МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: **«РОЗРОБКА ГРИ У ЖАНРІ TOWER DEFENCE»**

Виконав: студент 4-го курсу, групи КН-42
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Дігалевича Павла Олександровича

Керівник: кандидат технічних наук, доцент кафедри ЕММІТ,
Жуковський В.В.

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики та кібербезпеки Донецького національного університету імені Василя Стуса*
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)
Протокол № 11 від «30» травня 2024 р.

Острог, 2024

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*

Кафедра: *економіко-математичного моделювання та інформаційних технологій*

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри ЕММІТ
_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

ЗАВДАННЯ
на кваліфікаційну роботу/проєкт студента
Дігалевича Павла Олександровича

1. Тема роботи: ***Розробка гри у жанрі Tower Defence.***

керівник проєкту: Жуковський В.В., кандидат технічних наук, доцент кафедри ЕММІТ.

Затверджено наказом ректора НаУОА від 03.11 2023 року № 98.

2. Термін здачі студентом закінченої роботи/проєкту: 31 травня 2024 року.

3. Вихідні дані до роботи/проєкту: платформа Unity.

4. Перелік завдань, які належить виконати: головне меню, фонове зображення, фонова музика, система розміщення вежі, система здоров'я, система валюти, вежа отримання валюти, атакуюча вежа, генерація ворогів, логіка руху ворогів, логіка атаки ворогів, механіка програшу, нові вежі, нові вороги та анімації веж.

5. Перелік графічного матеріалу: графічний матеріал застосовується для зображення елементів інтерфейсу Unity та елементів коду.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Жуковський В.В.	01.12.2023	01.12.2023
2	Жуковський В.В.	01.12.2023	01.12.2023
3	Жуковський В.В.	01.12.2023	01.12.2023

7. Дата видачі завдання: 01.12.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Затвердження теми роботи/проєкту	до 31.10.2023р.	
2.	Постановка технічного завдання	до 01.12.2023 р.	
3.	Ознайомлення з документацією	до 10.12..2023 р.	
4.	Написання розділу 1	до 01.02.2024 р.	
5.	Написання розділу 2	до 01.03.2024 р.	
6.	Написання розділу 3	до 01.04.2024 р.	
7.	Тестування системи	до 01.04.2023р.	
8.	Виправлення помилок	до 20.04.2023р.	
9.	Попередній захист кваліфікаційної роботи/проєкту	до 31.05.2023р.	
10.	Здача кваліфікаційної роботи/проєкту на кафедру	до 31.05.2023 р.	

Студент: _____ Павло ДІГАЛЕВИЧ

(підпис)

Керівник кваліфікаційної роботи: _____ ЖУКОВСЬКИЙ В.В.

(підпис)

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: Розробка гри у жанрі Tower Defense.

Автор: Дігалевич Павло Олександрович

Науковий керівник: *Жуковський В.В., кандидат технічних наук, доцент кафедри ЕММІТ.*

Захищена «……»…… 2024 року.

Пояснювальна записка до кваліфікаційної роботи: *54 с., 44 рис., 2 діаг., 31 джерел.*

Ключові слова: *ігровий додаток, гра, розробка, Unity, Tower Defense, game app, game, develop.*

Короткий зміст праці:

Завданням кваліфікаційної роботи/проєкту була розробка ігрового додатка жанру Tower Defense на платформі Unity. Метою цього проєкту було не лише створення функціонального та захоплюючого ігрового продукту, але й набуття практичного досвіду в програмуванні, дизайні та оптимізації ігрових додатків.

Зазначений проєкт створювався для розвитку навичок розробки на платформі Unity на прикладі ігрового додатка у жанрі Tower Defense. У процесі роботи над проєктом використовувалися різні аспекти розробки ігор, включаючи створення ігрових механік, розробку штучного інтелекту ворогів, а також роботу з графікою та анімаціями. Крім того, велика увага приділялася оптимізації продуктивності та забезпеченню стабільної роботи додатка.

Основна розробка включала написання коду, створення ігрових рівнів, розробку систем. Заключний етап, тестування, передбачав виявлення та виправлення помилок, а також оптимізацію гри для забезпечення найкращого користувацького досвіду.

Завдяки роботі над цим проєктом, вдалося значно покращити навички в програмуванні на C#, при роботі з Unity, а також отримати досвід у створенні складних ігрових механік та інтерактивних середовищ.

The task of the qualifying work/project was the development of a game addition of the Tower Defense genre on the Unity platform. The goal of this project was not only to create a functional and exciting game product, but also to have practical experience in programming, design and optimization of game applications.

This project was created to develop development skills on the Unity platform using the example of a game add-on in the Tower Defense genre. The project used various aspects of game development, including the creation of game mechanics, the development of artificial intelligence of enemies, as well as work with graphics and animations. In addition, great attention was paid to optimizing performance and ensuring stable operation.

Main development included writing code, creating game levels, system development. The final stage, testing, predicting and fixing bugs, and optimizing the game to ensure the best user experience.

Thanks to work on this project, it was possible to significantly improve skills in programming in C#, when working with Unity, as well as gain experience in creating complex game mechanisms and interactive environments.

_____

(підпис автора)

Зміст

Вступ

Відеоігри переживають стрімке зростання в індустрії розваг.
Гравець може отримати знання про нові реалії, перевірити свої здібності героїв і подумки випробувати себе в ігровому світі.
З кожним роком ринок ігор відчуває сплеск попиту з новими можливостями як для гравців, так і для розробників.

Tower Defense - один з найпростіших жанрів відеоігор.
Завдяки захоплюючій ігровій механіці цей жанр відомий тим, що в ньому розміщуються вежі та турелі, які можна використовувати для нападу на ворогів, гравці повинні захищати свою базу чи територію.
Щоб покращити свої навички, жанр Tower Defense надає гравцям широкий вибір цікавих і стратегічних можливостей. Для розробки гри Tower Defense потрібне всебічне розуміння програмування графіки та ігрового дизайну.

Темою є реалізація базових механік для гри у жанрі Tower Defence.

Метою є практична реалізація гри у жанрі "Tower Defence" на платформі Unity.

Актуальність цього дослідження визначається інтересом до ігор у жанрі "Tower Defence" серед гравців та популярністю платформи Unity серед розробників ігор. Зараз, швидше за все, є велика кількість потенційних гравців, які б зіграли у гру такого жанру. А також молоді розробники, які прагнуть створювати власні ігри.

Об'єктом дослідження є гра жанру Tower Defence, де гравець захищається від ворогів за допомогою різних веж.

Предметом дослідження є середовище Unity при розробці гри жанру Tower Defence.

Методами дослідження визначено перелік завдань, що виступатимуть у ролі плану виконання роботи**.**

Розділ 1

Інформаційне забезпечення по об’єкту дослідження

1.1 Вступ в жанр Tower Defence:

Огляд жанру Tower Defence:

Tower Defence - це жанр, який базується на стратегічному плануванні оборони. Головною метою гравця є захистити певний об'єкт (зазвичай базу або територію) від наступаючих ворогів, використовуючи вежі або турелі[10].

Історія та популярність:

Жанр Tower Defence вперше набув популярності у ранні часи ігрової індустрії, і відтоді він став одним із найвідоміших жанрів серед гравців по усьому світу. Відомі представники, такі як "Plants vs. Zombies", "Defense Grid: The Awakening" та "Bloons Tower Defence", здобули популярність завдяки захопливому геймплею та викликам, які вони ставили гравцеві.

Основними елементами гри у жанрі Tower Defence є:

- **вежі (турелі):**
  головний засіб оборони гравця, який служить для захисту від ворогів. Вони можуть мати різні види зброї та характеристики.
- **вороги:**
  наступаючі сили, які намагаються знищити об'єкт, який гравець повинен захищати. Вони можуть мати різні характеристики та здатності.
- **шляхи:**
  маршрути, по яких рухаються вороги на шляху до цілі. Гравцю потрібно розташовувати свої вежі так, щоб максимально ефективно знищити ворогів на шляху.
- **ресурси:**
  гроші або ресурси, які гравцю потрібно збирати або заробляти для покупки та покращення веж.

1.2. Наявні аналоги:

При розгляді конкурентного оточення у сфері жанру Tower Defense важливо враховувати, що він вже має певну кількість успішних ігор, які вже здобули популярність серед гравців. Ось деякі з відомих представників жанру Tower Defense на платформі Unity:

- **"Bloons TD"(Рис.1):**
  ця серія ігор від Ninja Kiwi стала однією з найпопулярніших у жанрі "Tower Defense". Графікою та геймплеєм вона вигідно відрізняються від інших ігор цього жанру та надихає багатьох розробників.

Рис.1.1 Геймплей Bloons TD[5]

- **"Kingdom Rush"(Рис.2):**

  ігри з цієї серії від Ironhide Game Studio також здобули популярність завдяки цікавому сюжету, добре пророблені графіці та глибокому геймплею.

Рис.1.2 Геймплей Kingdom Rush[6]

- **"Plants vs. Zombies"(Рис.3):**

  гра від PopCap Games стала однією з найулюбленіших серед гравців завдяки своєму унікальному підходу, де гравці використовують рослини для боротьби з зомбі. Ця гра виділяється своєю яскравою графікою, гумором та цікавим геймплеєм.

Рис.1.3 Геймплей Plants vs Zombies[7]

Після огляду аналогів я вирішив орієнтуватися на геймплей Plants vs Zombies.

1.3. Загальні аспекти створення гри у жанрі Tower Defence:

Створення гри жанру Tower Defence передбачає використання різних механік та компонентів для створення геймплею гри, де гравцеві потрібно оборонятися від ворогів. Ось ключові принципи реалізації Tower Defence :

геймплей та мапа:

- сформуєте геймдизайн, мапу ігрового світу, на якій розташовані шляхи для ворожих одиниць та місця для розміщення об'єктів оборони.

- визначте механіку хвиль ворогів, що нападатимуть на головний об'єкт. Це може включати визначення часових інтервалів між хвилями ворогів і типів ворогів.

вежі та їх розміщення:

- створіть моделі веж їх характеристики, такі як вартість, сила атаки, дальність стрільби тощо.

- додайте систему розміщення веж на мапі. Гравці повинні мати можливість вибирати місця для розміщення веж, і це може вимагати певної взаємодії з інтерфейсом гри.

рух ворогів:

- визначте шляхи, які будуть проходити ворожі одиниці. Зазвичай це будуть певні лінії чи коридори на мапі.

- використовуйте систему шляхів для визначення маршрутів ходи ворогів вздовж заданих трас. Unity надає засоби для створення систем шляхів та контролю над рухом ворогів.

інтерфейс та керування:

- розробіть інтерфейс гри, який буде включати інформацію про гроші гравця, життя, поточні хвилі ворогів, вибрану вежу тощо.

- забезпечте можливість гравцю вибирати вежі для розміщення і покращення та продажу веж.

- додайте елементи керування, такі як кнопки для початку наступної хвилі ворогів.

ефекти та анімація:

- для поліпшення візуального враження додайте анімацію ворогів та веж.

- реалізуйте спеціальні ефекти, такі як вибухи чи анімація у разі завершення гри або програшу.

баланс гри та налаштування:

- тестуйте гру, щоб забезпечити баланс між важкістю та викликом для гравця.

- налаштуйте параметри веж, ворогів та ресурсів, щоб зробити гру цікавішою та збалансованою.

збереження прогресу та рекордів:

- додайте систему збереження прогресу гравця та ведення рекордів, які дозволяють гравцям відновлювати гру та порівнювати свої досягнення.

звуки та музика:

- використовуйте аудіо ресурси для створення звукового супроводу гри, включаючи звуки вибухів, пострілів та музику.

Розділ 2

Інформаційне забезпечення по предмету дослідження

2.1. Unity як платформа для розробки ігор:

короткий огляд:

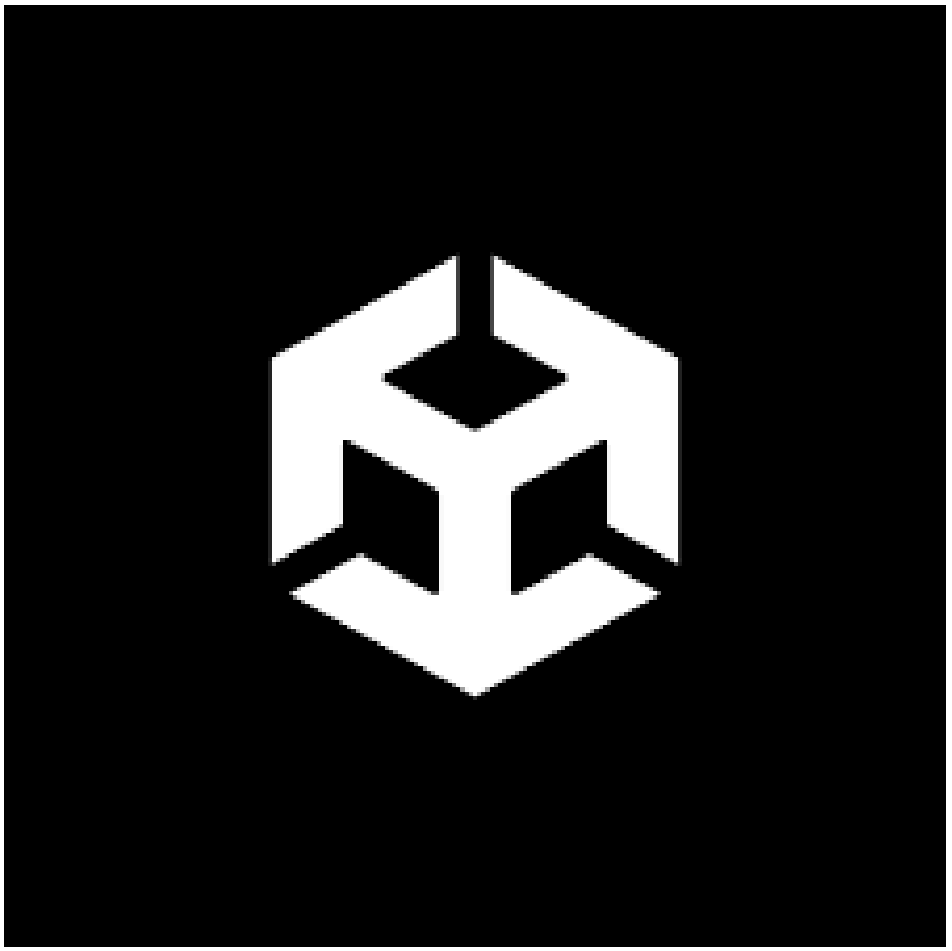


Рис.2.1 Логотип Unity[8]

Unity - це інтегроване середовище розробки (IDE), призначене для створення різнотипних ігор та додатків. Unity(Рис.1) відомий своєю актуальністю і популярністю серед розробників, оскільки цей ігровий двигун надає надзвичайно широкі можливості для розробки ігор на різних платформах[9].

актуальність:

- **багатоплатформність**:

  Unity дозволяє розробникам створювати ігри для різних платформ, таких як ПК, мобільні пристрої, консолі, віртуальна реальність (VR)[18] та інші. Що робить Unity актуальним інструментом, оскільки розробники можуть спрямовувати свої ігри на широку аудиторію[20].

Рис.2.2 Unity[20]

- **спільнота та ресурси:**

  Unity має досить велику спільноту розробників, що робить можливим отримання підтримки, допомоги та доступу до безлічі ресурсів, таких як документація[11], онлайн-курси та форуми.

основні переваги використання:

- **зручний інтерфейс:**

Рис.2.3 Інтерфейс Unity[19]

Unity має інтуїтивний інтерфейс, що дозволяє легко створювати та редагувати ігрові об'єкти та сцени.

- **графічний потенціал**:

Рис.2.4 Приклад графіки[12]

Unity підтримує розробку графічного контенту, включаючи 2D та 3D графіку, а також різні спеціальні ефекти.

Графічні функції Unity дозволяють керувати зовнішнім виглядом вашої програми та легко налаштовуються.

Ви можете використовувати графічні функції Unity для створення красивої, оптимізованої графіки на різних платформах, від мобільних до консолей високого класу та робочих столів.

- **фізика та анімація:**

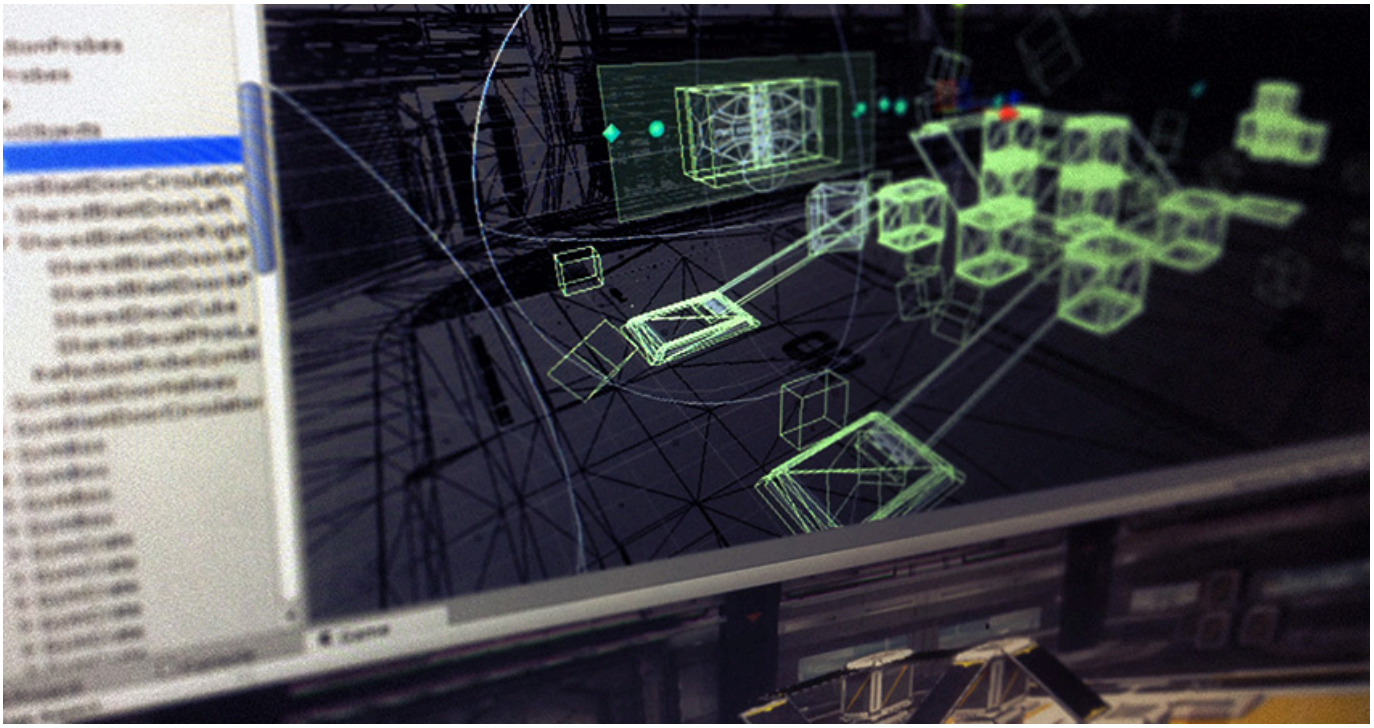


Рис.2.5 Фізика в Unity[13]

Unity надає інструменти для створення реалістичних фізичних ефектів та анімацій об'єктів.

Unity допомагає моделювати фізику у вашому проекті, щоб переконатися, що об’єкти правильно прискорюються та реагують на зіткнення, сила тяжіння та різні інші сили.

Unity надає різні реалізації фізичного механізму , які ви можете використовувати відповідно до потреб свого проекту:

3D, 2D, об’єктно-орієнтований або орієнтований на дані.

- **можливості програмування:**

Рис.2.6 Логотип C#[21]

розробка ігор в Unity відбувається з використанням мови програмування C#[21], що дає розробникам широкий простір для програмування ігрової логіки.

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research.

2.2. Основи розробки ігор на Unity:

Unity містить декілька ключових деталей, що дозволяють розробникам створювати ігри та інтерактивні додатки[11].

Давайте докладніше розглянемо, як працює Unity:

ігрові об'єкти та сцени:

- **ігрові об'єкти:**

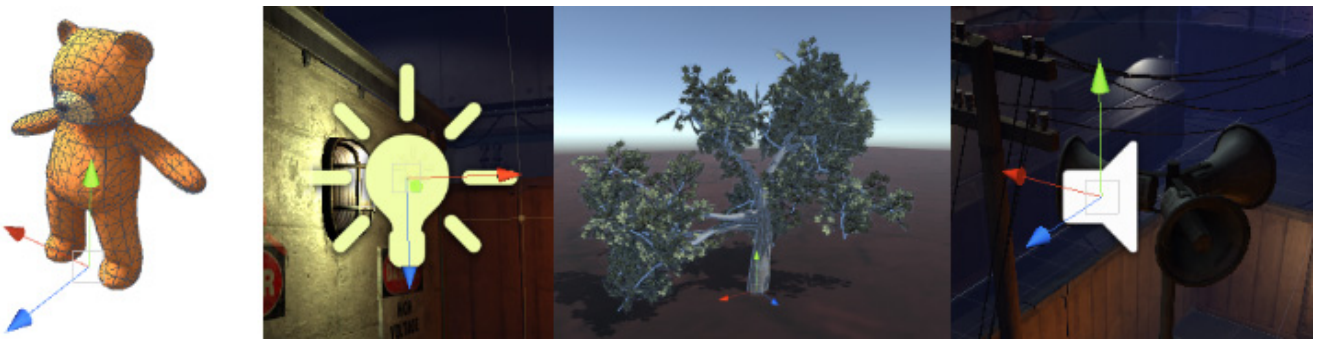


Рис.2.7 Створення ігрових об’єктів[22]

кожен об’єкт у вашій грі є GameObject , від персонажів і колекційних предметів до вогнів, камери та спецефекти. Однак GameObject не може нічого робити сам по собі; вам потрібно надати йому властивості, перш ніж він стане персонажем, середовищем або спеціальним ефектом.

GameObjects — це фундаментальні об’єкти в Unity, які представляють персонажів, реквізит і декорації. Самі по собі вони не досягають багато чого, але діють як контейнери для компонентів , які реалізують функціонал.

Щоб надати ігровому об’єкту властивості, необхідні для того, щоб стати світлом, деревом або камерою, вам потрібно додати до нього компоненти . Залежно від типу об’єкта, який ви хочете створити, ви додаєте різні комбінації компонентів до GameObject. Unity має багато різних типів вбудованих компонентів, і ви також можете створювати власні компоненти

- **сцени**:

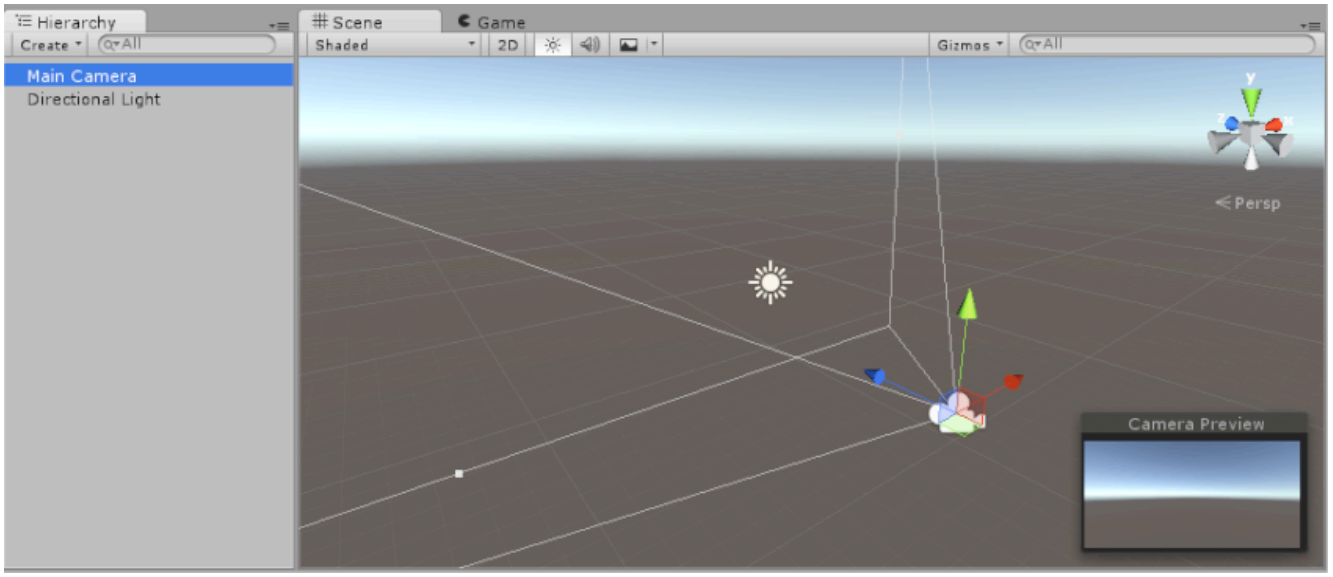


Рис.2.8 Приклад створення сцени[23]

Сцени — це місце, де ви працюєте з вмістом в Unity.
Це ресурси, які містять гру чи програму повністю або частково.

Наприклад, ви можете створити просту гру в одній сцені, тоді як для складнішої гри ви можете використовувати одну сцену на рівень, кожна зі своїм середовищем, персонажами, перешкодами, декораціями та інтерфейс користувача. У проекті можна створити будь-яку кількість сцен.

Коли ви створюєте новий проект і відкриваєте його вперше, Unity відкриває зразок сцени, який містить лише камеру і світло.

компоненти та скрипти:

- **компоненти**:

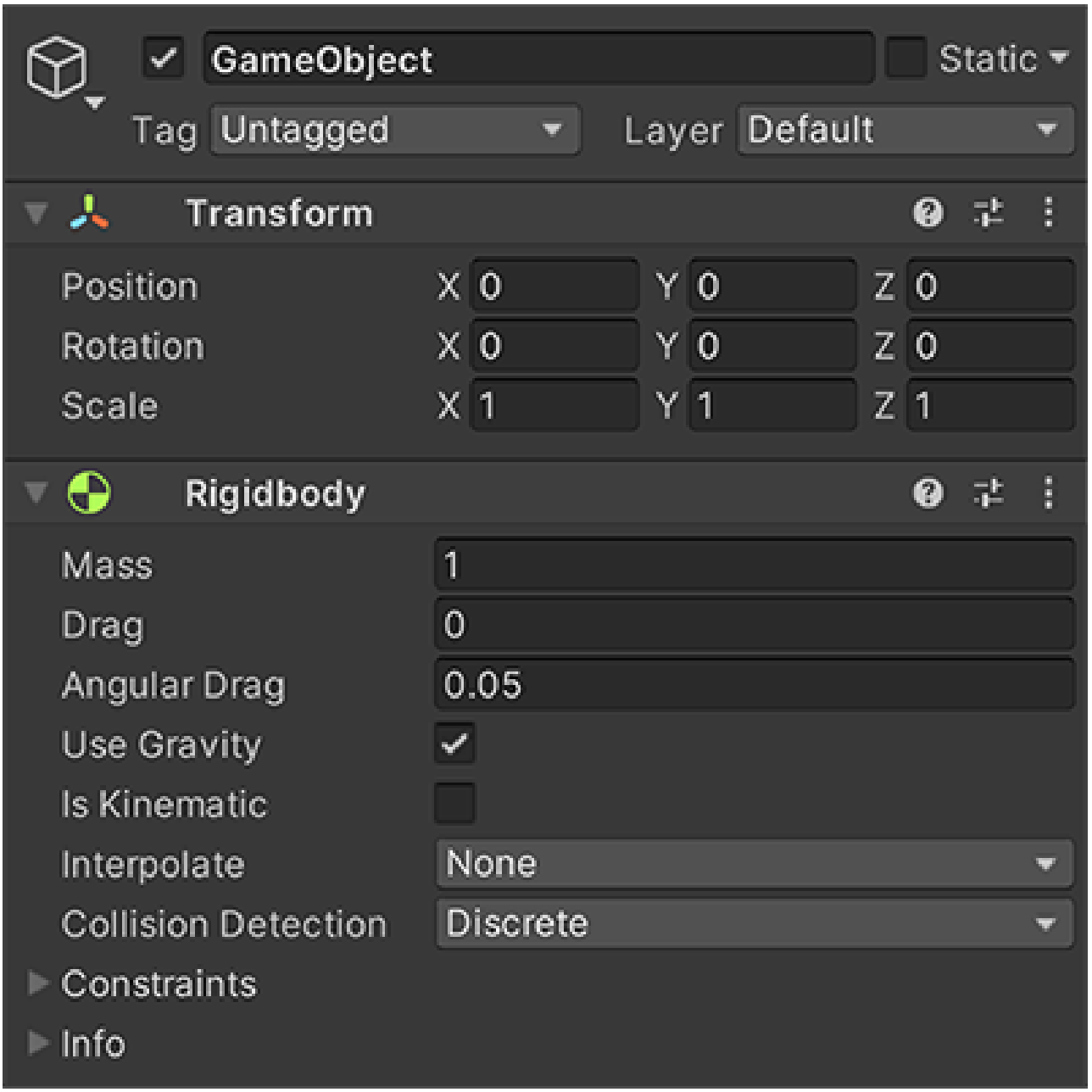


Рис.2.9 Приклад компонентів[24]

компоненти - це будівельні блоки ігрових об'єктів, які визначають їхню поведінку та властивості. Unity має декілька вбудованих компонентів, таких як Transform, Collider, Rigidbody та інші, а також дає розробникам створювати власні компоненти.

- **скрипти**:

Рис.2.10 Робота зі скриптами[14]

розробники використовують скрипти для програмування логіки та поведінки ігрових об'єктів. Скрипти дають можливість створювати ігровий інтелект, керувати анімацією, обробляти дії гравця та багато чого іншого.

фізика та анімація:

- Unity має вбудовану систему фізики, яка дозволяє симулювати рух та зіткнення об'єктів в грі, що робить світ гри реалістичнішим та дозволяє створювати фізичні головоломки та ефекти.

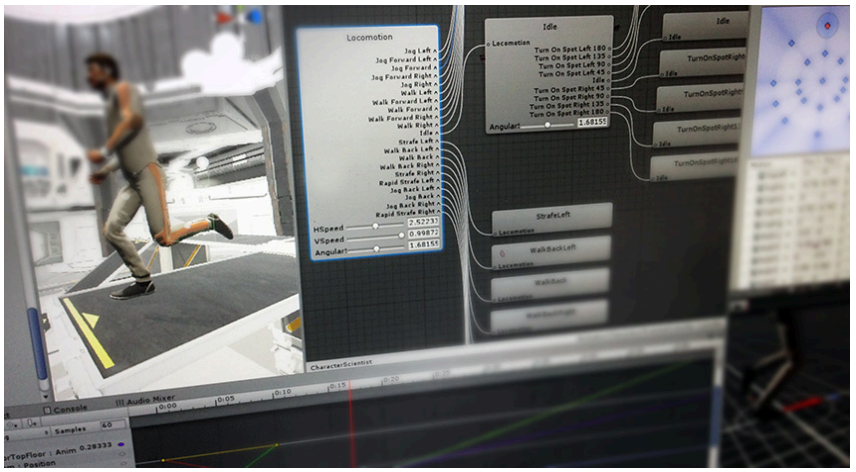


Рис.2.11 Робота з анімаціями в Unity

- Unity також підтримує анімацію об'єктів, де розробники можуть створювати рухи та анімацію для персонажів, об'єктів та інших ігрових елементів.

мультимедіа та ресурси:

Рис.2.12 Звуки в Unity[15]

- Unity дозволяє імпортувати графіку, звуки, відео та інші мультимедійні ресурси для використання в грі.
  Розробники мають можливість налаштувати освітлення, матеріали та текстури для досягнення бажаного стилю гри.

- засоби ресурсів Unity дозволяють організовувати та керувати активами, що спрощує роботу з ресурсами гри.

віртуальна камера:

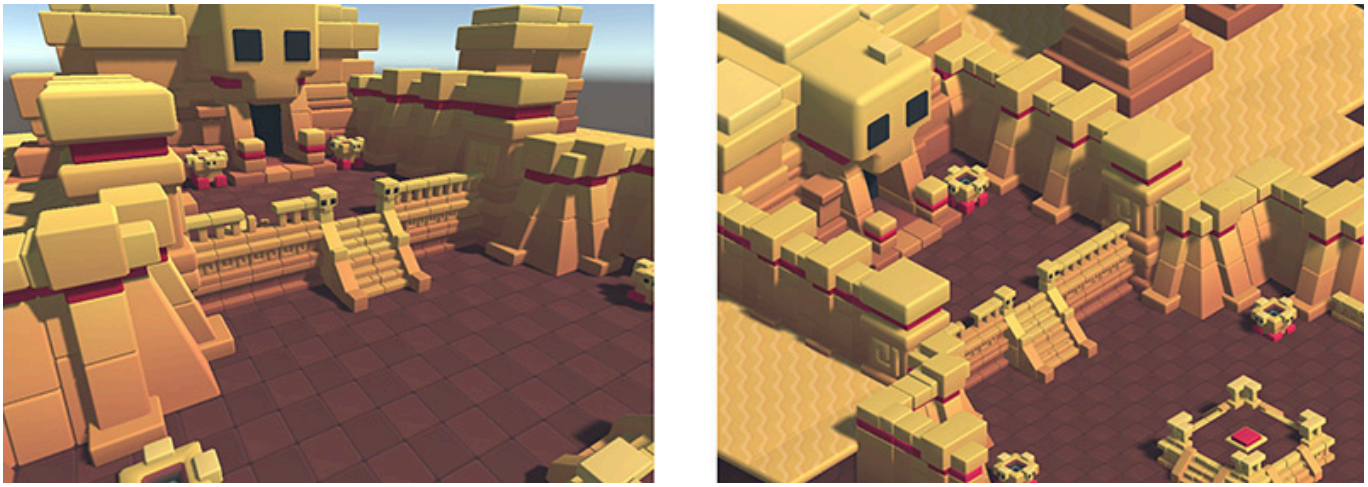


Рис.2.13 Приклади віртуальних камер[25]

в Unity можна налаштовувати камери для відображення гри з різних ракурсів та кутів огляду. Це дозволяє створювати різноманітні ефекти камери, включаючи зум, повороти та зміну фокуса.

Те, що бачить камера, визначається її трансформацією та компонентом Camera. Положення трансформації визначає точку огляду, його вісь вперед (Z) визначає напрямок перегляду, а його вісь і вісь угору (Y) визначають верхню частину екрана.

Параметри компонента «Камера» визначають розмір і форму області, яка потрапляє в область перегляду. З налаштованими цими параметрами камера може відображати те, що вона «бачить» на екрані. У міру того, як GameObject рухається та обертається, відображене зображення рухається та обертається відповідно.

взаємодія та фізика зв'язків:

Рис.2.14 Кнопка(найпростіший засіб взаємодії)[26]

Unity надає засоби для взаємодії ігрових об'єктів, включаючи детектори зіткнень та події, які спрацьовують при певних діях гравця чи ігрових об'єктів. Unity містить потужний рендеринг, що дає можливість створювати візуально захопливі гри з різними стилями графіки.

Більшість компонентів взаємодії мають деякі спільні риси.
Вони доступні для вибору, що означає, що вони мають спільні вбудовані функції для візуалізації переходів між станами (нормальний, виділений, натиснутий, вимкнений) і для переходу до інших вибраних за допомогою клавіатури чи контролера.
Ця спільна функція описана на сторінці «Вибір» .

Компоненти взаємодії мають принаймні одну подію UnityEvent, яка викликається, коли користувач взаємодіє з компонентом певним чином. Система інтерфейсу користувача вловлює та реєструє будь-які винятки, які поширюються поза кодом, приєднаним до UnityEvent.

2.3 Asset Store:

Unity Asset Store є домом для зростаючої бібліотеки безкоштовних і комерційних активів, створених як Unity Technologies, так і членами спільноти. Цей онлайн-ринок надає розробникам можливість швидко та ефективно знаходити й використовувати різноманітні ресурси, які можуть значно прискорити процес розробки.

Доступна велика різноманітність ресурсів, які охоплюють все: від текстур, моделей і анімації до цілих прикладів проектів, навчальних посібників і ресурсів розширення. Розробники можуть знайти текстури для створення реалістичних поверхонь, тривимірні моделі для додавання деталей до своїх сцен, анімації для оживлення персонажів та об'єктів, а також звукові ефекти та музичні композиції для створення атмосфери в їхніх іграх. Окрім цього, доступні скрипти та інші програмні компоненти, які можуть суттєво спростити процес кодування і впровадження різноманітних функцій.

Вміст, доступний в Asset Store, допомагає покращити ваш проект, гру чи програму та зменшити навантаження, необхідне для створення інструментів або моделей з нуля. Використовуючи готові активи, розробники можуть зосередитись на унікальних аспектах своїх проектів, заощаджуючи час і ресурси. Крім того, активи, придбані в Asset Store, часто супроводжуються документацією та прикладами, що полегшує їх інтеграцію та налаштування.

При розробці даного проєкту були використані дані асети:

- асет-пак Sunny Land Forest [1];

- backyard Top-Down Tileset [2];

- асет-пак Pixel Adventure 1 [3];

- асет-пак з музикою Adventure music and SFX [4];

Розділ 3

Процес розробки

3.1. Опис ідеї та вимог до продукту:

Гравець керує групою лісових стражів, які повинні захистити населений пункт від нападів диких звірів, які наближаються з глибин лісу.

Щоб ефективно стримувати напади та захищати населений пункт, гравець повинен розташовувати своїх воїнів на стратегічних позиціях.

Вимоги до продукту:

- головне меню
- фонове зображення
- фонова музика
- система розміщення вежі
- система здоров’я
- система валюти
- вежа отримання валюти
- атакуюча вежа
- спавнер ворогів
- логіка руху ворогів
- логіка атаки ворогів
- механіка програшу

3.2. Меню

Для створення головного меню, як показано на малюнку 1, було вирішено використати асети, кнопки та функції до них.

Для кнопки Start функцію ChangeScene, у скрипті Menu, для переходу між сценами.

Для кнопки Options функцію для зміни видимості вікон.

Для кнопки Quit функцію QuitGame, у скрипті Menu, для завершення виконання програми.

Також фонову музику було додано використовуючи вбудований в Unity компонент Audio Source у який додано у відповідне поле файл звуку.

Рис.3.1 Головне меню*

**: розроблено автором*

Для створення сторінки налаштувань як показано на малюнку 2 було вирішено використати текстові поля, асети, кнопки та функції до неї.

Рис.3.2 Меню Налаштування*

**: розроблено автором*

Для кнопки Back функцію зміни видимості вікон. Також було додано базовий Unity об’єкт – повзунок.

3.3. Геймдизайн

Було створене фонове зображення, додано фонові звуки, як показано на малюнку 3.

Фонове зображення створено за допомогою вбудованої в Unity системи TileMap та раніше знайдених асетів для неї, що дозволяє розставляти асети ніби пензликом з палітри.

Для цього потрібно:

1. Створіть новий об'єкт Tilemap:
    - в ієрархії (Hierarchy) клацніть правою кнопкою миші та виберіть 2D Object > Tilemap > Rectangular.
2. Імпортуйте спрайти:
    - перетягніть ваші спрайти в папку Assets у вікні Project.
3. Розділіть спрайт-лист на окремі тайли:
    - виберіть спрайт-лист в Project.
    - в Inspector змініть Sprite Mode на Multiple.
    - натисніть Sprite Editor та використовуйте інструмент розділення (Slice) для поділу листа на окремі спрайти.
4. Створіть Tile Assets:
    - виберіть спрайт або декілька спрайтів, які ви хочете використовувати як тайли.
    - клацніть правою кнопкою миші на виділених спрайтах і виберіть Create > 2D > Tiles > Rectangular.
    - дайте назву вашому новому Tile Asset та збережіть його.
5. Створіть нову палетку:
    - виберіть Window > 2D > Tile Palette.
    - у вікні Tile Palette натисніть кнопку Create New Palette.
    - введіть назву та натисніть Create.

6. Додайте тайли до палетки:
    - перетягніть Tile Assets з Project у вікно Tile Palette.
    - Unity попросить зберегти ці тайли в певну папку (зазвичай Assets).
7. Малюйте на Tilemap:
    - виберіть тайл з Tile Palette та малюйте на Tilemap в Scene.
    - використовуйте інструменти пензля, гумки та інші для редагування Tilemap.
8. Додайте компонент Tilemap Collider 2D:
    - виберіть об'єкт Tilemap в Hierarchy.
    - натисніть Add Component у вікні Inspector та додайте Tilemap Collider 2D.

Рис.3.3 Геймдизайн*

**: розроблено автором*

Фонові звуки додано за допомогою вбудованого в Unity компоненту Audio Source у який додано у відповідне поле файл звуку.

3.4. Вежі

Префаби[27] веж було створено як показано на малюнку 4 та 5.

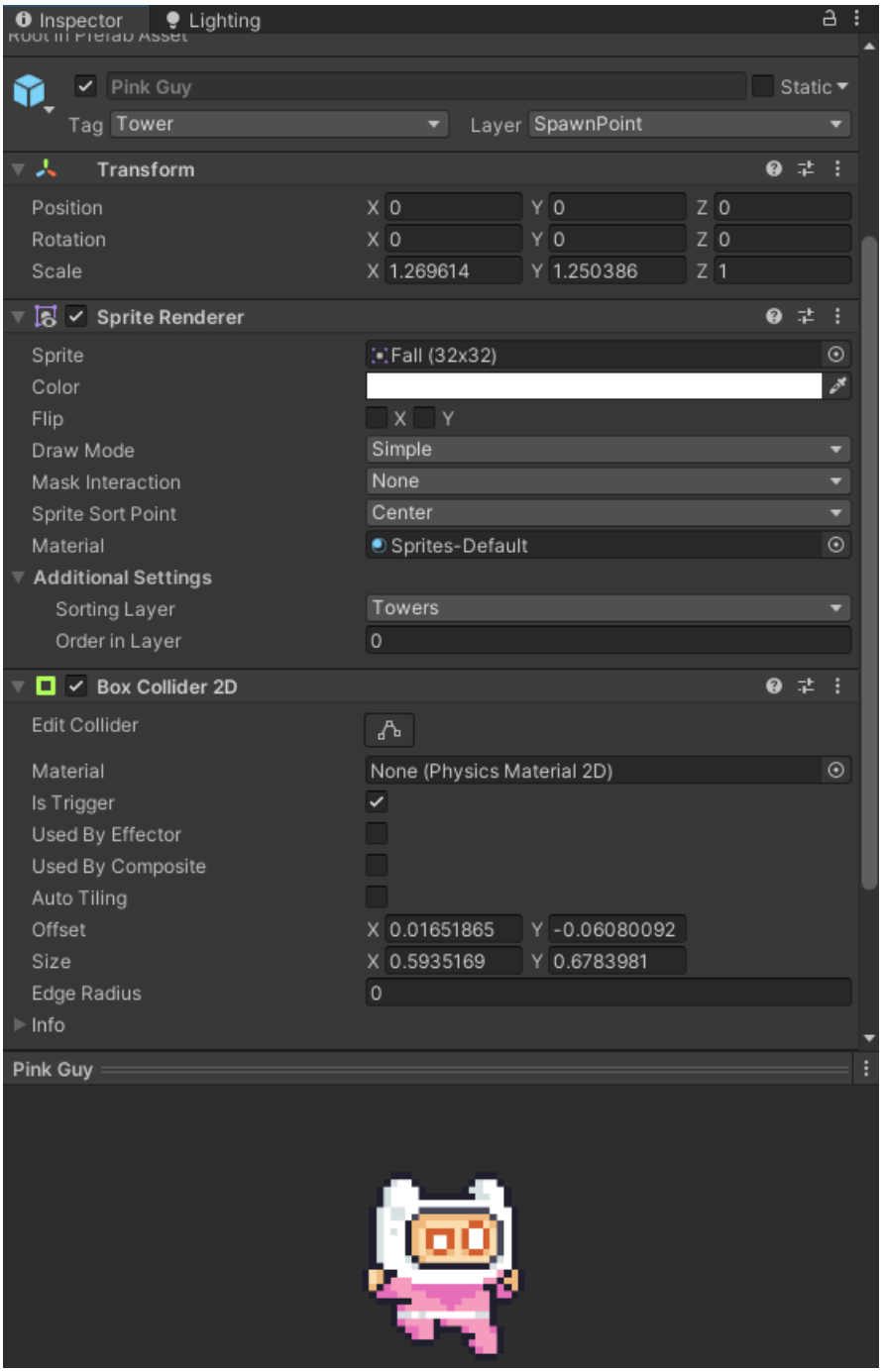


Рис.3.4 Об'єкт Pink Guy*

**: розроблено автором*

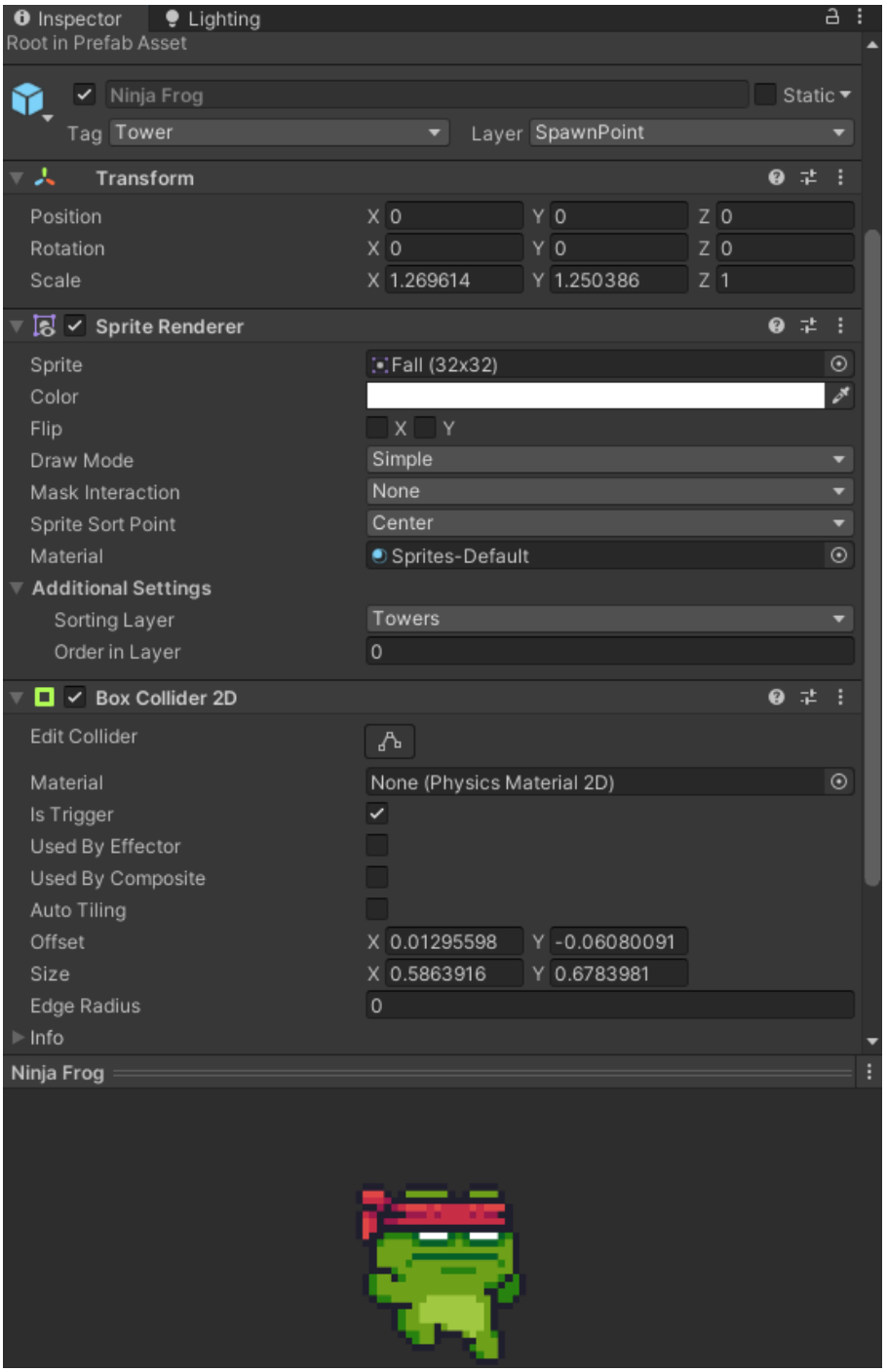


Рис.3.5 Об'єкт Ninja Frog*

**: розроблено автором*

Префаби веж створено за допомогою знайдених раніше спрайтів PinkGuy та NinjaFrog також Unity компоненту Box Collider 2D, що дає їм можливість взаємодіяти з навколишнім середовищем.

Вежа валюти

Розроблено вежу отримання валюти, як показано на малюнках 9 та 10. Вежа отримання валюти реалізована за допомогою знайденого раніше спрайту PinkGuy та скрипту IncomeTower, який рахує певний інтервал часу(5 секунд) і додає певну кількість валюти на баланс гравця.

Рис.3.6 Скільки було*

**: розроблено автором*

Рис.3.7 Скільки стало*

**: розроблено автором*

Також в цей момент над головою PinkGuy з’являється монетка.

Атакуюча вежа

Розроблено атакуючу вежу як показано на малюнку 11.

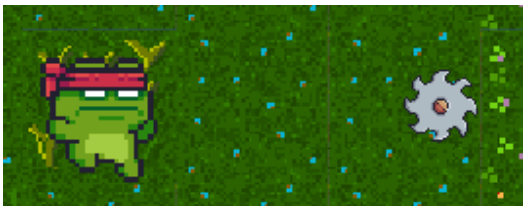


Рис.3.8 Об’єкт Ninja Frog*

**: розроблено автором*

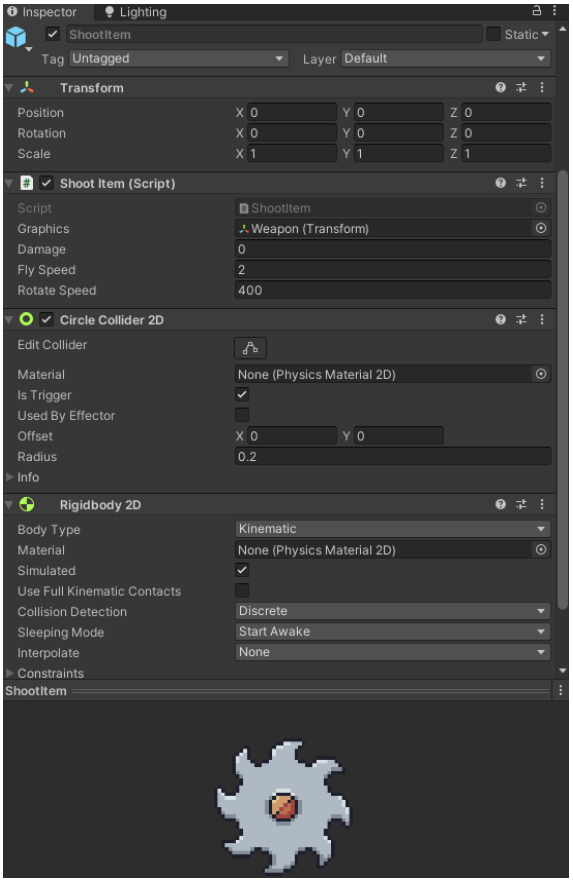


Рис.3.9 Shoot Item*

**: розроблено автором*

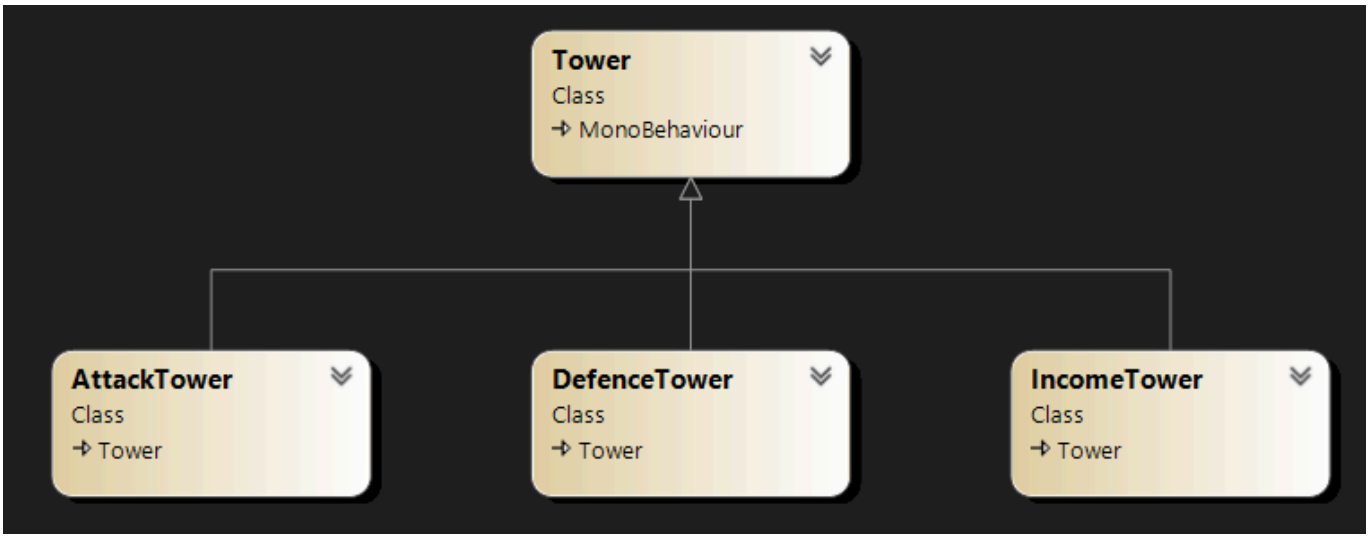
Атакуюча вежа реалізована за допомогою знайденого раніше спрайту NinjaFrog та скрипту AttackTower, який рахує певний інтервал часу, а тоді за певною логікою запускає снаряд(мал.12), який має власний скрипт, який перевіряє контакт з об’єктом, що має тег Enemy, у разі попадання знімає цьому об’єкту показник здоров’я, а якщо здоров’я в об’єкта закінчилося, то снаряд знищує його.

Вежа з базовими параметрами

Логікою створення цієї вежі є концепція ООП[28], а саме наслідування.

Наслідування класів — це концепція об'єктно-орієнтованого програмування (ООП), яка дозволяє одному класу (похідному або дочірньому) успадковувати властивості та методи іншого класу (базового або батьківського). Це дозволяє створювати нові класи на основі існуючих, що сприяє повторному використанню коду та спрощує його підтримку.

Створено базову вежу з базовими параметрами та розроблено механіку атаки ворогів як показано на малюнку 17.



Діаг.3.1 Діаграма залежності від базової вежі*

**: розроблено автором*

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


public class Tower : MonoBehaviour
{
    public int health;
    public int cost;


    public virtual bool LoseHealth(int amount)
    {
        health -= amount;
        if (health == 0)
        {
            Die();
            return true;
        }
        return false;
    }

    public void Die()
    {
        Debug.Log("Tower is dead");
        Destroy(gameObject);
    }
}
```

Рис.3.10 Код базової вежі*

**: розроблено автором*

Базова вежа була реалізована за допомогою перенесення усіх спільних функцій з усіх наявних веж та налаштування наслідування іншими вежами цієї як показано на діаграмі 2.

Захисна вежа

Захисна вежа буде виконувати функцію захисту інших веж.

Основні компоненти захисної вежі це Sprite Renderer, Box Collider 2D, Animator та скрипту Defence Tower, що наслідує загальний скрипт Tower. Скрипт Tower надає такі загальні параметри як Здоров’я та Ціна.

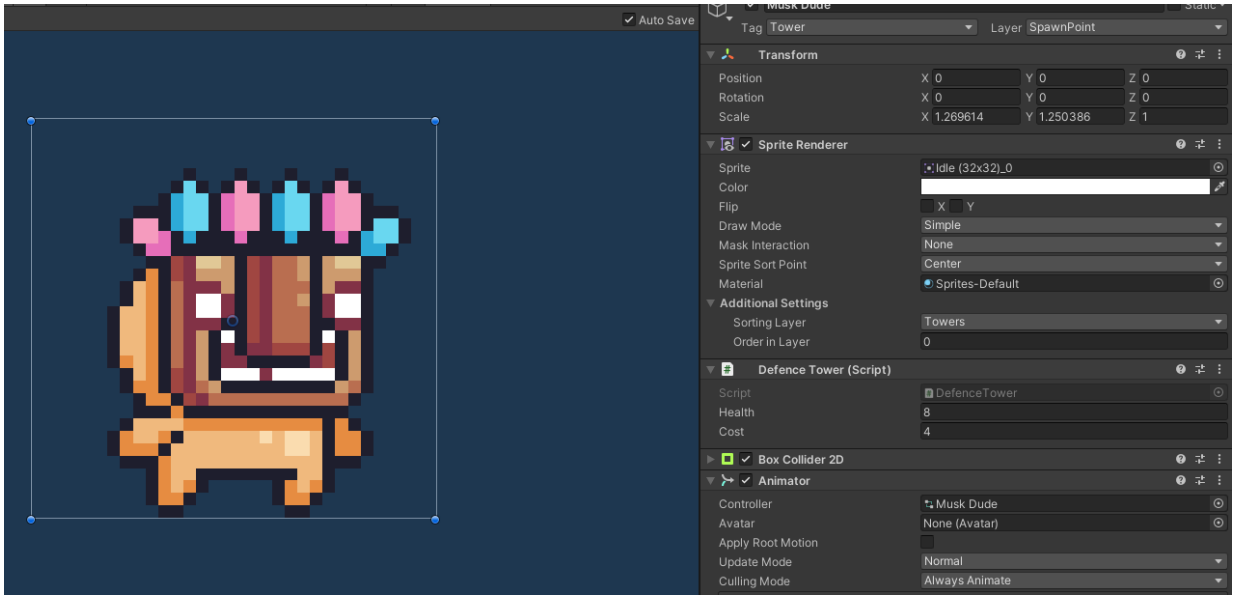


Рис.3.11 Вигляд ігрового об'єкта захисної вежі в редакторі Unity*

**: розроблено автором*

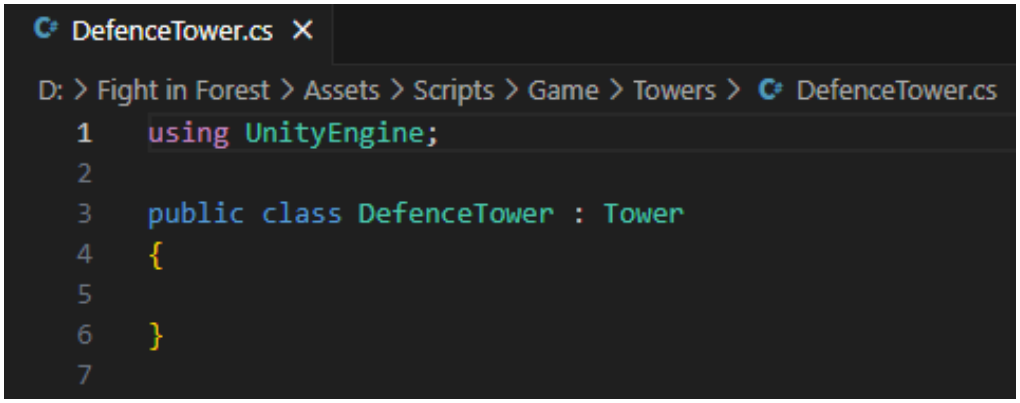


Рис.3.12 Скрипт захисної вежі*

**: розроблено автором*

Скрипт Defence Tower своєю чергою не надає вежі нових властивостей, а потрібен тільки для чіткої ієрархії. Захисна вежа має більше здоров’я(8) ніж інші вежі, але немає атаки. Також вона коштує 4 валюти.

Лазерна вежа

Лазерна вежа буде виконувати функцію захисту населеного пункту поза екраном.

Основні компоненти лазерної вежі це Sprite Renderer, Box Collider 2D, Animator та скрипт Lazer Tower, що наслідує загальний скрипт Tower.

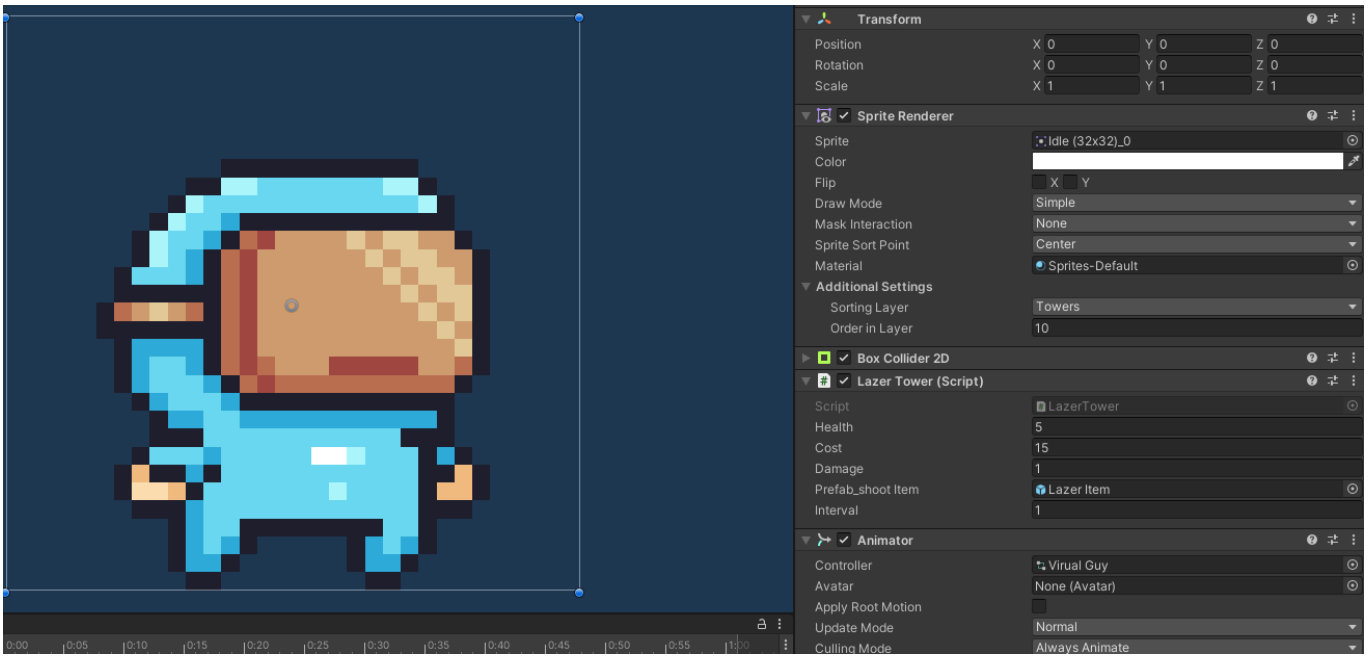


Рис.3.13 Вигляд ігрового об'єкта лазерної вежі в редакторі Unity*

**: розроблено автором*

Скрипт Tower надає такі загальні параметри як Здоров’я та Ціна. Скрипт Lazer Tower надає вежі ще такі властивості як Шкода та Інтервал (пострілів).

Також скрипт Lazer Tower використовує префаб Lazer Item: лазерний снаряд, яким стріляє цей персонаж. Як снаряд було створено об’єкт зі спрайтом блакитного прямокутника.

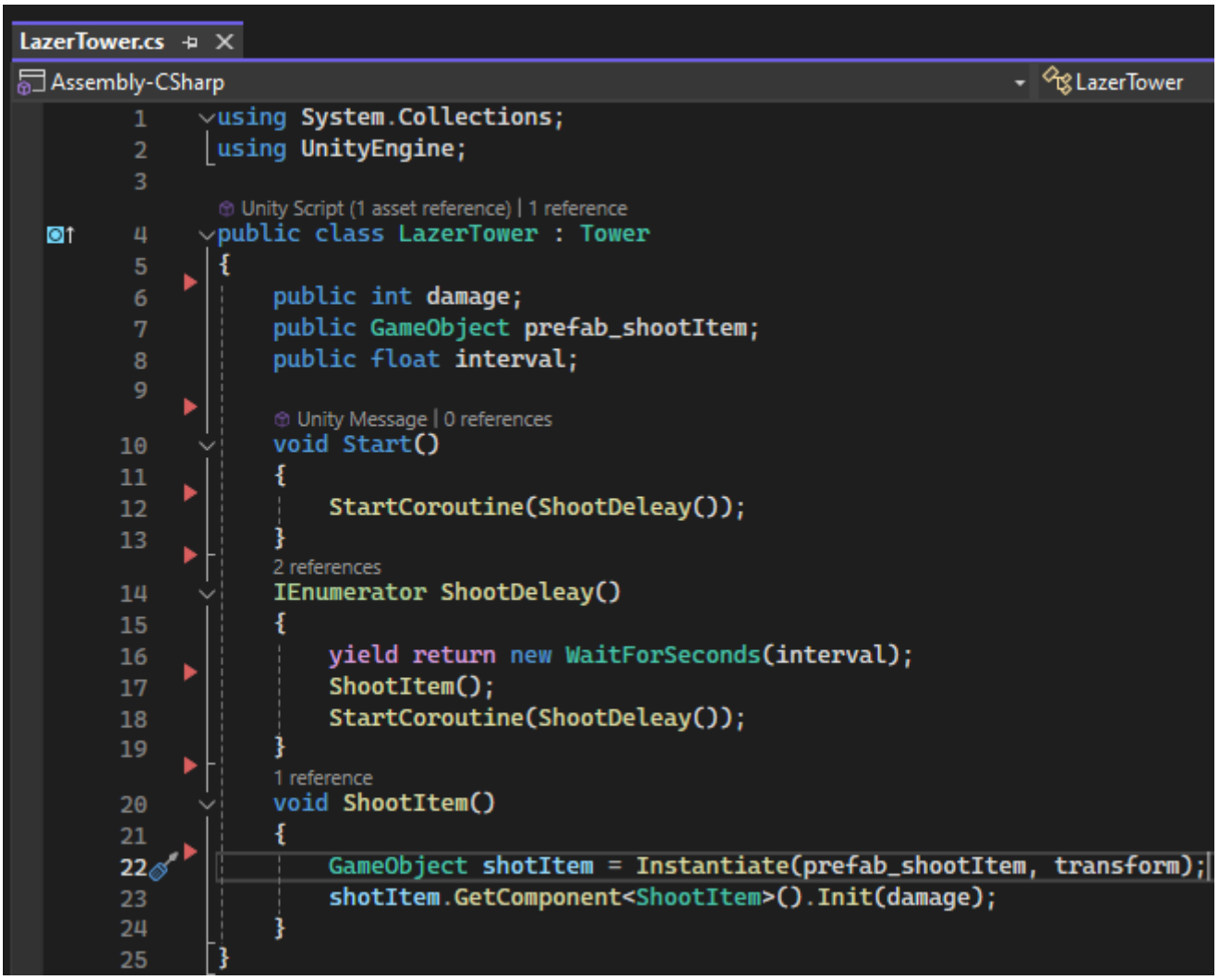


Рис.3.14 Скрипт лазерної вежі*

**: розроблено автором*

Лазерна вежа має менший інтервал атаки(1) ніж інші вежі, але коштує дорожче(15). Також у цієї вежі є такі властивості як здоров’я(5) та шкода(1).

Анімації веж

Для анімацій було використано Unity компонент Аніматор. В Аніматор було додано ряд спрайтів зарання відредагованих за допомогою Sprite Editor під відповідні параметри.

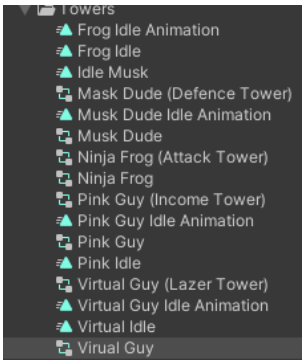


Рис.3.15 Список анімацій веж у редакторі Unity*

**: розроблено автором*

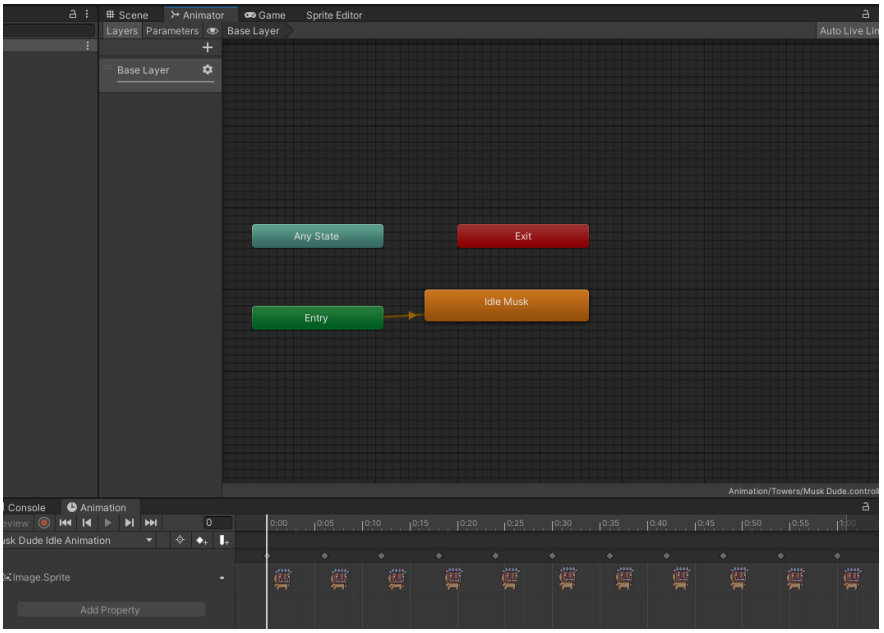


Рис.3.16 Дерево та вікно анімацій(захисної вежі) у редакторі Unity*

**: розроблено автором*

Анімації для префабів веж та веж на панелі вибору зроблені окремо, оскільки робляться по різному.

Для панелі вибору Аніматор посилається на компонент Image, а для префабу на компонент Sprite Render.

3.5. Системи

Розроблено систему розміщення вежі, як показано на малюнку 1.

Рис.3.17 Система розміщення вежі*

**: розроблено автором*

Систему розміщення вежі реалізовано за допомогою налаштованих префабів(що розставлені на спеціальній панелі) та скрипту Spawner, який зчитує натискання мишкою на певний префаб(що робить його вибраним), перевіряє наявність достатньої кількості валюти, зайнятість клітинки, а тоді лише дозволяє поставити вежу на обране місце.

Система здоров’я

Розроблено систему здоров’я, як показано на малюнку 2.

Рис.3.18 Система Здоров’я*

**: розроблено автором*

Система здоров’я реалізована за допомогою знайденого раніше спрайту піксельного серця, завантажуваного Unity об’єкту TextMeshPro та скрипту HealthSystem, що спочатку присвоює дефолтне значення здоров’я, потім зменшує його коли об’єкт з тегом Enemy заходить у зону трігеру, далі переприсвоює значення текстового поля, при наступних зменшеннях здоров’я буде перевіряти чи не стала кількість здоров’я менше одиниці, якщо це так, то викликає вікно програшу.

Система Валюти

Розроблено систему валюти як показано на малюнку 3.

Система валюти реалізована за допомогою скрипту CurrencySystem, що спочатку присвоює базове значення валюти, при покупці вежі перевіряє наявність потрібної кількості валюти, а потім зменшує її коли гравець ставить вежу на ту кількість валюти скільки коштує дана вежа, також кількість валюти може бути збільшена за допомогою вежі отримання валюти.

Усі ці зміни кількості валюти одразу ж переприсвоює значення для текстового поля.

Рис.3.19 Система Валюти*

**: розроблено автором*

3.6. Вороги

Після появи ворог рухається з точки появи(за екраном гравця) через ігрове поле(з права наліво) де стоять вежі. Отримує шкоду при попаданні снарядом вежею. Якщо він дійшов до вежі, то починає завдавати їй шкоди з певним інтервалом та анімацією. Якщо він дійшов до кінця шляху, заходячи в зону тригера, гравцю знімається одне здоров'я. Якщо ворог таким чином забирає останнє здоров'я у гравця це тригерить вікно програшу.

Розроблено префаби ворогів, як показано на малюнках 4 та 5.

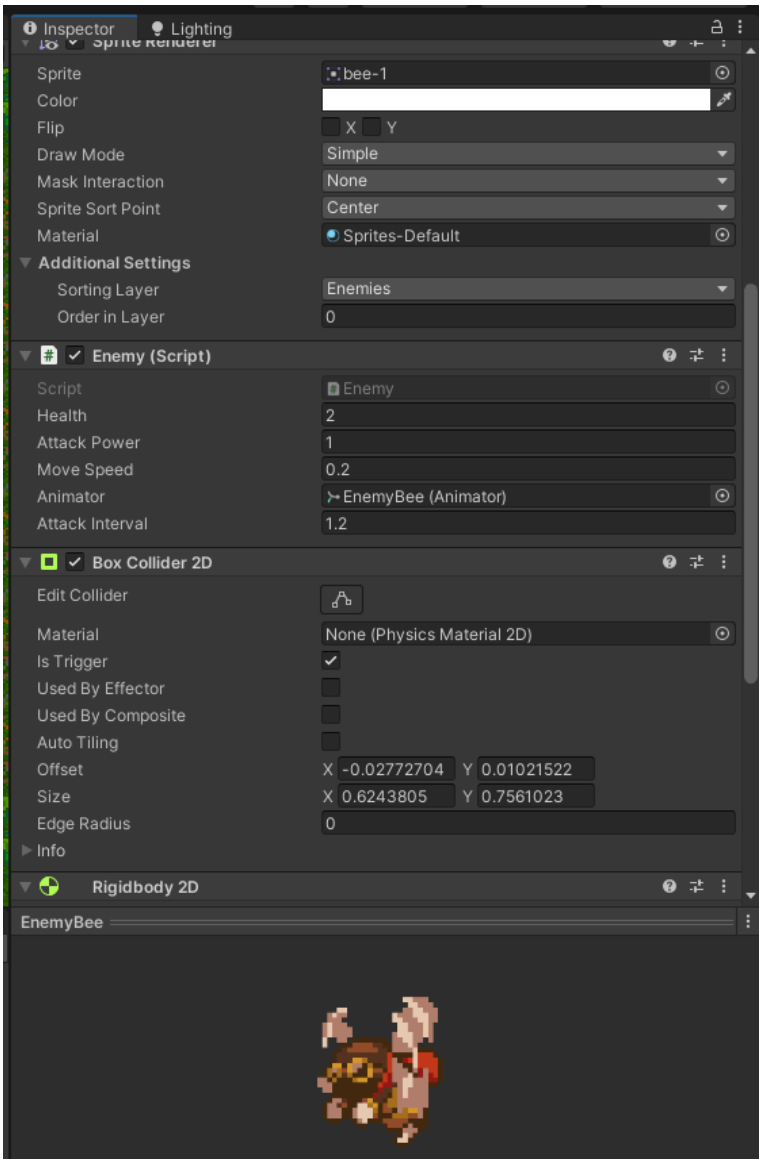


Рис.3.20 Bee*

**: розроблено автором*

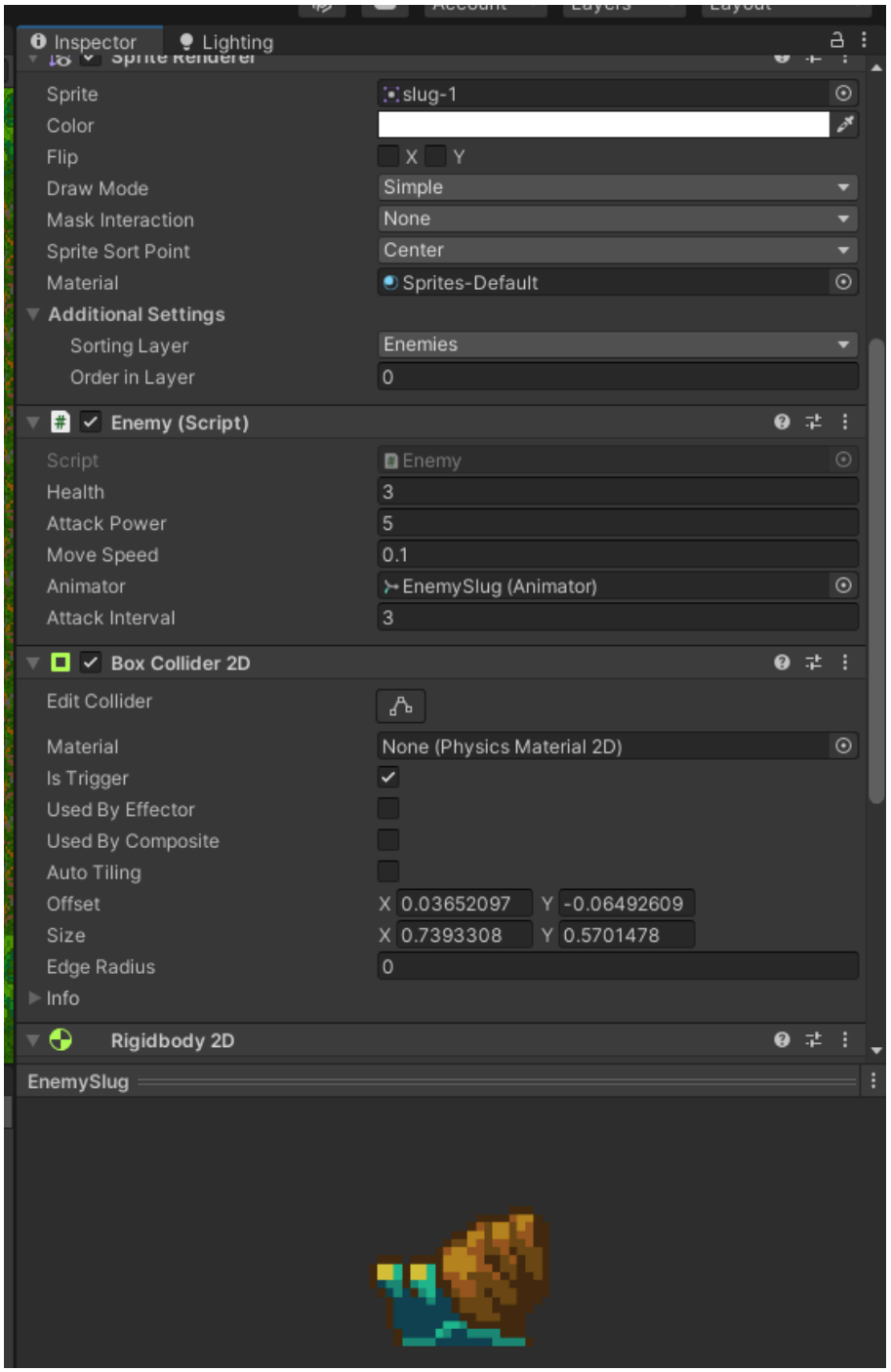


Рис.3.21 Slug*

**: розроблено автором*

Префаби ворогів були реалізовані за допомогою знайдених раніше спрайтів Bee та Slug, компонентів Box Collider 2D, що надає їм можливість взаємодіяти з зовнішнім середовищем та Rigidbody 2D, що надає їм фізичні властивості, а також скрипту Enemy, що надає їм функціонал руху, атаки та інше.

Генерація ворогів

Розроблено спавнер ворогів, логіку руху та атаки як показано на малюнку 6.

Спавнер ворогів реалізований за допомогою 6 точок на кінці кожної лінії за екраном гравця(на яких будуть з’являтися вороги) та скрипту, що запускає таймер інтервалу спавну(2 секунди), і по закінченню цього таймеру спавнить випадкового ворога(з наявних префабів) на випадковій лінії.

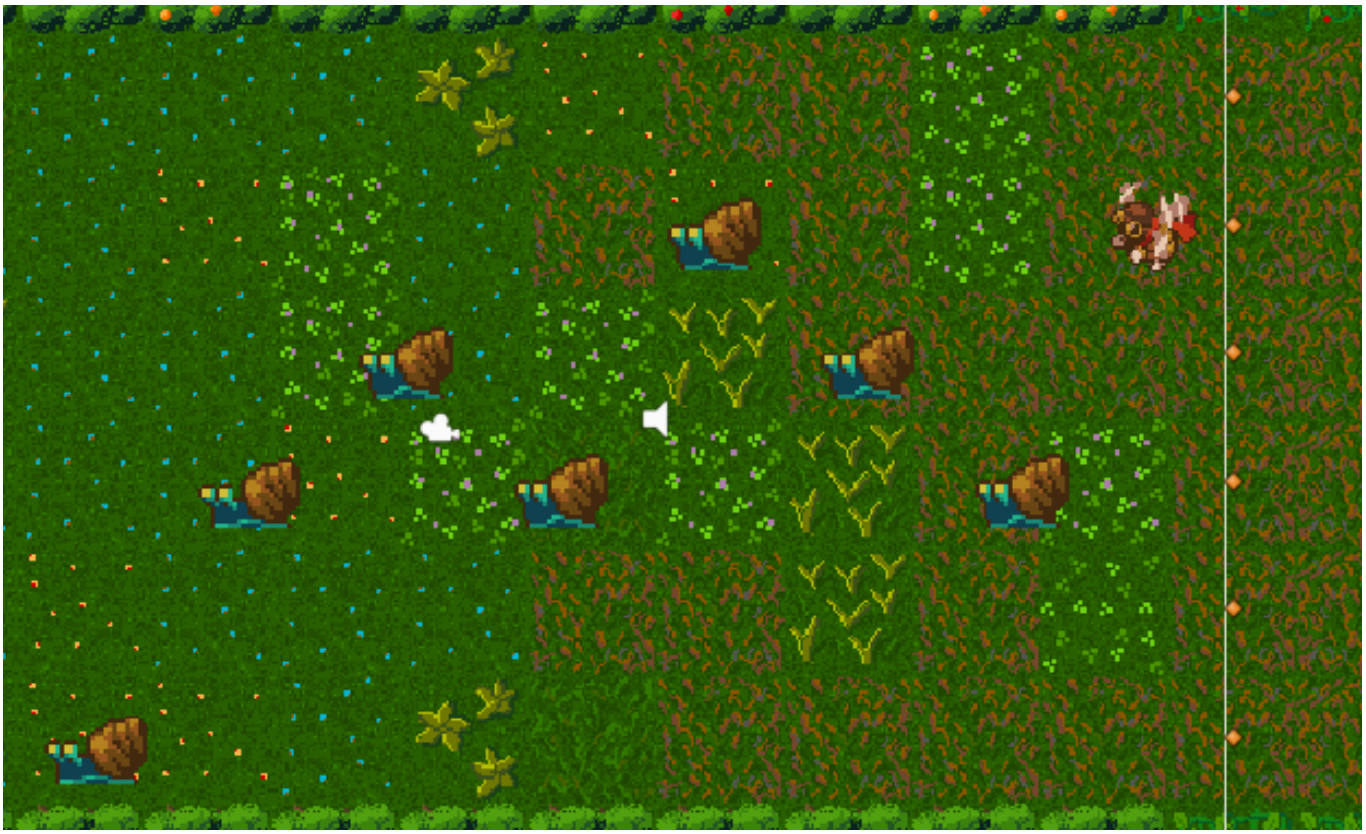


Рис.3.22 Спавнер ворогів*

**: розроблено автором*

Логіка руху ворогів реалізована у скрипті Enemy, використовуючи компонент Transform, що відповідає за позицію об’єкта у просторі.

Логіка атаки ворогів реалізована у скрипті Enemy, дістаючи дані про кількість здоров’я вежі та зменшуючи його при атаці на задану кількість і так доти доки не помре або вежа або ворог.

Анімації ворогів

Розроблено анімацію руху та атаки ворогів, як показано на малюнку 7 та діаграмі 1.

Анімація руху була реалізована за допомогою завантажуваного компонента Аніматор, що виходячи з назви надає можливість створити анімацію, та знайдених раніше спрайтів кожного елементу анімації.

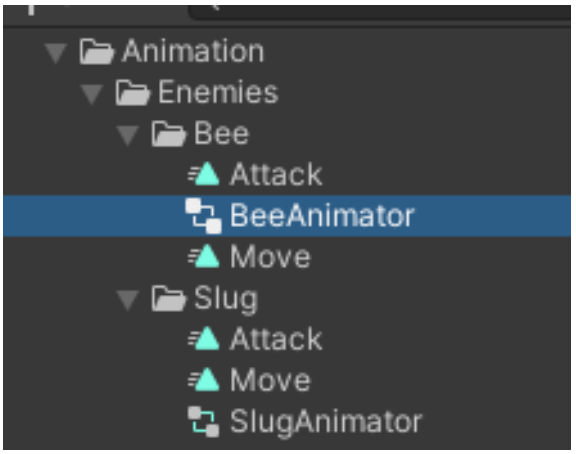
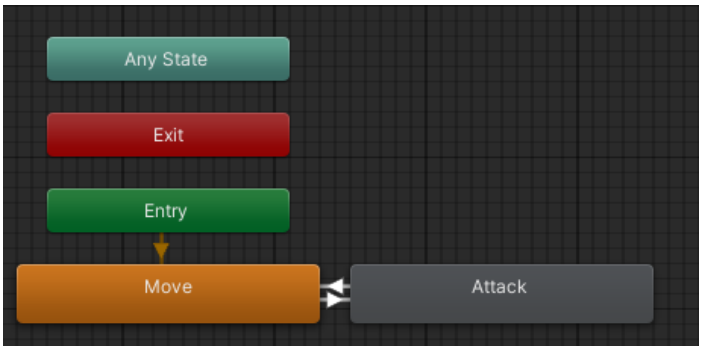


Рис.3.23 Анімації ворогів*

**: розроблено автором*



Діаг.3.2 Діаграма анімації*

**: розроблено автором*

Потім дописаний виклик цієї анімації у скрипті Enemy у функції Move.

Анімація атаки була реалізована за допомогою завантажуваного об'єкта Аніматор, що виходячи з назви надає можливість створити анімацію, та зміни довжини початкового спрайту протягом анімації.

Потім дописаний виклик цієї анімації у скрипті Enemy у функції Attack.

М’ясоїдна рослина

Надано йому властивостей за допомогою компонентів Sprite Renderer[29], Box Collider 2D[30], Animator[31] та скрипту Enemy.

Використовуючи надані скриптом Enemy поля було створено середній, між іншими, по властивостях ворог.

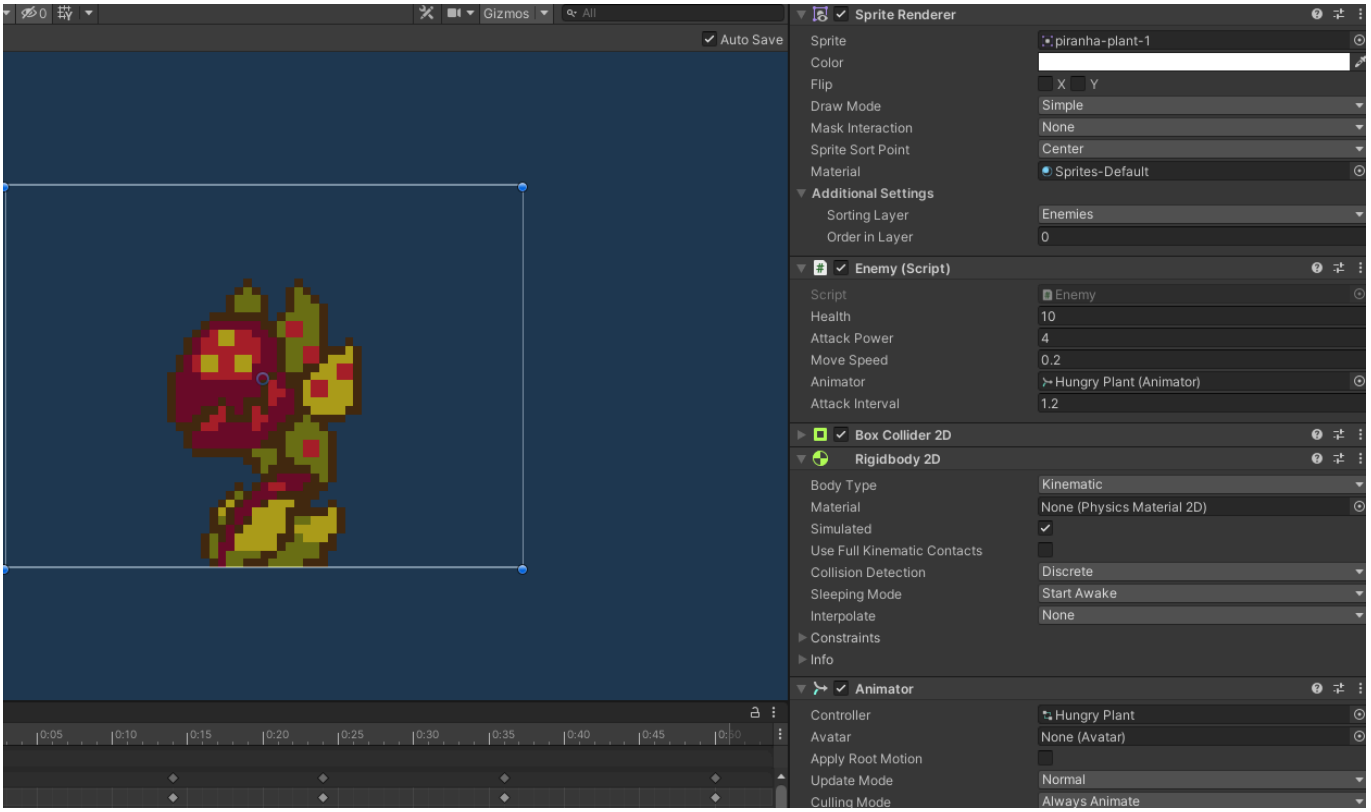


Рис.3.24 Вигляд ігрового об'єкта М’ясоїдної рослини в редакторі Unity*

**: розроблено автором*

Має 10 здоров’я, 4 шкоди, пересувається зі швидкістю 0.2, а б’є з інтервалом 1.2.

3.7. Вікно програшу

Розроблено вікно програшу, як показано на малюнку 1.

Рис.3.25 Вікно програшу*

**: розроблено автором*

Вікно програшу було реалізовано за допомогою Unity компоненту Текстове поле, знайденого раніше спрайту для кнопки, кнопки Restart, що запускає гру спочатку використовуючи функцію ChangeScene скрипту Menu для переходу на цю ж сцену, кнопки Menu, що відправляє гравця на сцену початкового меню, та фонової музики,
що додана за допомогою Unity компоненту Audio Source та доданого у відповідне поле файлу звуку.

3.8. Готовий білд

Витягнув білд(готову гру) з Unity, нічого не змінюючи в налаштуваннях крім основної картинки гри, як показано на малюнках 1 та 2.

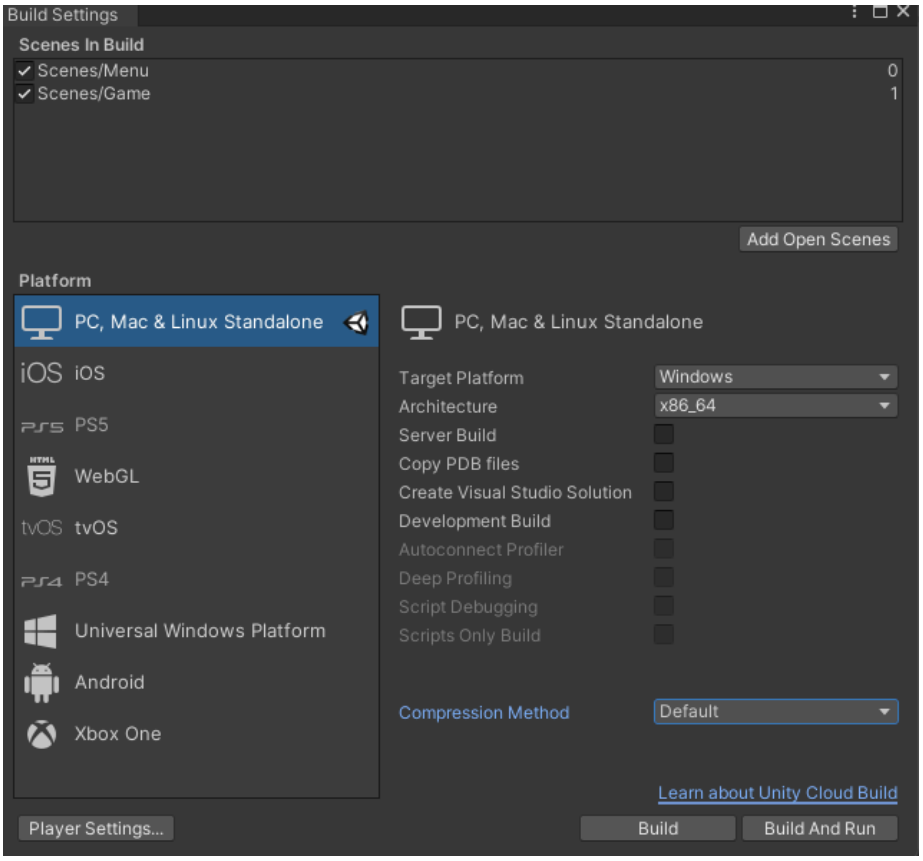


Рис.3.26 Вікно налаштування білду*

**: розроблено автором*

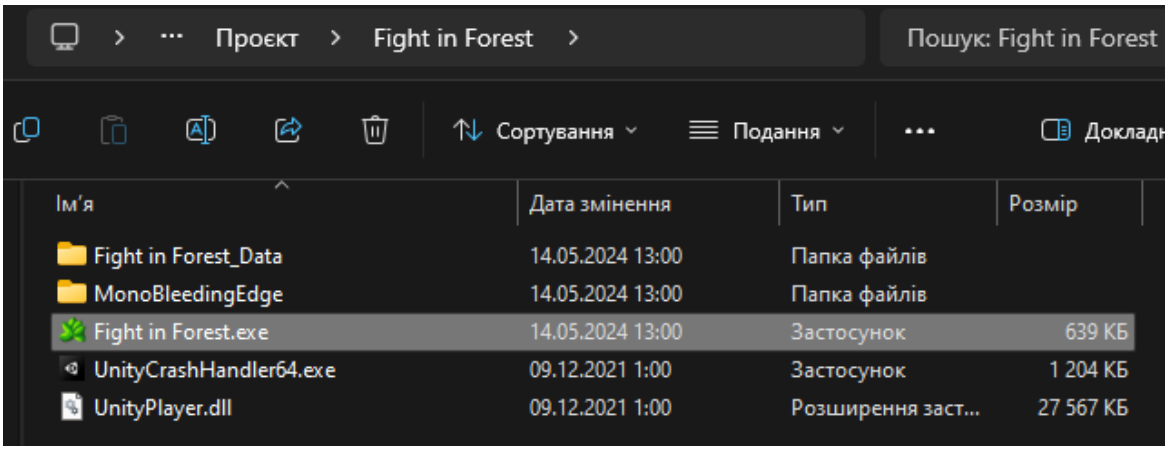


Рис.3.27 Exe-шка готового білду*

**: розроблено автором*

Висновки

Я задоволений результатами, отриманими в рамках цієї кваліфікаційної роботи, враховуючи все ще не великий досвід роботи з Unity. Після аналізу цілей і завдань проєкту встановлено, що кожен етап було виконано відповідно до вихідних вимог. А тобто виконано такі вимоги як: Головне меню, Фонове зображення, Фонова музика, Система розміщення вежі, Система здоров'я, Система валюти, Вежа отримання валюти, Атакуюча вежа, Спавнер ворогів, Логіка руху ворогів, Логіка атаки ворогів, Механіка програшу, Нові вежі, Новий ворог та Анімації веж.

Процес розробки не тільки допоміг досягти конкретних цілей проекту, але також допоміг розвинути навички використання різних функцій Unity, таких як анімація, префаби, використання Tile Maps тощо. Покращення навичок написання скриптів відбилося на якості коду та загальній продуктивності.

Загалом, цей проект не лише досягає своєї основної мети, а також є значущим у моєму професійному зростанні. Досягнення та знання будуть використані в наступних проєктах, що допоможе створити складніші та захопливіші ігрові враження.

Список використаних джерел

1. Асет-пак Sunny Land Forest: https://assetstore.unity.com/packages/2d/characters/sunny-land-forest-108124. (дата звернення: 05.12.2023 р.).
2. Backyard Top-Down Tileset: https://assetstore.unity.com/packages/2d/environments/backyard-top-down-tileset-53854. (дата звернення: 05.12.2023 р.).
3. Асет-пак Pixel Adventure 1: https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360. (дата звернення: 05.12.2023 р.).
4. Асет-пак з музикою Adventure music and SFX: https://assetstore.unity.com/packages/audio/music/adventure-music-and-sfx-221545. (дата звернення: 05.12.2023 р.).
5. Геймплей Bloons TD: https://uploads.golmedia.net/uploads/articles/article_media/20612905241641377784gol1.jpg. (дата звернення: 05.12.2023 р.).
6. Геймплей Kingdom Rush: https://cdn.cloudflare.steamstatic.com/steam/apps/246420/ss_502205852350bfd8d64070adf4c38f2ffdf86772.600x338.jpg?t=1715799745. (дата звернення: 05.12.2023 р.).
7. Геймплей Plants vs Zombies: https://cdn.cloudflare.steamstatic.com/steam/apps/3590/0000008164.600x338.jpg?t=1615390608. (дата звернення: 05.12.2023 р.).
8. Логотип Unity: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTFt3PPIDo3RWVopaPFy0nSRrCZ3nlijkMY-lHhJDQDBA&s. (дата звернення: 05.12.2023 р.).
9. Інформація про платформу Unity: https://uk.wikipedia.org/wiki/Unity_(%D1%96%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%80%D1%83%D1%88%D1%96%D0%B9). (дата звернення: 05.12.2023 р.).

10. Інформації про жанр Tower Defence: https://uk.wikipedia.org/wiki/Tower_defense. (дата звернення: 06.12.2023 р.).
11. Документація по Unity2D: https://docs.unity3d.com/Manual/Unity2D.html. (дата звернення: 06.12.2023 р.).
12. Документація по графіці: https://docs.unity3d.com/Manual/Graphics.html. (дата звернення: 06.12.2023 р.).
13. Документація з фізики: https://docs.unity3d.com/Manual/PhysicsSection.html. (дата звернення: 06.12.2023 р.).
14. Документація зі створення скриптів: https://docs.unity3d.com/Manual/ScriptingSection.html. (дата звернення: 06.12.2023 р.).
15. Документація по аудіо: https://docs.unity3d.com/Manual/Audio.html. (дата звернення: 06.12.2023 р.).
16. Документація по анімаціях: https://docs.unity3d.com/Manual/AnimationSection.html. (дата звернення: 06.12.2023 р.).
17. Документація по користувацькому інтерфейсу: https://docs.unity3d.com/Manual/UIToolkits.html. (дата звернення: 06.12.2023 р.).
18. Документація по розробці на VR: https://docs-unity3d-com.translate.goog/Manual/VROverview.html?_x_tr_sl=en&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=sc. (дата звернення: 06.12.2023 р.).
19. Інтерфейс Unity: https://docs-unity3d-com.translate.goog/Manual/UsingTheEditor.html?_x_tr_sl=en&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=sc.
20. Можливості Unity: https://foxminded.ua/dvyzhok-unity. (дата звернення: 06.12.2023 р.).
21. Інформація про C#: https://uk.wikipedia.org/wiki/C_Sharp. (дата звернення: 06.12.2023 р.).

22. Документація по ігрових об’єктах в Unity: https://docs.unity3d.com/Manual/GameObjects.html. (дата звернення: 07.12.2023 р.).
23. Документація по сценах: https://docs.unity3d.com/Manual/CreatingScenes.html.(дата звернення: 07.12.2023 р.).
24. Документація по компонентах: https://docs.unity3d.com/2019.4/Documentation/Manual/UsingComponents.html. (дата звернення: 07.12.2023 р.).
25. Документація по віртуальній камері: https://docs.unity3d.com/Manual/CamerasOverview.html. (дата звернення: 07.12.2023 р.).
26. Документація по об’єктах взаємодії: https://docs.unity.cn/2020.2/Documentation/Manual/UIInteractionComponents.html. (дата звернення: 07.12.2023 р.).
27. Документація по префабах: https://docs.unity3d.com/Manual/Prefabs.html. (дата звернення: 07.12.2023 р.).
28. Об’єктно орієнтоване програмування: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F. (дата звернення: 07.12.2023 р.).
29. Документація по Sprite Renderer: https://docs.unity3d.com/Manual/class-SpriteRenderer.html. (дата звернення: 07.12.2023 р.).
30. Документація по Box Collider 2D: https://docs.unity3d.com/Manual/class-BoxCollider2D.html. (дата звернення: 07.12.2023 р.).
31. Документація по Animator: https://docs.unity3d.com/ScriptReference/Animator.html. (дата звернення: 07.12.2023 р.).