**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

_____ Євген ДАВИДЕНКО

«_9_» __червня__ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

# ГРА В ЖАНРІ TOP-DOWN SHOOTER

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

**Здобувач** _____ **Роман ЧЕЧІНЬОВ**

«_9_» __червня___ 2025 р.

**Керівник роботи**

PhD, старший викладач _____ **Ігор КАНДИБА**

«_9_» ___червня__ 2025 р.

**Миколаїв – 2025**

**Завдання на виконання кваліфікаційної роботи**

Чорноморський національний університет імені Петра Могили

| | |
|---|---|
| Факультет | Комп'ютерних наук |
| Кафедра | Інженерії програмного забезпечення |
| Рівень вищої освіти | Перший (бакалаврський) |
| Освітній ступінь | Бакалавр |
| Спеціальність | 121 Інженерія програмного забезпечення |
| Освітня програма | Інженерія програмного забезпечення |

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«1» __лютого_____ 2025 р.

**ЗАВДАННЯ**

**на кваліфікаційну бакалаврську роботу здобувача**

**Чечіньов Роман**

1. Тема кваліфікаційної роботи «Гра в жанрі top-down shooter» затверджена наказом ректора ЧНУ ім. Петра Могили № _315_ від «_13_» ___листопада__ 2024 р.
2. Строк представлення кваліфікаційної роботи «18» _червня___ 2025 р.
3. Очікуваний результат роботи та початкові дані: Розроблена гра в жанрі top-down shooter на платформі Windows
4. Перелік питань, що підлягають розробці:
   - провести аналіз предметної області та схожих програмних застосунків;
   - спроектувати гру в жанрі top-down shooter;
   - обрати технології для розроблення гри в жанрі top-down shooter;

- визначити функціональні вимоги до гри;
- реалізувати програмний застосунок;
- тестування розробленого програмного застосунку.

5. Перелік графічних матеріалів:
   - Презентація
6. Консультанти:

| **Консультант** | **Кафедра (організація)** | **Частина роботи** |
|---|---|---|
| | | |
| | | |
| | | |

Дата видачі завдання «__13_» _____листопада____ 2024__р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання кваліфікаційної роботи**

Тема: **Гра в жанрі top-down shooter**

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1. | Розробка та затвердження завдання на виконання КБР | 1.02.2025 | 7.02.2025 | Виконано |
| 2. | Огляд літератури за темою роботи | 8.02.2025 | 13.02.2025 | Виконано |
| 3. | Складання календарного плану КБР | 14.02.2025 | 20.02.2025 | Виконано |
| 4. | Аналіз предметної області | 21.02.2025 | 25.02.2025 | Виконано |
| 5. | Розробка проєктних рішень | 26.02.2025 | 28.02.2025 | Виконано |
| 6. | Моделювання та конструювання ПЗ | 1.03.2025 | 15.03.2025 | Виконано |
| 7. | Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача | 16.03.2025 | 5.05.2025 | Виконано |
| 8. | Відгук керівника КБР | 6.05.2025 | 6.05.2025 | Виконано |
| 9. | Оформлення КБР та презентації | 7.05.2025 | 27.05.2025 | Виконано |
| 10. | Попередній захист | 28.05.2025 | 28.05.2025 | Виконано |
| 11. | Рецензування | 4.06.2025 | 4.06.2025 | Виконано |
| 12. | Завершення оформлення КБР та презентації | 5.06.2025 | 10.06.2025 | Виконано |
| 13. | Захист кваліфікаційної роботи | 24.06.2025 | 24.06.2025 | Виконано |

**Здобувач** _____ **Роман ЧЕЧІНЬОВ**

«22» __лютого_____ 2025 р.

**Керівник роботи**

PhD, Старший викладач _____ **Ігор КАНДИБА**

«22» __ лютого _____ 2025 р.

# АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

**Гра в жанрі top-down shooter**

Здобувач 409 гр.: Чечіньов Роман

Керівник: PhD, ст. викладач Кандиба Ігор

Актуальність обраної теми зумовлена не лише стрімким зростанням ринку комп'ютерних ігор як ключової сфери розваг, але й постійним пошуком інноваційних підходів у класичних жанрах. Розробка застосунку в жанрі top-down shooter, що потенційно поєднує перевірені часом механіки з сучасними трендами. Такий застосунок має потенціал не лише вдихнути нове життя в жанр та привнести в нього свіжі ідеї, але й сприяти його популяризації серед широкого кола гравців.

**Об’єктом роботи** є процес розробки гри в жанрі top-down shooter.

**Предметом роботи** є інструментарій для розробки гри в жанрі top-down shooter.

**Метою роботи** є розробка гри в жанрі top-down shooter, яка популяризує жанр серед гравців.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність теми, визначено мету, завдання, об’єкт та предмет дослідження

У першому розділі проведено детальний аналіз предметної області, а також проаналізовано існуючі ігри-аналоги для виявлення вдалих рішень

Другий розділ присвячено розробленню мокапів інтерфейсів ігрового застосунку. Також у цьому розділі розроблено специфікацію функціональних та нефункціональних вимог до програмного забезпечення.

Кваліфікаційна робота викладена на 62 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з _9_ найменувань та _0_ додатків. Праця містить _11_ таблиць та _50_ рисунків.

*Ключові слова: Гра, Godot Engine, shooter, top-down shooter, GDScript, 2D*

**ABSTRACT**

to the qualifying bachelor's thesis

**A top-down shooter game**

Student of 409 group: Chechinev Roman

Supervisor: PhD, Senior lecturer Kandyba Ihor

The relevance of the chosen topic is due not only to the rapid growth of the computer games market as a key area of entertainment, but also to the constant search for innovative approaches in classic genres. Development of an application in the top-down shooter genre, which potentially combines time-tested mechanics with modern trends. Such an application has the potential not only to breathe new life into the genre and bring fresh ideas to it, but also to promote its popularization among a wide range of players.

**The object of the work** is the process of developing a game in the top-down shooter genre.

**The subject of the work** is a toolkit for developing a game in the top-down shooter genre.

**The purpose of the work** is to develop a game in the top-down shooter genre that will popularize the genre among players.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

The introduction substantiates the relevance of the topic, defines the goal, objectives, object and subject of the study

The first section provides a detailed analysis of the subject area, as well as analyzes existing games-analogues to identify successful solutions

The second section is dedicated to the development of game application interface mockups. This section also develops a specification of functional and non-functional software requirements.

The qualification work is presented on 62 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 9 titles and 0 appendices. The work contains 11 tables and 50 figures.

*Keywords: Game, Godot Engine, shooter, top-down shooter, GDScript, 2D*

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

ПК – Персональний комп'ютер

ПЗ – Програмне забезпечення

API – Application Programming Interface

FPS – Frames Per Second

HUD – Heads-Up Display

OS – Operating System

RAM – Random Access Memory

RNG – Random Number Generation

UE – Unreal Engine

UI – User Interface

XP – Experience Points

# ВСТУП

У сучасному світі комп’ютерні ігри перетворилися з невідомого хобі на одну з домінуючих форм розваг, що за обсягом ринку та культурним впливом впевнено конкурує з кінематографом, музичною індустрією та телебаченням. Ця динамічна галузь не лише генерує мільярдні прибутки та демонструє стабільне зростання, але й об’єднує людей різних професій – програмістів, художників, дизайнерів, сценаристів, звукорежисерів, маркетологів – для створення одного комплексного продукту. Такий міждисциплінарний підхід сприяє створенню значної кількості робочих місць та стимулює розвиток суміжних галузей.

Більш того, ігрова індустрія часто виступає рушієм технологічного прогресу, висуваючи нові вимоги до апаратного забезпечення, графічних технологій, штучного інтелекту. Різноманіття жанрів та платформ свідчить про широке охоплення аудиторії та постійний пошук розробниками нових способів залучення та утримання гравців. Враховуючи стійку тенденцію до зростання прибутків та впливу ігрових застосунків за минулі роки, можна впевнено стверджувати, що розробка комп’ютерних ігор є актуальним та перспективним напрямом діяльності, який і надалі відіграватиме важливу роль у глобальній економіці та культурі.

**Мета**: розробка гри в жанрі top-down shooter задля популяризації жанру серед гравців

Для досягнення поставленої мети, необхідно виконати наступні завдання:

- провести аналіз предметної області та схожих програмних застосунків;
- спроєктувати гру в жанрі top-down shooter;
- обрати технології для розроблення гри в жанрі top-down shooter;
- визначити функціональні вимоги до гри;
- реалізувати програмний застосунок;
- тестування розробленого програмного застосунку.

**Об’єкт роботи**: процес розробки гри в жанрі top-down shooter

**Предмет роботи**: інструментарій розробки гри в жанрі top-down shooter

# 1 АНАЛІЗ СУЧАСНИХ ІГРОВИХ ЗАСТОСУНКІВ В ЖАНРІ TOP-DOWN SHOOTER

## 1.1 Аналіз предметної області

З кожним роком популярність комп’ютерних та мобільних ігор зростає як у світі, так і в Україні. За даними ресурсу Exploding Topics(2025) [1,2]:

– У всьому світі приблизно 3.32 мільярди людей які грають в ігри

– На сьогоднішній день ігрова індустрія коштує 522.46 мільярди доларів

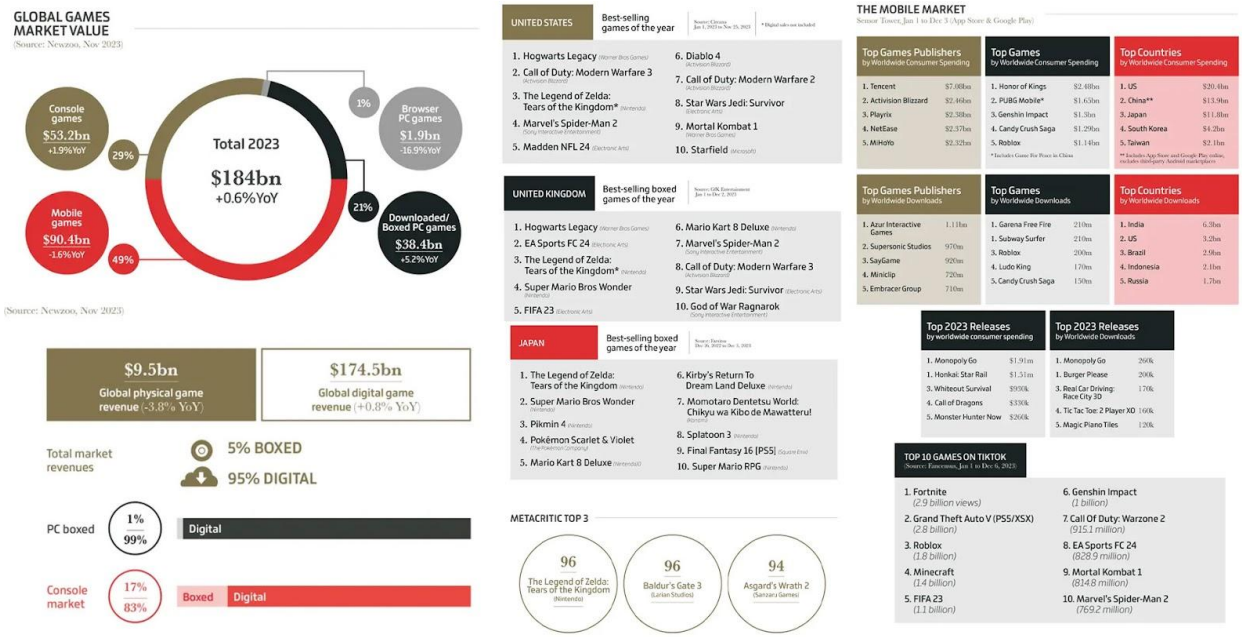


Рисунок 1.1 – Огляд глобального ринку ігор

В той же час незважаючи на відсутність майже всякого розвитку за минулі роки, Європейська федерація EGDF відзначає, що Українська ігрова індустрія є однією з найдинамічниших галузей у Європі,  а також налічувала понад 200 компаній та 5000 спеціалістів та мала 400 мільйонів доларів доходів у 2023 році [3].


Рисунок 1.2 – Огляд ігрових компаній України

Top-down shooter – є одним з класичних жанрів ігрових жанрів, особливістю якого є перспектива камери розташованої над ігровим персонажем. Зазвичай має динамічний геймплей та має багато підвидів: від сюжетних шутерів (Hotline Miami) до roguelike варіацій як Enter the Gungeon та Nuclear Throne [4].

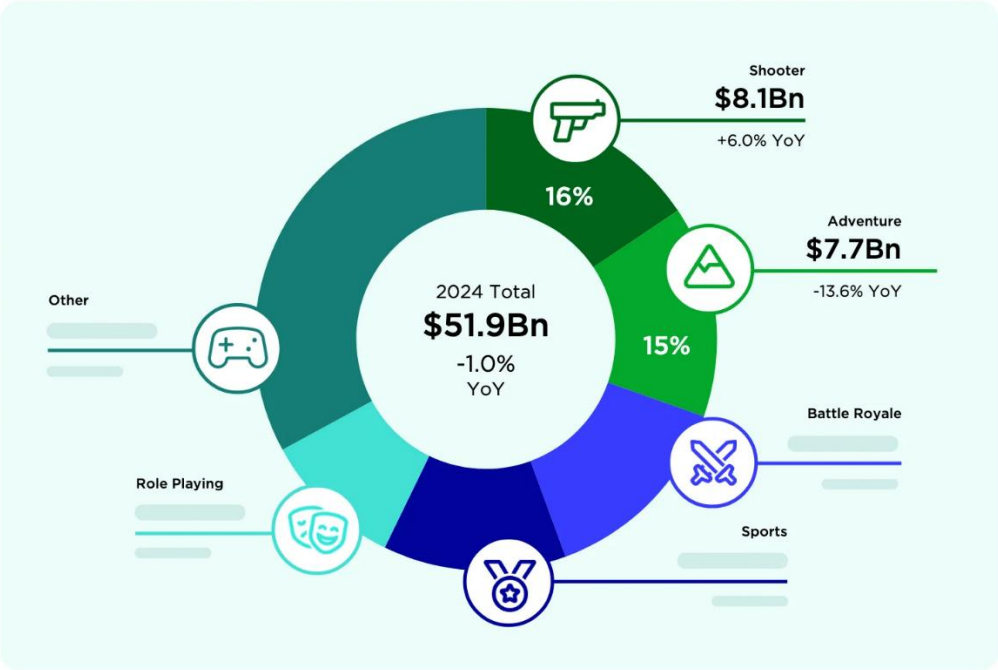


Рисунок 1.3 – Найпопулярніші жанри на консолях

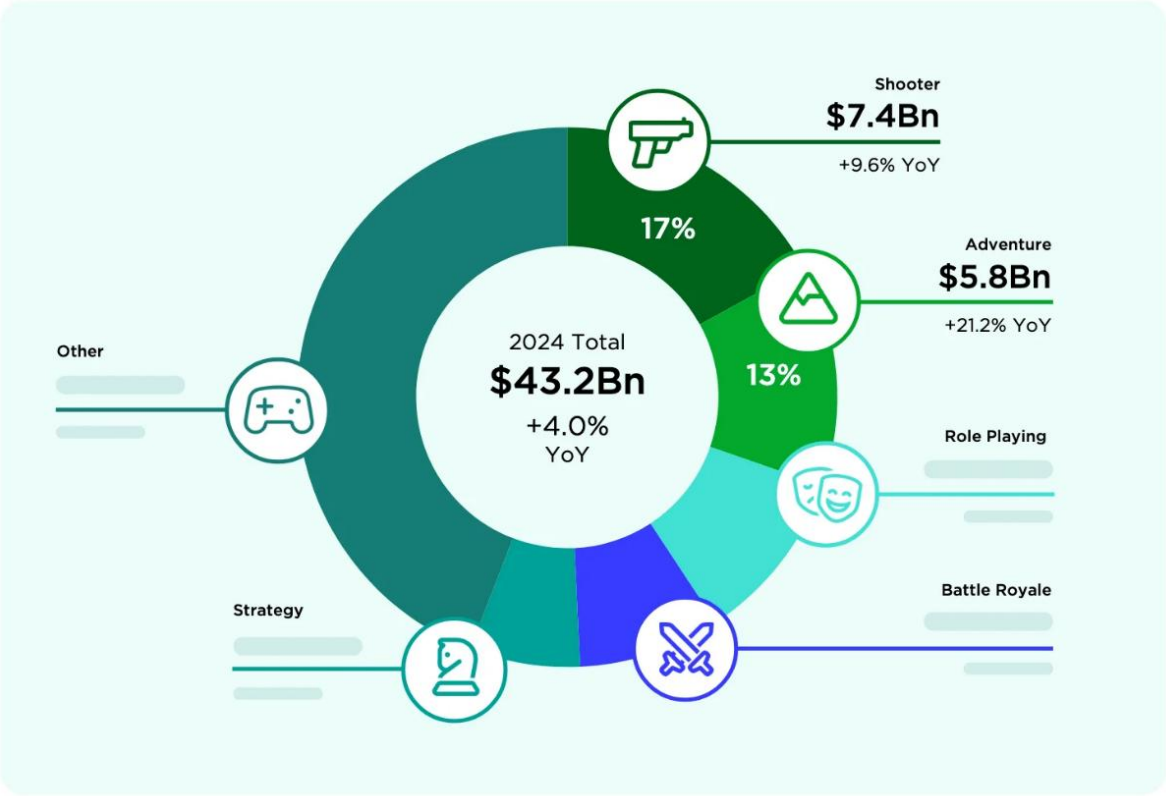


Рисунок 1.4 – Найпопулярніші жанри на персональних комп’ютерах

## 1.2 Аналіз аналогів

З першого погляду жанр top-down shooter виглядає дуже простим в своїй основі, але на його основі було створено багато культових та популярних ігор які буде сильно відрізняються одна від одної.

**Hotline Miami (Dennaton Games, 2012)**

Hotline Miami – це культовий top-down shooter, відомий своїм екстремально швидким темпом, високим рівнем жорстокості та унікальною неоновою стилістикою 80-х років у поєднанні з піксель-арт графікою. Гравець виступає в ролі безіменного протагоніста, який отримує загадкові телефонні дзвінки з інструкціями щодо масових вбивств. Геймплей вимагає швидкого планування, бездоганної реакції та багаторазових спроб для проходження рівнів, оскільки і гравець, і вороги гинуть від одного-двох влучань. Важливою частиною гри є система масок, що надають різні бонуси, та сюрреалістичний, фрагментований сюжет.

Таблиця 1.1 – Характеристики Hotline Miami

| **Назва** | Hotline Miami |
|---|---|
| Ігровий рушій | GameMaker: Studio |
| Платформи | PC (Windows, macOS, Linux), PS3, PS4, PS Vita, Nintendo Switch, Xbox One, Android |
| Переваги | – захоплюючий та адреналіновий геймплей;<br>– сильна візуальна ідентичність та атмосферний саундтрек;<br>– висока реіграбельність завдяки системі масок та оцінок;<br>– добре реалізоване відчуття удару та летальності. |
| Недоліки | – високий поріг входження;<br>– штучний інтелект ворогів іноді погано працює;<br>– складний для розуміння сюжет;<br>– графіка та спецефекти можуть спричинити епілептичні напади. |
| Основні механіки | – система масок з унікальними бонусами;<br>– система комбо та оцінок за стиль проходження. |
| Посилання | Сторінка в Steam |

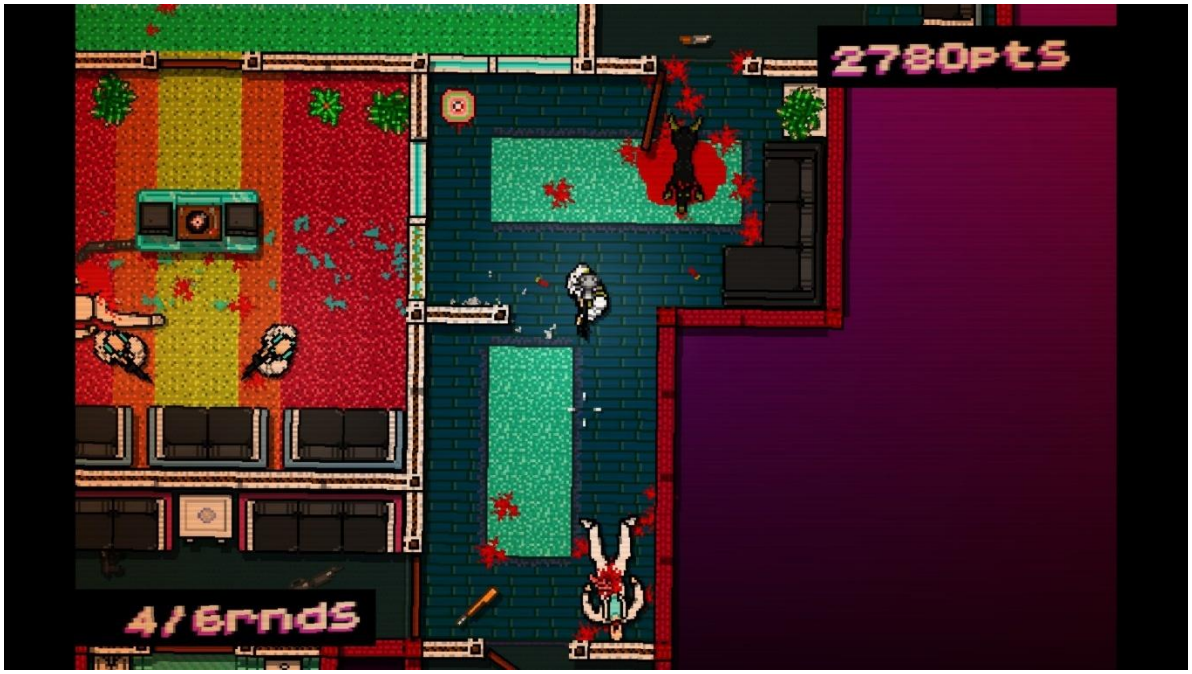


Рисунок 1.5 – Ігровий процес Hotline Miami

**Enter the Gungeon (Dodge Roll, 2016)**

Enter the Gungeon поєднує елементи top-down shooter з механіками bullet hell та roguelike. Гравець обирає одного з кількох персонажів і спускається у лабіринт, наповнений ворогами, пастками та босами. Основна мета – знайти легендарну зброю, що може вбити минуле. Гра вирізняється величезною кількістю унікальної та часто абсурдної зброї та предметів, які можуть взаємодіяти між собою. Ключовою механікою є перекат, що дозволяє ухилятися від ворожих куль. Процедурна генерація рівнів забезпечує високу реіграбельність.

Таблиця 1.2 – Характеристики Enter the Gungeon

| **Назва** | Enter the Gungeon |
|---|---|
| Ігровий рушій | Unity |
| Платформи | PC (Windows, macOS, Linux), PS4, Xbox One, Nintendo Switch |

Кінець таблиці 1.2

| | |
|---|---|
| Переваги | – велика кількість зброї, активних та пасивних предметів, ворогів та босів;<br>– система синергій між предметами та зброєю;<br>– висока реіграбельність завдяки системі процедурної генерації рівнів;<br>– наявність локального кооперативу. |
| Недоліки | – високий поріг входження;<br>– високий вплив RNG;<br>– багато часу для відкриття всього контенту гри;<br>– дуже інтенсивні візуальні ефекти від деяких ворогів та зброї. |
| Основні механіки | – процедурна генерація рівнів;<br>– мета-прогресії;<br>– система синегрій;<br>– локальний кооператив;<br>– механіка перекату. |
| Посилання | Сторінка в Steam |

Рисунок 1.6 - Ігровий процес Enter the Gungeon

**Helldivers (Arrowhead Game Studios, 2015)**

Helldivers – це кооперативний тактичний top-down shooter з акцентом на командну взаємодію та виконання завдань у ворожому середовищі. Гравці виступають у ролі елітних солдатів "Пекельних Десантників", що борються за Супер-Землю проти інопланетних загроз на процедурно генерованих планетах. Ключовою особливістю є система "Стратагем" – виклик з орбіти різноманітного озброєння, техніки, боєприпасів або підкріплення за допомогою введення спеціальних кодів. Гра відома високою складністю та наявністю постійного дружнього вогню, що вимагає координації та обережності.

Таблиця 1.3 – Характеристики Helldivers

| **Назва** | Helldivers |
|---|---|
| Ігровий рушій | Bitsquid (Stingray) |
| Платформи | PC (Windows), PS3, PS4, PS Vita |
| Переваги | – кооператив до 4 гравців;<br>– унікальна система Стратагем;<br>– висока складність та напруженість завдяки дружньому вогню;<br>– глобальна кампанія, де дії гравців впливають на хід війни. |

Кінець таблиці 1.3

| | |
|---|---|
| Недоліки | – одноманітні місії;<br>– висока залежність від дій інших гравців;<br>– одноманітний геймплей при грі в соло. |
| Основні механіки | – система стратегем;<br>– система прогресії відкриття нової зброї, спорядження та стратегем;<br>– три ворожі фракції з унікальними ворогами та тактиками;<br>– глобальна компанія. |
| Посилання | Сторінка в Steam |


Рисунок 1.7 - Ігровий процес Helldivers

**Nuclear Throne (Vlambeer, 2015)**

Nuclear Throne – це популярний roguelike top-down shooter, дія якого відбувається у пост-апокаліптичному світі. Гравець обирає одного з «мутантів», кожен з яких має унікальні активні та пасивні здібності, і намагається пробитися крізь натовпи ворогів до «Ядерного Трону». Гра характеризується дуже швидким темпом, мінімалістичною, але виразною піксель-арт графікою та високою складністю. Важливою механікою є збір радіації з переможених ворогів, яка дозволяє отримувати рівні та обирати нові мутації, що впливають на геймплей.

Таблиця 1.4 – Характеристики Nuclear Throne

| **Назва** | Nuclear Throne |
|---|---|
| Ігровий рушій | GameMaker: Studio |
| Платформи | PC (Windows, macOS, Linux), PS4, PS Vita, Nintendo Switch, Xbox One |
| Переваги | – швидкий, динамічний геймплей;<br>– висока реіграбельність завдяки roguelike елементам та системі мутацій;<br>– різноманітні персонажи з унікальними здібностями та однією альтернативною формою. |
| Недоліки | – надзвичайно висока складність для нових гравців;<br>– сильний вплив RNG;<br>– мінімалістичний сюжет та графіка;<br>– деякі мутації корисніші, ніж інші. |

Кінець таблиці 1.4

| | |
|---|---|
| Основні механіки | – процедурна генерація рівнів;<br>– персонажі з унікальними здібностями та однією альтернативною формою;<br>– система мутацій;<br>– велика кількість різноманітної зброї. |
| Посилання | Сторінка в Steam |


Рисунок 1.8 - Ігровий процес Nuclear Throne

Проаналізувавши виділені аналоги, можна скласти порівняльну таблицю.

Таблиця 1.5 – Порівняльна таблиця

| **Критерій** | **Hotline Miami** | **Enter the Gungeon** | **Helldivers** | **Nuclear Throne** |
|---|---|---|---|---|
| Ігровий рушій | GameMaker: Studio | Unity | Bitsquid (Stingray) | GameMaker: Studio |

Кінець таблиці 1.5

| | | | | |
|---|---|---|---|---|
| Складність | Висока | Висока | Висока | Дуже висока |
| Основні механіки | Маски, комбо | Перекат, синергії зброї та предметів | Стратегеми, командна взаємодія | Мутації, здібності персонажів |
| Система прогресії | Розблокування масок під час проходження компанії | Мета-прогресія за внутрішньоігрову валюту | Розблокування зброї, стратегем під час компанії | Розблокування альтернативних версій персонажів |
| Мультиплеєр | Немає | Локальний до 2 гравців | Онлайн до 4 гравців | Локальний до 2 гравців |

**Висновки до розділу 1**

В першому розділі проведено комплексний аналіз предметної області та існуючих аналогів ігрових застосунків з метою виявлення тенденцій та основних особливостей жанру та формування ні їх основі вимог до розроблюваного ПЗ

В підрозділі 1.1 розглянуто ігрову індустрію загалом у світі та в Україні, та тенденції її розвитку протягом останніх декількох років.

Встановлено стабільний ріст ігрової індустрії у світі із загальним прибутком в 184 мільярди доларів розділені між персональними комп’ютерами, консолями та мобільними телефонами.

Ігрова індустрія України визнана як одна з найдинамічніших в Європі з прибутком в 400 мільйонів доларів в 2023 році, серед більш ніж двохсот компаній з загальною кількістю працівників до 5000.

В підрозділі 1.2 досліджено існуючі аналоги, виявлено їх особливості, їх переваги та недоліки.

На основі розглянутих аналогів було встановлено, що жанр top-down shooter є дуже гнучким та може поєднуватись з багатьма іншими жанрами, що призводить до появи унікальних ігрових ситуацій.

# 2 МОДЕЛЮВАННЯ ГРИ В ЖАНРІ TOP-DOWN SHOOTER

## 2.1 Дослідження сучасних ігрових рушіїв

Опираючись на дослідження ігрових рушіїв Unity [5], Godot Engine [6] та Unreal Engine [7] можна зробити такі висновки про основне призначення, переваги та недоліки кожного рушія:

**Unity**

Авторами дослідження [5] було дано таке визначення ігрового рушія Unity:

Unity – кросплатформений середа розробки ігрових застосунків розроблена американською компанією Unity Technologies у 2005 році представлений на Всесвітній конференції розробників з метою полегшення процесу розробки ігрових застосунків. Unity підтримує більш ніж 25 різних платформ, такі як: персональні комп’ютери, ігрові консолі, мобільні телефони.

Unity є дуже популярним ігровим двигуном на якому розроблено безліч відомих та популярних ігор, такі як: Kerbal Space Program, Temple Run, Enter the Gungeon, Valheim, Ori and the Will of the Wisps, Hearthstone, Pathfinder: Wrath of the Righteous.

Головними перевагами Unity серед інших ігрових рушіїв є:

– велика кількість документації, покрокових гайдів та велика спільнота сформована за довгі роки існування Unity;

– мова програмування C# легша в вивчені на початкових етапах, що знижує поріг входу до розробки ігор;

– легко знайти ассети для ігор та своїх проектів через існування великої спільноти розробників;

– unity продовжують підтримувати та розвивати додаючи новий функціонал.

Серед недоліків Unity можна виділити:

– політика ліцензування;

– розмір проектів;

– потрібна додаткова оптимізація великих проектів;

– для новачків Unity може показатися складним через велику кількість функцій.

Рисунок 2.1 – Логотип Unity

**Godot Engine**

В досліджені [6] авторами було наведено такі особливості, переваги та недоліки ігрового рушія Godot Engine:

Godot Engine – це повністю безкоштовний ігровий рушій з відкритим вихідним кодом, що стрімко розвивається завдяки зусиллям великої міжнародної спільноти.

Його філософія полягає у наданні гнучкого, інтегрованого та вільного інструменту для розробників усіх рівнів. Godot вирізняється своєю унікальною архітектурою, побудованою на системі сцен та вузлів, яка дозволяє інтуїтивно конструювати ігрові об'єкти та логіку.

Хоча рушій є потужним інструментом і для 3D розробки, його історичною силою та однією з ключових переваг вважається першокласна підтримка 2D.

Багато інструментів та підходів для 2D інтегровані безпосередньо в ядро рушія, що робить процес створення двовимірних ігор надзвичайно зручним та ефективним.

Легка вага самого редактора, проста для вивчення мова GDScript та повна відсутність ліцензійних платежів роблять Godot особливо привабливим для інді-розробників та освітніх цілей.

Головними перевагами Godot Engine є:

– Відсутність ліцензійних обмежень;
– низький поріг входу через мову GDScript;
– гарна оптимізація для роботи з 2D;
– малий розмір рушія та проектів;
– швидкозростаюча активна спільнота.

Серед недоліків Godot Engine можна виділити:

– Гірші 3D можливості у порівнянні з Unity та UE;
– нерозвинена підтримка консолей;
– менший вибір готових асетів;
– більше компаній використовують Unity та UE, як наслідок менше вакансій для Godot-розробників на ринку праці.

Приклади ігор на Godot:

– Cassette Beasts — покрокова RPG з відкритим світом.
– The Garden Path — симулятор життя з атмосферною 2D графікою.
– Kingdoms of the Dump — ретро-гра у стилі SNES, створена з любов'ю фанатами RPG.


Рисунок 2.2 – Логотип Godot Engine

**Unreal Engine (UE)**

В досліджені [7] автори описують можливості використання інструментарію Unreal Engine так:

Unreal Engine, розроблений компанією Epic Games, є одним із стовпів сучасної ігрової індустрії, синонімом передових технологій та високобюджетних AAA-проєктів.

Цей рушій славиться своїми неперевершеними можливостями у сфері 3D-графіки, забезпечуючи фотореалістичний рендеринг в реальному часі завдяки таким технологіям, як глобальне освітлення Lumen та система віртуалізованої геометрії Nanite.

Хоча основною мовою програмування є C++, що вимагає глибоких знань, рушій також пропонує унікальну та дуже розвинену систему візуального скриптування Blueprints. Вона дозволяє дизайнерам та художникам створювати складну ігрову логіку та інтерактивність без написання жодного рядка коду, що значно прискорює ітерації та прототипування.

Unreal Engine також має базові інструменти для 2D розробки, але вони історично є менш пріоритетними для Epic Games і поступаються за гнучкістю та зручністю спеціалізованим рішенням Unity та Godot. Вибір Unreal Engine зазвичай обґрунтований потребою у найвищій якості 3D-графіки або при розробці масштабних протктів, де його комплексний інструментарій та оптимізації для високонавантажених сцен стають вирішальними.

Головними перевагами Unreal Engine є:

– Система Blueprint дозволяє швидко реалізовувати ігрову логіку;

– гарна оптимізація для роботи з 3D об’єктами;

– велика спільнота;

– безкоштовна ліцензія до досягнення доходу в 1 мільйон доларів.

Серед недоліків Unreal Engine можна виділити:

– орієнтація на 3D графіку залишило рушій без ефективних інструментів для роботи з 2D графікою;

– UE потребує потужного ПК для комфортної роботи;

– великий розмір проектів.

Ігри, створені в Unreal Engine:

– Fortnite – один із найвідоміших багатокористувацьких шутерів.

– Hellblade: Senua’s Sacrifice – приклад глибокої візуальної та наративної роботи.

– Gears 5, Street Fighter V, Final Fantasy VII Remake, The Matrix Awakens (демо UE5).


Рисунок 2.3 – Логотип Unreal Engine

**BuildBox**

Buildbox – це комерційний рушій, орієнтований насамперед на розробку ігор без програмування. Його основна мета – зробити створення ігор доступним для користувачів, які не мають технічного досвіду. Завдяки системі побудови логіки на основі блоків drag-and-drop, розробники можуть швидко створювати прототипи та повноцінні ігри.

Buildbox підтримує експорт на різні платформи, включаючи Android, iOS, Windows. Середовище розробки інтуїтивне та зручне, з попередньо налаштованими шаблонами і механіками.

Хоча рушій ідеально підходить для створення казуальних 2D ігор, його можливості обмежені при реалізації складних ігор або специфічної логіки. Крім того, повний функціонал доступний лише у платній версії.

Головними перевагами Buildbox є:

- простота використання та відсутність потреби у програмуванні;
- швидка побудова ігор завдяки drag-and-drop інтерфейсу;
- вбудовані шаблони та асети для початківців;
- підтримка мультиплатформного експорту.

Серед недоліків Buildbox можна виділити:

– обмежена функціональність у безкоштовній версії;
– відсутність гнучкості для реалізації складної логіки;
– відносно висока вартість підписки;
– не підходить для 3D-розробки.

Ігри, створені в Buildbox:

– Color Switch – гіперказуальний хіт, що став вірусним у 2016 році.
– The Line Zen – ще один приклад простої, але захоплюючої гри.
– Phases – стильна аркада з мінімалістичним геймплеєм.

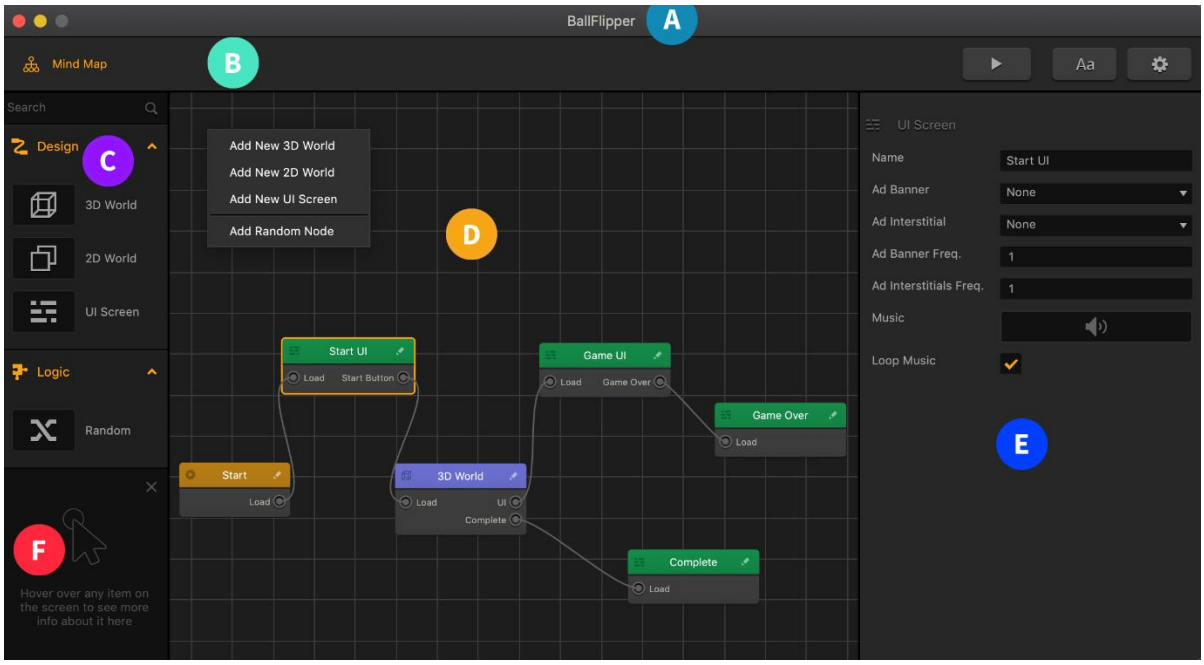


Рисунок 2.4 – Інтерфейс BuildBox

**GameMaker Studio 2**

GameMaker Studio 2 – це потужний 2D-орієнтований рушій, що поєднує в собі простоту використання з можливістю гнучкого програмування.

Його основною особливістю є підтримка як візуального програмування, так і власної мови GameMaker Language, що дозволяє створювати складну логіку для проєктів будь-якого рівня.

GameMaker Studio 2 користується популярністю серед інді-розробників та малих студій, особливо завдяки низькому порогу входу і високій продуктивності при створенні 2D-ігор.

Ігри на GameMaker Studio 2 мають гарну оптимізацію, а рушій підтримує експорт на різні платформи, включаючи ПК, консолі та мобільні пристрої.

Основні переваги GameMaker Studio 2:

- проста мова програмування GML;
- підтримка як коду, так і візуального програмування;
- стабільна і швидка робота з 2D-графікою;
- багатоплатформність і якісний експорт.

Недоліки рушія:

- платна ліцензія з додатковими витратами на експорт для окремих платформ;
- обмежена підтримка 3D;
- менше вбудованих інструментів для командної розробки;
- застарілий інтерфейс для деяких користувачів.

Відомі ігри, створені на GameMaker Studio 2:

- Undertale — культова інді-RPG.
- Hyper Light Drifter — стильний action-RPG.
- Katana ZERO — неоновий екшен-платформер.
- Hotline Miami — динамічний top-down shooter.

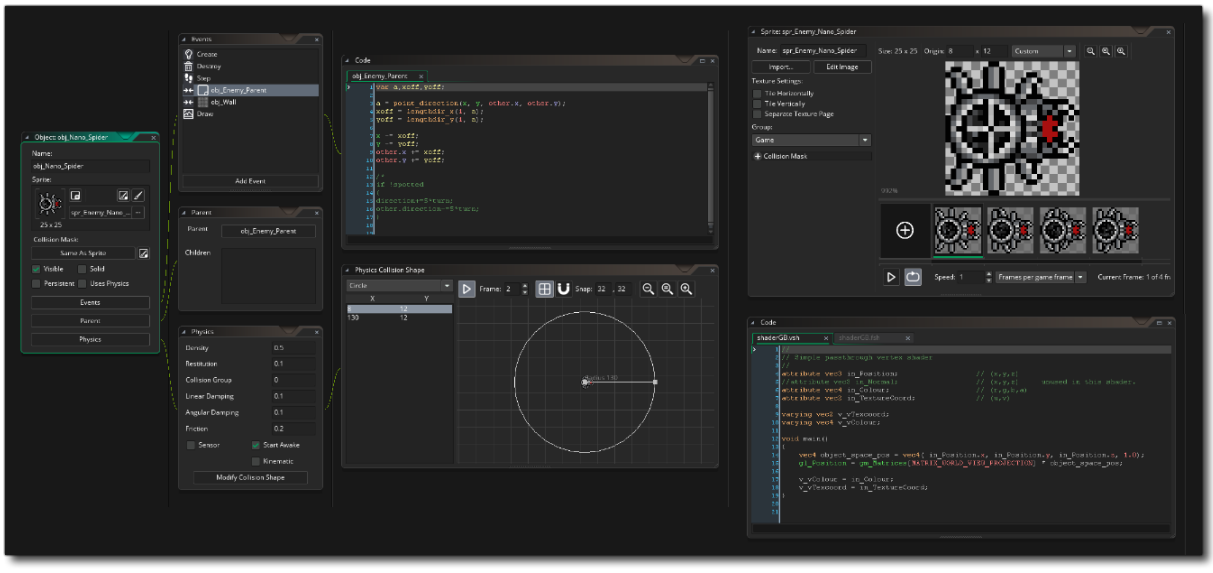

Рисунок 2.5 – Інтерфейс GameMaker Studio 2

### Construct

Construct — це ігровий рушій, що спеціалізується на 2D-розробці та орієнтований на користувачів без глибоких знань програмування. Він ідеально підходить для новачків, освітніх проєктів та інді-розробників, які прагнуть швидко створити ігровий прототип або повноцінну гру.

Основною особливістю Construct є візуальна система подій, яка дозволяє створювати логіку гри за допомогою умов та дій без необхідності писати код. Це забезпечує низький поріг входу і швидке навчання навіть для тих, хто ніколи не програмував.

Рушій працює прямо в браузері або через настільний застосунок, що дозволяє тестувати гру в режимі реального часу без компіляції. Також він підтримує експорт у HTML5, Android, iOS, Windows, WebGL та інші платформи.

Серед переваг Construct можна виділити:

- система подій, що дозволяє створювати ігри без коду;
- зручний інтерфейс і візуальні інструменти;
- можливість роботи в браузері;
- швидке тестування та прототипування.

Недоліки рушія:

- залежність від вебтехнологій, що впливає на продуктивність на деяких пристроях;
- обмеженість функцій у безкоштовній версії;
- менше можливостей для великих проєктів із складною логікою;
- обмежена підтримка 3D.

Відомі ігри, створені на Construct:

- The Next Penelope ─ інноваційна гоночна аркада.
- Mighty Goose
- Airscape: The Fall of Gravity ─ платформер із змінною гравітацією.

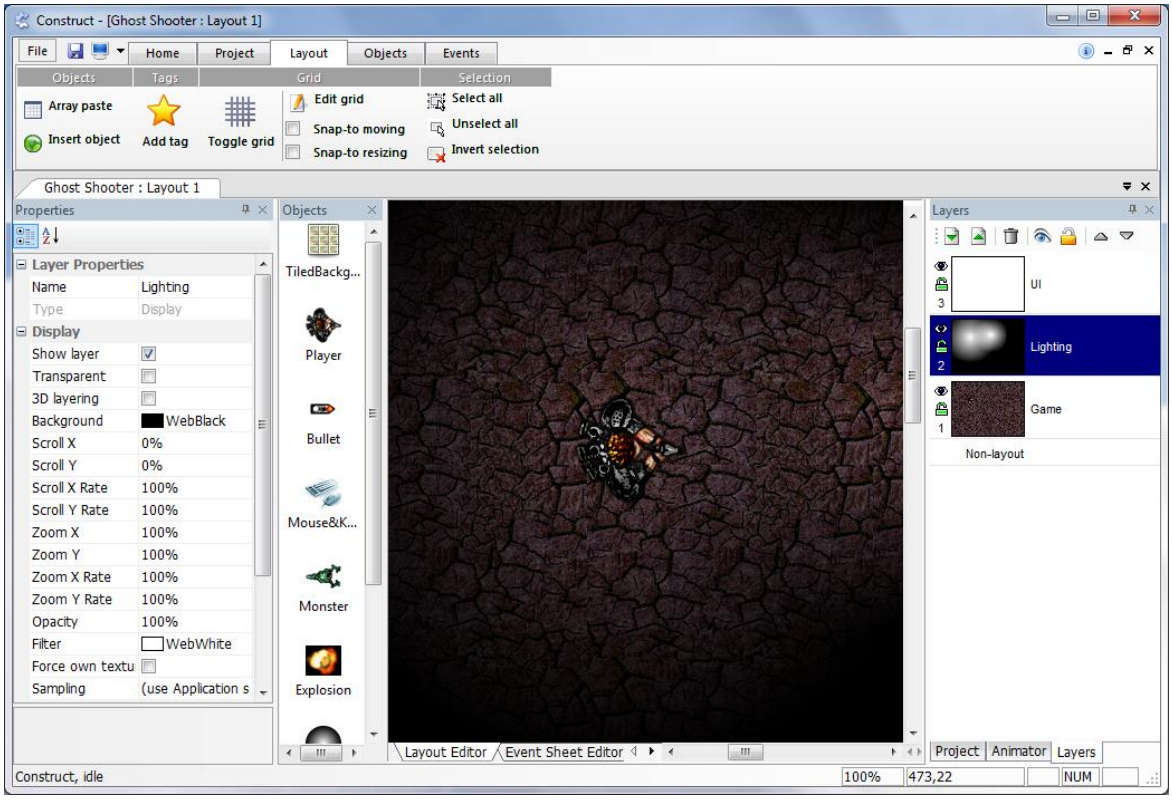


Рисунок 2.6 – Інтерфейс Construct

На основі всіх вище перелічених особливостей, переваг та недоліків кожного ігрового рушія, Godot Engine, є найкращим варіантом для виконання поставленої мети.

## 2.2 Специфікація вимог до програмного забезпечення

### Призначення системи

Ігровий застосунок створений для надання користувачу унікального досліду гри та розповсюдження та популяризації жанру серед звичайного користувача, створення великої спільноти зацікавленої в грі.

### Погодження та стандарти

– Розробка ведеться з використанням ігрового рушія Godot Engine

– Підтримка англійської та української мов

### Межі проекту

– Розробка однокористувацького застосунку для платформи Windows

– Реалізація основного ігрового циклу

– Впровадження системи мета-прогресії
– Не включає розробку мультиплеєра, підтримки інших платформ

**Сфера застосування**

Ігровий застосунок розрахований на чоловіків від 18 до 25 років, які:

– Зацікавлені в комп’ютерних іграх
– Мають базові навички роботи з ПК

**Загальна структура і склад системи**

– Ігровий рушій: Godot Engine
– Формат збереження даних: JSON

**Функції системи**

**Взаємодія з інтерфейсом**

Користувач може взаємодіяти з інтерфейсом користувача, натискати кнопки для початку гри, зміни налаштувань, меню апгрейдів та виходу з гри

**Керування персонажем**

Гравець може керувати персонажем та збирати предмети з ворогів

**Підвищення рівня**

Гравець може обирати серед чотирьох апгрейдів або зброї які він отримує після підвищення рівня.

**Придбання мета апгрейдів**

Гравець може заходити в меню мета апгрейдів та купувати їх за валюту зароблену під час гри

**Зміна налаштувань**

Гравець може відкривати меню налаштувань, змінювати значення в ньому та зберігати або скасовувати зміни

**Вимоги до інформаційного забезпечення**

**Джерела і зміст вхідної інформації**

Ввід гравця, конфігураційний файл, файли рушія

**Нормативно-довідкова інформація**

Внутрішні ID, структури даних для характеристик

**Вимоги до способів організації, збереження та ведення інформації**

Дані поточної сесії в пам’яті, мета-прогресія та налаштування в конфігураційному файлі

**Вимоги до технічного забезпечення**

Мінімальні системні вимоги: Windows 10 64-bit, 4GB RAM? DirectX 11, 4GB Disk Space, Intel Core 2 Duo E6320 (2*1866) or equivalent

**Вимоги до програмного забезпечення**

– Архітектура: на базі Godot Engine

– Системне ПЗ: Windows 10+, необхідні бібліотеки середовища виконання, драйвери

– Мережеве ПЗ: не вимагається

– ПЗ ведення ІБ: не вимагається

– Мова і технологія: Godot Engine, GDScript

**Вимоги до зовнішніх інтерфейсів**

– Інтерфейс користувача: Головне меню, HUD, меню паузи, меню підвищення рівня, екран результатів, меню налаштувань, меню апгрейдів

– Апаратний інтерфейс: Клавіатура, миша, монітор, навушники

– Програмний інтерфейс: API Godot Engine

**Властивості програмного забезпечення**

– Доступність: підтримка базових пристроїв введення

– Супроводжуваність: Модульність, читабельний, коментований код, використання системи контролю версій Git

– Переносимість: Підтримка Windows 10+ 64-bit

– Продуктивність: Стабільна частота кадрів в 60 FPS на мінімальних налаштуваннях, завантаження гри до 5 секунд

– Надійність: Автоматичне збереження даних, резервні копії

– Безпека: Захист файлу збереження від редагування користувачем, відсутність збору даних гравця

## Висновки до розділу 2

У другому розділі досліджено сучасні ігрові рушії, такі як: Unreal Engine, Godot Engine, Unity, BuildBox, GameMaker Studio 2, Construct. Проведено аналіз їх переваг, недоліків та особливостей, спираючись на який було вибрано ігровий рушій Godot Engine для реалізації 2D гри на основі таких переваг:

- висока оптимізацію для 2D-графіки;
- зручний вбудований редактор;
- підтримка власної простої мови програмування GDScript, схожої на Python;
- відкритий вихідний код і відсутність ліцензійних платежів;
- активна спільноту розробників;
- кросплатформеність і гнучкість у налаштуванні експорту проектів.

Окрім технічних переваг, рушій Godot забезпечує короткий цикл розробки завдяки широким можливостям для прототипування та тестування ігрових механік, що є критично важливим у процесі створення проекта.

Сформовано специфікацію вимог до розроблюваного програмного забезпечення, що дозволила чітко визначити призначення ігрового застосунку, його мети, його функціональні та нефункціональні вимог, описано основні ігрові механіки, вимоги до користувацького інтерфейсу, продуктивності на надійності застосунку.

# 3 МОДЕЛЮВАННЯ АРХІТЕКТУРИ ГРИ В ЖАНРІ TOP-DOWN SHOOTER

## 3.1 Створення сценаріїв використання системи

Для початкового визначення вимог до розроблюваного застосунку треба створити декілька сценаріїв використання.

Сценарій використання – це послідовність дій яку буде виконувати користувач для досягнення поставленої мети, їх використовують для визначення які функції повинна виконувати система, її функціональних та нефункціональних вимог.

Таблиця 3.1 – Сценарій зміни налаштувань

| **Use Case Section** | **Description** |
|---|---|
| Use Case Name | Зміна налаштувань гри |
| Scope | System |
| Level | User-goal |
| Primary Actor | Гравець |
| Stakeholders and interests | Гравець: Бажає налаштувати гучність звуків, музики, параметри графіки або керування для комфортної гри. |
| Preconditions | Гра запущена.<br>Гравець знаходиться в Головному меню або викликав Меню паузи під час гри. |
| Success guarantee | Зміни збережені в конфігураційному файлі.<br>Зміни застосовані до поточної сесії |
| Main Success Scenario | 1. Гравець обирає пункт "Налаштування" в Головному меню або Меню паузи.<br>2. Система відображає екран налаштувань з різними категоріями.<br>3. Гравець переходить змінює значення бажаних параметрів.<br>4. Гравець підтверджує зміни, натискаючи кнопку "Зберегти".<br>5. Система зберігає нові значення налаштувань у файлі конфігурації.<br>6. Система застосовує зміни.<br>7. Система повертає гравця до попереднього екрану. |
| Extensions | Якщо користувач на кроці 4 натиснув кнопку «Скасувати», то система не зберігає внесені зміни та повертає гравця до попереднього екрану. |

Кінець таблиці 3.1

| | |
|---|---|
| Special Requirements | Зміни гучності повинні застосовуватись миттєво для зворотного зв'язку.<br>Налаштування мають зберігатися між ігровими сесіями.<br>Інтерфейс налаштувань має бути інтуїтивно зрозумілим. |
| Technology and Data Variations List | Налаштування зберігаються у файлі JSON.<br>Налаштування звуку: загальна гучність, гучність музики, гучність ефектів. |

Таблиця 3.2 – Сценарій купівлі мета апгрейдів

| **Use Case Section** | **Description** |
|---|---|
| Use Case Name | Купівля мета апгрейдів |
| Scope | System |
| Level | User-goal |
| Primary Actor | Гравець |
| Stakeholders and interests | Гравець: Бажає покращити характеристики для майбутніх сесій. |
| Preconditions | Гра запущена.<br>Гравець знаходиться у відповідному меню мета-прогресії.<br>Гравець має достатньо внутрішньоігрової валюти |
| Success guarantee | Обраний гравцем апгрейд придбаний та активований для всіх наступних сесій.<br>Вартість апгрейду списана з балансу валюти гравця.<br>Стан мета-прогресії збережено. |
| Main Success Scenario | 1. Гравець переходить до меню мета-прогресії з Головного меню.<br>2. Система відображає доступні для купівлі апгрейди, їх опис, вартість та поточний баланс валюти гравця.<br>3. Гравець обирає покращення, яке бажає придбати.<br>4. Система відображає вікно підтвердження покупки з деталями та вартістю.<br>5. Гравець підтверджує покупку.<br>6. Система перевіряє наявність достатньої кількості валюти у гравця.<br>7. Система списує вартість апгрейда з балансу гравця.<br>8. Система розблоковує апгрейд, роблячи його доступним для використання в грі.<br>9. Система оновлює інтерфейс, відображаючи новий баланс валюти та статус апгрейда. |
| Extensions | Якщо на кроці 6 в гравця недостатньо валюти система виводить вікно з повідомленням та не вносить змін до балансу користувача.<br>Якщо апгрейд максимального рівня система відопбажає його недоступним для взаємодії |

Кінець таблиці 3.2

| | |
|---|---|
| Special Requirements | Чітке відображення ефекту кожного покращення та його вартості.<br>Баланс монет гравця має бути завжди видимим у магазині.<br>Прогрес мета-покращень має надійно зберігатися. |
| Technology and Data Variations List | Дані мета-прогресії зберігаються в форматі JSON |

Таблиця 3.3 – Сценарій початку гри

| **Use Case Section** | **Description** |
|---|---|
| Use Case Name | Початок гри |
| Scope | System |
| Level | User-goal |
| Primary Actor | Гравець |
| Stakeholders and interests | Гравець: Бажає почати гру |
| Preconditions | Гра запущена.<br>Гравець знаходиться в Головному меню. |
| Success guarantee | Ігровий рівень створений та завантажений<br>Персонаж гравця з'явився на ігровому полі<br>UI коректно відображається<br>Гра починає спавнити ворогів |
| Main Success Scenario | 1. Гравець обирає опцію «Почати гру».<br>2. Система починає завантаження ресурсів рівня.<br>3. Після завершення завантаження система розміщує персонажа, ворогів.<br>4. Система запускає ігровий таймер<br>5. Система активує керування персонажем для гравця.<br>Система відображає HUD з необхідною інформацією. |
| Extensions | Помилка завантаження ресурсів: система відображає повідомлення про помилку та повертає гравця до головного меню |
| Special Requirements | Час завантаження має бути мінімальним.<br>Характеристики персонажа маюсь враховувати придбані мета апгрейди |

Кінець таблиці 3.3

| | |
|---|---|
| Technology and Data Variations List | Процедурна генерація рівня поза межами зору гравця<br>Спавн ворогів залежить від ігрового таймеру |

Таблиця 3.4 – Сценарій процесу гри

| **Use Case Section** | **Description** |
|---|---|
| Use Case Name | Процес гри |
| Scope | System |
| Level | User-goal |
| Primary Actor | Гравець |
| Stakeholders and interests | Гравець: Бажає перемогти, заробити валюту для апгрейдів |
| Preconditions | Гравець почав гру<br>Гравець керує персонажем<br>На карті присутні вороги |
| Success guarantee | Гравець керує персонажем<br>Система автоматично атакує ворогів та генерує нових<br>. Вороги знищуються, залишаючи після себе XP або предмети |
| Main Success Scenario | 1. Гравець рухається натискаючи клавіши керування<br>2. Система автоматично атакує ворогів<br>3. Гравець підбирає XP<br>4. При досягнені певної кількості XP гравець підвищує рівень<br>5. Система ставить гру на паузу та викликає меню з 4 варіантами нових рівнів для існуючої зброї або предметів чи для отримання нових предметів чи зброї<br>6. Гравець обирає варіант<br>7. Система відновлює гру<br>. Гравець продовжує рух та бій повертаючись до кроку 1 |
| Extensions | . Здоров’я гравця досягає 0 – Система виводить повідомлення про поразку та повертає гравця до головного меню |
| Special Requirements | Гра не повинна плавно працювати при великій кількості ворогів<br>. Можливі апгрейди повинні випадати випадково |
| Technology and Data Variations List | Процедурна генерація рівня поза межами зору гравця<br>. Спавн ворогів залежить від ігрового таймеру |

Таблиця 3.5 – Сценарій завершення гри

| **Use Case Section** | **Description** |
|---|---|
| Use Case Name | Заверження гри |
| Scope | System |
| Level | User-goal |
| Primary Actor | Гравець |
| Stakeholders and interests | Гравець: Вижити протягом 30 хвилин |
| Preconditions | . Ігровий процес активний<br>. Гравець живий<br>. Ігровий таймер показує час менший за 30:00 |
| Success guarantee | . Таймер досягає 30:00<br>. Система генерує непереможного противника<br>. Гравець вмирає |
| Main Success Scenario | 1. Гравець продовжує грати<br>2. Ігровий таймер досягає значення 30:00<br>3. Система генерує непереможного ворога<br>4. Ворог вбиває гравця<br>5. Система зупиняє ігровий процес та повертає гравця до головного меню |

На основі створених сценаріїв використання буде розроблено інтерфейс системи та основі механіки гри.

## 3.2 Розробка діаграм класів та використання системи

UML-діаграми є зручним способом змоделювати функціонал застосунку на початку розробки проекту для більш детального уявлення його функціоналу та обмежень. Серед усіх UML-діаграм слід виділити діаграми класів та використання.

Діаграма класів описує класи які будуть наявні в застосунку, їх змінні та методи які ці класи будуть використовувати.

Діаграма використання описує можливості кожного актора представленого в системі

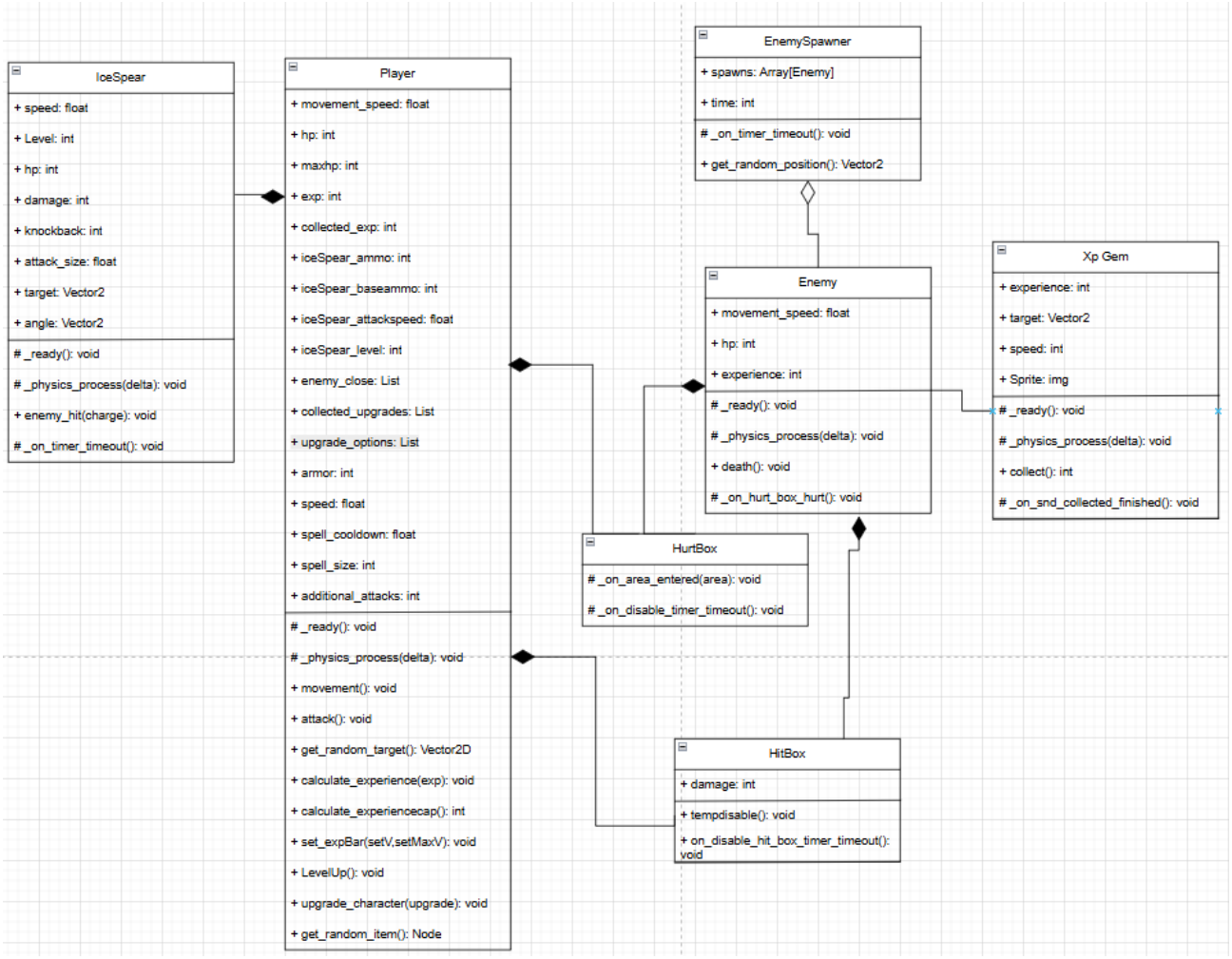


Рисунок 3.1 – Діаграма класів

Серед наведених на діаграмі класів зв’язків можна виділити три основних:

- асоціація: відношення класу Enemy до класу Xp Gem;
- агрегація: відношення між EnemySpawner та Enemy при якому Enemy є частиною EnemySpawner і при видаленні класу Enemy клас EnemySpawner не зникає;
- композиція: відношення між IceSpear, HitBox, HurtBox та Player при якому при винищені основного класу Player всі пов’язані класи теж будуть знищені.

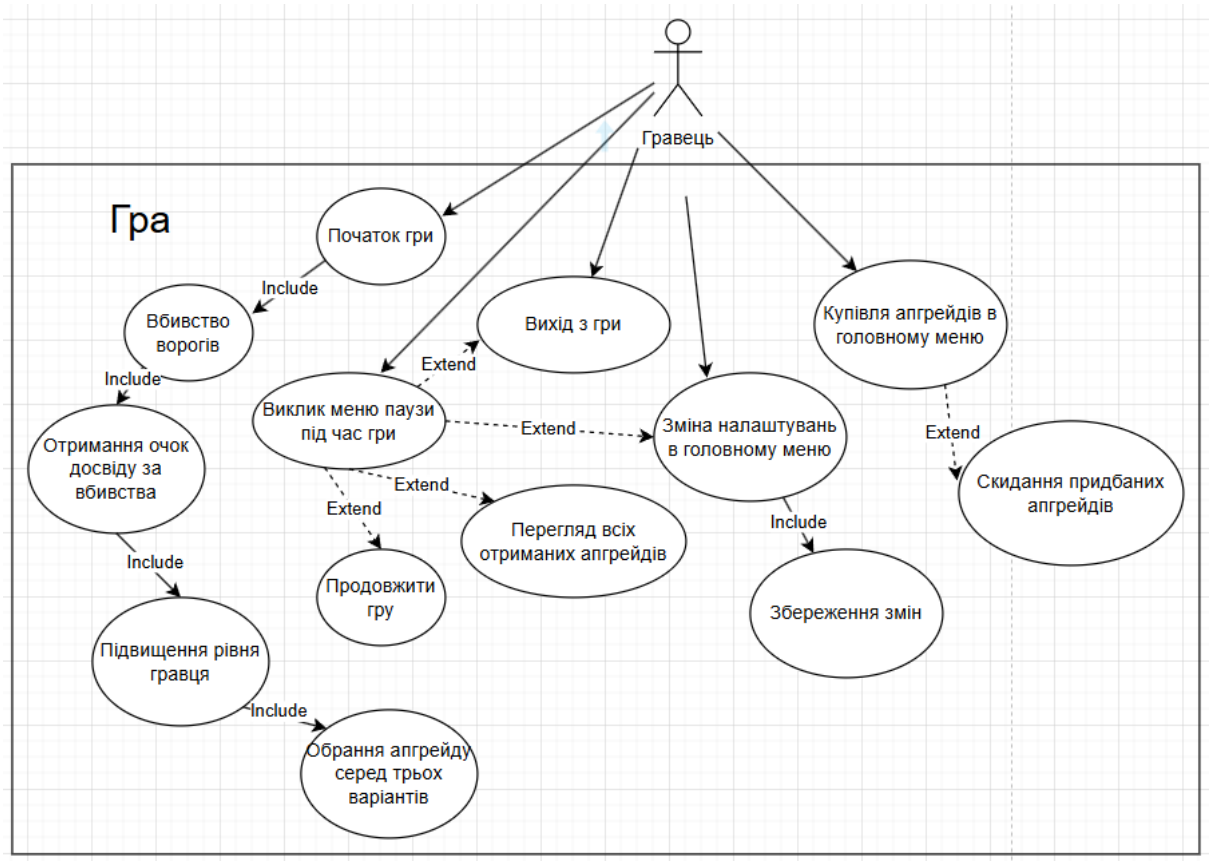


Рисунок 3.2 – Діаграма варіантів використання

На представленій діаграмі використання показано всі можливі дії з системою які користувач може виконувати, наявні два основних типи зв’язків:

– include: додаткова дія є обов’язковою;

– extend: дія яка не є обов'язковою, або декілька варіантів.

## 3.3 Моделювання інтерфейсу користувача для гри в жанрі top-down shooter

UI або Інтерфейс користувача є критично важливим компонентом будь-якої відеогри, виступаючи мостом між гравцем та ігровим світом. Він охоплює всі візуальні елементи, за допомогою яких гравець отримує інформацію та взаємодіє з грою: меню, індикатори здоров'я та ресурсів, карти, інвентар, діалогові вікна, кнопки та інші елементи керування. Ефективний UI не лише надає необхідну інформацію вчасно та зрозуміло, але й сприяє зануренню в ігровий процес, підкреслює атмосферу та стиль гри, і, що найголовніше, робить взаємодію

інтуїтивною та приємною. Погано спроектований інтерфейс, навпаки, може викликати роздратування, ускладнювати геймплей та руйнувати враження, незалежно від якості інших аспектів гри. Тому моделювання та ретельне проектування UI є невід'ємною частиною процесу розробки.

**Головне меню**

Головне меню як основа кожної гри слугує початковою точкою взаємодії гравця з грою до якого гравець потрапляє одразу після запуску гри. В головному меню відбувається вся навігація між іншими вікнами або функціями, такими як:

- Початок гри;
- Меню налаштувань;
- Меню апгрейдів;
- Список авторів;
- Кнопка виходу з гри.

Основними вимогами до головного меню є інтуїтивна зрозумілість, анімований задній фон та приємна, спокійна музика яка підкреслить атмосферу та стиль гри та допоможе гравцю глибше зануритись в ігровий процес.

Також, дуже багато розробників додає до головного меню новини щодо їх нових ігор, оновлень, тощо.


Рисунок 3.3 – Макет головного меню

**Меню підвищення рівня**

Цей екран активується безпосередньо під час ігрової сесії, коли персонаж гравця накопичує достатньо XP для отримання нового рівня. Його функція полягає в наданні гравцю можливість обрати тимчасові апгрейди або нові здібності, які будуть діяти лише протягом поточної ігрової сесії. Зазвичай пропонується обмежений випадковий вибір з кількох варіантів, що є ключовою механікою для жанрів roguelike та survivors-like, дозволяючи створювати унікальні комбінації зброї або предметів під час кожного забігу.


Рисунок 3.4 – Макет меню підвищення рівня

**Меню мета-апгрейдів**

На відміну від екрану підвищення рівня, це меню зазвичай доступне поза основним ігровим процесом, найчастіше з головного меню. Воно призначене для системи довгострокової прогресії. Тут гравець може витрачати ресурси зароблені під час ігрових сесій, на розблокування постійних апгрейдів, які впливатимуть на всі наступні ігрові сесії. Це може бути підвищення базових характеристик відкриття нових персонажів, розблокування нової зброї чи здібностей, або інші глобальні бонуси. Мета цього меню – мотивувати гравця до повторних проходжень та давати відчуття постійного розвитку.


Рисунок 3.5 – Макет меню мета-апгрейдів

**Меню налаштувань**

Основне призначення цього меню – надати гравцеві контроль над різними параметрами гри. Тут користувач може змінювати налаштування графіки, звуку, управління, мови інтерфейсу та інші опції, що впливають на комфорт та доступність гри. Наявність добре продуманого меню налаштувань є важливою для забезпечення позитивного користувацького досвіду на різних конфігураціях обладнання та для гравців з різними вподобаннями.


Рисунок 3.6 – Макет меню налаштувань

**Екран вибору персонажа**

Цей екран зазвичай з'являється перед початком нової ігрової сесії і дозволяє гравцеві обрати персонажа, яким він буде грати. Його функція – надати інформацію про доступних персонажів, їхні унікальні стартові характеристики, здібності або особливості. Гравець може порівняти різні варіанти та обрати того, чий стиль гри йому найбільше підходить. У деяких іграх цей екран також може слугувати для перегляду вже розблокованих персонажів та їхніх параметрів.

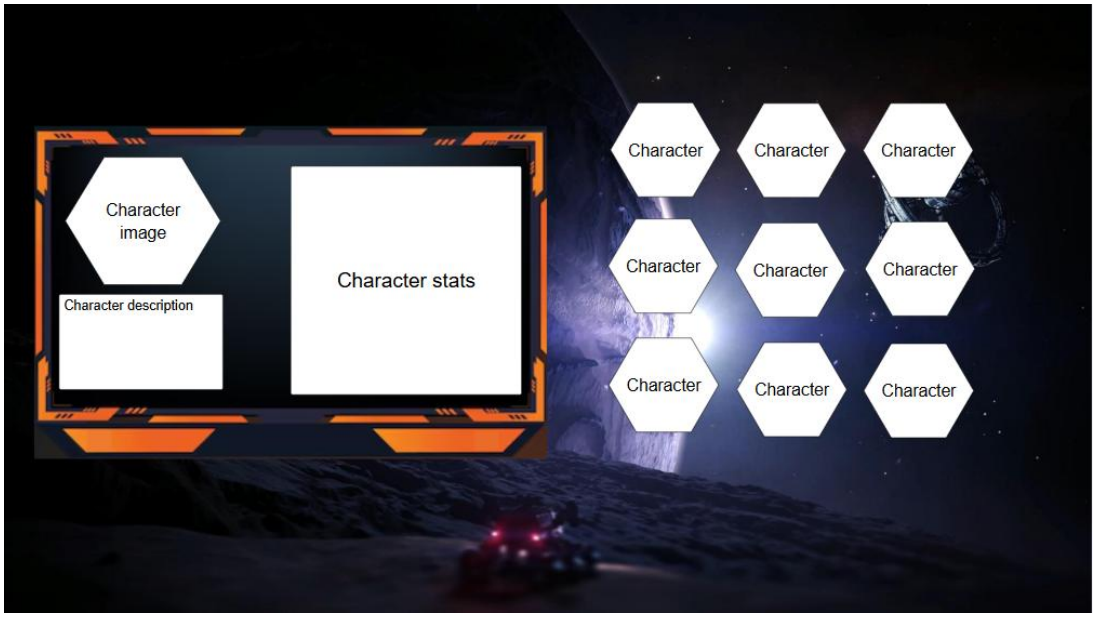


Рисунок 3.7 – Макет меню вибору персонажа

Описані екрани інтерфейсу користувача відіграють ключову роль у формуванні загального враження від гри та забезпеченні комфортної взаємодії гравця з ігровим світом. Головне меню виступає стартовою точкою, з якої починається досвід користувача, тому його інтуїтивність, візуальна привабливість і музичний супровід значною мірою впливають на перше враження про гру. Меню підвищення рівня вводить елемент тактичного вибору під час бою, дозволяючи гравцю підлаштовувати стиль гри під конкретну ситуацію. Меню мета-апгрейдів додає глибини й довгострокової мотивації, забезпечуючи поступове зростання сили персонажа і відкриваючи нові можливості.

Меню налаштувань є необхідним для адаптації гри під різні технічні умови та індивідуальні вподобання гравців, що підвищує загальну доступність гри. Екран

вибору персонажа дає змогу ознайомитися з унікальними особливостями ігрових героїв, сприяючи різноманітності геймплею та підвищенню інтересу до повторних ігрових сесій. У сукупності всі ці елементи інтерфейсу не лише виконують технічні функції, а й підкреслюють атмосферу та стиль гри, роблячи користувацький досвід цілісним, продуманим і захоплюючим.

## Висновки до розділу 3

У третьому розділі було розроблено п’ять детальних сценаріїв використання системи, які описують дії користувача під час взаємодії з ігровим застосунком. Кожен сценарій охоплює окрему гілку взаємодії, включаючи запуск гри, вибір персонажа, проходження рівнів, підвищення рівня, управління налаштуваннями та прогресом гравця. Ці сценарії слугують основою для подальшої розробки графічного інтерфейсу користувача та реалізації основних ігрових механік.

На основі визначених сценаріїв було створено діаграми використання, які наочно демонструють ключові функції, доступні користувачу, та взаємозв’язки між гравцем і підсистемами гри. Це дозволило краще структурувати архітектуру застосунку та визначити функціональні вимоги до кожного з модулів.

Також було розроблено діаграми класів, що відображають логічну структуру програми, включаючи класи, їх атрибути, методи та зв’язки між ними. Це сприяло більш чіткому розумінню того, як дані будуть організовані у грі та яким чином буде реалізовано основну логіку застосунку.

Розроблено мокапи інтерфейсів головного меню, екрану підвищення рівня, меню мета-прогресії, меню налаштувань та вікна вибору персонажа для ігрового застосунку. Що дозволило краще зрозуміти взаємодію користувача з грою.

# 4 КОДУВАННЯ ГРИ В ЖАНРІ TOP-DOWN SHOOTER

## 4.1 Створення головного героя

Кожна гра починається та закінчується ігровим персонажем яким керує гравець та способами якими гравець може взаємодіяти з ігровим світом, саме через це створенню головного персонажа слід приділити особливу увагу.

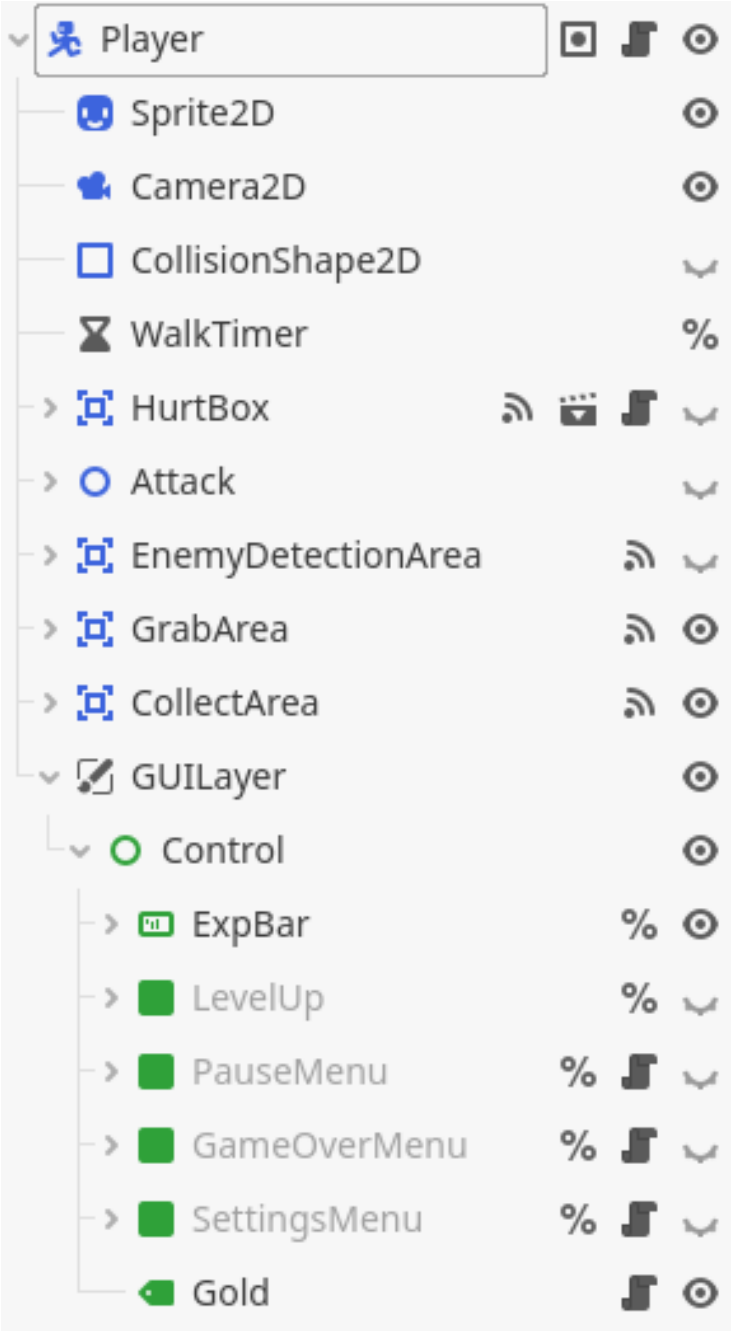


Рисунок 4.1 – Список вузлів гравця

В Godot Engine вся система складається з вузлів які виконують певні функції, серед них є вузли зеленого кольору для роботи з 2D-сценами, червоного кольору для роботи з 3D-сценами, зелені для створення користувацького інтерфейсу та білі як базові.

**Sprite2D** – вузол для задання спрайту або картинки ігрового персонажу.

**Camera2D** – вузол камери є дочірнім елементом вузла Player і через це буде постійно рухатись за гравцем.

**CollisionShape2D** – потрібен для взаємодії гравця з ворогами та предметами. Без колізії спрайт персонажа не може взаємодіяти з ворогами та предметами, що призводить до накладання цих спрайтів один на одного.

**WalkTimer** – потрібен для анімації головного героя, кожні 0.2 секунди фрейм анімації змінюється

**HurtBox** – Виконує схожу функцію, що і CollisionShape2D, але на відміну від нього реагує тільки на HitBox і при зіткненні яких буде нанесено пошкодження головному герою або ворогу. Має DisableTimer який вимикає отримання пошкоджень на 0.5 секунд і запускається при отриманні пошкоджень слугуючи таймером невразливості.

**Attack** – є вузлом який відповідає за всю зброю гравця та має два таймера для кожної зброї які відповідають за проміжки між атаками.

**EnemyDetectionArea** – записує кожного ворога який є в цій області для подальшого вибору випадкового ворога з цього списку для атаки.

**GrabArea** та **CollectArea** – потрібні для взаємодіє з предметами та очками досвіду які випадають з ворогів. Предмети які увійшли в GrabArea починають свій рух до гравця, а в CollectArea вони підбираються гравцем та зникають з карти.

**GUILayer** – вузол графічного інтерфейсу користувача який включає в себе полосу рівня, рівень та меню вибору апгрейду після підвищення рівня.

```
func _ready() -> void:
    attack()
    set_expBar(exp, calculate_experiencecap())
    Persistance.gain_bonus_stats(self)


func _physics_process(delta: float) -> void:
    movement()
```

Рисунок 4.2 – Основні функції Godot Engine

В Godot Engine є дві основні функції які використовуються в кожному вузлі: _ready та _physics_process. _ ready виконується при завантажені вузла, а _physics_process кожний фрейм або 60 разів на секунду.

В випадку з головним гравцем при завантажені вузла Player буде виконано функції attack() яка відповідає за запуск таймерів зброї та set_expBar яка заповнює полосу нового рівня. Функція movement() в _physics_process перевіряє які клавіши клавіатури натискає гравець та рухає його у відповідний напрямок.

```
func movement():
    var x_mov = Input.get_action_strength("right") - Input.get_action_strength("left")
    var y_mov = Input.get_action_strength("down") - Input.get_action_strength("up")
    var mov = Vector2(x_mov,y_mov)
    if mov.x > 0:
        sprite.flip_h = true
    elif mov.x < 0:
        sprite.flip_h = false
    if mov != Vector2.ZERO:
        if WalkTimer.is_stopped():
            if sprite.frame >= sprite.hframes - 1:
                sprite.frame = 0
            else:
                sprite.frame += 1
            WalkTimer.start()
    velocity = mov.normalized()*movement_speed
    move_and_slide()
```

Рисунок 4.3 – Функція movement()

move_and_slide() є вбудованою функцією Godot Engine для переміщення елементів.

```
func attack():
    if iceSpear_level > 0:
        iceSpearTimer.wait_time = iceSpear_attackspeed * (1-spell_cooldown)
        if iceSpearTimer.is_stopped():
            iceSpearTimer.start()
```

Рисунок 4.4 – Функція attack()

Функція attack() відповідає за перевірку наявності зброї в гравця та запускає таймери кожної зброї.

```
func _on_ice_spear_timer_timeout() -> void:
    iceSpear_ammo += iceSpear_baseammo + additional_attacks
    iceSpearAttackTimer.start()


func _on_ice_spear_attack_timer_timeout() -> void:
    if iceSpear_ammo > 0:
        var iceSpear_attack = iceSpear.instantiate()
        iceSpear_attack.position = position
        iceSpear_attack.target = get_random_target()
        iceSpear_attack.level = iceSpear_level
        add_child(iceSpear_attack)
        iceSpear_ammo -= 1
        if iceSpear_ammo > 0:
            iceSpearAttackTimer.start()
        else:
            iceSpearAttackTimer.stop()
```

Рисунок 4.5 – Таймери зброї

_on_ice_spear_timer_timeout() відповідає за швидкість атаки зброї, додає кількість снарядів які мають бути запущені до змінної iceSpear_ammo та запускає iceSpearAttackTimer.

_on_ice_spear_attack_timer_timeout() відповідає за проміжок між снарядами та має функцію get_random_target() яка обирає одного ворога зі списку тих хто увійшов до EnemyDetectionArea

### 4.2 Створення ворогів та спавнеру ворогів

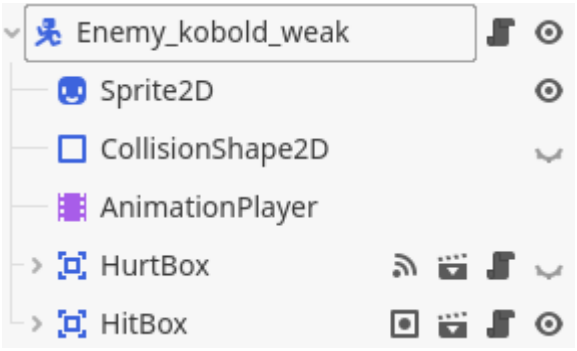


Рисунок 4.6 – Список вузлів ворогів

Кожен ворог складається з картинки або спрайту, форми колізії, вузла анімації та хітбоксу для нанесення пошкоджень та хертбоксу для отримання пошкоджень.

```
func _physics_process(_delta: float) -> void:
    var direction = global_position.direction_to(player.global_position)
    velocity = direction*movement_speed
    move_and_slide()

    if direction.x > 0.1:
        sprite.flip_h = true
    elif direction.x < -0.1:
        sprite.flip_h = false

func death():
    var new_gem = exp_gem.instantiate()
    new_gem.global_position = global_position
    new_gem.experience = experience
    loot_base.call_deferred("add_child", new_gem)
    queue_free()

func _on_hurt_box_hurt(damage: Variant) -> void:
    hp -= damage
    if hp <= 0:
        death()
```

Рисунок 4.7 – Список вузлів ворогів

В функції _physics_process ворог отримує глобальну позицію головного героя та рухається до неї. В функції death(), яка викликається коли здоров'я ворога стає менше нуля, створюється предмет яких потім може бути підібраний гравцем.

```
func _on_timer_timeout() -> void:
    time += 1
    var enemy_spawns = spawns
    for i in enemy_spawns:
        if time >= i.time_start and time <= i.time_end:
            if i.spawn_delay_counter < i.enmy_spawn_delay:
                i.spawn_delay_counter += 1
            else:
                i.spawn_delay_counter =0
                var new_enemy = i.enemy
                var counter =0
                while counter <= i.enemy_num:
                    var enemy_spawn = new_enemy.instantiate()
                    enemy_spawn.global_position = get_random_position()
                    add_child(enemy_spawn)
                    counter += 1
```

Рисунок 4.8 – Код для EnemySpawner

В вузлі EnemySpawner є дві функції: _on_timer_timeout() та get_random_position(). _on_timer_timeout() бере список ворогів та створює їх на випадковій позиції поза межами екрану гравця, а get_random_position() знаходить цю випадкову позицію.


Рисунок 4.9 – Список ворогів в Enemy Spawner

Time Start та Time End відповідають за початок та кінець появи певних ворогів, є основним методом для створення різних хвиль ворогів

Enemy містить ворога якого буде створено

Enemy Num – кількість ворогів які будуть створені за один цикл

Enemy Spawn Delay – проміжок часу в секундах між створенням ворогів

### 4.3 Створення системи підвищення рівня та списку апгрейдів

Система підвищення рівня є дуже важливою частиною будь якої гри з жанром rpg, вона надає відчуття розвитку та покращує користувацький досвід. Для реалізації цієї системи в вузлі Player є такі методи:

```
func calculate_experience(gem_exp):
    var exp_required = calculate_experiencecap()
    collected_exp += gem_exp
    if exp + collected_exp >= exp_required:
        collected_exp -= exp_required-exp
        exp_level += 1
        exp = 0
        exp_required = calculate_experiencecap()
        LevelUp()
    else:
        exp += collected_exp
        collected_exp = 0
    set_expBar(exp, exp_required)
```

Рисунок 4.10 – Calculate Exp

Метод calculate_experience() потрібен для розрахунку отриманих очок досвіду, підвищення рівня гравця при досягнені потрібного значення та оновлення графічного відображення отриманих очок досвіду.

```
func LevelUp():
    soundLevelUp.play()
    Lvl_label.text = str("Level: ", exp_level)
    var tween = levelPanel.create_tween()
    tween.tween_property(levelPanel,"position",Vector
    tween.play()
    levelPanel.visible = true
    var options = 0
    var optionsmax = 3
    while options < optionsmax:
        var option_choice = itemOtions.instantiate()
        option_choice.item = get_random_item()
        upgradeOptions.add_child(option_choice)
        options += 1
    get_tree().paused = true
```

Рисунок 4.11 – Calculate Exp

Метод LevelUp() викликається при досягнені потрібного значення очок досвіду та відповідає за появу меню з вибором апгрейда. Всередині використовується функція get_random_item() для отримання випадкового предмета або зброї зі списку усіх доступних апгрейдів.

```
func get_random_item():
    var dblist = []
    for i in UpgradeDb.UPGRADE:
        if i in collected_upgrades:
            pass
        elif i in upgrade_options:
            pass
        elif UpgradeDb.UPGRADE[i]["type"] == "item":
            pass
        elif UpgradeDb.UPGRADE[i]["prerequisite"].size()> 0:
            for n in UpgradeDb.UPGRADE[i]["prerequisite"]:
                if not n in collected_upgrades:
                    pass
                else:
                    dblist.append(i)
        else:
            dblist.append(i)
    if dblist.size() > 0:
        var randomItem = dblist.pick_random()
        upgrade_options.append(randomItem)
        return randomItem
    else:
        return null
```

Рисунок 4.12 – Метод GetRandomItem

## 4.4 Створення головного меню, меню паузи, поразки та налаштувань

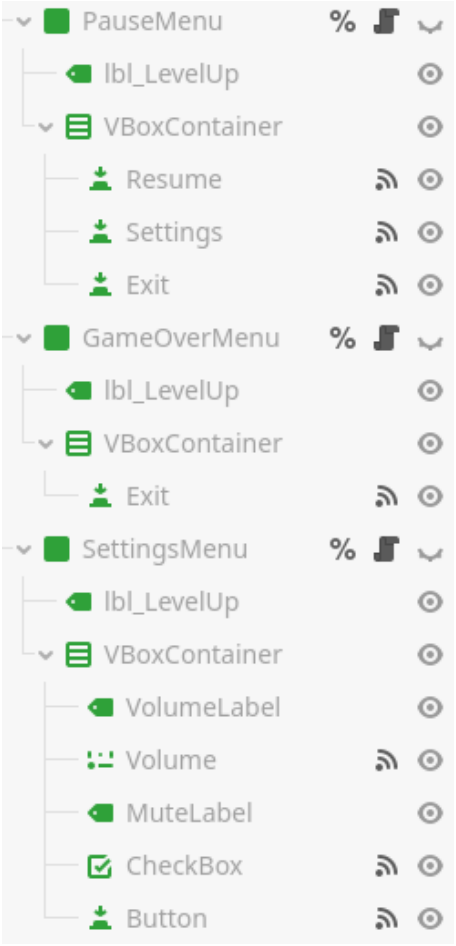


Рисунок 4.13 – Список компонентів в кожному меню

Для початку реалізації меню паузи потрібно зайти в Проект->Налаштування проекту->Карта введення-> додати нову дію (ESC). Додавши нову дію під назвою «Pause» на клавішу ESC ми зможемо використати її в коді та реалізувати меню паузи.

```
func _process(delta: float) -> void:
    testEsc()

func testEsc():
    if Input.is_action_just_pressed("Pause") and !get_tree().paused:
        PauseMenu.show()
        get_tree().paused = true
    elif Input.is_action_just_pressed("Pause") and get_tree().paused:
        PauseMenu.hide()
        get_tree().paused = false
```

Рисунок 4.14 – Список компонентів в кожному меню

Функція _process є вбудованою функцією в Godot Engine яка викликається кожний кадр і виконує функцію testEsc(), За допомогою Input.is_action_just_pressed("Pause") ми перевіряємо чи натиснув гравець клавішу ESC і якщо клавіша була натиснута і гра не стоїть на паузі, то викликається меню паузи та за допомогою get_tree().paused = true гра ставиться на паузу.

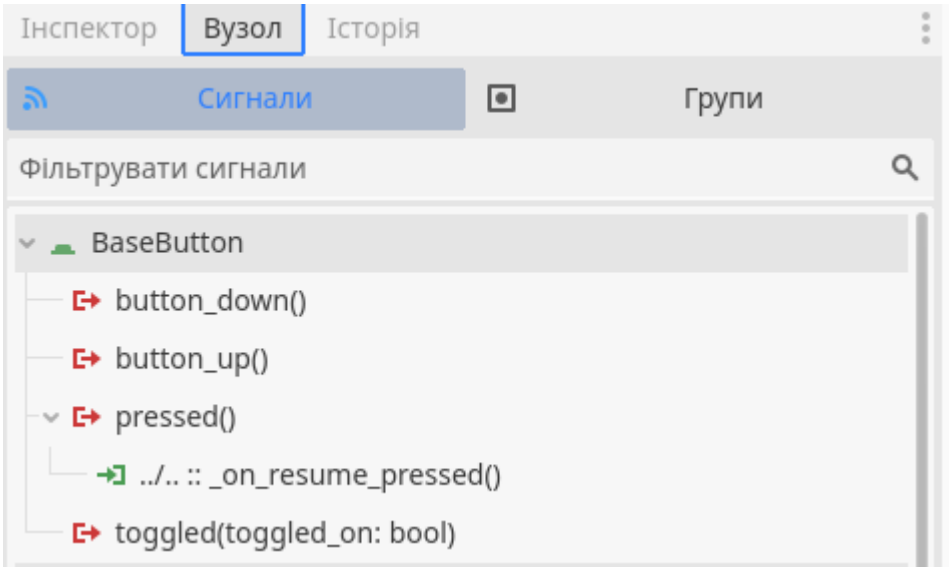


Рисунок 4.15 – Створення сигналів

Створивши скрипт в PauseMenu та виділивши одну з кнопок для взаємодії в меню паузи треба в правому меню вибрати Вузол->pressed(). Створені функції потрібні для роботи кнопок.

```
func _on_resume_pressed() -> void:
    PauseMenu.hide()
    get_tree().paused = false
func _on_settings_pressed() -> void:
    SettingsMenu.show()
    PauseMenu.hide()
func _on_exit_pressed() -> void:
    get_tree().paused = false
    get_tree().change_scene_to_file("res://Utility/menu.tscn")
```

Рисунок 4.16 – Дії, що відбуваються при натисканні кнопок

Функція _on_resume_pressed() приховує меню паузи та поновлю

Функція _on_settings_pressed() показує меню налаштувань

Функція _on_exit_pressed() поновлює гру та змінює сцену на головне меню

```
func _on_volume_value_changed(value: float) -> void:
    AudioServer.set_bus_volume_db(0,value)

func _on_check_box_toggled(toggled_on: bool) -> void:
    AudioServer.set_bus_mute(0,toggled_on)

func _on_button_pressed() -> void:
    main.SettingsMenu.hide()
    main.PauseMenu.show()
```

Рисунок 4.17 – Функції меню налаштувань

В меню налаштувань є один слайдер для зміни гучності та один checkbox для прибирання звуку. Використавши сигнали для взаємодії з цими елементами отримано дві функції _on_volume_value_changed та _on_check_box_toggle в яких за допомогою AudioServer.set_bus_volume_db змінюється гучність, а AudioServer.set_bus_mute прибирає звук.


Рисунок 4.18 – Головне меню

Головне меню виконує функції навігації гравця між різними сценами. Створивши сигнали для взаємодії з кнопка та використавши код get_tree().change_scene_to_file("res://World/world.tscn") для переходу між сценами та get_tree().quit() для виходу з гри було реалізовано навігацію гравця.

## 4.5 Створення меню апгрейдів

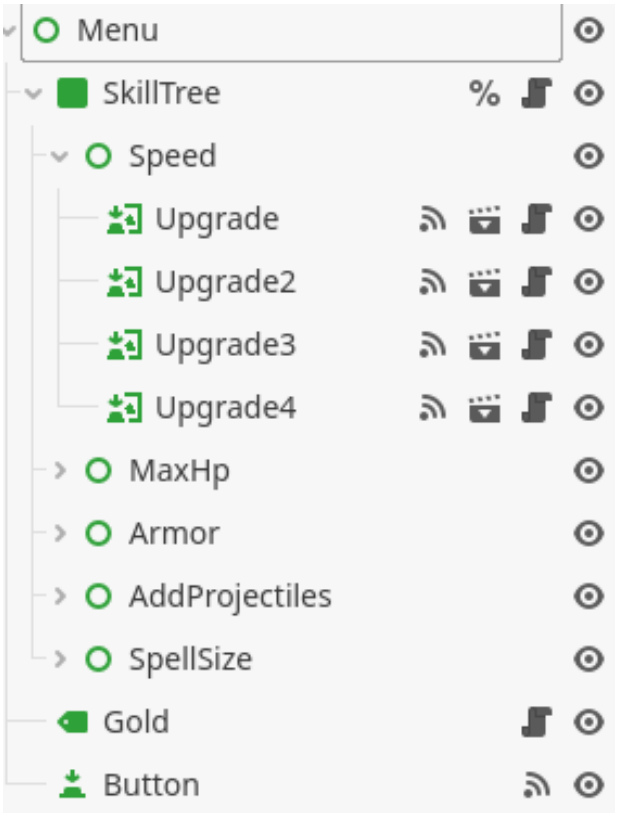


Рисунок 4.19 – Список компонентів в меню апгрейдів

SkillTree відповідає за обробку куплених апгрейдів, їх збереження та підвищення характеристик на їх основі.

Вузли Speed, MaxHp, Armor, AddProjectiles та SpellSize є окремими гілками які відповідають за свою певну характеристику, в середині цих вузлів є чотири кнопки при натисканні на які буде купуватись апгрейд та зніматись золото.

Вузол Gold відповідає за відображення загальної кількості валюту в гравця, а Button при натисканні повертає гравця до головного меню.

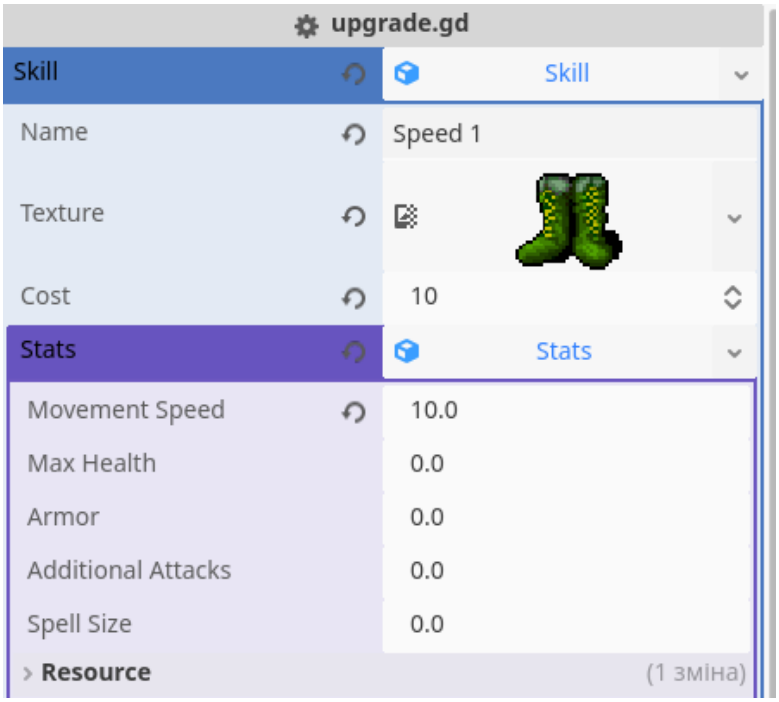


Рисунок 4.20 – Поля апгрейда

В кожного апгрейда є назва, текстура, вартість та клас характеристик в якому виставлені всі характеристики які дає апгрейд, все це дозволяє змінювати текстуру апгрейду на кожному етапі для кращого розуміння прогресу гри, а також дозволяє створювати різні гілки розвитку.

```
func _ready() -> void:
    load_data()

func save_data():
    config.save(PATH)

func set_data():
    config.set_value("Player","gold",gold)
    config.set_value("Player","skill_tree",skill_tree)

func set_and_save():
    set_data()
    save_data()

func load_data():
    if config.load(PATH) != OK:
        set_and_save()
    gold = config.get_value("Player","gold",1000)
    skill_tree = config.get_value("Player","skill_tree",[])
```

Рисунок 4.21 – Функції вузла save_data

Вузол save_data використовується для зберігання даних між ігровими сесіями, всередині вузла є дві змінні: gold для зберігання валюти та skill_tree для зберігання списку куплених апгрейдів.

Функція _ready викликає load_data() яка перевіряє файл конфігу player_data.cfg на наявність після чого отримує дані з нього за ключем Player з поля gold та skill_tree, або повертає дефолтне значення. Якщо з конфігураційним файлом виникли помилки, функція set_and_save() створює новий файл та записує в нього вказані значення.

```
func _ready() -> void:
    load_skill_tree()

func set_skill_tree():
    skill_tree = []
    for each_branch in get_children():
        var branch = []
        for upgrade in each_branch.get_children():
            branch.append(upgrade.enabled)
        skill_tree.append(branch)
    SaveData.skill_tree = skill_tree
    SaveData.set_and_save()

func load_skill_tree():
    if SaveData.skill_tree == []:
        set_skill_tree()

    skill_tree = SaveData.skill_tree
    for branch in get_children():
        for upgrade in branch.get_children():
            upgrade.enabled = skill_tree[branch.get_index()][upgrade.get_index
    get_total_stats()
```

Рисунок 4.22 – Функції вузла skill_tree

В вузлі skill_tree дві основних функції: set_skill_tree та load_skill_tree. В функції set_skill_tree за допомогою подвійного цикла відбувається проходження по

всіх дочірніх елементах, їх запис в список та збереження цих даних в конфігураційному файлі, ця функція потрібна для встановлення базових значень для древа апгрейдів. Функція load_skill_tree() зчитує дані з конфігураційного файлу та позначає апгрейди як куплені, після чого викликає функцію get_total_stats() яка проходить по всім гілкам та рахує всі характеристики куплених апгрейдів та додає їх гравцю

```
func is_upgradable() -> bool:
    if get_index() == 0:
        return true
    elif get_index() > 0:
        if get_parent().get_child(get_index() - 1).enabled == true:
            return true
        else:
            return false
    return false

func _on_pressed() -> void:
    if skill.cost <= SaveData.gold and is_upgradable() and not enabled:
        SaveData.gold -= skill.cost
        enabled = true
        get_parent().get_parent().set_skill_tree()
        get_parent().get_parent().get_total_stats()
```

Рисунок 4.23 – Функції вузла upgrade

Функція is_upgradable() відповідає за можливість купити апгрейд, апгрейд можна купити якщо це перший рівень в гілці або всі минулі рівні апгрейда були куплені, в інших випадках апгрейд купити неможливо. Функція _on_pressed() викликається коли гравець натискає на апгрейд та визначає чи достатньо коштів у гравця та чи можна купити апгрейд, після чого вичитає потрачені кошти та викликає функції set_skill_tree() та get_total_stats() для перерахунку куплених апгрейдів та загальних характеристик отриманих від всіх апгрейдів.

## 4.6 Створення керівництва користувача

**Зміна налаштувань гри**:

Для зміни налаштувань гри потрібно:

1. перебуваючи в головному меню або викликавши меню паузи під час гри, виберіть пункт «Налаштування»;
2. у вікні налаштувань будуть доступні наступні категорії:
   - **звук**: загальна гучність, гучність музики, гучність ефектів;
   - **графіка**: режим вікна, роздільна здатність, обмеження FPS;
   - **керування**: призначення клавіш руху, атаки.
3. змініть бажані параметри;
4. для збереження змін натисніть кнопку «Назад»;


Рисунок 4.24 – Зміна налаштувань

**Купівля апгрейдів**:

Щоб купити апгрейди для головного героя потрібно:

1. у головному меню оберіть пункт «Апгрейди»;
2. перед вами з’явиться список доступних апгрейдів з описами, вартістю та поточним балансом внутрішньоігрової валюти;

3. виберіть бажане покращення, після чого система покаже вікно підтвердження;
4. натисніть «Підтвердити»;
5. апгрейди, що досягли максимального рівня, будуть відображатися як недоступні.


Рисунок 4.25 – Купівля апгрейдів

**Початок гри**:

Щоб розпочати нову ігрову сесію потрібно:

1. у головному меню натисніть кнопку «Почати гру»;
2. система почне завантаження рівня;
3. після завершення завантаження на екрані з’явиться персонаж гравця, а також вороги;
4. увімкнеться ігровий таймер, керування стане активним;
5. інтерфейс HUD відобразить інформацію про здоров’я, час.


Рисунок 4.26 – Початок гри

**Процес гри**:

Основні дії які гравець може виконувати під час гри:

1. гравець керує персонажем за допомогою клавіш WASD;
2. система автоматично атакує ворогів у радіусі дії;
3. гравець підбирає XP, що випадає з ворогів;
4. при накопиченні достатньої кількості XP гравець отримує новий рівень;
5. гра призупиняється, і з’являється меню вибору одного з трьох варіантів;
6. гравець обирає один з трьох варіантів натискаючи на них;
7. гра відновлюється;
8. процес повторюється до завершення гри або поразки.


Рисунок 4.27 – Підвищення рівня

**Завершення гри:**

Гра завершується в трьох випадках:

1. Поразка
   - якщо здоров’я гравця досягає нуля з’являється повідомлення про поразку, і система повертає гравця до головного меню.
2. Перемога
   - таймер гри досягає 30 хвилин;

- гра створює непереможного ворога;
- гравець вмирає і система повертає гравця до головного меню.

3. Вихід гравця з гри
   - гравець викликає меню паузи натискаючи ESC;
   - гравець натискає клавішу «Вийти з гри»;
   - гра повертає гравця до головного меню.


Рисунок 4.28 – Завершення гри

**Взаємодія з меню паузи:**

1. під час гри натисніть клавішу ESC;
2. гра зупиниться і перед гравцем з’явиться меню паузи;
3. в меню паузи є чотири основні кнопки:
   - продовжити гру;
   - зміна налаштувань;
   - вихід з гри.
4. для продовження гри натисніть кнопку «Продовжити гру»;
5. для зміни налаштувань натисніть кнопку «зміна налаштувань», зробіть потрібні зміни та натисніть кнопку «Назад»;
6. для виходу з гри натисніть кнопку «Вихід з гри».

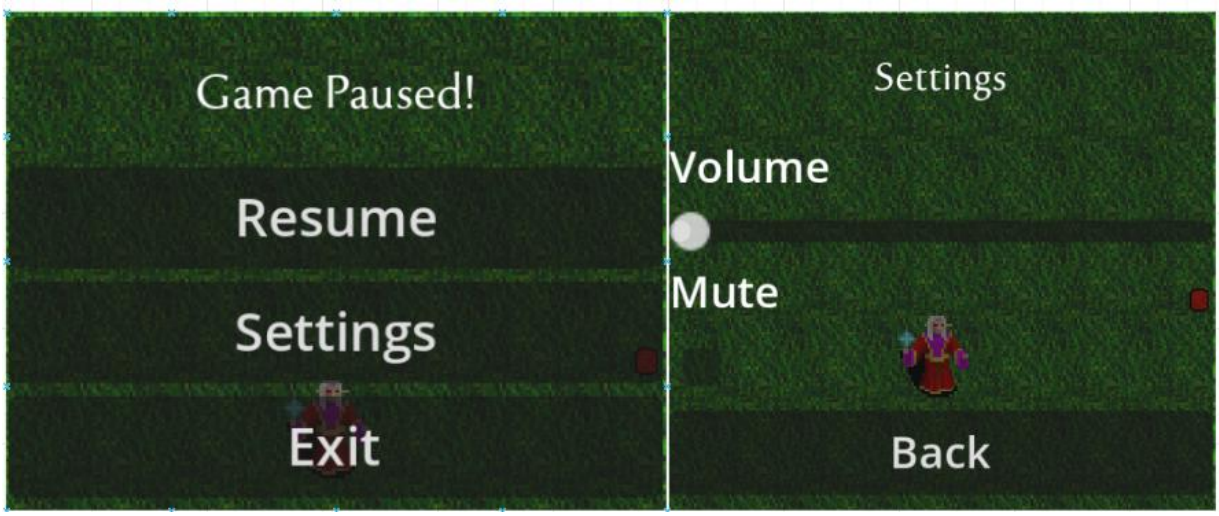


Рисунок 4.29 – Взаємодія з меню паузи

**Висновки до розділу 4**

У четвертому розділі розроблено основний функціонал гри, створено сцену з головним персонажем, ворогами, зброєю, спавнером ворогів та меню вибору апгрейдів. Розроблено код для взаємодіє всіх цих елементів між собою та користувачем. Створено інтуїтивно зрозумілий користувацький інтерфейс який дозволить гравцю зручно взаємодіяти з грою.

На основі розроблених в третьому розділі сценаріїв використання було створено керівництво користувача в якому детально описані дії які користувач має виконати для досягнення бажаного результату, це допоможе новим користувачам, які ще не знайомі з іграми краще зрозуміти що потрібно роботи в грі.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було досягнуто поставленої мету, виконавши наступні завдання:

- проведено аналіз предметної області та схожих застосунків;
- спроектовано гру в жанрі top-down shooter;
- обрано технології для розроблення гри в жанрі top-down shooter;
- визначено функціональні вимоги до гри;
- реалізовано програмний застосунок;
- протестовано розробленого програмного застосунку.

В першому розділі проведено дослідження ігрової індустрії в світі та Україні з метою визначення самих популярних ігрових жанрів та платформ, всі отриманні дані було використано для вибору тематики гри та основної аудиторії на яку вона розрахована. Також було проведено аналіз існуючих ігрових застосунків для визначення вдалих рішень, механік та загальних тенденцій в жанрі.

В другому розділі проведено дослідження існуючих ігрових рушіїв які можуть бути використані для розробки гри, їх особливостей, переваг та недоліків, на основі яких було прийнято рішення використовувати Godot Engine як найкраще рішення для створення 2D-гри. Розроблено специфікацію вимог до ігрового застосунку, що відповідають новітнім стандартам та тенденціям ігрової індустрії та на основі яких були створені основні механіки гри та користувацький інтерфейс.

В третьому розділі розроблено п’ять сценаріїв використання системи користувачем для кращого розуміння обмежень системи та задач яких користувач захоче досягти граючи в гру. Створено мокапи інтерфейсів з детальним описом функцій які вони виконують.

В четвертому розділі розроблено основний ігровий процес з функціями для керуванням персонажем, автоматичною атакою зброї, ворогів, меню підвищення рівня. Створено керівництво користувача для допомогти користувачам які вперше грають в гру.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. How Many Games Are There? URL: https://explodingtopics.com/blog/number-of-gamers (дата звернення 28.04.25)

2. Глобальна ігрова статистика за 2023 рік URL: https://gamedev.dou.ua/news/gaming-industry-in-numbers-2023/ (дата звернення 28.04.25)

3. Прибутки українських компаній за рік URL: https://ain.ua/2024/09/10/iaki-pitomo-ukrayinski-geimdev-studiyi-otrimali-naibilsi-pributki-za-rik/ (дата звернення 28.04.25)

4. Most Popular Video Game Genres in 2025: Revenue, Statistics URL: https://rocketbrush.com/blog/most-popular-video-game-genres-in-2024-revenue-statistics-genres-overview (дата звернення 28.04.25)

5. Mosler P., Gehring M., Dokonal W., Cizmeci M., Geist P., Haas T., Soares T., Sohlmayer C., Rüppel U. Using the Game Engine Unity Efficiently in Teaching: Development of a fully-automated webserver-based build pipeline. eCAADe (Education and Research in Computer Aided Architectural Design in Europe), 2023.

6. Kusheryanto A., Mirza A., Syaripudin A., Hutagalung D. D. Development of the Story of Life: A Narrative and Educational Game Using the Godot Engine for Android. *JISA(Jurnal Informatika dan Sains)*. 2025. Vol. 7, № 2. P. 115–124.

7. Sun Y., Wang H., Zhang Z., Diels C., Asadipour A. Executing realistic earthquake simulations in unreal engine with material calibration. *Computers & Graphics*. 2024. Vol. 124. P. 104091.

8. Game Enemy Design Starter Guide URL: https://gamedevfriends.com/enemy-design-starter-guide/ (дата звернення 28.04.25)

9. Game Progression and Progression Systems URL: https://gamedesignskills.com/game-design/game-progression/ (дата звернення 28.04.25)