МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»[1]
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: «**Розроблення комп'ютерної гри жанру платформер на основі рушія Unity**»

здобувача: Зденика Максима Володимировича

Науковий керівник:
д.т.н., професор
Шевченко К.Л

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
Тішков Б.О.

Київ 2025

[1]

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:
Керівник проєктної групи(гарант)
освітньо-професійної програми
Помазун О.М.
"____" _____2025 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри
Тішков Б.О.
"____" _____2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти **Зденика Максима Володимировича**

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: " Розроблення комп'ютерної гри жанру платформер на основі рушія Unity "

Тему затверджено наказом ректора Університету від « *7* » *березня 2025* р.
№ *466- ст*.

Кваліфікаційна бакалаврська робота виконується на матеріалах

План кваліфікаційної бакалаврської роботи

Розділ І
Характеристика та аналіз предметної галузі.
Розділ ІІ
Розробка вимог та моделювання гри.
Розділ ІІІ
Проектування та реалізація компонентів гри платформера.
Об'єкт дослідження
Процес побудови інтерактивного ігрового середовища у середовищі Unity з використанням мови C#.

Предмет дослідження

Сукупність методів, технологій та інструментів, що використовуються для проєктування, розробки та реалізації комп'ютерної гри у жанрі 3D платформера **.**

Мета кваліфікаційної бакалаврської роботи

Розробка прототипу тривимірної low poly гри у жанрі платформер, що включає базові механіки геймплею, візуальне оформлення та інтерактивний інтерфейс користувача. Проєкт передбачає реалізацію функціонального ігрового середовища на основі рушія Unity, із залученням зовнішніх 3D-інструментів, зокрема Blender, для створення моделей, анімацій і сцен.

У розділі I

Був проведений порівняльний аналіз найпопулярніших ігор жанру платформер і був обраний найкращий функіонал.

У розділі II

Була наведений аналіз та специфікація вимог гри. Визначені бізнес, функціональні та нефункціональні вимоги. Була розроблена інформаційна модель. Було наведено моделювання інформаційної підсистеми та розроблені діаграми класів, прецедентів та діаграма послідовності.

У розділі III

Тут була наведена технічна частина створення гри мовою програмування C#. Розписана реалізація колайдерів та генерація сегментів. Була описана система руху персонажу. Була наведена реалізація геймплейного UI. Та була наведена розробка бази даних через Json формат.

Завдання підготував
науковий керівник _____ Шевченко К.Л.
« 10 » березня 2025 р.

Завдання одержав
здобувач _____ Зденик М.В.
« 10 » березня 2025 р..

Відгук
про кваліфікаційну бакалаврську роботу
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

на тему
«Розроблення комп'ютерної гри жанру платформер на основі рушія Unity»

1. Актуальність теми: враховуючі стрімкий розвиток ігрової індустрії та зростання попиту ігрові продукти, де ігрові персонажі адаптивно реагують на дії гравця, створючі ефект живого середовища, тема роботи є актуальною
2. Позитивні риси кваліфікаційної бакалаврської роботи: в роботі проведено аналіз літературних джерел, сформульовано вимоги та основні задачі дослідження, розроблено інформаційне та програмне забезпечення, визначено структуру технічного забезпечення
3. Наявність самостійних розробок автора: автором самостійно виконані всі поставлені в індивідуальному завданні задачі
4. Цінність теоретичних висновків та практичних рекомендацій: на основі моделі адаптивної поведінки створено працездатний ігровий прототип, який може бути використаний як основа для подальших розробок освітніх або ігрових проєктів
5. Наявність недоліків: в роботі присутні відхилення у форматуванні тексту, порушені вимоги у розміщенні назв таблиць та рисунків, посилання на використані джерела зустрічаються лише епізодично.
6. Загальна оцінка кваліфікаційної бакалаврської роботи та її допущення до захисту перед ЕК: кваліфікаційна бакалаврська робота на тему «Розроблення комп'ютерної гри жанру платформер на основі рушія Unity» відповідає встановленим вимогам та рекомендується до захисту, а її автор, Зденик Максим Володимирович заслуговує на присвоєння освітньо-кваліфікаційного рівня «бакалавр» за спеціальністю 122 «Комп'ютерні науки»

Науковий керівник

_____ д.т.н., професор Шевченко К.Л.
(підпис)

"___"_____ 2025р.

Рецензія
на кваліфікаційну бакалаврську роботу
здобувача вищої освіти

_____

(прізвище, ім'я, по батькові)

Тема _____

Актуальність теми кваліфікаційної бакалаврської роботи і доцільність її розроблення:
ринок комп'ютерних ігрових продуктів останніми роками є одним з таких, що має найбільшу рентабельність. Особливим попитом користуються продукти, у яких ігрові персонажі адаптивно реагують на дії гравця. Тому тему представленої роботи можна вважати актуальною.
Якість проведеного дослідження:
робота містить ґрунтовний аналіз літературних джерел, аргументований вибір архітектурного та програмного забезпечення, все це виконано з достатньою якістю.
Позитивні риси кваліфікаційної бакалаврської роботи:
всю роботу, від постановки задачі до практичної реалізації, побудовано логічно. Кінцевим результатом є створення робочого прототипу.
Зауваження:
до зауважень слід віднести наявність великої кількості помилок, некоректне розташування назв таблиць. Як правило, в технічній документації вони вирівнюються по правому краю сторінки.
Практична значимість висновків і рекомендації:
автором створено працездатний ігровий прототип, який може бути використаний для подальших розробок освітніх або ігрових застосунків.
Вважаю, що кваліфікаційна бакалаврська робота на тему **«Розроблення комп'ютерної гри жанру платформер на основі рушія Unity»** відповідає встановленим вимогам та заслуговує позитивної оцінки, а її автор, Зденюк Максим Володимирович, заслуговує на присвоєння освітньо-кваліфікаційного рівня **«бакалавр»** за спеціальністю **122 «Комп'ютерні науки»**.

Місце роботи та посада рецензента
Науковий ступінь, учене звання (*за наявності*) :
Директор НДІ автоматизації експериментальних досліджень, к.т.н., доцент **Омельченко Ю.М**
Підпис засвідчую: _____

(посада, підпис)

Місце печатки організації, де працює рецензент

АНОТАЦІЯ

У кваліфікаційній роботі представлено повний цикл розробки 3D гри у жанрі платформера на основі рушія Unity. Проєкт охоплює теоретичний, аналітичний та практичний аспекти створення інтерактивного ігрового продукту, реалізованого мовою C# із використанням сучасних інструментів моделювання та розробки.

У першому розділі здійснено аналіз сучасних ігрових рушіїв, визначено жанрові особливості платформерів та обґрунтовано вибір технологій для реалізації проєкту. Проведено порівняльну характеристику функціоналу найбільш популярних систем розробки.

Другий розділ присвячено визначенню та специфікації вимог до ігрової системи. Розглянуто бізнес-вимоги, функціональні та нефункціональні критерії, а також створено відповідні діаграми класів, прецедентів, послідовностей та інформаційну модель, що відображає архітектуру гри.

У третьому розділі подано технічну реалізацію проєкту. Деталізовано механізм створення ігрових локацій, керування персонажем, генерації сегментів, обробки колайдерів та побудови UI-системи. Особливу увагу приділено створенню бази даних та збереженню прогресу за допомогою JSON-формату. Проєкт протестовано і адаптовано для подальшого розширення.

ABSTRACT

This bachelor's project presents the full development cycle of a 3D platformer game based on the Unity engine. The project covers theoretical, analytical, and practical aspects of creating an interactive game product implemented in C# using modern development and modeling tools.

The first section provides an analysis of current game engines, defines the genre characteristics of platformers, and justifies the choice of technologies for project implementation. A comparative overview of the most popular game development platforms and their functionality is conducted.

The second section focuses on defining and specifying the system requirements. It outlines business, functional, and non-functional requirements, and presents class diagrams, use case diagrams, sequence diagrams, and the information model representing the game's architecture.

The third section describes the technical implementation of the project. It details the development of game levels, character movement mechanics, segment generation, collider handling, and UI construction. Special attention is given to database creation and progress saving using the JSON format. The project was tested and is ready for future scalability.

РЕФЕРАТ

Бакалаврський проєкт складається з 67 сторінок, включає 5 таблиць, 36 ілюстрацій, список використаних джерел з 30 найменувань та додатки.

Темою роботи є створення 3D гри у жанрі платформера із застосуванням інструментарію Unity.

Ключові слова: гра, Unity, платформер, персонаж, геймдизайн, C#, Blender.

Об’єктом дослідження є процес побудови інтерактивного ігрового середовища у середовищі Unity з використанням мови C#. У ролі предметної області виступає технологія створення тривимірних ігор з акцентом на динаміку, перешкоди та взаємодію з об'єктами сцени.

Предмет дослідження — це сукупність методів, технологій та інструментів, що використовуються для проєктування, розробки та реалізації комп’ютерної гри у жанрі 3D платформера, зокрема механіки управління персонажем, генерація рівнів, обробка зіткнень, інтерфейс взаємодії з користувачем (UI), а також способи оптимізації продуктивності в середовищі Unity.

Метою кваліфікаційного проєкту є створення власної ігрової системи, що реалізує базові механіки руху, взаємодії з елементами середовища, збирання ігрових об'єктів, а також відображення інформації через UI. Проєкт дозволив не лише на практиці опанувати сучасні підходи до геймрозробки, а й проявити себе як фахівця з дизайну інтерактивних програмних систем.

До складу використаних інструментів входять: Unity, C#, Blender, Adobe Photoshop, Visual Studio.

Технічна реалізація здійснювалась на персональному комп’ютері, що забезпечував необхідну продуктивність для тестування й налагодження проєкту.

В результаті виконано повноцінну реалізацію гри, розроблено власний ігровий рівень, реалізовано механіки руху, взаємодії з перешкодами, систему збирання монет, а також розроблено інтерфейс для відображення результатів

гравця. Проведене тестування показало, що гра функціонує стабільно, а її механіки — працездатні.

Створений програмний продукт може використовуватись як інструмент для навчання, відпочинку або розвитку моторики та просторового мислення у гравців різного віку.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми.

Вибір теми дипломної роботи — Розроблення комп’ютерної гри жанру платформер на основі рушія Unity — є свідомим та обґрунтованим з кількох причин. По-перше, ігрова індустрія продовжує демонструвати стабільне зростання, що створює високий попит на спеціалістів у сфері розробки ігор, зокрема 3D-продуктів. Саме тривимірні ігри дають змогу створювати глибокі й реалістичні віртуальні світи, що відповідає сучасним очікуванням користувачів.

По-друге, рушій Unity є одним із найпотужніших та найзручніших інструментів для створення тривимірних ігор Його перевагами є кросплатформеність, велика база знань і підтримка спільноти, широкий набір вбудованих засобів для фізики, анімації, обробки подій та взаємодії з користувачем. Це дає змогу зосередитись не лише на технічній реалізації, а й на творчому підході до побудови ігрового процесу.

Крім того, дана тема дозволяє мені поєднати теоретичні знання, здобуті під час навчання (програмування, об'єктно-орієнтований підхід, алгоритми, комп’ютерна графіка) з їх практичним застосуванням у реальному програмному проєкті. Це особливо важливо для підготовки до подальшої професійної діяльності у сфері розробки програмного забезпечення або інтерактивних застосунків.

Таким чином, вибір теми обумовлений як особистою зацікавленістю, так і актуальністю напряму, технологічною доцільністю обраного інструментарію, а також прагненням здобути практичний досвід повного циклу створення 3D-ігрового проєкту — від моделювання сцени до реалізації логіки та взаємодії з гравцем.

Актуальність обраної теми дипломного проєкту визначається зростаючою роллю інтерактивних цифрових технологій у сучасному суспільстві, а також стабільним розвитком індустрії відеоігор як у комерційному, так і в освітньому та науковому середовищах. Сьогодні 3D-ігри виконують не лише розважальну

функцію, а й застосовуються в архітектурних візуалізаціях, тренажерах, медичних симуляціях, освітніх платформах і навіть у психологічній діагностиці.

Зростання попиту на інтерактивні продукти потребує підготовки фахівців, здатних створювати якісні, оптимізовані та кросплатформенні 3D-застосунки. Тому дослідження, спрямовані на розробку повноцінного ігрового продукту з використанням сучасного рушія, мають як теоретичну, так і прикладну цінність. Unity як інструмент розробки 3D-ігор забезпечує розширені можливості для реалізації складної логіки, інтерактивності, фізики та візуалізації, що робить його актуальним для дослідження в умовах навчального проєкту.

З наукової точки зору, проєкт поєднує в собі елементи комп’ютерної графіки, програмної інженерії, системного проєктування та взаємодії людини з комп’ютером. Практична значущість дипломного проєкту полягає в отриманні нового досвіду створення функціонального 3D-застосунку, узагальненні сучасних підходів до розробки ігор, а також у вивченні методів реалізації ігрової логіки в умовах конкретної програмної платформи.

Можемо зробити висновок, що дипломний проєкт є актуальним як з точки зору розвитку особистих професійних компетентностей, так і з огляду на потреби ринку IT-послуг, де розробники 3D-контенту мають значний попит. Проєкт сприяє практичному засвоєнню знань з інженерії програмного забезпечення, що є необхідною умовою формування кваліфікованого спеціаліста.

Теоретична значущість роботи полягає в систематизації знань щодо проєктування тривимірних ігрових застосунків, аналізі архітектури сучасних ігрових рушіїв, зокрема Unity, а також у дослідженні основних принципів побудови інтерактивних середовищ. Вона також охоплює вивчення алгоритмів керування об'єктами, моделювання поведінки користувача та побудови систем взаємодії в реальному часі.

Практична значущість полягає в створенні повноцінного прототипу 3D-гри, який можна застосовувати як демонстраційний або навчальний продукт. Отримані результати можуть бути використані в подальшій розробці більш складних ігрових або навчальних систем, а також у процесі професійної підготовки розробників .

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

UI (User Interface) – інтерфейс користувача, що забезпечує візуальну взаємодію гравця з грою. Включає такі елементи, як кнопки, лічильники, текстові поля та панелі.

HUD (Heads-Up Display) – постійно видимі на екрані графічні елементи, які відображають ключову інформацію для гравця (кількість монет, статус персонажа тощо).

3D – тривимірне представлення об’єктів у віртуальному просторі, з координатами по осях X, Y та Z.

FPS (Frames Per Second) – кількість кадрів на секунду; показник, що характеризує продуктивність гри на конкретному пристрої.

LOD (Level of Detail) – рівень деталізації об’єкта, який змінюється залежно від відстані до камери з метою оптимізації продуктивності.

Геймпле́й, **ґеймплей** — термін, яким називають особливості взаємодії людини з відеогрою.

РОЗДІЛ 1.
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

У сучасному світі ігрова індустрія посідає одне з провідних місць серед форм цифрового мистецтва, програмної інженерії та засобів масової комунікації. Розробка комп’ютерних ігор вже давно вийшла за межі розваги — сьогодні вона виступає повноцінною частиною інформаційної сфери, в якій поєднуються технології моделювання, геймдизайну, штучного інтелекту, а також психології сприйняття та взаємодії людини з цифровим середовищем.

Предметна галузь, у межах якої виконується ця дипломна робота, охоплює розробку інтерактивних візуальних застосунків — зокрема, тривимірних комп’ютерних ігор. Вона поєднує знання з кількох напрямів: програмування, комп’ютерної графіки, фізики руху, математичного моделювання, а також користувацького досвіду (UX) та дизайну інтерфейсів (UI). Ігрові проєкти потребують комплексного підходу, який включає як розробку функціонального коду, так і створення художньо продуманого візуального контенту, збалансованої механіки та логіки взаємодії.

Жанрово даний проєкт належить до категорії платформерів — це підвид аркадних ігор, де основний ігровий процес полягає у переміщенні персонажа по рівнях, що включають фізичні перешкоди, платформи, об’єкти для збирання та певні умови для завершення етапу. Платформери мають просту, але водночас глибоку механіку, яка добре підходить для навчальних і проєктних задач, оскільки дозволяє реалізувати базові принципи фізики, анімації та сценарного контролю.

Об’єктом дослідження у межах цієї роботи виступає процес створення 3D-гри в жанрі платформера, який охоплює повний цикл проєктування — від розробки ідеї та структури рівнів до технічної реалізації механік, графічного оформлення та програмування логіки гри у середовищі Unity. Також об’єктом уваги є застосування low poly стилістики для оптимізації візуальної складової без втрати естетичної

цілісності, що є важливим чинником при створенні ігор для широкої аудиторії та пристроїв із різним рівнем продуктивності.

На мою думку, важливим є як сам підхід до організації ігрового простору, так і реалізація взаємодії між гравцем та віртуальним світом, що досягається через поєднання технічних і творчих інструментів. Аналіз, проєктування та створення такої гри дозволяє сформувати практичні навички програмування, логічного моделювання, інтеграції візуального контенту, а також навички оптимізації, тестування та оформлення повноцінного програмного продукту у вигляді ігрової програми.

Вибір жанру для комп'ютерної гри є дуже важливим для реалізації багатьох аспектів: атмосфери гри, механік так кількості гравців.

Таблиця 1.1 – Перелік найпопулярніших жанрів

Жанр	**Особливості**
Платформер	Гравець переміщає персонажа по рівнях, стрибає між платформами, уникає пасток і перешкод. Часто вимагає точної координації дій.
Шутер від першої особи	Гравець бачить світ очима персонажа, основна механіка — стрільба по супротивниках із використанням різного озброєння.
Пісочниця	Ігри без фіксованої мети, де гравець самостійно створює, змінює та взаємодіє зі світом. Основний акцент — на свободі дій.
Метроїдванія	Поєднання платформера з елементами пригодницької гри: дослідження світу, відкриття нових зон, повернення в попередні області з новими здібностями.
Жахи	Ігри з атмосферою страху, напруження та обмеженим ресурсом. Часто фокусуються на виживанні та психологічному впливі.
Ігри на виживання	Гравець повинен вижити в агресивному середовищі, збираючи ресурси, уникаючи небезпек та підтримуючи життєві параметри персонажа.
Масова онлайн-гра	Ігри з великою кількістю гравців одночасно. Відкритий світ, PvP/PvE, соціальні взаємодії. Потребує постійного підключення до інтернету.

Жанр	Особливості
Ігри з відкритим світом	Гравець може вільно пересуватись величезною картою, обираючи завдання, сюжет або дослідження на власний розсуд.
Візуальна новела	Акцент на читанні історії, прийнятті рішень і сюжетних розгалуженнях. Геймплей мінімальний, але має глибоке наративне наповнення.

Джерело[2]

На відміну від шутерів або MMORPG, платформер не потребує складної системи AI, мережевої взаємодії чи дорогих графічних ефектів, що значно знижує вимоги до ресурсоємності та дозволяє зосередитися на геймплейній логіці, дизайні рівнів та візуальному стилі. Це також робить платформер зручним для подальшого тестування, оптимізації та масштабування проєкту.

Особливу перевагу жанру надає і можливість реалізувати гру у low poly стилі, який гармонійно поєднується з платформерною механікою — спрощена геометрія, чіткі контури платформ, контрастність елементів. Це створює естетично привабливе, але не перевантажене візуальне середовище, яке позитивно впливає на ігровий досвід.

Також варто зазначити, що платформер легко поєднується з іншими жанровими компонентами — його можна розширити елементами пригоди, раннера або навіть освітньої гри. Така гнучкість дає змогу у перспективі модифікувати проєкт без повного переписування архітектури гри.

З урахуванням наведеного, вибір жанру платформер для реалізації дипломного проєкту є логічним, обґрунтованим і виправданим як з навчально-методичної, так і з техніко-графічної точки зору. Це дозволяє не лише ефективно застосувати знання в галузі розробки ігор, але й створити повноцінний програмний продукт із геймплейною цінністю та естетично завершеним візуальним виконанням.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Жанр платформерів посідає особливе місце в історії відеоігор, адже він став одним із перших жанрових напрямів, що заклали основи інтерактивного геймплею, побудованого на координації дій гравця, фізичній взаємодії персонажа з об’єктами середовища та просторовому мисленні. Платформери з’явилися ще на початку 1980-х років, коли технічні обмеження не дозволяли реалізовувати складні візуальні ефекти чи розгалужені сюжети, і саме тому увага акцентувалась на чіткій, зрозумілій та захопливій механіці.

Першим визнаним платформером в історії вважається гра Donkey Kong (1981), розроблена компанією Nintendo. У цій грі гравець мав підніматися платформами вгору, уникаючи перешкод і рятуючи героїню — ця формула поклала початок ключовим елементам жанру: вертикальному або горизонтальному переміщенню, стрибкам через прірви та взаємодії з об’єктами. Успіх Donkey Kong сприяв появі низки наслідувачів і поступовому формуванню жанрових конвенцій.

Одним із найвпливовіших представників жанру стала серія Super Mario Bros. (Nintendo, 1985), яка стандартизувала горизонтальне проходження рівнів, збирання бонусів, боротьбу з ворогами та багаторівневу структуру. Саме Super Mario остаточно закріпила платформер як масовий жанр, доступний широкій аудиторії, і стала зразком для численних наступних проєктів.

У 1990-х роках розвиток жанру продовжився завдяки технічному вдосконаленню ігрових консолей та комп’ютерів. З’явилися перші 3D-платформери, серед яких знаковою стала Super Mario 64 (1996), яка перенесла класичну платформерну механіку в тривимірний простір. Ігри цього періоду експериментували з камерою, рівневим дизайном, фізикою та варіативністю рухів персонажа.

У XXI столітті жанр пережив своєрідне відродження завдяки незалежним розробникам (інді-сцені), які почали створювати стильні, інноваційні платформери з унікальними візуальними стилями та глибокими ідеями. Яскравими прикладами стали Limbo, Celeste, Hollow Knight, Ori and the Blind Forest — ці проєкти довели,

що платформери можуть бути не лише аркадними, а й емоційно насиченими, сюжетно глибокими та технічно витонченими.

Сьогодні платформери залишаються актуальними як для комерційних, так і для навчальних проєктів. Їхня універсальність, відносна простота реалізації, можливість комбінувати з іншими жанрами (метроїдванія, екшен, головоломка) і висока гнучкість роблять платформери ідеальною основою як для початківців, так і для досвідчених розробників.

Однією з найбільш відомих[1] ігор цього жанру є “Super Mario”(рис. 1.1). Ця ягра встановила стандарт для 2D платформерів і мала величезний вплив на всю ігрову культуру в цілому.

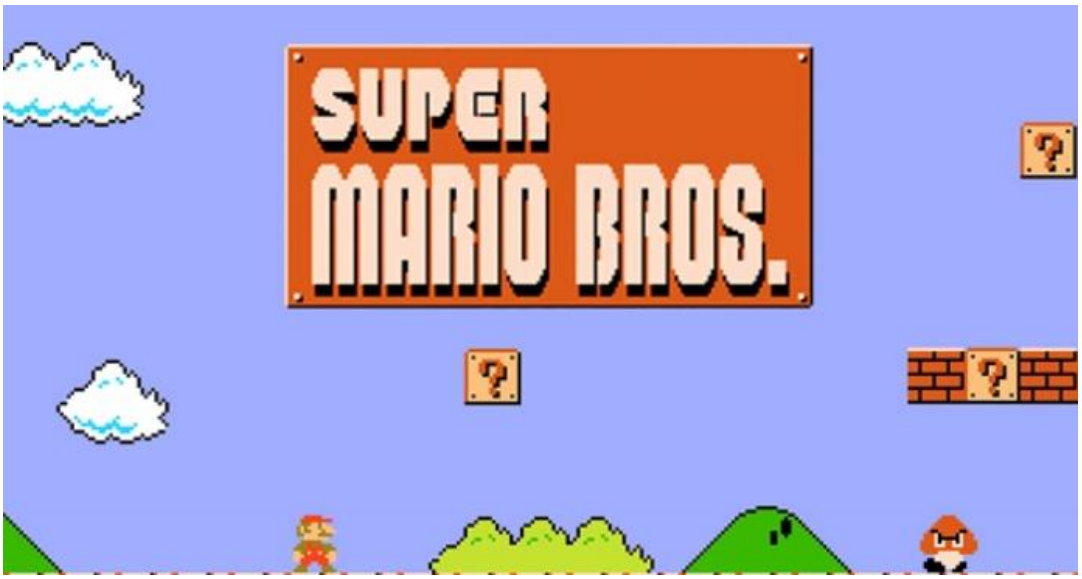


Рисунок 1.1 – Super Mario Bros

Джерело: [1]

Limbo (рис. 2.1) – 2010 року, яка є доказом того, що платформери можуть поєднувати в собі безліч різних жанрів та бути підходящими для “інді” студій. В центрі гри хлопчик який намагається знайти свою загублену сестру. Гра реалізована у дуже темних та тонах, що є не звичайним прийомом .

Рис. 1.2 – Limbo

Джерело: [1]

Crash Bandicoot(рис. 1.3) — це тривимірна платформерна гра, вперше випущена у 1996 році студією Naughty Dog для консолі Sony PlayStation. Гравець керує антропоморфним бандикутом на ім’я Креш, який долає низку рівнів, наповнених перешкодами, ворогами та пастками. Основною метою є просування вперед, збирання предметів (зокрема, яблук і ящиків), уникнення загроз та перемога над босами. [1]

Рис. 1.3 – Crash Bandicoot

Джерело: [1]

Spyro the Dragon (рис 1.3) — це класичний 3D-платформер, розроблений студією Insomniac Games та випущений у 1998 році для ігрової консолі Sony

PlayStation. Головним героєм виступає маленький фіолетовий дракон на ім’я Спайро, який подорожує різноманітними світах, визволяє заморожених старших драконів, збирає коштовні камені та бореться з ворогами. [1]

Рис. 1.4 – Spry the Dragon

Джерело: [1]

It Takes Two — це кооперативна пригодницька гра з елементами платформера, розроблена студією та випущена у 2021 році видавцем Electronic Arts. Гра створена виключно для проходження вдвох (локально або онлайн), що робить її унікальним прикладом повністю кооперативного дизайну. [1]

Рис. 1.5 – It take two

Джерело: [1]

На таблиці 2.1 наведено порівняльну характеристику найпопулярніших ігор платформерів.

Таблиця 1.2 – Порівняльна характеристика ігор [17]

Характеристика	SMB	Limbo	Crash Bandicoot	Spry the Dragon	It takes Two
3D простір	Ні	Ні	Так	Так	Так
Можливість гравця грати оналайн	НІ	Ні	Так	Так	Так
Можливість атакувати ворогів	Так	Ні	Так	Так	Ні
Можливість прокачувати персонажа	Ні	Ні	Так	Так	Ні
Можливість кастомізації	Ні	Ні	Ні	Ні	Так

Джерело: сформовано автором на основі виконаного дослідження

Зробивши порівняльну таблицю я можу сказати, що найкращий функціонал мають Spry the Dragon та Crash Bandicoot.

РОЗДІЛ 2.
РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ГРИ П

2.1 Аналіз і специфікація вимог до гри

Розробка будь-якої інтерактивної програмної системи, зокрема комп’ютерної гри, вимагає чіткого визначення вимог, що ставляться до майбутнього програмного продукту. Вимоги є фундаментом, на якому базується вся архітектура системи, її функціональні компоненти, поведінка та інтерфейс. Саме на цьому етапі визначається, які цілі має досягати система, які функції вона повинна реалізовувати, а також які обмеження, зовнішні умови та технічні особливості потрібно враховувати під час реалізації.

У загальному сенсі, вимоги — це формалізовані умови або можливості, яких має дотримуватись програмна система для задоволення очікувань користувача, відповідності технічним стандартам або укладеним угодам. Їх правильне визначення є складовою процесу системного інжинірингу, що безпосередньо впливає на успішність усього проєкту. Вимоги мають бути не лише коректними і чіткими, але й вимірними, перевірюваними, досяжними та не суперечити одна одній.

У першу чергу, хотів би визначити стейкхолдерів(stakeholders) для комп’ютерної гри.

– Гравець (кінцевий користувач) — очікує цікавої, зручної та стабільної гри.

– Розробник— реалізує функціональність, графіку, інтеграцію та оптимізацію.

– Менеджер проєкту— контролює якість, відповідність стандартам та академічним вимогам.

– Інверстори — інвестують у проєкт та керують, та задають його бюджетом.

– Тестувальники — перевіряють працездатність гри та звітують про помилки.

– Платформа публікації (умовно: Steam, itch.io) — має технічні та юридичні вимоги до гри (якщо планується розгортання).

Успішність розробки будь-якого програмного продукту, зокрема комп'ютерної гри, значною мірою залежить від здатності розробника вчасно і точно ідентифікувати та врахувати потреби, очікування й обмеження всіх зацікавлених сторін, або стейкхолдерів**.** Саме ці особи або організації впливають на постановку вимог, формування архітектури, оцінку якості проєкту та подальше його використання або впровадження.

Бізнес вимоги для комп'ютерної гри:

– Гра повинна бути цікавою та захопливою для цільової аудиторії, забезпечуючи позитивний ігровий досвід.

– Інтерфейс гри повинен бути інтуїтивно зрозумілим, легким для сприйняття та зручним для навігації.

– Гра повинна бути стабільною та технічно справною, не допускати фатальних помилок.

– Продукт має бути сумісним із цільовими платформами (наприклад, ПК, Android, iOS) і відповідати їх технічним та функціональним вимогам.

– Гра повинна підтримувати можливість масштабування та розширення

– Графічне оформлення гри має відповідати заявленій стилістиці, бути привабливим для користувача та не перевантажувати візуальне сприйняття.

– Час запуску гри повинен бути мінімальним, а навантаження ресурсів оптимізованим для стабільної роботи навіть на пристроях середньої продуктивності.

– Гра повинна мати документацію та інструкції для користувача, які пояснюють основні правила, керування та ігрові цілі.

– Проєкт гри має відповідати технічному завданню та очікуванням стейкхолдерів, зокрема — розробника, керівника, оцінювачів та користувача.

– Гра повинна мати документацію та інструкції для користувача**,** які пояснюють основні правила, керування та ігрові цілі.

Функціональні вимоги для комп’ютерної гри включають:

– Гра повинна забезпечувати керування персонажем — рух уперед/назад, стрибки, взаємодію з об’єктами та інші дії відповідно до механіки гри.

– Система повинна реагувати на зіткнення персонажа з об’єктами — визначати, чи це перешкода, бонус, ворог чи межа рівня.

– Гра повинна автоматично реєструвати збір предметів (наприклад, монет, кристалів) і оновлювати відповідні лічильники на екрані.

– Повинна бути реалізована система зміни стану гри, зокрема: старт гри, активна гра, поразка, завершення рівня, перезапуск.

– Інтерфейс гри повинен відображати основну інформацію — кількість зібраних предметів, поточний стан гри, підказки або повідомлення.

– Гра повинна забезпечувати можливість переходу між рівнями або сегментами гри відповідно до прогресу гравця.

Гра повинна надавати користувачеві можливість повноцінного керування ігровим персонажем у реальному часі. Основними діями є рух уперед або назад, стрибки, а також можливість взаємодії з об’єктами у просторі. Реалізація цього функціоналу має забезпечити високу чутливість до введення з клавіатури або контролера та відповідати очікуванням гравця щодо плавності й точності анімацій. Управління персонажем є центральною частиною геймплею і має критичний вплив на ігровий досвід.

Система повинна виявляти зіткнення персонажа з іншими об’єктами сцени, такими як перешкоди, бонуси, вороги чи межі рівня. Результатом зіткнення має бути певна реакція системи: зниження очок, завершення гри, активація анімації, зникнення зібраного об’єкта тощо. Ця функція реалізується за допомогою фізичних

колайдерів та відповідної логіки обробки подій. Правильна обробка колізій є запорукою інтерактивності та ігрового балансу.

Гра повинна автоматично реєструвати факт збирання об’єктів (монет, кристалів тощо) персонажем, видаляти ці об’єкти з ігрової сцени та оновлювати відповідні лічильники в інтерфейсі. Такий механізм не лише забезпечує елемент винагороди, але й мотивує гравця досліджувати простір гри та надає відчуття прогресу. Збір предметів супроводжується звуковими або візуальними ефектами, що підвищує загальну динаміку геймплею.

Повинна бути реалізована система станів гри, яка включає щонайменше такі етапи: початок гри, активна сесія, поразка (Game Over), завершення рівня, перезапуск. Перехід між станами має здійснюватися через події або взаємодію користувача (наприклад, натискання кнопки “Рестарт”). Наявність чіткої логіки станів дозволяє структурувати ігровий процес та забезпечити контроль за його перебігом.

Система повинна надавати гравцю візуальний інтерфейс, який відображає основну інформацію: кількість зібраних предметів, поточний стан гри, підказки або повідомлення про дії. Інтерфейс має бути адаптивним, зручним для сприйняття та не відволікати від основної дії. Елементи UI повинні оновлюватися в реальному часі та відповідати поточному стану гри.

Після завершення певного ігрового сегменту або досягнення заданої умови, система повинна забезпечити автоматичний або керований користувачем перехід до наступної ігрової зони або рівня. Це дозволяє реалізувати прогресивну структуру гри та створювати відчуття досягнення мети. Такий перехід може супроводжуватись анімацією, завантаженням нової сцени або зміною декорацій.

Можливі нефункціональні вимоги комп’ютерної гри:

– Продуктивність. Гра повинна працювати з частотою не нижче 30 кадрів за секунду на пристроях середньої продуктивності, без помітних підвисань чи затримок.

– Сумісність. Гра має коректно функціонувати на обраній цільовій платформі (наприклад, Windows, Android), відповідати її технічним вимогам і підтримувати стандартне апаратне забезпечення.

– Надійність. Система повинна бути стійкою до збоїв .

– Юзабіліті (зручність використання). Інтерфейс гри повинен бути інтуїтивно зрозумілим для користувача без потреби звертатися до додаткових інструкцій. Навігація та керування мають бути простими та логічними.

– Безпека. У разі роботи з особистими даними або збереженням прогресу користувача (у повнофункціональній версії) має бути передбачений захист інформації .

Було створено діаграми в середовищі Draw.io[12] з функціональними і нефункціональними вимогами і вони наведені на рисунках 2.1 – 2.5.

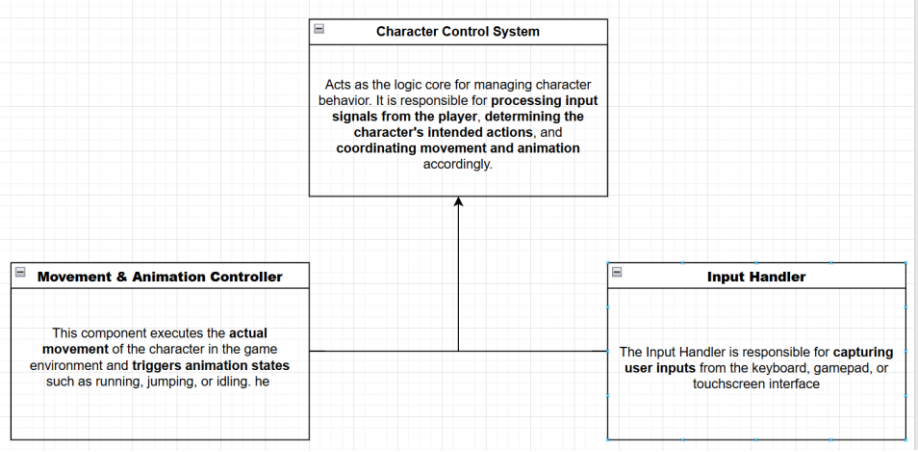


Рис. 2.1 - Функціональні вимоги системи керування

Джерело: сформовано автором на основі виконаного дослідження

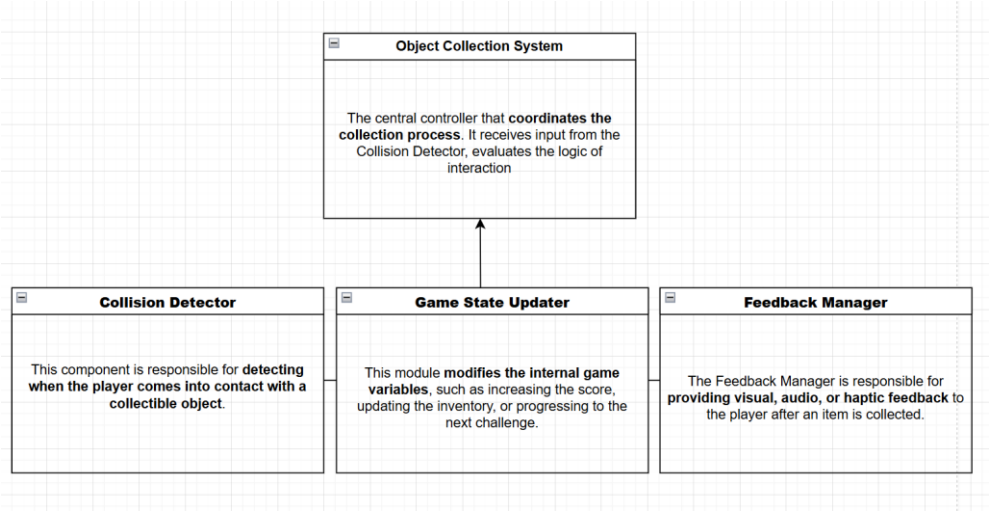


Рис. 2.2 - Функціональні вимоги системи збору об’єктів

Джерело: сформовано автором на основі виконаного дослідження

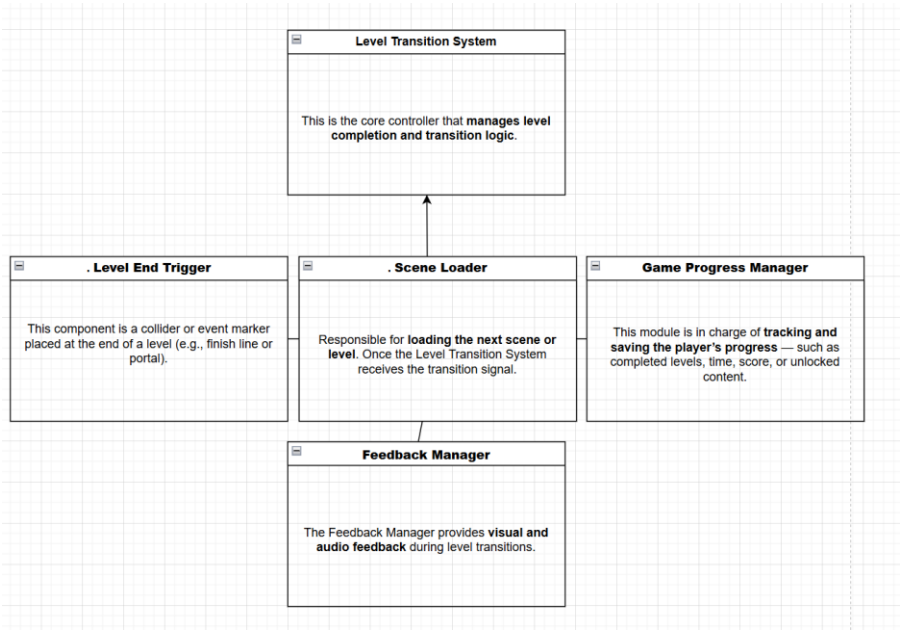


Рис. 2.3 - Функціональні вимоги системи генерації нових рівнів

Джерело: сформовано автором на основі виконаного дослідження

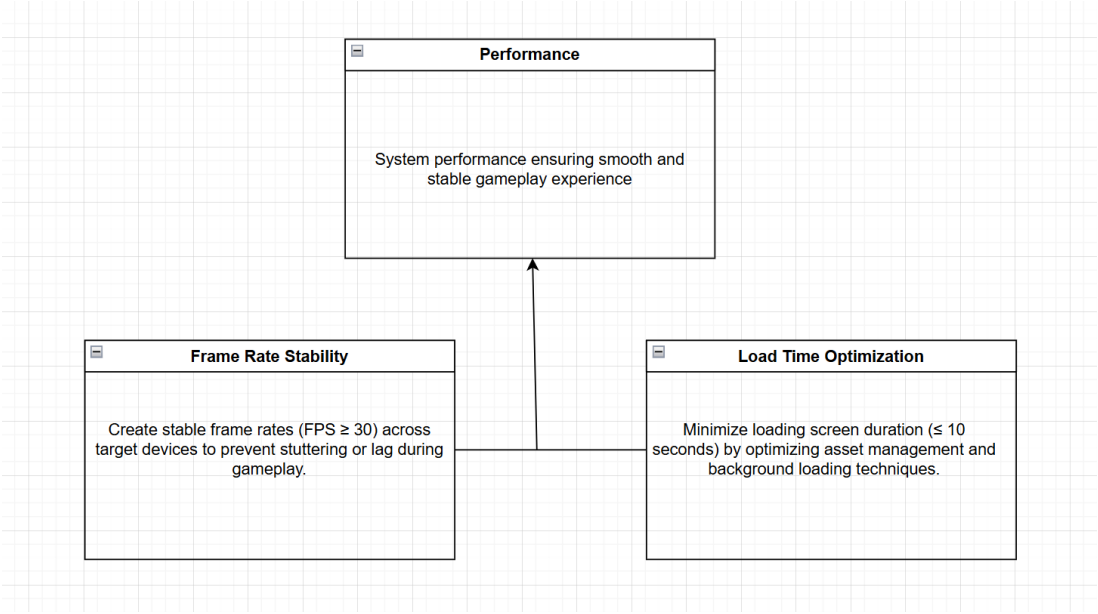


Рис. 2.4 - Нефункціональні вимоги для Performance

Джерело: сформовано автором на основі виконаного дослідження

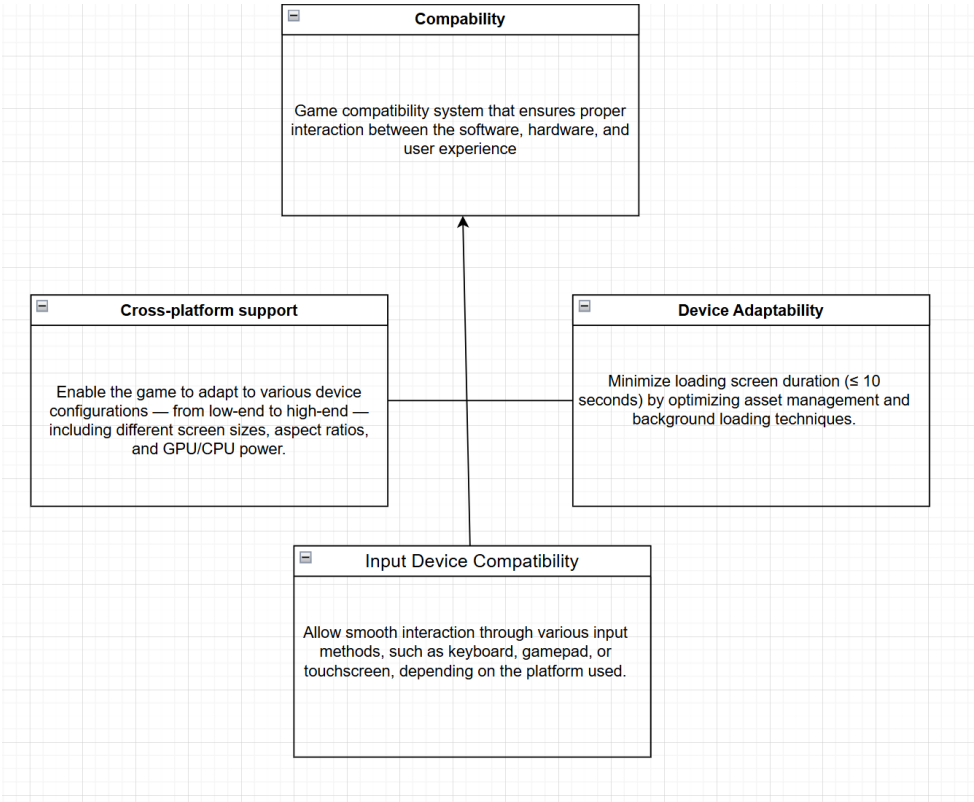


Рис. 2.5 - Нефункціональні вимоги для Сумісності

Джерело: сформовано автором на основі виконаного дослідження

2.2 Постановка розв’язання задачі

При моделюванні платформерної гри передбачає створення формалізованої репрезентації її ключових компонентів — таких як рух персонажа, взаємодія з ігровим середовищем, збирання об’єктів, переходи між рівнями та відображення стану гри. Така модель дозволяє не лише структурувати логіку ігрового процесу, а й забезпечити можливість тестування функціональних сценаріїв та оптимізації взаємодії між системними модулями.

Основною метою моделювання є розробка цілісної, структурованої моделі гри, яка дозволить перевірити коректність реалізації її механік, відповідність користувацькому досвіду, а також забезпечить основу для майбутнього масштабування або внесення змін. Це включає опис сценаріїв поведінки гравця, алгоритмів керування персонажем, обробки подій, а також взаємодії з користувацьким інтерфейсом.

У випадку платформерів особливу увагу необхідно приділити реалізації просторових переміщень, виявленню зіткнень з перешкодами, зміні станів персонажа, а також врахуванню часових характеристик (наприклад, затримка відгуку на введення). Дані ігрового процесу можна представити за допомогою структур даних, схем логіки, а також сценарних моделей, які забезпечують перевірку поведінки системи у різних ігрових умовах.

Характеристика задачі

Призначення гри-платформера полягає у створенні захоплюючого інтерактивного середовища, яке поєднує елементи просторової навігації, реактивного керування та дослідження рівнів. Така гра формує у гравця навички точності руху, орієнтації у тривимірному просторі, а також покращує зорову пам’ять і рефлекси. Основною особливістю є потреба в точному синхронному виконанні стрибків, ухилень і взаємодій з об’єктами у реальному часі.

Об’єкти, за управлінням якими розв’язується задача, включають: ігровий персонаж, яким керує користувач; платформи, перешкоди та бонусні об’єкти, з

якими він взаємодіє; сцени, що складаються з рівнів; а також інтерфейс, який відображає інформацію про стан гравця (очки, зібрані предмети, повідомлення про виконані досягнення).

Призначення вихідної інформації полягає у відображенні прогресу гравця та стану гри. До неї належать: кількість зібраних монет, досягнутий рівень, с, а також кінцевий результат у разі виграшу або поразки. Також ключовим призначенням є рекорд гравця та його відображення у списку лідерів. Крім того, інформація може включати підказки, повідомлення про помилки, індикатори взаємодії або статистику проходження.

Періодичність розв’язання задачі визначається тривалістю проходження рівня та частотою дій гравця. Оновлення вихідної інформації та реакції системи відбувається після кожної дії (стрибок, зіткнення, збір об’єкта), а також при завершенні рівня або у випадку програшу.

Автоматизоване розв’язання задачі реалізується шляхом запуску логіки гри, обробки подій фізичної взаємодії та використання вбудованих систем Unity (наприклад, фізика, UI, анімації), що забезпечують динамічну реакцію на події в ігровому просторі.

Зв’язки задачі показані на інформаційній моделі(рис 2.6)

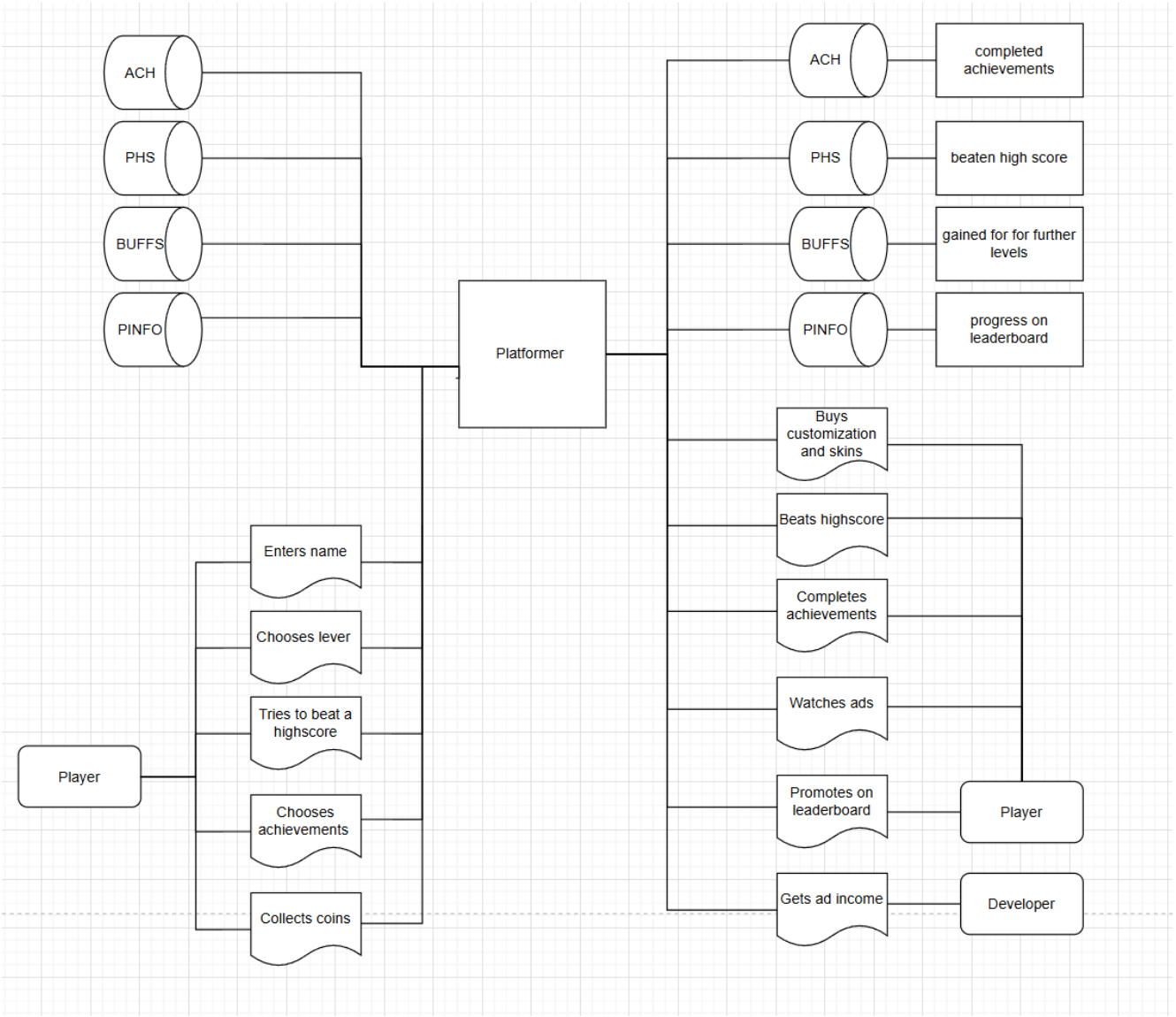


Рис. 2.6 – інформаційна модель

Джерело: сформовано автором на основі виконаного дослідження

Вхідні дані

Так як гравець починаю гру з самого початку кожен раз вхідних даних потребується мінімальна кількість. Гравець повинен ввести своє ігрове ім’я та рівень, який він хоче пройти. Також при початку у нього зберігається найкращий рахунок за попередні спроби і він має намагатися покращити його.. На таблиці 2.1 представлений перелік та опис вхідних повідомлень.

Таблиця 2.1 – перелік та опис вхідних повідомлень

№	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Досягнення	ACH	Файл	За потребою	Персональний комп’ютер
2	Найкращий результат	PHS	Файл	За потребою	Персональний комп’ютер
3	Покращення і підсилення гравця	BUFFS	Файл	За потребою	Персональний комп’ютер
4	Список гравців	PINFO	Файл	За потребою	Персональний комп’ютер
5	Введення ім'я	Enters Name	Віконна форма	За потребою	Персональний комп’ютер
6	Вибір рівня	Chosses level	Віконна форма	За потребою	Персональний комп’ютер
7	Ціль побити найкращий результат	Tries to beat highscore	Віконна форма	За потребою	Персональний комп’ютер
8	Вибирає досягнення	Chooses achievements	Віконна форма	За потребою	Персональний комп’ютер

9	Збирає монети	Collects coins	Віконна форма	За потребою	Персональний комп’ютер

Джерело: сформовано автором на основі виконаного дослідження

Вихідні дані

На виході гравець отримує певну суму монеток. На ці монетки він зможе купувати костюми і різні бонуси, які будуть його посилювати на наступних рівнях. Також можливо гравець отримає новий рекорд і це сприяє його руху на таблиці лідерів. Перелік вихідних даних наведений на таблиці 2.2.

Таблиця 2.2 – перелік та опис вихідних повідомлень

№ з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
1	Досягнення	ACH	Файл	Після поразки	За потребою	Гравець
2	Побитий рекорд	PHS	Файл	Після поразки	За потребою	Гравець,
3	Посилювання, які зібрав гравець	BUFFS	Файл	Після поразки	За потребою	Гравець
4	Список гравців	PINFO	Віконна форма	За потребою	За потребою	Гравець, розробник

5	Доступ до нових костюмів	Buys new customization	Віконна форма	Після поразки	За потребою	Гравець
6	Б’є свій рекорд	Beats highscore	Віконна форма	Після поразки	За потребою	Гравець
6	Просування в таблиці лідерів	Promotion on the leaderboard	Віконна форма	Після поразки	За потребою	Гравець, розробник
7	Виконане досягнення	Completes Achievement	Віконна форма	Після поразки	Миттєво	Гравець
8	Показ реклами	Ads being displayed	Віконна форма	Кожен місяць	Стандартний час банківського переказу	Гравець
9	Дохід від реклами	Ads income	Віконна форма	Кожен місяць	Час визначається банко або керівником	Розробник

Джерело: сформовано автором на основі виконаного дослідження

2.3 Моделювання інформаційної підсистеми

На рисунку 2.7 зображено діаграму прецедентів (Use Case Diagram), яка демонструє основні сценарії взаємодії гравця з ігровою системою. Такий тип діаграми дозволяє візуально представити функціональні вимоги до системи через дії користувача (акторів) та реакції програмного середовища.

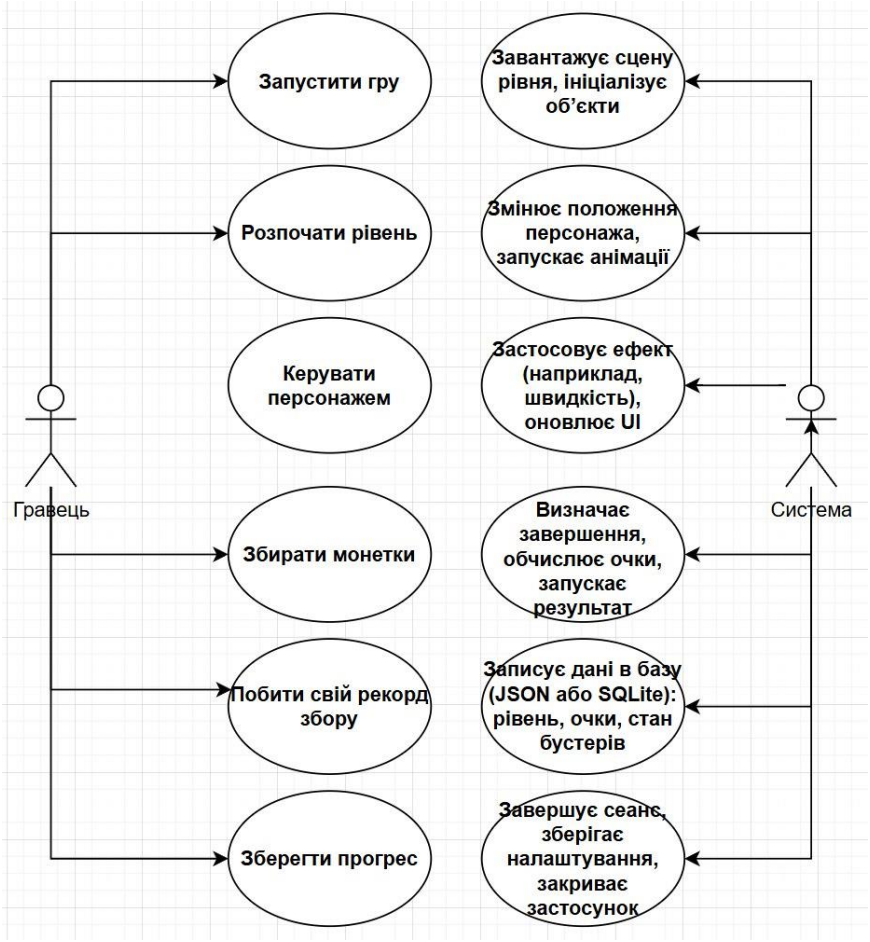


Рис. 2.7 – Діаграма прецедентів

Джерело: сформовано автором на основі виконаного дослідження

У ролі основного користувача виступає гравець, який взаємодіє з шістьма ключовими функціональними прецедентами:

– Запустити гру — гравець ініціює запуск гри, після чого система завантажує сцену рівня та ініціалізує всі необхідні об'єкти.

– Розпочати рівень — активується геймплей: система змінює положення персонажа, активує анімації та стартову логіку.

– Керувати персонажем — гравець взаємодіє з керуванням (переміщення, стрибки), а система відповідно змінює параметри руху, застосовує ефекти (наприклад, швидкість), оновлює інтерфейс користувача.

– Збирати монетки — основна геймплейна механіка, що супроводжується підрахунком очок, запуском UI-реакцій та активацією ефектів.

– Побити свій рекорд збору — система обробляє дані, визначає нові рекорди, зберігає їх у базу (наприклад, у форматі JSON або SQLite).

– Зберегти прогрес — при виході або завершенні гри система записує дані сеансу, зберігає налаштування та коректно завершує застосунок.

З боку системи (другий актор) автоматично виконуються усі допоміжні функції — ініціалізація сцен, збереження даних, обробка ефектів та оновлення інтерфейсу.

Дана діаграма дозволяє структуровано представити логіку гри з точки зору користувача, а також є основою для формалізації функціональних вимог до майбутнього ігрового додатку. Такий підхід сприяє ясності проєктування та подальшому розробленню технічного завдання.

На діаграмі класів (рис. 2.8) представлено основну об'єктно-орієнтовану структуру системи 3D-гри в жанрі платформера. Вона моделює ключові компоненти гри, їхні атрибути, методи та взаємозв’язки. Такий підхід дозволяє впорядкувати архітектуру проєкту, забезпечити масштабованість та логічну розподіленість функціоналу між окремими класами.

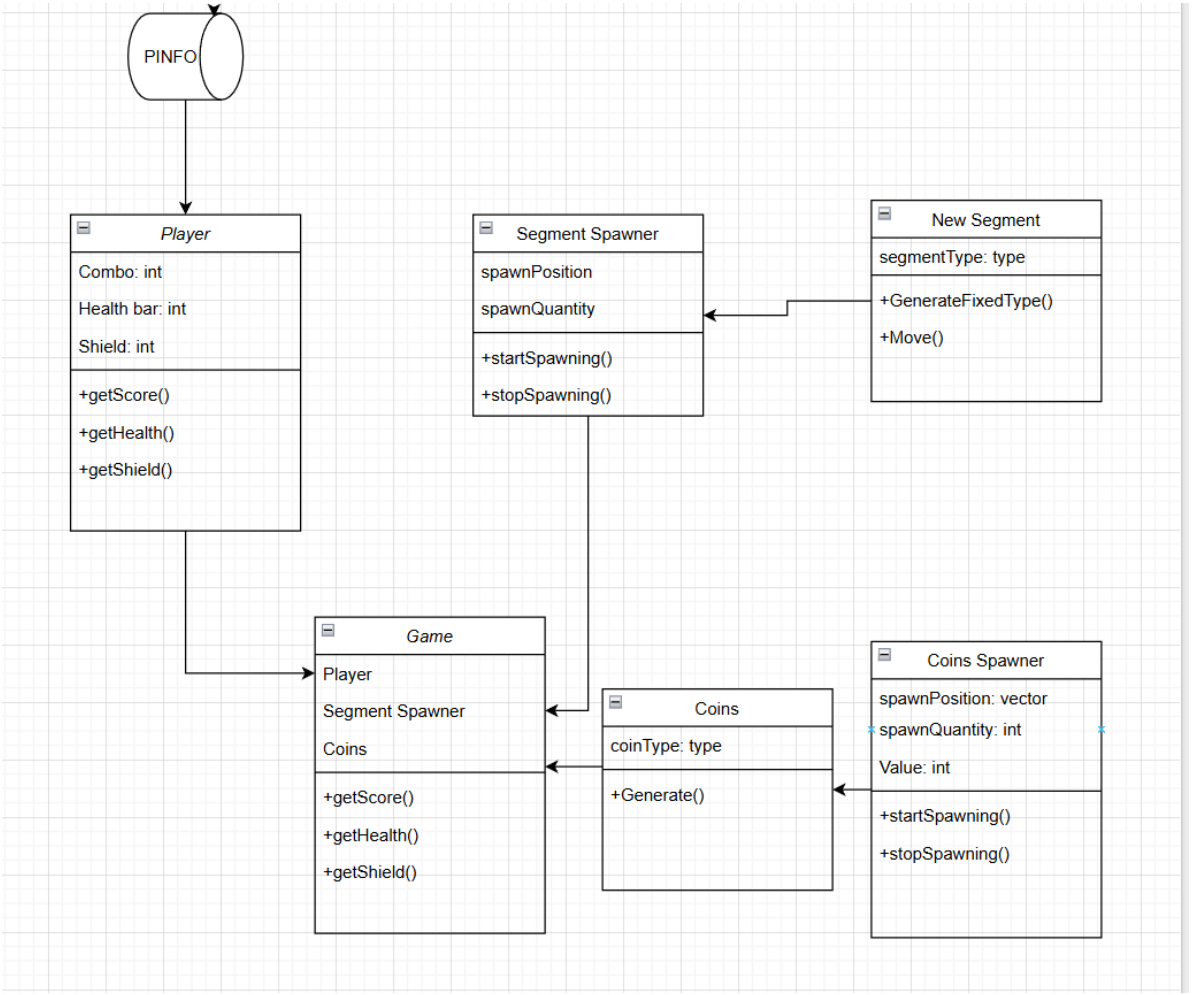


Рис. 2.8 – діаграма класів

Джерело: сформовано автором на основі виконаного дослідження

Центральним об’єктом є клас Game, який виступає як основний контролер усієї логіки гри. Він містить посилання на три ключові компоненти: Player, Segment Spawner і Coins. Через доступ до цих класів Game реалізує базову логіку: підрахунок очок, оновлення здоров’я, взаємодію з гравцем і генерацію ігрових об’єктів. У класі передбачені методи getScore(), getHealth() та getShield() — для отримання поточних параметрів ігрової сесії.

Клас Player моделює головного персонажа. Він містить внутрішні характеристики персонажа — combo, health bar, shield — та відповідні геттери для їх читання. Крім того, він взаємодіє з іншими класами системи, надаючи доступ до своєї внутрішньої інформації, необхідної для UI або ігрової логіки. Дані Player

можуть бути збережені або завантажені через зовнішній ресурс PINFO, що позначає джерело або сховище даних гравця.

Segment Spawner є відповідальним за генерацію ігрових сегментів (наприклад, платформ або ділянок рівня). Він має атрибути spawnPosition та spawnQuantity, які визначають координати появи й кількість об’єктів. Методи startSpawning() і stopSpawning() контролюють початок і завершення генерації. Кожен згенерований сегмент описується класом New Segment, який має властивість segmentType і методи GenerateFixedType() (визначення типу платформи) та Move() — для управління переміщенням об’єкта.

Клас Coins представляє ігрові монети, які гравець може збирати. Він містить атрибут coinType і метод Generate() для створення нових монет. Взаємодія із середовищем генерації здійснюється через клас Coins Spawner, де визначаються позиція, кількість і значення монет (value). Спавнер також має методи startSpawning() та stopSpawning() для контролю циклу появи об’єктів.

Уся структура побудована таким чином, щоб забезпечити модульність, тобто можливість незалежної розробки й тестування окремих частин системи. Взаємодія між класами здійснюється через чітко визначені інтерфейси .

На рисунку 2.9 представлена Sequence діаграма.

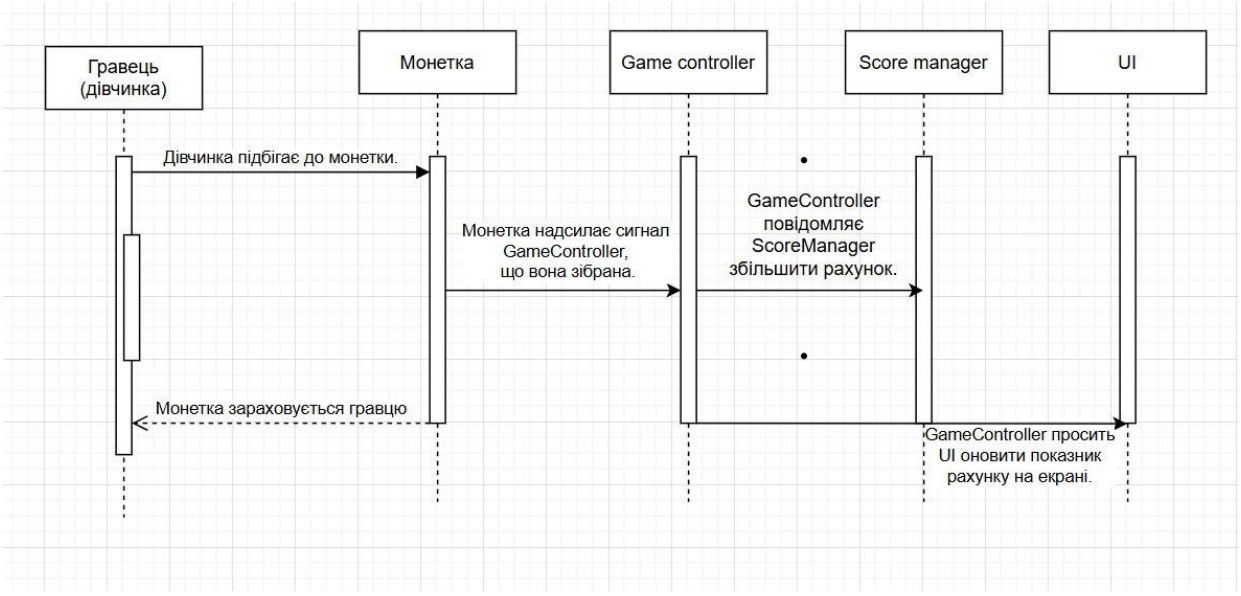


Рис. 2.9 – Sequence діаграма

Джерело: сформовано автором на основі виконаного дослідження

У ролі ініціатора сценарію виступає гравець (дівчинка), що наближається до об’єкта типу Монетка. У момент дотику до монети, вона розпізнається як зібрана, і Монетка передає сигнал до GameController про успішне зіткнення. Це повідомлення є тригером для запуску внутрішньої логіки обробки події.

Далі GameController звертається до об'єкта ScoreManager із запитом збільшити рахунок гравця. Цей крок є критичним для реалізації механіки нарахування очок у геймплеї. Після успішного оновлення кількості очок, GameController надсилає команду до інтерфейсу користувача (UI) з метою візуального оновлення показника рахунку на екрані гри.

На завершальному етапі діаграми відбувається зворотне завершення усіх дій — монетка зараховується гравцю, оновлений рахунок відображається в інтерфейсі, а система переходить до очікування наступної події.

Ця Sequence-діаграма демонструє чітко структуровану модель обміну повідомленнями між об’єктами, а також втілює важливий принцип програмної архітектури — розподілену відповідальність. Такий підхід підвищує гнучкість, забезпечує зручність тестування й масштабованість гри.

РОЗДІЛ 3

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ГРИ ПЛАТФОРМЕРА

3.1 Інформаційне забезпечення

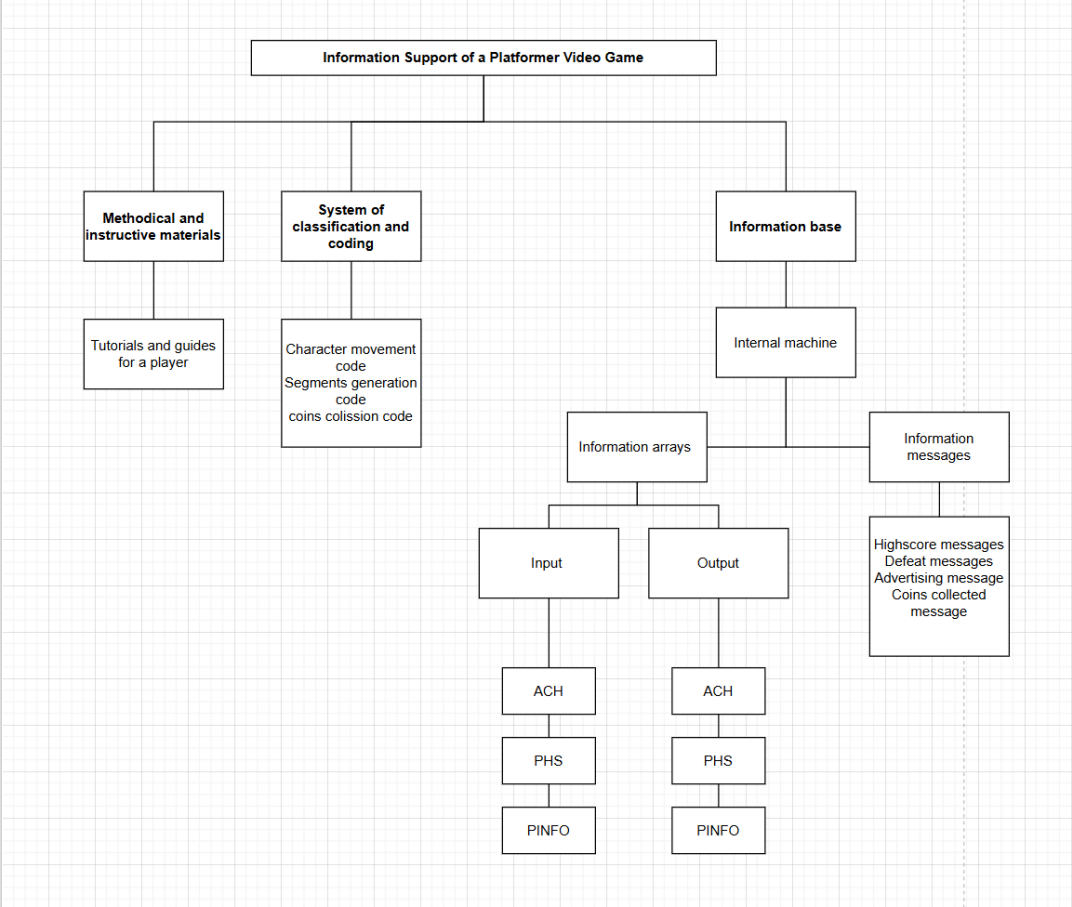


Рис. 3.1 – інформаційне забезпечення для гри платформера

Джерело: сформовано автором на основі виконаного дослідження

Представлена на рисунку 3.1 схема ілюструє структуру інформаційного забезпечення 3D гри у жанрі платформера, що була розроблена для формалізації й систематизації всіх інформаційних компонентів, необхідних для повноцінного функціонування гри. Основною метою створення даної схеми є визначення джерел, типів і потоків інформації, що використовуються у процесі гри — як на рівні ігрової логіки, так і в межах взаємодії з користувачем.

Інформаційне забезпечення поділяється на три основні підсистеми:

– Методичні та інструктивні матеріали, які включають посібники та підказки для гравця (tutorials and guides). Ці елементи мають на меті полегшити вивчення керування та правил гри, що підвищує рівень залучення користувача і покращує ігровий досвід.

– Система класифікації та кодування — охоплює основну логіку роботи з кодовими структурами, включаючи реалізацію руху персонажа, генерацію платформ (сегментів) та обробку зіткнень із монетками. Це забезпечує формалізацію внутрішньої логіки гри та чітку декомпозицію функціональних компонентів, що спрощує розробку, тестування та обслуговування гри.

Третім типом є інфрмаційна база, яка складається з таких елементів:

– Internal machine — внутрішній ігровий механізм, який керує масивами даних та обробляє інформаційні повідомлення.

– Інформаційні масиви (Information arrays) поділяються на вхідні (input) та вихідні (output).

До них належать:

– ACH (Achievements) — досягнення, яких досяг гравець;

– PHS (Previous Highscore) — збережене попереднє максимальне значення очок;

– PINFO (Player Info) — інформація про рахунок гравця, що зберігається в таблиці лідерів (leaderboard).

– Інформаційні повідомлення (Information messages), що включають повідомлення про нові рекорди, поразку, рекламу або факт збирання монет. Ці повідомлення передаються до інтерфейсу користувача та є важливою частиною UI-комунікації.

– Призначення та переваги схеми

Ця схема була розроблена з метою:

– структуризації всієї інформації, що циркулює у грі під час виконання основних дій (рух, збирання, взаємодія з UI);

– оптимізації роботи з даними через поділ на логічні блоки;

– підвищення зрозумілості архітектури гри як для розробника, так і для рецензента або потенційного командного розширення проєкту.

Перевагою цієї схеми є її модульність та розширюваність — при потребі вона легко адаптується до складніших проєктів. Чітке розділення між логікою, даними та інтерфейсом дозволяє ефективно управляти змінами та забезпечити відповідність вимогам як користувача, так і системи.

Організація збору і передання первинної інформації

У даному підрозділі визначено джерела, носії та правила збору первинної інформації, яка використовується для функціонування основних механік гри. Первинна інформація у 3D платформері — це вхідні дані, які зчитуються, обробляються та записуються в ігрову логіку або базу даних (у форматі JSON), і визначають ігровий прогрес, стан гравця, параметри об’єктів тощо.

До таких повідомлень належать: ім’я гравця, зібрані монети, кількість очок, інформація про виконані досягнення, а також сигнали зіткнення з перешкодами або предметами на рівні. Кожне з цих повідомлень має джерело, яке відповідає за своєчасне формування та передання даних у пам’ять гри.

Джерелами інформації виступають:

– внутрішні скрипти Unity (наприклад, контролер персонажа або генератор перешкод),

– компоненти керування анімаціями чи UI,

– вбудована база даних JSON (використовується для збереження прогресу, рахунку, рекордів),

– зовнішній файл налаштувань або таблиці результатів (опціонально),

– взаємодія з користувачем (гравець ініціює події через керування).

– Передача даних відбувається автоматично у відповідь на події в грі: збирання монети, завершення рівня, оновлення інтерфейсу або вихід із гри. Дані обробляються класами GameController, ScoreManager, UIController, після чого зберігаються у вигляді структурованих записів.

– Носіями інформації виступають:

– Оперативна пам’ять (під час сесії гри),

– Файли playerData.json у Application.persistentDataPath (постійне збереження),

– Компоненти сцени Unity (як об'єкти, що формують середовище, UI тощо).

– Також зазначається періодичність та спосіб подачі інформації:

– Після кожного зібраного об’єкта або події — негайна передача в RAM;

– Після завершення рівня або виходу з гри — збереження у JSON;

– Після завантаження гри — зчитування з файлу для ініціалізації стану.

Таким чином, система збору і передачі первинної інформації в грі забезпечує узгоджену, надійну ігрову логіку, дозволяє зберігати прогрес гравця та підтримує цілісність даних у процесі гри.

Побудова системи класифікації та кодування об'єктів у грі

У процесі розробки комп’ютерної гри у жанрі платформера було реалізовано систему класифікації і кодування ігрових об’єктів. Її основна мета — забезпечити зручну, однозначну і масштабовану систему ідентифікації усіх елементів гри, які використовуються в геймплейній логіці, обробці подій та збереженні інформації у базі даних (JSON).

Класифікація охоплює основні об’єкти гри, зокрема:

- типи платформ або сегментів;
- типи монет;
- типи перешкод;
- типи досягнень;
- стани гравця;
- повідомлення для інтерфейсу.

Для кожної категорії об'єктів було використано власний класифікатор, який реалізовано у вигляді перерахувань (enum) або стандартних скорочень у структурі JSON.

Проєктування форм первинних документів, машинограм та відеокадрів.

У процесі розробки гри важливо формалізувати структуру і представлення первинної інформації, яка використовується або зберігається у вигляді певних документів або даних. У контексті комп'ютерної гри документами виступають:

- машинограми (лог-файли подій) — події, що відбуваються у грі (зіткнення, збір монет, оновлення UI);
- інформаційні сигнали — внутрішні команди між скриптами (наприклад, ScoreManager -> UIController);
- відеокадри або знімки екрану — які можуть бути збережені під час тестування або досягнення гравцем певного результату;
- JSON-документи — локальна база даних, яка містить інформацію про гравця, очки, досягнення тощо.

Основні параметри документів:

– Найменування: наприклад, playerData.json, sessionLog.txt, achievements.json

– Ідентифікатор: кожен документ має унікальне ім'я або ключ (наприклад, "id": "ACH_10C")

– Користувачі: внутрішні скрипти Unity, які читають або записують ці документи (SaveManager, GameController, LeaderboardSystem)

– Періодичність оновлення:

- JSON-база — після завершення гри або рівня
- лог-файли — під час виконання подій
- UI-статуси — у реальному часі

– Термін зберігання: до моменту видалення або перезапису при новій сесії

– Форма подання:

- JSON (для основної бази)
- текстові файли (логи, відладка)
- графічні зображення (опціонально, скріншоти)

Надсилання документів

У разі онлайн-гри або мультиплеєра можлива інтеграція з серверною частиною — тоді деякі документи (наприклад, рекорди, статистика) можуть бути відправлені через інтернет або на email (наприклад, у вигляді аналітичного звіту чи архіву логів).

Структура інформаційних масивів

У цьому розділі наведено опис структури інформаційних масивів(табл. 3.1), що використовуються у JSON-базі даних гри. Масиви призначені для зберігання ключових ігрових параметрів, таких як очки, кількість зібраних монет, значення комбо та досягнення гравця. Визначено назви полів, їхні внутрішні ідентифікатори, умовні позначення для формулювання правил гри, а також формати даних для обробки в межах програмної логіки. Така структура дозволяє організувати інформацію ефективно, забезпечуючи простоту доступу, цілісність

Таблиця 3.1 – Опис структури масивів

0	Найменування	Ідентифікатор у програмі	Умовне позначення	Формат	Умова на значення	Обов’язкове поле	Індексне поле
1	Ім’я гравця	playerName	PN	string	не пусте	так	ІДД
2	Кількість очок	score	SC	int	>= 0	так	ІДД
3	Кількість монет	coins	CO	int	>= 0	ні	-
4	Ідентифікатор	id	ACH_ID	string	унікальне, не пусте	так	-
5	Статус досягнення	unlocked	ACH_U	bool	>= 0	так	ІДД

Джерело: сформовано автором на основі виконаного дослідження

Вибір СКБД

Використання СКБД у форматі JSON (тобто системи керування базами даних на основі збереження структурованих даних у .json файлі) є дуже вдалим рішенням для гри в Unity з кількох вагомих причин.

JSON не потребує складної інфраструктури, як у випадку з SQL-базами або серверними СКБД. У Unity достатньо одного скрипта для серіалізації та десеріалізації даних через JsonUtility, що суттєво знижує складність і прискорює розробку.

JSON — це формат, який легко читається не лише машиною, а й людиною. Його структура ієрархічна, а отже, чудово підходить для зберігання вкладених даних (наприклад, масивів досягнень, налаштувань гравця тощо).

Unity має вбудований механізм JsonUtility, що дозволяє легко перетворювати будь-які `C#` класи на JSON та навпаки. Це зручно і продуктивно для збереження.

Реалізація СКБД

У рамках розробки 3D гри-платформера виникає необхідність зберігати ігрову інформацію локально. Це стосується таких даних, як: очки гравця, кількість зібраних монет, досягнення, ім’я користувача, попередні результати, а також налаштування звуку чи графіки. Для цього було прийнято рішення використовувати локальну JSON-базу даних, що не потребує підключення до серверів або зовнішніх СУБД.

Призначення бази даних

Розроблена JSON-база даних у грі виконує такі функції:

- Зберігає прогрес гравця (score, combo, coins)
- Зберігає інформацію про гравця (ім’я, статус)
- Фіксує досягнення (Achievements) — виконані дії
- Зберігає таблицю рекордів (Leaderboard)
- Дозволяє зчитувати дані при запуску гри
- Дозволяє оновлювати та перезаписувати дані

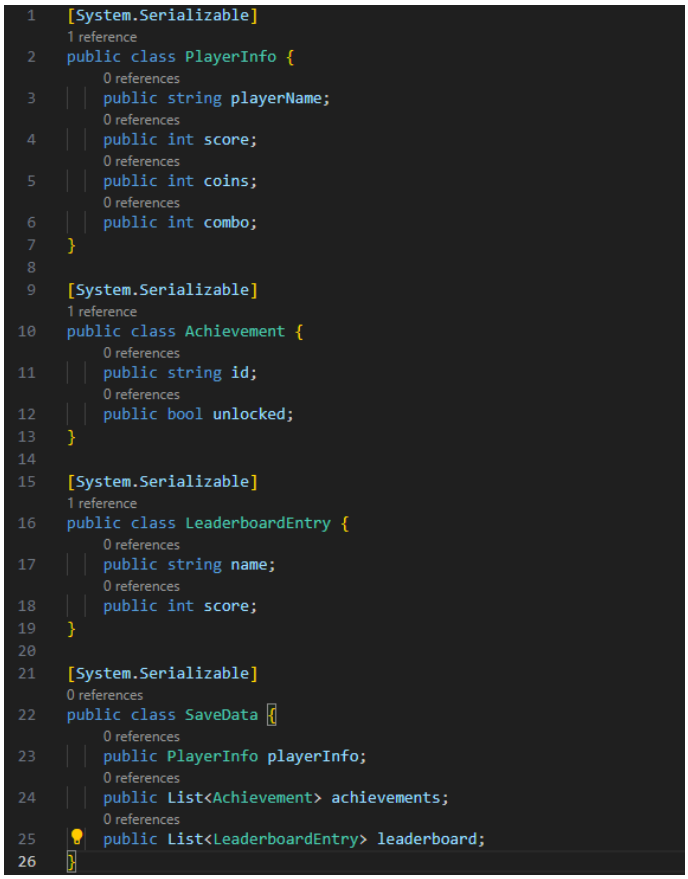
Структура JSON-файлу наведена на малюнку 3.2

```json
{
  "playerInfo": {
    "playerName": "Player1",
    "score": 1200,
    "coins": 88,
    "combo": 3
  },
  "achievements": [
    { "id": "first_jump", "unlocked": true },
    { "id": "collect_100_coins", "unlocked": false }
  ],
  "leaderboard": [
    { "name": "Max", "score": 2000 },
    { "name": "Player1", "score": 1200 }
  ]
}
```

Рис. 3.2 – структура Json файлу

Джерело: розроблено автором самостійно

У Unity створено окремий клас PlayerData.cs (рис. 3.3)



```
1   [System.Serializable]
    1 reference
2   public class PlayerInfo {
        0 references
3       public string playerName;
        0 references
4       public int score;
        0 references
5       public int coins;
        0 references
6       public int combo;
7   }
8
9   [System.Serializable]
    1 reference
10  public class Achievement {
        0 references
11      public string id;
        0 references
12      public bool unlocked;
13  }
14
15  [System.Serializable]
    1 reference
16  public class LeaderboardEntry {
        0 references
17      public string name;
        0 references
18      public int score;
19  }
20
21  [System.Serializable]
    0 references
22  public class SaveData {
        0 references
23      public PlayerInfo playerInfo;
        0 references
24      public List<Achievement> achievements;
        0 references
25      public List<LeaderboardEntry> leaderboard;
26  }
```

Рис. 3.3 – клас PlayerDate

Джерело: розроблено автором самостійно

Створення менеджера для збереження/завантаження. У скрипті SaveManager.cs (рис. 3.3) це реалізовано.

```
using System.IO;
using UnityEngine;

0 references
public class SaveManager : MonoBehaviour {
    4 references
    string path;

    0 references
    void Awake() {
        path = Application.persistentDataPath + "/data.json";
    }

    0 references
    public void Save(SaveData data) {
        string json = JsonUtility.ToJson(data, true);
        File.WriteAllText(path, json);
    }

    0 references
    public SaveData Load() {
        if (File.Exists(path)) {
            string json = File.ReadAllText(path);
            return JsonUtility.FromJson<SaveData>(json);
        } else {
            return new SaveData();
        }
    }
}
```

Рис. 3.4 – клас SaveManager

Джерело: розроблено автором самостійно

Інтеграція JSON як формату збереження даних у Unity забезпечила просту, надійну та ефективну систему локального збереження. Завдяки цьому гравець може продовжити гру з того місця, де зупинився, а також бачити свої рекорди та досягнення. Такий підхід є оптимальним для платформерів і демонструє сучасний стандарт роботи з даними у незалежній ігровій розробці.

База даних представлена у вигляді окремого .json-файлу, який автоматично створюється або оновлюється під час гри. Він зберігається в системній папці Application.persistentDataPath, що забезпечує доступність даних навіть після виходу з гри.

У момент запуску гри скрипт SaveManager зчитує файл і завантажує в пам’ять:

- ім’я гравця;
- кількість очок;

- зібрані монети;
- активний множник комбо;
- список досягнень;
- дані таблиці рекордів.

Упродовж гри ці значення можуть змінюватися (наприклад, при зборі монет або встановленні нового рекорду), а при завершенні рівня або виході з програми — автоматично зберігаються в той самий файл JSON.

3.2 Технічне забезпечення

У процесі розробки комп’ютерної гри у жанрі 3D платформера було прийнято рішення орієнтувати фінальний продукт на персональні комп’ютери (ПК) як основну цільову платформу. Такий вибір зумовлений низкою технічних, логістичних та функціональних чинників, які роблять саме ПК найбільш зручною і доцільною середовищем для реалізації та запуску проєкту в рамках навчального або інді-проєкту.

Персональні комп’ютери забезпечують максимальну сумісність з ігровим рушієм Unity**,** що був обраний як основна технологічна платформа. Unity має нативну підтримку побудови проєктів під Windows, macOS та Linux, але найбільш стабільною, поширеною та підтримуваною операційною системою для тестування й релізу є Windows 10/11, яка надає широкий набір інструментів для розробника, включно з діагностикою, відлагодженням та профілюванням гри. Також задля змагання з іншими гравцями, оновлення і синхронізації необхідне підключення до інтернету. Схему підключення я зобразив на малюнку 3.5

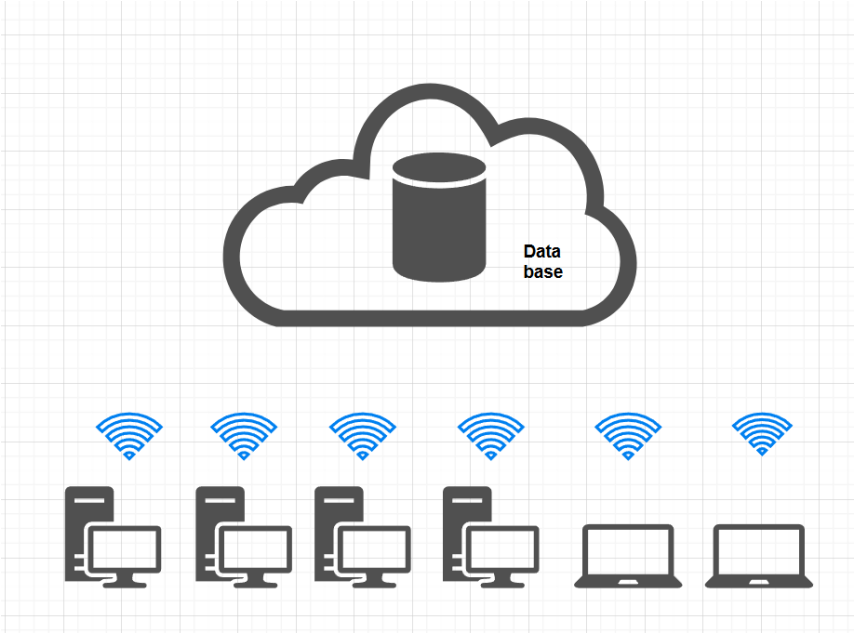


Рис. 3.5 - схема забезпечення синхронізації з сервером

Джерело: сформовано автором на основі виконаного дослідження

ПК дозволяє використовувати повноцінний клавіатурно-мишковий або геймпадовий інтерфейс**,** що значно розширює можливості управління гравцем. Для платформерів точність керування має критичне значення, а саме комп'ютерна платформа дозволяє забезпечити як високу чутливість введення, так і стабільний фреймрейт у межах необхідної продуктивності.

Крім того, розробка та тестування гри на ПК є технічно доступною — не потребує ліцензій розробника для мобільних платформ, не передбачає складної системи сертифікації або обмежень, як у випадку з консолями. Це дозволяє зосередитись безпосередньо на геймдизайні, механіках і якості реалізації, що є особливо важливим у навчальному контексті.

Таким чином, цільовою операційною системою для запуску гри є Windows 10/11, яка забезпечує сумісність із рушієм, драйверами, API, а також комфортну взаємодію кінцевого користувача з продуктом.

Моя гра не потребує дуже потужного пристрою для того щоб вона працювала добре. Через своє low poly оформлення вона може видавати найкращу продуктивність. Хочу зазначити, що якщо гравець хоче отримати найякіснішу

графіку він має мати досить потужний комп’ютер. Комплектуючими цього пристрою можуть бути:

– (RAM): мінімум 16 GB;

– Процесор (CPU): 6+ ядер з хорошою тактовою частотою – наприклад, Ryzen 5 3600X;

– Графічний процесор (GPU): дискретна відеокарта з ≥4 GB VRAM. Наприклад Nvidia RTX 3050;

– Накопичувач (SSD): NVMe SSD об’ємом ≥1 TB.

Такий пристрій забезпечить найкращу і найкомфортнішу якість гри.

3.3 Програмне забезпечення

Програми, які були використані в розробці гри:

– Unity

– Blender

– Visual studio

– Photoshop

Unity

Unity — це потужне середовище розробки, яке зарекомендувало себе як один із найзручніших та найбільш інтуїтивно зрозумілих інструментів для створення комп’ютерних ігор. Однією з ключових переваг цього рушія є його низький поріг входу, що робить його ідеальним вибором для студентських, навчальних та інді-проєктів. Unity дозволяє швидко перейти від ідеї до робочого прототипу завдяки гнучкому інтерфейсу, наявності вбудованих шаблонів та великій кількості готових компонентів.

Редактор Unity підтримує режим візуального редагування сцен, що значно пришвидшує побудову ігрового світу та дозволяє працювати з об’єктами без

необхідності писати додатковий код. Це особливо важливо для проектів із великою кількістю 3D-моделей, UI-компонентів та інтерактивних елементів.

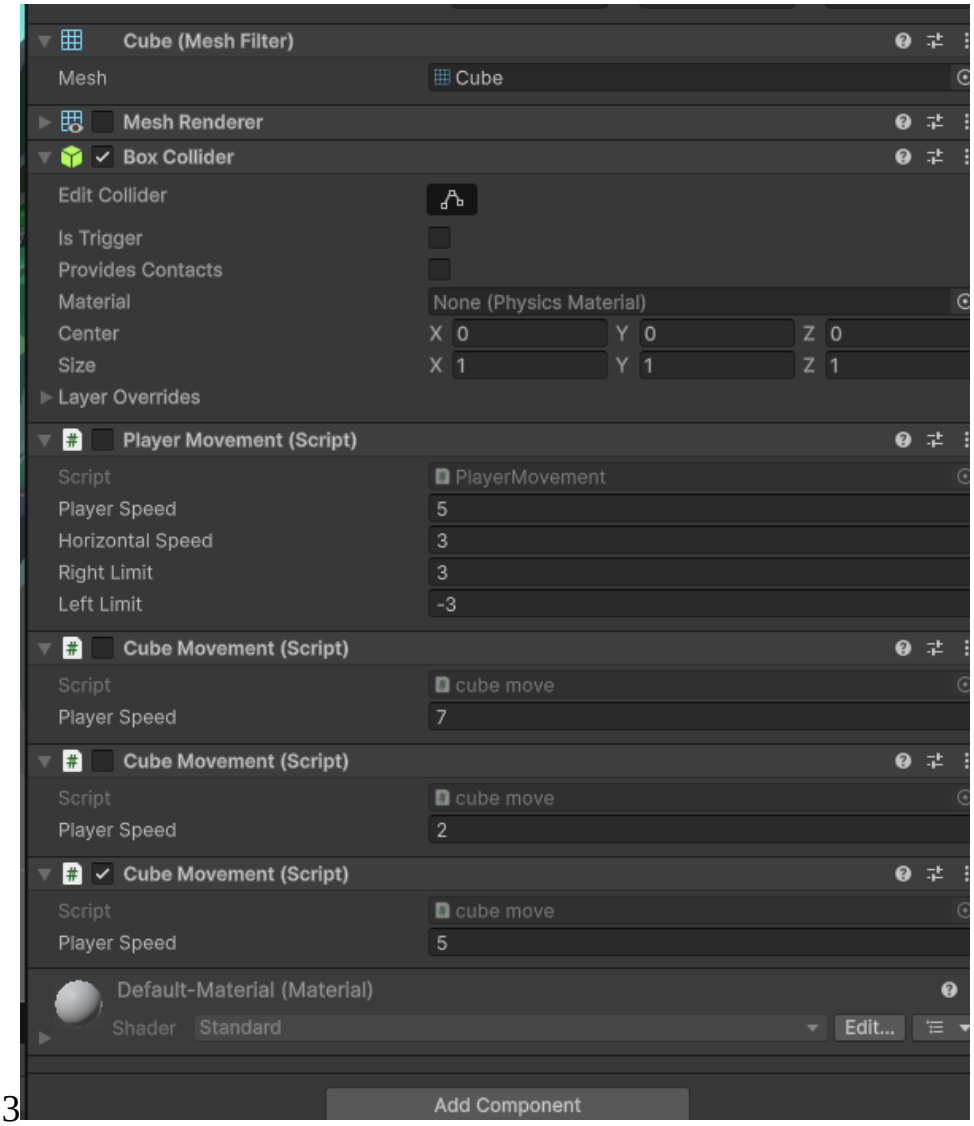
Крім того, Unity має зручну інтеграцію з мовою програмування C#, яка широко застосовується у розробці. Це забезпечує просту реалізацію логіки гри, обробки подій та керування анімаціями.

Ще однією перевагою є велика кількість навчальних матеріалів, відеоуроків, форумів і документації, що суттєво спрощує вирішення будь-яких технічних труднощів під час розробки.

Unity також підтримує вбудоване тестування та миттєве відлагодження гри без необхідності її компіляції щоразу. Це значно прискорює цикл розробки, особливо під час роботи з механікою персонажа, фізикою або UI-елементами. Більшість цих переваг я використовував у своєму проєкті і навів їх на рис. 3.6 –3.8.

Рис. 3.6 – відображення кількості зібраних монеток на UI

Джерело: розроблено автором самостійно



3

Рис. 3.7 – Реалізація генерації нових сегментів сцени в юніті

Джерело: розроблено автором самостійно

Visual studio

Насамперед, Visual Studio забезпечує повноцінну інтеграцію з Unity через плагін Visual Studio Tools for Unity, що дозволяє автоматично підключатися до проекту, працювати з автозаповненням методів API Unity, автоматично розпізнавати компоненти сцени та відображати помилки компіляції в реальному часі. Це значно знижує ймовірність синтаксичних і логічних помилок у коді.

Інтерфейс Visual Studio є інтуїтивно зрозумілим і дозволяє зручно працювати навіть над великим обсягом скриптів. Для low poly гри-платформера, де основна логіка пов’язана з керуванням персонажем, зіткненнями, збиранням об’єктів і

оновленням інтерфейсу, можливість структурованого управління класами та методами є вкрай важливою.

Visual Studio також надає розширені можливості для відлагодження (debugging). Розробник може встановлювати брейкпоінти, переглядати змінні в реальному часі, контролювати виконання коду покроково, що дозволяє швидко знаходити і виправляти помилки у геймплейній логіці або при роботі з даними (наприклад, у JSON-базі).

Окрім цього, Visual Studio підтримує розширення та плагіни, які допомагають автоматизувати частину роботи — від форматування коду до інтеграції з системами контролю версій (Git), що особливо корисно при роботі над більшими або спільними проєктами.

Код, який я писав задля цього проєкту в Visual Studio я розмістив на малюнках (3.8 – 3.12).

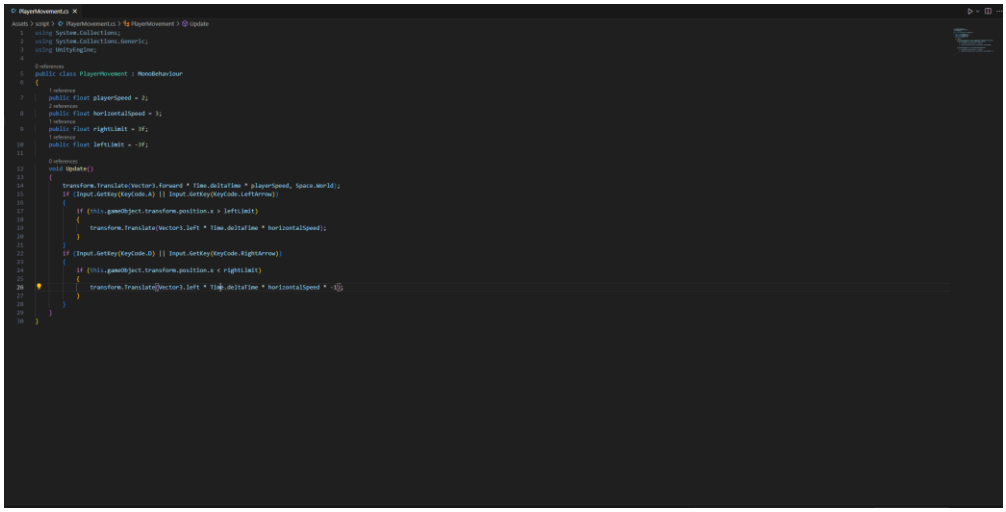


Рис. 3.8 – Реалізація коду для руху та обмеження з усіх боків сцени

Джерело: розроблено автором самостійно

Assets > script > SegmentGenerator.cs > ...

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

0 references
public class SegmentGenerator : MonoBehaviour
{
    1 reference
    public GameObject[] segmentPrefabs;
    3 references
    public Transform spawnPoint;
    1 reference
    public float delay = 10f;

    0 references
    void Start()
{
    if (spawnPoint == null)
    {
        Debug.LogError("Spawn Point не призначено!");
        return;
    }
    StartCoroutine(SegmentGen());
}
    1 reference
    IEnumerator SegmentGen()
    {
        foreach (GameObject segment in segmentPrefabs)
        {
            yield return new WaitForSeconds(delay);
            Instantiate(segment, spawnPoint.position, Quaternion.identity);
            spawnPoint.position += new Vector3(0, 0, 10); // рух точку вперед
        }
    }
}
```

Рис. 3.9 – Реалізація коду для генерації сегментів

Джерело: розроблено автором самостійно

Assets > script > rotation.cs > ...

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

0 references
public class CollectableRotate : MonoBehaviour
{

    1 reference
    [SerializeField] int rotateSpeed = 1;
    0 references
    void Update()
    {
        transform.Rotate(0, rotateSpeed, 0, Space.World);
    }
}
```

Рис. 3.10 – Реалізація коду для зацикленого повороту монеток

Джерело: розроблено автором самостійно

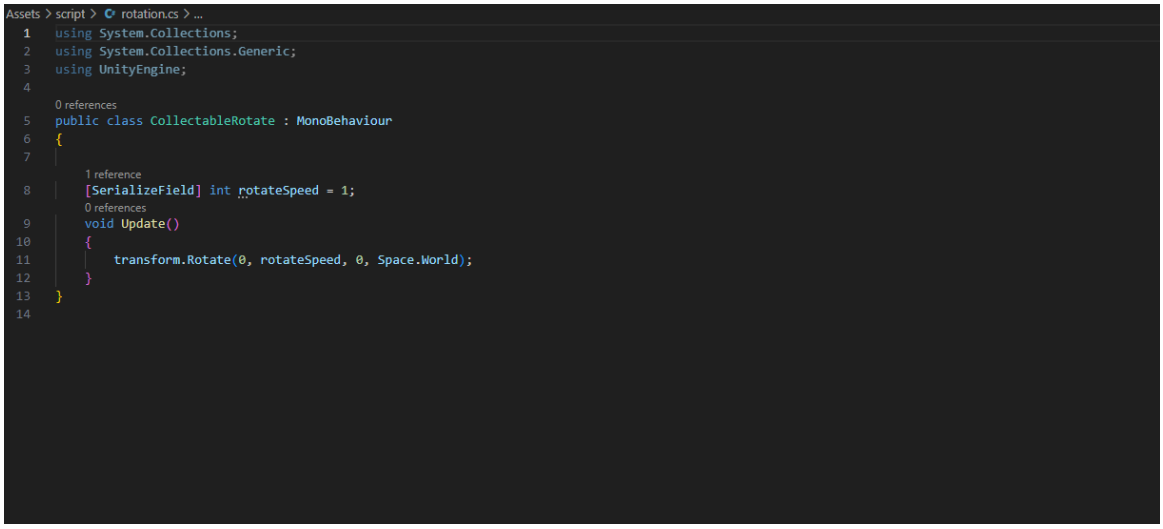
```
Assets > script > CoinsCount.cs > ...
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class MasterInfo : MonoBehaviour
{
    1 reference
    public static int coinCount = 0;
    1 reference
    [SerializeField] GameObject coinDisplay;

    0 references
    void Update()
    {
        coinDisplay.GetComponent<TMPro.TMP_Text>().text = "COINS: " + coinCount;
    }
}
```

Рис. 3.11 – Реалізація лічильнику монеток

Джерело: розроблено автором самостійно



```
Assets > script > rotation.cs > ...
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class CollectableRotate : MonoBehaviour
{

    1 reference
    [SerializeField] int rotateSpeed = 1;
    0 references
    void Update()
    {
        transform.Rotate(0, rotateSpeed, 0, Space.World);
    }
}
```

Рис. 3.12 – Реалізація коду для зацикленого повороту монеток

Джерело: розроблено автором самостійно

Blender

У межах розробки 3D гри-платформера важливу роль відіграє інструментарій для створення тривимірних моделей персонажів, оточення та об’єктів взаємодії. Для цього проєкту було обрано Blender — багатофункціональний програмний пакет для 3D-моделювання, анімації та текстурування. Такий вибір зумовлений не

лише широкими функціональними можливостями, а й високим рівнем сумісності з ігровими рушіями, зокрема Unity.

Blender є повністю безкоштовним програмним забезпеченням з відкритим кодом, що дозволяє використовувати його у будь-яких навчальних, творчих або комерційних проєктах без обмежень. Це особливо важливо в умовах індивідуального або студентського розроблення, де питання ліцензування може бути критичним фактором.

Однією з головних переваг Blender є підтримка low poly моделювання, що ідеально підходить для стилістики обраної гри. Низькополігональні моделі мають менший об’єм геометрії, що знижує навантаження на графічний процесор, прискорює рендеринг і забезпечує стабільну продуктивність у реальному часі. Blender надає інструменти для точного контролю кількості полігонів, оптимізації сіток та створення моделей із чіткими силуетами.

Окрім моделювання, Blender підтримує створення анімацій скелетного типу (rigging), що дозволяє гнучко управляти рухами персонажів. Це важливо для платформера, де ігрова динаміка напряму залежить від якості анімації — стрибків, бігу, зіткнень тощо. Отримані анімації легко експортуються у формати, сумісні з Unity (наприклад FBX або glTF).

Також серед переваг слід виділити розвинену спільноту, документацію, навчальні матеріали та постійну підтримку з боку розробників. Це дозволяє швидко опанувати необхідні функції та реалізувати візуально якісний контент навіть без великого досвіду.

Таким чином, Blender є оптимальним вибором як з точки зору функціоналу, так і доступності для створення моделей, анімацій і візуального контенту у рамках розробки 3D гри-платформера.

Для свого проєкту я вирішив розробити усі моделі самостійно. Задля оптимізації ресурсу я не створював UV розгортку, я створював їх в однотонних кольорах. Свої моделі я представив на рисунках 3.13 – 3.18.

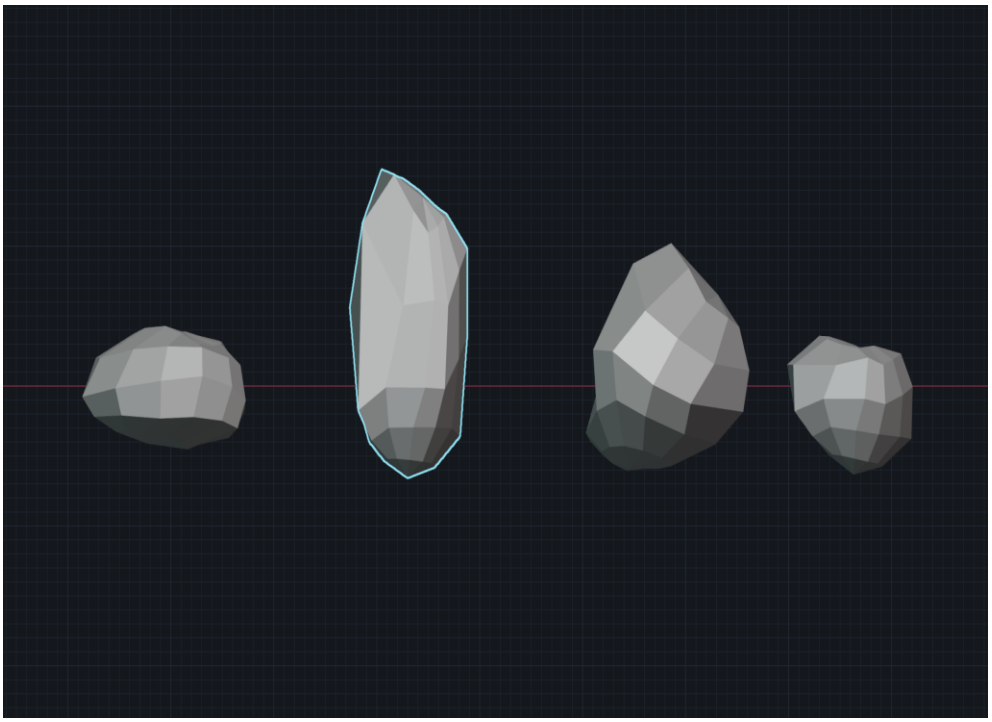


Рис. 3.13 - Різновиди моделей каменів

Джерело: розроблено автором самостійно

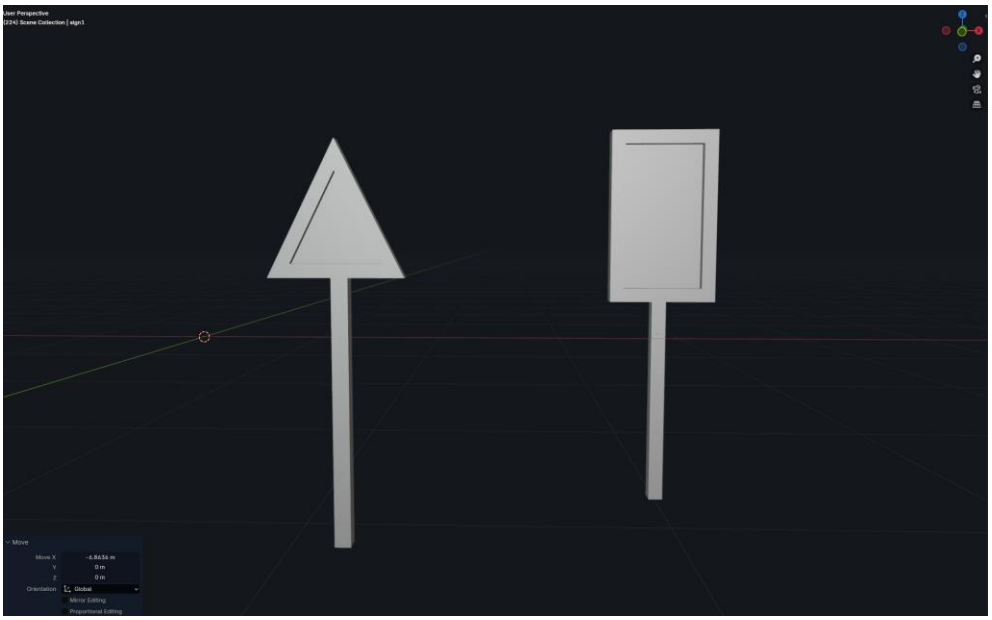


Рис. 3.14 – різновиди моделей знаків

Джерело: розроблено автором самостійно

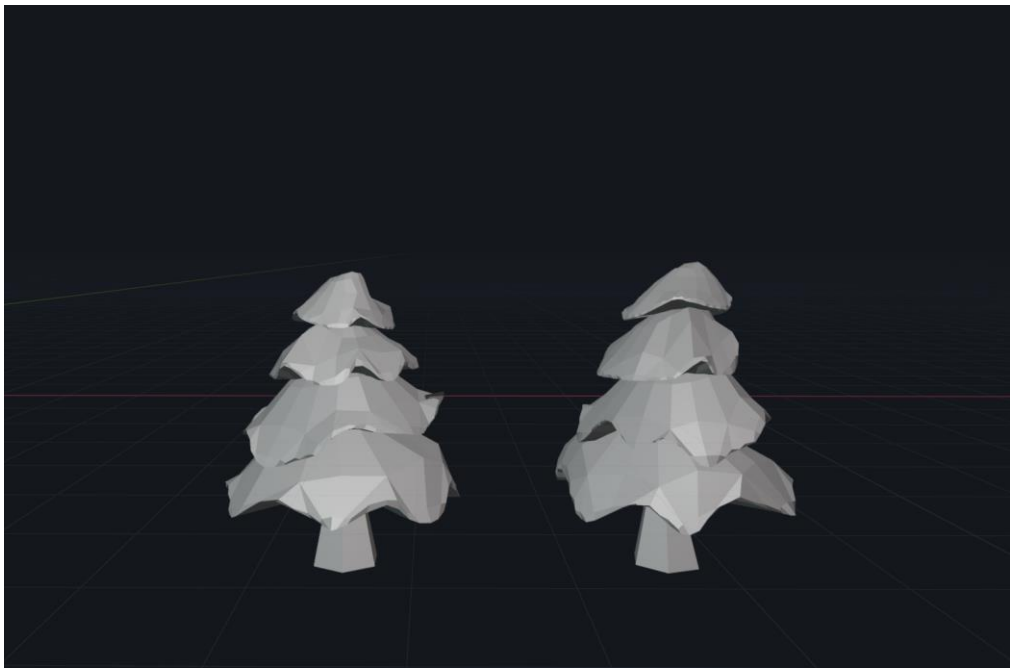


Рис. 3.15 – варіанти моделей ялинок

Джерело: розроблено автором самостійно

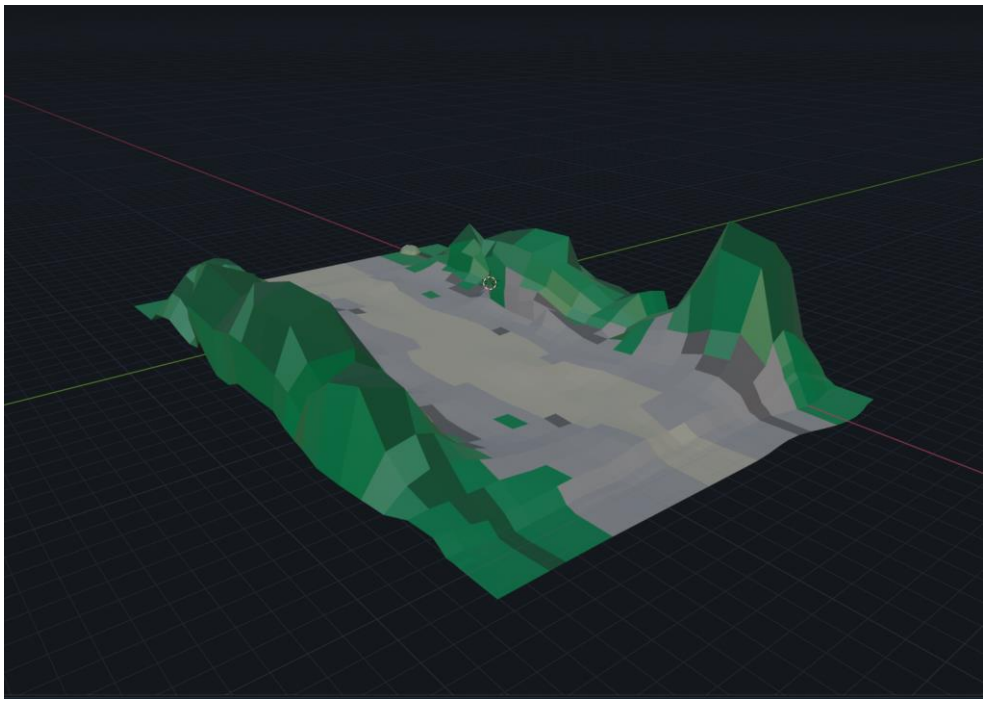


Рис. 3.16 – основна локація, по якій буде бігти дівчинка

Джерело: розроблено автором самостійно

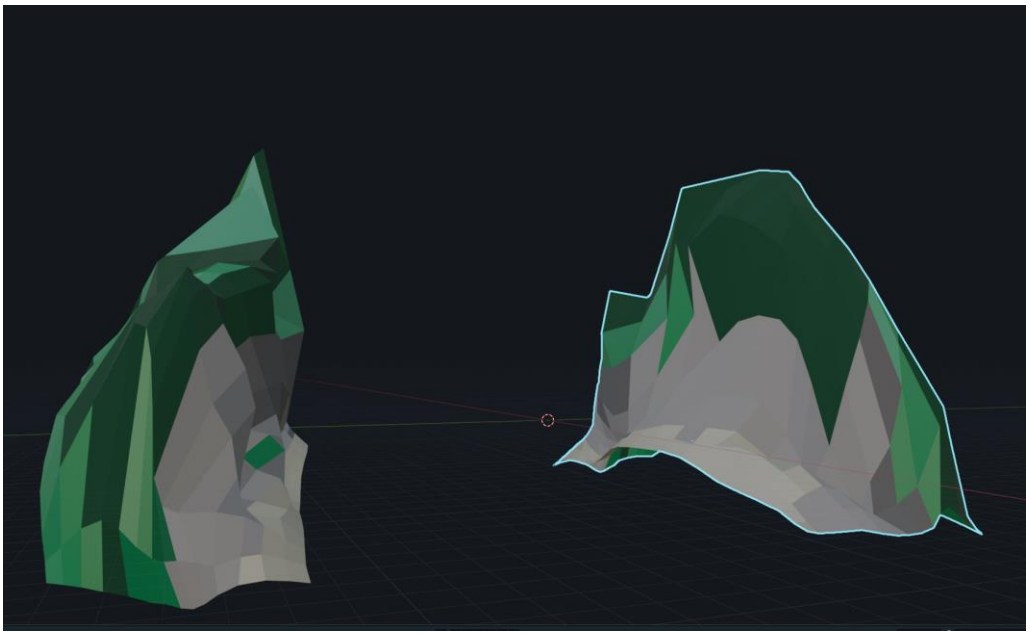


Рис. 3.17 – гори, які слугають заповненням фону гри

Джерело: розроблено автором самостійно

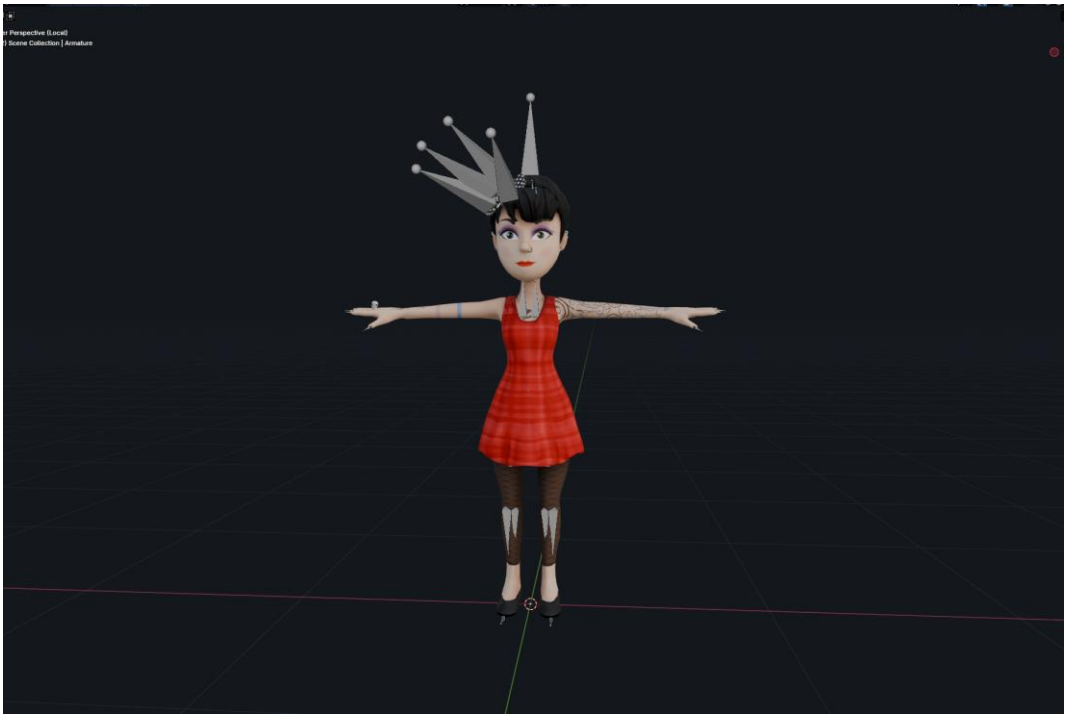


Рис. 3.18 – імпорт скелетної структури, яка відповідає основним частинам тіла персонажа із сайту “Mixamo”

Джерело[10]

Photoshop

У процесі розробки гри-платформера для оформлення інтерфейсу користувача було використано графічний редактор Adobe Photoshop. Цей інструмент дозволив створити візуально привабливі та функціональні елементи, що відповідають загальному стилю гри та забезпечують зручність взаємодії гравця з ігровим середовищем.

Зокрема, за допомогою Photoshop було розроблено дизайн екрана логіну — одного з перших елементів, з яким взаємодіє користувач при запуску гри. У графічному редакторі були створені такі компоненти, як фонова ілюстрація, логотип гри, вікна введення (input fields), кнопки входу та реєстрації. Усі елементи було витримано в стилістиці low poly та підібрано відповідно до колірної гами, характерної для гірського середовища, де відбуваються події гри.

Photoshop забезпечив високу точність позиціонування, якісну обробку шрифтів та ефектів (наприклад, тіні, підсвічування), що дозволило створити естетично завершений макет інтерфейсу. Створений дизайн було експортовано у вигляді графічних ресурсів (PNG), які згодом імпортовано до Unity для інтеграції у сцену логіну через UI-систему рушія.

Завдяки використанню Adobe Photoshop вдалося досягти візуальної цілісності та професійного вигляду стартового екрана гри, що суттєво покращує перше враження гравця та загальну якість проєкту.

3.4 Результат реалізації гри платформера

Уся реалізація базується на попередньо визначених вимогах та спроєктованій архітектурі, що дозволило побудувати модульну, гнучку та розширювану систему. Основна увага приділяється логіці ігрового процесу, керуванню подіями, взаємодії з UI, а також забезпеченню продуктивності та стабільності гри.

Результат повної реалізації гри представлено на рис. 3.19 – 3.20.

Рис 3.19 – головне меню гри

Джерело: розроблено автором самостійно

На цьому рисунку представлено головне меню, яке бачить гравець при заході у гру. З цього меню він може почати саму гру. Перейти в магазин, купити новий костюм і там його змінити. А у вкладці Leaderboard він може побачити свій рейтинг серед інших гравців, якщо він підключений до мережі. Хочу зазначити, що шрифти використані в цьому меню були взяті з сайту “Font Desk”[30].

Рис 3.20 – основний геймплей гри

Джерело: розроблено автором самостійно

На цьому малюнку як проходить основний геймплей гри. Гравець має зібрати монетки і число монеток він може побачити на вікні з права. При зіткненні з об’єктами гра закінчиться і кількість монеток нарахується на ігровий.

ВИСНОВКИ

У ході реалізації 3D гри у жанрі платформера на базі рушія Unity було продемонстровано здатність до самостійного проєктування ігрових механік, опанування сучасних інструментів розробки, а також підвищено практичні навички програмування, моделювання та інтеграції мультимедійного контенту.

Протягом виконання роботи було вирішено низку ключових завдань, а саме:

– сформульовано вимоги до функціональності та інтерфейсу майбутнього програмного продукту;

– обґрунтовано вибір програмного забезпечення та технологій, зокрема Unity і Blender;

– спроєктовано інформаційну та функціональну структуру гри;

– реалізовано архітектуру гри, модулі керування, механіки взаємодії та руху;

– створено дизайн UI-інтерфейсу з урахуванням потреб користувача;

– побудовано прототип ігрової сцени, що включає графіку, анімації та звук;

– здійснено тестування гри з метою перевірки коректності логіки, продуктивності та стабільності.

Практична цінність створеного продукту полягає у тому, що розроблена гра може використовуватися як інструмент дозвілля, розвитку моторики та швидкості реакції. Вона має інтуїтивно зрозумілий інтерфейс, яскравий візуальний.

У теоретичній частині (першому розділі) було розглянуто актуальні технології створення 3D-ігор, проаналізовано рушії та інструменти моделювання, обґрунтовано вибір Unity як основної платформи. У другому розділі визначено функціональні та нефункціональні вимоги, структуру гри, її архітектурні компоненти та принципи організації логіки. Третій розділ присвячено технічній

реалізації — опису компонентів, елементів інтерфейсу, ігрових механік, а також методам тестування і відлагодження.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Перелік найпопулярніших ігор платформерів.
URL: [https://www.playstation.com/uk-ua/editorial/best-3d-platformers/]

2. Класифікація платформерів:
URL: [https://youtu.be/VEnZzwW_OJM?si=wVqaF9W34_O6DAn4]

3. Офіційна документація Unity: "Creating scripts" — розділ мануалу, що описує створення, іменування та опис C#-скриптів у Unity. URL:[https://docs.unity3d.com/6000.1/Documentation/Manual/creating-scripts.html]

4. "Learn C# Scripting for Unity in 15 Minutes" (YouTube відео) — швидкий курс з основних аспектів C# у контексті Unity: керування персонажем, UI, ввод користувача. URL:[https://docs.unity3d.com/6000.1/Documentation/Manual/creating-scripts.html]

5. Learn C# Programming Basics for Unity Game Development – Part 1. URL: [https://youtu.be/TgqAqniukTg?si=5yaMPv_Tici7gnxA]

6. Reddit-обговорення про ресурси для вивчення C# і Unity: https://www.reddit.com/r/csharp/comments/con0d7/best_tutorial_to_learn_c_for_unity/?

7. "Create with Code" URL: https://learn.unity.com/course/create-with-code

8. Unity AssetStore .
URL:[https://assetstore.unity.com/?srsltid=AfmBOooO33vnicGLEH_wzUSOTs83_KVB1oxa-gND701BCdssWNEnB1Gb]

9. Mixamo: [https://www.mixamo.com/#/]

10. Відео, для покращення навички використання програмного забезпечення Blender:
[https://www.youtube.com/watch?v=QPh8h0hWkg0]

11. Instructables, "Creating Low Poly 3D Models Using Blender 2.8". URL: https://youtu.be/Z8sg0nHNTTo?si=2f5R8ILWhO2pqTuh

12. Draw.io. URL: https://app.diagrams.net/

13. Instructables, "Creating Low Poly 3D Models Using Blender 2.8. URL: https://youtu.be/1jHUY3qoBu8?si=lDo0AfGedPsSnKRD

14. Packt Publishing, Low Poly 3D Modeling in Blender. URL: https://github.com/PacktPublishing/Low-Poly-3D-Modeling-in-Blender?

15. PubNub, "Technical Overview of Unity Game Engine" URL: https://www.pubnub.com/guides/unity/?utm_source=chatgpt.com

16. Hussain et al. "Unity Game Development Engine: A Technical Survey". URL: https://github.com/PacktPublishing/Low-Poly-3D-Modeling-in-Blender?

17. DiVA portal, An Analysis of Platform Game Design. URL: https://www.diva-portal.org/smash/get/diva2%3A728079/FULLTEXT01.pdf?utm_source=chatgpt.com

18. Zamri et al., “The Effects of 10 User Interface (UI)” URL: https://www.researchgate.net/publication/354527166_The_Effect_Of_10_User_Interface_UI_Elements_On_Game_Design_Process?

19. eCampusOntario, "User interface – Game Design & Development" URL: https://ecampusontario.pressbooks.pub/gamedesigndevelopmenttextbook/chapter/user-interface/

20. Journal JETIR, "Influence Of UI/UX In Gaming. URL: https://www.jetir.org/papers/JETIR2308119.pdf?

21. Summerville et al., Procedural Content Generation via Machine Learning URL: https://arxiv.org/abs/1702.00539

22. Aramini et al., "An Integrated Framework for AI Assisted Level Design in 2D Platformers" URL: https://arxiv.org/abs/1804.09153?

23. Красюк О.Ю. Основи 3D-моделювання в Blender. — Одеса: ОНУ, 2022. — 198 с.

24. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson Education.URL: https://www.auhd.edu.ye/upfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf

25. Michael Hayter, «Unity & SQLite: Quick Start Guide (Part-1)» Джерело:

URL: https://michaelhayter.medium.com/unity-sqlite-quick-start-guide-part-1-8cba7ed22b9a

26. Програмування на C# для ігор. URL: https://learn.microsoft.com/en-us/dotnet/csharp/

27. Підручник з 3D-анімації для Blender. URL.https://docs.blender.org/manual/en/latest/

28. Лабораторний практикум з геймдизайну / за ред. І.І. Ковальчука. — Івано-Франківськ: НТУ, 2021. — 104 с.

29. Schell, J. (2019). *The Art of Game Design: A Book of Lenses* (3rd ed.). CRC Press. URL: https://www.inventoridigiochi.it/wp-content/uploads/2020/07/art-of-game-design.pdf

30. Stack Overflow, коментар до Unity + SQLite. URL: https://stackoverflow.com/questions/29935508/how-to-create-sqlite-database-in-unity-c

ДОДАТКИ

ДОДАТОК А

Головні менханіки:

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

public class PlayerMovement : MonoBehaviour
{
   public float playerSpeed = 2;
   public float horizontalSpeed = 3;
   public float rightLimit = 3f;
   public float leftLimit = -3f;

   void Update()
   {
      transform.Translate(Vector3.forward * Time.deltaTime * playerSpeed,
Space.World);
      if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))
      {
         if (this.gameObject.transform.position.x > leftLimit)
         {
            transform.Translate(Vector3.left * Time.deltaTime * horizontalSpeed);
         }
      }
      if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))
      {
         if (this.gameObject.transform.position.x < rightLimit)
         {
            transform.Translate(Vector3.left * Time.deltaTime * horizontalSpeed * -
1);
         }
      }
   }
}

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

public class SegmentGenerator : MonoBehaviour
{
   public GameObject[] segmentPrefabs;
   public Transform spawnPoint;
   public float delay = 10f;
```

```
  void Start()
{
   if (spawnPoint == null)
   {
      Debug.LogError("Spawn Point не призначено!");
      return;
   }
   StartCoroutine(SegmentGen());
}
   IEnumerator SegmentGen()
   {
      foreach (GameObject segment in segmentPrefabs)
      {
         yield return new WaitForSeconds(delay);
         Instantiate(segment, spawnPoint.position, Quaternion.identity);
         spawnPoint.position += new Vector3(0, 0, 10); // рух точку вперед
      }
   }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollectableRotate : MonoBehaviour
{

   [SerializeField] int rotateSpeed = 1;
   void Update()
   {
      transform.Rotate(0, rotateSpeed, 0, Space.World);
   }
}

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

public class MasterInfo : MonoBehaviour
{
   public static int coinCount = 0;
   [SerializeField] GameObject coinDisplay;

   void Update()
```

```
    {
        coinDisplay.GetComponent<TMPro.TMP_Text>().text = "COINS: " +
coinCount;
    }
}

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

public class SegmentGenerator : MonoBehaviour
{
    public GameObject[] segmentPrefabs;
    public Transform spawnPoint;
    public float delay = 10f;

  void Start()
{
    if (spawnPoint == null)
    {
        Debug.LogError("Spawn Point не призначено!");
        return;
    }
    StartCoroutine(SegmentGen());
}
    IEnumerator SegmentGen()
    {
        foreach (GameObject segment in segmentPrefabs)
        {
            yield return new WaitForSeconds(delay);
            Instantiate(segment, spawnPoint.position, Quaternion.identity);
            spawnPoint.position += new Vector3(0, 0, 10); // рух точку вперед
        }
    }
}
```

ДОДАТОК Б

StrikePlagiarism.com

Дата звіту 6/15/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kyiv National Economic University named after Vadym Hetman KNEU
Заголовок
Розробка 3Д гри платформер
Автор
Зденик Максим Володимирович
Науковий керівник / Експерт
Шевченко К.Л
підрозділ
кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.27% 1.27% КП 1

0.47% 0.47% КП 2

0.43% 0.43% КЦ

25
Довжина фрази для коефіцієнта подібності 2

9082
Кількість слів

68154
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	4

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/server/api/core/bitstreams/1b35cb3a-b293-4677-9674-83983838c914/content	43 0.47 %
2	https://ir.kneu.edu.ua/server/api/core/bitstreams/1b35cb3a-b293-4677-9674-83983838c914/content	20 0.22 %
3	КПІ_2025_123_Зданович! [redacted] 6/10/2025 Ukrainian national aviation university (Криворізький Фаховий коледж)	19 0.21 %

4	Тищенко.Б.Є, 242,1 ВКБР Розробка мобільної гри на базі Unreal Engine н.кер. Букатов Д.В. [redacted] 5/9/2025 European University (European University)	10 0.11 %
5	ФКНТ_2023_123м_Мікульський_ВВ [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	10 0.11 %
6	ООП 2024 КН22 Ковалик Є.С [redacted] 11/28/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	8 0.09 %
7	ООП 2024 КН22 Ковалик Є.С [redacted] 11/28/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	5 0.06 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.57 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	КПІ_2025_123_Зданович! [redacted] 6/10/2025 Ukrainian national aviation university (Криворізький Фаховий коледж)	19 (1) 0.21 %
2	ООП 2024 КН22 Ковалик Є.С [redacted] 11/28/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	13 (2) 0.14 %
3	Тищенко.Б.Є, 242,1 ВКБР Розробка мобільної гри на базі Unreal Engine н.кер. Букатов Д.В. [redacted] 5/9/2025 European University (European University)	10 (1) 0.11 %
4	ФКНТ_2023_123м_Мікульський_ВВ [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	10 (1) 0.11 %

з Інтернету (0.69 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/server/api/core/bitstreams/1b35cb3a-b293-4677-9674-83983838c914/content	63 (2) 0.69 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)