J. P., Sherif W. Unreal engine 4. x scripting with C++ cookbook: develop quality game components and solve scripting problems with the power of C++ and UE4, 2nd edition. Packt Publishing, Limited, 2019. 708 p.