МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук, доцент
_____ Давиденко Є. О
підпис

«___» _____20____ р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІГРОВИЙ ЗАСТОСУНОК В ЖАНРІ ПРИГОДНИЦЬКОГО RPG НА ОСНОВІ РУШІЯ UNITY

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.01 – 409. 22011018

Здобувач _____Б. М. Папіралін
підпис
«__» _____ 20__ р.

Керівник Ph.D., ст. викл. кафедри ІПЗ _____К. О. Антіпова
підпис
«__» _____ 20__ р.

Консультант канд. техн. наук., доцент _____А. О. Алексєєва
підпис
«__» _____ 20__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри, канд. техн. наук, доцент

_____ Давиденко Є. О

«_____» _____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувачу групи 409 факультету комп'ютерних наук

_____Папіралін Богдан Михайлович_____

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Ігровий застосунок в жанрі пригодницького RPG на основі рушія Unity_____

_____

_____

Затверджена наказом по ЧНУ від «22» ____грудня___ 2023__ р. № __269_____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

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

Ігровий застосунок на рушії Unity, з інтерактивними квестами та цікавим сюжетом. Вхідні дані для розробки включають документацію до інструментів розробки Unity _____

_____

_____

_____

_____

_____

4. Перелік питань, що підлягають розробці :

- аналіз сучасних тенденцій та існуючих рішень у сфері RPG пригодницьких ігор;
- вибір технологій для створення ігрового середовища на Unity;
- розробка детальної концепції ігрового світу, сюжету, системи персонажів, ігрових механік та взаємодій;
- проєктування архітектури програмного забезпечення;
- реалізація ігрових механік;
- розробка інтерфейсу користувача

5. Перелік графічних матеріалів

Презентація_____

_____

_____

6. Завдання до спеціальної частини

Дослідження питань пожежної безпеки та інтегральна оцінка важкості праці на робочому місці розробника програмного забезпечення._____

_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
канд. техн. наук, доцент Анна Олександрівна Алексєєва	екології	спеціальна частина з охорони праці

Керівник роботи _____PhD, ст. викл. Антіпова Катерина Олександрівна_____

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____ Папіралін Богдан Михайлович_____

(прізвище, ім'я, по батькові здобувача)

_____

(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: _Ігровий застосунок в жанрі пригодницького RPG на основі рушія Unity

№	Найменування	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	05.12.2023	08.12.2023	Виконано
2	Огляд літератури за темою роботи	01.03.2024	08.03.2024	Виконано
3	Складання календарного плану КРБ	11.03.2024	18.03.2024	Виконано
4	Аналіз предметної області	19.03.2024	26.03.2024	Виконано
5	Розробка проєктних рішень	27.03.2024	11.04.2024	Виконано
6	Моделювання та конструювання ПЗ	12.04.2024	26.04.2024	Виконано
7	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	29.04.2024	31.05.2024	Виконано
8	Оформлення КРБ та презентації	31.05.2024	03.06.2024	Виконано
9	Відгук керівника КРБ	03.06.2024	07.06.2024	Виконано
10	Розробка спеціальної частини з охорони праці	03.06.2024	07.06.2024	Виконано
11	Попередній захист	03.06.2024	07.06.2024	Виконано
12	Рецензування	10.06.2024	17.06.2024	Виконано
13	Завершення оформлення КРБ та презентації	17.06.2024	24.06.2024	Виконано
14	Захист кваліфікаційної роботи	25.06.2024	30.06.2024	

Розробив здобувач _____Папіралін Богдан Михайлович_____ _____

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

«__» _____ 2024 р.

Керівник роботи PhD, ст. викладач Антіпова Катерина Олександрівна_

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

«__» _____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок в жанрі пригодницького RPG на основі рушія Unity»

Здобувач 409 гр.: Папіралін Богдан Михайлович

Керівник: Ph.D., ст. викл. Антіпова К. О.

Актуальністю кваліфікаційної роботи є те, що розробка ігрових застосунків залишається однією з найбільш динамічних та інноваційних сфер у галузі програмної інженерії. Ігри в жанрі пригодницького RPG користуються високою популярністю серед широкого кола гравців завдяки їх здатності занурювати у вигадані світи, розвивати креативне мислення та навички вирішення проблем. Використання рушія Unity дозволяє реалізувати складні візуальні ефекти, динамічну графіку та багатоплатформеність, роблячи розробку доступною та ефективною.

Об'єктом кваліфікаційної роботи є процес розробки 3D-ігор в жанрі пригодницьких RPG.

Предметом кваліфікаційної роботи є технології розробки ігрових механік і взаємодії в 3D-грі на рушії Unity.

Метою кваліфікаційної роботи є розробка 3D-гри в жанрі пригодницької RPG для популяризації ігрових застосунків.

Кваліфікаційна робота складається з вступу, чотрьох розділів, висновків та переліку джерел посилань.

КРБ викладена на вісімдесят одній сторінці, вона містить чотири розділи, двадцять дев'ять ілюстрацій, три таблиці, п'ятнадцять джерел в переліку посилань.

Ключові слова: Unity, RPG, ігровий застосунок, пригодницькі ігри, програмне забезпечення.

ASTRACT

of the Bachelor's Thesis

"Game application in the genre of adventure RPG based on the Unity engine"

Student of group 409: Papiralin Bohdan Mykhailovych

Supervisor: Ph.D., Senior lecturer Antipova K. O.

The relevance of this qualification work lies in the fact that the development of gaming applications remains one of the most dynamic and innovative fields in software engineering. Adventure RPG games enjoy high popularity among a wide range of players due to their ability to immerse players in fictional worlds, develop creative thinking, and problem-solving skills. The use of the Unity engine allows for the implementation of complex visual effects, dynamic graphics, and cross-platform capabilities, making development accessible and efficient.

The object of the qualification work is the process of developing 3D games in the adventure RPG genre.

The subject of the qualification work is the technologies for developing game mechanics and interactions in a 3D game using the Unity engine.

The aim of the qualification work is to develop a 3D game in the adventure RPG genre to popularize gaming applications.

The qualification work consists of an introduction, four chapters, conclusions, and a list of references.

The qualification work is presented on eighty-one pages; it contains four chapters, twenty-nine illustrations, three tables, and fifteen sources in the list of references.

Key words: Unity, RPG, game application, adventure games, software.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КРБ – кваліфікаційна робота бакалавра

ПК – персональний комп'ютер

ПЗ – програмне забезпечення

ШІ – штучний інтелект

3D – 3-dimensional

HUD – (heads-up display) or status bar

JSON – JavaScript Object Notation

NPC – non-player character

RPG – Role-Playing Game

UI – User interface

URP – Universal Render Pipeline

ВСТУП

Актуальність. Розробка ігрових застосунків залишається однією з найбільш динамічних і інноваційних сфер у галузі програмної інженерії. Ігри в жанрі пригодницького RPG користуються високою популярністю серед широкого кола гравців завдяки їх здатності занурювати у вигадані світи, розвивати креативне мислення та навички вирішення проблем. Використання рушія Unity дозволяє реалізувати складні візуальні ефекти, динамічну графіку та багатоплатформеність, роблячи розробку доступною та ефективною.

Об'єктом кваліфікаційної роботи є процес розробки 3D-ігор в жанрі пригодницьких RPG.

Предметом кваліфікаційної роботи є технології розробки ігрових механік і взаємодії в 3D-грі на рушії Unity.

Метою кваліфікаційної роботи є розробка 3D-гри в жанрі пригодницької RPG для популяризації ігрових застосунків.

Для досягнення визначеної мети необхідно вирішити такі завдання:

- аналіз сучасних тенденцій та існуючих рішень у сфері RPG пригодницьких ігор;
- вибір технологій для створення ігрового середовища на Unity;
- розробка детальної концепції ігрового світу, сюжету, системи персонажів, ігрових механік та взаємодій;
- проєктування архітектури програмного забезпечення;
- реалізація ігрових механік;
- розробка інтерфейсу користувача

Аналіз ринку вказує на великий інтерес до пригодницьких RPG на Unity, що підкреслюється успіхом ігор "Pillars of Eternity" і "Chained Echoes", демонструючи гнучкість Unity у створенні унікальних ігрових досвідів та можливості для інновацій.

1 ВИЗНАЧЕННЯ ВИМОГ ДО ІГРОВОГО ЗАСТОСУНКУ

1.1 Аналіз ігрової індустрії

Останні роки ігрова індустрія зазнала значних змін, особливо в часи пандемії, яка в певному сенсі стала благословенням для цієї галузі. Продукти цієї індустрії забезпечили можливість взаємодії без фізичного контакту, що виявилося захищеним від економічної рецесії. Завдяки цьому було досягнуто високої залученості аудиторії, а також вдоволено потребу в соціальних зв'язках.

Проте, після цього періоду зростання інвестицій у галузь ми спостерігаємо певні негативні наслідки. Сьогодні галузь переживає втрату контролю і відсутність стратегічного бачення через надмірне захоплення попередніми успіхами.

На тлі цих подій 2023 рік виділяється як особливо успішний для ігрової індустрії, який став своєрідним відродженням на всіх ігрових платформах. Велика кількість високо оцінених ігор, таких як Alan Wake II, Monopoly GO!, Baldur's Gate 3, і Hogwarts Legacy, свідчить про значний прогрес і різноманіття на ринку ігор.

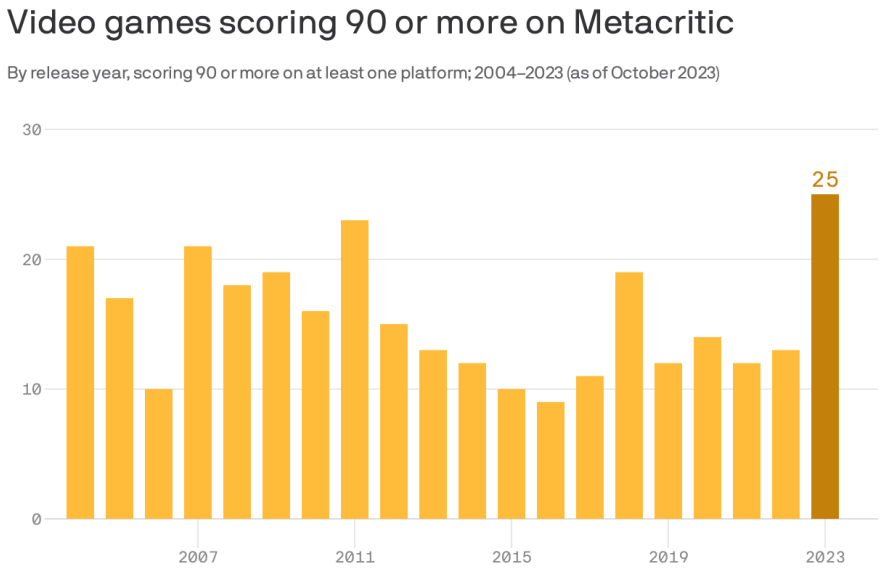

Рисунок 1.1 – Візуалізація кількості трендових ігор [1]

Незважаючи на те, що створення масштабних ігор вимагає значних часових вкладень – зазвичай не менше кількох років – минулий рік був насичений випусками великих хітів від провідних компаній. Цей факт викликає певні сумніви щодо того, чи зможе 2024 рік навіть частково відповідати рівню попереднього.

Проте, приклад «Palworld» демонструє, що індустрія відеоігор здатна на несподіванки. Менш відомі компанії мають потенціал раптово випустити гру, яка може конкурувати за продажами та кількістю одночасних користувачів з продукцією гігантів індустрії. Це підтверджує, що в індустрії завжди існує потенціал для інновацій, навіть якщо великі компанії не можуть щороку випускати нові хіти.

Очікується, що 2024 рік надасть можливості середнім та малим розробникам скористатися загальним інтересом до галузі та продемонструвати, що успіх у сфері ігор не обмежується лише великими бюджетами та розробкою великих проєктів.

Штучний інтелект (ШІ) продовжує залишатися в центрі уваги в технологічній галузі, стаючи ключовим інструментом для оптимізації та автоматизації процесів у виробництві. Однак, попри значний вплив на місцеве виробництво, ШІ ще не спричинив глобальних змін у базових правилах бізнесу, окрім як в оцінюванні стартапів, де воно виступає як значний коефіцієнт. Технічні обмеження, такі як нездатність точно створювати 3D-контент, ставлять під сумнів повсюдне впровадження ШІ у складні індустрії, зокрема у розробку ігор.

Зростання інтересу до технологій Web3 у ігровій індустрії свідчить про потенціал використання блокчейну для створення нових форм ігрових активів та економік, де гравці мають реальне право власності на ігрові предмети. Проте, реальний вплив Web3 на ігрові платформи залишається обмеженим, оскільки основна мотивація гравців — це розвага, а не заробіток. Опитування серед

розробників показує, що тільки мала частина з них активно використовує блокчейн, що вказує на потенційні складнощі у прийнятті цієї технології на масовому рівні.

Криптовалюти продовжують знаходити застосування в сфері цифрових платежів та верифікації прав власності, пропонуючи нові можливості для цифрової економіки. Однак, юридичні та регуляторні виклики залишаються вагомою перешкодою для глобального прийняття криптовалют, а потреба в подальшій розробці та стандартизації технології ускладнює її інтеграцію у традиційні фінансові системи.

Наразі, ШІ, Web3, та криптовалюти продовжують розвиватися, і передбачається, що в наступні роки вони зможуть змінити ландшафт багатьох індустрій, включаючи ігрову. Однак, значні зміни, які цілком перетворять ці сектори, не очікуються до 2025-2027 років, коли технології стануть достатньо зрілими для масового впровадження та вирішення існуючих технічних та регуляторних проблем.

У 2023 році ігровий ринок приніс дохід у $184 млрд, що на 0,6% більше, ніж у попередньому році. Невелике зниження відбулося у мобільному сегменті (-1,6%). Незважаючи на це, мобільні ігри залишаються явним лідером серед усіх платформ: їх частка в загальній виручці ігрового ринку складає 49%.

Друге місце займає консольний сегмент (29%), який, навпаки, продемонстрував зростання порівняно з минулим роком (+1,9%). Ця тенденція пов'язана з тим, що в 2023 році після затримок, пов'язаних з пандемією, було випущено велику кількість довгоочікуваних ігор: Hogwarts Legacy, The Legend of Zelda, Tears of the Kingdom, Final Fantasy XVI, Spider-Man 2, Forza Motorsport, Starfield і багато іншого. Сегмент комп'ютерних ігор продемонстрував найбільше зростання доходів (+5,2%), що, як і в разі консолей, сталося завдяки багатьом хітовим релізам.

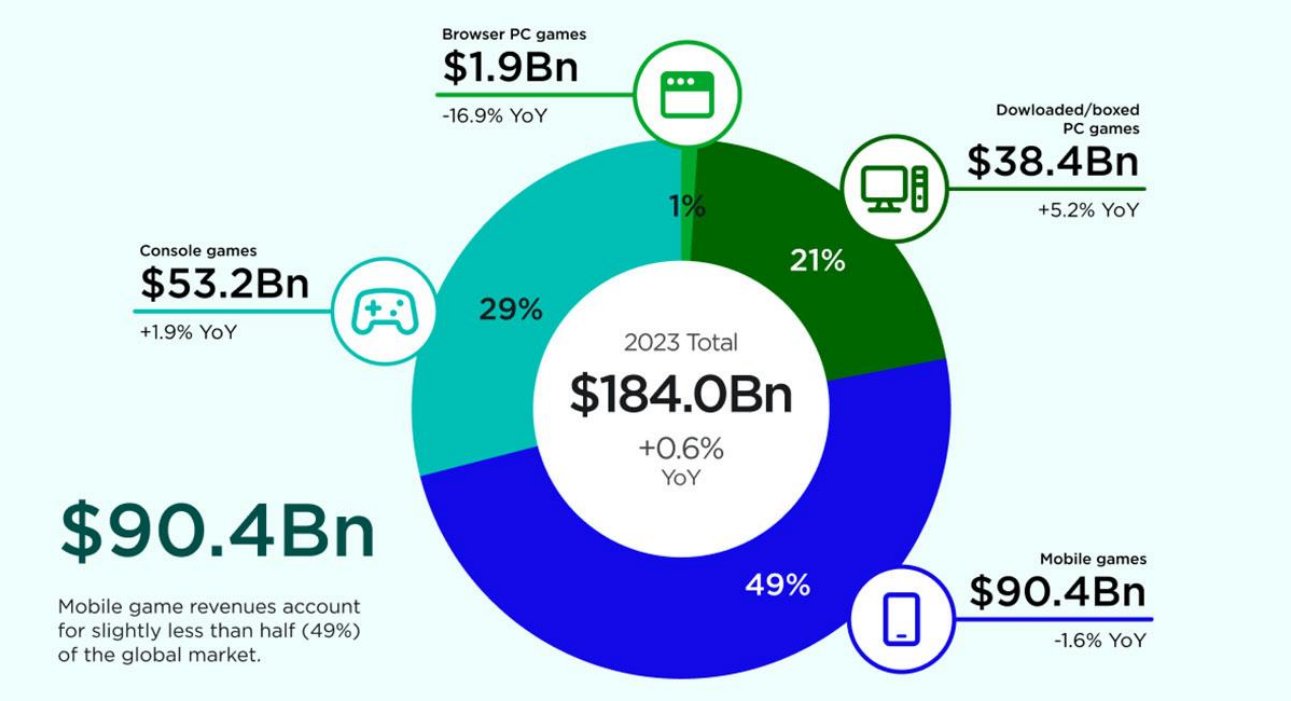

Рисунок 1.2 – Розподіл прибутку ігор по сегментаам[14]

Що стосується розподілу доходів за регіонами, то лідирує Азіатсько-Тихоокеанський регіон з його часткою 46%. Проте, у цьому регіоні відбулося зниження доходів на 0,8% порівняно з минулим роком, тоді як інші регіони залишилися в плюсі. Таке зниження пов'язане з особливостями державного регулювання в Китаї, які ускладнюють вихід ігор на ринок країни. Проте в минулому році тимчасова заборона на видачу ліцензій була знята, що знову відкрило компаніям шлях на китайський ігровий ринок.

Північна Америка продовжує утримувати свої сильні позиції, на її частку припадає 27% виручки завдяки популярності там консолей. За тенденцією появи нових гравців, найбільше зростання виручки спостерігалося на Близькому Сході та в Африці (4,7%) та в Латинській Америці (3,8%). На це вплинув ряд факторів, зокрема покращення інфраструктури мобільного інтернету, здешевлення доступу до інтернету, доступність ігор і зростання числа користувачів смартфонів.

Серед окремих країн варто згадати Саудівську Аравію, яка, прагнучи диверсифікувати свої джерела доходів, останнім часом інвестує в ігрову індустрію. Успіх країни особливо помітний на прикладі створеної у 2021 році компанії Savvy Games Group: у 2023 році компанія придбала Scopely, американського розробника і видавця мобільних ігор. Savvy Games Group також володіє кіберспортивною організацією ESL та платформою Facelt.

У Японії популярність комп'ютерних ігор продовжує швидко зростати, незважаючи на загальну рецесію після пандемії. Наприклад, у 2022 році зростання склало вражаючі 43% (з 131,3 млрд єн до 189,2 млрд єн).

Індустрія в цілому повертається до стабільності після постпандемічних коливань (реструктуризація ігрових компаній, звільнення співробітників, масово найнятих під час пандемічного «буму» тощо). За оцінками аналітиків, до 2026 року річний дохід ігрового ринку складе $205,7 млрд.

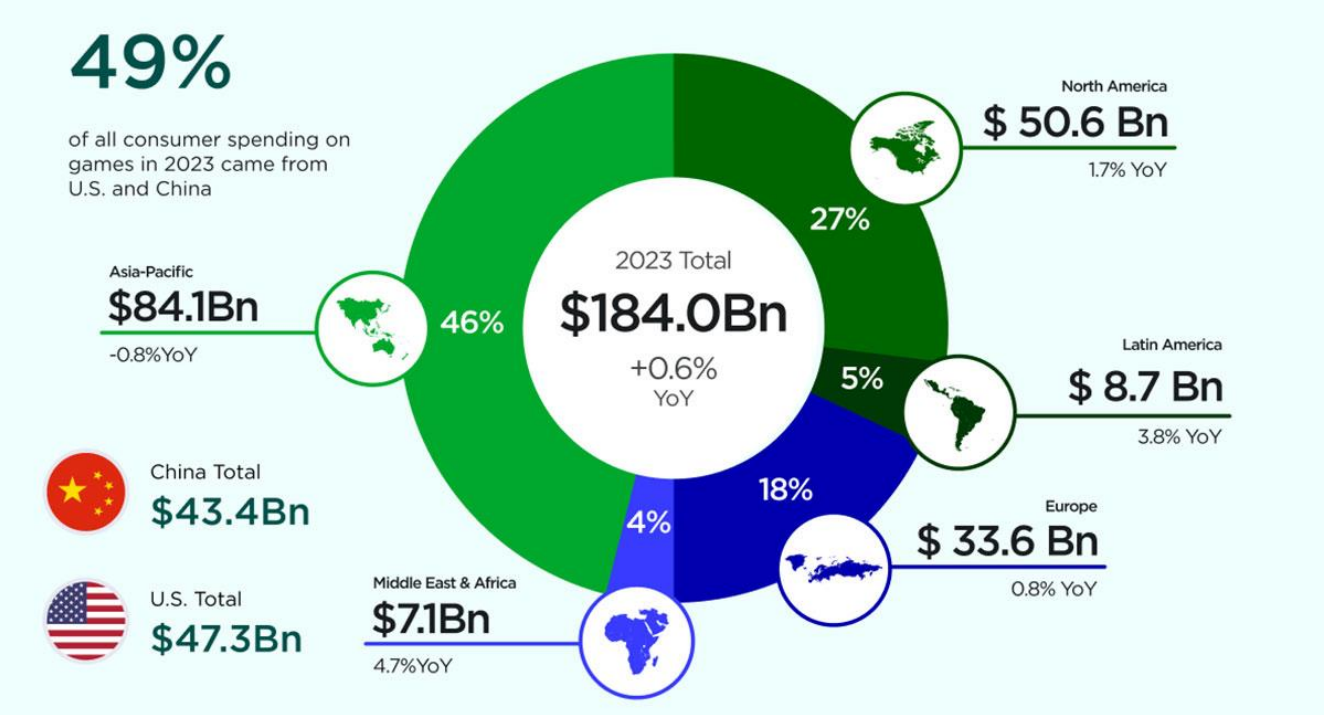

Рисунок 1.3 – Розподіл прибутку ігор по країнам [14]

Популярність конкретних жанрів варіюється від платформи до платформи. Однак серед загальних для всіх тенденцій відзначимо, що жанр «королівської

битви» відступає. Хіти, випущені для конкретної платформи, можуть викликати коливання статистики, але загальна картина виглядає так:

Найпопулярніший жанр на консолях — пригодницькі ігри, на які у 2023 році, як і у 2022 році, припало 17,1% доходів сегменту ринку. Трансмедійні ігри (такі як Hogwarts Legacy і Star Wars Jedi: Survivor) і ремейки відомих ігор минулих років (Dead Space, Resident Evil 4) виявилися особливо прибутковими.

На ПК лідирують шутери з 14,1% від загального доходу у 2023 році. Аналітики прогнозують зростання доходів від цього жанру на +4,9%. Такі хіти, як Valorant, Counter-Strike: Global Offensive, Call of Duty і Escape from Tarkov, зберегли свої позиції у 2023 році. Популярності жанру також сприяли однокористувацькі проєкти, такі як Payday 3 і Warhammer 40,000: Boltgun.

На мобільних платформах жанр RPG залишається найпопулярнішим. Незважаючи на невелике зниження доходів через зміни в політиці конфіденційності власників платформ, на частку рольових ігор припадає 23,1% загального доходу від мобільних ігор (більше 10% на всьому ігровому ринку).

Майбутнє ігрової індустрії виглядає світлим. Ігри пройшли довгий шлях: від нішового розваги до перспективного ринку з мільярдними доходами, демонструючи стабільний ріст рік за роком. Не дивно, що все більше і більше компаній починають цікавитися цією сферою як потенційною сферою для бізнесу, а ті, хто вже присутній на ринку, освоюють нові сегменти або регіони. [14]

1.2 Огляд та аналіз застосунків аналогів

Приклади аналогічних систем до ігрового застосунку в жанрі RPG :

Chained Echoes

Назва: Chained Echoes

Виробник: Matthias Linda

Архітектура: розроблена з використанням рушія Unity, як зазначено у відгуках.

Мова реалізації: C#.

Основні функції, характеристики:

Класичний 2D RPG з елементами фентезі, мечів, магії та мехів.

Швидкісні, пошагові битви без випадкових зустрічей, де бої відбуваються безпосередньо там, де ви зустріли ворога.

Вибір шляху гравця впливає на долю земель Valandis та її мешканців.

Різноманітність досліджень: пішки, на мехах або на повітряному кораблі

Багата система обладнання та навичок, статистика, здобич, побічні квести, великі міста, небезпечні підземелля та безліч міні-ігор.

Переваги:

Унікальне поєднання класичних та сучасних RPG елементів.

Глибока та захоплююча історія.

Стратегічна бойова система з інноваційною механікою "overdrive".

Широка доступність на різних платформах.

Високо оцінена графіка і музика.

Недоліки: Специфічні недоліки не були виявлені у наведених джерелах, але можливо, деякі гравці можуть вважати недоліком відхід від традиційних систем рівнів у RPG.

Джерело інформації : Офіційний сайт гри https://www.chainedechoes.com/.[4]

Рисунок 1.4 – Скріншот геймплею гри Chained Echoes [4]

Medieval Dynasty

Назва: Medieval Dynasty

Розробник: Render Cube

Виробник: Toplitz Productions

Архітектура: Гра розроблена для платформ Windows.

Мова реалізації: C++

Основні функції та характеристики:

Поєднання жанрів: Гра включає елементи виживання, симуляції, рольових ігор та будівництва міста

Відкритий світ: Гравці досліджують великий 3D середньовічний ландшафт

Динамічне середовище: Включає зміни пори року, динамічний цикл дня та ночі та реалістичні погодні умови

Крафтинг та будівництво: Обширна система крафтингу з понад 300 предметами та різноманітні будівлі для конструкції

Соціальні та сімейні динаміки: Гравці можуть встановлювати відносини, одружуватися і навіть виховувати сім'ю в межах гри

Переваги:

Занурювальний геймплей: Пропонує багатий, занурювальний ігровий досвід з поєднанням різних ігрових жанрів

Широкі можливості налаштування: Включає детальний створювач персонажів та динамічні взаємодії з середовищем

Взаємодіючий світ: Світ активно реагує на дії гравця, підвищуючи реалізм та залученість

Недоліки:

Потреба в шліфуванні: Гра вимагає значного шліфування, що може не сподобатись усім гравцям.

Одноманітність будівництва міст: Деякі гравці можуть вважати аспекти будівництва міст повторюваними з часом

Неідеальні механіки: Деякі елементи, такі як бойова система та полювання, можуть здаватися незграбними і впливати на загальну плавність гри.

Джерело інформації:

https://store.steampowered.com/app/1129580/Medieval_Dynasty/ [9]

Рисунок 1.5 – Скріншот геймплею гри Medieval Dynasty [9]

Hero of the Kingdom II

Назва: Hero of the Kingdom II

Розробник та Видавець: Lonely Troops

Архітектура: Гра розроблена для Windows, Linux і MacOS

Мова реалізації: C++

Основні функції та характеристики:

Пригоди та рольові елементи: Гравці виконують квести, взаємодіють з NPC та проходять через історію, сповнену завдань

Навички та квести: Гравці вивчають різноманітні навички, такі як риболовля, полювання та збір, і беруть участь у багатьох квестах

Дослідження: Гра має великий світ, який гравці можуть досліджувати, відвідуючи різні острови та середовища.

Збір предметів: Великий акцент робиться на пошуках корисних предметів, які допомагають у прогресі гри.

Переваги:

Захоплюючий сюжет: Сюжет про рятування сестри та боротьбу з піратами тримає гравців в напрузі.

Багатий геймплей: Пропонується суміш пригод, рольових елементів та стратегії з різними навичками та квестами.

Недоліки:

Обмежена повторюваність: Через лінійний сюжет і механіку гри, деякі гравці можуть відчути обмежену цінність повторного проходження після завершення основних цілей.

Простота: Деякі геймери можуть вважати механіки гри надто простими або не настільки складними порівняно з більш комплексними РПГ.

Джерело інформації: https://www.gog.com/game/hero_of_the_kingdom_ii [7]

Рисунок 1.6 – Скріншот геймплею гри Hero of the Kingdom II [7]

Огляд аналогічних систем у жанрі RPG вказує на значну різноманітність підходів до ігрового дизайну та механік. Сучасні розробники акцентують на глибині сюжету, інноваційних бойових системах та розширених можливостях взаємодії з ігровим світом. Значну увагу приділяється також мультиплатформенності, що дозволяє залучити ширший аудиторій гравців. Ці аспекти підкреслюють важливість врахування інновацій та трендів у розробці нових ігор.

Для успішного впровадження проєкту важливо забезпечити високу якість графіки та звуку, інтуїтивно зрозумілу і залучаючу бойову систему, та збалансовані ігрові механіки. Інтеграція динамічних змін середовища, таких як погодні умови чи час доби, може збагатити ігровий процес, зробити його більш реалістичним та непередбачуваним, що сприятиме зануренню гравців у віртуальний світ.

1.3 Специфікація вимог

1. Призначення та межі проєкту

1.1 Призначення системи (застосунку)

Ця 3D рольова гра розробляється для гравців, що цінують складні сюжети і можливість морального вибору. Гравець виступає в ролі персонажа, який через борги змушений приєднатись до бандитського угруповання. Гра пропонує варіанти вирішення ситуації: втеча, знищення банди або ж становлення її ватажком. Ціль гри — створити глибоко інтерактивний досвід, де вибір гравця визначає розвиток сюжету та взаємодії в ігровому світі.

1.3 Межі проєкту ПЗ

Проєкт обмежується розробкою на платформі Unity для ПК із підтримкою двох мов: англійської та української. Основні компоненти включають 3D графіку, систему квестів, бойові механіки, крафтинг і торгівлю. Проєкт не передбачає мультиплеєрний режим.

2. Загальний опис

2.1 Сфера застосування

Ця 3D RPG гра розроблена для використання на персональних комп'ютерах (PC) з операційними системами Windows. Гра підходить для одиночної гри і зосереджена на зануренні у вигаданий середньовічний світ, де гравці можуть взаємодіяти з персонажами, виконувати квести та розвивати свого персонажа в рамках бандитської ієрархії. Особлива увага приділена адаптивним сюжетним лініям, які реагують на вибори гравця, створюючи унікальний ігровий досвід при кожному проходженні.

2.2 Характеристики користувачів

Цільова аудиторія гри — це дорослі та підлітки віком від 16 років і старші, які цінують складність та глибину сюжету в рольових іграх. Користувачі, які мають досвід у грах жанру RPG, знайдуть цю гру особливо захоплюючою

завдяки її розгалуженій сюжетній лінії та комплексному розвитку персонажу. Гра також приваблива для тих, хто цінує можливість впливати на світ гри через моральні та стратегічні рішення. Користувачі повинні мати базові навички користування комп'ютером та здатність до читання та розуміння англійської або української мови, оскільки гра підтримує обидві мови.

2.3 Загальна структура і склад системи:

- управління персонажем: Інтерфейс для управління атрибутами, навичками, та розвитком персонажа;
- збереження гри.Можливості збереження та завантаження стану гри;
- система бою: Керування бойовими діями та інтеракціями з ворогами;
- збір ресурсів: Добування матеріалів необхідних у грі;
- система квестів: Управління, створення та відстеження квестів, вплив на сюжет та персонажа;
- меню гри: Інтерфейс, що дозволяє гравцям легко навігувати між різними розділами гри, налаштуваннями, та системами допомоги.

2.4 Загальні обмеження

Технічні обмеження: Вимога до високопродуктивного апаратного забезпечення, розробка лише під Windows, необхідність стабільного інтернет-з'єднання для онлайн функцій.

Правові обмеження: Дотримання авторських прав, захист конфіденційності даних, відповідність законам про цифрові товари.

Оперативні обмеження: Потреба в регулярних оновленнях та підтримці для стабільності та безпеки гри.

3. Функції системи

3.1 Функція системи – система бою

3.1.1 Опис функції

Система бою є центральним елементом геймплею, який забезпечує інтерактивність та динаміку в грі. Вона включає в себе різні види бойових дій,

такі як атака, оборона, використання спеціальних здібностей та магії. Система бою реалізована з метою надання гравцеві можливості стратегічного вибору та тактичного планування в залежності від ситуації на полі бою.

3.1.2 Вхідна і вихідна інформація

Вхідна інформація. Команди гравця (атака, оборона, використання предметів тощо), стан здоров'я персонажа, рівень енергії, доступні здібності.

Вихідна інформація. Результат бою, зміна стану здоров'я персонажів, зміна ігрового середовища, отримання досвіду та нагород.

3.1.3 Функціональні вимоги

Реалізація різноманітних бойових механік, які дозволяють гравцям використовувати широкий спектр стратегій.

Забезпечення реалістичної анімації та відповідних звукових ефектів для підвищення занурення в процес бою.

Інтеграція з системою штучного інтелекту для регулювання поведінки NPC під час бою.

Забезпечення балансу в бою для гарантування справедливих та захоплюючих битв.

3.2 Функція системи – управління персонажем

3.2.1 Опис функції

Система управління персонажем забезпечує контроль за рухом та камерою персонажа, дозволяючи гравцям здійснювати дії, такі як біг, прискорення, стрибки та обертання камери. Вона включає компоненти, що відстежують торкання землі, обертання тіла гравця та взаємодію з фізичними об'єктами у грі.

3.2.2 Вхідна і вихідна інформація

Вхідна інформація. Ввідні дані від миші та клавіатури для управління рухом персонажа та камерою, дані від датчика торкання землі.

Вихідна інформація. Поточний стан персонажа, включаючи його положення, обертання та стан на землі.

3.2.3 Функціональні вимоги

Розробка системи управління камерою, яка дозволяє гравцям обертати камеру за допомогою миші.

Забезпечення можливості управління рухом персонажа, включаючи біг, прискорення та стрибки, з використанням введення від клавіатури.

Інтеграція системи перевірки торкання землі, яка відстежує стан персонажа на землі та викликає відповідні події.

3.3 Функція системи – інвентар та управління предметами

3.3.1 Опис функції

Система інвентарю та управління предметами дозволяє гравцям зберігати, організовувати, використовувати та викидати предмети, які вони знаходять або створюють під час гри. Ця система спрощує управління ресурсами та обладунком, що є ключовим для успішного просування в грі.

3.3.2 Вхідна і вихідна інформація

Вхідна інформація. Команди гравця на переміщення, використання або викидання предметів.

Вихідна інформація. Зміни в інвентарі, стан предметів (наприклад, кількість або знос).

3.3.3 Функціональні вимоги

Інтуїтивно зрозумілий інтерфейс з можливістю категоризації та сортування предметів.

Підтримка швидкого доступу до ключових предметів, таких як зброя або зілля.

3.4 Функція системи – система квестів

3.4.1 Опис функції

Система квестів є ключовим елементом гри, який направляє гравця через ігровий світ та сюжет. Ця система включає різноманітні завдання та місії, що варіюються від простих до складних, і залучають гравця у взаємодії з NPC,

дослідженні локацій, зборі предметів, виконанні специфічних цілей, та боротьбі з противниками.

3.4.2 Вхідна і вихідна інформація

Вхідна інформація. Вибір квестів гравцем, дії, необхідні для виконання завдань, взаємодії з об'єктами та персонажами.

Вихідна інформація. Оновлення статусу квестів, нагороди, зміни в сюжеті та стані світу, реакції персонажів.

3.4.3 Функціональні вимоги

Розробка різноманітних квестів зі збалансованими цілями, які відповідають тематиці світу та персонажам. Квести мають включати як лінійні, так і нелінійні елементи, що дозволяють гравцям вибирати, яким шляхом слідувати.

Інтеграція сюжетних розвилок у квестах, щоб забезпечити глибину та розвиток історії. Важливі квести можуть мати вплив на загальний стан світу та відносини з іншими персонажами.

Система відстеження прогресу квестів, що дозволяє гравцям легко знаходити необхідну інформацію про поточні та завершені завдання, їх локації та статус.

Динамічна реакція світу на дії гравця у квестах. Наприклад, вибір гравця може вплинути на доступність майбутніх квестів, відносини з ключовими NPC або навіть на візуальний стан світу.

Забезпечення зворотного зв'язку та відповідних нагород за виконання квестів, що можуть включати досвід, предмети, валюту гри, зміни в репутації.

3.5 Функція системи – збереження гри

3.5.1 Опис функції

Система збереження гри дозволяє гравцям зберігати свій прогрес у будь-який момент гри або в автоматичному режимі на певних контрольних пунктах.

Це критично важлива функція, що надає гравцям можливість продовжувати гру з моменту останнього збереження, зберігаючи всі здобутки та вибори.

3.5.2 Вхідна і вихідна інформація

Вхідна інформація. Команда гравця на збереження гри або досягнення певної точки в грі, яка активує автозбереження.

Вихідна інформація. Файл збереженої гри, який містить всю інформацію про поточний стан гри.

3.5.3 Функціональні вимоги

Надійність системи збереження для запобігання втрати даних.

Забезпечення кількох слотів для збереження, щоб гравці могли зберігати різні стадії гри.

Інтерфейс для легкого доступу до збереження/завантаження станів гри.

Можливість автоматичного збереження в критичних моментах або після важливих подій.

3.6 Функція системи – торгівля з NPC

3.6.1 Опис функції

Система торгівлі з NPC дозволяє гравцям купувати, продавати та обмінювати предмети з різними NPC, що виконують роль торговців або інших персонажів у грі, залежно від їхньої ролі та локації.

3.6.2 Вхідна і вихідна інформація

Вхідна інформація. Перелік предметів у гравця та NPC, пропозиції цін, команди гравця на купівлю чи продаж.

Вихідна інформація. Зміни в інвентарі гравця та NPC, зміна балансу валюти гравця.

3.6.3 Функціональні вимоги

Забезпечення інтуїтивно зрозумілого інтерфейсу для ведення торгів.

Інтеграція з економічною системою гри для відслідковування економічних тенденцій і впливу гравця на ринок.

4. Вимоги до інформаційного забезпечення

4.1 Джерела і зміст вхідної інформації:

- ігрові дані: Параметри персонажів, статистика ворогів, мапи, зброя;
- мета-дані: Прогрес гравця, збереження стану гри;
- сюжетні елементи: Діалоги, описи місій, історія світу;
- конфігураційні дані: Графічні та аудіо налаштування, управління;
- зовнішні дані: Інтеграція з ігровими сервісами.

4.2 Нормативно-довідкова інформація:

- класифікатори: Категорії зброї, броні, зілля;
- системи рангування: Рівні складності, рівні персонажів;
- довідкові таблиці: Ефективність зброї, бонуси предметів.

4.3 Вимоги до організації, збереження та обробки інформації:

- бази даних: Бінарна і JSON серіалізація;
- безпека даних: Шифрування даних;
- резервне копіювання: Стратегії для відновлення даних;
- оптимізація продуктивності: Механізми кешування та асинхронного завантаження.

5. Вимоги до технічного забезпечення

Мінімальні вимоги:

- ОС: Windows 10 (64-бітні);
- процесор: Intel Core i5-3570K або AMD FX-8310;
- оперативна пам'ять: 8 ГБ;
- відеокарта: NVIDIA GeForce GTX 760 або AMD Radeon RX 470;
- DirectX: версія 12;
- місце на диску: 15 ГБ.

Рекомендовані вимоги:

- ОС: Windows 10 (64-бітні);
- процесор: Intel Core i7-4790 або AMD Ryzen 3 3200G;

- оперативна пам'ять: 12 ГБ;
- відеокарта: NVIDIA GeForce GTX 1060 6GB / GTX 1660 Super або Radeon RX 590;
- DirectX: версія 12;
- місце на диску: 15 ГБ.

Пристрої для розробки

Мінімальні специфікації: Quad-core Intel або AMD процесор, 16 GB RAM, сучасна графічна карта.

Тестувальні пристрої

Набір різноманітних комп'ютерів та мобільних пристроїв для перевірки сумісності та оптимізації гри під різні системи.

Мережеве обладнання

Надійний інтернет-маршрутизатор та комутатор для забезпечення стабільного інтернет-з'єднання, важливого для хмарних резервних копій та командної співпраці.

6. Вимоги до програмного забезпечення

6.1 Архітектура програмної системи

Використання Unity для створення ігрового двигуна, систем квестів, бою, навичок та UI. Архітектура підтримує модульність, масштабованість та надійність.

6.2 Системне програмне забезпечення

Сумісне з Windows та Linux; оптимізоване для ефективного використання ресурсів.

6.3 Мережне програмне забезпечення

Управління мережевими взаємодіями, включаючи синхронізацію з хмарними збереженнями та оновлення гри; забезпечення безпеки та стабільності з'єднань.

6.4 Програмне забезпечення для ведення інформаційної бази

Використання Player Prefs, бінарної та JSON серіалізації для зберігання даних; впровадження заходів безпеки та механізмів відновлення після збоїв.

6.5 Мова і технологія розробки ПЗ

Використання Unity та C# для розробки забезпечує швидкість та сумісність; підтримка великої спільноти розробників.

7. Вимоги до зовнішніх інтерфейсів

7.1 Інтерфейс користувача (UI)

Гра має інтуїтивно зрозумілий та відповідний UI, що включає інтерактивне головне меню, ігрові HUD елементи (індикатори здоров'я, мани, інвентар), екрани налаштувань, та інтерфейс квестів. UI має підтримувати різні роздільні здатності екрану та бути адаптованим під контроль за допомогою миші або клавіатури.

7.2 Апаратний інтерфейс

Гра розроблена для ПК та має підтримувати стандартні вхідні пристрої для ПК, включно з клавіатурою, мишею, та опційно — геймпадами. Рекомендовані системні вимоги (мінімальні та оптимальні конфігурації апаратного забезпечення) мають бути чітко вказані, включно з процесором, оперативною пам'яттю, графічною карткою, та місцем на жорсткому диску.

7.3 Програмний інтерфейс (API)

Хоча гра є однокористувацькою, можлива інтеграція з хмарними збереженнями (наприклад, Steam Cloud) для зберігання ігрових прогресів та налаштувань. Також може включатися інтеграція з ігровими платформами для досягнень та лідерських таблиць.

7.4 Комунікаційний протокол

Хоча гра не має онлайн мультиплеєра, використання протоколів для безпечного з'єднання з хмарними сервісами та оновленнями гри є важливим. Вимоги до безпеки даних та стабільності підключення через HTTPS, TCP/IP для підтримки безперебійних оновлень і синхронізацій.

8. Властивості програмного забезпечення

8.1 Доступність

Гра повинна бути доступна для гри 24/7, окрім планового технічного обслуговування, забезпечуючи гравцям можливість грати в будь-який час без обмежень.

8.2 Супроводжуваність (Ремонтопридатність)

Виявлення та виправлення будь-яких помилок мають бути здійснені в межах 72 годин з моменту їх повідомлення, забезпечуючи швидке реагування на звіти користувачів.

8.3 Переносимість

Гра має бути сумісною та функціональною на різних версіях Windows (Windows 10 та Windows 11) без потреби в модифікаціях або спеціальних налаштуваннях.

8.4 Продуктивність

Гра повинна підтримувати стабільну частоту відображення кадрів (мінімум 60 FPS) на мінімальних системних вимогах, а також забезпечувати гладкий ігровий досвід на оптимальних налаштуваннях.

8.5 Надійність

Гра має забезпечувати стабільну та безперервну роботу, уникнення системних збоїв у 99.9% часу використання, мінімізуючи ризик збоїв або втрати ігрового прогресу.

8.6 Безпека

Забезпечення захисту програмного забезпечення від несанкціонованого доступу, злому та інших видів кібератак. Впровадження заходів для забезпечення безпеки даних користувачів.

Локалізація

Гра має підтримувати повну локалізацію на англійську та українську мови, включаючи всі текстові діалоги, ігрові інструкції та UI.

Зручність використання

Інтерфейс користувача повинен бути інтуїтивно зрозумілим і легким для навігації, з мінімізацією помилок у вводі та взаємодії з гравцем.

Висновки до розділу 1

У процесі стратегічного планування та аналізу вимог до ігрового застосунку на рушії Unity було виконано всебічний аналіз ігрової індустрії, огляд аналогів та специфікацію вимог до майбутньої гри. Ретельне дослідження дозволило визначити призначення системи та її межі, забезпечити чітке розуміння сфери застосування, характеристик користувачів, структури та обмежень системи.

Розробка функцій системи базується на глибокому аналізі потреб користувачів та актуальних трендів ринку, що забезпечує створення комплексного і конкурентоспроможного продукту. Кінцева архітектура та вибір технологій для розробки програмного забезпечення є відображенням стратегічних рішень, прийнятих на основі аналітичних даних і спрямованих на досягнення оптимальної продуктивності, надійності та доступності ігрового застосунку. Таким чином, проведена робота є фундаментом для успішної розробки, запуску та подальшого розвитку проєкту.

2 РОЗРОБКА КОНЦЕПЦІЇ ТА МОДЕЛЮВАННЯ ГРИ

2.1 Концептуальний аналіз гри

Гра розгортається в умовному середньовічному світі, де гравець взаємодіє з угрупуванням бандитів. Взаємодія передбачає виконання різних квестів, які поділяються на кілька рівнів залежно від статусу персонажа у бандитській ієрархії: впливовий член бандитів, середній член бандитів та новачок.

Структура квестів

Впливовий член бандитів. Завдання на цьому рівні мають стратегічний характер та вплив на загальну динаміку групи, включаючи підкуп і залякування інших бандитів, співпрацю з владою, протистояння владі, викрадення важливих осіб, кару зрадників, захоплення територій і фінальне протистояння з головними антагоністами;

Середній член бандитів. Квести на цьому рівні зосереджені на тактичних аспектах бандитської діяльності, включаючи торгівлю зброєю, придушення свідків, захист територій, рейди на конкурентів, контрабанду рідкісних товарів, викрадення бандитських артефактів, втечу заручників, диверсії та звільнення від рабського ярма;

Новачок. Завдання для новачків спрямовані на введення гравця в бандитське життя, включаючи крадіжки, зтягнення боргів, ініціацію, шпигунство, збір інформації та налагодження зв'язків з іншими персонажами.

Гра пропонує динамічну взаємодію між персонажами з різних рівнів ієрархії, кожен з яких вносить свій вклад у загальну сюжетну лінію. Це дозволяє гравцям вибирати свій шлях у світі гри, що робить процес більш персоналізованим та занурюючим. Крім того, такий підхід до різноманітності квестів сприяє повторному проходженню гри, оскільки гравець може вибрати інші стратегії або ролі. Кожна роль і квест не тільки розкривають особливості

персонажа, але й відображають більш глибокі теми та конфлікти в рамках ігрового світу, створюючи багатошаровий сюжет.

Використання широкого спектру квестів, від простих завдань для новачків до складних сценаріїв для впливових членів банди, забезпечує постійне відчуття розвитку та зростання персонажа. Гравці можуть відчути наслідки своїх виборів, що надає додаткову вагу кожному рішенню в рамках гри.

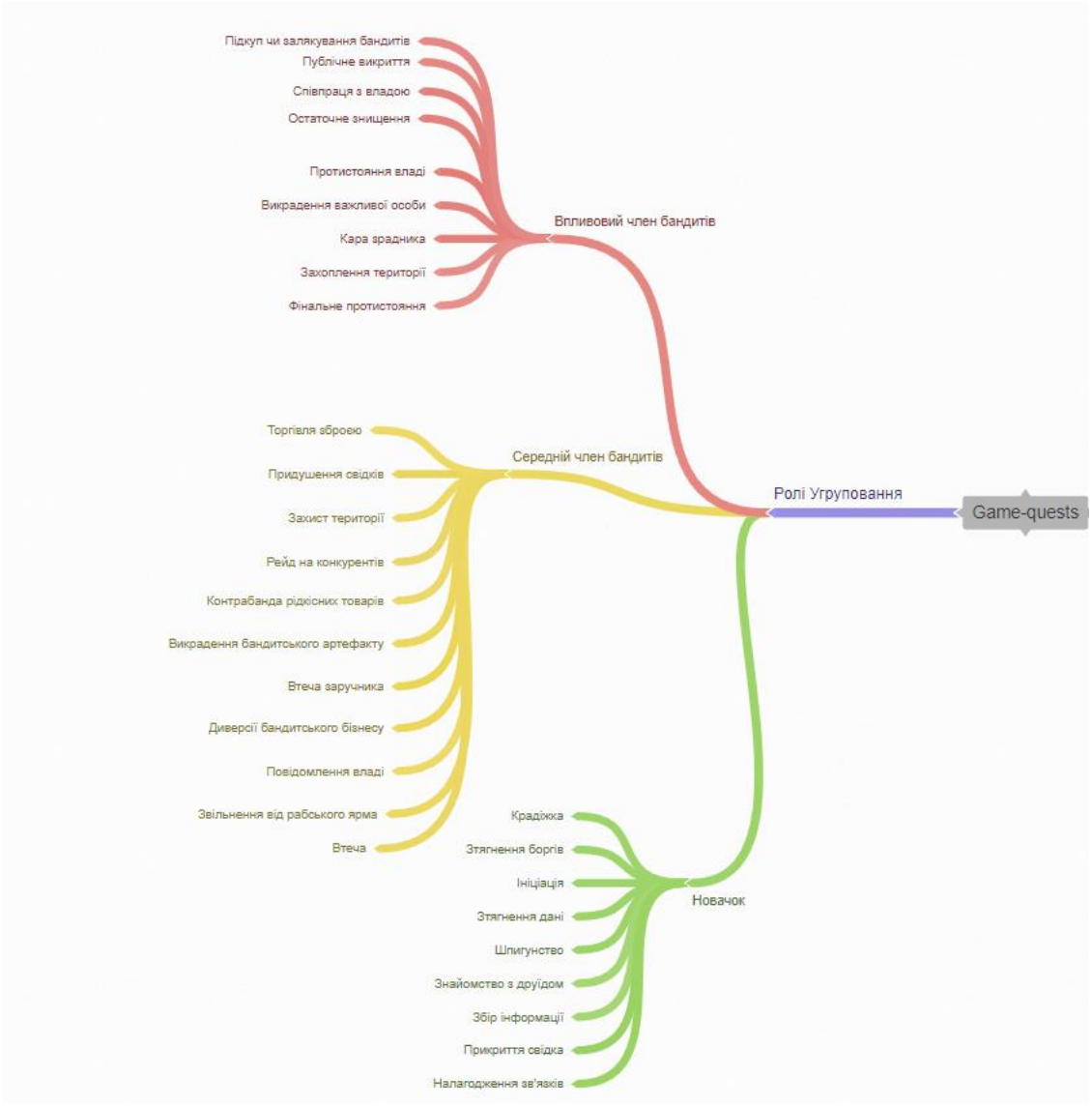

Рисунок 2.1 – Структура квестів

На представленій mind map діаграмі відображено структуру квестів у грі, розділену на три основні рівні ієрархії бандитської організації: впливовий член, середній член та новачок. Кожен рівень має свої специфічні завдання, що

відображаються різними кольорами на діаграмі, допомагаючи зорово розрізняти типи активностей, які має виконати гравець, залежно від його статусу та ролі в ієрархії.

2.2 Створення сценаріїв використання

Для детального огляду та забезпечення зрозумілої взаємодії між користувачем і системою були розроблені відповідні сценарії використання. Розробка Use Case сприяє налагодженню чітких вимог до функціональності продукту, а також допомагає уникнути потенційних проблем у процесі впровадження нових ігрових механік.

Таблиця 2.1 – Use Case Виконання квесту в RPG грі

Section	**Details**
Use Case Name	Виконання квесту
Scope	RPG гра на платформі Unity
Level	Виконати заданий квест і отримати відповідні нагороди.
Primary Actor	Гравець
Stakeholders and Interests	Гравець: Завершення квестів для здобуття досвіду, ресурсів і просування персонажа у грі.
Preconditions	Гравець має бути авторизований в Steam. Квест має бути доступний і відповідати рівню персонажа.
Success Guarantee	Гравець виконує всі задачі квесту. Гравець отримує відповідні нагороди.

Кінець таблиці 2.1

Main Success Scenario	Гравець отримує квест від NPC або через ігрове меню. Гравець вирушає до локацій, вказаних у квесті. Гравець збирає необхідні предмети або перемагає ворогів. Гравець повертається до NPC для здачі квесту. Гравець отримує нагороди за завершення квесту.
Extensions	9.1 Гравець не може знайти предмет або ворога: Система надає підказки або маркери на карті. Гравець використовує підказки для продовження пошуку. 9.2 Гравець помирає під час виконання квесту: Гравець воскресає в останній збереженій точці. Гравець може спробувати виконати квест знову або краще підготуватись.
Special Requirements	Інтерфейс користувача має чітко відображати стан квесту та його етапи. Швидкість завантаження ігрових рівнів має бути оптимізована для підтримки плавного геймплею.
Frequency of Occurrence	Квести можуть виконуватися часто, залежно від активності гравця.
Miscellaneous	Перед початком квесту система перевіряє, чи відповідає рівень персонажа вимогам квесту.

Таблиця 2.2 – Use Case Бій з ворогом

Section	Details
Use Case Name	Бій з ворогом
Scope	RPG гра на платформі Unity
Level	Взаємодія на рівні гравця з ігровим світом.
Primary Actor	Гравець

Продовження таблиці 2.2

Stakeholders and Interests	Гравець: здолати ворогів для отримання досвіду, предметів та прогресу в сюжеті.
Preconditions	Гравець має бути авторизований в Steam. Гравець має наявність потрібного озброєння та бойових здібностей. Вороги мають бути присутні у поточній локації.
Success Guarantee	Ефективне використання бойових команд та здібностей. Система коректно обробляє дії гравця та ворогів, показуючи результати бою.
Main Success Scenario	Гравець виявляє ворога в локації. Гравець ініціює бій, використовуючи доступні бойові здібності. Система обраховує та відображає ушкодження та результати атак. Гравець перемагає ворога. Система нараховує досвід та видає здобуті предмети.
Extensions	9.1 Гравець програє бій: Гравець може воскреснути у безпечному місці або останній точці збереження. Гравець може повторити бій після підготовки. 9.2 Система помилок під час бою: Гравець повідомляє про проблему. Система перезавантажується або виправляє помилку.

Кінець таблиці 2.2

Special Requirements	Система повинна відображати стан здоров'я, ману та інші бойові параметри у режимі реального часу. Висока швидкість реакції системи на команди гравця для забезпечення плавного та реалістичного бою.
Frequency of Occurrence	Бої з ворогами відбуваються часто, особливо у зонах з високою концентрацією ворогів або під час сюжетних місій.
Miscellaneous	Для підтримки залучення гравця, можна впровадити адаптивні музичні теми, які змінюються в залежності від інтенсивності бою та здоров'я гравця.

Таблиця 2.3 – Use Case Збір ресурсів

Section	**Details**
Use Case Name	Збір ресурсів
Scope	RPG гра на платформі Unity
Level	Збір ресурсів для крафтингу предметів або торгівлі.
Primary Actor	Гравець
Stakeholders and Interests	Гравець: Збирати ресурси необхідні для покращення спорядження та виконання квестів.
Preconditions	Гравець має бути авторизований в Steam. Ресурси мають бути доступні для збору в поточній локації.
Success Guarantee	Система належно відстежує і реєструє всі дії гравця під час збору ресурсів. Гравець знає, як та де збирати ресурси.
Main Success Scenario	Гравець виявляє ресурс у світі гри. Гравець використовує відповідний інструмент або здібності для збору ресурсу. Система підтверджує збір і додає ресурс до інвентарю гравця.

Кінець таблиці 2.2

Extensions	9.1 Ресурс не доступний: Система повідомляє гравця, що ресурс тимчасово недоступний. Гравець може повернутися пізніше або шукати альтернативні джерела. 9.2 Інструмент зламався: Гравець повинен відремонтувати або замінити інструмент перед продовженням збору.
Special Requirements	Інтерфейс має забезпечувати інформацію про доступність та кількість ресурсів у реальному часі. Система має оптимізувати процес збору ресурсів, забезпечуючи плавний геймплей без затримок.
Frequency of Occurrence	Збір ресурсів відбувається досить часто, особливо у зонах, багатих на ресурси або під час виконання побічних завдань.
Miscellaneous	Розробити події обмеженого часу, які збільшують наявність ресурсів у певних локаціях, стимулюючи гравців досліджувати нові зони.

У RPG грі на платформі Unity, виконання квестів, бій з ворогами та збір ресурсів є ключовими сценаріями використання, кожен з яких відіграє важливу роль у структурі та динаміці гри.

Виконання квестів є основою для сюжету і розвитку персонажів, оскільки це дозволяє гравцям відкривати нові ігрові зони, отримувати досвід і предмети. Ці завдання стимулюють дослідження світу та взаємодію з ігровими персонажами, що робить ігровий процес занурюючим та персоналізованим. Квести не тільки поглиблюють сюжетну лінію, але й мотивують гравців продовжувати гру, забезпечуючи стале просування та відкриття нових можливостей.

Бій з ворогом є центральним елементом багатьох RPG, що забезпечує емоційну напругу та реалізацію стратегічних навичок гравця в реальному часі. Ефективне використання бойових команд та навичок важливе для успішного просування в грі, що вимагає від гравців адаптації та покращення своїх тактик. Ці взаємодії не тільки підтримують інтерес та залученість гравця, але й важливі для балансу складності та рівня виклику в грі.

Збір ресурсів сприяє розширенню можливостей гравця через крафтинг та торгівлю. Цей процес забезпечує глибину геймплею, дозволяючи гравцям створювати або покращувати предмети, що використовуються у подальших боях чи квестах. Взаємодія з ресурсами стимулює дослідження світу гри та розвиток інтерактивності, забезпечуючи додаткові цілі та мотивацію для продовження гри.

Загалом, кожен з цих сценаріїв використання важливий для створення збалансованого і занурюючого ігрового досвіду, який підтримує зацікавленість та взаємодію гравців на всіх етапах гри.

2.3 Графічне моделювання головних елементів системи

Для зображення взаємодії між гравцем та системою гри була розроблена діаграма, яка демонструє як гравець взаємодіє з різними функціями гри. Ця діаграма відображає основні зв'язки та механізми управління, забезпечуючи зрозуміле представлення взаємодій усередині ігрового середовища.

Діаграма варіантів використання, ілюструє ключові сценарії використання в RPG грі на платформі Unity, демонструючи взаємозв'язки між різними функціями гри. Центральний елемент — гравець, який інтерактивно взаємодіє з системою гри, що включає різноманітні активності, такі як збереження гри, крафтинг, домівка, бойова система, синхронізація досягнень, управління персонажем та збір ресурсів.

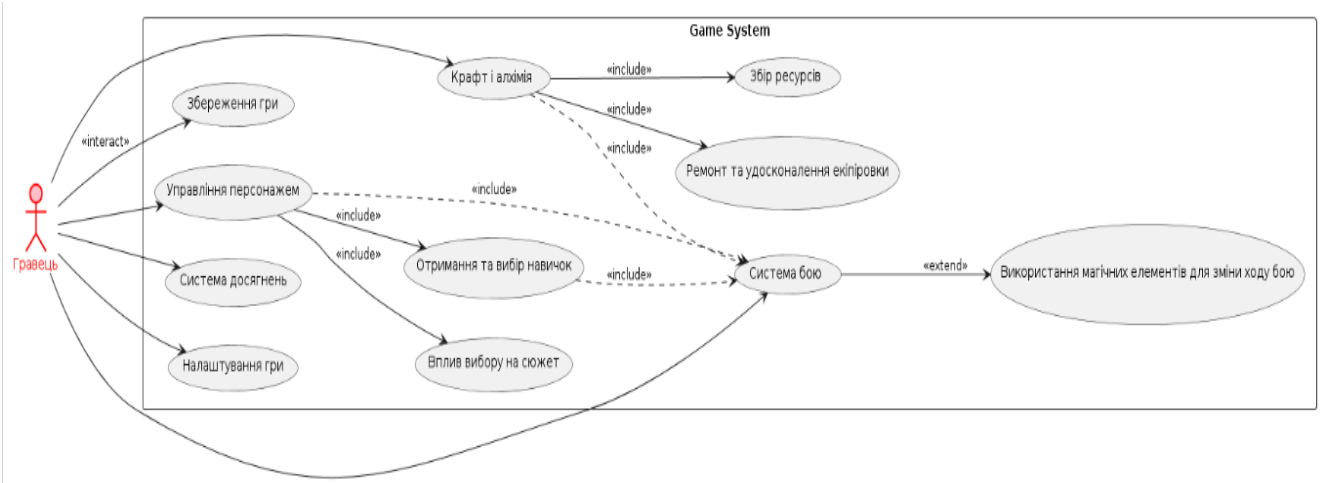

Рисунок 2.2 – Діаграма варіантів використання

Збереження гри є фундаментальним аспектом, який забезпечує можливість зберегти та відновити прогрес гравця в будь-який час, підтримуючи плавність гри та забезпечуючи відчуття безпеки. Функція крафту та управління домівкою дає гравцям можливість персоналізувати своє ігрове середовище та впливати на ігровий світ, що значно збільшує залученість та інтерес до гри. Ремонт та удосконалення екіпіровки також сприяють глибшій інтеракції та мотивації до розвитку персонажа.

Система бою є ключовою для забезпечення динаміки та емоційного залучення гри, дозволяючи гравцям використовувати та розвивати свої бойові навички в конфронтаціях з ворогами. Синхронізація досягнень інтегрована для підтримки соціального аспекту гри, дозволяючи гравцям ділитися своїми успіхами та залучати інших у свій ігровий процес.

Збір ресурсів виконує критичну роль у підтримці економіки гри та надає гравцям задачі для дослідження світу та активної участі у крафтингу. Всі ці елементи тісно пов'язані та мають взаємні залежності, що створює замкнений цикл ігрових активностей, кожна з яких впливає на інші та забезпечує безперервний розвиток та зацікавленість гравця.

Таким чином, діаграма підкреслює взаємозалежність і значення кожного аспекту гри, підтримуючи багатошаровий, динамічний і захоплюючий ігровий досвід, який стимулює гравців до постійного розвитку та взаємодії з ігровим світом.

Представлені діаграми діяльності відображають одні з основних систем в ігровому застосунку, ілюструючи як системи бою, торгівлі, та крафтингу організовані та як вони взаємодіють з гравцем. Кожна з цих діаграм візуалізує критичні механіки гри, які впливають на геймплей та занурення гравця у віртуальний світ.

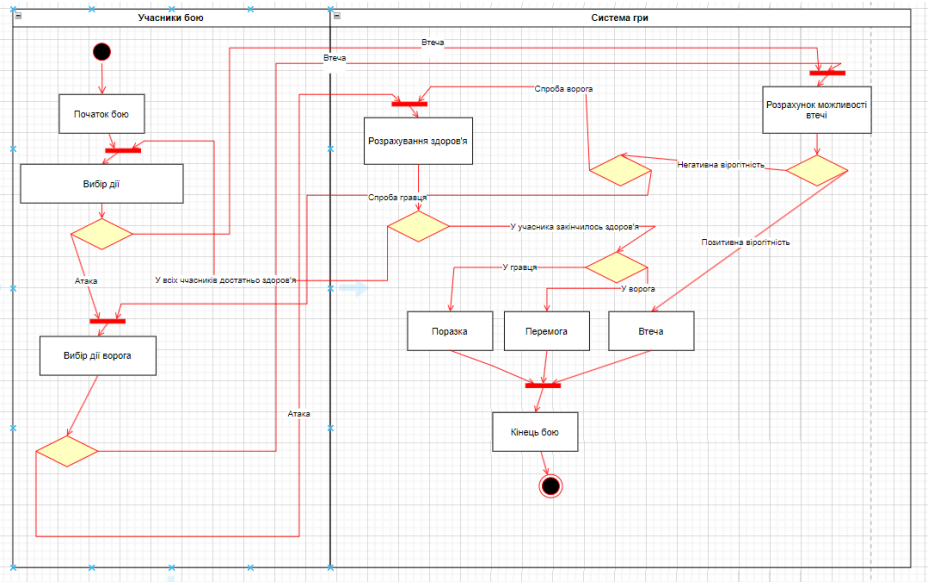

Рисунок 2.3 – Діаграма діяльності бойової системи

Система бою відображає процеси прийняття рішень під час бою, де гравці вибирають різні тактичні опції, такі як атака, захист, або втеча, в залежності від обставин на полі бою. Діаграма показує можливі результати цих рішень, включно з перемогою, поразкою, або відступом, кожне з яких має наслідки для подальшого просування в грі. Система бою безпосередньо впливає на прогрес гравця та дозволяє прокачувати навички персонажа.

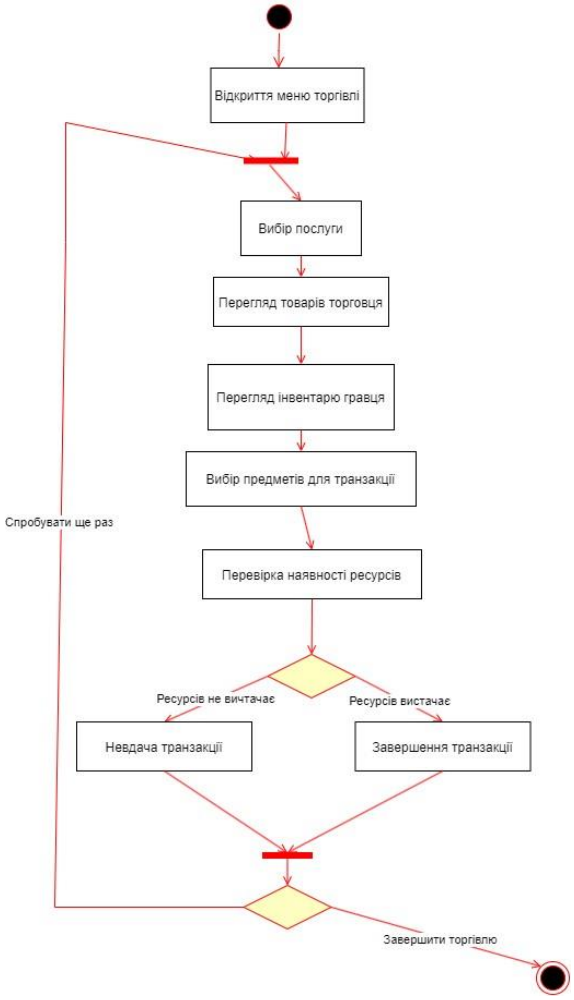

Рисунок 2.4 – Діаграма діяльності торгівлі

Торговельна система дозволяє гравцям взаємодіяти з ігровими торговцями, вибираючи між купівлею та продажем предметів. Діаграма описує процес перегляду інвентаря, вибору предметів для транзакцій, і перевірки наявності ресурсів для завершення транзакцій. Це дозволяє гравцям керувати своїми ресурсами та покращувати своє оснащення, що безпосередньо впливає на їхню здатність конкурувати в грі.

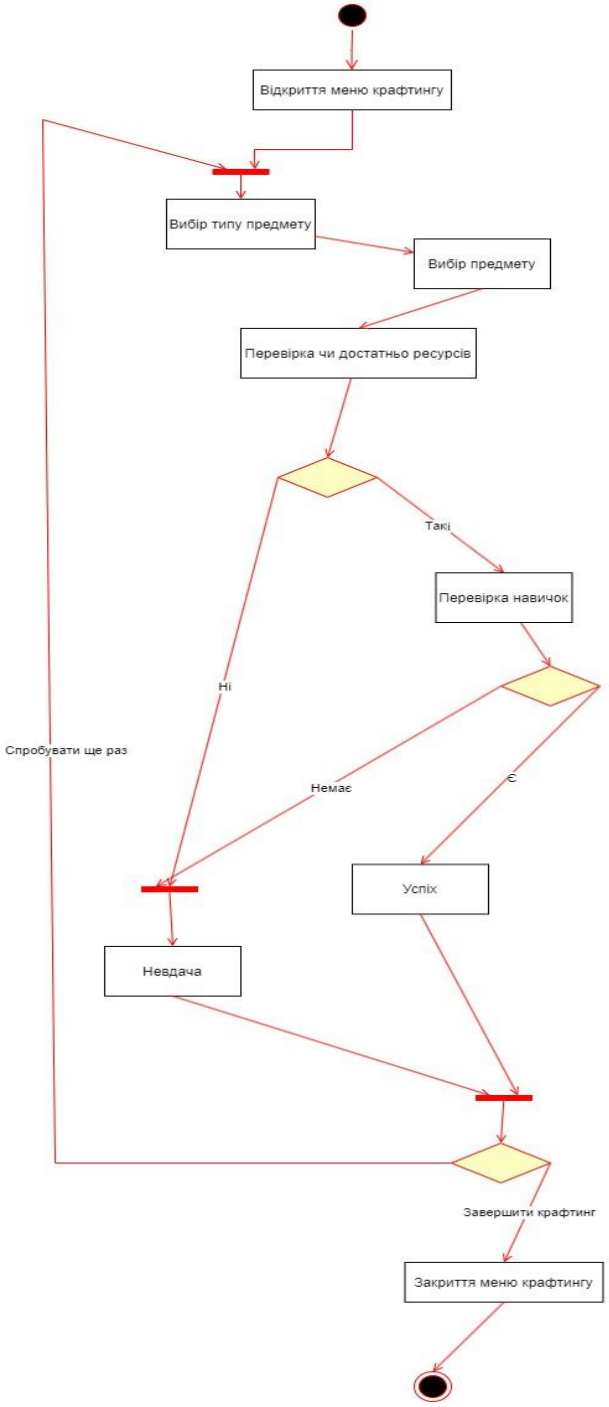

Рисунок 2.5 – Діаграма діяльності системи створення предметів

Система крафтингу ілюструє кроки, необхідні для створення нових предметів, які можуть бути використані в грі. Від відкриття меню крафтингу до вибору предметів для створення та перевірки наявності необхідних ресурсів, ця система дозволяє гравцям розширювати свої можливості через крафтинг

зібраних ресурсів. Успішний крафтинг веде до покращення екіпіровки та інших ключових аспектів ігрового процесу.

Отже, ці діаграми візуалізують одні з основних механік гри, кожна з яких є життєво важливою для забезпечення глибокого і захоплюючого ігрового досвіду, дозволяючи гравцям активно взаємодіяти з ігровим світом та впливати на свою долю в цьому світі.

2.4 Етапи розробки проєкту

Етапи розробки гри :

- концепція;
- пре-продакшн;
- продакшн;
- пост-продакшн;
- реліз;
- підтримка після запуску. [11]

Етап концепції в розробці відеоігор є ключовим, адже він закладає основу для всієї подальшої роботи над проєктом. На цьому етапі розробницька команда визначає основні аспекти гри: ідею, жанр, ключові механіки та інновації, які допоможуть виділити гру на ринку. Процес розпочинається з формулювання ідеї, де команда вибирає унікальну концепцію або взаємодію, яка приваблюватиме гравців. Встановлення жанру гри створює основні рамки для розробки механік та сюжету. Важливі механіки, такі як бойові системи, взаємодія з ігровим світом чи крафтинг, розробляються для підтримки геймплею.

Після визначення основної концепції команда приступає до створення базового дизайн-документу, який стане керівництвом для всіх аспектів гри. У цьому документі детально описуються сюжет гри, персонажі, ігровий світ та механіки, які будуть використані. Опис сюжету включає головну сюжетну лінію та ключові події, а персонажі описуються з урахуванням їхніх мотивацій та

ролей. Ігровий світ деталізується з урахуванням локацій та їхнього візуального та функціонального дизайну. Також планується розробка ігрових механік і інтерфейсу користувача для забезпечення інтуїтивно зрозумілої та захоплюючої взаємодії з грою.

Цей етап не лише визначає напрямок розвитку гри, але й забезпечує єдність візії серед усієї команди розробників, що дозволяє ефективно керувати подальшими етапами проєкту, такими як прототипування, продакшн і пост-продакшн. Від успішної реалізації концептуалізації залежить здатність гри залучати та утримувати увагу гравців, її комерційний успіх та вплив на ігрову індустрію.

На етапі пре-продакшну команда зосереджує свої зусилля на детальному плануванні та прототипуванні основних ігрових механік, що дозволяє тестувати і вдосконалювати ідеї перед значними інвестиціями в їх повномасштабне виробництво. Створення прототипів допомагає валідувати геймплей і інтерфейс, а також розробляється детальна проєктна документація, що охоплює всі аспекти гри від сюжету до технічних специфікацій.

Команда також працює над арт-концепціями, визначаючи візуальний стиль гри, який включає персонажів, локації та предмети. Це допомагає узгодити естетичне бачення проєкту і забезпечує основу для художньої продукції на наступних етапах. Технічне планування відіграє важливу роль, оскільки програмісти визначають архітектуру і технології, які будуть використані для реалізації гри. Обираються відповідні ігрові движки та інструменти, що забезпечують необхідну функціональність та продуктивність.

На етапі продакшну команда розробників фокусується на створенні візуального та аудіального контенту, програмуванні ігрових механік та інтеграції всіх компонентів у єдину систему. Здійснюється оптимізація продуктивності та стабільності гри, а також її тестування і полірування. Процес інтеграції вимагає

великої уваги до деталей та ретельної координації між командами для забезпечення єдності і неперервності ігрового досвіду.

Етап пост-продакшну є вирішальним для забезпечення довгострокового успіху та стабільності відеоігри на конкурентному ринку. Він включає детальне доопрацювання ігрових елементів, інтенсивне фінальне тестування та оптимізацію продуктивності гри на різних платформах. Під час цього етапу команда розробників зосереджує зусилля на детальному доопрацюванні ігрових елементів, включаючи візуальні деталі, анімації, інтерфейси користувача, а також балансування геймплею.

Основною метою є виправлення будь-яких помилок та недоліків, які були виявлені під час тестування, для забезпечення плавного ігрового процесу. Крім полірування, на етапі пост-продакшну проводиться інтенсивне фінальне тестування, що дозволяє виявити залишкові помилки та оптимізувати продуктивність гри на різних платформах. Це забезпечує, що гра функціонує стабільно та ефективно на всіх цільових пристроях, незалежно від їх технічних специфікацій.

Процес оптимізації також є важливою частиною пост-продакшну, оскільки він включає налагодження роботи ігрових двигунів та систем для забезпечення найкращого користувацького досвіду. Після завершення тестування та оптимізації команда готує гру до релізу, забезпечуючи дотримання всіх технічних вимог платформ і маркетингових стратегій для просування гри.

Таким чином, етап релізу в розробці відеоігор є кульмінацією всіх попередніх зусиль і вимагає чіткої координації між розробниками, маркетинговою командою та підтримкою для забезпечення успішного виходу гри на ринок та її подальшого процвітання. На початку етапу релізу команда забезпечує, що всі елементи гри належно інтегровані та оптимізовані, включаючи завершення всіх тестів на помилки та забезпечення стабільної роботи гри на різних платформах. Потім гра проходить процес здачі на платформах,

який вимагає відповідності до специфічних вимог і стандартів кожної платформи, що може включати різні форми сертифікації та схвалення. Разом з технічною підготовкою, команда здійснює масштабну маркетингову кампанію, яка включає рекламу, прес-релізи, участь у ігрових виставках, співпрацю з впливовими блогерами та інші заходи для залучення уваги до гри. Це також час для активного використання соціальних мереж і створення відео-контенту, щоб підвищити інтерес і очікування від гри.

Після запуску команда продовжує моніторинг гри для швидкого виявлення та вирішення будь-яких технічних проблем, які можуть виникнути. Важливим є збір відгуків від гравців, які можуть вказати на необхідність додаткових коректив або покращень. Команда також займається розробкою оновлень та додаткового контенту, що допомагає утримати інтерес до гри та підтримувати активну гравецьку базу.

Етап підтримки після запуску є важливим для забезпечення довгострокового успіху відеоігри, який охоплює ряд критичних діяльностей спрямованих на підтримку, оптимізацію та розширення ігрового досвіду. Починаючи з виправлення помилок, розробники активно моніторять зворотній зв'язок від гравців і швидко реагують на будь-які технічні проблеми, що виникають після релізу, щоб забезпечити стабільність та надійність гри. Крім технічного підтримання, важливою складовою пост-релізної діяльності є введення нового контенту, такого як доповнення, нові рівні або персонажі, що допомагають зберігати інтерес гравців та залучати нових. Це також включає організацію спеціальних ігрових подій та турнірів, які сприяють залученню спільноти та підтримують активне ігрове середовище.

Оптимізація продуктивності також відіграє ключову роль у пост-релізному етапі, оскільки гра має працювати ефективно на всіх підтримуваних платформах. Це забезпечує, що гра адаптується до різноманітного апаратного забезпечення та технічних специфікацій, забезпечуючи гладкий ігровий досвід для широкого

спектру користувачів. Зворотній зв'язок та спілкування зі спільнотою є життєво важливими для успіху гри. Розробники використовують форуми, соціальні мережі та інші платформи для збору відгуків та ідей, які можуть бути використані для подальшого розвитку гри. Таким чином, підтримка після запуску забезпечує не тільки виправлення невеликих помилок та поліпшення ігрового процесу, але й активне розширення ігрового світу та підтримання залученості та задоволеності гравців, що є критично важливим для тривалої популярності та комерційного успіху відеогри.[12]

Висновки до розділу 2

В процесі розробки концепції та моделей для пригодницького RPG, графічне моделювання та створення сценаріїв використання стали основою для визначення структури ігрового світу та взаємодій між компонентами системи. Концептуальний аналіз гри допоміг ідентифікувати ключові аспекти ігрового процесу та взаємовідносин персонажів, які відіграють критичну роль у створенні занурюючого досвіду для гравців. Окрім того, чітке розуміння етапів розробки забезпечує можливість ефективно управляти ресурсами та часом, гарантуючи високу якість кінцевого продукту.

Створені діаграми діяльності та варіантів використання відображають взаємозв’язки між ігровими об'єктами, що сприяє глибшому розумінню та поліпшенню інтерфейсів і механізмів управління. Такий підхід не тільки оптимізує розробку, але й забезпечує багатогранність і повторюваність ігрового процесу, роблячи гру більш привабливою для користувачів на різних етапах їхньої взаємодії з ігровим світом.

3 АРХІТЕКТУРА ТА КОМПОНЕНТИ ГРИ

3.1: Компоненти гри

Використання Асетів

Для створення навколишнього середовища у грі було використано асет "Slavic Medieval Village - Town Interior and Exterior Pack" з Unity Asset Store. Даний асет включає в себе багатий набір 3D об'єктів, що дозволяє відтворити деталізоване і реалістичне середовище. Використання цього асету значно спростило процес розробки, забезпечуючи високу якість графіки та відповідність середньовічному стилю.

Окрему увагу було приділено створенню неба, яке є важливою частиною загальної атмосфери гри. Для цього було завантажено спеціальний шейдер неба, який підтримує динамічні зміни, такі як перехід від дня до ночі, хмарність та погодні ефекти. Це рішення дозволило створити більш живий та динамічний світ, де зміна часу доби і погоди впливає на геймплей та сприйняття гравцем.

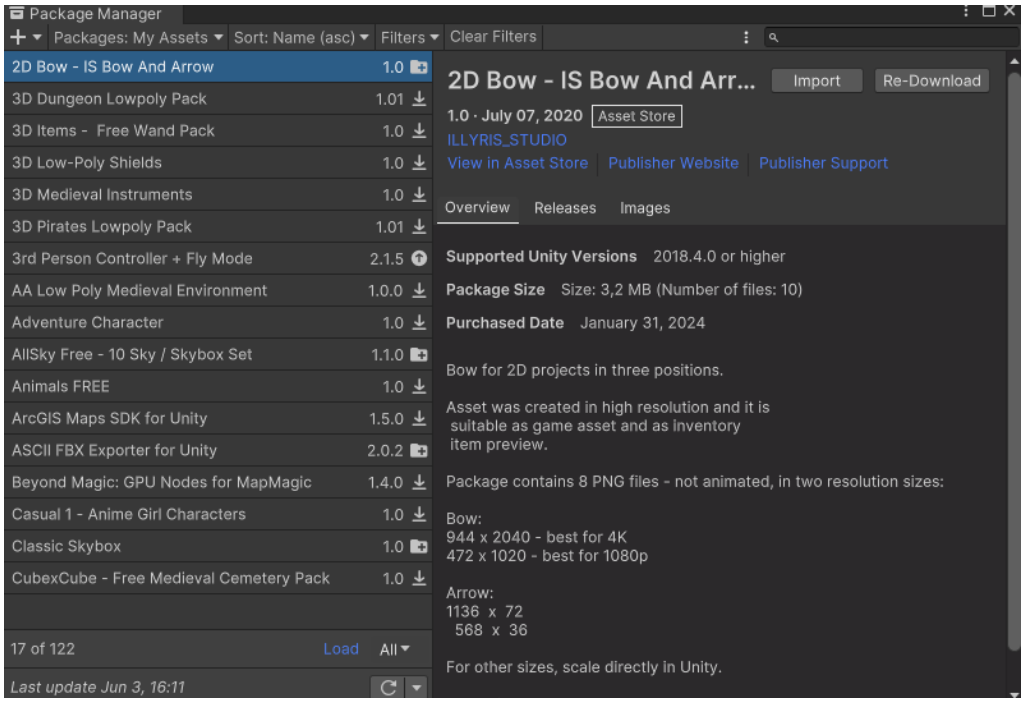

Рисунок 3.1 – Менеджер пакетів Unity

Рендеринг URP

Для реалізації водних поверхонь у грі було використано шейдер води з Universal Render Pipeline (URP). Цей шейдер дозволяє досягти високої якості відображення та реалістичних ефектів води, таких як хвилі, рефракція і віддзеркалення. Завдяки цьому вода у грі виглядає динамічною та природною, що значно підвищує рівень занурення гравця у гру. Вся гра використовує Universal Render Pipeline для рендерингу. Використання URP забезпечує високу продуктивність та якість зображення, підтримуючи сучасні графічні ефекти та оптимізовану обробку кадрів. Це дозволило реалізувати складні сцени з великою кількістю об'єктів та ефектів без значних втрат продуктивності, що є критично важливим для забезпечення плавного геймплею. URP було обрано в порівнянні з іншими рендерами через декілька ключових причин. По-перше, URP оптимізований для високої продуктивності, що дозволяє зменшити навантаження на апаратне забезпечення і підтримувати стабільний FPS навіть у складних сценах. Це робить його більш привабливим у порівнянні з HDRP, який орієнтований на високоякісні графічні ефекти для потужних систем. По-друге, URP надає широкий спектр налаштувань, що дозволяє точно контролювати процес рендерингу і адаптувати його під конкретні потреби проекту. Це забезпечує більшу гнучкість порівняно зі стандартним рендером (Built-in Render Pipeline), який може мати обмежені можливості для налаштувань та оптимізації. По-третє, URP забезпечує сумісність з різними шейдерами, включаючи шейдер води, що дозволяє створювати реалістичні ефекти води з хвилями, рефракцією та віддзеркаленням. Це робить його більш привабливим у порівнянні з іншими рендерами, які можуть мати обмежену підтримку сучасних графічних ефектів. По-четверте, URP підтримує сучасні графічні ефекти, такі як тіні, відблиски, рефлекси та інші, що дозволяє створювати більш реалістичне та візуально привабливе середовище. На відміну від стандартного рендеру, URP постійно оновлюється з додаванням нових функцій, що робить його більш перспективним

вибором для сучасних ігор. По-п'яте, URP підтримує різні платформи, включаючи мобільні пристрої, консолі та ПК, що робить його універсальним рішенням для багатоплатформених проектів. Це важлива перевага над HDRP, який вимагає більш потужного обладнання і краще підходить для високоякісних візуалізацій на ПК та консолях нового покоління. Враховуючи ці переваги, URP було обрано для рендерингу у грі, що дозволило досягти оптимального балансу між якістю графіки та продуктивністю, забезпечуючи захоплюючий ігровий досвід у середньовічному світі.

Рисунок 3.2 – Вигляд води в грі

Аніматор для персонажу

Було створено аніматор для головного персонажа, що включає анімації руху, бою, взаємодії з об'єктами та інші дії. Створення аніматора для персонажа дозволило досягти плавних та реалістичних анімацій, що підвищує загальну якість гри. Використання аніматора було необхідним для забезпечення високого рівня деталізації та динаміки рухів персонажа. Це дозволило створити більш правдоподібний ігровий досвід, де кожен рух виглядає природно і відповідно до контексту ситуації у грі.

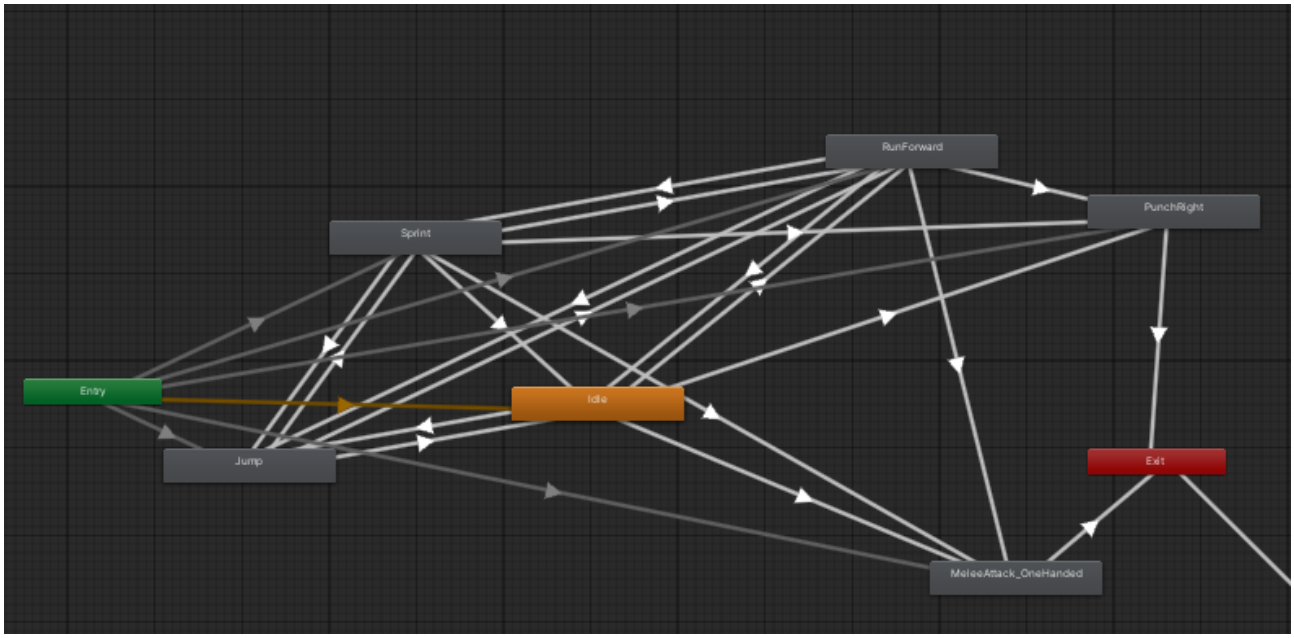

Рисунок 3.3 – Структура компоненту Animator

На діаграмі показано структуру аніматора для головного персонажа гри. Аніматор включає різні стани та переходи між ними, що забезпечують плавні та реалістичні анімації руху та дій персонажа.

Було створено аніматор для головного персонажа, який включає стани для різних типів рухів та дій, таких як ходьба, біг, стрибки, бойові атаки та взаємодія з об'єктами. Початковий стан (Entry) з'єднується зі станом спокою (Idle), що є основним станом персонажа, коли він не рухається.

Створено стани для бігу вперед (RunForward), спринту (Sprint) та стрибка (Jump), які дозволяють персонажу плавно переходити між різними типами руху. Для бойових дій додано стани для удару правою рукою (PunchRight) та одноразової атаки зброєю (MeleeAttack_OneHanded). Ці стани з'єднані між собою через систему переходів, що забезпечують плавні переходи між анімаціями залежно від дій гравця.

Виконано ретельне налаштування переходів між станами, включаючи умови та параметри, що контролюють, коли і як відбуваються ці переходи. Це дозволяє персонажу природно перемикатися між станами, забезпечуючи реалістичність та динамічність його дій.

Завдяки такій структурі аніматора, персонаж може виконувати різноманітні дії з високим рівнем деталізації та плавності, що значно підвищує якість геймплею та сприяє більш глибокому зануренню гравця у гру.

Інструменти ландшафту

Для створення ландшафту гри використовувалися різні інструменти ландшафту в Unity. Це дозволяє моделювати детальні та різноманітні поверхні, такі як гори, рівнини, річки та озера.

Для додання реалістичності та деталізації ландшафту використовувалися різні кисті. З їх допомогою було створено текстури ґрунту, рослинності та інших природних елементів. Це надає кожній місцевості унікальний вигляд і відчуття, підкреслюючи середньовічну атмосферу гри.

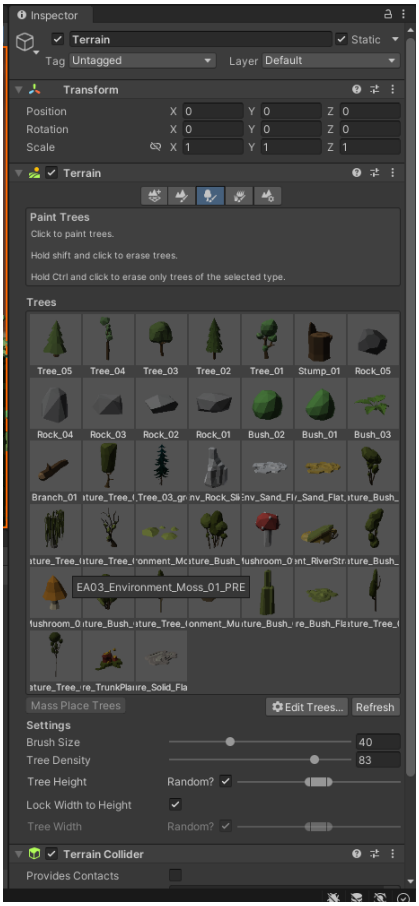

Рисунок 3.4 – Пензлі для створення дерев на Терейні

Для створення ландшафту гри було використано різні інструменти ландшафту в Unity. Використання цих інструментів дозволило моделювати детальні та різноманітні поверхні, такі як гори, рівнини, річки та озера, що значно підвищило рівень реалізму ігрового середовища. Розробка ландшафту включала кілька ключових етапів.

Спочатку було застосовано інструменти для зміни висоти поверхні, такі як Raise і Lower Height, що дозволило створювати різні рельєфні форми, від гірських хребтів до глибоких долин. Це додало ландшафту гри реалістичності та різноманітності, що підвищило загальну якість візуального представлення середовища.

Після створення базового рельєфу було виконано текстурування поверхні за допомогою кистей. Використання Paint Texture дозволило додати реалістичні текстури ґрунту, що надало поверхням природного вигляду. Для кожної ділянки ландшафту підбиралися відповідні текстури, що відображали різні типи ґрунту, трави та інші природні елементи.

Додано також кисті для створення дерев та рослинності. З їх допомогою було виконано посадку дерев, кущів та іншої рослинності, що додало ландшафту гри живості та природності. В процесі розробки використовувалися різні типи кистей, які дозволяли точно розміщувати об'єкти та створювати детальні і реалістичні сцени.

Було також застосовано інструменти для сглажування (Smooth), що дозволило вирівняти різкі переходи між різними ділянками рельєфу, створюючи більш природні та гармонійні ландшафти. Цей етап був важливим для забезпечення плавних переходів і реалістичності загальної картини.

Завдяки використанню цих інструментів і технік було створено детальний та реалістичний ландшафт, який підкреслює середньовічну атмосферу гри. Це дозволило гравцям повністю зануритися в ігровий світ, забезпечуючи високий рівень залучення та задоволення від гри.

3.2 Архітектура застосунку

Процес створення 3D RPG гри у стилі low poly є складним і багатогранним, вимагаючи від розробників уваги до деталей та організованого підходу. Один з ключових аспектів цього процесу - це розробка модульної архітектури, що дозволяє розділити систему на окремі, легко керовані частини. Такий підхід не лише спрощує розробку, але й робить гру більш зрозумілою та гнучкою для майбутніх оновлень. Використання модульної структури допомагає уникнути хаосу в коді та забезпечує можливість паралельної роботи над різними аспектами гри різними членами команди. Кожен з модулів має чітко визначені функції, що дозволяє розробникам зосередитися на конкретних елементах гри, таких як управління грою, інтерфейсом, квестами, інвентарем, діалогами, бойовою системою та системою збережень.

Кожен модуль включає набір класів, які взаємодіють між собою та з іншими модулями. Для кращого розуміння і управління цією структурою були створені діаграми класів. Ці діаграми наочно демонструють залежності між класами та їхні основні функції, що є надзвичайно корисним при проектуванні та розробці. Наприклад, модуль управління грою та інтерфейсом включає класи GameManager, Tooltip, DayCycleManager, InputManager, UiOpener, PauseMenuScript та MainMenu. Для управління квестами розроблені класи QuestManager, QuestData та QuestUI. Модулі інвентаря та предметів містять класи Inventory, Inventory_Item, Inventory_Window, InventoryDragHandler, InventoryItemSlot, Item та ItemTaker. Діалоги та взаємодія з NPC представлені класами CharacterDialogue, DialogueData, DialogueManager та DialogueUI. Бойова система включає класи Combat_1, DamageDealer та HP_Trigger, а система збережень – SaveSystem, SaveLoadManager та GameData. Нарешті, для модуля камери та руху гравця були створені класи PlayerCamera, Ground_Check та Control.

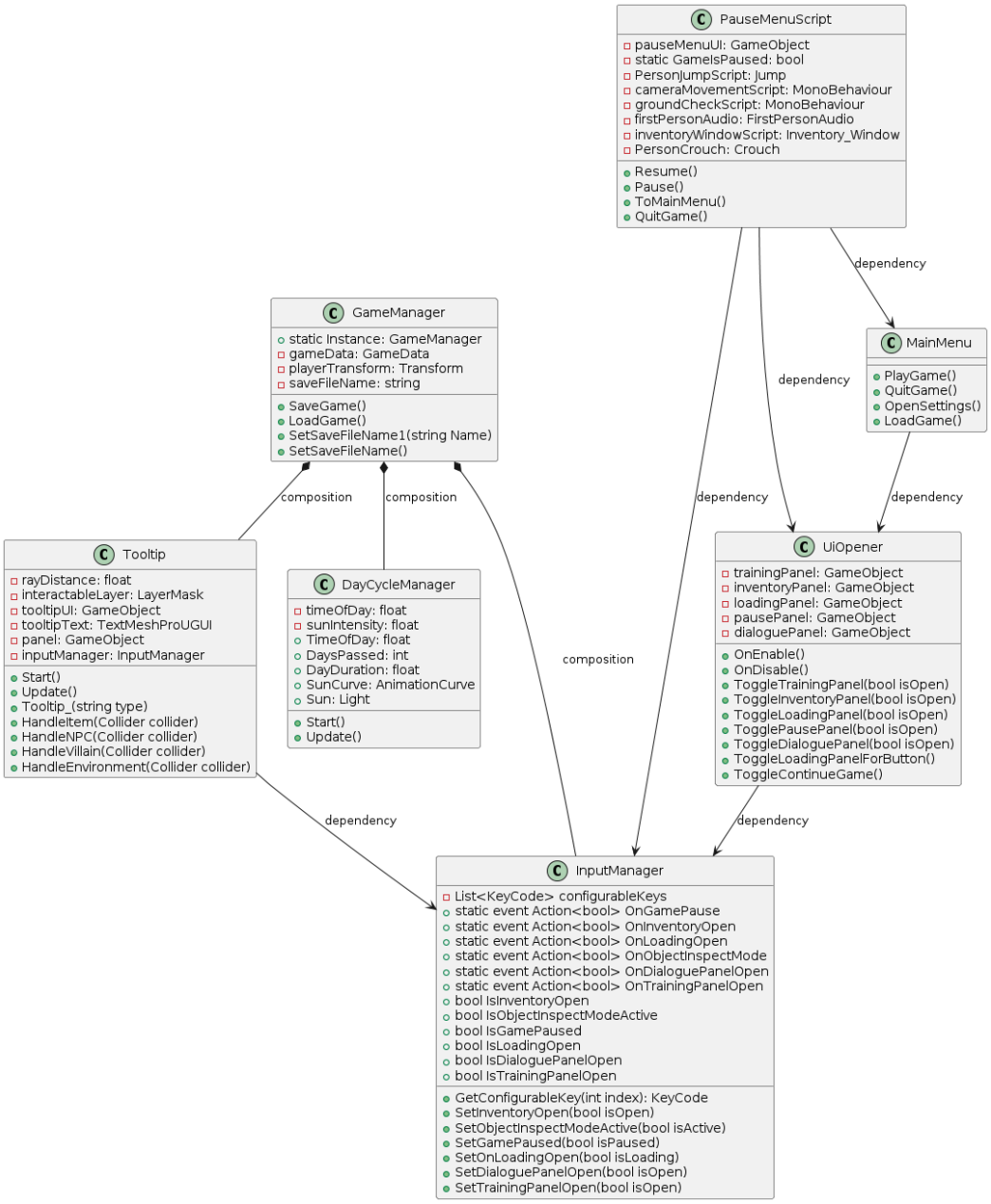

Рисунок 3.5 – Діаграма класів управління грою та інтерфейсом

Діаграма показує основні класи, що відповідають за управління грою та користувацьким інтерфейсом. GameManager є центральним класом, який керує станом гри, включаючи цикл дня та ночі (керується DayCycleManager), і взаємодіє з підказками (через Tooltip). InputManager обробляє ввідні дані від користувача і має залежності від кількох класів для передачі цих даних. UiOpener керує відкриттям та закриттям різних панелей інтерфейсу, таких як панелі інвентаря, паузи та діалогів.

PauseMenuScript управляє меню паузи і має залежності від InputManager, UiOpener та MainMenu для обробки ввідних даних і взаємодії з іншими частинами інтерфейсу. MainMenu управляє головним меню і також має залежність від UiOpener. Взаємозв'язки між класами допомагають забезпечити узгоджену і інтерактивну взаємодію користувача з грою, дозволяючи управляти станом гри, відображати підказки та реагувати на дії користувача.

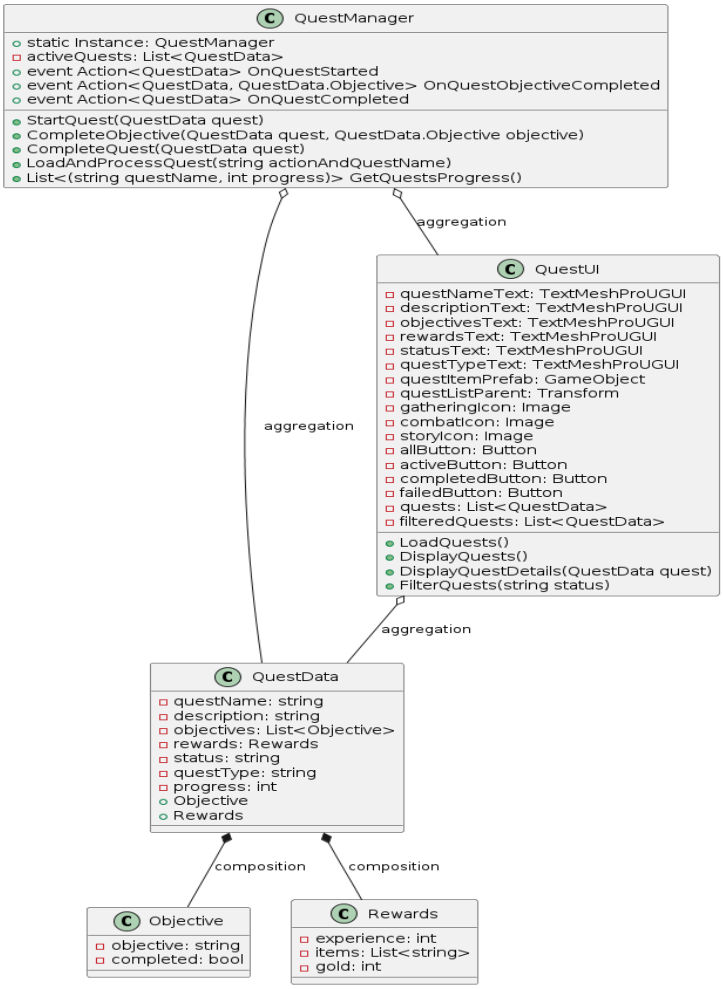

Рисунок 3.6 – Діаграма класів управління квестами

Діаграма показує класи, які керують системою квестів у грі. QuestManager є основним класом, який відповідає за управління активними квестами, початок та завершення квестів, а також за відстеження прогресу. QuestManager має агрегаційний зв'язок з QuestData, що зберігає дані про квести, включаючи назву, опис, цілі та нагороди. QuestData також містить композиційні зв'язки з класами

Objective та Rewards, які описують окремі цілі квесту та нагороди за їх виконання.

QuestUI відповідає за відображення інформації про квести на екрані, включаючи назву квесту, опис, цілі, нагороди та статус. Він має агрегаційний зв'язок з QuestData для отримання даних про квести, а також з QuestManager для взаємодії з системою управління квестами. Ця діаграма показує, як різні компоненти системи квестів взаємодіють між собою для забезпечення гравцю інформації про квести та їхній прогрес.

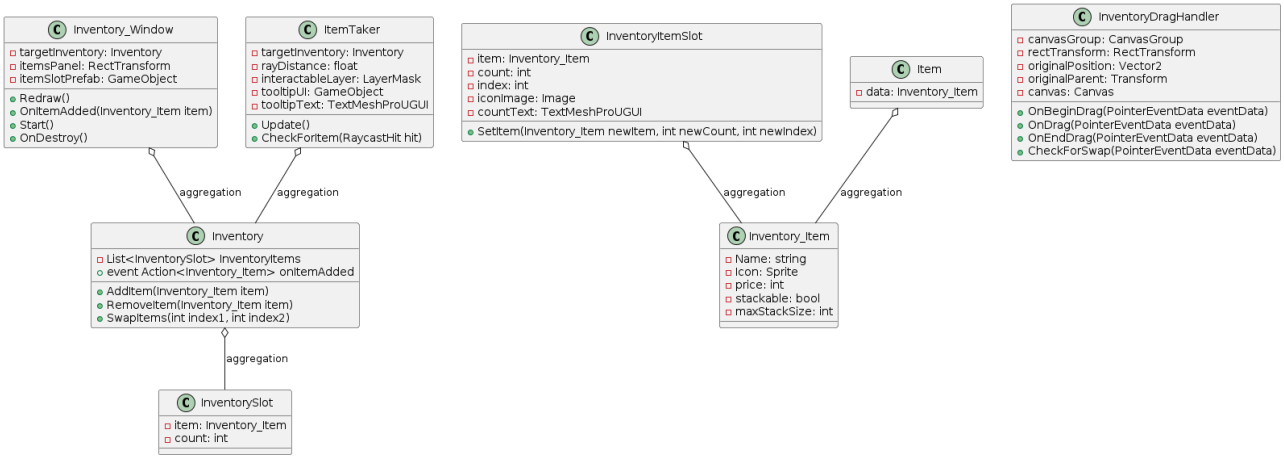

Рисунок 3.7 – Діаграма класів інвентаря та предметів

Діаграма показує класи, які керують системою інвентаря та предметів у грі. Inventory керує списком предметів гравця, додаючи, видаляючи та змінюючи їх. Inventory має агрегаційний зв'язок з InventorySlot, який представляє слот у інвентарі та містить інформацію про предмет та його кількість. Inventory_Item зберігає дані про предмети інвентаря, такі як назва, іконка, ціна, можливість складання в стек та максимальний розмір стека.

Inventory_Window відповідає за відображення інвентаря на екрані, маючи агрегаційний зв'язок з Inventory для взаємодії з інвентарем та відображення предметів. InventoryDragHandler управляє перетягуванням предметів у інвентарі, а InventoryItemSlot представляє слот предмета у інвентарі, маючи агрегаційний зв'язок з Inventory_Item. ItemTaker відстежує натискання клавіші для збору

предметів та має агрегаційний зв'язок з Inventory для додавання предметів до інвентаря.

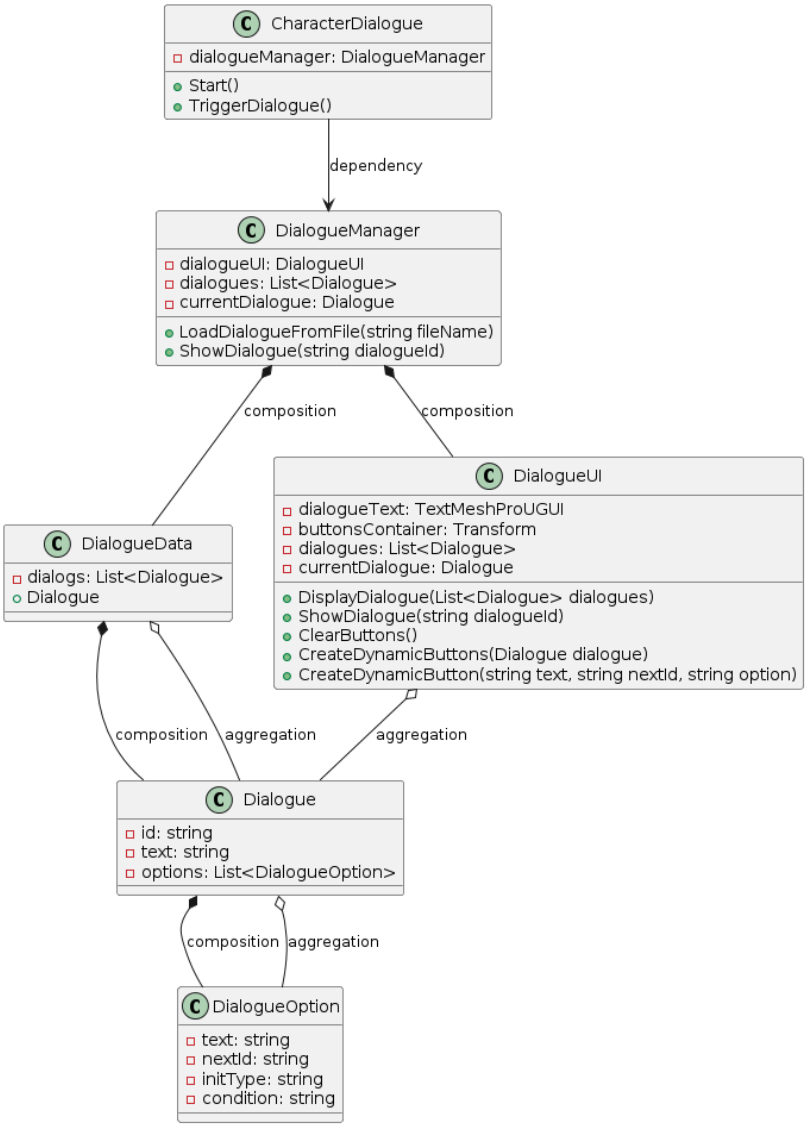

Рисунок 3.8 – Діаграма класів діалогів та взаємодії з NPC

Діаграма показує класи, які керують системою діалогів у грі. CharacterDialogue відповідає за ініціацію діалогів з персонажами та має залежність від DialogueManager для завантаження та відображення діалогів. DialogueManager управляє діалогами, завантажуючи їх з файлів та відображаючи за допомогою DialogueUI. DialogueManager має композиційний зв'язок з DialogueData, яка зберігає всі дані діалогів у грі.

DialogueData містить діалоги, кожен з яких складається з тексту та опцій (варіантів відповіді). Dialogue має композиційний зв'язок з DialogueOption, який представляє опцію в діалозі, включаючи текст, наступний ідентифікатор діалогу, тип ініціації та умову. DialogueUI відповідає за відображення діалогів на екрані, маючи агрегаційний зв'язок з Dialogue для взаємодії з діалогами та відображення їхніх деталей.

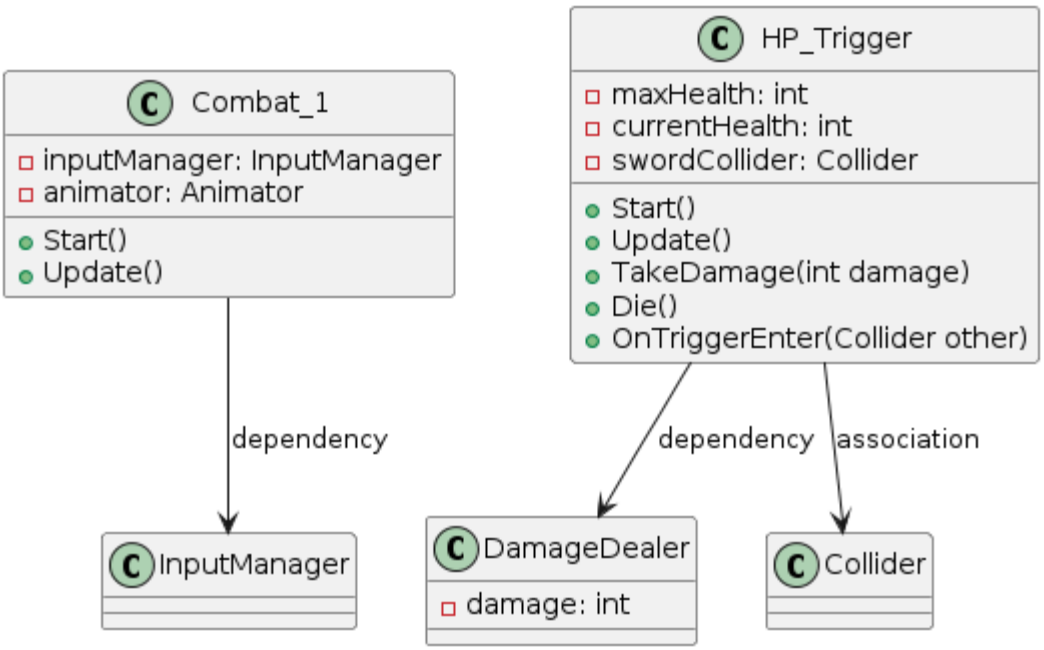

Рисунок 3.9 – Діаграма класів бойової системи

Діаграма показує класи, які керують бойовими діями у грі. Combat_1 управляє бойовими діями гравця та має залежність від InputManager для отримання ввідних даних. Combat_1 використовує Animator для виклику анімацій бойових дій. DamageDealer визначає кількість урону, що наноситься, і взаємодіє з іншими компонентами для обробки урону.

HP_Trigger відстежує здоров'я об'єкта та обробляє урон. Він має залежність від DamageDealer для отримання урону і використовує Collider для перевірки зіткнень та нанесення урону. Ця діаграма показує, як компоненти бойової системи взаємодіють між собою для забезпечення бойових механік у грі.

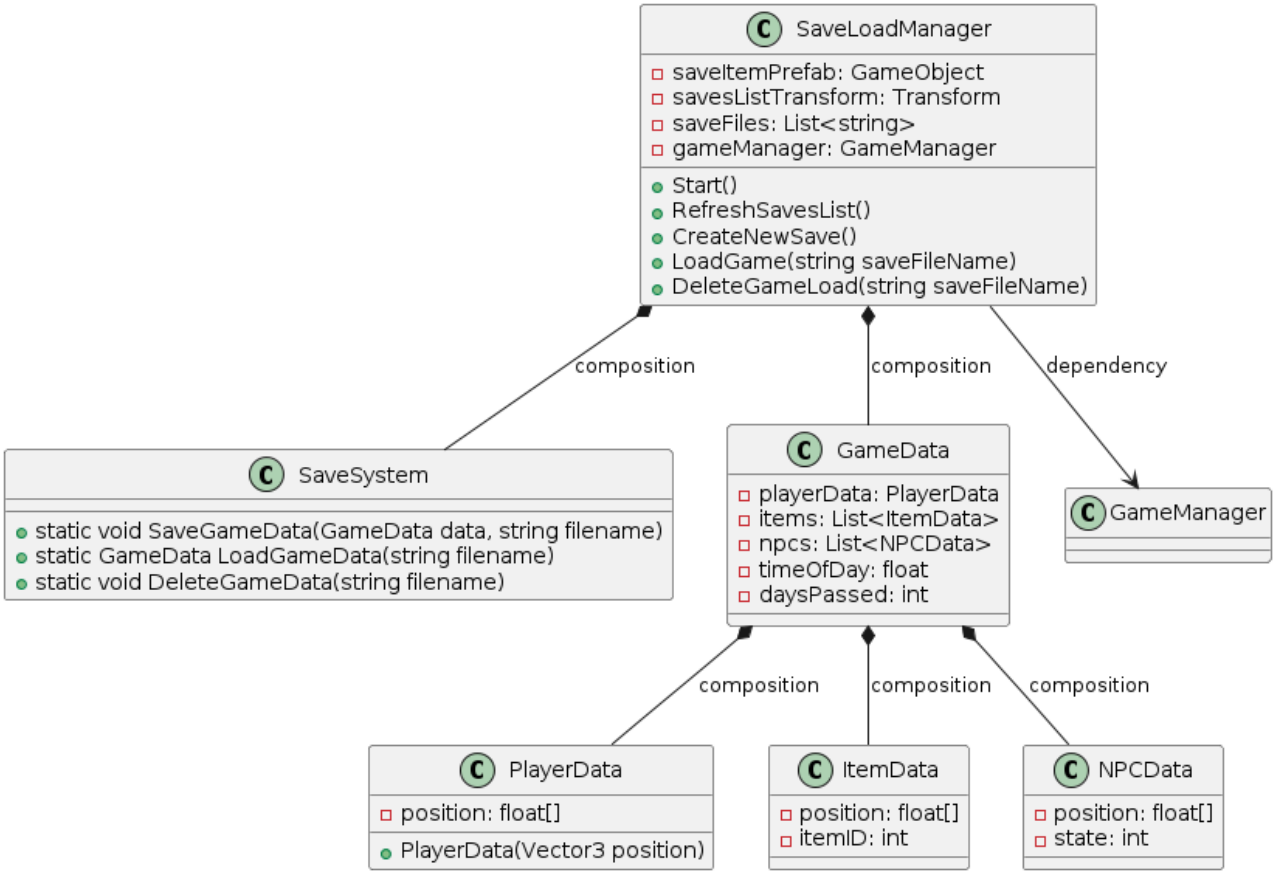

Рисунок 3.10 – Діаграма класів системи збережень

Діаграма показує класи, які керують системою збережень у грі. SaveSystem є статичним класом для збереження, завантаження та видалення даних гри. Він взаємодіє з даними гри через GameData. SaveLoadManager управляє збереженнями та має композиційний зв'язок з SaveSystem для використання його функцій. Він також має композиційний зв'язок з GameData для збереження та завантаження даних гри.

GameData є структурою даних для збереження інформації про стан гри, включаючи дані про гравця (PlayerData), предмети (ItemData) та NPC (NPCData). PlayerData, ItemData та NPCData мають композиційні зв'язки з GameData, оскільки вони є частинами даних гри і не можуть існувати незалежно. Ця діаграма показує, як система збережень взаємодіє з іншими компонентами гри для збереження та відновлення стану гри.

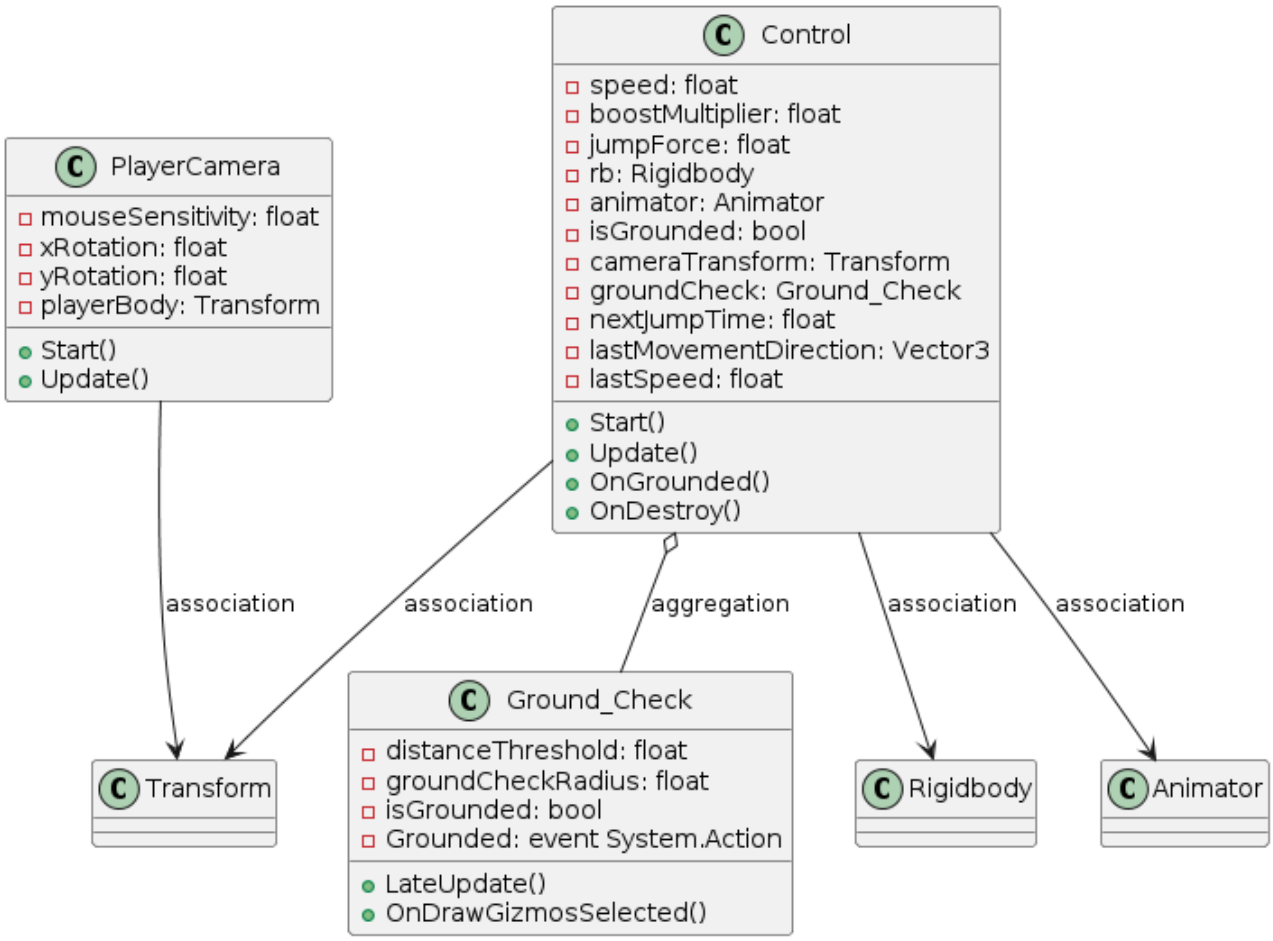

Рисунок 3.11 – Діаграма класів камери та руху гравця

Діаграма показує класи, які керують камерою та рухом гравця у грі. PlayerCamera управляє рухом камери гравця на основі ввідних даних від миші, маючи асоціаційний зв'язок з Transform для обертання тіла гравця. Ground_Check відстежує торкання землі, викликаючи подію Grounded при торканні землі, і відображає в редакторі, чи знаходиться об'єкт на землі.

Control управляє рухом гравця, включаючи біг, прискорення та стрибки. Він має асоціаційні зв'язки з Rigidbody та Animator для фізики та анімацій відповідно, і агрегаційний зв'язок з Ground_Check для визначення стану на землі. Control також використовує Transform для обертання камери та тіла гравця. Ця діаграма показує, як компоненти для керування камерою та рухом гравця взаємодіють між собою для забезпечення плавного та інтерактивного геймплею.

Висновки до розділу 3

Створення 3D RPG гри у стилі low poly передбачає використання структурованого підходу до розробки, де кожен компонент відповідає за певний аспект ігрового процесу. Розподіл системи на сім основних модулів забезпечує чітку організацію коду та полегшує підтримку проєкту. Впровадження різноманітних механік, таких як управління грою та інтерфейсом, обробка ввідних даних, система квестів, інвентар, діалоги, бойова система, система збережень та управління камерою, створює насичений ігровий досвід.

Для створення навколишнього середовища у грі використовували асети з Unity Asset Store, які включають 3D об'єкти для деталізованого середньовічного стилю. Спеціальний шейдер неба підтримує динамічні зміни часу доби та погодні ефекти, створюючи живий і динамічний світ. Гра використовує Universal Render Pipeline (URP) для високої продуктивності та якості зображення. Аніматор для головного персонажа забезпечує плавні анімації руху, бою та взаємодії з об'єктами. Інструменти ландшафту в Unity використовувалися для моделювання деталей ландшафту, таких як гори, рівнини та річки, що підкреслює середньовічну атмосферу гри.

Такий комплексний підхід дозволяє створити високоякісний продукт, який задовольняє очікування сучасних гравців та забезпечує тривалий інтерес до гри.

4 РОЗРОБКА ІГРОВОГО ЗАСТОСУНКУ

4.1. Управління грою та інтерфейс

GameManager є центральним класом для управління станом гри та забезпечення узгодженості всіх її компонентів. Цей клас реалізує патерн синглтон, що гарантує наявність лише одного екземпляра GameManager у грі. Це досягається за допомогою методу Awake(), який перевіряє, чи вже існує екземпляр Instance. Якщо екземпляра немає, він встановлюється на поточний об'єкт, який не знищується при завантаженні нових сцен.

Один з основних обов'язків GameManager — збереження та завантаження даних гри. Метод SaveGame() зберігає поточний стан гри, включаючи час доби, кількість пройдених днів та позицію гравця. Ці дані зберігаються у файл за допомогою системи збереження, забезпечуючи можливість відновлення гри з того ж місця, де вона була зупинена. GameManager також генерує унікальні імена файлів для збережень, що дозволяє мати декілька різних збережень гри.

Метод LoadGame() відновлює стан гри з файлу збереження. Він завантажує дані, включаючи час доби, кількість пройдених днів та позицію гравця, і застосовує їх до відповідних компонентів гри. Це дозволяє гравцю продовжувати гру з того ж місця, де вона була зупинена. Якщо файл збереження не знайдено, GameManager ініціалізує нову гру, забезпечуючи плавний старт ігрового процесу.

GameManager також взаємодіє з іншими основними компонентами гри, такими як DayCycleManager для управління циклом дня та ночі, і playerTransform для відстеження позиції гравця. Цей клас служить центральним вузлом, що забезпечує координацію та інтеграцію всіх ключових аспектів гри, роблячи її цілісною та узгодженою.

Приклад коду:

```
void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}

public void SaveGame()
{
    if (gameData == null)
    {
        gameData = new GameData();
    }
    SetSaveFileName();
    if (dayCycleManager != null)
    {
        gameData.timeOfDay = dayCycleManager.TimeOfDay;
        gameData.daysPassed = dayCycleManager.DaysPassed;
    }
    if (playerTransform != null)
    {
        gameData.playerData = new PlayerData(playerTransform.position);
        SaveSystem.SaveGameData(gameData, saveFileName);
    }
    else
    {
        Debug.LogError("Player Transform not assigned.");
    }
}
```

Метод Awake() реалізує патерн синглтон, гарантує, що існує лише один екземпляр GameManager у грі. Якщо екземпляр не існує, він створюється і зберігається між сценами. Якщо екземпляр вже існує, новий об'єкт знищується, щоб уникнути дублювання. Метод SaveGame() зберігає поточний стан гри, включаючи час доби, кількість пройдених днів та позицію гравця, у файл за допомогою SaveSystem.

4.2. Обробка ввідних даних

InputManager відповідає за обробку ввідних даних від користувача, що є одним з ключових елементів будь-якої інтерактивної гри. Він забезпечує реакцію на дії гравця, такі як рух, взаємодія з об'єктами, відкриття інвентаря, бойові дії та інші важливі аспекти геймплею. InputManager виступає в ролі посередника між користувачем і грою, переводячи натискання клавіш та рухи миші в конкретні дії в грі.

Однією з важливих функцій InputManager є обробка конфігурованих клавіш. Це означає, що гравці можуть налаштовувати управління під себе, змінюючи призначення клавіш для різних дій. InputManager зберігає ці налаштування і використовує їх для визначення того, які дії викликати при натисканні відповідних клавіш. Це підвищує зручність та індивідуалізацію ігрового процесу для кожного гравця.

Крім обробки базових дій, InputManager також керує складнішими станами гри, такими як відкриття меню паузи, режиму огляду об'єктів, діалогових вікон і тренувальних панелей. Він використовує події для сповіщення інших компонентів гри про зміни станів. Наприклад, при натисканні клавіші для відкриття інвентаря InputManager генерує подію, яку підписуються інші компоненти, такі як UiOpener, для відкриття відповідної панелі інтерфейсу.

Взаємодія InputManager з іншими компонентами гри є важливим аспектом його функціонування. Він інтегрується з GameManager для забезпечення коректної обробки ввідних даних під час збереження та завантаження гри, а також з різними UI-компонентами для управління інтерфейсом користувача. Завдяки InputManager, гра стає динамічною та інтерактивною, забезпечуючи гравцю можливість впливати на ігровий процес у реальному часі.

Приклад коду:

```
public KeyCode GetConfigurableKey(int index)
{
    return configurableKeys[index];
```

```
}

void Update()
{
    if (Input.GetKeyDown(GetConfigurableKey(0)))
    {
        // Handle action for the first configurable key
    }
}

void HandleMovement()
{
    float moveX = Input.GetAxis("Horizontal");
    float moveZ = Input.GetAxis("Vertical");

    Vector3 move = transform.right * moveX + transform.forward * moveZ;
    controller.Move(move * speed * Time.deltaTime);
}
```

Метод GetConfigurableKey() повертає конфігуровану клавішу за вказаним індексом, що дозволяє гравцю налаштовувати управління. Метод Update() викликається кожен кадр і обробляє ввідні дані від користувача для виконання дій. Метод HandleMovement() відповідає за рух гравця на основі ввідних даних від клавіш WASD.

4.3. Управління квестами

QuestManager є ключовим компонентом для управління системою квестів у грі. Він відповідає за створення, відстеження та завершення квестів, забезпечуючи інтерактивний ігровий процес, який мотивує гравця виконувати завдання і просуватися по сюжету. QuestManager виступає як центральний вузол для всіх дій, пов'язаних з квестами, координуючи їх з іншими елементами гри.

Основна функція QuestManager полягає у відстеженні активних квестів. Він зберігає список поточних завдань гравця, кожне з яких має свій унікальний ідентифікатор, опис, цілі та нагороди. QuestManager забезпечує управління статусами квестів, такими як "активний", "завершений" або "провалений", і оновлює ці статуси в міру виконання або провалу завдань.

Окрім управління статусами квестів, QuestManager також обробляє події, пов'язані з квестами. Наприклад, при виконанні певної цілі квесту, він генерує події, які можуть впливати на інші частини гри, такі як оновлення інтерфейсу користувача або зміна стану персонажів. Ці події дозволяють іншим компонентам гри, таким як QuestUI, відображати оновлену інформацію про прогрес квестів, а також виконувати додаткові дії, пов'язані з завершенням квестів.

Взаємодія QuestManager з іншими компонентами гри є важливим аспектом його роботи. Він інтегрується з QuestData для збереження та завантаження інформації про квести, а також з QuestUI для відображення поточних завдань гравця. Завдяки QuestManager, система квестів у грі стає інтерактивною та динамічною, забезпечуючи гравцю можливість брати участь у різноманітних завданнях, які впливають на сюжет ігри та взаємодію з іншими персонажами.

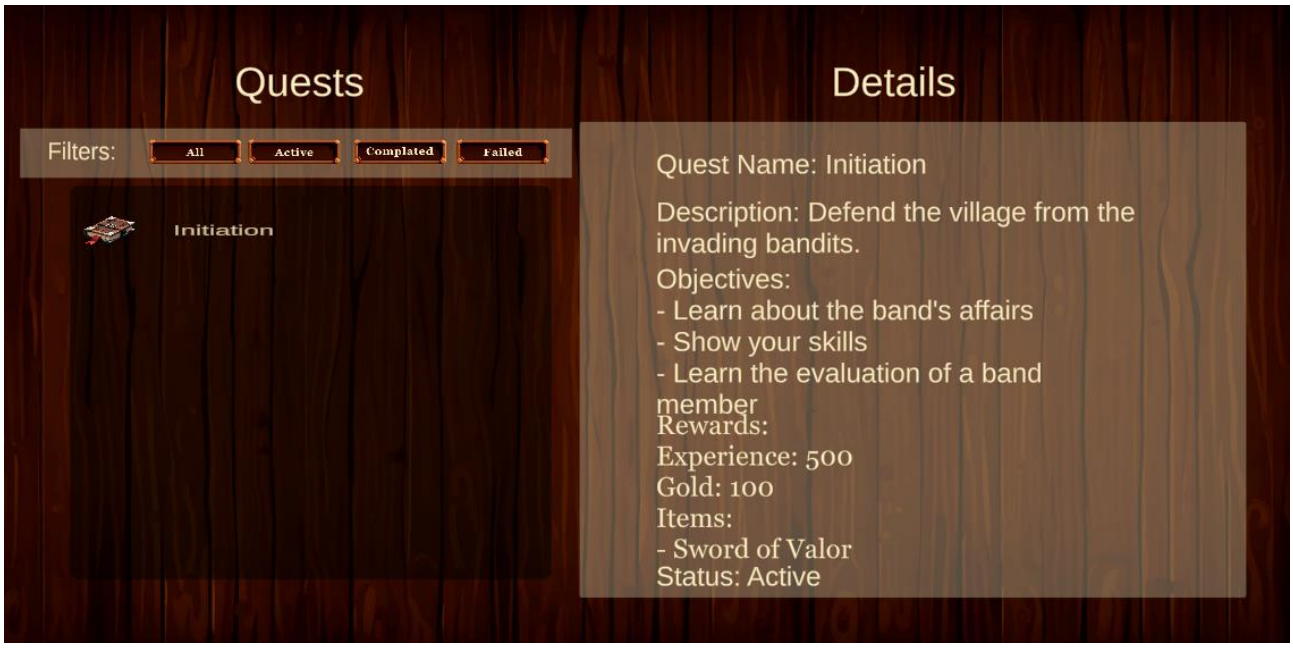

Рисунок 4.1 – Інтерфейс меню квестів

Приклад коду:

```
public void StartQuest(QuestData quest)
{
    activeQuests.Add(quest);
```

```
    OnQuestStarted?.Invoke(quest);
}

public void CompleteQuest(QuestData quest)
{
    quest.isCompleted = true;
    OnQuestCompleted?.Invoke(quest);
}

public List<QuestData> GetActiveQuests()
{
    return activeQuests;
}

public void UpdateQuest(QuestData quest)
{
    if (quest.IsComplete())
    {
        CompleteQuest(quest);
    }
}
```

Метод StartQuest() додає квест до списку активних квестів і викликає подію OnQuestStarted, що дозволяє іншим компонентам гри знати про початок нового квесту. Метод CompleteQuest() відзначає квест як завершений і викликає подію OnQuestCompleted. Метод GetActiveQuests() повертає список поточних активних квестів. Метод UpdateQuest() перевіряє, чи завершено квест, і якщо так, то викликає метод CompleteQuest().

4.4. Інвентар та предмети

Inventory є ключовим компонентом для управління предметами гравця в грі. Він відповідає за додавання, видалення та організацію предметів, забезпечуючи гравцю можливість ефективно керувати своїми ресурсами. Inventory є центральним вузлом для всіх операцій, пов'язаних з предметами, координуючи їх з іншими елементами гри, такими як інтерфейс користувача та механіка гри.

Основна функція Inventory полягає в управлінні списком предметів гравця. Він зберігає інформацію про всі предмети, які гравець зібрав, включаючи їх кількість, стан та властивості. Inventory використовує структуру даних, таку як список або масив, для зберігання предметів, що дозволяє легко додавати нові предмети, видаляти старі та змінювати існуючі.

Крім базових операцій, Inventory також забезпечує функціональність для організації предметів. Це включає можливість сортування предметів за різними критеріями, такими як тип, рідкість або дата отримання. Inventory може також підтримувати функції для об'єднання однакових предметів в стеки, що дозволяє економити місце та спрощувати управління великою кількістю предметів.

Взаємодія Inventory з іншими компонентами гри є важливим аспектом його роботи. Він інтегрується з InventoryUI для відображення предметів на екрані, дозволяючи гравцю бачити і керувати своїм інвентарем у зручному форматі. Inventory також взаємодіє з механіками гри, такими як система бою або крафтинг, забезпечуючи доступ до предметів, необхідних для виконання різних дій. Завдяки Inventory, гравці можуть ефективно керувати своїми ресурсами та взаємодіяти з ігровим світом, що є важливою частиною геймплею в багатьох RPG-іграх.

Приклад коду:

```
public void AddItem(Inventory_Item item)
{
    inventoryItems.Add(item);
    onItemAdded?.Invoke(item);
}

public void RemoveItem(Inventory_Item item)
{
    inventoryItems.Remove(item);
    onItemRemoved?.Invoke(item);
}

public List<Inventory_Item> GetItems()
{
```

```
    return inventoryItems;
}

public bool HasItem(Inventory_Item item)
{
    return inventoryItems.Contains(item);
}
```

Метод AddItem() додає предмет до інвентаря і викликає подію onItemAdded, що дозволяє іншим компонентам реагувати на додавання нового предмета. Метод RemoveItem() видаляє предмет з інвентаря і викликає подію onItemRemoved. Метод GetItems() повертає список усіх предметів в інвентарі. Метод HasItem() перевіряє, чи є певний предмет в інвентарі.

Рисунок 4.2 – Інтерфейс інвентарю

4.5. Діалоги та взаємодія з NPC

DialogueManager є ключовим компонентом для управління системою діалогів у грі. Він відповідає за завантаження, збереження та відображення діалогів, забезпечуючи взаємодію гравця з NPC та іншими персонажами у грі.

DialogueManager відіграє важливу роль у розвитку сюжету та наданні гравцю можливості приймати рішення, які впливають на хід гри.

Основна функція DialogueManager полягає у завантаженні діалогів з файлів або інших джерел даних. Ці діалоги можуть містити різноманітні опції, відповіді та гілки розмови, що дозволяє створювати складні та інтерактивні взаємодії між гравцем та NPC. DialogueManager зберігає ці діалоги у структурі даних, яка дозволяє легко доступати до них під час гри.

Крім завантаження, DialogueManager також забезпечує відображення діалогів на екрані. Він використовує спеціалізовані UI-компоненти, такі як DialogueUI, для відображення тексту діалогу та опцій вибору. Це дозволяє гравцю читати діалоги, вибирати відповіді та продовжувати розмови. DialogueManager також відповідає за обробку вибору гравця, визначаючи, яка гілка діалогу повинна бути відображена далі, на основі вибраних опцій.

Рисунок 4.3 – Інтерфейс діалогів

Взаємодія DialogueManager з іншими компонентами гри є критично важливою для забезпечення узгодженого та інтерактивного ігрового процесу. Він інтегрується з системою квестів, дозволяючи діалогам впливати на прогрес квестів, а також з системою збережень, забезпечуючи збереження стану діалогів.

Завдяки DialogueManager, гравці можуть взаємодіяти з персонажами гри у насичений та змістовний спосіб, впливаючи на розвиток сюжету та отримуючи важливу інформацію, яка допомагає просуватися у грі.

Приклад коду:

```
public void ShowDialogue(string dialogueId)
{
    currentDialogue = dialogues.Find(dialogue => dialogue.id == dialogueId);
    dialogueUI.DisplayDialogue(currentDialogue);
}

public void LoadDialogueFromFile(string fileName)
{
    string json = File.ReadAllText(Path.Combine(Application.streamingAssetsPath,
fileName));
    dialogues = JsonUtility.FromJson<DialogueData>(json).dialogs;
}
```

Метод ShowDialogue() знаходить діалог за його ID та відображає його за допомогою DialogueUI. Метод LoadDialogueFromFile() завантажує діалоги з файлу у форматі JSON та десеріалізує їх у список діалогів.

4.6. Управління циклом дня та ночі

DayCycleManager відповідає за управління циклом дня та ночі у грі, створюючи динамічне і реалістичне середовище. Цей компонент змінює положення та інтенсивність сонця в залежності від часу доби, що впливає на освітлення та загальну атмосферу гри. DayCycleManager відіграє важливу роль у створенні візуального та ігрового досвіду, який залежить від часу доби.

Основна функція DayCycleManager полягає в оновленні часу доби. Він використовує змінну TimeOfDay, яка збільшується кожен кадр на основі минулого часу (Time.deltaTime) та тривалості дня (DayDuration). Коли TimeOfDay досягає або перевищує 1, він скидається до 0, а кількість пройдених днів (DaysPassed) збільшується на 1. Це дозволяє реалізувати безперервний цикл дня та ночі у грі.

Крім оновлення часу, DayCycleManager також відповідає за зміну положення сонця. Він використовує Quaternion.Euler для обертання сонця навколо осі на основі TimeOfDay, що створює ефект руху сонця по небу. Це обертання впливає на освітлення сцени, змінюючи його напрямок і створюючи реалістичні тіні в залежності від часу доби.

Іншим важливим аспектом DayCycleManager є управління інтенсивністю сонця. Він використовує анімаційну криву (SunCurve), щоб змінювати інтенсивність світла сонця протягом дня. Це дозволяє створювати плавні переходи між різними рівнями освітлення, наприклад, світанок, полудень і захід сонця. Завдяки цьому, гра отримує реалістичний вигляд і відчуття змін часу доби, що збагачує загальний ігровий досвід.

Взаємодія DayCycleManager з іншими компонентами гри є важливою для забезпечення узгодженого середовища. Він може інтегруватися з системою збережень для збереження та відновлення поточного часу доби та кількості пройдених днів. Це дозволяє гравцям повертатися до гри в той самий час доби, в якому вони її залишили. Завдяки DayCycleManager, гра стає більш динамічною та захоплюючою, пропонуючи гравцям досвід, який змінюється в залежності від часу доби.

Приклад коду:

```
void Update()
{
    TimeOfDay += Time.deltaTime / DayDuration;
    if (TimeOfDay >= 1)
    {
        TimeOfDay = 0;
        DaysPassed++;
    }

    Sun.transform.localRotation = Quaternion.Euler(TimeOfDay * 360, 180, 0);
    Sun.intensity = sunIntensity * SunCurve.Evaluate(TimeOfDay);
}

void Start()
{
```

```
    sunIntensity = Sun.intensity;
}
```

Метод Update() оновлює час доби кожен кадр і обертає сонце відповідно до поточного часу доби, а також змінює його інтенсивність. Метод Start() ініціалізує інтенсивність сонця на початку.

4.7. Система збережень

SaveSystem

SaveSystem є ключовим компонентом для управління збереженням та завантаженням даних гри. Він відповідає за збереження стану гри у файл і відновлення цього стану при завантаженні гри. SaveSystem забезпечує безперебійне продовження ігрового процесу, дозволяючи гравцям повертатися до гри з того місця, де вони її залишили.

Основна функція SaveSystem полягає у збереженні даних гри. Він отримує об'єкт даних гри, який містить всю необхідну інформацію про стан гри, таку як позиція гравця, інвентар, прогрес квестів тощо. SaveSystem серіалізує цей об'єкт у формат, який можна записати у файл (наприклад, JSON або бінарний формат), і зберігає його на диск. Це забезпечує можливість зберігати прогрес гри у будь-який момент часу.

Крім збереження, SaveSystem також забезпечує завантаження даних гри. При завантаженні гри SaveSystem читає файл збереження, десеріалізує дані та відновлює об'єкт стану гри. Цей об'єкт передається відповідним компонентам гри, таким як GameManager, Inventory, QuestManager тощо, які оновлюють свій стан на основі завантажених даних. Це дозволяє гравцям продовжувати гру з того місця, де вони її зберегли.

SaveSystem також може забезпечувати видалення даних збереження. Це може бути корисно у випадках, коли гравець хоче почати нову гру з чистого аркуша або видалити непотрібні збереження. Видалення збережень виконується шляхом видалення відповідного файлу з диска.

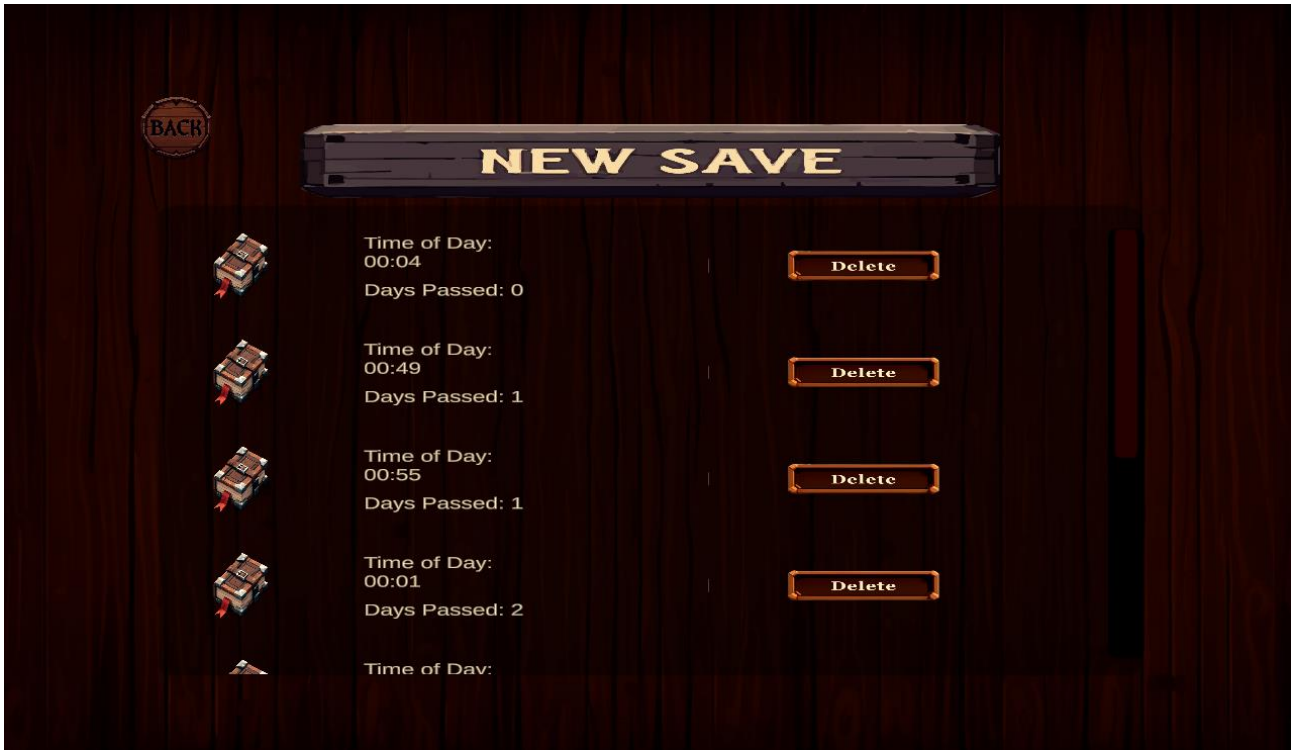

Рисунок 4.4 – Інтерфейс збережень

Взаємодія SaveSystem з іншими компонентами гри є критично важливою для забезпечення цілісності ігрового процесу. Він інтегрується з GameManager для ініціалізації збереження та завантаження гри, а також з іншими компонентами, такими як DayCycleManager і QuestManager, для збереження і відновлення їх стану. Завдяки SaveSystem, гравці можуть безперебійно зберігати і відновлювати свій прогрес у грі, що є важливою частиною користувацького досвіду у будь-якій сучасній грі.

Приклад коду:

```
public static void SaveGameData(GameData data, string filename)
{
    string json = JsonUtility.ToJson(data);
    File.WriteAllText(Path.Combine(Application.persistentDataPath, filename),
json);
}

public static GameData LoadGameData(string filename)
{
    string path = Path.Combine(Application.persistentDataPath, filename);
    if (File.Exists(path))
```

```
    {
        string json = File.ReadAllText(path);
        return JsonUtility.FromJson<GameData>(json);
    }
    else
    {
        Debug.LogWarning("Save file not found");
        return null;
    }
}
```

Метод SaveGameData() серіалізує об'єкт даних гри у формат JSON і записує його у файл. Метод LoadGameData() завантажує дані гри з файлу та десеріалізує їх, повертаючи об'єкт GameData.

4.8. Управління бойовою системою

Combat_1 є ключовим компонентом, що відповідає за управління бойовими діями гравця у грі. Він обробляє ввідні дані від користувача для виконання атак, взаємодіє з анімаціями та системою урону. Combat_1 відіграє важливу роль у забезпеченні динамічного і захоплюючого бою, що є центральним елементом геймплею у багатьох RPG-іграх.

Основна функція Combat_1 полягає в обробці ввідних даних від користувача для виконання бойових дій. Він реагує на натискання клавіш, такі як атака, блокування або використання спеціальних умінь. Combat_1 використовує ці ввідні дані для ініціації відповідних анімацій та дій у грі. Це дозволяє гравцю безпосередньо контролювати свого персонажа під час бою, виконуючи різноманітні комбінації атак та захисту.

Крім обробки ввідних даних, Combat_1 також взаємодіє з системою анімацій. Він використовує аніматор для запуску різних бойових анімацій, таких як удари мечем, блокування щитом або виконання спеціальних прийомів. Це забезпечує візуальну складову бою, роблячи його більш реалістичним і

захоплюючим. Синхронізація анімацій з діями гравця є критично важливою для забезпечення плавного ігрового досвіду.

Рисунок 4.5 – Анімація удару мечем

Іншим важливим аспектом Combat_1 є взаємодія з системою урону. При виконанні атаки, Combat_1 обчислює кількість урону, який наноситься ворогам, і передає ці дані до відповідних компонентів, таких як DamageDealer або HP_Trigger. Це забезпечує реалістичну модель бою, де кожен удар має свої наслідки, знижуючи здоров'я ворогів і впливаючи на їх поведінку.

Взаємодія Combat_1 з іншими компонентами гри є важливою для забезпечення цілісності бойової системи. Він інтегрується з InputManager для обробки ввідних даних, з Animator для управління анімаціями, а також з системою урону для обчислення і застосування урону. Завдяки Combat_1, бойові дії у грі стають інтерактивними та захоплюючими, забезпечуючи гравцю можливість безпосередньо впливати на хід бою і отримувати задоволення від ігрового процесу.

Приклад коду:

```
void Update()
{
    if (Input.GetButtonDown("Punch1"))
    {
        Attack();
    }
}

void Attack()
{
    animator.SetTrigger("Attack");
}
```

Метод Update() обробляє ввідні дані від користувача для виконання атак. Метод Punch1 ініціює анімацію атаки, використовуючи тригер в аніматорі.

4.9. Управління камерою та рухом гравця

PlayerCamera є ключовим компонентом для управління камерою гравця в грі. Він забезпечує огляд та контроль камери від першої або третьої особи, що значно впливає на користувацький досвід та ігровий процес. PlayerCamera відіграє важливу роль у забезпеченні плавного і зручного управління камерою, дозволяючи гравцям оглядати навколишнє середовище та орієнтуватися в грі.

Основна функція PlayerCamera полягає у відстеженні рухів миші і використанні цих даних для обертання камери. Це забезпечує природне і інтуїтивно зрозуміле управління, коли рухи миші переводяться в обертання камери, дозволяючи гравцям оглядати навколишнє середовище у всіх напрямках. Це особливо важливо для ігор від першої особи, де огляд від першої особи є основним способом взаємодії з грою.

Крім відстеження рухів миші, PlayerCamera також може забезпечувати функціональність наближення та віддалення камери. Це дозволяє гравцям змінювати відстань огляду, що може бути корисним для детальнішого розгляду об'єктів або для отримання кращого огляду навколишнього середовища.

PlayerCamera може використовувати різні методи для зміни відстані камери, такі як зміна поля зору або переміщення самої камери.

Іншим важливим аспектом PlayerCamera є плавність руху камери. Вона забезпечується за допомогою різних алгоритмів інтерполяції та згладжування, які роблять рух камери плавним і природним. Це значно покращує користувацький досвід, зменшуючи ривки і стрибки камери, що може викликати дискомфорт у гравців.

Взаємодія PlayerCamera з іншими компонентами гри є критично важливою для забезпечення цілісного ігрового процесу. Вона інтегрується з Control для відстеження рухів гравця та зміни орієнтації камери відповідно до рухів гравця. PlayerCamera також може взаємодіяти з іншими компонентами, такими як системи UI та інші елементи геймплею, щоб забезпечити синхронізацію огляду та взаємодії з інтерфейсом. Завдяки PlayerCamera, гравці можуть насолоджуватися зручним і захоплюючим управлінням камерою, що значно підвищує загальний ігровий досвід.

Приклад коду:

```
void Update()
{
    float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
    float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

    xRotation -= mouseY;
    xRotation = Mathf.Clamp(xRotation, -90f, 90f);

    transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
    playerBody.Rotate(Vector3.up * mouseX);
}

void Start()
{
    Cursor.lockState = CursorLockMode.Locked;
}
```

Пояснення: Метод Update() відстежує рухи миші та використовує їх для обертання камери. Метод Start() блокує курсор миші в центрі екрану для забезпечення зручного управління камерою.

4.10 Меню гри

На представлених зображеннях продемонстровано меню гри, яке взаємодіє з відповідним кодом для забезпечення динамічного управління інтерфейсом.

Рисунок 4.6 – Меню паузи

Рисунок 4.7 – Головне меню

Головне меню гри включає опції для завантаження збереженої гри, початку нової гри, налаштувань та виходу з гри. Це меню дозволяє гравцям швидко орієнтуватися і розпочати або продовжити свій ігровий процес.

Меню паузи, показане на другому зображенні, забезпечує можливість зупинити гру, завантажити збереження, зберегти поточний прогрес, змінити налаштування, вийти до головного меню або закрити гру. Це меню є важливим елементом, що дозволяє гравцям контролювати гру під час паузи.

Взаємодія з кодом, зокрема, реалізується через клас UiOpener, який відповідає за відкриття та закриття різних панелей інтерфейсу гри. Наприклад, для відкриття панелі інвентаря використовується метод ToggleInventoryPanel, який активує або деактивує панель інвентаря в залежності від сигналу

Приклад коду:

```
private void ToggleInventoryPanel(bool isOpen)
{
    if (inventoryPanel != null)
    {
        inventoryPanel.SetActive(isOpen);
        if (isOpen)
        {
            inventoryPanel.GetComponent<Inventory_Window>().Redraw();
        }
    }
}
```

Цей метод перевіряє наявність об'єкта inventoryPanel, встановлює його активність в залежності від параметра isOpen та оновлює відображення інвентаря за допомогою методу Redraw.

Подібним чином працюють й інші методи класу UiOpener, забезпечуючи зручне і гнучке управління різними панелями інтерфейсу гри, такими як панель завантаження, паузи, діалогів та тренувань. Це дозволяє гравцям мати повний контроль над інтерфейсом і взаємодією з грою, забезпечуючи комфортний ігровий процес.

Висновки до розділу 4

Розробка ігрового застосунку вимагає інтеграції багатьох компонентів, кожен з яких має вирішальне значення для створення повноцінного ігрового досвіду. Основним елементом управління грою є GameManager, який контролює стан гри, збереження та завантаження даних, а також взаємодію з іншими компонентами. Його реалізація через патерн синглтон гарантує, що в грі завжди існує лише один екземпляр цього класу, забезпечуючи узгодженість і стабільність.

Обробка ввідних даних користувача здійснюється через InputManager, який відповідає за реакцію на дії гравця, такі як рух, взаємодія з об'єктами та бойові дії. Цей компонент дозволяє налаштовувати управління під індивідуальні потреби гравця, що підвищує зручність і персоналізацію ігрового процесу. Він також керує складнішими станами гри, такими як відкриття меню паузи чи інвентаря, використовуючи події для синхронізації з іншими компонентами.

Система квестів, керована QuestManager, відіграє важливу роль у мотивації гравця виконувати завдання і просуватися по сюжету. Вона забезпечує створення, відстеження та завершення квестів, інтегруючись з іншими елементами гри для відображення прогресу і надання нагород. Це дозволяє створювати інтерактивний і динамічний ігровий процес, що відображає досягнення гравця і впливає на розвиток сюжету.

Інвентар, керований Inventory, є центральним вузлом для управління предметами гравця. Він забезпечує додавання, видалення та організацію предметів, дозволяючи гравцям ефективно керувати своїми ресурсами. Інвентар інтегрується з інтерфейсом користувача та іншими механіками гри.

Таким чином, взаємодія і узгодженість всіх цих компонентів є ключовими для створення цілісного і захоплюючого ігрового досвіду, де кожен елемент доповнює і підтримує інші, створюючи інтерактивний і реалістичний світ для гравця.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було проведено аналіз та огляд жанру RPG. Розглянуто його основні характеристики, включаючи можливість гравців зануритися у фантастичні світи, боротися зі злом, розвивати персонажів та досліджувати великі відкриті світи. Також були висвітлені відомі приклади RPG ігор.

Під час виконання кваліфікаційної роботи успішно виконані всі поставлені завдання:

- аналіз сучасних тенденцій та існуючих рішень у сфері RPG пригодницьких ігор;
- вибір технологій для створення ігрового середовища на Unity;
- розробка детальної концепції ігрового світу, сюжету, системи персонажів, ігрових механік та взаємодій;
- проєктування архітектури програмного забезпечення;
- реалізація ігрових механік;
- розробка інтерфейсу користувача

Спроєктовано архітектуру програмного забезпечення, виконано моделювання основних елементів системи, створено UML-діаграми класів. Реалізовано ігрові механіки, включаючи систему бою, розвитку персонажа, генерацію випадкових елементів гри, систему квестів та місій, анімацію, систему збереження та завантаження гри.

Використання сучасних технологій, таких як Unity Asset Store та Universal Render Pipeline, дозволило досягти високої якості графіки та продуктивності. Проаналізовано ігрові застосунки в жанрі RPG, визначено їх переваги та недоліки, що дозволило застосувати найкращі практики в даній роботі.

Таким чином, робота підтвердила актуальність і ефективність використання сучасних технологій у розробці ігрових застосунків, сприяючи популяризації ігрових застосунків та розвитку індустрії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Новини ігрової індустрії від Axios Gaming. URL: https://www.axios.com/newsletters/axios-gaming-e27ba10c-cf66-40ac-95ec-4f36fb320650.html (дата звернення: 22.03.2024).

2. Опитування Steam про обладнання та ПЗ: April 2024. URL: https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam?l=ukrainian (дата звернення: 25.03.2024).

3. Advanced Programming and Code Architecture. URL: https://unity.com/how-to/advanced-programming-and-code-architecture (дата звернення: 20.05.2024).

4. Chained Echoes. URL: https://www.chainedechoes.com/ (дата звернення: 11.05.2024).

5. Custom Renderer Features. URL: https://www.cyanilux.com/tutorials/custom-renderer-features/ (дата звернення: 11.06.2024).

6. Environment Design with Unity Terrain Features. URL: https://blog.logrocket.com/easy-environment-design-unity-terrain-features/ (дата звернення: 11.06.2024).

7. Hero of the Kingdom II. URL: https://www.gog.com/game/hero_of_the_kingdom_ii (дата звернення: 20.05.2024).

8. In depth analysis of the Gaming Industry. URL: https://bsic.it/in-depth-analysis-of-the-gaming-industry-1/ (дата звернення: 22.03. 2024).

9. Medieval Dynasty. URL: https://store.steampowered.com/news/app/1129580/view/3044969559734176490 (дата звернення: 20.05.2024).

10. The Gaming Industry in 2024: Upcoming Trends and What Not to Expect. URL: https://medium.com/my-games-company/the-gaming-industry-in-2024-

upcoming-trends-and-what-not-to-expect-cd282e7cb3cd (дата звернення: 19.03.2024).

11. The stages of the game development process. URL: https://stepico.com/blog/game-development-stages/ (дата звернення: 20.05.2024).

12. Unity Documentation. URL: https://docs.unity3d.com/Manual/UnityOverview.html (дата звернення: 02.06.2024).

13. Using the “Animation” and the “Animator” Windows in Unity. URL: https://ouzaniabdraouf.medium.com/ng-the-animation-how-to-use-the-animation-and-the-animator-windows-in-unity-f2199aacd2a6 (дата звернення: 05.06.2024).

14. Video Game Market Summary and Trends URL: https://games.logrusit.com/en/news/video-game-market-summary-and-trends/ (дата звернення: 21.03.2024).

15. Video game industry - Statistics & Facts. URL: https://www.statista.com/topics/868/video-games/#topicOverview (дата звернення: 21.03.2024).