

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ _____ ” _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Гра у жанрі стратегії в реальному часі з подійно-орієнтованою
архітектурою

Виконав студент IV курсу, групи ІП-15
(шифр групи)
Шляхтун Денис Михайлович
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник ст.викл., д-р філософії, Сарнацький В. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант асистент, Шулькевич Т. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент доцент каф. ІСТ, к.т.н., доцент, Жданова О. Г.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Шляхтуну Денису Михайловичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Гра у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою

керівник проєкту Сарнацький Владислав Віталійович, ст. викл., д-р філософії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Передпроєктне дослідження: постановка завдання дипломного проєктування, аналіз предметної області, аналіз існуючих рішень, аналіз та моделювання бізнес-процесів.

2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, розроблення функціональних вимог, розроблення нефункціональних вимог, аналіз системних вимог, аналіз економічних показників програмного забезпечення, постановка завдання на розробку програмного забезпечення.

3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, архітектурні рішення та обґрунтування вибору засобів розробки, конструювання програмного забезпечення, аналіз безпеки даних.

4) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.

5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна компонентів програмного забезпечення

3) Схема структурна класів програмного забезпечення

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв'язання задачі	04.04.2025	
3	Постановка та формалізація задачі	11.04.2025	
4	Розробка інформаційного забезпечення	14.04.2025	
5	Алгоритмізація задачі	22.04.2025	
6	Обґрунтування вибору використаних технічних засобів	23.04.2025	
7	Розробка програмного забезпечення	10.05.2025	
8	Налагодження програми	15.05.2025	
9	Виконання графічних документів	18.05.2025	
10	Оформлення пояснювальної записки	26.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент

(підпис)

Денис ШЛЯХТУН

(ініціали, прізвище)

Керівник

(підпис)

Владислав САРНАЦЬКИЙ

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 32 таблиці, 38 рисунків та 31 джерело – загалом 79 сторінок.

Дипломний проєкт присвячений для здобуття ігрового досвіду у жанрі стратегій в реальному часі, розвитку стратегічного й тактичного мислення під час розваг, освітніх цілей шляхом досліджень у галузі ігрового штучного інтелекту або створення нових систем гри через відкритий програмний код.

Мета дипломного проєкту – досягнення високого рівня залученості гравців за рахунок реалізації та оптимізації механік гри, досягнення високих показників продуктивності, у першу чергу частоти кадрів, достатніх для плавного ігрового процесу; забезпечення високої гнучкості розробки складних систем із багатократними зв'язками шляхом застосування подійно-орієнтованої архітектури.

У розділі «Передпроєктне обстеження предметної області» здійснено постановку завдання дипломного проєктування, аналіз предметної області та існуючих рішень, зокрема порівняння з аналогами, моделювання бізнес-процесів.

Розділ «Розроблення вимог до програмного забезпечення» присвячений розробці та аналізу варіантів використання, функціональних, нефункціональних та системних вимог, економічних показників.

У розділі «Конструювання та розроблення програмного забезпечення» здійснено розробку архітектури та конструювання програмного забезпечення.

Розділ «Аналіз якості та тестування програмного забезпечення» присвячений перевірці досягнення мети та поставлених вимог за допомогою аналізу якості та тестування.

У розділі «Розгортання та супровід програмного забезпечення» описано процеси розгортання програмного забезпечення та його подальшого супроводу.

КЛЮЧОВІ СЛОВА: ГРА, СТРАТЕГІЯ В РЕАЛЬНОМУ ЧАСІ, UNREAL ENGINE, ПОДІЙНО-ОРІЄНТОВАНА АРХІТЕКТУРА, VISUAL STUDIO, WINDOWS.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 32 tables, 38 figures and 31 sources – in total 79 pages.

The diploma project is dedicated to providing gameplay experience in the real-time strategy genre, developing strategic and tactical thinking through entertainment, and achieving educational goals through research in the field of game artificial intelligence or the creation of new game systems via open-source code.

The purpose of the diploma project is to achieve a high level of player engagement by implementing and optimizing game mechanics; to reach high performance metrics, primarily frame rate sufficient for smooth gameplay; and to ensure high flexibility in the development of complex systems with multiple interconnections by applying an event-driven architecture.

The section «Pre-project study of the subject area» defines the objectives of the diploma design, analyses the subject area and existing solutions, including comparison with analogues and modelling of business processes.

The section «Development of software requirements» focuses on the design and analysis of use cases, functional, non-functional, and system requirements, as well as economic indicators.

In the section «Software design and development», the architecture and construction of the software are described.

The section «Software quality analysis and testing» is devoted to verifying the achievement of goals and defined requirements through quality analysis and testing.

The section «Software deployment and maintenance» describes the processes of software deployment and its further support.

KEYWORDS: GAME, REAL-TIME STRATEGY, UNREAL ENGINE, EVENT-DRIVEN ARCHITECTURE, VISUAL STUDIO, WINDOWS.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ГРА У ЖАНРІ СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ З ПОДІЙНО-
ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ**

Технічне завдання

КПІ.ПІ-1530.045490.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Денис ШЛЯХТУН

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	4
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	5
3	ПРИЗНАЧЕННЯ РОЗРОБКИ	6
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
4.1	Вимоги до функціональних характеристик	7
4.1.1	Користувацького інтерфейсу	7
4.1.2	Система керування під час ігрового процесу	10
4.1.3	Керування юнітами	10
4.1.4	Керування будівлями	11
4.1.5	Керування ресурсами	12
4.1.6	Місії для проходження.....	12
4.1.7	Збереження гри	13
4.1.8	Звуковий супровід	13
4.2	Вимоги до надійності	13
4.3	Умови експлуатації.....	13
4.3.1	Вид обслуговування	14
4.3.2	Обслуговуючий персонал.....	14
4.4	Вимоги до складу і параметрів технічних засобів	14
4.5	Вимоги до інформаційної та програмної сумісності	14
4.5.1	Вимоги до вхідних даних	14
4.5.2	Вимоги до вихідних даних	15
4.5.3	Вимоги до мови розробки.....	15
4.5.4	Вимоги до середовища розробки.....	15
4.5.5	Вимоги до представленню вихідних кодів	15
4.6	Вимоги до маркування та пакування	15
4.7	Вимоги до транспортування та зберігання	15
4.8	Спеціальні вимоги.....	15
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	16
5.1	Попередній склад програмної документації	16

5.2	Спеціальні вимоги до програмної документації.....	16
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ	17
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	18

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Гра у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою.

Галузь застосування: Індустрія розваг.

Наведене технічне завдання поширюється на розробку гри у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою RTS Game, котра використовується для здобуття ігрового досвіду у жанрі RTS та призначена для широкого кола користувачів у сфері розваг, що прагнуть розважитися та застосувати свої навички для контролю військами, ресурсами на безпосередньому, тактичному та стратегічному рівнях.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки гри у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для розвитку стратегічного та тактичного мислення під час розваг.

Метою розробки є досягнення високого рівня залученості гравців за рахунок реалізації та оптимізації механік гри, досягнення високих показників продуктивності, у першу чергу частоти кадрів, достатніх для плавного ігрового процесу; забезпечення високої гнучкості розробки складних систем із багатократними зв'язками шляхом застосування подійно-орієнтованої архітектури.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

— головне меню з можливістю переходу в меню вибору рівня, меню завантаження збереження та можливістю вийти з гри, рисунок 4.1;

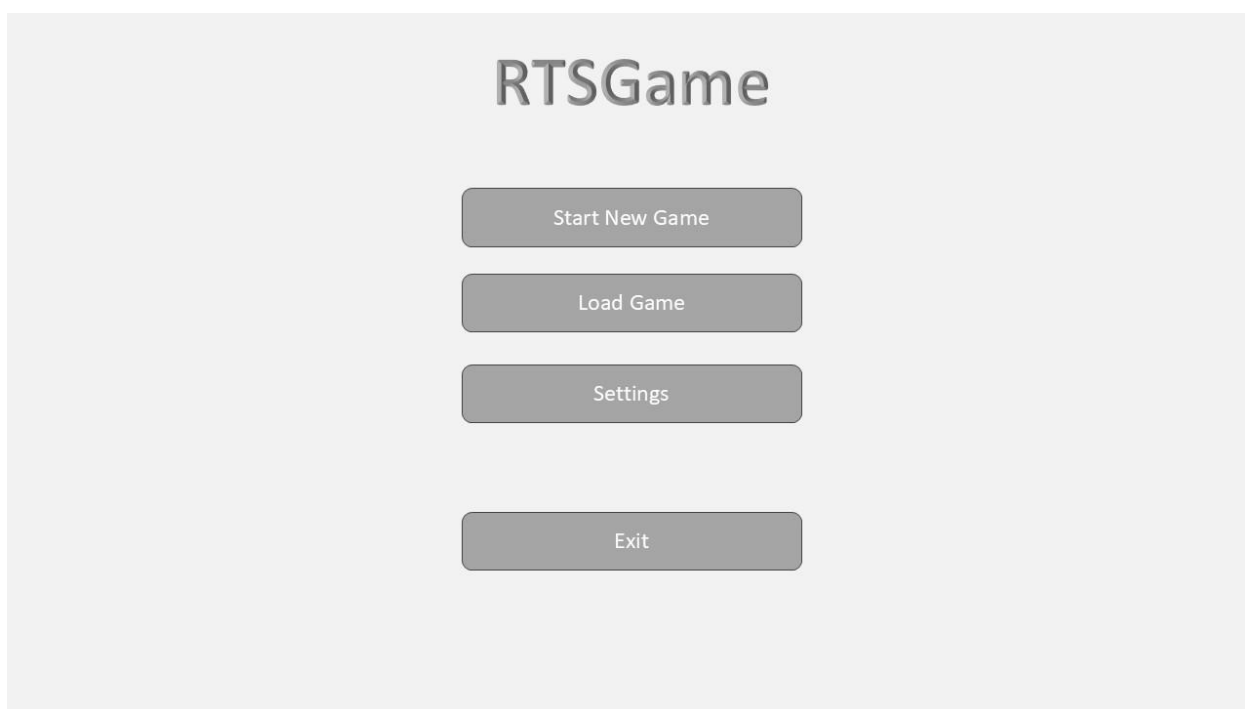


Рисунок 4.1 – Головне меню

— меню завантаження збереження, у якому здійснюється вибір збереження зі списку, завантаження, видалення збереження та вихід до попереднього меню, рисунок 4.2;

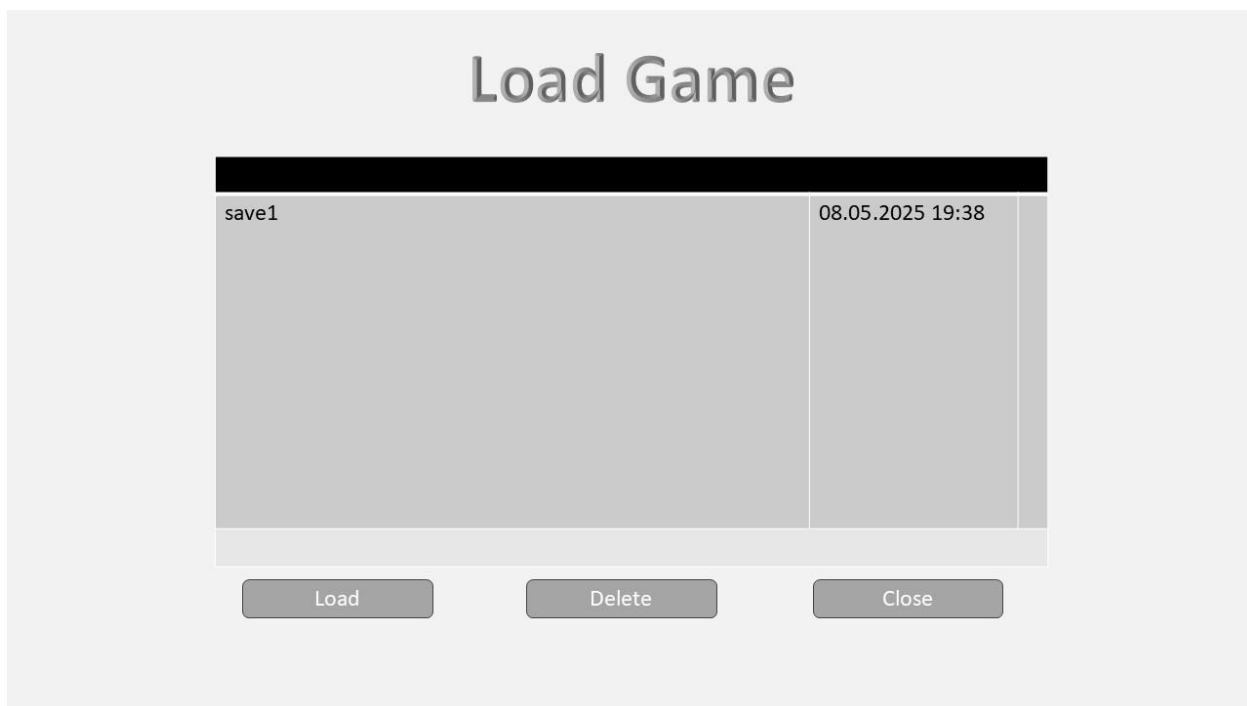


Рисунок 4.2 – Меню завантаження збереження

– меню створення нової гри, що передбачає вибір рівня, початок нової гри та повернення до попереднього меню, рисунок 4.3;



Рисунок 4.3 – Меню створення нової гри

– ігровий інтерфейс, передбачає відображення кількості ресурсів у верхній частині екрану, можливість відкрити меню паузи кнопкою у правому верхньому куті, динамічне відображення інформації про виділені юніти та

будівлі та кнопок можливих команд для виділених об'єктів у лівому нижньому куті, відображення мінікарти у правому нижньому куті, рисунок 4.4;

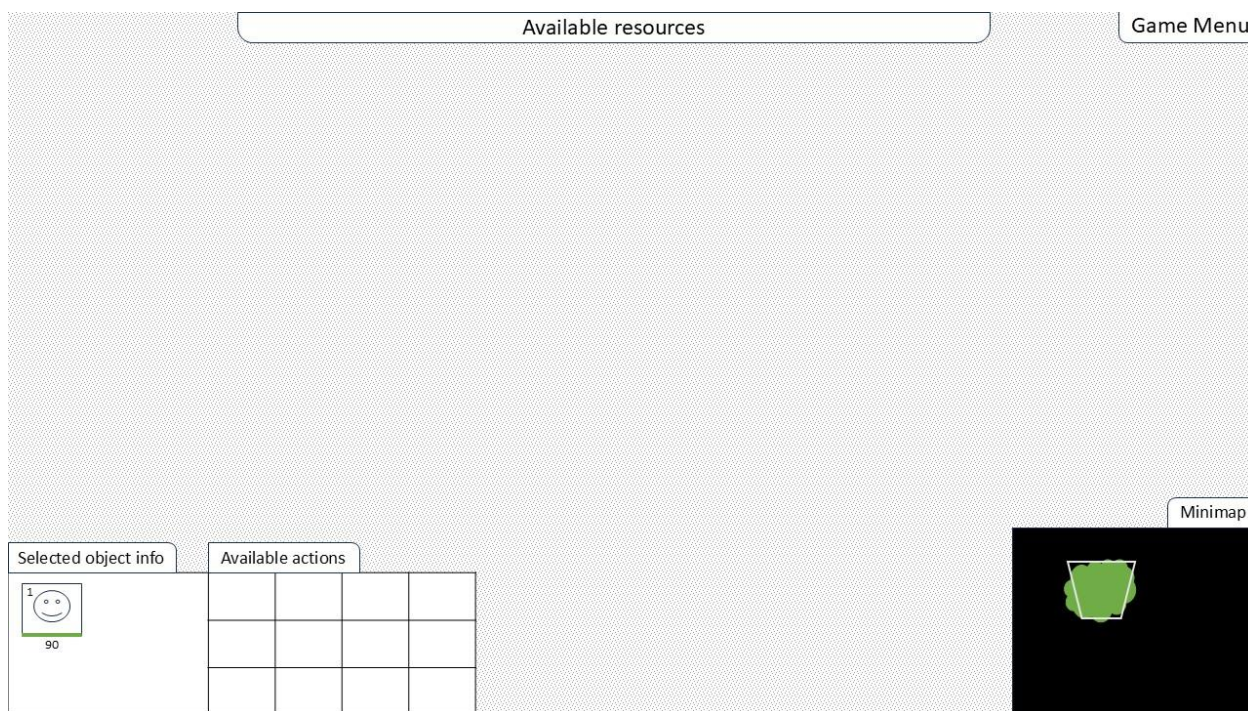


Рисунок 4.4 – Ігровий інтерфейс

– меню паузи, що передбачає продовження гри, перехід у меню збереження та завантаження, перехід у головне меню, рисунок 4.5;

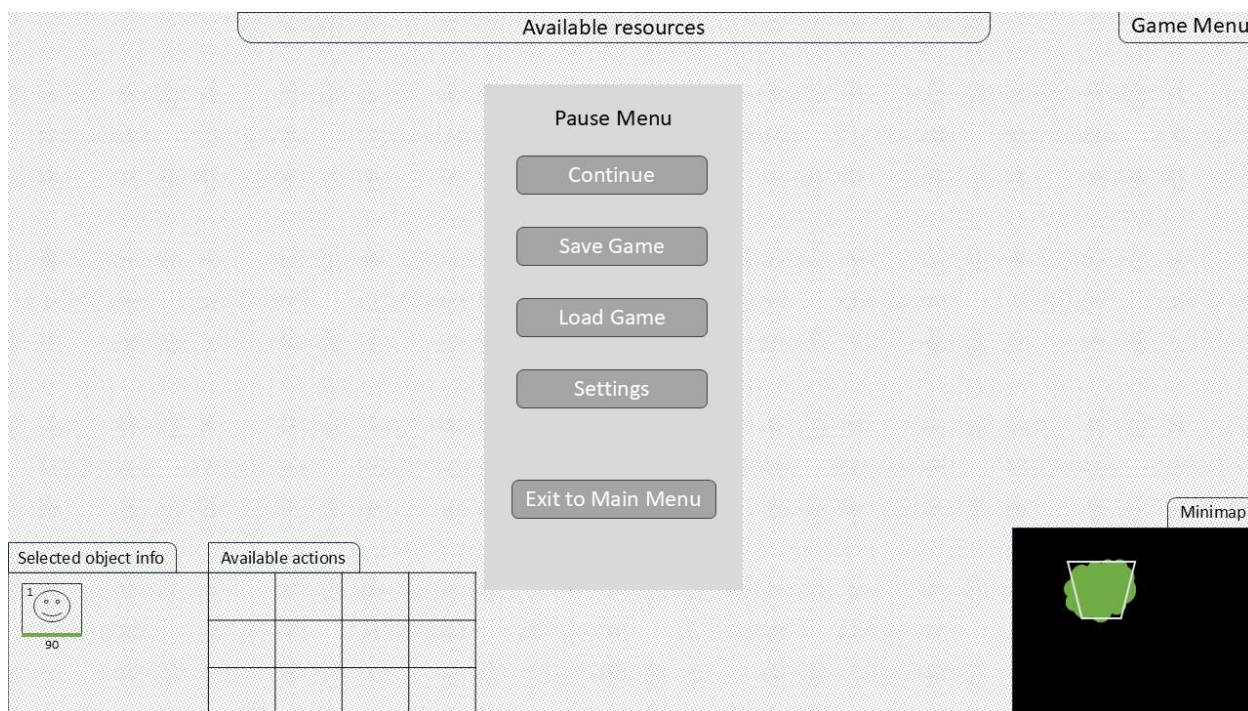


Рисунок 4.5 – Меню паузи

– меню збереження гри, що передбачає виведення списку наявних збережень, введення назви нового збереження, кнопок створення та видалення збереження, кнопки повернення до попереднього меню, рисунок 4.6.

Save Game

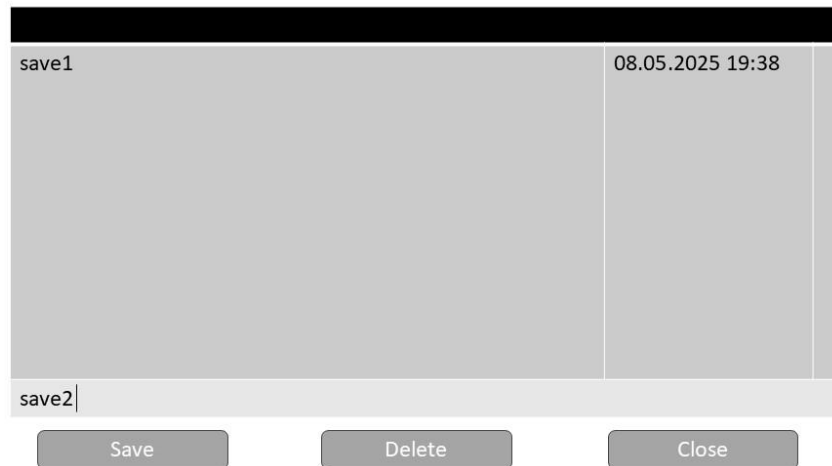


Рисунок 4.6 – Меню збереження

4.1.2 Система керування під час ігрового процесу:

- забезпечити вигляд камери згори;
- забезпечити рух камерою при переміщенні курсору до краю екрану, наприклад якщо курсор перемістити до верхнього краю екрану, то камера буде переміщуватися на північ по карті;
- забезпечити виділення об'єктів за допомогою натискання та утримування лівої кнопки миші;
- забезпечити керування виділеними об'єктами за допомогою кнопок ігрового інтерфейсу та натискання правої кнопки миші по об'єктам та місцю на карті.

4.1.3 Керування юнітами:

4.1.3.1 Забезпечити характеристики юніта:

- кількість одиниць здоров'я;

- кількість одиниць шкоди, що може наносити іншому об'єкту;
- дальність атаки;
- швидкість руху;
- належність до гравця чи ворога.

4.1.3.2 Забезпечити різні види юнітів:

- робітник – добування ресурсів та ремонт будівель, не може атакувати інші юніти;
- піхотинець – основний військовий юніт з ближньою атакою;
- стрілець – військовий юніт з дальньою атакою;
- вершник – військовий юніт з ближньою атакою та високою мобільністю.

4.1.3.3 Забезпечити можливість виконання наказів юнітами:

- виділення юнітів, яким будуть надаватися накази, за допомогою натискання лівої кнопки миші на юніта або утримування лівої кнопки миші і обведення прямокутником юнітів;
- наказ на рух юнітів на задану позицію;
- для робітників – наказ на добування ресурсів, який буде виконуватися циклічно: юніт підходить до ресурсу, добуває ресурс, йде до точки збереження ресурсів і знову повертається до ресурсу і повторює цей процес;
- для робітників – наказ на ремонт будівлі: юніт підходить до будівлі і ремонтує її до повної починки;
- для військових юнітів – наказ на атаку ворога: юніт переслідує ворога і при наближенні до радіуса атаки наносить шкоду;
- можливість перервати попередній наказ іншим наказом.

4.1.4 Керування будівлями:

4.1.4.1 Забезпечити характеристики будівлі:

- кількість одиниць здоров'я;
- належність до гравця чи ворога.

4.1.4.2 Забезпечити різні види будівель:

- ратуша – будівля для збереження ресурсів та створення нових робітників;
- казарма – будівля для створення нових військових юнітів;
- будинок – будівля для збільшення максимальної кількості юнітів гравця.

4.1.4.3 Забезпечити можливість керування будівлями:

- виділення будівель за допомогою натискання лівої кнопки миші на будівлю або утримування лівої кнопки миші і обведення прямокутником будівлі;
- створення нових юнітів за ресурси шляхом натискання відповідної кнопки ігрового інтерфейсу, якщо таку можливість передбачає вид будівлі.

4.1.5 Керування ресурсами:

4.1.5.1 Забезпечити різні види ресурсів:

- максимальна кількість юнітів гравця;
- золото.

4.1.5.2 Забезпечити можливість накопичення та використання ресурсів:

- збільшення максимальної кількості юнітів за допомогою будівель;
- добування та перенесення до ратуші золота робітниками;
- витрата золота на створення нових юнітів;
- відображення кількості ресурсів у верхній частині ігрового інтерфейсу.

4.1.6 Місії для проходження:

4.1.6.1 Забезпечити вміст місії:

- ігрове поле;
- початкові позиції будівель, юнітів гравця та супротивника, ресурсів;

- ціль для виконання гравцем.

4.1.6.2 Забезпечити роботу супротивника:

- керування військовими юнітами супротивника ігровим штучним інтелектом, що перешкоджатиме виконанню місії гравцем.

4.1.7 Збереження гри:

4.1.7.1 Забезпечити збереження прогресу гри:

- поточна кількість ресурсів;
- поточний стан будівель;
- поточний стан та накази юнітів.

4.1.7.2 Забезпечити завантаження прогресу гри:

- відновлення усіх збережених даних та запуск рівня.

4.1.7.3 Забезпечити роботу меню збереження та завантаження:

- перегляд списку збережень;
- збереження за назвою або завантаження;
- видалення виділеного збереження.

4.1.8 Звуковий супровід:

- забезпечити звукові ефекти на різні події в грі;
- забезпечити систему шарів фонової музики: шари музики повинні відтворюватися залежно від подій у грі.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача.

Забезпечити стабільність роботи без збоїв та зависань при тривалій грі та заданих системних вимогах.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на персональних комп'ютерах архітектури x64.

Мінімальна конфігурація технічних засобів:

- тип процесору: 4-ядерний процесор 2.5 ГГц;
- об'єм ОЗП: 8 Гб;
- відеокарта: NVIDIA GeForce GTX 960;
- місце на жорсткому диску: 1 Гб;
- миша та клавіатура.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7-10750H;
- об'єм ОЗП: 16 Гб;
- відеокарта: NVIDIA GeForce GTX 1060 Ti.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційної системи Windows 10/11 x64.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: збереження гри, що містять інформацію про стан юнітів, будівель та ресурсів.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: збереження гри, що містять інформацію про стан юнітів, будівель та ресурсів.

4.5.3 Вимоги до мови розробки

Розробку виконати інструментами системи візуального скриптування Blueprints та на мові програмування C++.

4.5.4 Вимоги до середовища розробки

Розробку виконати на ігровому рушії Unreal Engine та в середовищі розробки Visual Studio.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторію на Github.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна компонентів програмного забезпечення;
- схема структурна класів програмного забезпечення.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	15.03	
2.	Розробка технічного завдання	11.04	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	23.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	10.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	15.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	26.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	28.05	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

**Пояснювальна записка
до дипломного проєкту**

на тему: **Гра у жанрі стратегії в реальному часі з подійно-орієнтованою
архітектурою**

КПІ.ПІ-1530.045490.02.81

Київ – 2025

ЗМІСТ

ВСТУП	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	7
1.1 Постановка завдання дипломного проєктування	7
1.2 Аналіз предметної області	7
1.3 Аналіз існуючих рішень	10
1.3.1 Аналіз відомих програмних продуктів	10
1.3.2 Аналіз відомих алгоритмічних та технічних рішень	15
1.4 Аналіз та моделювання бізнес-процесів	16
Висновки до розділу	21
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	22
2.1 Варіанти використання програмного забезпечення	22
2.2 Розроблення функціональних вимог	29
2.3 Розроблення нефункціональних вимог	33
2.4 Аналіз системних вимог	33
2.5 Аналіз економічних показників програмного забезпечення	34
2.6 Постановка завдання на розробку програмного забезпечення	38
Висновки до розділу	39
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	41
3.1 Архітектура програмного забезпечення	41
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	43
3.3 Конструювання програмного забезпечення	45
3.3.1 Опис ігрових об'єктів	46
3.3.2 Опис допоміжних об'єктів	49
3.4 Аналіз безпеки даних	50
Висновки до розділу	50
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	52
4.1 Аналіз якості ПЗ	52

4.2	Опис процесів тестування.....	53
4.3	Опис контрольного прикладу	61
	Висновки до розділу	69
5	РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	71
5.1	Розгортання програмного забезпечення.....	71
5.2	Супровід програмного забезпечення	74
	Висновки до розділу	74
	ВИСНОВКИ.....	75
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	76
	ДОДАТОК А ЗВІТ ПОДІБНОСТІ.....	79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
RTS	– Real-time strategy – стратегія в реальному часі.
IT	– Інформаційні технології.
OC	– Операційна система.

ВСТУП

Розробка відеоігор є однією з найбільш динамічних галузей ІТ. Жанр стратегій у реальному часі є давно усталеним на ринку та має стабільну аудиторію. Ігри цього жанру передбачають керування армією та будівлями для знищення супротивника чи виконання поставлених завдань. Користувачі можуть розвивати свої навички керування ресурсами, стратегічного мислення та тактичного планування.

Актуальність розробки полягає не лише у розважальному аспекті, а й у складності реалізації ефективної архітектури програмного забезпечення, оптимізації взаємодії між численними ігровими об'єктами та реалізації штучного інтелекту.

Провідні ігрові компанії, такі як Blizzard Entertainment та Relic Entertainment, створили відомі проекти жанру та мають значні напрацювання у напрямку штучного інтелекту, мережевої синхронізації, оптимізації продуктивності тощо. Проте більшість їхніх рішень залишаються комерційною таємницею. У зв'язку з цим аналіз провідних тенденцій базується переважно на зовнішніх спостереженнях, публічних презентаціях, фрагментах виступів розробників на конференціях. Дослідники та ентузіасти змушені створювати власні проекти або працювати з відкритими аналогами, що намагаються відтворити класичний ігровий процес з відкритим вихідним кодом.

У рамках дипломного проекту буде розроблено гру в жанрі стратегії в реальному часі, що матиме базовий ігровий процес керування юнітами та будівлями, рівні для проходження, збереження та завантаження прогресу гри, програвання аудіо.

Розробка призначена для здобуття ігрового досвіду у жанрі RTS та розвитку стратегічного й тактичного мислення під час розваг. Розробка також може бути використана для досліджень у галузі ігрового штучного інтелекту, створення нових систем гри або освітніх цілей через відкритий програмний код.

Метою дипломного проєкту є досягнення високого рівня залученості гравців за рахунок реалізації та оптимізації механік гри, досягнення високих показників продуктивності, у першу чергу частоти кадрів, достатніх для плавного ігрового процесу; забезпечення високої гнучкості розробки складних систем із багатократними зв'язками шляхом застосування подійно-орієнтованої архітектури.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

- аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;
- аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;
- аналіз та моделювання бізнес-процесів;
- розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;
- аналіз економічних показників програмного забезпечення ДП;
- постановка завдання на розробку програмного забезпечення ДП;
- розроблення архітектури програмного забезпечення;
- розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;
- конструювання та розроблення програмного забезпечення;
- аналіз якості та тестування програмного забезпечення;
- розгортання та супровід програмного забезпечення;
- створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

Ігри в жанрі стратегії в реальному часі (RTS) — це спрощені військові симуляції. У грі жанру RTS гравець непрямо керує багатьма юнітами та будівлями, віддаючи накази з висоти пташиного польоту в реальному часі, щоб збирати ресурси, споруджувати будівлі та армію, а також знищувати сили супротивника. Юнітами називають робітників та військові одиниці, якими керує гравець чи ігровий штучний інтелект. Реальний час у таких іграх означає,

що гравці не ходять по черзі, а натомість можуть виконувати стільки дій, скільки фізично здатні, тоді як симуляція гри відбувається з певною частотою кадрів, щоб наблизити безперервний потік часу. До відомих ігор жанру RTS належать Dune II, Total Annihilation та серії Warcraft, Command & Conquer, Age of Empires і StarCraft [1].

Загалом, кожен матч у грі жанру RTS починається з декількома юнітами та/або будівлями в різних місцях на двовимірній місцевості (карті). Навколишні ресурси можна збирати, щоб виробляти додаткові юніти. У більшості ігор жанру RTS матч закінчується, коли один гравець знищує всі будівлі, що належать супернику [1].

Ігри жанру RTS мають різноманітні військові юніти, якими гравці ведуть війну, а також юніти та будівлі для допомоги в збиранні ресурсів, виробництві одиниць і покращеннях. Під час матчу гравці повинні збалансувати розвиток своєї економіки та армії, щоб вижити в поточний момент та досягнути успіху в майбутньому. Довгострокове планування та прийняття рішень на високому рівні часто називається макроменеджментом. Крім стратегічного прийняття рішень, гравці повинні ретельно контролювати свої юніти, щоб максимально підвищити їхню ефективність на полі бою. Групи юнітів можна переміщувати у вигідні позиції на карті, щоб оточити або уникнути ворога, а окремі юніти можна контролювати, щоб атакувати слабкого ворога або уникнути наступаючої атаки. Короткостроковий контроль і прийняття рішень з окремими юнітами часто називають мікроменеджментом. Середньострокове планування з групами юнітів називається тактикою [1].

Прикладом успішної гри у жанрі RTS є StarCraft, який став кіберспортивною грою. Гра дотримується більшості умов жанру. Гравці повинні створити базу та збирати ресурси для створення військових юнітів і ведення своїх армій у бій. Гра використовує мишу та клавіатуру, гравці натискають на одиниці або будівлі, щоб віддавати накази, які будуть виконані автоматично. Юніти та будівлі мають різні характеристики (кількість одиниць здоров'я, показник атаки, броня тощо) і спеціальні здібності, які впливають на

результати бою. Кожне посилення в грі впорядковане в «технологічному дереві»: окрім певної кількості ресурсів, потрібних для будівництва певних будівель, є передумовою для нових одиниць, будівель, технологій та покращень. Стандартні ігри закінчуються, коли гравець (або команда союзних гравців) знищує всі будівлі свого суперника [2].

Серед недоліків ігор у жанрі RTS можна навести високий поріг складності для новачків, потреба у постійній концентрації на керування окремими юнітами, обмежена доступність для гравців на гральних консолях через орієнтованість керування на клавіатуру та мишу, відсутність інновацій ігрових механік. Шляхами покращення можуть бути розробка більш інтуїтивних інтерфейсів та навчальних режимів, введення механізмів автоматизації деяких процесів, наприклад збору ресурсів; експериментування з новими механіками.

Для дипломного проєкту було обрано проблему складності розробки взаємопов'язаних компонентів гри та закритого програмного коду сучасних проєктів. Викликами при розробці гри у жанрі стратегії в реальному часі є:

- складність ігрових механік, зокрема керування ресурсами, юнітами, будівлями;
- складність тестування для забезпечення балансу та чесності гри;
- обмежена привабливість на ринку для гравців через недоліки, наведені раніше;
- складність оновлення та підтримки після випуску гри.

Для більш ефективної та гнучкої розробки у рамках дипломного проєкту запропоновано застосування подійно-орієнтованої архітектури, що використовується для створення високомасштабованих і високопродуктивних додатків. Вона також дуже адаптивна і може використовуватися як для невеликих додатків, так і для великих складних систем [3]. Такі характеристики є важливими для підтримання сталої частоти кадрів для плавності гри, а також для простоти додавання механік гри.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації гри у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

Для порівняння дипломного проєкту було обрано ігри «Age of Empires IV», «Warcraft III: Reforged» та «0 A.D.».

«Age of Empires IV» – історична стратегія в реальному часі, остання частина відомої серії стратегій в реальному часі та є одним з найяскравіших представників жанру останніх років (рисунок 1.1) [4].



Рисунок 1.1 – Гра «Age of Empires IV»

«Warcraft III: Reforged» – оновлена версія культової фентезійної гри «Warcraft III», що мала значний вплив на жанр стратегій в реальному часі. Гравець може грати за декілька фракцій з унікальними сюжетними лініями (рисунок 1.2) [5].



Рисунок 1.2 – Гра «Warcraft III: Reforged»

«0 A.D.» – безкоштовна стратегія в реальному часі, єдина гра з переліку з відкритим вихідним кодом. Юніти, архітектура та технології засновані на реальних історичних джерелах (рисунок 1.3) [6].



Рисунок 1.3 – Гра «0 A.D.»

Зі спільного між цими іграми та дипломним проєктом можна виділити реалізацію основних механік гри, зокрема керування юнітами та будівлями, керування камерою для переміщення по ігровій мапі, наявність однокористувачького режиму, можливість зберігати та завантажувати прогрес гри.

Порівняння дипломного проєкту з аналогами викладено в таблиці 1.1.

Таблиця 1.1 – Порівняння з аналогами

Критерій	Дипломний проєкт RTS Game	Age of Empires IV	Warcraft III Reforged	0 A.D.	Пояснення
Відкритий програмний код	+	-	-	+	Ігри зазвичай є комерційними кінцевими продуктами
Сучасна графіка	+	+	-	-	У дипломному проєкті використано рушій Unreal Engine 5, що використовується для реалістичних ігор
Динамічне відображення інтерфейсу	+	+	-	+	Приховування елементів інтерфейсу, що не використовуються в поточний момент

Продовження таблиці 1.1

Критерій	Дипломний проект RTS Game	Age of Empires IV	Warcraft III Reforged	0 A.D.	Пояснення
Шари фонові музики	+	-	-	-	Шари музики, що відтворюються залежно від подій у грі
Подійно- орієнтована архітектура	+	-	-	-	Хоча деякі ігри з порівняння мають закритий код, такий висновок можна зробити із публічних заяв та керівництв для модифікації ігор
Огляд докумен- товані архітектури для навчальних цілей	Поясню- вальна записка з діаграмами і роз'ясне- ннями	-	-	Велика доку- мента- ція без графіч- них схем	Дипломний проект має нижчий поріг входу для огляду у навчальних цілях

Продовження таблиці 1.1

Критерій	Дипломний проект RTS Game	Age of Empires IV	Warcraft III Reforged	0 A.D.	Пояснення
Можливе використа- ння для досліджень	+	-	-	+	Як приклад, розробка штучного інтелекту, який має доступ до систем гри
Ціна	0	Від 699 гривень	745 гривень	0	«Age of Empires IV» також має платні доповнення
Розмір команди розробників	1	≈1500	≈30	≈2600	Для «Warcraft III Reforged» узято кількість розробників оригіналу
Рушій	Unreal Engine 5	Essence Engine 5.0	Власний	Pyro genesis	Усі альтернативи розроблені на спеціально розроблених рушіях

Продовження таблиці 1.1

Критерій	Дипломний проект RTS Game	Age of Empires IV	Warcraft III Reforged	0 A.D.	Пояснення
Платформи окрім Windows	Можливо скомпільовати для Linux, macOS, Android, iOS, Xbox, PS, Nintendo Switch та інші	Xbox Series X/S, Xbox One	macOS	Linux, macOS, Free BSD, Open BSD	Для платформ, що не використовують клавіатуру та мишу, потрібно розробити іншу систему керування

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

У стратегіях в реальному часі активно використовується ігровий штучний інтелект. Серед основних методів можна перелічити такі, як:

- скінченний автомат – використовується для закодування стратегій у комерційних RTS-іграх, виділяються окремі стани, наприклад "атака", "збір ресурсів" або "ремонт" з переходами між ними, ці методи ефективні, але вони мають обмежену адаптивність і можуть бути легко використані суперниками;
- дерево поведінки – структура даних, що складається з вузлів, кожен з яких представляє окрему дію або умову;
- машинне навчання – навчання ігрового інтелекту на основі даних попередніх проходжень гри;
- міркування на основі прецедентів (case-based reasoning) – підпадає в категорію машинного навчання, використовується для динамічного вибору плану дій [7].

Для дипломного проєкту було обрано скінченний автомат для керування юнітами гравця, адже сам гравець задає перехід між станами за допомогою наказів і юніти потім виконують цей наказ. Для керування ворогом було обрано дерево поведінки, адже це простий спосіб задати дії та умови всередині місій, що буде проходити гравець. Такий метод не підійде для створення повноцінного ігрового ШІ супротивника, що діятиме так само, як і гравець, але він підходить для поставлених задач та не є вимогливим до обчислювальних ресурсів.

Дипломний проєкт не є веб-застосунком і не вимагає наявності окремого програмного забезпечення крім самої гри, тому не потрібна наявність мікросервісної, клієнт-серверної чи інших подібних архітектур. Протиставленням до розподілених систем є монолітна архітектура – єдина програма, що не з'єднана через протоколи віддаленого доступу [3].

Подійно-орієнтована архітектура традиційно вважається розподіленою і складається з роз'єднаних компонентів обробки подій, які асинхронно отримують і обробляють події [3]. У рамках дипломного проєкту події будуть здійснюватися між компонентами всередині гри. Подійно-орієнтована архітектура вважається високопродуктивною та масштабованою [3], що є важливим для підтримання сталої частоти кадрів для плавності гри, а також для простоти додавання механік гри.

У якості платформи обрано ОС Windows через специфіку керування стратегій у реальному часі мишою та клавіатурою [2] та велику кількість користувачів, відповідно до опитування Steam у квітні 2025 року [8], 57% користувачів мають ОС Windows 11 64 bit та 38% – Windows 10 64 bit.

1.4 Аналіз та моделювання бізнес-процесів

Для моделювання загального бізнес-процесу гри побудовано BPMN модель (рисунок 1.4).

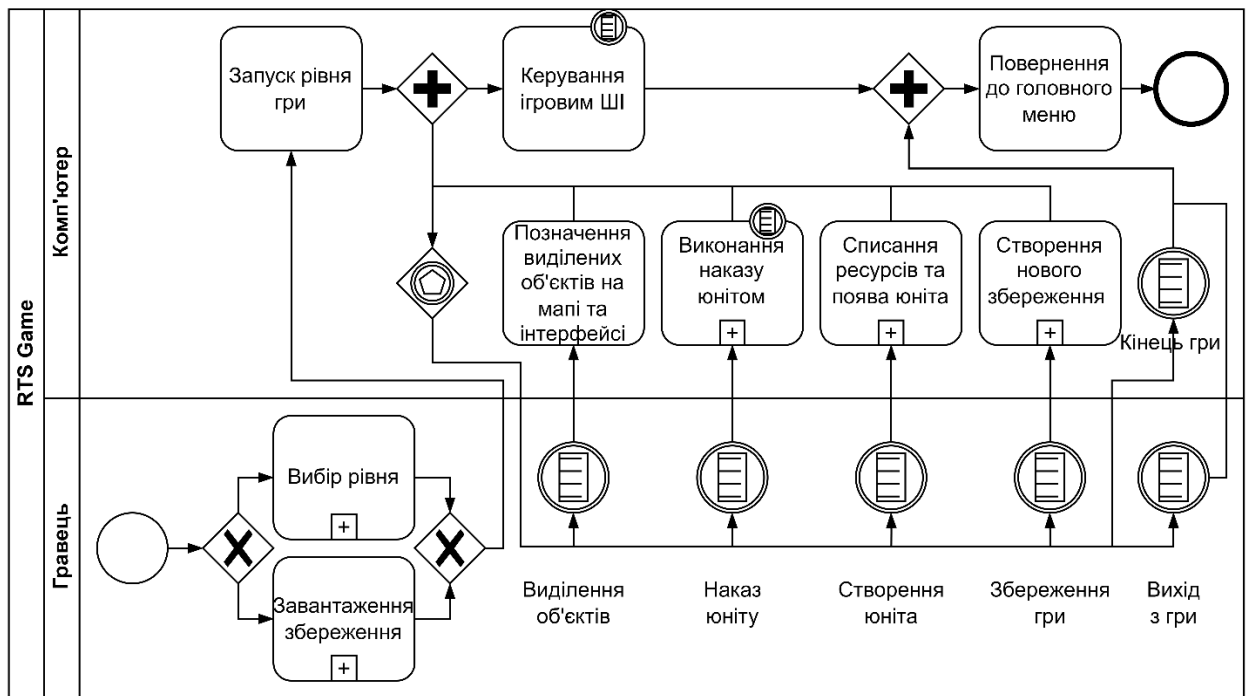


Рисунок 1.4 – Модель бізнес-процесу гри

Опис моделі загального бізнес-процесу гри:

- гравець обирає між вибором рівня та завантаженням збереженого прогресу гри;
- після вибору рівня чи завантаження збереження відбувається запуск рівня гри та початок ігрового процесу;
- супротивник починає керувати своїми юнітами і гра очікує на дії користувача;
- якщо гравець виділив об'єкти, то ці об'єкти повинні відобразитися списком на ігровому інтерфейсі;
- якщо гравець здійснив наказ юнітам, то гра повинна забезпечити виконання наказу цими юнітами;
- якщо гравець створить юніта, то повинні списатися ресурси і з'явитися новий юніт;
- якщо гравець хоче зберегти гру, то повинно створитися нове збереження;
- якщо досягнені умови завершення гри або якщо гравець хоче вийти із гри, то відбувається перехід до головного меню.

Підпроцес вибору рівня зображено на рисунку 1.5.

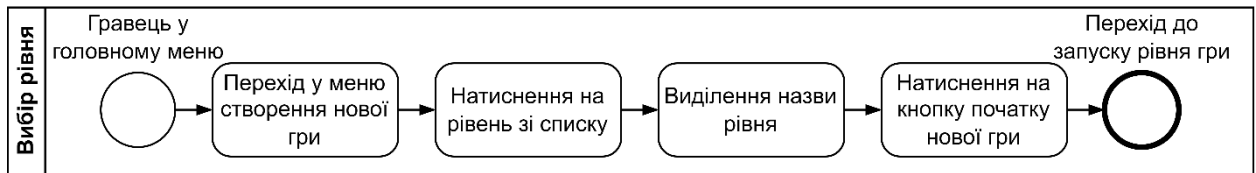


Рисунок 1.5 – Підпроцес вибору рівня

Опис підпроцесу вибору рівня:

- гравець переходить у меню створення нової гри;
- гравець натискає на рівень зі списку;
- назва рівня стає виділеною;
- гравець натискає на кнопку початку нової гри;
- здійснюється перехід до запуску рівня гри.

Підпроцес завантаження збереження зображено на рисунку 1.6.

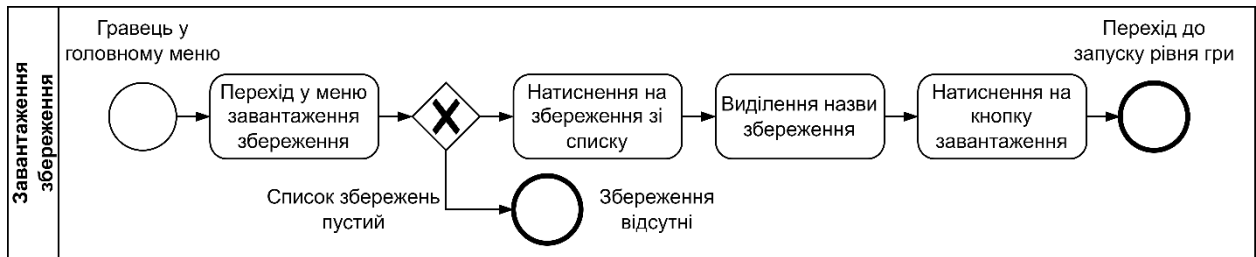


Рисунок 1.6 – Підпроцес завантаження збереження

Опис підпроцесу завантаження збереження:

- гравець переходить у меню створення нової гри;
- якщо список збережень пустий, то користувач не зможе запустити ігровий процес через завантаження збереження;
- гравець натискає на збереження зі списку;
- назва збереження стає виділеною;
- гравець натискає на кнопку завантаження;
- здійснюється перехід до запуску рівня гри.

Підпроцес виконання наказу юнітом зображено на рисунку 1.7.

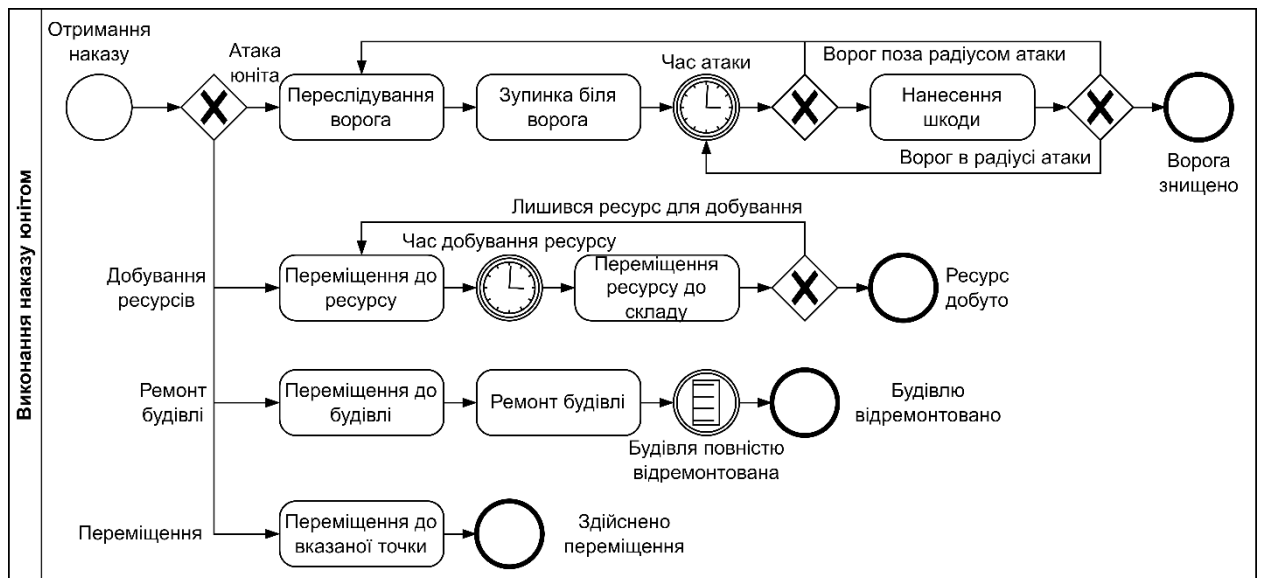


Рисунок 1.7 – Підпроцес виконання наказу юнітом

Опис підпроцесу виконання наказу юнітом:

На рисунку 1.4 позначено, що підпроцес може бути перерваним, це можливо здійснити новим наказом юніту, який вже виконує наказ.

Наказ на атаку юніта:

- юніт переслідує ворога;
- юніт зупиняється біля ворога;
- юніт здійснює атаку певний час;
- якщо ворог в радіусі дії, юніт наносить шкоду ворогу;
- якщо ворог залишається в радіусі атаки, то юніт повторно здійснює атаку, інакше переслідує ворога;
- наказ діє до знищення ворога.

Наказ на добування ресурсів:

- юніт переміщується до ресурсу;
- юніт певний проміжок часу добуває ресурс;
- юніт переміщується на склад ресурсів;
- якщо ще лишився ресурс, юніт повторно виконує наказ;
- якщо ресурс вичерпано, дія наказу закінчується.

Наказ на ремонт будівлі:

- юніт переміщується до будівлі;

- юніт ремонтує будівлю;
- наказ діє до повної починки будівлі.

Наказ на переміщення:

- юніт переміщується до заданої точки;
- наказ діє до досягнення заданої точки.

Підпроцес списання ресурсів та появи юніта зображено на рисунку 1.8.

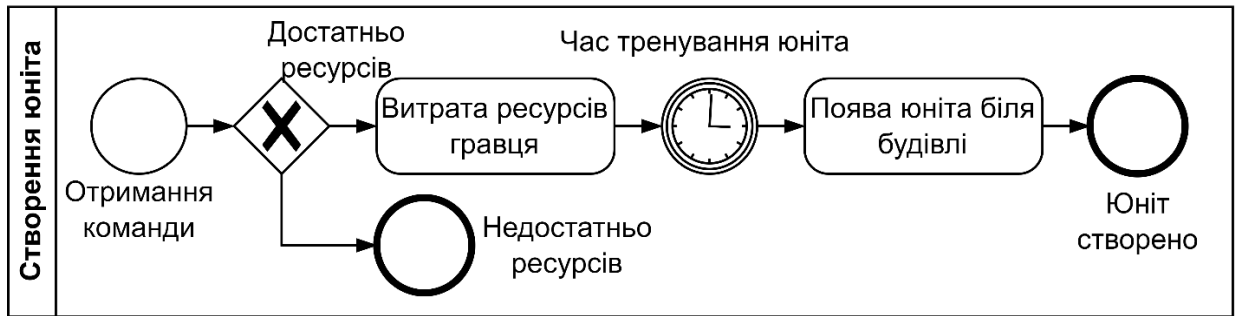


Рисунок 1.8 – Підпроцес списання ресурсів та появи юніта

Опис підпроцесу списання ресурсів та появи юніта:

- підпроцес виконується при достатній кількості ресурсів на створення юніта;
- ресурси гравця витрачаються в розмірі вартості юніта;
- здійснюється тренування юніта певний проміжок часу;
- юніт з’являється біля будівлі.

Підпроцес створення нового збереження зображено на рисунку 1.9.

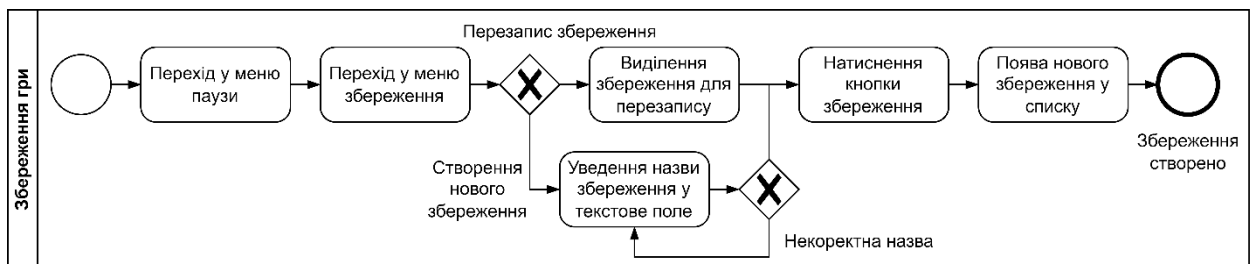


Рисунок 1.9 – Підпроцес створення нового збереження

Опис підпроцесу створення нового збереження:

- гравець переходить у меню паузи;
- гравець переходить у меню збереження;
- гравець може самостійно ввести назву збереження або виділити існуюче збереження для перезапису;

- гравець натискає на кнопку збереження;
- нове збереження з'являється у списку збережень;
- назва збереження повинна бути сумісною з файловою системою Windows.

Висновки до розділу

У даному розділі здійснено передпроектне обстеження предметної області.

Здійснено постановку завдання дипломного проєктування – визначено завдання, які потрібно виконати, зокрема здійснити аналіз предметної області та існуючих рішень, розробити вимоги до програмного забезпечення, розробити архітектуру програмного забезпечення, здійснити конструювання та розробку програмного забезпечення.

Проаналізовано предметну область, зокрема визначено, що ігри в жанрі стратегії в реальному часі є спрощеними військовими симуляціями, в яких гравець керує багатьма юнітами та будівлями на стратегічному, тактичному та ручному рівнях. Головною ціллю гравця є знищення усіх юнітів та будівель супротивника.

Здійснено аналіз існуючих рішень та визначено відмінності дипломного проєкту від аналогів: відкритий програмний код, сучасна графіка, динамічне відображення інтерфейсу, шари фонові музики. Для алгоритмічного наповнення керування юнітами гравцем та ігровим ШІ обрано скінченний автомат та дерево поведінки. У якості архітектури обрано монолітну з елементами подійно-орієнтованої. У якості платформи – операційну систему Windows.

Здійснено моделювання бізнес-процесу гри від запуску до перемоги чи поразки гравця та підпроцесів вибору рівня, завантаження збереження, виконання наказу юнітами, створення юнітів та створення нового збереження.

Продовження таблиці 2.1

Flow of Events	Гравець натискає кнопку початку нової гри, відкривається меню вибору рівня, гравець натискає на назву рівня зі списку, назва рівня виділяється, гравець натискає на кнопку почати нову гру, завантажується рівень, здійснюється перехід на екран ігрового процесу
Extension	-
Post-Condition	Перехід на екран ігрового процесу із завантаженим обраним рівнем

Таблиця 2.2 – Варіант використання UC-02

Use case name	Завантажити збереження
Use case ID	UC-02
Goals	Продовжити гру із збереженого прогресу гри
Actors	Гравець
Trigger	Гравець бажає продовжити гру, прогрес якої зберіг раніше
Pre-conditions	Запущена гра у головному меню
Flow of Events	Гравець натискає кнопку завантаження гри, відкривається меню завантаження, гравець натискає на назву збереження зі списку, збереження виділяється, гравець натискає на кнопку завантажити, завантажується збереження, здійснюється перехід на екран ігрового процесу.
Extension	Список збережень може бути пустим, якщо гравець ні разу не зберігав прогрес гри або видалив усі збереження. Гравець може видалити збереження у меню завантаження після виділення збереження натисканням кнопки видалення. Гравець також може завантажити прогрес гри з меню паузи ігрового екрану.
Post-Condition	Перехід на екран ігрового процесу із завантаженим збереження

Таблиця 2.3 – Варіант використання UC-03

Use case name	Грати
Use case ID	UC-03
Goals	Виконати ігрову місію
Actors	Гравець
Trigger	Гравець перейшов на екран ігрового процесу
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Гравець маніпулює ігровими об'єктами, гравець виконує поставлену місію, гравець перемагає, виводиться меню перемоги з можливістю перезапуску та повернення до головного меню
Extension	У випадку втрати усіх юнітів та будівель гравець програє та виводиться меню поразки
Post-Condition	Виведення меню завершення гри з можливістю перезапуску місії та переходу в головне меню

Таблиця 2.4 – Варіант використання UC-04

Use case name	Керувати камерою
Use case ID	UC-04
Goals	Перемістити камеру на потрібну позицію
Actors	Гравець
Trigger	Гравець бажає перемістити камеру
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Гравець підводить курсор миші під той край екрану, в яку сторону він бажає змістити камеру, камера зміщується, доки гравець не прибере курсор з краю екрану
Extension	Камера не може виходити за межі карти, тому вона перестає переміщатися на межі карти
Post-Condition	Камера перемістилася на нову позицію

Таблиця 2.5 – Варіант використання UC-05

Use case name	Виділяти об'єкти
Use case ID	UC-05
Goals	Виділити ігрові об'єкти
Actors	Гравець
Trigger	Гравець бажає виділити ігрові об'єкти для подальшої маніпуляції над ними
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Користувач утримує ліву кнопку миші, починається відображення прямокутника з одним кутом у місці початку утримання лівої кнопки та іншим з поточною позицією курсора, гравець обводить прямокутником об'єкти, що бажає виділити, гравець відпускає ліву кнопку миші
Extension	Замість утримування миші та виділення прямокутником гравець може натиснути на об'єкт
Post-Condition	Виведення списку виділених об'єктів у нижньому лівому куті ігрового інтерфейсу, підсвічення виділених ігрових об'єктів за допомогою кола навколо них

Таблиця 2.6 – Варіант використання UC-06

Use case name	Давати накази юнітам
Use case ID	UC-06
Goals	Почати виконання наказу юнітами
Actors	Гравець
Trigger	Гравець бажає надати наказ юнітам
Pre-conditions	Запущена гра на екрані ігрового процесу, виділений щонайменше один юніт
Flow of Events	Користувач натискає правою кнопкою миші по місцю на карті, будівлі, ресурсу або ворожому юніту, виділені юніти починають переміщення до вказаної точки чи об'єкту

Продовження таблиці 2.6

Extension	При натисканні по місцю на карті юніти отримують наказ на переміщення. При натисканні по будівлі юніти отримують наказ на ремонт будівлі, що триватиме до повної починки. При натисканні на ресурс юніти здійснюють циклічне добування ресурсу: переміщуються до ресурсу, здійснюють добування, переміщуються до складу ресурсів, вивантажують ресурс, повторюють процес. При натисканні на ворожого юніта юніти починають переслідування і при наближенні до юніта у радіус атаки здійснюють атаку, наказ діє до знищення ворожого юніта. Будь-який наказ можна відмінити будь-яким іншим наступним наказом.
Post-Condition	Юніт починає виконувати наказ

Таблиця 2.7 – Варіант використання UC-07

Use case name	Керувати будівлями
Use case ID	UC-07
Goals	Створити нового юніта
Actors	Гравець
Trigger	Гравець бажає створити нового юніта
Pre-conditions	Запущена гра на екрані ігрового процесу, виділена будівля, що здатна створювати юнітів
Flow of Events	Користувач натискає на кнопку ігрового інтерфейсу у лівому нижньому куті для створення нового юніта, здійснюється списання ресурсів, здійснюється очікування часу тренування, з'являється новий юніт біля будівлі
Extension	Для створення нового юніта повинно бути достатньо ресурсів
Post-Condition	Створення нового юніта

Таблиця 2.8 – Варіант використання UC-08

Use case name	Переглядати інформацію про об'єкт
Use case ID	UC-08
Goals	Перегляд інформації про виділені об'єкти
Actors	Гравець
Trigger	Гравець виділив ігрові об'єкти
Pre-conditions	Запущена гра на екрані ігрового процесу, виділено щонайменше один об'єкт
Flow of Events	Виведення іконок виділених об'єктів та їхньої кількості у лівому нижньому куті ігрового інтерфейсу
Extension	При виділенні лише одного юніта відображається кількість одиниць здоров'я та кількість одиниць шкоди, що може наносити іншому об'єкту При виділенні юнітів одного типу відображається кількість одиниць шкоди, що може наносити іншому об'єкту
Post-Condition	Виведення інформації про виділені об'єкти

Таблиця 2.9 – Варіант використання UC-09

Use case name	Керувати ресурсами
Use case ID	UC-09
Goals	Керування ресурсами гравця
Actors	Гравець
Trigger	Гравець бажає керувати ресурсами
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Виведення ресурсів у верхній частині ігрового інтерфейсу, гравець маніпулює ігровими об'єктами для зміни значення ресурсів.
Extension	Основними типами ресурсів є максимальна кількість юнітів гравця та золото.
Post-Condition	Зміна ресурсів гравця, що відображається у верхній частині ігрового інтерфейсу

Таблиця 2.10 – Варіант використання UC-10

Use case name	Добувати ресурси
Use case ID	UC-10
Goals	Збільшити кількість ресурсів гравця
Actors	Гравець
Trigger	Гравець бажає добути ресурси
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Гравець надає наказ юнітам на добування золота, юніти здійснюють наказ і приносять золото на склад, кількість золота збільшується у верхній частині ігрового інтерфейсу
Extension	Якщо є зруйновані будівлі гравця, гравець може надати наказ юнітам на починку будівлі і після починки збільшиться максимальна кількість юнітів гравця
Post-Condition	Збільшення кількості ресурсу

Таблиця 2.11 – Варіант використання UC-11

Use case name	Створити новий юніт
Use case ID	UC-11
Goals	Витратити зібрані ресурси на нового юніта
Actors	Гравець
Trigger	Гравець бажає витратити зібрані ресурси на створення нового юніта
Pre-conditions	Запущена гра на екрані ігрового процесу, гравець має достатньо ресурсів на створення нового юніта
Flow of Events	Гравець виділяє будівлю, натискає кнопку ігрового інтерфейсу на створення юніта, здійснюється списання ресурсів, через час тренування з'являється новий юніт біля будівлі
Extension	-
Post-Condition	Витрата ресурсів, створення нового юніта

Таблиця 2.12 – Варіант використання UC-12

Use case name	Зберегти прогрес гри
Use case ID	UC-12
Goals	Зберегти поточний прогрес гри
Actors	Гравець
Trigger	Гравець бажає зберегти поточний прогрес гри
Pre-conditions	Запущена гра на екрані ігрового процесу
Flow of Events	Гравець переходить на меню паузи натисканням кнопки у правому верхньому куті ігрового інтерфейсу, натискає на кнопку збереження гри, вводить у поле назву збереження, натискає на кнопку зберегти, нове збереження виводиться у списку збережень
Extension	Нове збереження не створиться, якщо назва пуста або несумісна із файловою системою Windows. При натисканні на збереження зі списку поле введення заповнюється назвою виділеного збереження. Гравець може видалити збереження після виділення збереження натисканням кнопки видалення.
Post-Condition	Створення нового збереження гри

2.2 Розроблення функціональних вимог

У таблиці 2.13 наведено загальну модель вимог, а в таблиці 2.14 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.2.

Таблиця 2.13 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Користувацький інтерфейс	FR-1	Високий	Високий
2	Керування ігровим процесом	FR-2	Високий	Високий
3	Керування юнітами	FR-3	Високий	Високий
3.1	Накази юнітів	FR-4	Високий	Високий
4	Керування будівлями	FR-5	Високий	Високий

Продовження таблиці 2.13

№	Назва	ID Вимоги	Пріоритети	Ризики
5	Керування ресурсами	FR-6	Високий	Високий
6	Місії для проходження	FR-7	Високий	Високий
6.1	Забезпечення роботи супротивника	FR-8	Високий	Високий
7	Керування прогресом гри	FR-9	Середній	Низький
7.1	Збереження прогресу гри	FR-10	Середній	Низький
7.2	Завантаження прогресу гри	FR-11	Середній	Низький
8	Звуковий супровід	FR-12	Низький	Низький
8.1	Звукові ефекти	FR-13	Низький	Низький
8.2	Шари фонової музики	FR-14	Низький	Низький

Таблиця 2.14 – Перелік функціональних вимог

Назва	Опис
FR-1	Користувацький інтерфейс. Гравець може взаємодіяти з грою через ігровий інтерфейс та множину різних меню: головне, створення нової гри, завантаження збереження, паузи, збереження.
FR-2	Керування ігровим процесом. Гравець може керувати ігровим процесом за допомогою клавіатури та миші. Для руху камери потрібно підвести курсор до краю екрану. Виділення об'єктів здійснюється утримуванням або натисканням лівої кнопки миші, виділені об'єкти обводяться колом та відображаються в ігровому інтерфейсі у лівому нижньому куті.
FR-3	Керування юнітами. Гравець може виділяти юніти та маніпулювати ними.
FR-4	Накази юнітів. Гравець може наказати юнітам переміститися, добувати ресурси, ремонтувати будівлю або атакувати ворога.

Продовження таблиці 2.14

Назва	Опис
FR-5	Керування будівлями. Гравець може виділяти будівлі та натиском кнопки ігрового інтерфейсу створювати нових юнітів.
FR-6	Керування ресурсами. Гравець має золото та максимальну кількість юнітів, кількість яких відображається у верхній частині ігрового інтерфейсу. Гравець може витратити ресурси на створення нових юнітів та отримати ресурси шляхом добування золота або починки зруйнованої будівлі.
FR-7	Місії для проходження. Гравець може обрати місію для проходження, що містить ігрове поле, ціль для виконання, початкові позиції будівель, юнітів гравця та супротивника, а також ресурсів.
FR-8	Забезпечення роботи супротивника. Супротивник повинен керувати юнітами та перешкоджати виконанню місії гравця.
FR-9	Керування прогресом гри. Гра має можливість збереження, завантаження та видалення прогресу гри за допомогою відповідних меню збережень, що виводять список збережень.
FR-10	Збереження прогресу гри. Гравець має змогу посеред гри відкрити меню збереження, ввести назву та зберегти прогрес гри, до списку додається збереження із вказаною назвою та часом створення.
FR-11	Завантаження прогресу гри. Гравець має змогу посеред гри або у головному меню відкрити меню завантаження збереження та відновити прогрес гри.

Продовження таблиці 2.14

Назва	Опис
FR-12	Звуковий супровід. Гравець має змогу чути звуковий відгук від гри.
FR-13	Звукові ефекти. Звукові ефекти програватимуться у відгук на події в грі для кращого розуміння та сигналізування.
FR-14	Шари фонової музики. Шари музики повинні програватися відповідно до подій на екрані, спокійна музика без битв і войовнича музика при атаці юнітів.

	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09	UC-10	UC-11	UC-12
FR-1	+	+	+		+		+	+	+	+	+	+
FR-2			+	+	+			+				
FR-3			+			+				+		
FR-4			+			+						
FR-5			+				+				+	
FR-6			+						+	+	+	
FR-7	+		+									
FR-8			+									
FR-9		+										+
FR-10												+
FR-11		+										
FR-12			+			+	+				+	
FR-13			+			+	+				+	
FR-14			+			+						

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Класифікація нефункціональних задач використана на основі посібника з бізнес-аналізу BABOK [9].

Продуктивність:

- гра повинна мати швидкий час відгуку, щоб гравець бачив результат своїх дій миттєво;
- гра повинна оновлюватися у 60 кадрів у секунду на рекомендованій конфігурації технічних засобів.

Надійність:

- гра не повинна закриватися чи зависати від звичайного користування.

Сумісність:

- гра повинна підтримувати різну роздільну здатність екранів.

Модульність:

- код гри повинен бути розділений на окремі модулі, щоб спростити внесення змін або оновлень у майбутньому.

2.4 Аналіз системних вимог

Програмне забезпечення повинно функціонувати на персональних комп'ютерах із ОС Windows 10 64-bit або Windows 11.

Мінімальна конфігурація технічних засобів визначена на основі рекомендацій розробників Unreal Engine [10]:

- тип процесору: 4-ядерний процесор 2.5 ГГц;
- об'єм ОЗП: 8 Гб;
- відеокарта: NVIDIA GeForce GTX 960;
- місце на жорсткому диску: 1 Гб;
- миша та клавіатура.

Рекомендована конфігурація технічних засобів, на якій проводилася розробка гри:

- тип процесору: Intel Core i7-10750H;
- об'єм ОЗП: 16 Гб;
- відеокарта: NVIDIA GeForce GTX 1060 Ti.

2.5 Аналіз економічних показників програмного забезпечення

Аналіз економічних показників здійснюється за допомогою аналізу функціональних точок [3].

Спочатку здійснимо оцінку RET, FTR, DET та складності.

Таблиця 2.15 – Аналіз ILF (Internal Logical Files)

Назва	Опис	DET	RET	Складність
Користувацький інтерфейс	Інтерфейс для взаємодії користувача з системою	20	5	Average
Керування ігровим процесом	Керування основними процесами гри	30	10	High
Керування юнітами	Керування діями юнітів у грі	25	8	High
Керування будівлями	Керування будівлями та їх функціями	22	7	High
Керування ресурсами	Керування ресурсами гри	20	6	Average
Місії для проходження	Керування місіями та завданнями	28	9	High
Керування прогресом гри	Збереження та завантаження прогресу гри	15	4	Average
Звуковий супровід	Керування звуковими ефектами та музикою	10	3	Low

Таблиця 2.16 – Аналіз EI (External Inputs)

Назва	Опис	DET	FTR	Складність
Користувацький інтерфейс	Введення даних користувачем через інтерфейс	20	2	Average
Керування ігровим процесом	Введення даних для керування ігровим процесом	30	5	High
Керування юнітами	Введення даних для керування юнітами	25	4	High
Керування будівлями	Введення даних для керування будівлями	22	3	High
Керування ресурсами	Введення даних для керування ресурсами	20	3	Average
Керування прогресом гри	Введення даних для збереження та завантаження гри	15	2	Average

Таблиця 2.17 – Аналіз EO (External Outputs)

Назва	Опис	DET	FTR	Складність
Користувацький інтерфейс	Виведення даних користувачу через інтерфейс	20	2	Average
Керування юнітами	Виведення даних для керування юнітами	25	4	High
Керування будівлями	Виведення даних для керування будівлями	22	3	High
Керування ресурсами	Виведення даних для керування ресурсами	20	3	Average
Місії для проходження	Виведення даних для керування місіями	28	4	High

Продовження таблиці 2.17

Назва	Опис	DET	FTR	Складність
Керування прогресом гри	Виведення даних для збереження та завантаження гри	15	2	Average
Звуковий супровід	Виведення даних для керування звуковими ефектами та музикою	10	1	Low

Потім здійснюються розрахунки відповідно до таблиці 2.18.

Таблиця 2.18 – Економічні розрахунки

Визначення типу оцінки	Програмний продукт			
Підрахунок кількості функцій в кожній категорії				
	Застосування вагових коефіцієнтів складності			
	Простий	Середній	Складний	Функціональні точки
Кількість введів	0*3	3*4	3*6	30
Кількість виводів	1*4	3*5	3*7	40
Кількість файлів	1*7	3*10	4*15	97
Обчислення ненормованої кількості функціональних точок UFPC				167
Основні характеристики системи (GSC)				
Фактори середовища		Рейтинг (0,1,2,3,4,5)		
Обмін даними		0 – Продукт не обмінюється даними через інтернет, всі дані зберігаються локально		
Розподілена обробка даних		0 – Система не виконує розподілену обробку даних, всі обчислення локальні		

Продовження таблиці 2.18

Основні характеристики системи (GSC)	
Фактори середовища	Рейтинг (0,1,2,3,4,5)
Вимоги до продуктивності	4 – Час відгуку критично важливий для забезпечення плавного геймплею, високі вимоги до продуктивності
Обмеження по апаратним ресурсам	3 – Продукт має значні вимоги до апаратних ресурсів, потребує потужних пристроїв для плавного геймплею
Транзакційне навантаження	2 – Число транзакцій середнє, піки можуть виникати під час активного використання, але спеціальні рішення не потрібні
Інтенсивність взаємодії з користувачем	5 – Вся гра є інтерактивною, користувачі активно взаємодіють з системою
Ергономіка (Ефективність роботи кінцевих користувачів)	4 – Інтерфейс оптимізований для швидкої та ефективної взаємодії з користувачем
Інтенсивність зміни даних (ILF) користувачами	2 – Дані змінюються користувачами рідко, переважно під час збереження прогресу гри
Складність обробки	3 – Обробка середньої складності, вимоги до логіки середні, багатопоточність обмежена
Повторне використання	2 – Продукт розробляється для конкретного сценарію використання, але деякі компоненти можуть бути повторно використані
Зручність інсталяції	4 – Установка проста та автоматична, мінімальні дії з боку користувача
Зручність адміністрування	0 – Адміністрування відсутнє
Використання декількох вузлів (портів)	0 – Використання декількох вузлів не передбачене

Продовження таблиці 2.18

Основні характеристики системи (GSC)	
Фактори середовища	Рейтинг (0,1,2,3,4,5)
Гнучкість	2 – Користувачі можуть налаштовувати певні аспекти гри, але гнучкість обмежена
Обчислення TDI	31
Загальні розрахунки по проєкту	
Обчислення нормуючого фактора VAF	$(31 * 0.01) + 0.65 = 0.96$
Обчислення нормованої кількості функціональних точок AFPC	$167 * 0.96 = 160$
Обчислення оцінки кількості рядків вихідного коду	$160 * 50 = 8000$

2.6 Постановка завдання на розробку програмного забезпечення

Завданням є розробити гру у жанрі стратегії в реальному часі, що реалізує визначену мету шляхом вирішення відповідних задач.

Метою розробки є досягнення високого рівня залученості гравців за рахунок реалізації та оптимізації механік гри, досягнення високих показників продуктивності, у першу чергу частоти кадрів, достатніх для плавного ігрового процесу; забезпечення високої гнучкості розробки складних систем із багатократними зв'язками шляхом застосування подійно-орієнтованої архітектури.

Для досягнення мети необхідно вирішити задачі:

- розробка графічного інтерфейсу, зокрема системи різних меню та ігрового інтерфейсу;
- розробка керування ігровим процесом, зокрема переміщення камери та виділення ігрових об'єктів.

- розробка керування юнітами, що мають виконувати набір різних наказів;
- розробка керування будівлями, що повинні за ресурси створювати юнітів;
- розробка місій для проходження з наявністю супротивника гравця;
- розробка збереження та завантаження прогресу гри.

Висновки до розділу

У розділі описано розроблення функціональних, нефункціональних та системних вимог, а також аналіз економічних показників та постановку завдання на розробку програмного забезпечення.

Визначено 12 варіантів використання програмного забезпечення, що відображені на діаграмі та описані у таблицях. Серед основних можна навести вибір рівня та завантаження збереження, що ведуть до ігрового процесу, а власне ігровий процес включає керування камерою, об'єктами, ресурсами та збереженнями.

Визначено 14 функціональних вимог, що відображені на загальній моделі вимог та детальніше описані в таблиці переліку функціональних вимог. Функціональними вимогами верхнього рівня є керування ігровим процесом, керування юнітами, керування будівлями, місії для проходження, керування збереження.

Визначено нефункціональні вимоги до продуктивності, надійності, сумісності та модульності.

Здійснено аналіз системних вимог, що налічує рекомендації до конфігурації технічних засобів, зокрема процесору, об'єму ОЗП, відеокарти, місцю на жорсткому диску, наявності миші та клавіатури.

Проаналізовано економічні показники за допомогою аналізу функціональних точок.

Здійснено постановку завдання на розробку програмного забезпечення, зокрема визначено мету розробки та задачі для вирішення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Архітектура відображена на рисунку 3.1 за допомогою моделі C4 діаграми 3 рівня, що містить компоненти та зв’язки між ними.

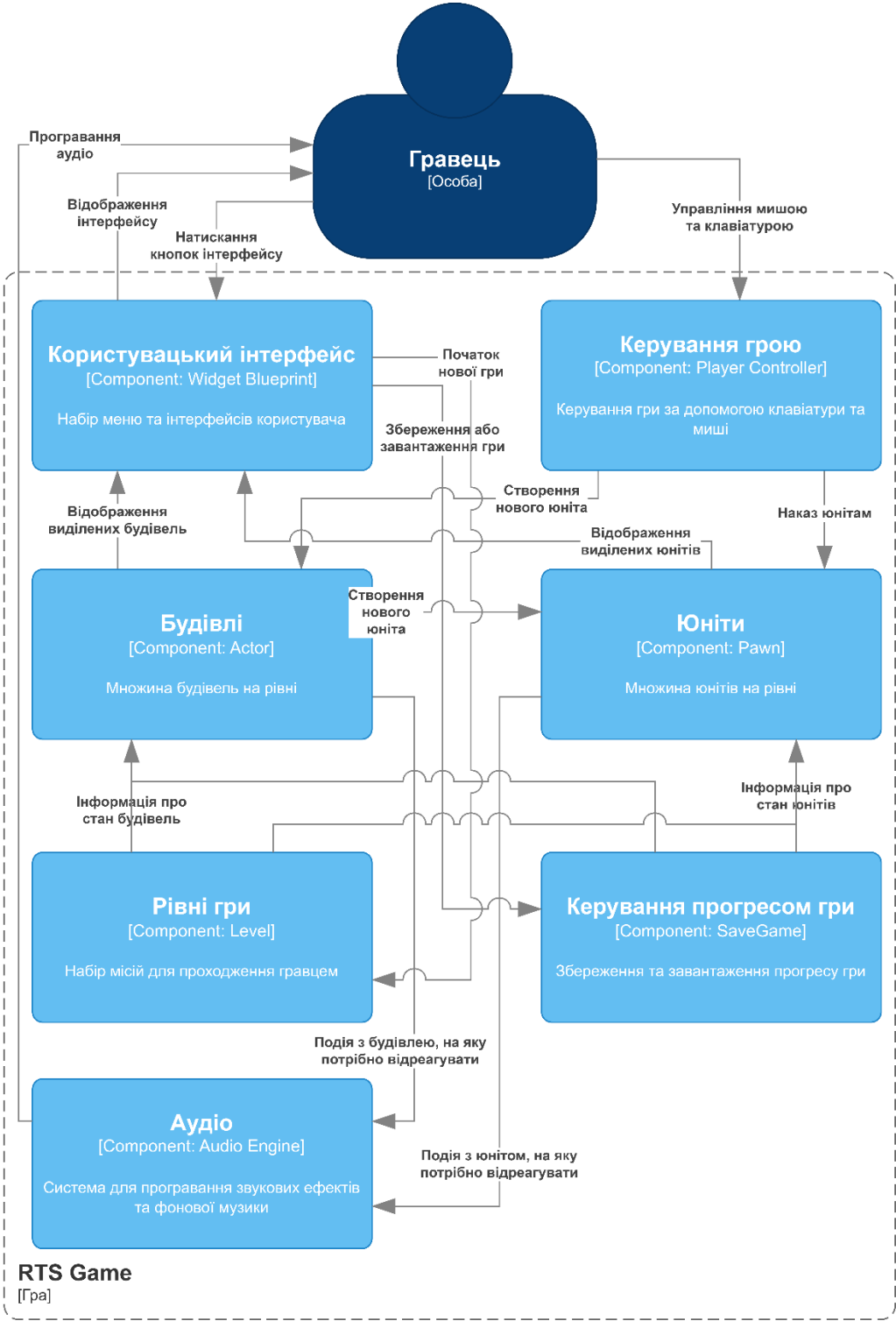


Рисунок 3.1 – Компонентна діаграма моделі C4

Детальний опис зв'язків між компонентами:

Користувацький інтерфейс:

- відображає інформацію гравцю;
- обробляє події натискання кнопок інтерфейсу;
- через різні меню здійснює початок, збереження та завантаження гри;
- на подію виділення юнітів та будівель відображає інформацію про них.

Керування грою:

- обробляє події маніпуляцій миші та клавіатури;
- генерує події наказу юнітів та керування будівлі.

Будівлі:

- обробляє події виділення та керування;
- генерує події на виділення для користувацького інтерфейсу;
- здійснює створення нових юнітів;
- генерує події на програвання звукових ефектів;
- надає інформацію компоненту збережень.

Юніти:

- обробляє події виділення та керування;
- генерує події на виділення для користувацького інтерфейсу;
- генерує події на програвання звукових ефектів;
- отримує команду від будівель на створення юніта;
- надає інформацію компоненту збережень.

Рівні гри:

- містить інформацію про будівлі та юніти;
- реагує на подію початку нової гри від користувацького інтерфейсу.

Керування прогресом гри:

- збирає, зберігає та відновлює інформацію про стан юнітів та будівель;
- реагує на події користувацького інтерфейсу на створення, видалення та завантаження прогресу гри.

Аудіо:

- програє аудіо гравцю;
- обробляє події будівель та юнітів та реагує звуковими ефектами або зміною фонові музики.

Загальна архітектура є монолітною, побудована на принципах подійно-орієнтованого підходу, що забезпечує слабке зв'язування компонентів, спрощує масштабування системи та полегшує тестування.

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Для створення гри необхідно розробити власний або обрати сторонній ігровий рушій. Власний ігровий рушій міг би бути кращим рішенням зі сторони оптимізації та цільової направленості на конкретний проєкт, але розробка ігрового рушія зайняла б багато часу через його складність. Приклад архітектури ігрового рушія зображено на рисунку 3.2 [11].

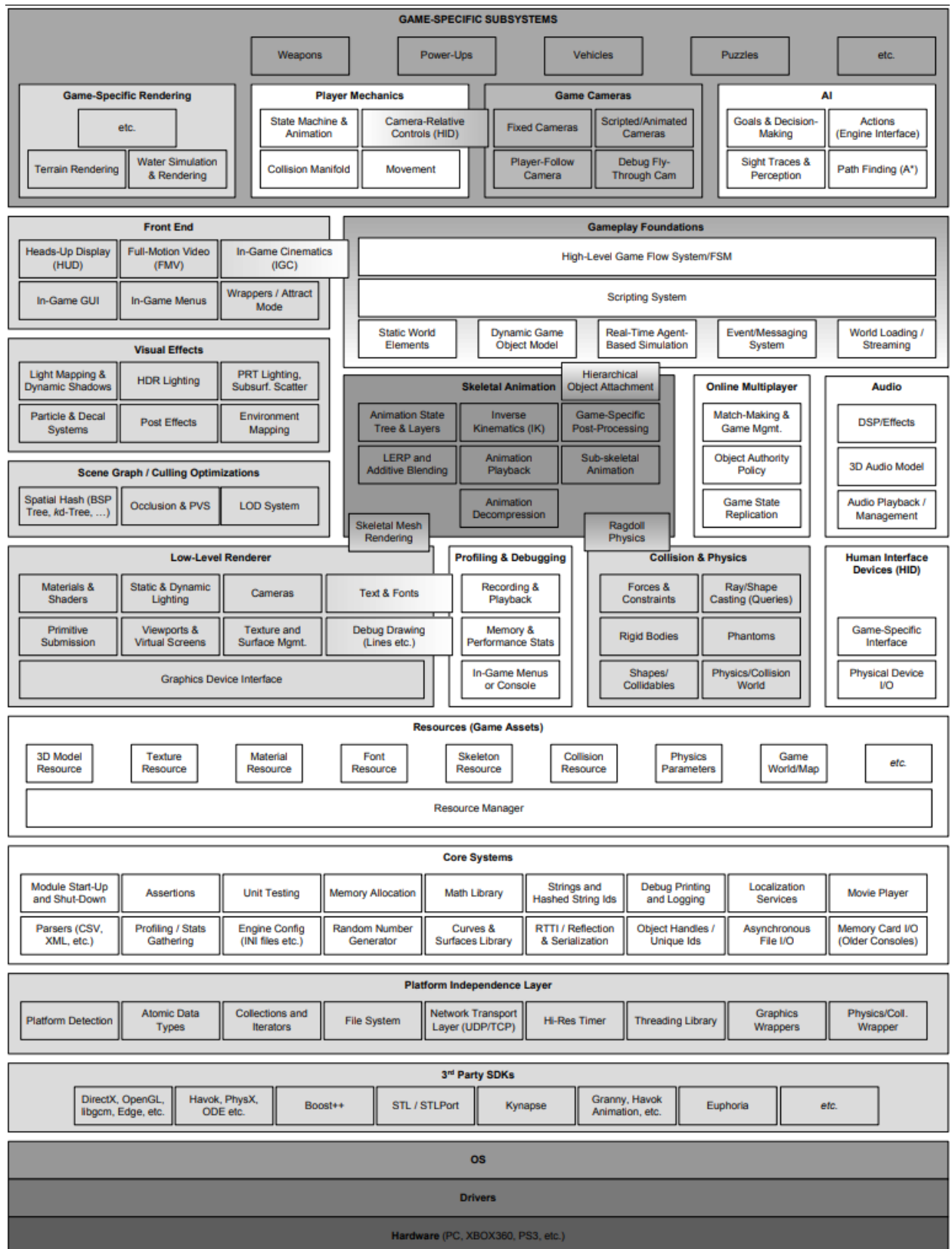


Рисунок 3.2 – Архітектура ігрового рушія

Тому обрано рішення для використання стороннього рушія. Серед безкоштовних 3D рушіїв можна розглянути Unreal Engine, Unity, Godot, Panda3D та CryEngine. Порівняння ігрових рушіїв описано у статті [12].

Unity [13] – простий у використанні рушій з великою кількістю навчальних матеріалів, використовує мову програмування C#.

Godot [14] – має сильну підтримку 2D і меншу для 3D, використовує мови програмування GDScript, C#, C++, має невелику спільноту.

Panda3D [15] – повністю безкоштовний ігровий рушій, використовує мови програмування Python, C++, має малу спільноту та слабку документацію.

CryEngine [16] – ігровий рушій, орієнтований на реалізм, використовує мови програмування C++, Lua, VisualScript, має невелику спільноту та мало навчальних матеріалів.

Для виконання дипломного проєкту обрано ігровий рушій Unreal Engine [17] через його високу якість графіки, використання мови C++ для більш низькорівневої оптимізації пам'яті та процесів, наявності великої кількості навчальних матеріалів. В Unreal Engine також застосовується система візуального скриптування Blueprints, блоки репрезентують окремі команди або змінні, а не заміщують програмування. Для написання програмного коду використовується IDE Visual Studio, що підтримується Unreal Engine.

3.3 Конструювання програмного забезпечення

Для розробки програмного забезпечення використовуються програми, що вказані в таблиці 3.1.

Таблиця 3.1 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Unreal Engine	Рушій для розробки 3D-ігор і візуалізацій, який забезпечує високоякісну графіку та широкі можливості для створення інтерактивного контенту

Продовження таблиці 3.1

№ п/п	Назва утиліти	Опис застосування
2	Visual Studio	Інтегроване середовище розробки від Microsoft, яке підтримує різні мови програмування та інструменти для створення програмного забезпечення

Тексти програмного коду наведені в окремому документі «Текст програми».

Для конструювання програмного забезпечення потрібно описати програмні об'єкти, що будуть розроблені для виконання функціональних задач.

3.3.1 Опис ігрових об'єктів

Основними об'єктами, якими маніпулює гравець, є юніти та будівлі, що описані в аналізі предметної області (розділ 1.2).

Юніти є нащадками базового класу Unreal Engine під назвою Character [18], що може керуватися гравцем або штучним інтелектом.

Юніти мають такі характеристики:

- кількість одиниць здоров'я;
- кількість одиниць шкоди, що може наносити іншому об'єкту;
- дальність атаки;
- швидкість руху;
- належність до гравця чи ворога.

Юніти поділяються на різні види:

– робітник – добування ресурсів та ремонт будівель, не може атакувати інші юніти;

- піхотинець – основний військовий юніт з ближньою атакою;
- стрілець – військовий юніт з дальньою атакою;

– вершник – військовий юніт з ближньою атакою та високою мобільністю.

Юніти виконують накази певний проміжок часу, тому їм присвоюється окремий стан виконання певного наказу. Такими станами є:

- наказ на бездіяльність;
- наказ на переміщення;
- наказ на збір ресурсів;
- наказ на атаку.

Юніти також мають окремий стан анімації:

- бездіяльність;
- рух;
- добування ресурсів;
- атака;
- ремонт;
- переслідування.

Концептуальна діаграма класів для розробки юнітів відображена на рисунку 3.3.

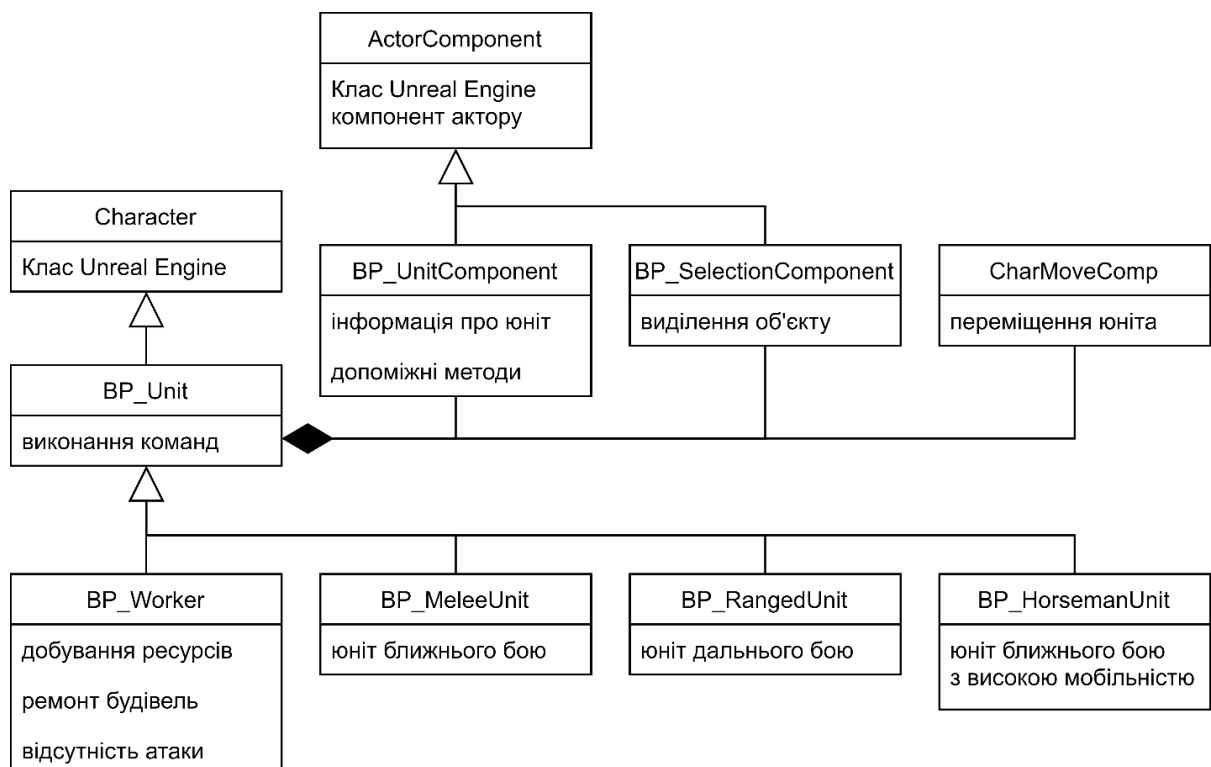


Рисунок 3.3 – Діаграма класів юнітів

Будівлі є нащадками базового класу Unreal Engine під назвою Actor [19], що є базовим класом для об'єктів, що можуть бути розміщені на рівні.

Будівлі мають такі характеристики:

- кількість одиниць здоров'я;
- належність до гравця чи ворога.

Будівлі поділяються на різні види:

- ратуша – будівля для збереження ресурсів та створення нових робітників;
- казарма – будівля для створення нових військових юнітів;
- будинок – будівля для збільшення максимальної кількості юнітів гравця.

Концептуальна діаграма класів для розробки будівель відображена на рисунку 3.4.

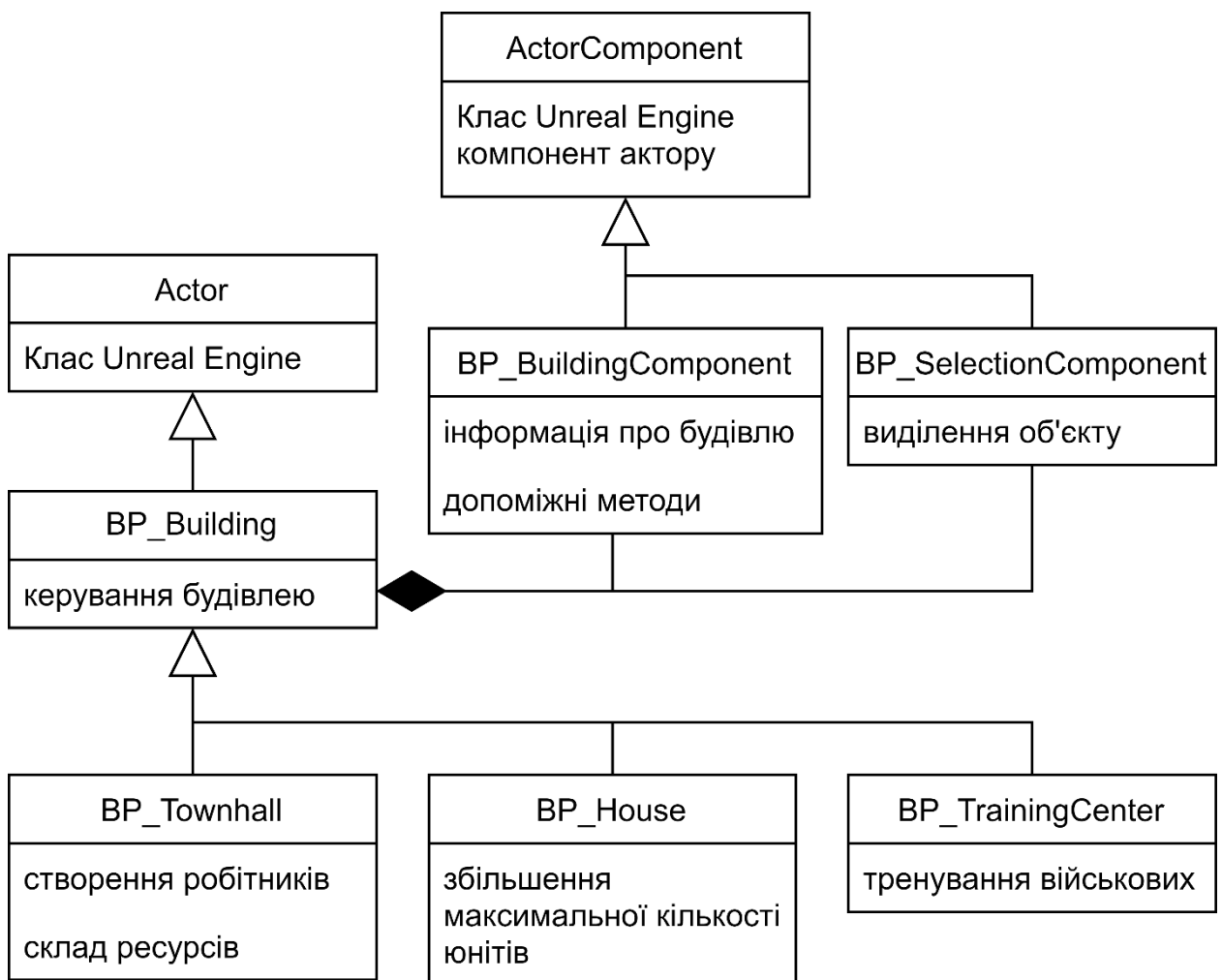


Рисунок 3.4 – Діаграма класів будівель

3.3.2 Опис допоміжних об'єктів

Користувацький інтерфейс реалізовано за допомогою UMG.UserWidget [20]. Для гри потрібні такі елементи інтерфейсу:

- головне меню;
- меню завантаження збереження;
- меню створення нової гри;
- ігровий інтерфейс;
- меню паузи;
- меню збереження;
- екрани перемоги та поразки;
- меню налаштувань.

Керування ігровим процесом реалізовано за допомогою ігрового контролера гравця, що є нащадком базового класу Unreal Engine під назвою PlayerController [21]. Контролер відповідає за:

- переміщення камери по площині XY за допомогою клавіш WASD та підведення курсору миші до краю екрану;
- виділення об'єктів лівою кнопкою миші за допомогою натискання або утримування і обведення прямокутником;
- видача наказу правою кнопкою миші виділеним юнітам;
- збереження списку виділених об'єктів;
- збереження кількості ресурсів гравця.

Рівні реалізовані шляхом розміщення юнітів, будівель та ресурсів на карті класу типу Level [22]. Рівні також містять цілі гравця по знищенню ворога для виконання місії.

Юніти супротивника мають поведінку, визначену за допомогою дерева поведінки. Приклад дерева поведінки, за яким юніт атакує при достатній кількості очок здоров'я і повертається на базу для відновлення, зображено на рисунку 3.5.

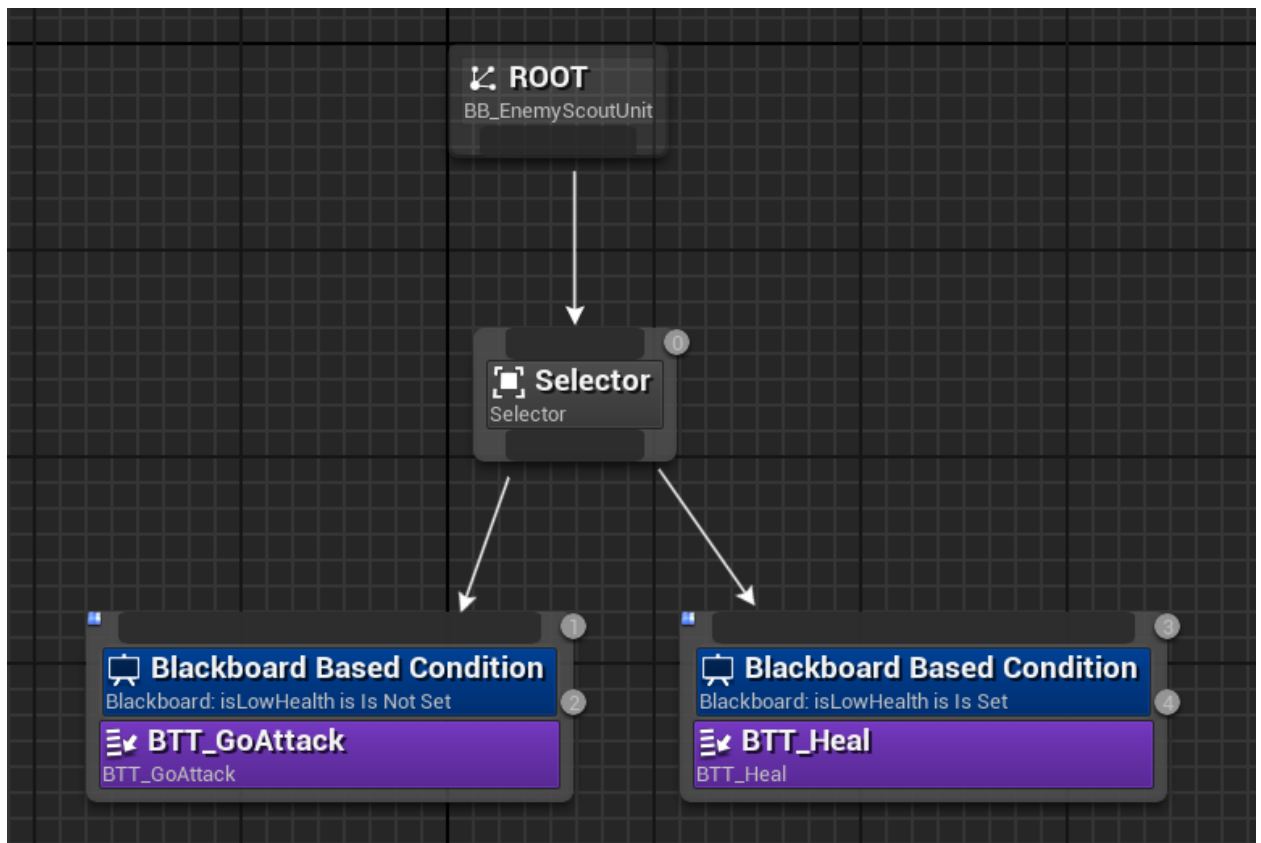


Рисунок 3.5 – Приклад дерева поведінки для юніта ворога

Збереження реалізовані за допомогою класу SaveGame [23], в об’єкті якого записуються дані про юніти, будівлі, ресурси на карті та гравця, час збереження, прогрес досягнення цілі місії та назву, яку задав користувач.

Звуковий супровід реалізовано за допомогою Unreal Audio Engine [24]. Шари музики передбачають декілька аудіодоріжок, які змінюють одна одну відповідно до подій гри. Шари музики програються відповідно до того, чи знаходиться зараз гравець під атакою, і загалом звучать для гравця як мирна мелодія та войовнича мелодія.

3.4 Аналіз безпеки даних

Питання, пов’язані з безпекою даних та некоректним використанням програмного забезпечення, повністю покладаються на користувача.

Висновки до розділу

У розділі здійснено опис конструювання та розробки програмного забезпечення, зокрема архітектури та засобів розробки.

Було запропоновано монолітну архітектуру з елементами подійно-орієнтованої, що складається з 7 компонентів. Гра є єдиним програмним забезпеченням, а компоненти генерують та обробляють події.

Було обґрунтовано, що для розробки потрібно обрати ігровий рушій, адже він є складним програмним застосунком з численною кількістю елементів. Після порівняння обрано Unreal Engine через високу якість графіки, використання мови C++ та системи візуального скриптування Blueprints.

Було детально описано потрібні елементи для розробки гри, зокрема ігровий контролер, збереження гри, звуковий супровід, юніти, будівлі. Також було описано інструменти розробки – Unreal Engine та Visual Studio.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення здійснено для перевірки досягнення мети розробки та виконання нефункціональних вимог.

Для підтвердження досягнення мети високого рівня залученості гравців можливо чисельно виміряти час проходження місій, співвідношення перемог та поразок, кількість перезапусків місій, кількість різних типів юнітів гравця, відсоток перемог на місії від кількості всіх гравців, час між запусками гри [25]. Однак, для репрезентативних результатів вимірювання потрібна велика кількість гравців та час.

Через обмеженість ресурсів було обрано метод гри вголос [26]. Гравець описує кожну свою дію у присутності розробника, як наче проводить пряму трансляцію для інших глядачів. Таким чином можливо зібрати відгук гравців для коригування гри до збільшення її залученості.

Для методу обрано 5 гравців та здійснено 3 ітерації проходження та оновлення гри. У результаті збору даних здійснено зміни до ігрового процесу: додано нові типи юнітів для більшого різноманіття гри – стрілець та вершник, відредаговано характеристики юнітів, зокрема кількість одиниць здоров'я та шкоди, внесено зміни до місій.

Для підтвердження плавності гри, що є метою та нефункціональною вимогою продуктивності, обрано показник частоти кадрів та визначено цільове значення при рекомендованій конфігурації технічних засобів – 60, що є оптимальним значенням для реагування на дії на екрані та підтримується більшістю екранів [27].

Для визначення частоти кадрів було увімкнено збір статистики у редакторі Unreal Engine під час бою між юнітами гравця та супротивника – потенційно найбільш навантаженої сцени. Результати збору значень частоти кадрів відображено на рисунку 4.1.

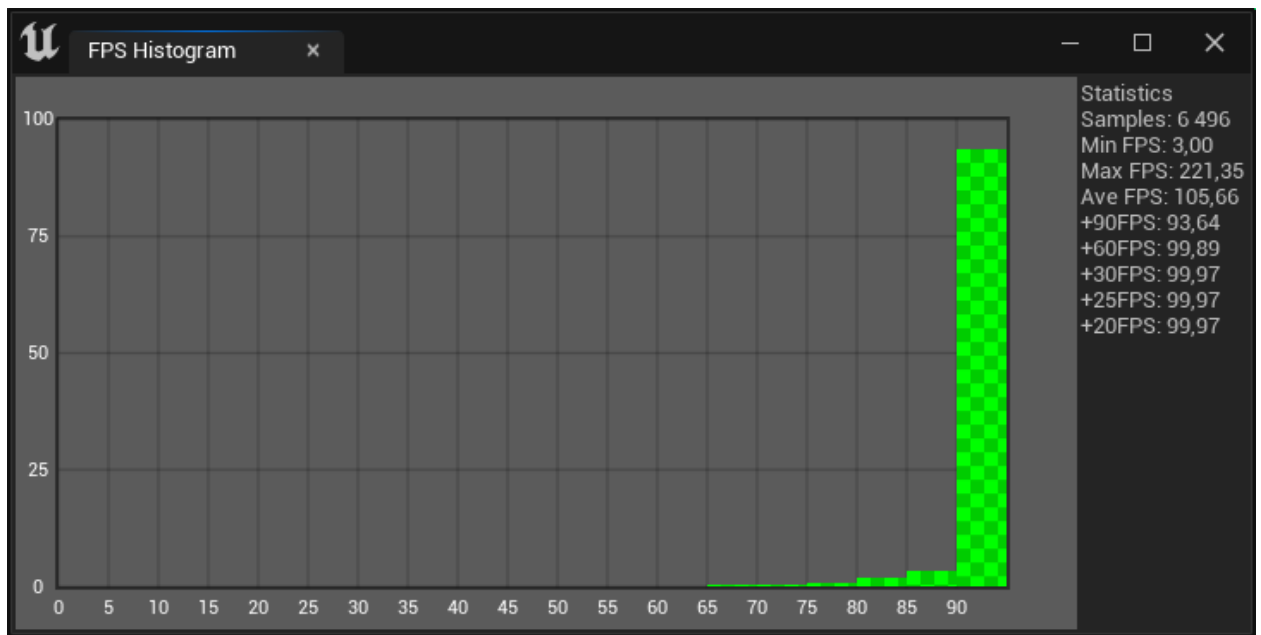


Рисунок 4.1 – Гістограма частоти кадрів

Відповідно до гістограми, 99,89% часу заміру частота кадрів становила більше 60 і середнє значення – 105, що значно перевищує цільовий показник.

Також у вимогах до продуктивності було зазначено швидкий час відгуку, що є часом кадру – оберненим значенням до частоти кадру. При середньому значенні частоти кадру 105 середній час кадру становить 9,5 мілісекунд.

Модульність та висока гнучкість розробки забезпечена використанням подійно-орієнтованої архітектури. Як приклад, при додаванні компоненту аудіо не потрібно було переписувати наявні компоненти, бо юніти та будівлі уже генерували події на атаку чи створення юніта, тому компоненту аудіо потрібно лише обробляти ці події.

Нефункціональні вимоги надійності та сумісності перевірені шляхом багаторазового запуску та проходження місій гри. При перевірці не було виявлено випадкових зависань, завершень роботи чи проблем з роботою при різних роздільних здатностях екрану.

4.2 Опис процесів тестування

Тестування виконується згідно з документом «Програма та методика тестування».

Поділ тестування від нижнього до верхнього рівнів здійснено відповідно до книги «Game Testing: All in One» [28].

Юніт-тести фокусуються на окремі невеликі фрагменти коду, наприклад функції чи класи, та забезпечують правильність базової логіки перед інтеграцією. Ці тести прописуються програмістами. Через мінливість розробки відеоігор, складність взаємодій та компонентів цей вид тестування є менш поширеним, ніж у традиційній розробці програмного забезпечення. У рамках дипломного проєкту було прописано юніт-тести для базових методів класів. Наприклад, метод, який розраховує кількість нанесеного персонажем шкоди, перевірявся на правильність обчислення вхідних та вихідних значень.

Інтеграційне тестування перевіряє взаємодію між окремими компонентами, наприклад механізм видачі та виконання наказів юнітами. У рамках дипломного проєкту виконано вручну.

Функціональне тестування охоплює перевірку завершеного функціоналу, наприклад, виконання наказів юнітами чи робота збереження. У рамках дипломного проєкту виконано вручну.

Плейтест фокусується на загальному досвіді гри, ігровому процесі, зручності управління, візуальному сприйнятті та балансі. Цей тип тестування проводиться вручну розробниками, тестувальниками або гравцями всередині компанії. Можна сказати, що досягнення мети високого рівня залученості гравця було перевірено за допомогою плейтесту.

Успішний результат автоматизованого тестування інструментами рушія Unreal Engine та прописаними юніт-тестами відображено на рисунку 4.2.

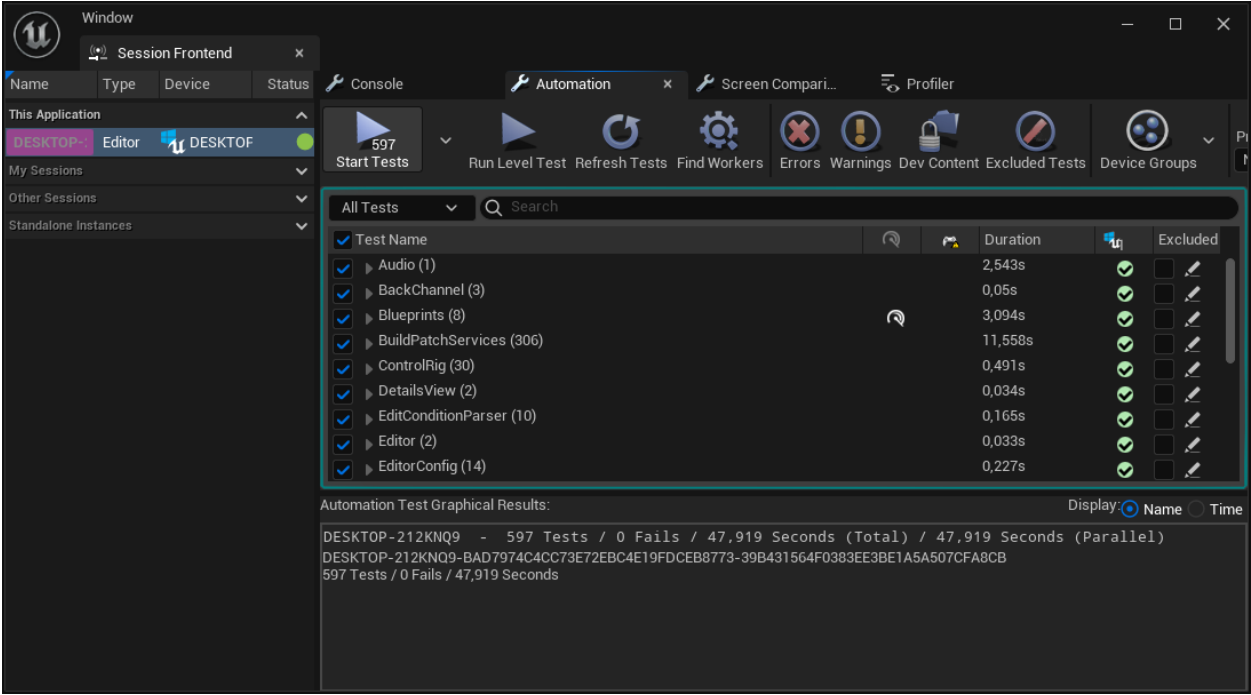


Рисунок 4.2 – Результат автоматизованого тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.12.

Таблиця 4.1 – Тест 1.1 Наказ на переміщення юніта

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Юніт та позиція для переміщення
Опис проведення тесту	Користувач виділяє юніта та натискає правою кнопкою миші на позицію, на яку хоче перемістити юніта.
Очікуваний результат	Юніт успішно переміщується у задану позицію.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.2 – Тест 1.2 Наказ на збір ресурсів юнітом

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Юніт, ресурс на карті для збору
Опис проведення тесту	Користувач виділяє юніта, натискає правою кнопкою миші на ресурс, який хоче добути.
Очікуваний результат	Юніт переміщується до ресурсу, певний проміжок часу добуває його, переміщується до найближчого складу ресурсів і число ресурсів оновлюється у верхній частині ігрового інтерфейсу. Якщо ресурс залишився на карті, то юніт повторно виконує наказ.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.3 – Тест 1.3 Наказ на атаку юнітом

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Юніт користувача, ворожий юніт
Опис проведення тесту	Користувач виділяє юніта, натискає правою кнопкою миші на ворожого юніта, якого хоче атакувати.
Очікуваний результат	Юніт починає переслідувати ворога і при наближенні в радіус атаки здійснює певний час окрему атаку і якщо ворог залишився в радіусі, то знімаються одиниці здоров'я. Юніт продовжує виконувати наказ до знищення ворога.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.4 – Тест 1.4 Створення нового юніта

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Будівля для створення юніта, тип юніта, ресурси користувача
Опис проведення тесту	Користувач виділяє будівлю, натискає кнопку інтерфейсу у лівому нижньому куті на створення юніта.
Очікуваний результат	Списуються ресурси, що відображаються у верхній частині ігрового інтерфейсу, через певний проміжок часу з'являється новий юніт заданого типу біля будівлі, що його створювала.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.5 – Тест 2.1 Збереження гри

Початковий стан системи	Користувач знаходиться у меню паузи
Вхідні дані	Назва збереження
Опис проведення тесту	Користувач натискає на кнопку збереження гри, вводить у поле назву збереження відповідно до правил назв файлової системи Windows, натискає на кнопку збереження.
Очікуваний результат	У списку збережень з'являється нове збереження із заданою назвою та часом збереження.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.6 – Тест 2.2 Завантаження збереження

Початковий стан системи	Користувач знаходиться у головному меню або меню паузи
Вхідні дані	Назва збереження
Опис проведення тесту	Користувач натискає на кнопку завантаження збереження у відповідному меню, натискає на збереження, що хоче відновити, та натискає на кнопку завантаження.
Очікуваний результат	Завантажується збереження з попередньо записаними позиціями юнітів, будівель, ресурсів, відновлюється процес гри та здійснюється перехід до екрану ігрового процесу.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.7 – Тест 2.3 Видалення збереження

Початковий стан системи	Користувач знаходиться в меню завантаження або збереження прогресу гри
Вхідні дані	Назва збереження
Опис проведення тесту	Користувач натискає на збереження, що хоче видалити, та натискає на кнопку видалення.
Очікуваний результат	Виділене збереження зникає зі списку збережень на екрані.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.8 – Тест 3.1 Програвання шарів фонової музики

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Юніти
Опис проведення тесту	Користувач здійснює атаку юнітами на ворога або підставляє своїх юнітів під атаку ворога.
Очікуваний результат	При нанесенні шкоди змінюються шари музики з мирної на войовничу мелодію, через певний проміжок часу після останнього нанесення шкоди шари музики повертаються на мирну мелодію.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.9 – Тест 3.2 Програвання звукових ефектів під час бою

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Юніти
Опис проведення тесту	Користувач здійснює атаку юнітами на ворога або підставляє своїх юнітів під атаку ворога.
Очікуваний результат	Під час атаки юнітом програється звуковий ефект.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.10 – Тест 4.1 Вибір місії

Початковий стан системи	Користувач знаходиться у головному меню
Вхідні дані	Назва місії
Опис проведення тесту	Користувач натискає на кнопку початку нової гри, натискає на місію та натискає на кнопку початку гри
Очікуваний результат	Здійснюється завантаження вмісту місії та перехід до екрану ігрового процесу.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.11 – Тест 5.1 Перемога гравця

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Дії гравця
Опис проведення тесту	Користувач виконує місію за допомогою наказів юнітів та створення нових юнітів.
Очікуваний результат	Відображається екран перемоги.
Фактичний результат	Збігається з очікуванням.

Таблиця 4.12 – Тест 5.2 Поразка гравця

Початковий стан системи	Користувач знаходиться на екрані ігрового процесу
Вхідні дані	Дії гравця
Опис проведення тесту	Користувач втрачає усі свої юніти та будівлі.
Очікуваний результат	Відображається екран поразки.
Фактичний результат	Збігається з очікуванням.

4.3 Опис контрольного прикладу

Контрольний приклад відображає послідовність процесів, описаних у підрозділі 1.4 при побудові діаграм BPMN.

Є два шляхи запуску ігрового процесу. Перший спосіб – почати нову гру за допомогою вибору рівня. На рисунку 4.3 відображено меню вибору місії. Потрібно натиснути на місію зі списку та потім на кнопку початку нової гри.

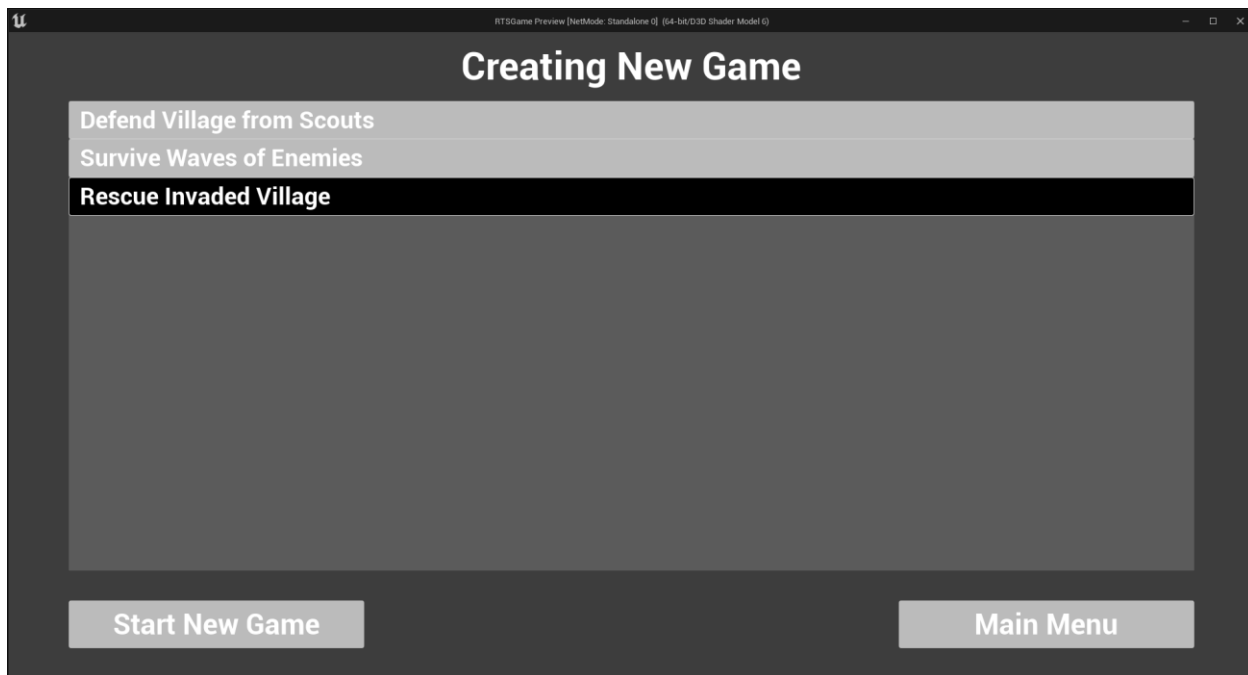


Рисунок 4.3 – Меню створення нової гри

Другий спосіб – завантажити збереження. На рисунку 4.4 відображено меню завантаження збереження. Потрібно натиснути на збереження зі списку та потім на кнопку завантаження.

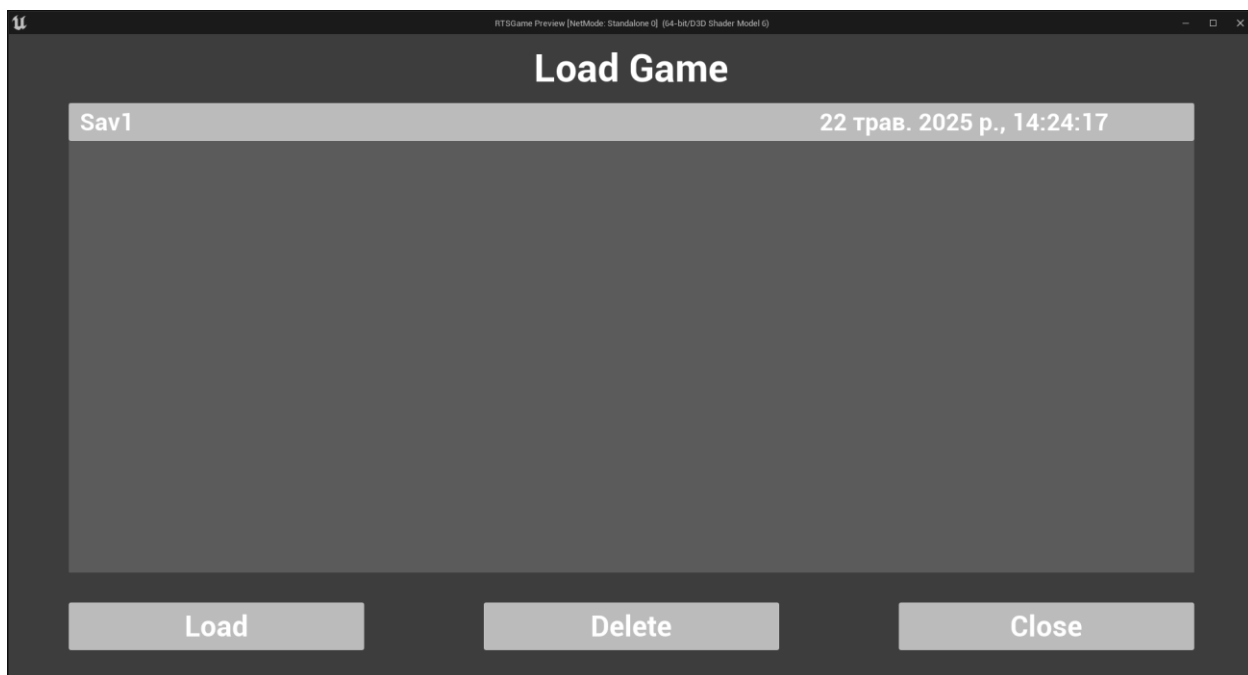


Рисунок 4.4 – Меню завантаження

Після створення нової гри або завантаження відображається екран ігрового інтерфейсу (рисунок 4.5).

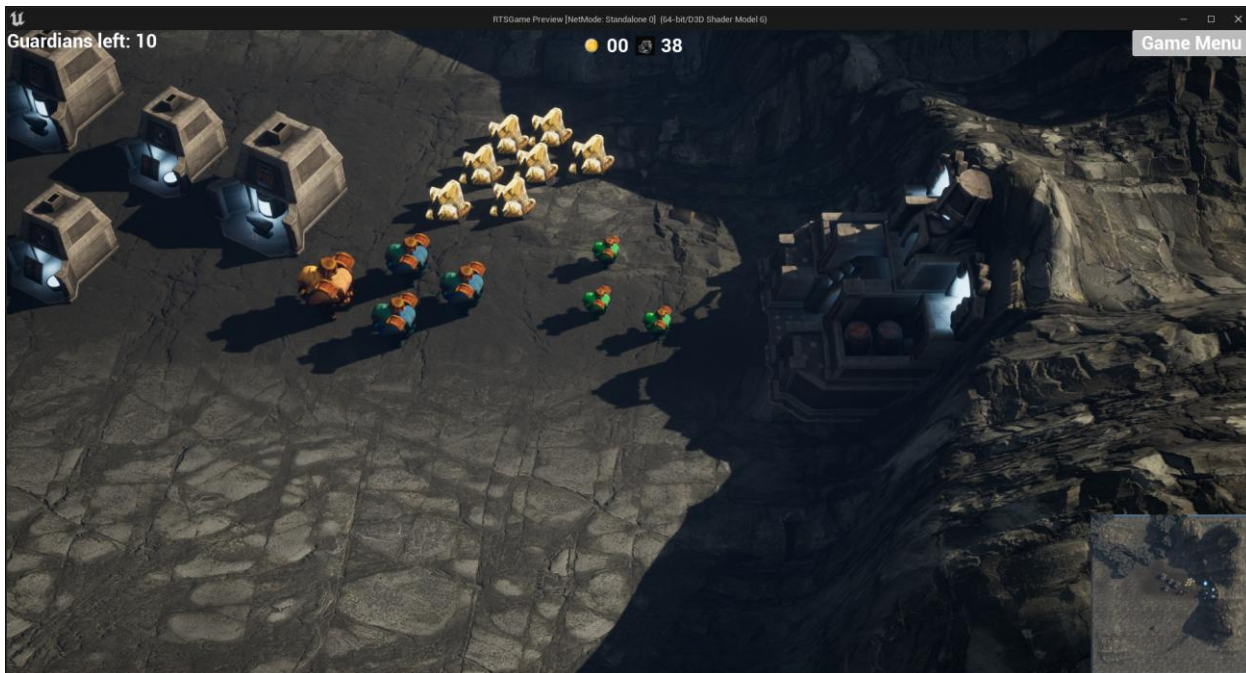


Рисунок 4.5 – Екран ігрового інтерфейсу

Процес виділення об'єктів утримуванням лівої кнопки мишою відображено на рисунку 4.6. Список об'єктів з інформацією про них відображається у лівому нижньому куті.

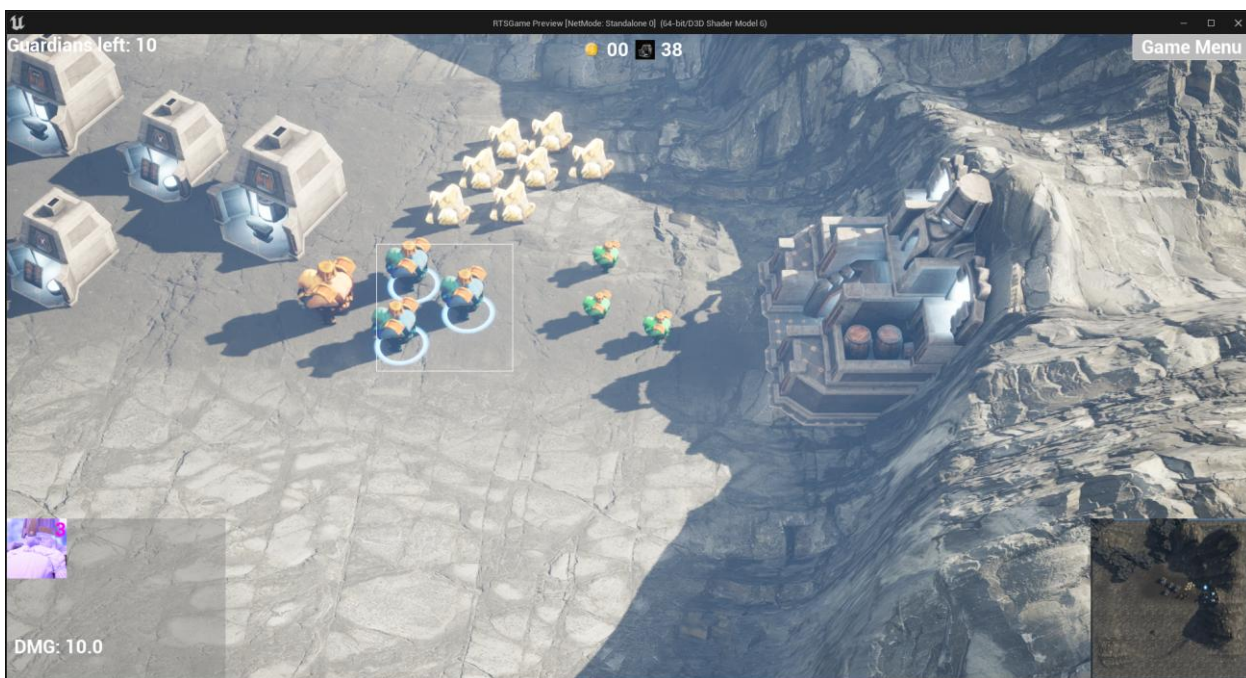


Рисунок 4.6 – Виділення ігрових об'єктів

Наказ на переміщення, здійснений натисканням правої кнопки миші для виділених юнітів, відображено на рисунку 4.7.

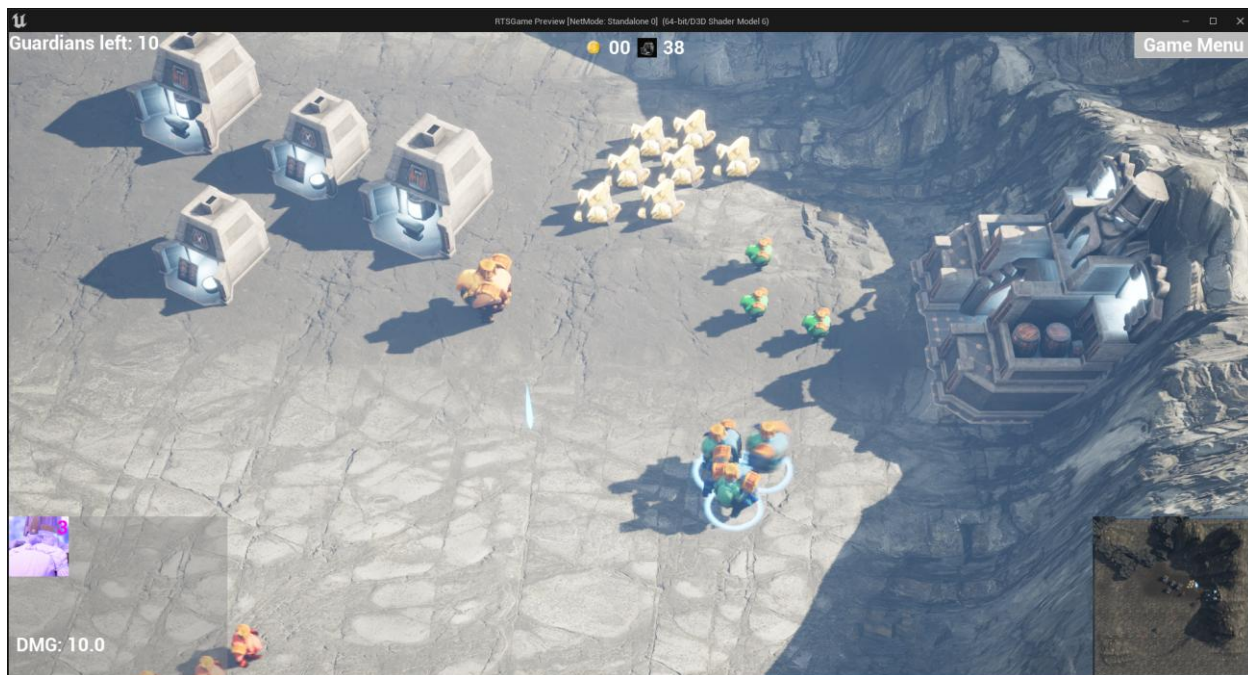


Рисунок 4.7 – Наказ на переміщення

Наказ на атаку, здійснений натисканням правою кнопкою миші по ворогу, відображено на рисунку 4.8. Юніти підходять до ворога, здійснюють атаку та у ворога зменшуються одиниці здоров'я, що відображаються шкалою над ним.



Рисунок 4.8 – Наказ на атаку

Після виділення робітника та натискання правої кнопки миші по будівлі здійснюється наказ на ремонт будівлі (рисунок 4.9). Робітник підійшов до будівлі і почав ремонт, при цьому кількість одиниць здоров'я будівлі зростає.



Рисунок 4.9 – Наказ на ремонт будівлі

Наказ на добування ресурсів відображено на рисунку 4.10. Юніти циклічно підходять до ресурсу, добувають його та відносять у ратушу.

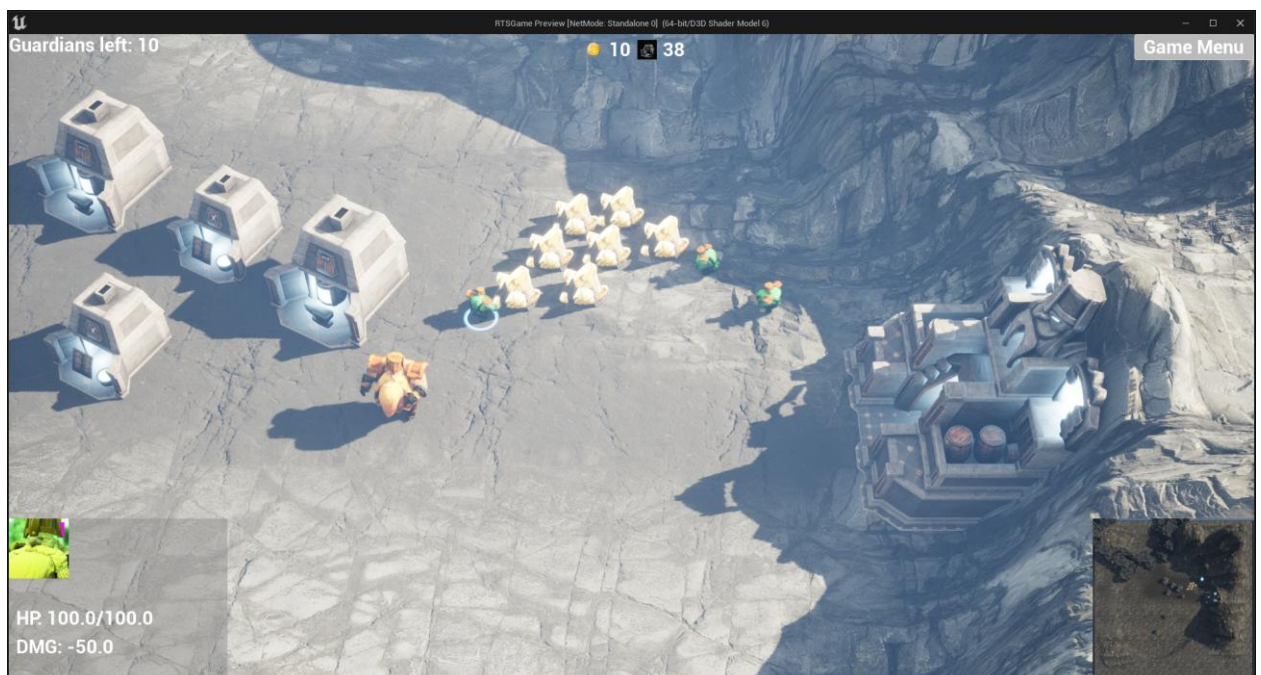


Рисунок 4.10 – Наказ на добування ресурсів

Збереження гри відображено на рисунку 4.11. Гравець вводить назву збереження у поле і натискає на кнопку зберегти.



Рисунок 4.11 – Збереження гри

Нове збереження відображається у списку збережень, зокрема у меню завантаження (рисунок 4.12).

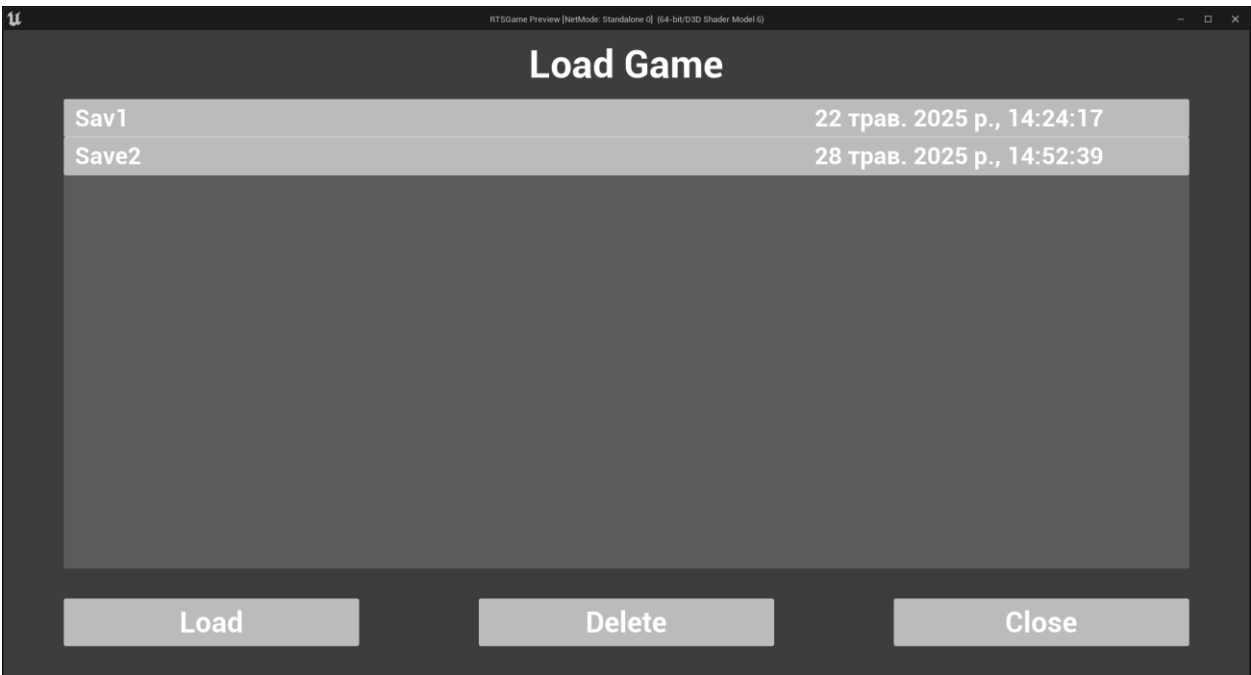


Рисунок 4.12 – Оновлення меню завантаження

Для створення юніта спочатку потрібно виділити будівлю, що може створювати юнітів (рисунок 4.13). При наведенні на кнопку створення юніта відображається необхідна кількість ресурсів.

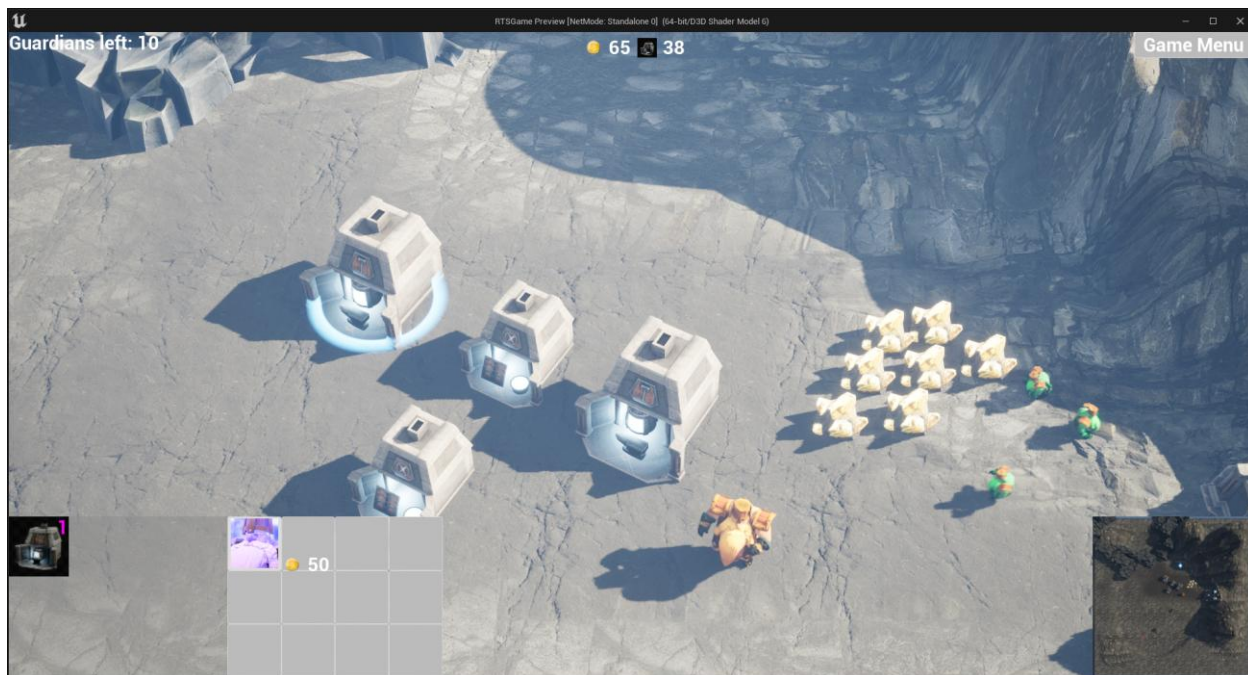


Рисунок 4.13 – Виділення казарми для створення юніта

Після натискання на кнопку створення юніта списуються ресурси та через деякий час з'являється новий юніт (рисунок 4.14).

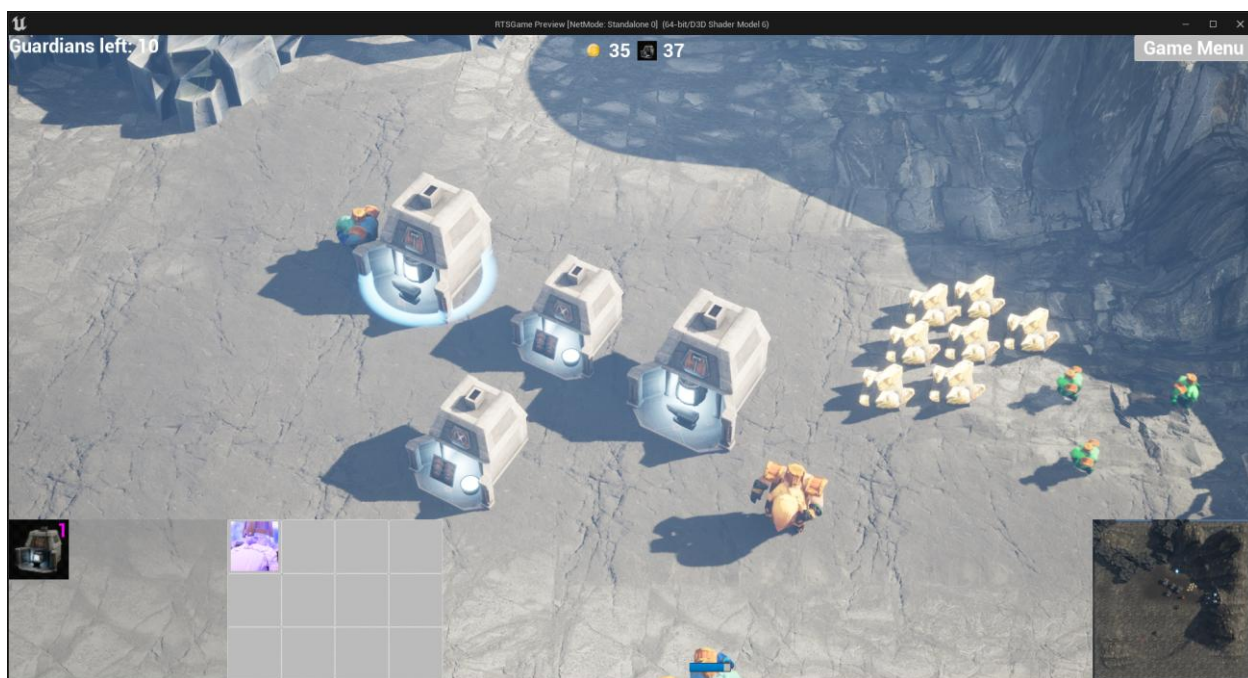


Рисунок 4.14 – Завершення створення нового юніта

Прогрес виконання місії відображено на рисунку 4.15.

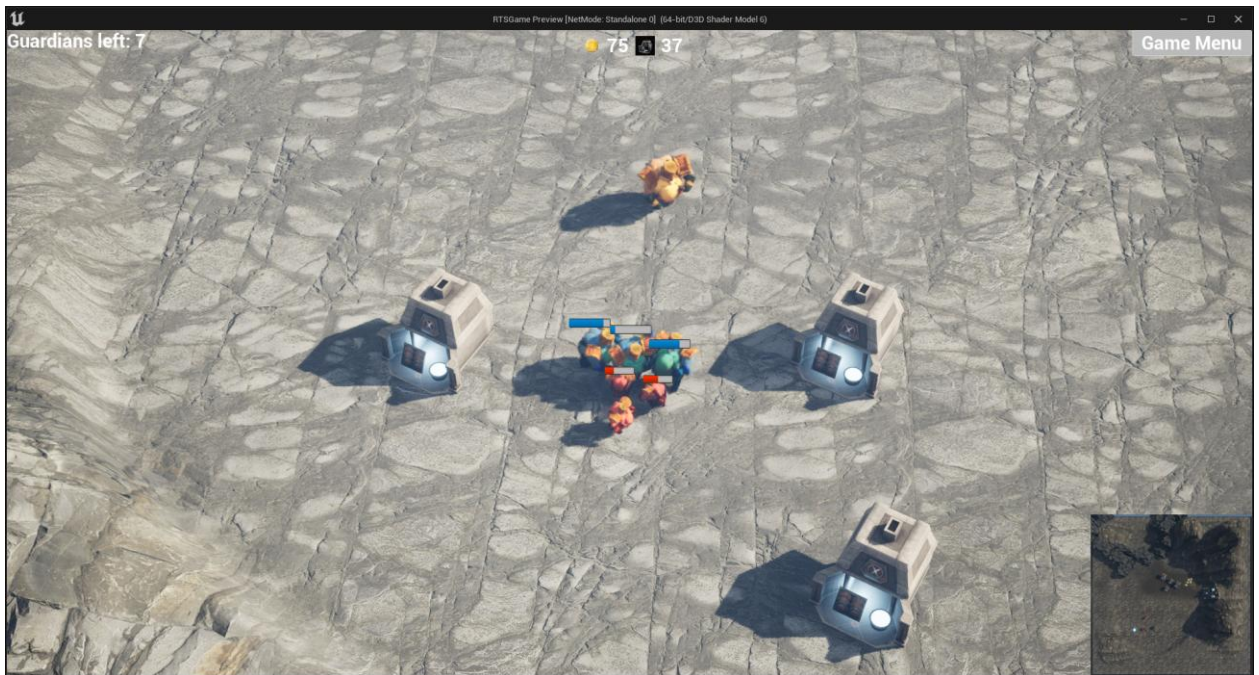


Рисунок 4.15 – Виконання місії

При виконанні місії, у даному випадку – після знищення 10 юнітів ворога, відображається екран перемоги (рисунок 4.16).

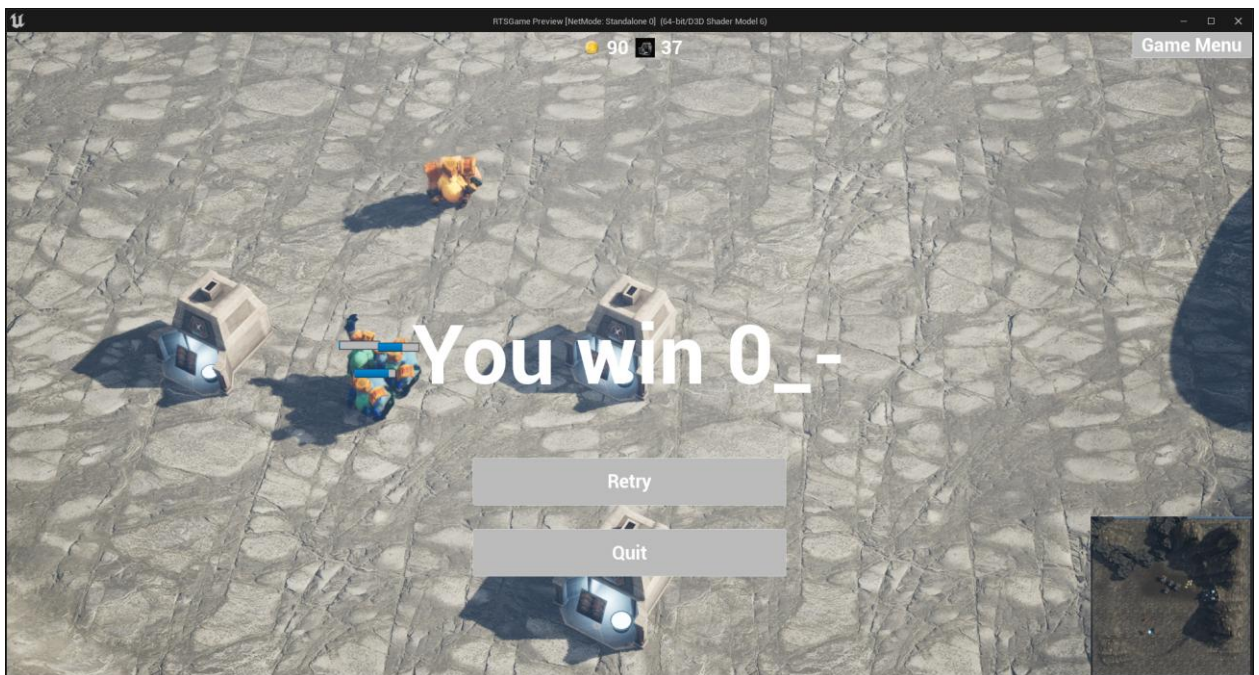


Рисунок 4.16 – Екран перемоги

При втраті робітників та ратуші відображається екран поразки (рисунок 4.17).

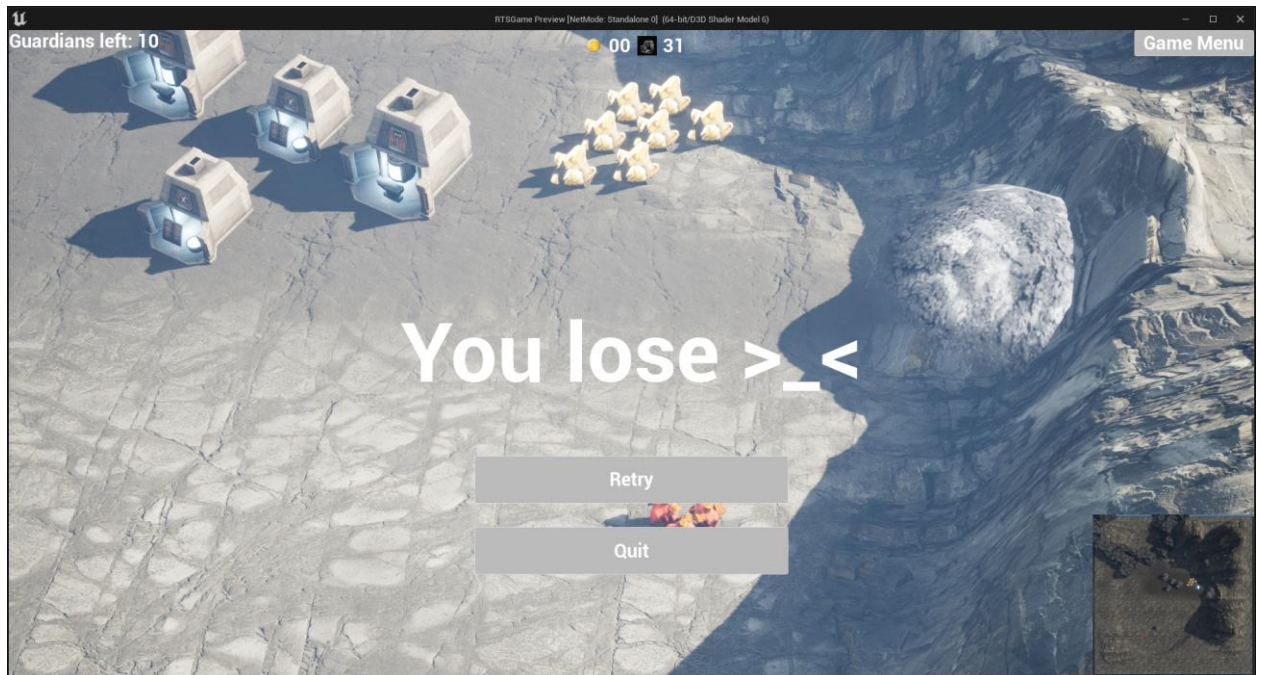


Рисунок 4.17 – Екран поразки

Контрольний приклад демонструє роботу всього функціоналу гри відповідно до описаних бізнес-процесів та функціональних вимог окрім звукового супроводу, що неможливо перевірити у текстовому чи графічному вигляді.

Висновки до розділу

У розділі здійснено аналіз якості та тестування програмного забезпечення для визначення виконання мети та вимог, поставлених у попередніх розділах.

Спочатку було здійснено аналіз якості програмного забезпечення. Доведено досягнення мети високого рівня залученості гравців за допомогою методу гри вголос 5 гравцями у результаті 3 ітерацій проходження та оновлення гри. Отримано середнє значення частоти кадрів 105, що перевершує цільовий показник у 60 кадрів та доводить досягнення мети високих показників продуктивності та нефункціональних вимог продуктивності. Модульність та висока гнучкість розробки забезпечена використанням подійно-орієнтованої архітектури. Нефункціональні вимоги надійності та сумісності перевірені багатократним проходженням гри.

Визначено рівні тестування та їхнє призначення. Юніт-тести було розроблено для низки елементарних методів програми. Інтеграційне та функціональне тестування здійснено вручну. Плейтест проведено у рамках досягнення мети високого рівня залученості гравців.

Здійснено опис контрольного прикладу, чим доведено виконання функціональних вимог та здатність гри слідувати визначеному бізнес-процесу.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для поширення гри потрібно спакувати гру в рушії Unreal Engine та створити інсталяційний файл.

Для пакування гри було використано послідовність дій з документації ігрового рушія [29].

Було створено новий користувацький профіль пакування (рисунок 5.1).

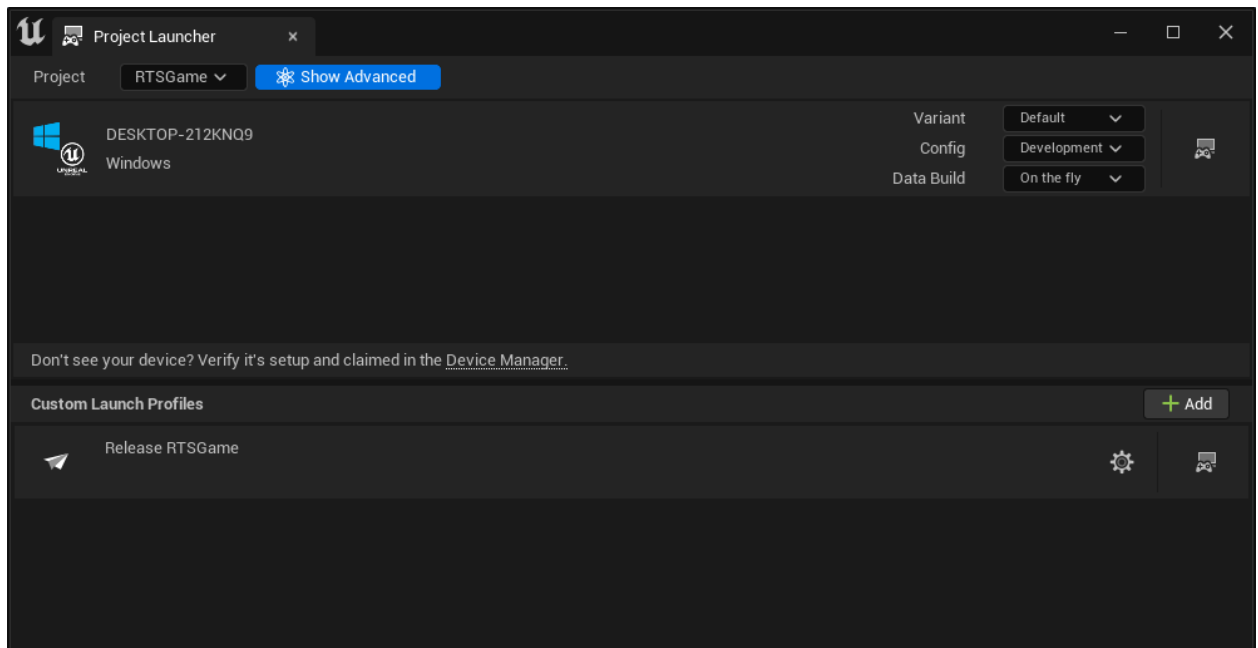


Рисунок 5.1 – Профілі пакування

Перелік налаштувань профілю:

- Build Configuration – Shipping;
- Cook – by the Book, Windows;
- Releases / DLC / Patching Settings – Create a release version of the game for distribution, Name of the new release to create – 1.0;
- Advanced Settings – Compress content, Save packages without versions, Store all content in a single file (UnrealPak), Cooker build configuration – Shipping;
- Package – Package & store locally;
- Deploy – Do not deploy.

Для створення інсталятора було застосовано InstallForge [30]. Головний екран програми відображено на рисунку 5.2.

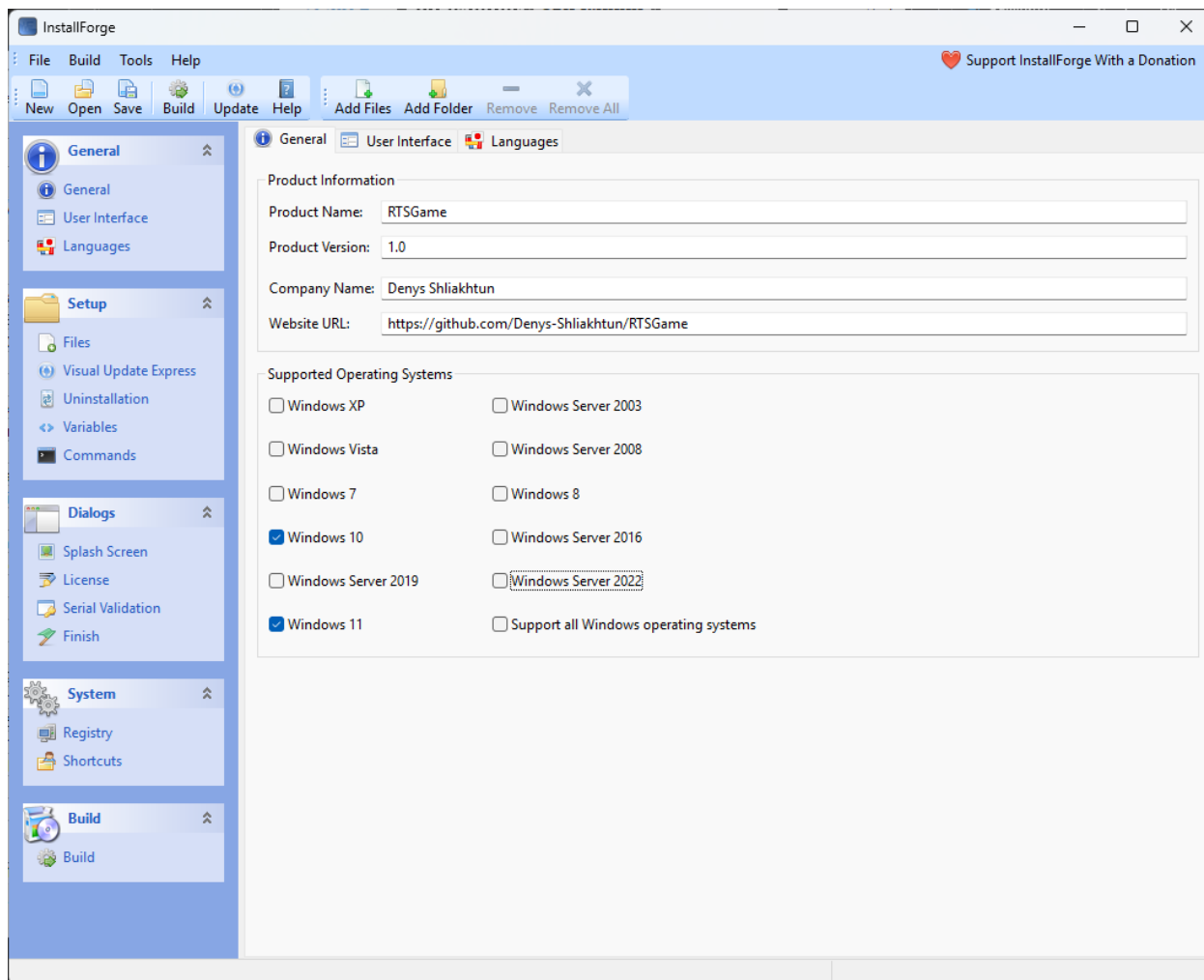


Рисунок 5.2 – Головний екран InstallForge
Результуючий інсталятор відображено на рисунку 5.3.

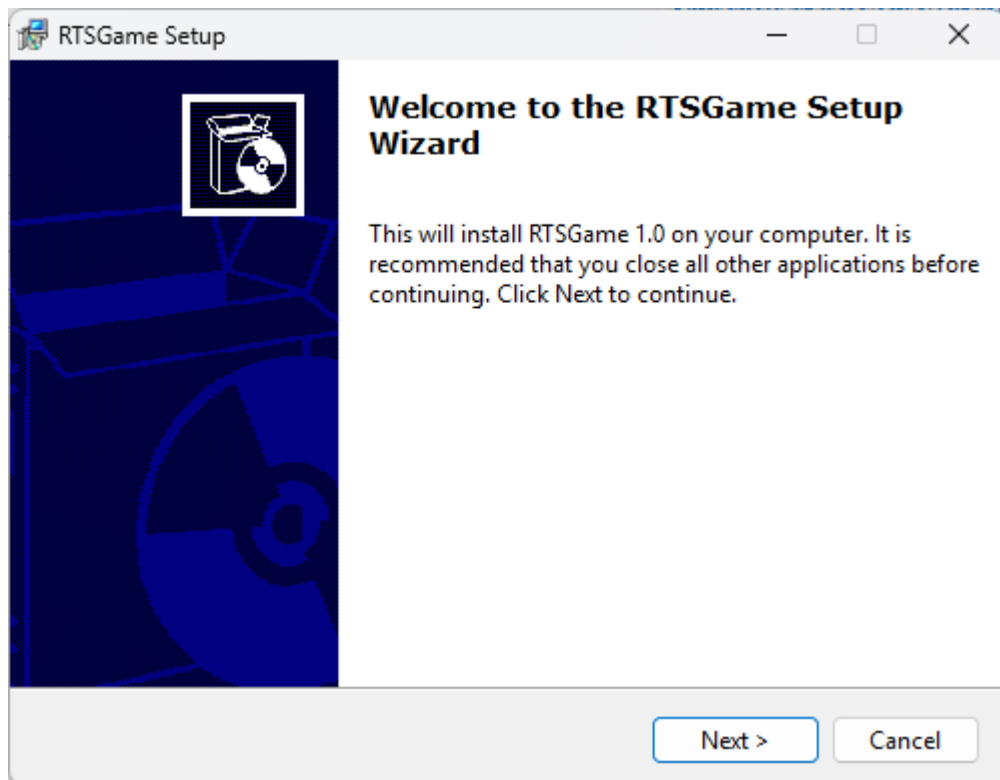


Рисунок 5.3 – Інсталятор гри

Також інсталятор надає можливість вибору папки (рисунок 5.4).

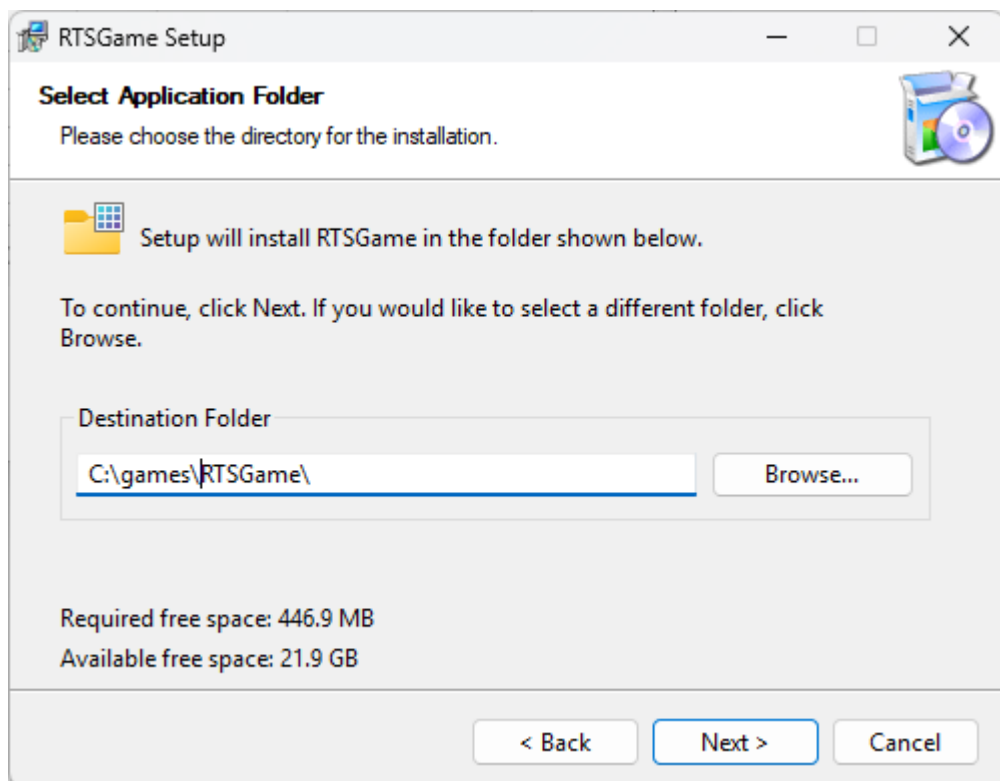


Рисунок 5.4 – Вибір папки для встановлення гри

Інсталяційний файл потім публікується у вкладці Releases репозиторію GitHub. Також у майбутньому можливо викласти гру на майданчику

дистрибуції відеоігор. Наприклад, itch.io [31] надає таку можливість безкоштовно.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво користувача».

Користувачі повинні мати можливість отримати нову версію гри з кожною версією. До того ж кожна нова версія гри повинна бути опублікована в репозиторії GitHub у вкладці Releases. Для встановлення нової версії гри потрібно видалити поточну версію, завантажити інсталяційний файл та повторити встановлення гри.

Висновки до розділу

У розділі було розглянуто процес підготовки гри до поширення та створення інсталяційного файлу. Для цього використовувались інструменти, що надаються рушієм Unreal Engine, а також стороннє програмне забезпечення InstallForge. У результаті виконаних дій створено користувацький профіль пакування з відповідними налаштуваннями, що дозволяють згенерувати фінальну версію гри для розповсюдження. Далі, за допомогою InstallForge, зібраний інсталяційний файл, який дозволяє користувачу встановити гру на своєму комп'ютері.

Отриманий інсталятор дозволяє обрати шлях встановлення та підходить для розміщення на загальнодоступних платформах. Це забезпечує зручний доступ до гри для користувачів і дає змогу швидко поширювати нові версії продукту.

Також було описано супровід програмного забезпечення. Передбачено, що кожна нова версія гри буде публікуватися у репозиторії GitHub, а для встановлення оновлень користувачам потрібно вручну замінювати попередню версію. Інструкція користувача надається окремим документом, що забезпечує зрозуміле встановлення та використання гри.

ВИСНОВКИ

У ході виконання дипломного проєкту розроблено гру у жанрі стратегії в реальному часі з використанням подійно-орієнтованої архітектури, що має базовий ігровий процес керування юнітами та будівлями, рівні для проходження, збереження та завантаження прогресу гри, програвання аудіо.

Мету розробки досягнуто: результати трьох ітерацій тестування методом гри вголос п'ятьма гравцями доводять високий рівень залученості гравців; середня частота кадрів 105 перевищує цільовий показник 60 для плавності гри; реалізація подійно-орієнтованої архітектури забезпечує необхідну гнучкість розробки складних компонентів із багатократними зв'язками.

У рамках дипломного проєкту виконано поставлені задачі. Розроблено графічний інтерфейс, що забезпечує взаємодію між гравцем та компонентами гри. Забезпечено керування ігровим процесом, зокрема переміщення камери, виділення ігрових об'єктів та видачу наказів юнітам. Розроблено керування юнітами, що можуть виконувати накази на переміщення, атаку, ремонт будівель, добування ресурсів. Розроблено керування будівлями, що можуть створювати нових юнітів за наявності ресурсів. Створено три місії з супротивником і завданнями для гравця. Розроблено збереження та завантаження прогресу гри, що дозволяє зберігати та відновлювати стан ігрової карти, прогрес місії та ресурси гравця.

Подальший розвиток проєкту є доцільним: можливе впровадження мережевої гри, вдосконаленого штучного інтелекту супротивника, а також інноваційних функцій — голосового керування юнітами, дипломатії у форматі діалогу або застосування генеративного штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Robertson, G., Watson, I. A Review of Real-Time Strategy Game AI. *AI Magazine*. 2014. № 35(4). С. 75-104. URL: <https://doi.org/10.1609/aimag.v35i4.2478> (дата звернення: 07.02.2025).
- 2) Wolf M. J. P. Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming / Ed. by Mark J. P. Wolf. Santa Barbara, CA: Greenwood, 2012. 740 с.
- 3) Richards M., Ford N. Fundamentals of Software Architecture. Sebastopol, CA: O'Reilly Media, Inc., 2020. 432 с.
- 4) Age of Empires IV. URL: <https://www.ageofempires.com/games/age-of-empires-iv/> (дата звернення: 23.02.2025).
- 5) Warcraft III: Reforged. URL: <https://warcraft3.blizzard.com/> (дата звернення: 23.02.2025).
- 6) 0 A.D. URL: <https://play0ad.com/> (дата звернення: 23.02.2025).
- 7) Ontañón S., Synnaeve G., Uriarte A., Richoux F., Churchill D., Preuss M. RTS AI: Problems and Techniques. *Encyclopedia of Computer Graphics and Games* / ed. by Lee N. Springer, Cham, 2015. С. 1-12. URL: https://doi.org/10.1007/978-3-319-08234-9_17-1 (дата звернення: 04.03.2025).
- 8) Опитування Steam про обладнання та ПЗ: April 2025. URL: <https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam> (дата звернення: 29.04.2025).
- 9) A Guide to the Business Analysis Body of Knowledge (BABOK Guide). International Institute of Business Analysis, 2015. 783 с.
- 10) Hardware and Software Specifications. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine?application_version=5.2 (дата звернення: 29.04.2025).
- 11) Gregory J. Game Engine Architecture, Second Edition. CRC Press, 2017. 1052 с.

- 12) Khan M. T., Bravo-Garcia F., Love L., Gujadhur M., Nyadu J. Analyzing Strengths and Weaknesses of Modern Game Engines. *International Journal of Computer Theory and Engineering*. 2023. Vol. 15, no. 1, С. 54-60.
- 13) Unity. URL: <https://unity.com/> (дата звернення: 05.05.2025).
- 14) Godot. URL: <https://godotengine.org/> (дата звернення: 05.05.2025).
- 15) Panda3D. URL: <https://www.panda3d.org/> (дата звернення: 05.05.2025).
- 16) CryEngine. URL: <https://www.cryengine.com/> (дата звернення: 05.05.2025).
- 17) Unreal Engine. URL: <https://www.unrealengine.com/> (дата звернення: 05.05.2025).
- 18) Characters. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/characters-in-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 19) Actors. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/actors-in-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 20) Widget Blueprints. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/widget-blueprints-in-umg-for-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 21) Player Controllers. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/player-controllers-in-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 22) Levels. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/levels-in-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 23) Saving and Loading Your Game. *Unreal Engine 5.2 Documentation*. URL: <https://dev.epicgames.com/documentation/en-us/unreal-engine/saving-and-loading-your-game-in-unreal-engine> (дата звернення: 07.05.2025).

- 24) Audio Engine Overview. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/audio-engine-overview-in-unreal-engine?application_version=5.2 (дата звернення: 07.05.2025).
- 25) Game Analytics: Maximizing the Value of Player Data / ed. by M.S. El-Nasr, A. Drachen, A. Canossa. London : Springer, 2013. 800 с.
- 26) Pellicone A., et al. Playing Aloud: Leveraging Game Commentary Culture for Playtesting. *IJGCMS*. 2022. Vol. 14, no. 1. С. 1-16. URL: <https://doi.org/10.4018/IJGCMS.296705> (дата звернення: 19.05.2025).
- 27) Claypool K.T., Claypool M. On frame rate and player performance in first person shooter games. *Multimedia Systems* 2007. No. 13. С. 3-17. URL: <https://doi.org/10.1007/s00530-007-0081-1> (дата звернення: 20.05.2025).
- 28) Schultz C. P., Bryant R. D. *Game Testing: All in One*. 3rd ed. Mercury Learning and Information, 2016. 500 с.
- 29) Releasing Your Project. *Unreal Engine 5.2 Documentation*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/preparing-unreal-engine-projects-for-release?application_version=5.2 (дата звернення: 26.05.2025).
- 30) InstallForge. URL: <https://installforge.net/> (дата звернення: 26.05.2025).
- 31) itch.io. URL: <https://itch.io/> (дата звернення: 29.05.2025).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Дата звіту

5/30/2025

Дата редагування

5/31/2025

✓

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute

Заголовок
ІП-15_Шляхтун_ПЗ

Автор Науковий керівник / Експерт
ШляхтунСарнацький В.В.

підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



Рисунок А.1 – Звіт подібності

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ГРА У ЖАНРІ СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ З ПОДІЙНО-
ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ**

Текст програми

КПІ.ПІ-1530.045490.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Денис ШЛЯХТУН

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
<https://github.com/Denys-Shliakhtun/RTSGame>

Файл CPP_Unit.cpp

Реалізація функціональної задачі керування юнітами, що виконують накази, у фрагменті наведено виконання наказу на атаку.

```
void ACP_P_Unit::AttackThisActor(AActor* ActorToAttack)
{
    if (!IsValid(ActorToAttack))
    {
        StopAttack();
        return;
    }

    ActorToGo = ActorToAttack;
    HandleNewUnitState(E_UnitState::ChaseToAttack);

    // Bind OnDestroyed event of the target
    ActorToAttack->OnDestroyed.AddDynamic(this,
    &ACP_P_Unit::OnTargetDestroyed);

    // Move to target
    AAIController* AICon = Cast<AAIController>(GetController());
    if (AICon && ActorToGo)
    {
        FAIRequestID MoveRequestID = AICon->MoveToActor(ActorToGo,
AttackRange);
        if (!MoveRequestID.IsValid())
        {
            AttackThisActor(ActorToGo); // Retry logic
            return;
        }
    }

    // Ensure the actor is still valid after initiating movement
    if (!IsValid(ActorToGo))
    {
        StopAttack();
        return;
    }

    // Rotate unit to face the target (only Z axis)
    FRotator LookAtRotation =
UKismetMathLibrary::FindLookAtRotation(GetActorLocation(), ActorToGo-
>GetActorLocation());
    SetActorRotation(FRotator(0.f, LookAtRotation.Yaw, 0.f));

    HandleNewUnitState(E_UnitState::Attacking);

    // Start attack timer (1 second interval)
    GetWorldTimerManager().SetTimer(AttackTimerHandle, this,
&ACP_P_Unit::Attack, AttackTime, true);
}

void ACP_P_Unit::StopAttack()
{
    HandleNewUnitState(E_UnitState::Idle);
    GetWorld()->GetTimerManager().ClearTimer(AttackTimerHandle);
}
```

```

}

void ACP_P_Unit::Attack()
{
    // If the target actor is invalid, stop attacking and return
    if (!IsValid(ActorToGo))
    {
        StopAttack();
        return;
    }

    // Check if the target is within range and unit is in Attacking state
    float DistanceToTarget = FVector::Dist(GetActorLocation(), ActorToGo-
>GetActorLocation());
    if (DistanceToTarget <= AttackRange && UnitState ==
E_UnitState::Attacking)
    {
        UGameplayStatics::ApplyDamage(ActorToGo, DamageToApply,
GetController(), this, nullptr);
        return;
    }

    // Otherwise, clear and invalidate the timer
    GetWorld()->GetTimerManager().ClearTimer(AttackTimerHandle);

    // If still attacking and target is valid, reattempt attack
    if (UnitState == E_UnitState::Attacking && IsValid(ActorToGo))
    {
        AttackThisActor(ActorToGo);
    }
    else
    {
        StopAttack();
    }
}

void ACP_P_Unit::HandleNewUnitState(E_UnitState NewState)
{
    UnitState = NewState;
}

void ACP_P_Unit::OnTargetDestroyed(AActor* DestroyedActor)
{
    StopAttack();
}

```

Файл UCPP_BuildingComponent.cpp

Реалізація функціональної задачі керування будівлями, у фрагменті наведено створення юнітів за ресурси.

```

void UCPP_BuildingComponent::ProducingUnitsCycle()
{
    // If the queue is empty, stop producing units
    if (UnitProductionQueue.IsEmpty())
    {
        IsProducingUnits = false;
        return;
    }

    IsProducingUnits = true;

    APlayerController* PC = GetWorld()->GetFirstPlayerController();

```

```

if (!PC) return;

// Get the first unit class in the queue and info about it
TSubclassOf<AActor> UnitClass = UnitProductionQueue[0];
FS_UnitPriceAndTime UnitPriceAndTime = GetUnitPriceAndTime(UnitClass);
int32 Gold = GetResource(TEXT("Gold"));
int32 Housing = GetResource(TEXT("Housing"));

// If enough gold and housing is available
if (UnitPriceAndTime.Price <= Gold && Housing > 0)
{
    ModifyResource(TEXT("Gold"), -UnitPriceAndTime.Price);

    // Find all components named "Goal" as valid spawn locations
    TArray<UActorComponent*> GoalComponents = GetOwner()-
>GetComponentsByClass(USceneComponent::StaticClass());
    TArray<USceneComponent*> ValidLocations;

    for (UActorComponent* Comp : GoalComponents)
    {
        if (Comp->GetName().Contains(TEXT("Goal")))
        {
            if (USceneComponent* Scene = Cast<USceneComponent>(Comp))
            {
                ValidLocations.Add(Scene);
            }
        }
    }

    // If at least one valid spawn location is found
    if (ValidLocations.Num() > 0)
    {
        // Pick a random spawn location
        FVector SpawnLocation = ValidLocations[FMath::RandRange(0,
ValidLocations.Num() - 1)]->GetComponentLocation();

        // Set a timer to simulate unit production time
        FTimerHandle TimerHandle;
        GetWorld()->GetTimerManager().SetTimer(TimerHandle, [this,
UnitClass, SpawnLocation]()
        {
            // Spawn the unit
            AActor* SpawnedUnit = GetWorld()-
>SpawnActor<AActor>(UnitClass, SpawnLocation, FRotator::ZeroRotator);
            if (SpawnedUnit)
            {
                // Notify that the unit was created
                OnUnitCreated.Broadcast(SpawnedUnit);
            }

            // Remove the produced unit from the queue
            UnitProductionQueue.RemoveAt(0);

            // Continue with the next unit
            ProducingUnitsCycle();

            }, UnitPriceAndTime.TimeToProduce, false);

        return;
    }
}

// If failed to produce unit, remove the unit and continue
UnitProductionQueue.RemoveAt(0);

```

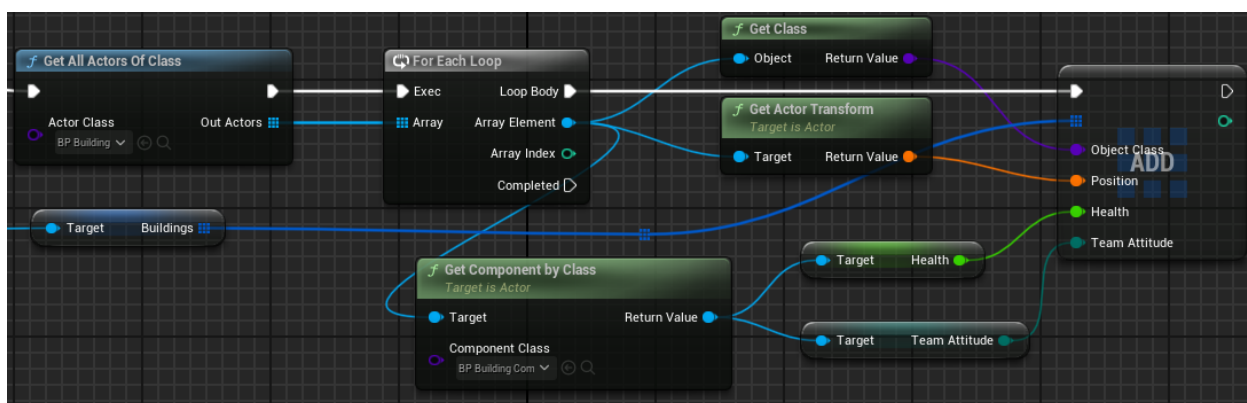
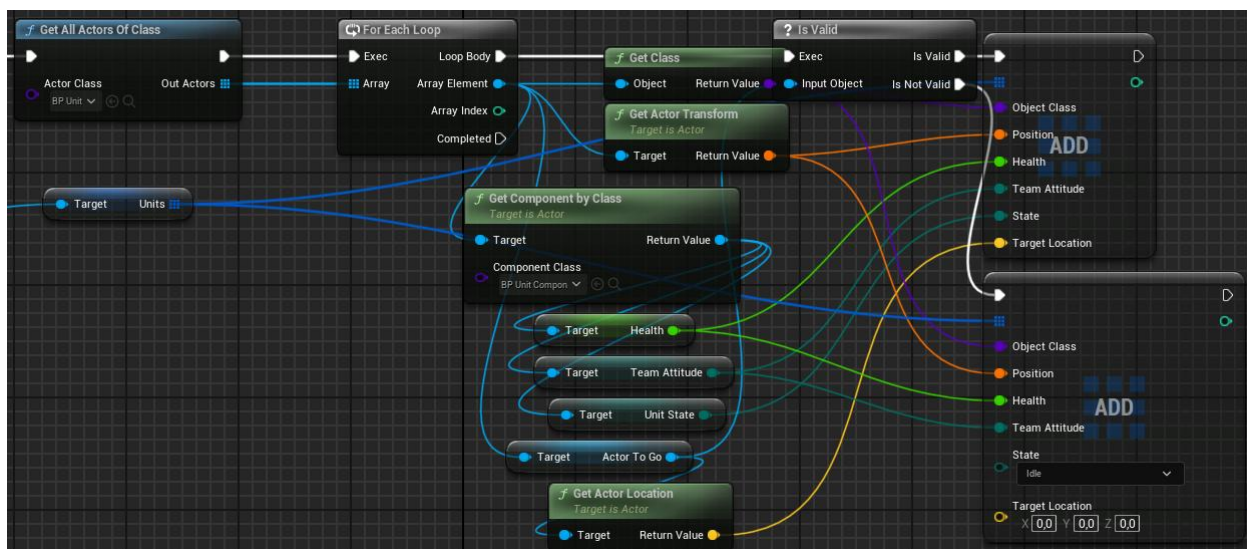
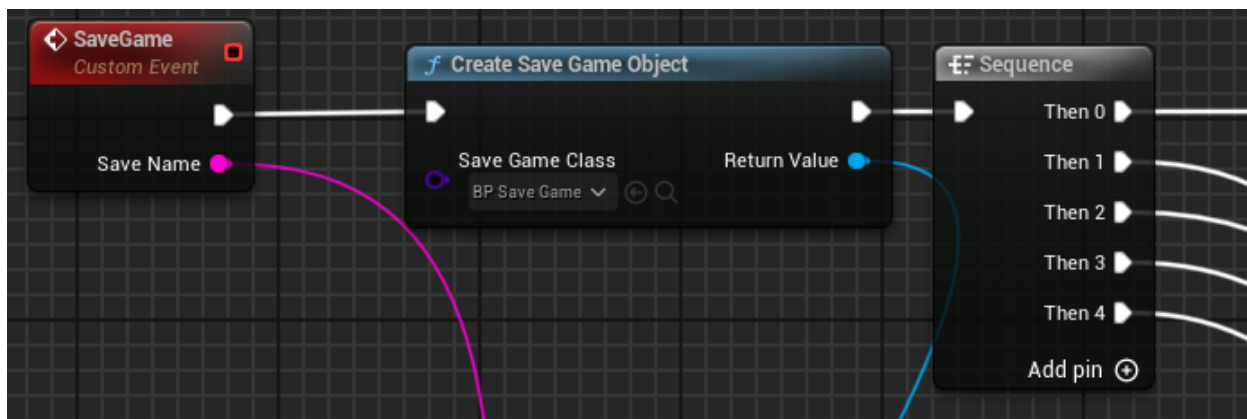
```

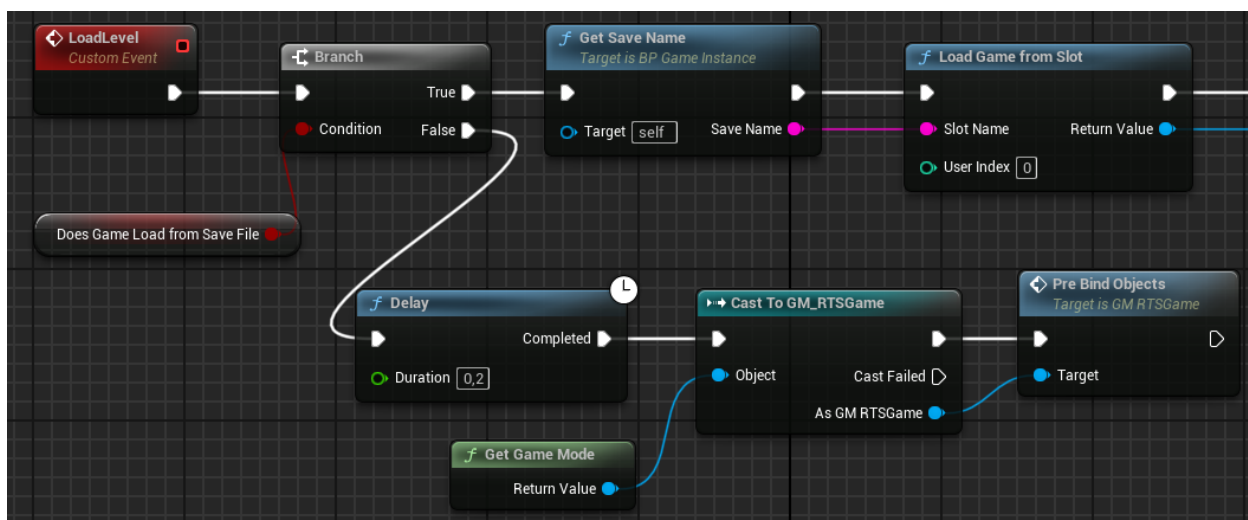
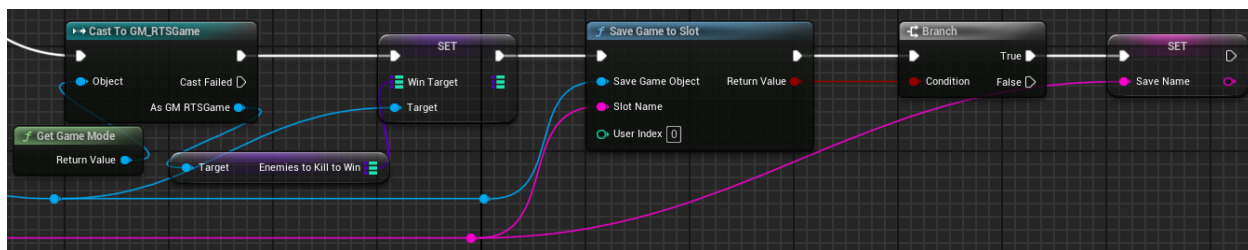
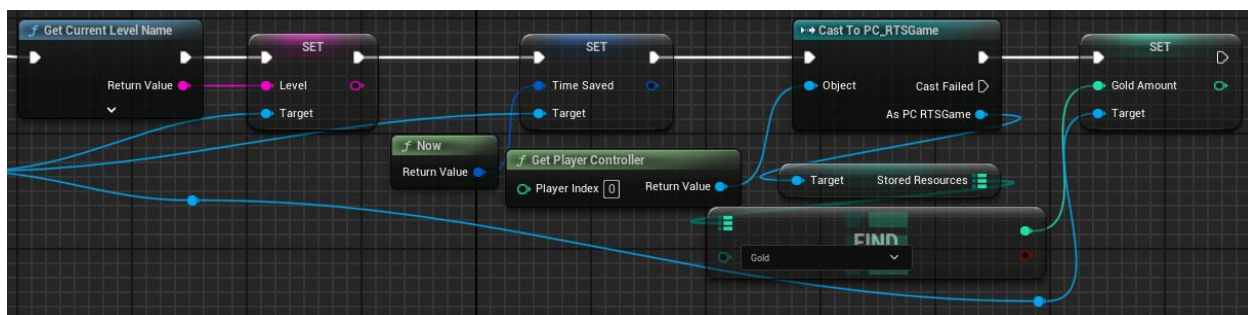
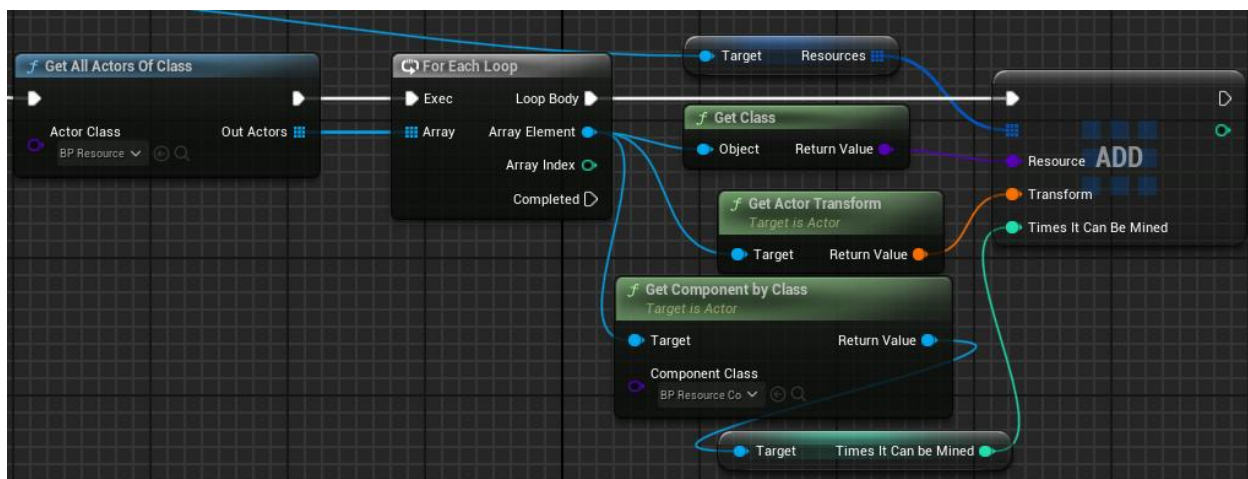
    ProducingUnitsCycle();
}

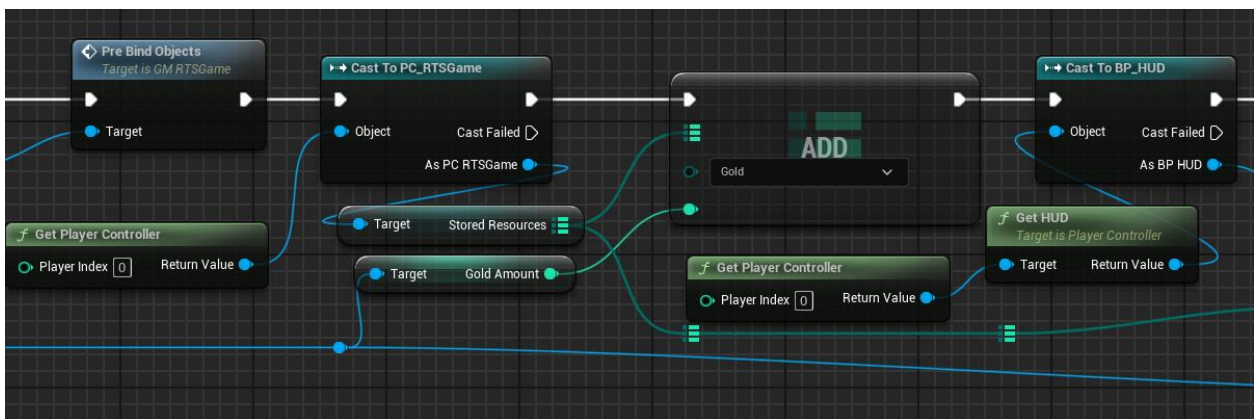
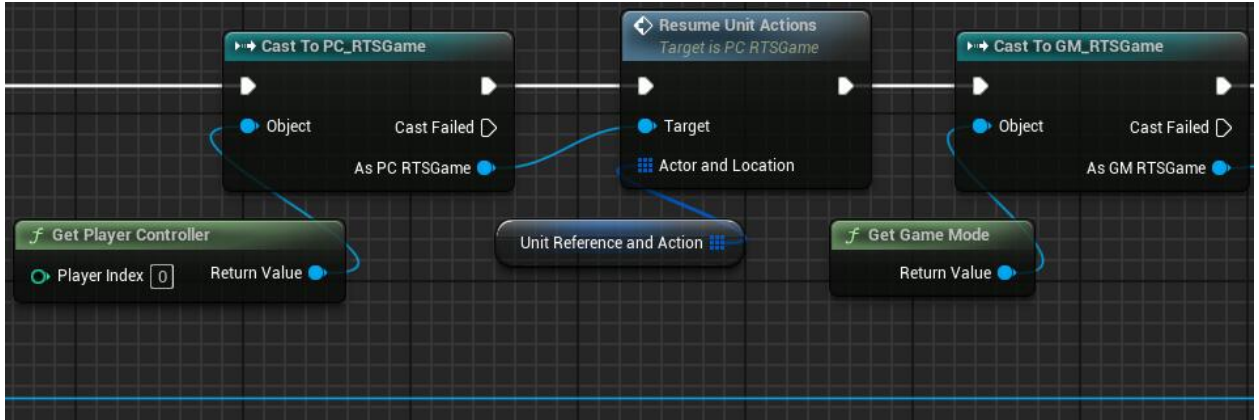
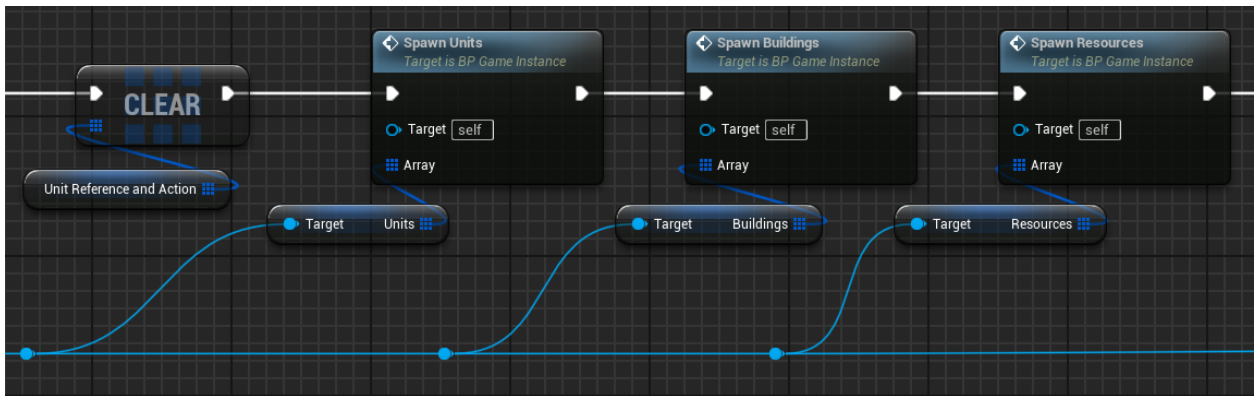
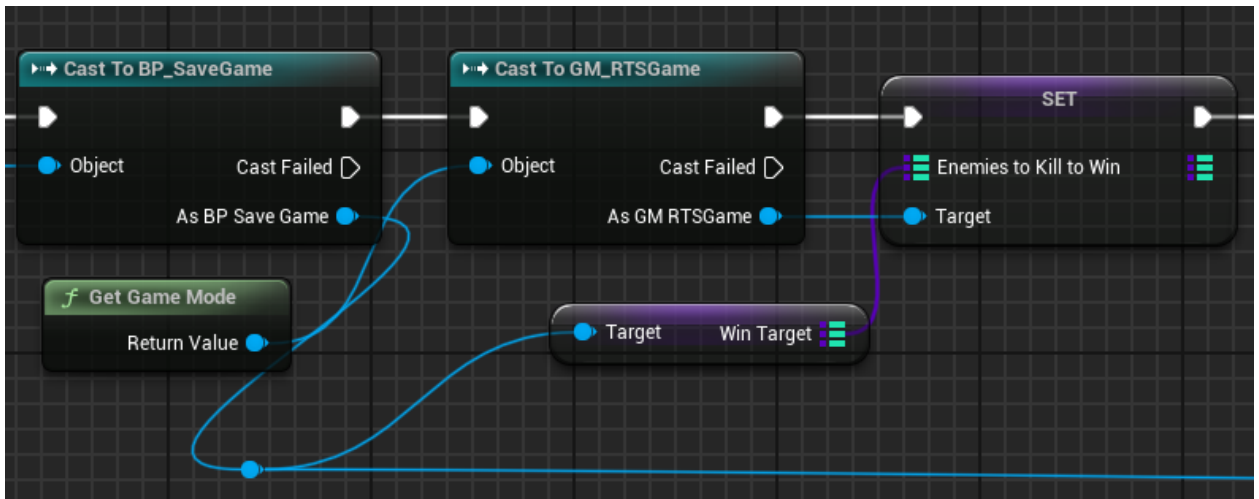
```

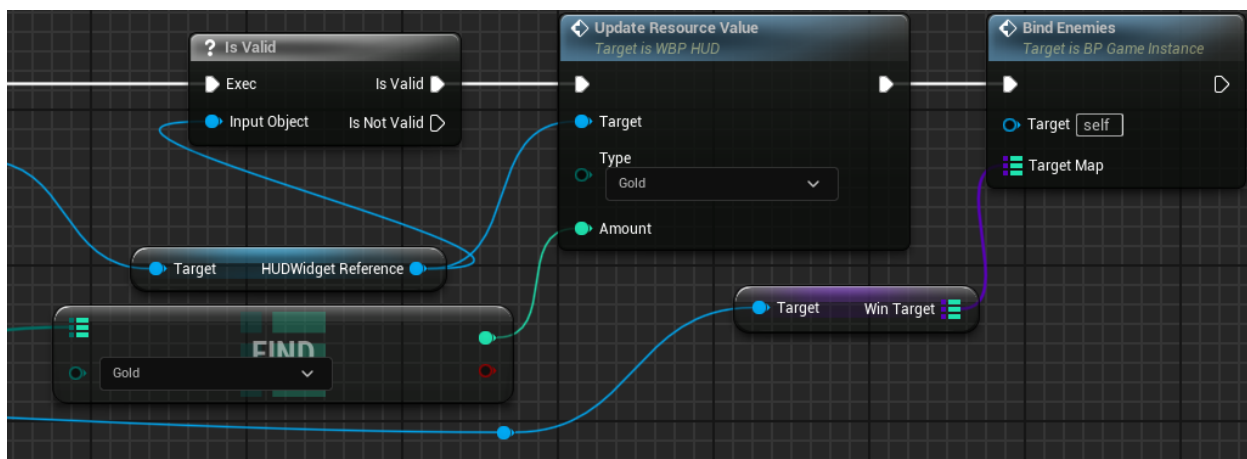
Файл BP_GameInstance

Реалізація функціональної задачі збереження та завантаження прогресу гри за допомогою Blueprints.



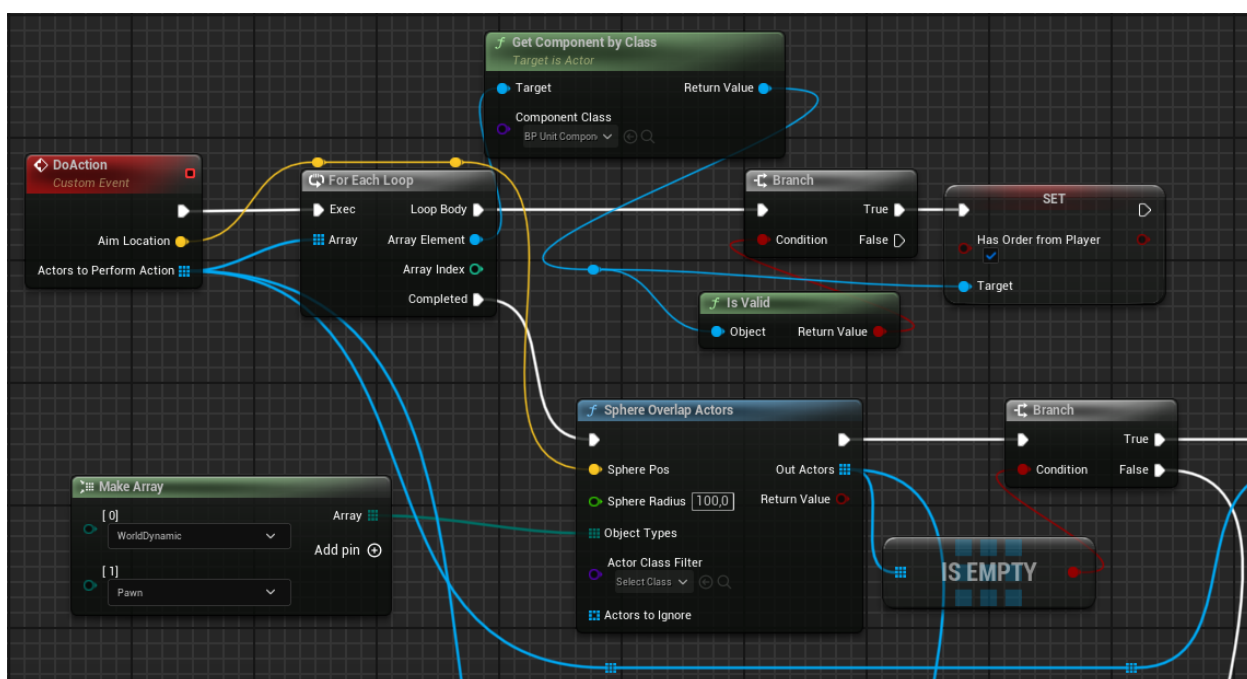
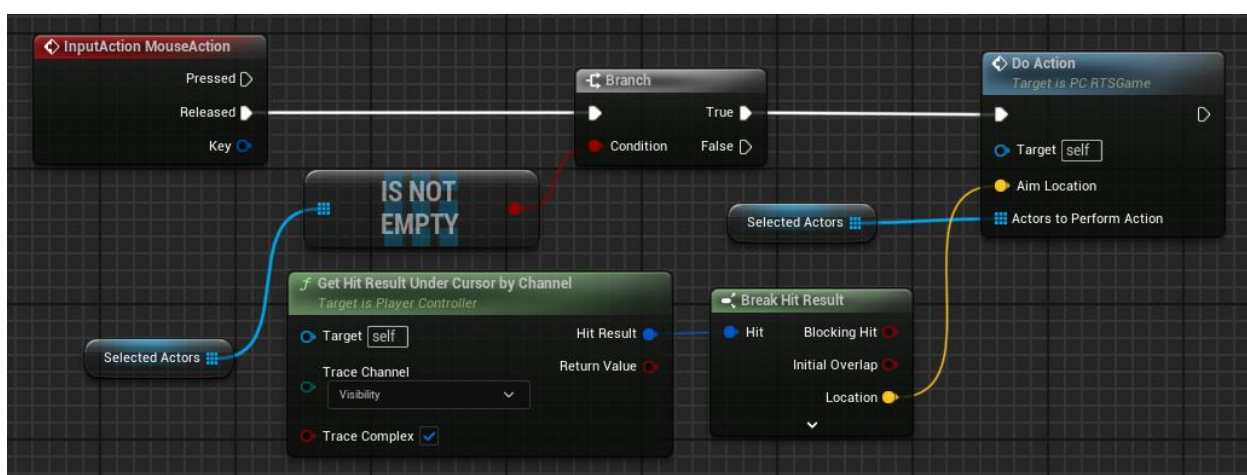


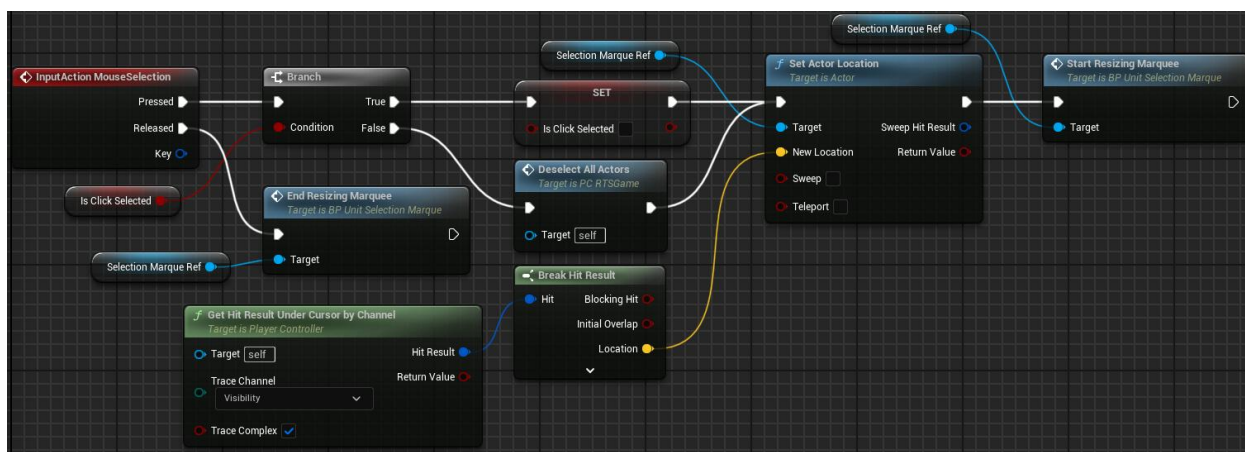
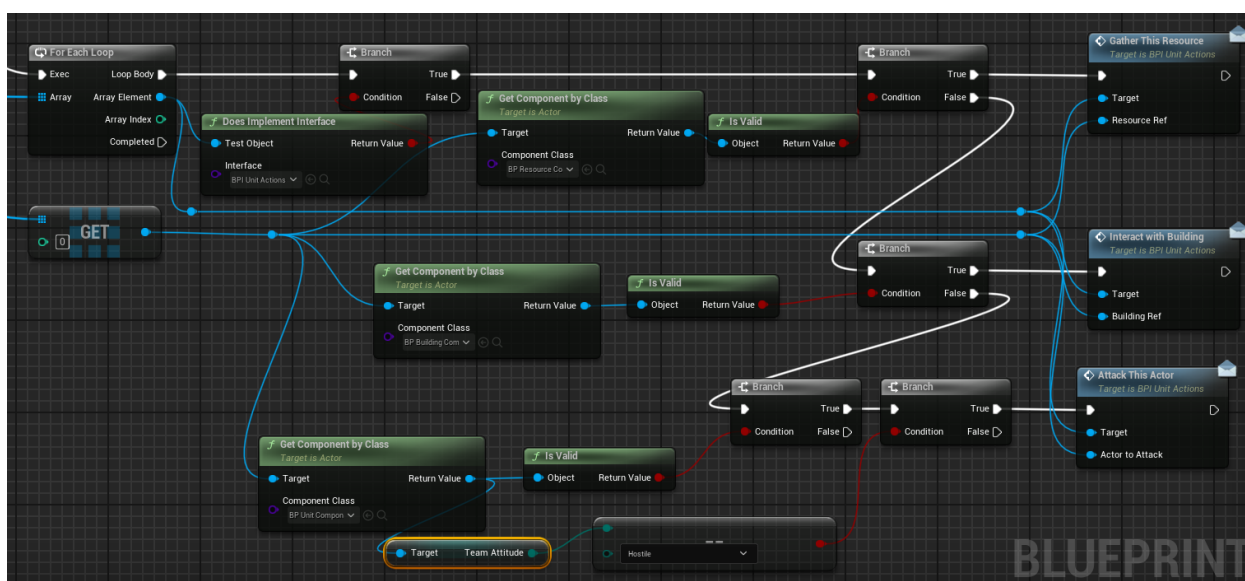
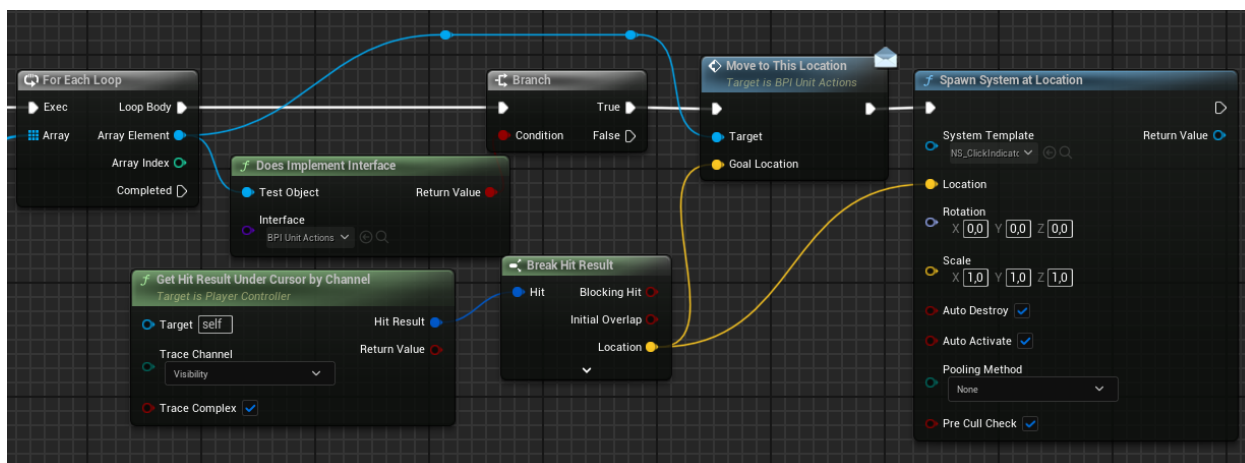




Файл PC_RTSGame

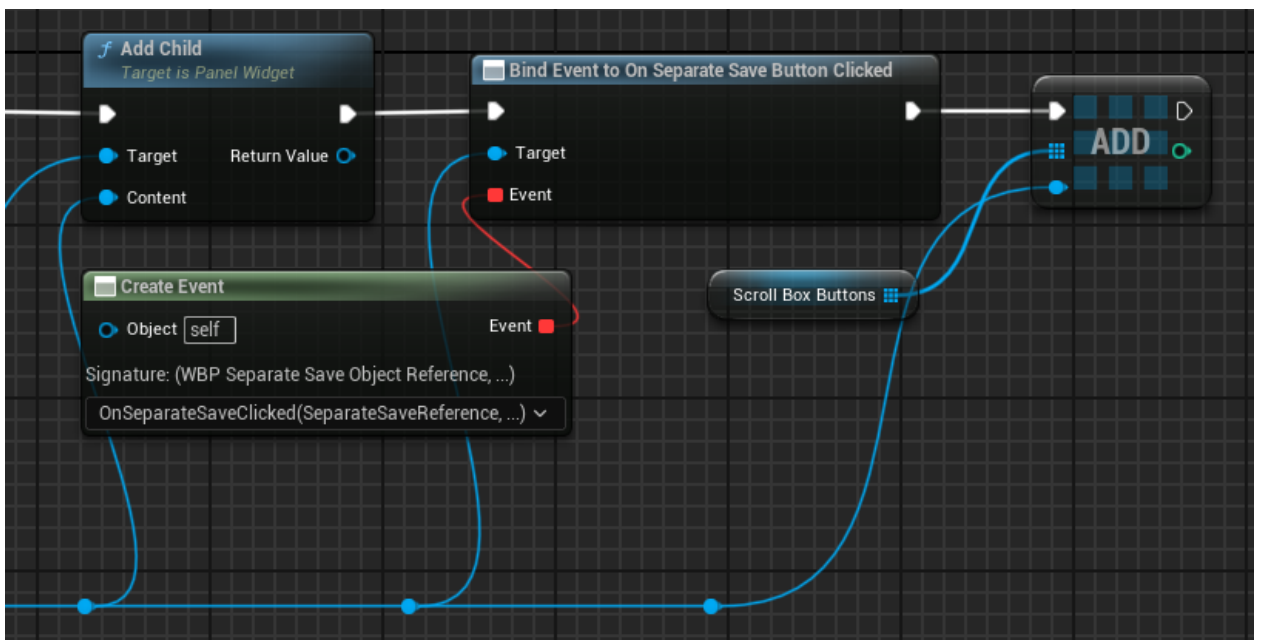
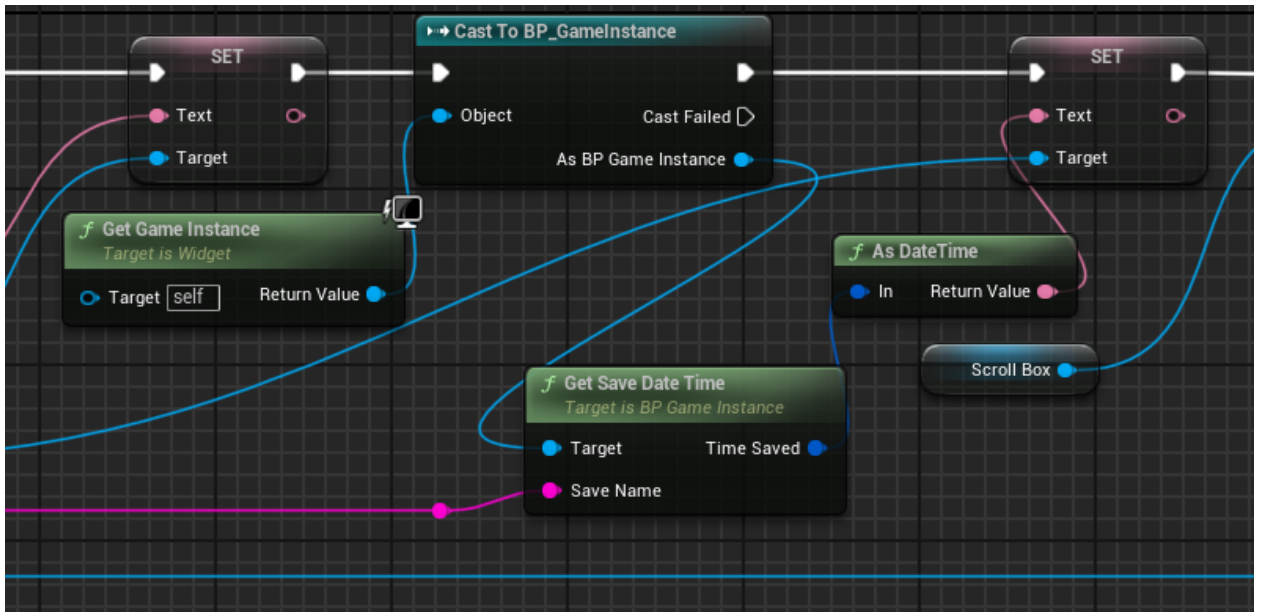
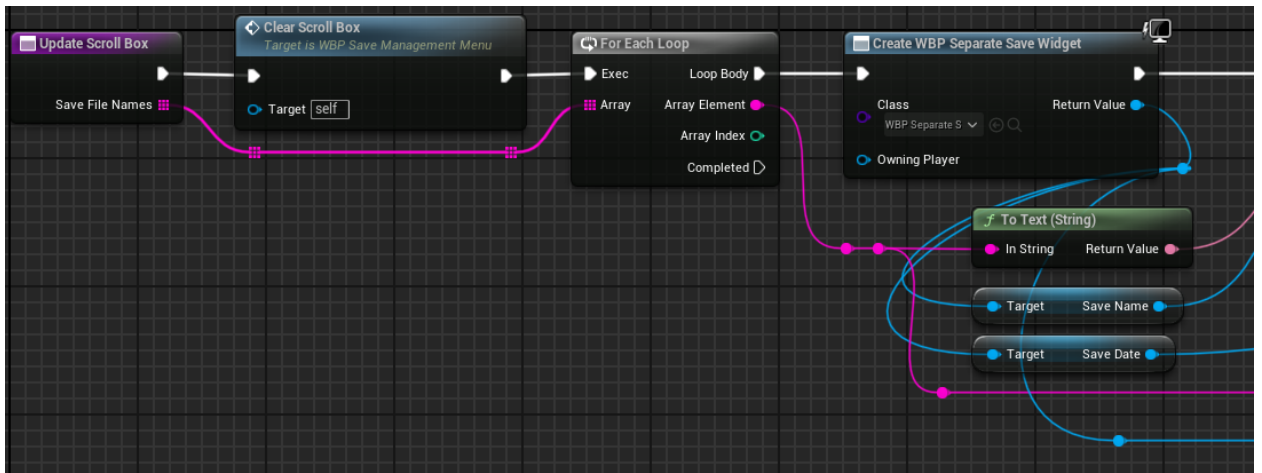
Реалізація функціональної вимоги керування ігровим процесом, наведено фрагмент Blueprints з видачею наказу юнітам правою кнопкою миші.





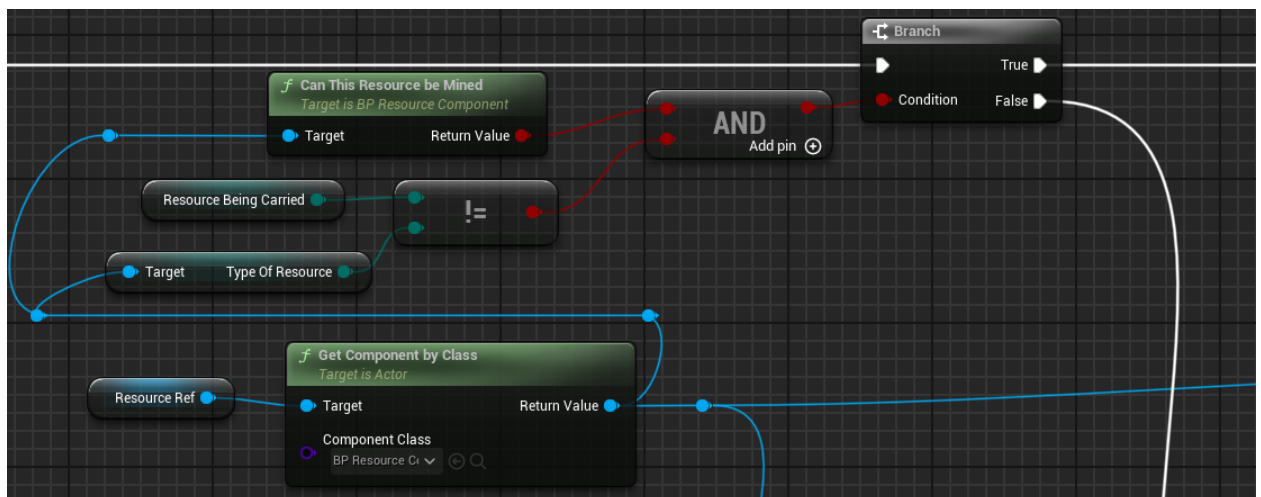
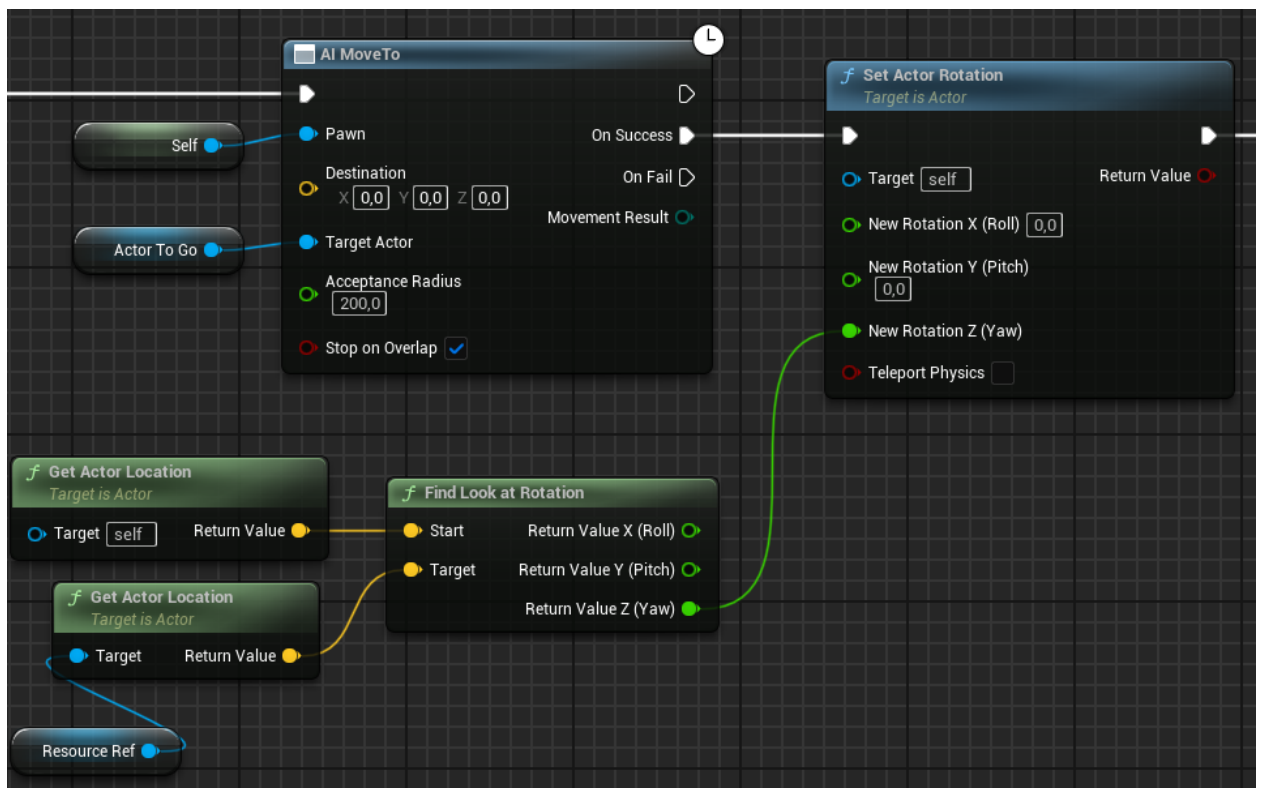
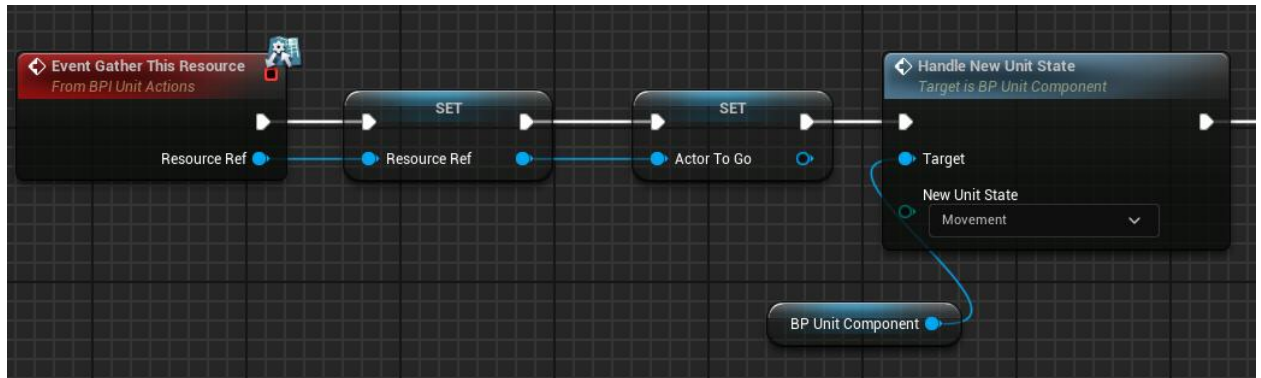
Файл WBP_SaveManagementMenu

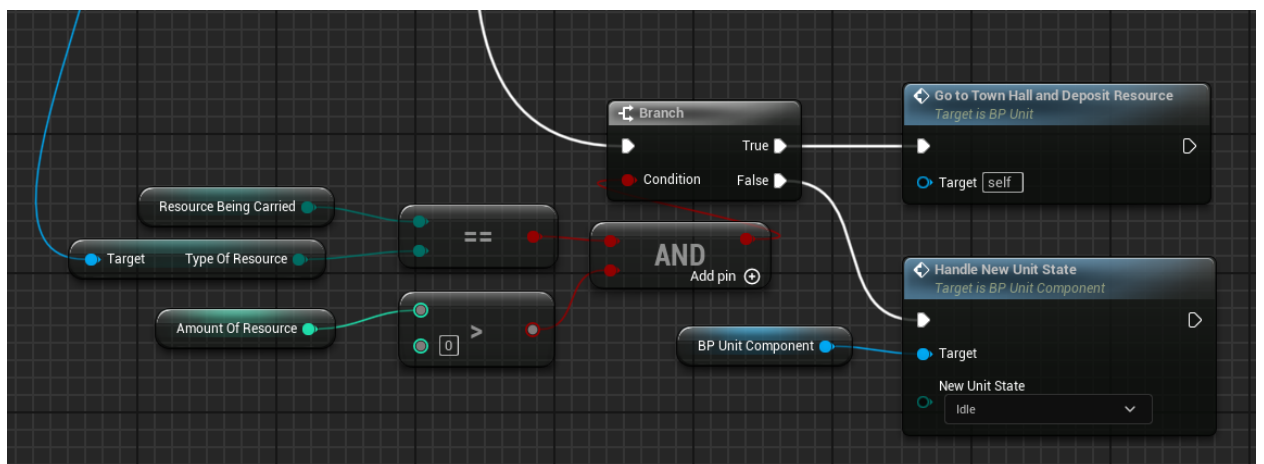
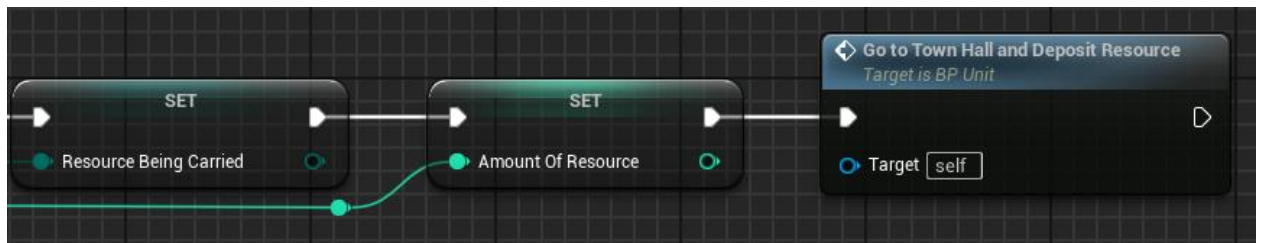
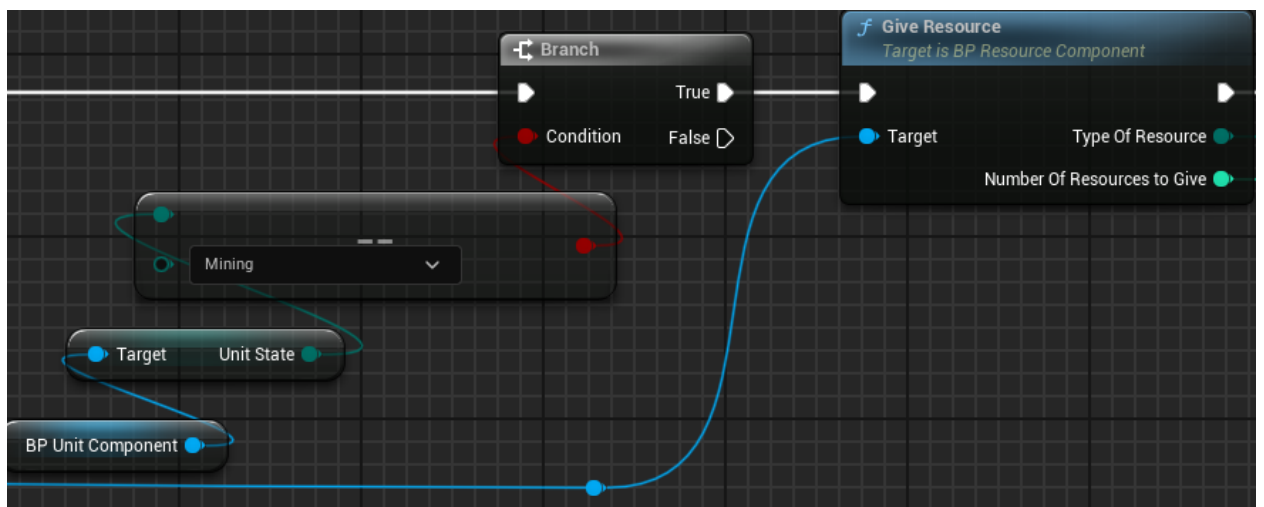
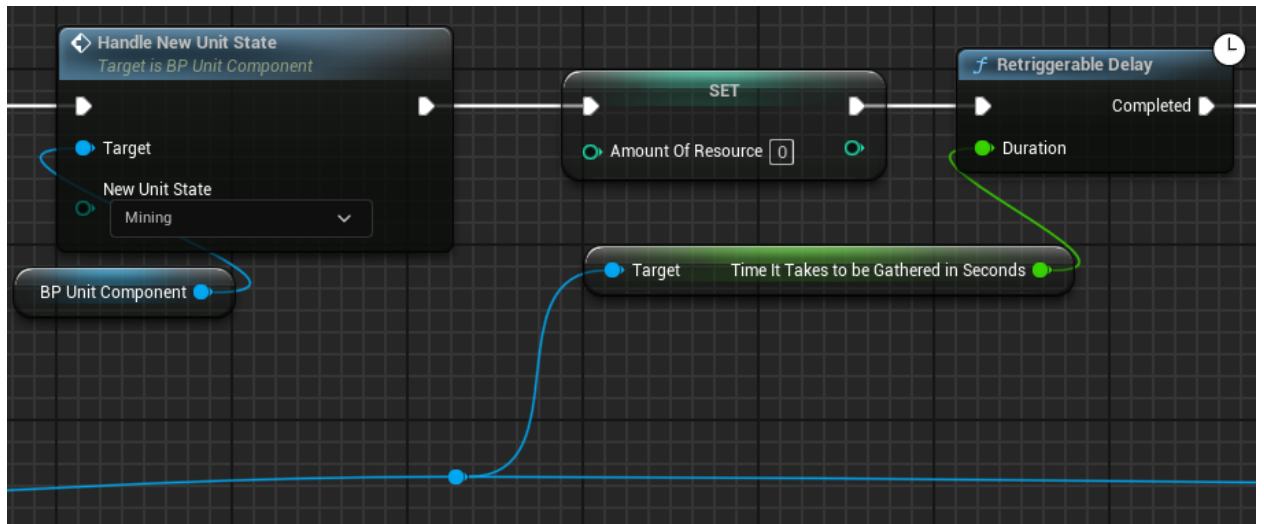
Реалізація функціональної задачі користувацького інтерфейсу на прикладі збереження та завантаження прогресу гри, наведено фрагмент Blueprints наповнення списку збережень.

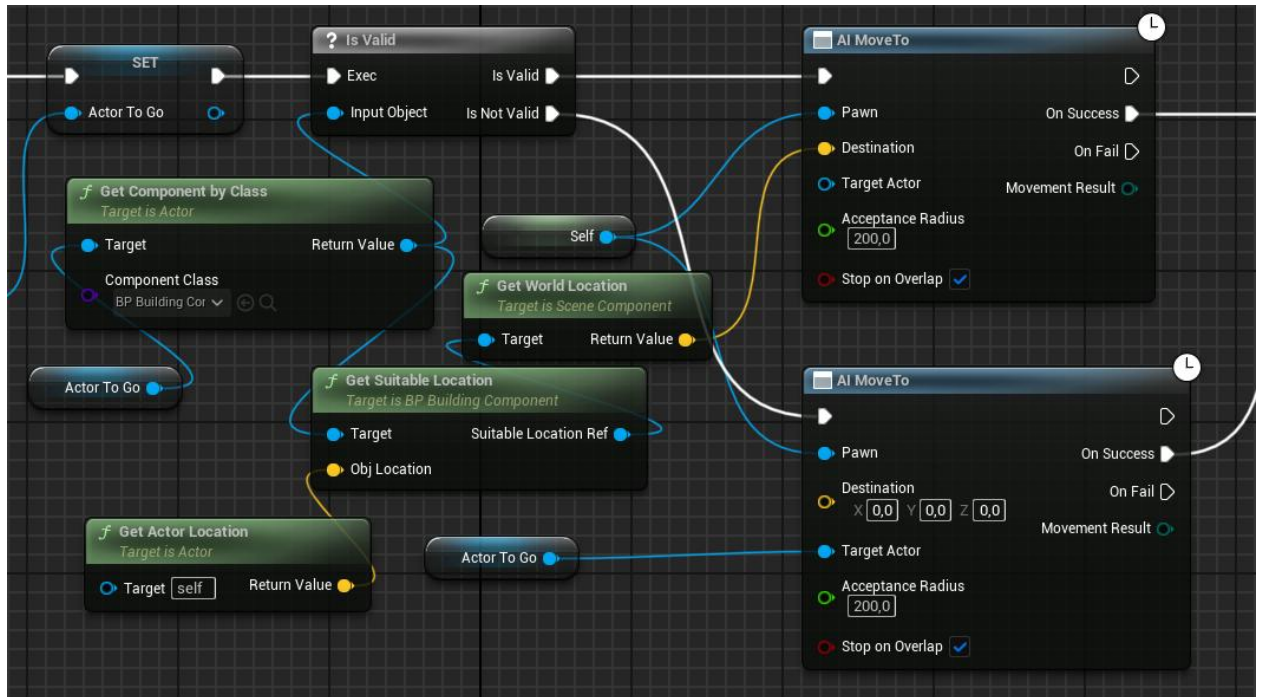
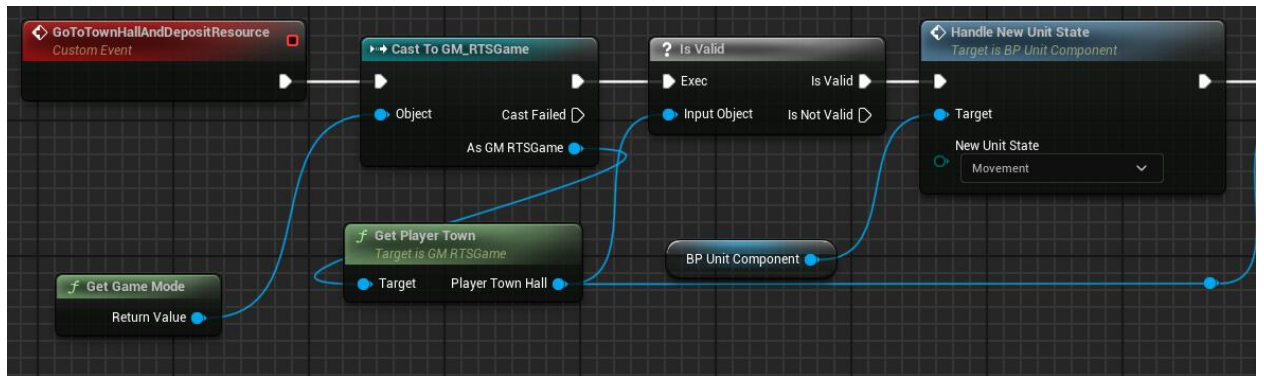


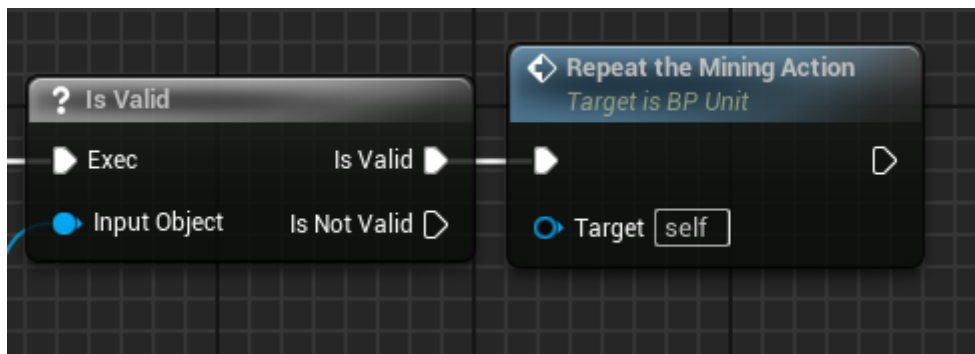
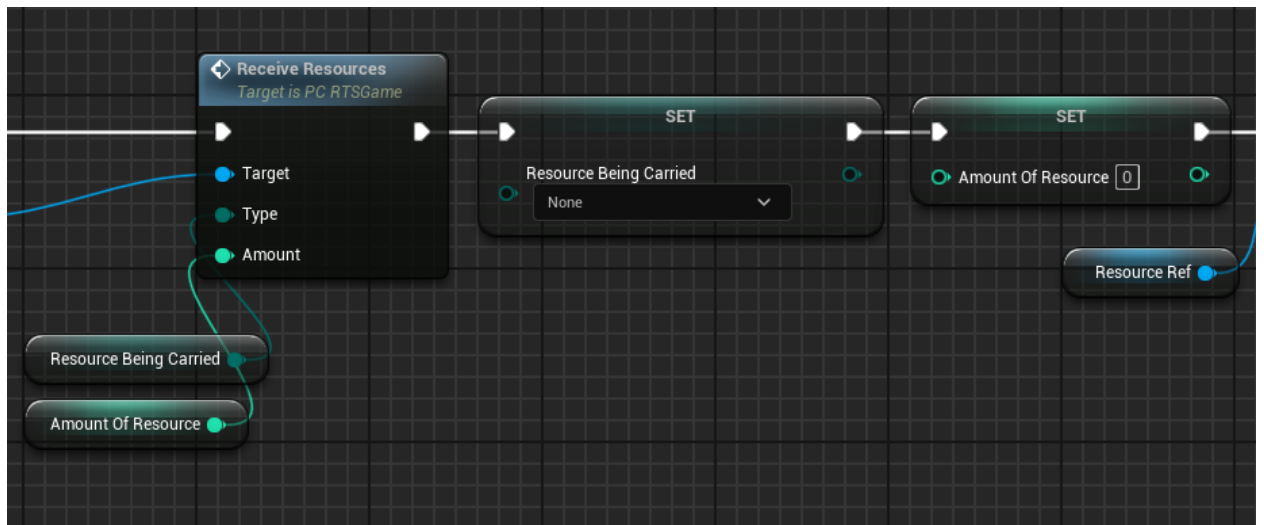
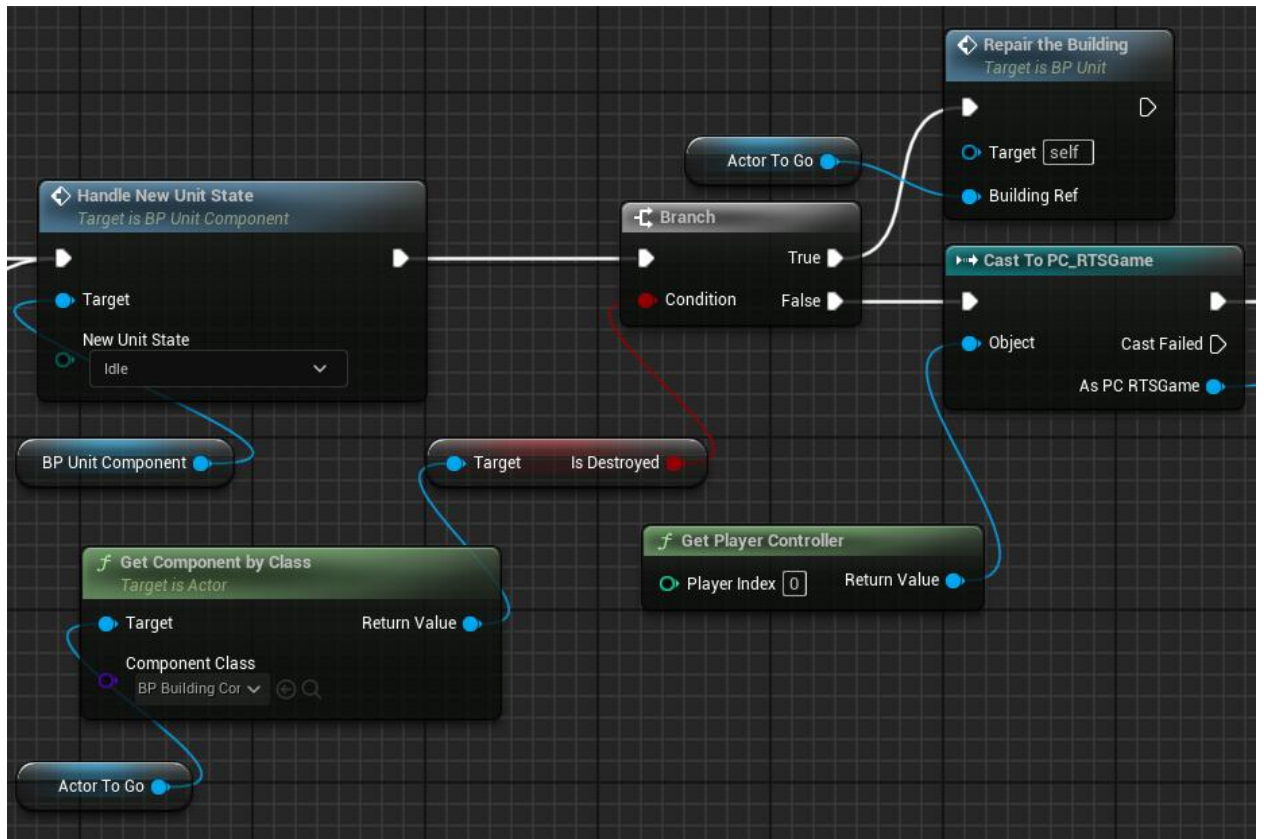
Файл BP_Worker

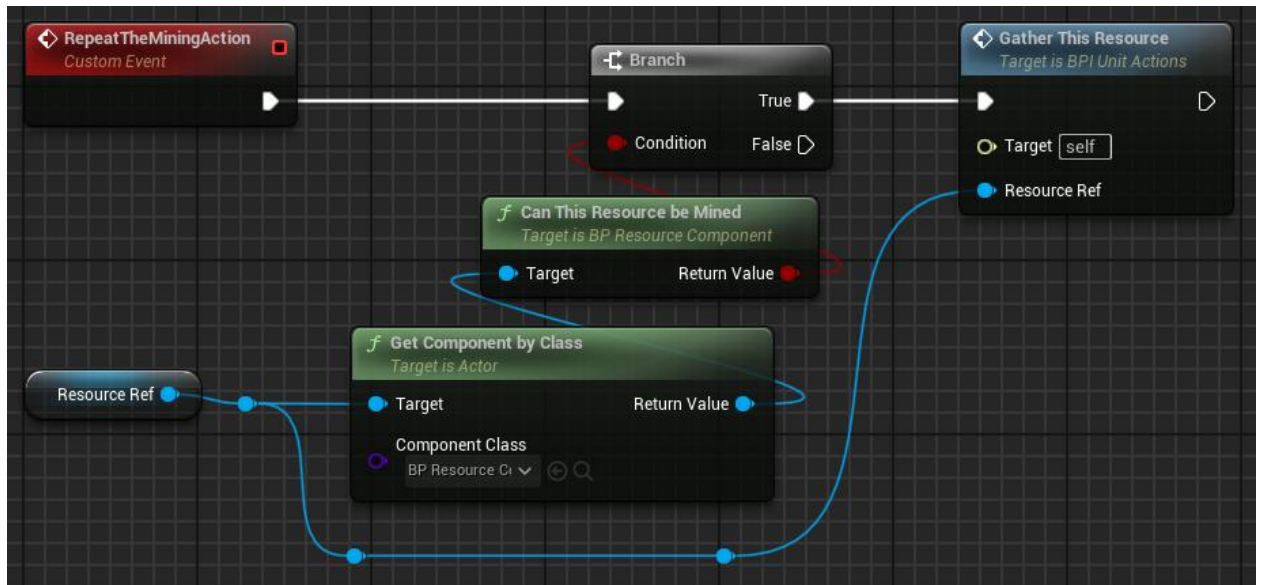
Реалізація функціональної задачі керування юнітами, що виконують накази, у фрагменті Blueprints наведено виконання наказу на добування ресурсів.











Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ГРА У ЖАНРІ СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ З ПОДІЙНО-
ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ**

Програма та методика тестування

КПІ.ПІ-1530.045490.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Денис ШЛЯХТУН

Київ – 2025

ЗМІСТ

1	ОБ'ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ.....	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є стратегія в реальному часі RTSGame на операційних системах Windows 10 та 11.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності гри з операційними системами Windows 10 та Windows 11;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;
- перевірка продуктивності гри;
- перевірка сумісності з різними роздільними здатностями екрану;
- перевірка стабільності роботи на відсутність зависань та завершень роботи.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- модульне тестування (юніт-тести) – полягає у перевірці окремих, ізольованих одиниць програмного коду, таких як функції чи класи. Застосовується для впевненості у правильності базової логіки ще до етапу інтеграції компонентів;
- інтеграційне тестування – перевіряє коректність взаємодії між окремими модулями системи;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- плейтест – фокусується на оцінці загального досвіду користувача: зручності управління, логіки ігрового процесу, візуального сприйняття та ігрового балансу;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об'єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «білої скриньки» – об'єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним;
- тестування «сірої скриньки» – об'єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватися наступні допоміжні засоби:

- Profiler Tool рушія Unreal Engine для визначення продуктивності;
- Automation Test Framework рушія Unreal Engine.

Порядок проведення тестування буде наступним:

- модульне тестування на перевірку окремих одиниць коду;
- інтеграційне тестування на коректність взаємодії між компонентами;
- функціональне тестування на перевірку відповідності реальної поведінки очікуваній;
- плейтест на перевірку залученості гравців та стабільності роботи;
- динамічне тестування на відповідність функціональним вимогам;
- тестування на дисплеях з різною роздільною здатністю екрана;
- тестування на пристроях з різною версією операційної системи.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ГРА У ЖАНРІ СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ З ПОДІЙНО-
ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ**

Керівництво користувача

КПІ.ПІ-1530.045490.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Денис ШЛЯХТУН

Київ – 2025

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи.....	7
3	ВИКОНАННЯ ПРОГРАМИ	8

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«RTSGame» – це гра у жанрі стратегії в реальному часі для операційних систем Windows 10 та 11. Це спрощена військова симуляція, у якій гравець непрямо керує багатьма юнітами та будівлями, віддаючи накази з висоти пташиного польоту в реальному часі, щоб збирати ресурси, споруджувати армію, а також знищувати сили супротивника. Користувач може завантажити з репозиторію інсталятор та встановити гру на своєму персональному комп'ютері.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи гри необхідне виконання мінімальних вимоги:

- тип процесору: 4-ядерний процесор 2.5 ГГц;
- об'єм ОЗП: 8 Гб;
- відеокарта: NVIDIA GeForce GTX 960;
- ОС Windows 10 64-bit або Windows 11;
- місце на жорсткому диску: 1 Гб;
- миша та клавіатура.

Рекомендована конфігурація комп'ютера:

- тип процесору: Intel Core i7-10750H;
- об'єм ОЗП: 16 Гб;
- відеокарта: NVIDIA GeForce GTX 1060 Ti.

2.2 Завантаження застосунку

На даний момент гру можна встановити власноруч, використовуючи відповідний інсталятор. Для цього необхідно завантажити його у вкладці Releases репозиторію GitHub, запустити та виконати відповідні вказівки. На рисунку 2.1 відображено початковий екран інсталятора.

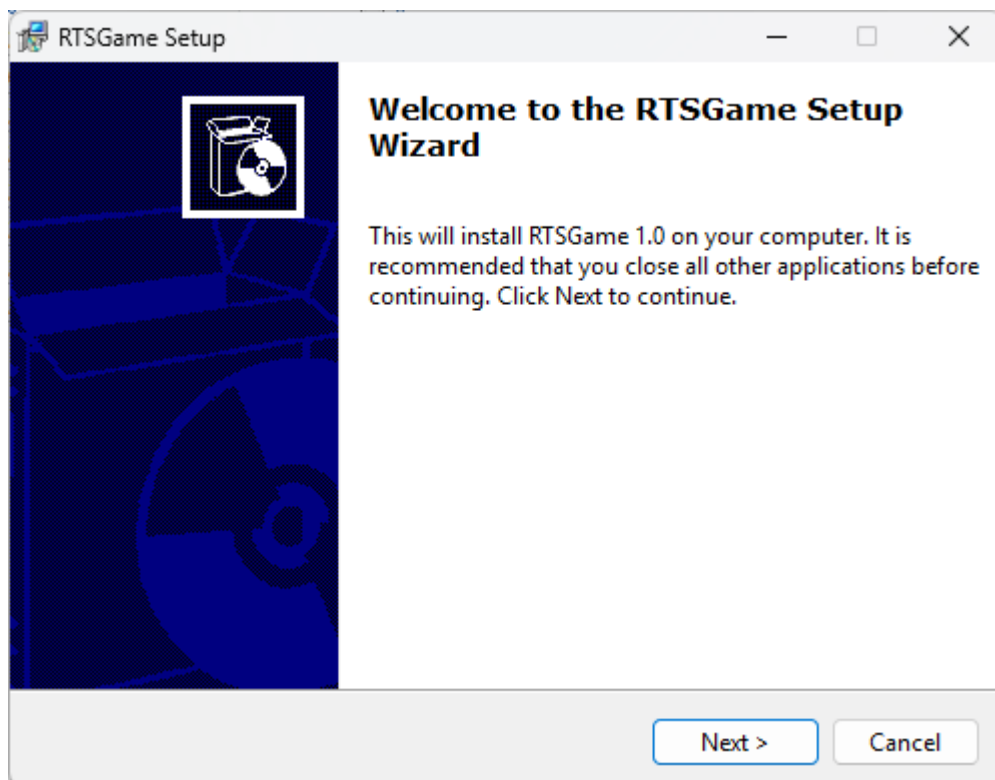


Рисунок 2.1 – Інсталятор RTSGame

Після натискання кнопки «Next» потрібно задати шлях встановлення гри у відповідному полі, що відображено на рисунку 2.2.

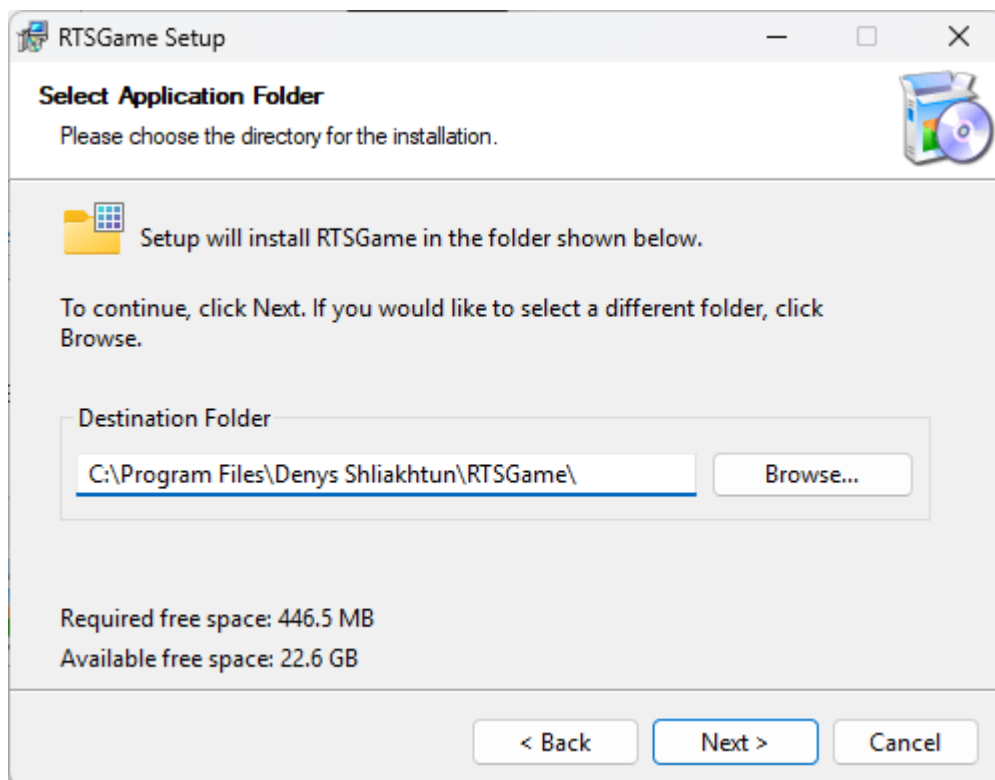


Рисунок 2.2 – Вибір шляху встановлення

Після натискання кнопки «Next» здійснюється встановлення гри (рисунок 2.3).

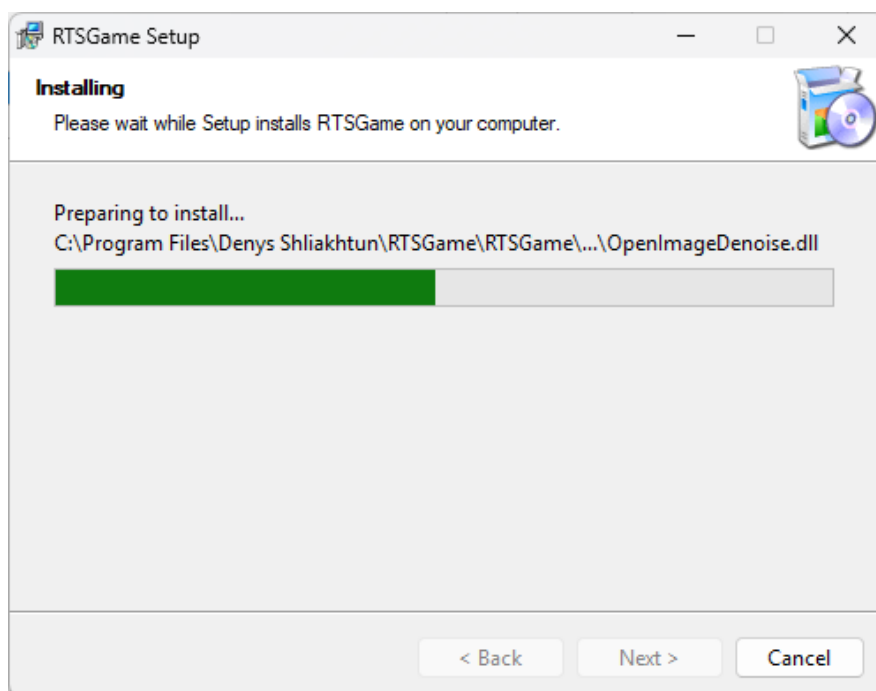


Рисунок 2.3 – Встановлення гри

Після завершення встановлення можна закривати інсталятор кнопкою «Finish» (рисунок 2.4).

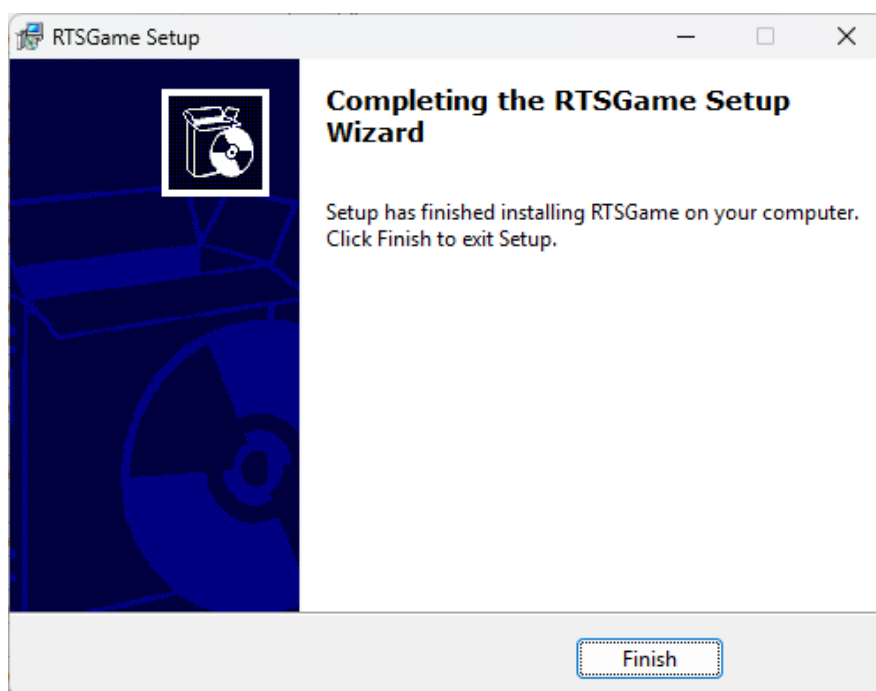


Рисунок 2.4 – Завершення встановлення гри

2.3 Перевірка коректної роботи

Після завершення встановлення гра з'явиться у папці, вказаній при встановленні. При запуску файлу RTSGame.exe повинно відкритися нове вікно з головним меню гри. Якщо є проблеми із запуском на рекомендованій конфігурації технічних засобів, потрібно у папці гри перейти до Engine\Extras\Redist\en-us та запустити файл UEPrereqSetup_x64.exe для встановлення програмних засобів, необхідних для запуску гри.

3 ВИКОНАННЯ ПРОГРАМИ

Як було зазначено у підрозділі 2.3, для запуску гри потрібно запустити файл RTSGame.exe у папці, вказаній користувачем при встановленні. Після відкриття файлу відкриється вікно гри на головному меню (рисунок 3.1).

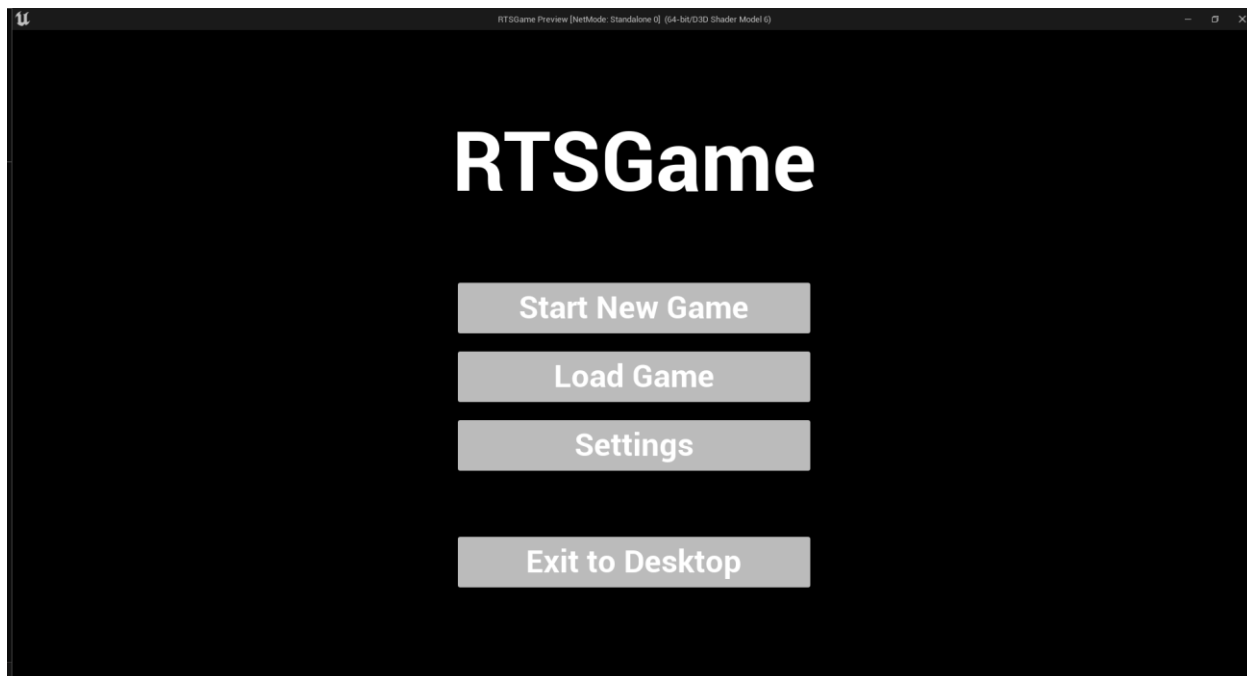


Рисунок 3.1 – Головне меню гри

Головне меню надає можливість переходу в меню створення нової гри (кнопка «Start New Game»), завантаження прогресу гри (кнопка «Load Game»), налаштувань (кнопка «Settings»), а також можливість вийти з гри (кнопка «Exit to Desktop»).

Користувач може запустити ігровий процес двома способами. Перший спосіб – почати нову гру за допомогою вибору рівня. На рисунку 3.2 відображено меню вибору місії.

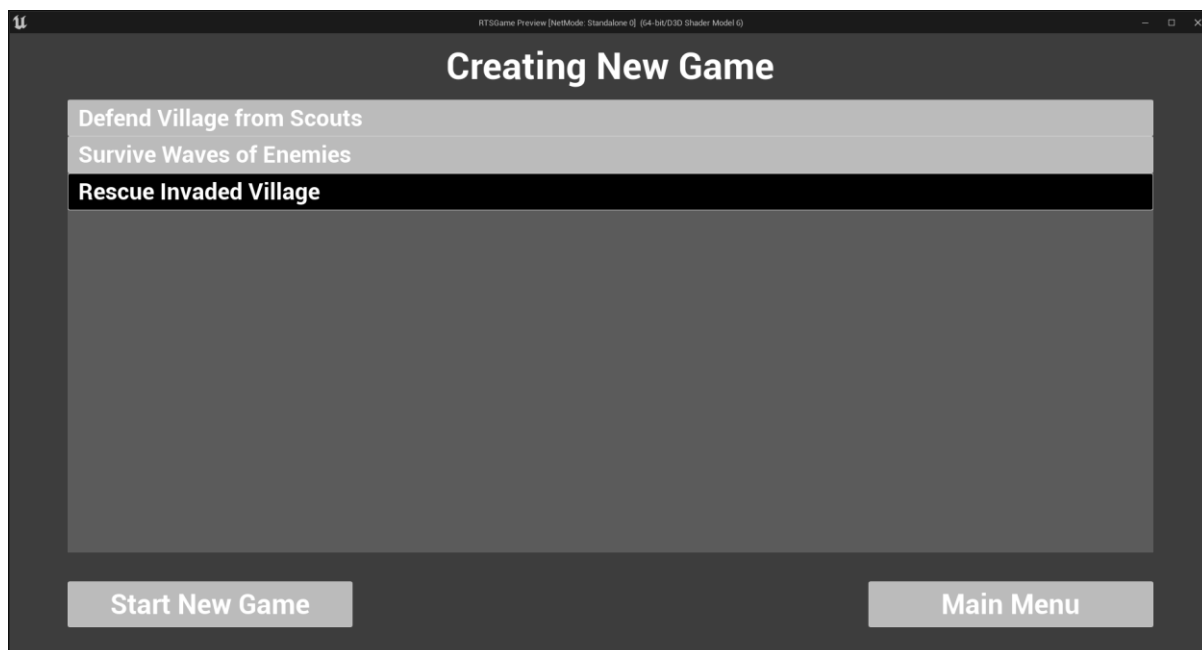


Рисунок 3.2 – Меню створення нової гри

Місії мають різне початкове розташування ігрових об'єктів та різні цілі для виконання. Користувач повинен виділити бажану місію лівою кнопкою миші, наприклад, на рисунку іншим кольором виділено місію «Rescue Invaded Village». Після натискання на кнопку «Start New Game» здійснюється завантаження рівня та перехід до ігрового процесу. Додатково, користувач може повернутися до головного меню натисканням кнопки «Main Menu».

Другий спосіб переходу до ігрового процесу – завантажити збереження. На рисунку 3.3 відображено меню завантаження збереження.

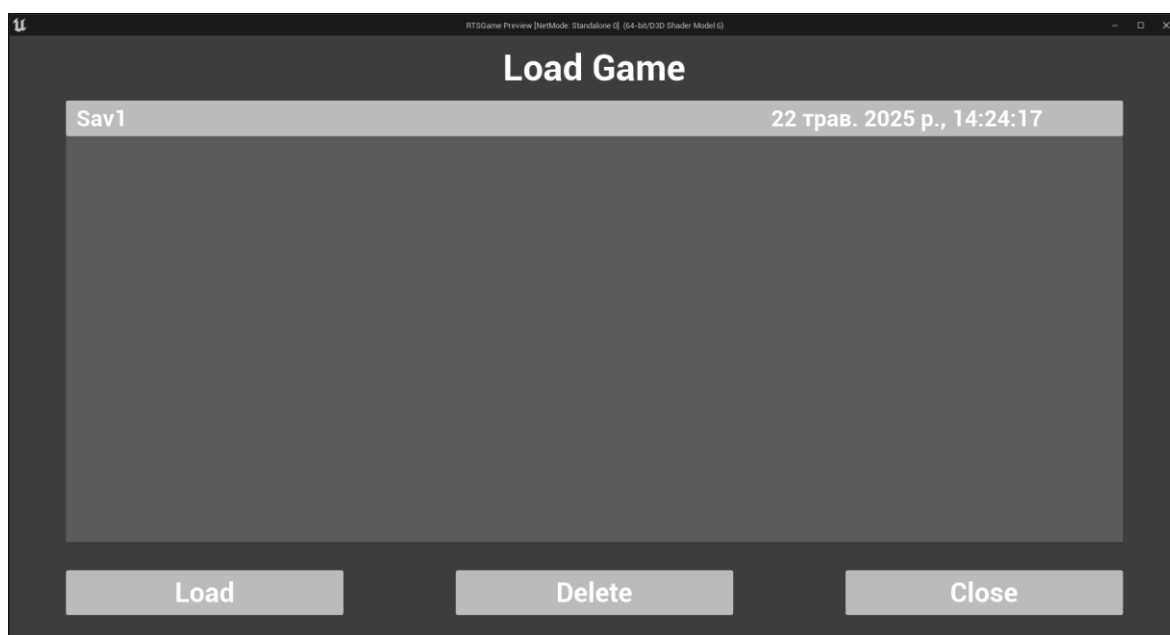


Рисунок 3.3 – Меню завантаження

Користувач повинен виділити збереження, яке він збирається завантажити, по аналогії з вибором рівня, а потім натиснути кнопку «Load». Після цього здійснюється завантаження рівня та перехід до ігрового процесу. Додатково, у меню завантаження користувач може видалити виділене збереження натисканням кнопки «Delete» або повернутися до головного меню натисканням кнопки «Close».

Після створення нової гри або завантаження прогресу відображається екран ігрового інтерфейсу (рисунок 3.4).

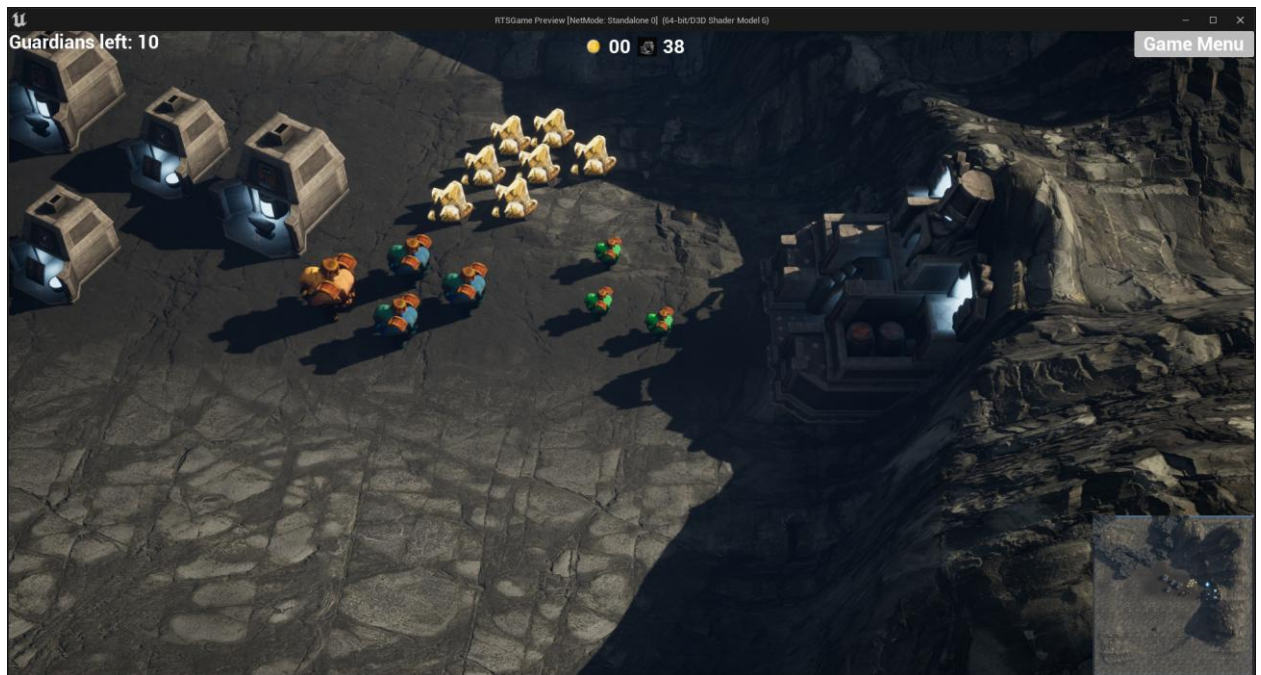


Рисунок 3.4 – Екран ігрового інтерфейсу

На ігровому інтерфейсі у лівому верхньому куті відображаються цілі гри, у верхній частині посередині – кількість наявних ресурсів, у верхньому правому куті – кнопка переходу в меню паузи та у нижньому правому куті – мінікарта.

Процес виділення об'єктів утримуванням лівої кнопки миші відображено на рисунку 3.5.



Рисунок 3.5 – Виділення ігрових об'єктів

Виділені об'єкти позначаються обведенням у формі кола. Після виділення ігрових об'єктів з'являється список у нижньому лівому куті ігрового інтерфейсу. Виділеним юнітам можливо видавати накази правою кнопкою миші.

Для видачі наказу на переміщення потрібно натиснути правою кнопкою миші на місці на карті, що відображено на рисунку 3.6.



Рисунок 3.6 – Наказ на переміщення

Одразу після натиснення правої кнопки миші на короткий проміжок часу з'являється відмітка, куди вирушать юніти.

Наказ на атаку, здійснений натисканням правою кнопкою миші по ворогу, відображено на рисунку 3.7.



Рисунок 3.7 – Наказ на атаку

Юніти підходять до ворога, здійснюють атаку та у ворога зменшуються одиниці здоров'я, що відображаються шкалою над ним.

Для здійснення наказу на ремонт будівлі потрібно виділити робітника та натиснути правою кнопкою миші по пошкодженій будівлі (рисунок 3.8).

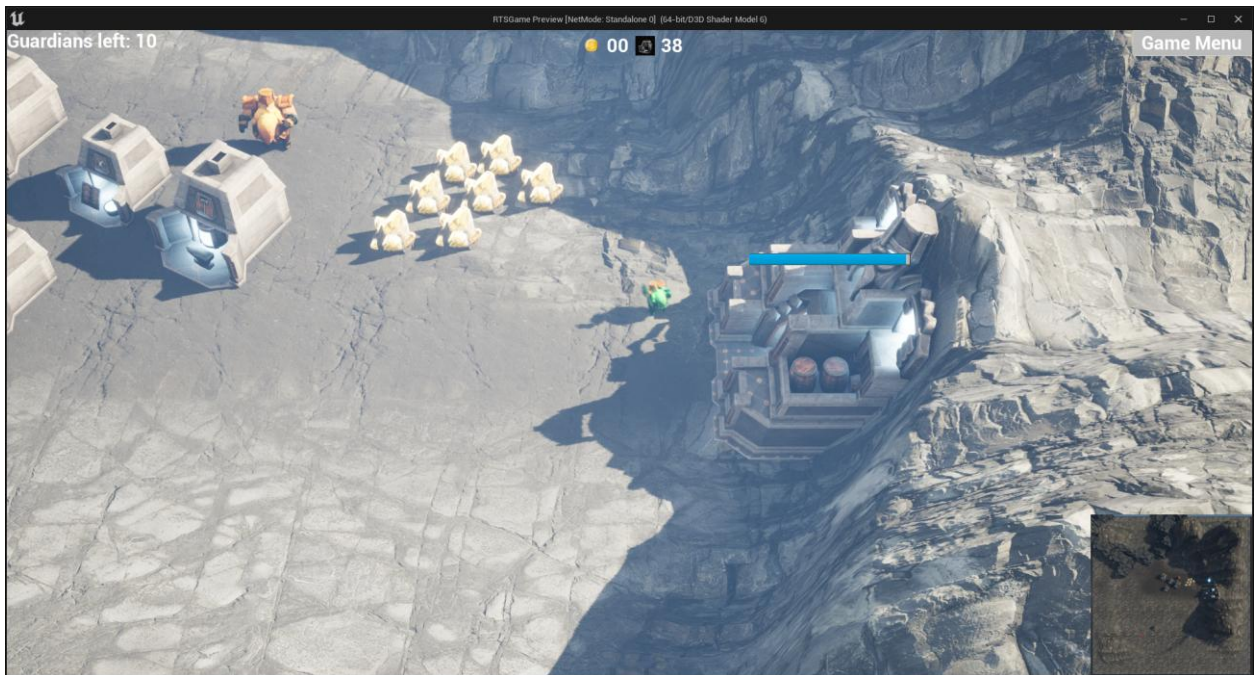


Рисунок 3.8 – Наказ на ремонт будівлі

Робітник підходить до будівлі і здійснює ремонт, при цьому кількість одиниць здоров'я будівлі зростає.

Для здійснення наказу на добування ресурсу потрібно виділити робітника та натиснути правою кнопкою миші по ресурсу (рисунок 3.9).

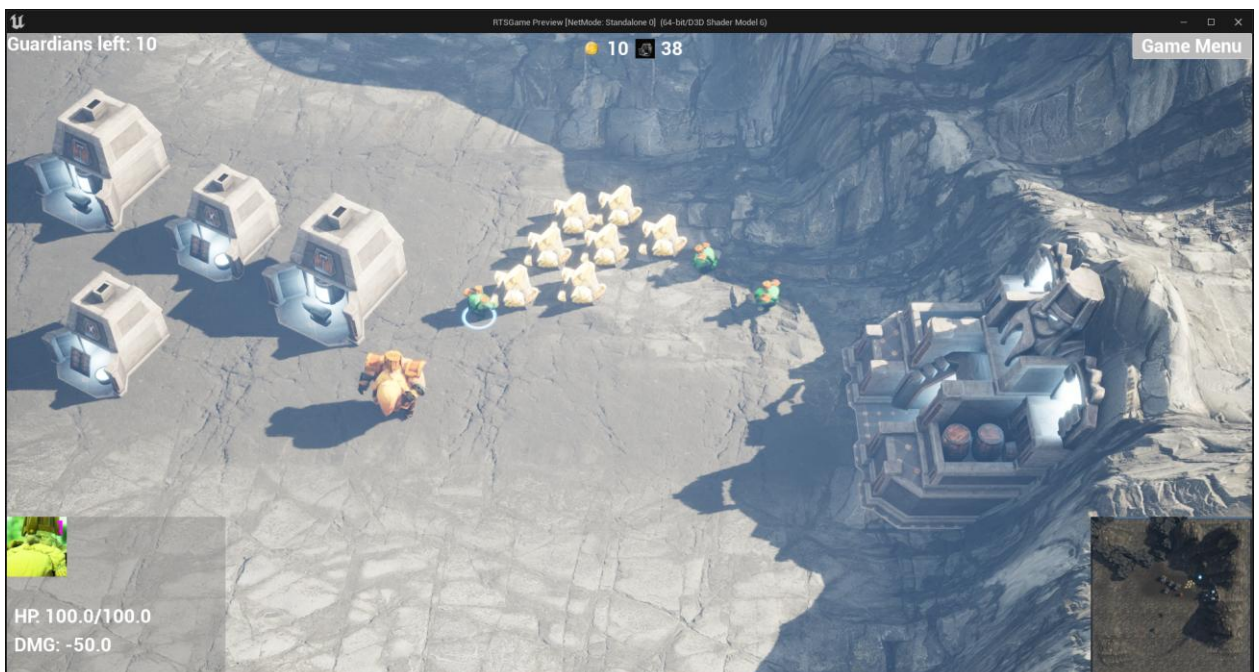


Рисунок 3.9 – Наказ на добування ресурсів

Юніти циклічно підходять до ресурсу, добувають його та відносять у ратушу, при цьому збільшується кількість золота у верхній частині ігрового інтерфейсу.

Для створення юніта спочатку потрібно виділити будівлю, що може створювати юнітів (рисунок 3.10). При наведенні на кнопку створення юніта відображається необхідна кількість ресурсів.

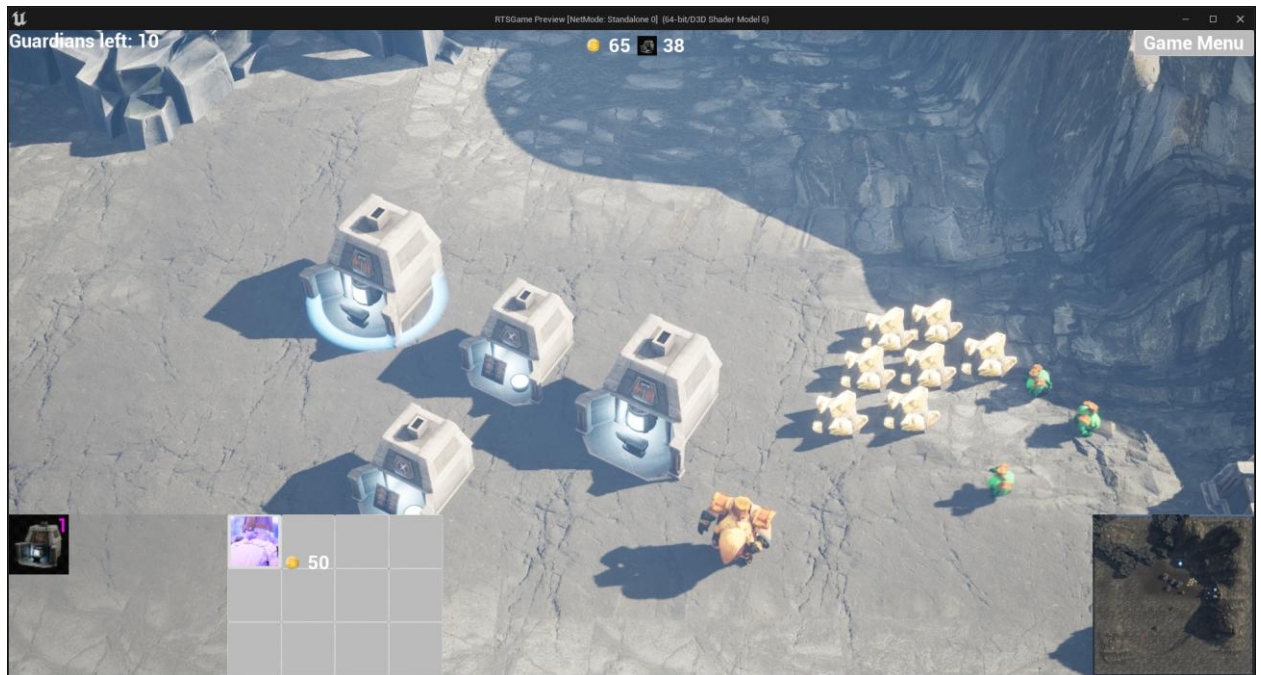


Рисунок 3.10 – Виділення казарми для створення юніта

Після натискання на кнопку створення юніта списуються ресурси та через деякий час з'являється новий юніт (рисунок 3.11).

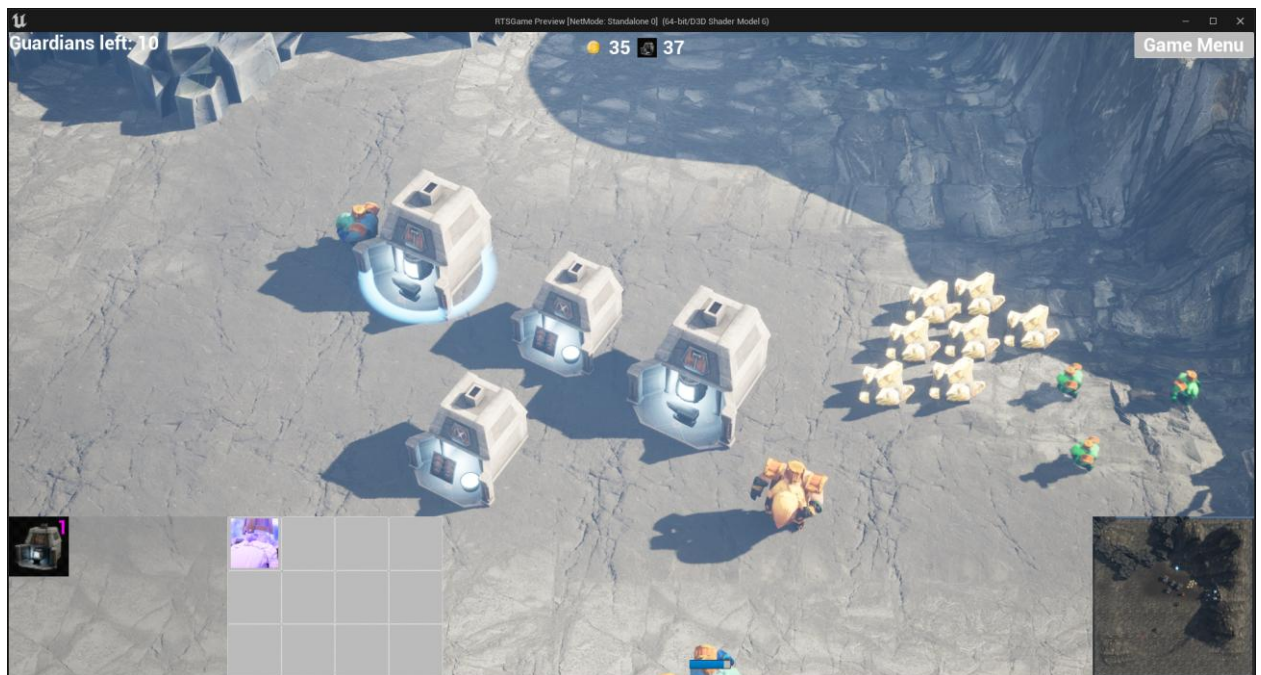


Рисунок 3.11 – Завершення створення нового юніта

Меню паузи відкривається натисканням кнопки «Game Menu» у правому верхньому куті ігрового інтерфейсу (рисунок 3.12).

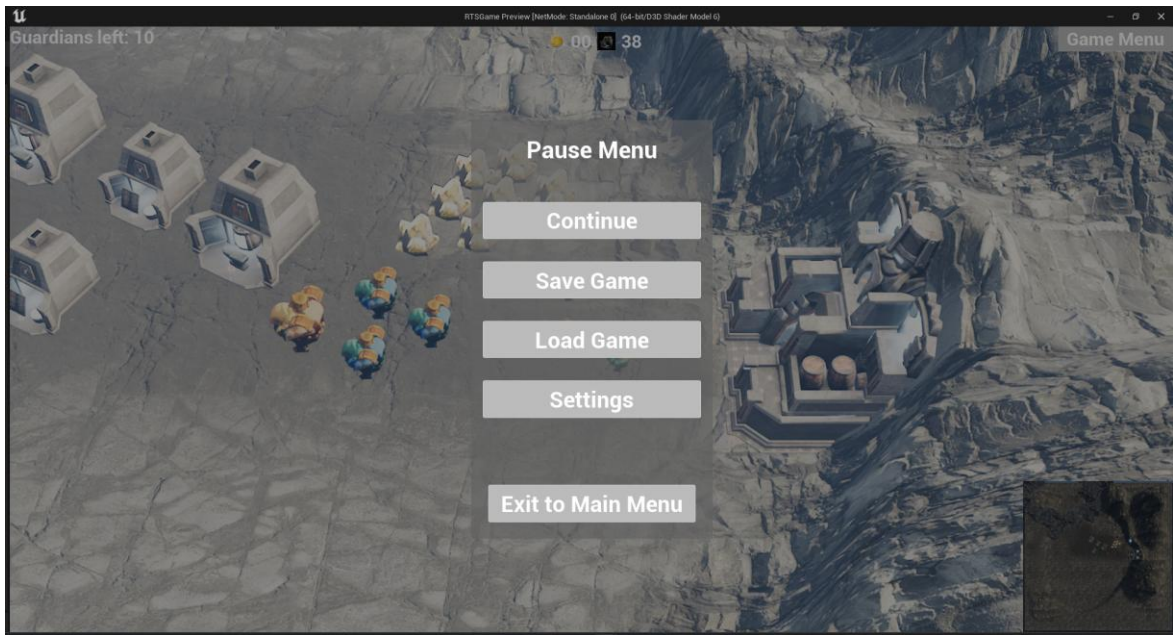


Рисунок 3.12 – Меню паузи

При відкритті меню паузи ігровий процес призупиняється. Для відновлення ігрового процесу потрібно натиснути на кнопку «Continue». Для переходу в меню збереження потрібно натиснути кнопку «Save Game». Для переходу в меню завантаження прогресу гри (рисунок 3.3) потрібно натиснути кнопку «Load Game». Для відкриття меню налаштувань потрібно натиснути кнопку «Settings». Для виходу в головне меню потрібно натиснути кнопку «Exit to Main Menu».

Меню збереження гри відображено на рисунку 3.13.



Рисунок 3.13 – Збереження гри

Гравець вводить назву збереження у поле відповідно до допустимих назв файлів системи Windows і натискає на кнопку «Save». Нове збереження відображається у списку збережень, зокрема у меню завантаження (рисунок 3.14).

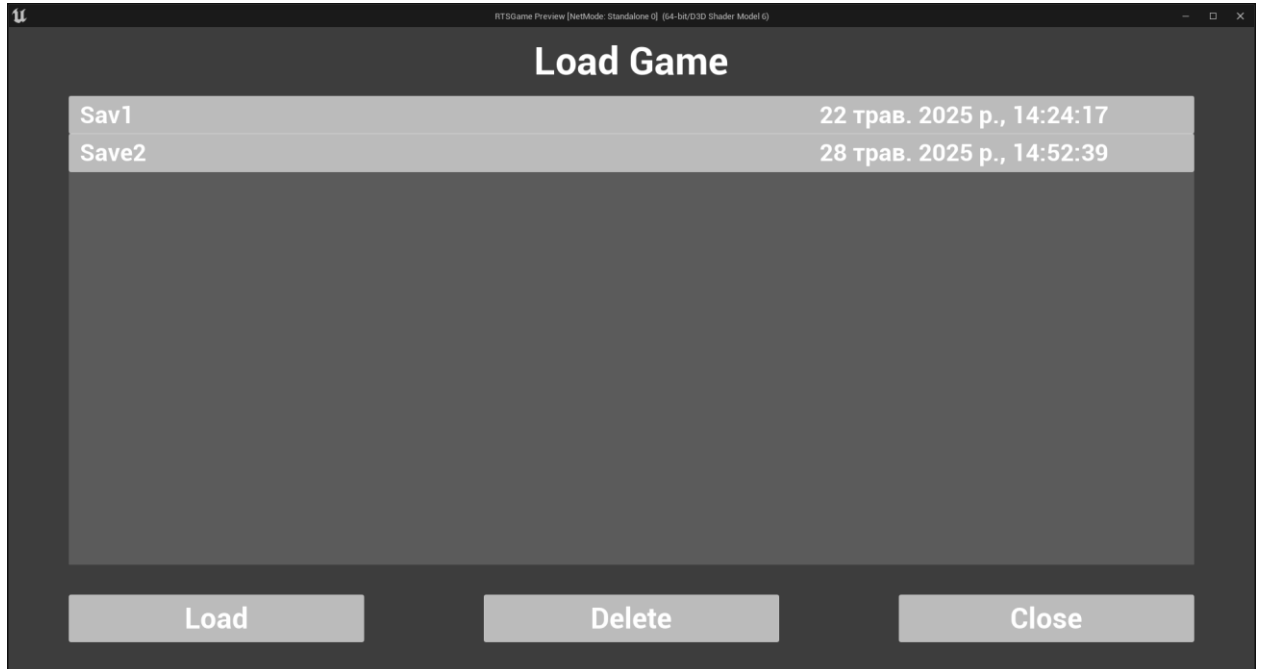


Рисунок 3.14 – Оновлення меню завантаження

Якщо гравець знищує юніти ворога, що задані місією, ціль місії оновлюється у верхньому лівому куті, що відображено на рисунку 3.15.

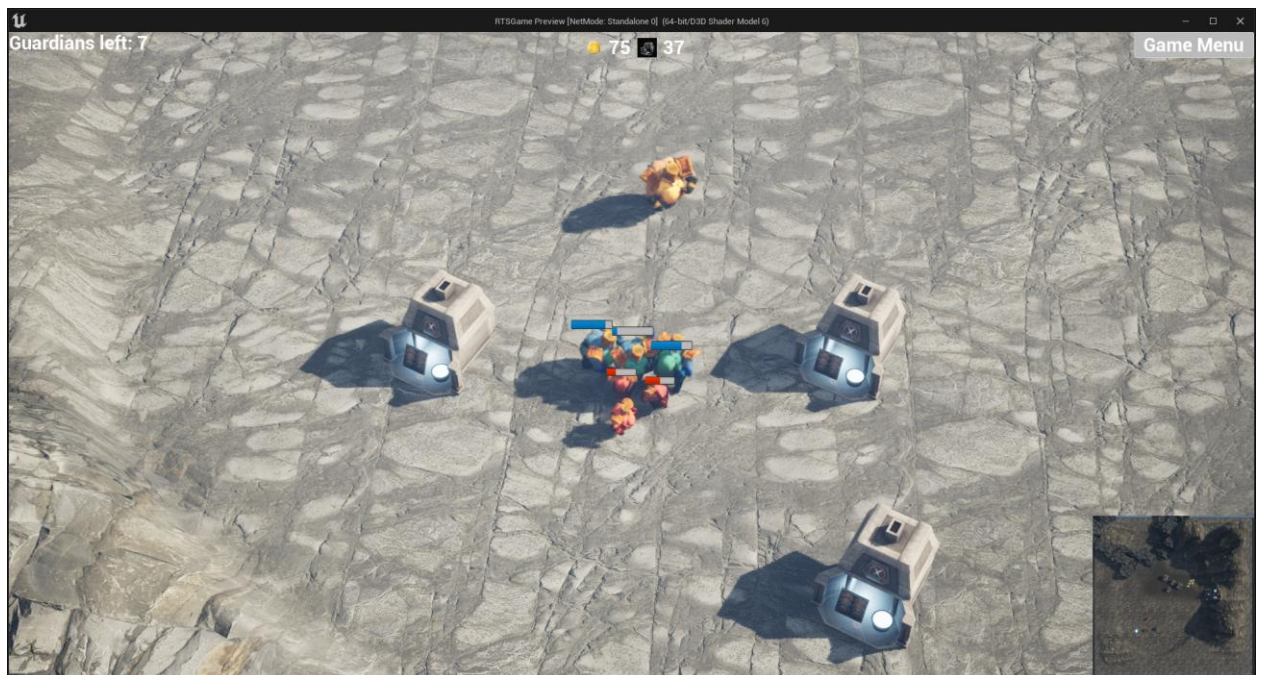


Рисунок 3.15 – Виконання місії

При виконанні місії, у даному випадку – після знищення 10 юнітів ворога, відображається екран перемоги (рисунк 3.16).

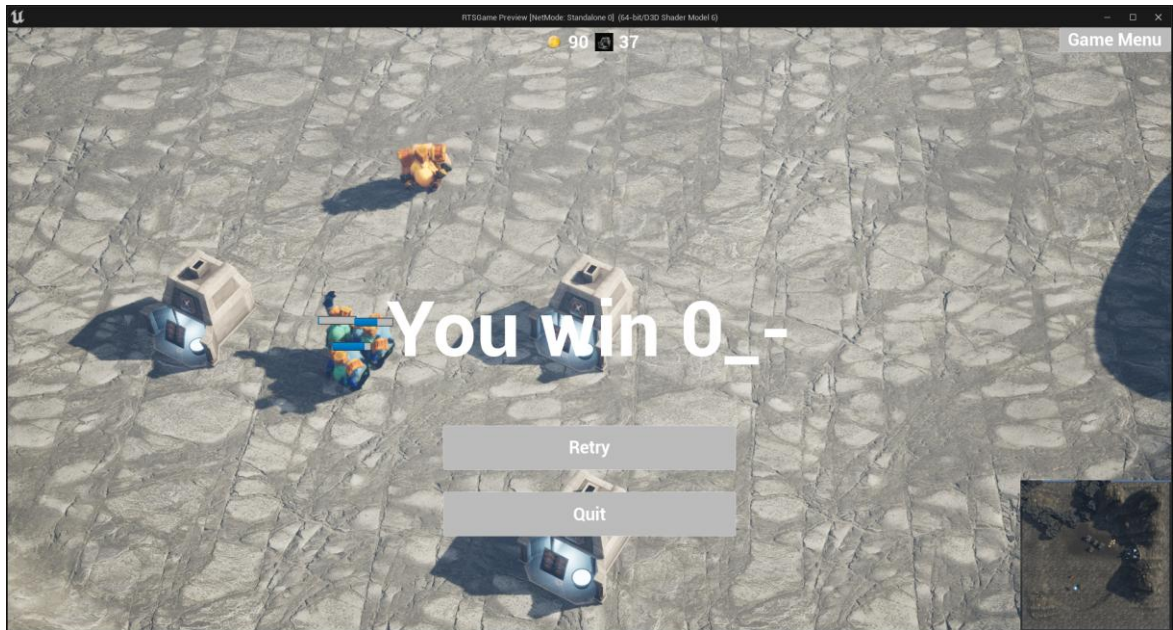


Рисунок 3.16 – Екран перемоги

Кнопкою «Retry» можна перезапустити місію та кнопкою «Quit» повернутися до головного меню гри.

При втраті робітників та ратуші відображається екран поразки (рисунк 3.17).

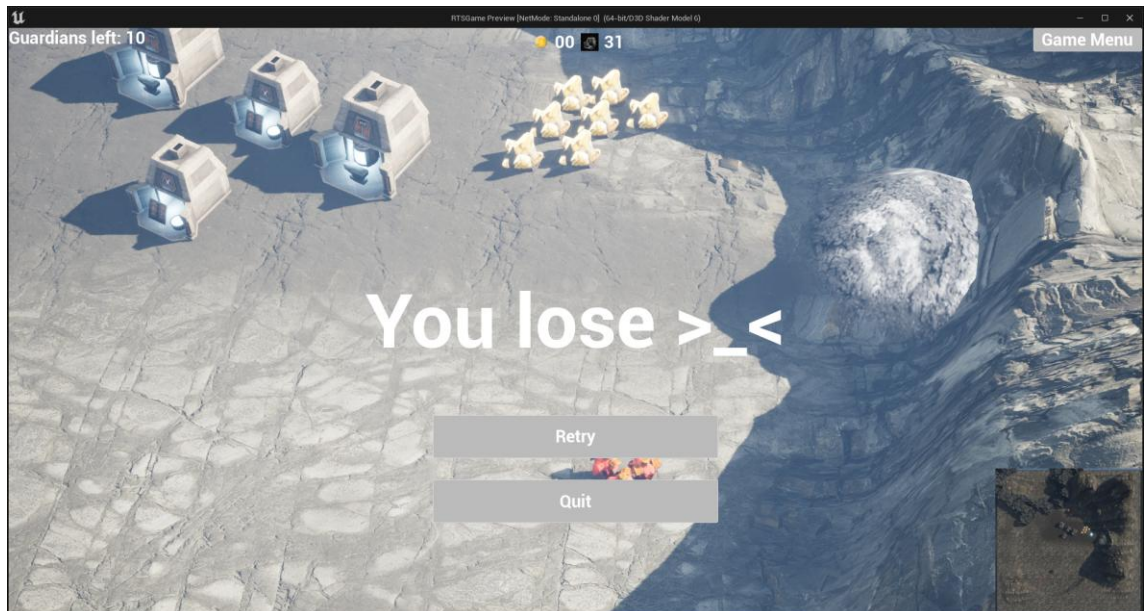


Рисунок 3.17 – Екран поразки

Після повернення до головного меню (рисунк 3.1) можливо знову почати нову гру, завантажити збереження чи вийти з гри.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ГРА У ЖАНРІ СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ З ПОДІЙНО-
ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ**

Графічний матеріал

КПІ.ПІ-1530.045490.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Владислав САРНАЦЬКИЙ

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

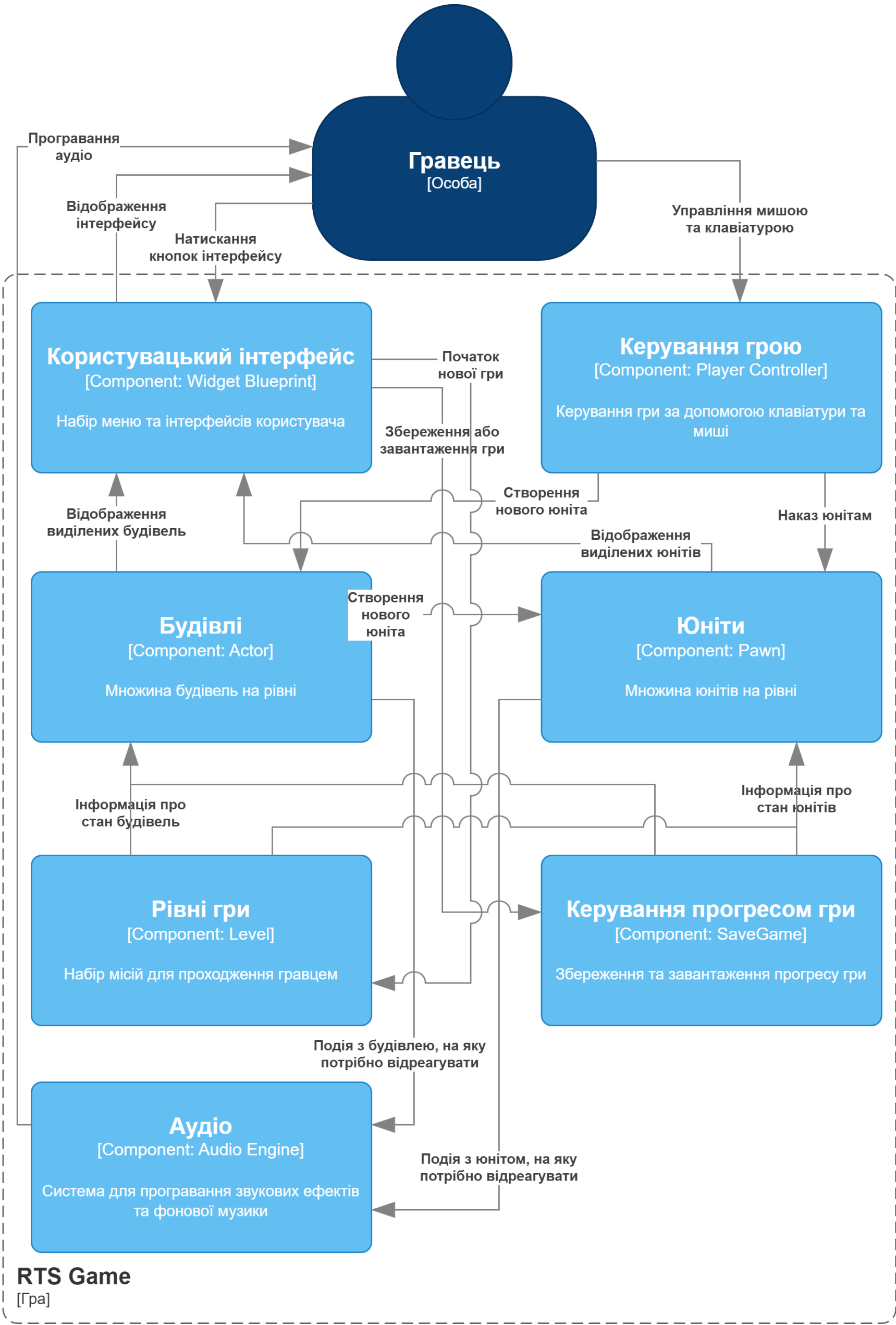
Виконавець:

_____ Денис ШЛЯХТУН

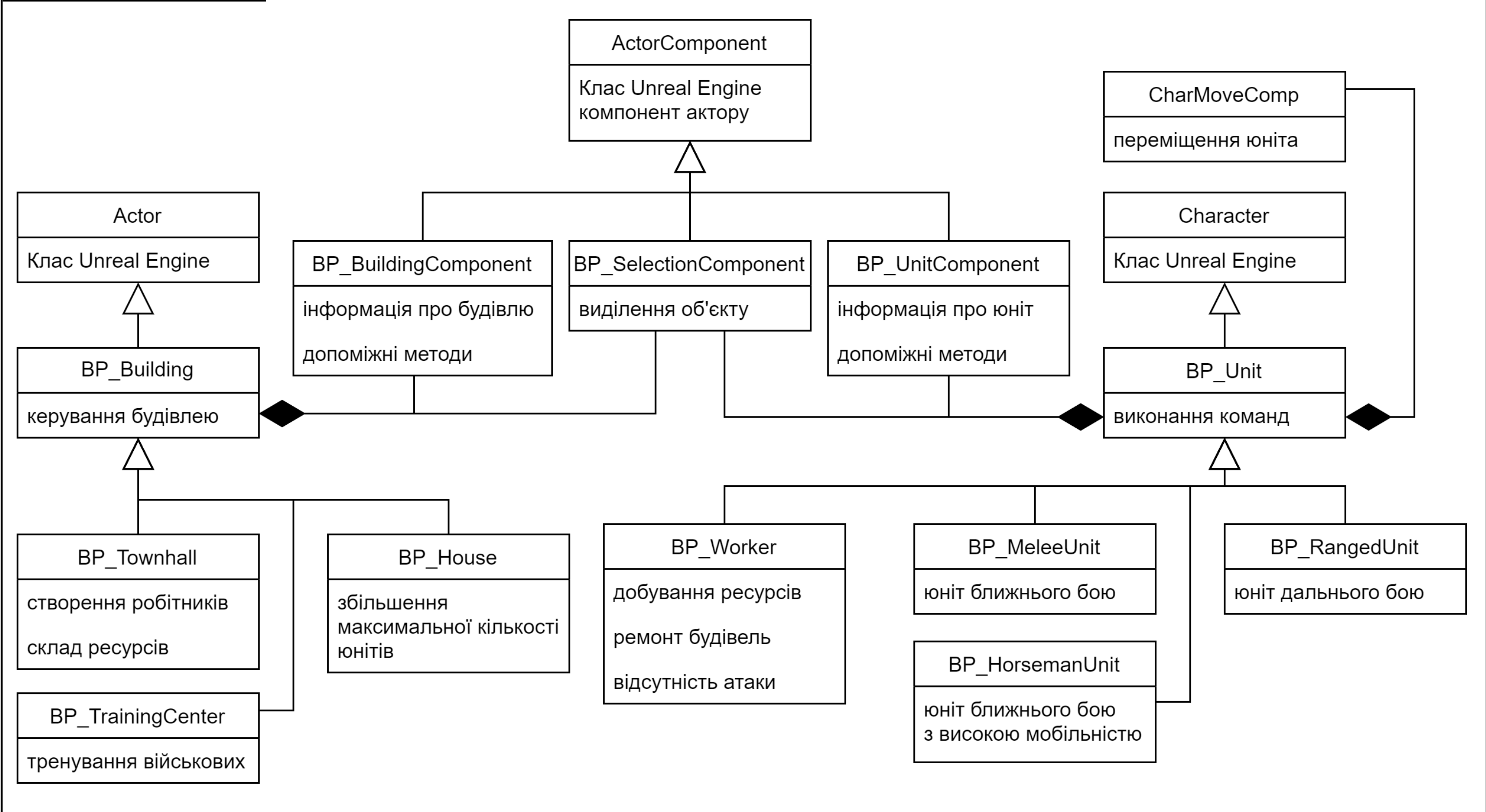
Київ – 2025



					КПІ.ІП-1530.045490.06.99.CCB					
					Схема структурна варіантів використання					
Зм.	Арк.	№ документа	Підпис	Дата						
Розробив		Шляхтун Д.М.			Гра у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою					
Перевірив		Сарнацький В.В.								
Т. контр.										
					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15					
Н. контр.		Шулькевич Т.В.								
Затвердив		Жаріков Е.В.								



					КПІ.ІП-1530.045490.06.99.CCM				
					Схема структурна компонентів програмного забезпечення		Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата					
Розробив		Шляхтун Д.М.							
Перевірив		Сарнацький В.В.							
Т. контр.							Аркуш 1		Аркушів 1
Н. контр.		Шулькевич Т.В.			Гра у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою		КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15		
Затвердив		Жаріков Е.В.							



					КПІ.ІП-1530.045490.06.99.ССК								
					Схема структурна класів програмного забезпечення				Літера		Маса	Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата					Аркуш 1		Аркушів 1		
Розробив		Шляхтун Д.М.											
Перевірів		Сарнацький В.В.											
Т. контр.													
Н. контр.		Шулькевич Т.В.			Гра у жанрі стратегії в реальному часі з подійно-орієнтованою архітектурою				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15				
Затвердив		Жаріков Е.В.											