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
«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

МОБІЛЬНА 2D ГРА У ЖАНРІ ADVENTURE НА ПЛАТФОРМІ UNITY

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409. 21810912

Студент _____Д. Д. Васильковський
«___»_____2022 р.

Керівник ст. викладач _____І. О. Кандиба
«___»_____2022 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
«___»_____2022 р.

Миколаїв 2022

Завдання на виконання кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Кандитат техн. наук, доцент
каф. інж. прогр. забезп.
_____Є. О. Давиднко
« __ » _____ 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

Васильковський Данило Дмитрович
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Мобільна 2D гра у жанрі Adventure на платформі Unity

Затверджена наказом по ЧНУ від «1» грудня 2022 р. № 314

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні:
Початкові дані: предметно-орієнтований контент, функціональні вимоги до ігрового застосунку.
Очікуваним результатом роботи є ігровий застосунок.

4. Перелік питань, що підлягають розробці:
Визначення ідеї гри, моделювання, створення основних ігрових механік гри, проектування програмного застосунку, створення інтерфейсу користувача, кодування та тестування гри.

5. Перелік графічних матеріалів:
Слайди презентації.

6. Завдання до спеціальної частини:
Сформувати перелік вимог до робочого місця, організації та обладнання робочих місць, санітарно-гігієнічних вимог, вимоги щодо освітлення, електробезпеки та пожежної безпеки.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Екології	Охорона праці

Керівник роботи ст. викладач Кандиба Ігор Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання
Васильковський Данил Дмитрович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання дипломної роботи

Тема:_ «Мобільна 2D гра у жанрі Adventure на платформі Unity»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	22.12.21	24.12.21	виконано
2	Огляд літератури за темою роботи	25.12.21	03.01.21	виконано
3	Складання календарного плану КРБ	05.01.21	8.01.21	виконано
4	Аналіз предметної області	10.01.21	24.01.21	виконано
5	Розробка ідеї гри	26.01.22	05.02.22	виконано
6	Розробка архітектури	07.02.22	19.02.22	виконано
7	Створення користувацького інтерфейсу.	21.02.22	22.04.22	виконано
8	Реалізація основних механік гри	23.04.22	10.05.22	виконано
9	Розробка спеціальної частини з охорони праці	11.05.22	13.05.22	виконано
10	Оформлення КРБ та презентації	16.05.22	20.05.22	виконано
11	Відгук керівника КРБ	23.05.22	28.05.22	виконано
12	Попередній захист	30.05.22	04.06.22	виконано
13	Рецензування	06.06.22	11.06.22	виконано
14	Завершення оформлення КРБ та презентації	13.06.22	18.06.22	виконано
15	Захист кваліфікаційної роботи			

Розробив студент ___Васильковський Данило Дмитрович__ _____

(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи старший викладач Кандиба Ігорь Олександрович

(*ступень, звання, прізвище, ім'я, по батькові)* _____

(підпис)

«____» _____ 2022 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Мобільна 2D гра у жанрі Andventure на платформі Unity»

Студент 409 гр.: Васильковський Данило Дмитрович

Керівник: старший викладач Кандиба Ігорь Олександрович.

Актуальність полягає у тому що ігрові мобільні за стосунки є одними із найпопулярніших у сфері розваг тому що деякі дослідження виявляють що мобільні ігри приносять прибуток набагато більший ніж ігри для персонального комп'ютеру або ігрових консолей. На цю популярність впливає те, що смартфон є у кожної людини в кишені і багато хто проводить свій вільний час використовуючи застосунки у смартфоні. Також більшість мобільних ігор направленні мають дуже велике охоплення аудиторії як різного полу так і різного віку.

Метою кваліфікаційної роботи є покращення ігрового процесу за рахунок вдосконалення графічного дизайну шляхом ігрового рушія Unity.

Об'єкт дослідження – ігровий процес гри у жанрі Adventure та шляхи його реалізації з використання рушію Unity.

Предмет дослідження – технології створення гри з ви користуванням рушію Unity та мови C#.

У першому розділі представлені основні поняття предметної сфери створення гри та дослідження особливостей двигуна Unity .

У другому розділі – створення ідеї гри.

У третьому розділі представлено опис процесу розробки ігрового застосунку.

В останньому розділі було розглянуто норми та заходи з охорони праці й техніки безпеки в кімнаті, де буде експлуатуватись програмний продукт.

В результаті виконаної роботи було зроблено висновки щодо створення ігрового за стосунку на платформі Unity.

Сторінок – 81, таблиць – 0, рисунків – 77, посилань – 0, додатків – 1.

ASTRACT

of the Bachelor’s Thesis

«Mobile 2D game in the Andventure genre on the Unity platform»

Student of group 409: Vasilkovskyi Danilo Dmytrovych

Supervisor: senior teacher Kandiba Igor Oleksandrovych.

The object of the study is the game process in the Adventure genre and ways of its implementation using the Unity engine.

The subject of the research - the technologies of creating games using the Unity engine and the C# language.

The goal is to improve the game process by improving the graphical design through the Unity game engine.

Relevance lies in the fact that the game mobile for the tables is one of the most popular in the sphere of entertainment.

The first chapter presents the basic concepts of the subject area of game development and features research engine Unity. In the second section - the creation of the game idea. In the third section, there is a description of the process of developing the game application. The last section was reviewed standards and procedures for health and safety in the room, where the product will be operated.

In the second section - creating the idea of the game.

The third section describes the process of developing a game application.

In the last section, the norms and measures on labor protection and safety in the room where the software product will be operated were considered.

As a result of the work was made conclusions on the creation of the game for the table on the platform Unity.

Pages - 81, Tables - 0, Figures - 77, References - 0, Supplements - 1.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Аватар – модель персонажа, якою керує користувач

Коллайдер – компонент, який має форму об’єкта і використовується для фізичних зіткнень

ПК - персональний комп'ютер

Prefab – збережений об’єкт у комплекті з усіма його компонентами, властивостями та дочірніми об’єктами

Рендер - процес отримання зображення по моделі за допомогою комп'ютерної програми

Скрипт - коротких описів дій, виконуваних системою

JSON – JavaScript Object Notation

Бос – складний ворог

Pop-up - це область відображення графічного інтерфейсу користувача (GUI), зазвичай невелике вікно, яке раптово з’являється («спливає») на передньому плані візуального інтерфейсу.

Геймплей – ігровий процес

ААА – умовна підмножина ігор які пошиються великими видавництвами

Event або івент – це повідомлення послане об’єктом після яке повідомить о здійсненій події.

ВСТУП

Актуальність кваліфікаційної роботи полягає в тому, що у наш час казуальні мобільні ігри стають все більш популярними, а деякі дослідження виявляють що мобільні ігри приносять прибуток набагато більший ніж ігри для персонального комп'ютеру або ігрових консолей. На цю популярність впливає те, що смартфон є у кожної людини в кишені і багато хто проводить свій вільний час використовуючи застосунки у смартфоні. Також більшість мобільних ігор направленні мають дуже велике охоплення аудиторії як різного полу так і різного віку.

У ігровій індустрії існують багато різних жанрів. Але на теперішній час жанри ігри виходять за рамки одного жанру комбінуючи у собі особистості різних жанрів. Так жанр Adventure у чистому виді майже не існуючий жанр. Сам цей жанр полягає у цікавих пригодах та історіях у який перебуває гравець. Але для більшого інтересу з боку гравця цей жанр комбінують з іншими такими як Action, RPG, Simulator тощо. Тому для гри було вибрано поєднання жанрів Adventure та Action. Жанр Adventure відповідає за історію та за мотивацію головного героя а жанр Action відповідає за геймплей.

Метою кваліфікаційної роботи є покращення ігрового процесу за рахунок вдосконалення графічного дизайну шляхом ігрового рушія Unity.

Для досягнення визначеної мети необхідно вирішити такі завдання:

1. дослідити предметну галузь;
2. проаналізувати аналоги, що вже існують;
3. розробити специфікацію вимог до гри у жанрі Adventure;
4. розробити архітектуру гри;
5. створити користувацький інтерфейс;
6. реалізація основних механік;
7. провести тестування розробленої гри;

Об’єктом кваліфікаційної роботи є ігровий процес гри у жанрі Adventure та шляхи його реалізації з використання рушію Unity.

Предметом кваліфікаційної роботи є технології створення гри з ви користуванням рушію Unity та мови C#.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Опис предметної сфери

Вирішення поставленої задачі вимагає розробки гри у жанрі Action-Adventure з простим та занурючим геймплеєм, але який буде кидати виклик гравцю. Користувачем гри є гравець який може налаштувати гру, подивитись рекорди які збереженні на девайсі та перейти до основного геймплею. Основний геймплей складається із того що посередині на сцені буде знаходитись аватар гравця а з різник сторін будуть виходити різні вороги які будуть намагатися в залежності від свого типу зашкодити гравцю в той ж час гравець повинен буде захищатись відстрілюсь від ворогів різними видами зброї та пересуваючись у обмеженому полі. Основна ідея це набрати як можна більше очків які будуть даватися після знищення ворога.

У грі будуть різни види ворогів які будуть відрізнятися один від одного як за графікою так і за способом зішкодження гравцю. Одні будуть повільно переслідувати гравця доки не доторкнуться його. Інші будуть також стріляти у гравця та задавати йому різні негативні ефекти. Також у грі будуть міні босси які будуть мати дуже багато здоров’я та боси з фазами унікальними діями але після яких будуть щедрі винагороди по типу скрині з якої також можна буде покращити здібності гравця у кількості одну, три або п’ять покращень не підіймаючи рівень гравця. З звичайних вбитих ворогів буде випадати із шансом деякі предмети наприклад: кристалик досвіду при підбиранні який буде давати досвід гравцю та після збору деяких кристаликів буде давати рівень гравцю та здібності на вибір гравця який буде посилювати характеристики гравця такі як броня, кількість здоров’я або кількість наносимо шкоди тощо. Також гравець може вибрати нові види зброї та вибирати посилення для них. Здібності також будуть ділитися на типи наприклад покращення параметрів самого гравця це пасивні навики, а отримання нової зброї або інших активних навичків такі як щит або дронів

навколо аватара, маскування це відносься до активних навичків і кожного типу здібності гравець може обрати лише 5 штук. Також після є ще один тип здібності це додатковий він поліпшує вже обрані навички та їх може бути необмежна кількість. Самі вороги будуть різні в залежності від фази. Кожна фаза буде мати свій набір ворогів та босса а сама фаза буде змінюватися за часом і звісно кожна наступна фаза буде складнішою за минулу. Сама гра буде закінчуватися коли у гравця не залишиться здоров'я, буде випливати вікно яке буде показувати рекорд гравця та яке запропонує внести рекорд у таблицю підписавши ім'я. Після того як гравець ввід дані вони будуть зберігатися на пристрої і також відправляться на сервер де їх буде оброблено та якщо рекорд підійде то він буде внесений і в базу даних.

1.2 Аналіз існуючих рішень для реалізації проекту

Godot – відкритий мультиплатформенний 2D та 3D ігровой двигун. Двигун є максимально інтегрованим та самодостатнім середовищем для розробки ігор. Середовище дозволяє розробникам створювати ігри з нуля, не користуючись ~~ніякими~~ жодними інструментами, за винятком тих, які інтегровані у середовище (елементи графіки, музичні треки тощо). Процес програмування також вимагає зовнішніх інструментів (хоча за необхідності використовувати зовнішній редактор, це можна зробити відносно легко).

Загальна архітектура двигуна побудована навколо концепції графу дерева зі спадкових «сцен». Кожен елемент сцени (нода), будь-якої миті сам може стати повноцінною сценою. Тому при розробці можна легко змінювати всю архітектуру проекту, розширювати її елементи в будь-який бік і працювати з комплексними сценами на рівні простих абстракцій.

Переваги платформи:

1) відкритий код;

2) мультиплатформенний;

3) вибір мови програмування;

4) зручна система плагінів;

Недоліки:

1) не може обробляти 2D та 3D разом
2) повільний рендерінг
3) високий поріг входження

Ще одним із відомих ігрових двигунів є Unreal Engine. Це один із відоміших двигунів на якому було зроблено багато ААА ігор для персональних комп’ютерів та ігрових консолей. Цей двигун має– широкий набір інструментів світового класу, що прості та доступні для інтеграції у робочі процеси і дозволяють розробникам отримати швидко результати не торкаючись коду. Для цього у двигун вбудована графічна мова програмування. Також технології рендерінгу дозволяють створювати ігрові застосунки з красивою та сучасною графіко. І в наш час гіганти ігрової індустрії починають використовувати цей двигун для розробки ігор на мобільних платформах. Одні із найвідоміших це Mortal Kombal Mobile, Linage 2: Revolution тощо.

Переваги платформи:

1) зручні інструменти для графіки різних типів;

2) має вбудовану графічну мову програмування;

3) гарні інструменти оптимізації.

Недоліки:

1) складність для великих проектів;

2) великий обсяг програмного забезпечення;

3) вибагливість до потужностей комп’ютера.

Unity — багатоплатформовий інструмент для розробки відеоігор і застосунків, і рушій, на якому вони працюють. Створені, за допомогою Unity, програми працюють на настільних комп'ютерних системах, мобільних

пристроях та гральних консолях у двох та тривимірній графіці та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Переваги платформи:

1) низький поріг входження;

2) багато Бібліотек;

3) зручний інтерфейс;

4) мультиплатформеність;

5) зручна обробка фізики;

6) безкоштовність.

Недоліки:

1) повільність;

2) великовагкий;

3) повільність.

Unity має дуже велику бібліотеку, де можливо знайти різні матеріали для гри: графіку, допоміжні інструменти та музику. На Unity можливо розробити гру, не маючи навичок в створенні 3D та 2D графіки, або написанні коду. На теперішній час, Unity є основною платформою для створення мобільних ігор особливо для створення казуальних ігор. Саме тому, краще вибрати Unity як платформу для початку роботи в сфері відеоігор. Саме цей двигун і обрано для проєкту.

В самому Unity є система компонентів. Компоненти визначають поведінку ігрових об’єктів, до яких вони прикріплені та керують ними. Компоненти мають ряд атрибутів які можна налаштовувати у вікні Inspector. У самому Unity є багато різних компонентів які можуть відповідати за камеру, світло, звук, роботу з колайдерами та з фізикою об’єктів, анімацією, графічний інтерфейс тощо. Але все ж для створення ігрових застосунків потрібно буде і створювати власні компоненти та налаштовувати їх. Кожен компонент можна створити за допомогою скрипта, у яких вже описується

особлива ігрова логіка та поведінка об’єктів які вже і прикріплюються к об’єктам як компоненти. Кожен такий компонент повинен унаслідуватися від вбудованого класу MonoBehaviour. Саме усі ці компоненти створюються за допомогою мови C#.

C# – це об’єктивно-орієнтовна мова програмування яка була створена у 1998 році інженерами компанії Microsoft. Ця мову програмування була створення для використання у платформі .NET. C# відноситься до сім’ї мови C та має схожий синтаксис. Мова має статеву типізацію, підтримує поліморфізм, перегрузку операторів, делегати, атрибути, івенти та мову запитів LINQ. У порівняні з C++, C# набагато простіше через те, що має керовану пам’ять, тобто сам керує розподіленням пам’яті та запобігає витоку тощо.

Також колись Unity використовував як альтернативу мову програмування JavaScript або як її називали користувачі UnityScript але розробники припинили його підтримку через те, що C# набув набагато більшу популярність.

Також у проекті використовується текстовий формат обміну даними JSON оснований на мові JavaScript. Це дуже легка мова для читання людиною. JSON використовується по системі як ключ: значення. У цьому форматі будуть зберігатися дані гравця та рекорди. Також він є більш одним із найзручніших для обміну між іншими мовами програмування, особливу для обміну із сервером.

Також у проекті використовується ще одна особливість: Unity ScriptableObject. Це контейнер даних який можна використовувати для збереження великих об’ємів даних, незалежно від екземпляру класа. Один із варіантів зменшити використовування пам’яті у проекті, уникаючи копіювання значень. Це дуже корисно, коли у проекті є Prefab який зберігає незмінні дані в прикріплених скриптах.

Для взаємодією з сервером будуть використанні скрипти PHP. PHP це скриптова мова програмування яку використовують для роботою з сервером.

Для роботою з базами даних використовується система MySQL.

MySQL – вільна система керування базами даних.

1.3 Аналіз існуючих мобільних 2D ігор у жанрі Adventure

Play Market, App Store, Steam та інші – це все магазини цифрових товарів де можна завантажити різні ігри, фільми, музику, книги та інше. Контент на них є безкоштовним та платним.

Багато геймдизанерів комбінують різні жанри та механіки які були взяти з різник відомих ігрових застосунків. Тому для знаходження ідей потрібно ознайомитись із різними іграми у різних жанрах, особливо цікавими є жанри 2D, action, shooter, top down, casual, adventure де можна виділити наступні ігри:

Vampire Survivors – це гра у жанрі rogulike. У центрі з’являється персонаж якого можна обрати перед початком та із різних сторін наступають різні скелети, зомбі, вампіри та інша нежить. І головна задача це втриматись як можна довше і у цьому допоможуть кристалики досвіду та прокачка і також отримання здібностей, які дозволять наносити різноманітної шкоди ворогам.

Рисунок 1.1 – Скріншот ігрового процесу Vampire Survivors

Asteroids – одна із перших відомих ігор створена компанією Atari, стала найвідомішою грою «золотого століття» на ігрових автоматах 80-років. Суть

гри була у тому, що посередині знаходиться корабель який може повертатися та рухатись і стріляти тільки вперед для того щоб відбиватися від астероїдів які летять на нього та отримання за це балів.

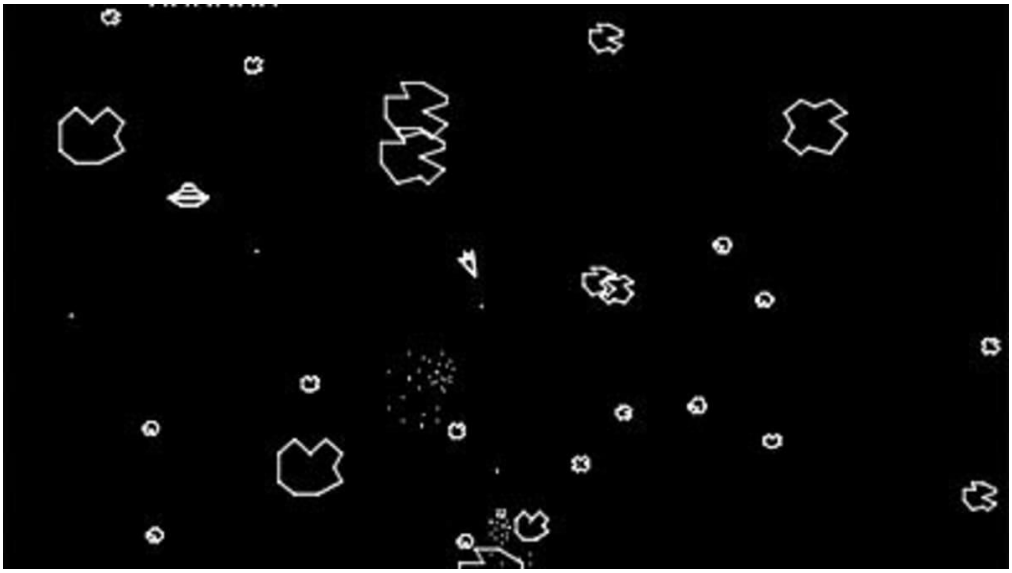

Рисунок 1.2 – Скріншот з гри Asteroids

Airstrikes 2D Skier Plane Shooter – ще одна гра у жанрі Action Adventure у якій гравець керує літаком та летить уперед де на нього летять різні вороги і задача протриматись за допомогою особистих здібностей та поліпшення характеристик літака і пройти рівень та дізнатися нову частину історії.

Рисунок 1.3 – Скріншот з гри Airstrikes 2D

1.4 Постановка завдання

Основна вимога проекту – це створити готову гру у жанрі Casual Action Adventure для мобільних платформ. Аналізуючи попередні розділи було виявлено основні механіки та ідеї для гри. Так керування кораблем буде здійснюватися джойстиком який буде знаходитись у лівому нижньому боку та виділені основні види озброєння:

– Звичайний постріл він буде знаходитись спочатку гри. Постріл здійснюється у напрямок повороту аватару та вестись автоматично якщо на лінії стрільби є противник.

– Наведений постріл це постріл який буде здійснюватись по найближчому ворогу та летіти у його бік

– Ракета при отриманні цього озброєння буде з’являтись кнопка та цифри з кількістю ракет у наявності самі ракети можна буде отримати із шансом з противників і при натисканні кнопки запускається ракета яка летить у напрямок повороту аватару та при доторканню до ворога здійснює вибух який наносить шкоди ворогам поруч.

– Лазер також як і з ракетою буде з’являтись кнопка при натисканні якої з’являється промінь на наносить великою шкоди у напрямок аватару, шкода настільки велика що тільки босів не вбиває з одного пострілу але цей постріл буде мате великий час на перезарядку.

Усі види озброєння крім звичайного можна отримати після підвищення рівню та всі вони вважаються активними здібностями. До активних здібностей також можна віднести такі:

– Щит – при наявності щита коли ворог доторкнеться гравець не отримає ніякої шкоди а сам щит знищиться але через час знову з’явиться.

– Маскування – теж навичок за допомогою якого гравець зможе на деякий час стати невразливим та зможе проходити крізь ворогів, але і сам не зможе стріляти.

– Дрони – цей навичок додає коло кравця дрона який крутиться по орбіті аватару та наносить шкоди ворогу при дотику.

– Бомба – аватар позаду себе буде створювати бомбу яка працює так само як і ракета але не рухається і здитанує через час або при дотику.

– Ривок – навичок із яким теж з’явиться кнопка після натискання якої гравець швидко рухається вперед та задає шкоди усім на своєму шляху.

– Подвійний постріл – це поліпшення основної зброї яке додає ще один постріл збоку і так кожне поліпшення до п’яти пострілів.

– Постріл за пострілом – також поліпшення до основної зброї після пострілу буде автоматично здійснюватись ще один постріл

Також є пасивні здібності:

– Швидкість пострілу – постріл буде здійснюватись швидше.

– Швидкість пулі – додає швидкість пулі.

– Збільшення шкоди - постріл наносить більше шкоди.

– Критична шкода шанс – додає шанс нанести звеличену шкоду ворогу.

– Збільшення здоров’я.

– Збільшення скорості.

– Броня – знижує шкоду яку гравець отримує від ворогів.

– Вдача – посилює шанс на випадіння предмету та кількості поліпшення з скрині.

– Збільшення отримання досвіду.

– Повільне відновлення здоров’я.

Вороги теж діляться за типами:

– Звичайний ворог – має середні параметри у швидкості, шкоди та здоров’я.

– Повільний - ворог із великою кількістю здоров’я але повільний.

– Швидкий – ворог який дуже швидко рухається.

– Міна – при наближенні до гравця детонує.

– Пила – з’являється та не переслідує гравця йде лише по прямої.

– Чорна діра – ворог який телепортує аватар гравця на початкову точку.

– Троянське коло – після знищення з’являться менші копії.

Також після знищення ворог може залишити після себе предмет:

– Малий кристалик досвіду додає трохи досвіду гравцю.

– Середній кристалик досвіду.

– Великий кристалик досвіду зазвичай падає з босів.

– Аптечка - відновлює здоров’я гравцю.

– Бомба - при контакті знищує ворогів навколо.

– Ракета – додає ракету якщо є необхідна зброя.

– Заморозка – заморожує ворогів на час.

Також у грі буде представлене поле яке має обмежену дистанцію та якщо вийти за межі аватар буде переміщено до початкової точки та буде нанесене зішкодження. Усі ці навички та механіки можна комбінувати між собою щоб протриматись як можна довше та покращувати свій особистий рекорд.

Для інтерфейсу необхідно визначити які самі необхідні сторінки і це:

– Головна сторінка із якою можна перейти на інші та почати геймплей.

– Сторінка лідерів у якій можна подивитись рекорди.

– Налаштування для налаштування гри.

– Сторінка інформації для отримання інформації про творця гри та для матеріальної допомоги.

– Сторінка геймплею у якій відображаються кнопки, джойстик та інформація про здоров’я та досвід гравця.

– Сторінка паузи.

– Сторінка завершення гри

І також декілька спливаючих вікон:

– Вікно скрині

– Вікно паузи

– Вікно отримання рівня

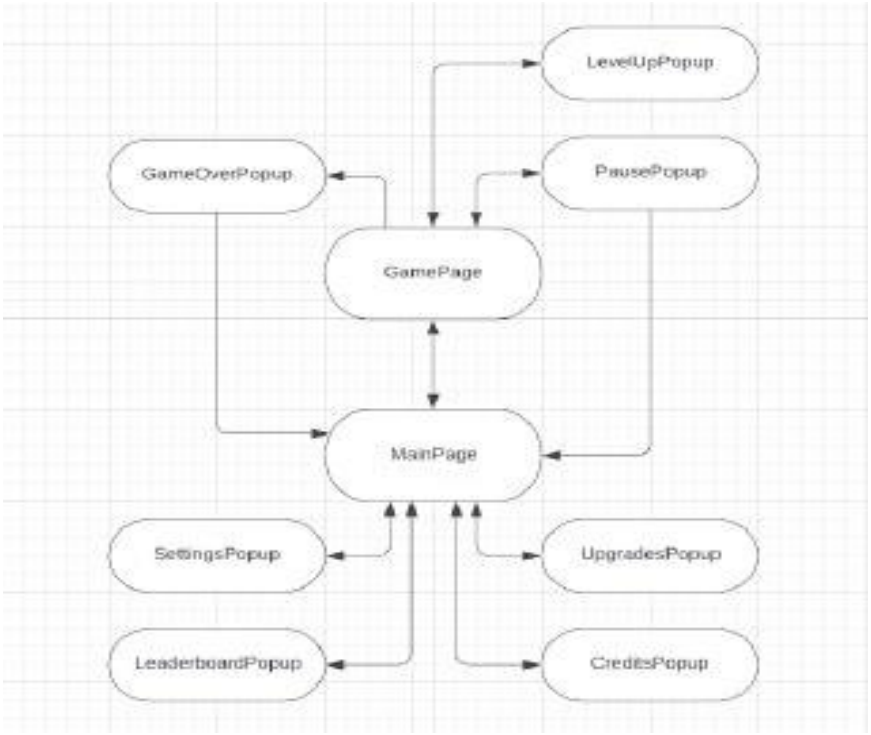

Рисунок 1.4 – Схема пересування між сторінками

1.5 Специфікація вимог до ПЗ

Специфікація вимог до гри - це повний опис поведінки системи що розробляється. Вона включає множину функціональних вимог які описують всі взаємодії, які користувачі мають з програмним забезпеченням.

Гра розробляється для використання у розважальних цілях. Сфера застосування: система розробляється для застосування у ігровій індустрії та ігрової діяльності. Загальна структура: головне меню, таблиця лідерів, ігрова сторінка, сторінка програшу, вікно підвищення рівня, вікно паузи. Загальне обмеження: гра може використовуватись лише на Android.

Мова і технологія розробки ПЗ:

1. Мова програмування – C#;
2. Ігровий двигун Unity Engine;
3. Середовище розробки Unity Editor 2021.1.15f1;
4. Adobe Illistrator 2020 - графічний редактор;
5. Система керування базами даних - MySql

Доступність: гра безкоштовна та доступна користувачам Andriod.

Продуктивність: гра повинна дуже швидко відповідати на кожну дію гравців та мати стабільні 60 FPS за для забезпечення комфортної гри.

Основні функції:

- переміщення гравця;
- автоматичний постріл;
- отримання шкоди;
- генерація ворогів
- знищення ворогів
- створення предметів
- підбирання предметів
- отримання нового рівня
- отримання нових здібностей
- генерування фаз
- створення боссу
- сундук
- завершення рівня та підрахунок даних
- внесення даних у базу даних та збереження на пристрої

Висновки до розділу 1

За результатами першого розділу було проаналізовано предметну сфери та було розглянуто основні технології для створення гри такі як рушій та мова програмування та розглянуто існуючи аналоги.

Крім того були розглянути аналоги інших мобільних ігрових проектів. З цих аналогів було розглянути основні особистості мобільних ігрових застосунків та було сформовано основні ідеї та механіки для гри. Було сформовано основні функції та насичення гри такі як здібності вороги та предмети.

Також було сформовано та обґрунтовано основний незвичний архітектурний підхід для проектів з використанням платформи Unity . Було сформовано види монетизації.

2 МОДЕЛЮВАННЯ ІГРОВОГО ПРОЦЕСУ

2.1 Діаграма варіантів використання та класів

Діаграма варіантів використання (сценаріїв поведінки, прецедентів) є вихідним концептуальним поданням системи в процесі її проектування і розробки. Ця діаграма складається з акторів, варіантів використання і відносин між ними.

Діаграма варіантів використання представлена на рисунку 2.1.

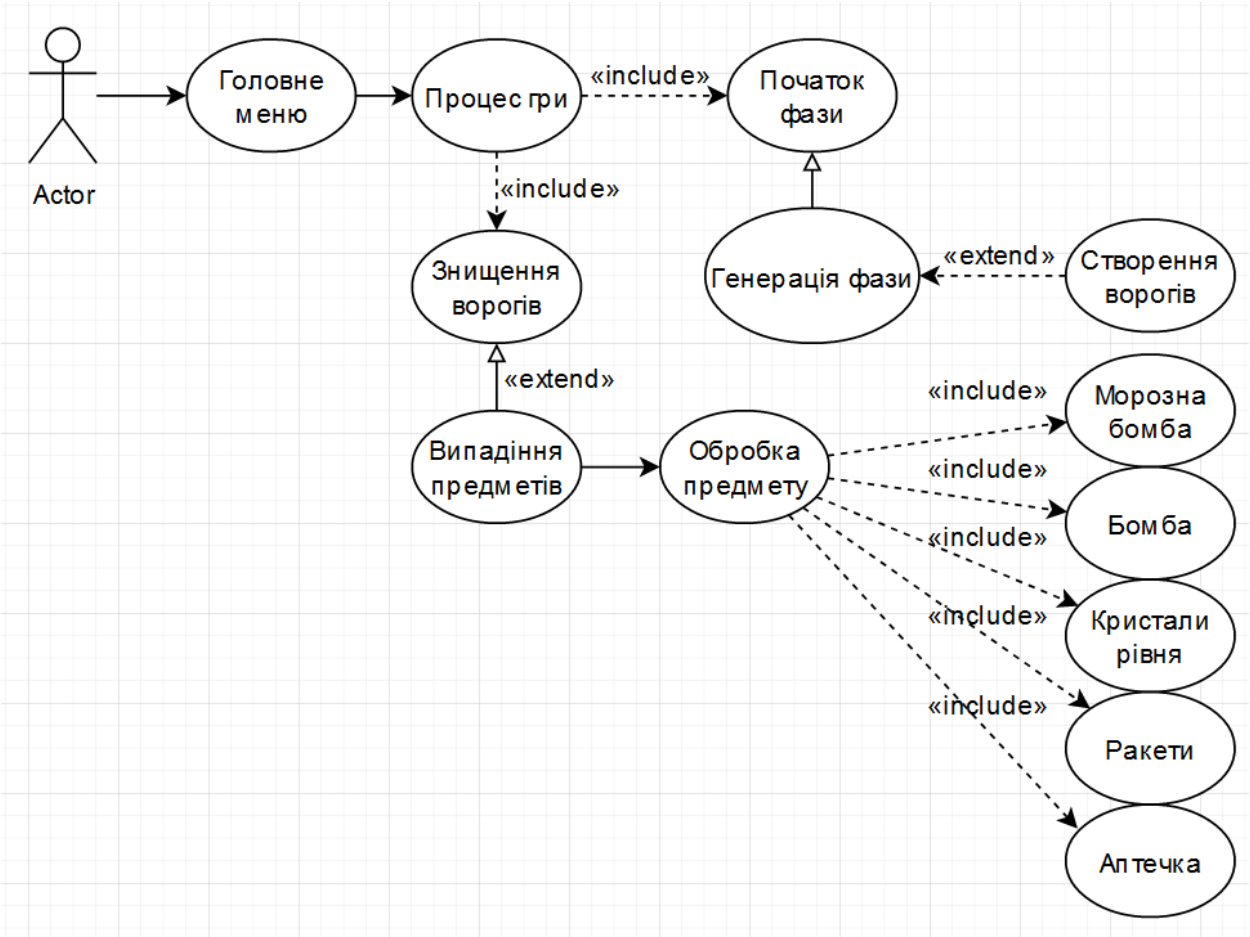

Рисунок 2.1 – Діаграма варіантів використання

Гра починається з головного меню з якого вже можна перейти до основного процесу гри. Після початку гри на екрані з різних боків з’являються вороги різних типів, сам тип вказується у фазах. У кожній фазі міститися свій набір ворогів, та час через який буде створено нового ворога.

Коли у фазі закінчиться набір ворогів, фаза оновиться та будуть з’являтися нові вороги.

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

- Ім'я;
- Ідентифікатор;
- Короткий опис;
- Актори (головні і другорядні);
- Передумови;
- Основний потік;
- Альтернативні потоки;
- Післяумови;
- Спеціальні вимоги.

Беручи в увагу усі елементи структури специфікації можна розробити наступні специфікації варіантів використання (ВВ):

А) Специфікація ВВ «показати данні таблиці лідерів»:

Ім’я: Показати данні таблиці лідерів

ID: 0

Короткий опис: система виведе на екран таблиці лідерів.

Основна діюча особо: гравець

Другорядні дійові особи: ні

Передумови:

1) Повинен бути зв’язок з Інтернетом

Основний потік:

1) ВВ починається коли гравець натисне кнопку «Leaderboard»
2) Система виведе гравцю локальну систему лідерів

Постумови:

1) Гравець натиснув кнопку «Global»
2) Система завантажила з БД глобальну таблицю лідерів

Альтернативні потоки:

1) Гравець повернувся у головне меню

Б) Специфікація ВВ «у змагальних цілях»:

Ім’я: у змагальних цілях

ID: 1

Короткий опис: система дасть гравцям змогу змагатися між собою

Основна діюча особа: гравець

Другорядні дійові особи: інші гравці

Передумови:

1) Повинен бути зв’язок з інтернетом

Основний потік:

1) ВВ почнеться коли гравець програє та введе свій нікнейм у спеціальне поле
2) Натиснувши кнопку «Back to menu» запуститься ВВ з ID - 0

Постумови:

1) Гравець почне нову гру без перевірки свого місця у рейтингу

Альтернативні потоки:

1) Гравець не зможе змагатися з іншими якщо не введе свій нікнейм.

В) Специфікація ВВ «скорочення часу»:

Ім’я: скорочення часу

ID: 2

Короткий опис: гравець скоротає час за грою аби відпочити від домашніх діл або роботи

Основна діюча особа: гравець

Другорядні дійові особи: ні

Передумови: ні

Основний потік:

1) ВВ почнеться коли гравець запустить систему

Постумови:

1) Гравець програв та вимкнув систему

Альтернативні потоки: ні

Г) Специфікація ВВ «грати у гру»:

Ім’я: грати у гру

ID: 3

Короткий опис: знайомство з грою, отримання ігрового досвіду

Основна діюча особа: гравець

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

Основний потік:

1) ВВ почнеться коли гравець натисне кнопку «Start»
2) Система почне відробляти свою функцію

Постумови:

1) Гравець керує персонажем та знищує ворогів підіймаючи предмети та прокачуючи ігрового персонажа

Альтернативні потоки:

1) Гравець повернеться у головне меню

Д) Специфікація ВВ «максимальна прокачка персонажа»:

Ім’я: максимальна прокачка персонажа

ID: 4

Короткий опис: досягнення максимальної прокачки гравця

Основна діюча особа: гравець

Другорядні дійові особи: ігровий персонаж, вороги, предмети

Основний потік:

1) ВВ почнеться коли запуститься ВВ з ID – 3.
2) Гравець керуючи ігровим персонажем – підіймає кристали рівня тим самим збільшує свій рівень та прокачуючи себе завдяки здібностям

Постумови:

1) Гравець досягне максимального рівня усіх свої здібностей

Альтернативні потоки:

1) Гравець програє

Аби будь-який з ВВ виконував свою роботу гравець повинен упевнитись у достатньому заряді свого пристрою аби під час виконання одного з ВВ пристрій не вимкнувся та дозволив гравцю отримати максимальне задоволення від конкретного ВВ.

2.3 Діаграма класів

Діаграма класів – це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, Такі як: класи, типи даних, їх зміст та їх відношення.

Діаграма класів архітектури представлена на рисунку 2.2

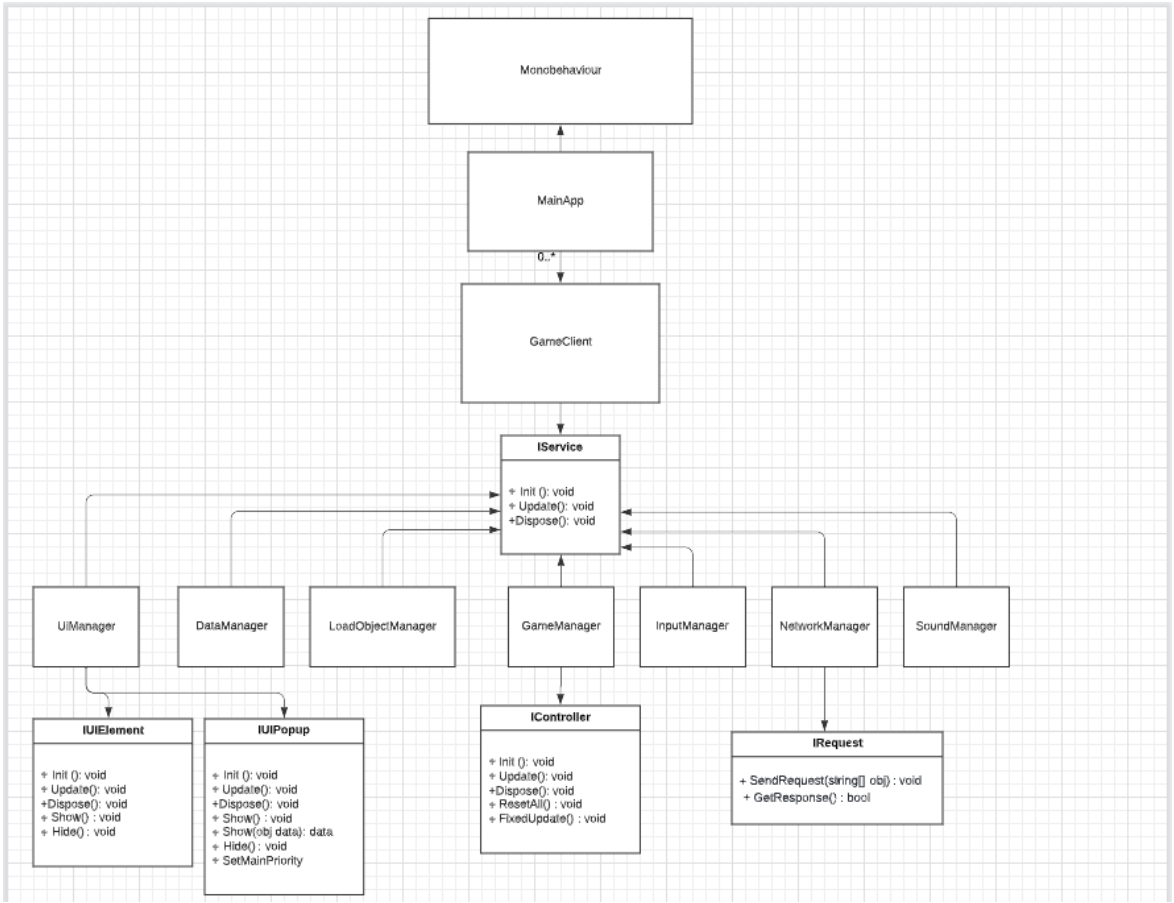

Рисунок 2.2 – діаграма класів архітектури

Сама архітектура незвична для проектів Unity тому що замість створення багатьох компонентів, створюється лише один MainApp і через

нього і проходять основні функції двигуна тобто обробка кадрів. Далі є GameClient який кєрує класами менеджерами через інтерфейс IService.

– UIManager керує користувацьким інтерфейсом через інтерфейси IUIElement, IUIPopup. Саме через цей менеджер проходить основна взаємодія з сторінками.

– DataManager використовується для збереження інформації та стану гри

– LoadObjectManager використовується для того щоб брати для використання різні звуки, префаби, або картинки.

– GameManager – головний менеджер який керує основним геймплеєм через класси які реалізують інтерфейс IController.

– InputManager класс який оброблює натискання кнопок на передає їх взаємодію через івенти.

– NetworkManger використовується для зв’язку з сервером та відправку запитів.

– SoundManager використовується для взаємодії з звуками.

Діаграма класів геймплею представлена на рисунку 2.3

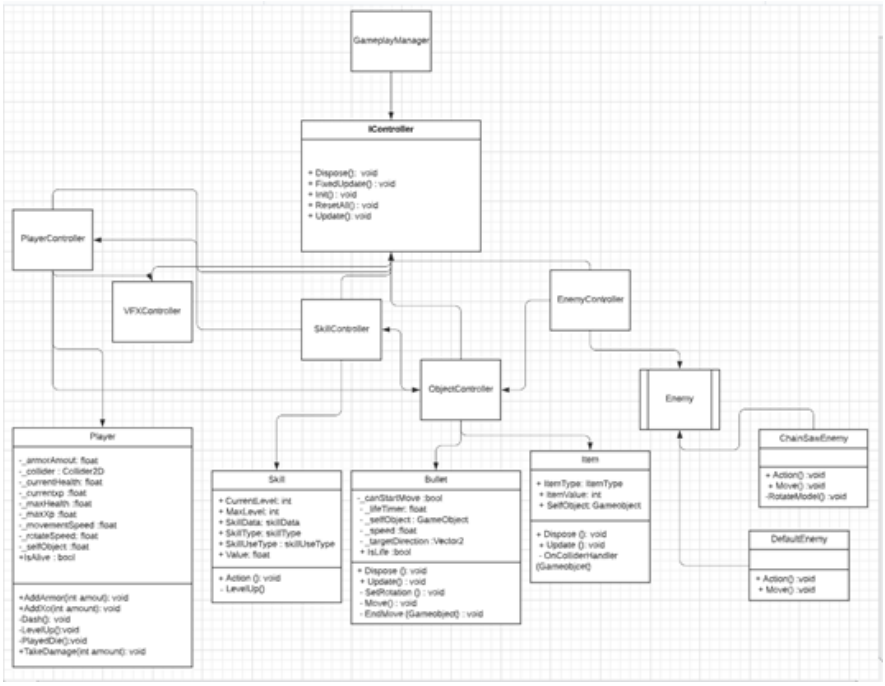

Рисунок 2.3 – діаграма класів геймплею

У діаграмі показані не всі класи, але показана основна взаємодія класів які керують саме геймплеєм. Є основний геймплейний менеджер який вже керує контролерами через інтерфейс IController.

– PlayerController контролер який керує моделлю гравця та його зброєю саме у ньому прописані основні функції покращання характеристик та здібностей гравця.

– VfxController контролер який містить у собі взаємодію з різними частками Vfx, наприклад: створює вибух після ракети або бомби, частинки після знищення ворога або гравця.

– SkillController містить у собі взаємодію з здібностями гравця та покращенням рівня.

– ObjectController оброблює такі об’єкти як пулі та предмети які з’являються після знищення ворога.

– EnemyController оброблює ворогів, також відповідає за змін фаз та створення нових ворогів.

2.4 **Діаграми переходів та діяльності**

Діаграма переходів станів (State Transition Diagrams, STD) - це UML діаграма, що демонструє поведінку розроблюваної програмної системи при отриманні керуючих впливів (ззовні). За допомогою діаграм переходів станів можна моделювати подальше функціонування системи на основі її попереднього і поточного функціонування.

Діаграму переходів стану поразки представлено на рисунку 2.4

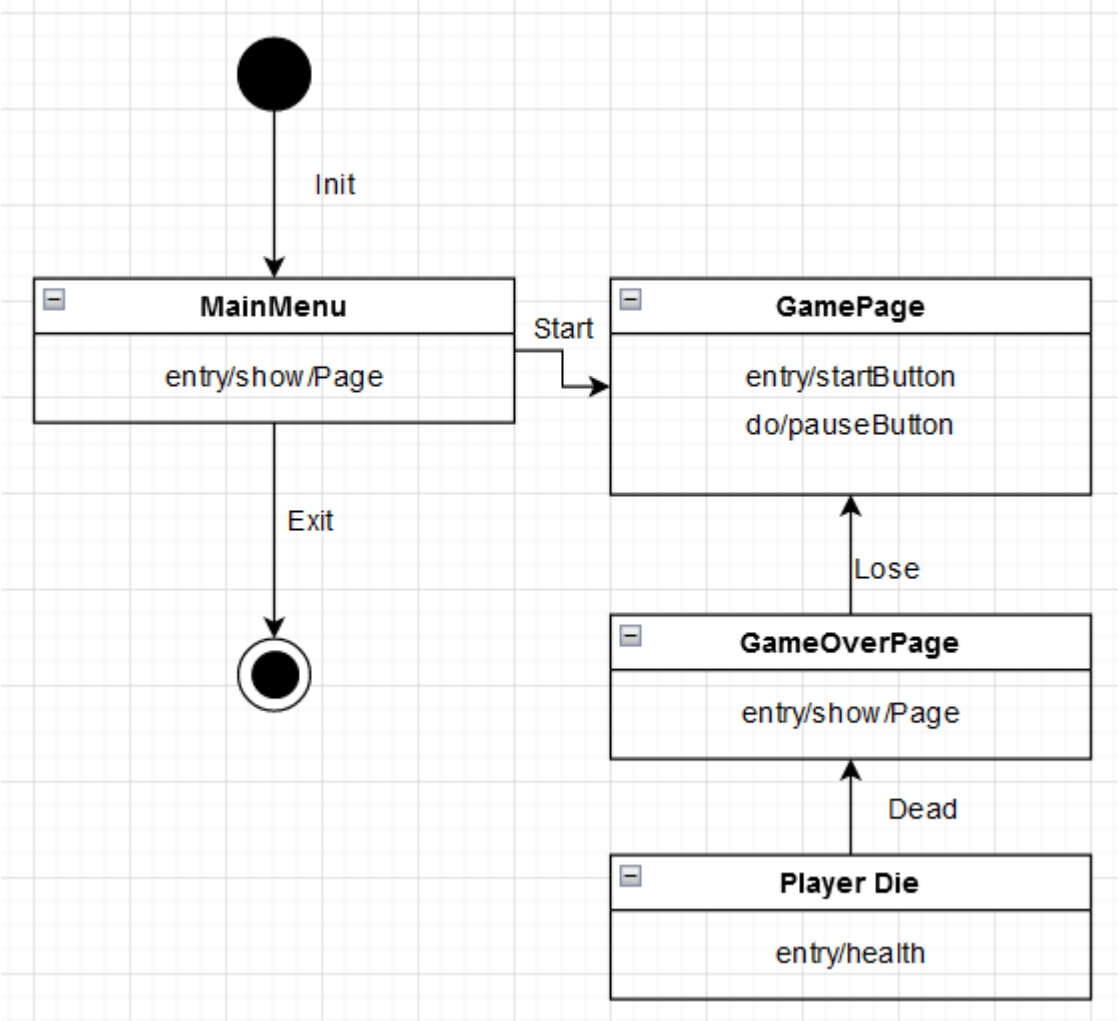

Рисунок 2.4– діаграма переходів стану поразки.

На діаграмі переходів станів поразки показано процес гри, в якому після смерті героя відкривається спливаюче вікно програшу зі статистикою за матч та можливістю ввести ім’я та зберегти цю статистику а також відправити до серверу і гравець має змогу перейти у головне меню.

Діаграму переходів стану активності гравця представлено на рисунку 2.5.

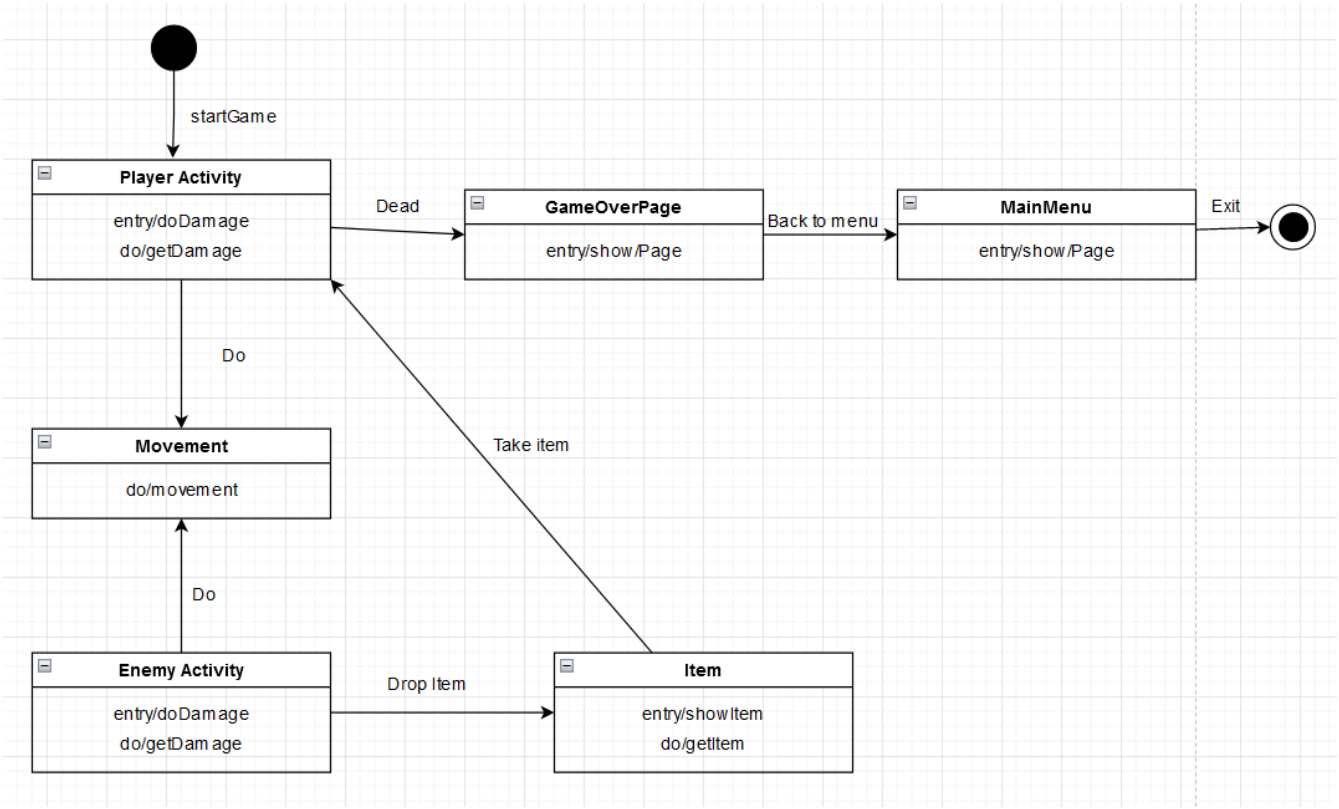

Рисунок 2.5 – діаграма переходів станів активності гравця

На діаграмі переходів станів активності гравця показано могливість пересування ігрока та можливість нанесення та отримання зішкодження де після закінчення здоров’я заявиться вікно повідомлення про поразку . Ворог має такі ж самі можливості але після смерті вже створює предмет який взаємодіє з гравцем.

Діаграма переходів станів процесу гри представлена на рисунку 2.6.

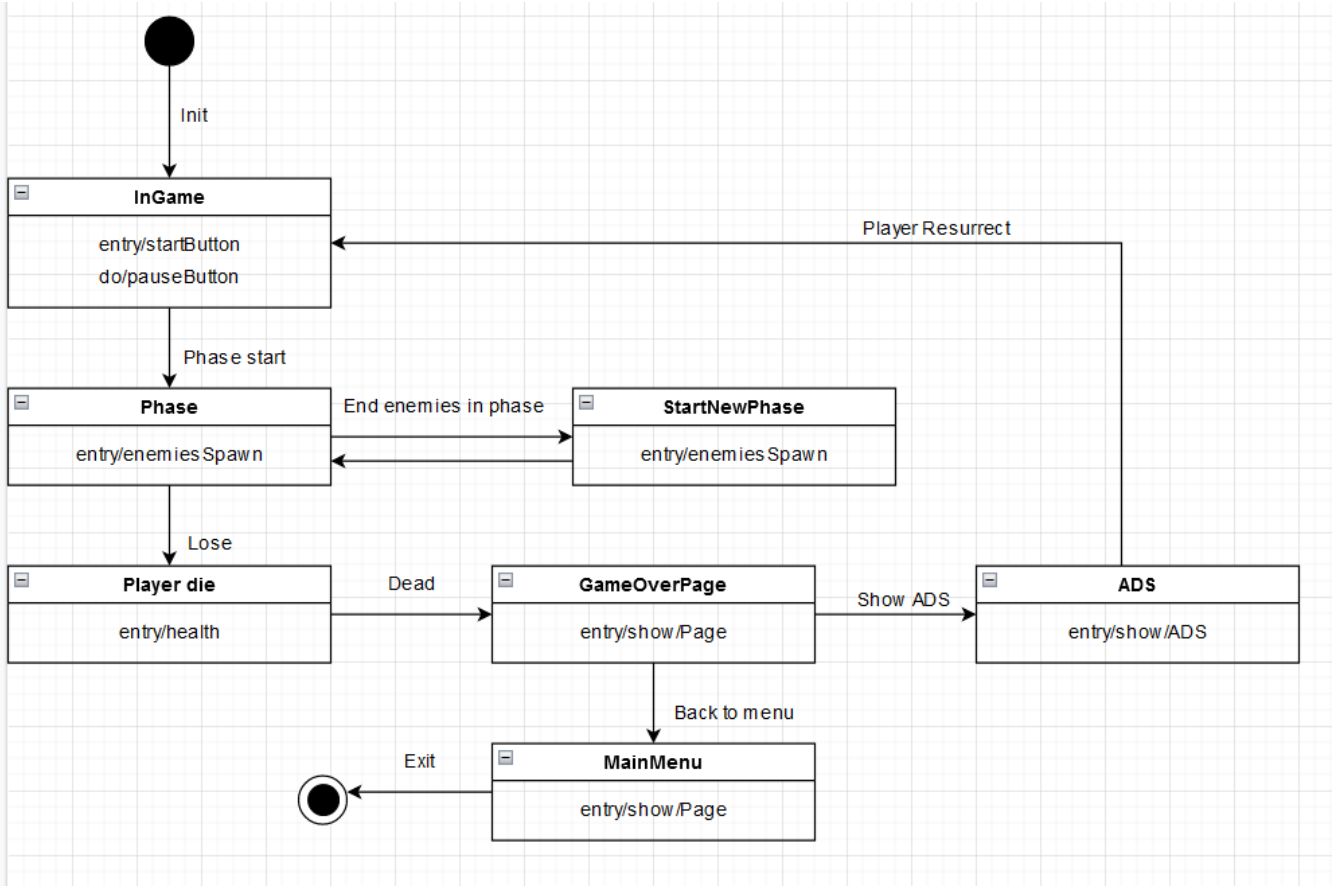

Рисунок 2.6 – діаграма переходів станів процесу гри

На діаграмі переходів станів процесу гри показано процес гри. Після старту починається генерація фази, на сцені з’являються вороги, після закінчення ворогів у фазі фаза оновиться. Також показано що після поразки гра закінчується. Але можна переглянути реклами і продовжити гру.

Діаграма діяльності представлена на рисунку 2.7.

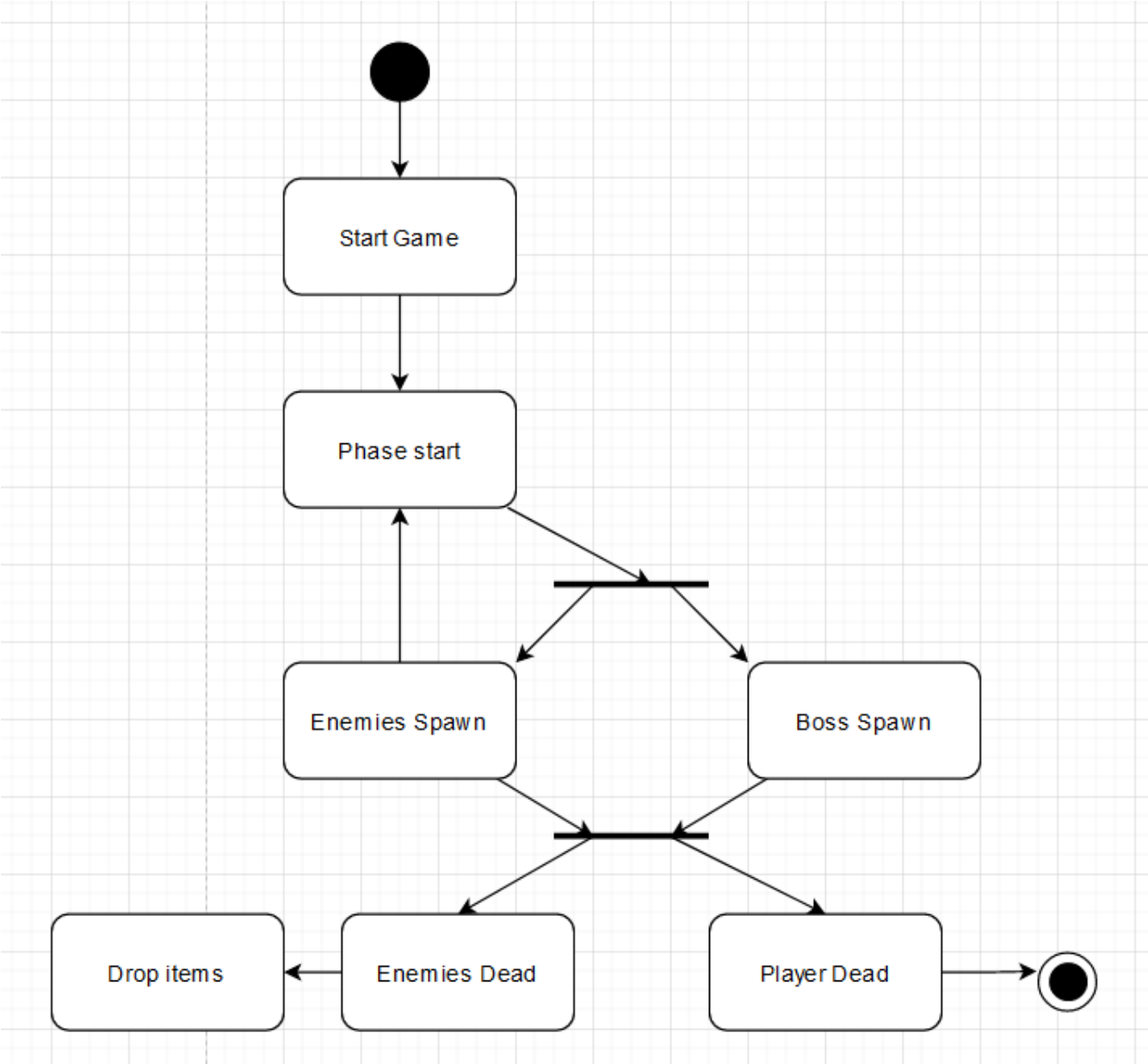

Рисунок 2.7– діаграма діяльності.

Діаграма діяльності – це UML-діаграма, на якій показані дії, специфікація виконуваної поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів - вкладених видів діяльності і окремих дій, з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого. Діаграми діяльності складаються з обмеженої кількості фігур, з'єднаних стрілками.

2.5 Діаграми розвертання

Діаграма розгортання — діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Діаграма розвертання представлена на рисунку 2.8

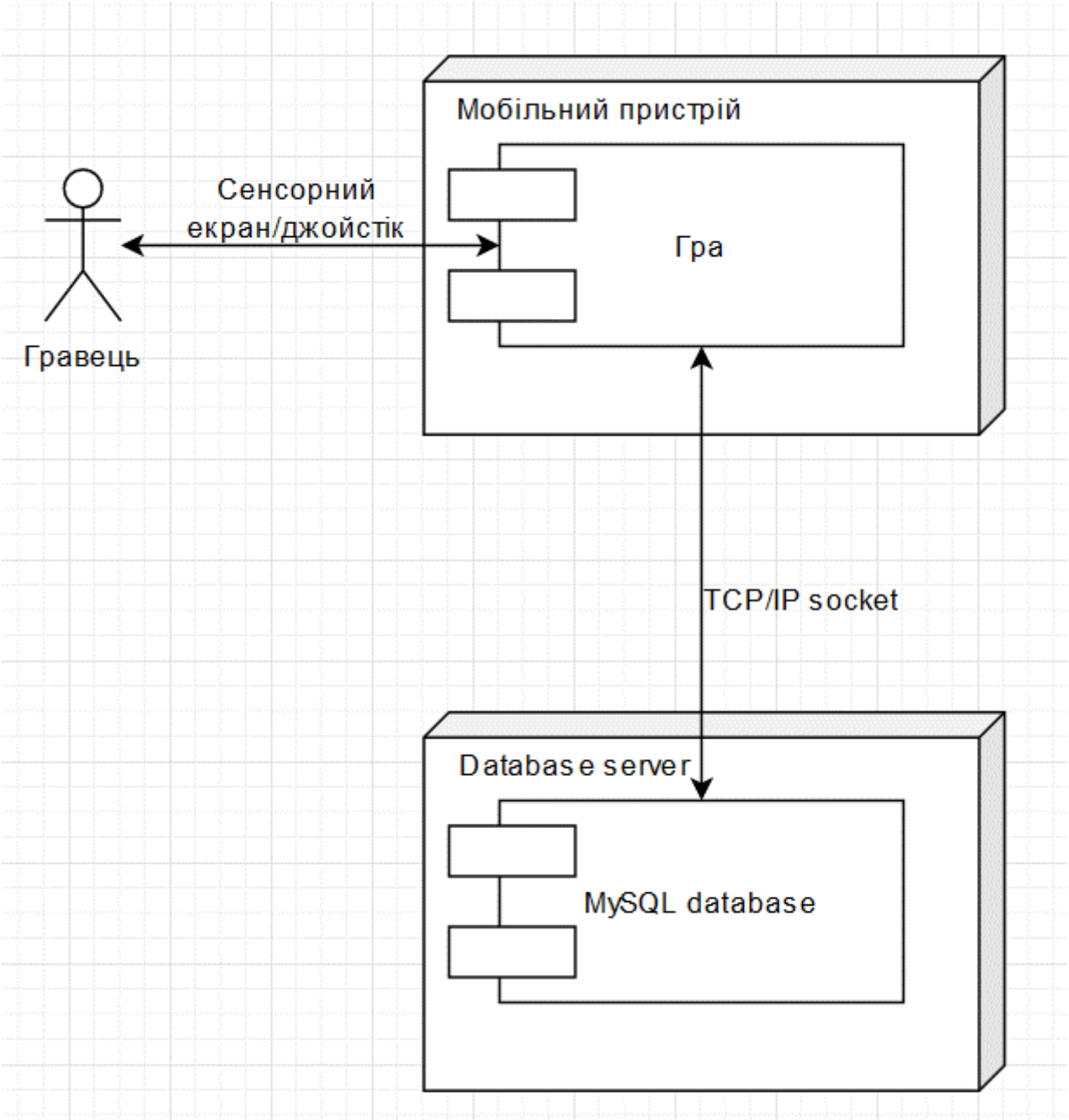

Рисунок 2.8 – діаграма розвертання

2.6 Діаграма бази даних

У базі даних представлення лише одна таблиця Records. Діаграма бази даних представлена на рисунку 2.9

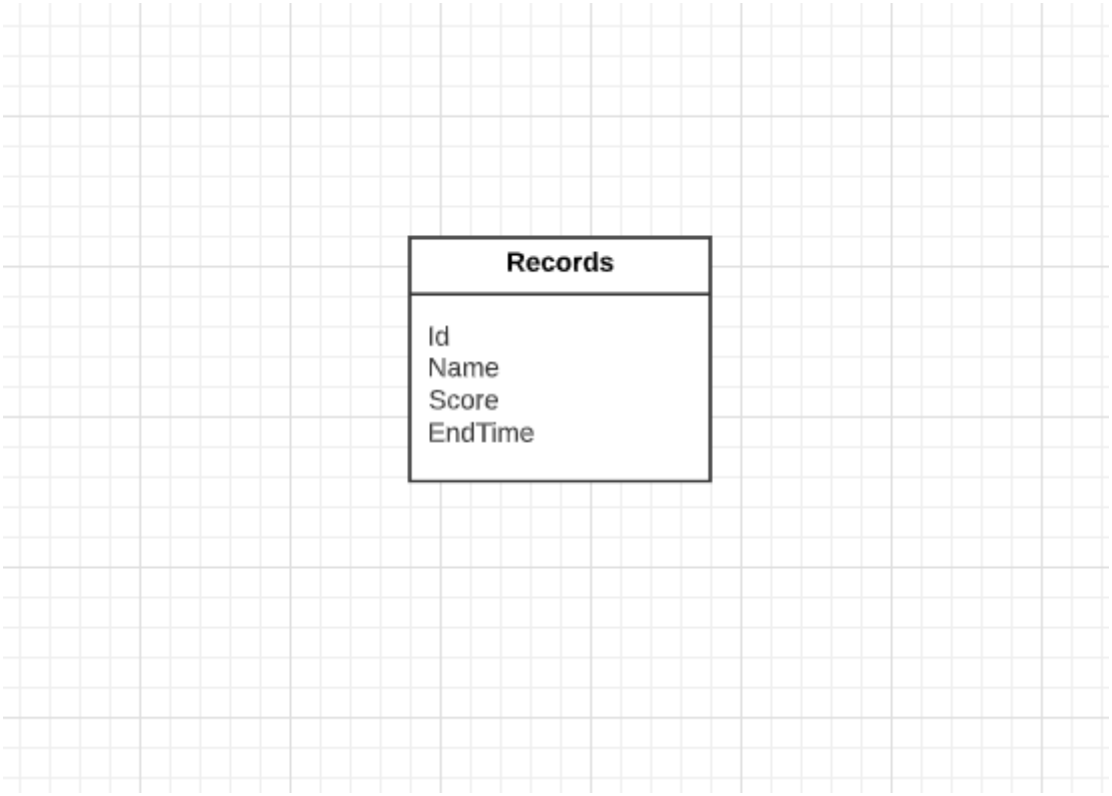

Рисунок 2.9 – Діаграма бази даних

Для гри достатньо лише однієї таблиці для того щоб розміщувати світові рекорди. Поле Name відповідає за ім’я яке введе користувач у поле у меню програшу. Поле Score відповідає за кількість балів обраних у грі та саме по цьому полі відбувається сортування. Та останнє поле це EndTime це час у який гравець закінчив свою спробу. До самих таблиці буде створено 2 запити:

1) перевірити можливість ввести нові данні, якщо даних більше 50 то перевірити чи більше рекорд за рекорд з найменшим значінням якщо так то видалити його та вставити новий;
2) отримати усі дані відсортовані по полю Score;

2.7 Створення користувацького інтерфейсу

При заходу у гру користувач побачить головне меню рисунок 2.10

Рисунок 2.10 – Головне меню

З головного меню користувач може переміститися до налаштування, старту геймплею, або перейти до таблиці рекордів також може перейти до вкладки покращення, налаштування та інформації про розробників але наразі ці функції недоступні і будуть дороблені у наступних версіях.

На сторінці рекордів рисунок 2.11 користувач може побачити рекорди які збережені на пристрої, або натиснувши кнопку Global перейти до світових рекордів які збережені у базі даних на сервері.

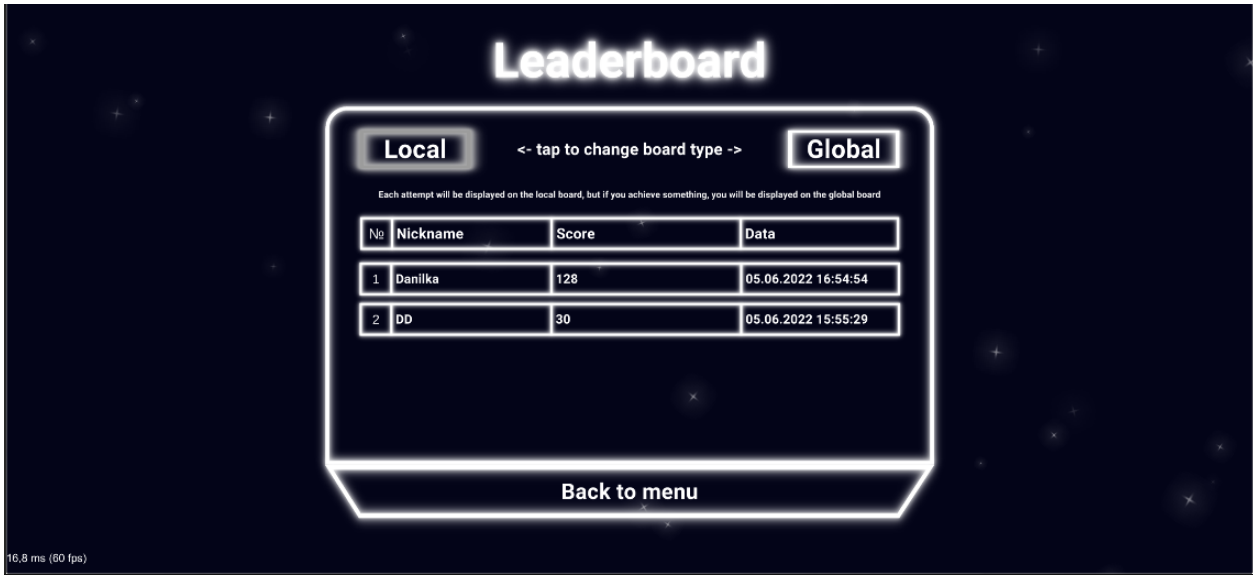

Рисунок 2.11 – Сторінка рекордів

Після старту гри буде і сама головна сторінка геймплею рисунок 2.12.

Рисунок 2.12 – Сторінка геймплею

На цій сторінці у нижньому лівому кутку знаходиться Joystick для керування рухом гравця. Сам гравець завжди знаходиться по середині екрану але він прив’язки до користувацького інтерфейсу. У верхньому лівому кутку знаходься здоров’я гравця та його поточний рекорд. Зверху смуга яка відповідає за досвід гравця та поточний рівень, по геймплею смуга буде зафарбовуватися та після повного зафарбування колір оновиться та цифра рівня збільшиться. У правому нижньому кутку можна побачити 4 кнопки, але вони будуть з’являтися при отриманні відповідної здібності. Кнопка ракети відповідає за запуск ракети та можна побачити поточну та максимальну кількість ракет. У правому кутку зверху кнопка до після натискання якої з’явиться меню паузи рисунок 2.13.

Рисунок 2.13 – Меню паузи

При появленні меню гра зупиниться та можна або відновити гру або повернутися до головного меню. Також можна побачити поточний рекорд.

При отриманні нового рівня з’явиться вікно покращення рівня рисунок 2.14.

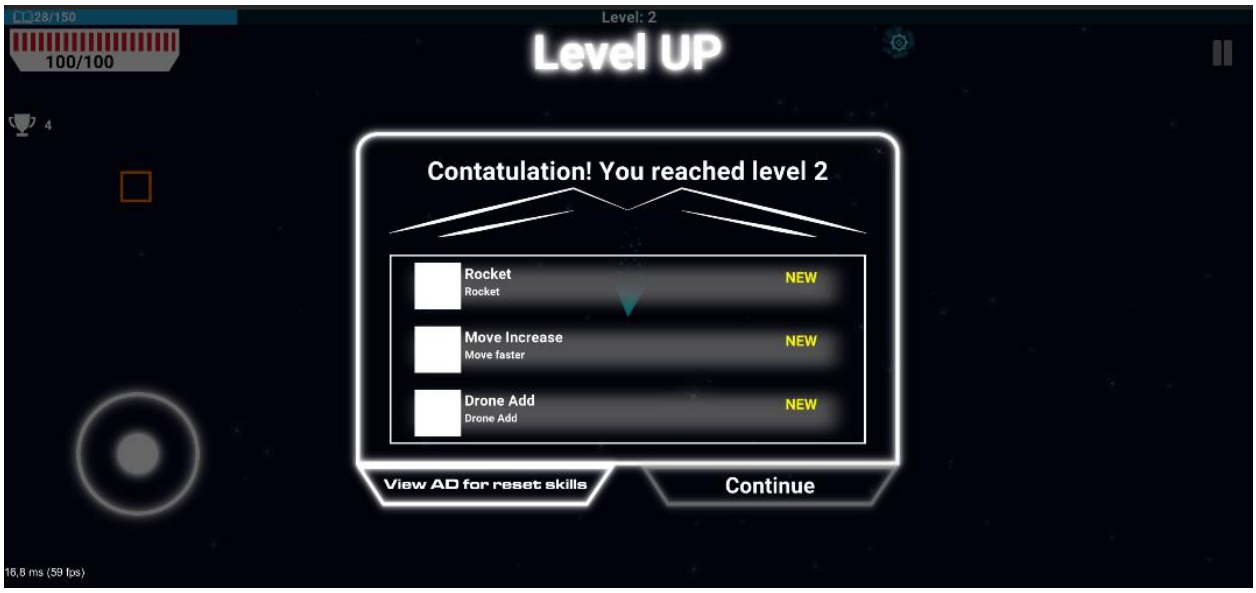

Рисунок 2.14 – Вікно покращення рівня

Вікно привітає гравця новим рівнем. Та заповнитися випадковими трьома здібностями у яких можна побачити їх назву, опис та ікону. Іконки з’являться у майбутніх оновленнях. Для отримання здібності потрібно обрати одну із трьох та натиснути кнопку продовжити. Також якщо гравцю не

подобаються випадкові здібності він може переглянути рекламу та оновити їх.

Після програшу з’явиться меню завершення гри рисунок 2.15

Рисунок 2.15 – Сторінка завершення гри

На сторінці гри можна побачити отриманий рекорд та ввести ім’я і зберегти на пристрої та відправити у базу даних та повернутися до головного меню. Але можна переглянути рекламу та відновити спробу але ця можливість доступна лише один раз за гру.

Висновки до розділу 2

Протягом другого розділу було створено діаграму взаємодії користувача з за стосунком. Створено та описано головні особистості діаграми архітектури гри та основного геймплею. Створено діаграми переходів станів та діяльності.

Розглянуто можливість взаємодії за стосунку з сервером та базою даних.

Також розроблено та описано основні елементи користувацького інтерфейсу такі як головне меню, таблиця рекордів, сторінка геймплею та сторінка програшу. Та вікна покращення рівня та паузи.

3 РЕАЛІЗАЦІЯ ГРИ З ВИКОРИСТАННЯМ ДВИГУНА UNITY

3.1 Створення архітектури гри

Для гри була вибрана архітектура у якій не використовуєтеся класи які унаслідуються від Monobehaviour тобто не використовуються компоненти а лише стандартний C#. Але все ж для взаємодією з двигуном один компонент повинен бути. Сам класс MainApp (Додаток А) зроблений по шаблону Singlton містить у собі обробку кадрів Update, та фіксовану обробку кадрів FixedUpdate. Для того щоб цей компонент працював необхідно створити на сцені пустий об’єкт та додати до нього цей скрипт як компонент Рисунок 3.1.

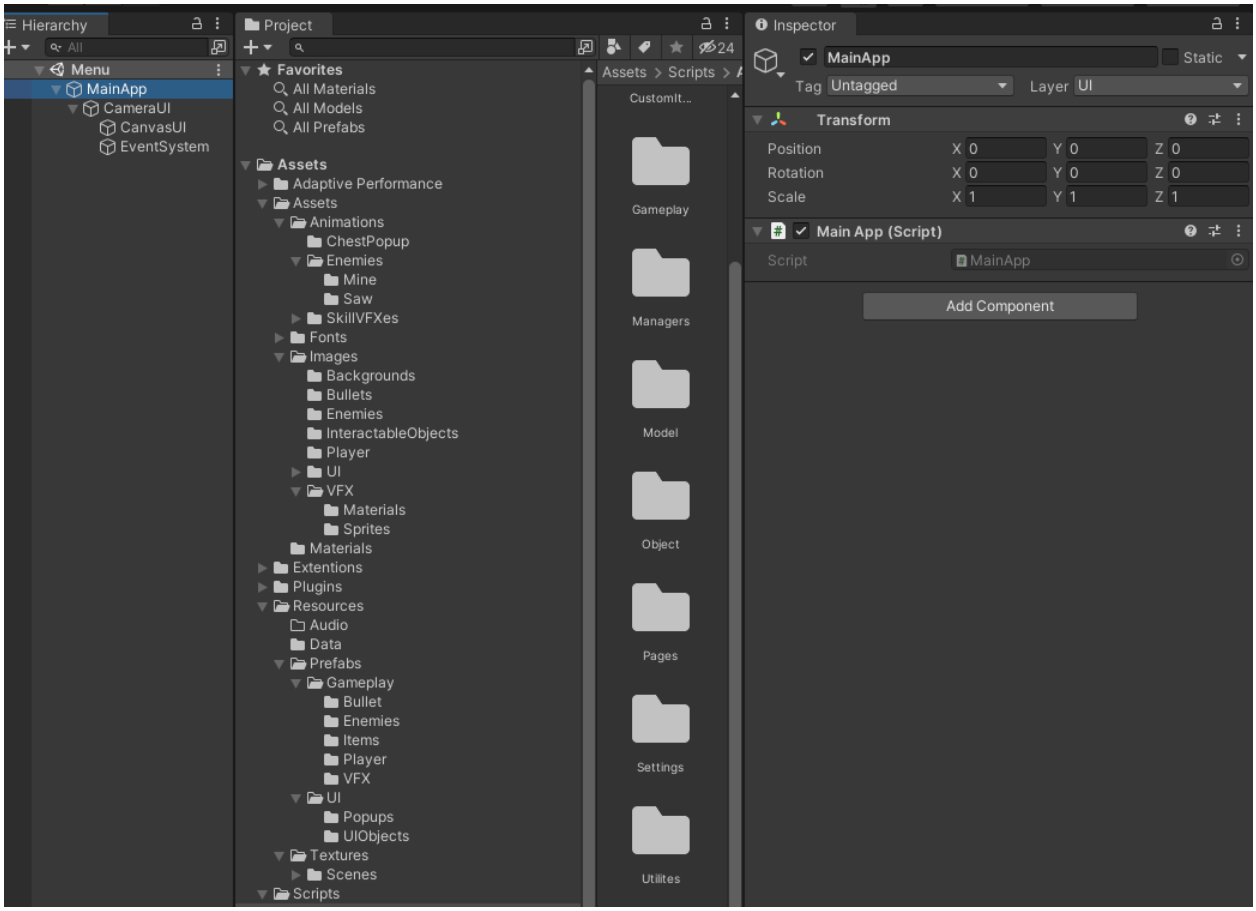

Рисунок 3.1 – Доданий компонент до пустого об’єкту

З початку на сцені вже розташована камера для обробки користувацького інтерфейсу та Canvas у якому і рендерить та розміщує у собі усі сторінка та вікна. І також EventSystem для обробки взаємодії з інтерфейсом.

Також потрібно створити ще один допоміжний компонент OnBehaviorHandler який буде вже міститись на ігрових об’єктах для того щоб обробляти Collider і взаємодію об’єктів при зштовхуванні.

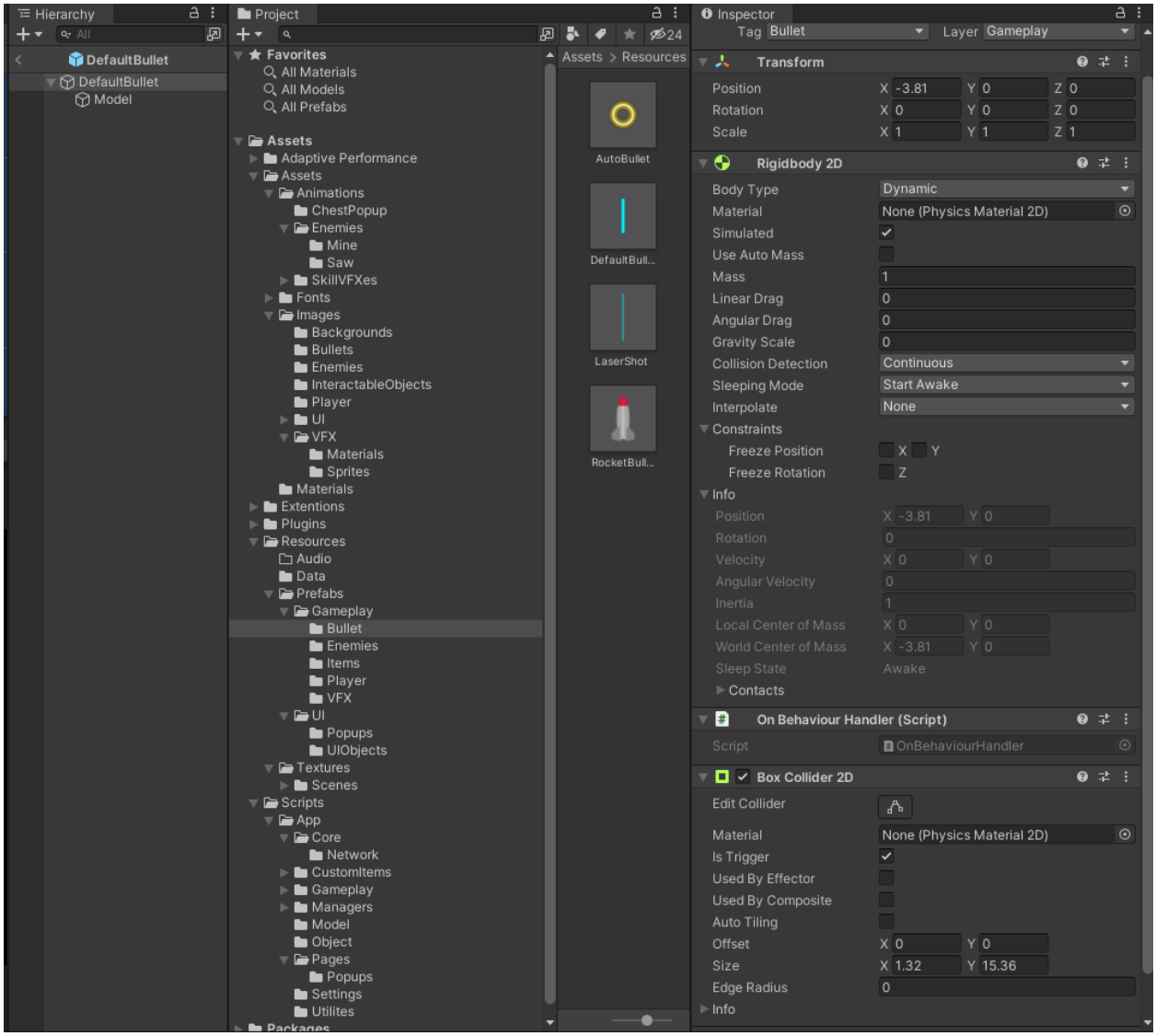

Рисунок 3.2 – Приклад доданого компоненту OnBehavoirHandler та Collider

Наступним кроком створення GameClient. Цей клієнт вже містить у собі усі менеджери які реалізують інтерфейс IService і зв’язує їх між собою. Також містить методи для зручного звертання до менеджера через унікальний інтерфейс який реалізує вже менеджер (Наприклад DataManager реалізує IDataManager).

Короткий опис усіх менеджерів та то для чого вони використовується описано у розділі 2. Але особливу увагу потрібно приділити UiManager

(Додаток Г). Цей менеджер містить ініціалізує усі сторінки та вікна які використовуються у грі а також містить у собі методи для включення необхідних елементів на виключення.

3.2 Програмування користувацького інтерфейсу

Усі графічні сторінки реалізують інтерфейс IUIElement а вікна IUIPopup Рисунок 3.3. У яких є методи ініціалізації, методи показу та хування. Також містять метод по кадрової обробки та методи відчистки. Але інтерфейс ще містить у собі можливість передати данні для подальшого використання та метод для того щоб вікно показувалось поверх інших елементів користувацького інтерфейсу.

```
1   using UnityEngine;
2
3   namespace TandC.RunIfYouWantToLive
4   {
5       public interface IUIElement
6       {
7           void Init();
8           void Show();
9           void Hide();
10          void Update();
11          void Dispose();
12      }
13
14      public interface IUIPopup
15      {
16          GameObject Self { get; }
17
18          void Init();
19          void Show();
20          void Show(object data);
21          void Hide();
22          void Update();
23          void Dispose();
24          void SetMainPriority();
25      }
26  }
```

Рисунок 3.3 – Інтерфейси IUIElement та IUIPopup

Перша сторінка яку бачить гравець це стартова сторінка яка містить у собі лише декілька кнопок для переходу на різні сторінки та старту геймплею але основу увагу має сторінка GamePage саме цю сторінку бачить гравець у процесі гри. При ініціалізації через LoadObjectManager до сцени

завантажується префаб сторінки та знаходяться усі елементи такі як різні тексти, картинки, панелі та кнопки. Усі кнопки через метод AddListener прив'язуються до методів які і будуть обробляти їх натискання. Наприклад кнопка запуску ракети яка активується при отриманні відповідної здібності, при натисканні на кнопку спрацьовує івент який оброблює метод. Вже сам метод передає до InputManager що кнопка була натиснута і вже цей менеджер відправляє до геймплейної частини та відбувається запуск ракети і кнопка стає неактивною на деякий час. Також сторінка містить обробку шкали досвіду гравця яка змінюється відповідно отриманому досвіду та шкала здоров'я яка реагує на зміну здоров'я гравця.

Також важлива сторінка у якій показуються рекорди LeaderBoardPage. У цій сторінці вже є свій особистий елемент таблиці UserEntry у якому виводить інформацію про рекорду через DataManger який розпаковує збережений файл на пристрої із рекордами або при натисканні кнопки Global при натисканні на яку робиться запит до серверу через NetworkManager та завантажує отриману інформацію з бази даних.

3.2 Програмування геймплею

Для геймплею використовується GameplayManager які містить у собі усі контролери та запити до них і також керує початком гри, її завершенням та переходом у паузу. При старті геймплею менеджер створює на сцені сам префаб геймплею Рисунок 3.4.

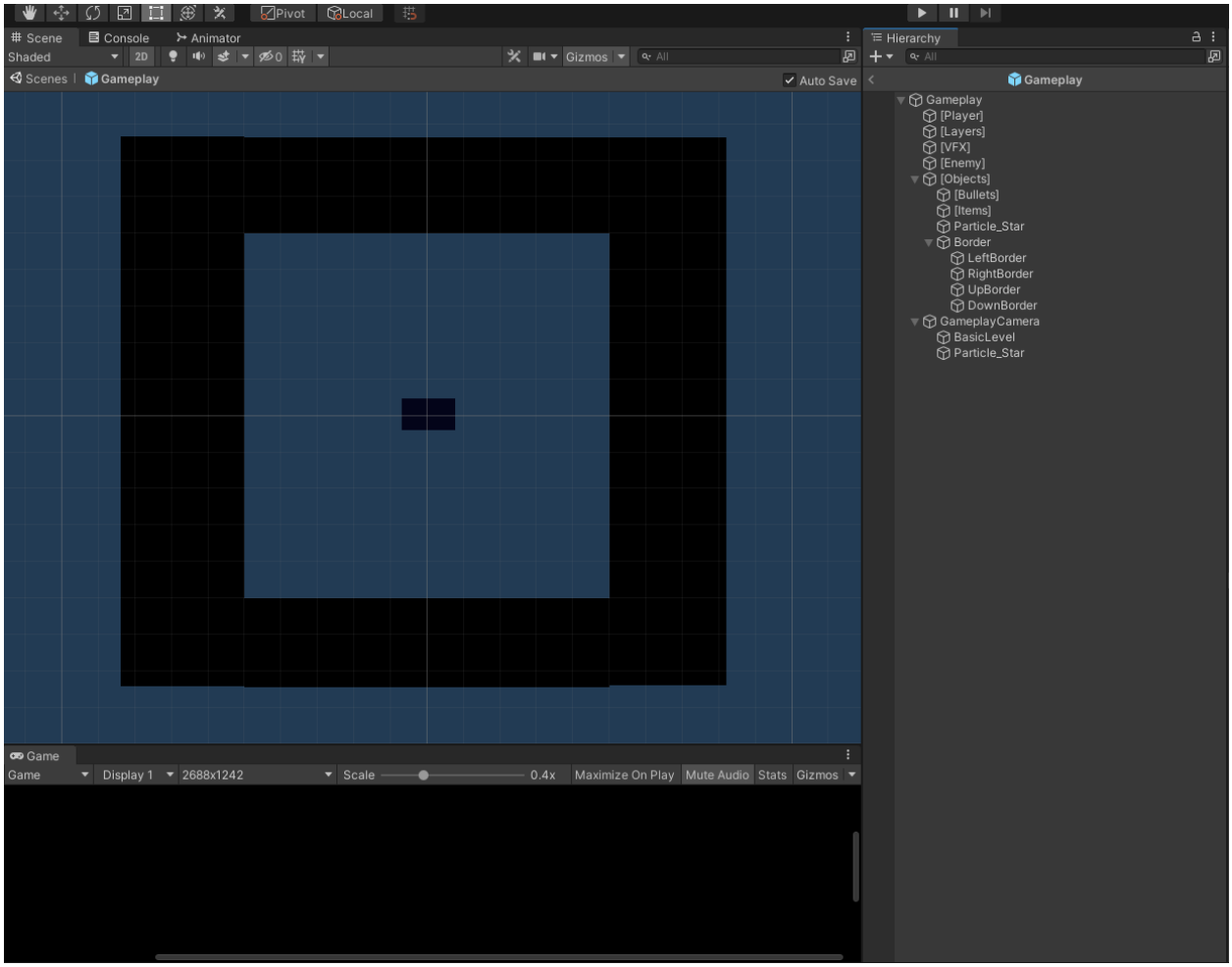

Рисунок 3.4 – Префаб геймплею

У префабі знаходяться різні пусті панелі у яких будуть розміщуватись відповідні елементи [Player] – Гравець, [VFX] – різні частиці Vfx, [Enemy] – вороги, [Bullets] – для пострілів. [Items] – для предметів. Також ще міститься границя до якої якщо доторкнеться гравець йому буде нанесенне пошкодження та переміщено у середину рівня. У геймплейній частині знаходиться і камера яка прив’язана до гравця і фон який вже прив’язаний до камери.

У грі повинна зберігатись інформація яку можна буде зручно змінювати через Unity а не через код. Для цього використовується ScriptableObject для якого потрібно створити спеціальний клас який і буде унаслідуватись від класу ScriptableObject.

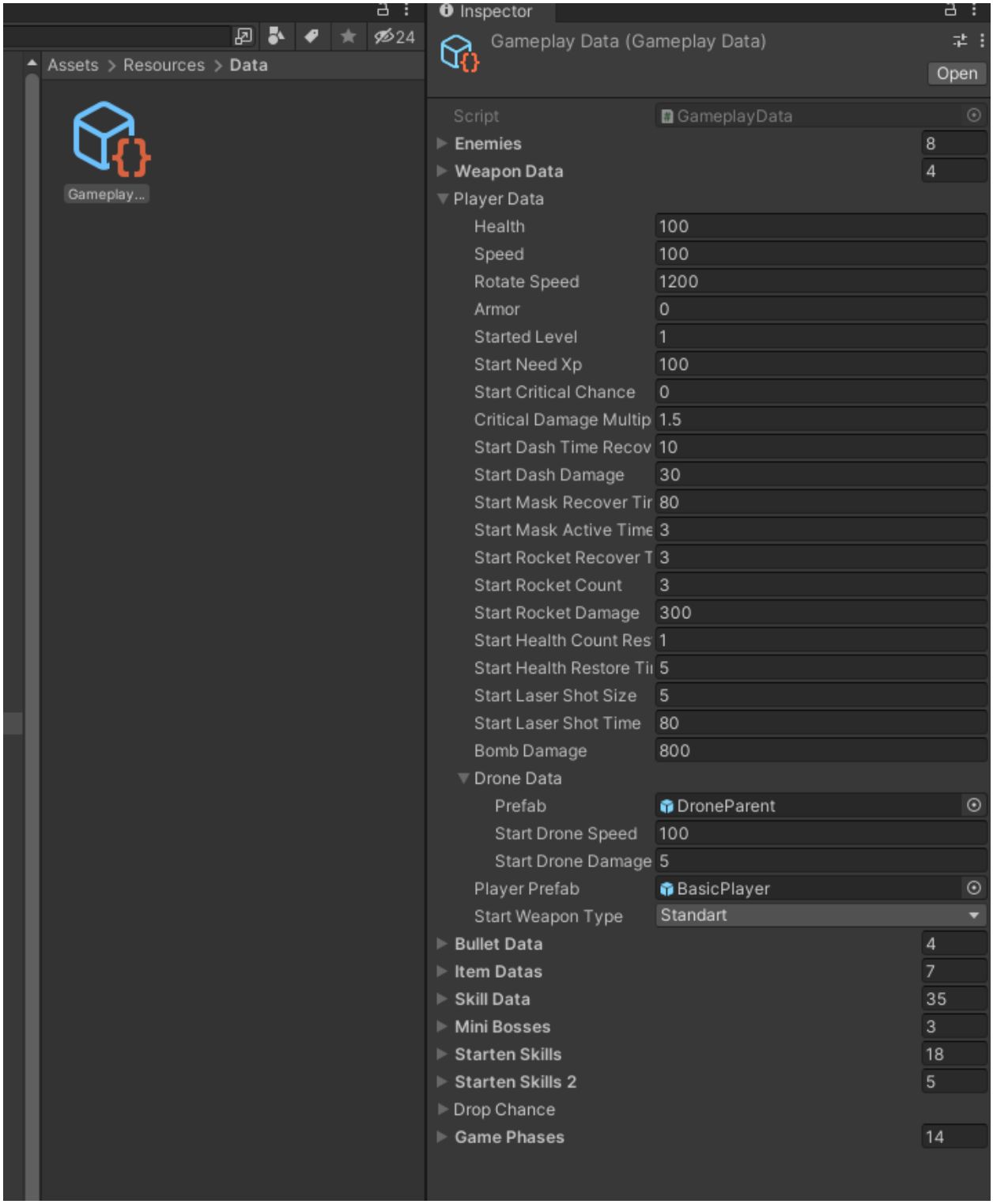

Рисунок 3.5 – Gameplay Data

3.3.1 Керування героєм

Для початку створення гравця потрібно зробити для нього префаб (Рисунок 3.6) який буде створюватись при початку геймплею.

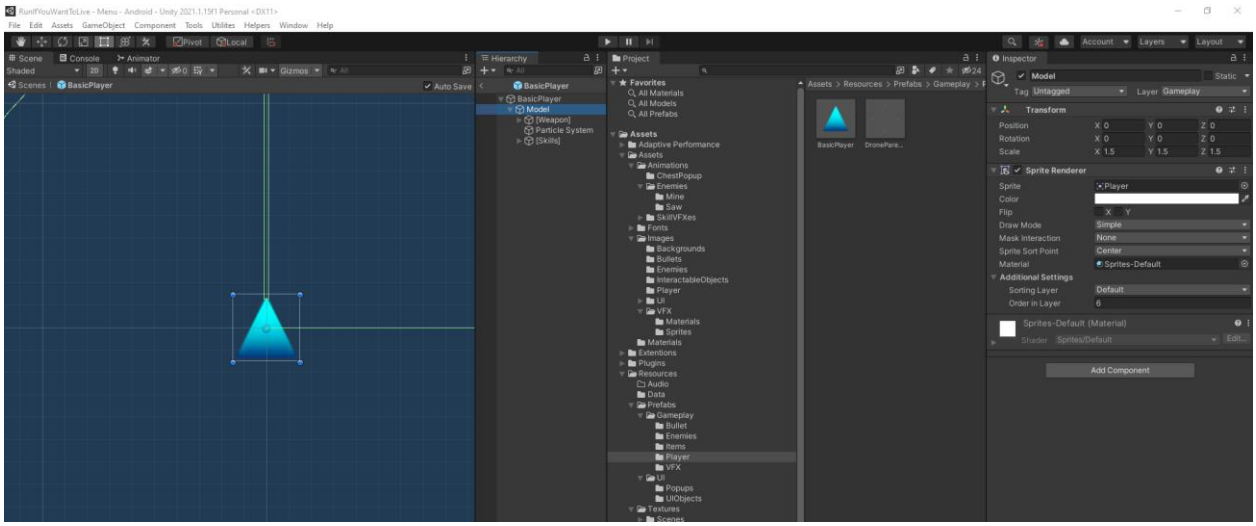

Рисунок 3.6 – Префаб гравця

Сам префаб це пустий об’єкт із колайдером для обробки зіштовхнення та панель у якій будуть розміщуватися обьекти для зброї і також панель для ефектів гравця. Сам об’єкт гравця створюється у PlayerController цей контроллер містить у собі гравця, та його зброю і також містить здібності та методи для покращення цих здібностей. Саме пересування гравця відбувається через керування джойстиком Рисунок 3.7

```
public void FixedUpdate()
{
    if (_selfObject == null || !IsAlive)
    {
        return;
    }
    if (_isDash)
    {
        Dash();
        //return;
    }
    if (IsMaskActive)
    {
        _maskTimer -= Time.deltaTime;
        if (_maskTimer <= 0)
        {
            EndMask();
        }
    }
    Vector2 movementDirection;
    movementDirection = new Vector2(_variableJoystick.Horizontal, _variableJoystick.Vertical);
    float inputMagnitude = Mathf.Clamp01(movementDirection.magnitude);
    movementDirection.Normalize();
    _selfObject.transform.Translate(movementDirection * _movementSpeed * inputMagnitude * Time.deltaTime, Space.World);
    if (_variableJoystick.Vertical != 0 && _variableJoystick.Horizontal != 0)
    {
        Quaternion toRotation = Quaternion.LookRotation(Vector3.forward, movementDirection);
        _selfObject.transform.rotation = Quaternion.RotateTowards(_selfObject.transform.rotation, toRotation, _rotateSpeed * Time.deltaTime);
    }
}
```

Рисунок 3.7 – Метод для пересування та повороту гравця.

3.3.2 Створення ворогів

Для створення ворога спочатку потрібно зробити префаб Рисунок 3.8.

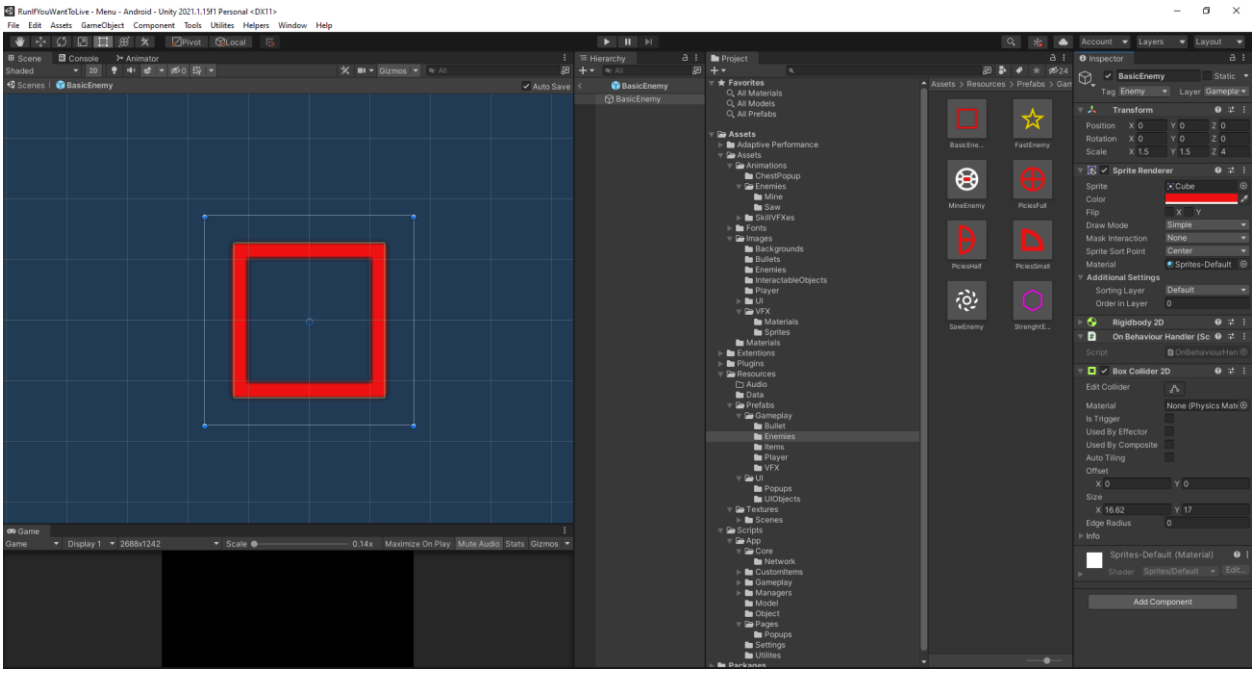

Рисунок 3.8 – Префаб звичайного ворога

Також у ворога є компонент Collider для обробки зштовхування, RigidBody який також відповідає за фізику об’єкта. За створення самих ворогів відповідає EnemyController, для цього э 3 функції, початок створення, отримання позиції створення та саме створення Рисунок 3.9

```
private Enemy SpawnEnemy(Enemies data, bool isBoss)
{
    Enemy enemy;

    switch (data.movementType)
    {
        case Enumerators.EnemyMovementType.ChainSawEnemy:
            enemy = new ChainSawEnemy(_parent, data, _playerController.Player.SelfTransform, SetSpawnPosition(), isBoss);
            break;
        default:
            enemy = new DefaultEnemy(_parent, data, _playerController.Player.SelfTransform, SetSpawnPosition(), isBoss);
            break;
    }
    enemy.IncreaseParam(_increaseEnemyParam);
    enemy.OnCollderEvent += EnemyBehaviorHandler;
    enemy.OnEndEnemyLifeTime += EnemyLifeTimeEnd;
    enemy.DestroyEvent += EnemyDestroyedEventHandler;
    return enemy;

}
public void StartSpawnEnemy()
{
    if (_enemies.Count >= 50)
    {
        return;
    }
    int enemyInPhaseId = UnityEngine.Random.Range(0, _enemysInPhase.Count);
    var enemyType = _enemysInPhase[enemyInPhaseId];
    _enemysInPhase.Remove(enemyType);
    _enemies.Add(SpawnEnemy(_gameplayData.GetEnemiesByType(enemyType), false));
}
private Vector2 SetSpawnPosition()
{
    Vector3 screenSize = new Vector3(_camWidth + 100, _camHeight + 100);
    Vector2 maxStartPosition = _playerController.Player.SelfTransform.position + screenSize;
    Vector2 minStartPosition = _playerController.Player.SelfTransform.position - screenSize;
    int chance = UnityEngine.Random.Range(1, 3);
    Vector2 firstPosition = new Vector2(0, 0);
    if (chance == 1)
        firstPosition = maxStartPosition;
    else if (chance == 2)
        firstPosition = minStartPosition;

    chance = UnityEngine.Random.Range(1, 3);
    Vector2 position = new Vector2(0, 0);
    if (chance == 1)
        position = new Vector2(UnityEngine.Random.Range(minStartPosition.x, maxStartPosition.x), firstPosition.y);
    else if (chance == 2)
        position = new Vector2(firstPosition.x, UnityEngine.Random.Range(minStartPosition.y, maxStartPosition.y));

    return position;
}
```

Рисунок 3.9 – методи для створення ворога

Самі данні для створення ворога потрібно також прописати у GameData Рисунок 3.10. У ньому прописане початкова кількість здоров’я, шкода, скільки гравець отримає очки, швидкість, початковий розмір, його тип та тип його пересування також шлях до префабу та його унікальний колір.

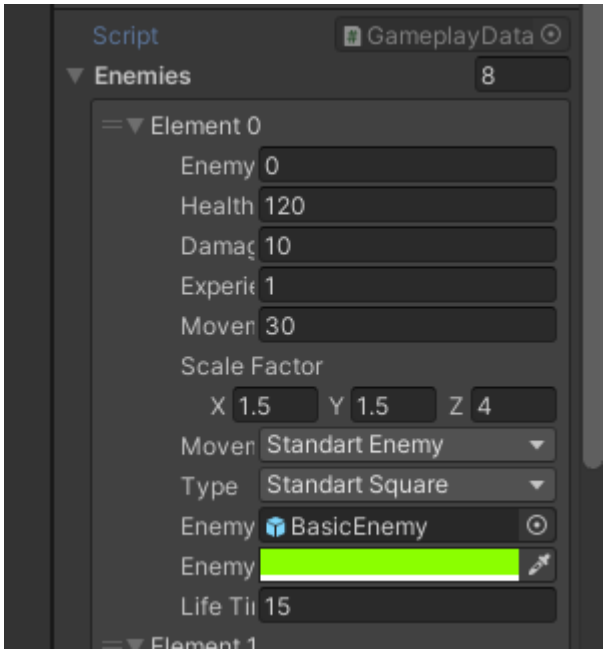

Рисунок 3.10 – Інформація про ворога

Для самого ворога потрібно створити відповідний клас. Так як у грі існують декілька ворогів з різними пересуванням то сам клас Enemy є абстрактним а усі вороги вже унаслідуються від нього і перевантажують методи пересування Рисунок 3.11.

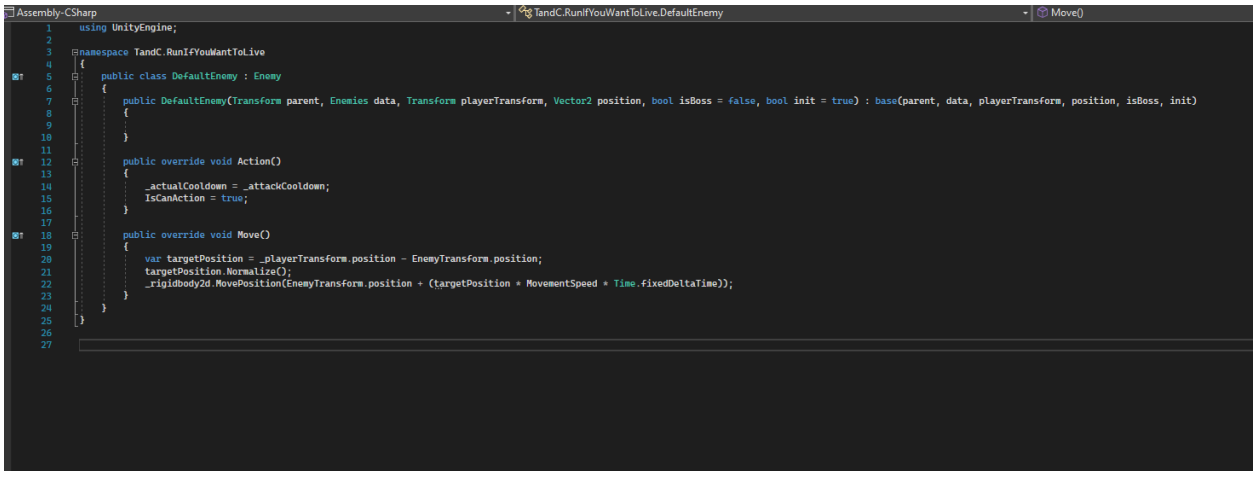

Рисунок 3.11 – Клас звичайного ворога який унаслідується від абстрактного Enemy

3.3.3 Система зброї

Для можливості зниження гравцю також потрібно додати зброю яка також міститься у PlayerController, усього у гравця на даний момент є 4 виду. зброї і кожна с з них має різний принцип дії, тому спочатку потрібно

створити також абстрактний клас Weapon. І від нього вже унаслідуються інші види зброї. Спочатку гри гравець має лише стандартний вид зброї але із отриманням нових здібностей може отримати і новий вид і PlayerController зареєструє її Рисунок 3.12

```
public void TakeWeapon(WeaponData data)
{
    Weapon weapon;
    bool isButtonWeapon = false;
    switch (data.type)
    {
        case Enumerators.WeaponType.Standart:
            _weaponLine = new WeaponLine(Player.SelfObject.transform.Find("Model/[Weapon]/ShootLine").gameObject);
            weapon = new DefaultWeapon();
            break;
        case Enumerators.WeaponType.RocketLauncer:
            isButtonWeapon = true;
            weapon = new RocketWeapon();
            break;
        case Enumerators.WeaponType.LaserGun:
            isButtonWeapon = true;
            weapon = new LaserWeapon();
            break;
        case Enumerators.WeaponType.AutoGun:
            weapon = new AutoWeapon();
            break;
        default:
            weapon = new DefaultWeapon();
            break;
    }
    weapon.Init(Player.SelfObject.transform.Find($"Model/[Weapon]/{data.weaponName}").gameObject, data, _gameData.GetBulletByType(data.type), _gameData.DropChance.StandartShotChance, isButtonWeapon);
    _allWeapons.Add(weapon);
}
public Weapon GetWeaponByType(Enumerators.WeaponType type)
```

Рисунок 3.12 – метод отримання нової зброї

Також кожну зброю потрібно прописати у GameData Рисунок 3.13. У якій прописано шлях до об’єкту зброї у префабі гравця, базову шкоду, швидкість пострілу та тип зброї.

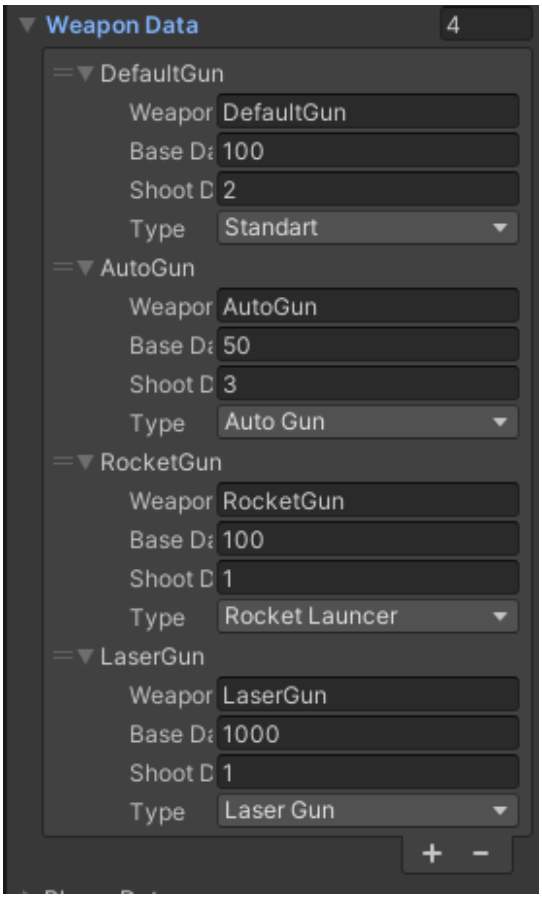

Рисунок 3.13 – Інформація про зброю у GameData

3.3.4 Система предметів

Після зштовхуванням ворога із пострілом ворог отримає зашкодження і якщо його здоровьє опустится до 0 то він знищеться та буде можливість що створится якись предмет через ObjcetController Рисунок 3.14

```
public void OnEnemyDeath(Enemy enemy)
{
    ItemData itemData = GetRandomItem(enemy.DropChance,enemy.IsBoss);
    if(itemData == null)
    {
        return;
    }
    int itemValue = InternalTools.GetRandomNumberInteger(itemData.itemValueMin, itemData.itemValueMax);
    Item item = new Item(itemData.prefab, _itemContainer, enemy.EnemyTransform.position, itemData.type, itemValue);

    item.ItemDestroyHandler += OnItemDestory;
    _items.Add(item);
}
```

Рисунок 3.14 – Створення предмету після зниження ворога

Предмет створюється на місці знищеного ворога і після взаємодії з об’єктом гравця також через івент передає свій тип до ObjectController який вже відповідно до свого типу взаємодіє з грою Рисунок 3.15.

```
}
private void OnItemDestory(Item item)
{
    switch (item.ItemType)
    {
        case ItemType.SmallXp:
        case ItemType.BigXp:
        case ItemType.MeduimXp:
            _playerController.AddXpToPlayer(item.ItemValue);
            break;
        case ItemType.Medecine:
            _playerController.RestoreHelathPlayer(item.ItemValue);
            break;
        case ItemType.FrozenBomb:
            _vfxController.SpawnFrozeBombVFX(item.SelfObject.transform.position);
            _enemyController.FrozeAllEnemy();
            break;
        case ItemType.Bomb:
            _bombBlow = new Blow(_vfxController.SpawnBombBlow(item.SelfObject.transform.position), _playerController.BombDamage, _gameplayData.DropChance.BombBlowChance);
            break;
        case ItemType.Chest:
            GameClient.Get<IUIManager>().DrawPopup<ChestPopup>(_skillsController.FillUpgradeList(item.ItemValue, true));
            break;
        case ItemType.RocketBox:
            if (!_playerController.OnGetRocketBox())
            {
                return;
            }
            break;
    }
    item.Dispose();
    _items.Remove(item);
}
```

Рисунок 3.15 – Дія предмету при зштовхуванням із гравцем

Також потрібно для кожного предмету створити свій префаб Рисунок 3.16

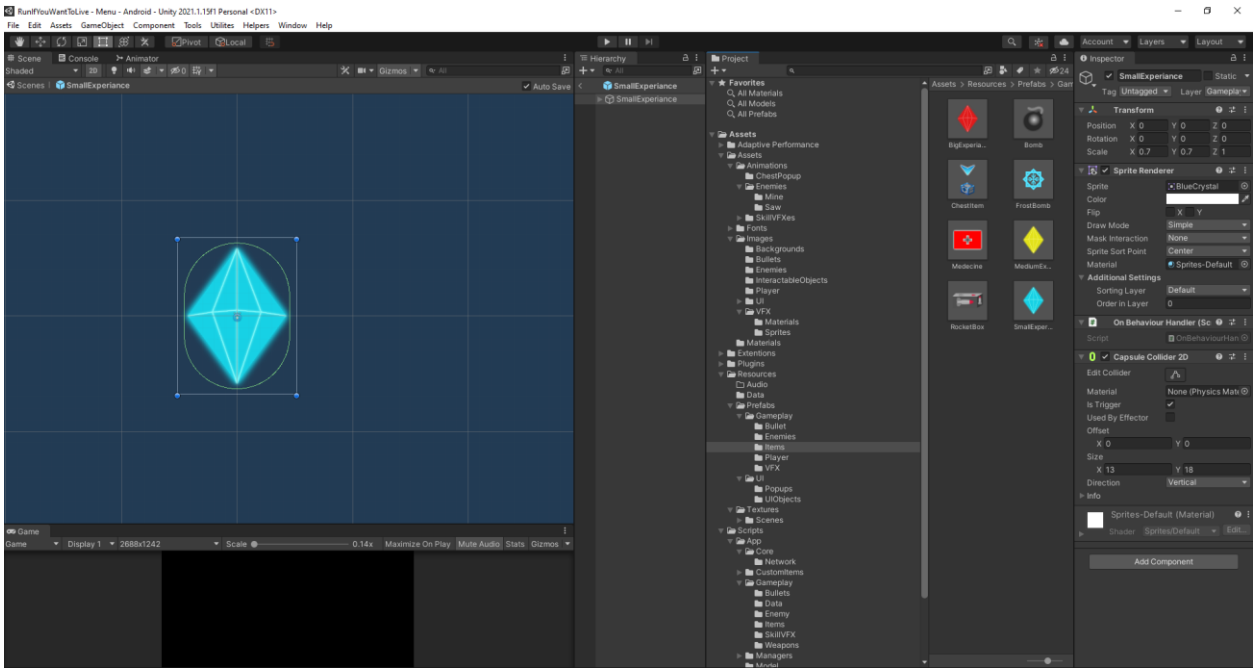

Рисунок 3.16 – Префаб кристалику досвіду

І для кожного предмету прописану інформацію у GameData Рисунок 3.17. Де прописано мінімальне та максимальне значення предмету, шлях до його префабу, тип та короткий опис.

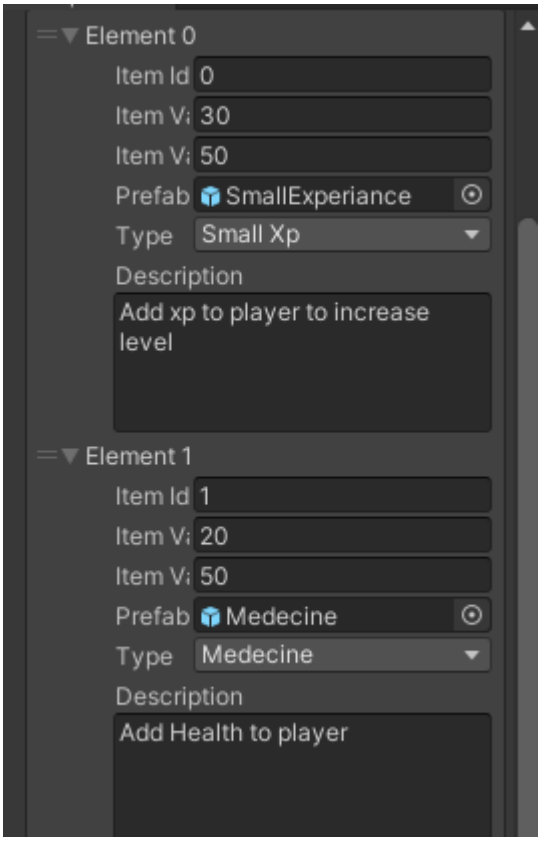

Рисунок 3.17 – Інформація про предмети

3.3.5 Система рівня та здібностей

При підбиранні кристалику досвіду до гравця передається його значення та підвищуються кількість досвіду. Якщо кількість досвіду привісила максимальну кількість то гравець отримає новий рівень та з’явиться вікно підвищення рівня яке наповниться випадковими здібностями із SkillController Рисунок 3.18.

```
public List<Skill> FillUpgradeList(int count = Constants.MAX_SKILLS_ON_LEVEL_UPGRADE, bool isForChest = false)
{
    List<Skill> tempList = new List<Skill>();
    if (isForChest)
    {
        for (int i = 0; i < AllAviableSkills.Count; i++)
        {
            if (AllAviableSkills[i].SkillUseType == SkillUseType.Additional)
            {
                tempList.Add(AllAviableSkills[i]);
            }
        }
    }
    else
    {
        tempList.AddRange(AllAviableSkills);
    }
    var upgradeSkills = new List<Skill>();
    for(int i = 0; i < count; i++)
    {
        if(tempList.Count <= 0)
        {
            break;
        }
        var skill = tempList[UnityEngine.Random.Range(0, tempList.Count)];
        tempList.Remove(skill);
        upgradeSkills.Add(skill);
    }
    if(upgradeSkills.Count <= 0)
    {
        return null;
    }
    return upgradeSkills;
}
```

Рисунок 3.18 – Отримання випадкових здібностей

І коли гравець обере здібність то цей контролер отримає тип цієї здібності та вже передасть її до PlayerController Наприклад гравець обере здібність підвищення кількості здоров’я то спочатку його обробить у SkillController Рисунок 3.19. Який передасть викличе метод у PlayerController Рисунок 3.20 і вже PlayerController передасть до об’єкту гравця що кількість здоров’я підвищена Рисунок 3.21.

```
}

private void SkillActionHandler(Skill skill)
{
    switch (skill.SkillType)
    {
        case SkillType.MaxHealthIncrease:
            _playerController.IncreaseMaxHealth(skill.Value);
            break;
        case SkillType.MovementSpeedIncrease:
            _playerController.IncreaseMovementSpeed(skill.Value);
            break;
        case SkillType.Shield:
            _vfxController.PlaySkillVFX(SkillType.Shield);
            _playerController.Player.SetupShieldSkill(_vfxController.GetSkillVFXByType(SkillType.Shield) as ShieldSkillVFX);
            RemoveSkill(SkillType.Shield);
            AddSkill(SkillType.ShieldRecoverTime);
            AddSkill(SkillType.ShieldHealthIncrease);
            break;
        case SkillType.Armor:
            _playerController.AddArmor((int)skill.Value);
            break;
        case SkillType.ShotAfterShot:
            _playerController.UpgradeWeaponShotAfterShot();
            break;
        case SkillType.DoubleShot:
            _playerController.UpgradeWeaponDoubleShot();
            break;
        case SkillType.BlowMina:
            RemoveSkill(SkillType.BlowMina);
            AddSkill(SkillType.BlowMinaRecover);
            AddSkill(SkillType.BlowMinaDamage);
            //Upgrade Start Mina
            break;
        case SkillType.BlowMinaRecover:
            //Upgrade Blow Mina Recover
            break;
        case SkillType.BlowMinaDamage:
            //Upgrade Blow Mina Damage
            break;
        case SkillType.ShieldRecoverTime:
            _playerController.Player.DecreaseShieldCooldownHandler(_vfxController.GetSkillVFXByType(SkillType.Shield) as ShieldSkillVFX, skill.Value);
            //Upgrade Shield Recover Time
            break;
        case SkillType.ShieldHealthIncrease:
            _playerController.Player.IncreaseShieldHealthHandler(_vfxController.GetSkillVFXByType(SkillType.Shield) as ShieldSkillVFX);
            //Upgrade ShieldHealthIncrease
            break;
        case SkillType.BulletSpeed:
            _playerController.IncreaseDefaultBulletSpeed<DefaultWeapon>((int)skill.Value);
            break;
        case SkillType.ShootDeleyDecrease:
            _playerController.DecraseDeafaultWeaponShootDeley<DefaultWeapon>(skill.Value);
            break;
        case SkillType.DamageIncrease:
            _playerController.IncreaseBulletDamage<DefaultWeapon>((int)skill.Value);
            break;
        case SkillType.Drone:
            _playerController.UpgradeDronesCount();
            if(skill.CurrentLevel <= 1)
            {
                AddSkill(SkillType.IncreaseDroneSpeed);
                AddSkill(SkillType.IncreaseDroneDamage);
            }
            //Upgrade DroneAroundPlayer
```

Рисунок 3.19 – Метод обробки отримання здібності

```
public void IncreaseMaxHealth(float amount)
{
    Player.IncreaseMaxHealth(amount);
}
```

Рисунок 3.20 – Метод обробки підвищення кількості здоров’я у PlayerController

```
public void IncreaseMaxHealth(float amount)
{
    _maxHealth += amount;
    _currentHealth += amount;
    if(_currentHealth > _maxHealth)
    {
        _currentHealth = _maxHealth;
    }
    HealthUpdateEvent?.Invoke(_currentHealth, _maxHealth);
}
```

Рисунок 3.21 метод підвищення рівня у PlayerObject

3.3.6 Завершення геймплею

Сам процес гри може йти нескінченно доки у гравець не програє. Для того щоб гравець програв була створена система фаз яку перемикає EnemyController Рисунок 3.22 коли у фазі закінчилися вороги. І якщо усі фази були закінченні то фази почнуться знову але вже у ворогів буде підвищена кількість здоров’я та шкоди.

```
private void SetNewPhase(int phaseId)
{
    if (CurrentPhaseIndex >= _gameplayData.gamePhases.Length)
    {
        IncreasePhaseIndex();
    }
    var miniBoss = _gameplayData.GetMiniBossByPhaseId(phaseId);
    if(miniBoss != null)
    {
        _enemies.Add(SpawnEnemy(miniBoss.enemyData, true));
    }
    CurrentPhaseIndex = phaseId;
    _currentPhase = _gameplayData.GetPhaseById(phaseId);
    _enemysInPhase = new List<Enumerators.EnemyType>();
    if (_currentPhase != null)
    {
        for(int i = 0; i < _currentPhase.enemyInPhase.Length; i++)
        {
            for(int j = 0; j < _currentPhase.EnemyCount[i]; j++)
            {
                _enemysInPhase.Add(_currentPhase.enemyInPhase[i]);
            }
        }
        _cooldownToSpawnEnemy = _currentPhase.spawnTime;
    }
}
public void IncreasePhaseIndex()
{
    CurrentPhaseIndex++;
    if (CurrentPhaseIndex >= _gameplayData.gamePhases.Length - 1)
    {
        _increaseEnemyParam += 0.25f;
        CurrentPhaseIndex = 0;
    }
    SetNewPhase(CurrentPhaseIndex);
}
```

Рисунок 3.22 – Перемикання фаз.

Самі фази потрібно наповнити у GameData Рисунок 3.23. У якій прописується Id фази, час через який буде створено нового ворога та типи ворогів і їх кількість.

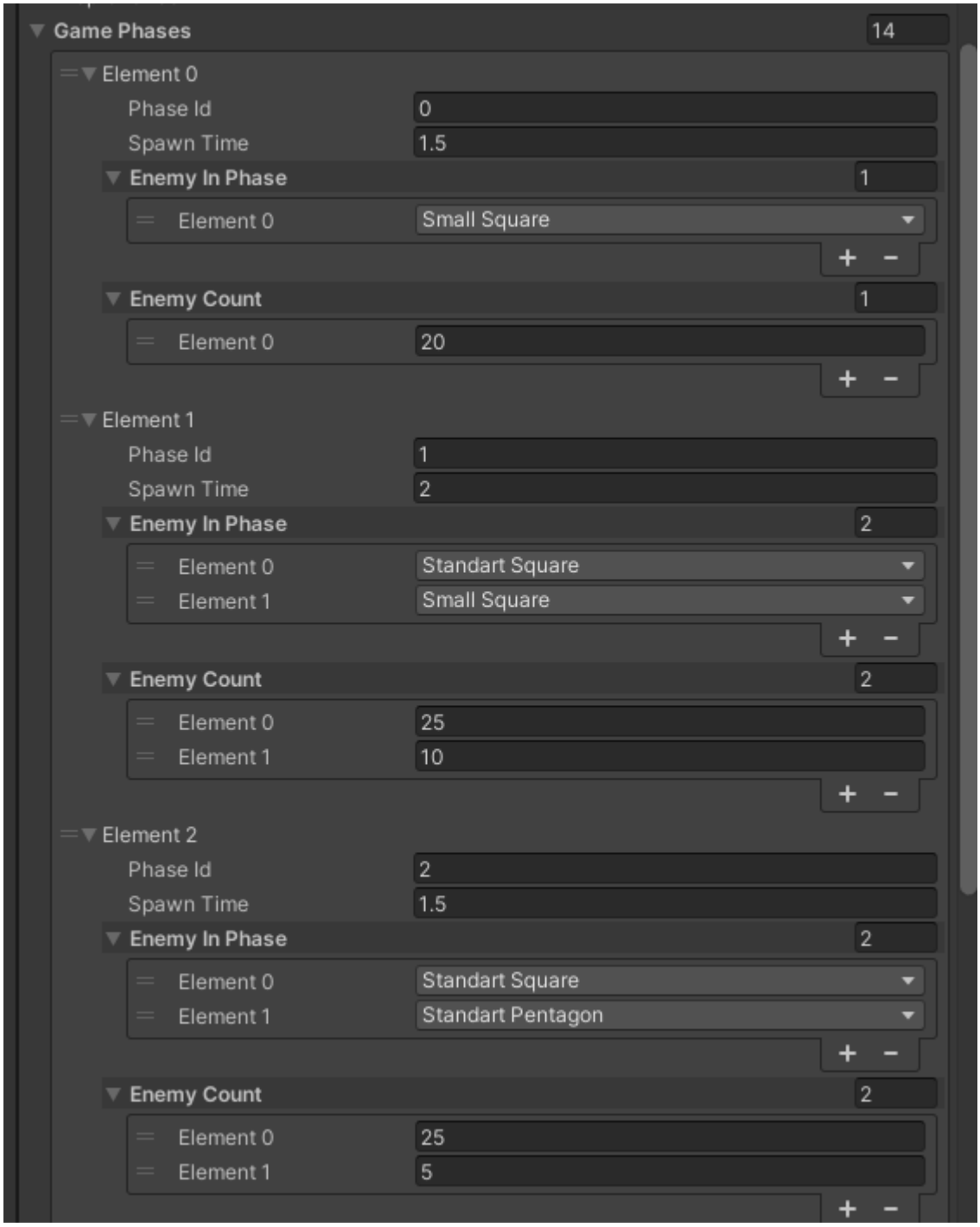

Рисунок 3.23 – Наповнення фаз

Після того як у гравця закінчиться здоров’я з’явиться вікно програшу у якому можна ввести ім’я та збережеться на пристрої Рисунок 3.24.

```
public void SaveCache(Enumerators.CacheDataType type)
{
    switch (type)
    {
        case Enumerators.CacheDataType.USER_LOCAL_DATA:
            {
                if (!File.Exists(_cacheDataPathes[type]))
                    File.Create(_cacheDataPathes[type]).Close();

                File.WriteAllText(_cacheDataPathes[type], JsonConvert.SerializeObject(CachedUserLocalData));
            }
            break;
        case Enumerators.CacheDataType.USER_RECORDS_DATA:
            {
                if (!File.Exists(_cacheDataPathes[type]))
                    File.Create(_cacheDataPathes[type]).Close();

                File.WriteAllText(_cacheDataPathes[type], JsonConvert.SerializeObject(UserLocalRecords));
            }
            break;
        default: break;
    }
}
public void AddRecord(RecordItem item)
{
    if(UserLocalRecords.Count >= Constants.MAX_RECORDS - 1)
    {
        RecordItem minimalRecord = UserLocalRecords[0];
        foreach (var record in UserLocalRecords)
        {
            //if(minimalRecord.Score == record.Score)
            //{
            //    if(record.EndTime < item.EndTime)
            //    {
            //        minimalRecord = record;
            //    }
            //}
            if(minimalRecord.Score <= record.Score)
            {
                minimalRecord = record;
                break;
            }
        }
        if(minimalRecord.Score > item.Score)
        {
            return;
        }
        UserLocalRecords.Remove(minimalRecord);
    }

    UserLocalRecords.Add(item);
    UserLocalRecords.Sort(delegate (RecordItem item1, RecordItem item2)
    {
        return item2.Score.CompareTo(item1.Score);
    });
}
```

Рисунок 3.24 – метод збереження рекордів на пристрої

3.4 Взаємодія з Сервером

Також рекорди зберігаються у базі даних яка розміщується на сервері. Спочатку потрібно орендувати хостинг та купити домен на якому і буде розміщуватись база даних та скрипти через які будуть іти запити для додавання інформації у базу даних або для отримання. Для оренди хостигу було обрано сервіс https://www.ukraine.com.ua/

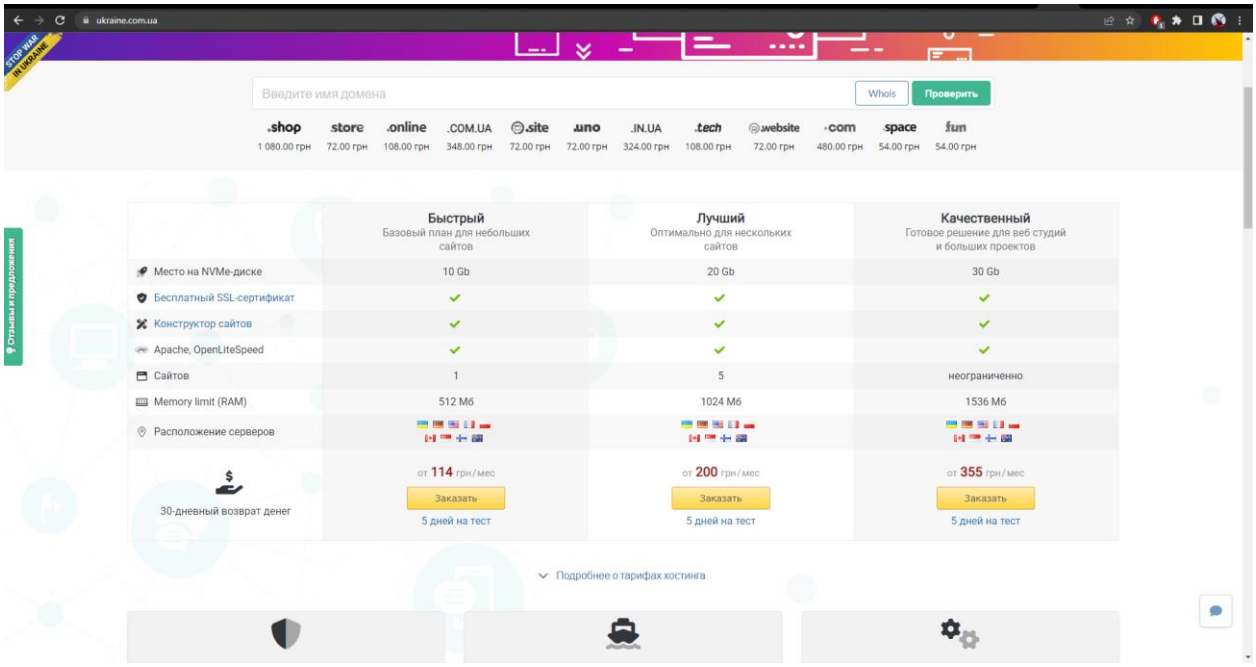

Рисунок 3.25 – інтерфейс сайту оренди хостингу

Після оренди Хостингу також потрібно зареєструвати домен. Після цього було отримано доступ до користувацької панелі Рисунок 3.26

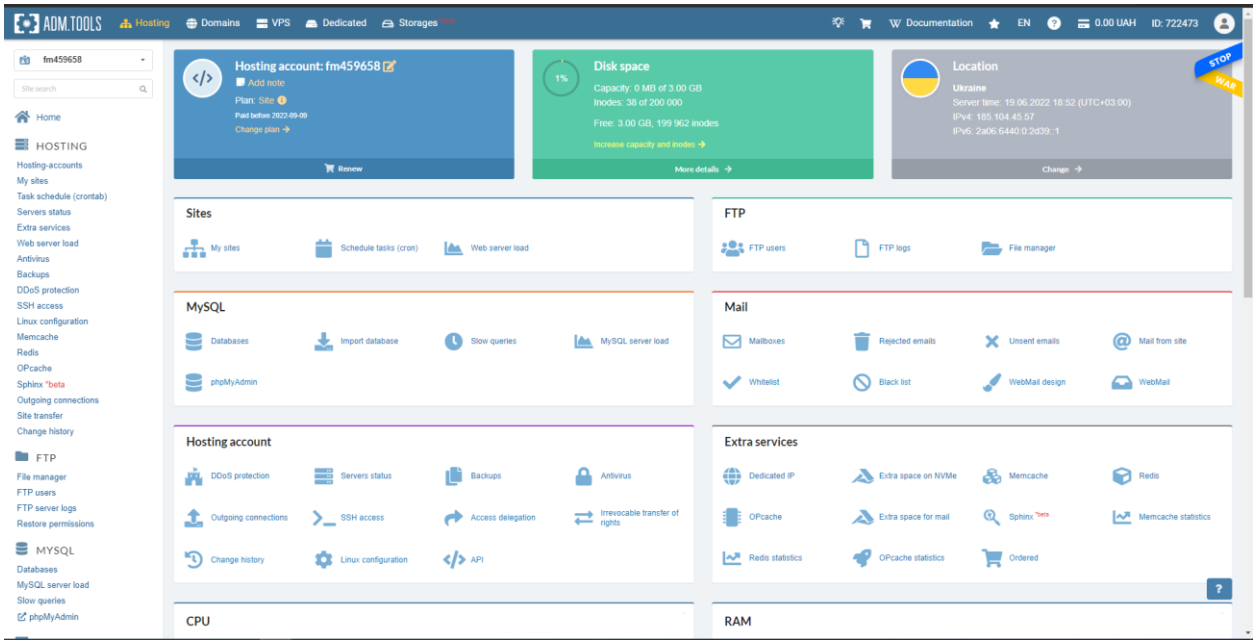

Рисунок 3.26 – інтерфейс сайту оренди хостингу

База даних створюється автоматично та через користувацьку панель можна перейти до PHPMyAdmin для налаштування та створення таблиць у базі данник Рисунок 3.27.

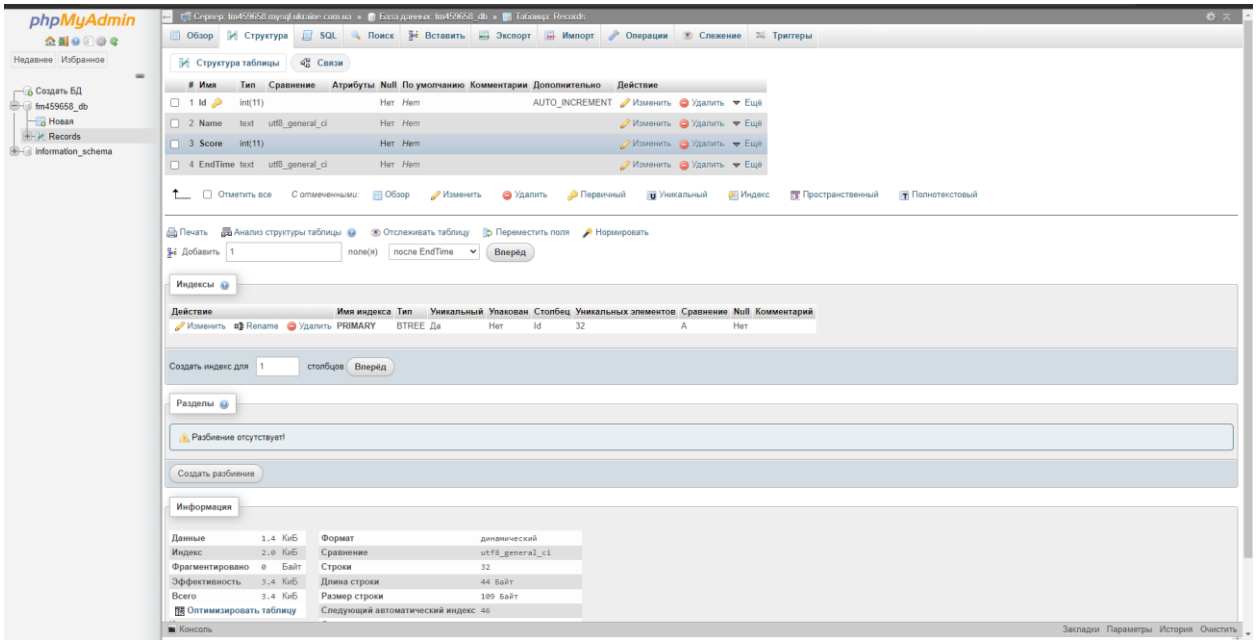

Рисунок 3.27 – Створена структура бази даних у PHPMyAdmin

Для запитів до БД використувється SQL а самі запити будуть у PHP скриптах. Для початку потрібно перейти до файлової системи хостингу Рисунок 3.28.

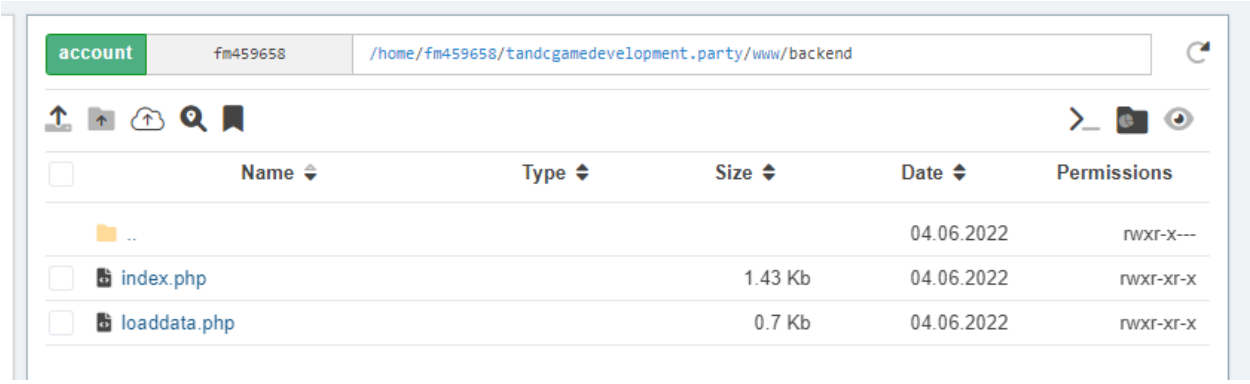

Рисунок 3.28 – Файловий менеджер хостингу

У самому менеджері необхідно створити директорію BackEnd у якій зберігається скрипт для внесення інформації у базу даних і скрипт через який сервер буде надсилати інформацію із бази даних до гри. У самій грі потрібно створити метод який буде надсилати запит до серверу та отримувати інформацію Рисунок 3.29

```
public void StartSend(string name, int score, string endTime)
{
    WWWForm form = new WWWForm();
    form.AddField("Name", name);
    form.AddField("Score", score);
    form.AddField("EndTime", endTime);
    MainApp.Instance.StartCoroutine(Send(form, Constants.DOMAINTOSEND, OnSendInfoSucces, OnSendInfoError));
}

public void StartGetData(Action<string> OnCompleteRequest = null, Action<string> OnErrorRequest = null)
{
    WWWForm form = new WWWForm();
    MainApp.Instance.StartCoroutine(Send(form, Constants.DOMAINTOGET, OnCompleteRequest, OnErrorRequest));
}

private void OnSendInfoSucces(string json)
{
    Debug.Log(json);
}
private void OnSendInfoError(string json)
{
    Debug.LogError(json);
}

private System.Collections.IEnumerator Send(WWWForm param, string url, Action<string> OnCompleteRequest = null, Action<string> OnErrorRequest = null)
{

    string path = _pathDomain + url;
    WWW www = new WWW(path, param);
    yield return www;
    if(www.error != null)
    {
        OnErrorRequest?.Invoke(www.error);
    }
    else
    {
        OnCompleteRequest?.Invoke(www.text);
    }

}

public bool IsHasInternetConnection()
{
    return !(UnityEngine.Application.internetReachability == UnityEngine.NetworkReachability.NotReachable);

}
```

Рисунок 3.30 – Надсилання запиту у NetworkManager

Висновки до розділу 3

Протягом розділу було створено та описано основні елементи архітектури такі як головний скрипт для зв’язку їх двигуном, скрипт для обробки зштовхування об’єктів і основні менеджери для взаємодії із сервером, грою, користувацьким інтерфейсом та файлами на пристрої.

Також було розроблено основні елементи ігрового процесу. Це створення гравця та керування ним. Було створено абстрактний клас від якого зручно додавати різні зброї які відрізняються між собою принципом дії. Був створений контролер через який створюються вороги. Сам клас ворогів який також є абстрактним для зручного додавань нових ворогів я яких різні дії та перемеміщення. Створено та описано систему здібностей які підсилюють гравця та систему предметів, які випадають після знищення ворога.

Було орендовано хостинг у якому створено БД. У Бд створено таблицю. Також створено 2 скрипти для взаємодії гри із сервером.

4 ТЕСТУВАННЯ ГРИ ТА МОНЕТИЗАЦІЯ

4.1 Функціональне тестування та огляд користувацького інтерфейсу

Функціональне тестування - це тестування з метою перевірки реалізованості функціональних вимог. Було проведено тести на пристрої Poco X4 з операційною системою Android.

Тест 1. Тестування відкриття сторінки рекордів.

Вхідні дані: На сторінки головного меню натиснути кнопку Leaderboard.

Очікуваний результат: Отримання рекордів які зберігаються на пристрої.

Отриманий результат: співпадає з очікуваним (Рисунок 4.1).

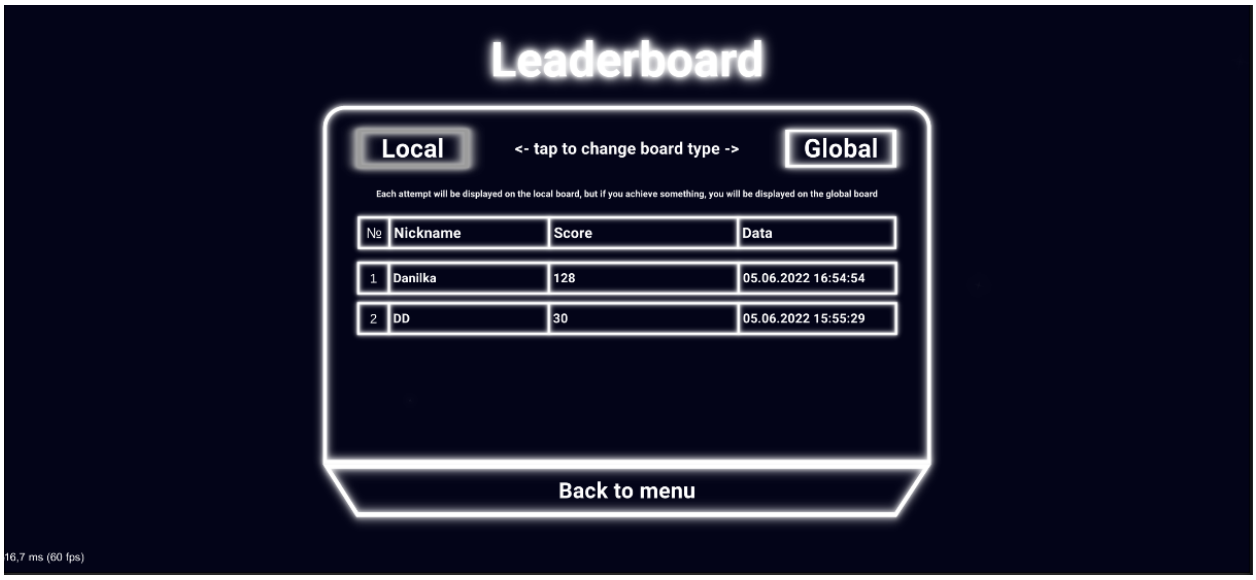

Рисунок 4.1 – Відкриття сторінки рекордів

Тест 2. Тестування пострілу гравця.

Вхідні дані: При наведенні гравця на ворога здійснюється постріл Рисунок 4.2.

Очікуваний результат: Після того як ворог зіштовхнувся із пострілом ворог знищується.

Отриманий результат: співпадає з очікуваним (Рисунок 4.3).

Рисунок 4.2 – здійсненнься пострілу

Рисунок 4.3 – знищення ворога

Тест 3. Тестування отримання здібності.

Вхідні дані: Об’єкт гравця зштовхується з кристалом досвіду.

Очікуваний результат: Підвищення досвіду гравця

Отриманий результат: співпадає з очікуваним (Рисунок 4.5).

Рисунок 4.4 – до отримання досвіду

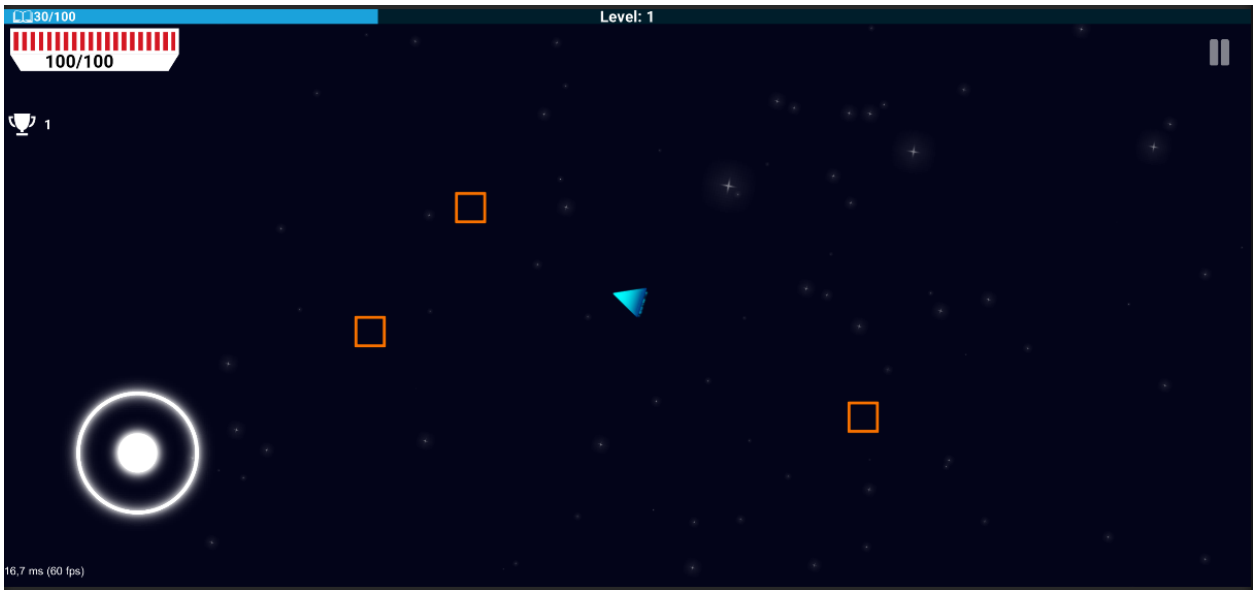

Рисунок 4.5 – отримання досвіду

Тест 4. Тестування отримання нового рівня.

Вхідні дані: Отримання необхідної кількості досвіду.

Очікуваний результат: Поява сторінки підвищення рівня де можна обрати необхідну здібність, та підвищення максимальної кількості досвіду.

Отриманий результат: співпадає з очікуваним (Рисунок 4.6).

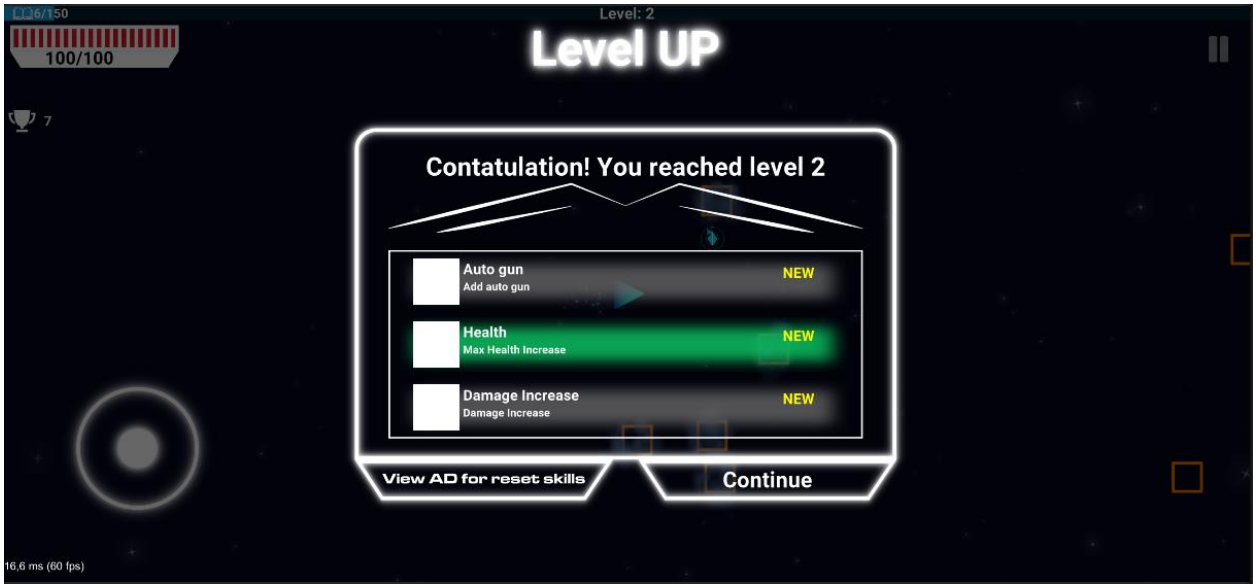

Рисунок 4.6 – Вікно підвищення рівня

Тест 5. Тестування отримання здібності для збільшення здоров'я.

Вхідні дані: Підвищення рівня і вибір здібності Рисунок 4.6.

Очікуваний результат: При виборі здібності кількість здоров'я гравця збільшиться на 25.

Отриманий результат: співпадає з очікуваним (Рисунок 4.7).

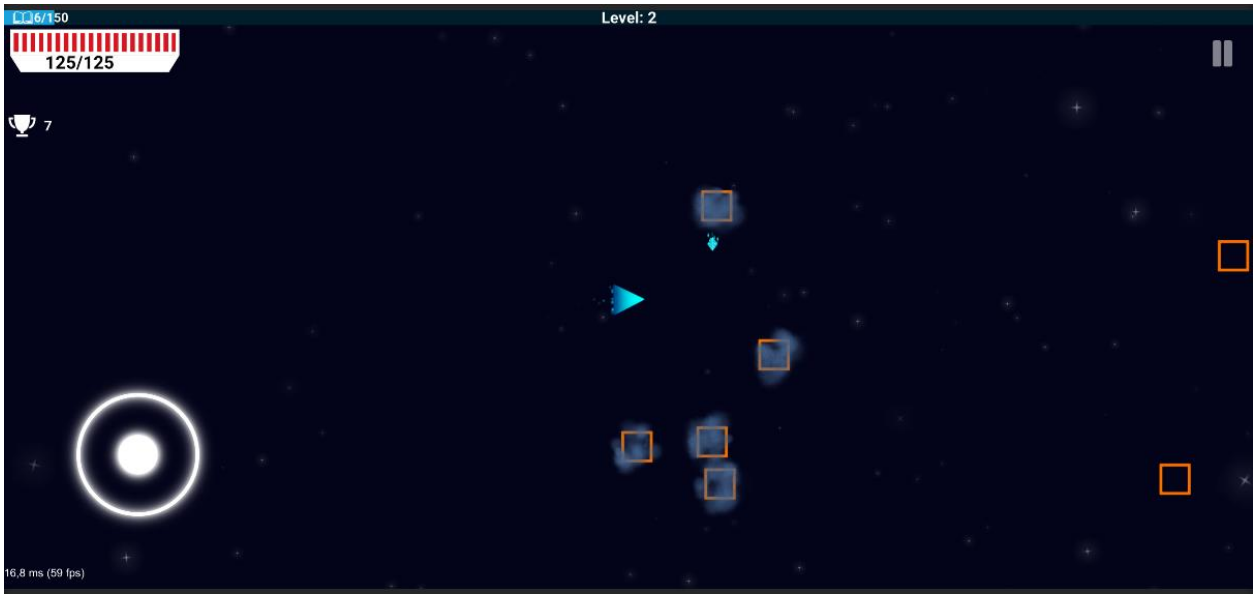

Рисунок 4.7 – Після збільшення здоров'я

Тест 6. Тестування появи міні босу.

Вхідні дані: Закінчення ворогів у фазі.

Очікуваний результат: При закінченні фази та при переключенні на іншу створення міні боса.

Отриманий результат: співпадає з очікуваним (Рисунок 4.8).

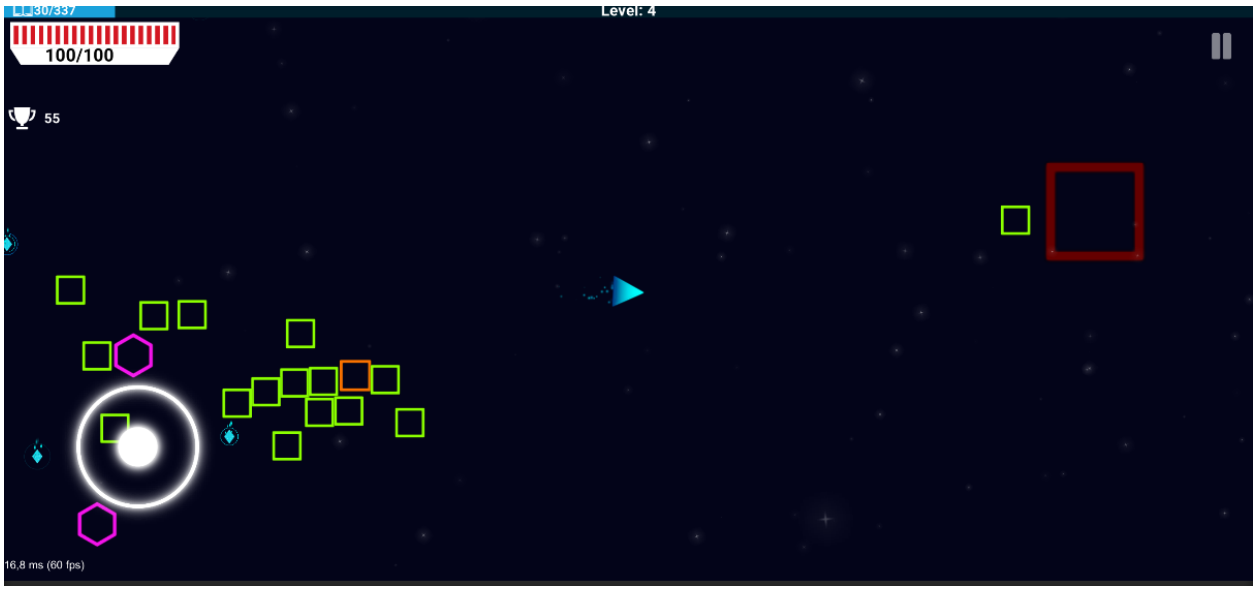

Рисунок 4.8 – Поява міні боса

Тест 7. Тестування програшу гравця та збереження данних на пристрої та у БД.

Вхідні дані: У гравця закінчується здоров’я .

Очікуваний результат: Поява вікна програшу (Рисунок 4.9) при введені ім’я та натискання кнопки продовжити збереження інформації на пристрої та у БД на сервері.

Отриманий результат: співпадає з очікуваним (Рисунок 4.10-4.11).

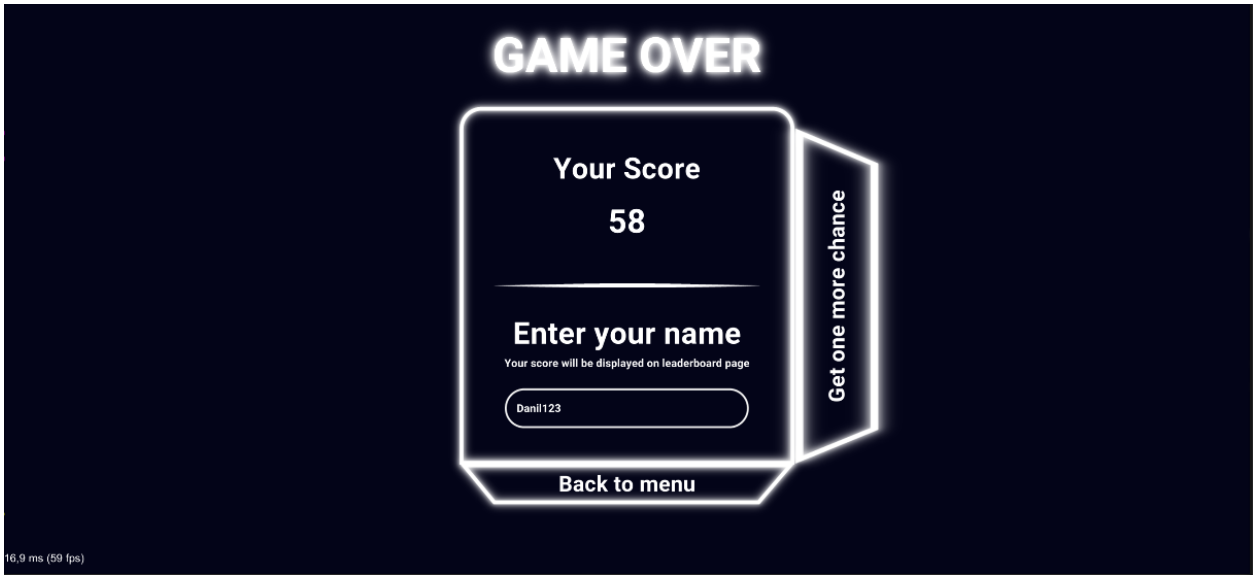

Рисунок 4.9 – Вікно програшу із введеним ім’ям

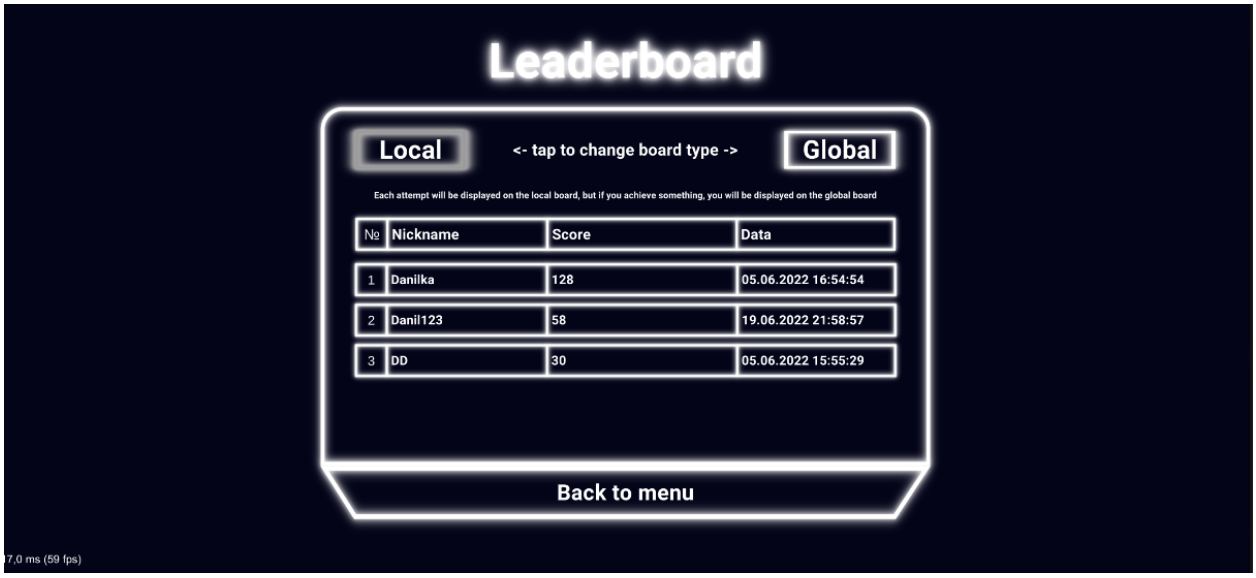

Рисунок 4.10 - Сторінка локальних рекордів із внесеним рекордом

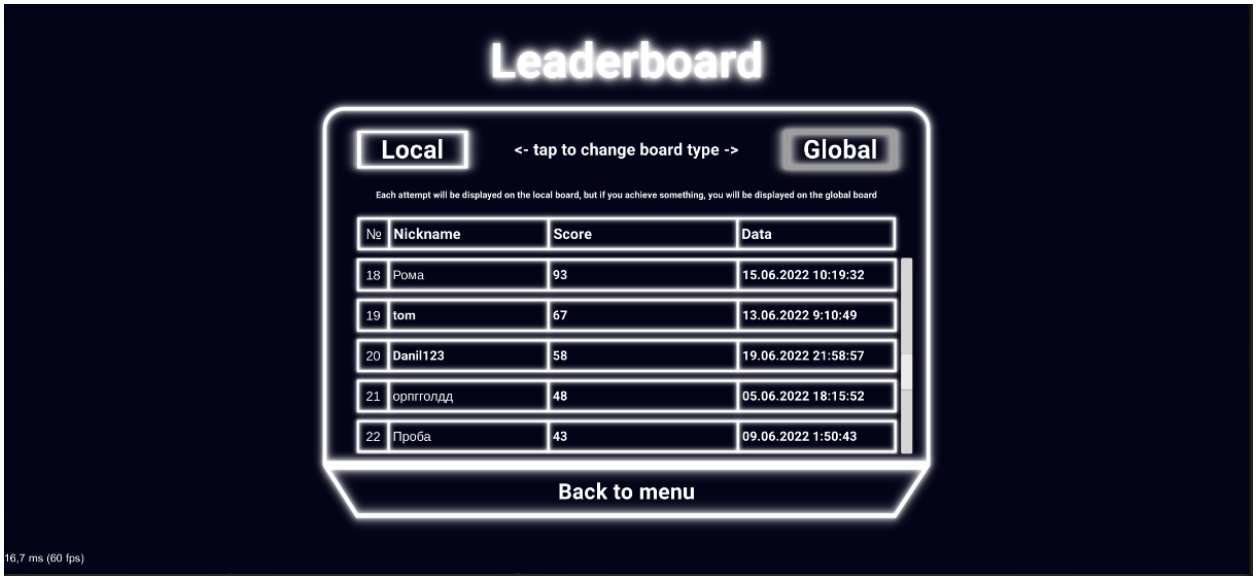

Рисунок 4.11 - Сторінка глобальних рекордів із внесеним рекордом

4.2 Монетизація гри

Основна монитизація гри їде з перегляду реклами користувачами для цього є декілька різних сервісів UnityAds, AdMod від Google, InMobi тощо. Для кожного потрібно завантажити та встановити спеціальний пагін, але для зручності буде використано лише UnityAds. Для чого спочатку потрібно зареєструвати за стосунок та отримати його Id Рисунок 4.12.

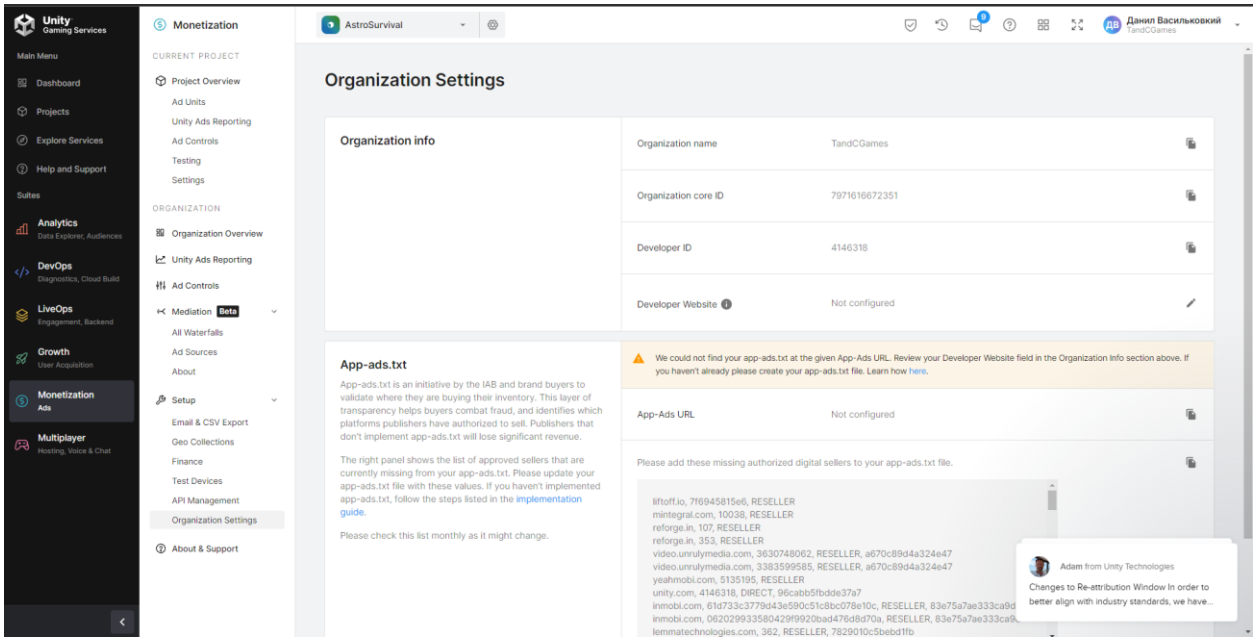

Рисунок 4.12 – Зареєстрований за стосунок у UnityDashboard

Наступним кроком буде встановлення спеціального плагіну для реклами UnityAds та через айди увійти у аккаунт Unity Рисунок 4.13

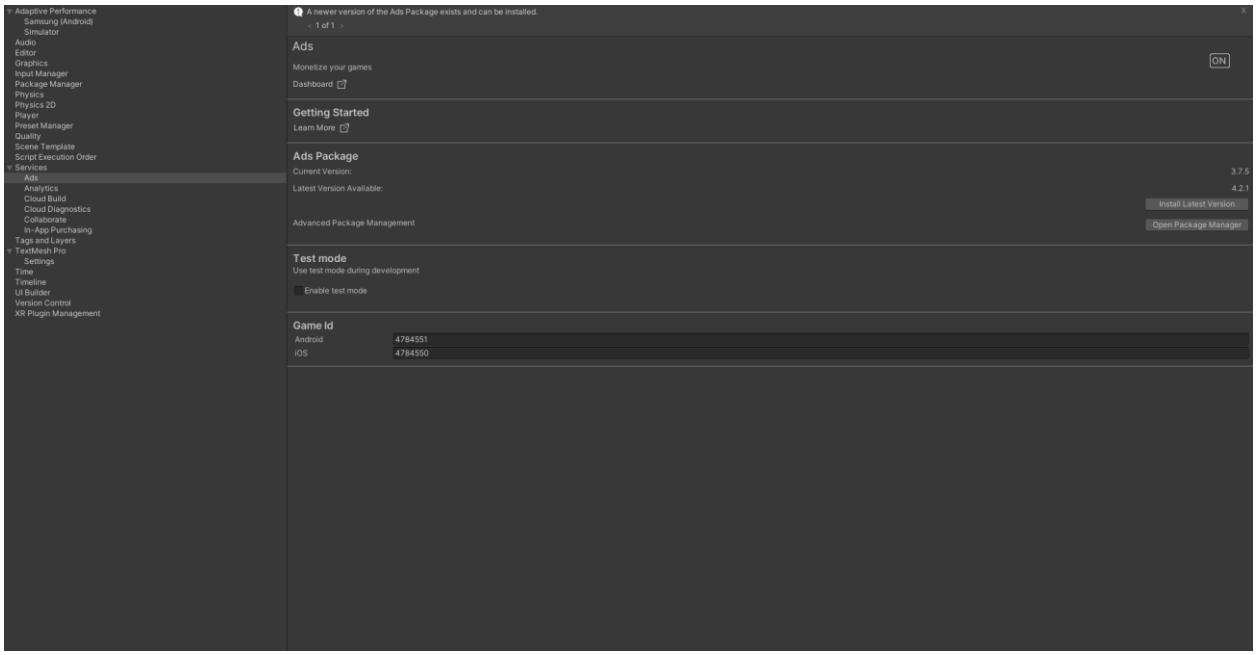

Рисунок 4.13 – вікно налаштування реклами у UnityEditor.

Для відображення самої реклами потрібно створити спеціальний AdsManager (Додаток Л). Цей менеджер створює відео та по івентам відповідає чи була показана реклама щоб гравець отримав винагороду чи ні і гравець нічого не отримав. Для перегляду потрібно звернутись до методу цього менеджеру Рисунок 4.14

```
private void OnRecieveButtonClickHandler()
{
    if (_isRecieveOneTime)
    {
        return;
    }
    _advarismetnManager.ShowAdsVideo(OnRecieveComplete, OnRecieveFailed);


}
private void OnRecieveComplete()
{
    _gameplayManager.GetController<PlayerController>().RecievePlayer();
    _isRecieveOneTime = false;
    _recieveButton.interactable = false;
    Hide();
    _gameplayManager.PauseGame(false);
    _uiManager.SetPage<GamePage>();
}
private void OnRecieveFailed()
{
    Debug.LogError("Failed");
}
```

Рисунок 4.14 – Приклад звертання для показу реклами

Одна із можливостей перегляду реклами за винагороду це при підвищенні рівня можна оновити здібності на нові якщо вони не подобаються гравцю Рисунок 4.15 – 4.17.

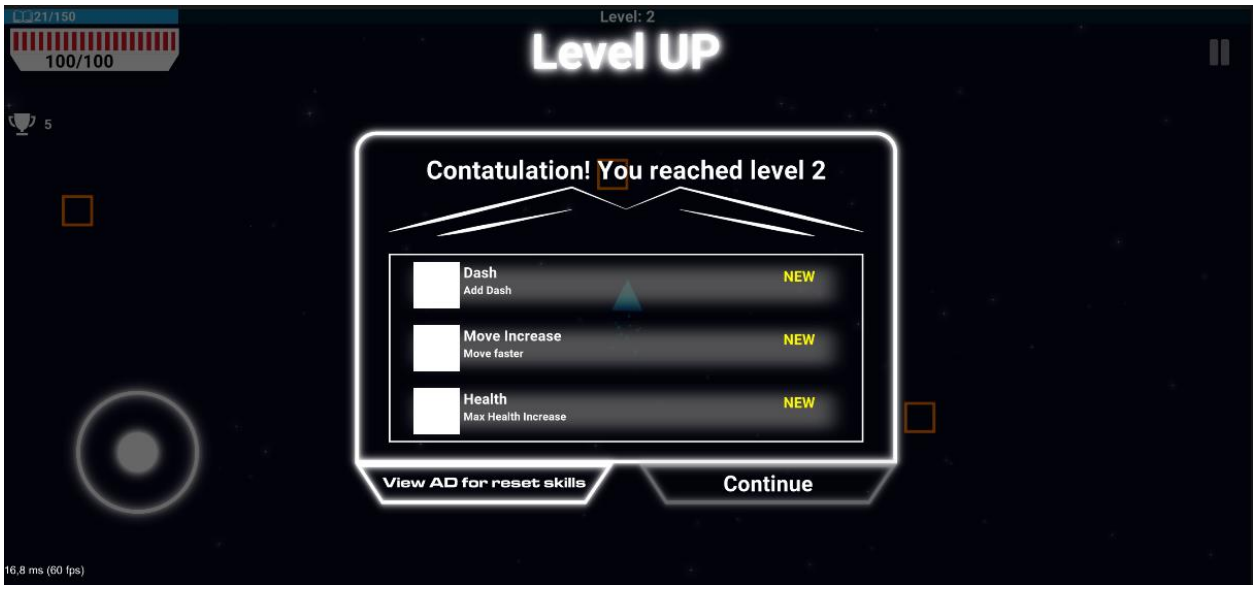

Рисунок 4.15 – До перегляду реклами

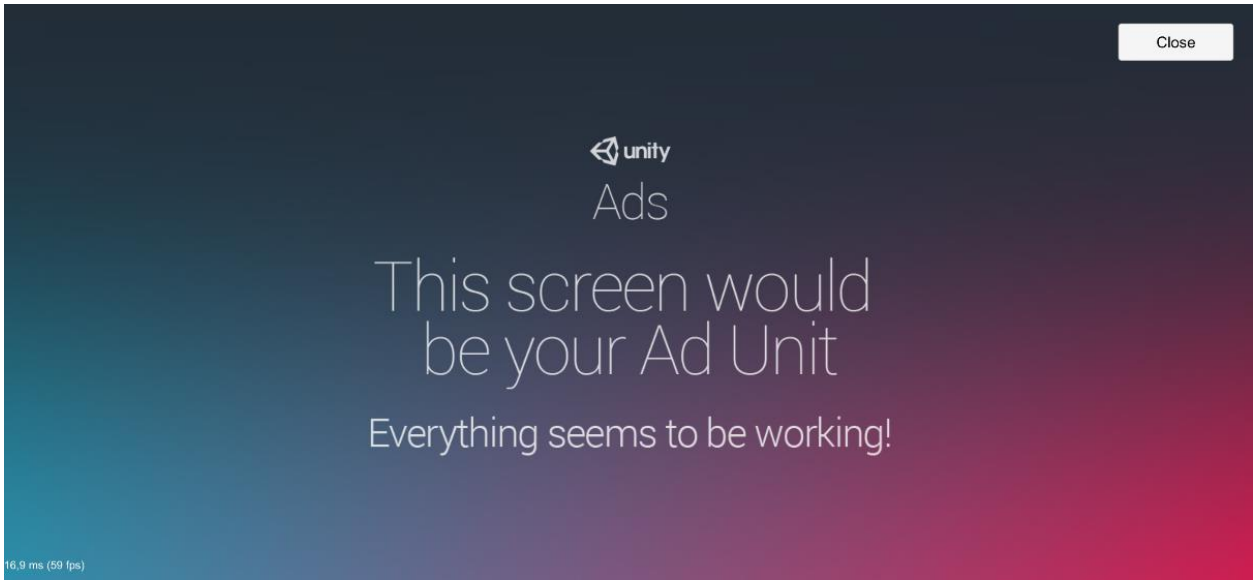

Рисунок 4.16 – Реклама

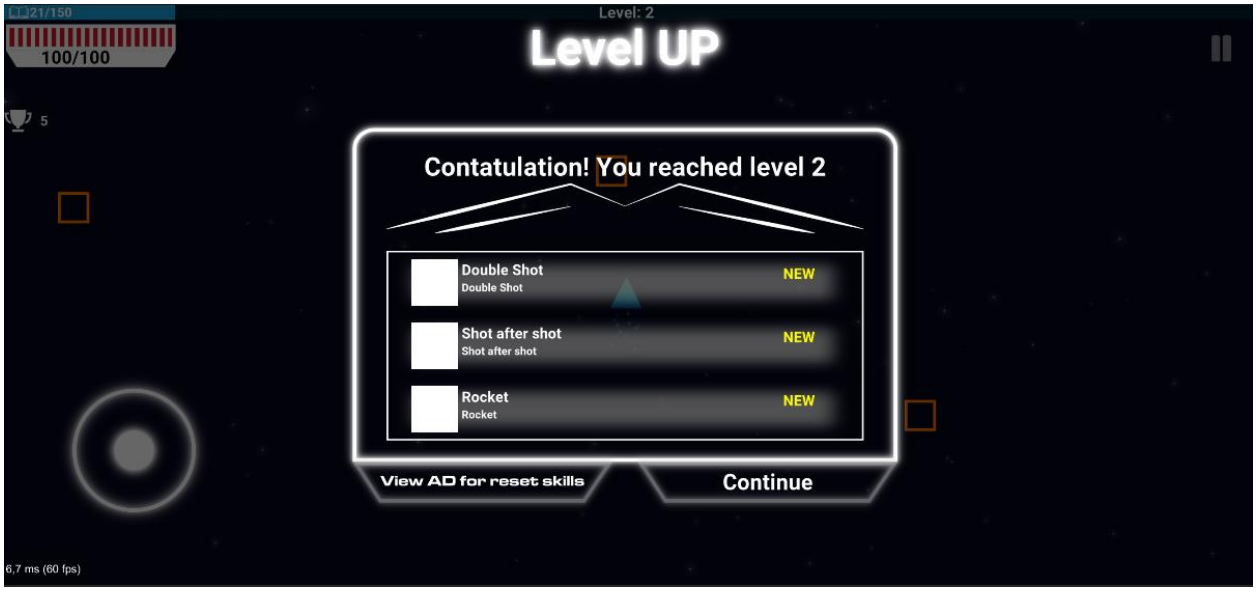

Рисунок 4.17 - Після перегляду реклами

Також ще одна можливість перегляду це після програшу. На сторінці програшу можна натиснути кнопку ViewAD та покажеться реклама після якої у гравця відновиться 100 одиниць здоров’я та усі вороги заморозяться і у гравця буде ще один шанс.

Unity ads збирає відомість про кількість переглядів реклами кожен день, у якому можна побачити кількість запитів до перегляду реклами, кількість переглядів яка зарахувалась та середню виплату за 1000 переглядів Рисунок 4.17.

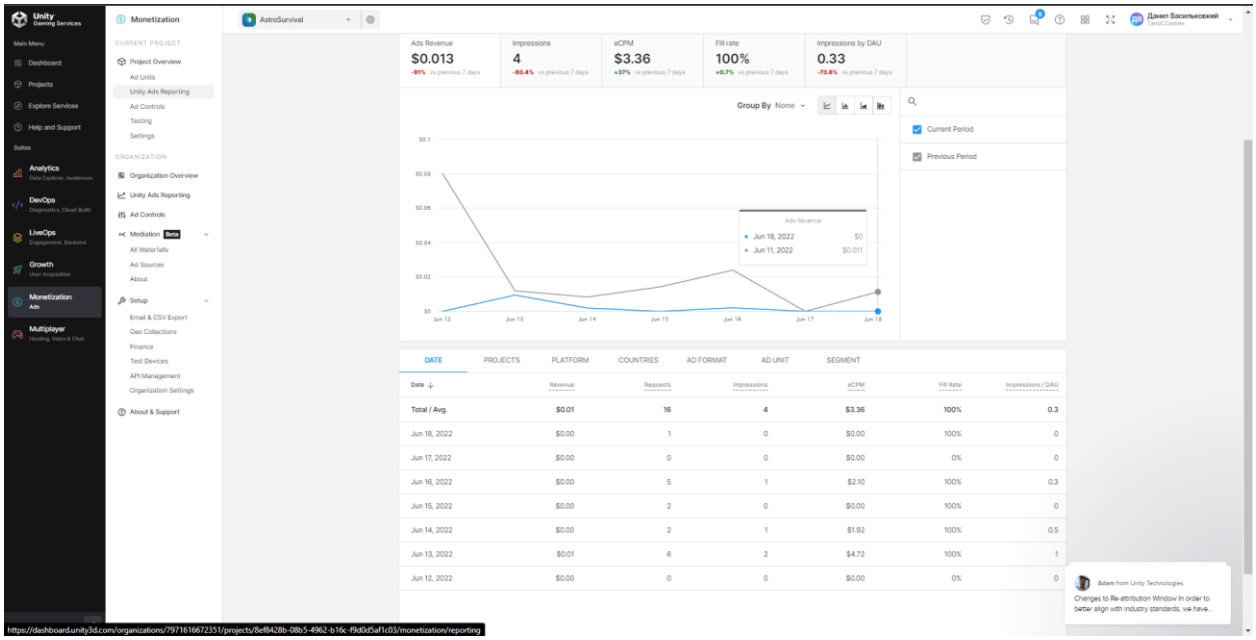

Рисунок 4.18 – Статистика у Unity Dashboard

4.3 Подальший розвиток гри

Також було проведено тести у яких прийняли участь 25 гравців з різним ігровим досвідом. Після цього тесту було оброблено та проаналізовано відгуки у результаті чого було сформовано декілька тез для покращення ігрового процесу.

- Прискорення ігрового процесу.
- Додавання більшої кількості ворогів.
- Зміна інкременту досвіду після підвищення рівня, тому що після 6 рівня дуже складно отримати новий рівень.
- Додати валюту до гри яка також падає з ворогів.
- Додати магазин у якому можна купити за валюту візуальні зміни для гравця.
- Додати звуки та музику.
- Додати сторінку керівництва де гравець який не має досвіду у іграх може дізнатись про основні механіки гри.
- Зміна дизайну інтерфейсу
- Додавання нових здібностей

- Можливість обрати одну здібність перед початком гри
- Можливість переглянути рекламу та отримати валюти гри.
- Додати анімації телепортації гравця, анімацію підбору предмету
- Додати анімацію для готовності пострілу
- Додати напрямок дії ракети та лазеру
- Додати радіус для по добору предметів
- Додати шифр даних
- Змінити фон ігрового процесу
- Додати графіку для границь.
- Розробити великого боса з унікальними механіками та винагородами.
- Можливість придбати прибирання реклами з гри а усі винагороди з реклами отримувати без її перегляду

Висновки до розділу 4

Протягом розділу було проведено функціональне тестування основних ігрових механік таких як постріл, знищення ворога та випадання з нього предметів. Було протестовано появу мінібоса та програшу гравця і внесення рекорду до бази даних та збереження на пристрої.

Було під'єднано можливість отримання прибутку з гри за допомогою UnityAds через перегляд реклами. Описана реєстрація проекту та створений менеджер який показує рекламу для оновлення здібностей або після програшу.

Також було проведені ігрові тестування серед аудиторії яка має різний ігровий досвід. З цього тестування було виявлено основні проблеми гри та оброблено тези для подальношого покращення проекту перед випуском його до Google Play.

ВИСНОВКИ

В процесі виконання дипломної роботи було розроблено ігровий 2D застосунок для мобільних платформ у жанрі який поєднує у собі жанри Adventure та Action. Проаналізовано основні технології які можуть використатись для розробки застосунку та було обрано платформу Unity із мовою C#. Проведено аналіз декількох інших ігрових застосунків з яких було сформовано основну ідею та мехінки для гри. Описано основні елементи такі як вороги, предмети та здібності гравця які мають бути реалізовані у грі. Також описано основні можливості гравця.

Створено діаграми классів які використовуються у проектуванні архітектури гри та ігрового процесу. Проаналізовано програмний підхід який зазвичай використовують для створення ігрових застосунків на платформі Unity який ще називають компонентим, і було прийнято і обґрунтоване рішення використання іншого підходу розробки із використанням чистого C# та мінімізації використання можливостей двигуна. Було створено та описано діаграму взаємодії, діаграму переходу стану та діаграму діяльності. Також створено та описано користувацький інтерфейс та переміщення між ним.

Програмно описано основні скрипти Архітектури та менеджери які використовуються у грі. Описано керування гравця та створення основних систем таких як зброя, предмети, вороги та здібності. Описано взаємодію із серверною частиною та можливість реєстрації і аренди особистого хостингу та домену на якому розміщено БД та декілька PHP скриптів.

Протестоно основні геймплейні механіки та описано можливість отримання прибутку з гри за допомогою перегляду реклами та роботу із плагіном UnityAds та обробка інформації через Unity Dashboard. Було проведено тестування у вибраній аудиторії після якого було сформовано основні тези для покращення гри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Либерти Д. Язык программирования C# // Программирование на C#. Санкт-Петербург. 2003. 688 с.

2. Албахари Дж., Албахари Б. C# 6.0. Справочник. Полное описание языка. М.: «Вильямс», 2018. 1040 с.

3. Шилдт Г. C# 4.0: полное руководство. М.: «Вильямс», 2010. 1056с.

4. Гарднер М. Математические головоломки и развлечения. Москва: МИР, 1999. 447с.

5. Хокинг Дж. Unity — в действии. Мультиплатформенная разработка на C#. СПб : Питер, 2016. 336 с.

6. Торн А. Искусство создания сценариев в Unity . -СПб : ДМК, 2016. 362с.

7. Документація по Unity URL: https://docs.unity3d.com/ru/current/Manual/index.html (дата звернення: 10.04.2022).

8. Developing 2D Games with Unity: Independent Game Programming with C# Jared HalpernNew York, NY, USA.

9. Vampire survival – URL: https://store.steampowered.com/app/1794680/Vampire_Survivors/ (дата звернення 12.03.2022).

10. Паттерни проектування - URL: https://refactoring.guru/ru/design-patterns/catalog (дата звернення 15.03.2022).

11. Asteroids - URL: https://www.crazygames.ru/igra/asteroids-atari (дата звернення 13.03.2022).

12. Годот лучше, чем Unity для разработки игры? Каковы плюсы и минусы? URL: https://otvetus.com/godot-luchshe-chem-unity-dlya-razrabotki-igri-kakovi-plyusi-i-minusi-31412 (дата звернення 12.03.2022). – Текст: електронний.

13. Движок Unity – особенности, преимущества и недостатки. URL: https://cubiq.ru/dvizhok-unity/ (дата звернення 12.03.2022). – Текст: електронний.

14. Game design basics: How to start creating video games. URL: https://www.cgspectrum.com/blog/game-design-basics-how-to-start-building-video-games (дата звернення 10.04.2022). – Текст: електронний.

ДОДАТОК А

Серцевий код застосунку

MainApp.cs

```
using System;
using UnityEngine;
using UnityEngine.SceneManagement;
namespace TandC.RunIfYouWantToLive{
   public class MainApp : MonoBehaviour{
      public delegate void MainAppDelegate(object param);
      public event MainAppDelegate OnLevelWasLoadedEvent;
      public event Action LateUpdateEvent;
      public event Action FixedUpdateEvent;
      private static MainApp _Instance;
      public static MainApp Instance
      {
         get { return _Instance; }
         private set { _Instance = value; }
      }
      float deltaTime = 0.0f;
      void OnGUI()
      {
         int w = Screen.width, h = Screen.height;
         GUIStyle style = new GUIStyle();
         int heightRect = h * 2 / 100;
         Rect rect = new Rect(0, h - heightRect * 2, w, heightRect);
         style.alignment = TextAnchor.UpperLeft;
         style.fontSize = h * 2 / 100;
         style.normal.textColor = Color.white;
         float msec = deltaTime * 1000.0f;
         float fps = 1.0f / deltaTime;
         string text = string.Format("{0:0.0} ms ({1:0.} fps)", msec, fps);
         GUI.Label(rect, text, style);
      }
      private void Awake(){
         if(Instance != null){
            Destroy(gameObject);
            return;
         }
         Instance = this;
         Application.targetFrameRate = 60;
         DontDestroyOnLoad(gameObject);
      }
      private void Start(){
         if (Instance == this)
         {
            GameClient.Instance.InitServices();
```

```
GameClient.Get<IAppStateManager>().ChangeAppState(Common.Enumerators.AppState.APP_INIT_L
OADING);
            SceneManager.sceneLoaded += SceneManager_sceneLoaded;

        }
    }

    private void Update()
    {
        deltaTime += (Time.unscaledDeltaTime - deltaTime) * 0.1f;
        if (Instance == this)
            GameClient.Instance.Update();
    }

    private void LateUpdate()
    {
        if (Instance == this)
        {
            if (LateUpdateEvent != null)
                LateUpdateEvent();
        }
    }

    private void FixedUpdate()
    {
        if (Instance == this)
        {
            if (FixedUpdateEvent != null)
                FixedUpdateEvent();
        }
    }

    private void OnDestroy()
    {
        if (Instance == this)
            GameClient.Instance.Dispose();
    }
    private void OnApplicationQuit()
    {
        if (Instance == this)
            GameClient.Instance.Dispose();
    }


    private void SceneManager_sceneLoaded(Scene arg0, LoadSceneMode arg1)
    {
        if (Instance == this)
        {
            if (OnLevelWasLoadedEvent != null)
                OnLevelWasLoadedEvent(arg0.buildIndex);
         }
     }
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Мобільна 2D гра у жанрі Adventure на платформі Unity

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

«ОХОРОНИ ПРАЦІ В ІТ-ІНДУСТРІЇ»

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ. – 409.21810906

Студент _____ Д.Д. Васильковський
підпис ініціали, прізвище
«__» _____ 2022 р.

Консультант канд. техн. наук, доцент(б. в. з.) _____ А. О. Алексєєва
підпис ініціали, прізвище
«__» _____ 2022 р.

Миколаїв 2022

ЗМІСТ

ВСТУП

В сучасних умовах формування інформаційного суспільства в Україні особливо важливим завданням є інформатизація усіх сфер господарської діяльності та формування нової генерації висококваліфікованих та професійно-компетентних фахівців у сфері ІТ-індустрії. Сучасний досвід запровадження інформаційної політики в Україні свідчить про певні потреби у фахівцях не тільки на діючих підприємствах, але і в інноваційних установах та організаціях.

На перший погляд, працівники сфери інформаційних технологій не схильні до ніяких виробничих ризиків, але сучасний розвиток науки та техніки привносить принципові нововведення у сфери матеріального виробництва. Разом з тим, захопившись вдосконаленням засобів праці їх творці залишили поза увагою проблеми людини в рамках своєрідної технічної та комп'ютерної революції. З широким впровадженням автоматизації та комп'ютеризації виникла потреба врахування психологічних можливостей людини, таких як швидкість реакції, особливості пам'яті та уваги, емоційний стан та ін. Поява операторської діяльності призвела до суттєвих змін у фаховій структурі праці. Зменшились фізична важкість праці, ризик виробничого травматизму, однак разом з тим, на людину що працює посилюється вплив нових, раніше не відомих чи мало вивчених різних несприятливих виробничих та психоемоційних чинників.

Праця людини, що відбувається в умовах надмірного нервово-емоційного навантаження, довготривалих статичних перевантажень, обмеженої рухової активності призводить до різноманітних відхилень у стані здоров’я, зокрема до перевтоми, зниження фізичної та розумової працездатності, неврозів, захворювань серцево-судинної системи тощо.

Отже потрібно виявити спосіб покращити працю фахівця інформаційних технологій.

1 Значення охорони праці у роботі фахівців з інформаційних технологій

Комп'ютери вже давно увійшли в повсякденне життя, а норми роботи з ними вже давно вкоренилися в зарубіжних IT-компаніях, проте на території України цей процес все ще триває, хоч і підходить до завершальної стадії. Саме тому розгляд даного питання є актуальним для такого міста, як Маріуполь, так як у столиці та деяких великих обласних центрах це питання практично вирішене. На перший погляд, робота за комп'ютером здається безпечною, але саме легковажність до неї може призвести до певних проблем у здоров'ї людини [1]. Професія програміста та інших фахівців IT-технологій пов'язана з колосальним розумовим напруженням. Розробники – настільки захоплені люди, що навіть відволікаючись від роботи над проектом, продовжують думати про роботу. Нерідко відпочинком вони вважають паралельну заміну основної діяльності, наприклад, читання профільної літератури, верстку сайтів, вивчення нових мов програмування. Однак мозок не може до безкінечності приймати виключно корисну інформацію, яку розробник прагне направляти в русло особистісного та професійного зростання [2]. Адже мозок людини не машина: він не може нескінченно зберігати і переробляти дані практично не втрачаючи продуктивності. Людина програє ПЕОМ за багатьма показниками (див. таблиці 1.1-1.2), тому для підтримки працездатності вимагає певних умов.

Таблиця 1.1 – Порівняння якісних характеристик ПЕОМ та властивостей людини

Характеристика ПЕОМ	**Властивості людини**
Нездатна до аналізу подій, розподілених у часі і просторі, а також розпізнавання	Здатний розпізнавати і використовувати інформацію при загальному аналізі обстановки для поділу явищ в часі і в виборі важливих відомостей
Можливість задавати програму вибір оптимального рішення, необхідну точність і швидкість обчислень	Обчислювальні можливості людини обмежені, виконуються повільно, з малою точністю, без вибору оптимального рішення
Навчання швидке, але обмежена використовуваною програмою. Характеристики згодом практично не змінюються	Здатність до вирішення логічних завдань визначається віком і навчанням. Потрібно періодичне тренування працездатності, яка знижується з віком
Байдужість до результатів роботи	Зацікавленість в результатах роботи, схильність до емоційних впливів, необхідність зміни умов для забезпечення життєдіяльності.

Таблиця 1.2 – Порівняльні дані окремих характеристик людини та ПЕОМ

Показники	Характеристика	
	Людини	ПЕОМ
Імовірність помилкових дій, відносні одиниці	10^{-2}	10-6
Час виконання операцій	0,4	10^{-2} -10^{-3}
Час звернення до пам’яті	10^{-2} -10^{2}	10^{-8} -10^{-2}
Час оперативної пам’яті	3 секунди	Тисячі годин

Тому зараз багато IT-компаній обладнують свої офіси кімнатами відпочинку, які забезпечують психофізіологічне розвантаження працівників. Окремим робочим столом з ноутбуком вже давно нікого не здивуєш. Тому,

бажаючи підвищити продуктивність працівників, міжнародні компанії змагаються, перетворюючи нудні одноманітні офіси в креативні простори, де нові ідеї народжуються без титанічних зусиль. Наприклад, робочі місця співробітників компанії Google в Цюріху нагадують гігантські вулики, а офіс шведського інтернет-провайдера Bahnhof розташувався в бомбосховищі часів холодної війни і походить на підземний притулок землян після глобальної катастрофи. А щоб співробітників не тягнуло додому, роботодавці створюють і можливість відпочивати, не відходячи від робочого місця, обладнавши басейни, ігрові кімнати та спортзали [3].

Адже можна цілий день просидіти біля монітора, а у вечорі відчути страшну втому, як від важкої фізичної праці. Однак це почуття помилкове і боротися з ним допоможе спорт. Активний відпочинок відмінно бадьорить, розганяє кров, додає сил. Чимало програмістів відчувають потребу в активному відпочинку на підсвідомому рівні, вибираючи спорт як хобі для проведення вільного часу.

При цьому не варто забувати, що умови праці програмістів також характеризуються можливістю впливу на них наступних небезпечних і шкідливих виробничих факторів: шуму; тепловиділень; завдати шкоди організму можуть не тільки високі, але і низькі температури; іонізуючі та неіонізуючі випромінювання: рентгенівське, інфрачервоне, електромагнітне випромінювання ВЧ і СВЧ діапазону; статичної електрики; недостатнє штучне та природнє освітлення; візуальні фактори: яскравість, контрастність, мерехтіння зображення, відблиски тощо [4].

За таких умов зростає роль та значення охорони праці, як системи правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

2 Інформаційна культура як важлива складова в управлінні охороною праці

Для підвищення ефективності системи управління охорони праці (СУОП) дуже важлива роль належить формуванню і розвитку інформаційної культури фахівців ІТ-технологій, яка впливає на удосконалення інформаційного контуру сучасних підприємств, дозволяє створювати надійні прогнози щодо стану умов праці, показників здоров'я та працездатності, виробничого травматизму і професійної захворюваності, визначати політику розвитку підприємств, установ та організацій на основі різноманітних стратегій охорони праці (інноваційні, маркетингові, інвестиційні, фінансові, технологічні, диверсифікаційні). Поряд з інформаційною культурою важливо використовувати в рамках СУОП «трикутник» її складових: правову (8,1 за 10-бальною шкалою), організаційну (8,0), управлінську (7,5) .

В управлінні охороною праці потрібно реалізувати основні положення, окремі теоретико-методологічні підходи інформаційного менеджменту. Головну роль та відповідальність за стан СУОП мають нести фахівці служби охорони праці сучасного підприємства.

Сучасне суспільство називають постіндустріальним, пост економічним, інформаційним, оскільки йдеться про багатосторонні і кардинальні зміни у розвитку цивілізації.

Інформаційне суспільство передбачає докорінну зміну, яка полягає у перетворенні інформації і знань у головний професійно-виробничий потенціал особистості, соціуму і держави.

На постіндустріальному етапі розвитку суспільства вирішальним фактором стає інформація. Її домінування ініціювала науково-технічна революція, яку ще іменують інформаційною, оскільки нею охоплена будь-яка інтелектуальна діяльність, починаючи з інформаційних образів штучного інтелекту у нових технологіях, економіки, і продовжуючи інформатизацією суспільства в умова світової глобалізації науки й освіти тощо [5].

Під інформаційною культурою розуміють сукупність, складову НІТ (новітні інформаційні технології), технологічну, правову, психологічну, соціологічну та ергономічну підсистеми, що сприяють спрямованому впливу на протікання соціальних процесів у суспільстві, колективі і вихованню свідомого відношення людини до праці, виконання прав та обов’язків [6].

Поняття інформаційної культури виникло в процесі активізації дослідницької уваги до механізмів інформаційного обміну у зв’язку зі значним підвищення ролі інформації в соціокультурних процесах суспільства, яке розглядають як інформаційне суспільство знань, де в центрі знаходяться інформаційні технології.

Робота з інформацією та інформаційна культура в цілому є одним з найважливіших компонентів спроб компанії управляти змінами. Є три принципові причини, в силу яких сьогодні необхідно дбати про інформаційну культуру компанії.

По-перше, вона все більше і більше стає найважливішою частиною загальної організаційної (корпоративної) культури компанії. Все більше компаній розуміють необхідність перетворень, орієнтованих на задоволення очікувань споживача. Щоб сьогодні впливати на майбутнє, потрібно уявляти собі на що вона буде схожа. А для цього потрібно працювати з різноманітною діловою, професійною, технологічною, соціальною, ринковою та політичною інформацією.

По-друге, інформаційні технології роблять можливим створення в компаніях комп'ютерних мереж, за допомогою яких йде спілкування між менеджерами, але важливо знати, як люди використовують цю інформацію. Саме по собі створення такої мережі з усіма її робочими станціями і мультимедійними можливостями не гарантує того, що інформація буде використовуватися більш розумно і більш ефективно.

По-третє, для різних функціональних служб, підрозділів та робочих груп сучасних підприємств в сфері охорони праці інформаційна культура різна, а це означає відмінність методологічних підходів до процесів усвідомлення, збору, організації, обробки, поширення і використання інформації. Тому багато менеджерів погодяться з тим, що корпоративна інформаційна культура важлива для вироблення різних стратегій охорони праці та запровадження відповідних заходів з її вдосконалення [7].

Для деяких галузей, таких як розробка програмного забезпечення, інформаційна культура є необхідною умовою виживання, тому що зміна технологій в розробці програмного забезпечення відбувається кожні 6-8 місяців, а інвестиції на підготовку персоналу і освоєння нової технології величезні і у великих компаніях варіюються від 1,5 до 2 млрд. доларів на рік [8].

Аналіз свідчить, що інформатизація та інтеграція комунікаційного простору України сприяє різкому підвищенню інформаційної та професійної компетентності, ділової активності, стимулюванню конкуренції, створенню інноваційних підприємств та організацій, нових робочих місць, зниженню витрат на утримання управлінського апарату [9].

Поряд із задачами і здобутками окреслилися негативи використання інформаційних технологій:

1) надмірне інформаційне навантаження, суть якого полягає у тому, що кількість корисної інформації, яка надходить до мережі, перевищує психофізіологічні можливості її сприйняття людиною;

2) велика кількість інформації, яка сприймається, але не є корисною для фахівців в даний момент;

3) інформаційний голод, причиною якого є саме надлишок інформації, викликаний інформаційним перенавантаженням;

4) «інформоманія» як хвороба людини, яка робить останню знеособленою, залежною від перебування в інформаційному просторі і

роботи з комп’ютером і чому вона віддає перевагу, уникаючи «живого» спілкування з людьми;

5) поява «кіберспільнот», що за своїми соціокультурними характеристиками набагато ближчі до представників інших культур у глобальному інформаційному просторі, ніж до своєї етнонаціональної спільноти чи решти населення, не охопленого Інтернетом;

6) індивідуалізм і дегуманізація способу життя «мешканців» Інтернету – відсутність готовності ділитися своїми знаннями.

Слід розуміти, що комп'ютерні технології, а особливо їх мережі істотно впливають на життєдіяльність людини, припускаючи глобалізацію і технократизацію суспільства. Але в ще більшій мірі цей вплив поширюється безпосередньо на центральну нервову систему, яка звикає працювати в дуже інтенсивному режимі багатозадачності, де вже переважають не тривалі логічні роздуми, а інтуїтивно-реактивні ланцюжки розумових формулювань у зв'язку з величезним обсягом оброблюваної щодня інформації, кількість якої зростає за експоненціальною швидкістю. Виникає припущення, що саме збільшення обсягу інформації та прискорення її обробки людиною може згубно вплинути на розвиток розумових здібностей людини.

Аналіз продуктивності розумової праці в найбільших за чисельністю фахівців IT-фірм показав, що велике значення з точки зору впливу на її результати має організаційна (корпоративна) культура. В цьому напрямі влаштовуються різні тімбілдинги, заходи, тренінги для розвитку персоналу. Також кожен керівник повинен добре розуміти свого співробітника, що саме для нього важливо, що його мотивує. Важливо відвести потрібну роль відповідному співробітнику, щоб він виконував ті завдання, які йому цікаві [10].

На подібних тренінгах в тому числі повинна розглядатися інформаційна культура працівника, в освоєнні, володінні, мотивуванні, застосуванні, перетворенні інформації із застосуванням сучасних

інформаційних технологій і використанням цих умінь в навчанні з охорони праці і в подальшій професійній діяльності. Особливо вони будуть корисні, як доповнення до існуючих інструктажів з охорони праці на підприємстві, або як контроль психологічного стану та взаємовідносин у колективі.

Інформаційна культура як інтегративне утворення абсолютно не зводиться до розрізнених знань, вмінь та навичок роботи за комп'ютером. Вона передбачає інформативну спрямованість цілісної особистості, яка володіє мотивацією до застосування і засвоєння нових даних. Інформаційну культуру можна розглядати, як одну з граней особистісного розвитку промислових робітників. Це шлях універсалізації якостей людини.

3 Вдосконалення умов праці фахівців з інформаційних технологій

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців ІТ-індустрії.

З метою належного правового забезпечення необхідно розширити ти доповнити перелік основних професій комп’ютерної галузі у національному класифікаторі ДК-003-2010, а також підготувати відповідний випуск у кваліфікаційному довіднику посад фахівців ІТ-індустрії, що сприятиме вирішенню питань їх соціального захисту, пенсійного забезпечення, атестації робочих місць основних професій за умовами праці на предмет подальших певних видів пільг та компенсацій за важкі шкідливі і небезпечні умови праці [11].

Важливим напрямом стосовно визначення професійної придатності фахівців з інформаційних технологій є проведення психофізіологічної експертизи відповідно до 5 статті Закону України «Про охорону праці».

Робота з комп’ютерами нового покоління характеризується певним психофізіологічними перенавантаженнями, втомою зорового аналізатора,

гіпокінезією, відсутність диференційованих норм праці при роботі з новою комп'ютерною технікою в залежності від віку, статі, категорії зорової роботи, режимів праці і відпочинку (протягом робочого дня, тижня, щорічного режиму відпусток).

Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців IT-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів.

Особлива роль з точки зору збереження та відновлення здоров'я працюючих в комп'ютерні' галузі належить попереднім та періодичним наглядам з подальшої психофізіологічної експертизи і встановленням професійної придатності при роботі з комп'ютерами нового покоління, який супроводжується виникненням певних факторів професійного ризику електротравматизму при їх ремонті та обслуговуванні. В цьому зв'язку необхідне запровадження експертизи на предмет безпечної експлуатації ПЕОМ, тобто офіційне підтвердження фактичних параметрів електробезпеки, їх відповідності вимогам нормативної документації фахівців, які проводять таку експертизу повинні пройти навчання і перевірку знань відповідно до вимог ДНАОП 0.00-8.20-99. За результатами експертизи повинні прийматися рішення про відповідність ПЕОМ нормам безпеки, терміни чергової експертизи, оформлюються протоколи вимірювань і випробувань, проведені у разі потреби розрахунки та експертний висновок.

Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців IT-індустрії.

Заслуговує на увагу зарубіжний досвід створення у приміщеннях та в зоні їх розміщення на територіях підприємств спеціальних візуальних комфортних умов та забезпечення вимог виробничої естетики, дотримання норм рівнів виробничого шуму та акустичної тиші за межами офісу. Також дуже важливим є використання в офісних приміщеннях та кабінетах психофізіологічного розвантаження функціональної музики, яка сприяє попередженню перевтоми і підтриманню необхідного рівня розумової працездатності фахівців комп'ютерної галузі.

В цьому напрямі заслуговує на увагу створення при великих центрах інформаційних технологій кімнат (кабінетів) психофізіологічного розвантаження працівників галузі (на 5 місць).

Зарубіжний досвід охорони праці при використанні новітніх інформаційних технологій та сучасного комп'ютерного обладнання передбачає з метою попередження наслідків монотонної праці, підвищення рівня рухової активності і покращення розумової працездатності фахівців IT-індустрії під час технологічних перерв участь у спеціальних облаштованих приміщеннях необхідним спортивним інвентарем та різними тренажерами відповідних фізичних вправ, індивідуальних тренінгових завдань відповідно до віку, статі та категорії зорової роботи. Такий підхід дозволяє зняти надлишкове психофізіологічне перевантаження, підвищити працездатність центральної нервової системи, попередити перевтому зорового аналізатора. Показана ефективність проведення різноманітних за своєю спрямованістю вправ робітників цієї галузі (приблизно на 5-30%).

Всі наведені заходи щодо вдосконалення охорони праці фахівців IT-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства.

Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

ВИСНОВКИ

Інноваційний розвиток економіки та соціальної сфери України вимагає надійного та сучасного інформаційного забезпечення, впровадження новітніх технологій у різних галузях господарського комплексу України. Ключовими професіями в цьому напрямі є фахівці ІТ-технологій.

Аналіз умов праці ІТ-фахівців свідчить про наявність та можливий вплив наступних шкідливих та небезпечних чинників: шуму; несприятливому мікроклімату; іонізуючі і неіонізуючі випромінювання; недостатнє штучне та природнє освітлення; візуальні фактори: надмірна яскравість, контрастність, мерехтіння зображення, відблиски тощо. Враховуючи що для забезпечення роботи квантових комп’ютерів потрібна напруга понад 1000В, проведення з ними.

Для підвищення ефективності СУОП дуже важлива роль належить формуванню і розвитку інформаційної культури фахівців ІТ-технологій, яка впливає на удосконалення інформаційного контуру сучасних підприємств, дозволяє створювати надійні прогнози щодо стану умов праці, показників здоров’я та працездатності, виробничого травматизму і професійної захворюваності, визначати політику розвитку підприємств, установ та організацій на основі різноманітних стратегій охорони праці.

З метою вдосконалення питань охорони праці фахівців з сучасних інформаційних технології розроблені системні заходи з правової оптимізації, соціального захисту працівників, діагностики їх професійної придатності, необхідності обґрунтування диференційованих норм праці при роботі з новою комп’ютерною технікою в залежності від віку, статі, категорії зорової роботи, режимів праці та відпочинку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ELARTU – Інституційний репозитарій ТНТУ імені Івана Пулюя: вебсайт.

URL: http://elartu.tntu.edu.ua/bitstream/lib/30726/3/dyplom_Veselovska.pdf

2. ELARTU – Інституційний репозитарій ТНТУ імені Івана Пулюя: вебсайт. Сервіс для адміністрування і обліку роботи автомобільної парковки

URL:http://elartu.tntu.edu.ua/bitstream/lib/31362/1/magisterska_Nadozirnii.pdf

3. Майже як в Google: чим дивують офіси українських IT-компаній. Головні новини шоубізу сьогодні. Тенденції моди, рецепти і здоровий спосіб життя. Новини здоров'я і фітнес, психологія. Новинки кіно і музики, афіша прем'єр: вебсайт. URL: https://lifestyle.segodnya.ua/ua/lifestyle/fun/pochti-kak-u-google-chem-udivlyayut-ofisy-ukrainskih-it-kompaniy--764025.html

4. П. С. Атаманчук, В. В. Мендерецький, О. П. Панчук, Р. М. Білик.Охорона праці в галузі: навчальний посібник.. URL: https://chmnu.edu.ua/wp-content/uploads/2016/07/Atamanchuk-P.-S.-ta-inshi-Ohorona-pratsi-v-galuzi.pdf

5. Система управління охороною праці. Освіта в Україні та за кордоном – Освіта.UA: вебсайт. URL:https://osvita.ua/vnz/reports/bjd/26509/

6. Інформаційна культура. Національна бібліотека України імені Ярослава Мудрого – провідний державний культурний,...: вебсайт. URL: https://ube.nlu.org.ua/article/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0%20%D0%BA%D1%83%D0%BB%D1%8C%D1%82%D1%83%D1%80%D0%B0

7. Лекції – Інформаційні технології в управлінні. Навчальний матеріал: вебсайт. URL:https://uadoc.zavantag.com/text/39822/index-6.html

8. Інформаційні системи і технології в економіці. webvemon: вебсайт. URL:https://sites.google.com/site/webvemon/iste-1

9. Інноваційні стратегії в системі підвищення конкурентоспроможності економіки України. Головна - Львівський національний університет імені Івана Франка: вебсайт. URL:https://www.lnu.edu.ua/wp-content/uploads/2016/11/dis_yurynets.pdf

10. Тема 1. КОРПОРАТИВНА КУЛЬТУРА. Наукова бібліотека ДВНЗ «Прикарпатський національний університет імені Василя Стефаника»: вебсайт. URL: http://194.44.152.155/elib/local/820.pdf

11. Про затвердження Порядку розроблення та затвердження кваліфікаційних характеристик: наказ М-ва соціальної політики України від 23 червня 2017 р. № 784/30652