КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА КРОСПЛАТФОРМНОЇ БАГАТОЖАНРОВОЇ ГРИ

Виконав студент 4-го курсу
Олександр БАНДАЛАК _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Євгеній ІВАНОВ _____
(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
« 29 » травня 2023 р.,
протокол № 11
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 51 (додатки – 2 сторінки) сторінка, 28 ілюстрацій, 46 джерел посилань, 1 додаток.

ANDROID, FIREBASE AUTHENTICATION, FIREBASE FIRESTORE, FIREBASE REALTIME DATABASE, LINUX, UNITY, WINDOWS, БАГАТОЖАНРОВІСТЬ, БАЗА ДАНИХ, ІГРОВА СЦЕНА, ІНТЕРФЕЙС, КРОСПЛАТФОРМЕНІСТЬ, ЛОКАЛІЗАЦІЯ, ОПЕРАЦІЙНА СИСТЕМА, ПАТЕРН.

Для виконання завдання була додатково використана платформа розробки мобільних та вебдодатків Firebase [1], а саме Firebase Firestore для збереження користувацьких даних, Firebase Realtime Database для синхронізації ігрових даних в реальному часі та Firebase Authentication для впровадження в додатку системи акаунтів. Платформа раніше використовувалася для розробки мобільних та вебдодатків, але наразі є можливість використовувати її для розробки додатків для комп’ютерів [2].

Об’єктом роботи є процес створення кросплатформних ігор з використанням набутих навичок та знань. Предметом роботи є кросплатформна багатожанрова гра.

Метою роботи є створення багатожанрової гри з можливістю встановлення на пристрої з різними операційними системами чи апаратними платформами.

Методи розроблення: інкапсуляція, наслідування, поліморфізм. Патерни: Команда, Посередник, Одинак та Спостерігач (патерн проектування – це типовий спосіб, схема чи шаблон розв’язання певної проблеми [3]). Принципи: KISS та DRY. Інструменти розроблення: багатоплатформний інструмент для розробки відеоігор і застосунків Unity [4], кросплатформне інтегроване середовище розробки програмного забезпечення Rider від JetBrains [5] та мова програмування C#.

Результати роботи: був створений кросплатформний застосунок, який дозволяє користувачам використовувати одного і того самого персонажа та

ігрову валюту для двох ігрових частин (частин з різними жанрами): інкрементальна гра та онлайн стратегія. Також користувач має змогу встановити гру на одну з наступних операційних систем: Android, Windows, Linux та Mac.

Розроблений застосунок є прикладом поєднання типової інкрементальної гри з функціоналом і можливостями притаманними даному жанру ігор та онлайн стратегії, в якій хід гри залежить від вибору користувача. Також даний проект можна розцінювати як приклад використання таких технологій як Firebase Firestore, Firebase Realtime Database та Firebase Authentication для створення не тільки мобільного застосунку, а і застосунку для комп'ютера.

Архітектура застосунку була створена так, щоб надалі можна було б з легкістю додавати нових персонажів чи ігрових ефектів. Також є можливість додати до гри третю частину з іншим ігровим жанром. Наприклад, пригодницьку, дієву чи іншу гру.

Є можливість розвивати стратегічну частину гри, додаючи до різних персонажів та предметів ігрові ефекти, які будуть впливати на хід гри. Крім ефектів, до гри можна додати різні ігрові режими, які будуть відрізнятися логікою та функціоналом. Оскільки персонажі та предмети між різними режимами спільні, то інкрементальна гра також буде розвиватися.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БД – база даних;

ОС – операційна система;

JSON – JavaScript Object Notation, текстовий формат обміну даними.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Доступність гри чи застосунку тільки на одній апаратній платформі чи тільки на одній ОС (операційній системі), скорочує кількість користувачів (а для платних застосунків – кількість потенційних клієнтів) у декілька разів і робить застосунок менш гнучким. Наприклад, за статистикою з онлайн платформи Statcounter [6], яка зображена на рис. А.1 (додаток А), можна побачити, що 54 відсотки ринку припадає на мобільні пристрої і 42 відсотки – на комп’ютери. Це означає, що розробивши додаток тільки на мобільний пристрій чи тільки на комп’ютер, розробник чи компанія втрачає приблизно половину потенційних користувачів чи клієнтів.

Але на цьому проблема не закінчується, оскільки для мобільних пристроїв та комп’ютерів існують різні ОС. Згідно з іншою статистикою з онлайн платформи Statcounter [7], яка зображена на рис. А.2 (додаток А), можна побачити, що 40 відсотків ринку припадають на пристрої з ОС Android, 31 відсоток на пристрої з ОС Windows, 16 відсотків на пристрої з ОС iOS, 7 відсотків на пристрої з ОС OS X. Отже, якщо розробити застосунок виключно для однієї ОС, то будуть втрати потенційних користувачів та клієнтів, які використовують інші ОС.

Актуальність роботи та підстави для її виконання. Навички та знання у створені кросплатформного застосунку є досить цінними, оскільки застосунок, який розроблений хоча б для двох ОС чи апаратних платформ, зможе бути використаний на більшій кількості пристроїв, ніж не кросплатформний застосунок. Також під час розробки кросплатформного застосунку можна отримати важливий досвід з використанням одних чи інших інструментів, чи сервісів для різних апаратних платформ, чи ОС. Оскільки навіть з використанням кросплатформних інструментів, можуть бути певні особливості у їхньому використанні на різних системах.

Мета й завдання роботи. Метою дипломної роботи є розробка кросплатформної багатожанрової гри. Для досягнення цієї мети поставлено наступні завдання:

— дослідити, які технології використовуються для створення кросплатформних додатків, зокрема 2D ігор;

— обрати одну чи декілька технологій, які будуть відповідати всім критеріям, потрібним для виконання поставленого завдання;

— ознайомитися з документацією кожної обраної технології;

— визначити основний та додатковий функціонал гри;

— розробити дизайн та потрібні елементи інтерфейсу для гри;

— створити на основі попередніх пунктів кросплатформну багатожанрову гру;

— перевірити належне функціювання гри на різних апаратних платформах та ОС.

Об'єкт, методи й засоби розроблення. Об’єктом є процес створення кросплатформних ігор з використанням набутих навичок та знань.

Для розробки самої гри використовувався інструмент Unity та його можливість створювати додатки для різних апаратних платформ та ОС. Зокрема, для наступних ОС: Windows, Android, Linux, Mac. Спочатку була розроблена частина гри з жанром інкрементальна гра. Розроблялися ігрові персонажі, анімації, ефекти, ігрова валюта та ігрова логіка. Для збереження даних користувача з першої частини гри в БД (базі даних), використовувалася платформа Firebase, а саме Firebase Firestore та Firebase Authentication.

Далі розроблялася друга частина гри з жанром стратегія. Для розробки використовувалися створені частини з першої частини. Для можливості користувачам взаємодіяти один з одним через Інтернет, був використаний ще один продукт платформи Firebase, а саме Firebase Realtime Database.

Паралельно з розробкою ігрових частин, до застосунку додавалася локалізація всіх текстових полів. Мови які підтримуються: українська та англійська.

Можливі сфери застосування. Розроблений застосунок може використовуватися як типова інкрементальна гра чи стратегія. Крім того, даний

проект є прикладом розробки кросплатформної гри з використанням кросплатформних можливостей Unity та Firebase.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РОЗРОБКИ КРОСПЛАТФОРМНИХ ІГОР

Для створення кросплатформних ігор існує велика кількість різноманітних технологій. Наприклад, платформи Unreal Engine, Cocos2d-x, Godot Engine, Unity та інші. Кожна платформа надає унікальний функціонал, який вирізняє її серед інших подібних платформ.

1.1 Дослідження платформи для створення ігор Unreal Engine

Unreal Engine – це багатоплатформна технологія, яка здатна розробляти 3D та 2D ігри [8]. Інтерфейс Unreal Engine зображений на рис. 1.1 [9]. Дана платформа відома розробкою високоякісної та швидкої 3D графіки, оскільки сама платформа та ігри, які на ній створюються, написані мовою програмування C++. Популярними іграми, створеними за допомогою Unreal Engine є Tomb Raider, The Matrix Awakens, Redfall, The Witcher, Echoes of the End та інші [10].

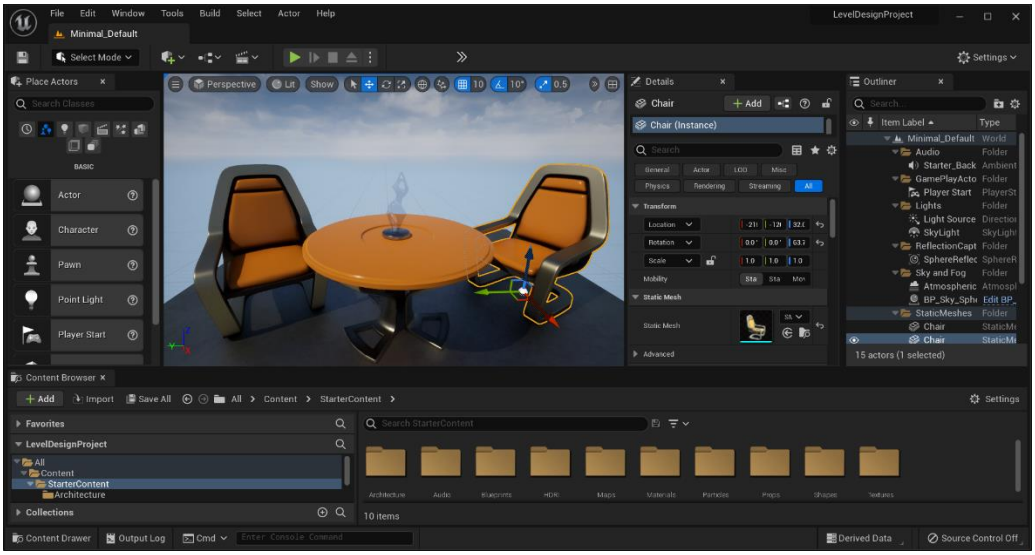


Рисунок 1.1 – Інтерфейс платформи Unreal Engine

Дана платформа дозволяє розробляти кросплатформні ігри на таких ОС як Windows, OS X, PlayStation, XBox, Nintendo Switch, iOS, Android та інших [11]. Але

здебільшого платформа позиціонується саме як технологія розробки 3D ігор з потужною графікою для комп'ютерів та консолей. Можливість створювати 2D ігри та мобільні ігри з'явилася в Unreal Engine тільки з четвертої версії платформи [12].

1.2 Дослідження платформи Cocos2d-x

Cocos2d-x – це платформа для розробки 2D та 3D ігор, яка дозволяє створювати ігри використовуючи такі мови програмування як C++, JavaScript та Lua. Інтерфейс Cocos2d-x зображений на рис. 1.2 [13]. Особливістю даної платформи є простота використання та вбудовані механізми, які додають фізику до створюваної гри [14].

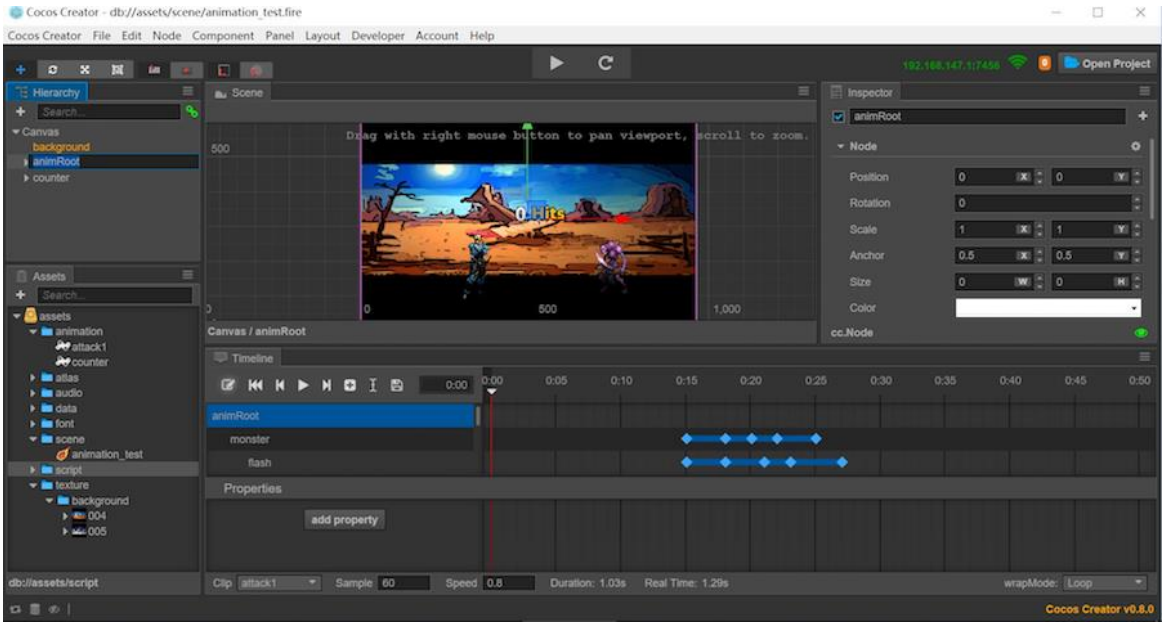


Рисунок 1.2 – Інтерфейс платформи Cocos2d-x

Спочатку платформа дозволяла створювати виключно 2D ігри, але з часом розробники додали можливість створювати 3D ігри. Серед популярних ігор, які були розроблені з використанням платформи Cocos2d-x, варто зазначити Top War: Battle, Hungry Shark Arena, Designville, Slash Brave та інші [15].

Дана платформа дозволяє розробляти кросплатформні ігри для таких ОС як iOS, Android, Tizen, Windows, Linux, OS X [14]. Попередня версія Cocos2d підтримувала розробку ігор тільки на платформи Windows, OS X, та Linux [14].

1.3 Дослідження платформи Godot Engine

Godot Engine – це безкоштовний багатоплатформний ігровий рушій, який дозволяє створювати кросплатформні 2D та 3D ігри. В даному рушію є вбудований редактор коду, рушій рендерингу графіки, інструмент для відтворення аудіо та інше [16]. Для розробки 3D ігор в Godot Engine присутні системи освітлення, фізики та підтримка матеріалів з різними властивостями [16]. На відміну від інших ігрових платформ, які для створення 2D ігор, просто прибирають одну вісь з тривимірної системи координат, Godot Engine використовує окремий рушій для 2D ігор, в якому всі інструменти налаштовані саме на 2D графіку [16]. Інтерфейс даної платформи зображений на рис. 1.3 [17]. Особливістю інтерфейсу платформи Godot Engine є те, що ним можна користуватися, використовуючи веббраузер, а не тільки комп'ютерний застосунок [17]. Також, дана платформа має власну мову програмування GDScript, яка схожа за синтаксисом на Python.

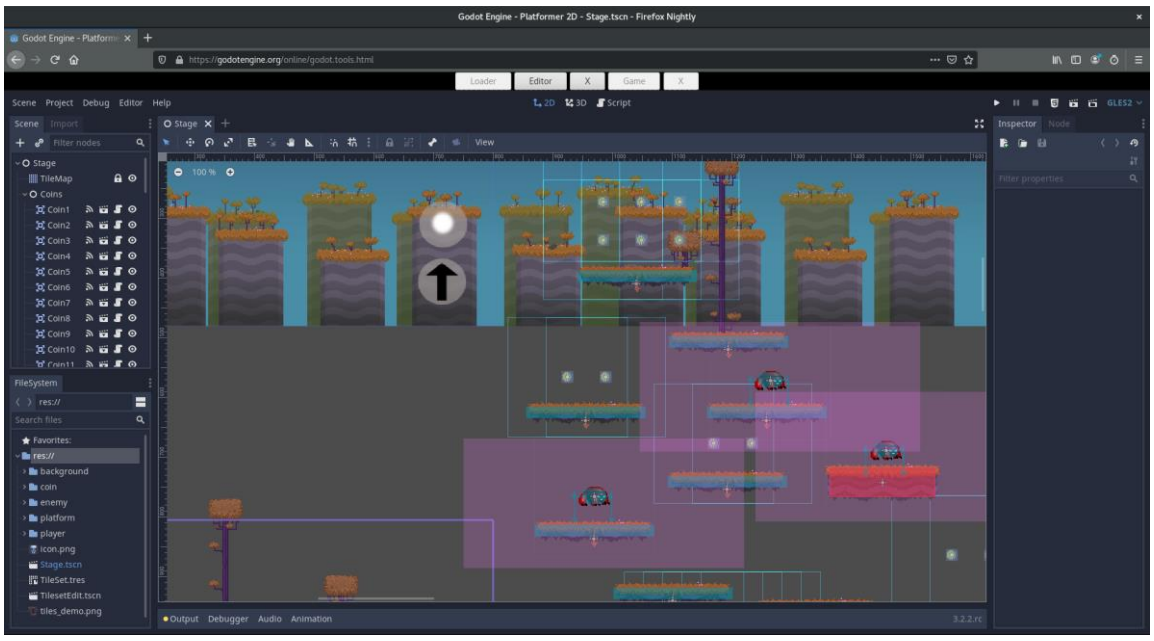


Рисунок 1.3 – Інтерфейс платформи Godot Engine

Дана платформа підтримує розробку кросплатформних ігор на наступні ОС: Windows, OS X, Linux, Android, iOS та інші [18]. Хоча платформа дозволяє створювати кросплатформні ігри, експорт проектів дещо відрізняється для різних платформ. Наприклад, для експорту гри для ОС iOS потрібно мати комп'ютер з ОС

OS X [18]. Крім того, зазначається, що через апаратні обмеження, певний функціонал гри не буде працювати однаково на різних ОС.

Хоча платформа не така відома як Unreal Engine чи Unity, але вже велика кількість розробників створюють ігри з використовуючи Godot Engine. Наприклад, A Most Extraordinary Gnome, Lumencraft, Cassette Beasts, Brotato та інші [19].

1.4 Дослідження платформи Unity

Unity – це потужний ігровий рушій та багатоплатформний інструмент для розробки кросплатформних 2D та 3D ігор [4]. В дану платформу вбудована фізика, зіткнення об’єктів, 3D-візуалізація, освітлення та інше [4]. Інтерфейс платформи зображений на рис. 1.4 [20]. В платформу вбудовані інструменти для створення анімацій, побудови 2D та 3D об’єктів та створення скелета для ігрових об’єктів [21].

Рисунок 1.4 – Інтерфейс платформи Unity

Unity популярна платформа серед розробників мобільних ігор, оскільки дозволяє створювати адаптивні сцени (сцени, які автоматично підлаштовуються під різні дисплеї пристроїв), які будуть добре відображатися на різних мобільних

пристроях. Крім того, ця можливість є плюсом і для розробки комп'ютерних застосунків.

Використовуючи Unity розробник може створювати ігри на такі платформи як Windows, OS X, Linux, iOS, Android, PS5, PS4, Xbox One та інші [22]. Крім створення 2D та 3D ігор, Unity дозволяє створювати ігри з розширеною реальністю (поєднання віртуальної, доповненої та змішаної реальностей [23]) для наступних платформ: ARKit, ARCore, Microsoft HoloLens, Windows Mixed Reality, PlayStation VR та інших [22].

Для написання коду на платформі Unity використовується кросплатформна мова програмування C#. Але вбудованого редактора коду в платформі не має, тому розробник має можливість самостійно обирати, яким інтегрованим середовищем розробки користуватися. Наприклад, наступні інтегровані середовища розробки зручно використовувати разом з Unity: Visual Studio Code, Rider, Visual Studio, Atom, MonoDevelop та інші [24].

Також платформа дозволяє додавати до власного проекту офіційні та користувацькі плагіни та розширення, що дозволяє використовувати готові механізми замість того, щоб створювати їх самостійно. Крім плагінів та розширень, Unity надає розробникам можливість публікувати та використовувати безкоштовні та платні ігрові зображення, моделі, звуки, шрифти та інше [25].

На Unity були створені популярні ігри, які були відомі у свій час. Наприклад, Pokemon GO, Beat Saber, Hearthstone, Cuphead, Hollow Knight та інші [26]. Серед перечислених ігор є не тільки 2D та 3D ігри, а також ігри із використанням змішаної реальності (Pokemon Go) та віртуальної реальності (Beat Saber) [23].

1.5 Дослідження платформи для розробки онлайн застосунків Firebase

У сучасному світі майже кожна відеогра використовує підключення до Інтернету. Навіть, якщо безпосередньо в грі немає онлайн режиму, то певна інформація про ігрові чи користувацькі дані зберігається у БД, підключення до якої відбувається з використанням Інтернету. Це потрібно, щоб користувач мав доступ

до своїх облікових даних після зміни пристрою, який він використовував. Крім того, багато ігор розраховані на взаємодію користувачів один з одним в режимі онлайн. Тому підключення гри чи додатка до онлайн БД покращує зручність користування додатком. Крім того, якщо гра кросплатформна, то користувач має змогу увійти в свій обліковий акаунт з пристроїв з різними ОС.

Більшість платформ для створення ігор мають власні інструменти для збереження ігрових даних та для додавання до гри онлайн режиму. Навіть якщо такого вбудованого інструменту немає, то його можна додати встановивши відповідний плагін.

Але також існують інструменти, які не прив’язані до певного ігрового рушія чи платформи. Наприклад, Firebase – платформа, яка надає сервіс для роботи з БД, аутентифікацією, хостингом та іншим [1]. Firebase дозволяє створювати кросплатформні додатки з можливістю зберігати користувацькі дані в онлайн БД (Firebase Firestore), синхронізувати ігрову інформацію в режимі реального часу (Firebase Realtime Database), реєструвати нових користувачів та авторизовувати раніше створених (Firebase Authentication), зберігати та поширювати файли великого розміру (Firebase Storage) та інше [27]. Тобто розробнику не потрібно власноруч створювати сервер, який буде зв’язувати додаток чи гру з БД [1]. Дана можливість полегшує та пришвидшує розробку додатків та ігор, для яких потрібно зберігати інформацію про користувача в БД.

Дана платформа може використовуватися додатками, які використовують наступні мови програмування: C++, C#, Java, JavaScript, Python, Ruby та інші [28]. Також, платформа надає інший інтерфейс для використання деяких мов програмування у певних платформах розробки додатків та ігор. Наприклад, для мови програмування C# та цієї ж мови, але з її використанням на платформі Unity, Firebase надає різний механізм використання своїх сервісів [28]. Також це стосується мови програмування Java з використанням платформи розробки мобільних додатків та ігор Android Studio.

Оскільки доступ до всіх сервісів даної платформи, здійснюється з використання мережевих технологій, то до однієї й тієї ж БД чи іншого сервісу

можна під’єднати додатки та ігри, які написані з використанням різних платформ розробки та мов програмування.

1.6 Вибір платформи для розробки проекту

Усі досліджені платформи для розробки ігор дозволяються створювати кросплатформні застосунки. Але найбільше функціоналу та підтримки різних ОС у Unreal Engine та Unity. Дані платформи дозволяють розробляти ігри для мобільних пристроїв (мобільні телефони та планшети), комп’ютерів та ігрових консолей.

Але особливості платформи Unreal Engine, вказують на те, що платформа здебільшого розрахована саме на розробку 3D ігор з потужною графікою. Тому для розробки даного проекту, можливості платформи Unreal Engine будуть надлишкові.

Платформа Unity більше підходить для розробки 2D гри та має більшу кількість підтримуваних платформ та ОС, на яких будуть працювати створені ігри. Крім того, для збереження інформації користувача та ігрових даних в онлайн БД буде використана платформа розробки онлайн застосунків Firebase, яка може надавати використання своїх сервісів для Unity. Гра, яка буде створена з використанням платформи Unity для різних платформ чи апаратних систем, буде використовувати спільну онлайн БД, яку буде надавати платформа Firebase. Таким чином, користувачі з мобільних пристроїв зможуть взаємодіяти в онлайн режимі з користувачами комп’ютерів.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ АЛГОРИТМІВ, ЯКІ ВИКОРИСТОВУЮТЬСЯ ПРИ РОЗРОБЦІ ІГОР

Математика та математичні алгоритми широко використовуються у іграх (відеоіграх). Рухи персонажів, падіння предметів, освітлення об’єктів, передбачення шляхів та дій були б неможливими без використання математики. Хоча більшість алгоритмів вбудовані в ігрові рушії, все одно потрібно розуміти як вони працюють, щоб ефективно їх використовувати. Існує безліч математичних алгоритмів, які розв’язують одну чи іншу задачу в розробці ігор. Наприклад, алгоритм Дейкстри для пошуку найкоротшого шляху у графах допомагає розв’язати проблему пошуку ігровими персонажами потрібного шляху [29]. Алгоритм побудови дерева рішень та методи індуктивного навчання допомагають ігровим персонажам приймати відповідні рішення спираючись на прописані правила.

2.1 Алгоритми пошуку даних

У іграх часто використовуються алгоритми для пошуку потрібної інформації з великого набору даних. Наприклад, для пошуку ігрових предметів за заданими параметрами або при підборі противника з подібними до гравця характеристиками. Оскільки пошук може займати велику кількість часу та ресурсів пристрою, ефективніше буде використовувати рішення даної проблеми у вигляді оптимізованих алгоритмів пошуку.

Одним з алгоритмів пошуку, який має логарифмічну складність, є алгоритм бінарного пошуку [30]. Даний алгоритм може здійснювати пошук тільки на відсортованій вибірці даних, але витрати на сортування компенсуються швидким пошуком у випадках, коли вибірка незмінна, або не часто змінюється. Наприклад, якщо потрібно знайти велику кількість ігрових предметів за певним параметром, то потрібно буде відсортувати всі предмети по заданому параметру тільки один раз, а

пошук буде здійснюватися багато разів, що в результаті зробить алгоритм в цілому ефективнішим [30].

Але якщо потрібно здійснювати пошук певного елемента за двома чи більшою кількістю параметрів, то алгоритм бінарного пошуку не буде ефективно працювати, оскільки сортування за декількома параметрами не буде точним. В такому випадку буде ефективніше працювати алгоритм KD-дерева [30]. Даний алгоритм витрачає додатковий час та пам’ять на побудову дерева, але після цього, час, який витрачатиметься на пошук елемента в даному дереві, буде логарифмічним, а не лінійним, як в наївних алгоритмах пошуку.

В основному алгоритм використовується для пошуку найближчої точки у багатовимірній системі координат. Наприклад, потрібно знайти противника для гравця, використовуючи декілька ігрових параметрів (рейтинг, рівень тощо). Кожний параметр розглядається як значення з осі координат, тому задача пошуку противника зводиться до задачі пошуку найближчої точки у багатовимірній системі координат, яка ефективно вирішується алгоритмом KD-дерева. Також алгоритм KD-дерева використовується для виявляння зіткнення між об'єктами у тривимірному просторі [31].

Для збільшення ефективності пошуку даних, пошук найближчого елементу зводять до пошуку елементу з певного радіуса. Для цього використовується алгоритм пошуку по радіусу. Даний алгоритм перед виконанням пошуку будує квадрі- або R-дерево, а потім здійснює пошук [32]. Наприклад, для пошуку противника для гравця, не обов’язково шукати найближчого по параметрах противника, інколи достатньо, щоб ці параметри попадали в певний радіус відносно параметрів гравця. Такий підхід зменшить час виконання пошуку противника, що в результаті зменшить час очікування для користувачів.

Якщо множина елементів, серед яких потрібно здійснювати пошук, постійно змінюється, то витрати часу та пам’яті на побудову дерева для алгоритму KD-дерева чи для алгоритму пошуку по радіусу або на сортування елементів для бінарного пошуку, будуть великими, що в результаті зменшить загальну

ефективність алгоритму пошуку. В такому випадку краще будуть працювати алгоритми, які не здійснюють попередньої підготовки множини елементів.

2.2 Алгоритми 2D та 3D морфінгу

Крім алгоритмів, які тісно пов’язані із ігровою логікою, також існують математичні алгоритми, які впливають на відображення і зміну ігрових елементів. Наприклад, алгоритми 2D та 3D морфінгу дозволяють плавно переводити об’єкти однієї форми в об’єкти іншої [33]. Для цього відбувається пошук найближчих точок першої фігури до точок другої. Для пошуку може використовуватися алгоритм KD-дерева. Під час знаходження найближчих точок також враховуються ребра, які їх з’єднують. Далі відбувається відображення першого об’єкта в другий, використовуючи формулу лінійної інтерполяції [33]. Плавний 3D перехід від конуса до сфери зображений на рис. 2.1 [34].

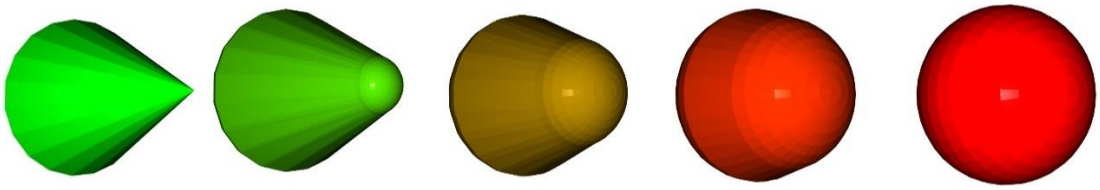


Рисунок 2.1 – Частини плавного перетворення конуса в сферу

Крім морфінгу 2D та 3D об’єктів, існує морфінг 2D зображень, в якому використовується перетворення кольорів пікселів вхідного зображення на кольори пікселів вихідного зображення. Перетворення 2D зображень аналогічно до перетворення 2D та 3D об’єктів, використовує лінійну інтерполяцію. В результаті створюється ілюзія плавної зміни зображень.

Дані алгоритми використовуються для створення ігрових анімацій, а саме для зміни форми ігрових персонажів чи інших деталей гри. Використання плавних переходів дозволяє створювати реалістичні ігрові анімації.

У більшості ігрових платформ морфінг об’єктів здійснюється з використанням вбудованих інструментів та інтерфейсу платформи. Тому

розробнику не обов’язково власноруч додавати морфінг об’єктів до створюваної гри. Але в іграх, де графіка є важливою частиною розробки, розробники можуть використовувати алгоритми морфінгу для покращення роботи та вигляду ігрової графіки.

2.3 Алгоритми інтерполяції

Для створення плавних анімацій руху ігрових персонажів та об’єктів, крім морфінгу, використовується інтерполяція кривих. Дані алгоритми дозволяють створювати нелінійні рухи об’єктів вздовж певної кривої лінії побудованої за проміжними точками між початковими та кінцевими координатами об’єктів [33].

Існує велика кількість алгоритмів інтерполяції кривої. Наприклад, квадратична, кубічна інтерполяції, інтерполяція сплайнами Біз’є, NURBS інтерполяція та інші [33]. На рис. 2.2 [35] зображена крива, яка була побудована алгоритмом інтерполяції Біз’є, який використовує вхідні точки, через які повинна проходити лінія, щоб побудувати криву.

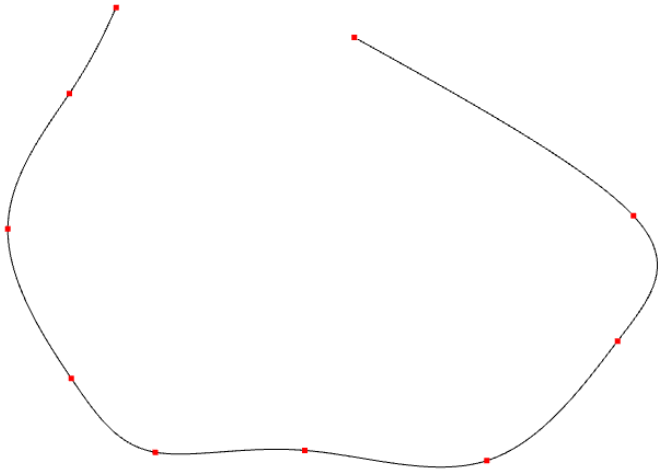


Рисунок 2.2 – Крива, побудована алгоритмом інтерполяції Біз’є

Рухи від точки до точки по нелінійному шляху виглядають природнішими і в результаті анімація буде виглядати реалістичною та якісною. Якщо потрібно створити анімацію руху ігрового об’єкта через певні точки на площині, то проміжні точки будуть вхідними даними для алгоритму інтерполяції. В результаті буде

отримана крива лінія, по якій буде плавно рухатися ігровий об’єкт. Інтерполяція кривих зазвичай є вбудованою можливістю більшості платформ розробки ігор, тому розробнику не потрібно самостійно реалізовувати подібні алгоритми.

Крім інтерполяції кривої, алгоритми інтерполяції використовуються для інтерполяції площин, яка дозволяє згладжувати форму ігрових об’єктів або будувати ігрові площини на основі вхідних точок [35]. Для інтерполяції площин використовуються алгоритми інтерполяції Біз’є та NURBS. З використанням інтерполяції у ігровій графіці, об’єкти стають плавними та реалістичними.

Також інтерполяція площин дозволяє будувати складні поверхні, використовуючи певний шаблон площини, який багато разів складається з іншими шаблонами, щоб в результаті отримати цілісну поверхню [35]. Даний шаблон створюється з вхідної множини точок використовуючи алгоритм інтерполяції. Такий підхід використовується у випадках, коли є обмежена кількість вхідних точок або даних про поверхню, інтерполяція площин може використовуватися для апроксимації невідомої поверхні [35]. Наприклад, на рис. 2.3 [36] зображена сіра поверхня, яка побудована алгоритмом NURBS інтерполяції, який використовує тільки вхідні (червоні) точки для побудови.

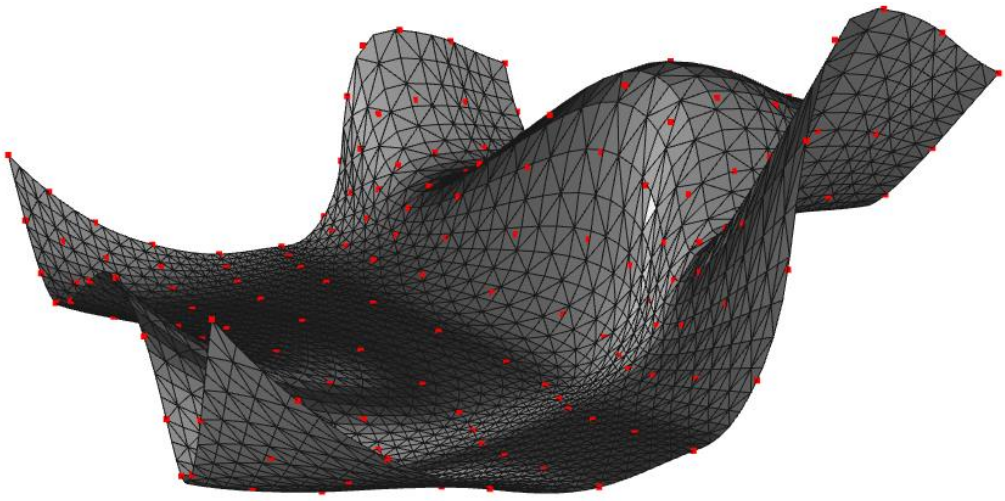


Рисунок 2.3 – Поверхня, побудована алгоритмом NURBS інтерполяції

РОЗДІЛ 3 РОЗРОБКА ГРИ З ВИКОРИСТАННЯМ ДОСЛІДЖЕНИХ ПЛАТФОРМ

Розроблена кросплатформна багатожанрова гра була створена з використанням платформи Unity та Firebase. Всі елементи на ігровій сцені підлаштовуються під різні розміри екранів, включаючи мобільні та комп'ютерні дисплеї.

Основною ідеєю розроблюваної гри є створення двох ігрових частин з різними ігровими жанрами (інкрементальна гра та стратегія), які будуть використовувати спільних ігрових персонажів та ігрову валюту. Тобто, обравши персонажа для інкрементальної гри, він буде обраний і для стратегії. Якщо під час взаємодії з грою отримати ігрову валюту в одному режимі, то дана валюта буде доступна і в іншому. Використовуючи ігрову валюту, гравець має можливість купувати ігрові предмети, які також є спільними між двома частинами гри. Ігрові предмети мають тільки косметичний характер (змінюють вигляд ігрового персонажа), але із розвитком проекту, ігрові предмети отримають унікальні ефекти, які будуть діяти і в інкрементальній грі і в стратегії.

3.1 Розробка інкрементальної гри

3.1.1 Механіка та унікальність інкрементальної гри

Інкрементальна гра – це жанр відеоігор, в якому гравці шляхом багаторазового натискання на певний ігровий об'єкт чи об'єкти, отримують певний результат у вигляді ігрової валюти, збільшення ігрового рівня чи отримання інших ігрових винагород [37]. Також даному жанру притаманне автоматичне натискання (натискання на ігровий об'єкт без участі гравця) та можливість його покращення. Через це гравець може залишити на деякий час свій пристрій із увімкненою грою, а після повернення гравець отримає певні винагороди. Розробники використовують

різні підходи для того, щоб протидіяти такій взаємодії з грою. Наприклад, може бути використане обмеження на отримання винагород без натискань або автоматичне блокування ігрового інтерфейсу, якщо гравець відсутній певний час. Подібний підхід був використаний в наступних інкрементальних іграх: Planet Clicker 2, Tap Titans 2: Idle Clicker RPG та Hero Factory – Idle tycoon.

Особливість розроблюваної інкрементальної гри буде унікальний підхід до розв'язання проблеми, описаної вище. Для того, щоб гравець не залишав гру, щоб отримати велику кількість винагород за автоматичні натискання, було вирішено додати ігрову подію, яка з'являється через певні випадкові проміжки часу та періодично понижує рівень здоров'я ігрового персонажа. Якщо рівень здоров'я ігрового персонажа опуститься нижче за нуль, то продовжити використовувати даного ігрового персонажа надалі буде неможливо. Крім того, отримані раніше нагороди будуть втрачені і гра закінчиться. Для продовження гри потрібно буде починати все спочатку. Щоб уникнути такого розвитку подій, гравець повинен фізичним натисканням на ігровий об'єкт, закінчити викликану подію.

3.1.2 Розробка основних частин інкрементальної гри

Для інкрементальної частини гри були розроблені наступні ігрові сцени: сцена завантаження, сцена оновлення, сцена з сюжетною історією, сцена для створення ігрового персонажа, основна сцена інкрементальної гри та кабінет гравця.

На сцені завантаження, відбувається перевірка підключення до БД та наявна версія гри. Якщо версія гри застаріла, то вмикається сцена з повідомленням про оновлення гри. На сцені для створення ігрового персонажа, яка зображена на рис. 3.1, був реалізований вибір одного персонажа із трьох різних типів. З подальшим розвитком проекту буде можливо додавати довільну кількість різних типів для персонажів. Різні персонажі відрізняються тільки виглядом у грі, але є можливість додати різні ігрові ефекти, які будуть впливати не тільки на інкрементальну частину гри, а й на стратегічну (другу) частину.

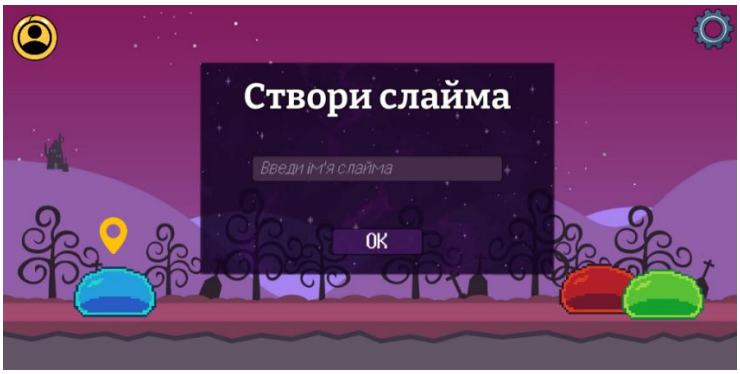


Рисунок 3.1 – Сцена для створення ігрового персонажа

Дані про ігрового персонажа (набір анімацій, зображень, ефектів тощо) використовуються на різних ігрових сценах. Для того, щоб уникнути повторного створення екземплярів класів, які будуть ці дані зберігати, було вирішено використати патерн Одинак [3]. Також даний патерн був використаний для збереження єдиного екземпляра даних про ігрові предмети та відтворення єдиного контролю над ігровими звуками та музикою на різних ігрових сценах. Основний код, який використовує патерн Одинак зображений на рис. 3.2. У 11 рядку коду додане статична змінна, в якій буде зберігатися єдиний екземпляр класу. Оскільки екземпляр даного класу повинен бути прикріпленим до ігрового об’єкта на сцені, то у 13 рядку, до класу додана функція, яка контролює, щоб екземпляр був унікальний та не видалявся після закриття сцени.

```
8   public class MusicManager : MonoBehaviour {
9       [SerializeField]
10      private AudioSource audioSource;  Changed in 1 asset
            14 usages
11      public static MusicManager Instance { get; private set; }
12
            Event function  Megarekrut65
13      private void Awake() {
14          if (Instance == null) {
15              Instance = this;
16          } else if (Instance != this) {
17              Destroy(gameObject);
18          }
19
20          LoadManager();
21          DontDestroyOnLoad(gameObject);
22      }
```

Рисунок 3.2 – Частина коду для контролю над ігровою музикою

Для створення ігрових анімацій використовувалася інтерполяція на проміжних точках переміщення ігрового об’єкта. В результаті лінійний рух замінився плавним рухом вздовж кривої лінії. З використанням інтерполяції, вигляд анімації став природнішим. Також для створення анімацій використовувалася зміна положення, вигляду, повороту та розміру ігрового персонажа. На рис. 3.3 зображені об’єднані кадри з відтворення анімації стрибка ігрового персонажа. В даній анімації використовується інтерполяція для руху ігрового персонажа вздовж кривої лінії. Також змінюється положення та вигляд ігрового персонажа.

Рисунок 3.3 Об’єднані кадри анімації стрибка ігрового персонажа

Для автоматичного натискання (автоматичного отримання нагород у вигляді ігрової валюти) був доданий механізм динамічного створення подібних ігрових об’єктів, які відрізняються кількістю ігрової валюти за активацію і кількістю ігрової валюти, яку кожної секунди буде отримувати гравець. Динамічне створення даних об’єктів дозволяє без змін в коді, додавати нові об’єкти чи видаляти старі. Також був доданий механізм покращення (підвищення рівня, кількості отримуваної ігрової валюти) до ігрових об’єктів, які дозволяють гравцеві автоматично отримувати ігрову валюту. Дане покращення відображається у вигляді половинок зірок та цілих зірок на ігровому об’єкті, який гравець покращив, як зображено на рис. 3.4.

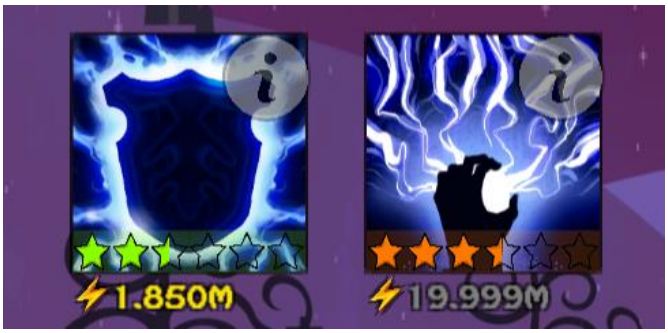


Рисунок 3.4 – Ігрові об’єкти для автоматичного отримання ігрової валюти

Для того, щоб залежність між ціною покращення ігрового об’єкта і кількістю ігрової валюти, яку він буде автоматично додавати гравцю, не була лінійною, було вирішено реалізувати приріст ціни та кількості отримуваної валюти з використанням відмінних математичних функцій. Наприклад, якщо для приросту ціни (залежно від рівня ігрового об’єкта) буде використовуватися степенева функція, а для приросту отримуваної ігрової валюти – квадратична чи кубічна функція, то залежність між ціною та отримуваною ігровою валютою буде нелінійна.

В даній частині грі, для ігрового персонажа доступні спеціальні ігрові ефекти. Відновлення втраченого рівня здоров’я, збільшення кількості отримуваної ігрової валюти під час натискань гравця, повернення всього рівня здоров’я персонажа після закінчення гри та інші ефекти зображені на рис. 3.5. Додавання ігрових ефектів було реалізовано так, щоб була утворена можливість розширення або скорочення кількості ігрових ефектів.

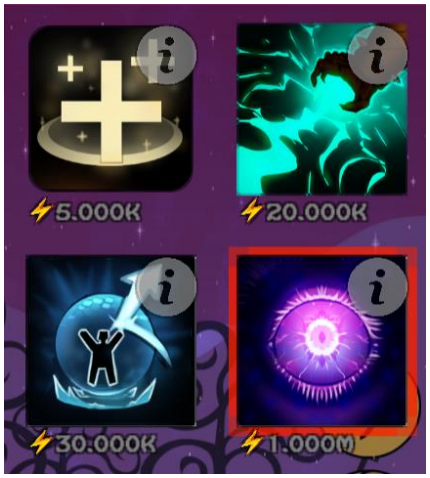


Рисунок 3.5 – Ігрові ефекти

Також було додано відображення короткого опису про ігрові ефекти та ігрові об’єкти для автоматичного отримання ігрової валюти, щоб проінформувати гравця.

Натискання на ігрові елементи спричиняють виклик багатьох подій. Наприклад, збільшення ігрової валюти, збільшення швидкості отримання ігрової валюти, збільшення рівня гравця та закінчення події, яка шкодить ігровому персонажу. За кожну подію відповідає окремий об’єкт. Події зв’язані одна з одною. Тому для того, щоб зменшити зв’язність між об’єктами, які відповідають за події, було вирішено використати патерн Посередник [3]. Був створений окремий об’єкт, який обробляє натискання на ігрові об’єкти та об’єднує події, які повинні бути зв’язаними між собою.

3.1.3 Великі числа та ігрова валюта

Після тривалого користування грою, ігрова валюта може сягати значень в декілька мільярдів і більше. Для написання коду для гри використовувалася мова програмування C#, в якій всі числові типи мають обмеження на максимальне значення [38]. В результаті значення ігрової валюти досягне значення, яке не зможе поміститися у звичайний числовий тип. Тому було вирішено використовувати реалізацію великих чисел у вигляді структури BigInteger з бібліотеки System.Numerics [38].

Великі числа займають багато місця у текстових полях на ігровій сцені. Рішенням може бути зменшення розміру шрифту для числа. Але такий підхід зменшить читабельність числа. Тому було вирішено використати абревіатури для таких чисел як тисячі, мільйони, мільярди тощо [39]. Наприклад, для відображення мільярда використовується англійська літера B (billion) і число «3 456 000 000» буде відображатися як «3,456B». Для відображення тисячі використовується англійська літера K і число «1 345» буде відображатися як «1,345K». Використовуючи абревіатури, можна отримати числа, які матимуть однакові кількість символів, що покращує відображення чисел в ігровій сцені та загальне сприйняття числа гравцем. На рис. 3.6 зображені абревіатури для інших чисел.

Зображення числа з використанням абревіатури використовується у всіх текстових полях, де відображається ігрова валюта для інкрементальної гри.

```
K = 1000,
M = 1000000,
B = 1000000000,
T = 1000000000000,
Qd = 1000000000000000,
Qr = 1000000000000000000
```

Рисунок 3.6 Абревіатури та відповідні числа

3.1.4 Кабінет гравця

Інформація про гравця, про поточного ігрового персонажа та про всіх ігрових персонажів, для яких гра закінчилася, відображається у кабінеті гравця. Після закінчення гри для певного ігрового персонажа, зберігається ігрова інформація, яка стосується даного ігрового персонажа як зображено на рис. 3.7.

Рисунок 3.7 – Список ігрових персонажів

Також в кабінеті гравця була додана можливість витрачати спільну між ігровими частинами ігрову валюту на ігрові предмети, які відображатимуться разом з ігровим персонажем. Отримані предмети прив’язуються до акаунта гравця,

а не до ігрового персонажа. Тому після закінчення гри для ігрового персонажа, отриманий предмет буде використовуватися разом з новим ігровим персонажем. Деякі ігрові предмети та ігровий предмет разом з персонажем зображені на рис. 3.8.

Ігрові предмети виконують кільки косметичну роль, щоб додати до ігрового персонажа іншого вигляду. З розвитком проекту до ігрових предметів будуть додані ігрові ефекти, які будуть впливати на ігровий процес.

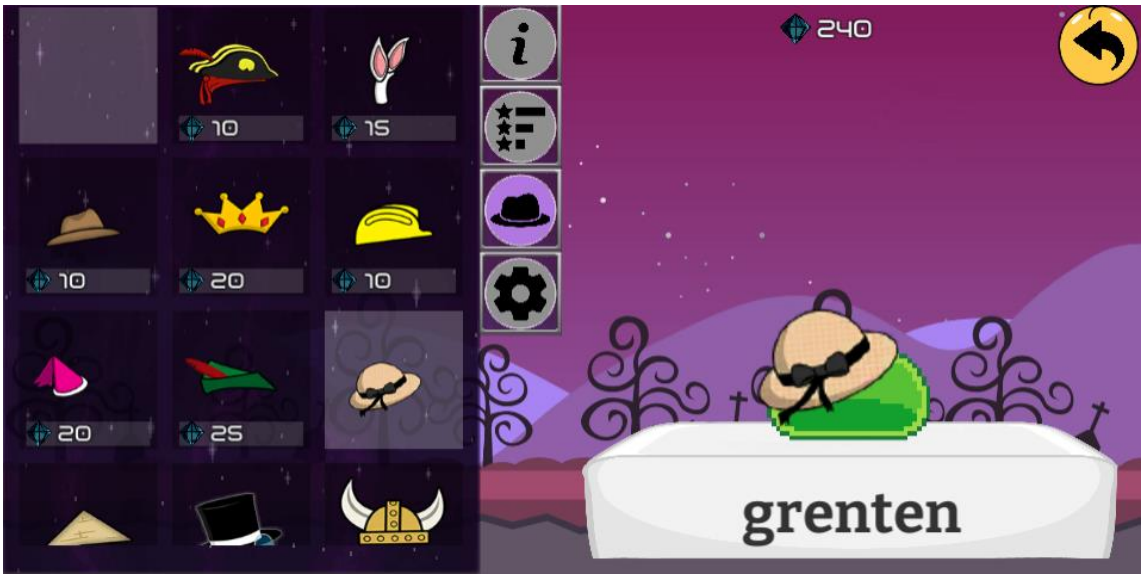


Рисунок 3.8 – Ігрові предмети та ігровий персонаж

3.2 Розробка частини гри з жанром стратегія

Дана частина гри використовує спільну ігрову валюту, ігрові предмети та ігрових персонажів з інкрементальною грою. Отримання нових предметів, зміна персонажів в одному з режимів буде змінювати предмети та персонажів в іншому. Отримання спільної ігрової валюти в першій частині гри здійснюється шляхом підвищення рівня гравця, а в другій частині гри – шляхом виграшів проти інших гравців. Ідея стратегії в другій частині гри подібна до ідеї всесвітньо відомої гри «Камінь, ножиці, папір» [40], але із зміненими правилами та ігровою механікою. Двоє гравців повинні обрати одну з трьох сторін для атаки (верх, середина та низ) та аналогічно обрати сторону для захисту. Якщо атакуюча сторона першого гравця

відмінна від захищаючої сторони другого, то другий гравець втратить певний рівень здоров’я, в іншому випадку втрата не відбудеться.

3.2.1 Розробка основних частин стратегічної гри

Для стратегічної частини гри були розроблені наступні ігрові сцени: ігрове меню, кімната очікування іншого гравця, сцена для поєдинку, сцена із результатами поєдинку та сцена авторизації та реєстрації. Також для даної частини гри використовуються деякі сцени з інкрементальної гри.

Були додані два режими гри. У першому (глобальному) режимі противник автоматично знаходиться, враховуючи рівень та рейтинг обох гравців. Коли гравець хоче розпочати гру, спочатку для нього підбирається противник з ігрової сцени, але якщо противник не знаходиться (через велику відмінність між параметрами гравців), то гравець сам поміщається в ігрову чергу і очікує поки його знайдуть.

Для пошуку використовується максимальний рівень гравця, який був отриманий для різних ігрових персонажів та ігровий рейтинг. Оскільки використовуються два параметри для пошуку, то пошук противника зводиться до вирішення проблеми пошуку найближчої точки у двовимірній системі координат.

Для реалізації даного пошуку, був використаний алгоритм пошуку по радіусу, але з використанням наївного пошуку, замість використання квадрі- або R-дерева. Оскільки ігрова черга постійно збільшується або зменшується, то час і пам’ять, які будуть витрачатися на побудову дерева, будуть знижувати загальну ефективність алгоритму. Але за рахунок використання пошуку по радіусу, загальність складність алгоритму з використанням наївного пошуку буде невелика.

Після проведення поєдинку в першому режимі гри, переможець отримає спільну ігрову валюту та додатковий ігровий рейтинг, а гравець, який програв втратить певну частину ігрового рейтингу. Для розрахунку змін рейтингів гравців була використана формула для знаходження академічного рейтингу гравців у шахи [41].

У другому (приватному) режимі гравець створює приватну кімнату, до якої може приєднатися тільки той гравець, який отримає автоматично створений приватний ідентифікатор кімнати. Тобто гравець може обрати з ким він хоче почати поєдинок не враховуючи ігровий рівень та ігровий рейтинг. На рис. 3.9 зображена сцена очікування противника в приватному режимі стратегічної гри.

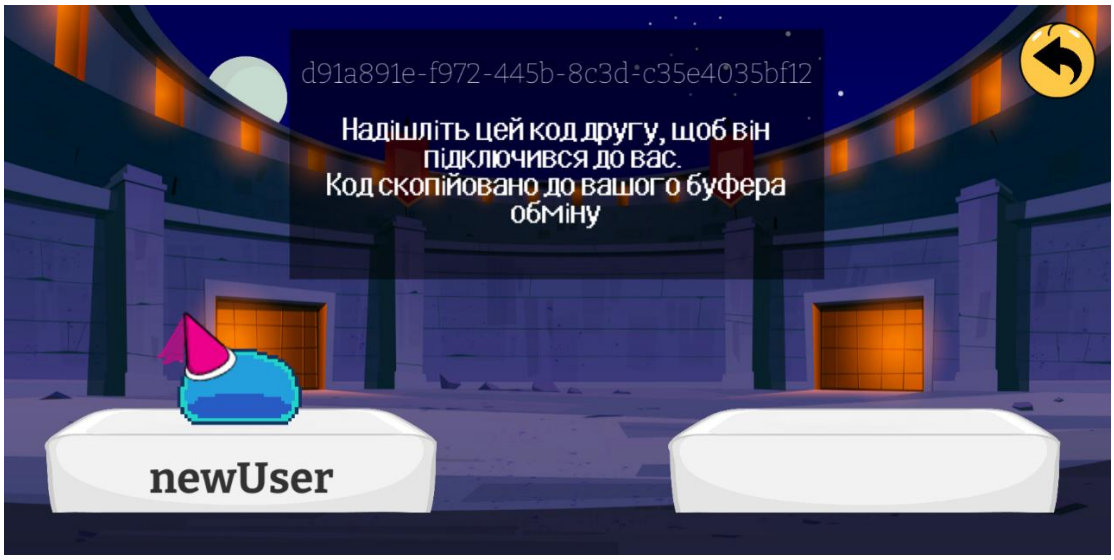


Рисунок 3.9 – кімната очікування противника

Також після поєдинку в даному режимі гри, було вирішено не видавати гравцям винагороди, щоб запобігти навмисному використанню даного режиму для швидкого підвищення ігрового рейтингу.

Згенерований ідентифікатор кімнати автоматично копіюється до буфера обміну використовуваного пристрою, щоб гравець мав змогу швидко передати ідентифікатор своєму потенційному противнику. Для передачі ідентифікатора приватної кімнати противнику використовуються сторонні інструменти та технології. Наприклад, месенджери чи соціальні мережі. З розвитком проекту механізм передачі ідентифікатора противнику буде вбудований в гру, щоб гравець не використовував сторонніх інструментів.

У ігровому меню було додане отримання списку гравців відсортованих за значенням ігрового рейтингу для відображення перших двадцяти п’яти гравців в загальному рейтингу. Також гравець має можливість отримати рейтинг гравців, який використовує максимальний рівень гравця для впорядкування елементів.

Рейтинг гравців зображений на рис. 3.10. Рейтинг перебудовується кожного разу як гравець вмикає сцену з ігровим меню, щоб врахувати зміни параметрів гравців, які могли відбутися за певний проміжок часу.

Рисунок 3.10 – Рейтинг гравців

Після знаходження противника в першому режимі або після підключення противника в другому, автоматично вмикається сцена для поєдинку, в якій гравець шляхом взаємодії з ігровим інтерфейсом, обирає сторону атаки та сторону захисту. Для пришвидшення вибору сторін, був доданий таймер, який відображається під час даного вибору. Після закінчення часу в таймері, буде обиратися сторона за замовчуванням, яка випадково обралася під час створення кімнати. Після кожного спрацьовування таймера буде обиратися одна і та сама сторона атаки та захисту. Даний механізм був реалізований у такому вигляді, для того, щоб гравець, який не обирає сторони, швидше втрачав рівень здоров’я. Вигляд таймера зображений на рис. 3.11.

Рисунок 3.11 – Ігровий таймер

На сцені для поєдинку відображається загальна інформація про двох гравців (ім’я гравця, ім’я ігрового персонажа, рейтинг та максимальний рівень). Також на сцені відображається поточний рівень здоров’я, який змінюється в залежності від обраних гравцями сторін.

Для відображення вибору певних сторін гравцями, були створені ігрові анімації, які відрізняються для кожної обраної сторони. Після вибору сторін двома гравцями, кожний гравець побачивши певну анімацію зрозуміє, яку сторону обрав його противник в цьому ході. Крім інтерполяції, зміни положення та розміру ігрового персонажа, також використовувався морфінг 2D об’єктів. На рис. 3.12.а та рис. 3.12.б зображені зміни ігрового персонажа з використанням морфінгу. Також анімації, які відтворюються під час атаки ігровим персонажем, автоматичну викликають анімацію отримання шкоди або одну з трьох анімацій захисту для ігрового персонажа противника. Поєднання анімацій атаки та захисту дозволяє зобразити взаємодію ігрових персонажів на ігровій сцені.

а)

б)

Рисунок 3.12 – Морфінг ігрового персонажа (зліва): а – початковий стан; б – стан після змінення форми

Також була додана обробка події закінчення гри, коли рівень здоров’я одного з гравців опуститься до нуля чи нижче. Перед завершенням гри, у випадку якщо був обраний глобальний режим, обраховуються зміни в рейтингах гравців і на сцені з результатами поєдинку відображаються отримані чи втрачені параметри гравця як зображено на рис. 3.13.а та рис. 3.13.б. Нове значення рейтингу гравця одразу буде враховуватися у наступному пошуку противника у глобальному режимі. Також зміни рейтингу будуть відображатися у списку гравців, якщо гравець матиме одне з найбільших значень рейтингу. Після поєдинку ігровий рівень не буде змінюватися. Щоб його підвищити, гравцеві потрібно взаємодіяти з першою ігровою частиною.

а) б)

Рисунок 3.13 – Результати поєдинку: а – виграш; б – програш

Якщо під час поєдинку, гравець вийде зі сцени з поєдинком, то він автоматично програє. Використовуючи можливості платформи Unity неможливо обробити подію виходу з додатка (подія, яка виникає, коли додаток примусово закривається). Тому для зарахування програшу та зниження рейтингу для гравця, який примусово вимкнув додаток під час поєдинку, був реалізований механізм збереження інформації про незавершений поєдинок. Якщо гравець не завершить поєдинок і примусово вимкне гру, то при наступному вході в гру, відкриється сцена з ігровими даними останнього поєдинку і гравець втратить рейтинг як після програшу. Даний механізм не дозволить гравцю уникнути зниження ігрового рейтингу.

3.2.2 Використання мережевих інструментів для взаємодії гравців

Взаємодія користувачів в даному режимі відбувається з використанням сервісів, які надає платформа Firebase, а саме використання Firebase Authentication та Firebase Realtime Database.

Була додана система акаунтів до створюваної гри. Акаунт спільний для двох ігрових частин, але для повноцінної взаємодії гравця з інкрементальною грою, необов'язково створювати ігровий акаунт. Для отримання доступу до другої ігрової частини, створення ігрового акаунту є обов'язковим. Для реєстрації нового користувача потрібні наступні дані: ім'я нового користувача, електронна адреса та пароль. Після створення ігрового акаунту, до сервісу Firebase Authentication буде доданий запис як зображено на рис. 3.14.

Identifier	Providers	Created ↓	Signed In	User UID
esquandor1337@gmail.com	✉	May 23, 2023	May 23, 2023	zarcZ3BlP9Vt44rvnglHdYJKhNg1
desktop@gmail.com	✉	May 23, 2023	May 27, 2023	Sd8fSlDXB8RR8YYvKl39B5C85fi2

Рисунок 3.14 – Список акаунтів на платформі Firebase

Доступ до проекту (БД та сервіс користувацьких акаунтів), який був створений на платформі Firebase надається тільки авторизованим користувачам, які є власниками проекту чи отримали доступ від власників проекту. Але навіть власники проекту не мають доступу до пароля користувача, який зареєструвався у грі. Сервіс Firebase Authentication шифрує отриманий при реєстрації пароль алгоритмом SCRYPT [42], тому пароль залишається захищеним навіть від власника проекту (розробника).

Після того як гравець обирає режим у стратегічній частині гри, до БД додається запис з інформацією про кімнату. Інформація про глобальні та приватні кімнати зберігається окремо. До інформації, яка додається до кімнати під час створення відноситься інформація про гравця (ім'я гравця, ім'я ігрового персонажа, тип ігрового персонажа, обраний ігровий предмет або його відсутність,

максимальний рівень та рейтинг) та про ігрові налаштування (максимальний рівень здоров’я гравців, значення сторін за замовчуванням для вибору сторони, коли гравець неактивний, інформація про гравця, який першим атакуватиме, час створення кімнати та час останньої зміни запису в БД гравцем, який створив кімнату). Запис інформації про кімнату з БД зображений на рис. 3.15.

Firebase Realtime Database використовує нереляційну БД [43], в якій замість таблиць використовуються документи.

Рисунок 3.15 – Інформація про кімнату

Час створення кімнати використовується для автоматичного видалення застарілих кімнат, інформація про які може залишитися в БД, якщо двоє гравців (противників) примусово вимкнуть гру. Час останньої зміни запису БД гравцем, який створив кімнату, використовується для перевірки активності даного гравця іншим гравцем (гравцем, який під’єднується до кімнати). Якщо час останньої зміни виявиться неактуальним (відмінність від поточного часу більше ніж 2-3 секунди), то гравець вважається неактивним і дані про кімнату автоматично видаляються із БД. В іншому випадку, гравець, який під’єднується, вносить власну інформацію (ім’я гравця, ім’я ігрового персонажа тощо) до БД і переходить до сцени з

поєдинком (якщо це глобальний режим) або до сцени очікування іншого гравця (якщо це приватний режим).

Піч час пошуку противника у глобальному режимі, гравець повинен отримати список всіх кімнат (гравців), які знаходяться в ігровій черзі. Отримавши кімнати, алгоритм пошуку по радіусу шукає противника з найближчими параметрами до гравця (максимальний рівень та рейтинг). Також при пошуку використовуються вагові коефіцієнти, щоб зменшити вплив максимального рівня на пошук і збільшити вплив рейтингу. Після знаходження противника з найближчими параметрами, гравець під’єднується до обраної кімнати використовуючи ідентифікатор, який прикріплений до знайденої інформації про противника. Якщо під час пошуку одного гравця, інший приєднається до кімнати з черги, то пошук може стати неактуальним і декілька гравців під’єднаються до однієї кімнати.

Для підтримання актуальності інформації було вирішено використовувати транзакцію [44] під час пошуку кімнати в ігровій черзі, щоб поки один гравець в рамках транзакції шукає ігрову кімнату, інші гравці будуть очікувати завершення пошуку, після якого обрана кімната буде видалена з ігрової черги. Оскільки транзакція блокує отримання або внесення змін в обраний документ (таблицю) в БД, то гравці, які вже під’єдналися до ігрової кімнати, не зможуть вносити взаємодіяти один з одним шляхом внесення змін в БД. Тому було вирішено зберігати основну інформацію про кімнату в окремому документі, щоб під час пошуку кімнати, транзакція не блокувала гравцям, які вже почали поєдинок, вносити зміни в БД.

Після знаходження противника, запис про кімнату видаляється з черги в БД, щоб інші гравці не розглядали дану кімнату під час пошуку. Також перевіряється активність гравця, який створив кімнату. Якщо гравець виявляється неактивним, то дані про кімнату видаляються з БД і пошук повторюється.

Якщо противник не був знайдений (велика відмінність між ігровими параметрами гравців або черга порожня), то гравець створює власну ігрову кімнату і додає до черги у БД інформацію про неї. Після того, як інший гравець знайде в

черзі дану кімнату і додасть до неї власну інформацію, гравець, який створив кімнату отримає відповідь від БД і також перейде до сцени з поєдинком.

Для того, щоб постійно не перевіряти чи до БД були внесені зміни, використовується патерн Спостерігач [3], реалізація якого доступна з використанням мови програмування C#. У сервісах, які надає Firebase Realtime Database даний патерн теж реалізований. Тому замість перевірки наявності змін в БД, БД самостійно інформує пристрій користувача про зміни в момент їх внесення. Даний підхід зменшує кількість постійних безрезультатних перевірок змін в БД.

Після початку поєдинку кожний гравець обирає одну з трьох сторін для атаки та захисту і вносить дану інформацію до відповідного запису про кімнату в БД. Користуючись патерном Спостерігач, було реалізоване отримання одним гравцем внесеної інформації про вибір сторони іншого гравця. Після отримання інформації про противника починається поєдинок, який зображується з використанням анімацій. Після завершення поєдинку кожний гравець вносить відповідний запис до БД про отримання попередньої інформації. Якщо гравець за обмежений проміжок часу не обере сторони атаки та захисту і пристрій гравця не відправить до БД інформацію про вибір сторони, то на пристрої противника для даного гравця обереться сторона за замовчуванням, яка була додана до інформації про кімнату під час її створення.

Кожного разу, коли гравець не обирає сторони для атаки та захисту, час на очікування відповіді другою стороною буде зменшуватися у два рази, оскільки гравець може примусово вимкнути гру або у гравця зникне підключення до інтернету. Тому інший гравець буде витрачати багато часу на очікування відповіді, яка не надійде. Якщо гравець власноруч обере сторони для атаки та захисту, то час очікування повернеться до норми. Подібний підхід використовується у комп’ютерних мережах при реалізації надійного протоколу передачі даних на транспортному рівні [45].

Аналогічно побудований механізм, який очікує відповідь від пристрою противника про успішне отримання обраних сторін та проведення поєдинку.

Під час поєдинку гравець може вийти. Щоб уникнути випадкового виходу зі сцени було додане діалогове вікно з підтвердженням виходу зображено на рис. 3.16. Якщо гравець вийде використавши відповідний елемент на сцені, то інформація про кімнату, до якої він був приєднаний видалиться. Інший гравець отримує відповідь від БД про видалення кімнати і розцінить це як програш гравця, який вийшов. Тому для другого гравця гра теж закінчиться, але він буде переможцем.

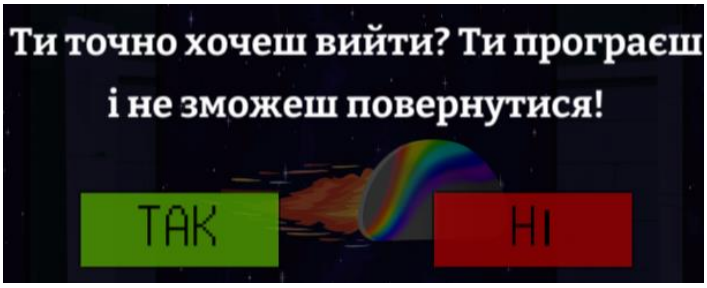


Рисунок 3.16 – Підтвердження виходу з поєдинку

3.3 Збереження та синхронізація даних з використанням локального сховища та бази даних

Уся інформація про гравця, ігрових персонажів, ігрових предметів та ігровий стан зберігається у локальному сховищі пристрою. Тому після вимкнення гри і повторному увімкненню результат попередньої взаємодії гравця з грою залишиться.

Після реєстрації чи авторизації гравця, дані крім локального сховища, зберігатимуться у БД, доступ до якої надає сервіс Firebase Firestore. Але дані у інкрементальній гри змінюються після кожного натискання на ігрові елементи. Постійне збереження змін у БД (період зберігання менше однієї секунди, що можливо при постійному натисканні гравцем на ігрові об’єкти) зменшить ефективність її роботи. Тому для збереження даних у БД був реалізований механізм, який отримує дані з локального сховища (в локальне сховище дані вносяться кожного разу, як вони зміняться) і відправляє їх до БД. Даний механізм

відправляє дані через встановлені проміжки часу та не залежить від кількості натискань на ігрові об’єкти користувачем.

Також в локальному сховищі та БД зберігається дата останнього збереження. Під час увімкнення гри відбувається синхронізація локальних даних та даних з БД. Обираються новіші дані. Якщо новіші дані будуть у локальному сховищі, то вони відправляться з оновленою датою до БД. В іншому випадку, дані з БД будуть додані до локального сховища. Крім того, даний механізм дозволяє отримати доступ до даних акаунту гравцям, які втратять пристрій або замінять його.

Дані неавторизованих гравців не будуть синхронізуватися з БД, оскільки для синхронізації потрібний ідентифікатор користувача, який доступний тільки для авторизованих користувачів. Також у БД прописані правила доступу, які зображені на рис. 3.17. Дані правила дозволяють вносити зміни тільки авторизованому користувачу у якого ідентифікатор збігається з назвою документа, в який вносяться зміни. Тому якщо користувач отримає іншим шляхом ідентифікатор (наприклад, якщо ідентифікатор буде зберігатися у локальному сховищі) але не буде авторизований, то зміни в БД не будуть внесені через відмову у доступі.

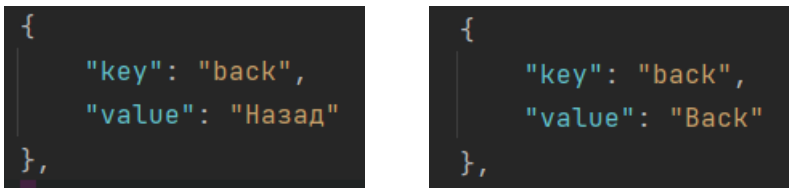
```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow update, write: if request.auth != null && request.auth.uid == userId;

    }
    match/users/{document=**}{
      allow read: if true;
    }
```

Рисунок 3.17 – Правила доступу до БД Firebase Firestore

Дані нового користувача, який не авторизувався, будуть зберігатися тільки в локальному сховищі. Але після успішної реєстрації, дані з локального сховища будуть надіслані до БД. На випадок авторизації гравця на пристрої, в локальному сховищі якого збережені дані іншого неавторизованого користувача, був доданий механізм обробки конфліктів. Користувач зможе перенести всі дані з акаунту на пристрій, перенести всі дані з пристрою до акаунта або об’єднати дані з акаунту та пристрою.

3.4 Локалізація

Одночасно з розробкою ігрових частин, усі слова, які відображаються для користувача, додавалися у два файли з форматом JSON (JavaScript Object Notation, текстовий формат обміну даними). Структура файлів зображена на рис. 3.18.а та рис. 3.18.б. До гри була додана локалізація на українську та англійську мови. До оперативної пам’яті пристрою завантажуються тільки файл із обраною мовою, що дозволяє заощадити оперативну пам’ять пристрою. Після обробки даний файл зберігається у вигляді словника (структура даних, яка зберігає разом з об’єктом ключ, використовуючи який, можна отримати об’єкт), що дозволяє швидко отримувати потрібні слова та речення використовуючи ключі, які спільні для кожної доданої мови. До текстових елементів додавався не текст, який потрібно відображати, а ключ, за яким потрібно отримати значення слова чи речення для обраної мови.



а) б)

Рисунок 3.18 – Ключ та значення у файлі: а – з українським перекладом; б – з англійським перекладом

Локалізація була реалізована без прив’язки до кількості мов, які будуть використовуватися у грі. Тому для додавання локалізації на іншу мову, достатньо створити файл з ключами з попередніх файлів та новими значеннями (перекладом) для обраної мови.

РОЗДІЛ 4 ПОРІВНЯННЯ ФУНКЦІОНУВАННЯ ГРИ ДЛЯ РІЗНИХ АПАРАТНИХ ПЛАТФОРМ ТА ОПЕРАЦІЙНИХ СИСТЕМ

Розроблена гра може використовуватися на мобільних пристроях (Android, iOS) та комп'ютерах (Windows, OS X, Linux). Для кожної ОС гра не буде відрізнятися. Але відмінності будуть для апаратних платформ, оскільки на мобільних пристроях відсутня кнопка для виходу з додатка, а на комп'ютерах вона обов'язкова. Для відмінного відображення ігрового об'єкта на сцені, був доданий механізм, який залежно від апаратної платформи пристрою користувача, відображає чи приховує певні ігрові елементи. Якщо апаратна платформа це мобільний пристрій, то ігровий об'єкт, який відповідає за вихід з гри, приховується від користувача. В іншому випадку він буде відображатися на сцені.

Також відмінності будуть у локалізації пристрою, оскільки немає універсального способу читання локального файлу на мобільних пристроях та комп'ютерах з використанням мови програмування C#. На мобільних пристроях використовується бібліотека UnityEngine.Networking та клас UnityWebRequest. Для комп'ютерів використовується бібліотека System.IO та клас File. Тому перевірка на тип апаратної платформи, була додана також під час читання файлу для локалізації.

Розроблена гра була протестована на наступних пристроях: ноутбук ASUS VivoBook з ОС Windows, мобільний телефон Samsung Galaxy A50 з ОС Android та планшет Pixus Hammer з ОС Android. Відсутність пристроїв з іншими ОС (OS X, Linux, iOS) обумовила неможливість проведення тестування гри на інших платформах.

На рис. 4.1.а, рис. 4.1.б та рис. 4.1.в зображені ігрові сцени для різних пристроїв. Комп'ютер, мобільний телефон та планшет відповідно. Усі ігрові елементи змінили розмір та положення залежно від розмірів дисплея пристроїв. Всі функції та можливості інкрементальної гри працюють належним чином на усіх пристроях. На кожному пристрої гравець шляхом натискання на ігровий екран отримує ігрову валюту. Автоматичне натискання для отримання ігрової валюти та

ігрові ефекти також функціонують належним чином. Ігрові дані зберігаються в локальному сховищі після взаємодії гравця з інтерфейсом гри. Зміна мови інтерфейсу гри відбувається на всіх пристроях.

а)

б)

в)

Рисунок 4.1 Основна сцена інкрементальної гри на: а – комп’ютері; б – мобільному телефоні; в – планшеті

Також належним чином працює реєстрація та авторизація користувача з використанням сервісу Firebase Authentication, синхронізація ігрових та користувацьких даних з БД з використанням сервісу Firebase Firestore та взаємодія гравців у стратегічній частині гри з використанням сервісу Firebase Realtime Database. На всіх пристроях був здійснений вхід в спільний акаунт і на кожному пристрої успішно була отримана інформація з БД.

Оскільки платформа Firebase та сервіси, які вона надає (Firebase Authentication, Firebase Firestore та Firebase Firestore) доступні на різних платформах, користувачі комп’ютерів можуть взаємодіяти із користувачами мобільних пристроїв в онлайн режимі використовуючи Firebase. Тип апаратної платформи чи ОС не будуть впливати на взаємодію користувачів.

ВИСНОВКИ

У роботі були досліджені інструменти для створення кросплатформних ігор Unreal Engine, Cocos2d-x, Godot Engine та Unity. Були проаналізовані особливості кожної платформи, а також переваги та недоліки. Після розгляду всіх досліджуваних платформ, було вирішено використовувати платформу для розробки кросплатформних ігор Unity.

Також було досліджено платформу для розробки онлайн застосунків Firebase. Сервіси, які надає дана платформа, можуть використовуватися разом з платформою Unity та функціонують на багатьох апаратних платформах та ОС.

Були розглянуті деякі математичні алгоритми, які використовуються у іграх. Алгоритми пошуку для знаходження ігрових об’єктів та знаходження противника для гравця. 2D та 3D морфінг об’єктів для створення плавних анімацій. Інтерполяція кривих для створення руху ігрових об’єктів вздовж кривої лінії та інтерполяція площин для створення поверхонь в 3D графіці.

У ході виконання роботи була розроблена кросплатформна багатожанрова гра, яка поєднує два жанри ігор: інкрементальну гру та стратегію. Розроблені ігрові частини використовують спільних ігрових персонажів та спільну ігрову валюту.

Дана гра доступна на ОС Windows, Linux, OS X, Android та iOS. Тестування функціонування гри були проведені на ОС Windows та Android та на апаратних платформах комп’ютер, мобільний телефон та планшет.

Розвиток проекту може відбуватися в декількох напрямках. Наприклад, проект побудований так, щоб можна було додати новий жанр (частину) гри без змінення раніше написаного коду, тобто проект може розширюватися без зайвих проблем. Також проект може розвиватися шляхом покращенням вже доданих ігрових частин. Наприклад, можна додати ігрові ефекти до різних ігрових предметів та ігрових персонажів, розширити можливості інкрементальної та стратегічної частин гри або додати нові ігрові режими до другої частини гри.

Створений проект може розглядатися як приклад кросплатформної гри написаної на платформі Unity з використанням багатоплатформного інструменту Firebase, який раніше був доступний тільки на мобільних пристроях, якщо вони

були створені з використанням платформи Unity. Також певні алгоритмічні рішення в проекті можна використовувати для створення інших ігор чи додатків. Наприклад, механізм створення ігрової черги в БД та пошуку противника з найближчими параметрами до гравця, використовуючи транзакцію та окремі документи (таблиці) в БД для збереження черги та інформації про кімнату.

Код та проект з платформи Unity розміщені у GitHub репозиторії [46]. Unity проект був скомпільований у наступні виконувані файли: .apk (для пристроїв з ОС Android), .exe (для пристроїв з ОС Windows), .x86_64 (для пристроїв з ОС Linux) та .app (для пристроїв з ОС OS X). Скомпілювати проект у виконуваний файл для ОС iOS не було можливості, оскільки дана компіляція вимагає наявність ОС OS X на пристрої, який виконує компіляцію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ЩО ТАКЕ FIREBASE? [Електронний ресурс] – Режим доступу до ресурсу: https://avada-media.ua/ua/services/firebase/.
2. Add Firebase to your Unity project – Set up a desktop workflow (beta) [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/unity/setup.
3. Design Patterns: Elements of Reusable Object-Oriented Software / E.Gamma, R. Helm, R. Johnson, J. Vlissides., 1994. – 416 с.
4. What is Unity? Everything you need to know [Електронний ресурс] – Режим доступу до ресурсу: https://www.androidauthority.com/what-is-unity-1131558/.
5. Nesteruk D. Rider Succinctly / Dmitri Nesteruk. – Morrisville, 2018. – 84 с.
6. Statcounter. Частка ринку настільних комп'ютерів, мобільних телефонів та планшетів у світі [Електронний ресурс] / Statcounter – Режим доступу до ресурсу: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#yearly-2020-2023-bar.
7. Statcounter. Частка ринку операційних систем у світі [Електронний ресурс] / Statcounter – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share#yearly-2020-2023-bar.
8. Galuzin A. UE4 Beginner's Crash Course: How to Start Learning and Using Unreal Engine 4 / Alex Galuzin., 2015. – 152 с.
9. Level Editor in Unreal Engine | Unreal Engine 5.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/5.0/en-US/level-editor-in-unreal-engine/.
10. Das A. Top 12 Unreal Engine 5 Games to Watch Out For [Електронний ресурс] / Ankush Das – Режим доступу до ресурсу: https://geekflare.com/unreal-engine-5-games/.
11. Build Multi-Platform Video Games - Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу: https://www.unrealengine.com/en-US/solutions/games.

12. Choosing Unreal Engine for Mobile Games: Things to Consider [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://game-ace.com/blog/unreal-engine-for-mobile-games/.

13. Editor interface overview [Електронний ресурс] – Режим доступу до ресурсу: https://docs.cocos.com/creator/2.4/manual/en/getting-started/basics/editor-overview.html.

14. Cocos2d - Advantage & Disadvantage [Електронний ресурс] – Режим доступу до ресурсу: https://eternitech.com/technologies/cocos2d/.

15. Luke. Top 10 Games Built With Cocos In 2021 [Електронний ресурс] / Luke. – 2021. – Режим доступу до ресурсу: https://www.cocos.com/en/post/top-10-games-built-with-cocos-in-2021.

16. Schardon L. What is Godot? The Free Engine for Making 2D & 3D Games [Електронний ресурс] / Lindsay Schardon. – 2023. – Режим доступу до ресурсу: https://gamedevacademy.org/what-is-godot/.

17. Alessandrelli F. Godot Editor running in a web browser [Електронний ресурс] / Fabio Alessandrelli. – 2020. – Режим доступу до ресурсу: https://godotengine.org/article/godot-editor-running-web-browser/.

18. Bradfield C. Godot Engine Game Development Projects / Chris Bradfield., 2018. – 298 с.

19. Showcase - Godot Engine [Електронний ресурс] – Режим доступу до ресурсу: https://godotengine.org/showcase/.

20. Unity 2019.3: Updates & improvements to Unity Editor workflows | New UI for Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/releases/2019-3/editor-tools.

21. Creating models for animation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/UsingHumanoidChars.html.

22. Jead. What platforms are supported by Unity? [Електронний ресурс] / Jead. – 2022. – Режим доступу до ресурсу: https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity-.

23. Що таке розширена реальність (XR): віртуальна (VR), доповнена (AR) і змішана (MR)? [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://gsminfo.com.ua/43036-shho-take-rozshyrena-realnist-xr-virtualna-vr-dopovnena-ar-i-zmishana-mr.html.

24. Olakunle O. 12 Best Unity IDE & editors for Faster Game Development [Електронний ресурс] / Olanrewaju Olakunle. – 2020. – Режим доступу до ресурсу: https://www.dunebook.com/best-unity-ide/.

25. Bring Your Ideas to Life with Unity Asset Store [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/pages/introduction-to-asset-store.

26. POTVIN J. 10 Best Games Developed In Unity, Ranked [Електронний ресурс] / JAMES POTVIN. – 2022. – Режим доступу до ресурсу: https://screenrant.com/best-games-developed-unity-ranked.

27. What is Firebase: Review, Pros and Cons, Alternatives | AltexSoft [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/.

28. Supported platforms, frameworks, libraries, and tools [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/libraries.

29. Millington I. Artificial Intelligence for Games / I. Millington, J. Funge., 2009. – 896 с.

30. Introduction to Algorithms, fourth edition / T.H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein., 2022. – 1312 с.

31. Ericson C. Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) / Christer Ericson., 2004. – 632 с.

32. Computational Geometry: Algorithms and Applications / M.de Berg, O. Cheong, M. van Kreveld, M. Overmars., 2008. – 398 с.

33. Parent Ph.D. R. Computer Animation: Algorithms and Techniques / Rick Parent Ph.D., 2012. – 542 с.

34. Морфінг 3D об’єктів [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Megarekrut65/3d-object-morphing.

35. Computer Graphics: Principles and Practice / [J. Hughes, A. van Dam, M. McGuire та ін.]., 2013. – 1264 с.

36. Інтерполяція кривих та 3D площин [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Megarekrut65/MFCG-2022.

37. Fiadotau M. Incremental Games [Електронний ресурс] / Mikhail Fiadotau. – 2017. – Режим доступу до ресурсу: https://link.springer.com/referenceworkentry/10.1007/978-3-319-08234-9_192-1.

38. Harris A. Microsoft C# Programming: For the Absolute Beginner / Andy Harris., 2002. – 512 с.

39. How To Abbreviate Million, Billion and Thousands on a Resume [Електронний ресурс] – Режим доступу до ресурсу: https://resumeworded.com/how-to-abbreviate-resume-key-advice.

40. Гусєва С. Камінь, ножиці, папір: історія найпопулярнішого способу вирішення суперечок та секрети перемоги [Електронний ресурс] / Софія Гусєва. – 2021. – Режим доступу до ресурсу: https://fun.24tv.ua/kamin-nozhitsi-papir-istoriya-pravila-strategiyi-ostanni-novini_n1651159.

41. HOW TO CALCULATE THE SCHOLASTIC RATING [Електронний ресурс] – Режим доступу до ресурсу: https://chess-math.org/scholastic-rating.

42. Wagner L. (VERY) BASIC INTRO TO THE SCRYPT HASH [Електронний ресурс] / Lane Wagner. – 2020. – Режим доступу до ресурсу: https://blog.boot.dev/cryptography/very-basic-intro-to-the-scrypt-hash/.

43. Sullivan D. NoSQL for Mere Mortals / Dan Sullivan., 2015. – 542 с.

44. Garcia-Molina H. Database Systems: The Complete Book / H. Garcia-Molina, J. Ullman, J. Widom., 2008. – 1248 с.

45. Kurose J. Computer Networking: A Top-Down Approach / J. Kurose, K. Ross., 2016. – 864 с.

46. Репозиторій з проектом [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Megarekrut65/Magical-Slimes-Triple-Choice.

ДОДАТОК А

Наведені діаграми з платформи Statcounter показують ситуацію на ринку комп’ютерних пристроїв та операційних систем

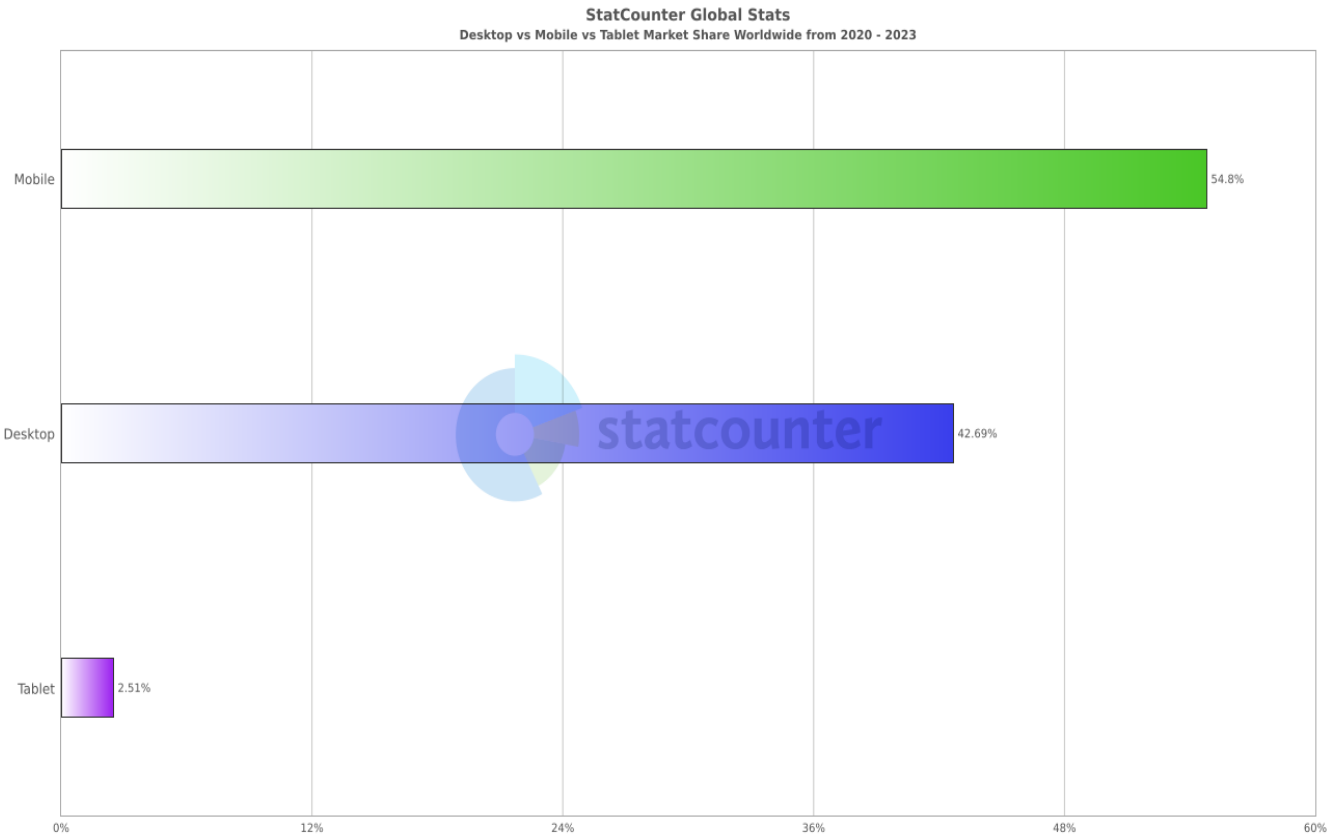


Рисунок А.1 – Частка ринку настільних комп'ютерів, мобільних телефонів та планшетів у світі на період 2020 – 2023

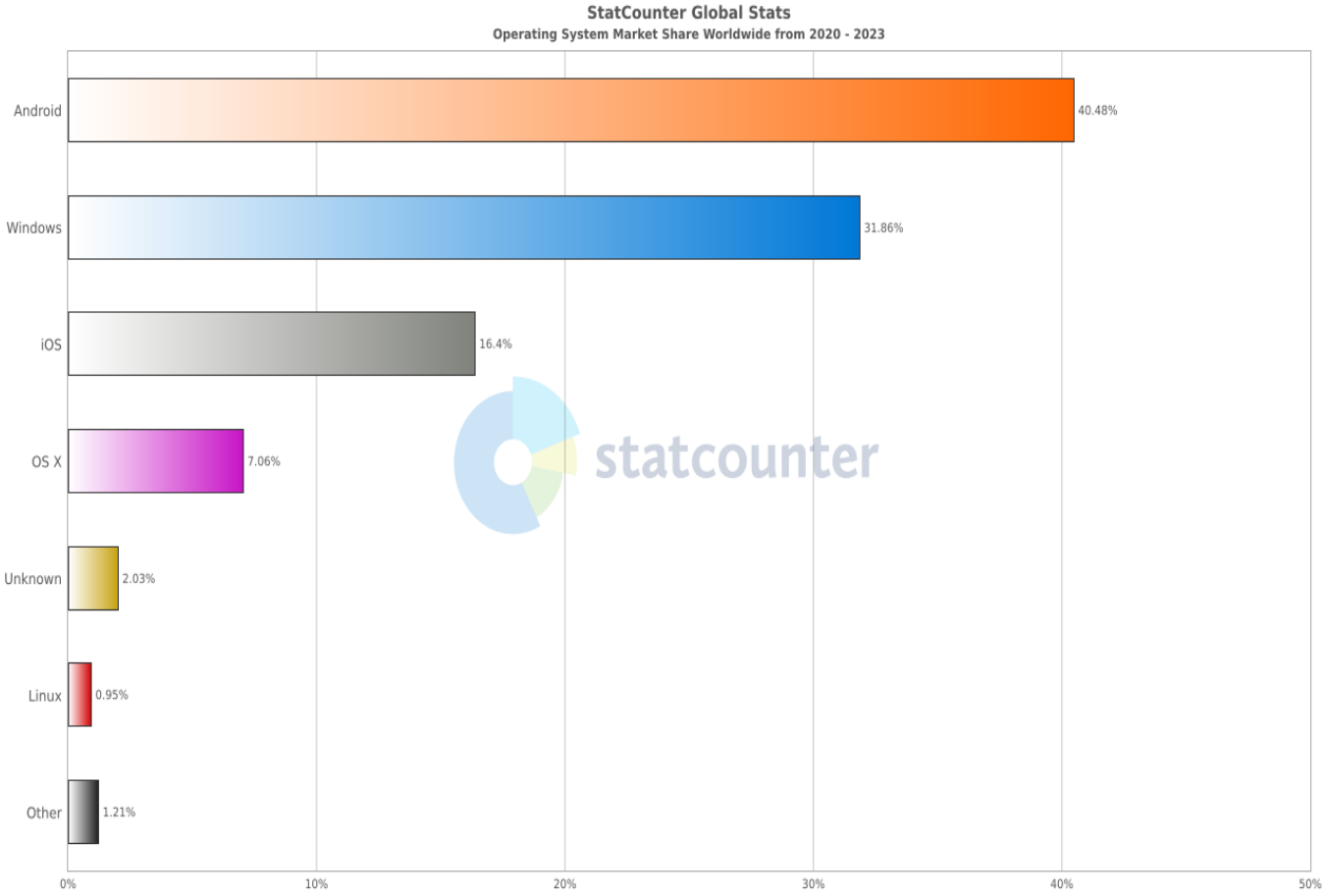


Рисунок А.2 – Частка ринку операційних систем у світі на період 2022 – 2023