МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри
_____ Є.О.Давиденко
«__» _____ _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Мобільний 2D–платформер на базі Unity

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.01 – 409.21810915

Студент _____ **О. П. Кисса**
підпис
«__» _____ 2022 р.

Керівник канд. техн. наук, доцент _____ **Г. В. Горбань**
підпис
«__» _____ 2022 р.

Консультант канд. техн. наук, доцент _____ **А. О. Алексєєва**
підпис
«__» _____ 2022 р.

м. Миколаїв – 2022 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПК – персональний комп'ютер

JSON – JavaScript Object Notation

ВСТУП

У наш час мобільні ігри стають все більш популярними для проведення вільного часу, як серед молоді так і серед дорослих. Ігри на смартфоні стали популярні завдяки своїй мобільності, користувачу більше не потрібно ПК чи ігрова консоль. Більшість ігор які були зроблені для ПК чи консолей перероблюють під мобільні платформи.

Сучасні відеоігри вже давно стали одним із видів мистецтва, оскільки з розвитком обчислювальної потужності сучасних комп'ютерів з'являються продукти, що вражають реалістичною графікою, величезними світами, сюжетами та сценами. Вони дуже різноманітні, тому кожен зможе знайти проект, який йому буде цікаво грати. Ринок ігрової індустрії показує, що Android ігри приносять 56% прибутків від мобільних ігор.

Об’єктом кваліфікаційної роботи є мобільний 2D–платформер на базі Unity.

Предметом кваліфікаційної роботи є технології створення платформеру на базі Unity.

Метою кваліфікаційної роботи є створення повноцінної гри на базі Unity, з використанням нових технологій програмування ігор.

Завдання для досягнення поставленої мети:

– дослідити об’єктну та предметну галузь;
– проаналізувати аналоги, що вже існують;
– розробити специфікацію вимог до гри у жанрі Platformer;
– розробити архітектуру гри;
– створити користувацький інтерфейс;
– реалізувати основні механіки;
– провести тестування розробленої гри.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Опис предметної сфери

Створення платформеру з простим та цікавим геймплеєм, але який кине виклик гравцеві. Користувач гри – це гравець, який може налаштувати гру та перейти до геймплею

У грі різні вороги вони відрізнятимуться один від одного як за графікою, так і за тим, як вони завдають шкоди гравцю. Також у грі будуть міні–боси у яких буде багато здоров'я та боси з унікальними діями, але після цього будуть щедрі нагороди на кшталт скрині з якої так само можна буде покращувати здібності гравця. Деякі повільно переслідуватимуть гравця, доки не вдарять його. Інші також стрілятимуть у гравця та накладатимуть на нього різні негативні ефекти.

Гравець також може обирати нові типи зброї та обирати для них посилення. Самі вороги будуть різні в залежності від рівня. Сама гра буде закінчуватися, коли гравець пройде весь рівень чи в нього буде мало здоров’я, буде випливати вікно з вибором почати заново гру чи вийти в головне меню.

1.2 Аналіз існуючих рішень для реалізації проєкту

Unity — багатофункціональний інструмент для створення відеоігор та застосунків. Unity дозволяє створювати програми, які працюють на більш ніж 25 різних платформах, включаючи персональні комп'ютери, ігрові консолі, мобільні пристрої, веб-програми та багато іншого. Випуск Unity відбувся у 2005 році і з того часу триває постійний розвиток [4]. Застосунки, створені за допомогою Unity, підтримує DirectX та OpenGL.

Переваги платформи:

- низький поріг входження;
- багато бібліотек;
- зручний інтерфейс;
- мультиплатформеність;

- зручна обробка фізики;
- безкоштовність;

Недоліки:

- повільність;
- велика вага;
- повільність;

Unreal Engine – це один із найвідоміших двигунів, на якому було зроблено багато відеоігор для персональних комп'ютерів та ігрових консолей. Цей застосунок має багатий набір інструментів світового класу і прості доступні до робочих процесів, які дозволяють розробникам получати швидкі результати, не торкаючись коду, тому що у застосунок інтегрована графічна мова програмування. Технології рендерингу також дозволяють створювати ігрові програми з красивою та сучасною графікою. А останнім часом гіганти ігрової індустрії починають освоювати цей двигун для ігор на мобільних платформах, одними з найвідоміших є Mortal Kombal Mobile, Linage 2: Revolution та інші [6].

Переваги платформи:

– зручні інструменти для графіки різних типів;
– має вбудовану графічну мову програмування;
– гарні інструменти оптимізації.

Недоліки:

– складність для великих проєктів;
– великий обсяг програмного забезпечення;
– вибагливість до потужностей комп'ютера.

CryEngine – ігровий застосунок, створений німецькою приватною компанією Crytek в 2002 році і спочатку використовувався в шутері від першої особи Far Cry. «CryEngine» – комерційний двигун, який пропонується для ліцензування третім сторонам. Станом на 30 березня 2006 року всі права на двигун належать Ubisoft. [5].

Двигун був ліцензований компанією NCSoft для MMORPG Aion, що розробляється: Tower of Eternity.

Ігровий двигун CryEngine – перший комерційний двигун Crytek. Його розробка була розпочата відразу після заснування компанії. Двигун спочатку розроблявся як технологічна демонстрація для американської компанії nVidia. Однак на виставці ECTS 2000 (англ. European Computer Trade Show — Європейська комп'ютерна виставка) Crytek справила велике враження на всіх великих видавців, відвідувачів та журналістів своєю технічною демонстрацією, що була показана у відділі nVidia. Після цього на основі двигуна було вирішено створити дві гри – "X–Isle" і "Engalus". Жодна з цих ігор так і не була випущена [5].

Переваги платформи:

– інноваційні можливості при розробці ігор;
– підтримка нових технологій: directx 12, vr, vulkan api;
– присутня можливість дописувати скрипти на мові c#;
– наявність попіксельного освітлення;
– підтримка карт відображень.

Недоліки:

– важкий у розумінні, вивченні та роботі;
– наявність багів у редакторі;
– обмежений вибір ассетів;
– обмеження та складності при розробці мережевих ігор;
– дуже скромна техпідтримка;
– слабке ком’юніті;
– невелика кількість документації по роботі з двигуном.

Unity – має дуже велику бібліотеку, де можна знайти різні матеріали для гри: графіку, інструменти та музику. З Unity ви можете розробити гру, не маючи навичок створення 3D– та 2D–графіки або написання коду. В даний час Unity є провідною платформою для створення мобільних ігор, в основному для створення казуальних ігор. Ось чому найкраще вибрати Unity як платформу для

початку роботи в індустрії відеоігор. Саме цей двигун і буде обрано для проекту.

Unity як така має компонентну систему. Компоненти визначають та керують поведінкою ігрових об'єктів, до яких вони приєднані. Компоненти мають багато атрибутів, які можна налаштувати у вікні Інспектора. Сам Unity має безліч різних компонентів, які можуть відповідати за камеру, світло, звук, роботу з collider та фізику об'єктів, анімацію, графічний інтерфейс і т.д. Проте для створення ігор вам доведеться створювати та налаштовувати свої власні компоненти. Кожен компонент можна створити за допомогою скриптів, які вже описують конкретну ігрову логіку та поведінку об'єктів, які вже пов'язані з об'єктами як компонентами. Тема оформлення цього компонента може бути успадкована від класу MonoBehavior. Самі ці компоненти створюються за допомогою бібліотеки C# mov.

C# – це об'єктивно–орієнтовна мова програмування, створена в 1998 інженерами компанії Microsoft [10]. Ця мова програмування була створена для використання на платформі .NET. C# відноситься до сім'ї мови C і має схожий синтаксис. Мова програмування строго типізована, підтримує поліморфізм, навантаження операторів, делегати, атрибути, події та мову запитів LINQ. У порівнянні з C++, C# набагато легше за рахунок того, що має керовану пам'ять, тобто сам керує виділенням пам'яті та запобігає витоку.

Крім того, Unity раніше використовувала мову програмування JavaScript як альтернативу, як її називали UnityScript, невдовзі розробники припинили її підтримку, оскільки C# набрав популярність і перевагу в розробці.

Також у проєкті застосовується формат текстового обміну даними JSON заснований на мові JavaScript. Це легка мова для читання людиною. JSON представляє запис у вигляді «ключ:значення». У цьому форматі будуть зберігатися рекорди гравця.

У проекті також використовується ще один сильний бік Unity–це ScriptableObject, контейнер даних, в якому можна зберігати великі обсяги даних незалежно від екземпляра класу. Один із варіантів – скоротити використання

пам'яті проекту, уникаючи копіювання значень. Це дуже корисно, коли в проекті є prefab, який зберігає дані в прикріплених скриптах [4].

1.3 Аналіз існуючих рішень

Багато геймдизайнерів поєднують різні жанри та механіки, взяті з відмінностей відомих ігрових додатків. Тож у пошуку ідей потрібно ознайомитися з різними іграми у різних жанрах. Таких як shooter, action, top down, casual, platformer де можна виділити наступні ігри: Rayman Origins, Cuphead, Hollow Knight.

Rayman Origins

Розробник (дистриб'ютор): Ubisoft

Архітектура: UbiArt Framework

Мова реалізації: C++

В цій грі ви можете:

– грати за різних персонажів;

– офлайн гра;

– розрахована на багато користувачів.

Переваги:

– деталізована графіка та яскравий, самобутній візуальний стиль;

– точне та чуйне управління;

– художня та геймплейна різноманітність;

– поступово зростаюча складність.

Недоліки:

– відкриття останніх рівнів вимагає педантичного пошуку секретів;

– спільне проходження;

Rayman Origins – гра–платформер з серії Rayman, що повідає про пригоди Реймана та Глобокса. Вперше про майбутню гру розповіли профільні ігрові видання, інформацію було підтверджено офіційним трейлером на E3 2010.

Гру по сюжету можна розділити на дві частини:

1. Порятунок німф;

2. Порятунок королів та знищення Небесного Міста.

Також у грі є бонусний розділ Земля Небіжчиків, доступ до якого можна отримати, зібравши всі 10 скелетних зубів і віддати їх мерцеві на Храпун–Дереві [1].

Рисунок 1.1 – Скріншот ігрового процесу Rayman Origins

Cuphead

Розробник (дистриб'ютор): StudioMDHR Entertainment

Архітектура: Unity

Мова реалізації: C#

В цій грі ви можете:

- розрахована на багато користувачів;
- пропуск рівнів;
- придбання поліпшень;
- Run–and–Gun.

Переваги:

- гра в ретро стилі;
- режим кооперативу;

– дуже гарна графіка та промальовані локації;

– гарний сюжет.

Недоліки:

– відкриття останніх рівнів вимагає педантичного пошуку секретів;

– спільне проходження – добавка без особливого сенсу.

Cuphead — комп'ютерна гра у жанрі run and gun та платформера, розроблена та видана канадською командою розробників StudioMDHR Entertainment. Анонс гри відбувся у 2013 році, а вихід для персональних комп'ютерів та Xbox One відбувся 29 вересня 2017 року. Керуючи персонажем на ім'я Чашек, гравець бореться із серією босів, щоб повернути борг дияволу [3].

Рисунок 1.2 – Скріншот з гри Cuphead

Hollow Knight

Розробник (дистриб’ютор): Team Cherry

Архітектура: Unity

Мова реалізації: C#

В цій грі ви можете:

– офлайн гра чи онлайн гра;

Переваги:

– деталізована графіка;

– самобутній візуальний стиль;

– художня та ігрова різноманітність;

– поступово зростаюча складність.

Недоліки:

– немає.

Hollow Knight (з англ. – «Порожнистий Лицар») — це мультиплатформна комп'ютерна гра в жанрі метроідванія, випущена інді–студією Team Cherry в 2017 році для Windows, на місяць пізніше була вбудована розробниками на Linux і macOS. Розробка гри була спонсоровано через сервіс Kickstarter. Гра повідає нам про пригоди та відкриття безіменного лицаря в давно покинутому королівстві комах Халлоунест. Деякі критики назвали гру однією з найатмосферніших і якісних метроідвань, а також класикою жанру [2].

Рисунок 1.3 – Скріншот з гри Hollow Knight

1.4 Постановка завдання

Основна вимога проєкту – це створити готову гру у жанрі Platformer для Android платформи. Переглядаючи попередні розділи, було виявлено основні

механіки та ідеї гри. Таким чином, керувати персонажем можна за допомогою джойстика, який розташовуватиметься в лівому нижньому кутку і кнопка стрибка і стрільби в правому нижньому кутку.

Для інтерфейсу необхідно визначити сторінки які необхідні і це:

– головна меню із якого можна перейти та почати геймплей;

– головний екран геймплею у якому зображанні кнопки, стік та інформація про здоров’я гравця;

– сторінка завершення гри.

– сторінка паузи;

І також декілька спливаючих вікон:

– вікно паузи;

– вікно вибору рівня.

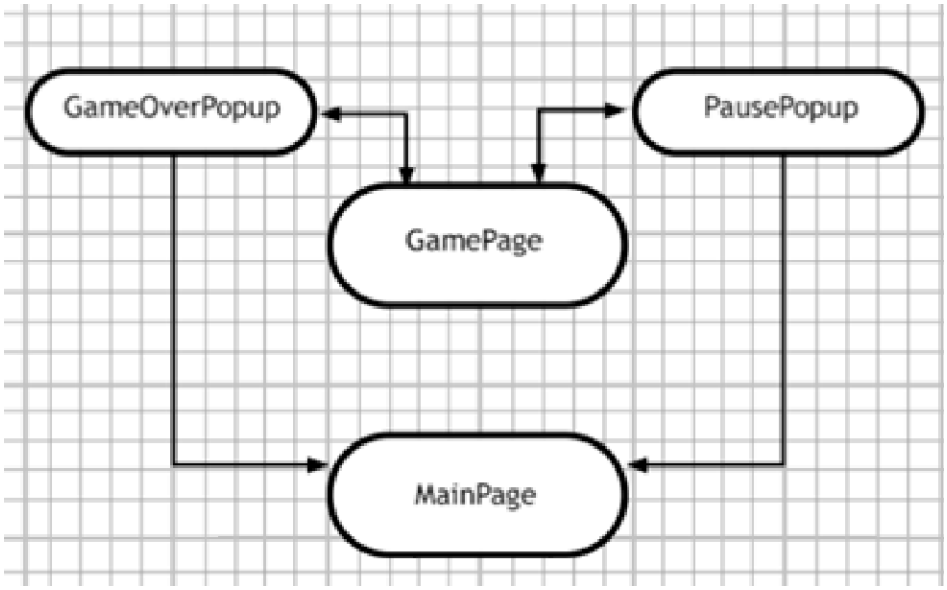

Рисунок 1.4 –Пересування між сторінками

1.5 Специфікація вимог

Основною задачею є створення гри та вдосконалення свої навичкок у розробці ігрових застосунків.

Вирішення цієї проблеми вимагає розробки гри в жанрі платформер із простим і цікавим геймплеєм, що кидає виклик гравцю.

Назва продукту: Platformer

Призначення ПЗ: розробка Android гри;

Сфера застосування: сфера гейм–індустрії;

Основні функції гри яка розробляється:

Ролі користувачів системи: гравець;

Апаратні вимоги: OS Android, 50 Мб вільного місця та ОЗУ 1 Гб або вище;

Програмні вимоги: версія ОС Android 7.0.0 та вище;

Інтерфейс користувача: чіткий та простий у розумінні;

Вимоги до програмного забезпечення:

Данний проєкт має запускатись без загальмовувань на останніх версіях Android та бути адаптивним до різного розширення.

Висновки до розділу 1

За результатами першого розділу було проаналізовано тематичну область та вивчено основні технології отримання щебеню та вивчено основні аналоги. Крім того, були розглянуті аналоги інших мобільних ігрових проєктів. З цих аналогів було розглянуті основні особливості мобільних ігрових застосунків. Також було сформовано та обгрутновано архітектурний підхід для проєктів з застосуванням платформи Unity.

Було зроблено специфікацію вимог до програмного забезпечення. Вона ілюструє основні функціонали програмного застосунку, який створюэться, та основні вимоги.

2 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ГРИ

2.1 Створення Usecase

Usecase – це текстовий опис сукупності сценаріїв, які можуть закінчитися успіхом чи ні, під час роботи користувача із системою задля досягнення певних цілей.

Сценарій – послідовність дій при взаємодії користувача із виконання певних операцій в системі.

Написання варіантів використання дозволяє чітко визначити, хто є користувачем системи, хто виконує його робочі сценарії та ціль використання системи. Варіанти використання - це функціональні та поведінкові вимоги до системи, які показують, що вона має робити[12]. Використовуються три форми листа:

1. Коротка – короткий опис абзацу одного зі сценаріїв (зазвичай успішна) робота системи. Виконується в ході початкового аналізу вимог до системи;
2. Поверхневий – поверхневий опис усіх сценаріїв у вільній формі. (основний та альтернативний) одного з варіантів використання. Виконується під час первинний аналіз системних вимог.
3. Повна – докладно описані всі кроки та дії, у тому числі попередні та Пост–умови виконання юзкейсу виконуються на етапі вибору з повного списку варіанти використання в коротких та поверхневих формах невеликої частини важливого (Критично для роботи системи) використовуються касові апарати [7].

Коротка форма «Генерація рівня»

Користувач заходить у гру. Натискає кнопку «Start». Потрапляє в головну сцени гри. Користувач проходить рівень та потрапляє на екран з вибором вихід з гри чи перейти до наступного рівня.

Поверхнева форма «Генерація рівня»

Користувач заходить у гру. Потрапляє у головне меню гри. Натискає «Start».

Таблиця 2.1 – Повна форма «Генерація рівня»

Usecase section	**Comment**
Use Case Name	Генерація рівня
Scope	Система генерації рівня
Level	Генерація рівня
Primary Actor	Користувач
Stakeholders and interests	Користувач – користування застосунком
Preconditions	Користувач повинен мати телефон із встановленою OS Android
Success guarantee	1. у користувача встановлена остання версія; 2. у користувача є мінімальні вимоги, які потрібен додаток;
Main Success Scenario	1.користувач входить у гру; 2.натискає на кнопку запуску гри; 3.потрапляє в ігровий рівень;
Special Requirements	1. OS Android має бути версії 6.0; 2. апаратне забезпечення повинне мати Оперативна пам'ять не менше 1 ГБ; 3. апаратне забезпечення повинне мати 50 МБ вільної пам'яті чи більше.
Frequency of Occurrence	50%

2.2 Діаграми варіантів використання

Діаграма варіантів використання – це концептуальне уявлення чи модель системи під час її проектування та розробки. Створення діаграми використання має такі цілі:

В проєкті було створено діаграми варіантів використання. На (рис.2.1) можна побачити діаграму варіантів використання мобільного застосунку.

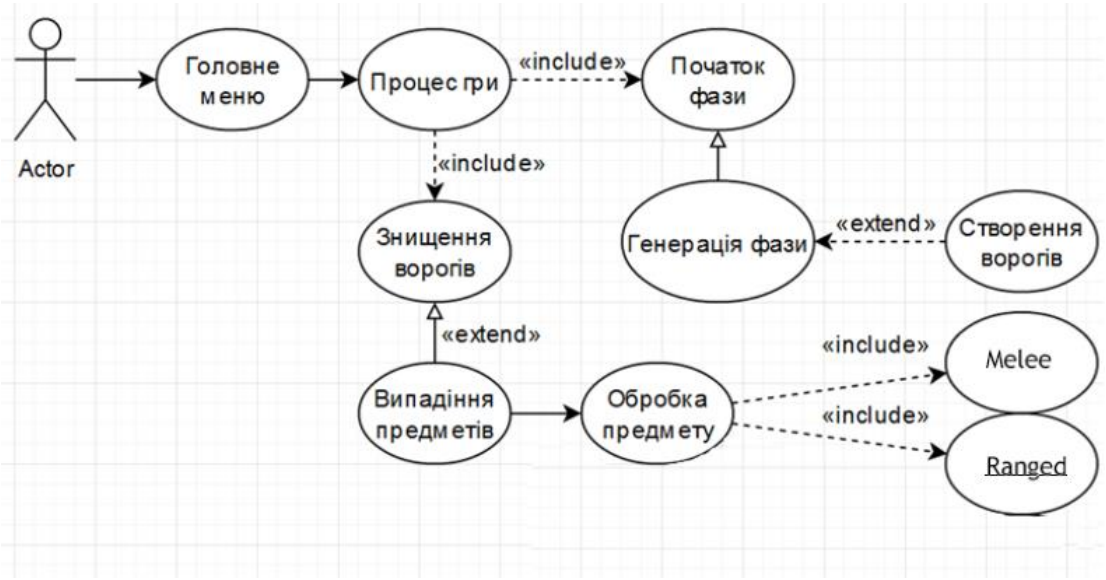

Рисунок 2.1 – Діаграма варіантів використання

Гравець вмикає гру та потрапляє до головного меню, де він переходить до гри. Користувач потрапляє на перший рівень гри.

Структура специфікації (опису) варіанта використання включає в себе наступні елементи:

– найменування;
– короткий опис;
– ідентифікатор;
– актори (головні та другорядні);
– передумови;
– основний напрямок;
– альтернативні потоки;
– післяумови;

– спеціальні вимоги.

А) Специфікація варіанту використання «скорочення часу»:

– Ім’я: скорочення часу;
– ID: 0;
– Короткий опис: гравець грає під час відпочинку від роботи чи домашніх справ;
– Основна дійова особа: гравець;
– Другорядні дійові особи: немає;
– Передумови: немає;
– Основний потік:
  - варіант використання почнеться, коли гравець запустить систему;

Постумови:

- гравець програв та вимкнув систему;

Альтернативні потоки: немає.

Б) Специфікація «грати в гру»:

Ім’я: грати в гру;

ID: 1;

Короткий опис: знайомство з застосунком та отримання ігрового досвіду;

Дійова особа: користувач;

Другорядні дійові особи: вороги, ігрові об’єкти, ігровий персонаж;

Основний потік:

- ВВ почнеться коли гравець натисне кнопку «Start»;
- Система почне відробляти свою функцію

Постумови:

- гравець грає персонажем та винищує ворогів, підіймаючи предмети та удосконалюючи ігрового персонажа;

Альтернативні потоки:

- гравець повернеться у головне меню.

2.3 Діаграма класів

Наступна діаграма є діаграмою класів. Діаграми класів є основним типом діаграм UML. Вони відбивають логічну структуру програмної системи. Це дуже впливає на процес генерації коду. Основні елементи діаграми класів є самі класи (classes) та відносини між ними (relationships).

Клас – сукупність логічних об’єктів, які мають характеристики та різну поведінку. Характеристики в ООП об’єктів певного класу представлені набором атрибутів, а поведінка – набором операцій.

Можна налаштувати 5 основних типів відносин між класами:

– асоціація (association) – визначає абстрактний зв’язок між класами;

– агрегація (aggregation) – відношення типу «частина – ціле», при якому час життя класа частини не співпадає з часом життя класа цілого;

– композиційний (composition) – відношення типу «частина – ціле», при якому час життя класу частини співпадає з часом життя класу цілого;

– наслідування (generalization) – відношення типу «загальне – часткове»;

– інстанціювання (instantiation) – визначає інстанціювання нового класу з параметризованого класу шляхом підставлення фактичних параметрів в формальні параметри шаблона [7].

Сама архітектура є незвичною для проектів єдності, оскільки замість створення багатьох компонентів створюється лише один Манапп і основні функції двигуна проходять через нього. Далі – Gameclient, який включений до класів через інтерфейс IService.

Uimanager – керує інтерфейсом користувача через інтерфейси iUipopup iUipopup. Саме завдяки цьому менеджеру йде головна взаємодія зі сторінками.

Datamanager – використовується для зберігання інформації та стану гри.

LoadObjectManager використовується для отримання різних звуків, колекцій або зображень.

GameManager – головний менеджер, який контролює головний геймплей через класи, що реалізують інтерфейс Icontroller.

InputManager – який обробляє кнопки, натискається на повідомлення про свою взаємодію через події.SoundManager використовується для взаємодії з звуками.

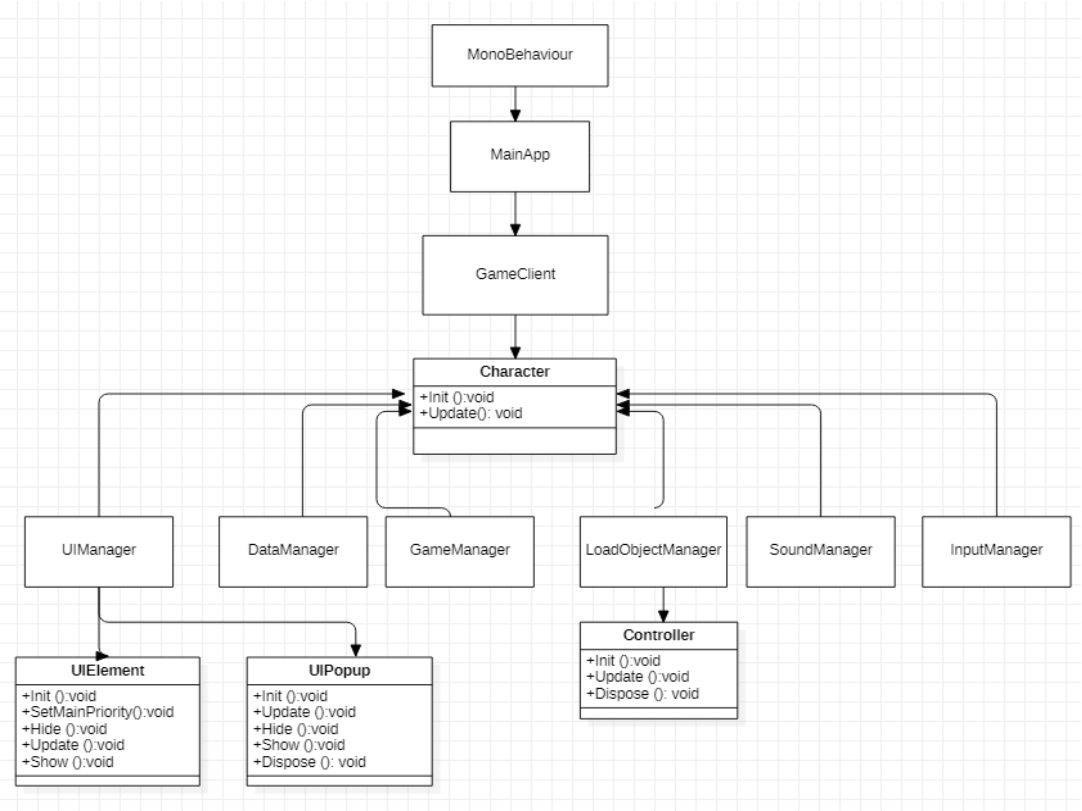

Рисунок 2.2 – Діаграма класів

2.4 Діаграми станів та переходів

Статистичні діаграми разом із діаграмами відображають деякі або всі сценарії, які виконуються в системі в цілому.

Діаграма станів зображує готову машину у вигляді графіка, піки яких є станами об'єкта, поведінка якого моделює, а переходи – це події, що переводять об'єкт з одного стану в інший. [11].

Стан (state) – це відмінна логіка, яка використовується для моделювання певної ситуації, дії або процесу. Кожний стан має назву та список внутрішніх дій. Перелік внутрішніх дій містить дії, які виконуються в процесі пошуку

системи чи об'єкта в певному стані. Кожна дія відображається у форматі "Період продуктивності" та "Ім'я дії", де "Період виконання" може приймати такі значення:

– OnEntry –виконується дія поки система переходить у цей стан;
– OnExit – виконується дія при виході з цього стану;
– Do –виконується дія перебуваючи в цьому стані;
– OnEvent –виконується дія коли відбувається певна подія [8].

Представлено схему станів і переходів ураження (рис. 2.3). Ви можете бачити, що значення за промовчанням введені в поля на панелі генерації, коли вона відкрита. Потім, коли місцезнаходження вибрано, запис записується з новими значеннями. Крім того, встановлення розміру перезаписує попередні значення. Ті самі події відбуваються з іншими полями. Зрештою, при натисканні на покоління проводиться подія, після чого генерується ігровий регіон.

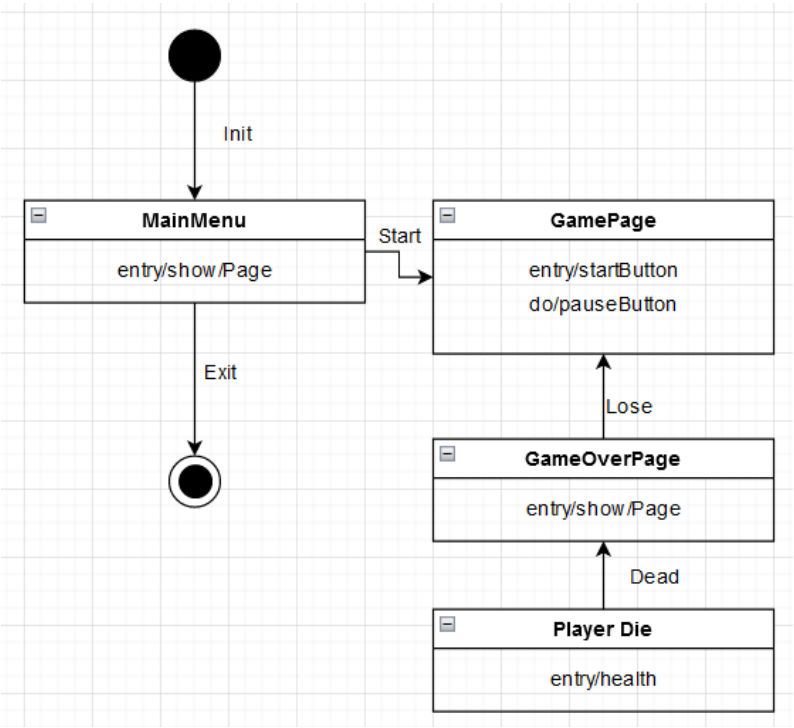

Рисунок 2.3 – Діаграма станів та переходів поразки

Діаграма станів та переходів процесу гри зображені на (рис. 2.4).

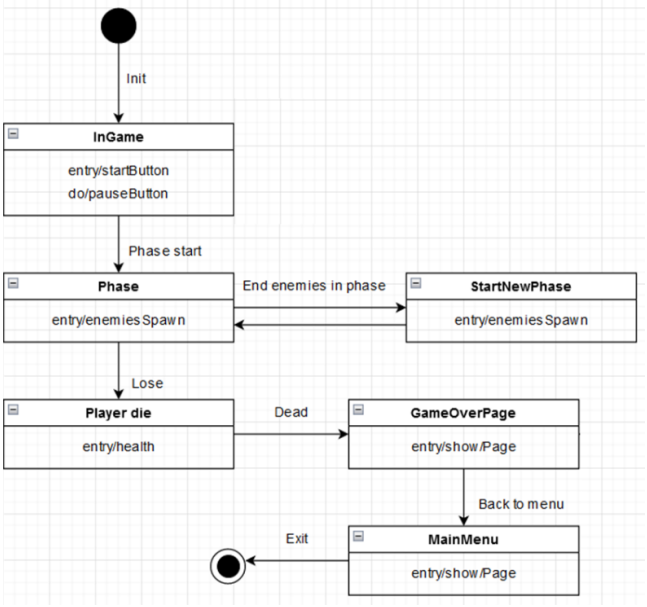

Рисунок 2.4 – Діаграма станів та переходів процесу гри

Відтворення ігрового процесу показує геймплей. Після початку фази покоління вороги з’являються на сцені, після появи ворогів у фазі фази будуть оновлені. Також проілюстровано, що після поразки гра закінчується. Але ви можете перезавантажити рівень та продовжити грати.

Висновки до розділу 2

В цьому розділі було створено різні діаграм, які, в свою чергу, повинні покращити якість програми для відображення її використання та функції. У цьому розділі такі типи діаграм, як діаграми взаємодії, використовують діаграми, діаграми класів, діаграми та переходи та переходи, розглянуті діаграмами використання.

3 ПРОЄКТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ

3.1 Користувацький інтерфейс

Інтерфейс користувача повинен взаємодіяти з користувачем Програма користувача вводить необхідні дані, натискає на різні кнопки досягти результату.

Інтерфейс гри чи будь якого іншого застосунку, створюють спеціальні люди (дизайнери) вони проектують макети UI інтерфейсу із приблизним видом (начерк) на ранніх етапах проєкту, а вже потім, коли більшість деталей узгоджуються, то створюють версію інтерфейсу якомога ближче до остаточної версії..

В нашому випадку було створено три мокапи користувацького інтерфейсу. Їх можна порівняти із реалізованою програмою і побачити, що вони максимально схожі.

Перший мокап присвячений головному меню відео – гри. У низу знаходяться дві кнопки почати гру та вихід с гри . На (рис. 3.1) та (рис. 3.2) можна побачити мокап інтерфейсу користувача та вже його фінальний варіант у проєкті.

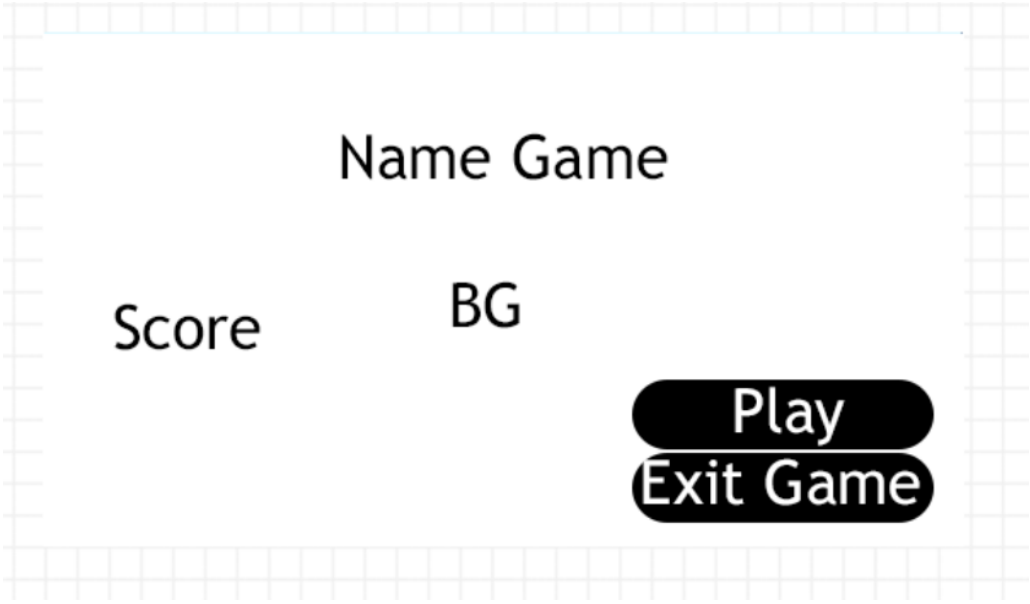

Рисунок 3.1 – Макет головного меню в грі

Рисунок 3.2 – Реалізований макет головного меню в грі

На наступному користувацькому інтерфейсі зображено макет та реалізація головної сцени гри. На (рис.3.3) продемонстровані кнопки завдяки котрим користувач може здійснювати рух ігрового персонажа, також у правому верхньому кутку знаходяться дві кнопки. Кнопка pause відповідає за завершення чи перезавантаження рівня гри, кнопка on/off відповідає за програвання музики. Готовий варіант макету зображений на (рис.3.4).

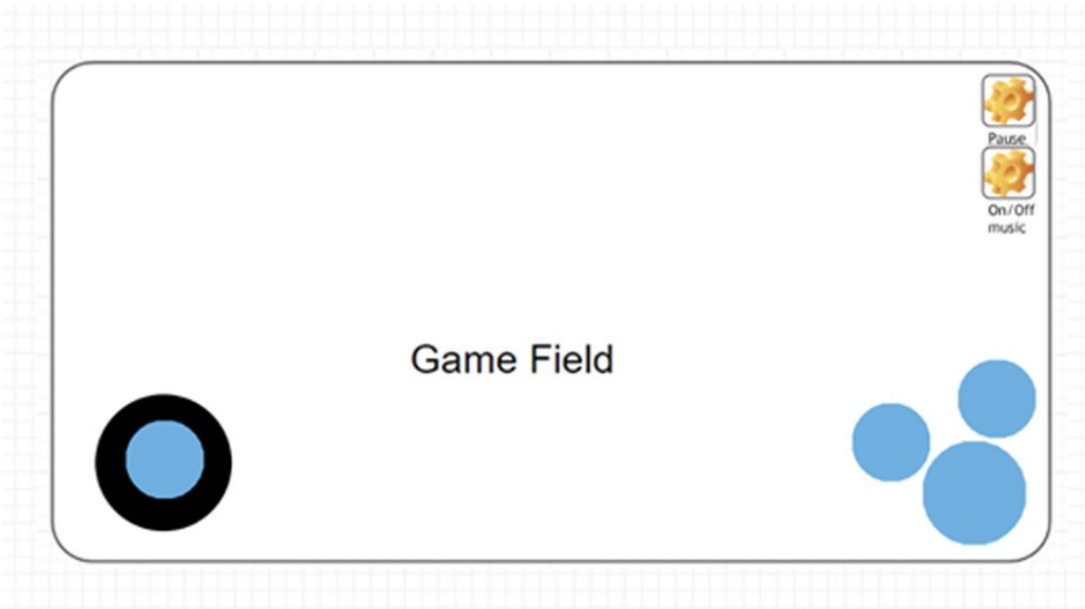

Рисунок 3.3 – Макет ігрового інтерфейсу

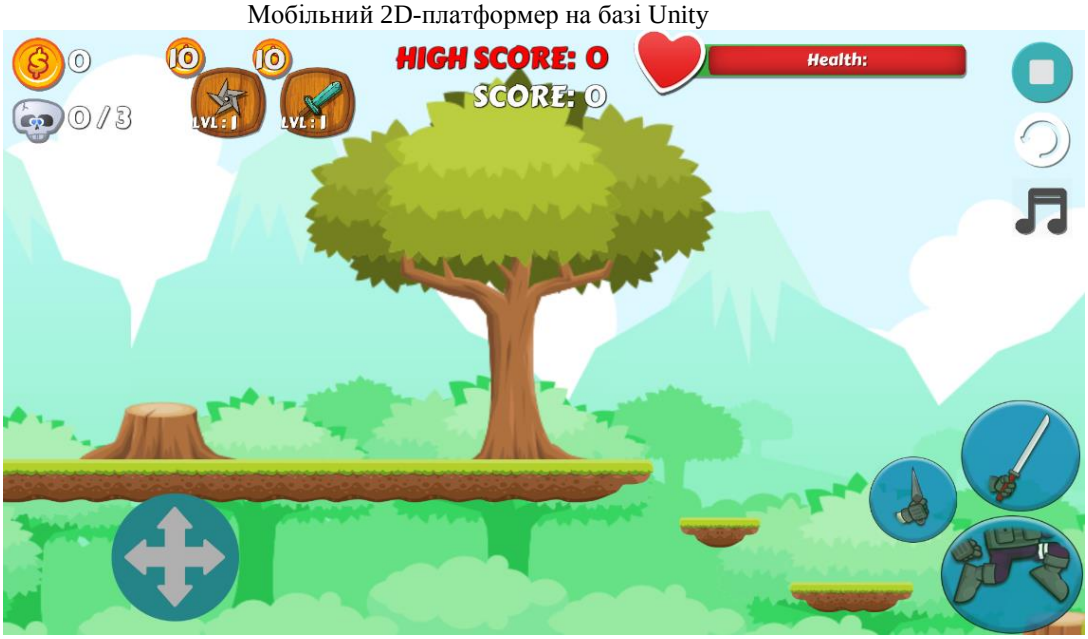

Рисунок 3.4 – Реалізація головної сцени гри

В наступному інтерфейсі зображено макет та реалізація Паузи в грі. На (рис. 3.5) зображений макет інтерфейсу.

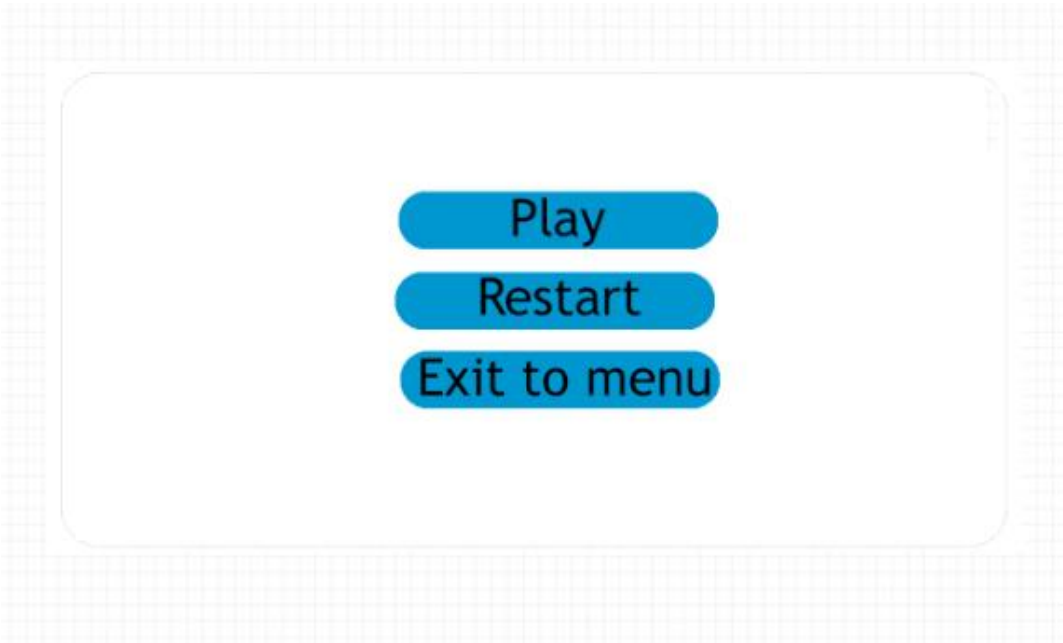

Рисунок 3.5 – Макет інтерфейсу паузи гри

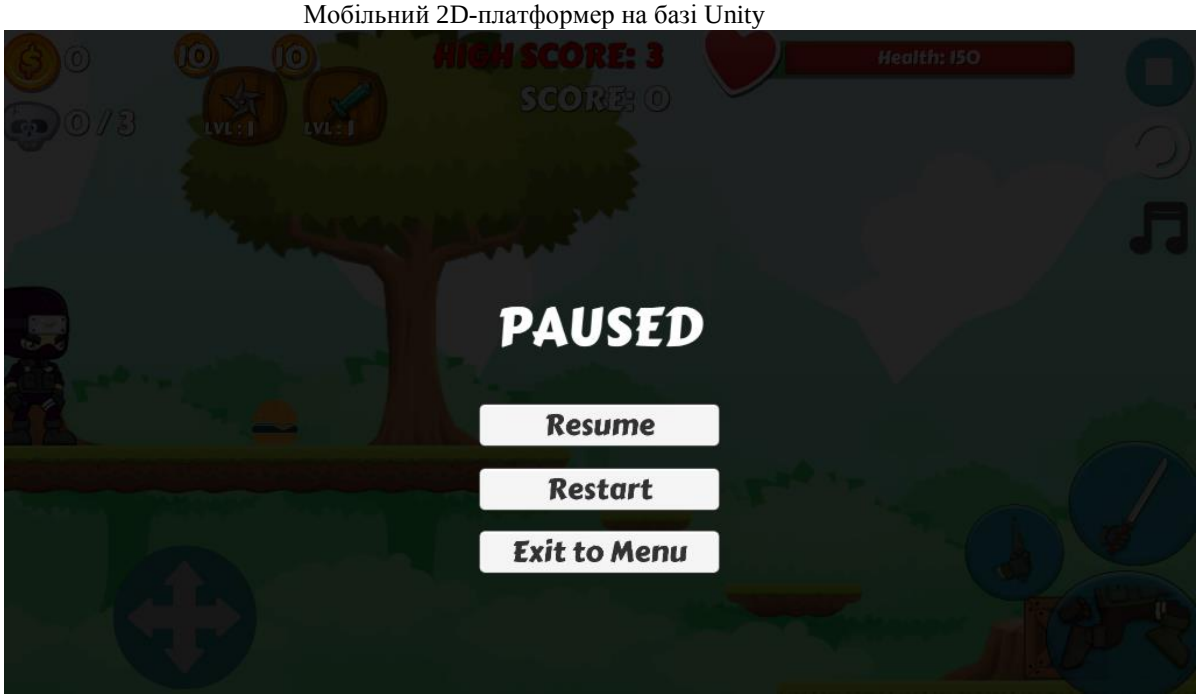

Рисунок 3.6 – Реалізація інтерфейсу паузи гри

На (рис 3.6) реалізований інтерфейс продемонстровані кнопки завдяки котрими користувач може продовжити гру чи перезавантажити рівень або вийти в головне меню.

3.2 Обґрунтування вибору середовища розробки

Починаючи розробляти будь–який програмний застосунок, замислюєшся над тим, у якому середовищі треба створити той чи інший програмний продукт. Для створення ігор є велика кількість ігрових рушіїв, як можна собі уявити.

З кожним роком їх стає все більше і більше. Різноманітність полягає у тому, що у кожного є переваги і недоліки. Один дозволяє розробляти ігри в 2D (наприклад, конструкція 2, Corona, Cocos2D–X), інший у 3D (Unreal Engine, Unity 5, CryEngine). Деякі з них безкоштовні, а для використання інших необхідно сплачувати щомісячну підписку або щорічну підписку. У табл. 3.1 та табл. 3.2 буде коротко наведено плюси та мінуси таких рушіїв як: CryEngine, Cocos2D–x Unity 3D та Unreal Engine,.

Таблиця 3.2 – Переваги та недоліки рушіїв CryEngine та Cocos2D–x

	CryEngine		Cocos2D–x	
1	Переваги	Недоліки	Переваги	Недоліки
2	GameSDK (бібліотека Google для ігор)	Стара документація	Безкоштовний	Тільки 2д ігри
3	realtime render	Мало користувачів	Низький поріг входження	
4	Графіка	Маленький магазин ассетів	швидкість	
		Складний процес складання білда	Декілька мов програмування підтримує	

Таблиця 3.1 – Переваги та недоліки рушіїв Unity 3D та Unreal Engine

№	Unity3D		Unreal Engine	
	Переваги	Недоліки	Переваги	Недоліки
1	Багато користувачів	Оптимізація	Візуальне програмування	Велика ціна на контент
2	Гарна документація	Руйнується білд через встановлення різних плагінів	Мультизадачність та універсальність	Мало універсального контенту
3	Кросплатформність	Часто ігри на телефони багато займають місця	Кросплатформість	Навантаження на ПК
4	Низький поріг входження		Гарана візуалізація	Великий поріг входження

Є ігрові компанії, які чомусь створили власні ігри. Деякі дозволяють створювати ігри на платформі, а це означає, що створення продукту на одній

платформі може бути перенесено на іншу. Наприклад, такі гіганти, як нереальний двигун, Кріггін та Єдність. Якщо ви зупинитесь на останніх трьох іграх, ви можете побачити деякі особливості кожного.

Якщо зупинитись на останніх трьох ігрових рушіїв, то можна виділити деякі особливості кожного із них.

CryEngine – один з найпопулярніших двигунів ігор, про що свідчить кількість "великих ігор", побудованих на своїй основі: Sniper II: Phantom Warrior, Giant, Ryse: Son of Rome, Cabal II, Far Cry. Це привернуло увагу розробників після кризи. Crysis показав новий графічний рівень з CryEngine. Пізніше було випущено кілька версій CryEngine, що підтримує нову функціональність..

Рисунок 3.7 – Логотип CryEngine

Unreal Engine – створено компанією Epic Games. Як і Unity, Unreal Engine дозволяє створювати відео – ігри для всіх операційних систем, консолей та мобільних платформ. [6]. Завдяки підтримці різних систем рендерінгу графіки OpenGL, Direct3D і т.д., підтримці різних звукових систем та можливостей мережевих ігор для Xbox Live, Windows Live і т.д. Unreal Engine можна використовувати для створення найрізноманітніших ігор, у тому числі

MMORPG (наприклад , на UE була створена Lineage II). Unreal Engine включає як комплект розробника (SDK), так і редактор.

Рисунок 3.8 – Логотип Unreal Engine

На відміну від Unity, Unreal Engine має з відкритим кодом і записується на C ++. Якщо ми поговоримо про порівняння та переваги Unreal Engine, вам потрібно зрозуміти, що єдність найкраще підходить для 2D/3D ігор на мобільних платформах, а UE4 дозволяє створювати розширену графіку. Нещодавно Epic Games намагається витягнути ковдру та вкрасти у 2D розробників та мобільних додатків, але статистика все ще показує, що вони все ще схильні вибирати єдність.

Що стосується складності, то тут можна сказати, що ігровий двигун розроблений для командної роботи, а стрибок у ньому дуже високий. Тому він не підходить для кількох розробників. Крім того, якщо ми говоримо про швидкість і якість, то якщо гра не велика, наприклад, мобільна, то Unity 3D все ще вигідніша. Зрештою, ви можете не відчути різниці, але швидкість виконання буде дуже помітною.

Unity 3D дозволяє розвивати ігри без спеціальних знань. Він використовує підхід, орієнтований на компонент, в якому розробник створює об'єкти (наприклад, головний герой) і додає до них різні компоненти

(наприклад, візуальний показ символу та способи його управління). Завдяки зручному інтерфейсу Drag & Drop та функціональним графічним редактором, двигун дозволяє намалювати картки та впорядкувати об'єкти в режимі реального часу та негайно перевірити результат.

Рисунок 3.9 – Логотип Unity

Друга перевага – це наявність величезної бібліотеки ослів і плагінів, з якою ви можете значно прискорити процес розвитку гри. Їх можна імпортувати і експортувати, цілі заготовки можуть бути додані до рівня гри, ворогів, патерни поведінки ШІ і так далі. Багато матеріалів доступні безкоштовно, інші пропонуються за невелику плату і за бажання ви можете створювати власний контент та публікувати його в Unity Asset Store і отримувати від цього прибуток.

Третя сильна сторона Unity 3D – підтримка великої кількості платформ, технологій, API[9]. Ігри створені на двигуні, можна легко імпортувати між Windows, Linux, OS X, Android, iOS, на консолі сімейств PlayStation, Xbox, Nintendo, на пристрої VR та AR. Unity підтримує DirectX та OpenGL, працює з усіма сучасними ефектами рендерингу, включаючи найновішу технологію реального часу Trace Trace.

Фізика твердих тіл, система Level of Detail, ragdoll і тканин, колізії між об'єктами, зіткнення між об'єктами, складними анімаціями – все це може бути

здійснено двигуном. Стереотипна думка про те, що двигун підходить лише для невеликих інді – ігор, і не в змозі видати прекрасну картину, вже не є актуальним: достатньо побачити техно – демо ADAM, Book of the Dead і The Blacksmith від творців середовища Unity.

І ще одна приємна відмінність, що навіть одна людина може розвинути гру. Це дуже хороша особливість, це означає, що поріг входу не такий високий, як той самий нереальний двигун. Комплекс цих факторів та особливостей спонукає невеликих компаній використовувати точно Unity 3D, а не будь –які інші ігрові рушійні рушії. У нашому випадку з тих же причин це Unity 3D.

Для того, щоб програма була дійсною, красивою, зрозумілою та якісною, вона повинна реалізувати основний архітектурний шаблон архітектурного шаблону MVC (модель–контроль) –ан), зазвичай використовується для створення інтерфейсів користувача, що ділиться на додаток на три взаємопов'язані частини.

Шаблон дизайну MVC ділиться цими основними компонентами завдяки повторному використанню коду та паралельної розробки. Ця архітектура, традиційно використовується для графічних інтерфейсів користувачів настільних ПК (GUI), стала популярною для розробки веб –додатків. Популярні мови програмування, такі як JavaScript, Python, Ruby, PHP, Java та C#, мають рамки MVC, які використовуються при розробці веб –додатків.

Модель складається з трьох частин:

– модель (дані, відокремлені від інтерфейсу користувача);

– контролер (дозволяє впливати на перегляд та передачу даних для обробки моделі, створеної діями користувача);

– вигляд (графічний інтерфейс користувача).

Unity 3D також підтримує MVC, що допомагає у правильній структурі ігрової архітектури.

Щоб полегшити з'ясування всіх можливостей таких рушіїв, як CryEngine, Unreal Engine, Unity3D та Cocos2D–X, створені таблиці. 3.3.

Таблиця 3.1 – Опис чотирьох ігрових рушіїв

		CryEngine	Unreal Engine	Unity3D	Cocos2D–x
1	Scripting language support	C++	C++	C#, JavaScript, Boo	C++, Lua, Javascript
2	Target platforms	PC, Xbox+, PlayStation3	Cross–platform	Cross–platform	Cross–platform
3	Documentation	https://docs.cryengine.com/	https://docs.unrealengine.com/en–US/index.html	https://docs.unity3d.com/Manual/index.html	https://docs.cocos2d–x.org/api–ref/
4	Free to use	+	+	+	+
5	Ease of use	–	–	+	+
6	2D & 3D Support	3D	3D	+	2D
7	Community	https://forum.cryengine.com/	https://communities.unrealengine.com/	https://forum.unity.com/	https://cocos2d–x.org/community
8	Asset Store	https://www.cryengine.com/marketplace	https://www.unrealengine.com/marketplace/en–US/store	https://assetstore.unity.com/	

Кінець Таблиці 3.1

9	Integrated Ad Monetization Framework	–	In–Game Ads	Unity Ads	Admob

3.3 Обґрунтування вибору платформи

Сьогодні майже у кожного є смартфон, майже кожен залежить від цього, хтось грає в ігри, хтось використовує соціальні мережі, для яких телефон є методом заробітку, і для якого це лише форма спілкування. Тому створення додатків для смартфонів – це дуже вигідний бізнес. Але виникає питання, яку платформу для розробки краще вибрати, Android або iOS. Ці титани вже давно борються за лідерство, всі намагаються з усією силою, щоб випередити свого опонента і довести йому та всьому світі, що вони є лідерами. Але поки що неможливо сказати, хто з них краще, всі вони мають свої плюси і мінуси. Як ми всі знаємо, власники Android набагато більші за яблука, через ціну, але в той же час, iOS–смартфони мають власні стандарти, і в той же час Apple дуже ретельно перевіряє всі відносини, перш ніж випустити їх у яблука . . «Частіше ви можете знайти зіпсований продукт, ніж Android, можливо, тому продукти Apple вважаються статусом.

Через те, що додатки значною мірою ретельно перевіряються та здійснюються вручну, а не автоматично, як і з Android, конкуренція між розробниками в магазині додатків (Apple App Store) не така велика, як на ігровому ринку (від Android) , також важливим фактором кількості питань додатків – ціна для облікового запису розробника. На ігровому ринку це символічні 25 доларів, які ви повинні платити один раз, і ви можете використовувати рахунок стільки разів, скільки захочете. Що ще можна сказати про магазин додатків, де щороку потрібно платити 100 доларів, це якась гарантія, що розробник добросовісно є як своїм продуктом, так і користувачам. Тому, якщо ви виберете будь –яку тему програми, то на ринку Play буде 100 варіантів, а в магазині додатків до 20.

Ігрові компанії вибирають кілька тактики: або випускають одразу на дві платформи, або спочатку на iOS, а потім на Android, або лише одну платформу.

У першому випадку це можливо з достатньою кількістю коштів для підтримки обох платформ та бажання негайно отримати прибуток від двох мобільних гігантів.

Другий метод використовується з міркувань безпеки самого ігрового продукту. Наприклад, гра має покупки всередині програми, де ви можете придбати щось для реальної валюти, що може спростити геймплей. Враховуючи, що відсоток зламаних ігор на iOS набагато менший, ніж на Android, тому спочатку вигідніше випустити його на iOS, оскільки розробники зможуть отримати великий прибуток, а потім на Android, коли ігровий продукт вже окупився. Але з Android все сумно, будь –яка гра може бути порушена, але ви можете встановити моди, які дозволять вам купувати предмети безкоштовно, це може значно завдати шкоди успіху компанії, яка створила гру.

Що ж, останній метод використовується для ексклюзивності або відсутності фінансів, все може бути тут.

Для поточного завдання все –таки краще використовувати операційну систему Android.

3.4 Ознайомлення із базовим функціоналом Unity 3D

Інтерфейс Unity дуже простий і легкий в використанні, якщо ви знаєте наступні частини та кнопки, ви можете використовувати Unity 3D без зовнішньої допомоги. Наприклад, візьміть проект у розробці, опишіть в ньому основні компоненти інтерфейсу користувача:

Рисунок 3.10 – Вікно Unity 3D

Опишемо основні елементи вікна застосунку:

1. Кнопка за допомогою якої можна обертатись у двовимірному простору на ігровій сцені;
2. Кнопка яка дозволяє рухатись у двовимірному простору на ігровій сцені;
3. Кнопка для обертання об'єктів у двовимірному просторі;
4. Кнопка для зміни об'єктів об'єктів у двовимірному просторі;
5. Кнопка «Play» – розпочати гру;
6. Кнопка «Stop» – зупинити гру;
7. Кнопка «Next frame» – відтворювати гру по кадру;
8. Scene – це ігрова сцена для розробника. Всі об'єкти, моделі, системи частинок тощо;
9. Вікно «Game» – показує нам, як саме гравець побачить саму гру;
10. Hierarchy – ієрархія предметів у грі. Тут відображаються всі об'єкти на сцені гри;
11. Project – ієрархія проекту. Містить усе, що включає проект (бібліотеки, матеріали, ресурси, сценарії тощо);
12. Inspector – тут переносяться компоненти об'єктів на сцені.

Тепер, коли у нас є уявлення про інтерфейс користувача Unity 3D, ми можемо перейти до основних компонентів Unity 3D, які включають:

1. Scenes – ігрові місця;
2. Scripts – сценарій у вибраній мові програмування, з якою створюється механіка гри;
3. Sprites – двовимірні зображення об'єктів, символів, фонів тощо;
4. Prefabs – чорні ігрові предмети;
5. Models – 3D –моделі, виготовлені в інших застосунках.

До всіх об’єктів прикріплюється компонент. По замовченню цей об’єкт завжди має компонент Transform який містить у собі інформацію про розташування, кути та розміри цього об’єкту.

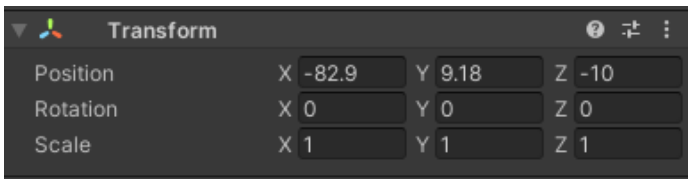

Рисунок 3.11 – Обов’язковий компонент будь якого об’єкту – Transform

Всі об’єкти які зберігаються проекті знаходяться в папці Assets, для більш легкого переміщення в проєкті існує вікно Project, що значно пришвидшує перехід між директоріями (рис.3.12).

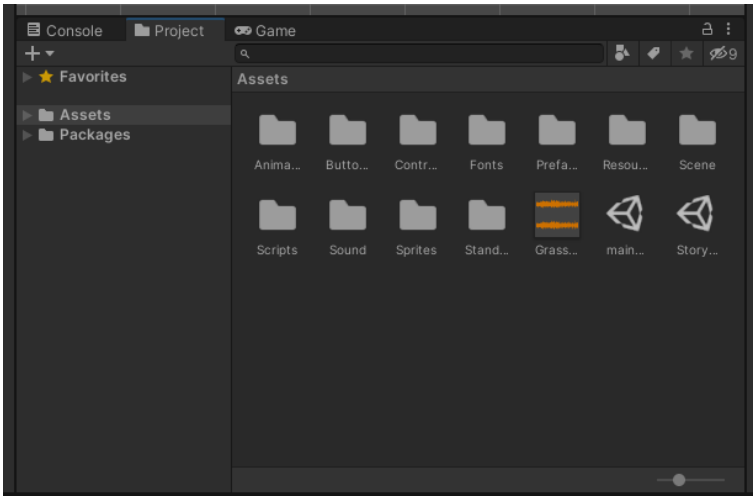

Рисунок 3.12 – Вікно Project

Щоб додати будь–який об'єкт на сцену, його потрібно перетягнути на ігрову сцену–це дуже зручно і швидко. Крім того, якщо ви хочете створити будь –який стандартний 3D примітив (куб, циліндр, сфера тощо) або 2D примітив (коло, квадрат тощо), потрібно натиснути з правою кнопкою миші. Вікно ієрархії та вибрати необхідне примітив у списку. Це зручно, якщо ще немає готових моделей, і необхідно перевірити механіку на таких примітивах – це прискорює та спрощує розвиток гри.

Ви можете додати інші компоненти до будь –якого об'єкта. Для цього виділіть об’єкт, у вікні інспектора відкрито клацніть кнопку "Додати компонент" та знайдіть необхідні компоненти.

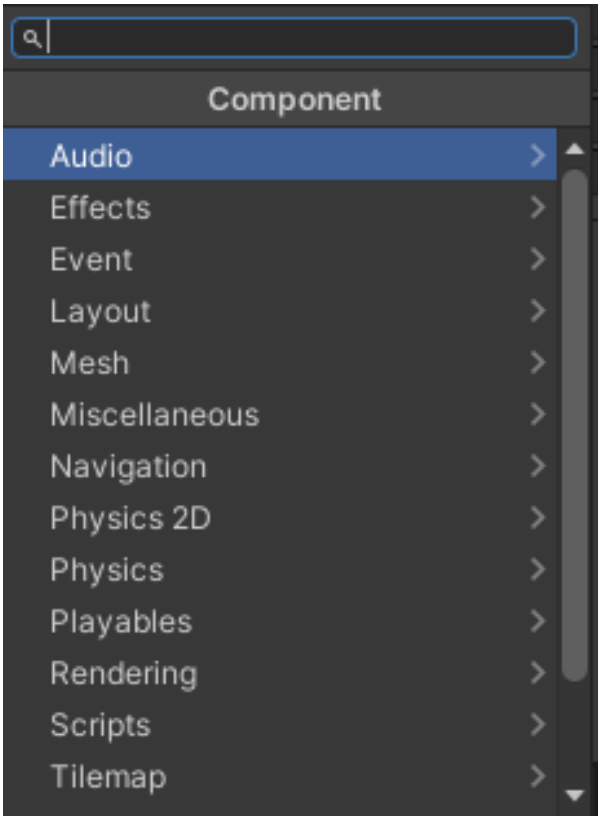

Рисунок 3.6 – Список різних класифікацій компонентів

Одним із найпоширеніших та часто використовуваним компонентом є:

1. Mesh Filter –задає форму 2д об’єкту;
2. Mesh Renderer – відображає 2д об’єект;
3. Mesh Collider – створює сітку навколо об’єкту в залежності від його Mesh Filter. Ця сітка допомагає об’єктам взаємодіяти один із одним;

4. Light – потрібен для відтворення світла та тіней;
5. Audio Source – існує для відтворення звуків;
6. Rigidbody – існує для відтворення фізики, а саме гравітації;
7. Camera –компонент який призначений для відображення ігрової сцени, завдяки цьому гравець може бачити гру;
8. Script – це файл коду, який прикріпляють до ігрового об’єкту, щоб він задав потрібну поведінку обєкту, яка написана в скрипті.

Висновки до розділу 3

В цьому розділі спроєктовано Platformer – відео – гру, також розглянуто і проаналізовано платформи для створення застосунків. Було проведено аналіз з виявленням переваг та недоліків, для розробки додатків було обрано платформу Unity та мову програмування C#. Саме ця платформа, і ця мова програмування буде найбільш практичною.

Також було продемонстровано користувацькі інтерфейси на яких продемонстровано як користувач може взаємодіяти с грою і були показані фінальні варіанти користувацького інтерфейсу вже у готовому проєкті.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

4.1 Керування героєм

Щоб почати створювати гравця, необхідно створити для нього збір (рис. 4.1), який створюється на початку гри.

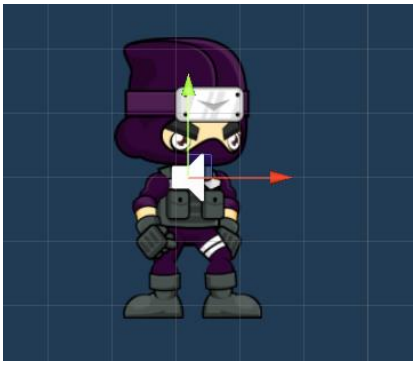

Рисунок 4.1 – Prefab гравця

Сам Prefab – це порожній об’єкт з collider для обробки зіткнення та панелі, яка буде містити предмети для зброї та панель для наслідків гравця. Об'єкт гравця створений у гравця, цей контролер включає гравця, його зброю, а також містить навички та методи вдосконалення цих навичок. Власний рух гравця відбувається через радість джойстика.

```
void UpdateVirtualAxes(Vector3 value)
{
    var delta = m_StartPos - value;
    delta.y = -delta.y;
    delta /= MovementRange;
    if (m_UseX)
    {
        m_HorizontalVirtualAxis.Update(-delta.x);
    }

    if (m_UseY)
    {
        m_VerticalVirtualAxis.Update(delta.y);
    }
}

void CreateVirtualAxes()
{
    // set axes to use
    m_UseX = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyHorizontal);
    m_UseY = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyVertical);

    // create new axes based on axes to use
    if (m_UseX)
    {
        m_HorizontalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(horizontalAxisName);
        CrossPlatformInputManager.RegisterVirtualAxis(m_HorizontalVirtualAxis);
    }
    if (m_UseY)
    {
        m_VerticalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(verticalAxisName);
        CrossPlatformInputManager.RegisterVirtualAxis(m_VerticalVirtualAxis);
    }
}
```

Рисунок 4.2 – Метод для пересування гравця.

4.2 Створення ворогів

Щоб створити ворога спочатку потрібно розробити prefab (рис. 4.3).

Рисунок 4.3 – Prefab ворога

У ворога також є компонент Collider для обробки зіткнень, Rigidbody, який також відповідає за фізику об'єкта. EnemyController контролер який відповідає за створення самих ворогів, для цього є 3 функції, початок створення, отримання позиції творіння та самого творіння.

```
public override void Start()
{
    //enemyRigidbody = GetComponent<Rigidbody2D>();
    currentStoryLevel = GameManager.Instance.StoryLevelUnlock;
    base.Start();
    Player.Instance.Dead += new DeadEventHandler(RemoveTarget);
    ChangeState(new IdleState());
    healthCanvas = transform.GetComponentInChildren<Canvas>();
    startPos = new Vector3 (transform.position.x, transform.position.y + 1.5f, transform.position.z);
    source = GetComponent<AudioSource>();
}

public void RemoveTarget()
{
    Target = null;
    ChangeState(new PatrolState());
}

//FUNCTION: Face the correct direction
private void LookAtTarget()
{
    if (Target != null)
    {
        float xDirection = Target.transform.position.x - transform.position.x;

        if (xDirection < 0 && facingRight || xDirection > 0 && !facingRight)
        {
            //FIX FOR SPINNING ENEMIES
            if (Vector2.Distance(transform.position, new Vector2(transform.position.x, Target.transform.position.y)) > 5)
            {
                RemoveTarget();
            }
            ChangeDirection();
        }

    }
}
```

Рисунок 4.4 – методи для створення ворога

Самі дані для створення ворога потрібно також прописати у GameData (рис. 4.5). У ньому прописане початкова кількість здоров’я, шкода, скільки

гравець отримає очок, початковий розмір, швидкість, його тип та пересування також шлях до prefab та його унікальний колір.

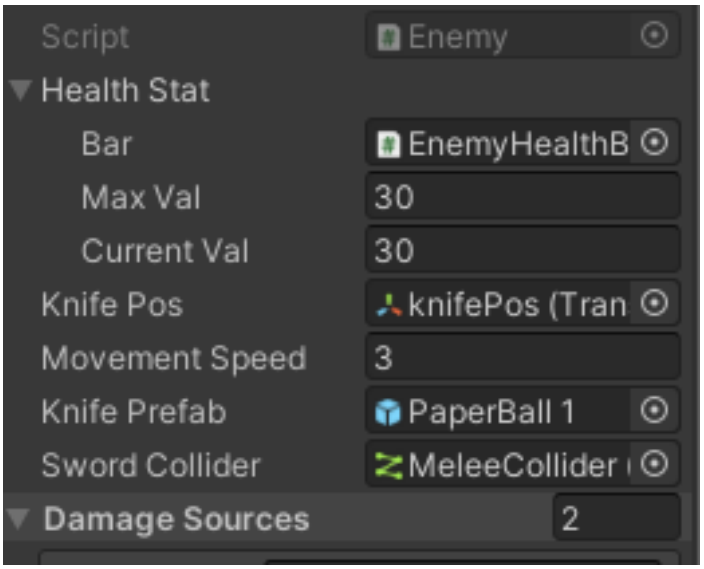

Рисунок 4.5 – Інформація про ворога

4.3 Система зброї

Для можливості боротьби с ворогом гравцю треба додати зброю, усього у гравця на є 2 види зброї, кожна с з них відтворює різні принципи дії: за допомогою меча гравець може підійти до ворога та застосувати меч та відняти 10 одиниць здоров’я у ворога, при використанні сюрикенів гравець може неподалік від ворога натиснути на кнопку пуску сюрикену та запустити його в ворога (рис. 4.6).

```
void Start()
{

    myRigidBody = GetComponent<Rigidbody2D>();
    StartCoroutine(Destroy());
}

private IEnumerator Destroy()
{
    yield return new WaitForSeconds(waitTime);
    //
    if (doFade)
    {
        StartCoroutine(Fadeout());
    }
    else
    {
        Destroy(gameObject);
    }
}

private IEnumerator Fadeout()
{
    float startAlpha = GetComponent<SpriteRenderer>().color.a;
    float rate = 1.0f / fadeTime;
    float progress = 0.0f; // edit this in the future

    while (progress < 1.0)
    {
        Color tmpColor = GetComponent<SpriteRenderer>().color;
        GetComponent<SpriteRenderer>().color = new Color(tmpColor.r, tmpColor.g, tmpColor.b, Mathf.Lerp(startAlpha, 0, progress));
        progress += rate * Time.deltaTime;
        yield return null;
    }
    Destroy(gameObject);
}

void FixedUpdate()
{
    myRigidBody.velocity = direction * speed;
}

public void Initialize(Vector2 direction)
{
    this.direction = direction;
}
```

Рисунок 4.6– Метод зброї

Також кожну зброю потрібно прописати у GameData (рис. 4.7). Тут прописано шлях до об’єкту зброї у prefabi гравця, швидкість постріла та тип зброї.

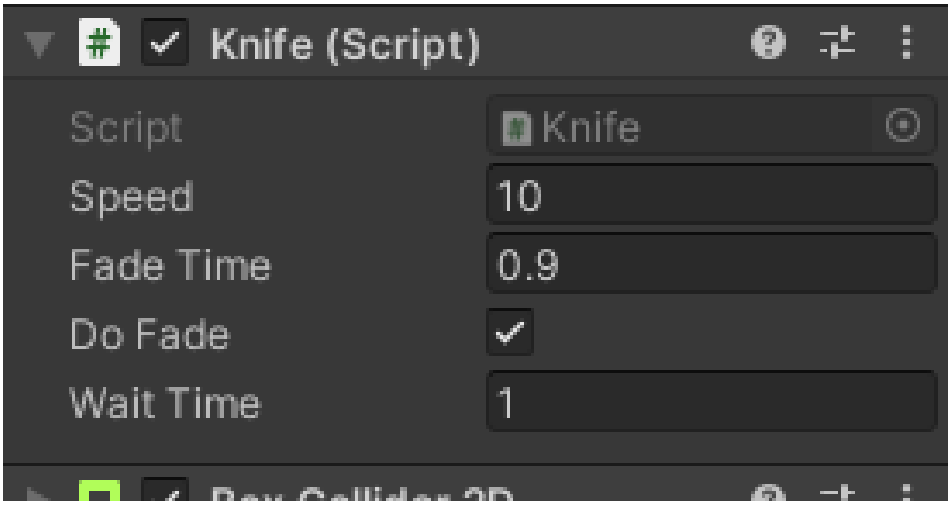

Рисунок 4.7 – Інформація про зброю Knife

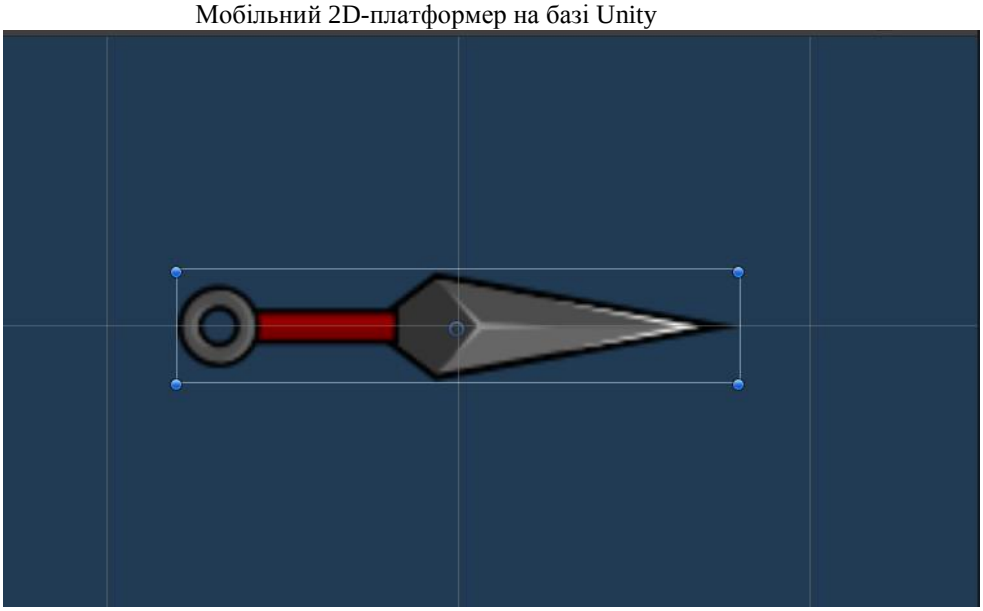

Рисунок 4.8 – Prefab зброї Knife

Реалізуємо систему озброєння для ворога, це буде паперова кулька (рис.4.9) за допомогою цього виду зброї ворог може наносити знешкодження гравцю віддалено.

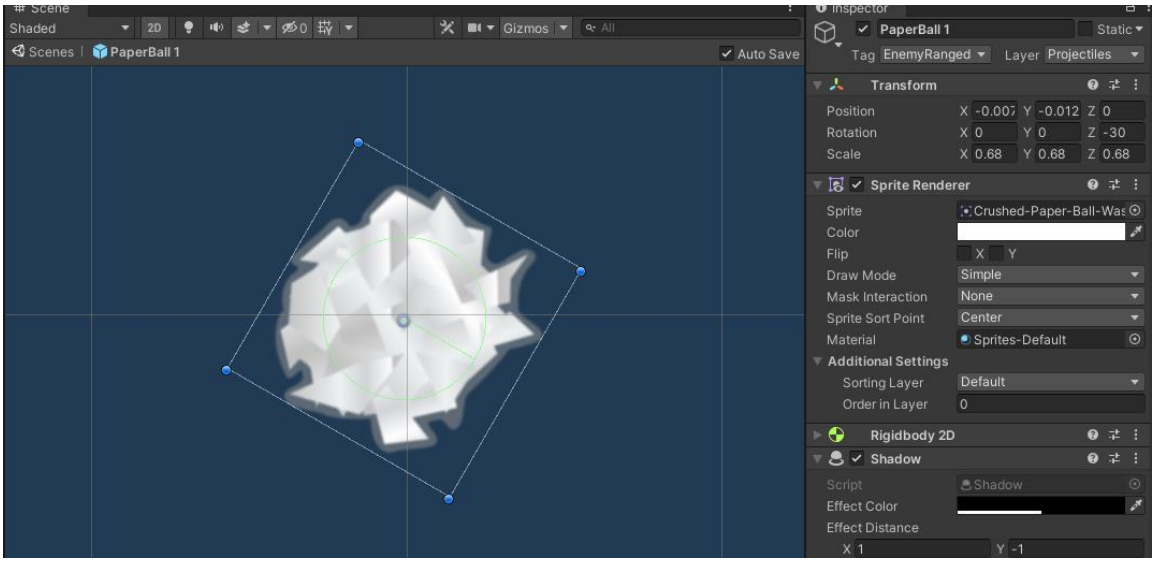

Рисунок 4.9 – Prefab паперової кульки

```
void Start()
{

    myRigidBody = GetComponent<Rigidbody2D>();
    StartCoroutine(Destroy());
}

private IEnumerator Destroy()
{
    yield return new WaitForSeconds(waitTime);
    //
    if (doFade)
    {
        StartCoroutine(Fadeout());
    }
    else
    {
        Destroy(gameObject);
    }
}

private IEnumerator Fadeout()
{
    float startAlpha = GetComponent<SpriteRenderer>().color.a;
    float rate = 1.0f / fadeTime;
    float progress = 0.0f; // edit this in the future

    while (progress < 1.0)
    {
        Color tmpColor = GetComponent<SpriteRenderer>().color;
        GetComponent<SpriteRenderer>().color = new Color(tmpColor.r, tmpColor.g, tmpColor.b, Mathf.Lerp(startAlpha, 0, progress));
        progress += rate * Time.deltaTime;
        yield return null;
    }
    Destroy(gameObject);
}

void FixedUpdate()
{
    myRigidBody.velocity = direction * speed;
}

public void Initialize(Vector2 direction)
{
    this.direction = direction;
}
```

Рисунок 4.10 – Метод зброї ворога

З озброєнням гравцю буде важко знищити ворога, це збільшує інтерес до гри так як підвищується цікавість перемогти та знищити ворога.

4.4 Система предметів

При знешкодження ворога кидком сюрикену чи мечем ворог отримає зашкодження і якщо його здоров’я знизиться до 0, то він помирає та буде можливість, що створиться якійсь предмет монетки (рис.4.11).

```
if (other.gameObject.tag == "Coin")
{
    GameManager.Instance.CollectedCoins++; //future: coinval
    Destroy(other.gameObject);
    CombatTextManager.Instance.CreateText(transform.position, "+1 G", Color.yellow, false, 0f, true);
    source.PlayOneShot(coinSound, 0.38F);
}
```

Рисунок 4.11 – Створення предмету після знищення ворога

Також потрібно для предмету котрий буде падати с ворога створити свій prefab (рис. 4.12).

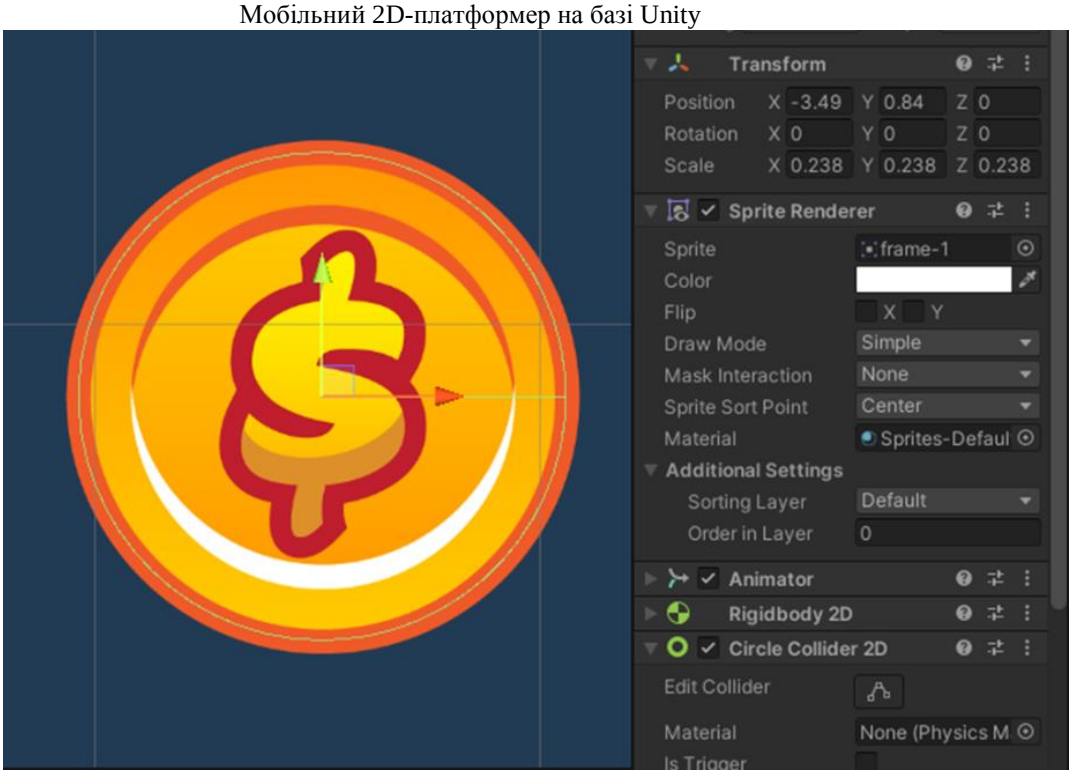

Рисунок 4.12 – Prefab монетки

За допомогою цього предмету монетки, гравець може придбати поліпшення для своєї зброї, та використовувати її в грі.

Також с імовірністю буде з’являтися предмет лікування персонажу, при підбиранні його персонаж буде отримувати 20 одиниць здоров’я (рис. 4.13) та (рис. 4.14).

```
if (other.gameObject.tag == "Burger")
{
    healthStat.CurrentVal += 20; //future: coinval
    Destroy(other.gameObject);
    CombatTextManager.Instance.CreateText(transform.position, "+20 HP", Color.green, false, 0f, true);
    source.PlayOneShot(eatSound, 0.2F);
}
```

Рисунок 4.13 – Створення предмету

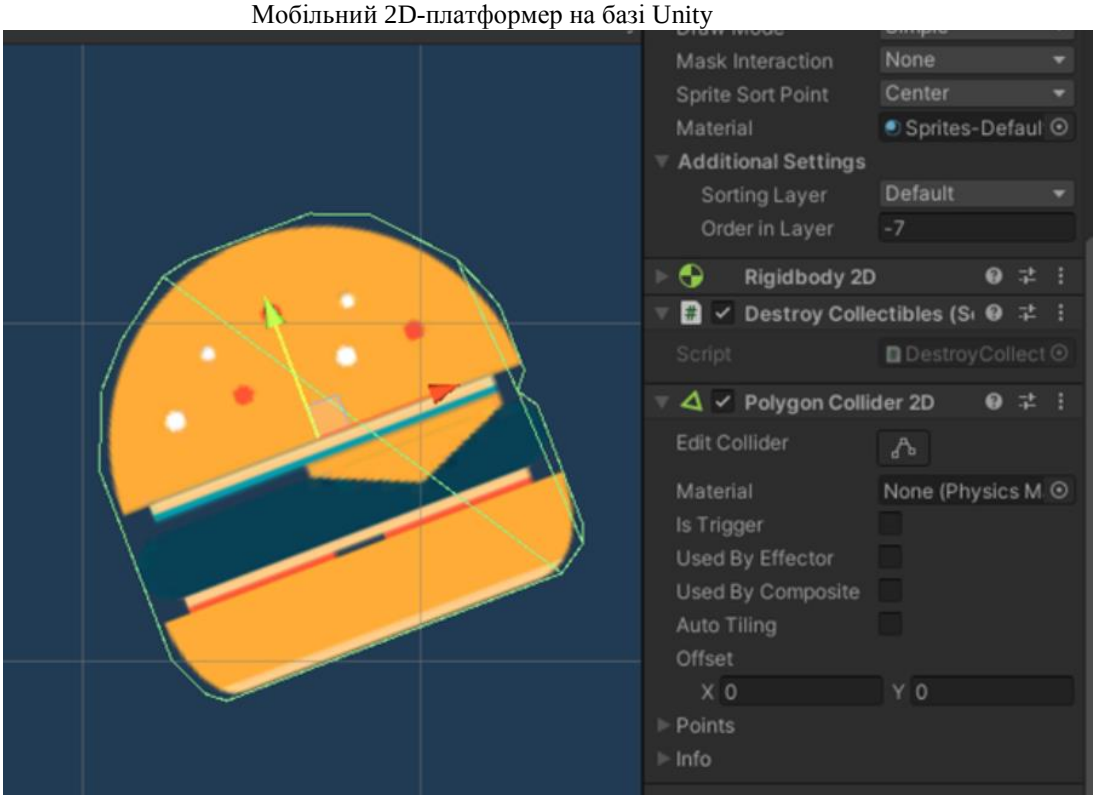

Рисунок 4.14 – Prefab бургера

Також в грі створено об’єкти з якими гравець може взаємодіяти (рис.4.15).

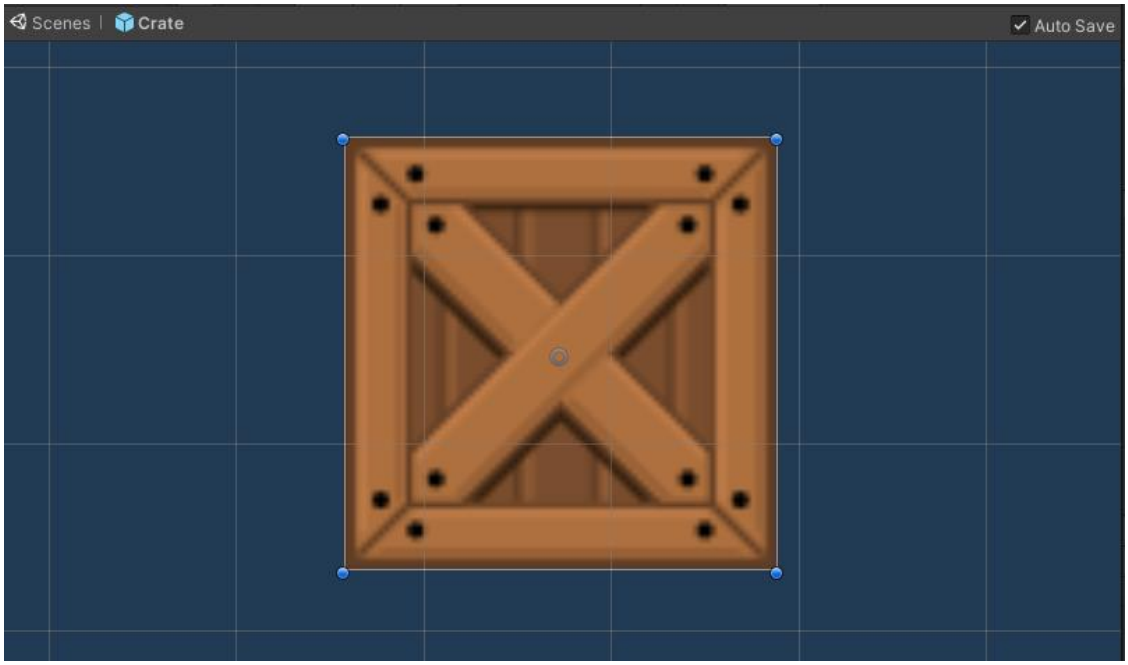

Рисунок 4.15 – Prefab ящику

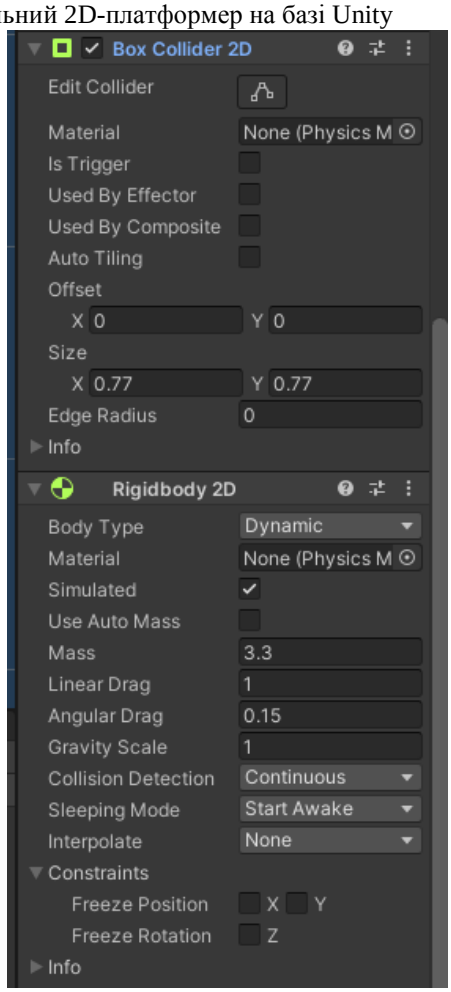

Рисунок 4.16 – Налаштування коллайдера ящику

До предмету ящик застосований box collider 2d завдяки котрому і утворюється взаємодія (рис.4.16).

Також в гру додано платформи по котрим гравець буде рухатися, всього в грі буде три платформи маленька (рис. 4.17), середня (рис. 4.18) та велика (рис. 4.19).

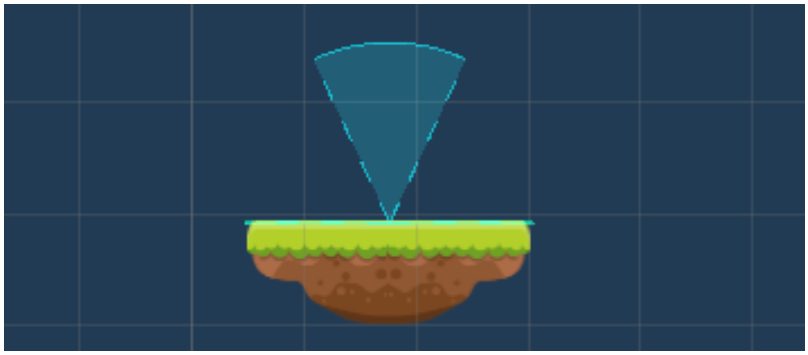

Рисунок 4.17 – Prefab маленької платформи

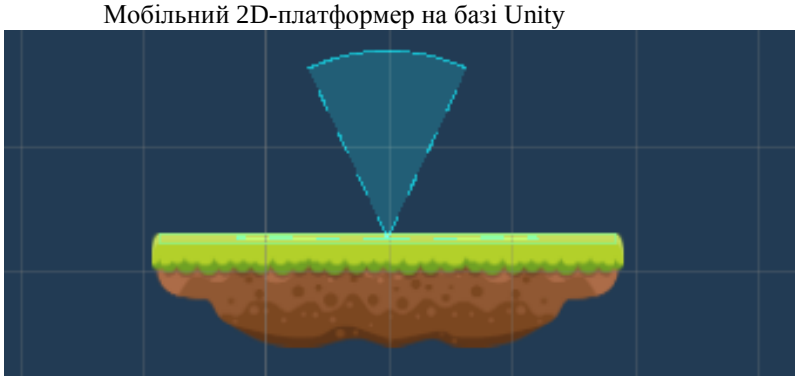

Рисунок 4.18 – Prefab середньої платформи

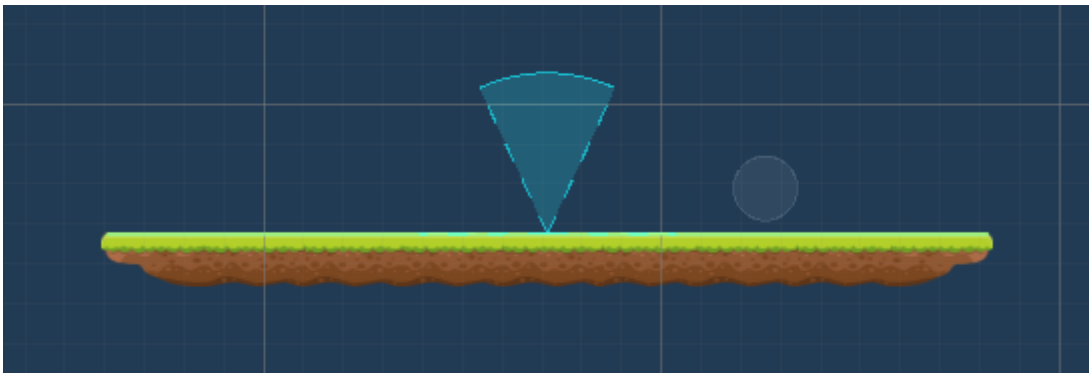

Рисунок 4.19 – Prefab великої платформи

Взаємодія с платформами відбувається завдяки Platform Effector 2D та Box Collider 2D (рис.4.20).

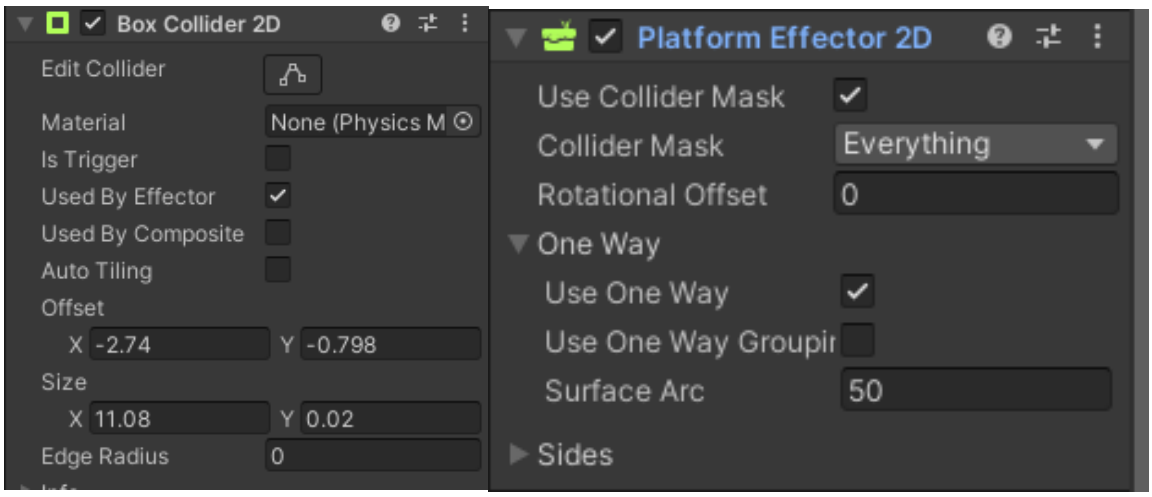

Рисунок 4.20 – Налаштування платформ

Для гарної атмосфери в грі було додано такі prefab як Дерево, пеньки та кущі (рис. 4.21).

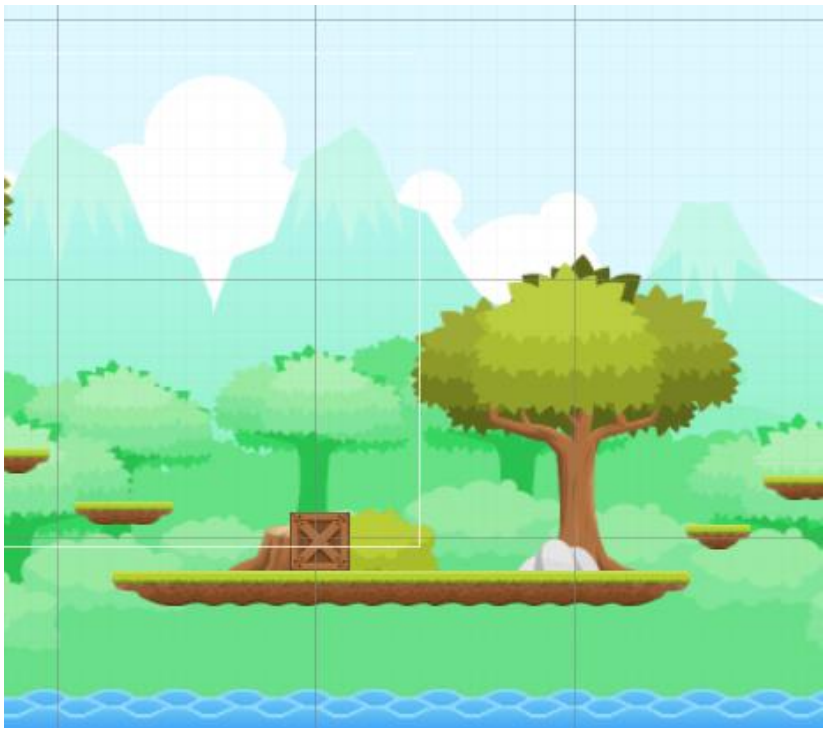

Рисунок 4.21 – Візуальний вигляд рівня

4.5 Система рівня та здібностей

При підбиранні монет до гравця передається його значення та підвищуються кількість монет. За ці монети гравець може придбати поліпшення для сюрикену або меча (рис. 4.22)

```
public void BtnUpgradePlayerMelee()
{
    if (GameManager.Instance.CollectedCoins >= GameManager.Instance.MeleeCost)
    {
        GameManager.Instance.PlayerMeleeDmg += 5;
        GameManager.Instance.CollectedCoins -= GameManager.Instance.MeleeCost;
        GameManager.Instance.MeleeLevel++;
        GameManager.Instance.MeleeCost = Mathf.FloorToInt(GameManager.Instance.MeleeCost * 1.23f);
        CombatTextManager.Instance.CreateText(transform.position, "Melee 5", Color.blue, false, 0f, true);
    }
}
public void BtnUpgradePlayerRanged()
{
    if (GameManager.Instance.CollectedCoins >= GameManager.Instance.RangedCost)
    {
        GameManager.Instance.PlayerRangedDmg += 3;
        GameManager.Instance.CollectedCoins -= GameManager.Instance.RangedCost;
        GameManager.Instance.RangedLevel++;
        GameManager.Instance.RangedCost = Mathf.FloorToInt(GameManager.Instance.RangedCost * 1.23f);
        CombatTextManager.Instance.CreateText(transform.position, "Ranged 3", Color.blue, false, 0f, true);
    }
}
```

Рисунок 4.22 – Метод поліпшення зброї

Коли гравець накопичив потрібну кількість монет, то він може придбати поліпшення, яке підвищує рівень зброї в гравця.

4.6 Завершення геймплею

Сам процес гри може продовжуватися довгий час, доки у гравець не програє. Для того щоб гравець він повинен використати усі свої життя і тільки тоді гра завершиться, і гравцю вийде повідомлення, розпочати гру заново чи вийти в головне меню (рис. 4.23).

```
override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
    deathTimer += Time.deltaTime;

    if (animator.GetComponent<Character>().tag == "Player")
    {
        if (deathTimer >= playerRespawnTime)
        {
            deathTimer = 0;
            animator.GetComponent<Character>().Death();
        }
    }
    else
    {
        if (deathTimer >= respawnTime)
        {
            deathTimer = 0;
            animator.GetComponent<Character>().Death();
        }
    }
```

Рисунок 4.23 –Метод здоров’я гравця.

4.7 Тестування та огляд користувацького інтерфейсу

Тестування – це тестування з метою перевірки функціональних вимог. Було проведено тести на пристрої Huawei P40 з операційною системою Harmony OS(Android).

Тест 1. Тестування відкриття сторінки рекордів.

Вхідні дані: Потрапити на головну сторінку гри.

Очікуваний результат: Перегляд рекорду який був набраний під час гри.

Отриманий результат: Відповідає реальності (рис. 4.1).

Рисунок 4.24 – Відкриття сторінки рекордів

Тест 2. Тестування меню виходу та перезавантаження рівня.

Вхідні дані: Для користувача відкривається вікно вибору з трьома кнопками.

Очікуваний результат:Після того як користувач обрав кнопку, натискає та потрапляє до обраного пункту з меню.

Отриманий результат: співпадає з очікуваним (рис. 4.25).

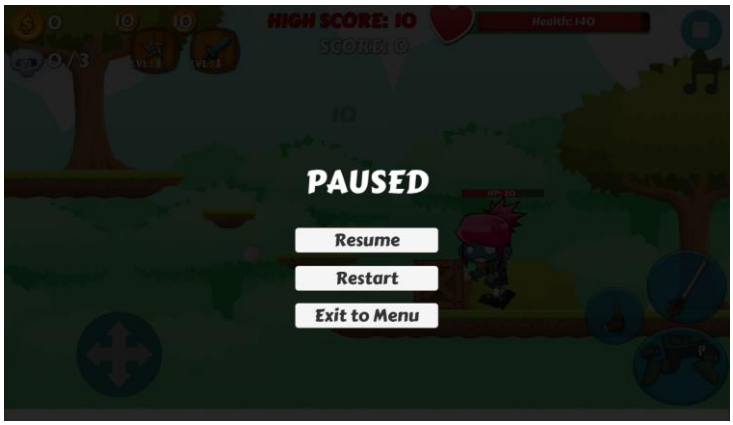

Рисунок 4.25 – Тестування кнопок переходу

Тест 3. Тестування здійснення кидка сюрикеном.

Вхідні дані: При повороті гравця до ворога здійснюється постріл (кидок) сюрикеном (рис.4.26.)

Очікуваний результат: Після того як ворог зустрівся із зброєю ворог отримую урон в 10 одиниць.

Отриманий результат: співпадає з очікуваним (рис. 4.26).

Рисунок 4.26 – Здійснення пострілу

Тест 4. Тестування отримання урону від ворога.

Вхідні дані: При підході гравця до ворога здійснюється кидок удар ворогом рис. 4.27.

Очікуваний результат: Після того як ворог наніс удар гравцю з гравця віднімається 10 одиниць здоров’я.

Отриманий результат: співпадає з очікуваним (рис. 4.27).

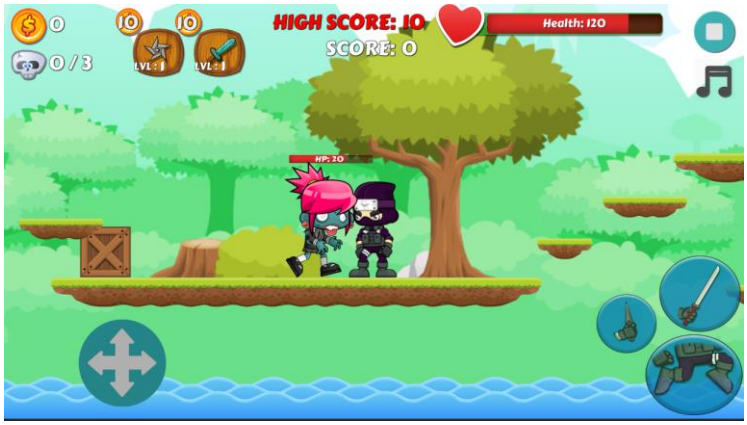

Рисунок 4.27 – Отримання шкоди від ворога

Тест 5. Тестування випадання монети з ворога.

Вхідні дані: При знешкодженні ворога отримання монетки рис. 4.28.

Очікуваний результат: Після того як ворог помер випадає монетка в 1 кількості.

Отриманий результат: співпадає з очікуваним (рис. 4.28).

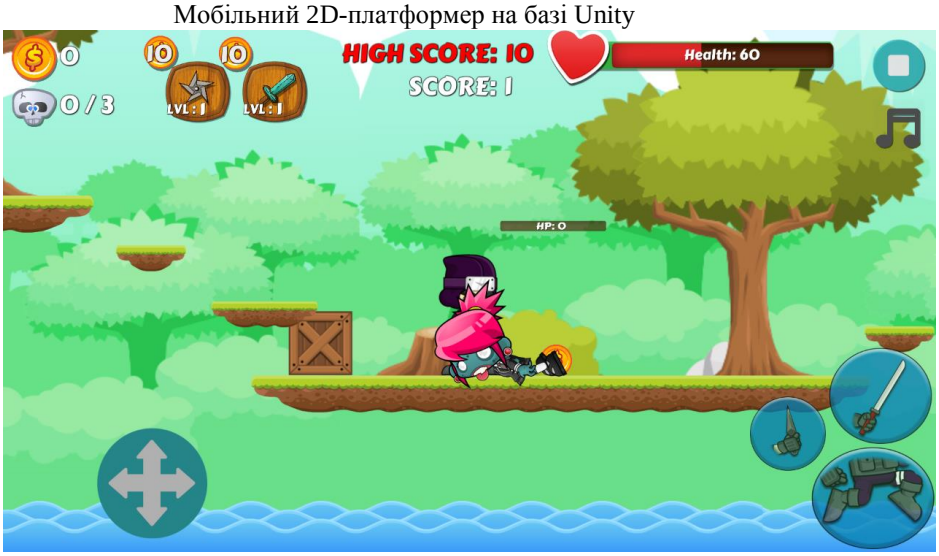

Рисунок 4.28 – Здійснення пострілу

Висновки до розділу 4

В цьому розділі було описано процес створення персонажів, ворогів, системи озброєння та покращення її та архітектури такі як головний скрипт для зв’язку, скрипт для обробки сутичок об'єктів та основних обробників для взаємодії з грою, інтерфейсом користувача та файлами на пристрої.

Також були розроблені основні елементи геймплея. Це створення гравців та їх управління. Був створений контролер, за допомогою якого створюються вороги. Сам клас ворогів також є абстрактним для легкого додавання нових ворогів з різними діями та рухами.

Крім того, ігрові тести проводилися серед аудиторії з різним ігровим досвідом. На основі цих тестів були визначені основні проблеми гри та були оброблені звіти для подальшого вдосконалення проекту до його виходу в Google Play.

ВИСНОВКИ

У рамках реалізації дипломної роботи було розроблено 2D ігровий додаток жанру Platform для мобільних пристроїв на операційній системі Android. Було проаналізовано існуючі рішення декількох інших ігрових застосунків з яких було сформовано основну ідею та механіку для гри.

Окремий розділ присвячений дизайну програмного забезпечення. Це один із найважливіших моментів у розробці програмного забезпечення. Це дозволяє побачити та уявити, як виглядатиме програмний продукт в кінці розробки. Наступний розділ присвячений створенню макета майбутнього інтерфейсу користувача. Також описано основні єлменти такі як вороги, предмети та здібності гравця які були реалізовані у грі. Також описано основні можливості гравця.

Окремий розділ був присвячений проєктуванню ігрового застосунку та було обрано середовище розробки. Серед чотирьох претендентів, був обраний ігровий рушій Unity 3D. Він легкий, але в той же час має велику кількість функціоналу. Завдяки кваліфікаційній роботі було здобуто велику кількість навичок у роботі із ігровим рушієм Unity 3D.

Також була протестована основна ігрова механіка гри та випробувана на обраній аудиторії, після чого було сформовано основні положення щодо покращення гри.

Поставлені задачі були виконані, мета кваліфікаційної роботи була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Інформація про гру Rayman Origins: вебсайт. URL:https://www.ubisoft.com/ru–ru/game/rayman/origins (дата звернення 20.04.2022).

2. Інформація про гру Hollow Knight: вебсайт. URL:https://store.steampowered.com/app/367520/Hollow_Knight/?l=russian (дата звернення 20.04.2022).

3. Інформація про гру Cuphead: вебсайт. URL: https://www.xbox.com/ru–RU/games/cuphead (дата звернення 20.05.2022).

4. Unity: вебсайт. URL: https://unity.com/ru (дата звернення 19.04.2022).

5. Cryengine: вебсайт. URL: https://www.cryengine.com (дата звернення 19.04.2022).

6. Unreal Engine: вебсайт. URL: https://www.unrealengine.com/ (дата звернення 19.04.2022).

7. Методичні рекомендації призначені для студентів 4–го курсу за спеціальністю 121 «Інженерія програмного забезпечення» галузі знань 12 «Інформаційні технології» під час виконання кваліфікаційних робіт на здобуття ступеня вищої освіти «Бакалавр». URL:https://moodle3.chmnu.edu.ua/mod/resource/view.php?id=472090 (дата звернення: 04.05.2021).

8. Переваги у роботі із Unity 3D. Житомирська політехнік, держаний університет. Тема презентація: Unity. Лекція 5. 23 с. URL: https://learn.ztu.edu.ua/pluginfile.php/159879/mod_resource/content/1/Лекція_5.pdf (дата звернення: 04.05.2022).

9. Документація по Unity URL: https://docs.unity3d.com/ru/current/Manual/index.html (дата звернення: 13.04.2022).

10. Вікіпедія: вебсайт. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 11.04.2022).

11. Eveegreen: вебсайт. URL: https://evergreens.com.ua/ua/articles/uml–diagrams.html https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 10.04.2022).

12. DOU: вебсайт. URL: https://dou.ua/lenta/articles/use–cases/ (дата звернення: 10.04.2022).