Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Швеця Нікіти Олександровича*
(ПІБ)

академічної групи *122-19-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення ігрового додатку за допомогою Unity*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Мещеряков Л.І.*			
розділів:				
спеціальний	*проф. Мещеряков Л.І.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-2* (група) *Швеця Нікіти Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення ігрового додатку за допомогою Unity*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *проф. Мещеряков Л.І.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Швець Н.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12,06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 63 с., 23 рис., 3 дод., 20 джерел.

Об'єкт розробки: «Розробка програмного забезпечення ігрового додатку за допомогою Unity».

Мета кваліфікаційної роботи: створити ігровий додаток, який сприятиме розвитку просторового мислення і надасть розуміння нетипової поведінки у грі.

У вступі проведений аналіз проблеми та поточний стан, уточнена мета роботи та галузь її застосування, доведена актуальність теми та постановлення завдання.

У першому розділі здійснений аналіз предметної галузі, визначена актуальність завдання та призначення розробки, сформульовано постановку завдання, вказані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрані платформи для розробки, проведено проектування та розробку програми, описано роботу програми, алгоритм та структура її функціонування, а також процес виклику та завантаження програми, визначено вхідні та вихідні дані, описано параметри технічних засобів.

У економічному розділі визначено трудомісткість розробленої інформаційної системи, розраховано вартість роботи зі створення програми та визначено час, необхідний для її створення.

Практичне значення полягає у створенні програмного додатка, який забезпечує можливість проводити час з задоволенням.

Актуальність розробленого програмного продукту полягає в створенні додатка, який буде популярним серед цільової аудиторії - гравців відеоігор.

Список ключових слів: КОМП'ЮТЕР, ІГРОВИЙ ДВИГУН, UNITY, АЛГОРИТМ, ІНФОРМАЦІЙНА СИСТЕМА, WINDOWS, МЕНЮ, ВКЛАДКА, ДОДАТОК.

ABSTRACT

Explanatory note: 62 p., 23 figures, 3 appendix, 20 sources.

Object of development: "Development of game application software using Unity".

The purpose of the qualification work: to create a game application that will promote the development of spatial thinking and provide an understanding of atypical behavior in the game.

The introduction analyzes the problem and the current state, specifies the purpose of the work and its scope, and proves the relevance of the topic and the task.

The first section analyzes the subject area, determines the relevance of the task and the purpose of the development, formulates the task statement, and specifies the requirements for software implementation, technologies, and software tools.

The second section analyzes existing solutions, selects development platforms, designs and develops the program, describes the program operation, algorithm and structure of its functioning, as well as the process of calling and loading the program, defines input and output data, and describes the parameters of technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of creating the program is calculated, and the time required for its creation is determined.

The practical significance is to create a software application that provides an opportunity to spend time with pleasure.

The relevance of the developed software product is to create an application that will be popular among the target audience - video game players.

List of keywords: COMPUTER, GAME ENGINE, UNITY, ALGORITHM, INFORMATION SYSTEM, WINDOWS, MENU, TAB, APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – персональний комп’ютер;

3D — three-dimensional;

MVC — Model-View-Controller;

IDE — Integrated Development Environment;

SSAO — Screen space ambient occlusion;

SDK — Software development kit;

GUI — graphical user interface;

VS — Visual Studio

PS — PlayStation;

PSP — PlayStation Portable;

OS — Operation System;

Mac OS — Macintosh Operating System;

UE — Unity Engine.

ВСТУП

Дана кваліфікаційна робота спрямована на вивчення та створення ігрового додатку в жанрі Arcade з використанням Unity-движка. Вона пов'язана зі спеціалізацією "Комп'ютерні науки" і відповідає загальним тематичним вимогам до кваліфікаційних робіт, а також переліку виробничих функцій, типових завдань та навичок, необхідних для бакалаврів цієї галузі.

Метою цієї роботи є вивчення засобів роботи з різними підсистемами операційної системи, зокрема подій вводу в системі та взаємодії з ними. Ці навички є важливими для розробки програмного забезпечення, яке контролює доступ та відстежує активність процесів і користувачів. Крім того, вони допоможуть краще розуміти механізми роботи з графічними об'єктами та їх взаємодію на низькому рівні.

Ця робота також дозволить ознайомитись з методами розробки ігрових додатків та розумінням багатовимірних ігрових процесів.

Однією з основних вимог до розробленого програмного забезпечення є стабільність його роботи. Unity Engine є вимогливим до графічного процесора, тому швидкість роботи додатку на менш потужних комп'ютерах може бути незадовільною.

Отже, метою цієї кваліфікаційної роботи є проектування та розробка захоплюючого ігрового додатку у жанрі Arcade для гравців різний категорій.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Аркадні ігри вважаються одним з найпоширеніших жанрів відеоігор і мають довгу історію. Вони з'явилися ще в 1970-х роках, коли були створені перші аркадні автомати з іграми, такими як Galaxy Game, Pong та [1]. З тих пір аркадні ігри постійно еволюціонували, а нові технології дозволили розробникам створювати все більш складні та захоплюючі геймплеї. На рис. 1.1. можна побачити одну із перших аркадних ігор.

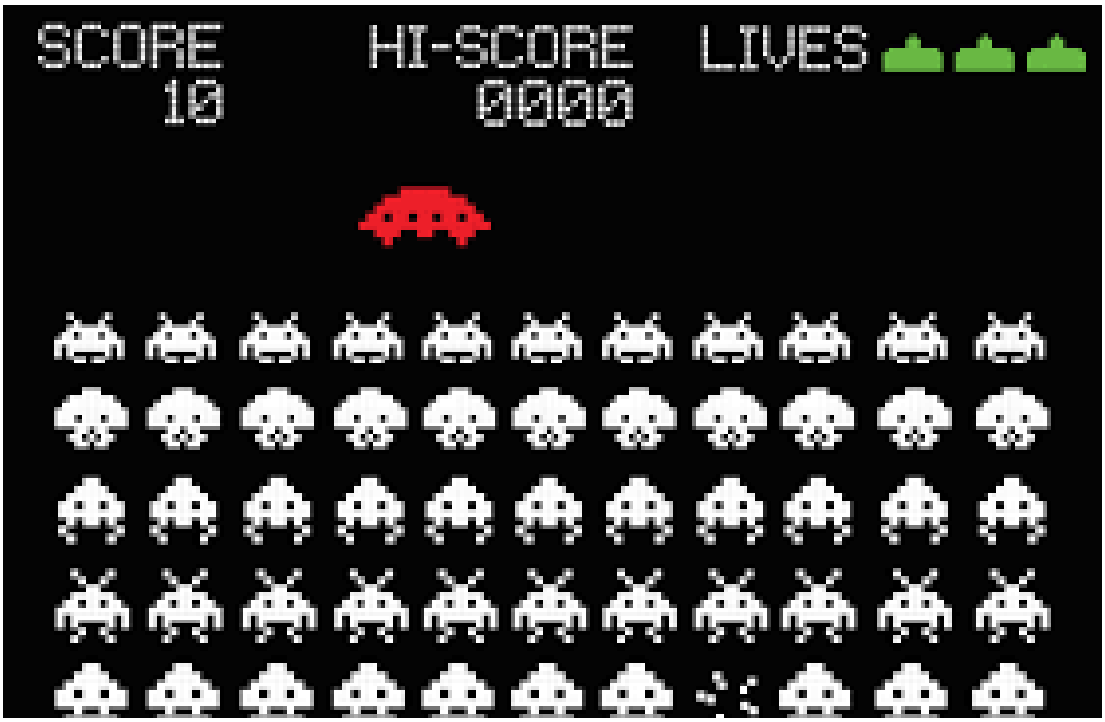

Рис. 1.1. Гра Space Invaders

Однією з найпоширеніших ігор є Subway Surfers, рис. 1.2., і вона є аркадною грою. На початку 2023 року гру завантажили понад 4 млрд. разів. Розробники похвалилися тим, що щодня в Subway Surfers грають понад 20 млн користувачів, а на місяць гру відвідують понад 150 млн геймерів.

Нагадаємо, вперше аркаду презентували 24 травня 2012 року дві данські компанії Kiloo і SYBO, тобто через два місяці грі виповнюється рівно 11 років. За весь цей час у грі практично нічого не змінилося: все та ж механіка і ті ж перешкоди. Змінюються тільки якість графіки, стиль оточення і скіни для бігунів. [2]

Рис. 1.2. Гра Subway Surfers

Сутність аркадних ігор полягає у виконанні швидких та веселих завдань, які часто вимагають від гравця вміння реагувати швидко та точно. Геймплей у таких іграх зазвичай базується на простих механіках, таких як стріляння, уникання перешкод або збирання предметів. Важливим елементом аркадних ігор є тактильність та відчуття контролю, яке гравець отримує під час гри.

Завдяки двигуну Unity, розробники мають доступ до потужних інструментів та ресурсів для створення аркадних ігор. Unity надає зручне середовище для розробки, яке включає в себе графічний редактор, систему фізики, інструменти анімації та скриптування, що дозволяє створювати складні механіки гри та реалістичну взаємодію об'єктів.

Історія двигуна, все почалося з розробки гри. Першим продуктом, що випустила Unity Technologies, був не ігровий движок, а гра. У 2005 році

засновники Unity, Йоахім Анте, Девід Хелгасон і Ніколас Френсіс, випустили відеогру для MacOS, через рік після створення своєї компанії, яка тоді називалася Over the Edge Entertainment. Гра була створена на движку, який вони розробили з нуля, з наміром ліцензувати движок іншим розробникам. [3]

Важливою частиною розробки аркадних ігор є дизайн рівнів. Розробники створюють рівні з відповідними викликами та перешкодами, які спонукають гравця до швидких реакцій та стратегічного мислення. Гнучкість Unity дозволяє створювати різноманітні рівні з унікальними геймплейними елементами.

Новизна результатів є важливим аспектом розробки аркадних ігор на двигуні Unity. Розробники можуть експериментувати з новими ідеями та механіками, використовуючи різноманітні функції Unity для створення унікального геймплею. Це дозволяє розробникам вирізнятися серед конкуренції та привертати увагу гравців.

У сучасному світі існує велика кількість аркадних ігор на різних платформах, включаючи комп'ютери, консолі та мобільні пристрої. Багато з них розроблені з використанням двигуна Unity, оскільки цей двигун надає розробникам гнучкість та потужність для створення відмінних ігрових досвідів.

Проте, незважаючи на наявність багатьох аналогів, існують прогалини в знаннях та нездійснені вимоги до виробів або розробок в галузі аркадних ігор. Наприклад, виникають технічні протиріччя, пов'язані з оптимізацією графічного рендерингу та управлінням ресурсами для досягнення плавності геймплею на різних платформах. Також можуть бути проблеми з безпекою та надійністю, особливо в онлайн-іграх, де необхідно забезпечити захист від шахрайства та злому.

Жанр Arcade (аркада) є одним з найдавніших у галузі відеоігор і пройшов довгий шлях розвитку. Навіть якщо його не можна вважати найпопулярнішим, він суттєво вплинув на інші жанри. Основною перевагою аркад є їхній короткий, але насичений геймплей. Розробники продовжують створювати нові проекти, що поєднують елементи аркад і інших жанрів, намагаючись привернути увагу більшої кількості гравців та створити щось цікаве й унікальне. Навіть якщо пік

популярності аркад припав на кінець 20-го століття, до цього дня в цьому жанрі є величезна кількість прихильників, які не дозволяють йому зникнути з картини ігрової індустрії.

Аркадні ігри на двигуні Unity застосовуються у різних сферах, включаючи розваги, навчання та тренування. Вони можуть бути використані в ігрових консолях, персональних комп'ютерах, мобільних пристроях та віртуальній реальності. Аркадні ігри дозволяють користувачам отримувати задоволення від швидкого та захоплюючого геймплею, а також розвивати рефлективні та реактивні навички.

1.2. Призначення розробки та галузь застосування

Комп’ютерні ігри є невід’ємною частиною повсякденного життя більшості сучасних молодих людей як в Україні, так і у всьому світі. В результаті виконання кваліфікаційної роботи буде створено гру типу Runner на платформі Unity 3D за допомогою використання мови програмування C#. Основна ціль даного продукту є використання в розважальних цілях та розслаблення від рутини.

Цей додаток призначений для користувачів ПК з операційною системою Microsoft Windows.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є

- Освітня програма 122 «Комп’ютерні науки»;
- Графік навчального процесу та навчальний план;

– Наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
– Завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення ігрового додатку за допомогою Unity».

1.4. Постановка завдання

Завдання даної кваліфікаційної роботи є розробка тривимірної гри типу раннер у жанрі Arcade на движку Unity. Програмний додаток створений для використання користувачами ПК з операційною системою Microsoft Windows.

Основні характеристики розробленої гри мають бути:

– реалізація ігрових механік для забезпечення гравця можливості переміщення;
– розробка нескінченної дороги;
– створення інтерфейсу ігрового додатку за допомогою можливостей ядра Unity Engine;
– тестування розробленого ігрового додатку.

Результатом розробки має бути 3D гра для користувачів ПК з операційною системою Microsoft Windows, в якому реалізовано геймплей, що базується на взаємодії користувача, розроблених механік ігрового додатку та візуального інтерфейсу.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для того, щоб правильно запустити програму, необхідно встановити DirectX до версії 11. Якщо використовується більш рання версія, програма не зможе запуститися через обмеження у використанні функцій у гральному движку.

DirectX є набором програмних інтерфейсів (API), розроблених для зручного і ефективного використання мультимедіа, програмування в галузі ігор та відео на операційних системах Microsoft Windows.

Головні особливості DirectX 11:

- Підтримка багатоядерних обчислень та нових моніторів формату 16:9
- Поява модуля тесселяції (збільшення кількості вершин)
- Збільшена кількість шейдерних бібліотек
- Підтримка штучного інтелекту
- Оновлення блоку обробки шейдерів
- Нові формати стиснення текстур і передавання шейдера в глибинний буфер

Недоліки DirectX 11:

- Високі потреби до ресурсів комп'ютера
- Відсутність підтримки попередніх ОС
- Відсутність можливості трасування променів [4]

Необхідно оновити версію драйвера відеокарти, яка встановлена в системі, оскільки правильна робота гри залежить від цього.

Оскільки проект вже зібраний у виконуваний файл, користувачу просто потрібно запустити його.

Введення даних здійснюється через зовнішні пристрої для вводу інформації, а саме клавіатури.

Виведення даних здійснюється за допомогою монітора.

1.5.2. Вимоги до інформаційної безпеки

Додаток запаковано в один файл, не потрібно жодних вимог щодо безпеки інформації. Крім того, користувачу не потрібно реєструватися або створювати обліковий запис всередині програми. З цієї причини ніяких вимог до інформаційної безпеки немає.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для коректного функціювання ігрового додатку, мінімальна технічна конфігурація ПК повинна бути наступною:

- операційна система (ОС): Windows 7 і більше;
- процесор: Intel Core i3;
- оперативна пам’ять: 2 Гб;
- відеокарта 1 Гб;
- DirectX, версія 11;
- місце на жорсткому диску: 750 Мб;
- роздільна здатність монітора: 1280x960 пікселів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Кваліфікаційна робота була розроблений на мові C# в середовищі розробки IDE Microsoft Visual Studio 2022 з використанням ігрового двигуна Unity 3D, та повинна бути сумісною з операційною системаю Microsoft Windows 7 та її пізнішими версіями.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Метою даної кваліфікаційної роботи є створення тривимірного ігрового додатку у жанрі Arcade з використанням движка Unity. Цей розроблений ігровий додаток буде призначений для гравців, які бажають розважитися і гарно провести час.

Основні функції ігрового додатку включають:

- Організацію геймплею;
- Призначення для використання на ПК;
- Наявність функціонального інтерфейсу.

Елемент аркадної системи полягає у втіленні геймплею за допомогою взаємодії механік та ігрових елементів, які є характерними для жанру аркадних ігор.

Механіка поведінки об'єктів у 3D просторі визначається розробленими функціями, що встановлюють правила взаємодії цих об'єктів. У даному випадку, використовується Unity Engine, який дозволяє створювати компоненти, що задають ці правила, шляхом написання скриптів на мові програмування C# для конкретних ігрових об'єктів.

У розробленій грі була втілена система перешкод, що є однією з особливостей аркадних ігор. Конкретно, був реалізований типовий ігровий процес - "Endless Runner", який полягає в безкінечному пересуванні гравця між з’являючимися платформами.

Додатково, у грі був реалізований елемент дистанції гравця та бонусних монет, які дозволяють йому спостерігати свій внутрішній прогрес під час геймплею.

2.2. Опис застосованих математичних методів.

Під час проектування та розробки даної інформаційної системи використовувалися лише прості арифметичні дії. Математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування.

Для створення двовимірного додатка для аркадних ігор були використані такі технології: Unity 2022.3.0 - платформа розробки, Microsoft Visual Studio 2022 - інтегроване середовище розробки, та мова програмування C#.

Unity - це багатоплатформовий інструмент для розробки двох та тривимірних застосунків та ігор. Він працює на операційних системах Windows і macOS. За допомогою Unity можна створювати застосунки, які працюють на різних платформах, включаючи Windows, macOS, Android, iOS, PlayStation 4, Nintendo Switch та інші. Також Unity надає можливість створювати інтернет-застосунки за допомогою спеціального модуля для браузера Unity та експериментальної реалізації для Adobe Flash Player. Застосунки, створені з використанням Unity, підтримують DirectX та OpenGL.

Unity є кросплатформовим 3D та 2D двигуном, який дозволяє розробляти ігри для різних платформ і операційних систем. Він є одним з найпопулярніших рушіїв у галузі розробки і використовується як одиночними розробниками, так і ігровими студіями. Unity відзначається своєю модульністю, зручною документацією та наявністю відкритого коду у платній версії. Він також підтримує DirectX та OpenGL.

Кожне ігрове ядро надає багато функцій, які використовуються у різних іграх. За допомогою Unity, розробники можуть отримати доступ до цих функцій, а також додати власні ігрові ресурси та код сценарію гри (скрипти). Unity також забезпечує можливість моделювання фізичного середовища, створення карт

нормалей, зображення ігрового світу у екранному просторі (SSAO), динамічні тіні та багато іншого.

В Unity вбудовані корисні функції, такі як внутрішній файловий провідник, вікно анімації, фізичні налаштування, робота з реєстром, робота з XML-файлами, підтримка написаних шейдерів, реклама та внутрішні покупки у грі. Unity також надає рішення для спільної розробки через Asset Store, підтримку мережі та генератор ландшафтів. Крім того, Unity дозволяє розширювати рушій власноруч та ділитися цими розширеннями з іншими користувачами.

Unity має дві основні переваги порівняно з іншими високоякісними ігровими ядрами: ефективний візуальний робочий процес і потужну підтримку багатьох платформ.

Є деякі переваги візуального робочого процесу. В оточенні розробки ігор інших інструментів часто потрібно відстежувати різні компоненти або налаштовувати власне інтегроване середовище розробки (Integrated Development Environment), послідовність збірки тощо. Проте в Unity робочий процес пов'язаний з добре продуманим візуальним редактором. Розробник складає сцени для наступної гри і об'єднує ігрові ресурси та код в інтерактивні об'єкти. Це дозволяє швидко і ефективно створювати професійні ігри, забезпечуючи продуктивність розробників і надаючи їм доступ до найновіших технологій розробки відеоігор.

Unity також має потужну підтримку для багатьох платформ. Це означає не лише можливість розгортання гри на різних платформах, таких як персональні комп'ютери, Інтернет, мобільні пристрої та консолі, але й доступ до різних інструментів програмування. Unity спочатку був розроблений для комп'ютерів Mac, а потім перенесений на комп'ютери з операційною системою Windows. Існує небагато ігрових ядер, що підтримують стільки цільових платформ розгортання, і жодне з них не забезпечує такої легкості розгортання, як Unity.

Крім цих переваг, варто відзначити модульну систему компонентів, яка використовується для побудови ігрових об'єктів. В такій системі "компоненти" є зв'язками функціональних елементів, тому об'єкти створюються як набір

компонентів, а не за фіксованою ієрархією класів. Замість успадкування, система компонентів пропонує альтернативний і більш гнучкий підхід до об'єктно-орієнтованого програмування, де ігрові об'єкти створюються шляхом комбінування компонентів. Порівняння підходів наведено на рисунках 2.1. та 2.2.

Рис. 2.1. Порівняння підходів при конструюванні ігрових об’єктів

Рис. 2.2. Порівняння підходів при конструюванні ігрових об’єктів

У системі компонентів об'єкти організовані в горизонтальну ієрархію, тому різні об'єкти складаються з відмінних наборів компонентів, а не зі структури успадковування, де різні об'єкти знаходяться на різних гілках дерева. Такий підхід спрощує створення прототипів, оскільки обробка необхідного набору компонентів є швидшою і легшою, порівняно з переставленням успадкованої структури під час зміни кожного об'єкта.

Більш тим, програміст має можливість створювати свою власну компонентну систему шляхом написання власного коду, але в Unity вже є готова та надійна компонентна система, яка органічно вбудована у візуальний редактор. Тому розробник може не тільки програмно керувати компонентами, але й встановлювати та розривати їх зв'язки безпосередньо у редакторі. Можливості розробки не обмежуються лише складанням об'єктів з готових компонентів; у коді програміст може використовувати успадкування та всі вже розроблені шаблони проектування.

Інтерфейс Unity складається з кількох компонентів: вкладка Scene, вкладка Game, панель інструментів, вкладка Hierarchy, панель Inspector, вкладки Project і Console (рисунок 2.3).

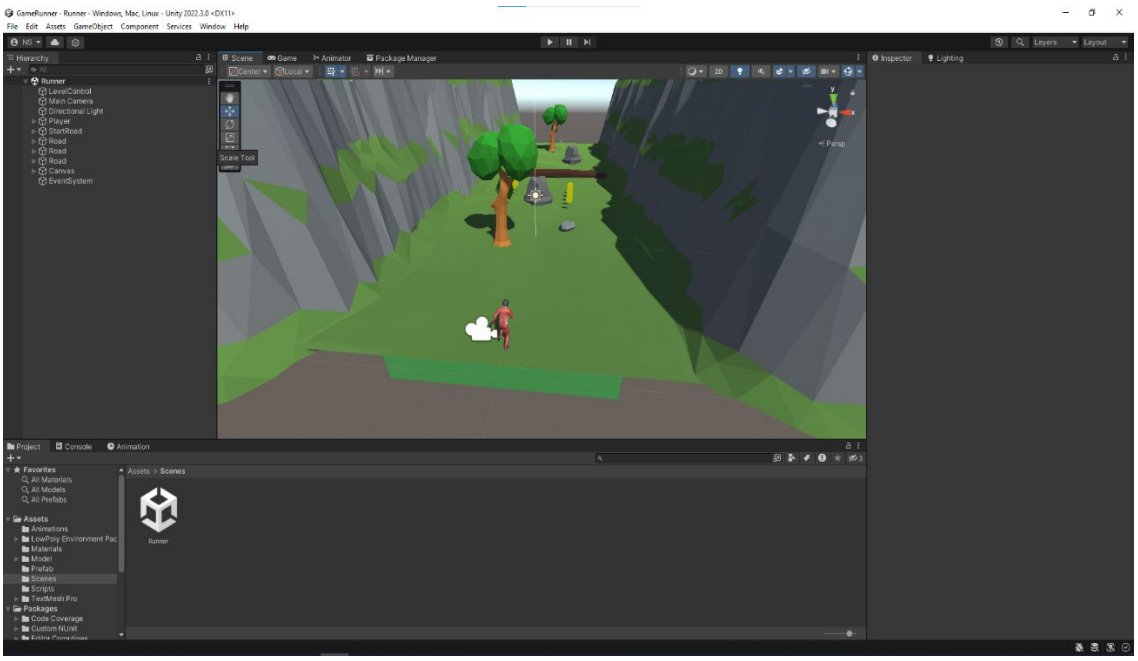

Рис. 2.3. Інтерфейс середовища Unity

Кожна частина інтерфейсу має своє призначення і грає важливу роль у процесі створення гри:

- Вкладка Project використовується для перегляду файлів.
- Вкладка Scene показує об'єкти, розміщені на сцені.
- Панель інструментів містить елементи керування сценою.
- Вкладка Hierarchy дозволяє змінювати зв'язки між об'єктами.
- Панель Inspector надає інформацію про виділені об'єкти та пов'язаний з ними код.
- Вкладка Game використовується для виконання тестування проекту, а вкладка Console відображає інформацію про помилки

В центрі інтерфейсу знаходиться вкладка "Scene" (рисунок 2.4), де можна переглянути вигляд світу гри та розміщувати об'єкти на сцені. Важливо зрозуміти, що те, що відображається на цій вкладці, відрізнятиметься від того, що буде показано під час активної гри. Гра відображається на вкладці "Game", яка зазвичай знаходиться поруч з вкладкою "Scene". Після запуску гри на вкладці "Game" починає відображатися активна гра.

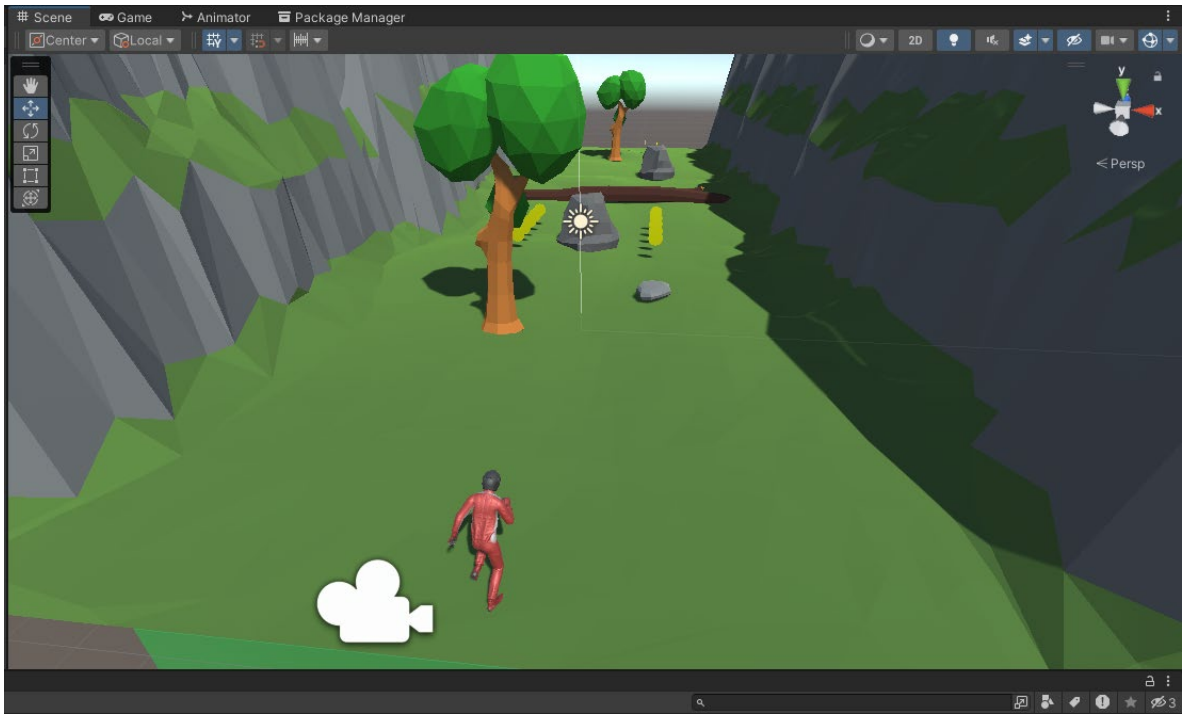

Рис. 2.4. Частина редактора, яка демонструє панель інструментів і вкладку Scene

Вкладка "Hierarchy" (рисунок 2.5) містить деревоподібний список всіх об'єктів на сцені, в якому гілки вкладені одна в одну відповідно до ієрархічних зв'язків між об'єктами. Ця вкладка дозволяє зручно виділяти об'єкти за їх назвами, уникнувши необхідності шукати їх на сцені. Ієрархічні зв'язки використовуються для з'єднання об'єктів один з одним.

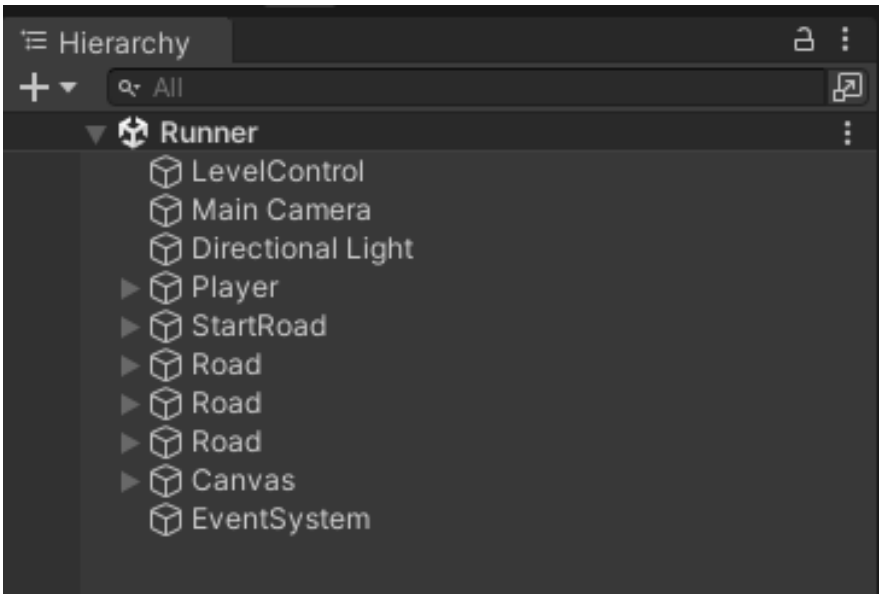

Рис. 2.5. Частина редактора, яка демонструє тільки вкладку Hierarchy

На панелі "Inspector" (рисунок 2.6) відображаються дані виділеного об'єкта в поточний момент. При виділенні різних об'єктів розробник одразу бачить, як змінюється вигляд панелі "Inspector". На цій панелі зазвичай відображається список компонентів об'єкта, і програміст може додавати нові компоненти до об'єктів або видаляти існуючі.

Усі ігрові об'єкти мають принаймні один компонент - "Transform", тому на панелі "Inspector" завжди будуть відображатися інформація про розташування та орієнтацію виділеного об'єкта. Багато об'єктів мають також списки компонентів, включаючи сценарії, які пов'язані з цими об'єктами.

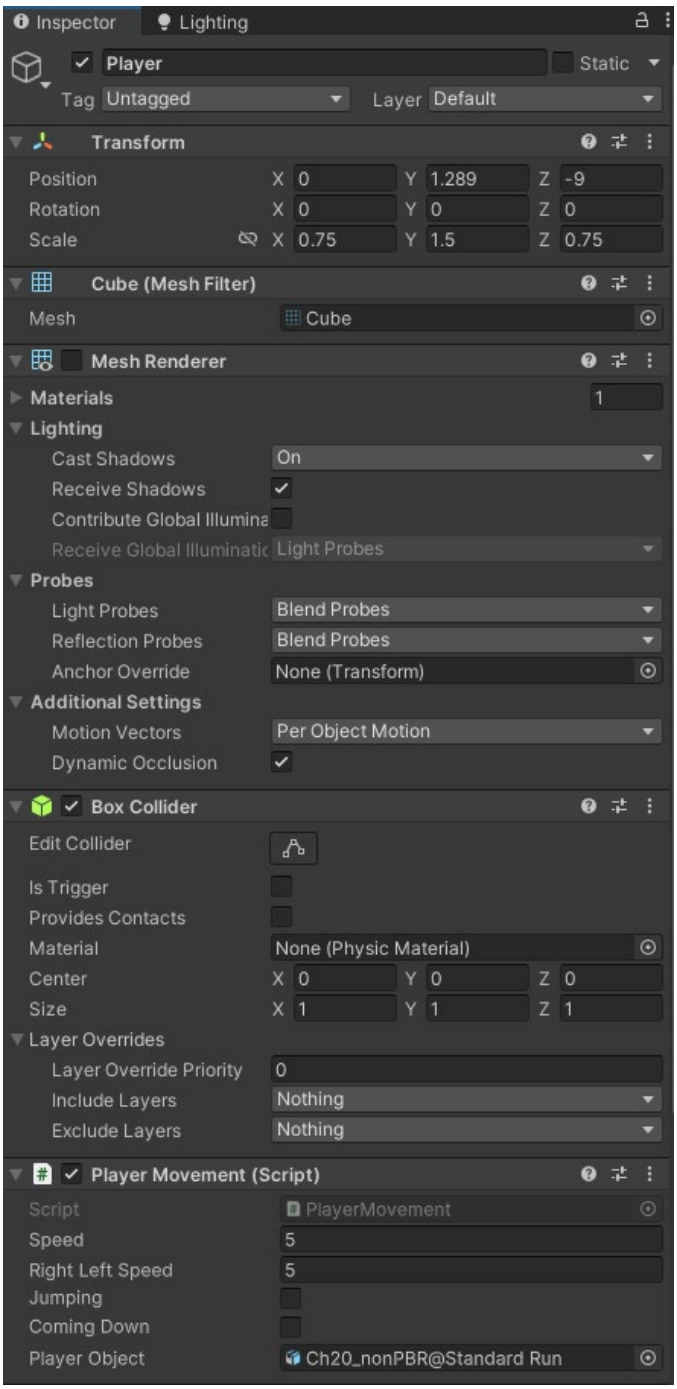

Рис. 2.6. Частина редактора, яка демонструє тільки вкладку Inspector

У нижній частині екрана знаходяться вкладки "Project" і "Console", які показані на рисунку 2.7. На вкладці "Project" відображаються всі ресурси проекту, такі як графічні фрагменти, файли скриптів, префаби та інші. Ліва частина цієї вкладки містить список папок проекту, а при виділенні папки праворуч відображаються файли, які знаходяться в цій папці. Навіть файли, що

не включені до будь-якої сцени, також відображаються на вкладці "Project". Крім того, на цій вкладці можна створювати нові сцени.

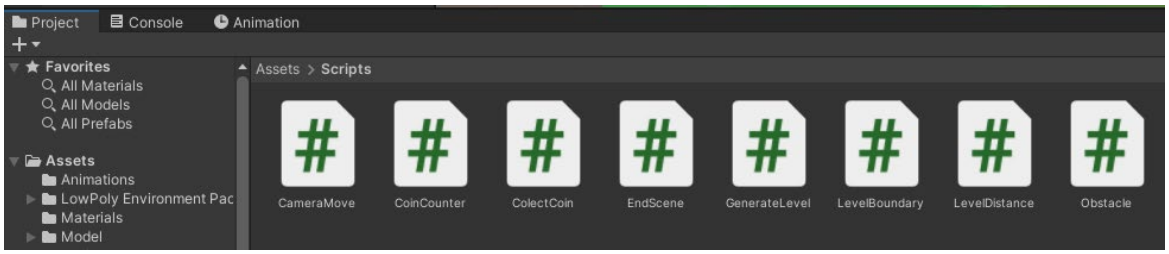

Рис. 2.7. Частина редактора, яка демонструє вкладку Project та Console

Вкладка "Console" є панеллю, на яку виводяться повідомлення, пов'язані з кодом. Це можуть бути розробником додані повідомлення для відлагодження програми або повідомлення Unity, що виникають при виявленні помилок у написаних сценаріях програмістом.

Для створення будь-якої гри в середовищі Unity, програміст повинен мати мінімум одну з доступних мов програмування (C# або JavaScript).

C# - це мова програмування з об'єктно-орієнтованою структурою та безпечною системою строго типізації, яка розроблена та належить компанії Microsoft. Синтаксис C# схожий на C++ і майже ідентичний з Java. Вона підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів та інші функції. Поточною версією є C# 8.0.

Головним компілятором для C# є Microsoft Visual C#, а також існують інші компілятори, такі як DotNetAnywhere, Microsoft Rotor, SharpDevelop, Mono, DotGNU.

Для мови програмування C# було розроблено багато бібліотек спеціально для двигуна UNITY 3D. Ось декілька основних бібліотек, без яких не обходиться жоден проект:

UnityEngine - основна бібліотека, яка забезпечує роботу всіх компонентів у грі та реалізує базовий клас "MonoBehavior", від якого успадковуються інші класи;

UnityEngine.SceneManagment - бібліотека, призначена для роботи зі сценами в UNITY;

UnityEngine.UI - бібліотека, створена для роботи з елементами користувацького інтерфейсу, такими як канвас, кнопки, слайдери, важелі, зображення та багато іншого.

Крім того, мова програмування C# дозволяє працювати з файлами, такими як XML, а також з реєстром, що дозволяє зберігати прогрес гравця та всі його налаштування.

Мова програмування C# має кілька переваг перед JavaScript та меншу кількість недоліків. Однією з переваг є строга типізація мови C#, в той час як JavaScript має динамічну типізацію. Хоча серед програмістів є різні точки зору на тему того, чи є динамічна перевірка типів оптимальним підходом, особливо для веб-розробки, але при розробці програм для певних платформ, таких як Windows, статична типізація часто є вигідною або навіть необхідною. В Unity навіть є директива "#pragma", яка забезпечує примусову статичну перевірку типів для мови JavaScript. Хоча технічно це можливо, це порушує один з основних принципів функціонування JavaScript, тому краще вибрати мову зі строгою типізацією, таку як C#. [4]

Поведінка ігрових об'єктів в Unity контролюється за допомогою компонентів, які приєднуються до них. Програміст може створювати власні компоненти, використовуючи сценарії (скрипти). Сценарії дозволяють активувати ігрові події, змінювати параметри компонентів та реагувати на введення користувача за допомогою різних методів.

Програмування в Unity відбувається у вигляді окремих файлів, а не всередині самої Unity. Розробник повідомляє Unity про місце розташування цих файлів. Файли сценаріїв можуть бути створені в самому додатку Unity, але в будь-якому випадку програмісту потрібно використовувати текстовий редактор або інтегроване середовище розробки (IDE), де буде писатися код для цих файлів, які спочатку є порожніми.

У комплекті з Unity постачається додаток Microsoft Visual Studio 2022, який є інтегрованим середовищем розробки (IDE) з підтримкою мови C# (див. рисунок 2.8). Програма Microsoft Visual Studio 2022 організовує файли в групи, відомі як рішення. Інструмент Unity автоматично генерує зміст всіх файлів сценаріїв у рішенні.

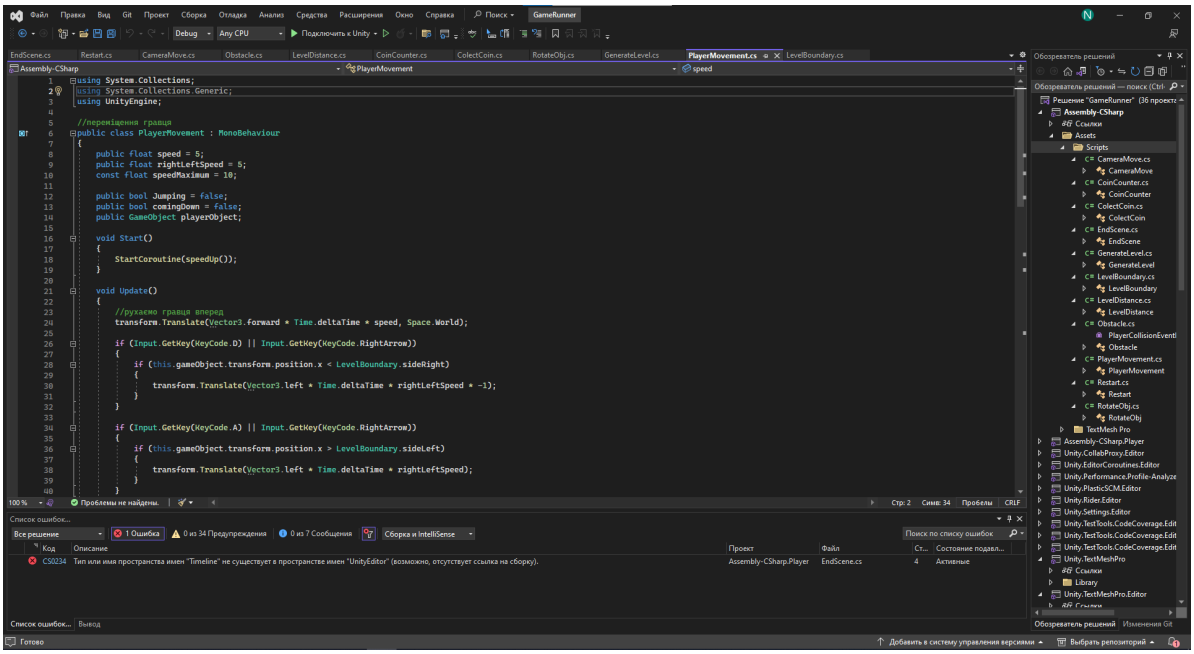

Рис. 2.8. Інтерфейс Microsoft Visual Studio 2022

Створення сценаріїв в Unity відбувається безпосередньо всередині самої програми. Програміст може створити новий скрипт, перейшовши до меню Create, розташованого у верхньому лівому куті панелі Project, або вибравши Assets > Create > C# Script в головному меню.

Після цього новий скрипт буде створено в обраній папці в панелі Project, і його ім'я буде виділено, щоб дозволити введення нового імені (рисунок 2.9).

Рис. 2.9. Створення нового сценарію NewBehaviourScript.cs

Подвійне клацання по скрипту в Unity відкриє його в редакторі скриптів. За замовчуванням Unity використовує Visual Studio 2022, але розробник може використати будь-який інший редактор з панелі External Tools у налаштуваннях двигуна Unity.

Скрипт взаємодіє зі внутрішніми механізмами Unity, створюючи клас, який успадковується від вбудованого класу MonoBehaviour. Цей клас можна розглядати як план компонента нового типу, який може бути прикріплений до ігрового об'єкта. Кожен раз, коли програміст прикріплює скрипт до ігрового об'єкта, створюється новий екземпляр об'єкта, визначений цим планом. Ім'я класу береться з імені файлу, що використовувалось при створенні скрипту. Ім'я класу і ім'я файлу повинні бути однаковими, щоб скрипт міг бути прикріплений до ігрового об'єкта.

Особливу увагу слід приділити двом функціям, що визначені всередині класу. Функція Update призначена для коду, який обробляє оновлення кадру для ігрового об'єкта. Вона може містити рухи, реакції на дії користувача та будь-які інші події, що відбуваються протягом гри. Часто перед виконанням дій у функції Update корисно форматувати змінні, зчитувати властивості та взаємодіяти з іншими ігровими об'єктами. Функція Start викликається Unity перед початком

гри (до першого виклику функції Update) і чудово підходить для ініціалізації змінних.

Скрипт визначає лише план компонента, і його код не буде активований, поки екземпляр скрипта не буде прикріплений до якогось об’єкта. Програміст може прикріпити скрипт до об'єкта, перетягуючи ассет скрипта на ігровий об'єкт в панелі Hierarchy або вікні Inspector для вибраного об'єкта. Також доступне підменю Scripts в меню Component, яке містить всі скрипти, доступні в проекті, створені вами. Екземпляр скрипта відображатиметься у вікні Inspector разом з іншими компонентами (рисунок 2.10).

Після прикріплення скрипт почне виконуватись, коли користувач натисне кнопку Play та почне гру.

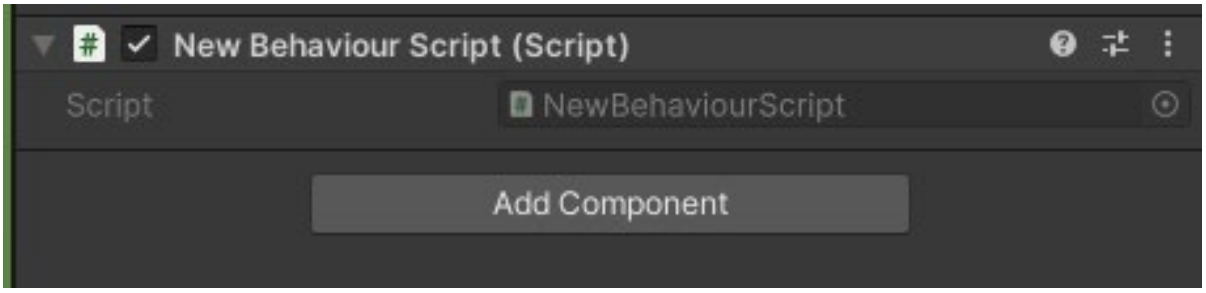

Рис. 2.10. Приєднаний скрипт NewBehaviourScript.cs

2.4. Опис структури системи та алгоритмів її функціонування

Цей ігровий застосунок на Unity складається з наступних складових частин:

Логічна структура системи:

Головна сцена гри:

- Відповідає за ініціалізацію гри і керування загальним потоком програми.
- Містить об'єкти, які відповідають за відображення графіки гри, такі як гравець, перешкоди та інші об'єкти.
- Здійснює обробку введення користувача, такого як керування гравцем.

Менеджер гри:

- Відповідає за керуванням усіма аспектами гри.

– Містить логіку для початку та завершення гри, розрахунку дистанції гравця.

Система фізики:

– Використовується для моделювання фізики гри, такої як рух об'єктів, зіткнення та гравітація.
– Забезпечує реалістичне поведінку об'єктів у грі.

Компоненти об'єктів:

– Кожен об'єкт в грі має свої компоненти, які відповідають за їхню поведінку.
– Наприклад, компонент "PlayerMovement" відповідає за рух головного персонажа, компонент "Obstacle" відповідає за виявлення зіткнень.

Алгоритм та функціонування програми:

– Гра працює за принципом безкінечного раннера, де гравець керує персонажем, який намагається уникати перешкод і зібрати бонуси.
– Алгоритм генерує різні перешкоди та монети.
– Гравець може керувати персонажем, використовуючи клавіші.
– Рахунок гравця збільшується залежно від пройденого шляху та кількості зібраних бонусів.
– Гра завершується, коли гравець зіштовхується з перешкодою або падає з платформи.

Під час проектування ігрового додатку було розроблено скрипти, які використовуються для управління компонентами (Components). Ці скрипти активують ігрові події, змінюють і встановлюють значення параметрів компонентів та можуть реагувати на дії користувача в будь-який необхідний спосіб.

Ігровий додаток має меню, яке викликається коли користувач врізається у який-небудь об’єкт на дорозі, та може натиснути кнопку рестарту левелу. Результат проектування відображається на рисунку 2.11.

Рис. 2.11. Меню закінчення гри, з кнопкою рестарту

Для проектування геймплею в ігровому процесі використовується окрема сцена, де розміщені тривимірні об'єкти.

Проектована сцена має типовий вигляд для раннеру і включає дорогу, на якій розташований головний персонаж, керований гравцем, а також ігрові перешкоди (рисунок 2.12).

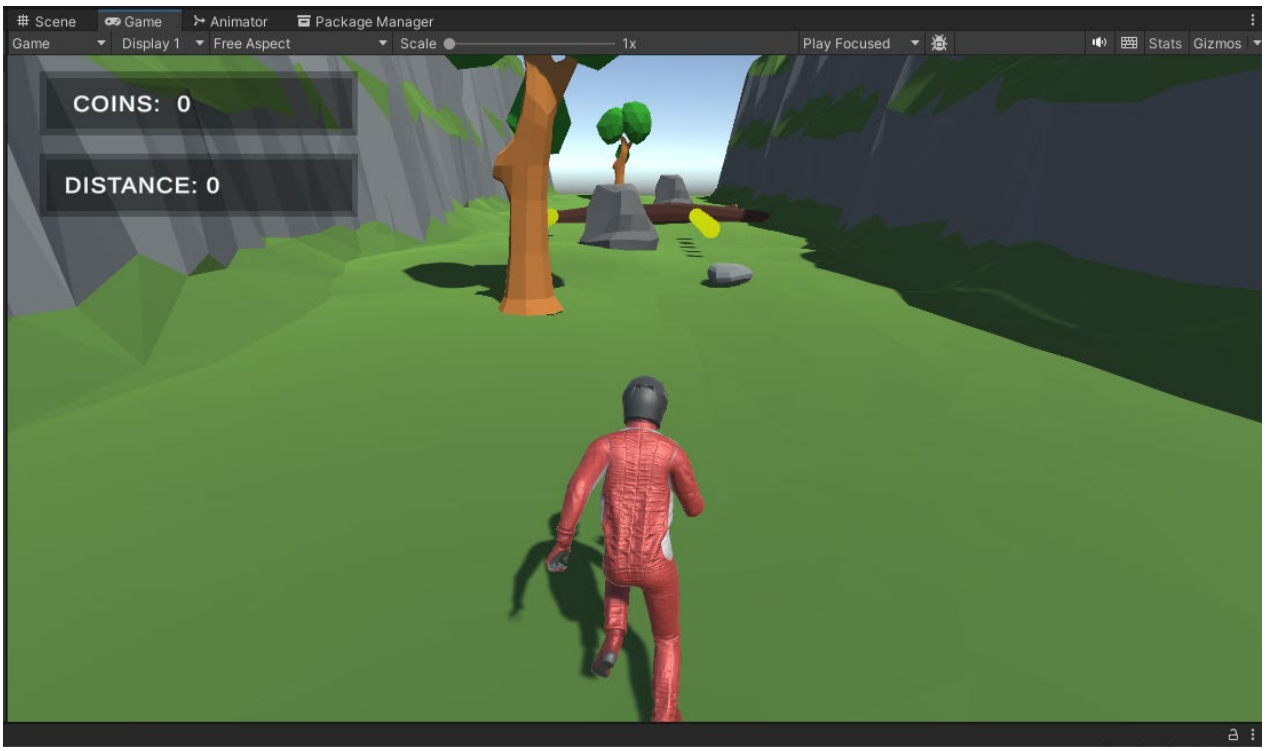

Рис. 2.12. Ігрова сцена

При розробці ігрового процесу, було створено наступні скрипти:

- PlayerMovement – скрипт, який забезпечує переміщення персонажу
- LevelBoundary – скрипт, який обмежує персонажа, щоб той не вийшов за межі дороги
- GenerateLevel – цей скрипт генерує дорогу з перешкодами
- RotateObj – у цьому скрипті реалізоване обертання об’єктів, а саме монети
- ColectCoin – скрипт, який додає зібрані монети до лічильника та видаляє їх на дорозі
- CoinCounter - у цьому скрипті реалізоване виведення кількості зібраних монет на екран
- LevelDistance – скрипт, який відображує відстань пройдену гравцем
- Obstacle – скрипт, який регеструє зіткнення з перешкодою
- CameraMove – цей скрипт контролює камеру
- Restart – скрипт, який завантажує гру спочатку в меню
- EndScene – у цьому скрипті реалізоване виведення меню рестарту з анімацією

Сценарії в Unity є компонентами, але тільки ті, що успадковують клас MonoBehavior, який визначає спосіб приєднання компонентів до ігрових об'єктів. Наслідування від цього класу автоматично додає два методи: Start(), який викликається при активації об'єкта (зазвичай після завантаження рівня), і Update(), який викликається в кожному кадрі. Код розміщений в цих методах виконується відповідно до їхньої функціональності.

Для керування ігровою камерою використовується скрипт CameraMove.cs на вкладці Inspector. У методі Start() встановлюється змінна move, яка представляє різницю між координатами розташування ігрових об'єктів. У методі LateUpdate() [5] реалізоване переміщення камери за допомогою Lerp(). [6] Без цього скрипта камера залишалась нерухомою, і геймплей закінчувався через декілька секунд.

Після налаштування камери, важливо забезпечити гравцю можливість переміщатись до певного місця та взаємодіяти з ігровими об'єктами.

Ці функціональності реалізовані за допомогою скрипту PlayerMovement.cs. В цьому скрипті містяться методи та змінні, які, наприклад, дозволяють гравцеві переміщатись у межах ігрового простору. Розроблені методи відповідають за внутрішню ігрову механіку, яка забезпечує взаємодію персонажа гравця з ігровим простором та об'єктами, що в ньому знаходяться. У скрипті PlayerMovement.cs реалізоване постійне переміщення по дорозі, а сам користувач за допомогою клавіатури керує положенням персонажу по горизонталі. Також користувач може виконувати стрибки для подолання поваленої колоди та малих каменів. На рисунку зображена реалізація ігрового процесу а саме стрибку (рис. 2.13).

Рис. 2.13. Реалізація ігрового процесу

У зв'язку з тим, що "збирання" є однією з ключових особливостей жанру Arcade, під час розробки ігрового додатку було реалізовано функціонал, який дозволяє гравцеві збирати ігрові об'єкти - монети. Для того, щоб привернути увагу гравця та викликати інтуїтивне розуміння, що потрібно збирати кристали, був створений скрипт RotateObj.cs, який забезпечує анімацію обертання кристалів. Ця анімація створюється за допомогою методу Update(), який викликається кожен кадр. У цьому методі обчислюється функція обертання компонента Transform для кристала. У компоненті Transform зберігаються координати розташування та розмір ігрових об'єктів. Результат роботи цього скрипта можна побачити на рисунках 2.14 та 2.15.

Рис. 2.14. Обертання кристалів реалізується у скрипті RotateObj.cs

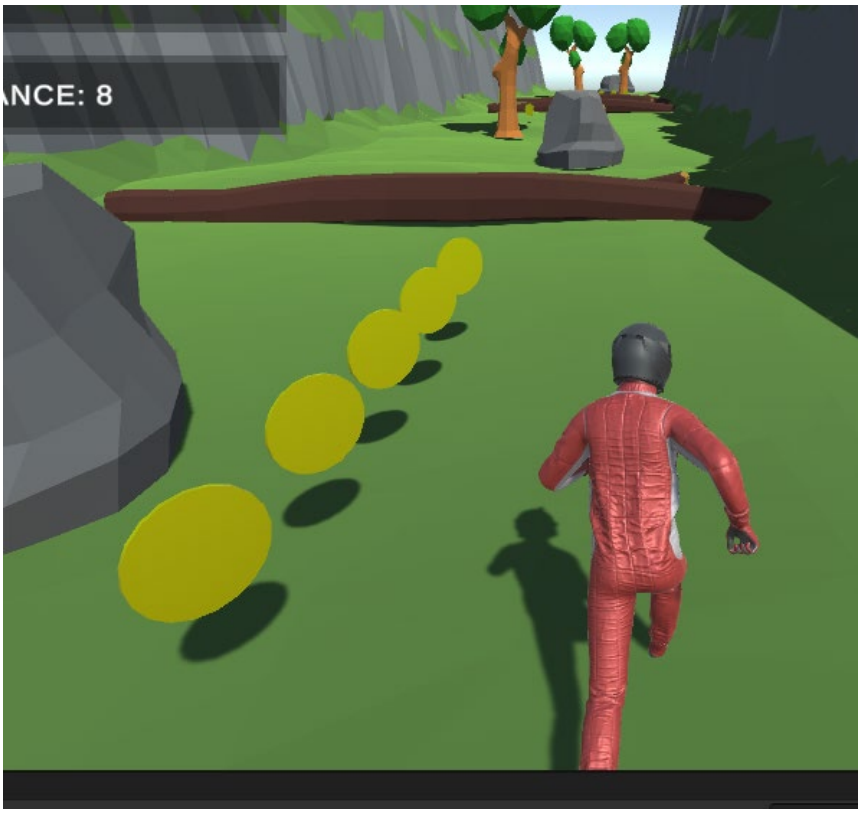

Рис. 2.15. Обертання кристалів, відображено з іншого ракурсу

Функціонал генерації дороги з перешкодами у ігровому додатку реалізовано за допомогою скрипту GenerateLevel.cs

Завдяки розробленим методам, дороги регулярно створюються з встановленим інтервалом часу, що забезпечує гравця потенційно нескінченним ігровим процесом. Дороги починають з'являтися після старту геймплею. Суть генерації полягає в тому, що до компоненту додаються координати майбутньої платформи, враховуючи її розмір.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Ігровий додаток отримує вхідні дані від пристроїв введення для керування ігровим процесом. Ці дані включають натискання клавіш на клавіатурі, позицію та натискання кнопок миші, а також використання сенсорного екрану на мобільних пристроях.

Згідно з поставленого завдання у якості вхідних даних програми виступають:

- переміщення персонажу;
- взаємодія з різними об'єктами;

Вихідними даними є:

- відображення анімацій;
- відображення положення головного героя;
- вивід результату гри.

2.6. Опис розробленої системи

Для того, щоб перетворити розроблений ігровий додаток у готовий продукт для споживача, необхідно згенерувати білд ігрового застосунку. Білд - це пакетування ігрового додатку, який може бути використаний гравцем. Пакетування здійснюється за допомогою функціоналу движка Unity Engine. Процес пакетування виконується автоматично, вам лише потрібно вказати цільову платформу для розгортання та натиснути кнопку "Build and Run".

Результатом пакетування ігрового додатку буде папка з готовою грою, яку можна використовувати гравцем (Рис. 2.16).

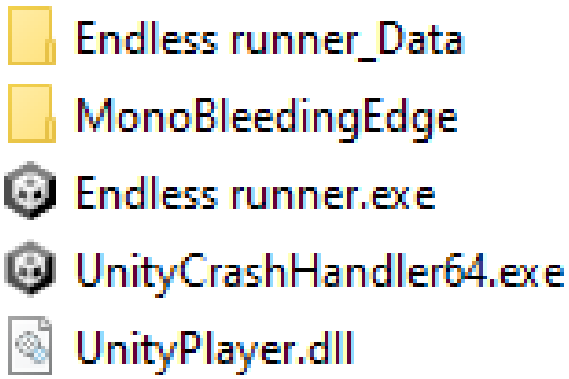

Рис. 2.16. Результат упаковки ігрового додатку

Для роботи з програмою достатньо просто запустити файл "Endless runner.exe". Після цього ігровий додаток запуститься. Якщо мінімальні технічні характеристики обладнання, на якому запускається цей додаток, не відповідають вимогам, користувач отримає повідомлення про це.

2.6.1. Використані технічні засоби

Для розробки даного ігрового додатку було використано наступні технічні засоби:

- Процесор: AMD Ryzen 5 2600 3.4 GHz;
- Відеокарта: Nvidia GeForce GTX 1650 Super;
- Оперативна пам’ять: 16 ГБ;
- Операційна система: Windows 10.
- Монітор
- Засобі периферії (клавіатура, миша)

2.6.2. Використані програмні засоби

Для розробки було використано такі програмні засоби:

- Unity Engine версії не менш, ніж 2022.3.0;
- Visual Studio версії 22.

2.6.3. Виклик та завантаження програми

Для запуску проекту потрібно завантажити архівний файл і розархівувати його за допомогою будь-якого архіватора. Після розпакування архіву достатньо лише запустити виконавчий файл (Endless runner.exe), який знаходиться у папці програми. Додаткової інсталяції не потрібно. На рисунку 2.17. показано розпакований вигляд архіву.

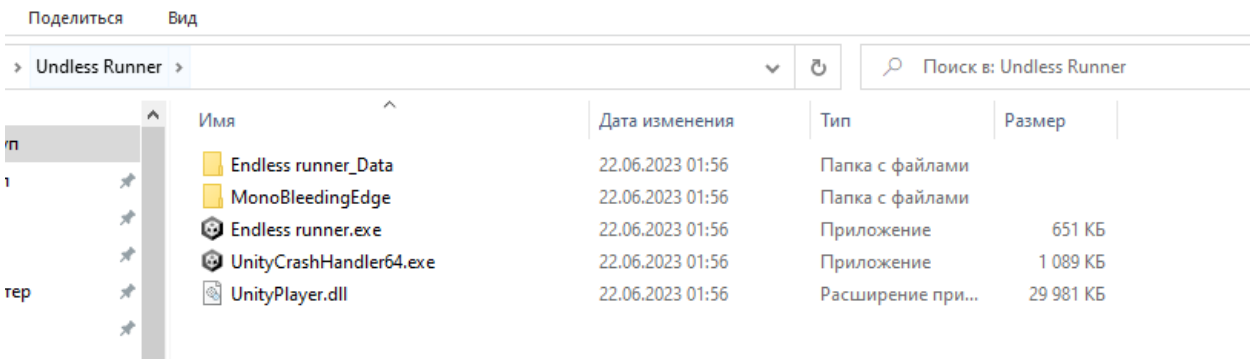

Рис. 2.17. Запуск ігрового додатку із папки

2.6.4 Опис інтерфейсу користувача

Під час створення тривимірної аркадної гри було розроблено функціональний інтерфейс користувача, який включає головне меню та відображення ігрового рахунку. Щоб все працювало належним чином, необхідно правильно налаштувати та запустити сам ігровий двигун.

Рис. 2.18. Екран завантаження гри

Рис. 2.19. Початок гри

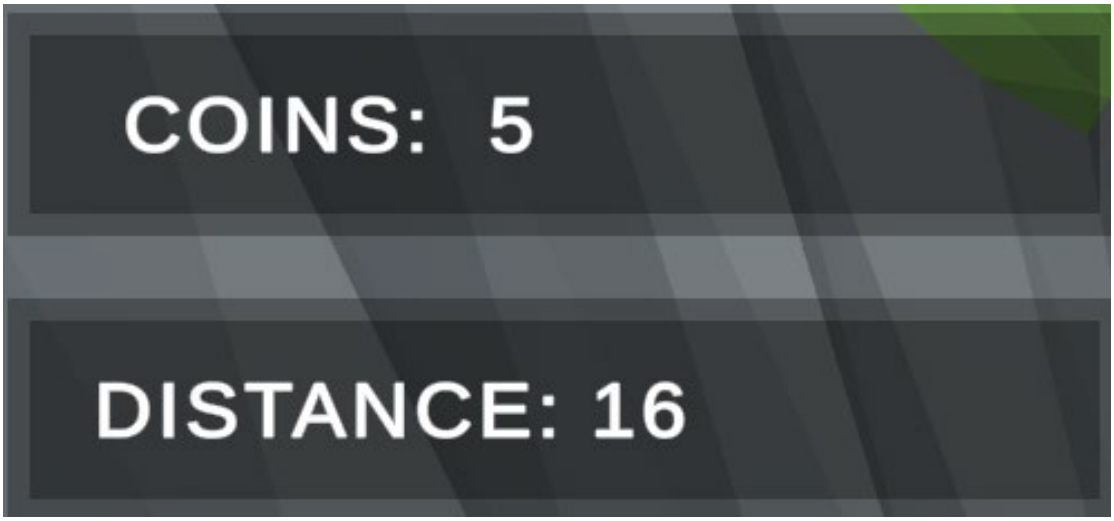

Рис. 2.20. Показники монет і дистанції

Рис. 2.21. Меню закінчення гри, з кнопкою рестарту

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Вхідні дані:

– передбачуване число операторів – 820;

– коефіцієнт корекції програми в ході її розробки – 0,2;

– коефіцієнт складності програми – 1,25;

– годинна заробітна плата програміста – 171,1 грн/год;

– коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,25;

– коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;

– вартість машино-години ЕОМ – 25 грн/год;

Заробітна плата за годину Junior Game Developers була вирахувана згідно «Української спільноти програмістів (DOU.UA)». Станом на 2022 рік зарплата Junior Game Developers вар'юється від 670$ до 980$. Вирахувавши середню заробітну плату програміста маємо плату 825$ у місяць. При курсі валют НБУ на кінець грудня 2022 року один американський долар дорівнює 36,5 грн, тому середня зарплата в гривнях дорівнює 30 112 грн. При стандартному графіку (176 годин/місяць) зарплата за годину буде становити близько 171,1 грн.

Виходячи з того, що для розробки даної системи необхідною є значна потужність ПК для безперервної роботи серверу, зберігання та підтримки великої кількості даних на сервері, вдалим рішенням буде оренда ноутбуку.

Вартість оренди ноутбуку на місяць становить 4000 грн. При стандартному графіку (160 годин/місяць) вартість машино-години ЕОМ за годину роботи буде становити 25 грн.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}, \text{ людино-годин}, \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \quad (3.2)$$

де q - передбачуване число операторів (820);

C - коефіцієнт складності програми (1,25);

p - коефіцієнт корекції програми в ході її розробки (0,2).

За формулою (3.2) умовне число операторів в програмі становить:

$$Q = 820 * 1{,}25 * (1 + 0{,}2) = 1230 \quad (3.3)$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q*B}{(75..85)*k}, \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,25);

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 2 до 3 років він складає 1,2.

Збільшення витрат праці внаслідок недостатнього опису завдання будемо приймати не більше 50% (B = 1,25).

З урахуванням коефіцієнта кваліфікації k = 1,1 за формулою (3.4) отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{820*1,25}{85*1,1} = 10,96, \text{ людино-годин}, \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, \quad (3.6)$$

де Q – умовне число операторів програми (820);

k – коефіцієнт кваліфікації програміста (1,1).

Згідно формулі (3.6), отримаємо:

$$t_a = \frac{820}{25*1,1} = 29,8, \text{ людино-годин}, \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \tag{3.8}$$

Маючи формулу (3.8) витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{820}{25*1,1} = 29,8, \text{ людино-годин}, \tag{3.9}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k}, \tag{3.10}$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання за формулою (3.10):

$$t_{\text{отл}} = \frac{820}{5*1,1} = 149,09, \text{ людино-годин}, \tag{3.11}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 * t_{\text{отл}}, \tag{3.12}$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.12):

$$t_{\text{отл}}^{k} = 1,5 * 149,09 = 223,6, \text{ людино-годин}, \tag{3.13}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \tag{3.14}$$

де $t_{\text{др}}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15..20)*k}, \tag{3.15}$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \tag{3.16}$$

Маючи формули (3.15), (3.16) та (3.14) обчислюємо витрати праці на документацію:

$$t_{\text{др}} = \frac{820}{20*1{,}1} = 37{,}27, \text{ людино-годин}, \tag{3.17}$$

$$t_{\text{до}} = 0{,}75 * 37{,}27 = 27{,}95, \text{ людино-годин}, \tag{3.18}$$

$$t_{\text{д}} = 37{,}27 + 27{,}95 = 65{,}22, \text{ людино-годин}, \tag{3.19}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 10{,}96 + 29{,}8 + 29{,}8 + 149{,}09 + 65{,}22 = 821{,}91,$$

людино-годин. (3.20)

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ КПО включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * С_{ПР}, \text{ грн,} \quad (3.22)$$

де t - загальна трудомісткість, людино-годин (821,91);

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 171,1 грн / год, за формулою (3.22) отримуємо:

$$З_{ЗП} = 821{,}91 * 171{,}1 = 140\,628, \text{ грн,} \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} * С_{Мч}, \text{ грн,} \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год (149,09 год);

$С_{Мч}$ - вартість машино-години ЕОМ, грн/год (25 грн/год).

Підставивши в формулу (3.24) відповідні значення, обчислюємо вартість необхідного для налагодження машинного часу:

$$З_{МВ} = 149{,}09 * 25 = 3\,727, \text{ грн,} \quad (3.25)$$

Звідси, за формулою (3.21), витрати на створення програмного продукту:

$$К_{ПО} = 140628 + 3727 = 144355, \text{ грн}, \quad (3.26)$$

Очікуваний період створення програмного застосунку:

$$T = \frac{t}{B_k * F_p}, \text{ міс}, \quad (3.27)$$

де B_k - число виконавців (1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p =176 годин).

За формулою 3.27 очікуваний період створення програмного забезпечення:

$$T = \frac{821,91}{1*176} \approx 4,6 \text{ міс}. \quad (3.28)$$

Висновки: Цей ігровий додаток був створений з метою надання гравцям можливості скоротити час та розважитися під час дозвілля. Розробка даного програмного забезпечення становить 144355 грн без додаткових витрат.

Згідно отриманих даних при розрахунках, очікуваний час розробки становить 3358 години, або 4,6 місяців, з урахуванням стандартного робочого графіку. Результат, що отримано, залежить від незначного коефіцієнта кваліфікації розробника, та у свою чергу включає час на виконання досліджень та розробку концепції для втілення поставленої задачі, надалі програмування за створеною концепцією, тестування програмного продукту та впровадження документації.

ВИСНОВКИ

Метою даної кваліфікаційної роботи була розробка тривимірного аркадного ігрового додатку, який забезпечує користувача захоплюючим ігровим процесом та дозволяє скоротити час. При розробці були використані сучасні методи та засоби для створення тривимірних ігор.

Розроблений проект призначений для розважальних цілей, але також сприяє розвитку мозкових здібностей користувача, поліпшуючи його реакцію та сприйнятливість рухомих ігрових об'єктів.

Програма працює під управлінням операційної системи Microsoft Windows, яка є широко поширеною серед цільової аудиторії продукту. Розробка виконувалась на мові програмування C#, яка дозволяє точно маніпулювати ресурсами та забезпечила досягнення задовільних результатів щодо швидкості роботи програми. В якості додаткового засобу був використаний ігровий двигун Unity Engine. Основним завданням було забезпечити задоволення користувачів під час гри.

У "Економічному розділі" була визначена трудомісткість розробки програмного забезпечення (821.91 чол-год), розраховані витрати на створення програмного забезпечення (144355 грн.) і приблизний період розробки (4.6 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D0%B0_(%D0%B3%D1%80%D0%B0)
2. https://itechua.com/internet/210303
3. https://gamedev.dou.ua/forums/topic/38048/
4. https://jrnl.nau.edu.ua/index.php/PIU/article/view/16847/24128
5. http://dspace.tnpu.edu.ua/bitstream/123456789/23649/1/IJREL2021.pdf
6. https://freeprog.org.ua/images/files/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0%20-%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%B2%20Unity.pdf
7. https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/28849/1/Unity%20December%202019%20-%20January%202020%20%283%29.pdf
8. https://sd.blackball.lv/books/18950-unity-games-by-tutorials-3rd-edition-2018?mode=read
9. https://resources.unity.com/games/level-up-your-code-with-game-programming-patterns
10. https://xn--d1ag.xn--e1a4c/tmp/Bibl_progr_Sb_187kn/%D0%98%D0%B7%D1%83%D1%87%D0%B0%D0%B5%D0%BC%20C%23%20%D1%87%D0%B5%D1%80%D0%B5%D0%B7%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D1%83%20%D0%B8%D0%B3%D1%80%20%D0%BD%D0%B0%20Unity.%205-%D0%B5%20%D0%B8%D0%B7%D0%B4.%20-%202022.pdf
11. https://dusithost.dusit.ac.th/~juthawut_cha/download/Unity3D%20Manual.pdf
12. https://docs.unity3d.com/Manual/class-QualitySettings.html

13. https://images.response.unity3d.com/Web/Unity/%7Ba58bfd8b-fc4d-4a60-a340-328a4a994c90%7D_2022-01-DG-Essential-Dev-Ops-Practices-e-book.pdf
14. https://cdn.unity3d.com/media/TheGameDesignerPlaybook_EBook.pdf
15. https://ptgmedia.pearsoncmg.com/images/9780672336966/samplepages/0672336960.pdf
16. https://www.researchgate.net/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey
17. https://www.computo-visual.org/Videojuegos/Libros/Learning%20C%23%20by%20Developing%20Games%20with%20Unity%203D.pdf
18. https://gamedevacademy.org/wp-content/uploads/2018/04/Learn-Unity-by-Creating-a-3D-Multi-Level-Platformer-Game.pdf
19. https://www.tutorialspoint.com/unity/unity_tutorial.pdf
20. https://www.cs.purdue.edu/cgvlab/courses/490590VR/notes/Introduction%20to%20Unity.pdf

ДОДАТОК А

КОД ПРОГРАМИ

CameraMove.cs

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

// Переміщення камери за гравцем
public class CameraMove : MonoBehaviour
{
    [SerializeField] private Transform player;
    private Vector3 move;

    void Start()
    {
        move = transform.position - player.position;
    }

    void LateUpdate()
    {
        Vector3 newPos = new Vector3(player.position.x + move.x, transform.position.y, player.position.z + move.z);

        transform.position = Vector3.Lerp(transform.position, newPos, 1f);
    }
}
```

Restart.cs

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

```
public class Restart : MonoBehaviour
{
    public void lvlRestart()
    {
        SceneManager.LoadScene(0);
    }
}
```

EndScene.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using static UnityEditor.Timeline.TimelinePlaybackControls;

public class EndScene : MonoBehaviour
{
    public GameObject liveCoins;
    public GameObject liveDistance;
    public GameObject endScreen;
    public GameObject fadeOut;

    private bool gameOver = false;

    void Start()
    {
        //вмикаємо зіткнення гравця з перешкодою
        Obstacle.OnPlayerCollision += EndGame;
    }

    private void OnDestroy()
    {
        // вимикаємо у разі знищення скрипта
        Obstacle.OnPlayerCollision -= EndGame;
```

```
    }

    void EndGame()
    {
        if (!gameOver)
        {
            gameOver = true;

            // Зупиняємо ігрові об'єкти і показуємо текст програшу
            liveCoins.SetActive(false);
            liveDistance.SetActive(false);
            endScreen.SetActive(true);
            fadeOut.SetActive(true);



            StartCoroutine(AdditionalActions());
        }
    }

    IEnumerator AdditionalActions()
    {

        yield return new WaitForSeconds(5);

    }
}
```

Obstacle.cs

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

// Подія зіткнення гравця з перешкодою
public delegate void PlayerCollisionEventHandler();
```

```
public class Obstacle : MonoBehaviour
{
    public GameObject player;
    public GameObject characterModel;
    public GameObject levelControl;

    // Подія зіткнення гравця з перешкодою
    public static event PlayerCollisionEventHandler OnPlayerCollision;

    void OnTriggerEnter(Collider other)
    {
        this.gameObject.GetComponent<BoxCollider>().enabled = false;
        player.GetComponent<PlayerMovement>().enabled = false;
        characterModel.GetComponent<Animator>().Play("Stumble Backwards");
        levelControl.GetComponent<LevelDistance>().enabled = false;

        // Генеруємо подію зіткнення гравця з перешкодою
        if (OnPlayerCollision != null)
        {
            OnPlayerCollision();
        }
    }
}
```

LevelDistance.cs

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

// відстань, пройдена гравцем
public class LevelDistance : MonoBehaviour
{
    public GameObject disDisplay;
```

```
    public GameObject disEndDisplay;
    public int disRun;
    public bool addingDis = false;
    public float disDelay = 0.35f;

    void Update()
    {
        if (addingDis == false)
        {
            addingDis = true;
            StartCoroutine(AddingDis());
        }
    }

    IEnumerator AddingDis()
    {
        disRun++;
        disDisplay.GetComponent<TMPro.TextMeshProUGUI>().text = "" + disRun;
        disEndDisplay.GetComponent<TMPro.TextMeshProUGUI>().text = "" + disRun;

        yield return new WaitForSeconds(disDelay);
        addingDis = false;
    }
}
```

CoinCounter.cs

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

//лічильник монет
public class CoinCounter : MonoBehaviour
{
    public static int coinCount;
```

```
    public GameObject coinDisplay;
    public GameObject coinDisplayEnd;

    //вивід кількості
    void Update()
    {
        coinDisplay.GetComponent<TMPro.TextMeshProUGUI>().text = "" + coinCount;

        coinDisplayEnd.GetComponent<TMPro.TextMeshProUGUI>().text = "" + coinCount;
    }
}
```

ColectCoin.cs

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

//збір та видалення монет
public class ColectCoin : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        CoinCounter.coinCount += 1;
        this.gameObject.SetActive(false);
    }
}
```

RotateObj.cs

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

//обертання монети
```

```
public class RotateObj : MonoBehaviour
{
    public int rotateSpeed = 1;

    void Update()
    {
        // Обертаємо об'єкт по осі Y
        transform.Rotate(0, rotateSpeed, 0, Space.World);
    }
}
```

GenerateLevel.cs

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

//генерація дороги
public class GenerateLevel : MonoBehaviour
{
    public GameObject[] road;
    public int zPosition = 50;
    public bool creatingRoad = false;
    public int roadNum;

    void Update()
    {
        if (creatingRoad == false)
        {
            creatingRoad = true;
            StartCoroutine(GenerateRoad());
        }
    }

    IEnumerator GenerateRoad()
```

```
    {
        //випадкова дорога зі списку
        roadNum = Random.Range(0, 3);
        Instantiate(road[roadNum], new Vector3(0, 0, zPosition), Quaternion.identity);
        zPosition += 50;
        yield return new WaitForSeconds(6);
        creatingRoad = false;
    }

}
```

PlayerMovement.cs

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

//переміщення гравця
public class PlayerMovement : MonoBehaviour
{
    public float speed = 5;
    public float rightLeftSpeed = 5;
    const float speedMaximum = 10;

    public bool Jumping = false;
    public bool comingDown = false;
    public GameObject playerObject;

    void Start()
    {
        StartCoroutine(speedUp());
    }

    void Update()
    {
        //рухаємо гравця вперед
```

```
        transform.Translate(Vector3.forward * Time.deltaTime * speed, Space.World);

        if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))
        {
            if (this.gameObject.transform.position.x < LevelBoundary.sideRight)
            {
                transform.Translate(Vector3.left * Time.deltaTime * rightLeftSpeed * -1);
            }
        }

        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.RightArrow))
        {
            if (this.gameObject.transform.position.x > LevelBoundary.sideLeft)
            {
                transform.Translate(Vector3.left * Time.deltaTime * rightLeftSpeed);
            }
        }

        if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow) ||
Input.GetKey(KeyCode.Space))
        {
            if (Jumping == false)
            {
                Jumping = true;
                playerObject.GetComponent<Animator>().Play("Jump");

                StartCoroutine(JumpSequence());
            }
        }

        if (Jumping == true)
        {
            if (comingDown == false)
            {
                transform.Translate(Vector3.up * Time.deltaTime * 4, Space.World);
```

```
            }

            if (comingDown == true)
            {
                transform.Translate(Vector3.up * Time.deltaTime * -4, Space.World);
            }
        }

    }

    IEnumerator JumpSequence()
    {
        float initialHeight = transform.position.y;

        yield return new WaitForSeconds(0.4f);
        comingDown = true;
        yield return new WaitForSeconds(0.4f);
        Jumping = false;

        comingDown = false;
        playerObject.GetComponent<Animator>().Play("Standard Run");
        transform.position = new Vector3(transform.position.x, initialHeight,
transform.position.z);
    }

    // прискорення
    private IEnumerator speedUp()
    {
        yield return new WaitForSeconds(10);
        if (speed<speedMaximum)
        {
            speed += 1;
            StartCoroutine(speedUp());
        }
```

```
    }
}
```

LevelBoundary.cs

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

//реалізація границі дороги
public class LevelBoundary : MonoBehaviour
{
    public static float sideRight = 3.5f;
    public static float sideLeft = -3.5f;

    public float internalRight;
    public float internalLeft;

    void Update()
    {
        internalRight = sideRight;
        internalLeft = sideLeft;
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Швець.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Швець.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і скомпільовану програму.
Презентація	
Презентація_Швець.pptx	Презентація кваліфікаційної роботи.