ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтерактивна гра на основі C#»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Герман СЕНОЗАЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Герман СЕНОЗАЦЬКИЙ
група КНДМ-61

Керівник: Сергій СЄРИХ
науковий ступінь, вчене звання к. т. н., доцент

Рецензент:
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сенозацький Герман Олегович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтерактивна гра на основі C#»

керівник кваліфікаційної роботи Сергій СЄРИХ к. т. н., доцент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи розробки програмного забезпечення з використанням Unity та мови програмування C#.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1. Дослідження інтерактивних наративів та методів їх реалізації у візуальних новелах.
   4.2. Проектування архітектури гри та підбір технологій для її реалізації.
   4.3. Розробка системи діалогів, виборів і сценарних гілок для інтерактивної гри.

5. Перелік графічного матеріалу: *презентація*
   5.1 Основні компоненти архітектури інтерактивної гри.
   5.2 Приклади реалізації діалогової системи.
   5.3 Підходи до проектування сценарних гілок.
   5.4 Графічні інтерфейси створеної гри.

6. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.24	виконано
2	Дослідження інтерактивних наративів та візуальних новел	30.09-11.10.24	виконано
3	Проектування архітектури гри та підбір технологій	14.10-25.10.24	виконано
4	Розробка діалогової системи	28.10-08.11.24	виконано
5	Реалізація виборів і сценарних гілок	11.11-22.12.24	виконано
6	Тестування та інтеграція компонентів гри	25.11-29.11.24	виконано
7	Оформлення роботи: вступ, висновки, реферат	02.12-06.12.24	виконано
8	Розробка демонстраційних матеріалів	09.12-13.12.24	виконано

Здобувач вищої освіти _____ Герман СЕНОЗАЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій СЄРИХ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 76 стор., 37 рис., 21 джерело.

Наукове завдання – розробка інтерактивної гри у жанрі візуальної новели із використанням мови програмування C# та рушія Unity.

Мета роботи – створити інтерактивну гру, яка забезпечує емоційне занурення гравця та дозволяє динамічно адаптувати сюжет залежно від його виборів.

Об'єкт дослідження – процес створення інтерактивних ігор із нелінійним наративом.

Предмет дослідження – технології розробки інтерактивних ігор у жанрі візуальних новел.

Методи дослідження – аналіз сучасних інтерактивних наративів і візуальних новел. Проектування архітектури гри. Алгоритмізація діалогових систем і сюжетних гілок. Використання методології об'єктно-орієнтованого програмування.

Короткий зміст роботи: Проведено аналіз інтерактивних наративів у сучасних відеоіграх, досліджено основні функції та механіки візуальних новел. Розглянуто особливості жанру, включаючи багатоваріантність сюжетів, взаємодію гравця з персонажами та можливість повторного проходження. Проектування гри виконано з використанням Unity та мови програмування C#. Реалізовано систему діалогів, виборів та інтерактивних меню з використанням Scriptable Objects. Тестування показало повну відповідність розробленого продукту поставленим завданням. Гра включає кілька кінцівок, що залежать від виборів гравця, забезпечуючи повторюваний ігровий досвід.

Ключові слова: інтерактивні ігри, візуальні новели, Unity, C#, сценарні гілки, діалогові системи.

ABSTRACT

Text part of the master's qualification work: 76 pages, 37 pictures, 21 sources.

Scientific task is to develop an interactive game in the visual novel genre using the C# programming language and the Unity engine.

Goal of the work is to create an interactive game that ensures emotional immersion of the player and dynamically adapts the storyline based on their choices.

Object of research – the process of creating interactive games with nonlinear narratives.

Subject of research – technologies for developing interactive games in the visual novel genre.

Research methods – analysis of modern interactive narratives and visual novels. Design of game architecture. Algorithmization of dialogue systems and branching storylines. Application of object-oriented programming methodology.

Summary of the work: An analysis of interactive narratives in modern video games was conducted, and the main features and mechanics of visual novels were studied. The unique characteristics of the genre were reviewed, including storyline branching, player-character interaction, and the possibility of replayability. The game design was implemented using Unity and the C# programming language. A dialogue system, choice mechanics, and interactive menus were developed using Scriptable Objects. Testing demonstrated that the developed product fully meets the set objectives. The game includes multiple endings, which depend on the player's choices, providing a replayable gameplay experience.

Keywords: interactive games, visual novels, Unity, C#, branching storylines, dialogue systems.

Зміст

ВСТУП

Інтерактивні наративи є важливою складовою сучасних відеоігор, адже вони дозволяють гравцям впливати на розвиток сюжету через вибір дій та рішень. Це забезпечує динамічність ігрового процесу, унікальний досвід для кожного користувача та сприяє зануренню у віртуальний світ. Завдяки таким елементам, відеоігри стають не лише джерелом розваг, а й потужним інструментом для емоційної взаємодії, розвитку критичного мислення та вивчення впливу поведінки на різні сценарії. Особливу увагу привертають ігри, які використовують технології програмування для створення динамічних, адаптивних наративів, здатних змінюватися в реальному часі залежно від рішень гравця.

Актуальність теми полягає у зростаючому попиті на ігри з інтерактивними наративами, що адаптуються до вибору гравця. Використання мови програмування C# у поєднанні з інтерактивним сценарним дизайном дозволяє створювати гнучкі та захопливі ігрові світи. Такий підхід робить гру не тільки цікавим дозвіллям, а й інструментом для дослідження емоційної взаємодії між гравцем та ігровими персонажами.

Метою дипломної роботи є дослідження підходів до розробки інтерактивних наративів у відеоіграх та створення прототипу гри, яка адаптується до емоційних рішень гравця.

Завданнями роботи є:

- Аналіз існуючих підходів до створення інтерактивних наративів.
- Розробка моделі інтерактивного сценарію, що змінюється відповідно до вибору гравця.
- Реалізація прототипу гри з використанням мови програмування C#.
- Тестування інтерактивної гри з метою оцінки ефективності її наративної структури.
- Вивчення впливу інтерактивного сюжету на залученість та задоволення гравця.

Об’єктом дослідження є процес створення та впровадження інтерактивних наративів у відеоігри.

Предметом дослідження є методи та технології розробки інтерактивних сюжетів, що адаптуються до рішень гравців, із використанням мови програмування C#.

Результати дослідження можуть бути використані для створення нових ігрових продуктів, орієнтованих на інтерактивність та динамічність, а також для вдосконалення існуючих ігор шляхом впровадження адаптивних сюжетних ліній. Розроблений прототип стане прикладом інтеграції сучасних програмних технологій із наративним дизайном, що дозволить гравцям отримати унікальний ігровий досвід, заснований на їхніх рішеннях та емоція

1 ЗАГАЛЬНА ЧАСТИНА

1.1 Постановка задачі

Метою дипломної роботи є створення інтерактивної гри на основі мови програмування C#, яка реалізує гнучкий інтерактивний наратив, що адаптується до виборів гравця та його емоційних рішень. Основна концепція гри полягає у створенні сюжетної системи, яка реагує на дії гравця в реальному часі, забезпечуючи унікальний ігровий досвід для кожного проходження.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Розробити архітектуру гри, що включає інструменти для зручного створення діалогів, монологів, задніх фонів та меню вибору гравця.
- Реалізувати систему інтерактивних діалогів, яка дозволяє гравцеві впливати на розвиток сюжету шляхом прийняття рішень.
- Інтегрувати засоби Unity для забезпечення зручного управління сценарними елементами, такими як зміна фону, тексту та вибіркових варіантів відповідей.
- Забезпечити гнучкість у зміні та масштабуванні контенту гри, використовуючи об'єктно-орієнтовані підходи до програмування.

Гра буде створена у жанрі візуального роману, де гравець, керуючи головним героєм, взаємодіє з трьома магічними істотами — духом лісу, феєю та тінню. Сюжет будується на основі емоційних виборів гравця, що визначають одну із шести можливих кінцівок. Важливим елементом є можливість повторного проходження гри з метою дослідження альтернативних сценаріїв.

Програмна реалізація гри буде здійснена за допомогою мови програмування C# у середовищі Unity. Вибір Unity зумовлений її потужними можливостями для створення інтерактивних наративів, підтримкою роботи з діалоговими системами та графічними інтерфейсами.

Очікуваним результатом є створення інтерактивного продукту, що поєднує сучасні технології програмування з творчим підходом до сценарного дизайну. Гра має продемонструвати, як інтерактивний наратив може впливати на емоційний досвід гравців та формувати динамічний сюжет.

1.2 Дослідження і аналіз об’єкта програмування

Комп'ютерні ігри є одним із найпопулярніших видів розваг у сучасному світі. Їхній розвиток розпочався у 1950–60-х роках, коли були створені перші примітивні прототипи, як-от Tennis for Two (1958) або Spacewar! (1962). Значний прорив у популярності ігор стався у 1980-х, із поширенням аркадних автоматів і домашніх консолей. Саме тоді почалася диверсифікація жанрів, які включали екшени, стратегії та симулятори. Однак серед різноманіття жанрів особливу увагу привернули інтерактивні наративи, які стали основою для жанру візуальних новел.

Жанр візуальних новел виник у 1980-х роках, але справжнього визнання набув у 1990-х в Японії завдяки проектам, які поєднували у собі текстові історії, ілюстрації та інтерактивність. Однією з перших відомих візуальних новел була Kanon (1999), яка задала нові стандарти для цього жанру, пропонуючи гравцям захопливий сюжет, багатовимірних персонажів і можливість впливати на розвиток історії. Популярність цього жанру зросла завдяки його здатності занурювати гравців у інтерактивний досвід, що поєднує літературний стиль із графічним дизайном і музикою. На рисунку 1.1 зображено типовий приклад інтерфейсу класичної візуальної новели.

Рисунок 1.1 – Інтерфейс класичної візуальної новели

Інтерактивні наративи у сучасних відеоіграх набувають все більшої популярності, адже дозволяють створювати унікальні ігрові сценарії, які залежать від вибору гравця. Це особливо актуально для жанру візуальних новел, де сюжет і механіка взаємодії повністю фокусуються на історії, емоційному впливі та особистій участі гравця. Візуальні новели є жанром відеоігор, який поєднує в собі текстовий сценарій, статичні або анімовані ілюстрації та інтерактивні елементи. Гравець, виконуючи роль головного героя, приймає ключові рішення, що впливають на розвиток сюжету, взаємини з персонажами та кінцівку гри.

У візуальних новелах сюжет є центральним елементом, який доповнюється візуальним оформленням, звуковими ефектами та музикою. Однією з ключових особливостей таких ігор є можливість повторного проходження, адже різні вибори гравця призводять до унікальних сценаріїв, розвитків подій та кінцівок. Наприклад, у грі Steins;Gate вибори гравця формують кілька різних сюжетних ліній, які взаємопов’язані та відкривають глибину персонажів та їхніх взаємин.

Різноманіття жанру візуальних новел можна класифікувати за рівнем інтерактивності. Класичні новели зосереджені на тексті та статичних зображеннях, залишаючи інтерактивність мінімальною. Такі ігри підходять для створення розгорнутих історій із численними діалогами та описами. У той же час

кінематографічні новели більше схожі на інтерактивне кіно, використовуючи озвучені діалоги, анімації та високоякісну графіку, як це реалізовано у грі Danganronpa. Цей підхід проілюстровано на рисунку 1.2, який демонструє різноманіття графічного оформлення в таких іграх.

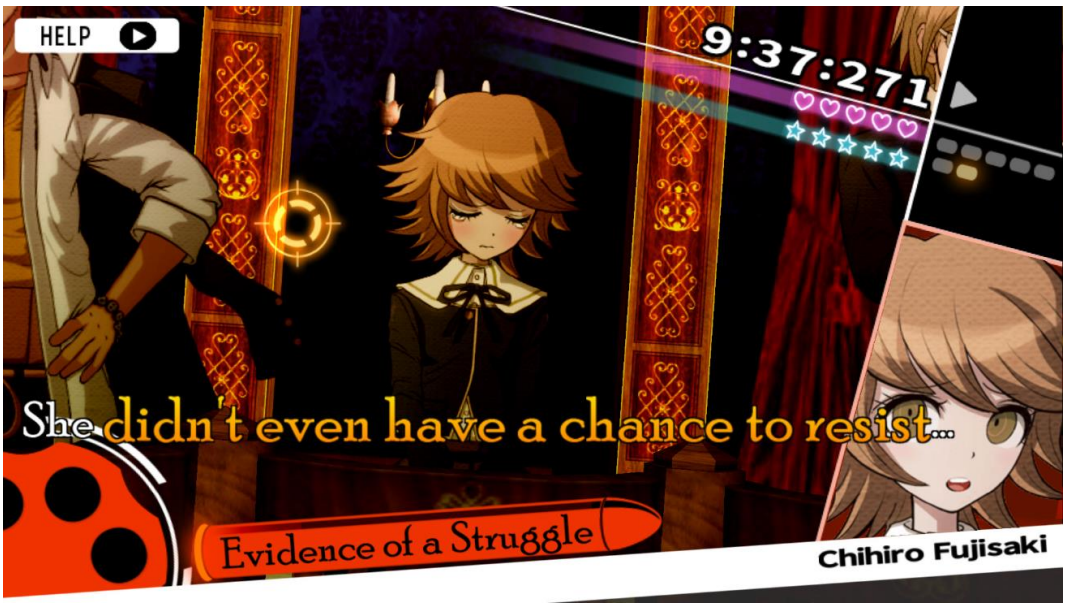

Рисунок 1.2 – Графічне оформлення у кінематографічних новелах

Ігри, які поєднують унікальні механіки з інтерактивними наративами, демонструють нові підходи до побудови сюжетів та залучення гравців. Наприклад, Oxenfree використовує інтерактивну систему діалогів у реальному часі, що робить ігровий процес більш динамічним. Репліки обираються без паузи, що імітує природний хід розмови та вимагає швидкої реакції від гравця.

Інший приклад — Undertale, гра, яка змінює уявлення про проходження сюжетів. Вона дозволяє гравцю пройти весь сюжет без насильства, використовуючи механіки переговорів або втечі. Такий підхід створює сильний емоційний зв’язок із персонажами, адже вибори гравця мають значний вплив на сюжет і кінцівки.

Ще одним популярним видом візуальних новел є ті, що включають додаткові геймплейні механіки, як-от міні-ігри чи головоломки. Наприклад, у серії Phoenix Wright: Ace Attorney гравець виступає адвокатом, який повинен вирішувати правові

головоломки, що впливають на розвиток справи та сюжету загалом. Ці ігри вміло поєднують інтерактивний наратив із елементами логічного мислення.

Візуальні новели також часто використовують функціонал, який дозволяє гравцю більше контролювати свій досвід. Наприклад, діалогові дерева пропонують вибір реплік, що формують стосунки між персонажами. Інтерактивні меню дають змогу вибирати різні дії чи напрями розвитку подій, а галереї сцен і персонажів дозволяють переглядати важливі моменти історії після завершення гри.

Основні функції візуальних новел включають систему вибору, яка визначає перебіг історії, зміну фонів і музики залежно від контексту сцени, а також зручний інтерфейс для читання діалогів і управління виборами. Важливим елементом є багатоваріантність історії, що дозволяє створювати численні кінцівки. Завдяки цьому гравці можуть досліджувати всі можливі результати своїх рішень, що підвищує цінність повторного проходження.

Таким чином, жанр візуальних новел демонструє, як ігровий процес може бути зосереджений на сюжеті та виборах гравця, створюючи емоційно насичений ігровий досвід.

Розглянемо детальніше існуючі інтерактивні ігри.

Розглянемо гру Detroit: Become Human, яка зображена на рисунку 1.3

Detroit: Become Human є кінематографічною інтерактивною грою, створеною студією Quantic Dream. Дія гри відбувається у майбутньому, де роботи-андроїди стали невід'ємною частиною суспільства. Гравець керує трьома головними персонажами-андроїдами, кожен із яких має свою унікальну сюжетну лінію. Головна особливість гри — система виборів і наслідків, яка впливає на подальший розвиток подій і кінцівку історії.

Основні функції: динамічні діалогові дерева, що змінюються залежно від виборів гравця; багатоваріантність сюжетів; кінематографічний стиль подачі; емоційна залученість через глибокий сценарій.

Detroit: Become Human була випущена 25 травня 2018 року компанією Quantic Dream.

Рисунок 1.3 - Ігровий процес гри Detroit: Become Human

Розглянемо гру Life is Strange, яка зображена на рисунку 1.4

Life is Strange — це епізодична інтерактивна пригода, розроблена студією Dontnod Entertainment. Сюжет гри обертається навколо Макс Колфілд, дівчини-фотографа, яка виявляє здатність повертати час назад. Гравець використовує цю здатність для прийняття ключових рішень, які змінюють розвиток подій у грі. Life is Strange привертає увагу своїм емоційним сюжетом і моральними дилемами.

Основні функції: система управління часом, багатоваріантність рішень і їх наслідків, сильна емоційна складова, музичний супровід, що доповнює атмосферу.

Life is Strange була випущена 30 січня 2015 року компанією Square Enix.

Рисунок 1.4 - Ігровий процес гри Life is Strange

Розглянемо гру The Walking Dead, яка зображена на рисунку 1.5

The Walking Dead — інтерактивна драма, створена студією Telltale Games. Гра базується на популярній серії коміксів і переносить гравця у світ постапокаліптичної зомбі-апокаліпсиса. Гравець керує персонажем Лі Евереттом, який повинен захищати дівчинку Клементину. Сюжет гри змінюється залежно від рішень гравця, створюючи атмосферу моральних дилем і напруги.

Основні функції: система моральних виборів, епізодична структура гри, глибокі взаємини між персонажами, емоційний вплив на гравця.

The Walking Dead була випущена 24 квітня 2012 року студією Telltale Games.

Рисунок 1.5 - Ігровий процес гри The Walking Dead

Розглянемо гру Heavy Rain, яка зображена на рисунку 1.6

Heavy Rain є інтерактивною драмою від студії Quantic Dream. Гра фокусується на історії чотирьох персонажів, які шукають викрадача дітей, відомого як "Майстер Орігамі". Гравець керує персонажами та приймає рішення, які змінюють сюжет і визначають, хто з героїв виживе до кінця історії. Heavy Rain вражає кінематографічною подачею і реалістичним зображенням емоційних переживань.

Основні функції: багатоваріантність кінцівок, нелінійний розвиток подій, управління кількома персонажами, кінематографічний стиль із сильним акцентом на емоції.

Heavy Rain була випущена 18 лютого 2010 року компанією Quantic Dream.

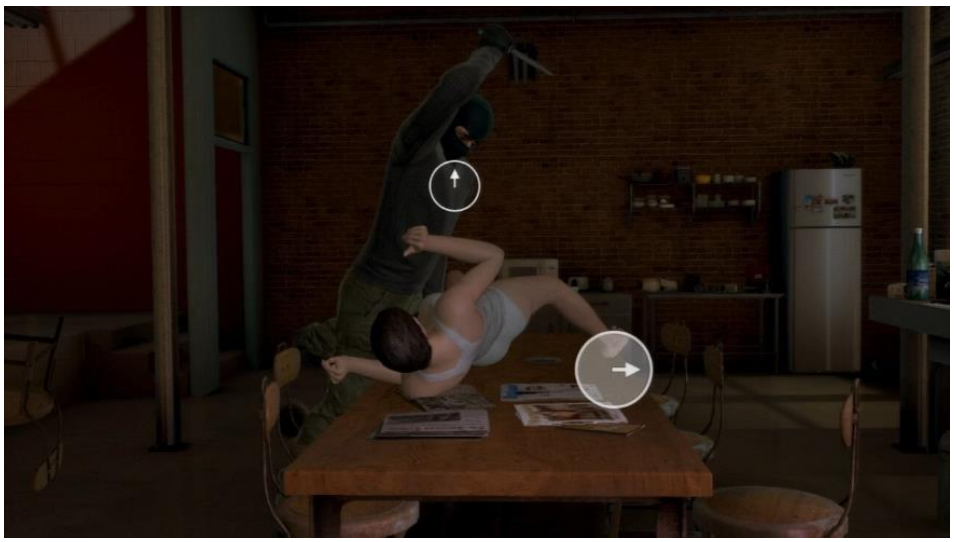

Рисунок 1.6 - Ігровий процес гри Heavy Rain

Ключовими функціональними елементами візуальних новел є:

- Інтерактивні діалогові дерева: вони дозволяють гравцю робити вибір, який безпосередньо впливає на сюжет.
- Системи пам’яті вибору: ці механіки забезпечують зміну подій залежно від рішень, прийнятих у попередніх етапах гри.
- Інтуїтивний інтерфейс: можливість пропуску сцен, перегляду попередніх виборів або використання зручних кнопок для взаємодії підвищує комфорт гри.

1.3 Використані програмні засоби

Unity – це сучасний рушій для розробки ігор, який дозволяє створювати ігри для багатьох платформ, включаючи Windows, macOS, Android, iOS, PlayStation,

Xbox і навіть веббраузери. Цей рушій підтримує 2D- та 3D-ігри, але особливо популярний серед розробників інтерактивних ігор, таких як візуальні новели.

Як зображено на рисунку 1.7, основне середовище Unity складається з вікна сцени, ієрархії об’єктів, інспектора властивостей та консолі.

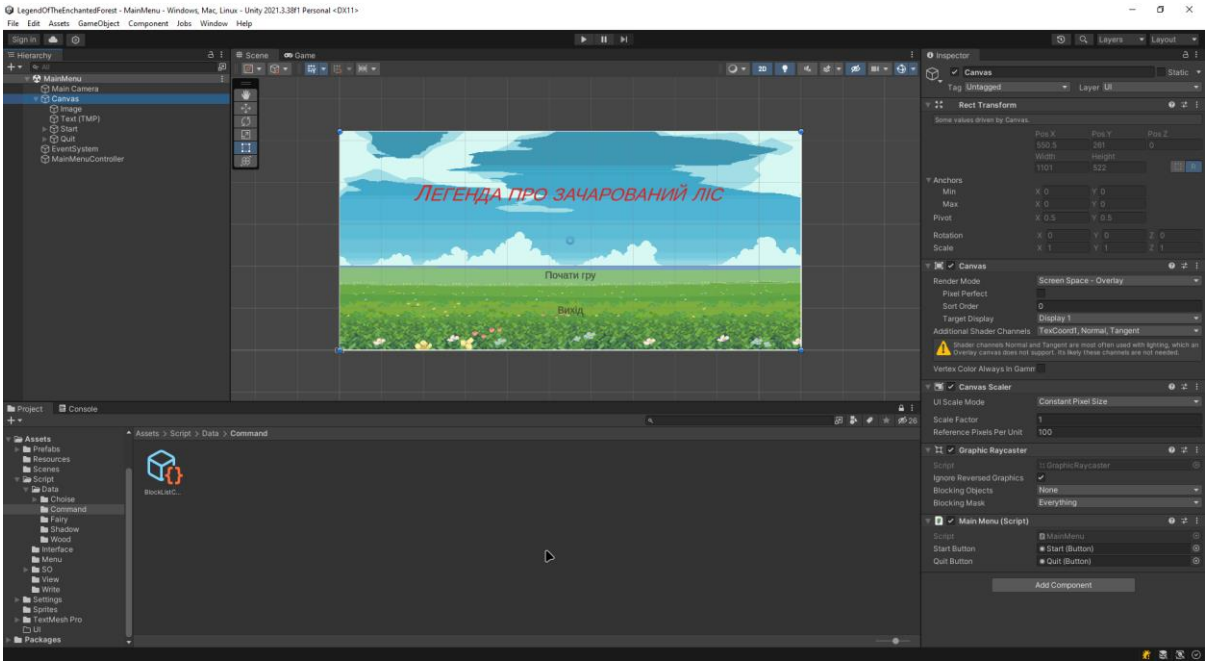

Рисунок 1.7 - Середовище розробки Unity

Для теми інтерактивних ігор Unity є оптимальним вибором через свою підтримку компонентної моделі, можливість інтеграції різноманітного контенту (зображення, звуки, відео) та широкий спектр налаштувань для управління логікою гри.

Unity працює на базі компонентної архітектури, де кожен об’єкт у грі може мати прикріплені до нього компоненти, такі як трансформація (Transform), фізичні властивості (Rigidbody) або користувацькі скрипти.

На рисунку 1.8 показано приклад налаштування компонентів для відображання інтерактивного персонажа, що використовується у грі.

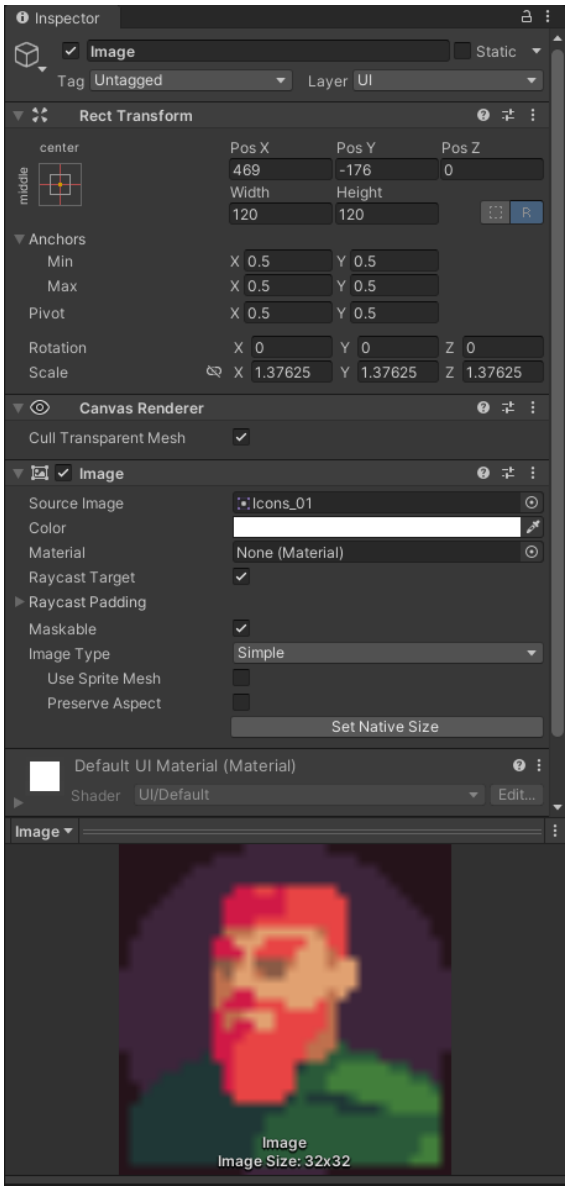

Рисунок 1.8 - Налаштування компонентів відображання персонажа в Unity

Інспектор (Inspector) дозволяє розробнику змінювати параметри об’єктів у реальному часі, наприклад, додавати зображення фону, змінювати поведінку персонажів або налаштовувати сценарії. Для інтерактивних ігор ця структура є ключовою, адже дозволяє модульно налаштовувати кожен елемент.

Unity пропонує потужні інструменти для розробки візуальних новел:

- Система UI: використовується для створення текстових блоків, кнопок вибору і діалогових вікон.
- Scriptable Objects: дозволяють зберігати сценарії, репліки, налаштування фонів і персонажів. Це дає змогу легко налаштовувати різні сцени гри без необхідності переписувати код.

- Ієрархія сцен: дозволяє змінювати локації гри і забезпечує послідовність у розповіді.

Unity підтримує інтеграцію мультимедіа для створення повноцінного ігрового досвіду:

- Зображення фонів: додаються як спрайти для відображення місць дії.
- Звукові ефекти: допомагають передати атмосферу гри, наприклад, звук лісу або шепіт персонажів.
- Музика: додає емоційну складову, супроводжуючи ключові моменти сюжету.
- Анімації: використовуються для оживлення персонажів, створення динамічних сцен чи візуалізації емоцій персонажів.

Для теми інтерактивних ігор, таких як візуальні новели, мультимедіа є важливим елементом, що допомагає створити занурення у світ гри.

Unity виділяється серед інших рушіїв завдяки своїй універсальності.

- Простота інтеграції діалогів, які є основою будь-якої візуальної новели.
- Підтримка системи вибору, яка дозволяє створювати розгалужені сценарії з багатьма кінцівками.
- Інструменти для тестування гри, зокрема для перевірки логіки вибору.

Visual Studio – це інтегроване середовище розробки (IDE), яке підтримує написання, тестування та налагодження коду на різних мовах програмування, включаючи C#.

Його основними перевагами є простота використання, наявність інструментів для автозаповнення коду (IntelliSense) та зручний інтерфейс для налагодження, який зображений на рисунку 1.9

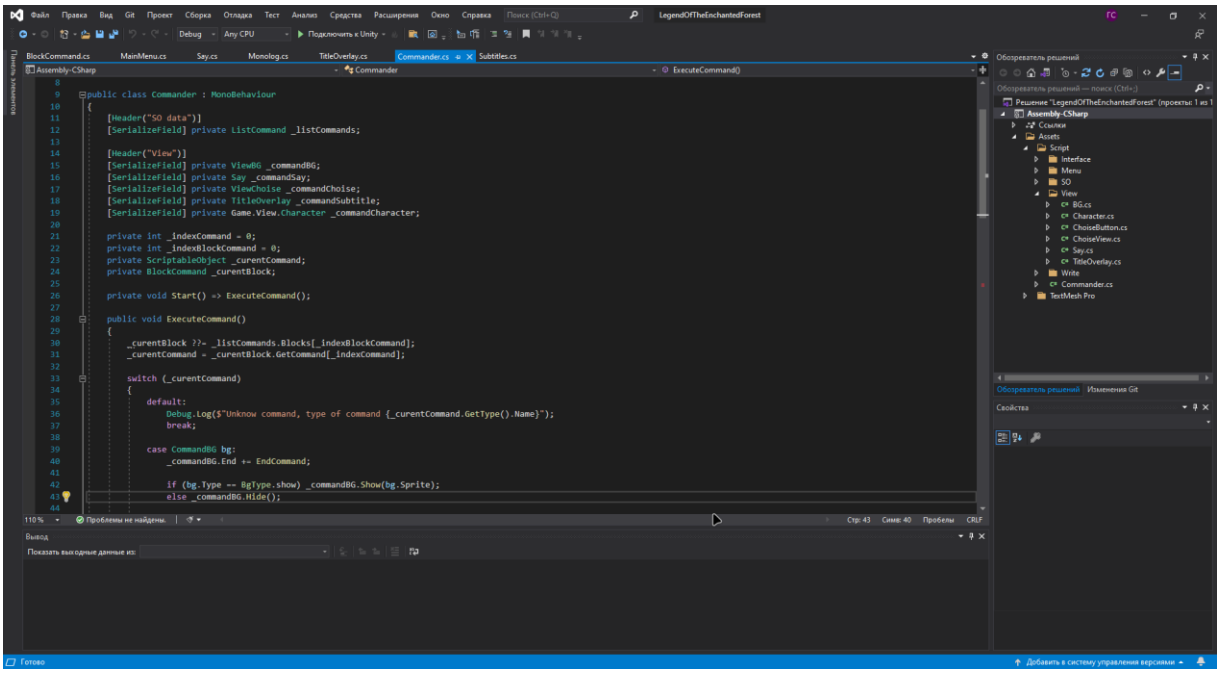

Рисунок 1.9 - Середовище розробки Microsoft Visual Studio

Редактор коду у Visual Studio надає такі можливості:

- Підсвічування синтаксису: допомагає швидко ідентифікувати помилки у коді.
- Автозаповнення IntelliSense: прискорює написання коду, пропонуючи варіанти методів, класів чи змінних.
- Форматування: автоматичне вирівнювання і організація коду.

Visual Studio має потужні інструменти для налагодження, що дозволяють аналізувати код у реальному часі. Наприклад, функція "breakpoints" (точки зупинки) допомагає знаходити помилки. Як зображено на рисунку 1.10, розробник може стежити за виконанням сценарію і змінювати параметри без зупинки програми.

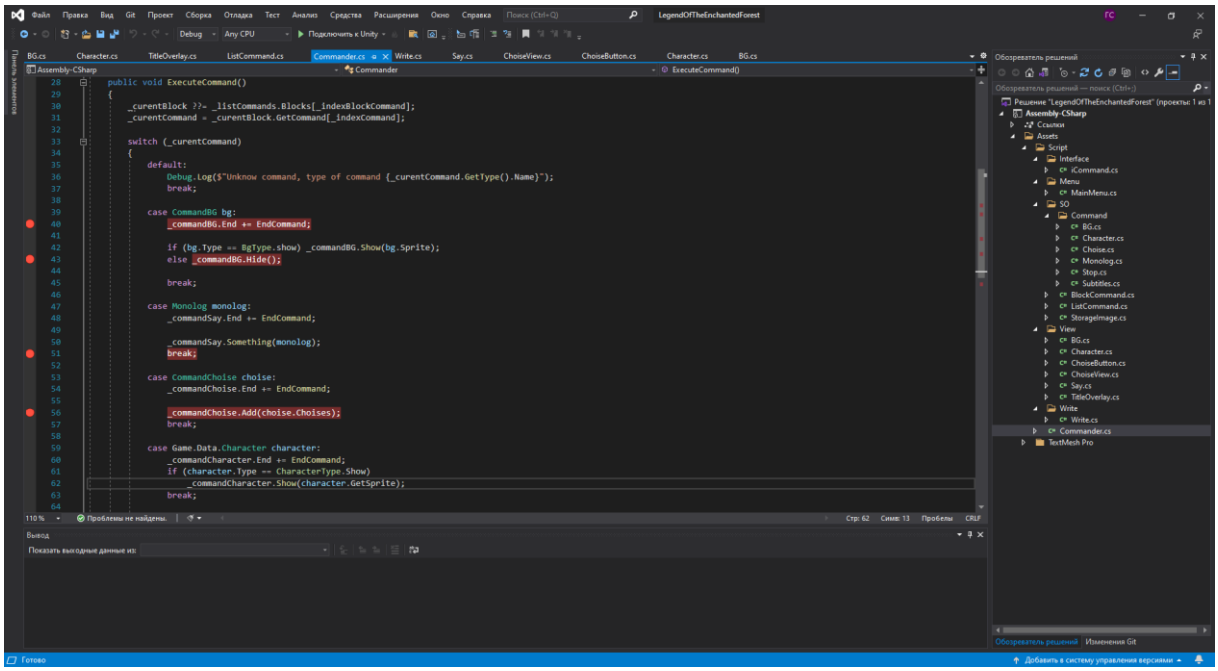

Рисунок 1.10 - Використання точок зупинки у Visual Studio

Visual Studio підтримує інтеграцію з Git прямо в середовищі розробки. Це дозволяє:

– Зберігати резервні копії проекту.

– Працювати в команді над одним проектом.

– Відстежувати зміни у коді та повертатися до попередніх версій.

Visual Studio підтримує встановлення додаткових плагінів, які покращують роботу з Unity та C#. Наприклад, існують розширення для оптимізації автозаповнення чи спрощення роботи з UI у Unity.

C# є основною мовою програмування для створення ігор в Unity, забезпечуючи високу продуктивність і широкий спектр можливостей. Завдяки синтаксису, який є одночасно простим для новачків і потужним для досвідчених розробників, C# дозволяє створювати інтерактивні ігрові механіки, зокрема, для візуальних новел.

Unity використовує C# для створення сценаріїв, які реалізують основну логіку гри. Наприклад, кожен об'єкт у грі може мати свій скрипт, що дозволяє керувати його поведінкою. Ці скрипти створюються і редагуються в інтегрованих

середовищах розробки, таких як Microsoft Visual Studio. На рисунку 1.11 показано приклад робочого середовища, в якому редагується скрипт на C#.

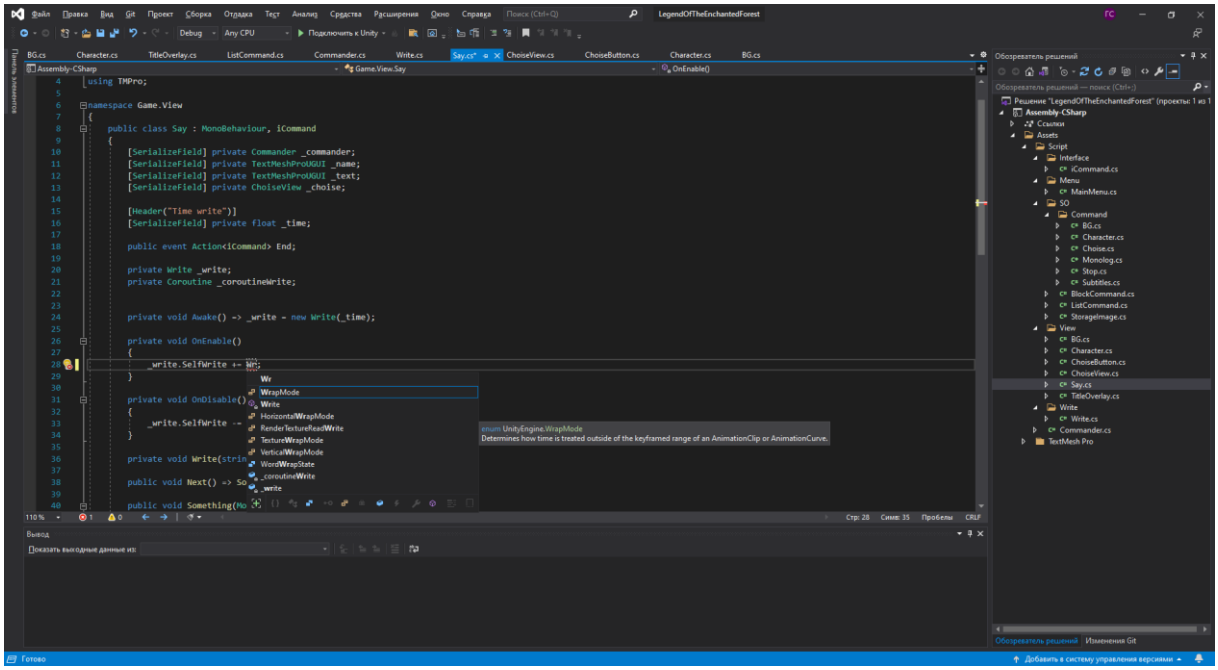

Рисунок 1.11 - Код на мові C# у середовищі Visual Studio

C# має вбудовані можливості для роботи з API Unity, що дозволяє легко взаємодіяти з об'єктами сцени. Наприклад, за допомогою сценаріїв можна змінювати позицію об'єктів, працювати з анімаціями або викликати функції при певних подіях, таких як натискання на екран. Це є критично важливим для створення інтерактивних діалогів, які є основною механікою візуальних новел.

Однією з важливих функцій C# є його здатність до реалізації складних систем логіки. Наприклад, для створення дерева діалогів використовуються класи і структури даних, які зберігають інформацію про вибір гравця. Ці вибори впливають на подальший розвиток сюжету, створюючи глибокий інтерактивний досвід. Як показано на рисунку 1.12, код на C# може реагувати на вибір гравця, змінюючи текст у діалоговому вікні або змінюючи фон залежно від рішення.

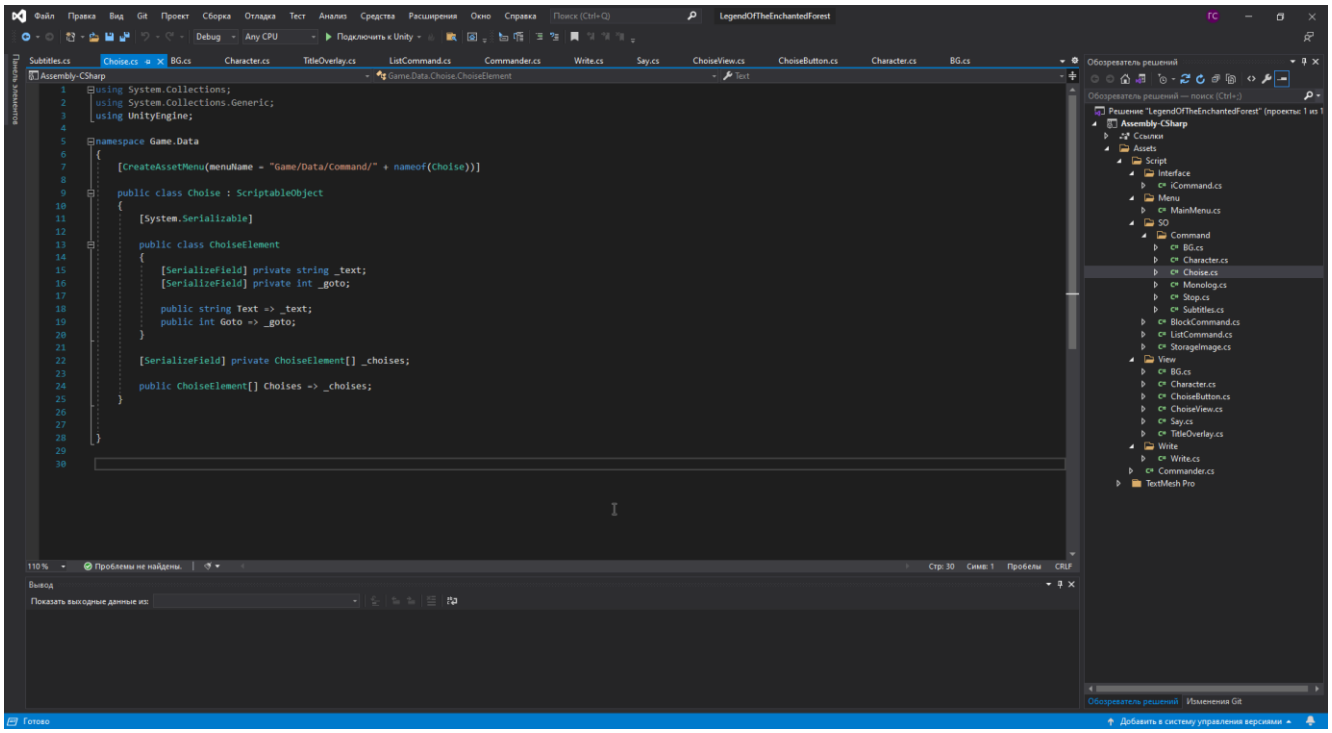

Рисунок 1.12 - Код для реалізації вибору в інтерактивних діалогах

C# також дозволяє інтегрувати анімації та елементи UI. Це забезпечує плавний перехід між сценами, ефекти натискання кнопок та взаємодію з іншими об'єктами гри. Завдяки підтримці подій і делегатів, C# дозволяє розробникам легко додавати динамічність до ігрового процесу, що є ключовим елементом у створенні сучасних візуальних новел.

1.4 Вимоги до апаратного та програмного забезпечення

Мінімальні вимоги до ПК:

- Процесор: Intel Pentium III 800 МГц або аналогічний;
- ОС: MS Windows 7;
- Оперативна пам’ять: 1 ГБ ОЗП;
- Графічний процесор: Intel HD Graphics або еквівалент із підтримкою DirectX 9.0;
- Вільне місце на жорсткому диску: 500 МБ.

Рекомендовані вимоги до ПК:

- Процесор: Intel Core i3 2.0 ГГц або вище;
- ОС: MS Windows 10;
- Оперативна пам’ять: 4 ГБ ОЗП;
- Графічний процесор: NVIDIA GeForce GTX 750 або аналогічний із підтримкою DirectX 11;
- Вільне місце на жорсткому диску: 1 ГБ.

2 ПРАКТИЧНА ЧАСТИНА

2.1 Створення та налагодження програми

На початку роботи над проєктом були визначені основні вимоги до системи, після чого її розробка виконувалася поетапно, у вигляді послідовних версій. Кожна версія представляла собою завершений і працездатний продукт, який поступово доповнювався новими функціональними можливостями.

Перша версія містила базові функції, необхідні для роботи програми, тоді як наступні версії розширювали її функціональність, поки не було отримано повну систему. Інкрементальна модель життєвого циклу була обрана для розробки через її зручність у реалізації ігрових систем, де є чітке уявлення про кінцевий результат, яка зображена на рисунку 2.1.

Ця модель дозволяє побачити результати роботи вже на ранніх етапах розробки. Після створення першої версії можна було коригувати вимоги, вдосконалювати продукт або запропонувати його подальшу оптимізацію.

Інкрементальна модель об’єднує елементи водоспадного підходу з ітераційною розробкою. Кожна лінійна послідовність створює інкремент — завершений функціональний компонент програмного забезпечення.

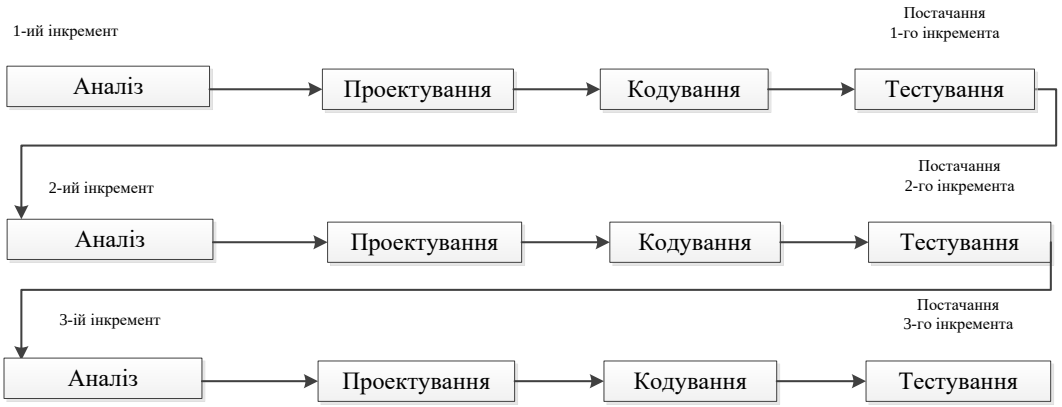

Рисунок 2.1 - Інкрементальна модель

Розробка програми розпочалася зі створення нового проєкту в Unity Hub. Для цього було обрано шаблон 2D (Built-In Render Pipeline), який є оптимальним для візуальних новел. На рисунку 2.2 показано процес створення проєкту в Unity Hub.

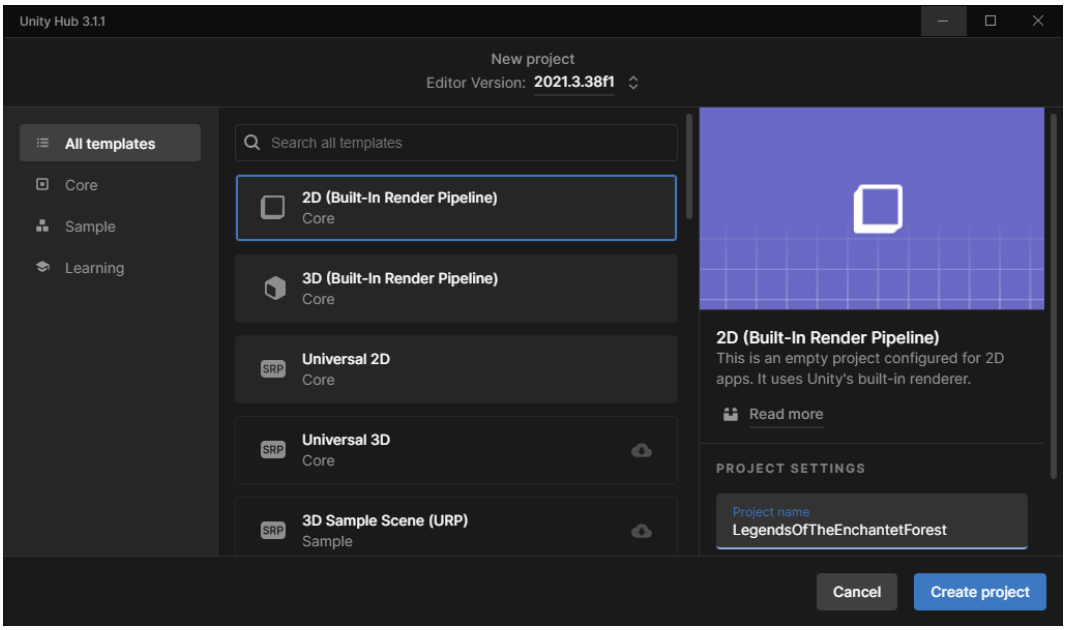

Рисунок 2.2 - Створення проєкту в Unity Hub

Після створення проєкту було організовано файлову структуру, яка включала такі папки:

– Prefabs - для готових об’єктів;
– Scenes - для сценаріїв гри;
– Scripts - для написання коду;
– Sprites - для графічних ресурсів;
– TextMeshPro - для налаштування текстових компонентів Unity;

Ця структура забезпечує зручне управління ресурсами проєкту та впорядкованість у процесі розробки. На рисунку 2.3 показано створену структуру папок.

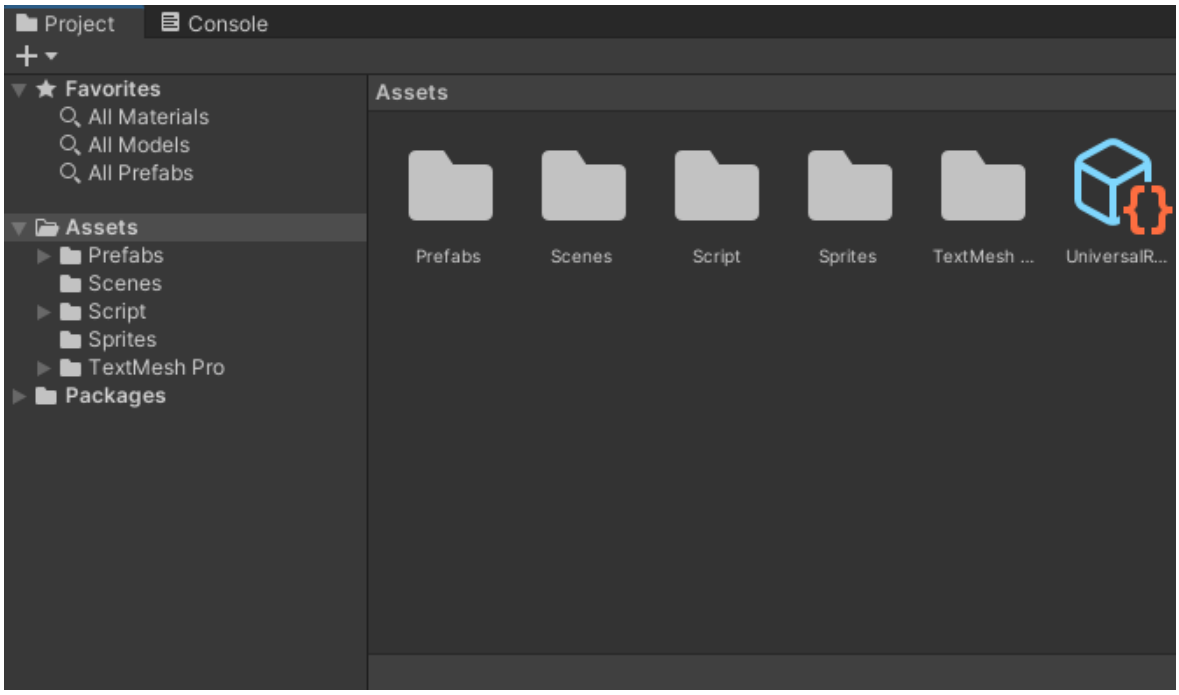

Рисунок 2.3 - Структура папок проєкту

Далі було виконано налаштування Build Settings, де до списку сцен додано MainMenu і Chapter1. Ці сцени відповідають основному меню гри та першому розділу. На рисунку 2.4 показано налаштування Build Settings.

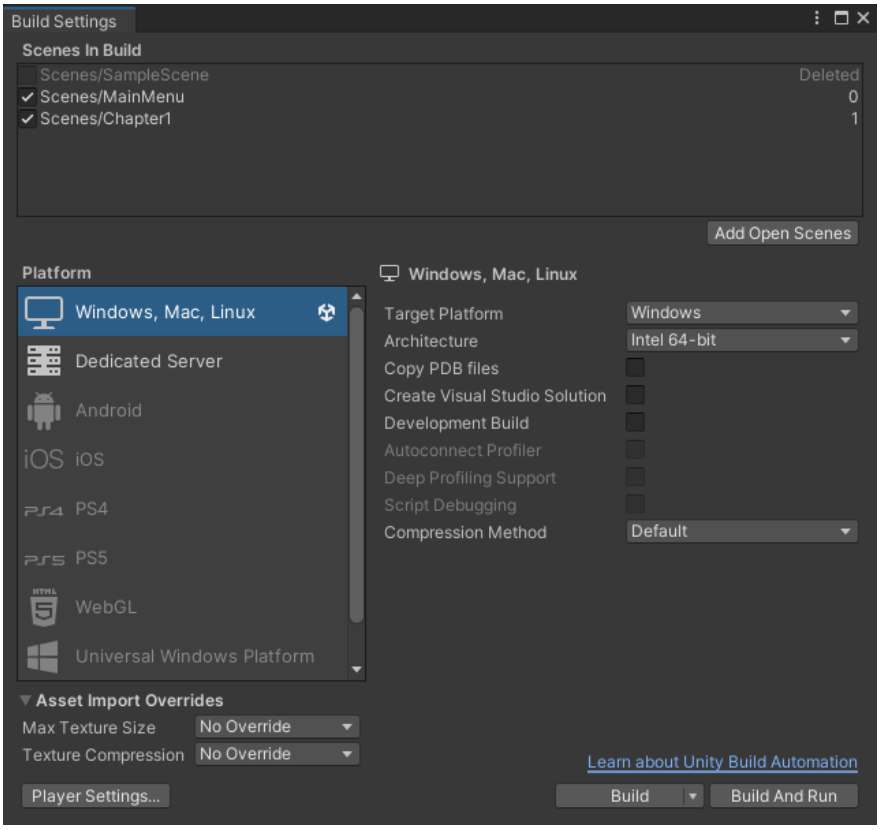

Рисунок 2.4 - Налаштування Build Settings

У проєкті створено дві основні сцени: MainMenu та Chapter1. Сцена MainMenu відповідає за відображення основного меню гри. Вона включає такі об'єкти:

- Canvas – основний об'єкт для управління інтерфейсом сцени. До нього додано компонент Main Menu (Script), що відповідає за взаємодію з кнопками.
- Кнопка Start – запускає перехід до першого розділу гри.
- Кнопка Quit – завершує роботу гри.

На рисунку 2.5 показано ієрархію сцени MainMenu.

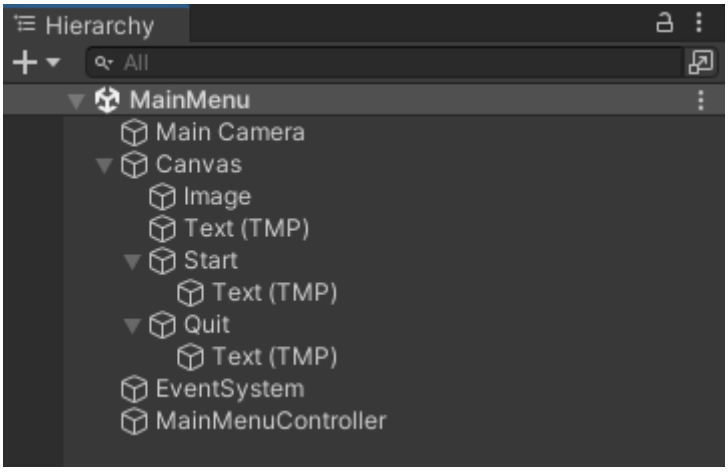

Рисунок 2.5 - Ієрархія сцени MainMenu

Сцена Chapter1 є першим розділом гри та включає наступні елементи:

- Canvas (Say) – об'єкт для відображення діалогів між персонажами. До нього додано компонент Say (Script), який містить посилання на зовнішні файли з текстами діалогів і забезпечує їх динамічне відображення на екрані.
- Titles – панель для відображення заголовку розділу.
- Background – об'єкт із фоновим зображенням, що задає атмосферу сцени.
- Character – спрайт персонажа, який змінюється залежно від розвитку сюжету.

Ієрархію сцени Chapter1 наведено на рисунку 2.6.

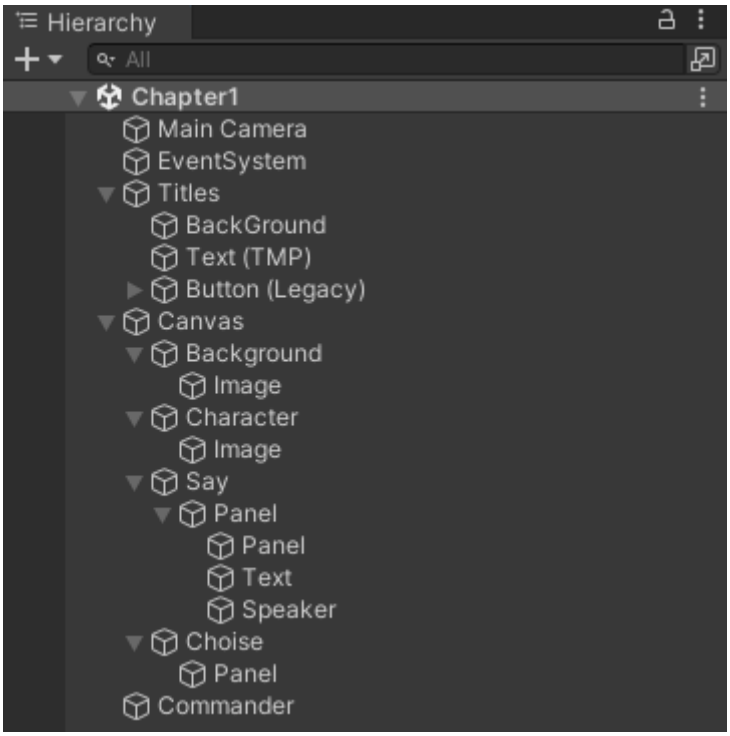

Рисунок 2.6 - Ієрархія сцени Chapter1

Також у меню Preferences, у розділі External Tools, було обрано Visual Studio як основний редактор коду. Це забезпечило інтеграцію середовища для зручного редагування скриптів. На рисунку 2.7 зображено налаштування External Tools у Unity.

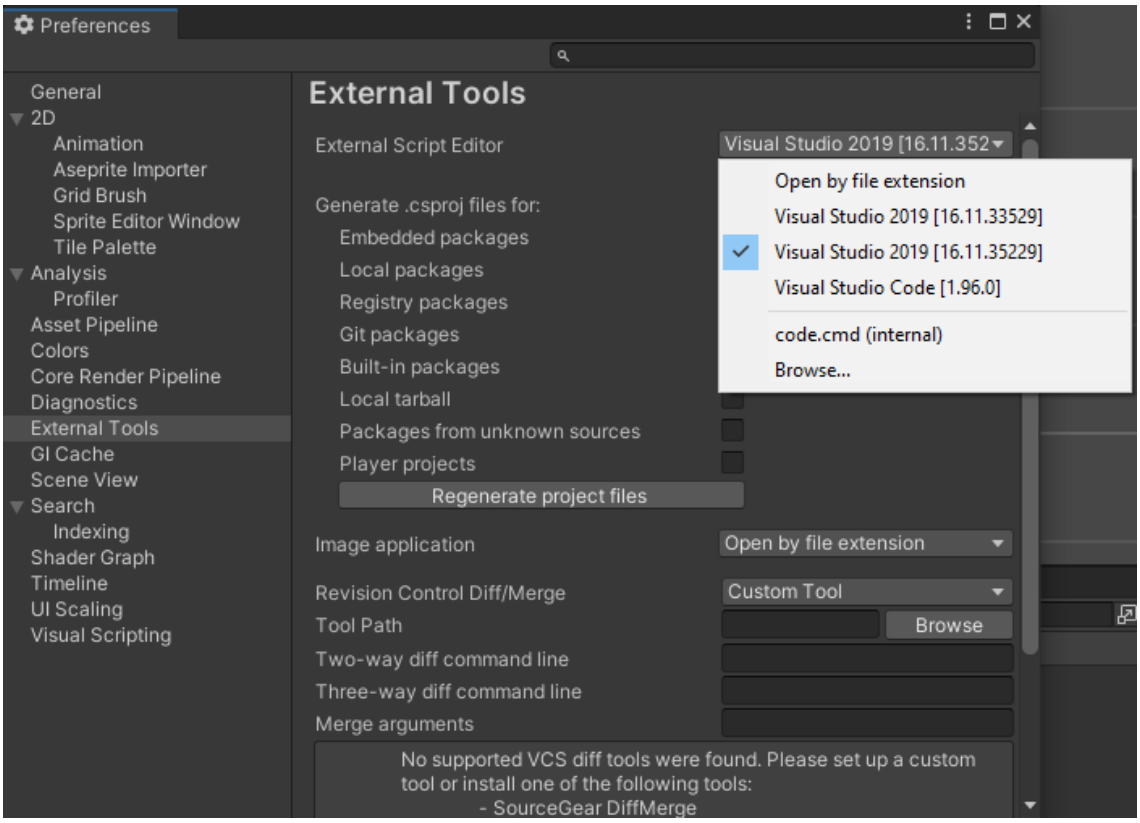

Рисунок 2.7 - Налаштування External Tools у Unity

Для зручності додавання діалогів у гру був створений текстовий документ, у якому прописано всі репліки персонажів, описи сцен і варіанти розвитку сюжету. Це дозволяє ефективно організувати сценарії та швидко переносити їх у Unity. У текстовому документі структура діалогів включає:

- Опис сцени: фон, атмосфера, початкові події.
- Слова автора: текст, що описує дії або події, які відбуваються.
- Репліки персонажів: кожна репліка супроводжується емоційним описом (наприклад, «зловісно», «з лукавою усмішкою»).
- Варіанти вибору гравця: вказано можливі дії або відповіді, які впливають на розвиток сюжету.

Приклад текстового документа виглядає наступним чином:

Тінь

- Обери мене, і я відкрию правду, якої ти боїшся.

Лісовик

- Йди зі мною, і я покажу тобі твої корені.

Фея

– Мій шлях найяскравіший. Хіба ти не хочеш відчути свободу?

Автор

– Кому ти довіриш свою долю?

Вибір: Тінь

Автор

– Ти простягаєш руку до Тіні. Її погляд наповнюється задоволенням. Вона проводить тебе до старого храму, де всі твої забуті спогади постають перед тобою, мов шматки розбитого дзеркала.

Тінь

– Тепер ти знаєш правду. Але чи зможеш ти жити з цим?

Автор

– Ти дізнаєшся, що втратив пам'ять через угоду з Тінню, яка забрала частину твоєї душі, аби врятувати когось дорогого. Тепер ти вільний, але частинка темряви назавжди залишиться в тобі.

Титри:

– Ти повернув свою пам'ять, але разом із нею — тягар минулого. Тепер ти живеш із правдою, яка переслідує тебе у кожному кроці.

Вибір: Лісовик

Автор

– Ти довіряєш Лісовику. Він веде тебе до величезного дерева, корені якого сягають глибоко в землю. Ти відчуваєш, як щось знайоме кличе тебе.

Лісовик

– Твоя душа завжди була частиною цього місця. Тепер ти вдома.

Автор

- Ти згадуєш, що колись був захисником лісу, який оберігав це місце від небезпеки. Втрата пам’яті стала наслідком давньої битви, але тепер ти повернувся до свого обов’язку.

Титри:

- Ти став частиною лісу, його захисником і його голосом. Природа тепер твоє життя, і ліс завжди буде вдячний тобі за твою вірність.

2.2 Опис програми та її алгоритмів

Діаграма прецедентів. Діаграма прецедентів — це модель, що описує функції системи та середовище, у якому виконуються основні процеси. На рисунку 2.8 представлено функції системи у вигляді діаграми прецедентів із одним актором - користувачем.

Користувач може:

- почати гру, обравши варіант із головного меню;
- вийти з гри, обравши варіант із головного меню;
- взаємодіяти з персонажами через систему діалогів;
- переглядати альтернативні сюжетні лінії, приймаючи різні рішення;
- завершити гру, отримавши одну з трьох можливих кінцівок.

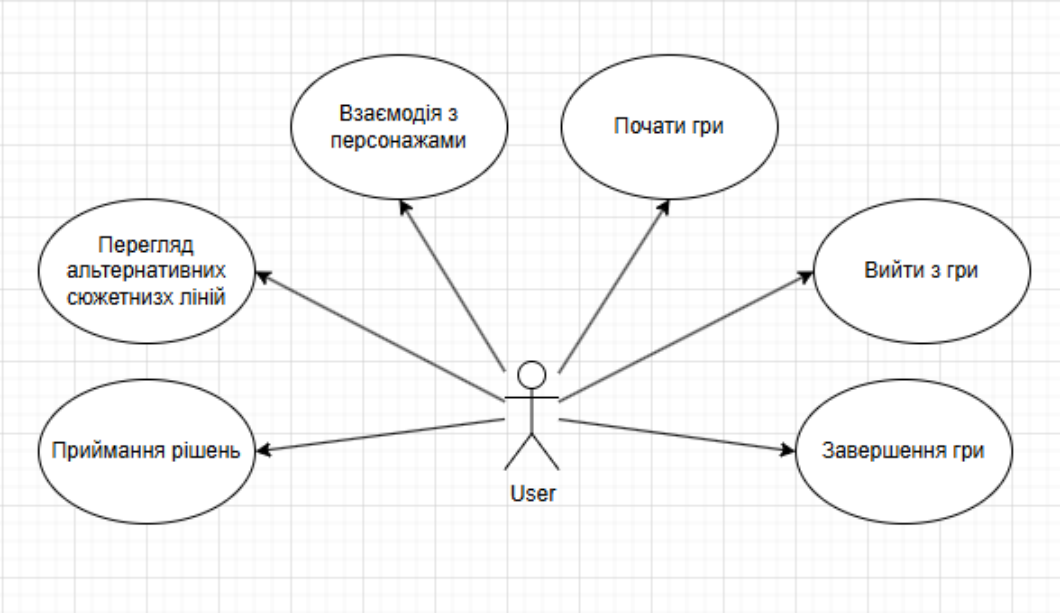

Рисунок 2.8 - Діаграма прецедентів для користувача

Діаграма класів. Діаграма класів відображає структуру системи, включаючи класи та зв’язки між ними. На рисунку 2.9 представлено діаграму класів програми.

1. Interface:
   - iCommand.cs — інтерфейс для команд, що задає базову структуру для виконання різних дій у системі.
2. Menu:
   - MainMenu.cs — відповідає за логіку головного меню гри, зокрема обробку виборів користувача.
3. SO (Scriptable Objects):
   - Command:
     - BG.cs — відображення фону сцени.
     - Character.cs — керування персонажами (позиції, анімації тощо).
     - Choise.cs — обробка вибору користувача у грі.
     - Monolog.cs — відображення монологів персонажів.
     - Stop.cs — реалізація паузи у грі або зупинки певної команди.
     - Subtitles.cs — виведення субтитрів на екран.
   - BlockCommand.cs — об'єднання кількох команд для послідовного виконання.

- ListCommand.cs — керування списком команд у грі.
- StorageImage.cs — збереження та відображення зображень у грі.

4. View:
   - BG.cs — класс збереження фону.
   - Character.cs — класс збереження зображення персонажів.
   - ChoiseButton.cs — класс збереження інтерактивних елементів.
   - ChoiseView.cs — класс збереження зображення виборів.
   - Say.cs — керування діалогами та їх відображення.
   - TitleOverlay.cs — показ заголовків або інших оверлейних елементів.
5. Write:
   - Write.cs — запис і виведення текстової інформації у процесі гри.
6. Commander.cs — основний клас, що координує виконання команд у грі.

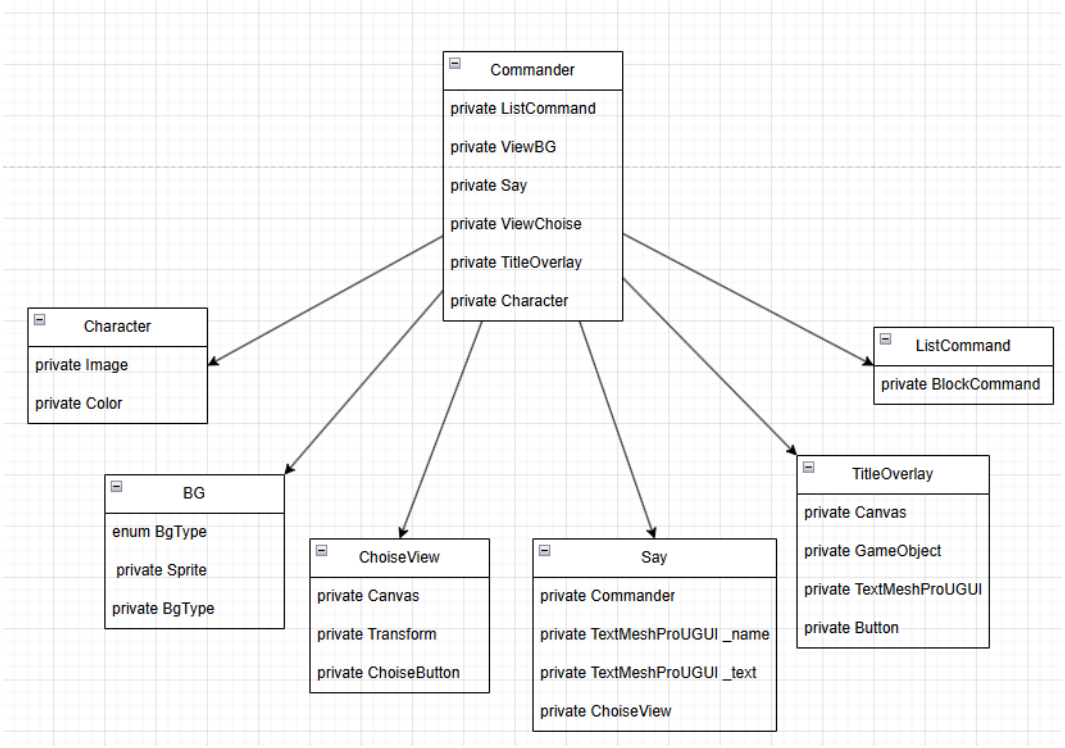

Рисунок 2.9 - Діаграма класів проекту

Алгоритм роботи системи

Після запуску гри користувач потрапляє в головне меню, де може обрати один із доступних варіантів: розпочати гру або завершити програму. При виборі пункту «Розпочати гру» гравець переноситься до першої сцени, де починається сюжет із пробудження героя в лісі.

У процесі гри:

Гравець взаємодіє з персонажами через систему діалогів, обираючи відповіді або дії.

Кожен вибір гравця впливає на подальший розвиток сюжету, активуючи відповідні сценарії та сцени.

Гра завершується однією з трьох кінцівок залежно від дій гравця.

На рисунку 2.10 представлений алгоритм роботи програми.

Рисунок 2.10 - Алгоритм роботи програми

2.3 Інструкція програміста

Для перегляду та редагування проєкту потрібно використовувати середовище розробки Unity Editor. Проєкт відкривається за допомогою файлу сцени або налаштувань у Assets. Для роботи в Unity слід завантажити Unity Hub та встановити необхідну версію Unity (наприклад, 2021.3.38f1, як у цьому прикладі).

Щоб створити новий проєкт або відкрити вже існуючий, у Unity Hub необхідно обрати вкладку Projects, після чого натиснути кнопку Open та вибрати шлях до файлів проєкту, зображено на рисунку 2.11. Обов'язково переконайтеся, що версія Unity відповідає версії, з якою проєкт розроблявся.

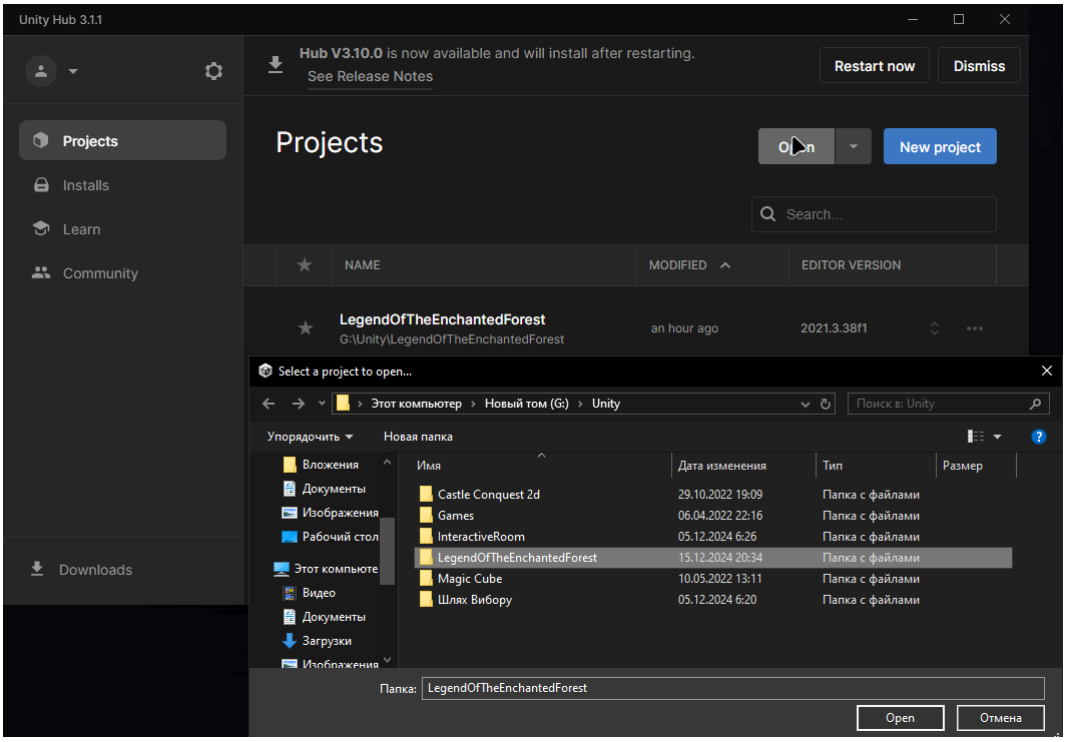

Рисунок 2.11 - Відкриття існуючого проєкту

Для налаштування редактора Unity необхідно:

- Відкрити Unity Hub та встановити шаблон 2D/3D для створення проєкту.
- Переконатися, що середовищі налаштовані такі компоненти, як TextMeshPro для роботи з текстом і Sprite Editor для редагування спрайтів.

– Організувати проєкт у директорії Assets з чіткою структурою (наприклад: Scripts, Prefabs, Scenes, Sprites тощо).

Для роботи з проєктом у Unity використовується механізм [CreateAssetMenu], який дозволяє створювати різні Scriptable Object для збереження даних та налаштувань гри. У вашому проєкті програміст може обирати типи об'єктів, такі як BG, BlockCommand, Character, Choise, ListCommand, Stop, Storage Image, Subtitles, що дає змогу зручно будувати сценарії.

Процес створення об'єктів через ПКМ у панелі Project, який зображено на рисунку 2.12:

Create → Game → Data → Command відкриває список доступних опцій. Після вибору одного з типів (наприклад, BlockCommand) у панелі Inspector можна налаштувати об'єкт відповідно до його функціоналу.

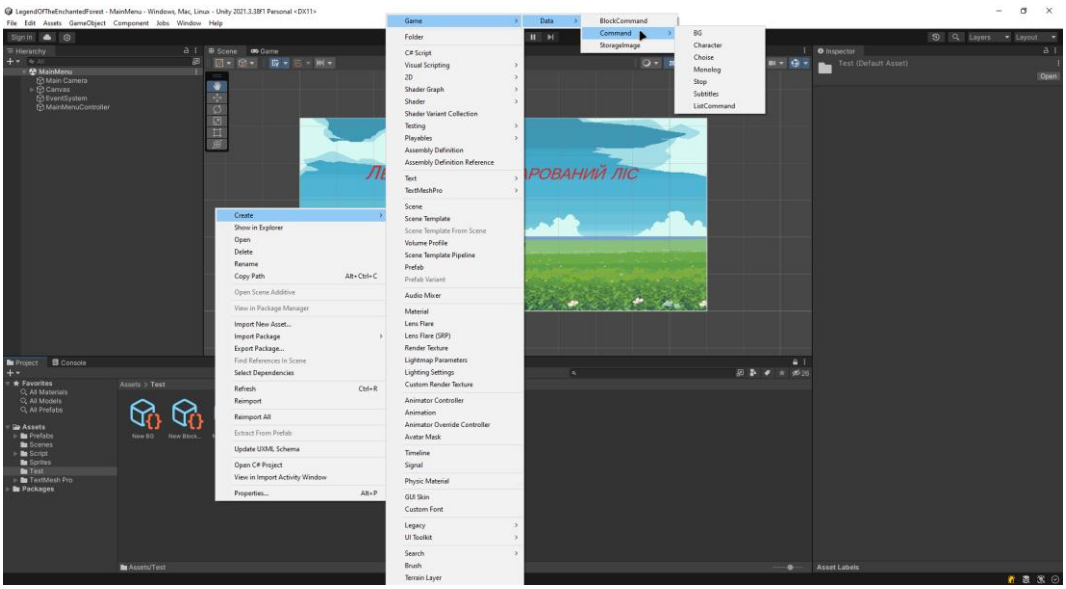

Рисунок 2.12 - Створення нового об’єкта

BG (Background). Цей тип об’єкта використовується для завантаження фону сцени. Після створення BG у полі Sprite необхідно завантажити зображення заднього фону, яке буде відображатися під час гри. У вікні Inspector (рис. 2.13) доступні такі поля:

– Sprite – у це поле додається зображення, яке буде використовуватися як фон сцени. Для цього необхідно вибрати файл зі списку наявних ресурсів або імпортувати нове зображення.
– Type – вибір способу відображення фону. Наприклад, параметр Show вказує на те, що фон повинен бути активованим і показаним одразу після виконання команди.

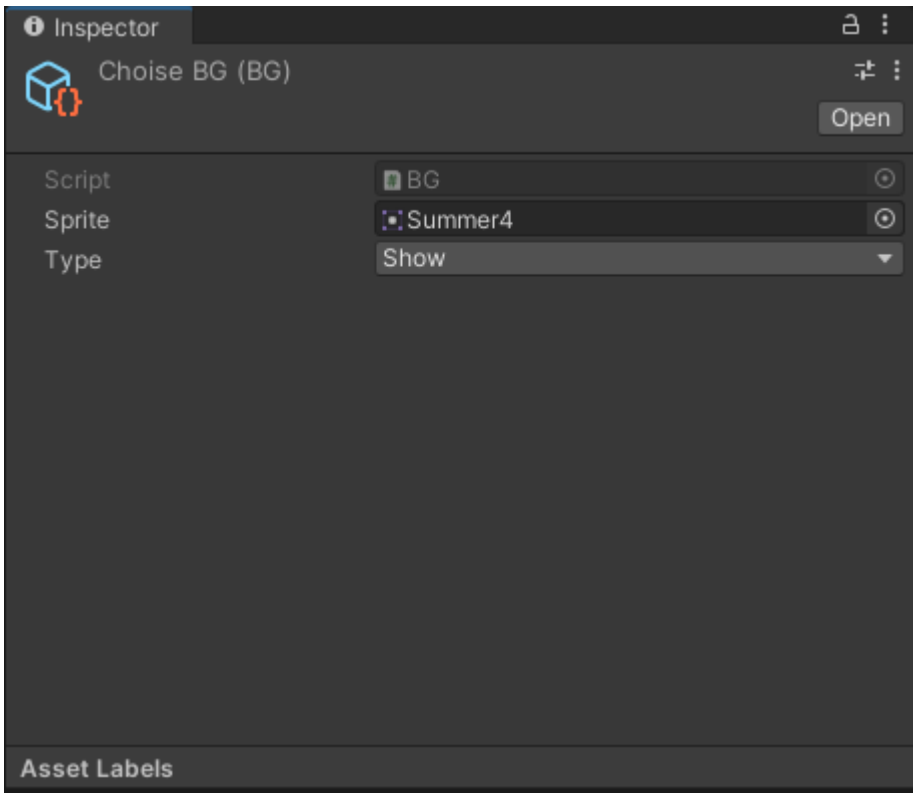

Рисунок 2.13 – Приклад заповненого меню BlockCommand

BlockCommand. Цей об’єкт використовується як контейнер для збереження списку команд, які виконуються послідовно. У вікні Inspector (рис. 2.14) доступні такі поля:

– Command - список команд, які додаються до цього блоку. Наприклад, це можуть бути Subtitles (текст діалогу), BG (фон) або Stop (команда зупинки). Для додавання елементів натискається кнопка +, а для видалення -.
– Element - кожен елемент відповідає окремій команді. Вибрані команди додаються до сценарію і виконуються по черзі, формуючи логіку гри.

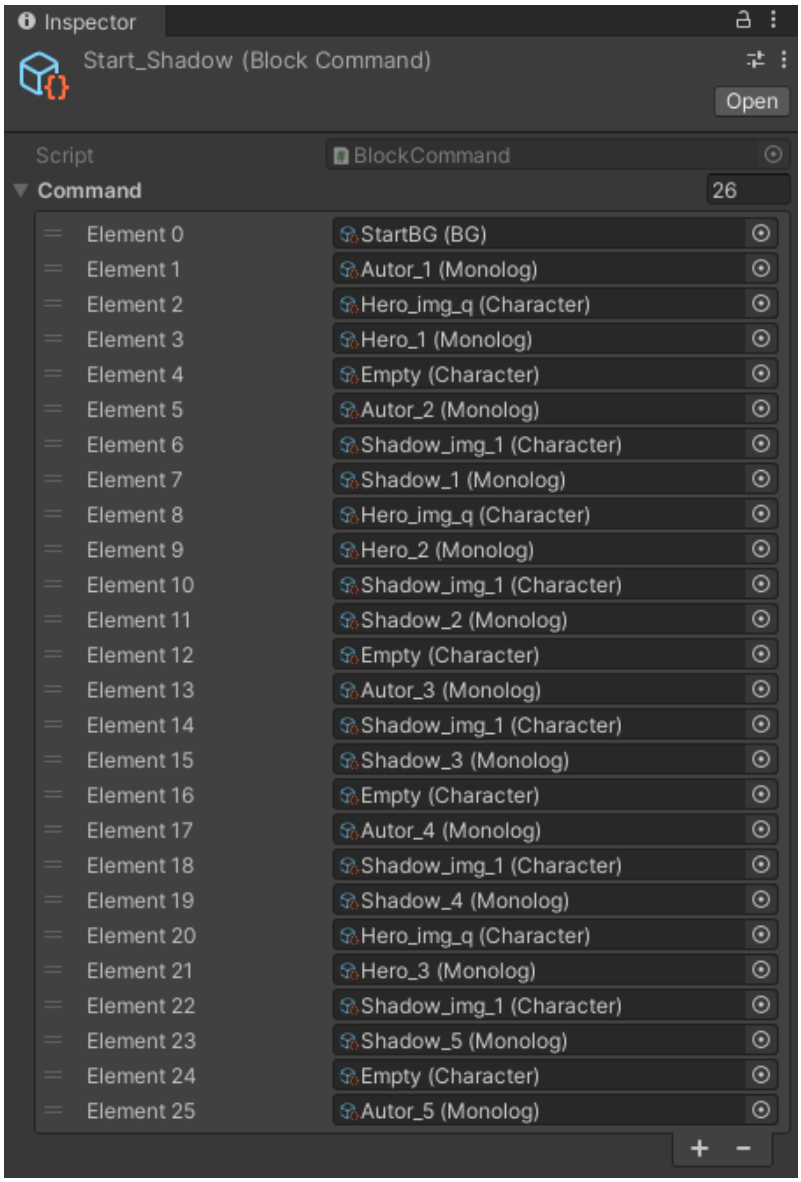

Рисунок 2.14 - Приклад заповненого меню BlockCommand

Character. Цей об'єкт відповідає за зображення персонажів у грі. У вікні Inspector(рис. 2.15) доступні такі поля:

- Storage – вибір місця збереження персонажа, з якої буде завантаження зображення персонажу.
- Name – ім'я персонажа, яка відповідає імені у місце збереження.
- Number – номер спрайту у списку у місці збереження.

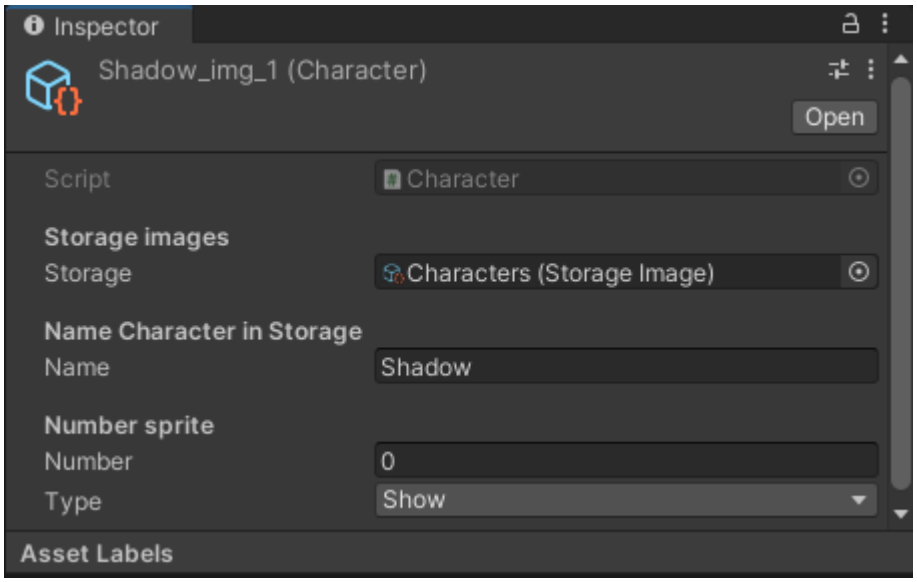

Рисунок 2.15 - Приклад заповненого меню Character

Choise. Об'єкт Choice створює варіанти вибору в діалогах гри. У вікні Inspector(рис. 2.16) доступні такі поля:

- Choise – список вибору, який додається до цього блоку. Для додавання елементів натискається кнопка +, а для видалення -.
- Text – текст вибору, який гравець бачить на екрані.
- Goto – посилання на об’єкт BlockCommand, що виконується після вибору гравцем конкретного варіанта.

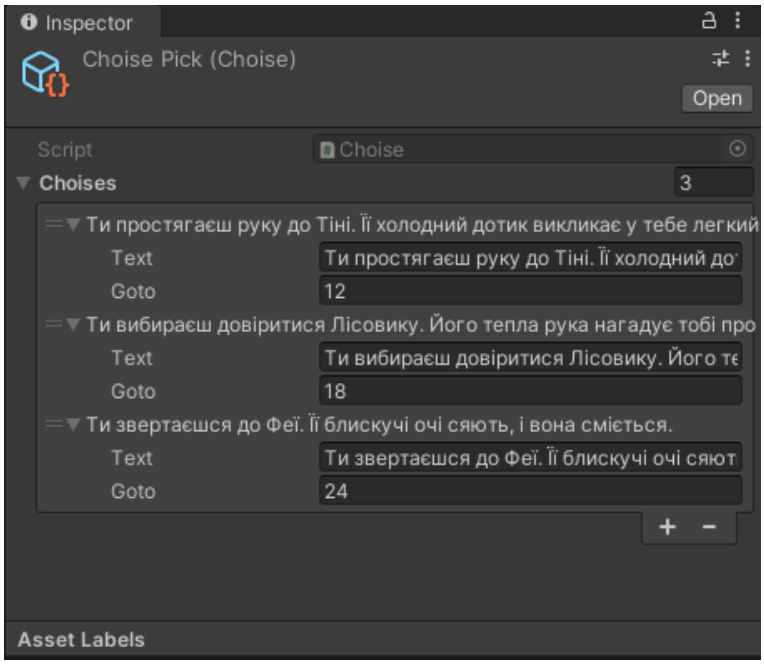

Рисунок 2.16 - Приклад заповненого меню Choise

ListCommand. Цей об'єкт містить список команд для виконання. Він схожий на BlockCommand, але використовується для групування великих наборів команд.

У вікні Inspector(рис. 2.17) доступні такі поля:

– Choise – список блокыв, який додається до листа команд. Для додавання елементів натискається кнопка +, а для видалення -.
– Element – список команд, які виконаються одна за одною. У це поле додаються посилання на Block Command.

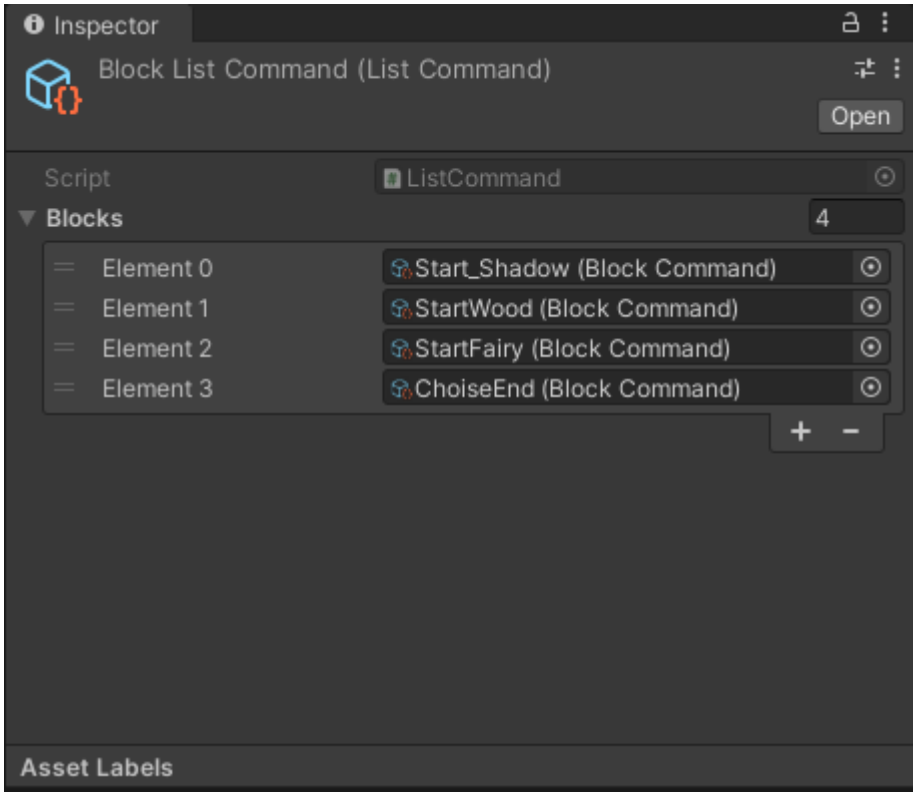

Рисунок 2.17 - Приклад заповненого меню ListCommand

Stop. Об'єкт Stop використовується для зупинки виконання команд у сценарії гри. Це може бути корисно для додавання паузи між діями або для розділення різних частин сценарію. Поля для цього об'єкта відсутні, він служить лише як команда зупинки у списку команд(рис. 2.18).

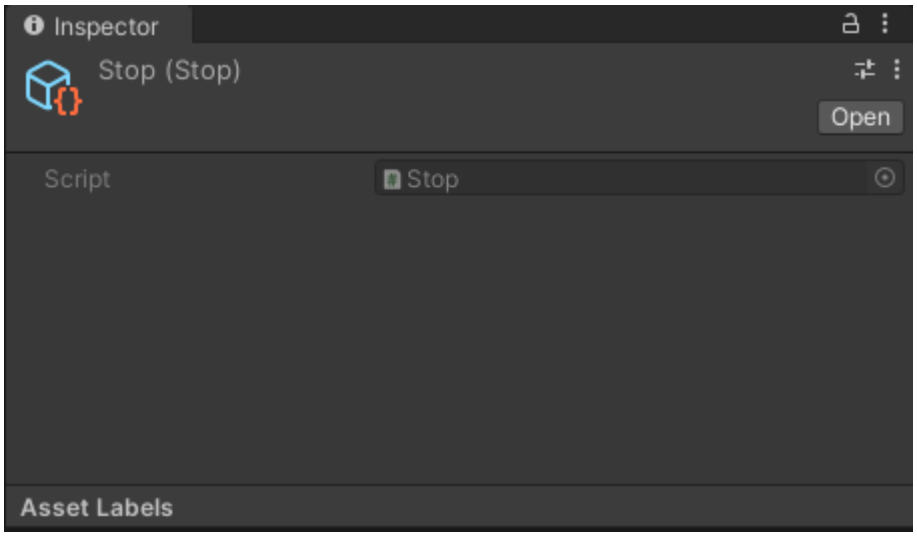

Рисунок 2.18 - Меню Stop

Storage Image. Об'єкт Storage Image використовується для збереження спрайтів персонажів, які можуть використовуватися у різних сценах гри. Кожен елемент цього об'єкта включає Ім'я персонажа та список спрайтів, які відображають різні емоції чи стани персонажа.

Як зображено на рисунку 2.19, об'єкт має функціонал для додавання нових елементів за допомогою кнопок «+» (для створення нового запису) та «-» (для видалення запису).

Поля для кожного елемента:

- Name – Ім'я персонажа, яке дозволяє пов'язати спрайти з певним героєм.
- Sprites – Список зображень (спрайтів), що додаються під певними номерами. Кожен номер відповідає конкретному стану або емоції персонажа.

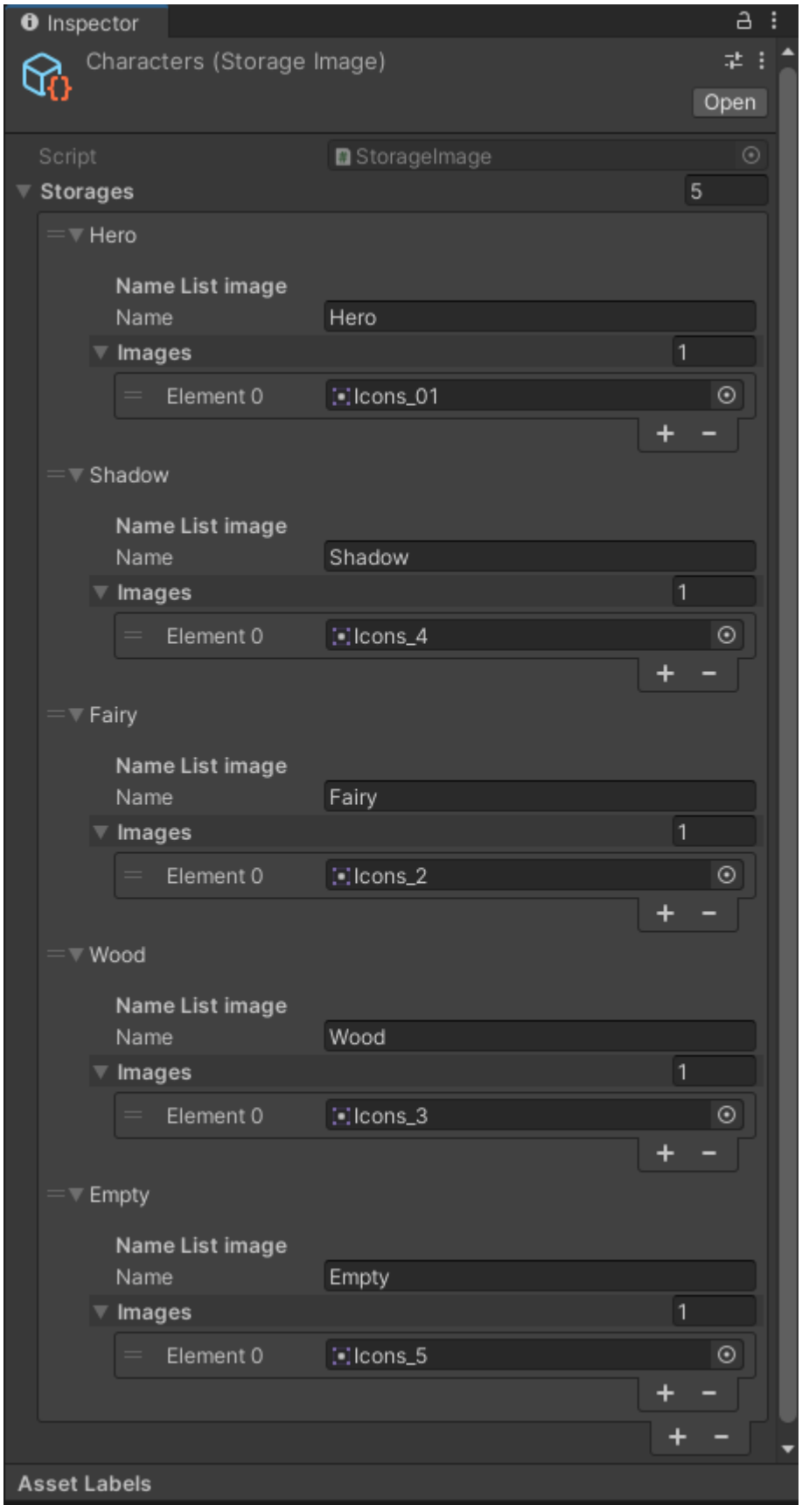

Рисунок 2.19 - Приклад заповненого меню Storage Image

Subtitles. Цей об'єкт відповідає за виведення текстових титрів на екран. Програміст створює Subtitles, вводить текст, який потрібно показати, і додає цей об'єкт у BlockCommand для послідовного відображення діалогів або повідомлень у грі. У вікні Inspector(рис. 2.20) доступні такі поля:

– Text – текст, який буде відображатися на екрані.

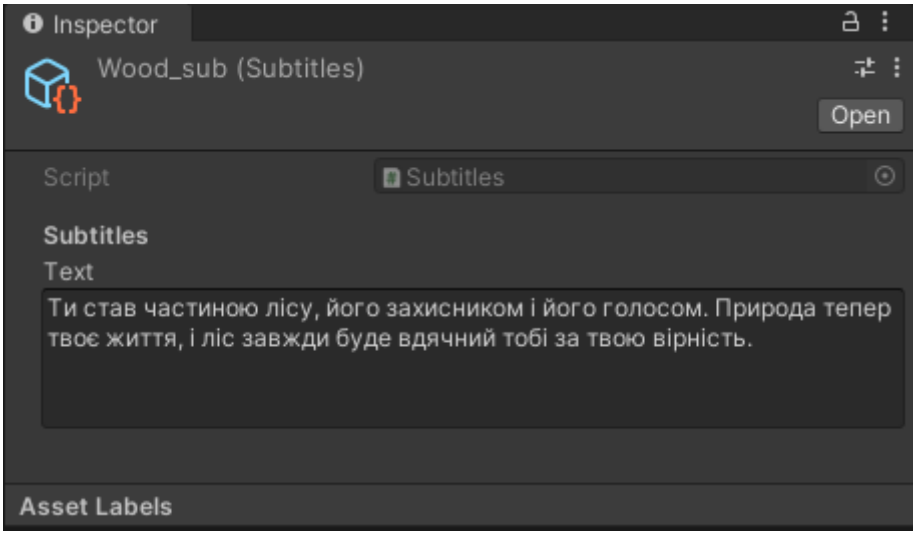

Рисунок 2.20 - Заповнення меню Subtitles

Організація проєкту

Усі створені об'єкти слід зберігати у відповідних папках у директорії Assets. Наприклад:

– Sprites – для графічних ресурсів.
– Scripts – для зберігання коду.
– Scenes – для рівнів гри.

Панель Inspector використовується для редагування параметрів кожного об'єкта. Створення та редагування об'єктів відбувається через контекстне меню, а налаштування списків або параметрів об'єкта через Inspector.

Ця інструкція дозволяє програмісту легко налаштовувати елементи гри, створювати інтерактивні сценарії та забезпечувати гнучке керування ресурсами.

2.4 Інструкція оператора

Після запуску гри відкривається головне меню, яке зображено на рисунку 2.21. У головному меню є декілька доступних кнопок, які забезпечують користувачеві можливість навігації по грі:

– «Почати гру» – запуск основного ігрового процесу.

– «Вихід» – завершення роботи гри.

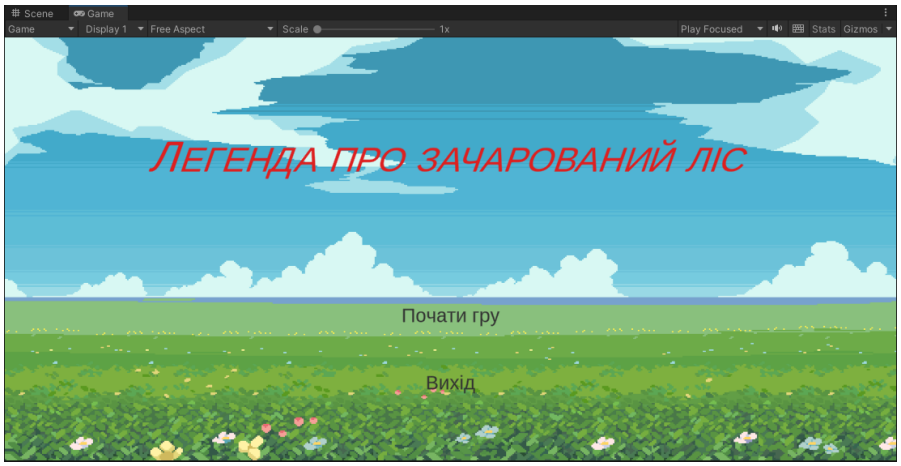

Рисунок 2.21 - Головне меню гри

Після натискання на кнопку «Почати гру», користувач потрапляє до першої сцени гри – знайомство з головним героєм. На екрані відображається текст вступу та діалоги, як показано на рисунку 2.22. Гравець взаємодіє з грою, читаючи текстові повідомлення та обираючи відповіді, що впливають на подальший розвиток сюжету.

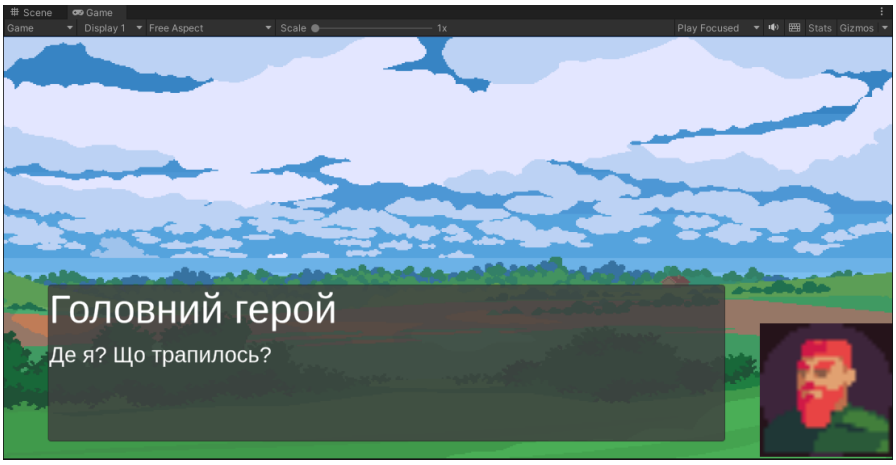

Рисунок 2.22 - Вступна сцена гри

Взаємодія з діалогами

Основна взаємодія користувача з грою відбувається через діалоги персонажів. У нижній частині екрана знаходиться текстове вікно, де відображаються репліки персонажів та описи подій (рисунок 2.13). Для переходу до наступної фрази гравець повинен натиснути на текстове вікно.

Таким чином, поступово розкривається сюжет гри, а також відбувається знайомство з персонажами, їхніми мотивами та вибір подальших дій. Приклад взаємодії з діалогами наведено на рисунку 2.23.

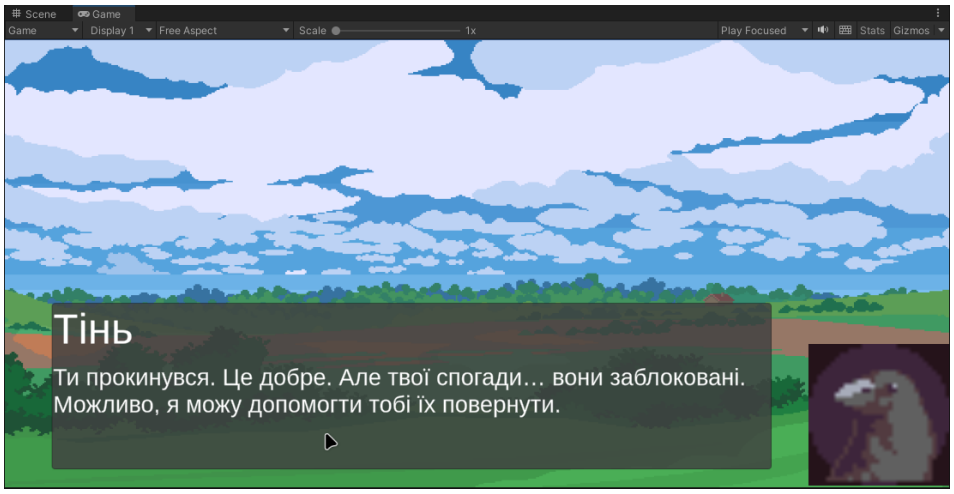

Рисунок 2.23 - Взаємодія з діалогами

На певних етапах гри гравець повинен зробити вибір відповіді, який визначає подальший розвиток сюжету. Варіанти відповідей відображаються у нижній частині екрана у вигляді кнопок, як показано на рисунку 2.24. Кожен обраний варіант може призвести до зміни діалогу, розвитку подій або вплинути на кінцівку гри.

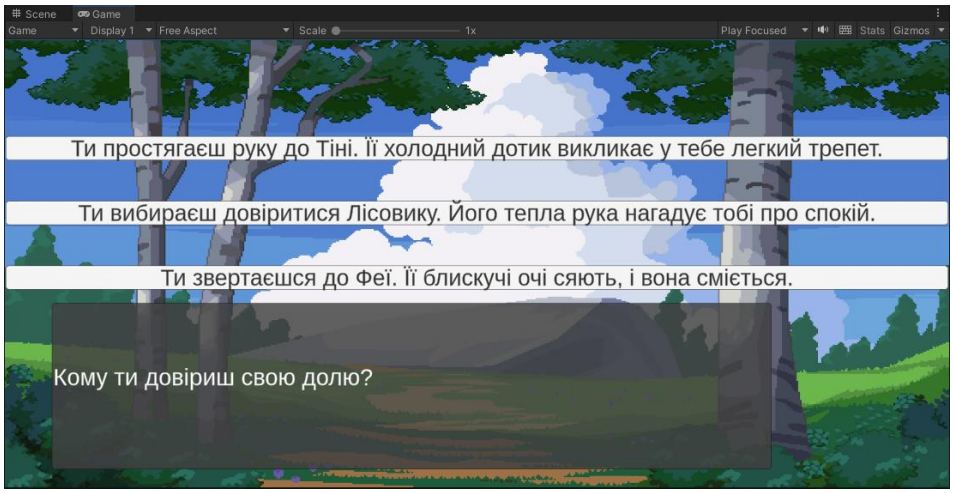

Рисунок 2.24 - Вибір відповіді у діалозі

Після завершення основного сюжету гравець отримує екран підсумкового результату, на якому відображається пройдене закінчення сюжету (рисунок 2.25). У залежності від виборів, зроблених під час гри, гравець може побачити одне з трьох можливих закінчень.

На цьому екрані також є кнопка «Вийти до головного меню», яка дозволяє гравцю повернутися на головний екран гри для повторного проходження або завершення роботи.

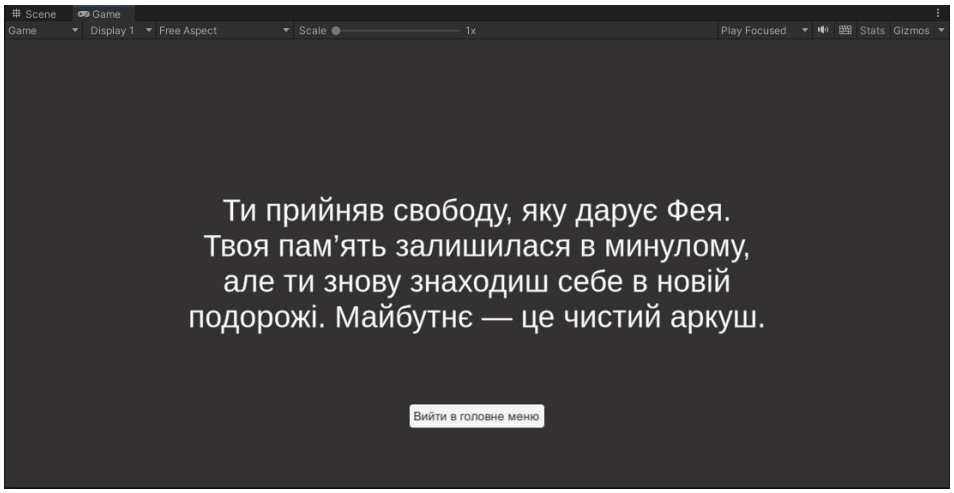

Рисунок 2.25 - Підсумкове закінчення гри

ВИСНОВКИ

Дипломна робота на тему «Дослідження інтерактивних наративів у іграх. Інтерактивна гра на основі C#» була проведена з метою аналізу інтерактивних наративів у відеоіграх, їхнього впливу на емоційний стан гравця, а також розробки інтерактивної гри, що демонструє можливості створення розгалужених сюжетів на основі користувацького вибору.

У ході роботи було розроблено візуальну новелу, яка включає один невеликій розділ, три можливих закінчень та інтерактивний наративний геймплей. Розроблене програмне забезпечення має практичне значення як приклад реалізації наративно-орієнтованої гри з можливістю варіативного розвитку сюжету залежно від рішень користувача.

Було розглянуто теоретичні основи інтерактивних наративів у відеоіграх. Проведено аналіз існуючих концепцій інтерактивного наративу, визначено його ключові елементи та вплив на ігровий досвід. Було досліджено популярні ігрові жанри та механіки, що сприяють створенню наративної глибини. Особлива увага приділена прикладам відомих ігор, таких як The Walking Dead, Life is Strange та Heavy Rain, які демонструють різні підходи до інтерактивного сторітелінгу.

Було розроблено архітектуру гри за допомогою UML-діаграм, що включають основні об'єкти та їх взаємодію. Використано Scriptable Object для організації даних, що забезпечило гнучкість у налаштуванні ігрових компонентів та сценаріїв.

Отже, дослідження інтерактивних наративів у відеоіграх продемонструвало значимість поєднання нелінійного сюжету та емоційного впливу на гравця. Впровадження інноваційних технологій та механік у візуальних новелах дозволяє досягти більшого занурення у гру та створює унікальний досвід для кожного користувача.

Результати дослідження відповідають поставленим завданням і сприяють розв'язанню проблеми створення глибоких та емоційно забарвлених інтерактивних сюжетів. Запропонований підхід до реалізації гри на основі C# у середовищі Unity

є практичним прикладом створення інтерактивних наративів для розгалужених історій.

На основі проведеного дослідження можна зробити висновок, що розробка інтерактивних ігор є актуальним та перспективним напрямом у сучасній індустрії відеоігор. Використання гнучких інструментів, таких як Scriptable Object у Unity, дозволяє значно спростити роботу над наративними проєктами та забезпечує можливість подальшого розвитку.

Таким чином, результати дослідження підтверджують важливість інтерактивного наративу як ключового інструменту для створення ігор, що здатні викликати емоційний відгук у гравців та стимулювати їх до прийняття рішень, які впливають на кінцевий результат історії.

ПЕРЕЛІК ПОСИЛАНЬ

1.Візуальні новели. Вікіпедія. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BD%D0%BE%D0%B2%D0%B5%D0%BB%D0%B0

2.Unity Documentation. Офіційна документація Unity. [Електронний ресурс]. Режим доступу: https://docs.unity.com

3.C# Programming Guide. Microsoft Docs. [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/

4.Дослідження візуальних новел [Електронний ресурс]. Режим доступу: https://www.gamedesignnovels.com

5.Інтерактивні наративи в іграх [Електронний ресурс]. Режим доступу: https://gamepedia.com/narrative-design

6.Гра Steins;Gate – офіційний сайт [Електронний ресурс]. Режим доступу: https://steins-gate.com

7.Галочкін О.В. Основи створення візуальних новел [Електронний ресурс]. Режим доступу: https://gameeducation.ua/articles/visual-novels

8.Danganronpa. Офіційний сайт гри [Електронний ресурс]. Режим доступу: https://www.danganronpa.com

9.Oxenfree – інтерактивні діалоги [Електронний ресурс]. Режим доступу: https://www.nightstudios.com/oxenfree

10.Undertale. Гра, яка змінює вибір гравців [Електронний ресурс]. Режим доступу: https://undertale.com

11.Phoenix Wright: Ace Attorney. Офіційний сайт [Електронний ресурс]. Режим доступу: https://www.capcom.com/ace-attorney

12.Game UI Design Essentials. [Електронний ресурс]. Режим доступу: https://uxdesign.cc/game-ui-essentials

13.Scriptable Objects in Unity. [Електронний ресурс]. Режим доступу: https://unity.com/learn/scriptable-objects

14.Design Patterns for Game Development. [Електронний ресурс]. Режим доступу: https://refactoring.guru/design-patterns/game-development

15.Microsoft Visual Studio – огляд [Електронний ресурс]. Режим доступу: https://visualstudio.microsoft.com

16.Шилдт Г. "C# 9.0 Complete Reference" / Г. Шилдт. 2020 рік. – 1224 с.

17.Фленов М. "Unity Game Development Bible" / М. Фленов. 2022 рік. – 875 с.

18.Основи гейм-дизайну: Як створювати сюжетні ігри [Електронний ресурс]. Режим доступу: https://gamedesign.ua

19.Підручник Unity для новачків [Електронний ресурс]. Режим доступу: https://learn.unity.com

20.AI в інтерактивних іграх [Електронний ресурс]. Режим доступу: https://ai-in-games.com

21.Інтерактивні історії: майбутнє ігор [Електронний ресурс]. Режим доступу: https://interactive-stories.net

ДОДАТОК А – Код програми

Код командера Commander

```
using UnityEngine;
using Game.Data;
using Game.View;
using ViewBG = Game.View.BG;
using CommandBG = Game.Data.BG;
using CommandChoise = Game.Data.Choise;
using ViewChoise = Game.View.ChoiseView;

public class Commander : MonoBehaviour
{
    [Header("SO data")]
    [SerializeField] private ListCommand _listCommands;

    [Header("View")]
    [SerializeField] private ViewBG _commandBG;
    [SerializeField] private Say _commandSay;
    [SerializeField] private ViewChoise _commandChoise;
    [SerializeField] private TitleOverlay _commandSubtitle;
    [SerializeField] private Game.View.Character _commandCharacter;

    private int _indexCommand = 0;
    private int _indexBlockCommand = 0;
    private ScriptableObject _curentCommand;
    private BlockCommand _curentBlock;

    private void Start() => ExecuteCommand();

    public void ExecuteCommand()
    {
        _curentBlock ??= _listCommands.Blocks[_indexBlockCommand];
        _curentCommand = _curentBlock.GetCommand[_indexCommand];

        switch (_curentCommand)
        {
            default:
                Debug.Log($"Unknow command, type of command {_curentCommand.GetType().Name}");
                break;
```

```
        case CommandBG bg:
            _commandBG.End += EndCommand;

            if (bg.Type == BgType.show) _commandBG.Show(bg.Sprite);
            else _commandBG.Hide();

            break;

        case Monolog monolog:
            _commandSay.End += EndCommand;

            _commandSay.Something(monolog);
            break;

        case CommandChoise choise:
            _commandChoise.End += EndCommand;

            _commandChoise.Add(choise.Choises);
            break;

        case Game.Data.Character character:
            _commandCharacter.End += EndCommand;
            if (character.Type == CharacterType.Show)
                _commandCharacter.Show(character.GetSprite);
            break;

        case Subtitles subtitles:
            _commandSubtitle.ShowTitle(subtitles.Text);
            EndCommand(_commandSubtitle);
            break;

        case Game.Data.Stop stop:
            stop.OnText();
            break;
    }
}

private void EndCommand(iCommand command)
{
    command.End -= EndCommand;

    ChoiseView choiseCommand = command as ChoiseView;
```

```
        if (choiseCommand != null)
        {
            _indexCommand = choiseCommand.LastChoiseIndex;
        }

        if (_indexCommand < _curentBlock.GetCommand.Length - 1)
        {
            _indexCommand++;
            ExecuteCommand();
        }
        else if (_indexBlockCommand < _listCommands.Blocks.Length - 1)
        {
            _indexCommand = 0;
            _indexBlockCommand++;
            _curentBlock = null;
            ExecuteCommand();
        }
    }
}
```

Код відображення Write

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using UnityEngine;


namespace Game
{
    public class Write
    {
        public enum Status
        {
            Enabled,
            Diabled
        }

        public event Action<string> SelfWrite;
        public event Action EndWrite;
```

```
        public Status SelfStatus = Status.Diabled;

        private StringBuilder _builder;
        private WaitForSeconds _wait;

        public Write(float time)
        {
            _builder = new StringBuilder();
            _wait = new WaitForSeconds(time);
        }

        public IEnumerator Get(string str)
        {
            SelfStatus = Status.Enabled;
            _builder.Clear();
            int wordIndex = 0;

            while(SelfStatus == Status.Enabled)
            {
                yield return _wait;
                if (wordIndex == str.Length) break;
                _builder.Append(str.Substring(wordIndex, length: 1));
                SelfWrite?.Invoke(_builder.ToString());
                wordIndex++;
            }

            SelfStatus = Status.Diabled;
            EndWrite?.Invoke();
        }
    }
}
```

Код команди BG

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Game.Data
{
    public enum BgType
    {
```

```
        show,
        hide
    }

    [CreateAssetMenu(menuName = "Game/Data/Command/" + nameof(BG))]
    public class BG : ScriptableObject
    {
        [SerializeField] private Sprite _sprite;
        [SerializeField] private BgType _type = BgType.show;

        public Sprite Sprite => _sprite;
        public BgType Type => _type;
    }
}
```

Код команди Character

```
using UnityEngine;

namespace Game.Data
{
    [CreateAssetMenu(menuName = "Game/Data/Command/"+nameof(Character))]
    public class Character : ScriptableObject
    {
        [Header("Storage images")]
        [SerializeField] private StorageImage _storage;
        [Header("Name Character in Storage")]
        [SerializeField] private string _name;
        [Header("Number sprite ")]
        [SerializeField] private int _number;
        [SerializeField] private CharacterType _type;

        public CharacterType Type => _type;

        public Sprite GetSprite => _storage.GetImage($"{_name}_{_number}");
    }

    public enum CharacterType
    {
        Show,
        Hide
    }
```

```
}
```

Код команди Choise

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Game.Data
{
    [CreateAssetMenu(menuName = "Game/Data/Command/" + nameof(Choise))]

    public class Choise : ScriptableObject
    {
        [System.Serializable]

        public class ChoiseElement
        {
            [SerializeField] private string _text;
            [SerializeField] private int _goto;

            public string Text => _text;
            public int Goto => _goto;
        }

        [SerializeField] private ChoiseElement[] _choises;

        public ChoiseElement[] Choises => _choises;
    }

}
```

Код команди Monolog

```
using UnityEngine;

namespace Game.Data
{
    [CreateAssetMenu(menuName = "Game/Data/Command/" + nameof(Monolog))]
    public class Monolog : ScriptableObject
    {
```

```
        [Header(header: "Name character")]
        [SerializeField] private string _name;
        [Header(header: "Character Speech")]
        [SerializeField, TextArea(5,10)] private string _text;

        public string Name => _name;
        public string Text => _text;
    }
}
```

Код команди Stop

```
using UnityEngine;

namespace Game.Data
{
    [CreateAssetMenu(menuName = "Game/Data/Command/" + nameof(Stop))]

    public class Stop : ScriptableObject
    {
        public void OnText() => Debug.Log("Stop command!");
    }
}
```

Код команди Subtitles

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Game.Data
{
    [CreateAssetMenu(menuName = "Game/Data/Command/" + nameof(Subtitles))]
    public class Subtitles : ScriptableObject
    {
        [Header(header: "Subtitles")]
        [SerializeField, TextArea(5, 10)] private string _text;

        public string Text => _text;
    }
}
```

Код команди BlockCommand

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Game.Data
{
   [CreateAssetMenu(menuName ="Game/Data/"+nameof(BlockCommand))]

   public class BlockCommand : ScriptableObject
   {
      [SerializeField] private ScriptableObject[] _command;

      public ScriptableObject[] GetCommand => _command;
   }
}
```

Код команди ListCommand

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Game.Data
{
   [CreateAssetMenu(menuName ="Game/Data/Command/"+nameof(ListCommand))]

   public class ListCommand : ScriptableObject
   {
      [SerializeField] private BlockCommand[] _blocks;

      public BlockCommand[] Blocks => _blocks;
   }
}
```

Код команди StorageImage

```
using UnityEngine;
using System.Linq;
using System;

namespace Game.Data
```

```
{
    [CreateAssetMenu(menuName = "Game/Data/" + nameof(StorageImage))]
    public class StorageImage : ScriptableObject
    {
        [System.Serializable]
        public class Storage
        {
            [Header("Name List image")]
            [SerializeField] private string _name;
            [SerializeField] private Sprite[] _images;

            public string Name => _name;
            public Sprite[] Images => _images;
        }

        [SerializeField] private Storage[] _storages;

        public Storage GetStorage(string name)
        {
            return _storages.FirstOrDefault(s => s.Name == name);
        }

        public Sprite GetImage(string path)
        {
            Sprite[] images = GetStorage(path.Split('_')[0])?.Images;

            if (images == null) return null;

            return images[Convert.ToInt32(path.Split('_')[1])];
        }
    }
}
```

Код відображення BG

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace Game.View
{
    public class BG : MonoBehaviour,iCommand
```

```
    {
        [SerializeField] private Image _self;

        public event Action<iCommand> End;

        private Color[] _colors = new Color[2] { new Color(1, 1, 1, 1), new Color(1, 1, 1, 0) };


        public void Show(Sprite source)
        {
            _self.sprite = source;
            _self.color = _colors[0];
            End?.Invoke(this);
        }

        public void Hide() => _self.color = _colors[1];


    }
}
```

Код відображення Character

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace Game.View
{
    public class Character : MonoBehaviour, iCommand
    {
        [SerializeField] private Image _self;

        public event Action<iCommand> End;

        private Color[] _colors = new Color[2] { new Color(1, 1, 1, 1), new Color(1, 1, 1, 0) };


        public void Show(Sprite characterSprite)
        {
            _self.sprite = characterSprite;
            _self.color = _colors[0];
```

```
            End?.Invoke(this);
        }

        public void Hide()
        {
            _self.color = _colors[1];
            _self.sprite = null;
            End?.Invoke(this);
        }
    }

}
```

Код відображення ChoiseButton

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Game.Data;

namespace Game.View
{
    public class ChoiseButton : MonoBehaviour
    {
        [SerializeField] private TextMeshProUGUI _text;
        [SerializeField] private Button _self;

        public void Show(Data.Choise.ChoiseElement choiseElement, ChoiseView choise)
        {
            _text.SetText(choiseElement.Text);
            _self.onClick.AddListener(() => choise.Onclick(choiseElement.Goto));
        }

        public void Hide() => Destroy(gameObject);
    }
}
```

Код відображення ChoiseView

```
using System.Collections.Generic;
using UnityEngine;
using Game.Data;
using System;
```

```
namespace Game.View
{
    public class ChoiseView : MonoBehaviour, iCommand
    {
        [SerializeField] private Canvas _self;
        [SerializeField] private Transform _parrent;
        [SerializeField] private ChoiseButton _prefabs;

        public event Action<iCommand> End;
        public int LastChoiseIndex = -1;
        private List<ChoiseButton> _choiseButtons = new List<ChoiseButton>();

        public void Show() => _self.enabled = true;

        public void Add(Choise.ChoiseElement[] choiseElement)
        {
            _choiseButtons ??= new List<ChoiseButton>(choiseElement.Length);

            for (int i = 0; i < choiseElement.Length; i++)
            {
                ChoiseButton tmp = Instantiate(_prefabs, _parrent);
                tmp.Show(choiseElement[i], this);
                _choiseButtons.Add(tmp);
            }
            Show();
        }

        public void Onclick(int commandIndex)
        {
            LastChoiseIndex = commandIndex - 1;
            End?.Invoke(this);
            Hide();
        }

        public void Hide()
        {
            for (int i = _choiseButtons.Count - 1; i >= 0; i--)
            {
                _choiseButtons[i].Hide();
            }
            _choiseButtons.Clear();
            _self.enabled = false;
```

```
        }
    }
}
```

Код відображення Say

```
using UnityEngine;
using Game.Data;
using System;
using TMPro;

namespace Game.View
{
    public class Say : MonoBehaviour, iCommand
    {
        [SerializeField] private Commander _commander;
        [SerializeField] private TextMeshProUGUI _name;
        [SerializeField] private TextMeshProUGUI _text;
        [SerializeField] private ChoiseView _choise;

        [Header("Time write")]
        [SerializeField] private float _time;

        public event Action<iCommand> End;

        private Write _write;
        private Coroutine _coroutineWrite;


        private void Awake() => _write = new Write(_time);

        private void OnEnable()
        {
            _write.SelfWrite += Write;
        }

        private void OnDisable()
        {
            _write.SelfWrite -= Write;
        }

        private void Write(string str) => _text.SetText(str);
```

```
        public void Next() => Something(null);

        public void Something(Monolog monologue)
        {
            if (_write.SelfStatus == Game.Write.Status.Enabled)
            {
                StopCoroutine(_coroutineWrite);
                _write.SelfStatus = Game.Write.Status.Diabled;
            }

            if (monologue == null)
            {
                End?.Invoke(this);
                return;
            }

            if (_write.SelfStatus == Game.Write.Status.Diabled) _coroutineWrite = StartCoroutine(_write.Get(monologue.Text));

                _name.SetText(monologue.Name);

        }
    }
}
```

Код відображення TitleOverlay

```
using System;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

namespace Game.View
{
    public class TitleOverlay : MonoBehaviour, iCommand
    {
        [Header("Canvas Settings")]
        [SerializeField] private Canvas _hiddenCanvas;
```

```
[SerializeField] private GameObject _background;
[SerializeField] private TextMeshProUGUI _titleText;
[SerializeField] private Button _exitButton;

public event Action<iCommand> End;

private void Awake()
{
    if (_hiddenCanvas != null)
    {
        _hiddenCanvas.enabled = false;
    }

    if (_exitButton != null)
    {
        _exitButton.gameObject.SetActive(false);
        _exitButton.onClick.AddListener(ExitToMainMenu);
    }
}

public void ShowTitle(string title)
{
    if (_background != null)
    {
        _background.SetActive(true);
    }

    if (_hiddenCanvas != null)
    {
        _hiddenCanvas.enabled = true;
    }

    if (_exitButton != null)
    {
        _exitButton.gameObject.SetActive(true);
    }

    if (_titleText != null)
    {
        _titleText.text = title;
    }
```

```
            End?.Invoke(this);
        }

        public void HideTitle()
        {
            if (_background != null)
            {
                _background.SetActive(false);
            }

            if (_hiddenCanvas != null)
            {
                _hiddenCanvas.enabled = false;
            }

            if (_exitButton != null)
            {
                _exitButton.gameObject.SetActive(false);
            }

            if (_titleText != null)
            {
                _titleText.text = string.Empty;
            }
        }

        private void ExitToMainMenu()
        {
            SceneManager.LoadScene("MainMenu");
        }
    }
}
```

ДОДАТОК Б – Демонстраційні матеріали

1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

ДИПЛОМНА РОБОТА
на ступінь вищої освіти магістр
із спеціальності 122 Комп'ютерні науки

Інтерактивна гра на основі C#

Виконав: студент 6 курсу, групи КНДМ-61
Сенозацький Герман Олегович
Керівник: к. т. н., доцент СЄРИХ С.О

Київ - 2024

2

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИПЛОМНОЇ РОБОТИ

Тема	Інтерактивна гра на основі C#
Мета дослідження	створити інтерактивну гру, яка забезпечує емоційне занурення гравця та дозволяє динамічно адаптувати сюжет залежно від його виборів.
Наукове завдання	розробка інтерактивної гри у жанрі візуальної новели із використанням мови програмування C# та рушія Unity.
Об'єкт дослідження	процес створення інтерактивних ігор із нелінійним наративом.
Предмет дослідження	технології розробки інтерактивних ігор у жанрі візуальних новел.

Постановка задачі

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Розробити архітектуру гри, що включає інструменти для зручного створення діалогів, монологів, задніх фонів та меню вибору гравця.
- Реалізувати систему інтерактивних діалогів, яка дозволяє гравцеві впливати на розвиток сюжету шляхом прийняття рішень.
- Інтегрувати засоби Unity для забезпечення зручного управління сценарними елементами, такими як зміна фону, тексту та вибіркових варіантів відповідей.
- Забезпечити гнучкість у зміні та масштабуванні контенту гри, використовуючи об'єктно-орієнтовані підходи до програмування.

Дослідження і аналіз об'єкта програмування

Жанр візуальних новел виник у 1980-х роках, але справжнього визнання набув у 1990-х в Японії завдяки проектам, які поєднували у собі текстові історії, ілюстрації та інтерактивність. Однією з перших відомих візуальних новел була Kanon (1999), яка задала нові стандарти для цього жанру, пропонуючи гравцям захопливий сюжет, багатовимірних персонажів і можливість впливати на розвиток історії. Популярність цього жанру зросла завдяки його здатності занурювати гравців у інтерактивний досвід, що поєднує літературний стиль із графічним дизайном і музикою.

Рисунок 1 - Ігровий процес гри Detroit: Become Human

Інтерактивні наративи у сучасних відеоіграх набувають все більшої популярності, адже дозволяють створювати унікальні ігрові сценарії, які залежать від вибору гравця. Це особливо актуально для жанру візуальних новел, де сюжет і механіка взаємодії повністю фокусуються на історії, емоційному впливі та особистій участі гравця. Візуальні новели є жанром відеоігор, який поєднує в собі текстовий сценарій, статичні або анімовані ілюстрації та інтерактивні елементи.

Опис мови програмування

Для реалізації поставленої задачі було використано середовище розробки Unity, Microsoft Visual Studio та мову програмування C#.

Unity – це платформа для створення 2D та 3D ігор, яка використовує C# для написання сценаріїв.

Microsoft Visual Studio – інтегроване середовище розробки, яке забезпечує підтримку Unity та мови C# для розробки ігор.

Мова програмування C# – об'єктно-орієнтована мова, яка використовується для написання сценаріїв у Unity.

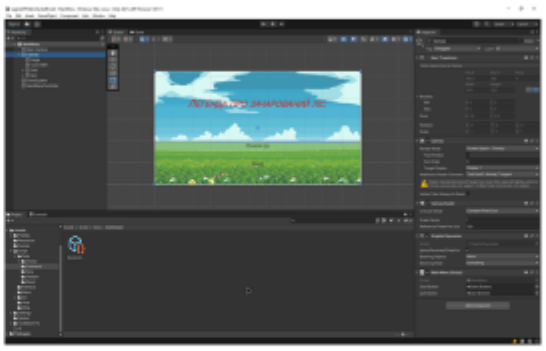

Рисунок 2 - Середовище розробки Unity

Життєвий цикл

Рисунок 3 – Інкрементальна модель

Діаграми прецедентів

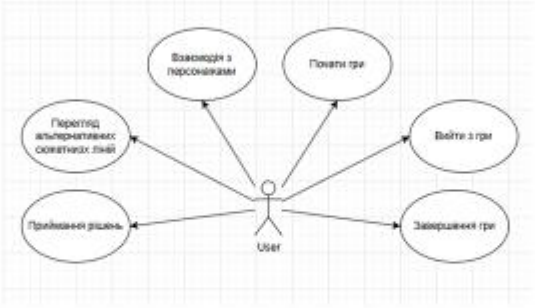

Рисунок 4 – Діаграма прецедентів для користувача

Діаграма класів

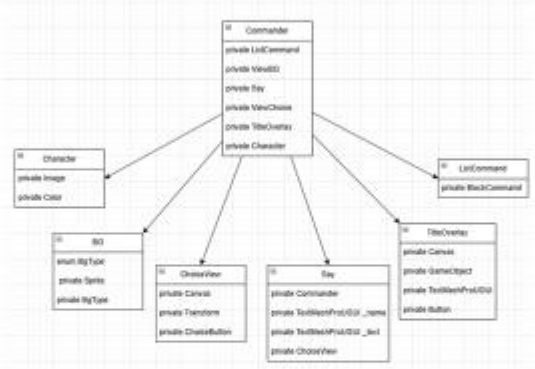

Рисунок 5 – Діаграма класів проекту

Алгоритм роботи системи

Після запуску гри користувач потрапляє в головне меню, де може обрати один із доступних варіантів: розпочати гру або завершити програму. При виборі пункту «Розпочати гру» гравець переноситься до першої сцени, де починається сюжет із пробудження героя в лісі.

У процесі гри:

- Гравець взаємодіє з персонажами через систему діалогів, обираючи відповіді або дії.
- Кожен вибір гравця впливає на подальший розвиток сюжету, активуючи відповідні сценарії та сцени.
- Гра завершується однією з трьох кінцівок залежно від дій гравця.

Рисунок 6 – Алгоритм роботи програми

Інтерфейс гри

Рисунок 7 – Головне меню гри

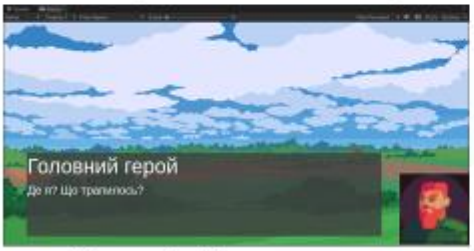

Рисунок 8 – Вступна сцена гри

Рисунок 9 – Взаємодія з діалогами

Рисунок 10 – Вибір відповіді у діалозі

ВИСНОВКИ

У ході роботи було розроблено візуальну новелу, яка включає один невеликій розділ, три можливих закінчень та інтерактивний наративний геймплей. Розроблене програмне забезпечення має практичне значення як приклад реалізації наративно-орієнтованої гри з можливістю варіативного розвитку сюжету залежно від рішень користувача.

Було розглянуто теоретичні основи інтерактивних наративів у відеоіграх. Проведено аналіз існуючих концепцій інтерактивного наративу, визначено його ключові елементи та вплив на ігровий досвід.

Було розроблено архітектуру гри за допомогою UML-діаграм, що включають основні об'єкти та їх взаємодію. Використано Scriptable Object для організації даних, що забезпечило гнучкість у налаштуванні ігрових компонентів та сценаріїв.

Таким чином, результати дослідження підтверджують важливість інтерактивного наративу як ключового інструменту для створення ігор, що здатні викликати емоційний відгук у гравців та стимулювати їх до прийняття рішень, які впливають на кінцевий результат історії.