МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проєкт

здобувача освіти денної форми навчання

РП.07.23.000.ДП

ЧУЛАКОВА ВЛАДИСЛАВА ОЛЕГОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Інженерія програмного забезпечення»**

Група: **4РП-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка ігрового застосунку у жанрі раннер-платформер

Проєктний матеріал складається з пояснювальної записки на **82** сторінках та графічного (презентаційного) матеріалу на **14** аркушах (слайдах).

Дипломник _____ (Чулаков В. О.)

Керівник _____ (Іванова Л. В.)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «25» 06 2024 р. Протокол ЕК № 2

Оцінка ЕК 5 (п'ять) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І. В.*

" 15 " 01 *2024 року*

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Чулакову Владиславу Олеговичу*

1. Тема проєкту *Розробка ігрового застосунку у жанрі раннер-платформер*

Затверджена наказом по коледжу від " *02* " *листопада* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту 10.06.2024

3. Вихідні дані до проєкту

1. Спроектувати сюжет ігрового застосунку.

2. Реалізувати графічний інтерфейс (GUI) ігрового застосунку

3. Спроектувати Level-дизайн ігрового застосунку

4. Спроектувати дизайн персонажів ігрового застосунку

5. Розробити ігровий застосунок засобами рушію Unity

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналіз предметної області. 2. Технології та засоби розробки (проєктування).

3. Проєктування дизайну гри. 4. Проєктування архітектури ігрового застосунку.

5. Розробка ігрового застосунку. 6. Тестування створеного ігрового застосунку.

7. Економічний розрахунок. 8. Аспекти охорона праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація Power Point – 14 слайдів

(Загальні відомості; Стек технологій; Навігація меню гри; Налаштування гри;

Архітектура гри; Створення головного персонажу; Створені префаби та матеріали;

Створення дизайну рівня; Програмний код генерації рівня; Графічний інтерфейс;

Геймплей з різним кутом камери; Геймплей з різною порою доби)

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Іванова Л. В.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання 15.01.2024р

Керівник *Іванова Л. В.* (підпис)

Завдання прийняв до виконання *Чулаков В. О.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Формування вступу*	*29.04.24*	виконано
2	*Аналіз предметної області*	*10.05.24*	виконано
3	*Підбір технічної літератури*	*19.05.24*	виконано
4	*Вибір технологій та засобів розробки (проєктування)*	*20.05.24*	виконано
5	*Проєктування дизайну ігрового застосунку*	*22.05.24*	виконано
6	*Проєктування архітектури ігрового застосунку*	*24.05.24*	виконано
7	*Розробка ігрового застосунку*	*27.05.24*	виконано
8	*Тестування створеного ігрового застосунку*	*29.05.24*	виконано
9	*Оформлення пояснювальної записки*	*31.05.24*	виконано
10	*Оформлення графічної (презентаційної) частини*	*01.06.24*	виконано
11	*Економічний розрахунок*	*02.06.24*	виконано
12	*Опис охорони праці та техніки безпеки*	*09.06.24*	виконано
13	*Аналіз результатів проєктування*	*13.06.24*	виконано
14	*Підготовка доповіді для захисту*	*16.06.24*	виконано

Дипломник (підпис)

Керівник (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 23 000. 00 ДП	Дипломний проєкт		
А4			РП 07. 23 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 23 000. 00 ДП ПЗ		
Розроб.		Чулаков В.О.		10.06	Розробка ігрового застосунку у жанрі раннер-платформер	Літ.	Аркуш 4 / Аркушів 82
Перевірив		Іванова Л.В.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06	ВСП "ОТФК ОНТУ" гр.4РП-07		
Затверд.		Кривченко Ю.В.		12.06			

ЗМІСТ

ВСТУП

Жанр раннер-платформер знаходиться у центрі уваги геймерів завдяки своїй динамічності та викликам, що пропонує. Ці ігри, які поєднують у собі швидкий біг, стрибки та подолання перешкод, створюють захопливу ігрову атмосферу, що приваблює гравців будь-якого віку. Популярність раннер-платформерів є джерелом натхнення для розробників, які постійно впроваджують інноваційні рішення для забезпечення плавного та захоплюючого ігрового процесу. Сучасні ігри цього жанру пропонують різноманітні рівні складності, персонажів і механіки, що тримають гравців у напрузі та заохочують до досягнення нових вершин.

Мета розробки нового раннер-платформера полягає у створенні гри, яка використовуватиме передові ідеї для оптимізації ключових аспектів геймплею. Вона буде включати різні рівні складності, системи досягнень, інтерактивні елементи та різноманітні режими камери, такі як 2D, ізометричний 3D, 3D від третьої особи та 3D від першої особи.

Розробка відбувається на платформі Unity 3D з використанням мови програмування C#, що дозволяє створювати захоплюючі тривимірні світи з високим рівнем деталізації та плавною анімацією. Використання мови C# забезпечує надійну та ефективну реалізацію ігрової логіки, що дозволяє створювати складні механіки та інтерактивні елементи.

Грає важливу роль у вдосконаленні навичок гравців та надає їм можливість насолоджуватися динамічним ігровим процесом. Практичне значення розробленої гри полягає в тому, що вона стане чудовим доповненням до колекції шанувальників жанру "раннер-платформер" та приверне нових гравців, які цінують високу якість та захопливий геймплей. Таким чином, новий раннер-платформер створить захоплюючий ігровий світ, який залишить незабутні враження у серцях гравців.

Ігрова індустрія постійно розвивається, й це надихає розробників на створення ще більш захоплюючих інтерактивних експерієнцій. Завдяки постійному попиту на нові ідеї та ігрові механіки, раннер-платформери залишаються актуальним і популярним жанром серед геймерів.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз предметної області

1.1.1 Порівняння наявних ігрових застосунків

Розглянемо кілька популярних мобільних ігор, які можна знайти на просторах інтернету.

Перший аналог – "Flappy Bird".

"Flappy Bird" – це проста і водночас дуже популярна аркадна гра, де гравець керує маленьким птахом, який пролітає між рядами труб, що розташовані зверху і знизу екрану. Мета гри – пролетіти якомога далі, не врізаючись у труби. Гра має мінімалістичний дизайн та легко зрозумілий геймплей, що робить її доступною для широкого кола гравців. Для управління птахом необхідно просто натискати на екран, щоб утримувати його в повітрі [1].

На рис. 1.1 зображено ігровий процес "Flappy Bird".

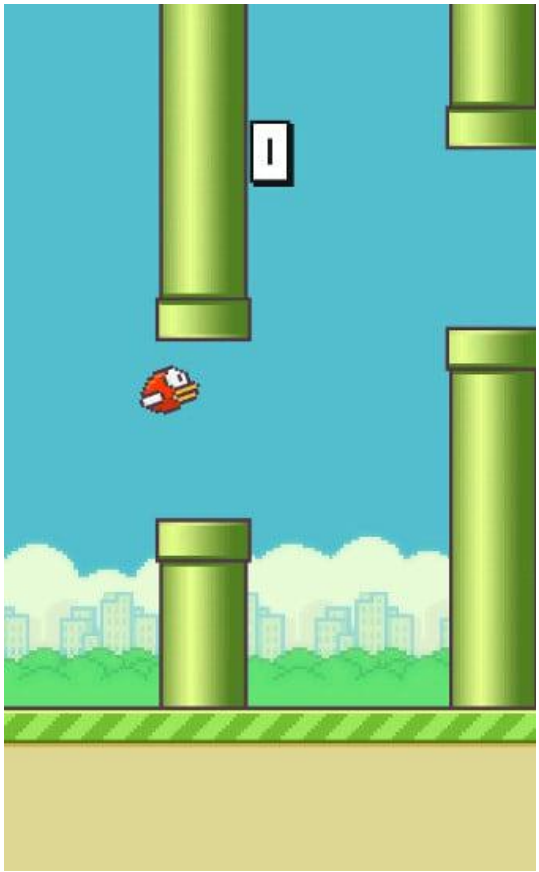


Рисунок 1.1. Ігровий процес "Flappy Bird"

Переваги "Flappy Bird":

Простота геймплею: Легкий у засвоєнні, але складний для майстерного проходження.

Мінімалістичний дизайн: Візуальний стиль гри приваблює своєю простотою.

Висока реіграбельність: Постійне бажання побити свій попередній рекорд робить гру дуже захоплюючою.

Відсутність потреби в Інтернеті: Гру можна грати офлайн.

Недоліки "Flappy Bird":

Висока складність: Гра може бути дуже складною для новачків, що може відлякати деяких гравців.

Однотипність геймплею: Відсутність різноманітності може швидко набриднути.

Відсутність оновлень: Оскільки гра більше не підтримується розробником, нових функцій чи виправлень чекати не варто.

Другий аналог – "Subway Surfers".

"Subway Surfers" – це нескінченна бігова гра, де гравець керує персонажем, який тікає від інспектора по залізничних коліях, ухиляючись від перешкод і збираючи монети та бонуси. Гра має барвистий 3D-графіку та динамічний геймплей, що приваблює гравців різного віку. Управління відбувається за допомогою свайпів по екрану для стрибків, ковзань та зміни напрямку руху [2].

На рис. 1.2 зображено ігровий процес "Subway Surfers".

Рисунок 1.2. Ігровий процес "Subway Surfers"

Переваги "Subway Surfers":

Барвиста графіка: Яскравий і привабливий візуальний стиль.

Динамічний геймплей: Постійна зміна швидкості та перешкод утримує гравців в напрузі.

Регулярні оновлення: Часті оновлення з новими персонажами, локаціями та завданнями.

Соціальні функції: Можливість змагатися з друзями через таблиці лідерів.

Недоліки:

Необхідність Інтернету: Для повноцінного використання всіх функцій гри потрібне підключення до Інтернету.

Наявність реклами: Часті рекламні оголошення можуть відволікати від гри.

Внутрішньоігрові покупки: Можливість придбання бонусів за реальні гроші може бути недосяжною для деяких гравців.

Третій аналог – "Temple Run".

"Temple Run" – це ще одна популярна нескінченна бігова гра, де гравець керує персонажем, який тікає з храму, уникаючи перешкод і збираючи монети та бонуси. Гра має атмосферний 3D-графіку та просте управління за допомогою свайпів і нахилів пристрою. Гравець повинен реагувати на зміну напрямку доріжки, стрибати через провалля та ковзати під перешкодами [3].

На рис. 1.3 зображено ігровий процес "Temple Run".

Рисунок 1.3. Стартовий екран "Temple Run"

Переваги "Temple Run":

Захоплюючий сюжет: Атмосфера пригодницької гри з елементами трилера.

Інтуїтивне управління: Просте та зручне управління за допомогою свайпів і нахилів.

Висока реіграбельність: Безліч бонусів та досягнень стимулюють грати знову і знову.

Мультиплатформеність: Гра доступна на різних платформах, включаючи мобільні пристрої та планшети.

Недоліки "Temple Run":

Однотипність геймплею: Відсутність різноманітності у перешкодах і локаціях може знижувати інтерес з часом.

Необхідність Інтернету: Деякі функції гри потребують підключення до Інтернету.

Внутрішньоігрові покупки: Деякі елементи гри можна придбати лише за реальні гроші.

Проведемо порівняльний аналіз аналогів.

У табл. 1.1 зображено порівняння аналогів.

Таблиця 1.1. Порівняння аналогів

Характеристика	**Flappy Bird**	**Subway Surfers**	**Temple Run**
Графіка	Мінімалістична	Барвиста 3D	Атмосферна 3D
Управління	Натискання на екран	Свайпи по екрану	Свайпи та нахили пристрою
Необхідність Інтернету	Немає	Є для деяких функцій	Є для деяких функцій
Реіграбельність	Висока	Висока	Висока
Внутрішньоігрові покупки	Відсутні	Є	Є
Регулярні оновлення	Відсутні	Є	Є

Аналізуючи таблицю порівняння трьох популярних ігор - "Flappy Bird", "Subway Surfers" та "Temple Run", можна зробити кілька висновків. Кожна гра має свої унікальні особливості, що приваблюють гравців різного типу.

"Flappy Bird" виділяється своєю мінімалістичною графікою та простим управлінням, що робить її привабливою для людей, які шукають легкості в геймплеї.

"Subway Surfers" та "Temple Run", з іншого боку, пропонують більш розвинений графічний стиль та складніші завдання, що вимагають від гравців більшої уваги та реакції. У той же час, "Subway Surfers" вирізняється більшою кількістю можливостей для взаємодії з іншими гравцями через таблиці лідерів та соціальні функції. У "Temple Run" більше акценту робиться на атмосфері та сюжеті гри, що робить її більш захоплюючою для гравців, які цінують насиченість геймплею. Обидві ці гри також мають систему внутрішньоігрових покупок, що дозволяє гравцям отримати додаткові бонуси.

Таким чином, кожна з цих ігор має свої переваги та недоліки, і вибір найкращої залежить від індивідуальних уподобань та очікувань гравця.

1.1.2 Обрані технології ігробудування

Розглянемо топ ігрових рушіїв перед розробкою.

За даними веб-сайту GameFromScratch, бачимо, що Unity займає найбільшу долю використання на глобальних гейм джемах [4].

На рис. 1.4 зображено долю використання ігрових рушіїв на глобальних геймд жемах.

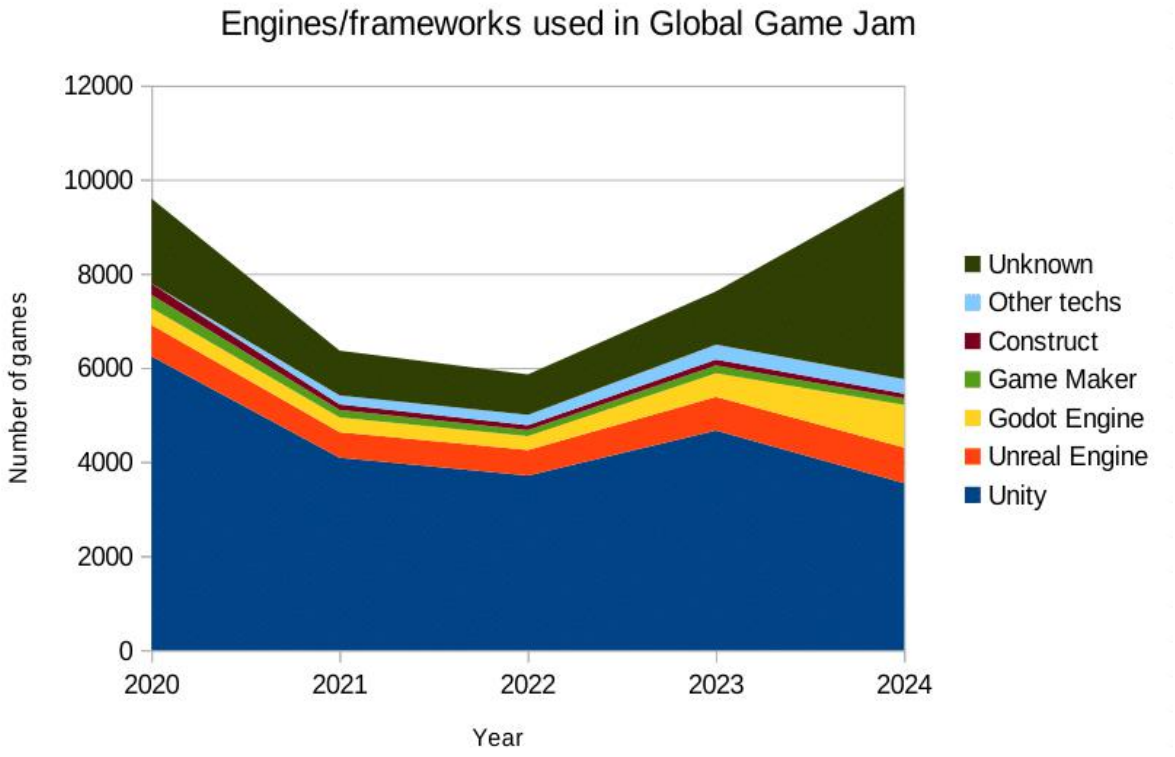


Рисунок 1.4. Доля використання ігрових рушіїв на глобальних геймд жемах

За даними веб-сайту hackr, бачимо, що мова програмування C# займає друге місце по комплексній кількості факторів [5].

На рис. 1.5 топ найкращих мов програмування для геймдеву.

Рисунок 1.5. Топ найкращих мов програмування для геймдеву

Таким образом, для ігрового проєкту даної дипломної роботи було обрано рушій Unity 3D та мову програмування C#.

Unity 3D являє собою інтегроване середовище розробки (IDE) та рушій для створення ігор і застосунків у 2D і 3D. Він надає розробникам інструменти для створення графічних ефектів, фізики, анімації, штучного інтелекту та мультиплеєра. Unity 3D підтримує різні платформи, включаючи мобільні пристрої, ПК, консолі та веб [6].

Переваги Unity 3D:

Швидкий процес розробки: Забезпечує швидкий та зручний спосіб створення ігор та застосунків.

Легка вивчаємість: Доступний для новачків, має багато матеріалів для самостійного навчання.

Велика спільнота та ресурси для підтримки: Існує велика спільнота користувачів, багато ресурсів, форумів та документації.

Підтримка багатьох платформ: Дозволяє розробляти для різних платформ, включаючи мобільні пристрої, ПК, консолі та веб.

Можливості мультиплеєра: Надає вбудовану підтримку для створення мережевих ігор.

Недоліки Unity 3D:

Обмежена графічна якість порівняно з Unreal Engine: У порівнянні з конкурентами, як Unreal Engine, графічна якість може бути менш вражаючою.

Менша ефективність великих проєктів: Деякі користувачі відзначають, що Unity може бути менш ефективним для великих проєктів з великою кількістю об'єктів.

Не дуже підходить для створення фотореалістичних ігор: В порівнянні з Unreal Engine, Unity може мати обмежені можливості для створення фотореалістичних ігор.

Проблеми з оптимізацією для деяких пристроїв: Для деяких платформ або пристроїв можуть виникати проблеми з оптимізацією та продуктивністю.

Проведемо порівняльний аналіз технологій.

У табл. 1.2 зображено порівняння технологій.

Таблиця 1.2. Порівняння технологій

Критерій	**Unity 3D**	**Unreal Engine**
Графічна якість	Стандартна, менше деталей	Висока, фотореалістична
Швидкість розробки	Висока	Середня
Вивчаемість	Вище	Нижче
Ресурси та підтримка	Багато документації, спільнота	Сильна технічна підтримка
Мультиплатформенність	Добре підтримується	Також добре підтримується

C# являє собою об'єктно-орієнтовану мову програмування, яка розроблена компанією Microsoft. Вона була створена для розробки різноманітних рішень на платформі .NET Framework. C# є потужною мовою, яка поєднує в собі елементи C і C++, а також додає власні удосконалення, такі як обробка винятків, автоматичне керування пам'яттю та безпека типів. Ця мова використовується широкою аудиторією розробників для створення різноманітних програм, включаючи ігри, веб-застосунки, мобільні застосунки та багато іншого [7].

Переваги використання C# в Unity 3D:

Простота вивчення: C# вважається дуже дружньою для початківців мовою програмування. Її синтаксис легко зрозуміти, що дозволяє новачкам швидко вловлювати основи програмування в Unity.

Інтеграція з Unity: C# є офіційною мовою програмування для Unity, тому всі можливості і функції Unity API легко доступні для використання в C#. Unity пропонує потужні інструменти і ресурси для розробників, які використовують C#.

Широкі можливості: C# - це мова загального призначення, яка підтримує широкий спектр функціональності, включаючи об'єктно-орієнтоване програмування, делегати, події, лямбда-вирази та багато іншого.

Ефективність: C# є високорівневою мовою, що дозволяє розробникам швидко писати та відлажувати код. Вона також дозволяє працювати з потужними бібліотеками і фреймворками для швидкого розроблення програм.

Недоліки використання C# в Unity 3D:

Швидкість виконання: Хоча C# є достатньо швидкою мовою, у порівнянні з іншими мовами програмування, такими як C++ або навіть JavaScript, вона може мати деякі обмеження щодо швидкості виконання.

Менша підтримка веб-ігор: У порівнянні з JavaScript, який може бути використаний для розробки веб-ігор, C# має меншу підтримку для цієї платформи.

Не всі функції можуть бути доступні: Деякі розширені можливості Unity можуть бути доступні лише через скрипти, написані на C++. Таким чином, деякі функції можуть бути обмежені для C# [8, 9].

Проведемо порівняльний аналіз підтримуваних мов програмування.

У табл. 1.3 зображено порівняльний аналіз підтримуваних мов програмування.

Таблиця 1.3. Порівняння мов програмування

Критерій	**C#**	**JavaScript**
Продуктивність	Вище	Нижче
Швидкість виконання	Швидше	Повільніше
Вивчаємість	Середня	Висока
Екосистема	Широка	Менш розвинена
Підтримка Unity	Найкраща, офіційна мова	Підтримка не офіційна

Інформація, подана в порівняннях, робить кілька важливих висновків:

Unity 3D відзначається швидким процесом розробки, легкістю вивчення та широкою підтримкою спільноти. Проте, порівняно з Unreal Engine, у нього обмежена графічна якість, а також можливі проблеми з оптимізацією для деяких пристроїв.

У порівнянні Unity 3D з Unreal Engine, Unity 3D може мати менш фотореалістичну графіку, але компенсує це швидкістю розробки та більшою вивчаємістю.

Вибір між мовами програмування C# та JavaScript залежить від декількох факторів. C# зазвичай вище за продуктивністю та швидкістю виконання, а також має офіційну підтримку Unity. З іншого боку, JavaScript може бути більш вивченою та мати більшу екосистему для деяких розробників.

Таким чином, маємо наступний стек використовуємих технологій.

На рис. 1.6 зображено стек використовуємих технологій.

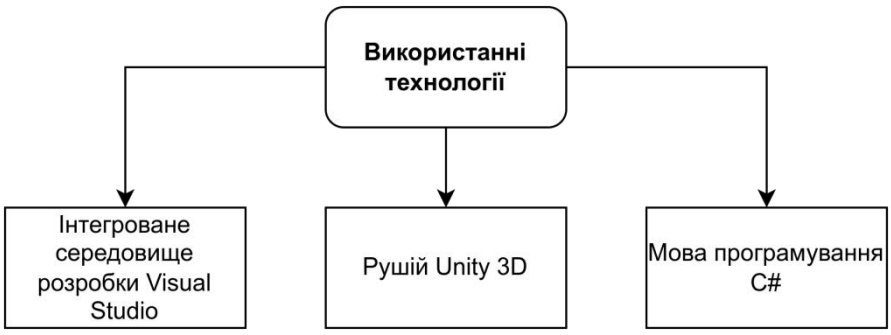


Рисунок 1.6. Стек використовуємих технологій

1.2 Проєктування ігрового застосунку

1.2.1 Технічне завдання на розробку

Технічне завдання на розробку ігрового застосунку містить наступні аспекти:

Наявність графічного інтерфейсу користувача (GUI): Графічний інтерфейс користувача (GUI) в грі має включати головне меню з кнопками для початку гри, завантаження збережень, налаштувань та виходу, а також меню паузи з можливістю збереження, зміни налаштувань і повернення до гри. Важливим елементом є відображення підказок та інструкцій, які допомагають гравцям орієнтуватися у грі.

Мінімалістична графіка: Мінімалістична графіка передбачає використання простих форм і однотонних текстур, чітких ліній та обмеженої палітри кольорів. Це допомагає створити чистий і сучасний вигляд гри, уникаючи перевантаження візуальними деталями.

Налаштування оточення: Налаштування оточення дозволяє гравцеві змінювати час доби в грі між днем і ніччю. Це може впливати на геймплей, наприклад, видимість, наявність певних персонажів чи подій. Перемикання між режимами повинно бути плавним, з відповідними змінами освітлення та тіней.

Налаштування складності: Налаштування складності надають гравцеві можливість обирати рівень викликів у грі, такі як "легкий", "середній" і "важкий". Це впливає на поведінку ворогів, доступні ресурси, час виконання завдань та інші аспекти геймплею. Опції складності мають бути чітко описані, щоб гравець знав, що очікувати від кожного рівня.

Налаштування кута камери: Налаштування кута камери дозволяють гравцеві перемикатися між перспективами першої та третьої особи. Це забезпечує різний ігровий досвід: від більш зануреного погляду від першої особи до кращого огляду навколишнього середовища від третьої особи. Перемикання повинно бути зручним та інтуїтивно зрозумілим, з належним врахуванням змін в управлінні ігровим персонажем.

Відображення статистики: Відображення статистики надає гравцеві важливу інформацію про його прогрес та досягнення в грі. Це може включати дані про час гри, зібрані ресурси, рівень здоров'я, кількість переможених ворогів тощо. Статистика повинна бути зрозумілою і легко доступною, з можливістю перегляду у будь-який момент гри через відповідне меню або інтерфейс.

На рис. 1.7 зображено навігацію по графічному інтерфейсу ігрового застосунку.

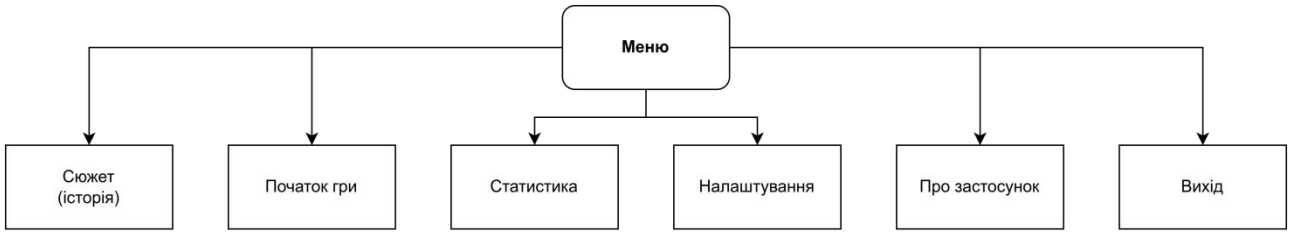


Рисунок 1.7. Навігація по ігровому застосунку

На рис. 1.8 зображено налаштування ігрового застосунку.

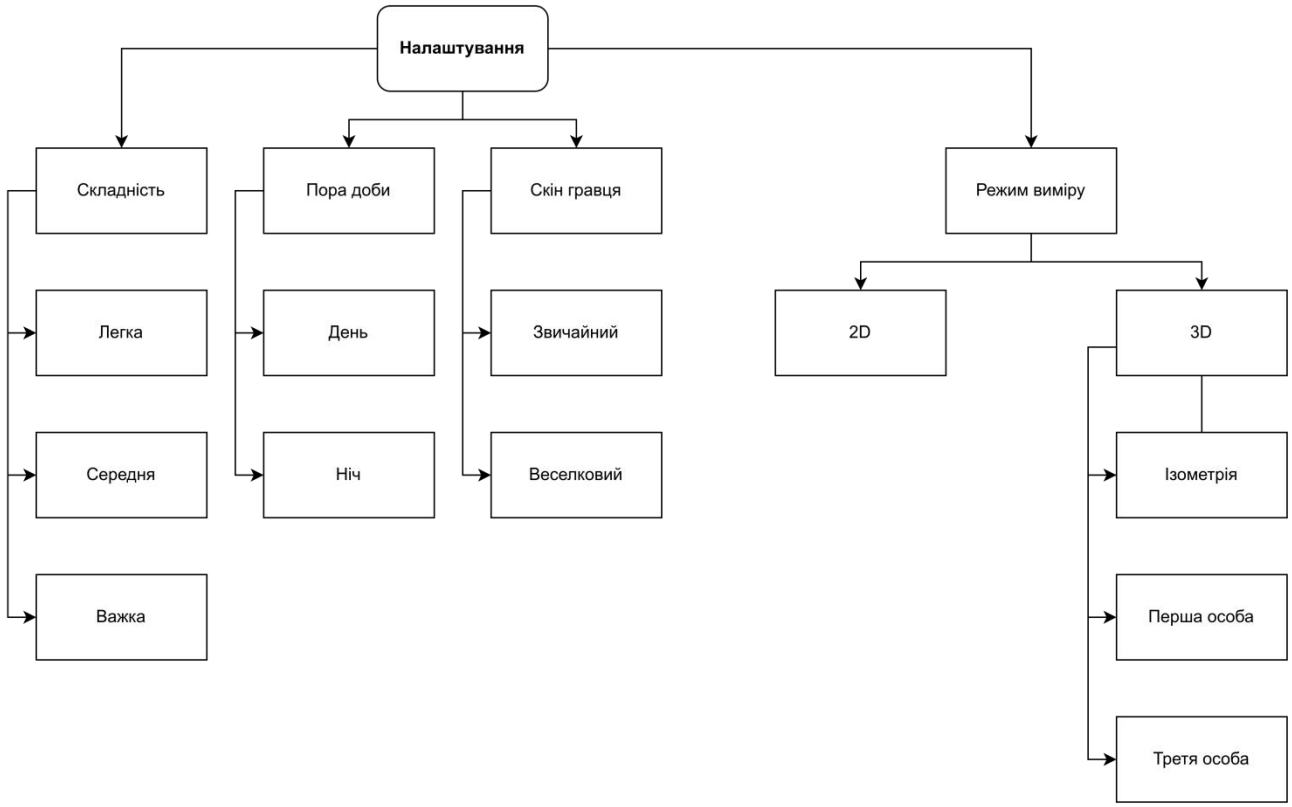


Рисунок 1.8. Налаштування ігрового застосунку

1.2.2 Проєктування сюжету гри

Сюжет гри полягає в тому, що головному персонажу потрібно протриматися якомога довше в ігровій сесії.

Під час виживання, він буде пересуватися по ігровому рівню, задача гравця уникати стін. У разі зіткнення зі стіною ігрова сесія закінчиться.

Під час ігрової сесії гравець може підібрати монети, що дадуть шанс на переродження, а також ворогів, котрих потрібно збивати зі свого шляху.

Складність даного ігрового процесу залежить від обраних налаштувань.

1.2.3 Проєктування дизайну персонажів

Головний персонаж та вороги в ігровому застосунку представляють собою шар з крилами, котрі нагадують снітч з фільму “Гарі потер”.

На рис. 1.9 зображено зовнішній вигляд снітчу.

Рисунок 1.9. Зовнішній вигляд снітчу

В самій грі головний персонаж та вороги представлені у вигляді префабу.

Префаб в Unity 3D являє собою збережений шаблон об'єкта, який може бути багаторазово використаний у сценах для створення копій цього об'єкта зі всіма його налаштуваннями та компонентами.

Префаби можуть включати в себе примітиви.

Примітиви в Unity 3D являють собою базові 3D-об'єкти, такі як куб, сфера та циліндр, які слугують будівельними блоками для створення складніших моделей та середовищ.

В даному випадку префаби головних персонажів та ворогів включать в себе наступні прмітиви Unity 3D:

1. Сфера (тіло).
2. Дві поверхні (крила).

На рис. 1.10 зображено склад примітивів префабу головного персонажу.

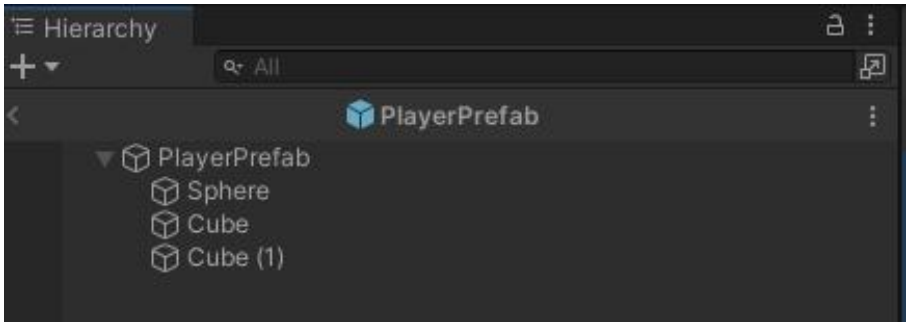


Рисунок 1.10. Склад примітивів префабу головного персонажу

На рис. 1.11 зображено зовнішній вигляд префабу головного персонажу гри.

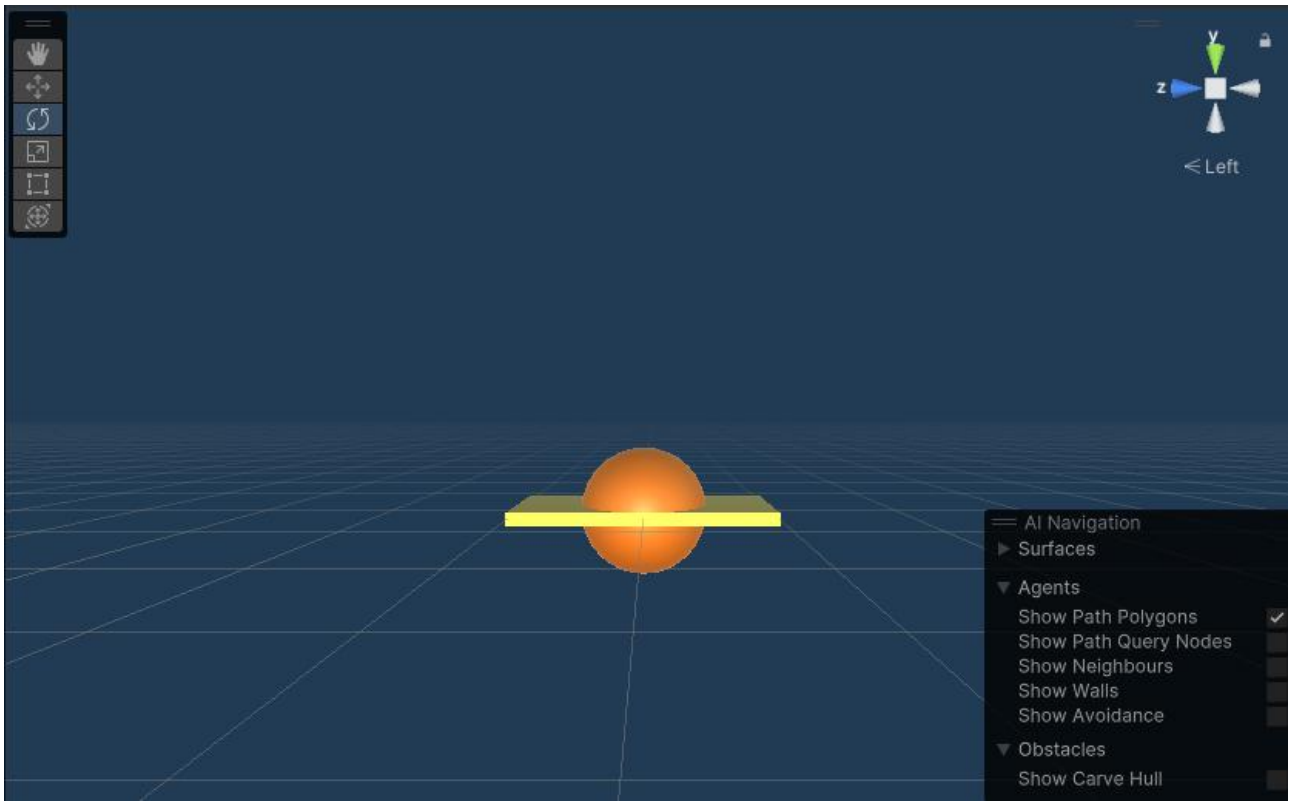


Рисунок 1.11. Префаб головного персонажу гри

На рис. 1.12 зображено зовнішній вигляд префабу ворогу.

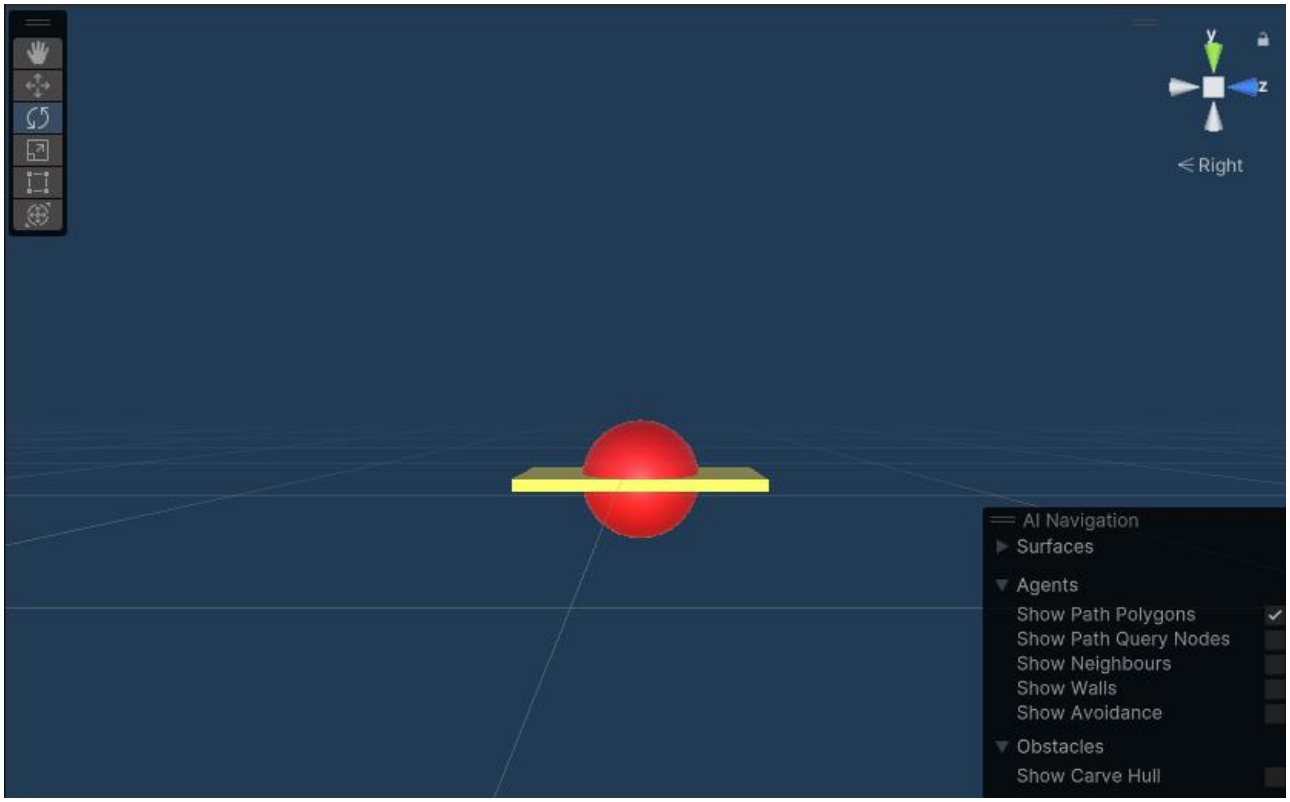


Рисунок 1.12. Префаб ворогу

1.2.4 Проєктування дизайну рівню

Рівень в ігрового застосунку представляє собою дорогу на котрій розташовані перешкоди у вигляді блоків стін. Створимо додаткові префаби для оточення.

На рис. 1.13 зображено фінальний перелік префабів.

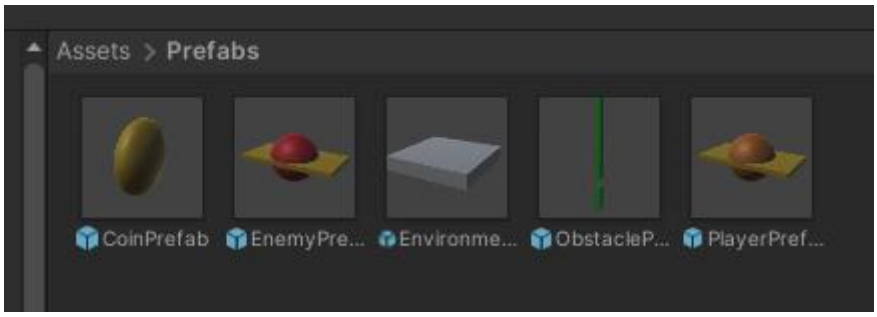


Рисунок 1.13. Фінальний перелік префабів

Для того, щоб префаби мали колір, на них потрібно застосувати матеріали.

Матеріали в Unity 3D являють собою візуальні характеристики, які визначають зовнішній вигляд 3D-об'єктів, включаючи колір, текстуру та

освітлення. Вони використовуються для надання реалістичності та деталізації вашим моделям.

На рис. 1.14 зображено фінальний перелік матеріалів.

Рисунок 1.14. Фінальний перелік матеріалів

Для розташування префабів в Unity 3D використовується сцена.

Сцена в Unity 3D являє собою віртуальний простір, де розмішуються та налаштовуєте всі елементи гри, такі як 3D-моделі, освітлення, камери та ігровий процес. Вона слугує основою для створення вашого ігрового світу.

На рис. 1.15 зображено фінальну сцену з розташованими префабами (дизайн гри).

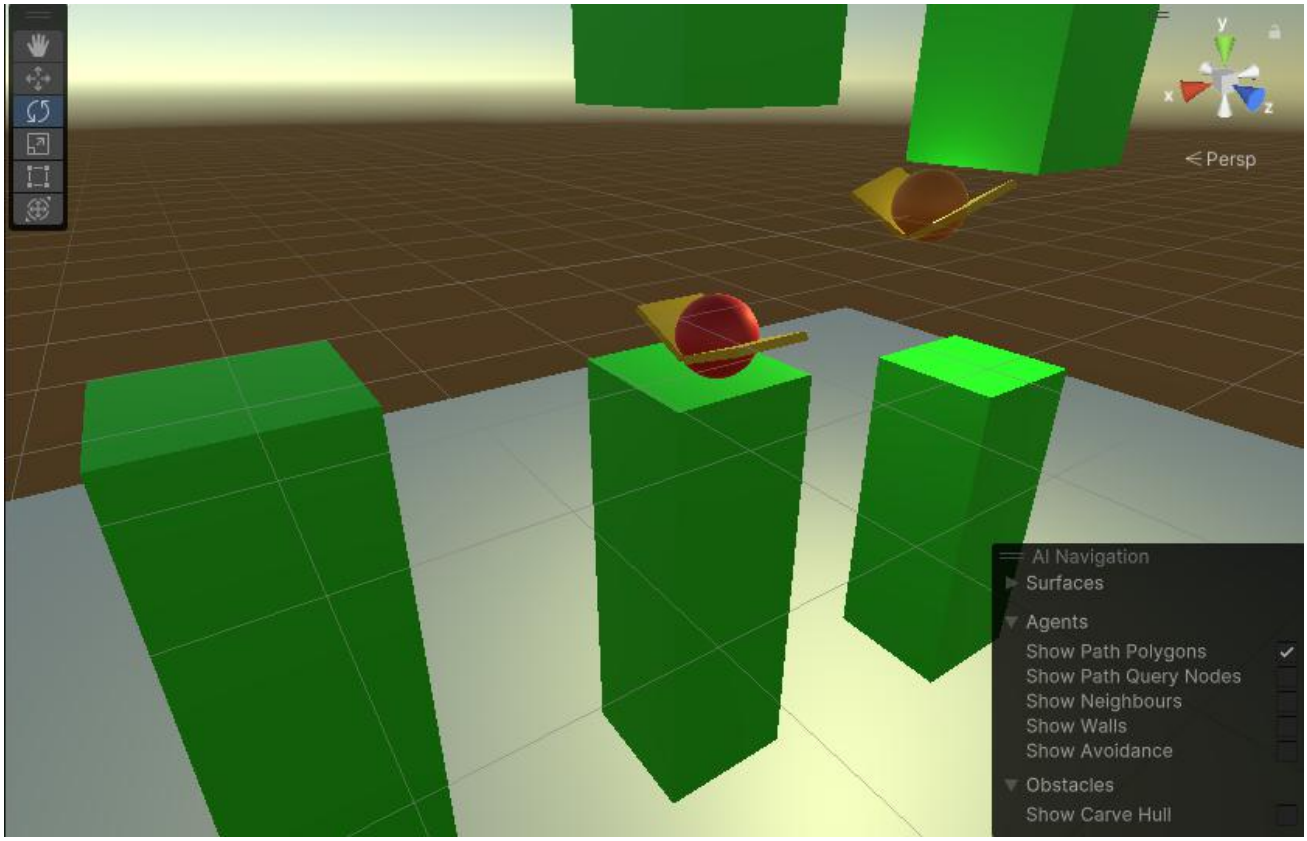


Рисунок 1.15. Фінальна сцена з розташованими префабами (дизайн гри)

1.2.5 Проєктування архітектури ігрового застосунку

Архітектура гри в Unity 3D скаладається з асетів.

Асети в Unity 3D являють собою цифрові файли, які використовуються для створення контенту у грі, такі як 3D-моделі, текстури, аудіо, анімації та сценарії. Вони можуть бути створені самостійно, завантажені з Інтернету або придбані в Asset Store Unity.

На рис. 1.16 зображено файлову архітектуру даного ігрового застосунку.

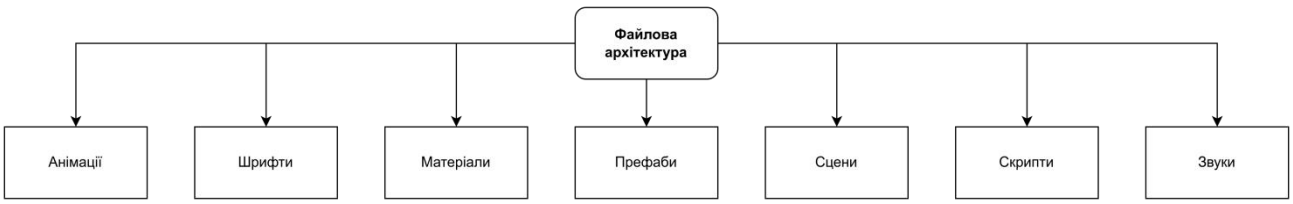


Рисунок 1.16. Файлова архітектура ігрового застосунку

На рис. 1.17 зображено фінальну файлову архітектуру в Unity 3D.

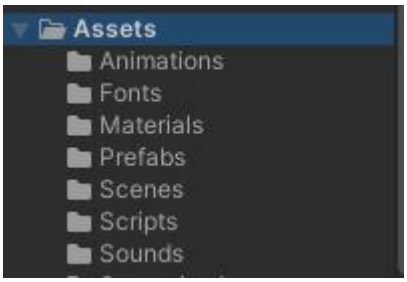


Рисунок 1.17. Фінальна файлова архітектуру в Unity 3D

Розглянемо створену архітектуру більш детально:

animations: Ця тека використовується для зберігання анімацій, які застосовуються до 3D-моделей. Анімації можуть бути створені самостійно, завантажені з Інтернету або придбані в Asset Store Unity.

fonts: Ця тека використовується для зберігання шрифтів, які використовуються в тексті. Шрифти можуть бути у форматах TTF, OTF або TrueType.

materials: Ця тека використовується для зберігання матеріалів, що визначають зовнішній вигляд 3D-об'єктів. Матеріали включають текстури, колір, освітлення та інші властивості.

prefabs: Ця тека використовується для зберігання префабів, які є шаблонами 3D-об'єктів. Префаби використовуються для швидкого та простого створення повторюваних елементів у грі.

scenes: Ця тека використовується для зберігання сцен, які є окремими рівнями або ігровими зонами у грі. Кожна сцена може містити 3D-моделі, освітлення, камери та інші елементи.

scripts: Ця тека використовується для зберігання сценаріїв, які є фрагментами коду, що керують поведінкою 3D-об'єктів та ігровим процесом. Сценарії пишуться мовою C#.

sounds: Ця тека використовується для зберігання звукових файлів, які використовуються у грі. Звукові файли можуть бути у форматах MP3, WAV або OGG.

1.2.6 Проєктування графічного інтерфейсу ігрового застосунку

Графічний інтерфейс ігрового застосунку складається з канвасів.

Канвас в Unity 3D являє собою компонент, який визначає область для розміщення елементів користувальницького інтерфейсу (UI). На канвасі Unity 3D можна розташувати різні елементи UI, такі як кнопки, текстові поля, зображення, панелі прокрутки та інші.

На рис. 1.18 зображено графічний інтерфейс ігрового застосунку (екран налаштувань).

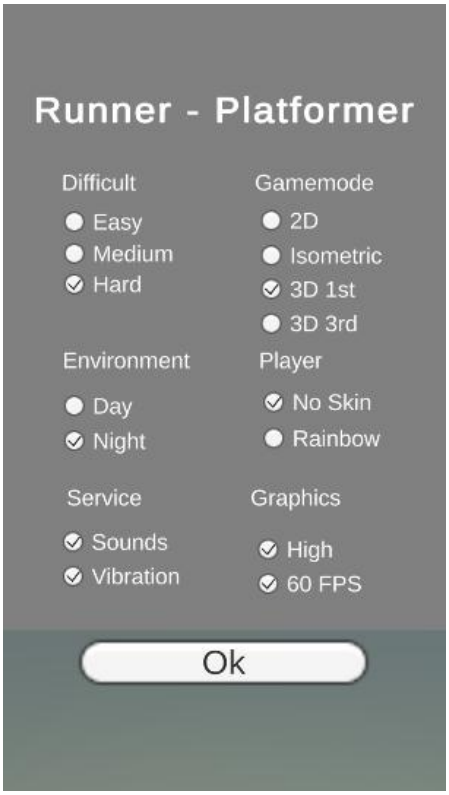


Рисунок 1.18. Графічний інтерфейс ігрового застосунку (екран налаштувань)

На рис. 1.19 зображено фінальну ієрархію елементів сцени (канвасів та перфабів) ігрового застосунку.

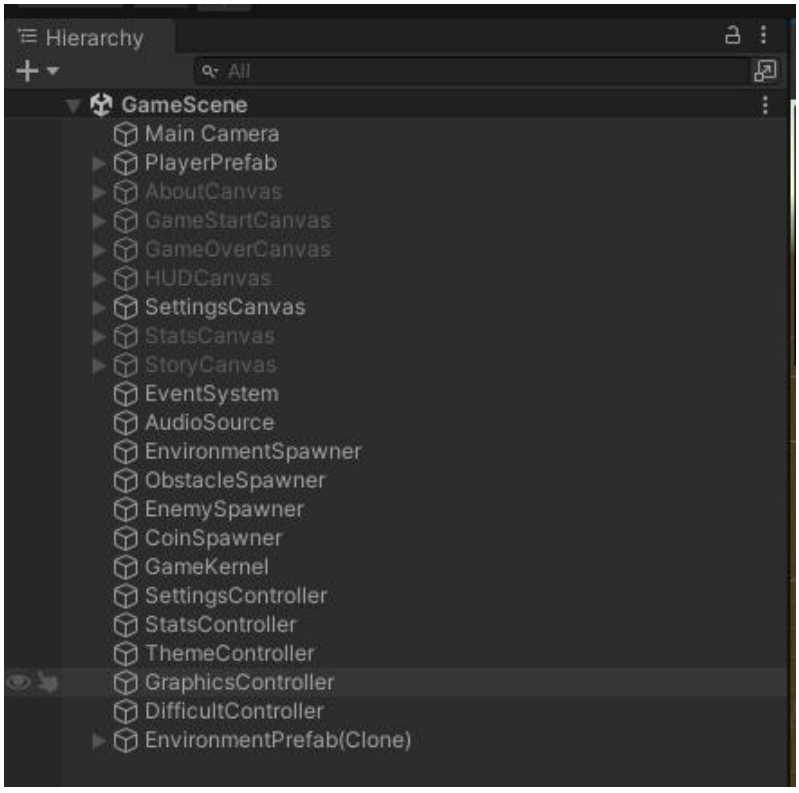


Рисунок 1.19. Фінальна ієрархія елементів сцени ігрового застосунку

1.3 Реалізація ігрового застосунку

1.3.1 Налаштування Unity проєкту

Unity проєкт створюється засобми Unity Hub. Unity Hub є інструментом для управління інсталяціями Unity Editor, створення нових проєктів та доступу до робочих матеріалів.

На рис. 1.20 зображено створений проєкт ігрового застосунку в Unity Hub.

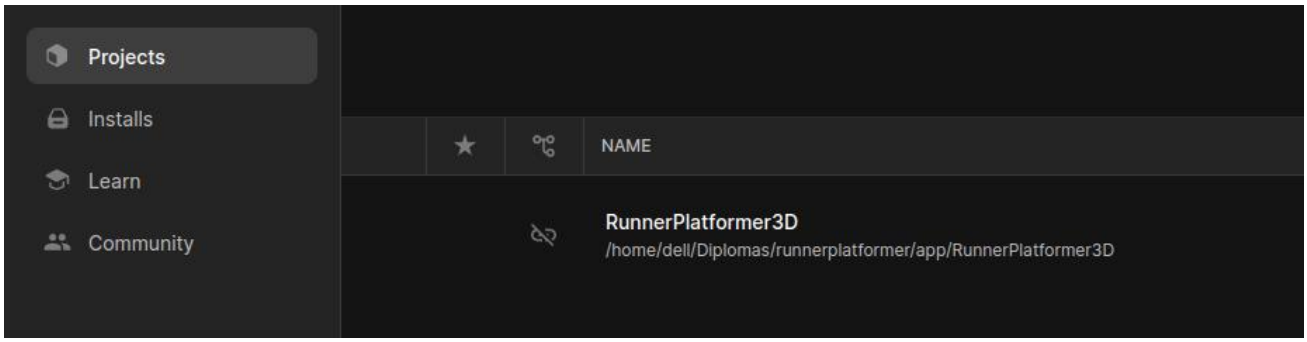


Рисунок 1.20. Створений проєкт ігрового застосунку в Unity Hub

1.3.2 Логіка персонажів

Для реалізації логіки того чи іншого об’єкту в Unity 3D використовуються компоненти.

У Unity 3D компонент являє собою блок коду, який додає певну функціональність до ігрового об'єкта. Компоненти використовуються для визначення поведінки об'єкта, його зовнішнього вигляду та взаємодії з іншими об'єктами в ігровому світі.

На рис. 1.21 зображено додані компоненти до префабу з гравцем.

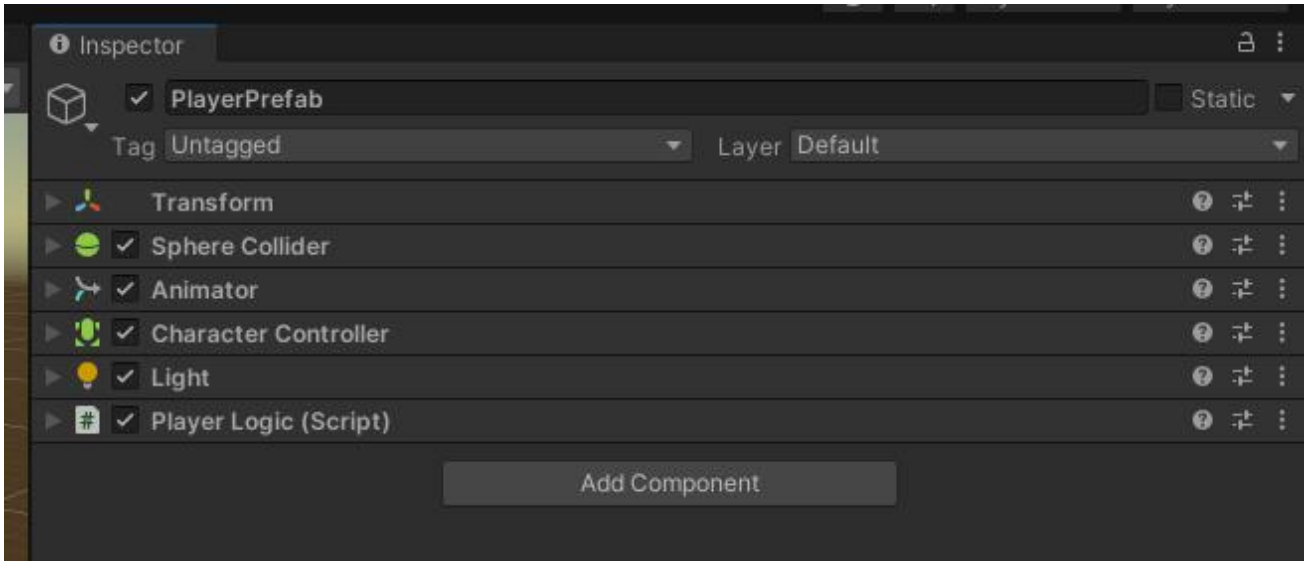


Рисунок 1.21. Додані компоненти до префабу з гравцем

Опишемо кожен компонент більш детально:

Transform: Цей компонент визначає положення, обертання та масштаб ігрового об'єкта в 3D-просторі. Transform є обов'язковим компонентом для всіх ігрових об'єктів. Sphere Collider: Цей компонент додає до ігрового об'єкта сферичну форму зіткнення. Sphere Collider використовується для виявлення зіткнень з іншими об'єктами в ігровому світі. Це може бути корисно для реалізації фізики, реагування на зіткнення або простого блокування переміщення об'єкта.

Animator: Цей компонент використовується для керування анімаціями 3D-моделі. Animator може відтворювати анімації з часової шкали, реагувати на події або переходити між анімаціями залежно від ігрових умов.

Character Controller: Цей компонент використовується для керування рухом персонажа в 3D-світі. Character Controller забезпечує реалістичне переміщення по ґрунту, сходах та інших нерівних поверхнях. Він також може використовуватися для запобігання падінню персонажа з країв або проходження крізь стіни.

Light: Цей компонент додає до ігрового об'єкта джерело світла. Light може висвітлювати інші об'єкти в сцені, створюючи реалістичні тіні та освітлення.

Існує декілька типів джерел світла, таких як спрямоване світло, точкове світло та прожектор.

Player Logic (Script): Цей компонент використовується для реалізації поведінки та ігрової логіки для персонажа гравця. Player Logic може включати код для керування рухом, взаємодії з предметами, бою з ворогами та інших аспектів ігрового процесу. Сценарій Player Logic пишеться мовою C#.

Розглянемо коду скрипту PlayerLogic.

Нижче навено код скрипту PlayerLogic.cs:

```
using UnityEngine;

public class PlayerLogic : MonoBehaviour
{
    public CharacterController controller;
    public Vector3 playerVelocity;
    public Quaternion downRotation;
    public Quaternion forwardRotation;
    public AudioSource source;

    public bool groundedPlayer;
    public float playerSpeed;
    public float jumpHeight = 1.0f;
    public float gravityValue = -9.81f;

    public GameKernel gameKernel;
    public StatsController statsController;

    public AudioClip wingSound;
    public AudioClip crashSound;
    public AudioClip scoreSound;

    public bool vibrate;

    public float timeRemaining = 5f;

    private void Start()
    {
        controller = GetComponent<CharacterController>();
        forwardRotation = Quaternion.Euler(0, 0, 30);
        downRotation = Quaternion.Euler(0, 0, -90);
    }

    void Update()
    {
        if (Time.timeScale > 0)
        {
            groundedPlayer = controller.isGrounded;

            if (groundedPlayer && playerVelocity.y < 0)
            {
                playerVelocity.y = 0f;
            }
```

```
                if (Input.GetMouseButtonDown(0))
                {
                    if (vibrate == true)
                    {
                        //Handheld.Vibrate(); for mobile
                    }
                    source.PlayOneShot(wingSound);
                    transform.rotation = forwardRotation;
                    playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f * gravityValue);
                }
                else
                {
                    playerVelocity.x = playerSpeed;
                    playerVelocity.y += gravityValue * Time.deltaTime;
                    controller.Move(playerVelocity * Time.deltaTime);
                    transform.rotation      =      Quaternion.Lerp(transform.rotation,
downRotation, playerSpeed * Time.deltaTime);
                }
            }

            if (gameKernel.isReview == true)
            {
                startTimer();
            }
        }

        public void OnBecameInvisible()
        {
            source.PlayOneShot(crashSound);
            gameKernel.GameOverState();
        }

        void startTimer()
        {
            if (timeRemaining > 0)
            {
                timeRemaining -= Time.deltaTime;
            }
            else if (timeRemaining <= 0)
            {
                timeRemaining = 5;
                gameKernel.isReview = false;
                statsController.coin = false;
            }

        }

        public void OnControllerColliderHit(ControllerColliderHit hit)
        {
            if (gameKernel.isReview == false)
            {
                source.PlayOneShot(crashSound);
                gameKernel.GameOverState();
            }
        }

        public void OnTriggerEnter(Collider collider)
        {
            if (collider.gameObject.tag == ("Obscale"))
            {
                statsController.sessionScore += 1;
```

```
                source.PlayOneShot(scoreSound);
            }

            if (collider.gameObject.tag == ("Enemy"))
            {
                Destroy(collider.gameObject);
            }

            if (collider.gameObject.tag == ("Coin"))
            {
                Destroy(collider.gameObject);
                statsController.coin = true;
            }
        }

    }
```

Цей код написаний на мові C# для використання в Unity, і визначає поведінку об'єкта під назвою PlayerLogic, який контролює гравця у грі.

CharacterController використовується для керування рухом гравця.

playerVelocity зберігає швидкість гравця у вигляді вектора.

downRotation та forwardRotation визначають повороти гравця.

AudioSource використовується для відтворення звукових ефектів.

groundedPlayer визначає, чи знаходиться гравець на землі.

playerSpeed задає швидкість руху гравця.

jumpHeight визначає висоту стрибка гравця.

gravityValue задає значення гравітації.

GameKernel та StatsController відповідають за стан гри та статистику відповідно.

wingSound, crashSound та scoreSound зберігають звукові ефекти для різних подій.

vibrate визначає, чи буде вібрація при натисканні.

timeRemaining задає таймер для рецензування.

У методі Start налаштовується початковий стан гравця.

Метод Update обробляє вхідні дані та оновлює стан гравця кожен кадр.

Якщо гравець натискає ліву кнопку миші, він стрибає і відтворюється звуковий ефект.

Якщо кнопка не натиснута, гравець рухається вперед і піддається гравітації.

Метод OnBecameInvisible викликається, коли гравець стає невидимим на екрані, відтворюється звуковий ефект і змінюється стан гри на "Гра закінчена".

Метод startTimer зменшує таймер і, якщо час вичерпано, скидає його та вимикає режим рецензування.

Метод OnControllerColliderHit викликається при зіткненні гравця з іншими об'єктами і змінює стан гри на "Гра закінчена", якщо гравець не в режимі рецензування.

Метод OnTriggerEnter обробляє зіткнення з об'єктами, такими як перешкоди, вороги та монети, оновлюючи рахунок, знищуючи об'єкти або відтворюючи звукові ефекти відповідно.

1.3.3 Логіка рівню

Для реалізації логіки того чи іншого об’єкту в Unity 3D використовуються компоненти.

На рис. 1.22 зображено додані компоненти до префабу з перешкодою.

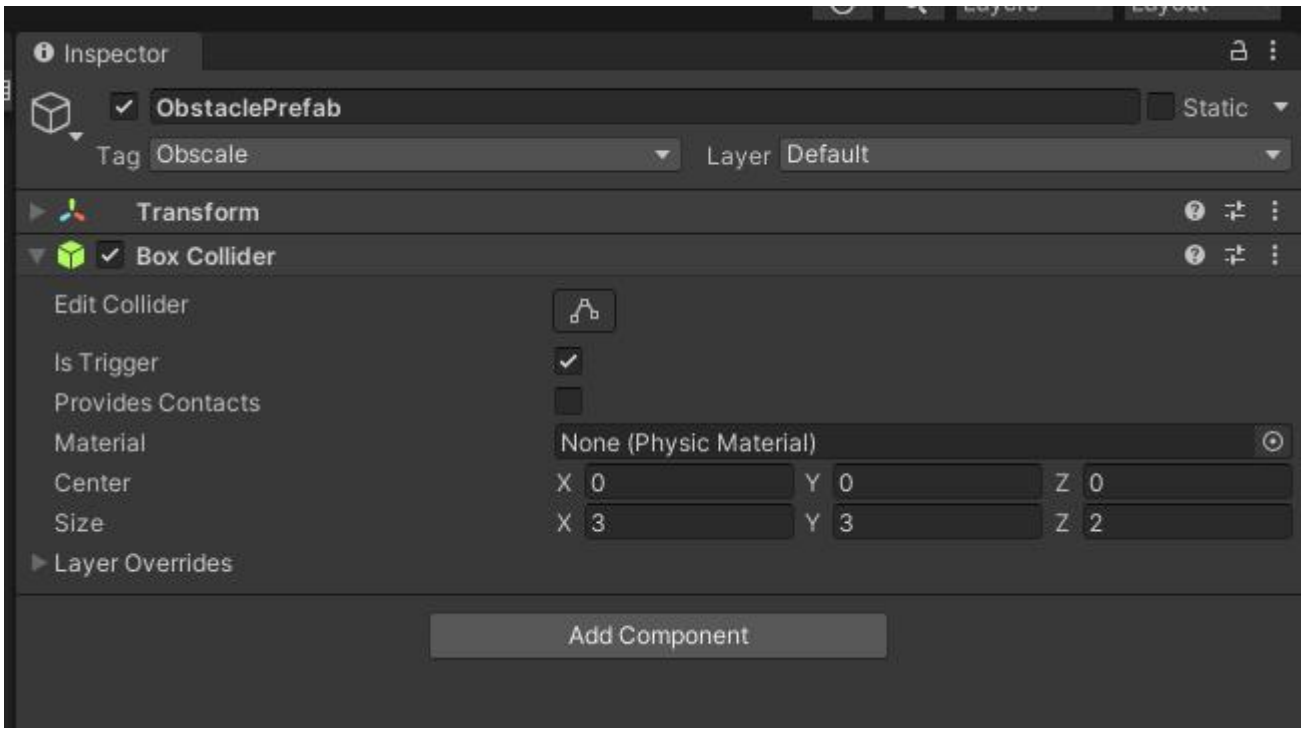


Рисунок 1.22. Додані компоненти до префабу з гравцем

Опишемо кожен компонент більш детально:

Transform: базовий компонент Unity 3D, який використовується для представлення позиції, обертання та масштабу об'єкта в 3D-просторі. Він є фундаментальним компонентом, який використовується практично для всіх об'єктів у вашій сцені.

Box Collider: Тип колайдера в Unity 3D, який використовується для визначення форми та розміру об'єкта для цілей виявлення зіткнень. Він описує об'єкт як прямокутну призму, що спрощує розрахунки зіткнень.

Для створення об’єктів під час ігрового процесу в Unity 3D можна використовувати спавнер.

Спавнер в Unity 3D являє собою скрипт або ігровий об'єкт, який відповідає за створення та розміщення інших об'єктів у сцені. Він діє як машина, що генерує елементи гри протягом певного часу.

На рис. 1.23 зображено додані компоненти на пустий об’єкт для спавнеру перешкод.

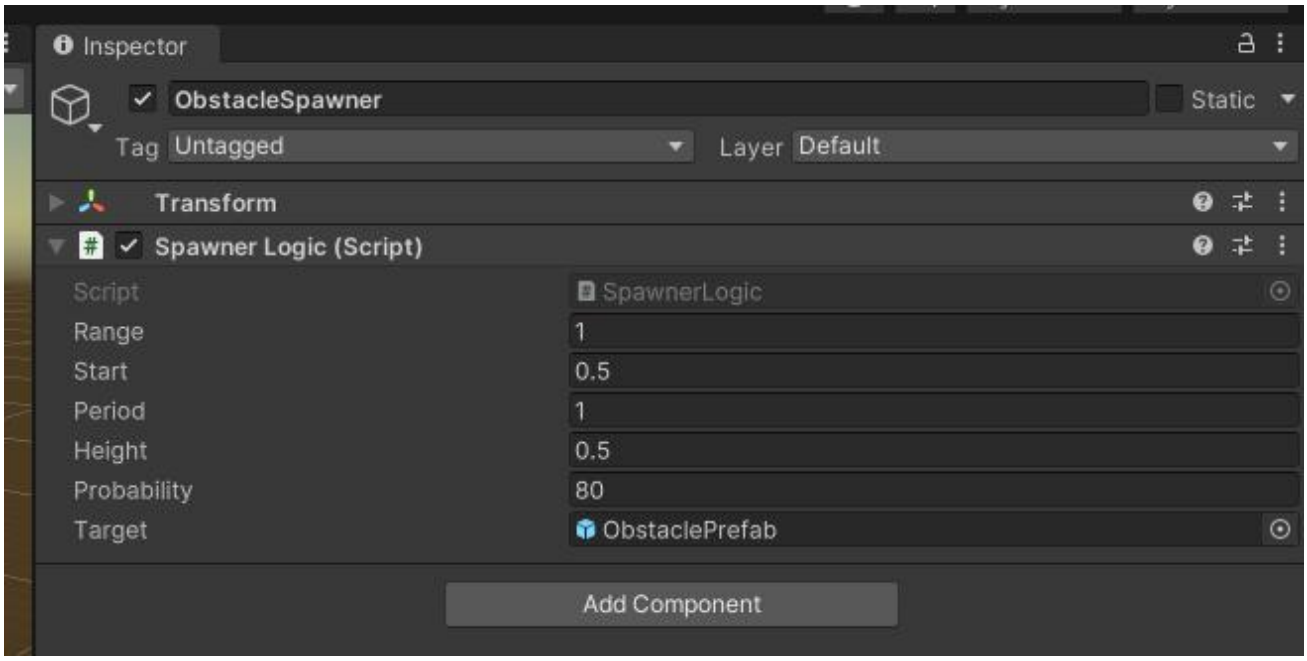


Рисунок 1.23. Додані компоненти на пустий об’єкт для спавнеру перешкод

Опишемо кожен компонент більш детально:

Transform: базовий компонент Unity 3D, який використовується для представлення позиції, обертання та масштабу об'єкта в 3D-просторі. Він є

фундаментальним компонентом, який використовується практично для всіх об'єктів у сцені.

SpawnerLogic: саморобний компонент у проєкті Unity 3D, а не базовий компонент Unity. Його точна функціональність відповідає за створення та керування появою інших об'єктів у сцені.

Розглянемо коду скрипту SpawnerLogic.

Нижче навено код скрипту SpawnerLogic.cs:

```
using UnityEngine;

public class SpawnerLogic : MonoBehaviour
{
    public float range;
    public float start;
    public float period;
    public float height;
    public int probability;

    public GameObject target;

    public void Start()
    {
        InvokeRepeating("Spawn", start, period);
    }

    void Spawn()
    {
        range += 5f;

        if (Random.Range(0, 100) < probability)
        {
            GameObject newTarget = Instantiate(target);
            newTarget.transform.position = transform.position + new Vector3(range,
UnityEngine.Random.Range(-height, height), Random.Range(-0.5f, 0.5f));
        }
    }

    void OnBecameInvisible()
    {
        Destroy(this.gameObject);
    }
}
```

Цей код написаний на мові C# для використання в Unity, і визначає поведінку об'єкта під назвою SpawnerLogic, який відповідальний за спавн інших об'єктів у грі.

Скрипт успадковується від класу MonoBehaviour, що дозволяє йому використовувати основні функції Unity, такі як Start і Update.

range визначає початкову дистанцію для спавну об'єктів.

start задає час затримки перед початком спавну.

period визначає інтервал часу між спавнами.

height визначає вертикальний діапазон, в якому можуть з'являтися нові об'єкти.

probability задає ймовірність спавну нового об'єкта при кожній спробі.

target є префабом об'єкта, який буде створюватись спавнером.

Метод Start викликається при запуску скрипта і встановлює повторюваний виклик методу Spawn з початковою затримкою і заданим періодом.

Метод Spawn відповідає за логіку спавну об'єктів.

З кожним викликом методу Spawn, значення range збільшується на 5 одиниць.

Використовується метод Random.Range для генерації випадкового числа від 0 до 100.

Якщо випадкове число менше, ніж значення probability, створюється новий об'єкт target.

Новий об'єкт розміщується в новій позиції, яка розраховується на основі range, випадкової вертикальної позиції в межах height, і невеликого випадкового горизонтального зсуву.

Метод OnBecameInvisible викликається, коли об'єкт спавнера стає невидимим на екрані.

У методі OnBecameInvisible об'єкт спавнера знищується.

InvokeRepeating метод викликає метод Spawn повторювано з заданим інтервалом.

Instantiate створює копію об'єкта target.

transform.position визначає поточну позицію об'єкта спавнера.

Скрипт ефективно управляє спавном об'єктів в залежності від ймовірності та параметрів часу.

Код реалізує механіку поступового збільшення дальності спавну, що дозволяє об'єктам з'являтись на різних відстанях від спавнера.

1.3.4 Логіка графічного інтерфейсу

Для управління графічним інтерфейсом та станом ігрового застосунку використовується концепція ядра гри.

В Unity 3D ядро гри являє собою скрипт, який слугує центральним хабом для керування загальним перебігом та станом вашої гри. Це не вбудований компонент Unity, а скрипт, який ви створюєте для керування різними аспектами гри.

Розглянемо коду скрипту GameKernel.

Нижче навено код скрипту GameKernel.cs:

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

public class GameKernel : MonoBehaviour
{
    public PlayerLogic playerLogic;
    public SpawnerLogic spawnerLogic;
    public ThemeController themeController;
    public StatsController statsController;
    public SettingsController settingsController;

    public GameObject GameStartCanvas;
    public GameObject GameOverCanvas;
    public GameObject StoryCanvas;
    public GameObject StatsCanvas;
    public GameObject SettingsCanvas;
    public GameObject AboutCanvas;
    public GameObject HUDCanvas;

    public Button ReviewGameOverCanvas;

    public bool isGameOver = false;
    public bool isReview = false;

    public void Start()
    {
        settingsController.DifficultToggleGroupChanged();
        settingsController.GamemodeToggleGroupChanged();
        settingsController.EnvironmentToggleGroupChanged();
        settingsController.PlayerToggleGroupChanged();
        settingsController.SoundToggleGroupChanged();
        settingsController.VibrateToggleGroupChanged();

        Time.timeScale = 0;

        GameStartCanvas.SetActive(true);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
```

```
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(false);
    }

    public void GameOverState()
    {
        isGameOver = true;
        isReview = false;

        Time.timeScale = 0;

        if (statsController.coin == true)
        {
            ReviewGameOverCanvas.interactable = true;
        }
        else
        {
            ReviewGameOverCanvas.interactable = false;
            themeController.BloodEffect();
        }

        statsController.SetStats();
        GameStartCanvas.SetActive(false);
        GameOverCanvas.SetActive(true);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(false);
    }

    public void StartGameState()
    {
        Time.timeScale = 1;

        GameStartCanvas.SetActive(false);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(true);
    }

    public void ReviewState()
    {
        isReview = true;
        isGameOver = false;

        Time.timeScale = 1;

        GameStartCanvas.SetActive(false);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(true);
    }

    public void SettingsState()
```

```
        {
            GameStartCanvas.SetActive(false);
            GameOverCanvas.SetActive(false);
            StoryCanvas.SetActive(false);
            StatsCanvas.SetActive(false);
            SettingsCanvas.SetActive(true);
            AboutCanvas.SetActive(false);
            HUDCanvas.SetActive(false);
        }

        public void AboutState()
        {
            GameStartCanvas.SetActive(false);
            GameOverCanvas.SetActive(false);
            StoryCanvas.SetActive(false);
            StatsCanvas.SetActive(false);
            SettingsCanvas.SetActive(false);
            AboutCanvas.SetActive(true);
            HUDCanvas.SetActive(false);
        }

        public void StoryState()
        {
            GameStartCanvas.SetActive(false);
            GameOverCanvas.SetActive(false);
            StoryCanvas.SetActive(true);
            StatsCanvas.SetActive(false);
            SettingsCanvas.SetActive(false);
            AboutCanvas.SetActive(false);
            HUDCanvas.SetActive(false);
        }

        public void StatsState()
        {
            statsController.SetStats();

            GameStartCanvas.SetActive(false);
            GameOverCanvas.SetActive(false);
            StoryCanvas.SetActive(false);
            StatsCanvas.SetActive(true);
            SettingsCanvas.SetActive(false);
            AboutCanvas.SetActive(false);
            HUDCanvas.SetActive(false);
        }

        public void MainMenuState()
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        }

        public void ExitState()
        {
            Application.Quit();
        }
        public void PauseResumeState()
        {
            if (Time.timeScale == 1)
            {
                Time.timeScale = 0;
            }
            else
```

```
            {
                Time.timeScale = 1;
            }
        }

    }
```

Цей код написаний на мові C# для використання в Unity, і визначає логіку гри через об'єкт під назвою GameKernel.

PlayerLogic, SpawnerLogic, ThemeController, StatsController, і SettingsController є різними компонентами гри, які керують відповідними аспектами.

GameStartCanvas, GameOverCanvas, StoryCanvas, StatsCanvas, SettingsCanvas, AboutCanvas, і HUDCanvas є різними UI-елементами для різних станів гри.

ReviewGameOverCanvas є кнопкою на екранному інтерфейсі, яка дозволяє гравцю переглянути стан після завершення гри.

isGameOver і isReview є булевими змінними, які відстежують стан гри.

У методі Start, різні налаштування ініціалізуються через виклик методів з SettingsController.

Початковий стан гри зупинений (Time.timeScale = 0), і активний лише GameStartCanvas.

Метод GameOverState змінює стан гри на "Гра закінчена", зупиняє час і активує відповідні UI-елементи.

У цьому стані, якщо гравець зібрав монету, кнопка ReviewGameOverCanvas стає активною; інакше, активується ефект крові.

Метод StartGameState починає гру, запускає час і активує HUD.

Метод ReviewState встановлює режим перегляду гри, запускає час і активує HUD.

Метод SettingsState активує налаштування UI, де гравець може змінювати налаштування гри.

Метод AboutState активує UI з інформацією про гру.

Метод StoryState активує UI для перегляду історії гри.

Метод StatsState оновлює і активує UI з ігровою статистикою.

Метод MainMenuState перезавантажує поточну сцену, повертаючи гру до початкового стану.

Метод ExitState закриває застосунок.

Метод PauseResumeState змінює стан гри між паузою і відновленням, перемикаючи Time.timeScale між 0 і 1.

У методі GameOverState, після активації GameOverCanvas, інші UI-елементи вимикаються.

Метод StartGameState вимикає всі інші UI-елементи, крім HUD.

Метод PauseResumeState дозволяє гравцю призупиняти та відновлювати гру, що важливо для підтримки гнучкості в геймплеї.

На рис. 1.24 зображено фінальні створені скрипти.

Рисунок 1.24. Фінальні створені скрипти

1.4 Тестування та огляд ігрового застосунку

Проведемо мануальне тестування ігрового застосунку з метою відтворення досвіду кінцевого користувача та пошуку помилок.

Мануальне тестування (ручне тестування) являє собою процес перевірки програмного забезпечення, під час якого тестувальники вручну виконують тестові сценарії без використання автоматизованих інструментів. Основна мета мануального тестування полягає в оцінці якості програмного забезпечення шляхом ідентифікації дефектів та недоліків, які можуть вплинути на кінцевий досвід користувача.

1.4.1 Тестування графічного інтерфейсу

Розглянемо головний екран ігрового застосунку.

На рис. 1.25 зображено тестування графічного інтерфейсу, а саме головного екрану.

Рисунок 1.25. Головний екран

На рис. 1.26 зображено тестування графічного інтерфейсу, а саме екрану з сюжетом.

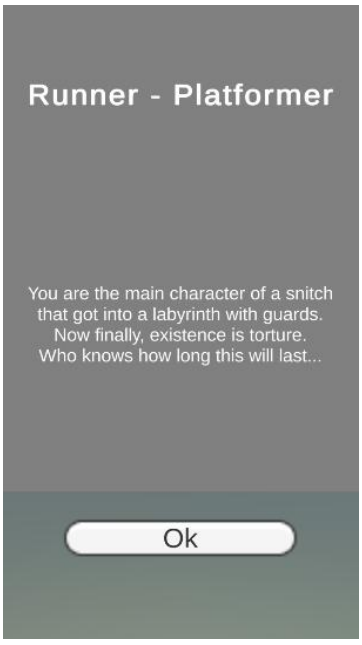


Рисунок 1.26. Сюжетний екран

На рис. 1.27 зображено тестування графічного інтерфейсу, а саме екрану зі статистикою.

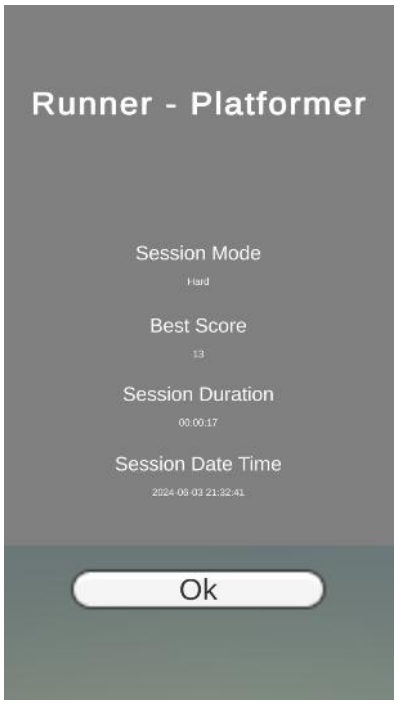


Рисунок 1.27. Екран статистики

На рис. 1.28 зображено тестування графічного інтерфейсу, а саме екрану зі налаштуваннями.

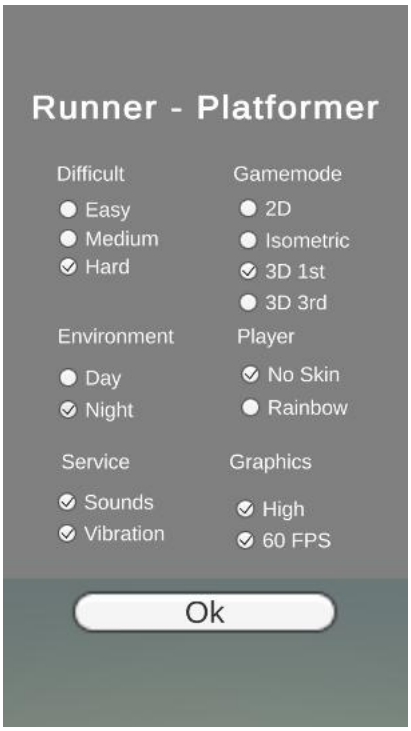


Рисунок 1.28. Екран налаштувань

На рис. 1.29 зображено тестування графічного інтерфейсу, а саме екрану зі довідкою.

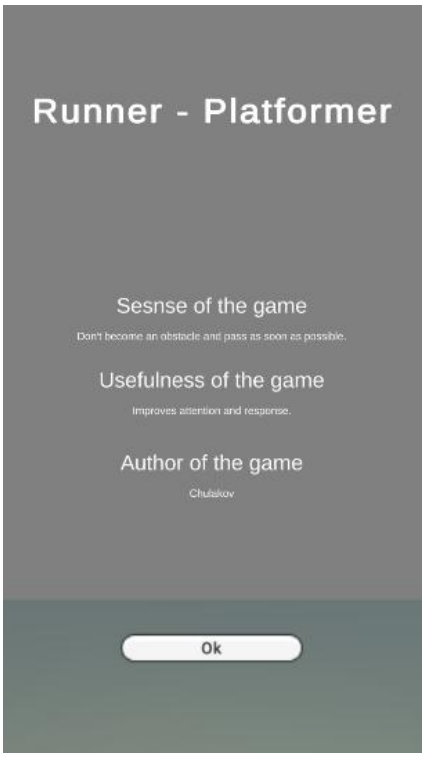


Рисунок 1.29. Екран довідки

На рис. 1.30 зображено тестування графічного інтерфейсу, а саме екрану про кінець гри.

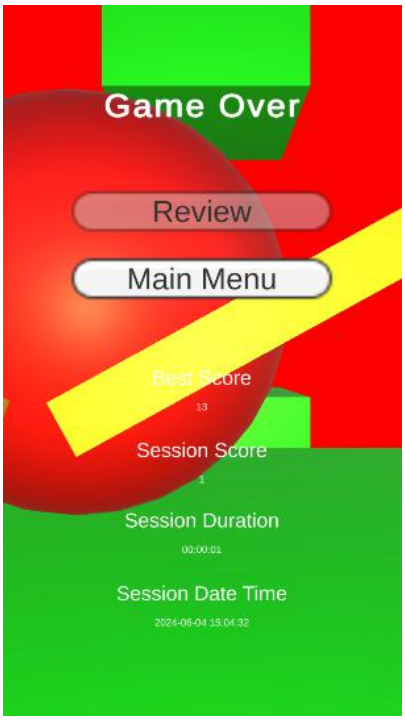


Рисунок 1.30. Екран про кінець гри

1.4.2 Тестування оточення

Розглянемо налаштування дні і ночі.

На рис. 1.31 зображено пори дня – день.

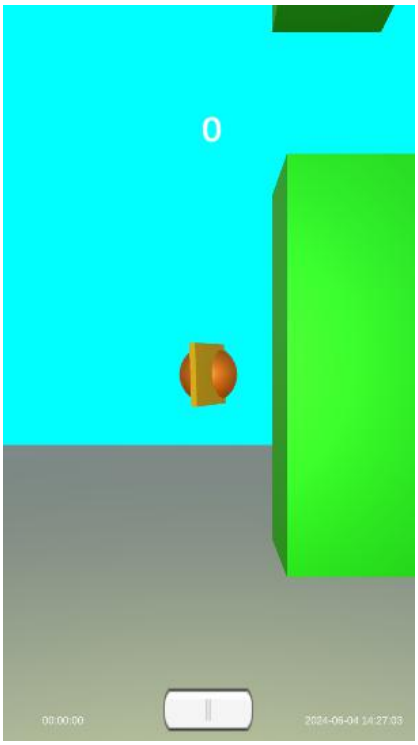


Рисунок 1.31. Пора дня – день

На рис. 1.32 зображено пори дня – ніч.

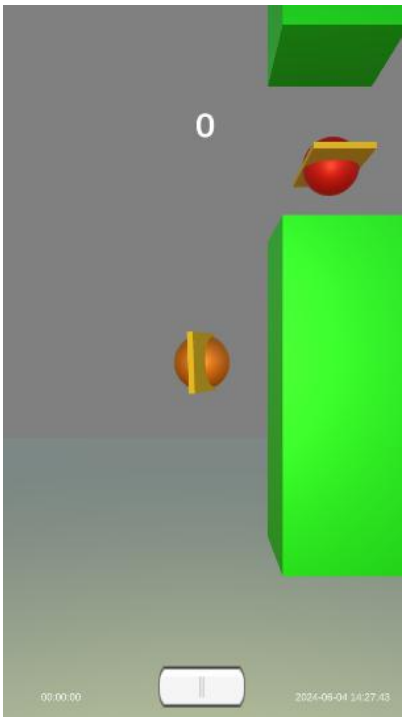


Рисунок 1.32. Пора дня – ніч

1.4.3 Тестування режимів камери

Розглянемо налаштування куту камери.

На рис. 1.33 зображено ігровий процес у режим 2D.

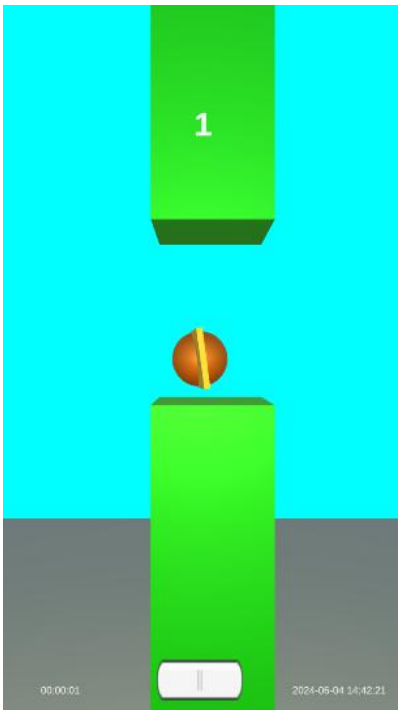


Рисунок 1.33. 2D режим

На рис. 1.34 зображено ігровий процес у ізометричному режимі 3D.

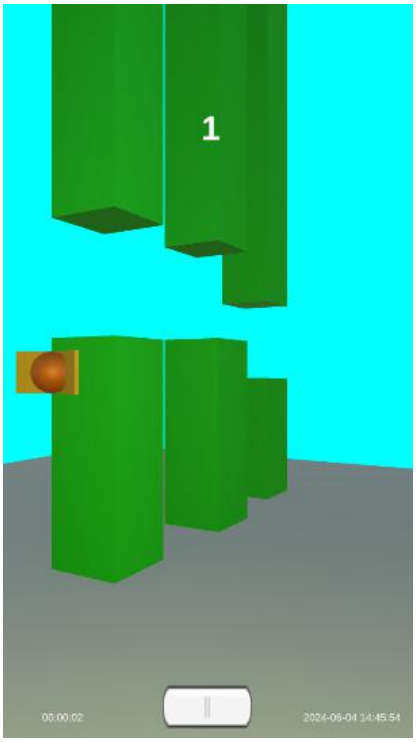


Рисунок 1.34. Ізометричний 3D режим

На рис. 1.35 зображено ігровий процес у режимі 3D від третьої особи.

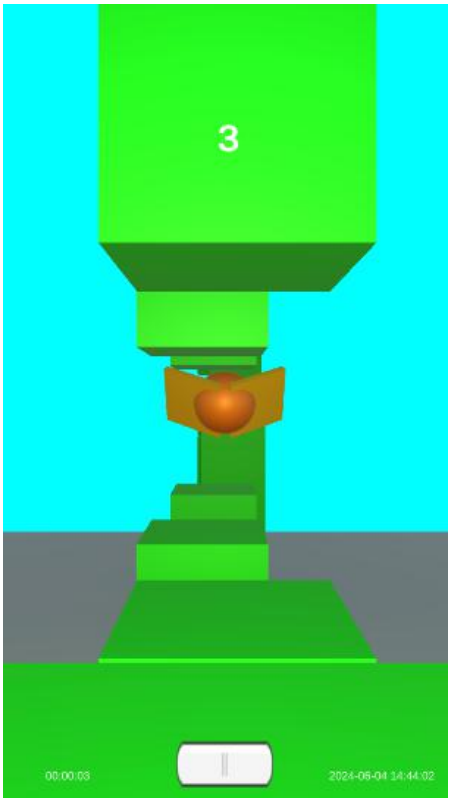


Рисунок 1.35. 3D режим від третьої особи

На рис. 1.36 зображено ігровий процес у режимі 3D від першої особи.

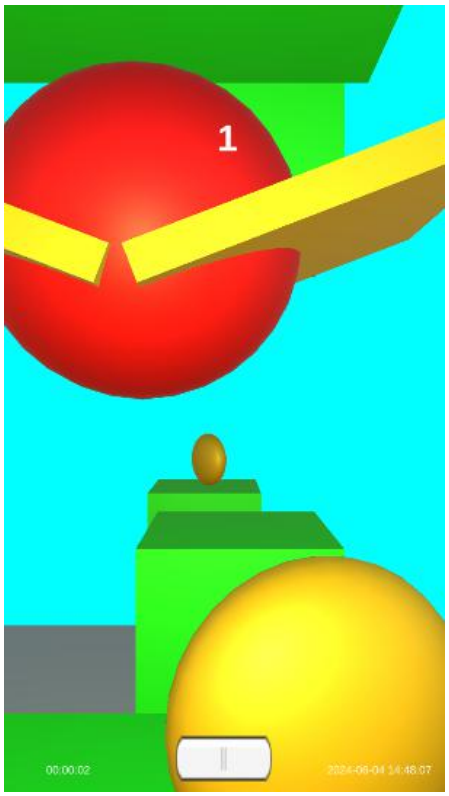


Рисунок 1.36. 3D режим від першої особи

1.4.4 Тестування складності

На рис. 1.37 зображено ігровий процес у режимі високої складності (відстань між перепонами дуже маленька).

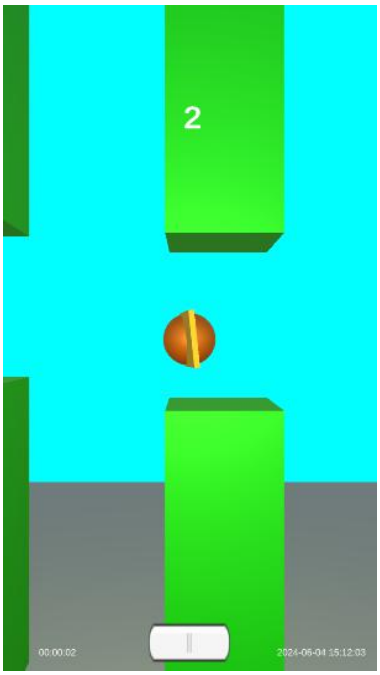


Рисунок 1.37. Режим високої складності

На рис. 1.38 зображено ігровий процес у режимі низької складності (відстань між перепонами дуже велика).

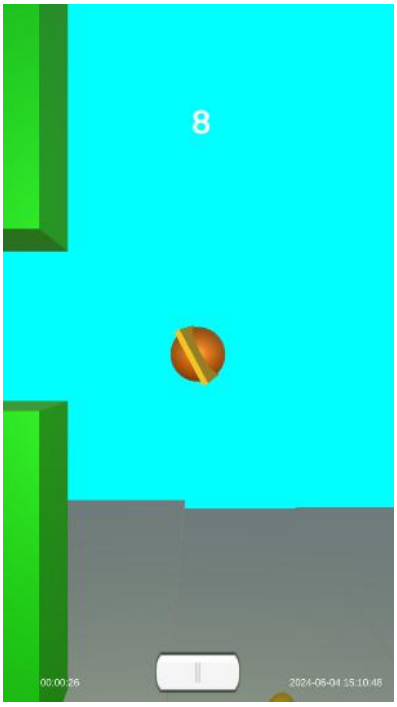


Рисунок 1.38. Режим низької складності

1.4.5 Результати тестування

Мануальне тестування гри-раннера платформера завершилося позитивно.

Графічний інтерфейс на головному екрані, сюжетному екрані, екрані зі статистикою, екрані налаштувань, екрані довідки та екрані про кінець гри працює коректно, всі елементи відображаються правильно, а кнопки функціонують належним чином.

Налаштування оточення, зокрема зміна дня та ночі, успішно перевірено, і жодних проблем не виявлено.

Тестування різних режимів камери, включаючи 2D режим, ізометричний 3D режим, 3D режим від третьої особи та 3D режим від першої особи, показало, що всі режими працюють без збоїв.

Високий та низький рівні складності також перевірені, і гра правильно реагує на ці налаштування, відповідно змінюючи відстань між перепонами.

Тестування підтвердило, що гра стабільно працює в усіх перевірених аспектах, забезпечуючи позитивний користувацький досвід. Загалом, тестування завершено успішно, і жодних критичних помилок не виявлено.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проєкті було створено ігровий застосунок у жанрі раннер-платформер, побудований з використанням сучасних технологій розробки, що відповідає актуальним вимогам ігрового ринку. Дослідження підтвердили, що створення такого застосунку є виправданим для покращення геймплею та надання користувачам нових вражень. Застосування новітніх технологій дозволяє швидко адаптуватися до змін у вподобаннях гравців, оптимізувати ігрові процеси та забезпечувати високий рівень задоволення від гри.

Програмне забезпечення було розроблено за допомогою Unity 3D, який виступає рушієм для створення ігрового середовища, та мови програмування C#, що забезпечила об'єктно-орієнтований підхід до кодування. Використання цих інструментів дозволило створити конкурентоспроможний ігровий продукт.

Ефективність програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

Проведемо розрахунки визначення трудомісткості розробки даного програмного продукту.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

У табл. 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
7. ПП загальної математики і ПП імітаційного моделювання	7800 – 8800

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл. 2.2.

Таблиця 2.2. Обсяг команд

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар=229х0,8=183,2 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \leftrightarrow L_1 \leftrightarrow K_H \quad (2.1)$$

$$T_{ТП} = T^a p \leftrightarrow L_2 \leftrightarrow K_H \quad (2.2)$$

$$T_{РП} = T^a p \leftrightarrow L_3 \leftrightarrow K_H \leftrightarrow K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеног параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Ттз=Та*L1*Кн=183,2*0,12*0,7=15,39 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проєкту

$$Ттп=Та*L2*Кн=183.2*0,11*0,7=17,42 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проєкту

$$Трп=Та*L3*Кн*Кт=183,2*0,61*0,7*0,7=54,76 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап:технічне завдання Nтз=2 (стр), розробка ТП Nтп=40(стр), розробка робочого проєкту Nрп=9 (стр), пояснювальна записка відповідно Nпз=25 (стр).

Розрахунок зведений у табл. 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин.**		
1.ТЗ	$T_{Ртз}=15,39$	$T_{кк}=0,7*N_{тз}=$ 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=$ 0,15*2=0,30
2.Розробка ТП	$T_{Ртп}=14,12$	$T_{кк}=0,7*N_{тп}=$ 0,7*40=28	$T_{нк}=0,15*N_{тп}=$ 0,15*40=6.0
3.Розробка РП	$T_{Ррп}=54,76$	$T_{кк}=0,7*N_{рп}=$ 0,7*12=8,4	$T_{нк}=0,15*N_{рп}=$ 0,15*12=1,8
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}=$ 1,5*21 =31,5	$T_{кк}=0,7*N_{тз}=$ 0,7*21=14,7	$T_{нк}=0,15*N_{пз}=$ 0,15*21 =3,15
Усього, в т.ч.:	179,52		
- на розробку	$T_{р}=115,77$		
- контроль		$T_{кк}=52,5$	
- нормоконтроль			$T_{нк}=11,25$

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у табл. 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 квітня 2024 року - 8000 гривень; мінімальну погодинну тарифну ставку – 46,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	115,77	50,00	5788,50
2.Контроль керівника	52,50	120,00	6300,00
3.Нормоконтроль	11,25	100,00	1125,00
Усього	-	-	Зо= 13 213,5

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в табл. 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	80	4.00	320,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \leftrightarrow Вм1 =$ 0,1x320=32,0
Усього				$В_м = В_{мі} + В_{тр\_з} =$ 352.0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в табл. 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	352.0	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	13 213,5	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	13 21,35	$Зд = 0,1 \leftrightarrow Зо$ =13 213,5х0,1
4.Відрахування до єдиного фонду соціального внеску	3 197,66	$Вє.с.в. = 0,22 \leftrightarrow (Зо + Зд)$ =0,22х (13 213,5+13 21,35)
5. Накладні витрати	5 285,4	$Внак. = 0,4 \leftrightarrow Зо$ =0,4 х 13 213,5
6. Повна собівартість	23 369,91	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 352.0+13 213,5+13 21,35+3 197,66+ 5 285,4

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100=(23\ 369,91*10)/100=23\ 36,99 \text{ грн.} \quad (2.4)$$

Де р – плановий рівень рентабельності (10-20%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_п+П=23\ 369,91+23\ 36,99 =25\ 706,90 \text{грн.} \quad (2.5)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р= Ц_о+ПДВ =25\ 706,90+25\ 706,90*0,2=30\ 848,28 \text{грн.} \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

В Україні приділяється велика увага питанням охорони життя і здоров'я своїх громадян, створенню безпечних умов праці роботодавцями, керівниками установ, організацій, проте кількість нещасних випадків, що трапляються на виробництві, залишається дуже великою.

Поліпшення умов та охорона праці стає одним з важливих напрямків матеріального та культурного рівня життя народу, а це, у свою чергу, сприяє зростанню якості та продуктивності праці, підвищенню соціально-економічних показників виробництва, зменшенню коштів на витрати від травматизму, професійних захворювань і аварій.

В дипломному розділі дипломного проекту розглядається питання охорони праці програміста на стадії вирішення ним питань розробки ігрового застосунку у жанрі раннер-платформер.

3.2 Небезпечні та шкідливі фактори в роботі програміста

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних та шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища (підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика тощо), надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

3.3 Вимоги до виробничого середовища

В сучасних умовах кожному працівнику необхідно постійно підтримувати високий фізичний, психологічний та фаховий рівень, запобігати виникненню випадків травматизму та профзахворювань.

Для установлення можливого впливу на здоров'я користувачів ВДТ виробничих чинників має значення ряд якісних характеристик робочого середовища. Це середовище у приміщеннях (офісах) в основному характеризується такими фізичними параметрами, як температура, вологість та електричний опір підлоги. Фізико-хімічні показники включають інформацію про вміст у повітрі іонів та різноманітних забруднювачів, а також деякі інші якісні характеристики середовища.

3.3.1 Мікроклімат

Висока температура повітря негативно позначається на функціональному стані людини. Хоч генерація теплоти дисплеєм досягає критичного рівня тільки у саму теплу пору року, необхідно створювати комфортні теплові умови постійно.

Оптимальні та допустимі мікрокліматичні параметри у приміщеннях повинні враховувати специфіку технологічного процесу при використанні комп'ютерів. Згідно з діючими у нашій країні нормативними документами (ДСН 3.3.6.042-99 «Державні санітарні норми мікроклімату виробничих приміщень») у холодні періоди року температура повітря, швидкість його руху та відносна вологість повітря повинні відповідно складати: 22-240С; 0,1 м/с; 40-60%. Температура повітря може коливатись у межах від 21 до 250С при збереженні інших параметрів мікроклімату.

В теплі періоди року температура повітря, його рухливість та відносна вологість повинні відповідно становити: 23-250С; 0,1-0,2 м/с; 40-60 %.

Оптимальним рівнем аероіонізації у зоні дихання користувача вважається вміст легких аерофонів обох знаків від 150 до 5000 у 1 см3 повітря.

Нормалізуючий вплив на склад повітря робочої зони справляють примусова вентиляція, захисні екрани (оснащені заземленням) та застосування іонізаторів.

3.3.2 Шум

Шум справляє негативний вплив на стан користувача, особливо при тривалому впливі . У користувача, діяльність якого пов’язана з переробкою інформації це виражається у знижені розумової працездатності, зростає кількість помилок, розвиток зорового стомлення, зміні відчуття кольорів, появі головного болю, послаблення уваги. Нормованим параметром шуму на робочих місцях є рівень 50 дБ. Основними заходами боротьби з шумом є усунення або ослаблення причин шуму в самому його джерелі у процесі проектування, використання засобів звукопоглинання, раціональне планування виробничих приміщень.

3.3.3 Освітлення

Освітлення у приміщеннях з ПК має бути змішаним – природним та штучним. Природне освітлення повинно здійснюватись у вигляді бічного освітлення та відповідати нормам ДБН В.2.5-28-2006 «Природне і штучне освітлення».

При природному освітленні слід передбачити наявність сонцезахисних засобів, що знижують перепади яскравостей між природним світлом та свіченням екрана ВДТ. З цією метою можна використовувати плівки з металізованим покриттям або жалюзі з вертикальними лам елями, що регулюються.

Штучне освітлення у приміщеннях з ВДТ треба здійснювати у вигляді комбінованої системи освітлення з використовуванням люмінесцентних джерел світла у світильниках загального освітлення. На робочих місцях має бути забезпечена рівномірна освітленість за допомогою переважно відбитого або розсіяного світлорозподілу. Світлових відблисків з клавіатури, екрана та від інших частин ВДТ у напрямку очей користувача не повинно бути.

Норма освітленості на робочих місцях складає 300-500лк.

3.3.4 Електробезпека

Причинами ураження працівника електрострумом можуть бути:

- Випадковий дотик до струмоведучих частин, у результаті ведення робіт поблизу або на цих частинах;
- Несправність захисних засобів, якими потерпілий доторкався до струмоведучих частин;

Значення сили струму, що проходить через організм людини, залежить від напруги, під якою перебуває людина й від опору ділянки тіла, до якого прикладена ця напруга. Джерелом живлячої напруги є мережа змінного струму з напругою 229В, на яку поширюється ГОСТ 25861-83.

Для попередження поразок електричним струмом необхідно чітко й у повному обсязі виконувати правила провадження робіт і правил технічної експлуатації. Необхідно виключити можливість доступу працівника до частин устаткування, що працює під небезпечною напругою, до неізольованим частинам, призначеним для роботи при малій напрузі й не підключеним до захисного заземлення, а також підводити електроживлення до ПЕОМ від розетки за допомогою спеціальної вилки із заземлюючим контактом.

3.3.5 Організація робочого місця

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування, ергономічним вимогам, з урахуванням характеру і особливостей трудової діяльності (ДСанПіН 3.3.2.-007-98).

Конструкція робочого місця й взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів дає можливість забезпечувати можливість індивідуального регулювання їх відповідно до потреб працівника для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей, на

відстані близько 70 см, що укладається в припустимі рамки від 60 до 90 см. Частота мерехтіння екрана дорівнює 100 Гц, що відповідає умові більше 70 Гц.

Для зниження нервово-емоційного напруження, стомлювання, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно впроваджувати виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98.

3.4 Пожежна безпека

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежогасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організацією своєчасної евакуації людей.

Всі приміщення повинні бути забезпечені первинними засобами пожежогасіння: пожежним водопостачанням (пожежні крани ПК), пожежні щити з набором пожежного інструменту, вуглекислотними або порошковими вогнегасниками. У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі.

ВИСНОВКИ

В межах дипломного проєкту було створено ігровий застосунок у жанрі раннер-платформер, побудований з використанням сучасних технологій розробки, що відповідає актуальним вимогам ігрового ринку. Дослідження підтвердили, що створення такого застосунку є виправданим для покращення геймплею та надання користувачам нових вражень. Застосування новітніх технологій дозволяє швидко адаптуватися до змін у вподобаннях гравців, оптимізувати ігрові процеси та забезпечувати високий рівень задоволення від гри.

Програмне забезпечення було розроблено за допомогою Unity 3D, який виступає рушієм для створення ігрового середовища, та мови програмування C#, що забезпечила об'єктно-орієнтований підхід до кодування. Використання цих інструментів дозволило створити конкурентоспроможний ігровий продукт.

Створений застосунок включає значний функціонал, що охоплює різні рівні складності, системи досягнень, інтерактивні елементи та численні режими камери, такі як 2D, ізометричний 3D, 3D від третьої особи та 3D від першої особи. Графічний інтерфейс працює коректно на всіх основних екранах, включаючи головний екран, сюжетний екран, екран зі статистикою, екран налаштувань, екран довідки та екран про кінець гри, забезпечуючи захоплюючий ігровий процес.

Пояснювальна записка охоплює всі аспекти, визначені технічним завданням, зокрема аналіз предметної області, детальний опис використаних технологій та інструментів, а також розрахунок економічної ефективності проєкту. Особливу увагу приділено питанням охорони праці, що забезпечують безпечне середовище для розробників, а також наведено перелік використаних джерел, які підтверджують наукову обґрунтованість та актуальність проєкту.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Flappy Bird. [Веб-сайт]. URL: https://flappybird.io/.
2. Subway Surfers. [Веб-сайт]. URL: https://play.google.com/store/apps/details?id=com.kiloo.subwaysurf&hl=en/.
3. Temple Run. [Веб-сайт]. URL: https://play.google.com/store/apps/details?id=com.imangi.templerun&hl=en/.
4. GameForScrath. [Веб-сайт]. URL: https://gamefromscratch.com/.
5. hackr.io. [Веб-сайт]. URL: https://hackr.io/.
6. Unity 3D Docs. [Веб-сайт]. URL: https://docs.unity3d.com/Manual/index.html/.
7. C# Docs. [Веб-сайт]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/.
8. М. В. Добролюбова. Процедурна складова мови C# (навч. посібник). – «КПІ Ігоря Сікорського», 2021.
9. Huw Collingbourne. The Little Book Of C# Programming. – «Paperback», 2019 (англ. мовою).

ДОДАТОК А. Програмний код основної логіки ігрового застосунку

```
// GameKernel.cs

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameKernel : MonoBehaviour
{
    public PlayerLogic playerLogic;
    public SpawnerLogic spawnerLogic;
    public ThemeController themeController;
    public StatsController statsController;
    public SettingsController settingsController;

    public GameObject GameStartCanvas;
    public GameObject GameOverCanvas;
    public GameObject StoryCanvas;
    public GameObject StatsCanvas;
    public GameObject SettingsCanvas;
    public GameObject AboutCanvas;
    public GameObject HUDCanvas;

    public Button ReviewGameOverCanvas;

    public bool isGameOver = false;
    public bool isReview = false;

    public void Start()
    {
        settingsController.DifficultToggleGroupChanged();
        settingsController.GamemodeToggleGroupChanged();
        settingsController.EnvironmentToggleGroupChanged();
        settingsController.PlayerToggleGroupChanged();
        settingsController.SoundToggleGroupChanged();
        settingsController.VibrateToggleGroupChanged();

        Time.timeScale = 0;

        GameStartCanvas.SetActive(true);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(false);
    }

    public void GameOverState()
    {
        isGameOver = true;
        isReview = false;

        Time.timeScale = 0;

        if (statsController.coin == true)
```

```
    {
        ReviewGameOverCanvas.interactable = true;
    }
    else
    {
        ReviewGameOverCanvas.interactable = false;
        themeController.BloodEffect();
    }

    statsController.SetStats();
    GameStartCanvas.SetActive(false);
    GameOverCanvas.SetActive(true);
    StoryCanvas.SetActive(false);
    StatsCanvas.SetActive(false);
    SettingsCanvas.SetActive(false);
    AboutCanvas.SetActive(false);
    HUDCanvas.SetActive(false);
}

public void StartGameState()
{
    Time.timeScale = 1;

    GameStartCanvas.SetActive(false);
    GameOverCanvas.SetActive(false);
    StoryCanvas.SetActive(false);
    StatsCanvas.SetActive(false);
    SettingsCanvas.SetActive(false);
    AboutCanvas.SetActive(false);
    HUDCanvas.SetActive(true);
}

public void ReviewState()
{
    isReview = true;
    isGameOver = false;

    Time.timeScale = 1;

    GameStartCanvas.SetActive(false);
    GameOverCanvas.SetActive(false);
    StoryCanvas.SetActive(false);
    StatsCanvas.SetActive(false);
    SettingsCanvas.SetActive(false);
    AboutCanvas.SetActive(false);
    HUDCanvas.SetActive(true);
}

public void SettingsState()
{
    GameStartCanvas.SetActive(false);
    GameOverCanvas.SetActive(false);
    StoryCanvas.SetActive(false);
    StatsCanvas.SetActive(false);
    SettingsCanvas.SetActive(true);
    AboutCanvas.SetActive(false);
    HUDCanvas.SetActive(false);
}

public void AboutState()
{
    GameStartCanvas.SetActive(false);
```

```
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(true);
        HUDCanvas.SetActive(false);
    }

    public void StoryState()
    {
        GameStartCanvas.SetActive(false);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(true);
        StatsCanvas.SetActive(false);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(false);
    }

    public void StatsState()
    {
        statsController.SetStats();

        GameStartCanvas.SetActive(false);
        GameOverCanvas.SetActive(false);
        StoryCanvas.SetActive(false);
        StatsCanvas.SetActive(true);
        SettingsCanvas.SetActive(false);
        AboutCanvas.SetActive(false);
        HUDCanvas.SetActive(false);
    }

    public void MainMenuState()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    public void ExitState()
    {
        Application.Quit();
    }
    public void PauseResumeState()
    {
        if (Time.timeScale == 1)
        {
            Time.timeScale = 0;
        }
        else
        {
            Time.timeScale = 1;
        }
    }

}

// SpawnerLogic.cs

using UnityEngine;

public class SpawnerLogic : MonoBehaviour
{
    public float range;
```

```
    public float start;
    public float period;
    public float height;
    public int probability;

    public GameObject target;

    public void Start()
    {
        InvokeRepeating("Spawn", start, period);
    }

    void Spawn()
    {
        range += 5f;

        if (Random.Range(0, 100) < probability)
        {
            GameObject newTarget = Instantiate(target);
            newTarget.transform.position = transform.position + new Vector3(range,
UnityEngine.Random.Range(-height, height), Random.Range(-0.5f, 0.5f));
        }
    }

    void OnBecameInvisible()
    {
        Destroy(this.gameObject);
    }
}

// PlayerLogic.cs

using UnityEngine;

public class PlayerLogic : MonoBehaviour
{
    public CharacterController controller;
    public Vector3 playerVelocity;
    public Quaternion downRotation;
    public Quaternion forwardRotation;
    public AudioSource source;

    public bool groundedPlayer;
    public float playerSpeed;
    public float jumpHeight = 1.0f;
    public float gravityValue = -9.81f;

    public GameKernel gameKernel;
    public StatsController statsController;

    public AudioClip wingSound;
    public AudioClip crashSound;
    public AudioClip scoreSound;

    public bool vibrate;

    public float timeRemaining = 5f;

    private void Start()
    {
        controller = GetComponent<CharacterController>();
        forwardRotation = Quaternion.Euler(0, 0, 30);
```

```
        downRotation = Quaternion.Euler(0, 0, -90);
    }

    void Update()
    {
        if (Time.timeScale > 0)
        {
            groundedPlayer = controller.isGrounded;

            if (groundedPlayer && playerVelocity.y < 0)
            {
                playerVelocity.y = 0f;
            }

            if (Input.GetMouseButtonDown(0))
            {
                if (vibrate == true)
                {
                    //Handheld.Vibrate(); for mobile
                }
                source.PlayOneShot(wingSound);
                transform.rotation = forwardRotation;
                playerVelocity.y += Mathf.Sqrt(jumpHeight * -3.0f * gravityValue);
            }
            else
            {
                playerVelocity.x = playerSpeed;
                playerVelocity.y += gravityValue * Time.deltaTime;
                controller.Move(playerVelocity * Time.deltaTime);
                transform.rotation = Quaternion.Lerp(transform.rotation, downRotation,
playerSpeed * Time.deltaTime);
            }
        }

        if (gameKernel.isReview == true)
        {
            startTimer();
        }
    }

    public void OnBecameInvisible()
    {
        source.PlayOneShot(crashSound);
        gameKernel.GameOverState();
    }

    void startTimer()
    {
        if (timeRemaining > 0)
        {
            timeRemaining -= Time.deltaTime;
        }
        else if (timeRemaining <= 0)
        {
            timeRemaining = 5;
            gameKernel.isReview = false;
            statsController.coin = false;
        }

    }

    public void OnControllerColliderHit(ControllerColliderHit hit)
```

```
    {
        if (gameKernel.isReview == false)
        {
            source.PlayOneShot(crashSound);
            gameKernel.GameOverState();
        }
    }

    public void OnTriggerEnter(Collider collider)
    {
        if (collider.gameObject.tag == ("Obscale"))
        {
            statsController.sessionScore += 1;
            source.PlayOneShot(scoreSound);
        }

        if (collider.gameObject.tag == ("Enemy"))
        {
            Destroy(collider.gameObject);
        }

        if (collider.gameObject.tag == ("Coin"))
        {
            Destroy(collider.gameObject);
            statsController.coin = true;
        }
    }

}

// CameraLogic.cs

using UnityEngine;

public class CameraLogic : MonoBehaviour
{
    public GameObject target;
    public GameKernel gameKernel;

    public Vector3 offsetPosition = new Vector3(0f, 4f, -4f);
    public Quaternion offsetRotation = Quaternion.Euler(0, 0, 0);
    public Space offsetPositionSpace = Space.World;

    public float shakeDuration = 1f;
    public float shakeAmount = 0.7f;
    public float decreaseFactor = 1.0f;
    Vector3 originalPosition;

    private void LateUpdate()
    {
        Refresh();
    }

    void OnEnable()
    {
        originalPosition = target.transform.position;
    }

    public void Refresh()
    {
        if (offsetPositionSpace == Space.Self)
        {
```

```
            transform.position = target.transform.TransformPoint(offsetPosition);
            transform.rotation = offsetRotation;
        }
        else
        {
            transform.position = target.transform.position + offsetPosition;
            transform.rotation = offsetRotation;
        }
    }
}

// SettingsController.cs

using UnityEngine;
using UnityEngine.UI;

public class SettingsController : MonoBehaviour
{
    public Toggle EasyToggle;
    public Toggle MediumToggle;
    public Toggle HardToggle;

    public Toggle TwoDToggle;
    public Toggle IsometricToggle;
    public Toggle ThreeDFirstPersonToggle;
    public Toggle ThreeDThirdPersonToggle;

    public Toggle DayToggle;
    public Toggle NightToggle;

    public Toggle DefaultToggle;
    public Toggle RainbowToggle;

    public Toggle SoundToggle;
    public Toggle VibrateToggle;

    public Toggle FPSToggle;
    public Toggle GraphicsToggle;

    public CameraLogic cameraLogic;
    public DifficultController difficultController;
    public ThemeController themeController;
    public GraphicsController graphicsController;

    public AudioSource audioSource;

    private void Start()
    {
        if (PlayerPrefs.HasKey("Gamemode"))
        {
            if (PlayerPrefs.GetInt("Gamemode") == 0)
            {
                TwoDToggle.isOn = true;
            }
            if (PlayerPrefs.GetInt("Gamemode") == 1)
            {
                IsometricToggle.isOn = true;
            }
            if (PlayerPrefs.GetInt("Gamemode") == 2)
            {
                ThreeDFirstPersonToggle.isOn = true;
            }
```

```
    if (PlayerPrefs.GetInt("Gamemode") == 3)
    {
        ThreeDThirdPersonToggle.isOn = true;
    }
}

if (PlayerPrefs.HasKey("Difficult"))
{
    if (PlayerPrefs.GetInt("Difficult") == 0)
    {
        EasyToggle.isOn = true;
    }
    if (PlayerPrefs.GetInt("Difficult") == 1)
    {
        MediumToggle.isOn = true;
    }
    if (PlayerPrefs.GetInt("Difficult") == 2)
    {
        HardToggle.isOn = true;
    }
}

if (PlayerPrefs.HasKey("Environment"))
{
    if (PlayerPrefs.GetInt("Environment") == 0)
    {
        DayToggle.isOn = true;
    }
    if (PlayerPrefs.GetInt("Environment") == 1)
    {
        NightToggle.isOn = true;
    }
}

if (PlayerPrefs.HasKey("Player"))
{
    if (PlayerPrefs.GetInt("Player") == 0)
    {
        DefaultToggle.isOn = true;
    }
    if (PlayerPrefs.GetInt("Player") == 1)
    {
        RainbowToggle.isOn = true;
    }
}

if (PlayerPrefs.HasKey("Graphic"))
{
    if (PlayerPrefs.GetInt("Graphic") == 0)
    {
        GraphicsToggle.isOn = true;
    }
    if (PlayerPrefs.GetInt("Graphic") == 1)
    {
        GraphicsToggle.isOn = false;
    }
}

if (PlayerPrefs.HasKey("FPS"))
{
    if (PlayerPrefs.GetInt("FPS") == 0)
    {
```

```
            FPSToggle.isOn = true;
        }
        if (PlayerPrefs.GetInt("FPS") == 1)
        {
            FPSToggle.isOn = false;
        }
    }

    if (PlayerPrefs.HasKey("Sound"))
    {
        if (PlayerPrefs.GetInt("Sound") == 0)
        {
            SoundToggle.isOn = true;
        }
        if (PlayerPrefs.GetInt("Sound") == 1)
        {
            SoundToggle.isOn = false;
        }
    }

    if (PlayerPrefs.HasKey("Vibrate"))
    {
        if (PlayerPrefs.GetInt("Vibrate") == 0)
        {
            VibrateToggle.isOn = true;
        }
        if (PlayerPrefs.GetInt("Vibrate") == 1)
        {
            VibrateToggle.isOn = false;
        }
    }
}

public void GamemodeToggleGroupChanged()
{
    if (TwoDToggle.isOn)
    {
        PlayerPrefs.SetInt("Gamemode", 0);
        PlayerPrefs.Save();

        cameraLogic.offsetPosition = new Vector3(0f, 0f, -10f);
        cameraLogic.offsetRotation = Quaternion.Euler(0, 0, 0);

    }
    if (IsometricToggle.isOn)
    {
        PlayerPrefs.SetInt("Gamemode", 1);
        PlayerPrefs.Save();

        cameraLogic.offsetPosition = new Vector3(-10f, 0f, -10f);
        cameraLogic.offsetRotation = Quaternion.Euler(0, 60, 0);
    }
    if (ThreeDFirstPersonToggle.isOn)
    {
        PlayerPrefs.SetInt("Gamemode", 2);
        PlayerPrefs.Save();

        cameraLogic.offsetPosition = new Vector3(0f, 0f, 0f);
        cameraLogic.offsetRotation = Quaternion.Euler(0, 90, 0);
    }
    if (ThreeDThirdPersonToggle.isOn)
    {
```

```
            PlayerPrefs.SetInt("Gamemode", 3);
            PlayerPrefs.Save();

            cameraLogic.offsetPosition = new Vector3(-10f, 0f, 0f);
            cameraLogic.offsetRotation = Quaternion.Euler(0, 90, 0);
        }
    }

    public void DifficultToggleGroupChanged()
    {
        if (EasyToggle.isOn)
        {
            PlayerPrefs.SetInt("Difficult", 0);
            PlayerPrefs.Save();

            difficultController.EasyMode();
        }
        if (MediumToggle.isOn)
        {
            PlayerPrefs.SetInt("Difficult", 1);
            PlayerPrefs.Save();

            difficultController.MediumMode();
        }
        if (HardToggle.isOn)
        {
            PlayerPrefs.SetInt("Difficult", 2);
            PlayerPrefs.Save();

            difficultController.HardMode();
        }
    }

    public void EnvironmentToggleGroupChanged()
    {
        if (DayToggle.isOn)
        {
            PlayerPrefs.SetInt("Environment", 0);
            PlayerPrefs.Save();

            themeController.DayEnvironment();
        }
        if (NightToggle.isOn)
        {
            PlayerPrefs.SetInt("Environment", 1);
            PlayerPrefs.Save();

            themeController.NightEnvironment();
        }
    }

    public void PlayerToggleGroupChanged()
    {
        if (DefaultToggle.isOn)
        {
            PlayerPrefs.SetInt("Player", 0);
            PlayerPrefs.Save();

            themeController.DefaultPlayer();
        }
        if (RainbowToggle.isOn)
        {
```

```
        PlayerPrefs.SetInt("Player", 1);
        PlayerPrefs.Save();

        themeController.RainbowPlayer();
    }
}

public void GraphicsToggleGroupChanged()
{
    if (GraphicsToggle.isOn == true)
    {
        PlayerPrefs.SetInt("Graphics", 0);
        PlayerPrefs.Save();

        graphicsController.GraphicsHigh();
    }
    if (GraphicsToggle.isOn == false)
    {
        PlayerPrefs.SetInt("Graphics", 1);
        PlayerPrefs.Save();

        graphicsController.GraphicsLow();
    }
}

public void FPSToggleGroupChanged()
{
    if (FPSToggle.isOn == true)
    {
        PlayerPrefs.SetInt("FPS", 0);
        PlayerPrefs.Save();

        graphicsController.FPS60();
    }
    if (FPSToggle.isOn == false)
    {
        PlayerPrefs.SetInt("FPS", 1);
        PlayerPrefs.Save();

        graphicsController.FPS30();
    }
}

public void SoundToggleGroupChanged()
{
    if (SoundToggle.isOn == true)
    {
        PlayerPrefs.SetInt("Sound", 0);
        PlayerPrefs.Save();

        audioSource.enabled = true;
    }
    if (SoundToggle.isOn == false)
    {
        PlayerPrefs.SetInt("Sound", 1);
        PlayerPrefs.Save();

        audioSource.enabled = false;
    }
}

public void VibrateToggleGroupChanged()
```

```
    {
        if (VibrateToggle.isOn == true)
        {
            PlayerPrefs.SetInt("Vibrate", 0);
            PlayerPrefs.Save();

        }
        if (VibrateToggle.isOn == false)
        {
            PlayerPrefs.SetInt("Vibrate", 1);
            PlayerPrefs.Save();

        }
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка ігрового застосунку у жанрі раннер-платформер

ЧУЛАКОВ ВЛАДИСЛАВ ОЛЕГОВИЧ

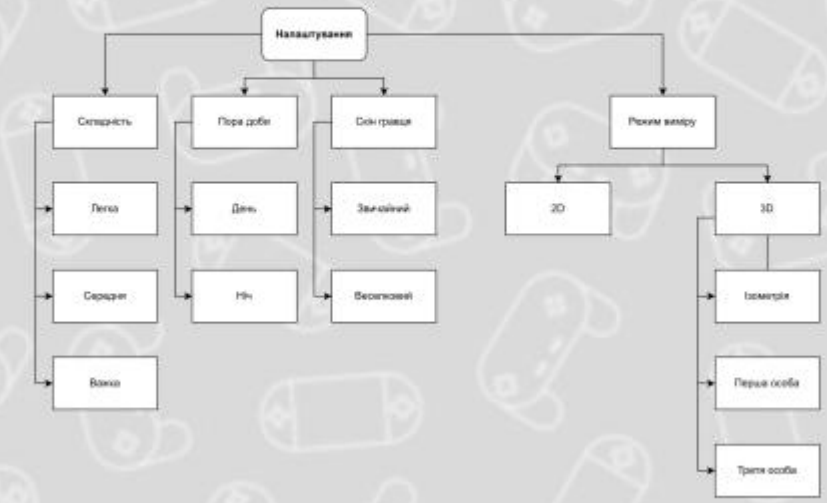
Загальні відомості

- Жанр раннер-платформер популярний завдяки своїй динамічності та викликам.
- Нова гра на Unity 3D з використанням C# включатиме різні рівні складності, системи досягнень та режими камери для захоплюючого геймплею.
- Вона має стати яскравим доповненням до жанру, привертаючи шанувальників цього жанру.

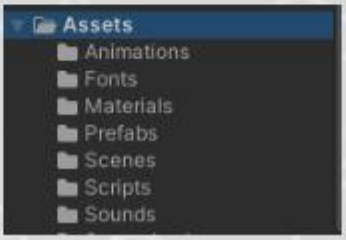
Стек технологій

Навігація ігрового застосунку

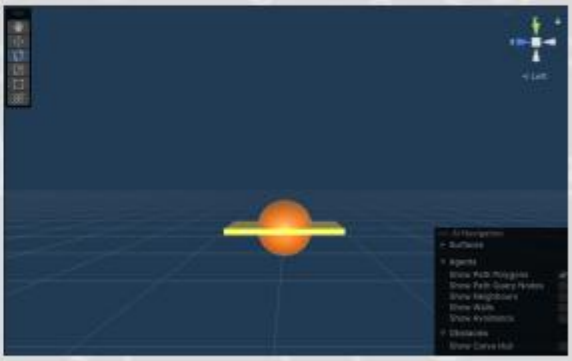
Налаштування ігрового застосунку



Архітектура ігрового застосунку

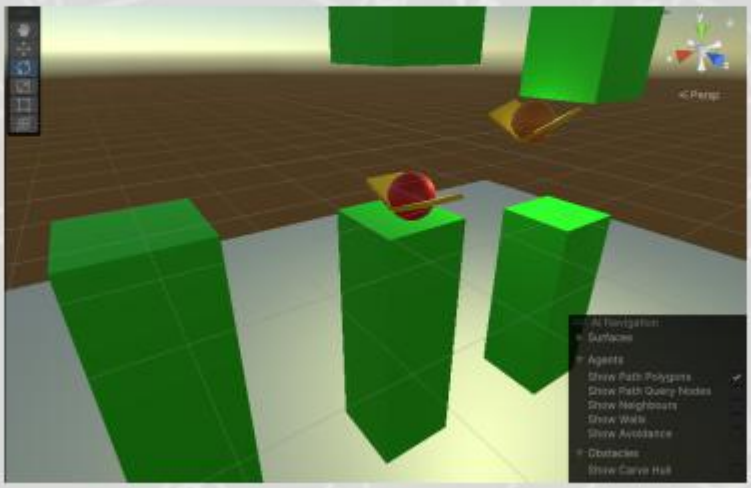


Створення головного персонажу



Створені префаби та матеріали

Створення дизайну рівня

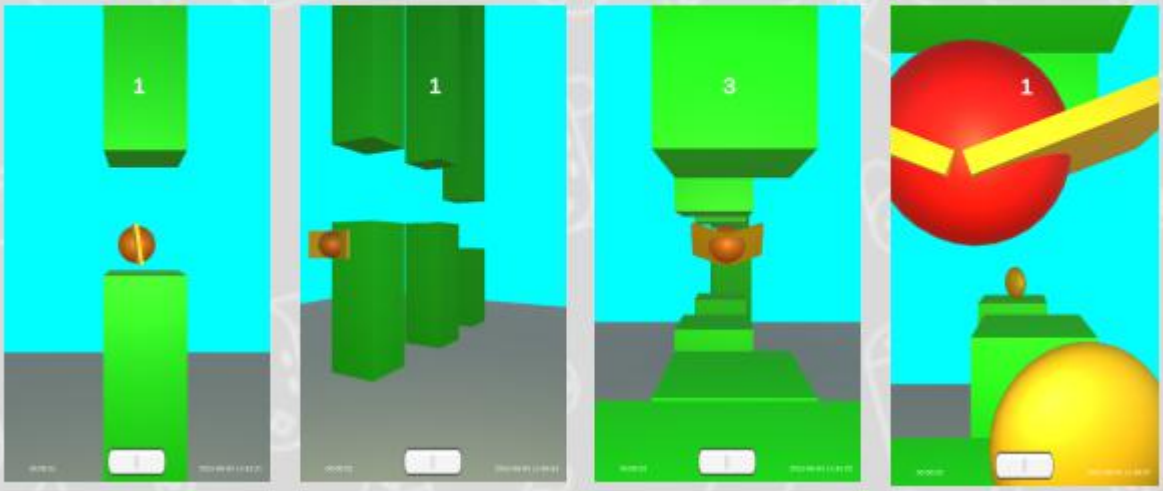


Програмний код генерації рівню

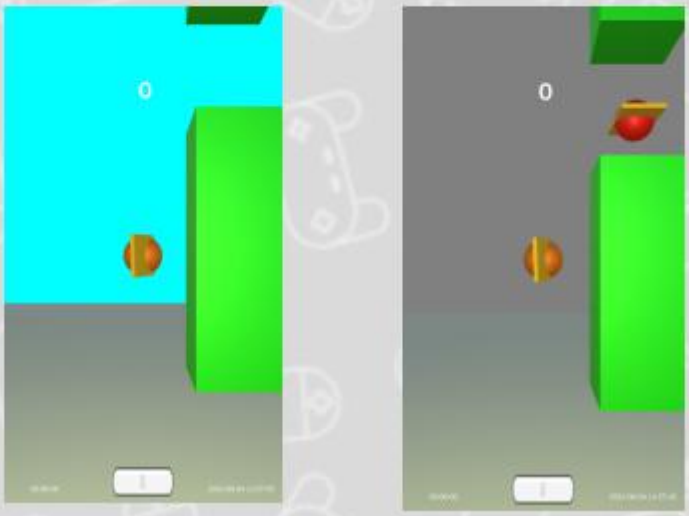
```
using UnityEngine;

public class SpawnerLogic : MonoBehaviour
{
    public float range;
    public float start;
    public float period;
    public float height;
    public int probability;

    public GameObject target;

    public void Start()
    {
        InvokeRepeating("Spawn", start, period);
    }

    void Spawn()
    {
        range += 5f;

        if (Random.Range(0, 100) < probability)
        {
            GameObject newTarget = Instantiate(target);
            newTarget.transform.position = transform.position
+ new Vector3(range, UnityEngine.Random.Range(-height,
height), Random.Range(-0.5f, 0.5f));
        }
    }

    void OnBecameInvisible()
    {
        Destroy(this.gameObject);
    }
}
```

Графічний інтерфейс ігрового застосунку

Геймплей з різним кутом камери



Геймплей з різною порою доби



Дякую за увагу

Готов відповісти на Ваші запитання

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Чулакову Владиславу Олеговичу

(прізвище, ім`я та по батькові)

Спеціальність: *121 «Інженерія програмного забезпечення» ;*

Освітня програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка ігрового застосунку у жанрі раннер-платформер*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 82 сторінки. У пояснювальній записці описано етапи розробки ігрового застосунку у жанрі раннер-платформер засобами рушію Unity. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Чулаков Владислав поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Чулаков Владислав під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Ніколаєв Валентин показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS, Unity 3D, MS PowerPoint, MS Visio та ін.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Іванова Лілія Вікторівна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Чулакова Владислава Олеговича

(прізвище, ім'я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Іванова Лілія Вікторівна*

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) *Розробка ігрового застосунку у жанрі раннер-платформер*

Обсяг розрахунково-пояснювальної записки *82* сторінок

Обсяг графічної (презентаційної) частини *14* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі створення ігор у жанрі раннер-платформер та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 14 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скриншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано ігровий застосунок у жанрі раннер-платформер.*

Ігровий застосунок містить налаштування складності, оточення, персонажу та куту камери. Крім того, також наявне збереження прогресу.

д) основні недоліки дипломного проекту

1. У якості ігрових об'єктів використовуються стандартні моделі примітивів, що надаються засобами ігрового рушію;

2. У роботі не представлено блок-схем алгоритмів для створених скриптів;

3. Наявні деякі помилки оформлення тексту пояснювальної записки

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *Василіу Євген Вікторович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ*

Підпис

«18» 06 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016336493

Дата перевірки:
08.06.2024 22:07:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 22:26:08 EEST

ID користувача:
100011688

Назва документа: 4РП-07_Чулаков_В

Кількість сторінок: 46 Кількість слів: 5452 Кількість символів: 42268 Розмір файлу: 1.64 MB ID файлу: 1016137265

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.34%
Схожість

Найбільша схожість: 0.4% з Інтернет-джерелом (https://discussions.unity.com/t/how-do-i-respawn-after-falling-below-a...

1.34% Джерела з Інтернету | 104 ..... Сторінка 48

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 7 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Чулаков Владислав Сергійович,
здобувач освіти гр. 4РП-07, та

Іванова Лілія Вікторівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка ігрового застосунку у жанрі раннер-платформер» (автор роботи – Чулаков В.О., керівник роботи – Іванова Л.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Чулаков В.О. /

Керівник _____ / Іванова Л.В. /

«10» червня 2024 р.