МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІГРОВИЙ ЗАСТОСУНОК В ЖАНРІ ROGUELIKE НА РУШІЇ UNITY

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.22130202

Виконав студент 4-го курсу, групи 402

_____*О. В. Дуров*

«___» _____ 2023 р.

Керівник: канд. техн. наук, доцент

_____*І. О. Калініна*

«___» _____ 2023 р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20__р.

ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

Видано студенту групи 402 факультету комп'ютерних наук Дурову Олександру Володимировичу.

1. Тема бакалаврської кваліфікаційної роботи «Ігровий застосунок в жанрі roguelike на рушії Unity».

Керівник роботи Калініна Ірина Олександрівна, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____20__р. № _____

2. Строк подання студентом роботи «___» ___ 20__р.

3. Вхідні (початкові) дані до роботи: готові графічні та звукові елементи, інтерфейс користувача, а також розроблені моделі генерації рівнів та інші параметри гри.

4. Перелік питань, які потрібно розглянути (зміст пояснювальної записки):

- аналіз жанру roguelike та його елементів;
- вибір рушія Unity та його можливостей для розробки гри;
- розробка геймплею гри, включаючи створення гравцевої механіки, ворогів, предметів та зброї, системи управління та інтерфейсу користувача;
- розробка алгоритмів генерації рівнів та розміщення елементів;

– тестування та налагодження гри.

5. Перелік графічних матеріалів: презентація.

6. Завдання до спеціальної частини: «Аналіз робочого місця розробника програмного забезпечення з точки зору техніки безпки»

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Боженко А. Л. викладач	

Керівник роботи _канд. техн. наук, доцент, Калініна І. О._

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання __Дуров О. В._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_29_» _____квітня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Ігровий застосунок в жанрі roguelike на рушії Unity

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2022	30.10.2022	Виконано
2	Отримання завдання на виконання БКР	22.11.2022	24.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	08.12.2022	09.12.2022	Виконано
4	Отримання завдання на переддипломну практику	26.04.2023	29.04.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2023	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	16.05.2023	Виконано
7	Виконання БКР: аналіз сучасного стану прогнозування часових рядів, огляд існуючих технологій, розробка додатку	16.05.2023	28.05.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	31.05.2023	14.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	16.06.2023	
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.06.2023	19.06.2023	
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент Дуров О. В. _____

(прізвище та ініціали) (*підпис*)

Керівник роботи канд. техн. наук, доцент, Калініна І. О. _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«09» грудня 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Дурова Олександра Володимировича

Тема: «Ігровий застосунок в жанрі roguelike на рушії Unity»

Актуальність теми полягає в тому, що жанр roguelike набуває все більшої популярності серед геймерської спільноти.

Об'єкт дослідження є процес розробки та аналізу ігрового застосунку в жанрі roguelike, що характеризується випадково генерованими рівнями, перманентною смертю персонажа та стратегічним плануванням.

Предмет дослідження є інформаційні технології, методи та алгоритми для створення ігрового застосунку в жанрі roguelike. Дослідження охоплює проектування геймплею, аналіз і розробку алгоритмів генерації рівнів, реалізацію системи перманентної смерті персонажа, стратегічне планування та використання відповідних інформаційних технологій для цих процесів.

Метою даного дослідження є проведення аналізу та вивчення процесу створення захопливого ігрового застосунку у жанрі roguelike з використанням рушія Unity.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків.

У першому розділі розглядається класифікаційне дослідження та аналіз існуючих розробок комп’ютерних ігор.

У другому розділі описано розробку проекту та описані різні види рушіїв для розробки ігор.

У третьому розділі описано реалізацію гри.

У четвертому розділі описана інструкція для користувача.

Бакалаврська кваліфікаційна робота складається з 67 сторінок; 3 таблиці; 29 рисунків; 27 джерел.

Ключові слова: *гра, рушій, roguelike, алгоритм, актуальність, Unity*.

ABSTRACT

bachelor's qualification work of a student of group 402 at Petro Mohyla Black Sea National University

Durov Oleksandr

on topic: "Roguelike game application on Unity engine"

The relevance of the topic is that the roguelike genre is becoming increasingly popular among the gaming community.

The object of study is the process of developing and analyzing a roguelike game application characterized by randomly generated levels, permanent character death, and strategic planning.

The subject of research is information technology, methods and algorithms for creating a game application in the roguelike genre. The study covers gameplay design, analysis and development of level generation algorithms, implementation of the permanent character death system, strategic planning and the use of appropriate information technology for these processes.

The purpose of this study is to analyze and study the process of creating an exciting game application in the roguelike genre using the Unity engine.

The explanatory note consists of an introduction, four sections, conclusions.

The first section discusses classification research and analysis of existing developments in computer games.

The second section describes the development of the project and describes various types of engines for game development.

The third section describes the implementation of the game.

The fourth section describes the user manual.

Bachelor's qualification work consists of 67 pages; 3 tables; 29 drawings; 27 sources.

Keywords: *game, engine, roguelike, algorithm, relevance, Unity*.

ЗМІСТ

ВСТУП

Ігрова індустрія вже зараз займає лідируючу позицію на світовому ринку завдяки стрімкому розвитку технологій та появі Інтернету. Комп'ютерні ігри стали більш доступними та популярними серед людей різного віку. На сьогоднішній день, комп'ютерні ігри використовуються не лише для розваг та відпочинку, але й для навчання та симуляції, що стимулює розвиток комп'ютерних технологій.

У країнах СНД (Співдружність Незалежних держав), розвиток ігрової індустрії ще не досяг світового рівня через відсутність необхідних спеціалістів та компаній-розробників. Однак, тема розробки ігор дуже актуальна та потребує розвитку та появи нових спеціалістів, щоб мати можливість конкурувати з зарубіжними розробниками та виходити на світовий ринок геймінгу та конкурувати зі світовими лідерами.

У жанрі roguelike, ігровий засіб на базі Unity є дуже популярним серед геймерів. Цей жанр характеризується генерацією рівнів та головних героїв випадковим чином, що забезпечує велику кількість різних варіантів проходження гри [1]. Розробка таких ігор вимагає високих технічних навичок та спеціалізованих знань в галузі програмування. Але це може бути дуже цікавим та вигідним напрямком розвитку для компаній-розробників, які прагнуть вийти на світовий ринок геймінгу та конкурувати зі світовими лідерами.

1 КЛАСИФІКАЦІЙНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ІСНУЮЧИХ РОЗРОБОК КОМП'ЮТЕРНИХ ІГОР

1.1 Аналіз та загальна характеристика комп’ютерних ігор

Комп'ютерна гра – комп'ютерна програма, яка організовує хід гри в спілкуванні з партнером по грі або сама виступає в ролі партнера. У процесі комп’ютерних ігор за допомогою спеціальних програм за певними алгоритмами моделюється безпосередня взаємодія між ігровими персонажами та користувачами (або групами користувачів) у віртуальному просторі. Спостерігайте за грою на екрані монітора, як гравець впливає на неї за допомогою клавіатури, «мишки», джойстика тощо [1].

Комп’ютерні ігри можуть бути створені на основі сторонніх джерел, книг та фільмів, але існують і приклади зворотного напрямку, коли на основі якоїсь гри чи ігрової серії починають з’являтися додаткові матеріли(DLC), що розширюють всесвіт гри та додають активності в грі. Також існують випадки, коли комп’ютерні ігри стають джерелом інспірації для створення книг, коміксів, фільмів та інших творів мистецтва [2–3]. Наприклад, ігрова серія "Відьмак" стала основою для телесеріалу та книжкової серії, а гра "Assassin's Creed" вдихнула життя у кінематографічний фільм і комікси. Такі спільні всесвіти стають все більш популярними і дозволяють фанатам глибше досліджувати світ гри та отримувати нові емоції від улюблених персонажів.

Більш того, деякі розроблені ігри виступають в якості навчального матеріалу або дозволяють використовувати гравців в науково-дослідних цілях. За деякими популярними іграми проводяться змагання різних видів масштабності, від регіональних до світових, такі змагання називають спортом та вони отпримали свою назву «кіберспорт».

Комп’ютерні ігри мають істотний вплив на сучасне суспільство, що останнім часом з’являється стійка тенденція до гейміфікації для неігрового прикладного програмного забезпечення.

Таким чином, в деяких європейських навчальних закладах почали використовувати відомі ігри для навчання, а для потреб армій створюються симулятори для тренуванния солдатів [4]. Деякі країни зараховують кіберспорт до офіційного виду спорту, та, наприклад, уряд США у 2011 році визнав комп’ютерні ігри окремим видом мистецтва, поряд із театром та кіно. З усього цього можна зробити висновок, що комп’ютерні ігри щільно влились до нашого нинішнього життя. Сфера їх використання за останні роки продовжує зростати, ігри використовують не тільки для розваг, але й для проведення наукових досліджень та навчання.

Внаслідок того, що критерії приналежності гри до того чи іншого жанру не визначені однозначно, класифікація комп’ютерних ігор недостатньо систематизована, і в різних джерелах дані про жанр конкретного проекту можуть розрізнятися. Проте, існує консенсус, до якого прийшли розробники ігор, і приналежність гри до одного з основних жанрів майже завжди можна визначити однозначно.

За основний критерій поділу жанрів беремо дії, які найбільш часто здійснюються в іграх цього жанру. Ігри діляться на три великі групи: ігри контролю, ігри дії та ігри інформації. Всі ігри, що входять до групи дуже схожі між собою, але одночасно з цим мають різні відхилення.

Виділено 15 основних геймплейних елементів з яких складається взагалі будь-яка гра: навчання, загадки, спілкування, роль, вивчення, збирання, ухилення, знищення, змагання, техніка, турбота, розвиток, контроль, тактика, планування. Ці жанри були поділені на 3 різні класифікації.

Таблиця 1.1 – Жанрова класифікація комп'ютерних ігор

Категорія	Ігри інформації	Ігри дій	Ігри контролю
Гібридні жанри	Action-RPG Roguelike	MMOFPS Survival	RTS MOBA
5 елементів	Open RPG	Open Action	Global Strategy
2 елемента	Puzzle Quest BrowseRPG Adventure	Platformer Stealth-Action Fighting Racing	Economical Tower Defense Wargame Cardgame
1 елемент	Education Test Contact Hero Toure	Arcade Horror Shooter Sport Simulator	Logic Tactic MicroControl Building Life Sim
Елементарні жанри	Навчання Загадки Спілкування Роль Вивчення	Збирання Ухилення Нищення Змагання Водіння	Турбота Створення Контроль Тактика Планування

Золотою серединою групи є «RPG (RolePlaying Game)» – «рольова гра». Ігри, в яких можна жити, вживатися в роль героя, в яких в якості головних достоїнств виставляють атмосферу, сюжет, ігровий світ.

Ігри дії: Головне в іграх цієї групи – руху, які необхідний здійснювати керуючи якимось тілом (людським або гуманоїдним), або технічним засобом.

Золотою серединою групи є «Action» (гра-бойовик). Найбільш динамічні ігри. Саме ці ігри прийнято хвалити за те, що вони розвивають швидкість реакції.

Ігри контролю: Група «гри контролю» складається з тих ігор, головна суть яких – планування подій і управління для досягнення переваги в подальшому. Сюди потрапляють всі види стратегій, різні економічні ігри, варгейми, тактики. Золотою серединою групи є «Strategy» (звичайна локальна стратегія).

Усі ігри поділяються на одиночні та мультиплеєрні. Також мультиплеєрні ігри поділяються між собою на:

– мультиплеєр на одному комп’ютері;

– мультиплеєрні оффлайн-ігри;

– масові онлайнові.

Саме завдяки цій класифікації встановлюються режими, що будуть присутні у грі. Одиночна гра – вид гри, у яких приймає участь одна людина. Зазвичай гравцю протистоїть нейромережа, а його метою є рух до кінця гри (проходження), накопичення ресурсів або прокачування навичок. Часто ці цілі комбінуються.

Іншим видом ігр є мультиплеєр. Цей режим гри пристосований до гри більше ніж однієї людини одночасно. В багатьох іграх одиночна гра там мултиплеєр комбінуються.

За візуальною складовою комп’ютерні ігри поділяються на:

– текстові – гра з мінімальною кількістю графічних елементів, а спілкування з гравцем відбувається за допомогою тексту;

– 2D – усі елементи гри розроблені за допомогою двовимірної графіки;

– 3D – усі елементи гри розроблені за допомогою тривимірної графіки.

Після ретельного аналізу класифікацій комп'ютерних ігор, було прийнято рішення розробити гру у жанрі Roguelike. Цей жанр має свої особливості, але він також має своїх прихильників. Головною особливістю Roguelike є процедурно-генерований світ, що забезпечує неповторність гри кожного разу, коли гравець починає заново. Гра має також високий рівень складності, де кожен крок може призвести до смерті гравця та повернення до початку гри.

У Roguelike гравець повинен досліджувати випадково згенеровані локації, збирати предмети та битися з різними монстрами. Ігровий процес зазвичай відбувається по ходах, тобто гравець зробив хід, потім монстри зроблять свої ходи. Крім того, у грі немає можливості зберегти свій прогрес під час гри, тому кожна спроба заново - новий шанс дійти до кінця гри.

1.2 Аналіз існуючих розробок

Roguelike – це жанр комп'ютерних ігор, який характеризується процедурно генерованими ігровими рівнями та безліччю персонажів, включаючи ворогів та NPC. Гравцеві потрібно проходити рівні, збирати різноманітні предмети, зброю, артефакти, інколи працювати з персонажами інших гравців, виконувати завдання та боротися з головними босами.

У рогаликах рівні зазвичай змінюються з кожним запуском гри, що робить кожну гру унікальною та непередбачуваною. Гравець може зіткнутися з непередбачуваними ситуаціями та знайти нові рішення в різних ситуаціях. Зазвичай зростає рівень складності з кожним рівнем. Гравцю необхідно бути уважним та обережним, оскільки на кожному кроці його чекають небезпеки. Основний принцип рогалика полягає у тому, що кожна гра – це новий досвід, який вимагає від гравця адаптації та уміння швидко реагувати на змінні умови. Гравець може зіграти за різних персонажів з різними вміннями та здібностями. Такі персонажі можуть бути різними за статтю, расою, класом та іншими параметрами, що дозволяє гравцеві вибирати той, який найбільше відповідає його ігральному стилю.

Жанр roguelike виник на початку 1980-х років. Його назва походить від гри Rogue, яка була випущена в 1980 році на комп'ютерах серії Unix. Rogue стала першою в своєму роді і створила основу для всього жанру. У Rogue гравець керує головним героєм, який провалюється в лабіринт, повний монстрів та скарбів. Головна мета полягає в тому, щоб дістатися до найглибшого рівня лабіринту, знайти Амулет Йолкіра і повернутися до поверхні, при цьому уникнувши небезпек та збираючи всілякі знаряддя.

В наступні роки після випуску Rogue, виникло багато інших ігрів, які використовували ті самі принципи та особливості. До найвідоміших з них належать NetHack (1987), Angband (1990) та Dungeon Crawl (1997). З часом жанр розвивався

та став більш складним та різноманітним, з'явилися нові елементи гри, такі як графіка та звукові ефекти [5].

Сьогодні жанр roguelike є досить популярним та продовжує заробляти шанувальників. У ньому ви зможете знайти велику кількість цікавих ігор, які зможуть викликати вам виклики та приємні враження. Оскільки технології з кінця 1980-х отримали стрімкий розвиток, аналізувати будемо саме сучасні приклади ігр roguelike, такі як:

– Hades;
– BPM – Bullets Per Minute;
– Curse of the Dead Gods;
– UnderMine.

1.2.1 Hades

Hades – це екшн-рогалик, розроблений Supergiant Games та випущений у вересні 2020 року. Гра отримала високі оцінки критиків та гравців за геймплей, сюжет, візуальний дизайн та музику.

У грі Hades гравці керують Загребою, сином грецького бога Ада, який намагається втекти з підземного царства та дістатися до світу живих. Загреба має битися з безліччю монстрів, щоб дістатися до поверхні. Однак, кожен раз, коли гравець помирає, Загреба повертається до підземного світу та починає свій шлях спочатку [6].

Головна особливість гри полягає в системі прокачування. Кожна спроба втечі дає гравцеві нові можливості для покращення характеристик свого героя. Гравець може використовувати різноманітні знаряддя, що знаходяться на своєму шляху, і прокачувати їх, щоб зробити свої наступні спроби втечі більш успішними.

Крім того, Hades вражає своїм візуальним дизайном, який поєднує в собі класичні мотиви грецької міфології з сучасними елементами. Гра також отримала

велику кількість похвальних відгуків за свій саундтрек, який був написаний Дарреном Корбеттом.

Рисунок 1.1 – Знімок екрана з гри Dead Cells

1.2.2 BPM

Bullets Per Minute – це шутер-рогалик з музичними елементами, розроблений однією людиною, Андрео Джовані, і випущений в вересні 2020 року. Головна особливість гри полягає в тому, що дії гравця синхронізуються з ритмом музики, що створює неповторний геймплей та атмосферу.

У грі Bullets Per Minute гравець бере на себе роль військового бога, який бореться зі злими духами, що завдають шкоди світу [7].

Одна з основних особливостей гри – це система відновлення здоров'я. Гравець може відновити своє здоров'я, якщо він успішно стріляє та рухається в такт музиці. Якщо ж гравець пропускає ритм музики, то його здоров'я не відновлюється.

Також гра має систему прокачування, яка дозволяє гравцю вдосконалювати свої здібності та знаряддя для боротьби зі злими духами. Гравець може знайти різноманітні зброї, броню та різні предмети на кожному рівні, щоб зробити своє наступне проходження ефективнішим.

Рисунок 1.2 – Знімок екрана з гри Bullets Per Minute

1.2.3 Curse of the Dead Gods

Curse of the Dead Gods – це екшн-рогалик, розроблений студією Passtech Games і випущений в лютому 2021 року. Гравець бере на себе роль відважного індіана Джонатана, який відправляється в давні храми, щоб знайти скарби і відшукати відповіді на свої питання.

Один з основних елементів гри – це система керування прокляттям. Гравець може активувати певні прокляття на початку кожного рівня, які надають йому різні переваги, але його ціна залежить від складності прокляття [8].

Також гравець може вдосконалювати свої здібності та отримувати нові зброї та предмети, щоб допомогти йому в боротьбі з ворогами. Гра має гнучку систему прокачування, яка дозволяє гравцю підлаштовувати свої здібності до свого стилю гри.

Окрім цього, Curse of the Dead Gods має досить незвичну бойову систему, де гравець може виконувати комбо, швидкі та сильні атаки, а також робити блокування та ухилятися від ударів ворогів. До того ж гра має дуже деталізовану графіку, що дозволяє поглибитися в атмосферу давніх храмів та руїн.

Рисунок 1.3 – Знімок екрана з гри Curse of the Dead Gods

1.2.4 UnderMine

UnderMine – це інді-рогалик, розроблений студією Thorium Entertainment і випущений в серпні 2020 року. Гравець бере на себе роль безіменного героя, який відправляється в підземелля, щоб зібрати ресурси, прокачати свої навички та боротися з небезпечними ворогами.

Головна механіка гри полягає у зборі золота та ресурсів, які гравець може використовувати для купівлі предметів, збільшення своїх здібностей та прокачування свого героя [9].

Одним з унікальних елементів гри є система ковтання, де гравець може з'їдати різні зілля та еліксири, які надають йому різні бонуси та переваги, але також можуть мати негативні наслідки. Гравець також може знаходити реліквії, які надають йому постійні бонуси та здібності.

UnderMine також має досить складну бойову систему, де гравець може виконувати комбо, ухилятися від ударів та використовувати свої здібності, щоб перемогти ворогів. Гра має різноманітність ворогів, які мають свої унікальні здібності та способи атаки.

Рисунок 1.4 – Знімок екрана з гри UnderMine

1.2.5 Таблиця порівняння різних ігор

Ця таблиця надає загальний огляд основних характеристик ігор Hades, BPM, Curse of the Dead Gods і UnderMine, таких як жанр, платформи, графіка, геймплей, режим гри, генерація рівнів, розвиток персонажа та критики/оцінки.

Таблиця 1.2 – Таблиця порівняння ігор

Характеристика	**Hades**	**BPM**	**Curse of the Dead Gods**	**UnderMine**
Жанр	ролева гра, roguelike	ритм-екшн, roguelike	екшн, roguelike	рогалик, екшн
Графіка	2D	3D	2D	2D
Геймплей	виживання, пошук зброї, битви з монстрами	ритмічні поєдинки, стрільба з пуль, генерація рівнів	дослідження підземелля, збирання скарбів, битви з ворогами	розкопування підземелля, збирання ресурсів, боротьба з монстрами
Режим гри	одиночний гравець	одиночний гравець	одиночний гравець	одиночний гравець
Генерація рівнів	процедурна	процедурна	процедурна	процедурна

Закінчення таблиці 1.2

Розвиток персонажа	збір божественних артефактів, отримання навичок, покращення зброї	відкриття нових зброї, здібностей та покращень	відкриття нових зброї, здібностей та покращень	збір артефактів, покращення здібностей персонажа
Критики і рейтинги	дуже позитивні відгуки, високі оцінки	добрі відгуки, середні оцінки	позитивні відгуки, високі оцінки	добрі відгуки, середні оцінки

Загальні ознаки, які можна виділити у вказаних іграх, є наступними:

– рогалик: Всі чотири ігри відносяться до жанру рогалик, що означає процедурну генерацію рівнів, випадковість подій і ворогів, а також втрату прогресу після смерті персонажа;

– одиночний режим: Всі ігри пропонують тільки одиночний режим гри, де гравець виступає в ролі одного персонажа і розробляє стратегію самостійно;

– збирання ресурсів та покращення: У всіх іграх присутні механіки збирання різних ресурсів, артефактів або зброї, які можуть покращувати вміння або здібності персонажа;

– екшн та битви з монстрами: Всі чотири ігри пропонують динамічний геймплей з акцентом на битви з різними монстрами та ворогами.

1.3 Рогалики в майбутньому: нові горизонти геймплею та технологій

Жанр roguelike, хоч і не є найбільш популярним серед комп'ютерних ігор, все ж зберігає свою віддану аудиторію та знаходить нових шанувальників. Розглянемо деякі статистичні дані, що описують стан цього жанру [10].

За даними SteamDB, у 2023 році у світі випущено більше 440 нових roguelike-ігор [11]. У той же час, за останій рік понад 9 мільйонів гравців грали в гру Binding of Isaac, яка є однією з найвідоміших roguelike-ігор.

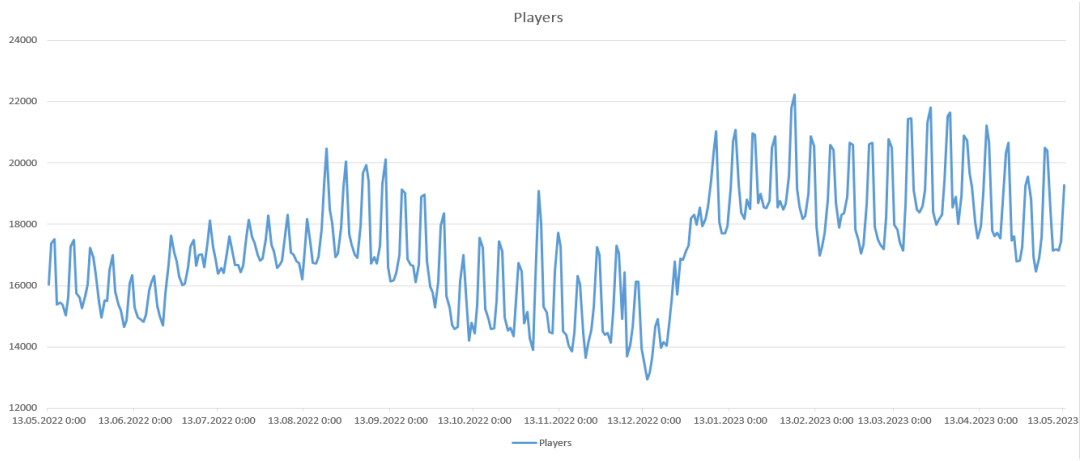

Рисунок 1.5 – Графік гравців за період 13.05.22 – 13.05.23

За даними порталу GitHyp, найпопулярнішою roguelike-ігрою на Twitch у 2022 році була Vampire Survivors. Однак, загалом, розділ roguelike не є найбільш популярним серед стрімерів та глядачів Twitch, займаючи тільки 1% загальної кількості глядацьких годин на платформі. На момент напису найпопулярнішою грою на Twitch є Muck та Dead Cells [12].

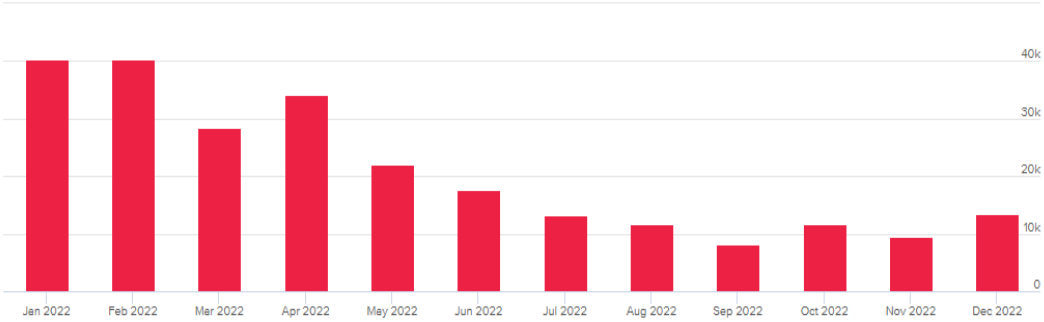

Рисунок 1.6 – рейтинг глядачів Vampire Survivor за 2022 рік

Тренди в розвитку roguelike-ігор також змінюються з часом. Якщо раніше були популярними ігри з різними видами зброї та ворожими монстрами, то в

останні роки розробники стали більше уваги приділяти механіці генерації випадкових рівнів та подій. Також стає все більш популярним режим гри в кооперативі, коли декілька гравців можуть пройти гру разом.

Отже, можна зробити висновок, що жанр roguelike не є найбільшим та найпопулярнішим серед комп'ютерних ігор, але все ж зберігає свою аудиторію та знаходить нових шанувальників.

Що стосується майбутнього розвитку цього жанру, можна очікувати, що розробники продовжуватимуть експериментувати з механіками генерації випадкових рівнів та подій, а також збільшуватимуть кількість доступних персонажів з унікальними вміннями та здібностями.

Режим гри в кооперативі вже зараз є досить популярним серед геймерів, але можна очікувати, що в майбутньому цей тренд ще більше зростатиме. Існує кілька причин, чому кооперативний режим буде привабливим для гравців.

По-перше, гра в кооперативному режимі дозволяє гравцям спільно пройти гру разом з друзями або іншими гравцями онлайн. Це створює можливість спілкування, взаємодії та спільної праці для досягнення спільних цілей. Граючи разом, гравці можуть обмінюватися досвідом, використовувати стратегію та координувати свої дії для успішного проходження гри. Це розширює соціальний аспект гри та додає взаємодії між гравцями.

По-друге, кооперативний режим дозволяє гравцям поділитися викликом та емоціями гри. Вони можуть разом переживати складності та успіхи, радіти досягненням та вирішувати головоломки разом. Це зближує гравців і створює спільні спогади та емоції, які можуть бути незабутніми.

Ще одним трендом в розвитку roguelike-ігор збільшення ролі наративу та історії є ще одним трендом в їх розвитку. Раніше такі ігри були спрямовані переважно на геймплей та випадковість подій, але тепер розробники все більше звертають увагу на наративну складову.

Зростання наративу в roguelike-іграх також додає різноманітності в процес гри. Він стає менш передбачуваним і більш цікавим, оскільки історія може

змінюватися в залежності від дій гравця. Це стимулює повторне проходження гри, адже кожне нове проходження може розкрити нові аспекти історії та світу гри.

Усі ці тенденції можуть привести до створення ще більш складних та глибоких ігор у жанрі roguelike, які зможуть привернути ще більше уваги гравців та стати популярнішими на ринку комп'ютерних ігор.

Жанр roguelike також може отримати популярність через розвиток віртуальної реальності та інших інноваційних технологій. Застосування VR-технологій дозволить гравцям більш інтенсивно занурюватися в гру та відчувати себе частинкою ігрового світу. Також розробники можуть експериментувати з механікою гри, щоб зробити її більш захопливою та динамічною, з використанням додаткових функцій та можливостей, які пропонують нові технології.

За останні роки, кількість roguelike-ігор зросла настільки, що багато з них вже стали піджанром, що відрізняються унікальними механіками та характеристиками. Наприклад, можна виділити такі піджанри, як "roguelite", "roguelike-like", "roguelike RPG" та інші.

Гра "Dead Cells" є прикладом "roguelite" ігри, яка містить в собі елементи жанру roguelike, але також має значний фокус на бойовій системі і прогресії персонажа.

Гра "Spelunky" є прикладом "roguelike-like" ігри, яка має схожу механіку генерації випадкових рівнів та постійної смертності персонажа, але може бути менш важкою і має менше елементів RPG.

Гра "Darkest Dungeon" є прикладом "roguelike RPG" ігри, яка має сильний фокус на розвитку персонажа та нарративі, але також має елементи випадкових рівнів та постійної смертності персонажів, що характерні для жанру roguelike.

Кожен з них має свої особливості та характеристики, які можуть привернути різні групи гравців.

Жанр roguelike постійно розвивається та еволюціонує, пропонуючи гравцям нові інноваційні ідеї та механіки гри. Останнім часом цей жанр став більш

доступним та зрозумілим для новачків завдяки спрощеним версіям, таким як "roguelite" або "roguelike-like".

Завдяки постійному розвитку та експериментації з новими ідеями, жанр roguelike має потенціал привернути ще більше гравців та зайняти своє місце на ринку комп'ютерних ігор.

Жанр roguelike є одним з тих жанрів, який може забезпечити безліч годин насолоди гравцям, завдяки своїм випадково згенерованим рівням і постійній смертності персонажів. Однак, він також може виявитися надзвичайно вимогливим та складним для тих, хто тільки починає грати у цей жанр. Тому важливо для розробників забезпечувати належну підготовку гравців до складності та випадковості ігрового процесу.

Одним з викликів для розробників є створення різноманітних та збалансованих геймплейних механік, які б дозволили гравцям відчувати себе унікальними та давали можливість змінювати ігровий досвід залежно від їх власних уподобань. Також важливо розвивати нарративний аспект гри, щоб створити цікавий та захоплюючий світ, який би зацікавив гравців і надихнув їх на подальшу гру.

Жанр roguelike також може бути цікавим для використання у навчальних цілях, наприклад, для розвитку навичок прийняття рішень, розвитку стратегічного мислення та аналітичних навичок. Деякі розробники вже створюють навчальні ігри на основі жанру roguelike, які можуть бути корисними для учнів та студентів.

У цілому, жанр roguelike є жанром з великим потенціалом та можливостями для розвитку та експериментацій, і ми можемо очікувати більше захоплюючих інноваційних ідей в майбутньому.

Висновки до розділу 1

Було проведено аналіз існуючих розробок комп'ютерних ігор і надана загальна характеристика цього жанру розваг. Було розглянуто кілька конкретних

ігор, зокрема Hades, BPM, Curse of the Dead Gods та UnderMine, які відрізняються за своїм геймплеєм та технологіями.

Загалом, розглянуті рогалики представляють різноманітність та інноваційність у жанрі комп'ютерних ігор. Вони пропонують унікальні геймплейні механіки, захоплюючі сюжети та інтригуючі світи. Розробники комп'ютерних ігор продовжують експериментувати з технологіями та геймплеєм, що сприяє появі нових і захоплюючих розваг.

Рогалики мають потенціал розвиватися в майбутньому, пропонуючи нові горизонти геймплею та технологій. Покращення графіки, використання віртуальної та доповненої реальності, розширення світів і можливостей для інтерактивності – це лише кілька напрямків, які можуть бути відкриті для рогаликів у майбутньому.

В цілому, аналіз існуючих розробок комп'ютерних ігор свідчить про постійний розвиток та розширення можливостей цього жанру. Гравці можуть насолоджуватися різноманітністю геймплею, якісною візуальною реалізацією та захоплюючими сюжетами. Захоплюючі рогалики продовжують приваблювати увагу гравців і спонукають до постійного дослідження нових ігрових світів та технологій.

2 РОЗРОБКА ПРОЄКТУ

2.1 Аналіз середовищ розробки та обґрунтування вибору технології розробки проекту

Ігровий рушій – це спеціальне програмне забезпечення, яке розроблене з метою створення комп'ютерних ігор та інших інтерактивних програм, які можуть відтворювати графіку у режимі реального часу [13].

Для проведення аналізу були обрані кілька популярних ігрових рушіїв: Unreal Engine 4, CryEngine 5 та Unity 2019.

2.1.1 Unreal Engine 4

Ігровий рушій, що розробляється та підтримується компанію Epic Games. Першою грою на цьому рушії став шутер від першої особи Unreal у 1998 році. (рис. 2.1). Спочатку був призначений для розробки шутерів від першої особи, але його наступні версії успішно застосовувалися в іграх самих різних жанрів, в тому числі стелс-іграх, файтингах та і масових багатокористувацьких рольових онлайн-іграх [14].

Unreal Engine підтримує багато платформ та операційних систем, включаючи Windows, Linux, MacOS, iOS, Android, Xbox One, Xbox 360, PlayStation 2, PlayStation 3, PlayStation 4, GameCube, Wii, Wii U, Nintendo Switch, PSP та Dreamcast.

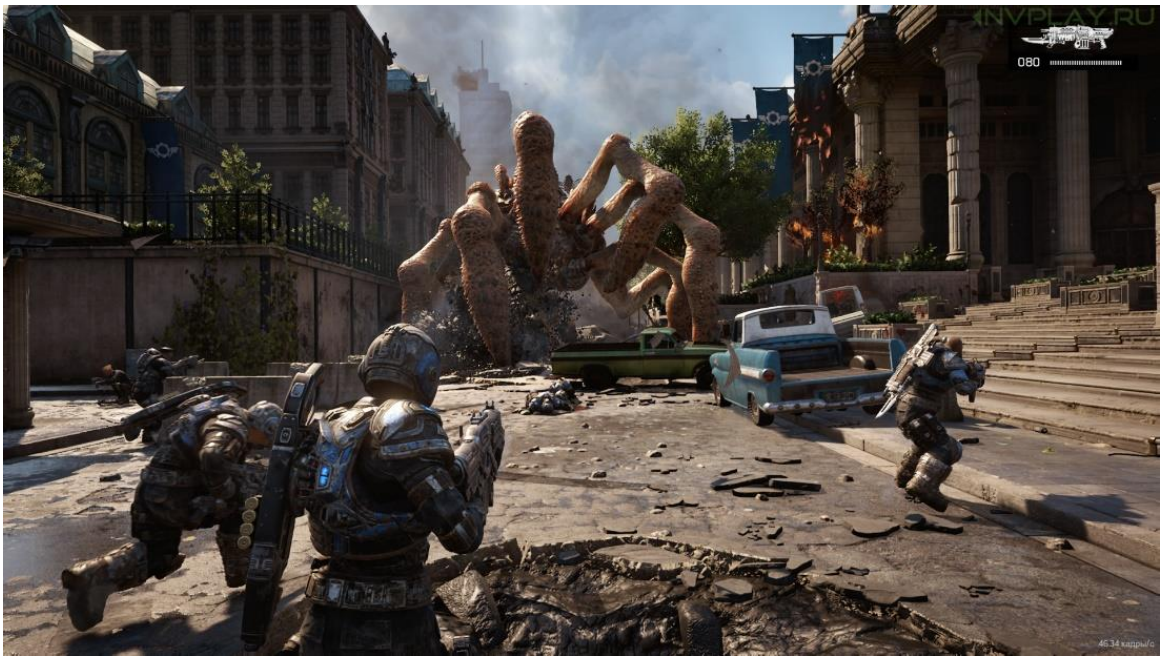

Рисунок 2.1 – Знімок екрана з гри на Unreal Engine 4

Для зручної роботи з редактором Unreal Engine рекомендовані мінімальні системні вимоги для персонального комп'ютера:

– операційна система: Windows 7 або новіше, або MacOS 10.14 або новіше, або Ubuntu 16.04 або новіше;

– процесор: Quad-core Intel або AMD, 2,5 ГГц або швидше;

– оперативна пам'ять: 8 ГБ;

– графічна карта: DirectX 11 або DirectX 12 сумісна графічна карта;

– вільне місце на жорсткому диску: 100 ГБ.

Рушій має зручний та простий інтерфейс, що вдосконалюється з кожною новою версією (рис 2.2).

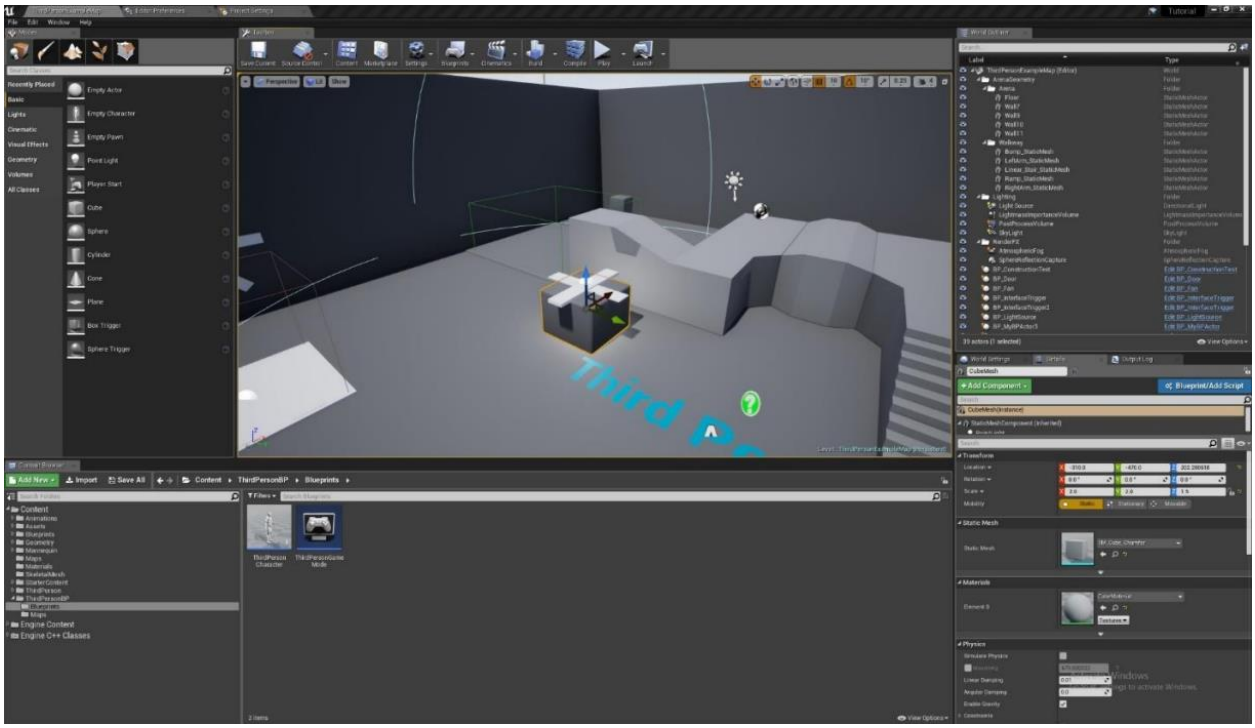

Рисунок 2.2 – Інтерфейс Unreal Engine 4

Завдяки розширеною конфігурації тіней, освітлення та рендерингу, цей двигун дозволяє досягти фотореалістичної графіки. Крім того, інструментарій двигуна значно полегшує роботу з ландшафтом та рослинністю.

Можна розробляти ігрову логіку без необхідності писати код завдяки системі візуальних скриптів Blueprint, але багато розробників все ж використовують мову програмування C++.

Для створення різних візуальних ефектів використовується вбудований редактор Cascade, який дозволяє налаштувати систему частинок з використанням різноманітних модулів. Також двигун має редактор матеріалів, який використовує штучне затінення та надає повний контроль над зовнішнім виглядом об'єктів та персонажів. Крім того, він має широкий спектр інструментів для редагування сітки та анімації. Отриманий результат можна переглянути та, при необхідності, редагувати в режимі реального часу.

Для створення відеороликів і анімаційних сцен використовується вбудований інструмент Sequencer, який дозволяє налаштовувати персонажів, камеру та освітлення.

Unreal Engine також підтримує віртуальну реальність і має розширені настройки для управління рухом. Розробник Epic Games співпрацює з провідними компаніями в галузі програмного і апаратного забезпечення, тому Unreal Engine відзначається високою якістю взаємодії з системами віртуальної та доповненої реальності. Рушій також має розширену систему штучного інтелекту, яка дозволяє створювати персонажів з реалістичною поведінкою в грі.

Офіційний веб-сайт рушія містить обширну документацію та навчальний матеріал для новачків у розробці. Також доступний магазин, де можна знайти різноманітні ресурси для розробки, такі як моделі персонажів та об'єктів, скрипти, звуки, анімації та додаткові плагіни для Unreal Engine.

Починаючи з 2015 року, Unreal Engine став доступним під вільною ліцензією, що дозволяє будь-кому використовувати його. Однак, якщо проект, розроблений на основі цього рушія, заробляє більше 3000 доларів на квартал, розробник повинен сплачувати 5% від доходу компанії Epic Games.

Unreal Engine найбільш підходить для глобальної розробки 3D-ігор, хоча він також має широкий вибір платформ для готових проектів та можливостей для розробки нових ігор.

2.1.2 Cry Engine 5

Остання версія гейм-движка, створена та випущена Crytek, відрізняється вражаючою візуальною якістю. Перша гра, що вийшла на цьому рушії, була відома Far Cry (рис. 2.3).

Рисунок 2.3 – Гра Far Cry 3

Хоча Cry Engine 5 претендує на роль кросплатформенного рушія, він підтримує лише чотири платформи: Windows, Xbox One, PlayStation 4 і Oculus Rift.

Для роботи з цим гейм-движком не потрібні дуже високі характеристики комп'ютера, адже він працює на відносно низькій системній навантаженості[15].

Але для ефективної роботи з Cry Engine 5 необхідні наступні характеристики комп'ютера:

– операційна система: Windows 7/8/10 (64-біт);

– процесор: Intel Core i5 або AMD FX-8350 (або краще);

– оперативна пам'ять: 8 ГБ RAM;

– відеокарта: NVIDIA GeForce GTX 660 або AMD Radeon HD 7850 (або краще);

– місце на жорсткому диску: принаймні 50 ГБ вільного простору;

– DirectX: версія 11;

– підтримка VR: Oculus Rift.

Cry Engine має простий і добре організований інтерфейс, що легко зрозуміти. Він має сильну фокус на візуальній якості проектів і надає розширений набір інструментів для досягнення вражаючого зображення (рис. 2.4).

Рушій також підтримує останню версію DirectX 12, яка відповідає за графічну складову гри. Одним з ключових елементів Cry Engine є фізично-оснований рендеринг (Physically Based Rendering), що дозволяє відтворити реалістичну взаємодію світла з матеріалами. Ця система використовує фізичні принципи, що надає об'єктам більш правдоподібний вигляд. Рушій також має динамічне відображення водних каустик, що забезпечує високу якість візуалізації води.

Cry Engine надає інструменти для ефективного згладжування та зниження пікселізації зображення. Також він має глобальну настройку освітлення, що дозволяє використовувати ширший діапазон світлотонів, ніж інші рушії. Це дозволяє покращити розпізнавання світлих, середніх і темних тонів без особливих зусиль.

Для редагування матеріалів Cry Engine має вбудований редактор під назвою Cry Engine Sandbox, який надає широкий спектр інструментів для створення рівнів та ігрових світів.

Один з цих інструментів Designer Tool – дозволяє редагувати моделі та експортувати створені матеріали в зовнішні редактори. Крім того, є редактор TrackView, який дозволяє створювати відеозаписи та інтерактивні сцени з об'єктами.

Подібно до Unreal Engine 4, у Cry Engine є система візуального скриптування під назвою FlowGraph. Цей інструмент дозволяє створювати та керувати логікою гри без необхідності писати код або скрипти вручну.

Cry Engine надає багато можливостей для анімації персонажів, включаючи параметричну кісткову анімацію. Він також має потужну систему штучного інтелекту, що дозволяє налаштовувати реалістичну поведінку персонажів.

Cry Engine надає можливість аналізувати продуктивність прямо під час гри, а вбудований інструмент під назвою Statoscope дозволяє графічно відстежувати використання ресурсів комп'ютера для подальшої оптимізації.

Так само, як і його попередник, Cry Engine має офіційний веб-сайт з доступними навчальними матеріалами і магазином, де можна знайти різноманітні ресурси для розробки.

Cry Engine має безкоштовну ліцензію і може бути використаний всіма без обмежень. Однак, якщо продукт, створений на основі Cry Engine, приносить дохід компанії або фізичній особі більше 5 тисяч доларів на рік, то 5% цього прибутку повинні бути сплачені компанії Crytek. Так само, як і Unreal Engine 4, Cry Engine є чудовим інструментом для розробки великих 3D-проектів.

2.1.3 Unity 2019

Unity – це ігровий рушій, створений компанією Unity Technologies, який може використовуватись для розробки ігор на різних платформах. Unity підтримує багато операційних систем, таких як Windows, Linux, MacOS, а також різні мобільні платформи, зокрема Android та iOS. Він також підтримує ігрові консолі, такі як PlayStation Vita, PlayStation 4, Xbox One, а також переносну консоль Nintendo Switch та Nintendo 3DS. Окрім того, Unity підтримує віртуальні реальність пристрої, такі як Oculus Rift, Steam VR, Gear VR, PlayStation VR, а також платформи для телевізорів, такі як Android TV, Smart TV та TvOS [16–17].

Для роботи з Unity необхідний комп'ютер, який відповідає мінімальним системним вимогам, які визначені для правильної роботи рушія:

– операційна система: Windows 7 SP1+, macOS 10.12+, Linux (Ubuntu 16.04, CentOS 7 та інші);

– процесор: Intel Core i5 або еквівалентний;

– оперативна пам'ять: 4 ГБ RAM;

– графічний процесор: з підтримкою Direct3D 11 або Metal, з підтримкою OpenGL ES 3.0;

– місце на жорсткому диску: 10 ГБ вільного простору;

– інтернет-підключення для завантаження додаткових ресурсів та оновлень;

– додаткові вимоги можуть бути залежними від конкретних функцій або платформ, з якими ви плануєте працювати.

Інтерфейс Unity призначений для зручного використання розробниками і має просту та зрозумілу структуру (рис. 2.6).

Unity є високофункціональною платформою розробки, яка надає широкий набір інструментів як для програмістів, так і для художників. Ці інструменти включають Timeline, який дозволяє створювати анімаційні сцени, Cinemachine для налаштування динамічних камер, Progressive Lightmapper для роботи з освітленням та підтримку Autodesk Maya для моделювання та 3D анімації. Unity також дозволяє розробляти проекти як у 2D, так і у 3D стилях. Рушій підтримує рушії NVIDIA PhysX для симуляції фізики об'єктів та Box2D для роботи з двовимірними об'єктами.

Unity славиться своєю високою оптимізацією, що безсумнівно впливає на якість і швидкість роботи проекту. Рушій підтримує різні мови програмування, зокрема C# та UnityScript, що дає розробникам велику гнучкість і можливість вибору найзручнішого інструменту для їх потреб.

Сайт Unity надає обширну документацію з різних напрямків, включаючи значну кількість навчальних матеріалів. Ці навчальні ресурси пропонуються у вигляді окремих уроків, які розглядають різні теми та аспекти розробки. Це дозволяє розробникам ефективно оволодіти різними аспектами Unity та отримати необхідні знання для реалізації своїх проектів.

У межах екосистеми Unity існує відомий магазин під назвою Asset Store, де розробники можуть знайти широкий вибір різноманітних матеріалів для своїх проектів. У цьому магазині доступні спрайти, моделі, скрипти, фонові зображення, а також різноманітні доповнення та розширення, що покращують функціональність

і можливості Unity. Asset Store надає зручний спосіб для розробників отримати якісні ресурси та інструменти, що допомагають їм прискорити і поліпшити процес розробки.

Unity пропонує три різні версії для придбання: Personal (безкоштовна), Plus (35$/місяць) та Pro (125$/місяць). Кожна версія має свої особливості та обмеження, але загалом вони надають доступ до унікального та різноманітного функціоналу для розробки різних типів продуктів. Різниця між версіями полягає в обсязі доступних можливостей та максимальному річному прибутку, який дозволений при використанні конкретної версії. Це дозволяє розробникам вибрати оптимальний план під свої потреби та бюджет.

Так як для реалізації нашого проекту головними критеріями вибору середи розробки є вільна ліцензія, навчальна документація, зручний та зрозумілий інтерфейс, низькі системні вимоги та інструменти для розробки 2D проектів, нашим вибором стане ігровий рушій Unity.

Для програмування в Unity використовується інтегроване середовище розробки (IDE) під назвою Microsoft Visual Studio. Це потужний редактор, який має можливість інтеграції з Unity і надає розробникам додатковий функціонал та зручні інструменти для роботи з кодом. За допомогою Microsoft Visual Studio розробники можуть ефективно писати та налагоджувати програмний код для своїх проектів в Unity.

2.1.4 Таблиця порівнянь різних рушіїв

Ця таблиця надає загальний огляд деяких основних характеристик трьох популярних двигунів для розробки ігор: Unreal Engine 4, CryEngine 5 та Unity 2019. Ці двигуни використовуються для створення ігор різного типу на різних платформах.

Таблиця 2.1 – Таблиця порівняння рушіїв

Характеристика	**Unreal Engine 4**	**CryEngine 5**	**Unity 2019**
Мова програмування	C++	C++,C#,Lua	C#,C++
Графічний движок	Вбудований	Вбудований	Вбудований
Фізичний движок	Вбудований (PhysX)	Вбудований (CryPhysics)	Вбудований (PhysX)
Нейромережа	Вбудований	Вбудований (CryAI)	Вбудований
Візуальні скрипти	Blueprints	FlowGraph	Візуальне скриптування (Visual Script, Playmaker та ін.)
Середовище розробки	Unreal Editor	Sandbox Editor	Unity Editor
Спільнота	Велика і активна	Середня	Велика і активна
Документація	Детальна і широка	Детальна і широка	Детальна і широка
Розширюваність	Висока	Висока	Висока

В результаті аналізу порівняння Unreal Engine 4, CryEngine 5 та Unity 2019, були виявлені наступні спільні характеристики:

– мова програмування: Всі три двигуни підтримують мови програмування C++ та C#. Unreal Engine 4 також підтримує мову програмування Lua;

– вбудовані графічний та фізичний двигуни: Unreal Engine 4 та CryEngine 5 мають вбудовані графічні та фізичні двигуни, які дозволяють створювати реалістичну графіку та моделювати фізичні ефекти. Unity 2019 також має вбудований графічний двигун;

– візуальне скриптування: Unreal Engine 4 використовує Blueprints, CryEngine 5 використовує FlowGraph, а Unity 2019 підтримує візуальне скриптування через різні інструменти, такі як Visual Script, Playmaker тощо;

– розширюваність: Всі три двигуни мають високу розширюваність і дозволяють розробникам розширювати функціонал за допомогою плагінів та налаштувань;

Узагальнюючи, Unreal Engine 4, CryEngine 5 та Unity 2019 мають деякі спільні характеристики, такі як мови програмування, підтримка різних платформ і вбудовані ресурси для створення графіки та фізики.

2.2 Загальний алгоритм реалізації проекту

Алгоритм розробки комп'ютерної гри не суттєво відрізняється від алгоритму розробки будь-якого іншого програмного продукту. Він також складається з трьох основних етапів:

– проектування;

– розробка;

– видання та підтримка.

На етапі проектування гри визначаються її мета і засоби розробки.

При визначенні мети гри вирішується, що саме буде приваблювати гравців і спонукати їх грати в створювану гру. Ця ідея є тісно пов'язаною з жанром гри. Наприклад, головна ідея рольових ігор (RPG) полягає в тому, щоб гравець міг пережити свою уявну роль, а головна ідея шутерів полягає в тому, щоб гравець міг взяти участь у реальних або фантастичних бойових діях. Тому, визначивши основну ідею гри, жанр майже автоматично вибирається.

Після визначення жанру і ідеї гри, наступним кроком є вибір сеттингу. Сеттинг визначає середовище, в якому відбуватиметься основна подія гри. Він визначає місце, час і умови дії. Вибір сеттингу може суттєво полегшити розробку

сценарію гри, тому його краще визначити заздалегідь, керуючись смаками цільової аудиторії.

Засоби розробки включають програмний код та ігровий рушій. Вибір засобів розробки визначає швидкість розробки та функціональність готового продукту. Програмний код залежить від платформи, на яку розробляється гра. Наприклад, для браузерних ігор можуть використовуватись мови програмування Java або Flash, а для комп'ютерних ігор оптимальним може бути використання мови програмування C#.

Ігровий рушій відповідає за відтворення фізики об'єктів, правил візуалізації графіки та інші технічні аспекти гри. При виборі ігрового рушія першочергово звертають увагу на його доступність та підтримувану мову програмування. Наприклад, ігровий рушій Unity дозволяє розробляти гри за допомогою мови програмування C# і має безкоштовну версію.

Після визначення мети гри та вибору засобів розробки настає другий етап – розробка.

Розробка є найбільшим і тривалим етапом реалізації проекту, який включає багато кроків, необхідних для створення функціонального продукту.

Спочатку необхідно визначитися з сюжетом та ігровою механікою. Ігрова механіка базується на цілях гри і визначає всі об'єкти та правила взаємодії гравця з ними. Зазвичай одночасно з розробкою ігрової механіки ведеться написання сюжету гри. Сюжет відіграє важливу роль і визначає, наскільки цікаво гравцеві буде грати у вашу гру. Сюжет може бути представлений у формі літературного сценарію, що описує основні події та персонажів гри, або режисерського сценарію, який надає детальний опис рівнів гри та подій, що відбуваються на цих рівнях.

Також на цьому етапі починається раннє розроблення графіки і дизайну гри. Засновуючись на сюжеті і попередньо обумовленому дизайну, створюються концепт-арті, що служать початковими ідеями для основного вигляду гри та персонажів.

Після визначення сюжету і ігрової механіки, найважливіша частина – розробка самої гри.

На основі сюжету та концепт-артів розпочинається створення персонажів та об'єктів гри, паралельно з цим розробляються ігрові рівні. При розробці ігрових рівнів спочатку створюється спрощений план, який схематично відображає сам рівень та включає предмети, з якими гравець буде взаємодіяти.

Після цього наступним кроком є створення першої версії рівня. Зазвичай це є проста локація з мінімальною кількістю необхідних предметів для проходження. Ця версія рівня використовується для тестування проходимості. Після проходження тесту рівень поступово заповнюється іншими об'єктами.

Незабаром після створення перших рівнів починається складання першого прототипу гри, який відомий як альфа-версія. Вона дозволяє розробникам провести тестування (альфа-тестування) основних механік гри і перевірити, наскільки вони відповідають поставленим вимогам. Часто в альфа-версіях гри об'єкти навіть не мають текстур або вони представлені у вигляді абстрактних об'єктів.

Якщо альфа-версія гри успішно проходить тестування, настає наступний етап розробки – доробка механіки та об'єктів гри.

На цьому етапі розробки гри проводиться доопрацювання рівнів та механіки, а також внесення перших сюжетних елементів, таких як відеоролики, сюжетні діалоги та кат-сцени. Також вирівнюються початкові помилки та дефекти у коді гри, які були виявлені під час тестування альфа-версії.

Після цього настає етап створення другого прототипу гри, який відомий як бета-версія. Бета-версія призначена для тестування гри на дефекти і проблеми. Фактично, бета-версія є практично готовою грою, і можуть відсутні лише незначні елементи, які не впливають на геймплей. Під час тестування бета-версії перевіряється все. Часто, особливо для мультиплеєрних ігор, до тестування запрошуються звичайні гравці, що дозволяє прискорити процес тестування і зняти частину навантаження з розробників.

Якщо гра успішно проходить бета-тестування, вона переходить до остаточного доопрацювання та виправлення критичних помилок. Потім відбувається збірка фінальної версії гри, і настає момент релізу.

Після релізу гри починається подальша підтримка продукту. Це включає випуск патчів для виправлення помилок у грі. Також для подовження життєвого циклу гри можуть бути випущені додаткові контент-пакети (DLC), які додають нові предмети або можливості для гравців.

Таким чином, зрозуміло, що етапи розробки комп'ютерних ігор мало відрізняються від етапів розробки будь-якого іншого програмного продукту.

2.3 Аналіз потенційної аудиторії споживачів

Один із важливих етапів у розробці будь-якого проекту – опис цільової аудиторії. Цей крок допомагає розробникам отримати краще розуміння своєї цільової аудиторії і враховувати їх потреби та уподобання під час розробки продукту. Розробка цільової аудиторії дозволяє зосередитися на деталях, які мають значення для цільової аудиторії, і враховувати їхні особливості під час процесу розробки. Це допомагає забезпечити, що продукт буде більш відповідати потребам та очікуванням цільової аудиторії.

Для цього проекту була визначена значна цільова аудиторія, яка охоплює людей у віці від 15 до 30 років. Вибір такої широкої категорії людей обумовлений кількома причинами. По-перше, ця вікова група відповідає середньому віку активних гравців комп'ютерних ігор, що стабільно зберігається на рівні від 30 до 40 років у країнах СНД. Таким чином, верхня межа цільової аудиторії була визначена з урахуванням цього фактора.

Ця гра спрямована на тих, хто хоче поглибитися у світ жанру roguelike і ознайомитися з його особливостями. Хоча гра не буде надто складною, вона все ж надає можливість зрозуміти основні механіки, які є характерними для цього жанру.

У цілому, цей проект може привернути увагу різноманітної аудиторії, яка зацікавлена у відкритті нових вражень у жанрі roguelike.

2.4 Актуальність проекту

У сучасних умовах ігрова індустрія швидко розвивається, збільшуючи кількість активних гравців і привертаючи нову аудиторію. Цей розвиток супроводжується з'явою нових студій і проектів, які приносять значний прибуток. Особливий розквіт спостерігається в інді-розробці, де незалежні розробники створюють цікаві і оригінальні проекти.

Можна з певністю стверджувати, що розроблюваний проект буде привабливим для цільової аудиторії. Жанр roguelike має значну популярність серед гравців, оскільки він пропонує унікальний і захоплюючий геймплей. Цей жанр характеризується випадково згенерованими рівнями, перманентною смертю персонажа та непередбачуваністю кожного проходження гри.

Завдяки своїй унікальності та складній природі, roguelike пропонує гравцям незабутні виклики та стимулює їх стратегічне мислення. Ігри цього жанру вимагають від гравців прийняття швидких рішень, адаптації до змінюючихся обставин та вміння використовувати обмежені ресурси ефективно.

Оскільки roguelike ігри зазвичай мають випадковий характер і високий рівень виклику, вони пропонують непередбачуваність та постійну мотивацію для гравців. Цей жанр дозволяє гравцям відкривати нові стратегії та досліджувати невідому територію, стимулюючи їхню творчість та ігровий розвиток.

Тому можна стверджувати, що розроблюваний проект у жанрі roguelike буде привабливим для аудиторії, оскільки він надасть гравцям незабутні виклики, випробує їхні стратегічні навички та забезпечить непередбачуваність та високий рівень геймплею.

2.5 Мета проекту

Мета проекту полягає у проведенні аналізу інструментів розробки комп'ютерних ігор з подальшим створенням комп'ютерної гри у жанрі roguelike. Головним завданням є розробка продукту, що використовує основні характеристики, що відрізняють жанр roguelike. Проект також спрямований на ознайомлення потенційної цільової аудиторії з цим захоплюючим, хоч і не найпоширенішим, жанром гри. Розробка гри дозволить отримати цінний досвід, який буде корисним не лише для майбутніх ігрових проектів, але й у різних інших сферах. Придбаний досвід розробки гри з використанням конкретного рушія також сприятиме прискоренню роботи над майбутніми проектами.

2.6 Функціонал проекту

Проект є комп'ютерною грою жанру roguelike, яка має на меті забезпечити розвагу та приємне проведення часу гравцем. Основні вимоги до проекту описані нижче.

1. Процедурно генеровані рівні: Гра повинна створювати різні рівні (локації) за допомогою алгоритму генерації, що дозволяє створювати нові, унікальні рівні кожного разу, коли гравець починає нову гру. Це забезпечує різноманітність та виклик у кожній грі.

2. Перманентна смерть: Головний аспект жанру roguelike полягає у тому, що гравець, в разі смерті персонажа, повинен починати гру спочатку з новим персонажем. Це додає виклику та підвищує геймплейну напруженість, оскільки кожне рішення гравця може мати наслідки.

3. Пошаговий геймплей: Гра повинна працювати за принципом ходів, де гравець виконує дії по черзі, а потім отримує відповідь від середовища гри. Це дозволяє гравцеві ретельно обмірковувати свої рухи та стратегію, а також реагувати на зміни у грі.

4. Випадковість: Елемент випадковості є важливим аспектом roguelike. Гра повинна містити елементи, які генеруються випадковим чином, такі як розташування ворогів, предметів, пасток тощо. Це забезпечує непередбачуваність гри та змушує гравця адаптуватися до нових ситуацій.

5. Поступове покращення: Гравець повинен мати можливість покращувати свого персонажа під час гри. Це може включати збирання предметів, отримання навичок, здобуття нових здібностей або підвищення рівня персонажа. Поступове покращення дозволяє гравцеві відчути прогрес та розвиток свого персонажа.

Крім основних вимог, до функціональної частини проекту також можуть бути додані інші елементи, такі як різні типи ворогів, боси, загадки, квести, система збереження гри, можливість гри в мультиплеєрному режимі тощо. Однак, виконання основних вимог є пріоритетом для створення відповідного roguelike-досвіду.

Висновки до розділу 2

У цьому розділі було детально досліджено різні середовища розробки ігор, зокрема Unreal Engine 4, Cry Engine 5 і Unity 2019. Проведено аналіз технічних характеристик цих ігрових рушіїв, їх інтерфейсу, наявності навчальної документації та доступу до роботи з ними. Після уважного розгляду було вирішено обрати Unity як ігровий рушій для розробки нашого прототипу. Було також проаналізовано середовище Microsoft Visual Studio, яке інтегрується з Unity і використовується для програмування проекту.

Після вивчення середовищ розробки був визначений основний алгоритм розробки комп'ютерної гри. Були описані ключові етапи створення будь-якого проекту, від написання сценарію до підтримки гри після випуску. Також був проведений аналіз цільової аудиторії нашого проекту, а також визначено актуальність та мету проекту. Завершенням розділу 2 є визначення основного функціоналу проекту, яким він має володіти після завершення розробки.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Графічне оформлення

Основною складовою частиною будь-якої комп’ютерної гри є її візуальна складова. Перед початком опису частини графічного оформлення, слід розібрати основні поняття, що використовуються у геймдизайні, такі як спрайт та тайл[18].

Тайл (з англ. Tile) – повторюваний фрагмент невеликих розмірів, який служить для створення зображень великих розмірів, а сам цей процес має назву Тайлова графіка. Методи тайлової графіки використовуються для створення рівнів у двовимірних та тривимірних іграх.

Спрайт (з англ. Sprite) – у двовимірних іграх є графічним зображенням якогось об’єкту, та може містити у собі декілька зображень, з яких можна зробити анімацію до об’єкту.

3.1.1 Спрайти ігрового героя

У якості персонажа у нашій гри буде використано спрайти падальника, який має окремі спрайти для створення анімації простою, отримання пошкоджень та атака яка буде також використана у грі для боротьби головного героя з перешкодами (рис. 3.2).

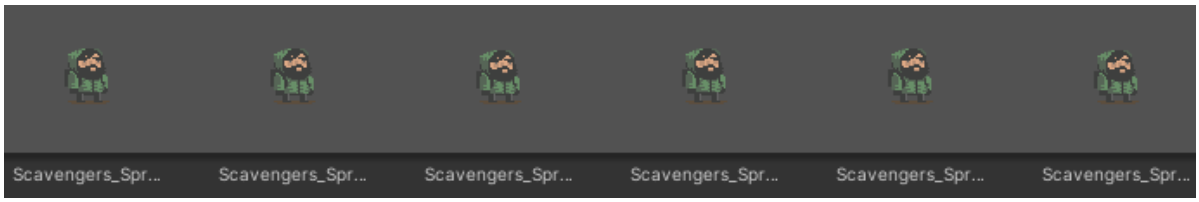

Рисунок 3.2 – Модель персонажа

3.1.2 Спрайти інтерфейсу

Після ігрового персонажа наступним кроком слід підібрати спрайти для фонового зображення, а також елементи візуальної складової гри. У якості фонового зображення було підібрано зображення, яке буде використано для фону сцени на який будуть накладатися інші візуальні елементи (рис. 3.3)

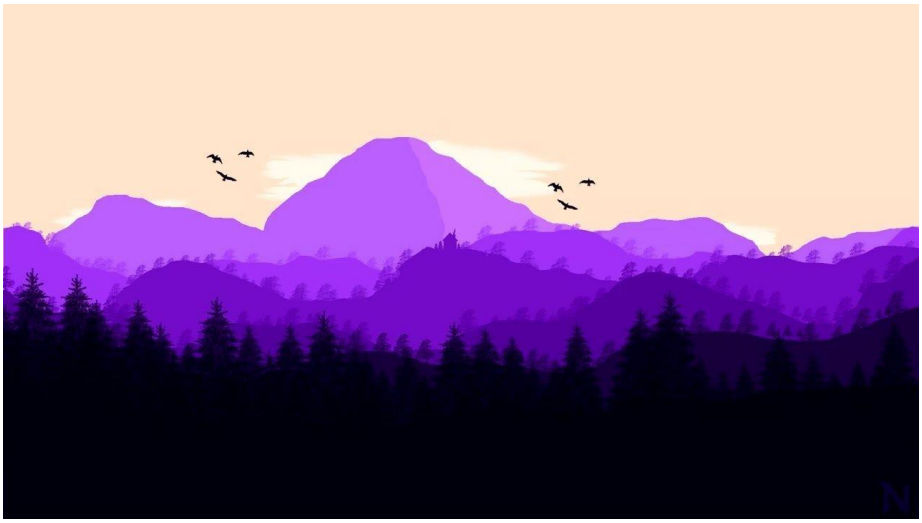

Рисунок 3.3 – Фон нового дня у грі

У грі були підібрані спрайти землі, перешкоджень та стін, які використовуються для візуального представлення об'єктів в грі.

Спрайти землі використовуються для відтворення поверхні, по якій персонаж може рухатися (рис. 3.4).

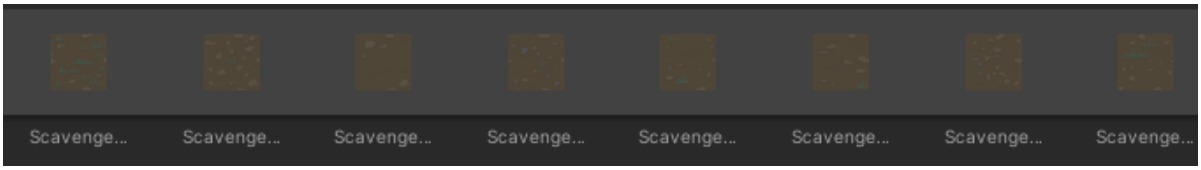

Рисунок 3.4 – Спрайти землі

Спрайти перешкоджень використовуються для створення об'єктів, які персонаж не може перетнути але може їх зламати (рис. 3.5).

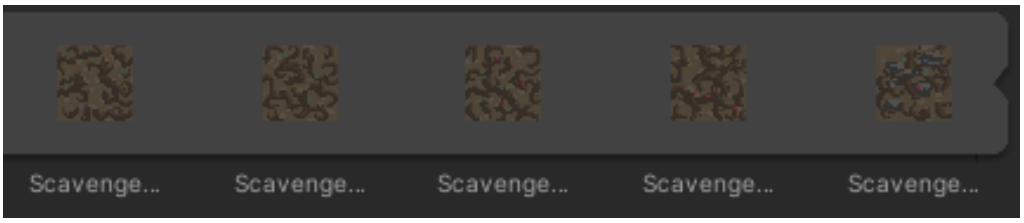

Рисунок 3.5 – Спрайти перешкоджень

Спрайти стін використовуються для обмеження меж гри. Вони представляють непрохідні об'єкти, які персонаж не може подолати (рис. 3.6).

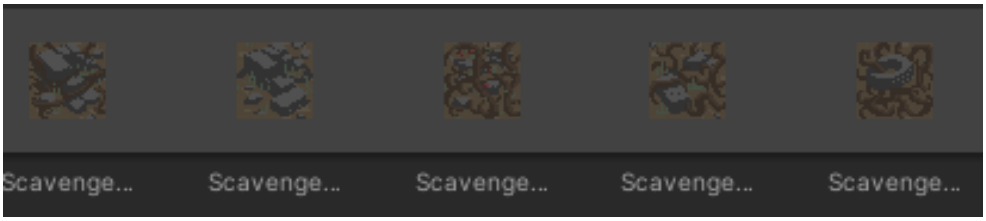

Рисунок 3.6 – Спрайти стін

3.1.3 Спрайт ворогів

Останнім елементом візуальної складової гри є спрайти ворогів, які будуть нападати на ігрового персонажа. Було обрано простий спрайт нежиті 2 видів. (рис. 3.7).

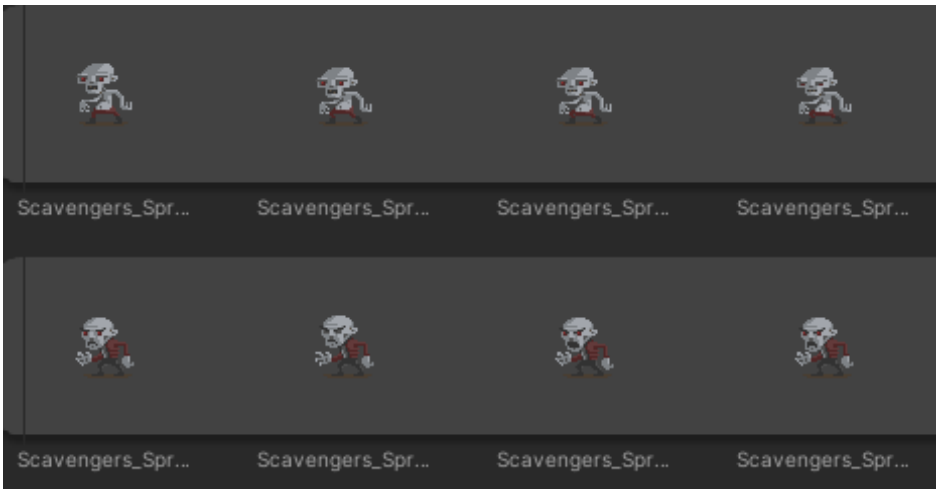

Рисунок 3.7 – Спрайти ворогів

3.2 Проектування гри

З візуальною складовою було вирішено, тому можна перейти до поетапного проектування елементів гри.

Перед початком програмування та написанням скриптів, потрібно вирішити проблему з накладанням візуальних елементів гри один на одного, для цього було використано функцію Tags & Layers, яка дозволяє призначити для кажного елементу свій шар, та регулювати який шар буде відображатись поверх іншого для запобігання виникнення візуальних проблем.

3.2.1 Фізичні властивості об’єктів

Перший крок у створенні прототипу гри полягає в наданні фізичних властивостей ігровим об'єктам. У випадку гри з видом зверху, ми починаємо з надання фізичних властивостей платформам, по яких буде рухатись персонаж. Це необхідно, оскільки без цього програма не буде враховувати платформи як фізичні об'єкти, і гравець буде нескінченно падати. Фізичні властивості об'єктів задаються за допомогою колайдерів – спеціальних компонентів, які визначають форму об'єкта для взаємодії з іншими об'єктами.

Тому, використовуючи функціонал Unity, спочатку ми надаємо кожній платформі компонент Box Collider 2D, який визначає фізичну форму платформи у вигляді прямокутника. Після того, як ми встановили фізичні властивості платформ, ми повинні зробити те ж саме для ігрового персонажа, але з деякими відмінностями. Оскільки платформи є нерухомими, а головний герой виконує різні дії, ми спочатку додаємо до нього компонент Capsule Collider 2D, який працює так само, як Box Collider 2D для платформ, але має форму капсули. Основним компонентом, який повинен бути у головного героя, є Rigidbody 2D, який надає можливість об'єкту рухатись у просторі, враховуючи фізичні закони. За допомогою

коду ми також можемо надавати різні властивості ігровому персонажу, що є ключовим для програмної частини нашого проекту.

3.2.2 Рух об’єктів

Наступним кроком розробки ігрового персонажа є написання скрипту для руху. Спершу задаємо змінні:

```
public float moveTime = 0.1f;
public LayerMask blockingLayer;
private BoxCollider2D boxCollider;
private Rigidbody2D rb2D;
private float inverseMoveTime;
private bool isMoving;
```

У цьому коді є поле rb2D типу Rigidbody2D, яке використовується для зберігання посилання на компонент Rigidbody2D, прикріплений до цього об'єкта.

Це поле rb2D використовується для виконання руху об'єкта у методі SmoothMovement. За допомогою методу MovePosition класу Rigidbody2D, рух об'єкта здійснюється до розрахованої позиції newPostion. Тому поле rb2D грає важливу роль у виконанні руху об'єкта.

Оскільки спершу ігровий об’єкт може рухатися на нескінченну кількість одиниць в просторі, розробник повинен встановити деякі правила. Швидкість руху персонажа визначається змінною moveTime, яка вказує час, який потрібно для пересування персонажа на одну одиницю в просторі. Ця змінна використовується для розрахунку швидкості переміщення об'єкта у методі SmoothMovement.

Воткнувши швидкість руху у метод SmoothMovement, код реалізує плавне переміщення об'єкта з поточної позиції до цільової позиції. Це досягається за допомогою корутину SmoothMovement, який виконується у циклі, зміщуючи об'єкт на деяку відстань у кожній ітерації. Цей процес триває, поки залишкова відстань до цільової позиції не стає дуже малою (float.Epsilon), і тоді корутин завершується.

Загальний алгоритм руху виглядає так:

1) Метод Move перевіряє, чи можна пересунути об'єкт у вказаному напрямку (xDir, yDir). Він перевіряє наявність перешкод за допомогою Physics2D.Linecast і повертає true, якщо рух успішний;

2) Якщо рух успішний, викликається корутин SmoothMovement, який починає плавне переміщення об'єкта до цільової позиції;

3) У корутині SmoothMovement об'єкт зміщується крок за кроком в напрямку цільової позиції, використовуючи Vector3.MoveTowards і Rigidbody2D.MovePosition;

4) Після досягнення цільової позиції об'єкт зупиняється, і рух закінчується.

3.2.3 Анімації для героя

Першим кроком у створенні анімації є повернення до функції Sprite Editor. Для створення анімацій використовуємо вбудовані функції Animation та Animator. (рис. 3.8).

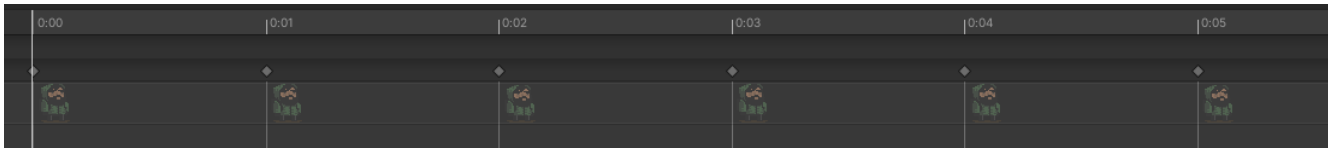

Рисунок 3.8 – Вікно Animation.

Наступним кроком буде встановлення зв’язків кожної створеної анімації між собою, щоб забезпечити переходи від анімації однієї дії зразу до іншої. (рис. 3.9).

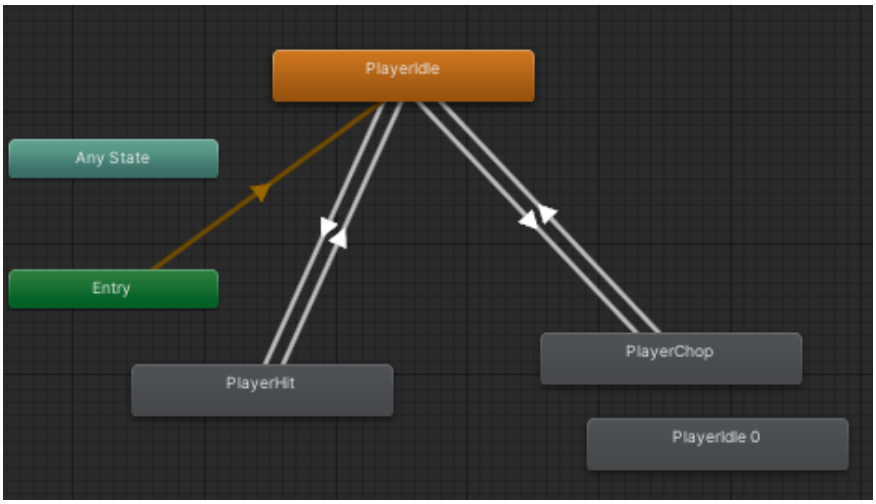

Рисунок 3.9 – Вікно зв’язків анімації.

3.2.4 Життя та гибель героя

Змінна "food" представляє кількість їжі, яку має персонаж. Вона ініціалізується значенням змінної "playerFoodPoints" з класу "GameManager" у методі "Start" за допомогою рядка "food = GameManager.instance.playerFoodPoints;". Це значення оновлюється при кожному зміщенні персонажа.

Коли персонаж зіткнеться зі стіною, викликається метод "OnCantMove", який зменшує "food" на величину "wallDamage" і відображає оновлене значення на екрані за допомогою рядка "foodText.text = "-" + loss + " Food: " + food:

```
protected override void OnCantMove<T>(T component)
{
    Wall hitWall = component as Wall;
    hitWall.DamageWall(wallDamage);
    animator.SetTrigger("playerChop");
}
```

Кожного разу, коли персонаж зіштовхується з об'єктом "Food" або "Soda", його кількість "food" збільшується на відповідну кількість балів (pointsPerFood або pointsPerSoda). Змінна "food" оновлюється, а нове значення відображається на екрані:

```
else if (other.tag == "Food")
{
    food += pointsPerFood;
    foodText.text = "+" + pointsPerFood + " Food: " + food;
    SoundManager.instance.RandomizeSfx(eatSound1, eatSound2);
    other.gameObject.SetActive(false);
}
else if (other.tag == "Soda")
{
    food += pointsPerSoda;
    foodText.text = "+" + pointsPerSoda + " Food: " + food;
    SoundManager.instance.RandomizeSfx(drinkSound1, drinkSound2);
    other.gameObject.SetActive(false);
}
```

Перевіряється, чи "food" менше або дорівнює нулю. Якщо це так, викликається метод "CheckIfGameOver", який відтворює звук гри, зупиняє фонову музику і викликає метод "GameOver" з класу "GameManager", що запускає відповідні дії для завершення гри:

```
private void CheckIfGameOver()
{
    if (food <= 0)
    {
        SoundManager.instance.PlaySingle(gameOverSound);
        SoundManager.instance.musicSource.Stop();
        GameManager.instance.GameOver();
    }
}
```

Ці рядки коду відповідають за життя персонажа та смерть персонажа в грі. Коли "food" досягає нуля або менше, гра закінчується і викликається метод "GameOver".

3.2.5 Інтелект ворогів

Інтелект ворогів реалізований в методі MoveEnemy(). В цьому методі ворог обчислює напрямок руху до гравця і викликає метод AttemptMove з параметром Player, щоб спробувати здійснити рух у напрямку гравця.

У разі, якщо ворог зіштовхується з гравцем під час спроби руху, викликається метод OnCantMove. У цьому методі ворог завдає шкоду гравцеві, викликаючи метод LoseFood гравця з параметром playerDamage, який визначає кількість очків їжі, яку гравець втратить. Також ворог виконує анімацію атаки і відтворює випадковий звуковий ефект зі списку attackSound1 та attackSound2 за допомогою SoundManager.instance.RandomizeSfx.

Таким чином, пошкодження від ворогів передається гравцю через метод LoseFood, а сама логіка руху ворогів і їх атаки реалізована в методах MoveEnemy та OnCantMove.

3.2.6 Графічний інтерфейс

Наступним етапом розробки буде налаштування першого графічного інтерфейсу. Спочатку розроблюємо смугу с життям для ворогів та для ігрового персонажа. (рис. 3.10) За допомогою функціоналу Unity створимо пустий графічний об’єкт, до якого вже буде написано функціонал, що буде працювати с графічним інтерфейсом.

Рисунок 3.10 – Графічний інтерфейс гри

Створюємо функцію SetHealth, яка буде відповідати за полоску життя ігрових об’єктів. Звертаючись до методу healthbarRect.localScale визначаємо значення змінної value, яка є показником поточного життя об’єкту, та за допомогою методу heathText.text виводимо текстову інформацію с приводу поточного життя на екран.

Копіюємо у вікні сцени створений скрипт до об’єкту гравця, та маємо такий саме результат графічном інтерфейсом (рис. 3.11).

Рисунок 3.11 – Їжа або здоров’я персонажа

3.2.7 Взаємодія між об’єктами

Реалізація графічного інтерфейсу (GUI) здійснюється за допомогою Unity UI системи. Основні елементи графічного інтерфейсу, які використовуються в коді, включають наступне:

1. Text: Використовується для відображення тексти у грі, такі як номер рівня ("Day 1") і повідомлення про кінець гри ("After X days, you starved.").

2. GameObject: Використовується для отримання посилань на різні об'єкти графічного інтерфейсу, такі як levelImage (зображення, що блокує гру під час налаштування рівня) і levelText (текст, що відображає номер рівня).

3. SetActive(bool): Метод, який використовується для включення або вимкнення GameObject, забезпечуючи показ або приховування елементів графічного інтерфейсу.

4. Invoke(string, float): Метод, який викликається з затримкою вказаною в секундах. Використовується для виклику функції HideLevelImage() після затримки levelStartDelay.

Ці елементи та методи використовуються для керування та відображення графічного інтерфейсу гри під час ініціалізації рівня, переходів між рівнями та кінця гри.

3.2.8 Генерація ворогів

Генерація ворогів здійснюється в методі SetupScene класу BoardManager.

Визначається кількість ворогів для даного рівня. У даному коді це реалізовано за допомогою наступного рядка:

```
int enemyCount = (int)Mathf.Log(level, 2f);
```

Ця лінія обчислює кількість ворогів на основі поточного рівня гри. Функція Mathf.Log використовується для обчислення логарифма числа level за основою 2. Результат приводиться до цілого числа за допомогою (int).

Потім викликається метод LayoutObjectAtRandom з аргументами enemyTiles, enemyCount і enemyCount. Цей метод розміщує ворогів на випадкових позиціях на дошці. Ось як він виглядає:

```
LayoutObjectAtRandom(enemyTiles, enemyCount, enemyCount);
```

Цей метод вибирає випадкові позиції зі списку доступних позицій (gridPositions) і розміщує на цих позиціях ворогів зі списку enemyTiles. Кількість ворогів обмежується значенням enemyCount.

3.2.9 Генерація рівня

Генерація рівня в даному коді відбувається випадково. В методі SetupScene викликаються різні функції для розміщення об'єктів (стін, їжі, ворогів тощо) на дошці. Ось як це відбувається:

1) Спочатку викликається функція BoardSetup, яка розміщує зовнішні стіни та фонову плитку дошки. Ця функція проходить по всіх клітинках дошки і випадковим чином вибирає плитку для кожної позиції;
2) Потім викликається функція InitialiseList, яка очищує список gridPositions і заповнює його всіма внутрішніми позиціями дошки. Цей список використовується для випадкового вибору позицій для розміщення об'єктів;
3) Далі викликаються функції LayoutObjectAtRandom для розміщення стін, їжі та ворогів. Ці функції випадковим чином вибирають позиції зі списку gridPositions і розміщують об'єкти на цих позиціях;
4) Нарешті, викликається Instantiate, щоб розмістити вихідний об'єкт на верхньому правому куті дошки.

Отже, розміщення стін, їжі, ворогів та вихідного об'єкта відбувається на випадкових позиціях на дошці. Кількість цих об'єктів також випадкова в межах заданих меж. Це призводить до випадково згенерованих рівнів у грі.

3.2.10 Умови для завершення гри

У даному коді умова для завершення гри вказана у функції GameOver(). Гра закінчується, коли гравець досягає 0 очків їжі (food points). При цьому виводиться

повідомлення з кількістю пройдених днів і повідомленням про поразку гравця. Після цього відображається чорний фон і гра вимикається.

Рисунок 3.12 – Кінець гри

3.3 Блок-схема роботи гри

Блок-схема описує процес гри зі створенням рівнів і управлінням їжею. Початок гри розпочинається зі створення початкових умов. Потім програма перевіряє, чи є рух у грі. Якщо є рух, гравець втрачає одиницю їжі. Після цього перевіряється, чи залишилась ще їжа. Якщо ні, гра закінчується. Якщо є, перевіряється, чи досягнуто нового рівня. Якщо так, створюється новий рівень і гра продовжується. Якщо ні, гравець продовжує рухатись в поточному рівні. Цей процес повторюється до досягнення кінцевого стану гри або до досягнення нового рівня.

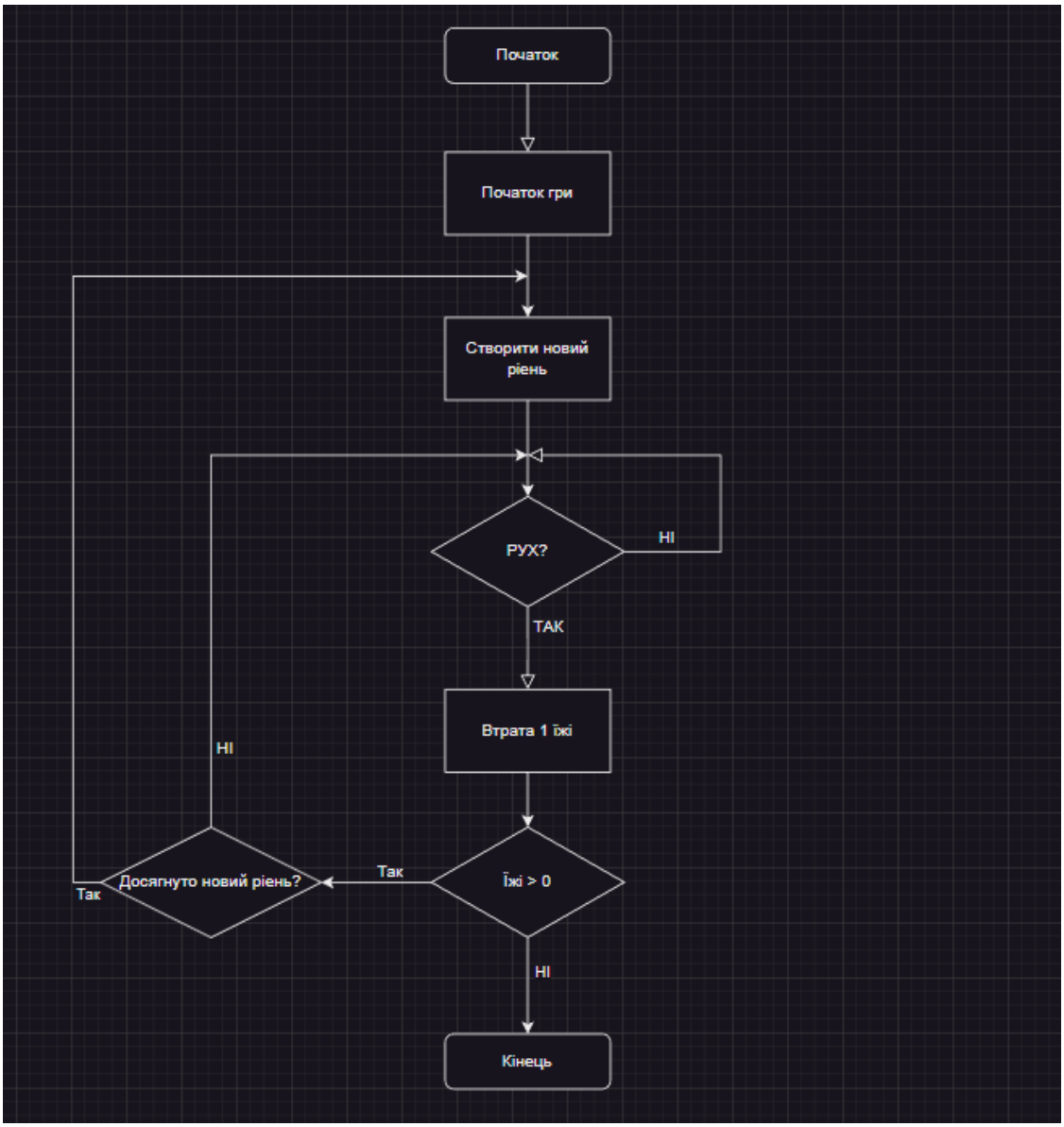

Рисунок 3.13 – блок-схема алгоритму гри

3.4 Діаграма варіантів використання

Діаграма використання (Use Case Diagram) – це інструмент моделювання, що використовується в UML, для опису функціональних вимог до системи з точки зору користувачів та зовнішніх акторів. Вона надає можливість уявити, як користувачі взаємодіють з системою та як система реагує на їх запити.

На діаграмі використання актори виступають у ролі різних осіб або систем, які мають взаємодіяти з системою, а використання (use case) представляють собою

дії або послідовності дій, які користувачі можуть виконувати з використанням системи для досягнення певної мети.

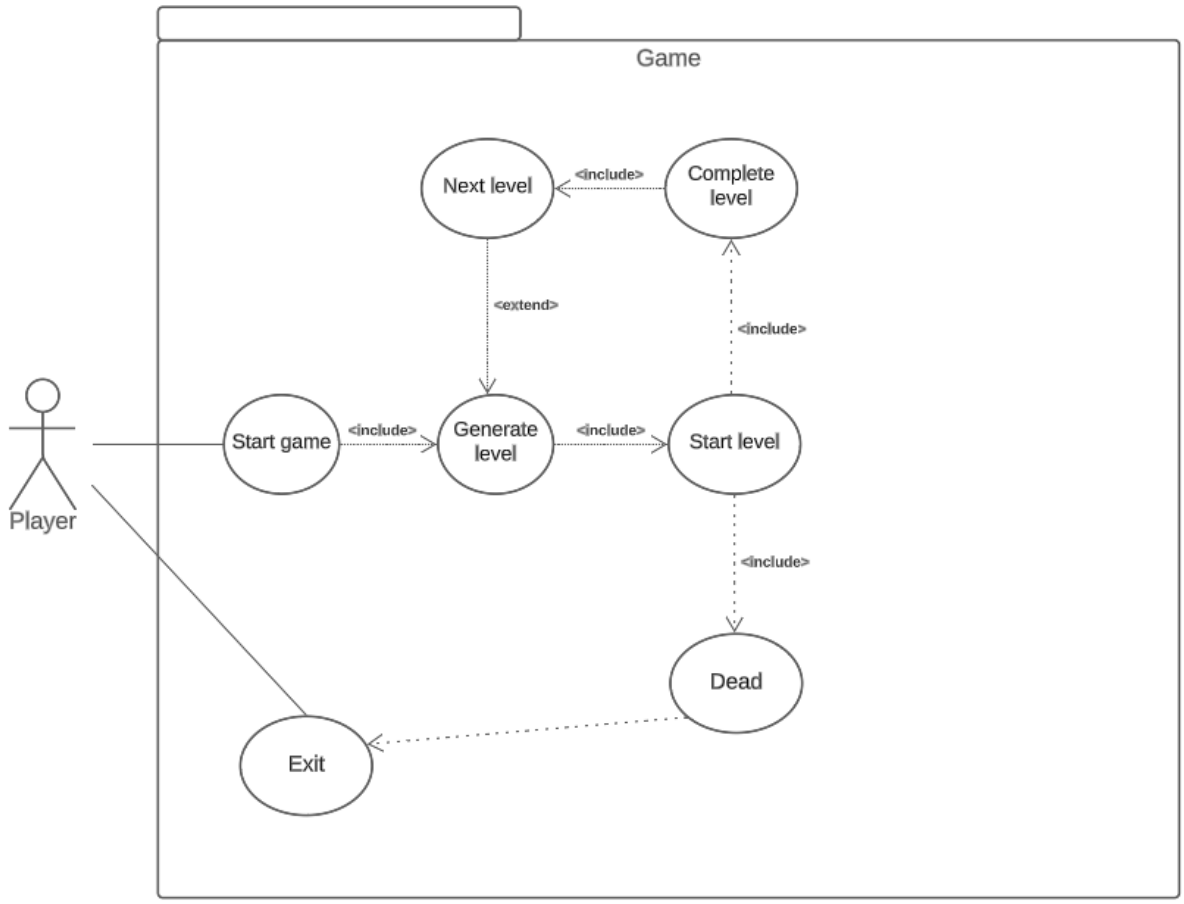

Рисунок 3.14 – Діаграма використання

Висновки до розділу 3

Даний розділ був цілком присвячений реалізації заданого темою проекту, який є 2D roguelike грою на Unity. Спершу було проаналізовано основні поняття графічної складової, такі як "тайл" та "спрайт", що використовуються в грі. Реалізація прототипу починалась з графічних елементів, таких як тайли та спрайти, які використовувалися у створенні грального світу. Описано основний функціонал, що використовувався для роботи з графікою.

Після аналізу функціоналу компонентів ігрового рушія, що надають об'єктам фізичні властивості, перейшли до написання коду. Починали з реалізації моделі героя гри, написання скриптів для його руху. Були описані основні класи, функції та методи, які використовувалися для надання життя герою. Також була представлена схема реалізації анімації головного героя під час руху та простою.

Було також докладно описано створення ворогів героя та написання штучного інтелекту для них. Визначені умови для смерті героя. Реалізовано простий графічний інтерфейс, який показує кількість днів та кількості життя. Були створені відповідні скрипти для реалізації цього інтерфейсу.

У наступному кроці реалізації було описано генерацію ворогів та рівнів у стилі roguelike. Були досліджені різні підходи до генерації ворожих персонажів і створення різноманітних типів ворогів з різними характеристиками. Написані скрипти для динамічної генерації рівнів, включаючи створення різних кімнат, коридорів та розташування ворогів у цих областях.

Також було реалізовано систему смерті гравця. Визначені умови, при яких гравець втрачає життя і гра завершується. Написані скрипти для обробки ситуації смерті гравця, включаючи анімації, звукові ефекти та перехід до екрану завершення гри.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Старт гри

Перед собою потенційний гравець вперше у грі побачить перший день своєї пригоди. У грі створено загальний фон та інтерфейс, які вражають своєю мінімалістичною естетикою та візуально втілені у стилі підземелля. Цей стиль використовується для створення атмосфери та відтворення враження глибини і таємничості підземних лабіринтів, які гравець буде досліджувати під час гри

Рисунок 4.1 – Перший день(рівень) гри

4.2 Інтерфейс та ігровий процес

Управління грою здійснюється за допомогою клавіш на клавіатурі. Гравець може використовувати клавіші W, A, S і D для керування своїм персонажем. Клавіша A відповідає за рух вліво, клавіша D – за рух вправо, клавіша W – за рух вверх, а клавіша S – за рух вниз. Це дозволяє гравцеві взаємодіяти з оточуючим

світом та пересуватися по ньому. При наближенні до перешкоди гравець може виконати удар за допомогою клавіш управління W, A, S, D.

Гравець також може бачити кількість його їжі або життів, що відображається на екрані. Усі ці елементи – персонаж, світ гри, вороги та ресурси – допомагають гравцеві зануритися у світ гри та створюють основу для захоплюючих пригод і викликів, які його чекають.

Рисунок 4.2 – Персонаж, навколишній світ, їжа, та кількість очок їжі

В даній грі, кожного разу, коли персонаж гравця рухається в будь-якому напрямку, його їжа зменшується на 1 одиницю. Це означає, що гравець споживає свою їжу під час переміщення. Їжа виступає як ресурс, який може використовуватися для різних цілей у грі, а також як показник життя персонажа.

Кожна одиниця їжі, яку гравець споживає, зменшує його запас їжі. Якщо запас їжі досягає нуля або стає меншим за нуль – гра закінчується.

Рисунок 4.3 – Зменшення їжі

Коли гравець знаходить їжу у грі, він може підійти до неї та підібрати її. Після підбору їжі, гравець отримує відновлення їжі у вигляді очків. Існують два види їжі, які можуть бути підібрані – ті, що відновлюють 10 очок їжі, і ті, що відновлюють 20 очків їжі.

Рисунок 4.4 – Поповнення їжі

Коли герой гравця успішно досягає вихідного пункту на поточному рівні гри, він переходить на новий рівень. Цей перехід на новий рівень супроводжується зміною складності гри та появою нових ворогів.

Рисунок 4.5 – Зустріч з ворогом

У грі, якщо персонаж гравця зіткнувся з неживим ворогом, він понесе втрату їжі. Втрата їжі є наслідком бойової ситуації та поразки в битві з ворогом. Втрата може бути різною залежно від типу ворога.

Існують два типи ворогів, з якими персонаж гравця може зіткнутися – вороги, які відбирають 10 очок їжі, і вороги, які відбирають 20 очок їжі. Ця втрата їжі впливає на рівень ресурсів персонажа гравця та може вплинути на його здатність до подальшого виживання в грі (рис 4.6).

Рисунок 4.6 – Поранення від ворога

4.3 Умови для завершення гри

У грі, однією з основних умов для завершення гри є смерть персонажа гравця. Це означає, що коли рівень їжі персонажа гравця досягає нуля, персонаж помирає і гра завершується.

Коли запас їжі персонажа гравця досягає нуля, це вказує на те, що гравець не зміг забезпечити персонажа достатньою кількістю ресурсів для виживання. Це може бути результатом недостатньої уваги до рівня їжі, поганих стратегічних рішень або непрофесійного управління ресурсами (рис 4.7).

Рисунок 4.7 – Смерть персонажа і кінець гри

Висновок до розділу 4

Даний розділ бакалаврського проекту присвячено інструкції користувача щодо роботи з прототипом гри. Було описано мінімалістичний інтерфейс та управління грою за допомогою клавіш на клавіатурі. Гравець може бачити рівень їжі та зіткнутися з ворогами, що впливають на ресурси персонажа та завершення гри при досягненні нульового рівня їжі.

У цьому розділі також зосереджено увагу на важливості стратегічного планування та управління ресурсами для успішного виживання в грі. Підбір їжі та уникнення ворогів стають ключовими елементами геймплею, вимагаючи уважного відстеження рівня ресурсів та мудрого використання зібраних ресурсів для досягнення успіху.

ВИСНОВКИ

Робота складається з чотирьох розділів, перший з яких містить загальні відомості про комп'ютерні ігри та їх жанрову класифікацію. Також було проведено аналіз основних переваг та недоліків існуючих сучасних прототипів.

Другий розділ пропонує опис вибраного жанру проекту, а також аналіз існуючих методів і функціоналу розробки ігор, а також обґрунтування вибору конкретного ігрового рушія для розробки. Також в розділі наведено повний алгоритм створення гри, проведено аналіз потенційної аудиторії гравців та загальної актуальності проекту, а також сформульовано мету розробки та функціонал гри, до якого вона має відповідати.

У третьому розділі детально описана повна поетапна реалізація проекту. Починаючи з визначення основних понять щодо візуальної складової проекту, надалі описано основні графічні компоненти та елементи, які були використані при розробці гри. Крім того, наведено докладний опис кожного етапу розробки, включаючи важливі методи та класи, які були використані під час написання коду. Окрім цього, розглянуто процес створення меню початку та завершення гри, з детальним описом його реалізації.

Завершальним результатом проектної роботи є робочий прототип гри у жанрі Roguelike, що не вимагає високих технічних характеристик. Прототип має просте меню, інтуїтивне управління та інтерфейс, чітку мету гри та умови для її завершення. Даний прототип гри може бути подальше доповнений новими деталями та елементами і офіційно випущений як повноцінна гра на основі певного інтернет-ресурсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп'ютерна гра. URL: https://esu.com.ua/article-4393 (дата звернення: 10.05.2023).

2. Melissinos C., Mika M., Broun E. Art of video games: from pac-man to mass effect. Rizzoli International Publications, Incorporated, 2015. 216 p.

3. Bissell T. Extra lives: why video games matter. New York : Pantheon Books, 2010. 218 p.

4. Соколовська Т. П. Електронні засоби навчання: позитивні й негативні фактори використання їх у навчанні // Пробл. сучас. підручника: Зб. наук. пр. К., 2010. Вип. 10.

5. История жанра roguelike: от rogue до binding of isaac. URL: https://habr.com/ru/articles/493890/ (дата звернення: 10.05.2023).

6. Hades. URL: https://www.supergiantgames.com/games/hades/ (дата звернення: 10.05.2023).

7. Bpm: bullets per minute. URL: https://www.bpmgame.com/ (дата звернення: 10.05.2023).

8. Curse of the dead gods. URL: https://dead-gods.com/ (дата звернення: 10.05.2023).

9. UnderMine. URL: https://undermine.game/ (дата звернення: 10.05.2023).

10. Milgard S. Stock market: playing the market game. Independently Published, 2019.

11. Rogue-like – tag stats. URL: https://steamspy.com/tag/Rogue-like (дата звернення: 20.05.2023).

12. Statista market forecast.URL: https://www.statista.com/outlook/dmo/app/games/role-playing-games/worldwide#revenue (дата звернення: 20.05.2023).

13. The 10 best video game engines. URL: https://www.gamedesigning.org/career/video-game-engines (дата звернення: 20.05.2023).

14. Unreal engine. URL: https://www.unrealengine.com/ (дата звернення: 20.05.2023).

15. Cryengine. URL: https://www.cryengine.com/ (дата звернення: 20.05.2023).

16. Unity. URL: https://unity.com (дата звернення: 20.05.2023).

17. Unity documentation.
URL: https://docs.unity3d.com/Manual/index.html (дата звернення: 20.05.2023).

18. Пояснення про спрайти, тайли.
URL: https://www.econdude.pw/2017/08/chto-takoe-graficheskij-sprajtsprite.html (дата звернення: 20.05.2023).

19. Roguelike tutorials. URL: http://rogueliketutorials.com/ (дата звернення: 20.05.2023).

20. R/roguelikedev. URL: https://www.reddit.com/r/roguelikedev/ (дата звернення: 20.05.2023).

21. Baron D. Game development patterns with unity 2021: explore practical game development using industry design patterns and best practices in unity and C#. Packt Publishing, Limited, 2021. 179 p.

22. Ferrone H. Learning C# by developing games with unity 2019: code in C# and build 3D games with unity, 4th edition. Packt Publishing, 2019. 342 p.

23. Halpern J. Developing 2D games with unity: independent game programming with C#. Apress, 2018. 408 p.

24. Hocking J. Unity in action: multiplatform game development in C# with unity 5. Manning Publications, 2015. 352 p.

25. Sung K., Smith G. Basic math for game development with unity 3D. Berkeley, CA : Apress, 2019. URL: https://doi.org/10.1007/978-1-4842-5443-1 (date of access: 26.05.2023).

26. Game developer.
URL: https://www.gamasutra.com/blogs/JoshGe/20160905/279995/Roguelike_Development_with_Unity_Introduction.php (дата звернення: 20.05.2023).

27. Thorn A. Mastering unity scripting. Packt Publishing, 2015. 380