Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка комп'ютерної гри «Runestone Rumble» у жанрі Платформер засобами рушія Unity

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Бодлак А.В.
(підпис) (прізвище та ініціали)

Керівник _____ Гром'як Р.С.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«___» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Бодлаку Артуру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка комп'ютерної гри «Runestone Rumble» у жанрі Платформер засобами рушію Unity

Керівник роботи Гром'як Роман Сильвестрович, кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29* « *квітня* 2024 року № *4/7-470*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Документація ігрового рушія Unity. Приклади кодів для реалізації ігрових механік. Наукові статті та книги з теорії розробки ігор. Технічна документація для Visual Studio. Матеріали та інструменти для створення анімацій і графіки (Aseprite). Звіти з тестування ігрових механік. Посібники та відеоуроки з програмування на C#. Статті та підручники з проектування рівнів та ігрових механік. Матеріали з вебінарів та семінарів з розробки ігор.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Актуальність теми. Мета і задачі дослідження. Практичне значення результатів. Розділ 1. Аналіз предметної області та постановка завдання для розробки гри «Runestone Rumble».(1.1. Вступ до предметної області платформерних ігор. 1.2. Ключові елементи платформерних ігор. 1.3. Аналіз існуючих рішень. 1.4. Постановка основного завдання на створення гри. 1.5. Обґрунтування вибору інструментів та технологій.) Розділ 2. Проектна частина та розробка концепцій. (2.1. Основні елементи гри та принципи роботи. 2.2. Структура рівнів та віртуальна камера. 2.3. Планування ігрового світу і систем взаємодії. 2.4. Звукове оформлення.) Розділ 3. Практична частина та реалізація ігрових механік.(3.1. Керування персонажем. 3.2. Системи взаємодії з гравцем. 3.3. Освітлення та камера.)
4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки (Додаток А. Частина скрипта реалізація рухом персонажем, Додаток Б Реалізація скрипта контролю платформами, Додаток В Реалізація скрипта можливості зміни гравітації, Додаток Е Реалізація скрипта можливості зміни гравітації)
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С. к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел для створення відеогри	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо жанру платформера,	09.02.2024-20.04.2024	*Виконано*
	вибір програмних забезпечень.		
	Розроблення гри «Runstone Rumble»		
5.	Оформлення розділу «Аналіз предметної області та	26.04.2024-02.05.2024	*Виконано*
	постановка завдання для розробки гри «Runestone Rumble»		
6.	Оформлення розділу «Проектна частина та розробка	02.05.2024-08.05.2024	*Виконано*
	концепцій.»		
7.	Оформлення розділу «Практична частина та реалізація	15.05.2024-16.05.2024	*Виконано*
	ігрових механік»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2024-06.06.2024	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	07.06.2024-08.06.2024	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	09.06.2024-11.06.2024	*Виконано*
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи	25.06.2024	

Студент _____ Бодлак А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гром'як Р.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка комп'ютерної гри "Runstone Rumble" у жанрі платформер засобами рушія Unity // Кваліфікаційна робота освітнього рівня «Бакалавр» // Бодлак Артур Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-41 // Тернопіль, 2024 // С.68 , рис. – 15, табл. – 1, слайди. – 15, додат. – 4, бібліогр. – 35.

Ключові слова: платформер, ігровий рушій, Unity, анімація, комп'ютерна гра, програмування, 2D-гра, інтерфейс користувача.

Кваліфікаційна робота присвячена дослідженню розробки комп'ютерної гри «Runestone Rumble» у жанрі платформер з використанням рушія Unity.

У першому розділі кваліфікаційної роботи описано аналіз предметної області та постановка завдання для розробки гри. Висвітлено історію розвитку платформерних ігор, основні елементи жанру, а також обґрунтовано вибір інструментів та технологій для реалізації проекту. Розглянуто існуючі рішення в жанрі платформерів і проаналізовано їхні ключові характеристики.

У другому розділі кваліфікаційної роботи розглянуто проектну частину та розробку концепцій гри. Досліджено основні елементи гри та принципи роботи, описано структуру рівнів та взаємодію гравця з ігровим середовищем. Подано детальну розробку ігрових механік, включаючи систему керування персонажем, анімації, та використання рунічних каменів.

У третьому розділі кваліфікаційної роботи описано практичну частину та реалізацію ігрових механік. Проаналізовано технічні аспекти розробки, проведено тестування гри та налагодження для забезпечення її коректної роботи. Проведено оптимізацію ігрових ресурсів і коду для покращення продуктивності.

ANNOTATION

Development of the computer game "Runstone Rumble" in the Platform Genre using the Unity Engine // Qualification work of the educational level «Bachelor» // Bodlak Artur // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2024 // P.68, fig. - 15, tabl. - 1, slides. - 15, annexes. – 4, references - 35.

Keywords: platformer, game engine, Unity, animation, computer game, programming, 2D game, user interface.

The qualification work is dedicated to the development of the computer game "Runstone Rumble" in the platformer genre using the Unity engine.

The first section of the qualification paper describes the analysis of the subject area and the task setting for game development. It highlights the history of platformer games, the main elements of the genre, and justifies the choice of tools and technologies for the project implementation. Existing solutions in the platformer genre are reviewed and their key characteristics are analyzed.

The second section of the qualification paper examines the project part and the development of the game concepts. It explores the main elements of the game and principles of operation, describes the structure of the levels and the interaction of the player with the game environment. Detailed development of game mechanics, including the character control system, animations, and the use of runestones is provided.

The third section of the qualification paper describes the practical part and the implementation of game mechanics. The technical aspects of development are analyzed, game testing and debugging are conducted to ensure its correct operation. Game resources and code are optimized to improve performance.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Unity – Ігровий рушій для розробки дво та тривимірних ігор

Visual Studio – Інтегроване середовище розробки (IDE) від Microsoft.

Анімація – Процес створення рухомих зображень.

Платформер – Жанр комп'ютерних ігор, де основною механікою є стрибки між платформами.

Рівень – Структурний елемент комп'ютерної гри, який представляє собою окремий ігровий етап або зону.

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ГРИ «RUNESTONE RUMBLE» 8
1.1 Вступ до предметної області платформерних ігор 8
1.2 Ключові елементи платформерних ігор 12
1.3 Аналіз існуючих рішень 14
1.4 Постановка основного завдання на створення гри 18
1.5 Обґрунтування вибору інструментів та технологій 21
1.6 Висновок до першого розділу 26
РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА ТА РОЗРОБКА КОНЦЕПЦІЙ 28
2.1 Основні елементи гри та принципи роботи 28
2.2 Структура рівнів та віртуальна камера 33
2.3 Планування ігрового світу і систем взаємодії 38
2.4 Звукове оформлення 41
2.5 Висновок до другого розділу 43
РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА ТА РЕАЛІЗАЦІЯ ІГРОВИХ МЕХАНІК 45
3.1 Керування персонажем 45
3.2 Системи взаємодії з гравцем 50
3.3 Освітлення та камера 53
3.4 Висновок до третього розділу 55
РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 57
4.1 Ергономічні аспекти організації робочого місця розробника комп'ютерних ігор 57
4.2 Вимоги безпеки до робочих місць для виконання робіт 60
4.3 Висновки до четвертого розділу 62
ВИСНОВКИ 64
ПЕРЕЛІК ДЖЕРЕЛ 66

ВСТУП

Актуальність теми. В сучасному світі індустрія відеоігор стрімко розвивається, стаючи однією з найважливіших і найприбутковіших галузей розважальної індустрії. Платформерні ігри, як один з найстаріших і найпопулярніших жанрів, продовжують утримувати лідируючі позиції завдяки своїй динамічності. Зважаючи на швидкий розвиток технологій та постійне зростання вимог споживачів, створення інноваційних ігор є важливим аспектом. У цьому контексті розробка комп'ютерної гри «Runestone Rumble» у жанрі платформер із використанням рушія Unity є актуальним напрямком сучасних досліджень у галузі комп'ютерних наук та ігрової індустрії.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є створення комп'ютерної гри у жанрі платформер з використанням механік рунічних каменів, що забезпечить унікальний ігровий досвід та високу інтерактивність. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати стан досліджень у галузі розробки платформерних ігор, визначити основні тенденції та успішні приклади.

– Обґрунтувати вибір технологій та інструментів для розробки гри.

– Розробити концепцію гри з унікальними механіками рунічних каменів.

– Реалізувати основні елементи гри, включаючи систему керування персонажем, структуру рівнів та візуальні ефекти.

– Провести тестування та налагодження гри для забезпечення її коректної роботи.

Практичне значення одержаних результатів. Отримані в результаті даної кваліфікаційної роботи результати можуть бути використані для подальшого вдосконалення методів розробки ігор, навчання студентів відповідних спеціальностей, а також у комерційних проектах з розробки відеоігор.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ГРИ «RUNESTONE RUMBLE»

1.1 Вступ до предметної області платформерних ігор

Платформні ігри один з найстаріших і найпопулярніших жанрів в індустрії відеоігор. Цей жанр характеризується ігровим процесом, який включає стрибки між платформами, подолання перешкод і ворогів, а також виконання різноманітних завдань. Платформери можуть бути як двовимірними (2D), так і тривимірними (3D), але основна механіка залишається незмінною [1].

Основні риси жанру платформерних ігор. Переміщення по платформах. Основною механікою платформерів є стрибки між різними платформами. Гравець повинен точно розраховувати час і силу стрибка, щоб уникати падіння та інших небезпек.

Платформери зазвичай включають різноманітні перешкоди, такі як ями, шипи, рухомі платформи, які гравець повинен подолати. Це додає геймплею складності та викликів. У більшості платформерів гравець стикається з ворогами, яких потрібно уникати або перемагати. Це можуть бути різні істоти, машини або інші персонажі.

Збирання монет, дорогоцінних каменів або інших предметів є важливою частиною ігрового процесу. Це може бути як додаткове завдання, так і необхідна умова для просування по сюжету гри.

Платформери складаються з рівнів, кожен з яких має свою унікальну архітектуру і завдання. Рівні можуть бути об'єднані в світи або епізоди, що надає грі структуру та прогресію. Платформери включають кілька піджанрів, кожен з яких має свої особливості.

Самою першою відомою грою у стилі 2D є «Tennis for Two», створеною американцем Вільямом Гіггсом у 1958 році [2]. Хоча гра не є платформером у його розумінні но це можна сказати є засновник 2D ігор в певному розумінні.

Класично піджанр платформера включає традиційні ігри з двовимірною графікою, такі як «Super Mario Bros» і «Sonic the Hedgehog». Гравець пересувається вліво або вправо, стрибаючи по платформах і долаючи перешкоди.

Метроїдванія у цьому піджанрі поєднує елементи платформерів з дослідженням і прогресивною відкритістю світу. Ігри цього типу, такі як «Metroid» і «Castlevania: Symphony of the Night», пропонують нелінійну структуру рівнів і вимагають від гравця пошуку нових здібностей для просування по грі.

Тривимірні платформери з розвитком технологій з'явилися 3D-платформери, такі як «Super Mario 64» і «Banjo-Kazooie». Ці ігри дозволяють гравцям вільно переміщуватися в тривимірному просторі, додаючи нові виміри до геймплею.

Інді-платформери з початку 2010-х років платформери стали популярними серед незалежних розробників. Ігри, такі як «Celeste» і «Hollow Knight», поєднують ретро-естетику з інноваційними механіками і глибокими сюжетами.

Платформерні ігри є одним з найбільш впливових жанрів в історії відеоігор. Їх розвиток можна розділити на кілька ключових етапів, кожен з яких вніс свій внесок у формування жанру, який ми знаємо сьогодні.

Виділемо етапи розвитку 2D відеоігор як явища у світі [3]. Перші кроки (1970-ті – початок 1980-х років). Платформерні ігри почали з'являтися наприкінці 1970-х років, коли індустрія відеоігор перебувала ще на стадії становлення.

Однією з перших ігор, яка вплинула на жанр, була «Space Panic» (1980) від Universal. Хоча вона не мала можливості стрибків, яка стала ключовою для майбутніх платформерів, вона включала елементи лазання по драбинах і уникання ворогів, що стало основою для багатьох наступних ігор.

Золотий вік аркадних ігор (середина 1980-х років) Справжній прорив у жанрі стався в 1981 році з виходом «Donkey Kong» від Nintendo. Гра представила головного героя Маріо (тоді відомого як Jumpman), який повинен був рятувати дівчину від гігантської мавпи, стрибаючи через різноманітні перешкоди і

підіймаючись по драбинах. Ця гра стала першою, яка включала повноцінні механіки стрибків і підйомів, що стали невід'ємною частиною жанру.

Встановлення стандартів (кінець 1980-х – початок 1990-х років) В 1985 році Nintendo випустила «Super Mario Bros.», яка стала визначною подією для жанру платформерів. Ця гра встановила нові стандарти для дизайну рівнів, плавності керування і різноманітності ігрових механік.

Гравці керували Маріо, який пересувався через різні світи, стрибав через перешкоди, збирав монети і боровся з ворогами. «Super Mario Bros.» до сьогодні залишається однією з найвідоміших і найуспішніших відеоігор в історії, продано більше 40 мільйонів копій цієї гри.

Епоха консолей і поява нових героїв (середина 1990-х років) З розвитком 16-бітних консолей, таких як Sega Genesis і Super Nintendo Entertainment System (SNES), платформерні ігри отримали новий імпульс.

У 1991 році Sega випустила «Sonic the Hedgehog», яка стала іконою для компанії та сильною конкурентною відповіддю на успіх Маріо. Гра відзначалася високою швидкістю і плавністю керування, а також яскравою графікою і запам'ятовуваним саундтреком (Sheff, 1993).

В цей же період, у 1994 році, Rare випустила «Donkey Kong Country» для SNES, яка вразила гравців своєю передовою графікою і анімацією, що використовували 3D моделі, перетворені в 2D спрайти. Ця гра стала однією з найуспішніших на платформі SNES, закріпивши популярність серії Donkey Kong (Donovan, 2010).

Перехід до 3D і нові виклики (кінець 1990-х – початок 2000-х років) Перехід до тривимірної графіки відкрив нові можливості для жанру платформерів. Гра «Super Mario 64» (1996), випущена для Nintendo 64, стала революційною, представивши повноцінний тривимірний світ для дослідження. Гравці могли вільно переміщуватися в усіх напрямках, стрибати і взаємодіяти з об'єктами в тривимірному просторі. Ця гра стала основою для багатьох майбутніх 3D-платформерів (Wolf, 2008).

Інші значущі 3D-платформери того часу включають «Crash Bandicoot» (1996) від Naughty Dog для PlayStation і «Banjo-Kazooie» (1998) від Rare для Nintendo 64. Ці ігри розширили межі жанру, пропонуючи більш складні світи, історії і персонажів.

Сучасний стан і вплив на інші жанри (2010-ті роки – сьогодення). З початку 2010-х років платформерні ігри знову набули популярності завдяки незалежним розробникам.

Такі ігри, як «Super Meat Boy» (2010) і «Celeste» (2018), поєднали ретро-естетику з сучасними механіками, здобувши визнання критиків і гравців. Багато сучасних ігор, таких як екшн-адвенчури та метройдванії, використовують механіки, запозичені з платформерів.

Наприклад, ігри серії «Metroid» та «Castlevania» поєднують елементи платформерів з відкритим світом та дослідженням, створюючи унікальний ігровий досвід (Wolf, 2008).

Платформерні ігри продовжують еволюціонувати, адаптуючись до нових технологій та тенденцій. Вони залишаються важливою частиною індустрії відеоігор, впливаючи на дизайн ігор та формуючи досвід гравців по всьому світу.

Entertainment Software Rating Board (ESRB) — це організація, яка займається класифікацією відеоігор та іншого розважального програмного забезпечення в Північній Америці. ESRB створена у 1994 році для забезпечення належної інформації про зміст ігрових продуктів, що допомагає споживачам, особливо батькам, приймати обґрунтовані рішення про придбання та використання відеоігор. ESRB використовує різні рейтингові категорії для визначення відповідності ігор для різних вікових груп [4].

Early Childhood (EC) ігри, які підходять для дітей від трьох років. Вони не містять матеріалів, які батьки вважали неприйнятними. Everyone (E) ігри, які підходять для всіх вікових груп. Можуть містити мінімальне насильство або рідкісні незначні слова.

Everyone 10+ (E10+) ігри, які підходять для дітей від десяти років і старше. Можуть містити деяке насильство, незначні грубі вислови або мінімальні натяки

на теми для дорослих. Teen (T) ігри, які підходять для підлітків від тринадцяти років і старше. Можуть містити насильство, грубі вислови або теми для дорослих. Mature (M) для осіб від сімнадцяти років і старше. Можуть містити інтенсивне насильство, кров, сексуальні теми або грубу мову. Adults Only (AO) які підходять тільки для дорослих від вісімнадцяти років. Містять тривалі сцени інтенсивного насильства або сексуальні теми.

Окрім вікових рейтингів, ESRB також надає контентні дескриптори, які описують конкретні елементи гри, що вплинули на її рейтинг. До таких дескрипторів можуть входити: насильство, мова, використання наркотиків та алкоголю, сексуальний зміст і азартні ігри. Наприклад, гра може мати рейтинг "Mature" з дескрипторами "Violence" та "Strong Language", що означає, що в грі присутнє інтенсивне насильство і груба мова.

1.2 Ключові елементи платформерних ігор

Платформерні ігри відомі своїми унікальними механіками і елементами, які роблять їх відмінними від інших жанрів. Розглянемо детальніше ключові елементи, які визначають платформерні ігри [5].

Рівні (Levels) є основними елементами геймплею платформерних ігор. Вони являють собою окремі ігрові зони, через які гравець повинен пройти, долаючи перешкоди і ворогів.

Розміщення платформ різної ширини, висоти та відстані одне від одного є ключем до створення цікавого та динамічного геймплею. Платформи можуть використовуватися для перевірки точності стрибків гравця, створення складних маршрутів або приховування секретів.

Кожен рівень має унікальну архітектуру, стиль і складність, що створює різноманітний і захоплюючий ігровий досвід. У класичних платформерах, таких як «Super Mario Bros», рівні зазвичай розділені на світи, кожен з яких містить кілька підрівнів і закінчується битвою з босом [6].

У сучасних іграх рівні можуть бути значно більшими і складнішими, часто містять підрівні і секретні зони, що підвищує реіграбельність гри. На рисунку 1.1 продемоснтровано один з рівнів гри.

Рисунок 1.1 – Ілюстрація рівня з гри «Super Mario Bros»

Стрибки (Jumping) є однією з основних механік платформерних ігор. Вони дозволяють персонажу долати перешкоди, уникати ворогів і досягати нових платформ. Різні ігри мають різні фізичні моделі стрибків, що додає їм унікальності. У «Sonic the Hedgehog», наприклад, стрибки є дуже швидкими і точними, що дозволяє гравцю підтримувати високу швидкість протягом гри. У «Super Mario 64» стрибки є більш складними і включають додаткові механіки, такі як подвійні і потрійні стрибки, що відкриває більше можливостей для дослідження ігрового світу.

Збирання предметів (Collectibles) є ще однією ключовою механікою платформерних ігор. Предмети, які можна зібрати, можуть включати монети, дорогоцінні камені, спеціальні предмети, що дають додаткові можливості або покращення. Ці предмети додають додатковий рівень мотивації для гравця і часто є необхідними для досягнення 100% проходження гри. У «Crash Bandicoot» гравці збирають яблука і дорогоцінні камені, які допомагають розблокувати нові

рівні і покращення. У «Donkey Kong Country» збирання бананів і літер «KONG» дає додаткові життя і бонуси [6].

Платформи (Platforms) є основним елементом рівнів у платформерних іграх. Вони можуть бути статичними або рухомими, мати різні форми і розміри, що додає різноманітності і складності до ігрового процесу. У деяких іграх платформи можуть взаємодіяти з іншими елементами гри, такими як перемикачі, механізми і пастки.

1.3 Аналіз існуючих рішень

Платформерні ігри продовжують розвиватися, і багато сучасних ігор стали популярними завдяки своїм інноваційним механікам, привабливій графіці та захоплюючим сюжетам. Розглянемо детальніше деякі з найвідоміших сучасних платформерних ігор, включаючи цікаві факти та нововведення.

Hollow Knight (2017) від Team Cherry вийшла у 2017 році і швидко стала культовою серед гравців зовніший вигляд гри продемонстровано на рисунку 1.2. Гра відзначається своїм атмосферним світом, складними боями та глибокою історією. У грі гравець досліджує великий відкритий світ, бореться з ворогами та боссами, знаходить нові здібності та покращення. Гра поєднує елементи платформера з метроїдванією.

Цікавий факт «Hollow Knight» була створена трьома розробниками, які фінансували проєкт через Kickstarter (платформа для підтримки проєктів). Гра отримала схвальні відгуки за свою атмосферу, дизайн рівнів та саундтрек. З нововведень гра впровадила складну систему боїв і багатошарову історію, що розкривається через дослідження світу.

Окрім згаданих характеристик, «Hollow Knight» пропонує безліч інших цікавих елементів, які роблять її унікальною. Королівство Халлоунест, де розгортається дія гри, сповнене таємниць і секретів. Гравці можуть досліджувати його в будь-якому порядку, відкриваючи нові локації, знаходячи приховані

скарби та розкриваючи захоплюючі історії. Світ «Hollow Knight» населений дивовижними істотами, кожна з яких має свою унікальну особистість та історію.

Гравці можуть спілкуватися з ними, отримувати завдання та дізнаватися більше про історяю гри. У грі представлені епічні битви з босами, які потребують не лише спритності та реакції, але й стратегічного мислення. Кожен бос має свій унікальний стиль бою та слабкі місця, які потрібно вивчити [7]. Музика створена композитором Christopher Larkin, чудово доповнює атмосферу гри та підкреслює емоції кожної сцени. «Hollow Knight» стала не лише культовою грою серед фанатів, але й мала значний вплив на ігрову індустрію. Її успіх надихнув інших розробників на створення подібних ігор, що поєднують в собі елементи платформера, метроїдванії та дослідження відкритого світу.

Рисунок 1.2 – Скріншот з гри Hollow Knight

Ori and the Blind Forest (2015) та її продовження Ori and the Will of the Wisps (2020) від Moon Studios є одними з найкрасивіших платформерних ігор, відомих своїм вражаючим візуальним стилем та емоційними історіями зовнішній вигляд [8] продемонстровано на рисунк 1.3. Геймплейна складова гравець керує Ori,

маленьким духом, що досліджує чарівний ліс, бореться з ворогами та вирішує головоломки. Ігри включають складні платформи та бойові механіки.

Ori and the Blind Forest створювалася протягом чотирьох років і включала міжнародну команду розробників, які працювали дистанційно. Дані ігри встановили нові стандарти для інді-ігор щодо графіки та музики, використовуючи художній стиль, натхненний анімаційними фільмами.

Рисунок 1.3 – Скріншот з гри Ori and the Blind Forest

Celeste яку продемонстровано на рисунку 1.4 від Matt Makes Games вийшла у 2018 році і отримала багато нагород за свій геймплей, дизайн рівнів і сюжет. Гравець керує Меделін, яка підіймається на гору Селеста, долаючи фізичні та емоційні перешкоди. Геймплейні особлвості це точні стрибки, використання спеціальних механік, таких як dash і grab, для подолання складних рівнів.

«Celeste» була створена лише двома розробниками. Гра отримала нагороди за найкращий інді-проєкт та найкращий саундтрек. Гра включає механіки, які допомагають гравцям покращити свої навички, наприклад, повтори після смерті, які показують помилки гравця.

Кожен стрибок і рух Меделін відчувається чітко й під контролем гравця, що робить подолання складних перешкод неймовірно вигідним. Різноманітні механіки як (Dash та grab) додають новий вимір геймплею, дозволяючи гравцям виконувати складні маневри та досліджувати нові шляхи [7].

Рівні гри кидають виклик гравцям, але вони завжди справедливі й винагороджують за наполегливість. Повторення після смерті, механіка дозволяє гравцям негайно відновитися після невдалої спроби, негайно вивчаючи на своїх помилках.

Рисунок 1.4 – Скріншот з гри Celeste

Dead Cells від Motion Twin є поєднанням платформера і roguelike, вийшла у 2018 році. Гра відзначається своєю складністю, процедурною генерацією рівнів та швидким геймплеєм.

Гравець досліджує замок, бореться з ворогами, збирає зброю і покращення, кожне проходження гри є унікальним завдяки процедурній генерації рівнів. Dead Cells отримала нагороду за найкращий action-геймплей на The Game Awards 2018. Гра має великий вплив на інші roguelike-ігри.

Гра поєднує складність roguelike – це піджанр рольових відеоігор, визначними особливостями якого є випадкове, процедурне створення рівнів з елементами та елменти метроїдванії, що робить її унікальною в жанрі платформерів. Скріншот візуальної складової гри на рисунку 1.5.

Рисунок 1.5 – Скріншот з гри Dead Cells

В аналізі існуючих рішень було розглянуто низку популярних ігор у жанрі платформер, що дозволило ідентифікувати ключові аспекти, які сприяють їх успіху. Серед важливих характеристик виявилися інноваційний геймплей, захоплюючий сюжет, а також якісна графіка та аудіо супровід. Такі елементи як керування гравцем, різноманітність рівнів і чітка візуальна логіка мають вирішальне значення для залучення та утримання уваги гравців.

1.4 Постановка основного завдання на створення гри

«Runestone Rumble» – це головоломка-платформер, яка поєднує в собі елементи класичних платформерів і стратегічних головоломок. Основна концепція гри полягає у використанні рунічних каменів для маніпулювання

навколишнім середовищем з метою вирішення головоломок і подолання перешкод.

У сучасному світі ігрової індустрії платформери залишаються одним з найпопулярніших жанрів, оскільки вони пропонують захоплюючий геймплей і випробування для гравців. «Runestone Rumble» спрямована на те, щоб привнести нові елементи в цей жанр за допомогою механік рунічних каменів, які дозволять гравцям взаємодіяти з ігровим світом новими і цікавими способами.

«Runestone Rumble» пропонує унікальний ігровий досвід, де гравець керує персонажем, що використовує рунічні камені для маніпулювання навколишнім середовищем. Кожен рунічний камінь має свої унікальні властивості і може створювати платформи, переміщувати об'єкти, змінювати гравітацію та багато іншого. Гравцю необхідно творчо використовувати ці можливості для вирішення головоломок і подолання перешкод на своєму шляху.

Гра складається з рівнів, які стають дедалі складнішими, кожен з яких пропонує нові виклики і головоломки. Гравець повинен використовувати рунічні камені для створення платформ, переміщення об'єктів, зміни гравітації та інших маніпуляцій з навколишнім середовищем, щоб досягти мети рівня. Крім того, гравець зіткнеться з різними небезпеками та ворогами, яких потрібно уникати або перемагати.

Основною метою гри є досягнення виходу на кожному рівні, використовуючи рунічні камені для вирішення головоломок і подолання перешкод. Крім основних рівнів, гра може включати додаткові завдання і секретні області, які відкриваються при певних умовах.

Гра «Runestone Rumble» будується навколо механіки рунічних каменів, які дозволяють гравцеві взаємодіяти з ігровим середовищем у різний спосіб. Ось деякі з можливих взаємодій:

Гравець може використовувати рунічні камені, щоб створювати нові платформи, на яких можна стояти або стрибати. Це відкриває доступ до нових областей рівня і допомагає обходити небезпеки.

Деякі рунічні камені дозволяють змінювати напрямок гравітації, що дозволяє гравцеві ходити по стінах або стелі, відкриваючи нові способи дослідження рівня.

На приклад на одному з рівнів гравець зустрічає непрохідну яму. Використовуючи рунічний камінь, він створює платформу, що дозволяє безпечно переправитися через неї.

Головоломки у «Runestone Rumble» розроблені таким чином, щоб пропонувати гравцю різні способи їх вирішення. Це досягається завдяки різноманітним властивостям рунічних каменів і можливості комбінувати їх ефекти. Кожна головоломка може мати кілька можливих рішень. Наприклад, для подолання високої стіни гравець може створити платформу за допомогою одного каменя або змінити гравітацію іншим каменем і пройти по стелі.

Рішення гравця можуть впливати на подальший хід гри. Наприклад, обраний спосіб вирішення головоломки може відкрити нові шляхи або доступ до додаткових ресурсів. Дії гравця можуть динамічно змінювати рівень, наприклад, переміщення об'єктів або активація механізмів, що створює нові виклики і можливості.

Гра повинна мати темну графіку, що відповідає вибраному стилю. Вибір темної графіки обумовлений кількома факторами. По-перше, темна графіка створює атмосферу таємничості і небезпеки, що підходить для головоломки-платформера, де гравець повинен досліджувати і взаємодіяти з навколишнім світом. Така атмосфера допомагає занурити гравця в ігровий світ і робить гру більш захоплюючою.

Темні тони і контрастні кольори дозволяють виділити важливі елементи на екрані, такі як персонажі, рунічні камені і платформи. Це полегшує орієнтацію в ігровому просторі і допомагає гравцю зосередитися на виконанні завдань. Темна графіка також додає візуальної глибини і робить гру візуально привабливою, що може позитивно вплинути на загальне сприйняття гри.

Всі анімації персонажів, включаючи біг, стрибки, смерть, повинні бути плавними та чітко вираженими Плавні і чітко виражені анімації персонажів є

важливою частиною гри, оскільки вони забезпечують зворотний зв'язок для гравця і покращують загальний ігровий досвід. Кожен рух персонажа повинен бути плавним і природним, що допомагає гравцю краще контролювати свої дії і робить гру більш приємною. Наприклад, анімації бігу, стрибків і смерті повинні бути детально опрацьовані, щоб гравець міг чітко розуміти, що відбувається на екрані. Це важливо для досягнення високого рівня занурення в гру і забезпечення комфортного ігрового процесу. Крім того, якісні анімації допомагають створити унікальний стиль гри і роблять персонажів більш виразними і запам'ятовуваними.

1.5 Обґрунтування вибору інструментів та технологій

У розробці сучасних відеоігор використовується безліч потужних інструментів і рушіїв, що дозволяють створювати високоякісний контент. Серед них особливо виділяються три основні рушії Unity, Unreal Engine та Godot. У цьому розділі буде розглянуто вибір рушія для розробки гри «Runestone Rumble».

Unity відзначається своєю доступністю та легкістю у використанні завдяки інтуїтивно зрозумілому інтерфейсу. Це дозволяє розробникам швидко освоїти редактор і негайно розпочати роботу над проєктом. Така характеристика особливо важлива для інді-розробників та невеликих команд, які можуть не мати великого досвіду роботи з ігровими рушіями [15].

Крім того, рушій пропонує широку підтримку документації та навчальних матеріалів. Величезна кількість офіційних та неофіційних ресурсів для навчання, включаючи документацію, відеоуроки, курси та активне ком'юніті, значно спрощує процес навчання та вирішення проблем, що виникають під час розробки [16].

Unity відзначається своєю гнучкістю та розширюваністю, що дозволяє створювати ігри різних жанрів, включаючи платформери, головоломки, екшн,

RPG та багато інших. Ця універсальність дозволяє розробникам легко адаптувати рушій під специфічні потреби кожного проєкту, забезпечуючи високий рівень індивідуалізації та оптимізації.

Однією з ключових переваг рушія є потужні інструменти для анімації. Рушій пропонує розширені можливості для створення анімацій за допомогою Animator, що дозволяє розробляти складні анімаційні переходи та поведінку персонажів. Це сприяє досягненню високого рівня деталізації та реалістичності анімацій, що є критичним для створення захоплюючих ігор з глибоким зануренням.

Окрім цього, Unity підтримує великий набір інструментів та плагінів, які можуть бути додані для розширення функціональності рушія. Це включає інструменти для роботи з фізикою, освітленням, звуковими ефектами, а також інтеграцію зі сторонніми сервісами та платформами. Така підтримка робить Unity надзвичайно гнучким і зручним для використання в різних типах ігрових проєктів.

Unreal Engine – це потужний ігровий рушій, розроблений компанією Epic Games. Він широко використовується для створення високоякісних 3D-ігор завдяки своїй передовій графічній технології та багатому набору інструментів. Unreal Engine підтримує розробку ігор для різних платформ, включаючи ПК, консолі, мобільні пристрої та VR. Переваги Unreal Engine:

- Висока якість графіки. Unreal Engine відомий своїми можливостями для створення реалістичної 3D-графіки завдяки вбудованим шейдерам, системам освітлення та постобробки.
- Блупринти (Blueprints). Візуальне скриптування дозволяє розробникам створювати ігрову логіку без написання коду, що спрощує процес розробки для новачків.
- Інструменти для розробки AAA-ігор. Unreal Engine містить потужні інструменти для роботи з анімацією, фізикою та штучним інтелектом, що дозволяє створювати складні ігрові проєкти.

Unreal Engine має багато переваг, проте ці плюси не є релевантними для розробки «Runestone Rumble» через специфіку проєкту. Тепер розглянемо переваги недоліки:

- Складність освоєння. Unreal Engine є потужним інструментом, але його інтерфейс та робочий процес можуть бути більш складними для новачків та інді-розробників. Unity пропонує більш інтуїтивно зрозумілий інтерфейс, що спрощує процес освоєння та розробки.
- Вимоги до обладнання. Unreal Engine може вимагати більш потужне обладнання для роботи, що може бути проблемою для невеликих команд або індивідуальних розробників з обмеженими ресурсами. Unity є більш легким у використанні і може працювати на менш потужних комп'ютерах.

Хоча Unreal Engine відомий своїми можливостями для створення реалістичної 3D-графіки, це не є основною вимогою для «Runestone Rumble». Гра є платформером з акцентом на головоломки, де більш важливими є чіткість і зрозумілість візуальних елементів, ніж реалістична графіка. Unity надає достатні інструменти для створення високоякісної 2D- та 3D-графіки, яка підходить для стилю гри, без необхідності використовувати потужні шейдери і системи постобробки, які пропонує Unreal Engine.

Візуальне скриптування в Unreal Engine, відоме як блупринти, дійсно спрощує процес розробки для новачків, дозволяючи створювати ігрову логіку без написання коду. Однак, для розробки «Runestone Rumble» пріоритетом є ефективність і гнучкість кодування на C#, яке підтримує Unity.

Unreal Engine містить потужні інструменти для роботи з анімацією, фізикою та штучним інтелектом, що дозволяє створювати складні AAA-проєкти.

Проте «Runestone Rumble» не є AAA-грою і не вимагає такого рівня складності. Гра фокусується на головоломках і платформінгу, де важлива точність та швидкість реалізації, ніж надмірна деталізація і складні механіки фізики. Unity забезпечує достатні інструменти для створення якісної анімації і фізики, необхідні для реалізації задуму «Runestone Rumble», без надмірних ресурсів та функцій, які пропонує Unreal Engine.

Godot – це безкоштовний і відкритий ігровий рушій, який здобуває популярність серед інді-розробників завдяки своїй гнучкості та простоті використання. Рушій підтримує розробку як 2D, так і 3D-ігор, пропонуючи широкий набір інструментів для створення ігрового контенту.

Godot має свої переваги, але ці плюси не є релевантними для розробки «Runestone Rumble» через специфіку проєкту та вимоги до інструментів

Godot є повністю відкритим і доступним для розробників, що дозволяє вносити зміни та налаштовувати рушій під специфічні потреби проєкту, ця перевага не є критичною для розробки. У проєкту уже є чітко визначені вимоги і цілі для гри, які можуть бути повністю реалізовані за допомогою функціональності, що вже надається Unity. Крім того, відкритий вихідний код Godot вимагає додаткових ресурсів для підтримки та налаштування рушія, що може відволікати від основної роботи над грою.

Інтуїтивно зрозумілий інтерфейс та зручні інструменти рушія дійсно роблять його легким для освоєння навіть для новачків. Однак, маючи досвід роботи з Unity, робить перехід на Godot недоцільним. Знання і досвід, набуті при роботі з Unity, дозволяють ефективно використовувати цей рушій без необхідності витрачати час на освоєння нової платформи.

Godot дозволяє створювати ігри для різних платформ, включаючи ПК, мобільні пристрої та веб-браузери. Проте Unity також пропонує потужну підтримку для розробки під різні платформи, включаючи ПК, мобільні пристрої, консолі та VR. Unity має доведену стабільність і більш зрілу підтримку платформ, що забезпечує надійність і оптимізацію гри на різних пристроях. Таким чином, перевага Godot у підтримці платформ не є вирішальною для «Runestone Rumble», оскільки Unity вже забезпечує всі необхідні можливості. Тому вибір Unity є більш доцільним для досягнення цілей проєкту.

Для розробки гри було обрано Visual Studio як основну інтегровану середу розробки (IDE). Це рішення обґрунтоване кількома ключовими перевагами Visual Studio над іншими IDE, такими як JetBrains Rider та інші. Переваги Visual Studio:

- Інтеграція з Unity, Visual Studio надає спеціальні інструменти та розширення для роботи з Unity, що значно спрощує процес розробки. Інтеграція дозволяє легко налагоджувати та тестувати ігрові скрипти, забезпечуючи безшовний робочий процес.
- Розширене автодоповнення Visual Studio пропонує IntelliSense, який забезпечує розширене автодоповнення коду, включаючи методи, властивості, змінні та інші елементи. Це допомагає зменшити кількість помилок і прискорити процес написання коду [12].
- Потужний дебаггер включає потужні інструменти для налагодження коду, що дозволяє легко виявляти і виправляти помилки в коді. Дебаггер підтримує точки зупинки, огляд змінних у реальному часі та аналіз стеку викликів, що є важливим для розробки складних ігрових механік [13].
- Visual Studio має інтуїтивно зрозумілий інтерфейс, який дозволяє легко налаштовувати робоче середовище під власні потреби. Панелі інструментів, вікна та інші елементи можна переміщати та налаштовувати, що підвищує ефективність роботи.
- Широкі можливості розширенн підтримує численні розширення та плагіни, які можуть додатково розширювати функціональність IDE, включаючи спеціалізовані інструменти для Unity, розширене автодоповнення та інші корисні функції [10].
- Інтеграція з системами контролю версій Visual Studio підтримує інтеграцію з різними системами контролю версій, такими як Git та TFS. Це дозволяє ефективно керувати змінами у коді та працювати над проектом у команді [11].

Visual Studio був обраний для розробки «Runestone Rumble» завдяки своїм потужним інструментам для розробки, зручності використання, гнучкості та вбудованим засобам для командної роботи. Інтеграція з Unity, розширене автодоповнення, потужний дебаггер та можливості для спільної роботи роблять Visual Studio ідеальним вибором для цього проекту. Переваги Visual Studio над

іншими IDE, такими як JetBrains Rider і Visual Studio Code, роблять його найбільш підходящим інструментом для ефективної та якісної розробки гри [9].

Aseprite є спеціалізованим інструментом для створення та редагування піксельної графіки. Вибір Aseprite для розробки «Runestone Rumble» обумовлений кількома ключовими перевагами, які роблять його оптимальним для створення високоякісної піксельної графіки.

- Просунуті можливості редагування. Aseprite надає широкий набір інструментів для редагування піксельної графіки, включаючи пензлі, інструменти заливки, лінії, фігури та інші. Це дозволяє створювати детальні і якісні піксельні зображення.
- Підтримує створення анімацій за допомогою шарів та кадрів. Розробники можуть легко створювати плавні анімації персонажів, об'єктів та ефектів, що додає динамічності грі.
- Інтуїтивний і зручний інтерфейс, який спрощує процес створення і редагування графіки. Це дозволяє розробникам швидко освоїти інструмент і зосередитися на творчому процесі.
- Підтримка різних форматів. Експорт графіки в різних форматах, включаючи PNG, GIF, та інші, що дозволяє легко інтегрувати створені ресурси в Unity.
- Aseprite дозволяє налаштовувати інструменти та робочий простір під власні потреби, що підвищує ефективність роботи. Користувачі можуть створювати власні пензлі, налаштовувати панелі інструментів та зберігати шаблони для швидкого доступу.

1.6 Висновок до першого розділу

Перший розділ дипломної роботи було присвячено аналізу предметної області та постановці завдання на розробку гри Runestone Rumble. В ході дослідження було вивчено історію розвитку жанру платформер, його основні елементи, а також сучасний стан платформеру як жанру.

Аналіз розвитку ігор-платформерів показав, що цей ігровий жанр пройшов довгий інтерактивний шлях від простих 2D-карт до сучасних багаторівневих 3D-проектів. Елементи гри-платформера, такі як стрибки, збирання предметів, проходження рівнів та подолання перешкод, залишаються незмінними, але збагачуються новими механіками та вдосконалюються з кожним новим поколінням ігор.

Розглянуті сучасні ігри-платформери, такі як Hollow Knight, Ori and the Blind Forest, Celeste та Dead Cells, продемонстрували інноваційні підходи до ігрового дизайну, захопливі сюжети та високу якість виконання. Ці ігри встановили нові стандарти жанру, збагативши його нововведеннями, які мають вплив на подальший розвиток платформерів. Постановка завдання для створення гри Runestone Rumble включала визначення основної ігрової механіки та використання концепції рунічних каменів для маніпулювання ігровим оточенням. Це дає можливість створити унікальний ігровий досвід, де гравець повинен використовувати логіку та стратегію для вирішення головоломок та подолання перешкод.

Автоматизовані системи обробки економічної інформації, що використовуються у комп'ютерних іграх, також мають свої особливості [33]. Розробка інформаційного забезпечення для цих систем є важливою складовою процесу проектування гри [34]. Та інші схожі джерела [35].

Таким чином, проведений аналіз і постановка завдань створюють міцний фундамент для подальшої розробки гри Runestone Rumble, даючи розуміння основних аспектів жанру платформер і визначаючи напрямок еволюції проекту.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА ТА РОЗРОБКА КОНЦЕПЦІЙ

2.1 Основні елементи гри та принципи роботи

Гра «Runestone Rumble» проектується на рушії Unity, який забезпечує гнучкість і потужність для розробки 2D платформерів. Unity надає багатий набір інструментів і можливостей для створення високоякісних ігор з багатою графікою, складними механіками та інтерактивним ігровим світом [18]. Основні компоненти гри включають ігровий цикл, модулі керування, рендеринг та анімації [23].

У платформерних іграх, таких як «Runestone Rumble», керування персонажем є надзвичайно важливим, оскільки від цього залежить, наскільки комфортно і захоплююче буде грати. Керування персонажем включає кілька компонентів, кожен з яких забезпечує плавний і точний контроль над діями персонажа.

Обробка введення від гравця. Перший крок у керуванні персонажем – це зчитування команд від гравця. У Unity для цього використовується система введення, яка дозволяє зчитувати натискання клавіш, рухи миші та інші введення [24].

Гравець використовує клавіші для переміщення персонажа вліво та вправо. Це забезпечує гнучкий контроль над рухами, що є важливим для точного керування у платформері.

Для стрибка використовується клавіша пробілу. Можливість стрибати є ключовою механікою у платформерних іграх, дозволяючи гравцю долати перешкоди. Основний рух персонажа реалізується шляхом зміни його положення в просторі гри. Це досягається за допомогою компонентів Rigidbody2D і Collider2D, які забезпечують фізичну симуляцію [14].

Використання фізичного компоненту Rigidbody2D дозволяє плавно змінювати швидкість персонажа, що забезпечує природний рух і відчуття контролю [19].

Стрибок реалізується додаванням вертикальної сили до персонажа. Це дозволяє персонажу підніматися в повітря і долати перешкоди. Важливою деталлю є врахування фізики, щоб стрибки були реалістичними та передбачуваними для гравця. Говорячи про стрибок у сучасних платформерах не можливо не згадати про такі параметри як «буфер стрибку» і «час койота», це фундаментально важливі аспекти які помагають керування прижком відчуватися комфортно.

Як працює буфер стрибка гра «пам'ятає» ваш натиск кнопки стрибка протягом дуже короткого часу, наприклад, кількох мілісекунд. Якщо персонаж торкнеться землі або платформи протягом цього часу, гра автоматично виконає стрибок. Це означає, що навіть якщо ви натиснули кнопку стрибка трохи раніше, ніж персонаж досягне землі, стрибок все одно відбудеться.

Спосіб роботи часу койота такий, після того, як персонаж залишає платформу, гра надає дуже короткий період зазвичай кілька мілісекунд, протягом якого стрибок все ще можливий. Це означає, що якщо ви натиснете кнопку стрибка у цей час, персонаж все одно виконає стрибок, навіть якщо вже не стоїть на платформі.

Це дає гравцеві трохи більше часу для реакції. Ви можете виправити свою помилку і стрибнути, навіть якщо трохи запізнилися з натисканням кнопки. Зменшує стрес і фрустрацію, особливо у важких і швидких ігрових ситуаціях.

Гравець може активувати рунічні камені, що дозволяють змінювати властивості навколишнього середовища, наприклад, змінювати гравітацію або активувати платформи.

Механіка рунічних каменів у грі «Runestone Rumble» є ключовою складовою, що робить гру унікальною і захоплюючою. Для того, щоб пояснити потенціал цих механік людям, які не знайомі з комп'ютерними іграми або їх розробкою, розглянемо їх детальніше.

Рунічні камені – це спеціальні об'єкти в грі, які гравець може знайти або отримати під час проходження рівнів. Кожен рунічний камінь має свої унікальні властивості, які можуть змінювати правила гри або взаємодіяти з навколишнім середовищем у різний спосіб. Основні можливості рунічних каменів описано далі. Зміна гравітції та як це працює.

Коли гравець активує рунічний камінь, гравітація у грі змінюється на протилежну. Це означає, що персонаж може «ходити по стелі» або стрибати вгору, як би стрибав вниз.

За цими діями є потенціал для майбутніх змін у грі. Зміна гравітації відкриває нові можливості для вирішення головоломок і подолання перешкод. Гравець може уникати пасток, пересуватися по незвичних маршрутах і досліджувати приховані частини рівнів.

Також серед механік присутня можливість вмикати та вимикати певні види платформ. Інший рунічний камінь дозволяє гравцю вмикати або вимикати певні платформи. Це може означати, що платформи з'являються або зникають за бажанням гравця.

Ця механіка додає до гри елемент головоломки. Гравець повинен обдумати, коли і де активувати або деактивувати платформи, щоб досягти цілі. Це вимагає логічного мислення і планування, роблячи гру більш інтелектуально стимулюючою. Гравці можуть створювати власні шляхи через рівні, відкривати нові маршрути і знаходити унікальні рішення для подолання перешкод. Також перераховані можливості зручно працюють на всіх платформах що є важливим для проекту [20].

Механіка рунічних каменів значно збагачує ігровий процес кількома способами:

- Інтерактивність. Гравці активно взаємодіють з ігровим світом, змінюючи його властивості. Це робить гру більш залучаючою, оскільки гравці можуть безпосередньо впливати на хід подій у грі.

- Стратегія та тактика. Впровадження рунічних каменів додає елементи стратегії та тактики. Гравцям потрібно обдумувати свої дії та планувати їх наперед, що додає глибину ігровому процесу.
- Різноманітність геймплею. Різні рунічні камені додають різноманітність до гри, дозволяючи гравцям використовувати різні підходи для проходження рівнів. Це робить кожне проходження унікальним і захоплюючим.
- Залучення гравців. Унікальні механіки, такі як рунічні камені, утримують інтерес гравців, оскільки вони постійно відкривають нові можливості і шляхи для вирішення проблем. Це стимулює гравців повертатися до гри знову і знову.

Візуальний дизайн є важливою частиною будь-якої комп'ютерної гри, оскільки він визначає, як гра виглядає і відчувається для гравця [21]. У грі «Runestone Rumble» використовується кілька ключових елементів візуального дизайну, таких як освітлення, ефекти частинок та анімації, щоб створити захоплюючий ігровий досвід.

Освітлення відіграє важливу роль у створенні атмосфери гри. У «Runestone Rumble» використовується система Lit з Unity, яка дозволяє створювати реалістичні світлові ефекти. Ось як це працює і чому це важливо:

- Створення атмосфери. Темне освітлення і тіні допомагають створити атмосферу темряви і пустоти, підкреслюючи настрій гри. Це особливо важливо для гри, яка має містичний або похмурий тон.
- Динамічне освітлення. Освітлення в грі динамічно змінюється залежно від дій гравця та оточення. Наприклад, коли гравець рухається по рівню, джерела світла можуть змінювати свою яскравість або напрямок, створюючи реалістичні тіні і світлові ефекти. Це додає грі глибини і робить її більш реалістичною.
- Навігація і увага гравця. Освітлення також допомагає направляти увагу гравця і покращувати навігацію. Яскравіші області можуть вказувати на важливі об'єкти або шляхи, які гравець повинен дослідити, тоді як темніші області можуть приховувати небезпеки або секрети.

Ефекти частинок використовуються для створення візуальних ефектів, таких як дим, вогонь, магічні ефекти від рунічних каменів. Наприклад, коли персонаж використовує рунічний камінь, магічний ефект може проявлятися у вигляді частинок, що розлітаються, підсилюючи відчуття магії і таємниці.

Ефекти роблять гру більш реалістичною. Наприклад, дим і вогонь можуть додати відчуття реальності до сцени, коли персонаж проходить через палаючу область.

Аніматор Unity продемонстрований на рисунку 2.1 використовується для керування анімаціями персонажа.

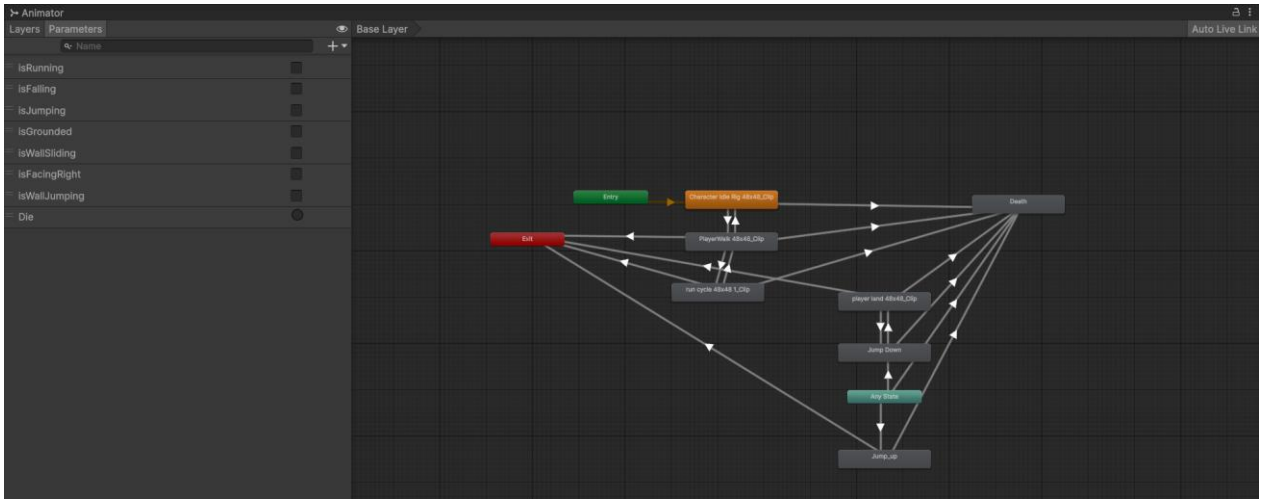

Рисунок 2.1 – Аніматор Unity з реалізаціює анімацій гри гравця

Animator Controller – це об'єкт, який містить всі анімації для конкретного персонажа або об'єкта, а також правила для переходів між цими анімаціями. Animator Controller визначає, які анімації повинні відтворюватися і коли. Кожен рух персонажа, такий як біг, стрибки, смерть, має власний анімаційний кліп. Ці кліпи складаються з серії зображень, які відтворюються послідовно, створюючи ілюзію руху.

Аніматор Unity визначає правила переходу між різними станами анімацій, наприклад, від бігу до стрибка, від стрибка до приземлення. Ці переходи налаштовуються так, щоб рухи персонажа виглядали плавно і природно. Дані

переходи відображаються направленими стрілками від анімації до анімації які являються прямокутниками з заокругленими кутами на рисунку 2.1.

Параметри використовуються для керування анімаціями в реальному часі. Це можуть бути такі параметри, як швидкість бігу, висота стрибка, стан здоров'я персонажа. Параметри визначають, яка анімація буде відтворюватися в даний момент і як вона буде змінюватися залежно від дій гравця та внутрішньої логіки гри. Також в Animator Controller присутні три стани які не відносяться до створюваних анімацій, а являються готовими інструментами для проєктування керування анімаціями.

Entry State початковий стан, з якого починається відтворення анімації. Коли гра починається або персонаж створюється, анімація починається з цього стану. Entry State автоматично переходить до першого стану анімації, визначеного в Animator Controller.

Exit State кінцевий стан, який можна використовувати для виходу з поточної анімації. Зазвичай використовується для завершення певних анімаційних послідовностей або переходу до іншої анімації чи стану.

Any State спеціальний стан, який дозволяє здійснювати переходи до будь-якого іншого стану. Це корисно, коли потрібно миттєво переключитися на іншу анімацію, незалежно від того, яка анімація відтворюється в даний момент. Наприклад, перехід до анімації стрибка або атаки може відбуватися з будь-якого стану.

2.2 Структура рівнів та віртуальна камера

Структура рівнів у грі «Runestone Rumble» є основою ігрового процесу і визначає, як гравці взаємодіють з навколишнім середовищем. Рівні створюються з використанням тайлів, які є основними будівельними блоками для створення ігрових локацій.

Тайл (від англ. «tile» - плитка) – це маленький графічний блок, який використовується для побудови ігрових рівнів. Кожен тайл є частиною більшого зображення або мозаїки, яка створює цілісну картину.

Тайли є основними будівельними блоками для створення рівнів у грі. Вони представляють собою невеликі графічні елементи, які повторюються для формування великих поверхонь, таких як підлога, стіни, платформи та інші структури в грі. Використання тайлів дозволяє розробникам легко і ефективно створювати складні ігрові середовища [22].

Тайли зазвичай мають однаковий розмір, наприклад, 16x16 або 32x32 пікселів, що дозволяє легко розташовувати їх у сітці. У грі можуть використовуватися різні типи тайлів, наприклад, тверді блоки, які створюють підлогу або стіни, і спеціальні тайли, які можуть бути платформами або перешкодами.

Використання тайлів забезпечує узгоджений візуальний стиль у всіх рівнях гри. Це створює відчуття єдності і гармонії в ігровому середовищі. Тайли ефективно використовують пам'ять і ресурси, оскільки однакові графічні елементи повторюються багаторазово. Це дозволяє створювати великі і складні рівні без значного навантаження на апаратне забезпечення.

TileSet (набір тайлів) є колекцією тайлів, що використовуються для створення ігрових рівнів. Уявіть TileSet як набір будівельних блоків, з яких ви можете створювати різноманітні структури в грі. Кожен тайл у наборі має своє ункальне графічне зображення. Один з наборів який буде задіяний у грі «Runestone Rumble» продемоснтровано на рисунку 2.2.

Додатково, тайли можуть бути анімованими, що дозволяє створювати динамічні елементи середовища, такі як мерехтіння вогнів або рух води. Використання анімованих тайлів додає грі візуальної привабливості та динаміки. Крім того, деякі тайли можуть змінювати свої властивості в залежності від дій гравця, наприклад, активуватися або деактивуватися після натискання на важіль або досягнення певної умови в грі.

Ще одним важливим аспектом є використання тайлів для створення багаторівневих структур, де гравець може переміщуватися між різними поверхами або платформами. Це додає глибині ігровому процесу і сприяє створенню більш складних і цікавих рівнів. Збалансоване поєднання різних типів тайлів дозволяє досягти високого рівня деталізації і різноманітності в ігрових локаціях.

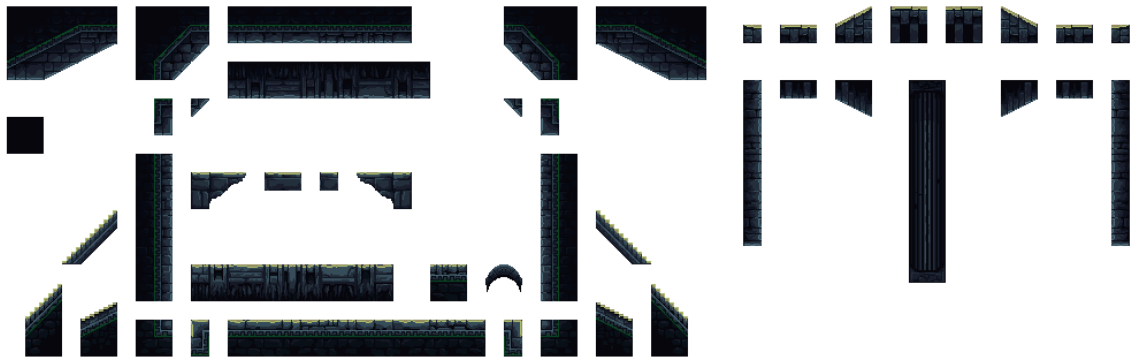

Рисунок 2.2 – TileSet для побудови локацій

TileSet включає всі необхідні графічні елементи для побудови ігрового світу. Це можуть бути різні види підлог, стін, платформ, а також декоративні елементи. Розробники використовують TileSet для швидкого і зручного створення рівнів, оскільки всі елементи вже підготовлені і зібрані в одному місці.

При побудові рівнів важливим аспектом є використання шарів. Шари дозволяють організувати різні елементи рівня в окремі логічні групи, що полегшує керування складними ігровими середовищами та забезпечує узгодженість у відображенні. Фон, або задній фон, створює атмосферу ігрового рівня, додаючи глибину і візуальну привабливість. Це можуть бути статичні зображення або анімовані сцени, наприклад, пейзажі гір, лісу або темних печер, які не впливають на геймплей, але створюють візуальний контекст.

Рівень землі містить основні тайли, які формують підлогу, стіни і платформи, по яких гравець може пересуватися. Цей шар визначає основний

ігровий простір і включає тайли підлоги, які підтримують вагу персонажа, та стіни, що обмежують рух.

Об'єкти переднього плану, або Foreground Layer, можуть частково закривати гравця або інші елементи, додаючи глибину і складність до візуального сприйняття рівня. Це можуть бути рослини або інші декоративні елементи, які знаходяться ближче до камери.

Освітлення використовується для налаштування джерел світла і тіней, що створюють атмосферу і підкреслюють певні частини рівня. Наприклад, смолоскипи або світлові фільтри, які створюють м'які тіні і підсвічують важливі області.

Дизайн рівнів є процесом створення ігрових середовищ, які є цікавими, викликовими і збалансованими для гравця. У грі «Runestone Rumble» рівні проектуються з урахуванням кількох ключових принципів. По-перше, баланс викликів і винагород. Рівні повинні бути достатньо складними, щоб тримати гравця в напрузі, але не настільки важкими, щоб викликати фрустрацію.

Винагороди, такі як нові рунічні камені або покращення, стимулюють гравця продовжувати гру. Наприклад, складна головоломка з використанням рунічних каменів може відкривати доступ до нової області після її вирішення.

По-друге, потік ігрового процесу. Рівні повинні забезпечувати плавний і логічний потік геймплею, де гравець інтуїтивно розуміє, куди йти і що робити далі. Це можуть бути дорожні знаки або візуальні підказки, такі як світлові ефекти, що вказують напрямок руху.

По-третє, різноманітність середовища. Різноманітність у дизайні рівнів підтримує інтерес гравця, пропонуючи нові візуальні і геймплейні елементи на кожному рівні. Наприклад, перехід від підземелля до відкритого лісу може вводити нові види ворогів і пасток.

Нарешті, використання механік гри. Рівні повинні активно використовувати основні ігрові механіки, такі як використання рунічних каменів для вирішення головоломок і подолання перешкод. Наприклад, рівні, де гравець повинен використовувати зміну гравітації для досягнення важкодоступних

місць. Це робить гру цікавою і захоплюючою, забезпечуючи гравцям задоволення від проходження кожного рівня.

Віртуальна камера в платформерних іграх є важливим інструментом, який визначає, як гравці бачать і взаємодіють з ігровим світом. У грі «Runestone Rumble» використовується система Cinemachine з Unity, яка забезпечує плавне і зручне керування камерою.

Віртуальна камера забезпечує гравцю оптимальний огляд ігрового світу, слідуючи за персонажем і адаптуючи зображення залежно від ситуації. Це дуже важливо для створення комфортного і привабливого ігрового досвіду.

Камера автоматично стежить за рухами персонажа, забезпечуючи, щоб він завжди був у центрі або в оптимальному положенні на екрані. Це дозволяє гравцю зосередитися на грі, не турбуючись про те, що персонаж може вийти за межі видимості. Віртуальна камера також може автоматично змінювати масштаб або кут огляду залежно від ситуації у грі. Наприклад, якщо персонаж знаходиться в тісному приміщенні, камера може наблизитися, щоб гравець міг краще бачити деталі оточення.

Cinemachine забезпечує високий рівень контролю і налаштування камери, що робить гру більш зручною і приємною для гравця. Орієнтоване налаштування камери продемоснтровано на рисунку 2.3.

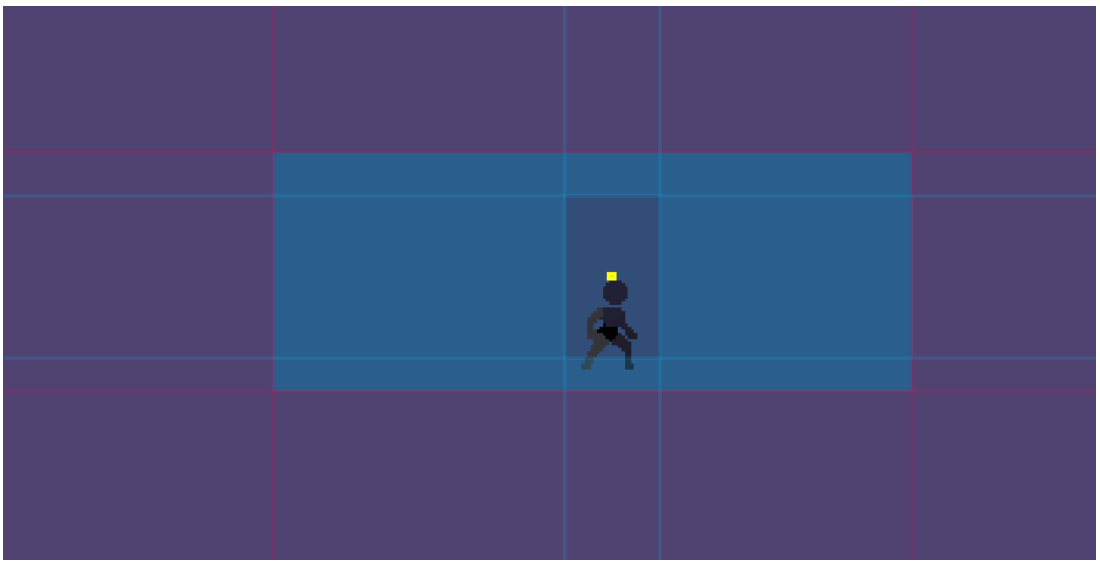

Рисунок 2.3 – Налаштування камери Cinemachine

Розглядаючи рисунок 2.3 віртуальна камера слідує за певною точкою, яка знаходиться над персонажем. Це означає, що камера завжди зосереджена на позиції персонажа, забезпечуючи, що він завжди знаходиться в центрі уваги.

Cinemachine забезпечує плавний рух камери без ривків або затримок у голубій зоні. Це називається інтерполяцією процес плавного переходу камери між різними положеннями.

Це допомагає уникнути різких рухів і ривків, роблячи переміщення камери більш природним і приємним для ока. Коли персонаж рухається, камера плавно слідує за ним, використовуючи інтерполяцію, щоб згладити зміни положення. Це дозволяє камері адаптуватися до швидких рухів персонажа і підтримувати плавний рух навіть у динамічних сценах.

Червону зону яку видно на рисунку 2.3 є індекатором для точки що в ці межі краще не потрапляти, або якщо ж виникла така ігрова ситуція, то швидко поспішити опинитися в рамках голубої зони.

Динамічне поле зору означає, що камера може змінювати масштаб або кут огляду залежно від ігрових умов. Це додає гнучкості і адаптивності, що робить гру більш інтерактивною і цікавою.

Камера також може змінювати кут огляду, щоб показати важливі деталі або забезпечити кращий огляд під час складних маневрів. Наприклад, при стрибках на висоту або під час швидкого руху камера може підлаштовуватися, щоб гравець завжди бачив, куди рухатися далі.

2.3 Планування ігрового світу і систем взаємодії

Дизайн ігрового світу є аспектом розробки гри «Runestone Rumble». Він включає в себе створення візуально привабливих ігрових середовищ, які забезпечують захоплюючий ігровий досвід. Основні елементи дизайну ігрового середовища включають вибір стилю, палітри кольорів, розробку локацій та інтерактивних об'єктів.

Стиль і палітра кольорів визначають загальний вигляд гри і створюють її унікальну атмосферу. У грі використовується темний стиль з яскравими акцентами у вигляді освітлення для підкреслення важливих деталей. Це допомагає створити атмосферу загадковості і напруги, що підходить для гри в жанрі платформер.

Темні відтінки синього, зеленого та фіолетового домінують у кольоровій палітрі, створюючи відчуття темряви і пустоти. Яскраві акценти, такі як червоні або жовті елементи, використовуються для підкреслення важливих об'єктів або областей, на які гравець повинен звернути увагу.

Локації в грі ретельно спроектовані для забезпечення різноманітного ігрового досвіду. Кожна локація має свій унікальний дизайн і виклики, що дозволяє гравцям досліджувати нові місця і вирішувати нові головоломки.

Наприклад, у грі можуть бути підземелля, печери, замки та інші місця, кожне з яких має свої особливості. Підземелля можуть бути наповнені пастками, тоді як печери можуть мати складні головоломки і секретні проходи. Замки можуть містити великі відкриті простори з рухомими платформами і іншими інтерактивними елементами.

Інтерактивні об'єкти є важливою частиною ігрового середовища, оскільки вони додають динаміки і взаємодії до гри. Ці об'єкти включають пастки, місця збереження та рунічні камені, які гравець може використовувати для взаємодії з навколишнім середовищем [17]. Усі інтерактивні об’єкти продемонстровано на рисунку 2.4.

Особливу увагу в дизайні локацій приділено деталям, що створюють атмосферу і занурення. Наприклад, в печерах може бути використано ефект тіні та світла, щоб підкреслити їхню загадковість та небезпеку.

В замках часто використовуються елементи готичної архітектури, що надають їм величності та історичної глибини. Кожна локація має власні унікальні звукові ефекти, які посилюють загальну атмосферу гри, роблячи її більш живою і захоплюючою для гравців.

Рисунок 2.4 – Орієнтовані інтерактивні елементи гри «Runestone Rumble».

На представленому зображенні проднмонстровано інтерактивні об'єкти які плануються починаючи зліва на право проведемо опис їх функціоналу.

Маленький кусок руни гравітації. Цей об'єкт сам по собі нічого не дає, але разом із великою руною гравітації дозволяє гравцю повторно змінити гравітацію. Це важливо, оскільки руна гравітації працює один раз до приземлення на землю. Гравець повинен стратегічно використовувати ці камені для подолання перешкод і досягнення нових зон.

Велика руна гравітації. Використовується для зміни гравітації персонажа, що дозволяє йому ходити по стелі. Це додає новий вимір до геймплею, оскільки гравець повинен враховувати, як зміна гравітації впливає на його можливість рухатися і взаємодіяти з об'єктами.

Руна створення платформ. Дозволяє гравцю створювати платформи для подолання великих розривів або досягнення високих місць. Це дає гравцю більше свободи в пересуванні і вирішенні головоломок.

Чекпоінти або точки збережння зменшують фрустрацію гравця, дозволяючи йому відновлюватися з останнього збереженого місця у випадку поразки. Це забезпечує безперервний прогрес у грі і підтримує мотивацію продовжувати гру.

Шипи один із запланованих різновидів пасток, які моментально вбивають персонажа при дотику. Механіка смерті додає елемент виклику і небезпеки, змушуючи гравця бути уважним і обережним. Коли персонаж тим чи іншим

способом помирає відтворюється екран смерті продемоснтрованому на рисунку 2.5 який надає можливість відротися на останньому місці збереження.

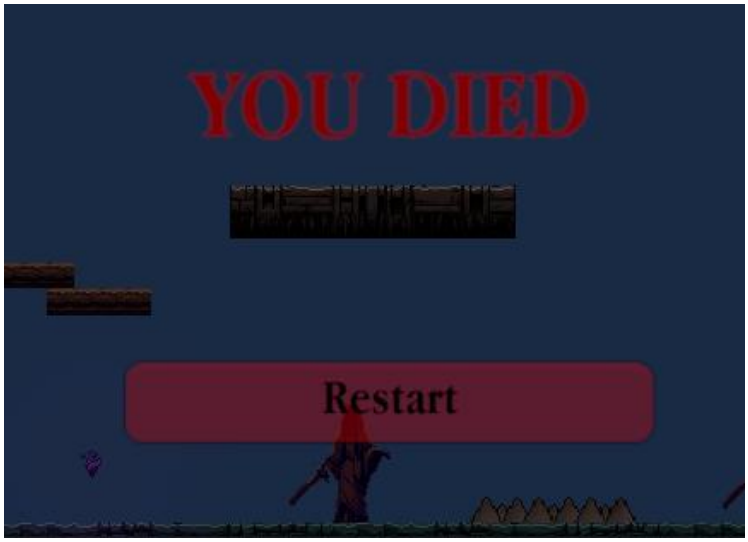

Рисунок 2.5 – Частина екрану смерті

Всі ці елементи разом створюють насичений і захоплюючий ігровий досвід, який тримає гравців у напрузі і стимулює до дослідження ігрового світу. «Runestone Rumble» пропонує унікальну суміш викликів, головоломок і інтерактивності, що робить гру привабливою і незабутньою для гравців.

2.4 Звукове оформлення

Звукове оформлення є частиною розробки гри «Runestone Rumble», оскільки воно значно підсилює атмосферу гри і надає глибини ігровому досвіду. Вибір правильних звукових ефектів і музичного супроводу допомагає створити більш реалістичний і занурюючий світ для гравців.

Звуковий дизайн в іграх включає створення звукових ефектів, музичного супроводу і голосових акторських виступів, які разом працюють для створення повного і занурюючого аудіо-ландшафту. Історія звукового оформлення в іграх почалася з простих звукових ефектів у ранніх аркадних іграх, які згодом

перетворилися на складні музичні композиції та високоякісні звукові ефекти в сучасних іграх.

У ранніх відеоіграх звукове оформлення було обмежено технічними можливостями апаратного забезпечення. Прості звукові ефекти, такі як «біп» та «біп-біп», використовувалися для позначення подій у грі, таких як збирання предметів або стрибки. З часом, розвиток технологій дозволив використовувати багатоканальний звук, цифрові звукові ефекти та музичні треки, що значно покращило аудіовізуальне сприйняття гри.

Сьогодні звукове оформлення є невід'ємною частиною ігрового процесу, що включає створення атмосферного фону, реалістичних звукових ефектів та епічних музичних композицій. В сучасних іграх звукове оформлення допомагає створювати емоційний зв'язок з гравцем, підсилюючи відчуття занурення у ігровий світ.

Для створення звукового оформлення гри «Runestone Rumble» використовувалися різні онлайн-ресурси, які пропонують стокові звуки. Одним з таких ресурсів є сайт Pixabay, де можна знайти широкий вибір безкоштовних звукових ефектів. Цей сайт надає звуки високої якості, які можна використовувати у різних сценах гри, таких як звуки кроків, стрибків, зіткнень та інших дій персонажа.

Ще одним важливим ресурсом для пошуку звукових ефектів є Artlist. Цей сайт пропонує велику колекцію звукових ефектів і музичних треків, які можуть бути використані для створення унікальної атмосфери гри. Використання таких ресурсів дозволяє швидко знаходити необхідні звуки і інтегрувати їх у гру, що значно прискорює процес розробки.

Музичний супровід потрібен у створенні настрою і атмосфери гри. Для музика була знайдена на Unity Asset Store. Цей ресурс пропонує великий вибір музичних треків, спеціально створених для використання у відеоіграх.

Вибір музичних треків для гри базувався на необхідності створити напружену і захоплюючу атмосферу, яка б відповідала стилю і тематиці гри.

Інтеграція звуків і музики у гру Інтеграція звукових ефектів і музики у гру «Runestone Rumble» здійснювалася з використанням інструментів Unity. Це включає налаштування звукових файлів і їх інтеграцію у різні частини гри. Звукові ефекти додаються до анімацій персонажа, взаємодій з об'єктами та інших подій у грі, щоб забезпечити реалістичний і насичений звуковий досвід.

Музичний супровід налаштовується таким чином, щоб він плавно змінювався залежно від ігрових умов. Наприклад, при переході з однієї локації до іншої музика може змінюватися, підлаштовуючись під нову атмосферу і настрій. Це забезпечує безперервний і занурюючий аудіо-досвід, який підтримує загальну атмосферу гри.

2.5 Висновок до другого розділу

Другий розділ роботи був присвячений проектуванню та розробці концепцій гри «Runestone Rumble». В ході дослідження було детально розглянуто архітектуру гри, структуру рівнів, використання віртуальної камери, а також розробку ігрового середовища.

Проектування гри базується на використанні рушія Unity, який надає розробникам широкий набір інструментів для створення високоякісних 2D платформерів. Було розглянуто ключові компоненти гри, такі як ігровий цикл, модулі керування, рендеринг та анімації. Це дозволило створити чітку та злагоджену архітектуру гри, що забезпечує її стабільну роботу та високу продуктивність.

Структура рівнів в грі базується на використанні тайлів, що дозволяє створювати складні ігрові середовища з повторюваних елементів. Це забезпечує ефективне використання ресурсів і створення узгодженого візуального стилю. Важливим аспектом є використання шарів, які дозволяють організувати різні елементи рівня в окремі логічні групи, що полегшує керування складними ігровими середовищами.

Віртуальна камера реалізована за допомогою системи Cinemachine, яка забезпечує плавне і зручне керування камерою. Це дозволяє створити оптимальний огляд ігрового світу, забезпечуючи гравцю комфортний і приємний ігровий досвід. Використання динамічного поля зору і обмеження руху камери допомагають зберегти увагу гравця на важливих елементах гри.

Розробка ігрового середовища включає вибір стилю і палітри кольорів, створення графічних ресурсів та інтеграцію звукових ефектів. У «Runestone Rumble» використовується темний стиль з яскравими акцентами, що допомагає створити атмосферу загадковості і напруги. Інтерактивні об'єкти, такі як рунічні камені, пастки та точки збереження, додають динаміки і взаємодії до гри, роблячи її більш захоплюючою і цікавою для гравців.

Таким чином, другий розділ роботи закладає основи для подальшої розробки гри, забезпечуючи чітке розуміння її архітектури, дизайну рівнів та віртуальної камери. Ці аспекти є критично важливими для створення високоякісної гри, яка забезпечить гравцям захоплюючий ігровий досвід.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА ТА РЕАЛІЗАЦІЯ ІГРОВИХ МЕХАНІК

3.1 Керування персонажем

У цій частині розглянемо, як реалізовано керування персонажем у грі «Runestone Rumble». Основна увага приділяється обробці введення користувача, реалізації буфера стрибка, часу койота та збереженню інерції. Ці механіки забезпечують плавний і відзивчивий контроль над персонажем, що є важливим для ігор жанру платформер. Інспектор компонентів які включені до персонажа продемонстровано на 3.1.

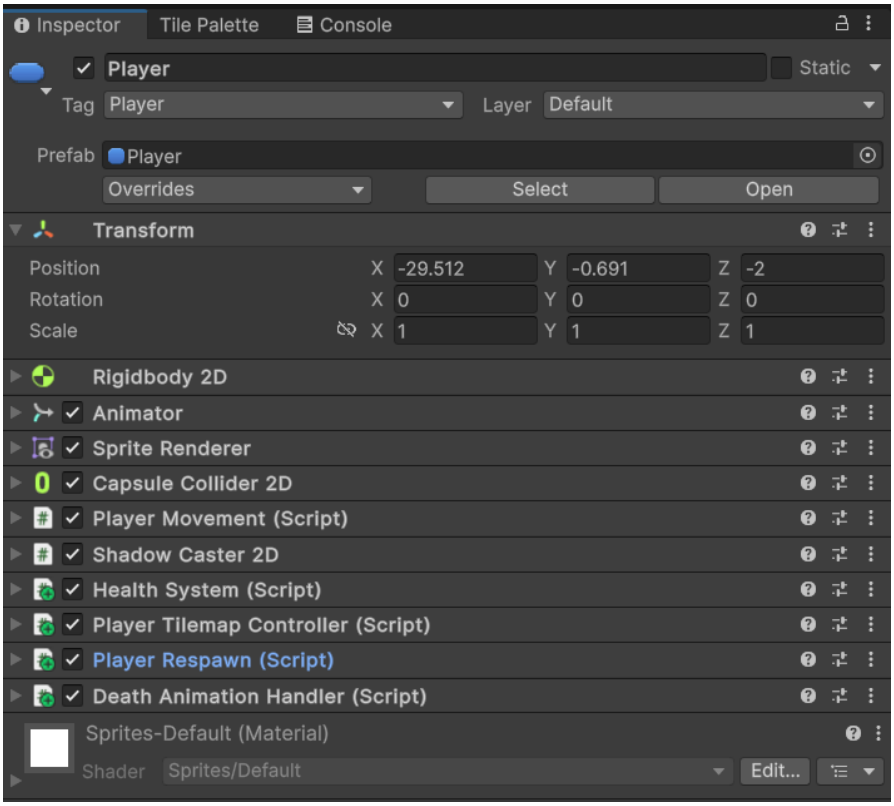

Рисунок 3.1 – Компоненти які включені для персонажа гри

Компонент ‘Rigidbody2D’ було додано до персонажа для забезпечення фізичної симуляції його руху. Цей компонент дозволяє персонажу взаємодіяти з

іншими фізичними об'єктами у грі, а також забезпечує застосування гравітації, сил та зіткнень.

Визначає силу гравітації ‘Gravity Scale’, яка впливає на персонажа. Встановлення цього параметру дозволяє налаштувати, наскільки сильно гравітація буде тягнути персонажа вниз.

Для більшості платформерів зазвичай використовується значення 1, але у випадку перевертання гравітації це значення може змінюватися. Вибір режиму (Continuous) у ‘Collision Detection’ для уникнення пропуску колізій при високих швидкостях. Це важливо для забезпечення точних зіткнень і взаємодій у грі.

Фіксація обертання по осі ‘Z’ для запобігання небажаного обертання персонажа при зіткненнях, що забезпечує стабільне вертикальне положення персонажа під час гри.

Обробка введення користувача є фундаментальною частиною керування персонажем. Вона включає в себе зчитування команд від гравця через клавіатуру. В скрипті ‘PlayerMovement’ розміщеному у додатку А, використовується метод ‘Update’, який зчитує горизонтальні та вертикальні команди, а також визначає, коли гравець натискає або відпускає клавіші для стрибка або зміни гравітації.

В скрипті ‘PlayerMovement’ використовується метод, який зчитує горизонтальні та вертикальні команди (_moveInput.x та _moveInput.y), а також визначає, коли гравець натискає або відпускає клавіші для стрибка (Space, C, J) або зміни гравітації (F).

Ці змінні використовуються для визначення напрямку руху персонажа та виконання відповідних дій. Наприклад, ‘_moveInput.x’ відповідає за горизонтальний рух персонажа, тоді як ‘_moveInput.y’ може використовуватися для вертикальних дій.

Для налаштування параметрів руху та стрибків використовуються змінні та структури даних, що зберігаються в об'єкті ‘PlayerData’. Це включає такі параметри, як максимальна швидкість руху (runMaxSpeed), сила стрибка (jumpForce), час буфера стрибка (jumpInputBufferTime) і час койота (coyoteTime).

Налаштування параметрів може дозволити адаптувати поведінку персонажа для специфічних вимоги геймдизайну. Налаштування будуть дозволяти легко та швидко доналаштовувати поведінку персонажів відповідно до вимог гри. До прикладу, збільшення значення runMaxSpeed дозволить персонажу бігати швидше, а зміна jumpForce вплине на висоту його стрибків.

Компонент ‘CapsuleCollider2D’ було додано для визначення меж колізій персонажа. Цей компонент задає область, в якій персонаж може взаємодіяти з іншими об'єктами в ігровому середовищі. Розмір коллайдера налаштовується так, щоб він відповідав формі персонажа. Це забезпечує точні зіткнення з поверхнями, платформами і перешкодами.

Зміщення коллайдера може бути налаштоване для коригування його позиції відносно центру персонажа. Це дозволяє точно визначити межі колізій для більш природної взаємодії.

Компонент ‘Animator’ використовується для керування анімаціями персонажа. Цей компонент дозволяє змінювати анімаційні стани персонажа, такі як біг, стрибки, падіння, і забезпечує плавні переходи між ними. Параметри вказані на рисунку 3.2.

Рисунок 3.2 – Параметри для відтворення анімацій

Параметри анімації використовуються для керування станами анімацій. Наприклад, isRunning, isJumping, isFalling визначають, в якому стані знаходиться персонаж, і відповідно змінюють анімацію. Різні стани анімації, такі як Idle, Run,

Jump, Fall, задаються в аніматорі і зв'язуються з параметрами анімації. Це забезпечує плавні переходи між різними діями персонажа.

Компонент з рисунку 3.1 ‘Shadow Caster 2D’ стандартний скрипт для розширення Lit у Unity використовується для створення тіней персонажа, що додає глибину і реалістичність до візуального оформлення гри. Дозволяє налаштувати тіні, що падають від персонажа, роблячи гру більш атмосферною.

Скрипт ‘Player Tilemap Controller’ розміщений у додатку Б відповідає за взаємодію персонажа з тайлмепами, включаючи активацію та деактивацію платформ. Логіка, що визначає, як персонаж взаємодіє з платформами та іншими елементами тайлмепа.

Параметри, що представлені на рисунку 3.3, визначають різні аспекти руху і поведінки персонажа у грі «Runestone Rumble». Кожен з цих параметрів впливає на те, як персонаж взаємодіє з оточенням і реагує на команди гравця.

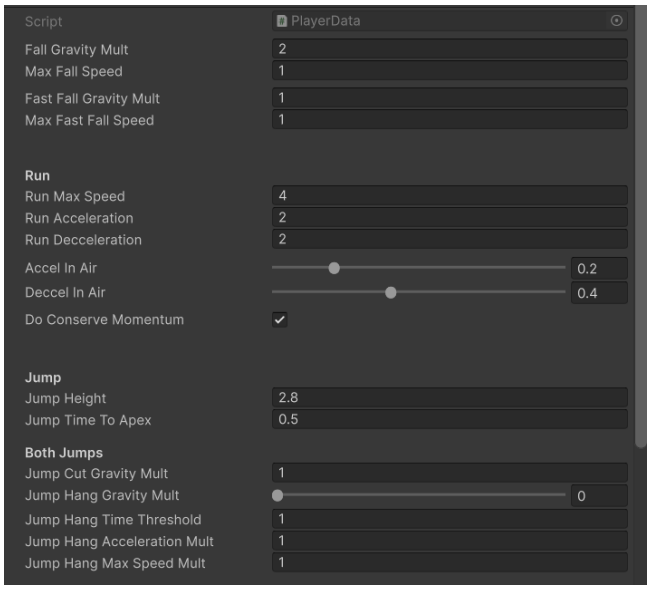

Рисунок 3.3 – Параметри скрипта PlayerMovement

Параметри, що представлені на зображенні, визначають різні аспекти руху і поведінки персонажа у грі «Runestone Rumble».Усі праметри та їх опис подано у таблиці 3.1. Ці параметри дозволяють точно налаштувати поведінку персонажа у грі, забезпечуючи реалістичний і плавний рух. Налаштування кожного

параметра дозволяє адаптувати ігровий процес під специфічні вимоги гри та покращити загальний досвід гравця.

Таблиця 3.1 – Параметри руху персонажа з скрипта ‘Player Movement’

Параметр	**Значення**	**Опис**
Fall Gravity Mult	2	Множник гравітації під час падіння.
Max Fall Speed	1	Максимальна швидкість падіння персонажа.
Fast Fall Gravity Mult	1	Множник гравітації для швидкого падіння.
Max Fast Fall Speed	1	Максимальна швидкість швидкого падіння.
Run Max Speed	4	Максимальна швидкість бігу персонажа.
Run Acceleration	2	Прискорення під час бігу.
Run Deceleration	2	Уповільнення під час бігу.
Accel In Air	0.2	Прискорення в повітрі.
Deccel In Air	0.4	Уповільнення в повітрі.
Do Conserve Momentum	true	Збереження інерції під час руху.
Jump Height	2.8	Висота стрибка персонажа.
Jump Time To Apex	0.5	Час до досягнення вершини стрибка.
Jump Cut Gravity Mult	1	Множник гравітації при обрізанні стрибка.
Jump Hang Gravity Mult	1	Множник гравітації під час зависання в стрибку.
Jump Hang Time Threshold	1	Поріг часу зависання в стрибку.
Jump Hang Acceleration Mult	1	Множник прискорення під час зависання в стрибку.
Jump Hang Max Speed Mult	1	Множник максимальної швидкості під час зависання в стрибку.

Параметри фізики падіння визначають, як персонаж взаємодіє з гравітацією під час падіння. Вони включають множники гравітації для звичайного та швидкого падіння, а також максимальні швидкості для обох типів падіння. Наприклад, значення 2 для параметра ‘Fall Gravity Mult’ означає, що гравітація під час падіння вдвічі сильніша, що прискорює падіння персонажа.

Параметри бігу визначають, як персонаж рухається під час бігу. Вони включають максимальну швидкість бігу, прискорення та уповільнення, а також спеціальні параметри для руху в повітрі. Наприклад, значення 4 для параметра ‘Run Max Speed’ дозволяє персонажу досягати високої швидкості під час бігу, а значення 0.2 для ‘Accel In Air’ визначає, як швидко персонаж прискорюється під час польоту.

Параметри стрибків визначають, як персонаж виконує стрибки. Вони включають висоту стрибка, час до досягнення вершини стрибка, а також спеціальні параметри для керування гравітацією під час стрибків. Наприклад, значення 2.8 для параметра ‘Jump Height’ визначає, наскільки високо персонаж може стрибати, а значення 1 для ‘Jump Hang Gravity Mult’ означає, що гравітація не змінюється під час зависання в стрибку.

3.2 Системи взаємодії з гравцем

Розробка включає створення ігрового середовища, де гравець взаємодіє з різними об'єктами та механіками відповідно. Визначальною частиною взаємодії з системою керування персонажа, використання рунічних каменів, активування точок збереження та взаємодія з небезпеками.

Варто розпочати з раніше загадного у другому розділі маленького куска руни гравітації. Реалізація цієї механіки здійснена через скрипт ‘GravityFlipPickup’, який відповідає за активацію, деактивацію та відновлення об'єкта після його підбирання. Даний скрипт розімщено у додатку В.

Скрипт складається з кількох ключових частин, які забезпечують функціонування механіки малого куска руни гравітації. Компоненти об'єкта включатають наступні:

- RespawnTime час, через який об'єкт відновлюється після підбирання.
- SpriteRenderer компонент, що відповідає за відображення спрайта об'єкта.
- PickupCollider колайдер об'єкта, який визначає його фізичні межі для взаємодії з гравцем.
- LightScript скрипт освітлення об'єкта, який додає візуальні ефекти.

Метод ‘OnTriggerEnter2D’ доданий до об’єкта виконується, коли будь-який інший об'єкт з фізичним колайдером входить у зону навколо нашого об'єкта (так звану зону тригера). Припустимо це працює як сигналізація, яка спрацьовує, коли хтось заходить у кімнату. Якщо цей об'єкт є гравцем (визначається за допомогою мітки ‘Player’), відбуваються дії послідовно вказані в скрипті.

Спочатку буде отримуватися посилання на компонент ‘PlayerMovement’ гравця, що в свою чергу дозволяє скрипту виконувати відповідні дії з основними функціями керування персонажем. Зокрема, метод ‘GrantExtraGravityFlip()’ надає можливість повторного використання зміни гравітації. Це забезпечує гнучкість у геймплеї, дозволяючи гравцю вирішувати головоломки більш ефективно.

Чому було реалізовано саме таким чином. Дана реалізація легко дозволить гравцю повторно змінювати гравітацію, що з великою ймовірнісю додає стратегії до геймплею та дозволяє гравцю вирішувати головоломки більш творчо.

Візьмемо такий приклад, гравець може міняти гравітацію кілька разів для заради досягнення важкодоступних місць. Компонент ‘RespawnPickup()’ який забезпечує автоматичне відновлення об'єкта через певний час, що зберігає динамічність у гри і дозволить повторне використання механіки. В свою чергу це зробить рівень цікавішим оскільки гравець може знову використовувати об'єкт через деякий час.

Використання методу для активації/деактивації компонентів може позволити досить ефективно керувати ресурсами, відключати деякі непотрібні компоненти, коли об'єкт не активний. Такий спосіб допоможе частков знизити навантаження на систему та збільшить продуктивність гри.

Тригерні зони і теги використовуються для взаємодії з гравцем забезпечуючи простір і зрозумілу реалізацію, яку легко налаштувати і розширити при необхідності.

При розробці наступних магічних рун, було задіяно схожі методи, різниця полягає лише у способах реалізації їх головних принципів роботи. Для руни гравітації це використання методів зі скрипта ‘PlayerMovement’ щоб напряму впливати на гравітацію гравця.

Прогамний код скрипту ‘PlayerRespawn’ розміщеного у додатку Е має декілька ключових частин, вони забезпечують функціонування механіки відродження. Ця механіка буде забезпечувати гравцю можливість продовжувати гру з певної точки після смерті. Компоненти та їх опис подано далі:

- InitialPosition початкова позиція гравця на початку гри;
- RespawnPosition позиція відродження гравця після смерті;
- ActiveCheckpointID ідентифікатор активного чекпоінта;
- Checkpoints словник, що зберігає позиції чекпоінтів;
- GravityStates словник, що зберігає стани гравітації для чекпоінта;
- HasCheckpoint прапорець, що вказує, чи є активний чекпоінт.

Відновлення стану гравця та його позиції після смерті забезпечує безперервність геймплею. Гравець може зосередитися над потенційними вирішеннями головоломок та подоланні бар’єрів без необхідності починати з самого початку.

Використання словників при зберігання позицій і станів гравітації чекпоінтів позволить керувати ресурсом гри, зберігаючи важливу інформацію про прогрес гравця. Чітко встановлених способів для відновлення чекпоінтів робить реалізацію простою для розуміння і налаштування. Відповідно позволяє зручно маштабувати проєкт.

3.3 Освітлення та камера

Створення атмосфери у грі зроблено за допомогою реалістичного методу освітлення у Unity. Компоненти які входять у комплекс освітлення юніті:

- Directional Light використовується для створення загального освітлення сцени. Даний компонент імітує світло, це дозволяє викликати відчуття природного вигляду освітлення.
- Point Light використовується для освітлення конкретних областей сцени, таких як смолоскипи або ліхтарі. Це створює ефект локальних джерел світла.
- Spot Light використовується для направленого освітлення, що підсвічує певні деталі або області сцени, додаючи акцентні світлові ефекти.

Ці компоненти реалізовані дуже просто і ясно, достатньо розмістити у редакторі створити необхідні парметри і налалі копіювати цей компонент для відтворення ідентичного освітлення.

Насиченість компонента Point Light встановлена на низьке значення для отримання ніжного загального освітлення сцени. На рисунку 3.4 продемонстровано як це виглядає у редакторі.

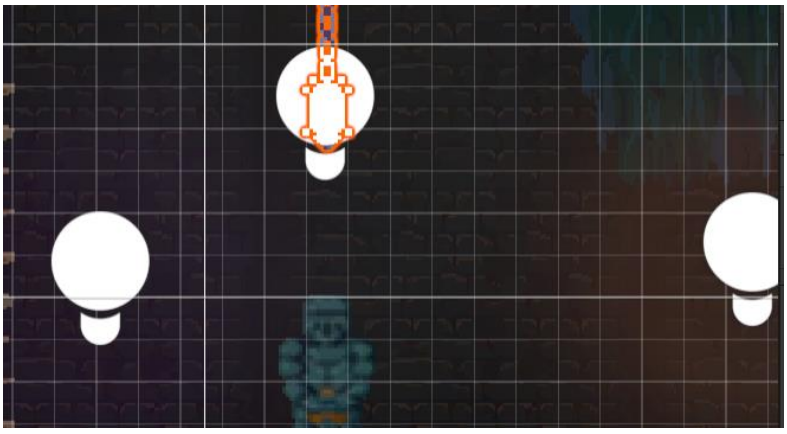

Рисунок 3.4 – Розташування елементу освітлення Point Light у редакторі Unity

За даним принципом були розміщені інші види освітлення такий як наприклад Point Light, як зрозуміло це точкове світло, його було націлено на зони щоб привернути увагу гравця.

Встановлені у стратегічно важливих місцях, таких як коридори, підземелля та біля об'єктів, що потребують акцентного освітлення. Для ліхтарів використано жовті та червоні відтінки для створення контрасту з холодним загальним освітленням.

У грі була використана система Cinemachine з Unity для підтримки оптимального огляду ігрового світу та керування камерою. Ключові компоненти, налаштовані для цієї функції, включають CinemachineVirtualCamera та CinemachineBrain. Налаштування виконанні над камерою продемоснтровано на рисунку 3.5.

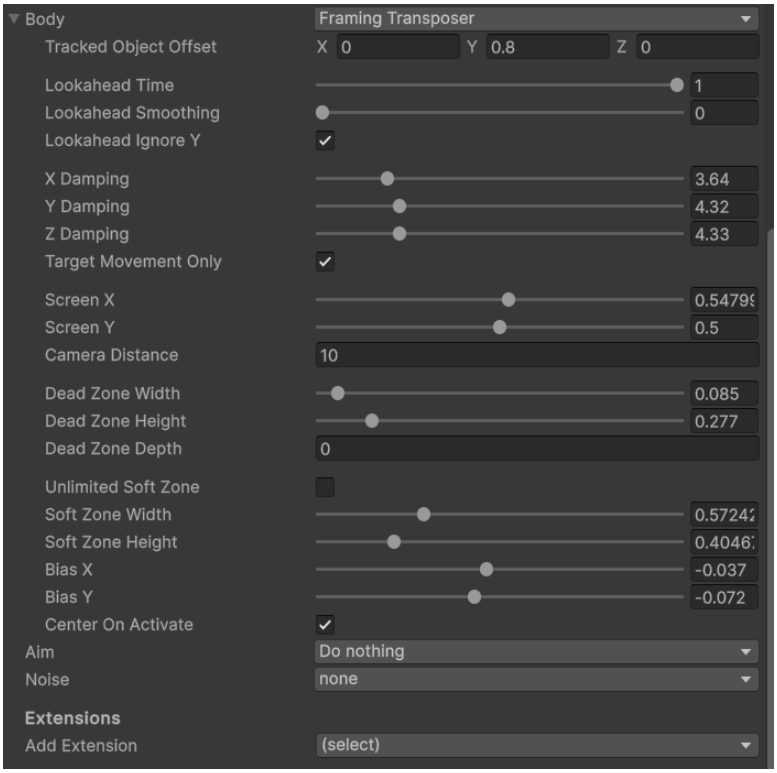

Рисунок 3.5 – Налаштування CinemachineBrain у редакторі Unity

CinemachineVirtualCamera було налаштовано таким чином, щоб камера завжди слідкувала за головним персонажем гри. Це було зроблено, вказавши камері, що завжди триматиме головного персонажа в центрі уваги. Це знову ж таки відрегулювало зсуви параметрів по X, Y.

Відстань та поле зору камери були налаштовані для забезпечення оптимального огляду ігрового світу. Відстань камери встановлена на 10, що

впливає на масштаб сцени, забезпечуючи чіткий огляд найближчих об'єктів та просторів.

Крім того, були встановлені обмеження, щоб камера не виходила за межі рівня, забезпечуючи гравцю постійний огляд тільки допустимих частин ігрового світу. Компонент ‘BrainSwitch’ був налаштований для плавного перемикання між різними віртуальними камерами залежно від ігрових ситуацій. Це дозволяє плавними переходами та не дозволяє пошатнуватися камері під час руху.

Параметр ‘Lookahead Time’, встановлений на 1, задає глибину, на яку камера дивитимететься в напрямку руху персонажа. Це симулює е. Налаштування Lookahead Ignore Y дозволить ігнорувати зміни по Y, коли прогнозується рух кам.

Параметри демпфування налаштовані таким чином що визначає, наскільки агресивно камера буде просуватися за гравцем по осях X, Y та Z. Контрольний параметр рух мети.

Значення screen X встановлено на 0.5479, що означає, що персонаж буде трохи праворуч від центру екрану. Значення screen Y встановлено на 0.5, що означає, що персонаж буде знаходитися приблизно в центрі вертикально. Параметри ‘Dead Zone Width’ і ‘Dead Zone’. Це уникне обертання камери при незначних рухах персонажа. Параметри ширини ‘Soft Zone’ і висоти, встановлені на 0.5724 і 0.4046 відповідно, визначають зону, всередині якої камера буде плавно слідувати за персонажем, забезпечуючи плавне переміщення. Включення параметра ‘Center On Activate’ дає можливість автоматично центрувати камеру на гравцях.

3.4 Висновок до третього розділу

Розділ присвячений практичній реалізації ігрового світу та механік гри «Runestone Rumble». В процесі розробки було застосовано сучасні методи та інструменти, які дозволили створити гнучку та інтерактивну ігрову середу. Для

контролю камерою була використана система Cinemachine з Unity, завдяки чому гравець може легко та плавно управляти камерою. Створення камери дозволяє гравцеві ніколи не втрачати найкращий огляд ігрового світу, що надає гравцю зручність та приємність гри. Використання параметрів, таких як час пошуку, вирівнювання по часу, демпфування X та Y та інших, дозволяє камері автоматично адаптуватися до рухів персонажа, забезпечуючи стабільний та комфортний огляд.

Скрипти взаємодії, такі як ‘PlayerRespawn’, ‘GravityFlipPickup’ та інші, забезпечують багатогранні можливості для гравця у взаємодії з ігровим світом. Вони дозволяють змінювати гравітацію, активувати платформи, зберігати прогрес у грі через чекпоінти та багато іншого. Ці механіки додають глибини та різноманіття до геймплею, роблячи його більш захоплюючим.

Реалізація систем освітлення, камери та взаємодії з гравцем у грі «Runestone Rumble» продемонструвала важливість інтеграції сучасних технологій та підходів до створення ігрових механік. Це дозволило створити не лише технічно досконалий продукт, але й захоплюючий ігровий досвід, який сприяє більш глибокому залученню гравців. Використання Cinemachine та ретельно налаштованих параметрів камери забезпечило високий рівень комфорту та зручності під час гри, а системи взаємодії з гравцем розширили можливості ігрового процесу, зробивши його більш інтуїтивним та цікавим.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні аспекти організації робочого місця розробника комп'ютерних ігор

Ергономіка робочого місця є критично важливим аспектом у забезпеченні здоров'я та продуктивності розробників комп'ютерних ігор. Розробка ігор – це інтенсивний процес, що вимагає тривалого перебування перед комп'ютером, високої концентрації уваги та виконання дрібних маніпуляцій з обладнанням. Правильне проектування робочого простору допомагає запобігти фізичному дискомфорту, втомі та можливим професійним захворюванням, які можуть негативно вплинути на здоров'я розробника та якість його роботи [25].

Піднімати питання ергономіки робочого місця надзвичайно важливо, оскільки воно безпосередньо стосується якості життя працівників. Невідповідні умови праці можуть призвести до розвитку хронічних захворювань, таких як тунельний синдром, остеохондроз, проблеми зі зором, а також до загальної втоми та зниження продуктивності. З іншого боку, оптимально організоване робоче місце може значно підвищити ефективність роботи, сприяти кращому самопочуттю працівників та зменшити ризики для їхнього здоров'я [26].

Забезпечення комфортних умов праці включає в себе кілька ключових аспектів. Зручність розташування обладнання дозволяє уникнути непотрібних напружень та зберігати правильну поставу під час роботи. Налаштування робочого столу і стільця відповідно до анатомічних потреб розробника допомагає запобігти фізичним навантаженням на спину, шию та руки. Освітлення робочого місця має бути налаштоване таким чином, щоб мінімізувати навантаження на очі та забезпечити комфортні умови для роботи в будь-який час доби.

У сучасному світі, де технології постійно розвиваються, а вимоги до продуктивності працівників зростають, питання ергономіки робочого місця стає

ще більш актуальним. Відповідне проектування робочого простору є важливою складовою створення здорового та ефективного робочого середовища для розробників комп'ютерних ігор. Врахування ергономічних аспектів дозволяє не тільки підвищити продуктивність, але й забезпечити довготривале збереження здоров'я працівників, що є важливим як для окремих працівників, так і для організацій в цілому.

Одним із ключових факторів забезпечення ергономічності робочого місця є правильне розташування обладнання. Монітор повинен бути розташований на рівні очей або трохи нижче, щоб запобігти напрузі м'язів шиї та плечей. Відстань до монітора повинна становити приблизно 50-70 см, щоб уникнути напруження очей. Клавіатура та миша повинні бути розташовані на зручній висоті, щоб руки користувача знаходилися у природному положенні. Це допоможе уникнути напруги м'язів рук і запобігти розвитку тунельного синдрому.

Правильне налаштування робочого столу і стільця також є важливим для забезпечення комфортних умов праці. Стілець повинен мати регульовану висоту, підлокітники та підтримку для попереку. Висота стільця повинна дозволяти ногам стояти рівно на підлозі або на підставці, забезпечуючи підтримку для колін під кутом 90°. Робочий стіл повинен бути достатньо великим, щоб розмістити все необхідне обладнання, і мати регульовану висоту для забезпечення зручної роботи. Використання столів з можливістю регулювання висоти допомагає чергувати роботу сидячи та стоячи, що сприяє зниженню ризиків для здоров'я, пов'язаних з тривалим сидінням.

Освітлення робочого місця є важливим фактором, що впливає на зоровий комфорт і продуктивність. Робоче місце повинно бути добре освітлене, бажано природним світлом. Штучне освітлення має бути рівномірним і достатньо яскравим, щоб уникнути напруги очей. Використання настільних ламп з регульованою інтенсивністю світла допомагає забезпечити оптимальні умови для роботи. Також важливо уникати відблисків на екрані монітора. Для цього монітор слід розташовувати так, щоб на нього не падало пряме світло від вікон

або освітлювальних приладів. Використання антиблікових екранів також може бути ефективним рішенням для зменшення навантаження на очі.

Поряд із фізичними аспектами, важливим є забезпечення психологічного комфорту на робочому місці. Це включає створення приємної атмосфери, можливість перерв та організацію робочого часу, що дозволяє уникнути стресу і перевтоми. Важливо забезпечити можливість регулярних перерв для відпочинку очей та розминки, що допомагає підтримувати високу продуктивність і здоров'я розробників.

Забезпечення ергономічних умов на робочому місці розробника гри є важливим аспектом охорони праці. Правильне розташування обладнання, налаштування робочого столу і стільця, освітлення та психологічний комфорт є ключовими факторами, що впливають на здоров'я та продуктивність працівників. Врахування цих аспектів дозволяє знизити ризики професійних захворювань, підвищити комфорт і ефективність роботи, а також сприяє загальному добробуту розробників [27].

Крім основних аспектів ергономіки, варто враховувати також нормативні вимоги, які регулюють робочі місця для розробників програмного забезпечення. В Україні існує низка нормативних документів, що визначають вимоги до ергономічних умов праці. Наприклад, ДСТУ 8604:2015 "Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги" встановлює основні вимоги до організації робочих місць, зокрема щодо висоти робочого столу, крісла, розташування обладнання та освітлення [25].

Важливою складовою забезпечення ергономічних умов праці є дотримання гігієнічних норм. Робоче місце повинно відповідати гігієнічним вимогам щодо мікроклімату, рівня шуму та освітлення. Наприклад, температура повітря в робочому приміщенні повинна бути в межах 18-22 °C, рівень відносної вологості – 40-60%, а рівень шуму – не перевищувати 50 дБ. Освітленість робочої поверхні повинна бути не менше 500 люкс, що забезпечує комфортні умови для зорової роботи.

Окрім цього, варто звернути увагу на організацію робочого часу. Робота за комп'ютером повинна бути організована таким чином, щоб уникати тривалого статичного навантаження та напруження очей. Рекомендується робити перерви кожні 1-2 години роботи, під час яких можна виконувати прості вправи для очей, шиї, рук та спини. Це допомагає зняти напругу та запобігти розвитку професійних захворювань.

Дотримання цих вимог сприяє створенню безпечних та комфортних умов праці, що є важливим для забезпечення продуктивності та здоров'я працівників.

4.2 Вимоги безпеки до робочих місць для виконання робіт

Вимоги безпеки до робочих місць для виконання робіт охоплюють широкий спектр аспектів, які мають бути враховані для забезпечення безпеки та здоров'я працівників. Вони включають фізичні, хімічні, біологічні та психофізіологічні фактори, що можуть впливати на умови праці та загальний стан працівника.

Фізичні фактори включають освітлення, мікроклімат, шум, вібрацію та радіаційне випромінювання. Робоче місце повинно бути обладнане таким чином, щоб забезпечити оптимальний рівень освітлення, відповідний до характеру виконуваної роботи. Згідно з нормативними документами, освітленість робочих місць повинна відповідати вимогам ДСТУ EN 12464-1:2014. Крім того, освітлення повинно бути рівномірним, щоб уникнути тіней та відблисків, що можуть спричиняти зорове напруження [28].

Мікроклімат приміщення повинен підтримуватись на рівні, що забезпечує комфортні умови для праці. Наприклад, температура повітря в робочому приміщенні повинна бути в межах 18-22 °C, рівень відносної вологості – 40-60%, а швидкість руху повітря – не більше 0,2 м/с. Важливо також забезпечити регулярну вентиляцію приміщення для підтримки свіжого повітря.

Рівень шуму та вібрації повинен бути знижений до мінімуму, щоб уникнути негативного впливу на здоров'я працівників. Допустимі рівні шуму встановлені ДСН 3.3.6.037-99 [29], а вібрації – ДСН 3.3.6.039-99 [30]. Використання шумозахисних конструкцій та віброізоляції допомагає знизити рівень цих факторів.

У випадку роботи з джерелами радіації, повинні бути вжиті заходи для захисту працівників від її шкідливого впливу, зокрема використання екранування та дотримання безпечних відстаней.

Хімічні фактори включають наявність шкідливих речовин у повітрі робочої зони. Для запобігання негативному впливу хімічних речовин на здоров'я працівників, робоче місце повинно бути обладнане ефективними системами вентиляції та фільтрації повітря. Всі хімічні речовини повинні зберігатися у спеціальних контейнерах та відповідно маркуватися. Наприклад, кожна ємність з хімічними речовинами повинна мати етикетку з інформацією про небезпеку та заходи безпеки.

Працівники повинні бути забезпечені засобами індивідуального захисту, такими як маски, рукавички та захисні окуляри. Крім того, слід проводити регулярні інструктажі щодо безпечного поводження з хімічними речовинами та надання першої допомоги у разі контакту з ними.

Біологічні фактори включають можливість контакту з біологічно активними матеріалами або організмами, що можуть становити загрозу для здоров'я. Для захисту працівників від біологічних ризиків, робоче місце повинно бути обладнане відповідними засобами захисту, такими як біозахисні костюми, рукавички та маски. Працівники мають бути проінформовані про потенційні ризики та способи їх уникнення, а також повинні знати правила гігієни та санітарії.

Психофізіологічні фактори включають психічне та емоційне навантаження, яке може впливати на працездатність та загальний стан працівника. Для мінімізації негативного впливу цих факторів, робоче місце повинно бути організоване таким чином, щоб забезпечити комфортні умови

праці. Це включає оптимальне планування робочого часу, наявність місць для відпочинку, а також проведення релаксаційних заходів, таких як фізичні вправи або перерви для розслаблення.

Важливо забезпечити регулярні інструктажі з охорони праці та навчання працівників правилам безпеки на робочому місці. Це допомагає підвищити рівень обізнаності працівників щодо можливих ризиків та методів їх запобігання. Відповідно до НПАОП 0.00-4.12-05, кожен працівник повинен проходити первинний, повторний, позаплановий та цільовий інструктажі з охорони праці [31].

Крім того, важливим аспектом є проведення регулярних медичних оглядів для працівників, що дозволяє вчасно виявити та запобігти можливим захворюванням, пов'язаним з умовами праці. Робочі місця повинні бути обладнані необхідними засобами для надання першої медичної допомоги у разі нещасного випадку.

Дотримання цих вимог забезпечує безпечні умови праці, знижує ризик травматизму та професійних захворювань, а також сприяє загальному поліпшенню умов праці та підвищенню продуктивності працівників

4.3 Висновки до четвертого розділу

У цьому розділі ми детально розглянули ключові аспекти забезпечення безпеки та ергономіки робочих місць для виконання робіт у сфері розробки комп'ютерних ігор. Встановлено, що ергономічні аспекти організації робочого місця мають вирішальне значення для підтримки здоров'я та продуктивності працівників. Забезпечення правильного розташування обладнання, налаштування робочого столу і стільця, а також відповідного освітлення сприяє зниженню фізичного напруження та запобіганню професійним захворюванням.

Вимоги безпеки до робочих місць включають контроль фізичних, хімічних, біологічних та психофізіологічних факторів. Зокрема, оптимальне

освітлення, відповідний мікроклімат, зниження рівня шуму та вібрації є важливими елементами забезпечення комфортних умов праці. Дотримання нормативних вимог щодо зберігання та використання хімічних речовин, а також забезпечення працівників засобами індивідуального захисту допомагає уникнути шкідливого впливу на здоров'я.

Значна увага приділена психофізіологічним аспектам, зокрема організації робочого часу та забезпеченню психологічного комфорту на робочому місці. Регулярні інструктажі та навчання з охорони праці підвищують обізнаність працівників щодо можливих ризиків та методів їх запобігання.

Проведення регулярних медичних оглядів є невід'ємною частиною забезпечення безпечних умов праці, дозволяючи своєчасно виявляти та запобігати можливим захворюванням. Оснащення робочих місць засобами для надання першої медичної допомоги у разі нещасних випадків додатково підвищує рівень безпеки.

Дотримання комплексних вимог безпеки та ергономіки на робочих місцях розробників комп'ютерних ігор є критично важливим для підтримки їхнього здоров'я та продуктивності. Інтеграція цих вимог у повсякденну практику сприяє створенню здорового та безпечного робочого середовища, що, в свою чергу, позитивно впливає на ефективність роботи та загальний добробут працівників.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи досягнуто кілька ключових та важливих результатів, що свідчать про успішне виконання поставлених на роботу завдань. В першу чергу, було здійснено аналіз предметної області по платформерних іграх, що в свою чергу дозволило визначити основні тенденції та характеристики жанру.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

- Подано огляд основних етапів розвитку жанру платформерів, визначено їх ключові характеристики.
- Розглянуто сучасні тенденції та приклади успішних платформерних ігор, таких як Hollow Knight, Ori and the Blind Forest, Celeste та Dead Cells.
- Висвітлено основні механіки та елементи, що визначають платформерні ігри, включаючи стрибки, збирання предметів, та структуру рівнів.
- Проаналізовано вплив рунічних каменів на геймплей та їхні можливості для інтерактивної взаємодії з ігровим середовищем.

В другому розділі кваліфікаційної роботи:

- Досліджено особливості використання ігрового рушія Unity для розробки платформерних ігор.
- Обґрунтовано вибір інструментів і технологій, таких як Visual Studio для кодування та Aseprite для створення піксельної графіки.
- Сформовано концепцію гри «Runestone Rumble» з акцентом на унікальні механіки рунічних каменів, що додають стратегії та тактики в ігровий процес.

В третьому розділі кваліфікаційної роботи:

- Розроблено основні компоненти гри, включаючи систему керування персонажем, інтерактивні механіки рунічних каменів та структуру рівнів.
- Запропоновано візуальний дизайн гри з використанням освітлення, ефектів частинок та анімацій для створення захоплюючого ігрового досвіду.
- Спроєктовано рівні з урахуванням використання тайлів та шарів для створення багатошарового ігрового середовища.

- Протестовано ключові елементи гри, забезпечуючи їхню коректну роботу та узгодженість з загальною концепцією.

У розділі «Безпека життєдіяльності, основи охорони праці»:

- Висвітлено аспекти забезпечення безпеки та охорони праці при розробці та тестуванні програмного забезпечення, а також при використанні комп’ютерних ігор.
- Розглянуто заходи щодо зменшення негативного впливу на здоров’я розробників та користувачів програмного продукту.

Розробка комп'ютерної гри «Runestone Rumble» у жанрі платформер за допомогою рушія Unity дозволила створити унікальний продукт з цікавими ігровими механіками та високоякісним візуальним дизайном. Проведені дослідження та реалізація проекту підтвердили можливості сучасних інструментів для розробки ігор, забезпечуючи ефективний та захоплюючий ігровий досвід для кінцевого користувача.

ПЕРЕЛІК ДЖЕРЕЛ

1. Kucklich, Julian. "Electronic Platform Games: From Donkey Kong to Super Mario Odyssey." Encyclopedia of Video Game History, 2022. 375 p.

2. Steven L. Kent. "The Ultimate History of Video Games, Volume Two: From Pong to Pokémon and Beyond...The Story Behind the Craze That Touched Our Lives and Changed the World." Crown, 2021. 592p.

3. Donovan T. Replay: the history of video games. East Sussex, England: Yellow Ant, 2019. 501 p.

4. ESRB ratings guides, categories, content descriptors. ESRB Ratings. URL: https://www.esrb.org/ratings-guide/ (дата звернення 10.06.2024).

5. Усі жанри комп'ютерних ігор. UA PLAY. URL: https://uaplay.com.ua/usizhanry-pc-ihor/ (дата звернення 10.06.2024).

6. Jesse Schell. "The Art of Game Design: A Book of Lenses, Third Edition." CRC Press, 2019. 600 p.

7. Rachel Kowert, Thorsten Quandt. "The Video Game Debate 2: Revisiting the Physical, Social, and Psychological Effects of Video Games." Routledge, 2020. 144 p.

8. Технічна документація Microsoft. Microsoft; вебсайт. URL: https://docs.microsoft.com/en-us/ (дата звернення 10.06.2024).

9. Michael Halvorson. "Microsoft Visual Basic 2019: Step by Step." Microsoft Press, 2019. 624p.

10. Build. Test. Deploy. Microsoft. URL: https://dotnet.microsoft.com/en-us/ (дата звернення 10.06.2024).

11. Стаття релізу змін у версії Visual Studio 2022. Microsoft: вебсайт. URL: https://docs.microsoft.com/en-us/visualstudio/releases/2022/release-notes-preview (дата звернення 10.06.2024).

12. Стаття про оновлення версій C#. Microsoft: вебсайт. URL: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-10 (дата звернення 10.06.2024).

13. Докуметнація мови C#. Microsoft: вебсайт. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення 10.06.2024).

14. Joe Hocking. "Unity in Action, Third Edition: Multiplatform game development in C#." Manning Publications, 2022. 416 p.

15. Alan Thorn. "Unity Game Optimization: Enhance and extend the performance of your Unity games." Packt Publishing, 2021. 363p.

16. Tom Miller, Dean Johnson. "Learning Unity 2021: Build, manage, and deploy games across multiple platforms." 203 p.

17. Thorn A. "Pro Unity Game Development with C#: Leverage the Power of Unity and C# to Build Amazing Games." Apress, 2021. 380 p.

18. Корнілов А.В. UNITY. Повне керівництво. Наука і техніка. 2021. 209 с

19. Buttfield-Addison P., Manning J., Nugent T. Unity game development cookbook: essentials for every game. O'Reilly Media, 2019. 406 p.

20. Хокінг Д. Unity в дії: мультиплатформна розробка на C#. СПб : Луцьк, 2016. 336 p.

21. Джон Меннінг, Перріс Батфілд-Еддісон. Unity для розробника. Наука і техніка, 2019. 352 p.

22. Smith M., Ferns S. Unity 2021 cookbook: over 160 recipes to take your unity game development skills to the next level. Packt Publishing, Limited, 2021.

23. Thorn A. "Unity Animation Essentials." Packt Publishing, 2023. 196 p.

24. Thorn A. Unity 2018 By Example. Packt Publishing, 2015, 402 p.

25. ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги»

26. ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»

27. ДСТУ EN 547-3-2001 «Безпечність машин. Розміри людського тіла. Частина 3. Антропометричні дані»

28. ДСТУ EN 12464-1:2016 «Світло та освітлення. Освітлення робочих місць. Частина 1. Внутрішні робочі місця»

29. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»

30. ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної та локальної вібрації»

31. НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці»

32. . Желібо Є. Безпека життєдіяльності / Є. Желібо, В. Зацарний., 2019. – 344 с.

33. Автоматизовані системи обробки економічної інформації Підрчуник/ Г. В. Лавіанський,.., Р.С. Гром'як. К. Вища школа, 1995. 287с:іл.

34. Проектування інформаційного забезпечення систем обрабоки економічної інформації /Гром'як Р.С., Боднар І.В. - Тернопіль:ТНЕУ, 2009, 89 с.

35. Hromyak R., Nemish V. (2023). Estimation of the structural p parameter for a number of structutal materials. Scientific Journal of TNTU (Tern.), vol 112, no 4, pp. 67-72.

ДОДАТКИ

Додаток А

Частина скрипта реалізація рухом персонажем ‘PlayerMovement’

```
using System.Collections;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public PlayerData Data;

    #region Variables
    public Rigidbody2D RB { get; private set; }
    public Animator animator;
    [SerializeField] private AudioManager audioManager;
    public bool IsFacingRight { get; private set; }
    public bool IsJumping { get; private set; }
    public bool IsWallJumping { get; set; }
    public bool CanFlipGravity { get; set; }
    private bool canFlipGravityInAir = true; // Додаємо цю змінну для
контролю зміни гравітації в повітрі
    private bool extraGravityFlipGranted = false; // Додаємо цю змінну для
об'єкта, що дозволяє ще раз змінити гравітацію в повітрі

    public float LastOnGroundTime { get; private set; }
    public float LastOnWallTime { get; private set; }
    public float LastOnWallRightTime { get; private set; }
    public float LastOnWallLeftTime { get; private set; }
    private bool wasGrounded = true;
    private bool _isJumpCut;
    private bool _isJumpFalling;
    private bool _isGravityFlipped;
    private float _wallJumpStartTime;
    private int _lastWallJumpDir;
    private bool isWalking = false;
    public Vector2 _moveInput;
    public float LastPressedJumpTime { get; private set; }

    [Header("Checks")]
    [SerializeField] public Transform _groundCheckPoint;
    [SerializeField] public Vector2 _groundCheckSize = new Vector2(0.49f,
0.03f);
    [Space(5)]
    [SerializeField] private Transform _frontWallCheckPoint;
    [SerializeField] private Transform _backWallCheckPoint;
    [SerializeField] private Vector2 _wallCheckSize = new Vector2(0.5f,
1f);

    [Header("Layers & Tags")]
    [SerializeField] public LayerMask _groundLayer;
    #endregion

    private void Awake()
    {
        RB = GetComponent<Rigidbody2D>();
```

```
        animator = GetComponent<Animator>(); // Ensure Animator is
attached to the same GameObject
    }

    private void Start()
    {
        SetGravityScale(Data.gravityScale);
        IsFacingRight = true;
    }

    private void Update()
    {
        LastOnGroundTime -= Time.deltaTime;
        LastOnWallTime -= Time.deltaTime;
        LastOnWallRightTime -= Time.deltaTime;
        LastOnWallLeftTime -= Time.deltaTime;
        LastPressedJumpTime -= Time.deltaTime;

        _moveInput.x = Input.GetAxisRaw("Horizontal");
        _moveInput.y = Input.GetAxisRaw("Vertical");

        if (_moveInput.x != 0)
            CheckDirectionToFace(_moveInput.x > 0);

        if (Input.GetKeyDown(KeyCode.Space) ||
Input.GetKeyDown(KeyCode.C) || Input.GetKeyDown(KeyCode.J))
        {
            OnJumpInput();
        }

        if (Input.GetKeyUp(KeyCode.Space) || Input.GetKeyUp(KeyCode.C) ||
Input.GetKeyUp(KeyCode.J))
        {
            OnJumpUpInput();
        }

        if (CanFlipGravity && Input.GetKeyDown(KeyCode.F))
        {
            FlipGravity();
        }

        bool isGroundedNow =
Physics2D.OverlapBox(_groundCheckPoint.position, _groundCheckSize, 0,
_groundLayer);

        if (!wasGrounded && isGroundedNow)
        {
            // Гравець щойно приземлився
            if (audioManager != null)
            {
                audioManager.PlayLandingSound();
            }
            canFlipGravityInAir = true; // Дозволяємо зміну гравітації
після приземлення
            extraGravityFlipGranted = false; // Скидаємо додаткову зміну
гравітації після приземлення
        }
```

```
        wasGrounded = isGroundedNow; // Оновлюємо стан приземлення

        if (isGroundedNow)
        {
            LastOnGroundTime = Data.coyoteTime;
        }

        if         (((Physics2D.OverlapBox(_frontWallCheckPoint.position,
_wallCheckSize, 0, _groundLayer) && IsFacingRight)
            ||        (Physics2D.OverlapBox(_backWallCheckPoint.position,
_wallCheckSize, 0, _groundLayer) && !IsFacingRight)) && !IsWallJumping)
            LastOnWallRightTime = Data.coyoteTime;

        if         (((Physics2D.OverlapBox(_frontWallCheckPoint.position,
_wallCheckSize, 0, _groundLayer) && !IsFacingRight)
            ||        (Physics2D.OverlapBox(_backWallCheckPoint.position,
_wallCheckSize, 0, _groundLayer) && IsFacingRight)) && !IsWallJumping)
            LastOnWallLeftTime = Data.coyoteTime;

        LastOnWallTime          =          Mathf.Max(LastOnWallLeftTime,
LastOnWallRightTime);

        // Визначення стану падіння
        if ((!_isGravityFlipped && RB.velocity.y < 0 && !isGroundedNow) ||
(_isGravityFlipped && RB.velocity.y > 0 && !isGroundedNow))
        {
            _isJumpFalling = true;
        }
        else
        {
            _isJumpFalling = false;
        }

        if  (IsJumping  &&  (_isGravityFlipped  ?  RB.velocity.y  >  0  :
RB.velocity.y < 0))
        {
            IsJumping = false;

            if (!IsWallJumping)
                _isJumpFalling = true;
        }

        if   (IsWallJumping   &&   Time.time   -   _wallJumpStartTime   >
Data.wallJumpTime)
        {
            IsWallJumping = false;
        }

        if (LastOnGroundTime > 0 && !IsJumping && !IsWallJumping)
        {
            _isJumpCut = false;

            if (!IsJumping)
                _isJumpFalling = false;
        }

        if (CanJump() && LastPressedJumpTime > 0)
```

```
        {
            IsJumping = true;
            IsWallJumping = false;
            _isJumpCut = false;
            _isJumpFalling = false;
            Jump();
        }
        else if (CanWallJump() && LastPressedJumpTime > 0)
        {
            IsWallJumping = true;
            IsJumping = false;
            _isJumpCut = false;
            _isJumpFalling = false;
            _wallJumpStartTime = Time.time;
            _lastWallJumpDir = (LastOnWallRightTime > 0) ? -1 : 1;

            WallJump(_lastWallJumpDir);
        }


        // Додаємо перевірку на стіну під час падіння
        if (_isJumpFalling && _moveInput.x != 0 && IsTouchingWall())
        {
            _moveInput.x = 0; // Блокуємо рух в сторону стіни
        }

        // Відстежуємо стан ходьби
        if (_moveInput.x != 0 && isGroundedNow && !isWalking)
        {
            StartWalking();
        }
        else if ((_moveInput.x == 0 || !isGroundedNow) && isWalking)
        {
            StopWalking();
        }
        UpdateAnimationParameters();
    }
    public Vector2 GetMoveInput()
    {
        return _moveInput;
    }
    private void StartWalking()
    {
        isWalking = true;
        if (audioManager != null)
        {
            audioManager.StartFootsteps();
        }
    }
    private void StopWalking()
    {
        isWalking = false;
        if (audioManager != null)
        {
            audioManager.StopFootsteps();
        }
    }
```

Додаток Б

Реалізація скрипта контролю платформами ‘Player Tilemap Controller’

```
using UnityEngine;
using UnityEngine.Tilemaps;

public class PlayerTilemapController : MonoBehaviour
{
    [SerializeField] private float toggleRadius = 5f; // Радіус зони навколо гравця для керування платформами
    [SerializeField] private Tilemap platformTilemap; // Tilemap, яка містить платформи
    [SerializeField] private float toggleCooldown = 2f; // Час відновлення в секундах


    private bool canToggleTilemap = false;
    private float lastToggleTime;

    private void Update()
    {
        if (canToggleTilemap && Input.GetKeyDown(KeyCode.Q) && Time.time >= lastToggleTime + toggleCooldown)
        {
            TogglePlatforms();
            lastToggleTime = Time.time; // Оновлюємо час останньої активації
        }
    }

    public void EnableTilemapToggle()
    {
        canToggleTilemap = true;
        // Ви можете додати тут логіку для відображення UI, що показує, що можливість активована
    }

    private void TogglePlatforms()
    {
        Vector3Int playerCellPos = platformTilemap.WorldToCell(transform.position);
        BoundsInt bounds = new BoundsInt(
            playerCellPos - new Vector3Int(Mathf.FloorToInt(toggleRadius), Mathf.FloorToInt(toggleRadius), 0),
            new Vector3Int(Mathf.CeilToInt(toggleRadius) * 2, Mathf.CeilToInt(toggleRadius) * 2, 1)
        );


        for (int x = bounds.xMin; x <= bounds.xMax; x++)
        {
            for (int y = bounds.yMin; y <= bounds.yMax; y++)
            {
```

```
                Vector3Int pos = new Vector3Int(x, y, 0);
                if (platformTilemap.HasTile(pos))
                {
                    PlatformToggle           platformToggle           =
platformTilemap.GetComponent<PlatformToggle>();
                    if (platformToggle != null)
                    {
                        platformToggle.ToggleTile(pos);

                    }
                }
            }
        }

    }

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.blue;
        Gizmos.DrawWireSphere(transform.position, toggleRadius);
    }
}
```

Додаток В

Реалізація скрипта можливості зміни гравітації ‘GravityFlipPickup’

```
using System.Collections;
using UnityEngine;

public class GravityFlipPickup : MonoBehaviour
{
    [SerializeField] private float respawnTime = 5f; // Час відновлення об'єкта після підбирання
    [SerializeField] private SpriteRenderer spriteRenderer; // Спрайт рендерер об'єкта
    [SerializeField] private Collider2D pickupCollider; // Колайдер об'єкта
    [SerializeField] private MonoBehaviour lightScript; // Скрипт освітлення об'єкта

    private void Start()
    {
        if (spriteRenderer == null)
        {
            spriteRenderer = GetComponent<SpriteRenderer>();
        }

        if (pickupCollider == null)
        {
            pickupCollider = GetComponent<Collider2D>();
        }

        if (lightScript == null)
        {
            lightScript = GetComponent<MonoBehaviour>(); // Замініть на ваш скрипт освітлення, якщо він інший
        }
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            PlayerMovement player = collision.GetComponent<PlayerMovement>();
            if (player != null)
            {
                player.GrantExtraGravityFlip(); // Надаємо можливість ще раз змінити гравітацію
                StartCoroutine(RespawnPickup()); // Починаємо процес відновлення об'єкта
                SetActiveState(false); // Вимикаємо об'єкт після підбирання
            }
        }
    }

    private IEnumerator RespawnPickup()
```

```
    {
        yield return new WaitForSeconds(respawnTime);
        SetActiveState(true); // Відновлюємо об'єкт після певного часу
    }

    private void SetActiveState(bool state)
    {
        spriteRenderer.enabled = state; // Включаємо/вимикаємо спрайт
        pickupCollider.enabled = state; // Включаємо/вимикаємо колайдер
        if (lightScript != null)
        {
            lightScript.enabled = state; // Включаємо/вимикаємо скрипт
освітлення
        }
    }
}
```

Додаток Е

Реалізація скрипта можливості зміни гравітації ‘PlayerRespawn’

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

public class PlayerRespawn : MonoBehaviour
{
    private Vector3 initialPosition;
    private Vector3 respawnPosition;
    private int activeCheckpointID = -1;
    private Dictionary<int, Vector3> checkpoints = new Dictionary<int,
Vector3>();
    private Dictionary<int, bool> gravityStates = new Dictionary<int,
bool>();
    private bool hasCheckpoint = false;
    private PlayerMovement playerMovement;

    private void Start()
    {
        initialPosition = transform.position; // Початкова позиція гравця
        respawnPosition = initialPosition;
        playerMovement = GetComponent<PlayerMovement>();
    }

    public void SetCheckpoint(int checkpointID, Vector3
newRespawnPosition, bool isGravityFlipped)
    {
        if (checkpoints.ContainsKey(checkpointID))
        {
            checkpoints[checkpointID] = newRespawnPosition;
            gravityStates[checkpointID] = isGravityFlipped;
        }
        else
        {
            checkpoints.Add(checkpointID, newRespawnPosition);
            gravityStates.Add(checkpointID, isGravityFlipped);
        }

        activeCheckpointID = checkpointID;
        respawnPosition = newRespawnPosition;
        hasCheckpoint = true;

        ResetOtherCheckpoints(checkpointID);
    }

    private void ResetOtherCheckpoints(int activeID)
    {
        Checkpoint[] allCheckpoints = FindObjectsOfType<Checkpoint>();
        foreach (Checkpoint cp in allCheckpoints)
        {
            if (cp.checkpointID != activeID)
            {
                cp.ResetCheckpoint();
            }
        }
```

```
    }

    public void Respawn()
    {
        if (hasCheckpoint)
        {
            transform.position = respawnPosition;
            bool isGravityFlipped = gravityStates[activeCheckpointID];
            playerMovement.SetGravityFlipped(isGravityFlipped);
        }
        else
        {
            transform.position = initialPosition;
            playerMovement.SetGravityFlipped(false);
        }

        // Відновити здоров'я або інші параметри гравця при відродженні
тут
        HealthSystem healthSystem = GetComponent<HealthSystem>();
        if (healthSystem != null)
        {
            healthSystem.Respawn();
        }
    }
}
```