Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок в жанрі Shoot'em-up. Механіки головного героя, UI, анімація, реалізація та модифікація зброї
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-7

Хом’яков К.І.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ас. Матвєєв Д.І.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Факультет Комп’ютерних наук

Кафедра Програмної Інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва)

ЗАТВЕРДЖУЮ

_____
(підпис)

«___» _____ 20 24р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Хом’якову Кирилу Ігоровичу*
(прізвище, ім’я, по батькові)

1. Тема роботи: Ігровий програмний застосунок в жанрі Shoot'em-up. Механіки головного героя, UI, анімація, реалізація та модифікація зброї.

Затверджена наказом по університету від 20.05.2024 р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10.06.2024.

3. Вихідні дані до роботи*: Розробити бойову систему з боку гравця та програмні інструменти для створення екземплярів зброї, систему взаємодії гравця із персонажем. Також створити самі екземпляри зброї для гравця. Реалізувати різні механіки зброї. Реалізацію системи виконати на мові програмування C# та ігровому двигуні Unity3D.*

4. Перелік питань, що потрібно опрацювати в роботі:

Вступ, аналіз предметної галузі, перелік вимог до програмного забезпечення, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування програмного забезпечення, впровадження програмного забезпечення.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Створення специфікації ПЗ	22.04.2024	*виконано*
3	Проектування ПЗ	29.04.2024	*виконано*
4	Розробка ПЗ	06.04.2024	*виконано*
5	Тестування ПЗ	13.05.2024	*виконано*
6	Оформлення пояснювальної записки	20.05.2024	*виконано*
7	Підготовка презентації та доповіді	27.05.2024	*виконано*
8	Попередній захист	06.05.2024	*виконано*
9	Нормоконтроль, рецензування	06.05.2024	*виконано*
10	Здача роботи у електронний архів	07.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	09.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Хом’яков К.І
(підпис)

Керівник роботи _____ ас. Матвєєв Д.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 93 с., 59 рис., 1 таблиця, 14 джерел.

АЛГОРИТМ, ГРАВЕЦЬ, ЗБРОЯ, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, ІНСТРУМЕНТ, STFCW, СИСТЕМА БОЮ, C#, JET BRAINS RIDER, SHOOT`EM UP, UNITY3D.

Об’єктом роботи є розробка системи бою з боку гравця та реалізація можливостей взаємодії гравця із його віртуальним персонажем, для ігрового програмного застосунку у жанрі shoot`em up. Реалізація STFCW (з англ. програмний інструмент для створення зброї) – інструменту для створення різноманітних екземплярів зброї.

Метою розробки є планування та програмна реалізація системи для взаємодії гравця із його персонажем, бойових функцій персонажу. А також зручної системи для швидкого створення, та введення зброї у гру

Методи рішення – використання ігрового рушія Unity3D та мови програмування C# для реалізації системи бою гравця та для розробки системи створення зброї. IDE Rider для більш швидкого та якісного написання програмного коду. Adobe Illustrator для створення візуальних елементів користувацького інтерфейсу.

У результаті роботи сплановано та програмно реалізовано систему для швидкого створення зброї гравця. Створено набір екземплярів різної зброї гравця для бойової системи. Розроблено власну систему бою гравця, ґрунтуючись на аналізі систем бою у схожих проектах.

ALGORITHM, PLAYER, WEAPON, GAME APPLICATION, TOOL, COMBAT SYSTEM, C#, JET BRAINS RIDER, SHOOT’EM UP, UNITY3D.

The object of this work is the development of a combat system from the player's perspective and the implementation of player interaction capabilities with their virtual character for a shoot 'em up game application. The implementation of STFCW (Software Tool for Creating Weapons) – a tool for creating various weapon instances.

The aim of the development is to plan and programmatically implement a system for player-character interaction, character combat functions, as well as a convenient system for quickly creating and introducing weapons into the game.

Methods used – utilizing the Unity3D game engine and the C# programming language for implementing the player's combat system and developing the weapon creation system. IDE Rider is used for faster and higher quality coding. Adobe Illustrator is used for creating visual elements of the user interface.

As a result of the work, a system for quickly creating player weapons has been planned and programmatically implemented. A set of various player weapon instances has been created for the combat system. A custom player combat system has been developed based on the analysis of combat systems in similar projects.

Я, Хом’яков Кирило Ігорович, студент гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Система зброї та бою для ігрового програмного застосунку у жанрі shoot`em up», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

PC – Personal Computer

ОС – Операційна Система

STFCW – Software Tool For Creating Weapons

UI – User Interface

ВСТУП

Shoot 'em up відноситься до піджанру шутерів, які є частиною більшої категорії екшн-ігор. Ці ігри зазвичай мають вид зверху або збоку і дозволяють гравцям використовувати вогнепальну зброю для враження віддалених цілей. Основна ціль гравця полягає в тому, щоб якомога швидше знищити будь-які рухомі об’єкти, які можуть стати загрозою. У деяких іграх персонаж може витримати декілька ударів, тоді як в інших - один удар призводить до його знищення. Успіх гравця в таких іграх залежить від його швидкої реакції та здатності запам’ятовувати послідовності атак ворогів. Інша особливість ігор цього жанру - це швидкий темп гри.

Початки жанру можна простежити від гри «Spacewar» однієї з найперших комп’ютерних ігор, розробленої в 1962 році, яка з часом поширилася в залах ігрових автоматів у першій половині 1970-х років. Протягом 1970-х років завдяки іграм типу Space Invaders та Asteroids популярність жанру зростала. Найбільша популярність жанру припадає на 1980-ті та початок 1990-х років. У середині 1990-х років ігри в жанрі зайняли свою нішу. Починаючи з цього часу, ігри базуються на умовному типовому геймдизайні, створеному протягом 1980-х років.

Shoot'em up має багато своїх піджанрів, можемо виділити такі основні піджанри: фіксований шутер, рейковий шутер, тунельний шутер, скролл-шутер, мультиспрямований шутер, кульове пекло, беги та стріляй [1].

Наш застосунок відноситься до мультиспрямованого шутеру. Мультиспрямований шутер, або multidirectional shooter англійською, це тип ігор, де гравцю надається повна свобода руху на 360 градусів. Головний герой може обертатися та рухатися в будь-якому напрямку. Ці ігри, в яких один джойстик контролює рух, а другий визначає напрямок стрільби, іноді відомі як twin-stick shooters (англ. – «шутери для двох джойстиків»).

Механіка бою у цьому жанрі зазвичай була проста. Гравець мав декілька видів зброї. Зброя відрізнялася траєкторією польоту снаряда, кількістю випущених снарядів за раз, рівнем шкоди яку наносив снаряд противнику. Гравцю було

потрібно постійно маневрувати поміж снарядів що летять у його бік. Здоров’я гравця було обмежено, тому кожне влучання у гравця швидко наближувало його до поразки. Але жанр розвивався та з часом система ставала складнішою та різноманітнішою. Як приклад, з’явилась механіка поліпшення зброї та деяких параметрів персонажа гравця (наприклад кількість здоров’я), між рівнями гри. А також зброя стала більш різноманітною між собою.

У цьому проекті ставиться на меті створити динамічну систему бою. Ставка робиться на реакцію гравця та удосконалення його навичок рухів та стрільби у рамках гри. Швидкі противники постійно вимушують гравця відходити та відстрілюватись, маневруючи поміж противників, снарядів та різних перешкод (ландшафт, різні оточуючі об’єкти навколишнього світу).

Гравець має енергетичний щит який з часом відновлюється сам. Це дозволяє збалансувати гру щоб гравцю було легше адаптуватись до динаміки гри та рівня складності. Здоров’я само по собі відновлюватись не буде та втрата усіх очок здоров’я стане поразкою для гравця.

Для підтримки динаміки – гравець зможе прискорювати біг для простішого маневрування у складних ситуаціях, та робити ривки у різні напрями. Ці та подобні дії витрачають витривалість гравця, це буде обмежувати гравця у та примусить його планувати свої дії на перед та грати більш тактично.

Важливим моментом є створити цікаву та незвичайну зброю. Кожен вид зброї повинен мати свої особливості відрізнятись за відчуттям гри. Один вид зброї більш вигідно використовувати у одних ситуаціях та зовсім не вигідно у інших. Це також додає для гравця розмірковувань про те коли та яку зброю краще використовувати.

У сукупності усі ці елементи бою створюють швидку динаміку та вимушують гравця швидко планувати свої дії та тактику ведення бою. Це робить гру більш цікавою та різноманітною, що в сою чергу приваблює гравців.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

1.1.1 Аналіз конкурентних проектів

Перш за все гра повинна подобатись гравцям, які будуть її купувати. Успіх проекту залежить від того наскільки гра буде приваблювати гравців до покупки та наскільки вона буде їм подобатись після покупки. Механіки гри та її загальний вигляд повинні підтримувати зацікавленість гравця не тільки на презентаціях, а і під час гри, не можна розчарувати гравця який купив гру, це вимусить покупців відчувати себе ошуканими. Коли ми зможемо привабити гравця, він придбає гру та отримає саме те що хотів, це створить для продукту гарну репутацію. Гарна репутація посприяє ще більшій кількості покупок, а також це може створити позитивні умови для створення доповнень або продовжень гри. Спираючись на це можна сказати що головним є розуміння того, які елементи та особливості гри будуть більше всього приваблювати гравця у обраному для гри жанрі. Тому аналіз побудовано перш за все саме на механіках та особливостях у іграх такого самого жанру або у іграх які у основі мають такі самі механіки як і наш проект.

Helldivers 1 – ізометричний кооперативний twin-stick shooter (с англ. — «шутер для двох стиків») у sci-fi стилістиці (с англ. — «науково фантачтична»). У цій грі гравцю потрібно проходити рівні визволяючи різні планети від декількох ворожих фракцій, гра розрахована на кооператив так що проходити краще рівні разом із друзями.

У грі є гарна різноманітність зброї, але здебільшого кожна рушниця не є дуже особливою на відмінність від іншої. Якась стріляє у автоматичному режимі, якась дробом, у однієї більша шкода, але низька скорострільність у іншої навпаки. Тобто зброя є важливим аспектом гри і зроблена не погано, але не є її головною особливістю яка буде приваблювати гравця. У грі є цікава механіка «стратагемів» – це особлива атака яку може викликати гравець під час гри яка може значно змінювати хід битви. Ця механіка робить гру цікавішою, на більш веселішою. Також бажання грати у гру додає прогресія розвитку власного персонажа гравця.

Гравець може відкривати нову зброю, стратагеми та броню. Але головним аспектом веселощів у грі є кооператив. Граючи с друзями гравці самі будуть створювати веселі та незвичайні ситуації використовуючи можливості гри.

З недоліків гри можна виділити те що у гру нецікаво грати одному, вона розрахована на компанію друзів і це є одночасно плюсом та мінусом гри. Не усі будуть мати компанію, у когось не так багато вільного часу та важко домовитись з іншими про зручний час проведення ігрової сесії. Тому це є проблемою гра втрачає немало аудиторії не маючи повноцінного режиму гри для одного гравця. Грати одному можна, але гра на це на розрахована через що це набагато нудніше та гравець швидко втратить зацікавленість граючи сам.

Cuphead – один із найсучасніших та успішніших представників жанру shoot`m up. Cuphead відноситься одразу до двох піджанрів – горизонтальний скролл-шутер (англ. horizontally scrolling shooter) та біжи та стріляй (англ. run and gun). Цей проект має насамперед відносно прості механіки та не приносить чогось особливого з точки зору механік. Але те поєднання простих на перший погляд механік створює дуже особливий та складний проект. Цей проект входить до списку найскладніших ігор, які виходили за останній час.

Важливою особливістю Cuphead є її візуальний стиль. Гра не намагається робити інноваційну графіку, а натомість робить ставку на особливий та новий для відеоігор візуальний стиль (див. рисунок 1.1). Розробники обрали стиль мультфільмів 30-х років. Цей стиль є приємним для ока, виглядає особливо та не потребує великих потужностей, від платформи на яку встановлена гра. Головною особливістю цього стилю є те що він простий та для створення візуальної частини гри не потрібно великої кількості ресурсів.

Варто сказати що не дивлячись на особливий та візуально гарний стиль він може заважати, у розпалі битви на екрані дуже багато об’єктів: снаряди, противники, сам гравець, багато анімованих об’єктів. Все це буде плутати гравця та заважати йому орієнтуватись у ситуації, а якщо у гру грають два гравця у кооперативному режимі то тут проблема стає ще більш помітною. На екрані буде ще більше об’єктів, а гравці час від часу будуть плутатись де чий персонаж. Але

дивлячись на те що це гра розрахована на складність проходження, то можна сказати що це ще один елемент для ускладнення гри. Це залишається на розсуд самого гравця, одного це буде дратувати, інший буде це сприймати як ще одне випробування. Тому для вирішення цієї проблеми у гри є чорно-білий візуальний режим, позбавлення від великої кількості кольорів розгружує екран та дає змогу гравцю більше концентрувати увагу на персонажі та противниках.

Рисунок 1.1 – Візуальний стиль Cuphead (зроблено самостійно)

Складність є ще одною основною особливістю проекту. Гра кидає виклик гравцю, а гравець його приймає. Цей прийом не новий та є дуже ефективним. Успішним цей прийом робить те, що коли гравець нарешті проходить складний рівень на який він витратив багато часу, він відчуває насолоду та полегшення. Саме через це відчуття гравець знову хоче повертатись до гри, а передчуття насолоди викликає у гравця ще й азарт. Також великим плюсом цього проекту є функція кооперативу. Кооператив є гарною можливістю для гравця пройти гру з другом або навіть повернутись до гри ще раз, після її проходження на одинці, щоб пройти її ще й з другом.

Складність Cuphead досягається за рахунок:

– швидкої динаміки гри, гравцю потрібно бігти з одного боку екрану у

інший, поки він не досягне кінця рівня;

- гравець змушений постійно ухилятися від великої кількості снарядів та самих ворогів, які з’являються з усіх напрямів;
- необхідно багато та швидко переміщуватись, що ускладнено не тільки ворогами, а й перешкодами;
- кількість очок гравця обмежена та їх важко відновлювати, гравцю потрібно постійно бути обережним й майже досконало точним у діях;
- бої з босами – кожен бос є окремим випробуванням для навичок гравця, а також має свою поведінку, через це гравцю потрібно знаходити особливий підхід до кожного боса.

Alien shooter – це серія ігор у жанрі shoot`em up, яка вийшла у 2003 році та має дві основні частини. Гравцю потрібно проходити рівні, на кожному з них гравцю потрібно виконати деякі прості завдання, але головною складністю є велика кількість противників, які будуть перешкоджати гравцю досягнути його мети на рівні. Противники йдуть на гравця хвилями, нові хвилі з’являються коли гравець починає проходити далі по рівню. У гравця є непоганий арсенал із 9 видів зброї, кожен вид має якісь свої особливості. На початку гравцю доступний один від зброї, відкрити нову гравець може або поміж рівнів за зібрані на минулих рівнях гроші, або знаходячи нову зброю у схованих частинах рівня. У схованих частинах рівня гравець знаходить: нову зброю, гроші, амуніцію – це вимушує гравця добре досліджувати кожен рівень. Головна цікавість гри полягає у веселощі, а не наприклад у складності, як у випадку із Cuphead. Гравцю весело грати у гру через велику кількість ворогів, та добре підібраний для гравця арсенал зброї, для їх знищення. Також веселощів для гравця додають різні візуальні ефекти які програються під час пострілів та від улучань по ворогам. Alien shooter є гарно зробленим проектом саме для того щоб гравець міг відволіктись від буденності та весело провести час відпочивши у грі.

Як і у інших у Alien shooter є мінуси, головною проблемою наприклад першої частини було те що у грі було не так багато вмісту. Через це гравцю було цікаво

один раз пройти гру, але він не завжди хотів повертатись. Зброя хоч і була цікавою та зробленою особливою між собою, але її було не так багато, видів противників було не багато, завдання на рівнях були однотипні та так само не цікаві. Якби ця гра мала більше вмісту гравці могли б повертатись до неї частіше та більше були би зацікавлені у продовжені.

Друга частина серії же стала роботою над помилками розробників, там з'явився сюжет, багато вмісту, біль широка та пророблена система прогресії персонажу гравця.

Але не дивлячись на всі ці виправлення друга частина не була успішна. Сюжет був нудний, рівень складності погано збалансований, був занадто різкий стрибок складності між рівнями, велика кількість технічних проблем. Розробники хотіли зробити все та одразу, через це вийшов продукт середньої якості, бо вони не могли сконцентрувати увагу розвинені сильних аспектів гри.

1.1.2 Аналіз ринку з точки зору жанрів

Ще одним важливим аспектом для аналізу є розуміння ринку відеоігор. Ґрунтуючись на звіті від Newzoo's [2] можна зробити висновок що найпопулярнішим жанром на PC на момент 2024 року є шутери (див. риснок 1.2).

У 2023 році шутери відіграли важливу роль, заробивши 14,1% від загальних доходів від ігор на PC. Очікується, що цей жанр продовжить зростати з темпом 4,9% на рік, що значно перевищує загальний темп зростання ринку персональних комп'ютерів, який становить 1,6%.

Шутери продовжують тримати лідируючі позиції в ігровій індустрії, будучи на сьогодні найбільш вподобаним жанром відеоігор. Ці ігрові проекти відрізняються акцентом на бойові дії з використанням зброї та різноманітність навколишнього середовища. Розробка ігор цього жанру вимагає від розробників детального балансу між геймплейними механіками та візуальним наративом, задовольняючи потреби гравців у тактичній глибині та безпосередній, емоційно насиченій дії.

Це значить що попит на ігри цього жанру великий та випуск гри шутера є

може бути вигідною можливістю. Але також варто розуміти що конкуренція у цьому жанрі буде великою, з’явиться велика кількість проектів із якими меншим проектам буде важко конкурувати. Тому гарним виходом буде випуск гри яка є піджанром шутеру, таким чином розробники зіткнуться із меншою кількістю конкуренції, будуть мати більшу вірогідність на те що їх проект буде помічено покупцями, та при цьому залишаться у рамках популярного жанру.

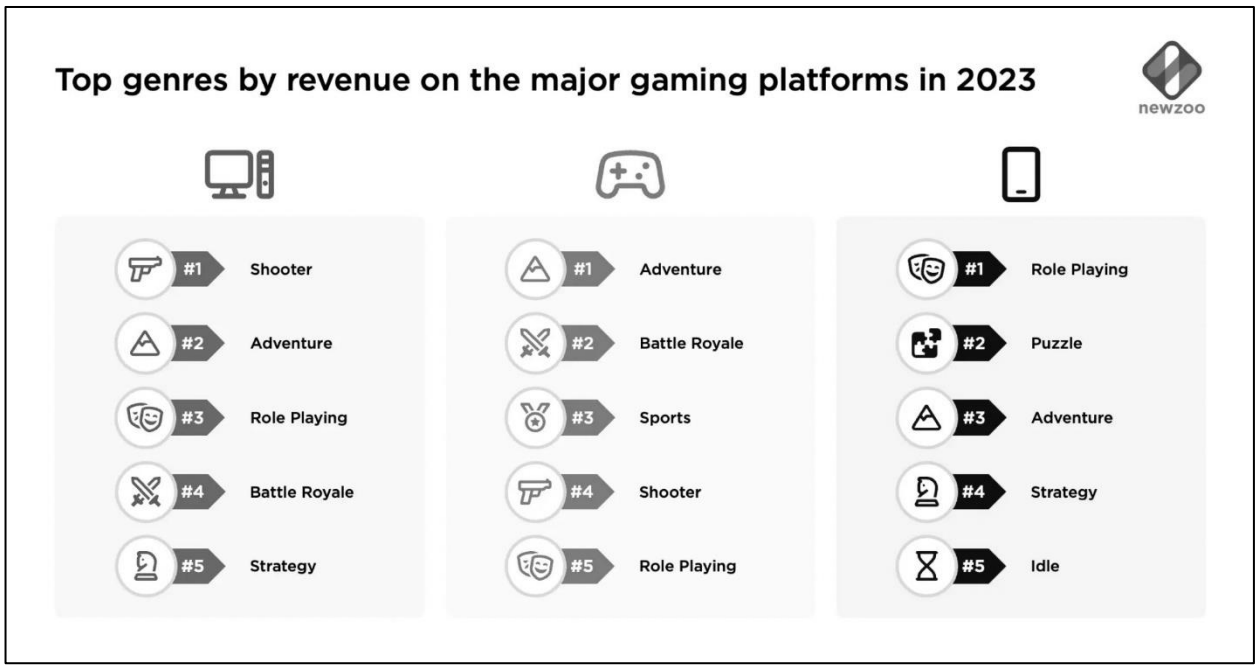


Рисунок 1.2 – Статистика популярних жанрів на різних платформах (за даними [2])

Варто відмітити що шутер так само входить до топ п’яти популярних жанрів на консолях та займає четверту позицію, це робить жанр ще більш привабливим для розробників та дає сприятливі умови для випуску гри ще й на консолі.

1.1.3 Аналіз ринку з точки зору цільової аудиторії

Спільнота геймерів стає все більш різнобічною з погляду вікової категорії, оскільки число як молодих, так і досвідчених гравців продовжує зростати. Чоловіки витрачають на відеоігри більше 6 годин, що лише на годину більше, ніж час, який жінки приділяють грам - 5 годин.

В контексті постійної трансформації геймінгової індустрії, важливим є

розуміння демографічної структури геймерської спільноти. Це передбачає аналіз таких аспектів, як віковий та статевий розподіл гравців, що надає важливу інформацію про пріоритети, поведінку та мотивацію різних сегментів геймерів.

Кількість гравців у відеоігри, яка становила 2.7 мільярди в 2020 році, зросла на 5.6% і досягла 3.2 мільярди в 2023 році. Очікується, що до 2024 року ця кількість зросте до 3.34 мільярдів. Велика частина цього зростання відбулася завдяки ринку Азіатсько-Тихоокеанського регіону. З урахуванням повернення до нормального життя, ці показники продовжать зростати.

Дані про вік гравців по всьому світу свідчать про постійну еволюцію ігрового середовища. Середній вік гравців у відеоігри продовжує зростати, досягаючи відмітки від 35 до 44 років у 2022 році. Свіжий норвезький відгук 2020 року виявив, що серед дітей та підлітків віком від 9 до 18 років 96% хлопців та 76% дівчаток регулярно грають у відеоігри. У 2018 році частка граючих хлопців була на тому ж рівні, тоді як лише 63% дівчат визнали себе геймерами [3].

Чоловіча аудиторія придбає відеоігри втричі частіше, ніж жіноча. Це пов’язано з тим, як жінки сприймають геймінг. Незважаючи на те, що за останні роки кількість жінок-геймерів зросла, вони все одно сприймають це як лише один із способів проведення вільного часу. Тільки 6% жінок вважають себе «геймерами», у порівнянні з 15% чоловіків. Чоловіки великими шансами захоплюються відеоіграми. Тому для них звичайно купувати більше ігор [4].

Шутери є вибором для 35% гравців у багатокористувацьких іграх. Екшн-ігри займають друге місце з 28%, а казуальні ігри слідують за ними з 27% [5].

Також 66% людей вважають, що графічна якість є вирішальною при виборі гри для покупки. 63% респондентів вказують, що вартість є ключовим фактором для них. Також, 61% цінують гру з цікавим сюжетом. Інші аспекти, які впливають на рішення про покупку гри, включають “присутність у улюбленій серії” (51% вважають це важливим) та “сумісність з онлайн-іграми” (50%) [5].

На 2023 рік у світі налічується 3,22 мільярда геймерів. Проте це число очікується зросте до 3,32 мільярда геймерів до 2024 року. З’явлення мережі 5G також сприяло збільшенню кількості геймерів по всьому світу. Великі гравці ринку

вже вживають кроки в цьому напрямку, щоб здобути конкурентну перевагу. Наприклад, у січні 2022 року AT&T оголосила про співпрацю з NVIDIA Corp. для надання високоякісного хмарного геймінгу на 5G. Ця ініціатива дозволить користувачам грати в близько 100 безкоштовних ігор на платформах PC, таких як Epic Games Store, Steam, Ubisoft Connect, Origin, GOG тощо. Реалістична графіка на базі технології NVIDIA RTX та розширені функції штучного інтелекту нададуть геймерам віртуальний світовий досвід.

Глобальний розмір ринку відеоігор у 2022 році оцінювався в 217,06 мільярда доларів США і очікується, що він зросте з 13,4% річною темпом до 583,69 мільярда доларів США до 2030 року. Цей ріст пов’язаний з постійним трендом онлайн-геймінгу, з’явленням високої пропускної здатності мережі.

З цієї статистики можна зробити висновки що ринок відеоігор за останній час буде тільки зростати, оскільки кількість геймерів становиться більше з кожним роком. Також молодь показує великий відсоток граючих людей, це також свідчить про подальше зростання ринку.

Враховуючи що наш проект є піджанром шутеру, можна сказати що головною аудиторією гри будуть чоловіки від 16 до 38 років. Хлопці більше зацікавлені у цьому жанрі оскільки полюбляють здебільшого динаміку та екшн.

1.2 Виявлення та актуалізація рішень

Провівши аналіз можна сказати що для того щоб гра мала успіх, їй потрібно мати свої особливості та сильні сторони, які будуть вигідніші та доречні у обраному жанрі.

Кажучи про shoot`em up доречним буде зосередити акцент на таких аспектах геймплею:

- веселий та динамічний геймплей;
- особлива та різноманітна зброя у арсеналі гравця;
- велика кількість різних ворогів;
- прогресія розвитку персонажа гравця ;

- досить складний, але добре збалансований геймплей;
- гарна візуальна складова проекту.

Якщо взяти саме бойову систему найскладнішим буде досягнути золотого балансу – щоб гра кидала гравцю виклик випробувати себе, але при цьому йому не було надто складно та зацікавленість у грі залишалась. Гра не повинна набридати гравцю після першої ж ігрової сесії, та повинна підтримувати якомога більше зацікавленість у грі. Бойова система повинна утримувати зацікавленість гравця та надавати йому почуття ніби він дійсно відчуває кожен постріл кожне влучання по противнику. Досягати цього потрібно враховуючи багато аспектів: звук зброї, візуальний вигляд як самої зброї так і ефектів пострілів та анімацій персонажа при пострілах, ефекти та звуки від влучання противника, реакція зовнішнього середовища на постріли або навіть реакція користувацького інтерфейсу (наприклад тряска якихось елементів, анімації прицілa тощо).

Важливо створити власні особливі та цікаві поєднання механік, щоб продукт міг конкурувати із іншими продуктами ринку. Проекту потрібно виділятись на фоні інших проектів та бути помітним, це є одним із найголовніших принципів якого треба дотримуватись – проект повинен виділятись та мати щось особливе та своє. Не обов’язково що б це були повністю нові механіки яких ніде раніше не було або що б це був повністю стилістично особливий проект. Звісно якщо є змога додати щось повністю своє то це буде перевагою. Але такі інноваційні ідеї є не завжди, у такому разі потрібно використовувати старі та прості механіки, але можна змінити поєднання цих механік, утворюючи комбінації яких раніше не було. Або змінити погляд на старі механіки та застосувати їх не так як це робили раніше у інших проектах.

1.3 Постановка задачі

Під час реалізації проекту потрібно реалізувати бойову систему з боку гравця та його взаємодію із віртуальним аватаром. Реалізація повинна враховувати усі важливі аспекти систем такі як: програмна реалізація системи, звукові ефекти,

візуальні ефекти, анімації, баланс зброї.

Також система повинна враховувати особливості системи противників та мати шляхи взаємодії із противниками (нанесення шкоди, накладання негативних ефектів, зміна стану противників). Система повинна також взаємодіяти із навколишнім середовищем: створювати різні візуальні та звукові ефекти від влучань у ландшафт, залишати сліди від влучань. Бойова система повинна враховувати стани персонажа та обмежувати доступ переходу до інших станів до того моменту поки не закінчиться поточний стан (наприклад гравець не може стріляти під час того поки змінює зброю тощо). Для зброї повинні бути реалізовані різні параметри, щоб мати можливості створювати особливу, не схожу між собою зброю.

Для прискорення створення різних екземплярів зброї потрібно розробити програмний інструмент який дозволять швидко створювати ці екземпляри. Цей інструмент повинен надавати розробнику можливість створювати різні варіації зброї по шаблону, змінюючи параметри зброї. Інструмент створення зброї по шаблону має на меті прискорення процесу розробки екземплярів зброї шляхом можливості швидко завдавати параметри які повинні бути у будь якої зброї (рівень шкоди, скорострільність, режим заряджання тощо), при цьому залишивши можливість для модифікації параметрів зброї для створення особливої та не схожої одна на іншу зброї.

2 ПЕРЕЛІК ВИМОГ ДО БОЙОВОЇ СИСТЕМИ

2.1 Постановка мети

Метою проекту є реалізація бойової системи персонажа гравця та STFCW для створення зброї по шаблону. А також систему взаємодії гравця із персонажем. Системи повинна бути виконана на ігровому рушії Unity3D на мові програмування C#. Вона повинна працювати на платформі PC на ОС Windows.

STFCW повинен структурувати процес створення зброї використовуючи шаблони для налаштування параметрів зброї. А також давати розробнику можливість створення особливих видів зброї легким шляхом.

Бойова система повинна бути виконана враховуючи способи взаємодії з іншими системами гри (противники, навколишнє середовище, параметри самого персонажу гравця). Також система повинна задовільнять потреби гравця, створюючи цікавий та захоплюючий геймплей. Гравець повинен відчувати його взаємодію із зброєю. Також бойова частина гри повинна бути добре збалансована, мати гарну візуальну складову та доречні не заважаючи звукові ефекти.

Взаємодія гравця із його персонажем повинна виглядати автентично, гравець не повинен відчувати дискомфорт при керуванні персонажем. Керування повинно бути чуйним та коректно реагувати на виклики різноманітних функцій діями гравця. Також анімації персонажа повинні виглядати якомога природніше (схоже на людину у житті).

Також усі механіки повинні бути реалізовані відповідно із дизайн документом (див. додаток В).

2.2 Загальний опис

Система взаємодії персонажа реалізовуватиме слухання сигналів з пристроїв вводу та реакції персонажа на ці сигнали. Вона буде мати контролер гравця, через нього будуть взаємодіяти із гравцем інші системи гри.

Бойова система гравця реалізує можливість гравця за допомогою зброї, взаємодіяти із іншими системами ігрового застосунку. Система складається з ряду

контролерів. Вони надають доступ до необхідних даних про персонажа гравця іншим елементам системи, дозоляють змінювати деякі дані персонажа, відповідають за зміни стану персонажа та обмежують доступ до визначених дій під час різних станів персонажа.

Бойова система взаємодіє із іншими через контролери або інтерфейси. Також бойова система не відповідає за реалізацію функціоналу інтерфейсу з боку інших систем застосунку, бойова система лише взаємодіє із об’єктами, які наслідують та вже реалізовували інтерфейс з свого боку.

STFCW являє собою систему з ряду конфігураційних скриптів та одного головного, який реалізує основні функції зброї та зберігає у собі загальну інформацію про параметри зброї та дозволяє налаштовувати ці параметри.

Для структуризації даних та зручнішого створення зброї та подальшої модифікації системи, Головний конфігураційний файл складається з ряду вторинних конфігураційних файлів. Кожен з них відповідає за окрему групу даних або функціонал зброї, зберігає дані та функції пов’язані тільки з цим визначеним функціоналом.

2.3 Загальні обмеження

Бойова система як й ігровий застосунок повинна бути реалізована на мові програмування C# та використовувати ігровий рушій Unity3D версії 2022.

Бойова система повинна відповідати наступним обмеженням:

- бойова система не відповідає за реалізацію інтерфейсів, а лише використовує їх для взаємодії із іншими системами;
- реалізація бойової системи стосується лише персонажа гравця та його зброї;
- STFCW потребує модель зброї із точками-об’єктами на які він буде орієнтуватись при виконанні функції;
- STFCW повинна реалізовувати тільки функціонал для зброї дальнього бою;

- точки орієнтири потрібно розміщувати вручну, оскільки у кожної моделі зброї своя геометрія.

Система повинна працювати на ОС Windows 10 або11 – версія ОС повинна бути 64-го біта розрядності.

2.4 Припущення та залежності

Проект має такі припущення та залежності:

- для роботи бойової системи потрібна система противників яка буде реалізувати функціонал інтерфейсів, які використовує бойова система;
- STFCW також потребує реалізацію інтерфейсів іншою системою; інструмент можна легко модифікувати у разі потреби, але він все одно буде потребувати щось для ідентифікації противників, замість інтерфейсу.
- система адаптована тільки під ввід сигналів з клавіатури та миші
- бойова система може бути використана та модифікована на версіях рушія нижче або вище аніж Unity3D 2022.

Бойова система не використовує останніх інструментів наявних тільки у 2022 версії рушія, тому її можна використовувати на версіях нижче до версії 2019 LTS, але тестування проводилось тільки на версії 2022.

Система може бути легко модифікована під інші пристрої вводу (наприклад геймпад).Вона має для цього необхідний функціонал оскільки використовує Input System – компонент для відстеження вводу з різних пристроїв [6].

STFCW також може бути легко модифікований, щоб не потребувати інтерфейсів, але інструменту все одно буде потрібен об’єкт (скрипт, компонент, інтерфейс тощо) по якому він буде визначати ті об’єкти які можуть отримувати шкоду та можуть бути знищені.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ БОЙОВОЇ СИСТЕМИ

3.1 Проектування UML

Важливість використання діаграм прецедентів на стадії проектування архітектури програмного забезпечення не може бути недооцінена. Діаграми прецедентів відіграють ключову роль у розумінні та аналізі вимог до системи. Вони є візуальним зображенням взаємодії між користувачами та системою, що допомагає розробникам краще зрозуміти, як система повинна функціонувати.

Отже для системи взаємодії гравця з персонажем нашого ігрового застосунку було розроблено діаграму прецедентів (див. рисунок 3.1).

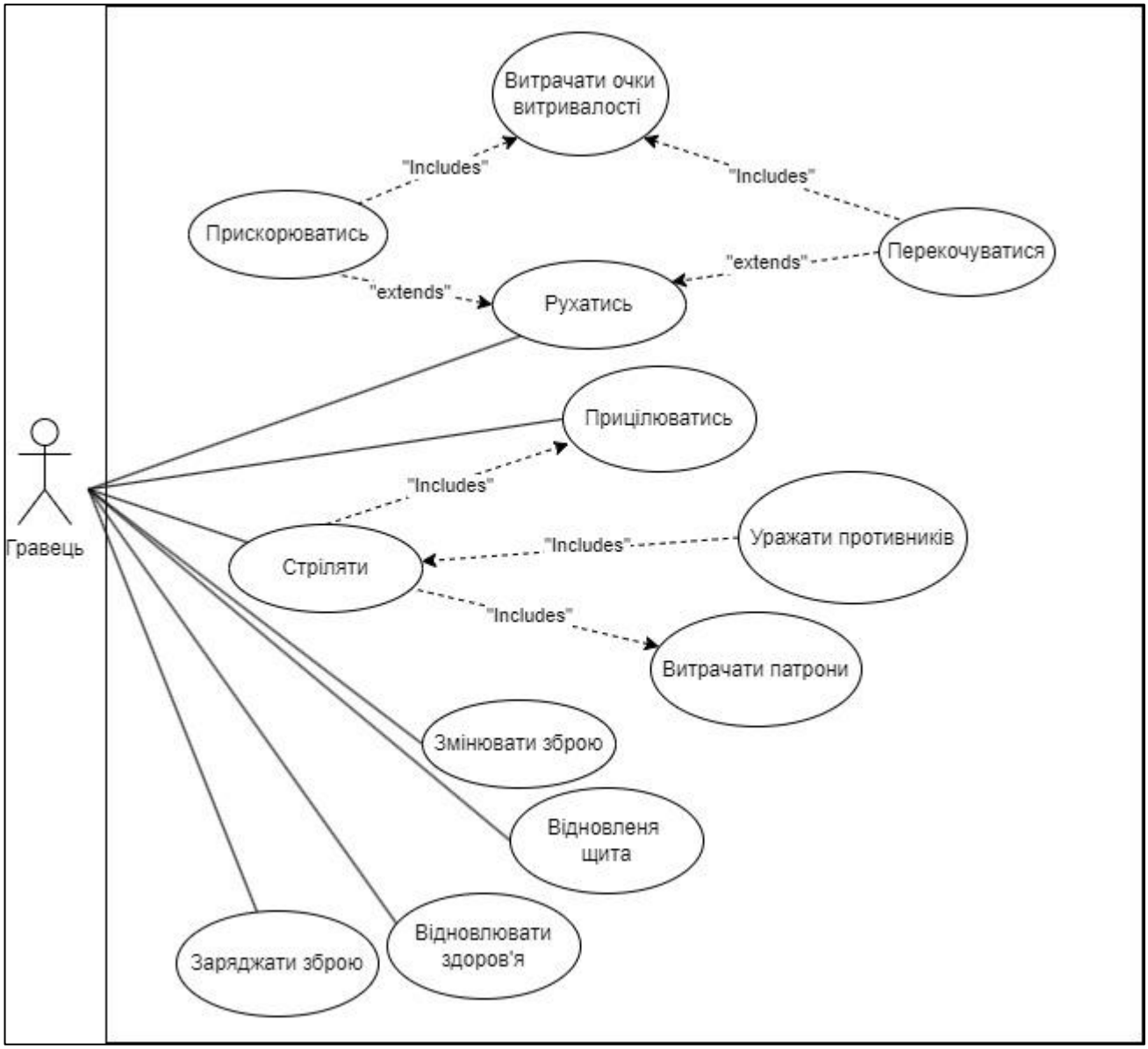


Рисунок 3.1 – Діаграма прецедентів системи взаємодії гравця з персонажем

(рисунок зроблено власноруч)

Ця діаграма прецедентів репрезентує взаємодію гравця із системою його персонажа. Діаграма прецедентів є важливим інструментом у стадії проектування архітектури програмного забезпечення. Вона допомагає розробнику розуміти та аналізувати вимоги, забезпечує ефективне спілкування із зацікавленими сторонами та є основою для інших артефактів розробки.

Бойова система персонажа повинна реалізовувати усі важливі елементи бою: переміщення, атаки, систему власного здоров’я персонажа, та реалізовувати усі додаткові елементи, які потрібні для функціонування або покращення системи.

Діаграма класів є важливим інструментом при розробці проектів, особливо в контексті проекту відеогри. Діаграма класів допомагає розробникам візуалізувати структуру системи, вказуючи на відносини між різними класами та об’єктами. Вона допомагає у плануванні та прогнозуванні ресурсів, необхідних для реалізації проекту.

Для STFCW було розроблено діаграму класів, яка відображає взаємовідносин між конфігураційними скриптами на яких побудовано STFCW (див. рисунок 3.2)

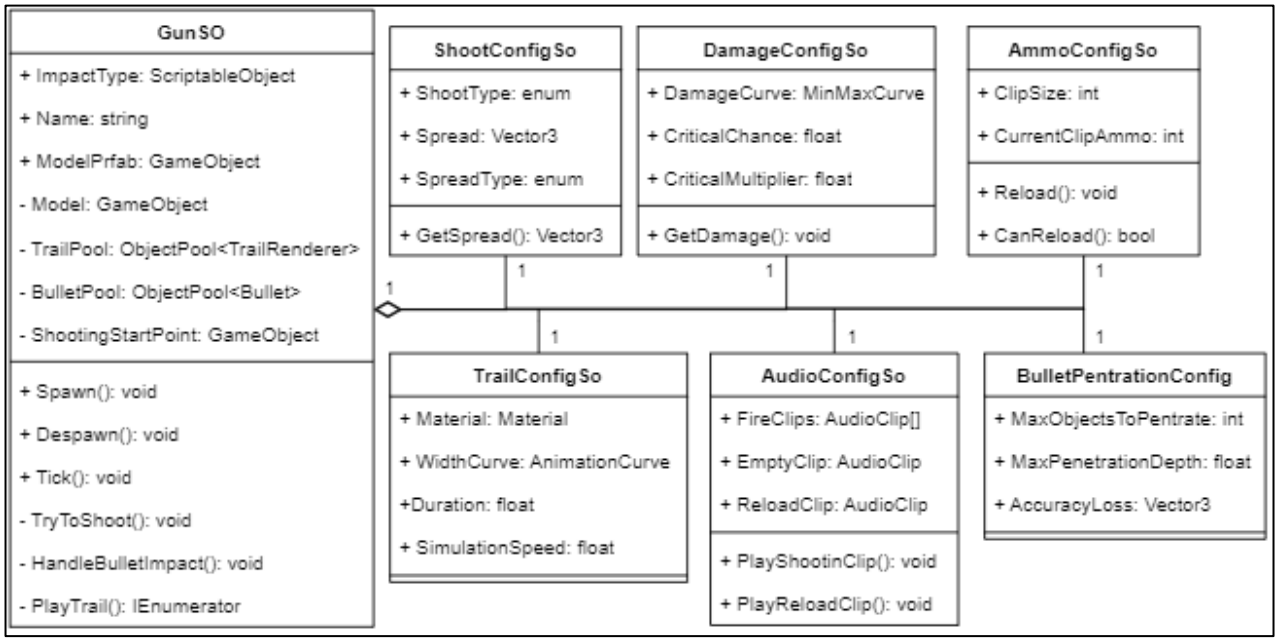


Рисунок 3.2 – Діаграма класів взаємодії конфігураційних скриптів (рисунок зроблено власноруч)

Загалом STFCW побудовано на створенні одного головного конфігураційного файлу «GunSo». Цей головний файл реалізує основний функціонал зброї та зберігає ключову інформацію про зброю. Він також складається з ряду менших конфігураційних файлів. Кожен з менших файлів відповідає за свою гілку функціоналу та зберігає відповідну інформацію щодо цього функціоналу, яку можна налаштовувати.

Таке розбиття на менші файли дозволяє нам зробити систему структурованою, де за кожну функціональну частину зброї відповідає свій окремий конфігураційний скрипт, який можна налаштовувати для досягнення необхідного ефекту. У цілому, ідея полягає в тому щоб розробник при створенні зброї міг швидко створити необхідний екземпляр, налаштовуючи та створюючи для кожної зброї свій набор конфігураційних файлів.

При такому підході створення зброї більше нагадує складання конструктора. Бо весь головний функціонал зброї вже буде реалізовано у конфігураційному файлі, розробнику залишиться налаштувати конфігурації та зайнятись редагуванням ігрового об’єкту-моделі на сцені редактору ігрового рушія.

3.2 Проектування архітектури застосунку

Система взаємодії гравця із персонажем складається з ряду контролерів та допоміжних скриптів які відповідають за поведінку персонажа, та зброї.

Система складається з:

а) контролерів:

1) Контролер гравця – «PlayerController»;
2) Контролер анімацій – «AnimationsController»;
3) Контролер стрільби – «ShootController»;
4) Контролер звуків – «SoundController»;

б) допоміжних скриптів:

1) Скрипт який відповідає за створення об’єктів зброї у гравця та зміни зброї між собою– «PlayerGunSelector»;

2) Скрипт застосовує модифікатори шкоди для зброї – «GunModifiersApplier».

Для кращого розуміння взаємодії гравця з персонажем до неї також було створено діаграму класів (див. рисунок 3.3).

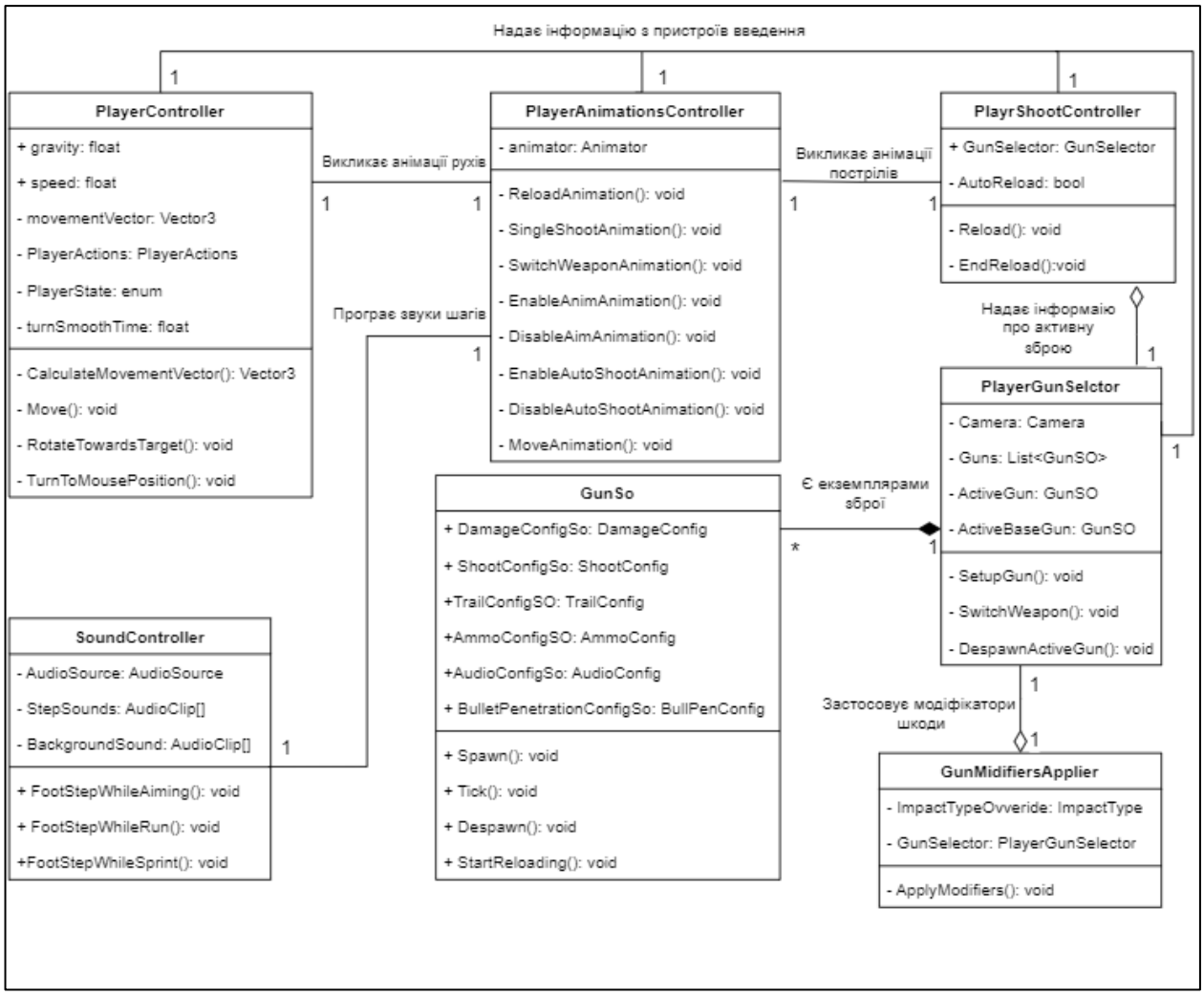


Рисунок 3.3 – Діаграма класів бойової системи (рисунок зроблено власноруч)

Розглянемо усі компоненти системи більш подрібніше.

Контролер гравця – виконує важливу роль системи, він отримує усі данні про сигнали з пристроїв вводу. Він надає доступ до цих даних іншим елементам системи через ряд методів «Get», кожен з яких надає інформацію про конкретні сигнали, після чого інші елементи системи активують відповідні цим сигналам дії персонажа. Також цей контролер відповідає за рухи персонажа та змінює координати персонажа відносно навколишнього світу гри через трьох вимірний

вектор. Не менш важливою функцією контролера є зберігання інформації про стан гравця, яку також використовують інші елементи системи, для розуміння чи можна робити деякі дії у поточний момент або ні.

Контролер анімацій – відповідає за комунікацію персонажа із компонентом «Animator» [7]. Цей компонент відповідає за налаштування, програвання, переходи анімацій доступних для персонажа гравця. Контролер відстежує ряд подій [8] (англ. «events»). Коли відбувається виклик контролер звертається до аніматора та програється необхідна анімація.

Контролер звуків – завдяки цьому контролеру можна легко налаштовувати звуки гри. Контролер відповідає за такі звуки: звуки пострілів, звуки шагів, музикальний супровід.

Контролер стрільби – слідкує за кількістю патронів у магазині зброї та відповідає за перезаряджання. Реагує на сигнали вводу які отримує з контролеру гравця та викликає функцію стрільби у активної зброї гравця, якщо це дозволяє стан магазину із патронами зброї.

Скрипт зміни зброї – відповідає за створення зброї у персонажа гравця на сцені гри[9], слідкує за списком наявної зброї у персонажа та відповідає за зміну зброї.

Скрипт застосування модифікаторів – реалізує можливість системи STFCW модифікувати шкоду від снарядів (наприклад: нанесення шкоди по області, накладання негативних ефектів на противників, модифікація параметрів зброї).

3.3 Проектування UI / UX

Перше що побачить гравець у грі це головне меню. З нього гравець може розпочати нову гру або продовжити ту що вже почав, а також налаштувати гру та вийти з неї (див. рисунок 3.4).

Рисунок 3.4 – Схематичне представлення головного меню (рисунок зроблено власноруч

Гравець має змогу ставити гру на паузу, під час чого він буде бачити меню паузи. Це меню дозволить гравцю продовжити гру, знявши її з паузи. А також налаштувати чи покинуту гру (див рисунок 3.5).

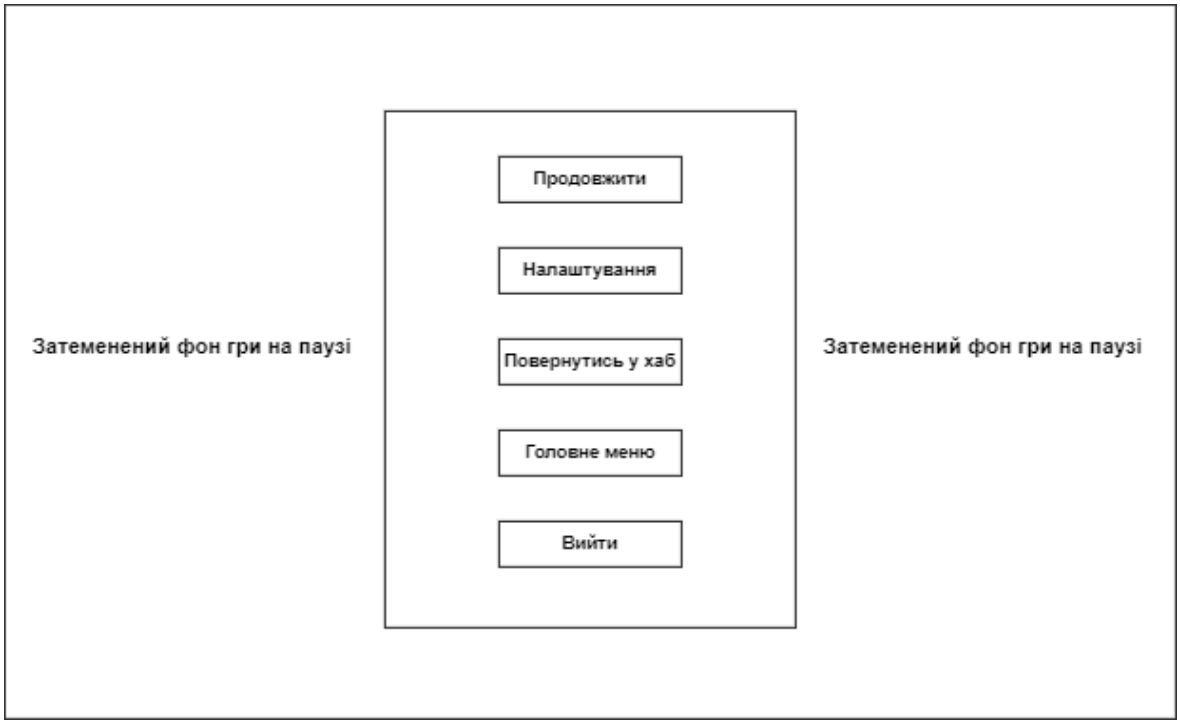


Рисунок 3.5 – Меню паузи (рисунок зроблено власноруч)

Під час основного ігрового процесу гравець буде бачити UI який буде відображати усю основну інформацію необхідну для гравця під час ігрового процесу. Ця інформація буде здебільшого про стан персонажа (кількість очок здоров’я, щита, витривалості) та ресурси персонажа (ресурс для створення патронів та кількість патронів у зброї) див рисунок 3.6.

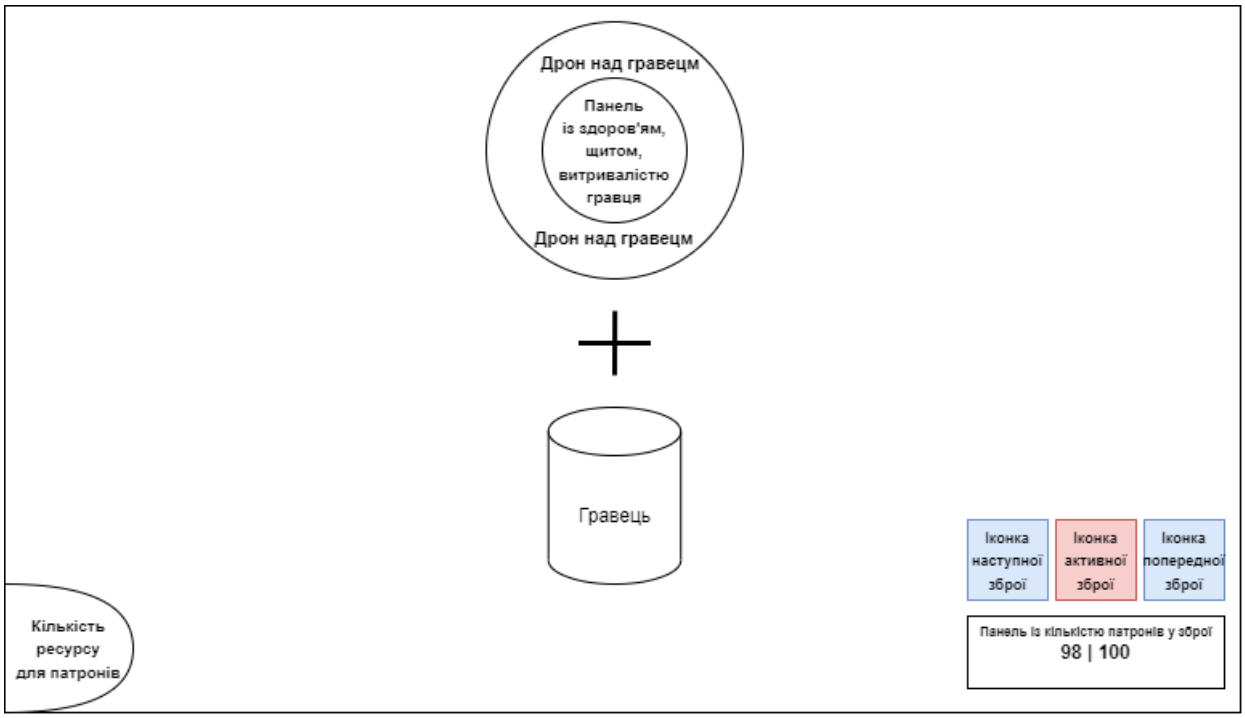


Рисунок 3.6 – Схематичне представлення UI гравця під час основного ігрового процесу (рисунок зроблено власноруч)

У правому нижньому куті розташовано ряд елементів пов’язаних із зброєю. Три іконки, які відображають те як гравець може міняти зброю. Середня виділена ікона відображує активну зброю гравця, а дві іконки по бокам наступну та попередню зброю. Нижче під іконками зброї панель яка відображає кількість патронів у активній зброї (поточна кількість патронів у магазині ліворуч, максимальна можлива кількість патронів у магазині поточної зброї праворуч). Над гравцем постійно літає дрон-помічник (пропорції гравця та дрона збільшені на схемі для розбірливості схеми), одна із його функцій відображати стан основних параметрів гравця (здоров’я, щит, витривалість). У лівому нижньому куту

відображається інформація про кількість ресурсу потрібного для створення патронів. Гравець може постійно переміщувати приціл по екрану за допомогою миші, приціл не є статичним.

Також окремо створено схему панелі яка знаходиться на дроні та відображає параметри персонажа (див рисунок 3.7).

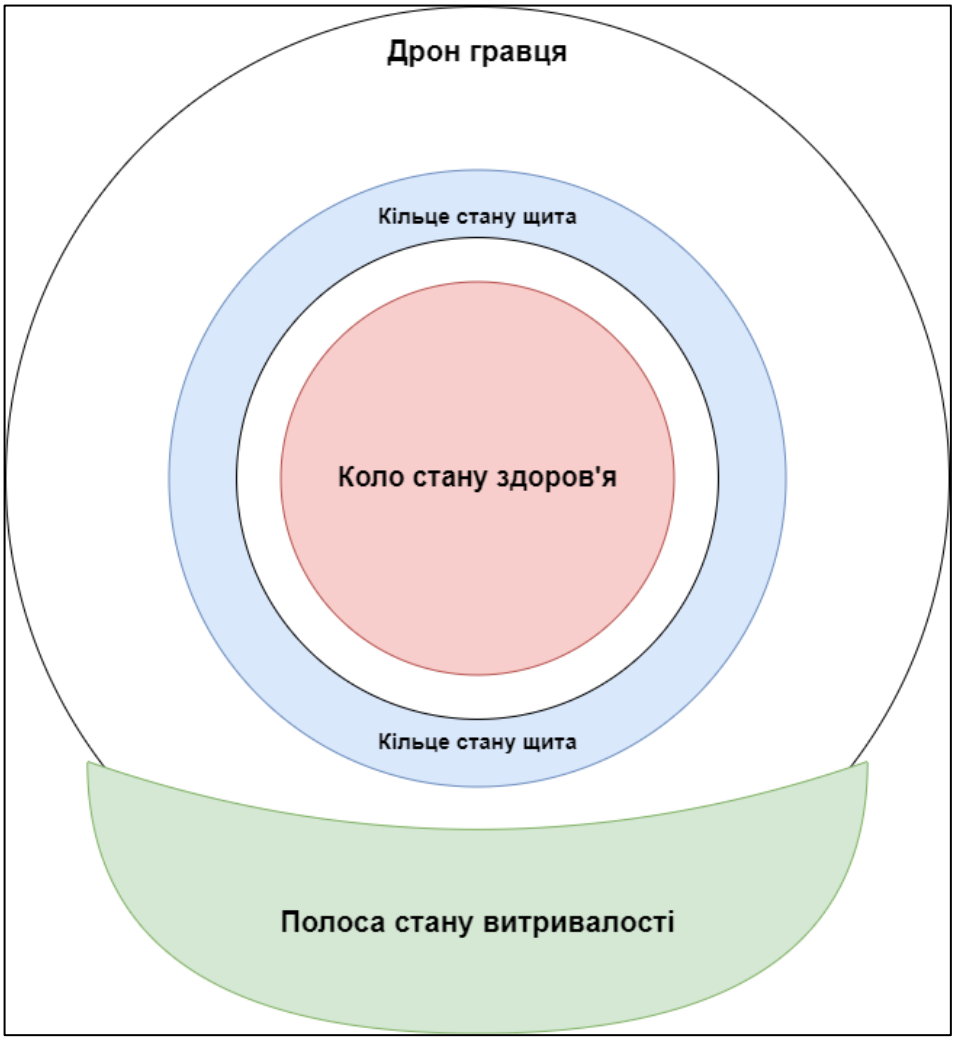


Рисунок 3.7 – схематичне представлення UI панелі для відображення параметрів гравця (рисунок зроблено власноруч)

Зовнішнє кільце відображає стан щита, внутріше коло стан здоров’я, а полоса знизу під кільцем щита відображає стан витривалості персонажу.

Панель із параметрами персонажа розміщено на дроні над гравцем для досягнення більш комфортного ігрового процесу гравця . Це дозволить гравцю не дивитись по кутам вишукуючи інформацію про стан персонажа та відволікаючись

від того щоб відбувається навколо персонажа. Оскільки гра ставить на меті створення динамічного ігрового процесу, це буде зручним рішенням, яке дозолить гравцю бачити одразу і персонажа та те що навколо нього, та стан цього персонажа. Слідкувати за станом персонажа є надважливим підчас гри, тому саме це елемент винесено окремо ближче до центру екрану для досягнення гарного ігрового досвіду. Також щоб гравцю було легше слідкувати чи є й нього у магазині зброї набої, буде створено ряд функцій, які дозволяють більш легко відслідковувати стан магазину, не дивлячись на те що панель з цією інформацією розташована у куті. Цифри панелі будуть ставати красного кольору та починати тремтіти для привернення уваги гравця. Також кожна зброя буде мати особливий звук її останнього пострілу, цей звук дасть гравцю розуміння що потрібно перезаряджати зброю.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Переміщення персонажу

Переміщення є однією з основних функцій персонажу. Воно відбувається завдяки клавішам «W» «A», «S», «D» при натисканні цих клавіш розраховується значення трьох вимірного вектору. «W» та «S» будуть змінювати значення за віссю Z зменшуючи та збільшуючи значення в залежності від зажатої кнопки, «W» збільшує значення, а «S» зменшує. «A» та «S» змінюють значення за віссю X аналогічно до попереднього прикладу. Цей вектор передається до «PlayerController» де вже виконується переміщення персонажу по у координатах ігрового світу. Зміна вектору переміщення також надає системі розуміння що гравець рухається внаслідок чого система включає відповідну анімацію персонажу.

Під час стрільби або прицілювання персонаж йде повільно та завжди дивиться у точку прицілу, тобто він може дивитись униз та йти спиною вперед або йти боком ліворуч дивлячись у ту саму точку. Але тут є проблема, якщо ми будемо просто викликати відповідні анімації то це буде виглядати погано, наче щось у грі зламалось. Бо наприклад коли гравець цілиться у праву частину екрану та йде ліворуч буде програватись анімація, як він йде боком ліворуч, а повинна була бути анімація руху спиною. Фактично персонаж переміщується дійсно ліворуч, але гра у такому випадку не знає куди персонаж дивиться, та просто включає анімації залежно від зміни вектору руху. Для того щоб гра розуміла напрям переміщення залежно від того куди дивиться персонаж було розроблено метод «CalculateMovementVector»:

```
private Vector3 CalculateMovementVector()
{
    float h = _direction.x;
    float v = _direction.z;

    Vector3 cameraR = cam.transform.right;
    Vector3 cameraF = cam.transform.forward;

    cameraR.y = 0;
    cameraF.y = 0;

    if (_isLShiftDown)
```

```
        {
            maxMovementMagnitude = 1.5f;
        }
        else
        {
            maxMovementMagnitude = 1f;
        }

        Vector3 movementVector = cameraF.normalized * v +
cameraR.normalized * h;
        movementVector = Vector3.ClampMagnitude(movementVector,
maxMovementMagnitude);

        float targetMagnitude = _isLShiftDown &&
playerStamina.GetStaminaPoints() > 0 ? 1.5f : 1f;
        float lerpedMagnitude =
            Mathf.MoveTowards(this.movementVector.magnitude,
targetMagnitude, 1.1f * Time.deltaTime);
        movementVector = movementVector.normalized * lerpedMagnitude;

        Vector3 relativeVector =
transform.InverseTransformDirection(movementVector);
        OnSend_X_Z_Pos?.Invoke(relativeVector.x, relativeVector.z);

        return movementVector;
    }
```

Метод обчислює вектор руху на основі переднього та правого напрямів камери, помножених на значення введення за віссю X та Z. Він комбінує два напрями, щоб отримати бажаний напрямок руху щодо камери.

Проведемо розрахунок вектору руху за формулою 4.1:

$$\vec{m} = \widehat{c_f} * v + \widehat{c_r} * h \tag{4.1}$$

де m – вектор руху;

c_f – вектор від камери направлений вперед;

c_r – вектор від камери направлений праворуч;

v – зміщення персонажа за віссю X;

h – зміщення персонажа за віссю Z.

2

Далі ми перетворюємо напрям цього вектору руху з світових (система координат відносно всієї ігрової сцени) координат у локальні (система координат конкретного об'єкту, у нашому випадку персонажа). Це дозволяє нам дізнатись

направлення руху персонажа відносно нього, а не сцени. Далі ми передаємо ці локальні значення осі X та Z у аніматор, за рахунок чого і програється необхідна нам анімація.

Важливо також правильно налаштувати аніматор нашого персонажа. Розглянемо налаштування Blend Tree[10] (елемент аніматора, який дозволяє нам поєднати анімації у одну групу та перемикати їх між собою залежно від значення встановлених до групи змінних) для групи анімацій, які вмикаються коли гравець цілиться або стріляє (див. рисунок 4.1) .

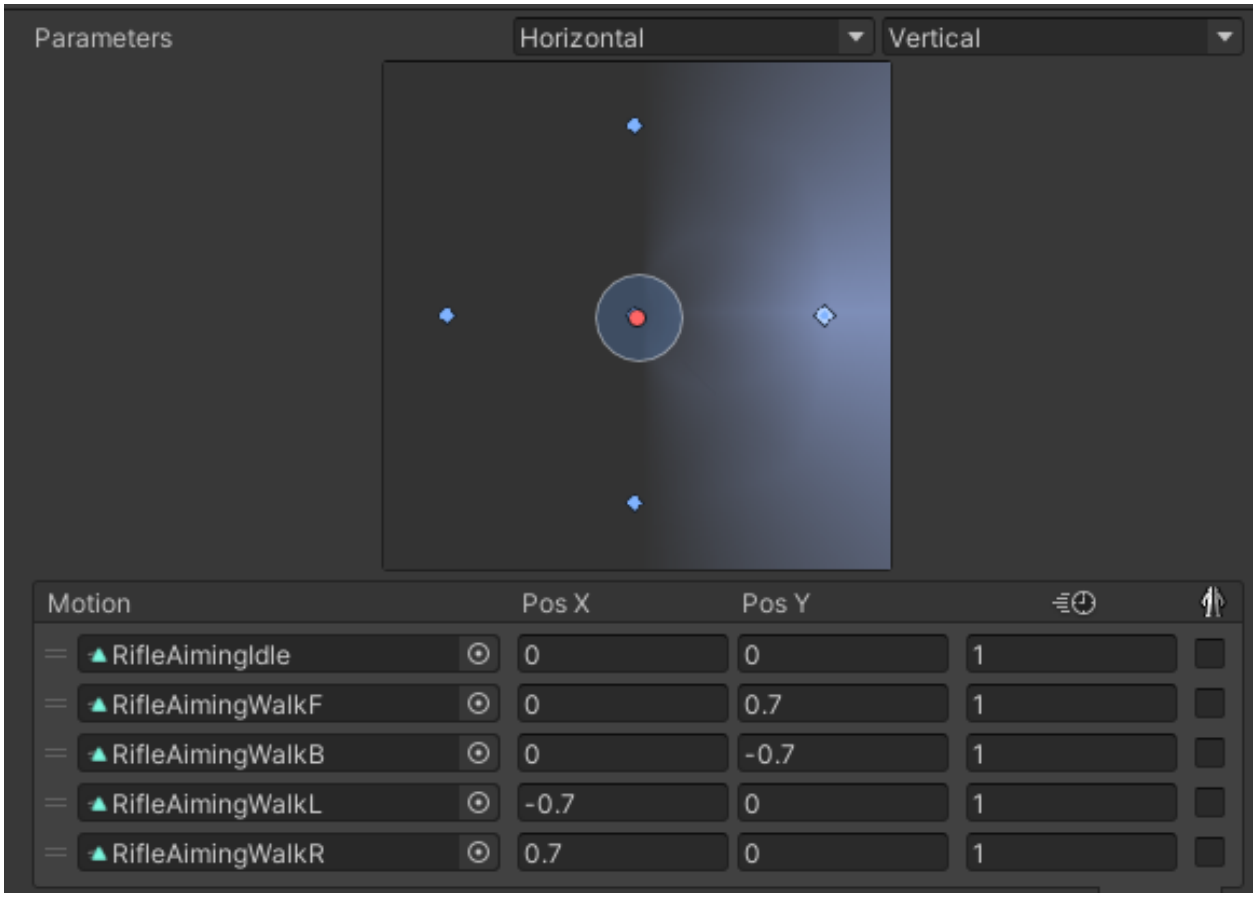


Рисунок 4.1 – налаштування Blend tree для анімацій руху під час прицілювання та стрільби (рисунок зроблено самостійно)

Тут ми бачимо цю групу анімацій, та значення при наближенні до яких будуть вмикатись відповідні анімації. Якраз тут наш аніматор і буде використовувати локальні значення X та Z, які ми передавали до цього та за рахунок них буде перемикати анімації. Важливою можливістю є те що аніматор

може об’єднувати деякі анімації коли значення знаходиться поміж ними.

4.2 Реалізація функціоналу зброї

Екземпляр зброї складається з ряду конфігураційних файлів, які є частинами головного такого файлу, який поєднує у собі усі інші.

Розглянемо як працює стрільба нашої зброї. За неї відповідає метод «TryToShoot». Подивимось на основну функціональну частину відповідальну за постріл:

```
     for (int i = 0; i < ShootConfig.BulletPerShot; i++)
     {
         Vector3 spreadAmount = ShootConfig.GetSpread(Time.time -
InitialClickTime);

         Ray ray =
ActiveCamera.ScreenPointToRay(Mouse.current.position.value);
         RaycastHit hit;

         if (Physics.Raycast(ray, out hit, Mathf.Infinity,
ShootConfig.HitMask))
         {
             Vector3 shootDirection = (hit.point -
ShootingStartPoint.transform.position) +

Model.transform.TransformDirection(spreadAmount);
             if (ShootConfig.IsHitScan)
             {
                 DoHitScanShoot(shootDirection, GetRaycastOrigin(),
ShootingStartPoint.transform.position);
             }
             else
             {
                 DoProjectileShoot(shootDirection.normalized);
             }
         }
         else
         {
             Vector3 shootDirection = ray.direction;
             if (ShootConfig.IsHitScan)
             {
                 DoHitScanShoot(shootDirection, GetRaycastOrigin(),
ShootingStartPoint.transform.position);
             }
             else
             {
                 DoProjectileShoot(shootDirection.normalized);
             }
         }
     }
```

Спочатку ми дивимось скільки набоїв потрібно випустити за один постріл та оброблюємо це циклом «for». Далі розраховуємо розкид набою (те як розраховується розкид буде розглянуто далі), та пускаємо луч від камери до точки куди цілиться гравець, у цю точку і будемо направляти пулю при пострілі далі. Перевіряємо чи є у цьому місці якийсь об’єкт. Якщо є то розраховуємо направлення польоту пулі на цей раз вже на основі точки звідки пуля вилетить та куди вона повинна прилетіти. Далі застосовуємо наш розкид пулі та випускаємо набої з зброї, перевіряючи яким саме методом. У грі реалізовано два методи пострілу. Постріл за рахунок промінню який майже моментально розрахує чи попадає постріл у ціль або ні, та постріл за рахунок об’єкту, який створить фізичний об’єкт пулі на сцені та зарахує попадання коли ця пуля зіштовхнеться із чимось.

Також у самому початку, при кожному пострілі цей метод перевіряє, чи можна вже починати новий постріл, дивлячись на те скільки часу пройшло з моменту минулого пострілу та опираючись на скорострільність для конкретного екземпляру зброї.

Так виглядає ця перевірка та запис останнього пострілу:

```
if (Time.time > ShootConfig.FireRate + LastShootTime)
{
      LastShootTime = Time.time;
}
```

Розкид набоїв ми розраховуємо передаючи у метод «GetSpread» час пострілу який пройшов з моменту начала пострілу. Цей час використовується для збільшення розкиду при веденні автоматичного вогню по цілі.

Розглянемо сам метод:

```
public Vector3 GetSpread(float ShootTime = 0)
{
Vector3 spread = Vector3.zero;

if (SpreadType == BulletSpreadType.Simple)
```

```
    {
        spread = Vector3.Lerp(
            new Vector3(
                Random.Range(-MinSpread.x, MinSpread.x),
                Random.Range(-MinSpread.y, MinSpread.y),
                Random.Range(-MinSpread.z, MinSpread.z)
            ),
            new Vector3(
                Random.Range(-Spread.x, Spread.x),
                Random.Range(-Spread.y, Spread.y),
                Random.Range(-Spread.z, Spread.z)
            ),
            Mathf.Clamp01(ShootTime / MaxSpreadTime) //Чем дольше спрей
тем больше разброс
        );
    }
    else if (SpreadType == BulletSpreadType.TextureBased)
    {
        spread = GetTextureDirection(ShootTime);
        spread *= SpreadMultiplier;
    }

    return spread;
    }
```

Для кожної зброї ми маємо заданий мінімальний та максимальний поріг розкиду за кожною віссю координат (у нашому випадку це: x, y, z). Використовуючи ці пороги ми генеруємо випадкові вектори у рамках цих максимального та мінімального обмеження. Далі ми лінійно інтерполюємо вихідний вектор між двома згенерованими випадкового (вектор діапазону мінімального розкиду та вектор діапазону максимального розкиду). Інтерполяція відбувається залежно від часу безперервної стрільби, чим довше цей час тим більше буде розкид.

Не менш важливим методом є «HandleBulletsImpact», цей метод відповідає за функціонал попадання патрону по противнику або якомусь іншому об’єкту.

Метод: «HandleBulletsImpact»:

```
    private void HandleBulletImpact(
        float DistanceTraveled,
        Vector3 HitLocation,
        Vector3 HitNormal,
        Collider HitCollider,
        int ObjectsPenetrated = 0)
    {
        SurfaceManager.Instance.HandleImpact(
```

```
                HitCollider.gameObject,
                HitLocation,
                HitNormal,
                ImpactType,
                0
            );

            if (HitCollider.TryGetComponent(out IDamageable damageable))
            {
                float maxPercentDamage = 1;
                if (BulletPenetrationConfig != null && ObjectsPenetrated > 0)
                {
                    for (int i = 0; i < ObjectsPenetrated; i++)
                    {
                        maxPercentDamage *=
BulletPenetrationConfig.DamageRetentionPercentage;
                    }
                }

        damageable.TakeDamage(DamageConfig.GetDamage(DistanceTraveled,
maxPercentDamage, chargedDamageMultiplier));
            }
        }
```

У першу чергу функція відтворює візуальний ефект від попадання патрону по будь-якому об’єкту. Це виконується за рахунок реалізованого самостійно класу «SurfaceManager». Цей менеджер бере колайдер[11] (це компонент, який визначає форму об'єкта з метою фізичного зіткнення) об'єкту, на цьому колайдері визначає точку попадання патрону, завдяки «HitLocation», та «HitNormal», параметрах переданих у метод. Далі в залежності від значення «ImpactType» програється відповідний до цього значення візуальний та звуковий ефект попадання по об’єкту.

Далі ми визначаємо чи не був об’єкт по якому ми попали - «IDamageable». Якщо цей об’єкт реалізовує цей інтерфейс, то він може отримати шкоду. Перевіряємо чи не було це попадання наслідком прострілу минулого об’єкту, якщо це не перший об’єкт у який улучив патрон, то ми множимо шкоду від набою на коефіцієнт який або зменшить шкоду чи навпаки збільшить, в залежності від налаштування зброї. Далі ми застосовуємо шкоду через метод «TakeDamage» нашого інтерфейсу «IDamageable».

4.3 Реалізація ефектів від попадання набою по об’єктах

Для реалізації цього функціоналу було створенню окремий клас

«SurfaceManager», він дозволяє нам визначати для різної зброї різні ефекти від попадання по об’єктах. При цьому ефект від попадання по різним текстурам буде різний якщо провести налаштування «SurfaceManager» та для конкретних типів попадання («ImapctType» – змінна що визначає тип попадання) зброї встановити відповідні ефекти попадання до відповідних текстур.

Розглянемо як працює цей клас. Обробка влучення в террейн[12] (це об'єкт, який є великою, плоскою поверхнею, яку можна редагувати для створення складних ландшафтів):

```
     if (HitObject.TryGetComponent<Terrain>(out Terrain terrain))
     {
         List<TextureAlpha> activeTextures =
GetActiveTexturesFromTerrain(terrain, HitPoint);
         foreach (TextureAlpha activeTexture in activeTextures)
         {
             SurfaceType surfaceType = Surfaces.Find(surface =>
surface.Albedo == activeTexture.Texture);
             if (surfaceType != null)
             {
                 foreach (Surface.SurfaceImpactTypeEffect typeEffect in
surfaceType.Surface.ImpactTypeEffects)
                 {
                     if (typeEffect.ImpactType == Impact)
                     {
                         PlayEffects(HitPoint, HitNormal,
typeEffect.SurfaceEffect, activeTexture.Alpha);
                     }
                 }
             }
             else
             {
                 foreach (Surface.SurfaceImpactTypeEffect typeEffect in
DefaultSurface.ImpactTypeEffects)
                 {
                     if (typeEffect.ImpactType == Impact)
                     {
                         PlayEffects(HitPoint, HitNormal,
typeEffect.SurfaceEffect, 1);
                     }
                 }
             }
         }
     }
```

У цьому блоці коду відбувається перевірка на те чи не влучили ми в террейн, та якщо це так, то далі йде обробка цього влучання. Ми беремо лист активних

текстур в точці попадання у террейн. Для кожної активної текстури ми шукаємо встановлений для неї ефект, якщо такого ефекту для якоїсь текстури немає, то ми програємо стандартний ефект влучання.

Далі якщо влучання було не у террейн, то ми перевіряємо чи не є це об’єкт із рендер компонентом[13]. Далі відбувається аналогічний як і у випадку с террейном процес. Ми перевіряємо текстури у точці влучання та програємо відповідний до текстур ефект, якщо такий є.

Таким чином цей метод дає змогу гнучко налаштовувати ефекти від влучання у різні текстури та для різних видів попадань (наприклад: звичайна пуля, граната, вогонь). Метод підтримує влучання як у террейни так і у звичайні об’єкти з рендером, забезпечуючи універсальну обробку усіх можливих поверхонь у грі.

Також «SurfaceManager» дозволяє при потребі працювати не тільки із улучанням набоїв по об’єктах. Система побудовано спеціально таким чином щоб буди універсальною. Вона може програвати ефекти як візуальні так і звукові і від інших взаємодій між фізичними об’єктами. Наприклад якщо реалізувати передачу даних про місце дотику ноги персонажа із поверхнею, то «SurfaceManager» зможе програвати ефекти від шагів гравця по ландшафту. Це роби систему унікальною та дуже корисною, вона дозволяє насичить гру різноманітними ефектами що буде додавати краси та реалістичності для гри.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування ігрового застосунку

Тестування нашого ігрового застосунку проводилось згідно з розробленим заздалегідь тест-планом, який наведено у додатку Г. Для реалізації тест плану було створено Mind Map (див. рисунок 5.1).

Усі тести проводились за методом «чорного ящику» (Blackbox).

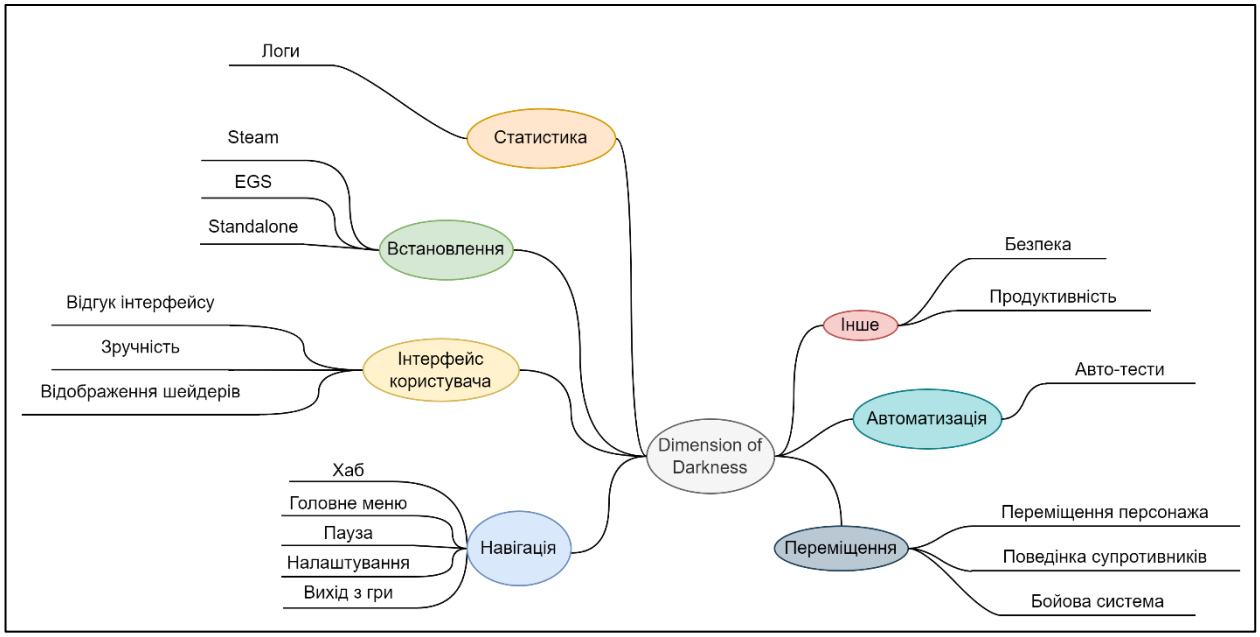


Рисунок 5.1 – Mind Map для реалізації тест-плану

«Діаграма думок» дає нам змогу розподілити проект на декілька частин:

- навігація;
- UI ігрового застосунку;
- статистика;
- автоматизація;
- переміщення та бойова система;
- інше
- встановлення

Таке розділення за частинами дозволить нам виконати тестування проекту поетапно.

5.2 Виявленні помилки у виконанні коду

За допомогою тест-плану, Mind Map, було проведено ряд тестів ігрового застосунку. Після чого було сформовано таблицю із виявленими помилками у виконанні коду (див. додаток Д).

Таблиця містить таку інформацію про кожну помилку:

- опис помилки;
- назва помилки;
- компонент системи;
- пріоритет;
- критичність;
- кроки відтворення;
- очікування результат;
- фактичний результат.

У цій таблиці наведено ряд тестів, які було проведено під час тестування проекту. Кожна знайдена помилка має свій пріоритет та критичність. Де найменший рівень критичності та пріоритету P1 та S1, а найбільший рівень P4 та S4 відповідно.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Соціальне впровадження гри є важливим з різних причин. Гра, яка має сильну соціальну складову, може створити активну спільноту гравців. Це може призвести до підвищення вірності гравців, оскільки вони відчувають відношення до інших учасників спільноти. Коли гравці діляться своїми досягненнями або враженнями від гри в соціальних мережах, це може служити потужним інструментом вірального маркетингу. Соціальні медіа та форуми можуть служити важливими каналами для збору відгуків від гравців. Розробники можуть використовувати ці відгуки для вдосконалення гри. Соціальні медіа можуть служити ефективним каналом для надання підтримки гравцям, вирішення проблем та відповідей на запитання.

Отже, ведення соціальних мереж для проекту є одним із ключових моментів, оскільки це може допомогти залучити більше уваги до вашого проекту. Соціальні мережі є потужними каналами комунікації, які дозволяють вам досягти широкої аудиторії. Їх можна використовувати для розповсюдження новин про ваш проект, демонстрації прогресу та залучення гравців до обговорення. Ведення соціальних мереж для проекту може бути важливим інструментом для його просування та успіху.

Для успішного впровадження відеогри в сучасному ринковому середовищі важливо розробити ефективну маркетингову стратегію.

Після впровадження гри слід уважно відслідковувати відгуки гравців. Аналізувати їх коментарі, виявляти слабкі місця гри та вносити відповідні зміни. Відгуки можуть бути цінним джерелом інформації для подальшого вдосконалення.

Для нашого проекту було створено декілька сторінок у соціальних мережах.

А саме :

- Instagram
- Telegram
- YouTube
- Facebook
- Twitter

У кожній соціальній мережі постійно підтримувалась активність. Весь час публікувалися новини щодо прогресу розробки проекту. Приводились опитування серед аудиторії гри, стосовно того що гравцям більше б сподобалось у грі, до яких моментів варто більше докласти зусиль на думку гравців тощо. Найбільш активна спільнота була у мережі Telegram, тому там найчастіше проводились опитування серед користувачів нашого каналу у мережі. А найбільш ексклюзивні публікації були у мережі YouTube, там було не мало відео про процес розробки гри, а також про її концепцію та майбутні механіки. Найменша активність спостерігалась у Facebook тому, більша частина аудиторії гри не користувалась цією соціальною мережею. Через це було прийнято рішення не робити для Facebook особливі пости, а просто дублювати туди пости із інших мереж. Twitter та Instagram показали середню активність гравців, там здебільшого публікувалися новини, але до того у Instagram також часто проводились опитування аудиторії. Враховуючи що для проекту не було ніяких маркетингових компаній, та він набирав аудиторію виключно з соціальних мереж, на даний момент можна сказати що соціальне впровадження було частково вдалим. Вдалим бо проект зміг привернути увагу нової аудиторії, але частково вдалим бо цієї нової аудиторії було не так багато як могло би бути.

Успіх проекту залежить від впровадження ефективних маркетингових стратегій та відкритості до змін на основі відгуків гравців, та у цілому у взаємодії із вашою головною аудиторією.

ВИСНОВКИ

Під час виконання цієї кваліфікаційної роботи бакалавра було розроблено бойову систему для демонстраційного ігрового застосунку у жанрі shoot`em up. Проект було розроблено з рушієм Unity3D 2022, мові програмування C# для ОС Windows 10/11.

Для реалізації бойової системи та системи взаємодії гравця з персонажем було проведено аналіз предметної галузі. Аналіз проводився з точки зору проектів з аналогічним жанром або механіками, також проводився аналіз ринку відеоігор за жанрами та аналіз цільової відеоігор за останній час.

Було сформовано перелік вимог до систем гравця. Перелік вимог надає розуміння концепції проекту та нюанси що стосуються його створення, у технічному сенсі.

Під час планування архітектури проекту було детально розглянуто, як саму бойову систему та систему взаємодії із персонажем, а також STFCW. Для розуміння архітектури системи було створено ряд діаграм, які відображають взаємодію елементів у системі. Діаграми були створені як для систем персонажа так і для STFCW.

Бойова система та STFCW можуть бути легко модифіковані та покращені. STFCW може бути розширена за рахунок додавання нових параметрів для зброї та нового функціоналу, для цього проведено роботу з структуризації коду та STFCW елементів для більш простого покращення систему у разі потреби.

Сама бойова система може бути покращена за рахунок додавання підтримки інших пристроїв введення сигналів, додавання більшого різноманіття анімація, для цього у головному конфігураційному файлі STFCW передбачено елемент «Animator Override Controllers» [14], завдяки цьому елементу можна легко міняти анімації до кожного виду зброї.

Враховуючи вищезазначене, можна зробити висновок, що розроблена бойова система для ігрового застосунку у жанрі shoot’em up є ефективною та гнучкою. Вона дозволяє легко модифікувати та покращувати існуючі механіки, що робить її

адаптивною до змін у вимогах та тенденціях ринку.

Система взаємодії гравця із персонажем відповідає необхідній реактивності управління. Персонаж швидко відгукається на команди гравця при цьому він не здається, занадто легким або важким, що дає відчуття наче це справжня людина у сенсі фізичних властивостей тіла. Також система вводить усі необхідні обмеження для персонажу щоб все виглядало автентично (наприклад персонаж не може стріляти під час перекочування або перезаряджання, що було би просто не логічна).

Візуально усі дії персонажа також виглядають автентичними та коректними. Анімації добре налаштовані, візуальні ефекти від пострілів та влучань набоїв є різноманітними та яскравими, гарними. Звукові ефекти також є різноманітними для різних екземплярів зброї, гра має усі необхідні звукові ефекти (від шагів гравця й пострілів, до звукового супроводу).

Також створено гнучку систему – «SurfaceManager» – для програвання візуальних та звукових ефектів від попадання набоїв по якомусь об’єкту. Важливою можливістю цієї системи є те що вона може використовуватись також і для інших механік (наприклад візуальні і звукові ефекти від шагів персонажа).

Для проекту проведено залучання нової аудиторії завдяки веденню сторінок різноманітних соціальних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wikipedia Shoot`em up genre [Електронний ресурс]: https://uk.wikipedia.org/wiki/Shoot_%27em_up (дата звернення: 12.12.2023).

2. Free report: Newzoo's Global Games Market Report 2023 - 2024 [Електронний ресурс]: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version (дата звернення: 15.12.2023).

3. Gamer Demographics from 2024 [Електронний ресурс]: https://playtoday.co/blog/stats/gamer-demographics/ (дата звернення: 22.12.2023)

4. Earnest Video Game Demographics [Електронний ресурс]: https://www.earnest.com/blog/the-demographics-of-video-gaming/ (дата звернення: 24.07.2023).

5. ESA 2020 Essential Facts About the Video Game Industry [Електронний ресурс]: https://www.theesa.com/2020-essential-facts/ (дата звернення: 17.01.2024).

6. Unity3D Input System [Електронний ресурс]: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.8/manual/index.html (дата звернення: 25.01.2024).

7. Unity3D Animator component [Електронний ресурс]: https://docs.unity3d.com/Manual/class-Animator.html (дата звернення: 15.02.2024).

8. Handle and raise events C# [Електронний ресурс]: https://docs.unity3d.com/Manual/class-Animator.html (дата звернення: 24.02.2024).

9. Unity 3D Scenes [Електронний ресурс]: https://docs.unity3d.com/Manual/CreatingScenes.html (дата звернення: 24.02.2024).

10. Unity Blend Trees [Електронний ресурс]: https://docs.unity3d.com/Manual/class-BlendTree.html (дата звернення: 17.03.2024).

11. Unity Collider [Електронний ресурс]: https://docs.unity3d.com/ScriptReference/Collider.html (дата звернення: 26.03.2024).

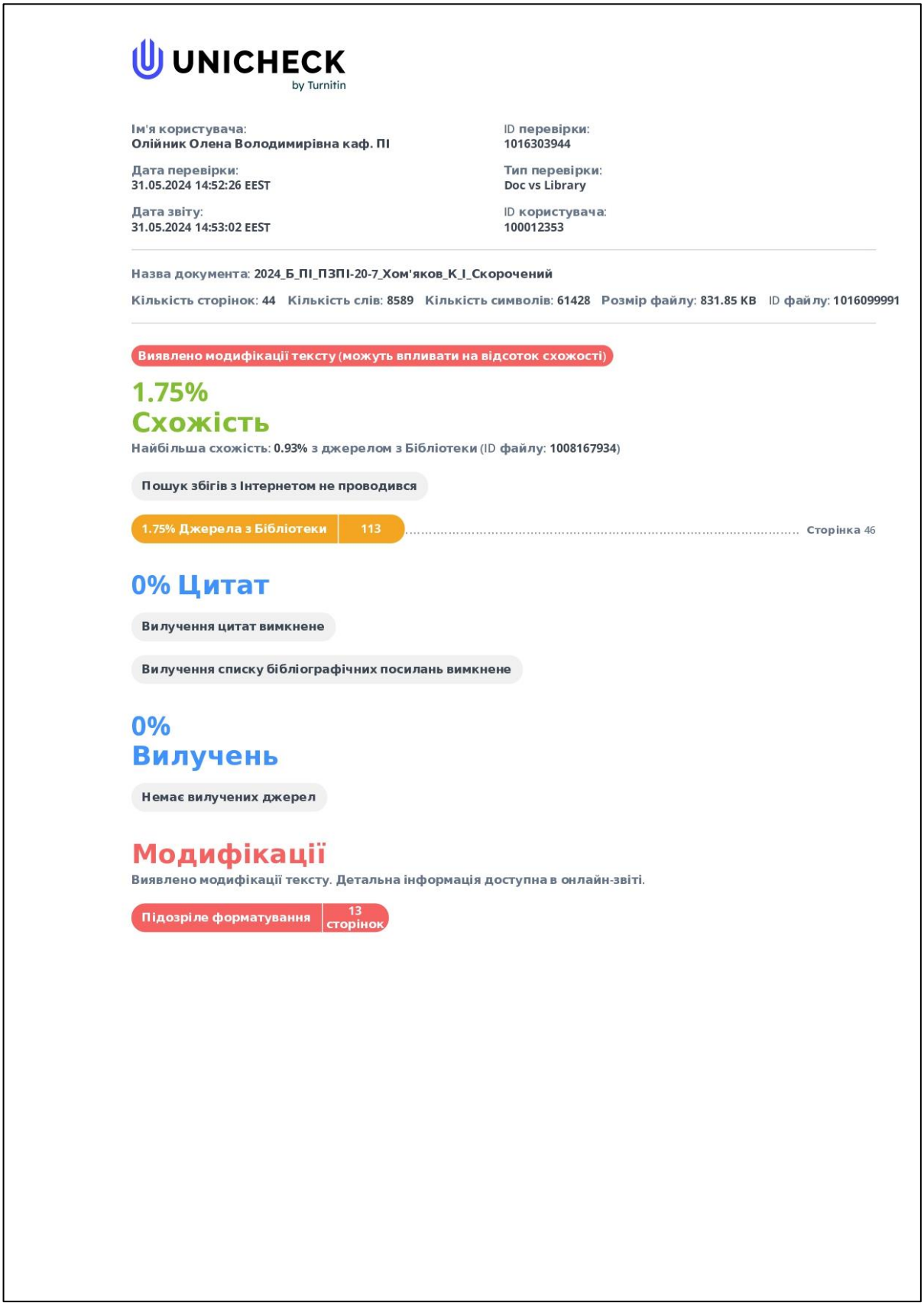
12. Unity Creating and editing Terrains [Електронний ресурс]: https://docs.unity3d.com/Manual/terrain-UsingTerrains.html (дата звернення: 12.04.2024).

13. Unity Mesh Renderer component [Електронний ресурс]: https://docs.unity3d.com/Manual/class-MeshRenderer.html (дата звернення: 27.042.2024).

14. Unity3D Animator Override Controllers [Електронний ресурс]: https://docs.unity3d.com/Manual/AnimatorOverrideController.html (дата звернення: 05.04.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016303944

Дата перевірки:
31.05.2024 14:52:26 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
31.05.2024 14:53:02 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-7_Хом'яков_К_І_Скорочений

Кількість сторінок: 44 Кількість слів: 8589 Кількість символів: 61428 Розмір файлу: 831.85 KB ID файлу: 1016099991

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.75%
Схожість

Найбільша схожість: 0.93% з джерелом з Бібліотеки (ID файлу: 1008167934)

Пошук збігів з Інтернетом не проводився

1.75% Джерела з Бібліотеки 113 ..... Сторінка 46

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 13 сторінок

Рисунок А.1 – Звіт Unicheck

ДОДАТОК Б

Концепт-документ до гри

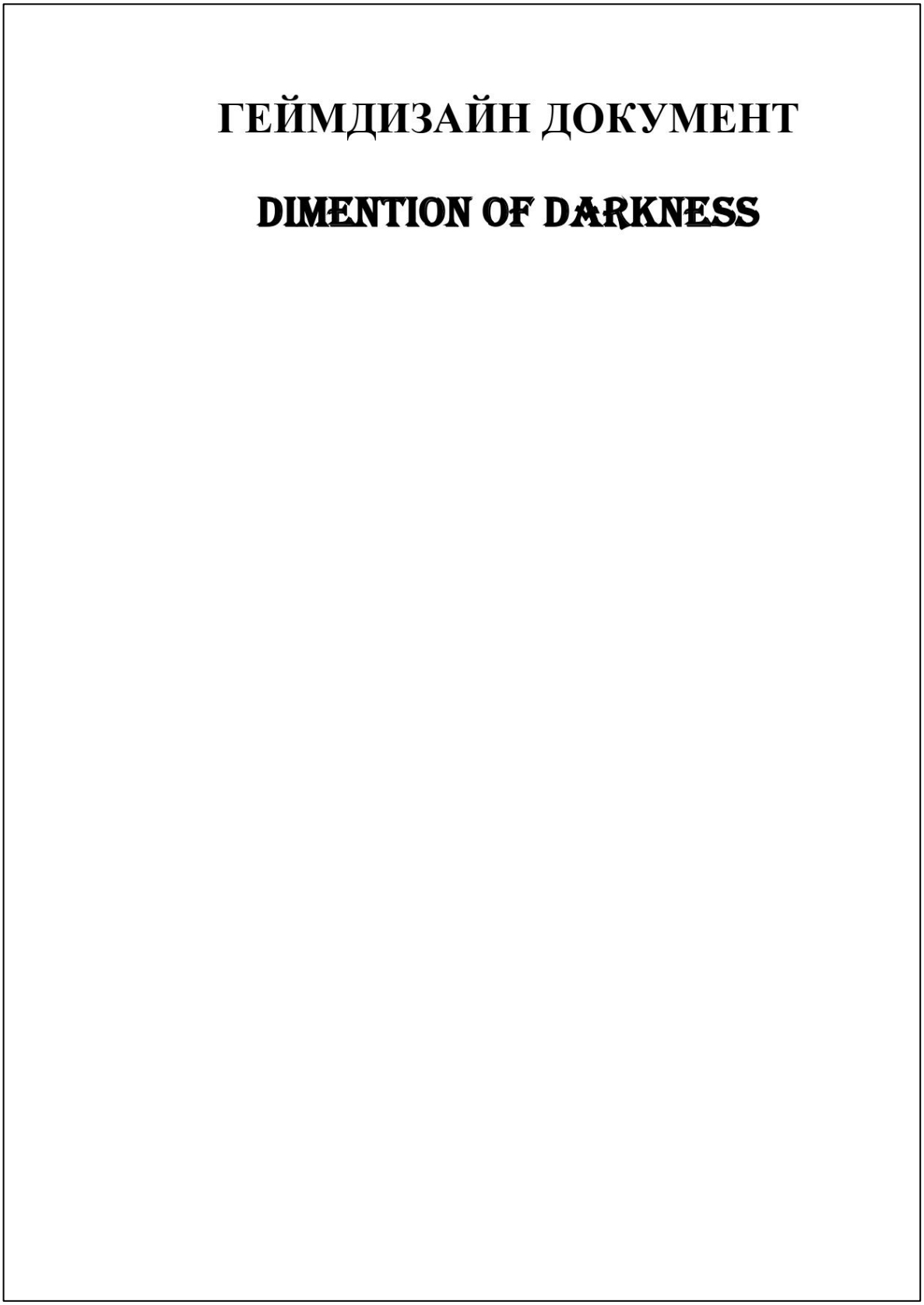


Рисунок В.1 – Сторінка 1 концепт-документу до гри

1 СПЕЦИФІКАЦІЯ

1.1 Тетра

Механіка: Shoot`em-up

Технологія: PC

Історія: ГГ-науковець

Естетика: Fantastic + Darkness

1.2 Цільова аудіторія

Головною аудиторією нашої будуть чоловіки 16-40 років. Цю аудиторію мають привабити: динамічна система бою, графічні ефекти зброї, кооператив, можливість безмежно розвивати персонажа.

1.3 USP

- Спробуйте себе у ролі відчайдушного героя у невідомому темному вимірюванні, який намагається повернутись до дому;
- отримайте задоволення від динамічної бойової системи, знищуючи ворогів;
- не майте обмежень по рівню розвинення персонажа;
- безмежно знищуйте ворогів їдучих постійним натиском на вас.

1.4 Час ігрової сесії

Гра не має рівнів або місій тощо. Гра складається із забігів. Кожен забіг, на початковій стадії гри, може продовжуватись від 5 до 30 хвилин. З подальшим прогресом гравця середній час забігу буде збільшуватись.

Рисунок В.2 – Сторінка 2 концепт-документу до гри

2 ГЕЙМПЛЕЙ

Головною метою гравця за гру буде назбирати достатню кількість енергії, за один забіг, щоб відремонтувати пристрій який допоможе повернутись нашому герою додому. Гравцю треба назбирати необхідну кількість енергії саме за один забіг, бо поміж забігами енергія накопичуватись не буде та при старті нового забігу стара енергія знищується. Це і є основне випробування для гравця. Також йому буде заважати збирати енергію постійний натиск потвор. Щоб мати змогу назбирати достатньо енергії за один раз, гравцю треба зробити багато забігів щоб вдосконалити зброю костюм тощо.

Гра ділиться на два етапи: забіг та відпочинок на базі. Забіг є основним етапом під час якого гравець повинен отримати основне задоволення від гри. Під час відпочинку на базі гравець має змогу: витратити зібрані ресурси на вдосконалення спорядження, відпочити та зберегти гру.

Рисунок В.3 – Сторінка 3 концепт-документу до гри

3 ІСТОРІЯ ТА СЮЖЕТ

3.1 Вступ

Гра не є сюжетно орієнтованою, але гра має історію та мінімальний сюжет щоб гравець розумів логіку та причини того що відбувається під час гри.

3.2 Історія

2087 рік, вчені у науково-дослідницькому центрі досліджують паралельні вимірювання та можливості із ними взаємодіяти. Одного разу 32-річному вченому вдається винайти пристрій який може розривати простір та створювати стабільний портал із іншим вимірюванням, але він навіть не має здогадів що можна очікувати від того вимірювання. Цікавість бере верх над групою вчених та вони віршують розпочати експеримент по запуску зв'язуючого пристрою, не отримавши на це дозвіл від керівництва та не попередивши інших вчених.

Десяте листопада 2087 року, зв'язуючий пристрій під назвою PSDC (Possibly a Stable Dimensional Connector) не справляється з енергетичним навантаженням та бокові опори які тримали портал ламаються, портал одразу збільшується у розмірі та доходячи до розмірів усієї лабораторії, PSDC вимикається через перенавантаження, в наслідок чого лабораторія опиняється у іншому вимірювання. Дослідники мають деяке обладнання та генератор у лабораторії що дозволить їм прожити деякий час. Дослідникам треба відремонтувати PSDC тому вони вирішують відправитись за межі лабораторії для досліджень та вирішення проблеми.

Під час першого виходу з лабораторії, Уолтер, перший дослідник доброволець, бачить перед собою всюди тьму та густий також темний туман. Трохи озирнувшись Уолтер побачить дивне створіння схоже чи на павука чи на людину та воно вочевидь було налаштоване вороже. Зреагувавши Уолтер вбиває це створіння із пістолета, який йому видали. Трохи дослідивши довкілля, він розуміє, завдяки його спорядженню він

Рисунок В.4 – Сторінка 4 концепт-документу до гри

може збирати з довкілля та створінь що тут живуть енергію, це допоможе вченим відремонтувати PSDC та повернутись до дому. Також потвори що тут живуть, як помітив Уолтер, створені із різних матеріалів що дозволяє використовувати їх залишки як матеріал для АНТ (atomic handheld transducer), це пристрій який є майже у кожного у 2087 році, він може розбирати речі на окремі атоми та створювати з них те що хоче власник пристрою.

Повернувшись до лабораторії Уолтер помічає що ємності його рюкзаку для збору енергії недостатньо щоб назбирати енергії за раз. Але що набагато гірше Уолтер розуміє темна енергія яку він назбирав дуже швидко анігілює з тою енергією що телепортувалась в це вимірювання. Через це вчені не мають змоги просто накопичувати енергію, значить їм потрібно вдосконалити рюкзак Уолтера щоб він міг принести багато енергії за раз.

Подальшою метою Уолтера є зібрати достатню кількість ресурсів щоб вдосконалити своє обладнання та відремонтувати PSDC щоб повернутись додому, але створіння що живуть у цьому вимірюванні не дадуть йому це так просто зробити.

Рисунок В.5 – Сторінка 5 концепт-документу до гри

4 МЕХАНІКИ

4.1 Механіки гравця

Поява (у забігу)

Для того, щоб почати забіг, гравцеві (усім гравцям) необхідно зайти в телепорт. Після того почнеться сам забіг.

Поява (на базі)

Після того, як гравець у забігу загине, він опиниться на базі. Кількість твердотільного матеріалу скидається до 0.

Ривок та стрибок

Ривок дозволяє гравцеві практично моментально переміститися на певну дистанцію у напрямку руху гравця. Гравець може зробити ривок, натиснувши на Space. Ривок витрачатиме певну кількість витривалості.

Отримання твердотільного матеріалу та темної енергії з убитих супротивників

Після смерті противника, на рахунок гравця переходитиме певна кількість темної енергії, а також із противника може випасти якась кількість твердотільного матеріалу, з якого гравець зможе створити нові предмети.

Рисунок В.6 – Сторінка 6 концепт-документу до гри

4.2 Механіки супротивника

Спавн

Вороги з'являтимуться навколо гравця, поза його полем зору. Частота появи противників залежить від часового інтервалу, а також від просування гравця по карті. Також будуть додаватися/мінятись типи противників, це залежатиме від просування гравця по карті та кількості накопичених одиниць темної енергії під час забігу.

Атака

Вороги зможуть використовувати різні види атак, такі як удари у ближньому бою, стрілянина зі зброї, кидання снарядів, магічні атаки тощо. Атаки деяких супротивників матимуть певний патерн. Залежно від типу ворога, буде змінюватися швидкість/збиток атаки. Також існуватимуть особливі атаки, наприклад атаки по області, уповільнення, зменшення огляду тощо.

Смерть

Смерть ворога супроводжуватиметься анімацією та/або візуальними ефектами. Після смерті противника на їхньому місці з'являтиметься ресурс, який гравець може підібрати та використати надалі. Також смерть противників може супроводжуватися певними особливими ефектами смерті, наприклад вибухом.

Рисунок В.7 – Сторінка 7 концепт-документу до гри

4.3 Механіки світла та темряви

Механіка освітлення

Направляючи світло на туман, гравець буде розсіювати його, поки не перенаправить світло в іншу область. У освітленій області не можуть сповнитися супротивники. Для того, щоб висвітлити якусь область, існують певні предмети, що витрачаються, або поліпшення.

Механіка туману

У всіх місцях карти, які гравець ніяк не висвітлює, він бачитиме чорний туман. Навіть якщо гравець одного разу вже висвітлив область, то, переставши висвітлювати область, до неї повернеться туман.

4.4 Механіки зброї

Перезаряджання

Практично у всієї зброї в грі є свій час на перезарядку після пострілу, при цьому найчастіше зброя уповільнюватиме гравця під час перезарядки, а такі рухи як біг або стрибки через перешкоди збиватимуть таймер перезарядки, повертаючи його до початку відліку.

Заряд пострілу

Деяка зброя у грі завдає постріли після процесу заряду певної тривалості. Заряд здійснюється затисканням ЛКМ, після того, як гравець відпустить ЛКМ зброю зробить удар/постріл. Більшість зброї у грі з подібною механікою матиме максимальну межу заряду для удару/пострілу. Чим більшої потужності заряд, тим більшої сили буде постріл.

Рисунок В.8 – Сторінка 8 концепт-документу до гри

Простріл противників

Деяка зброя у грі прострілюватиме супротивників наскрізь, наприклад, така зброя, як снайперська гвинтівка. При пострілі, куля з такої зброї пробиватиме всіх супротивників на своєму шляху, зупиняючись (зникаючи) на певній відстані.

4.5 Механіки костюму персонажа

Загальні відомості

Енергетичний щит

Костюм має вбудований енергетичний щит, він має певну кількість очок, які будуть забиратися при отриманні збитків від супротивників. Після того, як щит буде розряджений, гравець почне втрачати очки здоров'я від отриманої шкоди. Щит починає автоматично поступово заряджатися до максимального заряду поки гравець не отримує шкоди від супротивників певний час. Отримання втрат під час заряду щита зіб'є процес зарядки, кількість очок зберегтися на тому рівні, де зупинилася зарядка.

Рисунок В.9 – Сторінка 9 концепт-документу до гри

5 ІНТЕРФЕС КОРИСТУВАЧА

5.1 HUD

На екрані гравець бачить:

- рівень свого здоров’я, витривалості та енергощита костюма;
- слоти зі зброєю та обрану зброю, а також максимальну та фактичну кількість патронів у магазині.;
- фактичну та необхідну для підвищення рівня гри кількість темної енергії.

5.2 Система контролю

Гравець має доступ до наступного контролю:

- Переміщення – [W][A][S][D];
- оглянутись навколо – миша;
- ривок/перестрибнути перешкоду – [Space];
- біг – [Shift];
- постріл/прицілювання – [LMB]/[RMB];
- особлива атака зброї – [Q];
- взаємодія із інтерактивними предметами – [E];
- меню паузи – [Esc].

5.3 Аудіо ефекти

Гра буде музичний супровід. Звукові ефекти від пострілів та заряду зброї, противники та персонаж також будуть мати свої звукові ефекти. А також деякі предмети навколо будуть створювати звукові ефекти.

Рисунок В.10 – Сторінка 10 концепт-документу до гри

6 РЕФЕРЕНСИ

6.1 Головні герої

Усі виглядають приблизно однаково. Білий костюм-екзоскелет, зріст десь 1.8м

Рисунок В.11 – Сторінка 11 концепт-документу до гри

Рисунок В.12 – Сторінка 12 концепт-документу до гри

Рисунок В.13 – Сторінка 13 концепт-документу до гри

6.2 Противники

Повністю чорні створіння, місяцями схожі на людей, місцями на тварин.

Замість очей та роту чорні впадини.

Рисунок В.14 – Сторінка 14 концепт-документу до гри

Рисунок В.15 – Сторінка 15 концепт-документу до гри

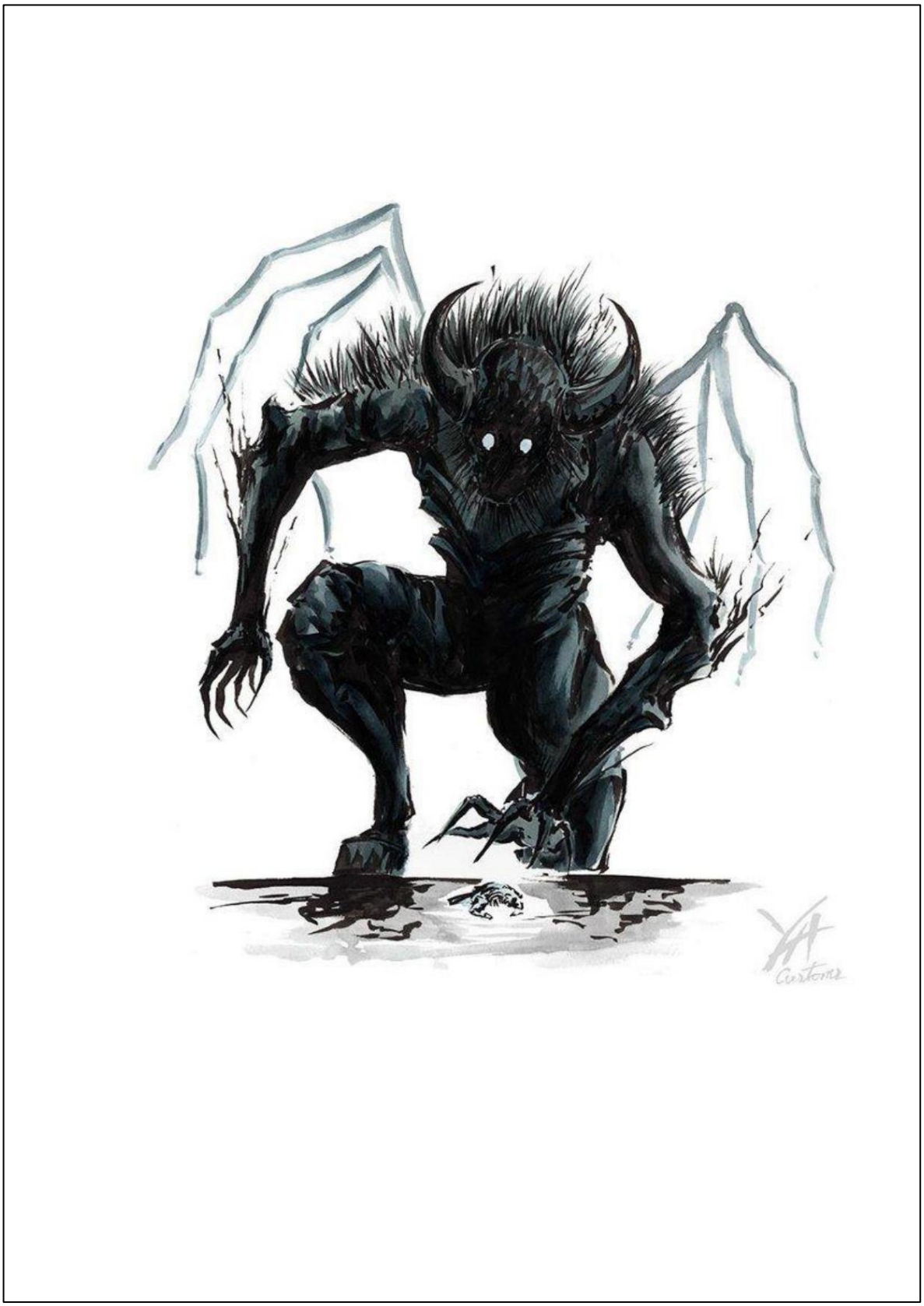


Рисунок В.16 – Сторінка 16 концепт-документу до гри

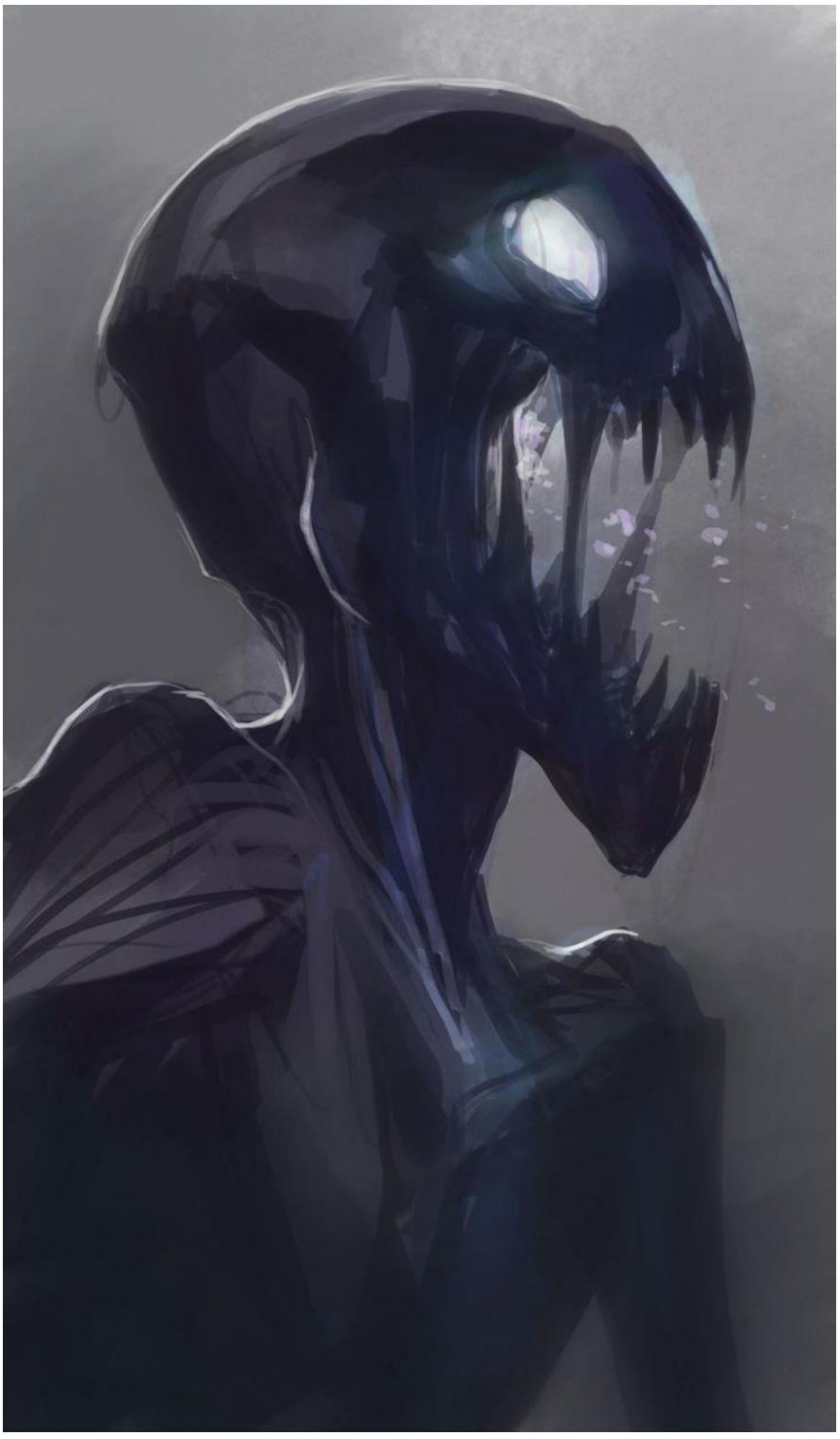


Рисунок В.17 – Сторінка 17 концепт-документу до гри

6.3 Біом

Перший біом це звичайна земля рожево-коричневого кольору. По краям біомів будуть розломи, поламані мости, автомобілі та інше. Вид згори.

Рисунок В.18 – Сторінка 18 концепт-документу до гри

Рисунок В.19 – Сторінка 19 концепт-документу до гри

6.4 Лабораторія

Приміщення з великою кількістю поламаної техніки, такої як комп'ютери, прінтери, якісь колби і так далі. В лабораторії будуть місцями горстки землі, із полу стирчатимуть сталагміти. Задня частина лабораторії буде завалена та зливатиметься із землею. Тільки вид згори.

Рисунок В.20 – Сторінка 20 концепт-документу до гри

Рисунок В.21 – Сторінка 21 концепт-документу до гри

Рисунок В.22 – Сторінка 22 концепт-документу до гри

6.5 Туман

Туман буде всюди, окрім шляху де світитиме ліхтар, або освітлювальні предмети.

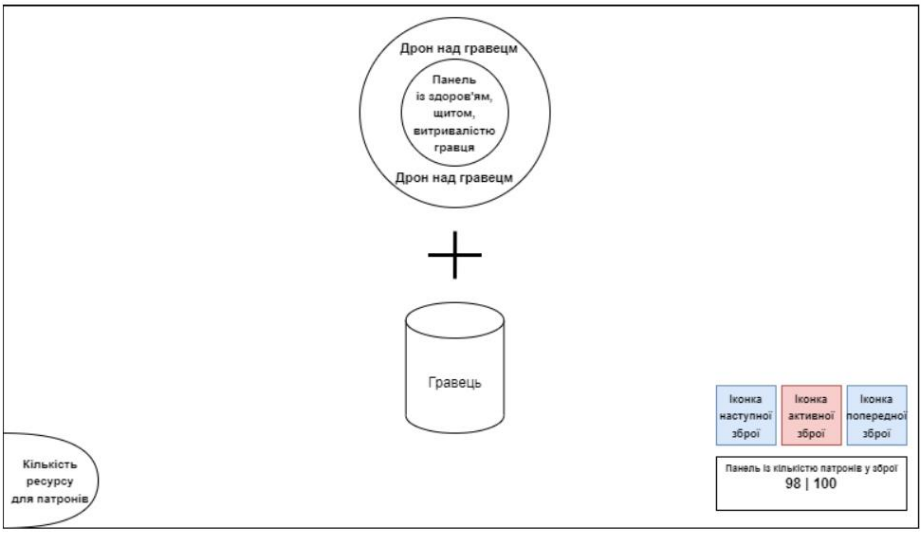
6.6 Головне меню

Головне меню буде складатися із фону, та кнопок «Одиночна гра», «Мультиплеєр», «Налаштування» та «Вихід».

Рисунок В.23 – Сторінка 23 концепт-документу до гри

6.7 Меню налаштувань

6.8 Ігровий інтерфейс



7 РОЗГОРТАННЯ ТА РОЗРОБКА

7.1 Порядок встановлення

Рисунок В.24 – Сторінка 24 концепт-документу до гри

Гру можливо буде інсталювати через інсталятор або через Steam/EGS

7.2 Системні вимоги

Операційна система: Windows 8 або кращє.

Процесор: Intel 2.77GHz Quad-core.

ОЗП: 8 Gb.

Відеокарта: NVIDIA GTX 550 Ti.

DirectX: 10.

Місце на диску: до 1Gb.

8 РОЗРОБКА

Гра буде розроблена на рушії Unity.

Репозиторій: https://github.com/Dayman267/ZombieSurviveSmash1.

Рисунок В.25 – Сторінка 25 концепт-документу до гри

ДОДАТОК В

План тестування

Назва проекту

Dimension of Darkness

План випробувань і тестування

Test Plan

Версія 1.0

Version 1.0

Рисунок Г.1 – Сторінка 1 плану тестування

Історія змін (Revision History)

Дата	Версія	Детальний опис	ПІБ Автора
18.03.2024	1.0	Створення тест-плану	Попов Д.М.
			Хом’яков К.І.
			Луценко В.В.

Рисунок Г.2 – Сторінка 2 плану тестування

1. ВСТУП

1.1 Мета

Метою складання даного тест плану є опис процесу тестування гри «Dimension of Darkness». Мета документу - координація роботи процесу розробки у сфері контролю якості продукту. Документ призначений групі тестування для ознайомлення з характером майбутніх робіт, аналізу і розбиття на підзадачі. Документ дозволяє отримати уявлення про заходи з тестування проекту

1.2 Довідкова інформація

Уолтер - головний герой, який опинився у паралельному вимірі після невдалого експерименту. У процесі дослідження цього виміру він виявляє темну енергію, яка може допомогти йому повернутися додому. Упродовж гри гравець досліджує нове оточення, бореться з небезпечними істотами та збирає ресурси для повернення додому, але він має подолати безліч перешкод, щоб досягти своєї мети.

«Dimension of Darkness» – гра у жанрі 2.5D shoot’em up з елементами Roguelike, розроблена для операційної системи Windows. Поки наявна лише англійська локалізація.

1.3 Галузь застосування

Тестування необхідне для досягнення визначених критеріїв якості, що визначені у тестовому плані. Основні мети тестування включають:

- функціональне тестуваня;
- підтвердження якості та надійності програмного забезпечення;
- покращення зручності використання графічного інтерфейсу користувача;
- оптимізацію програмного продукту.

1.4 Визначення проекту

Документ	Створено або доступно	Отримано або перевірено	Автор або ресурс	Примітки
Дизайн-документ гри	Так	Так	Попов Д.М. Хом’яков К.І. Луценко В.В.	

Рисунок Г.3 – Сторінка 3 плану тестування

2. ВИМОГИ ДО ТЕСТУВАННЯ

Функціональне тестування:

- Переміщення персонажа;
- Система освітлення;
- Бойова система;
- Система генерування карти;
- Поведінка супротивників.

Тестування інтерфейсу користувача:

- Перевірка часу відгуку на взаємодії користувача з інтерфейсом;
- Перевірка правильності відображення текстур на персонажах та локаціях.

Тестування встановлення:

- Перевірка наявності гри, що встановлюється.

Рисунок Г.4 – Сторінка 4 плану тестування

3. СТРАТЕГІЯ ТЕСТУВАННЯ

3.1 Типи тестування

3.1.1 Функціональне тестування

Мета випробування	Протестувати механіки гри
Технічний прийом	Протестувати функції: • Переміщення персонажа; • Система освітлення; • Бойова система; • Система генерування карти; • Поведінка супротивників.
Критерії завершення	Усі випробування були проведені

Функціональне тестування в геймдеві розглядає основний функціонал гри, щоб переконатися, що він працює правильно і відповідає вимогам. Це включає перевірку різних аспектів гри, таких як управління персонажем, взаємодія з об'єктами, виконання завдань та багато іншого. Крім того, функціональне тестування включає в себе перевірку колізій.

3.1.2 Тестування інтерфейсу користувача

Мета випробування	Переконатися, що інтерфейс користувача зручний, текстури та анімації правильно відображаються.
Технічний прийом	Перевірити правильність відображення текстур на персонажах, локаціях. Перевірити зручність користувацького інтерфейсу.
Критерії завершення	Усі випробування були проведені.

Тестування інтерфейсу користувача (UI) в геймдеві є важливою частиною процесу розробки, оскільки від правильної роботи інтерфейсу залежить зручність користування грою та користувацький досвід.

Рисунок Г.5 – Сторінка 5 плану тестування

3.1.3 Тестування встановлення гри

Мета випробування	Переконатися, що гра правильно встановлюється
Технічний прийом	Ручна або автоматична установка. Перевірка наявності гри, що встановлюється
Критерії завершення	Усі випробування були проведені

3.2 Інструменти

	Інструмент	Постачальник	Версія
Контроль версій	GitHub	GitHub, Inc.	2024
Управління проєктами	GitHub Projects	GitHub, Inc.	2024
Відслідковування дефектів	Jira	Atlassian	2024

Рисунок Г.6 – Сторінка 6 плану тестування

4. РЕСУРСИ

4.1 Ролі

Людські ресурси		
Працівник	Рекомендований мінімальний обсяг осіб	Конкретні обов'язки або коментарі
Тест-менеджер, Менеджер з тестування	1	Забезпечує управління наглядом. Обов'язки: - технічна підтримка, - придбання відповідних ресурсів, - забезпечення управлінської звітності
Конструктор тестів	1	Визначення, пріоритетів, і реалізація тестів. Обов'язки: - створення плану тестування, - генерація тестових моделей, - оцінка ефективності тестових зусиль
Тестувальник	1	Виконання тестів. Обов'язки: - виконання тестів, - журнал результатів, - відновлення в журналі реєстрації після помилок, - документ зміни.

4.2 Система

Системні ресурси	
Ресурс	Ім'я або тип
ПК для тестування	Персональний ПК
Репозиторій тестування Посилання	https://github.com/Dayman267/DimensionOfDarkness.git

Рисунок Г.7 – Сторінка 7 плану тестування

5. ЕТАПИ ПРОЄКТУ

Таблиця 5.1 – Етапи проєкту

Етап	Обсяг робіт	Дата початку	Дата закінчення
План випробувань	10	10.03.2024	11.03.2024
Тест - дизайн	15	12.03.2024	13.03.2024
Реалізація випробувань	28	14.03.2024	16.03.2024
Виконання тесту	35	18.03.2024	20.03.2024
Оцінка випробувань	7	22.03.2024	23.03.2024

Рисунок Г.8 – Сторінка 1 плану тестування

6. КІНЦЕВИЙ ПРОДУКТ

6.1 Тестова модель

У ході виконання тестування будуть отримані такі елементи: − план тестування – опис цілей та стратегій тестування, методів реалізації процесу тестування; − тест-кейси – документи, що відповідають примірникам тестового сценарію. Повинні містити: унікальний номер, опис, кроки відтворення, пріоритет, важливість тестового сценарію та очікуваний результат; − баг-репорти – звіти про знайдені недоліки в системі із зазначенням рівня серйозності проблеми.

6.2 Звіти з дефектів

Звіти з дефектів будуть створені з використанням текстового процесору MS Word, систем баг-трекінгу Jira, а також систем управління проектом GitHub Project.

Рисунок Г.9 – Сторінка 9 плану тестування

ДОДАТОК Г

Баг-репорти

Таблиця Д.1 – Помилка 1

Назва багу	Персонаж не розвертається повністю
Короткий опис	При швидкому натисканні кнопок руху персонажа починає переміщувати у необхідний бік, але повністю він не розвертається.
Компонент додатку	Player Controller(Player)
Важливість	S4 Низький
Пріоритет	P3 Низький
Кроки відтворення	1. Швидко натиснути на клавішу руху
Фактичний результат	Візуально дивно виглядає що персонаж зміщується у потрібному напрямку, але не розвертається туди.
Очікуваний результат	При швидкому натисканні персонаж повністю повернеться у необхідний бік.

Таблиця Д.2 – Помилка 2

Назва багу	Біг на одному місті
Короткий опис	Якщо натиснути протилежні клавіші руху персонаж буде бігти на місці.
Компонент додатку	Player Controller (Player, Animator Controller)
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	1. Натиснути клавышу бігу в один бік 2. Не зупиняючись натиснути клавішу руху у протилежний бік 3. Не відпускати клавіши руху
Фактичний результат	Програється анімація бігу при тому що персонаж стоїть на місці.
Очікуваний результат	Персонаж стоїть на місці та програється анімація спокійного стану.

Таблиця Д.3 – Помилка 3

Назва багу	Стрільба з місця
Короткий опис	Якщо почати стріляти з місця не прицілюючись то персонаж сам не увійде у режим прицілювання
Компонент додатку	Player Controller (Player)
Важливість	S1 Блокуюча
Пріоритет	P1 Високий
Кроки відтворення	1. Не натискати праву кнопку миші 2. Почати стріляти натиснувши ліву кнопку миші
Фактичний результат	Не можливо нормально стріляти бо не входить у режим прицілювання
Очікуваний результат	Персонаж при стрільбі, завжди входить у режим прицілювання.

Таблиця Д.4 – Помилка 4

Назва багу	Авторозворот ломає анімацію
Короткий опис	Під час прицілювання при натисканні різних клавіш руху персонаж хоче розвернутись через що його починає трохи трести
Компонент додатку	Player Controller (Player)
Важливість	S3 Середня
Пріоритет	P3 Низький
Кроки відтворення	1. Увійти у режим прицілювання 2. Нажати будь-яку клавішу руху
Фактичний результат	Персонажа трясе під час прицілювання
Очікуваний результат	Авторозворот не працює під час прицілювання та персонажа не трясе.

Таблиця Д.5 – Помилка 5

Назва багу	VFX пострілу зброї програється
Короткий опис	VFX стрільби програється весь час
Компонент додатку	GunSO (ShootingSystem)
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	1. Запустити гру з гвинтівкою
Фактичний результат	VFX програється весь час
Очікуваний результат	VFX програється тільки під час стрільби

Таблиця Д.6 – Помилка 6

Назва багу	Постріли повертаються до гравця
Короткий опис	Під час стрільби можна побачити як постріли повертаються до гравця, повертаючись до пула пострілів.
Компонент додатку	GunSO (ShootingSystem)
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	1. Почати стріляти 2. Змініти напрям стрільби
Фактичний результат	Видно як пулі телепортуються до гравця
Очікуваний результат	Постріли не видно коли вони повертаються

Таблиця Д.7 – Помилка 7

Назва багу	Зброя з’їжджає з руки
Короткий опис	При стрільбі з зброї з великою скорострільністю, зброя з'їжджає з руки з часом
Компонент додатку	GunSo (Shooting System)
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	1. Взяти зброї з великою скорострільністю 2. Почти стріляти
Фактичний результат	Зброя з’їжджає з руки
Очікуваний результат	Зброя залишається на місці

Таблиця Д.8 – Помилка 8

Назва багу	Сильне освітленя пострілу
Короткий опис	Під час пострілу занадто сильне освітлення
Компонент додатку	Light (Weapon, Flash, ParticalSystem)
Важливість	S3 Середня
Пріоритет	P3 Низький
Кроки відтворення	1. Постріляти
Фактичний результат	Занадто сильне освітлення
Очікуваний результат	Не сильне освітлення від пострілу

Таблиця Д.9 – Помилка 9

Назва багу	Сильний розкид пострілів
Короткий опис	Під час руху дуже сильний розкид пострілів, не можливо прицілитись
Компонент додатку	Player, GunSO (Shooting System)
Важливість	S1 Блокуюча
Пріоритет	P1 Високий
Автор	Луценко В.В.
Призначений на	Хом’яков К.І.
Кроки відтворення	1. Почати стріляти 2. Почати рухатись
Фактичний результат	Сильний розкид під час руху
Очікуваний результат	Розкид під час руху не змінюється
Прикріплений файл	1.9.mkv

Таблиця Д.10 – Помилка 10

Назва багу	Направлення стрільби не точно у місце прицілювання
Короткий опис	При стрільбі постріли летять не точно у направленні миші та не точно у ту точку де миша
Компонент додатку	GunSo (ShootingSystem)
Важливість	S1 Блокуюча
Пріоритет	P1 Високий
Кроки відтворення	1. Почати прицілювання 2. Почати стріляти
Фактичний результат	Постріли не точно у місце прицілювання
Очікуваний результат	Постріли точно у місце прицілювання

ДОДАТОК Д

Слайди презентації

Рисунок Д.1 – Слайд 1

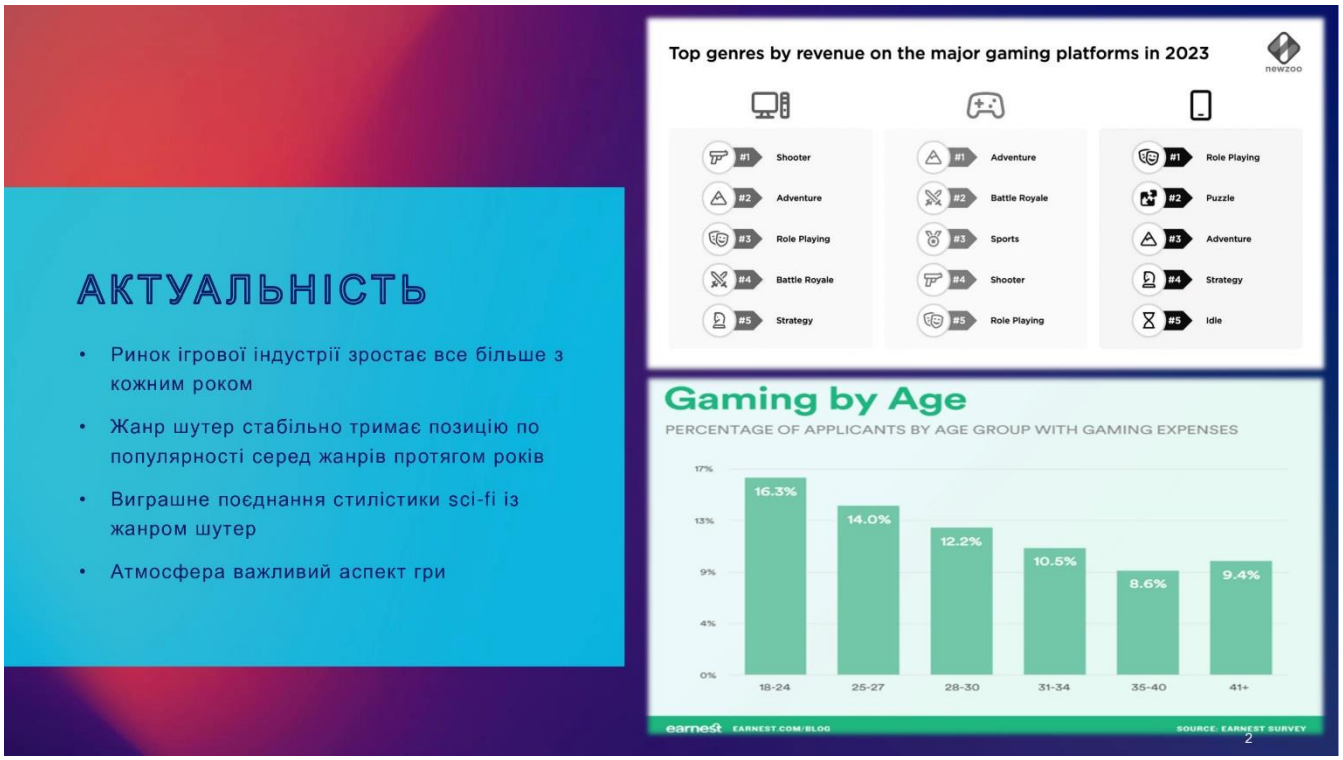


Рисунок Д.2 – Слайд 2

Рисунок Д.3 – Слайд 3

Рисунок Д.4 – Слайд 4

Рисунок Д.5 – Слайд 5

Рисунок Д.6 – Слайд 6

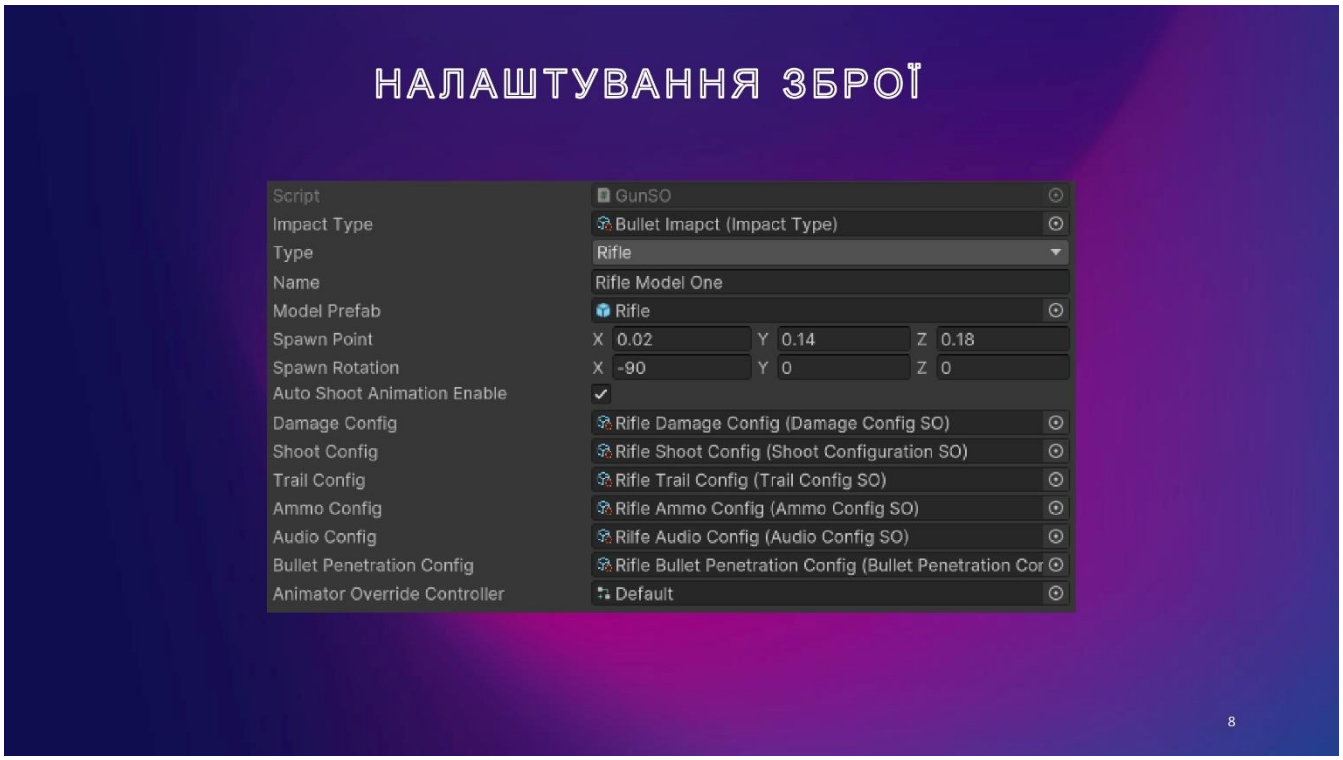


Рисунок Д7 – Слайд 7

Рисунок Д.8 – Слайд 8

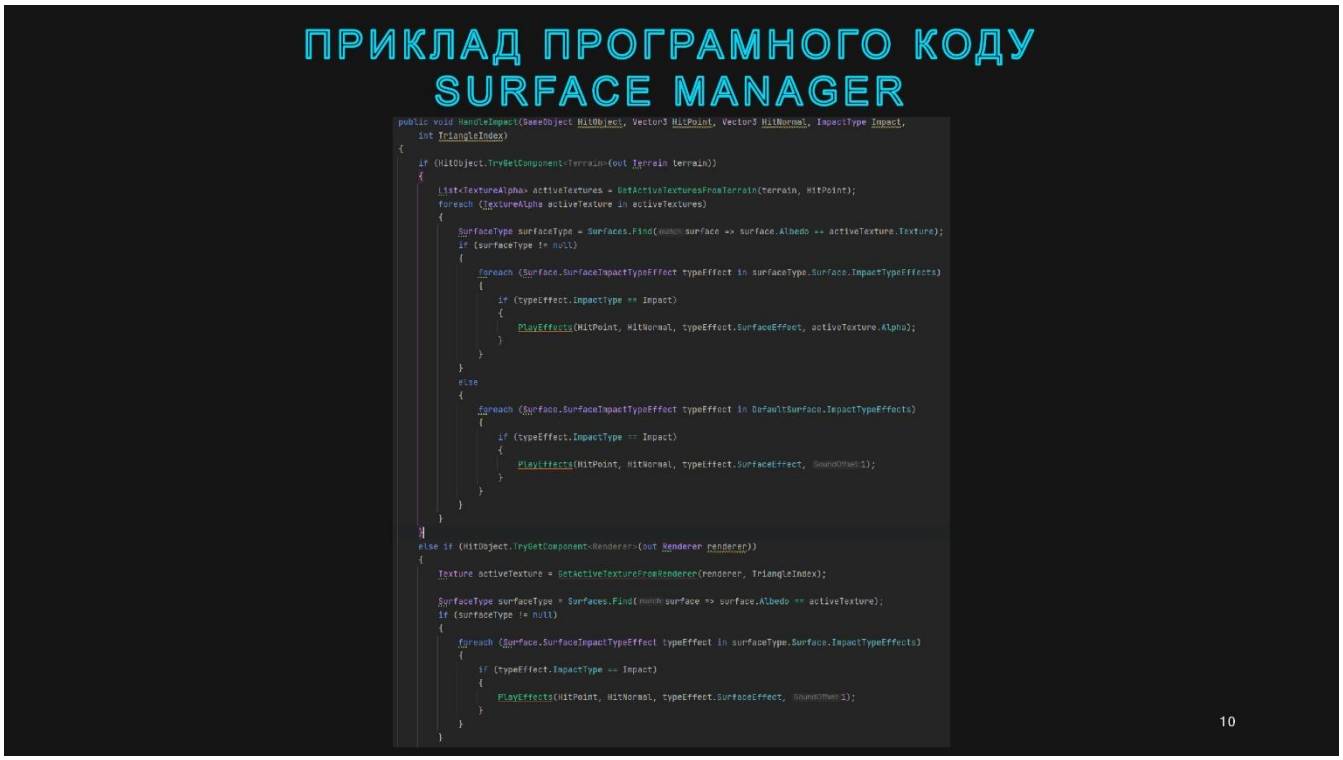


Рисунок Д.9 – Слайд 9

Рисунок Д.10 – Слайд 10

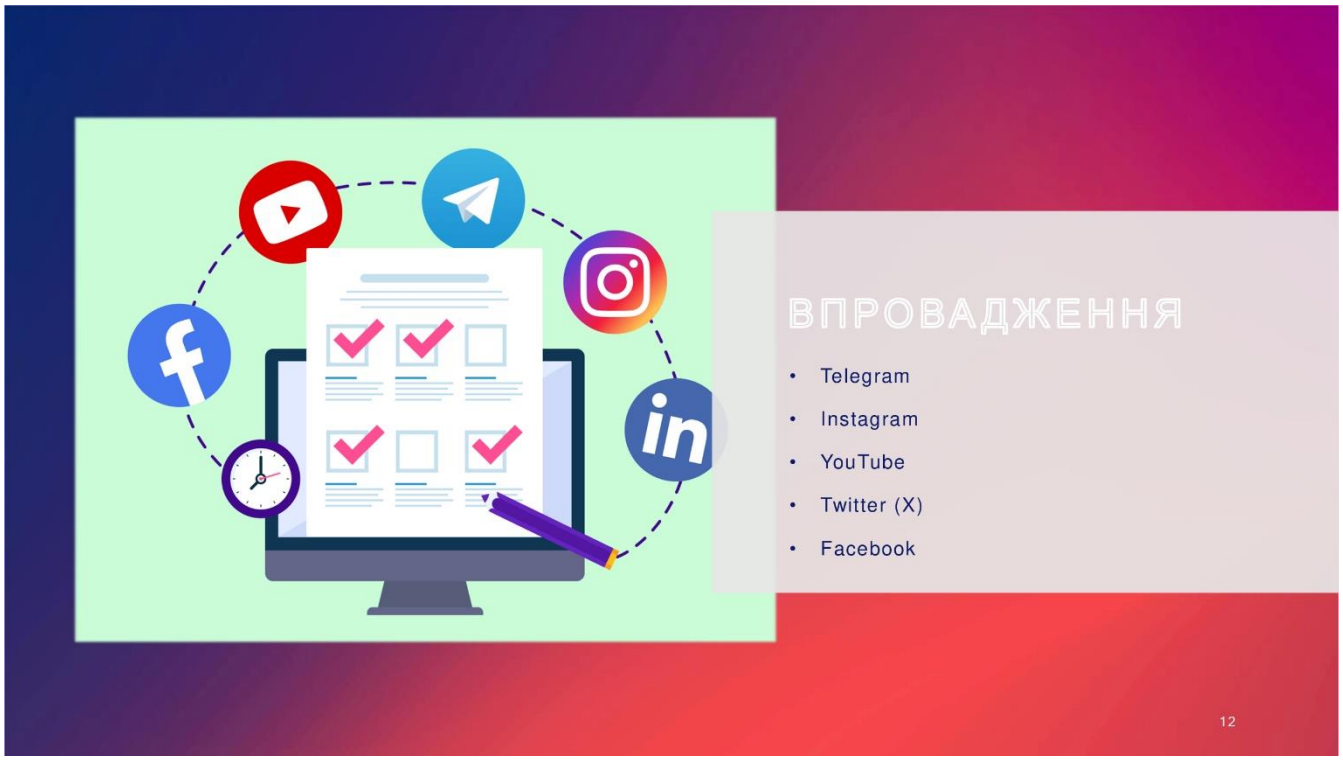


Рисунок Д.11 – Слайд 11

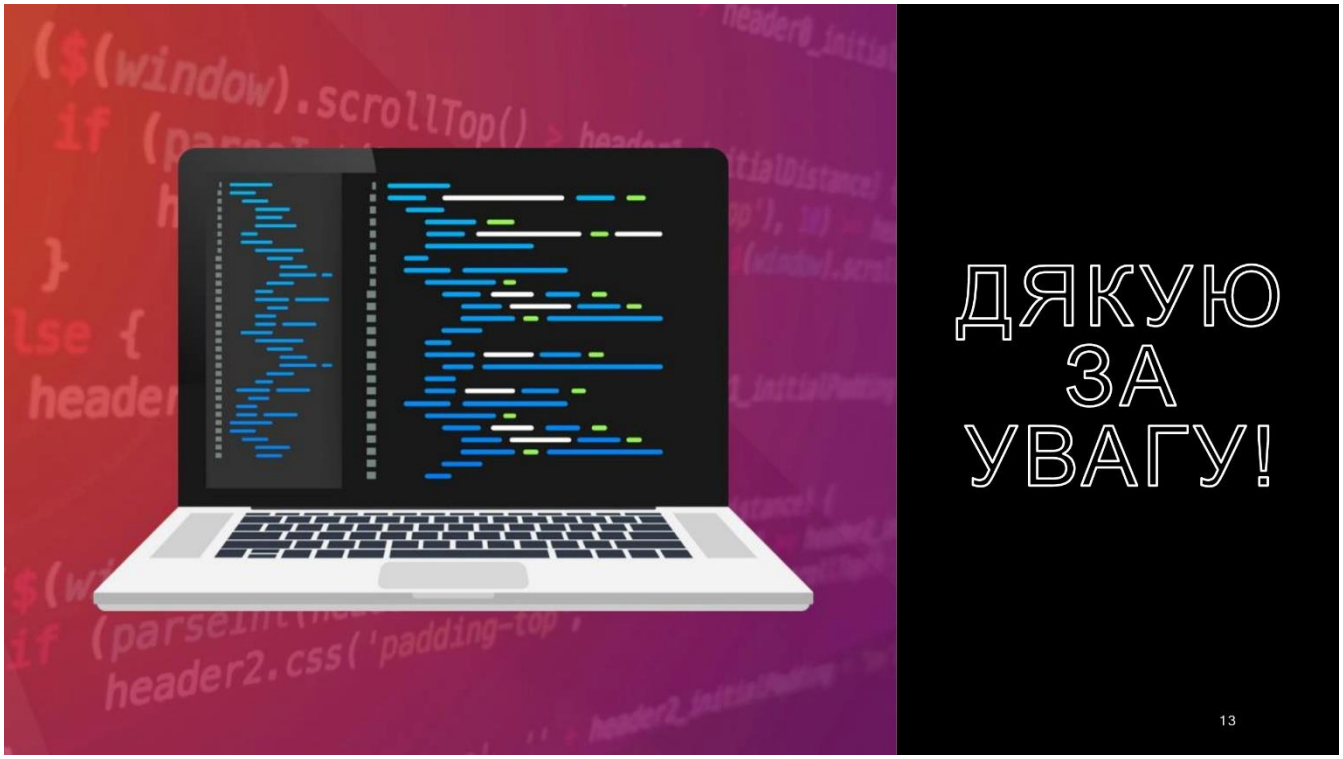


Рисунок Д.12 – Слайд 12