МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Острозька академія»

Економічний факультет

Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ

на здобуття освітнього ступеня бакалавра

на тему: **«РОЗРОБКА ІГРОВОГО ЗАСТОСУНКУ ПІД МОБІЛЬНІ ПЛАТФОРМИ В ЖАНРІ АРКАДИ НА ІГРОВОМУ РУШІЮ UNITY»**

Виконав: студент 4 курсу, групи КН-41 першого (бакалаврського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Комп'ютерні науки»
Кухта Сергій Петрович

Керівник: кандидат технічних наук, доцент кафедри ЕММІТ
Шевченко Галина Володимирівна

Рецензент: Front-end Developer "DOODLE", LLC
Місай Володимир Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 11 від «18» травня 2023 р.

Острог, 2023

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*
Кафедра: *економіко-математичного моделювання та інформаційних технологій*
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

З А В Д А Н Н Я
на кваліфікаційну роботу/проєкт студента
Кухти Сергія Петровича

1. Тема роботи: Розробка ігрового застосунку під мобільні платформи в жанрі аркади на ігровому рушію Unity.

Керівник проєкту: Шевченко Галина Володимирівна.

Затверджено наказами ректора НаУОА: від 31 жовтня 2022 року №77, від 28 квітня 2023 року №39.

2. Термін здачі студентом закінченої роботи/проєкту: 31 травня 2023 року.

3. Вихідні дані до роботи/проєкту: дана робота полягає у розробці мобільного застосунку в жанрі аркади з використанням Unity, C#, фреймворків: Zenject, UniRx.

4. Перелік завдань, які належить виконати: створити дизайн застосунку, створити 3д моделі, реалізувати основну механіку, створити конфіг з економікою застосунку, створити генератор рівнів, добавити звуки, добавити вібрації, добавити туторіал, оптимізувати застосунок.

5. Перелік графічного матеріалу: рисунки, таблиці.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

1	Шевченко Г.В.	01.12.2022	01.12.2022
2	Шевченко Г.В.	01.12.2022	01.12.2022
3	Шевченко Г.В.	01.12.2022	01.12.2022

7. Дата видачі завдання: 01.12.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Затвердження теми проєкту	до 31.10.2022 р.	
2.	Постановка завдання	до 01.12.2022 р.	
3.	Розробка графіки	до 01.01.2023 р.	
4.	Розробка архітектури	до 01.02.2023 р.	
5.	Розробка модулів застосунку	до 01.03.2023 р.	
6.	Попередній захист кваліфікаційної роботи	до 18.05.2023 р.	
7.	Здача кваліфікаційної роботи/проєкту на кафедру	31.05.2023 р.	

Студент: _____ Сергій КУХТА

Керівник кваліфікаційної роботи: _____ Галина ШЕВЧЕНКО

АНОТАЦІЯ

кваліфікаційної роботи/проєкту

на здобуття освітнього ступеня бакалавра

Тема: *Розробка ігрового застосунку під мобільні платформи в жанрі аркади на ігровому рушію Unity.*

Автор: *Кухта Сергій Петрович.*

Науковий керівник: *Шевченко Галина Володимирівна.*

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *71 с., 46 рис., 8 табл., 8 джерел.*

Ключові слова: *Unity, C#, розробка мобільних ігор, інтерфейс користувача, архітектура.*

Короткий зміст праці:

Завданням кваліфікаційної роботи/проєкту було розробка ігрового застосунку під мобільні платформи в жанрі аркади на ігровому рушію Unity з використанням мови програмування c#.

Зазначений проєкт є реалізацією ігрового застосунку на Unity. З можливістю генерації необмеженої кількості рівнів, з усіма необхідними модулями застосунку: туторіалом, збереженням, звуками та вібраціями. З логічною основною механікою гри, яка полягає в об'єднанні ножів та розрізанні фруктів, щоб заповнити блендер та отримати сік.

The task of the qualification work/project was to develop a game application for mobile platforms in the arcade genre using the Unity game engine using the c# programming language. This project is an implementation of a game application in Unity. With the ability to generate an unlimited number of levels, with all the necessary application modules: tutorial, save, sounds, and vibrations. With a logical basic game mechanic, which is to combine knives and cut fruit to fill a blender and get juice.

ЗМІСТ

ВСТУП

Мобільні ігри набувають все більшої популярності. За оцінками, у 2023 році в ігри на своїх смартфонах грали більш ніж 2 мільярди людей. Цей тренд пояснюється швидким розвитком та покращенням характеристик смартфонів, широким асортиментом доступних ігор і зручністю грати у них в будь-який час і в будь якому місці. Аркадні ігри стали одними з найпопулярніших жанрів, завдяки простій механіці і можливості відпочити та зняти стрес. Яскрава графіка та швидкий рух ігрових об'єктів допомагають забути про проблеми і відпочити під час гри, що стає особливо важливим в нашому сучасному світі.

Жанр аркади є одним з найпопулярніших жанрів серед мобільних додатків, оскільки має просту і зрозумілу механіку, яка дозволяє користувачам легко відчути себе в грі. Основна мета аркад - бути веселою і захоплюючою розвагою.

Цей мобільний застосунок пропонує дещо нове та оригінальне - комбінування ножів для нарізання фруктів та створення соку. Ця механіка дозволяє гравцям зануритися в цікавий ігровий процес і відчути задоволення від створення нових комбінацій ножів. Крім того, гра може допомогти розвинути логіку і покращити пам'ять, що може стати в нагоді не тільки в грі, але і в повсякденному житті.

Мета кваліфікаційної роботи/проєкту – реалізація мобільного застосунку в жанрі аркади.

Нами були вирішені перелік таких завдань:

1. Опис предметного середовища
2. Огляд наявних аналогів
3. Розробка технічного завдання
4. Реалізація функціоналу системи

5. Тестування

Об’єктом дослідження даної кваліфікаційної роботи/проєкту є створення повноцінного мобільного застосунку у жанрі аркад.

Предметом дослідження є список технологій та фреймворків для розробки застосунку, а саме: Unity, C#, Zenject, UniRx, DotWeen, ShaderGraph.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Вибір середовища розробки

1.1.1. Вибір ігрового рушія

Вибирати будемо з двох найпопулярніших рушіїв, а саме Unity та Unreal Engine. Але для початку, потрібно визначити що таке ігровий рушій, та що в ньому повинно бути.

Ігровий рушій - це програмний фреймворк, який надає розробникам інструменти та функціональність, необхідні для створення відеоігор. По суті, це набір бібліотек, API та інструментів, які використовуються для спрощення процесу розробки ігор шляхом надання готових модулів та компонентів.

Unity - це кросплатформенний ігровий двигун, який підтримує різні платформи, такі як Windows, macOS, Linux, Android, iOS та веб-браузери. Він популярний серед інді-розробників ігор та невеликих студій, оскільки його відносно легко вивчити та використовувати. Unity використовує C# як основну мову програмування і має велику спільноту розробників, що дозволяє легко знайти навчальні посібники та ресурси в Інтернеті [1]. (Рис. 1.1.)

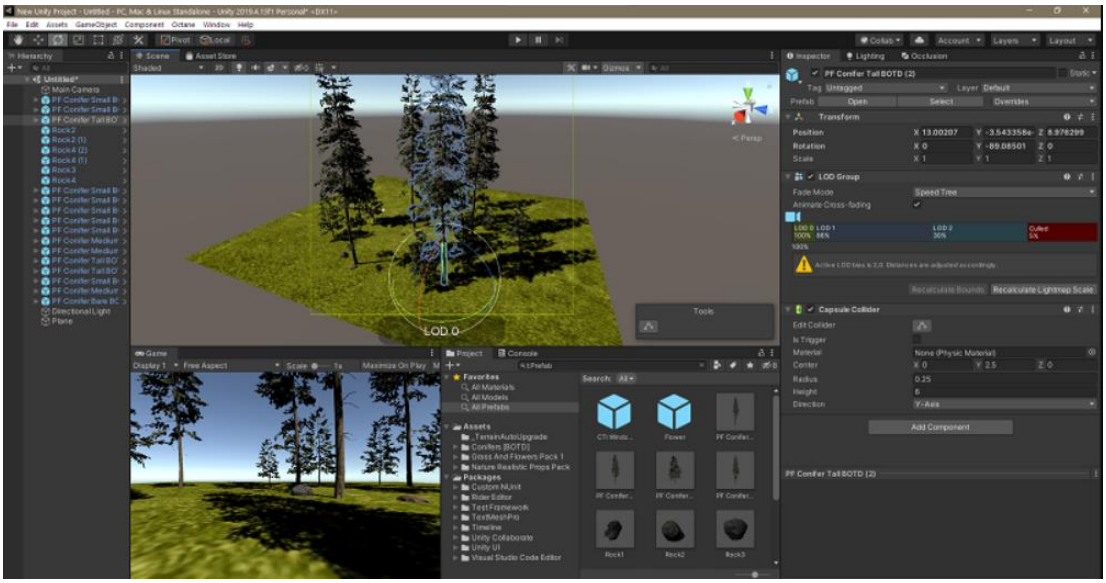

Рис 1.1. Інтерфейс Unity

Джерело: [https://uk.wikipedia.org/wiki/Unity_%28ігровий_рушій%29]

Unreal Engine - це ігровий двигун, розроблений компанією Epic Games, який в основному використовується для розробки високоякісних AAA-ігор для консолей та ПК. Він відомий своєю передовою графікою та можливостями рендерингу, що робить його популярним вибором серед великих студій розробки ігор. Unreal Engine підтримує багато платформ, включаючи Windows, macOS, Linux, iOS, Android та консолі. Він використовує C++ як основну мову програмування. (Рис. 1.2)

Рис. 1.2. Інтерфейс Unreal Engine

Джерело: [https://beforesandafters.com/2021/05/27/you-can-get-early-access-to-unreal-engine-5-now/]

Відмінності між Unity та Unreal Engine наведені в наступній таблиці.

Таблиця 1.1

Порівняльна таблиця ігрових рушіїв

Критерії	Unity	Unreal Engine
Наявність безкоштовної версії	Присутня	Присутня
Кросплатформенна підтримка	Присутня	Присутня

Вимоги до обладнання	Середні	Високі
Складність вивчення	Середня	Висока
Широка підтримка мобільних додатків	Присутня	Недостатня
Можливості рендеренгу	Середні	Високі

В нашому випадку важливими критеріями є: простота використання, складність вивчення, наявність безкоштовної версії, достатня підтримка версій ігор під мобільні платформи. З цим краще справляється Unity, на ньому і зупинимось.

1.1.2. Вибір середовища для розробки 3д моделей

Вибирати будем серед двох найпопулярніших середовищ в ігровій індустрії, а саме Blender і Autodesk Maya.

Maya - програмне забезпечення для 3D-моделювання, анімації та рендерингу, розроблене компанією Autodesk, яке завдяки своїм потужним можливостям і гнучкості вже багато років широко використовується в кіно, ігровій та телевізійній індустрії. Maya відома своїм широким інструментарієм, який включає розширені можливості анімації персонажів, потужні інструменти анімації, робочий процес на основі часової шкали та рушій для імітації динаміки різних матеріалів, волосся та рідин.

Можливості моделювання в Maya включають як інструменти полігонального моделювання, так і інструменти NURBS, що дозволяють створювати детальні моделі з точними кривими та поверхнями.

Blender - це безкоштовне програмне забезпечення з відкритим вихідним кодом, яке надає можливості для 3D-моделювання, анімації та візуалізації. Він є популярний завдяки зручному інтерфейсу, безплатній версії, великій бібліотеці плагінів, та налаштованим гарячим клавішам, що дозволяє навіть початківцям легко почати створювати 3D-моделі та анімацію [2].

Blender надає широкий спектр інструментів для створення детальних моделей і текстур, включаючи ліплення, малювання текстур і інструменти для композиції. Крім того, він містить вбудований ігровий рушій, який дозволяє створювати інтерактивні 3D-ефекти без використання зовнішнього програмного забезпечення.

Відмінності між Autodesk Maya та Blender наведені в наступній таблиці.

Таблиця 1.2

Порівняльна таблиця середовищ розробки 3д моделей

Критерії	Autodesk Maya	Blender
Наявність безкоштовної версії	Безплатний пробний період 30 днів	Присутня
Наявність відкритого вихідного коду	Відсутня	Присутня
Базові можливості для 3д моделювання	Присутні	Присутні
Складність вивчення	Висока	Середня
Підтримка формату експорту для Unity	Присутня	Присутня

В цьому випадку, для вибору середовища для розробки 3д моделей, ключовим фактором буде наявність безкоштовної версії, оскільки обидва середовища добре справляються з потребами моделювання. Тому вибираємо Blender.

1.1.3. Вибір середовища розробки 2д графіки

Серед доступних варіантів, які будуть вирішувати потреби гри, були вибрані Adobe Photoshop та Affinity Photo.

Affinity Photo - це програма для редагування растрових зображень, яка надає повний набір інструментів для професійного редагування фотографій та графічного дизайну. Affinity Photo має сучасний та інтуїтивно зрозумілий інтерфейс, який полегшує навігацію та використання різних інструментів та функцій. Програма

пропонує широкий спектр інструментів для редагування фотографій, таких як експозиція, криві та баланс кольорів, а також інструменти цифрового малювання, такі як пензлі та олівці.

Photoshop - це програма для редагування растрових зображень, яка протягом десятиліть була основою креативної індустрії. Ця програма пропонує широкий спектр інструментів і функцій для професійного редагування фотографій, цифрового малювання і графічного дизайну, а також широко використовується в розробці ігор для створення текстур і ресурсів.

Photoshop має повний набір інструментів і функцій, які включають коригувальні шари, фільтри і маски шарів для редагування фотографій, а також інструменти цифрового малювання [7].

Відмінності між Unity та Unreal Engine наведені в наступній таблиці.

Таблиця 1.3

Порівняльна таблиця середовищ розробки 2д графіки

Критерії	Adobe Photoshop	Affinity Photo
Базові можливості редагування фотографій	Присутні	Присутні
Наявність безкоштовної версії	Безплатний пробний період 30 днів	Відсутня
Використання в розробці ігор	Високе	Середнє
Наявність туторіалів	Високе	Середнє

У випадку з середовищем для розробки 2д графіки, ключову роль відіграють базові можливості редагування зображень та кількість туторіалів в інтернеті. З цим краще справляється Adobe Photoshop, тому вибираємо його.

1.1.4. Вибір середовища написання коду

Серед можливих варіантів, найкращими є Visual Studio та Rider.

Visual Studio - це комплексне середовище розробки Microsoft, яке широко використовується для розробки програмного забезпечення, включаючи ігри, завдяки своїм потужним функціям та простоті використання. Воно надає розширений набір інструментів для кодування, налагодження та тестування, а також легко налаштовується відповідно до потреб окремих розробників та команд.

Rider - це середовище розробки, розроблене компанією JetBrains. Воно має розширений набір інструментів та функцій для кодування, налагодження та тестування, а також підтримує декілька мов програмування. Rider також має сильну інтеграцію з ігровим движком Unity, що робить його популярним вибором для розробників ігор. У ньому є редактор, специфічний для Unity, який надає засоби завершення коду, підсвічування синтаксису та налагодження, що полегшує роботу розробників. Також, в ньому є вбудований плагін ReSharper, який сильно пришвидшує роботу з проектами.

ReSharper - це плагін, який розроблений компанією JetBrains та надає розширену допомогу в написанні коду та підвищенні продуктивності розробників в Visual Studio та Rider. Він забезпечує широкий спектр функцій для аналізу коду, рефакторингу та навігації, що дозволяє розробникам швидко та ефективно створювати високоякісний код. ReSharper широко використовується в .NET-спільноті розробників.

Відмінності між Rider та Visual Studio наведені в наступній таблиці.

Таблиця 1.4

Порівняльна таблиця середовищ розробки коду

Критерії	Rider	Visual Studio
Наявність безкоштовної версії	Безкоштовна для студентів	Присутня

Швидкодія середовища	Висока	Середня
Наявність ReSharper	Вбудований	Потрібно встановлювати додатково
Рефакторинг коду	Присутній	Присутній

Як середовище для розробки коду для Unity, краще підходить Rider, оскільки швидкість розробки гри в цій сфері відіграє ключову роль.

1.1.5. Висновки

Проаналізувавши всі доступні середовища, їх аналоги, відмінності між ними, був обраний наступний стек:

1. Ігровий двигун - Unity
2. Середовище розробки коду - Rider
3. Середовище розробки 3д моделей - Blender
4. Середовище розробки 2д графіки - Photoshop

1.2. Опис предметного середовища

Unity наразі є найпопулярнішим ігровим рушієм для розробки мобільних ігор. Ігри та додатки, створені за допомогою Unity, можна експортувати на 15 платформ, включаючи iOS, Android, Windows, Linux, PlayStation та інші. Для забезпечення своєї функціональності Unity використовує багато проміжного програмного забезпечення, такого як DirectX, OpenGL, Beast, SubStance, Fmod, PhysX та RakNet.

Однією з ключових переваг використання Unity для розробки ігор є його великий магазин компонентів. В магазині можна завантажувати та імпортувати моделі, текстури, шейдери, допоміжні скрипти та інші матеріали, які можна використовувати як в освітніх, так і в комерційних цілях. Компонентно-орієнтований підхід Unity дозволяє розробникам створювати різні об'єкти та налаштовувати їх, а також додавати нові компоненти за потреби. (Рис. 1.1)

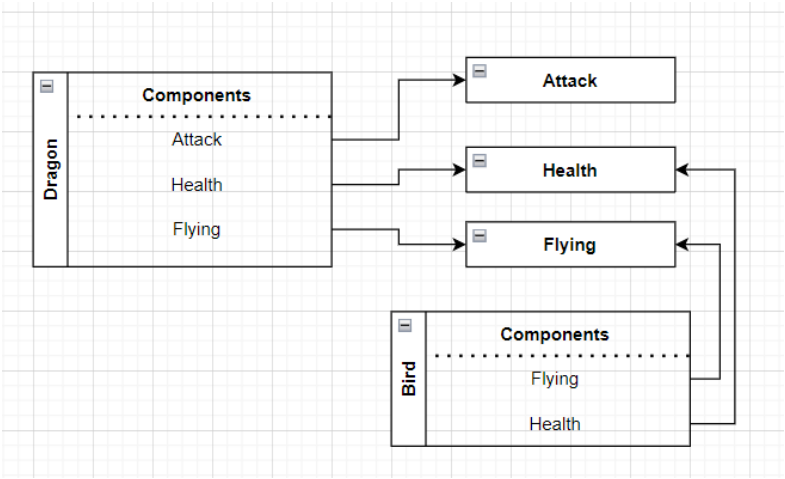

Рис. 1.1 - Зв’язок між об’єктами та компонентами

Джерело: [створено автором]

Рушій Unity також є безкоштовним, якщо річний оборот компанії не перевищує $100 000 або якщо він використовується в некомерційних цілях. Крім того, він має багато вбудованих функцій, включаючи візуальне написання сценаріїв, систему анімації та систему створення частинок. Unity відносно легко вивчити і він має зручний інтерфейс, що робить його кращим вибором для багатьох розробників. Він навіть використовується в інших галузях, крім ігрової, таких як автомобільна промисловість, де його застосовують для створення високоякісних візуалізацій складних моделей у реальному часі.

Основною мовою для програмування на Unity є C#.

C# - це потужна мова програмування, яка широко використовується для розробки ігор, особливо в поєднанні з ігровим рушієм Unity. Однією з причин популярності C# серед розробників ігор є те, що вона пропонує баланс продуктивності та простоти використання. C# - це мова високого рівня, яку легко вивчити, і вона надає такі функції, як збір сміття та автоматичне керування пам'яттю, які роблять програмування менш схильним до помилок та більш ефективним.

C# також є безпечною за типом мовою, що означає, що вона допомагає запобігти поширеним помилкам програмування, забезпечуючи дотримання суворих правил

набору тексту. Це може бути особливо корисно при розробці ігор, де помилки можуть коштувати дорого і їх важко налагоджувати [3].

Варто зазначити, що Unity побудований з використанням C++ для основних функцій, таких як візуалізація графіки, симуляція фізики та обробка звуку. Однак Unity надає API для написання сценаріїв на C#, що дозволяє розробникам ігор створювати логіку гри та контент за допомогою цієї мови.

1.3. Вибір платформ для розміщення гри

Публікувати наш додаток будем на двох платформах: Google Play Market та App Store. Ці платформи є найбільшими ринками для мобільних додатків на пристроях Android та iOS відповідно. Android є найбільш широко використовуваною мобільною платформою у світі, а Google Play Market є офіційним магазином додатків для Android. Щоб опублікувати свою гру в Play Market, потрібно створити обліковий запис розробника Google Play і заплатити 25 євро. (Рис. 1.2)

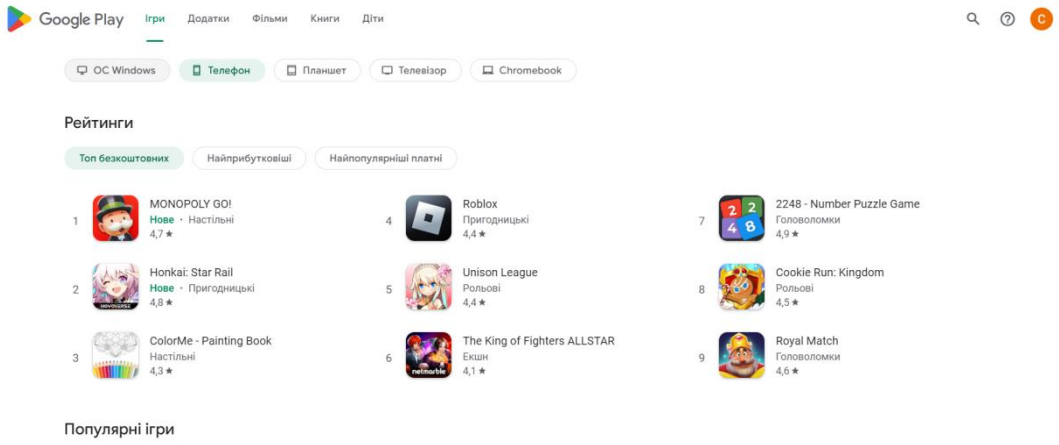

Рис.1.2 - Магазин додатків Play Market

Джерело: [https://play.google.com/store/games]

IOS використовується виключно на пристроях Apple, а App Store є офіційним магазином додатків для цієї платформи. Щоб опублікувати свою гру в App Store, потрібно створити обліковий запис розробника Apple і платити 99 доларів на рік. (Рис. 1.3)

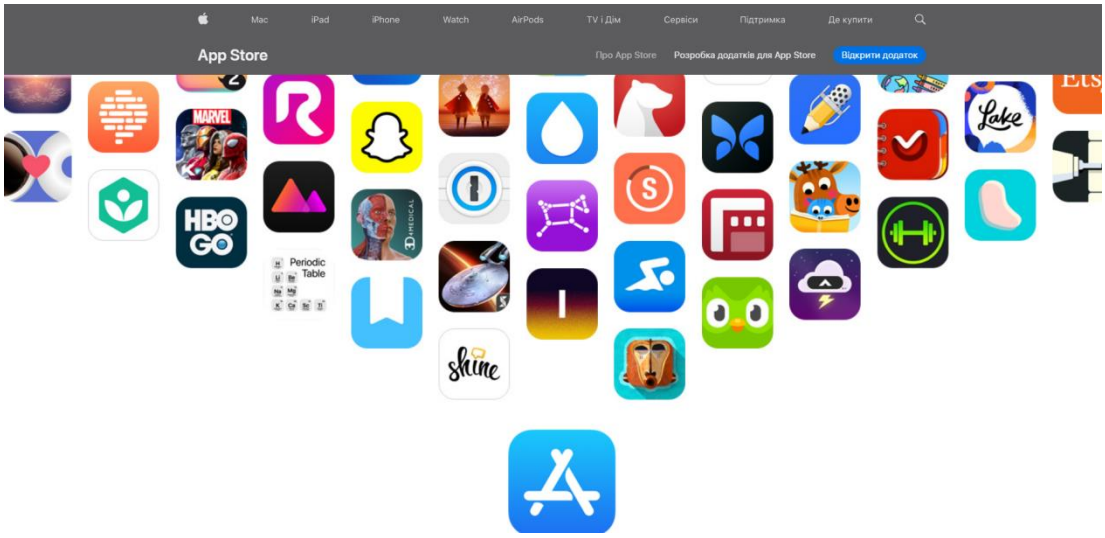

Рис. 1.3 - Магазин додатків AppStore

Джерело: [https://www.apple.com/app-store/]

Android має більше переваг, ніж iOS, особливо коли справа доходить до публікації додатків. Розміщення застосунків на iOS може бути проблематичним через незрозумілі причини відхилення запитів на хостинг, що може призвести до необхідності створення додаткового облікового запису розробника Apple. Також, пристрої iOS, як правило, коштують дорожче, ніж пристрої Android, тому потенційних користувачів від цієї платформи можна очікувати менше. Однак, iOS має більш уніфіковане апаратне та програмне середовище, що полегшує стабільну роботу на різних пристроях.

Отож, щоб не втратити потенційних гравців, результат кваліфікаційної роботи буде опублікований в PlayMarket та AppStore.

1.4. Аналіз конкурентів

Ігровий ринок гіпер казуальних ігор або аркад влаштований так, що на кожну популярну гру, знайдеться десяток клонів. Знаючи це, потрібно проаналізувати конкурентів, знайти доступні метрики таких ігор і дізнатись чи буде результат кваліфікаційної роботи в тренді. Основною механікою гри є об’єднання і розставляння елементів з подальшим скиданням їх на фрукти. Для знаходження

конкурентів достатньо в пошуку магазину додатків Google Play Market написати слово «Merge».

Були вибрані наступні додатки:

1. Merge Miners
2. Merge Town
3. Rush Royale - Custle Clash

Merge Miners - це додаток від студії Supersonic. У грі досить проста механіка, що добре для потенційних гравців. У ній потрібно комбінувати лопати, щоб зробити їх сильнішими, розміщувати їх у правильних положеннях, а потім кидати вниз, щоб знищувати блоки і заробляти монети, і в той же час насолоджуватися ефектами. У грі красива графіка і приємні звуки з поєднанням вібрації, а також широкий вибір ресурсів і предметів для збору і комбінування.

Виконуючи кваліфікаційну роботу, будем орієнтуватись на цей додаток, як основа для механіки гри. Хоч гра є цікавою і легкою, з дизайном, освітленням у неї не все так добре. Також є проблема з логікою рівнів, якщо робити не все правильно, що є типовим для гравців, буває неможливо пройти рівень і доводиться проходити гру заново. (Рис. 1.4)

Рис. 1.4 - Гра Merge Miners

Джерело:
[https://play.google.com/store/apps/details?id=com.tcg.fpsdungeonminer&hl=en_US]

Скориставшись веб-сервісом appmagic.rocks, можна переглянути інформацію про гру, про кількість її завантажень та про дохід, тим самим дізнавшись чи є механіка в тренді. (Рис. 1.5)

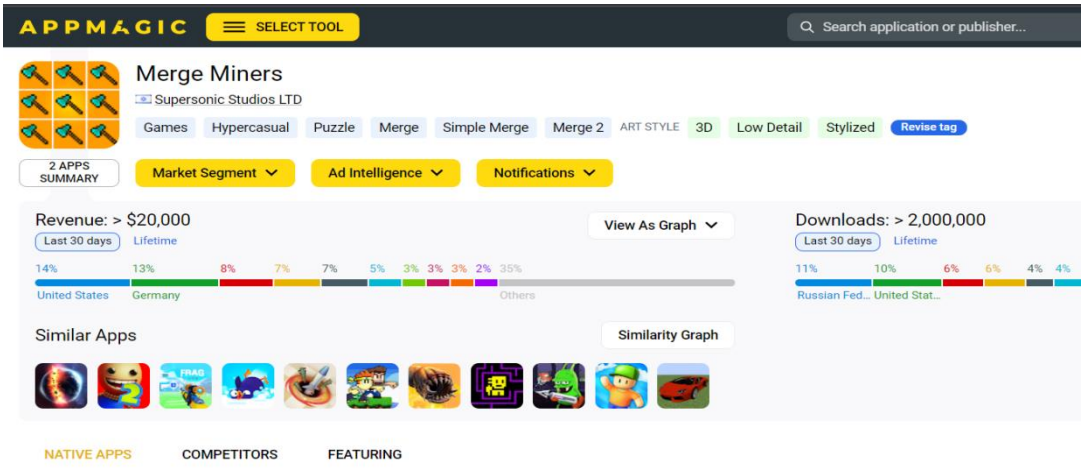

Рис. 1.5 - Статистика гри Merge Miners в appmagic

Джерело: [https://appmagic.rocks/google-play/merge-miners/com.tcg.fpsdungeonminer/info]

Гра Merge Town від Gram Games пропонує гравцеві створити своє власне місто. Гравці отримують ділянку землі та можуть об'єднувати будинки, щоб створювати більші та багатші будівлі. Кожен будинок, що був збудований, принесе гроші, які можна використовувати на покупку нових будівель, що дозволяє розширювати місто та збільшувати прибуток. Гра також пропонує використовувати ресурси, щоб відкривати нові будинки та збільшувати місто у різних місцях. (Рис. 1.6)

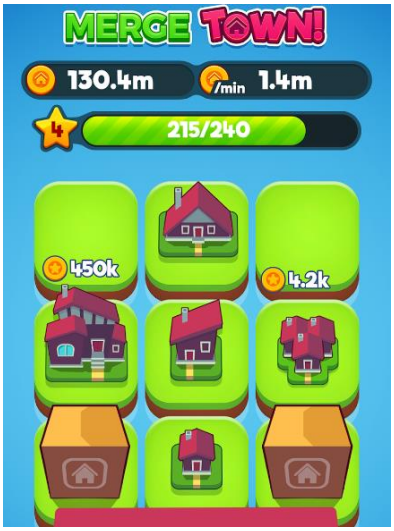

Рис 1.6 - Гра Merge Town

Джерело: [https://play.google.com/store/apps/details?id=com.gramgames.mergetown]

Гра є доволі простою, і складно уявити що вона може приносити непоганий дохід, але проаналізувавши інформацію з веб-сервісу appmagic.rocks, видно, що механіка таки в тренді. (Рис. 1.7)

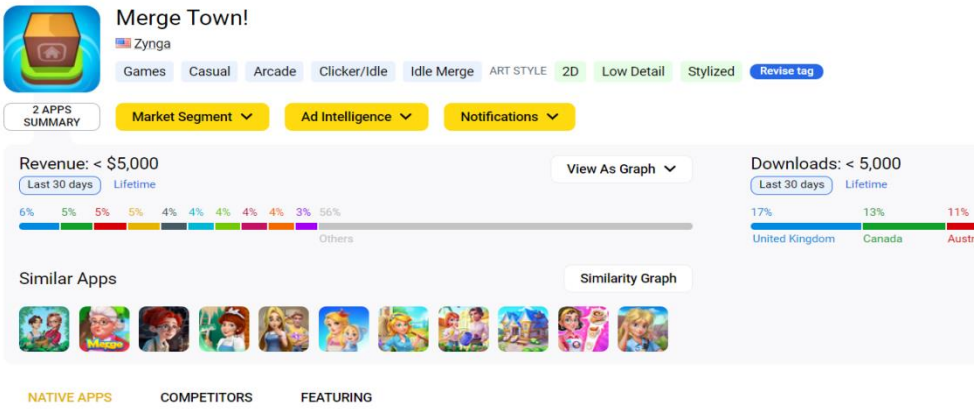

Рис. 1.7 - Статистика гри Merge Town в appmagic

Джерело: [https://appmagic.rocks/iphone/merge-town-/1232693016/info]

Ще однією грою для аналізу є Rush Royale - Castle Clash. Це проект від студії Upwake, який поєднує в собі кілька жанрів, таких як карткова гра, стратегія, RPG і Tower Defense. У грі потрібно захищати свої території, встановлюючи та об’єднуючи оборонні споруди на полі, які представлені у вигляді героїв, таких як лучники, маги, воїни та інші, що виконують роль веж. (Рис. 1.8)

Рис. 1.8 - Гра Rush Royale - Castle Clash

Джерело: [https://play.google.com/store/apps/details?id=com.my.defense]

Механіка цієї гри вже набагато складніша, додаток в трендах і його підтримують багато розробників. (Рис. 1.9)

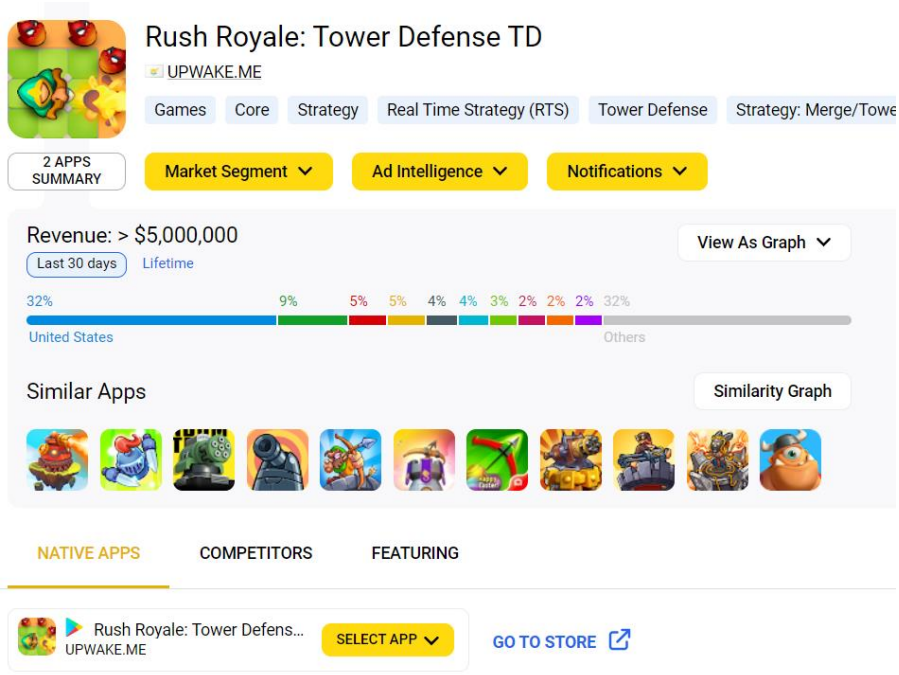

Рис. 1.9 - Статистика Rush Royale: Tower Defense в appmagic

Джерело: [https://appmagic.rocks/google-play/rush-royale-tower-defense-td/com.my.defense/info]

Проаналізувавши попередніх конкурентів можна провести SWOT-аналіз нашої гри.

Таблиця 1.5

SWOT-аналіз результату кваліфікаційної роботи

Сильні сторони	**Слабкі сторони**
Красива графіка Багато рівнів Проста та приємна механіка	Рівні однотипні, що може надоїсти
Можливості	**Загрози**
Нові рівні з перешкодами Нові види ножів	Недостатнє просування у магазинах мобільних додатків Схожі ігри з великим бюджетом

Висновок до розділу 1

У цьому розділі було обрано стек з необхідних технологій для виконання кваліфікаційної роботи. Також було проаналізовані магазини ігрових додатків, їх плюси та мінуси, а також можливі проблеми які будуть виникати на етапі публікації. Також було проаналізовані схожі мобільні застосунки в магазинах ігор. Використовуючи веб-сервіс appmagic була перевірена актуальність кваліфікаційної роботи, довели, що механіка гри знаходиться в трендах. Зробивши SWOT-аналіз застосунку, вивчили можливі перспективи та загрози для додатку.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Постановка задачі

Для досягнення мети кваліфікаційної роботи, а саме: «реалізація мобільного застосунку в жанрі аркади», потрібно виконати перелік основних завдань:

1. Реалізувати рух ножа за позицією вказівника (пальця гравця).
2. Реалізувати об’єднання ножів, анімацію збільшення при об’єднанні.
3. Реалізувати фізичний політ ножів вниз та їх обертання.
4. Розробити цікаву анімацію розрізання фрукта.
5. Добавити тріщини на фрукти.
6. Добавити блендер та реалізувати заповнення соком.
7. Реалізувати появу фруктів в блендері при розрізанні.

2.2. Опис необхідних математичних процесів

2.2.1. Виміри та система координат

Виміри та системи координат є ключовими поняттями у розробці ігор.

У 2D-графіці вимірювання виражаються в пікселях, які є найменшими одиницями зображення, що можуть бути відображені на екрані. Роздільна здатність зображення визначається його розміром у пікселях, де вища роздільна здатність призводить до більшої кількості пікселів і деталізації зображення.

Система координат у 2D - графіці має дві осі: вісь x та вісь y. Початок системи координат знаходиться у верхньому лівому куті полотна, значення по осі x збільшуються вправо, а значення по осі y збільшуються вниз. (Рис. 2.1)

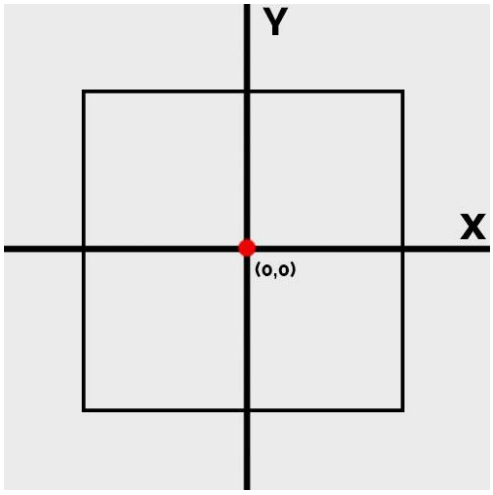

Рис. 2.1 - Двовимірна система координат

Джерело: [створено автором]

У 3D - графіці вимірювання виражаються в одиницях довжини, таких як метри або фути. Масштаб 3D - сцени визначається розміром і розташуванням об'єктів у ній, де більші об'єкти мають більший вплив на загальний масштаб.

Система координат у 3D - графіці має три осі: вісь x, вісь y та вісь z. Початок системи координат знаходиться в центрі сцени, при цьому значення x збільшуються вправо, значення y збільшуються вгору, а значення z збільшуються в напрямку до глядача. Така система координат відома як правостороння система координат. Щоб легше запам’ятати яка сторона за що відповідає, можна використовувати правило кольору - червоний, зелений, синій (RGB). Зазвичай, вісь x позначена червоним кольором, вісь y - зеленим, вісь z - синім. (Рис. 2.2)

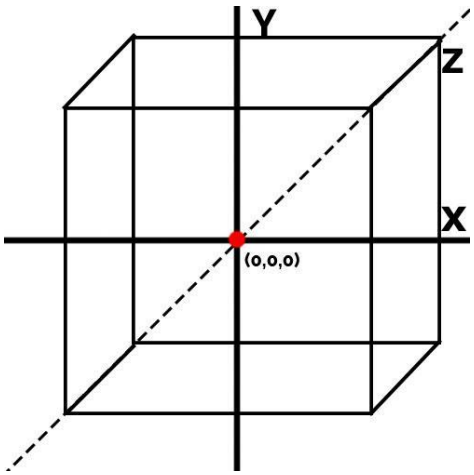

Рис. 2.2 - Тривимірна система координат

Джерело: [створено автором]

2.2.2. Глобальна та локальна системи координат

Глобальна система координат використовується для представлення загального ігрового світу і має початок у фіксованій точці, наприклад у центрі мапи або у початковій позиції гравця. Осі x, y та z використовуються для представлення трьох вимірів ігрового світу, а об'єкти розташовуються в межах цієї системи координат. (Рис. 2.3)

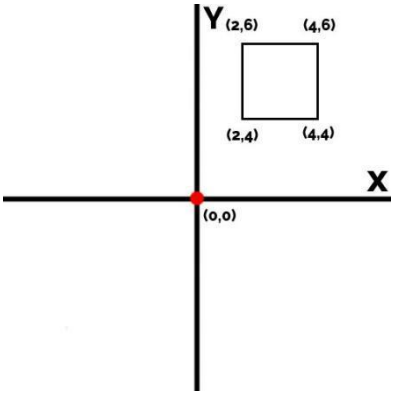

Рис. 2.3 - Глобальна система координат

Джерело: [створено автором]

Локальна система координат використовується для представлення окремих об'єктів у світі гри, наприклад, персонажів. Ця система координат зазвичай має початок у центрі об'єкта та використовується для представлення його положення та

орієнтації відносно глобальної системи координат. Використання локальної системи координат корисно при анімації та русі персонажів, оскільки рух та обертання обчислюються відносно його локальної системи координат. (Рис. 2.4)

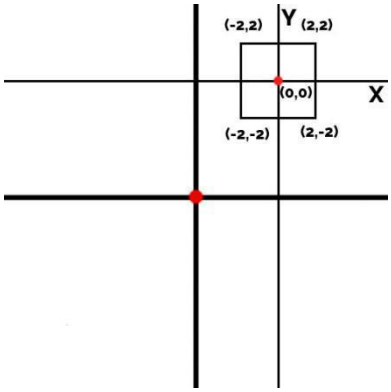

Рис. 2.4 - Локальна система координат

Джерело: [створено автором]

2.2.3. Матриці перетворень

В Unity для всіх перетворень векторів, є готові методи, але насправді все відбувається складніше за допомогою матриць перетворень. Оскільки кваліфікаційна робота передбачає створення мобільного застосунку в 3д просторі, то і матриці перетворень будуть розглянуті в тривимірній системі координат.

Отож, координати любої точки можна представити в вигляді вектору (x, y, z). Матриця перетворень дозволяє модифікувати значення вектору. Це потрібно, щоб ми могли змінювати та анімувати об’єкт, модифікувати його позицію, поворот чи розмір, оскільки всі вони є векторами в тривимірному просторі.

Матриця трансформацій виглядає як матриця розміром 3x3. (Рис. 2.5)

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Рис. 2.5 - Матриця перетворень 3x3

Джерело: [створено автором]

Кожна комбінація елементів матриці відповідає різним перетворенням. Щоб модифікувати початковий вектор, потрібно помножити його на матрицю. З цього можна відобразити формулу. (Рис. 2.6)

$$(\mathrm{x}' \quad y' \quad \mathrm{z}')=(\mathrm{x} \quad y \quad z)*\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Рис. 2.6 - Формула перетворення вектора

Джерело: [створено автором]

2.2.4. Перетворення масштабування

Модифікація масштабування є найлегшою операцією. Для того, щоб змінити розмір вектору, потрібно помножити початковий вектор (x, y, z) на діагональ матриці (a, e, i). (Рис. 2.7)

$$(\mathrm{x}' \quad y' \quad \mathrm{z}')=(\mathrm{x} \quad y \quad z)*\begin{pmatrix} a & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & i \end{pmatrix}$$

Рис. 2.7 - Формула модифікації розміру вектору

Джерело: [створено автором]

Щоб краще зрозуміти формулу, розберем її на прикладі.

Візьмем початковий вектор розміру (2, 2.5, 3) і помножем його на діагональ матриці (2, 2, 3). (Рис. 2.8)

$$(\mathrm{x}' \quad y' \quad \mathrm{z}')=(2 \quad 2.5 \quad 3)*\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} = (2 * 2 \quad 2.5 * 2 \quad 3 * 3) = (4 \quad 5 \quad 9)$$

Рис. 2.8 - Формула зміни розміру матриці

Джерело: [створено автором]

2.2.5. Трансформація переміщення

Трансформація переміщення зсуває початковий вектор, і є найбільш вживаною трансформацією. У випадку тривимірного простору, будем працювати з трьома значеннями матриці (g, h, i). (Рис. 2.9)

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ g & h & i \end{pmatrix}$$

Рис. 2.9 - Матриця переміщення

Джерело: [створено автором]

Розглянем приклад. Нам потрібно зсунути початковий вектор (5, 6, 4) на (10, -20, -10) одиниць по осі x, y, z відповідно. Для цього використовуємо формулу. (Рис. 2.10)

$$(x' \quad y' \quad z') = (5 \quad 6 \quad 4) * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & -20 & -10 \end{pmatrix} = (5 + 10 \quad 6 + (-20) \quad 4 + (-10)) =$$

(15 -14 -6)

Рис. 2.10 - Формула трансформації переміщення

Джерело: [створено автором]

2.2.6. Трансформація повороту

Трансформація повороту є найскладнішою з операцій, оскільки використовує тригонометрію та для обчислень повороту в тривимірному просторі потрібно три формули для кожної з осей координат (x, y, z).

Загальна формула обчислення повороту. (Рис. 2.11)

$$R = R(x) * R(y) * R(z)$$

Рис. 2.11 - Загальна формула повороту

Джерело: [створено автором]

Формула обчислення повороту по осі x. (Рис. 2.12)

$$R(x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & sin\,\theta & \cos\theta \end{pmatrix}$$

Рис. 2.12 - Обчислення повороту по осі x

Джерело: [створено автором]

Формула обчислення повороту по осі y. (Рис. 2.13)

$$R(y) = \begin{pmatrix} \cos\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & \cos\theta \end{pmatrix}$$

Рис. 2.13 - Обчислення повороту по осі y

Джерело: [створено автором]

Формула обчислення повороту по осі z. (Рис. 2.14)

$$R(z) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Рис. 2.14 - Обчислення повороту по осі z

Джерело: [створено автором]

Розглянемо тепер операцію на прикладі. Потрібно початковий вектор повороту (20, 45, 90) повернути на на (60, 90, 60) градусів відповідно.

Для цього необхідно знайти початкову (20, 45, 90) та наступну матрицю поворотів (60, 90, 60) і в кінці перемножити їх щоб отримати фінальний поворот. (Рис. 2.15)

$$R(start) = R(x) * R(y) * R(z) =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos 20 & -sin20 \\ 0 & sin20 & \cos 20 \end{pmatrix} * \begin{pmatrix} 0.7 & 0 & 0.7 \\ 0 & 1 & 0 \\ 0.7 & 0 & 0.7 \end{pmatrix} * \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -0.3 & -0.9 \\ 1 & 0.1 & 0.2 \\ 0 & 0.9 & -0.3 \end{pmatrix}$$

Рис 2.15 - Обчислення стартового повороту

Джерело: [створено автором]

Аналогічний чином обчислюємо вектор (60, 90, 60). (Рис. 2.16)

$$R(next) = R(x) * R(y) * R(z) = \begin{pmatrix} 0.5 & -0.1 & -0.8 \\ 0.8 & 0.3 & -0.4 \\ 0 & -0.9 & 0.3 \end{pmatrix}$$

Рис 2.16 - Обчислення наступного повороту

Джерело: [створено автором]

Перемножим ці матриці і отримаєм фінальну матрицю перетворень. (Рис. 2.17)

$$\begin{pmatrix} 0 & -0.3 & -0.9 \\ 1 & 0.1 & 0.2 \\ 0 & 0.9 & -0.3 \end{pmatrix} * \begin{pmatrix} 0.5 & -0.1 & -0.8 \\ 0.8 & 0.3 & -0.4 \\ 0 & -0.9 & 0.3 \end{pmatrix} = \begin{pmatrix} -0.3 & 0.3 & -0.8 \\ 0.3 & 0.9 & 0.3 \\ 0.8 & -0.3 & 0.3 \end{pmatrix}$$

Рис. 2.17 - Фінальна матриця перетворень

Джерело: [створено автором]

З цього можна зробити висновок, що операції поворотів є більш складними і потребують більше часу для виконання, тому потрібно оптимізовувати код. У випадку, коли потрібно зробити поворот тільки по двох осях, краще використовувати двовимірні обчислення, оскільки там лише одна формула, одна операція.

2.3. Життєвий цикл Unity

Дуже важливо розуміти життєвий цикл в Unity, щоб уникати багів і створювати оптимізовані ігри. Розглянемо наступні, найбільш вживані події:

1. Awake - виклик за 1 кадр до старту сцени.
2. OnEnable - виклик при увімкненні об’єкта.
3. Start - виклик при старті сени.
4. Update - виклик кожний кадр, тобто якщо в грі 100fps - метод буде викликаний 100 разів в секунду.
5. LateUpdate - як Update, але в кінці кадру, корисний для руху камери за гравцем.
6. FixedUpdate - виклик кожних n секунд, де n - стале значення в налаштуваннях, зазвичай 0.2 секунди. Корисний для фізичних операцій.

7. OnDisable - виклик при вимкненні об’єкта.

8. OnDestroy - виклик при знищенні об’єкта.

Можна відобразити це діаграмою. (Рис. 2.18)

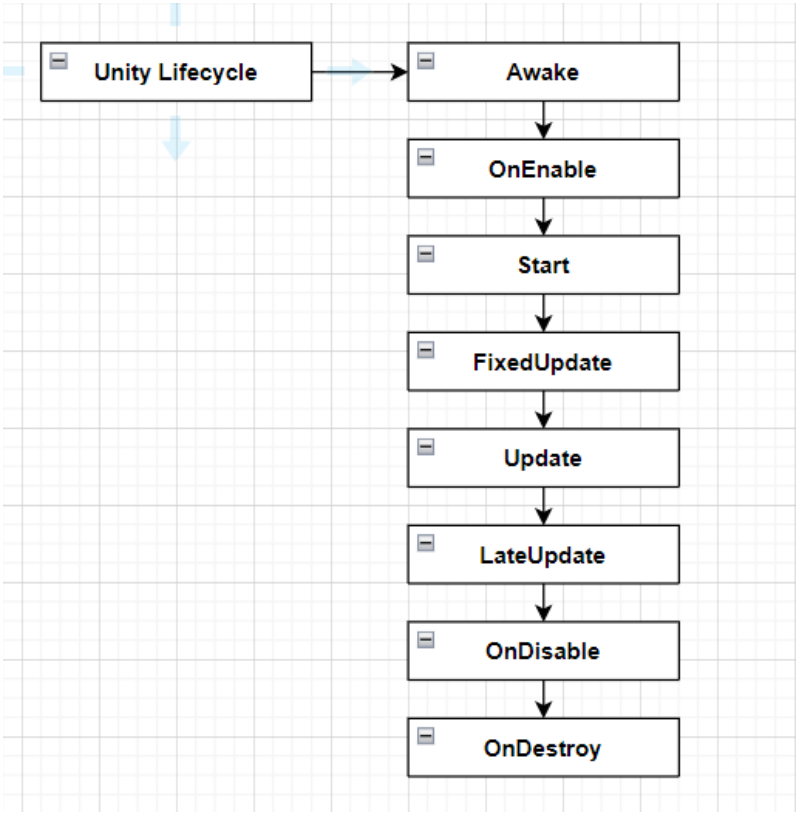

Рис. 2.18 - Життєвий цикл Unity

Джерело: [створено автором]

Дуже важливим, при виконанні кваліфікаційної роботи, розуміти порядок виконання скриптів, і сформувати архітектуру мобільного застосунку.

Часто використовується метод Update для оновлення значень гри. Зазвичай в таких іграх все погано з оптимізацією. Потрібно якнайменше використовувати метод Update, натомість використовувати події.

Проте, події і двигун Unity не дуже добре разом працюють, через невизначеність порядку виконання скриптів. Проблема в тому, що не можна визначити який з методів життєвого циклу двигуна для кожного об’єку буде

викликатись першим. В такому випадку ми просто отримаєм помилку і гра перестане працювати. (Рис. 2.19)

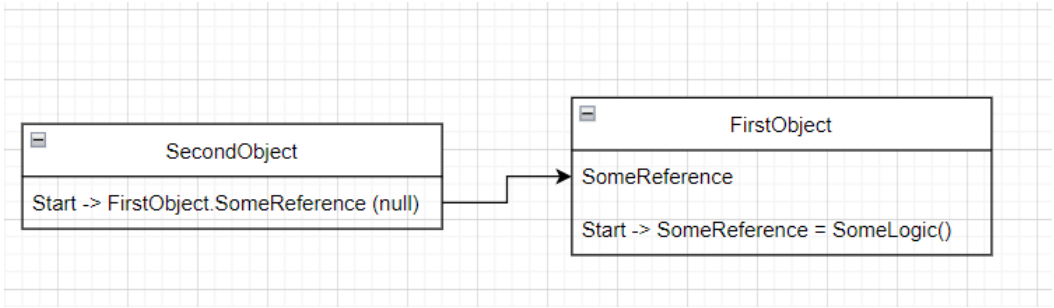

Рис. 2.19 - Невизначеність порядку виклику методу Start

Джерело: [створено автором]

Як зазначалось раніше, перед методом Start є ще один метод - Awake. Можна спочатку зробити обчислення в ньому, тоді другий об’єкт точно отримає дані. Але це теж поганий варіант, оскільки Awake викликається за кадр до старту сцени, і якщо перевантажити цей метод, то сцена буде довго завантажуватись. І де гарантія, що перший об’єкт не буде залежати від ще одного об’єкту, який також робить обчислення в методі Awake? Потрібно добре продумати архітектуру і вирішити це питання щоб уникнути багів.

2.4. Базові операції Blender

Для виконання кваліфікаційної роботи потрібно буде створити багато моделей в Blender, тому необхідно знати базові операції при роботі з моделями.

Додавання та видалення об'єктів: Щоб додати об'єкт, потрібно натиснути *Shift + A* і вибрати тип об'єкта, який потрібно додати. Щоб видалити об'єкт, потрібно його вибрати і натиснути клавішу *X*.

Режим редагування: Щоб редагувати вершини, ребра і грані об'єкта, потрібно вибрати об'єкт і натиснути клавішу *Tab*, щоб увійти в режим редагування.

Екструдування: Щоб створити нову геометрію шляхом розширення існуючих граней або ребер, потрібно вибрати геометрію, яку потрібно екструдувати, і натиснути клавішу *E*.

Масштабування: Щоб змінити розмір об'єкта або вибраної геометрії, потрібно натиснути клавішу *S*.

Обертання: Щоб повернути об'єкт або вибрану геометрію, потрібно натиснути клавішу *R*.

Додавання та зміна матеріалів: Щоб додати новий матеріал до об'єкта, потрібно перейти на вкладку «Матеріали» на панелі властивостей і натиснути кнопку «Додати матеріал». Щоб змінити матеріал, потрібно вибрати його і змінити властивості на вкладці матеріалів.

Висновки до розділу 2

У цьому розділі була проаналізоване інформаційне та програмне забезпечення. Був описаний необхідний функціонал програм, що були вибрані для реалізації кваліфікаційної роботи у першому розділі, а також було проаналізовано математичне забезпечення, необхідне для реалізації механік гри.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Засобами розробки кваліфікаційної роботи є програми: Unity, Blender, Adobe Photoshop, Jetbrains Rider, а також бібліотеки та фреймворки: UniRx, Zenject, Addressables, DotWeen.

3.2. Опис програмної реалізації

3.2.1. Реалізація архітектури застосунку

Архітектура в мобільних є дуже важливою, тому потрібно обдумати її з самого початку. Вона повинна легко розширюватись та вирішувати проблеми зазначені в розділі 2.3. Для її реалізації будем використовувати фреймворки Zenject та UniRx.

Zenject - це фреймворк для інверсії керування та управління залежностями для Unity, розроблений для спрощення створення інтерфейсів та залежностей між об'єктами. Він дає змогу декларувати залежності на рівні класу та ін'єктувати їх у створювані об'єкти в автоматизований спосіб.

Інверсія керування - це абстрактний принцип, набір рекомендацій для написання слабко пов'язаного коду. Суть якого в тому, що кожен компонент системи має бути якомога більш ізольованим від інших, не покладаючись у своїй роботі на деталі конкретної реалізації інших компонентів.

Zenject надає широкі можливості для налаштування та конфігурування залежностей, які можна здійснювати через кастомні фабрики, провайдери та інші механізми. Також, він підтримує ієрархію контейнерів, що дозволяє створювати багатомодульні додатки з ізольованими контекстами залежностей [4]. (Рис. 3.1)

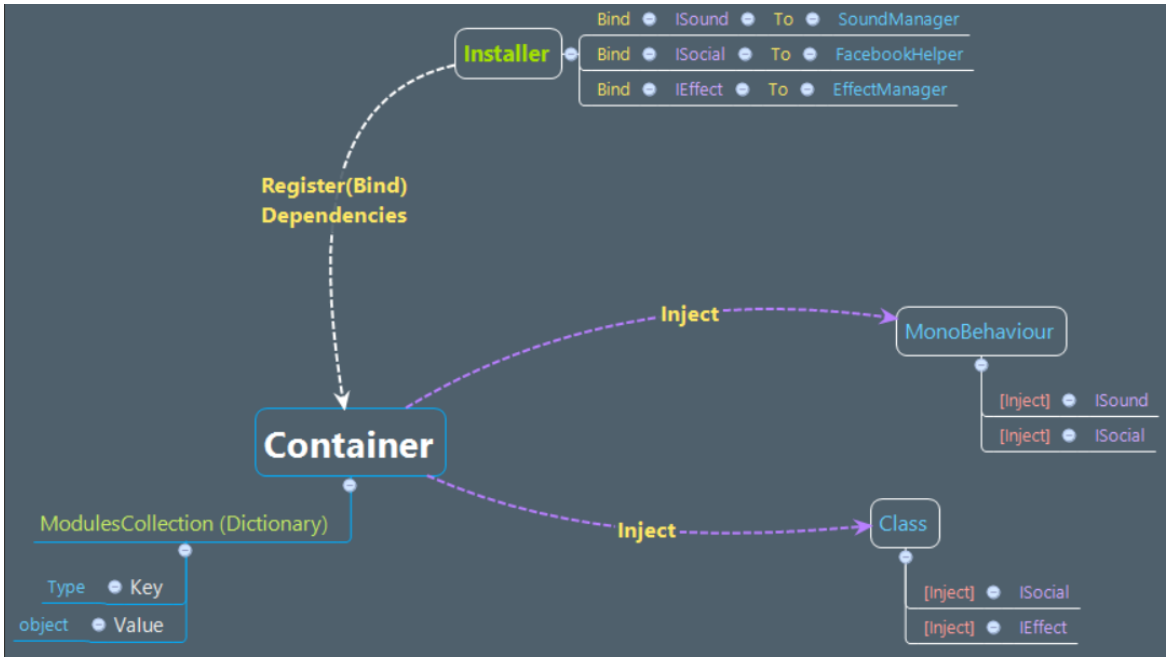

Рис. 3.1 - Функціонал та робота Zenject

Джерело: [створено автором]

UniRx - це реактивна бібліотека для Unity, яка дозволяє легко і ефективно розробляти асинхронний код з використанням парадигми програмування, відомої як "реактивне програмування". Ця бібліотека дозволяє працювати з подіями, потоками даних та іншими асинхронними процесами з більшою ефективністю та зручністю, ніж традиційний підхід до програмування.

UniRx підтримує багато різних типів потоків, таких як Observable, Subject, AsyncSubject та ReplaySubject, які дозволяють споживачам даних отримувати дані асинхронно та реагувати на зміни в реальному часі.

Observable - потік даних, за яким можуть спостерігати.

Subject - потік даних, за яким можна і спостерігати, і опублікувати зміни.

AsyncSubject - потік даних, який публікує лише останнє значення.

ReplaySubject - потік даних, який публікує всі значення для спостерігачів ігноруючи момент їхньої підписки.

Крім того, UniRx дозволяє зручно працювати з корутинами та іншими асинхронними процесами, що робить його відмінним вибором для розробки ігор та інтерактивних додатків в Unity [5]. (Рис. 3.2)

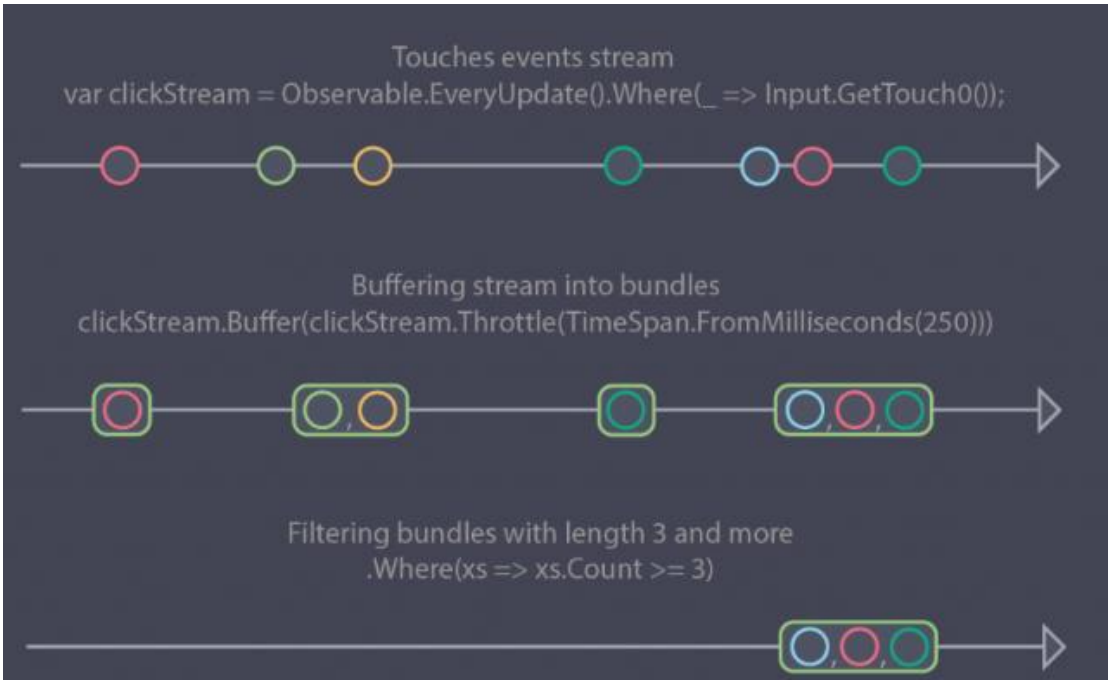

Рис. 3.2 - Потоки даних в UniRx

Джерело: [створено автором]

Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Також, для архітектури, анімацій буде використовуватись фреймворк DotWeen. **Dotween** - це плагін для Unity, який дозволяє легко створювати різні анімації і ефекти для ігор та додатків. Він пропонує багато різноманітних функцій та інструментів для налаштування та управління анімаціями, таких як різні криві експоненційної інтерполяції, затримки, розмиття, зміна розміру, повороти та багато іншого.

Один з головних переваг Dotween полягає в його простоті використання. Його API дуже інтуїтивно зрозумілий та простий, що дозволяє швидко розуміти, як

працювати з різними функціями. Він також працює досить швидко, що є важливим для багатьох ігор та додатків [6]. (Рис. 3.3)

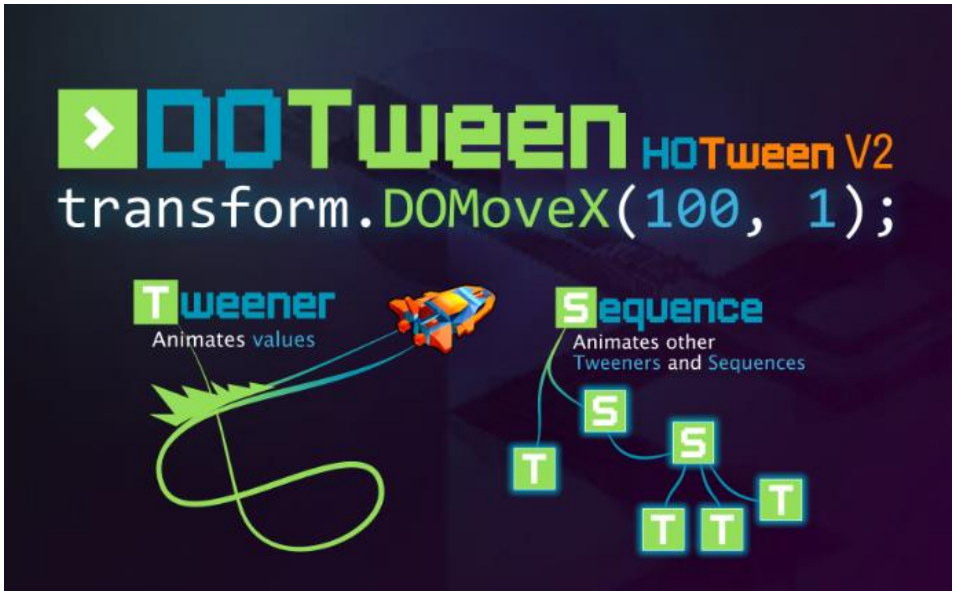

Рис. 3.3 - Приклад роботи DotWeen

Джерело: [https://assetstore.unity.com/packages/tools/animation/dotween-hotween-v2-27676]

Тепер, маючи всі необхідні фреймворки, можна реалізовувати архітектуру.

Кожна гра проходить певну ініціалізацію під час її запуску. Це може бути загрузка збережень, авторизація, довантаження необхідних даних. Логічно, що для такого функціоналу потрібно використати паттерн State Machine.

StateMachine - шаблон проєктування, що реалізує скінченний автомат в об’єктно-орієнтованому програмуванні. Він реалізується шляхом створення для кожного стану скінченного автомата класу - спадкоємця інтерфейсу та дозволяє об'єктові варіювати свою поведінку залежно від внутрішнього стану. (Рис. 3.4)

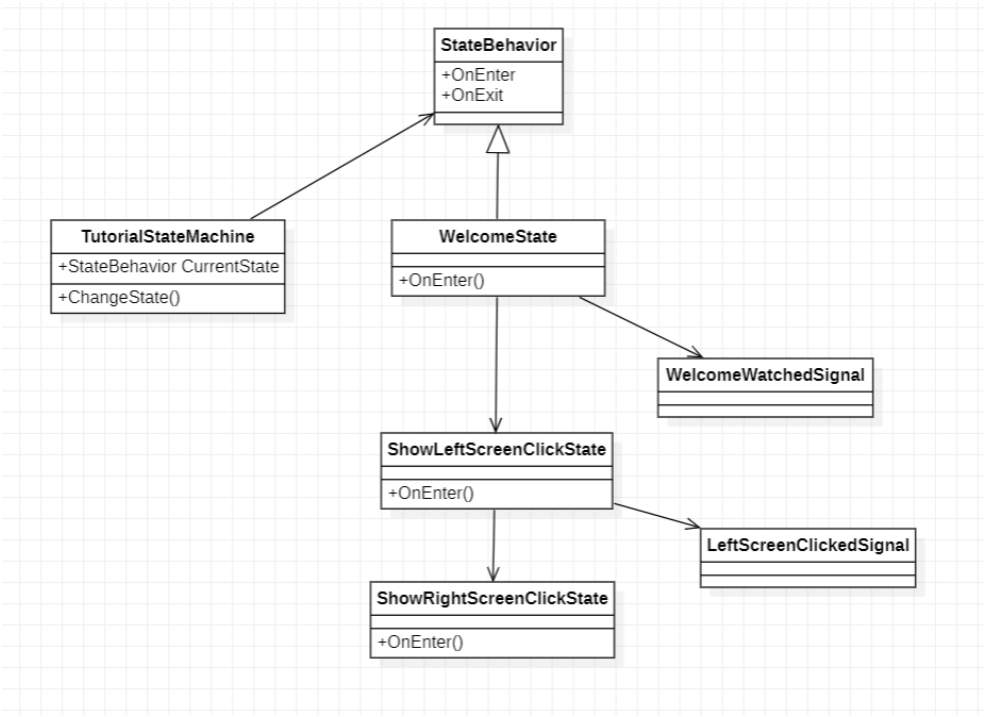

Рис. 3.4 - Приклад реалізації патерну StateMachine

Джерело: [створено автором]

Проте, для ініціалізації даних він не підійде, але буде використаний для ініціалізації сцени рівня в грі. Проблема в тому, що така реалізація потребує кожний раз запуску гри з якоїсь стартової сцени, що є неважливим самій грі, але ускладнює розробку в сердовищі Unity. Тому, для вирішення такої проблеми буде створений клас PlayerAccount, в якому будуть зберігатись всі дані гри які потрібно зберігати і зчитувати. Цей PlayerAccount буде декларований на рівні всього контексту гри, і буде ініціалізуватися після всіх декларацій цього контексту. А оскільки, кожний контекст сцени намагатиметься викликати контекст гри, якщо він ще не був ініціалізованим, то з любої сцени можна буде отримати всі дані. (Рис. 3.5)

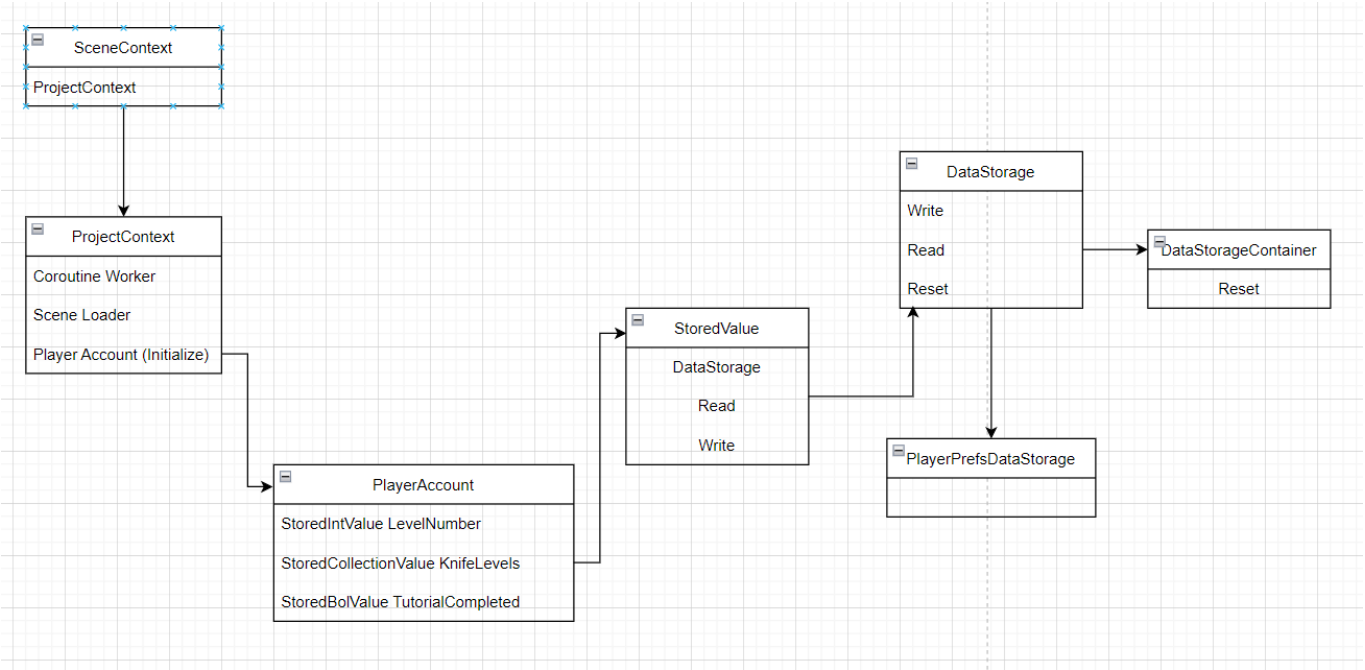

Рис. 3.5 - Реалізація ініціалізації даних

Джерело: [створено автором]

Як видно з рисунка 3.5, для зберігання використовується PlayerPrefsDataStorage. Цей контейнер буде використовувати клас Unity PlayerPrefs для зберігання та зчитування даних. Клас DataStorage є абстрактним, тому також є можливість замінити реалізацію на json, тощо.

PlayerPrefs - це простий спосіб зберігання та отримання даних про уподобання гравців або стан гри в іграх Unity. Цей інструмент надає базове сховище ключ-значення в реєстрі, де можна зберігати та завантажувати такі дані, як ім'я гравця, рахунок гри, налаштування звуку тощо. PlayerPrefs простий у використанні і доступний з будь-якого скрипту. Можна зберігати дані різних типів, включаючи цілі числа, числа з плаваючою комою та рядки, призначивши кожному значенню ключ. Для подальшого отримання значень використовуються саме ці ключі.

Тепер, після ініціалізації даних, можна загрузити сцену гри. Для сцени гри також потрібно зробити архітектуру. Тут вже буде використовуватись патерн StateMachine. Для сцени рівня потрібні наступні стани:

1. TutorialState

2. TutorialMergeState
3. TutorialBuyState
4. TutorialScrollState
5. TutorialDropState
6. CreateLevelState
7. StartLevelState
8. LevelState
9. WinState
10. LoseState

Зобразимо це на діаграмі. (Рис. 3.6)

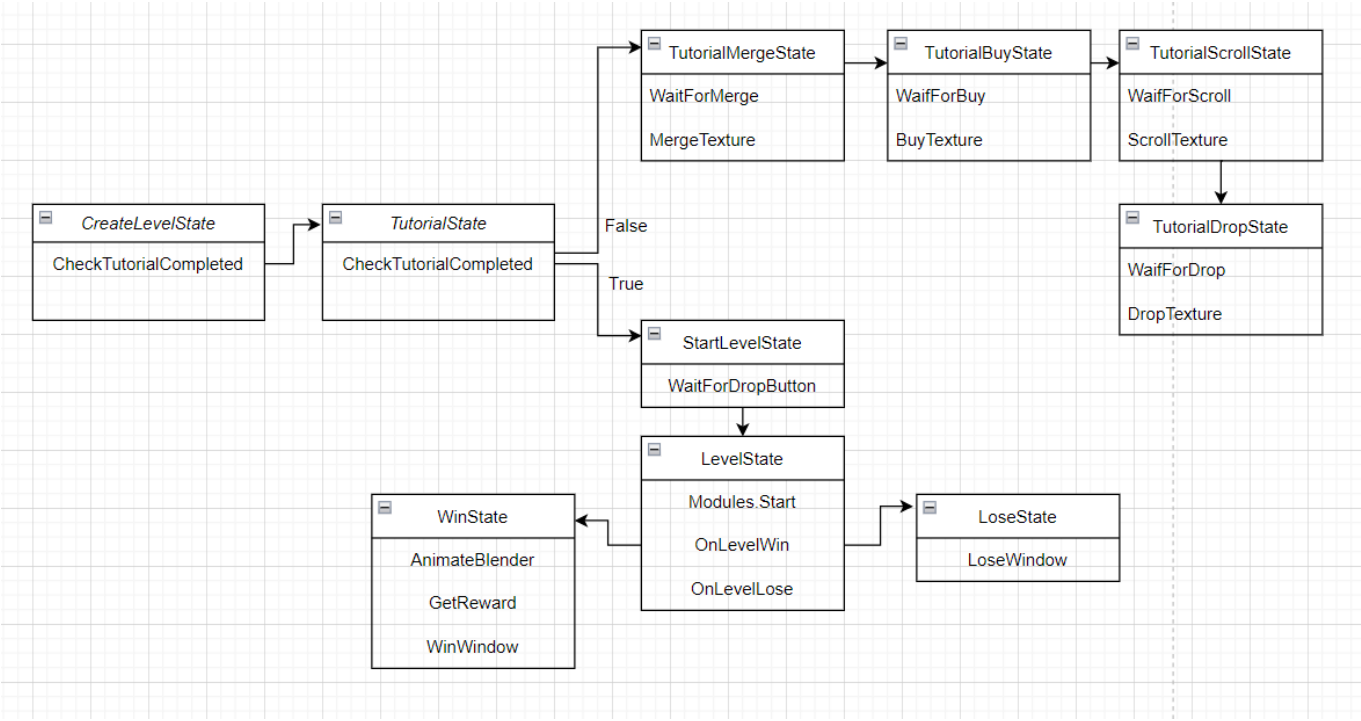

Рис. 3.6 - Архітектура рівня

Джерело: [створено автором]

Для решти модулів, скриптів, залежностей застосунку буде використовуватись наступна архітектура залежностей. (Рис. 3.7)

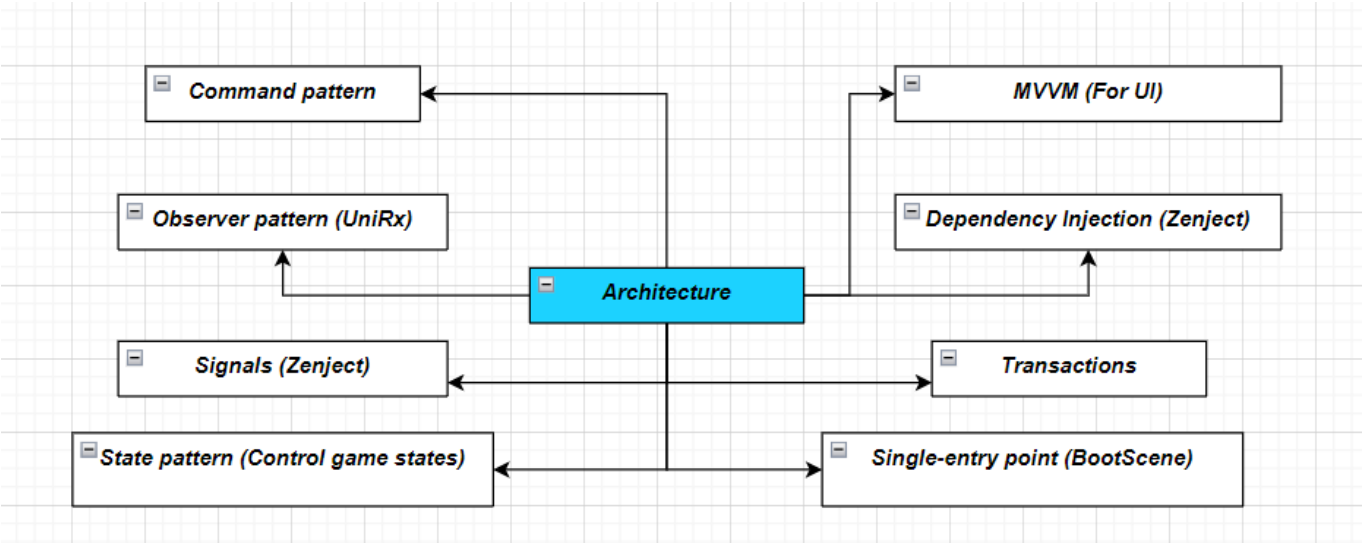

Рис. 3.7 - Архітектура залежностей

Джерело: [створено автором]

Для модулів гри будуть використовуватись сервіси, якими вже будуть користуватись ігрові об’єкти за потреби.

В більшості випадків це буде виглядати так: (Рис. 3.8)

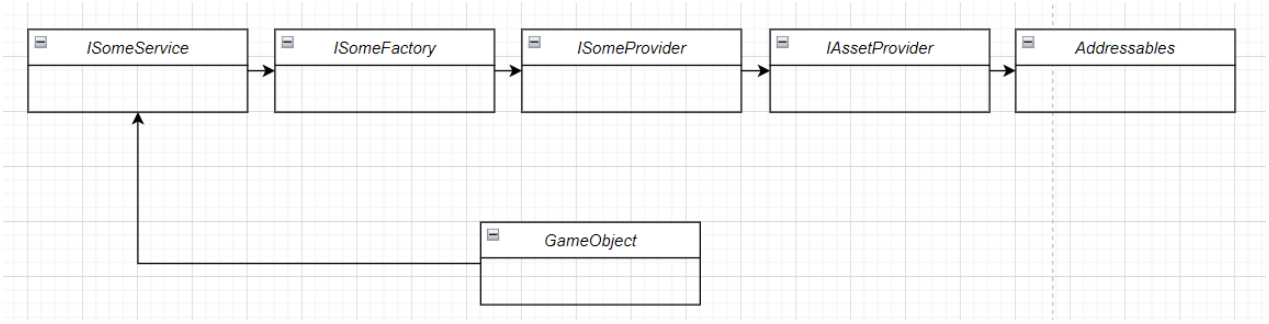

Рис. 3.8 - Діаграма відношення між сутностями

Джерело: [створено автором]

3.2.2. Створення моделей для гри

Для реалізації кваліфікаційної роботи потрібно багато моделей, які будуть створені використовуючи Blender.

3.2.3. Створення моделей ножів

Логіка гри полягає в тому, щоб об’єднувати ножі та отримувати ніж вищого рівня. В залежності від рівня ножа, модель має відрізнятись, щоб користувач бачив прогресс. Для цього необхідно створити щонайменше 10 таких моделей.

Створюємо моделі ножів, використовуючи базові операції роботи з моделями. (Рис. 3.9)

Рис. 3.9 - Створені ножі в Blender

Джерело: [створено автором]

3.2.4. Створення моделей ножів

Таким самим чином створюємо моделі для фруктів. (Рис. 3.10)

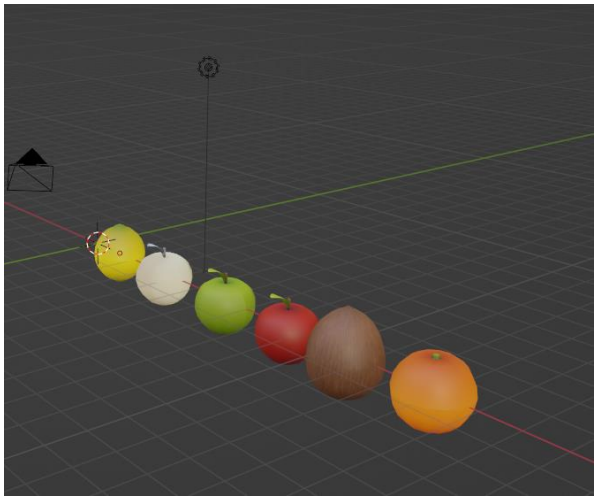

Рис. 3.10 - Створені фрукти в Blender

Джерело: [створено автором]

Для фруктів необхідно виконати подвійну роботу, оскільки необхідною буде анімація їх розрізання. (Рис. 3.11)

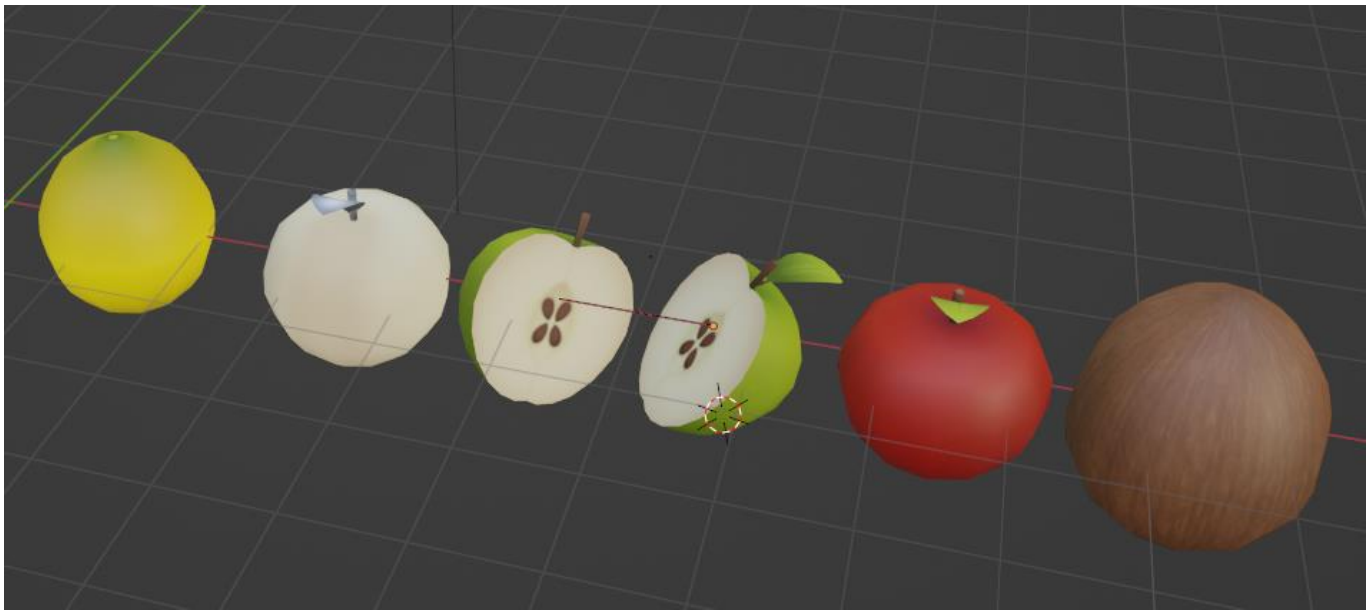

Рис. 3.11 - Розрізаний фрукт яблука

Джерело: [створено автором]

3.2.5. Створення моделей блендеру та подарунку

Також, потрібен блендер, який буде заповнюватись. Створюємо модель блендеру. (Рис. 3.12)

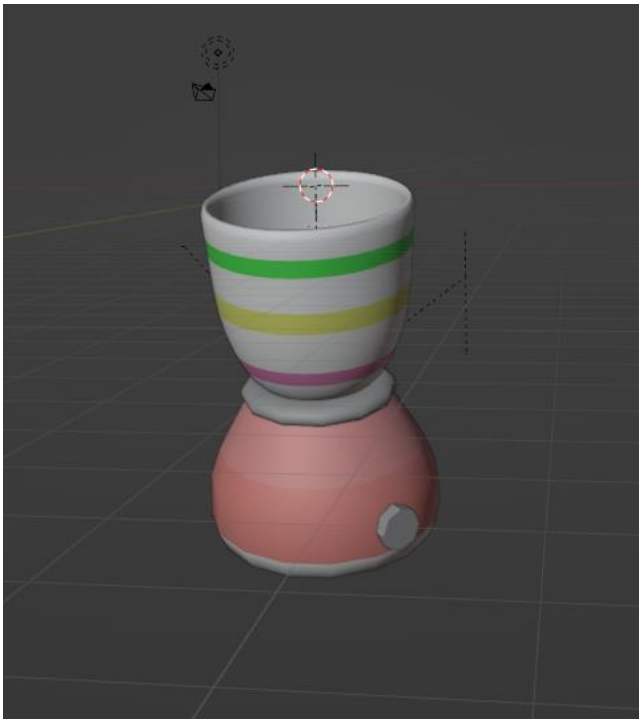

Рис. 3.12 - Модель блендеру

Джерело: [створено автором]

Логіка гри повинна бути логічно побудована. На блендері можна помітити три лінії, вони зроблені для того, щоб показати ступінь заповнення соку, і отримати відповідний подарунок який знаходиться навпроти цієї лінії.

Створимо модель такого подарунку. (Рис. 3.13)

Рис. 3.13 - Модель подарунку

Джерело: [створено автором]

Зараз на ньому немає матеріалу, добавим його в Unity, щоб відобразити ступінь подарунку. (Рис. 3.14)

Рис - 3.14. Префаби моделі подарунку

Джерело: [створено автором]

3.2.6. Реалізація кольорової гами

Успішні ігри в маркетплейсах мають крутий дизайн і добре продуману кольорову гаму, щоб користувачі легко привикали до гри та відчували її особливість. Тому для реалізації кваліфікаційної роботи потрібно розробити таку кольорову гаму.

Для цього, в одній гамі потрібно зробити:

1. Дизайн рівня
2. Дизайн інтерфейсу
3. Дизайн іконки

Щоб досягти такого ефекту, потрібно буде використати Toon Shader.

Toon Shading - це стиль рендерингу, призначений для імітації 2D-плоских поверхонь на 3D-поверхнях. Якщо використати його в застосунку для всіх моделей, вони всі будуть виглядати в одному приємному стилі. (Рис. 3.15)

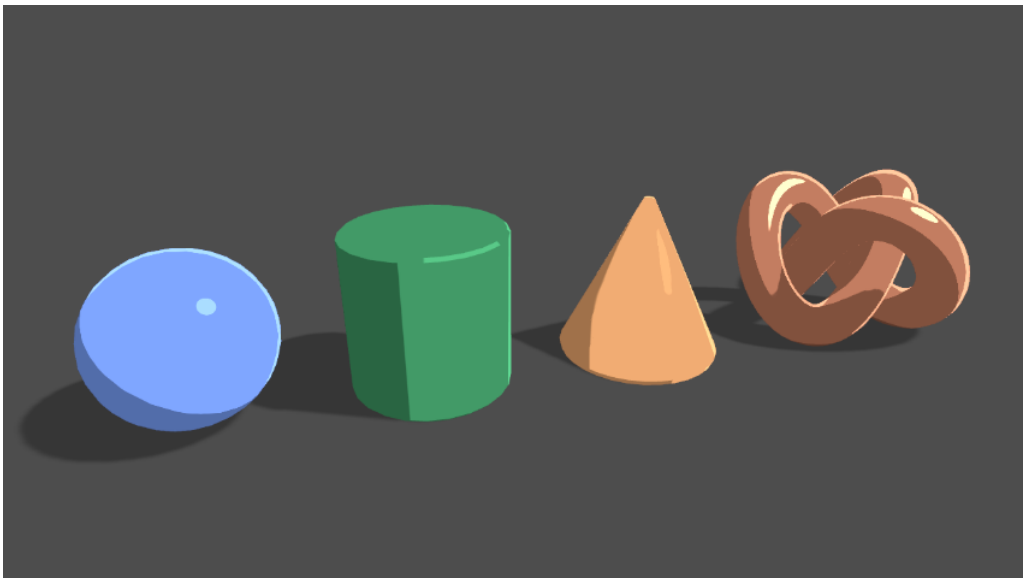

Рис. 3.15 - Приклад Toon Shader

Джерело: [створено автором]

Змінюєм матеріали на ножах і переносим їх на сцену. Отримуємо такий результат: (Рис. 3.16)

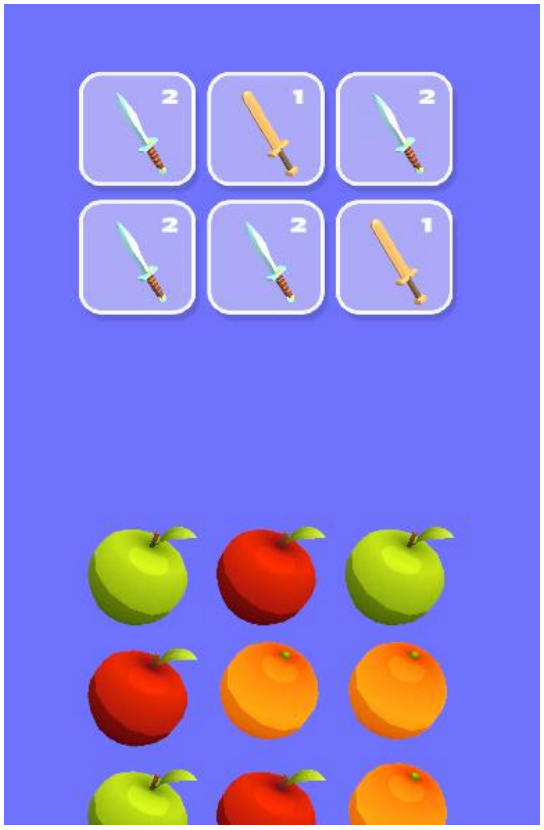

Рис. 3.16 - Вигляд моделей в грі з використанням Toon Shader

Джерело: [створено автором]

Добавимо інтерфейс для застосунку. Потрібні кнопки для купівлі нового ножа та для старту рівня, а також кнопка налаштувань, інтерфейс для монет і цілі рівня. (Рис. 3.17)

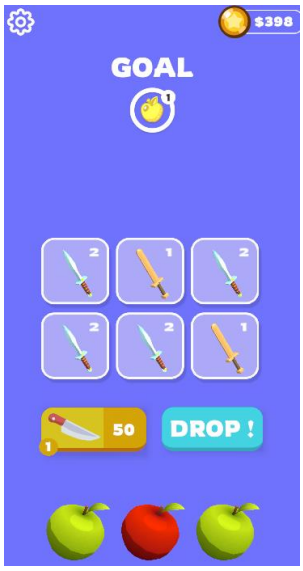

Рис. 3.17 - Вигляд рівня з інтерфейсом

Джерело: [створено автором]

Також необхідно змінити задній фон, зробити його простим та приємним, а також більш контрастним. (Рис. 3.18)

Рис. 3.18 - Кінцевий вигляд гри

Джерело: [створено автором]

3.2.6. Реалізація адаптації застосунку

Адаптивний інтерфейс - це здатність користувацького інтерфейсу адаптувати свій макет та елементи залежно від розміру екрану, співвідношення сторін або орієнтації пристрою. Це важливо для створення послідовного і зручного інтерфейсу на різних пристроях і платформах.

Для реалізації адаптивного інтерфейсу використовується компонент CanvasScaler. Розширенням по замовчуванню є 1080x1920. (Рис. 3.19)

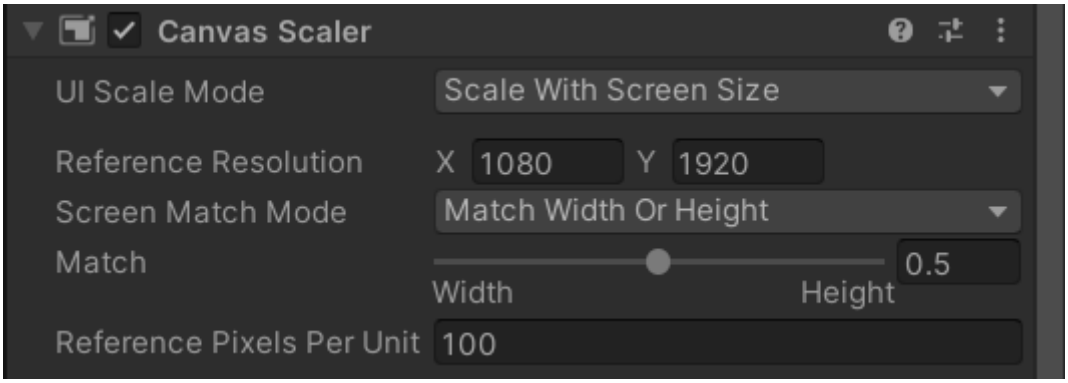

Рис. 3.19 - Компонент CanvasScaler

Джерело: [створено автором]

Для реалізації адаптивності 3d середовища, використовують спеціальний скрипт для ортографічної камери, який змінює поле Size, тим самим віддаляючи або приближуючи об’єкти до камери. Таким чином, знаючи стандартне розширення і розширення девайсу, можна знайти різницю між ними і пропорційно змінити поле Size камери. (Рис. 3.20)

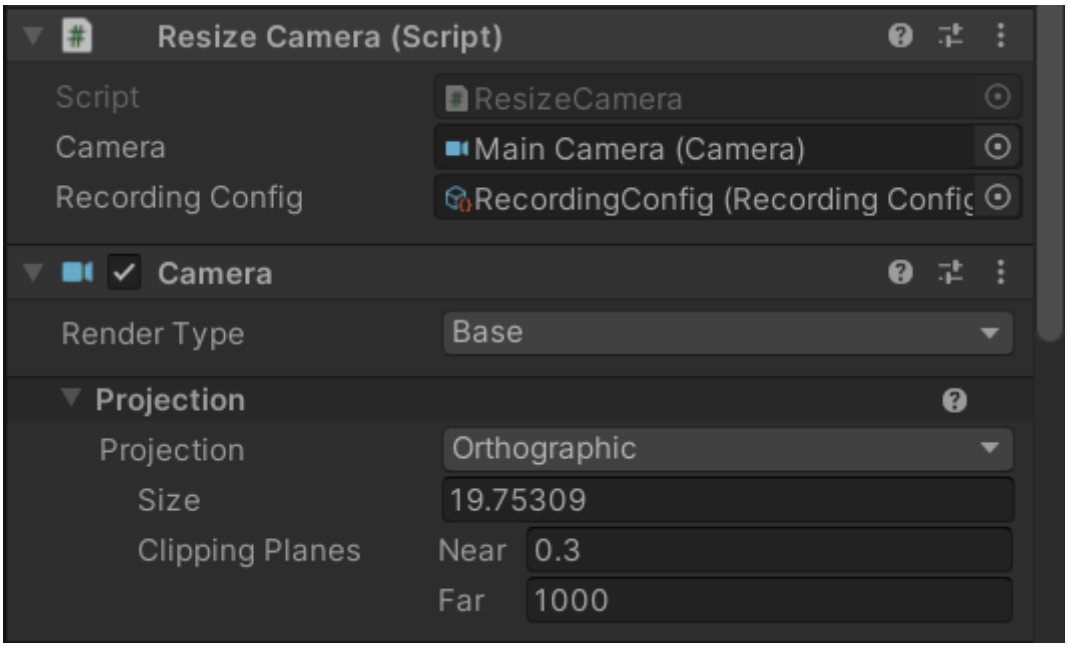

Рис. 3.20 - Скрипт для адаптиву камери

Джерело: [створено автором]

3.2.7. Реалізація механіки гри

Основною механікою гри є об'єднання ножів та розрізання фруктів. Для того, щоб об’єднувати ножі, необхідно знати координати дотику, а також момент, коли дотик відбувається. Для знаходження координатів дотику скористаємось методом Camera.ScreenToWorldPoint(Vector3 worldPoint), який конвертує точку дотику по екрані в ігровий світ Unity. Для знаходження моменту початку і кінця дотику будем використовувати подію OnMouseUp() і OnMouseDown(), цей API надає базовий клас Unity MonoBehaviour. Також потрібний метод для об’єднання ножів, який знищить два об’єкти зі сцени, і створить новий відповідно до рівня. Варто зазначити, що для такої цілі нам необхідні 10 префабів самих моделей, без скриптів на них, і лише один префаб з функціоналом. (Рис. 3.21)

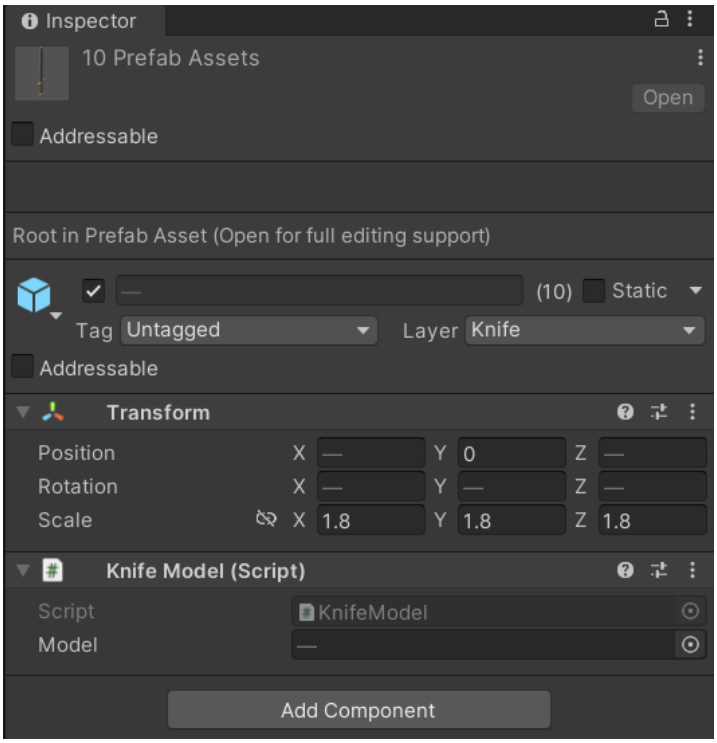

Рис. 3.21 - Пусті префаби ножів

Ось так виглядатиме об’єкт з функціоналом. (Рис. 3.22)

Джерело: [створено автором]

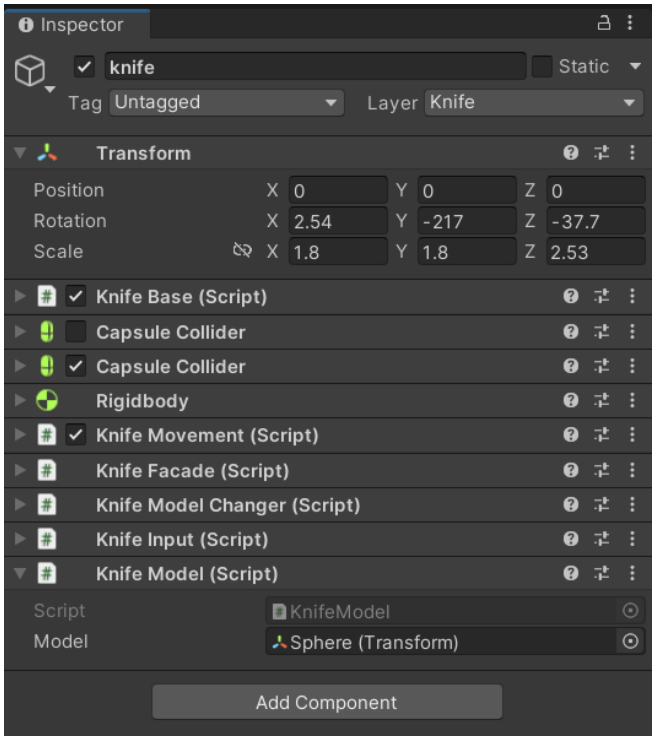

Рис. 3.22 - Скрипти для ножів на одному префабі

Джерело: [створено автором]

Така реалізація дає змогу підміняти дочірній елемент, лише його вигляд, залишаючи основний функціонал без змін. А також дає можливість створити конфіг для легкої зміни вигляду ножа відносно його рівня. (Рис. 3.23)

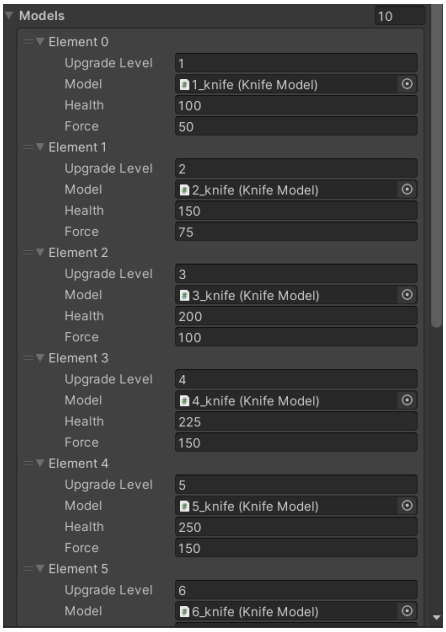

Рис. 3.23 - Конфіг для ножів

Джерело: [створено автором]

Як видно з конфігу, у нас є можливість не лише міняти вигляд ножа, але й його характеристики, такі як сила і життя. Сила - це урон який дається фруктові при попаданні.

Такий самий конфіг реалізуєм для фруктів. (Рис. 3.24)

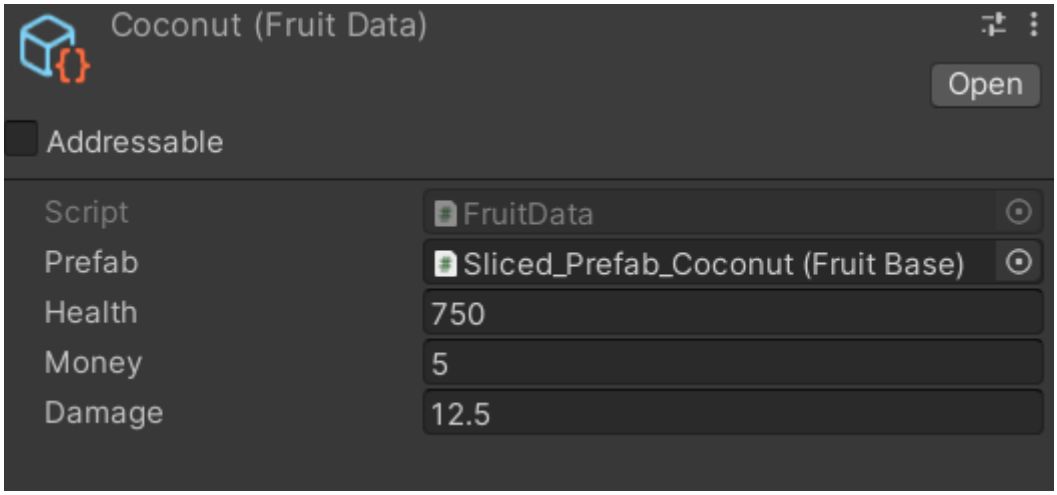

Рис. 3.24 - Конфіг для фрукта Coconut

Джерело: [створено автором]

При кліку по кнопці Drop, ножі полетять вниз, використовуючи фізичний компонент Rigidbody, а також метод фреймворку Dotween під назвою DoRotate, при колізії з фруктом, ніж відбивається, перекручується, і знову летить вниз. (Рис. 3.25)

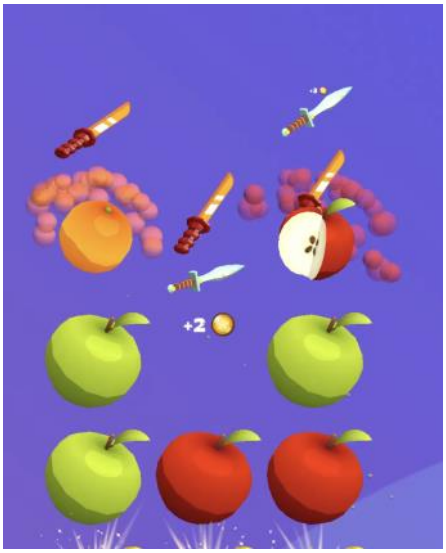

Рис. 3.25 - Падіння ножів на фрукти

Джерело: [створено автором]

Логіка розрізання має два варіанти:

1. Ніж розрізав з першого разу, фрукт полетів в блендер, і ніж, не відштовхуючись далі полетів вниз.

2. Ніж з першого разу не розрізав, відштовхнувся, і знову полетів на фрукт.

З таким функціоналом гра відчувається приємніше.

3.2.8. Прогрессія гри

Гра має 120 рівнів, які повинні ускладнюватись. Для цього реалізовані 6 типів фруктів та 10 рівнів ножів. З кожним наступним рівнем кількість рядків, що потрібно знищити, буде рости, також будуть добавлятись нові колонки, максимум їх 5, і кількість золотих яблук, яких потрібно розрізати. Після 50-го рівня, колонки та рядки будуть перетасовуватись, щоб рівні відрізнялись. (Рис. 3.26)

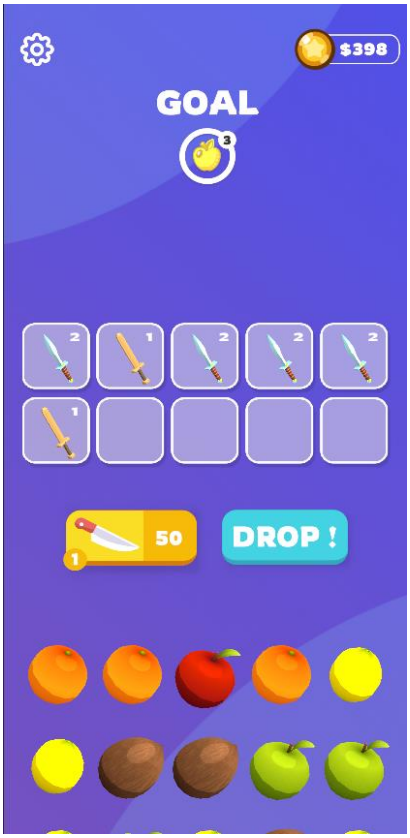

Рис. 3.26 - Прогрессія рівнів

Джерело: [створено автором]

3.2.9. Генератор рівнів

Для створення рівнів буде реалізований спеціальний генератор, який працюватиме лише в редакторі і не буде включений в білд. Генератор буде брати попередній рівень, добавляти до нього новий рядок з рандомно вибраними фруктами, також вставляючи в середину та на кінець колонки вибірковим чином подарунки. Як вже згадувалось, після 50-го рівня, рівні будуть лише перетасовуватись.

Зберігатись дані будуть в ScriptableObject.

ScriptableObject - спеціальний об’єкт для Unity, що живе весь час, поки запущений процесс, для зберігання даних, конфігів.

Для цього об’єкта буде створений спеціальний вигляд, щоб бачити різницю між рівнями. В вигляді буде автоматично виводитись статистика про рівень. (Рис. 3.27)

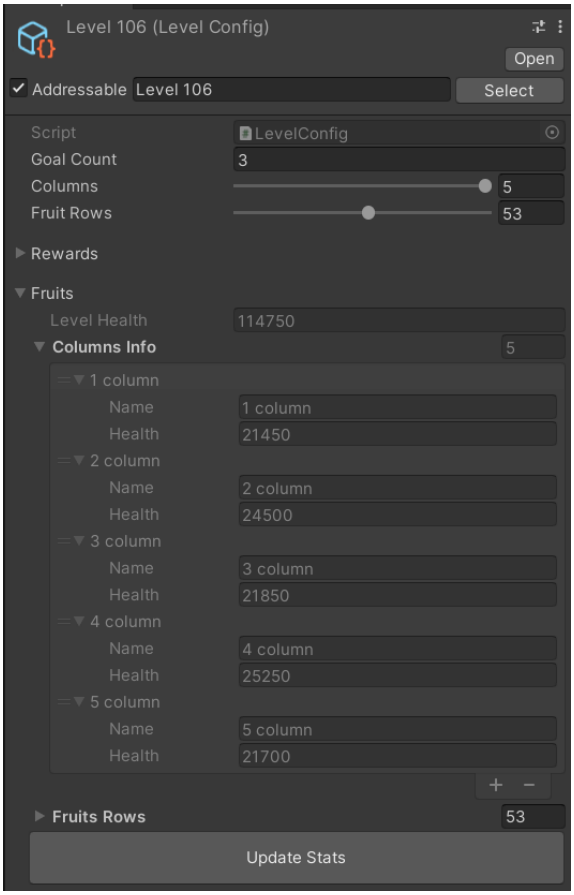

Рис. 3.27 - Вигляд конфігурації рівня.

Джерело: [створено автором]

Генератор рівнів буде автоматично створювати такі файли, щоб потім мати змогу їх відредагувати за потреби. (Рис. 3.28)

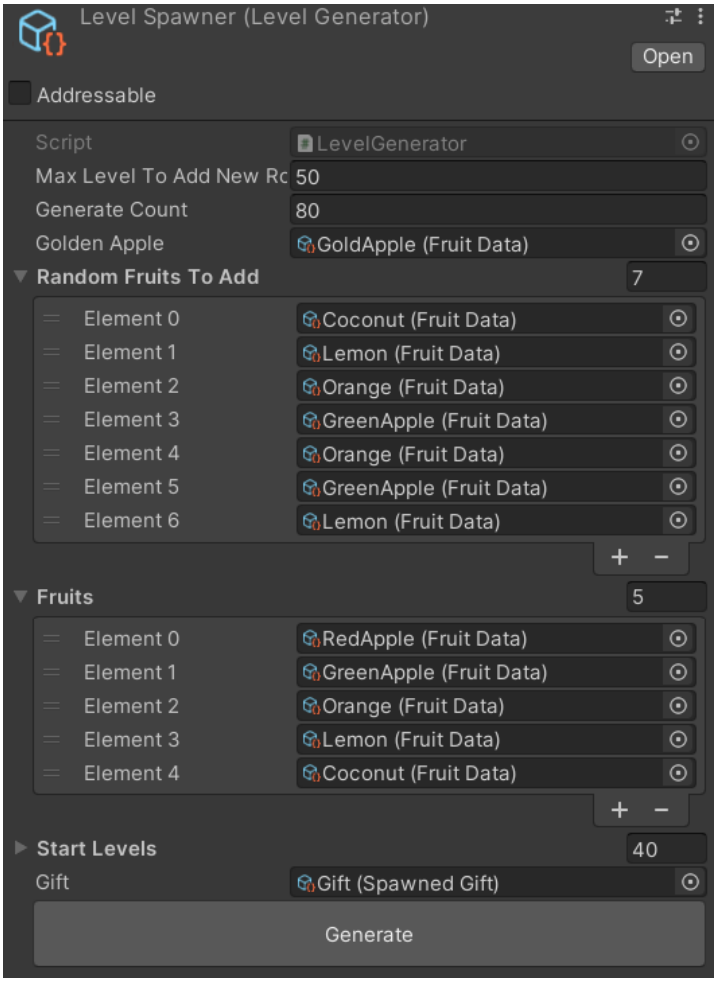

Рис. 3.28 - Генератор рівнів.

Джерело: [створено автором]

3.2.10. Туторіал

Туторіал буде з’являтись на першому рівні. У ньому буде чотири стани:

1. Мердж ножів
2. Клік по кнопці купівлі ножа
3. Скролл вниз рівня, щоб побачити всі елемента
4. Клік по кнопці старту

Після завершення останнього, туторіал буде пройдений і більше не буде з’являтись. (Рис. 3.29)

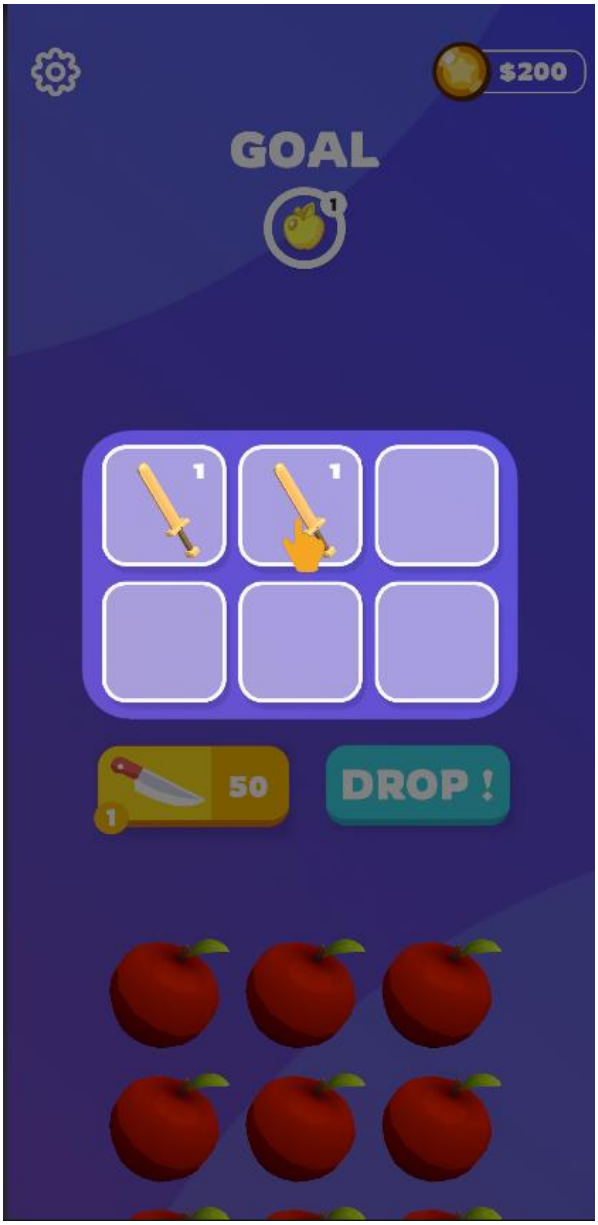

Рис. 3.29 - Вигляд туторіалу.

Джерело: [створено автором]

Для реалізації такого вигляду, буде використаний Adobe Photoshop. У ньому буде загружений скріншот з гри, потім ми виріжимо потрібну область і зменшимо прозорість. Також, для адаптивності такої текстури, потрібно, щоб її розмір був мінімум 2500x2500 пікселів. (Рис. 3.30)

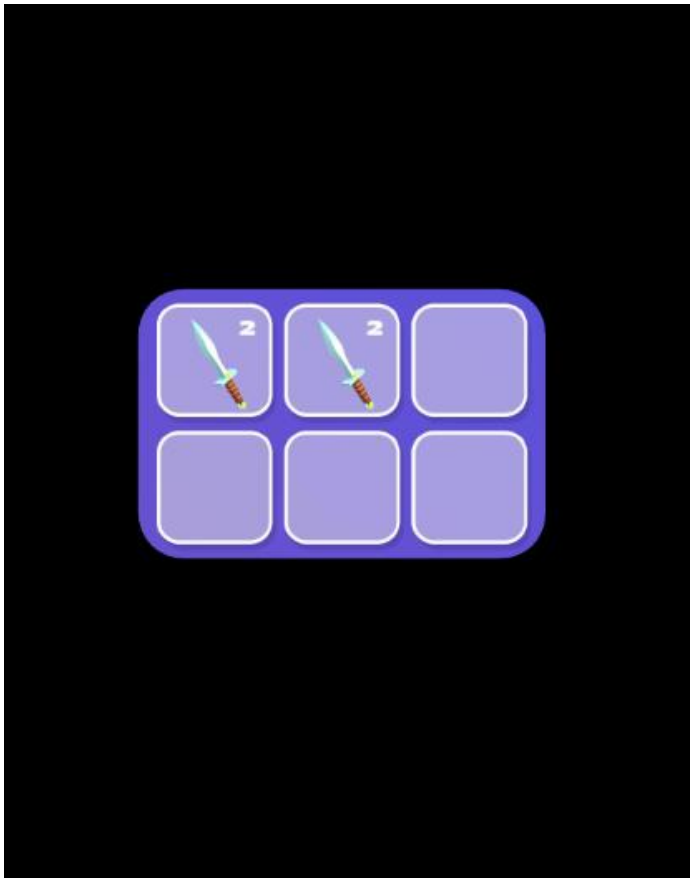

Рис. 3.30 - Реалізації текстури для туторіалу.

Джерело: [створено автором]

3.2.11. Звуки

У грі буде багато звуків. Для цього буде розроблений спеціальний клас - сервіс, щоб відігравати їх з любого місця в коді, та для легкого добавлення нових. Слід зазначити, що для звуку розрізання фруктів потрібна буде оптимізація, оскільки таких звуків на 50-му рівні буде дуже багато, і пристрій просто не зможе нормально їх відігравати. Для таких цілей в сервіс буде добавлений паттерн ObjectPooling.

ObjectPooling - Паттерн, для створення об’єктів. Під час старту сцени, створюється певна кількість об’єктів і поміщається в контейнер. Замість того, щоб створювати новий екземпляр, об'єкт витягується з контейнера, і його стан скидається за потреби. Коли об'єкт більше не потрібен, його повертають до контейнера для подальшого використання.

У нашому випадку ми доповнимо цей патерн, добавимо максимальну кількість об’єктів в контейнері і час, після якого такий об’єкт може знову бути вибраним з контейнера. Така реалізація підвищить фпс та вирішить проблему з багатьма звуками. Конфіг для звуку також буде зберігатись в ScriptableObject. (Рис. 3.31)

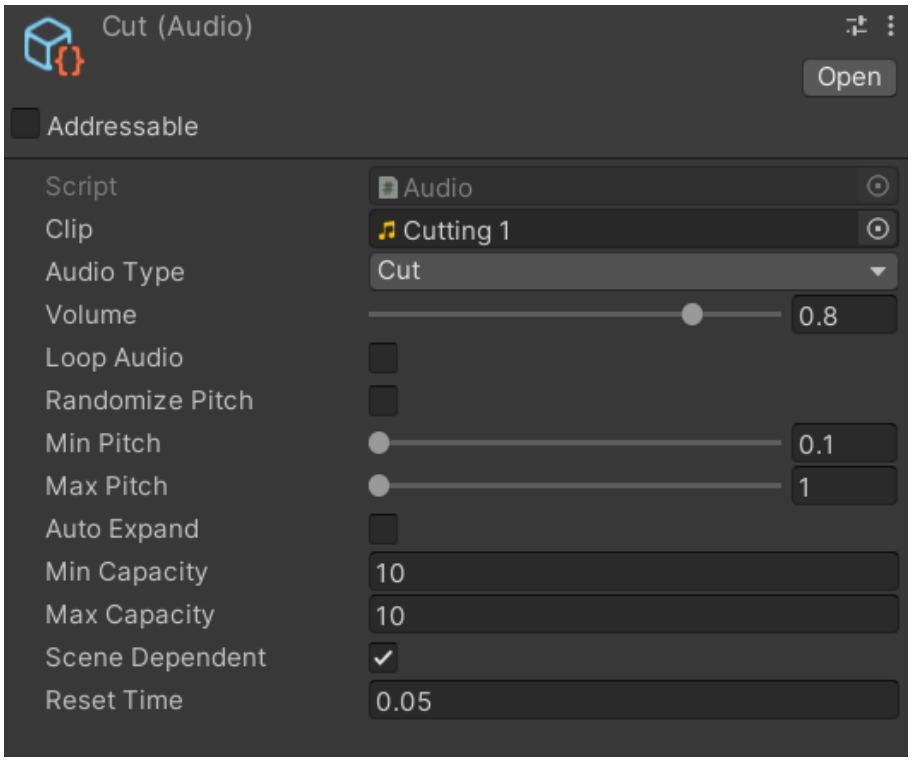

Рис. 3.31 - Конфігурація звуку.

Джерело: [створено автором]

3.2.11. Вібрації

Вібрації будуть реалізовані за допомогою плагіну від Lofelt Studio - Nice Vibrations.

Nice Vibrations - це просте, але потужне рішення для додавання тактильного зворотного зв'язку високої чіткості (HD) у ігри для iOS, Android, ПК та консолей. Загалом, телефони на iOS працюють набагато краще, ніж телефони на Android, і забезпечують більший контроль, тому користувачі отримають набагато більше задоволення від використання iPhone [8].

Такі вібрації будуть використані для всіх типів звуків, а також для блендеру, що зробить відчуття гри набагато кращим. (Рис. 3.32)

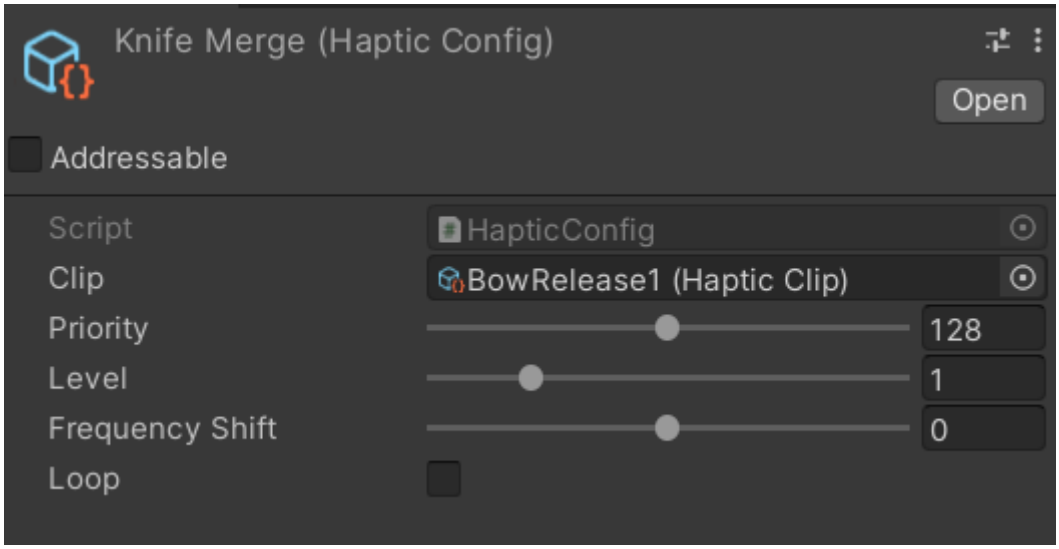

Рис. 3.32 - Конфігурація вібрації.

Джерело: [створено автором]

Висновок до розділу 3

У розділі 3 кваліфікаційної роботи було проаналізоване програмне та технічне забезпечення. Також була розписана архітектура застосунку та показані проблеми, які вона вирішує. У розділі була проведена основна робота із застосунком та були детально описані модулі гри.

РОЗДІЛ 4
ТЕСТУВАННЯ

4.1. Функціональне тестування

Для перевірки функціональних можливостей результату кваліфікаційної роботи, буде проведене функціональне тестування.

Функціональне тестування - це вид тестування програмного забезпечення, що перевіряє, чи функціонує програма чи система вірно і відповідає вимогам. Це включає оцінку поведінки програмного забезпечення за допомогою тестування різних вхідних даних та порівняння вихідних даних з очікуваними результатами. Такий вид тестування має вирішальне значення для виявлення дефектів та гарантування того, що програмне забезпечення правильно виконує свої функції до того, як воно буде випущено. Далі, у таблиці 4.1, є результати тестування застосунку.

Таблиця 4.1

Результати функціонального тестування

№	Назва тесту	Очікуваний результат	Отриманий результат	Тест пройде-ний?
1	Загрузка меню сцени	При загрузці гри початковою сценою є меню	Гра загрузилась. Відображається меню сцена	+
2	Робота кнопки «Play»	При натиску на кно-пку «Play» гра переходить в сцену рівня	При натиску на кно- пку «Play» гра перейшла в сцену рівня	+
3	Наявність туторіалу на першому рівні	При першому запуску на першому рівні повинен з’явитись туторіал	Туторіал з’явився на першому рівні	+
4	Вікно налаштувань	При кліку на налаштування відкривається вікно налаштувань	При кліку на налаштування відкрилось вікно	+

			налаштувань	
5	Розрізання фрукту	При розрізанні фрукту, він повинен полетіти в блендер	Фрукт летить в блендер	+
6	Рух камери	Камера повинна рухатись за першим ножем, після його знищення вертатись до найближчого	Камера рухається за ножем і повертається до найближчого	+
7	Знищення всіх цілей	При знищенні всіх цілей повинен показуватись екран виграшу	Цілі знищились, екран виграшу показався	+
8	Знищення всіх ножів	При знищенні всіх ножів, якщо цілі не були зібрані, показується екран програшу	Ножі знищились, цілі не зібрані, екран програшу показаний	+
9	Робота звуків	При увімкненому налаштуванні «Звуки», у грі повинні відтворюватись звуки	Налаштування «Звуки» увімкненно, звуки відтворюються	+
10	Робота вібрацій	При увімкненному налаштуванні «Вібрації», у грі повинні відтворюватись вібрації	Налаштування «Вібрації» увімкнено, вібрації відтворюються	+

Висновок до розділу 4

У розділі 4 кваліфікаційної роботи був проведений аналіз роботи застосунку, його функціональні можливості. Була перевірена поведінка гри, у результаті якої виявлено, що застосунок працює правильно і відповідає початковим вимогам.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи була проаналізована актуальність ігрової індустрії. Був проведений аналіз жанру аркада, а також вивчені плюси та популярність такого виду ігор.

Були проаналізовані популярні ігрові рушії, середовища для розробки 3д моделей, середовища 2д графіки, а також редактори коду. Були наведені у таблиці плюси та мінуси кожного з них, а також ключові особливості відбору. У розділі 1 було отримане необхідне програмне забезпечення для виконання кваліфікаційної роботи.

У розділі 2 був проведений аналіз базових можливостей вибраного програмного забезпечення. Також була вивчена необхідна математична база для роботи з системами координат, операціями пересування, повороту об’єкта у такій системі. Також були наведені проблеми ігрового рушія Unity у побудові архітектури застосунку.

У розділі 3 був проведений детальний опис реалізації гри. Спершу була описана архітектура проекту та фреймворки, які допоможуть її реалізувати. Також був проаналізований стек технологій, який будем використовувати для створення гри. Були проаналізовані всі модулі застосунку та їх реалізація в відповідних середовищах розробки.

У розділі 4 провели тестування застосунку. Перевірили основні моменти гри та їх відповідність до поставленої мети роботи кваліфікаційної роботи. У результаті тестування було вияснено, що застосунок працює правильно та відповідає початковим вимогам.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unity documentation URL: https://docs.unity3d.com/Manual/index.html. (дата звернення: 25.12.2022 р.).
2. Blender documentation URL: https://docs.blender.org. (дата звернення: 07.01.2023).
3. C# documentation URL: https://learn.microsoft.com/enus/dotnet/csharp. (дата звернення: 15.01.2023).
4. Zenject documentation URL: https://github.com/modesttree/Zenject. (дата звернення: 05.02.2023).
5. UniRx documentation URL: https://github.com/neuecc/UniRx. (дата звернення: 18.02.2023).
6. Dotween documentation URL: http://dotween.demigiant.com/documentation.php. (дата звернення: 07.03.2023).
7. Photoshop documentation URL: https://helpx.adobe.com/photoshop/user-guide.html. (дата звернення: 22.03.2023).
8. Nice Vibrations documentation URL: https://nice-vibrations docs.moremountains.com/. (дата звернення: 01.04.2023).

ДОДАТКИ

ДОДАТОК А

Покликання на гру в Play Market

«Knife Cut Merge Hit» URL:

https://play.google.com/store/apps/details?id=com.DanMax.KnifeCutMergeHit&hl=uk&gl=US

Покликання на гру в App Store

«Knife Cut Merge Hit» URL:

https://apps.apple.com/ua/app/knife-cut-merge-hit/id1669234733