**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

**Спеціальність: 123 «Комп'ютерна інженерія»**

**Освітня програма: «Комп'ютерна інженерія»**

**Група: 2БКС-27**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

**здобувача освіти денної форми навчання**

## БКС.27.08.000.КРБ

***ВОЛКОВА МИКОЛИ АНДРІЙОВИЧА***

**м. Одеса**
**2023 р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: **«Комп'ютерна інженерія»**

Група: **2БКС-27**

# ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему:

« Розробка ігрового проекту на базі ігрового двигуна Unity »

Проектний матеріал складається з пояснювальної записки на 56 сторінках та графічного (презентаційного) матеріалу на 11 аркушах (слайдах)

Виконавець _____ (Волков М.А.)

Керівник проекту _____ (Кунуп Т.В.)

**Консультанти:**

з охорони праці _____ ( Чорновол Н.І. )

з дотримання вимог ЄСКД _____ ( Петрашова В.І.)

старший консультант _____ ( Кривченко Ю.В. )

**До захисту допущений**

Завідувачка кафедри _____ ( Іванова Л.В. )

Завідувач відділення _____ (Скорнякова О.В.)

Захист «20» 06 2023 р. Протокол ДКК № 1

Оцінка ЕК 5 (відмінно)

Секретар ДКК _____

# АНОТАЦІЯ

Тема дипломної роботи «Розробка ігрового проекту на базі ігрового двигуна Unity».

У цій роботі описаний принцип розробки відеоігор і застосунків використовуючи багатоплатформовий інструмент і двигун Unity.

У аналітичній частині приведене технічне завдання до проекту, а також принцип розробки відеоігор та застосунків за подомогою двигуна Unity. Концептуалізація та розробка ігрового проекту демонструє підготовчий етап, на якому розроблюється план виконання проекту та його візуальна складова. У розділі “Реалізація та технологічні аспекти розробки” представлена покрокова інструкція реалізації програмного продукту використовуючи такий рушій як Unity та мову програмування C#. Останнім розділом стала охорона праці під час розробки програмного продукту.

Ключові слова: Unity, C#, NPC, UI, Колайдер, Скрипт.

# ANNOTATION

Thesis topic " Development of a game project based on a game engine Unity ". This paper describes the principle of developing a video games and applications using the multi-platform tool and engine Unity.

In the analytical part the technical task to the project, and also the principle of development of video games and applications using the Unity engine. Conceptualization and development of a game project demonstrates the preparatory stage in which the project implementation plan and its visual components. The section "Implementation and technological aspects of development" presents step-by-step instructions for implementing a software product using an engine such as Unity and the C# programming language. The last section was labor protection during the development of the application.

Keywords: Unity, C#, NPC, UI, Collider, Script.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Відділення *комп'ютерних систем* Кафедра *комп'ютерної інженерії*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР **Беркань І.В.**

"_____" _____202____р.

# ЗАВДАННЯ

**на кваліфікаційну роботу бакалавра**

Здобувачеві (здобувачці) освіти Волкову Миколі Андрійовичу

(прізвище, ім`я, по батькові)

1. Тема кваліфікаційної роботи

Розробка ігрового проекту на базі ігрового двигуна Unity

затверджена наказом по коледжу від "17" 10 2022 р. № 225-А2-ОД

2. Термін здачі кваліфікаційної роботи 20.06.2023р.

3. Вихідні дані до роботи

1. Ігровий двигун Unity
2. Сюжетна лінія та визначення геймплею
3. Набір готових 3D моделей та анімацій
4. Мова програмування для розробки проекту C#

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

Технологічний розділ розробки ігрового проекту.

Розділ охорони праці.

Висновки.

Перелік використаних джерел інформації.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Технології та інструменти; Опис ігрового проекту; Архітектура ігрового проекту;

Процес розробки ігрового проекту; Висновки.

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів роботи, що стосується їх

| Розділ | Консультант | ПІДПИС | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| *Основний* | *Кунуп Т.В.* | | |
| *Охорона праці* | *Чорновол В.І.* | | |
| *Нормоконтроль* | *Петрашова В.І.* | | |
| *Старший консультант* | *Кривченко Ю.В.* | | |
| | | | |

7. Дата видачі завдання 02.06.2023

Керівник роботи Кунуп Т.В.
(підпис)

Завдання прийняв до виконання Волков М.А.
(підпис)

**КАЛЕНДАРНИЙ ПЛАН**

| Пор. № | Назва етапів кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Вступ | 4.05.2023 | Виконано |
| 2. | Визначення технічного завдання на роботу | 8.05.2023 | Виконано |
| 3. | Огляд технологій програмування | 10.05.2023 | Виконано |
| 4. | Реалізація ігрового проекту | 15.05.2023 | Виконано |
| 5. | Тестування ігрового проекту | 17.05.2023 | Виконано |
| 6. | Розділ охорони праці | 22.05.2023 | Виконано |
| 7. | Висновки | 26.05.2023 | Виконано |
| 8. | Перелік літератури | 28.05.2023 | Виконано |
| 9. | Оформлення пояснювальної записки | 31.05.2023 | Виконано |
| 10. | Оформлення графічної частини | 05.06.2023 | Виконано |
| 11. | Малий захист кваліфікаційної роботи | 15.06.2023 | Виконано |
| | | | |
| | | | |
| | | | |
| | | | |

Виконавець Волков М.А.
(підпис)

Керівник роботи Кунуп Т.В.
(підпис)

| Формат | Зона | Поз. | Позначення | Назва | Кіл. | Примітка |
|---|---|---|---|---|---|---|
| | | | | ***Документація*** | | |
| | | | | | | |
| | | | *БКС 27. 08 000. 00 КРБ* | *Кваліфікаційна робота* | | |
| | | | | | | |
| *А4* | | | *БКС 27. 08 000. 00 КРБ ПЗ* | *Пояснювальна* | *1* | |
| | | | | *записка* | | |

| Зм. | Арк. | № докум. | Підпис | Дата | *БКС 27. 08 000. 00 КРБ ПЗ* | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Волков М.А. | | | Розробка ігрового проекту на базі ігрового двигуна Unity | Літ. | Аркуш | Аркушів |
| Перевірив | | Куруп Т.В. | | | | НДП | | |
| Н. Контр. | | Петрашова В.І. | | | | ВСП «ОТФК ОНТУ» гр.2БКС-27 | | |
| Затверд. | | Кривченко Ю.В. | | | | | | |

**ЗМІСТ**

# ВСТУП

Ця кваліфікаційна робота бакалавра присвячена розробці ігрового проекту на базі ігрового двигуна Unity. Ігри є важливим аспектом сучасної розвагової індустрії та мають значний потенціал як забавка, так і навчальний інструмент. Розробка ігрового проекту на базі Unity дозволяє поєднати творчість з технічними навичками, створюючи інтерактивні та захоплюючі віртуальні світи.

Метою цієї роботи є дослідження та розробка ігрового проекту з використанням ігрового двигуна Unity. У процесі реалізації цієї мети будуть вивчені основні концепції та функціональні можливості Unity, розроблені різні елементи гри, такі як графічний інтерфейс, фізика, штучний інтелект та анімації.

В процесі виконання кваліфікаційної роботи буде проведено аналіз існуючих ігрових проектів, які були розроблені з використанням Unity, а також будуть вивчені кращі практики та прийоми, що допоможуть удосконалити розробку власного ігрового проекту. Практична частина роботи включатиме створення прототипу ігрового проекту з використанням Unity, його тестування та оптимізацію.

# 1. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

## 1.1 Технічне завдання для ігрового проекту

Перед мною постало завдання розробити ігровий проект на базі ігрового двигуна Unity. Цей проект передбачає створення Village Defence – FPS гри, у якій гравцю доведеться захищати цивільних мешканців від ворогів, що з’являються під час хвиль нападу. Розробка ігрового проекту має наступні вимоги:

1. Створення гри з трьома хвилями нападу.
2. Створення ворогів різних типів.
3. Створення ігрової локації.
4. Керування персонажем гравця за допомогою клавіатури та миші.
5. Розміщення об’єктів та NPC відповідно до дизайну гри.
6. Реалізація механіки гри, включаючи пересування персонажа, взаємодію з об'єктами, бойової системи, поведінки NPC та зміни хвиль нападу.
7. Управління камерою.
8. Реалізація системи здоров'я гравця та ворогів.
9. Розробка іммерсивного ігрового середовища з деталізованими об'єктами, текстурами та освітленням.
10. Використання анімацій для персонажів та об'єктів у грі.
11. Проект повинен бути розроблений з використанням ігрового двигуна Unity.
12. Гра повинна працювати на операційних системах Windows, macOS та Linux.
13. Мова програмування: C#.

## 1.2 Проектування ігрового проекту

### 1.2.1 Проектування геймплею

Геймплей визначає основні правила, механіки та елементи взаємодії між гравцем та грою, які забезпечують незабутній досвід гравця.

Під час проектування геймплею ми ретельно розробляємо сюжетну лінію та основні події гри. Визначаємо головний конфлікт, мету гравця та можливі шляхи її досягнення. Це допомагає створити цікаву та захоплюючу історію, яка буде мотивувати гравця просуватися далі у грі.

Крім того, проводиться аналіз різних ігрових механік, які можуть бути використані у проекті. Це можуть бути бойові системи, головоломки, розв'язання загадок, гра на реакцію тощо. Вибір ігрових механік пов'язаний з жанром гри та цільовою аудиторією, до якої вона спрямована.

Наступним кроком є проектування ігрових механік та рівнів складності. Визначаємо, які навички та ресурси гравець може отримати протягом гри, які перешкоди йому доведеться подолати та які нагороди він може отримати за успішне виконання завдань.

Далі розробляються рівні гри, включаючи створення карт гри, розташування об'єктів та ігрових елементів, а також визначення логіки їх взаємодії. Кожен рівень може мати свою унікальну мету та складність, що додасть різноманітності геймплею.

За мету гри гравцеві буде необхідно здолати усіх ворогів швидше, ніж вони вб’ють всіх цивільних мешканців селища чи самого гравця. На початку хвилі з’являтимуться вороги та мешканці селища. Кожен ворог обирає випадкову ціль та прямує до неї. Коли ворог підійде достатньо близько до цілі він почне атакувати її. Якщо гравець підійде близько до ворога, то ворог почне наздоганяти гравця та намагатися вбити його. Гравець має можливість втекти від ворога. Якщо гравець втече від ворога, він знайде нову ціль серед цивільних мешканців та почне намагатися вбити її.

На локації селища розмістимо крамницю, у якій гравець зможе придбати нову зброю за кошти, отримані за вбивство ворогів. Кожна зброя матиме власні анімації атаки та характеристики.

Зробимо декілька хвиль, які будуть відрізнятися кількістю ворогів, їх типами та складністю. У разі програшу гравець матиме вибір, розпочати поточну хвилю з початку чи видалити весь свій прогрес та почати гру з початку.

### 1.2.2 Створення сцен та об’єктів

Для розробки ігрових проектів на базі ігрового двигуна Unity використовується візуальна середовище розробки, що дозволяє зручно та ефективно створювати сцени та об'єкти. Unity надає можливість створювати 3D- та 2D-сцени, а також розміщувати на них об'єкти, які взаємодіють з гравцем.

Першим кроком у створенні сцени є визначення її макету та розмірів. Для цього використовуються редактор сцен Unity, де можна додавати та розміщувати різні об'єкти, встановлювати їх положення, розміри та повороти. Крім того, редактор дозволяє налаштовувати освітлення, тіні, камери та інші параметри сцени.

У створенні об'єктів гри велику роль відіграють моделі. Unity підтримує різні формати моделей, такі як FBX, OBJ, 3DS та інші. Розробник може використовувати готові моделі зі свободних або комерційних джерел, або створювати власні моделі у спеціалізованих 3D-редакторах, таких як Blender або Autodesk Maya.

Після додавання моделей до сцени, розробник може встановлювати їх властивості та параметри, такі як фізика, колізії, анімація та інші. Unity надає потужні інструменти для налаштування та контролю поведінки об'єктів у грі.

Для забезпечення взаємодії об'єктів у грі, використовуються скрипти. Unity підтримує мови програмування C# та UnityScript, які дозволяють розробнику створювати скрипти для керування поведінкою об'єктів. Скрипти дозволяють

об'єктам реагувати на дії гравця, виконувати розрахунки, змінювати стани та забезпечувати різноманітну функціональність.

Використавши завантаженні моделі створимо сцену невеликого селища, з різними будівлями, NPC, кладовищем, декораціями та крамницею.

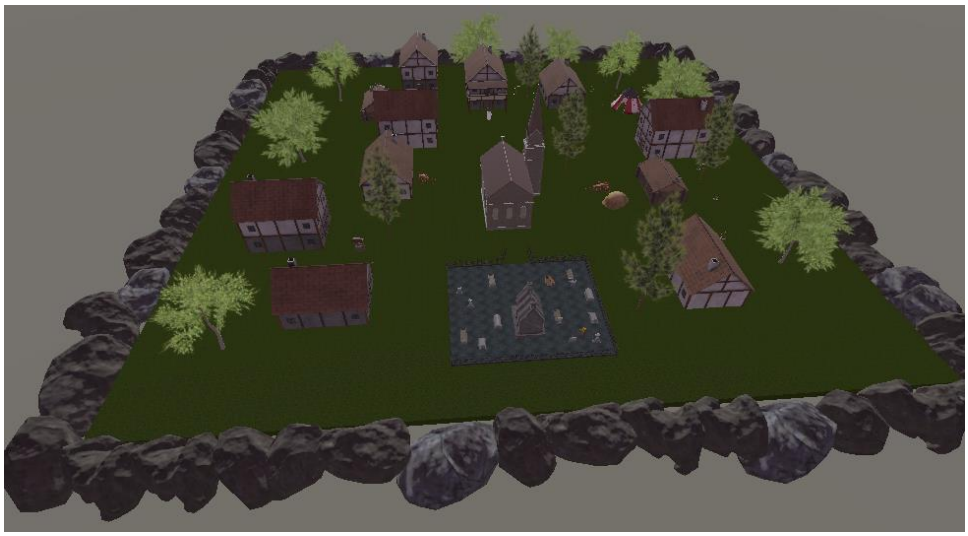


Рисунок 1.1  Загальний вигляд локації селища

Також додамо у якості функціональних декорацій на сцену рослини. У разі потреби гравця у відновленні здоров’я він зможе підійти до рослини та з’їсти її плоди, якщо вони є. Розмістимо їх у різних місцях локації.

Рисунок 1.2  3D модель рослини з баклажанами

Розмістимо на локації кладовище, на якому будуть з’являтися вороги. Для створення кладовища завантажимо 3D моделі склепу, надгробних плит, паркану та воріт.

Рисунок 1.3  Префаб кладовища

Розмістимо крамницю, у якій гравець зможе придбати нову чи замінити зброю. Встановимо завантажену модель крамниці на локації. Для того щоб гравець мав можливість взаємодіяти з крамницею додамо на неї колайдер.

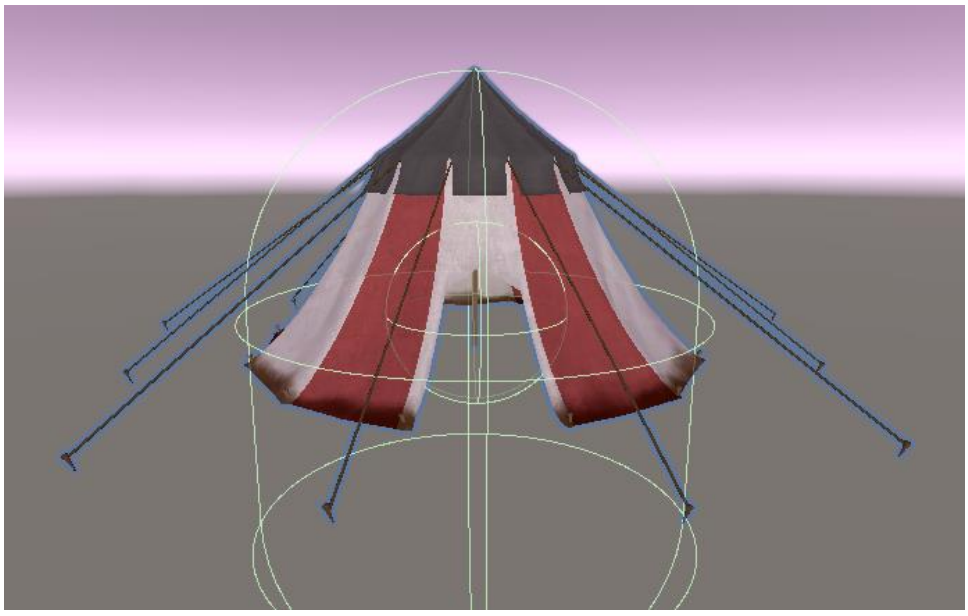


Рисунок 1.4 Префаб крамниці

На локації селища розмістимо завантажені 3D моделі будинків та декорації. Після розміщення моделей локація має наступний вигляд:

Рисунок 1.5  Розташування будинків та декорацій на локації

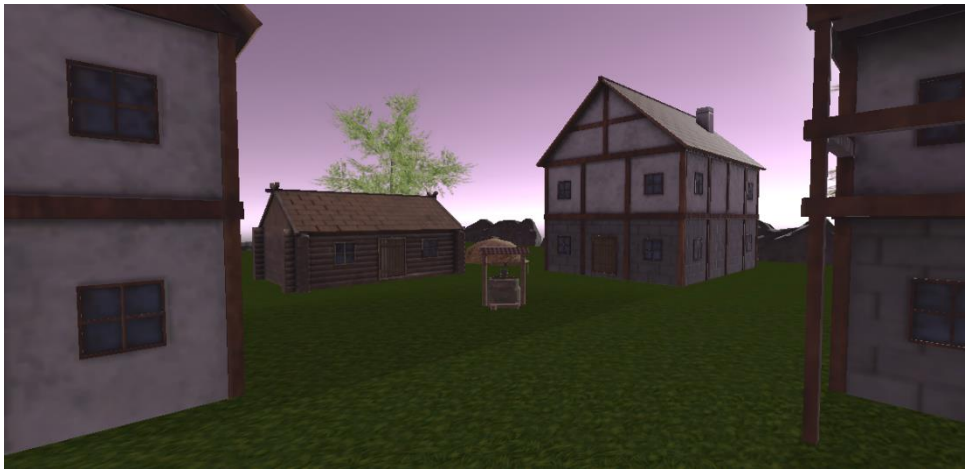


Рисунок 1.6 Розташування будинків та декорацій на локації

Рисунок 1.7  Вигляд церкви на локації

### 1.2.3 Графічне оформлення та ефекти

Один із важливих аспектів графічного оформлення - це візуальні ефекти, такі як освітлення, тіні, частинки, пост-процесинг та багато інших. Unity має широкі можливості для створення цих ефектів, використовуючи свою систему візуалізації та шейдерів.

Освітлення впливає на атмосферу та настрій гри. Unity підтримує різні види освітлення, включаючи точкове, напрямне, плямкове та глобальне освітлення. За допомогою цих типів освітлення можна створювати різні ефекти, такі як реалістичні тіні, блики, відблиски та інші.

Частинкові ефекти дозволяють створювати рухомі об'єкти, дим, вогонь, вибухи та багато іншого. Unity надає можливість використовувати систему частинок, що дозволяє контролювати поведінку та вигляд частинок. Розробник

може встановлювати швидкість, напрямок, розмір, кольори та інші параметри частинок, що дозволяє створювати вражаючі ефекти.

Пост-процесинг є ще одним важливим елементом графічного оформлення. Це процес, який застосовується після рендерингу кадра і дозволяє покращити його якість та стилістику. Unity має вбудовані інструменти пост-процесингу, що дозволяють розробнику додавати різні ефекти, такі як розмиття, колірні фільтри, корекція освітлення та інші, що покращують візуальний вигляд гри.

Графічне оформлення та ефекти відіграють важливу роль у створенні якісного ігрового проекту на базі ігрового двигуна Unity, забезпечуючи захоплюючі візуальні враження та залучаючи гравців до гри.

У проекті реалізуємо зміну часу доби. Для цього будемо змінювати положення сонця на небі та його яскравості. У налаштуваннях проекту змінимо Skybox сцени та джерело світла (рисунок 1.9). Skybox зазвичай представляє собою текстуру або набір текстур, розміщених на внутрішній поверхні куба, що оточує сцену. Ці текстури зображують панораму або краєвид з усіх сторін неба. Куб знаходиться великою відстанню від камери, що дозволяє йому служити фоном для 3D-сцени, незалежно від положення камери або об'єктів у сцені.

При використанні Skybox комп'ютерна програма, яка відтворює 3D-сцену, проеціює відповідні частини Skybox на задній план сцени, що створює враження, що об'єкти розташовані в реальному світі з навколишнім оточенням.

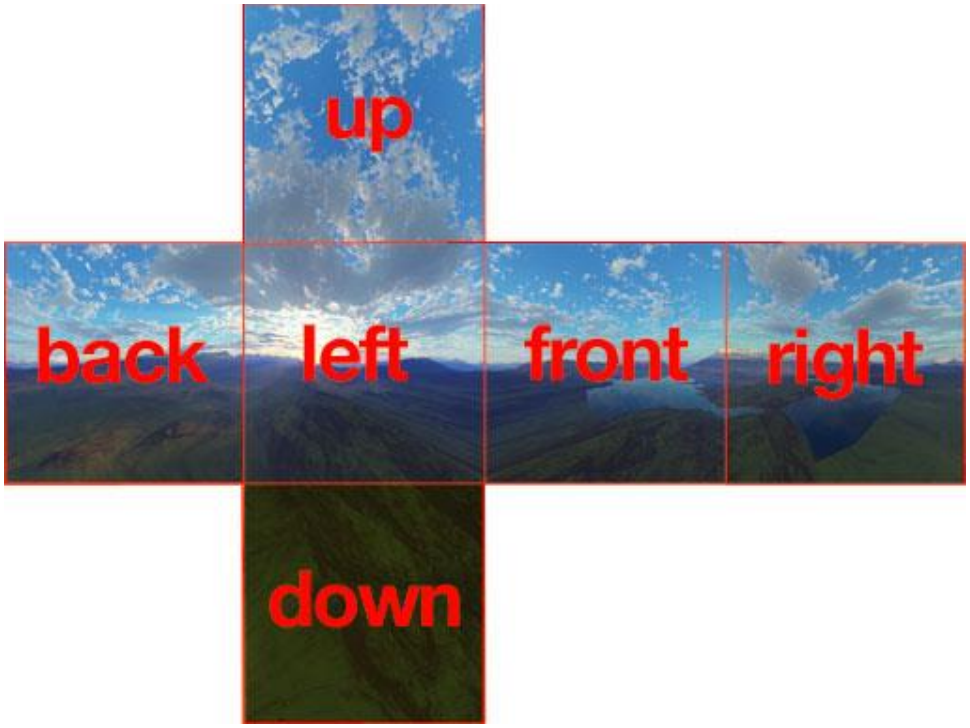



Рисунок 1.8  Приклад розгортки Skybox

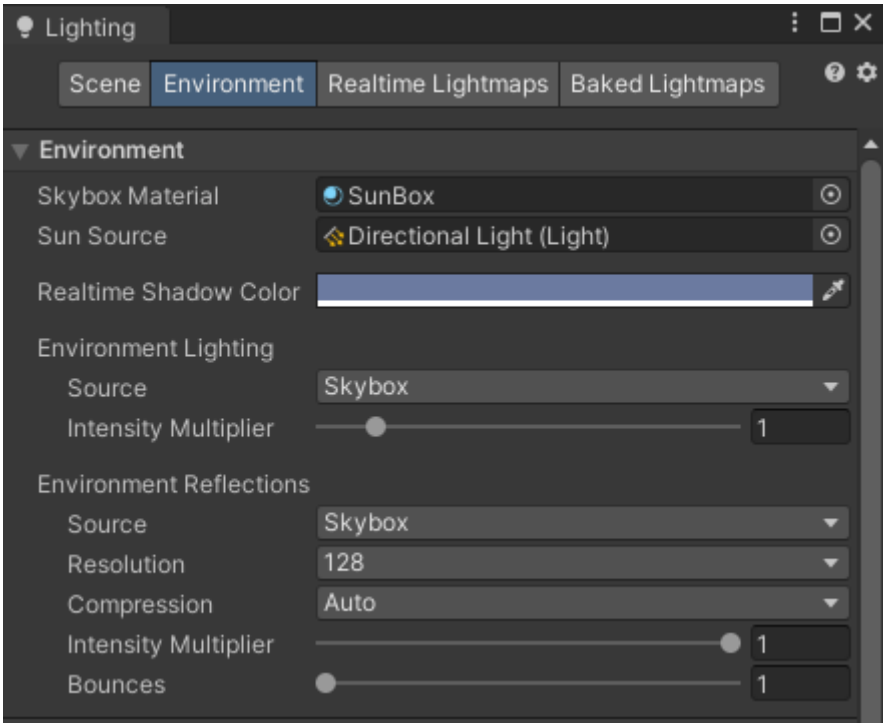



Рисунок 1.9  Вікно налаштувань світла проекту

Unity надає можливість створювати анімації для різних об’єктів. Створимо для кожного типу зброї власні анімації атаки та блокування. Для цього скористуємося вкладкою Animation. Щоб створити анімацію для будь-якого об’єкту потрібно його обрати та натиснути кнопку Create у цій вкладці. Після створення анімації у вкладці з’являється тайм-лайн анімації та кнопки налаштування анімації. На тайм-лайні анімації ми можемо обирати необхідний нам кадр та налаштовувати позицію чи обертання об’єкту відносну обраного кадру. Після створення анімації ми маємо можливість додавати події до будь-якого кадру.

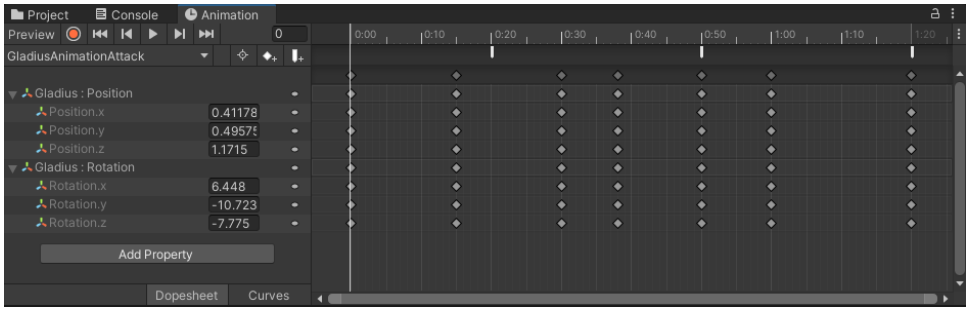



Рисунок 1.10 Вкладка Animation та налаштування анімації

Додамо візуальних ефектів при нанесенні шкоди ворогам. Для цього у Unity є система частинок - Particles System. Інструменти редактора Unity надають можливість створювати будь-які частинки, налаштовуючи їх та надаваючи їм функціональності. Для нашого проекту завантажимо готове рішення. Додамо завантажену систему частинок до префабу наших ворогів та розмістимо іх у необхідній позиції. При нанесенні шкоди ворогам ми будемо вмикати ці частинки.

Рисунок 1.11 – Анімація системи частинок при нанесенні шкоди ворогам

Також додамо систему частинок до ворогів, які мають особливі механіки. Під час лікування ворогів у них з’являється відповідна анімація системи частинок.

Рисунок 1.12  Анімація системи частинок при лікуванні ворога

Під час воскресіння інших ворогів у жреця також з’являється анімація частинок.

Рисунок 1.13  Анімація системи частинок при воскресінні інших ворогів

### 1.2.4 Інтерфейс користувача

Інтерфейс користувача є важливою складовою будь-якого ігрового проекту, оскільки він забезпечує взаємодію гравця з грою та передачу необхідної інформації.

Unity надає розробнику потужні інструменти для створення інтерфейсу користувача. Редактор інтерфейсу Unity дозволяє легко створювати різноманітні елементи UI, такі як кнопки, текстові поля, полоси прокрутки, меню та інші. Ці елементи можна розміщувати на екрані гри, встановлювати їх положення, розміри та стилістику.

Для стилізації елементів UI використовуються графічні компоненти, такі як фонові зображення, текстури, колірні схеми та шрифти. Unity дозволяє легко

імпортувати та використовувати графічні ресурси, що дозволяє розробникам створювати унікальний та привабливий вигляд інтерфейсу.

Додатково до графічного оформлення, функціональність інтерфейсу користувача визначається через роботу зі скриптами. Скрипти дозволяють встановлювати взаємодію елементів UI з ігровим світом та логікою гри. Наприклад, кнопка може мати скрипт, який викликає певну дію при натисканні на неї, а текстове поле може відображати певну інформацію залежно від стану гри.

Під час розробки інтерфейсу користувача, важливо враховувати зручність та доступність для гравців. Інтерфейс повинен бути інтуїтивно зрозумілим, ергономічним та ефективним у використанні. Також, адаптивний дизайн може бути використаний для пристосування інтерфейсу до різних пристроїв та роздільних здатностей екрану.

Створимо основне UI вікно для гравця. На ньому розмістимо таймер, гаманець та показник здоров’я. Розмістимо пустий об’єкт зверху, у нього додамо спеціальний UI об’єкт з можливістю розміщення тексту TextMeshPro. У налаштуваннях цього об’єкту оберемо колір тексту, його розмір та інші налаштування. Для гаманцю зробимо теж саме та додамо іконку валюти. Для цього використуємо інший UI об’єкт - Image. Показник здоров’я розмістимо знизу. Він складатиметься з текстового об’єкту, який буде вказувати поточну кількість одиниць здоровья та малюнку, який буде заповнюватися відповідно поточної кількості одиниць здоров’я.


Рисунок 1.14  Зовнішній виглід основного вікна

UI вікно програшу міститиме текст програшу та дві кнопки. Гравець матиме вибір між перезавантаженням поточної хвилі та видаленням його прогресу. Для цього у вікні програшу розмістимо текстовий об’єкт та дві кнопки. У якості кнопки в Unity може бути будь-який об’єкт з малюнком та компонентом Button.


Рисунок 1.15  Зовнішній вигляд вікна програшу

Зробимо екран завантаження. Для цього створимо окрему сцену та канвас на ній. Розмістимо фоновий малюнок, лого з назвою гри та шкалу прогресу завантаження.


Рисунок 1.16 Екран завантаження гри

## 1.3 Реалізація ігрового проекту

### 1.3.1 Розробка архітектури проекту

Для розробки ігрового проекту ми використовуємо ігровий двигун Unity. Для зміни рівнів (хвиль монстрів) реалізуємо логіку об’єкта, який буде зберігати інформацію про хвилі, та взаємодіяти з ними. Уся взаємодія з хвилями повинна виконуватись через цей об’єкт. Через нього ми зможемо почати або завершити любу хвилю, отримати інформацію про NPC та взаємодіяти з хвилями. Хвилі будуть зберігати розміщених на ній NPC та передавати інформацію до об’єкта - тримача при зверненні.

Створимо декілька типів ворогів, які будуть відрізнятись між собою візуально та матимуть різні механіки, а саме:

Звичайний ворог. Матиме модель скелета, має можливість бути озброєним зброєю, завдяки якій буде завдавати більше шкоди гравцю та матиме більше одиниць здоров’я. Додаткової механіки не матиме, так як це звичайний ворог.

Жрець. Матиме модель жреця з палицею та більшу кількість одиниць шкоди та здоров’я ніж у звичайного ворога. Додатковою механікою є воскресіння інших ворогів, якщо вони знаходяться неподалік від жерця.

Чумний лікар. Матиме модель чумного лікаря з жезлом та велику кількість одиниць шкоди та здоровья. Додатковою механікою є лікування поранених ворогів якщо вони знаходяться неподалік від лікаря.

Рисунок 1.17  Види ворогів у грі

Створимо скрипт для управління гравцем, який буде зчитувати введення з клавіатури та реалізовувати його. Також для гравця потрібен основний скрипт, через який інші скрипти будуть взаємодіяти з гравцем. У ньому будуть зберігатися додаткові скрипти, з якими пов’язаний гравець.

Для всіх персонажів створимо скрипт **Health** для зберігання та обробки інформації стосовно їх здоров’я. Задля нанесенні шкоди персонажу створимо метод GetDamage до якого будемо передавати кількість шкоди яку необхідно завдати.

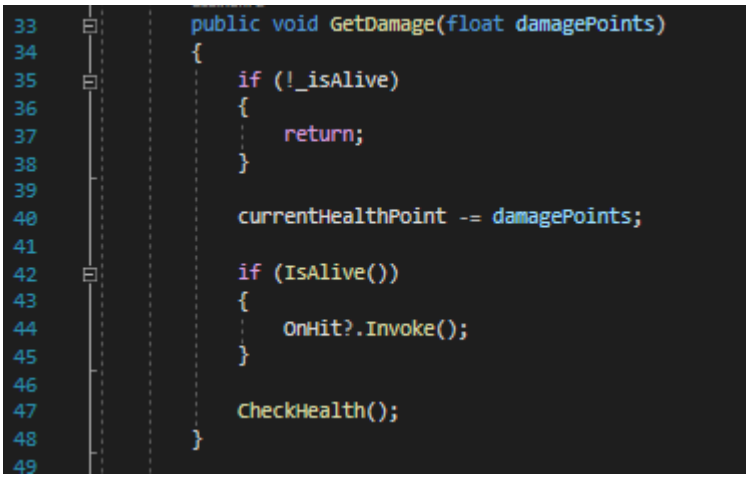



Рисунок 1.18 Код методу GetDamage

Після нанесення шкоди персонажу нам необхідно перевірити його стан. Для цього створимо метод CheckHealth. Якщо у персонажа не залишилось одиниць здоров'я тоді викликаємо подію OnDeath. На цю подію при ініціалізації персонажа підписується його основний скрипт.

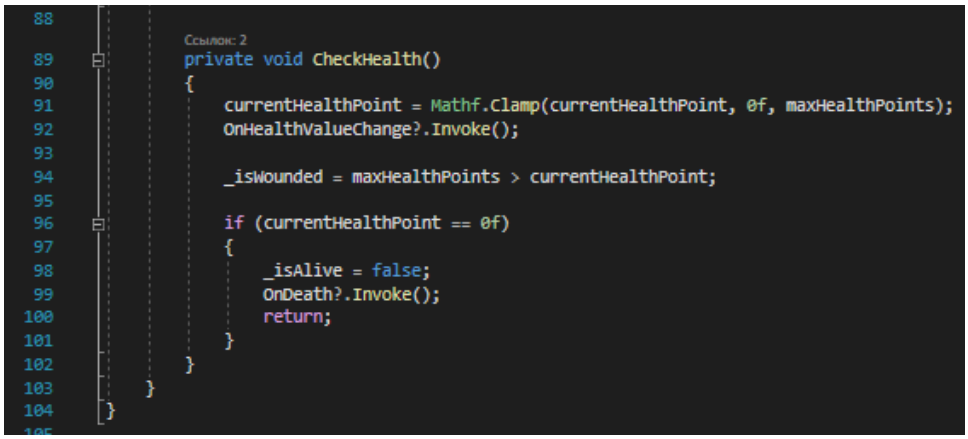



Рисунок 1.19 Код методу CheckHealth

Додамо менеджер гри **GameManager**, у якому буде реалізована зміна часу доби з подальшою взаємодією з хвилями та обробка прогресу гри. У ньому ми зможемо задавати швидкість зміни часу доби, позицію сонця та його яскравість.

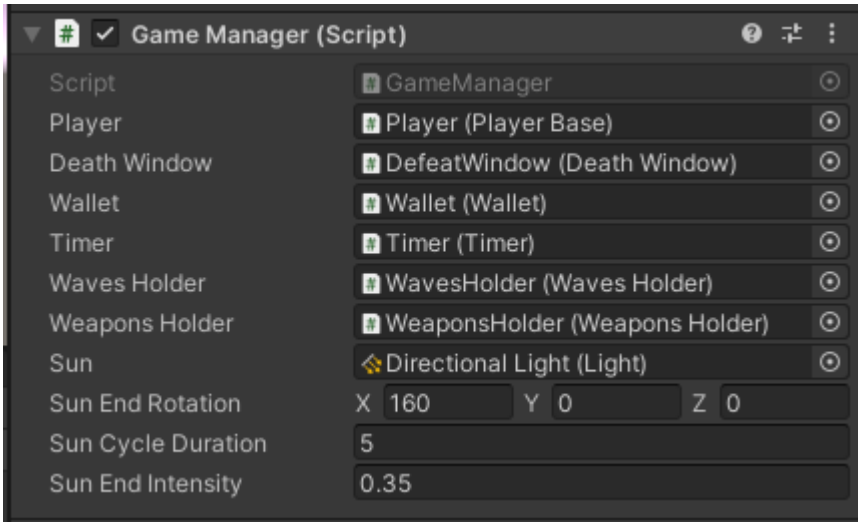



Рисунок 1.20 Вигляд менеджеру гри у редакторі Unity

Для зберігання та взаємодії зі зброєю гравця створимо скрипт **WeaponsHolder**. Скрипт міститиме усі види зброї. Зміна обраної зброї, атака чи блок будуть визиватись через цей скрипт. Під час інітіалізації скрипта він буде брати характеристики зброї з загального конфігу зброї та передавати їх до кожної зброї окремо.

```
33      private void InitializeHolder()
34      {
35          if (weapons.Any())
36          {
37              WeaponInfo currentWeapon = weaponsConfig.weapons.Find(w => w.state == ItemState.Equipped);
38              _currentWeapon = weapons.Find(w => w.GetId() == currentWeapon.id);
39              weapons.ForEach(w =>
40              {
41                  WeaponInfo weapon = weaponsConfig.weapons.Find(i => i.id == w.GetId());
42                  float damage = weapon.damage;
43                  int critChance = weapon.criticalDamageChance;
44                  float critPercente = weapon.criticalDamagePercent;
45                  w.InitializeWeapon(damage, critChance, critPercente);
46              });
47              _currentWeapon.SelectWeapon();
48              weaponsConfig.OnApply += UpdateWeapon;
49          }
50      }
51
52      private void UpdateWeapon()
53      {
54          var selectedWeapon = weaponsConfig.weapons.Find(w => w.state == ItemState.Equipped);
55          SelectWeapon(selectedWeapon.id);
56      }
```

Рисунок 1.21 Код методу інітіалізації WeaponsHolder

Кожен вид зброї матиме власну 3d - модель та скрипт, який буде зберігати унікальний номер зброї та посилання на компонент - аніматор. На кожній зброї буде колайдер який буде реагувати на контакт з ворогами при атаці та блоці.

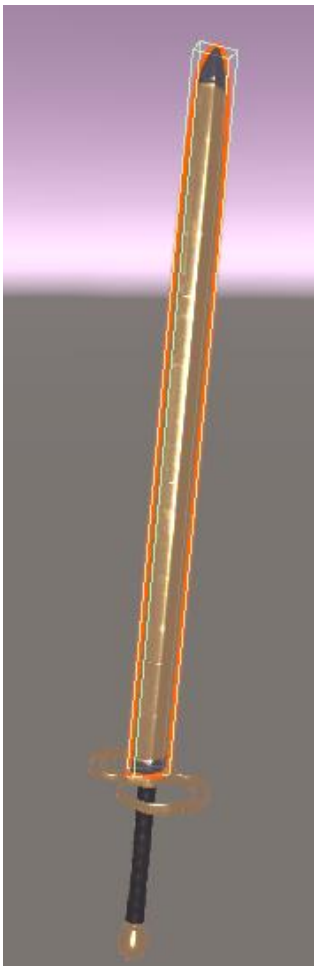


Рисунок 1.22  Розташування колайдеру на зброї

Зміна зброї буде реалізована через покупку її у магазині за внутрішньоігрову валюту, яку гравець буде отримувати при вбивстві ворогів та зберігатиметься у скрипті гаманця **Wallet**. Цей скрипт зберігатиме інформацію про кількість грошей у гравця та оброблювати її. Будь-яка взаємодія з гаманцем гравця проходить через цей скрипт.

```
ссылка: 1
public void IncreaseValue(int value)
{
    coinsValue += value;
    OnValueChange?.Invoke();
    Save();
}

ссылка: 1
public void DecreaseValue(int value)
{
    coinsValue -= value;
    OnValueChange?.Invoke();
    Save();
}

Ссылок: 3
private void Save()
{
    PlayerPrefs.SetInt(_walletKey, coinsValue);
    PlayerPrefs.Save();
}
```

Рисунок 1.23 Скриншот коду методів скрипту гаманця

### 1.3.2 Реалізація геймплею

Створимо скрипт **Wave** для роботи та взаємодії з хвилями монстрів. У цьому скрипті створюємо змінну id яка буде зберігати унікальний номер хвилі та список зі всіма NPC які будуть у хвилі. Для швидкого доступу до окремих типів NPC створимо поля які міститимуть NPC за різними типами. Вони будуть заповнюватись при старті роботи скрипта.

```
public class Wave : MonoBehaviour
{
    [SerializeField] private int id;
    [SerializeField] private List<NPCBase> npcs;

    private List<NPCBase> _enemies;
    private List<NPCBase> _citizens;

    public Action<bool> OnWaveEnd;

    Сообщение Unity | Ссылок: 0
    private void Awake()
    {
        _enemies = GetTypeNPCs(NPC_Type.Enemy, true);
        _citizens = GetTypeNPCs(NPC_Type.Citizen, true);
    }
```

Рисунок 1.24 Скриншот полів класу Wave

На початку роботи скрипта ми підписуємось на подію смерті для кожного NPC. При цій події ми перевіряємо поля з NPC, чи залишився хоч один NPC живим. Якщо всі NPC окремого виду гинуть то скрипт викликає подію кінця хвилі та передає значення стану хвилі, переміг гравець чи програв.

```
26      private void Start()
27      {
28          _enemies.ForEach(e => e.OnDeath += CheckWaveWin);
29          _citizens.ForEach(c => c.OnDeath += CheckWaveLose);
30      }
31
        Ссылок: 3
32      private void CheckWaveLose()
33      {
34          var aliveCitizens = _citizens.Where(c => c.IsAlive()).ToList().Count;
35          if (aliveCitizens <= 0)
36          {
37              OnWaveEnd?.Invoke(false);
38              _citizens.ForEach(c => c.OnDeath -= CheckWaveLose);
39          }
40      }
41
        Ссылок: 3
42      private void CheckWaveWin()
43      {
44          var aliveEnemiesCount = _enemies.Where(e => e.IsAlive()).ToList().Count;
45          if (aliveEnemiesCount <= 0)
46          {
47              OnWaveEnd?.Invoke(true);
48              _enemies.ForEach(e => e.OnDeath -= CheckWaveWin);
49          }
50      }
```

Рисунок 1.25 – Скриншот коду методів перевірки стану хвилі

Для зберігання та взаємодії з хвилями створюємо скрипт **WavesHolder**. В нього передаємо усі хвилі та додаємо посилання на гравця для взаємодії з ним. Під час виклику події OnWaveEnd у поточній хвилі ми з цього скрипта звертаємось до менеджера гри та передаємо стан хвилі, програв чи виграв гравець.

Скрипт **GameManager** при закінченні поточної хвилі у разі виграшу гравця буде зберігати інформацію про те що поточна хвиля пройдена та запускати наступну хвилю. У разі програшу гравець побачить вікно програшу, у якому зможе почати поточну хвилю з початку або видалити всю збережену інформацію о прогресі гравця та почати гру з початку. На початку роботи скрипт буде запускати таймер з відліком, після чого почнеться збережена хвиля,

яку гравець ще не пройшов. Зміна дня та вечора, взаємодія з хвилями та збереженням даних виконується лише через цей скрипт.

```
111         private void OnWaveEnd(bool state)
112         {
113             if (state)
114             {
115                 SaveProgress(state);
116                 SwitchDayTime();
117             }
118             else
119             {
120                 player.LoseControl();
121                 OnPlayerDeath();
122             }
123         }
```

Рисунок 1.26 Скриншот коду метода OnWaveEnd

```
75          private void SaveProgress(bool state)
76          {
77              wavesHolder.OnCurrentWaveEnd -= OnWaveEnd;
78
79              if (state)
80              {
81                  int oldWaveId = PlayerPrefs.GetInt(_savedWaveIdKey, 1);
82                  int nextWaveId = oldWaveId + 1;
83                  if (nextWaveId > _wavesCount)
84                  {
85                      FinishGame();
86                      return;
87                  }
88                  PlayerPrefs.SetInt(_savedWaveIdKey, oldWaveId + 1);
89                  PlayerPrefs.Save();
90              }
91
92              timer.StartCountDown();
93          }
```

Рисунок 1.27 Скриншот коду метода SaveProgress

```
134         private void SwitchDayTime()
135         {
136             Vector3 rotation = _isDay ? sunEndRotation : _sunStartRotation;
137             float intensity = _isDay ? sunEndIntensity : _sunStartIntensity;
138
139             Sequence seq = DOTween.Sequence();
140             seq.Append(sun.DOIntensity(intensity, sunCycleDuration)).Join(sun.transform
141                 .DOLocalRotate(rotation, sunCycleDuration)).AppendCallback(() =>
142             {
143                 if (_isDay)
144                 {
145                     LoadWave();
146                 }
147
148                 _isDay = !_isDay;
149             });
150         }
```

Рисунок 1.28 Скриншот коду метода SwitchDayTime

Зробимо скрипт **PlayerControl** для обробки управлінням та фізикою гравця. Додамо посилання на фізичні компоненти гравця, змінні зі значеннями для швидкості гравця, висоти стрибка, тощо... (рисунок 1.29)

Кожен кадр як працює скрипт ми зчитуємо клавіши клавіатури та миші задля можливості управління персонажем. При натисканні окремих кнопок виконується відповідна дія. Також у цьому скрипті ми реалізуємо перевірку на те, чи стоїть на землі гравець та взаємодію з іншими об’єктами.

```
public class PlayerControl : MonoBehaviour
{
    [SerializeField] private PlayerBase player;
    [SerializeField] private Transform orientation;
    [SerializeField] private CharacterController controller;
    [SerializeField] private float speed;
    [SerializeField] private float busySpeed;
    [SerializeField] [Range(0f, 0.5f)] private float moveSmoothTime;
    [SerializeField] private float gravity;
    [SerializeField] private float jumpCooldown;
    [SerializeField] private float jumpHeight;
    [SerializeField] private LayerMask groundLayer;
    [SerializeField] private Collider collider;
    [SerializeField] private float interactDistance;
    [SerializeField] private LayerMask interactableLayer;

    [SerializeField] private WeaponsHolder weaponsHolder;
```

Рисунок 1.29 Скриншот полів класу PlayerControl

```
47      private void Update()
48      {
49          if (player.IsAlive() && player.IsInGame())
50          {
51              GroundChecker();
52              ReadInput();
53              PlayerMovement();
54          }
55      }
```

Рисунок 1.30 Скриншот коду системного метода Update

```
77      private void ReadInput()
78      {
79          _horizontalInput = Input.GetAxisRaw("Horizontal");
80          _verticalInput = Input.GetAxisRaw("Vertical");
81
82          if (Input.GetKey(KeyCode.Space) && _isGrounded && _canJump)
83          {
84              StartCoroutine(JumpCoroutine());
85          }
86
87          if (Input.GetKey(KeyCode.Mouse0))
88          {
89              weaponsHolder.WeaponAttack();
90          }
91
92          if (Input.GetKeyDown(KeyCode.Mouse1))
93          {
94              weaponsHolder.WeaponBlock();
95          }
96
97          if (Input.GetKeyDown(KeyCode.E))
98          {
99              TryToInteract();
100         }
101     }
```

Рисунок 1.31 Скриншот коду метода ReadInput

```
124     private void PlayerMovement()
125     {
126         Vector2 targetDir = new Vector2(_horizontalInput, _verticalInput);
127         targetDir.Normalize();
128         _currentDirection = Vector2.SmoothDamp(_currentDirection, targetDir,
129             ref _currentDirectionVelocity, moveSmoothTime);
130         _velocityY += gravity * 2f * Time.deltaTime;
131         Vector3 velocity = (transform.forward * _currentDirection.y
132             + transform.right * _currentDirection.x) * speed + Vector3.up * _velocityY;
133         controller.Move(velocity * Time.deltaTime);
134     }
```

Рисунок 1.32 Скриншот коду метода PlayerMovement

```
67      private void GroundChecker()
68      {
69          bool oldState = _isGrounded;
70          _isGrounded = Physics.Raycast(collider.bounds.center, Vector3.down,
71              collider.bounds.size.y * 0.5f + 0.2f, groundLayer);
72          if (!_isGrounded && controller.velocity.y < -1f)
73          {
74              _velocityY = -8f;
75          }
76      }
```

Рисунок 1.33 Скриншот коду метода GroundChecker

Основний скрипт гравця назвемо **PlayerBase**. У ньому додамо посилання на скрипти управління, здоровья, гаманця та фізичних компонентів.

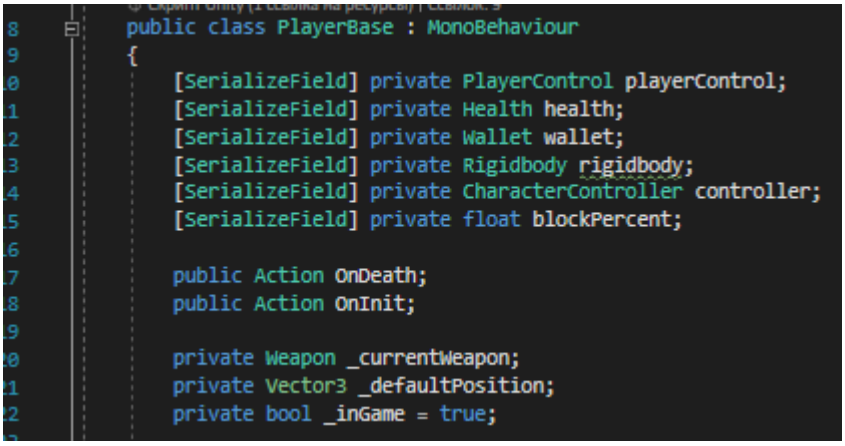



Рисунок 1.34 Скриншот полів класу PlayerBase

У цьому скрипті ми оброблюємо взаємодію між гравцем та іншими скриптами. Для того щоб гравець отримав шкоду від ворогів викликається метод GetDamage у який передається шкода ворога. Якщо гравець встиг у час отримання шкоди активувати блок то отримана шкода зменшується.

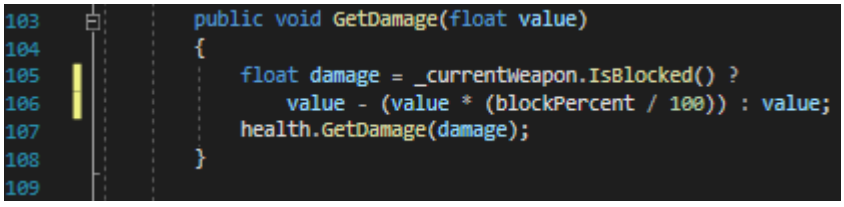



Рисунок 1.35 Скриншот коду метода GetDamage

Для усіх NPC створимо загальний базовий клас **NPCBase**. Успадкування є однією з основних особливостей об’єктно-орієнтованих мов програмувань, якою є мова C#. Завдяки цьому ми можемо створити поля та методи, які будуть доступні до виконання та модифікації у класах, які успадковуються від базового класу. Додамо посилання на фізичні та інші компоненти, характеристики NPC та вкажемо його тип.

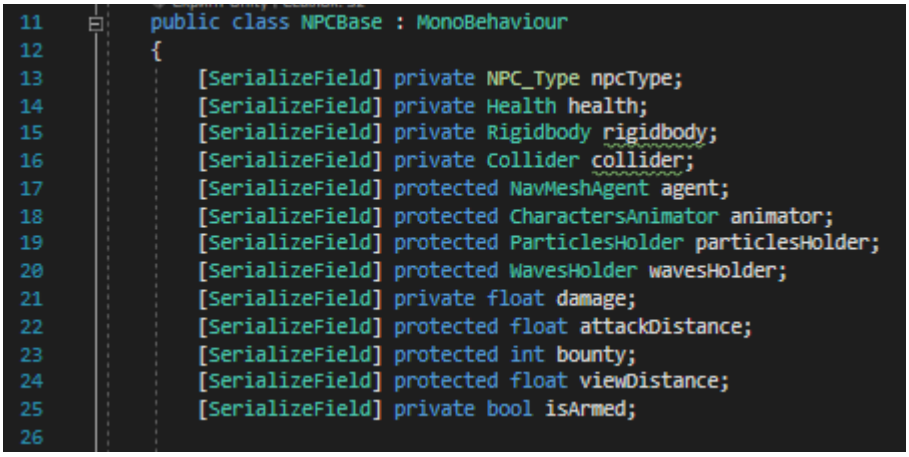



Рисунок 1.36 Скриншот полів класу NPCBase

Реалізуємо загальні методи, які будуть використовувати декілька успадкованих класів. Такі класи можуть використовувати ці методи, якщо є доступ, та модифікувати їх. Розглянемо корутину FollowCoroutine. Модифікатор доступу *protected* надає доступ до цієї корутини у цьому та успадкованих класах. Корутина при виклику отримає посилання на компонент Transform якогось об'єкт. Доки дистанція між NPC та цілью більша за дистанцію атаки NPC він рухається у сторону до цілі. Коли NPC дійде до цілі він намагається вдарити ціль. Цю корутину використовуватимуть усі NPC які є ворогами та будуть переслідувати гравця або мешканців села.

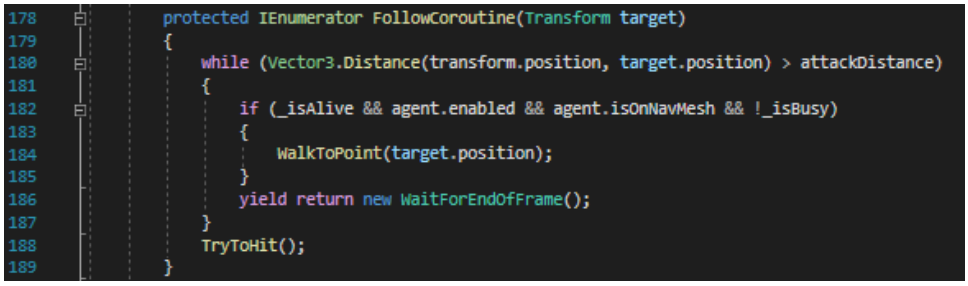



Рисунок 1.37 Скриншот код корутини FollowCoroutine

Створимо скрипт для зброї. Він буде зберігати інформацію про шкоду, шанс критичного удару, кількість шкоди при ньому та посилання на аніматор і детектор колізії. При старті скрипт підписується на подію детектора, яка

передає до метода OnTargetHit посилання на NPC ціль. Ворогу наноситься шкода, яка рахується за формулою.

```
68      private void OnTargetHit(NPCBase target)
69      {
70          float damageToDeal = CalculateDamage();
71          target.GetDamage(damageToDeal);
72      }
73
        ссылка: 1
74      private float CalculateDamage()
75      {
76          float DealDamage = 0;
77          int luckyNumber = Random.Range(0, 101);
78
79          if (luckyNumber <= _criticalDamageChance)
80          {
81              DealDamage = (_damage * 0.01f) * _criticalDamagePercente;
82          }
83          else
84          {
85              DealDamage = _damage;
86          }
87
88          return DealDamage;
89      }
```

Рисунок 1.38 Скриншот коду методів класу Weapon

Усі характеристики зброї зберігаються у конфігі **WeaponsConfig**. Зберігаються значення унікального номеру, назви зброї, малюнку зброї, вартості покупки зброї, шкоди, шансу критичної шкоди, критичної шкоди, стану використання. На початку роботи скрипту **WeaponsHolder** зброя отримує значення для локальних полей із конфігу.

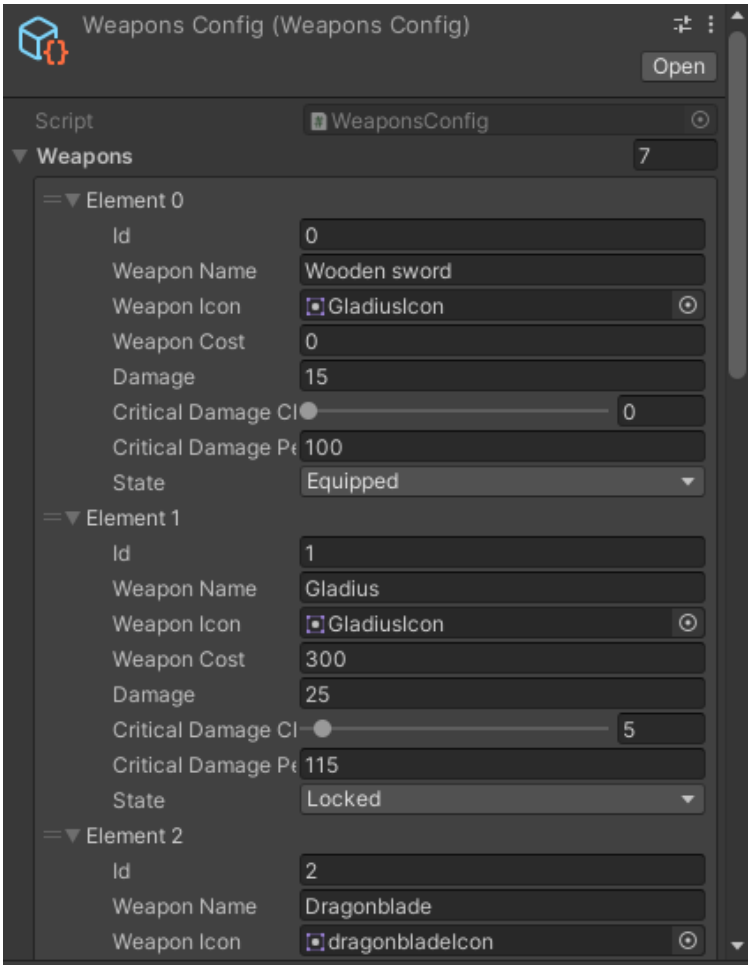



Рисунок 1.39 Вигляд конфігу у інспекторі Unity

### 1.3.3 Графічна реалізація

Для графічної реалізації проекту довелося завантажити 2D та 3D моделі з офіційного онлайн-ринку ресурсів проекту Assets store.

Для NPC були завантажені наступні моделі:

Рисунок 1.40 3D модель мешканця села

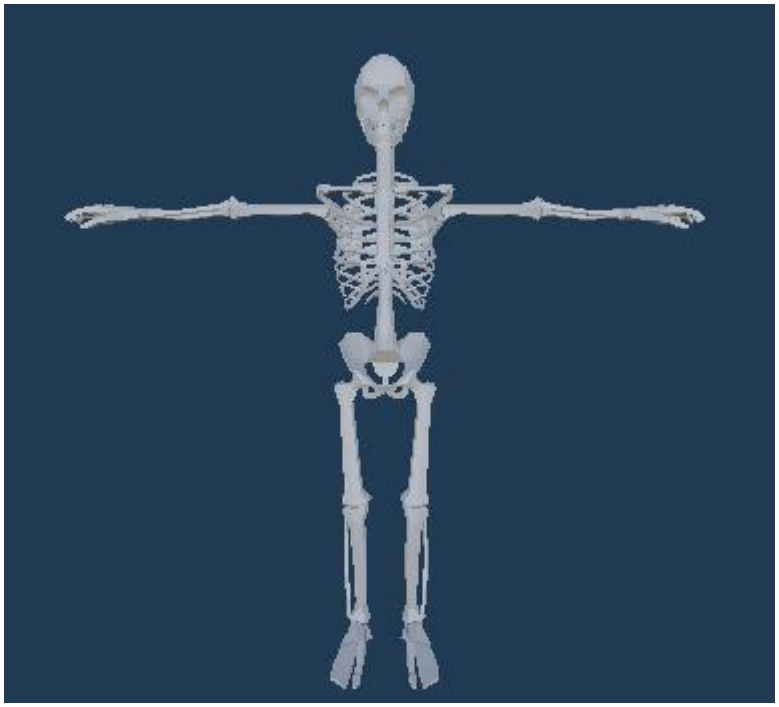


Рисунок 1.41 3D модель звичайного ворога

Рисунок 1.42 3D модель жреця

Рисунок 1.43 3D модель чумного лікаря

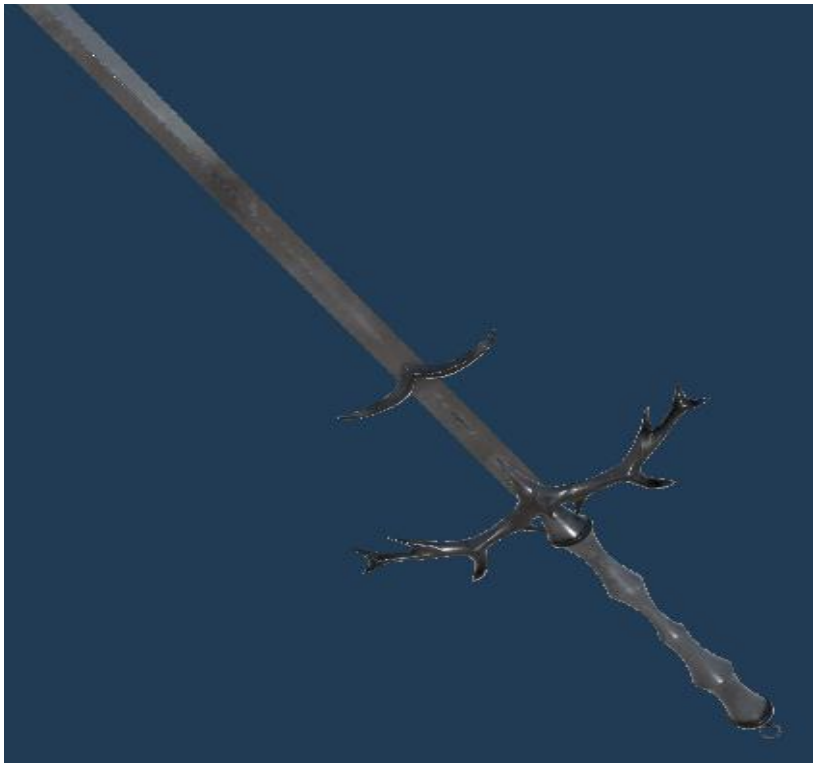


Рисунок 1.44 3D модель зброї

На сцені є крамниця, у якій гравець може придбати зброю. При взаємодії гравця з крамницею з’являється UI вікно крамниці.

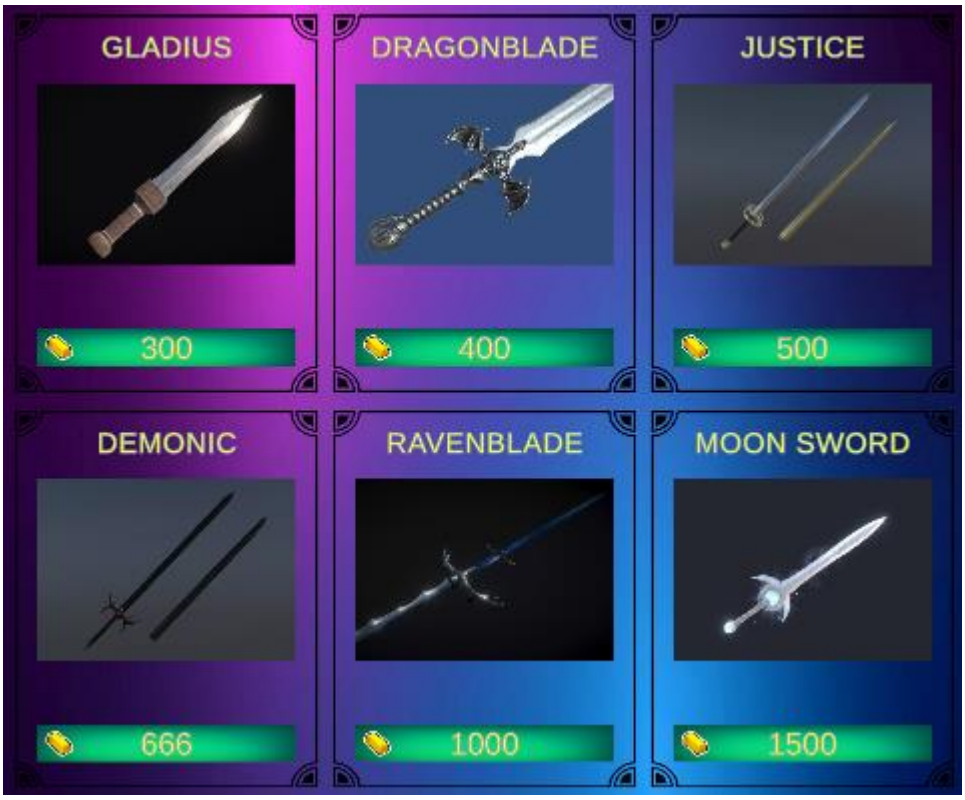



Рисунок 1.45 Вигляд інтерфейсу крамниці

Вікно крамниці - UI об’єкт, який знаходиться на канвасі, який в свою чергу знаходиться поверх екрану. Вікно містить 6 комірок зі зброєю та її інформацією. Зробимо об’єкт комірки та зробимо з нього префаб. У комірці розмістимо текст з назвою зброї, її стан чи вартість покупки та малюнок.

Зробимо скрипт для комірки, який буде зберігати її унікальний номер та посилання на об’єкти які використовують інформацію зброї. При відкритті вікна скрипт передає інформацію про зброю до кожної комірці.


Рисунок 1.46 Вигляд префабу комірки

```
40      private void UpdateItemInfo(ShopItem item)
41      {
42          WeaponInfo weapon = weaponsConfig.weapons
43              .Find(w => w.id == item.GetId());
44          item.SetInfo(weapon.weaponName, weapon.weaponIcon,
45              weapon.weaponCost, weapon.state);
46      }
```

Рисунок 1.47 Скриншот коду метода UpdateItemInfo

### 1.3.4 Тестування та оптимізація гри

Тестування є важливим етапом розробки, який дозволяє виявити та усунути помилки, недоліки та некоректну роботу застосунку перед його випуском. Unity надає розробникам потужні інструменти для автоматизації тестування, що спрощує процес виявлення та виправлення помилок. Розробники можуть створювати тестові сценарії, проводити модульні, інтеграційні та регресійні тести, а також використовувати інструменти для профілювання та аналізу продуктивності.

Оптимізація є ще однією важливою частиною розробки застосунків з використанням Unity. Це процес, який має на меті покращити продуктивність та ефективність роботи застосунку. Unity надає розробникам інструменти та підходи для оптимізації графіки, фізики, алгоритмів та іншого функціоналу. Застосування оптимізаційних методів дозволяє зменшити навантаження на процесор, покращити швидкість відгуку застосунку, забезпечити плавну роботу та забезпечити задоволення користувачів.

Одним з важливих аспектів оптимізації роботи застосунків, розроблених з використанням Unity, є використання пулів об'єктів. Пул об'єктів - це техніка, яка дозволяє попередньо створювати та управляти певною кількістю об'єктів у пам'яті, щоб уникнути накладних витрат на створення та знищення об'єктів під час роботи застосунку.

У Unity, пули об'єктів можуть бути використані для оптимізації ресурсоємних операцій, таких як створення, знищення та активація об'єктів у грі. Замість того, щоб створювати нові об'єкти в реальному часі, розробники можуть попередньо створити пул об'єктів заздалегідь та управляти ними під час роботи застосунку.

Це дозволяє значно зменшити навантаження на систему та забезпечити плавну та ефективну роботу застосунку. Використання пулів об'єктів допомагає уникнути непотрібних затрат пам'яті та обчислювальних ресурсів, забезпечуючи кращу продуктивність та швидкодію.

Profiler є вбудованим інструментом в Unity, який дозволяє аналізувати продуктивність гри та виявляти можливі проблеми. Він надає детальну інформацію про використання CPU, GPU, пам'яті та інших ресурсів під час виконання гри. З допомогою Profiler можна виявити місця зайвого споживання ресурсів, довготривалі операції, перевантаження та інші проблеми, що впливають на продуктивність гри.

Профайлер Unity надає графічне представлення результатів аналізу, включаючи графіки, діаграми та статистику. Він дозволяє відстежувати час виконання окремих функцій, виділяти гарячі точки (hotspots) та отримувати розширену інформацію про шляхи виклику функцій.

Використання Profiler дозволяє виявити та виправити проблеми продуктивності, такі як зайве використання CPU, перевищення лімітів пам'яті, зависання та інші проблеми, що можуть призводити до низької швидкодії гри або недостатньої реагування.

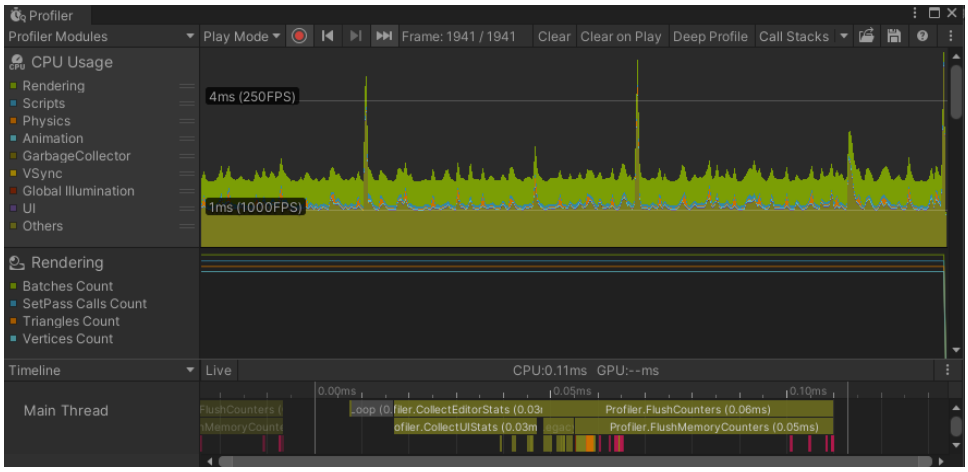



Рисунок 1.48 Зовнішній вигляд інструменту Profiler

З розробленим ігровим проектом можна ознайомитись за посиланням – [https://github.com/Jidkiy88/VillageDefence](https://github.com/Jidkiy88/VillageDefence)

# 2. ОХОРОНА ПРАЦІ

Охорона праці є невід'ємною складовою сучасного виробництва і має вирішальне значення для забезпечення безпеки та здоров'я працівників. Розробка заходів з охорони праці у дипломному проекті є необхідною, оскільки вона спрямована на покращення умов праці та запобігання можливим ризикам та нещасним випадкам.

Основні завдання з охорони праці випливають з Конституції України, Закону України про охорону праці, основ трудового законодавства та законодавчих актів. Вони включають забезпечення безпеки та здоров'я працівників, запобігання професійним захворюванням, визначення нормативів і вимог щодо умов праці, регулювання питань щодо організації та здійснення контролю за дотриманням правил охорони праці.

Для аналізу умов праці відповідно до кваліфікаційної роботи бакалавра було обрано робоче місце оператора ПК на підприємстві. Воно є актуальним, оскільки все більше людей працюють з використанням комп'ютерів та інших електронних пристроїв.

## 2.1 Аналіз та безпека умов праці працівника на робочому місці

### 2.1.1 Організація робочого місця

Для аналізу умов праці та оцінки шкідливих та небезпечних факторів виробничого середовища, що можуть мати місце під час експлуатації устаткування, необхідно враховувати фізичні, хімічні, біологічні та психофізіологічні аспекти.

Фізичні фактори:

· Неправильна організація робочого місця може призводити до незручної позиції тіла, неправильної постави, що сприяє розвитку м'язово-скелетних захворювань.

· Відсутність адекватного освітлення може спричинити зорове напруження та погіршення зору.

· Недостатня аерація та провітрювання приміщення можуть спричинити збільшення концентрації шкідливих речовин у повітрі та погіршення якості вдихуваного повітря.

Психофізіологічні фактори:

· Довга робота за комп'ютером може призводити до перенапруження зору та розвитку синдрому комп'ютерного зору.

· Відсутність відповідного відпочинку та перерви на роботу може призвести до збільшення психологічного напруження, стресу та виснаження.

Всі елементи робочого місця та їх взаємне розташування повинні відповідати ергономічним вимогам, які враховують характер і особливості трудової діяльності (згідно з ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Робочі місця краще розташовувати так, щоб природне світло падало збоку, переважно зліва. Екран має бути розміщений на оптимальній відстані від очей користувача, що становить 600-700 мм, але не ближче, ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів. Монітор має бути розташований прямо перед працівником, на рівні очей або трохи нижче.

Щоб працівник мав достатньо простору для комфортної роботи, ширина робочого столу повинна бути не менше 120 см, а глибина - не менше 80 см. Висота столу повинна бути від 70 до 75 см, а висота стільця - від 40 до 50 см. Важливо, щоб працівник міг сидіти зі спиною прямою, а підлокітники стільця забезпечували підтримку для плечей та зап'ястків.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до працюючого. Конструкція клавіатури повинна передбачати опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-150.

Комп’ютерна миша має бути розташована поруч з клавіатурою на такій висоті, щоб плечі були розслаблені, а зап'ястя не перебували під натиском або в

неприродному положенні. Рекомендована кутова постановка мишки - близько 20-30 градусів.

Працівник повинен мати підтримку для нижньої спини та дотримуватися рівної постави. Рекомендується робити перерви на розтяжку та фізичні вправи кожні 30-60 хвилин роботи за комп'ютером. Під час перерв можна робити прості фізичні вправи, розтяжку м'язів та відпочивання очей.

На один комп'ютер повинно відводиться не менше 6 м2 площі, при цьому об'єм приміщення повинен бути не менше 24 м3.

Фактичні значення: площа робочого місця 7 м2. Об’єм приміщення 120 м3.

### 2.1.2 Вимоги безпеки до мікроклімату виробничих приміщень, освітлення та шуму

Основними документами, які визначають параметри мікроклімату виробничих приміщень, є ДСН 3.3.6.042-99 та ГОСТ 12.1.005-88. Ці параметри регулюються для робочої зони - визначеного простору, де знаходяться робочі місця. Згідно з нормативним документом ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Температура повітря в приміщенні має бути 21-26 °С. Рекомендована вологість повітря становить 40-60%.

Робочі приміщення повинні бути оснащені достатнім природним та штучним освітленням, щоб забезпечити достатню якість та кількість світла на робочих місцях. Природне освітлення здійснюється через вікна, орієнтовані переважно на схід. Штучне освітлення в приміщені здійснюється системою загального рівномірного освітлення. Рівень освітлення на робочому місці за комп’ютером має бути приблизно в діапазоні від 300 до 500 люкс для забезпечення комфортного та ефективного робочого середовища. Освітлення повинно бути розташоване таким чином, щоб уникнути блискіття на робочій поверхні та утворення непотрібних тіней.

Основне джерело шуму є кондиціонер влітку. Шум має 38 Дб, при роботі за комп’ютером шум не заважає.

Для запобігання виникнення інших шумів у відповідності з ГОСТ 12.1.029-80 зниження шуму й вібрації в приміщенні передбаченні звукоізоляція вікон та дверей.

## 2.2 Пожежна безпека

Вибухо- та пожежонебезпечні речовини і матеріали на робочому місці та в приміщенні: електричні розетки та перемикачі світла, комп'ютер та його компоненти, кондиціонер, принтер, різні електричні пристрої та зарядні пристрої.

Вибухо- та пожежонебезпечні характеристики:

Електричні розетки та перемикачі світла: коротке замикання може виникнути через пошкодження проводки, вологу або перевантаження мережі. Загорання може виникнути внаслідок перегріву проводки або іскріння.

Комп'ютер та його компоненти: перегрів може виникнути через несправність системи охолодження, пил або перевантаження компонентів. Загорання може виникнути внаслідок короткого замикання або перегріву компонентів.

Кондиціонер: перегрів може виникнути через несправність системи охолодження, забруднення фільтрів або неправильне використання. Загорання може виникнути внаслідок короткого замикання або перегріву компресора.

Принтер: погане підключення, несправність електричного кабелю або живлення, перегрів джерела живлення принтера можуть спричинити коротке замикання і виникнення пожежі.

Різні електричні пристрої та зарядні пристрої: перегрів може виникнути через несправність або неправильне використання. Загорання може виникнути внаслідок короткого замикання або перегріву пристроїв.

Аналіз можливих місць і причин загорань і вибухів у приміщенні:

· Неправильне використання або несправність електроприладів

· Перевантаження електричної мережі

· Несправність або пошкодження електропроводки

· Використання неякісних або несумісних зарядних пристроїв

· Недотримання правил експлуатації електронної техніки

Засоби пожежогасіння:

Пінні вогнегасники використовують для гасіння тліючих матеріалів, горючих рідин. Однак, їх не можна застосовувати для гасіння устаткування, що знаходиться під напругою, для гасіння сильно нагрітих або розплавлених речовин, а також речовин, які вступають з водою в хімічну реакцію, що супроводжується інтенсивним виділенням тепла і розбризкуванням горючої суміші. Також пінні вогнегасники не підходять для гасіння речовин, горіння яких проходить без доступу повітря (бавовна, піроксилін і тому подібне), горючих металів (натрій, магній). Ще один недолік пінних вогнегасників — вузький температурний діапазон: від +5°С до +50°С.

Порошкові вогнегасники – найпоширеніший тип вогнегасників. Вони оснащені порошком, який при попаданні на джерело пожежі заглушує його. Порошкові вогнегасники підходять для гасіння пожеж класу А, В і С, тобто для гасіння твердих речовин, що горять, рідин і газів. Їх можна використовувати у приміщеннях, на вулиці, на автотранспорті та в інших місцях. Використовується для гасіння пожеж від електроприладів, комп'ютерів, кондиціонерів та інших електроустановок.

Вуглекислотні вогнегасники гасять загорання за рахунок значного охолодження зони горіння струминою вуглекислоти $CO_2$, яка, випаровуючись, перетворюється на вуглекислий газ, який не підтримує горіння. До недоліків вуглекислотних вогнегасників можна віднести можливість обмороження вуглекислотою при необережному їх використанні. Вуглекислотний вогнегасник ефективний для гасіння пожеж класів В та С (рідини та електроустановки). Використовується для гасіння пожеж від електроприладів, комп'ютерів, кондиціонерів та інших електроустановок без пошкодження обладнання.

# ВИСНОВКИ

У моєї кваліфікаційної роботи бакалавра був розроблений ігровий проект на базі ігрового двигуна Unity. Ігровий проект має назву Village Defence.

Розробка ігрового проекту на базі Unity проходила за такими пунктами, як: планування, проектування та ітеративного тестування. Розробник повинен ретельно аналізувати вимоги проекту, розробляти прототипи та поетапно реалізовувати функціональність гри, забезпечуючи зручний та задовільний геймплей.

В процесі розробки ігрового проекту на базі Unity важливо дотримуватися принципів ефективності та оптимізації. Правильна робота з ресурсами, управління пам'яттю та оптимізація швидкодії гри є ключовими факторами успіху.

Успішна розробка ігрового проекту на базі Unity вимагає від розробника творчого мислення, вміння працювати у команді та ефективно вирішувати проблеми, що виникають під час процесу розробки.

Розробка ігрового проекту на базі Unity може бути захоплюючим та задоволенням робочим процесом, а також відкривати можливості для подальшого розвитку у галузі ігрової розробки.

Отже, розробка ігрового проекту на базі Unity є складним, але захоплюючим процесом, який вимагає від розробника глибоких знань і навичок. Unity надає розробникам потужний інструментарій для створення високоякісних ігрових проектів, що можуть бути успішними на ринку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ


1. Unity documentation: [Електронне джерело] - загальна документація роботи з двигуном Unity;
   URL - https://docs.unity3d.com/Manual/index.html
2. Wikipedia: [Електронне джерело] - Skybox;
   URL – https://en.wikipedia.org/wiki/Skybox_(video_games)
3. GameDevAcademy: [Електронне джерело] – Animation;
   URL – https://gamedevacademy.org/unity-animator-tutorial/

# Додаток А

**Слайди мультимедійної презентації**

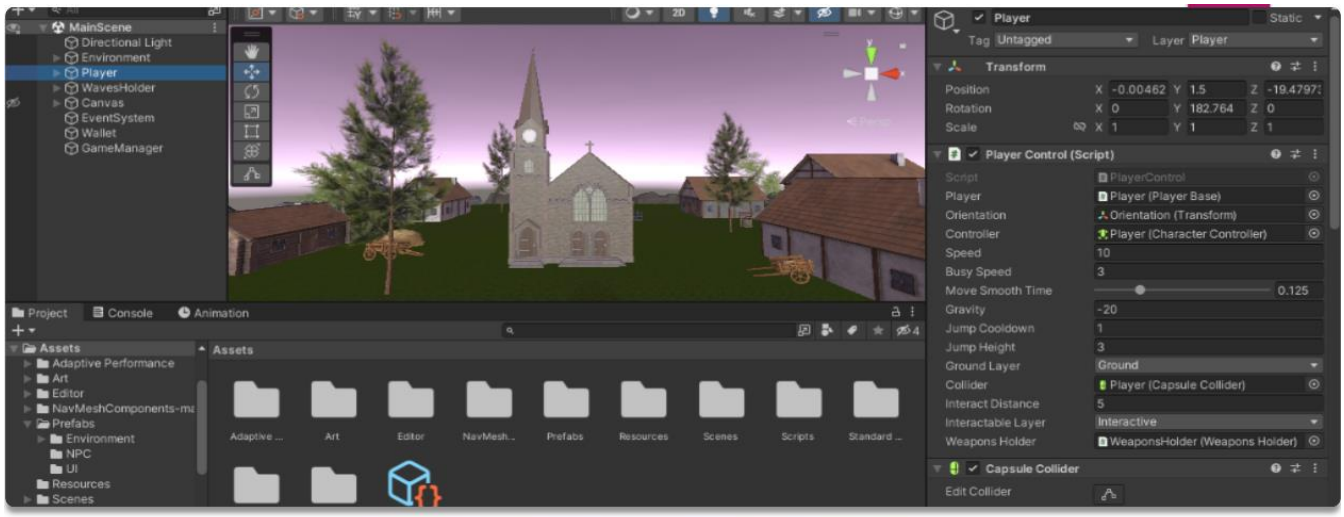
## Технології та інструменти

### Ігрові двигуни:

- Unity
- Unreal Engine
- Godot

### Мови програмування:

- C#
- C++
- GDScript, C#, C++



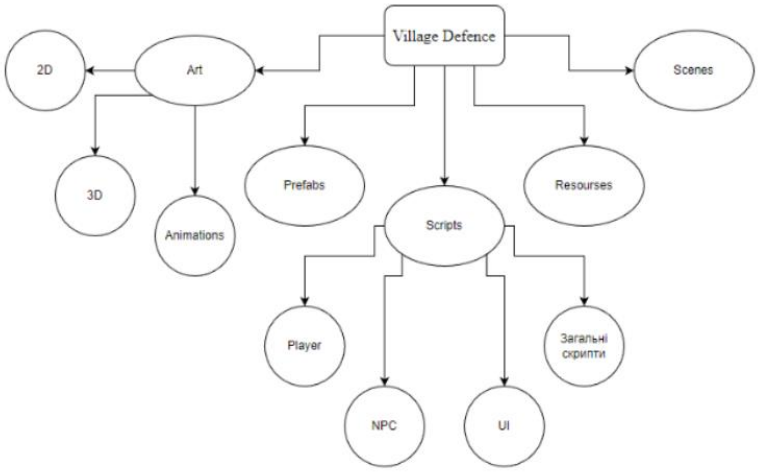

## Редактор Unity

Інструменти та функціонал

# Опис ігрового проекту

- Мета гри
- Механіки
- Геймдизайн

# Структура файлів ігрового проекту




# Архітектура ігрового проекту

- Гравець
- Менеджер гри
- Боти

## Гравець

### PlayerControl

- Управління персонажем

### PlayerBase

- Взаємодія з іншими скриптами
- Основні функції

### Health

- Керування системою здоров'я

## Менеджер гри

- Керування етапами гри
- Взаємодія між скриптами

# Боти

## NPCBase

- Базовий клас усіх NPC

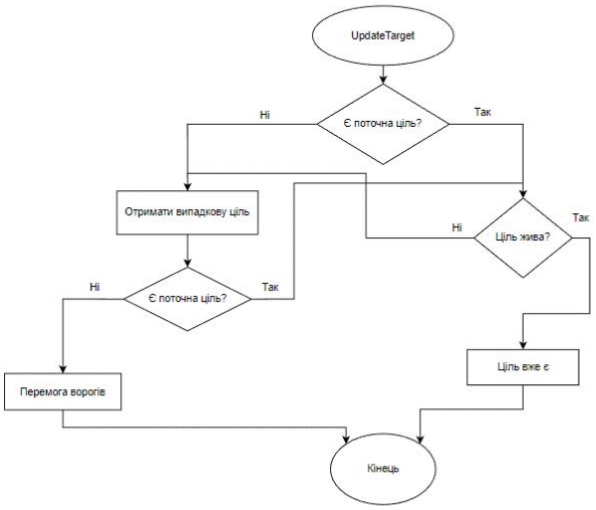
## Власний клас

- Власний клас, який доповнює базову поведінку боту

## Health

- Керування системою здоров'я

# Блок-схема логіки пошуку цілей

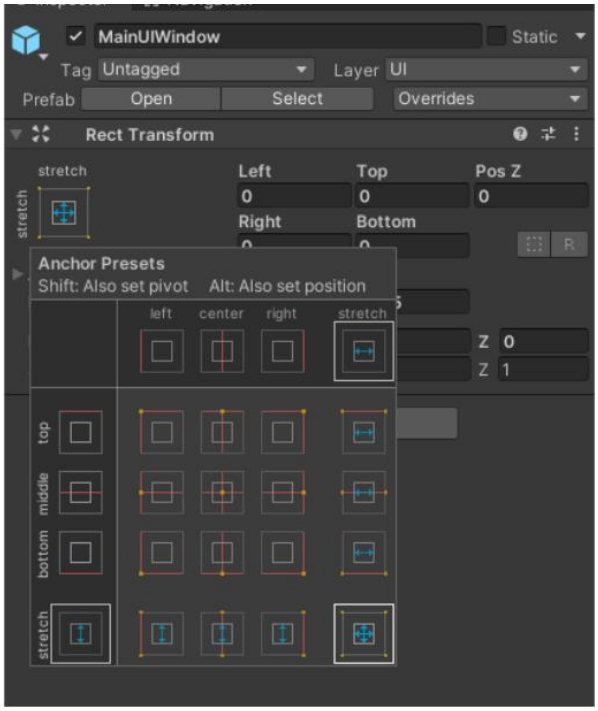


# Процес розробки ігрового проекту

- Бойова система
  - Зміна хвиль
  - Адаптація UI

## Адаптація UI

Адаптація UI (User Interface) у редакторі Unity



**ДОЗВІЛ**
**НА РОЗМІЩЕННЯ**
**ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА**
**В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»**

Ми, що нижче підписалися,

***Волков Микола Андрійович***,
здобувач освіти гр. 2БКС-27, та

***Кунуп Тетяна Василівна***,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту молодшого спеціаліста на тему:

***«Розробка ігрового проекту на платформі Unity» (автор роботи – Волков М.А., керівник роботи – Кунуп Т.В.)***

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Волков М.А. /

Керівник _____ / Кунуп Т.В. /

«15» 06 2023 р.

UNICHECK

Ім'я користувача:
**Наталія Вікторівна Копусь**

ID перевірки:
**1015595541**

Дата перевірки:
14.06.2023 10:04:39 EEST

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

Дата звіту:
14.06.2023 10:24:15 EEST

ID користувача:
**100011688**

Назва документа: **2БКС-27 Волков М.А**

Кількість сторінок: **48** Кількість слів: **5074** Кількість символів: **39864** Розмір файлу: **5.57 MB** ID файлу: **1015244344**

## 2.27%
## Схожість

Найбільша схожість: **0.67%** з Інтернет-джерелом (http://nuczu.edu.ua/sciencearchive/Conferences/sbornik_24_11_16.pdf)

**2.27% Джерела з Інтернету** **425** Сторінка 50

**Не знайдено джерел з Бібліотеки**

## 0% Цитат

**Вилучення цитат вимкнене**

**Вилучення списку бібліографічних посилань вимкнене**

## 0%
## Вилучень

**Немає вилучених джерел**

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ КОЛЕДЖ ОНАХТ»

# РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра
відділення комп'ютерних систем

*Волкову Миколі Андрійовичу*

(прізвище, ім`я та по батькові)

Напрямку підготовки *123 «Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Кунуп Т.В.*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи **« Розробка ігрового проекту на базі ігрового двигуна Unity»**

Обсяг пояснювальної записки 56 сторінок

Обсяг графічної (презентаційної) частини проекту 12 аркушів (слайдів)

## ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаної роботи завданню

Робота відповідає технічному завданню до дипломного проекту. Виконана у відповідності з вимогами.

б) характеристика виконання кожного розділу роботи

Ця кваліфікаційна робота бакалавра присвячена розробці ігрового проекту на базі ігрового двигуна Unity. Ігри є важливим аспектом сучасної розвагової індустрії та мають значний потенціал як забавка, так і навчальний інструмент. Розробка ігрового проекту на базі Unity дозволяє поєднати творчість з технічними навичками, створюючи інтерактивні та захоплюючі віртуальні світи.

в) оцінка якості виконання графічної (презентаційної) частини роботи і пояснювальної записки

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей роботи

У моєї кваліфікаційної роботи бакалавра був розроблений ігровий проект на базі ігрового двигуна Unity. Ігровий проект має назву Village Defence. Розробка ігрового проекту на базі Unity проходила за такими пунктами, як: планування, проектування та ітеративного тестування. Розробник повинен ретельно аналізувати вимоги проекту, розробляти прототипи та поетапно реалізовувати функціональність гри, забезпечуючи зручний та задовільний геймплей

д) основні недоліки роботи

У тексті пояснювальної записки відсутні посилання на використану літературу,

У розділі охорони праці наведені відомі нормативні вимоги загального плану замість конкретних розрахунків освітлення приміщення, вентиляції, рівня шуму.

Оцінка розрахункової частини 5(відмінно)

Оцінка графічної (презентаційної) частини 5 (відмінно)

Загальна оцінка 5(відмінно)

Прізвище, ім'я та по батькові рецензента Василіу Євген Вікторович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ, декан факультету інформаційних технологій та кібербезпеки

«16» 06 2023 р.

(підпис)

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

# ВІДГУК

керівника на кваліфікаційну роботу бакалавра здобувача (здобувачки) освіти відділення комп'ютерних систем

Волкова Микола Андрійовича

(прізвище, ім`я та по батькові)

Спеціальність: 123 «Комп'ютерна інженерія»"

Освітня програма: «Комп'ютерна інженерія»

Тема дипломного проекту: Розробка ігрового проекту на платформі Unity

## ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки)

Кваліфікаційна робота виконано відповідно технічному завданню. Пояснювальна записка містить 51 сторінку. У пояснювальній записці виконано опис етапів розробки структури програми. Графічна частина складається з 12 слайдів мультимедійної презентації, які також містять креслення, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом:

Протягом всього строку роботи над кваліфікаційною роботою та переддипломної практики здобувач освіти Волков М.А. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника

в) теоретична підготовка випускника (випускниці):

Здобувач освіти Волков М.А. під час роботи над кваліфікаційною роботою вивчив достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка здобувача добра і він готовий до захисту кваліфікаційної роботи

г) вміння розв'язувати виробничі та конструкторські питання

Під час роботи над кваліфікаційною роботою здобувач освіти Волков М.А мав змогу самостійно приймати окремі рішення з реалізації ігрового проекту на платформі Unity працювати над поставленим завданням, розробив програмний продукт за допомогою мови програмування C#.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Кунуп Тетяна Василівна

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії,

Підпис

«15» 06 2023 р.