ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

“УНІВЕРСИТЕТ ЕКОНОМИКИ ТА ПРАВА ”КРОК”

КВАЛІФІКАЦІЙНА РОБОТА

Тема: “Комп'ютерна гра "Demonic Hunter" з використанням алгоритмів інтелектуальної поведінки ботів”

Ступінь вищої освіти - бакалавр

Спеціальність – 122 «Комп'ютерні науки»

Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав : здобувач 4 курсу

групи КН-21

Дмитро ВЕЛИЧКО

Керівник: зав. кафедри комп'ютерних наук,

к.е.н., с.н.с., доцент

Сергій МІЧКІВСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.

Здобувач: _____

(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп’ютерних наук
_Сергій МІЧКІВСЬКИЙ
«_____»____20___р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Величко Дмитро Михайлович

Тема роботи	Комп'ютерна гра "Demonic Hunter" з використанням алгоритмів інтелектуальної поведінки ботів (комплексна робота)
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Розробити програмне забезпечення 2D-гри на Unity з елементами платформінгу, з реалізацією інтелектуальної поведінки ботів.
Посилання на джерела інформації (не більше п’яти найменувань, які рекомендує науковий керівник)	1. Ваша перша 2D гра // Документація до Godot Engine 4.4 українською мовою – URL: https://docs.godotengine.org/uk/4.x/getting_started/first_2d_game/index.html , 2. Level-дизайн // Дизайн рівнів для 2D-ігор: основні патерни – URL: https://avada-media.ua/services/disain-urovney-dlya-2d-igr-osnovnye-patterny/ ,
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп’ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 24 грудня 2024 р.

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступеня бакалавра Дмитро ВЕЛИЧКО

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
Основний етап			
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
Завершальний етап			
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Сергій МІЧКІВСЬКИЙ

Здобувач освітнього ступеня бакалавра Дмитро ВЕЛИЧКО

Величко Д.М. Комп'ютерна гра "Demonic Hunter" з використанням алгоритмів інтелектуальної поведінки ботів (комплексна робота)

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп’ютерні науки (освітня програма – Комп’ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп’ютерних наук, Київ, 2025.

Розроблено гру в 2D форматі та піксель-арт, пост-апокаліптичному сетингу, з акцентом на ігрові механіки, атмосферу та унікальність стилю, з використанням алгоритмів інтелектуальної поведінки ботів та інструментів розробки Unity, Aseprite.

Ключові слова: ігри, 2D, піксель-арт, пост-апокалипсис, шутер.

Рис. 26. Бібліограф.: 17 найм.

Velichko D.M. Computer game "Demonic Hunter" using algorithms of intelligent behavior of bots (complex work)

Explanatory note of the qualification work in the specialty 122 – Computer Science (educational program – Computer Science) Bachelor’s degree. – Higher Educational Institution “University of Economics and Law “KROK”, Educational and Scientific Institute of Information and Communication Technologies, Department of Computer Science, Kyiv, 2025.

The game was developed in 2D format and pixel art, in a post-apocalyptic setting, with an emphasis on game mechanics, atmosphere, and unique style, using algorithms for intelligent bot behavior and Unity and Aseprite development tools.

Keywords: games, 2D, pixel art, post-apocalypse, shooter.

Fig. 26. Bibliography: 17 Items.

ЗМІСТ

ВСТУП

Актуальність теми. Якщо провести аналіз з 2010 року до сьогодні, можна побачити, що інтерес гравців до ігор у цьому жанрі залишається стабільним. Піксельні ігри від інді-розробників неодноразово ставали справжнім вибухом в інформаційному просторі. Проте, щоб зацікавити користувача, необхідно створити або захопливий сюжет та унікальну атмосферу, що дозволить повністю зануритися у світ гри, або ж ретельно опрацьовану піксельну графіку, яка передає особливий художній стиль. Багато гравців, які полюбляють такий жанр і стиль, будуть раді побачити щось нове.

Проблемна ситуація. Основні труднощі виникають ще на початковому етапі створення таких проєктів. Створити унікальний ігровий процес та піксельну графіку, яка зможе передати настрій і атмосферу, є складним завданням. Важливо наситити намальований світ відчуттям загрози у жорстокому пост-апокаліптичному середовищі, створюючи реальну небезпеку для ігрового персонажа. Для цього необхідно розробити унікальні механіки та нововведення, які зроблять гру цікавою та самобутньою. Також важливо знайти баланс між сучасним стилем піксель-арту та класичною естетикою старих консолей. Окрему складність становить створення глибокого та захопливого сюжету, який гармонійно впишеться в загальну атмосферу гри. В іграх цього стилю важливо розробити унікальні або нові ігрові механіки, які зможуть повністю занурити користувача в ігровий процес. Інді-розробники мають можливість створити як простий, так і більш інформативний та продуманий інтерфейс, залежно від концепції гри. Бойові механіки також повинні бути логічними та добре опрацьованими. Система бою має бути інтуїтивно зрозумілою, як для досвідчених гравців, так і для новачків, які лише знайомляться з таким жанром.

Мета дослідження: створити гру у 2D форматі та піксель-арт, пост-апокаліптичному сетингу, з акцентом на ігрові механіки, атмосферу та унікальність стилю, з використанням алгоритмів інтелектуальної поведінки ботів.

Завдання на дослідження полягають в наступному;

проаналізувати існуючі аналоги гри;

проаналізувати алгоритми інтелектуальної поведінки ботів;

спроєктувати елементи та архітектуру гри;

створити оригінальний інтерфейс гри;

розробити програмне забезпечення.

Об'єктом дослідження є ігровий процес платформера у пост-апокаліптичному сетингу.

Предметом дослідження є ігрова індустрія 2D піксель-арт платформерів з акцентом на ігрові механіки та алгоритми інтелектуальної поведінки ботів.

Методи дослідження це аналіз ринку ігор в такому жанрі, виокремлення декількох вже вдалих проектів, спроба відтворити свій візуальний стиль чи геймплейну частину. Аналіз алгоритмів інтелектуальної поведінки ботів.

Практичне завдання. Розроблена 2D гра на ігровому рушії Unity. В якій буде свій візуальний стиль також геймплей і впровадження інтелектуальної поведінки ботів.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (17 найменувань). Пояснювальна записка містить 26 рисунків. Загальний обсяг пояснювальної записки складає 56 сторінки, основний зміст викладено на 54 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ З РОЗРОБКИ 2D-ГРИ "DEMONIC HUNTER"

1.1 Предметна область аналіз аналогів

Що таке комп’ютерні ігри - взаємодія людини (групи людей) з комп’ютером або декількох людей між собою за допомогою комп’ютера для розваг, навчання чи тренування. Під час створюється імітація прямої взаємодії у віртуальному просторі між персонажами та користувачем.

Ігри можна поділити за жанрами, такими як шутери, стратегії, RPG та симулятори та інші. Вони також відрізняються за стилем графіки: піксельна, реалістична, казуальна і мінімалістична тощо. Крім того, ігри можуть бути на одного чи декількох гравців.

Однією з головних проблем 2D ігор є або сюжетна сторона або нецікава геймплейна частина.

Основні проблеми 2D гри:

- якщо говорячи через призму 2D полотна то в такому разі у нас є лише обмеженість у рухах персонажів, противників та і в інших аспектах. Можливість реалізовувати щось важке не вийде;
- деякі ігри мають обмеження в механіках тому вони можуть швидко стати не цікавими гравцю;
- також поширена проблема в дизайні, а точніше у візуальному стилі. Гра може виглядати в мінімалістичному стилі, а команда із ну кажучи десь 5 чоловік може бачити проєкт по різному і в кінцевому продукті можуть деякі спрайти не підходити одне до одного. Тому потрібне бачення візуальної концепції.

З самого початку, коли починаєш робити якийсь масштабний чи не дуже проект потрібно розробити стартовий макет точніше – замальовки чи ескізи майбутньої гри.

Це можна зробити декількома способами або використати звичайний це ручку або олівець та звичайний блокнот чи бумагу і зробити пробний

стартовий макет на якому буде розписаний приблизний сюжет можливо комікси чи вставки, які потім використаються у движку. Або це все зобразити в графічних редакторах і тому подібних.

Намалювати локації на яких буде відтворюватись геймплейна частина і розпис механік, які допоможуть сильніше понуритись в цей світ. Не варто забувати ще про противників загрози з якими зіткнеться користувач.

Власний інтерфейс, добавити до цього контури та обводку, щоб це все виглядало більш об’ємнішим та структурованим для погляду і реалізувати все таким чином під концепцію майбутньої розробки. Як я це все бачив на статях і різних форумах можна взяти в приклад одну давнішню студію розробки під назвою ID Software.

Взявши в приклад цю команду розробки, якій на той час працювали лише два розробника ігор, їм потрібно була ідея для гри. До цього в команда розробників в створювали шутер в 3D – просторі, який мав назву Wolfenstein 3D. В цій гри відомий програміст Джон Кармак створив революцію в жанрі таких ігор, що дам можливість переміщювати камеру головного героя аби передати ефект шагу хоча на самому ділі камера була статична.

Це вдалося досягти за допомогою модифікованого коду гри. Виділивши деякий час на створення наступної серії ігор Команда змогла розширитися після успіху Wolfenstein 3D. Черпавши натхнення під час перегляду одного фільму в якому грав відомий актор Том Круз він промовив одне слово, яке сильно запало в думки одного з розробників. Це слово мало назву з чотирьох букв – DOOM.

Побалакавши з іншими колегами по цеху вони взяли цю абревіатуру, як назву для наступної гри. Сюжет на той час це приблизно 1993 роки не мав важливу роль, як говорив Джон Кармак. То сюжет міг поміститися на одній сторінці разом з випущеною грою і був дуже коротким.

Дії відбували з розповіді самих розробників в нашому часі, де на світі з людьми та демонами раптово відкрився портал і почалось захоплення світу монстрами. Головний герой був звичайним солдатом, який виконував свої

обов’язки та винищував противників. Як кажучи сюжет і помістився навіть в два рядочки.

На той час коли гра встигла вийти в 1993 році, комп’ютерні обчислення були на досить обмеженому рівні то для команди розробки стала конкретна задача, як утворити графічну частину та не сильно навантажувати системи гравців в той час. Всі асети та спрайти малювати тоді в ручну. В звичайному магазині з дитячими іграшки вдалось знайти пластмасову зрою, яку в подальшому перемалювати для використання в грі. Тому тепер можна дізнатись, що дробовик був змальований з справжньої іграшки. Також ще виділивши графічну частину можна підмітити, що всі текстури були по різному намальовані в кожному джерелу світла. Взявши той самий пластмасову рушницю, то він був намальований у різних світових відтінках. В темному просторі він мав сіріший колір та спалах від пострілу був темнішим і таку аналогію можна привести до кожного елемента зброї.

Період розробки був дуже проблемний бо потрібно для цього всього ще розробити власний двигун і впоратись в обмежений строк. Грошей в компанії було не досить багато бо через контракти по випуску першої гри були підписані з іншою компанією, яка брала гру Wolfenstein і портували її на свої диски, а дохід був розписаний по документам таким чином - 50 відсотків одній корпорації, яка робили ці диски і 50 відсотків самим розробникам.

Для Джона Кармака, як основного програміста випала важка участь у розробці свого Двигуна і оптимізації його під системи колишніх геймерів. Ставалось таке, що він залишався в офісі працювати днями та ночами, добавляти постійно нові введення та оптимізацію аби все коректно працювало. Його співробітники, які працювали разом могли через нестачу відпочинку засинати на робочих місцях. Тому коли в них через труднощі вдалось написати першу версію для компіляції проекту почали наступні кранчі назвавши їх так по розробці проєкту. Труднощі труднощами але все ж таки не дивлячись на безсонні ночі команді все таки вдалось розробити гру, яка розійшлась великим

тиражом і після невеликого відпочинку розробники почали робити сіквел наступної гри.

Якщо передивлятись всі можливі стилі малювання то в основному можна виділити які я зміг дізнатися це 4 методи:

- казуальний стиль;
- реалістичний стиль;
- мінімалістичний стиль;
- ретро стиль;

Почнемо розбирати все по пунктам.

Візьмемо казуальний стиль дизайну. Що можна про нього розповісти. Найчастіше його можна зустріти в більшості платформерах і іграх реалізованих в Unity для більш насиченого дизайну в картинці.

Говорячи про контури в спрайтах то більшу увагу художники вони приділяють чітким контурам і добавлення об'єму, щоб це все виглядали більш насичено.

Беручи до уваги наступний стиль дизайну під назвою Реалізм. Тут з самого початку вже з самої назви можна уже виконати свої думки. Цей стиль більш підходить до таких ігор в яких велика увага йде на сюжетну сторону та візуальну схожість з реальним світом. А точніше всі піксель будуть відтворюватись по схожим предметам з реального світу і мати об’єм в кожному з спрайтів. Картинка при цьому буде виглядати в 2D – просторі але візуал буде походити на реальність.

Розповідаючи про Мінімалістичний стиль дизайну можна розказати такі терміни. Тут інді-розробники можуть виконувати всі замальовки в трьох чи чотирьох відтінках. Або з мого досвіду знаю одну інді - гру в якій художники змогли вкласти навіть в два кольори. Це може також дати своє унікальне бачення проєкту і відтворити свій стиль. В цілому такі ігри які створені з обмеженою світловою палітрою користуються невеликою популярність але кожна гра може дати свій унікальний досвід.

Ретро – стиль дизайну. В самої назви беремо таку аналогію, як і з другого пункту. Цей вид графіки переносити гравця в часи тих самих 90 років

коли буле ще перші приставки, квадратні монітори та інші системи. Коли на ті роки можна представити ігри з ти самих консолей. Кожна мала свою індивідуальність і цікавий геймплейний план. Можна також розказати про те що консолі були слабкими на ті часи і приходилось знаходити нові рішення для універсально стиля. Використовувались кольори, які були для кожної консолі різні і графіка малювалась також відповідно тих відтінків аби передати об'ємність всіх спрайтів. Використовували нестандартні кольори такі як червоний, синій та інші. Це добавляло ізюминку до кожного намальованого предмету в грі.

Можна виділити жанрів 5 до цієї теми:

Стратегії;

Симулятори;

Гонки;

Пригоди;

Спорт.

Жанр Стратегії був розроблений для аудиторії, яка полюбляє робити власні відбудовані держави чи війні в реальному часі проти гравців або комп'ютера. В цілому основна задача кожного жанру як і інших - це дати користувачу прогресію і економічний базис. З таких відомих стратегій, які я міг виокремити Frostpunk, Sid Meier's Civilization і Citi Skylines. Кожна гра по своєму унікальна.

Говорячи по Frostpunk – це стратегія, яка перевіряє гравця на керування поселенням під час холодної зими де ресурсом являються навіть люди. Потрібно досягти гарної прогресії вашого поселення, автоматизувати добуванні різних ресурсів (Дерево, Вугілля, Сталь, їжа та інше.) Основна ціль гри постаратись не померти від останньої бурі в кінці основного сюжету. І щоб вашу роль капітана не забрав хтось інший. Точніше не буду поверженим.

Sir Meier's Civilizations – це стратегія, в реальному часі де нам потрібно взяти роль одно з десятка наведених керувачів тих часів та розвити своє поселення від кам'яного віку до майбутньої прогресуючої держави. В грі

можуть бути реальні гравці чи комп’ютери, можна як домовитись з ним та вкласти мир або розв’язати війну на свою користь але треба враховувати ризики та не програти в цій світовій гонці. Є різні епохи і вам потрібно в обмежений час встигнути розвинуться на скільки це може буть можливо та переходити у новий вік для наступних досліджень.

City Skylines – це стратегія, де вам потрібно відбудувати свій город від маленького поселення до величезного мегаполісу. В грі можна Основна ціль це відбудувати місто, щоб була гарна інфраструктура, підтримка економіки та виконання послуг жителів вашого поселення. Це досить популярний проект серед гравців бо в ньому захоплюючий ігровий процес та гарна підтримка різними модифікаціями від користувачів.

Симулятори - це такі досить реалістичні ігри, які підходять для обмеженої аудиторії. ігровий процес в таких дуже специфічний. Він більше приближений до реальних професій. З таких знайоми проєктів можу виділити – Euro Truck Simulator, Ready or Not, Dirt Rally. Можна поставити себе на місце одно з професіоналів свого діла та спробувати різні напрямки в грі.

Euro Truck Simulator – це гра в якій ти вливаєшся в сферу вантажоперевезень, набираєшся навичок на орендних фурах. Після здобутих навичок ти розумієш, що умови праці тебе не задовольняють і хочеш зробити свою компанію але з перекривання всіх недоліків попередніх фірм. Дається змога купівля одного з багатьох гаражів на вибір в різних регіонах Європи та вдосконалюєш свій бізнес, щоб забезпечити гідні умови працівникам та стати гарантом якості в сфері перевезень.

Dirt Rally – це класичний симулятор для любителів спорту по ралі. Вам дається можливість створити ігрового персонажа, яким доведеться керувати різними автомобілями на вибір. Прогресія в проєктів зображену у вигляді невеличкої кар’єри. Починаючи з стартових автомобілів на вибір потрібно досягти перші перемоги в турнірах і отримати гроші. Так поки не вийде відкрити сучасні авто і нові траси для змагань. Є схожі проєкти але цей на мій досвід гарний конкурент серед більшості подібних ігор. Простий і зручний

інтерфейс, добре виконана симуляція, а саме поїздки, як на самих звичайних авто та до самих відомих машин які брали участь у реальних гонках займаючись за лідерські місця. Цікавий проект, якій знайде свого користувача.

Ready or Not – тактичний симулятор де вам, як в ролі звичайного поліцейського потрібно виконувати місії від уряду по затримці противників. Точніше кажучи можуть бути, як звичайні бандити так і кримінальні авторитети, які торгують зброєю чи чимсь гірше. Потрібно звичайно вибрати для себе або своєї команди комп'ютерних чи реальний гравців знаряджання. Це може бути зброя, автоматні набої, тактичне предмет, елементи броні та інше. Після основної підготовки вибираємо сценарій для завдання і головною ціллю для гри стоїть не вбивство, а затримання. Зброю потрібно використовувати лише в крайніх ситуація. Досить специфічний проект, який зможу вподобати обмеженій авдиторії.

Гоночні ігри – цей жанр дозволить гравцю брати участь у змаганнях на автомобілях, мотоциклах або інших транспортних засобах. Виділю таких проєкти - *Need for Speed, Forza Horizon, The Crew*. Вони можуть бути як аркадними, так і реалістичними симуляторами. Масовий бум жанру припав на 2000 роки, коли серія ігор *Need for Speed* робила геймплейну частину та смішно кажучи сюжетну привабливою та захоплюючою.

Пригоди - орієнтовані на сюжет, дослідження світу та взаємодію з персонажами. Можна виокремити - *Tomb Raider, Kingdom Come Deliverance, Fallout*. Вони можуть мати відкритий світ і елементи інших жанрів. Частіше всього для гравця передають цікавий сюжет та ігрові вставки аби ще сильніше проникнутись до таких ігор.

Жанр спортивних проєктів, реалізовує відчуття змагання та суперництва між гравцями. До цього всього можна віднести – FIFA будь якого року, серію ігор NBA про баскетбол та тому подібних. Такі ігри орієнтовані на мою думку для людей, яким потрібені змагання та перемоги у віртуальному просторі. Розробники цих проєктів можуть підійти з усією серйозністю. Добавивши в них реалістичність та простір для створення свої ігрових ситуацій

1.2 Аналіз аналогів

Розглянемо ряд аналогів реалізації ігор 2D форматі та піксель-арт, пост-апокаліптичному сетингу.

На рис 1.1 зображено геймплейний відрізок з гри The Underground Man 2. Якщо порівнювати відгуки з різних сайтів і тому подібних, брати до уваги форуми та інше. Зробивши невелику статистику серед всіх відгуків то можна побачити що хтось може написати про проблему в візуальному стилі, хтось може розказати, як йому не сподобався сюжет і так до безкінечності але на мою думку з побаченого геймпленого ролику я виявив мабуть дизайнерську неточність. Точніше деякі локації можуть не підходити під стиль гри, а так, якщо дивитись на інтерфейс та геймплей він досить привабливий та захоплюючий.

Рисунок 1.1 – Геймплей гри The Underground Man 2

Джерело [10]

Дивлячись на рис 1.2 з гри Scatteria. Пост апокаліптичний шутер в зроблений в піксель-арті. Роблячи невеликий підрахунок середнього рейтингу по платформі Steam можна побачити, що в більшій половини хто зміг пограти в цей проєкт залишив гарний відгук, тому з побачених геймплейних скріншотів та роликів можу виділити захоплюючий дизайн рівнів і інтересні механіки бою. Якщо вдатись в мінуси кажучи не сподобалось лише далека камер від гравців, мабуть це зроблено спеціально в шутер стиль, щоб ухватити великий масштаб карти та бачити великий обсяг всіх спрайтів і чіткіше орієнтуватися на місцевості.

Рисунок 1.2 – Геймплей гри Scatteria

Джерело [9]

На рис. 1.3 був створений з відрізка 2D – гри під назвою 2DMBIE. Розробкою займався один інді-розробник, який виклав цей проект на ресурс де кожен може поділитись своїм першими чи вже готовими проектами. По стилю - це мальовка в піксель – арті. Жанр тут пост-апокаліпсис і розроблена у 2D.

Відзначаючи цю гру можна в декількох плюсах та мінусах, які можна передивитися на сторінці сайта з вже записаного геймплейного ролику. Тут досить цікавий геймплей, можна брати участь одному або з друзями відбиватися від хвиль зомбі і намагатись не померти. В ньому дуже легко розібратись та відразу почати грать. Цікавий та свій дизайн гри в піксельному стилі. Використаний стиль – ретро, який дає цікавий досвід від проведеного часу.

Рисунок 1.3 – Геймплей гри 2DMBIE

Джерело [8]

Проходячись по мінусам гри можна виділити. Наповнення невелике, тому за декілька проведених сесій може почати набридати однотипність ворогів та рівнів. Виділити з мінусів можна наповненість різними механіками чи геймплейними нововведеннями.

1.3 Постановка задачі

Проєкт являє собою 2D гру у піксельному стилі, яка розгортається у пост-апокаліптичному світі після глобального спалаху вірусу.

Основна мета гри полягає у тому, щоб гравець, виступаючи в ролі колишнього військового під іменем Джиммі, найнятого американським урядом, зібрав три склянки з частинками вакцини та створив повноцінний препарат для боротьби з вірусом.

Це стане головною сюжетною лінією гри, що мотивуватиме гравця досліджувати Рівні та знаходити секрети, боротися із загрозами і вижити.

Назва в гри буде – Demonic Hunter.

Проєкт дає можливість побувати у похмурому, безнадійному світі, сповненому небезпекою. Локації недалекого майбутнього будуть наповнені залишками цивілізації, яка дізнавалась щось нове все зруйнувала своїми експериментами,

Ворогами головного героя стануть зомбі, мутанти та тому подібні, що утворилися під час спалаху віруса.

Основна мета в грі це збір частин вакцини –герой має знайти три склянки, які містять різні компоненти вакцини. Ці частинки знаходитимуться в різних локаціях і для їх пошуку треба буде гарно все дослідити.

Атмосфера – гра реалізовуватиме стильну піксельну графіку, яка передаватиме безнадійність і темряву пост-апокаліптичного світу.

Дослідження та виживання – гравець зможе відвідувати різні локації, такі як Покинуте місто, Втрачений ліс та Каналізацію, що стали смертельно небезпечними.

Має бути реалізована бойова система – гравець може використовувати зброю, як ближнього бою так і дальнього. Але треба слідкувати за своїми запасами бо вони рано чи пізно скінчяться. Тому їх треба знаходити або на рівні або з вбитих ворогів.

Гра буде поєднувати елементи шутеру, виживання та дослідження, а її основна мета – передати напружену атмосферу пост – апокаліпсису і передати

гравцю весь страх смерті та втрати всіх накопичених ресурсів. В світі що був зруйнований людством наш головний герой дасть нове життя якщо зможе впоратись з усіми недугами, які випадуть на його плечі.

Особливістю гри є сіруваті кольори, що передає самотність і депресію в світі. Піксельне малювання, що передає похмурість, гнусність та безнадійність пост-апокаліптичного світу. Гра матиме депресивний, тяжкий і мертвий стиль в мальовці, щоб передати всю важкість атмосфери.

Як це все буде виглядати:

Сірувата палітра – будуть використовуватись тони з малою насиченість ну точніше темнішою рисовкою.

Виражені об’єкти – Намальовані доми та лісосмуги і інші об’єкти будуть на задньому фоні сприяють на атмосферу, що не вийшло пережити цьому світу. Таким чином можна підкреслити сірість намальованого світу.

Наповнення і самі моделі буду виглядати тусклими та пригніченими.

Головна мета такого стиля малювання передати гравцю, емоцію переживання, жорстокості та самотності в цьому світі.

Серед невеликої кількості переглянутих робіт, які виконували інді-розробники, я дізнався, що конкуренція в такому стилі та жанрі досить невелика. Багато проєктів зроблені на гарному рівні, але їм чогось не вистачає. Тому розробка такого проєкту є доцільною, щоб, можливо, викликати інтерес у майбутньої аудиторії, яка захоче спробувати пограти в гру в сетингу пост - апокаліпсису.

Гра буде створена на двигуні Unity. Дії відбуватимуться у пост-апокаліптичному сетингу, де стався масштабний спалах вірусу. Гра буде у 2D, тому реалізація деяких механік може мати певні обмеження.

Ми граємо за військового у відставці на ім'я Джиммі, якого завербувало американське уряд як досвідченого найманця для допомоги у вирішенні проблеми з вірусом. Його головне завдання – знайти три фрагменти вакцини, щоб у майбутньому дати людям надію на порятунок.

Оскільки гра відбуватиметься у 2D, необхідно якісно намалювати всі спрайти, щоб передати атмосферу пригніченості та похмурості. Це дозволить користувачам отримати унікальний досвід, який можна порівняти з враженнями від великих ігор, розроблених великими компаніями.

У межах проєкту повинно бути реалізовано:

програмна частина - код має бути добре написаним, зрозумілим і оптимізованим;

інтерфейс - розробити простий та інтуїтивно зрозумілий ui/ux;

бойова система - передбачити як дальній, так і ближній бій;

інтелект ворогів - надати противникам алгоритми поведінки;

графіка - створити тьмяний, але виразний піксель-арт стиль.

Висновок до розділу 1

Аналіз різних жанрів ігор дозволяє краще зрозуміти сильні сторони кожного з них та їхні потенційні недоліки. Кожен жанр має свою унікальну атмосферу, механіки та підхід до взаємодії з гравцем. Особливої уваги заслуговують інді-ігри, які часто вражають оригінальністю, креативністю та глибоким змістом попри обмежений бюджет. Завдяки таким розробникам геймінг розвивається не тільки технічно, а й ідейно. Порівняння жанрів та прикладів ігор показало, що інновації та емоційна глибина можуть йти поруч із простотою, особливо у світі інді-розробки.

РОЗДІЛ 2
ПРОЄКТУВАННЯ

2.1 Діаграма варіантів використання Use Case Diagram

На рис. 2.1 представлено Use Case Diagram.

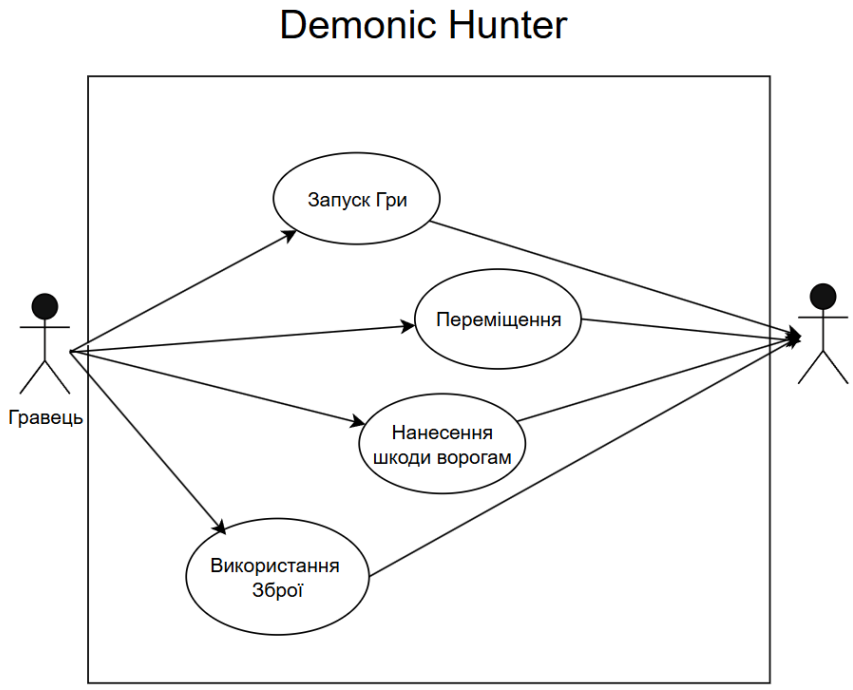

Рисунок 2.1 – Діаграма Use Case

Джерело: розроблене автором

Позначення до таблиці:

- Гравець – позначається як основний об’єкт;
- Коло буде позначати функцію, яку може виконати користувач;
- Стрілочка до функції позначає запит, який робить користувач;
- Стрілочка, яка йде від функції, показує гравцю результат в проєкті.

У нас, як у користувача, при запуску гри є певні алгоритми. Починаючи з головного меню, користувач може налаштовувати параметри та заходити у представлені інтерфейси. Після натискання кнопки *Play* завантажується рівень.

Коли користувач переміщується, подаючи команду грі, вона показує йому результат. Так само можна пояснити і дію зі зброєю, натискаємо клавішу відбувається обробка запиту, після чого на екрані бачимо постріл.

2.2 Діаграма діяльності Activiti Diagram

На рис.2.2 діаграму активності (Activity Diagram).

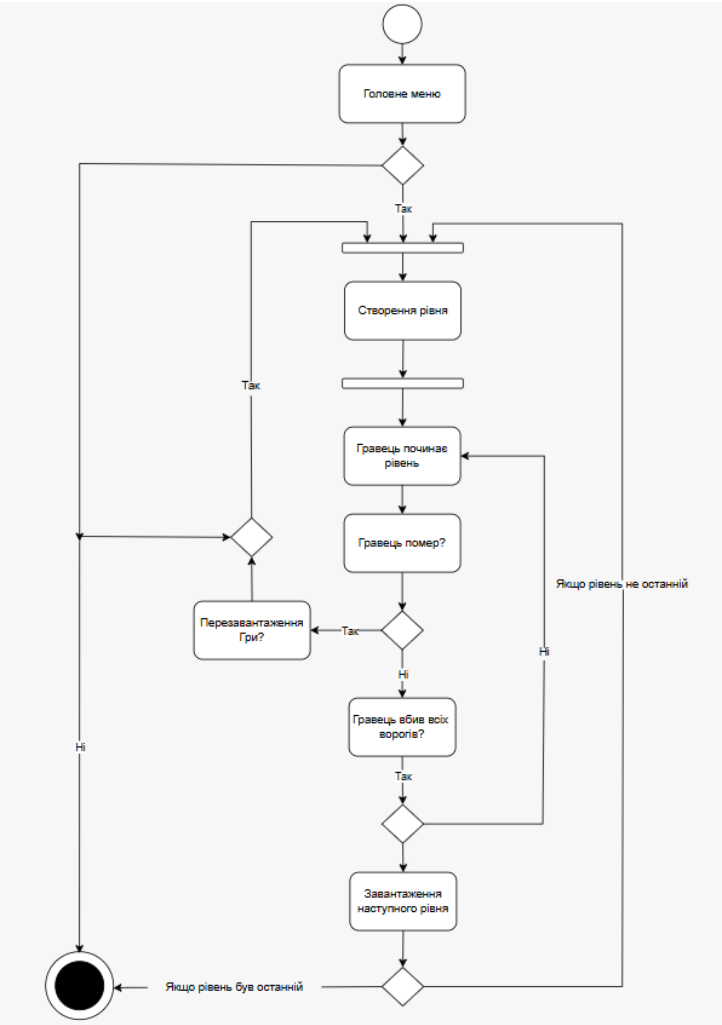

Рисунок 2.2 – Activity Diagram

Джерело: розроблене автором

Позначення на рис. 2.2:

- Чорний круг означає гравця і його початок дій;
- Ромб позначає в діаграмі умову для перевірки на два стани;

- Стрілочки вказують на послідовність операцій за діями гравця;

- Прямокутник вказує на вже вписану дію для її гравець вибере активність;

- Білий круг в рамці позначає кінець всіх можливих дій та алгоритмів.

Пояснення до цієї діграми під рис 2.2. Дивлячись на цю діаграму користувач бачити перед собою стартове меню в якому може вибрати чи почати йому грати чи вийти на робочий стіл через кнопки.

На діаграмі, користувач бачить перед собою стартове меню, в якому можна обрати “Почати гру або Вийти на робочий стіл” за допомогою відповідних кнопок.

Наступним етапом є вибір кнопки Play. Після цього гравця завантажує на заздалегідь підготовлений рівень із ворогами та іншими труднощами.

Після початку гри для гравця активуються два можливих алгоритми:

- якщо гравець помирає (не впорався із викликами), завантажується меню, де йому потрібно натиснути кнопку, щоб розпочати рівень спочатку;
- якщо користувач зміг подолати ворогів, на локації з’являється точка виходу, при дотику до неї персонаж переходить на наступний рівень.

Далі також можливі два варіанти розвитку подій:

- якщо рівень не останній запускається алгоритм створення нового рівня;
- якщо рівень був останнім гра завершується.

2.3 Діаграма класів Class Diagram

На рис. 2.3 представлено діаграму класів. В коді Hero до якого підв’язані ще два дочірніх класи такій принцип роботи. Anim відповідає за анімації персонажа в залежності від ситуації. Змінна JumpForce відповідає за силу стрибка, RidgeBody 2D дозволяє гравцю переміщюватись. Йде наступна прив’язка методів, які відповідають за анімації число сердець та завантаженні

відповідних меню коли серця закінчяться, все прописано у PlayerHealth. Gun Створює префаб, заготовлений об’єкт на сцені і точку через яку будуть вилітати кулі.

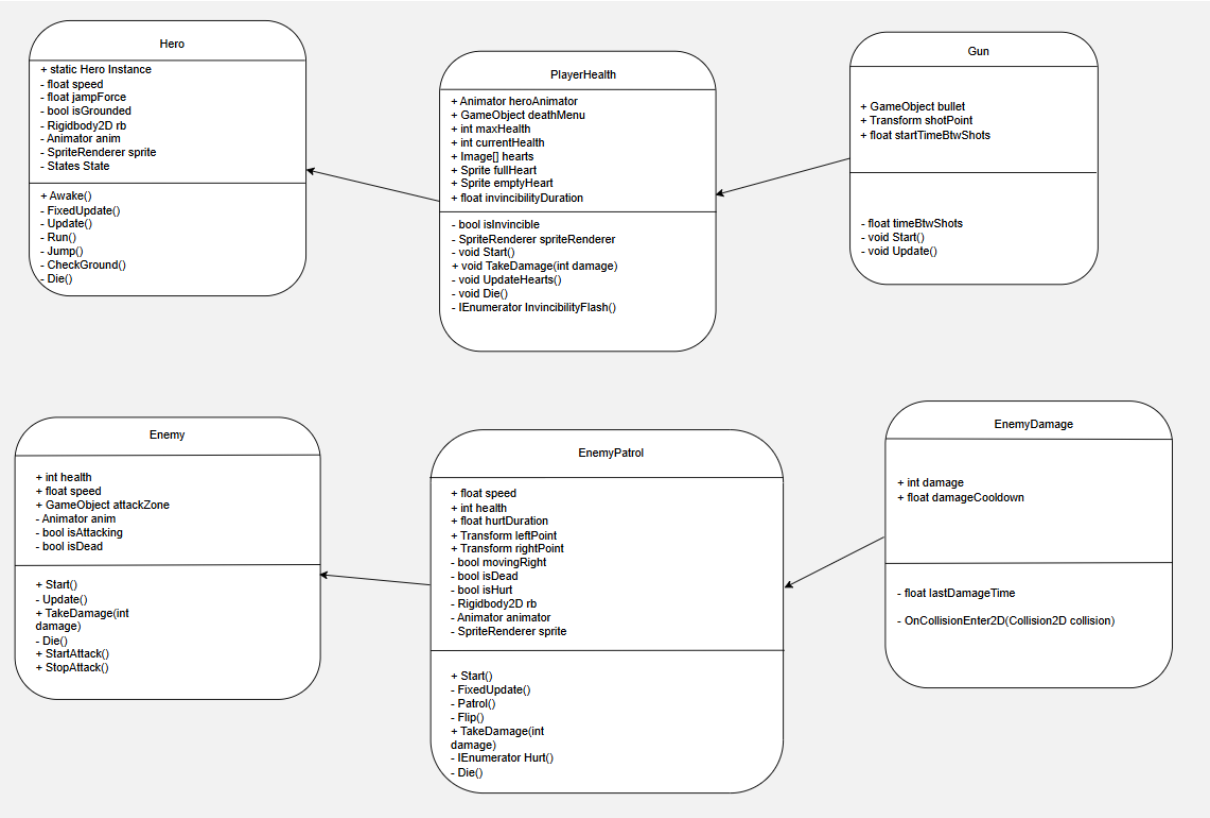

Рисунок 2.3 – Діаграма Класів

Джерело: розроблене автором

Позначки до рис 2.3:

- Скруглений квадрат позначає набір з перемінних та функцій;
- Плюс позначає змінна буде публічною;
- Мінус позначає що клас буде приватним;
- Стрілочка позначає прив’язку до батьківського об’єкта.

У Enemy можна побачити дві змінні, які відповідають за атаки коли ворог бачить гравця, а інша коли гравця не видно він продовжує патрулювати. Також не забуваємо, що у ворога прописана кількість здоров’я та шкода, яку він буде

завдавати гравцю. Патрулювання - це дві невидимі для гравця точки по яким він ходитиме.

Нижче приведу пояснення до кожної з показаних приватних та публічних класів докладніше по Hero, PlayerHealth та Gun.

Параметри в головного героя Hero:

- static Hero Instance це об’єкт героя в сцені, щоб скрипти які до нього були прикріплені змогли звернутись;

- float speed він відповідає за швидкість героя;

- float jumpForce даний параметр дає змогу коригувати сили стрибка в героя;

- bool isGrounded тут йде перевірка чи стоїть герой на землі;

- Ridgebode2D rd цей параметр дає герою фізику в переміщення по сцені;

- Animator anim відповідає за анімації в залежності в ситуації що робить гравець;

- SpriteRenderer sprite який використовується спрайт, точіше картинка на сцені;

- States State йде перелік станів у коді до героя;

- Awake () перевірка чи з’явиться герой на сцені при натисканні кнопки Play;

- FixedUpdate () перевірка на біг персонажа в коді коли гравець натискає кнопки;

- Update () перевірка на натискання кнопок;

- Run () функція, яка відповідає за біг гравця;

- Jump () функція для стрибка;

- Die () коли в гравця не стане здоров’я чи він впаде за карту програється смерть.

Тепер перечислимо пункти про здоров’є PlayerHealth:

- Animator hero відповідає за анімації пов’язані з здоров’єм коли гравець отримує шкоду;

- GameObject deathMenu меню, яке з’являється при смерті;

- int maxHealth параметр заданий через код для максимального здоров'я персонажа;

- int currentHealth тут буде поточне здоров'я персонажа;

- Image[] hearts масив з сердець, який вставляється в інспектор;

- Sprite fullHeart, emptyHeart це два спрайти картинок з повним та пустим серцем, які теж налаштовуються через інспектор;

- float invincibilityDuration затримка при отриманні шкоди, щоб в гравця був час відійти від противника;

- Start () метод який ставить максимальне здоров'я гравця в інтерфейсі;

- TakeDamage(int damage) перевірка на отриману кількість шкоди від противника;

- UpdateHearts () оновлення сердець в інтерфейсі від кількості шкоди;

- Die () перевірка коли в гравця не залишиться здоров'я то він помре;

- InvincibilityFlash() відповідає за мерехтіння героя при отриманні шкоди.

Наступним буде йти зброя Gun:

- GameObject bullet прив'язаний об'єкт кулі через інспектор;

- Transform shotPoint точка, яка прив'язана та зброї для пострілу;

- float startTimeBtwShots відповідає за затримку між пострілами;

- Start () ініціалізація таймеру на постріл;

- Update () перевірка на клік героя, щоб виконати постріл.

Залишилось ще пояснити про ворогів Enemy:

- int health тут позначено здоров'я ворога;

- GameObject attackZone позначаємо зону атаки противника;

- Animator anim йде підключення анімацій;

- bool isAttacking це перевірка чи акакує зараз ворог;

- bool isDead ще одна перевірка на смерть ворога;

- Start () налаштування противника для програвання анімації коли починається сцена;

- Update () тут йде перевірка на ситуацію в сцені та зміна анімацій від події;

-TakeDamage(int damage) перевірка на отримання шкоди від гравця;

- StartAttack () тут позначається атака ворога коли нас бачить;

- StopAttack () зупиняємо атаку для патрулювання коли не бачим героя.

Поясним як працює патрулювання EnemyPatrol:

- float speed зазначимо швидкість для противника;

- int health скільки в противника буде здоров'я;

- float hurtDuration тут йде невеличка затримка в анімація, щоб гравець побачив зміну анімацій в противника при отриманні шкоди;

- Transform leftPoint, rightPoint тут вказані 2 невидимих для гравця точки по яким пересувається противник;

- bool movingRight тут задаємо параметр для зміни спрайта по Y в залежності в який напрямок йде противник;

- bool isDead, bool isHurt позначимо в цих двох пунктах статуси ворога при якому він отримає шкоду і помре;

- Rigidbody2D rb добавляємо противнику фізику для переміщення;

- Animator animator, SpriteRenderer sprite в першому позначаємо анімації які перед цим налаштували в аніматорі. Інший відповідає за картинку, яку позначали на сцені;

- Start () налаштування для патруля;

- FixedUpdate () тут йдуть дві точки для переміщення противника по ним;

- Patrol () коли противник бачить героя він зупиняється і атакує, а коли втрачає його с поля зору повертається до патрулювання;

- Flip () відповідає за зміну напрямку ворога від сторони в яку він прямує;

- TakeDamage(int damage) перевірка на отримання шкоди;

- Hurt () оброблення шкоди від гравця;

- Die () ворог помирає коли закінчується життя.

Лишилось пояснить, як працює шкода від ворога EnemyDamage:

- int damage позначає кількість шкоди ворогу;

- float damageCooldown робим затримку між атаки, щоб противник не атакував постійно;

- OnCollisionEnter2D(Collision2D collision) тут позначений метод при якому ворог наносить шкоду персонажу, якщо він зітнеться з персонажем.

2.4 Опис архітектури програмного продукту

Усе починається з рушія Unity, де зібрані всі скрипти, асети та звуки. Налаштування сцени в проєкті розпочинається з того, що в сцені є чотири локації: три з них бойові, одна для відпочинку. Налаштування сцени здійснюється через скрипт Paralax, який відповідає за налаштування кожного зі слоїв, щоб при русі героя задній фон рухався з різною швидкістю й створював ефект глибини.

Після запуску гри користувач бачить фон із кнопками та логотипом гри. *У меню доступні п’ять кнопок:*

play початок гри;

options налаштування гучності звуків і музики;

save викликає меню для збереження гри або його закриття;

vaccine показує мету гри знайти кейс із антивірусом і зібрати склянки;

quit відповідає за вихід з гри;

У героя є один батьківський об’єкт, до якого під’єднано п’ять дочірніх об’єктів, кожен з яких виконує свою роль:

основний об’єкт Hero, на якому прикріплені всі важливі скрипти - Hero.cs, PlayerInventory, PlayerHealth.cs, Inventory.cs;

дочірній об’єкт з усіма анімаціями героя;

точка плеча Shoulder, до якої кріпляться руки. Спрайт рук, який обертається за напрямком миші. Зброя, до якої прив'язаний скрипт Gun.cs; там налаштовані об'єкт кулі та точка вильоту кулі.

Таким чином працює основний алгоритм руху й стрільби персонажа. Також у персонажа є інтерфейс із кнопкою для призупинення гри або переходу в меню. На екрані гравця відображається рюкзак з інвентарем - при натисканні на іконку рюкзака відкривається меню з п’ятьма слотами, також видно здоров’я гравця та кількість зібраних монет.

Кожен ворог має три об’єкти:

основний об’єкт із прикріпленими скриптами - Enemy.cs, EnemyPatrol.cs, EnemyDamage.cs.;

об’єкт спрайта з анімаціями;

зона атаки це невелика зона, при вході в яку ворог починає атакувати гравця, а при виході гравця з зони продовжує патрулювати.

Перехід відбувається через спеціальні зони на сцені, візуально виділені інтерактивними елементами й предметами. У кожній локації будуть противники й монетки для збору. Гравець рухатиметься далі, доки не знайде кейс із вакциною.

Висновок до розділу 2

У даному підрозділі були розглянуті основні діаграми, що описують архітектуру гри.

Діаграма варіантів використання Use Case показує взаємодію гравця з основними функціями гри, такими як початок гри, управління інвентарем та збереження прогресу. Діаграма активності ілюструє послідовність дій користувача під час гри, наприклад, пересування між локаціями або взаємодію з ворогами. Діаграма класів демонструє структуру основних класів проєкту та їхні взаємозв'язки.

Усі діаграми допомагають краще зрозуміти логіку роботи гри та взаємодію її основних елементів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ

3.1 Інструменти розробки

Розповідаючи про розробку продукту, потрібно обрати рушій. Серед тих, які я вже розглядав, це Unreal Engine, Godot і Unity. Кожен із них підходить для створення ігор у різних середовищах, але хочу коротко розповісти про них.

Рушій Unreal Engine більше підходить для 3D-простору та ігор від першої особи або з видом з-за спини. Його головна перевага - це чудова графіка, яка навіть при простому проєкті виглядає на високому рівні. Незважаючи на те, що цей рушій часто використовують великі студії, ним можуть користуватися і звичайні розробники, які хочуть створити власну гру. Приклад створених ігор: Spider-Man гра про людину – павука, Batman серія ігор про людину кажана;

Наступний рушій - Godot. Він також досить популярний серед розробників. Поширюється безкоштовно та підтримує як 2D, так і 3D розробку. З мого досвіду, його найчастіше використовують для створення мобільних ігор.

У перевагах можна перечислити наступне - відкритий код, не потрібно платити кошти за використання.

Приклади ігор на Godot: Until Then гра у стилі візуальної новели де привабливий піксельний стиль і цікавими кінцівками, Deponia гра в жанрі Point and Click, графіка була намальована від руки та з цікавими загадками.

Unity — рушій, який буде використаний у нашому проєкті. Це один із найпоширеніших і найвідоміших рушіїв, особливо зручний для 2D-ігор, зокрема платформерів.

Перечислимо інтерфейс у Unity - центральне вікно дає змогу переглядати та налаштовувати сцену, нижнє вікно відповідає за розміщені папки скрипти та інше, ліве вікно показує ієрархію де розробник бачить всі об’єкти на сцені та може створювати нові для редагування, праве віконце відповідає за інспектор, який дозволяє налаштувати будь-який вибраний об’єкт.

Можна сказати, що Unity зручно налаштовується, має зрозумілу структуру і з ним легко працювати.

Графічний редактор Aseprite – це популярна програма для розробки піксельної графіки, схожа на Photoshop, але більш спеціалізована. Буде використана у нашому проєкті для подальшого перегляду.

З плюсів цього додатка можна сказати - простий інтерфейс, який можна адаптувати під свої потреби, та зручне налаштування гарячих клавіш.

Графіка для нашого проєкту створюється саме в Aseprite. У демо можна побачити, як передано загальний настрій гри - депресивний, пригнічений, що добре пасує до задуманої атмосфери.

3.2 Написання коду для елементів гри.

Для початку потрібно налаштувати задній фон для зовнішнього виду локації (рис. 3.1). На цьому етапі ми створюємо шари в ієрархії проєкту зліва та дублюємо наш фон кілька разів, щоб розтягнути його в обидві сторони для перегляду камерою.

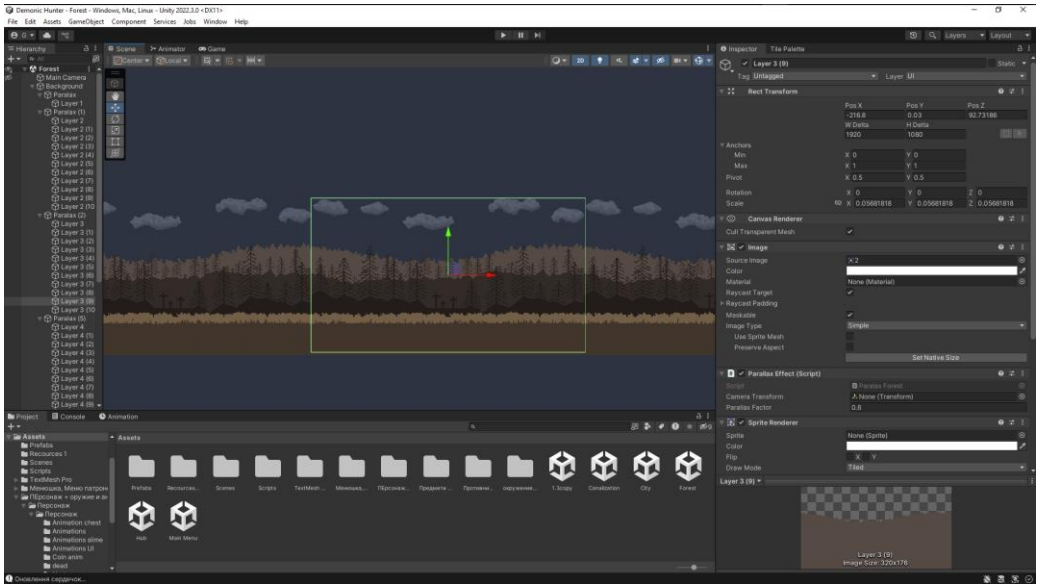

Рисунок 3.1 – Налаштування заднього фону

Джерело: розроблене автором

У подальшому для кожного з цих шарів потрібно буде створити скрипт у відповідних папках, який відповідатиме за паралакс-ефект — коли зображення рухаються з різною швидкістю, щоб передати відчуття глибини заднього фону та наложити його у інспектор.

Аби це добряче працювало та не виглядало статично через інспектор у кожного спрайта (рис. 3.2), який відповідає за фон ми робимо налаштування пересування саме цього фону відносно того, як буде рухатись гравець по рівню. Задаємо числа від 0.9 до 0.1 на самих дальніх ставим більші значення для ближніх шарів менші.

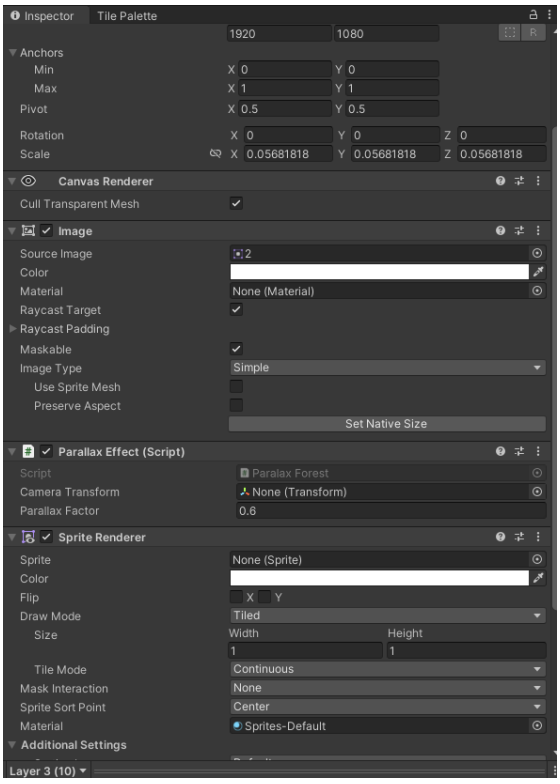

Рисунок 3.2 – Фотографія з інспектора

Джерело: розроблене автором

Нижче буде прикладений скрипт який відповідає за налаштування паралаксу і матиме назву Paralax.cs:

```
using UnityEngine;

public class ParallaxEffect : MonoBehaviour
```

```
{
    [SerializeField] private Transform cameraTransform;
    [SerializeField] private float parallaxFactor;

    private Vector3 previousCameraPosition;

    private void Start()
    {
        if (!cameraTransform)
            cameraTransform = Camera.main.transform;

        previousCameraPosition = cameraTransform.position;
    }

    private void LateUpdate()
    {
        Vector3 delta = cameraTransform.position - previousCameraPosition;
        transform.position += new Vector3(delta.x * parallaxFactor, delta.y * parallaxFactor, 0);
        previousCameraPosition = cameraTransform.position;
    }
}
```

Наступний етап коли вдалось відредагувати фон під розміри камери та налаштувати паралакс у інспекторі потрібно створити платформи на яких буде стояти гравець.

На цьому рис. 3.3 можна побачити створені платформи які будуть використовуватись для персонажу. Через ієрархію ми створюємо об’єкти редагуємо під розмір камери і добавляємо в Інспекторі компонент під назвою Box Colider 2D.

На рис. 3.4 зображений головний герой, який складається з кількох частин, а саме: персонаж, точка кріплення плечей, руки, зброя та точка стріляння. Оскільки ми створюємо гру з можливістю огляду на 360 градусів, подальше налаштування зброї повинно коригуватись вручну. Для кожного з цих об’єктів був створений окремий скрипт, а налаштування в інспекторі здійснювались на основі стандартних параметрів.

На рис. 3.5 можна побачити налаштування інспектора в цьому об’єкті. З’явились поля в інспекторі де потрібно перенести з ієрархії у відповідні поля.

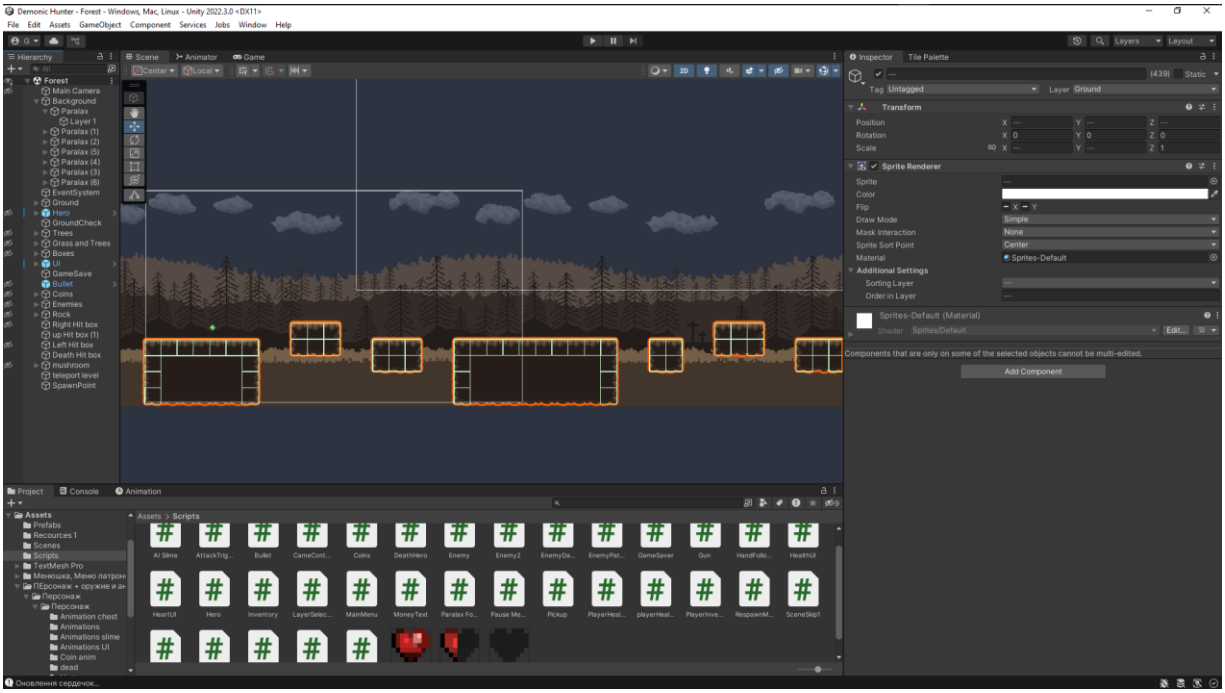

Рисунок 3.3 – Платформи для рівня

Джерело: розроблене автором

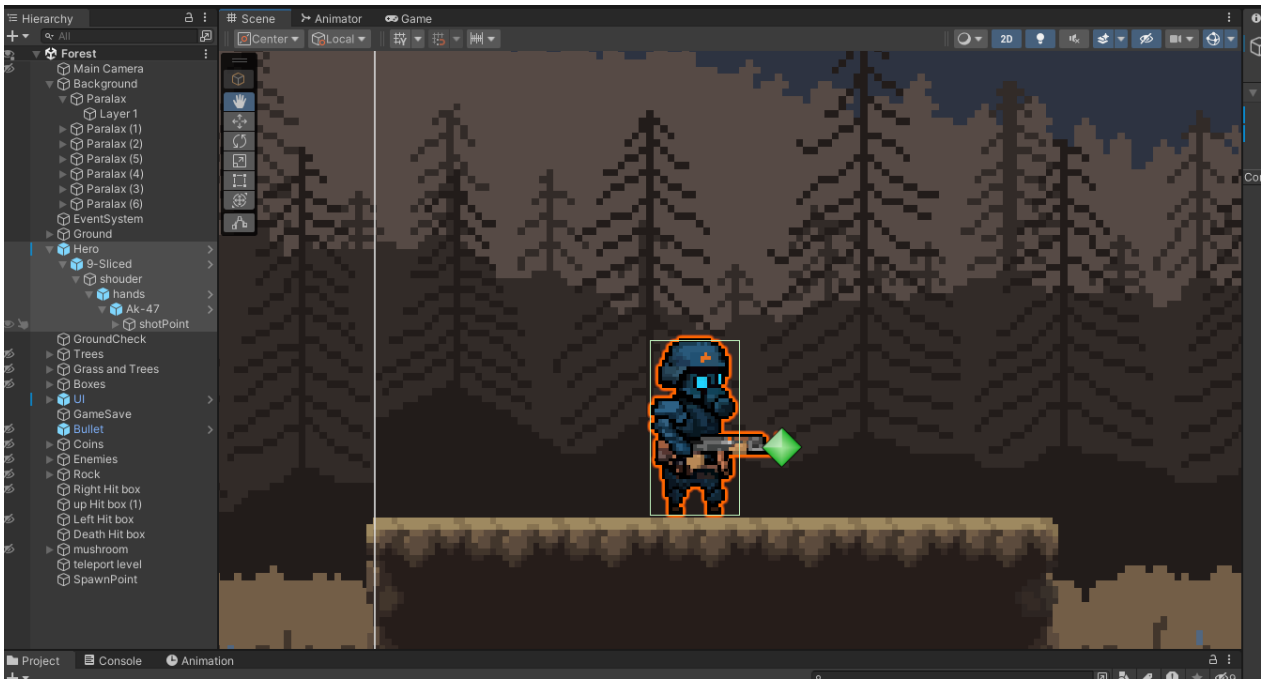

Рисунок 3.4 – Створення персонажу

Джерело: розроблене автором

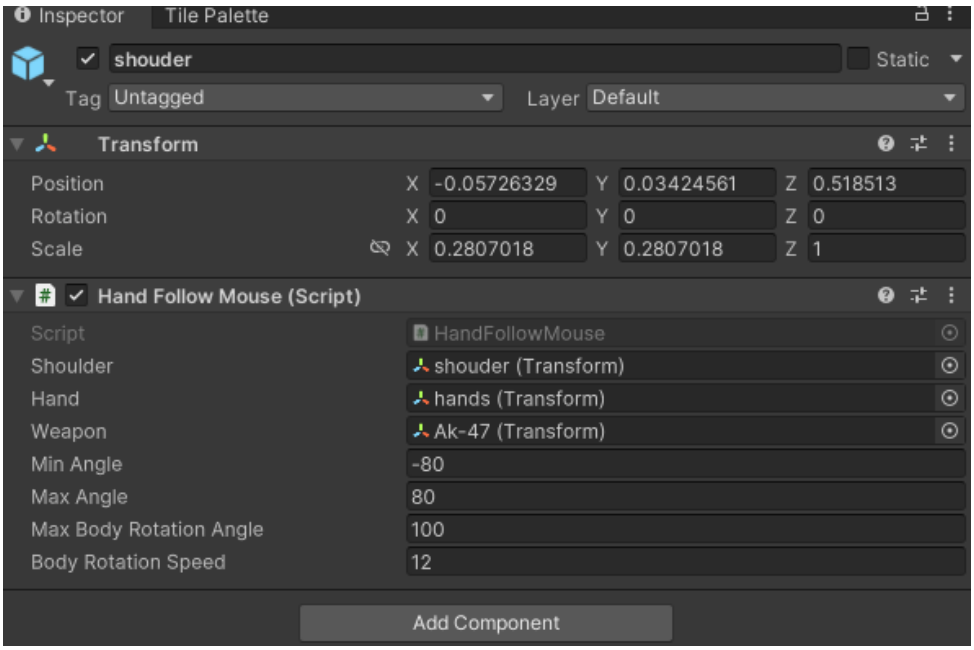

Рисунок 3.5 – Точка кріплення плеча

Джерело: розроблене автором

Код для об’єкту Gun.cs:

```
using UnityEngine;

public class HandFollowMouse : MonoBehaviour
{
   public Transform shoulder;
   public Transform hand;
   public Transform weapon;
   public float minAngle = -80f;
   public float maxAngle = 80f;
   public float maxBodyRotationAngle = 100f;
   public float bodyRotationSpeed = 12f;
   void Update()
   {
      if (shoulder == null || Camera.main == null) return;

      Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
      Vector3 direction = (mousePos - shoulder.position).normalized;
      float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;

      shoulder.rotation = Quaternion.Euler(0f, 0f, angle);

      if (hand != null)
         hand.rotation = Quaternion.Euler(0f, 0f, angle);
      if (weapon != null)
         weapon.rotation = Quaternion.Euler(0f, 0f, angle);
   }
}
```

На наступному етапі реалізуємо для персонажа зброю (рис. 3.6) та стрільбу з неї. Щоб усе працювало коректно, потрібно створити точку, з якої вилітатимуть кулі, а також окремо підготувати спрайт кулі.

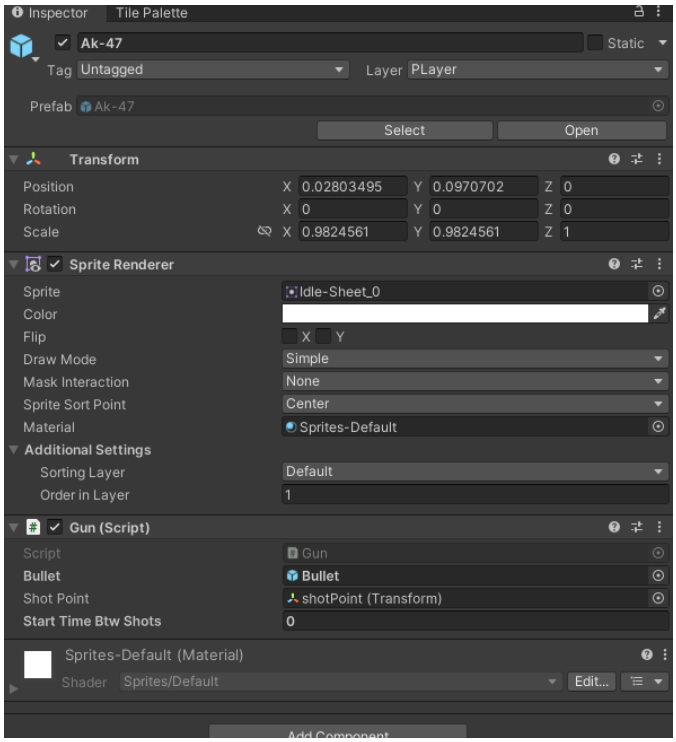

Рисунок 3.6 – Налаштування зброї

Джерело: розроблене автором

Наведено код, який відповідає за зброю - Gun.cs:

```
using System.Collections;
using UnityEngine;

public class Gun : MonoBehaviour
{
    public GameObject bullet;
    public Transform shotPoint;
    public float startTimeBtwShots = 0.5f;

    private float timeBtwShots;

    void Start()
    {
        if (bullet == null)
        {
            Transform foundBullet = transform.Find("Bullet");
```

```
            if (foundBullet != null)
            {
                bullet = foundBullet.gameObject;
                Debug.Log("Bullet prefab був знайдений як дочірній об'єкт.");
            }
            else
            {
                Debug.LogWarning("Bullet префаб не заданий!");
            }
        }

        if (shotPoint == null)
        {
            Transform foundShotPoint = transform.Find("ShotPoint");
            if (foundShotPoint != null)
            {
                shotPoint = foundShotPoint;
                Debug.Log("ShotPoint знайдено автоматично.");
            }
            else
            {
                Debug.LogWarning("ShotPoint не заданий!");
            }
        }

        timeBtwShots = 0f;

    void Update()
    {
        if (bullet == null || shotPoint == null)
            return;

        if (timeBtwShots <= 0)
        {
            if (Input.GetMouseButtonDown(0))
            {
                GameObject b = Instantiate(bullet, shotPoint.position, transform.rotation);
                Destroy(b, 2f);
                timeBtwShots = startTimeBtwShots;
            }
        }
        else
        {
            timeBtwShots -= Time.deltaTime;
        }
    }
}
```

Залишилось лише перевірити чи все правильно налаштована в інспекторі по об’єктах.

Далі коли в нас вже готовий персонаж, який вміє переміщюватись та стріляти з рушниць. Лишилось налаштувати йому анімації через компонент Animator. Вибираємо нашого персонажа і знаходимо у вкладинках Animation. Створюємо папку та на таймлайні добавляємо анімацію і потім їх редагуємо в Аніматорі (рис. 3.7).

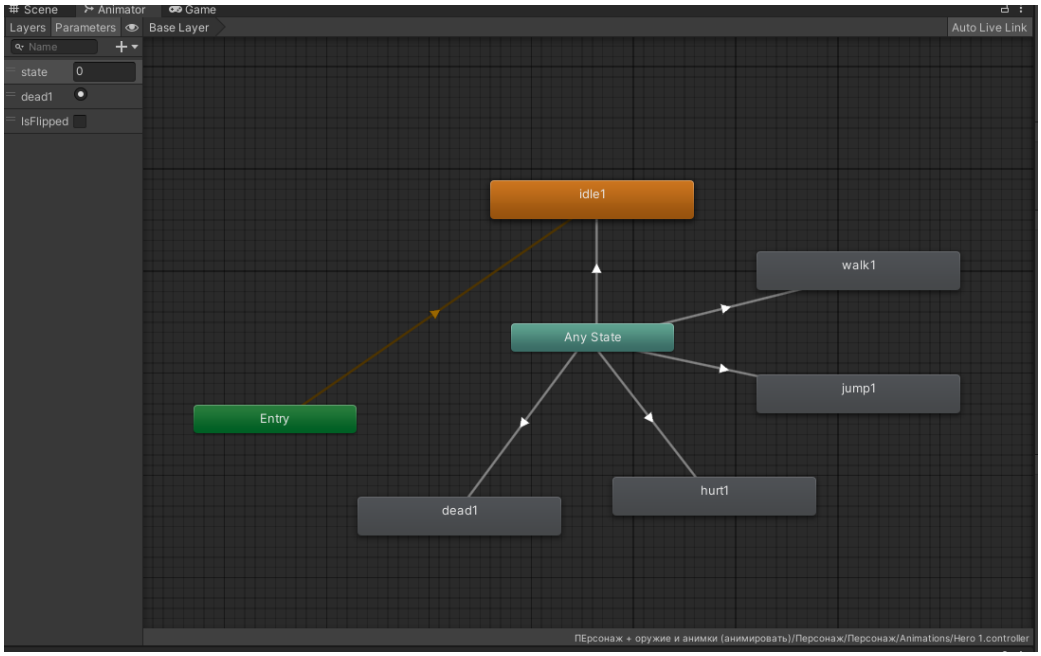

Рисунок 3.7 –Налаштування Аніматора

Джерело: розроблене автором

В ній ми керуємо правилами анімацій героя. Від стану Any State ми робимо порядок анімацій:

- idle анімація спокою;
- walk анімація ходьби;
- jump анімація стрибку;
- hurt анімація отримання шкоди;
- dead анімація смерті.

Також в самому інспекторі добавляємо правила, а точніше задамо номер анімації, яка буде викликатись від подій на сцені. Від анімації idle ставимо число 0 і до анімації смерті числа йдуть по порядку.

В наступному 3.8 етапі відтворимо ворогів, які будуть майбутньою загрозою на рівнях. Для кожного з ворогів буде написаний один скрипт який буде відповідати за здоров’я противника, шкоду, а також скрипт патрулювання.

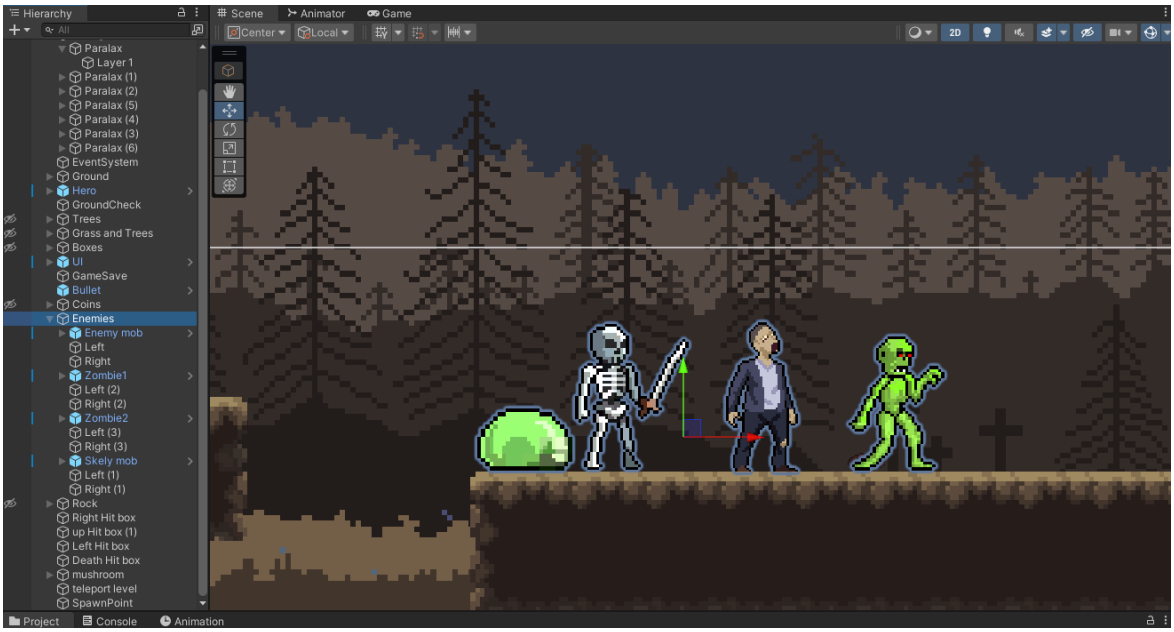

Рисунок 3.8 –Налаштування Аніматора

Джерело: розроблене автором

Настуний скрипт буде відповідати за поведінку противника на рівні. Щоб все працювало потрібно створити два пустих об’єкти через ієрархію і зазначити їх на рівні. Потім потрібно ці 2 елементи перетягнути у скрипт де присутні 2 відповідних поля EnemyPatrol.cs:

```
using System;
using UnityEngine;

public class EnemyPatrol : MonoBehaviour
{

    public float speed = 2f;
    public int health = 3;
    public float hurtDuration = 0.5f;
    private bool movingRight = true;
    private bool isDead = false;
    private bool isHurt = false;
    private Rigidbody2D rb;
    private Animator animator;
```

```
private SpriteRenderer sprite;
public Transform leftPoint;
public Transform rightPoint;

private void Start()
{
    rb = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();
}

private void FixedUpdate()
{
    if (!isDead && !isHurt)
    {
        Patrol();
    }
    else
    {
        rb.velocity = new Vector2(0f, rb.velocity.y);
    }
}

private void Patrol()
{
    if (movingRight)
    {
        rb.velocity = new Vector2(speed, rb.velocity.y);
        if (transform.position.x >= rightPoint.position.x)
            Flip();
    }
    else
    {
        rb.velocity = new Vector2(-speed, rb.velocity.y);
        if (transform.position.x <= leftPoint.position.x)
            Flip();
    }


}

private void Flip()
{
    movingRight = !movingRight;

    Vector3 scale = transform.localScale;
    scale.x = movingRight ? Mathf.Abs(scale.x) : -Mathf.Abs(scale.x);
    transform.localScale = scale;
}

public void TakeDamage(int damage)
{
    if (isDead) return;
```

```
        health -= damage;
        StartCoroutine(Hurt());

        if (health <= 0)
        {
            Die();
        }
    }

    private System.Collections.IEnumerator Hurt()
    {
        isHurt = true;
        animator.SetTrigger("hurt");
        yield return new WaitForSeconds(hurtDuration);
        isHurt = false;
    }



    private void Die()
    {
        isDead = true;
        animator.SetTrigger("dead");
        Destroy(gameObject, 1.5f);
    }
```

Наступний скрипт ворога відповідатиме за його здоров'я, підключення анімацій через аніматор та їх подальший виклик у коді. Скрипт матиме назву Enemy.cs:

```
using System.Collections;
using UnityEngine;

public class Enemy2 : MonoBehaviour
{
    public int health;
    public float speed;
    public GameObject attackZone;

    private Animator anim;
    private bool isAttacking = false;
    private bool isDead = false;

    void Start()
    {
        anim = GetComponent<Animator>();
        if (attackZone != null)
            attackZone.SetActive(false);
    }

    private void Update()
```

```
{
    if (isDead) return;

    if (health <= 0)
    {
        Die();
        return;
    }

    if (!isAttacking)
    {
        transform.Translate(Vector2.up * speed * Time.deltaTime);
        anim.SetBool("attack", false);
    }
}

public void TakeDamage(int damage)
{
    if (isDead) return;

    health -= damage;
    anim.SetTrigger("hurt");
}

private void Die()
{
    isDead = true;
    anim.SetTrigger("dead");
    anim.SetBool("attack", false);

    if (attackZone != null)
        attackZone.SetActive(false);

    Destroy(gameObject, 1f);
}

public void StartAttack()
{
    if (isDead) return;

    isAttacking = true;
    anim.SetBool("attack", true);

    if (attackZone != null)
        attackZone.SetActive(true);
}

public void StopAttack()
{
    if (isDead) return;

    isAttacking = false;
```

```
        anim.SetBool("attack", false);

        if (attackZone != null)
            attackZone.SetActive(false);
    }
}
```

На цьому малюнку 3.8, який представлений нижче можна розгледіти таку невеличку структуру. Тут що ми бачимо налаштований аніматор з усіма правилами та переходами в анімації до противника. Він присутній для кожного з противників на локації. Біля центральної сцени можемо побачити два параметри Layers та Parametrs. Нам потрібно зайти в параметри та призначити 4 пункти: 2 з них будуть мати перемінну bool, а 2 інших trigger. Далі до кожної з анімації буде правило в інспекторі.

На рис. 3.9 можна побачити такі об’єкти. На кожному з ворогів присутні Box Collider та Rigidbody. Перший компонент відповідає за колізії об’єкта, щоб ворог міг наносити шкоду персонажу та не був "прозорим" під час зіткнення. Другий компонент забезпечує фізичну взаємодію, зокрема переміщення.

Також біля ніг противника можна побачити ще одну зону колізії - це зона атаки. Коли герой входить у цю зону, противник зупиняється та починає атакувати. Щойно герой виходить за межі ворог припиняє атаку й повертається до патрулювання території.

Залишається доповнити інформацію про інші локації та їхнє наповнення. У грі планується чотири основні локації: Локація відпочинку, Мертвий ліс, Зруйноване місто та Каналізація. Тут буде коротко розглянуто, як реалізовано перехід між локаціями та їхнє наповнення.

На рис. 3.10 показано, як працює перехід між локаціями. Тут розробник бачить порожній об’єкт із квадратним колайдером, на який накладено скрипт, що відповідає за зміну локації.

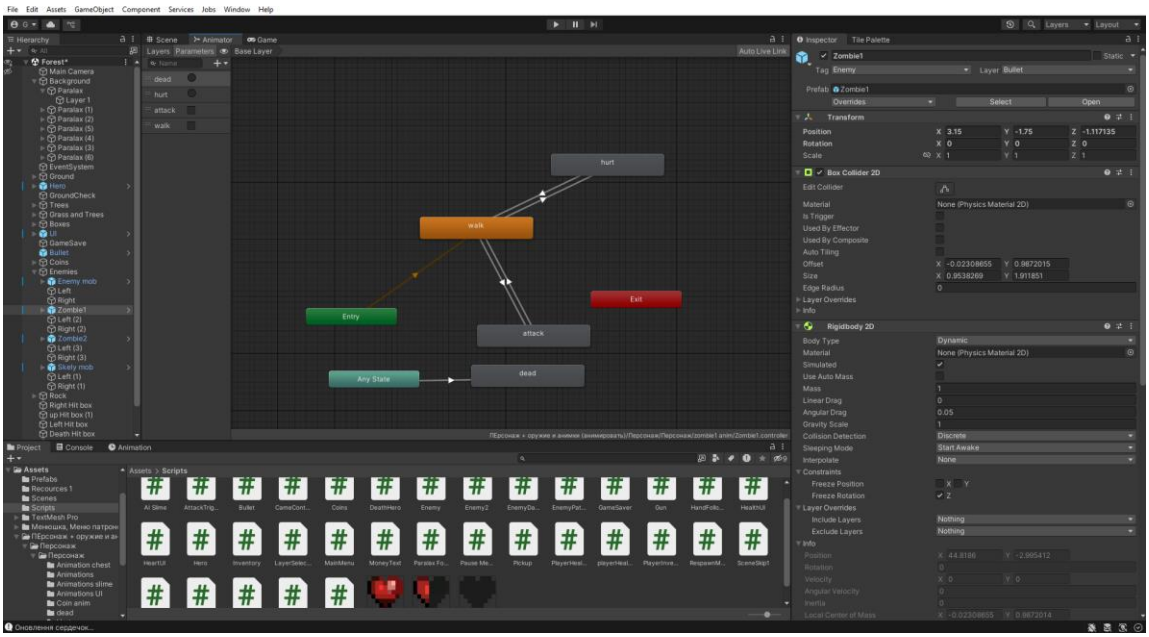

Рисунок 3.9 –Налаштування анімацій противника

Джерело: розроблене автором

Рисунок 3.10 –Налаштування противників

Джерело: розроблене автором

Лишається тільки в інспекторі правильно вписати назву локації, яку створили на новій сцені і через скрипт зробити перехід.

На рис. 3.11 зображено приблизний вигляд локації міста. На цьому фото не видно всієї локації але правіше ще присутні противники, блоки, та інше.

На рис. 3.12 можна побачити теж приблизне наповнення до локації каналізація. Тут зображено приблизний варіант створеного рівня на якому можна побачити монетки і противників.

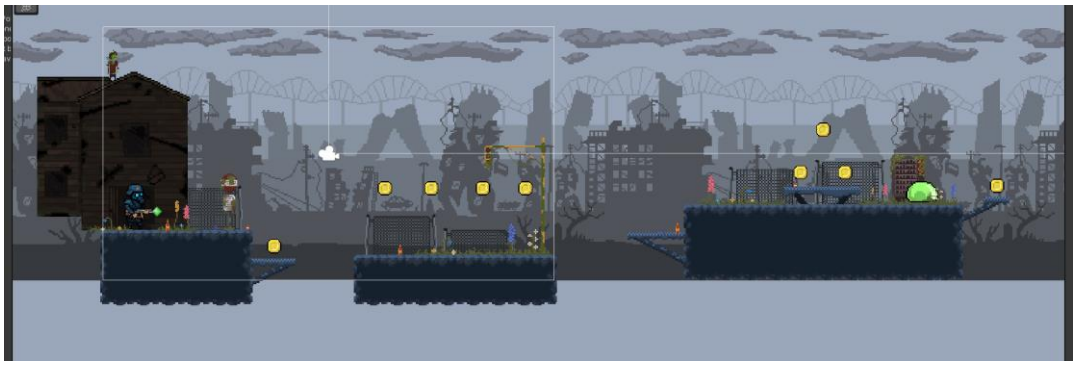

Рисунок 3.11 –Локація міста

Джерело: розроблене автором

Рисунок 3.12 –Локація каналізація

Джерело: розроблене автором

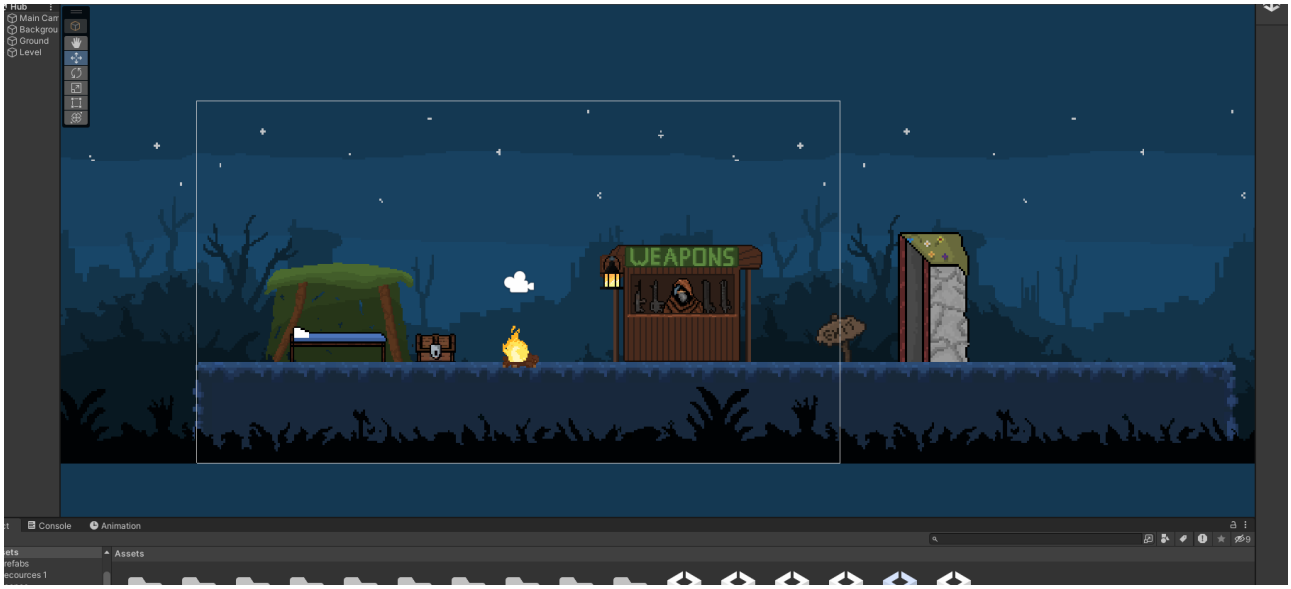

Рисунок 3.13 –Локація відпочинку

Джерело: розроблене автором

На цьому рисунку 3.13 можна побачити останню локацію. Вона буде відповідати за невеличкий відпочинок між бойовими локаціями. Можна буде перевести дух та поговорити з торгівельником.

3.3 Користувацький інтерфейс UI/UX.

У цьому пункті ми детально розглянемо весь користувацький інтерфейс нашого проєкту — від початку й до кінця. У цьому розділі буде розкрито інформацію про головне меню, анімації та пов’язані з ними скрипти.

Дивлячись на рис 3.14 можна побачити як буде виглядати головне меню в проєкті. На ньому зображено задній фон, якій був створений в програмі Aseprite і бачимо персонажа, а також логотип з функціональними кнопками. У кожної кнопки є анімація коли на неї натискаєш спрайт змінюється. При натисканні на кнопки можна буде побачити різні інтерфейси.

На рисунку 3.15 можна побачити функціонал кнопки Options та її налаштування. Тут використано два зображення, також додано дві кнопки й два слайдери . Кнопки OK та Cancel мають у інспекторі дві події OnClick, на

які реагують при натисканні. Перша подія відповідає за приховування інтерфейсу, а друга — за повернення до головного меню з кнопками.

Рисунок 3.14 –Локація відпочинку

Джерело: розроблене автором

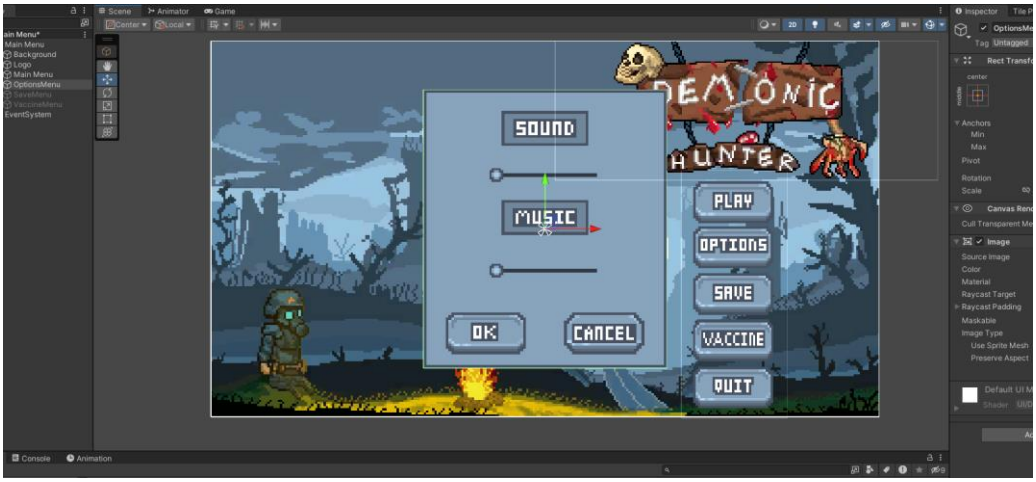

Рисунок 3.15 –Кнопка Options

Джерело: розроблене автором

В цьому рис 3.16 показано меню збереження гри. В нас є невеличке меню з 2 кнопками та картинкою з написом. Перша кнопка під назвою Cancel має

подію, яка просто закриває це меню з переходом в головне меню. Інша кнопка Save вона повинна відповідати за збереження по події Onclick

Рисунок 3.16 –Меню Збереження

Джерело: розроблене автором

Рисунок 3.17 –Меню Вакцини

Джерело: розроблене автором

На рисунку 3.17 зображено меню вакцини. Гравцеві потрібно буде знайти кейс із колбами антивірусу в одній із локацій. Як це повинно працювати - коли наприкінці гри ми піднімемо цей кейс, меню має оновитися. Також у

ньому присутні дві кнопки одна відповідає за перехід до головного меню, а інша змінює зображення та оновлює це меню.

Також не потрібно забувати скрипт головного меню в якому кнопка Play відповідає за завантаження рівня, потрібно лише вказати в самому коді назву сцени.

На рисунку 3.18 зображено користувацький інтерфейс гравця. На екрані можна побачити чотири елементи. Кнопка паузи відкриває відповідне меню - у цей момент гра призупиняється, і з'являються дві кнопки. Перша відповідає за продовження гри, а інша перекидає гравця до головного меню. Для обох кнопок налаштовані події через OnClick, які відповідають за взаємодію з різними меню.

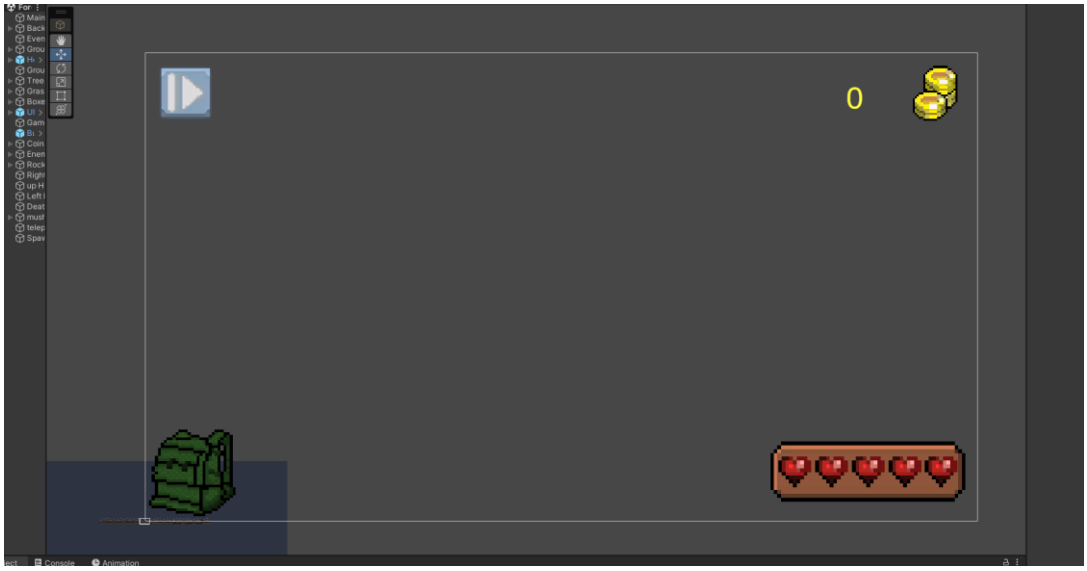

Рисунок 3.18 – Інтерфейс головного гравця

Джерело: розроблене автором

Наступним елементом в цьому інтерфейсі є іконка рюкзака — вона позначає наш невеликий інвентар. При натисканні на цю іконку з'являються 5 слотів для предметів, а також до кожного з них прив'язана кнопка Викинути предмет. Як це реалізовано: спочатку створюємо 5 порожніх квадратів, які підлаштовуємо під розмір інвентаря. Потім додаємо окрему кнопку, що

відповідає за викидання предмета. Потрібно в самому інспекторі ще розставити порядок для іконок від нуля до кількості слотів в моєму випадку п’ять.

І потрібно реалізувати підбір через предмети, які візьме гравець. Для цього нам потрібно зробити невеличкі махінації, а саме взяти предмет створюється для початку в інтерфейсі для того щоб зробити з нього іконку. І робимо такий же предмет але вже на рівні його потрібно перетворити на префаб і додати ще один скрипт. Потрібно в юніті створити папку під назвою Prefabs і додати туди цей предмет. Виходить що на рівні добавляємо йому Box Collider і ставим галочку на Triger аби його можна було підібрати.

Наступним що можна описати в рис 3.19 здоров’я персонажа. Потрібно на елементі інтерфейсу створити п’ять фотографій на яких будуть відображені серця. Їх потрбіно помістити під батьківський об’єкт і добавити їм такий скрипт який буде налаштований через інспектор. Потрібно добавити сюди всі серця з ієрархії також добавити картинку пустого та повного серяця які буду відображатись при отиманні шкоди.

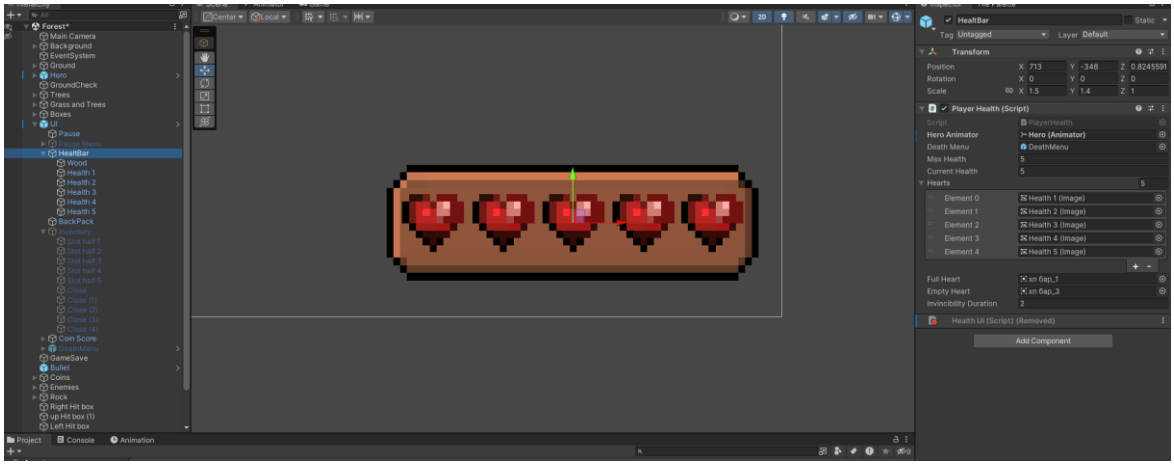

Рисунок 3.19 –Інтерфейс здоров’я

Джерело: розроблене автором

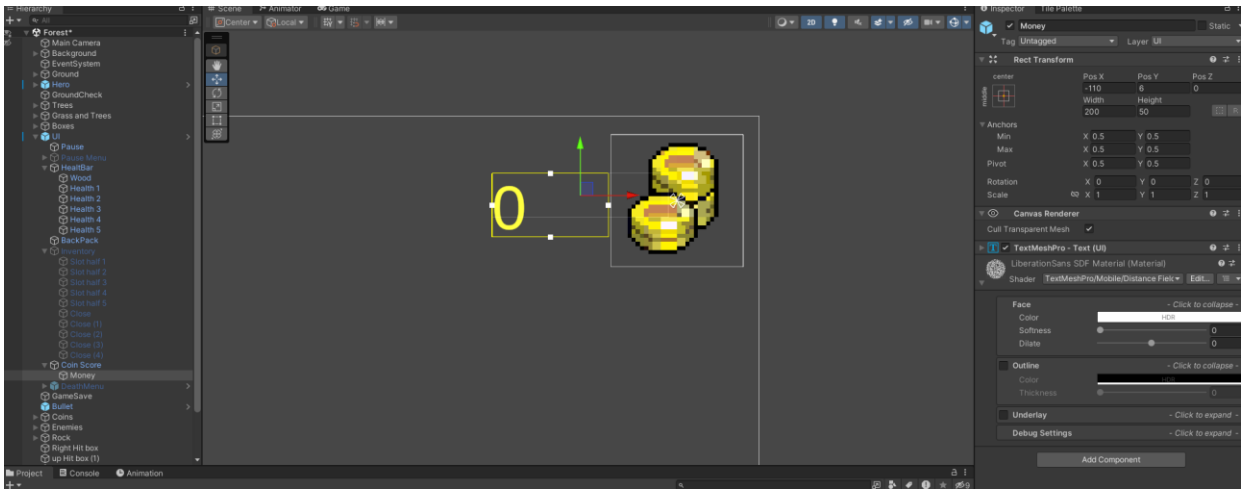

Рисунок 3.20 –Інтерфейс монет

Джерело: розроблене автором

Останні елементом нашого ігрового інтерфейса залишилось рахунок монет. Ми добавляємо фотографію монет плюс числовий рахунок.

3.4 Тестування.

З уже вище зазначеного я можу зробити невеличкий підсумок по майже готовому проєкту. В процесі виконання з додаванням нових скриптів та об’єктів часто з’являлись нові баги та проблеми, які кожен раз мені вдавалось вирішити. Наприклад:

- Зникання текстур при зміні значень в скриптах;
- Збивались кординати об’єктів на сцені;
- Проблема з анімаціями противників;
- Рушій неодноразово вилітав;

У проєкті на даний час більшість ідей вдалось втілити у дійсність. Проте і чимало ідей та концепцій не вдалось реалізувати.

Якщо пройтись по технічній частині по UI/UX з скриптами та іншим об’єктам. У більшості таких процесів відбувається проблеми тому в подальшій роботі їх вдавалось вирішувати. Говорячи про скрипти в них теж було чимало проблем.

Добавляючи в код щось нове могли збиватись інші налаштування, які пов’язані з цим об’єктом. Вирішувались в основному всі проблема перезавантаженням рушія. Але відбувалась не досить приємна ситуація коли весь проєкт не зберігся.

3.5 Використання.

Короткий опис дій, які може робити користувач у проєкті:

У грі ви керуєте персонажем, що досліджує пост-апокаліптичний світ. Переміщення здійснюється клавішами A, D, стрибок за допомогою кнопки Space, а прицілювання та стрільба мишею;

Натиснувши на кнопку рюкзака, ви можете відкрити інвентар і переглянути зібрані предмети;

На екрані відображається кількість здоров’я та монет;

Під час проходження ви зустрінете противників, уникайте їх атак і використовуйте зброю для захисту;

У спеціальних зонах які будуть чітко позначені можна переміщатись між локаціями;

На локації відпочинку доступна взаємодія з торговцем і збереження прогресу;

Мета гри, знайти кейс з антивірусом, подолати труднощі на рівнях, такі як платформінг і вороги, та не померти.;

У будь-який момент можна призупинити гру через кнопку, щоб повернутись в головне меню чи продовжити гру.

Висновок до розділу 3

У цьому розділі було детально описано програмну частину. Кожен ворог має власний код свій невеличкий інтелект, який реагує на нашого гравця.

Також було продемонстровано приклади ігрового інтерфейсу - від головного меню до функціональних кнопок, які реагують на дії гравця.

Інтерфейс створений у стриманій стилістиці, що гармонійно поєднується із загальним сетингом гри. Усі елементи спрямовані на створення унікального ігрового досвіду.

Також було коротко описано тестування гри на різні баги та проблеми, які виконувались і технічні зброї при розробці інших механік в реалізації.

Ну забуваємо про пам’ятку до гравця, саме тут було написано про управління головного героя, які будуть чекати труднощі на заготовлених рівнях і невеличкий екскурс по налаштуванням проєкта та іншим параметрам.

ВИСНОВКИ

З усіх перелічених вище пунктів можна підбити невеликий підсумок. Дивлячись зі сторони програміста, який не має великого досвіду в кодуванні, можу сказати одне, працювати над цим проєктом було дуже цікаво.

Говорячи про сама гру, цілі, які планувались реалізувати на базовому рівні, нам вдалося втілити хоча й не без труднощів. Щодо самої гри - вона виглядає досить унікально, має наповнені локації з власною стилістикою, які добре передають атмосферу. Противники реагують на гравця, мають свою поведінку.

Інтерфейс виконаний мінімалістично, не відволікає увагу — і, на мою думку, майже всі ідеї були реалізовані. Загалом, цей проєкт, за бажанням, можна викласти на безкоштовні платформи, щоб показати, що є розробники, які зацікавлені в такому жанрі і створюють щось нове.

Програмна частина була прописано коротенько і по змісту, щоб не було великої кількості непотрібно коду. Описані всі налаштування з анімаціями перевірками до них. Конкретика в розписі до кожного з пунктів у класах для точного опису.

Було проведене тестування в знаходженні проблем та багів. Виконувались перевірки та належне виправлення по коду аби все працювало без проблем та збоїв.

Реалізація цього проєкту є доцільною з огляду на популярність 2D-ігор у жанрі пост - апокаліпсису, особливо в піксельному стилі. Такий стиль дозволяє передати атмосферу з обмеженими ресурсами та водночас не потребує великих витрат на графіку.

Гра має потенціал зайняти свою нішу серед подібних інді проєктів. У майбутньому її можна буде викласти на безкоштовні платформи, щоб поділитися результатами з іншими гравцями та отримати зворотний зв'язок.

Це також стане хорошим прикладом реалізації повноцінної гри з нуля та чудовою практикою для подальших, масштабніших проєктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Жанри відеоігор URL: https://uk.wikipedia.org/wiki/%D0%96%D0%B0%D0%BD%D1%80%D0%B8_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80

2. Мода на ретро і проблеми сучаного піксель-арту URL: https://dtf.ru/flood/55677-moda-na-retro-i-problemy-sovremennogo-piksel-arta

3. Казуальні ігри: вдавана простота на межі високого Митецтва URL: https://www.prostranstvo.media/uk/kazualni-igry-vdavana-prostota-na-mezhi-vysokogo-mystecztva/

4. Doom 1993 URL: https://sourcegames.fandom.com/ru/wiki/Doom_(1993)

5. Як створювали гру "Doom": 17 цікавих фактів культового шутера від першої особи URL: https://nevsedoma.com.ua/672325-kak-sozdavali-igru-doom-17-interesnyh-faktov-kultovom-shutere-ot-pervogo-lica-16-foto.html

6. Дізнайтеся, як створювати піксельну графіку URL:https://www.adobe.com/ua/creativecloud/design/discover/pixel-art.html

7. Піксельна графіка URL:https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%BA%D1%81%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0_%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B0

8. Проєкт інді - розробника URL: https://2dmbie.itch.io/2dmbie

9. 2D платформер в пост-апокаліпсисі в жанрі – шутер URl: https://sazem.itch.io/scatteria

10. Гра в піксельному стилі під назвою - The Underground Man 2 URL: store.steampowered.com/app/1486950/The_Underground_Man_2/?l

11. Створення візуальних ефектів у Unity: посібник для розробників URL: https://foxminded.ua/ru/vizualnie-effekti-v-unity/

12. Ігри, у які грають роботи URL: https://nauka.ua/article/igri-u-yaki-grayut-roboti-navishcho-shi-vchat-gri-u-shahi-j-dota-2-ta-recept-peremog-vid-algoritmiv

13. Створення 2D гри на Unity за годину! Короткий гайд URL: https://itproger.com/ua/news/sozdanie-2d-igri-na-unity-za-chas-kratkiy-gayd

14. Синемашина для 2D: Поради та рекомендації URL: https://unity.com/ru/blog/engine-platform/cinemachine-2d-tips-and-tricks

15. Перша бойова система в 2D-платформері! URL: https://gamin.me/posts/17790

16. Мічківський С. Microsoft Office (Word, Excel, Outlook …) : навч. посіб. / С. Мічківський, Д. Балдик, В. Головань; Східноукр. нац. ун-т ім. В. Даля, Аграр. ф-т. – Київ : [Вид-во Східноукр. нац. ун-т ім. В. Даля], 2023. – 128 с. – URL: https://dspace.snu.edu.ua/handle/123456789/1723

17. Vysochyn I., Michkivskyy S. Using machine learning for translation and speech generation in e-book reading applications. Держава, регіони, підприємництво: інформаційні, суспільно-правові, соціально-економічні аспекти розвитку: матеріали VI Міжнародної наукової конференції (5-6 грудня 2024 р., м. Київ). Київ: Університет "КРОК", 2024. С.62-64 – URL: https://dspace.krok.edu.ua/items/6e1488e1-9e21-4282-af10-2e3bca48d2bc