Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«МОДЕЛЮВАННЯ ЛЮДИНО-МАШИННОЇ ВЗАЄМОДІЇ З УПРАВЛІННЯМ ЇЇ МЕТАПРАВИЛАМИ (ЗА ЗРАЗКОМ DON'T STARVE) (BACKEND)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б.	Група
1	Кузьмін Антон Дмитрович	КН-П-201
2	Юркевіч Володимир Олександрович	КН-П-201
3	Руденко Андрій Олегович	КН-П-201
4	Андрющенко Владислав Павлович	КН-Д-201
5	Латій Марія Андріївна	КН-Д-201

Автор звіту

_____ Кузьмін Антон

Одеса – 2024

АНОТАЦІЯ

Дипломна робота присвячена аналізу та розробці ігрового середовища, проєктуванню різних ігрових механік та створенню програмної архітектури, яка дозволяє швидко додавати новий ігровий контент.

Проєкт базується на грі Don’t Starve, основними механіками якої є дослідження ігрового світу та ремесло (комбінування предметів). До проєкту додано нові способи взаємодії з неігровими персонажами, зокрема розширено бойову систему та введено механіку кооперації, де неігрові персонажі допомагають гравцю. Також розроблено систему виконання завдань за винагороду в ігровому середовищі.

Мета дослідження полягає у визначенні узагальнених характеристик ігрового контенту та аналізі способів їх гнучкого та швидкого інтегрування в ігрове середовище за допомогою ігрового рушія та середовища розробки Unity, а також його внутрішніх інструментів (Cinemachine, ScriptableObjects) та сторонніх пакетів (Photon).

Для розробки було обрано рушій Unity через його велику бібліотеку безкоштовних ресурсів, які є необхідними для створення повноцінного ігрового продукту, але виходять за межі компетенцій команди, наприклад, звукове супроводження.

Організація командної роботи та відстеження виконаних завдань здійснювалися за допомогою дошок Trello. У процесі розробки широко використовувалася методологія екстремального програмування, зокрема парне програмування, що дозволило прискорити процес розробки та забезпечити обмін досвідом і знаннями серед команди програмістів.

ЗМІСТ

ВСТУП

У сучасному світі індустрія інтерактивних розваг перетворилася з локального захоплення на масовий феномен, який охоплює сотні мільйонів людей по всьому світу. Цей сектор продовжує динамічно розвиватися, впроваджуючи нові напрями, інновації та створюючи робочі місця. Сьогодні відеоігри демонструють свою культурну та мистецьку цінність, пропонуючи захопливі світи та неймовірні історії, доповнені цікавим ігровим процесом.

Жодна відеогра не існує без програмного коду, який оживляє її елементи. Сучасні тенденції вимагають від розробників створення архітектури, що дозволяє швидко додавати новий вміст без зміни основного коду. Це особливо актуально для ігор, які потребують довгострокової підтримки та оновлень.

З урахуванням цих вимог, наша команда зосередилася на розробці архітектури, яка дозволяє легко інтегрувати нові елементи, забезпечуючи гравцям постійно свіжий досвід.

Цей документ включає аналіз програмної системи, проєктування архітектури, процес розробки та майбутню підтримку. Також описані проблеми, що виникли під час розробки, і шляхи їх вирішення. Додатково наведені методи організації командної роботи над проєктом.

У наступному розділі представлено технічне завдання проєкту. У роботі розглядається створення системи керування ігровим аватаром, його характеристиками, інвентарем та взаємодією з оточенням, а також поведінка неігрових персонажів і система ремесел. У висновках підсумовано виконану роботу.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Моделювання людино-машинної взаємодії з управлінням її метаправилами (за зразком Don’t Starve)

Загальний опис.

Задачею програмного забезпечення (ПЗ) є створення системи взаємодії людини (гравця) та елементами ігрового середовища. Функцією системи є забезпечення інтерактивності довкілля: взаємодії гравця з елементами середовища, неігровими персонажами, з інтерфейсом користувача та іншими гравцями.

Кожна підсистема відповідає за окрему частину досвіду гравця, серед них: система ремесла - поєднання предметів оточення для отримання нових предметів, система досягнень - використання предметів для отримання винагороди, система дослідження внутрішньо ігрового контенту, система подолання перешкод для перевірки навичок та знань гравця про метаправила системи. Також система містить підсистеми, що забезпечуть створення ігрового оточення, з’єднання до 4 гравців через локальну мережу та мережу Інтернет для спільної взаємодії з єдиним ігровим оточенням та реалізації персоналізації та самовираження гравця в середині ігрової системи.

Призначення.

Даний проєкт розробляється для створення відеогри, яка надає гравцям можливість взаємодії з віртуальним світом та іншими гравцями. Головна мета гри полягає в наданні гравцям тривалого та захоплюючого ігрового досвіду через різноманітні ігрові механіки, такі як дослідження, ремесло, бій, кооперація та виконання завдань. Гра орієнтована на підтримку регулярного оновлення контенту, що дозволить зберігати інтерес гравців протягом довгого часу та забезпечити їх повернення до гри.

Вимоги до функціоналу.

Головне меню

Стартовий інтерфейс ПЗ (головна меню або стартова сцена) служить вихідною точкою для користувача та повинна містити можливість налаштувань звуку, налаштування кастомізації ігрового аватара користувача, створення власного ігрового оточення (світу) для одиночної гри, гри по локальній мережі та грі по мережі Інтернет, а також під'єднання до існуючих ігрових оточень в локальній мережі та мережі Інтернет.

Системи взаємодії

Системи взаємодії гравця з системою (ігрові механіки) забезпечують гравця задачами та цілями в середині ігрової системи (світу).

- Механіка комбінування предметів для отримання нових (система ремесла) заключається в пошуку предметів в ігровому світі та комбінуванні даних предметів. Комбінування предметів відбувається за допомогою спеціальних предметів ігрового оточення. Для розробників потрібно забезпечити зручний інструмент для додавання нових предметів та їх комбінацій
- Механіка взаємодії з активними об'єктами ігрового оточення (неігровими персонажами, НІП). Неігрові персонажи - набір персонажів, керування якими здійснюється системою і взаємодія гравця з якими може різнитися залежно від цілі гравця та типу НІПа. Для розробників необхідно забезпечити зручний інструмент для додавання нових НІПів.
    - Діалог. Це тип взаємодії з неігровими персонажами, в результаті якої гравець отримує завдання, які необхідно виконати всередині системи для отримання винагороди.
    - Бій. Це тип взаємодії, суть якої полягає в змаганні гравця з системою, в якій гравцю, використовуючи наявні інструменти

необхідно знизити показник НІПа до нуля раніше, ніж НІП зробить це у гравця. За перемогу в змаганні гравець отримує винагороду у вигляді предметів, а за поразку - покарання у вигляді втрати предметів та прогресу.

- Кооперація. Це тип взаємодії з неігровими персонажами, суть якої полягає у використанні спеціальних предметів з метою переведення НІПа під керування гравцем.

Багатокористувацький режим

Багатокористувацький режим - це спеціальний режим гри, який забезпечує з'єднання до 4 гравців на різних пристроях в рамках єдиного ігрового оточення. Даний режим має реалізовувати з'єднання по локальній мережі та мережі Інтернет.

Архітектура.

Архітектура програмного забезпечення відеогри включає в себе кілька підсистем, які взаємодіють між собою для забезпечення цілісного ігрового досвіду. Нижче наведено основні компоненти архітектури та їх опис:

Основні компоненти

Ядро

Опис: Основні механізми гри, включаючи головний цикл, керування станами гри, завантаження ігрових сцен тощо.

Функції:

- Менеджер станів гри
- Завантаження/збереження прогресу
- Керування ресурсами

Гравець

Опис: Всі аспекти, що стосуються гравця, включаючи управління аватаром, інвентарем тощо.

Функції:

- Контроль аватара
- Інвентар та ремесло

Середовище

Опис: Система, що створює та керує ігровим світом, включаючи генерацію рівнів, ресурси та інтерактивні об'єкти.

Функції:

- Генерація рівнів
- Розміщення ресурсів
- Керування об'єктами оточення

Неігрові персонажі

Опис: Логіка та поведінка всіх неігрових персонажів (НІПів), включаючи ворогів, союзників та нейтральних персонажів.

Функції:

- Логіка поведінки НІПів
- Система діалогів
- Взаємодія з гравцем (бій, кооперація)

Завдання

Опис: Система завдань, яка забезпечує створення та керування квестами та місіями для гравця.

Функції:

- Створення квестів
- Відслідковування прогресу
- Винагороди за виконання завдань

Багатокористувацький режим

Опис: Інфраструктура для підключення гравців через локальну мережу та мережу Інтернет для спільної гри.

Функції:

- З'єднання гравців
- Синхронізація станів гри
- Керування сесіями

Інтерфейс користувача

Опис: Всі елементи інтерфейсу користувача, включаючи головне меню, інвентар, кастомізацію та інші елементи управління.

Функції:

- Головне меню
- Налаштування
- Інтерактивні елементи

Технології.
Зберігання вихідних кодів ПЗ здійснюється за допомогою системи контроля версій (вибір системи узгоджується групою проєкту).
Структура файлів та каталогів повинна відображати архітектуру ПЗ:

- Design
- Scripts
    - Core
    - Player

- Environment
- NPCs
- Quests
- Multiplayer
- UI

Для розробки рекомендуються наступні технології створення:

- Ігровий процес: Ігровий рушій Unity та мова програмування C#
- Дизайн: Adobe Photoshop, Adobe Illustrator, FontEditor, Adobe InDesign
- Публікація: веб-ресурс itch.io та технологія WebGL

Команда проєкту.

Для виконання поставлених завдань робота команди передбачається окремо за кожною частиною ігрової функціональності.

При команді в 5 учасників троє учасників відповідають за створення програмних систем, 2 відповідають за візуальну частину наповнення ігрового контенту.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У результаті ретельного аналізу вимог технічного завдання до проєкту наша команда вирішила використовувати ігровий рушій Unity для розробки завдяки його високій продуктивності, зручності використання, сучасним можливостям розробки та обширній бібліотеці ресурсів спільноти. Unity дозволяє створювати якісні ігри, використовуючи численні інструменти, які спрощують процес розробки та забезпечують високу продуктивність кінцевого продукту.

Основу програмної архітектури складають такі компоненти:

Core - Основні механізми гри, включаючи головний цикл, керування станами гри, завантаження ігрових сцен тощо.

Player - Усі аспекти, що стосуються гравця, включаючи управління аватаром, інвентарем тощо.

Environment - Система, що створює та керує ігровим світом, включаючи генерацію рівнів, ресурси та інтерактивні об'єкти.

NPCs - Логіка та поведінка всіх неігрових персонажів (НІПів), включаючи ворогів, союзників та нейтральних персонажів.

Quests - Система завдань, яка забезпечує створення та керування квестами та місіями для гравця.

Multiplayer - Інфраструктура для підключення гравців через локальну мережу та Інтернет для спільної гри.

UI - Всі елементи інтерфейсу користувача, включаючи головне меню, інвентар, кастомізацію та інші елементи управління.

Design - Набір готових елементів інтерфейсу для головного меню, меню створення ігрового світу, меню підключення до гри через мережу та меню в середині гри, такі як меню паузи, меню гравця, інвентар, показники.

Частина "Design" включає створення та налаштування елементів інтерфейсу, які забезпечують зручність користування грою. Для меню гравця та інвентарю, за допомогою інструментів Unity, додано анімації відкриття та закриття, що надає грі динамічний та привабливий вигляд. Також "Design" включає набір зображень (спрайтів) для аватара гравця, неігрових персонажів, елементів ігрового середовища, а також такі елементи брендингу, як логотип та шрифт. Важливим аспектом було забезпечення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє гравцям легко взаємодіяти з грою.

Програмна частина проєкту була повністю реалізована з використанням мови C# та Unity Scripting API. Широко використовувалися додаткові пакети та інструменти Unity, такі як Cinemachine для налаштування камери, що забезпечує плавне та якісне управління камерою в грі, а також сценарні об’єкти (Scriptable Objects), які спрощують зберігання і управління даними. Для реалізації багатокористувацького режиму було використано сторонній пакет Photon, який обрали через його зручність та легкість впровадження до проєктів Unity, що дозволило швидко та ефективно реалізувати функціонал мережевої гри.

Для збереження ігрового прогресу та створених ігрових світів використовувалася технологія JSON. Це забезпечує зручний та гнучкий спосіб зберігання даних, дозволяючи легко зберігати та завантажувати стан гри.

При проєктуванні та розробці програмної архітектури широко застосовувалися різні патерни об’єктно-орієнтованого проєктування, такі як "Машина станів" (State Machine) та Model-View-Controller (MVC). Використання цих патернів дозволило створити структуровану та модульну архітектуру, яка полегшує підтримку та розширення проєкту. Головним принципом проєктування було дотримання принципів SOLID,

що забезпечує високу якість коду, легкість його розширення та підтримки, а також низьку зв’язаність (low coupling) між компонентами системи.

Командна робота була організована за допомогою елементів різних методологій керування проєктами, що забезпечило ефективність і злагодженість процесу розробки. Використовувалися елементи SCRUM з тижневими спринтами, що дозволило чітко планувати та контролювати виконання задач на короткі проміжки часу. Для відстежування потоку задач використовувалися дошки Kanban, які забезпечували візуалізацію процесу роботи і допомагали команді бачити прогрес виконання завдань у режимі реального часу.

Для прискорення розробки програмного коду, передачі досвіду між членами команди та уникнення такого явища як "вежа знань", використовувався елемент екстремального програмування, а саме парне програмування. Цей підхід дозволив підвищити якість коду, зменшити кількість помилок та забезпечити безперервний обмін знаннями між розробниками. Парне програмування сприяло кращому розумінню коду всіма членами команди та збільшило гнучкість у випадку зміни складу команди чи ролей всередині неї.

Вихідні файли проєкту зберігаються у публічному репозиторії на GitHub, що дозволяє легко відстежувати зміни, співпрацювати з іншими розробниками та забезпечувати контроль версій. Репозиторій GitHub доступний за посиланням: GitHub.

РОЗДІЛ 2
ДІАЛОГОВА СИСТЕМА

Діалогова система є важливим елементом багатьох ігор, оскільки вона забезпечує взаємодію між гравцем та неігровими персонажами (NPC). В даній роботі розглядається реалізація діалогової системи в ігровому движку Unity з використанням Scriptable Objects, що дозволяє створювати гнучкі та масштабовані рішення.

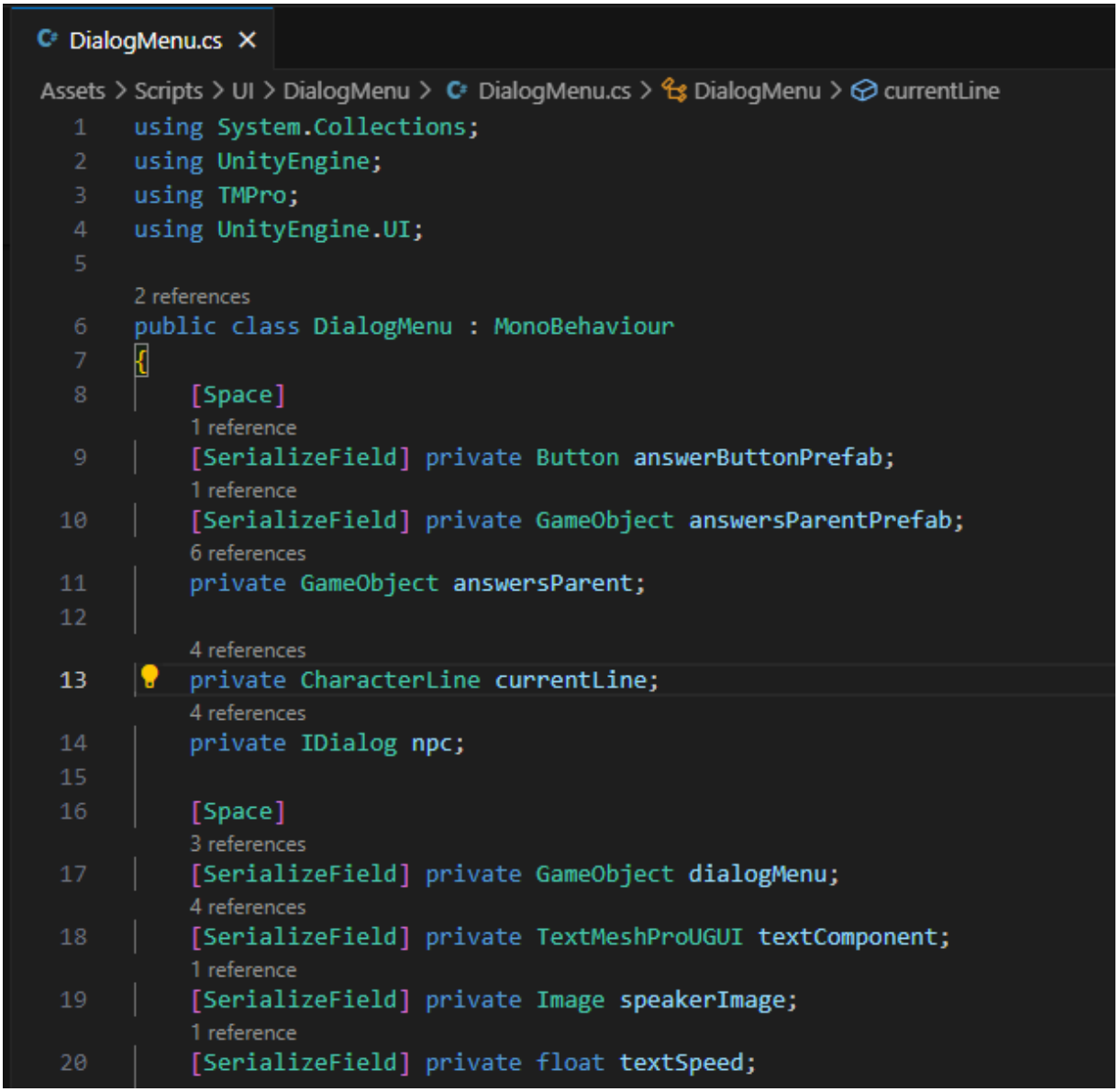
Scriptable Objects в Unity надають зручний спосіб зберігання та організації даних. Вони є незалежними від сцени об'єктами, які можуть бути збережені у вигляді ресурсів. Це дозволяє зберігати дані діалогів окремо від ігрових об'єктів, що полегшує їх редагування та повторне використання.

CharecterLine.cs

Assets > Scripts > UI > DialogMenu > CharecterLine.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   [CreateAssetMenu(fileName ="Character Line", menuName ="Dialog System/Character Line")]
    0 references
6   public class CharacterLine : ScriptableObject
7   {
        1 reference
8       [SerializeField] private string line;
        0 references
9       public string Line => line;
        1 reference
10      [SerializeField] private List<PlayerAnswers> answers;
        0 references
11      public List<PlayerAnswers> Answers => answers;
        1 reference
12      [SerializeField] private QuestsSO quest;
        0 references
13      public QuestsSO Quest => quest;
14  }
```

Спочатку був створений клас CharacterLine для зберігання діалогів. Для цього ми створюємо клас, що успадковується від ScriptableObject. Потім був створений менеджер діалогів що відповідає за відображення діалогів на екрані.



Використання Scriptable Objects для реалізації діалогової системи в Unity забезпечує гнучкість і зручність у роботі з даними. Це дозволяє легко додавати нові діалоги, редагувати існуючі та повторно використовувати їх для різних NPC. Такий підхід значно спрощує розробку та підтримку ігрового контенту.

РОЗДІЛ 3
КВЕСТОВА СИСТЕМА

Квестова система є важливим компонентом багатьох ігор, оскільки вона забезпечує основну мотивацію для гравця та створює структуру ігрового процесу. В даній роботі розглядається реалізація квестової системи в ігровому движку Unity з використанням Scriptable Objects, що дозволяє легко додавати та управляти квестами для кожного NPC. Квести можна брати через інтеграцію з діалоговою системою, використовуючи кнопки для взаємодії.

Scriptable Objects в Unity забезпечують зручний спосіб зберігання та організації даних, дозволяючи створювати гнучкі та повторно використовувані квестові об'єкти. Це дозволяє зберігати дані квестів окремо від ігрових об'єктів, що полегшує їх редагування та повторне використання.

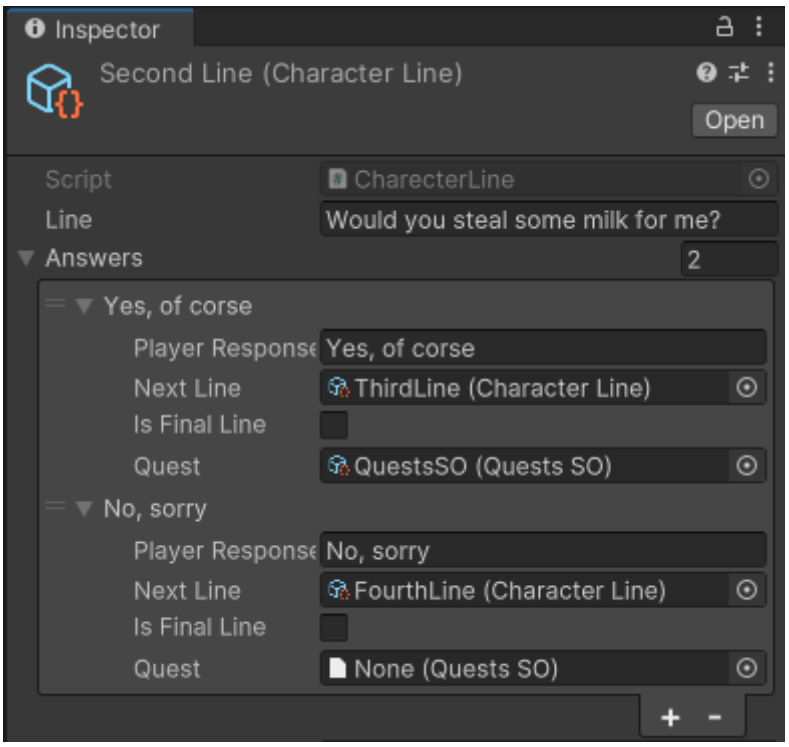
Спочатку необхідно створити клас для зберігання квестів. Для цього ми створюємо клас, що успадковується від ScriptableObject:

QuestsSO.cs

Assets > Scripts > Quests > QuestsSO.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   [CreateAssetMenu(fileName = "QuestsSO", menuName = "Quests/QuestsSO")]
    0 references
6   public class QuestsSO : ScriptableObject
7   {
        1 reference
8       [SerializeField] private string questGiver;
        0 references
9       public string QuestGiver => questGiver;
        1 reference
10      [SerializeField] public string description;
        0 references
11      public string Description => description;
12  }
```

Такі ScriptableObject можна додавати до діалогів персонажів і це автоматично буде активувати цей квест у менеджері квестів.



Менеджер квестів відповідає за управління активними квестами гравця.

Квестовий журнал (Quest Log) є важливим елементом будь-якої квестової системи, оскільки він дозволяє гравцеві відслідковувати свій прогрес у виконанні квестів. У нашій реалізації квестовий журнал відображає список активних квестів та статус їх виконання.

Квестовий журнал реалізований у вигляді окремого UI-компонента, який можна викликати за допомогою кнопки в грі.

Квестовий журнал інтегрується з менеджером квестів (Quest Manager), який веде облік активних квестів та керує їхнім станом. Менеджер квестів надає список активних квестів та інформацію про їх статус для оновлення вмісту квестового журналу.

Використання Scriptable Objects для реалізації квестової системи в Unity забезпечує гнучкість і зручність у роботі з даними. Інтеграція квестової системи з діалоговою системою дозволяє легко брати квести через інтерфейс діалогу, що значно спрощує взаємодію гравця з NPC. Квестовий журнал дозволяє гравцям відслідковувати прогрес виконання квестів, що робить систему ще більш інтуїтивно зрозумілою та зручною. Такий підхід значно полегшує розробку та підтримку ігрового контенту, роблячи систему квестів легко розширюваною та масштабованою.

ВИСНОВКИ

У процесі виконання дипломної роботи було проведено всебічне дослідження та реалізацію ігрового середовища, що ґрунтується на українській дохристиянській культурі та міфології. Основними результатами є наступні:

В основі проєкту лежить аналіз і розширення ігрових механік, базуючись на популярній грі Don’t Starve. Додано нові способи взаємодії з неігровими персонажами, розширено бойову систему та введено механіку кооперації, що дозволяє НІПам допомагати гравцю.

Розробка проєкту здійснювалася з використанням Unity, C#, Visual Studio, WebGL та інших інструментів. Для створення багатокористувацького режиму застосовано пакет Photon, що забезпечив зручність і ефективність реалізації мережевої гри.

В процесі розробки широко застосовувалися об'єктно-орієнтовані патерни проєктування, що дозволило створити структуровану та модульну архітектуру. Принципи SOLID забезпечили високу якість коду, легкість його розширення та підтримки.

Впровадження елементів SCRUM з тижневими спринтами та використання дошок Kanban сприяло ефективній організації командної роботи. Парне програмування дозволило підвищити якість коду та забезпечити безперервний обмін знаннями між розробниками.

Використання машини станів та сценарних об’єктів дозволило легко розширювати та модифікувати поведінку НІПів, що забезпечує гнучкість та масштабованість системи. Застосування JSON для збереження ігрового прогресу забезпечило зручність та гнучкість у зберіганні даних.

Загалом, проведена робота демонструє можливості створення якісного ігрового контенту з використанням сучасних технологій та методологій. Результати можуть бути використані для подальшого розвитку проєкту та впровадження нових ігрових механік.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ НАВЕДЕНОЇ ЧАСТИНИ ПРОЄКТУ

№ п.п.	Ресурс	Призначення
1.	Unity; Розробник: Unity Technologies; Документація: посилання	Використано в якості ігрового рушія та середовища розробки
2.	Мова програмування C# Розробник: Microsoft Документація: посилання	Була обрана мовою програмування для реалізації ігрової логіки
3.	Visual Studio; Розробник: Microsoft Документація: посилання	Використано в якості головного інструмента редактору коду
4.	Cinemachine; Розробник: Unity Technologies; Документація: посилання	Використано для налаштування ігрової камери