МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри, канд. техн. наук,
доцент _____ Є. О. Давіденко
«___»___2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІГРОВИЙ ЗАСТОСУНОК В ЖАНРІ ROGUELIKE. РОЗРОБКА ІГРОВОГО СЕРЕДОВИЩА

Спеціальність «Інженерія програмного забезпечення»
121 – КРБ.1 – 409.21910918

Студент _____Ю. С. Полухін
«___»___2023 р

Керівник Phd, ст. викладач _____І. О. Кандиба
«___»___2023 р

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
«___»___2023 р

м. Миколаїв – 2023 рік

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

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Полухіну Юрію Сергійовчиу_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Ігровий застосунок в жанрі Roguelike. Розробка ігрового середовища_____

_____

____

Затверджена наказом по ЧНУ від «17» _березня____ 2023 р. № ____60____

2. Строк представлення кваліфікаційної роботи «____» _____ 2023 р.

3.Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є середовище для ігрового застосунку в жанрі Roguelike

_____

4. Перелік питань, що підлягають розробці:

- проведення аналізу аналогічних систем та дослідження предметної області;
- формування вимог до розроблюваного ігрового середовища;
- проєктування ігрового середовища, створення відповідних моделей;

- розробка ігрового середовища;
- тестування роботи розробленого ігрового середовища;
- аналіз зібраних даних щодо результатів розробки.

5. Перелік графічних матеріалів:

презентація._____

___

6. Завдання до спеціальної частини

_____

__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи PhD, ст. викладач Кандиба Ігор Олександрович

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Полухін Юрій Сергійович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання « 20 » __березня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Ігровий застосунок в жанрі Roguelike. Розробка ігровго середовища_____

__

	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	20.03.2023 р.	21.03.2023 р.	виконано
2.	Огляд літератури за темою роботи	23.03.2023 р.	27.03.2023 р.	виконано
3.	Складання календарного плану КРБ	28.03.2023 р.	29.03.2023 р.	виконано
4.	Аналіз предметної області	01.04.2023 р.	03.04.2023 р.	виконано
5.	Розробка проєктних рішень	05.04.2023 р.	07.04.2023 р.	виконано
6.	Моделювання та конструювання ПЗ	10.04.2023 р.	14.04.2023 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	15.04.2023 р.	31.05.2023 р.	виконано
8.	Розробка спеціальної частини з охорони праці	08.05..2023 р.	15.05.2023 р.	виконано
9.	Оформлення КРБ та презентації	22.05.2023 р.		виконано
10.	Відгук керівника КРБ			виконано
11.	Попередній захист			
12.	Завершення оформлення КРБ та презентації			
13.	Рецензування			
14.	Захист кваліфікаційної роботи			

Розробив студент ____Полухін Юрій Сергійович_____

(прізвище, ім'я, по батькові)

(підпис)

«__» _____ 2023 р.

Керівник роботи _PhD, ст. викладч_Кандиба Ігор Олександрович_____

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

«__» _____

2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок в жанрі Roguelike. Розробко ігровго середовища»

Студент 409 гр.: Полухін Юрій Сергійович

Керівник:Phd, ст. викладач Кандиба Ігор Олександрович

Ігри жанру Roguelike набули значної популярності серед геймерів завдяки своїй складній та непередбачуваній природі. Цей жанр продовжує приваблювати відданих шанувальників, створюючи попит на нові та інноваційні ігри в стилі Roguelike. Об’єктом кваліфікаційної роботи є процеси пов’язані із організацією та створенням ігрового застосунку. Unity залишається одним з провідних ігрових рушіїв в індустрії, відомим своєю доступністю, універсальністю та потужним набором функцій. Багато розробників обирають Unity як найкращий рушій для створення ігор у стилі Roguelike завдяки його широкому розповсюдженню, обширній документації та активній підтримці спільноти.

Об’єктом кваліфікаційної роботи є процес розробки ігрового середовища гри в жанрі Roguelike.

Для досягнення цієї мети було вирішено наступні завдання:

1. Проведено аналіз сучасних застосунків того ж жанру. Визначено їх сильні та слабкі сторони.
2. Складено специфікацію вимог до програмного забезпечення ігрового застосунку у жанрі Rogue Like.
3. Спроєктовано та змодельовано ігрове середовище для гри у жанрі Roguelike.
4. реалізовано ігрове середовище для гри в жанрі Roguelike:
   - 4.1. створено палітру для кольорового стилю гри;
   - 4.2. розроблено графічний стиль;
   - 4.3. створено ігрову графіку;
   - 4.4. створено групи звукових ефектів;

4.5. налаштовано ігрову камеру;

4.6. розроблено систему для багатокористувацької гри;

5. Проведено тестування та апробацію ігрового середовища.

У першому розділі КРБ проведено аналіз предметної сфери розробки комп’ютерних ігор

Другий розділ включає в себе моделювання складових частин ігрвого середовища.

Третій розділ включає в себе програмну реалізацію ігрового середовища.

Результатом КРБ є ігрове середовище для гри в жанрі Roguelike яке чудово комбінується з ігровими механіками, та створює приємне та динамічне зображення.

КРБ викладена на 71 сторінок, вона містить 4 розділи, 34 ілюстрацій, 17 таблиці, 6 джерел в переліку посилань.

Ключові слова: *Розробка ігор, рушій Unity, 2D-гра, ігрове середовище, графіка, звуковий ефект, розробка, візуалізація, оптимізація.*

ABSTRACT

of the Bachelor's Thesis

"Game Application in the Roguelike Genre: Development of a Gaming Environment"

Student of Group 409: Yuriy Sergiyovych Polukhin

Supervisor: PhD, Senior Lecturer I.O. Kandiba

Roguelike games have gained significant popularity among gamers due to their challenging and unpredictable nature. This genre continues to attract dedicated fans, creating a demand for new and innovative Roguelike-style games. The subject of this research work is the processes associated with the organization and creation of a game application. Unity remains one of the leading game engines in the industry, known for its accessibility, versatility, and powerful set of features. Many developers choose Unity as the best engine for creating Roguelike games, thanks to its widespread usage, extensive documentation, and active community support.

The research objective focuses on the processes related to the organization and creation of a game environment for Roguelike genre games.

To achieve this goal, the following tasks have been identified:

1. An analysis of contemporary applications within the same genre has been conducted, identifying their strengths and weaknesses.
2. The software requirements specification for the Roguelike game application has been created.
3. The game environment for the Roguelike genre has been designed and modeled.
4. The game environment for playing in the Roguelike genre has been implemented:
   - 4.1. A color palette has been created for the game's visual style.
   - 4.2. The graphic style has been developed.

4.3. The game graphics have been created.

4.4. Sound effect groups have been created.

4.5. The game camera has been configured.

4.6. A multiplayer system has been developed for the game.

5. The game environment has been tested and evaluated.

The first section of the research paper (KRB) includes an analysis of the subject area of computer game development.

The second section comprises the modeling of the components of the game environment.

The third section involves the software implementation of the game environment.

The result of the KRB is a game environment for Roguelike genre that integrates well with the gameplay mechanics and creates a pleasant and dynamic depiction.

The KRB consists of 71 pages, divided into four sections, with 34 illustrations, 17 tables, and six references in the bibliography.

Keywords: *Game development, Unity engine, 2D game, game environment, graphics, sound effects, development, visualization, optimization.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CCU	-	Concurrent User - це метрика, що вказує на кількість одночасних користувачів, які можуть підключатися до сервера або ігрового застосунку у певний момент часу.
RPC	-	Remote Procedure Call - це запит, що надсилається з клієнтського додатку на віддалений сервер для виконання певної функції або процедури.

ВСТУП

В сучасному світі комп'ютерні ігри займають важливе місце у нашому житті. Це розважальна форма, яка дозволяє нам втекти від повсякденних турбот і пережити безліч пригод. Завдяки розвитку технологій, графіки та можливості гри значно покращилися, що дозволило розширити аудиторію і ввести ігри в нові галузі наукових досліджень та освітніх програм.

Ринок ігрової індустрії постійно зростає, пропонуючи гравцям все нові ігри з різноманітними жанрами та сценаріями. Ігри в жанрі Roguelike є однією з цих галузей, які набувають все більшої популярності. Ці ігри характеризуються генерацією рівнів, перманентними смертями героїв та випадковістю елементів гри. Вони створюють унікальний досвід, який різний кожного разу, коли гравець починає нову гру.

У зв'язку з цим, розробка ігрових додатків у жанрі Roguelike на базі рушія Unity має великий потенціал, оскільки може зацікавити багатьох гравців та дозволити їм отримати унікальний досвід гри. У даному звіті буде розглянуто процес розробки ігрового середовища у жанрі Roguelike на рушії Unity, включаючи створення механіки гри, генерацію рівнів та персонажів, а також тестування та оптимізацію.

Існує багато ігрових рушіїв, таких як Unreal Engine, CryEngine, GameMaker та інші. Кожен з них має свої переваги та недоліки і придатний для розробки ігор в різних жанрах та форматах.

Актуальність Unity на сьогодні полягає в тому, що цей рушій забезпечує розробникам швидкий та ефективний процес створення ігор в різних жанрах та форматах. Він також дозволяє розробникам створювати ігри для різних платформ, що дозволяє розробникам максимально використовувати потенціал ринку комп'ютерних ігор.

Окрім того, Unity надає можливість створювати не тільки ігри, але й

різноманітні додатки, такі як симулятори, медичні програми, програми для візуалізації даних та інші, що розширює область використання рушія.

Отже, актуальність рушія Unity полягає в його потужній функціональності, надійності, широкому спектрі можливостей для створення ігор та додатків

Об'єктом кваліфікаційної роботи є процес розробки ігрового середовища гри в жанрі Roguelike.

Предмет кваліфікаційної роботи є інструментальні засоби та інформаційні технології розробки ігрового середовища для гри в жанрі Roguelike.

Метою кваліфікаційної роботи є популяризація ігрового жанру «Roguelike», шляхом розробки застосунку на базі Unity.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Провести аналіз сучасних застосунків того ж жанру. Визначити їх сильні та слабкі сторони.
2. Скласти специфікацію вимог до програмного забезпечення ігрового застосунку у жанрі Rogue Like.
3. Запроєктувати та змоделювати середовище гри у жанрі Roguelike.
4. Розробити функціональні модулі ПЗ та елементи ігрового середовища, які дозволять користувачу швидко зануритись у ігровий процес та отримувати від нього задоволення:
   - 4.1. створити палітру кольорового стилю гри;
   - 4.2. розробити графічний стиль;
   - 4.3. створити ігрову графіку;
   - 4.4. створити групи звукових ефектів;
   - 4.5. налаштувати ігрову камеру;
   - 4.6. розробити систему для багатокористувацької гри.
5. Протестувати ігрове середовище.

1 АНАЛІЗ ІГРОВИХ ЗАСТОСУНКІВ В ЖАНРІ ROGUELIKE

1.1 Опис галузі розробки комп’ютерних ігор

Галузь розробки комп'ютерних ігор - це галузь індустрії розваг, що охоплює процес створення відеоігор для різних платформ, таких як ПК, консолі, мобільні телефони та інші.

Розробка комп'ютерних ігор охоплює різноманітні етапи, такі як планування, проєктування, програмування, тестування, оптимізацію та випуск гри. Кожен етап є дуже важливим для успішної розробки гри, і зазвичай залежить від здібностей та професійних навичок розробників.

Серед основних елементів, що входять до предметної сфери розробки комп'ютерних ігор, можна виділити наступні:

Дизайн ігри: це охоплює розробку концепції гри, вибір жанру, створення історії, персонажів, геймплею, а також дизайн артоб'єктів та інтерфейсу.

Програмування: розробка програмного забезпечення для створення логіки ігри, створення графіки, звуків, музики, ефектів та інших елементів гри.

Графіка - це охоплює створення 2D та 3D-графіки, текстур, анімації, освітлення, ефектів та інших візуальних елементів.

Аудіо: створення звукових ефектів, музики, голосових акторів та інших аудіоелементів, що використовуються в грі.

Тестування: проведення тестування гри на наявність помилок, багів, виявлення проблем зі сумісністю, а також оцінка рівня складності гри.

У цій галузі працюють різноманітні фахівці, такі як програмісти, гейм-дизайнери, художники, аудіоінженери, тестувальники, продюсери та інші. Кожен з них має важливу роль у створенні успішної гри.

Також в галузі розробки ігор існує декілька жанрів, серед яких можна виділити Action, шутери, стратегії, рольові ігри, симулятори, спортивні ігри тощо. Кожен жанр має свої особливості та вимоги до розробки.

Одним з ключових аспектів розробки комп'ютерних ігор є їхнє комерційне використання. Розробники та видавці постійно працюють над тим, щоб зробити свої ігри успішними на ринку, збільшити прибуток та отримати популярність серед гравців.

У сучасному світі комп'ютерні ігри є важливим аспектом розваг та культури. Вони дозволяють гравцям перенестися у віртуальний світ, взаємодіяти з іншими гравцями та випробувати свої навички та стратегічне мислення. У цьому розумінні галузь розробки комп'ютерних ігор є важливою складовою сучасної культури та розваг.

Також важливим аспектом розробки комп'ютерних ігор є технічна складова. Розробники ігор повинні мати розуміння різних технологій та програмних засобів, що використовуються для створення ігор. До цих засобів можна віднести різноманітні ігрові движки, мови програмування, графічні та аудіоредактори та інші програмні засоби.

При розробці комп'ютерних ігор також важливо враховувати потреби та бажання гравців. Розробники повинні аналізувати та вивчати різні тренди, збирати та аналізувати дані щодо гри та поведінки гравців, щоб зрозуміти, що саме приваблює гравців у грі та як можна поліпшити їхнє враження від неї.

У сучасному світі комп'ютерні ігри мають великий потенціал не лише як засіб розваг та відпочинку, але як засіб навчання та розвитку. Ігрові технології можуть використовуватись у різних галузях, таких як освіта, медицина, бізнес. В цьому контексті розробка комп'ютерних ігор має великий потенціал для розвитку та впровадження нових технологій та ідей.

1.2 Аналіз жанру Roguelike

RogueLike - це жанр комп'ютерних ігор, який має своє коріння в ігрі Rogue, яка була створена у 1980-х роках. Основна ідея Rogue та інших ігор жанру полягає в тому, що гравець керує персонажем, який знаходиться у випадково згенерованому підземеллі. Гравець повинен досліджувати підземелля, збирати різні предмети та зброю, боротися з монстрами та іншими небезпеками, та шукати вихід з підземелля.

Жанр RogueLike отримав свою назву від гри Rogue та характеризується рядом особливостей. Одна з найважливіших особливостей - це процедурна генерація рівнів, тобто кожне нове підземелля генерується випадковим чином, що дозволяє гравцеві досліджувати нові території та забезпечує велику переіграбельність гри. Також, відмінністю RogueLike від інших жанрів є постійний рівень складності та реалістичність, що означає, що гравець може втратити всі свої досягнення та зміститися на початок гри при невдалій спробі.

Іншою важливою особливістю RogueLike є система перманентності, яка означає, що гравець повинен бути дуже уважним та сконцентрованим на своїх діях, оскільки кожна помилка може коштувати йому життя персонажа. Втім, навіть після смерті персонажа, гравець може продовжити гру з новим персонажем, зберігаючи деякі досягнення та знання про підземелля.

Геймдизайн гри в жанрі RogueLike - це процес створення гри, який охоплює розробку геймплею, балансування гри та налаштування системи процедурної генерації рівнів.

Починаючи з геймплею, головною метою геймдизайнера є створення гри, яка буде цікавою та привабливою для гравця. У жанрі RogueLike це часто досягається шляхом створення системи бойових механік, яка буде складною та реалістичною. Також важливою частиною геймплею є збір різних предметів та

зброї, що дозволяє гравцеві покращувати свої навички та підвищувати свій рівень.

Наступним етапом є балансування гри. Це означає, що геймдизайнер повинен визначити, які монстри будуть зустрічатися гравцю, яка буде їхня сила та складність, а також налаштувати ігрові параметри, які впливають на складність гри. Наприклад, геймдизайнер може змінювати частоту з'явлення певних предметів та зброї, щоб збалансувати гру та зробити її цікавішою.

Найважливішою частиною геймдизайну у жанрі RogueLike є процедурна генерація рівнів. Геймдизайнер повинен створити систему, яка може створювати нові рівні випадковим чином, з урахуванням різних параметрів, таких як рівень складності, кількість ворогів та типів перешкод. Таким чином, кожна гра буде унікальною та відрізнятиметься від попередньої.

У процесі геймдизайну у жанрі RogueLike важливо пам'ятати, що гравці часто шукають викликів та складнощів, тому гра повинна бути достатньо складною, але при цьому не занадто непростою, щоб зберегти інтерес гравців. Також важливо враховувати фідбек від гравців та вдосконалювати гру, щоб зробити її більш захопливою та динамічною.

Окрім цього, геймдизайнер повинен забезпечувати наявність різних варіантів прогресування гравця в грі, що дає можливість гравцеві зростати та розвиватися. Наприклад, гравець може отримувати досвід, збирати предмети та зброю, які допомагають покращити його характеристики, або розблоковувати нові рівні та секрети.

Також важливою частиною геймдизайну є створення цікавих історій та персонажів, які додають грі додаткового настрою та інтриги. Наприклад, гравець може зустрічати різних персонажів, які допомагають або заважають йому у проходженні гри.

В цілому, геймдизайн у жанрі RogueLike - це складний та багатогранний процес, який потребує від геймдизайнера розуміння потреб та бажань гравців, а також вміння створювати захоплюючу та унікальну гру з непередбачуваним геймплеєм та великою кількістю можливостей для розвитку та прогресування гравця.

1.3 Аналіз ігрових застосунків у жанрі Roguelike

Назва: Nuclear Throne.

Розробник: розроблена незалежною гральною студією Vlambeer і випущена у 2015 році.

Мова реалізації: C++.

Ігровий рушій: GameMaker: Studio.

Платформи: Linux, Windows, OS X, PS4, Vita, Switch, Xbox One.

Переваги:

1. Велика різноманітність зброї, ворогів, пасивних бонусів;
2. динамічне керування та бойова система;
3. цікаві боси та ігровий всесвіт.

Недоліки:

1. Через свою особливу випадкову генерацію рівні виглядають непривабливо;
2. відсутність кооперативних режимів.

Рисунок 1.1 - Інтерфейс гри Nuclear Throne

Назва: Enter the Gungeon.

Розробник: Dodge Roll.

Мова програмування: C#.

Ігровий рушій: Unity.

Платформи: Microsoft Windows, macOS, Linux, PlayStation 4, Nintendo Switch, Xbox One, Amazon Luna.

Переваги:

1. Цікавий геймплей з нелінійною структурою та великою кількістю зброї і предметів;
2. Відмінна музика та анімація;
3. Висока реграбельність;
4. Багато елементів гри залежать від випадковості, що додає різноманітності геймплею.

Недоліки:

5. Висока складність гри, яка може відштовхнути деяких гравців;
6. Відсутність кооперативних режимів;

7. Деякі зброї та предмети можуть бути не ефективними або нецікавими для гравців.

Рисунок 1.2 - Інтерфейс гри Enter the Gungeon

Назва: Dead Cells.

Розробник: Motion Twin.

Мова програмування: OCaml.

Ігровий рушій: Custom Engine.

Платформи: Microsoft Windows, macOS, Linux, PlayStation 4, Xbox One, Nintendo Switch, iOS, Android.

Переваги:

1. Чудовий геймплей, що поєднує елементи рогалика та метроіда;
2. Красива графіка та розробка анімації;
3. Багато видів зброї та екіпірування з унікальними особливостями;
4. Висока реграбельність завдяки генерації рівнів.

Недоліки:

1. Висока складність гри, що може відштовхувати деяких гравців;
2. Іноді відбуваються деякі затримки у завантаженні рівнів на Nintendo Switch;
3. Без можливості паузи гри в режимі гри на Nintendo Switch.

Рисунок 1.3 - Інтерфейс гри Enter the Gungeon

Назва: Skul: The Hero Slayer.

Розробник: SouthPAW Games.

Мова програмування: C#.

Ігровий рушій: Unity.

Платформи: Microsoft Windows, Nintendo Switch, PlayStation 4, PlayStation 5, Xbox One, Xbox Series X/S.

Переваги:

1. Цікавий геймплей з нелінійною структурою та великою кількістю персонажів та зброї;
2. Гарна графіка та анімація, що роблять гру візуально привабливою;
3. Різноманітність геймплею завдяки можливості переключення між різними формами Skul;
4. Висока реграбельність завдяки генерації рівнів та різноманітністю персонажів.

Недоліки:

1. Гра може бути занадто складною для деяких гравців;
2. Деякі аспекти гри, наприклад, управління персонажем, можуть бути незручними для деяких гравців;

3. Незначна кількість багів та проблем з оптимізацією на деяких платформах.

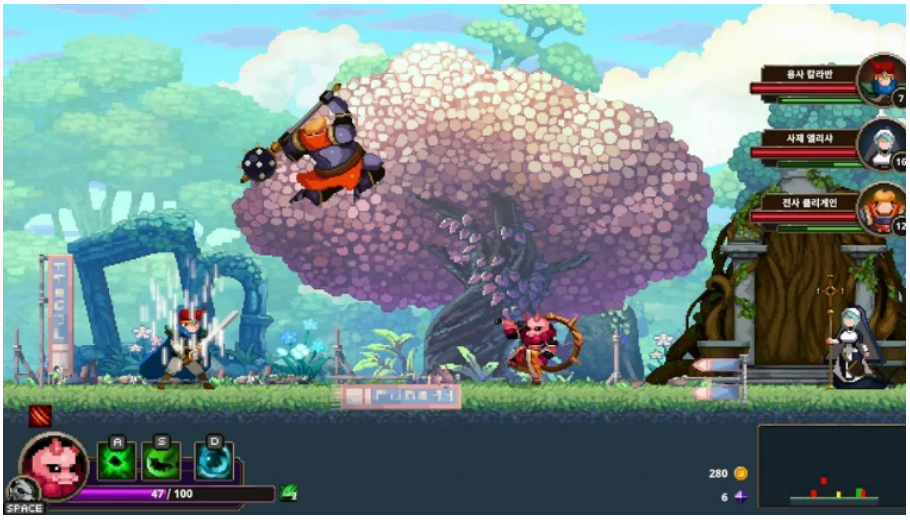

Рисунок 1.4 - Інтерфейс гри Enter the Gungeon

1.4 Специфікація вимог до програмного забезпечення

Призначення застосунку:

Ігровий застосунок призначений для надання гравцям можливості зануритися в світ, де кожен геймплей унікальний та непередбачуваний. Гравці відчувають себе у ролі персонажа, який починає гру зі зброєю та ресурсами на початку кожного рівня, та має боротися з різноманітними ворогами, які можуть з'являтися випадковим чином. Основна ідея цього полягає в тому, що кожна гра є унікальною та непередбачуваною завдяки процедурно згенерованому світу, ворогам, предметам та ситуаціям. Гравець може зіткнутися зі зброєю, яку не використовував раніше, зі створінням, яке він не знає, або зі зміною характеристик свого героя після зустрічі з особливими предметами. Гравці повинні добре продумувати кожен свій крок та вирішувати проблеми на ходу, оскільки помилки можуть коштувати їм життя та відправити на початок гри. Загалом, ігровий застосунок в жанрі Rouguelike призначений для того, щоб

надати гравцям унікального досвіду гри, де кожен раунд буде відрізнятися від попереднього. Це гра, яка пропонує виклики та можливості для розвитку навичок та стратегічного мислення, а також принесе багато задоволення люб

Опис структури системи:

Структура системи спроектованого ігрового застосунку включає наступні компоненти:

1. Інтерфейс користувача: Це інтерфейс системи, який включає в себе різні меню, кнопки, діалогові вікна, шкали характеристик персонажу та зброї.
2. Система керування: Це компонент структури системи, який визначає те яким чином користувач буде впливати, керувати та взаємодіяти з ігровим середовищем.
3. Система багатокористувацької гри: Це компонент структури системи, що дозволяє декільком користувачам з різних точок світу грати разом в одній ігровій сесії.
4. Система створення та пошуку ігрових кімнат: Це компонент структури системи, що дозволяє створювати ігрові кімнати з особливими налаштуваннями чи шукати їх.
5. Система генерації ігрового поля: Це компонент структури системи, що відповідає за генерації ігрових ігрових рівнів, предметів та ворогів.
6. Система зміни зовнішнього вигляду ігрового персонажу: Це компонент структури системи, що дозволяє налаштовувати та зберігати зовнішній вигляд ігрового персонажа під бажання користувача.
7. Системи налаштувань: Це компонент структури системи, що дозволяє налаштовувати глобальні частини ігрового застосунку такі як: клавіші керування, гучність звукових ефектів, вікно ігрового застосунку.

Засоби апаратної та програмної реалізації:

Апаратні та програмні засоби реалізації ігрового застосунку залежать від конкретних вимог проекту, але деякі популярні інструменти будуть використані:

1. Ігровий рушій Unity: один з найпопулярніших ігрових рушіїв, що пропонує велику кількість інструментів та компонентів для створення ігрових застосунків у будь якому жанрі.
2. Photon Cloud: один з найпопулярніших сервісів для створення багатокористувацьких застосунків, що надає сервер та компоненти для інтегрування в середовище Unity.
3. WavePad: безкоштовна програма для створення та редагування звуків формату wav.
4. Visual Studio: середовище для написання та компіляції скриптів на мові C#.
5. Fork: програмне забезпечення для роботи з системою контролю версій git.
6. Photoshop CS6: програмне забезпечення для створення та обробки 2D зображень.

Призначення та межі проєкту:

- призначення системи (застосунку), для якої розробляється програмне забезпечення: програмне забезпечення розробляється для ігрового застосунку в жанрі Roguelike.
- межі проекту ПЗ: межами програмного проекту є створення динамічного, конкурентоспроможного ігрового онлайн застосунку в жанрі Roguelike.

Загальний опис:

- сфера застосування: Ігровий застосунок в дозволить гравцям насолоджуватися випадково згенерованими рівнями і безліччю різноманітних

ворогів та предметів. Буде використовуватися як засіб розваги та відпочинку, а також як інструмент для вивчення нових навичок та розвитку логічного мислення. Крім того, гра може використовуватися як інструмент для соціалізації, коли гравці обговорюють свої досягнення та спільно вивчають гру.

- характеристики користувачів: Користувачами додатку будуть гравці.
- загальна структура і склад системи: Системи складається з інтерфейсу користувача, серверу на Photon Cloud, системи глобальних налаштувань застосунку, системи керування, системи зміни зовнішнього вигляду ігрового персонажу, системи багатокористувацької гри та системи пошуку та створення ігрових кімнат.
- загальні обмеження: Система не буде включати в себе функції створення модифікації та редагування ігрових рівнів.

Функції системи:

- Функція глобального налаштування ігрового застосунку:

Опис функції. Функція глобального налаштування ігрового застосунку дозволяє користувачу налаштувати вікно ігрового застосунку, загальну гучність звукових ефектів та гучність окремих звукових ефектів, клавіші керування.

Вхідна та вихідна інформація. Користувач вносить зміни до певного поля налаштувань та зберігає ці зміни. В залежності від змін можуть змінитися: розмір вікна ігрового застосунку, максимальна частота кадрів, гучність звукових ефектів, клавіші керування.

Функціональні вимоги. Система повинна забезпечувати зрозумілу систему зміни налаштувань та коректно зберігати їх між ігровими сесіями.

- Функція пошуку ігрових кімнат:

Опис функції. Функція дозволяє користувачу отримати список всіх кімнат створених іншими користувачами, надає можливість шукати кімнати за назвою та приєднуватися до них.

Вхідна та вихідна інформація. Користувач відкриває вікно пошуку кімнат та отримує список всіх доступних кімнат. Також користувач може ввести назву кімнати і отримати список всіх кімнат назва якої має відповідну послідовність символів, що ввів користувач.

Функціональні вимоги. Система повинна коректно відображати список кімнат та надавати можливість приєднатися до однієї з них.

- Функція створення кімнат:

Опис функції. Функція дозволяє користувачу створити кімнату з власними налаштуваннями та отримати статус MasterClient для поточної кімнати.

Вхідна та вихідна інформація. Користувач вводить назву кімнати, вказує максимальну кількість гравців, ігровий режим, може заблокувати кімнату задавши пароль. Після створення кімнати користувач опиниться на сцені підготовки до гри, де може очікувати поки до його кімнати приєднаються інші користувачі, почати гру, чи закрити кімнату покинувши її.

Функціональні вимоги. Система повинно коректно створювати кімнати з налаштуваннями, що вказав користувач та додавати її до списку інших кімнат. Давати можливість почати гру для користувачів, що знаходяться в даній кімнаті, чи покинути кімнату.

- Функція зміни зовнішнього виглдяу ігрового персонажу:

Опис функції. Функція дозволяє користувачу змінити зовнішній вигляд ігровго персонажу.

Вхідна та вихідна інформація. Користувач задає зовнішній вигляд игровому персонажу за допомогою відповідно інтерфейсу, та зберігає її. Під час гри ігровий персонаж приймає той зовнішній вигляд який обрав користувач.

Функціональні вимог. Система повинна надавати користувачу можливість змінювати зовнішній вигляд ігрового персонажу, коректно зберігати її меж ігровими сесіями та завантажувати під час гри.

- Функція керування ігровим персонажем:

Опис функції. Функція дозволяє користувачу керувати своїм ігровим персонажем за допомогою натискання клавіш.

Вхідна та вихідна інформації. Користувач натискає певну клавішу, а його ігровий персонаж виконує відповідну до цієї клавіші функцію, якщо це можливо.

Функціональні вимоги. Система повинна сканувати натисненні користувачем клавіші, та давати команди ігровому персонажу, що прив’язані до відповідних клавіш.

- Функція синхронізації ігрових об’єктів між користувачами:

Опис функції: Функція дозволяє синхронізувати певні параметри об’єктів, або події між користувачами.

Вхідна та вихідна інформація. Користувач робить запит на певну дію до користувача MasterClient. MasterClient отримавши запит обробляє його та синхронізує з іншими користувачами.

Функціональні вимоги. Система повинна стабільно та коректно синхронізувати необхідні параметри чи події на сцені й об’єктах між користувачами.

Вимоги до інформаційного забезпечення:

- Джерела та зміст вхідної інформації (даних): Система буде спиратися на інформацію про налаштування ігрового застосунку, звонішності ігрвого персонажу та поточної кімнати(якщо користувач знаходиться в багатокористувацькій грі).
- Нормативно-правова та довідкова інформація (класифікатори, довідники тощо): Система відповідатиме відповідним законам та нормативним актам, що стосуються ігрових застосунків.
- Вимоги до методів організації, зберігання та підтримки інформації: Система повинна мати стабільний сервер для обміну інформацією між

користувачами, там надійні методи зберігання локальної інформації про налаштування.

Вимоги до апаратного забезпечення:

Для роботи системи в багатокористувацьких режимах потрібні сервери з достатньою обчислювальною потужністю та обсягом пам'яті для обробки великих обсягів трафіку. Локальна гра не потребує апаратного забезпечення.

Вимоги до програмного забезпечення:

- Архітектура програмного забезпечення: Система повинна мати масштабовану та ефективну програмну архітектуру з чітким розподілом між інтерфейсом користувача, сервером та внутрішніми системами.

- Програмне забезпечення системи: Система повинна використовувати надійне та сучасне системне програмне забезпечення, з регулярними оновленнями.

- Мережеве програмне забезпечення: Система повинна використовувати безпечне та надійне мережеве програмне забезпечення для зв'язку між інтерфейсом користувача та внутрішніми системами.

- Мова та технологія розробки програмного забезпечення: Програмне забезпечення для ігрового астосунку буде розроблено з використанням сучасного та надійного програмного стеку. Інтерфейс користувача буде побудований з використанням інструментів Unity, популярної та ефективної мов програмування C#. Сервер так компоненти для його інтеграції в середовище Unity будуть отримані від сервісу Photon Cloud.

Вимоги до зовнішнього інтерфейсу:

- Інтерфейс користувача: інтерфейс користувача для ігрового застосунку повинен бути інтуїтивно зрозумілим, зручним і швидким у користуванні. Він включатиме функції налаштувань гри, створення ігрової кімнати, відображення списків кімнат, запуску локальної гри, виходу з застосунку, відображення різного роду статистики про ігрового персонажа, предмети та зброю. Користувацький інтерфейс буде доступний з настільних пристроїв і буде розроблений таким чином, щоб забезпечити однаковий користувацький досвід на всіх платформах.
- Апаратний інтерфейс: системи ігрового застосунку буде розроблена для роботи на стандартному апаратному забезпеченні та операційних системах. Користувачі будуть мати мінімальну кількість вимог до апаратного забезпечення завдяки високому рівню оптимізації гри.
- Інтерфейс програмного забезпечення: програмний інтерфейс системи буде заснований за допомогої інструментарію середовища Unity, а саме компонентів Canvas, GraphicRaycaster, Image, Button, Slider, CanvasScaler та багатьох інших.
- Протокол зв'язку: Photon Cloud використовує власний протокол Photon, який розроблений для забезпечення високої продуктивності та мінімальної затримки при передачі даних між клієнтами та серверами. Протокол Photon заснований на UDP (User Datagram Protocol), але включає додаткову логіку для обробки помилок і забезпечення надійності передачі даних. Він також підтримує шифрування даних для забезпечення безпеки та конфіденційності інформації, що передається. Протокол Photon є основним протоколом для зв'язку між клієнтськими програмами та серверами Photon Cloud.

Властивості програмного забезпечення:

- Доступність: доступ до програмного забезпечення буде надаватися користувачам через купівлю на відповідних торгівельних майданчиках.

- Супроводжуваність: Система ігрового застосунку буде розроблена таким чином, щоб її було легко оновлювати. Кодова база буде добре структурована, з чітким розподілом обов'язків між різними модулями та компонентами. Система буде спроектована таким чином, щоб забезпечити легке розгортання оновлень та нових функцій, не спричиняючи перебоїв у роботі користувачів або системи в цілому.
- Переносимість: Система ігрового застосунку буде розроблена таким чином, щоб бути портативною, тобто її можна було розгортати на різних платформах та інфраструктурах без значних модифікацій. Система буде побудована з використанням відкритих стандартів і технологій, що гарантує її роботу на будь-якому обладнанні та операційній системі, які їх підтримують.
- Продуктивність: Система ігрового застосунку буде розроблена з високою продуктивністю, що гарантує, її здібність обробляти великі обсяги локальних запитів та запитів до серверу, без значних втрат продуктивності.
- Надійність: надійність - це здатність системи виконувати свої функції послідовно і без помилок протягом тривалого часу. У контексті ігрового застосунку надійність має велике значення для того, щоб користувачі могли довіряти системі і отримувати приємний досвід від користування. Система повинна бути спроектована таким чином, щоб мінімізувати кількість помилок та незбалансованих механік. Для забезпечення надійності в системі повинні бути передбачені заходи для коректної обробки помилок та винятків. Це включає надання користувачам інформативних повідомлень про помилки за допомогої діілогових вікон. Крім того, система повинна бути спроектована таким чином, щоб справлятися з великою кількістю запитів від користувачів.
- Безпека: система буде передбачати заходи для захисту даних користувачів, включаючи шифрування конфіденційних даних завдяки функціям, що надає сервіс Photon Cloud.

Висновки до розділу 1

В розділі описано особливості галузі розробки комп’ютерних ігор, включаючи основні етапи процесу створення ігор, важливі аспекти геймдизайну, використання сучасних технологій, вплив на ринок розваг та роль ігор у суспільстві. Наведено опис ігрового жанру RogueLike, який характеризується випадково генерованими рівнями, перманентною смертю персонажа та різноманітними випробуваннями, зброями та предметами, що знаходяться у грі. Порівняно існуючі аналоги ігр в жанрі RogueLike, відмічено їх переваги та недоліки. Наведено специфікацію вимог для застосунку, що розробляється.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІГРОВОГО СЕРЕДОВИЩА

2.1 Діаграма прецедентів

Діаграмма використання (Use case diagram) - це одна з діаграм в UML (Unified Modeling Language), яка використовується для моделювання функціональних вимог до системи. Діаграма використання дозволяє описати, як користувачі (актори) взаємодіють з системою, тобто як вони використовують її функціональні можливості, щоб досягти своїх цілей.

На діаграмі використання відображається список акторів (користувачів), які взаємодіють з системою, та список сценаріїв (юз-кейсів), які описують, як актори використовують функціональні можливості системи. Сценарії відображаються у вигляді овалів або прямокутників з назвою функції або послідовністю операцій, які повинні бути виконані.

Діаграма використання дозволяє зосередитися на функціональних вимогах системи, а також зрозуміти, як користувачі будуть взаємодіяти з системою та які вимоги вони мають до неї. Вона може бути використана як інструмент для аналізу, проектування та комунікації вимог між розробниками програмного забезпечення, менеджерами проектів та клієнтами.

Складено глосарій проекту (див Таблицю-2.1)

Таблиця 2.1 – глосарій проекту

Гравець	Користувач, який використовує базовий офлайновим функціоналом гри
Хост	Користувач, який створює кімнату для мультиплеєру

Edit Settings	Редагування налаштувань гри
Customization	Кастомізація зовнішності ігрового персонажа
Single Play	Звичайний ігровий режим
Find Room	Можливість під'єднатися до створеної хостом кімнати
Create Room	Можливість створити кімнату для подальшого під’єднання інших гравців
Multi Player	Режим гри с кількома гравцями в одній кімнаті

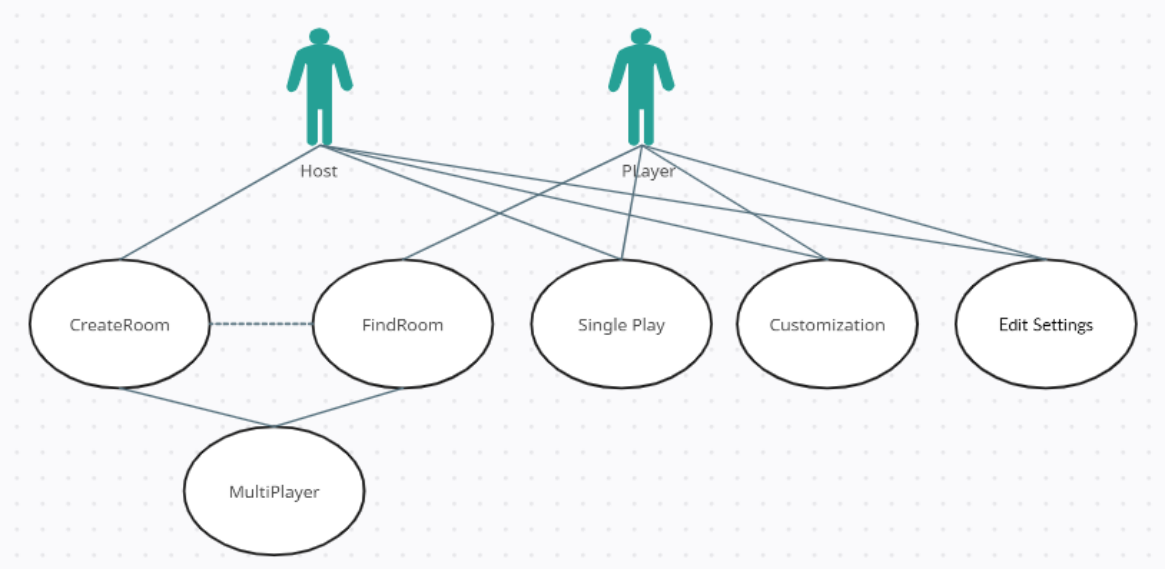

Рисунок 2.1 – Діаграма варіантів використання для розробки ігрового застосунку

Опис дійових осіб

Гравець (Player) – входить в систему, редагує налаштування гри, зовнішній вигляд персонажа. Має можливість грати однокористувацькому і багатокористувацькому режимі. У другому випадку може тільки приєднатися до вже створеної хостом кімнати.

Хост (Host player) — входить в систему, редагує налаштування гри, зовнішній вигляд персонажа. Має можливість грати як в однокористувацькому і багатокористувацькому режимі. У іншому випадку може тільки створити кімнату, до якої в подальшому приєднуються інші гравці.

Варіант використання "Редагування налаштувань"

Передумови: відсутні.

Пост умови: якщо варіант використання виконаний успішно, коривач змінить налаштування системи. В іншому випадку стан системи не змінюється.

На таблицях 2.2 — 2.4 наведено перебіг варіанту використання “ Редагування налаштувань”.

Таблиця 2.2 - Головний розділ сценарію виконання варіанта використання" Редагування налаштувань"

Варіант використання	Редагування налаштувань
Актори	Гравець, Хост
Короткий опис	Описує редагування доступних налаштувань застосунку
Мета	Змінити налаштування системи
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.3 - Типовий хід подій сценарію виконання варіанта використання «Редагування налаштувань»

Дії актора	**Відгук системи**
1. Користувач вносить зміни в налаштування	2. Система просить користувача підтвердити налаштування
3. Користувач підтверджує зміни в налаштуваннях	4. Система зберігає внесені зміни

Таблиця 2.2 - Винятки сценарію виконання варіанта використання "Вхід в систему"

Дії актора	**Відгук системи**
Виняток №1. Користувач не підтверджує зміни в налаштуваннях	
	4. Система не зберігає внесені зміни

Варіант використання "Кастомізація"

Передумови: відсутні.

Пост умови: якщо варіант використання виконаний успішно, користувач змінить зовнішній вигляд грального персонажа. В іншому випадку стан системи не змінюється.

На таблицях 2.5 — 2.6 наведено перебіг варіантів використання «Кастомізація».

Таблиця 2.5 - Головний розділ сценарію виконання варіанта використання «Кастомізація»

Варіант використання	Кастомізація

Актори	Гравець, Хост
Короткий опис	Дозволяє редагувати зовнішній вигляд аватара гравця
Мета	Змінити зовнішній вигляд ігрового аватара
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.6 - Типовий хід подій сценарію виконання варіанта використання «Кастомізація»

Дії актора	**Відгук системи**
1. Користувач вносить звіни в зовнішній вигляд	2. Система зберігає внесені зміни

Варіант використання "Одиночна гра"

Передумови: відсутні.

Пост умови: якщо варіант використання виконаний успішно, користувач переміщується на ігрову сцену.

На таблицях 2.7 — 2.1.8 наведено перебіг варіантів використання «Кастомізація».

Таблиця 2.7 - Головний розділ сценарію виконання варіанта використання «Одиночна гра»

Варіант використання	Одиночна гра

Актори	Гравець, Хост
Короткий опис	Основний ігровий режим де користувач проводитиме більшу частину часу
Мета	Розпочати гру
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.8 - Типовий хід подій сценарію виконання варіанта використання «Одиночна гра»

Дії актора	**Відгук системи**
1. Користувач натискає кнопку «Play» у головному меню	2. Система створює для гравця рівень та розпочинає ігровий процес

Варіант використання "Знайти кімнату"

Передумови: відсутні.

Пост умови: якщо варіант використання виконаний успішно, користувач приєднується до кімнат, що існує і з якої в подальшому може початися мультиплеєр.

На таблицях 2.9 — 2.12 наведено перебіг варіантів використання «Знайти кімнату».

Таблиця 2.9 - Головний розділ сценарію виконання варіанта використання «Знайти кімнату»

Варіант використання	Знайти кімнату
Актори	Гравець
Короткий опис	Можливість знайти і приєднатися до існуючої ігрової кімнати створеної хостом
Мета	Приєднатися до кімнати
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.10 - Типовий хід подій сценарію виконання варіанта використання «Знайти кімнату»

Дії актора	**Відгук системи**
1. Користувач вводить інформацію про кімнату	2. Система виводить інформацію про створені кімнати
3. Користувач обирає кімнату	4.Система просить ввести пароль до кімнати
5. Користувач вводить пароль	6. Система переміщує гравця до кімнати

Таблиця 2.11 - Винятки сценарію виконання варіанта використання «Знайти кімнату»

Дії актора	**Відгук системи**

Виняток №1. Користувач вводить неправильну інформацію про кімнату	
	2. Система виводить інформацію про створені кімнати
3. Користувач перевіряє інформацію і вводить правильно	4. Система виводить інформацію про створені кімнати

Таблиця 2.12 - Винятки сценарію виконання варіанта використання «Знайти кімнату»

Дії актора	**Відгук системи**
Виняток №1. Користувач вводить неправильну неправильний пароль	
	6. Система повідомляє про про помилку у введені паролю
7. Користувач виправляє помилку	8. Система переміщує гравця до кімнати

Варіант використання "Створити кімнату"

Передумови: відсутні.

Пост умови: якщо варіант використання виконаний успішно, користувач створить ігрову кімнату до якої в подальшому зможуть приєднатися інші гравці.

На таблицях 2.13 — 2.14 наведено перебіг варіантів використання «Знайти кімнату».

Таблиця 2.13 - Головний розділ сценарію виконання варіанта використання «Створити кімнату»

Варіант використання	Створити кімнату

Актори	Хост
Короткий опис	Можливість створити ігрову кімнату для подальшого мультиплеєру
Мета	Створити кімнату
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.14 - Типовий хід подій сценарію виконання варіанта використання «Створити кімнату»

Дії актора	**Відгук системи**
1. Користувач вводить інформацію про кімнату та створює її	2. Система вносить кімнату у список створених

Варіант використання "Мультиплеєр"

Передумови: користувач знаходиться в кімнаті як хост, ябо як звичайний гравець.

Пост умови: якщо варіант використання виконаний успішно, користувач разом з усіма присутніми у кімнаті гравцями розпочнуть мультиплеєрну сесію.

На таблицях 2.15 — 2.17 наведено перебіг варіантів використання «Знайти кімнату».

Таблиця 2.15 - Головний розділ сценарію виконання варіанта використання «Мультиплеєр»

Варіант використання	Мультиплеєр

Актори	Хост, Гравець
Короткий опис	Можливість розпочати мультиплеєрну сесію з присутніми у кімнаті гравціми
Мета	Розпочати мультиплеєрну сесію
Тип	Базовий
Посилання на інші варіанти використання	

Таблиця 2.16 - Типовий хід подій сценарію виконання варіанта використання «Мультиплеєр»

Дії актора	**Відгук системи**
1. Користувач підтверджує свою готовність до гри	2. Система пропонує хосту запустити гру
3. Хост підтверджує початок гри	4. Розпочинається мультиплеєрна сесія

Таблиця 2.17 - Винятки сценарію виконання варіанта використання «Мультиплеєр»

Дії актора	**Відгук системи**
Виняток №1. Користувач не підтверджує свою готовність до гри	
	2. Система звертає увагу гравця на відповідний елемент інтерфейсу

3. Користувач підтверджує свою готовність до гри	

Візуальна карта застосунку (Application map) - це графічне представлення архітектури програмного забезпечення, яке показує взаємозв'язки між компонентами та їх залежності. Вона може бути використана для відображення структури системи, включаючи компоненти, модулі, підсистеми, сервіси, бази даних та інші складові.

Візуальна карта застосунку надає можливість швидко отримати загальне уявлення про структуру системи та її компоненти, а також виявити проблемні або складні місця, такі як залежності між компонентами, дублювання функцій, незгоди у інтерфейсах та інші аспекти, що можуть впливати на продуктивність та ефективність системи.

Візуальна карта застосунку може бути використана як інструмент для аналізу та планування проектів, а також для комунікації між розробниками, менеджерами та іншими учасниками проекту. Вона також може бути використана для документування системи та її компонентів, що сприяє полегшенню її підтримки та розвитку.

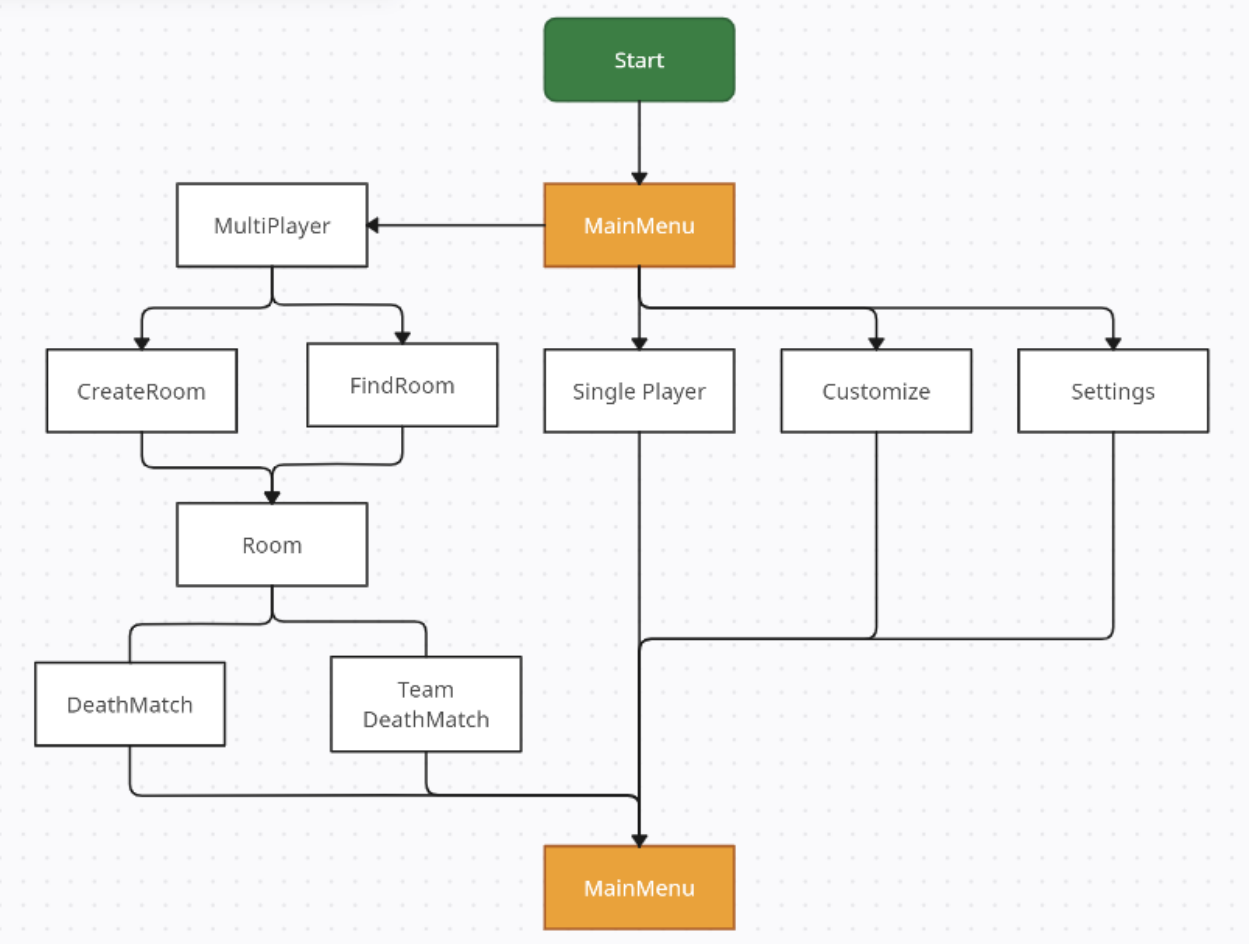

Рисунок 2.2 — Візуальна карта ігрового застосунку

2.2 Графіка, анімації та звукові ефекти

Графіка, звукові ефекти, їх поєднання та стиль мають дуже великий вплив на сприйняття користувачем ігрового процесу. Уся графіка в грі має бути стилізованою, й мати свої критерії. Звуки також мають свій стиль, який складається з гучності, степені реалістичності, музики, її жанру, настрою та якості.

Графіка повинна бути намальована для 2D простору в стилі Pixel art. Дуже важливою частиною графіки є палітра яка використовується при створенні графіки. Палітру можно створити самому чи користуватися вже існуючою. Вона

складається з відтінків різних кольорів які гармонічно комбінуються та утворюють стиль кольору.

При створенні графічних елементів важливо правильно використовувати переходи між відтінками, та не забувати під яким кутом будуть знаходитися графічні елементи відносно користувача. Користувач буде бачити ігрове поле під непрямим кутом, може бачити відразу декілька сторін об’єкту, тобто графіка створюватиме імітацію об’єму. Палітра має складатися з яскравих кольорів, щоб графіка передавала позитивний настрій гри, а графічні ефекти були яскравими.

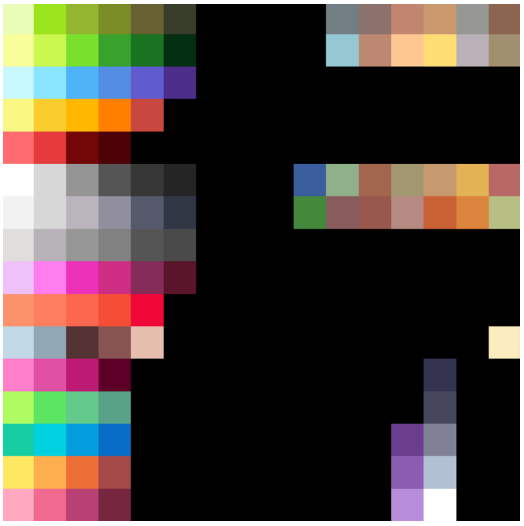

Рисунок 2.3 – Палітра для графіки

2.3 Ігровий інтерфейс

Ігровий інтерфейс користувача - це базова складова будь-якого ігрового середовища. Через ігровий інтерфейс користувач взаємодіє з ігровими механіками, отримує певну інформацію, повідомлення чи попередження. На кожній ігровій сцені повинен бути унікальний інтерфейс, можливості якого будуть відповідати механікам ігрової сцени.

Першим базовим ігровим інтерфейсом є інтерфейс головного меню. Тут користувач має набір кнопок кожна з яких направлятиме його до наступного

інтерфейсу чи завантажувати наступну ігрову сцену. Також у головному меню буде відображатися ігровий персонаж користувача з актуальними налаштуваннями зовнішності.

Інтерфейс головного меню буде складатися з:

- Кнопка Play: кнопка, що запускає гру на одинці у режимі Roguelike.
- Кнопка Multiplayer: кнопка, що відкриває перед користувачем новий інтерфейс - інтерфейс меню пошуку та створення кімнати для гри у мережі.
- Кнопка Settings: кнопка, яка відкриває перед користувачем інший інтерфейс - інтерфейс вікна налаштувань гри.
- Кнопка Customize: кнопка, яка відкриває інший інтерфейс - інтерфейс налаштування зовнішності ігрового персонажу.
- Кнопка Exit: кнопка, яка буде закривати ігровий застосунок.

Рисунок 2.4 – Макет головного меню

Виходячи з опису функцій кнопок головного меню виникають ще декілька інтерфейсів які можна зустріти на сцені головного меню.

Інтерфейс налаштувань зовнішності ігрового персонажу: цей інтерфейс надаватиме користувачу набір інтерактивних елементів, які надаватимуть користувачу інформацію, про поточну частину одяги, що зараз налаштовується,

можливі елементи відповідної частини одягу, які можна одягнути на ігрового персонажа, можливість перемикати поточну частину одягу та можливість повернутися до головного меню.

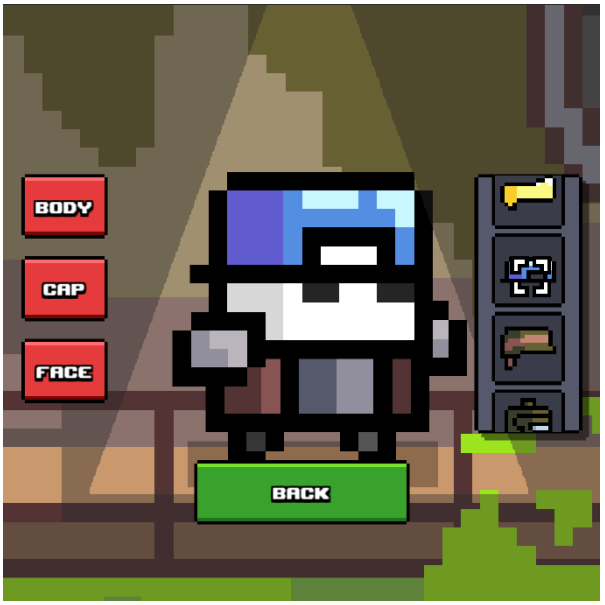

Рисунок 2.5 – Макет інтерфейсу налаштувань зовнішності персонажу

Інтерфейс вікна налаштувань гри: цей інтерфейс надає можливість набір інтерактивних елементів, які можуть змінювати поточні налаштування ігрового застосунку, відображати актуальну інформацію про певні налаштування та зберігати їх. Це інтерфейс буде існувати на всіх можливих ігрових сценах, що надасть можливість користувачу змінити налаштування у будь який момент гри.

Рисунок 2.6 – Макет інтерфейсу вікна налаштувань гри

Інтерфейс пошуку та створення кімнати для гри у мережі: інтерфейс, що надає користувачу обрати чи він хоче створити свою кімнату, чи хоче приєднатися до існуючої.

Рисунок 2.6 – Макет інтерфейсу меню пошуку та створення кімнати для гри у мережі

Інтерфейс вікна створення кімнати для гри у мережі: інтерфейс, що надає користувачу можливість створити кімнату для гри у мережі з власними

налаштуваннями, назви, ігрового режиму, максимальної кількості гравців та приватності.

Рисунок 2.7 – Макет інтерфейсу вікна створення кімнати для гри у мережу

Інтерфейс вікна пошуку кімнати для гри у мережі: інтерфейс, що надає користувачу можливість знайти та приєднатися до кімнати для гри у мережі.

Рисунок 2.8 – Макет інтерфейсу вікна пошуку кімнати для гри у мережу

Наступна сцена з якою може зустрітися користувач є сцена підготовки користувачів до гри. Тут вони натискають кнопку Ready, якщо вони готові до гри

чи кнопку Not Ready, якщо не готові до гри. Творець кімнати може натиснути кнопку Play, якщо всі гравці готові до гри, після чого почнеться зворотній відлік до початку гри. Також він може натиснути кнопку Cancel, після чого зворотній відлік буде перервано. Всім користувачам доступна кнопка Leave, після натиснення на яку користувач повернеться до головного меню.

Рисунок 2.9 – Інтерфейс сцени GetReadyScene

Останнім та головним інтерфейсом гри є інтерфейс сцени GameScene. На ньому розташовані наступні елементи:

1. Шкала здоров’я - візуально відображає параметри максимального та поточного значення здоров’я ігрового персонажу

2. Панель зброї - розташована в лівому нижньому куті екрана. Відображає іконку поточної зброї, іконку типу набоїв, запас відповідного типу набоїв та параметри магазину зброї.

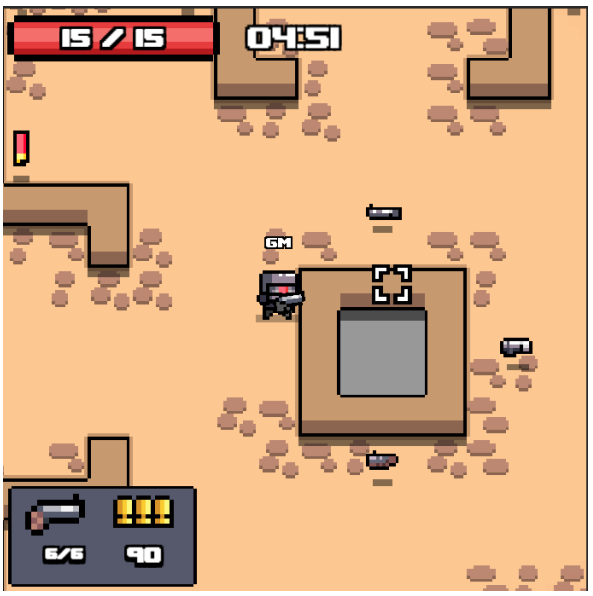

Рисунок 2.10 – Інтерфейс сцени GameScene

Також до ігрового інтерфейсу належить курсор. Та спливаючі повідомлення про отримані бонуси чи зброю.

Одним з найголовніших елементів ігрового процесу є ігрова камера. Вона дуже сильно впливає на сприйняття користувачем ігрового процесу. Тому для ігрової камери було завантажено окремий набір компонентів під назвою CameraCinemachine. Він надає велику кількість можливостей для роботи з ефектами камери.

За допомогою цього компонента для камери було створено ефект переслідування ігрового персонажа, тряски при вибухах чи пострілах, та пересування камери за ігровим курсором.

2.5 Багатокористувацька гра

Система багатокористувацької гри повинна надавати можливість користувачам грати разом, тобто синхронізувати їх дії та зміни на ігровому полі в межах ігрової кімнати. Сервіс Photon Cloud пропонує свій свій інструментарій для реалізації багакоритсувацької гри. Він надає сервер з безкоштовним та

платними планами CCU. Також має свій набір пакетів для інтеграції в Unity такі як Photon Pun 2 Free та Photon 2 ++. Завантаживши Photon Pun 2 Free в середовище Unity можна налаштувати зв’ язок між застосунком та сервером. Для роботи в середовищі Unity пакет Photon Pun 2 Free надає велику кількість компонентів та класів для реалізації середовища багатокористувацької гри. Основним компонентом є компонент PhotonView, який має бути присвоєний будь-якому компоненту, який потрібно синхронізувати. Компонент PhotonView синхронізує інформацію між клієнтами за допомогою RPC-запитів, які зберігають у собі інформацію та користувачів, які її отримають. На базі цього можна створити велику кількість користувацьких компонентів для синхронізації користувацьких типів даних.

Висновок до розділу 2

У цьому розділі було проведено аналіз потреб користувачів та складено Use case для кожного з акторів, включаючи їх дії в ігровому застосунку. На основі цього були створені макети ігрового застосунку та його карта, що допомагають візуалізувати структуру застосунку та навігацію для користувача.

Спираючись на поставлене завдання та вище описані дані було спроектовано макети інтерфейсів для ігрового застосунку, створено, палітру, набори графіки, звукових та графічних ефектів, змодельовано систему створення та пошуку кімнат для багатокористувацької гри й систему синхронізації сцен, об’ єктів, параметрів та подій між користувачами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО СЕРЕДОВИЩА ЗАСТОСУНКУ В ЖАНРІ ROGUELIKE

3.1 Опис ідеї та створення плану розробки

Для виконання поставлених вимог потрібно було придумати графічний та звуковий дизайн для гри на базі яких будуть створені ігрові рівні, інтерфейси, об’єкти та деякі механіки. Створити систему для гри мережею, та оптимізовану систему синхронізації користувачів. Для реалізації задуманої ідеї було створено план розробки ігрового середовища:

1) малювання графіки;
2) створення графічних об’єктів та анімацій;
3) створення звукових ефектів на базі анімацій та іншого звукового супроводження;
4) створення системи кімнат для гри по мережі та синхронізації користувачів

3.2 Малювання графіки

Персонажі: дизайн ігрових персонажів було спроектовано таким чином, щоб вони здавалися милими. Для цього пропорції тіла були дуже сильно змінені. Для створення образу милого персонажу голова, очі та руки мають бути більшими за інші частини тіла.

Частини тіла ігрового персонажу користувача намальовані як окремі елементи, так як вони можуть окремо змінюватися.

Також зберігається пропорції між ігровими персонажами таким чином, щоб між ними не було дефектної різниці в розмірах.

У більшості персонажів є спільні стани рухів:

1) idle;

2) walk;

Тому для кожного персонажа намальовані покадрові анімації цих станів.

Рисунок 3.1 – Покадрова анімація персонажів

Зброя: зброя в грі має бути дуже різноманітної. Різноманітність повинна заключатися в механіках, типу снарядів, дизайну та анімаціях перезарядки. Деяка зброє може бути автоматичною, а деяка зброя перед пострілом повинно імітувати накопичення енергії. Зброє може бути різних типі:

1) ближнього бою;
2) дальнього бою;
3) змішаного типу;

В кожної зброї є декілька основних станів станів від яких залежить анімація зброя:

1) attack;
2) calm;
3) reload;
4) charge;

Для кожного з цих станів намальовано покадрові анімації. Особливої уваги потрібно зазначити анімаціям перезарядки так як їх створення це особливий процесс. Анімація перезарядки зброї повинна бути дуже детальною, динамічною

та унікальною. Зброя повинна заздалегідь спроектована таким чином, щоб мати певні деталі з якими потім буде створюватися анімація перезарядки. При певних рухах зброя повинна смикатися вверх чи вниз, від неї можуть бути відділені певні деталі, чи вона може кардинально змінювати свою форму на деяких кадрах.

Рисунок 3.2 – Покадрова анімація зброї

Візуальні ефекти: візуальні ефекти можуть бути намальовані як покадрова анімація, або у вигляді певних фрагментів з яких потім буде складатися ефект в середовищі Unity. Вони повинні бути яскравими та деталізованими так як вони дуже сильно впливають на сприйняття користувачем певних ігрових подій.

Рисунок 3.3 – Покадрова анімація графічних ефектів

Об’єкти середовища: об’єктами середовища можуть бути будь-які об’єкти з яких складається ігровий рівень. Ігровий рівень може складатися з тайлів та окремих об’єктів. Тайли намальовані по клітинкам, так як потім будуть розміщені на сітці.

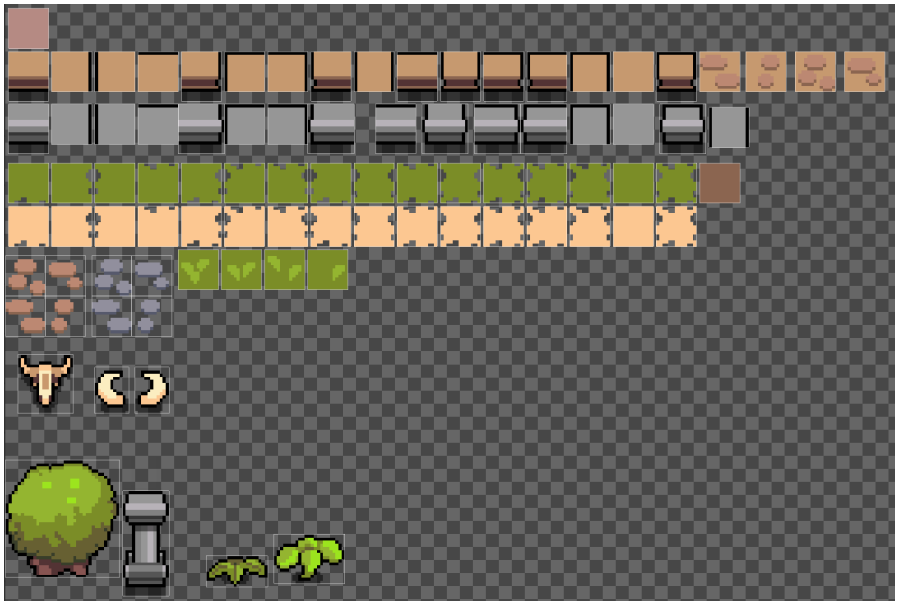

Рисунок 3.4 – Тайли та декорації

Складові інтерфейсу: інтерфейс буде складатися з певних кнопок, іконок та панелей. Тому для інтерфейсу було намальовани набір базових зображень які потім будуть певним чином налаштовуватися в середовищі Unity.

Рисунок 3.5 – Складові інтерфейсу

Бонусні предмети: бонусні предмети - це предмети після взаємодії з якими користувачу будуть надані певні бонусні характеристики чи зброя. Всі предметі в грі можуть з’являтися після вбивства ворогів, з скриньок чи на місці умовної точки через певні інтервали часу. Вони будуть зображені як обертаючийся об’єкт - тобто в них буде покадрова анімація в якій буде імітовано 3D. Малювання такої анімації складається з чотирьох основних кадрів та чотирьох проміжних, що дуже сильно спрощує процес зображення такого об’єкту.

Рисунок 3.6 – Покадрові анімації обертаючихся об’єктів

3.3 Створення графічних об’єктів та анімацій

Для роботи з графікою Unity пропонує велику кількість інструментів та компонентів. Кожен графічний елемент на сцені повинен мати компонент SpriteRenderer, або Image, якщо об’єкт є складовою Canvas. Canvas - це регульоване полотно на якому можуть бути розміщені елементи інтерфейсу, розмір та положення яких автоматично буде підстроюватися під розширення екрану користувача в залежності від налаштувань Canvas.

Уся 2D графіка в Unity сортується по шарам, індексам, положенням на сцені та в ієрархії об’єктів. В Unity існують загальні налаштування для 2D графіки, та окремі налаштування для кожного графічного об’єкту.

Розглянемо компонент SpriteRenderer. У вікні інспектора він має наступні поля для налаштувань:

1. Sprite - зберігає посилання на поточне зображення.

2. Color - зберігає посилання на об’єкт типу Color, який відповідає за колір відтінку зображення.

3. Flip - дозволяє перевернути зображення по осі x чи y не торкаючись компоненту Transform.

4. Draw Mode - поле для налаштування типу відображення.

5. Mask Interaction - налаштовує взаємодію компоненту з графічними масками.

6. Sprite Sort Point - налаштовує умовний центр зображення, спираючись на який, буде відбуватися його сортування.

7. Material - зберігає посилання на матеріал, що використовується для зображення.

Також в ньому є вкладка Additional Settings, яка розвертає додаткові поля для налаштувань:

1. Sorting Layer - налаштовує шар сортування зображення

2. Order in layer - налаштовує чергу сортування в шарі сортування

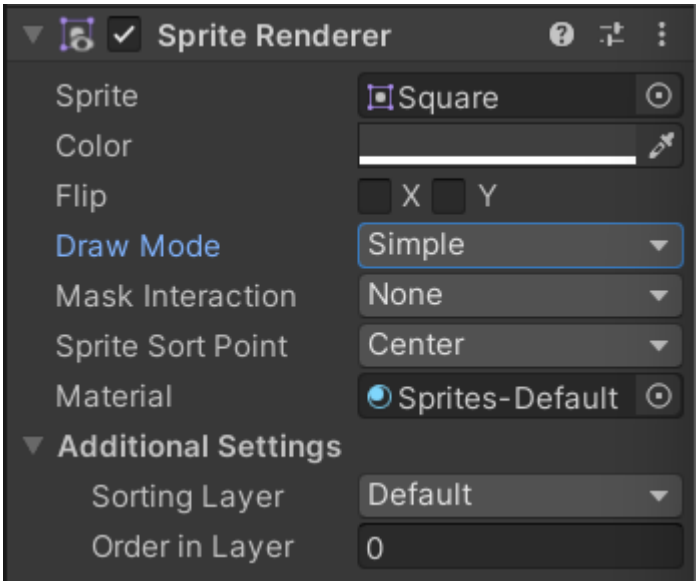

Рисунок 3.7 – Компонент Sprite Renderer у вікні Inspector

Для роботи з графічними файлами Unity використовує компоненти Sprite, Texture 2D, та інструмент Sprite Editor. Кожне 2D зображення має тип Sprite. Графіка перед завантаженням в середовище Unity може бути запакована в атласи. Атлас - це графічний файл який складається з набору зображень, які потім будуть розмічені та виділені в ньому як окремі зображення. Для розмітки атласів Unity пропонує інструмент Sprite Editor. Він надає можливість розмітити та відокремити зображенні на атласі, налаштувати фізичну форму цих зображень, їх Sprite Sorting Point та багато іншого. Для того, щоб графічний файл можно було поділити на окремі спрайти він повинен мати спеціальні налаштування, а саме поле Sprite Mode повинно дорівнювати Multiple(по замовчуванню воно дорівнює Single).

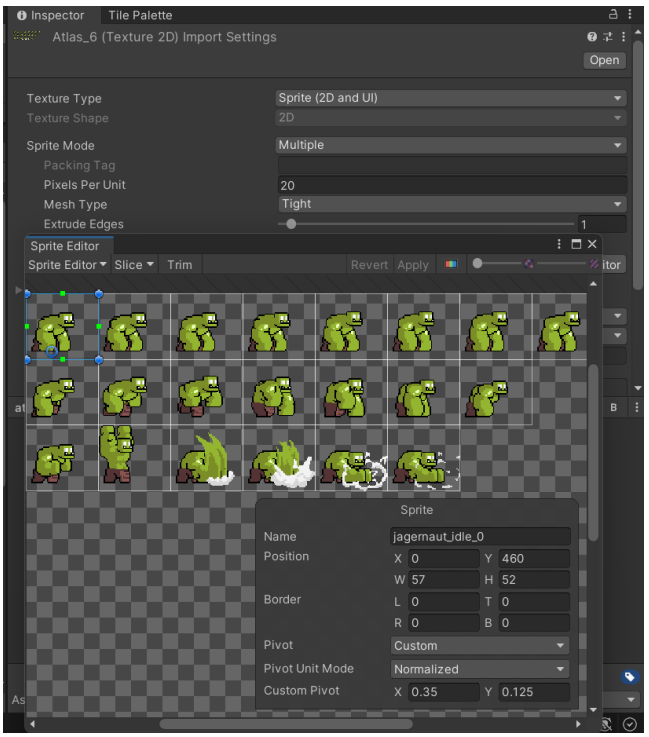

Рисунок 3.8 – Налаштування атласу за допомогою інструменті Sprite Editor

Намалювавши персонажа чи графічний ефект ми розміщуємо їх зображенні на атласі та завантажуємо у середовище Unity. На атласі можна розміщувати таку кількість зображень, яку дозволяю розмір атласу. В середовищі Unity за допомогою інструменту Sprite Editor, можна виділити частину зображення, яка нам потрібна, та відокремити її як окремий Sprit, давши йому назву та налаштувавши центр.

Розібравшись як працювати з 2D графікою в середовищі Unity перейдемо до анімацій. Для роботи з анімаціями Unity пропонує компоненти Animation, Animator Controller та інструменти Animation та Animator для роботи з ними.

Компонент Animation зберігає у собі назву анімації та ключі. Ключ анімації зберігає у собі інформацію про часовий проміжок та зміни що відбулися на ньому. Для роботи з Animation рушій Unity пропонує інструмент Animation. В якому можна налаштувати ключі та зміни, що відбулися на певних проміжках часу. Також можна запустити демонстрацію анімацій на певному об’єкті чи поставити демонстрацію на паузу. Інструмент Animation має два режими:

1. Режим налаштувань - це режим в якому будь які зміни, що відбулись на об’єкту з компонентом Animator Controller чи на дочірніх від нього об’єктах, будуть зберігатися в ключі на вказаному проміжку часі для поточної анімації.

2. Режим перегляду - це режим в якому будь які зміни, що відбулись на об’єкту з компонентом Animator Controller чи на дочірніх від нього об’єктах, не зберігаються.

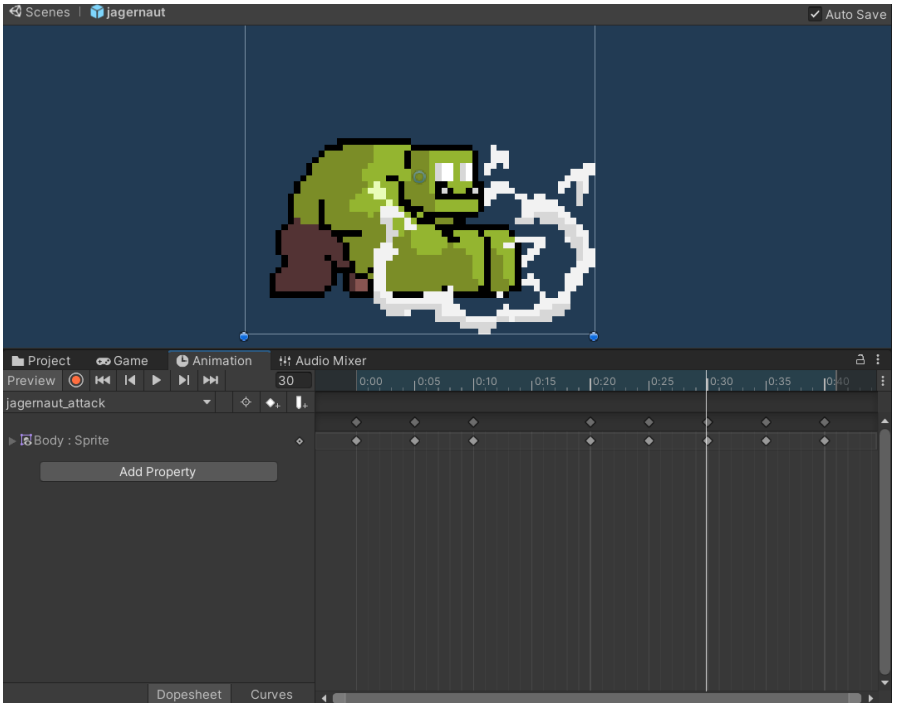

Рисунок 3.9 – Інструмент Animation

Animation Controller це компонент, що зберігає в собі певний набір анімацій, та налаштування взаємодій цих анімацій. Для того щоб налаштувати анімації в Animator Controller треба їх додати до масиву анімації. Після цього у вікні Animator ми може налаштувати зв’язки анімацій та умови зміни анімацій за допомогою параметрів. Існує декілька типів параметрів:

1. Int
2. bool
3. float
4. trigger

Анімації у вікні Animator розташовані по спеціальним блокам. Також існують блоки по замовчуванню:

1. Exit

2. Entry

3. Any State

4. Default State

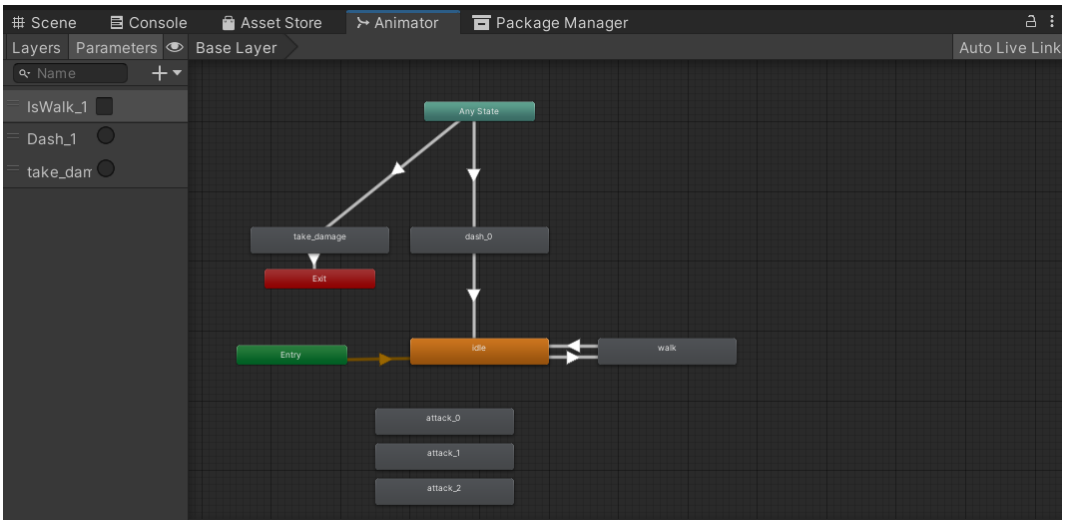

Рисунок 3.10 – Інструмент Animator

Всі анімації у проекті покадрові. Тобто ключі зберігають інформацію про зміну зображення в компоненті Sprite Renderer чи Image.

Також середовище Unity пропонує компонент Particle System для роботи з графічними ефектами. Він дозволяє створювати різні 2D та 3D ефекти які складаються з динамічних частинок. В цьому компоненті є налаштування життєвого циклу частинок, їх швидкості, кольору, зображення, розміру, траєкторії руху та багато іншого. Однією з особливостей цього компонента є можливість створювати покадрову анімацію, задавши массив зображень, які будуть змінюватися на протязі життєвого циклу кожної частинки.

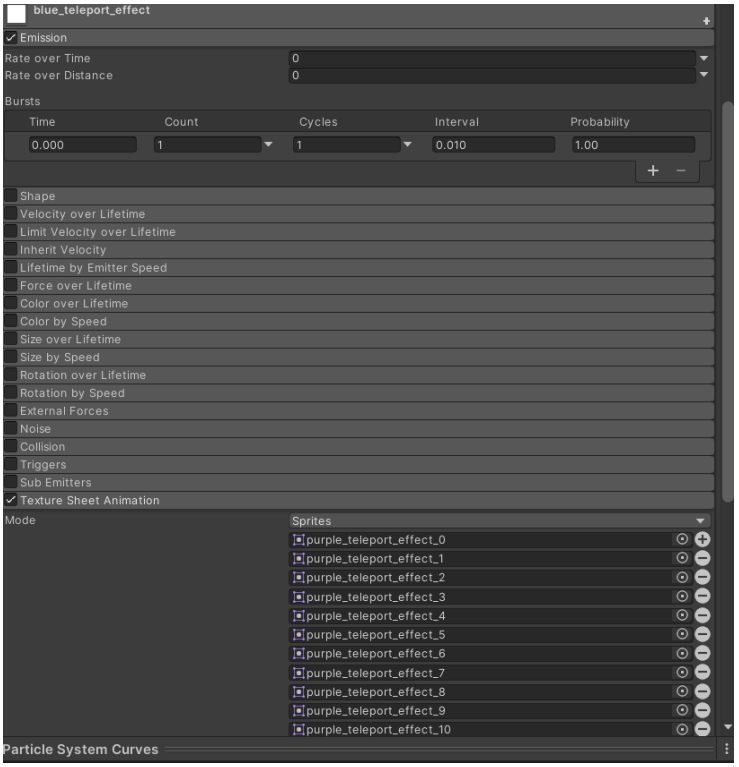

Рисунок 3.11 – Компонент Particle System

Для створення ігрового поля Unity надає інструмент Tile Palette та компоненти Tilemap й Grid. Tile Palette - це інструмент що дає можливістю малювати заданими текстурами по сітці. Тобто при наявності зображені ігрової поверхні можна додати їх до Tile Palette й потім створювати з них ігрову карту.

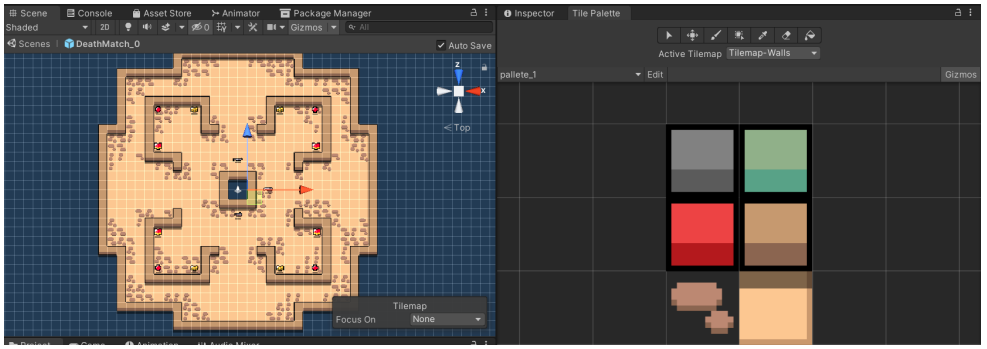

Рисунок 3.12 – Інструмент Tile Palette

Для роботи з інтерфейсом Unity пропонує величезний набір компонентів, таких як: Canvas, Button, Toggle, Scroll View, Image, Graphic Raycaster та інші. Canvas - це компонент, що утворює полотно, розмір якого залежить від розширення екрану, та розміру вікна застосунку. Положення та розмір об’єктів розташованих на ньому регулюється автоматично, але базові положення можна налаштувати заздалегідь. Компонент Graphic Raycaster робить Canvas інтерактивним, тобто без цього компонента жодна кнопка чи будь який інший інтерактивний об’єкт, що розташована на цьому Canvas, не буде реагувати на натискання чи будь яку іншу спробу інтерактивності.

Розглянемо компонент Button, так як він частіше всього зустрічається серед об’єктів інтерфейсу. Компонент Button має наступні поля:

1. поле Interactible яке регулює інтерактивність компоненту Button
2. поле OnClick в яке можна записати події, які будуть відбуватися при натисканні на кнопку.

3.4 Створення звукових ефектів

Для створення звукових ефектів на платформі Unity Assets Store було придбано декілька пакетів із звуками. Робота зі звуками відбувається у середовищі WavePad. Це програмне забезпечення надає достатньо широкий функціонал для створення та налаштування звуків.

Варто відмітити процес створення звуків для зброї. Вони створюються спираючись на анімацію під час якою вони повинні звучати. Кожен звук у звуковій доріжці має співпадати по часу з відповідним рухом в анімації. Анімація і звук програються одночасно і створюють гармонійне поєднання.

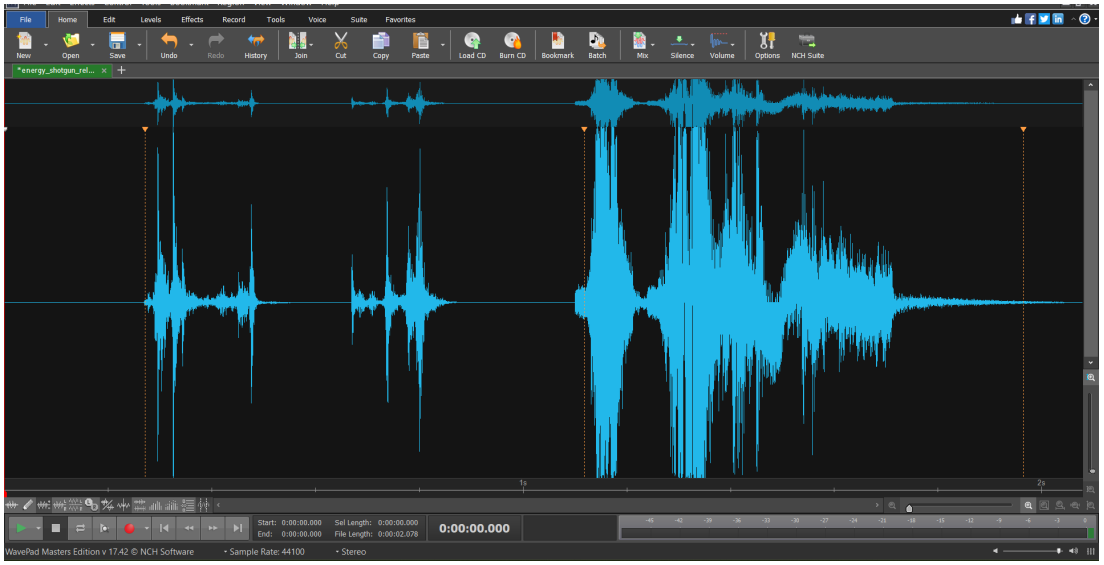

Рисунок 3.13 – Інструмент Tile Palette

Для роботи з звуковими ефектами Unity пропонує інструмент Audio Mixer та компоненти Audio Clip, Audio Source, Audio Listener, Audio Mixer Group. Інструмент Audio Mixer дозволяє створити та налаштувати групи звукових ефектів одна з яких існує по замовчуванню - группа Master. Було створено наступні групи звуків:

1. группа Effects - группа звукових ефекті, які відповідають за певні ігрові події.
2. группа Music - группа звукових ефектів, які відповідають за фонову музику в грі.
3. группа Ambient - група звукових ефектів, що відповідають за звуки оточення.

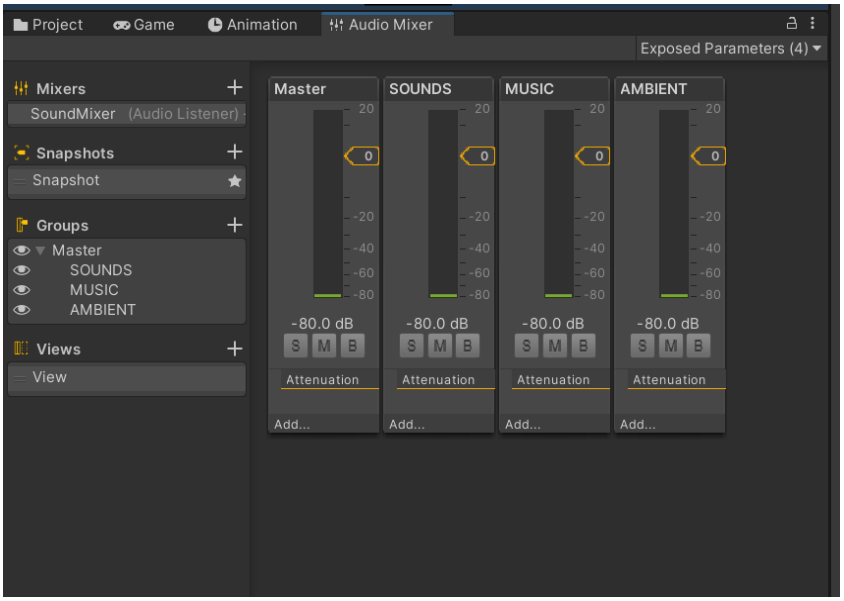

Рисунок 3.14 – Інструмент Audio Mixer

Компонент Audio Clip - компонент, що зберігає в собі налаштування аудіофайлу. Він дозволяє оптимізувати аудіофайл, що зменшить кількість займаної ним пам’яті та навантаження на систему під час звучання.

Компонент Audio Source - компонент, що відповідає за джерело звуку. Він має поле в якому зберігається посилання на Audio Clip об’єкт, який буде звучати з цього джерела, я також налаштування гучності та відстані на якій Audio Listener компонент буду чути цей звук.

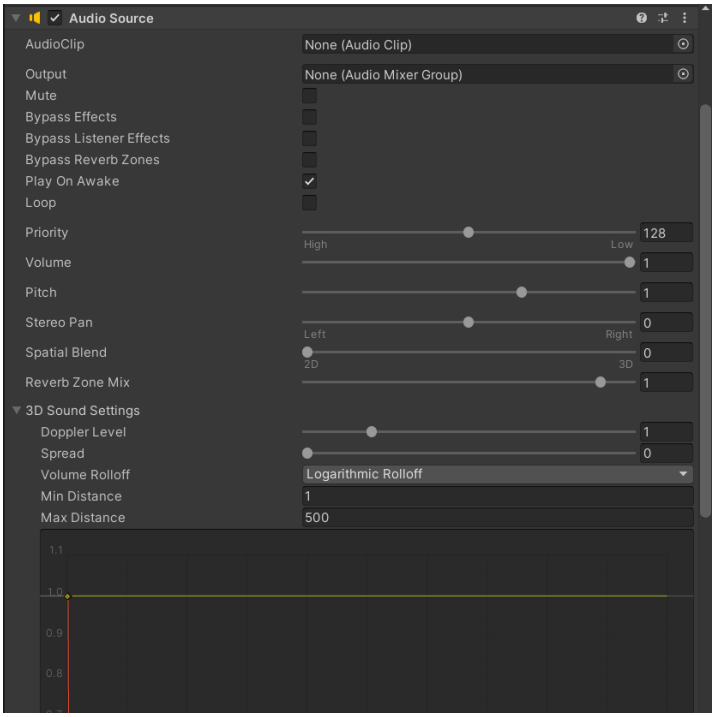

Рисунок 3.15 – Компонент Audio Clip

Для того щоб звук став 3D - тобто ставав гучнішими в залежності від відстані до Audio Listener треба встановити параметр Spatial Blend на занчення 1. Також для звуків які повинні грати циклічно можна встановити параметр loop на true.

Пакети звукових ефектів та музики було придбано на платформі дистрибуції Unity Assets Store. Деякі звуки після придбання допрацьовувалися у програмі WavePad.

3.5 Створення системи кімнат для гри у мережі та синхронізації користувачів.

Система мережевої гри доступна завдяки встановленому пакету Photon PUN. Він надає великий та надійний інструментарій для створення гри з можливістю грати у мережі, але потребує певних налаштувань.

По перше треба зареєструватися на офіційному сайті Photon та замовити там, для початку безкоштовний сервер на 20 CCU. Після створення безкоштовного сервера, завантажується спеціальний пакет в Unity проект, та налаштовується з’єднання між сервером та проектом.

Налаштувавши з’єднання між сервером та проектом можна переходити до підключення. Для підключення треба скористатися новою бібліотекою PhotonNetwork.

```
45  private void Start()
46  {
47      if (!PhotonNetwork.IsConnected) TryConnect();
48  }

49  public void TryConnect()
50  {
51      PhotonNetwork.ConnectUsingSettings();
52  }
```

Рисунок 3.16 – Фрагмент коду в якому відбувається підключення до серверу

Підключення до серверу відбувається при запуску сцени Init в якій відбувається ініціалізація та завантаження всіх налаштувань для подальшої роботи ігрового застосунку. Підключення відбувається через метод ConnectUsingSettings();

Після вдалого підключення викликається метод OnConnectedToMaster() який було перевизначено таким чином щоб він встановлював базові параметри локального користувача та підключав його до лоббі по замовчуванню.

```
public override void OnConnectedToMaster()
{
    Debug.Log("You has been connected to: " + PhotonNetwork.CloudRegion);

    if (PlayerPrefs.HasKey(GameSystem.Constancs.PREFS_APPEREANCE_KEY))
    {
        ExitGames.Client.Photon.Hashtable appereanceLoad = new ExitGames.Client.Photon.Hashtable();
        appereanceLoad.Add(GameSystem.Constancs.PLAYER_APPEREANCE_KEY, PlayerPrefs.GetString(GameSystem.Constancs.PREFS_APPEREANCE_KEY));

        PhotonNetwork.LocalPlayer.SetCustomProperties(appereanceLoad);
    }
    GameSystem.Actions.SetDeffaulLocalPLayerProperties();

    OnConnectedToServer?.Invoke();

    PhotonNetwork.JoinLobby();
}
```

Рисунок 3.17 – Фрагмент коду з перевизначенням методу OnConnectedToMaster()

Підключившись до лоббі ми можемо створювати в ньому кімнати та приєднуватися до існуючих. Для цього в PhotonNetwork є методи CreateRoom() та JoinRoom().

```
59  public bool CreateRoom(string roomName, int maxPlayers, GameSystem.Enums.GameModes gameMode, bool isLocked, int password = 0)
60  {
61      ExitGames.Client.Photon.Hashtable tempProperties = new ExitGames.Client.Photon.Hashtable();
62
63      tempProperties.Add(GameSystem.Constancs.ROOM_GAME_MODE_KEY, gameMode);
64      tempProperties.Add(GameSystem.Constancs.ROOM_LOCKED_KEY, isLocked);
65      tempProperties.Add(GameSystem.Constancs.ROOM_PASSWORD_KEY, password);
66      tempProperties.Add(GameSystem.Constancs.ROOM_IS_GAME_STARTED_KEY, false);
67      tempProperties.Add(GameSystem.Constancs.ROOM_IS_START_DELAY_KEY, false);
68      tempProperties.Add(GameSystem.Constancs.ROOM_IS_GAME_COMPLETE_KEY, false);
69      tempProperties.Add(GameSystem.Constancs.ROOM_IS_MASTER_LOADED_KEY, false);
70
71      string[] lobbyProperties = { GameSystem.Constancs.ROOM_GAME_MODE_KEY, GameSystem.Constancs.ROOM_LOCKED_KEY, GameSystem.Constancs.ROOM_PASSWORD_KEY };
72
73      RoomOptions newRoomOptions = new RoomOptions();
74
75      newRoomOptions.IsOpen = true;
76      newRoomOptions.MaxPlayers = (byte)maxPlayers;
77      newRoomOptions.CustomRoomProperties = tempProperties;
78      newRoomOptions.CustomRoomPropertiesForLobby = lobbyProperties;
79
80      return PhotonNetwork.CreateRoom(roomName, newRoomOptions);
81  }
```

Рисунок 3.18 – Фрагмент коду з користувацьким методом CreateRoom()

Користувачі, що знаходяться в кімнаті мають синхронізуватися між собою. Для цього було створено систему запитів до MasterClient. MasterClient - це користувач, який створив кімнату.

Система запитів до MasterClient працює таким чином, що при кожній спробі зробити будь що користувач спочатку відправляє запрос до MasterClient, отримавши запит той обробляє його на своїй стороні та відправляє цей запит всім останнім користувачам включаючи того, хто створив запит. Таким чином

MasterClient є користувачем який може грати та зберігати актуальну інформацію про поточні позиції та параметри всіх користувачів та об’єктів. Завдяки цій системи користувачі з низької швидкістю інтернету не будуть отримувати переваг перед іншими користувачами.

```
private void Ask()
{
    photonView.RPC("ReceiveAsk", RpcTarget.MasterClient);
}

[PunRPC]
0 references
private void ReceiveAsk()
{
    photonView.RPC("ReceiveOrder", RpcTarget.Others);
}

[PunRPC]
0 references
private void ReceiveOrder()
{
    //Do something
}
```

Рисунок 3.19 – Приклад роботи запитів

Запити також можуть переносити з собою певну інформацію у вигляді базових типів даних. Але для оптимізації цих самих вся інформація передається як ти byte.

Висновок до розділу 3

У цьому розділі було розроблене ігрове середовище для гри в жанрі Roguelike. Було створено стилізовану графіку та звуки. Також було налаштовано підключення до віддаленого сервера Photon так створено оптимізовану систему синхронізації користувачів.

ВИСНОВКИ

Було досягнуто поставленої мети шляхом виконання поставлених завдань повному обсязі. Проведено аналіз сучасних застосунків жанру Roguelike та порівняння їх переваг і недоліків. Визначено основний більшості аналогів - відсутність мультиплеєру. Складено специфікацію вимог до програмного забезпечення ігрового застосунку і його середовища, враховуючи всі функції. Було спроектовано ігрове середовище, враховуючи геймплейні, графічні та функціональні особливості гри, на базі чого була розроблена модель ігрового застосунку та його середовища. Розроблено функціональні модулі та елементи ігрового середовища, які дозволять користувачу швидко зануритись у ігровий процес та отримувати від нього задоволення. Створено кольорову палітру для графіки. Розроблено графічний стиль, на базі якого було створено графіку для гри. Було проведено роботу із звуковим супроводженням в грі, в результаті якої створено великий набір звукових ефектів та музикальних доріжок. Налаштовано ігрову камеру, ефекти якої додають особливостей геймплею та віддачу при певних подіях під час ігрового процесу. Було розроблено систему багатокористувацької гри з можливістю створювати кімнати та приєднуватись до них, а також оптимізовану систему синхронізації користувачів під час багатокористувацької ігрової сесії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Unity Documentation: https://docs.unity.com/(Last accessed: 01.05.2023)
2. Photon introduction: https://doc.photonengine.com/pun/current/getting-started/pun-intro(Last accessed: 09.05.2023)
3. Photon Unity Networking 2 Introduction: https://doc-api.photonengine.com/en/PUN/v2/index.html (Last accessed: 25.04.2023)
4. Adobe Photoshop tutorial: https://help.adobe.com/archive/en/photoshop/cs6/photoshop_reference.pdf (Last accessed: 24.04.2023)
5. WavePad user guide: https://help.nchsoftware.com/help/en/wavepad/win/help.pdf (Last accessed: 28.04.2023)
6. Cinemachine Documentation: https://docs.unity3d.com/Packages/com.unity.cinemachine@2.3/manual/index.html (Last accessed: 03.04.2023)
7. Knowable magazine: https://knowablemagazine.org/article/mind/2019/video-games-educational-benefits (Last accessed: 02.05.2022).
8. Anguera, J. A., Boccanfuso, J., Rintoul, J. L., Al-Hashimi, O., Faraji, F., Janowich, J., et al. Video game training enhances cognitive control in older adults. Nature 501, 2013, 97–101. doi: 10.1038/nature12486.