МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІГРОВИЙ ЗАСТОСУНОК В ЖАНРІ ПЛАТФОРМЕР НА РУШІЇ UNITY

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910127

Виконала студентка 4-го курсу, групи 401
_____ Л .В. Ященко
«__» _____р.

Керівник: ст. викладач
_____С. Ю. Боровльова
«__» _____р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 401 факультету комп'ютерних наук Ященко Лілії Віталіївні.

1. Тема кваліфікаційної роботи «Ігровий застосунок в жанрі платформер на рушії Unity».

Керівник роботи Боровльова Світлана Юріївна, старший викладач.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: ринок ігрової індустрії, підходи до створення 2D відеогри.

Очікуваний результат: розроблена 2D комп'ютерна гра в жанрі платформер, шляхом використання середовища розробки Unity.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- розкрити теоретичні засади створення 2D відеогри;
- огляд існуючих аналогів;
- обґрунтувати вибір інструментальних засобів розробки гри;
- розробити та здійснити програмну реалізацію 2D гри.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Охорона праці на робочих місцях у відділі розробки програмного забезпечення ЧНУ ім. Петра Могили».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	А. О. Алексєєва канд. тенх. наук, доцент	

Керівник роботи __д-р техн. наук, проф. Кондратенко Ю. П._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Ященко Л. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Ігровий застосунок в жанрі платформер на рушії Unity

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2022	27.10.2022	Виконано
2	Отримання завдання на виконання БКР	10.11.2022	12.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	04.12.2022	06.12.2022	Виконано
4	Отримання завдання на переддипломну практику	26.04.2023	26.04.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: аналіз існуючих аналогів, написання вступу, завершення розробки застосунку, написання розділів.	15.05.2023	22.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	01.06.2023	22.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	27.06.2023	27.06.2023	Виконано

Розробив студент __Ященко Л. В._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ ст. викладач, Боровльова С. Ю _____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_10_» _____12_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студентки групи 401 ЧНУ ім. Петра Могили

Ященко Лілії Віталіївни

Тема: «Ігровий застосунок в жанрі платформер на рушії Unity»

Актуальність роботи полягає в зниженні стресу гравця під час гри у врахуванні военного стану, де люди стикаються зі збільшеним рівнем стресу та напруги і необхідністю відволіктися.

Об'єкт роботи – процеси розробки двовимірного ігрового застосунку у жанрі платформер для ПК.

Предмет роботи – алгоритми та підходи до розробки комп’ютерних 2D ігор в середовищі Unity.

Метою бакалаврської кваліфікаційної роботи є популяризація відеогри серед непрофесійних гравців, шляхом створення проєкту в Unity з простим але інтенсивним геймплеєм та атмосферним сеттінгом, що задовольнить потреби таких користувачів та буде позитивно впливати на їх психологічний стан.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розглядається історія виникнення відеоігор, різновиди жанрів гри та аналоги ігрових застосунків.

У другому розділі обґрунтовано вибір інструментів для розробки 2D гри.

У третьому розділі було описано процес розробки застосунку.

В результаті розроблено 2D гру в жанрі платформер.

Бакалаврська кваліфікаційна робота містить 78 сторінок, 49 рисунків, 3 таблиці, 25 використаних джерел та 1 додаток.

Ключові слова: гра, Unity, персонаж, клас, спрайт.

ABSTRACT

to the bachelor's qualification work by the student of group 401 of Petro Mohyla Black Sea National University
Yashchenko Liliia Vitaliivna
Topic: "Game application in the platform genre on the Unity engine"

The relevance of the work is to reduce the stress of the player during the game, taking into account the state of war, where people are faced with an increased level of stress and tension and the need to be distracted.

The object of the work is the process of developing a two-dimensional game application in the platformer genre for PC.

The subject of the work is algorithms and approaches to the development of 2D computer games in the Unity environment.

The aim of the bachelor's qualification work is to popularize the video game among non-professional players by creating a project in Unity with a simple but intense gameplay and an atmospheric setting that will satisfy the needs of such users and have a positive effect on their psychological state.

The explanatory note consists of an introduction, four chapters, conclusions and appendices.

The first chapter examines the history of video games, different game genres and analogs of game applications.

In the second section, the choice of tools for the development of a 2D game is justified.

The third chapter described the application development process.

As a result, a 2D game in the platformer genre was developed.

The bachelor thesis contains 78 pages, 49 figures, 3 tables, 25 used sources and 1 appendix.

Key words: game, Unity, character, class, sprite.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

BG – BackGround

ВСТУП

В сучасному світі ігрова індустрія постійно зростає та розвивається, надаючи мільйонам людей можливість відчути емоційне задоволення та взаємодію з віртуальними світами. Ігри стають не лише засобом розваги, але й предметом досліджень для вивчення їх впливу на мозок та психіку людини. У цьому контексті створення ігрових застосунків стає важливою та актуальною темою, що вимагає наукового підходу та розуміння принципів, які лежать в основі цього процесу.

Один з жанрів ігор, який здобуває популярність серед гравців, є платформер. Він відображає ігровий світ, де головний герой пересувається по різних платформах, збираючи предмети та перестрибуючи перешкоди. Створення платформерів вимагає не лише технічної компетентності, але й розуміння впливу цього жанру на мозок та психіку гравців.

Наприклад, дослідження, проведені Мелісою Стоут, професором психології з Університету Нотр-Дам, показали, що гра у відеоігри може покращити увагу та концентрацію гравця. Рухливість та виклики, які присутні в платформерах, сприяють активному мисленню та розвитку реакційних навичок.

Також, дослідження К. Янга, опубліковані в журналі "Computers in Human Behavior", вказують на те, що гра у відеоігри може позитивно впливати на настрій та самопочуття гравця.

У роботі будуть розглянуті основні принципи розробки ігрового застосунку на рушії Unity, зокрема, процес створення головного героя, налаштування фізики, створення рівнів та ігрових об'єктів. Також будуть проаналізовані особливості розробки 2D-графіки та анімації для платформера.

Об’єктом є процеси розробки двовимірного ігрового застосунку у жанрі платформер для ПК.

Предметом є алгоритми та підходи до розробки комп’ютерних 2D ігор в середовищі Unity.

Метою даної роботи є популяризація відеогри серед непрофесійних гравців, шляхом створення проєкту в Unity з простим але інтенсивним геймплеєм та атмосферним сеттінгом, що задовольнить потреби таких користувачів та буде позитивно впливати на їх психологічний стан. Завдання для досягнення поставленої мети:

- аналіз сучасного стану в ігровій індустрії;
- аналіз та обґрунтування вибору інструментальних засобів для розробки 2D проєкту;
- з’ясування аспектів розробки 2D гри за допомогою двигуна;
- постановка ідеї відеогри;
- розробка та програмна реалізацій 2D гри.

1 АНАЛІЗ СФЕРИ ІГРОВОЇ ІНДУСТРІЇ

1.1 Історія створення ігор

Ігри-платформери – це жанр відеоігор, у яких основна мета гравця полягає у подоланні серії рівнів, що складаються з платформ – різних рухомих або нерухомих об'єктів, що перешкоджають просуванню головного героя [1].

У платформерах зазвичай присутні такі елементи геймплею, як стрибки, біг, рухи вліво/вправо, збирання предметів і боротьба з ворогами. Головний герой може бути персонажем з пригодницького або фантастичного світу, наприклад, мавпою, що бігає по джунглях, космічним воїном або маленьким чоловічком, що шукає шлях до виходу з лабіринту.

Платформери мають різний рівень складності, від легких ігор для початківців до дуже складних та вимогливих для досвідчених гравців. Вони можуть бути підібрані на будь-який смак – від класичних 2D ігор, до сучасних 3D-платформерів зі складними головоломками та великими відкритими світами.

Деякі з найвідоміших прикладів ігор-платформерів включають в себе такі творіння, як Super Mario Bros., Sonic the Hedgehog, Donkey Kong Country, Rayman, Ori and the Blind Forest, Hollow Knight та інші.

Першою відомою 2D відеогрою є гра "Tennis for Two", яку розробив американський фізик Вільям Гіггс у 1958 році [2]. Гра була створена для показу в рамках "Дня науки" в лабораторії Брукгейвенського національного лабораторії, де Гіггс працював. Вона була створена на основі електронного осцилографа, який використовувався для відображення траєкторії руху м'яча на екрані.

У грі "Tennis for Two" гравці керували ракеткою, яка мала захопити і відправити м'яч назад до опонента. Гра була створена для двох гравців, і вона була дуже простою – не було жодних графічних ефектів, а лише лінії, які відображали траєкторію руху м'яча на екрані.

Хоча гра "Tennis for Two" не була комерційним продуктом і не набула широкої популярності, вона вважається однією з перших відеоігор і відкрила дорогу для подальшого розвитку цього напрямку в галузі розваг (рис. 1.1).

Рисунок 1.1 – Гра " Tennis for Two"

2D відеоігри з'явилися ще в далекому 1970 році і поступово розвивалися протягом наступних десятиліть. Від самого початку ці ігри були дуже простими, з мінімальною кількістю деталей і обмеженими можливостями. Однак, з часом вони стали більш складними, отримали нові функції та можливості, що зробило їх більш цікавими та затребуваними серед гравців.

Можна виділити декілька найважливіших етапів розвитку 2D відеоігор [3].

Ранні 2D ігри (1970-1980-і роки) – перші 2D ігри були дуже простими і складалися з мінімальної кількості деталей. Серед них можна виділити Space Invaders, Pac-Man, Galaxian та інші.

Візьмемо до прикладу гру Galaxian. Galaxian – це класична аркадна відеогра, розроблена компанією Namco у 1979 році. У грі гравець контролює космічний корабель, який має боротися зі своїми ворогами, які літають в формаціях на екрані. Головна мета гри полягає в знищенні всіх ворожих кораблів, які літають в різних формаціях згори до низу екрану. Корабель гравця може рухатись вправо та вліво, він може відкривати вогонь зі свого бластера, щоб знищувати ворожі кораблі, які

літають в його напрямку або від нього відхиляються. Гра складається зі ста рівнів, на кожному з яких з'являється нова формація ворожих кораблів.

Так, як і в інших іграх, у гри Galaxian є свої переваги та недоліки.

Переваги гри Galaxian:

простий та легкий для зрозуміння геймплей (геймплей – компонент гри, що відповідає за взаємодію гри та гравця), що зробив гру дуже популярною серед гравців;

- високий рівень складності, який приваблював більш досвідчених гравців;
- цікава музика та звукові ефекти, які додають грі атмосферності та драйву.

Недоліки гри Galaxian:

- відсутність глибокої сюжетної лінії, що робить гру більш аркадною та менш насиченою;
- обмежений набір ворожих кораблів та формацій, що робить гру більш передбачуваною та менш різноманітною;
- гра може виглядати застарілою в порівнянні з більш сучасними іграми з подібною механікою геймплею.

У цілому, Galaxian є класичною грою з простим, але веселим геймплеєм та добре підійде для тих, хто любить аркадні ігри (рис. 1.2).

Рисунок 1.2 – Інтерфейс відеогри Galaxian

Золота ера аркадних ігор (1980-1990-і роки) – в цей період 2D ігри отримали значний розвиток і стали більш складними та цікавими. В цей час були створені такі ігри, як Donkey Kong, Super Mario Bros., Street Fighter, Mortal Kombat та інші.

Розглянемо культову гру своїх часів – Donkey Kong. Donkey Kong – це класична аркадна гра, яка була розроблена компанією Nintendo в 1981 році. Головним героєм гри є Донкі Конг – велика гірська горила, яка викрала головну героїню гри, Пауліну. Гравець у ролі Маріо повинен пройти через різні рівні, подолати виклики та перешкоди, щоб врятувати головну героїню та перемогти Донкі Конга.

Геймплей Donkey Kong полягає в тому, що гравець керує Маріо, який повинен пройти кілька рівнів, щоб дістатися до верхівки будівлі, де Донкі Конг тримає заручницю. Кожен рівень складається з різних платформ та перешкод, таких як бочки, землетруси, вогняні кулі та інші. Гравець повинен уникати цих перешкод, пересуватися вгору та врешті-решт досягти вершини побудови. У процесі проходження рівнів гравець може зібрати різні бонуси, такі як молот, що дозволяє знищити перешкоди, та інші.

Переваги гри Donkey Kong полягають у її класичному геймплеї та культовому статусі. Гра відома своєю простотою та доступністю для гравців різних вікових категорій, що робить її привабливою для багатьох людей. Крім того, гра має високу репутацію серед шанувальників класичних аркадних ігор від Nintendo.

Одним із недоліків гри є те, що гра досить коротка, складаючи всього 4 рівні. Також, гра має деякі недоліки у графіці та звуку порівняно з більш сучасними іграми (рис. 1.3).

Рисунок 1.3 – Інтерфейс відеогри Donkey Kong

Ера 16-бітних ігор (1990-2000-і роки) – з'явилися нові платформи, такі як Sega Genesis та Super Nintendo Entertainment System, що дозволили розробникам створювати більш складні та деталізовані 2D ігри. У цей період з'явилися такі ігри, як Sonic the Hedgehog, Super Metroid, Castlevania: Symphony of the Night та інші.

Розглянемо на прикладі гри Sonic the Hedgehog. Sonic the Hedgehog – це відеогра, яка вийшла в 1991 році і стала однією з найвідоміших ігор на приставці Sega Genesis. Головним героєм гри є Сонік-їжак, який має швидкість світла і здатність обертатися в кулю. Гравці втілюються у ролі Соніка, рухаючись зі швидкістю в межах рівня, збираючи кільця, знищуючи ворогів та шукаючи виходи з кожного рівня.

Головною перевагою геймплею Sonic the Hedgehog є швидкість та ритмічність. Гравці відчувають, що їхній персонаж рухається дуже швидко, що додає грі енергії та драйву. Крім того, гра має чудову музику та кольорову графіку, що створює настрій та підтримує веселу атмосферу.

Серед недоліків гри можна зазначити те, що деякі рівні можуть бути досить складними, і для проходження їх потрібна багато вправності та досвіду. Крім того, деякі елементи гри можуть викликати стрес та незручності, наприклад, втрата кільць під час зіткнення з ворогами або перешкодами.

Загалом, Sonic the Hedgehog – це захоплююча гра, яка стала легендою серед відеоігор і надихнула на створення багатьох інших ігор зі швидкими персонажами. Її швидкість та динаміка приваблюють гравців й до цього дня, і вона продовжує залишатися однією з найбільш популярних ігор у світі (рис. 1.4).

Рисунок 1.4 – Інтерфейс відеогри Sonic the Hedgehog

Сучасні 2D ігри (після 2000-их років) – з появою нових технологій та інструментів, таких як Unity та Unreal Engine, створення 2D ігор стало набагато легшим та доступнішим. В сучасний час 2D ігри включають в себе як класичні ретро-ігри, так і нові ігри зі складними головоломками та відкритим світом для дослідження. Серед сучасних 2D ігор можна виділити такі хіти, як Hollow Knight, Celeste, Ori and the Blind Forest, Shovel Knight та інші.

Hollow Knight – це пригодницька ігра в жанрі метроїдванії, розроблена студією Team Cherry. Головний герой гри – комаха-листоїд, яка подорожує підземним світом Hallownest, досліджуючи його та борючись з небезпеками, що населяють цей світ.

Геймплей Hollow Knight складається з дослідження великого світу, знаходження та збору різноманітних предметів, включаючи гроші, реліквії, артефакти та інші предмети. Гравець може розвивати свого героя, вивчаючи нові навички та вдосконалюючи свою зброю та купуючи спеціальні предмети, що допоможуть йому подолати перешкоди в грі.

Однією з переваг Hollow Knight є його надзвичайно пророблений світ з його унікальними локаціями та глибокою історією. Гра має чудову графіку та звуковий дизайн, що доповнює ігральний досвід.

Також варто зазначити, що у грі висока складність, що може бути і перевагою, і недоліком. Деякі гравці можуть вважати її занадто складною, особливо в босс-битвах та складних локаціях. Крім того, деякі гравці можуть відчувати, що гра занадто довга або повільна, особливо на початку гри.

Узагалі ж, Hollow Knight – це високоякісна гра, яка надає гравцям багато годин насолоди та пригод в унікальному ігровому світі (рис. 1.5).

Рисунок 1.5 – Інтерфейс відеогри Hollow Knight

У сучасній ері відеоігор, багато розробників використовують 2D графіку для створення унікальних ігрових світів. Вони поєднують в собі різні елементи жанрів та надають гравцям можливість насолоджуватись красивою графікою та захоплюючим геймплеєм.

Також варто зазначити, що розвиток інтернету та мобільних технологій дозволяє створювати та розповсюджувати 2D ігри на різних платформах. Це дозволяє гравцям грати в ігри на своїх смартфонах, планшетах, переносних консолях та інших пристроях.

1.2 Класифікація відеоігор

Відеоігри – це розважальна форма, яка стала дуже популярною за останні десятиліття. Сьогодні існує безліч жанрів та типів відеоігор, що можуть відрізнятися за стилістикою, геймплеєм, механікою та іншими характеристиками. У зв'язку з цим, необхідно розглянути класифікацію відеоігор для більш детального розуміння цього виду розваги.

Однією з найбільш популярних класифікацій відеоігор є класифікація за жанрами. Жанри відеоігор визначаються за їх характерними ознаками, такими як

головна мета гри, геймплей, типи завдань, тематика та інші. До основних жанрів відеоігор належать [3]:

- екшн: головна мета гри полягає у боротьбі з противниками та виконанні різноманітних завдань, що вимагають швидкості реакції та майстерності в обіймах зброї;
- аркади: головна мета полягає у збиранні бонусів, виконанні завдань та перешкоджанні негативним факторам в грі;
- рольові ігри: гравець грає в ролі персонажа, розвиває його та виконує завдання у віртуальному світі;
- симулятори: гравець може зануритися в віртуальний світ та відчути себе в ролі справжньої людини, керуючи певними виробами, транспортом або життєвими ситуаціями;
- стратегії: головна мета полягає у побудові та управлінні власною армією, розробці та вдосконаленні власного бази та боротьбі з іншими гравцями;
- гонки: головна мета полягає в перемозі в гонках на різних видах транспорту;
- спортивні ігри: гравець грає роль спортсмена, виконуючи завдання та перемагаючи в змаганнях;
- пригодницькі ігри: головна мета полягає у виконанні завдань та розв'язанні головоломок у віртуальному світі;
- хоррори: головна мета полягає у створенні напруження та жаху від ігрової ситуації.

Існує також класифікація відеоігор за платформами, на яких вони доступні. Найбільш поширеними платформами є персональні комп'ютери, ігрові консолі та мобільні пристрої. Кожна з цих платформ має свої особливості, від яких залежить геймплей відеоігор. Наприклад, ігрові консолі зазвичай мають більш потужну апаратну частину, що дозволяє створювати відмінну графіку та більш складні ігрові механіки.

Окрім цього, існує класифікація відеоігор за віковими категоріями. В залежності від віку гравця визначається відповідний рівень складності гри, її тематика та візуальний стиль. Зазвичай ігри розділяють на категорії, які зібрані в Таблиці 1.1 [4]:

Таблиця 1.1 – Розподіл ігор за віковими категоріями

Категорія	Вік	Опис
EC	Для дітей молодшого віку	Гра підходить для дітей від 3 років і не містить матеріалів, які батьки могли б вважати невідповідними. Продукти, що отримали цей рейтинг, спочатку розробляються для дітей і зазвичай є розвиваючі ігри
E	Для всіх	Зміст цілком підходить для будь-якого віку. Такі ігри можуть сподобатися і дорослим. Ігри з цим рейтингом можуть містити мінімальне насильство, переважно мультиплікаційного характеру, без жорстокості

Продовження таблиці 1.1

E10+	Для всіх від 10 років і старше	Проєкти з даним рейтингом можуть містити більше мультиплікаційного або м'якого насильства, або дещо відверті сцени, або мінімальну кількість крові
T	Підліткам	Гра підходить для осіб від 13 років. Проєкти цієї категорії можуть містити насильство, непристойні сцени, грубий гумор, в міру відвертий сексуальний вміст, кров або нечасте використання ненормативної лексики
M	Для дорослих	Матеріали гри не підходять для осіб молодше 17 років. Можуть містити жорстоке насильство, велику кількість крові з розчленуванням, непристойні сексуальні сцени чи грубу ненормативну лексику

Закінчення таблиці 1.1

AO	Тільки для дорослих	Зміст гри тільки для дорослих старше 18 років. Продукти цієї категорії можуть містити тривалі сцени жорстокого насильства та/або дуже відвертий сексуальний вміст

Також важливо враховувати різні жанри відеоігор при їх класифікації. Наприклад, у ролевих іграх гравець бере на себе роль персонажа, який зазвичай виконує різні завдання, взаємодіє з іншими персонажами та розвиває свої навички. У шутерах головна мета полягає у знищенні ворогів за допомогою зброї. У пригодницьких іграх гравець виконує різні завдання, взаємодіє з іншими персонажами та розв'язує головоломки.

Також можна класифікувати відеоігри за рівнем відкритості світу. У відкритих світах гравець має можливість вільно переміщатися по світу, виконувати різні завдання та взаємодіяти з іншими персонажами. У лінійних іграх гравець має обмежений вибір дій та можливостей переміщення по світу.

Залежно від способу гри, відеоігри можуть бути класифіковані як одиночні або мультиплеєрні. У одиночних іграх гравець грає сам, зазвичай виконуючи різні завдання або проходячи історію. У мультиплеєрних іграх гравці можуть грати разом, взаємодіючи між собою та конкуруючи один з одним.

Нарешті, можна класифікувати відеоігри за способом керування. У традиційних відеоіграх керування зазвичай здійснюється за допомогою геймпаду або клавіатури та миші. Однак, з появою розумних технологій, таких як голосове та жестове керування, можливості керування відеоіграми стають ще більш різноманітними.

Усі ці класифікації не є жорсткими категоріями, оскільки багато відеоігор можуть належати до декількох категорій одночасно. Наприклад, відкритий світ може бути присутнім у пригодницькій грі, а також у рольовій грі. Однак, класифікація відеоігор допомагає краще зрозуміти суть гри та відібрати ту, яка найбільше підходить певному гравцю.

Класифікація відеоігор є важливою для розробників, оскільки вона дозволяє їм краще зорієнтуватися в тому, яку аудиторію вони хочуть залучити до своєї гри. Наприклад, якщо розробники планують створити відеоігру для дітей, вони повинні дотримуватися вікової класифікації та створювати гру, яка відповідає розвиваючим потребам маленьких гравців.

Класифікація відеоігор також є важливою для батьків, які хочуть знати, які ігри є підходящими для їхніх дітей. Вони можуть користуватися різними класифікаціями, щоб знайти гру, яка відповідає віковій категорії та інтересам їхньої дитини.

Нарешті, класифікація відеоігор також важлива для гравців, оскільки вона допомагає їм знайти гру, яка найбільше відповідає їхнім інтересам та уподобанням. Вони можуть використовувати різні класифікації, щоб знайти гру, яка має певні характеристики, такі як жанр, графічний стиль, механіки гри тощо.

Загалом, класифікація відеоігор є важливим інструментом для розуміння та аналізування світу відеоігор. Вона дозволяє нам краще зрозуміти те, що робить гру унікальною та цікавою для різних аудиторій, а також допомагає нам знайти гру, яка найкраще відповідає нашим інтересам та уподобанням.

Проте варто зазначити, що класифікація відеоігор є динамічною та змінною, оскільки з часом з'являються нові технології та інновації, що змінюють способи розробки та відтворення відеоігор. Тому, класифікації можуть змінюватися та доповнюватися з часом, щоб відображати найновіші тренди та технології в галузі відеоігор.

У кінцевому рахунку, класифікація відеоігор є корисним інструментом для тих, хто працює в галузі відеоігор або просто любить грати в них. Вона допомагає

зрозуміти суть гри та відібрати ту, яка найбільше підходить певному гравцю, а також допомагає розробникам зорієнтуватися в тому, яку аудиторію вони хочуть залучити до своєї гри. Таким чином, класифікація відеоігор є важливим елементом галузі відеоігор та сприяє подальшому розвитку цієї індустрії.

1.2.1 Екшн

Жанр гри – екшн є одним з найпопулярніших і найширших жанрів в індустрії відеоігор. Він об'єднує в собі елементи стрілянини, бійки, пригод, платформери та інші механіки гри, що мають на меті створити динамічний та захоплюючий геймплей.

Основні характеристики жанру гри – екшн [5]:

1) динаміка. Екшн ігри мають динамічний та енергійний геймплей, що передається гравцю через швидкість, рухливість та жвавість дій героїв. Гравці зазвичай повинні реагувати на швидкі та неочікувані події в грі, такі як атаки ворогів та виконання завдань за обмежений час;

2) дойова система. Бойова система в екшн іграх може бути різноманітною, від простої бійки між двома героями до складних комбо-атак та використання зброї. Гравці повинні володіти знаннями про різні способи бою та знати, як ефективно використовувати їх для перемоги в боях;

3) дізноманітність. Жанр гри – екшн охоплює широкий спектр ігор, від класичних платформерів та стрілянин до пригодницьких ігор та ігор з відкритим світом. Вони можуть бути засновані на різних темах, від фантастики до історичних епох, і мати різні візуальні стилі;

4) пригоди та історії. Більшість ігор створюють емоційність через головну нитку сюжету, який може бути пов'язаний з виконанням різноманітних завдань та місій. Гравці можуть взяти на себе роль героя, який бореться зі злом, врятувати світ від катастрофи, чи знайти відповідь на складні питання, що стоять перед ними.

Історії можуть бути наповнені елементами фантастики, пригод та культури, що збагачує гру та робить її більш цікавою;

5) мультиплеєр. Більшість ігор в жанрі гри – екшн мають режим мультиплеєра, що дозволяє гравцям змагатися між собою в онлайн чи локальному режимі. Це дає можливість насолоджуватися грою з друзями та відчувати власні успіхи та перемоги.

Розглянемо успішну гру в жанрі екшн на прикладі проєкту God of War. God of War – це екшн-пригода, створена студією Santa Monica Studio і видана Sony Interactive Entertainment в 2018 році. Гра є четвертою за рахунком в серії God of War та продовженням попередніх ігор.

Головний герой гри – Кратос, бог війни з грецької міфології, який цього разу переїхав до скандинавської міфології та знаходиться в пошуках нового початку для себе та свого сина Атрея. Ігровий процес полягає в розв'язуванні головоломок, битвах з різними ворогами та босами, а також відкриттям величезного відкритого світу. Ця гра відрізняється від попередніх ігор серії тим, що має новий вигляд, механіку гри та історію.

Однією з переваг гри є неймовірно деталізований світ та чудова графіка. Різні локації, такі як глибокі ліси, затерті віками гірські шляхи та містичні печери, дозволяють гравцям зануритися у світ гри та відчути себе частиною нього. Бойова система гри також є однією з найкращих у жанрі екшн, оскільки вона поєднує в собі швидкість та силу, а також різноманітність зброї та магії.

Недоліком гри може бути відносно повільний темп гри, особливо на початку, коли гравцям потрібно звикнути до нової механіки та геймплею. Також деякі гравці можуть вважати, що гра є дуже вимогливою, особливо на вищих рівнях складності, тому вона може бути викликом для менш досвідчених гравців (рис. 1.6).

Рисунок 1.6 – Інтерфейс відеогри God of War

1.2.2 Аркади

Жанр гри – аркади є одним з найстаріших та найпопулярніших жанрів комп'ютерних ігор. Він з'явився у 70-х роках минулого століття, коли комп'ютерні ігри ще були у своїй початковій стадії розвитку. У цьому жанрі гравцеві потрібно управляти персонажем, який знаходиться у віртуальному світі, де йому належить збирати предмети, боротися з ворогами та пройти якомога більше рівнів.

Основні характеристики жанру гри-аркади [3]:

1) простота управління. Одна з основних рис цього жанру гри-аркади – це простота управління. Гравцю потрібно лише користуватися декількома клавішами, щоб управляти персонажем, збирати предмети та боротися з ворогами;

2) швидкість. Іншою характеристикою аркадних ігор є їхня швидкість. Гравцеві доводиться діяти швидко та точно, щоб уникнути пасток та перемогти ворогів;

3) різноманітність. Ігри-аркади можуть бути різноманітними за жанром та настроєм. Наприклад, серед найвідоміших аркадних ігор можна назвати Space Invaders, Pac-Man, Tetris, Sonic the Hedgehog та інші. Кожна з цих ігор має свій унікальний сюжет та геймплей;

4) високий рівень складності. Багато аркадних ігор характеризуються високим рівнем складності. Гравець повинен бути досконалим у своїх навичках, щоб пройти всі рівні та перемогти ворогів;

5) відсутність сюжету. У багатьох аркадних іграх відсутній чіткий сюжет. Гравцеві не потрібно розуміти складну історію чи вивчати персонажів. Їм лише потрібно виконувати певні завдання, щоб перемогти в грі;

6) відсутність збереження прогресу. У більшості аркадних ігор немає можливості зберігати прогрес. Це означає, що гравець повинен починати гру спочатку, якщо він загубив усі свої життя. Така характеристика гри стимулює гравця до більш високої майстерності та вимагає від нього більшої уваги та концентрації;

7) рекорди. Більшість аркадних ігор мають систему рекордів, де гравці можуть змагатися за те, хто зможе пройти гру найшвидше або набрати найбільшу кількість очок. Це створює стимул для гравців, щоб зробити краще та підняти свій рівень майстерності.

Прикладами аркадних ігор можуть бути такі відомі творіння, як Pac-Man, Space Invaders, Donkey Kong, Tetris, Super Mario Bros., Sonic the Hedgehog та інші. Кожна з цих ігор має свої унікальні особливості та геймплей.

Розглянемо на прикладі гри Pac-Man.

Pac-Man – це класична аркадна гра, вперше випущена в Японії в 1980 році компанією Namco. У грі гравець керує головним героєм – жовтим круглим персонажем, який повинен з'їсти всі кульки на лабіринті, уникати привидів і збирати бонуси, щоб набрати балів і пройти наступний рівень. Гра складається з 256 рівнів та має чотири привиди-противники (Blinky, Pinky, Inky і Clyde), кожен з яких має свій характер та рухається по лабіринту з різною швидкістю і тактикою.

Гра має просту, але ефективну музику та звукові ефекти, що робить її легко впізнаваною та незабутньою.

Гравець отримує додаткові бали за збирання бонусів та знищення привидів, якщо вони перебувають у стані "вразливості". Має низку складнощів, таких як складність управління на високих рівнях, невелику кількість часу для проходження рівня, та стратегічний розмір лабіринту.

Можна виділити такі переваги гри:

- гра має простий, але ефективний геймплей, що зробив її культовою;
- гра пропонує низку різноманітних викликів, що дозволяє гравцеві розвивати свої навички та стратегії гри.

Pac-Man - це класична гра, яка стала піонером жанру, що в наш час є визнаним культурним явищем.

Недоліки гри:

- гра має досить просту графіку, що не задовольняє сучасні вимоги;
- Pac-Man має обмежену кількість можливостей і рівнів, що може призвести до швидкого набридання;
- на високих рівнях гра може стати досить складною і стресовою для деяких гравців;
- невеликий розмір лабіринту та обмежена кількість різних ворогів можуть призвести до того, що гра стає одноманітною і передбачуваною.

У цілому, Pac-Man – це класична гра, яка зарекомендувала себе як одна з найпопулярніших аркадних ігор у світі. Її простий, але ефективний геймплей, незабутня музика та звукові ефекти, а також культовий статус зробили її незабутньою. Незважаючи на обмежені можливості та недоліки, Pac-Man залишається однією з найуспішніших та культових ігор у світі відеоігор (рис. 1.7).

Рисунок 1.7 – Інтерфейс відеогри Pac-Man

1.2.3 Рольові ігри

Рольові ігри, або RPG (Role-Playing Games), є одним з найбільш популярних жанрів комп'ютерних ігор. Вони дозволяють гравцям поглибитися в інший світ, стати героєм і розвиватися в межах вигаданого уявного світу.

Жанр гри рольові ігри – є одним з найбільш складних та динамічних жанрів в світі комп'ютерних ігор. Він базується на ідеї гравця, який вибирає персонажа, розвиває його та виконує завдання в ігровому світі. Рольові ігри часто мають великі, продумані світи з складними персонажами, інтригуючими історіями та завданнями.

Характерними рисами RPG [6] є відкритий світ, розвиток персонажа, інвентар та система бойових дій. У RPG гравець зазвичай вибирає персонажа з різними характеристиками, здібностями та навичками. Персонаж може бути унікальним і мати власні цілі та мотиви, а також бути частиною історії гри.

Однією з ключових особливостей RPG є розвиток персонажа. Гравець може змінювати різні характеристики свого персонажа, такі як рівень, здібності, навички та інше, в результаті чого персонаж стає сильнішим та ефективнішим у бою. Іншою важливою характеристикою RPG є інвентар, що дозволяє гравцю збирати, купувати та продавати різні предмети, які допомагають персонажу в його подорожі.

Також є можливість здійснювати бойові дії в грі. Це може бути реалістична бойова система, яка дозволяє гравцеві управляти своїм персонажем під час бою, або система автоматичних бойових дій, де гравець вибирає тільки стратегію бою.

Ще однією характеристикою RPG є історія та завдання. Гра може мати розгалужену історію з різними сюжетними лініями, що дозволяє гравцю вибирати, як розвивати історію. Завдання в грі можуть бути як важкими, так і легкими, але вони завжди вимагають від гравця розв'язування певних завдань, які допомагають розвивати історію гри та персонажа.

Найбільш популярні RPG це серії Final Fantasy, Elder Scrolls, Dragon Age, Mass Effect, а також відома гра World of Warcraft. Ці ігри мають великі, детально

виведені світи, змістовну історію, багато персонажів та завдань, що дозволяє гравцям максимально поглибитися в ігровий світ.

Рольові ігри є одним з найбільш популярних та динамічних жанрів комп'ютерних ігор. Вони дозволяють гравцям поглибитися в інший світ, стати героєм та розвиватися в межах вигаданого світу. Різноманітність персонажів та їх взаємодій, гнучкість в розвитку історії та можливість вибору впливають на геймплей та роблять гру цікавішою та більш варіативною.

Одним з головних факторів успіху рольових ігор є можливість взаємодії з іншими гравцями. Багато RPG пропонують мультиплеєрний режим гри, який дозволяє гравцям грати разом, співпрацювати та змагатися один з одним. Це збільшує різноманітність і глибину геймплею, забезпечуючи нові виклики та можливості для гравців.

Ще одним фактором успіху RPG є їх соціальна функція. Ці ігри часто є місцем спілкування для гравців з різних країн та культур, що дає можливість знайомитися з новими людьми та заводити друзів. Гравці можуть спілкуватися в межах гри, обговорювати ігрові питання, тактики та спільно вирішувати завдання.

Однак, як і будь-який інший жанр ігор, RPG мають свої недоліки. Часто гра може бути дуже складною та заплутаною, що може відлякувати новачків. Крім того, багато RPG мають дуже великий обсяг, що може займати багато часу та вимагати від гравців великої уваги та концентрації.

Узагалі, RPG є доволі важливим та популярним жанром комп'ютерних ігор. Вони дозволяють гравцям поглибитися в інший світ та стати частиною захоплюючих історій. Рольові ігри мають багато різноманітних характеристик, що роблять їх унікальними та цікавими для гравців. Вони надають можливість розвивати свої навички, спілкуватися з іншими гравцями, досліджувати світи та пригоди, а також відчувати емоційне занурення в ігровій реальності. Завдяки цьому, RPG стали невід'ємною частиною геймінгової культури та продовжують здобувати популярність серед гравців усіх вікових категорій.

Розглянемо на прикладі відомої гри World of Warcraft.

World of Warcraft (WoW) – це одна з найпопулярніших онлайн-ігор жанру MMORPG, розроблена компанією Blizzard Entertainment і випущена у 2004 році.

Характеристики гри:

- WoW дозволяє гравцеві вибрати одну з двох фракцій: Альянс або Орда, кожна з яких має свою унікальну історію, персонажів, локації та квести;
- у грі існує велика кількість класів персонажів (мисливець, маг, воїн тощо), які мають свої унікальні можливості та стиль гри;
- гравець може спілкуватися з іншими гравцями, об'єднуватися в гільдії, виконувати завдання разом, або ж змагатися один з одним в PvP режимі (гравець проти гравця);
- у грі є велика кількість локацій з різною тематикою та рівнем складності, включаючи відкритий світ, де гравець може рухатися вільно, і підземелля, де гравці можуть брати участь у рейдах та подібних великих подіях;
- WoW постійно оновлюється, додаючи нові локації, класи персонажів, предмети та інші функції;
- геймплей гри полягає в виконанні завдань (квестів) та збільшенні рівня персонажа. Гравець може обирати, які квести виконувати та який стиль гри він обере: складати команду для подолання складних підземель, займатися крафтом, грати у PvP режимі та багато іншого.

Переваги гри World of Warcraft:

- велика кількість контенту та можливостей для гравців;
- можливість взаємодії з іншими гравцями та об'єднання в гільдії;
- постійні оновлення та додатковий контент дозволяють гравцям завжди знайти щось нове і цікаве в грі;
- висока якість графіки та звукового супроводу;
- гнучкий геймплей, що дозволяє гравцям обирати свій власний стиль гри та підходити до завдань по-різному.

Недоліки гри World of Warcraft:

– гра може вимагати багато часу та зусиль від гравця, щоб досягти високих рівнів та отримати кращі предмети;

– іноді гра може стати монотонною, оскільки багато завдань повторюються;

– залежності від гри можуть стати проблемою для деяких гравців;

– вартість гри та її додатків може бути високою для деяких гравців.

У загальному, World of Warcraft – це велика та детально пророблена гра з доволі високою якістю графіки та звукового супроводу, що дозволяє гравцям відчути себе частиною віртуального світу. Однак, гра може вимагати багато часу та зусиль від гравців, щоб досягти високих рівнів та отримати кращі предмети (рис. 1.8).

Рисунок 1.8 – Інтерфейс відеогри World of Warcraft

1.2.4 Симулятори

Жанр гри-симулятори – це один із найпопулярніших жанрів в галузі комп’ютерних ігор. Симулятори намагаються якомога більш точно відтворити реальність, надаючи гравцям можливість зануритися у віртуальний світ і відчути себе в ролі різних персонажів або професій.

Жанр симуляторів може бути досить різноманітним, охоплюючи різні галузі життя - від авіації та автомобілів до сільського господарства та бізнесу. Симулятори можуть бути використані як для розваг, так і для навчання професійних навичок.

Характеристикою жанру гри-симулятора є те, що вони намагаються якомога більш точно відтворити реальність. Це означає, що гравці можуть відчути себе в ролі реального персонажа або професіонала, і вони повинні дотримуватися правил і законів, які стосуються цієї професії. Наприклад, у симуляторі авіації гравець повинен дотримуватися правил польоту, включаючи правила взлету і посадки, а також виконувати інші обов’язки пілота.

Однією з головних особливостей симуляторів є їх реалістичність. Графіка, звук та інші деталі створюють відчуття, що гравець дійсно знаходиться в реальному світі. Наприклад, у симуляторі автомобілів гравець може відчути, як руль реагує на керування автомобілем, як змінюється швидкість, коли він прискорюється або гальмує.

Існують різні типи симуляторів, такі як [6]:

- Симулятори польотів. Вони надають гравцеві можливість зануритися в світ авіації та відчути себе в ролі пілота різних типів літаків. Вони можуть включати різні варіанти, від реалістичних симуляторів військової авіації до простих аркадних ігор.
- Симулятори автомобілів. Ці ігри надають гравцеві можливість відчути себе за кермом різних видів автомобілів. Вони можуть бути реалістичними іграми, що відтворюють повністю реальні умови водіння, або аркадними іграми, де гравець може виконувати надзвичайні трюки.
- Симулятори життя. Ці ігри намагаються відтворити реалістичні умови життя, де гравець може стати часткою віртуального світу. Такі ігри можуть включати сімейне життя, кар'єру та ділові відносини.
- Симулятори спорту. Ці ігри надають гравцеві можливість відчути себе у ролі спортсмена та взяти участь в різних видах спорту. Це можуть бути футбольні, баскетбольні, гольфові та інші ігри.

- Симулятори ведення бізнесу. Ці ігри дозволяють гравцеві взяти на себе роль власника бізнесу та відповідати за управління всіма його аспектами, від виробництва та реклами до продажу продуктів та фінансових питань.
- Симулятори будівництва. Ці ігри дозволяють гравцеві взяти на себе роль архітектора та будівельника, проєктуючи та будуючи різні будівлі та споруди, включаючи житлові будинки, комерційні центри, парки та інше.

Найбільш відомими прикладами симуляторів є ігри The Sims, SimCity, RollerCoaster Tycoon, Microsoft Flight Simulator, Euro Truck Simulator, Farming Simulator та інші. Кожна з цих ігор має свої унікальні особливості та може зацікавити гравців з різними інтересами.

Розглянемо на прикладі гри-симулятора The Sims:

The Sims – це відома комп'ютерна гра, створена розробниками Maxis і EA Games, яка дозволяє гравцям керувати віртуальним життям персонажів – сімей, створювати їхні домівки та дивитися за їх потребами. Тобто, можна сказати, що гра The Sims є не лише симулятором реального життя, а й симулятором будівництва.

Гравці мають можливість створити свого власного персонажа, обрати його зовнішній вигляд, характеристики, заняття, взаємини з іншими персонажами тощо. Потім можна створити для персонажів будинок, обставити його меблями та декором. Головна мета гри – забезпечити персонажів всіма необхідними речами для щасливого життя, вирішувати їхні проблеми, розвивати їх особистість та досягати життєвих цілей.

Переваги гри:

- великий вибір можливостей і налаштувань для створення персонажів та їхнього життя;
- гра має велику кількість доповнень, які дозволяють розширити функціонал гри та додати нові можливості;
- гра цікава як для дітей, так і для дорослих.

Недоліки гри:

- гра може швидко надокучити, якщо в неї грати дуже довго;

- деякі гравці можуть відчувати відсутність головної мети в грі, оскільки The Sims – це симулятор життя, а не пригодницька гра з чітким сюжетом;
- час, який гравці витрачають на гру, може бути дуже довгим, особливо якщо граються кілька персонажів одночасно;
- гра може вимагати відносно потужного комп'ютера, особливо якщо користувачі встановлюють додаткові пакети розширення.

Взагалі, The Sims – це дуже цікава та розвиваюча гра, яка дозволяє гравцям створити власний світ та керувати життям своїх персонажів. Хоча гра має деякі недоліки, вона є дуже популярною серед гравців усього світу та є однією з найбільш відомих і успішних комп'ютерних ігор усіх часів (рис. 1.9).

Рисунок 1.9 – Інтерфейс відеогри The Sims

1.2.5 Стратегії

Жанр гри – стратегії є одним з найстаріших та найпопулярніших жанрів в історії відеоігор. Цей жанр включає в себе різноманітні ігри, де головним завданням гравця є планування та управління ресурсами з метою досягнення певної мети або перемоги над опонентами.

Основні характеристики жанру гри-стратегії, наведені в Таблиці 1.2 [7]:

Таблиця 1.2 – Характеристики жанру гри-стратегії

Характеристика	Опис
Управління ресурсами	Головним завданням гравця є планування та управління різними ресурсами, такими як гроші, люди, матеріали, енергія, щоб досягнути певної мети.
Розробка стратегії	Гравець повинен розробляти стратегію, яка дозволить йому досягти мети, використовуючи наявні ресурси та інформацію про опонентів.
Тактичні рішення	Гравець повинен приймати рішення на основі зміни обставин у грі, залежно від дій опонентів, зміни ресурсів та інших факторів.
Боротьба за ресурси	Гравець повинен конкурувати з іншими гравцями за різні ресурси, що може призвести до зміни стратегії та тактики.
Розвиток та підвищення рівня	Гравець повинен здійснювати розвиток своєї бази, підвищувати рівень військової техніки та військових дій, щоб збільшити свої шанси на перемогу.

Розглянемо на прикладі гри Starcraft II:

Starcraft II – це військова стратегія в реальному часі, яка відбувається у фантастичному світі. Гравець повинен керувати однією з трьох рас – людьми, зергами або протосами, розвивати свою базу та військову техніку, планувати та виконувати атаки на опонентів. У грі є декілька способів перемоги, таких як

військова перемога, досягнення певної кількості ресурсів або завоювання контрольних точок на карті (рис. 1.10).

Рисунок 1.10 – Інтерфейс відеогри Starcraft II

1.2.6 Гонки

У цьому жанрі гравцям пропонуються різноманітні автомобілі та траси, на яких вони можуть змагатися з іншими гравцями або комп'ютером.

Характеристики жанру гонок [5]:

- Швидкість: гонки – це жанр, в якому швидкість має надзвичайно велике значення. Гравці мають змагатися за швидкість, керуючи різними транспортними засобами.
- Геймплей. Гонки – це жанр, який вимагає від гравців навичок та вмінь управління транспортним засобом. Це може включати в себе використання різних додаткових функцій, таких як гальма, ручне управління та різні режими їзди.
- Різноманітність. Жанр гонок включає в себе різні типи гонок, такі як формула-1, ралі, дрифт та інші. Кожен з них має свої власні правила та характеристики, що додає глибини та цікавості жанру.
- Мультиплеєр. Багато гонок мають режим мультиплеєра, де гравці можуть змагатися між собою в онлайн-режимі або на одному екрані. Це дозволяє гравцям змагатися з реальними противниками, що підвищує рівень складності та цікавості гри.

- Кампанія. Багато гонок мають режим кампанії, який дозволяє гравцям змагатися в різних змаганнях та підвищувати атлетичні навички свого героя, отримувати нові транспортні засоби та покращувати їх характеристики.
- Візуальні ефекти. Багато гонок мають вражаючі візуальні ефекти, які дозволяють гравцям відчувати адреналін та занурюватися в гру. Це може включати в себе швидкі камери, ефекти руйнування та вражаючі ефекти погоди.

Наприклад, гра Need for Speed (рис. 1.11) – одна з найпопулярніших гонок у світі. Гравці можуть змагатися на різних трасах та використовувати різні види автомобілів, від спортивних машин до екзотичних автомобілів. Гра має вражаючі візуальні ефекти та гарну фізику руху. Основна мета гри полягає в тому, щоб стати кращим гонщиком, перемагаючи в заїздах, виконуючи завдання та отримуючи гроші для покращення свого автомобіля.

Серед переваг можна виділити:

- реалістична графіка та деталізовані автомобілі;
- велика кількість автомобілів та можливості їх покращення;
- захоплюючий геймплей та режим мультиплеєра;
- динамічний та енергійний саундтрек.

Але також, у гри є і свої недоліки:

- деякі ігри серії можуть бути занадто легкими або складними для деяких гравців;
- не завжди зручне керування автомобілем, особливо на деяких пристроях;
- відсутність нововведень та інновацій в деяких іграх серії, що може зробити їх менш привабливими для фанатів серії.

Рисунок 1.11 – Інтерфейс відеогри Need for Speed

1.2.7 Спортивні ігри

Цей жанр включає в себе широкий спектр ігор, які розробляються для відтворення різних видів спорту, від футболу та баскетболу до гольфу та тенісу.

Спортивні ігри є унікальними, оскільки вони поєднують у собі елементи реалістичного моделювання ігрових ситуацій та симуляції спортивних заходів. Гравці можуть відчути себе учасниками реальних спортивних змагань, де вони можуть проявити свої навички та стратегічні здібності для перемоги [6].

Однією з основних характеристик жанру гри "Спортивні ігри" є реалістичне відтворення фізики та механіки гри. Розробники стараються максимально точно відтворити рухи гравців, рух та поведінку м'ячів та інших спортивних реквізитів. Це дозволяє гравцям насолоджуватися максимально реалістичним досвідом спорту, а також використовувати свої навички та стратегічні здібності для перемоги.

Серед прикладів найпопулярніших ігор жанру "Спортивні ігри" можна виокремити таку популярну гру, як "FIFA" (рис. 1.12).

Рисунок 1.12 – Інтерфейс відеогри FIFA

1.2.8 Пригодницькі ігри

Цей жанр включає в себе ігри, які зазвичай мають складну сюжетну лінію, елементи дослідження, головоломки та інші елементи, які змушують гравців думати та вирішувати завдання [7].

Один з головних елементів пригодницьких ігор – це складний сюжет, який може розкриватися протягом усієї гри. Часто він пов’язаний з дослідженням різних місць та збором різних предметів. Наприклад, у грі "Uncharted (рис. 1.13), головний герой Натан Дрейк знаходиться на пошуках легендарного скарбу, який пов’язаний з його родовими корінням. У ході гри Натан має досліджувати різні місця, вирішувати головоломки та боротися зі злочинцями. Цей елемент дозволяє гравцям глибше занурюватися в історію гри та поступово розкривати таємниці.

Іншим елементом пригодницьких ігор є головоломки. Вони можуть бути дуже різними - від простих головоломок з логічними завданнями до більш складних, які вимагають вирішення складних ребусів та використання різних предметів.

Рисунок 1.13 – Інтерфейс відеогри Uncharted

1.2.8 Хоррори

Цей жанр відрізняється від інших жанрів тим, що його основна мета - створити атмосферу страху і неспокою у гравців. Хоррор-ігри прагнуть спровокувати в гравцях емоції страху, напруження та незручності, використовуючи засоби, такі як страхові сцени, музичний супровід, освітлення і звукові ефекти.

Характеристики жанру "Хоррори" включають в себе наступне [7]:

- створення атмосфери страху: одним з головних завдань хоррор-ігор є створення атмосфери страху, що має бути настільки переконливою, щоб гравці відчували страх і напруження від кожного кроку;
- інтенсивний геймплей: в іграх цього жанру часто використовують геймплей, який вимагає від гравців швидкого реагування на різні ситуації. Це може включати в себе швидкий біг від ворогів, шукачів та різних об'єктів, які становлять зогрозу, а також швидку реакцію на пастки і підступні обманки;
- створення напруження: хоррор-ігри також здатні створювати напруження, застосовуючи такі елементи, як темна атмосфера, зловісна музика, страшні звуки і освітлення;
- складна історія: ігри цього жанру мають часто складну історію, яка пропонує гравцям відкривати різні сюжетні лінії, розкривати таємниці та розв'язувати головоломки;
- необхідність стратегії: хоррор-ігри вимагають від гравців здатності до стратегічного мислення. Це означає, що гравці повинні ретельно планувати свої дії, враховуючи можливі пастки та ворогів;
- різноманітність ворогів та обманок: хоррор-ігри зазвичай містять різноманітних ворогів та обманки, які змушують гравців бути завжди насторожі;
- великий рівень непередбачуваності: хоррор-ігри можуть бути дуже непередбачуваними. Вони можуть змінюватися в залежності від дій гравців, що створює додаткові виклики для тих, хто грає;

– гра зі стресом: хоррор-ігри можуть бути важкими для людей, які страждають від тривожності, оскільки вони можуть спричинити стрес та тривогу;

– використання реалізму: хоррор-ігри можуть використовувати реалістичні ефекти та сцени, що додають реалізму та підсилюють ефект жаху.

Хоррор-ігри можуть бути різних типів. Деякі з них фокусуються на бійках та стрільбі, інші – на головоломках та розвідуванні. Дещо менш відомі ігри можуть пропонувати елементи рольової гри або відкритого світу. Однак, незважаючи на різноманітність, всі хоррор-ігри мають спільну мету: створити в гравцях атмосферу страху та напруження.

Одним з прикладів хоррор-ігри є "Outlast", в якій гравець грає в ролі журналіста, який здійснює розслідування в лікарні для психічно хворих , де трапляються дивні і страшні події. Гравець повинен збирати докази та відкривати таємниці, при цьому уникаючи зустрічей з небезпечними персонажами та загибелі. Гра відзначається відмінною графікою, звуковими ефектами та детально розробленим сюжетом (рис. 1.14).

Рисунок 1.14 – Інтерфейс відеогри Outlast

1.3 Наукове підґрунтя при розробці 2D гри

Ігрова індустрія дуже популярна в наш час, що дозволяє створювати розробникам все більш складніші за механікою та реалізацією ігри, які потребують високих характеристик комп’ютера. Але не всі люди полюбляють досліджувати

відкритий світ, як було наведено в прикладах вище, або зіштовхуватися зі сценами насильства. Тому за основу гри був взятий саме 2D простір.

1.3.1 Переваги 2D ігор

2D графіка дає можливість чітких та деталізованих графічних елементів, що робить інтерфейс більш зрозумілим та зручним для користувача.

Також, багато гравців почувають себе зв'язаними з класичним стилем 2D ігор, що дає їм можливість насолоджуватись грою у відчутті ностальгії. Але іноді це почуття може бути хибним, коли минуле сприймається надто романтично і у людини може виникнути почуття приємного суму за подією або проміжком часу, в якому вона ніколи не була [8]. Аналогічне відчуття може скластись у людей, граючи в 2D ігри, через їх найбільшу популярність в минулому та на початку зародження ігор як таких, через їх простоту та атмосферу невимушеності. Це призведе до збільшення цільової аудиторії гравців, адже навіть нове покоління не проти пограти в найпростіші 2D ігри.

Класичні 2D ігри з часом стали культовими та є символом геймінгу.

В 2D іграх гравець зазвичай має менше варіантів переміщення, оскільки гра працює в двовимірному просторі. Це робить управління героями більш простим та зрозумілим.

Для запуску 2D ігор не потрібно мати дуже потужний комп'ютер або ігрову консоль. Це дозволяє гравцям з менш потужним обладнанням насолоджуватись іграми та забезпечує більшу доступність.

1.3.2 Жанр 2D гри

За основу було взято пригодницький жанр. В грі присутні елементи челенжу, які допомагають гравцю зануритися у всесвіт гри та позбутися психологічної напруги. Суть, в основному, полягає в тому, що граючи за персонажа-дівчинку, потрібно долати різні перешкоди, що вимагає більшої концентрації та навичок для проходження рівня.

У грі відсутні елементи агресії та жорстокості (вбивств, насильства, тощо), які можуть викликати у гравця асоціації з жорстокістю у реальному світі. Існує декілька досліджень, які досліджують вплив насильства в іграх на психіку людини. Наприклад, одне з досліджень було проведене в 2015 році в Університеті Массачусетса, де вчені досліджували вплив насильницького відеоігрового контенту на агресивність гравців. В результаті дослідження було виявлено, що гравці, які грали в ігри з насильством, були більш агресивними, проявляли симптоми тривожності та депресивності в порівнянні з гравцями, які грали в ігри без насильства.

Висновки до розділу 1

Жанр платформерів є одним з найпопулярніших у світі ігор, особливо 2D платформерів. 2D ігри платформери пропонують унікальний геймплей, який поєднує скакання, перешкоди та розв'язування головоломок. Вони надають простий та доступний досвід гри, з великою кількістю рівнів та викликів. Їхні переваги включають високу мобільність, можливість створювати вражаючу графіку та можливості для творчої свободи в розробці гри. 2D платформери можуть бути дуже привабливими для розробників та гравців завдяки своїй простоті та захоплюючому геймплею.

2 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ КОМП'ЮТЕРНОЇ ГРИ

2.1 Платформа .NET та мова програмування C#

Платформа .NET та мова програмування C# є надзвичайно потужними інструментами для розробки різноманітних програм, включаючи 2D ігри.

Платформа .NET є розповсюдженою та широко використовуваною серед програмістів, оскільки надає ряд важливих переваг. Самою важливою складовою платформи .NET є її основа – середовище виконання Common Language Runtime (CLR). CLR забезпечує виконання коду, написаного на різних мовах програмування, включаючи C#. Він відповідає за управління пам'яттю, збірку сміття, безпеку та інші аспекти виконання програм [9].

Мова програмування C# є однією з найпопулярніших мов для розробки програмного забезпечення, включаючи ігри. Вона поєднує в собі простоту використання з потужними можливостями. C# є статично типізованою мовою, що означає, що помилки виявляються на етапі компіляції, що полегшує відлагодження програм. Вона також підтримує об'єктно-орієнтований підхід до програмування, дозволяючи організувати код у логічні блоки, що полегшує розуміння і підтримку програм.

Однією з ключових переваг платформи .NET є її широка підтримка. Вона має велику базу розробників, які активно використовують цю платформу і популярні мови програмування, такі як C#. Це означає, що для розв'язання будь-яких проблем або отримання підтримки завжди є велика спільнота, з якою можна обмінюватись досвідом, задавати питання і отримувати відповіді. Це забезпечує швидкий розвиток та покращення навичок програмування.

Однією з найважливіших переваг платформи .NET є її масштабованість [10]. Вона може бути використана для розробки як невеликих, так і великих проєктів. Тут можна створювати 2D ігри різної складності, використовуючи різні бібліотеки та фреймворки, доступні в середовищі .NET. Наприклад, програміст може використовувати MonoGame або Unity для розробки графічних компонентів власної

гри. Ці інструменти надають широкі можливості для роботи з графікою, анімацією, фізикою та багатьма іншими аспектами 2D гри.

Ще однією важливою перевагою платформи .NET є її крос-платформеність. Ви можете розробляти 2D гри на платформі .NET і запускати їх на різних операційних системах, таких як Windows, macOS та Linux. Це дозволяє досягти більшої аудиторії ігрових користувачів і розповсюджувати ваші гри на різних платформах без необхідності переписувати весь код з нуля.

Окрім цього, платформа .NET має багато інструментів для підтримки розробки ігор. Visual Studio, інтегроване середовище розробки, надає потужні можливості для створення, відлагодження та тестування програм на базі .NET. Крім того, існують численні сторонні бібліотеки та ресурси, які полегшують процес розробки ігор на платформі .NET [12].

Взагалі, платформа .NET та мова програмування C# є відмінним вибором для розробки 2D гри. Вони забезпечують потужність, швидкість розробки, масштабованість та крос-платформену підтримку. За допомогою .NET та C# можна створювати вражаючі, використовуючи різноманітні інструменти та ресурси, доступні в цьому середовищі. Незалежно від того, чи користувач новачок у розробці ігор чи досвідчений програміст, він знайдете все необхідне для успішної реалізації своєї ідеї гри на платформі .NET та за допомогою мови C#.

Однією зі значущих переваг платформи .NET та мови C# є їх середовище і наявність великої кількості сторонніх бібліотек, фреймворків та інструментів. Наприклад, можна використовувати бібліотеку XNA або MonoGame для розробки графічних компонентів і контролю вводу у 2D іграх. Ці бібліотеки надають широкі можливості для роботи зі звуком, анімацією, шейдерами та багатьма іншими аспектами гри. Крім того, можна використовувати бібліотеки, такі як Farseer Physics Engine, для реалістичного моделювання фізики у іграх.

Також варто зазначити, що платформа .NET та мова C# пропонують хорошу підтримку для розробки ігор з використанням шаблону архітектури Model-View-Controller (MVC) або Model-View-ViewModel (MVVM). Ці архітектурні підходи

дозволяють чітко розділити логіку гри від графічного представлення та даних, що спрощує розробку, тестування та модифікацію гри в майбутньому.

Навіть якщо програміст вже знайомий з іншими мовами програмування, перехід до C# буде досить простим. C# має схожий синтаксис з іншими популярними мовами, такими як Java або C++, що полегшує перехід і дозволяє використовувати ваші наявні знання в новому контексті.

Важливим аспектом розробки гри на платформі .NET та мові C# є підтримка різних інструментів для розробки, тестування та відлагодження коду. Один з найпопулярніших інструментів – Visual Studio, надає потужні можливості для створення, збирання та налагодження програмного коду.

Окрім Visual Studio, існує також ряд інших інтегрованих середовищ розробки, які підтримують розробку на платформі .NET та мові C#. Наприклад, Rider, Xamarin Studio та Visual Studio Code є популярними альтернативами з відкритим вихідним кодом, які надають багато зручних функцій для розробки гри.

Крім того, .NET і C# активно розвиваються і підтримуються Microsoft та громадою розробників. Це означає, що програміст буде мати доступ до оновлень, виправлень помилок та нових функцій, що поліпшують продуктивність та можливості платформи. Регулярні оновлення дозволяють залишатися впевненими, що гра буде сумісною з останніми версіями платформи та мови програмування (рис. 2.1).

Рисунок 2.1 – Логотип .NET

Мова програмування C# є однією з найпопулярніших мов програмування для розробки програмних продуктів, включаючи ігри. Вона має багато переваг і особливостей, які роблять її привабливим вибором для створення 2D гри.

Однією з ключових переваг C# є його простота і зрозумілість [13]. Мова була розроблена з метою спростити програмування і зробити код більш зрозумілим для розробників. Синтаксис C# логічний і схожий на інші мови програмування, такі як Java або C++, тому він відносно легко вивчається та розуміється. Це особливо корисно для початківців у галузі розробки ігор.

Ще одна перевага мови C# – це її масштабованість та підтримка об'єктно-орієнтованого програмування (ООП). Це означає, що ви можете створювати класи, об'єкти та ієрархії класів, що сприяє більш структурованому та модульному підходу до розробки гри. ООП дозволяє створювати повторно використовувані компоненти, полегшує управління кодом та забезпечує зручність у розробці та розширенні гри.

Крім того, C# має потужну систему типів і автоматичне управління пам'яттю [14], що допомагає запобігти багатьом типовим помилкам, пов'язаним з управлінням пам'яттю, таким як витоки пам'яті або некоректні посилання. Це спрощує процес розробки і знижує ризик виникнення помилок, особливо великих і складних проєктів.

Однією з додаткових переваг мови C# для розробки 2D ігор є наявність розширених функцій інтегрованих розробничих середовищ (IDE) та інструментів, що спрощують процес розробки. Наприклад, Visual Studio, що є основним IDE для розробки на платформі .NET, надає розширений набір інструментів для аналізу коду, налагодження, профілювання та автоматизованого тестування. Це допомагає виявляти та виправляти помилки, покращувати продуктивність і оптимізувати роботу гри.

Ще однією перевагою мови програмування C# для розробки 2D ігор є наявність розширених фреймворків і бібліотек, які спеціально розроблені для роботи з графікою, анімацією та фізикою. Наприклад, Unity – один з найпопулярніших фреймворків для розробки ігор, що базується на мові C#. Unity

надає широкий набір інструментів та функцій, які спрощують створення 2D гри, включаючи системи фізики, анімації, керування введенням та інші.

Додатковою перевагою мови C# є її інтеграція з іншими інструментами та технологіями Microsoft, такими як Windows Forms, WPF (Windows Presentation Foundation) і UWP (Universal Windows Platform). Це дає можливість створювати інтерфейс користувача для гри, використовуючи широкий набір елементів керування та стильових можливостей.

Однією з сильних сторін мови C# є її продуктивність. Через вбудоване управління пам'яттю та оптимізації, C# може забезпечувати високу продуктивність і швидкодію. Це особливо важливо для ігор, де потрібна реалізація складних алгоритмів, швидкої графіки та фізики.

Узагалі, мова C# є гнучкою, масштабованою та потужною мовою програмування, яка має багато переваг для розробки 2D ігор. Вона поєднує простоту та зрозумілість синтаксису, підтримку об'єктно-орієнтованого програмування, широкий вибір фреймворків та бібліотек, інтеграцію з платформою .NET та іншими інструментами. Вибір мови C# для розробки 2D гри дозволить вам ефективно працювати над проєктом, забезпечуючи швидкість, якість та гнучкість розробки.

2.2 Середовище розробки Visual Studio

Середовище розробки Visual Studio є одним з найпопулярніших і потужних інструментів для створення програмного забезпечення, включаючи розробку 2D ігор. Воно надає широкий набір функціональних можливостей та інструментів, які роблять його перевагою перед іншими середовищами розробки.

Перш за все, Visual Studio забезпечує потужну підтримку мови програмування C#, яка є однією з найпоширеніших мов для розробки ігор. Інтегрована підтримка C# в Visual Studio дозволяє легко створювати, редагувати, налагоджувати та відлагоджувати код гри. В цьому середовищі можна скористатися

багатими функціональними можливостями мови C#, включаючи об'єктно-орієнтоване програмування, делегати, події та багато іншого [12].

Visual Studio надає потужні інструменти для управління проєктом та збірки. Дозволяє створювати різні проєкти, включаючи бібліотеки класів, модулі, компоненти та інше, що допомагає організувати проєкт розробки 2D гри. Visual Studio автоматично стежить за залежностями між файлами, виконує збірку та забезпечує простий спосіб керування вашим проєктом.

Один з великих плюсів Visual Studio полягає в його потужній системі налагодження. Тут можна крок за кроком відстежувати виконання написаної програми, перевіряти значення змінних, використовувати точки зупинки та багато іншого. Це дозволяє ефективно відлагоджувати гру, виявляти та виправляти помилки та забезпечувати якісний розвиток запланованого проєкту.

Крім того, Visual Studio надає інтеграцію з різними фреймворками та бібліотеками, які широко використовуються для розробки ігор. Наприклад, тут можна легко інтегрувати Unity або інші фреймворки у проєкт Visual Studio. Це дає можливість використовувати потужні функції цих фреймворків, такі як системи фізики, обробка введення, робота з графікою та звуком, для створення 2D гри.

Не можна не згадати також про активну спільноту розробників, яка підтримує Visual Studio. Існує безліч ресурсів, форумів, підручників, блогів та інших джерел, де можна отримати допомогу, відповіді на запитання та поради від досвідчених розробників.

Загалом, використання Visual Studio для розробки 2D гри має багато переваг. Від широкого набору функціональних можливостей та інструментів до потужної системи налагодження та інтеграції з різними фреймворками – це середовище розробки надає всі необхідні інструменти для створення високоякісних 2D ігор. Не дивно, що Visual Studio залишається популярним вибором серед розробників у галузі геймдеву (геймдев – Game development) (рис. 2.2).

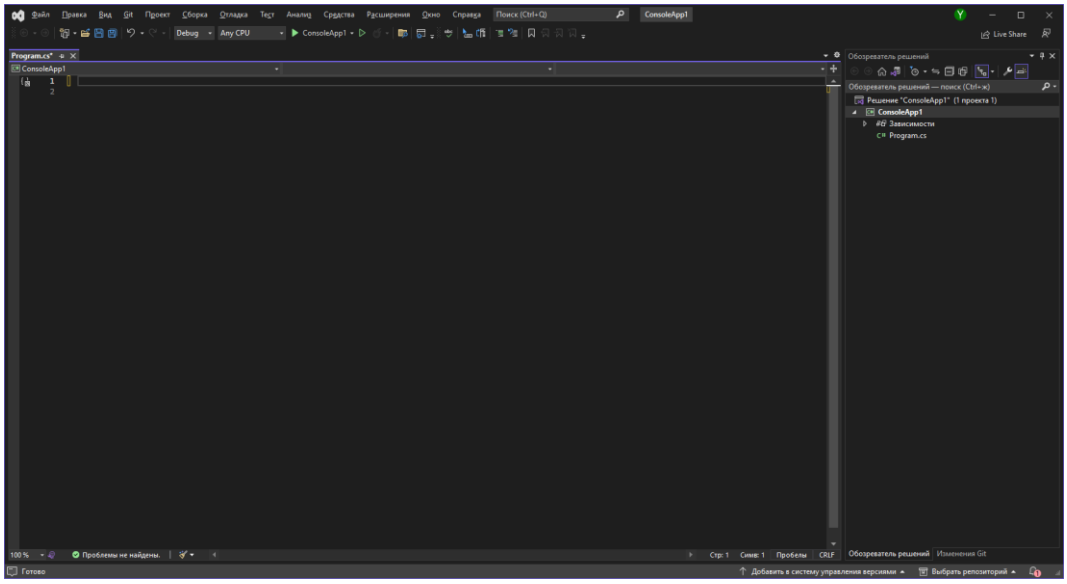

Рисунок 2.2 – Інтерфейс Visual Studio

2.3 Двигун для розробки відеоігор Unity

Unity є одним з найпопулярніших двигунів для розробки ігор, і має численні переваги, особливо при створенні ігор. Ось деякі з найважливіших переваг Unity [15, 16]:

1) крос-платформеність: Unity дозволяє розробляти ігри, які працюють на різних платформах, включаючи Windows, macOS, Linux, iOS, Android та інші. Це дозволяє залучити більше користувачів і розповсюджувати гру на різних пристроях без необхідності переписувати код;

2) готові компоненти та ресурси: Unity надає багато готових компонентів, ресурсів та інструментів, які полегшують створення гри. Наприклад, можна використовувати готові 2D фізичні двигуни, анімаційні системи, системи частинок, системи колізій та багато іншого. Це значно прискорює процес розробки, зменшує зусилля та сприяє створенню професійних графічних ефектів;

3) легкість використання: Unity має дружній інтерфейс користувача, який дозволяє легко розуміти та працювати з різними аспектами розробки гри. Інтерфейс Unity складається зі зручних панелей, вікон та редакторів, які дозволяють вам візуально налаштувати графічні об'єкти, налаштувати фізику, створювати скрипти, управляти сценами та багато іншого;

4) мови програмування: Unity підтримує кілька мов програмування, включаючи C#, яка є однією з найпоширеніших мов у галузі розробки ігор. Використання C# дозволяє використовувати потужні можливості цієї мови, з легкістю створювати складну логіку гри, маніпулювати об'єктами та взаємодіяти з іншими системами Unity;

5) велика спільнота розробників: Unity має широку спільноту розробників, яка активно співпрацює, надає поради, розв'язує проблеми та надає велику кількість ресурсів, які допомагають розвиватися та вдосконалювати навички в розробці гри. Форуми, блоги, вебінари та документація Unity допоможуть вирішити питання та знайти рішення навіть для найскладніших завдань.

Оглядаючи всі ці переваги, Unity є відмінним вибором для розробки власної гри. Він надає потужність, гнучкість та різноманітність інструментів, які допоможуть реалізувати свою ідею та створити захоплюючий геймплей для аудиторії. З Unity можна швидко розпочати розробку, ефективно працювати з графікою, фізикою, анімацією та іншими аспектами гри, що дозволить створити вражаючу гру (рис. 2.3).

Рисунок 2.3 – Логотип Unity

Висновки до розділу 2

Рушій Unity є потужним інструментом для створення ігор, особливо для 2D та 3D графіки. Він надає широкі можливості для створення реалістичних графічних об'єктів, фізики, анімації та іншого функціоналу, що допомагає

розробникам створювати захоплюючі ігрові світи. Unity також має велику спільноту розробників, розширення та підтримку, що полегшує процес розробки і дозволяє досягти бажаних результатів.

Середовище розробки Visual Studio є потужним інтегрованим середовищем, що надає розробникам широкі можливості для написання, налагодження та управління кодом. Воно забезпечує зручну роботу з мовою програмування C# та іншими мовами, автодоповнення, відступи, контекстну довідку та багато інших корисних функцій. Visual Studio також підтримує інтеграцію з Unity, що дозволяє зручно розробляти ігри безпосередньо в середовищі розробки.

Усі три компоненти - мова програмування C#, рушій Unity та середовище розробки Visual Studio - взаємодіють між собою і створюють потужну комбінацію для розробки ігор.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ 2D ГРИ

3.1 Опис ідеї та створення плану розробки

Для виконання поставленої задачі було прийнято рішення створити гру-платформер, в якому гравець керує персонажем і прокладає шлях через різні рівні, збираючи кристали і досягаючи дерева для переходу на наступний рівень. Гра має головне меню, в якому гравець може обрати рівень, а також кнопку "Reset", яка скидає прогрес гравця.

Для реалізації задуманої ідеї було створено план розробки гри:

- розробка головного меню;
- створення графічного інтерфейсу головного меню;
- додавання кнопок для вибору рівня та скидання прогресу;
- налаштування логіки кнопок для переходу до обраного рівня або скидання прогресу;
- розробка першого рівня;
- створення графічних елементів рівня, таких як платформи, вода та дерево;
- налаштування руху персонажа по платформам;
- додавання фізики для зіткнень персонажа з водою та збирання кристалів;
- логіка завершення рівня, коли персонаж доходить до дерева;
- розробка другого рівня;
- створення платформ, що рухаються та падаючих платформ;
- додавання елемента челленджу та збору кристалів на цьому рівні;
- налаштування лічильника кристалів та оновлення його стану;
- розробка третього рівня;
- встановлення драбини для збору кристалів у верхній частині рівня;
- поліпшення графіки та анімацій;
- додавання деталей до графічних елементів рівнів та персонажа;

- встановлення анімацій для персонажа, включаючи рух, стрибки та анімацію в спокійному стані персонажу.

3.2 Загальні поняття до створення сцени

3.2.1 Імпорт ресурсів

Unity надає можливість імпортувати різноманітні ресурси, такі як графіка, звуки, моделі, анімації та інші, для використання в проєкті. Цей процес дозволяє розробникам додавати власні контент і різні елементи до гри, роблячи її унікальною та привабливою.

Заходимо в папку, в якій містяться готові ресурси та обираємо файл з розширенням .unitypackage, 2 рази тиснемо на файл (рис. 3.1).

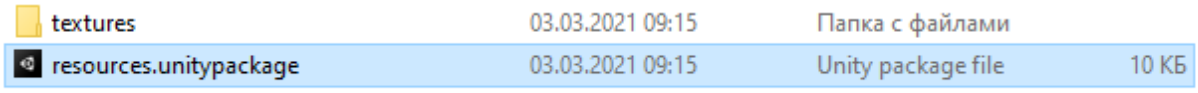

Рисунок 3.1 – Приклад готової збірки ресурсів

Висвітиться вікно, в якому обираємо потрібні файли, та імпортуємо їх в проєкт (рис. 3.2).

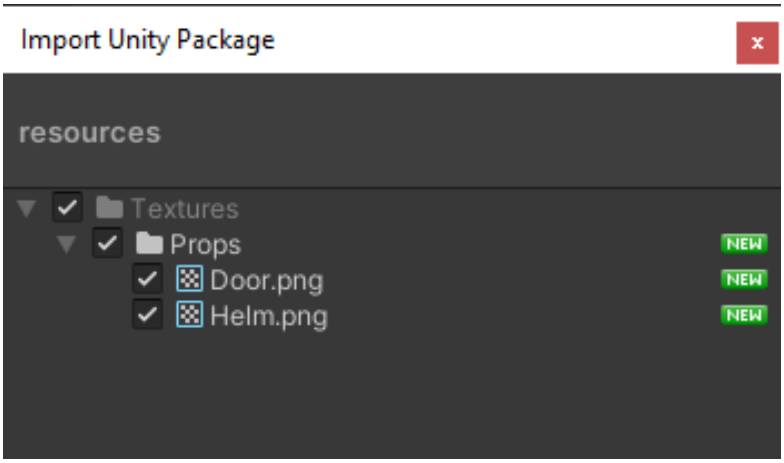

Рисунок 3.2 – Імпорт даних

Частіше приходиться використовувати інший спосіб. Обираємо папку, в якій містяться потрібні файли та спрайти, виділяємо їх (рис. 3.3).

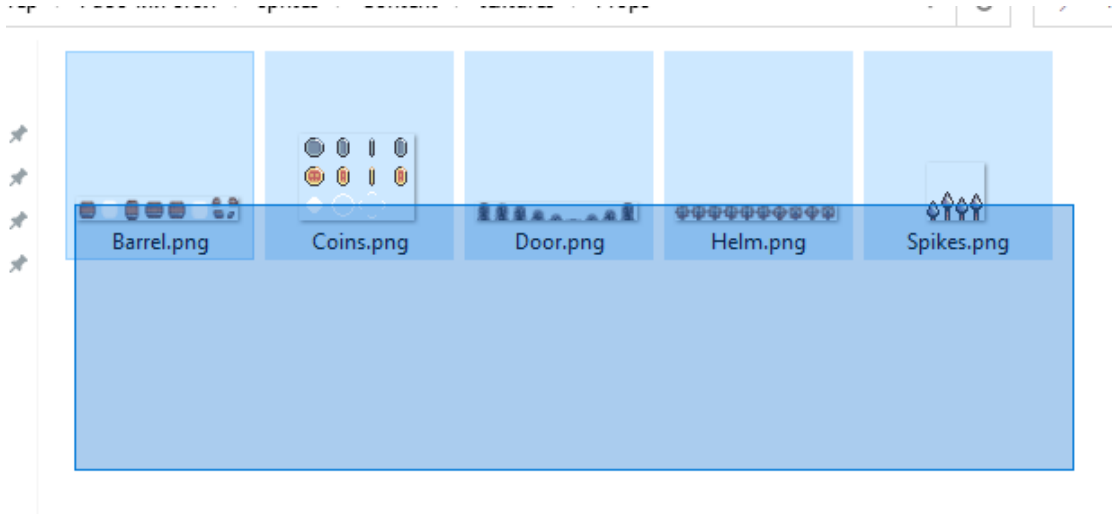

Рисунок 3.3 – Дані в вигляді спрайтів

І перетаскуємо їх в уже створену папку в Unity (рис. 3.4).

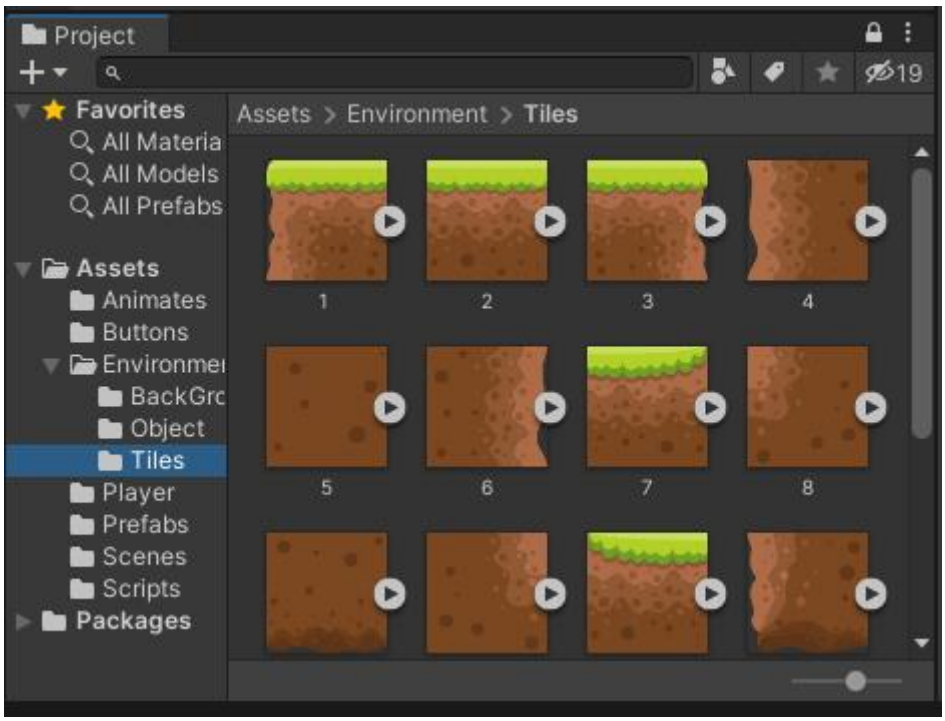

Рисунок 3.4 – Ієрархія об’єктів в Unity

Є декілька підходів до того, як зберігати ресурси:

- розділення по типам даних (зберігаємо ресурси одного типу в одній дерикторії);
- зберігання різних типів ресурсів в одній папці, пов’язаною з приналежністю до якогось об’єкта.

В проєкті буде використовуватись змішаний тип зберігання ресурсів для більшої зручності.

3.2.2 Налаштування текстур

Налаштування текстур є важливою складовою розробки гри в Unity, оскільки вона має значний вплив на візуальний аспект гри, її продуктивність та розмір файлу.

Текстури можуть бути використані для різних цілей. Наприклад, для опису нормалей або спрайтів. Тому потрібно обирати тип текстури.

Виділяємо об’єкт, який хочемо налаштувати. Якщо спрайт складається з великої кількості анімованих рисунків, в графі Sprite Mode треба обрати Multiple (рис. 3.5).

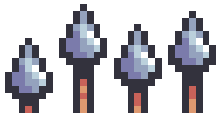

Рисунок 3.5 – Спрайт, який складається з анімованих рисунків

Якщо ж спрайт лише один – то single (рис. 3.6).

Рисунок 3.6 – Приклад одиночного спрайту

Графа Pixels Per Unit визначає кількість пікселів в одному юніті. Координати в сцені визначаються в юнітах, і тут визначається скільки пікселів із текстури буде в одному юніті.

Кожен спрайт, який є на екрані являє собою не просто двовимірну картинку, а меш. Тобто, плоский полігональний 3D об’єкт, на який натягнута відповідна текстура. Параметром Mesh Type можна указати якої форми буде меш заданого об’єкту (рис. 3.6).

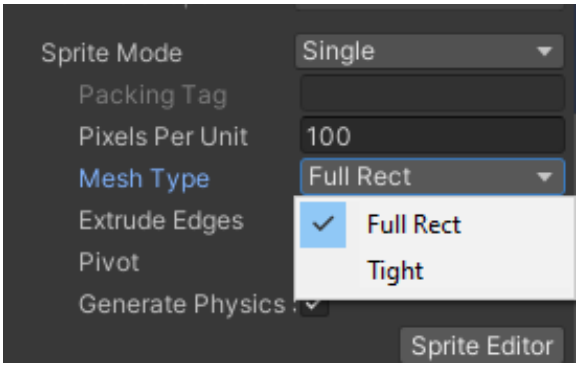

Рисунок 3.6 – Графа Mesh Type з випадаючим полем

Tight – використовується для того, щоб програма згенерувала меш, який є близьким до текстури спрайту, тобто «обтікає» його. Обираючи цей режим, зменшується кількість ресурсів на відмальовування невидимих пікселів (рис. 3.7).

Рисунок 3.7 – Вплив режиму Tight на спрайт

Full Rect – не намагається створювати складний полігональний меш, а лише вимальовує прямокутник навколо спрайту, у якого є всього 4 вершини (рис. 3.8).

Рисунок 3.8 – Вплив режиму Full Rect на спрайт

Тому, для оптимізації гри, було обрано режим Tight.

Графа Extrude Edges визначає, скільки пікселів залишається по краях згенерованого меша (рис. 3.9).

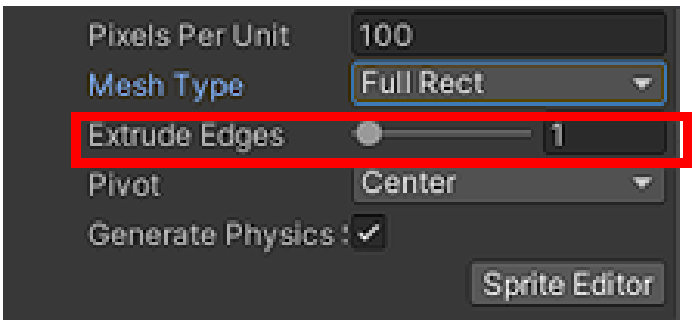

Рисунок 3.9 – Графа Extrude Edges

Точка прив'язки текстури (Pivot) є важливою функцією в Unity, яка визначає, як текстура або графічний елемент буде позиціоновано та масштабовано в грі [17]. Вона впливає на вигляд та поведінку об'єктів під час розміщення текстур та графіки на ній. Вона є дуже важливою складовою і її корисно змінювати, наприклад, для персонажу гри, щоб точка прив’язки була знизу персонажа (рис. 3.10).

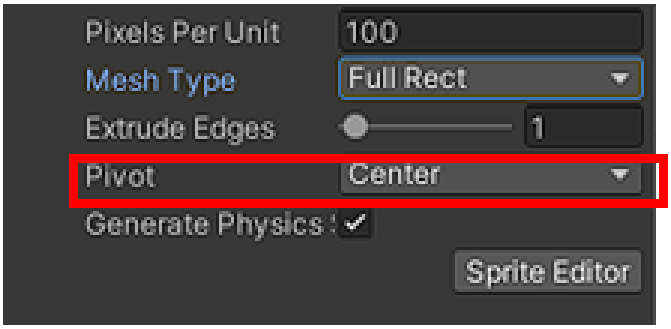

Рисунок 3.10 – Точка прив'язки текстури Pivot

Не менш важливою є графа Generate Physics. Вона потрібна для того, щоб по границям спрайту вона згенерувала полігон для фізики. Це потрібно, наприклад для відловлювання зіткнення з різними об’єктами, або взаємодіяти з фізикою світу (рис. 3.11).

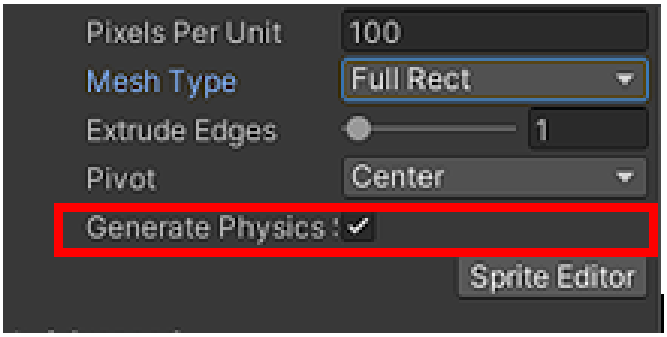

Рисунок 3.11 – Графа Generate Physics

3.3 Створення Головного меню до гри

Головне меню являє собою вікно привітання гравця, та дає йому можливість обрати перший рівень, або зкинути прогрес гри кнопкою Reset. У випадку, якщо гравець пройшов перший рівень, то в головному меню відкривається доступ до другого рівня. Якщо пройшов другий рівень, то до третього. Це зроблено з метою поступового проходження гри – з менш складного рівня, до найскладнішого, а також з метою ознайомлення гравця з механікою гри та звикання до управління персонажем.

В головному меню було створено такі об’єкти (рис. 3.12):

– Камера (Main Camera). Камера визначає перспективу та огляд головного меню гри. Вона відповідає за відображення того, що відбувається на сцені та показує гравцю зображення. Камера може мати різні налаштування, такі як поле огляду, позицію, орієнтацію, налаштування освітлення та інші.

– Задній фон (BG). Задній фон відповідає за візуальне оформлення головного меню гри. Він представляє собою зображення, яке створює атмосферу та

відображають тематику гри. Задній фон може мати властивості налаштування, такі як розмір, масштаб, прозорість та інші.

– Система подій (Event System). Система подій відповідає за обробку взаємодії користувача з головним меню гри. Вона забезпечує можливість натискання кнопок. Система подій дозволяє встановлювати реакції на події та керувати переходами між різними елементами головного меню.

– Контролер (Controller). Контролер відповідає за керування взаємодією між елементами головного меню та виконання дій на основі вибору користувача.

– Канвас (Canvas). Канвас є центральним елементом головного меню гри, який зв'язує всі інші об'єкти разом. Він визначає простір, на якому розміщуються кнопки, текст та зображення. Канвас також відповідає за налаштування масштабування та положення елементів, щоб забезпечити адаптивність до різних розмірів екрану.

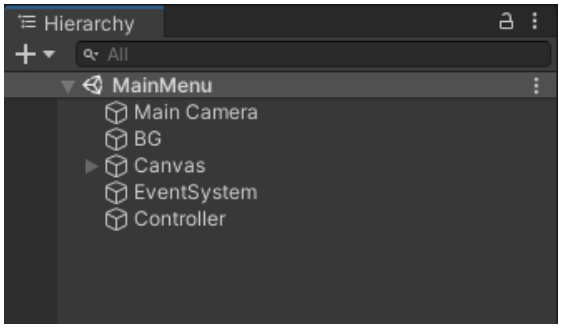

Рисунок 3.12 – Ієрархія головного меню гри

Ці об'єкти разом створюють головне меню гри, яке надає гравцю можливість взаємодіяти з різними функціями та налаштуваннями. Кожен об'єкт виконує свою роль в системі та співпрацює з іншими елементами, створюючи зручне та ефективне головне меню для користувача (рис. 3.13).

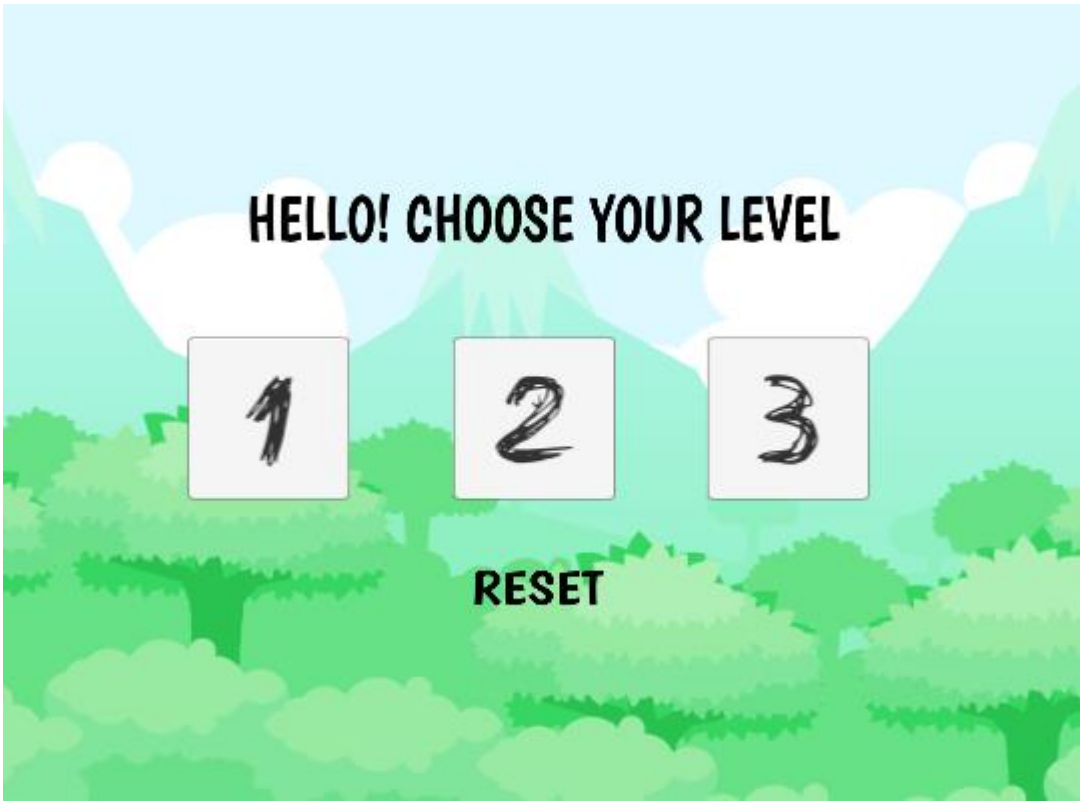

Рисунок 3.13 – Головне меню гри

3.4 Структурування проєкту та створення ігрових механік

Основною метою рівня гри було створити непередбачувані ситуації та завдання для гравця, а також забезпечити надзвичайну геймплейну динаміку. Для досягнення цього було використано кілька ключових функцій для персонажа. Перш за все, персонаж мав можливість стояти на землі, що забезпечувало стійкість та контроль під час руху. Для подолання перешкод, була додана можливість перестрибувати через воду, яка виступала як небезпечний елемент середовища. Однак, якщо персонаж доторкається до води, він помирає та респавниться на початок рівня, забезпечуючи відчуття відповідальності за власні дії.

Додатковим функціоналом для персонажа було введено подвійний стрибок, що дозволяє долати більші відстані та здійснювати складніші маневри в грі. Це розширює можливості гравця та додає елемент стратегії при проходженні рівнів.

Окрім цього, у грі був доданий лічильник кристалів, який оновлюється при смерті персонажа або переході на новий рівень. Цей елемент геймплею стимулює

гравця зібрати якомога більше кристалів та виконати завдання рівня з найкращим результатом.

Для цього на сцену було додано відповідні елементи, які допоможуть створити задумане (рис. 3.14).

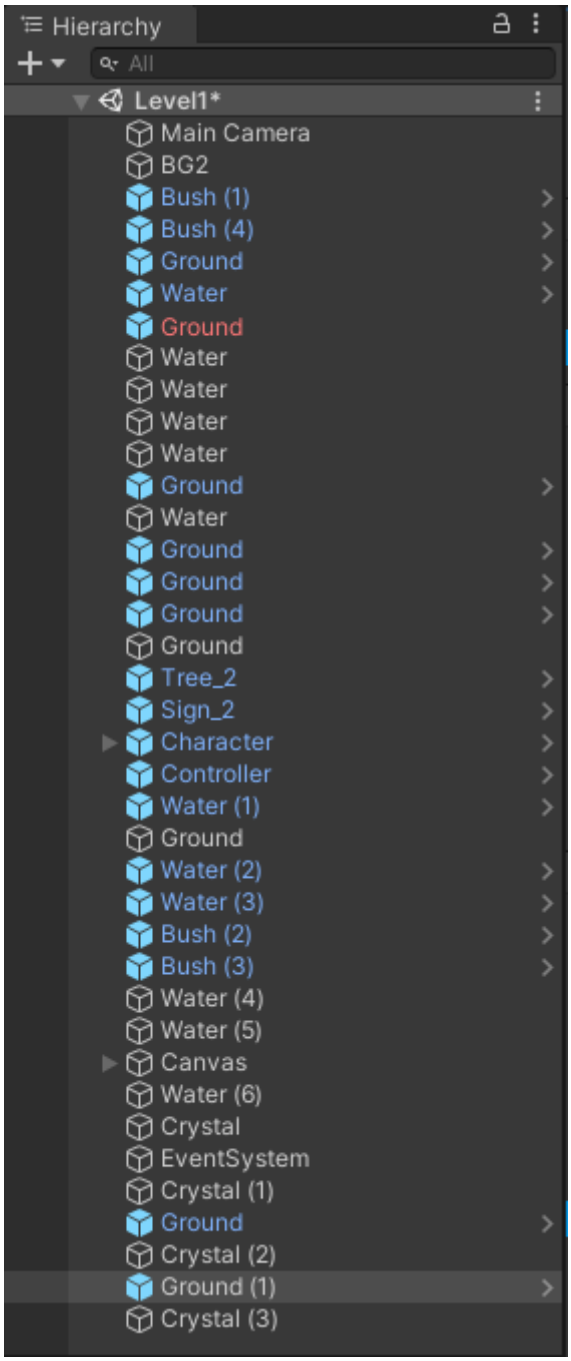

Рисунок 3.14 – Ієрархія елементів першого рівня

За блок землі, на якому і може стояти персонаж, відповідає елемент Ground. До нього було додано такі елементи, як Sprite Renderer та Box Collider 2D.

Компонент Sprite Renderer є одним з основних компонентів, що використовується в Unity для візуального відображення спрайтів. Цей компонент дозволяє прикріпити спрайт до об'єкта і відтворити його на сцені гри.

Основна роль Sprite Renderer полягає в установці відповідного спрайту для відображення об'єкта на екрані. Властивість Sprite Renderer містить зображення, яке використовується для відтворення об'єкта. Можна вибрати спрайт з ресурсів або завантажити його зовнішньо [18].

Крім того, Sprite Renderer має різні властивості, які дозволяють контролювати відображення спрайту. Наприклад, можна встановити прозорість, колір або масштаб спрайту. Ці властивості дозволяють налаштовувати вигляд спрайту відповідно до потреб гри (рис. 3.15).

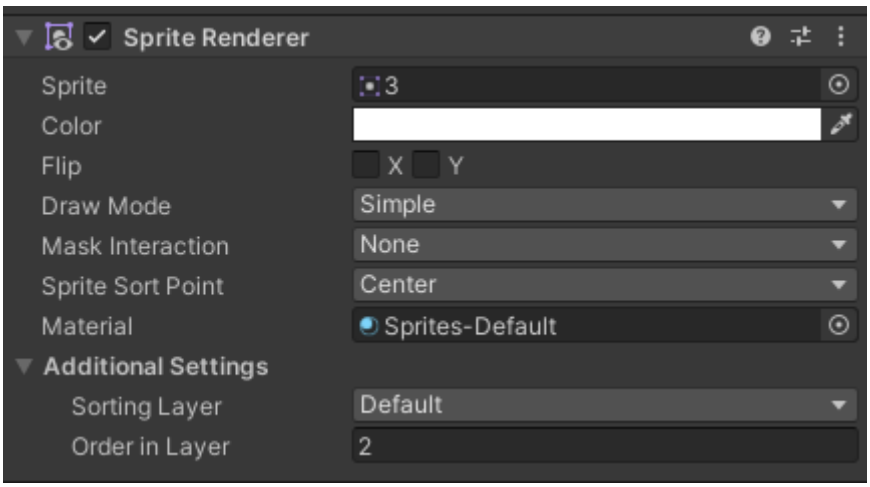

Рисунок 3.15 – Налаштований компонент Sprite Renderer для кожного блоку землі

Компонент Box Collider 2D використовується для задання колізійного об'єкта в 2D просторі. Він додає прямокутний колайдер до об'єкта, що визначає його межі і форму для взаємодії з іншими об'єктами у грі.

Box Collider 2D має ряд властивостей, які дозволяють налаштовувати колізійний об'єкт. За допомогою нього можна встановити розмір та положення колайдера, його обробку подій зіткнень, фізичні властивості та багато іншого.

Одна з ключових особливостей Box Collider 2D полягає в тому, що він використовується для визначення фізичних взаємодій об'єктів. Наприклад, якщо персонаж доторкається до колайдера, можна встановити відповідні дії, такі як відтворення різних анімацій, зупинку гравця або спричинення інших ефектів.

Компонент Box Collider 2D використовується для реалізації реалістичних колізій між об'єктами у 2D грі. Він додає важливий аспект фізичної взаємодії між об'єктами та дозволяє програмувати поведінку об'єктів під час зіткнень [19] (рис. 3.16).

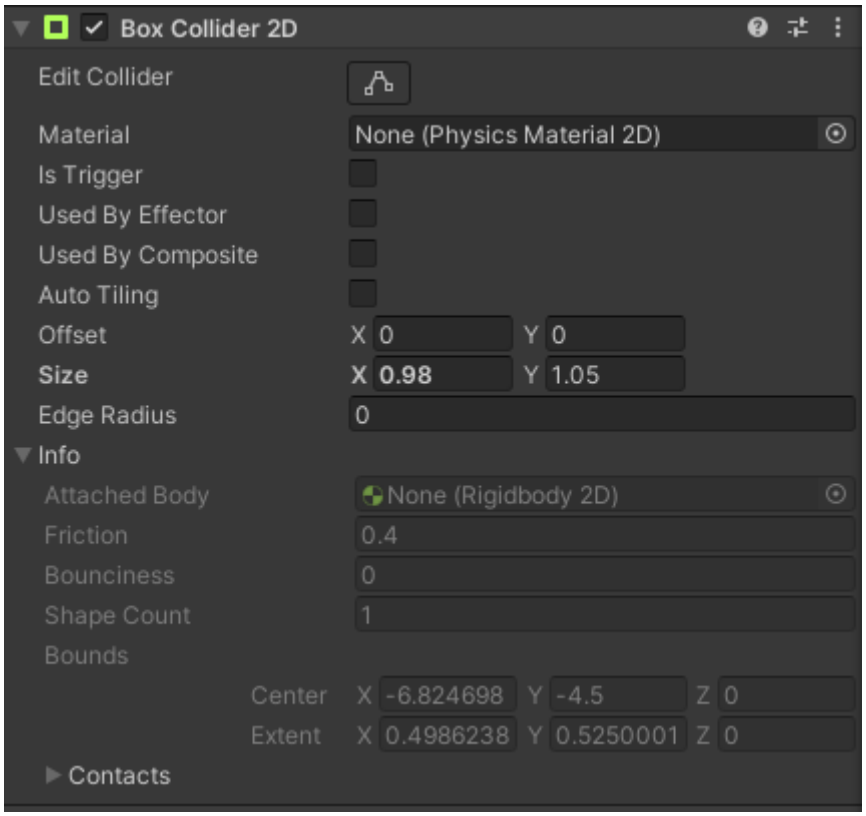

Рисунок 3.16 – Налаштований компонент Box Collider 2D для кожного блоку землі

Використання шарів в Unity є важливою та потужною функцією при розробці ігрових проєктів. Шари дозволяють організовувати та контролювати різні аспекти гри, такі як візуалізація, колізії, взаємодія об'єктів та інші важливі елементи геймплею.

Одним з основних аспектів використання шарів є візуалізація об'єктів у сцені. За допомогою шарів можна призначити об'єктам конкретні графічні налаштування, такі як текстури, матеріали та інші візуальні ефекти. Це дозволяє створювати різноманітні елементи гри з унікальним виглядом та атмосферою, що сприяє залученню гравців та покращує їхнє враження.

Крім того, шари використовуються для налаштування колізій об'єктів у грі. Колізії визначають зіткнення та взаємодію об'єктів у грі, що є ключовим аспектом для правильної фізичної взаємодії між об'єктами та їхньої реакції на дії гравця. Використання шарів дозволяє призначати об'єктам різні колізійні шари, які визначають, які об'єкти можуть взаємодіяти між собою, а які – ні. Це робить можливим створення складних систем колізій та реалістичного фізичного моделювання у грі [20].

Ще одним важливим аспектом використання шарів є керування взаємодією об'єктів та різних систем гри. Завдяки шарам можна налаштовувати взаємодію об'єктів з різними системами, такими як система подій та інші. Це дозволяє створювати комплексні та динамічні ігрові ситуації, де різні об'єкти взаємодіють між собою залежно від їхніх шарів [20].

Також шари можуть використовуватись для управління видимістю об'єктів у сцені. Завдяки шарам можна приховувати або відображати об'єкти на певних етапах гри або в певних умовах. Це дозволяє створювати динамічні зміни в грі, включаючи анімації, зміни рівнів деталізації та інші візуальні ефекти (рис. 3.17).

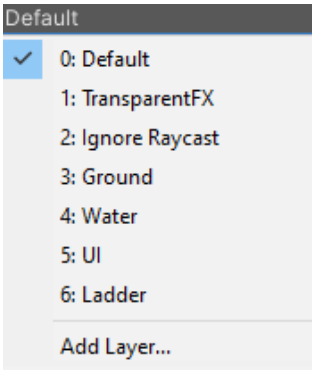

Рисунок 3.17 – Використовувані шари в 2D грі

Тож, для об’єкту землі, було створено та обрано шар Ground.

Ті ж самі маніпуляції були пророблені і для елементу води – Water, але замість шару Ground, був створений та обраний шар Water.

Не менш важливим є елемент кристалу (Crystal), який додає до гри елемент челелнджу. До елементу Crystal було додано компонент Sprite Renderer та Circle Collider 2D.

Circle Collider 2D, як і з його назви, представляє коллайдер у формі кола. Він має один основний параметр – радіус. Радіус визначає розмір коллайдера і визначає, яка область вважається колізійною. Все, що перетинає цей коллайдер, вважається колізійним [21].

Одна з відмінностей між Circle Collider 2D та Box Collider 2D полягає у формі коллайдера. Circle Collider 2D обмежений круглою формою, тоді як Box Collider 2D може мати прямокутну форму з різними співвідношеннями сторін.

Ще одна важлива різниця полягає в ефективності обчислень. Через свою просту форму, Circle Collider 2D використовує менше ресурсів для обчислення колізій, особливо коли має справу з багатьма об'єктами. З іншого боку, Box Collider 2D може бути більш ресурсоємним, оскільки потребує більше обчислень для розрахунку колізій залежно від форми та орієнтації коллайдера [22] (рис. 3.18).

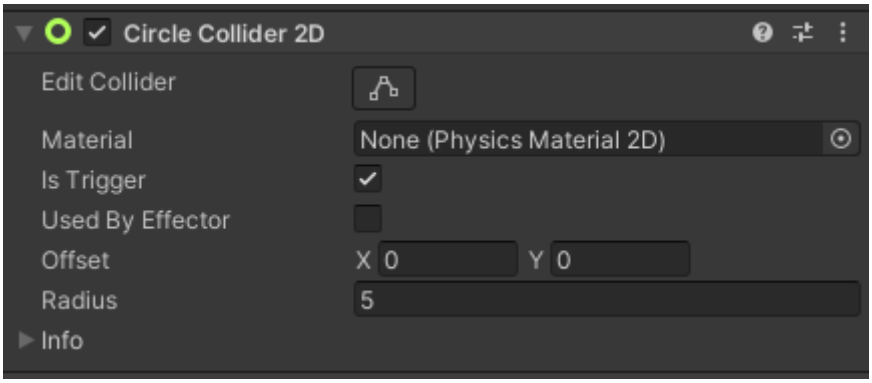

Рисунок 3.18 – Налаштований компонент Circle Collider 2D для кожного кристалу в грі

Графа "Is Trigger" є важливою частиною компонентів Circle Collider 2D та Box Collider 2D в Unity. Ця графа визначає, чи колайдер повинен функціонувати як "тригер" або "твердий об'єкт" [23].

Коли графа "Is Trigger" вимкнена, колайдер вважається твердим об'єктом. Це означає, що колайдер буде виявляти зіткнення з іншими колайдерами та взаємодіяти з ними, блокуючи рух об'єктів. Якщо два тверді об'єкти зіштовхнуться, вони будуть взаємодіяти фізично, змінюючи свої швидкості та рух.

Коли графа "Is Trigger" увімкнена, колайдер вважається тригером. Тригери не блокують рух об'єктів, але вони викликають подію зіткнення, коли інший колайдер перетинає їх область. Це дозволяє реагувати на зіткнення без фізичної взаємодії.

В даному випадку, використання графи "Is Trigger" викликатиме відловлювання зіткнення персонажу з кристалом, спрацювання скрипта для знищення кристалу, з яким пересічеться персонаж, та рахуванням уже здобутих кристалів в спеціальному лічильнику.

Для реалізації ідеї лічильника, було створено новий Canvas та додано до нього як дочірні об’єкти – Image та Text.

Об’єкт Image складається з, безпосередньо, картинки самого кристала та скрипта, який відповідає за додавання зібраних кристалів у вигляді цифри. Зв’язуючим компонентом є об’єкт Text, який відображає в собі полічену кількість зібраних кристалів.

Також було додано рухому платформу, яка підвищує рівень складності. За платформу в ієрархії відповідає об’єкт Crate. До нього було додано скрипт, який дозволяє платформі рухатись за заданою траєкторією та з певним інтервалом часу, та компонент Box Circle Collider 2D.

Додатковим елементом челенджу було додано падаючу платформу, яка через певний інтервал часу повертається на своє положення. Щоб реалізувати задуману ідею, було обрано звичайний спрайт, який використовувався для землі (Ground), щоб гравець візуально не зміг відрізнити відмінності від звичайної платформи

землі. Але на відміну від попередніх, цей об’єкт Ground має компонент Rigidbody 2D.

Компонент Rigidbody 2D є одним з ключових елементів фізичної симуляції в Unity для 2D грі. Він дозволяє об'єкту взаємодіяти з фізичним середовищем, враховуючи закони фізики, такі як гравітація, колізії та рух.

Основна функція компонента Rigidbody 2D – надати об'єкту масу, впливати на його рух та розраховувати фізичні колізії з іншими об'єктами у сцені. При додаванні компонента Rigidbody 2D до об'єкту, він отримує фізичні властивості, які дозволяють змінювати його поведінку відповідно до законів фізики.

Деякі з основних властивостей, які можна налаштовувати на компоненті Rigidbody 2D, включають:

- маса (Mass): визначає вагу об'єкта та його вплив на інші об'єкти у сцені. Об'єкти з більшою масою виявляють більшу силу під час зіткнень;
- гравітація (Gravity Scale): встановлює силу гравітації, що діє на об'єкт. За замовчуванням, значення 1 відповідає нормальній силі гравітації, а значення 0 вимикає гравітацію;
- кінематика (Is Kinematic): якщо встановлено значення "так", об'єкт не буде реагувати на фізичні сили у сцені та не буде впливати на інші об'єкти. Це корисно для статичних об'єктів, таких як платформи або стіни;
- демпфування (Linear Drag, Angular Drag): коефіцієнти демпфування визначають, як швидко об'єкт втрачає швидкість під час руху або обертання. Вони використовуються для моделювання опору середовища або повітряного опору;
- обмеження руху (Constraints): дозволяє обмежувати рух об'єкта по осі X або Y, забороняючи його рухатися у визначеному напрямку. Це корисно, коли потрібно обмежити рух об'єкта на певну площину або вісь.

Компонент Rigidbody 2D використовується для надання об'єкту фізичних властивостей та реалістичної поведінки. Він дозволяє персонажу реагувати на гравітацію, взаємодіяти з іншими об'єктами, виконувати рух, а також враховувати

колізії з іншими об'єктами у сцені. Використання компонента Rigidbody 2D дозволяє створювати реалістичну та захоплюючу геймплейну взаємодію в грі.

Також до даного об’єкту було написано скрипт, який і реалізовує падіння та відновлення платформи.

Символом завершення рівня є елемент дерева (Tree), при зіткненні з яким, персонаж переходить на наступний рівень. До цього елементу було додано скрипт, який відловлює зіткнення персонажа з деревом та переходом на наступний рівень. Також було додано компонент Box Collider 2D, розміри якого було підігнано під стовбур дерева для того, щоб візуалізувати доторкання до стовбура дерева, а не до самого спрайту, який має квадратну форму та є прозорим в ділянках, де нема текстури дерева. Тобто, якщо спрацював би цей тригер, то персонаж переходив би на інший рівень, зтикаючись з повітрям біля дерева, а не, безпосередньо, з самим деревом, що створювало б незрозумілість і нелогічність у гравця.

3.5 Створення персонажа гри

У грі присутній персонаж-дівчинка, так як жіноча стать асоціюється з красою і позитивними якостями.

По-перше, представлення дівчинки у якості головної героїні може стати прикладом для інших дівчат, які грають у гру, що дівчата можуть бути сильними, спритними та безстрашними, які йдуть наперекір перешкодам. Це може допомогти розширити роль жінок у гральній індустрії та посилити позитивні стереотипи щодо жінок.

По-друге, головна героїня-дівчинка може мати позитивний вплив на гравців з точки зору ідентифікації з персонажем. Жіночі гравці можуть легше відчувати зв'язок з головною героїнею, що може підвищити рівень їх задоволення від гри та збільшити їхню мотивацію продовжувати грати.

Усі ці фактори разом можуть допомогти збільшити рівень зацікавленості гравців у грі та зробити її більш привабливою для широкої аудиторії (рис. 3.19).

Рисунок 3.19 – Персонаж-дівчинка гри

Щоб втілити ігрову механіку персонажу, потрібно для початку надати можливість йому рухатись. Для цього був створений об’єкт з назвою GroundCheck, який перевірятиме, чи стоїть персонаж на землі – це буде використано для створення коректної анімації.

Сам персонаж містить в собі компонент Rigitbody 2D, який був налаштований саме під нього (рис. 3.20).

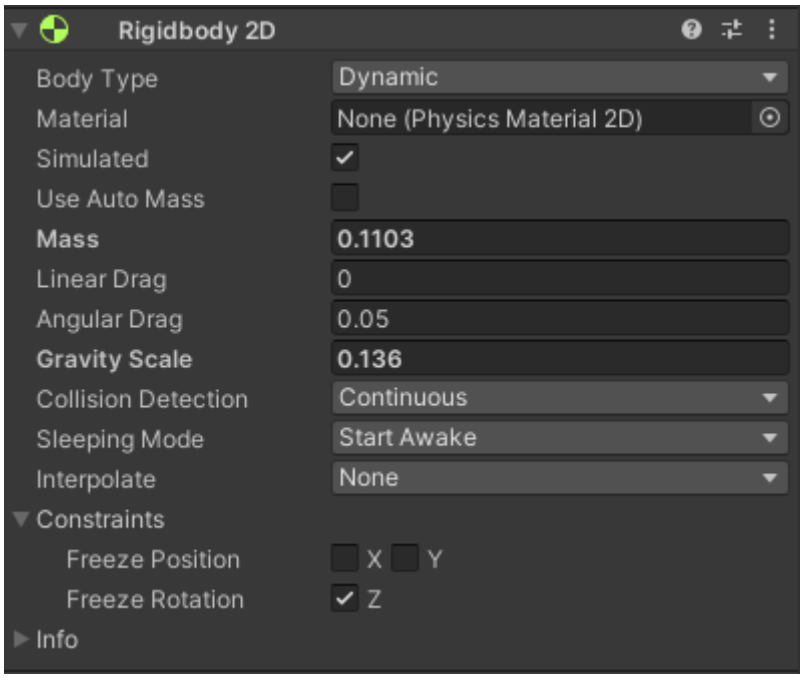

Рисунок 3.20 – Налаштований компонент Rigitbody 2D для персонажа

Графа Body Type містить три режими: Dynamic, Kinematic, Static. Характеристика кожного з цих режимів наведена в Таблиці 3.1:

Таблиця 3.1 – Характеристика режимів Body Type

Назва	Характеристика
Dynamic	Об'єкт рухається під впливом фізики, його поведінка залежить від сил і прикладених до нього зовнішніх сил.
Kinematic	Об'єкт керується програмно, ігнорує фізичну симуляцію. Він може рухатися, але його рух не залежить від фізичних сил.
Static	Об'єкт не рухається під впливом фізики. Його положення фіксоване і не змінюється.

Так як персонаж буде взаємодіяти з фізикою, було обрано режим Dynamic.

Графа Simulated вказує, чи потрібно симулювати фізику для даного об'єкта. Якщо він встановлений в значення True, то фізика буде впливати на об'єкт, якщо ж він встановлений в значення False, то фізика не буде враховуватися для цього об'єкта. Тож було встановлено значення True.

Графа Mass відповідає за масу об'єкта та визначає його опір руху та взаємодію з іншими об'єктами в сцені. Об'єкти з більшою масою будуть важче рухатися та відчувати силу гравітації більш інтенсивно.

Gravity Scale – цей параметр визначає, як сила гравітації впливає на об'єкт. Значення 1 відповідає стандартному впливу гравітації, значення більше 1 збільшує вплив гравітації, а значення менше 1 зменшує його.

Collision Detection – цей параметр визначає метод виявлення зіткнень об'єкта з іншими об'єктами. Є декілька варіантів:

- Discrete. Виявлення зіткнень здійснюється методом дискретного зіткнення. Він є найбільш точним, але може бути вимогливим до ресурсів.

- Continuous. Виявлення зіткнень здійснюється методом неперервного зіткнення. Він є менш точним, але менш вимогливим до ресурсів.
- Continuous Dynamic. Виявлення зіткнень здійснюється методом неперервного зіткнення, з врахуванням руху об'єктів. Він є найбільш точним, але найбільш вимогливим до ресурсів.
- None. Виявлення зіткнень вимкнено.

Sleeping Mode – цей параметр визначає, як об'єкт переходить у сплячий режим, коли він неактивний. Сплячий режим дозволяє зменшити навантаження на фізичну симуляцію для неактивних об'єктів і економити ресурси. Є декілька варіантів:

- Never Sleep. Об'єкт ніколи не переходить у сплячий режим.
- Start Awake. Об'єкт починає у активному стані і не переходить у сплячий режим.
- Start Asleep. Об'єкт починає у сплячому режимі, і симуляція фізики активується, якщо він зіткнеться з активним об'єктом або отримає зовнішній вплив.

Кожен з цих параметрів має важливе значення в реалізації фізичної поведінки об'єктів у грі. Налаштування їх правильно допомогло створити задовольняючу фізичну симуляцію для персонажа.

Також до об’єкта персонажа було додано компонент Box Collider 2D, щоб відслідковувати зіткнення, скрипт управління персонажем Player Controller та компонент Animator.

Компонент Animator є одним з ключових елементів у розробці ігрового проєкту. Він відповідає за анімацію об'єктів в грі та керує їхнім рухом, зміною станів і взаємодією з оточуючим середовищем.

Основна концепція, яка використовується в компоненті Animator, - це станова машина (state machine). Станова машина складається зі станів (states), переходів (transitions) та анімаційних параметрів (animation parameters). Кожен стан представляє певний аспект анімації, наприклад, стояння, біг або стрибок [24] (рис. 3.21).

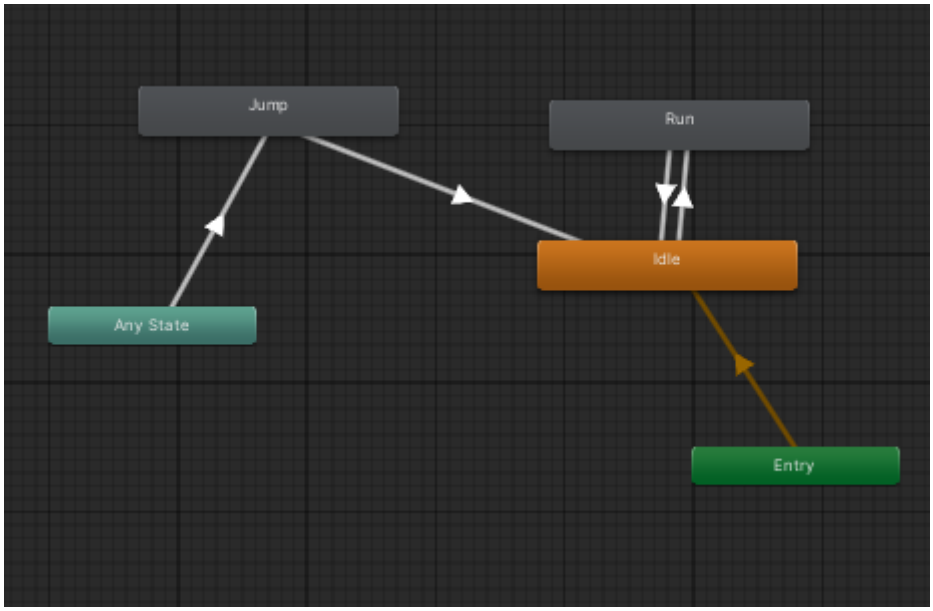

Рисунок 3.21 – Створення анімації для персонажа

Зміна однієї анімації на іншу буде залежати від встановленої швидкості.

Тому було створено Значення Speed з плаваючою крапкою і його було додано до переходу з анімації з бігу до спокійного стану і навпаки (рис. 3.22).

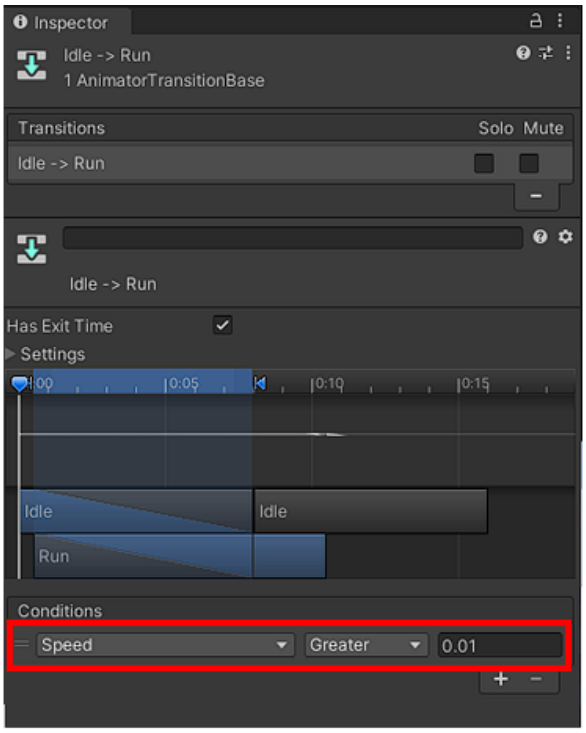

Рисунок 3.22 – Інспектор переходу анімації зі спокійного стану в біг

Для плавного переходу анімацій було встановлено значення 0.01.

Аналогічну роботу було прорoблено для всіх типів анімацій персонажа, а також створено відповідні змінні.

3.6 Програмна реалізація гри

3.6.1 Лічильник кристалів

Для реалізації ідеї лічильника кристалів було додано об’єкт Canvas, а його дочірніми об’єктами було зроблено елемент Image та Text. До об’єкту Text було під’єднано скрипт CrystalCounter (рис. 3.23).

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

Рисунок 3.23 – Використані бібліотеки

Перша бібліотека буде використовуватись при написанні всіх скриптів. UnityEngine є однією з основних бібліотек розробки ігор в середовищі Unity. Вона надає розширений функціонал для створення ігрових об'єктів, керування сценою, роботи з фізикою, анімацією та багатьма іншими аспектами ігрової розробки.

UnityEngine включає в себе класи і структури, які дозволяють створювати, маніпулювати та взаємодіяти з об'єктами у грі. Наприклад, клас GameObject представляє ігровий об'єкт, який може містити компоненти, такі як візуальні моделі, колайдери та інші [25].

Бібліотека UnityEngine також містить класи для роботи з фізикою. Клас Rigidbody дозволяє задавати фізичні властивості об'єктів, такі як маса, твердість та динамічні властивості. Це дозволяє об'єктам рухатися під впливом сил, взаємодіяти з іншими об'єктами за допомогою фізичних зіткнень та поводитися реалістично в ігровому середовищі.

Завдяки бібліотеці UnityEngine.UI, можна створювати зручні та інтерактивні інтерфейси для користувачів, включаючи кнопки, меню, панелі та інші елементи, які взаємодіють з грою.

Вона надає можливість створювати елементи інтерфейсу, такі як кнопки, текстові поля, панелі та інші, які взаємодіють з грою. Unity.UI дозволяє легко налаштовувати зовнішній вигляд та поведінку інтерфейсних елементів, а також виконувати операції з ними, такі як зміна тексту, відображення анімацій або реагування на користувацькі дії. Ця бібліотека спрощує розробку зручного та привабливого інтерфейсу в ігровому середовищі Unity, що допомагає покращити взаємодію гравця з грою.

Бібліотека UnityEngine.UI також надає розширені можливості для управління графічними елементами інтерфейсу в середовищі Unity. Вона містить класи для створення, налаштування та взаємодії з різноманітними елементами UI, такими як кнопки, текстові поля, зображення, прокручування, вкладки та інші.

За допомогою бібліотеки UnityEngine.UI можна встановлювати різні властивості елементів інтерфейсу, такі як колір, розмір, положення, шрифт, текстуру та інші. Це дозволяє розробникам створювати кастомізовані інтерфейси, що відповідають вимогам конкретної гри.

Окрім налаштування зовнішнього вигляду, бібліотека також дозволяє встановлювати реакцію елементів UI на події та взаємодіяти з ними. За допомогою скриптів Unity, розробники можуть прив'язувати функції до подій, таких як натискання кнопки або зміна значення текстового поля, щоб виконувати певні дії під час взаємодії користувача з інтерфейсом.

Бібліотека UnityEngine.UI дозволяє створювати інтуїтивно зрозумілі та ефективні інтерфейси для користувачів, що покращує взаємодію гравця з грою і створює зручне та привабливе візуальне середовище.

Unity має багато головних методів, які використовуються під час розробки ігор. Кожен з цих методів має свою власну функціональність і призначення, які допомагають управляти поведінкою об'єктів у грі та взаємодіяти з користувачем (рис. 3.24).

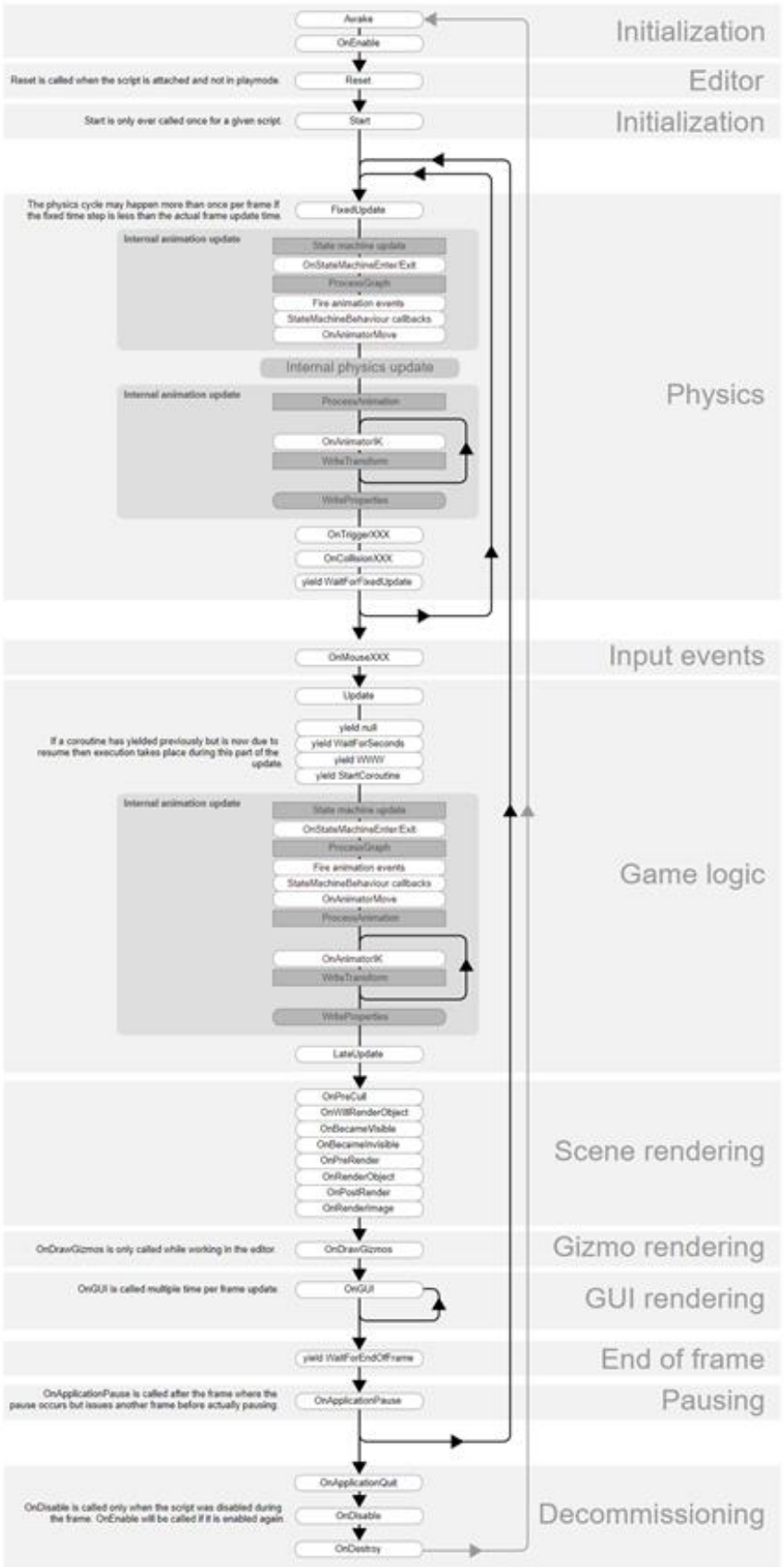

Рисунок 3.24 – Послідовність роботи вбудованих методів Unity

Першим етапом написання скрипта є оголошення глобальної змінної crystalCountText, яке є посиланням на текстове поле для відображення лічильника.

Наступним кроком є написання самої логіки лічильнику. Для цього було використано вбудовані методи Start() та Update().

Метод Start() викликається один раз при запуску сцени або об'єкта і використовується для початкової ініціалізації. В ньому зазвичай виконується початкове налаштування компонентів та змінних.

В цьому методі було отримано посилання на скрипт CrystallCollect (рис. 3.25).

```
private void Start()
{
    crystalController = FindObjectOfType<CrystallCollect>();
}
```

Рисунок 3.25 – Код методу Start()

Метод Update() викликається на кожному кадрі і використовується для оновлення логіки гри. В ньому зазвичай виконуються перевірки вводу користувача, рух об'єктів, зміна станів тощо.

У цьому методі було написано логіку оновлення значення лічильника на текстовому полі (рис. 3.26).

```
private void Update()
{
    crystalCountText.text = "X" + CrystallCollect.crystalCount.ToString();
}
```

Рисунок 3.26 – Код методу Update()

В свою чергу, до самого елементу кристала було додано скрипт CrystallCollect, на який і посилався попередній скрипт (рис. 3.27).

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("Player"))
    {
        crystalCount++;
        Destroy(gameObject);
    }
}
```

Рисунок 3.27 – Код методу OnTriggerEnter2D()

В методі OnTriggerEnter2D() було реалізовано головну частину логіки роботи кристалів, а саме перевірка тегу персонажа, збільшення лічильника кристалів та знищення об'єкта кристала.

3.6.2 Управління персонажем та його логіка

До об’єкта персонажа було додано скрипт PlayerController, в якому і були прописані всі механіки (рис. 3.28).

```
private CrystallCollect crystalController;
Rigidbody2D rigidBody;
public float moveSpeed;
public float jumpForce;
bool isFacingRight = true;
Animator anim;
public Vector2 moveVector;
bool isGrounded;
public Transform groundCheck;
float groundRadius = 0.5f;
public LayerMask whatIsGround;
private int extraJump;
public int extraJumpValue;
```

Рисунок 3.28 – Змінні скрипта PlayerController

Для початку було оголошено змінні, потрібні для обчислень:

- crystalController: використовується для отримання доступу до компонента CrystallCollect із сцени для збору кристалів;
- rigidBody: використовується для доступу до компонента Rigidbody2D гравця, щоб керувати його фізичним рухом;
- moveSpeed: визначає швидкість руху гравця;

- jumpForce: визначає силу стрибка гравця;
- isFacingRight: вказує на те, чи звернений гравець вправо;
- anim: використовується для доступу до компонента Animator гравця для управління анімацією;
- moveVector: визначає вектор руху гравця;
- isGrounded: показує, чи гравець знаходиться на землі;
- groundCheck: вказує на місцезнаходження об'єкту, що перевіряє, чи знаходиться гравець на землі;
- groundRadius: визначає радіус об'єкту, який перевіряє, чи знаходиться гравець на землі;
- whatIsGround: визначає, які шари вважаються землею для перевірки зіткнень;
- extraJump: визначає кількість додаткових стрибків, які гравець може здійснити;
- extraJumpValue: визначає початкову кількість додаткових стрибків гравця.

Далі було створено метод для керування рухом гравця Run() (рис. 3.29).

```
public void Run()
{
    float moveInput = Input.GetAxis("Horizontal");
    anim.SetFloat("Speed", Mathf.Abs(moveInput));


    rigidBody.velocity = new Vector2(moveInput * moveSpeed, rigidBody.velocity.y);
    if (moveInput > 0 && !isFacingRight)
        Flip();
    else if (moveInput < 0 && isFacingRight)
        Flip();
}
```

Рисунок 3.29 – Метод для керування рухом гравця Run()

Та для того, щоб створена раніше анімація працювала коректно, потрібно було створити метод, який міг би змінювати орієнтацію персонажа, якщо той дивиться вправо, або вліво (рис. 3.30).

```
void Flip()
{

    isFacingRight = !isFacingRight;

    Vector3 theScale = transform.localScale;

    theScale.x *= -1;

    transform.localScale = theScale;
}
```

Рисунок 3.31 – Код методу Flip

Також важливим аспектом є написання логіки смерті персонажа. Було поставлене завдання зробити респавн гравця на початок поточного рівня. Під час зіткнення гравця з елементом води, персонаж переноситься на початок рівня зі скинутим прогресом та розпочинає спробу наново.

Для цього було використано вбудований метод OnTriggerEnter2D(Collider2D other). Він викликається, коли об'єкт перетинає тригер іншого об'єкта. Використовується для обробки подій тригерів (рис. 3.32).

```
if (other.tag == "Respawn")
{
    SceneManager.LoadScene(1);
    crystalController.ResetCrystals();
}
```

Рисунок 3.32 – Фрагмент коду з метода OnTriggerEnter2D(Collider2D other)

Висновки до розділу 3

Розробка гри в середовищі Unity включає використання різних бібліотек та написання скриптів. Для створення ігрових об'єктів, керування сценою та роботи з фізикою використовується бібліотека UnityEngine. Для створення інтерфейсу користувача використовується бібліотека UnityEngine.UI. Головні методи, які використовуються під час розробки ігор, включають Start() для початкової ініціалізації та Update() для оновлення логіки гри на кожному кадрі.

Розроблено скрипт для ігрового персонажа, який використовується в середовищі Unity. Скрипт включає логіку руху, зміну орієнтації, стрибок та зіткнень

з об'єктами. Лічильник кристалів було реалізовано за допомогою об'єкта Canvas та дочірніх об'єктів Image та Text. Для програмування використовувалися бібліотеки UnityEngine та UnityEngine.UI, які надають функціонал для розробки ігор та створення інтерактивного інтерфейсу. Методи Start() та Update() використовувалися для початкової ініціалізації та оновлення логіки гри. Скрипт персонажа мав логіку збирання кристалів, зміну станів при зіткненні з об'єктами, рух та анімацію. Крім того, була реалізована логіка смерті персонажа, яка переносила його на початок рівня.

ВИСНОВКИ

В результаті проведеного дослідження сучасного стану в ігровій індустрії було виявлено значну популярність 2D ігор та зростаючий попит на них серед гравців.

Під час аналізу та обґрунтування вибору інструментальних засобів для розробки 2D проєкту було враховано різноманітні фактори, включаючи доступність, функціональність та специфіку проєкту. У результаті вибір був зроблений на користь середовища Unity, яке надає широкі можливості для розробки 2D ігор та має зручний інтерфейс.

Дослідження аспектів розробки 2D гри за допомогою двигуна показало, що такий підхід дозволяє ефективно використовувати готові компоненти та функції, спрощуючи процес розробки.

Задача постановки ідеї відеогри була успішно виконана шляхом розробки концепції гри та визначення головних механік, що сприяє захоплюючому геймплею.

В результаті розробки та програмної реалізації 2D гри було успішно створено функціональний застосунок, який включає в себе інтерактивність, рух об'єктів, логіку зіткнень, анімацію та інші необхідні елементи для заданого ігрового проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. William Hosch. Encyclopedia Britannica: electronic platform game, 2009. 375 p.

2. Alexander S. They create worlds: the story of the people and companies that shaped the video game industry, vol. I: 1971-1982. Taylor & Francis Group, 2019. 586 p.

3. Donovan T. Replay: the history of video games. East Sussex, England: Yellow Ant, 2010. 501 p.

4. ESRB ratings guides, categories, content descriptors. ESRB Ratings. URL: https://www.esrb.org/ratings-guide/ (date of access: 14.06.2023).

5. Усі жанри комп'ютерних ігор. UA PLAY. URL: https://uaplay.com.ua/usi-zhanry-pc-ihor/ (дата звернення: 14.06.2023).

6. Wolf M. J. P. Encyclopedia of video games: the culture, technology, and art of gaming. ABC-CLIO, LLC, 2021.

7. Moore M. Basics of game design. Taylor & Francis Group, 2011. 378 p.

8. Quandt T., Kowert R. Video game debate: unravelling the physical, social, and psychological effects of video games. Taylor & Francis Group, 2015. 204 p

9. Технічна документація Microsoft. Microsoft; вебсайт. URL: https://docs.microsoft.com/ru-ru/ (дата звернення: 14.06.2023).

10. Microsoft. Introduction to visual basic. NET programming with Microsoft. NET: msm2559bcppb. Microsoft Corporation, 2002.

11. .NET | Build. Test. Deploy. Microsoft. URL: https://dotnet.microsoft.com/en-us/ (date of access: 14.06.2023).

12. 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 14.06.2023).

13. Стаття про оновлення версій C#. Microsoft: вебсайт. URL: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-10 (дата звернення 14.06.2023).

14. Докуметнація мови C#. Microsoft: вебсайт. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення 14.06.2023).

15. Calabrese D. Unity 2D game development. Packt Publishing, 2014. 126 p.

16. Торн Алан. Мистецтво створення сценаріїв в Unity. СПб: ДМК. 2016. 362 с.

17. Unity game development essentials. Packt Publishing, 2009. 203 p.

18. Thorn A. Mastering unity scripting. Packt Publishing, 2015. 380 p.

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

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

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

22. Джон Меннінг, Перріс Батфілд-Еддісон. Unity для розробника. Наука і техніка, 2018. 352 p.

23. 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.

24. Торн Алан. Основи анімації в Unity, ДМК-Прес. 2019. 196 p.

25. Thorn A. Unity 2018 By Example. Packt Publishing, 2015, 300 p.

ДОДАТОК А

Лістинг коду застосунку

Скрипт CrystalCounter:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class CrystalCounter : MonoBehaviour
{
    public Text crystalCountText;

    private CrystallCollect crystalController;

    private void Start()
    {
        crystalController = FindObjectOfType<CrystallCollect>();
    }

    private void Update()
    {
        crystalCountText.text = "X" + CrystallCollect.crystalCount.ToString();
    }
}
```

Скрипт CrystallCollect:

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

public class CrystallCollect : MonoBehaviour
{
    public static int crystalCount=0;

    private PlayerController playerController;
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            crystalCount++;
            Destroy(gameObject);
        }
    }
    public void ResetCrystals()
    {
        crystalCount = 0;
    }

}
```

Скрипт PlayerController:

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

```
using UnityEngine.SceneManagement;


public class PlayerController : MonoBehaviour
{
    private CrystallCollect crystalController;
    Rigidbody2D rigidBody;
    public float moveSpeed;
    public float jumpForce;
    bool isFacingRight = true;
    Animator anim;
    public Vector2 moveVector;
    bool isGrounded;
    public Transform groundCheck;
    float groundRadius = 0.5f;
    public LayerMask whatIsGround;
    private int extraJump;
    public int extraJumpValue;
    // Use this for initialization
    void Start()
    {
        extraJump = extraJumpValue;
        anim = GetComponent<Animator>();

        rigidBody = GetComponent<Rigidbody2D>();
        crystalController = FindObjectOfType<CrystallCollect>();
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        Run();

        isGrounded = Physics2D.OverlapCircle(groundCheck.position, groundRadius,
whatIsGround);
        anim.SetBool("Ground", isGrounded);
        anim.SetFloat("vSpeed", rigidBody.velocity.y);
        if (!isGrounded)
            return;
    }



    public void Run()
    {
        float moveInput = Input.GetAxis("Horizontal");
        anim.SetFloat("Speed", Mathf.Abs(moveInput));


        rigidBody.velocity = new Vector2(moveInput * moveSpeed,
rigidBody.velocity.y);
        if (moveInput > 0 && !isFacingRight)
            Flip();
        else if (moveInput < 0 && isFacingRight)
            Flip();
    }

    void Flip()
    {

        isFacingRight = !isFacingRight;

        Vector3 theScale = transform.localScale;
```

```
            theScale.x *= -1;

            transform.localScale = theScale;
        }

        void Update()
        {
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                SceneManager.LoadScene("MainMenu");
            }
            if (isGrounded == true)
            {
                extraJump = extraJumpValue;
            }
            if (isGrounded && Input.GetKeyDown(KeyCode.UpArrow) && extraJump > 0)
            {
                anim.SetBool("Ground", false);
                rigidBody.AddForce(Vector2.up * jumpForce);
                extraJump--;
            }
            else if (isGrounded && Input.GetKeyDown(KeyCode.UpArrow) && extraJump == 0
    && isGrounded == true)
            {
                rigidBody.AddForce(Vector2.up * jumpForce);
            }
        }
        private void OnTriggerEnter2D(Collider2D other)
        {
            if (other.tag == "Respawn")
            {
                SceneManager.LoadScene(1);
                crystalController.ResetCrystals();
            }
            if (other.tag == "Respawn1")
            {
                SceneManager.LoadScene(2);
                crystalController.ResetCrystals();
            }
            if (other.tag == "Respawn2")
            {
                SceneManager.LoadScene(3);
                crystalController.ResetCrystals();
            }

        }
        private void OnCollisionEnter2D(Collision2D collision)
        {
            if (collision.gameObject.tag.Equals("Platform"))
            {
                this.transform.parent = collision.transform;
            }
        }
        private void OnCollisionExit2D(Collision2D collision)
        {
            if (collision.gameObject.tag.Equals("Platform"))
            {
                this.transform.parent = null;
            }
        }

    }
```