МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Гра в жанрі 2D платформер на базі ігрового двигуна UNITY»

Виконавець: Мар'яна СЕРГІЙЧУК

Керівник: к.т.н., доцент Володимир ДРОВОВОЗОВ

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *комп'ютерних інформаційних технологій*

Спеціальність *122 «Комп'ютерні науки»*

Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Сергійчук Мар'яни Геннадіївни
(ПІБ випускника)

1. Тема роботи: «Гра в жанрі 2D платформер на базі ігрового двигуна UNITY» затверджена наказом ректора № 623/ст від 01.05.2023р.

2. Термін виконання роботи: з 15 травня 2023 року по 25 червня 2023 року.

3. Вихідні дані до роботи: Гра в жанрі 2D платформер на базі ігрового двигуна UNITY.

4. Зміст пояснювальної записки: 1. Аналіз предметної області. 2. Вибір засобів програмної реалізації. 3. Реалізація проекту.

5. Перелік обов'язкового ілюстративного матеріалу: 1. Поняття та класифікація комп'ютерних ігор. 2. Основні етапи створення комп'ютерних ігор. 3. Вибір засобів програмної реалізації. 4. Концепція та сценарій гри. 5. Графічне оформлення. 6. Фізичні властивості об'єктів. 7. Анімація об'єктів. 8. Звукові ефекти. 9. Розробка меню початку, паузи та кінця гри. 10. Аналіз отриманого результату.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області. Написання 1 розділу, представлення керівнику.	15.05.2023-20.05.2023	
2.	Вибір та опис засобів програмної реалізації. Написання 2 розділу, представлення керівнику.	21.05.2023-27.05.2023	
3.	Розробка 2D гри у жанрі платформер. Написання 3 розділу, представлення керівнику.	28.05.2023-09.06.2023	
4.	Загальне редагування та друк пояснювальної записки.	10.06.2023-12.06.2023	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	12.06.2023-15.06.2023	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	16.06.2023-18.06.2023	

7. Дата видачі завдання 15.05.2023р.

Керівник кваліфікаційної роботи _____ Володимир ДРОВОВОЗОВ
(підпис керівника)

Завдання прийняв до виконання _____ Мар'яна СЕРГІЙЧУК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Гра в жанрі 2D платформер на базі ігрового двигуна UNITY» містить: 60 сторінок, 32 рисунки, 17 інформаційних джерел.

Об’єкт дослідження – гра в жанрі 2D платформер.

Предмет дослідження – технології розробки гри у жанрі 2D платформер.

Мета кваліфікаційної роботи – розробити 2D гру у жанрі платформер з використанням найефективнішого програмного забезпечення.

Методи дослідження – логічний, аналізу, порівняльний, обробка літературних джерел та проектування.

Прототип гри, що розробляється, є цінним активом, який може бути використаний для різних цілей. Він може стати основою для подальшого розвитку та вдосконалення гри. Також, прототип може бути використаний для демонстрації потенціалу та можливостей даної розробки в загальному користуванні.

Для розробки комп’ютерної гри знайдено та використано найефективніші програмні засоби, а також різноманітні двовимірні об’єкти з доступних джерел, готові матеріали та текстури.

КОМП’ЮТЕРНА ГРА, UNITY, ІГРОВИЙ ДВИГУН, C#.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

3D	–	Тривимірне
2D	–	Двовимірне
Геймплей	–	Термін, яким називають особливості взаємодії людини з відеогрою.
ПК	–	Персональний комп’ютер
Мультиплеєр	–	Багатокористувацька гра

ВСТУП

У сучасному світі комп'ютерні технології відіграють важливу роль у розвитку суспільства. Їх вплив на економіку, науку, освіту та культуру стає все більш суттєвим, надаючи нові можливості та спонукаючи до пошуку інноваційних рішень.

У даний час комп'ютерні ігри займають важливе місце в сфері розваг та розвитку технологій. Індустрія відеоігор з’явилася не так давно, порівнюючи з іншими подібними сферами, але вже займає одну з лідируючих позицій на світовому ринку і має велику популярність. Це пов’язано зі стрімким розвитком і розповсюдженням комп’ютерних технологій, особливо, появі мережі Інтернет, що надає доступ до комп’ютерних ігор як через стаціонарні ПК, так і через планшети, смартфони і звичайні телефони. Саме завдяки цьому відеоігри більш доступні для кінцевого користувача, аніж інші види розваг, що є їх вагомою перевагою.

З розвитком ігрової індустрії з’являються нові ідеї, технології та досягнення. На сьогоднішній день комп’ютерні ігри - це не тільки гарно проведений час, це яскраві емоції, враження, незабутній досвід, це спільноти однодумців, що стрімко розвиваються, це професія і це, з недавніх часів, спорт.

Для багатьох людей відеоігри займають величезну частину життя, ставши не просто розвагою, а повноцінним хобі або способом виразити свою креативність та взаємодіяти з іншими гравцями по всьому світу. Відеоігри стають платформою для спілкування, співпраці, змагань та вирішення складних завдань. Вони також впливають на розвиток навичок, таких як стратегічне мислення, координація рухів, реакція на швидкі зміни. Ця залученість до ігрового світу спонукає розробників створювати все більш захопливі та інноваційні ігри, а також використовувати нові технології для покращення геймплею та візуального досвіду.

Актуальність теми кваліфікаційної роботи «Гра в жанрі 2D платформер на базі ігрового двигуна UNITY» ґрунтується на тому, що сьогодні ігрова індустрія перебуває в стані стрімкого росту та розвитку. Комп'ютерні ігри стають все більш популярними серед різних груп користувачів, включаючи

дітей, підлітків, молодь та дорослих. Це створює велику потребу в розробці якісних і захоплюючих ігрових продуктів. Дана кваліфікаційна робота відповідає актуальним тенденціям розвитку ігрової індустрії та відкриває можливості для подальшого дослідження та розвитку в області розробки комп'ютерних ігор.

Об’єкт дослідження – гра у жанрі 2D платформер.

Предмет дослідження – технології розробки гри у жанрі 2D платформер.

Мета кваліфікаційної роботи – створення комп’ютерної гри на ігровому двигуні Unity, яка може стати основою для наступних програмних продуктів, значно скоротивши час їх створення.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

- Виконати аналіз предметної області;
- Проаналізувати ринок комп’ютерних ігор;
- Дослідити основні етапи створення комп’ютерної гри;
- Провести огляд програмних засобів для розробки ігор;
- Створити концепцію та сценарій гри;
- Реалізувати гру на основі даних аналізу;
- Провести аналіз отриманого результату.

Для досягнення поставленої мети й виконання завдань використано наступні методи: логічний, аналізу, порівняльний, обробка літературних джерел та проектування.

Практичне значення отриманих результатів. Результати кваліфікаційної роботи в подальшому можуть бути використані як в освітніх цілях, так і для практичного застосування у створенні власних ігрових проектів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Дослідження поняття комп’ютерних ігор

Традиційно комп'ютерна гра є програмним забезпеченням, призначеним для забезпечення ігрового процесу. Під час гри використовується спеціальна програма, що створює імітацію взаємодії між ігровим персонажем та користувачем (або групою користувачів) у віртуальному просторі за певним алгоритмом. Для здійснення гри використовуються різноманітні пристрої введення, такі як комп'ютерна миша, клавіатура, камера, джойстик тощо [1].

Комп'ютерна гра, так само як і традиційна гра, є симуляцією реальності, де гравець свідомо взаємодіє зі світом, який є віртуальним. Проте, в комп'ютерних іграх існують обмеження в просторі, часі та можливостях. Це відрізняє їх від традиційних ігор. Візуальні ефекти комп'ютерних ігор створюються розробниками і є результатом їхньої творчої діяльності, а не власної уяви гравця. У комп'ютерних іграх також використовуються правила, які впроваджені у їхній алгоритм.

У сучасному світі комп’ютерні ігри стали невід’ємною частиною повсякденного життя багатьох людей. Саме тому вони часто створюються на основі зовнішніх джерел, наприклад фільмів, книг, серіалів або навіть міфів та історичних подій. Однак з часом почали з’являтися випадки, коли вже відомі ігрові серії випускають додаткові матеріали, які розширюють уявний світ гри.

Крім того, спеціально розроблені ігри можуть бути використані як навчальний матеріал або для наукових досліджень, але такі ігри випускаються досить рідко.

Ігрова індустрія має настільки вагомий вплив на сучасне суспільство, що деякі ігри вже давно стали об’єктом аматорських і професійних змагань, що призвело до створення цілком нового виду спорту - кіберспорту.

Вплив комп'ютерних ігор на сучасне суспільство має значний масштаб, що спричиняє розповсюдження індустрії ігор у сфері прикладного програмного забезпечення. Наприклад, деякі європейські навчальні заклади використовують відомі ігри у процесі навчання, а для тренування солдатів створюються спеціальні симулятори. Багато країн визнали кіберспорт як офіційний вид спорту, а в 2011 році уряд США надав комп'ютерним іграм статус окремого виду мистецтва, нарівні з театром та кіно. [2].

Комп'ютерні ігри стали неодмінною частиною сучасного життя суспільства, і за останні декілька років їх використання значно розширилось. Тепер ігри використовуються людством не тільки задля розваг та відпочинку, як було на початку, але й навчання та наукових досліджень.

1.2. Класифікація комп’ютерних ігор

Аналогічно до музичних або літературних творів, відеоігри можна категоризувати за жанрами, які базуються на спільних характеристиках ігрового процесу та його цілях. Кількість комп’ютерних ігор просто величезна, тому вони можуть класифікуватися за характеристиками, завданнями чи за типом геймплея [3]. Це призводить до створення категорій або жанрів, що поділяються на піджанри. Одна гра може належати до кількох жанрів одночасно. Незважаючи на це, в ході розвитку комп'ютерних ігор склалася наступна класифікація:

1) RPG (RolePlaying Game) (Рольова гра) - жанр ігор, в якому гравець «грає роль», певного персонажа, який має визначені характеристики та навички. Цей жанр ігор походить від настільних рольових ігор і використовує їхню ігрову механіку.

2) Arcade (аркада) - це жанр відеоігор, в яких геймплей (ігровий процес) заснований на швидкій реакції гравця, обмеженій свободі керованих персонажів і простому управлінні.

3) Platformer (Платформер) - жанр ігор, в яких головний герой здійснює стрибки по різноманітних платформах і подолання перешкод, збираючи необхідні предмети для завершення рівня.

4) Action (Екшн) - жанр ігор, в якому акцент робиться на використанні фізичних навичок гравця, включаючи координацію рухів і швидкість реакції. Гравець повинен вчасно виконати потрібні дії з обмеженим часом. Небойові виклики можуть включати уникання пасток, проходження рівнів у встановлений термін.

5) Simulation (симулятор) - жанр ігор, в якому в різних ступенях реалістично відтворюються аспекти реального життя. Симулятори можуть охоплювати широкий спектр сфер, включаючи симулятори побачень, авіасимулятори та багато інших.

6) Strategy (стратегія) - жанр ігор, де успіх залежить від уміння розробляти та виконувати стратегічні плани. У більш вузькому розумінні, стратегічні ігри симулюють збройні конфлікти, де гравець керує арміями чи країнами, вимагаючи стратегічного мислення.

7) Sport (спортивна гра) - жанр ігор, в якому відтворюються реальні види спорту, такі як футбол, хокей, баскетбол, або вигадані види спорту, наприклад, квідич з книжки "Гаррі Поттер".

8) Educational (навчальна гра) - жанр ігор, в якому гравець отримує навички та знання шляхом виконання різних завдань у грі. Зазвичай такі ігри призначені для розвитку дітей, але також існують версії для дорослих.

9) Adventure (пригода) - жанр ігор, в якому гравець вирушає в подорож по захопливій історії і має виконувати різноманітні завдання, просуваючись по сюжету і досліджуючи світ.

За основним критерієм поділу жанрів в іграх враховуються дії, які найчастіше здійснюються гравцем під час гри. Звичайно, ігри можуть бути класифіковані в різні групи залежно від основного акценту, який вони роблять на певних типах дій. Загалом, ігри можна поділити на три великі групи: ігри контролю, ігри дій та ігри інформації. (табл. 1.1).

Таблиця 1.1. – Жанрова класифікація комп’ютерних ігор

	Ігри контролю	Ігри дії	Ігри інформації
Опис	Ігри, які мають забезпечити контроль над ситуацією або обмежити вибір дій гравця	Ігри, у яких гравець може взаємодіяти зі світом гри та виконувати дії, які впливають на події гри	Ігри, які мають на меті передати інформацію гравцю
Мета	Розвивати або перевіряти навички управління та контролю	Розважати, розвивати навички взаємодії зі світом гри та вирішення завдань	Передати корисну інформацію гравцю
Цільова аудиторія	Люди, які люблять контролювати ситуацію та вирішувати головоломки	Люди, які люблять діяти та взаємодіяти зі світом гри	Люди, які цікавляться певною темою та бажають дізнатися більше про неї
Навички, що розвиває	Логічне мислення, увага, концентрація, стратегічне планування	Реакція, координація рухів, прийняття рішень у складних ситуаціях	Знання певної теми, пам’ять, розвиток
Приклади ігор	Всі види стратегій, різні економічні ігри, варгейми, тактики, шахи, головоломки.	Екшн-ігри, гонки, ігри-пригоди, платформери.	Симулятори, енциклопедії, квізи, рольові ігри, пригоди.

Усі ігри можна поділити за кількістю гравців на одиночні та мультиплеєрні. Завдяки такій системі класифікації можна встановити режими, які будуть присутні в грі.

Одиночна гра - це тип гри, в якому бере участь одна людина. Зазвичай гравцеві протистоїть штучний інтелект, а його метою є досягнення кінцевої точки гри (проходження), збір ресурсів або підвищення своїх навичок. Часто ці цілі поєднуються.

Іншим видом гри є мультиплеєр. Цей режим гри призначений для гри більше ніж однієї людини одночасно. В багатьох іграх комбінують одиночну гру та мультиплеєр.

За візуальним уявленням комп'ютерні ігри можна класифікувати наступним чином:

– текстові ігри - це ігри з мінімальним графічним представленням, в

яких взаємодія з гравцем відбувається за допомогою тексту;

– 2D ігри - в цих іграх усі елементи зображені у вигляді двовимірної графіки, включаючи спрайти;

– 3D ігри - в таких іграх усі елементи відображаються у тривимірних моделях, що надає їм реалістичний вигляд.

За типом платформи, на яких гра використовується:

– ПК;

– ігрові приставки / консолі;

– мобільні телефони.

1.3. Основні етапи створення комп’ютерних ігор

Для створення високоякісної та успішної комп'ютерної гри кожному розробнику необхідно пройти всі етапи від ідеї до введення додатку в експлуатацію. Основні етапи створення відеоігор можна поділити на п'ять складових. (рис. 1.1).

Рис. 1.1. Основні етапи створення комп’ютерних ігор

Етап аналізу та планування. Вивчається ринку комп'ютерних ігор, виконується аналіз популярних трендів. Також визначаються цілі ігри, цільова аудиторії та основна концепція геймплею. Розробляється детальний плану створення гри, включаючи часові рамки та ресурси.

Етап проектування. Розробляється ідея гри, визначається жанр, встановлюються основні механіки гри, розробляється концептуальний дизайн. Розробляється детальний дизайн гри, розробляються графічні елементи, музика та звукові ефекти, створюються складні рівні та локації.

Етап розробки включає створення програмного забезпечення для реалізації задуманої гри і визначення її ігрової механіки. Ігрова механіка є набором правил, що забезпечують інтерактивну взаємодію між гравцем і грою.

Основою ігрової механіки є об'єкти, кожен з яких має свої властивості. Вона визначає способи керування об'єктами гравцем, тобто реакцію на певні дії після натискання кнопок.

Графічне оформлення також має велике значення. Якщо раніше графіка в іграх була помітно полігонна, то сьогодні вона стала надзвичайно реалістичною з деталізованим промальовуванням і реалістичним освітленням. Створення образів об'єктів є початковим етапом, на основі яких розробляються 2D або 3D-моделі. Для об'єктів, що рухаються в ході гри, створюється анімація.

Звукове оформлення, хоча й не є обов'язковим, відіграє важливу роль у створенні атмосфери гри та поглибленні гравця. Фонова музика передає емоційний настрій, а звукові ефекти, такі як звуки взаємодії гравця з предметами або звуки кроків персонажів, створюють реалістичну акустичну обстановку.

Етап тестування та відкладки. Проводяться тестування гри, виявляються і виправляються помилки, розробляються варіанти гри для різних пристроїв та платформ.

Етап релізу та підтримки. На цьому етапі гра готова до випуску на ринок, встановлюються ціни та виводяться відповідні рекламні кампанії.

Після випуску гри проводиться постійна підтримка, включаючи виправлення помилок, випуск оновлень, додаткового контенту та взаємодію з користувачами.

1.4. Огляд та аналіз додатків-аналогів

Аналіз та дослідження аналогічних додатків комп'ютерних ігор у жанрі платформер є важливим етапом у процесі створення власної гри. Це дозволяє з'ясувати, які вже існуючі ігри в цьому жанрі наявні на ринку, оцінити їх особливості та успіх, а також виявити можливі прогалини або недоліки, які можуть бути уникнуті у власному проекті.

Платформери стали відомим жанром, що введе багатьох у світ відеоігор. Для деяких Super Mario Bros. стала першою грою, хтось захопився швидкістю Соніка, а інші долали перешкоди у Crash Bandicoot. Цей жанр все ще популярний, і на ПК є багато варіантів платформерів для вибору [4]. Під час виконання кваліфікаційної роботи було розглянуто різні проекти та ігри в мережі Інтернет, серед яких можна виділити кілька найуспішніших додатків.

Першою для аналізу обрано гру Dead Cells. Це платформер-"рогалик" з елементами "метроідванії", розроблений французькою студією Motion Twin. Гра доступна для операційних систем Windows, MacOS, Linux, а також для консолей Nintendo Switch, PlayStation 4 і Xbox One пізніше (рис. 1.2) [5]. У подібних іграх, персонаж у Dead Cells має лише одне життя - якщо він загине, гравець повинен буде почати гру спочатку. Замість постійного бектрекінгу локацій тут смерть персонажа, після якої, втім, зберігаються знання про локації, і можна швидко виявити необхідні проходи та секретні місця.

В грі гравці пробираються через похмурий фентезійний світ, в якому відсутні безпечні зони та точки збереження. У процесі гри, головний герой досліджує різні підземелля, зустрічає й перемагає різних ворогів, а також збирає предмети. У нього є можливість використовувати різні види зброї, яку він знаходить або отримує під час гри, починаючи від мечів і металевих ножів, і до гранат, капканів та турельних установок.

Рис. 1.2. Знімок екрана з гри Dead Cells

Властивості зброї та предметів у грі є випадковими і змінюються в залежності від рівня складності.

Кожен рівень і підземелля генеруються випадковим чином. Гра автоматично створює складні лабіринти, використовуючи раніше підготовлені елементи, та випадковим чином розміщує предмети та ворогів. Гравець має багато можливостей помирати, але в цьому процесі він набуває досвіду і вчиться на своїх помилках.

Dead Cells отримала високі оцінки критиків, завдяки своїй приємній візуальній графіці, складному геймплею та високій загальній якості гри.

Наступною грою для аналізу ринку ігор обрано Hollow Knight. Це комп'ютерна гра в жанрі метроідванія, яку розробила інді-студія Team Cherry. Гра була випущена 24 лютого 2017 року для операційної системи Windows (рис. 1.3) [6]. Через місяць після випуску, розробники також зробили доступною гру на платформи Linux і macOS.

Рис. 1.3. Знімок екрана з гри Hollow Knight

Основна механіка гри Hollow Knight полягає у дослідженні великого і цілісного світу, де гравець зустрічає платформи, секрети та ворогів, з якими йому доводиться боротися. Гравець має можливість досліджувати світ, використовуючи спеціальні поліпшення для переміщення та навички бойової системи, наприклад, відбивання ворожих атак. Як у багатьох іграх метроідваніі, деякі області мають перешкоди, які можна подолати лише після перемоги над певними босами та отримання покращень, які вони дають. Тому гравцям потрібно повертатися до раніше відвіданих зон, щоб знайти секрети або продовжити сюжет. Деякі зони змінюються в процесі гри, що може приносити неочікувані сюрпризи. Кожна зона пов'язана з іншими через кілька переходів, тому гра може бути пройдена різними шляхами. Кожна зона має свою карту, але вона доступна лише після знаходження картографа в цій зоні та придбання карти за місцеву валюту (або покупку її в магазині карт за вищою ціною), після чого гравець сам заповнює карту, відвідуючи різні локації.

Бойова система полягає в використанні головним героєм цвяха — місцевого аналога меча. Атакувати можна в чотири сторони: вгору, вниз,

праворуч, ліворуч. Цвях може бути покращений кілька разів у коваля, також можна вивчити спеціальні прийоми володіння цвяхом у майстрів бою при зустрічі з ними. Удари по ворогам накопичують спеціальний ресурс душу (Soul) в спеціальний посудину, який можна використовувати або лікування, або використання спеціальних прийомів.

Якщо гравець гине, він втрачає всю накопичену валюту і третину основної судини душі, і на місці смерті залишається ворожа тінь (Shade). Якщо вбити тінь, то гравець поверне всю накопичену раніше валюту та головну посудину душі [7].

Hollow Knight володіє красивою графікою, зручним та чуйним керуванням, складним, але захоплюючим геймплеєм.

На останок розглянуто гру Mark of the Ninja. Це комп'ютерна гра в жанрі стелс-екшен, розроблена канадською студією Klei Entertainment для Xbox 360 і PC, випущена в сервісах Xbox Live Arcade і Steam (рис. 1.4) [8].

У грі Mark of the Ninja гравець приймає управління ніндзя, який опиняється в різних ситуаціях, досліджує різні локації і виконує завдання, що вимагають обережності та вправності. Головний герой має вміння ховатися у вентиляційних отворах, каналізаційних люках та за предметами інтер'єру.

Ніндзя розпоряджається різними засобами для виживання та перемоги. Наприклад, у нього є меч, що дозволяє виконувати безшумні вбивства жертв, які не підозрюють небезпеки, а також бамбукові дротики, які можна використовувати для відволікання та знищення джерел світла. Також доступне різне інше спорядження. Гра пропонує гравцеві різні підходи до проходження рівнів, включаючи уникання супротивників, хування, використання відволікань та вирішення головоломок. У процесі проходження гравець може купувати і відкривати додаткове спорядження для атакуючих та відволікаючих дій.

У грі, практично кожну дію, таку як знищення або обхід супротивника, хування тіла чи вимикання живлення пасток, супроводжує нагородження гравця очками. Ці очки можна використовувати для відкриття нових прийомів, поліпшення спорядження та виконання бонусних завдань. Крім

того, гравець може збирати артефакти та вирішувати головоломки, що приносять додаткові бонуси.

Mark of the Ninja - це високоякісний стелс-екшен зі стильним графічним оформленням, яке допомагає створити підходящу атмосферу для всієї історії гри. Вона виділяється серед інших двовимірних ігор свого жанру і надає гравцям велику свободу вибору стратегій та тактик. Гравець може самостійно вирішувати, яким шляхом пройти рівні, і розвивати власний стиль гри.

Рис. 1.4. Знімок екрана з гри Mark of the Ninja

Жанр платформерів відомий своїм елементом колекціонування предметів, які є необхідними для проходження рівнів. Гравець управляє персонажем, який переміщується між платформами різних розмірів, висот та масштабів, що залежить від типу і складності гри. Більшість платформерів мають фантастичні елементи та мальовану графіку. Головними героями таких ігор можуть бути як звичайні люди у незвичайних обставинах, так і міфічні істоти (наприклад, гноми, дракони) або антропоморфні тварини.

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі проведено детальний аналіз предметної області, пов'язаної з комп'ютерними іграми. Дослідження поняття комп'ютерних ігор показало їх значення та важливість у сучасному світі. Класифікація комп'ютерних ігор дозволила розподілити їх на різні категорії залежно від їх основних характеристик та жанрів.

Детально розглянуто основні етапи створення комп'ютерних ігор. Визначено, що процес розробки гри складається з наступних етапів: аналізу та планування, проектування, розробки, тестування та релізу. Детально досліджено кожен з цих етапів та виявлено їх взаємозв'язок та важливість для успішної розробки гри. Цей аналіз дозволив виявити ключові етапи та процеси, які необхідно враховувати під час створення комп'ютерних ігор.

Огляд та аналіз додатків-аналогів були проведені з метою визначити наявність схожих програмних рішень на ринку, а також для виявлення їх переваг та недоліків. Це дозволило отримати уявлення про існуючі тенденції та інновації у галузі комп'ютерних ігор, а також зробити висновки про потенційні можливості для подальшого дослідження та розробки. Огляд показав, що в першу чергу треба звернути увагу на зручність інтерфейсу та дизайну, тому що платформери не є іграми зі складним сюжетом, характеризуються простотою геймплею і основний акцент в цьому жанрі ігор робиться на візуальну складову, тобто зрозумілий інтерфейс та гарна картинка.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Мова програмування C#

Розробка програмного забезпечення зазвичай вимагає вибору мови програмування, яка найкраще відповідає вимогам проекту. Однією з популярних мов програмування є C# (C-Sharp), яка розроблена компанією Microsoft. C# є об’єктно-орієнтованою мовою програмування зі збіркою потужних функцій та можливостей, які дозволяють розробникам створювати різноманітне програмне забезпечення.

C# входить до сімейства мов C/C++, але водночас має власні особливості. Вона базується на об'єктно-орієнтованому підході, що дозволяє розробникам моделювати реальні об'єкти та їх взаємодію. Чистий синтаксис мови спрощує розробку та зрозуміння коду, а автоматичне управління пам'яттю полегшує роботу з пам'яттю та зменшує ймовірність помилок.

Однією з найбільш привабливих особливостей мови C# є її простий та зрозумілий синтаксис. Це робить її легко засвоюваною для новачків у програмуванні, а також допомагає зменшити кількість помилок у коді. Крім того, C# підтримує автоматичне управління пам'яттю, що сприяє ефективному використанню ресурсів та зменшує ризик виникнення утечок пам'яті.

C# має підтримку кросплатформеної розробки завдяки проекту .NET Core. Це означає, що можна написати код на C# і запускати його на різних операційних системах без необхідності внесення значних змін у вихідний код. Це дозволяє створювати кросплатформенні додатки, які працюють на

Windows, macOS та Linux без додаткового зусилля. Крім того, C# також може бути використана для розробки мобільних додатків під управлінням операційних систем Android та iOS за допомогою фреймворка Xamarin.

Ігрова розробка є ще одним сегментом, де мова C# виявляє себе дуже ефективно. Ігровий двигун Unity, який є одним з найпопулярніших ігрових движків у галузі, використовує мову C# як основну мову програмування. Це дозволяє розробникам створювати складні та захоплюючі ігри з використанням широкого спектру функцій та бібліотек, що доступні у мові C#.

Окрім стандартних функцій та бібліотек, існує також велика спільнота розробників, яка активно працює над розширенням мови C# та створенням власних бібліотек і фреймворків. Це сприяє активному обміну знаннями та досвідом, а також надає доступ до великої кількості готових рішень та кодових зразків, які можуть значно полегшити розробку програмного забезпечення.

Мова C# відзначається високим рівнем абстракції, зрозумілим синтаксисом та широкими можливостями, що робить її потужним інструментом для розробки програмного забезпечення в різних галузях, включаючи ігрову індустрію. Використання мови C# дозволяє розробникам ефективно створювати високоякісні ігри та програми з меншими зусиллями і часом, що сприяє прискоренню процесу розробки та досягненню бажаних результатів.

2.2. Аналіз існуючих ігрових движків

Ігрові движки є потужними інструментами для розробки ігор, які дозволяють розробникам ефективно використовувати свої знання та ресурси для створення як 2D, так і 3D ігор. Ці движки надають набір інструментів, які спрощують такі завдання, як програмування геймплею, робота з графікою, фізикою, звуком, штучним інтелектом та іншими аспектами гри.

Вони надають основну архітектуру, на яку розробники можуть покластися, щоб створити власну гру. Ігрові движки забезпечують широкий

спектр функцій і інструментів, таких як введення, рендеринг, сценарії, виявлення зіткнень, штучний інтелект і багато іншого.

Ігрові движки дозволяють розробникам ефективно використовувати ці готові компоненти, замість того, щоб розробляти їх з нуля. Це дозволяє зосередитися на унікальних аспектах гри, таких як дизайн персонажів, текстури, фізика, геймплей та інші елементи. Використання ігрових движків допомагає прискорити процес розробки ігор, зменшити витрати на розробку та забезпечити високу якість готового продукту.

При виборі ігрового движка розробники ігор повинні враховувати свої потреби і цілі проекту, а також оцінювати доступні рішення і їх можливості. Кожен ігровий движок має свої особливості, переваги і обмеження, і важливо знайти той, який найкраще відповідає вимогам проекту. [9]

Існує велика кількість ігрових двигунів, які мають багато відмінностей, як, наприклад, безкоштовний та платний доступ. Вони включають прості бібліотеки для популярних мов програмування, а також повноцінні редактори з розширеним функціоналом.

Для аналізу були взяті одні з найпопулярніших ігрові двигуни, а саме: Unity, Unreal Engine та CryEngine.

2.2.1. Unity

Unity - це ігровий двигун та інтегроване середовище розробки, що використовується для створення ігор, віртуальної реальності (VR), доповненої реальності (AR) та інтерактивних додатків. Він має широкий спектр можливостей та інструментів для розробників, що дозволяє їм створювати професійні ігри з різноманітними геймплейними механіками та візуальними ефектами.

Основна мета ігрового движка, такого полягає в наданні розробникам ігор потужного та надійного набору інструментів і максимально спростити використання движка для розробників ігор будь-якого рівня кваліфікації (у тому числі для початківців). Компанія Unity Technologies також розширила своє охоплення в інших галузях, зосередившись на розробці 3D у реальному

часі, що зробило Unity одним із найпотужніших ігрових двигунів. [10]

Unity надає можливість розробляти ігри для різних платформ, включаючи комп'ютери, мобільні пристрої, ігрові консолі та віртуальну реальність. Це робить Unity універсальним інструментом для досягнення широкої аудиторії та максимізації потенційної бази користувачів.

Один з ключових аспектів Unity - це його гнучкість та розширюваність. Він має велику кількість доступних плагінів та розширень, які дозволяють розробникам розширити функціональність двигуна та використовувати спеціалізовані інструменти для своїх проектів.

Інтерфейс Unity є інтуїтивно зрозумілим та добре організованим, що полегшує роботу розробників та зменшує час, потрібний для освоєння інструментів та процесу розробки. Він також має потужні функції для роботи з графікою, анімацією, фізикою, штучним інтелектом та звуком, що дозволяє створювати вражаючі візуальні та аудіоелементи для ігор (рис. 2.1).

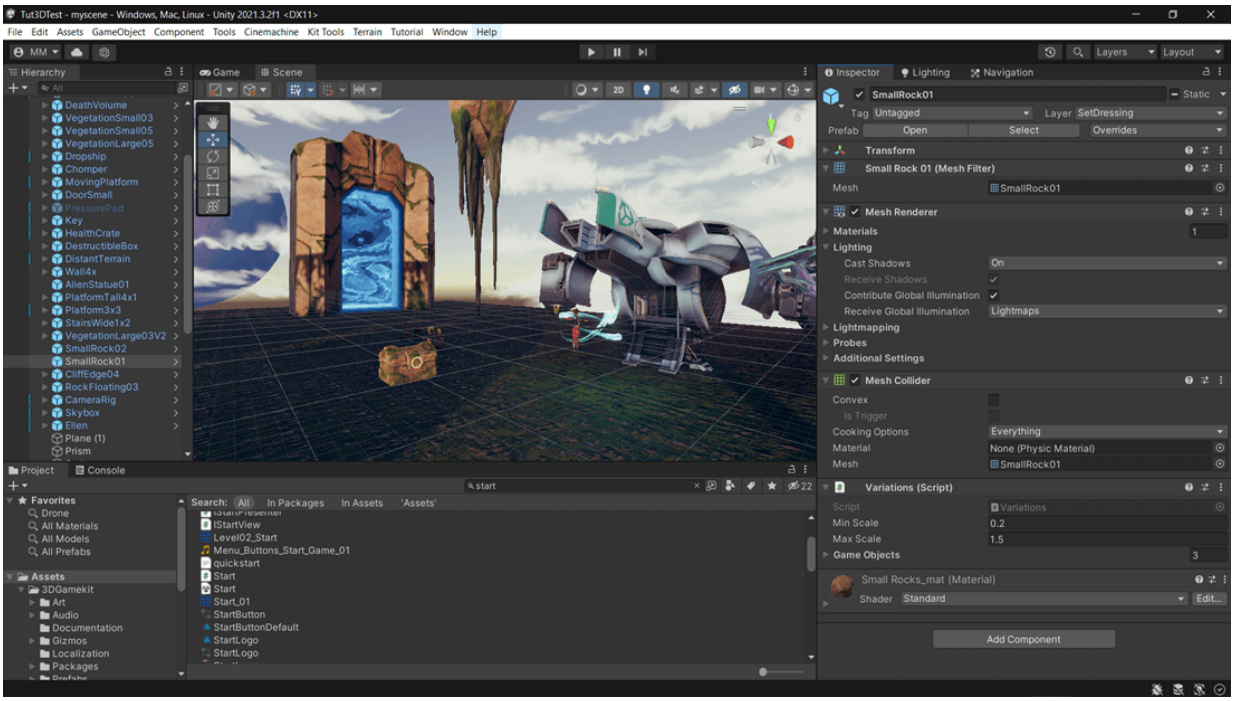

Рис. 2.1. Інтерфейс ігрового двигуна Unity

Unity має наступні переваги:

– Хрестплатформеність. Unity дозволяє розробляти ігри для різних платформ, включаючи комп'ютери, мобільні пристрої та консолі;

– Розширюваність. Unity має велику кількість плагінів та розширень,

що дозволяють розширити функціонал двигуна та використовувати спеціалізовані інструменти;

– Інтуїтивний інтерфейс. Unity має добре організований та легкий у використанні інтерфейс, що спрощує процес розробки;

– Потужність: Unity надає потужні функції для роботи з графікою, анімацією, фізикою, штучним інтелектом та звуком.

Недоліки Unity включають:

– Висока вартість. Деякі функціональності та додаткові плагіни Unity можуть вимагати додаткових витрат;

– Обмеження продуктивності. В деяких випадках Unity може бути менш продуктивним у порівнянні з іншими ігровими двигунами при обробці великих обсягів даних або складних графічних ефектів;

– Залежність від розробників. Unity оновлюється та розвивається з часом, що може вимагати від розробників постійного оновлення та знання останніх версій та функціональності.

2.2.2. Unreal Engine

Для спрощення завдань програмістів у 1996 році було створено Unreal Engine — ігровий двигун, який спочатку використовувався для розробки ігор від першої особи. Згодом, його застосування значно розширилося (рис. 2.2). Це стало можливим завдяки простоті розробки і відсутності потреби в освоєнні складної мови програмування C++.

На початковому етапі використання Unreal Engine було платним. Однак, у 2015 році розробники вирішили випустити безкоштовну пробну версію, яка набула значної популярності. Це пояснюється можливістю створювати складні ігри, що не можна було розробити за допомогою інших безкоштовних двигунів.

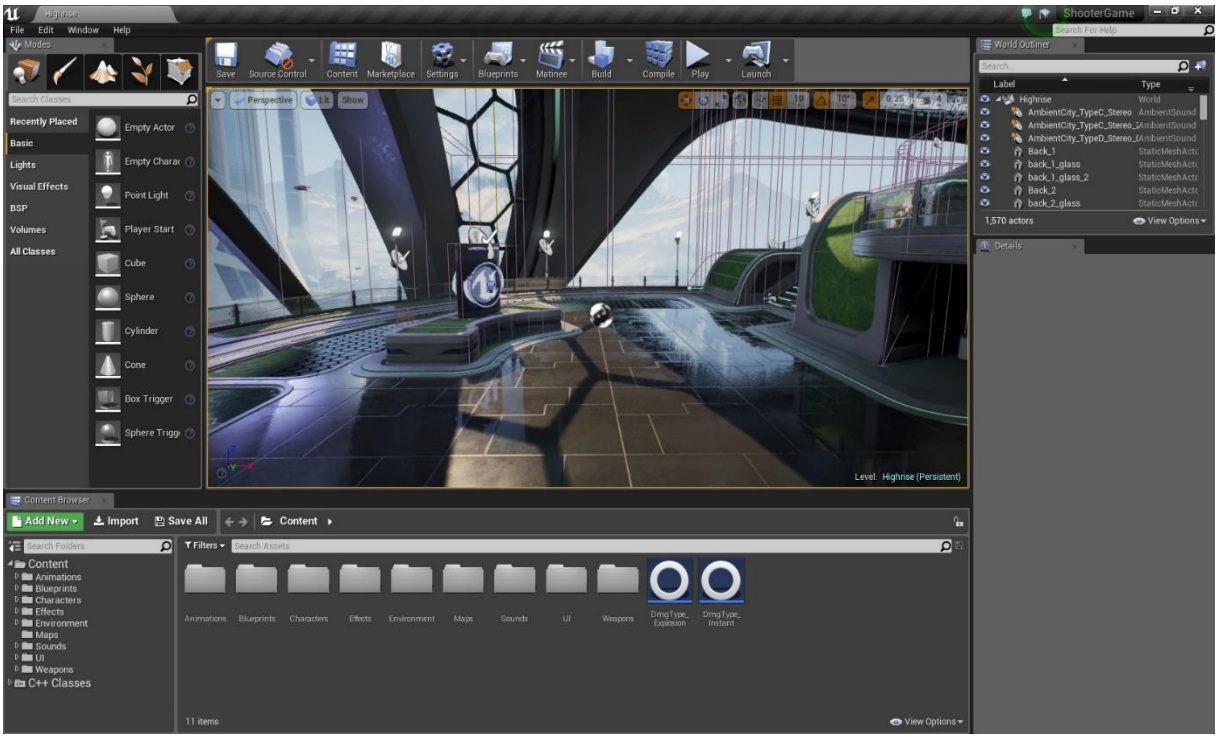

Рис. 2.2. Інтерфейс середовища розробки Unreal Engine

Розроблений на мові програмування C++ рушій Unreal Engine дозволяє створювати ігри для різних операційних систем і платформ, зокрема Microsoft Windows, Linux, Mac OS і Mac OS X, консолей Xbox, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3, Dreamcast і Nintendo GameCube. У грудні 2009 року Марк Рейн продемонстрував роботу рушія Unreal Engine 3 на iPod Touch і iPhone 3GS. У березні 2010 року також була показана робота рушія на комунікаторі Palm Pre, що працює на мобільній платформі webOS. [11]

Особливості створення ігор на Unreal Engine:

– Рушій Unreal Engine має модульну систему компонентів, що дозволяє переносити ігри та інші розробки з однієї платформи на іншу;

– Будучи розробленим на мові програмування C++, Unreal Engine дозволяє створювати ігри для платформ, як Windows, Mac OS і Linux;

– Останнім часом розробники Unreal Engine зосереджуються на створенні кейсів для платформи Android, що дозволяє створювати ігри для мобільних пристроїв. [12]

Переваги Unreal Engine:

– Простота використання. Цей рушій є дуже легким у роботі, його можна

освоїти навіть розробнику без досвіду програмування. Навігація та налаштування подібні до інших програмних засобів;

– Підтримка великої кількості функцій, що дозволяє розробникам не маючи обмежень створювати свої продукти;

– Універсальність та доступність. Unreal Engine широко використовується як досвідченими розробниками, так і новачками у галузі створення ігор;

– Активна спільнота користувачів Unreal Engine, яка постійно створює навчальні матеріали, ділиться порадами та допомагає один одному вирішувати проблеми, пов'язані з використанням рушія;

– Вбудована система візуального скриптингу, яка дозволяє легко будувати ігрову логіку;

– Умовно безкоштовний рушій. Розробники не сплачують за використання Unreal Engine, якщо їх гра не відноситься до високоуспішних. У випадку, якщо гра заробляє понад $1,000,000, відсоток від доходу складає 5%. Це справедливі та вигідні умови; [13]

– Мультиплатформеність. Цей рушій підтримує розробку ігор для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Завдяки цьому розробники можуть створювати ігри, які можна запускати на різних пристроях і платформах, забезпечуючи широкий охоплення аудиторії.

Недоліки Unreal Engine:

– Високі вимоги до апаратного забезпечення. Unreal Engine вимагає потужного обладнання для ефективної роботи. Для розробки та запуску ігор на Unreal Engine необхідно мати комп'ютер з достатньою кількістю оперативної пам'яті та потужним графічним процесором;

– Великий обсяг проекту. Проекти, розроблені на Unreal Engine, можуть мати значний обсяг через використання різноманітних компонентів та ресурсів, таких як моделі, текстури, звукові ефекти тощо. Це може призвести до збільшення обсягу виконуваного файлу гри та вимагати більшої потужності

обробки ЦП та обсягу пам'яті для ефективного відтворення гри.

2.2.3. CryEngine

CryEngine є ігровим рушієм, розробленим німецькою компанією Crytek, і який вперше був використаний у відеогрі Far Cry (рис. 2.3). [14]

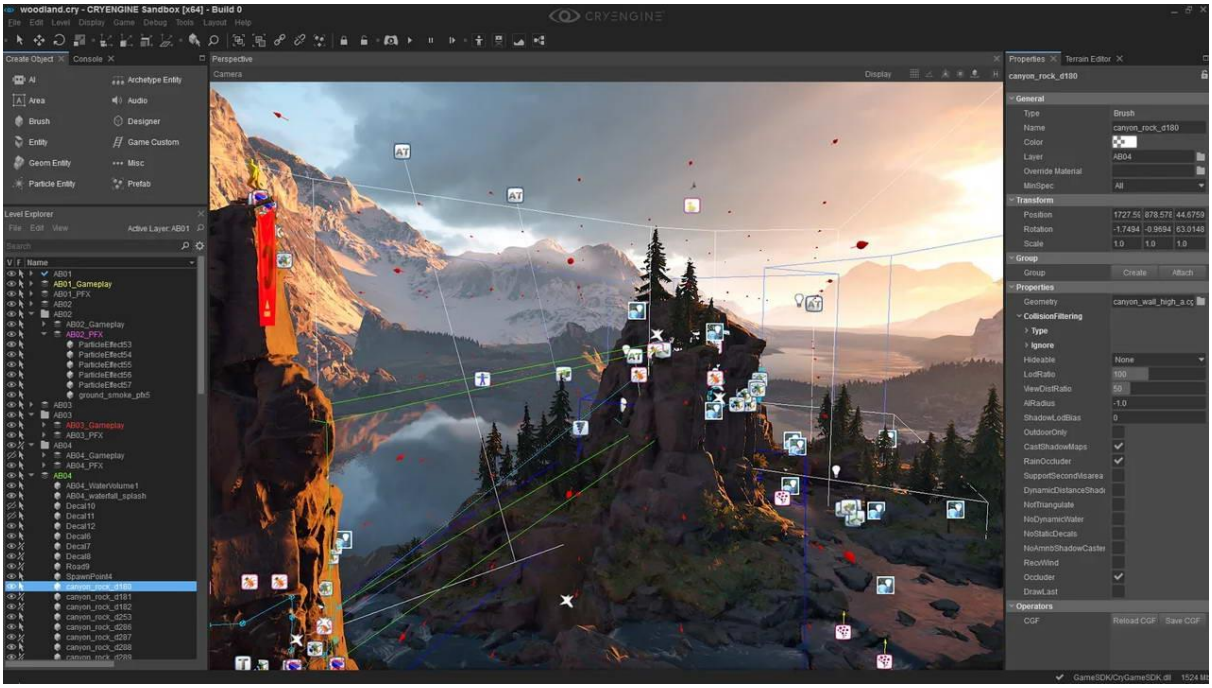

Рис. 2.3. Інтерфейс ігрового двигуна CryENGINE

Цей движок відомий своєю потужною графікою та широкими можливостями, які надають розробникам ігор велику свободу творчості. Він має потужні редактори для рівнів, анімації, штучного інтелекту та фізики, що дозволяє реалізувати складні механіки та інтерактивність у своїх іграх.

Рушій CryENGINE має широкий функціонал і можливості, які дозволяють розробникам використовувати передову 3D-графіку та масштабовані обчислення. Цей двигун має значний набір вбудованих інструментів, які дозволяють створювати ігри без необхідності використання сторонніх програм. До цього часу нікому не вдалося повністю використати всі можливості CryENGINE, що робить його ідеальним інструментом для створення ігор нового покоління. Вбудована система анімації включає параметричну скелетну анімацію, процедурне деформування руху і систему

індивідуальних персонажів.

Особливості програми CryENGINE:

– Можливість створення онлайн ігор;
– Підтримка платформ для ПК, Xbox і PlayStation;
– Використання передових графічних технологій;
– Наявність зручного редактора рівнів;
– Можливості оптимізації гри під GPU-рендеринг;
– Імпорт моделей і текстур з графічних редакторів Maya і 3ds Max;
– Програмування місій у вільному форматі;
– Зручний режим тестування ігрового процесу і меню ігор;
– Створення великих закритих і відкритих локацій;
– Використання передового штучного інтелекту;
– Наявність всіх необхідних інструментів для повноцінної розробки без використання сторонніх програм;
– Користувальницький інтерфейс рушія реалізований англійською мовою. [15]

Переваги CryENGINE:

– Графічна якість. CryENGINE відомий своєю вражаючою графікою та реалістичними візуальними ефектами. Він забезпечує високу деталізацію, реалістичне освітлення і фотореалістичні текстури, що дозволяє створювати неймовірно реалістичні ігрові світи;

– Фізична модель. CryENGINE має потужну фізичну модель, яка дозволяє реалістично відтворювати рухи об'єктів, взаємодію об'єктів з навколишнім середовищем та передавати різні фізичні ефекти;

– Інструменти розробки. CryENGINE надає розширений набір інструментів для розробки ігрових проектів. Він має зручний візуальний редактор, що дозволяє швидше створювати та налаштовувати ігрові об'єкти, механіки та інтерактивні елементи.

Недоліки CryENGINE:

– Високі вимоги до апаратного забезпечення. Один з недоліків CryENGINE полягає у високих вимогах до обчислювальної потужності

комп'ютера. Це може вплинути на доступність розробки для деяких користувачів з менш потужними системами;

– Складність в освоєнні. CryENGINE має досить складну структуру та інтерфейс, що може вимагати часу та зусиль для освоєння. Необхідно мати досвід у розробці ігор або вивчати документацію та посібники для ефективного використання цього движка;

– Ліцензійні витрати. Використання CryENGINE пов'язане з ліцензійними витратами, особливо для комерційних проектів. Це може стати чинником обмеження для незалежних розробників або студій з обмеженими фінансовими ресурсами.

За результатами проведеного дослідження засобів розробки програмного забезпечення визначено, що середовище програмування Unity найкраще підходить для розробки комп’ютерної гри у жанрі 2D платформер, оскільки має в цілому зрозумілий набір інструментів, зручний інтерфейс, відносно простий синтаксис та має у наявності вільну ліцензію і навчальну документацію.

2.3. Середовище розробки Microsoft Visual Studio

Написання коду в обмежений час - це надзвичайно важлива навичка, якою повинен володіти кожен розробник програмного забезпечення, щоб бути конкурентоспроможним на ринку і досягати високої продуктивності. Вибір правильного інтегрованого середовища розробки (IDE) або редактора коду відіграє важливу роль у створенні та підтримці високоякісного коду. З урахуванням зростаючої кількості коду та різноманітних мов програмування, важливо, щоб розробники програмного забезпечення обирали найбільш зручні та правильні редактори для досягнення своїх цілей.

У Unity реалізація програмного коду відбувається за допомогою Microsoft Visual Studio, яке має можливість інтеграції додаткового функціоналу для Unity.

Microsoft Visual Studio є інтегрованим середовищем розробки

програмного забезпечення, розробленим компанією Microsoft (рис. 2.4). Це середовище дозволяє створювати різноманітні програмні продукти, включаючи консольні програми, програми з графічним інтерфейсом, такі як віконні додатки Windows Forms, а також веб-додатки та інші. [16]

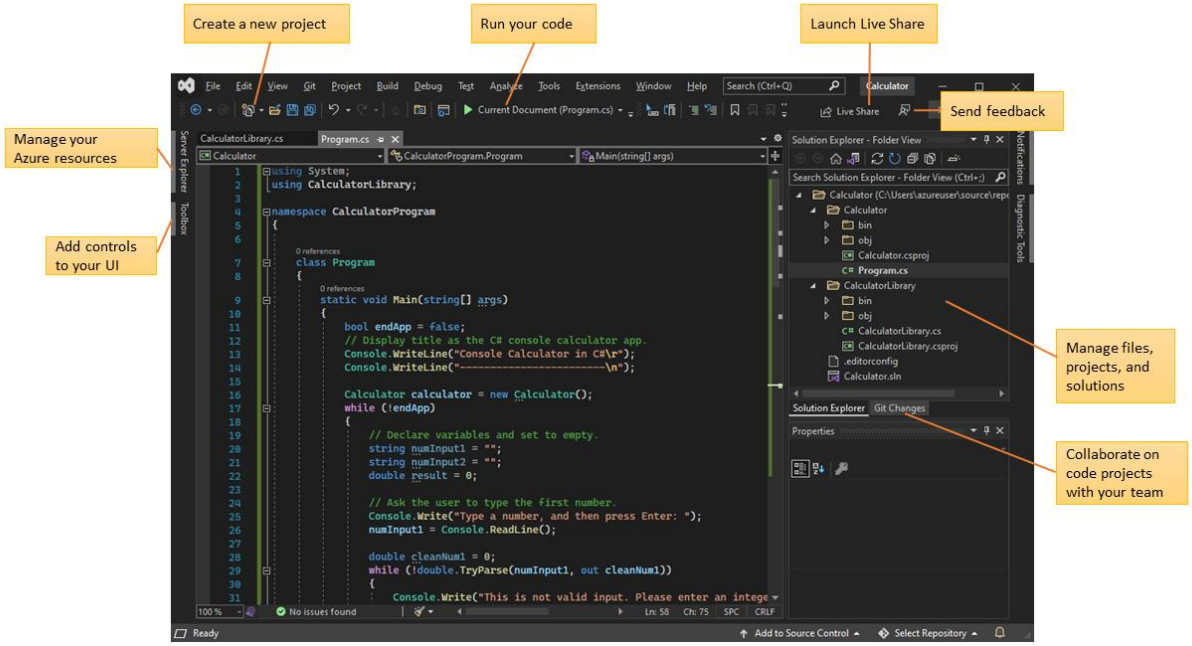

Рис. 2.4. Інтерфейс Visual Studio

Основні характеристики і можливості Microsoft Visual Studio підтримує широкий спектр мов програмування, включаючи C++, C#, Visual Basic, F#, Python та інші. Воно також має інтегровані інструменти розробки, такі як візуальний редактор коду, відладчик, систему контролю версій, дизайнер інтерфейсу користувача і багато інших.

Microsoft Visual Studio забезпечує широкі можливості для командної розробки, зокрема спільну роботу над проектами, інтегровану зворотну зв'язок та засоби збірки та розгортання програм. Воно також має розширювану архітектуру, що дозволяє встановлювати сторонні плагіни та розширення для покращення функціональності.

Середовище розробки Microsoft Visual Studio є потужним інструментом для професійної розробки програмного забезпечення. Воно надає розробникам зручне та ефективне середовище для створення, налагодження

та управління проектами. Завдяки його функціональності і розширюваності, розробники можуть ефективно працювати над своїми проектами та реалізувати свої ідеї.

Основні переваги та можливості Microsoft Visual Studio включають наступне:

– Розширена підтримка мов програмування. Visual Studio підтримує різні мови програмування, такі як C++, C#, Visual Basic, F#, Python і багато інших. Це дає розробникам можливість вибрати мову, яка найкраще відповідає їх потребам і навичкам;

– Інтегровані інструменти розробки. Visual Studio має багато вбудованих інструментів, які полегшують розробку програмного забезпечення. Це включає в себе візуальний редактор коду, відладчик, систему контролю версій, дизайнер інтерфейсу користувача, профілер продуктивності та інші;

– Розширюваність і плагіни. Visual Studio можна розширювати за допомогою різних плагінів і розширень, які надають додаткові функціональні можливості. Розробники можуть встановлювати сторонні плагіни для підтримки специфічних технологій, фреймворків або інструментів розробки;

– Інтеграція з іншими продуктами Microsoft є однією з сильних сторінок Visual Studio. Це середовище розробки має глибоку інтеграцію з різними продуктами Microsoft, включаючи Azure, SQL Server, SharePoint та інші. Завдяки цій інтеграції, розробники отримують зручний доступ до функціональності цих продуктів та можуть легко взаємодіяти з ними під час розробки своїх проектів.

– Підтримка командної розробки. Visual Studio забезпечує розширені можливості для командної розробки, включаючи системи контролю версій, спільну роботу над проектами, інтегровану зворотна зв'язок та засоби збірки та розгортання програм.

Незважаючи на багато переваг, Visual Studio також має деякі недоліки, такі як:

– Великі вимоги до системи. Visual Studio може вимагати високих обчислювальних ресурсів та диского простору. Це може бути обмеженням для менш потужних комп'ютерів або віртуальних середовищ;

– Вартість. Деякі версії Visual Studio, особливо для комерційних використань, мають високу вартість ліцензування. Це може бути фактором, що обмежує доступність для незалежних розробників з обмеженими фінансовими ресурсами;

– Великий розмір інсталяційного пакету. Інсталяційний пакет Visual Studio може бути великим, особливо з урахуванням включених функціональних компонентів і підтримуваних мов програмування.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі розглянуто засоби програмної реалізації ігрового застосунку. Була розглянута мова програмування C#, яка є популярною та широко використовується для розробки відеоігор. Вона володіє потужними можливостями та надійною синтаксичною структурою, що дозволяє зручно реалізовувати функціонал ігрових проектів.

Було проведено аналіз існуючих ігрових двигунів, зокрема Unity, Unreal Engine та CryEngine. Виявлено, що кожен з них має свої особливості, переваги та недоліки. Цей аналіз дав змогу обґрунтувати вибір Unity як ігрового двигуна для реалізації проекту.

Досліджено та проаналізовано середовище розробки Microsoft Visual Studio, яке є одним з найпопулярніших інструментів для розробки програмного забезпечення. Його потужність, зручність і багатий набір інструментів дозволяють розробникам ефективно працювати над проектами та забезпечувати високу якість коду.

Засновуючись на проведеному аналізі та дослідженні, вибір мови програмування C#, ігрового двигуна Unity та середовища розробки Microsoft Visual Studio є обґрунтованим і оптимальним для реалізації проекту гри у жанрі 2D платформер. Ці засоби надають широкі можливості для розробки ігрового функціоналу, забезпечують зручну роботу розробників та дозволяють досягти бажаного результату в процесі реалізації проекту.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1. Концепція та сценарій гри

Вибраним жанром гри є платформер. Платформер – це жанр комп'ютерних ігор, що належить до категорії аркадних ігор. Основними елементами геймплею є стрибання по платформам, подолання різних перешкод, а також завершення одного рівня і перехід на наступний, який зазвичай є складнішим або має інші відмінності. Хоча ігри цього жанру не є складними у реалізації, вони здатні захопити гравця на тривалі години і зацікавити широку аудиторію.

У центрі сюжету знаходиться головний герой, а саме дракон, який мешкає у незвіданому світі. Він має крила, завдяки яким може декілька секунд парити в небі, та вміє наносити урон у вигляді вогняного дихання.

Головне завдання дракона – це бігти вперед, долаючи різноманітні перешкоди. На шляху головному герою траплятимуться безліч небезпек, які заважатимуть і будуть намагатися його вбити. Чим далі просуватиметься дракон, тим небезпечніше ставатимуть локації, і тим складніше буде бігти вперед. Чим довше користувач гратиме в цю гру, тим краще і вправніше ставатиме його керування головним персонажем, що дозволить проходити якомога більше рівнів і встановлювати нові особисті рекорди.

3.2. Цільова аудиторія

Визначення цільової аудиторії має велике значення при розробці гри, оскільки воно допомагає уточнити інтереси, потреби і вимоги гравців, на яких спрямована гра. Це допомагає розробнику створити гру, яка відповідає потребам цільової аудиторії, та підвищити її привабливість для цих гравців.

Визначення цільової аудиторії також допомагає в ефективному маркетингу і просуванні гри, орієнтуючи рекламні кампанії та стратегії на цільову аудиторію. Враховуючи інтереси та вимоги гравців, розробники можуть створити більш успішну та задоволену аудиторію гру.

Цільовою аудиторією гри є геймери, які цікавляться жанром платформерів і мають інтерес до 2D ігор. Це можуть бути як любителі ігор, так і досвідчені гравці, які насолоджуються викликами, які надає цей жанр. Оскільки платформери мають простий, але веселий геймплей, вони можуть привернути увагу широкого спектру гравців, включаючи як дорослих, так і підлітків.

3.3. Графічне оформлення

Основною складовою будь-якої комп'ютерної гри є її візуальна частина. Для початку опису графічного оформлення потрібно розглянути основні терміни, що використовуються у геймдизайні, такі як "спрайт" і "тайл".

Тайл - це малий повторюваний фрагмент, що використовується для створення великих зображень, і цей процес називається "тайлова графіка".

Спрайт - це графічне зображення об'єкта, що використовується у двовимірних іграх і може містити кілька зображень для анімації об'єкта.

Для реалізації графічної частини прототипу використовується готовий набір двовимірних спрайтів з офіційного магазину Unity Asset Store, який включає спрайти різних жанрів і типів ігор. Було обрано спрайти і текстури, що відповідають концепції гри. Для роботи зі спрайтами використовується вбудована функція Sprite Editor, яку має Unity (рис. 3.1).

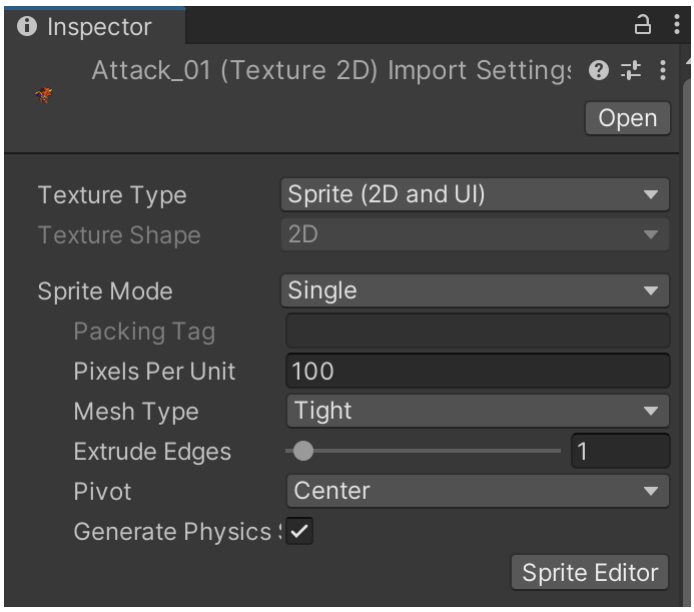

Рис. 3.1. Вбудована функція Sprite Editor

Функція Sprite Editor у Unity надає можливість відкривати великі спрайт-об'єкти та розділяти їх на складові частини для подальшої анімації.

3.3.1. Спрайти головного персонажу

Для візуального представлення головного героя був використаний спрайт дракона, який складається з різних спрайтів для створення анімації декількох дій, таких як стояння, біг, стрибки, отримання пошкоджень та смерть (рис. 3.2). Додатково завантажено спрайт для боротьби з ворогами у вигляді вогняного дихання (рис. 3.3).

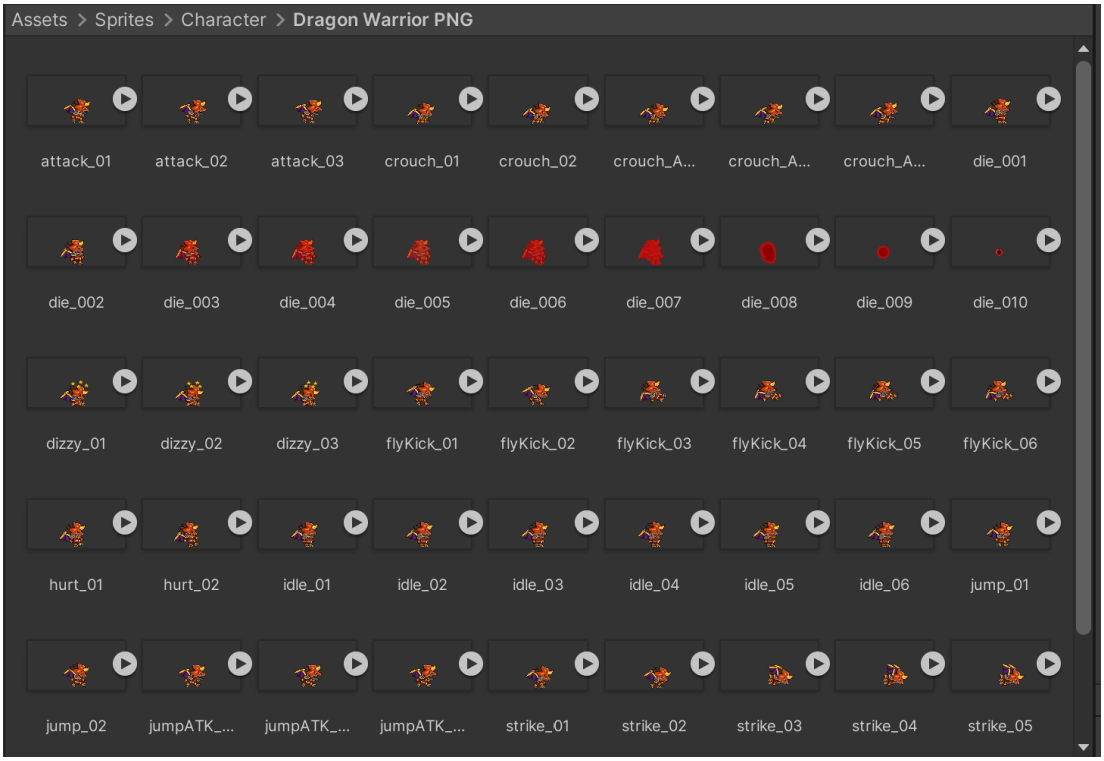

Рис. 3.2. Спрайти головного персонажу

Рис. 3.3. Спрайт вогняного шару

3.3.2. Спрайти інтерфейсу

Наступним кроком у реалізації гри є вибір спрайтів для фонового зображення. Крім того, також були використані візуальні елементи для складання графічного інтерфейсу гри.

Для використання в якості фонового зображення були вибрані спрайти, доступні в офіційному магазині Unity Asset Store. (рис. 3.4).

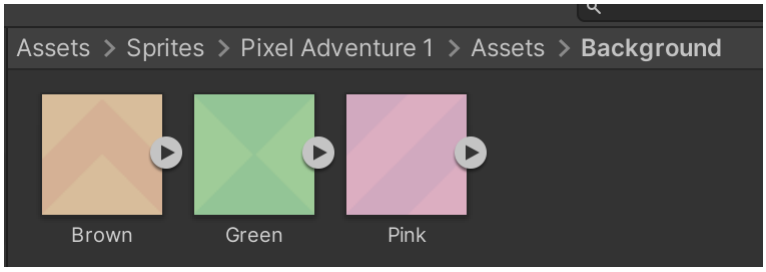

Рис. 3.4. Спрайти фонового зображення

Для відображення візуальних елементів гри було використано спрайти платформ у вигляді землі з травою, по яким буде стрибати та бігати ігровий персонаж (рис. 3.5).

Самі спрайти були поділені за допомогою функції Sprite Editor.

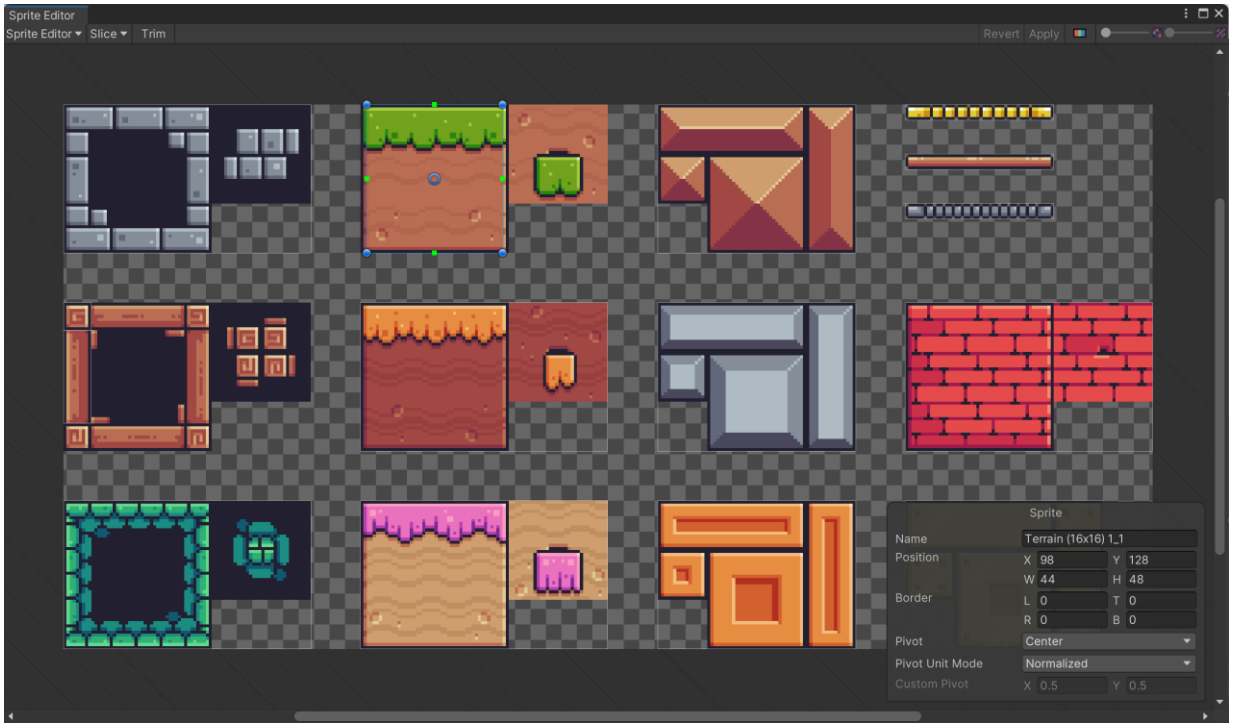

Рис. 3.5. Спрайти платформ

3.3.3. Спрайти ворога

Для графічного відображення ворога обрано спрайт лицаря, який буде нападати на дракона. Він, так само як і головний персонаж, буде анімованим, тому використовуються спрайти для створення анімації зазнавання шкоди та смерті (рис. 3.6). Лицар, як і дракон, матиме вміння стріляти вогняним шаром для нанесення ураження.

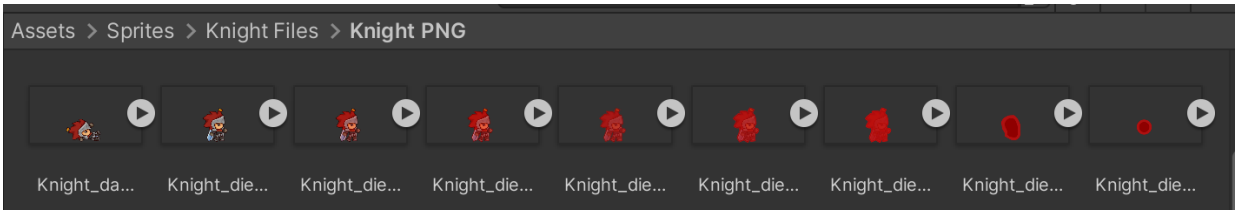

Рис. 3.6. Спрайти ворога

3.4. Етап проектування гри

Для розробки комп'ютерної гри було використано ігровий двигун Unity

та середовище розробки програмного забезпечення Microsoft Visual Studio 2019 для написання програмного коду.

При запуску Unity з'являється вікно проекту (рис. 3.7), яке містить різні панелі, такі як вікно Scene, вікно Animator і вікно Game. Вікно Scene використовується для створення композиції рівня та додавання нових об'єктів, вікно Animator надає можливість створювати і редагувати анімації об'єктів, а вікно Game показує перспективу з камери, демонструючи вигляд гри на поточний момент.

Ліворуч розташоване вікно Hierarchy, в якому відображаються всі об'єкти, які беруть участь у поточній сцені.

Праворуч знаходиться вікно Inspector, яке відображає поточні властивості обраного об'єкта.

У нижній частині програми розташовані три вікна: Project, Animation і Console. У вікні Project відображаються всі об'єкти, що були додані до поточної гри, включаючи скрипти, анімації та інше. Вікно Animation використовується для створення зв'язків і правил переходу між різними анімаціями. Console призначена для відображення помилок та винятків, які можуть виникнути під час роботи гри.

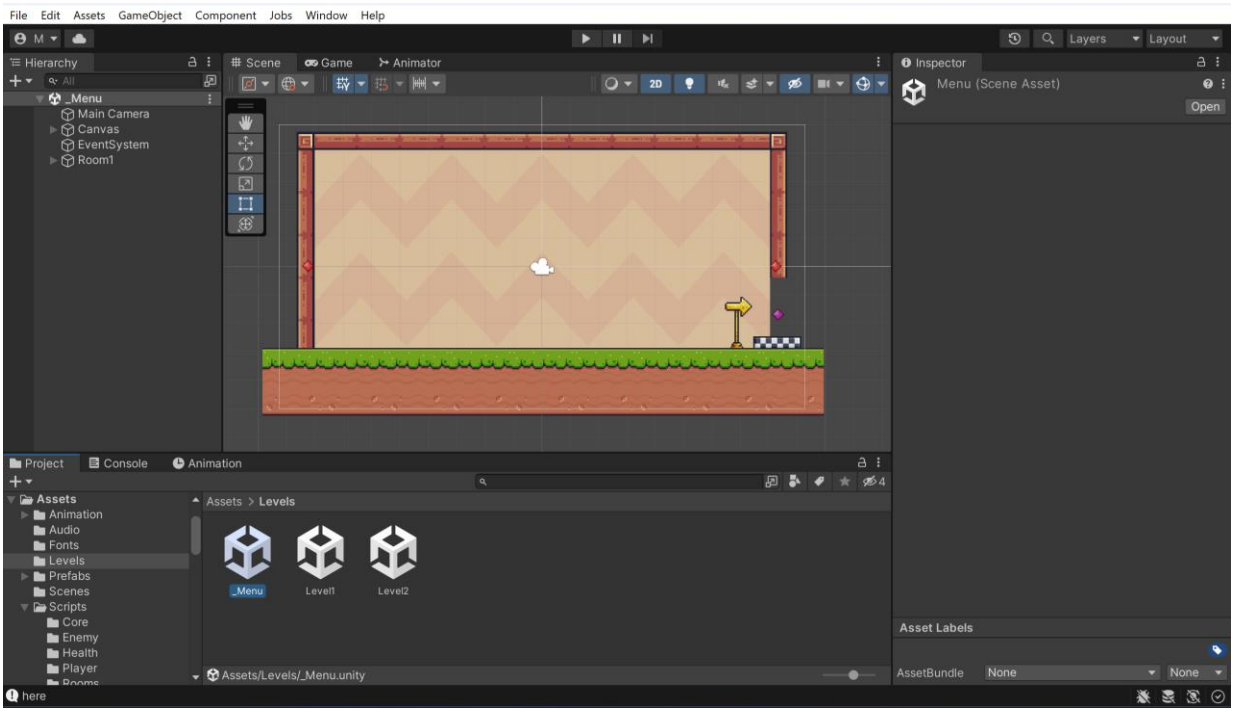

Рис. 3.7. Вікно проекту Unity

3.4.1. Фізичні властивості об’єктів

Насамперед, першим етапом створення прототипу гри є надання об’єктам фізичних властивостей. Перш за все, необхідно додати до сцени платформу, якою буде переміщатися головний герой і надати їй фізичні властивості. Без неї гравець просто буде нескінченно падати у просторі, тому що програма не буде вважати платформи фізичними об’єктами.

Спочатку необхідно додати сам спрайт платформи, який був створений раніше. Для цього достатньо перетягнути його з папки у вікно проекту. Після цього спрайт має з'явитися серед усіх об'єктів, які вже були додані до гри. Потім необхідно додати обрану платформу до поточної сцени. Це можна зробити двома способами: перетягнути з вікна Project до вікна Scene або спочатку вікна Hierarchy створити окремий порожній об'єкт, до якого потім застосувати спрайт платформи.

За допомогою функціоналу Unity спочатку додається компонент Box Collider 2D створеній платформі, що надає їй фізичну форму куба. Це прямокутник у формі з визначеним положенням, шириною та висотою в локальному координатному просторі спрайту. [17]

У Unity існує два види колайдерів: для 2D та 3D об'єктів. Так як створюється двовимірна гра, то і колайдер потрібно додавати двовимірний.

Щоб додати колайдер до об'єкта, необхідно виділити його у вікні Hierarchy, а потім додати компонент колайдера у вікні Inspector за допомогою кнопки Add Component (рис. 3.8). Для налаштування розмірів колайдера необхідно скористатися кнопкою Edit Collider.

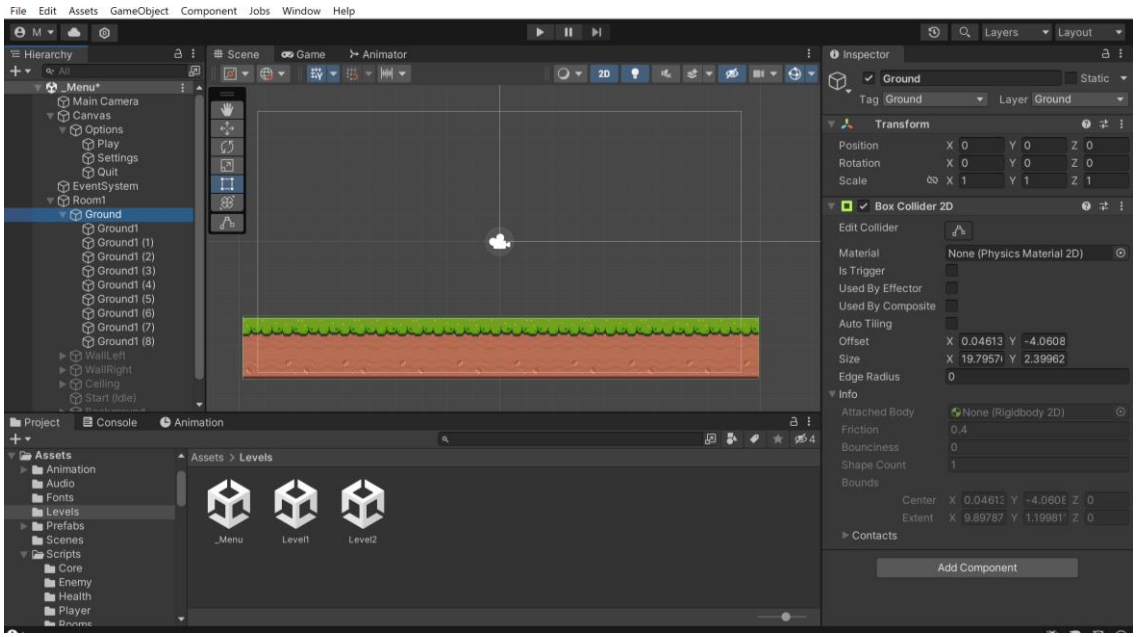

Рис. 3.8. Платформа у вікні Scene і її властивості

Тепер треба створити головного героя гри. Для цього так само додається Box Collider 2D для надання фізичних властивостей об’єкту і додається до нього уже обраний спрайт героя. Наступним кроком є додавання компоненту Rigidbody 2D (рис. 3.9).

Компонент Rigidbody 2D в Unity є одним з ключових елементів для моделювання фізичної поведінки об'єктів у двовимірному просторі гри. Він надає можливість об'єктам рухатись, взаємодіяти з фізичним середовищем та реагувати на сили, що діють на них.

Крім цього, компонент Rigidbody 2D працює в парі з коллайдерами (Collider 2D), що дозволяє об'єктам взаємодіяти з іншими об'єктами, виявляти зіткнення та реагувати на них.

Загалом, компонент Rigidbody 2D допомагає створити реалістичну фізичну поведінку об'єктів у двовимірному просторі гри, що є важливим аспектом для багатьох типів ігор, зокрема платформерів.

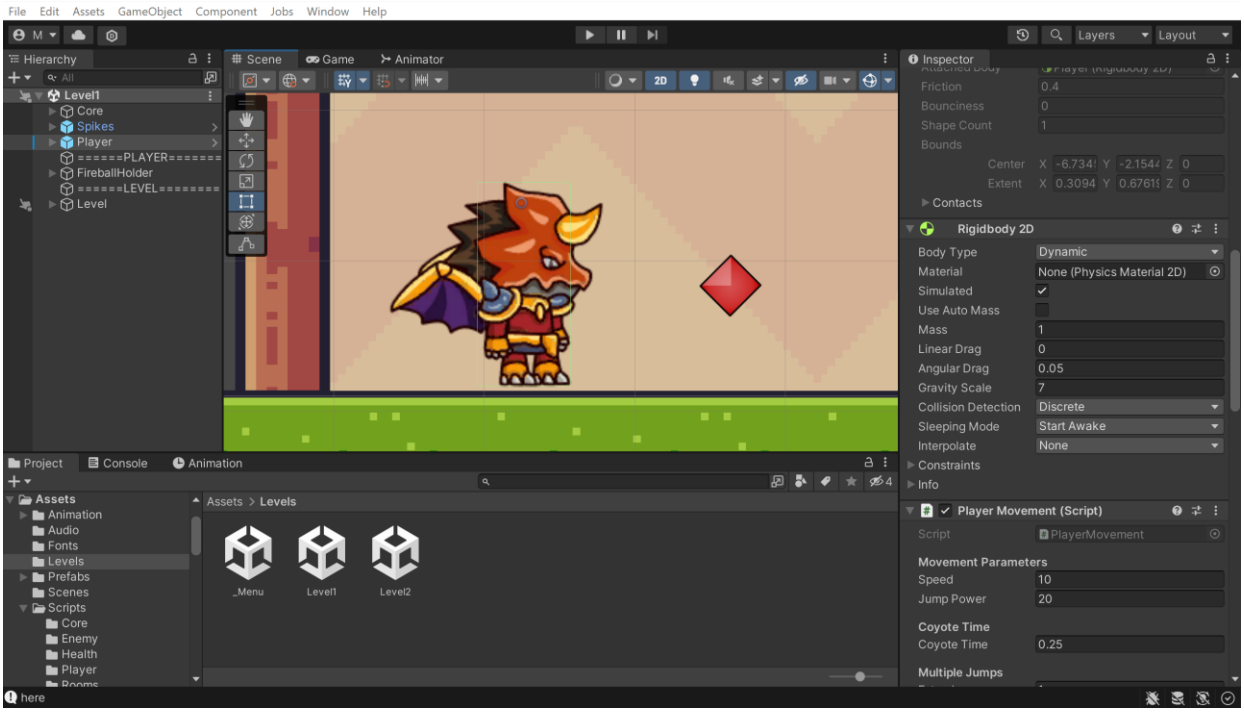

Рис. 3.9. Головний персонаж у вікні Scene і його компонент Rigidbody 2D

3.4.2. Рух об’єктів

Оскільки головний герой повинен рухатися і виконувати різні дії, потрібно створити скрипт мовою C#, який буде називатися PlayerMovement. Це можна зробити кількома способами: створити у вікні Project або створити прямо на потрібному об'єкті у вікні Inspector. Коли скрипт буде створений, треба відкрити його, і тоді Microsoft Visual Studio автоматично запуститься. У коді вже будуть підключені основні бібліотеки, а також створено стандартний метод Update.

Наступним кроком розробки ігрового персонажа є написання скрипту для руху та стрибків. Спочатку задаються змінні (рис. 3.10):

```
private Rigidbody2D body;
private Animator anim;
private BoxCollider2D boxCollider;
private float horizontalInput;
```

Рис. 3.10. Змінні для скрипта PlayerMovement

Змінна body буде мати значення класу Rigibody2D, що використовується для здійснення фізичних дій об'єктом.

Змінна anim буде приймати значення класу Animator, який

використовується для анімації дій бігу та простою (рис. 3.11).

```
//Set animator parameters
anim.SetBool("run", horizontalInput != 0);
anim.SetBool("grounded", isGrounded());
```

Рис. 3.11. Встановлення параметрів класу Animator

Змінна boxCollider прийматиме значення класу BoxCollider2D, який використовується для надання фізичних властивостей об’єкту.

Змінна horizontalInput буде використовуватися для управління рухом персонажа. Для цього необхідно скористатися класом Input, який відповідає за отримання введення, та його методом GetAxis(), що відповідає за горизонтальний рух персонажа. Метод GetAxis("Horizontal") повертає значення -1, якщо персонаж рухається вліво, та 1, якщо рухається вправо. Значення цього методу присвоюється змінній horizontalInput.

Для того, щоб головний герой мав можливість дивитися в обидві сторони, тобто і вправо і вліво, залежно від того, в яку сторону він рухається, задаються наступні умови (рис. 3.12):

```
private void Update()
{
    horizontalInput = Input.GetAxis("Horizontal");

    //Flip player when moving left-right
    if (horizontalInput > 0.01f)
        transform.localScale = Vector3.one;
    else if (horizontalInput < -0.01f)
        transform.localScale = new Vector3(-1, 1, 1);
```

Рис. 3.12. Код для повороту героя в обидві сторони

Якщо змінна horizontalInput приймає значення більше 0.01f (об’єкт рухається вправо), то метод transform.localScale, який відповідає за поворот об’єкту, надає йому значення по осям x, y, z відповідно 1, 1, 1, що змушує головного героя дивитись праворуч. І аналогічно, якщо змінна horizontalInput буде приймати значення менше -0.01f, методом transform.localScale по осям x, y, z об’єкт прийме значення відповідно -1, 1, 1, тобто персонаж

розвернеться по осі x і буде дивитись ліворуч.

Наступним кроком буде надати герою можливість стрибати. Як додаткові дії для більшої кількості вмінь персонажа додано також подвійний стрибок та стрибки по стіні.

Необхідно звернутися до класу Input і його методу GetKeyDown, за допомогою якого виконується перевірка чи натиснена клавіша пробілу (Space). Якщо так, то викликається метод Jump(), який відповідає за виконання стрибка.

Метод Jump() містить послідовність перевірок і дій, пов'язаних з різними умовами стрибка:

– Для реалізації стрибків перевіряється, чи виконується одна з умов, при яких стрибок не виконується, тоді виконання методу завершується, і ніякий стрибок не відбувається;

– перевіряється, чи гравець знаходиться на стіні, тоді виконується метод WallJump(), який реалізує стрибок від стіни.

– перевіряється, чи гравець знаходиться на землі (isGrounded()), тоді гравець здійснює звичайний стрибок шляхом зміни його вертикальної швидкості (body.velocity.y) на значення jumpPower.

– перевіряється, чи значення coyoteCounter більше 0. coyoteCounter відповідає за короткий проміжок часу після зникнення контакту з землею, коли гравець все ще може зробити звичайний стрибок. Якщо це так, то гравець здійснює звичайний стрибок.

– Якщо значення coyoteCounter менше або дорівнює 0, перевіряється умова if (jumpCounter > 0). jumpCounter відповідає за кількість доступних додаткових стрибків. Якщо jumpCounter більше 0, гравець здійснює стрибок (подвійний стрибок) і зменшує значення jumpCounter на 1.

також необхідно створити у героя порожній дочірній об'єкт. Цей порожній об'єкт називатиметься isGrounded, за допомогою нього визначатиметься, перебуває персонаж на землі чи ні. Це необхідно, щоб гравець не міг нескінченно підніматися вертикально вгору

при постійному натисканні клавіші стрибка.

Записуються наступні рядки коду (рис. 3.13):

```
    //Jump
    if (Input.GetKeyDown(KeyCode.Space))
        Jump();
}

2 references
private void Jump()
{
    if (coyoteCounter <= 0 && !onWall() && jumpCounter <= 0) return;
    //If coyote counter is 0 or less and not on the wall and don't have any extra jumps don't do anything

    SoundManager.instance.PlaySound(jumpSound);

    if (onWall())
        WallJump();
    else
    {
        if (isGrounded())
            body.velocity = new Vector2(body.velocity.x, jumpPower);
        else
        {
            //If not on the ground and coyote counter bigger than 0 do a normal jump
            if (coyoteCounter > 0)
                body.velocity = new Vector2(body.velocity.x, jumpPower);
            else
            {
                if (jumpCounter > 0) //If we have extra jumps then jump and decrease the jump counter
                {
                    body.velocity = new Vector2(body.velocity.x, jumpPower);
                    jumpCounter--;
                }
            }
        }

        //Reset coyote counter to 0 to avoid double jumps
        coyoteCounter = 0;
    }
}
```

Рис. 3.13. Код для реалізації стрибків

3.4.3. Анімація об’єктів

Наступним етапом є створення анімації для об’єктів.

Для головного персонажа створюватимуться декілька анімацій, а саме: бігу, стрибка, смерті, зазнавання шкоди, атаки та простою. Використовуватимуться обрані і завантажені з офіційного сайту спрайти. За допомогою функції Sprite Editor поділяється великий спрайт астронавта та беремо

Для цього додаємо до проекту скрипти для всіх цих анімацій. Далі необхідно створити Animator Controller і приєднати його до персонажа. Цей контролер визначає правила та зв'язки для анімацій конкретного об'єкта.

Для створення самостійних анімацій потрібно у вікні Animation створити новий анімаційний кліп, вибравши відповідний об'єкт, а потім

упорядкувати всі кадри анімації у цьому вікні. Тут можна також налаштувати швидкість анімації.

Після створення всіх необхідних анімаційних кліпів треба перейти до вікна Animator (рис. 3.14). Тут створюються зв'язки і налаштовуються умови переходів між різними анімаціями, які були створені раніше.

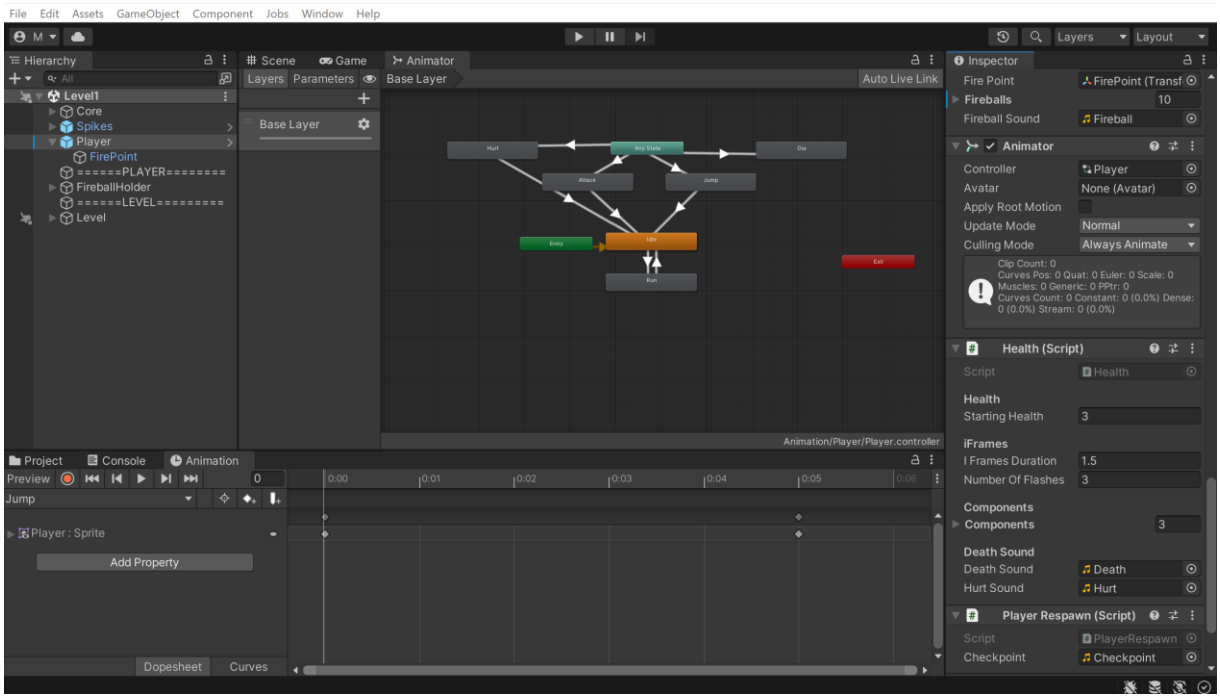

Рис. 3.14. Створення анімації

Після створення всіх анімацій, настає момент, коли необхідно прив'язати їх до конкретних клавіш клавіатури, кнопок миші, консолі тощо. Ця функціональність відома як система введення. Система введення є основою інтерактивності у проектах з реальним часом.

Прив'язка дій до логіки коду та активація різних пристроїв та варіантів керування можлива за допомогою візуального інтерфейсу вікна Input Manager (рис. 3.15).

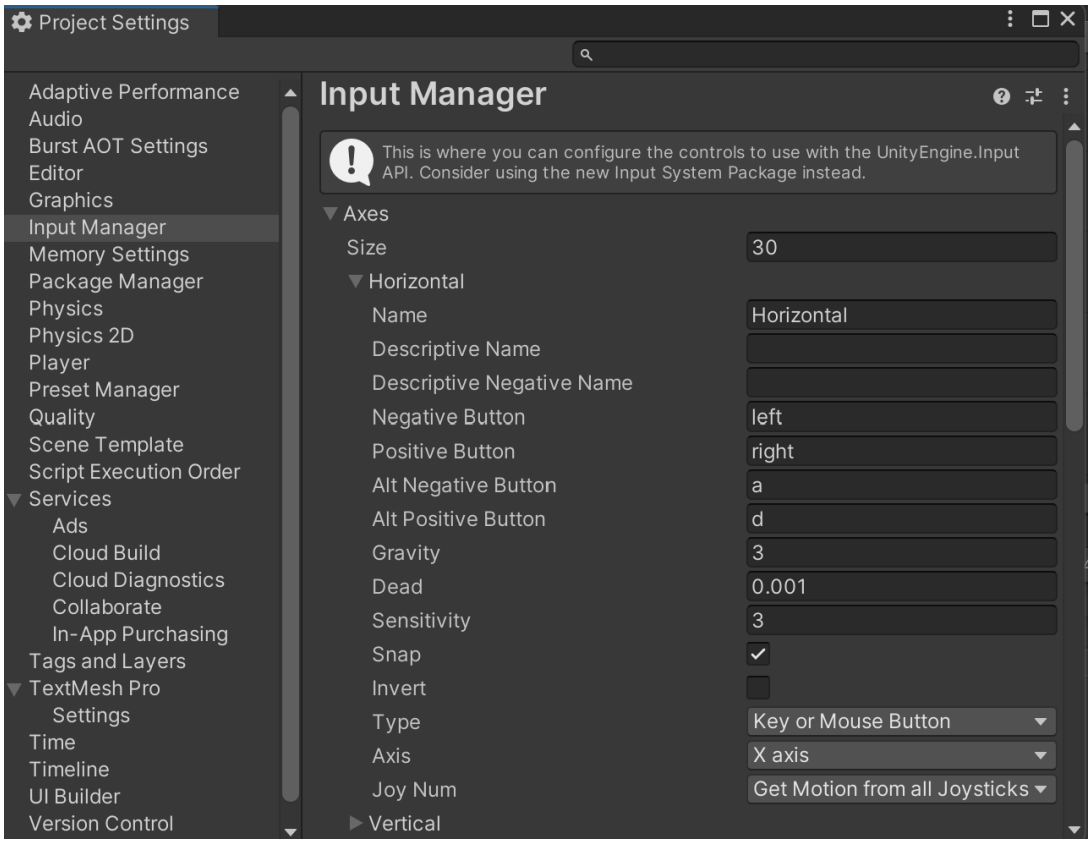

Рис. 3.15. Панель налаштування введення для руху по горизонтальній осі

Кожна вісь пов'язана з двома кнопками на джойстику, миші або клавіатурі. Поле "Name" використовується для ідентифікації осі в скрипті. "Positive button" - це кнопка, яка збільшує значення осі в позитивному напрямку. "Alt Positive Button" - альтернативна кнопка, яка збільшує значення осі в протилежному напрямку. "Gravity" - це швидкість, з якою вісь повертається до нейтрального положення, коли кнопки не натиснуті, вимірюється в одиницях за секунду. "Dead" - це розмір мертвої зони для аналогових пристроїв. Всі значення, які потрапляють у цей діапазон, вважаються нейтральними. "Sensitivity" - це швидкість, з якою вісь рухатиметься до цільового значення, вимірюється в одиницях за секунду. Ця функція працює тільки для цифрових пристроїв. "Type" - тип введення, який керує даною віссю.

3.4.4. Звукові ефекти

Звукові ефекти для гри відграють велике значення. Вони допомагають

покращити геймплей, передають інформацію гравцеві, впливають на настрій та забезпечують задоволення від гри. Вони додають реалізму та взаємодію з віртуальним світом, створюючи звукове оточення, яке залучає гравця. Звукові ефекти також впливають на настрій гравця. Вони можуть створювати напруженість, радість, сум, страх або будь-який інший емоційний стан, що відповідає задуму гри. Правильно підібраний звук може підсилити емоції та враження, які гравець отримує від гри.

В Unity для відтворення звуків використовуються компоненти AudioSource та AudioClip. Компонент AudioSource відповідає за управління відтворенням звукових ефектів, а AudioClip містить самі аудіо дані (рис. 3.16).

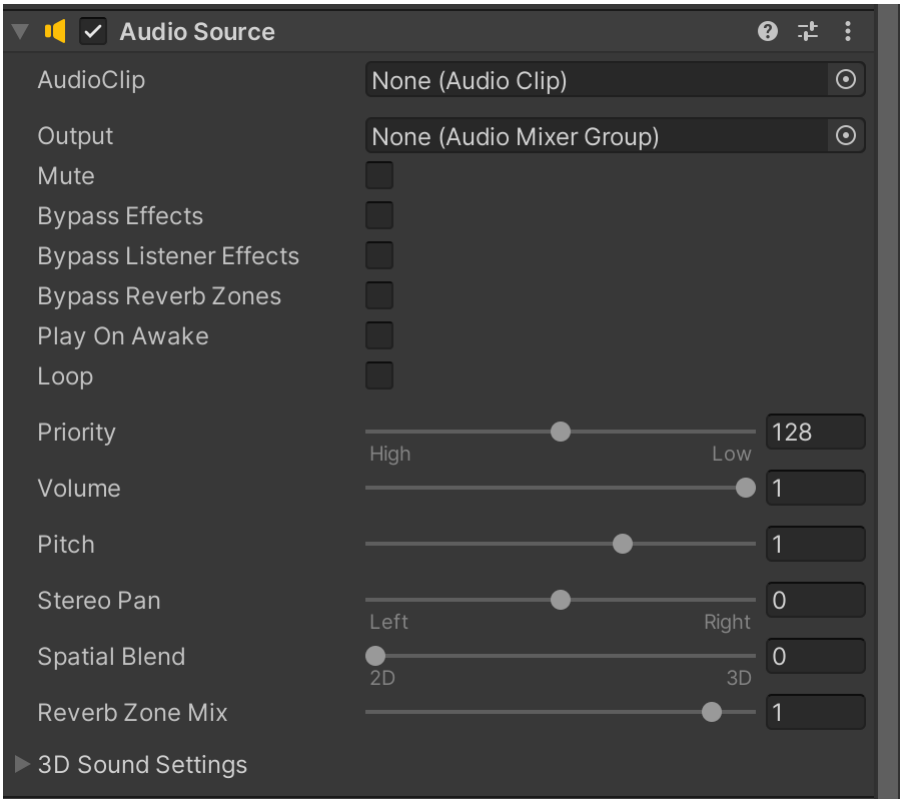

Рис. 3.16. Налаштування звукових ефектів

Компонент AudioSource має різноманітні параметри, які дозволяють налаштовувати звукові ефекти в грі. Наприклад, можна задати AudioClip, який буде відтворюватись, встановити гучність звуку, налаштувати панораму (співвідношення між лівим та правим каналами), контролювати ефекти просторового звучання (наприклад, ефект віддалення або

приближення звуку), встановити певні затримки між відтворенням, та багато іншого.

AudioClip, з свого боку, містить аудіо дані, такі як хвиля звуку та його параметри. Можна імпортувати власні звукові файли в Unity, створити новий AudioClip або використовувати вбудовані звукові ресурси.

Використання компонентів AudioSource та AudioClip дозволяє реалізувати багатогранний звуковий дизайн в грі, поглиблюючи відчуття віртуального світу гри.

Всі дії у грі мають певні звуки (рис. 3.17). Наприклад, у головного героя озвучуються стрибок, зазнавання шкоди, атака, смерть. Також для атмосфери гри було додано фонову музику.

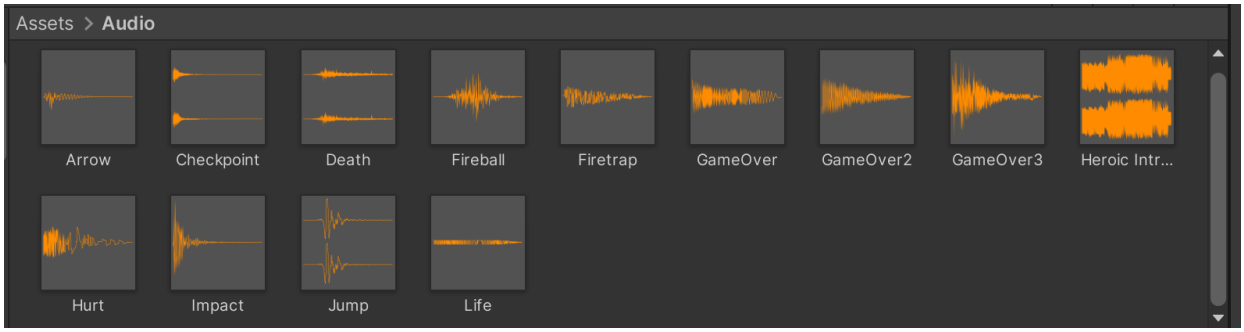

Рис. 3.17. Звукові файли гри

3.4.5. Розробка меню початку, паузи та кінця гри

Стартове меню гри є ключовим елементом, який впливає на перше враження гравця та його подальшу взаємодію з грою. Важливо створити яскравий, зрозумілий та зручний інтерфейс, який зацікавить і залучить гравця.

Для досягнення цього потрібно враховувати кілька факторів. Перш за все, дизайн стартового меню повинен бути привабливим і відповідати стилістиці гри. По-друге, стартове меню повинно бути зрозумілим та легким у використанні. Треба використовувати інтуїтивно зрозумілі символи та піктограми, щоб показати основні функції і можливості гри. Розміщувати кнопки необхідно таким чином, щоб вони були легкодоступні та зручні для натискання. По-третє, гравець повинен мати можливість легко навігувати по

стартовому меню. Необхідно забезпечити наявність кнопок "Грати", "Налаштування", "Вийти" та інших важливих функцій.

Головне меню гри складається з таких елементів як основні кнопки, що мають такі функції (рис. 3.18):

– Play – почати гру;
– Settings – налаштувати гру;
– Quit – вийти з гри.

Рис. 3.18. Початкове меню гри

Під час гри гравець має можливість поставити гру на паузу зі збереженням поточного стану гри.

За допомогою скрипта задаються умови, за яких можлива пауза у грі. Нижче наведено код реалізації даної можливості (рис. 3.19).

```
public void PauseGame(bool status)
{
    //If status == true pause | if status == false unpause
    pauseScreen.SetActive(status);

    //When pause status is true change timescale to 0 (time stops)
    //when it's false change it back to 1 (time goes by normally)
    if (status)
        Time.timeScale = 0;
    else
        Time.timeScale = 1;
}
```

Рис. 3.19. Код для реалізації паузи гри

Функція PauseGame активує або деактивує об'єкт pauseScreen, який представляє собою екран паузи в грі. Якщо status дорівнює true, екран паузи стає видимим, якщо status дорівнює false, екран паузи приховується.

Далі, в залежності від значення status, виконується налаштування часової шкали (timescale). Якщо status дорівнює true, значення timescale встановлюється на 0, що призводить до зупинки часу в грі. Якщо status дорівнює false, значення timescale встановлюється на 1, що дозволяє часу в грі йти нормально.

Це дозволяє контролювати стан паузи в грі шляхом активації або деактивації екрану паузи, а також зупиняти або відновлювати часову шкалу гри відповідно до стану паузи.

Коли головний персонаж вмирає, гра завершується і з’являється меню кінця гри (рис. 3.20). У ньому гравець має можливість розпочати гру заново (Restart), повернутися до початкового меню (Back to menu) та вийти з гри (Quit).

Рис. 3.20. Меню кінця гри

Скрипт містить кілька методів для керування різними функціями меню кінця гри (рис. 3.21).

```
//Activate game over screen
1 reference
public void GameOver()
{
    gameOverScreen.SetActive(true);
    SoundManager.instance.PlaySound(gameOverSound);
}

//Restart level
0 references
public void Restart()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}

//Main Menu
0 references
public void MainMenu()
{
    SceneManager.LoadScene(0);
}

//Quit game/exit play mode if in Editor
0 references
public void Quit()
{
    Application.Quit(); //Quits the game (only works in build)

UNITY_EDITOR
    UnityEditor.EditorApplication.isPlaying = false; //Exits play mode (will only be executed in the editor)
dif
}
```

Рис. 3.21. Код для реалізації кінцевого меню

Метод GameOver() активує екран кінця гри. Він також відтворює звук кінця гри, використовуючи метод PlaySound(). Метод Restart() перезавантажує поточний рівень, викликаючи метод LoadScene(). Метод MainMenu()

переходить до головного меню гри. Метод Quit() виходить з гри або з режиму гри (якщо виконується в редакторі).

3.5. Аналіз отриманого результату

Аналізуючи отриманий результат, можна визначити, що гра складається з трьох рівнів, що вказує на наявність поступового прогресу і зростання складності для гравця. Головний герой в цій грі має здолати різні перешкоди, що включає уникання ворожих шипів (рис. 3.22), ухилення від стріл (рис. 3.23) тощо. Однак, особливу увагу заслуговує факт того, що головний герой повинен здолати лицаря, який має здатність стріляти вогняним шаром для нанесення ураження (рис. 3.24).

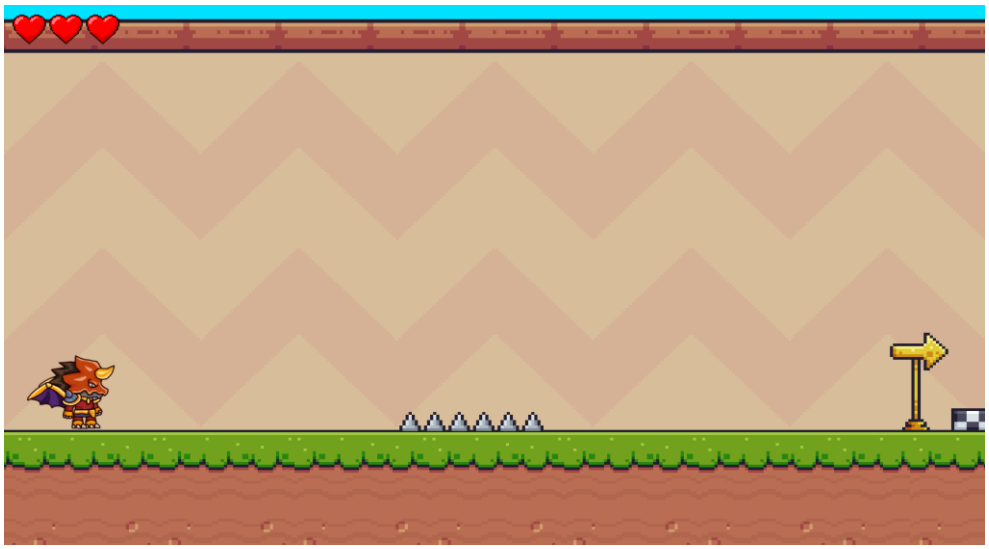

Рис. 3.22. Перший рівень гри

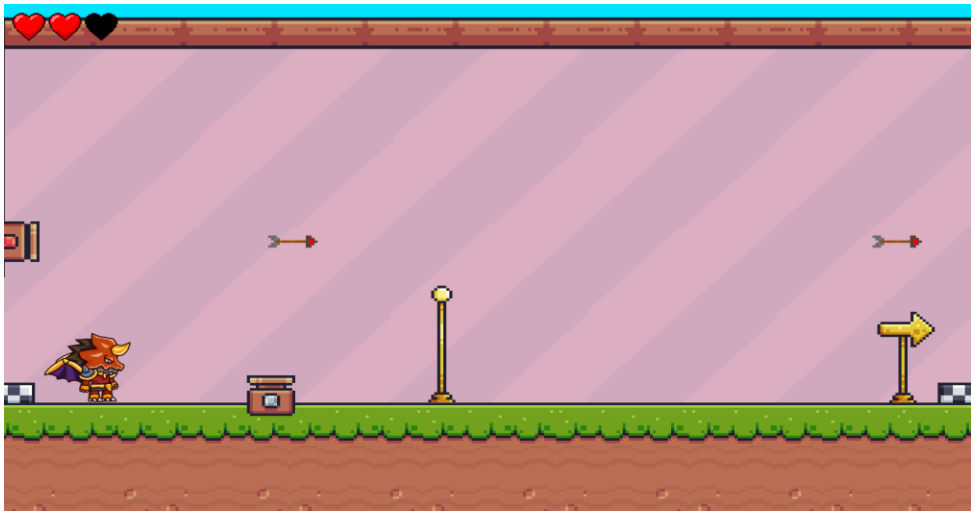

Рис. 3.23. Другий рівень гри

Рис. 3.24. Третій рівень гри

Після завершення налаштування ігрових сцен, анімацій, звуку та інших складових, створюється збірка гри. У налаштуваннях збірки можна вибрати цільову платформу, таку як Windows, MacOS, Linux або інші. Після вибору платформи, можна здійснити збірку гри, яка включатиме всі необхідні ресурси, скрипти та файли, необхідні для її виконання.

Згенерованою збіркою гри можна поділитися з іншими користувачами, використовуючи різні методи поширення, наприклад, завантаження на платформи постачальників контенту або розповсюдження через Інтернет. Це дозволяє гравцям насолоджуватися грою на своїх пристроях і ділитися нею з іншими.

Для подальшого розвитку та покращення гри можна розглянути наступні етапи:

- Додавання нових рівнів, перешкод, ворогів, вмінь головного героя;
- Поліпшення графічної якості гри та звукових ефектів;
- Створення кросплатформених версій гри;
- Розповсюдження на таких платформах, як Steam, App Store, Play Store.
- Збір відгуків користувачів з метою аналізу та покращення гри.

ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі було розглянуто реалізацію заданого темою проекту.

Була визначена концепція та сценарій гри, що надали основу для подальшої розробки. Була встановлена цільова аудиторія гри, яка дозволила спрямувати увагу на потреби та вподобання гравців.

Графічне оформлення гри було ретельно продумано, зокрема створені спрайти головного героя, інтерфейсу та ворогів, що створило візуально привабливе середовище для гравців.

Етап проектування гри охоплював розробку фізичних властивостей об'єктів, руху об'єктів, анімації та звукових ефектів. Ці аспекти були детально вивчені та реалізовані, щоб забезпечити плавний та реалістичний геймплей.

Окремо було розглянуто розробку меню початку, паузи та кінця гри. Були створені відповідні екрани та функціональність, яка дозволяла гравцям з комфортом взаємодіяти з грою та керувати її налаштуваннями.

В результаті аналізу отриманого результату можна зробити висновок, що реалізація проекту пройшла успішно і відповідає запланованим цілям та вимогам. Розроблена гра має потенціал стати захоплюючим та цікавим досвідом для цільової аудиторії. Подальші кроки для розвитку та покращення гри можуть включати розширення механік, додавання нових рівнів та завдань, покращення графічного оформлення та оптимізацію продукту для різних платформ.

ВИСНОВКИ

Постійний прогрес у розвитку комп’ютерних технологій розширює можливості геймдеву та приводить до виникнення нових тенденцій у галузі комп'ютерних ігор.

Крім того, ігрова індустрія стає все більш доступною для різних категорій користувачів. Широка аудиторія, включаючи дітей, підлітків, дорослих та навіть літніх людей, проявляє зацікавленість до ігор і використовує їх як засіб розваги, відпочинку та соціалізації.

Оскільки ігри стають все популярнішими та приносять значний прибуток, ця галузь привертає все більше талановитих розробників і інвесторів. Створення якісних ігор вимагає команди творчих професіоналів, які здатні створити неперевершений геймплей, захоплюючі сюжети та незабутні емоції для гравців.

Враховуючи все вищевикладене, можна зробити висновок, що галузь комп'ютерних ігор у жанрі 2D платформер на базі ігрового двигуна Unity має великий потенціал та перспективи подальшого розвитку. Розробка таких ігор сприяє збагаченню ринку ігор та задоволенню потреб різноманітної аудиторії гравців. Зростання інтересу до ігор, покращення технологій та доступність розвитку ігрових проектів роблять цю галузь привабливою для молодих спеціалістів та підприємницьких зусиль. Ігри залишаються не тільки джерелом розваги, але й потужним засобом візуалізації інформації, соціалізації та навчання, що робить їх значущим фактором у сучасному суспільстві.

У процесі роботи було проведено аналіз предметної області, вивчено поняття комп'ютерних ігор, класифікацію ігор та розглянуто основні етапи їх створення. Аналіз ефективності різних технологій та засобів, що використовуються для реалізації ігрових застосунків, дозволив визначити оптимальний набір інструментів для розробки працездатного та актуального проекту.

Також було проведено дослідження та аналіз додатків-аналогів для з'ясування особливостей та технічних рішень, що використовуються в сучасних іграх. Визначено їх переваги і недоліки, інформація про які була використана

у подальшій розробці ігрового застосунку.

Вибір засобів програмної реалізації базувався на використанні мови програмування C# та ігрового двигуна Unity. Обраний ігровий двигун дозволив зручно та ефективно реалізувати графічне оформлення, фізичні властивості об'єктів, рух та анімацію об'єктів, а також звукові ефекти. Розглянуто середовище розробки Microsoft Visual Studio, яке було обрано для подальшої реалізації проекту.

У процесі проектування заданого темою проекту було створено концепцію та сценарій гри, визначена цільова аудиторія. Було проведено графічне оформлення гри, створено спрайти головного героя, інтерфейсу та ворогів. Також були розглянуті етапи проектування гри, включаючи фізичні властивості об'єктів, рух об'єктів, анімацію та звукові ефекти. Розроблено меню початку, паузи та кінця гри, що є важливими елементами для забезпечення зручного користувацького інтерфейсу.

Аналіз отриманого результату свідчить про успішне виконання проекту та досягнення запланованих цілей. Розроблена гра має потенціал для подальшого розвитку та покращення, зокрема шляхом додавання нових рівнів, механік гри та розширення геймплею. Процес реалізації гри дозволив поглибити знання та навички у галузі розробки комп'ютерних ігор, особливо в контексті 2D платформерів та використання ігрового двигуна Unity.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Креативна економіка та креативні індустрії: навч. посібн. [Електронне видання]. – Житомир: Державний університет «Житомирська політехніка», 2020. – 45 с.
2. [Електронний ресурс] Вікіпедія — комп'ютерна гра. Режим доступу — https://en.wikipedia.org/wiki/PC_game.
3. [Електронний ресурс] UAPLAY – жанри комп'ютерних ігор. Режим доступу — https://uaplay.com.ua/usi-zhanry-pc-ihor/.
4. [Електронний ресурс] UAPLAY – платформер. Режим доступу — https://uaplay.com.ua/tag/platformer/.
5. [Електронний ресурс] Steam – гра Dead Cells. Режим доступу — https://store.steampowered.com/app/588650/Dead_Cells/?.
6. [Електронний ресурс] Steam – гра Hollow Knight. Режим доступу — https://store.steampowered.com/app/367520/Hollow_Knight/? .
7. [Електронний ресурс] Hollowknight.fandom – гра Hollow Knight. Режим доступу — https://hollowknight.fandom.com/wiki/Hollow_Knight_(game).
8. [Електронний ресурс] Klei – гра Mark of the Ninja. Режим доступу — https://www.klei.com/games/mark-ninja.
9. Paris Buttfield-Addison. Unity Game Development Essentials. – O'Reilly Media. – 2019. – 216 s.
10. [Електронний ресурс] GamedevAcademy – ігровий движок Unity. Режим доступу — https://gamedevacademy.org/what-is-unity.
11. Gregory Jason. Game Engine Architecture. – New York, CRC Press. – 2019. – No. 3. – s. 12 – 22.
12. [Електронний ресурс] GolosKarpat.info – ігровий движок Unreal Engine. Режим доступу — https://goloskarpat.info/associated/63c5623f4a18e/.
13. [Електронний ресурс] Report.if – ігровий движок Unreal Engine. Режим доступу — https://report.if.ua/rozvagy/unreal-engine-korystuvackyj-oglyad-platformy-dlya-stvorennya-mobilnoyi-gry/.

14. [Електронний ресурс] Вікіпедія — ігровий движок CryEngine. Режим доступу — https://uk.wikipedia.org/wiki/CryEngine_(%D1%81%D0%B5%D1%80%D1%96%D1%8F_%D1%80%D1%83%D1%88%D1%96%D1%97%D0%B2).

15. [Електронний ресурс] Igrodom — ігровий движок CryEngine. Режим доступу — https://igrodrom.net/ua/files/cryengine-84/.

16. [Електронний ресурс] Informatics — середовище розробки Visual Studio. Режим доступу — https://informatics.in.ua/programming_csharp/part_01.php.

17. [Електронний ресурс] Docs.unity3d – Box Collider 2D. Режим доступу — https://docs.unity3d.com/Manual/class-BoxCollider2D.html.