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 *Шагояна Роберта Арамовича*
(ПІБ)

академічної групи *122-18ск-2*
(шифр)

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

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

на тему: *Проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity*

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

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск-2* *Шагояна Роберта Арамовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity*

затверджена наказом ректора НТУ «ДП» від *07.06.2021* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***24.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Шагоян Р.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 85 с., 26 рис., 3 дод., 21 джерел.

Об'єкт розробки: ігровий додаток з використанням неевклідової геометрії.

Мета кваліфікаційної роботи: створення ігрового додатку що допоможе розвинути просторове мислення, та дати розуміння нетипової поведінки ігрового додатку.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного додатка, що надає можливість провести час із задоволенням.

Актуальність розробленого програмного продукту полягає в створенні такого додатку, який матиме попит серед цільової аудиторії – гравців у відеоігри.

Список ключових слів: ІГРОВИЙ ДВИГУН, КОМПОНЕНТ, КОМП’ЮТЕР, ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК, АЛГОРИТМ, ПРОЕКТУВАННЯ, МЕНЮ, ВКЛАДКА, ДОДАТОК.

ABSTRACT

Explanatory note: 85 p., 26 img., 3 apps., 21 sources.

Object of development: game application using non-Euclidean geometry.

The purpose of the diploma project is creating a game application that will help develop spatial thinking, and give an understanding of the atypical behavior of the game application.

The introduction examines the current problem statement, specifies the purpose of the qualification work and the area of its application, justifies the relevance of the topic and specifies the problem statement.

In the first section carries out the analysis of the subject area, determines the relevance of the task and the dedication of the development, creates task statement, the software and hardware requirements of the product, specifies technologies and tools for development.

In the second section analyzes the existing solutions, chose the platform for development, creates design and finish development of the product, describes the algorithm, structure and architecture solutions in the system, defines the input and output data, describes characteristics of the technical resources used, describes how to run a program, features of user interaction, differences between serve and client parts of product.

The economic section defines the complexity of the information system, calculates the cost of work on creating the application and defines the time for its creation.

The practical significance is creation a product which provides an opportunity to spend time with pleasure.

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

Keywords: GAME ENGINE, COMPONENT, COMPUTER, INFORMATION SYSTEM, ACCOUNTING, ALGORITHM, DESIGN, MENU, TAB, APPENDIX.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

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;

VR — Virtual Reality;

Mac OS — Macintosh Operating System;

UE — Unity Engine.

ЗМІСТ

ВСТУП

Завдання даної кваліфікаційної роботи та об'єкт його діяльності безпосередньо пов'язані з напрямом підготовки та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 122 «Комп’ютерні науки».

Тематикою кваліфікаційної роботи є проектування та розробка ігрового додатку у жанрі Arcade на движку Unity.

Метою кваліфікаційної роботи є вивчення засобів роботи з багатьма підсистемами операційної системи. Перш за все це події вводу в системі та взаємодія з ними. Ці навички є дуже актуальними в сфері розробки програмного забезпечення для контролю доступу та моніторингу активності процесів та користувачів. Також вони допоможуть краще зрозуміти механізми роботи з графічними об’єктами та їх взаємодію на досить низькому рівні.

Дана робота дозволяє познайомитись з методами розробок ігрових додатків та розуміння багатовимірних ігрових додатків.

Основною вимогою до даного програмного забезпечення є стабільність роботи. Оскільки Unity Engine – двигун досить вимогливий до графічного процесору, що зумовлено якістю графіки, швидкість роботи додатку на не потужних персональних комп’ютерах може бути не дуже стабільною.

Отже задачею даної кваліфікаційної роботи є проектування та розробка ігрового додатку, що забезпечує гравця захоплюючим ігровим процесом у жанрі Arcade.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Розробка відеогри – це процес створення відеогри, яким займається розробник відеоігор, котрий може бути як однією людиною так і компанією з сотнями співробітників. Гра може розроблятися як силами кількох людей з обмеженим бюджетом, так і спираючись на фінансування від видавця. Тривалість та вартість розробки залежить від складності проекту. Оскільки відеогра є комп'ютерною програмою, її робота, технічні можливості, контент та ігровий процес, забезпечується програмним кодом. Розробка гри включає такі ж етапи, що і розробка програмного забезпечення, але передбачає більше роботи над контентом і створення ігрових механік. Сучасні ігри здебільшого засновані на готових програмних модулях – ігрових ядрах, де вже реалізовані базові функції, здатні зв'язувати воєдино графіку, звук, об'єкти і їх рухи. Щоб налаштувати ядро для реалізації конкретного задуму програмісти доопрацьовують його, додаючи потрібні функції. Існують як вільні ігрові ядра, доступні будь-кому, так і ті, що вимагають отримання ліцензії на їх використання. Деякі ядра розраховані на створення ігор конкретного жанру, інші - універсальні. Не кожне ядро може забезпечити однакові внутрішньоігрові можливості та рівень графіки.

Деякі види ядер дозволяють створювати ігри для різних платформ, наприклад, Unity Engine підтримує розробку програмного забезпечення для PC, Xbox 360, PlayStation 4, PlayStation VR, Oculus, Android, iOS та багато інших.

Будь-яке ігрове ядро надає безліч функціональних можливостей, які застосовуються в різних іграх. Реалізована на такому ядрі гра отримує всі ці

функціональні можливості, крім того, додаються її власні ігрові ресурси і код ігрового сценарію (скрипти). Unity надає можливість моделювання фізичних середовищ, зображення ігрового світу в екранному просторі (Screen Space Ambient Occlusion, SSAO), динамічні тіні та багато іншого. Подібним набором функціональних можливостей можуть похвалитися багато ігрових ядер, але у Unity є дві основні переваги перед іншими передовими інструментами розробки ігор: надзвичайно продуктивний візуальний робочий процес і потужна мультиплатформенна підтримка.

Візуальний робочий процес має певні переваги. У той час, як інші інструменти розробки ігор часто являють собою збірку розрізнених частин, які необхідно контролювати, чи, можливо, бібліотеку, для роботи з якою потрібно налаштовувати власну інтегровану середу розробки (Integrated Development Environment, IDE), послідовність збірки ті інше в цьому роді, робочий процес в Unity прив’язаний до складно продуманого візуального редактору. У цьому редакторі розробник компонує сцени майбутньої гри, пов’язуючи ігрові ресурси і код в інтерактивні об’єкти. Саме він дозволяє швидко і раціонально створювати професійні ігри, забезпечуючи продуктивність праці розробників і надаючи в їх розпорядження перелік найсучасніших технологій в сфері розробки відеоігор.

У наборі інструментів Unity існує потужна мультиплатформенна підтримка. В даному випадку, мається на увазі не тільки місце розгортання (ви можете розгорнути гру на персональному комп'ютері, в Інтернеті, на мобільному пристрої або на консолі), а й інструменти розробки. Існує досить небагато ігрових ядер, що підтримують таку ж кількість цільових платформ розгортання, і жодне з них не робить операцію розгортання настільки легкою.

Крім цих двох основних переваг, слід відзначити ще одну - це забезпечення модульною системою компонентів, яка використовується для конструювання ігрових об’єктів. «Компонентами» в такій системі є комбіновані пакети функціональних елементів, тому об’єкти створюються як набори компонентів, а

не як жорстка ієрархія класів. Іншими словами, компонентна система являє собою альтернативний (і зазвичай більш гнучкий) підхід до об'єктно-орієнтованого програмування, в якому ігрові об'єкти створюються шляхом об'єднання, а не успадкування. Порівняння підходів демонструє діаграма на рисунку 1.1.

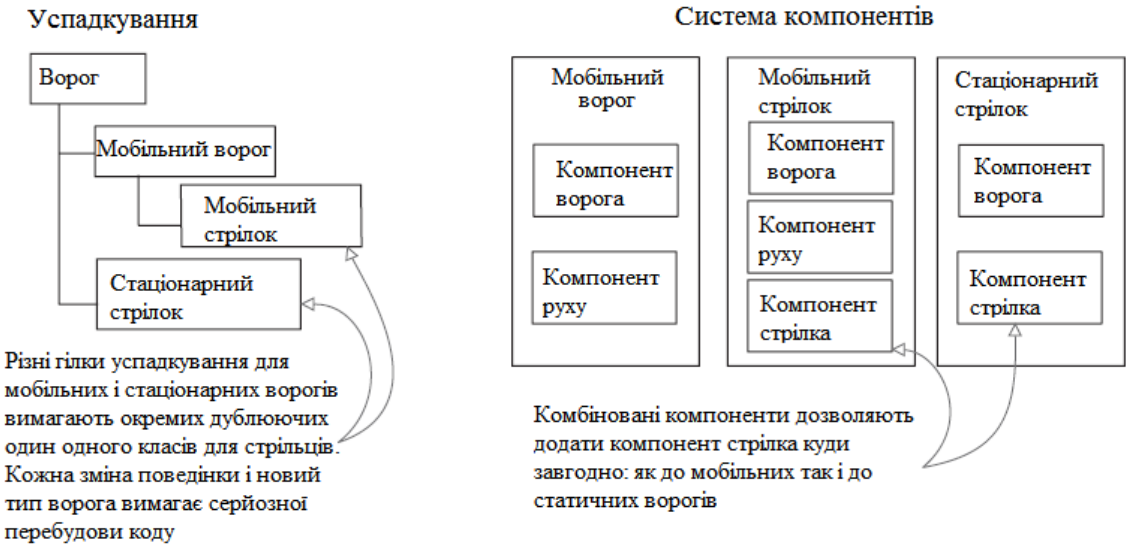

Рисунок 1.1. Порівняння успадкування і компонентної системи

В компонентній системі об'єкт існує в горизонтальній ієрархії, тому різні об'єкти складаються з різних наборів компонентів, а не зі структури успадкування, в якій різні об'єкти виявляються на різних гілках дерева. Така компоновка полегшує створення прототипів, тому що опрацювати потрібний набір компонентів значно швидше і простіше, ніж перебудовувати ланцюжок успадкування при зміні кожного об'єкта. Крім цього, програміст має можливість написати код, який реалізує його власну компонентну систему, але в Unity вже існує цілком надійний варіант такої системи, органічно вбудований в візуальний редактор. Тому розробник має можливість не тільки програмно керувати компонентами, але і встановлювати і розривати зв'язок між ними в редакторі. Можливості розробки не обмежуються складанням об'єктів з готових деталей; в коді програміст може користуватися успадкуванням і всіма

напрацьованими на його базі паттернами проектування.

Для створення будь-якої гри в середовищі Unity, програміст повинен, як мінімум, володіти однією з доступних (на Unity) мов програмування: C# або JavaScript. Мова програмування C# має ряд переваг перед мовою JavaScript і значно менше недоліків. Однією з переваг є той факт, що мова C# строго типізована, чого не можна сказати про JavaScript. Серед програмістів в наш час існують різні точки зору з приводу того, чи є динамічна перевірка типів оптимальним підходом, наприклад до веб-розробки, але при написанні програм для певних платформ (таких, як Windows) вона часто вигідна, а часом необхідна навіть статична типізація. У Unity навіть додана директива #pragma, що примусово забезпечує статичну перевірку типів в мові JavaScript. Хоча з технічної сторони таке цілком допустимо, але при цьому порушується один з основних принципів функціонування JavaScript. Тому краще обрати мову зі строгою типізацією - C#.

Поведінка ігрових обєктів контролюється за допомогою компонентів (Components), які приєднуються до них. Ядро Unity дозволяє програмісту створювати власні компоненти, використовуючи сценарії (скрипти). Вони дозволяють активувати ігрові події, змінювати параметри компонентів, і відповідати на введення користувача будь-яким способом.

Програмування здійснюється не всередині Unity, код існує у вигляді окремих файлів, місце розташування яких програміст повідомляє Unity. Файли сценаріїв можуть створюватися в додатку Unity, але в будь-якому випадку, розробнику буде необхідно використовувати текстовий редактор або IDE, де буде писатися код для цих спочатку порожніх файлів.

У 1970-х відеоігри стали одним з найпопулярніших розваг. У них грали вдома або в барах і кафе, де вони встановлювалися у вигляді гральних автоматів в коридорах, які нагадували аркаду. Звідси назва самого першого жанру електронних ігор - «аркади» (найперші гри, такі як «Зоряні війни», «Понг», «Покемон», «Арканоід» - все це аркадні ігри). Незважаючи на деяку

примітивність, цей жанр є цілком оригінальним і його важко порівняти з будь-яким різновидом відомих звичайних ігор. Крім того, в ньому чітко видно, як ігровий досвід знаходить залежність від технологічних пристроїв (екрану, електронної начинки, спеціальних приладів управління).

У світовій практиці аркадами називаються ігри для аркадних ігрових автоматів. Це не окремий жанр ігор, а, скоріше, ігровий напрямок. Комп'ютерна або відеогра називається «аркадною» в тому випадку, якщо вона безпосередньо перенесена з автомата або ж схожа по концепції з іграми для автоматів.

У сучасному розумінні, Аркада (англ. Arcade) - це жанр відеоігор, ігровий процес (геймплей) яких заснований на швидкій реакції гравця, мінімальному ступені свободи керованих персонажів і простому управлінні.

Класичні аркади характеризуються зазначеними надалі властивостями. Гра на одному екрані. У класичних аркадах весь ігровий процес зосереджений на одному екрані. Перш за все це обумовлено історично, що сталося через технічні обмеження, але в той же час це значно впливало на гейм дизайн. Так, гравці в будь-який момент часу могли бачити весь ігровий світ і приймати рішення, виходячи з повної інформації про його стан. Безліч ігор жанру мало більше одного екрану, і вони змінювали один одного як рівні. Характерними прикладами тут є Joust, Pac-Man, Mario Bros. Нескінченна гра. Потенційно гравці можуть грати в аркаду нескінченне час, і відповідно, не можуть виграти. Це впливало на те, що гравці робили виклик самі собі - наскільки довго вони зможуть протриматися. В аркадах гравець ніколи не вигравав, і кожна гра закінчувалася поразкою. У той же час, ігри проектувалися таким чином, щоб з часом гравцеві ставало все складніше, і таким чином нескінченна гра пропонувала нескінченну складність. Дана ситуація змінилася з появою ринку домашніх комп'ютерів, коли видавці змінили своє бажання, щоб гравці проходили гру і після цього хотіли купити нову. Ігровий рахунок. Практично всі класичні аркади включають в себе ігровий рахунок. Коли гравець отримує бали за виконання різних цілей або завдань. Ігрові бали дозволяють гравцеві

зрозуміти, наскільки добре він грав, незважаючи на те, що виграти неможливо. При цьому типовий час гри середнього гравця становить близько двох хвилин, а у досвідченого до десятків хвилин. На підставі даної особливості у аркад, як правило, є таблиця рекордів, де гравець може поруч зі своїми результатами ввести свої ініціали, і тим самим порівнювати себе з іншими гравцями. Швидке навчання, простий ігровий процес. Для класичних аркад характерним є те, що гравцям легко навчитися геймплею, але стає практично неможливим стати майстром в грі через її складності. Разом з тим, якщо гравець гине в аркаді, то це практично завжди відбувається з його вини. В таких іграх немає «спеціальних комбінацій клавіш», які гравець повинен вивчити з документації для того, щоб зробити щось особливе. Дуже мало ігор розширюють концепт за допомогою очок здоров'я, щитів або таблеток сили. Це пов'язано з тим, що з комерційної точки зору аркадах було необхідно охопити якомога більший спектр гравців, тобто, фактично кожна людина в барі або магазині повинен бути здатним підійти і спробувати зіграти. У той же час, простий ігровий процес не має на увазі що він «поганий» або «обмежений», - він може бути «елегантним» та «відполірованим».

Жанр Arcade (аркада) є одним із найстаріших в ігровій індустрії і пройшов довгий шлях розвитку. Незважаючи на те, що його важко назвати самим популярним, він справив великий вплив на інші жанри. Головною перевагою аркад є короткий за часом, але інтенсивний ігровий процес. До сих пір розробники створюють все нові і нові проекти, що поєднують в собі елементи як аркадних, так і інших жанрів, намагаючись привернути увагу більшої кількості людей і створити щось цікаве і унікальне. Незважаючи на те, що пік популярності аркад припав на кінець 20-го століття, але і до цього дня у цього жанру залишається величезна кількість шанувальників, які не дають цьому жанру закінчити своє існування.

1.2. Призначення розробки та постановка задачі

Ціль розробки - проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity.

Розроблений ігровий додаток буде використовуватися гравцями, які бажають відпочити та гарно провести час. З цим допоможуть відеоігри - вони здатні відвернути увагу від навколишніх проблем та подарувати гравцю частку позитиву.

Основною метою програми є організація ігрового процесу (геймплею), що базується на взаємодії користувача і ігрового пристрою за допомогою візуального інтерфейсу.

Ігровий додаток призначений для користувачів ПК з операційною системою 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 з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

- освітня програма спеціальності 122 “Комп’ютерні науки”;

- навчальний план та графік навчального процесу;

- наказ ректора Національного технічного університету “Дніпровська політехніка” № 317-с від 07.06.2021р;

- завдання на кваліфікаційну роботу на тему “Проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity”.

1.4. Постановка завдання

Головною метою кваліфікаційної роботи є проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity. Ігровий додаток призначений для користувачів ПК з операційною системою Windows. Основними характеристиками розробки повинні бути:

- реалізація ігрових механік для забезпечення гравців можливостями керування ігровим процесом всередині тривимірного ігрового простору;
- розробка декількох ігрових сцен та можливість переходу між ними;
- створення інтерфейсу ігрового додатку за допомогою можливостей ядра Unity Engine;
- створення головного меню для ігрового додатку;
- тестування розробленого ігрового додатку.

Поставлена задача може бути досягнута при виконанні наступних вимог:

- вивчення предметної галузі завдання;
- проведення порівняльної характеристики можливостей аналогічних програм;
- вибір платформи розробки;
- імпортування тривимірних об’єктів з магазину Asset Store в Unity Engine;
- написання програмного коду.

Кінцевим результатом розробки має бути ігровий додаток для користувачів ПК з операційною системою Windows, в якому реалізовано ігровий процес (геймплей), що базується на взаємодії користувача, апаратної складової ПК, розроблених механік ігрового додатку та візуального інтерфейсу.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для коректного запуску додатку потрібно встановити Directx версії 11, на більш ранніх версіях додаток не може бути запущений, через набір використаних функцій у ігровому двигуні.

DirectX - це набір API, розроблених для простого і ефективного вирішення завдань з мультимедіа, ігровим і відео-програмуванням, під операційні системи Windows від Microsoft.

Версія драйвера відеокарти, встановленого в системі повинна бути оновлена, оскільки від цього залежить коректна робота ігрового додатку.

Оскільки проект вже запакований у виконувальний файл, від користувача потрібно тільки запустити його.

Ввід даних повинен бути реалізований через периферійні пристрої для вводу інформації та їх взаємодію з інтуїтивно зрозумілим для користувачів інтерфейсом розробленого ігрового додатку . Вивід даних здійснюється за допомогою монітору.

Також при запуску можливі оповіщення про можливі проблеми:

Warning — нестандартна ситуація з програмою. Не являється критичною, але користувачеві варто приділити їй увагу.

Error — помилка в роботі програми. Деякі функції або файли могли бути недоступні.

1.5.2. Вимоги до інформаційної безпеки

Оскільки додаток вже запакований, якихось вимог до інформаційної безпеки не потрібно, достатньо просто дотримуватись ієрархії внутрішніх файлів, для уникнення можливих збоїв додатку. Крім цього у користувача немає

необхідності реєструватися і створювати аккаунт 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 64 bit;
- процесор: Intel Core i3 (1,4 ГГц);
- ОЗП: 1 Гб;
- відеокарта NVidia gtx 450, 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

Сучасні комп'ютерні ігри вимагають чималі вимоги до комп'ютерної апаратної частини. Основою є швидкий центральний процесор (англ. CPU) для коректної роботи внутрішньоігрових механізмів. Багатоядерні процесори дали можливість обробляти декілька завдань одночасно, що означало більш складну графіку, більш просунутий штучний інтелект і внутрішньоігрову фізику.

Тривимірні ігри вимагають потужного графічного процесора (англ. GPU), який прискорює процес відтворення складних графічних сцен в режимі реального часу. Також можливе використання декількох графічних процесорів в одному комп'ютері за допомогою таких технологій, як Scalable Link Interface (NVidia) або CrossFire (ATI).

Для забезпечення тривимірного якісного звуку в іграх необхідно мати

звукову карту (вбудовані аналоги часто не мають функцій для 3D і більш чистого звуку).

Для керування ігровим персонажем використовуються клавіатури, миші, джойстики для авіасимуляторів, керма для гоночних ігор і геймпади для ігор, орієнтованих на консольне управління.

На продуктивність і геймплей також впливає версія драйвера відеокарти, встановленого в системі.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення інформаційної системи

Результатом даної кваліфікаційної роботи має бути тривимірний ігровий додаток у жанрі Arcade створений за допомогою движку Unity. Розроблений ігровий додаток буде використовуватися гравцями, які бажають відпочити та гарно провести час.

Основне призначення ігрового додатку:

- організація ігрового процесу (геймплею);
- ігровий додаток призначений для використання на платформі ПК (Windows);
- спроектований додаток забезпечено функціональним інтерфейсом, що складається з екрану головного меню та екрану з ігровим процесом;

Головне меню надає користувачу можливість обрати певну опцію програми. При обиранні пункту меню – запуск ігрового процесу, буде запущено ігровий процес, в якому користувачу надається можливість протестувати функціонал гри, що складається з п’яти головних елементів:

- аркадна система;
- механіка поведінки ігрових об’єктів у тривимірному просторі;
- система платформ;
- particles-система;
- рахунок гравця.

Елемент аркадної системи полягає у тому, що взаємодія розроблених механік та ігрових елементів реалізує ігровий процес (геймплей) який є характерним для ігор у жанрі Arcade.

Механіка поведінки ігрових об’єктів у тривимірному просторі — це результат роботи розроблених функцій, які встановлюють правила взаємодії ігрових об’єктів, створених за допомогою Unity Engine. Це досягається завдяки компонентній системі, що використовується в Unity Engine: на певний ігровий об’єкт створюється скрипт (сценарій), написаний на мові програмування C#.

Система платформ — у розробленому ігровому додатку було реалізовано систему платформ, яка є однією з особливостей жанру Arcade. Було розроблено особливий вид ігрового процесу — Endless Runner, підтип жанру Arcade, в якому гравцеві необхідно пересуватись нескінченно з’являючимися платформами.

Particles-система — система створює частки, які служать візуальними графічними ефектами, в випадкових точках в межах заздалегідь визначеного простору. Ці частки можуть мати форму, наприклад, сфери або конуса. Система визначає час життя самої частинки, діапазон створення, і коли час життя частинки закінчується — система її знищує.

Рахунок гравця — в ігровому додатку було реалізовано елемент ігрового процесу, який дозволяє гравцеві спостерігати його внутрішньоігровий рахунок.

2.2. Опис застосованих математичних методів

При розробці тривимірного ігрового додатку у жанрі Arcade, було використано такі математичні методи: операції над векторами та арифметичні оператори додавання, віднімання, множення, ділення.

Операції над векторами, комутативність: $\overline{a} + \overline{b} = \overline{b} + \overline{a}$.

Арифметичні оператори: “+”, “-”, “*”, “/”.

2.3. Опис використаної архітектури та шаблонів проектування

При розробці тривимірного ігрового додатку у жанрі Arcade, було обрано класичний шаблон програмування MVC (Model-View-Controller) - це схема, що передбачає поділ даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компонента, щоб кожен з них можна було незалежно модифікувати.

Model відповідає за внутрішню логіку ігрового додатку. В проекті цей елемент шаблону проектування охоплює зберігання даних, а також правила і алгоритми обробки інформації.

View відповідає за відображення даних Моделі. Цей рівень виконує лише своєчасне відображення стану Model. За обробку дій користувача відповідає наступний рівень шаблону проектування MVC - Controller.

Controller встановлює зв'язок між рівнем Model і діями користувача, отриманими в результаті взаємодії з рівнем View. Фактично на кожну дію, яку може зробити користувач на рівні View, повинен бути визначений оброблювач подій в Controller. Цей оброблювач виконає відповідні маніпуляції над Model і в разі необхідності повідомить View про наявність змін.

При розробці тривимірного ігрового додатку було використано вбудоване в Unity Engine IDE - Microsoft Visual Studio 2019. Завдяки вікну Оглядач рішень, можна зручно переглянути структуру проекту та скрипти на мові програмування C#. За допомогою функціоналу Visual Studio 2019, файли проекту зручно збираються у рішення (solution). Рішення являє собою структуру для організації проектів в Visual Studio. Воно зберігає відомості про стан проектів в двох файлах:

- SLN-файл (на основі тексту, загальний);
- suo-файл (двійковий, параметри рішення для конкретного користувача).

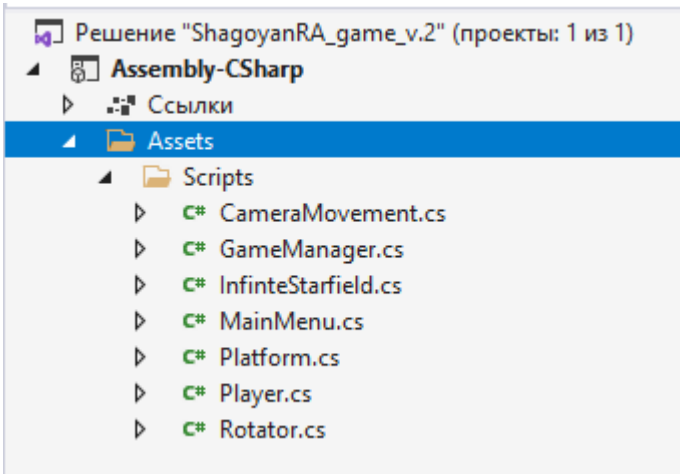

Рис. 2.1. Структура організації файлів скриптів у Visual Studio 2019

2.4. Опис використаних технологій та мов програмування

Для проектування та розробки тривимірного ігрового додатку у жанрі Arcade було використано технології Unity 2020.3.11f1, інтегроване середовище розробки Microsoft Visual Studio 2019 та мову програмування C#.

Середовище Unity дозволяє створювати ігрові додатки під більшість популярних платформ. За допомогою технологій даного ядра розробляються ігри, які можливо запускати на персональних комп'ютерах (які працюють під Windows, MacOS, Linux), на смартфонах і планшетах (iOS, Android, Windows Phone) та на ігрових консолях (PS, Xbox, Wii).

Будь-яке ігрове ядро надає безліч функціональних можливостей, які застосовуються в різних іграх. Реалізована на такому ядрі гра отримує всі ці функціональні можливості, крім того, додаються її власні ігрові ресурси і код ігрового сценарію (скрипти). Unity надає можливість моделювання фізичних середовищ, карти нормалей, зображення ігрового світу в екранному просторі (Screen Space Ambient Occlusion, SSAO), динамічні тіні та багато іншого. Подібним набором функціональних можливостей можуть похвалитися багато ігрових ядер, але у Unity є дві основні переваги перед іншими передовими

інструментами розробки ігор: надзвичайно продуктивний візуальний робочий процес і потужна мультиплатформенна підтримка.

Візуальний робочий процес має певні переваги. У той час, як інші інструменти розробки ігор часто являють собою збірку розрізнених частин, які необхідно контролювати, чи, можливо, бібліотеку, для роботи з якою потрібно налаштовувати власну інтегровану середу розробки (Integrated Development Environment, IDE), послідовність збірки ті інше в цьому роді, робочий процес в Unity прив'язаний до складно продуманого візуального редактору. У цьому редакторі розробник компонує сцени майбутньої гри, пов'язуючи ігрові ресурси і код в інтерактивні об'єкти. Саме він дозволяє швидко і раціонально створювати професійні ігри, забезпечуючи продуктивність праці розробників і надаючи в їх розпорядження перелік найсучасніших технологій в сфері розробки відеоігор.

У наборі інструментів Unity існує потужна мультиплатформенна підтримка. В даному випадку, мається на увазі не тільки місце розгортання (ви можете розгорнути гру на персональному комп'ютері, в Інтернеті, на мобільному пристрої або на консолі), а й інструменти розробки. Ця незалежність від платформи стала результатом того, що спочатку інструмент Unity призначався виключно для комп'ютерів Mac, а пізніше був перенесений на машини з операційними системами сімейства Windows. Існує досить небагато ігрових ядер, що підтримують таку ж кількість цільових платформ розгортання, і жодне з них не робить операцію розгортання настільки легкою.

Крім цих двох основних переваг, слід відзначити ще одну - це забезпечення модульною системою компонентів, яка використовується для конструювання ігрових об'єктів. «Компонентами» в такій системі є комбіновані пакети функціональних елементів, тому об'єкти створюються як набори компонентів, а не як жорстка ієрархія класів. Іншими словами, компонентна система являє собою альтернативний (і зазвичай більш гнучкий) підхід до об'єктно-орієнтованого програмування, в якому ігрові об'єкти створюються шляхом

об’єднання, а не успадкування. Порівняння підходів демонструє діаграма на рисунку 2.2.

В компонентній системі об’єкт існує в горизонтальній ієрархії, тому різні об’єкти складаються з різних наборів компонентів, а не зі структури успадкування, в якій різні об’єкти виявляються на різних гілках дерева. Така компоновка полегшує створення прототипів, тому що опрацювати потрібний набір компонентів значно швидше і простіше, ніж перебудовувати ланцюжок успадкування при зміні кожного об’єкта.

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

Крім цього, програміст має можливість написати код, який реалізує його власну компонентну систему, але в Unity вже існує цілком надійний варіант такої системи, органічно вбудований в візуальний редактор. Тому розробник має можливість не тільки програмно керувати компонентами, але і встановлювати і розривати зв’язок між ними в редакторі. Можливості розробки не обмежуються складанням об’єктів з готових деталей; в коді програміст може користуватися успадкуванням і всіма напрацьованими на його базі паттернами проектування.

Інтерфейс Unity поділяється на кілька частин: вкладка Scene, вкладка Game, панель інструментів, вкладка Hierarchy, панель Inspector, вкладки Project і Console (рисунок 2.3).

У кожної частини інтерфейсу є власне призначення, при цьому всі вони грають важливу роль в процесі створення гри:

- перегляд файлів здійснюється на вкладці Project;
- розміщені на сцені об’єкти відображається на вкладці Scene;
- панель інструментів містить елементи керування сценою;
- на вкладці Hierarchy надається можна змінювати зв'язок між об’єктами;
- панель Inspector надає інформацію про виділені об’єкти, а також про пов’язаний з ними код;
- виконання тестування проекту виконується на вкладці Game, одночасно з цим, на вкладці Console виводиться інформація про помилки.

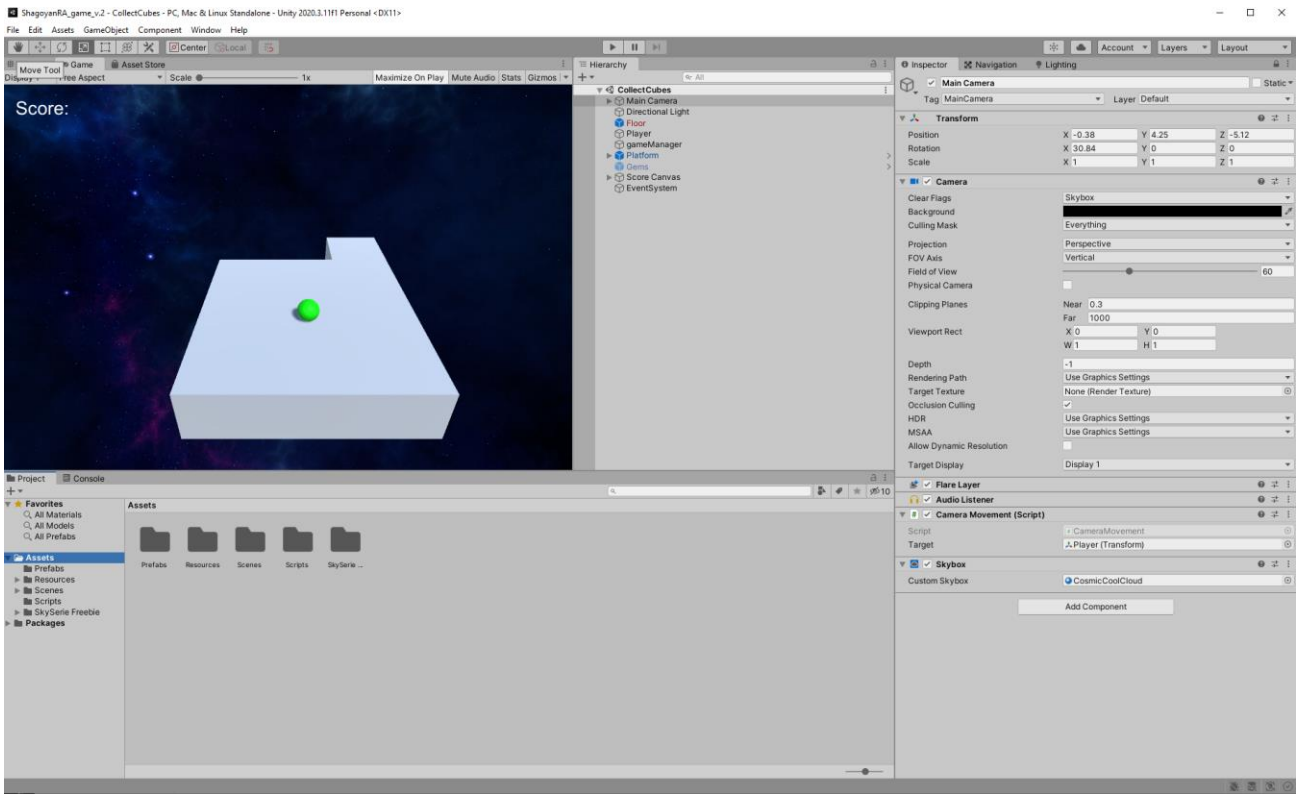

Рис. 2.3. Інтерфейс середовища Unity

В центрі інтерфейсу розташована вкладка Scene (рисунок 2.4). Саме тут можна побачити, як виглядає світ ігри, і розташовувати об’єкти на сцені. Слід розуміти, що все те, що відображається на цій вкладці відрізнятиметься від того, що буде відображатися в самому процесі гри. Ігрове представлення (процес гри) відображається на вкладці Game, що зазвичай розташована поруч з вкладкою Scene. Після запуску гри на вкладці Game починає відображатися ігрове представлення.

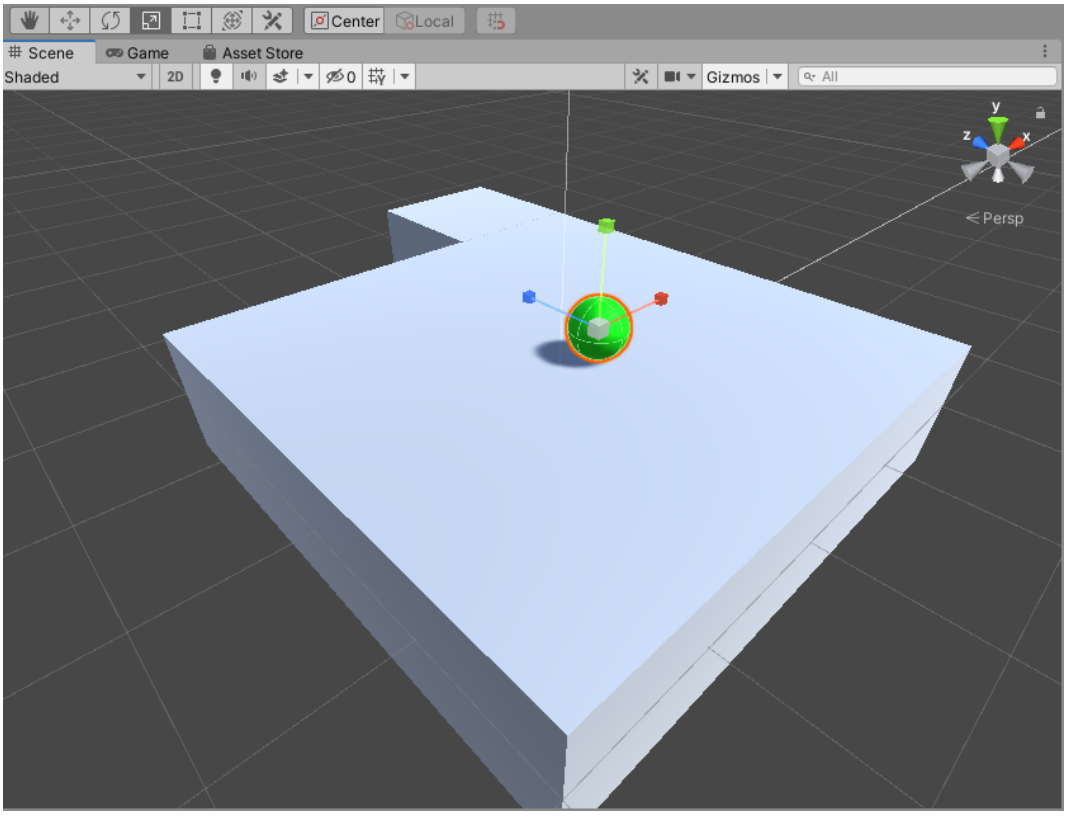

Рис. 2.4. Частина редактора, яка демонструє тільки панель інструментів і вкладку Scene

Вкладка Hierarchy (рисунок 2.5) містить список всіх присутніх на сцені об’єктів у вигляді дерева, гілки якого вкладені одна в одну відповідно до ієрархічних зв’язків між об’єктами. Вона надає можливість виділення об’єктів

по іменах, позбавляючи від необхідності пошуку об’єкта на сцені. Ієрархічні зв’язки поєднують об’єкти один з одним.

На панелі Inspector (рисунок 2.6) відображаються дані виділеного в поточний момент об’єкта. При виділенні різних об’єктів, розробник одразу ж побачить, як зміниться вигляд панелі Inspector. Інформація, що відображається, здебільшого являє собою список компонентів, причому програмісту надається можливість додавати до об’єктів нові компоненти або видаляти існуючі.

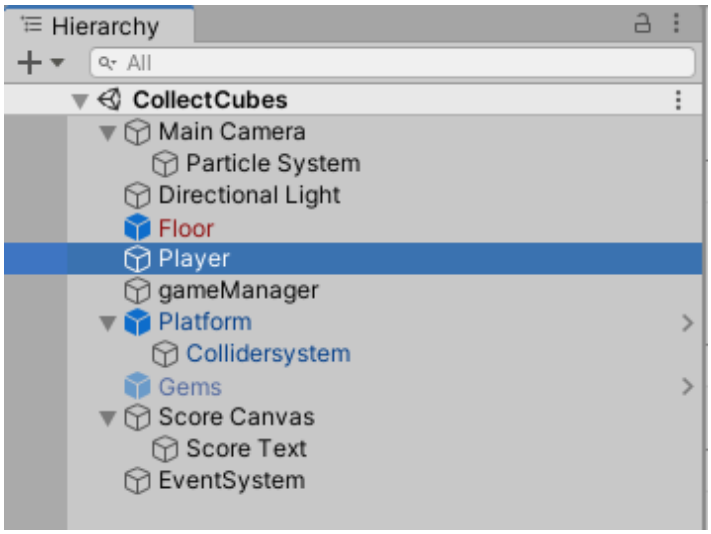

Рис. 2.5. Частина редактора, яка демонструє тільки вкладку Hierarchy

Всі ігрові об’єкти містять принаймні один компонент – Transform, тому на панелі Inspector завжди відображатимуться хоча б відомості про місцезнаходження і орієнтацію виділеного об’єкта. У багатьох об’єктів є цілі списки компонентів, в тому числі пов’язані з цими об’єктами сценарії.

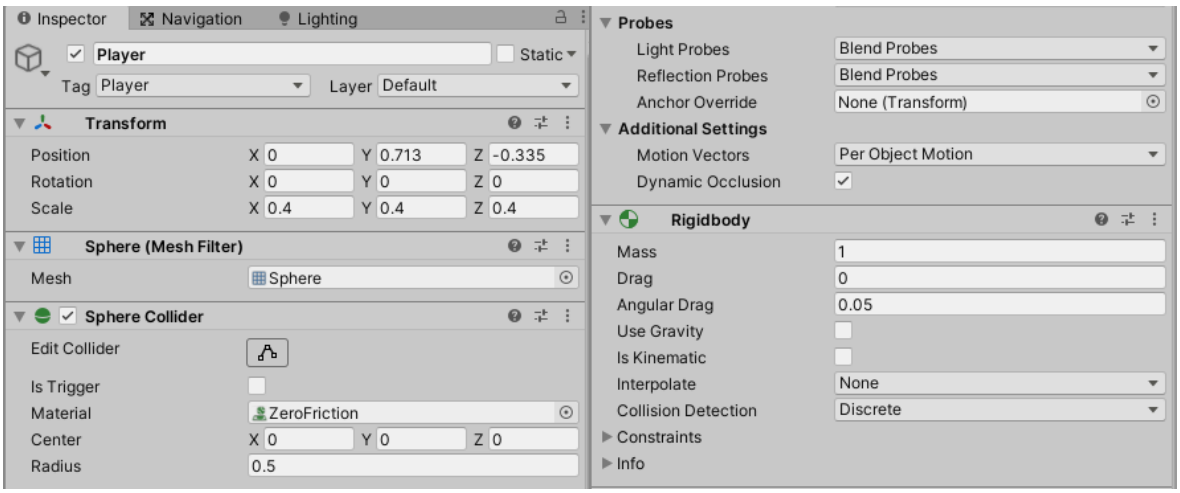

Рис. 2.6. Частина редактора, яка демонструє тільки вкладку Inspector

У нижній частині екрана знаходяться показані на рисунок 2.7 та 2.8 вкладки Project і Console. На вкладці Project відображаються всі ресурси (графічні фрагменти, файли скриптів, префаби та ін.) проекту. У лівій частині цієї вкладки знаходиться список папок проекту; при виділенні папки, файли, які знаходяться в ній, відображається праворуч. На вкладці Project відображаються навіть ті файли, що не включені до якої-небудь сцени. На цій вкладці також з’являються створенні сцени.

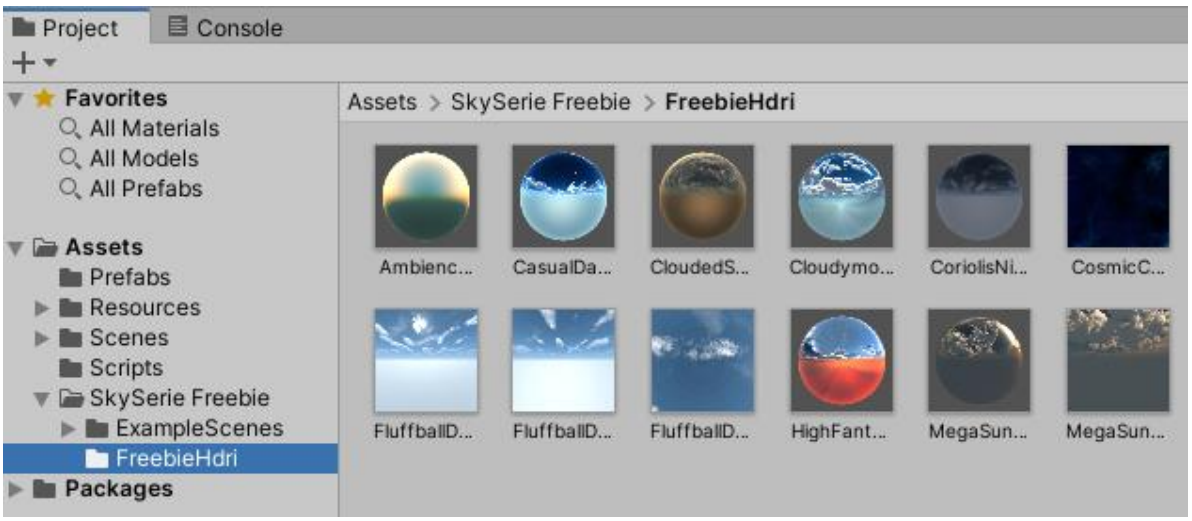

Рис. 2.7. Частина редактора, яка демонструє тільки вкладку Project та Console

Вкладка Console – панель, на яку виводяться повідомлення, пов’язані з кодом. Іноді це навмисно вставлені розробником в програму повідомлення відладчика, іноді це повідомлення Unity, що з’являються при виявленні помилок в написаному програмістом сценарії.

Для створення будь-якої гри в середовищі Unity, програміст повинен, як мінімум, володіти однією з доступних (на Unity) мов програмування: C# або JavaScript. Мова програмування C# має ряд переваг перед мовою JavaScript і значно менше недоліків. Однією з переваг є той факт, що мова C# строго типізована, чого не можна сказати про JavaScript. Серед програмістів в наш час існують різні точки зору з приводу того, чи є динамічна перевірка типів оптимальним підходом, наприклад до веб-розробки, але при написанні програм для певних платформ (таких, як Windows) вона часто вигідна, а часом необхідна навіть статична типізація. У Unity навіть додана директива #pragma, що примусово забезпечує статичну перевірку типів в мові JavaScript. Хоча з технічної сторони таке цілком допустимо, але при цьому порушується один з основних принципів функціонування JavaScript. Тому краще обрати мову зі строгою типізацією – C#.

Поведінка ігрових об’єктів контролюється за допомогою компонентів (Components), які приєднуються до них. Ядро Unity дозволяє програмісту створювати власні компоненти, використовуючи сценарії (скрипти). Вони дозволяють активувати ігрові події, змінювати параметри компонентів, і відповідати на введення користувача будь-яким способом.

Програмування здійснюється не всередині Unity, код існує у вигляді окремих файлів, місце розташування яких програміст повідомляє Unity. Файли сценаріїв можуть створюватися в додатку Unity, але в будь-якому випадку, розробнику буде необхідно використовувати текстовий редактор або IDE, де буде писатися код для цих спочатку порожніх файлів.

У комплекті з Unity поставляється додаток Microsoft Visual Studio 2019 - інтегроване середовище розробки (IDE) з підтримкою мови C# (рисунок 2.8). Програма Microsoft Visual Studio 2019 об'єднує файли в групи, що називаються рішеннями (solution). Інструмент Unity генерує зміст всіх файлів сценаріїв в рішення автоматично.

Створення сценаріїв здійснюється безпосередньо всередині Unity. Програміст може створити скрипт використовуючи меню Create в лівому верхньому кутку панелі Project або вибравши Assets > Create > C# Script (або JavaScript) в головному меню.

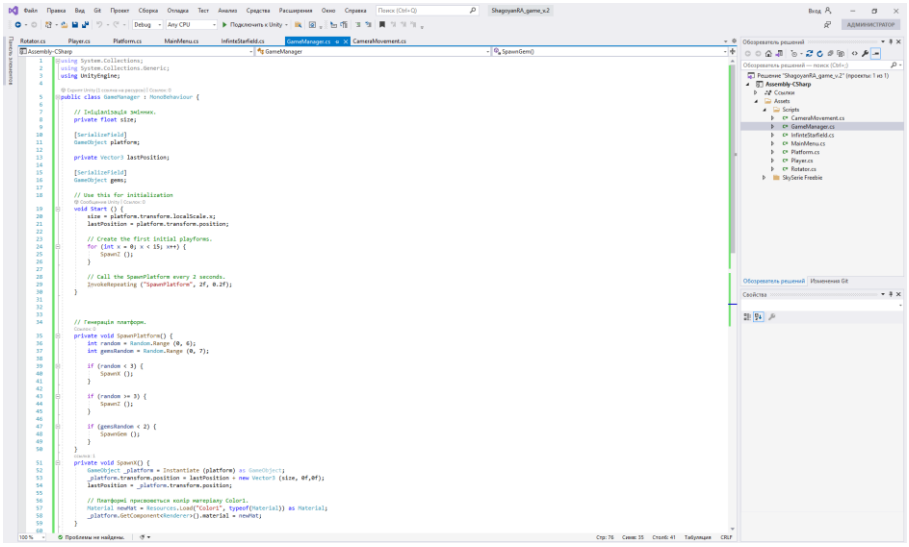

Рис. 2.8. Інтерфейс Microsoft Visual Studio 2019

Новий скрипт буде створено в папці, яку було обрано в панелі Project. Ім'я нового скрипта буде виділено, пропонуючи ввести нове ім'я (рисунок 2.9).

Рис. 2.9. Створення нового сценарію NewBehaviourScript.cs

Після подвійного клацання по скрипту в Unity, він буде відкритий в редакторі скриптів. За замовчуванням Unity буде використовувати Visual Studio 2019, але розробнику надається можливість обрати будь-який редактор з панелі External Tools в налаштуваннях Unity.

Скрипт взаємодіє з внутрішніми механізмами Unity за рахунок створення класу, який був успадкований від вбудованого класу, що називається MonoBehaviour. Слід вважати клас, як план компоненту нового типу, який може бути прикріплений до ігрового об’єкту. Кожен раз, коли програміст приєднує скрипт до ігрового об’єкту, створюється новий екземпляр об’єкту, визначений цим планом. Ім’я класу береться з імені, яке було вказано при створенні файлу. Ім’я класу і ім’я файлу повинні бути однаковими, для того, щоб скрипт міг бути приєднаний до ігрового об’єкту.

Окремо слід розглянути дві функції, визначені всередині класу. Функція Update – це місце для розміщення коду, який буде обробляти оновлення кадру для ігрового об'єкта. Це може бути рух, спрацьовування дій і відповідна реакція на введення користувача, в основному все, що повинно бути оброблено з плином часу у ігровому процесі. Щоб дозволити функції Update виконувати свою роботу, часто буває корисно форматувати змінні, считати властивості і здійснити зв'язок з іншими ігровими об'єктами до того, як будуть здійснені будь-які дії. Функція Start буде викликана Unity до початку ігрового процесу (тобто до першого виклику функції Update), і це ідеальне місце для виконання ініціалізації змінних.

Скрипт визначає тільки план компонента і, таким чином, ніякий його код не буде активований до тих пір, поки екземпляр скрипта НЕ буде приєднаний до ігрового об'єкту. Програміст може прикріпити скрипт перетягуванням ассета скрипта на ігровий об'єкт в панелі Hierarchy або через вікно Inspector обраного ігрового об'єкта. Є також підменю Scripts в меню Component, яке містить всі скрипти, доступні в проекті, створені вами. Екземпляр скрипта виглядає так само, як і інші компоненти у вікні Inspector (рисунок 2.10).

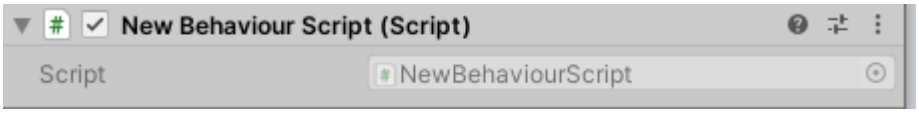

Рис. 2.10. Скрипт NewBehaviourScript.cs у представленні компонента

Після приєднання скрипт почне працювати, коли буде натиснуто Play і запущено гру.

2.5. Опис структури комп'ютерної системи та алгоритмів її функціонування

Розроблений проект має кореневу структуру, зображену на рисунку 2.11. Лістинг коду міститься у Додатку А.

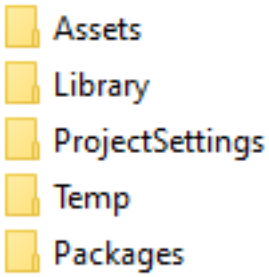

Рис. 2.11. Структура проекту розробленого ігрового додатку

Головні папки проекту:

- Assets – в цій папці містяться усі файли скриптів, матеріалів та 3-D об'єктів;
- Library – в папці містяться мета дані, кеш та інші файли проекту;
- ProjectSettings – в цій папці містяться конфігураційні файли;
- Temp – папка зберігає тимчасові файли;
- Packages – спеціальна папка із файлами ядра Unity.

При проектуванні ігрового додатку були розроблені скрипти, за допомогою яких здійснюється керування компонентами (Components). Вони активують ігрові події, змінюють та встановлюють певні значення параметрів компонентів, та можуть реагувати на дії користувача будь-яким необхідним чином.

Ігровий додаток забезпечено головним меню користувача, за допомогою якого він має можливість зручно обрати необхідний пункт для функціонування програми певним чином (тобто запуск геймплею або вихід). Результат проектування показаний на вкладці ігрової сцени, на якій розташовані компоненти головного меню (рисунок 2.12).

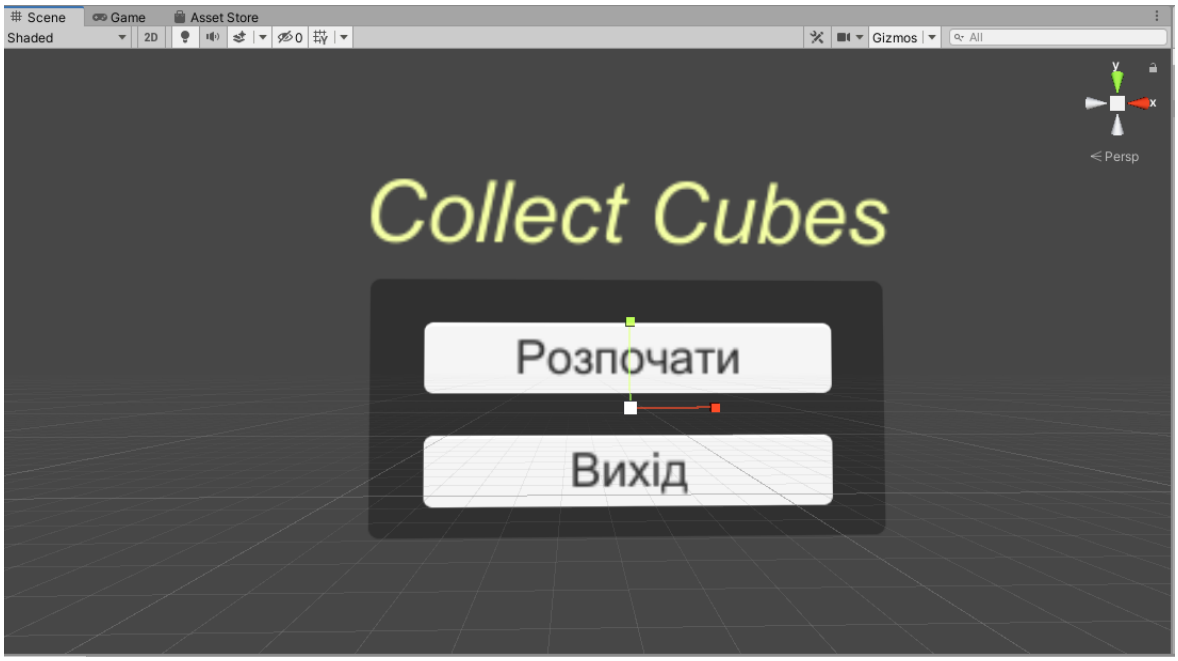

Рис. 2.12. Ігрова сцена із спроектованим прототипом головного меню

Ігровий процес проектується на іншій сцені, на якій було розташовано певні 3-D об’єкти у тривимірному просторі.

Спроектована сцена має вигляд типового для жанру Arcade рівня, на якому розташований головний об’єкт, яким керує гравець та встановлено ігрову платформу, яка є точкою спавну ігрового об’єкта гравця.

Крім того, на сцені розташовано камеру, завдяки який виконується рендер зображення ігрового простору та виводиться на монітор ПК.

Також на сцені встановлено спеціальний компонент SkyBox, який імітує особливий навколишній простір (рисунок 2.13).

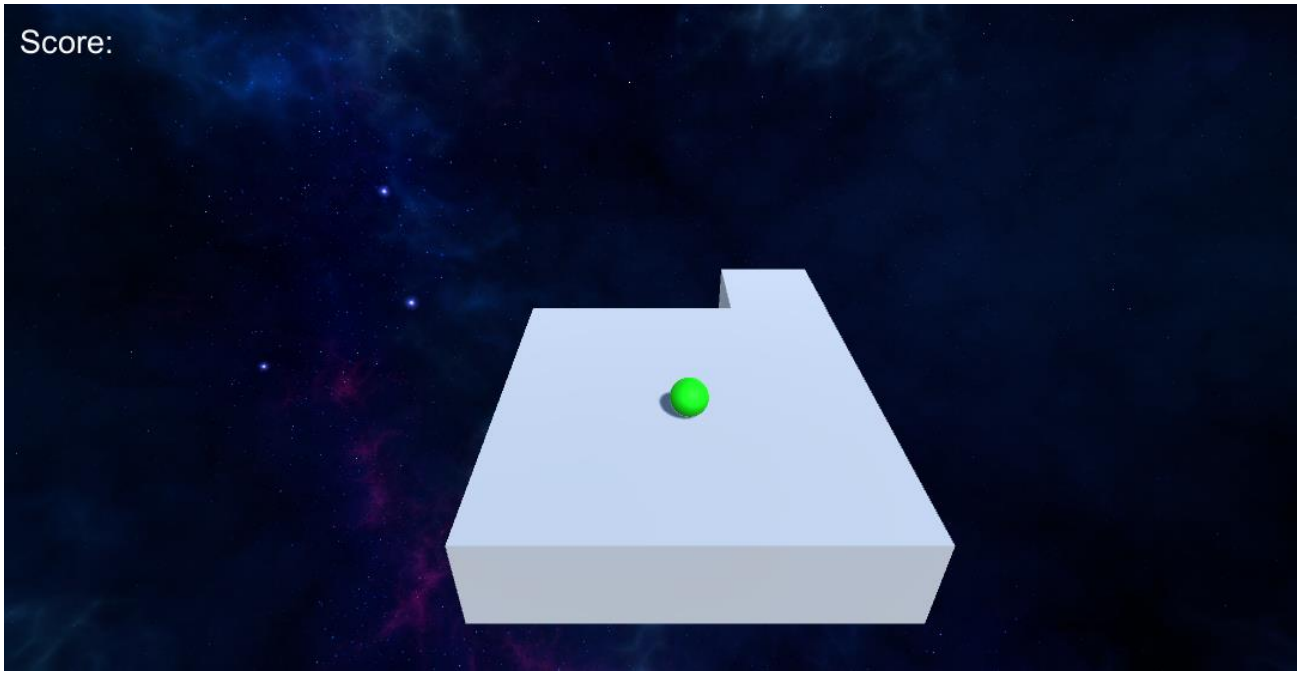

Рис. 2.13. Ігрова сцена із внутрішньоігровим елементом SkyBox

При розробці ігрового процесу, було створено наступні скрипти:

- CameraMovement.cs – контролює камеру;
- InfiniteStarfield.cs – контролює Particles-систему;
- Platform.cs – в цьому скрипті було реалізовано механіку взаємодії платформи с персонажем гравця;
- Player.cs – цей скрипт забезпечує гравця механіками ігрового процесу;
- Rotator.cs – цей скрипт реалізує механіку анімації кристалів;
- GameManager.cs – скрипт контролює генерацію платформ, що забезпечує умовно нескінченний ігровий процес;
- MainMenu.cs – у скрипті реалізовано функціонал ігрового меню.

Сценарії в Unity представляють собою компоненти (але лише ті, що наслідуються від класу MonoBehavior, базового класу компонент-сценаріїв). Цей клас визначає спосіб приєднання компонентів до ігрових об'єктів. Спадкування від даного класу надає пару автоматично додаваємих методів. Це метод Start(), викликається при активації об'єкта (а вона, як правило, наступає після завантаження вмісту рівня об'єкта), і метод Update(), викликається в кожному кадрі. Відповідно, код запускається, коли він розміщується в цих попередньо встановлених методах.

Для керування ігровою камерою, необхідно контролювати її параметри на вкладці Inspector за допомогою скрипта CameraMovement.cs. У методі void Start() призначаємо змінній offset значеня – різницю між значеннями компонентів transform, в яких знаходяться координати розташування ігрових об’єктів. Після цього у методі void Update виконуємо перевірку на можливість гравця пересуватися. Ця перевірка виконується для того, щоб коли гравець впав з платформи, камера переставала слідкувати за його персонажем. Якщо цього скрипта не було би створено, то при пересуванні персонажа гравця, ігрова камера завжди залишалась на одному й тому ж самому місці і ігровий процес закінчувався після декількох секунд з його початку.

Функціонування Particles-системи реалізовано у скрипті InfiniteStarfield.cs. Ця система відповідає за генерацію візуальних часток, які з’являються у всьому ігровому просторі. Частки забезпечують більш краще відчуття тривимірності завдяки створеній перспективі. У розробленому скрипті прописані правила генерації візуальних часток, а саме: встановлено максимальну кількість часток, їх розмір, координати генерації, визначено відстань однієї частки від іншої, а також густину розподілу у внутрішньо ігровому просторі.

Після налаштування камери, необхідно надати гравцю можливість рухатись до певного місця та взаємодіяти з ігровими об’єктами. Це здійснюється за допомогою скрипту Player.cs. В цьому скрипті знаходяться

методи і змінні, завдяки яким, наприклад, здійснюється пересування гравця у ігровому просторі. Розроблені методи реалізують внутрішньо ігрову механіку взаємодії персонажа гравця з ігровим простором та об’єктами в ньому. Наприклад, проводиться перевірка на стан персонажа гравця для того, щоб було зрозуміло, чи знаходиться він на платформі? Якщо ні і він впав, то ігровий процес закінчується і гравець повертається до головного меню. Також у цьому скрипті є такий метод OnTriggerEnter, який відповідає за збирання ігрового елементу – кристалу. Якщо при пересуванні персонажа гравця у ігровому просторі, його компонент Collider стикається із компонентом Collider кристалу, то кристал збирається, оновлюється ігровий рахунок (рекорд) і кристал знищується, викликаючи метод, який створює візуальний ефект знищення (рисунок 2.14).

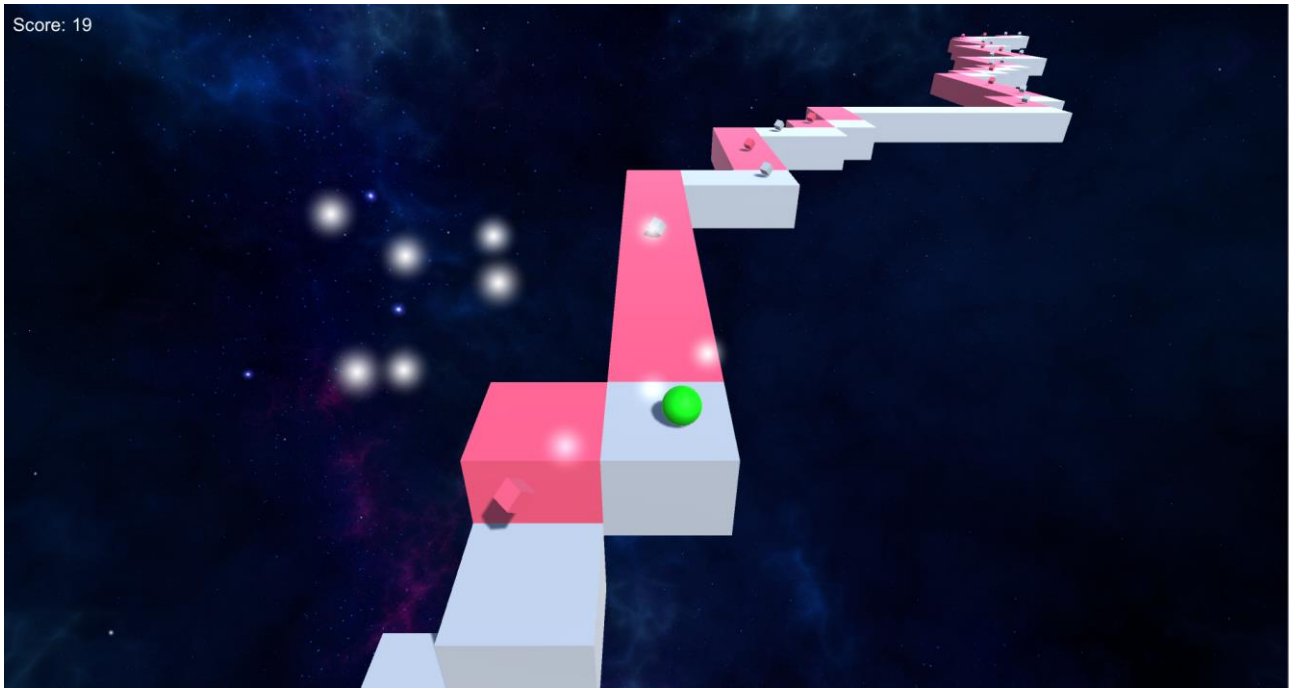

Рис. 2.14. Функціонал скрипту Player.cs полягає у взаємодії персонажа гравця з внутрішньоігровими об’єктами

Взаємодія ігрового персонажу з платформами здійснюється за допомогою скрипту Platform.cs. Якщо компонент Collider персонажа гравця викликає подію OnTriggerEnter, то при цій умові викликається метод FallDown(), який створює гравітаційний ефект, і платформа падає. А після невеликої затримки вона знищується завдяки методу Destroy(). Результат роботи скрипта відображено на рисунку 2.15.

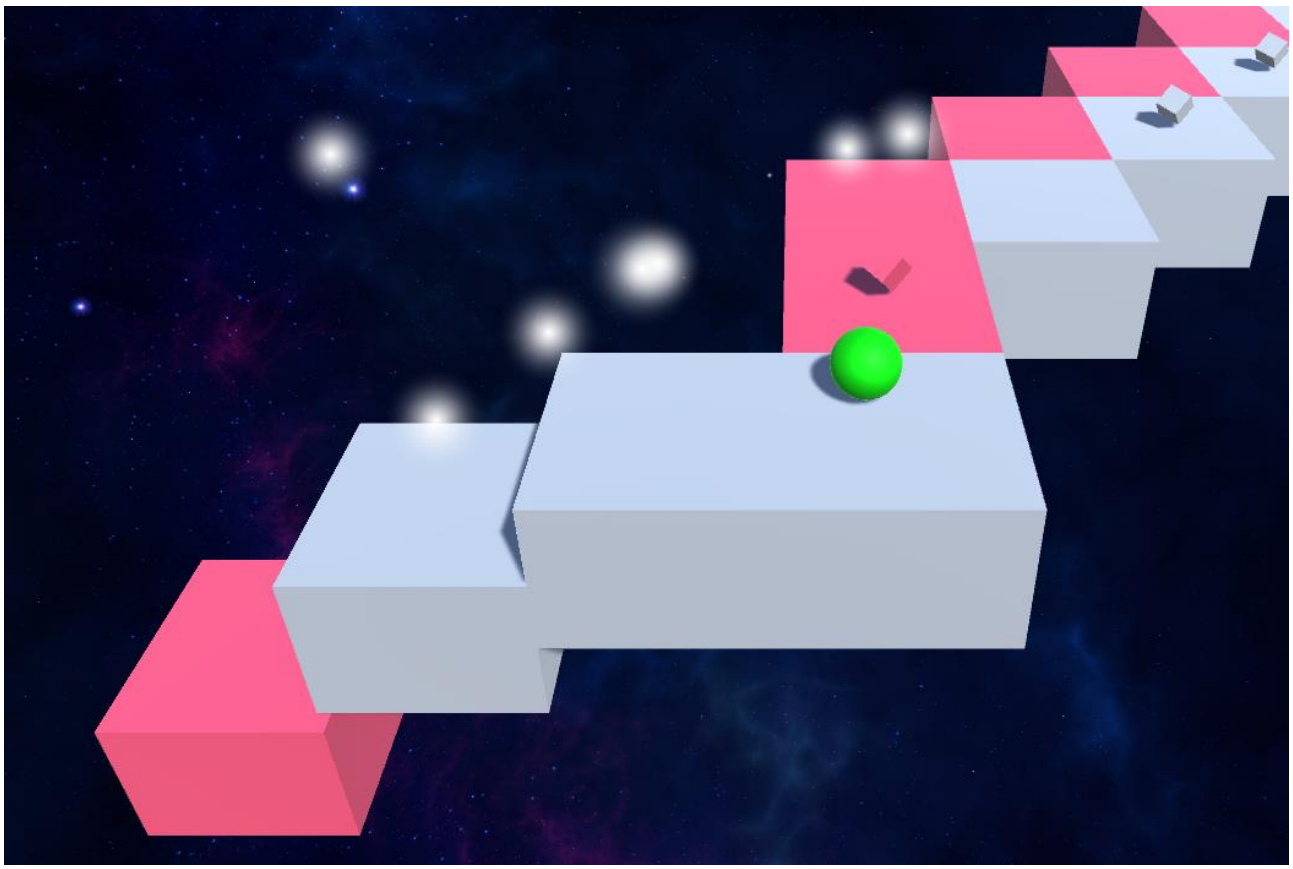

Рис. 2.15. Платформи падають у результаті дії скрипта Platform.cs

Оскільки однією з головних особливостей жанру Arcade є “збирання”, то при розробці ігрового додатку було створено функціонал, який забезпечує ігровий процес можливістю збирання ігрових об’єктів – кристалів. Щоб зацікавити гравця та дати йому інтуїтивно зрозуміти, що потрібно збирати, було розроблено скрипт Rotator.cs, який забезпечує кристали анімацією обертання.

Анімація створюється завдяки методу Update(), який викликається кожну зміну кадру. В середині цього методу було обчислено функцію обертання в компоненті Transform для кристалу. В цьому компоненті знаходяться координати розташування та розміру ігрових об’єктів. Результат роботи цього скрипта відображено на рисунках 2.16 та 2.17

Рис. 2.16. Обертання кристалів реалізується у скрипті Rotator.cs

Рис. 2.17. Обертання кристалів, відображено з іншого ракурсу.

Систему генерації платформ і кристалів у ігровому додатку було реалізовано в скрипті GameManager.cs. Завдяки розробленим методам виконується Spawn платформ через встановлений проміжок часу. Це забезпечує

гравця потенційно нескінченним ігровим процесом. Платформи починають з’являтися після старту геймплею. За генерацію платформ по осі X відповідає метод SpawnX(), за генерацію платформ по осі Z відповідає метод SpawnZ. Сутність генерації полягає у тому, що до компоненту Transform.position додаються координати майбутньої платформи, беручи до уваги її розмір. Після обчислення позиції майбутньої платформи виконується Instantiate, що створює тривимірний об’єкт за заданими координатами. Після генерацій випадкової кількості платформ на одній осі, ось змінюється. Це забезпечує створення ігрового рівня нелінійним та цікавим. Результат роботи цього скрипта відображено на рисунку 2.18.

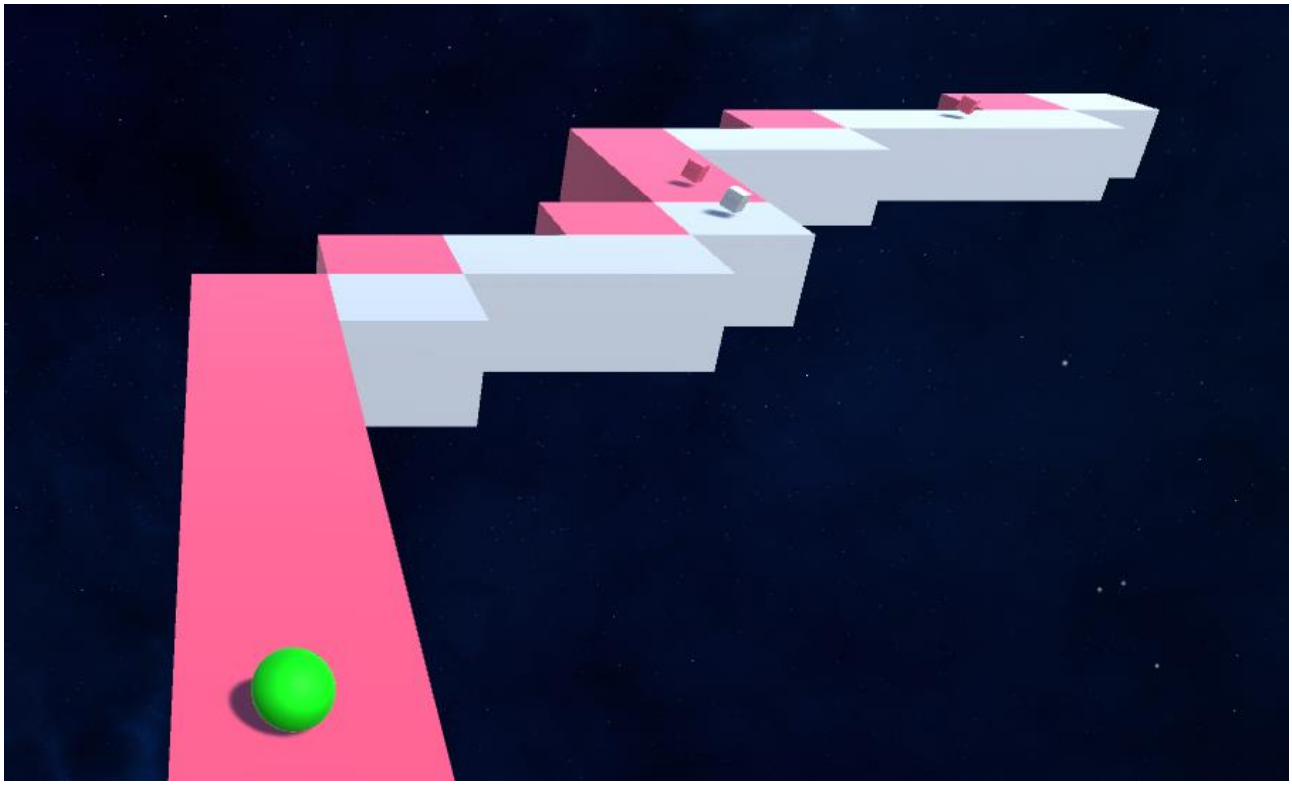

Рис. 2.18. Генерація платформ у результаті роботи GameManager.cs.

Скрипт MainMenu.cs відповідає за можливість гравця запустити ігровий процес, або вийти із ігрового додатку. До головного меню гравець потрапляє

при запуску ігрового додатку з файлу, у якого розширення “.exe”, а також після того, як персонаж гравця впаде з платформи. Сутність роботи скрипта MainMenu.cs полягає у взаємодії гравця з розробленим графічним інтерфейсом. При натисканні на кнопку-елемент GUI, яка запускає ігровий процес, викликається метод, що змінює ігровий рівень з Головного меню на саму гру. Це здійснюється завдяки зміні ігрових сцен. Інакше кажучи, ігрова сцена з головного меню міняється на ігрову сцену з самим ігровим процесом. Результат роботи скрипта MainMenu.cs відображено на рисунку 2.19.

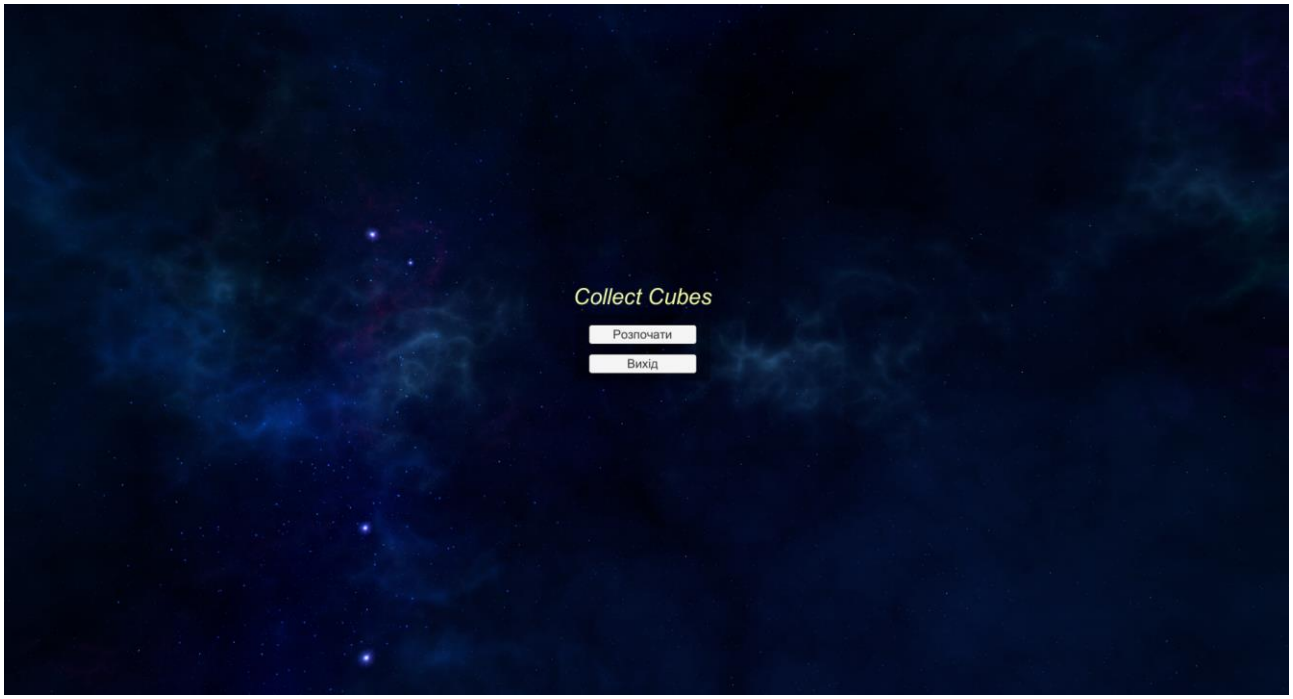

Рис. 2.19. Головне меню ігрового додатку із розробленим графічним інтерфейсом користувача.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Ігровий двіжок Unity Engine працює з моделлю подій (Event).

Events – це події, які викликаються з коду гри, якщо виконується певна умова. Вони дають змогу здійснити ряд дій у відповідь на певні події, що відбуваються в грі.

Наприклад, завдяки обробці події натискання лівої кнопки миші, в розробленому ігровому додатку реалізується керування персонажем гравця. В даній програмі вхідними даними є натискання лівої кнопки миші.

Користувач ліву кнопку миші генерується подія з кодом клавіші — 0, після цього, обробник події перехоплює цю подію, та викликає метод ChangeDirection(), який відповідає за зміну напрямку пересування персонажа гравця.

Вихідними даними розробленого ігрового додатку можна вважати результат роботи камери. Камера є пристроєм, яка захоплюють і відображає світ гравцеві. Шляхом настройки і маніпулювання камерою, можна зробити презентацію гри дійсно унікальною. Можливо мати необмежену кількість камер в сцені, але в проекті розробки ігрового додатку є лише одна. Завдяки камері, гравець може спостерігати візуальні дані з монітору.

2.7. Опис розробленого програмного продукту

Щоб перетворити розроблений ігровий додаток на готовий для споживача продукт, необхідно створити білд ігрового застосунку. Білд – це упаковка ігрового додатку для його споживання гравцем. Упаковка виконується завдяки функціоналу движка Unity Engine. Процес упаковки виконується автоматично, необхідно лише вказати цільову платформу розгортання та натиснути кнопку Build and Run.

Результатом упаковки ігрового додатку є папка із готовою для використання гравцем грою (рисунок 2.20).

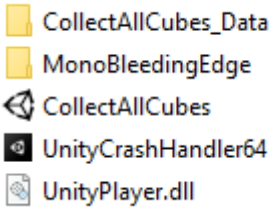

Рис. 2.20. Результат упаковки ігрового додатку для використання гравцем

Для роботи з програмою, досить запустити файл “CollectAllCubes”. Після чого запускається ігровий додаток. Якщо мінімальні технічні характеристики обладнання на якому було запущено даний додаток не будуть відповідати, користувач отримає повідомлення про це.

2.7.1. Використані технічні засоби

При розробці та тестуванні системи була використана клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

- процесор класу Intel i3;
- монітор 60 гц;
- не менше 2000Мб ОЗУ;
- 3Гб вільного місяця для ігрового двигуна;
- 229 МБ вільного місця для ігрового додатку;
- клавіатура;
- миша.

2.7.2. Використані програмні засоби

Для роботи програмного засобу необхідні такі програмні засоби:

- Unity Engine версії не менш, ніж 2020.3.11f1;
- Visual Studio версії не менш, ніж 17.

2.7.3. Виклик та завантаження програми

Для роботи з розробленим ігровим додатком, необхідно запустити файл з назвою “CollectAllCubes” з папки із ігровим додатком (рисунок 2.21).

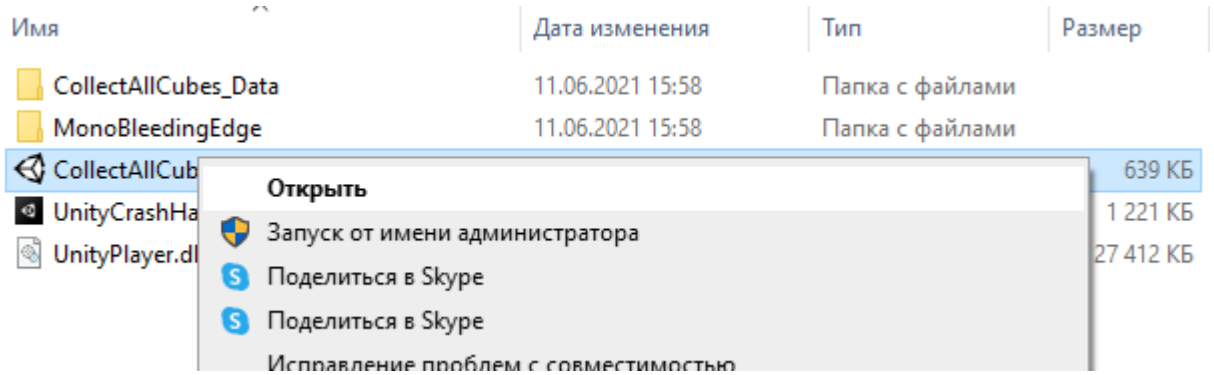

Рис. 2.21. Запуск ігрового додатку із папки

Після запуску файлу CollectAllCubes, користувач побачить екран завантаження гри (рисунок 2.22)

Рис. 2.22. Екран завантаження гри

Після вдалої загрузки ігрового додатку, якщо ПК користувача відповідає вимогам, вказаним у пункті 1.5.3., відкриється перша ігрова сцена – Головне меню (рисунок 2.19), а натиснувши кнопку-елемент GUI із надписом “Розпочати”, запуститься ігровий процес (рисунок 2.23).

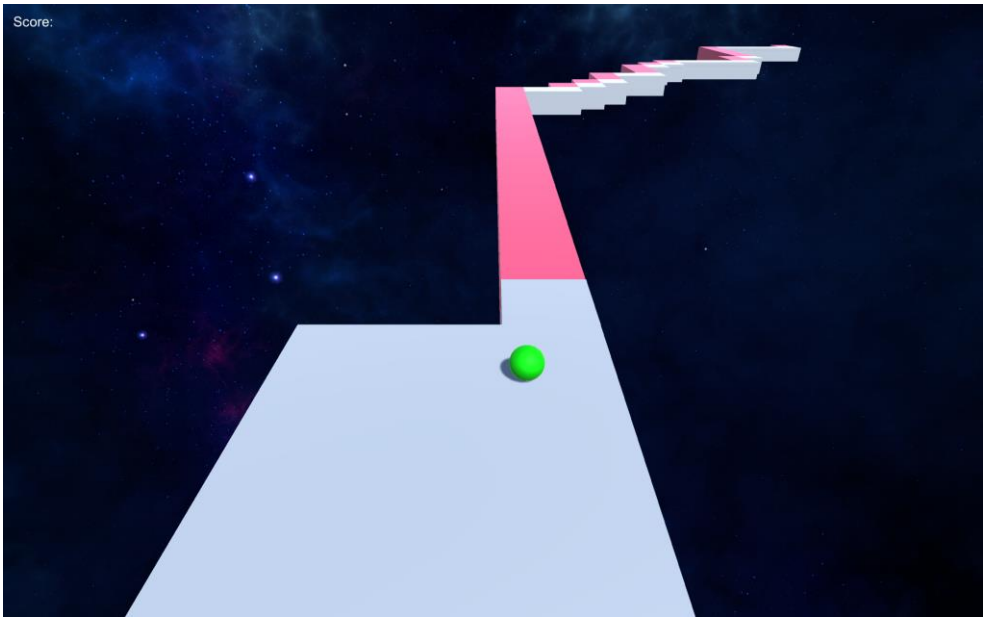

Рис. 2.23. Початок ігрового процесу

2.7.4. Опис інтерфейсу користувача

При розробці тривимірного ігрового додатку у жанрі Arcade, було спроектовано та розроблено функціональний інтерфейс користувача – головне меню та елемент відображення ігрового рахунку. Для коректної роботи за стосунку, необхідно дочекатися вдалого налаштування та запуску самого ігрового двигуна.

При запуску ігрового додатку, буде запущено головне меню із розробленим інтерфейсом користувача, завдяки натисканню на елементи якого, гравець зможе запустити ігровий процес. Головне меню (рисунок 2.24) складається із таких елементів графічного інтерфейсу користувача:

- Надпис “Collect Cubes”, цей елемент служить для відображення назви відеогри;
- Інтерактивна кнопка із надписом “Розпочати”, елемент запускає сцену із ігровим процесом;
- Інтерактивна кнопка із надписом “Вихід”, елемент завершує роботу ігрового додатку

Рис. 2.24. Частина інтерфейсу користувача “Головне меню”

Також елементом інтерфейсу користувача є надпис “Score”. Головна роль цього елементу полягає у відображенні ігрового рахунку у реальному часі (рисунок 2.25):

Рис. 2.25. Елемент інтерфейсу користувача “Score”

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

1. Початкові дані:
2. передбачуване число операторів програми – 1048;
3. коефіцієнт складності програми – 1,3;
4. коефіцієнт корекції програми в ході її розробки – 0,07;
5. годинна заробітна плата програміста– 58 грн/год;
6. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
7. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
8. вартість машино-години ЕОМ –13 грн/год.

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:

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

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 програмногму забезпеченні, яке розробляється.

Умовне число операторів (тегів):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1300);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}3 \cdot 1048 \cdot (1 + 0{,}05) = 1430{,}52$$

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

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{(1430 \cdot 1{,}2)}{(75 \cdot 1{,}2)} = 19 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\text{ , людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), людино-годин:

$$t_a = \frac{1165}{(20\cdot 1{,}2)} = 49 \text{ людино-годин.}$$

Витрати на складання програми по готовій схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\text{ , людино-годин.}$$

$$t_n = \frac{(1165\cdot 1{,}2)}{(20\cdot 1{,}2)} = 58 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\text{ , людино-годин.}$$

$$t_{отл} = \frac{1165}{(5\cdot 1{,}2)} = 194 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл} \text{ , людино-годин.}$$

$$t^{k}_{отл} = 1{,}5 \cdot 194 = 291 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

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

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

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

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = \frac{1165}{(18 \cdot 1{,}2)} = 54 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 \cdot 316 = 41 \text{ людино-годин.}$$

$$t_{д} = 54 + 41 = 95 \text{ людино-годин.}$$

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

$t = 50 + 22 + 49 + 58 + 194 + 95 = 468$ людино-годин.

3.2. Розрахунок витрат на створення додатку

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

$K_{ПО}=З_{ЗП}+З_{МВ}$, грн.

Заробітна плата виконавців визначається за формулою:

$З_{ЗП}=t \cdot C_{ПР}$, грн,

де: t - загальна трудомісткість, людино-годин;

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

З урахуванням того, що середня годинна зарплата програміста становить 58 грн / год, отримуємо:

$З_{ЗП}$ = 468 · 58 = 27,144 грн.

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

$З_{мв}=t_{отл} \cdot C_{мч}$, грн, (3.3)

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$З_{мв}$= 224 · 13 = 2912 грн.

Звідси витрати на створення програмного продукту:

$K_{ПО}$ = 32,760 + 2,912 = 35,672 грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p},$$ міс.

де B_k- число виконавців (дорівнює 1);

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

Звідси витрати на створення програмного продукту:

$$T = \frac{468}{(1 \cdot 176)} \approx 3 \text{ міс.}$$

Висновок: ігровий додаток розроблений з метою забезпечення гравця способом скоротати час та скрасити дозвілля. Вартість даного додатку становить 35,672 грн. та не потребує додаткових витрат при розробці проекту. Очікуваний час розробки - 3 місяці. Цей термін пов'язаний зі значною кількістю операторів і включає в себе час для дослідження та розробки алгоритму розв'язання задачі, розробку дизайну, створення ігрового додатку та підготовку документації.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка тривимірного ігрового додатку у жанрі Arcade, що забезпечує користувача ігровим процесом, та дає можливість скоротити час за грою, в умовах карантину. При розробці були взяті до уваги сучасні методи та засоби для розробки тривимірних ігор.

Розроблений проект призначений для розважальних цілей але допомагає користувачу розвивати мозкові здібності – покращити реакцію та сприйнятливість рухаючихся ігрових об’єктів.

Програма працює під керування Windows OS, яка широко використовується цільовою аудиторією продукту. Розробка велась на високорівневій мові програмування C#, яка надає змогу досить точно оперувати ресурсами й дозволила досягти достатніх результатів в швидкості роботи програми. Допоміжними методами було використано ігровий двигун Unity Engine. Головною задачею було те, щоб користувач проводив час із задоволенням.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (468 чол-год), підраховані витрати на створення програмного забезпечення (35,672 грн.) і гаданий період розробки (3 міс.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alan Thorn — Mastering Unity Scripting, 2015. (Last review 07 January 2021)
2. Alan Thorn — Unity Animation Essentials, 2015. (Last review 07 January 2021)
3. Alan Thorn — How to Cheat in Unity 5: Tips and Tricks for Game Development, 2015. (Last review 07 January 2021)
4. Rouse, Richard. Game Design: Theory & Practice : [англ.]. — 2. — Los Rios Boulevard, Plano, Texas, USA : Wordware Publishing, 2004. — 698 p. — ISBN 1-55622-912-7. (Last review 24 April 2021)
5. Moore, Michael E. Basics of Game Design : [англ.]. — 2. — New York, USA : CRC Press, 2011. — 376 p. — ISBN 13: 978-1-4398-6776-1. (Last review 24 April 2021)
6. Rogers, Scott. Level Up! The Guide to Great Video Game Design : [англ.]. — 2. — USA : Wiley, 2010. — 492 p. — ISBN 978-0-470-68867-0. (Last review 24 April 2021)
7. Schwab, Brian. AI Game Engine Programming : [англ.]. — 2. — Canada : Course Technology, 2009. — 710 p. — ISBN 978-1-5845-0572-3. (Last review 24 April 2021)
8. Kent L., Steven. The Ultimate History Of Video Games : [англ.]. — 1. — New York : Three Rivers Press, 2001. — 608 p. — ISBN 0-7615-3643-4. (Last review 14 May 2021)
9. Joseph Hocking — Unity in Action. Multiplatform game development in C# with Unity 5, 2015. (Last review 14 May 2021)
10. Chris Dickinson — Unity 5 Game Optimization, 2015. (Last visit 14 May 2021)
11. Kenny Lammers — Unity Shaders and Effects Cookbook, 2013. (Last review 14 May 2021)
12. Linowes J.: Unity Projects/ Linowes J.-2015.-286 p. (Last review 14 May 2021)

13. Barrat, James. The latest invention of mankind / James Barrath. - M., 2015. 299 p. (Last review 14 May 2021)
14. Платов, В. Я. Розробка та організація ігрових додатків. Підручник / В.Я. Платов. - М.:, 2015. - 192 с. (Останній перегляд 2 Червня 2021)
15. Любанова, Т.П. Бізнес-план: досвід, проблеми. Зміст бізнес-плану, приклад розробки / Т.П. Любанова, Л.В. Мясоєдова, Т.А. Грамотенко, и др.. - М.: Приор, 2012. - 204 с. 59 (Останній перегляд 2 Червня 2021)
16. Хорхе, Паласиос Unity 5.x. Програмування в іграх. Керівництво / Паласиос Хорхе. - М.: ДМК Пресс, 2017. - 427 с. (Останній перегляд 2 Червня 2021)
17. Алгазинов, Э. К. Аналіз та компютерне моделювання інформаційних процесів та систем / Э.К. Алгазинов, А.А. Сирота. - М.: Диалог-Мифи, 2009. - 416 с. (Останній перегляд 2 Червня 2021)
18. Kim, J. Modeling and Optimization of a Tree Based on Virtual Reality for Immersive Virtual Landscape Generation. Symmetry 2016, 8, 93. (Last review 02 June 2021)
19. Slater, M.; Usoh, M. Simulating peripheral vision in immersive virtual environments. Comput. Graph. 1993, 17, 643–653. (Last visit 02 review 2021)
20. Jeong, K.; Lee, J.; Kim, J. A Study on New Virtual Reality System in Maze Terrain. Int. J. Hum. Comput. Interact. 2018, 34, 129–145. (Last review 02 June 2021)
21. Goldstone W. Unity Game Development Essentials. / W Goldstone. Birmingham: Packt Publishing Ltd., — 2009. — 316 c. (Last review 02 June 2021)

ДОДАТОК А

КОД ПРОГРАМИ

Platform.cs

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

public class Platform : MonoBehaviour {

	void OnTriggerEnter(Collider collider) {

	if (collider.gameObject.tag == "Player") {


	Invoke ("FallDown", 0.6f);

	}
	}

	private void FallDown() {

	this.GetComponentInParent<Rigidbody>().isKinematic = false;

	Destroy (this.transform.parent.gameObject, 2f);
	}
}
```

Rotator.cs

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

```
public class Rotator : MonoBehaviour {

	// Update is called once per frame
	void Update () {

	transform.Rotate (new Vector3(15,20,45) * Time.deltaTime);
	}
}
```

MainMenu.cs

```
using UnityEngine;
using System.Collections;

public class MainMenu : MonoBehaviour {

	public void StartGame() {
	Application.LoadLevel(1);
	}
	public void Quit() {
	Application.Quit();
	}
}
```

CameraMovement.cs

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

public class CameraMovement : MonoBehaviour {

	// Initialize variables.
```

```
        [SerializeField]
        Transform target;

        Vector3 offset;

        // Use this for initialization
        void Start () {

        offset = target.transform.position - this.transform.position;
        }

        // Update is called once per frame
        void Update () {
        // Якщо гравець не впав з платформи, камера слідкує за ігровим об'єктом гравця
        if (target.gameObject.GetComponent<Player> ().canMove) {

        Vector3 RequiredPosition = target.transform.position - offset;


        this.transform.position = Vector3.Lerp (this.transform.position, RequiredPosition, 1.5f);

        }
        }
}
```

InfiniteStarfield.cs

```
using UnityEngine;
using System.Collections;

public class InfinteStarfield : MonoBehaviour {


        private Transform tx;
        private ParticleSystem.Particle[] points;
```

```
public int starsMax = 100;
public float starSize = 0.1f;
public float starDistance = 5f;
public float starClipDistance = 0f;
private float starDistanceSqr;
private float starClipDistanceSqr;


// Use this for initialization
void Start () {
tx = transform;
starDistanceSqr = starDistance * starDistance;
starClipDistanceSqr = starClipDistance * starClipDistance;
}
private void CreateStars() {
points = new ParticleSystem.Particle[starsMax];
for (int i = 0; i < starsMax; i++) {
points[i].position = Random.insideUnitSphere * starDistance + tx.position;
points[i].color = new Color(1,1,1, 1);
points[i].size = starSize;
}
}

// Update is called once per frame
void Update () {
float distance = 5;

//transform.position = transform.position + Camera.main.transform.forward * distance * Time.deltaTime;
if ( points == null ) CreateStars();
for (int i = 0; i < starsMax; i++) {
if ((points[i].position - tx.position).sqrMagnitude > starDistanceSqr) {
points[i].position = Random.insideUnitSphere.normalized * starDistance + tx.position;
```

```
            }
            if ((points[i].position - tx.position).sqrMagnitude <= starClipDistanceSqr) {
            float percent = (points[i].position - tx.position).sqrMagnitude / starClipDistanceSqr;
            points[i].color = new Color(1,1,1, percent);
            points[i].size = percent * starSize;
            }
            }

            GetComponent<ParticleSystem>().SetParticles(points, points.Length);

            }
}
```

Player.cs

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

public class Player : MonoBehaviour {

            // Ініціалізація змінних.
            private Rigidbody rb;
            private bool isMovingRight = false;
            private bool hasPlayerStarted = false;

            [SerializeField]
            float speed = 4f;

            [HideInInspector]
            public bool canMove = true;

            [SerializeField]
```

```
GameObject particle;

[SerializeField]
GameObject gems;

[SerializeField]
private Text scoreText;
private int score = 0;


// Use this for initialization
void Start () {

rb = this.GetComponent<Rigidbody> ();
}

// Update is called once per frame
void Update () {

if (Input.GetMouseButtonDown (0) && canMove == true) {



// Якщо гра ще не почалась.


if (hasPlayerStarted == false) {



hasPlayerStarted = true;
```

```
StartCoroutine (ShowGems (2.5f));



}

ChangeBoolean ();
ChangeDirection ();
}
if (Physics.Raycast (this.transform.position, Vector3.down * 2) == false) {
FallDown ();
}
}
// Затримка генерації кристалів.
IEnumerator ShowGems (float count) {
yield return new WaitForSeconds (count);

// Перевірка чи впав гравець до генерації кристалів.

if (canMove == true) {
gems.SetActive (true);

}
}

// Контролюємо напрямок пересування гравця.
private void ChangeBoolean() {

isMovingRight = !isMovingRight;
}
private void ChangeDirection() {

if (isMovingRight == true) {
rb.velocity = new Vector3 (speed, 0f, 0f);
}
```

```
else {
rb.velocity = new Vector3 (0f,0f,speed);
}
}
// Гравець падає з платформи.
private void FallDown() {
canMove = false;
rb.velocity = new Vector3 (0f,-4f,0f);
// Повертаємося до головного меню.
StartCoroutine (ReturnToMainMenu (2.8f));
}
// До головного меню
IEnumerator ReturnToMainMenu (float count) {
yield return new WaitForSeconds (count);
Application.LoadLevel(0);
}


// Гравець збирає кристал.
void OnTriggerEnter(Collider other) {

if (other.gameObject.tag == "Gem") {




// Знищуємо кристал.
Destroy (other.gameObject);

// Викликаємо систему Particles.
GameObject _particle = Instantiate (particle) as GameObject;
_particle.transform.position = this.transform.position;
Destroy (_particle, 1f);
```

```
        // Оновлюємо рекорд.
        score++;
        scoreText.text = "Score: " + score.ToString();
        }
        }
}
```

UnityCompiler.exe1.txt

Base path: 'D:/Program Files/Unity/2020.3.11f1/Editor/Data', plugins path 'D:/Program Files/Unity/2020.3.11f1/Editor/Data/PlaybackEngines'

Cmd: initializeCompiler

Cmd: compileSnippet

insize=1647 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDBASE uKW=DIRECTIONAL SHADOWS_SCREEN LIGHTPROBE_SH _PARALLAXMAP dKW=FOG_LINEAR FOG_EXP FOG_EXP2 INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP SHADOWS_SHADOWMASK DYNAMICLIGHTMAP_ON LIGHTMAP_ON LIGHTMAP_SHADOW_MIXING DIRLIGHTMAP_COMBINED VERTEXLIGHT_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS

UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=227 mask=6 start=68 ok=1 outsize=2910

Cmd: compileSnippet
insize=1647 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDBASE uKW=DIRECTIONAL SHADOWS_SCREEN _EMISSION _PARALLAXMAP dKW=FOG_LINEAR FOG_EXP FOG_EXP2 INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _DETAIL_MULX2 _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _GLOSSYREFLECTIONS_OFF LIGHTPROBE_SH SHADOWS_SHADOWMASK DYNAMICLIGHTMAP_ON LIGHTMAP_ON LIGHTMAP_SHADOW_MIXING DIRLIGHTMAP_COMBINED UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=106 ok=1 outsize=3966

Cmd: compileSnippet
insize=1487 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD_DELTA cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP

SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDADD uKW=POINT SHADOWS_CUBE dKW=FOG_LINEAR FOG_EXP FOG_EXP2 _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 _PARALLAXMAP DIRECTIONAL SPOT POINT_COOKIE DIRECTIONAL_COOKIE SHADOWS_SHADOWMASK LIGHTMAP_SHADOW_MIXING SHADOWS_DEPTH SHADOWS_SOFT SHADOWS_SCREEN UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=106 ok=1 outsize=4674

Cmd: compileSnippet insize=1311 file=Assets/DefaultResourcesExtra/Standard pass=ShadowCaster cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_SHADOWCASTER uKW=SHADOWS_CUBE dKW=INSTANCING_ON _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _PARALLAXMAP SHADOWS_DEPTH UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING

UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=227 mask=6 start=139 ok=1 outsize=1330

Cmd: compileSnippet insize=1590 file=Assets/DefaultResourcesExtra/Standard pass=DEFERRED cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_DEFERRED uKW=LIGHTPROBE_SH dKW=INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _PARALLAXMAP DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1838

Cmd: compileSnippet

insize=1590 file=Assets/DefaultResourcesExtra/Standard pass=DEFERRED cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_DEFERRED uKW=LIGHTPROBE_SH _EMISSION _PARALLAXMAP dKW=INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=3002

Cmd: compileSnippet
insize=1590 file=Assets/DefaultResourcesExtra/Standard pass=DEFERRED cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR

UNITY_PASS_DEFERRED uKW=_PARALLAXMAP dKW=_EMISSION INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1578 UNITY_PASS_DEFERRED uKW=_PARALLAXMAP dKW=_EMISSION INSTANCING_ON

_NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1578 UNITY_PASS_DEFERRED uKW=_PARALLAXMAP dKW=_EMISSION INSTANCING_ON

_NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS

UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1578
UNITY_PASS_DEFERRED uKW=_PARALLAXMAP dKW=_EMISSION INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1578

Cmd: compileSnippet

insize=1442 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDBASE uKW=DIRECTIONAL VERTEXLIGHT_ON _NORMALMAP dKW=FOG_LINEAR FOG_EXP FOG_EXP2 _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP LIGHTPROBE_SH SHADOWS_SHADOWMASK DYNAMICLIGHTMAP_ON LIGHTMAP_ON LIGHTMAP_SHADOW_MIXING SHADOWS_SCREEN UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=1 mask=6 start=237 ok=1 outsize=2126

Cmd: compileSnippet
insize=1442 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD cachingPP=1

UNITY_PASS_DEFERRED uKW=_PARALLAXMAP dKW=_EMISSION INSTANCING_ON _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED UNITY_HDR_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS

UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Fragment platform=d3d11 reqs=227 mask=6 start=168 ok=1 outsize=1578
ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0
pKW=UNITY_ENABLE_REFLECTION_BUFFERS
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS
UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION
UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP
SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA
UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR
UNITY_PASS_FORWARDBASE uKW=DIRECTIONAL SHADOWS_SCREEN
LIGHTPROBE_SH dKW=_EMISSION FOG_LINEAR FOG_EXP FOG_EXP2 _NORMALMAP
_ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
_METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
_SPECULARHIGHLIGHTS_OFF _GLOSSYREFLECTIONS_OFF
SHADOWS_SHADOWMASK DYNAMICLIGHTMAP_ON LIGHTMAP_ON
LIGHTMAP_SHADOW_MIXING UNITY_NO_DXT5nm
UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS
UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Fragment platform=d3d11 reqs=227 mask=6 start=237 ok=1 outsize=7222

Cmd: compileSnippet
insize=1295 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD_DELTA cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDADD uKW=SPOT dKW=FOG_LINEAR FOG_EXP FOG_EXP2 _NORMALMAP _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP POINT DIRECTIONAL POINT_COOKIE DIRECTIONAL_COOKIE SHADOWS_SHADOWMASK LIGHTMAP_SHADOW_MIXING SHADOWS_DEPTH SHADOWS_SCREEN SHADOWS_CUBE UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=1 mask=6 start=272 ok=1 outsize=2102

Cmd: compileSnippet
insize=1295 file=Assets/DefaultResourcesExtra/Standard pass=FORWARD_DELTA cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR

UNITY_PASS_FORWARDADD uKW=POINT SHADOWS_CUBE _NORMALMAP dKW=FOG_LINEAR FOG_EXP FOG_EXP2 _ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A _SPECULARHIGHLIGHTS_OFF _DETAIL_MULX2 DIRECTIONAL SPOT POINT_COOKIE DIRECTIONAL_COOKIE SHADOWS_SHADOWMASK LIGHTMAP_SHADOW_MIXING SHADOWS_DEPTH SHADOWS_SCREEN UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=1 mask=6 start=272 ok=1 outsize=2442

Cmd: compileSnippet insize=1004 file=Assets/DefaultResourcesExtra/Standard pass=ShadowCaster cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_SHADOWCASTER uKW=SHADOWS_CUBE dKW=_ALPHATEST_ON _ALPHABLEND_ON _ALPHAPREMULTIPLY_ON _METALLICGLOSSMAP _SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A SHADOWS_DEPTH UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER

UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=1 mask=6 start=301 ok=1 outsize=250

Cmd: compileSnippet insize=17587 file=Assets/DefaultResourcesExtra/Legacy Shaders/Diffuse pass=FORWARD cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR UNITY_PASS_FORWARDBASE uKW=DIRECTIONAL VERTEXLIGHT_ON dKW=INSTANCING_ON FOG_LINEAR FOG_EXP FOG_EXP2 LIGHTPROBE_SH SHADOWS_SHADOWMASK DYNAMICLIGHTMAP_ON LIGHTMAP_ON LIGHTMAP_SHADOW_MIXING DIRLIGHTMAP_COMBINED SHADOWS_SCREEN UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS UNITY_METAL_SHADOWS_USE_POINT_FILTERING UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Vertex platform=d3d11 reqs=33 mask=6 start=20 ok=1 outsize=1250

Cmd: compileSnippet insize=4377 file=Assets/DefaultResourcesExtra/Legacy Shaders/Diffuse pass=FORWARD cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0

pKW=UNITY_ENABLE_REFLECTION_BUFFERS
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS
UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION
UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP
SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA
UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR
UNITY_PASS_FORWARDADD uKW=DIRECTIONAL dKW=FOG_LINEAR FOG_EXP
FOG_EXP2 POINT SPOT POINT_COOKIE DIRECTIONAL_COOKIE UNITY_NO_DXT5nm
UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS
UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Vertex platform=d3d11 reqs=33 mask=6 start=263 ok=1 outsize=1250

Cmd: compileSnippet
insize=12683 file=Assets/DefaultResourcesExtra/Legacy Shaders/Diffuse pass=PREPASS
cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0
pKW=UNITY_ENABLE_REFLECTION_BUFFERS
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS
UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION
UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP
SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA
UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR
UNITY_PASS_PREPASSFINAL uKW=UNITY_HDR_ON dKW=INSTANCING_ON
FOG_LINEAR FOG_EXP FOG_EXP2 LIGHTPROBE_SH DYNAMICLIGHTMAP_ON
SHADOWS_SHADOWMASK LIGHTMAP_ON DIRLIGHTMAP_COMBINED
UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS
UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2

UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=33 mask=6 start=109 ok=1 outsize=674

Cmd: compileSnippet insize=1562 file=Assets/DefaultResourcesExtra/Hidden/Internal-PrePassLighting pass=Pass 1 cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2 UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3 UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0 type=Fragment platform=d3d11 reqs=227 mask=6 start=104 ok=1 outsize=378

Cmd: compileSnippet insize=731 file=Assets/DefaultResourcesExtra/Sprites/Default pass=Pass 0 cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0 pKW=UNITY_ENABLE_REFLECTION_BUFFERS UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR uKW=PIXELSNAP_ON dKW=ETC1_EXTERNAL_ALPHA INSTANCING_ON UNITY_NO_DXT5nm UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS

UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Fragment platform=d3d11 reqs=1 mask=6 start=32 ok=1 outsize=454

Cmd: compileSnippet
insize=3023 file=Assets/DefaultResourcesExtra/UI/UI/Default pass=Default cachingPP=1
ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0
pKW=UNITY_ENABLE_REFLECTION_BUFFERS
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS
UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION
UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP
SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA
UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR
uKW=UNITY_UI_CLIP_RECT UNITY_UI_ALPHACLIP dKW=UNITY_NO_DXT5nm
UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS
UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Fragment platform=d3d11 reqs=1 mask=6 start=49 ok=1 outsize=794

Cmd: compileSnippet
insize=6280 file=Assets/DefaultResourcesExtra/Hidden/VideoDecode
pass=Flip_RGBA_To_RGBA cachingPP=1 ppOnly=0 stripLineD=0 buildPlatform=19 rsLen=0

pKW=UNITY_ENABLE_REFLECTION_BUFFERS
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS
UNITY_PBS_USE_BRDF1 UNITY_SPECCUBE_BOX_PROJECTION
UNITY_SPECCUBE_BLENDING UNITY_ENABLE_DETAIL_NORMALMAP
SHADER_API_DESKTOP UNITY_COLORSPACE_GAMMA
UNITY_LIGHT_PROBE_PROXY_VOLUME UNITY_LIGHTMAP_FULL_HDR
uKW=ADJUST_TO_LINEARSPACE dKW=UNITY_NO_DXT5nm
UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS
UNITY_METAL_SHADOWS_USE_POINT_FILTERING
UNITY_NO_SCREENSPACE_SHADOWS UNITY_PBS_USE_BRDF2
UNITY_PBS_USE_BRDF3 UNITY_NO_FULL_STANDARD_SHADER
UNITY_HARDWARE_TIER1 UNITY_HARDWARE_TIER2 UNITY_HARDWARE_TIER3
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS
UNITY_LIGHTMAP_DLDR_ENCODING UNITY_LIGHTMAP_RGBM_ENCODING
UNITY_VIRTUAL_TEXTURING UNITY_PRETRANSFORM_TO_DISPLAY_ORIENTATION
UNITY_ASTC_NORMALMAP_ENCODING SHADER_API_GLES30 flags=0 lang=0
type=Fragment platform=d3d11 reqs=33 mask=6 start=205 ok=1 outsize=330

Cmd: shutdown

GameManager.cs

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

public class GameManager : MonoBehaviour {

    // Ініціалізація змінних.
    private float size;

    [SerializeField]
    GameObject platform;
```

```
private Vector3 lastPosition;

[SerializeField]
GameObject gems;

// Use this for initialization
void Start () {

size = platform.transform.localScale.x;

lastPosition = platform.transform.position;


// Create the first initial playforms.

for (int x = 0; x < 15; x++) {


SpawnZ ();

}


// Call the SpawnPlatform every 2 seconds.

InvokeRepeating ("SpawnPlatform", 2f, 0.2f);
}



// Генерація платформ.
private void SpawnPlatform() {

int random = Random.Range (0, 6);
```

```
int gemsRandom = Random.Range (0, 7);


if (random < 3) {
SpawnX ();
}
if (random >= 3) {
SpawnZ ();
}
if (gemsRandom < 2) {
SpawnGem ();
}
}
private void SpawnX() {
GameObject _platform = Instantiate (platform) as GameObject;
_platform.transform.position = lastPosition + new Vector3 (size, 0f,0f);
lastPosition = _platform.transform.position;
// Платформі присвоюється колір матеріалу Color1.
Material newMat = Resources.Load("Color1", typeof(Material)) as Material;
_platform.GetComponent<Renderer>().material = newMat;
}
private void SpawnZ() {
GameObject _platform = Instantiate (platform) as GameObject;
_platform.transform.position = lastPosition + new Vector3 (0f,0f,size);
lastPosition = _platform.transform.position;
// Платформі присвоюється колір матеріалу Color2
Material newMat = Resources.Load("Color2", typeof(Material)) as Material;
_platform.GetComponent<Renderer>().material = newMat;
}
```

```
        // Генерація кристалів.
        private void SpawnGem() {

        Instantiate (gems, lastPosition+ new Vector3(0f,0.7f,0f), gems.transform.rotation);
        // Присвоюємо колір для кристалу.

        int rand = Random.Range(1,3);

        Material newMat = Resources.Load("Color" + (rand.ToString()), typeof(Material)) as Material;

        gems.GetComponent<Renderer>().material = newMat;
        }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Диплом_Шагоян.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word
Диплом_Шагоян.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Програма.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Шагоян.ppt	Презентація кваліфікаційної роботи

ВІДГУК

на кваліфікаційну роботу бакалавра

на тему:

"Проектування та розробка тривимірного ігрового додатку у жанрі Arcade на движку Unity".

студента групи 122-18СК-2 Шагояна Роберта Арамовича

Розроблене в кваліфікаційній роботі програмне забезпечення призначене для створення інформаційної системи, яке спроможне організовувати користувача з ігровим додатком, яке використовуватиме самий користувач.

Актуальність розробленого програмного продукту полягає в створенні такого додатку, який матиме попит серед цільової аудиторії – гравців у відеоігри. Перевага запропонованої програми в тому, що вона захоплює ігровим процесом та інтуїтивно зрозуміла.

Для вирішення поставлених задач при розробці додатку були здійснені наступні дії:

1. Розробка логічної моделі програми.
2. Проектування ігрового додатку.
3. Розробка архітектури програмного забезпечення.
4. Розробка простого і зрозумілого інтерфейсу програми.

Додаток розроблений на мові C# в середовищі Unity Engine з використанням Microsoft Visual Studio 2019.

Практична значимість створення даного програмного продукту полягає в забезпеченні користувача ігровим процесом гри у жанрі Arcade.

Працездатність представленої програми підтверджена налагоджувальними випробуваннями та тестуванням програми.

В економічному розділі визначено трудомісткість розробленого додатку, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення. Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра за спеціальністю 122 Комп’ютерні науки.

Оформлення пояснювальної записки до кваліфікаційної роботи виконано відповідно до стандартів на програмну документацію. Недоліком розглянутої роботи є неповний опис алгоритму функціонування системи.

Кваліфікаційна робота виконана самостійно та заслуговує оцінки 92 бала, «відмінно», а студенту Шагояну Роберту Арамовичу присвоєння кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Керівник кваліфікаційної роботи
ас. каф. ПЗКС, к.т.н. **Приходченко С.Д.**

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

на тему:

"Розробка 3D ігрового застосунку з використанням технологій віртуальної реальності та застосуванням неевклідової геометрії."

студента групи 122-18СК-2 Шагояна Роберта Арамовича

Кваліфікаційна робота виконана в повному обсязі в співвідношенні з технічним завданням.

Основною метою програми є організація ігрового процесу (геймплею), що базується на взаємодії користувача і ігрового пристрою за допомогою візуального інтерфейсу. Було розроблено ігровий додаток у жанрі Arcade.

Тема проекту є актуальною на ринку програмного забезпечення. Аналогів таких продуктів на ринку не багато, ігрові додатки завжди мали попит на ринку.

В якості інструмента для реалізації була використана мова програмування C#. Сам проект був реалізований за допомогою Unity Engine.

У вступі проведений аналіз аналогів вирішення проблеми в цілому, аналоги конкретного методу вирішення проблеми, перераховані їх переваги й недоліки. На основі них були сформовані вимоги до програного забезпечення. В теоретичній частині наявні дані про мету розробки, сферу застосування продукту та його актуальність. Описані функціональне призначення й логічна структура. Вибір технологій й підхід до реалізації повністю обґрунтовані.

Результати реалізації проекту добре описані й підтверджені тестуванням. Проектна документація описує усі можливі варіанти взаємодії з продуктом.

Список літератури налічує більше 20 джерел, що свідчить про вміння працювати з літературою та іншими джерелами інформації.

Робота оцінюється на 93 бала «відмінно».

Рецензент кваліфікаційної роботи

к.т.н., доцент каф. БІТ **Герасіна О.В.**