НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Ігровий проєкт на основі Unity»

Виконав:
студент IV курсу, групи ІС-91
Гідзун Денис Володимирович _____

Керівник:
Ст. викладач, к.т.н.
Орленко Сергій Петрович _____

Рецензент:
Доцент каф. ОТ, к.т.н.,
Порєв Віктор Миколайович _____

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

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Олександр РОЛІК

«___»_____20__р.

ЗАВДАННЯ

на дипломний проєкт студенту

Гідзуну Денису Володимировичу

1. Тема проєкту «Ігровий проєкт на основі Unity», керівник проєкту Орленко Сергій Петрович, к.т.н., старший викладач, затверджені наказом по університету від «31» травня 2023р. №2101-с.

2. Термін подання студентом проєкту: «12» червня 2023р.

3. Вихідні дані до проєкту: *Технічне завдання.*

4. Зміст пояснювальної записки:
1. Аналіз предметної області та планування розробки гри: сучасний стан ігрової індустрії, огляд ігрових движків, обґрунтування вибору движка unity для розробки 2d платформера, концепція та ідея гри, основні механіки гри та їхній вплив на геймплей, аналіз схожих ігор на ринку та їх особливостей, визначення цільової аудиторії для "fox adventure", стратегія розробки та планування проєкту.
2. Інформаційне забезпечення: вхідні дані, вихідні дані, вибір та імпорт графічних ассетів та звуків, імпорт та налаштування ассетів, налаштування інтерфейсу користувача.
3. Розробка компонентів гри: засоби розробки, створення карт рівнів за допомогою tilemap, створення ігрового персонажа, опис інтерактивних об'єктів і елементів середовища, загрозливі об'єкти, інтерфейс користувача, організація коду та структура проєкту, збереження даних гри.
4. Математичний розділ: змістовна постановка задачі, звичайний ворог, ворог на базі алгоритму A.*

5. Технологічний розділ: керівництво користувача, Тестування функціональних вимог.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
1. Діаграма станів і переходів.
2. Діаграма класів.
3. Діаграма просторів імен.
4. Діаграма варіантів використання.

6. Дата видачі завдання 1 березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	25.02.23	
2.	Аналіз предметної області	17.04.23-23.04.23	
3.	Огляд існуючих аналогів	24.04.23-30.04.23	
4.	Постановка та формалізація задачі	01.05.23-07.05.23	
5.	Опис функціональності програмного забезпечення	08.05.23-14.05.23	
6.	Аналіз розробленого рішення	15.05.23-21.05.23	
7.	Оформлення текстової документації	22.05.23-8.06.23	

Студент Денис ГІДЗУН

Керівник проєкту Сергій ОРЛЕНКО

АНОТАЦІЯ

Гідзун Д.В. Ігровий проєкт на основі Unity. КПІ ім. Ігоря Сікорського, Київ, 2023.

Пояснювальна записка дипломного проєкту містить 70 с. тексту, 28 рисунків, 20 таблиць, посилання на 15 літературних джерела та 4 кресленики.

Ключові слова: Unity, розробка гри, 2D відеогра, інтерактивні елементи, механіки гри, стратегічне планування, компоненти гри.

Об’єктом розробки є комп’ютерна гра на основі Unity.

Мета розробки – створення ефективної, гнучкої та цікавої гри, спрямованої на задоволення потреб широкого спектру гравців.

У рамках даного проєкту було створено повноцінний ігровий продукт, використовуючи сучасні технології та інструменти розробки. Для реалізації цієї мети було використано інструменти та технології, включаючи Unity як основний рушій для розробки гри, мову програмування C# для написання ігрової логіки, Rider як основне середовище розробки та Git для контролю версій.

Результатом роботи є повноцінна 2D відеогра, яка представляє унікальний ігровий досвід з використанням інноваційних механік гри та інтерактивних елементів.

Отримані результати можуть бути корисними при розробці аналогічних або подібних ігрових проєктів, де важливим є поєднання стратегічного планування, креативного вирішення проблем і захоплюючого геймплею.

ABSTRACT

Gidzun D.V. Game project based on Unity. KPI named after Igor Sikorsky, Kyiv, 2023.

The explanatory note of the diploma contains 70 pages of text, 28 figures, 20 tables, references to 15 literary sources and 4 drafts.

Keywords: Unity, game development, 2D video game, interactive elements, game mechanics, strategic planning, game components.

The object of development is a computer game based on Unity.

The aim of the development is to create an efficient, flexible and interesting game, aimed at satisfying the needs of a wide range of players.

As part of this project, a full-fledged game product was created using modern technologies and development tools. To achieve this goal, tools and technologies were used, including Unity as the main engine for game development, the C# programming language for writing game logic, Rider as the main development environment, and Git for version control.

The result of the work is a full-fledged 2D video game that provides a unique gaming experience using innovative game mechanics and interactive elements.

The results obtained may be useful in the development of similar or similar gaming projects, where the combination of strategic planning, creative problem solving, and captivating gameplay is important.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС91.030БАК.004 ПЗ	Пояснювальна записка	70		
6	А3	ІС91.030БАК.004 Д1	Діаграма станів і	1		
7			переходів			
8	А3	ІС91.030БАК.004 Д2	Діаграма класів	1		
9						
10	А3	ІС91.030БАК.004 Д3	Діаграма просторів імен	1		
11						
12	А3	ІС91.030БАК.004 Д4	Діаграма варіантів	1		
13			використання			
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	І С91.030БАК.004 ТП		
Розроб.		Гідзун Д.В.			Ігровий проєкт на основі Unity. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Орленко С.П.				Т	
						КПІ ім. Ігоря Сікорського	
Затв.							

Пояснювальна записка

до дипломного проєкту

на тему: «Ігровий проєкт на основі Unity»

Київ – 2023 року

ЗМІСТ

ВСТУП

Сучасне життя неможливо уявити без цифрових технологій. Вони проникли в усі сфери нашого буття - від освіти та медицини до розваг і спорту. Особливе місце в цьому списку займають комп'ютерні ігри - вони давно перестали бути просто засобом розваги і перетворились на важливу частину культури, справжню індустрію, що впливає на мільйони людей по всьому світу.

Актуальність даної дипломної роботи випливає з високої динаміки розвитку галузі розробки комп'ютерних ігор, її важливості в сучасному світі, а також невід'ємного вкладу в культурний та соціальний контекст сучасного суспільства. Крім того, розробка комп'ютерних ігор займає перші позиції серед напрямків IT-індустрії, що визначає значущість та перспективність цього напрямку дослідження.

Об'єктом дослідження є створення 2D платформера на ігровому движку Unity. Цей конкретний процес було вибрано, оскільки він включає ряд специфічних етапів та характеристик, що є ключовими для розуміння механіки розробки ігор цього жанру. Предметом ж дослідження, в свою чергу, є конкретні елементи цього процесу, такі як розробка, тестування та реалізація 2D платформера.

Вибір саме 2D платформера для дослідження обумовлений декількома факторами. По-перше, цей жанр є одним з найбільш поширених і впізнаваних у світі відеоігор, що дозволяє вивчити основи розробки гри на конкретному і добре відомому прикладі. По-друге, розробка 2D платформера має свої специфічні виклики і нюанси, що додає значущості даному дослідженню.

Цей движок є одним з найбільш популярних на ринку, що обумовлено його гнучкістю, потужністю та доступністю для розробників різного рівня.

Метою даної роботи є створення 2D платформера на движку Unity, у процесі якого буде освоєно основні аспекти розробки відеогри, включаючи графічні ресурси, анімацію, програмування, фізику гри, оптимізацію та тестування. Для досягнення цієї мети були поставлені наступні завдання:

– аналіз сучасних ігрових движків та обґрунтування вибору Unity;

– розробка концепції та ідеї 2D платформера;

– освоєння основних компонентів розробки гри на движку Unity;

– розробка процесу оптимізації та тестування гри.

Практичне значення одержаних результатів полягає в можливості застосування отриманих знань та досвіду в реальних проєктах розробки ігор. Отримана інформація та навички також можуть бути використані в освітніх цілях для підготовки спеціалістів в галузі розробки ігор.

Таким чином, дана дипломна робота є цілісним дослідженням процесу розробки 2D платформера, що включає в себе всі основні етапи розробки відеогри від формування ідеї до її реалізації та тестування. Результати цього дослідження можуть слугувати основою для подальших наукових досліджень в цій області, а також застосовуватися на практиці в процесі розробки ігор.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПЛАНУВАННЯ РОЗРОБКИ ГРИ

1.1. Сучасний стан ігрової індустрії

Відео ігрова індустрія за останні десятиліття переживає бурхливий розвиток, стаючи однією з найбільш прибуткових та динамічних галузей розваг. Поєднуючи елементи технології, мистецтва і дизайну, відеоігри сьогодні відіграють важливу роль у культурі, освіті, науці та, звичайно ж, у сфері розваг.

За даними дослідницької компанії Newzoo [1], глобальний ринок відеоігор продовжує зростати. В 2022 році ринок оцінюється в $185 мільярдів і у 2023 році він оцінюється приблизно в $200 мільярдів, що включає продажі ігор, послуг, обладнання та рекламу. Ключовими драйверами цього зростання є поширення мобільних ігор, екосистема е-спорту, стримінгові платформи та соціальні медіа.

Сучасні ігри включають широкий спектр жанрів та форматів, від простих інді-проєктів до складних багатокористувацьких онлайн-ігор. Серед основних тенденцій в ігровій індустрії можна відзначити поширення VR/AR технологій, використання штучного інтелекту та машинного навчання, розвиток cloud gaming, а також появу нових форм монетизації, таких як NFT та blockchain.

Щодо ринку 2D платформерів, він продовжує зберігати свою актуальність. Цей жанр ігор, який бере свої корені від класичних ігор 80-х та 90-х років, знову набирає популярності завдяки своїй простоті, доступності та ностальгічному чарівництву. Ігри, такі як "Celeste", "Hollow Knight" та "Shovel Knight", показують, що 2D платформери можуть бути не тільки комерційно успішними, але й високо оціненими критиками та гравцями за свою графіку,

Однією з особливостей сучасного ринку 2D платформерів є його різноманітність. Від класичних "run and jump" ігор до більш складних пазл-платформерів, від ретро-піксельної графіки до сучасних художніх стилів, від казуальних мобільних ігор до складних консольних та ПК проєктів - вибір дуже великий.

Крім того, 2D платформери часто використовуються як основа для

інноваційних механік та експериментів з геймплеєм. Вони дозволяють розробникам фокусуватися на створенні унікальних ігрових вражень, замість того, щоб витрачати час і ресурси на високоякісну 3D графіку або складні системи AI.

У цьому контексті, розробка 2D платформера на движку Unity відкриває великі можливості для творчості, експерименту та успіху на ринку відеоігор.

1.2. Огляд ігрових движків

Ігрові движки – це спеціалізоване програмне забезпечення, що використовується для створення відеоігор. Вони надають набір інструментів для розробки, що спрощують створення ігор, дозволяючи розробникам зосередитись на геймплеї та дизайні [15].

Сьогодні на ринку представлено чимало ігрових движків, серед яких Unity, Unreal Engine, Godot, GameMaker, CryEngine та інші. Вибір движка залежить від багатьох факторів, включаючи потреби проєкту, вимоги до процесу розробки, доступність ресурсів для навчання та підтримку з боку спільноти.

Unity – один з найбільш популярних ігрових движків на сьогоднішній день. Він пропонує широкий спектр інструментів для розробки ігор різних жанрів та платформ. Unity підтримує 2D та 3D графіку, має вбудовану систему фізики, дозволяє працювати з анімацією та звуками, а також має потужну систему скриптів на базі C#. Крім того, Unity має велику активну спільноту розробників, що постійно допомагає у вирішенні проблем та розширює можливості движка через сторонні плагіни та бібліотеки [2].

Unreal Engine – потужний ігровий движок, який часто використовується для створення великих трипл-А ігор. Він включає в себе високоякісну систему рендерингу, потужні інструменти для створення AI, фізики, анімації та багато іншого. Основна мова програмування – C++, але також є візуальна система програмування Blueprints [3].

Godot – це вільний і відкритий ігровий движок, який набирає популярності

завдяки своїй гнучкості та легкості в освоєнні. Він підтримує розробку як 2D, так і 3D ігор, має власну мову програмування GDScript, а також підтримує C# [4].

GameMaker Studio – це движок, який спеціалізується на 2D іграх. Його основна перевага це простота освоєння, що робить його вибором номер один для початківців. Хоча для програмування в GameMaker Studio використовується власна мова GML (GameMaker Language), движок також надає можливість використовувати візуальну систему програмування, що дозволяє розробляти ігри без написання коду. Однак, порівняно з Unity і Unreal Engine, GameMaker Studio має більше обмежень щодо 3D розробки і підтримки платформ [4].

Кожен із цих движків має свої сильні та слабкі сторони, але вибір між ними в більшості випадків залежить від конкретного проєкту, його вимог та досвіду команди. Для розробки 2D платформера, як в нашому випадку, Unity є ідеальним вибором завдяки широкому набору інструментів для 2D, мультиплатформності, великій спільноті та легкості освоєння.

1.3. Обґрунтування вибору движка Unity для розробки 2D платформера

В контексті цієї роботи, основними причинами вибору Unity як основного ігрового движка є його підтримка 2D графіки, простота використання, потужність та гнучкість, доступність ресурсів для навчання, а також широкі можливості для оптимізації та портування гри на різні платформи. Вибір Unity для розробки 2D платформера можна обґрунтувати наступними причинами:

– мультиплатформність: Unity підтримує більше 25 різних платформ, включаючи Windows, MacOS, Android, iOS, PlayStation, Xbox, Nintendo Switch та багато інших. Це дозволяє розробникам легко портувати гру на різні платформи;

– 2D та 3D інструменти: Unity надає широкий набір інструментів для роботи як з 2D, так і з 3D графікою. Зокрема, для 2D гри доступні спрайти, анімації, фізика, плиткові карти та багато іншого;

– скриптова мова C#: Unity використовує C# як основну мову програмування,

що є дуже потужною, гнучкою та мовою, що широко використовується;

– спільнота та підтримка: Unity має одну з найбільших спільнот розробників ігор, що означає велику кількість навчальних матеріалів, форумів, готових рішень і скриптів, що доступні для використання;

– Asset Store: Unity Asset Store – це магазин, де розробники можуть придбати та продавати готові активи, включаючи 3D моделі, звуки, скрипти та інше. Це може значно спростити процес розробки та зекономити час;

– інтеграція з іншими інструментами: Unity легко інтегрується з різними інструментами розробки, такими як Visual Studio для редактування коду, Photoshop для обробки графіки, а також Blender для створення та імпорту 3D моделей. Це робить процес розробки більш зручним та ефективним;

– оптимізація та рендеринг: Unity включає в себе потужні інструменти для оптимізації та рендерингу, що дозволяє створювати гладкі та високопродуктивні ігри, незалежно від рівня складності;

– інструменти для тестування та відлагодження: Unity надає вбудовані інструменти для тестування ігор, включаючи систему відлагодження, що дозволяє виявляти та виправляти помилки в коді;

– широкі можливості для монетизації: Unity надає інструменти та сервіси, які допомагають монетизувати гру, включаючи вбудовану підтримку реклами, системи внутрішніх покупок, аналітику та інше.

З урахуванням всіх цих переваг, Unity стає відмінним вибором для розробки 2D платформера. Цей движок дозволяє розробникам зосередитися на головному - створенні унікального ігрового досвіду, забезпечуючи при цьому потужні і гнучкі інструменти для реалізації їх творчих ідей.

1.4. Концепція та ідея гри

Гра має назву "Fox Adventure" – це змістовний 2D платформер, в якому гравець виконує роль хитрої лисиці, що вирушає в складну подорож по рівнях,

збираючи при цьому кристали. Центральну концепцію гри складає не лише необхідність успішного проходження рівнів, а й збір усіх доступних кристалів з метою заробити максимальну кількість зірок.

Для розуміння поведінки гравця та взаємодії з грою було розроблено діаграму варіантів використання, яка допомагає візуалізувати різні дії, що виконуються гравцем, та відповіді системи на ці дії. Це не тільки спрощує процес розробки, але і забезпечує більш цілісне розуміння того, як гравець взаємодіятиме з грою. Дана діаграма наведена у графічних матеріалах (ІС91.030БАК.004 Д4).

Для створення гри було обрано двовимірний стиль (2D), який не тільки дозволяє втілити в життя найрізноманітніші сюжетні лінії та ігрові елементи, але й надає розробникам велику свободу дій у плані візуального оформлення гри. 2D-стиль, на відміну від 3D, також має певні переваги з точки зору технічного виконання та оптимізації, що забезпечує більшу доступність гри для широкого кола гравців.

Щодо назви гри, "Fox Adventure" було обрано з міркувань відображення основної концепції та персонажа гри. Ця назва відразу наголошує на головному герої – лисиці, а слово "Adventure" вказує на пригодницьку суть гри.

Гра розрахована на широку аудиторію, поєднуючи в собі простоту управління, яка приваблива для новачків, і складність проходження рівнів, яка зацікавить більш досвідчених гравців. Кожен рівень вимагає від гравця не тільки реакції та координації, але і стратегічного планування, оскільки для збору всіх кристалів потрібно ретельно обдумати маршрут.

Управління лисицею в грі "Fox Adventure" включає в себе переміщення по рівню вліво та вправо, стрибки, та інтерактивні дії з різними об'єктами на карті, такими як перемикачі. Ці перемикачі можуть активувати або переміщати платформи, дозволяючи лисиці подолати перешкоди та отримати доступ до нових областей.

Гравець має певну кількість здоров'я, яке втрачається при зіткненні з противниками або пастками. Це додає елемент стратегії, оскільки гравець повинен

обирати найбезпечніший маршрут і уникати небезпек. Здоров'я може бути поповнено, якщо лисиця з'їсть фрукт під деревом, що стимулює гравця до додаткового дослідження карти.

Головна мета гравця в "Fox Adventure" - зібрати всі кристали на рівні. Це не тільки дозволяє гравцю прогресуватися в грі, але і впливає на кількість зірок, які він отримує після проходження рівня. Більше зібраних кристалів означає більше зірок, що в свою чергу слугує показником ефективності гравця.

Зірки в грі "Fox Adventure" виконують дві функції. По-перше, вони виступають як система оцінювання, дозволяючи гравцям порівнювати свої навички та досягнення. По-друге, зірки можуть слугувати стимулом для гравця повернутися та переграти рівень, намагаючись зібрати більше кристалів та отримати всі зірки.

Кожен рівень в "Fox Adventure" унікальний та має власні виклики. Незалежно від того, чи це лабіринт з перемикачами, який веде до кристалів, чи пастки та противники, які блокують шлях - кожен рівень вимагає від гравця уважності та ретельного планування.

В цілому, "Fox Adventure" - це гра, яка викликає інтерес та зацікавленість гравців завдяки своїй простоті, але складній грі. Це гра, яка не тільки розважає, але й стимулює гравців до розвитку своїх навичок та стратегічного мислення. Концепція гри базується на поєднанні простого, але привабливого геймплею з елементами стратегії та планування, що робить "Fox Adventure" цікавою та захоплюючою грою для широкого кола гравців.

1.5. Основні механіки гри та їхній вплив на геймплей

Механіки гри – це основні елементи, які я використовую для визначення того, як гравець взаємодіє з моєю грою та її елементами. У "Fox Adventure" основними механіками, які я розробив, є рух, збір предметів, взаємодія з об'єктами середовища, уникнення ворогів, а також відновлення здоров'я.

Рух є основою будь-якого платформера. У моїй грі гравець контролює лисицю, яка може переміщуватися вліво, вправо, стрибати, пригинатись, а також використовувати спеціальні платформи для пересування у вертикальному та горизонтальному напрямку. Ця механіка не тільки забезпечує рух по рівнях, але й вимагає від гравця стратегічного планування та реакції на динамічно змінюване середовище.

Збір предметів є другою важливою механікою в "Fox Adventure". Лисиця має збирати кристали, які розставлені по рівнях. Ці кристали не тільки підвищують загальний рахунок гравця, але й визначають кількість зірок, яку він отримує після завершення рівня.

Взаємодія з об'єктами середовища – це ще одна важлива механіка гри. Це включає активізацію перемикачів, які рухають платформи або відкривають нові шляхи, та збір фруктів для відновлення здоров'я. Ці механіки додають глибину геймплею, вимагаючи від гравця більше ніж просто бігати і стрибати.

Вороги та пастки поставляють виклики гравцю. Лисиця повинна уникати противників і пасток, що вимагає від гравця не тільки швидкості, але й тактичного мислення та планування. Дехто з противників може вимагати особливих стратегій, щоб уникнути або знищити, що додає варіативності та свіжості до геймплею.

Відновлення здоров'я є ключовим аспектом виживання в "Fox Adventure". Лисиця може відновити здоров'я, з'ївши фрукти, які розміщенні під деревами на рівнях. Ця механіка заохочує гравців до дослідження рівнів та уважного відношення до свого здоров'я, оскільки здоров'я не відновлюється автоматично.

Всі ці механіки разом створюють цікавий геймплей, який заохочує гравців до дослідження, стратегічного планування та використання різних тактик для досягнення найкращого результату на кожному рівні. Я ретельно продумав та реалізував кожну механіку так, щоб вона відповідала загальній тематиці та цілям гри.

1.6. Аналіз схожих ігор на ринку та їх особливостей

Аналіз ігор, подібних до "Fox Adventure", важливий для розуміння конкурентного середовища, а також для виявлення слабких та сильних сторін власного продукту. Наведу декілька прикладів ігор, які мають схожі елементи геймплею, а також особливості, які вони впроваджують.

"Super Mario Bros.": Ця класична гра в жанрі платформера використовує просту, але викликаючу механіку просування через рівні, уникнення ворогів та збір предметів. "Super Mario Bros." зосереджується на точних стрибках і швидкості. Хоча "Fox Adventure" та "Super Mario Bros." діляться жанром, "Fox Adventure" впроваджує додаткові механіки, такі як взаємодія з об'єктами середовища та відновлення здоров'я. Ігровий процес Ігровий процес "Super Mario Bros." зображений на рисунку 1.1.

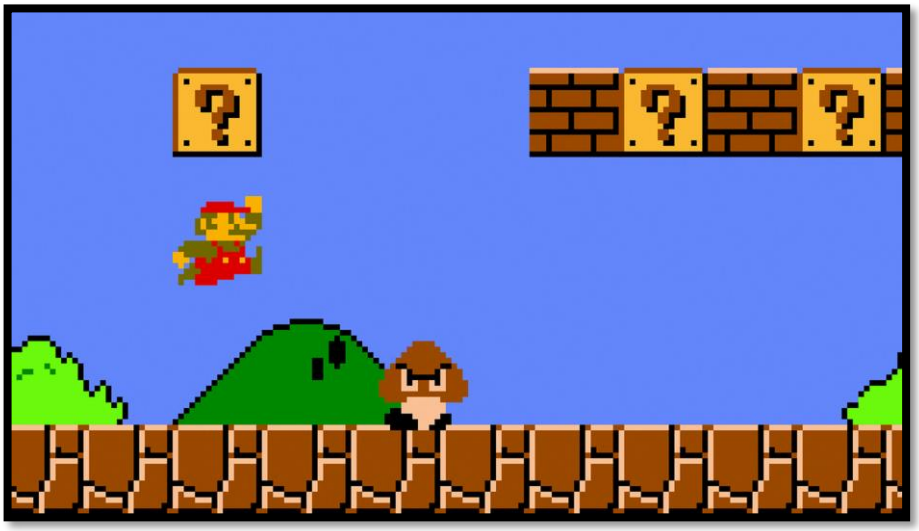

Рисунок 1.1 – Ігровий процес Super Mario Bros

"Celeste": Цей платформер включає в себе складні механіки, які вимагають високої точності та розуміння гравцем фізики гри. "Celeste" також пропонує викликаючу сюжетну лінію, що є важливим аспектом для привабливості гри. В "Fox Adventure" я намагався створити викликаючий геймплей, хоча і з менш складними механіками, та додати елементи сюжету, щоб збільшити зацікавленість

гравця. Ігровий процес Ігровий процес "Celeste" зображений на рисунку 1.2.

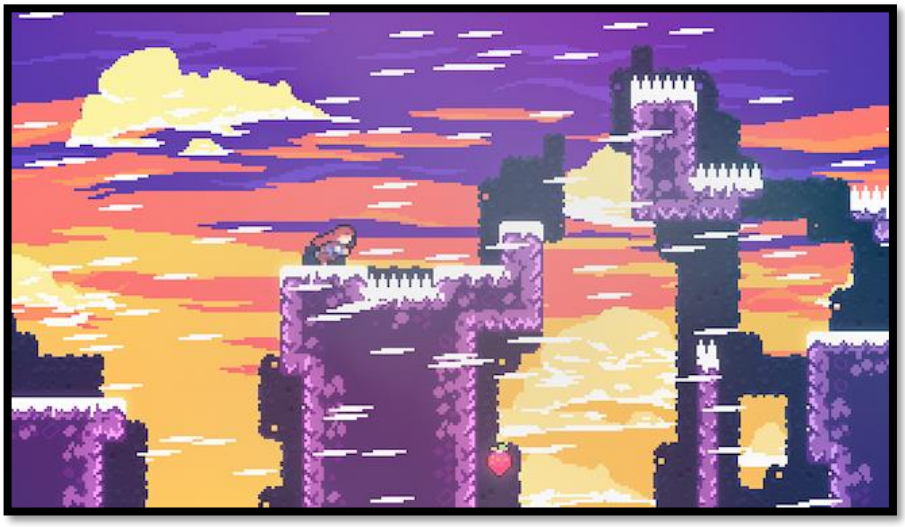

Рисунок 1.2 – Ігровий процес Celeste

"Hollow Knight": Ця гра комбінує елементи платформера і Metroidvania, пропонуючи гравцям великий відкритий світ для дослідження. "Hollow Knight" також включає бої з босами та систему вдосконалення персонажа. Хоча "Fox Adventure" не має такої глибокої системи прогресії, я впровадив елементи дослідження та взаємодії з об'єктами оточення, щоб додати більше глибини геймплею. Ігровий процес Ігровий процес "Hollow Knight" зображений на рисунку 1.3.

Рисунок 1.3 Ігровий процес Hollow Knight

"Ori and the Blind Forest": Цей графічно чарівний платформер використовує унікальні механіки переміщення, а також зосереджується на важливості дослідження та розгадування головоломок. "Ori and the Blind Forest" також має сильний сюжет, що пронизує гру. Хоча "Fox Adventure" не має такої сильної сюжетної лінії, я намагався впровадити елементи дослідження та головоломки, щоб збільшити зацікавленість гравця та різноманітність геймплею. Ігровий процес Ігровий процес "Ori and the Blind Forest" зображений на рисунку 1.4.

Рисунок 1.4 – Ігровий процес Ori and the Blind Forest

"Rayman Legends" – це інший платформер, який включає різноманітні механіки, включаючи біг, стрибки, бої та використання спеціальних здібностей для проходження рівнів. "Rayman Legends" також включає багаторівневу систему оцінки, аналогічну тій, яку я використовую в "Fox Adventure", для оцінки продуктивності гравця на рівнях. Ігровий процес Ігровий процес "Rayman Legends" зображений на рисунку 1.5.

Рисунок 1.5 – Ігровий процес Rayman Legends

Аналізуючи ці ігри, я зрозумів, які аспекти платформерів є найбільш успішними та відповідають моїй цілі в грі. Я також зміг визначити, які аспекти моєї гри відрізняються від існуючих ігор, і використати цю інформацію для вдосконалення моєї гри.

1.7. Визначення цільової аудиторії для "Fox Adventure"

Цільова аудиторія – це група людей, до яких спрямована гра. Визначення цільової аудиторії допомагає у розробці стратегій маркетингу, дизайну гри та її механік. Існує кілька факторів, які враховуються при визначенні цільової аудиторії, включаючи вік, стать, інтереси, навички геймінгу та багато іншого.

Гра розроблявся з метою привабити широку аудиторію геймерів. Основними факторами при визначенні цільової аудиторії були:

– вікова категорія: Гра розрахована на гравців віком від 7 років і старше. Вона має просту, але цікаву механіку, яка відповідає цій віковій категорії;

– навички геймінгу: Гра підходить як для новачків, так і для досвідчених гравців, завдяки простим для освоєння, але глибоким механікам гри;

– інтереси: "Fox Adventure" сподобається гравцям, які люблять пригоди, дослідження, головоломки та виклик;

– стать: Гра не має статевої спрямованості та розрахована на гравців будь-якої статі;

– розуміння цільової аудиторії важливе для успіху "Fox Adventure". Це допомогло мені прийняти ключові рішення щодо дизайну, геймплею та маркетингу гри, що, в свою чергу, вплине на її прийнятність та успіх на ринку ігор;

– платформа: Цільовою аудиторією "Fox Adventure" є користувачі ПК. Однак, у майбутньому планується розширити доступність гри на консолі та мобільні платформи, щоб привабити ще більше гравців;

– стиль гри: Цільова аудиторія "Fox Adventure" - це люди, які цінують ігри з високоякісною 2D графікою, інтригуючим сюжетом та елементами пригодницького геймплею.

Враховуючи ці фактори, я можу створити гру, яка відповідає потребам та інтересам цільової аудиторії. В майбутньому, я планую продовжувати оцінювати відгуки гравців, щоб вдосконалити "Fox Adventure" та забезпечити, щоб вона залишалась привабливою для цільової аудиторії.

1.8. Стратегія розробки та планування проєкту

Стратегія розробки та планування гри ретельно виважується і розробляється з метою забезпечити ефективне виконання робіт, раціональне розподілення ресурсів та вчасний випуск гри на ринок. Основні етапи стратегії розробки та планування проєкту включають:

– визначення цілей проєкту: На початковому етапі формуються основні цілі та завдання проєкту, такі як створення захоплюючого 2D платформера з унікальними механіками гри, забезпечення привабливого геймплею для цільової аудиторії та випуск гри на платформі ПК;

– розробка концепції гри: На цьому етапі створюється детальний опис

концепції гри, включаючи сюжет, головних персонажів, особливості геймплею та візуальний стиль;

– вивчення доступних технологій: Після визначення концепції гри проводиться дослідження щодо доступних ігрових движків та інструментів, щоб вибрати найбільш підходящий для реалізації для моєї гри. В результаті обирається движок Unity;

– розробка детального плану проєкту: На основі досліджень та визначення цілей проєкту створюється детальний план розробки, який включає розподіл завдань, терміни виконання та необхідні ресурси;

– реалізація проєкту: Згідно з планом, проєкт розподіляється на етапи, такі як створення прототипу гри, розробка ассетів, програмування механік, інтеграція аудіо та візуальних ефектів, тестування та налагодження;

– тестування: Після завершення розробки гри проводиться тестування для виявлення та усунення помилок, оптимізації геймплею та забезпечення якості продукту.

Висновок до розділу

У першому розділі було проведено аналіз сучасного стану ігрової індустрії та її тенденцій. Це дослідження дозволило зрозуміти, що ігрова індустрія продовжує активно розвиватися і змінюватися, адаптуючись до потреб гравців. Визначено, що жанр 2D платформерів займає важливе місце в ігровій індустрії, що обумовлено його доступністю для гравців різного віку та досвіду гри.

Досліджено різні ігрові движки, серед яких Unity, Unreal Engine, Godot, GameMaker Studio. Кожен з движків був аналізований з точки зору його функціональності, складності вивчення та можливостей для розробки 2D платформерів. На основі проведеного аналізу було вирішено вибрати движок Unity для розробки гри.

Обґрунтування вибору движка Unity базується на його широких можливостях для розробки 2D ігор, гнучкості, великій кількості навчальних

матеріалів, а також на його популярності в ігровій індустрії. Unity дозволяє створювати гнучкі і динамічні 2D ігри, що відповідають високим вимогам сучасних гравців.

Розробка концепції та ідеї гри включала вибір назви гри, стилю гри, основні механіки які будуть реалізовані та мету гри. Важливим аспектом стало створення цікавого і захоплюючого геймплею, який би приваблював гравців та підтримував їхній інтерес до гри.

Дослідження основних механік гри та їх впливу на геймплей допомогло розробити ефективну систему взаємодії гравця з грою. Було визначено ключові елементи, які впливають на відчуття гравця від гри, та розроблено механіки, які максимально покращують це відчуття.

У цьому розділі було здійснено аналіз схожих ігор на ринку, що допомогло визначити ключові тенденції та особливості сучасних 2D платформерів. Цей аналіз дав можливість визначити, що вже є на ринку, які ідеї вже були реалізовані, і що нового можна пропонувати гравцям.

Дослідження цільової аудиторії "Fox Adventure" дозволило краще зрозуміти потреби та вимоги гравців, що зрештою вплине на успіх проєкту. Визначено, що гра "Fox Adventure" спрямована на широку аудиторію гравців, які цінують класичні 2D платформери з інноваційними механіками та цікавим сценарієм.

Стратегія розробки та планування проєкту було обговорено на останньому етапі цього розділу. Було визначено ключові етапи розробки гри.

У цілому, цей розділ формує тверду основу для подальшої реалізації проєкту детально планує всі етапи його розробки, визначає основні ризики та способи їх уникнення, а також встановлює цілі і очікування від фінального продукту.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Вхідні дані

Вхідні дані в грі в основному пов'язані з керуванням персонажем та вибором опцій в грі:

– керування персонажем: Гравець використовує клавіші на клавіатурі для керування персонажем у грі. Він може рухати персонажа вліво, вправо, стрибати, а також використовувати спеціальні здібності персонажа;

– вибір опцій: Гравець може вибирати різні опції у грі, такі як режим екрану (повний або віконний), розширення екрану та налаштування звуку;

– взаємодія з об'єктами гри: Гравець може взаємодіяти з різними об'єктами у грі, такими як збирання кристалів, відкриття дверей, перемикання важелів тощо.

2.2. Вихідні дані

Вихідні дані в грі "Fox Adventure" це інформація, яку гра надає гравцеві в ході гри та після її завершення:

– результати гри: Після завершення рівня гра показує гравцю його результати, такі як кількість зібраних кристалів, час, за який був пройдений рівень, кількість зірок, отриманих за проходження рівня тощо;

– повідомлення про досягнення: Якщо гравець виконав певну дію або досяг певної мети у грі, гра повідомляє йому про це, показуючи спеціальне повідомлення про досягнення;

– стан гри: Гра постійно надає гравцеві інформацію про стан гри, таку як кількість здоров'я персонажа, кількість зібраних кристалів, поточний час гри тощо;

– відгуки на дії гравця: Коли гравець взаємодіє з об'єктами гри, відбувається відповідна реакція, наприклад, зміна зовнішнього вигляду об'єкта, звуковий сигнал або анімація.

2.3. Вибір та імпорт графічних ассетів та звуків

Для створення гри "Fox Adventure" було обрано кілька ассетів, які надають основну графіку та звук для гри.

"Sunny Land" [6]: Цей ассет було обрано як основний для гри через його багатий набір графічних елементів і звуків, які добре підходять для 2D платформера. Він включає в себе спрайти для головного персонажа, ворогів, фонів, рівнів та предметів, а також музику для гри. Зразок графічних ассетів з пакету "Sunny Land" зображений на рисунку 2.1.

Рисунок 2.1 – Зразок графічних ассетів з пакету "Sunny Land"

"371 Simple Buttons Pack" [7]: Цей пакет був використаний для створення кнопок у користувацькому інтерфейсі гри. Він надає велику кількість готових до використання кнопок з різними формами та кольорами, що дозволяє легко налаштувати інтерфейс під потреби гри. Зразок графічних ассетів з пакету "371 Simple Buttons Pack" зображений на рисунку 2.2.

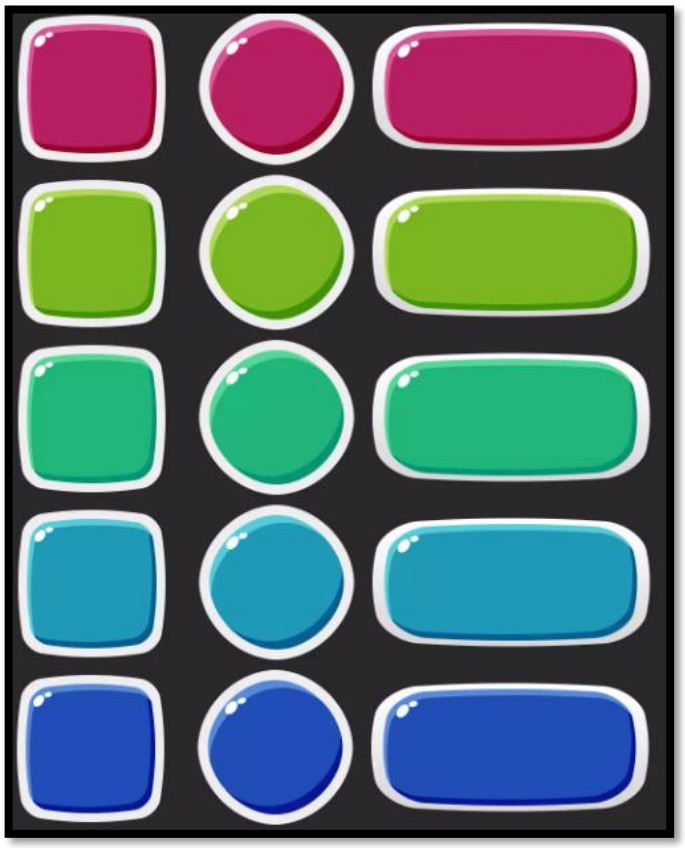

Рисунок 2.2 – Зразок графічних ассетів з пакету "371 Simple Buttons Pack"

Окрім цих ассетів, було також використано кілька спрайтів, скачаних з інших джерел за межами Asset Store. Ці спрайти допомогли додати унікальність та індивідуальність до дизайну гри.

2.4. Імпорт та налаштування ассетів

Після вибору відповідних ассетів, вони були імпортовані в проєкт Unity. Під час процесу імпорту, ассети автоматично конвертуються в формати, які оптимізовані для використання в Unity.

Після імпорту, ассети були налаштовані для використання у грі. Для спрайтів було налаштовано їхні параметри, такі як розмір, орієнтацію, колір та інше. Для звуків було налаштовано їхні параметри, такі як гучність, глибину та можливість зациклити звуки.

Завдяки інструментам Unity, процес імпорту та налаштування ассетів став

простим та зручним, дозволяючи швидко і легко інтегрувати нові елементи в гру.

2.5. Налаштування інтерфейсу користувача

Одним з ключових елементів будь-якої відеогри є її користувацький інтерфейс (UI). UI гри має бути інтуїтивно зрозумілим і легким у використанні, щоб гравці могли легко взаємодіяти з грою і отримувати задоволення від процесу гри.

В грі використовуються наступні екрани інтерфейсу, кожен з яких має своє визначене призначення, описані у таблиці 2.1.

Таблиця 2.1 – Екрани інтерфейсу

Екран інтерфейсу	Призначення
Головне меню	Це вхідна точка в гру, перший екран, який гравець бачить після запуску. З нього можна перейти до гри, налаштувань, вибору рівнів та інших розділів.
Меню налаштувань	Дозволяє користувачу змінювати розширення екрану, режим повного екрану та регулювати гучності звуку.
Меню вибору рівнів	Надає можливість гравцеві вибирати рівень для гри з доступних варіантів.
Меню паузи	Доступне в будь-який час гри, дозволяє гравцю призупинити гру,

	повернутися до головного меню, змінити налаштування гри або перейти до меню налаштувань.

Ця структура інтерфейсу забезпечує зручний доступ до основних функцій гри, забезпечуючи інтуїтивно зрозумілі механіки навігації для гравців.

Кнопки для навігації по меню було створено з використанням графічних елементів з ассету "371 Simple Buttons Pack". Всі ці елементи, разом з графічними ассетами з "Sunny Land", створюють приємний і цікавий вигляд інтерфейсу користувача гри.

Висновок до розділу

У другому розділі було розглянуто вхідні та вихідні дані гри "Fox Adventure", а також процес вибору та імпорту графічних ассетів та звуків для гри. Було зазначено, що вхідні дані в грі пов'язані з керуванням персонажем, вибором опцій та взаємодією з об'єктами гри. Вихідні дані включають результати гри, повідомлення про досягнення, стан гри та відгуки на дії гравця.

Далі було описано процес вибору та імпорту графічних ассетів та звуків для гри "Fox Adventure". Основним графічним ассетом був обраний пакет "Sunny Land", який мав багатий набір графічних елементів та звуків, включаючи спрайти для персонажів, фонів, рівнів та предметів, а також музику. Для створення інтерфейсу був використаний пакет "371 Simple Buttons Pack", який надавав готові кнопки з різними формами та кольорами.

Крім того, було зазначено процес імпорту та налаштування ассетів в Unity, включаючи налаштування параметрів спрайтів та звуків для використання в грі.

Нарешті, було розглянуто налаштування інтерфейсу користувача гри "Fox Adventure". Були описані різні екрани інтерфейсу, включаючи головне меню, меню

налаштувань, меню вибору рівнів та меню паузи. Зазначено, що графічні елементи з пакету "371 Simple Buttons Pack" були використані для створення кнопок в інтерфейсі.

Було показано, які дані використовуються в грі "Fox Adventure" і як вони імпортуються, налаштовуються та використовуються для створення забезпечення гри. Всі ці елементи допомагають створити зручний та привабливий геймплей для гравців, що сприяє позитивному досвіду гри.

Детальний аналіз вхідних та вихідних даних, вибір ассетів, налаштування графіки і звуку, а також створення інтуїтивного і привабливого інтерфейсу користувача в грі "Fox Adventure" виявилися ключовими елементами в розробці відеогри.

Процеси, які були виконані в цьому розділі, покривають значну частину розробки відеогри, починаючи від аналізу вхідних та вихідних даних для забезпечення точної взаємодії з гравцем, закінчуючи створенням красивого і привабливого інтерфейсу користувача. Вони також включають вибір та імпорт ассетів, що дає грі унікальний вигляд і відчуття.

Важливо відмітити, що кожен з цих елементів був обраний та налаштований з врахуванням потреб і очікувань гравців. Детальний аналіз вхідних та вихідних даних дозволяє гарантувати, що гра відповідає на дії гравця точно і вчасно, а красиві графіка і звук створюють цікавий геймплей.

Також було виявлено, що важливою частиною процесу є налаштування інтерфейсу користувача. Зручний і інтуїтивно зрозумілий інтерфейс користувача є одним з ключових факторів, які впливають на загальну оцінку гри гравцями.

3 РОЗРОБКА КОМПОНЕНТІВ ГРИ

3.1 Засоби розробки

При розробці "Fox Adventure" було використано два основних інструменти – Unity [8] редактор та Rider [9] від JetBrains.

Unity є відомим движком для розробки ігор, який забезпечує розробникам потужний набір інструментів для створення вражаючого ігрового досвіду. Цей редактор було обрано для розробки "Fox Adventure" через його зручний інтерфейс та потужні можливості для 2D ігор. Дизайнери і програмісти використовували Unity для створення ігрових ассетів, формування ігрового середовища, налаштування ігрової механіки, а також для написання та впровадження скриптів на мові C# [10]. Робоче середовище Unity зображене на рисунку 3.1.

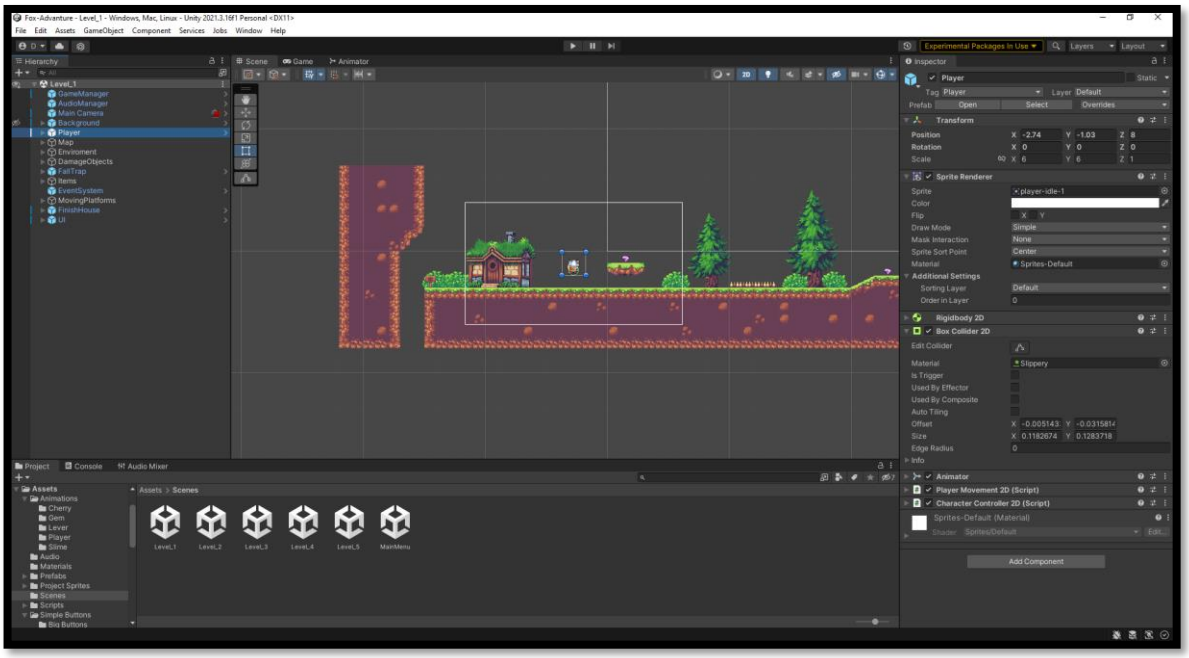

Рисунок 3.1 – Робоче середовище Unity

Rider від JetBrains був обраний як основний інструмент для написання та редагування коду. Цей IDE надає широкі можливості для розробки на C#, включаючи засоби рефакторингу, налагодження, автоматичного вирівнювання коду, а також управління версіями. Rider ідеально підходить для роботи з

проєктами на Unity, оскільки він пропонує плавну інтеграцію з цим движком, що дозволяє розробникам вносити зміни в код безпосередньо в процесі роботи над проєктом. Інтерфейс Rider зображений на рисунку 3.2.

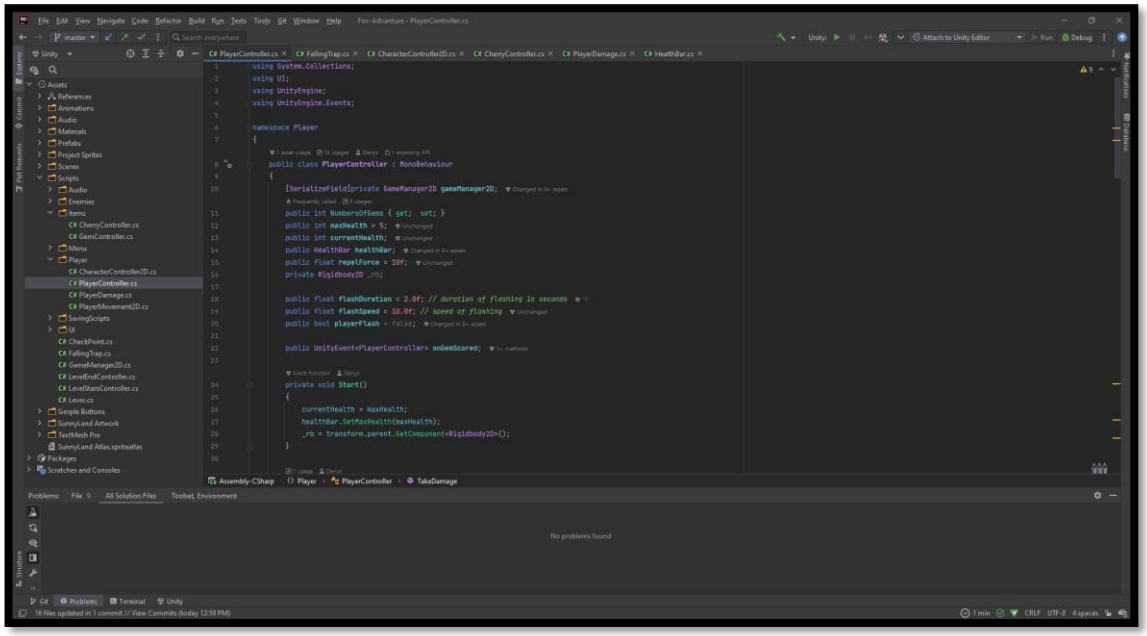

Рисунок 3.2 – Інтерфейс Rider під час роботи

Використання Unity та Rider спільно дало можливість ефективно розробляти та налаштовувати "Fox Adventure.

3.2 Створення карт рівнів за допомогою Tilemap

Під час створення 2D-ігор одним із ключових аспектів є розробка дизайну рівнів. В Unity для цього існує потужний інструмент під назвою Tilemap. Ця система дозволяє розробникам легко та ефективно створювати багаторівневі та деталізовані 2D-сцени за допомогою тайлів (плиток) – маленьких, повторюваних фрагментів графіки.

У "Fox Adventure" система Tilemap була використана для розробки деталізованих, живих і цікавих для дослідження рівнів. Кожен рівень було розроблено з використанням великої кількості унікальних тайлів, що разом

формують вражаючий ігровий ландшафт. Вся ця процедура виконувалася безпосередньо в редакторі Unity за допомогою інтерфейсу Tilemap. Вигляд Tilemap зображений на рисунку 3.3.

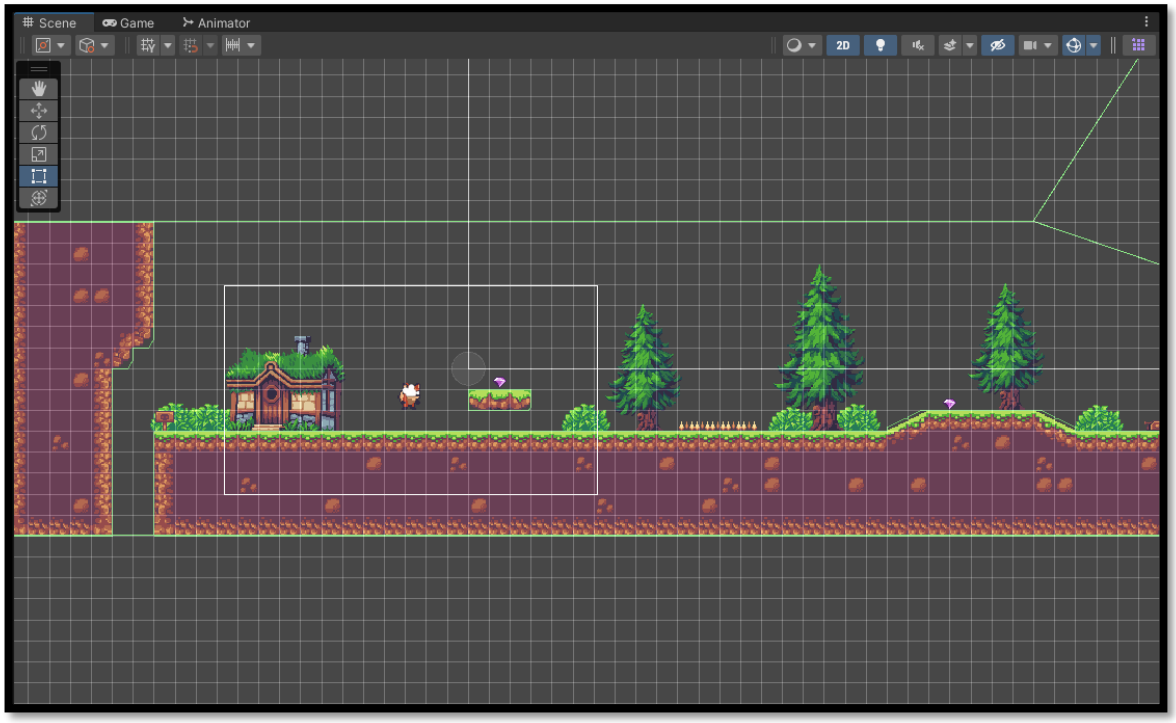

Рисунок 3.3 – Процес створення карт рівнів за допомогою Tilemap

Tilemap в Unity має багато переваг. Зокрема, він дозволяє легко модифікувати та налаштовувати дизайн рівнів, змінювати фрагменти сцени та експериментувати з різними стилями та концепціями дизайну. Крім того, система Tilemap дозволяє організувати тайли в шари, що сприяє створенню більш складних та інтерактивних сцен.

Для більш ефективного використання системи Tilemap, Unity надає інструмент Tile Palette. Це панель, що зберігає та відображає всі доступні тайли для розробки рівнів. За допомогою Tile Palette, розробники можуть швидко та легко перетягувати тайли на карту рівня, створюючи дизайн в реальному часі. Вигляд Tile Palette зображений на рисунку 3.4.

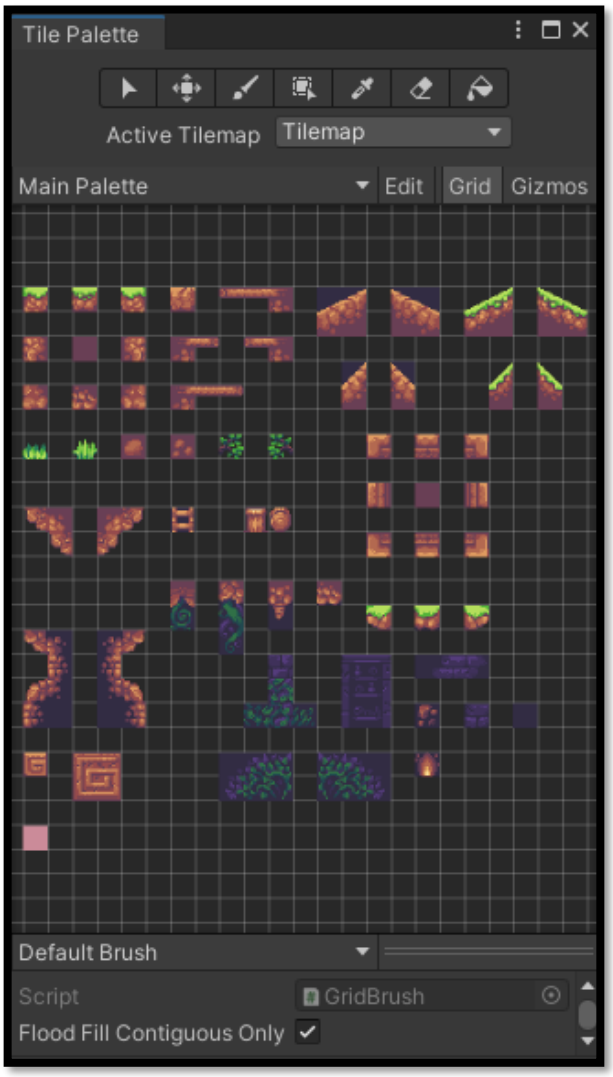

Рисунок 3.4 – Tile Palette в процесі роботи

У сукупності, використання Tilemap та Tile Palette в Unity дозволяє створювати багатовимірні, візуально привабливі та гнучкі для налаштувань ігрові середовища. Це значно полегшує процес розробки 2D-ігор.

3.3 Створення ігрового персонажа

3.3.1 Створення об’єкта персонажа

Для головного героя гри "Fox Adventure" було обрано персонажа у вигляді лисиці. Прийнято рішення використовувати готовий графічний ассет, що забезпечив ефективність розробки та дозволив зосередитися на інших важливих

елементах гри.

Цей графічний ассет містить високоякісні 2D спрайти і анімації лисиці, які відповідають потребам нашого платформера. Було вирішено інтегрувати цей ассет в ігровий движок Unity, що дозволило нам легко налаштувати рухи та дії персонажа відповідно до вимог гри. На рисунку 3.5 зображено, як виглядає ігровий персонаж.

Рисунок 3.5 – Ігровий персонаж

Слід зазначити, що використання готових ассетів є поширеною практикою в ігровій індустрії, особливо серед інді-розробників та невеликих студій. Вони дозволяють зекономити час та ресурси, при цьому не втрачаючи в якості.

В результаті, з використанням готового ассета, ми успішно створили ігрового персонажа – лисицю. Цей персонаж не лише виконує функції управління в грі, але й стає ключовим елементом ігрового оповідання.

Ігровий персонаж має ряд ключових компонентів, які об'єднуються для створення бажаної поведінки та взаємодії з середовищем гри. Ці компоненти включають: Sprite Renderer, Rigidbody2D, Box Collider 2D, Animator, PlayerMovement2D, PlayerController та CharacterController2D.

Всі ці компоненти разом відповідають за візуалізацію персонажа, його рухи, взаємодію з оточуючим середовищем та управління цими рухами. Більш детальний опис цих компонентів представлений в Таблиці 3.1.

Таблиця 3.1 – Компоненти ігрового персонажа

Компонент	Опис
Sprite Renderer	Відповідає за відображення спрайта персонажа в грі. Управляє такими властивостями, як колір та прозорість спрайта, а також дозволяє змінювати спрайт персонажа в режимі реального часу.
Rigidbody2D	Додає фізику 2D до об'єкта. Дозволяє об'єкту реагувати на сили та зіткнення в грі. Відповідає за управління рухом об'єкта.
Box Collider 2D	Створює область зіткнення навколо об'єкта, що дозволяє йому взаємодіяти з іншими об'єктами в грі через систему фізики Unity.
Animator	Управляє анімаціями персонажа. Використовує дані з Animator Controller для управління поточним станом анімації та переходами між анімаціями.
PlayerMovement2D	Відповідає за управління рухом персонажа. Використовує вхідні дані від користувача (наприклад, клавіші управління) для визначення того, як персонаж має рухатися та взаємодіяти з оточенням.
CharacterController2D	Управляє специфічною для персонажа поведінкою, такою як стрибки та повзання. Використовує інформацію з Rigidbody2D та PlayerMovement2D для виконання цих дій в режимі реального часу.
PlayerController	Управляє ключовими характеристиками героя, такими як здоров'я, кількість зібраних дорогоцінних каменів, взаємодія з оточенням, та реагування на ігрові події. Він також управляє такими діями, як отримання шкоди, відновлення здоров'я, взаємодія з

	збиральними об'єктами та відображення стану здоров'я героя в інтерфейсі користувача.

3.3.2 Створення анімації персонажа

Кожен персонаж повинен бути в русі, тому далі буде створена анімація для нього. Анімація персонажа є одним з ключових елементів, що надає ігровому досвіду життєвості та динамічності. У грі використовується набір анімацій, що включає рухи ходьби, стрибку, повзання та бездіяльності, що здійснює персонаж під час гри.

Щоб створити ці анімації в Unity, спочатку потрібно імпортувати спрайти персонажа в движок. Після імпорту, спрайти можна використати для створення анімаційних кліпів за допомогою вбудованого інструменту анімації Unity – Animator.

Animator – це могутній інструмент для створення та управління анімаціями. Він дозволяє налаштовувати анімаційні переходи, змінювати швидкість анімації, а також створювати складні анімаційні сценарії з використанням логіки станів.

Animator складається з двох основних компонентів: контролера анімації (Animator Controller) і вікна Animator.

Animator Controller використовується для організації та управління всіма анімаціями в сцені. Він включає в себе набір анімаційних кліпів і параметрів, що визначають, як і коли будуть використовуватись окремі анімації. Вікно створення анімації зображене на рисунку 3.6.

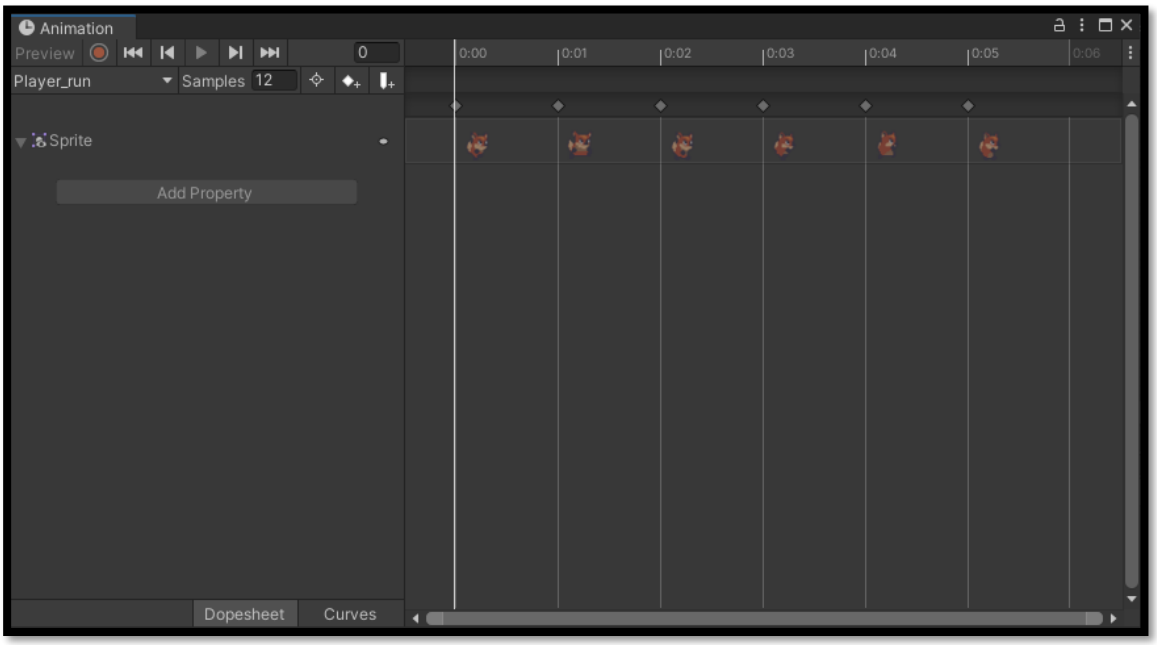

Рисунок 3.6 – Вікно створення анімації бігу для головного персонажу

Вікно Animator, з іншого боку, є середовищем, в якому можна візуально організувати і управляти анімаціями, станами та переходами між ними. Воно включає в себе ряд вкладок, включаючи вкладки Layers (шари), Parameters (параметри) і Controller (контролер).

У вкладці Layers можна створити різні шари анімації, що дозволяє контролювати декілька анімацій одночасно.

Вкладка Parameters дозволяє створювати та управлять різними параметрами, які впливають на поведінку анімацій. Ці параметри можуть бути числовими значеннями, булевими змінними або тригерами, і вони можуть бути використані для контролю переходів між станами.

Вкладка Parameters в Unity Animator дозволяє вводити і контролювати ці параметри в реальному часі, що дозволяє адаптувати анімацію до різних сценаріїв ігрового процесу.

Далі наведена таблиця 3.2 з описами параметрів, які були використані у проєкті.

Таблиця 3.2 – Параметри Animator Controller для управління анімацією персонажа

Параметр	Тип	Опис
Speed	float	Використовується для контролю швидкості персонажа. Дозволяє перехід зі стану бездіяльності до стану бігу.
IsJumping	bool	Застосовується для запуску анімації стрибка персонажа.
IsCrouching	bool	Застосовується для запуску анімації присідання персонажа.

В результаті, за допомогою інструменту Animator в Unity, було створено динамічну анімацію для нашого ігрового персонажа. Також було створено діаграму станів і переходів для анімацій гравця, дана діаграма зображена у графічних матеріалах (ІС91.030БАК.004 Д1). Ця анімація не лише прикрашає візуальний стиль гри, але й додає відчуття реалізму та життєвості, що сприяє заглибленню гравця в ігровий процес.

3.3.3 Налаштування камери гравця

Управління камерою - важлива частина створення ефективної гри, оскільки воно впливає на те, як гравці сприймають і взаємодіють з ігровим світом. В даному проєкті для управління камерою було використано Cinemachine – потужний плагін Unity, що дозволяє розробникам легко створювати складні системи камери без необхідності писати великого обсягу коду. Cinemachine пропонує різноманітні можливості, зокрема слідкування за об'єктами, плавні переходи між різними камерами та багато іншого.

З іншого боку, для контролю за камерою було використано Confiner. Confiner – це компонент Cinemachine, який обмежує рух камери в межах певної області. Це

може бути особливо корисно в 2D-іграх для уникнення показу поза межами ігрового світу. Основні компоненти камери представлені на рисунку 3.7.

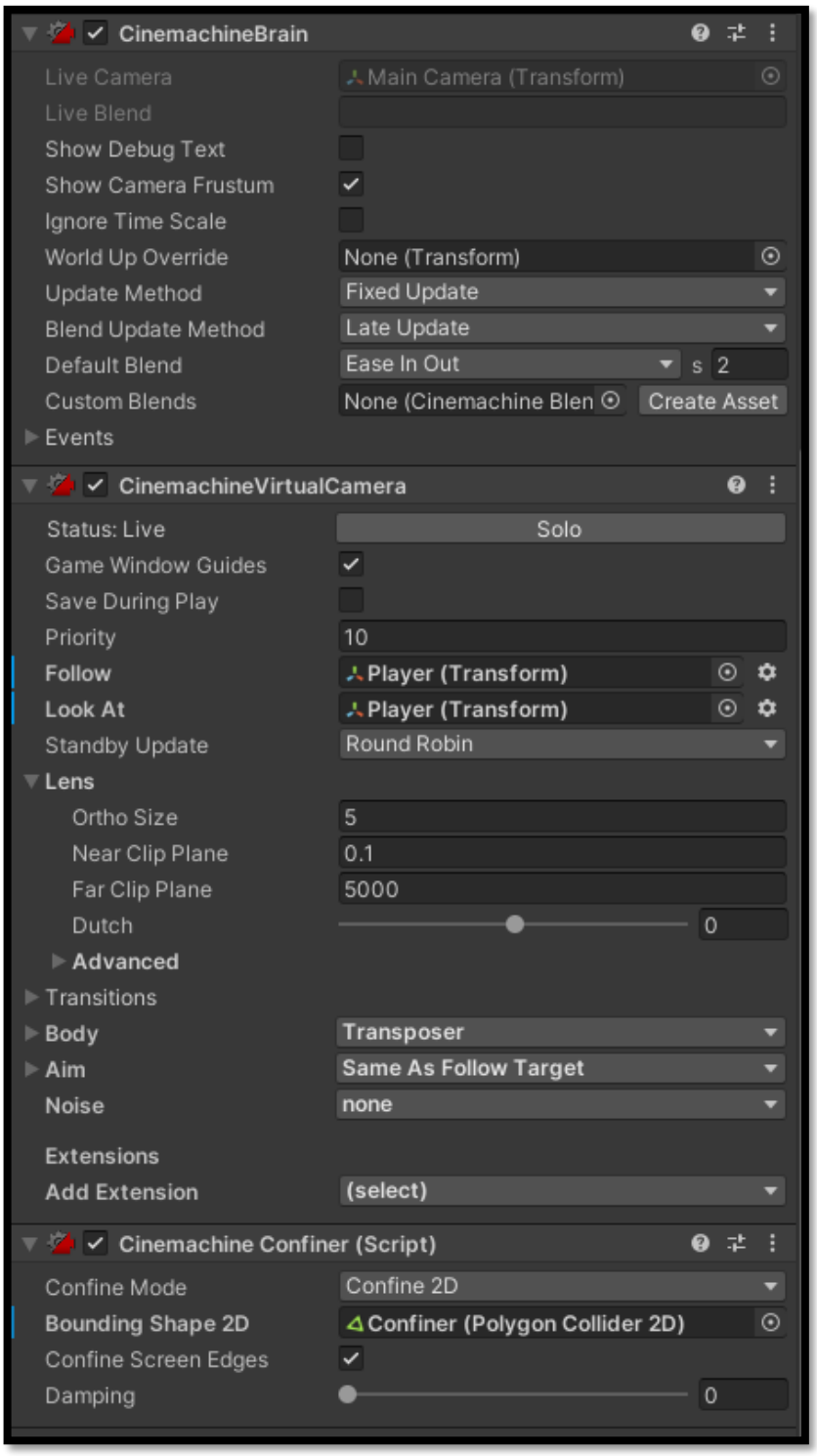

Рисунок 3.7 – Компоненти камери гравця

Компонент Cinemachine Virtual Camera представляє віртуальну камеру, яка

може слідкувати за певним об'єктом та контролювати його розташування на екрані. Він також може плавно перемикатися між різними точками зору за допомогою різних "переходів".

Специфічні налаштування Cinemachine Virtual Camera та Cinemachine Confiner представлені в таблиці 3.2.

Таблиця 3.2 – Налаштування камери

Компонент	Налаштування	Опис
Cinemachine Virtual Camera	Follow	Об'єкт, за яким слідкує камера.
	Look At	Об'єкт, на який спрямована камера.
Cinemachine Confiner	Bounding Shape 2D	Область, в межах якої може рухатися камера.
Cinemachine Brain	Blend Update Method	Визначає, як будуть відбуватися переходи між віртуальними камерами. FixedUpdate - переходи відбуваються за фіксовані кроки часу, що корисно для фізично орієнтованих сцен. Update - переходи відбуваються при кожному кадрі, що забезпечує плавний вигляд для більшості сцен. LateUpdate - переходи відбуваються після того, як всі інші об'єкти сцени були оновлені; це може бути корисно для ситуацій, коли камера має відслідковувати дії, які

		відбуваються в ході оновлення інших об'єктів.
	Default Blend	Визначає, як буде відбуватися перехід між віртуальними камерами за замовчуванням. Можна вибрати тип згладжування (наприклад, лінійний, ease in, ease out) та тривалість зміни.
	Custom Blends	Дозволяє визначити власні зміни для певних пар віртуальних камер.

Разом ці налаштування дозволяють створити гнучку систему камери, яка адаптується до руху гравця та забезпечує оптимальне відображення ігрового світу.

3.4 Опис інтерактивних об'єктів і елементів середовища

3.4.1 Вишня

Вишня в грі виступає як типовий "health pack", дуже знайомий геймерам концепт, що використовується в багатьох іграх. Ідея полягає в тому, що певні предмети, зібрані персонажем під час його подорожі, можуть відновити його здоров'я, часто візуалізовано в грі.

В грі вишня є саме таким предмет. При зборі вишні, здоров'я персонажа поповнюється на одну одиницю, що може бути вирішальним фактором в виживанні в умовах ворожого середовища. Це також стимулює гравця проводити більше часу, досліджуючи рівень. У випадку коли гравець має повне здоров’я, то вишню зібрати не вийде.

Важливою складовою елемента "вишня" є його візуальне представлення, зображено на рисунку 3.8. Зображення вишні є яскравим, привабливим і легко

розпізнаваним, що сприяє швидкому визначенню його як позитивного для персонажа предмета.

Рисунок 3.8 – Інтерактивний об'єкт - вишня, який поповнює здоров'я персонажа

Також варто зазначити, що вишня в грі має анімацію бездіяльності. Коли вишня не взаємодіє з персонажем, вона постійно рухається та обертається, створюючи візуальний динамізм і привертаючи увагу гравця. Це допомагає гравцю швидко визначити її розташування у середовищі і спонукає до взаємодії.

Цей об’єкт у грі використовує низку компонентів для взаємодії з гравцем. Нижче наведено таблицю 3.3, яка описує ці компоненти.

Таблиця 3.3 – Компоненти інтерактивний об'єкта - вишня

Компонент	Опис
Transform	Визначає положення, обертання та масштаб об'єкта вишні в сцені.
Sprite Renderer	Відповідає за відображення спрайта вишні, що дозволяє гравцю бачити вишню на екрані.
Circle Collider 2D	Додає фізичну колізію до об'єкта вишні, дозволяючи йому взаємодіяти з іншими об'єктами у грі, які мають колайдери. Цей компонент має галочку напроти параметра IsTrigger, що означає, що колайдер

	використовується для того, щоб спрацьовувати на певні події (триггери), а не для фізичної колізії з іншими об'єктами.
Animator	Керує анімаціями вишні, включаючи анімацію бездіяльності.
CherryController	Скрипт, який визначає поведінку вишні під час взаємодії з гравцем. Коли гравець торкається вишні, цей контролер збільшує здоров'я гравця та видаляє вишню з сцени.

CherryController має простір імен Items, що допомагає організувати код та вказує, що цей контролер використовується для об'єктів типу "предмети".

3.4.2 Дорогоцінний Камінь

Дорогоцінний камінь в грі представляє собою об'єкт колекціонування, зображений на рисунку 3.9. Зібрані камені впливають на рейтинг зірок, який гравець отримує після проходження рівня. Кількість зібраних каменів визначає кількість отриманих зірок, що є мотивацією для гравця досліджувати рівень глибше і детальніше.

Важливо підкреслити, що дорогоцінні камені, як і вишні, мають привабливий вигляд, що робить їх легко видимими і відрізняє їх від інших об'єктів середовища.

Рисунок 3.9 – Дорогоцінний камінь - об'єкт, що збирається персонажем для отримання зірок після проходження рівня

Дорогоцінний камінь також має анімацію бездіяльності, він обертається та трохи рухається вверх і вниз [11]. Ця анімація не тільки збільшує візуальну привабливість об’єкта, але і позначає його важливість для гравця, підсилюючи мотивацію зібрати його.

Дорогоцінний камінь у грі використовує декілька компонентів, подібних до вишні, але має і свої відмінності. Замість CherryController, дорогоцінний камінь використовує GemController, який контролює його поведінку.

Компонент GemController призначений для контролю поведінки дорогоцінного каменя в грі. Цей компонент має такі параметри, які можна налаштовувати у редакторі Unity:

– Speed: цей параметр визначає швидкість, з якою дорогоцінний камінь піднімається та опускається. Чим вище ця величина, тим швидше камінь буде рухатися.

У GemController також присутній метод OnTriggerEnter2D, який активується при зіткненні з гравцем. Коли гравець зіштовхується з дорогоцінним каменем, GemController викликає метод GemCollected() від PlayerController. Це додає зібраний камінь до загального рахунку гравця і вимикає камінь в сцені.

Загалом, GemController – це важливий компонент, який відповідає за динаміку руху дорогоцінного каменя і його взаємодію з гравцем.

3.4.3 Важіль

Важіль у грі виступає як механізм взаємодії, який при включенні переміщує певну платформу на визначену відстань. Коли гравець взаємодіє з важелем повторно, платформа повертається на свою первісну позицію.

Ця функція дозволяє створювати різноманітні головоломки на рівнях, де гравцю потрібно визначити правильний порядок взаємодії з важелями для досягнення мети. Крім того, ця взаємодія з важелями може використовуватися для відкриття секретних місць або зон на рівні, додаючи ще одну глибину до геймплею.

Важелі розташовані по всьому рівню, що забезпечує необхідний рівень виклику для гравця, а також заохочує його до дослідження та взаємодії з середовищем.

При вході гравця в зону дії важеля, тобто в область, де можлива взаємодія з ним, над важелем з'являється іконка відповідної кнопки. Гравцю необхідно натиснути цю кнопку для активації важеля. Активація важеля здійснює контрольований рух платформи, до якої він прив'язаний. Після активації, платформа змінює своє положення, рухаючись в заданому напрямку. Взаємодія з важелем забезпечує візуальний зворотний зв'язок для гравця, що дозволяє йому краще розуміти, як його дії впливають на середовище гри. На рисунках 3.10 та 3.11 зображено як виглядає важіль та його вплив після взаємодії.

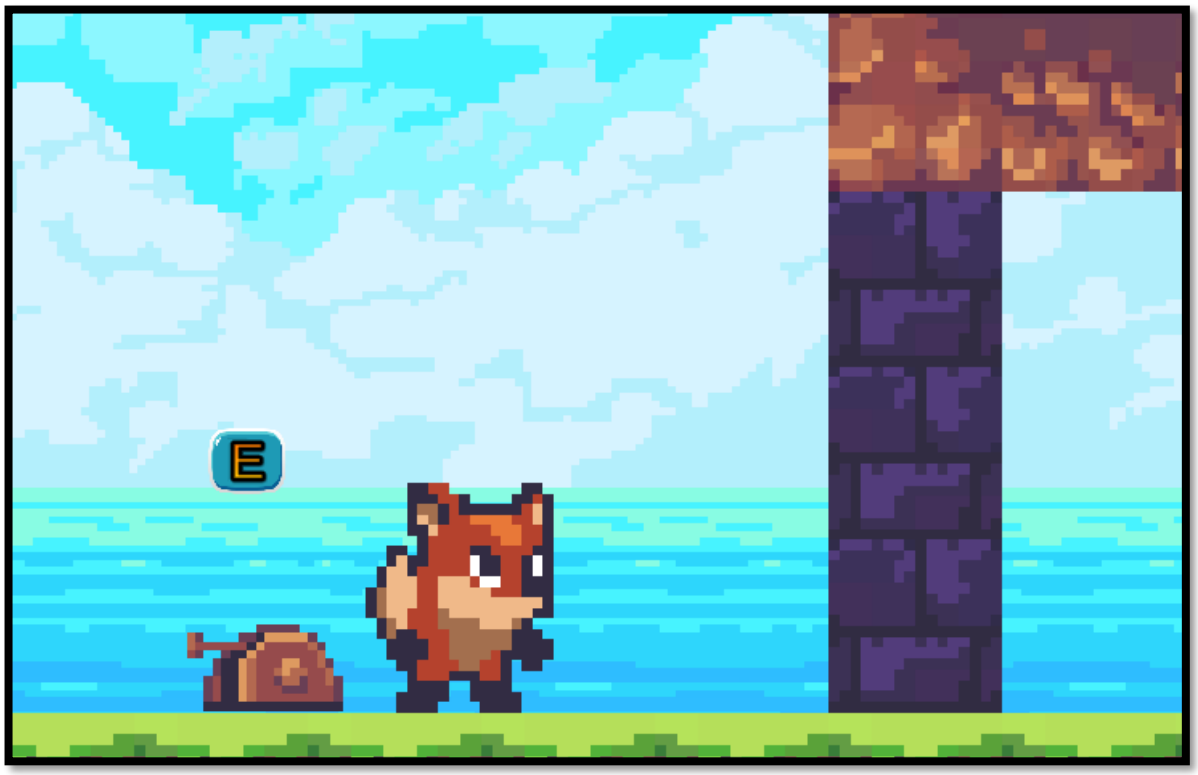

Рисунок 3.10 – Важіль до активації

Рисунок 3.11 – Важіль після активації та його вплив на середовище

Важіль у грі має наступні компоненти: Transform, Circle Collider 2D, Animator і Lever. Основна логіка поведінки важелю реалізована у скрипті Lever. Нижче представлена таблиця, яка описує основні параметри компонента Lever, його інтерфейс редактора Unity зображений на рисунку 3.12.

Таблиця 3.4 – Опис параметрів компонента Lever

Параметр	Опис
objectToMove	Об'єкт, який буде рухатися при активуванні важеля.
directions	Напрямок руху об'єкта при активуванні важеля. Може бути встановлений на Верх, Низ, Ліво або Право.
animator	Аніматор, який відповідає за анімацію важеля.
canvas	Canvas, на якому зображена іконка кнопки, яку гравець повинен натиснути для активування важеля.

moveDistance	Відстань, на яку буде зміщений об'єкт при активуванні важеля.
moveSpeed	Швидкість зміщення об'єкта при активуванні важеля.

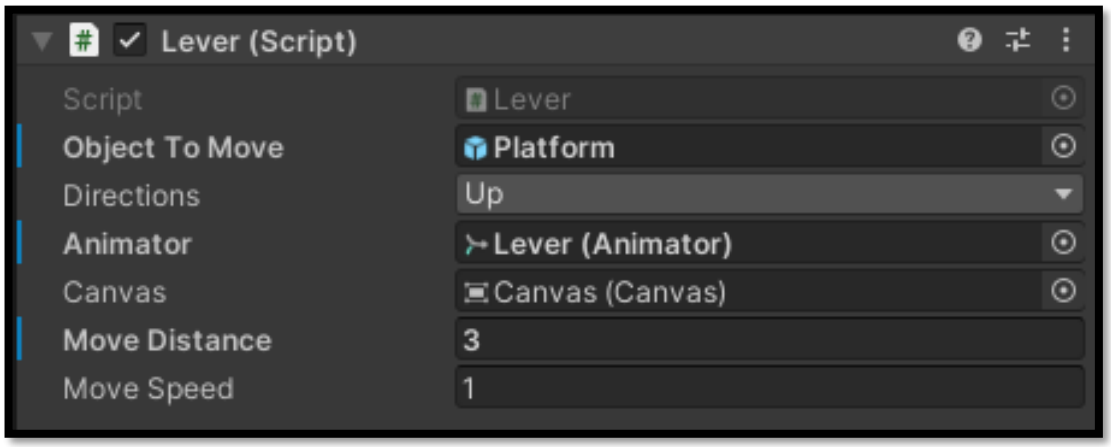

Рисунок 3.12 – Представляє інтерфейс редактора Unity для компонента Lever.

Всі ці елементи середовища поєднуються, щоб створити багатий, цікавий геймплей, змушуючи гравця взаємодіяти з середовищем, збирати предмети та досліджувати світ гри.

3.4.4 Кінцевий будинок

Кінцевий будинок служить ключовим елементом завершення рівня, зображений на рисунку 3.13. Це велика будівля, яка має унікальний дизайн, що відрізняє її від інших будівель у грі. При взаємодії з Кінцевим будинком, гравець переходить на наступний рівень або, якщо це був останній рівень, повертається до головного меню.

Рисунок 3.13 – Кінцевий будинок

Так само як і з важелем, при вході гравця в зону взаємодії з Кінцевим будинком з'являється іконка відповідної кнопки, що підказує гравцеві про можливість взаємодії з ним. Після взаємодії з цим об’єктом, гравець може побачити кількість зірок, отриманих за проходження рівня, у меню рівнів.

Даний об’єкт має наступні компоненти: Transform, SpriteRenderer, Circle Collider 2D і LevelEndController.

– Transform: визначає позицію, обертання і масштаб об'єкта в просторі сцени;

– SpriteRenderer: відповідає за відображення графічного зображення, або спрайту, для об'єкта;

– Circle Collider 2D: додає об'єкту колову область зіткнення, що визначає, коли і як об'єкт реагує на зіткнення з іншими об'єктами. Для цього компонента встановлено галочку IsTrigger, що робить об'єкт "триггером", внаслідок чого інші об'єкти можуть проникнути в область зіткнення, а зіткнення буде оброблено через події тригера;

– LevelEndController: кастомний компонент, що відповідає за логіку завершення рівня.

Основні параметри компонента LevelEndController, що визначають поведінку кінцевого будинку, представлені у таблиці 3.5.

Таблиця 3.5 – Опис параметрів компонента Lever

Параметр	Опис
canvas	Canvas, що відображає візуальні елементи при зіткненні гравця з виходом з рівня.
player	Ссилка на компонент PlayerController, який контролює гравця.
gemsForStars	Масив, що визначає, скільки драгоцінних каменів потрібно для зароблення кожної з зірок на рівні.

Ці параметри можна налаштувати безпосередньо у редакторі Unity для досягнення бажаного поведінки кінцевого будинку, інтерфейс цього компонента зображений на рисунку 3.14.

Рисунок 3.14 – Інтерфейс у редакторі Unity для компонента LevelEndController.

Важливо відзначити, що гравець може перейти на наступний рівень, навіть якщо у нього немає дорогоцінних каменів. Це дозволяє гравцям з різними стилями гри просуватися вперед незалежно від того, наскільки детально вони досліджують кожен рівень. Однак, щоб отримати максимальну кількість зірок і повністю завершити гру, гравцю слід досліджувати рівні якомога глибше, збираючи всі дорогоцінні камені.

Загалом, кожен об'єкт середовища розроблений з метою створити глибоку і пригодницьку атмосферу, що підсилюється різноманітністю механік гри і сюжетним прогресом. Будинки в грі відіграють важливу роль в цьому процесі, створюючи точки орієнтації для гравця і додаючи контексту до середовища гри.

3.5 Загрозливі об'єкти

У грі існує два загрозливі об'єкти, один з них це шипи. Шипи в грі представляють серйозну загрозу для персонажа, зображені на рисунку 3.15. Якщо персонаж торкнеться шипів, його відкидає назад, він отримує одну одиницю шкоди і починає мерехтіти на декілька секунд.

Шипи можуть бути розташовані як на землі, так і на стелі, наприклад, в печерах, що змушує гравця бути уважним і стратегічним у своїх діях.

Рисунок 3.15 – Інтерактивний об'єкт – шипи

Другим загрозливим об’єктом є прірви. Прірви в грі слугують як фінальні загрози для гравця. Якщо персонаж впаде в прірву, він вмирає негайно, не зважаючи на кількість здоров'я. Це відрізняє прірви від звичайних ям, в які можна впасти без шкоди для персонажа.

Біля прірв зазвичай розташовуються таблички з черепом, які попереджають гравця про небезпеку, приклад прірви зображений на рисунку 3.16. Це заохочує гравця до стратегічного планування своїх дій, уникаючи прірв і вибираючи найбезпечніші шляхи.

Рисунок 3.16 – Прірва

Об'єкти-пастки, такі як шипи та прірви, в грі мають декілька спільних компонентів, але використовують різні скрипти для унікальної поведінки пасток, ці компоненти представлені у таблиці 3.6.

Таблиця 3.6 – Опис параметрів компонента Lever

Об'єкт	Компонент	Параметр	Опис
Шип & Прірва	Transform	-	Визначає позицію, обертання і масштаб об'єкта в просторі сцени.
Шип & Прірва	SpriteRenderer	-	Відповідає за відображення графічного зображення, або спрайту, для об'єкта. Для прірви це зображення черепа, яке індикують їх небезпечність.
Шип & Прірва	Box Collider 2D	IsTrigger	Додає об'єкту прямокутну область зіткнення, яка реагує на зіткнення з іншими об'єктами. IsTrigger встановлено в true, що дозволяє іншим об'єктам проникати в область зіткнення, а зіткнення обробляються через події тригера.
Шип	Spike	damage	Кількість шкоди, що наноситься гравцю при контакті з шипом.
Прірва	FallingTrap	-	Скрипт негайно "вбиває" гравця при контакті. Не має додаткових параметрів для налаштування.

Ці об'єкти-пастки можна налаштовувати в редакторі Unity, змінюючи їхні параметри, для створення унікальних викликів на різних рівнях гри. Отже, об'єкти навколишнього середовища підвищують рівень виклику в грі, вимагаючи від гравця виважених та уважних дій для уникнення небезпеки.

3.6 Інтерфейс користувача

Інтерфейс користувача в грі простий та інтуїтивний, що допомагає гравцю легко зорієнтуватися в грі.

У верхній лівій частині екрану розташовано індикатор зібраних дорогоцінних каменів. Іконка дорогоцінного каменя знаходиться поряд із числовим значенням, яке вказує на поточну кількість каменів, зібраних гравцем.

У правій нижній частині екрану розташована полоса здоров'я персонажа. Ця полоса відображає поточний стан здоров'я персонажа, дозволяючи гравцю оцінити, скільки ударів він може витримати перед поразкою. Вигляд інтерфейсу користувача зображений на рисунку 3.17.

Рисунок 3.17 – Інтерфейс користувача

Інтерфейс користувача важливий елемент гри, який допомагає гравцю краще розуміти свої прогрес та поточний стан.

3.7 Організація коду та структура проєкту

Організація коду та структура проєкту відіграють важливу роль у розробці програмного продукту. При створенні великого проєкту, як наприклад відеогра, є важливо забезпечити зручність та ефективність роботи з кодом. В цьому процесі допомагають такі концепції, як простір імен та діаграми класів.

Простір імен допомагають у групуванні пов'язаних класів та об'єктів, уникненні конфліктів імен, а також поліпшують читабельність коду. Вони забезпечують структурованість та логічність у розподілі об'єктів програми.

Діаграми класів відображають структуру системи за допомогою показу класів, їх атрибутів і взаємозв'язків між ними. Вони дозволяють візуалізувати структуру програми, спрощують розуміння коду та забезпечують зручність у налагодженні взаємодії між компонентами системи.

Було створено діаграму простір імен та діаграму класів, які дозволили забезпечити ефективну структуру проєкту та забезпечити легкість розуміння. Ці діаграми наведені у графічних матеріалах (ІС91.030БАК.004 Д2, ІС91.030БАК.004 Д3).

3.8 Збереження даних гри

Створення стабільної та ефективної системи збереження даних в комп'ютерній грі - це завдання, що потребує досконалого розуміння і використання різних технологій. У грі використовується комбінація двох стратегій збереження: PlayerPrefs для зберігання прогресу гравця і бінарне збереження для зберігання налаштувань гравця та стану гри.

PlayerPrefs: Це вбудований в Unity механізм збереження даних, простий у використанні та ефективний для зберігання невеликих обсягів даних. В даному випадку цей механізм використовується для збереження прогресу гравця на кожному рівні, включаючи кількість отриманих гравцем зірок. Це дозволяє швидко

і просто записати та відновити ці дані, забезпечуючи неперервність прогресу гравця.

Бінарне збереження: Надає більше гнучкості та безпеки в порівнянні з PlayerPrefs, оскільки воно дозволяє зберігати більші обсяги даних і робить це більш безпечним способом. Бінарне збереження використовується для збереження налаштувань гравця та більш складних даних про стан гри, включаючи місцеположення персонажа, кількість здоров'я і зібраних каменів.

Для кожного рівня було створено детальний файл з назвою "player_" + назва рівня + ".bin". Ці файли надають потрібну гнучкість та детальність для зберігання даних гри.

Завдяки цим двом методам була створена система збереження даних, яка забезпечує плавний, приємний досвід гравця. Гравці можуть бути впевнені, що їх прогрес та налаштування зберігаються безпечно, дозволяючи їм насолоджуватися грою без страху втрати свого прогресу.

Основні класи та методи, що беруть участь у збереженні даних, подані у Таблиці 3.7.

Таблиця 3.7 – Опис класів та методів, використаних системою збереження даних.

Клас	Метод/Конструктор	Опис
SaveSystem	SavePlayerData()	Зберігає інформацію про гравця, включаючи його поточне здоров'я, кількість зібраних каменів та місце розташування. Дані серіалізуються і зберігаються в бінарному файлі.
SaveSystem	LoadPlayer()	Завантажує збережені дані гравця з бінарного файлу. Якщо файл існує, дані десеріалізуються і повертаються.

SaveSystem	SavePlayerSettings()	Зберігає налаштування гравця, такі як рівень гучності, стан повноекранного режиму і вибране розширення екрана.
SaveSystem	LoadPlayerSettings()	Завантажує збережені налаштування гравця. Якщо файл існує, дані десеріалізуються і повертаються.
SaveSystem	ResetPlayerData()	Скидає збережені дані гравця, видаляючи відповідний файл.
PlayerData	PlayerData()	Конструктор, який збирає дані з контролера гравця (PlayerController) та компонента руху гравця (PlayerMovement2D). Він зберігає поточне здоров'я гравця, кількість зібраних каменів та поточну позицію гравця.
PlayerSettings	PlayerSettings()	Конструктор, який збирає дані з меню налаштувань (SettingMenu). Він зберігає поточне розширення екрана, рівень гучності і стан повноекранного режиму.

У грі використовуються чекпоінти, які допомагають гравцю відновити свій прогрес у разі поразки або виходу з гри. Чекпоінт (від англ. checkpoint, що можна перекласти як "контрольний пункт") - це термін у відеоіграх, що вказує на точку або зону на мапі, де прогрес гравця автоматично зберігається. Це забезпечує можливість відновити гру з цього місця у разі поразки, виходу з гри або початку нової сесії гри, чекпоінт зображений на рисунку 3.18.

Рисунок 3.18 – Вигляд чекпоінта у грі

У грі чекпоінт представлений у вигляді дому на дереві, що візуально відрізняється від інших об'єктів на мапі. Будинок на дереві було обрано через те, що дома символізують безпечну зона. На мапі всього існує три види будинків: на початку рівня, у кінці та на контрольних пунктах. При досягненні чекпоінта, прогрес гравця зберігається, дозволяючи йому продовжити гру з цього місця при наступному запуску гри. Процес збереження даних на чекпоінтах описаний у Таблиці 3.6.

Дім на дереві використовується як візуальний маркер чекпоінта, вказуючи гравцю, що ця зона безпечна і її досягнення приведе до збереження прогресу.

Висновок до третього розділу

Подробиці створення ігрового персонажа були розкриті в цьому розділі. Зокрема, наголошено на важливості анімації персонажа, що не тільки придає життєвість та унікальність ігровому герою, але і є ключовим елементом, який

допомагає занурити гравця в ігровий світ. Додатково, налаштування камери гравця було визначено як важливий етап, що забезпечує оптимальний вигляд гри для користувача.

У розділі було також описано створення інтерактивних об'єктів та елементів середовища. Це включає розробку вишень, дорогоцінних каменів, важелів, та кінцевого будинку. Кожен з цих об'єктів має свою роль і впливає на динаміку гри, створюючи унікальний досвід для гравця. Деталі їхньої роботи, а також інформація про використані класи, представлені у таблицях 3.3, 3.4, 3.5 та 3.6.

Загрозливі об'єкти, що описуються в цьому розділі, додають грі напруженості та виклику. Вони надають гравцеві можливість стратегічно планувати свої дії, а також додають ігровому середовищу додаткового контексту та інтересу.

Розробка інтерфейсу користувача є ще одним важливим аспектом. Комфорт та зручність користувача є первинними пріоритетами при створенні ігрового досвіду, і правильно створений інтерфейс може значно покращити цей досвід.

На завершення розділу 3 було описано систему збереження даних гри. Ця система є ключовою для відтворення стану гри при повторному вході гравця. Система використовує бінарне збереження для ефективного зберігання даних, що дозволяє зберегти інформацію про рівень, здоров'я персонажа, кількість зібраних дорогоцінних каменів і т.д. Всі деталі цієї системи описано у таблиці 3.7.

Отже, у розділі 3 була продемонстрована робота над різними компонентами ігри, що показали, наскільки різносторонньою і комплексною може бути розробка відеоігри. Важливо зрозуміти, що кожен компонент гри має важливе значення, і кожен з них вимагає ретельної розробки та тестування для створення якісного ігрового досвіду.

4 МАТЕМАТИЧНИЙ РОЗДІЛ

4.1Змістовна постановка задачі

Метою цього розділу є створення ворогів для гри, які будуть представляти собою не просто статичні перешкоди для гравця, але й динамічні сутності, здатні реагувати на дії гравця. Розумні вороги в грі, що здатні адаптуватися до дій гравця і приймати рішення на основі поточного стану гри, забезпечують більш глибокий та більш захоплюючий геймплей.

Вороги, які просто переміщуються вздовж заданого шляху або повторюють певний шаблон поведінки, можуть бути цікавими лише у короткостроковій перспективі, але вони можуть стати передбачуваними і тому менш цікавими з часом. Натомість, вороги, що використовують більш розумні та адаптивні механіки, такі як алгоритм A*, можуть постійно змінювати своє поведінку в залежності від дій гравця, що робить їх більш непередбачуваними і додає до гри більшій глибини.

Таким чином, метою цього розділу є створення двох типів ворогів:

– звичайний ворог, що рухається вздовж певного шляху;

– розумний ворог, що використовує алгоритм A* для пошуку шляху до гравця.

В обох випадках необхідно врахувати різні аспекти ігрового процесу, такі як фізичні обмеження середовища гри, можливі дії гравця та інші. Детальніше про реалізацію цих ворогів буде описано в наступних пунктах.

4.2 Звичайний ворог

Першим звичайним ворогом буде "Слайм", його вигляд зображений на рисунку 4.1.

Рисунок 4.1 – Вигляд "Слайма" у грі

"Слайм", який є одним з базових типів ворогів, характеризується спростованим патерном поведінки: він пересувається по визначеному шляху, роблячи періодичні стрибки. "Слайм" має такі компоненти, представлені у таблиці 4.1.

Таблиця 4.1 – Опис компонентів об’єкта Slime.

Компонент	Опис
Transform	Визначає положення, обертання і масштаб об'єкта в грі.
SpriteRenderer	Використовується для відображення спрайта об'єкта у сцені.
Rigidbody2D	Дозволяє об'єкту взаємодіяти з 2D фізикою в двигуні Unity. У контексті "Слайма", воно контролює рух і стрибки.
CircleCollider2D	Компонент визначає область зіткнення з гравцем.
BoxCollider2D	Компонент визначає область зіткнення з об'єктами на рівні (наприклад, платформами).
Slime (контролер поведінки)	Скрипт, що керує поведінкою слайма, включаючи рух, стрибки, анімацію, та взаємодію з гравцем.

Animator	Керує анімацією об'єкта. В даному випадку, здійснює контроль над анімаціями руху і стрибків слайма.
PlayerDamage	Скрипт, який відповідає за нанесення шкоди гравцеві при зіткненні з слаймом.

Параметри, що впливають на поведінку Слайма, представлені в таблиці 4.2.

Таблиця 4.2 – Параметри, що впливають на поведінку Слайма

Назва параметра	Опис
speed	Швидкість руху Слайма
jumpHeight	Висота стрибка Слайма
jumpDuration	Тривалість стрибка Слайма
waitTimeMin	Мінімальний час очікування перед наступним стрибком
waitTimeMax	Максимальний час очікування перед наступним стрибком
minArcAngle	Мінімальний кут стрибка
maxArcAngle	Максимальний кут стрибка

Існує дві важливі формули, які впливають на поведінку ворога.

Перша з них це швидкість стрибка, використовується для обчислення вектору швидкості під час стрибка. Це забезпечує природність руху, притаманну реальному світу.

Параметри стрибка "Слайма" визначаються шляхом обчислення вектора швидкості стрибка на основі висоти стрибка, тривалості стрибка, швидкості руху і сили гравітації. Математичне представлення формули швидкості (4.1).

$$V = (S, H) * \frac{\sqrt{2 * G * D}}{D}, \quad (4.1)$$

де V – вектор швидкості стрибка; S – швидкість руху слайма; H – висота стрибка слайма; G – сила гравітації (в Unity це властивість Physics2D.gravity.magnitude); D – тривалість стрибка слайма.

Ця формула дозволяє обчислити вектор швидкості, необхідний для стрибка на визначену висоту за визначений час, враховуючи силу гравітації.

Друга важлива формула це кут стрибка, використовується для обчислення напрямку стрибка. Це робить поведінку "Слайм" непередбачуваною і цікавою для гравця. Математичне представлення формули стрибка (4.2) можна записати так:

$$D = Q(0,0,\alpha) * R, \quad (4.2)$$

де D – вектор напрямку стрибка; α – випадковий кут стрибка, обраний з діапазону від minArcAngle до maxArcAngle; Q(0, 0, α) – кватерніон, який представляє обертання на кут α; R – базовий вектор, що вказує направо (або горизонтально).

Кватерніони є числовою системою, що розширює комплексні числа, і вони мають важливе застосування у тривимірному просторі для компактного представлення обертань [12]. Кожен кватерніон складається з однієї реальної частини і трьох уявних частин, і зазвичай його представляють у вигляді четвірки чисел (w, x, y, z).

У нашому конкретному випадку використання Quaternion.Euler(0, 0, α) в Unity створює кватерніон, який представляє обертання на кут α градусів навколо осі Z. Потім це обертання застосовується до вектора (Vector2.right), який вказує напрямок праворуч, щоб отримати новий вектор, що вказує напрямок стрибка.

Це означає, що ми обертаємо базовий вектор (R) на кут α, використовуючи кватерніон обертання (Q), щоб отримати вектор напрямку стрибка (D).

4.3 Ворог на базі алгоритму A*

"Гриф" – це більш складний тип ворога, який використовує алгоритм A* для визначення найкоротшого шляху до гравця. "Гриф" зображений на рисунку 4.2.

Рисунок 4.2 – "Гриф" у грі

Алгоритм A* – це пошуковий алгоритм, який широко використовується в області штучного інтелекту для визначення найкоротшого шляху між двома точками в графі [14,15].

Алгоритм A* працює за принципом "найкращого першого" пошуку, що означає, що він завжди розглядає найкращий шлях, визначений за допомогою функції (4.3) оцінки f(n), яка є сумою двох інших функцій

$$f(n) = g(n) + h(n), \quad (4.3)$$

де n – поточний вузол графу; g(n) – вартість шляху від початкового вузла до вузла n; h(n) – евристична оцінка вартості шляху від вузла n до цільового вузла.

Функція (4.3) g(n) відображає відому вартість шляху від початку до поточного вузла, тоді як h(n) є евристичною оцінкою вартості шляху від поточного вузла до мети. Сума цих двох функцій дає оцінку загальної вартості шляху через вузол n.

Функція g(n) обчислюється як:

$$g(n) = g(p) + c(p, n), \tag{4.4}$$

де p – попередній вузол відносно вузла n; g(p) – вартість шляху від початкового вузла до вузла p; c(p, n) – вартість переходу від вузла p до вузла n.

Функція (4.5) h(n), евристична оцінка, обчислюється використовуючи евклідову відстань, як найменшу відстань між вузлом n та цільовим вузлом:

$$h(n) = \sqrt{(n_x - g_x)^2 + (n_y - g_y)^2}, \tag{4.5}$$

де n_x, n_y – координати поточного вузла; g_x, g_y – координати цільового вузла.

У контексті ворога "Гриф" алгоритм A* використовується для визначення найкоротшого шляху до гравця по ділянці гри, що уявляє собою граф. Використовуючи цей алгоритм, "Гриф" може ефективно навігуватися по складній області гри і швидко знаходити гравця.

Висновок до четвертого розділу

У математичному розділі було детально розглянуто математичні аспекти, які стосуються поведінки двох типів ворогів у розроблюваній грі - "Слайма" і "Грифа". Ці аспекти включають розуміння обертань і руху в комп'ютерній графіці, а також використання алгоритмів для пошуку ворогів.

Для "Слайма" проаналізовані різні формули, які використовуються для керування стрибками цього ворога. Основою для цих формул є закони класичної механіки, які регулюють рух об'єктів. Використання таких формул дозволяє додати рівень реалістичності до руху ворога, забезпечуючи одночасно достатній рівень непередбачуваності, щоб зробити гру захоплюючою. Також було досліджена

механіка кватерніонів

Для "Грифа" було розглянуто використання алгоритму A* для визначення найкоротшого шляху до гравця. Алгоритм A* – це важливий інструмент в області штучного інтелекту для розробки ігор, який дозволяє ворогам ефективно і ефективно навігуватися по складним ігровим сценам. Розібрано основні концепції алгоритму A* та проаналізовано основні формули, які використовуються в алгоритмі.

Математичний розділ підкреслює важливість використання відповідних математичних та фізичних принципів при розробці поведінки ворогів в комп'ютерних іграх. Це не тільки створює реалістичні та цікаві взаємодії між гравцем та ворогами, але також дозволяє грі бути більш розширюваною і адаптованою. Поряд з розумінням ігрового дизайну і програмування, математичне розуміння цих механік є важливим фактором для розробки успішних ігор.

Однак, необхідно зауважити, що ці аспекти поведінки ворога є лише частиною більш широкої картини розробки гри.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Після запуску програми користувач опиняється на головному екрані гри, де доступні кілька дій:

– “Options”: Натискаючи на цю кнопку, користувач потрапляє до меню налаштувань. Тут можна змінити ряд параметрів гри, включаючи налаштування звуку, розширення екрану і режим повного екрану. Якщо користувач хоче повернутися до головного меню, він може натиснути кнопку "Back";

– “Play”: Натиснувши на цю кнопку, користувач потрапляє до меню вибору рівнів. Тут доступні п'ять рівнів для вибору. Щоб повернутися до головного меню, користувач може натиснути кнопку "Back".

В “Меню вибору рівнів” існує п’ять рівнів: Level 1, Level 2, ..., Level 5. Користувач може вибрати один із п'яти рівнів для гри, натиснувши на відповідну кнопку. Після цього обраний рівень починає завантажуватися.

У процесі гри доступні наступні дії:

ESC: Натиснувши цю кнопку на клавіатурі, гра призупиняється, і відкривається меню паузи. З цього меню користувач може повернутися до гри, зайти до налаштувань, повернутися до головного меню або вийти з гри.

У випадку смерті персонажа гравця, відображається екран смерті, зображений на рисунку 5.1, з двома опціями:

– “Restart”: Почати рівень заново;

– “Main Menu”: Вийти до головного меню гри.

Рисунок 5.1 – Екран після смерті персонажа

Всі елементи інтерфейсу користувача, включаючи кнопки управління, меню налаштувань і екрани гри, розроблені таким чином, щоб бути інтуїтивно зрозумілими для користувачів будь-якого рівня досвіду. Це робить "Fox Adventure" легко доступною для всіх, хто хоче насолодитися веселою та захоплюючою грою.

Це керівництво користувача надає основну інформацію про навігацію і можливості в грі, що дозволяє гравцям легко розпочати гру і налаштовувати її за своїм смаком.

5.2 Тестування функціональних вимог

Для забезпечення надійної роботи гри було розроблено набір тестів, результати яких представлені у таблицях 5.1 – 5.9.

Таблиця 5.1 – Перевірка функції запуску певного рівня гравцем

Назва	Перевірка функції запуску певного рівня гравцем
Початковий стан	Гравець знаходиться на сторінці вибору рівнів.
Виконані дії	1. Гравець натискає на кнопку рівня, який він хоче запустити.

Результат	Обраний рівень починає завантажуватися.
Відповідність вимогам	Відповідає

Таблиця 5.2 – Перевірка можливості пересування гравця кнопками на клавіатурі

Назва	Перевірка можливості пересування гравця кнопками на клавіатурі
Початковий стан	Гравець знаходиться на будь-якому рівні гри.
Виконані дії	Гравець натискає кнопки на клавіатурі для переміщення персонажа.
Результат	Персонаж переміщується відповідно до натиснутих кнопок.
Відповідність вимогам	Відповідає

Таблиця 5.3 – Перевірка можливості стрибання персонажа

Назва	Перевірка можливості стрибання персонажа
Початковий стан	Гравець знаходиться на будь-якому рівні гри.
Виконані дії	Гравець натискає кнопку стрибка на клавіатурі.
Результат	Персонаж стрибає.
Відповідність вимогам	Відповідає

Таблиця 5.4 – Перевірка збору дорогоцінних каменів

Назва	Перевірка збору дорогоцінних каменів
Початковий стан	Гравець знаходиться на будь-якому рівні гри, де є дорогоцінні камені.
Виконані дії	Гравець переміщує персонажа до дорогоцінного каменя.
Результат	Персонаж збирає дорогоцінний камінь.
Відповідність вимогам	Відповідає

Таблиця 5.5 – Перевірка збору вишні гравцем

Назва	Перевірка збору вишні гравцем
Початковий стан	Гравець знаходиться в грі, має не повну кількість здоров'я
Виконані дії	1. Гравець переміщується до вишні. 2. Гравець збирає вишню.
Результат	Здоров'я гравця збільшилося на одиницю
Відповідність вимогам	Відповідає

Таблиця 5.6 – Перевірка отримання шкоди від ворогів

Назва	Перевірка отримання шкоди від ворогів
Початковий стан	Гравець знаходиться на будь-якому рівні гри, де є вороги.
Виконані дії	Гравець наближається до ворога.
Результат	Персонаж отримує шкоду.
Відповідність вимогам	Відповідає

Таблиця 5.7 – Перевірка закінчення рівня і отримання зірок в меню вибору рівнів

Назва	Перевірка закінчення рівня і отримання зірок в меню вибору рівнів
Початковий стан	Гравець знаходиться на будь-якому рівні гри.
Виконані дії	1. Гравець завершує рівень. 2. Гравець повертається до меню вибору рівнів.
Результат	У меню вибору рівнів відображається кількість зірок, зароблених на цьому рівні.
Відповідність вимогам	Відповідає

Таблиця 5.8 – Перевірка роботи чекпоінтів

Назва	Перевірка роботи чекпоінтів
Початковий стан	Гравець знаходиться на будь-якому рівні гри, де є чекпоінти.
Виконані дії	1. Гравець доходить до чекпоінта. 2. Персонаж гравця помирає.
Результат	Персонаж гравця відроджується на останньому активному чекпоінті.
Відповідність вимогам	Відповідає

Таблиця 5.9 – Перевірка системи зберігання налаштувань гравця

Назва	Перевірка системи зберігання налаштувань гравця
Початковий стан	Гравець знаходиться в меню налаштувань.
Виконані дії	1. Гравець змінює налаштування гри. 2. Гравець виходить з гри 3. Гравець знову запускає гру.
Результат	Всі змінені налаштування збереглися і діють після повторного запуску гри.
Відповідність вимогам	Відповідає

Висновок до п’ятого розділу

У цьому розділі було розглянуто технологічні аспекти гри. Було подано керівництво користувача, яке детально описує процеси, пов'язані з використанням гри: від запуску програми до керування персонажем та вибору рівнів. Потім було проведено тестування функціональних вимог. Цей розділ дає загальне уявлення про роботу і використання гри. Завдяки детальному керівництву користувача та тестуванню можна гарантувати, що гра відповідає всім поставленим вимогам і є зручною та інтуїтивно зрозумілою для кінцевого користувача.

ВИСНОВОК

Виконання цього дипломного проєкту дало можливість заглиблено вивчити та розуміти галузь розробки відеоігор. Аналіз предметної області, включаючи дослідження сучасного стану ігрової індустрії та різних ігрових движків, дозволив обрати найбільш підходящий движок – Unity.

Процес планування та розробки гри включав в себе створення детальної концепції гри, аналіз основних механік і їх вплив на геймплей, дослідження цільової аудиторії, а також аналіз схожих ігор на ринку. Було виконано вибір та імпорт необхідних графічних активів та звуків, налаштування інтерфейсу користувача, та було проведено розробку основних компонентів гри, включаючи створення карт рівнів, розробку головного персонажа, інтерактивних об'єктів і елементів середовища, а також систему збереження даних гри.

Математична модель поведінки ворогів, реалізована на основі алгоритму A*, відіграла важливу роль у розробці геймплею, створюючи динамічне і змістовне середовище для гравця.

Додатково до вже виконаного аналізу та розробки, відбулося налаштування фізики персонажів, використання компонентів Unity для реалізації руху гравця та взаємодії з середовищем, розробка UI та UX елементів, які включають меню, навігаційні елементи та інтерфейс здоров'я.

Окрім розробки гри, також була зроблена велика робота по впровадженню гнучкого та ефективного процесу розробки, включаючи управління версіями за допомогою системи контролю версій Git, що дозволило забезпечити безперебійний робочий процес та зручне співробітництво.

Також слід зазначити використання різноманітних бібліотек та інструментів, таких як TextMeshPro для керування текстовим контентом, Animator для створення плавних та натуральних анімацій, та ряд інших інструментів для керування звуковими ефектами, управлінням камери та інтерфейсу користувача.

Проєкт також включав створення детальних діаграм, які допомагають

візуалізувати архітектуру гри та процеси в ній. Діаграми класів, станів, просторів імен та варіантів використання допомагають у візуалізації взаємодії між різними елементами гри та їх функціоналу.

Завершальним етапом стала розробка керівництва для користувача та проведення тестування, яке підтвердило відповідність розробленої гри встановленим вимогам.

Кінцева версія гри надає гравцям багато функцій та можливостей, включаючи різноманітні рівні, цікаві головоломки, динамічні противники та змінні середовища.

Високий рівень ретельної розробки і планування, а також дотримання балансу між геймплеєм і візуальною привабливістю, гарантують, що "Fox Adventure" є високоякісним продуктом, що задовольняє потреби різних гравців.

Таким чином, як результат виконання цього проєкту, було створено повноцінний 2D платформер, який включає в себе всі основні механіки сучасних ігор даного жанру. Гра має потенціал для подальшого розвитку та удосконалення.

Виконання цього проєкту дало можливість глибше зрозуміти роботу сучасних ігрових движків, а також розвинути навички роботи з комплексними системами та алгоритмами. Цей досвід стане важливою базою для подальшої роботи в області розробки відеоігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Newzoo Global Games Market Report". Newzoo. [Електронний ресурс]. URL: https://newzoo.com/

2. "Newzoo Global Games Market Report". Newzoo. [Електронний ресурс]. URL: "Unity Game Engine". Unity Technologies. URL: https://unity.com/

3. "Unreal Engine". Epic Games. [Електронний ресурс]. URL: https://www.unrealengine.com/

4. "Godot Engine". Godot. [Електронний ресурс]. URL: https://godotengine.org/

5. "GameMaker Studio". YoYo Games. [Електронний ресурс]. URL: https://www.yoyogames.com/

6. Unity Asset Store. Sunny Land by Ansimuz [Електронний ресурс]. URL: https://assetstore.unity.com/packages/2d/characters/sunny-land-103349

7. Unity Asset Store. 371 Simple Buttons Pack by DEV.X [Електронний ресурс]. URL: https://assetstore.unity.com/packages/2d/gui/icons/371-simple-buttons-pack-97516

8. Unity Technologies. Unity User Manual (Version 2023.1). Unity Documentation [Електронний ресурс]. URL: https://docs.unity3d.com/Manual/index.html

9. JetBrains. Rider Help. JetBrains [Електронний ресурс]. URL: https://www.jetbrains.com/help/rider/

10. Microsoft Corporation. C# Programming Guide. Microsoft Docs [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/

11. Hocking, J. Multiplatform Game Development in C#. 2nd Edition. Sams Publishing.

12. Furrer, F., Burri, M., Achtelik, M., & Siegwart, R. (2020). Quaternion attitude estimation from vector observations using a matrix kalman filter. Proceedings of the American Control Conference.

13. Rabin, S. (2018). "Improvements to A* Search for Real-Time Applications". Game AI Pro 360: Guide to Pathfinding. CRC Press. Pp. 91-110.

14. Harabor, D. D., & Grastien, A. (2017). "Online Graph Pruning for Pathfinding on Grid Maps". IEEE Transactions on Computational Intelligence and AI in Games.

15. Schell, J. (2019). The Art of Game Design: A Book of Lenses, Third Edition.