МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,

доцент _____Є. О. Давиденко
(підпис)

«__» ____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ЕКШН-ГРА НА ОСНОВІ РУШІЯ UNITY

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21910920

Студент: _____ Є. О. Твердий
(підпис)

«__»_____ 2023 р.

Керівник: канд. техн. наук, доцент _____Є. О. Давиденко
(підпис)

«__»_____ 2023 р.

Консультант: канд. техн. наук, доцент _____А. О. Алексєєва
(підпис)

«__»_____ 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра МогилиФакультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії програмного забезпечення, доцент, канд. техн. наук.
_____Є. О. Давиденко
«_____»_____20___р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Твердому Євгенію Олеговичу_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
_____«Екшн-гра на основі рушія Unity»_____
_____

Затверджена наказом по ЧНУ ім. Петра могили від 17»березня 2023 р. № 60

2. Строк представлення кваліфікаційної роботи «___»_____20___р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
_____Розроблена мобільна гра жанру екшн_____

4. Перелік питань, що підлягають розробці:
_____Дослідити існуючі аналоги, дослідити можливості та особливості роботи з двигуном Unity, розробити концепцію гри та її геймплей, інтерфейс та графічний дизайн, необхідні механіки та логіку гри, а також протестувати та відлагодити гру.

5. Перелік графічних матеріалів
_____Презентація_____

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
канд. техн. наук, доцент Алексєєва Анна Олександрівна	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи доцент, канд. техн. наук Давиденко Євген Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Твердий Євгеній Олегович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «_____»_____20_____р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема:_____«Екшн-гра на основі рушія Unity»_____
_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	3.11.2022р.	7.11.2022р.	виконано
2.	Огляд літератури за темою роботи	22.11.2022р.	25.11.2022р.	виконано
3.	Складання календарного плану КРБ	28.11.2022р.	30.11.2022р.	виконано
4.	Аналіз предметної області	01.12.2022р.	02.12.2022р.	виконано
5.	Розробка проєктних рішень	6.12.2022р.	09.12.2022р.	виконано
6.	Моделювання та конструювання ПЗ	14.12.2022р.	21.12.2022р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування	25.12.2022р.	28.04.2023р.	виконано
8.	Розробка спеціальної частини з охорони праці	16.05.2023р.	22.05.2023р.	виконано
9.	Відгук керівника КРБ	24.05.2023р.	24.05.2023р.	виконано
10.	Оформлення КРБ та презентації	26.05.2023р.	05.06.2023р.	виконано
11.	Попередній захист	07.06.2023р.	07.06.2023р.	виконано
12.	Рецензування	09.06.2023р.	13.06.2023р.	виконано
13.	Завершення оформлення КРБ та презентації	14.06.2023р.	14.06.2023р.	виконано
14.	Захист кваліфікаційної роботи	28.06.2023р.	28.06.2023р	виконано

Розробив студент _____Твердий Є. О._____ _____
(прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

Керівник роботи _канд. техн. наук, доцент Давиденко Є. О._ _____
(посада, прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Екшн-гра на основі рушія Unity»

Студент 409 гр.: Твердий Євгеній Олегович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Кваліфікаційна робота бакалавра присвячена дослідженню аналогів та розробці ігрового застосунку в жанрі екшн.

Об’єктом кваліфікаційної роботи є процес розробки екшн-гри за допомогою рушія Unity.

Предметом кваліфікаційної роботи є інструменти та технології розробки екшн-гри.

Метою кваліфікаційної роботи є популяризація розважальної сфери життя для підвищення позитивного настрою шляхом розробки екшн-гри з використання можливостей рушія Unity.

Кваліфікаційна робота складається з фахового розділу та спеціальної частини з охорони праці та безпеки в надзвичайних ситуаціях.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків, спеціального розділу з охорони праці та переліку джерел посилання.

У вступі визначається актуальність, науково-практичне значення обраної теми, мету і завдання роботи, а також об’єкт та предмет дослідження.

У першому розділі описується аналітична частина, тобто огляд існуючих аналогів та те завдяки чому було сформовано розуміння предметної області.

У другому розділі описується процес розробки проєктних рішень, які допомагають задовольнити вимоги до програмного забезпечення. Цей процес включає в себе моделювання об'єкту та предмету дослідження, а також створення функціональних та інформаційних моделей програмного забезпечення.

У третьому розділі описується процес розробки, вибір мови програмування, вибір рушія, розробка концепції гри, її геймплей, діаграм та інтерфейсу.

У четвертому розділі демонструється проведена з розробки робота.

У висновках проводиться аналіз проведеної роботи та отриманих результатів.

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

КРБ викладена на 63 сторінки, вона містить 4 розділи, 54 ілюстрацій, 4 таблиці, 17 джерел в переліку посилань.

Ключові слова: Unity, розробка ігор, C#

ABSTRACT

of the Bachelor’s Thesis

«Action game based on the Unity engine»

Student of group 409: Tverdyi Yevhenii Olegovich

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor

Davydenko Y. O.

The bachelor's qualification work is devoted to the study of analogues and the development of game applications in the action genre.

The object of the qualification work is the process of developing an action game using the Unity engine.

The subject of the qualification work is the tools and technologies of action game development.

The method of the qualification work is to popularize the improvement of the entertainment sphere of life for a positive mood by developing an action game using the capabilities of the Unity engine.

The qualification work consists of a professional section and a special part on occupational health and safety in emergency situations.

The explanatory note of the bachelor's thesis consists of an introduction, three sections, conclusions, appendices, a special section on labor protection and a list of reference sources.

In the introduction, the latest relevance, scientific and practical significance of the chosen topic, method and task of the work, as well as the object and subject of the research.

The first chapter describes the analytical part, i.e. the review of existing analogues and the way in which the understanding of the subject area was formed.

The second chapter describes the process of developing project solutions that help meet software requirements. This process includes the modeling of the object and subject of research, as well as the creation of functional and informational software models.

The third chapter describes the development process, the choice of a programming language, the choice of an engine, the development of the game concept, its gameplay, diagrams and interface.

The fourth chapter demonstrates the development work.

In the conclusions of the results of the analysis of the work carried out and the results obtained.

The special part on occupational health and safety in emergency situations deals with safety techniques when working in office premises with computer equipment.

qualifying work of the bachelor is laid out on 63 pages, it contains 4 chapters, 54 illustrations, 4 tables, 17 sources in the list of references.

Keywords: Unity, game development, C#

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КРБ – кваліфікаційна робота бакалавра

ОС – операційна система

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

UML – unified modeling language

UI – user interface

UX – user expirience

ВСТУП

Розробка ігор на основі рушія Unity є однією з найбільш актуальних і важливих задач в індустрії розваг в сучасному світі. Ігри стали дуже популярним видом розваг, що має величезний вплив на культуру і масову свідомість. Більшість сучасних ігор є продуктом колективної роботи, де команди розробників використовують різні інструменти і технології для створення ігрового світу.

Unity є одним з найпопулярніших інструментів для розробки ігор, який дозволяє розробникам створювати ігри для різних платформ, включаючи ПК, мобільні телефони та ігрові консолі. На даний момент Unity використовують мільйони розробників по всьому світу, що свідчить про його важливість і значення.

Розробка ігор на основі рушія Unity є складним технічним процесом, який вимагає високої кваліфікації та знань розробників. Проте, завдяки Unity, навіть початківці можуть створювати свої власні ігри з використанням готових шаблонів та ресурсів, що дозволяє розширити коло розробників та збільшити кількість створених ігор.

Таким чином, розробка ігор на основі рушія Unity є актуальною та має величезний науково-практичний та культурний вплив на сучасний світ. Вона не тільки дозволяє розробникам створювати захоплюючі ігри для користувачів, а й сприяє розвитку комп'ютерних технологій та підвищенню технічної культури в цілому.

Об’єктом кваліфікаційної роботи є процес розробки екшн гри на основі рушія Unity.

Предметом кваліфікаційної роботи є інструменти та технології розробки екшн-гри. Дослідження включає вивчення основних принципів розробки ігор на основі рушія Unity, проєктування ігрового процесу, створення графіки, анімації та звуків, налагодження і тестування гри на різних пристроях. В результаті

дослідження буде створена екшн гра, яка зможе бути використана як для розваг, так і для навчання та тренування навичок.

Метою кваліфікаційної роботи є популяризація розважальної сфери життя для підвищення позитивного настрою шляхом розробки екшн гри з використання можливостей рушія Unity.

Для досягнення цієї мети необхідно вирішити наступні завдання:

- дослідження існуючих аналогів;
- дослідити можливості та особливості роботи з рушієм Unity;
- розробити концепцію гри та її геймплей;
- розробити інтерфейс та графічний дизайн гри;
- розробити необхідні механіки гри та її логіку;
- протестувати та налагодити гру.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Відеоігри є одним з найбільш популярних форматів розваг у сучасному світі. За останні десятиліття відеоігри стали дуже популярними, і цей тренд продовжується. За даними досліджень (рис. 1.1), в 2022 році глобальний ринок відеоігор досяг близько 196 мільярдів доларів, що свідчить про значну популярність цього розважального формату.

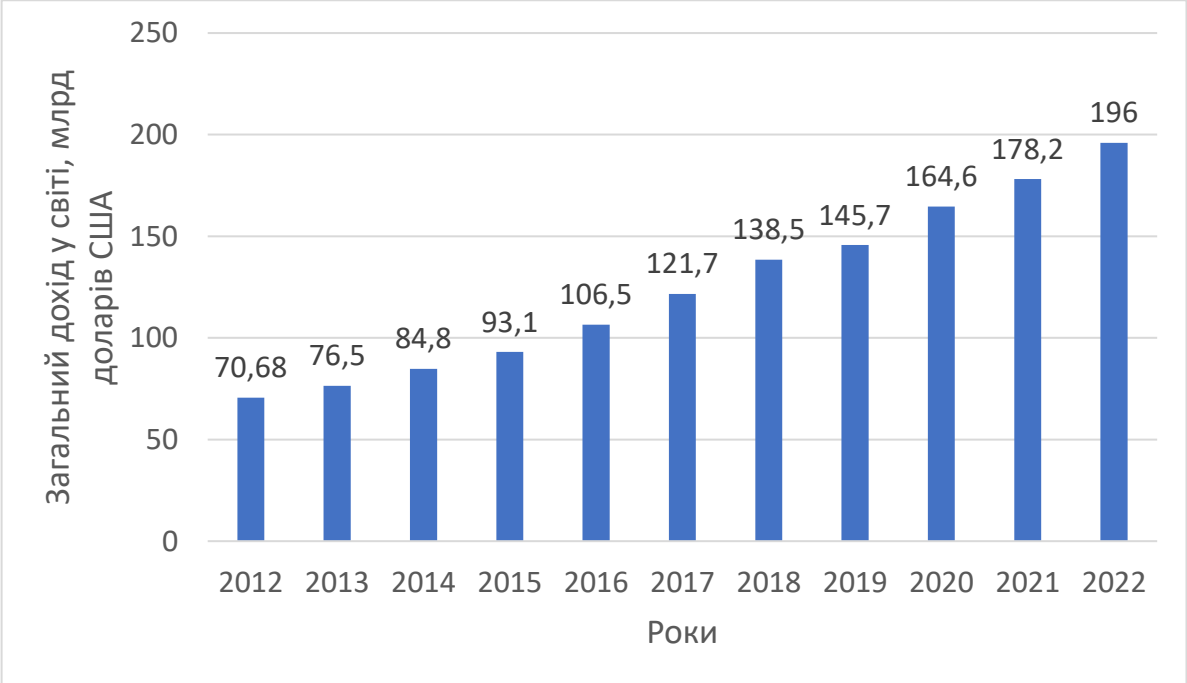


Рисунок 1.1 – Діаграма загальних річних доходів від відеоігор

Однією з причин популярності відеоігор є доступність цього формату. Ігри можна грати на різних платформах, таких як ПК, консолі, мобільні пристрої, а також в Інтернеті. Це означає, що відеоігри доступні для широкого кола користувачів.

Крім того, відеоігри надають можливість зануритися в іншу реальність та досліджувати уявні світи, що дозволяє користувачам відчути себе частинкою іншого світу. Відеоігри також надають можливість розважатися та відволікатися від повсякденних проблем.

Крім того, зростання популярності відеоігор також пов'язано з появою ігор з різноманітними жанрами, такими як екшн, рольові ігри, спортивні ігри, стратегії, імітації та інші. Це надає можливість користувачам вибирати гру, яка найбільше відповідає їхнім інтересам та вподобанням.

Загалом, популярність відеоігор у сучасному світі продовжує зростати, і цей формат стає все більш важливим елементом культури та розваг.

Також багато ігор розробляється на основі рушія Unity. Для розробки гри на основі ігрового рушія Unity нам знадобиться розуміння кількох різних аспектів:

– Програмування: Unity використовує мови програмування, такі як C# або JavaScript, тому вам потрібно мати базові знання цих мов програмування. Ви повинні знати, як створювати, читати та редагувати код. Вам також потрібно буде знати, які бібліотеки та фреймворки можуть допомогти вам покращити вашу роботу з кодом.

– Дизайн гри: Якщо ви розробляєте гру, вам потрібно мати знання про те, як створити відчуття гри та як зробити гру цікавішою для гравців. Вам також потрібно буде знати, які функції та елементи вам потрібні для гри.

– Моделювання та анімація: Unity має вбудовану підтримку 3D-моделювання та анімації, тому вам потрібно буде знати, як працювати зі 3D-моделями та як створювати анімації.

– Звук: Звук може значно покращити гру, тому вам потрібно буде знати, як працювати зі звуковими ефектами та музикою, щоб зробити вашу гру більш захоплюючою.

– Інтеграція з платформами: Якщо ви плануєте випустити свою гру на певній платформі, вам потрібно буде знати, як інтегрувати свою гру з цією платформою, наприклад, з Android або iOS.

До прикладу в 2015 році вийшла неймовірно гарна 2D гра на основі рушія Unity: Ori and the Blind Forest (рис. 1.2), від студії Moon Studios [1]. Сюжет якої

описує пригоди лісового духа Орі в подобі звірятка. Лише за перший рік було продано 420 копій гри в Steam.

Рисунок 1.2 – Скріншот геймплею гри

Гра випущена для таких платформ як Microsoft Windows, Xbox One, а з 2019 року й для Nintendo Switch.

Також дана гра перемогла на щорічній церемонії нагородження The Game Awards 2015 у номінації Best Art Direction, що говорить про те на скільки постарались розробники, дизайнери та художники гри у створенні свого проєкту.

Багато людей вважають що рушій Unity не є професійним рішенням, та годиться лише для створення низькопрофільних та не складних проєктів. Але аналізуючи попередньо написане можна з впевненістю сказати, що не важливо який рушій для створення гри буде обрано, важливо те на скільки команда, яка буде займатись розробкою того чи іншого проєкту, буде віддано, плідно та клопітно працювати й пророблять всі деталі від коду до дизайну, адже не можна зробити уклін на щось одне.

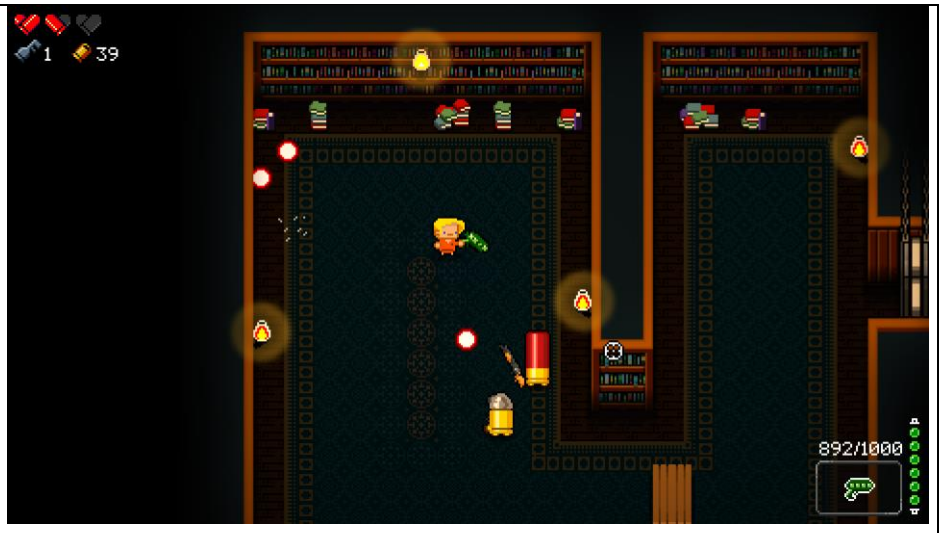
1.2 Аналіз аналогів

Enter the Gungeon

Таблиця 1.1 – Характеристика гри Enter the Gungeon

Назва характеристики	Опис
Назва	Enter the Gungeon
Розробник	Dodge Roll, Devolver Digital
Архітектура	Desktop application
Мова реалізації	Не відома
Переліік функцій, характеристик	1. Бігати по локаціям. 2. Стріляти у ворогів та перешкоди. 3. Оминати ворогів та перешкоди. 4. Керувати налаштуваннями геймплею, відео та аудіо. 5. 2D.
Аналіз переваг та недоліків	Переваги: 1. Багатокористувацький режим 2. Приємна графіка 3. Захопливі баталії Недоліки: 1. Шрифт тексту в грі важко читати, але згодом звикаєш

Кінець таблиці 1.1

Приклад зображень	 Рисунок 1.3 – Зображення геймплею
Джерело інформації [2]	https://store.steampowered.com/app/311690/Enter_the_Gungeon/

Серія ігор: Risk of Rain

Таблиця 1.2 – Характеристика серії ігор: Risk of Rain

Назва характеристики	Опис
Назва	Серія ігор: Risk of Rain
Розробник	Gearbox Publishing
Архітектура	Desktop application
Мова реалізації	Не відома
Переліік функцій, характеристик	1. Бігати по локаціям. 2. Стріляти у ворогів та перешкоди. 3. Оминати ворогів та перешкоди. 4. Прокачувати спорядження 5. Керувати налаштуваннями геймплею, відео та аудіо. 6. 3D.

Кінець таблиці 1.2

Аналіз переваг та недоліків	Переваги: 1. Багатокористувацький режим 2. Яскрава 3D графіка 3. Захопливі баталії Недоліки: 1. Є не значні баги у вигляді провалів текстур які іноді заважають ігровому процесу
Приклад зображень	 Рисунок 1.4 – Зображення геймплею
Джерело інформації [3]	https://store.steampowered.com/app/632360/Risk_of_Rain_2/

Vampire Survivors

Таблиця 1.3 – Характеристика гри Vampire Survivors

Назва характеристики	Опис
Назва	Vampire Survivors
Розробник	Poncle
Архітектура	Desktop application
Мова реалізації	Не відома

Кінець таблиці 1.3

Переліік функцій, характеристик	1. Бігати по локаціям. 2. Винищувати ворогів та перешкоди. 3. Оминати ворогів та перешкоди. 4. Прокачувати спорядження та персонажів 5. Керувати налаштуваннями геймплею, відео та аудіо. 6. 3D.
Аналіз переваг та недоліків	Переваги: 1. Захопливі баталії 2. Купа персонажів та зброї Недоліки: 1. Не дуже приємна графіка 2. Керувати треба лише переміщенням персонажа 3. Одноманітність 4. Однокористувацька гра
Приклад зображень	 Рисунок 1.5 – Зображення геймплею
Джерело інформації [4]	https://store.steampowered.com/app/1794680/Vampire_Survivors/

1.3 Специфікації вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням гри є принесення задоволення гравцям, покращення настрою та можливість відволіктись чи розслабитись після важкого дня.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення гри буде використано ігровий рушій Unity Engine.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 28.05.2023р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Цей застосунок призначений для того, щоб користувач міг проводити свій вільний час з дозвілля та відпочинку, використовуючи його.

Характеристика користувачів

Основні характеристики користувачів: наявність смартфону.

Загальні обмеження

Обмежень немає.

ФУНКЦІЇ СИСТЕМИ

Функція персональних рекордів

Опис функції

Функція покращення проходження викликає у гравця більше почуття азарту та задоволення від гри.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

Основним джерелом вхідної інформації є користувач.

Вимоги до способів організації, збереження та ведення інформації

Усі данні зберігаються у пам’яті смартфона.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

– ОС: Android 7 і вище

– Оперативна пам’ять: 4 GB

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура програмної системи складається лише з клієнтської частини.

Системне програмне забезпечення

Гра розроблена на платформі Unity з використанням мови програмування C#.

Мережне програмне забезпечення

Для створення застосунку було використано ОС Windows 11.

Мова і технологія розробки ПЗ

Гра була розроблена на платформі Unity з використанням мови програмування C#.

Інтерфейс користувача

Інтерфейс має задовольняти усі вимоги UI та UX дизайну, задля легкого розуміння користувачем роботи системи.

Апаратний інтерфейс

Апаратним інтерфейсом є смартфон користувача, з операційною системою Android 7, або вище.

Програмний інтерфейс

У ході розробки було використано дві категорії Unity Scripting API :

UnityEditor (Animations, Events тощо) та UnityEngine (Analytics, Audio тощо).

Комунікаційний протокол

Відсутній.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Гра є доступною для будь-якого користувача, за умови наявності у користувача апаратного інтерфейсу.

Супроводжуваність

Гра не потребує супроводжуваності.

Переносимість

Програмне забезпечення може працювати на ОС Android (7 версія та вище).

Продуктивність

Продуктивність роботи ПЗ залежить від характеристик смартфону.

Надійність

Відсутня.

Безпека

Відсутня.

ІНШІ ВИМОГИ

Усі вимоги сформовано та описано вище, доповнення не вимагається.

Висновки до розділу 1

Під час написання першого розділу було проведено дослідження популярності відеоігор у світі. Також проведено аналіз складності роботи з таким ігровим рушієм як Unity та розглянуто популярність ігор створених на основі цього рушія. За для створення кращої гри було проведено аналіз аналогів від інших розробників. Виявлено їх переваги та недоліки, архітектуру та мову реалізації за для розуміння що слід робити у власному проєкті. Після проведеного аналізу було встановлено специфікації вимог до програмного забезпечення.

На основі першого розділу можна у підсумку сказати, що розробити гарну гру доволі складно навіть для цілої команди розробників. Але завдяки цьому стало відомо яких помилок не варто припускатись.

2 АНАЛІЗ СИСТЕМИ

2.1 Створення сценаріїв

Коротка форма написання usecase для зарачі «Початок гри»:

Користувач захотів зіграти в гру та запускає її на своєму смартфоні. Гра запускається та відображає на екрані головне меню гри. Користувач натискає на кнопку «Грати» та система завантажує рівень гри. На екрані смартфону буде відображено карту гри, стік для керування персонажем, сам персонаж та кнопка паузи.

Поверхнева форма написання usecase для задачі «Початок гри»:

Головний сценарій (успішний):

Користувач захотів зіграти в гру та запускає її на своєму смартфоні. Гра запускається та відображає на екрані головне меню гри. Користувач натискає на кнопку «Грати» та система завантажує перший рівень гри або останній який зберігся, якщо гравець грає не вперше. На екрані смартфону буде відображено карту гри, стік для керування персонажем, сам персонаж та кнопка паузи.

Альтернативні сценарії:

1. Гравець не може зіграти в гру з останнього збереженого рівня, адже на смартфоні було замало місця для збереження інформації про гру.

2. Гравець не може зіграти в гру адже смартфон не проходить по мінімально необхідним вимогам до характеристик для гри.

Повна форма написання usecase для задачі «Початок гри»:

Таблиця 2.1 – Usecase

Use section	**Comment**
Use Case Name	Початок гри
Scope	Гра для смартфонів
Level	Пограти в гру на смартфоні
Primary Actor	Користувач

Продовження таблиці 2.1

Stakeholders and interests	1. Користувач. Зацікавлений в тому аби запустити та зіграти в гру
Preconditions	1. Користувач тримає розблокований смартфон в руках
Main Success Scenario	1. Користувач знаходить гру на робочому столі смартфона 2. Користувач запускає гру 3. Гра запускається 4. Гра відображає на екрані смартфону головне меню 5. Користувач натискає на кнопку «Грати» 6. Гра завантажує перший рівень гри 7. Гра відображає на екрані смартфона інтерфейс керування 8. Користувач натискає на стік керування персонажем 9. Система обробляє натискання та переміщає персонажа по карті гри 10. Система створює перешкоди та не ігрових персонажів як противників для користувача
Extensions	1. У будь який момент користувач хоче припинити грати 1) Користувач натискає на кнопку паузи 2) Користувач натискає на кнопку «Завершити гру» 3) Система зберігає дані про гру 4) Користувач бачe робочий стіл свого смартфону

Кінець таблиці 2.1

	2. Користувач вирішує змінити налаштування гри не виходячи з рівня 1) Користувач натискає а кнопку паузи 2) Натискає на кнопки ввімк/вимк звуку чи вібрації 3) Система оброблює натискання 4) Користувач повертається до рівня 3. Користувач вирішив змінити налаштування в головному меню 1) Користувач натискає на кнопки вмик/вимк звуку чи вібрації 2) Система обробляє натискання та змінює налаштування
Special Requirements	Смартфон що проходить мінімально необхідні вимоги для запуску гри
Frequency of Occurrence	Гра може працювати майже безкінечно
Miscellaneous	Збереження даних гри

2.2 Створення діаграм сценаріїв використання

Діаграми варіантів використання (usecase diagrams) призначені для відображення взаємодії між системою та користувачами, які використовують її функціонал [5]. Актори на діаграмі відображаються у вигляді символу людини, а варіанти використання – у вигляді еліпса. У варіантах використання описуються можливі дії користувачів та відповідні реакції системи на ці дії.

Розглянемо діаграму сценаріїв використання гри що проєктується (рис. 2.1).

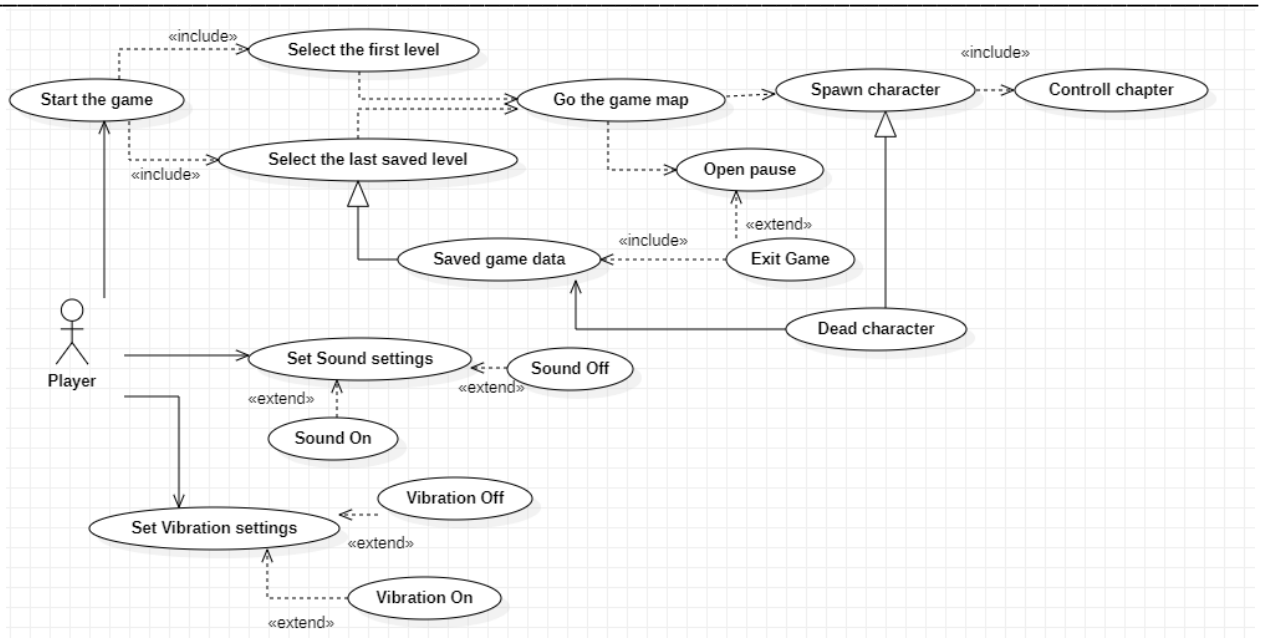


Рисунок 2.1 – Діаграма сценаріїв використання

На даній діаграмі зображено варіанти використання гри. Гравець може змінити налаштування звуку та вібрації, а також розпочати гру. Протягом гри гравець може вийти у меню паузи, та покинути гру. Також гравець може керувати ігровим персонажем.

2.3 Побудова діаграм взаємодії (послідовності та кооперації)

Діаграми послідовності є одним зі способів опису сценаріїв використання. Їх перевага полягає у можливості визначити взаємодію між компонентами системи та потоки повідомлень на ранніх етапах її розробки. Це дозволяє в подальшому перетворити ці компоненти та потоки повідомлень на конкретні класи (об’єкти) та методи цих об’єктів. Також за допомогою діаграм послідовності можна встановити, які події (Actions) модель системи буде підтримувати та обробляти [6].

Діаграма послідовностей є однією з діаграм взаємодії в UML, що дозволяє описати поведінкові аспекти системи, зосереджуючись на взаємодії об'єктів в часі. Основна мета діаграми полягає в відображенні часових аспектів передачі та прийому повідомлень між об'єктами системи. На діаграмі послідовності кожен об'єкт зображується у вигляді прямокутника, що розташований на вершині

пунктирної вертикальної лінії, що називається лінією життя об'єкту. Лінія життя є відображенням життєвого циклу об'єкту в процесі його взаємодії з іншими об'єктами системи.

Розглянемо діаграму послідовності для гри що проектується (рис. 2.2).

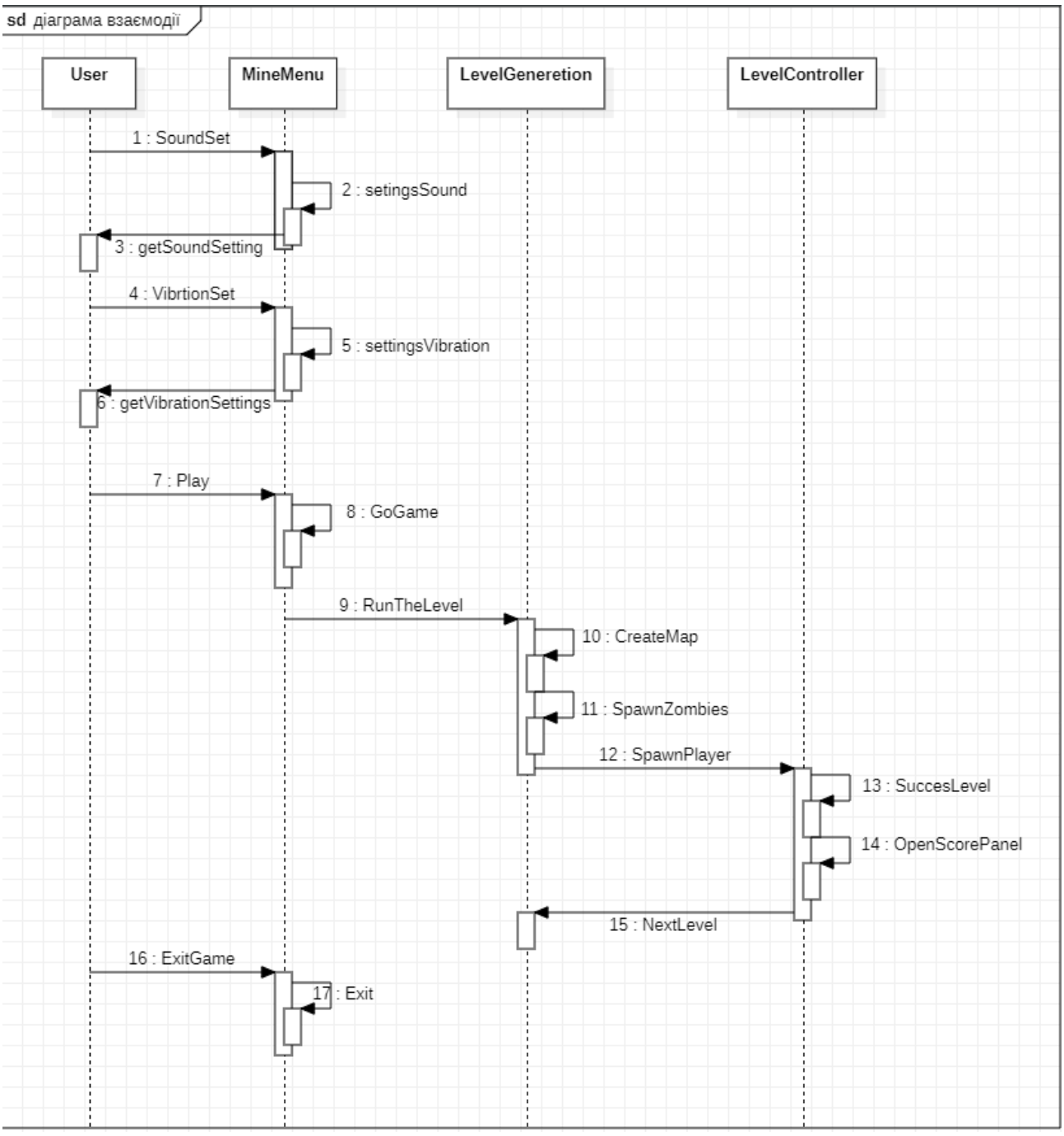


Рисунок 2.2 – Діаграма послідовності

На даній діаграмі показано взаємодію гравця з різними елементами гри, таких як головне меню, та саме рівні гри.

Якщо попередня діаграма служить для візуалізації тимчасових аспектів взаємодії, то діаграма кооперації призначена для специфікації структурних аспектів взаємодії. Головна особливість діаграми кооперації полягає в можливості графічно зобразити не тільки послідовність взаємодії, але і всі структурні відношення між об'єктами, що беруть участь в цій взаємодії.

Діаграма кооперації використовується для зображення взаємодії між об'єктами, які відіграють певні ролі у системі. На діаграмі відображаються об'єкти у вигляді прямокутників з їх іменами, класами та можливими атрибутами. Асоціації між об'єктами зображуються з'єднувальними лініями, і можуть бути підписані іменами асоціацій та ролями, які об'єкти відіграють у цих асоціаціях. Динамічні зв'язки, такі як потоки повідомлень, також можуть бути відображені на діаграмі [7].

Розглянемо діаграму кооперації для гри що проектується (рис. 2.3).

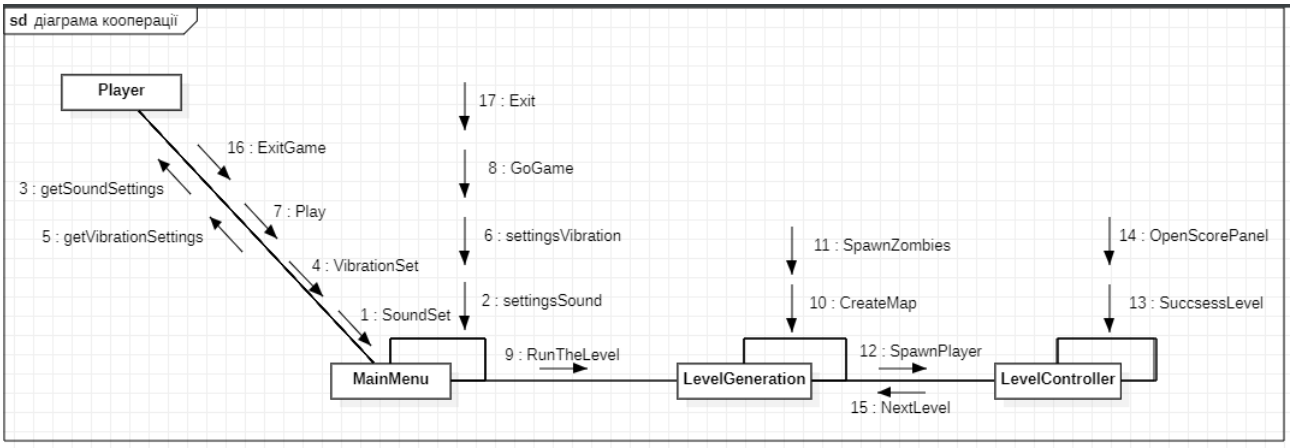


Рисунок 2.3 – Діаграма кооперації

На відміну від діаграми послідовності, діаграма кооперації не відображає час у вигляді окремого виміру, але може вказувати порядкові номери повідомлень для визначення послідовності взаємодій. Якщо потрібно більш детально описати взаємозв'язки між об'єктами в реальному часі, тоді краще використовувати діаграму послідовності.

2.4 Побудова діаграм станів

Діаграми станів та переходів (statechart diagrams) Діаграми станів та переходів є графічними інструментами, які використовуються для відображення складних процесів в системах [8]. Діаграма станів моделює поведінку об'єкта, який може перебувати у різних станах, а переходи вказують на події, які можуть відбуватися і переводити об'єкт з одного стану в інший. Такі діаграми допомагають краще розуміти, як система поводиться у різних умовах та які дії вона виконує в певних ситуаціях. Разом із діаграмами діяльності та взаємодії, діаграми станів та переходів допомагають визначити та уточнити вимоги до системи та спростити її розробку.

Елементами діаграми є:

- круг, що вказує початковий стан;
- круг з маленькою крапкою посередині, що вказує на кінцевий стан;
- стрілка, що вказує на перехід.
- назва події, що викликає перехід, відзначається над/під стрілкою.

Основна мета цієї діаграми – описати можливу послідовність станів і переходів, які разом характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма стану представляє динамічну поведінку об’єкта на основі специфікації відповіді об’єкта на певне сприйняття події. Системи, які реагують на зовнішні дії інших систем або користувачів, іноді називають реактивними. Якщо така дія розпочинається у випадковий момент години, тоді поговоріть про асинхронну поведінку моделі.

На загальній діаграмі станів (рис. 2.4) представлено загальний приклад функціонування гри.

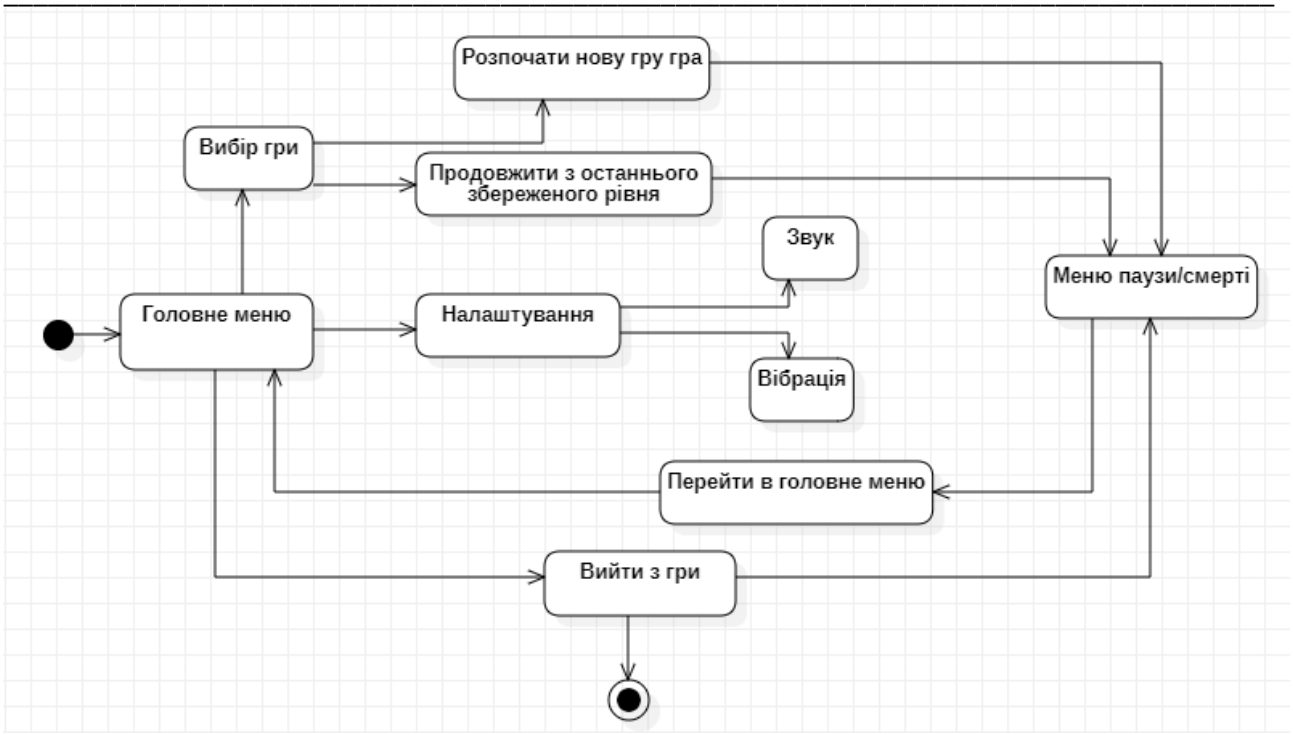


Рисунок 2.4 – Загальна діаграма стану

На загальній діаграмі стану показані такі можливості як вибір варіанту гри, де гравець може обрати почати йому нову гру чи продовжити ту що він не закінчив. Також є показана можливість зміни налаштувань а саме налаштувань звуку ти вібрації. Також в головному меню гри є можливість вийти з гри, яка також доступна і з меню паузи чи смерті гравця.

Наступною діаграмою зображена діаграма станів гравця (рис. 2.5).

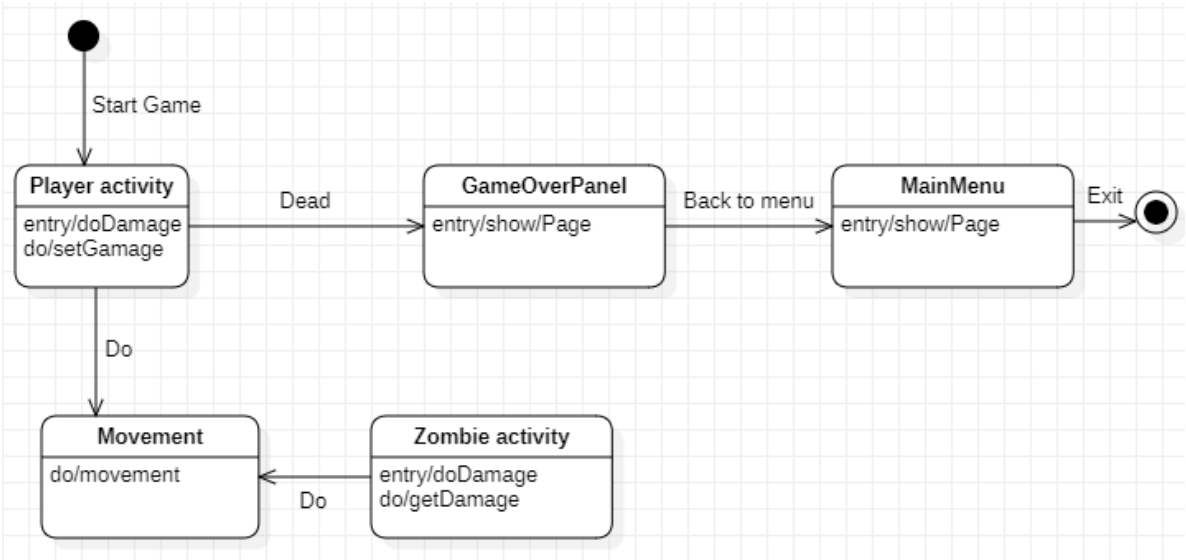


Рисунок 2.5 – Діаграма станів гравця

На діаграмі переходів станів активності гравця показано можливість пересування гравця та можливість нанесення та отримання пошкодження де після закінчення здоров’я заявиться вікно повідомлення про поразку. Ворог має такі ж самі можливості.

Для продемонстрована діаграма станів геймплею (рис. 2.6).

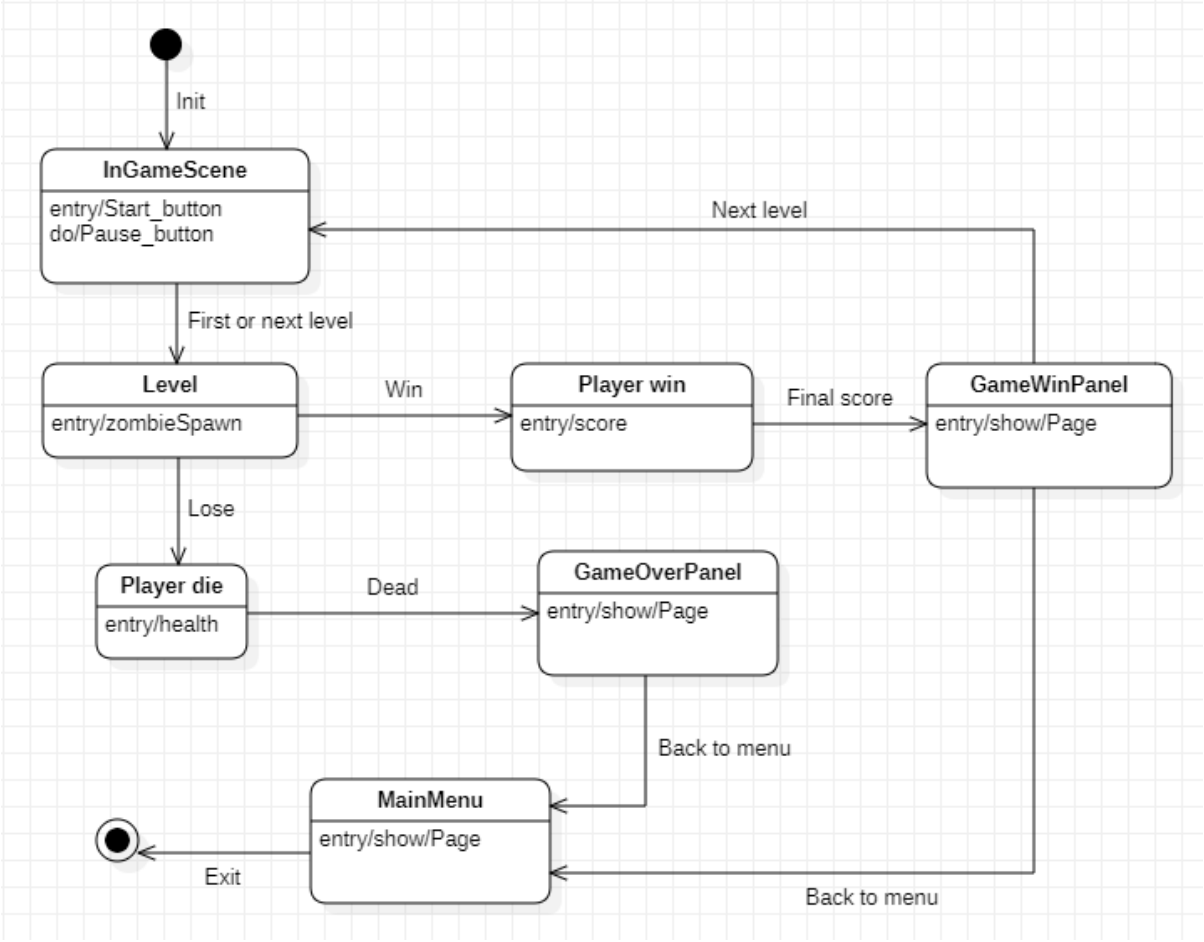


Рисунок 2.6 – Діаграма станів геймплею

На діаграмі переходів станів геймплею показано процес гри. Після старту починається генерація рівня, на сцені з’являються вороги, після закінчення ворогів у гравця відкривається переможне вікно з якого можна перейти до наступного рівня або ж до головного меню. Також показано що після поразки гра закінчується.

2.5 Створення мокапів

Для створення мокапів гри (рис. 2.7-2.11), що розробляється було використано онлайн інструмент Moqups.

Moqups – це онлайн-інструмент для створення макетів, веб-сайтів, дизайну інтерфейсу користувача та інших візуальних матеріалів. Він дозволяє користувачам створювати прототипи веб-сайтів та додатків, проводити тестування та демонструвати свої ідеї колегам та клієнтам [9].

Moqups має багатий набір інструментів, які дозволяють користувачам створювати візуальні матеріали без необхідності знань програмування або графічного дизайну. В інструментарії Moqups є набір форм, кнопок, іконок та інших елементів, які можна використовувати для створення веб-сторінок та інтерфейсів.

Крім того, Moqups дозволяє користувачам працювати в команді, обмінюватися даними та зберігати свої проекти в хмарі. Користувачі можуть запрошувати інших учасників проекту, щоб спільно працювати над ним, а також відстежувати зміни та коментарі.

Moqups має безкоштовний тарифний план з обмеженими функціями, а також платні плани з більшим набором інструментів та можливостей. Загалом, Moqups є потужним інструментом для дизайнерів, розробників та інших професіоналів, які працюють з веб-дизайном та інтерфейсами користувачів.

Першим розробленим мокапом був мокап головного меню гри (рис. 2.7).

Рисунок 2.7 – Мокап головного меню гри

Головне меню гри. В ньому гравець може побачити скількох зомбі він знищив за весь час гри. Також може увімкнути та вимкнути вібрацію або звук в грі. Також на даному екрані розташована кнопка Play яка дозволяє перейти до вибору гри. Даний екран відображається коли гравець тільки заходить в гру, коли завершує грати, або ж коли виходить з меню варіантів гри .

Другим розробленим мокапом був мокап вибору варіантів гри (рис. 2.8).

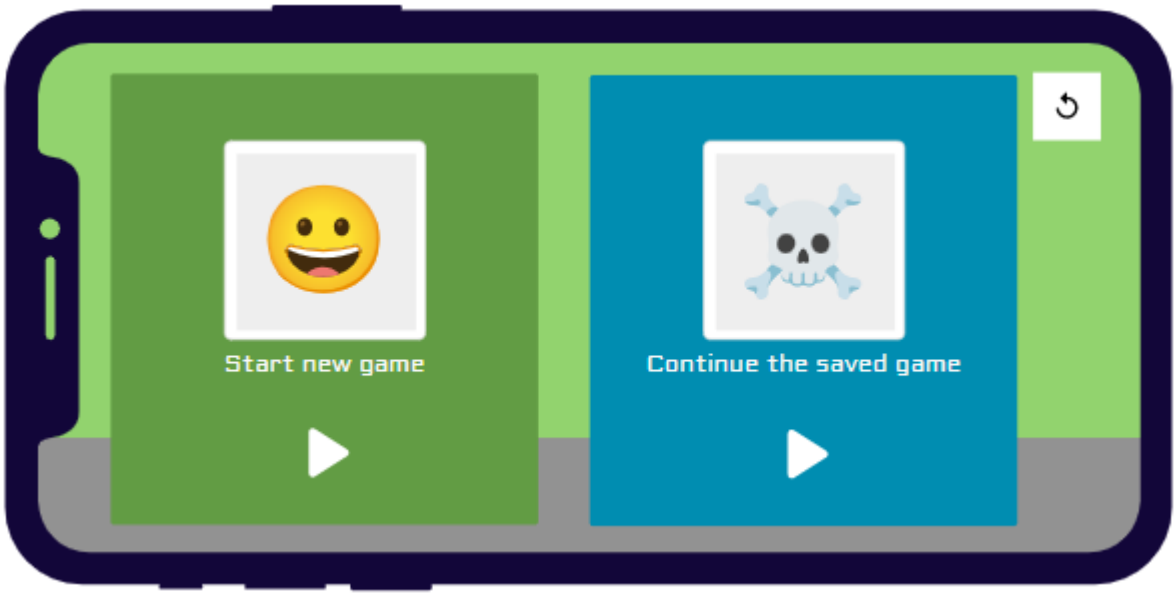


Рисунок 2.8 – Варіанти гри

Варіанти гри. Дане меню дозволяє почати нову гру або повернутись до останньої збереженої локації. Також є можливість повернутися до головного меню гри. Даний екран відображається лише коли гравець натискає на кнопку Play на екрані головного меню гри.

Третім розробленим мокапом був мокап рівня гри (рис. 2.9).

Рисунок 2.9 – Рівень

Рівень. На даний екран можна потрапити тільки натиснувши один з варіантів гри в попередньому меню (рис. 2.8). На даному екрані відображається кількість знищених зомбі за час проходження рівня. Також є джойстік керування переміщенням персонажу, дві кнопки дії та кнопка паузи.

Четвертим розробленим мокапом був мокап меню паузи (рис. 2.10).

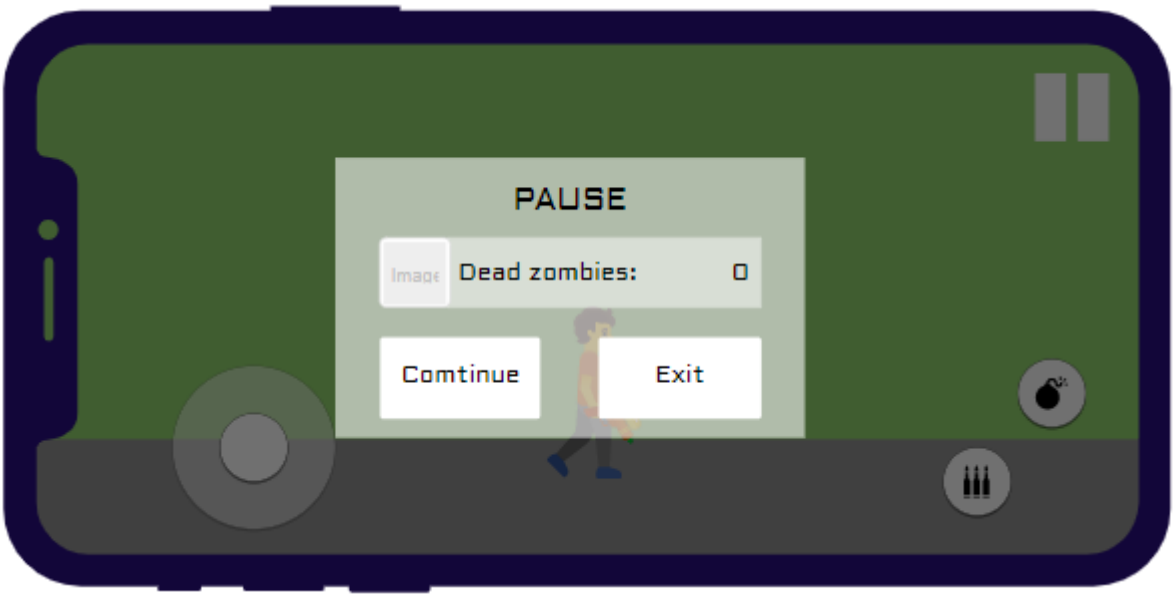


Рисунок 2.10 – Меню паузи

Меню паузи доступне лише під час гри після натискання кнопки паузи. На даному екрані відображаються кількість мертвих зомбі та дві кнопки: Continue за допомогою якої можна продовжити гру, та Exit за допомогою якої можна повернутися до головного меню гри (рис. 2.7).

Першим розробленим мокапом був мокап меню смерті (рис. 2.11).

Рисунок 2.11 – Меню сперті персонажа

Меню сперті персонажа доступне лише під час гри після смерті ігрового персонажа. На даному екрані відображаються кількість мертвих зомбі та кнопка Exit за допомогою якої можна повернутися до головного меню гри (рис. 2.7).

Висновки до розділу 2

Під час написання другого розділу було досліджено створення UML діаграм таких як: діаграми станів, використання та взаємодії. Після чого було створені власні діаграми. Також було описано сценарії використання у різних формах (коротка, поверхнева та повна). За допомогою професійного інструменту Moqups, після його дослідження, було створено мокапи для гри що проєктується.

У підсумку можна сказати, що після проведеної роботи зрозуміло у яку сторону рухатись та з чого починати роботу щодо проєкту.

3 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙ

3.1 Розробка UML-діаграм

У процесі моделювання та проєктування програмного забезпечення часто використовуються діаграми, а одними з найпоширеніших та зручних є UML-діаграми. Вони дозволяють описати бізнес-процеси, системне проєктування та відобразити організаційні структури.

Під час моделювання гри в другому розділі були використані діаграми взаємодії, діяльності та станів, оскільки вони найкраще відображають структуру та вимоги до гри. На етапі проєктування були використані діаграми класів, станів та переходів, компонентів та пакетів, оскільки вони детально відображають поведінку, архітектуру ПЗ та апаратну частину. Для створення цих діаграм було використано застосунок StarUML [10].

3.1.1 Діаграма класів

Діаграми класів є одним з основних інструментів моделювання програмних систем з використанням мови UML [11]. Ці діаграми відображають структуру системи, яка складається з класів, які містять атрибути та методи, що визначають поведінку класу. Класи можуть мати відносини між собою, такі як асоціації, агрегації та композиції, які відображають структурні зв'язки між класами.

Класи відображають сутності, що відповідають реальним об'єктам або абстракціям в програмній системі. У класах можуть бути визначені атрибути, які відображають властивості об'єктів, та методи, які визначають поведінку класу. Відношення між класами відображають зв'язки між об'єктами різних класів, такі як агрегація, композиція та асоціація.

Діаграми класів дуже корисні при проектуванні програмних систем, оскільки вони дають зрозуміти логічну структуру системи та зв'язки між її

складовими частинами. Крім того, вони можуть бути використані для автоматичної генерації програмного коду, що дозволяє значно скоротити час розробки системи та зменшити кількість помилок в коді. Далі зображено діаграму класів (рис. 3.1) застосунку для КРБ.

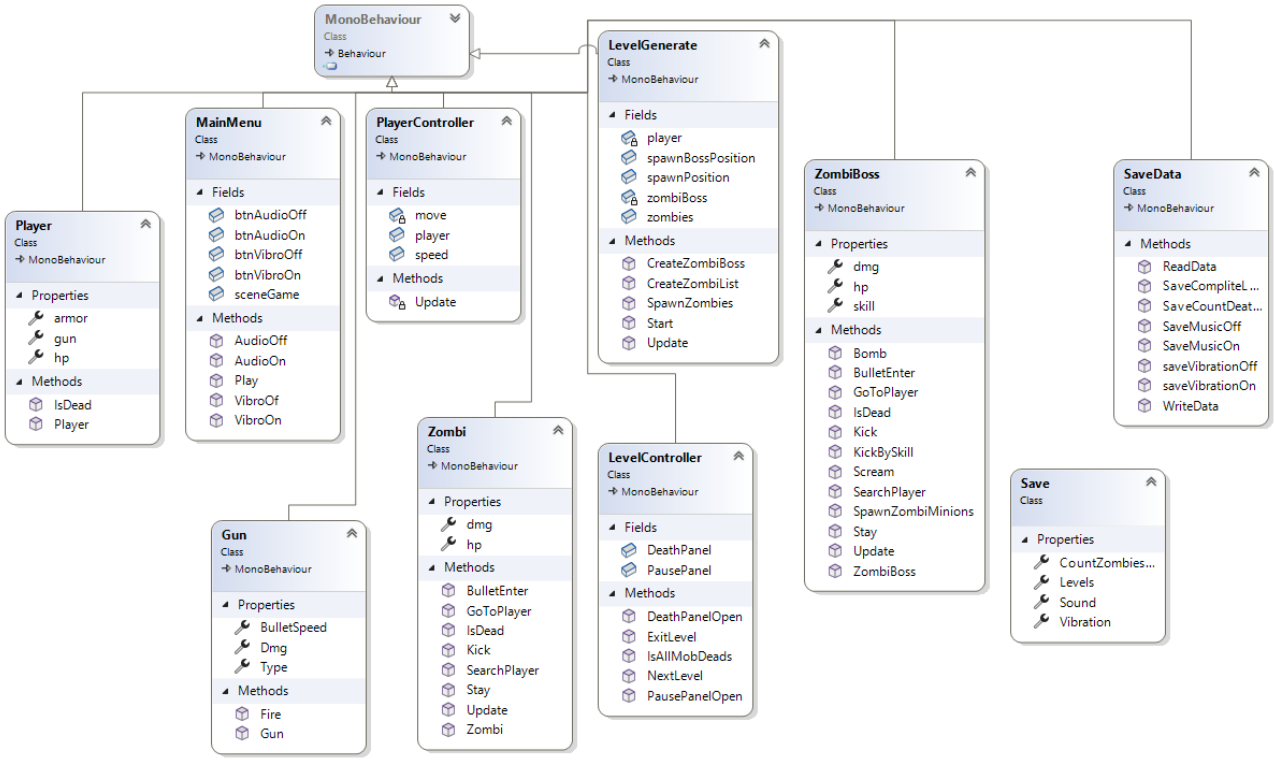


Рисунок 3.1 – Діаграма класів

Клас Player – відповідає за головний клас персонажа гравця.

Клас PlayerController – відповідає за керування персонажем.

Клас MainMenu – відповідає за роботу головного меню гри.

Клас Gun – відповідає за роботу зброї у грі.

Клас Zombie – відповідає за дії зомбі у грі.

Клас ZombieBoss – відповідає за дії босів рівнів.

Клас LevelGenegate – відповідає за генерацію рівня.

Клас LevelController – відповідає за керування геймпею.

Клас Save та SaveData – відповідають за збереження інформації про налаштування та пройдені рівні.

3.1.2 Діаграми компонентів та розгортання

Діаграми компонентів (component diagrams) показують, як програмна система представлена фізично у вигляді окремих елементів, що називаються компонентами (components) [12]. Кожен компонент має своє ім'я, мову програмування, в якій він реалізований, і список класів, які до нього відносяться. Фізично компонент може бути зображений як окремий файл, директорія або навіть окремий програмний продукт.

Демонстрація діаграми компонентів (рис. 3.2).

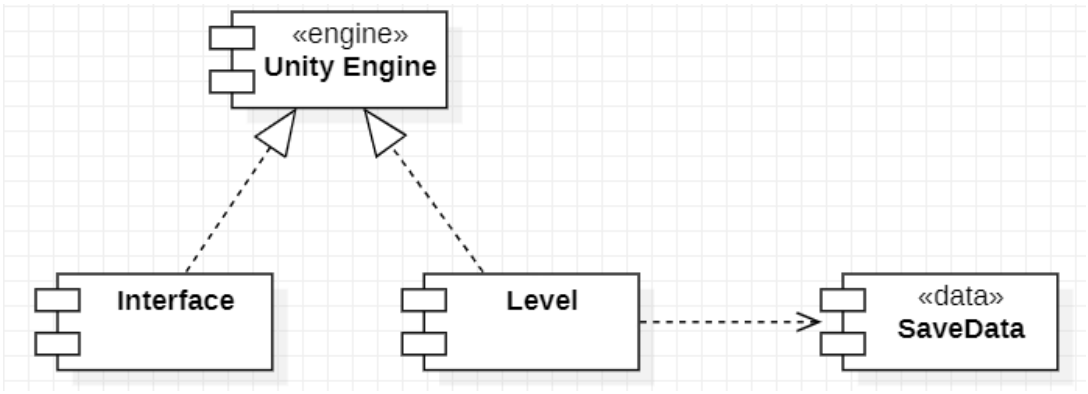


Рисунок 3.2 – Діаграма компонентів

На діаграмах впровадження буде показано екземпляри компонентів та їх асоціації. На них буде показано вузли, які є фізичними ресурсами, окремими комп’ютерами. Крім того, на них показують інтерфейси і об’єкти (екземпляри класів).

Продемонстровано діаграму впровадження (рис. 3.3)

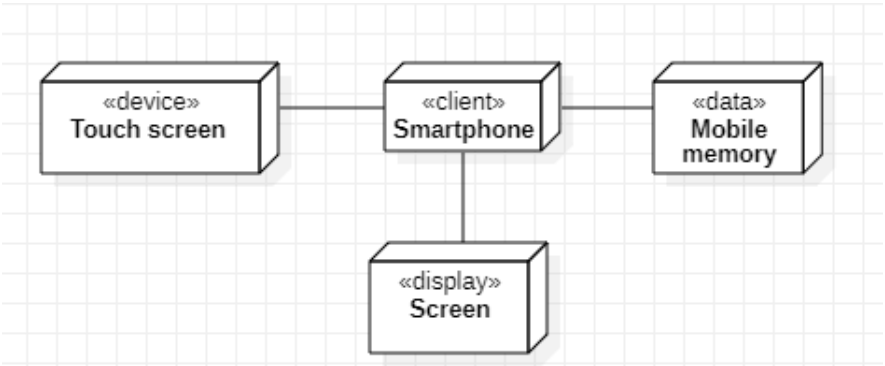


Рисунок 3.3 – Діаграма впровадження

Наступною продемонстрована діаграма розгортання (рис. 3.4).

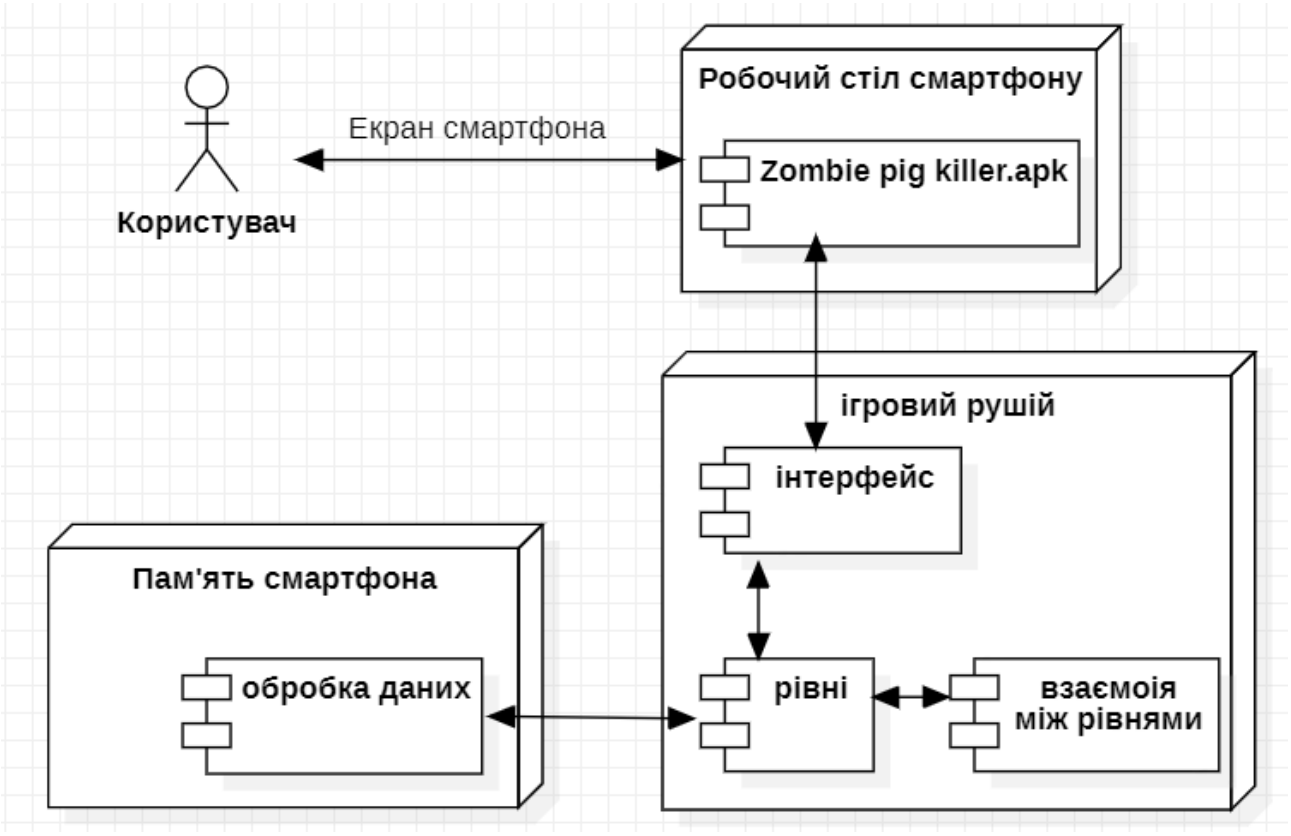


Рисунок 3.4 – Діаграма розгортання

Діаграма розгортання є графічним зображенням інфраструктури, на якій буде фізично розміщено застосунок [13]. Вона відображає топологію системи, розташування компонентів на різних вузлах, зв'язки між компонентами та маршрути передачі даних між вузлами. Діаграма допомагає раціонально організувати компоненти, що безпосередньо впливає на продуктивність системи, а також вирішувати питання, пов'язані з безпекою та іншими допоміжними аспектами.

3.1.3 Діаграма пакетів

Діаграмою пакетів є діаграма, що містить пакети класів і залежності між ними. Строго кажучи, пакети і залежності є елементами діаграми класів, тобто діаграма пакетів – це всього лише форма діаграми класів. Однак на практиці причини побудови таких діаграм різні [14].

Окрім стандартних відношень залежності в UML існує два спеціальних види залежності, які визначено між пакетами:

– імпорт пакету;

– злиття (суміщення, об'єднання) пакету.

Діаграми пакетів можуть використовувати пакети, які містять прецеденти для демонстрації функціональності програмного забезпечення системи. Діаграми можуть використовувати пакети, які показують різноманітні шари програмного комплексу для ілюстрації його архітектури, що складається з різних шарів. Залежності між цими пакетами можуть бути наділені позначками / стереотипами, щоби вказати механізм зв'язку між шарами.

Розглянемо діаграму пакетів до гри що проектується (рис. 3.5).

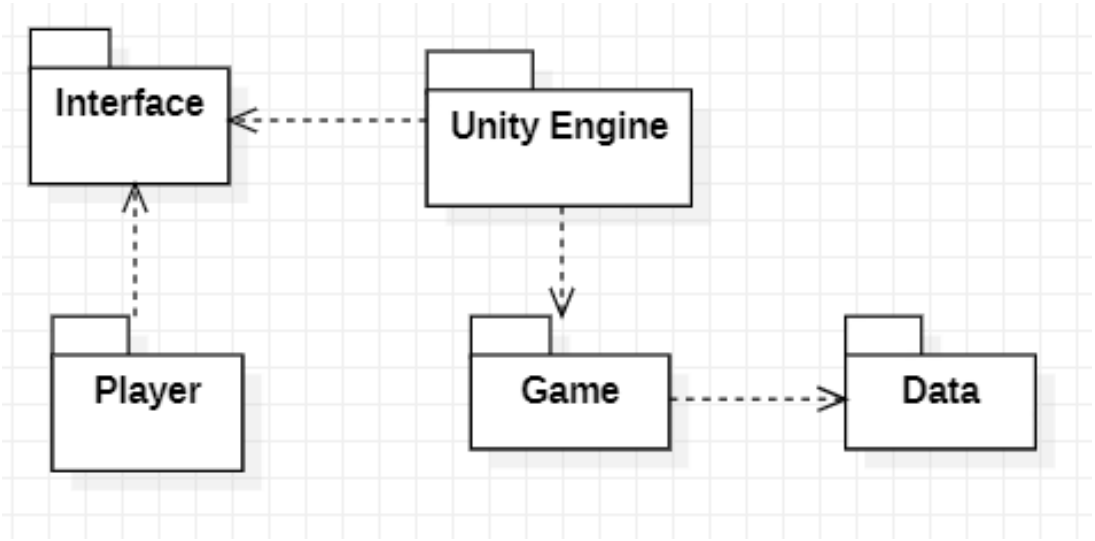


Рисунок 3.5 – Діаграма пакетів

Пакет Player, що відповідає за дії гравця, та Uniti Engine, що виступає рушієм всього проекту, не залежать від інших пакетів, натомість вони впливають на пакети Interface, який реагує на взаємодію гравця з інтерфейсом та тим що відбувається у грі, та Game. В свою чергу пакет Game, що відповідає за геймплей, впливає на пакет Data, який містить у собі логіку збереження необхідної для гри інформації.

3.2 Огляд технологій

Під час розробки гри було використано такий стек технологій(рис. 3.6):

– мова програмування: C#;

– ігровий рушій: Unity;

– графічний редактор: Photoshop.

Рисунок 3.6 – Стек технологій з логотипами відповідно

Вибір стеку технологій має велике значення, оскільки він впливає на продуктивність, вимоги до апаратних компонентів та надійність програмного забезпечення. Для кожної задачі важливо підібрати відповідний інструмент для її виконання.

3.2.1 Мова програмування

C# (C-Sharp) є мовою програмування, розробленою компанією Microsoft, яка поєднує в собі елементи об'єктно-орієнтованого та компонентного програмування. Вона є однією з основних мов для розробки програмного забезпечення на платформі .NET [15].

Основні особливості C# включають:

1. Об'єктно-орієнтований підхід: C# підтримує принципи класів, об'єктів, спадкування, поліморфізму та інших концепцій ООП, що сприяє створенню модульного та розширюваного коду.

2. Багатопоточність: Мова має вбудовану підтримку багатопоточності, що дозволяє розробляти паралельні програми для ефективної використання ресурсів системи.

3. Управління пам'яттю: C# використовує автоматичне управління пам'яттю (Garbage Collection), що спрощує роботу з пам'яттю та дозволяє уникнути багатьох типових помилок, пов'язаних з витіками пам'яті.

4. Широкий функціонал .NET Framework: C# базується на .NET Framework, що надає багато готових класів та бібліотек для швидкого розроблення різноманітних програмних рішень.

5. Платформонезалежність: Код, написаний на C#, може бути виконаний на різних платформах, таких як Windows, macOS та Linux, завдяки платформі .NET Core.

C# знаходить широке застосування в розробці різноманітних додатків, включаючи веб-додатки, мобільні додатки, настільні програми, ігри та інші програмні рішення. Вона має потужні інструменти розробки, велику спільноту розробників та надійну підтримку з боку Microsoft, що робить її привабливим вибором для розробників. З C# можна створювати високоякісні, масштабовані та ефективні програмні рішення. Вона має розширену функціональність для роботи з базами даних, мережевими операціями, графікою, криптографією та багатьма іншими аспектами розробки програмного забезпечення.

C# також підтримує розробку веб-додатків з використанням ASP.NET, що дозволяє створювати потужні та сучасні веб-сайти та API. Засоби розробки, такі як Visual Studio, надають зручне середовище для програмування на C# і підтримують широкий набір інструментів для тестування, налагодження та розгортання додатків.

Крім того, C# є однією з основних мов для розробки додатків під платформу Unity, що дозволяє створювати ігри та інтерактивні додатки для різних платформ, включаючи комп'ютери, мобільні пристрої та віртуальну реальність.

Загалом, C# є потужною та гнучкою мовою програмування, яка надає розробникам широкі можливості для створення різноманітних програмних рішень, незалежно від їх масштабу та платформи.

3.2.2 Ігровий рушій

Unity – це потужний інтегрований середовище розробки (IDE) та движок для створення ігор, віртуальної реальності (VR), доповненої реальності (AR) та інших інтерактивних додатків. Він був розроблений компанією Unity Technologies і має широкий спектр функцій та можливостей для розробки ігор у 2D та 3D середовищах [16].

Основні особливості Unity включають:

1. Кросплатформеність: Unity дозволяє розробляти ігри та додатки для різних платформ, таких як комп'ютери, мобільні пристрої (Android, iOS), консолі (PlayStation, Xbox), віртуальна реальність (Oculus Rift, HTC Vive) та інші. Розроблені проекти можуть бути експортовані на різні платформи з використанням єдиного коду.

2. Візуальний редактор: Unity має інтуїтивний візуальний редактор, який дозволяє легко створювати, редагувати та розміщувати об'єкти у сцені, налаштовувати їх властивості, створювати анімацію та налаштовувати фізику.

3. Скриптовий рушій: Unity використовує скриптовий рушій, який базується на мові програмування C#. Розробники можуть створювати скрипти, які контролюють поведінку об'єктів у грі, реалізовувати логіку гри, створювати інтерактивність та реалізовувати різноманітні функціональні можливості.

4. Розширення та активи: Unity підтримує використання активів, які є готовими компонентами, що допомагають в розробці ігор. У Unity Store доступні безліч безкоштовних та платних активів, таких як графічні ефекти, моделі персонажів.

5. Фізика та анімація: Unity надає потужні інструменти для реалістичного моделювання фізики об'єктів у грі. Ви можете налаштовувати взаємодію об'єктів, симулювати гравітацію, колізії та інші фізичні ефекти. Крім того, Unity підтримує створення складних анімацій для персонажів та об'єктів у грі.

6. Віртуальна реальність та доповнена реальність: Unity надає вбудовану підтримку для розробки VR та AR додатків. Це дозволяє створювати іммерсивні віртуальні світи та інтерактивні AR додатки для різних пристроїв.

Unity є одним з найпопулярніших інструментів для розробки ігор та інтерактивних додатків завдяки своїй потужності, гнучкості та розширюваності. Він дозволяє розробникам мати повний контроль над процесом розробки та створювати вражаючі творіння у світі геймдеву та інтерактивних додатків.

3.2.3 Графічний редактор

Photoshop – це відомий графічний редактор, розроблений компанією Adobe. Він є одним з найпопулярніших інструментів для редагування та обробки растрових зображень [17].

Основні особливості Photoshop включають:

1. Редагування зображень: Photoshop надає широкий набір інструментів для редагування зображень, таких як корекція кольору, яскравості та контрастності, видалення недоліків, ретушування шкіри, робота з шарами та масками, вирізання об'єктів, зміна розміру та кадрування тощо.

2. Творчість та дизайн: Photoshop є потужним інструментом для створення графічних дизайнів, логотипів, постерів, ілюстрацій та інших творчих проектів. Він має різноманітні кисті, фільтри, текстові ефекти та інші можливості для створення унікальних графічних елементів.

3. Робота з шарами: Photoshop дозволяє працювати зі шарами, що дозволяє розділити зображення на окремі елементи для зручного редагування. Це дає можливість незалежно контролювати різні частини зображення, змінювати їх порядок, налаштовувати прозорість та застосовувати різні ефекти.

4. Робота з форматами файлів: Photoshop підтримує широкий спектр форматів файлів, включаючи JPEG, PNG, GIF, TIFF, PSD (власний формат Photoshop) та багато інших. Це дає можливість зберігати та експортувати зображення у різних форматах залежно від потреб проекту.

5. Автоматизація та пакетна обробка: Photoshop має функціонал для автоматизації деяких завдань та пакетної обробки зображень. Це дозволяє зменшити час, потрібний для повторюваних операцій, шляхом створення дієвих дій та скриптів.

Photoshop є незамінним інструментом для професіоналів графічного дизайну, фотографії та інших сфер, де потрібна робота з растровими зображеннями. Він надає безліч можливостей для творчості та редагування, дозволяючи створювати вражаючі графічні роботи.

Висновки до розділу 3

Третій розділ присвячений проєктуванню гри і огляду використаних технологій, таких як мова програмування (C#), ігровий рушій (Unity Engine) і графічний редактор (Photoshop). Під час проєктування застосунку було набуто нових навичок у розробці UML-діаграм, зокрема класів, станів та переходів, а також пакетів. Кожна створена діаграма була детально і лаконічно описана, що полегшує подальшу роботу з ними.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ГРИ НА ОСНОВІ РУШІЯ UNITY

4.1 Створення спрайтів для гри

Одною з головних складових гри є графіка. Всі елементи графіки для гри були створенні за допомогою графічного редактора Photoshop. Основними інструментами для малювання було обрано: Олівець, Прямокутна область, Заливка, Швидке виділення.

Так було створено по-кадрову анімацію ходьби для головного героя гри (рис. 4.1)

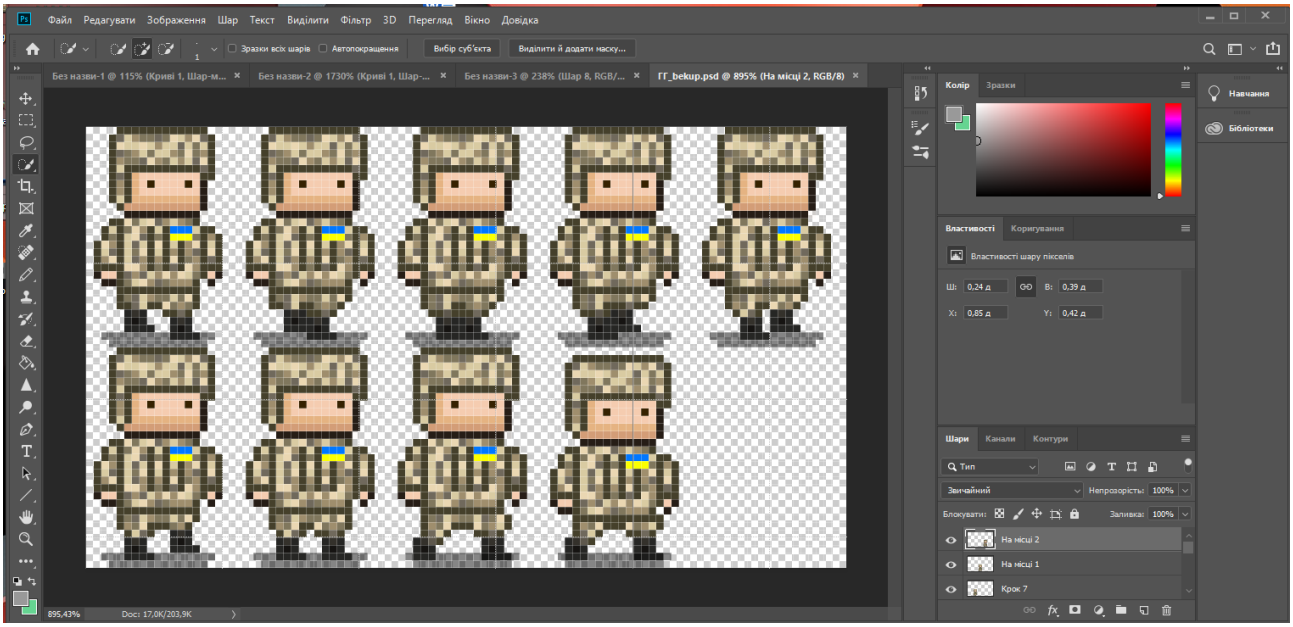


Рисунок 4.1 – Спрайти по-кадрової анімації головного героя гри

Подібна анімація була створена і для зомбі, що є противниками в грі. Також для зомбі була створена анімація атаки (рис. 4.2).

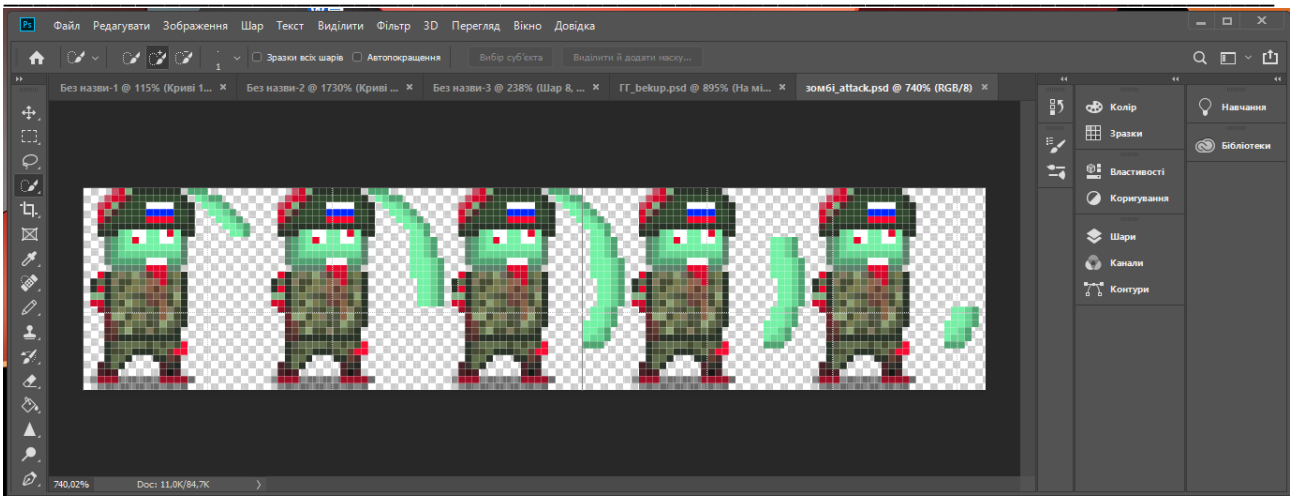


Рисунок 4.2 – Спрайти атаки зомбі

Також було створено спрайти UI елементів, зброї, будинків, та інших елементів локацій.

4.2 Створення головного меню гри

Для головного меню в грі було створено Canvas в якому знаходяться кнопки взаємодії з налаштуваннями звуку та вібрації, кнопка початку гри а також особистий рахунок знищених зомбі (рис. 4.3).

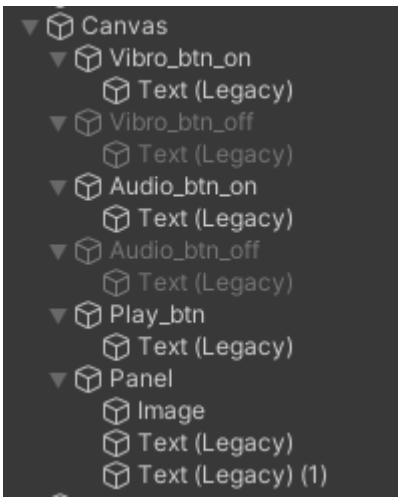


Рисунок 4.3 – Canvas з UI елементами всередині

Далі для того щоб замінити стандартні спрайти UI елементів на раніше намальовані треба було перетворити малюнок на мультиспрайтовий об’єкт для

цього в інспекторі треба було Sprite Mode обрати Multiple, Filter Mode – Point (no filter) та натиснути на кнопку Sprite Editor(рис. 4.4)

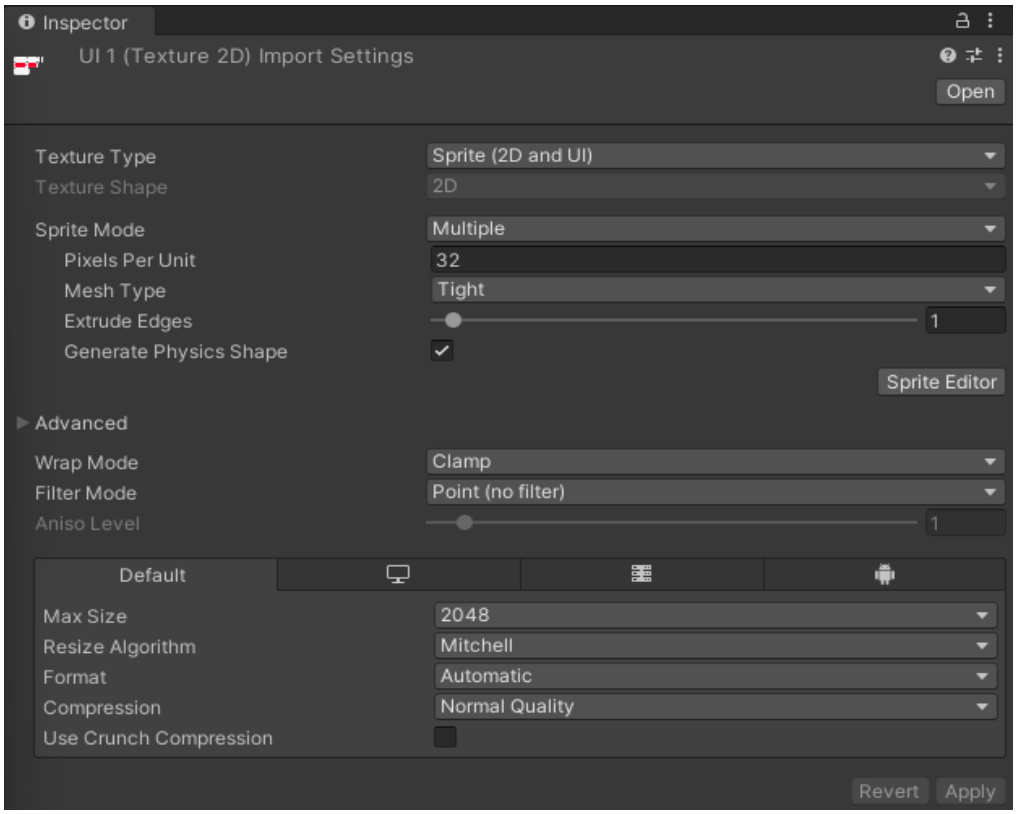


Рисунок 4.4 – Вікно інспектора налаштувань спрайтів

Далі в вікні Sprite Editor треба було виділити всі елементи в окремі прямокутники та дати їм ім’я (рис. 4.5).

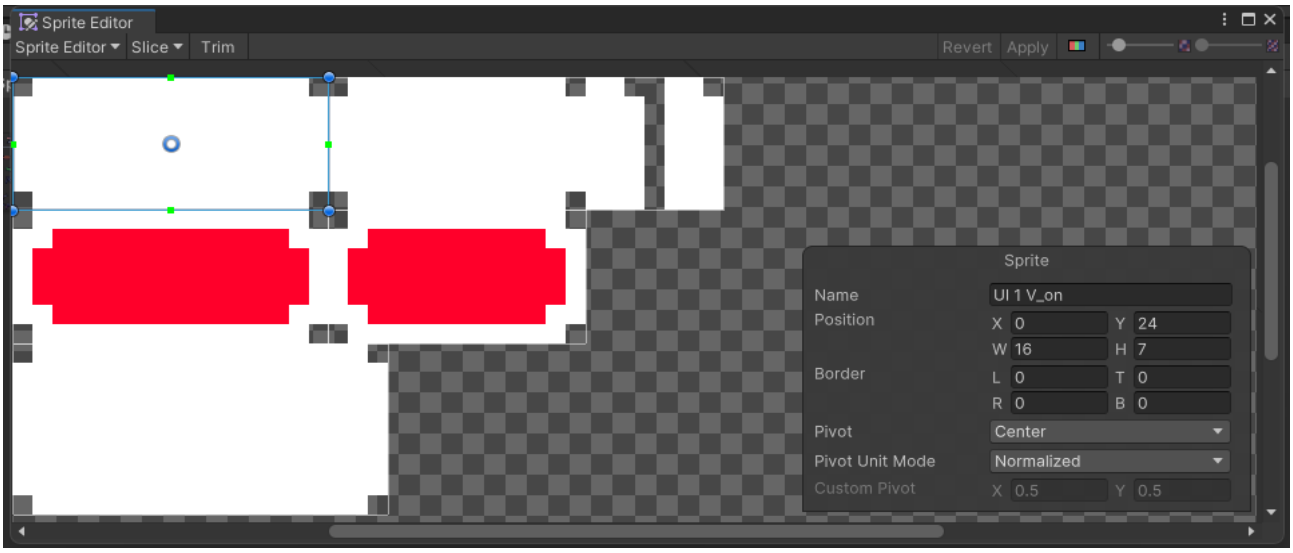


Рисунок 4.5 – Вікно Sprite Editor

Далі в вікні інспектора було змінено налаштування кнопок та інших UI елементів, змінено написи та шрифти після чого мені прийняло ось такий вигляд (рис. 4.6).

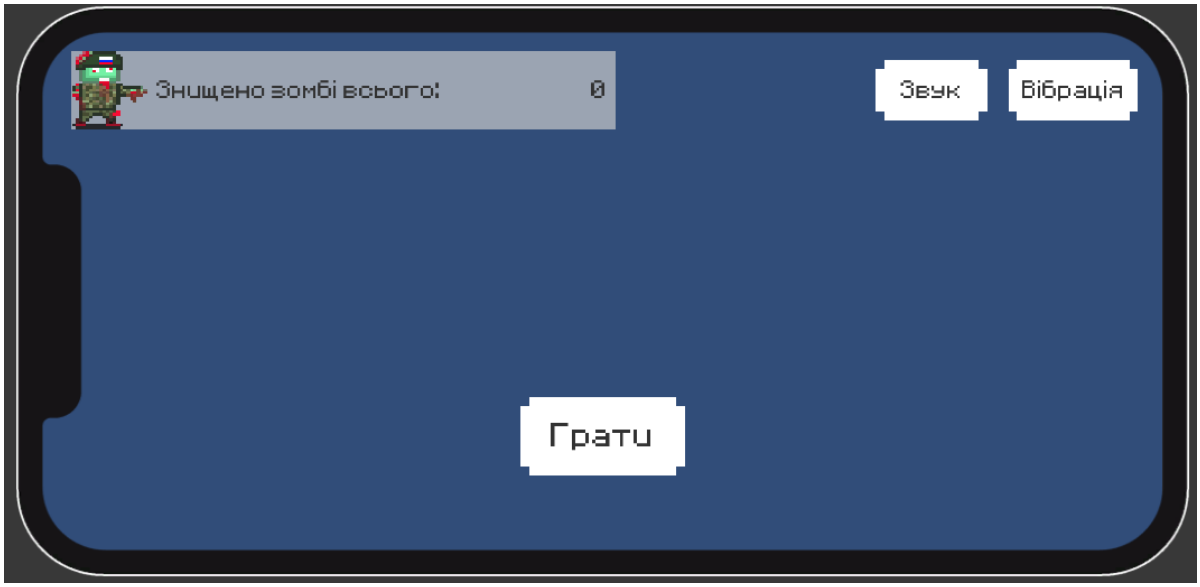


Рисунок 4.6 – Головне меню

Далі з інших спрайтів було створено фон для головного меню й його фінальний вигляд було змінено (рис. 4.7).

Рисунок 4.7 – Головне меню

Таким чином було створено головне меню для гри.

4.3 Створення героя гри

Для початку було створено порожній об’єкт Player та в нього додано початковий спрайт анімації а також зброю (рис. 4.8).

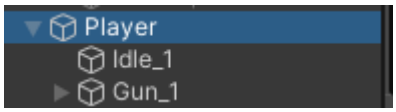


Рисунок 4.8 – Об’єкт Player з усіма дочірніми

Далі для об’єкта Player було додано компоненти Rigidbody 2D та Capsule Collider 2D (рис. 4.9) для можливості керувати та взаємодіяти з іншими об’єктами в грі.

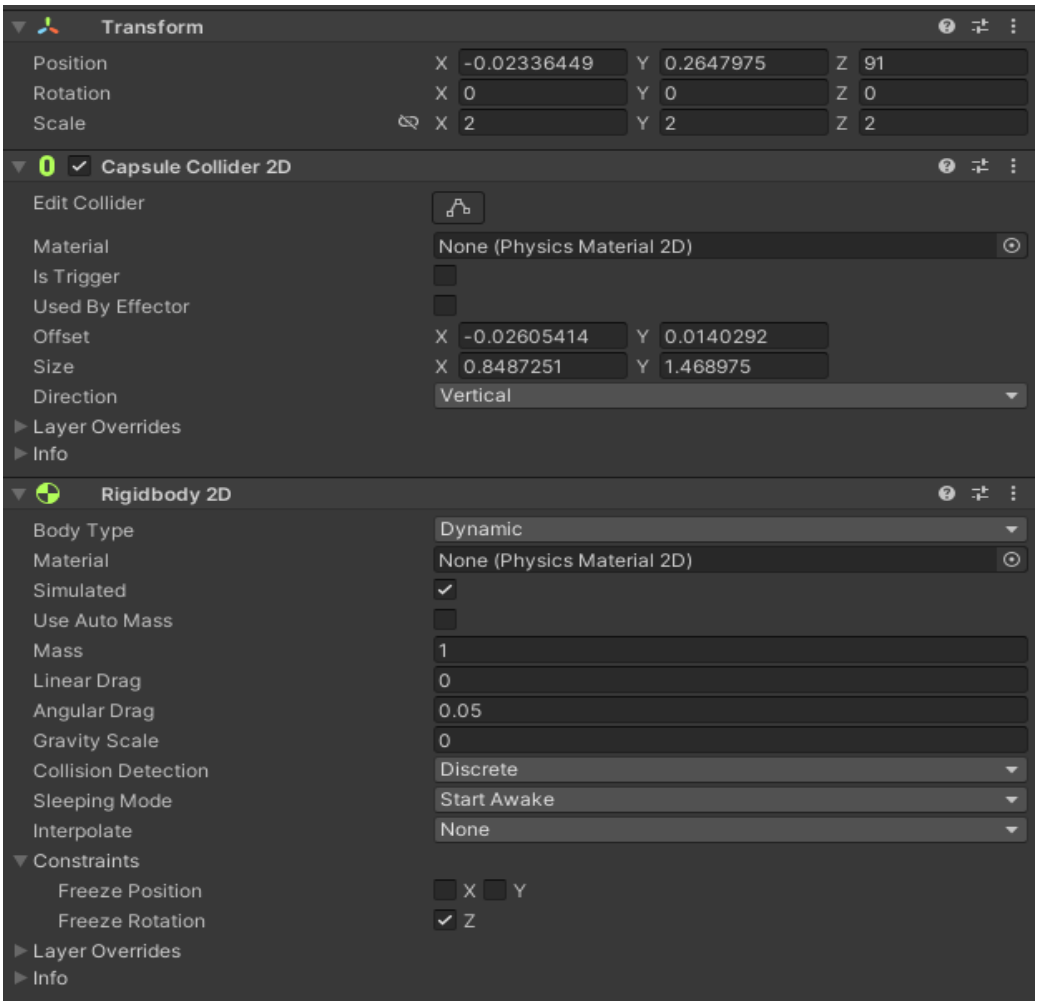


Рисунок 4.9 – Інспектор об’єкту Player

Після чого за допомогою інструменту Animation та Animator, а також раніше намальованих спрайтів по-кадрової анімаці було створено анімації для персонажа (рис. 4.10): спокій та біг.

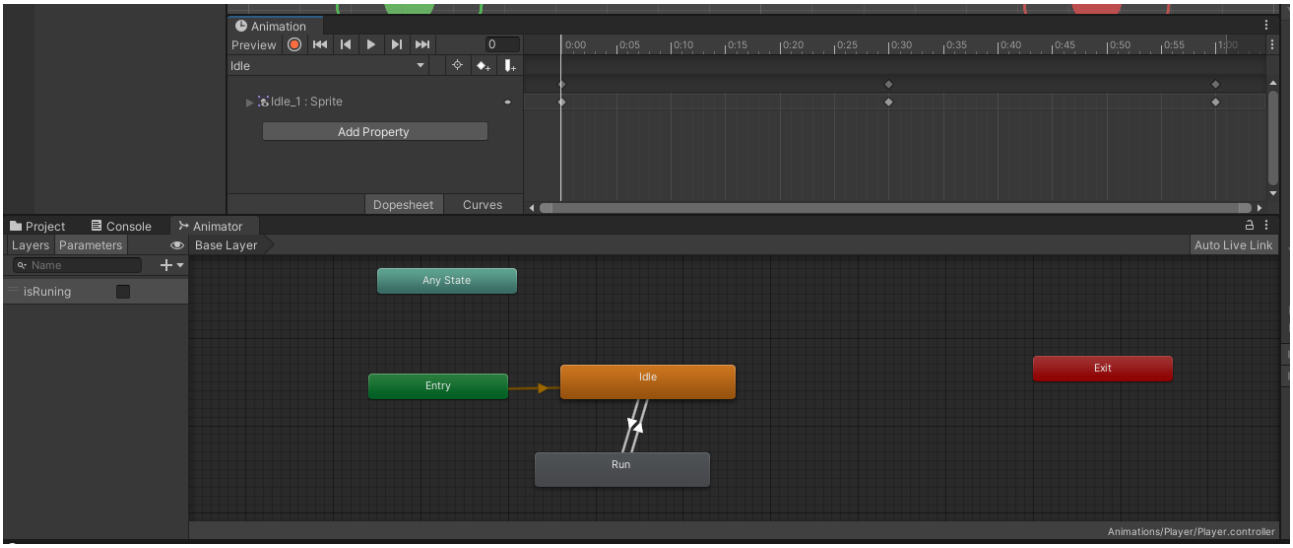


Рисунок 4.10 – Створення анімацій

Таким чином було створено персонажа для гри (рис. 4.11).

Рисунок 4.11 – Персонаж для гри

Персонаж для гри вийшов доволі цікавим.

4.4 Керування персонажем

Для керування персонажем було створено C#-скрипт PlayerController який відповідає за переміщення персонажа. А також було додано джойстик керування.

Клас PlayerController містить в собі метод що спрацьовує від початку появи об’єкту на сцені, в нашому випадку від появи персонажа, та знаходить компоненти взаємодії з персонажем як твердим тілом та анімації. Також містить у собі метод обробки кадрів Update, слідкує за джойстиком та запускає анімації, та метод фіксованої обробки кадрів FixUpdate, відносно джойстика переміщує персонажа по карті (рис. 4.12).

```
Unity Message | 0 references
void Update()
{
    move.x = joystick.Horizontal;
    move.y = joystick.Vertical;

    if(move.x == 0)
    {
        animator.SetBool("isRuning", false);
    }
    else
    {
        animator.SetBool("isRuning", true);
    }

    if(!facingRight && move.x > 0 )
    {
        Flip();
    }
    else if (facingRight && move.x < 0)
    {
        Flip();
    }

    if(health<= 0)
    {
        Death();
    }
}
Unity Message | 0 references
private void FixedUpdate()
{
    player.MovePosition(player.position + move*speed*Time.fixedDeltaTime);
}
```

Рисунок 4.12 – Методи керування персонажем та анімацією.

Щоб даний скрипт працював його необхідно повісити на наш об’єкт Player (рис. 4.13) та заповнити необхідні поля.

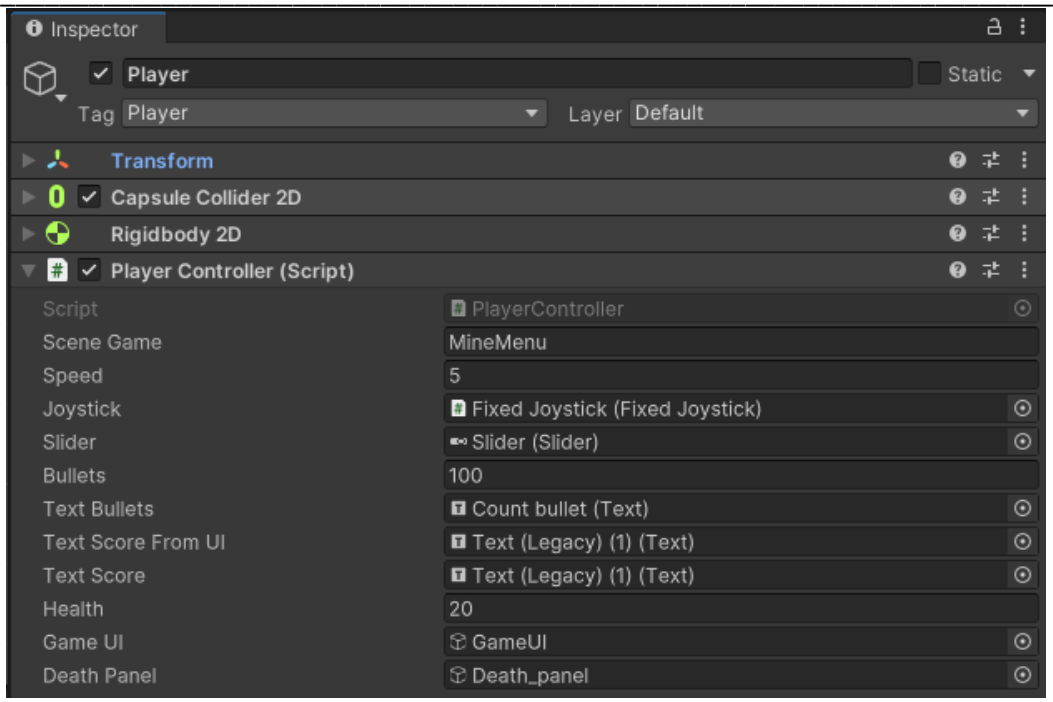


Рисунок 4.13 – Доданий скрипт до об’єкту Player

Для коректної роботи скрипта треба було заповнити необхідні поля які відображаються в меню інстпектора об’єкта Player.

4.5 Створення зброї

Раніше до об’єкту Player було додано об’єкт Gun, тепер для нього також створено кулі та скрипти.

Для об’єкту Gun створено відповідний клас що за допомогою методу Update слідкує за рухами джойстика та вистрілює якщо гравець його торкається, кожен вистріл керується методом Shoot (рис. 4.14).

```
Unity Message | 0 references
private void Update()
{
    rotZ = Mathf.Atan2(joystick.Vertical, joystick.Horizontal) * Mathf.Rad2Deg;

    transform.rotation = Quaternion.Euler(0f, 0f, rotZ);

    if (timeBtwShorts <= 0)
    {
        if (joystick.Horizontal != 0 || joystick.Vertical != 0)
        {
            if(player.bullets > 0)
            {
                Shoot();
            }
        }
    }
    else
        timeBtwShorts -= Time.deltaTime;
}
1 reference
public void Shoot()
{
    player.ChangeBullet(-1);
    Instantiate(bullet, ShotPoint.position, transform.rotation);
    timeBtwShorts = startBtwShorts;
}
```

Рисунок 4.14 – Методи прицілювання та стрільби.

Для куль було створено клас Bullet а також порожній об’єкт ShootPoint, який є точкою вильоту куль (рис. 4.15).

Рисунок 4.15 – Об’єкт Gun та ShootPoint

Для об’єкта Gun теж необхідно заповнити поля скрипта для його правильної роботи.

4.6 Створення ворогів

Для ворога було створено порожній префаб Zombie (рис. 4.16).

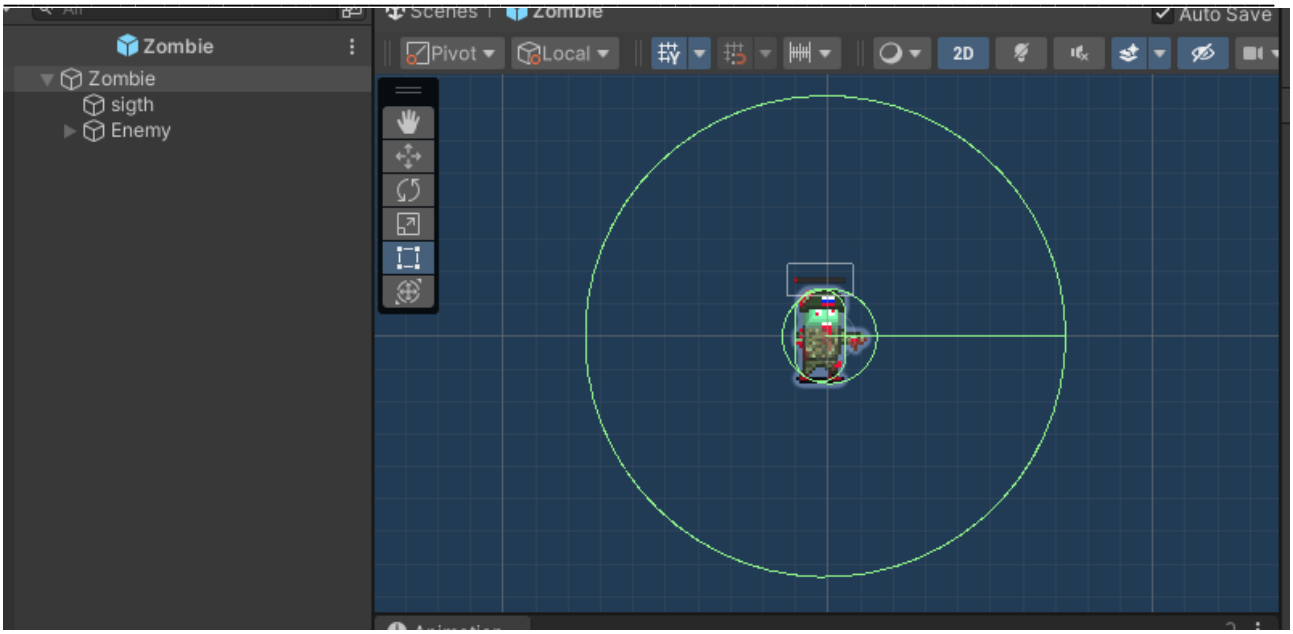


Рисунок 4.16 – Префаб об’єкту Zombie

Для його створення було створено порожній об’єкт Zombie та в нього додано 2 об’єкти sight, для керування полем зору зомбі, та Enemy, для керування переміщенням, атаками.

Для об’єкту sight було додано Circle Collider 2D який є тригером та додано скрипт SightEnimy який слідкує за цим колайдером та коли в ньому з’являється об’єкт з тегом Player то він сигналізує зомбі що час рухатись до об’єкту Player (рис. 4.17).

```
public class SightEnemy : MonoBehaviour
{
    private bool isInSight = false;
    public Transform pos;
    Unity Message | 0 references
    public void Update()
    {
        transform.position = pos.position;
    }
    Unity Message | 0 references
    public void OnTriggerStay2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            isInSight = true;
        }
    }
    1 reference
    public bool GetIsInSight() { return isInSight; }
}
```

Рисунок 4.17 – Код класу SightEnemy

Для об’єкту Enemy було створено анімації аналогічно тим які були на об’єкті Player у підпункті 4.3 цього розділу. Також було додано компоненти Rigidbody 2d та Capsule Collider 2D для переміщення та Circle Collider 2D для атаки. Всі ці компоненти можна побачити у інспекторі об’єкту Enemy (рис. 4.18).

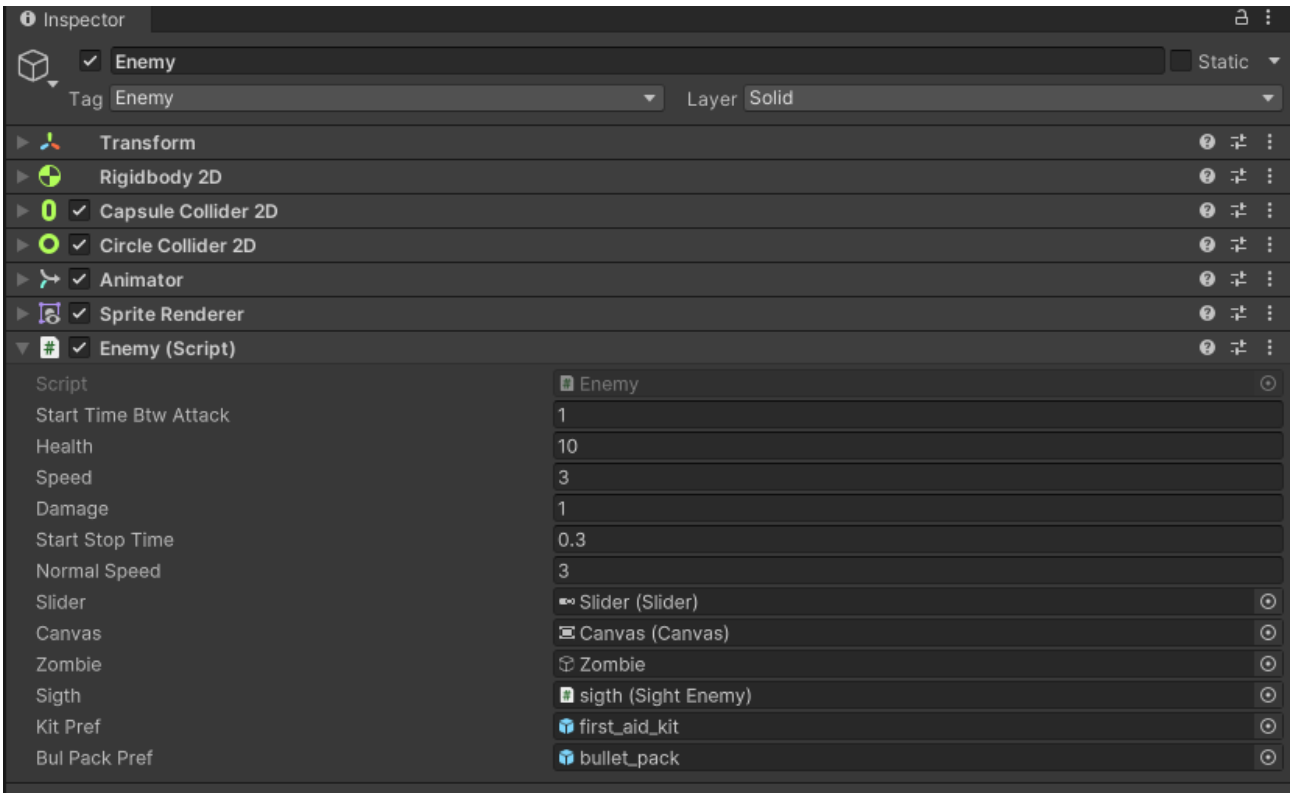


Рисунок 4.18 – Інспектор об’єкту Enemy

Також було написано скрипт Enemy який і керує переміщеннями ворога. Поля цього скрипта видно на рисунку вище. Вони також були заповнені для коректної роботи скрипта.

4.7 Створення додаткових предметів

Головними додатковими предметами у грі є аптечка та боєприпаси. Для них було створено префаби (рис.4.19-4.20).

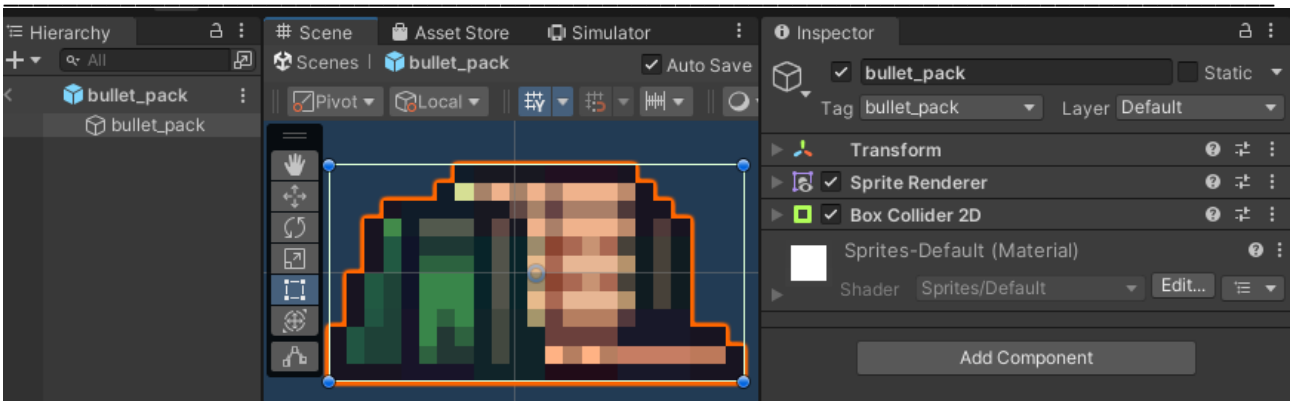


Рисунок 4.19 – Префаб боєприпасів

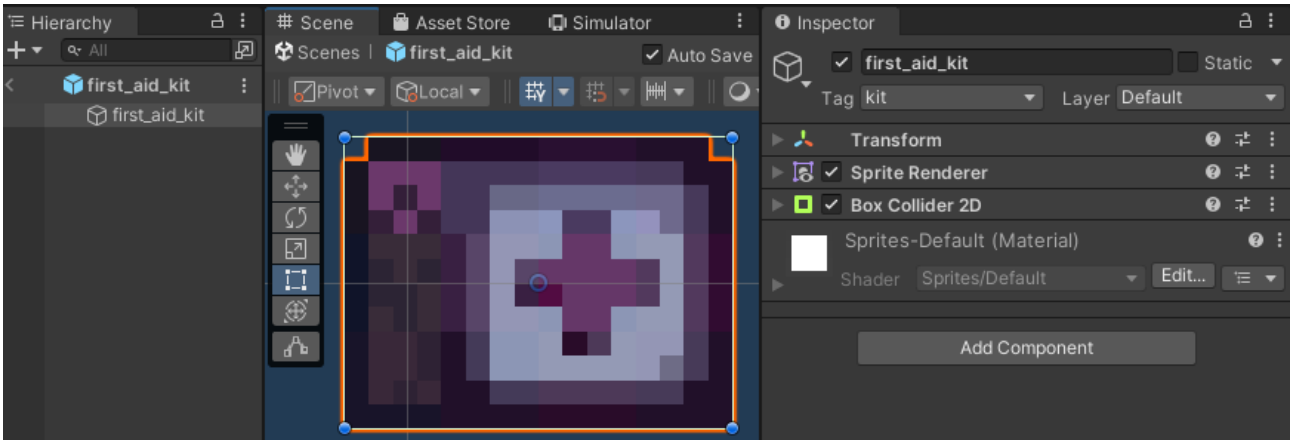


Рисунок 4.20 – Префаб аптечки

Так як ці предмети мають з’являтись після смерті ворога то у скрипті Enemy за допомогою методу SpawnHelp() в якому з певною імовірністю можуть з’явитися як патрони так і аптечка (рис. 4.21).

```
public void SpawnHelp()
{
    Random rand = new Random();
    if(rand.Next()%2 <= 2)
    {
        Instantiate(bulPackPref, transform.position, Quaternion.identity);
    }
    else if(rand.Next() % 2 >= 9)
    {
        Instantiate(kitPref, transform.position, Quaternion.identity);
    }
}
```

Рисунок 4.21 – Метод спавну додаткових предметів

Також у скрипті PlayerController було описано можливість підйому цих додаткових предметів за допомогою методу OnTriggerEnter2D який слідкує чи потрапляє у головний колайдер об’єкту Player об’єкти з тегами «kit» або «bullet_pack» (рис. 4.22).

```
Unity Message | 0 references
public void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("kit"))
    {
        ChangeHealth(5);
        CheckHealth();
        Destroy(collision.gameObject);
    }
    else if (collision.CompareTag("bullet_pack"))
    {
        ChangeBullet(10);
        Destroy(collision.gameObject);
    }
}
```

Рисунок 4.22 – Метод підбору додаткових предметів

Після того як гравець піднімає аптечку то до його здоров’я додається 5 очків, а якщо боєприпаси, то до кількості боєприпасів додається 10.

4.8 Генерація мапи гри

Для генерації мапи гри було створено 5 префабів локацій з точками спавну локацій та ворогів (рис. 4.22).

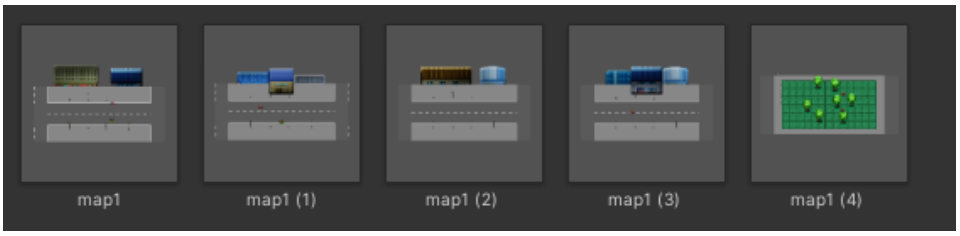


Рисунок 4.22 – Префаби локацій

Також було створено 2 скрипти для генерації самої мапи а також спавну ворогів.

Скрипт MapSpawn (рис. 4.23) генерує один з п’яти префабів мап на точках спавну мап в самих префабів, також є перевірка чи не згенерована на тій точці мапа, якщо згенерована то попередня точка зникає.

```
public class MapSpawn : MonoBehaviour
{
    private Maps mapVar;
    private int rand;
    public bool spawned = false;
    private float waitTime = 3f;

    Unity Message | 0 references
    private void Start()
    {
        mapVar = GameObject.FindGameObjectWithTag("Maps").GetComponent<Maps>();
        Destroy(gameObject, waitTime);
        Invoke("Spawn", 0.5f);
    }
    0 references
    public void Spawn()
    {
        if(!spawned)
        {
            if(GameObject.FindGameObjectsWithTag("Map").Length < 12)
            {

                rand = Random.Range(0, mapVar.maps.Length);
                Instantiate(mapVar.maps[rand], transform.position, mapVar.maps[rand].transform.rotation);
                spawned = true;
            }
        }
    }
    Unity Message | 0 references
    private void OnTriggerStay2D(Collider2D collision)
    {
        if(collision.CompareTag("MapPoint") && collision.GetComponent<MapSpawn>().spawned) {
            Destroy(gameObject);
        }
    }
}
```

Рисунок 4.23 – Клас MapSpawn

Скрипт ZombieSpawner (рис. 4.24) знаходить всі об’єкти з тегом «ZombiePoint», що є точками спавну для ворогів, та створює зомбі на місці цих точок.

```
public class ZombieSpawner : MonoBehaviour
{
    public GameObject zombie;
    private GameObject[] zombieSpawnPosition;
    public Text textSpawn;

    Unity Message | 0 references
    void Start()
    {
        textSpawn.text = 0.ToString();
        Invoke("Spawn", 1f);
    }
    0 references
    private void Spawn()
    {
        zombieSpawnPosition = GameObject.FindGameObjectsWithTag("ZombiePoint");
        for(int i = 0; i<zombieSpawnPosition.Length; i++)
        {
            Instantiate(zombie, zombieSpawnPosition[i].transform.position, zombieSpawnPosition[i].transform.rotation);
        }
    }
}
```

Рисунок 4.24 – Клас ZombieSpawner

Також скрипт має невеличку затримку для того щоб мапа з усіма точками спавну встигла згенеруватись.

4.9 Тестування гри

Функціональне тестування – це процес перевірки відповідності функціональних вимог реалізованим функціям. Тести було проведено на пристрої Xiaomi 11 Lite з операційною системою Android.

Тест 1. Зміна налаштувань звуку та вібрації у головному меню (рис. 4.25).

Вхідні дані: У головному меню натиснути на кнопки Звук та Вібрація.

Очікуваний результат: Звук та вібрація мають зникнути а самі кнопки змінити свій колір.

Отриманий результат: Результат співпадає з очікуваннями (рис. 4.26).

Рисунок 4.25 – Головне меню

Рисунок 4.2 – Зміна налаштувань

Тест 2. Початок гри

Вхідні дані: користувач натискає на кнопку грати.

Очікуваний результат: завантажиться рівень гри

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

Рисунок 4.27 – Рівень гри

Тест 3. Тестування пострілу гравця.

Вхідні дані: гравець тисне на джойстик червоного кольору.

Очікуваний результат: вилітає куля та при влучанні у ворога його життя знижується

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

Рисунок 4.28 – Постріл гравця

Тест 4. Тестування підбору бонусів.

Вхідні дані: гравець підходить до боєприпасів.

Очікуваний результат: боєприпаси зникають а кількість патронів зростає.

Отриманий результат: співпадає з очікуваним (рис. 4.29-4.30).

Рисунок 4.29 – Кількість патронів до підйому боєприпасів

Рисунок 4.30 – Кількість патронів після підйому боєприпасів

Тест 5. Тестування смерті гравця.

Вхідні дані: вороги атакують гравця.

Очікуваний результат: здоров’я гравця впаде до 0 та відкриється вікно завершення гри.

Отриманий результат: співпадає з очікуваним (рис. 4.31).

Рисунок 4.31 – Вікно завершення гри

Всі проведені тестування мали успішний результат.

Висновки до розділу 4

Протягом розділу було створено та описано елементи гри на основі ігрового рушія Unity. Створено скрипти для контролю персонажем, генерації рівня, та контролю ворогами. Також створено необхідні префаби, сцени та UI елементи.

Для цього всього було намальовано багато спрайтів для таких елементів, як: UI елементи, по-кадрова анімація для ворогів та головного персонажу, зброя та додаткові предмети, будинки та дороги.

Також було проведено функціональне тестування гри за 5 сценаріями, таких як: зміна налаштувань звуку та вібрації, перехід на рівні, постріли гравця, підйом додаткових предметів та смерть гравця. Тестування показало, що все працює належним чином.

ВИСНОВКИ

В ході виконання завдань поставлених у кваліфікаційній роботі бакалавра було розроблено гру на основі рушія Unity у жанрі екшн. Проаналізовано основні технології які можуть використатись для розробки застосунку та було обрано рушій Unity з мовою C#. Було проведено дослідження кількох інших ігрових застосунків, таких як: Risk of Rain, Enter the Gungeon, Vampire Survivors . З яких була здобута основна ідея та механіка для нашої гри. Записано основні компоненти, такі як вороги, предмети та здібності гравця, які планується реалізувати у грі. Також були описані основні можливості гравця.

Також було розроблено проєкт екшн-гри на основі рушія Unity, яка буде популяризувати ігри у розважальній сфері життя для підвищення позитивного настрою. Для цього було досліджено багато матеріалу з приводу розробки різних видів діаграм. Всі діаграми було створено за допомогою програмного забезпечення StarUML від компанії MKLabs Co. Було розроблено діаграми взаємодії, класів, компонентів, пакетів, стану та переходів, розгортання.

Програмно описано основні скрипти та менеджери які використовуються у грі. Описано керування гравця та створення основних систем таких як зброя, предмети, мапа та вороги. Проведено тестування основних геймплейних механік.

Завдання, які були поставленні в кваліфікаційній роботі бакалавра, були виконані у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Moon Studios GmbH. Ori and the Blind Forest. Steam. URL: https://store.steampowered.com/app/261570/Ori_and_the_Blind_Forest/ (date of access: 31.01.2023).

2. Devolver Digital. Enter the gungeon. Steam. URL: https://store.steampowered.com/app/311690/Enter_the_Gungeon/ (date of access: 31.01.2023).

3. Gearbox Publishing. Risk of rain 2. Steam. URL: https://store.steampowered.com/app/632360/Risk_of_Rain_2/ (date of access: 31.01.2023).

4. Poncle. Vampire survivors. Steam. URL: https://store.steampowered.com/app/1794680/Vampire_Survivors/ (date of access: 31.01.2023).

5. Use case diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/use-case-diagram (date of access: 14.03.2023).

6. Sequence diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/sequence-diagram (date of access: 16.03.2023).

7. Communication Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/communication-diagram (date of access: 18.03.2023).

8. Statechart Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/statechart-diagram (date of access: 20.03.2023).

9. Online Mockup. Wireframe & UI Prototyping Tool. Moqups. URL: https://moqups.com/ (date of access: 02.04.2023).

10. Табунщик Г. В., Каплієнко Т. І., Петрова О. А. Життєвий цикл та розроблення інформаційних систем. *Проектування та моделювання програмного забезпечення сучасних інформаційних систем*. Запоріжжя, 2016. С. 8–12.

11. Class diagram. StarUML documentation. URL: https://docs.staruml.io/working-with-uml-diagrams/class-diagram (date of access: 02.04.2023).

12. Component Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/component-diagram (date of access: 22.03.2023).

13. Profile Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/profile-diagram (date of access: 26.03.2023).

14. Package Diagram - StarUML documentation. *Introduction - StarUML documentation*. URL: https://docs.staruml.io/working-with-uml-diagrams/package-diagram (date of access: 30.03.2023).

15. Chan J. Chapter 1: Introduction to C#. *Learn C# in one day and learn it well : C# for beginners with hands-on project : the only book you*. United States, 2017. P. 10–11.

16. Learn. *Unity*. URL: https://unity.com/learn (date of access: 03.05.2023).

17. Photoshop. *Adobe*. URL: https://www.adobe.com/ua/products/photoshop.html (date of access: 04.05.2023).