Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням
комп’ютерних та видавничих
технологій

Чубей О.О. /_____/

підпис

«___» _____ 2020 р.

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту
освітньо-кваліфікаційного рівня «молодший спеціаліст»
зі спеціальності 122 “Комп'ютерні науки та інформаційні технології”
на тему: “Комп‘ютерна гра в стилі фентезі”

Студент групи К-47 Шувалов В.Ю. _____
(підпис)

Керівник проєкту Кульчинська Н.З. _____
(підпис)

Консультанти:

з техніко-економічного обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Гавришків Н.Г. _____
(підпис)

Тернопіль - 2020

Галицький коледж імені В’ячеслава Чорновола

відділення комп’ютерних та видавничих технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням
комп’ютерних та видавничих
технологій

Чубей О.О. /_____/

підпис

«___» _____ 2019 р.

# ЗАВДАННЯ

## на дипломне проєктування

на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»

студенту Шувалову Володимиру Юрійовичу

(прізвище, ім’я та по-батькові студента)

1. Тема проєкту “Комп‘ютерна гра в стилі фентезі”
затверджена наказом по коледжу від “ 20 ” листопада 2019 р., № 212-н
2. Термін здачі студентом завершеного проєкту “___” _____2020 р.
3. Вихідні дані до проєкту - результати аналізу програмної області ігрових програм в стилі фентезі, результати аналізу попиту на ринку комп’ютерних ігор

4. Перелік питань, які повинні бути розроблені в проєкті:
а) основна частина - опис об’єкту інформатизації, проєктування інформаційної системи, реалізація програмного продукту, тестування

б) техніко-економічне обгрунтування -
аналіз ринку, розрахунок витрат на проєктування, обґрунтування необхідності розробки

5. Перелік графічного матеріалу – схема популярності використання ігрових рушіїв, діаграма варіантів використання, структура елементу “GameInfoInstance”, діаграма діяльності

6. Консультанти проєкту: _Меленчук Л.І._____

| Розділ | Консультанти | Підпис, дата | |
|---|---|---|---|
| | | Завдання видано | Завдання прийнято |
| з техніко-економічного обґрунтування | Меленчук Л.І. _____ | | |

# КАЛЕНДАРНИЙ ПЛАН

## дипломного проєктування

| № п/п | Найменування етапу | Терміни | |
|---|---|---|---|
| | | початку | завершення |
| 1 | Вибір теми, ознайомлення з вимогами до проєктування. | 11.11.19 | 18.11.19 |
| 2 | Огляд типових рішень. | 19.11.19 | 25.11.19 |
| 3 | Дослідження засобів та технологій реалізації проєкту. | 26.11.19 | 15.12.19 |
| 4 | Проєктування програмного продукту: дослідження предметної області, формування вимог до продукту, виділення та представлення функціональних вимог до ПП та робота над структурою програмного додатку. | 16.12.19 | 27.01.20 |
| 5 | Встановлення та налаштування програмного забезпечення. | 28.01.20 | 11.02.20 |
| 6 | Проєктування структури програмного продукту. | 12.02.20 | 15.03.20 |
| 7 | Розробка програмного продукту. | 16.03.20 | 27.04.20 |
| 8 | Доопрацювання модулів, підсистем і т.д. | 28.04.20 | 06.05.20 |
| 10 | Опрацювання питань з техніко-економічного обгрунтування. | 07.05.20 | 20.05.20 |
| 11 | Тестування на налагодження програмного продукту. | 21.05.20 | 25.05.20 |
| 12 | Робота над основним розділом пояснювальної записки. | 26.05.20 | 10.06.20 |
| 13 | Оформлення пояснювальної записки, додатків. | 11.06.20 | 17.06.20 |
| 14 | Попередній захист дипломного проєкту, доопрацювання над проєктом. | 19.06.20 | |
| 15 | Підготовка до захисту дипломного проєкту | 10.06.20 | 25.06.20 |
| 16 | Захист дипломного проєкту | 26.06.20 | |

Дата видачі завдання “_____” _____ 2019 р.

Керівник _____/ /

Завдання прийняв до виконання _____/

# Реферат

Дипломний проєкт. Тема «Комп’ютерна гра в стилі фентезі». Шувалов Володимир Юрійович. Сторінок – 76, рисунків – 64, джерел – 9, додатків – 3.

Об’єкт дослідження – ігрові продукти спроєктовані і розроблені в жанрі First/Third-person Shooter, засоби розробки ігор.

Метою дипломного проєкту є розробка ігрового продукту в жанрі Shooter на базі ігрового рушія Unreal Engine 4.

Завданням проєкту є аналіз засобів реалізації комп’ютерних ігор, проєктування та розробка ігрового продукту в жанрі Shooter.

Для розробки системи було використано середовище розробки Unreal Engine, продукт створено з допомогою програм, таких як:

- Vroid.
- Visual Studio.
- Blender.
- Fuse.

Написання скриптів та тригерів здійснюється за допомогою двох мов програмування: скриптової мови програмування C++ та графічної мови програмування BluePrints.

Результат – запроєктований та розроблений ігровий продукт на базі ігрового рушія Unreal Engine. Продукт має простий і зручний користувацький інтерфейс.

СИСТЕМА, UNREAL ENGINE, C++, BLUEPRINTS, РЕАЛІЗАЦІЯ ПРОЄКТУ, TRIGGER, PLAY.

# Abstract

Diploma project. Subject «Computer game with fantasy style». Shuvalov Volodymyr Yuriyovych. Pages – 76, figures – 64, sources – 9, attachments – 3.

Object of research - gaming products are designed and developed in the genre of First/Third-person Shooter.

The purpose of the diploma project is to develop a gaming product in the genre of Shooter based on the Unreal Engine 4 game engine.

The objective of the project is to develop a gaming product in the genre of Shooter.

To develop the system, the Unreal Engine development environment was used, the product was created using programs such as:

- Vroid.
- Visual Studio.
- Blender.
- Fuse.

For writing scripts and triggers was used two languages: scripted programming language C++ and graphic programming language BluePrints.

As the result of that is a gaming product designed and developed on the basis of the Unreal Engine game engine. The product has a simple and user-friendly interface.

SYSTEM, UNREAL ENGINE, C++, BLUEPRINTS, REALIZATION OF A PROJECT, TRIGGER, PLAY.

ЗМІСТ



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ДП.КН 20.428.21.000 ПЗ | | |
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розроб. | | Шувалов В.Ю. | | | Комп’ютерна гра в стилі фентезі | Літ. | |
| Перевір. | | Кульчинська Н.З. | | | | Арк. 5 | Аркушів 75 |
| Реценз.. | | Посвятовська О.Б. | | | | ГК. ВКВТ. К - 47 | |
| Н.контр. | | Гавришків Н.Г. | | | | | |
| Зав. відділ. | | Чубей О.О. | | | | | |

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

UE – Unreal Engine

VS – Visual Studio

ІТ – інформаційні технології

ПП – програмний продукт

ПЗ – програмне забезпечення

BS – BlueScript

BP – BluePrints

LAN – Local area network

VR – Virtual Reality

RPG – Role-playing game

SAS – Steam Advanced Sessions

AS – Assets Store

AMD - Advanced Micro Devices

AAA – Triple-A

FPS – First Person Shooter

EDSAC – Electronic Delay Storage Automatic Computer

OS – Operating System

IVG – Independent Video Games

CE – CryEngine

PC – Personal computer

GM – GameMode

# ВСТУП

Індустрія комп’ютерних ігор на сьогоднішній день є надзвичайно різноманітною та активно розвивається. Відеогра або комп'ютерна гра – комп'ютерний додаток, який призначений для роботи на ПК.

Сучасні комп’ютерні ігри – це одні з найбільш вимогливих додатків на ПК. Геймери купляють потужні комп’ютери для запуску останніх ігор для плавного процесу гри без обмежень.

Книги, фільми, розповіді – це в багатьох випадках і служить основою для майбутньої гри. Завдяки грі можна зануритись детальніше в історію світу, переживати за улюблених персонажів та добре провести час.

Розробником комп’ютерної гри може бути одна людина, група людей або ж фірма. Комерційні ігри великих масштабів зазвичай розробляються компанією або ж компаніями розробників, які спеціалізуються на комп’ютерних іграх. Але фінансування забезпечує в більшості випадків більша компанія. В іншому випадку, компанії можуть мати свої платформи для поширення ігор і функціонувати без допомоги компаній-видавців.

Через розвиток інді-ігор велика частина розробників комп’ютерних ігор тепер мають можливість роботи над своїми проєктами і без допомоги великих компаній, а також не мають ніяких зобов’язань перед ними.

Інді-ігри (від англ. Independent Video Games) – це комп’ютерні ігри, які були створені одною людиною, або ж групою людей, але без підтримки з боку великих фірм.

Метою даного дипломного проєкту є розробка комп’ютерної гри, яка зможе задовільнити потреби користувача в якісній графічній складовій, ігровому сюжеті та розважальному контенті.

Об'єктом дослідження є відеогра в жанрі First and Third-person Shooter.

Предмет дослідження – методи, технології розробки комп’ютерної відеогри в жанрі Shooter.

# 1 ОПИС ОБ’ЄКТУ ІНФОРМАТИЗАЦІЇ

## 1.1 Характеристика об’єкту інформатизації

Комп'ютерні ігри вперше були розроблені в 1950-х і 1960-х роках. Платформами для запуску були університетські мейнфрейми і комп'ютери EDSAC.

Наразі розробка дорогих за б’юджетом ігор може вартувати десятки мільйонів доларів (простий приклад: серія ігор Call of Duty), разом з затратами коштів, росте і час розробки та її оптимізація.

Відеогра Shooter з елементами Zombie Survival – це жанр відеоігор з знищенням ворогів із різноманітної вогнепальної зброї у головній ролі, але не обов’язково. Для цього жанру характерним є вид «очима» головного героя, або як ще називають – «Вид від першого лиця». Крім того може бути і зміна виду з першого до третього лиця – тобто зміна положення камери для показу самого персонажа, але все інше залишається незмінним.

В грі гравець може керувати одним персонажем, який є озброєним деякою кількістю зброї, а також додатковим багажем – гранатами, ножами тощо. Завданням гравця є відстріл ворогів за допомогою вогнепальної зброї і просуватися по рівнях та локаціях вперед. Зазвичай у шутері пояснюється, що собою являє персонаж гравця, чому на нього полюють вороги, яка їх мета, і яка ціль гравця. Сюжет може подаватися одразу під час гри, за допомогою вступних роликів, розмов персонажів, скриптових сцен і тд.

Кількість здоров'я ігрового персонажа в більшості випадків вимірюється у відсотках. 100% – повний запас здоров’я, 0% – мертвий. Інколи в деяких іграх рівень здоров'я впливає на боєздатність, тобто поранений персонаж не може бігти, через розмиття екрану не можливо влучно цілитися і т. д. Здоров'я логічно повинно зменшуватись як від атак ворогів, так і після падіння з висоти, або ж коли гравець потрапить в пастку та інші агресивні середовища (лаву, кислоту, гарячу пару). Аптечки допомагають відновити здоров’я частково або повністю, залежно від її типу. Також у персонажа може бути броня, вона буде

визначати стійкість персонажа до пошкоджень. Зазвичай броня теж вимірюється у відсотках, або ж у числах.

Перенесення на собі певної кількості зброї і боєприпасів це ще один аспект персонажа, але носити в руках може лише одну зброю. Деякі ігри мають моделі реальних існуючих або історичних автоматів та пістолетів, які намальовані реалістично та мають свої власні назви та інші параметри, такі як місткість магазину, скорострільність, віддачу і точність, кількість боєприпасів. Часто в шутерах присутні корисні предмети, які можна носити з собою – бинти, аптечки, карти, радари, передавачі.

На сьогоднішній день розробники ігор не хочуть також включати в жанр Шутера ще й деякі елементи Zombie Survival, через те, що не всі геймери сприймуть це позитивно і щоб не зіпсувати собі рейтинги на сайтах зі статистикою.

Деякі ігри були не набувають популярності через свою концепцію, дуже багато суперечливих моментів, де сам користувач не міг віднайти суті та знаходив безліч непотрібних речей. Прикладом такої гри може стати «Prey».

Ігри також могли бути зовсім не оптимізовані та з безліччю помилок на програмному рівні. Мапи занадто темні, багато головоломок, які починають з часом набридати. Такою грою була «You are empty».

Фільми та ігри з елементами «Зомбі виживання» вже не такі популярні, якими були у 2008-2013 роках, це вже не так цікаво та здебільшого вони мають просто дуже схожі кроки та моменти – перестріляти усіх зомбі та вижити. І з певного цікавого концепту гри або ж фільму просто перетворюється в набридливу нудну рутину.

## 1.2 Аналіз існуючих рішень

Сьогодні багато компаній займається розробкою комп’ютерних ігор. І кожен розробник чи група розробників мають можливість обрати жанр та платформу для своєї гри. Йдучи в ногу з часом, ігри стають все складніші та більш функціональні.

Кожен розробник обирає певний жанр та найбільш оптимальну платформу для своєї гри та часу, тому що щоразу ігри стають все прогресивніші та з великим набором функцій.

В нашому сьогоденні в індустрії комп'ютерних ігор найпопулярнішими та найприбутковішими жанрами є:

- Shooter.
- Battle Royale.
- Survival.
- RPG.

Ігровий рушій це важливий вибір, вибирати його потрібно з огляду на жанр майбутньої гри. Ігри, які популярні на даний момент були розроблені та досі розробляються на таких рушіях, як: Unreal Engine, Unity, CryEngine.

Розробники відеоігор просуваються все далі і далі в якості та цікавості ігрових світів, графіки, сюжету, діалогів та персонажів, тому що так можна привернути цікавість гравців та отримати прибуток. Не менш важливим фактором є наповнення світу, будинки, та інші деталі.

Розглянемо ігрові продукти, проведемо аналіз та визначимо, які недоліки та переваги є в переглянутих продуктах.

Для прикладу візьмемо гру Left 4 Dead – комп'ютерна гра, яка виконана в жанрі Шутер від першої особи та Хорор виживання.

Left 4 Dead може бути одиночним або ж мультиплеєрним шутером від першої особи, де кожен гравець приєднується до команди виживаючих, щоб перемогти ватагу зомбі. В команді виживаючих може бути як одна людина, так і від 2 до 5 людей. Гра триває впродовж кількох хвиль атак зомбі, які закінчуються, коли всі зомбі мертві (рисунок 1.1, рисунок 1.2).


Рисунок 1.1 – Знімок екрану з гри (момент битви під час атаки)

Перехід між локаціями тут присутній, тобто після закінчення матчу відбувається голосування за певні карти, після цього відбувається зміна локації. В інших випадках гравці  грають тільки на картах, які були вибрані в лоббі для гри.


Рисунок 1.2 – Знімок екрану з гри (гравці в стилі фентезі)

Іншим прикладом було обрано відеогру Overwatch. Це науково-фантастична відеогра жанру командного шутера від першої особи, представлена компанією Blizzard Entertainment. Знімки екрану з даної гри показані на рисунках 1.3-1.4.


Рисунок 1.3 – Гра за певного героя на локації «Ханамура»

Динамічна зміна героїв та безліч ефектів від здатностей персонажів для повного занурення у геймплей.

Рисунок 1.4 – Суперздатність персонажа «Райнхард»

Гра зосереджена на збройному протистоянні двох команд з 6-и учасників, котрі борються на спеціальних місцевостях (картах) в різних регіонах Землі.

Ще один приклад – відеогра Savage The Battle for Newerth. Це мультиплатформенна комп'ютерна гра, в жанрі шутера від першої особи в стилі фентезі. Усі ігрові дії та ситуації проходять в далекому майбутньому наукової фантазії, коли людство відновило суспільство після апокаліпсису. Знімки екрану подано на рисунках 1.5-1.6.

Рисунок 1.5 – Процес гри на певній місцевості

Ця гра проходить тільки онлайн, оскільки вона не включає режим одного гравця. Кожен матч відбувається на карті різного розміру. В одному матчі є дві або більше команд, які можуть бути або людьми, або звірами (більшість карт мають одну команду людини та одну команду звірів, але можлива будь-яка конфігурація). Мета гри – знищити первинну структуру ворога – «Міцний держав» для людської раси, або «Логовик» для раси звірів.

Відсутність будь якого вступного посібника трохи зіпсувала рейтинг гри, але через широкий спектр стилів гравців та красиву графіку для того часу гра отримала високі бали на рейтинг-сайтах.


Рисунок 1.6 – Переміщення до іншого місця переходу

З вище поданих рисунків та опису можемо зробити висновок, що ключову роль в наш час в іграх займає не тільки захопливий геймплей, але і певна історія та хороша графіка, сприйнятлива для гравця.

Для хорошої гри має бути порядок на базі програмного коду – баги тільки шкодять сприйняттю гри користувачем. Також не менш важлива мета та сюжет гри, користувач може просто не зрозуміти для чого ця гра, і навіщо він її завантажив на свій пристрій.

Left 4 Dead це гра, яка продемонструвала ефективність рушія (Source) у свій час, але на даний момент, у зв'язку з великою кількістю сучасних пристроїв на різних платформах у ній з’явилося багато помилок, та багів, особливо це спостерігається на процесорах від AMD старих поколінь. Також через витік початкового коду гри CS: GO, ця гра стала небезпечною для простих користувачів. Є докази того, що через реверсивне підключення для комп'ютерів, хакери змогли отримати доступ до комп'ютерів користувачів та викрасти певну інформацію.

Overwatch – гра, яка є дуже динамічною, вона не є прийнятною для людей, в яких є навіть незначні проблеми із зором. Не зважаючи на свою оптимізованість, у неї також спостерігаються проблеми з оптимізацією на процесорах AMD.

Єдина проблема гри Savage The Battle for Newerth це те, що серверів майже не залишилось. З плином часу, гравці покинули цю гру в пошуках інших ігор, в яких є довготривала підтримка серверів та постійні оновлення від розробників.

### 1.3 Аналіз наявних рушіїв на ринку.

Детальний аналіз ринку показав, що найкращі рушії на даний момент для створення 3D гри та підтримання її в майбутному це:

- Unreal Engine 4.
- Unity.
- Source.
- CryEngine.

Unreal Engine 4 це дуже масштабний рушій, на якому можна творити усе, що заманеться. У ньому реалізована підтримка майже усіх платформ, в тому числі Linux. Радує наявність набір шаблонів майбутньої гри при створення проєкту, що не скажеш про інші рушії. Список шаблонів подано нижче:

- гонки;
- персонаж з видом від «третього» лиця;
- персонаж з видом від «першого» лиця;
- 2D платформер.
- VR проєкт з налаштуваннями.
- та багато інших.

Але це не означає, що гра вже створена – це тільки початкова точка, з якої вже можна щось творити.

Також Unreal Engine 4 має свій магазин з іграми та відокремленим розділом для розробників з асетами (Assets), де можна придбати певні елементи, або ж цілі готові проєкти [2].

Unity це вже рушій, який задіяний в дуже багатьох відомих проєктах, таких як:

- Rust.
- Risk of Pain 1/ Risk of Pain 2.
- Forest.

Цей рушій, через свою будову, не може видавати велику кількість кадрів. Але в нього малий поріг входу для старту відносно Unreal Engine 4 (там потрібно все вчити з нуля), Unity має дуже велику базу розробників, які щоденно записують туторіали та гайди, і ігри можна робити легко.

Unity також має маркетплейс як і Unreal Engine 4, але в ньому набагато більше усіляких додатків та примочок, які можуть дуже допомогти.

Source це рушій від компанії Valve. Так як розробити власну гру з нуля на ньому достатньо важко, розробники зазвичай роблять свої глобальні модифікації, які згодом вже і перетворюють у свої проєкти. Прикладами таких проєктів є:

- Garry`s Mod.
- Fistful of Frags.
- Kreedz Climbing.

Рушій здатен видавати велику частоту кадрів — це потрібно для кіберспорту. Має не дуже інтуїтивний редактор та SDK для імпорту та експорту об’єктів.

CryEngine це вже старий рушій. На ньому проєкти зараз велика рідкість. В нього дуже погана оптимізація, ігри банально не мають підтримки багатопотоковості процесора, через це і маленька частота кадрів. Прикладом масштабної гри на цьому рушії це серія ігор Crysis. Навіть з найпотужнішими процесорами ігри на цьому рушії не могли видати більше ніж 40-80 кадрів, тому, що гра споживала тільки одне ядро процесора.

Люди до сих пір стараються модернізувати цей рушій власноруч, але він має дуже складну та погано написану будову коду, розібрати тяжко. Тому і вибір рушія падає лише на 3 перших зверху.

### 1.4 Обґрунтування доцільності створення ігрового продукту

Передумови шутера можна знайти вже в грі Maze War, розробка якої відбувалась на початку в 1973, а також у грі Spasim, яка з'явилася в 1974 році. Значно пізніше, вийшла гра Doom. Ця гра була однією з таких ігор, яка зробила жанр більш масовим і здобула велику популярність, та сильно вплинувши на розвиток комп'ютерних ігор. Так, після виходу Doom всі подібні ігри іменувалися «клонами Doom». В 1999 році модифікація Counter-Strike для Half-Life також вплинула на жанр.

Протягом останніх років тенденцією серед технологічного суспільства є зміна якості та насиченості ігрового контенту. Люди більше не хочуть купувати те, що вони вже бачили, тому розробникам потрібно кожного разу вигадувати все нові і кращі сценарії, все красивішу і реалістичнішу графіку.

Для розробки комп`ютерних ігор використовують ігровий рушій. Популярність ігрових рушіїв подано на рисунку 1.7

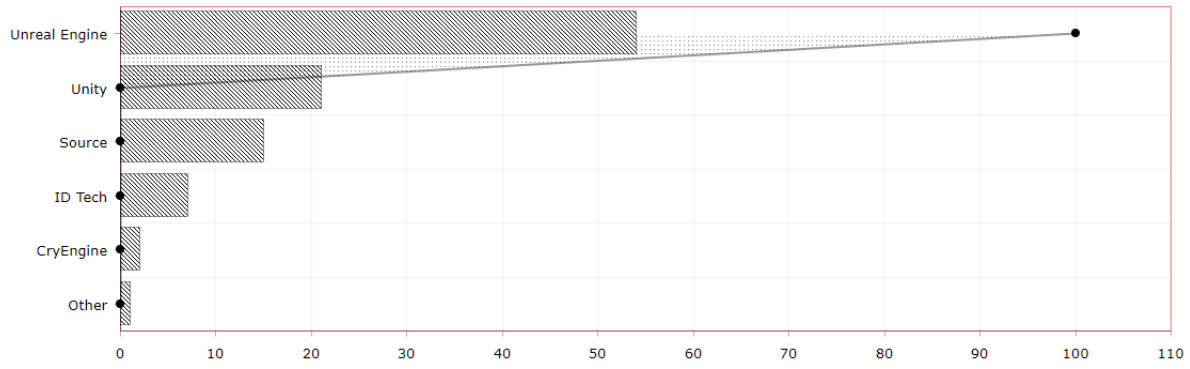



Рисунок 1.7 – Популярність використання ігрових рушіїв

Ігровий рушій Unreal Engine включає в себе максимум інструментів для подальшої розробки та наповнення гри, які можна легко доповнити скачуючи необхідні оновлення в Assets Store. Це знижує захаращення апаратного

забезпечення розробника. Завдяки невеликим вимогам до апаратної складової програмування гри на рушії Unity стає можливим не тільки на комп'ютерах із максимальними показниками продуктивності, а і на звичайних персональних комп'ютерах.

Ігровий рушій Unreal Engine написаний мовою C++, дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X, та для консолей. В рушії UE4 дозволено програмування на декількох мовах, а саме C++ та власна мова програмування BluePrints. Тут присутні: хороша продуктивність та оптимізація та безліч функції, налаштувань та кастомізація [2].

Тому, для даного проєкту було обрано в якості середовища для розробки рушій Unreal Engine 4, а жанр гри «First and Third-person Shooter».

Такий жанр ігрового продукту має досить добрі перспективи тому, що зазвичай такі ігри добре сприймаються користувачами, і навіть при деяких недопрацюваннях чи недогляді зі сторони тестувальників гри, користувачі можуть самі вказати на проблеми, щоб вирішити їх якнайшвидше.

На даний час «шутери» дуже популярні, займають найвищі місця з рейтингів ігор та прямих ефірів на різних стрімінгових площадках, таких як:

- Twitch.
- Youtube.
- Mixer.

Тому, виходячи з усіх вище вказаних факторів, було прийнято рішення про створення ігровго продукту на базі жанру «First and Third-person Shooter». А для різноманітності, буде додано певні елементи фентезі та виживання. Ідея такого проєкту реальна, і її можна «перенести» в життя.

## 1.5 Постановка задачі.

Після детального аналізу наявних ігрових продуктів було сформульовано такі вимоги до даного проєкту:

- добре наповнені локації та детальна пост-обробка;

− реалізація ворогів-ботів для заповнення в разі відсутності людей на сервері;

− зрозумілий інтерфейс без глибокого вникнення;

− наявність добре функціонуючої зброї;

− наявність камери смерті та статистики.

Ігровий продукт повинен виконувати такі функції:

− управління персонажем: дозволятиме користувачеві контролювати персонажа в грі, його дії, переміщення по локаціях за допомогою клавіш “W”, “A”, “S”, “D”, “Ctrl”, “Alt”. Також використання клавіш для взаємодії зі зброєю, або ж персонажами за допомогою клавіш “1”, “2” “3”, “4”, “F1”, “F2”;

− функція перемикання виду: системою перебачено перемикання виду персонажа від першого лиця в вид від третього лиця. Це зроблено для більш кращої кастомізації та вподобань користувача, також це допомагає краще бачити противників;

− якщо в персонажа, яким керує користувач закінчуються “HP” гра закінчується та повертається до останнього збереженого моменту в проходженні сюжетної лінійки гри;

− клавіша “F” відповідає за зміну персонажа, для безпечного пересування та переключення на іншого персонажа.

# 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1 Опис предметної області

Шутер від першої особи (англ. First-person Shooter, FPS) – жанр комп'ютерних ігор, в яких ігровий процес ґрунтується на боях з використанням вогнепальної або будь-якої іншої зброї з видом від першої особи таким чином, щоб гравець сприймав те, що відбувається. В цілому шутери від першої особи мають схожі риси з іншими шутерами, які, в свою чергу, є одним з варіантів екшн-ігор. З часу появи, в цих іграх використовувалася передова для свого часу 2,5D або 3D-графіка, стимулююча виробників до створення більш досконалого обладнання, а багатокористувацькі режими стали їх невід'ємною частиною.

Відповідно до результатів аналізу даного ігрового жанру, популярних ігор та поставлених вимог до ігрового продукту, було спроєктовано ряд функцій, виконання яких повинен забезпечити даний проєкт. На рисунку 2.1 подано діаграму варіантів використання.


Рисунок 2.1 – Діаграма варіантів використання

Уся логіка гри складається з певних запитів та відповідей на них. Користувач, після запуску гри, має доступ до певних кнопок, які виконують дії. Кнопка “Грати в соло” та “Грати в мультиплеєрі” переносить користувача на вибір мапи (по замовчуванню вона одна) та вибору персонажа.

Кнопка “Знайти” виконує запит на пошук інших користувачів в інтернеті або ж в локальній мережі, залежить від вибору пошуку користувачем.

Кнопка “Налаштування” виконує показ меню з налаштуваннями користувача.

Кнопка “Вихід” виконує вихід з гри. Загальний алгоритм робити системи подано на додатку А з використанням діаграми діяльності.

## 2.2 Розробка сценарію гри

Для початку повинна бути обрана певна ідея, що визначає зміст гри. Для написання ідеї існує кілька шаблонів.

У цьому програмному продукті буде застосовано типовий шаблон “шутеру з елементами зомбі виживання” – деякий гравець “Х” повинен вбити наступаючу на нього хвилю “зомбі”. “Зомбі” постійно переслідують гравця, навіть коли гравець заховається за перешкодами, які, по ідеї, мали би його укрити від очей “зомбі”.

Але в цьому проєкті було додано оригінальну ідею, яку можна покращити в подальшій розробці – зміна персонажів в реальному часі. Тобто гравцю тепер потрібно думати за двох персонажів одночасно – одним постійно втікати та ховатись, а іншим персонажем відбиватись від хвилі недругів.

Зазвичай в таких жанрах ігор є певна мета та спосіб здобуття деяких призів, або ж візуальних предметів з певною частотою появи на певній локації (це з'явилось через тенденцію останніх років, розробники додають це майже всюди, щоб отримати матеріальну вигоду для себе).

Сценарій цієї гри базується на житті селян, які мирно проживали на деякій території та працювали – тобто виконували певні однотипні дії. Одного дня поширилась чутка, що деякий фермер побачив на горизонті велике

нашестя незрозумілих організмів, які прямують до поселення та все знищують на своєму шляху. Через страх селян, було прийнято рішення про евакуацію з поселення у зв'язку з нашестям “зомбі”. Таким чином, на локації залишається дівчина із своїм другом, який був з нею поруч усе життя, щоб удвох побороти «нашестя зомбі» та врятувати місце свого проживання від знищення.

Метою цієї гри є захист дівчини, щоб вона не отримала шкоди. Ціль гравця – це відстріл усіх ворогів, наскільки це можливо, та просто виживання.

З кожним інтервалом часу ворогів стає дедалі більше. Також тут є реалізована цікава механіка відстрілу, ідею якою запозичено з класичних шутерів – вцілив в голову ворога – вбив з першого вистрілу. Це мотивує гравця старатись прицільно стріляти, бути вправнішим та справлятись швидше з поставленою задачею.

## 2.3 Обґрунтування технологій та засобів реалізації

Для розробки даного проєкту було обрано ігровий рушій Unreal Engine 4 тому, що це середовище розробки двох і тривимірних додатків та ігор, що працює під керуванням операційних системам Windows і OS X. Цей рушій дозволяє створювати ігри для більшості платформ: Microsoft Windows, Linux, консолей Xbox, Xbox 360, Mac OS і Mac OS X, PlayStation 2, PlayStation 3, PlayStation Portable, Wii, Dreamcast і Nintendo GameCube. Цей ігровий рушій популярний серед інді-розробників. В першу чергу цілком очевидно – в цьому середовищі можна створити гру, яке буде запускатися майже на всіх пристроях, навіть на слабких ноутбуках або смартфонах.

Кожна гра тут виходить унікальною: від найпростіших платформерів до ігор ААА класу. Також для розробників є сайт для публікації своїх проєктів, для тесту іншими користувачами і відгуків.

Переваги використання Unreal Engine.

Дві мови для розробки ігор: C++ та BluePrints – це головна перевага. Мова C++ потрібна для кращої оптимізації ігор на UE4 і також можна написати увесь проєкт лише на цій мові програмування, але зазвичай

використовують дві мови разом. Blueprints це система візуального скриптування, тобто це графічний код, з яким легше працювати. Майже всю логіку можна збудувати саме на цій мові, але без C++ також не обійтись.

Також інша перевага – кросплатформеність, тобто увесь код, що написаний на рушії Unreal Engine 4, може бути перенесений майже одразу на інші платформи – Android, iOS, PS, Xbox. Відповідно, витрата часу на це йде в рази менша [2].

Третя перевага – хороший магазин Unreal Marketplace. Тут наявний величезний вибір різних ассетів, мешей, моделей, скриптів, деталей, ефектів, локацій, деякі з них можна отримати безкоштовно для некомерційного користування.

Остання перевага – це підтримка зі сторони розробників рушія та інших розробників ігор на сайті-форумі Unreal Engine. Майже одразу можна отримати відповідь на своє питання, або ж знайти вже наявні питання з подальшими вирішеннями проблем.

Крім цього, для розробки персонажів гри було використано такі редактори, як Vroid та Blender.

Про редактор «VRoid Studio» мало хто чув, бо розробляють його в Японії на движку Unity (судячи по UnityPlayer.dll). І до недавнього часу, він був доступний тільки для учасників закритого бета тесту [4].

Але буквально недавно, стала доступна рання версія для всіх бажаючих. Наявною проблемою для більшості користувачів це наявні доступні мови інтерефейсу – японська та англійська. Доступні версії як для Windows (x64), так і Mac OS.

Отже, VRoid Studio – це дуже проста програма для створення власних моделей персонажів (чоловічого або жіночого родів) в стилі аніме, або ж фентезі.

Blender – це пакет для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, малювання, після-обробки відео, а також створення відеоігор [3].

Особливостями пакету є малий розмір, висока швидкість рендерингу, наявність версій для багатьох операційних систем – FreeBSD, GNU/Linux, Mac OS X, SGI Irix 6.5, Sun Solaris 2.8, Microsoft Windows, SkyOS, MorphOS та Pocket PC.

Саме Blender було вибрано для створення деяких об’єктів для наповнення локацій та для обробки деяких моделей.

Переваги Blender:

- безоплатне поширення;
- багатоплатформність;
- великий набір функцій.

## 2.4 Проєктування структури системи

Створення гри потрібно почати з повного аналізу того, що поставлено в задачі розробки.

Усі елементи ігрового рушія Unreal Engine 4 представлені у вигляді об'єктів, вони володіють певними характеристиками, і клас визначає доступні характеристики. Будь-який клас є нащадком класу object. Основними класами і об'єктами є такі:

- Актор (actor) – це базовий клас, він містить всі об'єкти, що мають відношення до процесу гри і мають певні координати у просторі.

- Підкласом актора є пішак (pawn), від англ. pawn – той, ким маніпулюють – це фізична модель гравця або об'єкта, який керується у грі штучним інтелектом. Методи і способи керування пішаком описуються спеціальним об'єктом, який називається контролером.

- В даному випадку, контролер описує тільки загальну поведінку пішака під час гри, а такі характеристики, як, наприклад, “здоров’я” або відстань, на якій пішак може “почути” звуки, задаються для кожного об'єкта окремо.

- Світ, рівень (world, game level) – це об'єкт, що описує властивості ігрового простору, у якому розташовуються актори, наприклад, силу тяжіння

й освітлення, наявність опадів, тощо. Може містити такі властивості ігрового процесу, як, наприклад, режим гри, для якої призначений даний рівень.

Так як режим гри відомий, потрібно визначити, що буде містити ті чи інші елементи. Для підтримки виклику інших елементів в площі окремого елементу потрібно описати структуру елемента, який буде тримати в собі зв'язки та пересилання запитів до інших елементів. У цьому проєкті усі елементи між собою зв'язані, та ніяк не можуть існувати окремо.

Єдиним та головним таким елементом буде "GameInfoInstance". Його структура буде подана на рисунку 2.3.

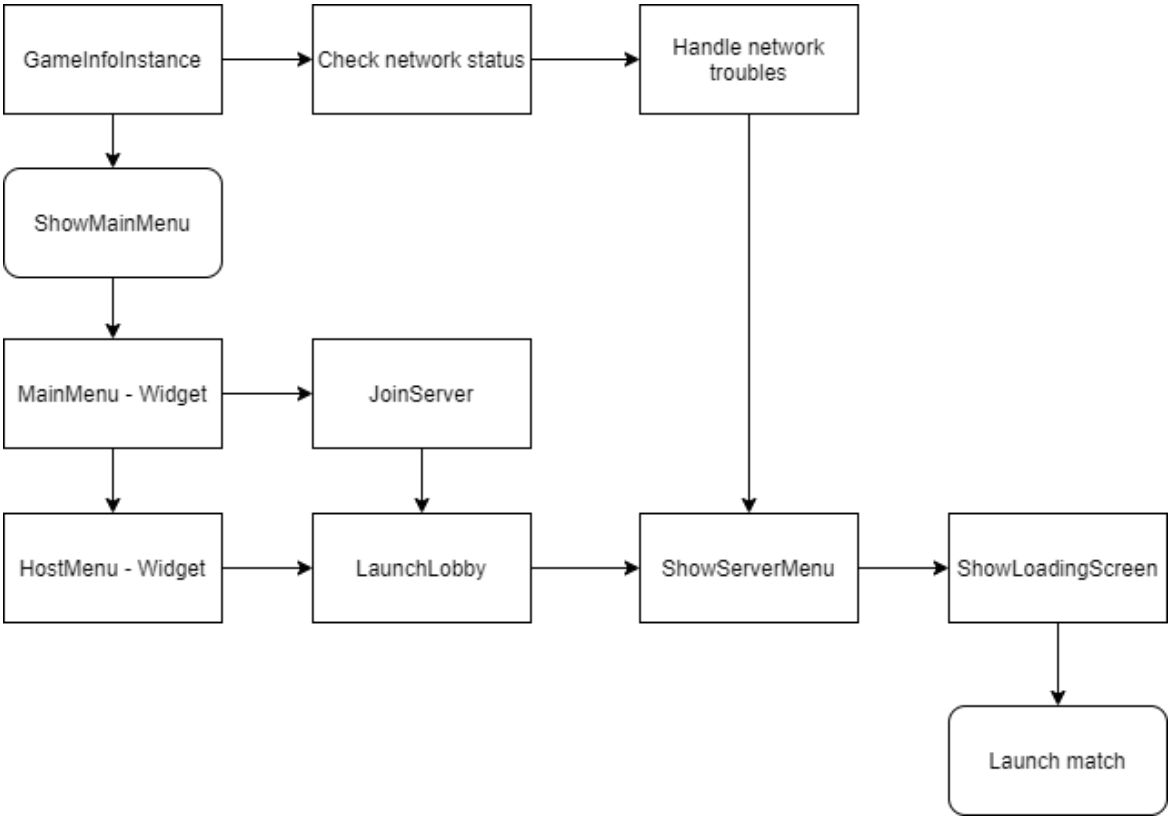



Рисунок 2.3 – Структура елементу "GameInfoInstance"

Елемент "GameInfoInstance" включає в собі такі віджети:

- Widget "Main Menu" – головне меню гри.
- Widget "Loading Screen" – екран завантаження.
- Widget "Host Menu" – меню вибору мапи, керування гравцями, зміна персонажів.
- Widget "Options Menu" – налаштування імені гравця та його аватару.

Також включає передачу даних у інші елементи, які відповідають за інші аспекти гри, мап та персонажів. Це такі елементи:

- Structure “PlayerInfo” – зберігаються дані про гравця.
- BluePrint Class “GameplayGM” – описує створення певного вибраного персонажа гравцем.
- BluePrint Class “GameplayPC” – завантаження інформації про гравця на сервер, підтримка чату.
- BluePrint Class “LobbyGM” – постійне оновлення кімнати з усіма гравцями в реальному часі, додавання та виключення гравців, підтримка стану доступності персонажів, запуск матчу.
- BluePrint Class “LobbyPC” – показ кімнати очікування, оновлення статусу гравця, постійне оновлення стану доступності персонажів, запуск матчу.

Також включає в собі підтримку класів, які відповідають за функції під час гри.

- BluePrint Class “CS_Shoot” – обробка запиту на відтворення звуку пострілів.
- BluePrint Class “AN_Footstep” – обробка запиту на відтворення звуку ходи персонажем.

Також елемент “GameInfoInstance” передає інформацію до персонажів, які будуть задіяні під час гри. Список персонажів:

- BluePrint Class “0_Base” – головний персонаж, яким потрібно утікати.
- BluePrint Class “BP_Player” – головний персонаж, яким потрібно вести перестрілку.
- BluePrint Class “BP_Zombie” – персонаж, який є нападаючим на гравця.

# 3.РЕАЛІЗАЦІЯ ІГРОВОГО ПРОДУКТУ

## 3.1 Реалізація загальної логіки гри

Так як гра буде підтримувати мультиплеєр, написання гри потрібно почати з саме цього. Тому для початку було створено сам проєкт на рушії Unreal Engine 4, після чого через відповідне контекстне меню було вибрано пункт “Blueprint Class”.

“Blueprint Class” – це певний елемент в Unreal Engine 4, який містить в собі опис певної активності та поведінки якогось об’єкту. Це дозволяє творцям контенту легко додавати функціональність поверх існуючих класів геймплея. Креслення створюються всередині Unreal Editor візуально, замість того, щоб вводити код, і зберігаються як об’єкти в пакеті вмісту. По суті, вони визначають новий клас, який потім може бути розміщений на картах, як екземпляри, які ведуть себе, як і будь-який інший тип такого ж класу.

Таких елементів небагато, але вони будуть в собі тримати логіку мультиплеєру та подальших режимів гри. Саме у них було описано наступний код, поданий нижче на рисунках.

Код елемента “GameplayGM” подано на рисунку 3.1.

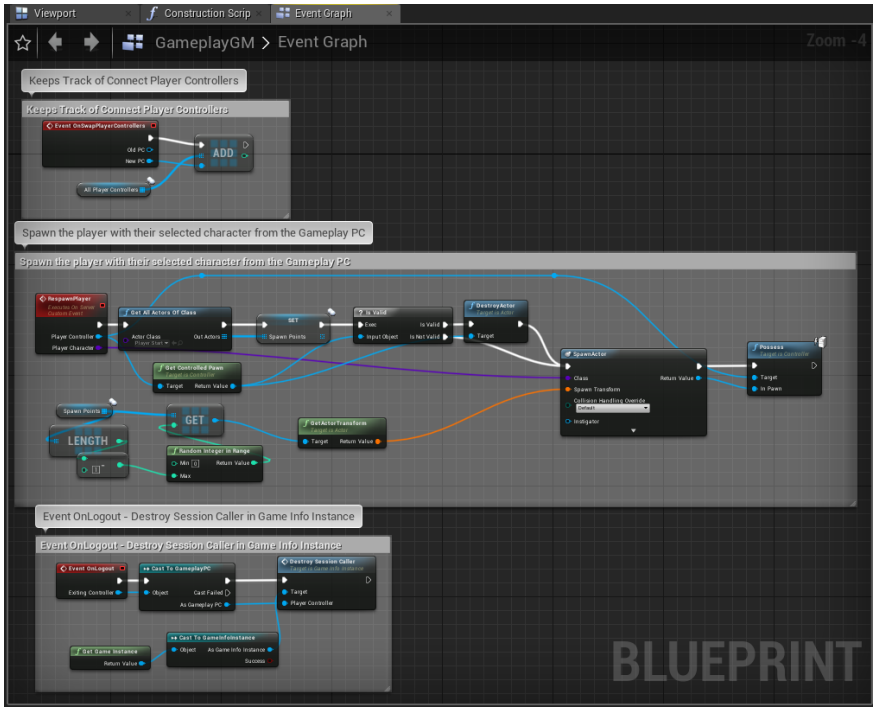



Рисунок 3.1 – Код елемента “GameplayGM”

Код елемента “GameplayPC” подано на рисунку 3.2.

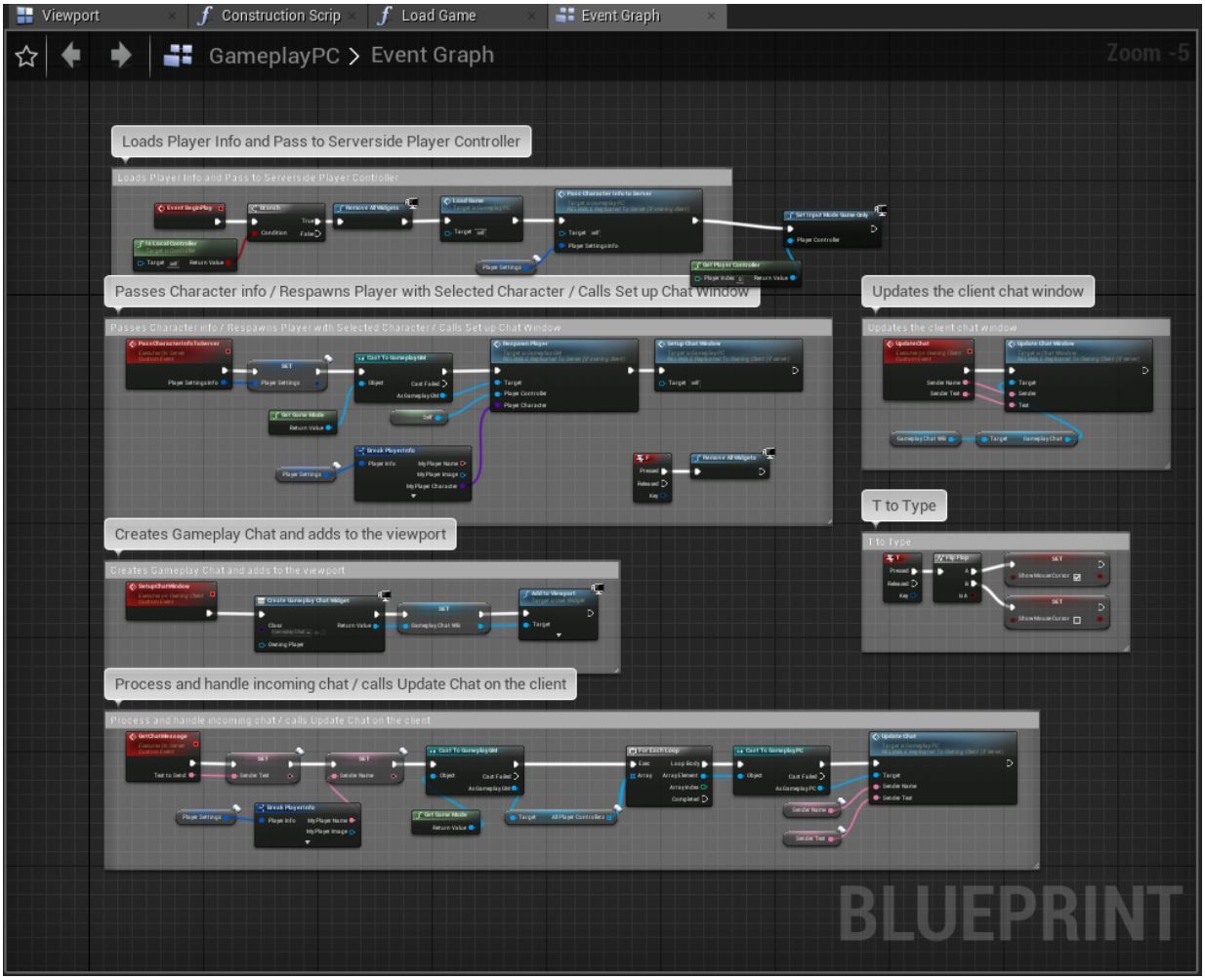



Рисунок 3.2 – Код елемента “GameplayPC”

Для реалізації логіки лоббі гри також було створено декілька елементів “Blueprint Class”. Тут описано багато коду через те, що потрібно врахувати більшість проблем, які можуть бути не очевидні, але бути дуже критичними.

Частини коду елемента “LobbyGM” будуть подані на декількох рисунках нижче.

Перша частина коду елемента “LobbyGM” подана на рисунку 3.3

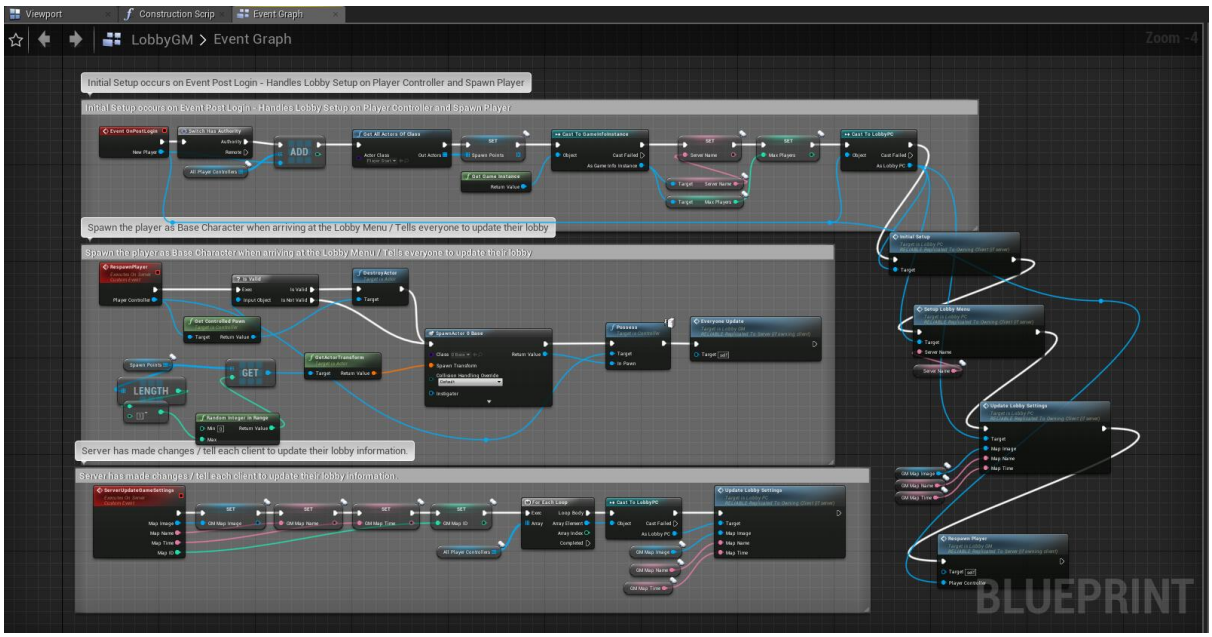



Рисунок 3.3 – Перша частина коду елемента “LobbyGM”

Друга частина коду елемента “LobbyGM” подана на рисунку 3.4

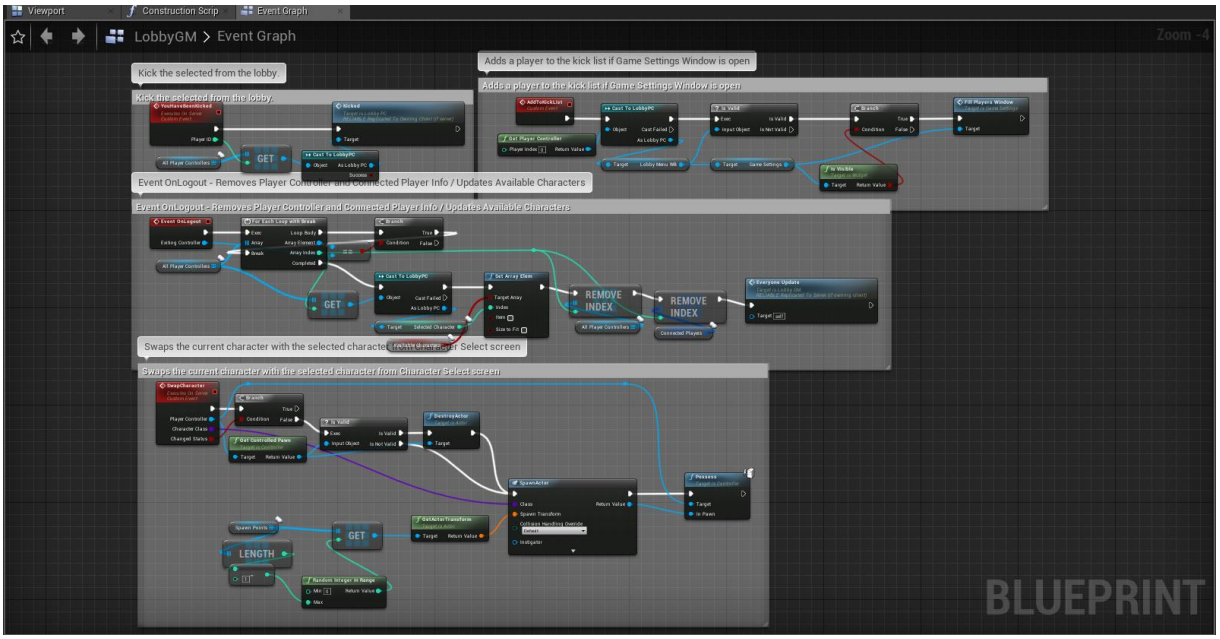



Рисунок 3.4 – Друга частина коду елемента “LobbyGM”

Третя частина коду елемента “LobbyGM” подана на рисунку 3.5

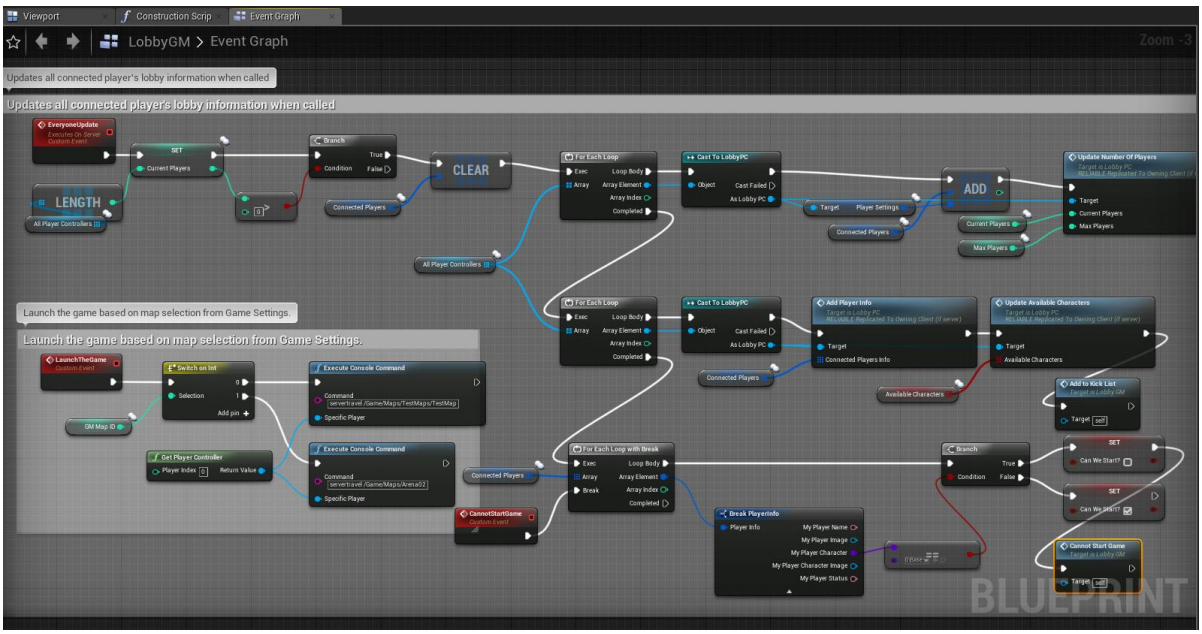



Рисунок 3.5 – Третя частина коду елемента “LobbyGM”

Частини коду елемента “LobbyPC” будуть подані на декількох рисунках нижче.

Перша частина коду елемента “LobbyPC” подана на рисунку 3.6

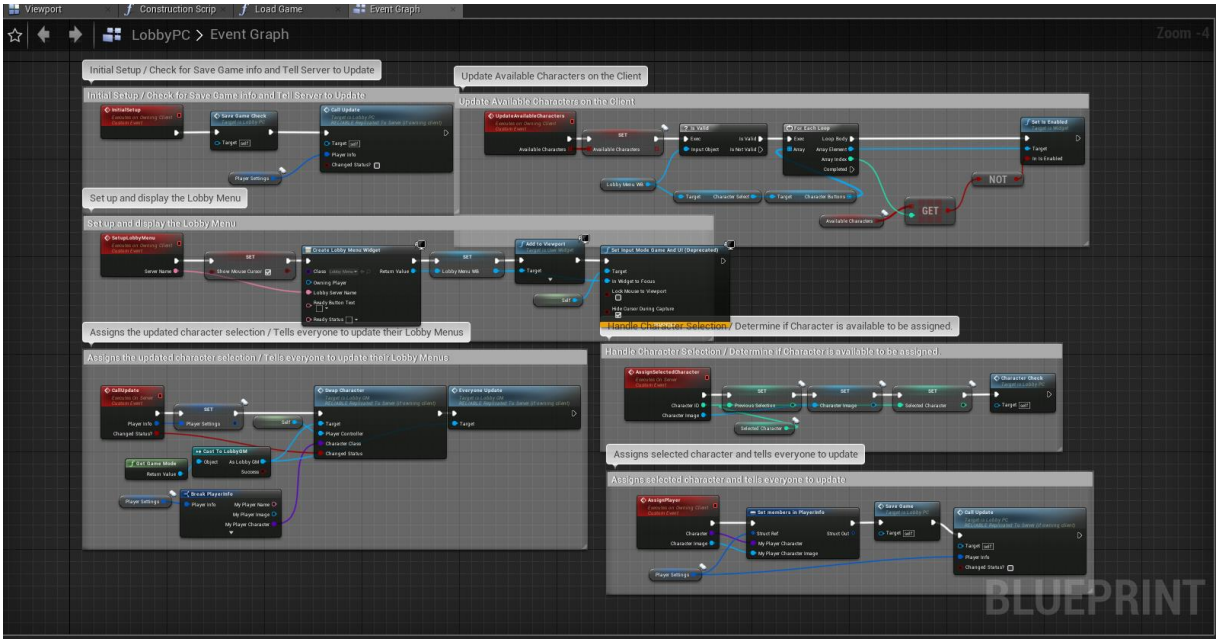



Рисунок 3.6 – Перша частина коду елемента “LobbyPC”

Друга частина коду елемента “LobbyPC” подана на рисунку 3.7

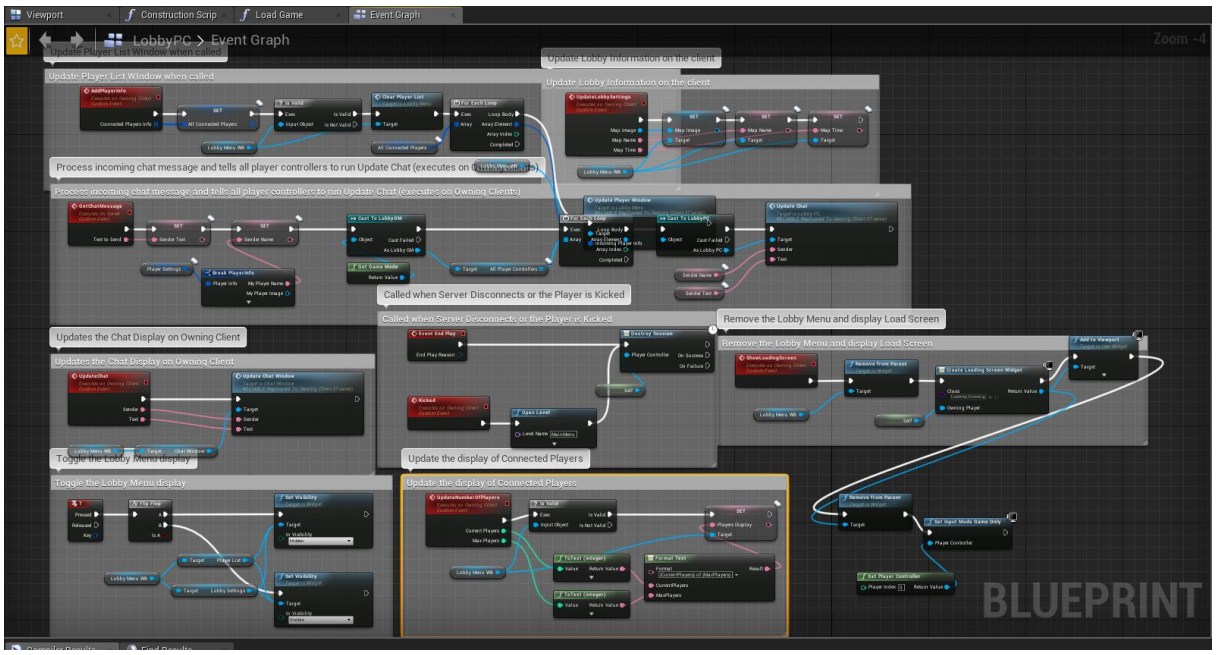



Рисунок 3.7 – Друга частина коду елемента “LobbyPC”

Третя частина коду елемента “LobbyPC” подана на рисунку 3.8

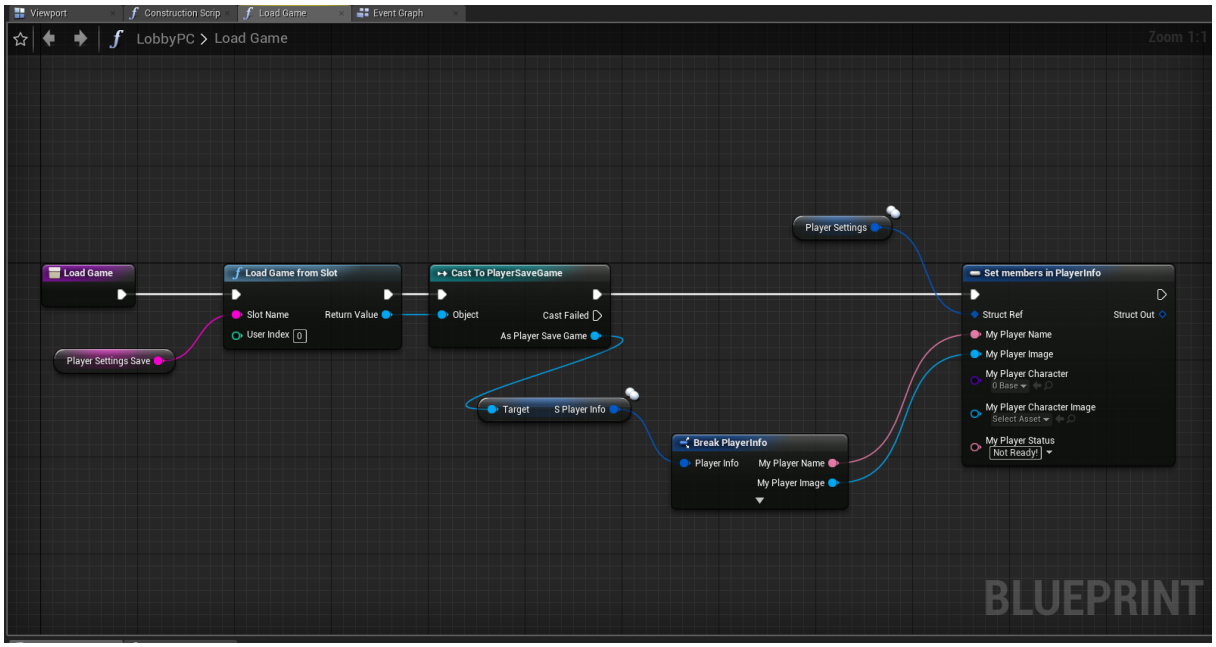



Рисунок 3.8 – Третя частина коду елемента “LobbyPC”

Для написання логіки логіки переходу з одного елемента на інший, та логіку віджетів після різних натиснень було створено ще один елемент під назвою “GameInfoInstance”. Сама його назва вже говорить, що це певні вимоги у грі. Частини коду подані нижче на рисунках [8].

Перша частина коду елемента “GameInfoInstance” подана на рисунку 3.9.

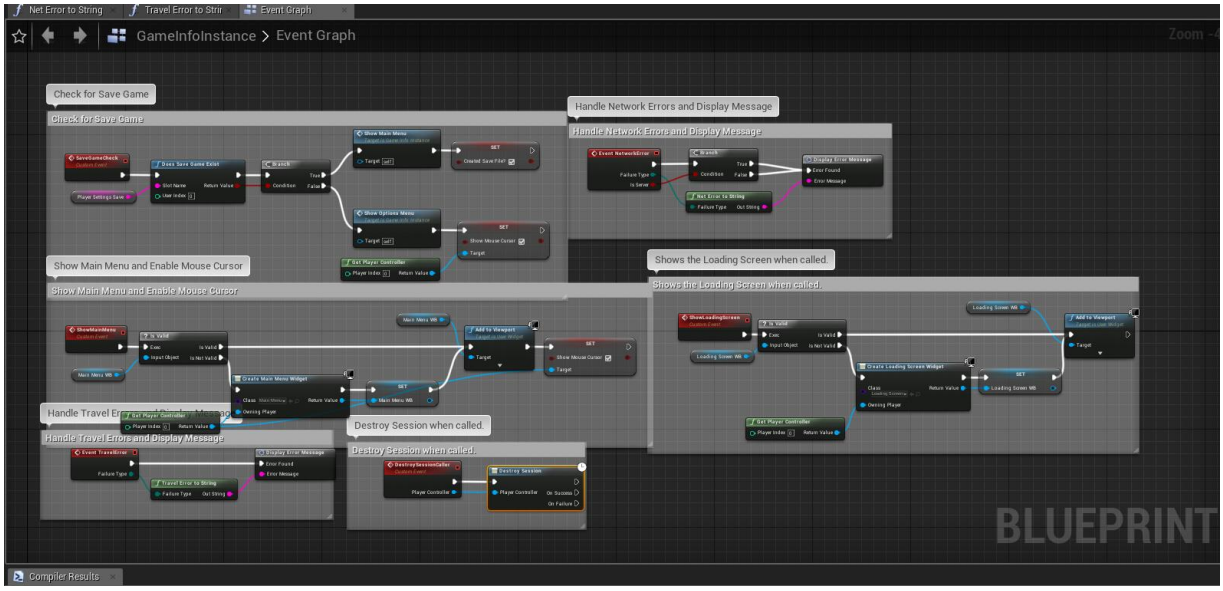



Рисунок 3.9 – Перша частина коду елемента “GameInfoInstance”

Друга частина коду елемента “GameInfoInstance” подана на рисунку 3.10.

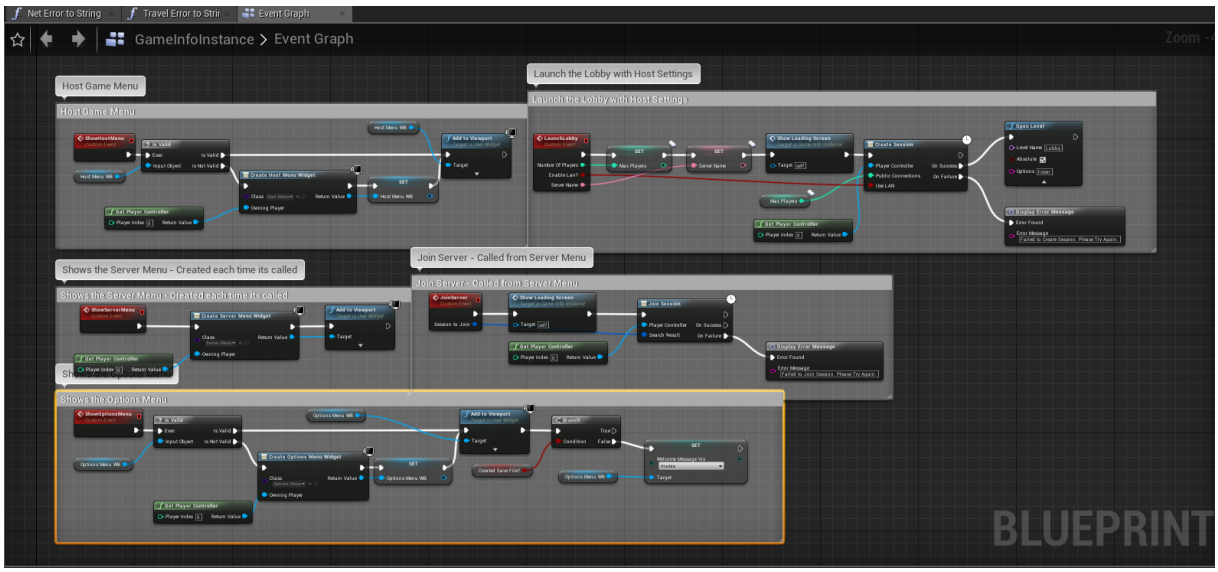



Рисунок 3.10 – Друга частина коду елемента “GameInfoInstance”

Третя частина коду елемента “GameInfoInstance” подана на рисунку 3.11.

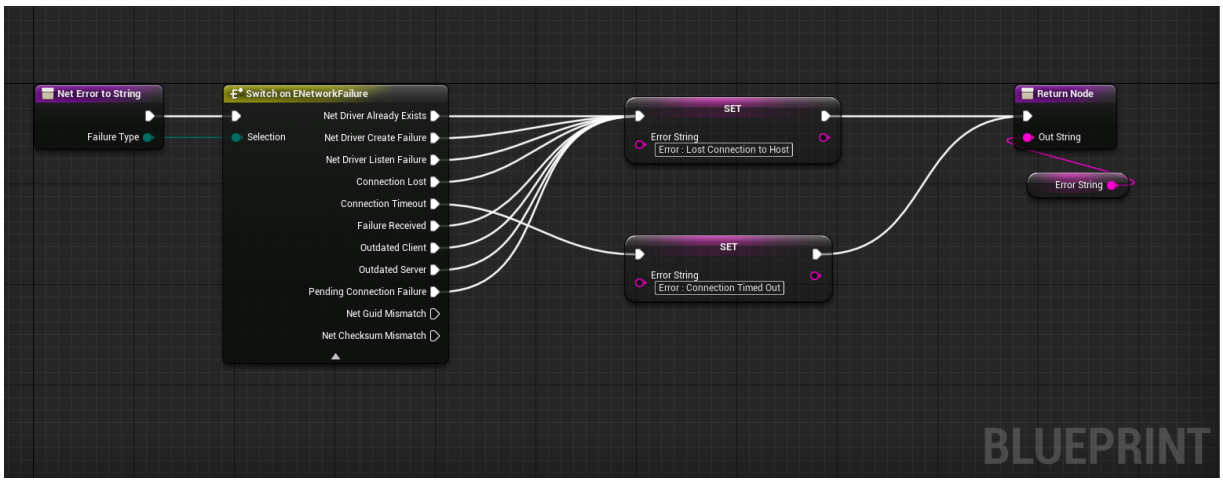



Рисунок 3.11 – Третя частина коду елемента “GameInfoInstance”

Четверта частина коду елемента “GameInfoInstance” подана на рисунку 3.12.

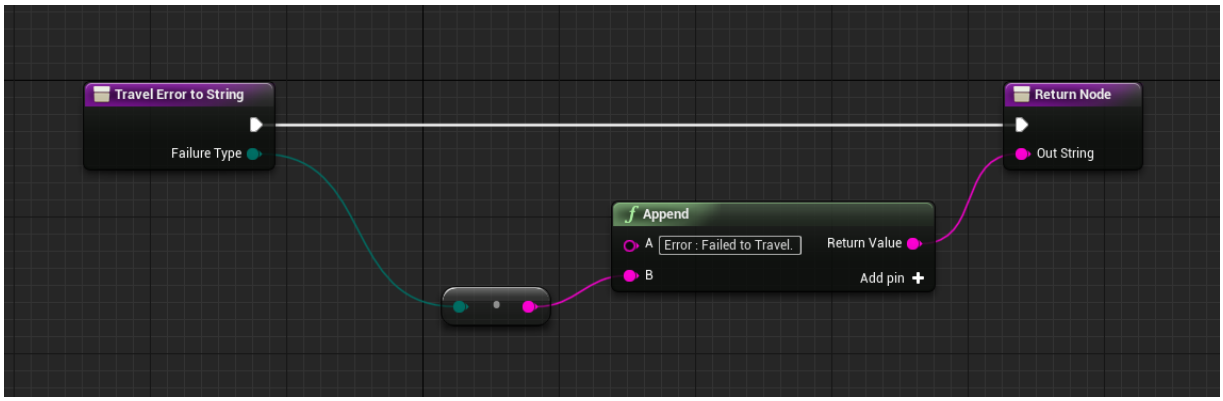



Рисунок 3.12 – Четверта частина коду елемента “GameInfoInstance”

Логіка для персонажа проходить наступним чином. Створюється “Blueprint Class” елемент, в моєму випадку їх 3. Для кожного описується своя фізика переміщення, взаємодія з предметами, тощо.

BP_Player це персонаж, який буде захищати іншого персонажа, тобто у нього буде зброя, а це додаткові анімації, тож це додає певної складності в написанні коду.

Вигляд персонажу подано на рисунку 3.13.


Рисунок 3.13 – Вигляд персонажу “BP_Player”

Код персонажа подано частинами нижче. Перша частина коду елемента “BP_Player” подана на рисунку 3.14.

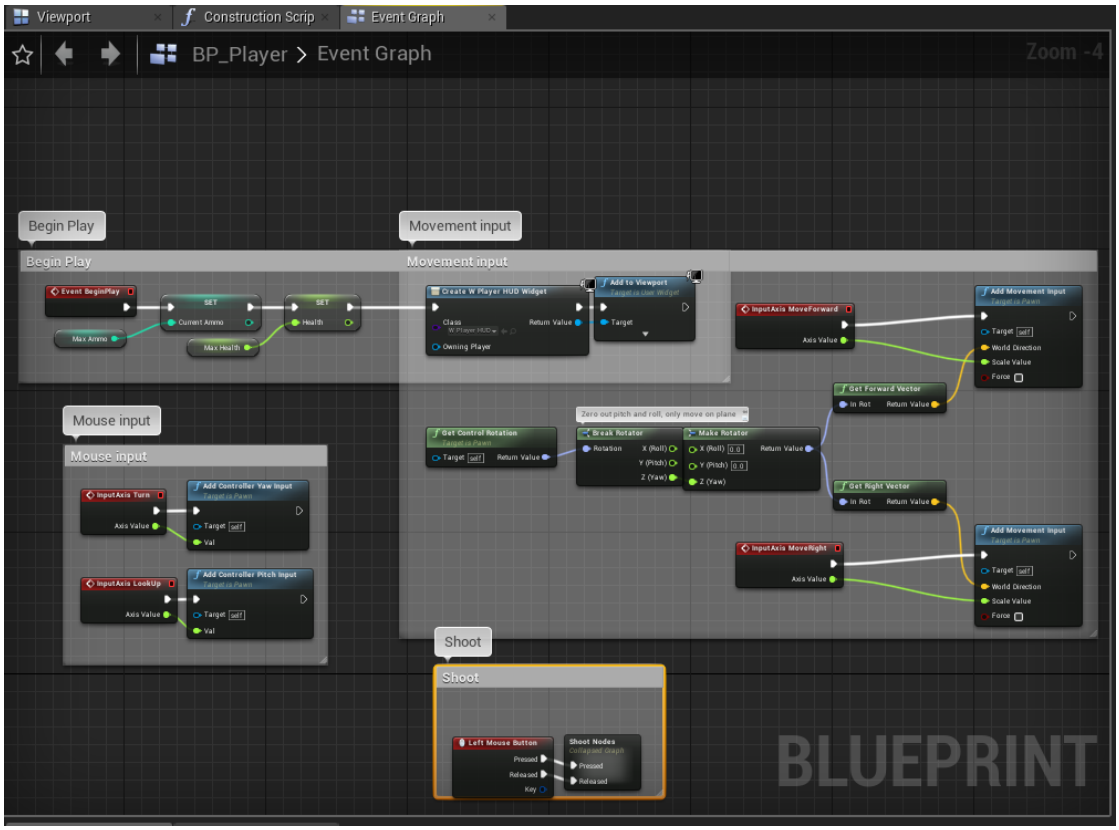



Рисунок 3.14 – Перша частина коду елемента “BP_Player”

Друга частина коду елемента “BP_Player” подана на рисунку 3.15. Тут описано логіку перезарядки зброї, отримання шкоди від ворогів та прицілювання зброєю, тобто як саме буде стріляти персонаж в тому чи іншому стані.

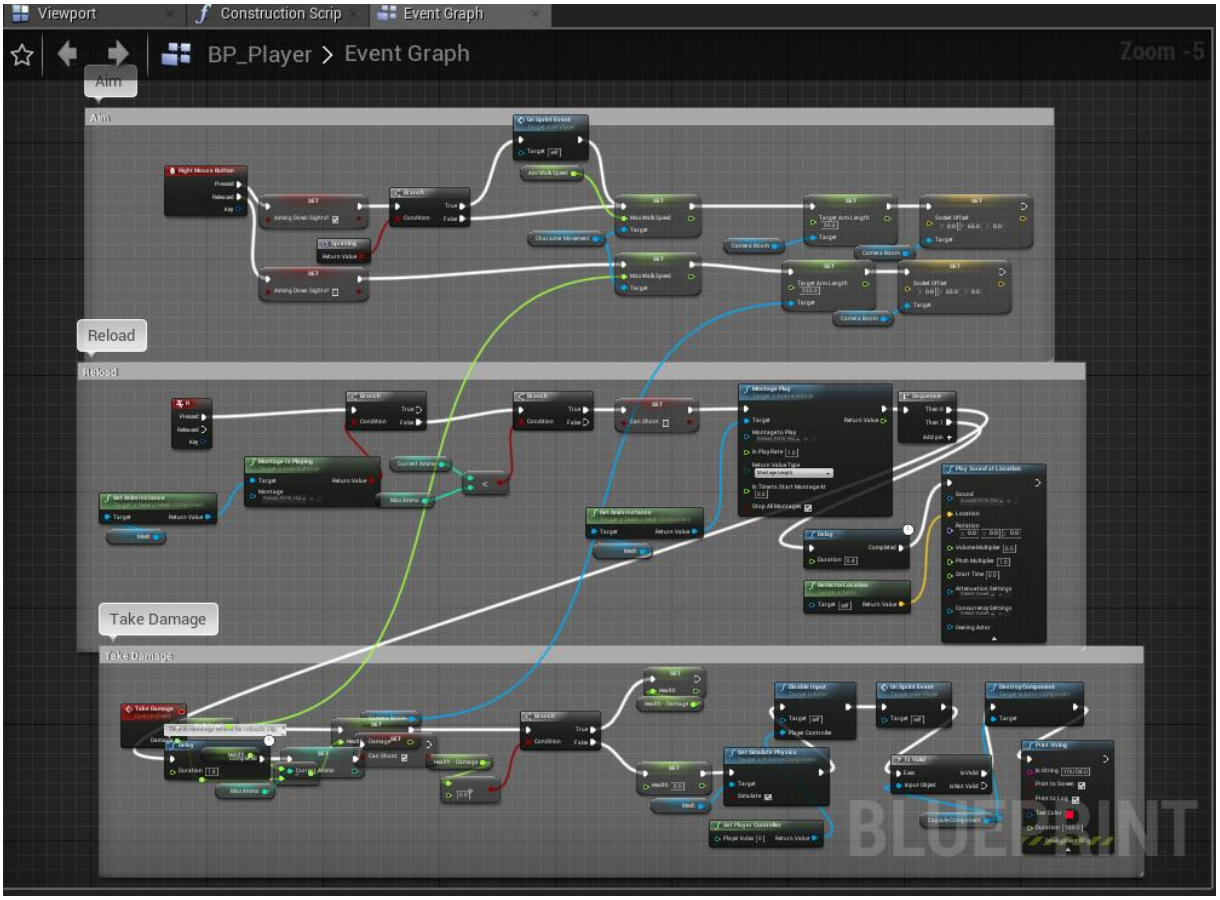



Рисунок 3.15 – Друга частина коду елемента “BP_Player”

Третя частина коду елемента “BP_Player” подана на рисунку 3.16. Тут описано логіку стрибка персонажа, його біг (з перевіркою на зациклення, дуже часта помилка), меню паузи під час гри. Зроблено це через те, що код буде виконуватись на усіх локаціях, тому не потрібно нагромаджувати той самий код декілька разів.

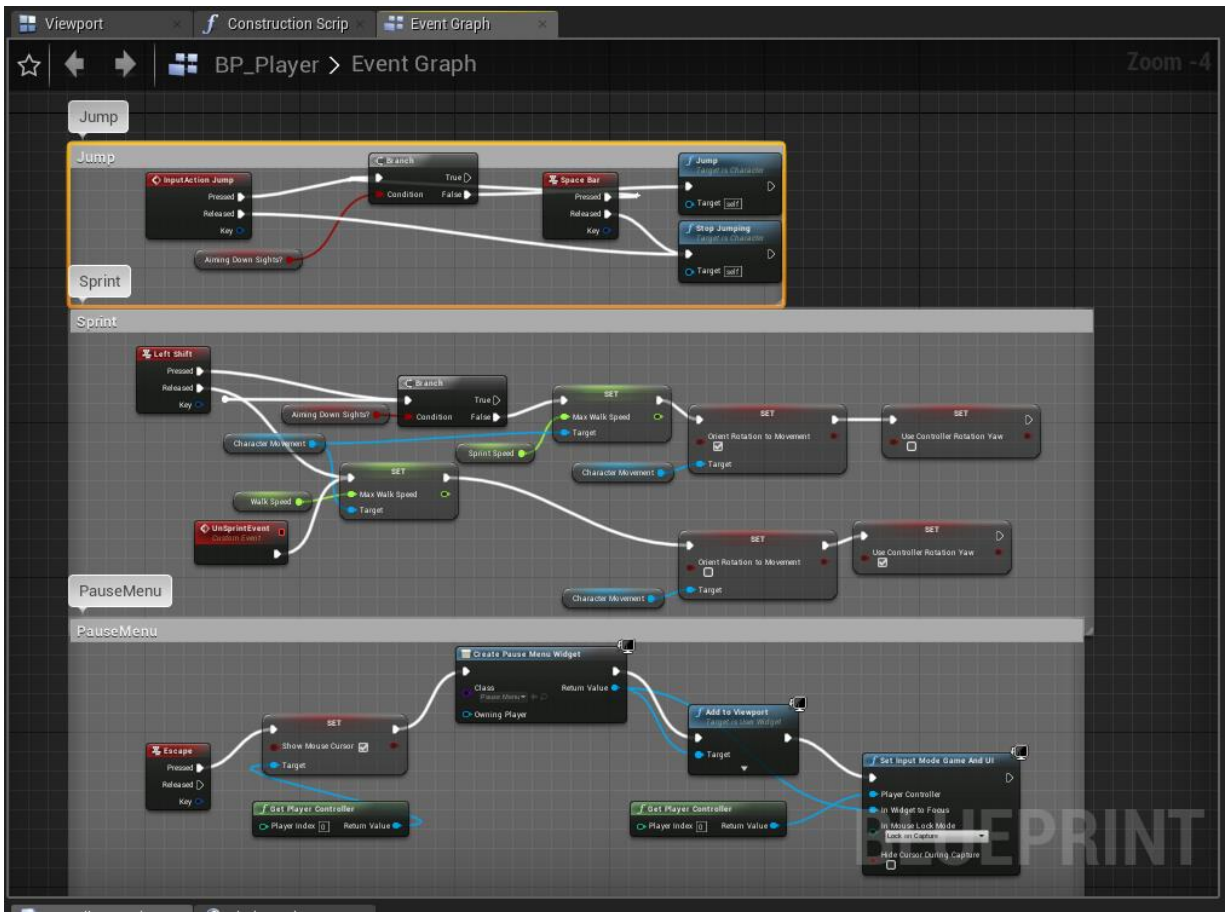



Рисунок 3.16 – Третя частина коду елемента “BP_Player”

0_Base це персонаж, який буде під захистом іншого персонажа, тобто у нього не буде зброї, у нього немає багато коду. Але у цього персонажа присутня зміна камери від першого лиця до третього. Камера на голові у цього персонажа свідчить про те, що в нього “правдивий” вид від першого лиця. Тобто, іншими словами, реалістичне хитання головою, а не статична незворушна сцена. Вигляд персонажу подано на рисунку 3.17.

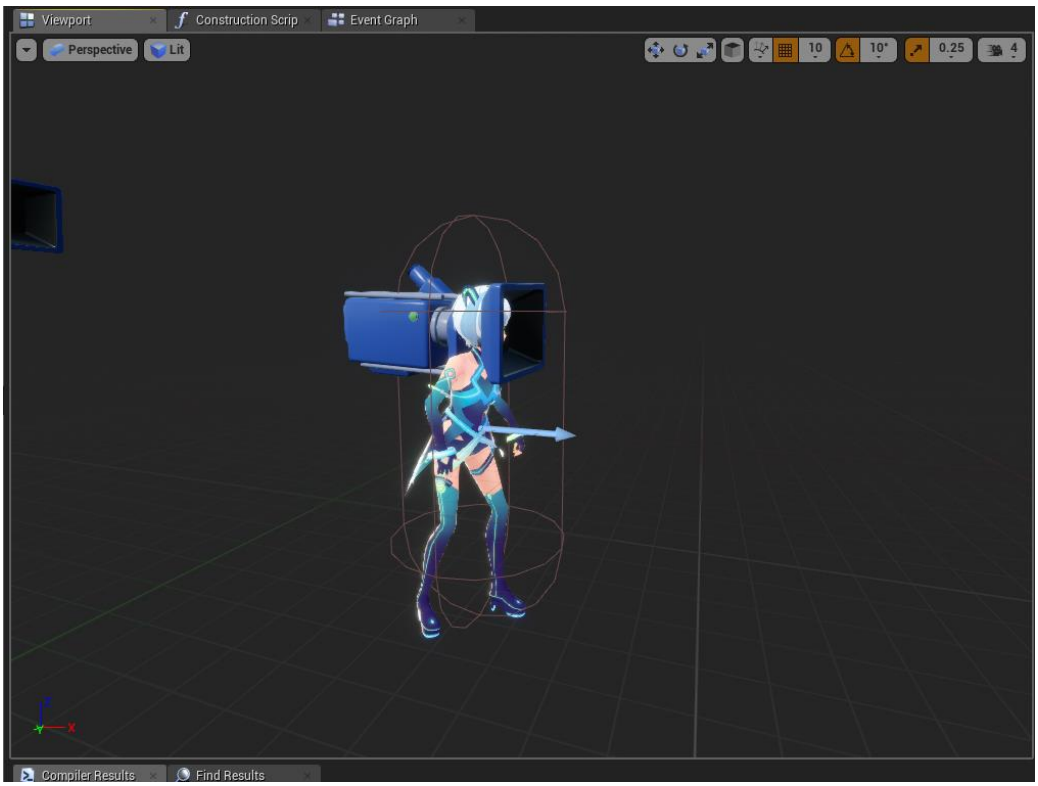



Рисунок 3.17 – Вигляд персонажу “0_Base”

Код персонажа “0_Base” подано на рисунку 3.18.

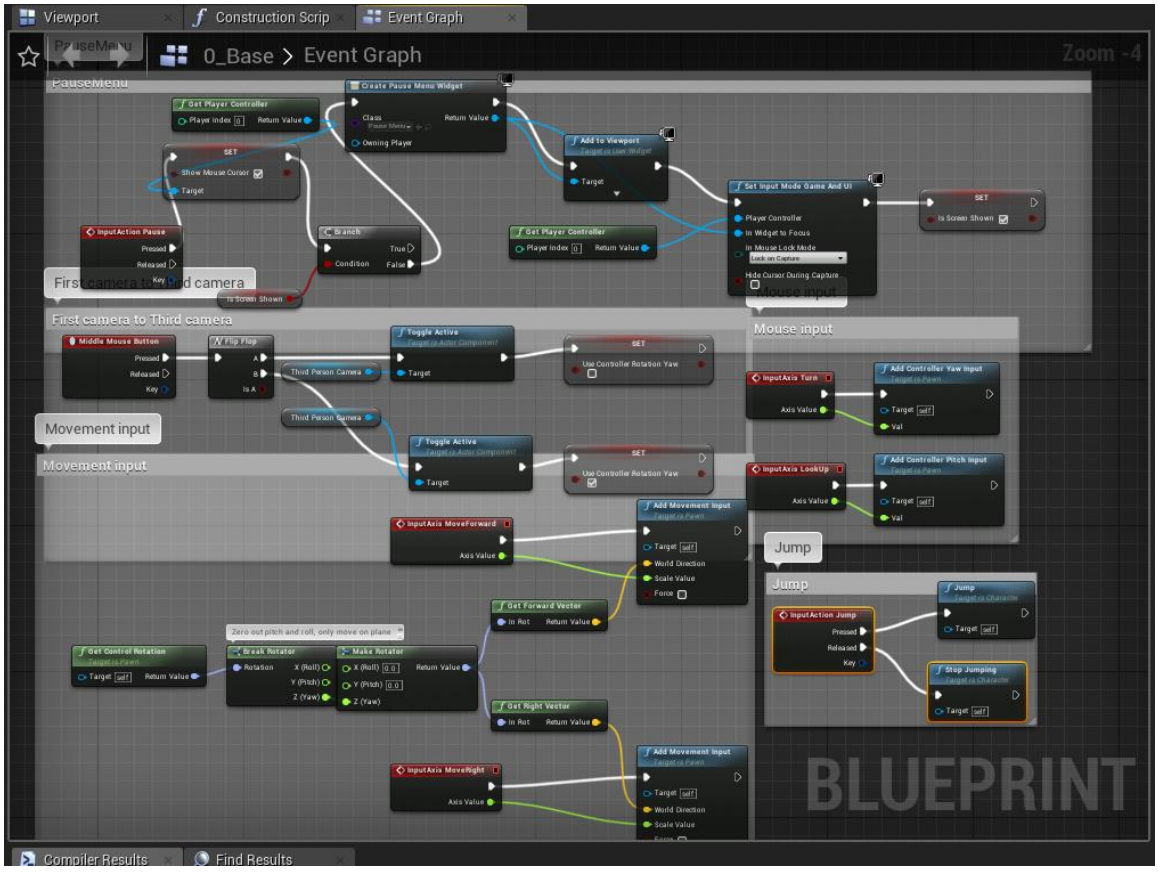



Рисунок 3.18 – Код елемента “0_Base”

BP_Zombie це персонаж “зомбі”, який буде нападати на гравця [9]. Вигляд персонажу подано на рисунку 3.19.

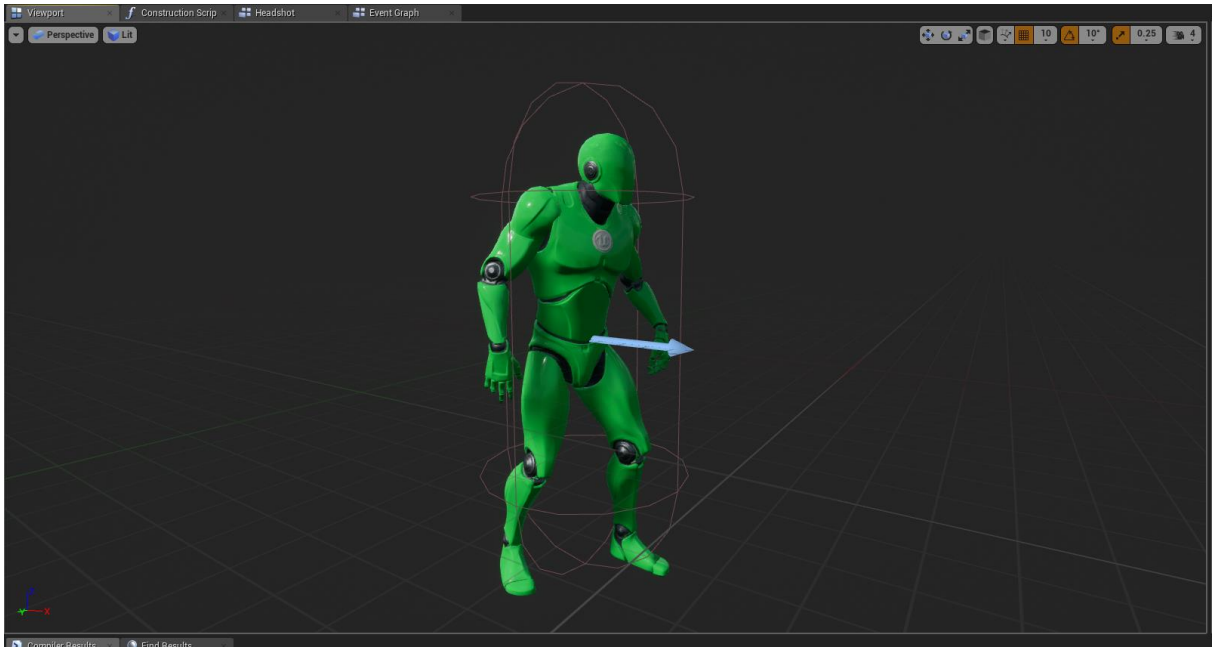



Рисунок 3.19 – Код елемента “0_Base”

Перша частина коду персонажа “BP_Zombie” подано на рисунку 3.20. Це логіка обробки попадання гравцем в “зомбі” в голову.

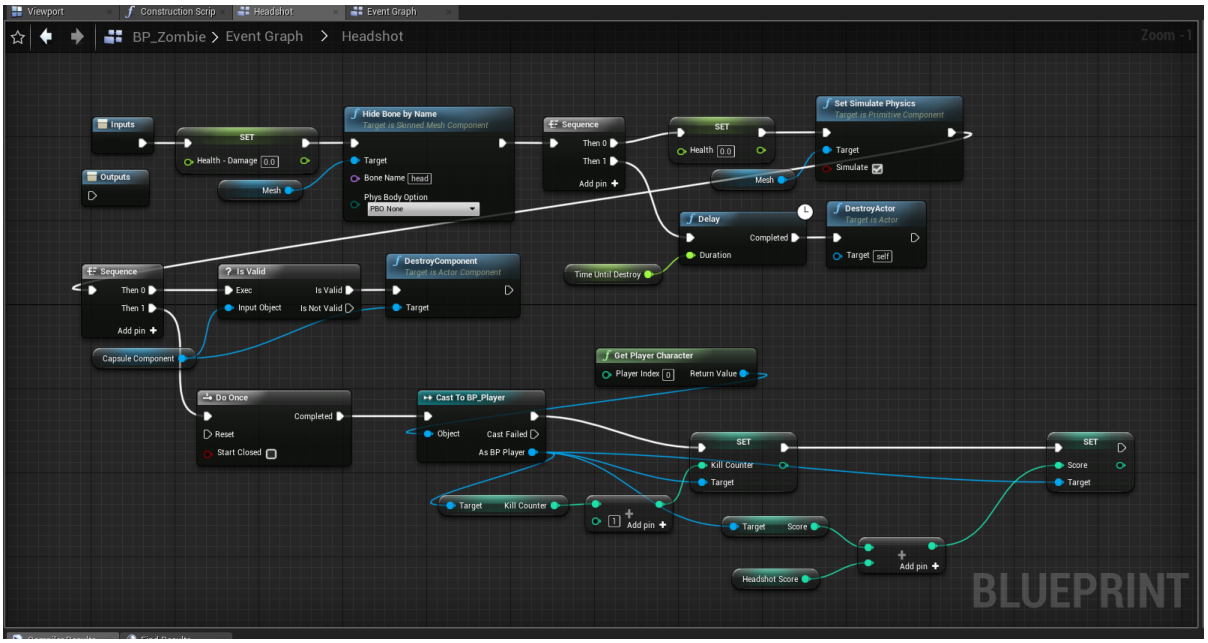



Рисунок 3.20 – Перша частина елемента “BP_Zombie”

Друга частина коду персонажа “BP_Zombie” подано на рисунку 3.21. Цей код відповідає за переслідування гравця.

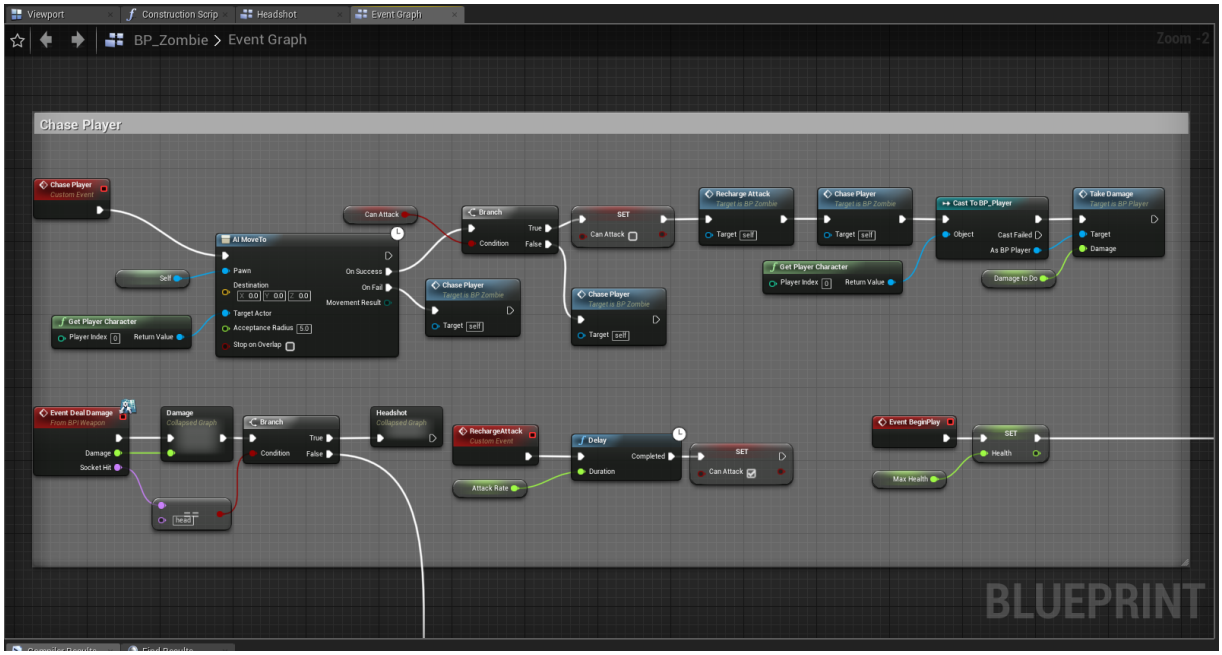



Рисунок 3.21 – Друга частина елемента “BP_Zombie”

Третя частина коду персонажа “BP_Zombie” подано на рисунку 3.22. Цей код відповідає за смерть персонажа та результат падіння тіла (з головою чи без, враховуючи як влучить гравець).

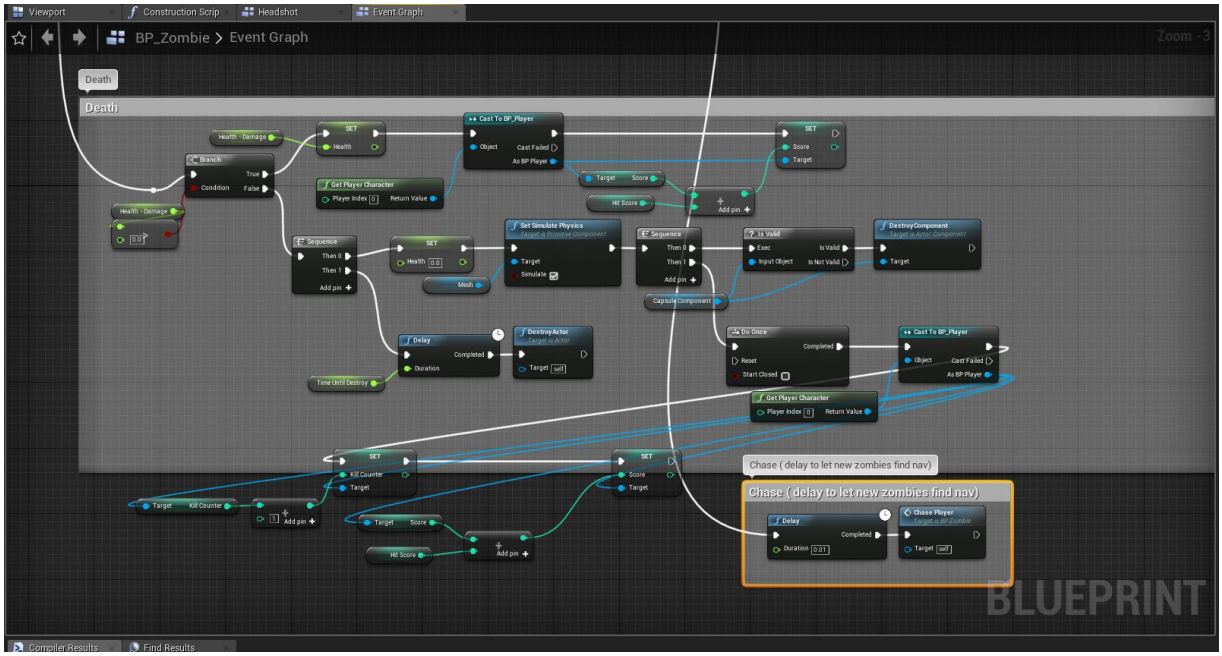



Рисунок 3.22 – Третя частина елемента “BP_Zombie”

3.2 Реалізація інтерфейсу та головного меню ігрового продукту

Для створення інтерфейсу в цьому ігровому додатку було обрано мову програмування C++ та BluePrints, а також програмне середовище для створення дво- та тривимірних ігор Unreal Engine 4.

Інтерфейс, як звичайно, складається з певних кнопок, тексту, в деяких випадках текстовими полями, певними картинками, відео. В даному додатку будуть реалізовані кнопки для переходу в налаштування гри, створення сеансу для гри, пошуку сеансів для подальшої гри, виходу з гри.

Для створення такого меню потрібно для початку створити об'єкт “Widget Bluprint”. Створюється він шляхом кліку правою кнопкою миші по робочій області програми та вибору відповідного об’єкту. Показано на рисунку 3.23.

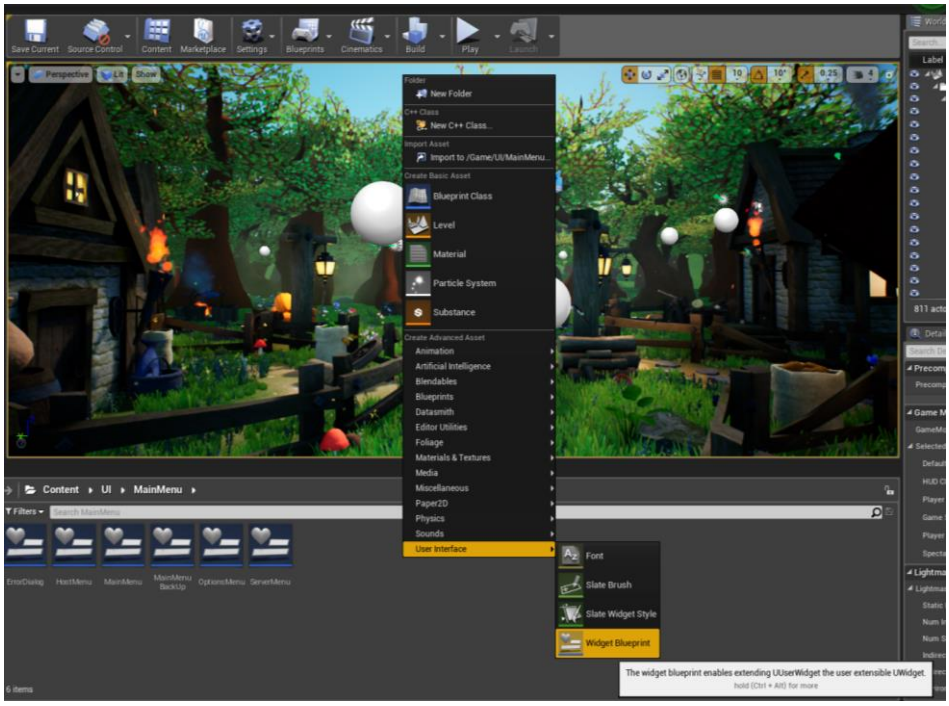



Рисунок 3.23 – Створення об’єкту “Widget Bluprint”

Після переходу на робочу область з віджетом, показано новостворений об’єкт. З лівої сторони є великий вибір об’єктів, які можна помістити на сам віджет. Для цього проєкту буде доцільно створити декілька кнопок для навігації інтерфейсом та для загальної зрозумілості продукту, показано на рисунку 3.24.

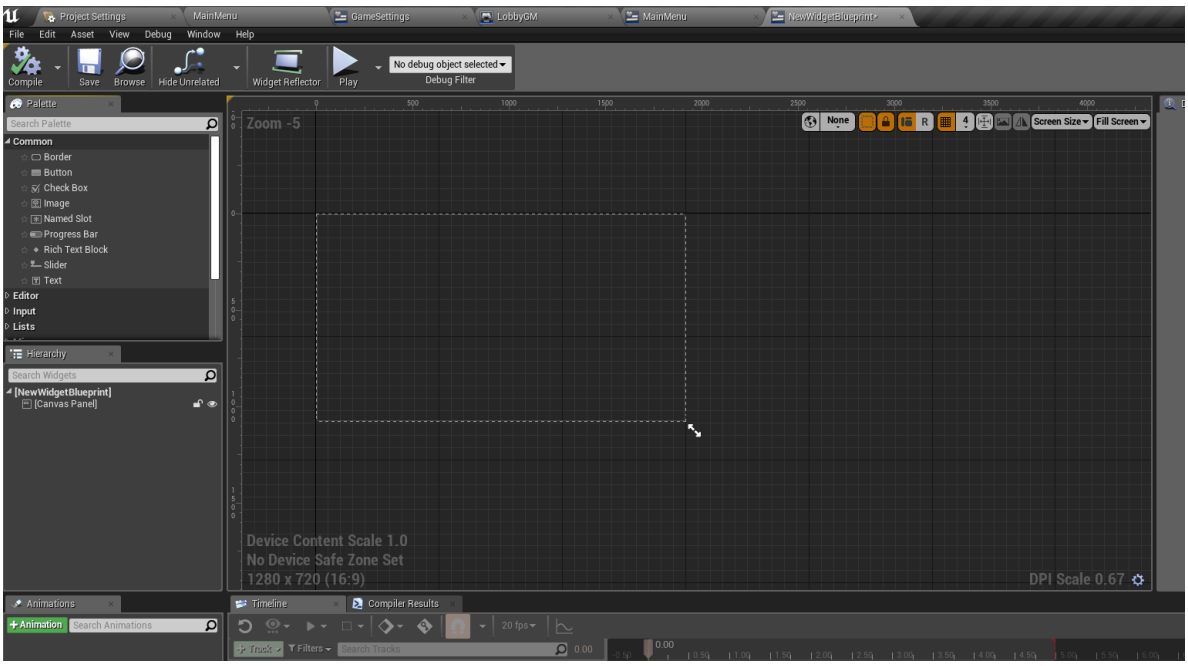



Рисунок 3.24 – Створений віджет

Створення кнопок проходить просто, потрібно перенести з лівої сторони потрібний об’єкт на сцену. Тож, буде перенесено 4 кнопки та посилання для ще одного віджету, для подальшої роботи з ним, показано на рисунку 3.25.

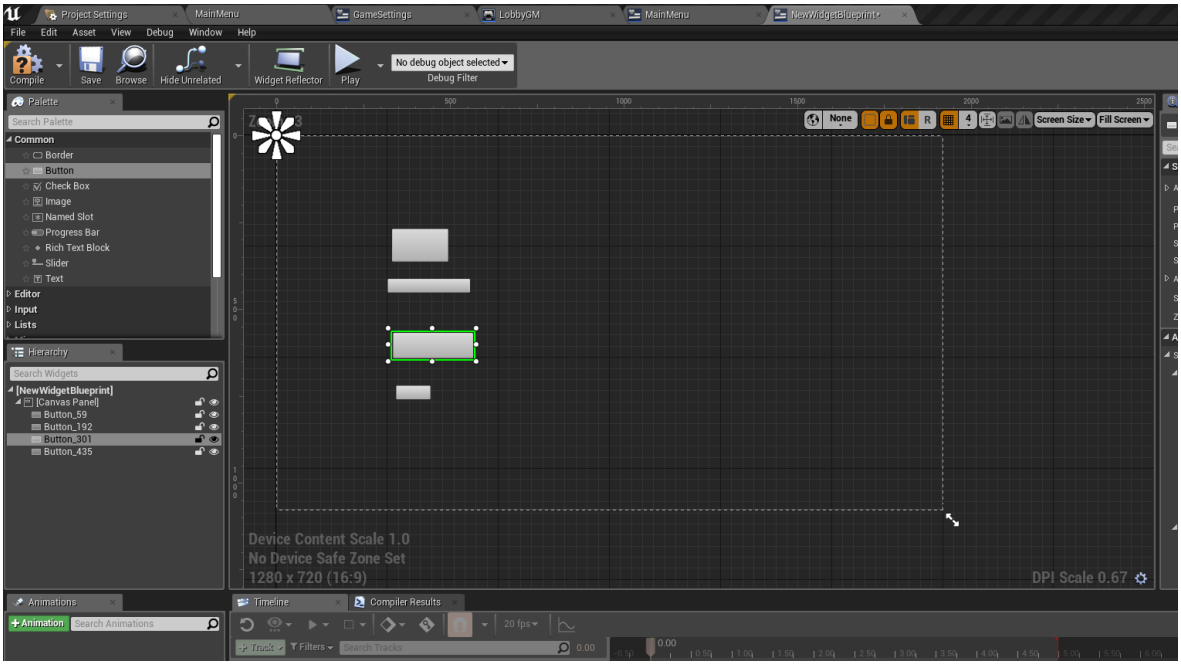



Рисунок 3.25 - Подальша конфігурація віджету

Для програмування кнопки потрібно назвати кнопку зрозумілим ім'ям, щоб не заплутатись в подальшому, потім перейти з режиму Designer в режим

Graph, саме тут буде проходити програмування кнопки мовою програмування BluePrints. Показано на рисунку 3.26.

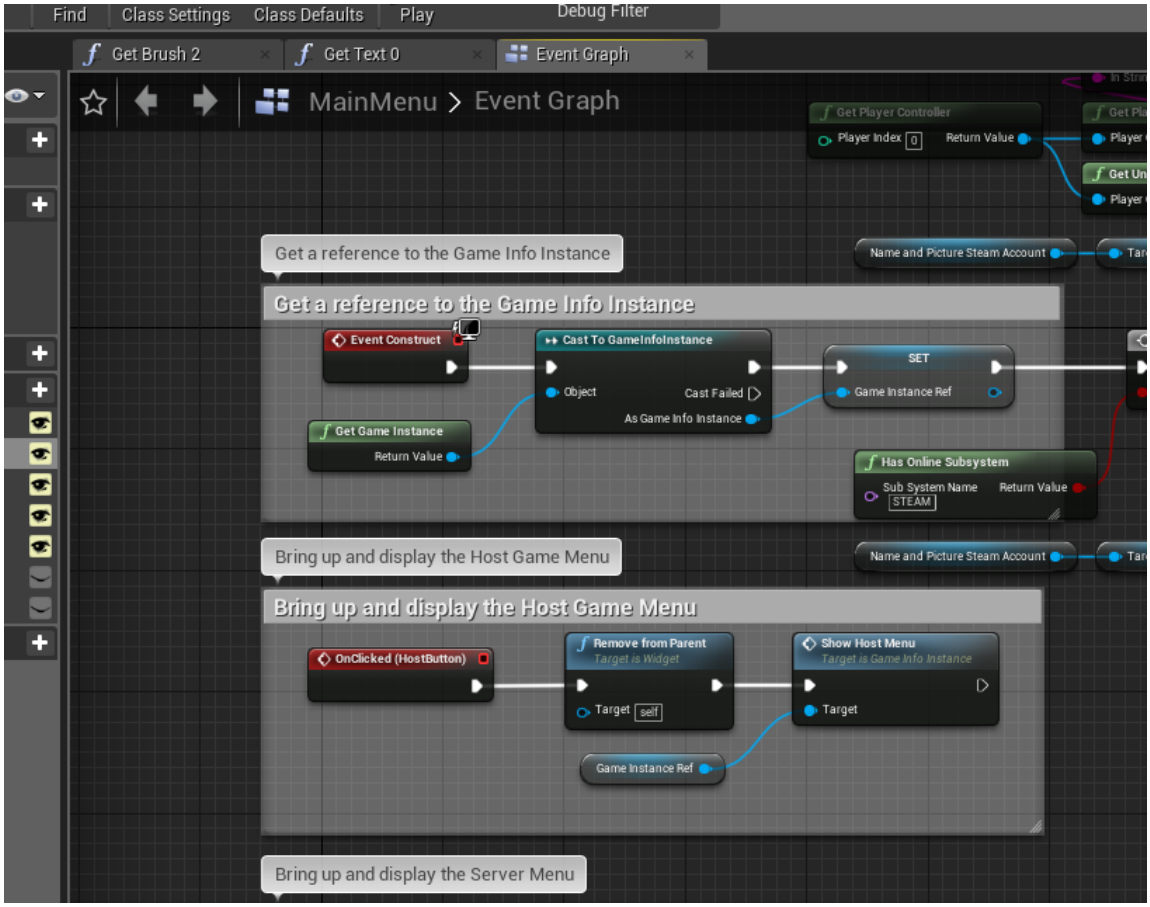



Рисунок 3.26 – Програмування кнопки

Для більш чіткого розуміння для користувача, на кнопки було додано картинки, які пояснюють, що та чи інша кнопка робить та за що відповідає. Також, як було описано раніше, на цей віджет додано ще один віджет. Це зроблено для того, щоб зменшити навантаження на процесор при завантаженні інформації з серверів Steam, та для загальної зрозумілості проєкту. Тому зараз додаток працює плавно, без помилок. Подано на рисунку 3.27.

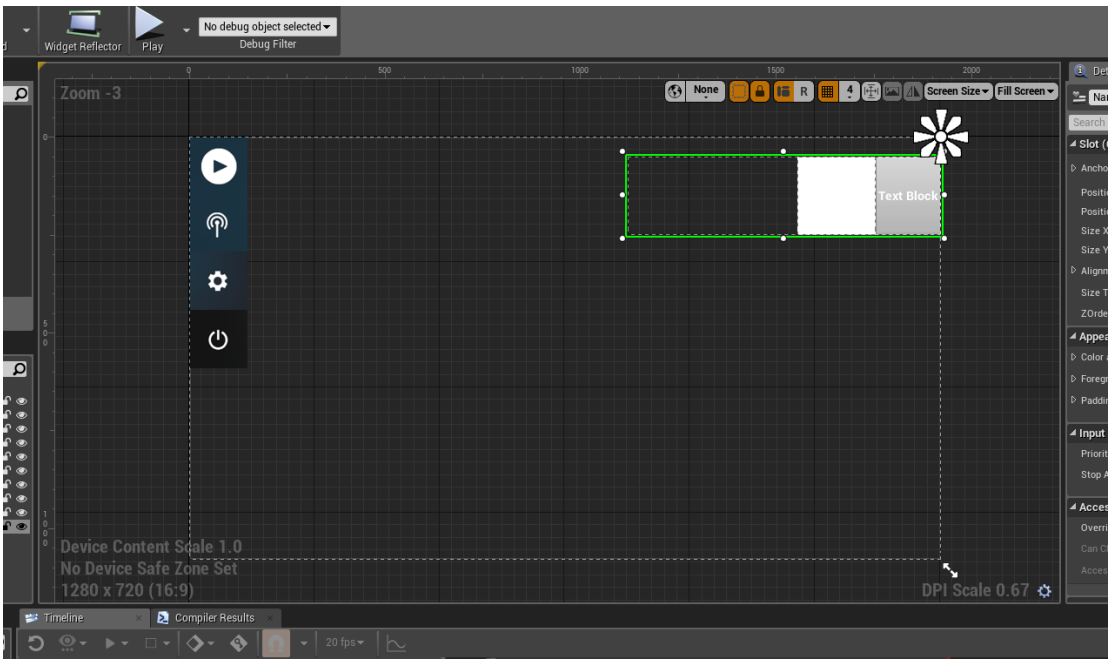



Рисунок 3.27 – Налаштування другого віджету

Скрипт віджета в віджеті складний, в ньому описана інтеграція та підключення до серверів Steam, щоб отримати інформацію з профілю Steam [7]. В цьому також задіяний плагін — Advanced Steam Sessions. Після отримання доступу віджет завантажує картинку та аватар профіля Steam. Наведена на рисунку 3.28.

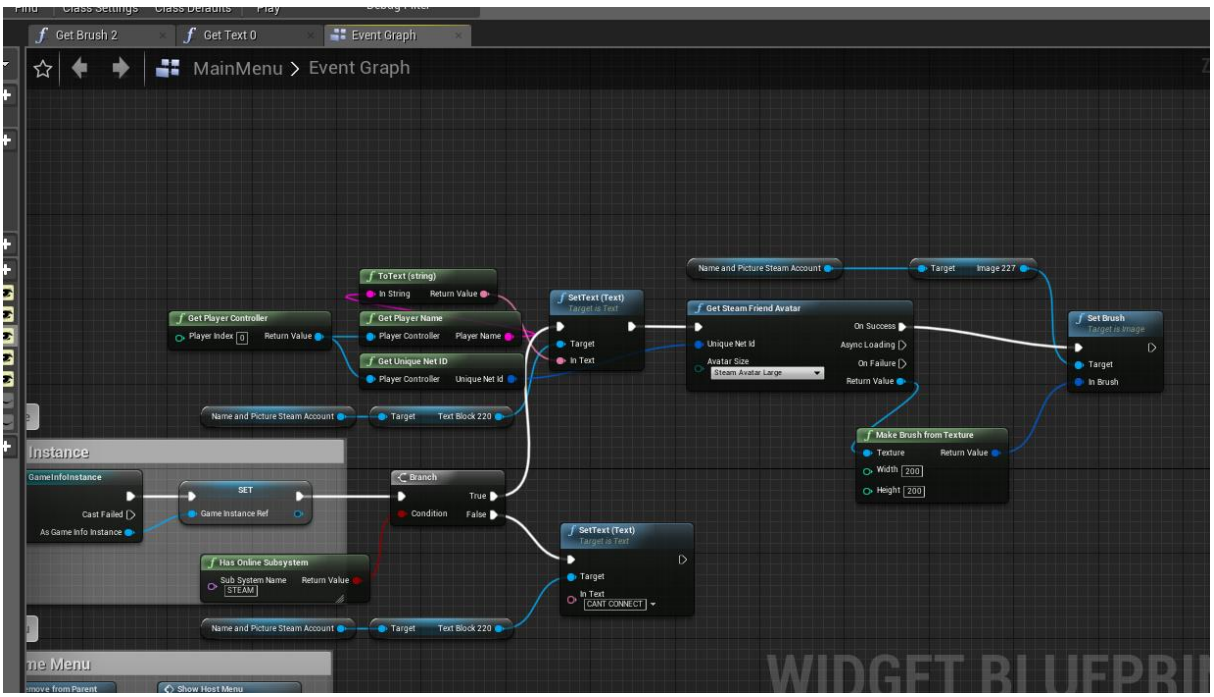



Рисунок 3.28 – Демонстрація скрипту віджета

Для створення красивого польоту камери в головному меню, при запуску гри було використано “Level Sequence”. Після надання цьому об'єкту певного імені відкривається вікно налаштувань, подано на рисунку 3.29. Тут можна створити траєкторію польоту камери, частоту кадрів (15,30,60,120, 144,240), нарізка певних моментів, віддалення або приближення камери.

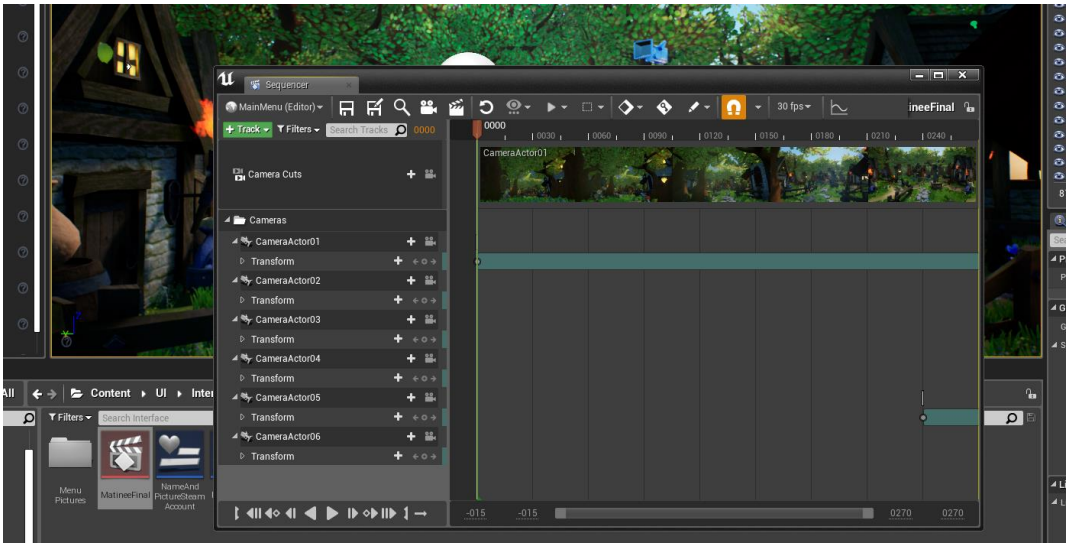



Рисунок 3.29 – Налаштування польоту камери

Створення вибору персонажу це важлива стадія розробки гри. Для цього було створено віджет “CharacterSelect”. Для розуміння вибору було імпортовано певні зображення, які визначають, кого вибере гравець. Подано на рисунку 3.30.

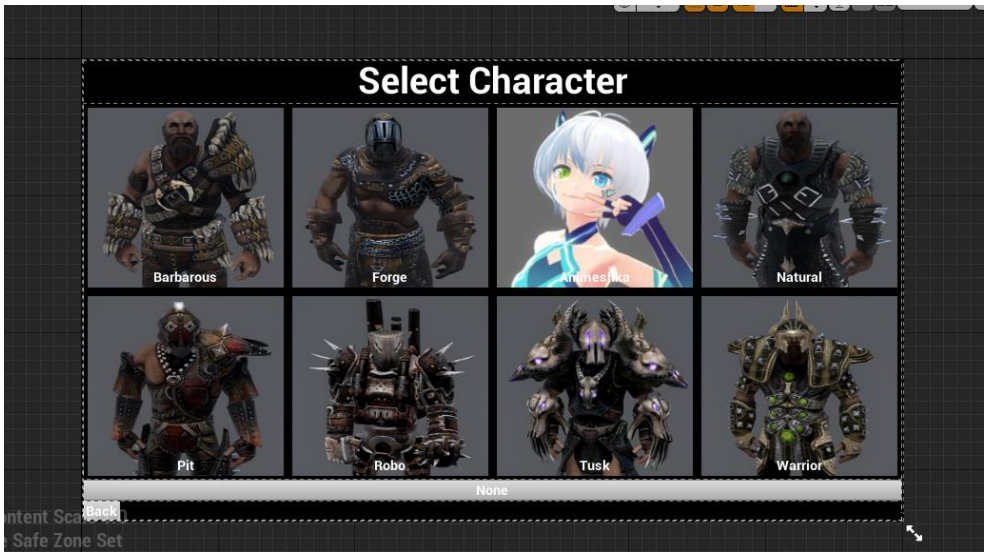



Рисунок 3.30 – Віджет вибору персонажа

Для віджету “CharacterSelect”, було написаний скрипт, який виконує дію вибору персонажа. Показано на рисунку 3.31

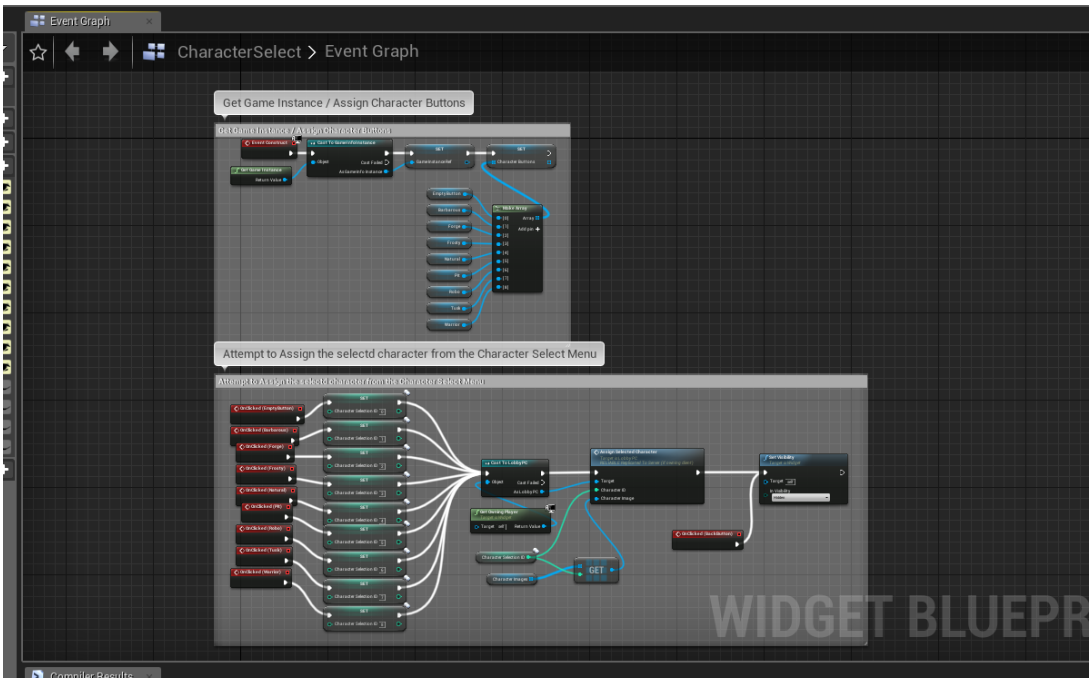



Рисунок 3.31 – Скрипт віджета вибору персонажа

3.3 Реалізація персонажів та їх рухів, рівні гри

Моделі персонажів для керування та взаємодії було створено за допомогою програми Vroid Studio. Усі моделі створені в стилі фентезі, та повинні бути привабливими для гравців. Через своєрідність такого стилю, моделі створювати не важко, через відсутність детальної обробки тіла (мускули). Для цього було обрано програму Vroid через такі переваги:

- Простий та зрозумілий інтерфейс.
- Великий вибір налаштувань усіх частин тіла.
- Експорт моделі одразу в потрібному форматі без додаткових маніпуляцій [4].

Для створення моделі в даній програмі необхідно створити об’єкт, для цього можна вибрати готовий варіант моделі, яку потім модна редагувати, або ж, створити модель з нуля, вибравши пункт “New avatar”, показано на рисунку 3.32.

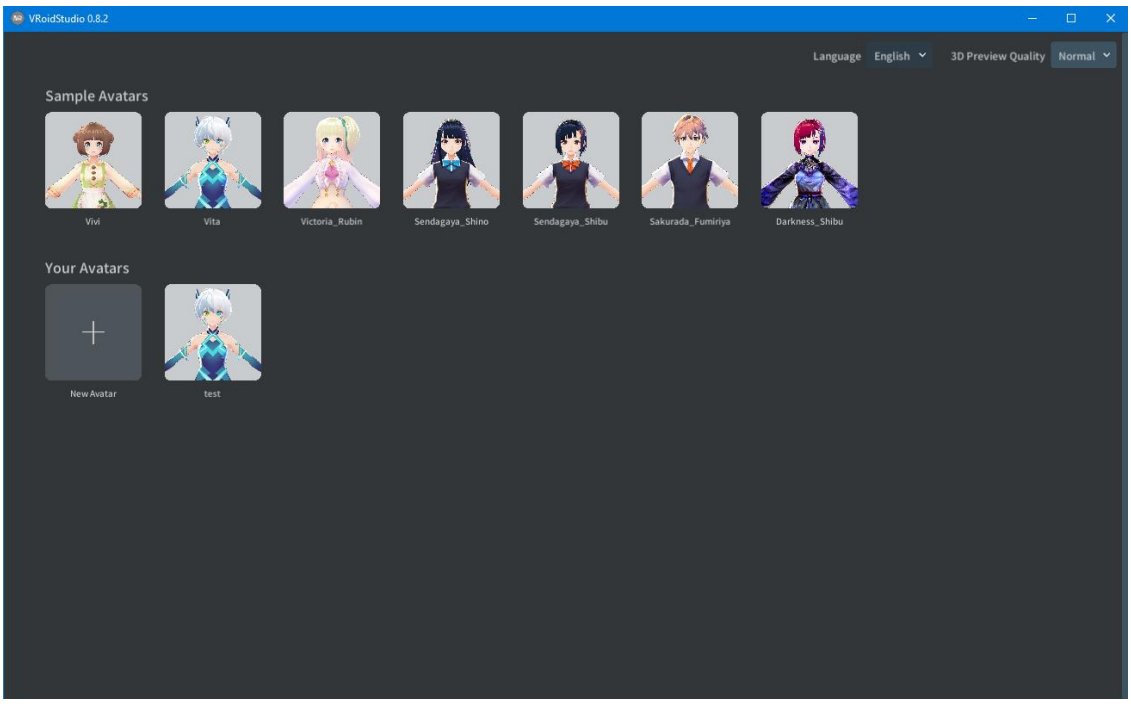
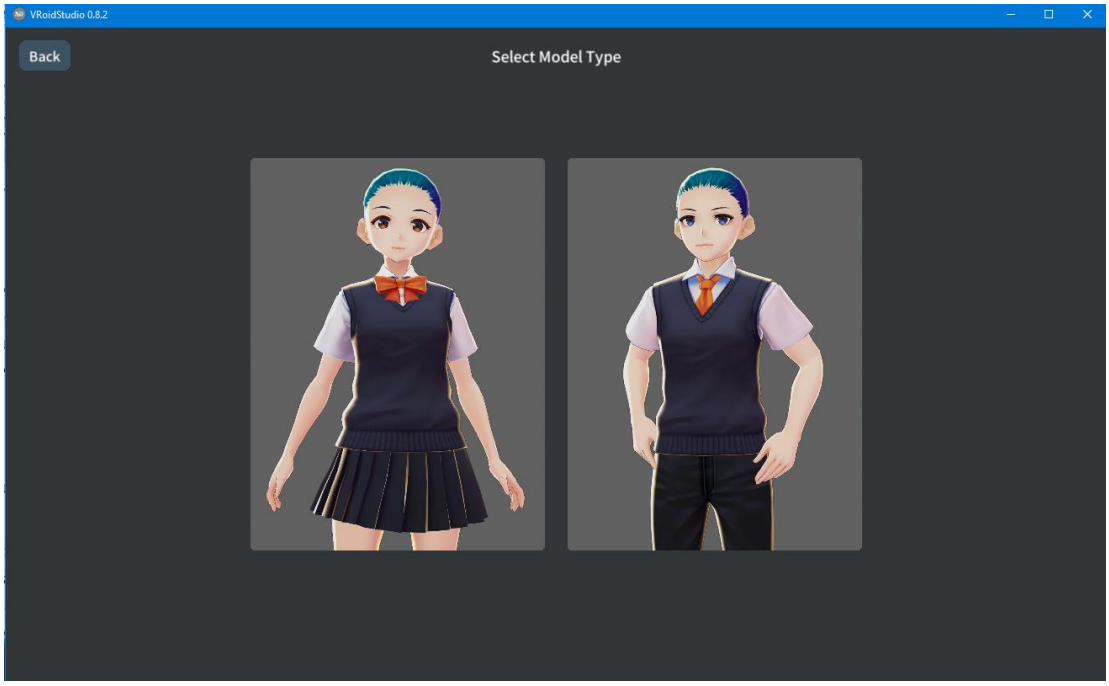



Рисунок 3.32 - Створення моделі у Vroid

Наступним кроком потрібно вибрати стать. Показано на рисунку 3.33.




Рисункок 3.33 – Вибір статі

Наступним етапом у створенні нашої моделі є редактування певних повзунків, які потребують змін. Їх можна змінювати за допомогою миші, або ж набором певного значення з клавіатури. Подано на рисунку 3.34

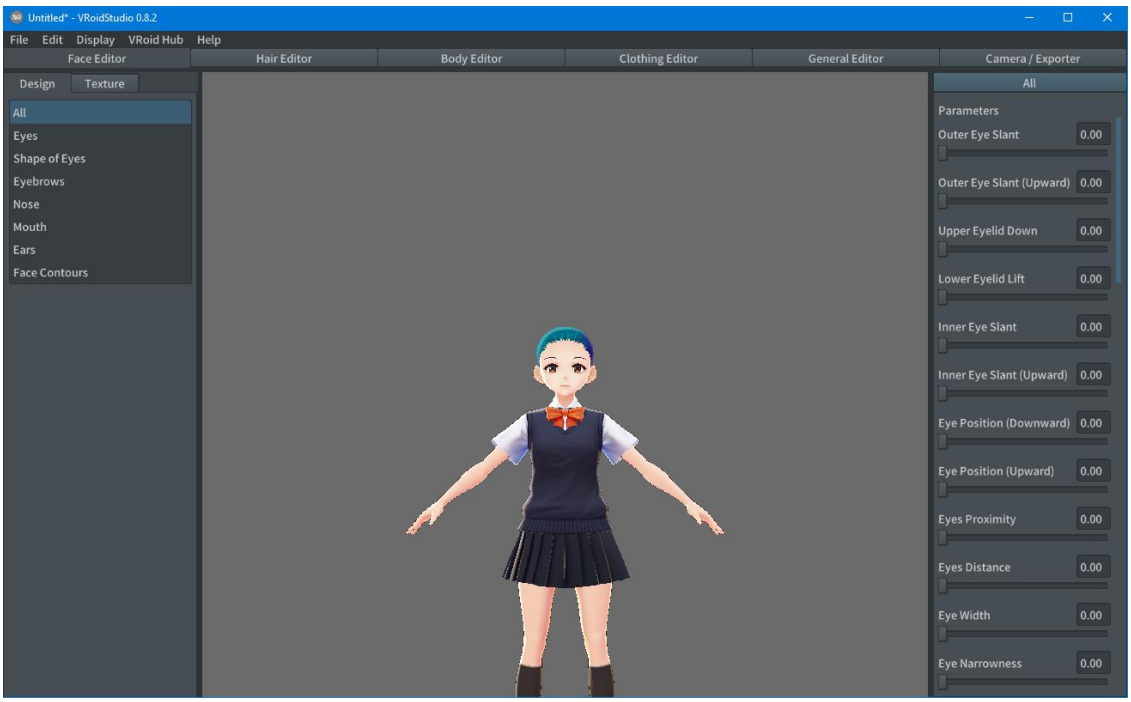



Рисунок 3.34 – Робоча область створення моделі

Для збереження результату потрібно перейти на вкладку “Camera/Exporter”, після цього натиснути на пункт Export зліва. Далі потрібно натиснути вибрати якість текстур та вибрати потрібний формат для експорту, для подальшого використовування в інших програмах Результат виконання цієї дії показано на рисунку 3.35.

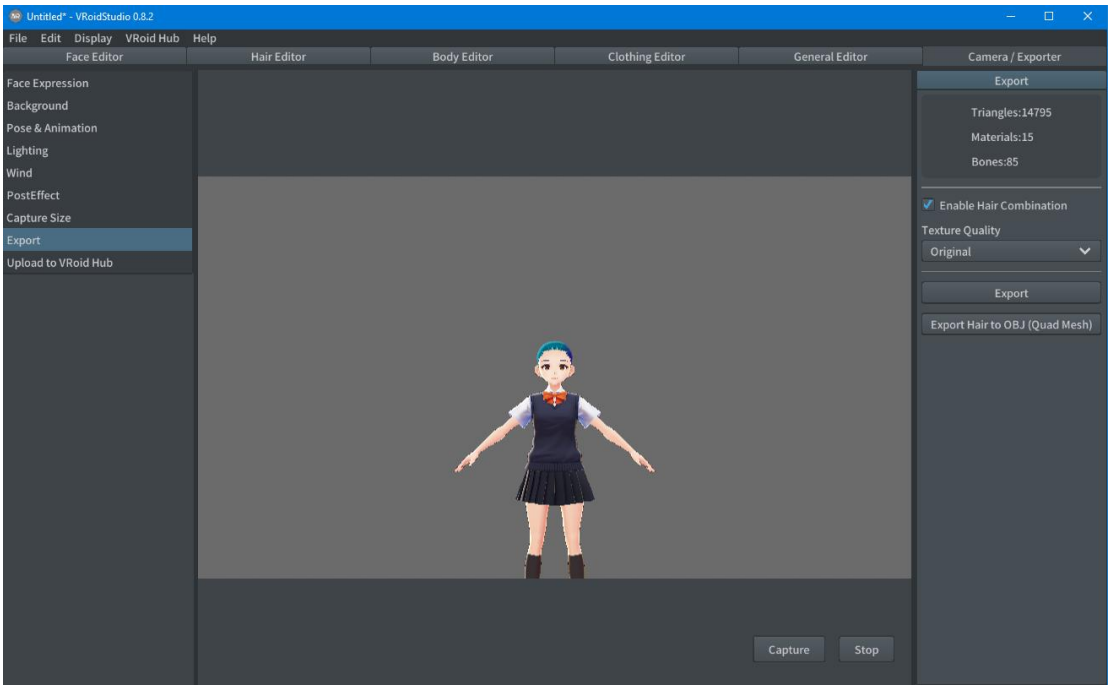



Рисунок 3.35 – Збереження моделі

Накладання анімації на модель з цієї програми дещо складний, тому оптимальний спосіб це використання спеціального плагіну для Unreal Engine 4. В цьому допоможе адаптований плагін VRM4U, створений спеціально для підтримки моделей з Vroid Studio [4].

Для початку потрібно зробити так, щоб модель була здатна до прийняття в себе певної анімації. Потрібно перетворити позу A-Pose у позу T-Pose. Це можна зробити через функцію “Set up Rig”. Ця функція автоматично підхопить “кістки” нашої моделі та перетворить модель у T-Pose. Показано на рисунку 3.14

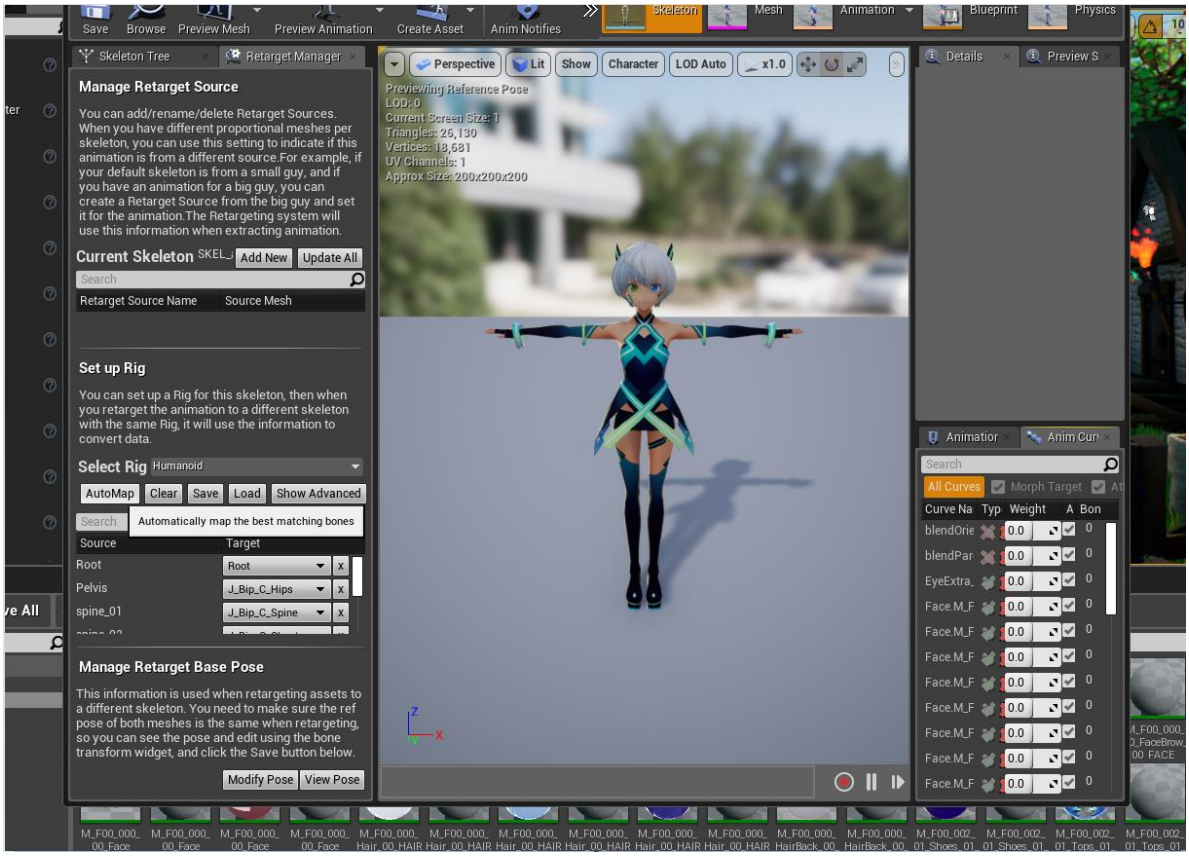



Рисунок 3.36 – Результат перетворення пози

Також плагін допомагає після експорту моделі з Vroid скласти усі файли докупи. Це потрібно тому, що в папці з моделлю знаходиться більше 1000 текстур, а це би зайняло багато часу в простого користувача для з'єднання їх воєдино. Подано на рисунку 3.36

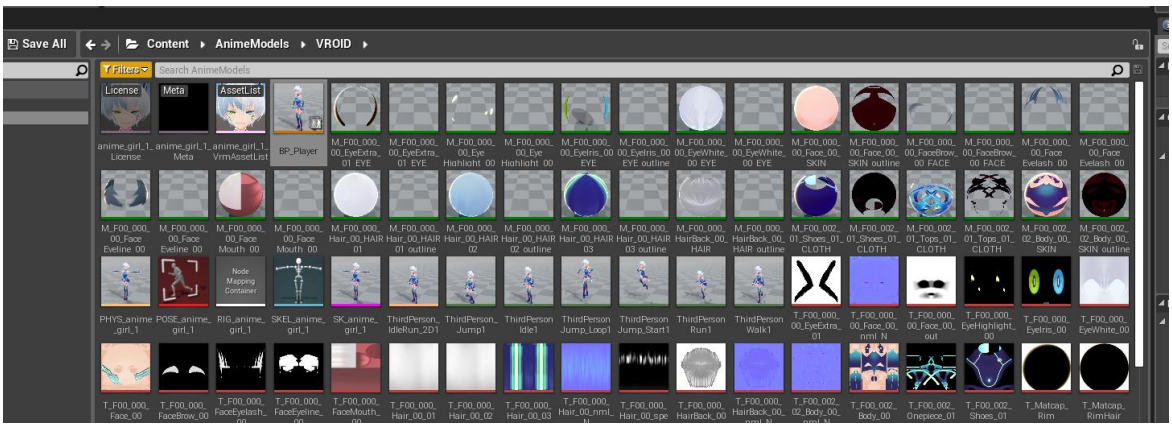



Рисунок 3.36 – Результат складання текстур моделі

Після всіх маніпуляцій можна використовувати стандартні анімації з Unreal Engine 4 персонажа або ж використувати свої власні анімації [5]. Подано на рисунку 3.37

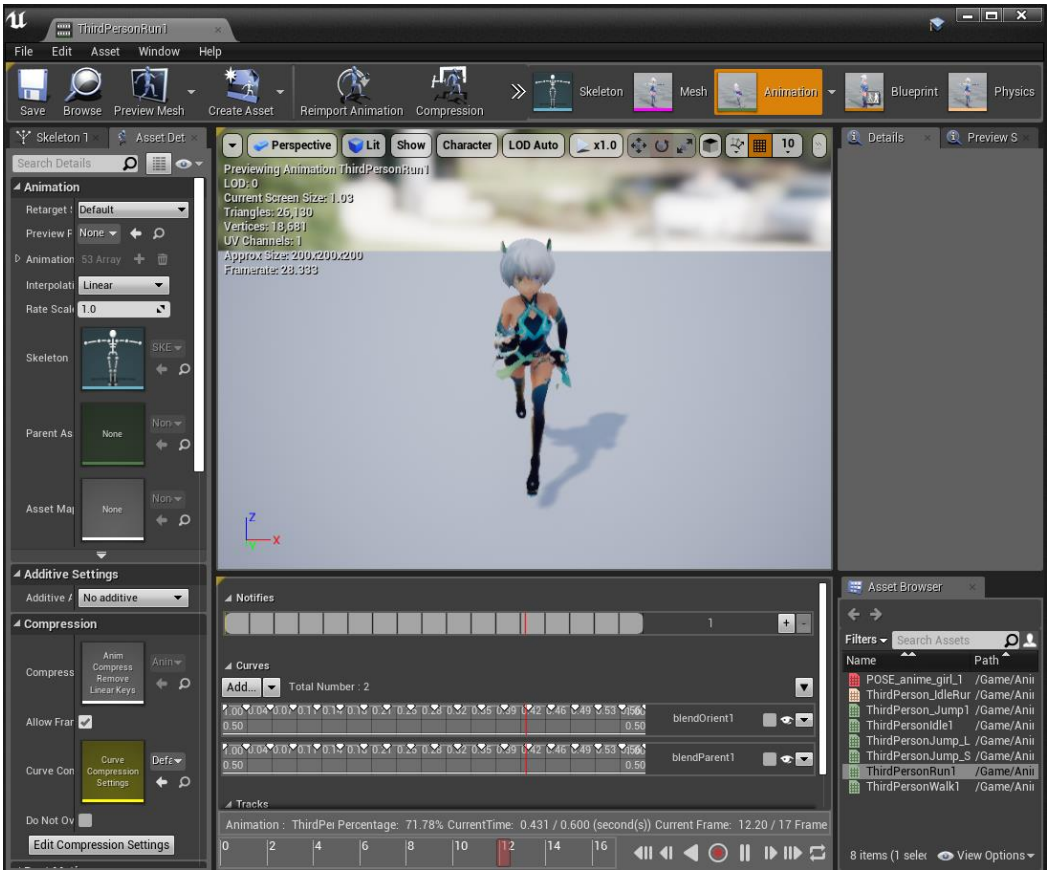



Рисунок 3.37 – Результат накладання анімації на модель

Реалізація рівнів та локацій в ігровому процесі.

Для створення локації потрібно створити спеціальний елемент “Level”.

Через меню редактора цього елемента можна створити власну територію, на якій будуть проводитись подальші дії. Подано на рисунку 3.38.

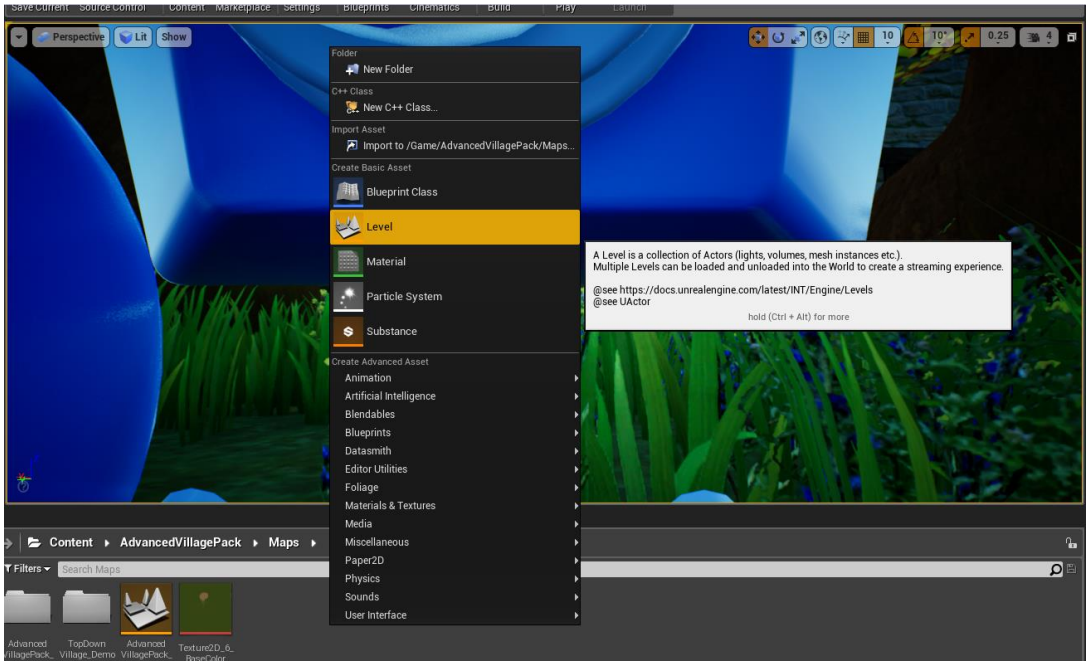



Рисунок 3.38 – Результат створення елементу “Level”

З допомогою спеціальних кистей створювалися:

- рівнина;
- просте озеро;
- заповнення карти (каміння, дерева, трава);

Приклад створення місцевості в рушії Unreal Engine 4 наведено нижче. Рисунок 3.39.

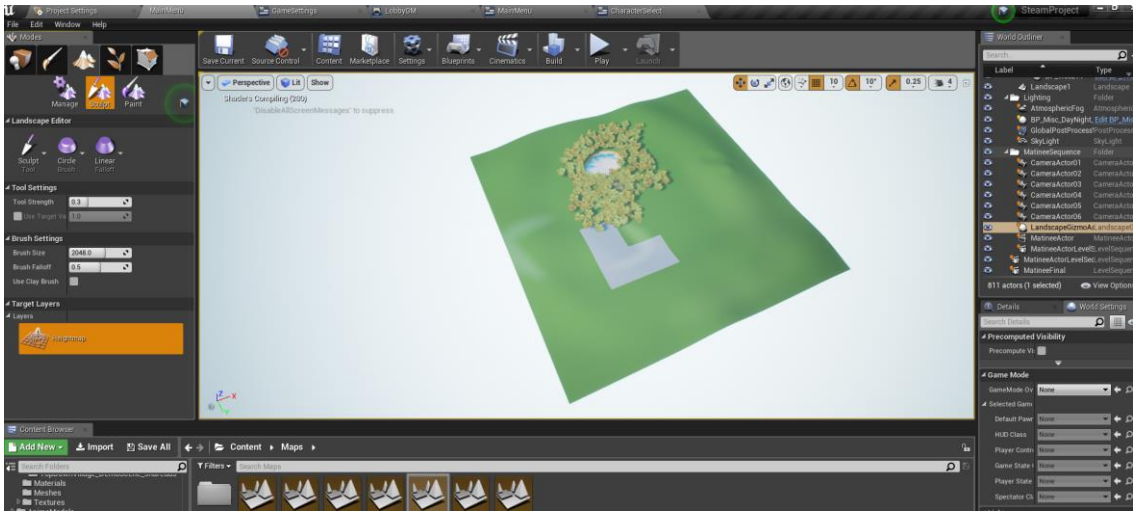


Рисунок 3.39 - Створення початкової локації

Щоб надати макету локації вигляду зеленини, потрібно надати макету певну вибрану текстуру. Для цього потрібно вибрати макет, з правого боку на панелі вибрати вкладку “Details” та знайти пункт “Landscape Material”. Текстура може бути дефолтною, або ж імпортованою самим користувачем. Важливо, щоб текстура мала сумісність з цим середовищем розробки, через нехтування цим з’являються недоліки. Подано на рисунку 3.40

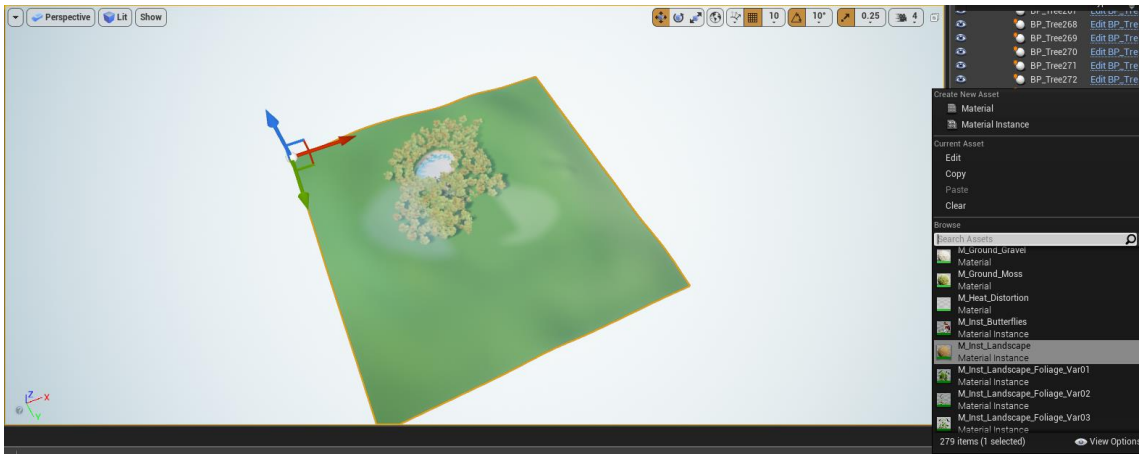



Рисунок 3.40 – Накладення текстури на макет

Для заповнення місцевості було вибрано багато об’єктів, але зупинимось поки що на деревах. Ними можна наповнювати карту за допомогою кисті, але це буде дуже замітно та не професійно. В моєму випадку, заповнення велося мануально, тобто вручну. Показано на рисунку 3.41

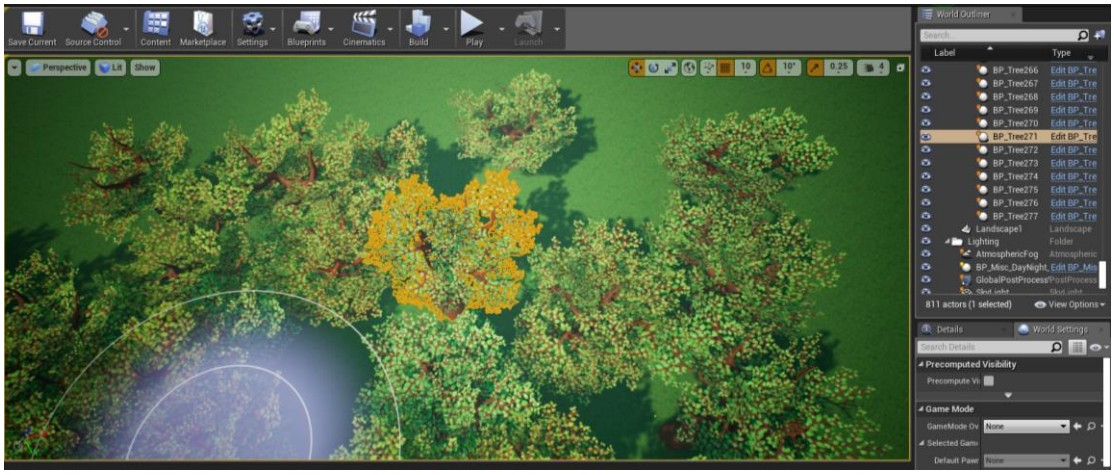



Рисунок 3.41 – Заповнення локації деревами

Для створення динамічного світла і лампах та будинках, було написано певний BluePrints код. Тобто усі лампи та будинки будуть загорятися в вечірню та нічну пори. Подано на рисунку 3.42


Рисунок 3.42 – Загоряння лампи

Будинки та їх текстури були імпортовані з програми Blender, до них також написаний код [3].Подано на рисунку 3.44

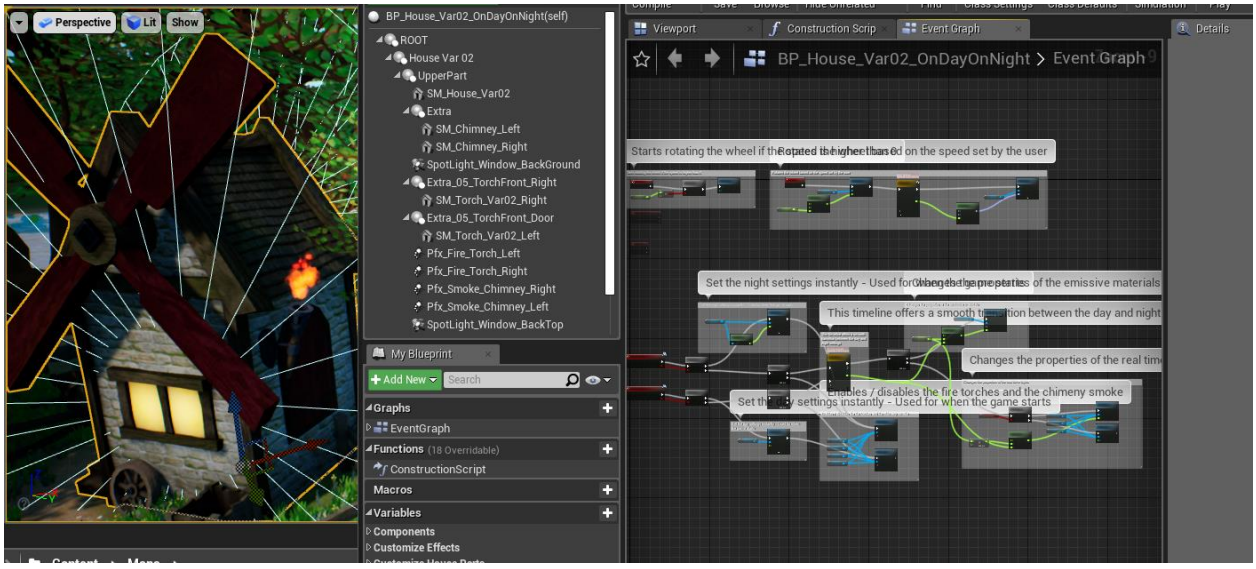



Рисунок 3.44 – Будинок та його код

Метелики на цій мапі також присутні. Вони потрібні для підтримання атмосфери. Вони мають в собі код для анімації та циклічної зміни траекторії польоту. Показано на рисунку 3.45.

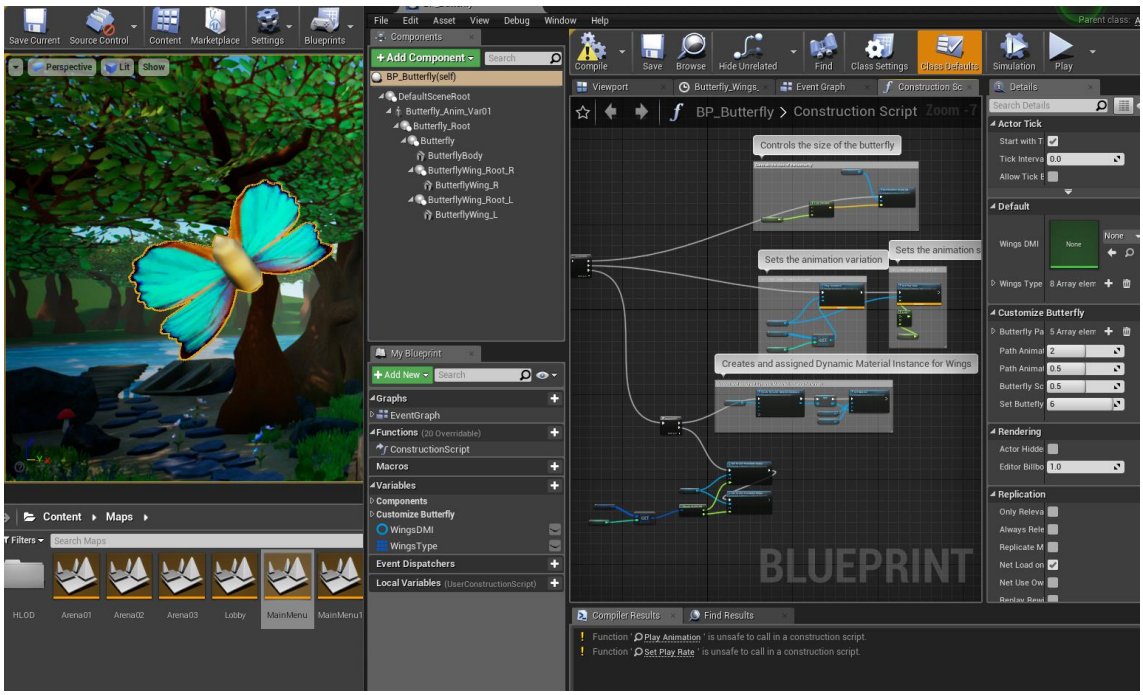



Рисунок 3.45 – Метелик та його код

Загалом, для реалізації гри було створено більше 200 скриптів та функцій, та частину програмного коду на мові програмування C++, який подано в додатку Б. Для підтримки мережевого коду було використано плагін Steam Advanced Sessions, для підтримки імпорту моделей з програми Vroid використано плагін VRM4U.

# 4 ТЕСТУВАННЯ

Тестування певної програми або ж системи — це спосіб перевірки, який включає в собі набори спеціальних тестів (позитивних, негативних) з відомими наперед результатами. Це потрібно для того, щоб забезпечити якість програмного продукту, для комфортної взаємодії користувача та системи в цілому.

Простіше кажучи, це оцінка якості продукту. Його мета - виявити помилки, які можуть бути критичними.

Для тестування можна побудувати такий план:

− проходження смоук тесту (Smoke test), потрібен для того, щоб поверхово, без знання внутрішньої будови продукту або ж коду, провести тест;

− контроль результатів тестів на кожному етапі життєвого циклу;

− створення певних тест-кейсів та лістингів для перевірки окремих елементів продукту, на перше місце виноситься найкритичніші точки продукту;

− виконання тестів та аналіз результатів тестування;

− якщо відбулась зміна документації, або ж глобальна зміна інтерфейсу програмного продукту - повторне тестування.

Проведемо тестування ігрового програмного продукту. Найперше потрібно клікнути двічі по файлу запуску гри. При запуску гри користувач в першу чергу бачить стартове меню гри.

На стартовому екрані реалізовано 5 інтерактивних кнопок, при натисканні на кнопку «New Game» користувач переноситься на стартову локацію гри яка представлена парком в передмісті, меню гри показано на рисунку 4.1.


Рисунок 4.1 – Головне меню гри

Кнопка “Play” відкриває інший віджет, в якому можна вписати назву сервера, на якому будуть відбуватися дії, кількість гравців та метод гри (LAN або Internet). Подано на рисунку 4.2.

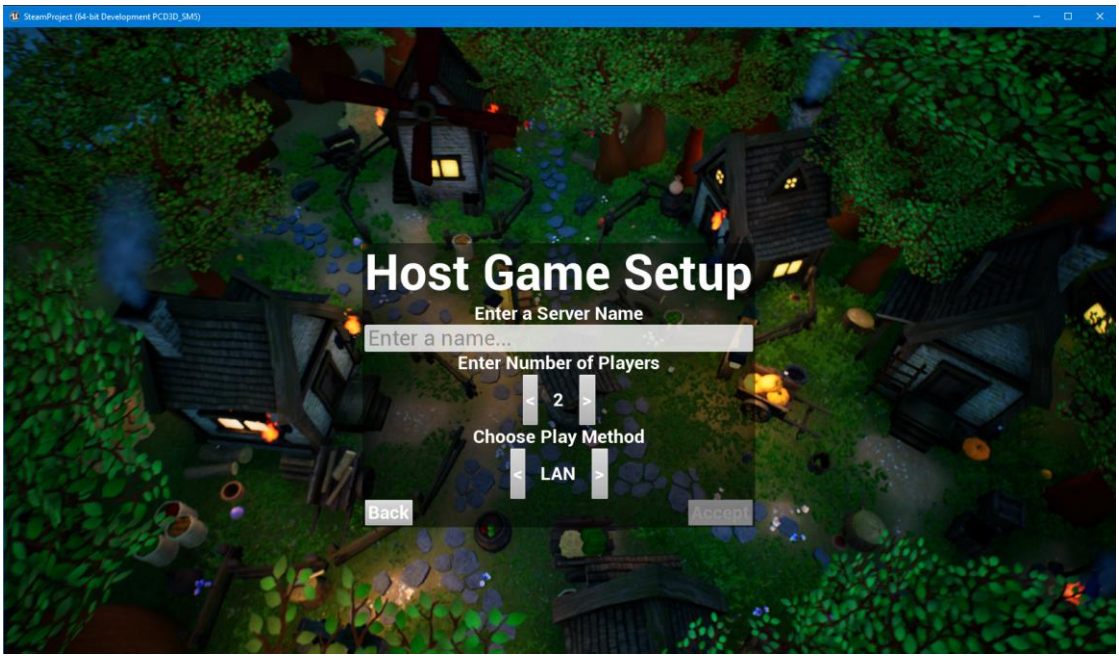



Рисунок 4.2 – Host Game Setup

Після вибору усіх даних користувача переносить на іншу локацію з іншим віджетом, для подальшого маніпулювання лоббі. Тут можна надати певний час раунду, вибрати свого персонажа, почати раунд, покинути лоббі.

Також тут присутня функція чатування. З правого боку показано усіх учасників лоббі, присутня функція вигнання користувача з лоббі. Подано на рисунку 4.3.

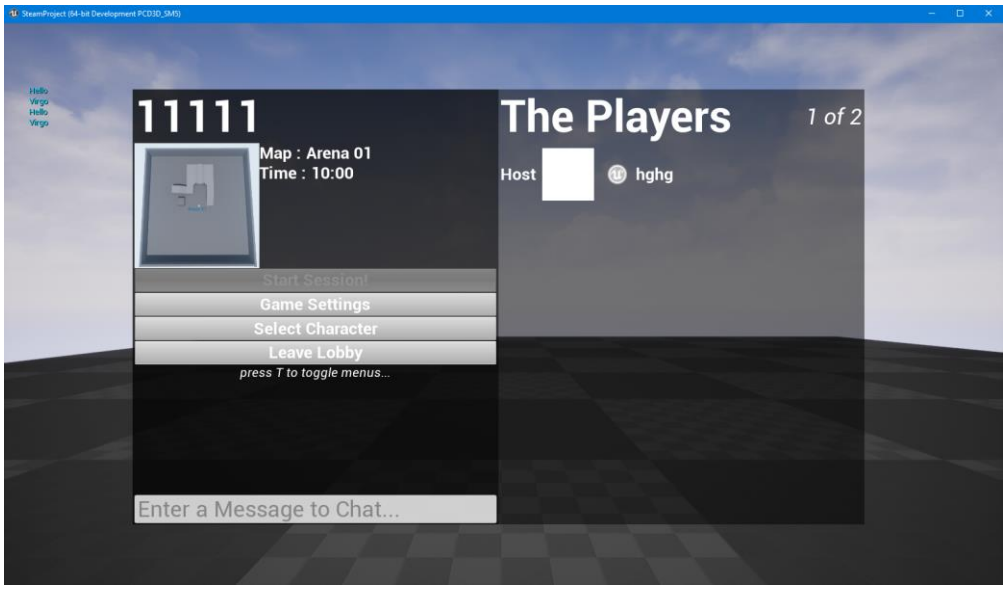



Рисунок 4.3 – Лоббі налаштувань раунду

На рисунку 4.4 показано віджет налаштувань раунду, а саме: часу, мапи, та вигнання користувача.


Рисунок 4.4 – Детальніше налаштування раунду

Для вибору персонажа потрібно натиснути кнопку “Select Character”, та вибрати з нижче поданих картинок певного персонажа. Подано на рисунку 4.5

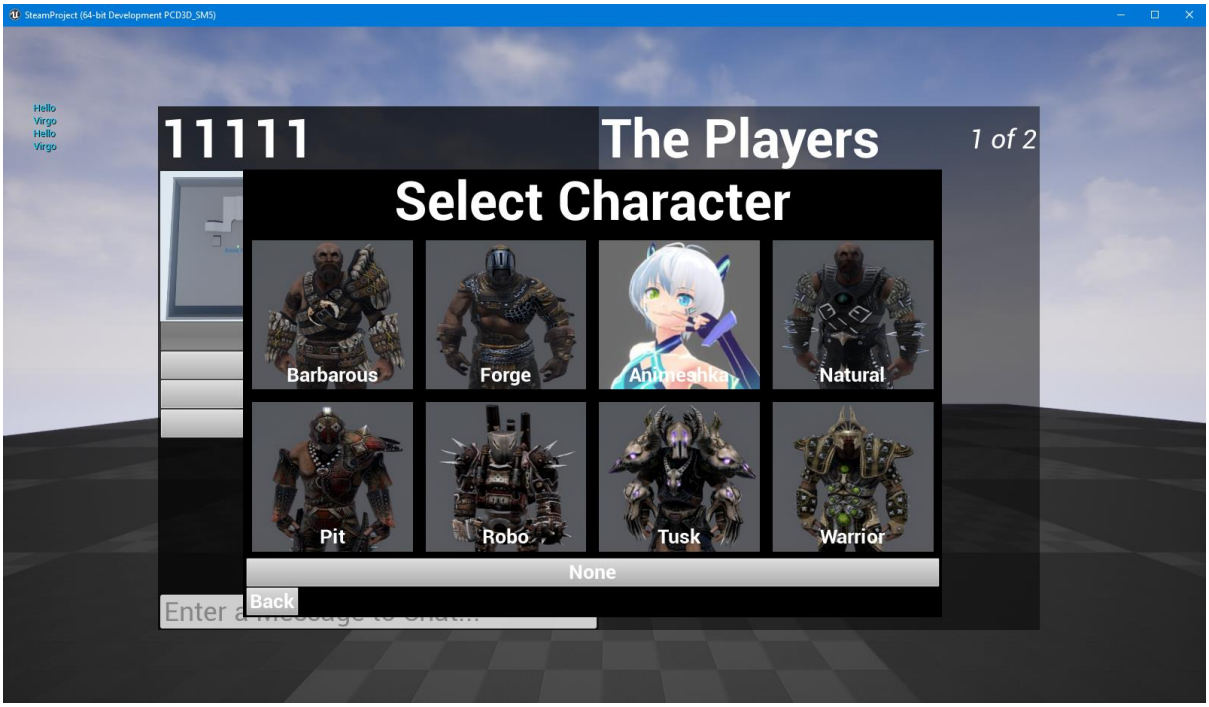



Рисунок 4.5 – Детальніше налаштування раунду

Після усіх маніпуляцій потрібно натиснути “Start Session” і гра перенесе гравця на ігрове поле. Подано на рисунку 4.6


Рисунок 4.6 – Ігрове поле

Під час гри можна посилати повідомлення у чат. На рисунку 4.7 показано його роботу.


Рисунок 4.7 – Робота чату

Зміна гри від третього лиця до першого відбувається через клік на колесо миші. Показано на рисунку 4.8

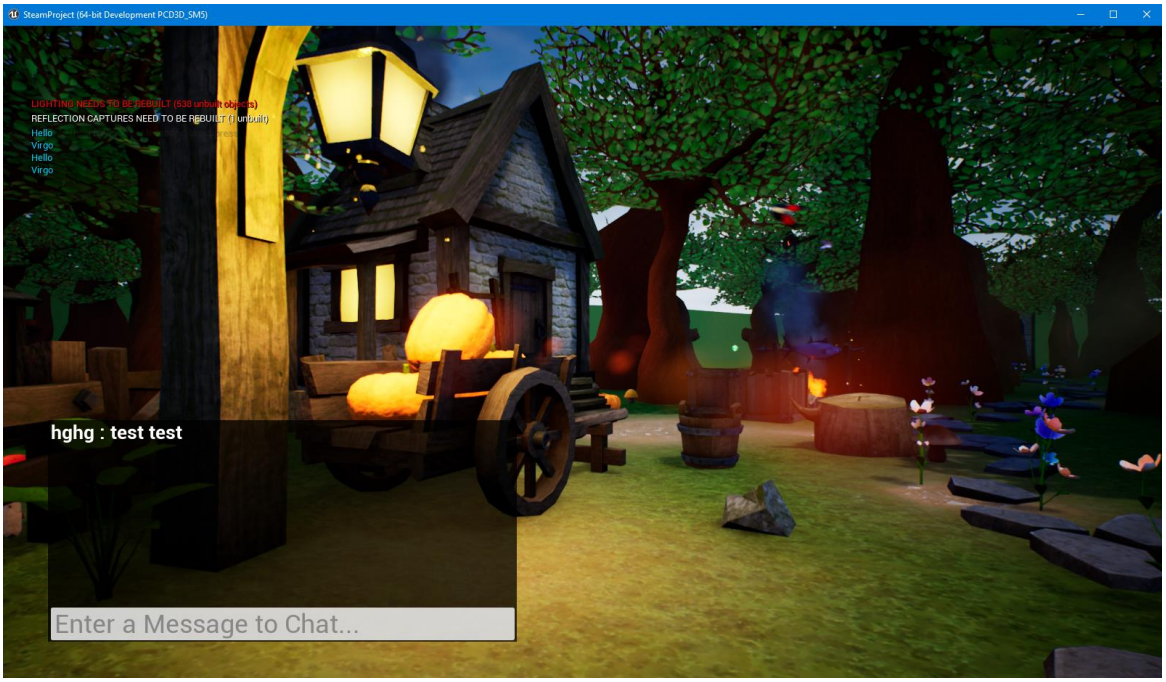



Рисунок 4.8 – Зміна вигляду гри

Приближення камери або ж прицілювання персонажа через натиснення правої кнопки миші. Подано на рисунку 4.10


Рисунок 4.10 – Прицілювання персонажем

Через потужню складову рушія Unreal Engine 4, йому потрібно робити багато обрахувань, які зв’язані з плануванням місцевості, підтримкою багатьох текстур, шейдерів, моделей. Мінімальні та рекомендовані характеристики комп’ютера подані в додатку В.

Отже, під час тестування гри помилок не виявлено, всі функції працюють належним чином, персонажі рухаються, як і було запрограмовано, ігровий продукт повністю відповідає поставленим вимогам.

# 5 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Створення якісної продукції передбачає проведення техніко-економічних розрахунків, з метою визначення оптимальних шляхів вирішення технічних питань під час допроєктної підготовки, проєктування, впровадження, експлуатації, обслуговування, ремонту та утилізації електротехнічного обладнання. Тому невід'ємною частиною є формування знань і набуття практичних навичок в вирішенні таких питань: техніко-економічне обґрунтування даного проєктного рішення; розрахунок витрат на проєктування.

## 5.1 Аналіз ринку

Створення комп'ютерної гри це відповідальна справа, яка потребує великих затрат часу. Гру потрібно постійно підтримувати в належному стані, враховуючи усі методи та технології створення інших ігор.

Ця гра це компіляція, збірка вже існуючих ігор на даний момент, неможливо створити щось нове. Саме це і характеризує цей проєкт - модернізація чогось старого, або ж його редизайн це те, що може залучити нових гравців.

Потенційним замовником цього продукту вважається простий геймер. Для гра підійде для тих, хто не хоче грати тільки в ігри, які спрямовані на повний реалізм, та залежить тільки від одної смерті, що для великої частини гравців є проблемою.

Потребою гравців від цієї гри буде хороший геймплей, який може затягнути, велика частота кадрів для комфортної гри, підтримка старих платформ. Це не дуже легко зробити в цю епоху, кожного тижня виходять анонси про нові комплектуючі для ПК та про нові смартфони.

Конкурентами конкретно цьому проєкту складають великі компанії, такі як: Valve — CS:GO, Blizzard — Overwatch та багато інших, які розраховані на

трішки іншу аудиторію в силу свого стилю, графіку, методом маніпулювання героєм і т.д.

5.2 Розрахунок витрат на проєктування

Розмір заробітної плати залежить від складності та умов виконуваної роботи, кількості виконавців, професійно-ділових якостей працівника, результатів його праці.

Витрати на розробку гри включають:

- заробітну плату розробників;
- відрахування у спеціальні державні фонди;
- накладні витрати;
- інші витрати;
- витрати на використання комп'ютерної техніки.
- витрати на використання ліцензованих програм для створення гри, моделей, анімацій (таблиця 5.1).

Таблиця 5.1 – Кошторис витрат на проєктування

| Найменування статей витрат | Сума, грн | Обґрунтування |
|---|---|---|
| 1 Зарплата проєктувальників. | 376740 з/п, грн. | Нарахована зарплата |
| 2. Відрахування на соціальні потреби. | 82882 грн. | Ставка ЄСВ 22 % |
| 3. Контрагентські роботи і послуги. | 56505 грн. | 15% |
| 4. Витрати на відрядження. | 7000 грн. | Проживання трьох працівників + витрати на проїзд |
| 5. Інші прямі витрати. | 154700 грн. | Складають 45% від заробітньої плати |

| | | |
|---|---|---|
| 6. Усього прямих витрат. | 677827 грн. | Підсумкова вартість пунктів 1-5 |
| 7. Накладні витрати. | 24720 грн. | Складають 45% від видатків заробітньої плати |
| 8. Планові накопичення. | 175636 грн. | Складають 25% від суми прямих і накладних витрат |
| 9.Усього, кошторисна вартість проєкту. | 878183 грн. | Сума прямих і накладних витрат та планових накопичень |
| 10. Податок на додану вартість. | 175636 грн. | Визначено по діючому нормативу від кошторисної вартості проєкту - 20% |
| 11. Загалом, договірна ціна розробки Зп. | 1053819 грн. | Сума кошторисної вартості роботи та податку на додану вартість і визначає ті витрати на проєктування |

Зарплата проєктувальників.

Заробітна плата працівників визначається виходячи з кількості виконавців, діючих посадових окладів та кількості місяців їх участі в розробці проєкту.

Отже, до розробки даного проєктного рішення були залучені такі спеціалісти, як: C++ Developer, Animation Lead, BluePrint Dev, Unreal Engine Lead.

Розрахунок зарплати показано нижче по формі таблиці 5.2.

Таблиця 5.2 – Розрахунок заробітної плати проєктувальників

| N | Посада | Оклад, | Відрахування | Кількість | | Сума |
|---|---|---|---|---|---|---|
| 1 | C++ Developer | 12000 | 2340 грн/міс | 1 чол. | 12 місяців | 115920 з/п, грн. |
| 2 | Animation Lead | 15000 | 2925 грн/міс | 1 чол. | 12 місяців | 144900 з/п, грн. |
| 3 | BluePrint Dev, Unreal Engine Lead | 12000 | 2340 грн/міс | 1 чол. | 12 місяців | 115920 з/п, грн. |
| | | Усього зарплати: | | | | 376740 з/п, грн. |

Оклад

Посадові оклади визначаються множенням ставки першого розряду (2102 грн.) на тарифний коефіцієнт потрібної посади з округленням до цілого значення (таблиця 5.3).

Таблиця 5.3 – Посадові оклади

| ПОСАДА | Розряд | Ктар |
|---|---|---|
| C++ Developer | 15 | 2,58 |
| Animation Lead | 14 | 2,42 |
| BluePrint Dev, Unreal Engine Lead | 12 | 2,12 |

Відрахуваня

Витрати підприємства на соціальні заходи виникають внаслідок нарахувань сум в залежності від розміру фактичних витрат на оплату праці працівників, які включають витрати на основну і додаткову заробітну плату та здійснення інших видів заохочень та виплат.

Обов’язкові відрахування здійснюються на:

- податок на доходи фізичних осіб;
- військовий збір;
- єдиний внесок;

C++ Developer:

- Рахуємо податок на доходи фізичних осіб: 12000 Х 18% (ставка податку на доходи фізичних осіб) = 20865 грн.
- Рахуємо військовий збір: 12000 Х 1,5% (ставка військового збору) = 1738 грн.
- Рахуємо єдиний внесок: 12000 Х 22% (ставка ЄСВ) = 25502 грн.
- Утримання становлять – 22603 грн. (20865 грн. + 1738 грн.).
- До виплати працівникові – 115920 грн. (12*(12000 грн. – 2340 грн.)).

Animation Lead:

- Рахуємо податок на доходи фізичних осіб: 144900 Х 18% (ставка податку на доходи фізичних осіб) = 26082 грн.

– Рахуємо військовий збір: 144900 Х 1,5% (ставка військового збору) = 2173 грн.

– Рахуємо єдиний внесок: 144900 Х 22% (ставка ЄСВ) = 31878 грн.

– Утримання становлять – 2925 грн. (26082 грн. + 2173 грн.).

– До виплати працівникові – 144900 грн. (12*(15000 грн. – 2925 грн.)).

BluePrint Dev, Unreal Engine Lead:

– Рахуємо податок на доходи фізичних осіб: 115920 Х 18% (ставка податку на доходи фізичних осіб) = 20865 грн.

– Рахуємо військовий збір: 12000 Х 1,5% (ставка військового збору) = 1738 грн.

– Рахуємо єдиний внесок: 12000 Х 22% (ставка ЄСВ) = 25502 грн.

– Утримання становлять – 22603 грн. (20865 грн. + 1738 грн.).

– До виплати працівникові – 115920 грн. (12*(12000 грн. – 2340 грн.)).

## 5.3 Обґрунтування необхідності розробки

Цей проєкт розрахований на аудиторію молодих гравців, які також, не зважаючи на гру, могли би приносити певний прибуток компанії. Це через покупки певних моделей на героїв або ж зброї.

Гра створена на сучасному рушії під назвою Unreal Engine для того, щоб бути конкурентноспроможною на ринку, та для подальшої підтримки продукту в майбутньому.

Так як ігри такого жанру як шутер та королівська битва наразі дуже популярні, було прийнято рішення про створення гри такого ж жанру, щоб підняти свій економічний план та свій рівень розуміння проєктування та розробки ігор такого плану.

Ціль цього проєкту, створити таку гру для геймерів, яку б не хотілося вимикати їм, щоб досліджувати її далі і далі.

Створення комп’ютерної гри це відповідальна справа, яка потребує великих затрат часу. Гру потрібно постійно підтримувати в належному стані, враховуючи усі методи та технології створення інших ігор.

Ця гра це компіляція, збірка вже існуючих ігор на даний момент, неможливо створити щось нове. Саме це і характеризує цей проєкт - модернізація чогось старого, або ж його редизайн це те, що може залучити нових гравців.

Потенційним замовником цього продукту вважається простий геймер. Для гра підійде для тих, хто не хоче грати тільки в ігри, які спрямовані на повний реалізм, та залежить тільки від одної смерті, що для великої частини гравців є проблемою.

Потребою гравців від цієї гри буде хороший геймплей, який може затягнути, велика частота кадрів для комфортної гри, підтримка старих платформ. Це не дуже легко зробити в цю епоху, кожного тижня виходять анонси про нові комплектуючі для ПК та про нові смартфони.

Конкурентами конкретно цьому проєкту складають великі компанії, такі як: Valve — CS:GO, Blizzard — Overwatch та багато інших, які розраховані на трішки іншу аудиторію в силу свого стилю, графіку, методом маніпулювання героєм і т.д.

# ВИСНОВКИ

Метою виконання дипломного проєкту було створення комп'ютерної гри в жанрі "First and Third-person Shooter" у стилі фентезі. У цій грі гравець керує двома персонажами, які, відповідно до сценарію гри, намагаються перемогти "зомбі" і відвоювати територію свого поселення.

Під час роботи над дипломним проєктом було виконано такі завдання:

- проведений аналіз ігрових продуктів-аналогів;
- проведений аналіз ігрових рушіїв;
- створено сценарій гри;
- визначені вимоги до функціоналу;
- спроєктовано та реалізовано віджети стартового та інших меню;
- реалізовано основні дії головних персонажів;
- реалізовано ігровий продукт на рушії Unreal Engine 4;
- проведено тестування ігрового продукту.

Програмний продукт розроблено згідно усіх поставлених вимог та вказівок. В майбутньому планується допрацювання функціоналу гри – створення повної сюжетної лінії, у якій будуть задіяні фактори погодних умов – вітру, туману, снігу, а також створення ще одної окремої мапи для цього режиму гри. Також розширення початкової локації та більша варіативність проходження рівнів. Розширена кількість зброї, додавання певних гранат, які можуть наносити пошкодження ворогу, так і зцілювати себе та інших союзних персонажів.

Гра виступає основою для розробки мультиплеєрних ігор, тому що написаний мережевий код є доступом до серверів Steam. За допомогою цього коду можна отримати інформацію з профілю гравця: його аватар, його ім'я на платформі Steam, його друзів. Код можна певним чином покращити, наприклад, додати перевірки на помилки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. MagicaVoxel - свободный воксельный 3d редактор. *Pikabu* : веб-сайт. URL: https://pikabu.ru/story/magicavoxel__svobodnyiy_vokselnyiy_3d_redaktor (дата звернення: 15.01.2020).

2. Blueprints Quick Start Guide. *Unreal Engine* : веб-сайт. URL: https://docs.unrealengine.com/en-US/Engine/Blueprints/QuickStart/index.html (Дата звернення: 02.01.2020).

3. Blender is our future. *Blender* : веб-сайт. URL: https://www.blender.org (Дата звернення: 05.01.2020).

4. GitHub – VROID. *GitHub* : веб-сайт. URL: https://github.com/VROID (Дата звернення: 02.01.2020).

5. Vroid 3D models. *Sketchfab* : веб-сайт. URL: https://sketchfab.com /tags/vroid (Дата звернення: 02.01.2020).

6. Mixamo. Get animated. *Mixamo* : веб-сайт. URL: http://mixamo.com (Дата звернення: 03.01.2020).

7. Steamworks API Overview. *SteamWorks* : веб-сайт. URL: https://partner. steamgames.com/doc/sdk/api (Дата звернення: 15.01.2020).

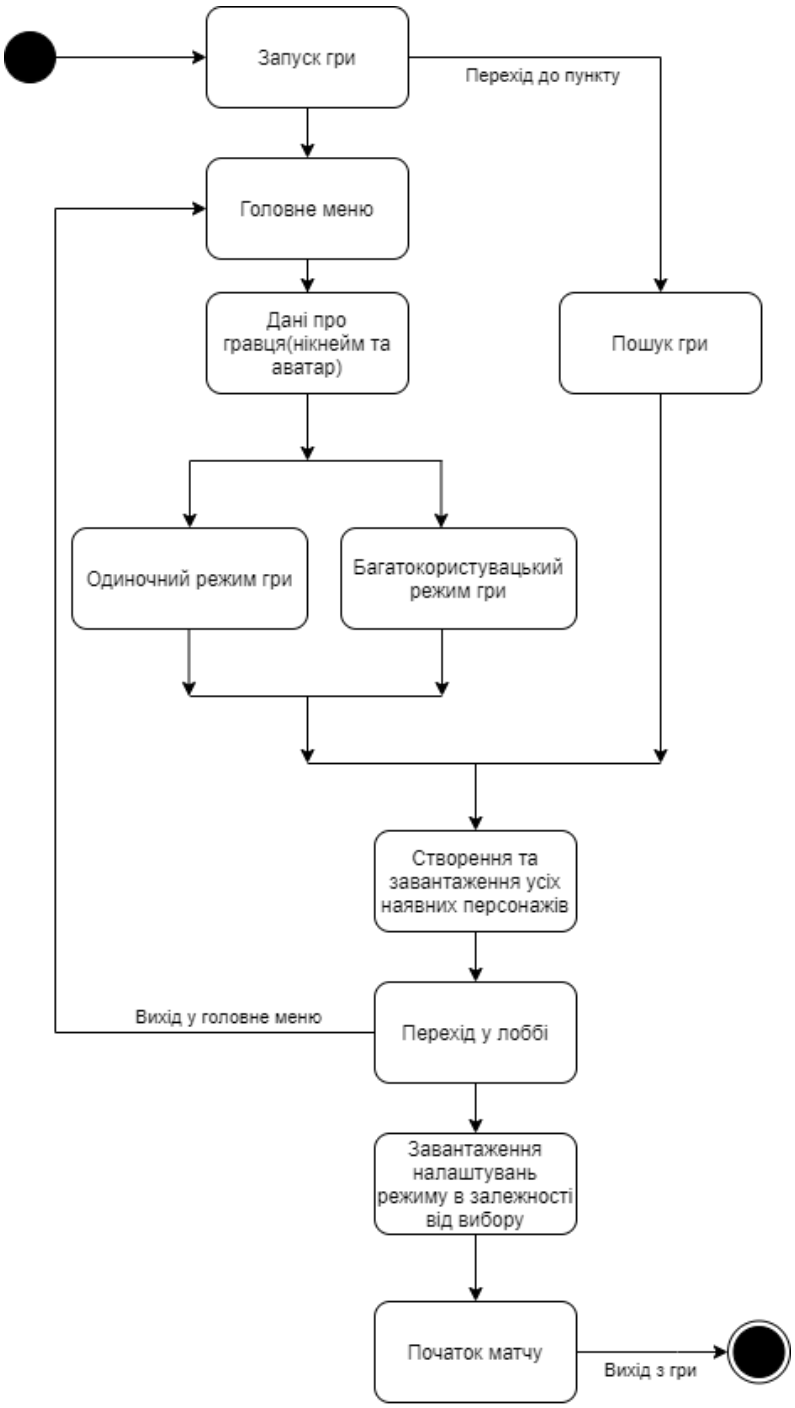
8. UE4 Tutorial: How to Connect a Multiplayer Game with Steam. *Medium* : веб-сайт. URL: https://medium.com/swlh/ue4-tutorial-how-to-connect-amulti player-game-with-steam-ccc89bd8d8a9 (Дата звернення: 16.01.2020).

9. Darmowe 3D Unreal Modele. *Free3D* : веб-сайт. URL: https://free3d.com/3d-models/unreal (Дата звернення: 17.01.2020).

# ДОДАТКИ

## Додаток А

## Діаграма діяльності



# Додаток Б

## Код А - “0_Base” у середовищі Visual Studio

```
#pragma once

#include "CoreMinimal.h"

#include "UObject/ObjectMacros.h"

#include "UObject/UObjectGlobals.h"

#include "Templates/SubclassOf.h"

#include "UObject/CoreNet.h"

#include "Engine/NetSerialization.h"

#include "Engine/EngineTypes.h"

#include "Components/ActorComponent.h"

#include "GameFramework/Actor.h"

#include "GameFramework/Pawn.h"

#include "Animation/AnimationAsset.h"

#include "GameFramework/RootMotionSource.h"

#include "Character.generated.h"

class AController;

class FDebugDisplayInfo;

class UAnimMontage;

class UArrowComponent;

class UCapsuleComponent;

class UCharacterMovementComponent;

class UPawnMovementComponent;

class UPrimitiveComponent;

class USkeletalMeshComponent;

struct FAnimMontageInstance;

DECLARE_DYNAMIC_MULTICAST_DELEGATE_ThreeParams(FMovementModeChangedSignature, class
ACharacter*, Character, EMovementMode, PrevMovementMode, uint8, PreviousCustomMode);

DECLARE_DYNAMIC_MULTICAST_DELEGATE_ThreeParams(FCharacterMovementUpdatedSignature, float,
DeltaSeconds, FVector, OldLocation, FVector, OldVelocity);

DECLARE_DYNAMIC_MULTICAST_DELEGATE(FCharacterReachedApexSignature);

DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FLandedSignature, const FHitResult&, Hit);

USTRUCT()

struct FSimulatedRootMotionReplicatedMove

{

        GENERATED_USTRUCT_BODY()

        UPROPERTY()
```

```
	float Time;

	UPROPERTY()

	FRepRootMotionMontage RootMotion;

};

namespace MovementBaseUtility

{

	ENGINE_API bool IsDynamicBase(const UPrimitiveComponent* MovementBase);

	FORCEINLINE bool UseRelativeLocation(const UPrimitiveComponent* MovementBase)

	{

		return IsDynamicBase(MovementBase);

	}

	ENGINE_API void AddTickDependency(FTickFunction& BasedObjectTick, UPrimitiveComponent*
NewBase);

	ENGINE_API void RemoveTickDependency(FTickFunction& BasedObjectTick,
UPrimitiveComponent* OldBase);

	ENGINE_API FVector GetMovementBaseVelocity(const UPrimitiveComponent* MovementBase,
const FName BoneName);

	ENGINE_API FVector GetMovementBaseTangentialVelocity(const UPrimitiveComponent*
MovementBase, const FName BoneName, const FVector& WorldLocation);

	ENGINE_API bool GetMovementBaseTransform(const UPrimitiveComponent* MovementBase, const
FName BoneName, FVector& OutLocation, FQuat& OutQuat);

}

USTRUCT()

struct FBasedMovementInfo

{

	GENERATED_USTRUCT_BODY()

	UPROPERTY()

	UPrimitiveComponent* MovementBase;

	UPROPERTY()

	FName BoneName;

	UPROPERTY()

	FVector_NetQuantize100 Location;

	UPROPERTY()

	FRotator Rotation;

	UPROPERTY()

	bool bServerHasBaseComponent;

	UPROPERTY()

	bool bRelativeRotation;

	UPROPERTY()
```

```
        bool bServerHasVelocity;
        FORCEINLINE bool HasRelativeLocation() const
        {
                return MovementBaseUtility::UseRelativeLocation(MovementBase);
        }
        FORCEINLINE bool HasRelativeRotation() const
        {
                return bRelativeRotation && HasRelativeLocation();
        }
        FORCEINLINE bool IsBaseUnresolved() const
        {
                return (MovementBase == nullptr) && bServerHasBaseComponent;
        }
};
UCLASS(config=Game, BlueprintType, meta=(ShortTooltip="A character is a type of Pawn that
includes the ability to walk around."))
class ENGINE_API ACharacter : public APawn
{
        GENERATED_BODY()
public:
        ACharacter(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());


        void GetLifetimeReplicatedProps(TArray<FLifetimeProperty>& OutLifetimeProps) const
override;


private:
        UPROPERTY(Category=Character, VisibleAnywhere, BlueprintReadOnly,
meta=(AllowPrivateAccess = "true"))
        USkeletalMeshComponent* Mesh;
        UPROPERTY(Category=Character, VisibleAnywhere, BlueprintReadOnly,
meta=(AllowPrivateAccess = "true"))
        UCharacterMovementComponent* CharacterMovement;
        UPROPERTY(Category=Character, VisibleAnywhere, BlueprintReadOnly,
meta=(AllowPrivateAccess = "true"))
        UCapsuleComponent* CapsuleComponent;
#if WITH_EDITORONLY_DATA
        UPROPERTY()
        UArrowComponent* ArrowComponent;
#endif
```

```
public:

	UFUNCTION(unreliable, server, WithValidation)

	void ServerMove(float TimeStamp, FVector_NetQuantize10 InAccel, FVector_NetQuantize100
ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View, UPrimitiveComponent*
ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	void ServerMove_Implementation(float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	bool ServerMove_Validate(float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	UFUNCTION(unreliable, server, WithValidation)

	void ServerMoveNoBase(float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View,
uint8 ClientMovementMode);

	void ServerMoveNoBase_Implementation(float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View,
uint8 ClientMovementMode);

	bool ServerMoveNoBase_Validate(float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 CompressedMoveFlags, uint8 ClientRoll, uint32 View,
uint8 ClientMovementMode);

	UFUNCTION(unreliable, server, WithValidation)

	void ServerMoveDual(float TimeStamp0, FVector_NetQuantize10 InAccel0, uint8
PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	void ServerMoveDual_Implementation(float TimeStamp0, FVector_NetQuantize10 InAccel0,
uint8 PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	bool ServerMoveDual_Validate(float TimeStamp0, FVector_NetQuantize10 InAccel0, uint8
PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	UFUNCTION(unreliable, server, WithValidation)

	void ServerMoveDualNoBase(float TimeStamp0, FVector_NetQuantize10 InAccel0, uint8
PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View, uint8
ClientMovementMode);

	void ServerMoveDualNoBase_Implementation(float TimeStamp0, FVector_NetQuantize10
InAccel0, uint8 PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View, uint8
ClientMovementMode);

	bool ServerMoveDualNoBase_Validate(float TimeStamp0, FVector_NetQuantize10 InAccel0,
uint8 PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View, uint8
ClientMovementMode);


	/** Replicated function sent by client to server - contains client movement and view
info for two moves. First move is non root motion, second is root motion. */
```

```
	UFUNCTION(unreliable, server, WithValidation)

	void ServerMoveDualHybridRootMotion(float TimeStamp0, FVector_NetQuantize10 InAccel0,
uint8 PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	void ServerMoveDualHybridRootMotion_Implementation(float TimeStamp0,
FVector_NetQuantize10 InAccel0, uint8 PendingFlags, uint32 View0, float TimeStamp,
FVector_NetQuantize10 InAccel, FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8
ClientRoll, uint32 View, UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName,
uint8 ClientMovementMode);

	bool ServerMoveDualHybridRootMotion_Validate(float TimeStamp0, FVector_NetQuantize10
InAccel0, uint8 PendingFlags, uint32 View0, float TimeStamp, FVector_NetQuantize10 InAccel,
FVector_NetQuantize100 ClientLoc, uint8 NewFlags, uint8 ClientRoll, uint32 View,
UPrimitiveComponent* ClientMovementBase, FName ClientBaseBoneName, uint8 ClientMovementMode);

	UFUNCTION(unreliable, server, WithValidation)

	void ServerMoveOld(float OldTimeStamp, FVector_NetQuantize10 OldAccel, uint8
OldMoveFlags);

	void ServerMoveOld_Implementation(float OldTimeStamp, FVector_NetQuantize10 OldAccel,
uint8 OldMoveFlags);

	bool ServerMoveOld_Validate(float OldTimeStamp, FVector_NetQuantize10 OldAccel, uint8
OldMoveFlags);

	UFUNCTION(unreliable, client)

	void ClientAckGoodMove(float TimeStamp);

	void ClientAckGoodMove_Implementation(float TimeStamp);

	UFUNCTION(unreliable, client)

	void ClientAdjustPosition(float TimeStamp, FVector NewLoc, FVector NewVel,
UPrimitiveComponent* NewBase, FName NewBaseBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	void ClientAdjustPosition_Implementation(float TimeStamp, FVector NewLoc, FVector
NewVel, UPrimitiveComponent* NewBase, FName NewBaseBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	UFUNCTION(unreliable, client)

	void ClientVeryShortAdjustPosition(float TimeStamp, FVector NewLoc,
UPrimitiveComponent* NewBase, FName NewBaseBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	void ClientVeryShortAdjustPosition_Implementation(float TimeStamp, FVector NewLoc,
UPrimitiveComponent* NewBase, FName NewBaseBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	UFUNCTION(unreliable, client)

	void ClientAdjustRootMotionPosition(float TimeStamp, float ServerMontageTrackPosition,
FVector ServerLoc, FVector_NetQuantizeNormal ServerRotation, float ServerVelZ,
UPrimitiveComponent* ServerBase, FName ServerBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	void ClientAdjustRootMotionPosition_Implementation(float TimeStamp, float
ServerMontageTrackPosition, FVector ServerLoc, FVector_NetQuantizeNormal ServerRotation, float
ServerVelZ, UPrimitiveComponent* ServerBase, FName ServerBoneName, bool bHasBase, bool
bBaseRelativePosition, uint8 ServerMovementMode);

	UFUNCTION(unreliable, client
```

Додаток В

Таблиця В.1 - Вимоги до апаратного забезпечення

| Рекомендовані характеристики комп’ютера | |
|---|---|
| Процесор | Intel Core i7 6700 3.4 ghz |
| Відеокарта | NVIDIA GeForce 1060 3GB |
| Оперативна пам’ять | 16GB |
| Інтернет | стабільне підключення 128 кб/с |
| Роздільна здатність монітора | 1920x1080 |
| Мінімальні характеристики комп’ютера | |
| Процесор | Intel Core i5 6500 2.8 ghz |
| Відеокарта | NVIDIA GeForce 1030 2GB |
| Оперативна пам’ять | 8GB |
| Інтернет | стабільне підключення 64 кб/с |
| Роздільна здатність монітора | 1366x768 |

# ВІДГУК
## на дипломний проект
## студента відділення комп'ютерних та видавничих технологій
## Галицького коледжу імені В'ячеслава Чорновола

студента IV курсу групи К-47

Шуволова Володимира
(прізвище та ініціали)

Спеціальність *122 „Комп'ютерні науки та інформаційні технології"*

Керівник ДП Кульчинська Н.З.

Тема: Комп'ютерна гра в стилі фентезі

1. Загальна характеристика студента Відповідальний та пунктуальний, вчасно та на високому рівні виконує поставлені завдання, вміє використовувати інформ. джерела, готувати та вирішувати фахові зав.

2. Практична або теоретична цінність опрацьованих питань Детально проаналізовано ринок комп'ютерних ігор, створений ігровий додаток налаштовано для підключ. до ігрових серверів; проект повністю готовий до викор.

3. Недоліки роботи
Гра має високі вимоги до апаратного забезпеч.

4. Загальний висновок
Створений додаток відповідає поставленим вимогам, готовий до практичного використання

Керівник дипломного проекту Кульчинська Н.З.
(прізвище та ініціали)

25.06.2020р.

РЕЦЕНЗІЯ
на дипломний проект
студента відділення комп ' ютерних та видавничих технологій
Галицького коледжу імені В'ячеслава Чорновола

студента IV курсу групи К-47

_____ Шувалова Володимира Юрійовича_____
(прізвище та ініціали)

Спеціальність *122„Комп'ютерні науки та інформаційні технології"*

Обсяг дипломного проекту: _76_ стор.
Кількість рисунків __64_арк.

Тема: Комп'ютерна гра в стилі фентезі

1. Актуальність теми:__створення комп'ютерних інді-ігор в жанрі «фентезі» задовільняють потреби коистувача якісній графічній складовій, розважальному контенті

2. Практична або теоретична цінність опрацьованих питань_____
полягає в створенні повнофункціонального ігрового продукту відповідної тематики, автор мотивує своє рішення щодо вибраних методів та засобів розробки та вдало їх використовує для реалізації поставлених завдань, створений ігровий продукт має мережевий код, який є доступом до серверів Steam.

3. Недоліки роботи__ слабке посилання на використані джерела , що в цілому не впливає на функціонал продукту

4. Загальний висновок____
дипломний проект виконаний згідно вимог і заслуговує на високу оцінку, а студент – присвоєння кваліфікації "техніка-програміста"

Рецензент_____ Посвятовська О.Б _____
(прізвище та ініціали рецензента)

«_25_» ___06_____2020р. викладач комп'ютерних дисциплін
Галицького коледжу імені В'ячеслава Чорновола

UNICHECK

Ім'я користувача:
**Наталя Кульчинська**

ID перевірки:
**1003802559**

Дата перевірки:
**05.06.2020 12:13:22 EEST**

Тип перевірки:
**Doc vs Internet + Library**

Дата звіту:
**16.02.2021 10:28:04 EET**

ID користувача:
**100004382**

---

Назва документа: **ДП Шувалов К47 перевірка**

Кількість сторінок: **62** Кількість слів: **7222** Кількість символів: **49999** Розмір файлу: **42.67 MB** ID файлу: **1003817361**

---

# 1.68%
# Схожість

Найбільша схожість: **1.02%** з Інтернет-джерелом (http://cad.kpi.ua/attachments/093_2016d_Avramenko.pdf)

| 1.68% Джерела з Інтернету | 30 | Сторінка 64 |
|---|---|---|

Не знайдено джерел з Бібліотеки

# 0.6% Цитат

| Цитати | 1 | Сторінка 65 |
|---|---|---|

Вилучення списку бібліографічних посилань вимкнене

# 2.41%
# Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

| 2.41% Вилучення з Інтернету | 57 | Сторінка 66 |
|---|---|---|

Немає вилучених бібліотечних джерел