Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтерактивна новела на основі Unity з механізмом прийняття рішень

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Олексій ШАФАРУК
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Владислав ХОЛЄВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Шафаруку Олексію Михайловичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інтерактивна новела на основі Unity з механізмом прийняття рішень

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1) затверджена тема роботи;
2) технології розробки проєкту;
3) документації до обраних технологій;
4) ресурси для наповнення ігрового застосунку.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області;
2) аналіз існуючих аналогів ігрових застосунків;
3) аналіз та вибір технологій розробки;
4) реалізація програмної частини;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	27.05.25-30.05.25	
2	Аналіз існуючих програмних рішень	01.06.25-02.06.25	
3	Аналіз та вибір технологій розробки	03.06.25-05.06.25	
4	Розробка графічного інтерфейсу та програмної	06.06.25-10.06.25	
	частини застосунку		
5	Тестування застосунку та усунення багів	11.06.25-12.06.25	
6	Оформлення кваліфікаційної роботи	13.06.25-14.06.25	
7	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Владислав ХОЛЄВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 59 с., 19 рис., 1 дод., 12 джерел.

ВІДЕОГРА, ІГРОВИЙ ПРОЦЕС, СЮЖЕТ, СЮЖЕТНА ЛІНІЯ, КЛАС, ІНТЕРФЕЙС, СЦЕНА, КОРИСТУВАЧ, ПЕРСОНАЖ, NPC, КОНТРОЛЕР, КАРТА, МЕТОД, ФУНКЦІЯ, ЛОКАЦІЯ.

Метою кваліфікаційної роботи є розробка інтерактивної відеогри новели на основні двигуна Unity з механізмом прийняття рішень.

У ході виконання кваліфікаційної роботи було використано необхідні ресурси для створення програмного застосунку. Використовувалися такі технології, як: ігровий двигун Unity, середовище розробки Visual Studio 2022, мова програмування C# та JSON для налаштування файлової системи збереження та завантаження даних.

Перед основним процесом створення застосунку – ознайомлено з обраним напрямком розробки, а також з наявними на ігровому ринку аналогами.

Результатом розробки проєкту є повноцінна відеогра, в якій користувач може провести свій час.

ABSTRACT

Bachelor’s thesis: 59 pages, 19 figures, 14 appendices, 12 sources.

VIDEO GAME, GAME PROCESS, PLOT, STORYLINE, CLASS, INTERFACE, SCENE, USER, CHARACTER, NPC, CONTROLLER, MAP, METHOD, FUNCTION, LOCATION.

The major goal of this thesis is to develop an interactive video game novella based on the Unity engine with a decision-making mechanism.

During the qualification work, the necessary resources were used to create a software application. The following technologies were used: the Unity game engine, the Visual Studio 2022 development environment, the C# programming language and JSON for configuring the data saving and loading system.

Before the main process of creating the application, the selected development direction was familiarized with, as well as with analogues available on the gaming market.

The results of the project development are a full-fledged video game in which the user can spend their time.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ООП – об’єктно-орієнтоване програмування

ПК – персональний комп’ютер

2D – двовимірна графіка (англ., 2-Dimensional)

C# – об’єктно-орієнтована мова програмування

IDE – інтегроване середовище розробки (англ., Integrated Development Environment)

JSON – текстовий формат обміну даними (англ., JavaScript Object Notation)

NPC – некерований гравцем персонаж (англ., Non-Playable Character)

QTE – короткочасна подія (англ., Quick Time Events)

ВСТУП

В наш час стрімко розвивається сфера інформаційних технологій. Напрямок цієї сфери, а саме розробка ігрових застосунків, значно зацікавлює багатьох фахівців з галузі інформаційних технологій. Багато спеціалістів витрачають свої сили на розробку відеогри, яка у майбутньому можливо сподобається користувачам.

На ринку ігрових застосунків налічується величезна кількість різноманітних проєктів. Люди, які цікавляться сферою відеоігор, з дуже великим шансом знайдуть для себе підходящій варіант. Основними критеріями під час вибору відеогри, в якій користувач хоче провести час, є її жанр, платформа, ігровий процес, стилістика, системні вимоги, ціна та відгуки.

Для когось відеоігри це не просто відпочинок та розвага, а й їх робота. Є ігрові дисципліни, в яких гравці змагаються між собою поодинці або у команді. Часто в таких дисциплінах проводяться турніри та вони транслюються на велику кількість глядачів. Також є люди, які пов’язують своє життя з діяльністю стрімера. Контентом для трансляцій слугує ігрове наповнення.

Інтерактивні сюжетно орієнтовані відеоігри користуються популярністю серед гравців. Такі відеоігри дозволяють не тільки зануритися в сюжет, але й самому керувати історією. Всі ці фактори свідчать про те, що напрямок розробки відеоігор швидко розвивається та набирає все більшої популярності в сучасному світі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Інтерактивною новелою називають різновид художнього твору, в якому сюжет не є лінійним та фіксованим, він залежить від вибору людини, яка ознайомлюється з історією.

Відеогра в кваліфікаційній роботі реалізується саме у вигляді інтерактивної новели. Механізм прийняття рішень у відеогрі дозволить називати її інтерактивною, бо обрані гравцем рішення впливають на подальші зміни в сюжетній лінії. Цей жанр обрано для того, щоб гравець приймав активну участь у сюжетному розвитку, бо людині більше подобається, коли вона сама приймає участь в розвитку подій.

В якості розробника застосунок створюється з метою реалізації внутрішніх механізмів та функціональності, які могли б збільшити кваліфікацію в цьому напрямку розробки.

Сюжет гри подається гравцю через діалоги та текстове наповнення. Так, як гра передбачає втручання гравця в сюжету, то звичайно він має розгалуження, які відповідають різним ігровим ситуаціям [1].

Основний елемент відеогри – це механізм прийняття рішень, саме цей механізм дозволяє гравцю впливати на сюжет, як він бажає. У відеогрі присутні звичайні вибори, які впливають на сюжетну лінію, та вибори з додатковими механізмами. Під додатковими механіками маються на увазі вибори залежні від: атрибутів головного героя, стану взаємовідносин з NPC, шансів на успіх [2].

Завершенням відеогри користувач може вважати момент, коли він пройшов всі локації ігрового світу.

Основні елементи відеогри: карта ігрового світу, сюжетне дерево, головний герой, графічний інтерфейс, механізм прийняття рішень. Ігровий процес складається з того, що гравець запускає гру на своєму ПК обирає почати нову гру або завантажити збереження. Далі гравець обирає локацію

на карті ігрового світу та знайомиться з сюжетною лінією та приймає рішення, які йому подобаються. Локації йдуть у послідовному порядку, тобто щоб перейти до другої локації потрібно завершити першу і так далі. Таким чином гравець рухається по сюжетній лінії та локаціям до завершення гри. По завершенню гри користувач отримає одну з наявних кінцівок.

Цільовою аудиторію розроблювальної відеогри можна вважати людей віком від 16 до 30 років будь-якої статі, які цікавляться ігровою індустрією. Гравці можуть зацікавитися відеогрою з метою гарного проведення вільного часу або ознайомлення з ігровим сюжетом, або відволіктися від ігор, які потребують більш активних роздумів та швидкої реакції. Але грою також можуть зацікавитися й інші категорії людей. Все буде залежати від їх уподобань та темпераменту.

Перед розробкою ігрового застосунку необхідно було проаналізувати існуючі аналоги. Це важливо зробити, щоб підкреслити позитивні та негативні сторони існуючих проєктів та використовувати отриманні знання при реалізації власного.

Надалі розглянемо такі існуючі аналоги відеоігор:

- Choice of Life;
- Reigns;
- Detroit: Become Human;
- Until Dawn.

Найбільш популярні відеоігри цього списку – Detroit: Become Human та Until Dawn. Для останньої було випущено модернізовану версію. Відеоігри Choice of Life та Reigns набули меншої популярності, але вони більш схожі на розроблювальну відеогру, тому їх слід розібрати більш конкретно.

Загалом успішні відеоігри розробляються не однією людиною, а великою компанією, в якій кожен співробітник бере на себе якусь частину роботи. Тому у випадку коли відеогра розроблюється однією людиною, слід проаналізувати та виділити основні моменти, які потрібно реалізувати та вирішувати основні проблеми з якими зіткнемося в процесі розробки [3].

1.1 Відеогра Choice of Life

В цьому розділі розглянемо найбільш схожу по наповненню та оформленню відеогру Choice of Life. ЇЇ випущено в 2020 році компанією Blazing Planet Studio. На рисунку 1.1 бачимо лаконічне оформлення, текстову вставку з сюжетним наповненням та два варіанти вибору подій у формі карточок.

Рисунок 1.1 – Кадр з відеогри Choice of Life

З переваг відеогри можна виділити більш простий процес розробки через те, що вона має 2D формат та спрощений графічний інтерфейс. Тому подібний застосунок можна було б реалізувати використовуючи порівняно невеликі ресурси та робочу силу. В більшості випадків гравець вибирає з двох варіантів іноді з трьох. Це свідчить про не сильно розгалужену сюжетну лінію та більш чітку логіку вибору, що також спрощую реалізацію.

Недоліками гри є обмежена глибина сюжету через реалізовану в більшості випадків модель з двома виборами. Також через спрощений графічний інтерфейс гравець відчуває менше занурення в ігровий процес.

Додатково можна виділити те, що не всім користувачам сподобається просто обирати варіант в картковому вигляді, бо деякі бажають порухатися за персонажа в ігровому середовищі або пограти в щось більш динамічне.

Також у відеогрі реалізовано систему серць, тобто в процесі гравець може зробити вибори, які призведуть до смерті ігрового персонажу. Це означає, що деякі вибори не ведуть до якихось глобальних змін в сюжетній лінії, а лише впливають на те, чи потрібно буде гравцю спочатку проходити ігрову ситуацію.

Відеогра розповідає вигадану історію персонажа, який народився в часи середньовіччя звичайним селянином. В процесі гравець визначає виборами напрям в якому буде рухатися головний герой та ким він стане. Наприклад головний персонаж може стати лицарем, злодієм або королем.

З цієї відеогри можна запозичити простоту інтерфейсу, оформлення механіки прийняття рішень та можливо щось подібне до системи сердець [4].

1.2 Відеогра Reigns

Reigns – симулятор управління королівством. Основний геймплей відеогри складається з балансування чотирьох основних параметрів королівства, а саме військом, грошима, народом та церквою. На рисунку 1.2 зображено кадр з відеогри, на якому бачимо сюжетний текст, можливість вибору дії, параметри королівства та скільки років гравець займає королівську посаду.

Гравцю надходять події, які трапляються в королівстві та які потрібно вирішити так, щоб тримати баланс параметрів королівства. Нехтування одним з параметрів призводить до завершення гри однією з кінцівок в яких головний персонаж вмирає.

В цілому мета гри – протриматися на королівській посаді, як можна більше ігрових років та розвинути своє королівство. Відеогра відрізняється своїм багаторазовим проходженням та генерацією випадкових подій. Гра

завжди починається приблизно однаково з етапу становлення на посаду короля, але процес правління кожний раз трохи різний. Тобто кожна ігрова сесія чимось відрізняється від попередніх, бо ігрові події надходять в різному порядку, але самі події не змінюються.

Також гра має багато специфічних виборів, які призводять до несподіваних прихованих результатів. Ці результати часто можуть показатися гравцю не дуже логічними, що для деяких може слугувати негативною якістю.

Ця відеогра має найбільш лінійний сюжет в порівнянні з іншими розглянутими. У випадковий час ігрового процесу відбувається деяка подія, яка запускає ланцюг виборів з однієї сфери. Наприклад до короля може прийти знахар із зіллям та пропонувати свої послуги. Це буде тригером для цілого ланцюга подій пов’язаних з цим ігровим персонажем. Але подія зі знахарем може не початися через випадкове генерування, тому ланцюг подій в такому випадку не почнеться. Отже, у відеогрі більш менш лінійний сюжет побудований на ланцюгах подій з різними ігровими персонажами.

Рисунок 1.2 – Інтерфейс та прийняття рішення у відеогрі Reigns

1.3 Відеогра Detroit: Become Human

Detroit: Become Human – кінематографічна інтерактивна гра випущена в 2020 році компанією Quantic Dream. Гравець керує трьома головними персонажами для кожного з яких приймає рішення, що впливають на подальші події, які відбудуться з персонажами. В цій грі є результати виборів, які будуть впливати, як в конкретному епізоді так і більш глобальні, які впливають в подальших епізодах. На рисунку 1.3 проілюстровано сцену одного з виборів у першому епізоді.

Відеогра розповідає історію про життя створених людьми андроїдів, тобто штучного інтелекту, якому придали людського вигляду та впровадили в своє повсякденне життя. Гравець грає за трьох андроїдів, які поступово становляться девіантами. Кожен з них починає виділятися від інших та набуває самосвідомості. Людям це звичайно не подобається і вони борються з такою поведінкою, тому це призводить до конфліктів.

Головні персонажі андроїди: Конор, Маркус, Кера.

Конор – андроїд поліцейський, який був наданий для розслідування справ, пов’язаних із девіантними андроїдами. Також додатковою його ціллю було збирання інформації, яка допоможе зрозуміти, чому андроїди стають девіантами. Цей персонаж більше за інших боровся з признаками своєї девіантності та намагався уникнути цього, але близько до кінця історії гравцю пропонують вибір, який може залишити Конора справним андроїдом або зробити з нього девіанта.

Маркус – андроїд призваний допомагати відомому художнику, бо останній був хворий та не міг ходити самостійно. Але в один момент життя цього андроїду повністю перевертається, коли в їх дім завітає син художника і залежно від вибору користувача художник або син помирає. В результаті чого звинувачують саме андроїда та викидають його на звалище. Маркус виживає та протягом гри готує план повстання штучного інтелекту проти людей.

Кера – андроїд домогосподарка, яка живе разом із сім’єю в якій відсутня мати маленької дівчини, а батько веде поганий спосіб життя та іноді навіть дуже лякає свою дитину. По сюжету гри в один з вечорів, коли батько знову сильно розлютився на дівчинку, то Кера приймає рішення втікати разом із нею. У підсумку сюжет для цього персонажу складається з того, що Кера з дівчиною шукають місце, де б могли жити спокійно та ні за що не переживаючи.

Сюжетні лінії трьох головних героїв стикаються в деяких моментах та вибір за одного персонажу може впливати на іншого.

З переваг гри можна виділити високоякісну графіку, анімації та голосове акторське наповнення. Це дозволяє створити ефекти реалістичності та занурення гравця. В деяких моментах присутні глибокі сюжетні вибори, які можуть зачепити емоції гравця. На відміну від розглянутих попередньо відеоігор, в цій більша частина сцен з виборами не обмежується двома варіантами, а складається з трьох або чотирьох. Також гра має систему з декількома головними персонажами, які можуть впливати один на одного, що також більше зацікавлює [5].

У відеогрі також присутня цікава діаграма епізодів. На діаграмі користувач може продивитися сценарій пройденого епізоду та які варіанти розвитку подій він обрав і до чого вони призвели. Також можна побачити закриті місця, в які гравець не попав та відсоток людей, які обирали різні варіанти подій.

Відеогра має високоякісну графіку, тому не всі ПК зможуть подужати таку напругу. Також в деяких випадках користувачі зіткалися з моментами, коли застосунок припиняв роботи, через критичні помилки або нестабільну поведінку гри. Як вже було наведено, ігрові персонажі взаємодіють один з одним, але не на такому рівні, як хотілося б. Більшість взаємодій це поверхневі сюжетній епізоди, які не дуже на щось впливають. Також у грі є багато виборів, які є фіктивними або мають мінімальний вплив на сюжетну лінію.

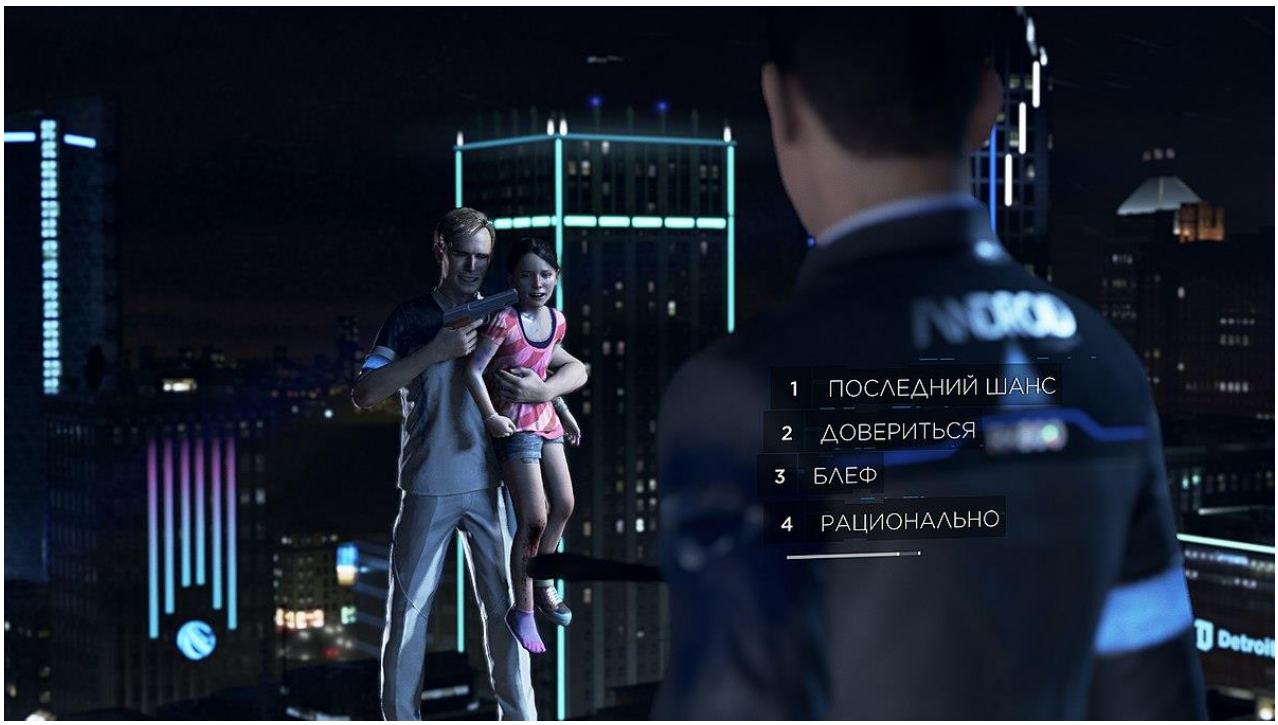

Рисунок 1.3 – Один з виборів в першій місії/епізоді відеогри

1.4 Відеогра Until Dawn

Until Dawn – відеогра у жанрі інтерактивного жаху випущена в 2015 році компанією Supermassive Games. А в 2024 році було видано модифіковану версію. Гра розповідає історію про вісім друзів, які повернулися до будинку одного з них, щоб влаштувати для себе невелике свято. Будинок розташований у горах та вони планують провести там усі вихідні. Але в процесі гри кожен з персонажів може зіткнутися з жахливими подіями, які можуть призвести до смерті.

Гравець керує за кожного з восьми головних персонажів. Кожен з них може загинути або вижити в залежності від виборів. На відміну від Detroit: Become Human в цій грі персонажі взаємодіють між собою майже постійно, та вибір за одного з персонажів може призвести до смерті іншого. Для загиблого персонажа в такому випадку закінчується сюжетна лінія. Ціль гри пережити вихідні за всіх персонажів, але можна й досягати інших кінцівок, яких в грі багато через, те що персонажі помирають та для них обривається сюжетна лінія. На рисунку 1.4 зображено персонажів за яких керує гравець.

Рисунок 1.4 – Персонажі Until Dawn

З переваг Until Dawn виділимо тісну взаємодію ігрових персонажів. Вибори за одного з персонажів можуть призвести до критичних змін в сюжетній лінії для іншого або навіть до її обриву, у випадку смерті персонажу. Гра в більшості надає три варіанти вибору в подіях, що трохи розширює варіативність. Випущена модифікована версія має ще більш якісну графіку. Жанр жаху посилює емоції та залучення гравця. В грі присутній малий відсоток фіктивних виборів та багато виборів можуть вплинути на фінальний результат.

З недоліків виділимо не зручне повторне проходження. Наприклад в порівнянні з тою ж Detroit: Become Human, в якій можна повторно пройти кожен епізод окремо, в Until Dawn так зробити не вийде. Тому користувачу потрібно буде витратити більше часу, якщо він захоче отримати іншу кінцівку. На початку гри темп розвитку сюжету йде трохи повільно, що також може не всім сподобатися.

Важливим елементом Until Dawn є QTE, який використовується дуже часто в процесі гри. Ця механіка не всім подобається, але для жанру жаху без цього елементу точно не обійтись, бо він не дає гравцю перепочити. Це також допомагає досягти мету жанру, а саме налякати гравця.

Detroit: Become Human та Until Dawn також мають специфічний недолік. Відеоігри виконано з високоякісною графікою, анімаціями, продуманим сюжетом. Для створення такого продукту потрібно багато коштів, ресурсів, людей. Не кожна компанія може собі це дозволити.

1.5 Постановка задачі

Метою кваліфікаційної роботи є створення інтерактивної відеогри новели на основі двигуна Unity з механізмом прийняття рішень для одного користувача.

На основі розглянутих існуючих аналогів виділено їх сильні та слабкі сторони та запозичено деякі елементи. Обрано спрощений графічний інтерфейс 2D формату, через те що розробка проводиться однією людиною і займе менше часу та ресурсів. Кількість виборів в різних ситуаціях варіюється від двох до чотирьох, що зменшує варіативність в тих місцях сюжету, де це не потрібно, щоб створювати менше фіктивних виборів та збільшує варіативність в разі вибору між чотирма варіантами. Ігрове наповнення буде складатися з просування гравця по локаціям, в яких він буде приймати рішення. Застосунок буде включати в себе карту ігрового світу та чотири локації на ній. На кожній з ігрових сцен користувач буде приймати свої рішення, що будуть призводити до зміни сюжетної лінії.

Для досягнення поставленої мети потрібно буде виконати наступні завдання:

- реалізація потрібної та коректної функціональності;
- створення інтуїтивно привабливого та зрозумілого інтерфейсу;
- налаштування файлової системи для зберігання та завантаження даних;
- втілення сюжетної лінії;
- реалізація механізму збереження ігрового прогресу;
- тестування всіх аспектів створеного застосунку та усунення багів.

2 ВИКОРИСТОВУВАНІ ТЕХНОЛОГІЇ

В цьому розділі розглянемо технології, які знадобилися в процесі розробки проєкту. В якості ігрового двигуна обрано платформу Unity. Обрано мову програмування – C#, IDE – Visual Studio 2022. Для реалізації зберігання та завантаження інформації використовується технологія JSON. В якості архітектурного шаблону, для організації та структуризації коду використовується MVC шаблон. Створення ігрової карти світу виконувалось в онлайн застосунку Inkarnate. Також використовувалися технології штучного інтелекту для генерування зображень, які використовуються у відеогрі. Здебільшого використовувався штучний інтелект Leonardo AI.

2.1 Ігровий двигун Unity

Unity – потужна платформа для розробки комп'ютерних ігор, розроблена компанією Unity Technologies (рисунок 2.1). Для розробки проєкту використовується версія 2022.3.22f1. Unity надає розробнику широкий набір інструментів для створення ігрових графічних застосунків [6].

Рисунок 2.1 – Логотип ігрового двигуна Unity

Основні переваги Unity:

- підтримка різних платформ;
- інтерфейс графічного редактору зрозумілий та приємний на вигляд;
- безкоштовне використання;
- підтримка 2D та 3D форматів;

- повноцінна інтеграція з Visual Studio та мовою програмування C#;
- велика спільнота, яка надає багато потрібної інформації розробникам;
- Asset Store надає різноманітні рішення, програмні модулі та інше [7].

2.2 Середовище розробки Visual Studio 2022

Visual Studio 2022 – інтегроване середовище розробки, розроблене компанією Microsoft (рисунок 2.2). Використовується для створення різноманітних програмних застосунків. Можливість розробки консольних застосунків та застосунків з графічним інтерфейсом. Підтримує багату кількість різних мов програмування включаючи найбільш відомі. Також середовище підтримує мову програмування C#, яка використовується при розробці ігрового застосунку. Середовище дозволяє розробнику писати, редагувати, відлагоджувати програмний код [8].

У рамках проєкту Visual Studio використовувалась в цілях написання файлів скриптів та моделей, які потім прив’язуються до Unity об’єктів.

В якості основних переваг середовища розробки виділимо наступні:

- повна інтеграція з платформою Unity;
- механіки для зручного налагодження;
- інтелектуальні підказки під час написання коду;
- зручний та зрозумілий інтерфейс;
- підтримка великої кількості бібліотек, розширень та іншого.

Рисунок 2.2 – Логотип середовища розробки Visual Studio

2.3 Мова програмування C#

Для написання коду ігрового застосунку обрано мову програмування C#, розроблена компанією Microsoft. За допомогою цієї мови програмування реалізуються різноманітні веб-застосунки, мобільні застосунки, відеоігри та інше.

Основні переваги мови програмування C#:

- інтуїтивно зрозумілий синтаксис;
- підтримка принципів ООП;
- сумісність з технологіями Visual Studio та Unity;
- вбудований збирач сміття;
- активна спільнота;
- підтримка великої кількості бібліотек та фреймворків.

Мову програмування C# обрано через те, що вона є основною мовою для розробки на платформі Unity. Весь код реалізовано виключно за допомогою мови цієї мови. Загалом зв’язка трьох технологій Unity, Visual Studio та C#, є сучасним та гарним рішенням, яке постійно використовується при розробці різних проєктів. З саме цими технологіями набуто більшість знань впродовж навчального процесу, що також дає неабиякі переваги під час роботи над реалізацією проєкту [9].

2.4 Технологія зберігання даних JSON

JSON – один з найпопулярніших форматів збереження та представлення даних. Використовується для зберігання структурованих даних та їх передачі. Структура JSON-файлів зручна та зрозуміла, людина може з легкістю знайти потрібну їй інформацію (рисунок 2.3). Технологія надає можливість збереження складних класів, списків, структур. JSON гнучкий в разі зміни структури даних, які потрібно зберігати. Сумісний з Unity та C# за допомогою використання необхідних бібліотек.

JSON використовується в проєкті для зберігання ігрової інформації. Інформація, що зберігається в проєкті:

- характеристики ігрового персонажу;
- стан гри;
- інформація про NPC;
- сюжет ігрових локацій;
- файли збережень ігрового прогресу.

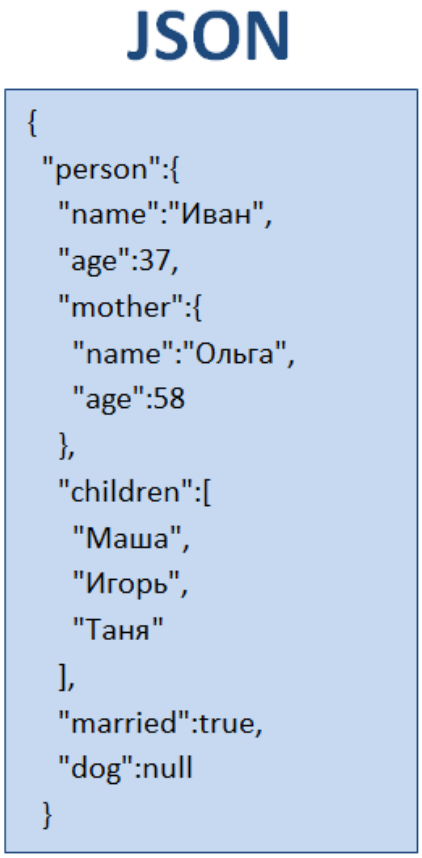

Рисунок 2.3 – JSON формат представлення даних

2.5 MVC шаблон програмування

В якості шаблону програмування обрано MVC. Він розділяє програму на 3 пов’язані частини: модель, контролер та представлення. Модель зберігає в собі дані. Контролер відповідає за логіку взаємодії користувача і системи. Представлення відповідає за відображення зовнішнього вигляду застосунку.

На рисунку 2.4 можна переглянути схематичне представлення шаблону у випадку розроблюваного проєкту. Ключова ідея використання цього шаблону є те, що розділивши програму на три частини вона стає більш гнучкою. ЇЇ буде легше модифікувати та конфігурувати, а також програма стане більш структурованою та зрозумілою на вигляд.

У випадку нашого ігрового застосунку роль представлення відіграє графічне наповнення в ігровому двигуні Unity. До моделі відносяться класи, які містять в собі різні дані, наприклад класи героя, локації, NPC, стану гри, ігрові вибори, сцени та інше. В якості контролера слугують класи, які виконують логіку гри під час взаємодії гравця з графічним інтерфейсом [10].

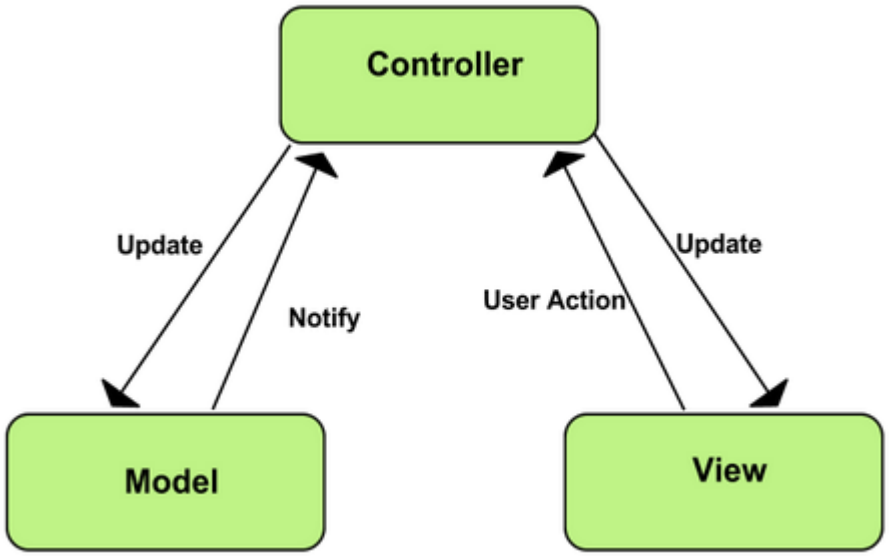

Рисунок 2.4 – MVC шаблон

2.6 Онлайн платформа створення карт Inkarnate

Ігровий застосунок передбачає використання ігрової карти світу. Для створення карти обрано онлайн інструмент Inkarnate (рисунок 2.5). Здебільшого Inkarnate обрано через його безкоштовність на відміну від багатьох інших інструментів. Також на цій платформі зрозумілий та не перевантажений графічний інтерфейс. Присутня велика бібліотека з

елементами, які можна використовувати про створені карти. Із створенням карти зіткнулися вперше і навичок в цій сфері було небагато, але в Інтернеті було багато різних відео, інструкцій та керівництв про цей застосунок. На відміну від інших інструментів для Inkarnate було знайдено набагато більше інформації, що також зіграло велику роль під час вибору технології створення карт [11].

Рисунок 2.5 – Логотип онлайн застосунку Inkarnate

2.7 Пошук та генерування ігрових зображень

Було обрано спосіб пошуку підходящих зображень та генерацію або зміну зображень за допомогою штучного інтелекту, через відсутність навичок для самостійного створення графічних зображень для насичення відеогри.

Деякі зображення для відеогри знайдено в Інтернеті або на відповідних сайтах. Деякі зображення генерувалися за допомогою штучного інтелекту. Здебільшого використовувався штучний інтелект Leonardo AI. Його рекомендували різні особи, а також в нього більш розширений безкоштовний потенціал в порівнянні з іншими. Кожний день в цьому сервісі оновлюються безкоштовні кредити для генерації зображень. Також можна вказувати за основу своє зображення або надавати зображення в якості основи стилю.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

Ознайомимося з структурою файлової системи розроблюваного застосунку всередині платформи Unity (рисунок 3.1).

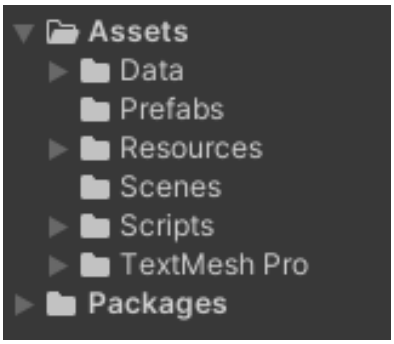

Рисунок 3.1 – Файлова структура застосунку

Проєкт містить головний каталог Assets, який зберігає в собі всі необхідні частини застосунку. До нього входять папки: Data, Prefabs, Resources, Scripts, Scenes.

Присутній каталог Packages, який включає в себе всі підключені пакети, які надають можливість використовувати специфічні методи. Одним з підключених пакетів є Newtonsoft JSON, який дозволяє користуватися технологією JSON.

Каталог Data містить в собі дані потрібні для ігрового застосунку, які зберігаються в JSON файлах. В якості основних даних, що зберігаються в цих файлах можна виділити:

- дані потрібні для початку ігрового процесу;
- збереженні ігрові сеанси;
- сюжетне наповнення.

У відеогрі є локації до яких може зайти користувач та рухатися по сюжету. Так ось сюжетні лінії для кожної локації було вирішено зберігати в окремих файлах з відповідними назвами.

Це дозволяє більш зручніше використовувати інформацію з цих файлів, а також легше їх модифікувати у разі потреби.

Каталог Prefabs включає в себе шаблони об’єктів. Такі шаблони створюються один раз і потім можуть використовуватися в різних місцях, без потрібності повторного налаштування. Наприклад в нашому ігровому застосунку використовується шаблон кнопки вибору. Він потрібен, бо в різних ігрових сценах різна кількість виборів, тому для кожної сцени кнопки виборів програмно повторно створюються по шаблону в потрібній кількості.

У каталозі Resources знаходяться ресурси, які використовуються при наповненні ігрового застосунку. Там є зображення, які використовуються для насичення відеогри, а також аудіо файли, які задають звукове доповнення.

Каталог Scenes містить в собі усі сцени відеогри. Всього є три основні сцени: головне меню, карта ігрового світу та сцена локацій. Редагування графічного наповнення відеогри відбувається саме в цих сценах.

Головне меню складається з кнопок запуску нового ігрового сеансу, завантаження збереженого ігрового прогресу, зміни гучності та кнопки виходу. Зміна гучності відбувається за допомогою перетягування повзунка, який відповідає за неї.

В сцені ігрової карти є чотири кнопки, які відповідають за вхід у локації, а також кнопка налаштувань, яка викликає відповідне вікно.

Сцена з ігровими локаціями містить фонове наповнення, поле в якому виводиться сюжетна лінія для гравця, та кнопки вибору рішення. Може виникнути питання, як може одна сцена використовуватися одразу для всіх локацій. Це відбувається завдяки тому, що до кожного ігрового епізоду прив’язано відповідне посилання на графічне наповнення локації. Тому коли гравець заходить в різні локації він бачить різне оформлення, але з використанням лише однієї сцени.

Каталог Scripts включає в себе всі написані файли скриптів для ігрового застосунку. Як вже зазначалося при написанні коду спираємося на MVC шаблон програмування, тому цей каталог також містить власну

структуру. Він ділиться на три основні каталоги з програмними файлами: Core, Models, UIControllers. Більш детально розберемо каталог із файлами скриптів в наступних розділах.

3.1 Структура каталогу Scripts

Каталог Scripts поділяється на три частини, які використовуються для різних призначень. Частина моделей відповідає за класи об’єкти, дані з яких використовуються під час виконання коду ігрового застосунку. Також є частина, яка відповідає за керування взаємодії з графічних інтерфейсом відеогри.

Каталог Core відповідає за стани в яких перебуває застосунок, за роботу з збереженням або завантаженням даних та логіку гри. На рисунку 3.2 проілюстровано структуру папки Scripts.

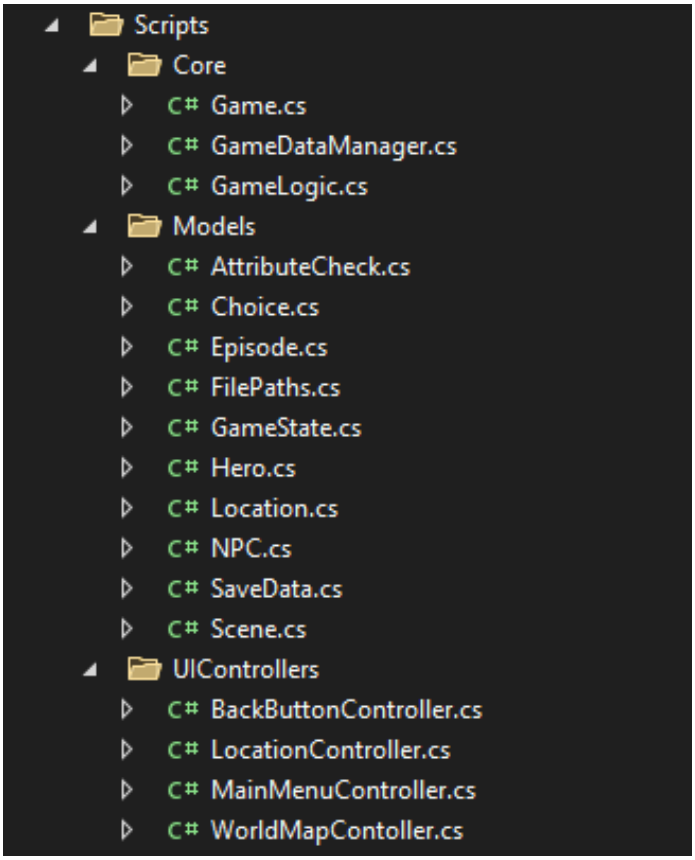

Рисунок 3.2 – Каталог Scripts

3.2 Каталог Models

3.2.1 Класи Location, Episode, Scene

Класи відповідають за представлення ігрового сюжету, який поділяється на локації, епізоди та сцени. Локації містять в собі епізоди, а епізоди – сцени. Сюжет поділений на такі частини для зручності роботи з ним, його модифікації та збереження.

Клас Location відповідає за локації та включає в себе назву, номер та опис локації, а також словник епізодів, які включені до цієї локації (лістинг 3.1).

Саме у вигляді цього класу зберігаємо дані в JSON файлах в якості сюжетної лінії. Назва локації дозволяє гравцю переходити в потрібну локацію після її вибору на ігровій карті.

Відеогра має на увазі лінійне проходження від першої до останньої локації, тому їх номери дозволяють розуміти чи вже було пройдено потрібну локацію, щоб користувач міг перейти до наступної.

Лістинг 3.1 – Властивості класу Location

```
public class Location
{
    public string Name { get; set; }
    public int Number { get; set; }
    public string Description { get; set; }
    public Dictionary<string, Episode>? Episodes { get; set; }
}
```

Клас Episode також містить в собі назву та номер епізоду (лістинг 3.2). Номер епізоду використовується для циклічного проходження по всім епізодам, бо ця частина також є лінійною, на відмінну від частини зі сценами, які зберігаються в епізодах. Додатково в класі є шлях до файлу із зображенням, яке завантажується в якості графічного фону локації та словник з сценами.

Цей клас було створено з двох причин. По-перше, для більш частішої зміни ігрового фону, ніж якщо б у всієї локації був один фон. По-друге для регулювання системи збереження ігрового прогресу.

В ігровому застосунку присутні автоматичне збереження та ручне збереження гравцем. Автоматичне збереження спрацьовує після проходження кожного епізоду, а гравець може зберегти прогрес в будь-який момент самостійно.

Ці два способи відрізняються тим, що при автоматичному збереженні завантажена гра буде завжди починатися з першої сцени епізоду, а якщо гру зберіг користувач, то завантажена гра почнеться саме з тієї сцени на якій він використав функцію збереження.

Так ось епізоди потрібні для того, щоб автоматичне збереження не відбувалося або занадто часто при проходженні кожної сцени, або занадто рідко при проходженні локації.

Лістинг 3.2 – Властивості класу Episode

```
public class Episode
{
    public string Name { get; set; }
    public int Number { get; set; }
    public string SpritePath { get; set; }
    public Dictionary<string, Scene> Scenes { get; set; }
}
```

В класі Scene є два параметри: сюжетний текст сцени та список з виборами (лістинг 3.3). Сюжетний текст виводиться у відповідний текстовий елемент на ігровій сцені, а усі вибори відображаються в якості кнопок вибору.

Лістинг 3.3 – Властивості класу Scene

```
public class Scene
{
    public string? Description { get; set; }
    public List<Choice>? Choices { get; set; }
}
```

3.2.2 Класи Hero та NPC

Клас Hero складається з двох параметрів: ім’я головного персонажу та його атрибутів. Ім’я головного персонажа можна задавати на початку гри, після чого це ім’я буде використовуватися в сюжеті. В JSON файлах, які містять сюжетне наповнення, є позначки, які замінюються на ім’я герою. Атрибути головного герою впливають на можливість здійснення деяких виборів, а також на їх успішність або невдачу

Наприклад, у відеогрі є вибори, які будуть заблоковані у випадку, коли у головного персонажу недостатньо певних атрибутів. Також присутні вибори результатами яких може бути успіх або невдача.

У процесу гри гравець також буде здійснювати вибори, які вплинуть на характеристики персонажу, що надалі вплине на інші вибори.

Властивості класу NPC наведено в лістингу 3.4. В ньому присутні такі наступні властивості: ім’я неігрового персонажу, числове позначення стану відносин NPC з головним героєм та текстове уявлення стану відносин через створений допоміжний перелік.

За допомогою механізму пам’яті NPC та їх стану користувач також обмежується у виборах, які може здійснювати. Наприклад, якщо він у певній ситуації герой не буде мати належний стан з неігровим персонажем, то користувач не зможе обрати цей вибір.

Лістинг 3.4 – Властивості класу NPC

```
public class NPC
{
    public string Name { get; set; }
    public int RelationshipWithHero { get; set; }
    public NPCRelationState relationState { get; set; }
}
```

Також в класі є два методи оновлення стану. Ці методи наведено у лістингу 3.5. Метод UpdateRelationship оновлює числове уявлення стану, а метод UpdateRelationshipType за числовим уявленням оновлює тип стану.

Лістинг 3.5 – Методи UpdateRelationship та UpdateRelationshipType

```
public void UpdateRelationship(int value)
{
    RelationshipWithHero += value;
    if (RelationshipWithHero < 1)
        RelationshipWithHero = 1;
    if (RelationshipWithHero > 12)
        RelationshipWithHero = 12;
    relationState = UpdateRelationshipType();
}
private NPCRelationState UpdateRelationshipType()
{
    return RelationshipWithHero switch
    {
        >= 1 and <= 4 => NPCRelationState.Enemy,
        >= 5 and <= 8 => NPCRelationState.Neutral,
        >= 9 and <= 12 => NPCRelationState.Friend,
        _ => relationState
    };
}
```

3.2.3 Класи Choice та AttributeCheck

Клас Choice містить інформацію про вибори гравця (лістинг 3.6). В цьому класі зберігається інформація про текст вибору, наступну ігрову сцену, дозволені атрибути головного герою для здійснення вибору, зміна атрибутів герою, необхідний стан з NPC вибору та зміна стану в разі необхідності.

Кожний вибір має посилання на відповідну для нього ігрову сцену, тобто таким чином і реалізується не лінійність сюжету. Деякі сцени гравець побачить відповідно до його виборів, а деякі – ні.

Також клас містить інформацію, щодо зміни атрибутів та стану NPC. Ці властивості задані не для всіх виборів. Відповідно деякі вибори мають обмеження, щодо допустимого стану атрибутів або відносин з неігровим персонажем.

Лістинг 3.6 – Властивості класу Choice

```
public class Choice
{
    public string? Text { get; set; }
    public string? NextScene { get; set; }
```

```
    public AttributeCheck? Check { get; set; }
    public Dictionary<string, int>? RequiredAttributes { get;
set; }
    public Dictionary<string, int>? AttributeChanges { get; set;
}
    public Dictionary<string, int>? NPCChanges { get; set; }
    public Dictionary<string, List<NPCRelationState>>?
RequiredNpcStates { get; set; }
     }
```

Як можна побачити в класі присутня ще одна властивість типу AttributeCheck (лістинг 3.7), яка відповідає за можливість успіху або невдачі в момент прийняття одного з рішень.

Наприклад, якщо гравець обирає варіант для якого потрібен рівень певного атрибути рівний десяти, а в головного героя лише шість, то користувач отримає в результаті невдачу.

Лістинг 3.7 – Властивості класу AttributeCheck

```
public class AttributeCheck
{
    public string Attribute { get; set; }
    public int Threshold { get; set; }
    public string OnSuccess { get; set; }
    public string OnFail { get; set; }
}
```

Властивість атрибут визначає за яким атрибутом виконуємо перевірку. Властивість мінімального порогу, який потрібно перетнути для успіху та два результати на випадок успіху або невдачі.

3.2.4 Класи GameState, SaveData та FilePaths

Клас GameState відповідає за місце у відеогрі, в якому знаходиться користувач у реальний момент часу. Він містить ідентифікатори локації, епізоду та сцени в яких може знаходитись гравець. Він постійно оновлює ці дані. Також присутній словник з пройденими локаціями, який використовується для того, щоб користувач проходив локації послідовно. В

цей словник записуються номери вже пройдених локацій раніше. Цей клас є однією з основних частин, які зберігаємо для подальшого продовження ігрового процесу, щоб потім продовжити з потрібної точки.

Клас SaveData (лістинг 3.8) містить в собі всі необхідні властивості, які потрібно збігати. Клас зберігає всі властивості з класів GameState та Hero та деяку інформацію про NPC та час збереження.

Лістинг 3.8 – Властивості класу SaveData

```
public class SaveData
{
    public string HeroName { get; set; }
    public string CurrentLocationId {  get; set; }
    public string CurrentEpisodeId { get; set; }
    public string CurrentSceneId { get; set; }
    public Dictionary<string, int> CompletedLocations { get;
set; }
    public Dictionary<string, int> HeroAttributes { get; set; }
    public Dictionary<string, NPC> NpcMemory { get; set; }
    public string SaveTime { get; set; }
}
```

Клас FilePaths – статичний клас, який містить незмінні шляхи. В ньому зберігаються шляхи до файлів з сюжетним наповненням, файлів збережень ігрового прогресу та файлу з стандартними даними, які використовуються для початку нового ігрового сеансу.

3.3 Каталог UIControllers

3.3.1 Базовий клас MonoBehaviour

MonoBehaviour – клас, який дозволяє користуватися спеціальними методами, які дають можливість взаємодіяти з Unity об’єктами. Всі класи контролери успадковуються від цього базового класу. Цей клас дає можливість використовувати наступні основні методи: Start, Awake, Destroy, DontDestroyOnLoad, GetComponent, FindGameObjectWithTag та інші.

Start метод виконується перед завантаженням сцени для об’єкту, з яким цей метод працює. Destroy або DontDestroyOnLoad методи, що видаляють ігровий об’єкт або забороняють автоматично знищувати об’єкт відповідно, наприклад під час завантаження іншої сцени.

GetComponent дозволяє отримати дочірній компонент об’єкту до якого звертаємося. Також є схожий метод GetComponentInChildren, який дозволяє знайти потрібний компонент у прив’язаному до потрібного об’єкта дочірнього об’єкту.

Метод FindGameObjectWithTag допомагає знайти об’єкт на сцені за допомогою його тегу. Теги можна створювати власноруч всередині платформи Unity та потім прикріпляти за потрібним об’єктом.

Альтернативою слугує метод, коли в самому інспекторі Unity прикріпляємо потрібний об’єкт до змінної файлу скрипта, який також прив’язано до іншого об’єкту [12].

3.3.2 Контролер MainMenuController

Клас MainMenuController бере на себе задачу керування об’єктами першої сцени відеогри. Він містить в собі функціонал для обробки кнопок запуску нової гри та завантаження збереженого прогресу.

Метод OnStartGameClick прикріплено до відповідної кнопки та він відповідає за активацію панелі з введенням ігрового імені персонажу. Функція OnApplyHeroNameAndStartGameClick (лістинг 3.9) оброблює натискання на кнопку підтвердження імені та його передачу до змінної класу Hero.

Лістинг 3.9 – Метод OnApplyHeroNameAndStartGameClick

```
public void OnApplyHeroNameAndStartGameClick()
{
    string heroName =
heroNamePanel.GetComponentInChildren<TMP_InputField>().text;
    if (heroName.Count() <= 1)
```

```
    {
        return;
    }
    var saveData =
GameDataManager.Instance.LoadGame(FilePaths.DefaultGameSettingsP
ath);
    saveData.HeroName = heroName;
    Game.Instance.InitGame(saveData);
    heroNamePanel.SetActive(false);
    SceneManager.LoadScene("WorldMap");
}
```

Метод ShowSaveList прив’язано до кнопки завантаження ігрового прогресу. Цей метод відповідає за відображення панелі з кнопками виборами можливих збережених ігрових сеансів. Для початку видаляємо всі раніше створені кнопки для випадку, якщо це буде вважатися вже застарілим варіантом. Перевіряємо каталог, в якому зберігається всі файли збережень та за допомогою методу GetFiles записуємо шляхи до них у потрібну змінну.

Далі перебираємо масив шляхів та звертаємося до кожного з файлів. Отримуємо інформацію з файлів до змінної. В цей момент не потрібні всі дані з файлів, а тільки час збереження та ім’я головного герою, щоб при виборі збереженого ігрового процесу користувач розумів, яке збереження йому потрібне. Для кожної кнопки додаємо обробник події натискання на неї. У випадку натискання на кнопку переходимо до ігрової карти та вже повторно зчитуємо обраний файл збереження та отримуємо необхідні дані.

3.3.3 Контролер SettingsController

Контролер SettingsController відповідає за меню налаштувань відеогри. Об’єкт цього класу створюється один раз на початку гри на першій сцені та залишається доступним на всіх сценах допоки застосунок не буде закрито.

Для того, щоб забезпечити наявність лише одного екземпляра цього об’єкта використовується метод Awake (лістинг 3.10). Цей метод викликається під час ініціалізації об’єкта до того як буде виповнено функцію Start.

Лістинг 3.10 – Метод Awake

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

В методі перевіряється чи існує екземпляр SettingsController. Якщо він існує, то видаляємо цей об’єкт та одразу ж виходимо з методу. Якщо екземпляра не існує, то зберігаємо поточний об’єкт та забороняємо видаляти його при завантаженні нової сцени використовуючи метод DontDestroyOnLoad.

В класі SettingsController присутній функціонал, який відповідає за можливість регулювання гучності музики та звукових ефектів та перемикання режимів відображення відеогри. Ці методи отримують оновлені дані від об’єктів графічного інтерфейсу та виконують відповідні зміни для них.

Так, як на всіх сценах зберігається один екземляр цього класу, тому треба коректно відмальовувати кнопки навігації по ігровому застосунку. Це здійснюється за допомогою отримання поточної сцени та в залежності від неї на екрані відмальовуються потрібні кнопки. Меню налаштувань містить в собі кнопку збереження ігрового прогресу налаштувань. Вона просто викликає метод SaveGame з класу GameDataManager.

3.3.4 Контролер WorldMapContoller

Контролер WorldMapContoller аналізує, яка саме локація була обрана на ігровій карті та запрошує завантаження сюжетної лінії саме для цієї локації.

Також контролер слідкує за тим, чи коректну локацію було обрано користувачем. Це відбувається в методі CanEnterLocation (лістинг 3.11), який слідкує за тим, щоб інформація вірно завантажилася, а також перевіряє чи була пройдена попередня локація або обрана локація і виводить відповідне повідомлення на сцені, якщо до локації не можна зайти.

Лістинг 3.11 – Метод CanEnterLocation

```
private bool CanEnterLocation(Location location)
{
    if (location == null) { return false; }
    if (location.Episodes == null || location.Episodes.Count ==
0) {
        return false;
    }
    bool previousLocationCompleted =
Game.Instance._gameState.CompletedLocations.
        Any(pair => pair.Value == location.Number - 1);
    bool chosedLocationCompleted =
Game.Instance._gameState.CompletedLocations.
        Any(pair => pair.Value == location.Number);
    if (chosedLocationCompleted){
        messagePanel.GetComponentInChildren<TMP_Text>().text =
$"Ви вже пройшли локацію {location.Name}";
        return false;
    }
    if (!previousLocationCompleted && location.Number != 1){
        messagePanel.GetComponentInChildren<TMP_Text>().text =
$"Спочатку пройдіть попередню локацію";
        return false;
    }
    return true;
}
```

Повідомлення, яке користувач бачить у разі не коректного вибору локації залишається на сцені близько двох секунд після чого починає зникати.

Цей механізм контролює метод FadeMessage (лістинг 3.12). У методі встановлюємо значення часу життя повідомлення та часу його поступового зникнення. Після чого чекаємо, коли час життя повідомлення закінчиться та переходимо до циклу плавного зникнення, за допомогою зменшення прозорості. Чекаємо коли повідомлення зникне та вимикаємо панель з ним.

Лістинг 3.12 – FadeMessage

```
private IEnumerator FadeMessage()
{
    float messageDuration = 2f;
    float fadeDuration = 0.5f;
    CanvasGroup canvasGroup =
messagePanel.GetComponent<CanvasGroup>();
    canvasGroup.alpha = 1;
    yield return new WaitForSeconds(messageDuration);
    float elapsedTime = 0;
    while (elapsedTime < fadeDuration){
        canvasGroup.alpha = Mathf.Lerp(1, 0, elapsedTime /
fadeDuration);
        elapsedTime += Time.deltaTime;
        yield return null;
    }
    canvasGroup.alpha = 0;
    messagePanel.SetActive(false);
}
```

3.3.5 Контролер LocationController

Контролер LocationController відповідає за керування сценою з ігровими локаціями. В класі у функції Start запускається безпосередньо сюжетна лінія локації.

Метод StartCoroutine запускає корутину для переданого до неї методу. У нашому випадку передаємо RunEpisodesCoroutine, а вже потім в цьому методі також викликаємо StartCoroutine та передаємо до неї RunSceneCoroutine.

Метод RunEpisodesCoroutine – корутина в якій перебираємо у циклі всі доступні для локації епізоди. У ньому враховується випадок, коли користувач заходить до гри із збереженим прогресом. В такому випадку функція пропускає всі епізоди до потрібного, спираючись на переданий ідентифікатор епізоду. Для епізодів викликаємо функцію RunSceneCoroutine, яка в циклі перебирає всі сцени в епізоді.

Функція RunSceneCoroutine на початку перевіряє чи передали до неї збережений прогрес. Перевірка відбувається за булевим прапорцем, що передається до функції в якості параметру. Якщо він встановлений в

значення правди, то епізод починається з потрібної сцени за вказаним ідентифікатором. Якщо він встановлений в неправду, то епізод починається з першої сцени. Також заповнюємо список можливих виборів (лістинг 3.13).

Потім потрапляємо у цикл, який триває поки у словнику сцен присутня сцена відповідна до потрібного ідентифікатору, який отримуємо з класу Choice після здійснення вибору. У циклі заповнюємо текстовий елемент з сюжетом та видаляємо всі кнопки виборів, щоб згодом створити нові (лістинг 3.14). Ця операція проводиться, бо не в кожній сцені однакова кількість виборів, тому для кожної сцени заново створюються кнопки виборів за допомогою переданого шаблону. Деякі заблоковані вибори не з’являться на сцені у вигляді кнопки через певні умови, які описуються в класі GameLogic.

Також після кожного вибору в разі потреби оновлюємо параметри для головного герою або стан NPC.

Лістинг 3.13 – Перехід до потрібної сцени та заповнення можливих виборів

```
if (!continueFromSaved ||
string.IsNullOrEmpty(_gameState.CurrentSceneId) ||
!scenes.ContainsKey(_gameState.CurrentSceneId)){
    _gameState.CurrentSceneId = scenes.First().Key;
}
GameDataManager.Instance.SaveGame(isAuto: true);
while (scenes.ContainsKey(_gameState.CurrentSceneId))
{
    var scene = scenes[_gameState.CurrentSceneId];
    descriptionText.text = scene.Description;
    ClearChoicesUI();
    var availableChoices = new List<Choice>();
    if (scene.Choices != null)
    {
        foreach (var choice in scene.Choices)
        {
            If (GameLogic.Instance.IsChoiceAvailable(choice))
            {
                availableChoices.Add(choice);
            }
        }
     }
if (availableChoices.Count == 0){
    yield break;
}
```

Лістинг 3.14 – Створення кнопок виборів та очікування вибору користувача

```
bool choiceMade = false;
Choice selectedChoice = null;
foreach (var choice in availableChoices)
{
    var btnObj = Instantiate(choiceButtonPrefab,
choicesContainer);
    var btnText = btnObj.GetComponentInChildren<TMP_Text>();
    var btn = btnObj.GetComponent<Button>();
    btnText.text = choice.Text;
    btn.onClick.AddListener(() =>
    {
        selectedChoice = choice;
        choiceMade = true;
    });
}
yield return new WaitUntil(() => choiceMade);
GameLogic.Instance.ApplyAttributeChanges(selectedChoice);
GameLogic.Instance.ApplyNPCMemoryChanges(selectedChoice);
if (!string.IsNullOrEmpty(selectedChoice.NextScene) &&
scenes.ContainsKey(selectedChoice.NextScene))
{
    _gameState.CurrentSceneId =  electedChoice.NextScene;
}
else{
    break;
}
```

На сцені присутня кнопка, що використовується для відображення параметрів головного герою. До цієї кнопки прив'язаний метод OnHeroStatsClick (лістинг 3.15). Він оновлює текстові об’єкти новими даними та активує відповідну панель, на якій вони відображаються. В цій функції здійснюється оновлення текстових даних за допомогою позначок.

Лістинг 3.15 – Метод OnHeroStatsClick

```
public void OnHeroStatsClick()
{
    var attributesList = _game._hero.Attributes.ToList();
    titleHeroStats.text = titleHeroStats.text.Replace("{hero}",
_game._hero.Name);
    for (int i = 0; i< stats.Length; i++){
        stats[i].text = stats[i].text.Replace("{stat}",
attributesList[i].Value.ToString());
    }
    statsPanel.SetActive(true);
}
```

У контролері також є метод LocationEnd, який викликається під час завершення локації. Коли локацію завершено виводиться відповідна панель та гравець може перейти до наступної локації.

3.4 Класи каталогу Core

Клас Game зберігає в собі всю інформацію про ігровий сеанс. Це синглтон, який створюється один раз на початку запуску застосунку. В ньому зберігається інформація про героя, локації, стан гри, NPC.

Цей клас прив’язано до відповідного об’єкту на першій сцені головного меню в Unity. Цей клас містить в собі Awake метод, який відповідає за той самий функціонал, як і у випадку з SettingsConrotller.

Також в Game є метод InitGame, який ініціює всі потрібні властивості цього класу. Всі дані для ініціювання отримуємо з JSON файлів.

Клас GameDataManager відповідає за методи завантаження та збереження даних, які використовуються протягом ігрового сеансу, а також для отримання сюжетного наповнення для переданої локації.

Цей клас використовує SaveData , який містить всі потрібні властивості, які потрібно завантажити з файлової системи або зробити їх збереження. Саме в цей клас завантажується вся інформація. Для збереження даних використовуємо клас JsonConvert та метод класу SerializeObject. Передаємо до методу об’єкт типу SaveData та зберігаємо дані до файлу.

Для завантаження даних з JSON файлу викликаємо метод DeserializeObject з потрібним типом. Для використання цієї функціональності було підключено пакет Newtonsoft JSON, який надає можливість використовувати методи класу JsonConvert.

Клас GameLogic містить в собі методи ігрової логіки. В ньому є два методи для оновлення параметрів герою та NPC. Оновлені параметри функції отримуємо з класу Choice. Метод класу IsChoiceAvailable (лістинг 3.16) перевіряє чи доступний переданий до нього вибір. Для характеристик

персонажу перевіряється чи є вони допустимими та чи достатньо їх для здійснення вибору. Для NPC перевіряється чи підходящий стан відносин має головний герой з неігровим персонажем.

Лістинг 3.16 – Метод IsChoiceAvailable

```
public bool IsChoiceAvailable(Choice choice)
{
    if (choice.RequiredAttributes != null)
    {
        Hero hero = Game.Instance._hero;
        foreach (var attr in choice.RequiredAttributes)
        {
            int heroValue = hero.GetAttribute(attr.Key);
            if (heroValue < attr.Value) {
                return false;
            }
            if (heroValue > 100 || heroValue < 0) {
                return false;
            }
        }
    }
    if (choice.RequiredNpcStates != null)
    {
        foreach (var pair in choice.RequiredNpcStates)
        {
            string npcName = pair.Key;
            List<NPCRelationState> allowedStates = pair.Value;
            foreach (var npc in Game.Instance._npcs)
            {
                if (npc.Key == npcName)
                {
                    if
(!allowedStates.Contains(npc.Value.relationState))
                    {
                        return false;
                    }
                }
            }
        }
    }
    return true;
}
```

У функції ResolveChoice (лістинг 3.17) передаємо Choice та порівнює значення характеристики вказане в властивості Check зі значенням відповідної характеристики головного герою.

Якщо значення характеристики герою більше, ніж потрібне значення, то користувача буде напрямлено до наступної сцени вказаної в властивості OnSuccess. Якщо значення характеристики герою менше, то наступна сцена буде взята з поля OnFail.

Лістинг 3.17 – ResolveChoice

```
public string ResolveChoice(Choice choice)
{
    if (choice.Check != null)
    {
        var attribute = choice.Check.Attribute;
        var threshold = choice.Check.Threshold;
        var heroValue =
Game.Instance._hero.Attributes.ContainsKey(attribute)?
Game.Instance._hero.Attributes[attribute] : 0;
        if (heroValue >= threshold)
        {
           return choice.Check.OnSuccess;
        }
        else
        {
            return choice.Check.OnFail;
        }
    }
    else
    {
        return choice.NextScene;
    }
}
```

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Після запуску ігрового застосунку користувач побачить перед собою першу сцену з головним меню проілюстровану на рисунку 4.1.

Рисунок 4.1 – Головне меню

На ігровому меню гравець побачить назву гри та три кнопки. Перша з кнопок відповідає за початок нового ігрового сеансу, тобто сеансу з нульовим ігровим прогресом.

Друга кнопка відповідає за завантаження набутого раніше прогресу. Цей прогрес автоматично оновлюється під час гри в моменти переходу між сюжетними епізодами або користувач може виконати збереження прогресу сам, використавши відповідну функцію на одній з наступних ігрових сцен. Після натискання на кнопку завантаження гравець побачить невеличке вікно з можливістю вибору одного з збережених файлів прогресу (рисунок 4.2). Потрібно натиснути на одне з них, щоб продовжити ігровий сеанс.

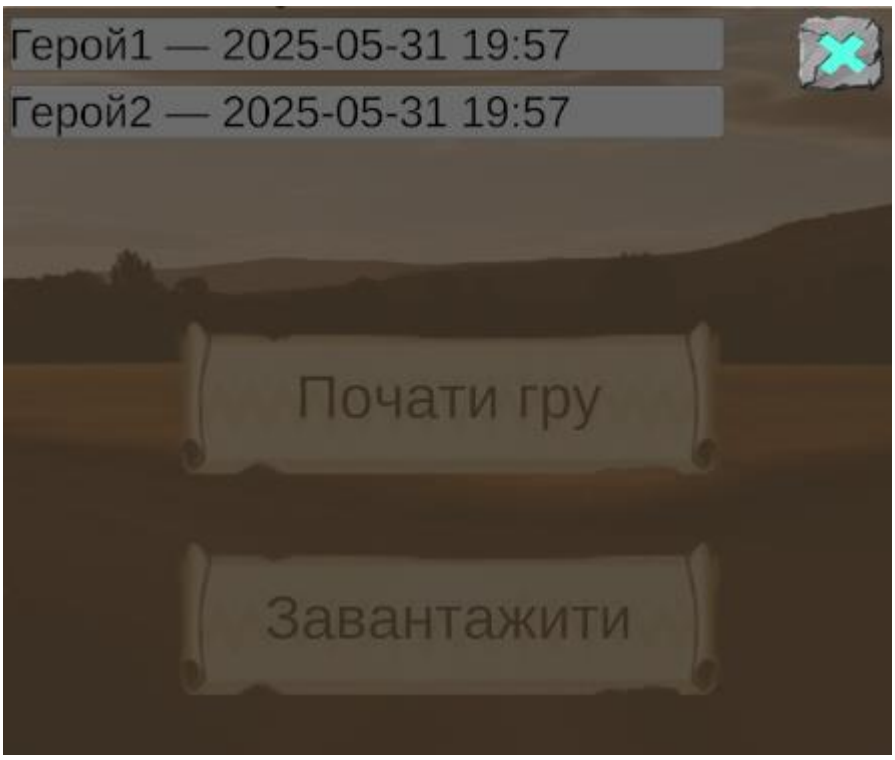

Рисунок 4.2 – Меню вибору збереження

Якщо розглядати випадок, коли гравець натискає на кнопку початку нової гри, то після цього потрібно буде ввести ім’я для ігрового персонажу у відповідне поле (рисунок 4.3). Для імені є тільки два обмеження:

- не повинно бути пустим;
- повинно складатися хоча б з двох символів або літер.

Після того, як користувач заповнить поле з ім’ям та підтвердить його, відкриється сцена з картою ігрового світу.

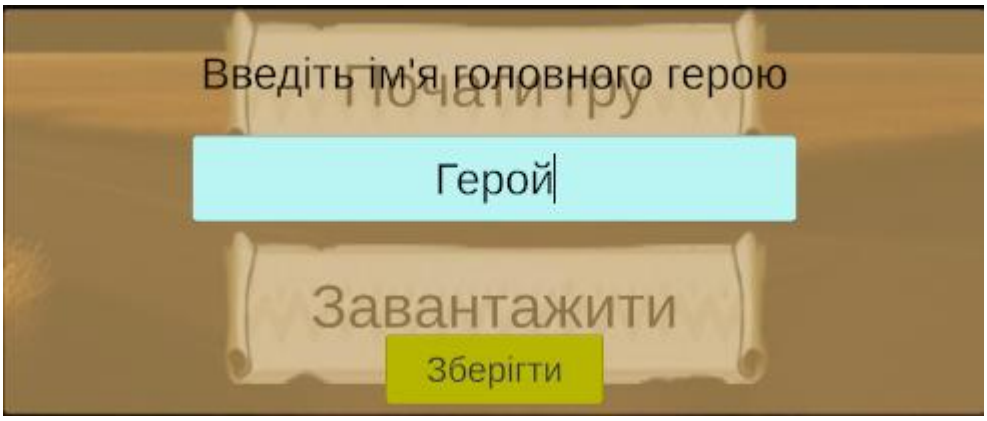

Рисунок 4.3 – Поле для імені персонажу

Остання кнопка на сцені викликає меню налаштувань (рисунок 4.4), в якому можна налаштувати гучність фонової музики у грі, гучність ефектів під час взаємодії з графічним інтерфейсом та режим відображення, тобто у вигляді вікна або повноекранний. Також в цьому вікні є кнопки, які відповідають за навігацію у застосунку. За допомогою цих кнопок користувач може пересуватись по сценам відеогри.

Рисунок 4.4 – Меню налаштувань

На рисунку 4.5 зображена ігрова карта світу, на якій бачимо чотири кнопки локацій та кнопка виклику меню налаштувань. Користувач може обрати на карті одну з локацій, до якої хоче зайти.

На даний момент в ігровому застосунку передбачено, сюжетною лінією, послідовне проходження локацій від першої до останньої. Локації потрібно проходити в наступному порядку: Village, Forest, Church, Town. Закінченням відеогри, можна вважати проходження всіх чотирьох локацій.

Рисунок 4.5 – Ігрова карта світу

Також, наприклад, якщо користувач обирає другу локацію не пройшовши попередньо першу, то він побачить відповідне сповіщення (рисунок 4.6). Також можливий випадок, коли гравець обирає ту локацію, яку вже пройшов до цього. Для такого випадку також є відповідне повідомлення. Це повідомлення затримується на ігровому екрані близько двох секунд після чого зникає.

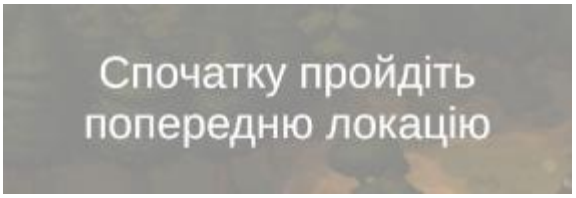

Рисунок 4.6 – Повідомлення

Після того, як користувач обрав локацію, він побачить відповідну сцену з обраною ним локацією (рисунок 4.7). На цій сцені знаходяться кнопки вибору рішення, кнопка налаштувань та кнопка перегляду статистики ігрового персонажу.

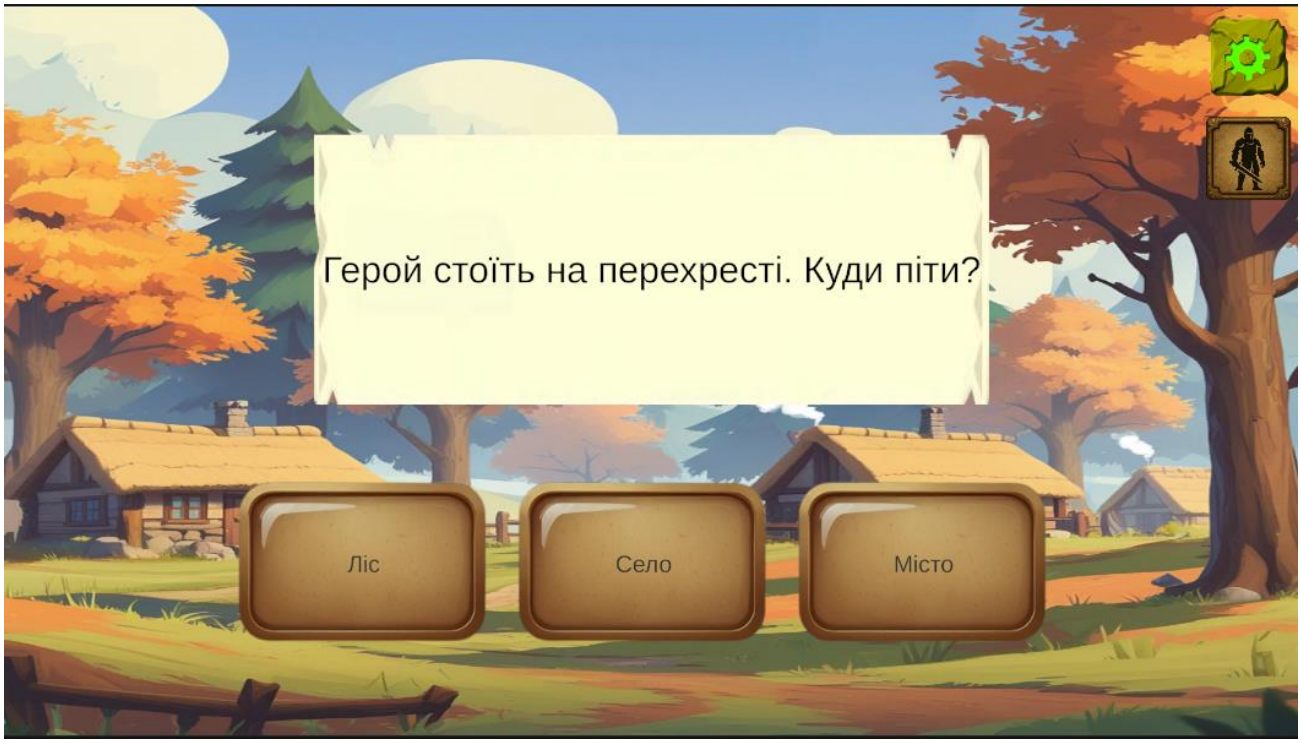

Рисунок 4.7 – Локація Village

Після натискання на кнопку перегляду статистики головного герою відкриється відповідне вікно (рисунок 4.8). В ньому можна переглянути параметри, які на даний ігровий момент має головний герой. По цим параметрам можна додатково розуміти свої шанси під час деяких виборів.

Рисунок 4.8 – Характеристики персонажу

Також на сцені з локацією є текстовий елемент, в якому описується ігрова ситуація. Після того, як гравець обере один з запропонованих виборів, він перейде до наступної сюжетної сцени, а при зміні епізодів буде змінюватися ігровий фон. Після кожного епізоду відбувається автоматичне збереження прогресу. Для ручного збереження слід натиснути на кнопку налаштувань. У вікні, що з'явилося буде присутня нова кнопка для збереження ігрового прогресу.

Після проходження локації виведеться відповідне повідомлення та можна буде приступити до іншої.

Отже, основна ціль гравця – дійти до кінця сюжетної лінії та отримати відповідну, до проходження, кінцівку. Якщо гравець налаштований отримати гарну кінцівку, то під час ігрового процесу йому необхідно приймати найбільш вигідні рішення.

Обрані гравцем рішення будуть впливати на характеристики головного герою, які в свою чергу впливають на розблокування деяких кращих виборів та збільшення відсотків на успіх в деяких ситуаціях. Зміну характеристик після вибору користувач зможе перевіряти натискаючи на відповідну кнопку графічного інтерфейсу. Гравець зможе аналізувати свій вибір для наступних спроб.

Також щодо успіху або невдачі при виборах, де можливий такий результат. Для гравця не буде явно показано чи було отримано успішне рішення чи ні. Користувач може здогадуватись про це, лише покладаючись на сюжетний опис. Це додасть складності під час проходження та прийняття ігрових рішень.

Тому для отримання кращої кінцівки треба продумувати свої рішення. Але в цілому не всі гравця прагнуть отримати саме найкращу кінцівку. Деякі просто хочуть провести свій час в ігровому світі та приймати рішення, які їм подобаються більше.

ВИСНОВКИ

Досягнуто поставлену мету на кваліфікаційну роботу, а саме розробка інтерактивної відеогри новели на основі Unity з механізмом прийняття рішень.

Виконано всі поставлені завдання та досягнуто цілей роботи. Створено повноцінний ігровий застосунок з сюжетним наповненням в 2D форматі. Тестування застосунку виявило багату кількість невідповідностей та багів, з яких більшість було усунуто та перевірено коректність роботи.

Для реалізації поставленої задачі використано інтерактивну середу розробки Visual Studio 2022, мову програмування C#, ігровий двигун Unity, технологію зберігання даних JSON, інструмент створення ігрових карт Inkarnate.

В цілому при розробці ігрового застосунку було досягнуто потрібних цілей, але можливі подальші вдосконалення, які плануються бути втіленими в майбутньому. Ось перелік вдосконалень та додаткової функціональності, які можна було б втілити в подальших версіях:

- реалізація більшої кількості механізмів, які могли б впливати на вибори гравця у процесі відеогри;
- покращення графічних елементів відеогри;
- вдосконалення сюжетної лінії та можливе розширення ігрової карти світу в разі потреби;
- втілення деякої системи статистики, в якій гравець зможе продивитися певну інформації про свої ігрові сеанси;
- локалізація застосунку;
- реалізація застосунку помічника, для зручної роботи з сюжетом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Breault M. Narrative design: the craft of writing for games. Taylor & Francis Group, 2020. 192 p.

2. Interactive storytelling for video games: a player-centered approach to creating memorable characters and stories / ed. by K. Chris. Burlington, MA : Focal Press, 2011. 319 p.

3. Zimmerman E., Salen K. Rules of play: game design fundamentals. The MIT Press, 2003. 688 p.

4. Swink S. Game feel: a game designer's guide to virtual sensation. Taylor & Francis Group, 2008. 376 p.

5. Koster R. Theory of fun for game design. O'Reilly Media, Incorporated, 2013.

6. Unity manual. *Unity - Manual: Unity 6.1 User Manual*. URL: https://docs.unity3d.com/Manual/index.html (date of access: 03.06.2025).

7. Unity - scripting API:. *Unity - Manual: Unity 6.1 User Manual*. URL: https://docs.unity3d.com/ScriptReference/ (date of access: 03.06.2025).

8. Visual Studio product family documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/visualstudio/ (date of access: 04.06.2025).

9. C# Guide - .NET managed language. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (date of access: 08.06.2025).

10. Freeman A. Pro ASP. NET MVC 5. Springer, 2014.

11. Inkarnate - create fantasy maps online. *Inkarnate - Create Fantasy Maps Online*. URL: https://inkarnate.com/faq/ (date of access: 07.06.2025).

12. Lnc m., William E. Learning C# by developing games with unity: c# programming for unity game development - 2020. Independently Published, 2020.