МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Ігровий додаток Expulsion of Invaders»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-81 Слободяник Олександр Валентинович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою «___» _____2022 р.

Науковий керівник _____ к.т.н., доц., Федотова Н. А.

(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент_____

(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____В. В. Шендрик
«__» _____2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Слободяник Олександр Валентинович

1 Тема роботи Ігровий додаток «Expulsion of Invaders»

керівник роботи *Федотова Наталія Анатоліївна, к.т.н., доцент*,

затверджені наказом по університету від « 27 » квітня 2022 р. №0301-IV

2 Строк подання студентом роботи «10»червня2022 р.

3 Вхідні дані до роботи правила ігрового процесу, системні вимоги

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, аналіз предметної області, визначення актуальності дослідження, аналіз існуючих продуктів-аналогів, постановка задачі, моделювання та проектування, моделювання програмної реалізації, моделювання варіантів використання, проектування процесного алгоритму, розробка ігрового додатку, вибір програмних продуктів для розробки, програмна реалізація, тестування та працездатність, висновки, список літератури, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Федотова Н.А.*		
Моделювання та проектування	*Федотова Н.А.*		
Розробка ігрового додатку	*Федотова Н.А.*		

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	14.11.21	
2	Оформлення технічного завдання	25.12.21	
3	Аналіз предметної області	27.01.22	
4	Створення ідеї	04.02.22	
5	Розробка ігрового додатку	21.04.22	
6	Тестування	05.05.22	
7	Перевірка працездатності	19.05.22	
8	Реліз ігрового додатку	19.05.22	
9	Оформлення пояснювальної записки	10.06.22	

Студент _____ Слободяник О.В.
(підпис)

Керівник роботи _____ к.т.н., доц. Федотова Н.А..
(підпис)

РЕФЕРАТ

Тема дипломної роботи: «Ігровий додаток Expulsion of Invaders».

Пояснювальна записка дипломної роботи включає в себе: вступ, перший розділ «Аналіз предметної області», другий розділ «Моделювання та проектування», третій розділ «Розробка ігрового додатку», висновки, список використаних джерел, додаток А з технічним завданням, додаток Б з плануванням робіт, додаток В з програмними кодами до ігрового додатку.

Перший розділ «Аналіз предметної області» включає: визначення актуальності роботи, аналіз продуктів-аналогів та постановку задачі.

Другий розділ «Моделювання та проектування» включає: моделювання програмної реалізації, моделювання варіантів використання, проектування процесного алгоритму.

Третій розділ «Розробка ігрового додатку» включає вибір програмних продуктів для розробки, програмну реалізацію, тестування та працездатність.

Результатом дипломної роботи є готовий ігровий додаток «Expulsion of Invaders», реалізований на цифровому ринку Play Market.

Кваліфікаційна робота містить 75 сторінок, 2 таблиці, 36 рисунків, список літератури 18 найменувань, 3 додатки.

Ключові слова: МОБІЛЬНА ГРА, ДОДАТОК, ВІДЕОГРА, UNITY, PLAY MARKET, ДВОВИМІРНА АРКАДА.

ЗМІСТ

ВСТУП

Ігрова індустрія набирає все більше і більше обертів з кожним днем. За останній рік вона майже стала найприбутковішим бізнесом серед медіа-простору. Це приблизно 176 мільярдів доларів США [1].

Розробка відеоігор – це серйозний бізнес, але так було не завжди. Раніше, на початку 80-90-х років, створення їх вважалося елементом технологічного прогресу. Розроблювалася велика кількість ігрових додатків під різні платформи: прикладний комп'ютер, ігрові автомати, ігрові приставки. Кожен розробник намагався швидко обійти конкурентів, створюючи різноманітні жанри на різні платформи, комбінуючи елементи ігрових процесів різних ігор. Одним із таких жанрів стали двовимірні аркади.

Зараз кожна людина має смартфон, на який зручно завантажити декілька ігрових новинок, щоб розважитися та скоротати час за грою. Найпопулярнішою платформою для розробки відеоігор за останні роки стали, як раз, мобільні пристрої, а саме Android та iOS платформи. Таким чином, утворився мобільний геймінг. Чудовим прикладом є такі проекти, як «PUBG Mobile», «Hearthstone», «The Elder Scrolls: Blades» та інші.

Отже, метою даного дослідження є розробка ігрового додатку жанру «двовимірна аркада» під платформу Android на цікаву та актуальну тематику.

Для досягнення мети проекту необхідно виконати наступні задачі:

- дослідити предметну область та проаналізувати ігрові додатки-аналоги;
- підібрати або створити дизайн та звукову складову ігрового додатку;
- запрограмувати скриптову частину гри;
- виконати зборку відеогри у файл відповідно платформі;
- налаштувати Google-рекламу у створеній відеогрі;
- виконати тестування ігрового додатку;
- виконати реліз гри на цифровий ринок Play Market.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення актуальності дослідження

Відеогра – це програмний продукт з розважальною або навчальною метою, в якому користувач взаємодіє з ігровим процесом за допомогою інтерфейсу [2]. Всього на даному етапі технологічного прогресу існує три види графічної реалізації відеоігор: двовимірні, тривимірні та текстові. Також відеоігри подібно до музики та літератури розподіляють на жанри, основні з яких класифікують так: Action ігри, симулятори, пригодницькі ігри, стратегії, головоломки та RPG ігри [3]. Через швидкий розвиток ігрової індустрії з’являється багато нових жанрів, які базуються на комбінації декількох інших.

З розвитком ігрової індустрії, як і у всього нового, почали з’являтися супротивники. Набула поширення інформація про фізичну та психічну шкоду відеоігор для людини. Згодом почали проводитися дослідження науковців і шум навколо цієї теми затих. Але, як показав досвід, відеоігри не лише не завдавали шкоди людині, а й позитивно впливали на розвиток різних частин мозку. Головоломки розвивають розумові здібності та логіку, Action ігри вдосконалюють моторику та реакцію гравця, сюжетні ігри подібно книгам розповідають історію та інше [4].

Аркада – це жанр відеоігор, якому притаманний швидкий за часом, але інтенсивний ігровий процес (геймплей). У сучасності основою цього жанру стали аркадні ігрові автомати, які були популярні з кінця 1970-х до середини 1980-х років [5].

Мобільна платформа набула популярності для розробки ігрових додатків на початку 21-го століття. Протягом кількох десятків років ігрова індустрія у сфері мобільного геймінгу розвилася настільки, що деякі ігрові релізи на мобільні пристрої випереджають за кількістю гравців суміжні релізи під інші платформи.

Найпоширенішими мобільними платформами на даний час є Android та iOS – більше 3 мільярдів активних пристроїв та більше 1,65 мільярдів активних пристроїв відповідно [6]. Ми вирішили використовувати мобільну платформу Android тому, що

вона охоплює більшу частину користувачів. Також основними особливостями мобільних платформ загалом є наступні:

– популярність – кожна людина використовує мобільні пристрої у повсякденному житті [6];

– доступність – середній діапазон цін на пристрої не виходить за рамки мінімальної місячної зарплатні в Україні [7, 8];

– зручність – сенсорний екран заміняє клавіатуру та комп'ютерну мишу, тому для користуванням смартфоном не треба ніяких додаткових пристроїв;

– потужність – сучасні мобільні пристрої підтримують ігрові додатки різних обсягів та вимог.

З кожним кроком технологічного прогресу розробники ігрових додатків поступово відмовляються від створення окремих ігрових рушіїв для нових відеоігор. Саме через це використання середовищ розробки ігрових додатків стає популярнішим, розробникам не потрібно витрачати час на програмування нового ігрового рушія, треба лише налаштувати вже готовий. Лідерами ринку середовищ розробки ігрових додатків на даний момент часу є Unity та Unreal Engine. Кожен з цих рушіїв має власні особливості, такі як мови програмування, графічні можливості та інші. Середовище розробки Unity використовує мову програмування C# та сумісне з різними платформами, так як повністю кросплатформенний, тому саме Unity ми обрали як ігровий рушій нашої відеогри [9].

Серед різних жанрів відеоігор існують фактори впливу на успіх проекту. У нашому випадку, для двовимірної аркади такими факторами є наступні: тематика, складність, рейтинг серед гравців, візуальне та аудіо-візуальне оформлення. Саме поєднання цих факторів визначає скільки користувачів завантажить ігровий додаток, та на який час вони залишаться прихильними цьому продукту.

Тому, розробка нового ігрового додатку жанру аркада на актуальну тематику є важливим питанням для популяризації жанру та поширення відеоігор у цілому.

1.2 Аналіз існуючих продуктів-аналогів

Із моменту створення жанру двовимірних аркад була реалізована величезна кількість відеоігор цього напрямку, через це створити продукт з унікальним ігровим процесом цього жанру досить важко. Також за останні роки велика частина цих розробок торкнулася мобільних пристроїв. Тому розробникам залишається лише шукати актуальну тематику гри або створювати римейки вже існуючих ігор з власним доповненням.

Першим кроком у плануванні проекту було дослідження існуючих найсучасніших та водночас популярних аналогів відеоігор жанру аркада. Також головним критерієм під час досліду аналогів є платформа гри, тобто у нашому випадку це мобільна платформа Android. Об’єктами дослідження стали такі популярні ігрові додатки, як: «Duet», «Hyperforma» та «High Risers». Вони займають топ кращих жанру на цифровому ринку Play Market [10].

Ігровий додаток «Duet» від розробника «Kumobius Games» створений у мінімалістичному стилі, шрифт підібрано відповідно [11]. Також використовувалися нейтральні кольори оформлення: відтінки сірого, чорного та білого. Колір основного об’єкту, яким керує користувач гри, має контрастні синій та червоний кольори (рис. 1.1).

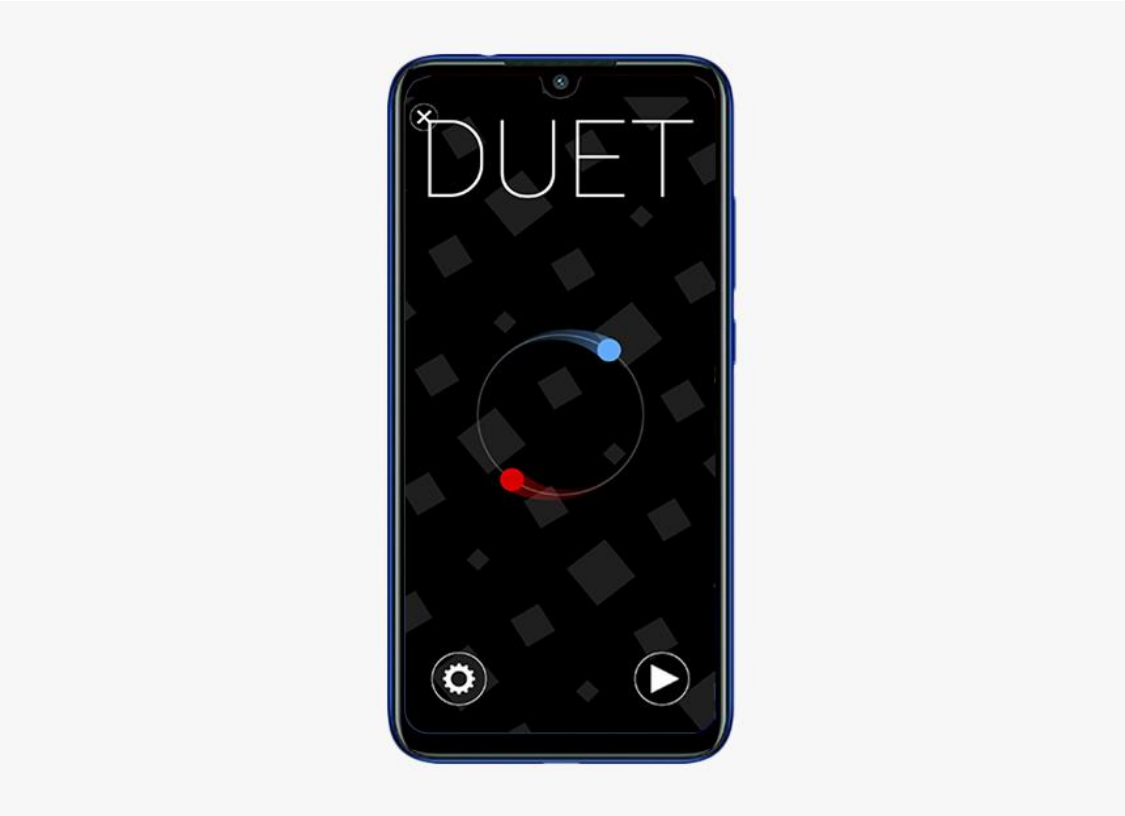


Рисунок 1.1 – Головне меню мобільної гри «Duet»

Сенс геймплею цього ігрового додатку полягає у тому, щоб ухилятися від різних ігрових об’єктів, які поступово переміщуються у напрямку гравця. Також ці об’єкти можуть бути різних розмірів, можуть бути динамічні або статичні. Гравець керує двома кулями різних кольорів, які треба переміщати відносно центру кола за годинниковою стрілкою або проти неї так, щоб не доторкнутися інших ігрових об’єктів (рис. 1.2).

Рисунок 1.2 – Геймплей мобільної гри «Duet»

Якщо ці кулі торкаються якогось об’єкту, то гравець переміщується на початок рівня. Також цей ігровий додаток має декілька режимів: декілька платних режимів з проходженням рівнів, один безкоштовний сюжетний режим та безкоштовний нескінченний (рис. 1.3).

Ігровий додаток «Duet» не має рекламної інтеграції, але є платний контент, з якого розробник отримує прибуток.

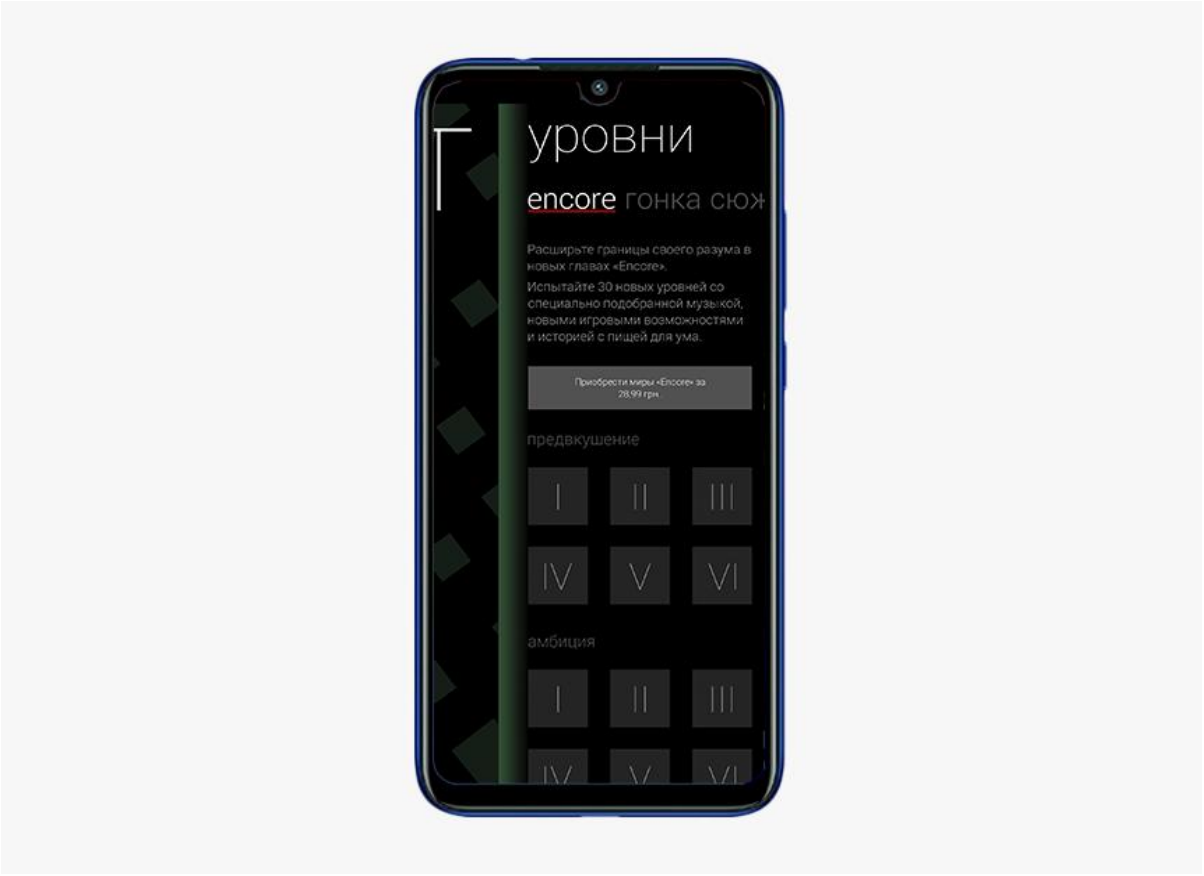


Рисунок 1.3 – Режими мобільної гри «Duet»

Наступним продуктом аналізу є відеогра «Hyperforma» від студії-розробника «HeroCraft Ltd» [12]. Цей ігровий додаток виконаний у футуристичному стилі з елементами мінімалізму. Основними кольорами оформлення використано відтінки червоного, помаранчевого та жовтого кольорів (рис. 1.4).

Рисунок 1.4 – Головне меню мобільної гри «Hyperforma»

Головною метою гри є проходження рівнів різної складності, руйнуючи червоні сфери, які оточені перешкодами. Деякі перешкоди, які представляють собою ігрові об’єкти, можна знищити. Під контролем гравця вогняний елемент, який він запускає за траєкторією. Врізаючись він знищує об’єкт, або можна також використати його для повторного пострілу, який спрямований векторно до центру червоної сфери (рис. 1.5).

Рисунок 1.5 – Ігровий процес мобільної гри «Hyperforma»

Сама гра має у собі лише один режим – це проходження рівнів, тобто змагання між гравцями відсутнє. Також відеогра має професійне аудіо супроводження, яке досить добре підходить до самого ігрового процесу.

Цей ігровий додаток не має інтегрованої реклами, а має платний контент. Також на цифровому ринку Play Market присутня преміум версія гри, яка включає у собі весь платний контент (рис. 1.6).

Рисунок 1.6 – Меню після проходження рівня мобільної гри «Hyperforma»

Останнім аналогом для порівняння є «High Risers» від розробника «Kumobius Games» [13]. Цей проект виконаний у піксельному стилі, з використанням оформлення з низько-полігональними текстурами. Все меню гри зроблено білим кольором на фоні сцени початку гри (рис. 1.7).

Рисунок 1.7 – Головне меню мобільної гри «High Risers»

Основою ігрового процесу цієї гри є збирання монет шляхом нескінченного руху вгору з перешкодами. Гравець керує лише стрибками персонажу, а сам він постійно переміщується, відбиваючись від ігрових об’єктів. Кожен пройдений поверх записується, а рекорди зберігаються на сервері. Якщо персонаж не влучає у ігровий об’єкт, то він падає та гра починається знову (рис. 1.8).

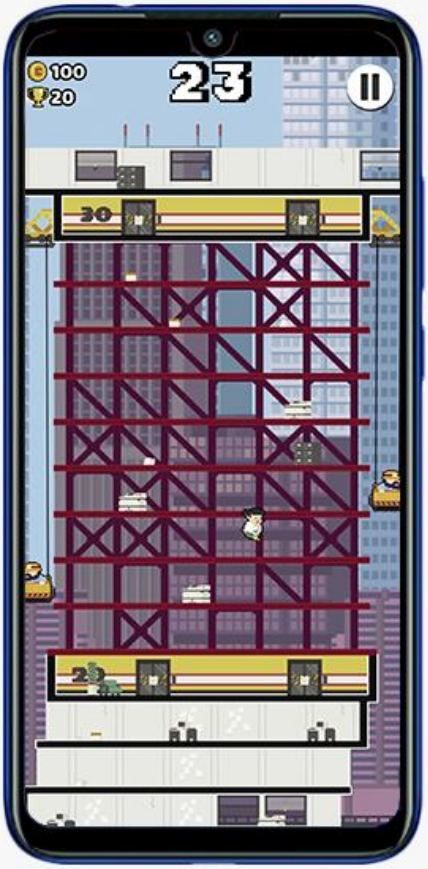


Рисунок 1.8 – Геймплей мобільної гри «High Risers»

Монетизація цього ігрового додатку полягає у продажі скінів персонажу та ігрової локації. Також користувач має можливість купити ігрові монети за реальну валюту. Інтегрована реклама відсутня (рис. 1.9).

Після детального аналізу кожного аналогу ігрового додатку, було визначено їх переваги та недоліки. Результати порівняння представлені в таблиці 1.1.

Рисунок 1.9 – Способи монетизації мобільної гри «High Risers»

Далі приведена таблиця 1.1 порівняння різних особливостей та спільних рис продуктів-аналогів, які були проаналізовані.

Дані з таблиці 1.1 надають змогу звернути увагу на різні переваги і недоліки проаналізованих аналогів, які можна використати, або усунути з розроблюваного проекту. Створюваний ігровий додаток повинен мати двовимірну реалізацію, якісний музикальний супровід, рейтинг серед гравців, інтегровану рекламу та простоту управління.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів ігрових додатків

Мобільний ігровий додаток / **Характеристика**	***«Duet»***	***«Hyperforma»***	***«High Risers»***
Двовимірна реалізація	+	+	+
Різноманітність геймплею	+	-	-
Якісний музикальний супровід	+	+	-
Рейтинг серед гравців	+	-	+
Наявність інтегрованої реклами	-	-	-
Наявність скінів	-	-	+
Зручність і простота управління	+	-	+
Мальовничий дизайн	-	+	-

1.3 Постановка задачі

Основною метою даного дослідження є розробка ігрового додатку під назвою «Expulsion of invaders» жанру двовимірної аркади для мобільної платформи Android на цікаву або актуальну тематику на базі ігрового рушію Unity.

Основні вимоги до створюваного ігрового додатку є наступними:

– якісний музикальний супровід;

– наявність рейтингу серед гравців;

– інтегрована реклама;

– простота управління;

– стриманий дизайн.

Для досягнення мети проекту необхідно виконати наступні задачі:

– охарактеризувати актуальність теми, дослідити предметну область та проаналізувати ігрові додатки-аналоги;

- створити ідею;
- підібрати або створити дизайну гри;
- створити звукову складову ігрового додатку;
- запрограмувати скриптової частини гри;
- розробити дизайн інтерфейсу користувача;
- розробити дизайн головного меню та інших складових ігрового додатку;
- виконати зборку відеогри у файл відповідно платформі;
- налаштувати Google-рекламу у створеній відеогрі;
- виконати тестування ігрового додатку;
- виконати реліз гри на цифровий ринок Play Market.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Моделювання програмної реалізації

Графічним представленням методології функціонального моделювання називають контекстну діаграму IDEF0. Такий метод аналізу та документації бізнес-процесів був розроблений у середині 1970-х років для використання серед виробничих галузей. З технологічним розвитком та появою електронно-обчислювальних машин напрям використання контекстних діаграм IDEF0 розширився. Сьогодні їх використовують майже у всіх сферах життєдіяльності людини, насамперед, у світі інформаційних технологій [14].

Для графічного представлення систем використовуються: функції, вхідні дані, вихідні дані, механізми та об’єкти управління. Алгоритм діаграми полягає у тому, що функція отримує вхідні дані, а управління за допомогою механізмів створює умови для отримання очікуваних вихідних даних. Також ці процеси розбиваються на дрібні підпроцеси, представлення яких називають декомпозицією. Діаграма IDEF0 складається з блоків та стрілок, які використовуються для побудови за таким принципом:

– Блок виконує роль функції;
– Стрілки, які входять ліворуч до блоку, виконують роль вхідних даних;
– Стрілки, які входять зверху до блоку, виконують роль управління;
– Стрілки, які входять знизу до блоку, виконують роль механізмів;
– Стрілки, які виходять праворуч від блоку, виконують роль вихідних даних [15].

Для створення функціональної моделі ігрового додатку використовувався сервіс draw.io, в якому можна власноруч створити та конфігурувати вигляд діаграми IDEF0 та її декомпозицій з погляду розробника та користувача. Маючи теоретичне представлення структури діаграми та головний процес, а саме «Реалізація ігрового

додатку Expulsion of Invaders», створюються такі параметри з погляду розробника рис.2.1 а:

– вхідні дані: технічне завдання;

– управління: правила ігрового процесу, системні вимоги;

– механізми: програмне забезпечення, апаратне забезпечення, розробник, тестувальник;

– вихідні дані: готовий ігровий додаток.

Та з погляду користувача рис.2.1 б:

– вхідні дані: пристрої вводу;

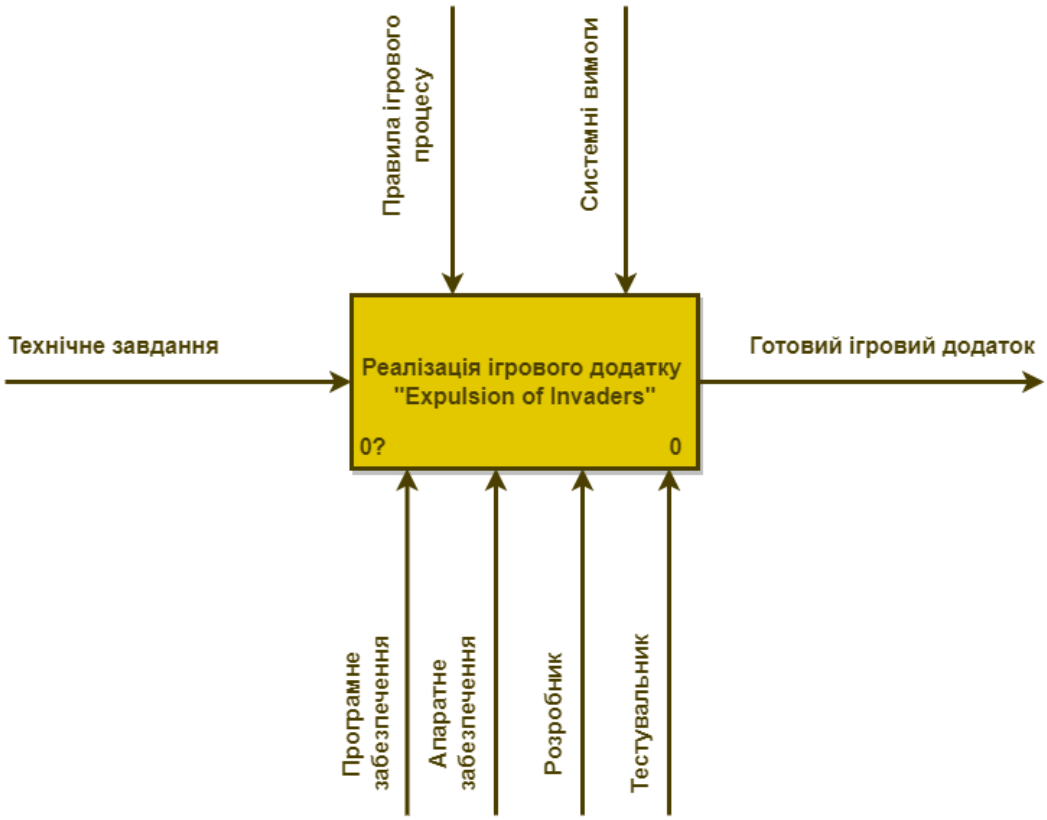
– управління: правила ігрового процесу, системні вимоги;

– механізми: програмне забезпечення, апаратне забезпечення, розробник, тестувальник;

– вихідні дані: результат гри.

На рисунку 2.1 зображена модель контекстної діаграми реалізації ігрового додатку «Expulsion of Invaders».

Далі здійснюємо розбиття створених контекстних діаграм на декілька підпроцесів. На рисунку 2.2 (а,б) зображена декомпозиція діаграм ігрового додатку «Expulsion of Invaders».

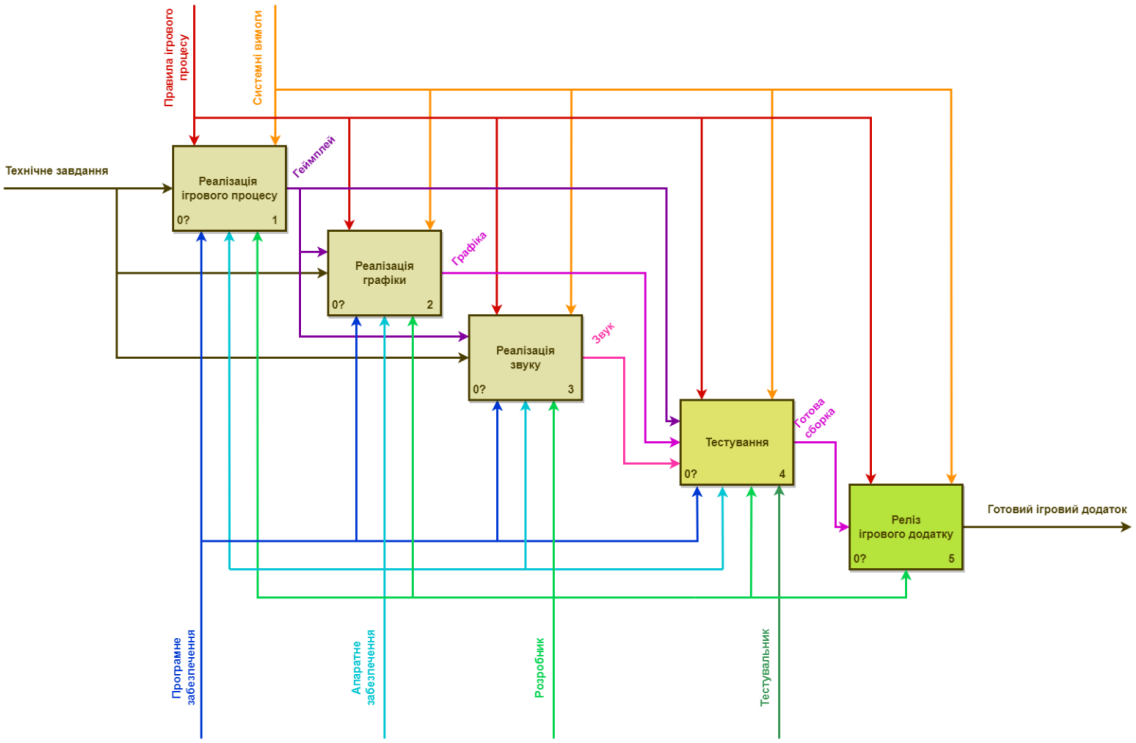


а)

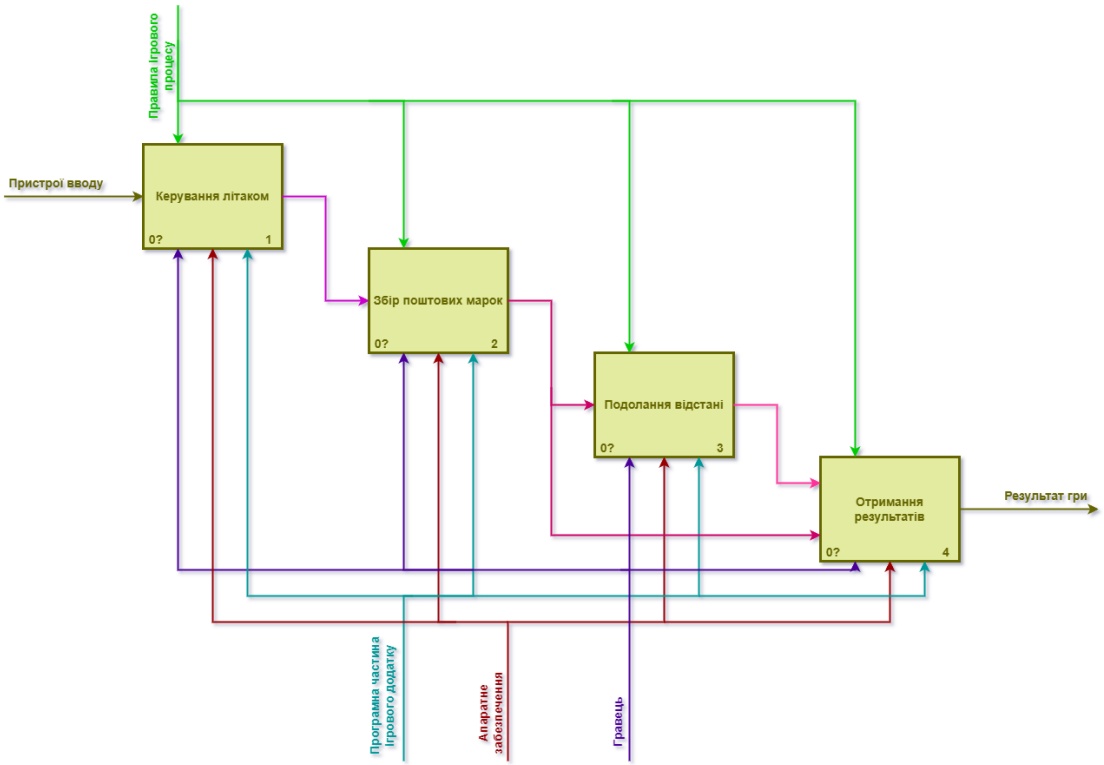
б)

Рисунок 2.1 – Контекстна діаграма IDEF0

(а – с погляду розробника, б – з погляду користувача)



а)



б)

Рисунок 2.2 – Декомпозиція контекстної діаграми IDEF0

(а – с погляду розробника, б – з погляду користувача)

2.2 Моделювання варіантів використання

Графічним представленням варіантів використання в системі називають діаграмою прецедентів UML. Діаграма варіантів використання описує послідовність дій системи у відповідь на зовнішні втручання актора. Актор – це суб'єкт керування системою, в ролі якого можуть бути як користувачі, так і будь-які системи. Також діаграма може доповнюватися інструкцією у вигляді текстового опису маніпуляцій, які виконує актор над системою [16].

Спираючись на технічне завдання, визначаємо актора та варіанти використання в системі ігрового додатку «Expulsion of Invaders».

Актори в системі:

- гравець-користувач

Варіанти використання в системі:

- *початок гри* – переходить на сцену гри та запускає ігровий процес додатку;
- *налаштування гри* – переходить на сцену налаштування параметрів гри;
- *список лідерів* – відкриває вікно Google Play Services з таблицями лідерів серед двох параметрів;
- *вихід зі гри* – виходить зі гри завершенням процесу ігрового додатку;
- *параметри звуку* – керує вмиканням та вимиканням звуку ігрового додатку;
- *параметри аккаунту* – керує входом або виходом з облікового запису Google Play;
- *параметри графіки* – керує рівнем графічної візуалізації ігрового додатку;
- *пауза* – призупиняє ігровий процес;
- *перезапуск* – перезавантажує ігровий процес спочатку;
- *вихід до головного меню* – переходить на сцену головного меню, завершуючи ігровий процес.

Діаграму прецедентів UML ігрового додатку «Expulsion of Invaders» зображено на рисунку 2.3.

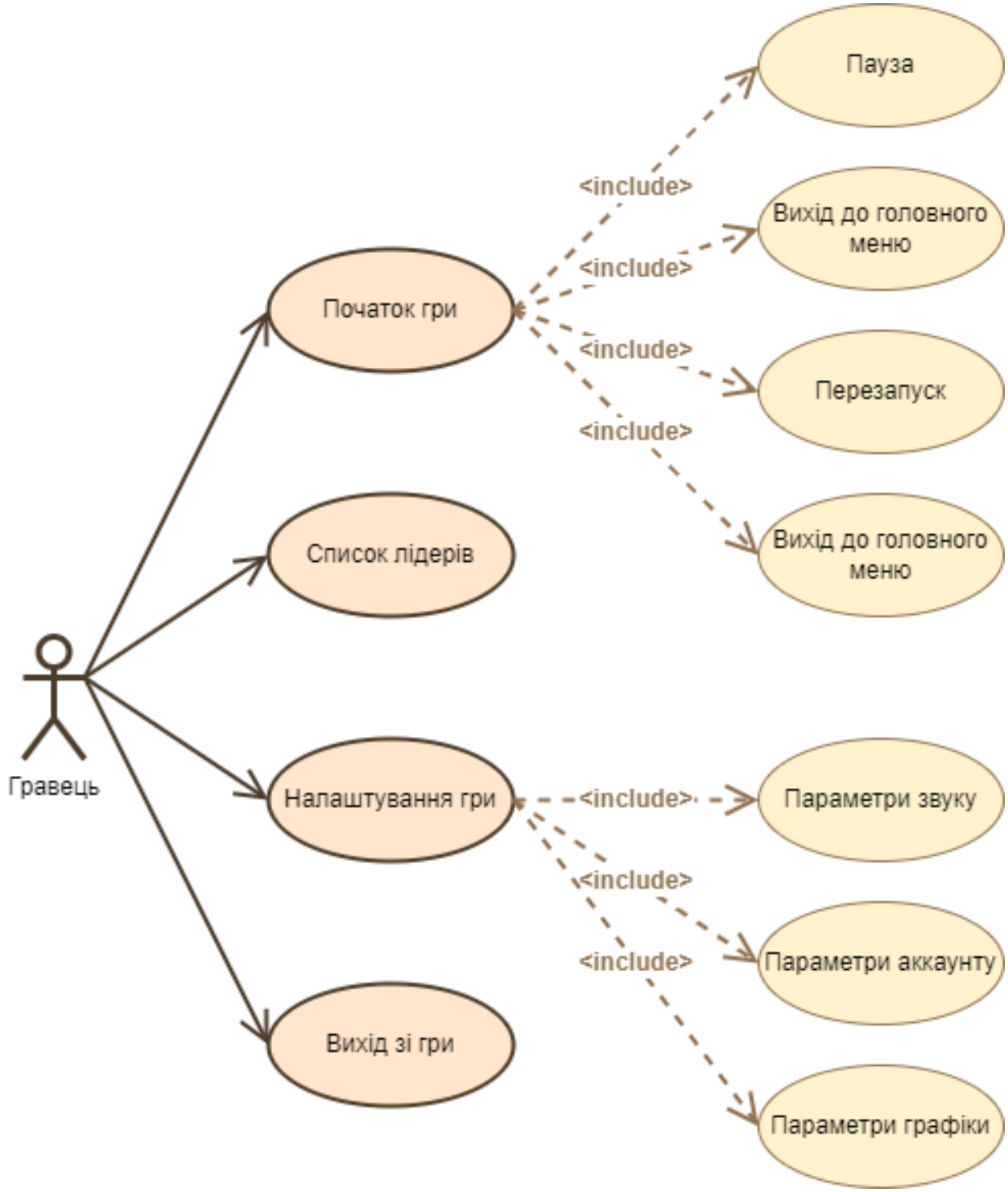


Рисунок 2.3 – Діаграма варіантів використання UML

2.3 Проектування процесного алгоритму

Графічне представлення процесного алгоритму називають блок-схемою. Головна мета блок-схеми – зображення послідовності у вигляді геометричних фігур та стрілок. Основними функціями блок-схеми визначають такі, як:

– початок та кінець – показує вхід та вихід у середовище;
– умова – показує рішення перемикального типу;
– функція – показує вхід іншої блок-схеми;
– ввід та вивід – показує форму обробки вводу та виводу;
– цикл – показує назву та параметри циклу;
– з’єднувач – показує перехід між частинами блок-схеми[17].

На рисунку 2.4 зображено блок-схему ігрового додатку «Expulsion of Invaders».

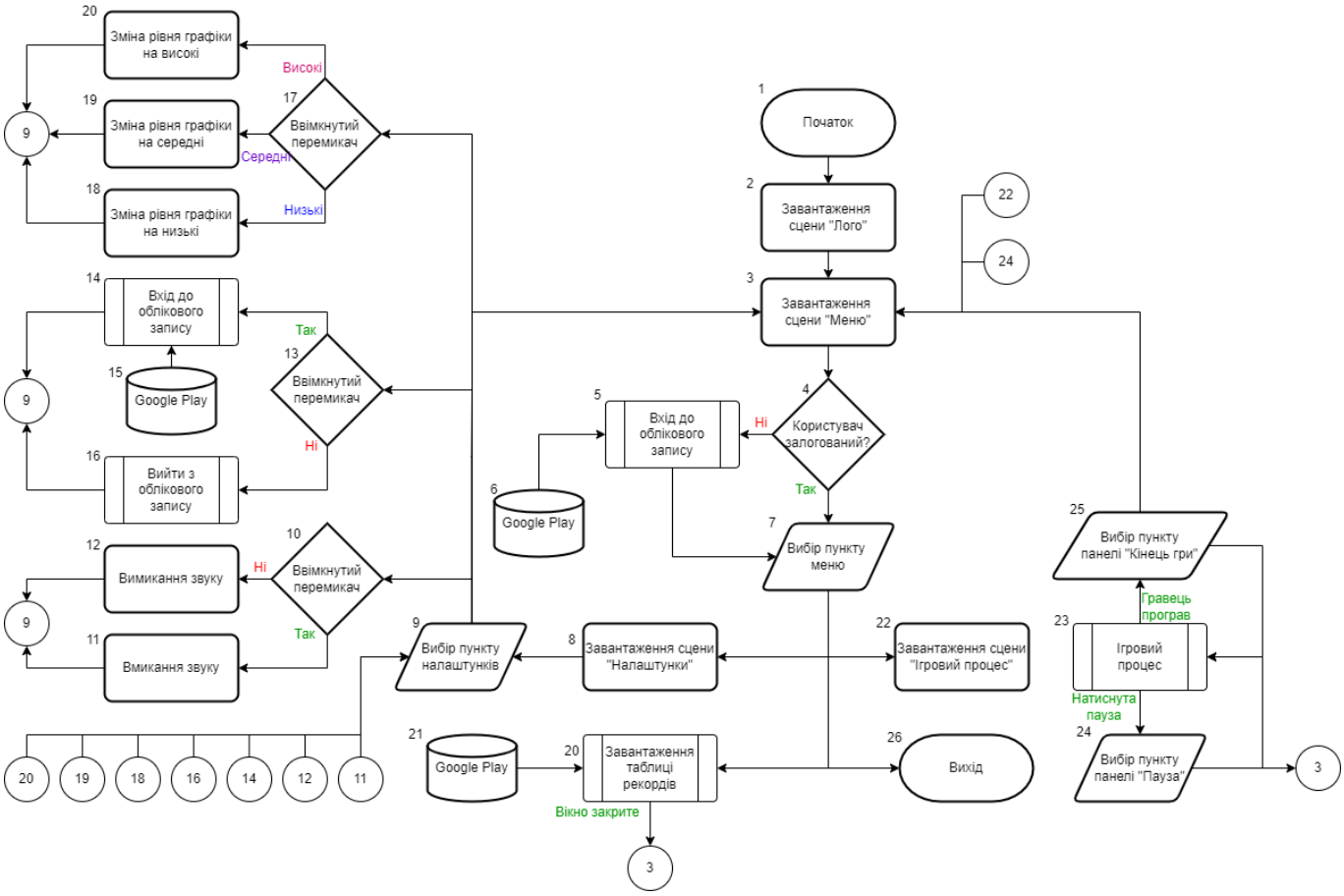


Рисунок 2.4 – Блок-схема ігрового додатку

3 РОЗРОБКА ІГРОВОГО ДОДАТКУ

3.1 Вибір програмних продуктів для розробки

Після аналізу продуктів-аналогів, постановки задачі, проектування систем та власних навичок було вирішено використовувати так програмні продукти для розробки ігрового додатку «Expulsion of Invaders»:

– Adobe Illustrator використовується для створення дизайну ігрових об’єктів у векторному представленні. На даний момент часу цей програмний продукт є найпопулярнішим та простим у вивченні.

– Adobe Photoshop використовується для створення растрових елементів ігрового додатку: фон, елементи інтерфейсу та двовимірної анімації. Також цей растровий редактор допомагав підігнати усю графічну складову в одному стилі за допомогою колірної корекції.

– FL Studio використовується для інтегрування звукового складника відеогри: синтез, еквалізація, компресія, зведення та майстеринг. За його допомогою було створено нові та відредаговано існуючи звукові ефекти.

– Visual Studio використовується в якості редагування коду відеогри та впровадження у ядро ігрового рушію Unity задля тестування та дебагу коду.

– Unity є головним програмним продуктом для розробки ігрового додатку. Це ігровий рушій, який використовує мову програмування C# .NET та підтримує різні плагіни, серед яких Google Play Services та Google AdMob.

– Google Play Console використовується для завантаження реалізованого ігрового додатку на цифровий ринок Play Market.

3.2 Програмна реалізація

Першим кроком у розробці ігрового додатку є створення проекту ігрового рушію Unity. Так як розроблюваний ігровий додаток має двовимірне представлення, та цільовою платформою розробки є Android, використовується вбудований шаблон проекту «2D Mobile» (рис. 3.1).

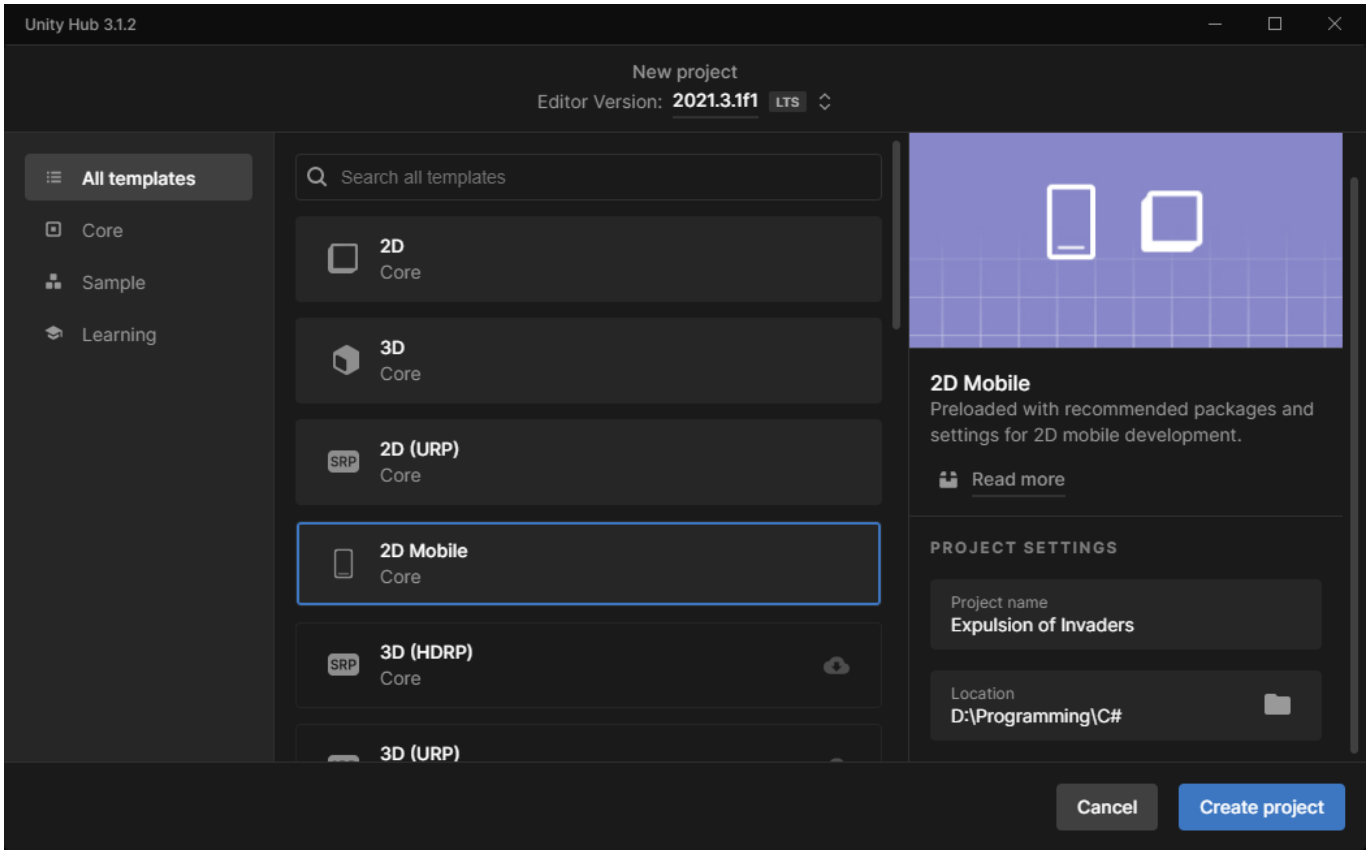


Рисунок 3.1 – Створення проекту Unity

Наступним кроком змінюємо платформу зборки на Android (рис. 3.2). Також налаштовуємо параметри Player проекту – створюємо ключ продукту, заповнюємо назву ігрового додатку, розробника, версію та інше.

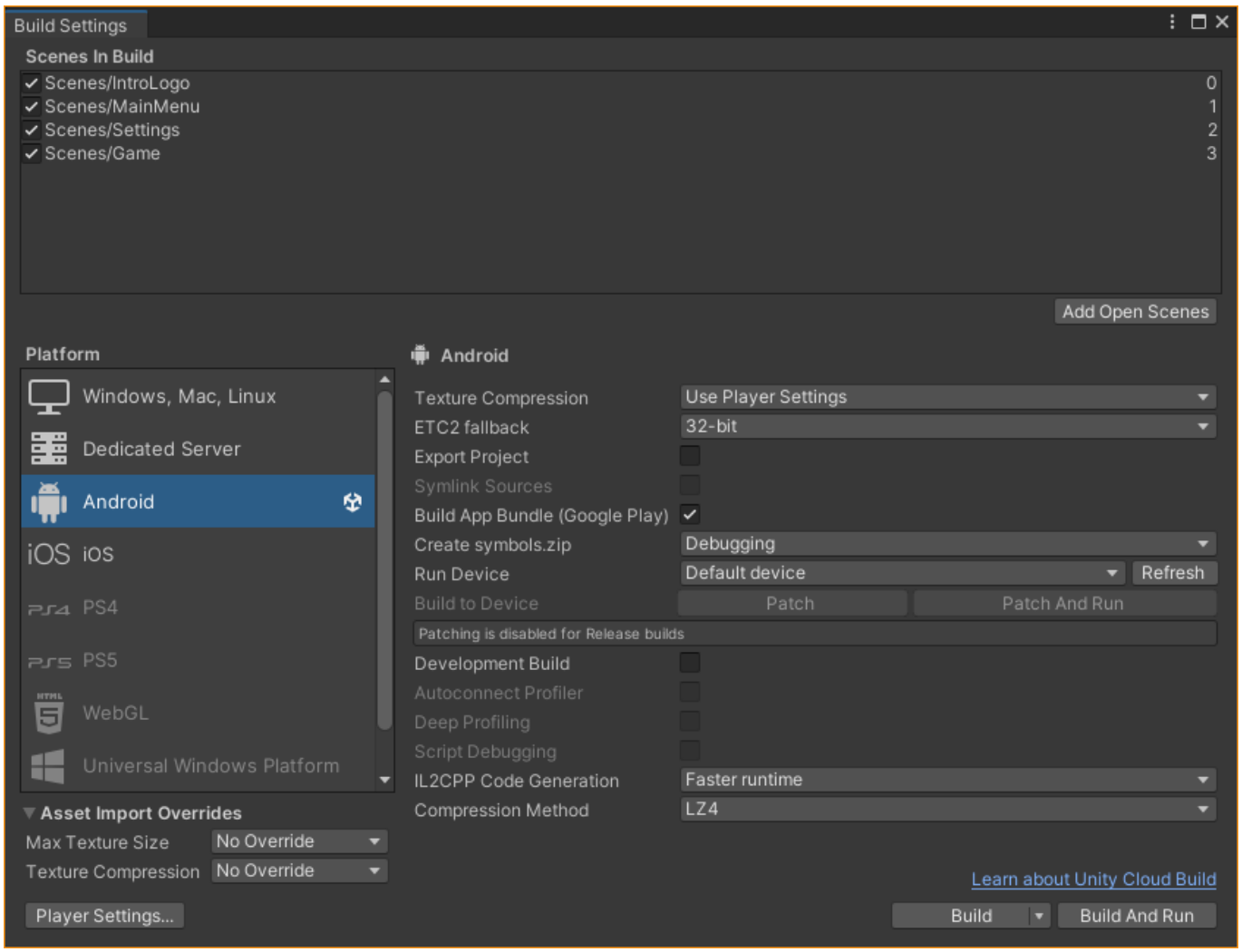


Рисунок 3.2 – Параметри зборки проекту

Далі розпочинаємо основний етап розробки, а саме створення сцен, ігрових об’єктів та префабів на цих сценах. Також важливо відокремити ігрові об’єкти, які задіяні у самому ігровому процесі, та об’єкти інтерфейсу, так як вони мають різні шкали виміру (ігрові об’єкти використовують координати відносно ігрового світу, а об’єкти інтерфейсу – координати екрану). Після цього сцени мають наповнення, показане на рисунках 3.3 – 3.6.

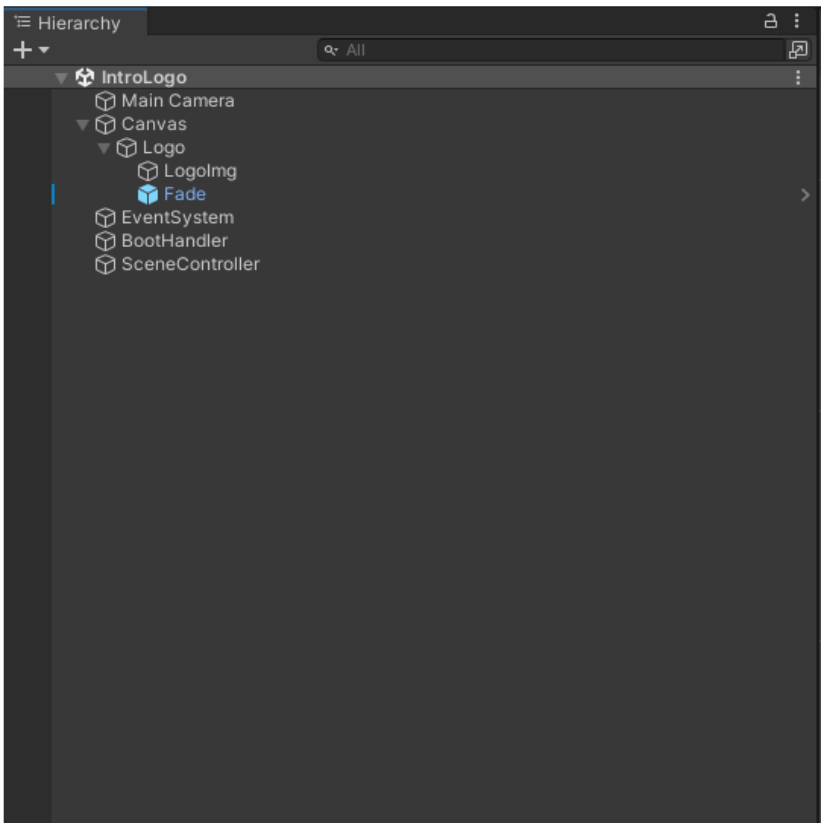


Рисунок 3.3 – Ієрархія наповнення сцени IntroLogo

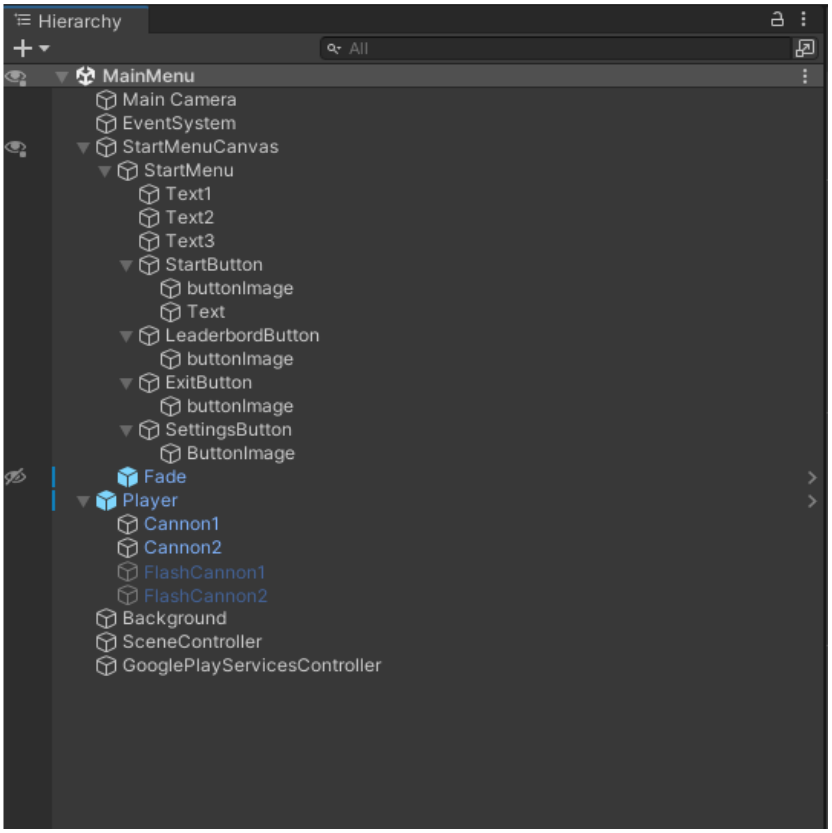


Рисунок 3.4 – Ієрархія наповнення сцени MainMenu

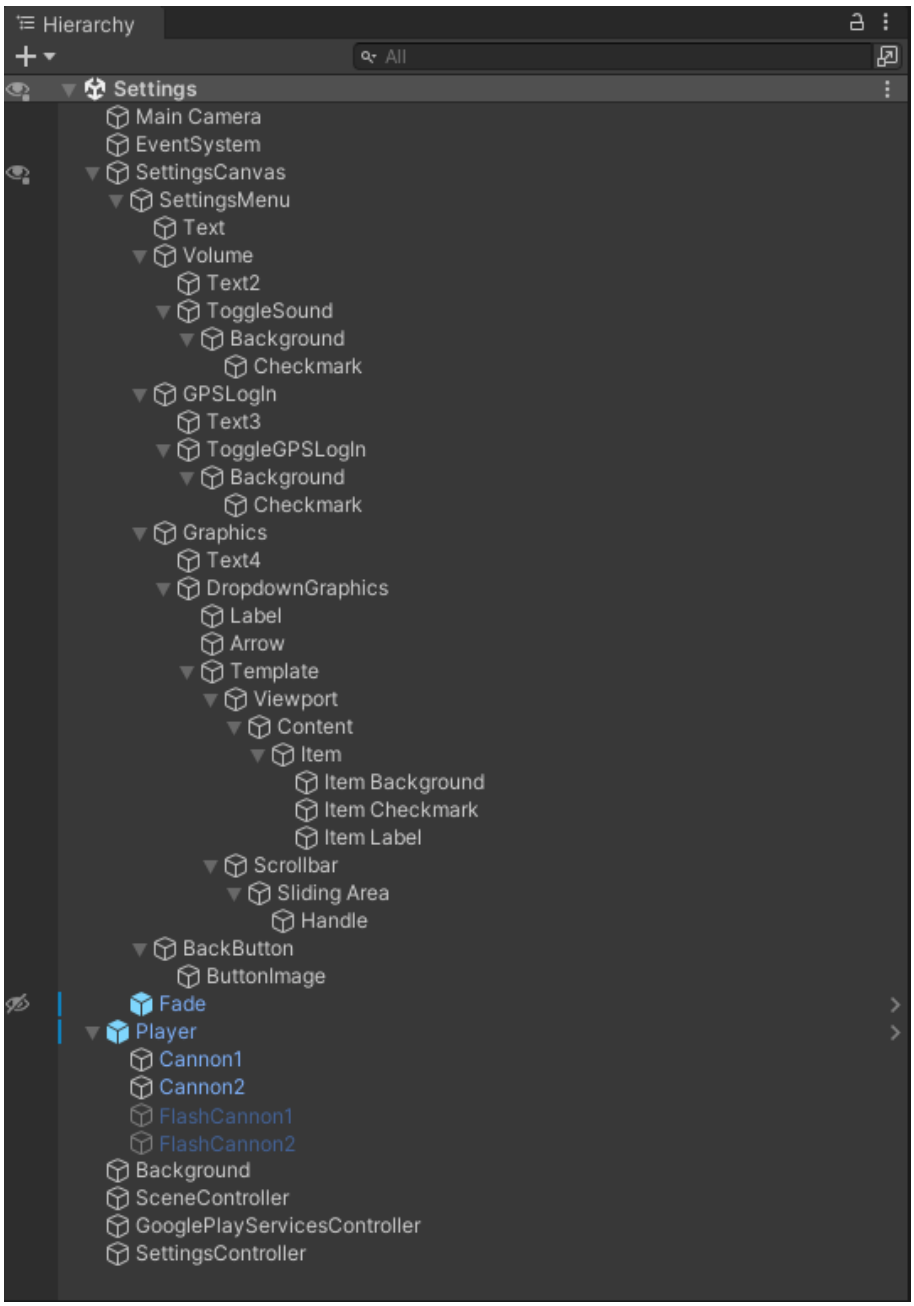


Рисунок 3.5 – Ієрархія наповнення сцени Settings

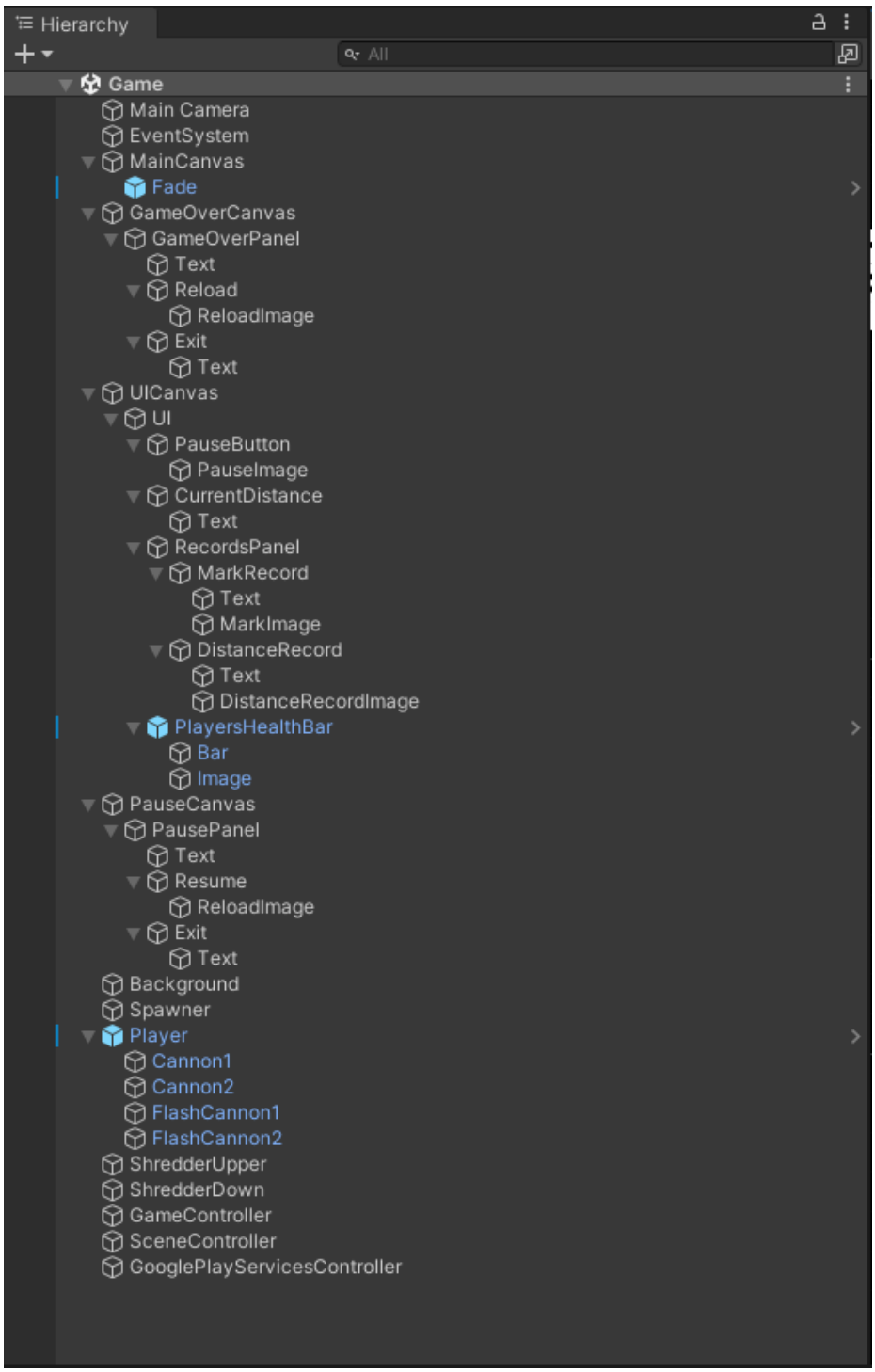


Рисунок 3.6 – Ієрархія наповнення сцени Game

Наступним кроком імпортуємо створені ігрові асети та прикріплюємо їх до створених ігрових об’єктів. Для об’єктів, які матимуть анімацію, нарізаємо спрайт на кадри за допомогою вбудованого SpriteEditor та створюємо на основі нарізаних кадрів анімацію. Приклади обробки спрайтів та наповнення компонентами ігрових об’єктів зображено на рисунку 3.7 та рисунку 3.8 відповідно.

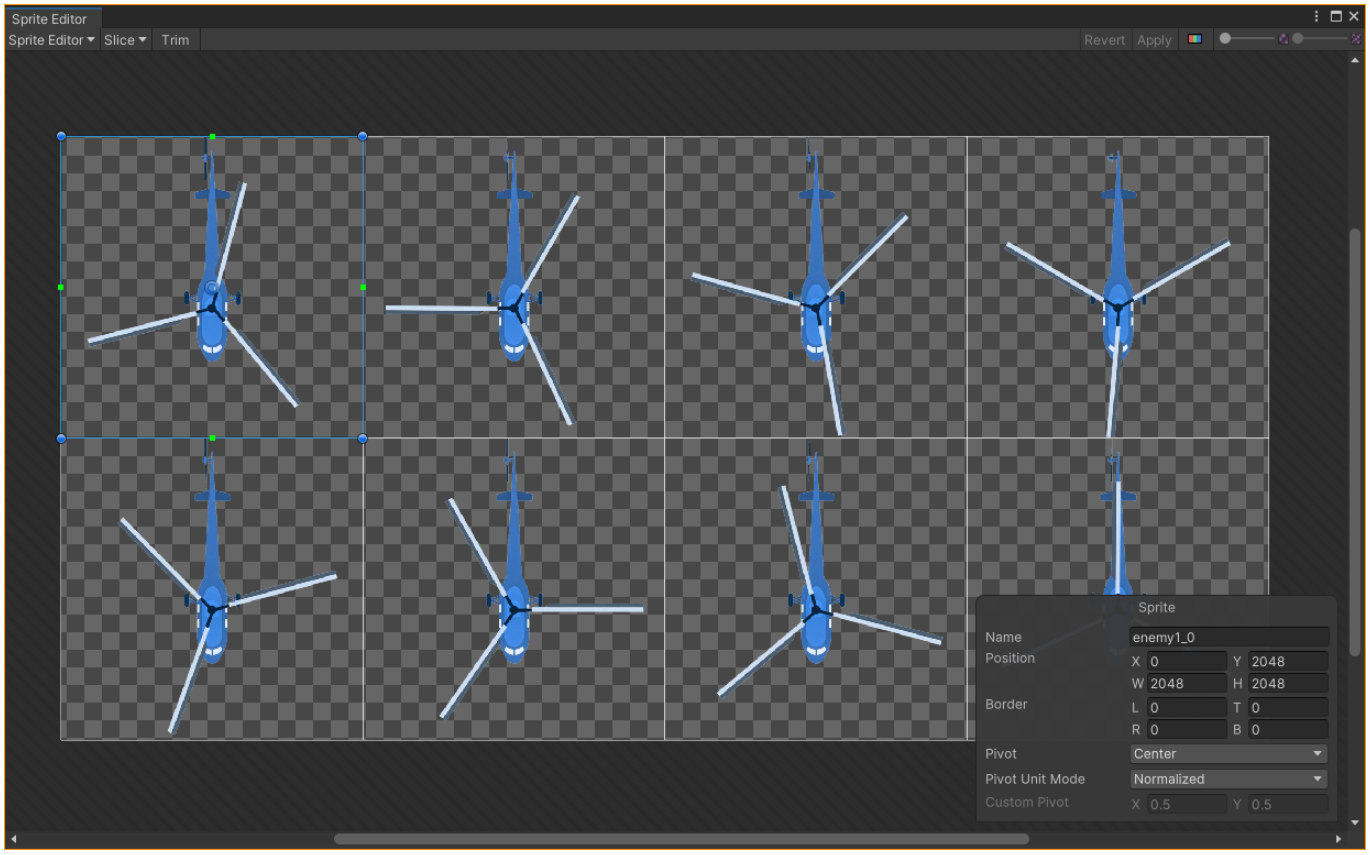


Рисунок 3.7 – Приклад створення анімації

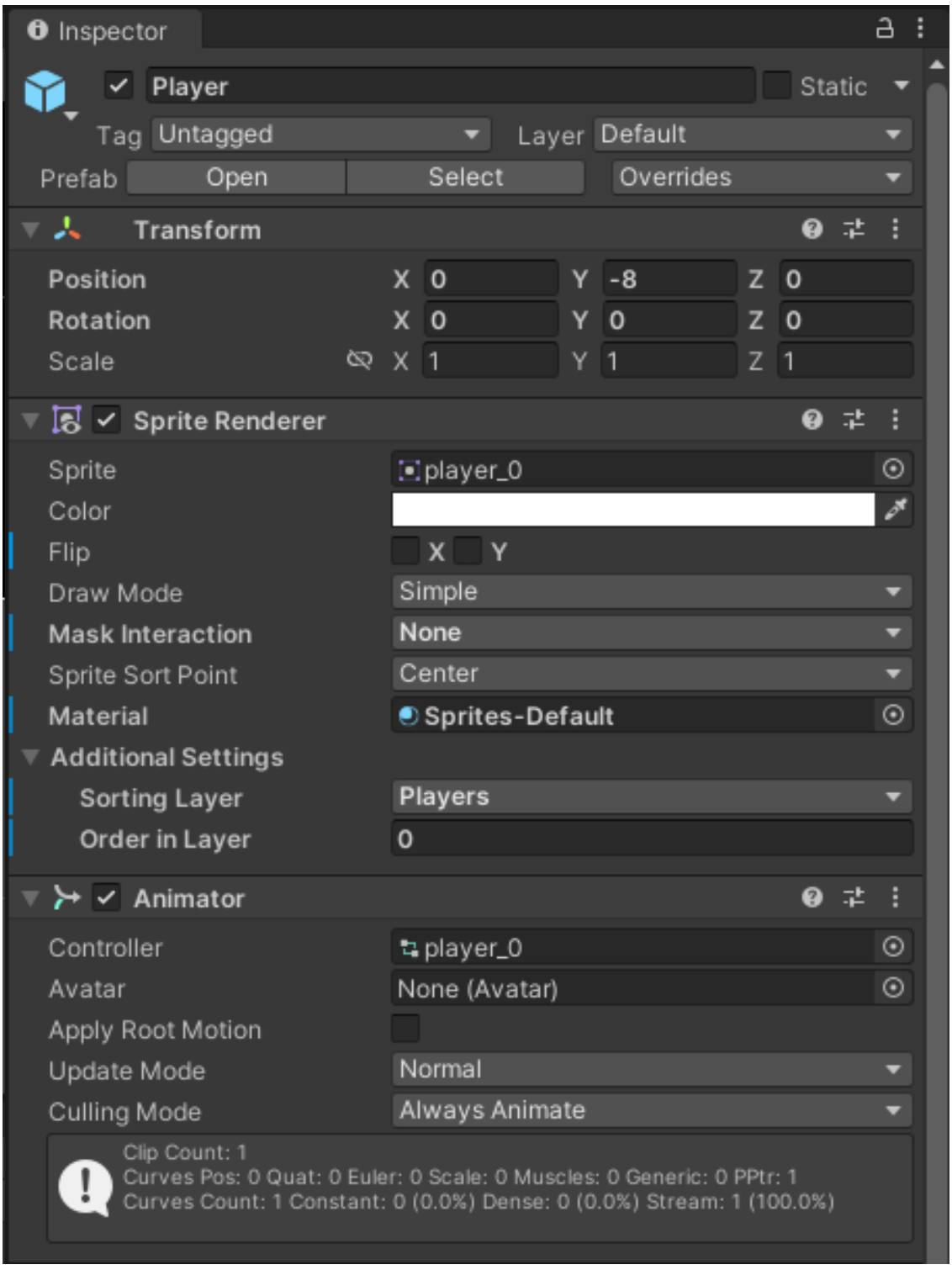


Рисунок 3.8 – Приклад компонентів ігрових об’єктів

Далі переходимо до скриптової частини розробки. Створюємо логіку ігрового додатку та його ігровий процес. Короткий опис алгоритмів кожного зі скриптів подано у таблиці 3.1.

Таблиця 3.1 – Опис скриптів та їх алгоритмів

Назва скрипту	**Що робить**
ParallaxController.cs	Переміщує матеріал ігрового фону, створюючи ефект постійного переміщення гравця
PlayerController.cs	Відповідає за переміщення та існування гравця на ігровому полі
ShotController.cs	Відповідає за вогневе озброєння гравця
Shredder.cs	Знищує ігрові об’єкти які виходять за поля екрану
Spawner.cs	Випадково генерує противників на ігровому полі
BulletState.cs	Відповідає за фізику снарядів гравця
EnemyBulletState.cs	Відповідає за фізику снарядів противників
EnemyController.cs	Відповідає за переміщення, тип та озброєння противників
EnemyEnum.cs	Перерахований тип противників
MovementEnum.cs	Перерахований тип напрямків руху
AdMobController.cs	Відповідає за ініціалізацію та показ міжсторінкової реклами сервісу Google AdMob
GooglePlayerServicesConnector.cs	Реалізує вхід до облікового запису та підключення таблиці рекордів сервісу Google Play
BootHandler.cs	Відповідає за управління початкової сцени з логотипом розробника
DistanceCounter.cs	Підраховує пройдену відстань та записує її у реєстр пам’яті та у таблицю рекордів
GameController.cs	Відповідає за управлінням інтерфейсу ігрового процесу
HPBarController.cs	Реалізовує систему життів у противників

Назва скрипту	Що робить
HPBarUI	Реалізовує систему життів у гравця
MarkCounter.cs	Підраховує кількість зібраних поштових марок та записує їх у реєстр пам’яті та у таблицю рекордів
SceneController.cs	Відповідає за перехід між сценами
SettingsController.cs	Відповідає за параметри налаштування ігрового додатку
GPS.cs	Відповідає за параметри підключення реклами та пакетів Google

Повний код скриптів подано у додатку Г. Проект завантажений на інтернет-репозиторій GitHub [18].

Додаємо створені скрипти до ігрових об’єктів та префабів. Підключаємо ігрові об’єкти та скрипти між собою. Приклад готових сцен наведено на рисунках 3.9 – 3.12.

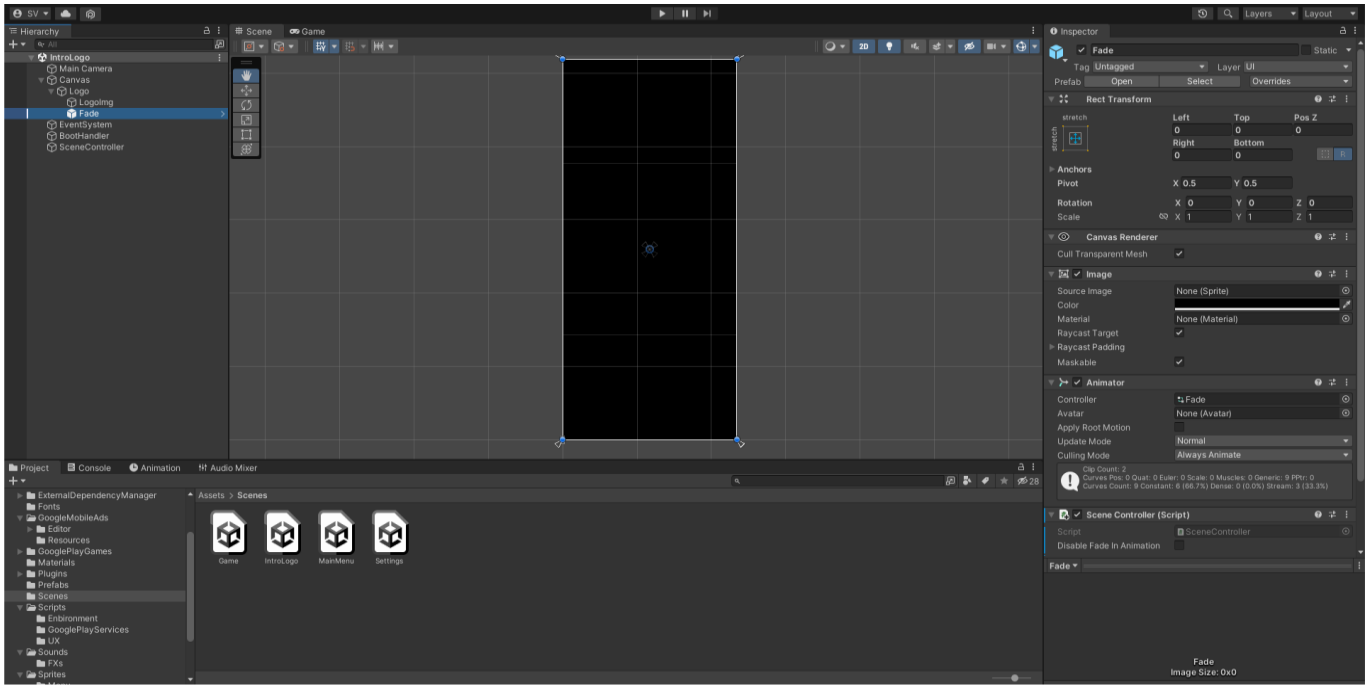


Рисунок 3.9 – Повний вигляд сцени IntroLogo

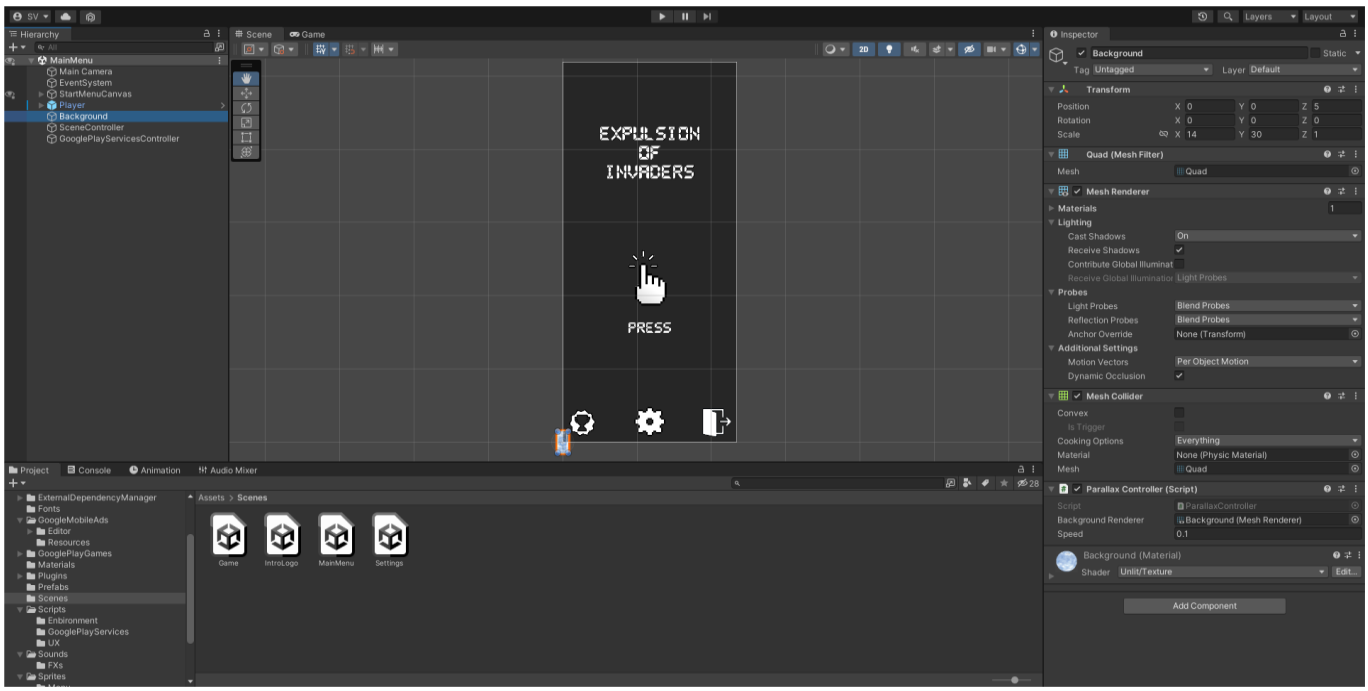


Рисунок 3.10 – Повний вигляд сцени MainMenu

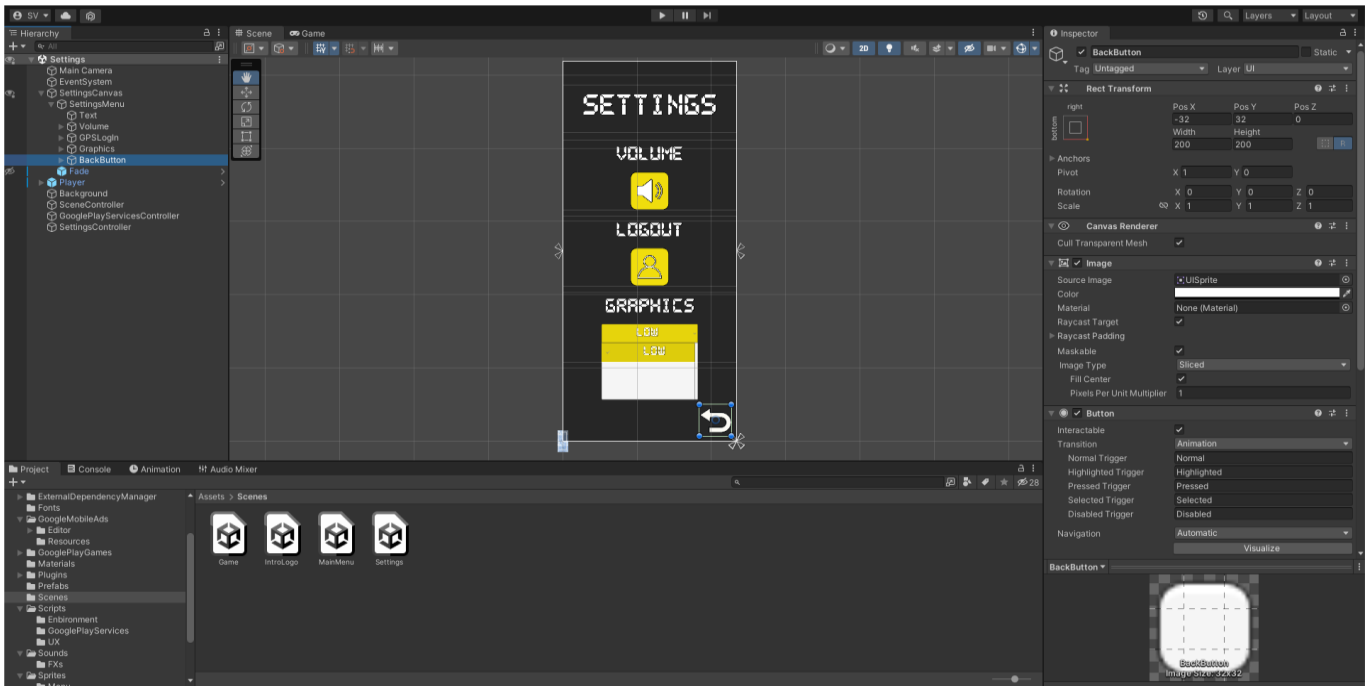


Рисунок 3.11 – Повний вигляд сцени Settings

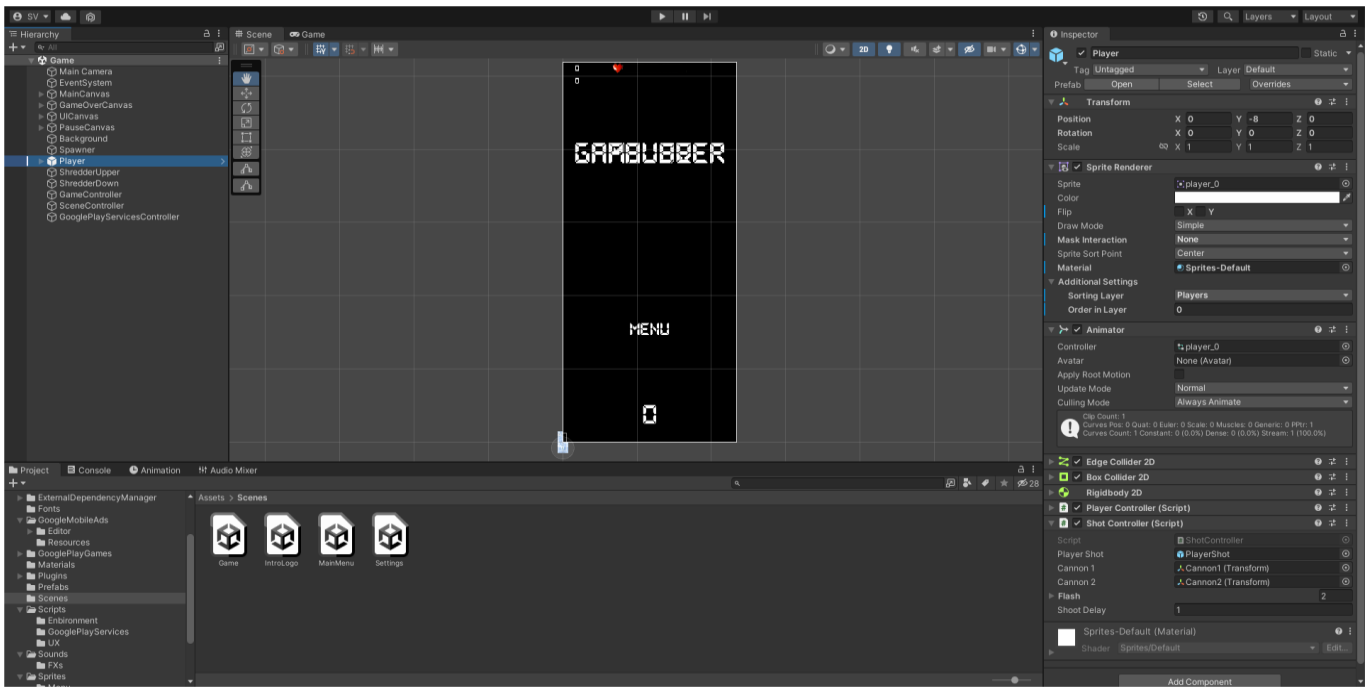


Рисунок 3.12 – Повний вигляд сцени Game

Останнім кроком налаштовуємо обліковий запис Google Developer Console, проект на ресурсі Google Play Console, компілюємо зборку в .aab файл та завантажуємо його на перевірку до цифрового ринку Play Market. Після проходження тестувань та перевірок публікуємо збірку на сторінці ігрового додатку.

3.3 Тестування та працездатність

Завантажуємо зборку внутрішнього тесту з Google Play Console. Встановлюємо зборку та запускаємо її. Першою повинна завантажитися сцена з логотипом розробника. Далі асинхронно проводиться вхід в обліковий запис Google Play для підключення до таблиці рекордів (рис. 3.13).

Рисунок 3.13 – Стартова сцена ігрового додатку «Expulsion of Invaders»

Наступною сценою завантажується головне меню ігрового додатку, на якому розташовано чотири абстракті кнопки: таблиця рекордів, налаштування, вихід та початок гри (рис. 3.14).

Рисунок 3.14 – Головне меню ігрового додатку «Expulsion of Invaders»

Далі натискаємо кнопку налаштувань та переходимо на сцену самих параметрів налаштувань (рис. 3.15).

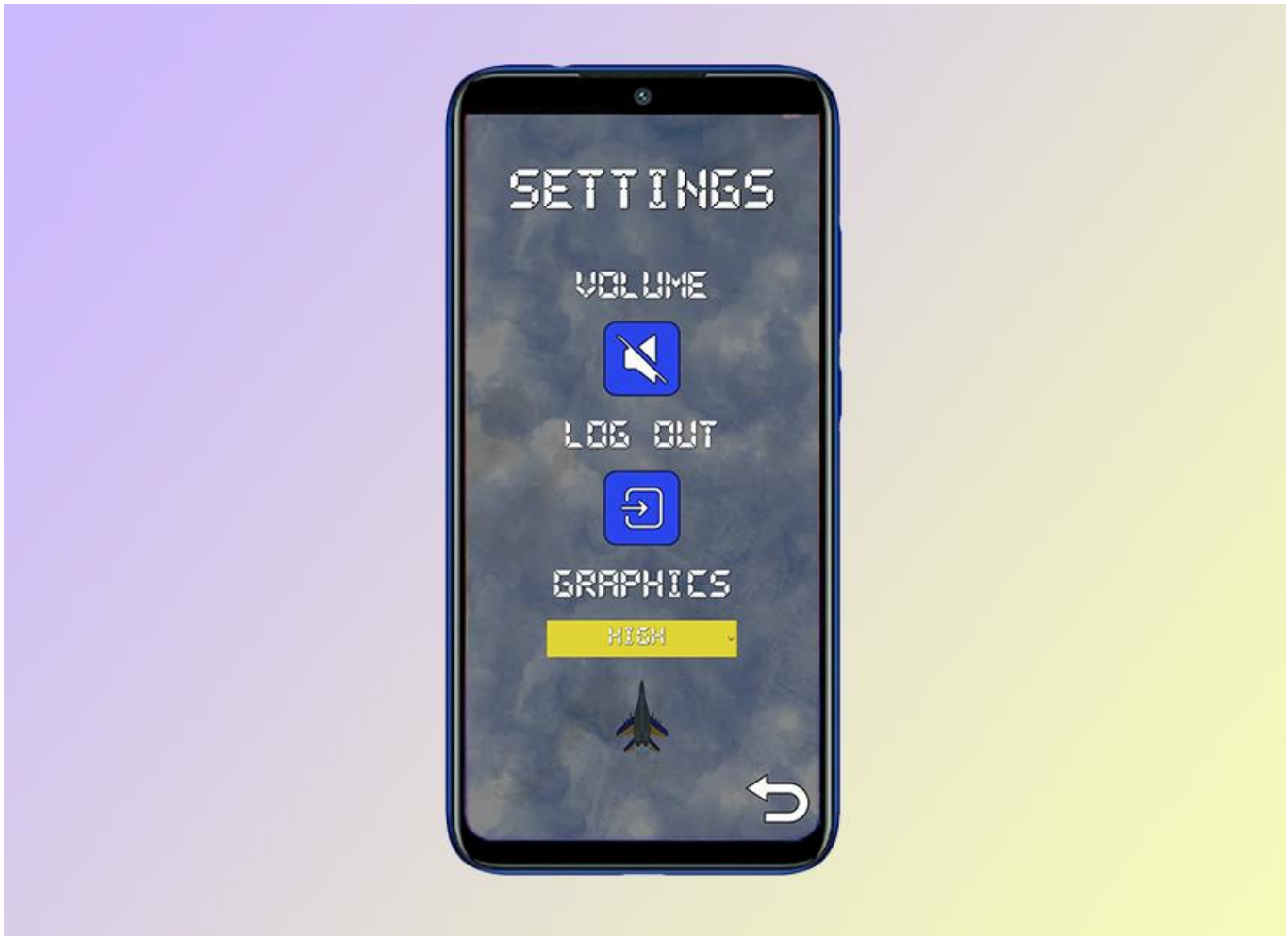


Рисунок 3.15 – Налаштування ігрового додатку «Expulsion of Invaders»

Після чого тестуємо систему налаштувань шляхом змін параметрів та повернення до головного меню. Після повторного заходу до налаштувань спостерігаємо, що змінені параметри збереглися (рис. 3.16).

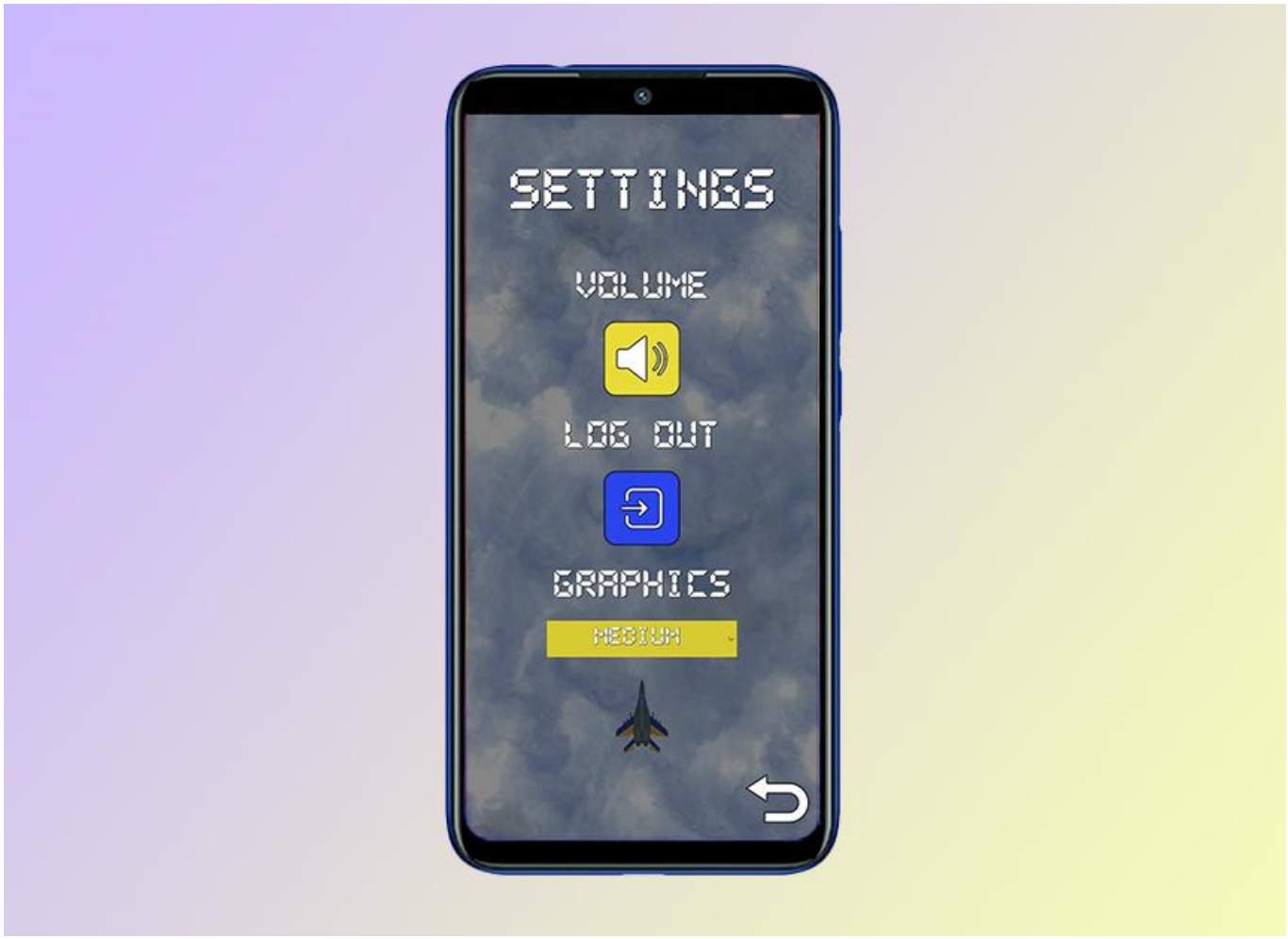


Рисунок 3.16 – Зміна налаштувань ігрового додатку «Expulsion of Invaders»

Наступним кроком переходимо до вікна таблиць рекордів. Перевіряємо чи збігається кількість та назви таблиць у ігровому додатку та сервісу Google Play Console (рис. 3.17).

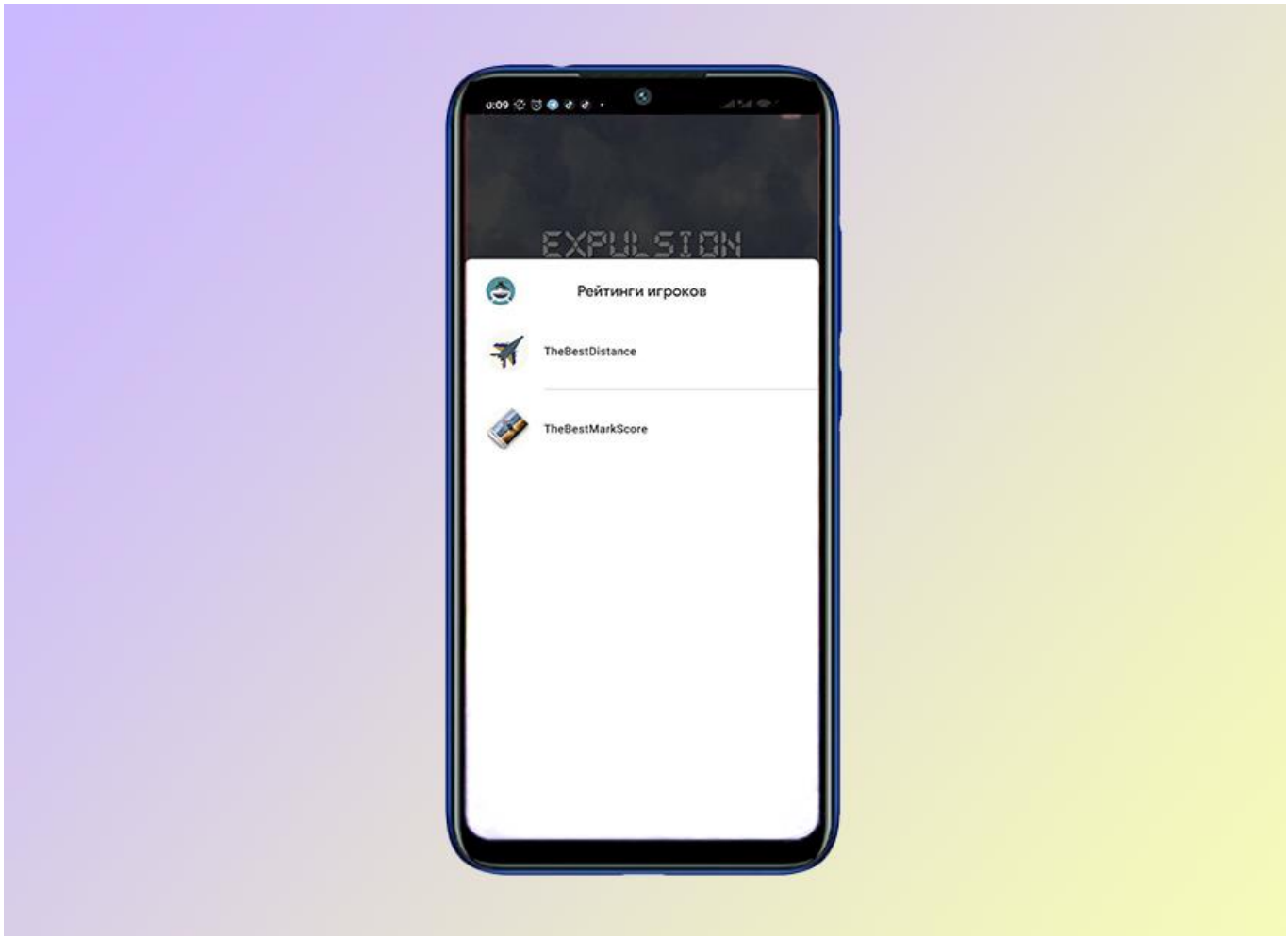


Рисунок 3.17 – Вікно таблиць рекордів ігрового додатку «Expulsion of Invaders»

Далі переходимо до підменю кожної таблиці та перевіряємо результати рекордів серед обраних параметрів (рис. 3.18 – рис. 3.19).

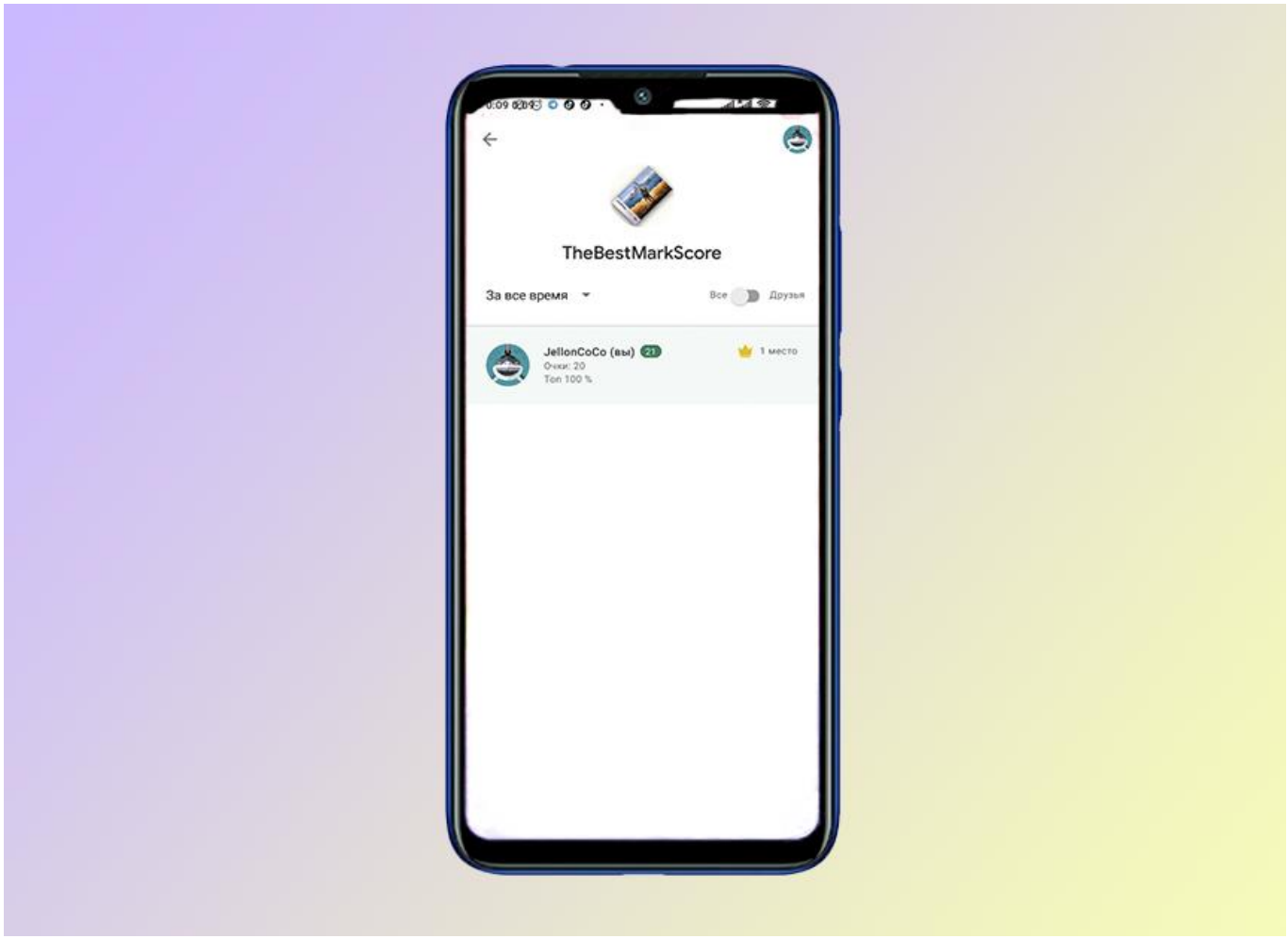


Рисунок 3.18 – Таблиця рекордів 1 ігрового додатку «Expulsion of Invaders»

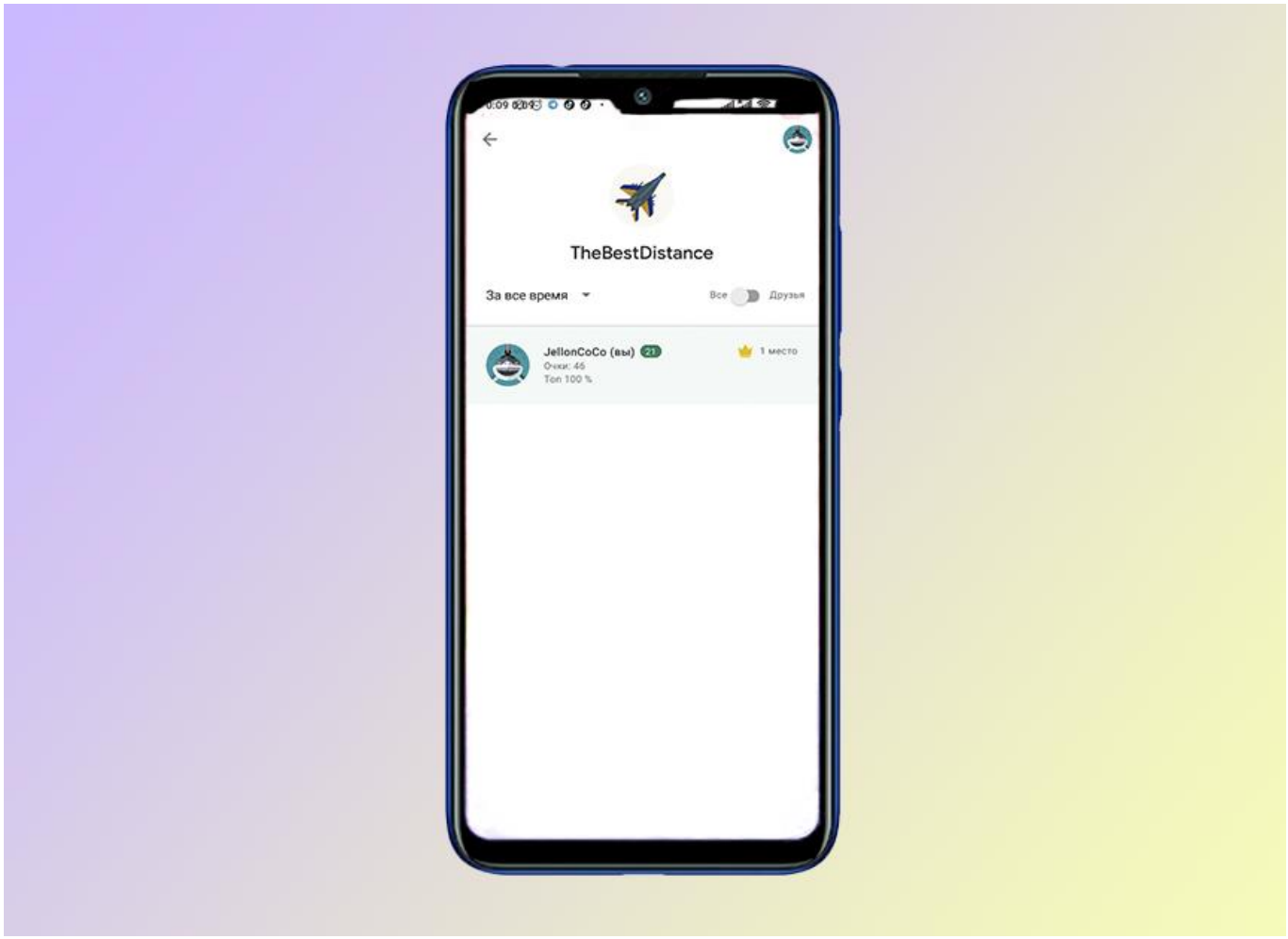


Рисунок 3.19 – Таблиця рекордів 2 ігрового додатку «Expulsion of Invaders»

Повертаємося до головного меню та переходимо до тестування основного ігрового процесу ігрового додатку (рис.3.20 – 3.22).

Так як жанр ігрового додатку це аркада, головною метою його є поставлення рекордів та змагання у таблиці лідерів серед інших гравців. Основою ігрового процесу є геймплей відеогри «1942», суть якого полягає у знищуванні противників та побиття рекордів. Також додано ігрові монети у вигляді поштової марки, які треба збирати та змагатися за найбільшу їх кількість. Тематика гри змінена на сучасну проблему агресії росії проти України, тому граючи в ігровий додаток «Expulsion of Invaders», користувач виконує роль Привиду Києва та знищує ворожу авіацію.

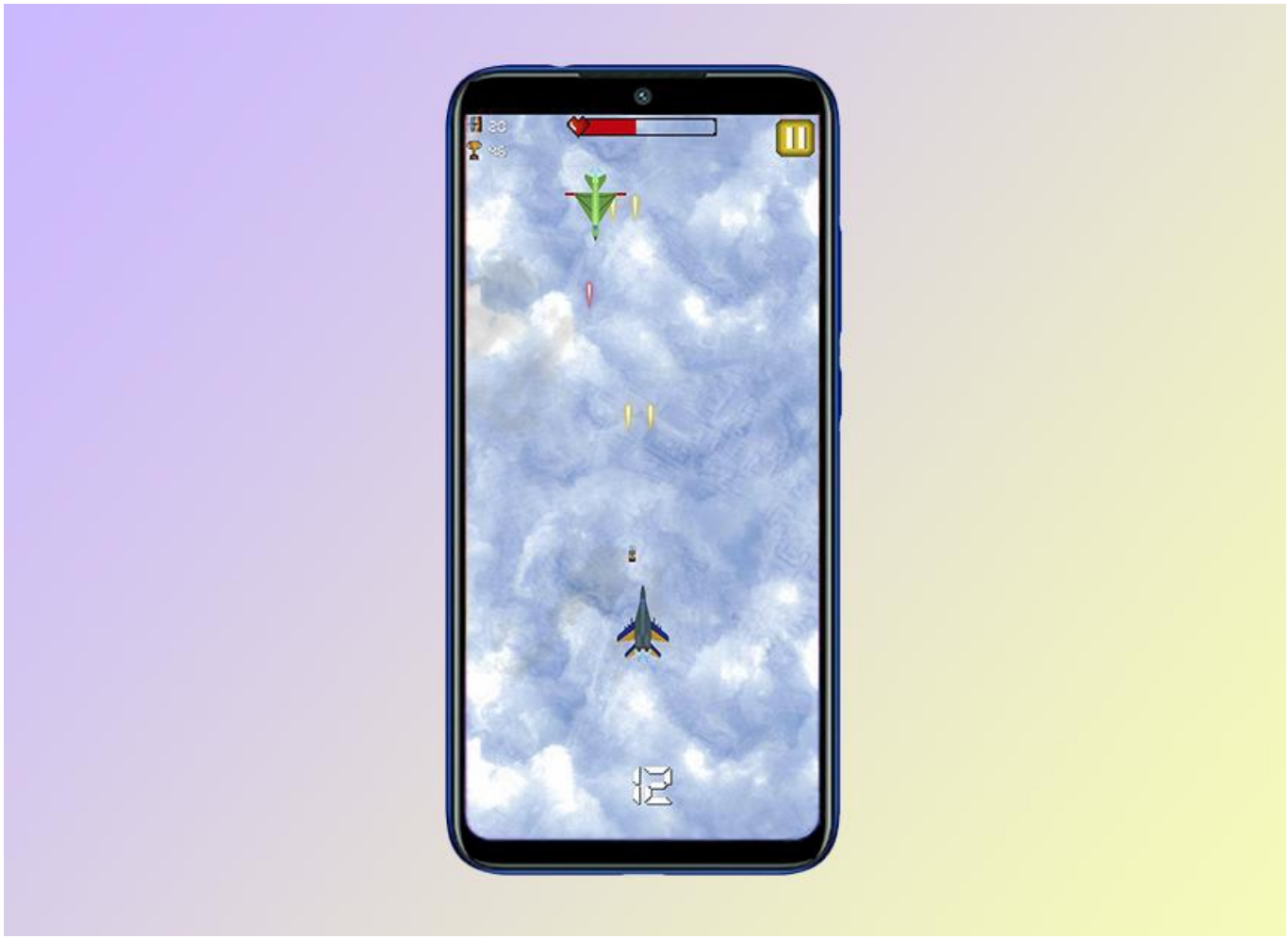


Рисунок 3.20 – Геймплей ігрового додатку «Expulsion of Invaders»

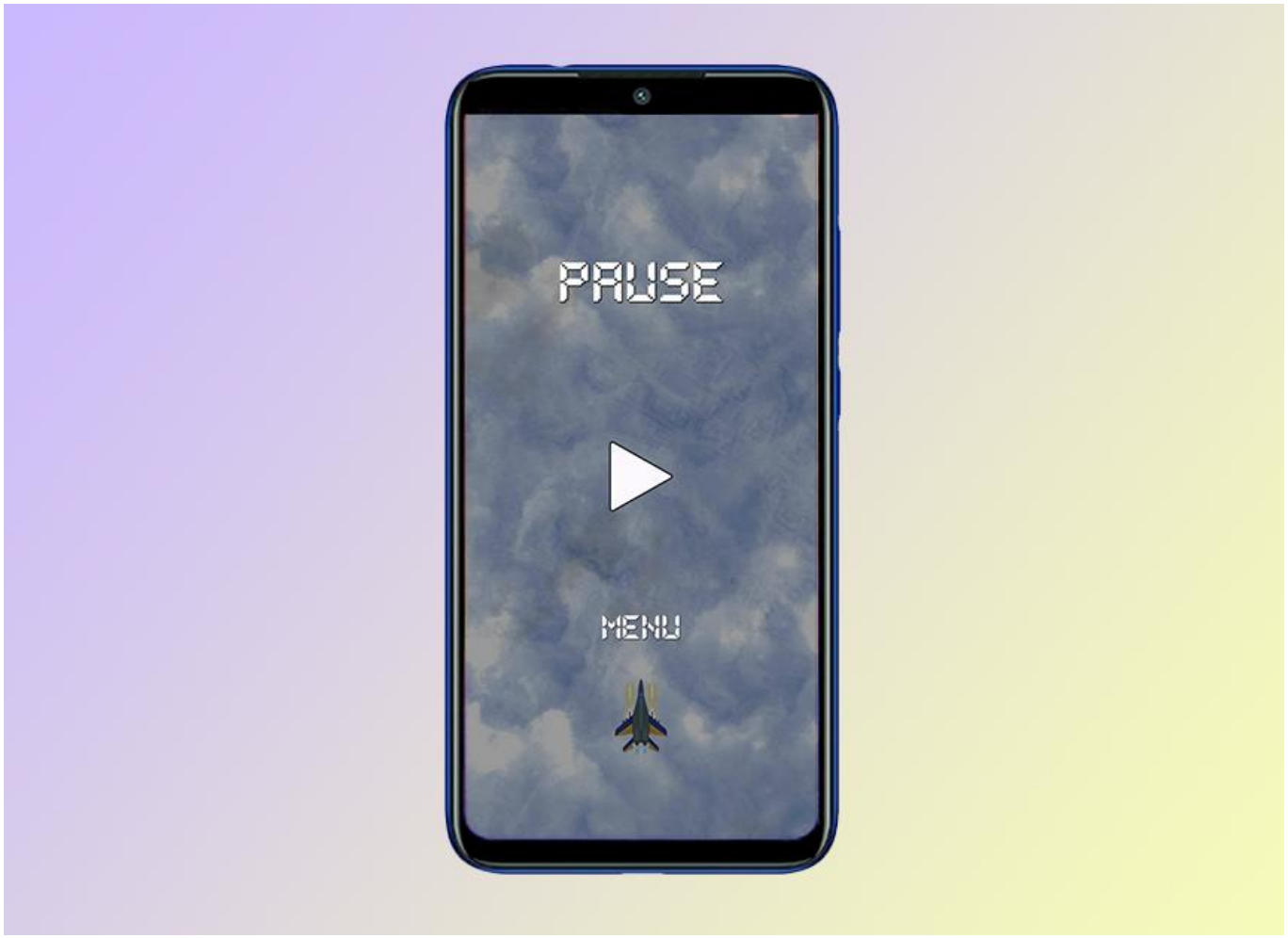


Рисунок 3.21 – Пауза геймплею ігрового додатку «Expulsion of Invaders»

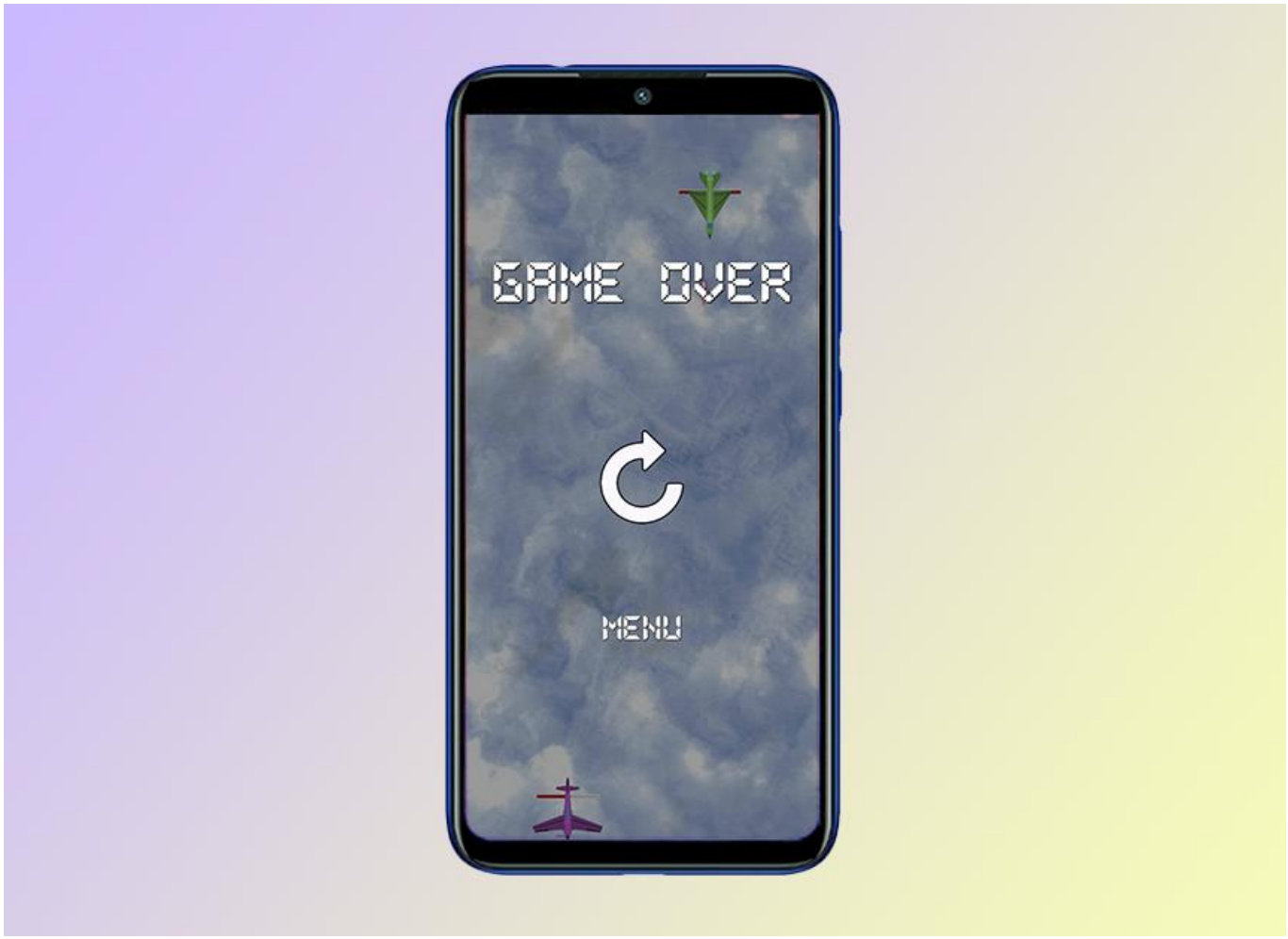


Рисунок 3.22 – Кінець гри ігрового додатку «Expulsion of Invaders»

Після кожної смерті результати записуються до реєстру, якщо локальний рекорд пристрою був побитий, то його результат завантажується до таблиці рекордів Google Play. Після кожних трьох смертей з’являється міжсторінкова рекламна інтеграція від сервісу Google AdMob (рис. 3.23). Після перегляду рекламної інтеграції гравець може продовжити насолоджуватися ігровим процесом ігрового додатку.

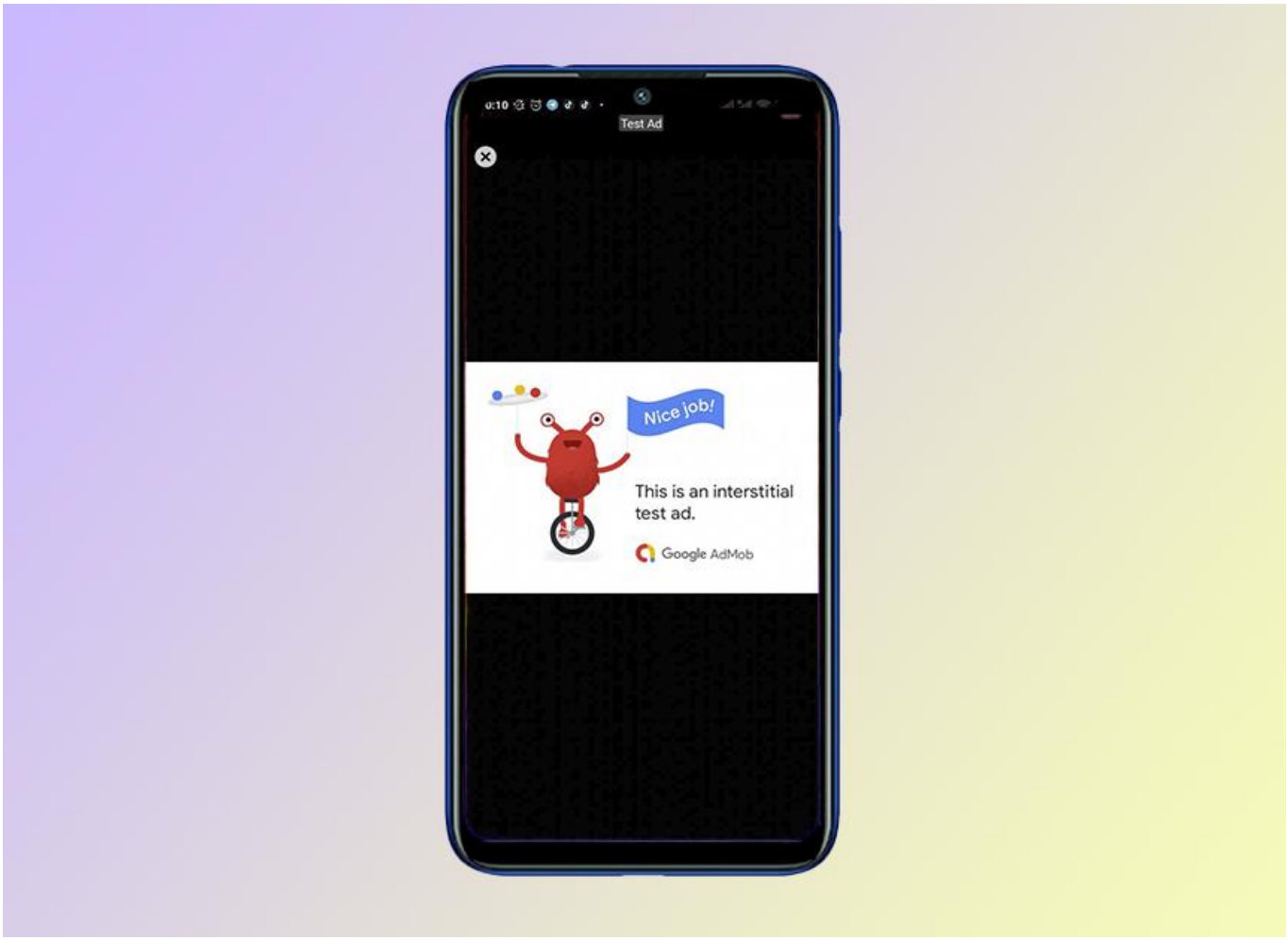


Рисунок 3.23 – Рекламна інтеграція додатку «Expulsion of Invaders»

Рекламна інтеграція монетизується за допомогою того ж сервісу Google AdMob. За певну кількість переглядів, переходів за посиланням реклами сервіс платить грошову винагороду розробнику.

ВИСНОВКИ

Для досягнення мети проекту першочергово було охарактеризовано актуальність теми шляхом дослідження предметної області та аналізу ігрових додатків-аналогів. За об’єкт дослідження предметної області було обрано двовимірні ігрові додатки жанру аркада на цільовій платформі Android. Далі було обрано три ігрові додатки-аналоги: «Duet», «Hyperforma» та «High Risers».

Наступним процесом у розробці ігрового додатку стало проектування та моделювання проекту. До цього входить створення ідеї, моделювання контекстних діаграм та діаграми прецедентів, проектування блок-схеми. За основу ігрового додатку було обрано тему російської агресії щодо України та простий та захопливий ігровий процес, в якому треба, керуючи літаком, знищувати ворожу авіацію та збирати поштові марки.

Далі була проведена розробка ігрового додатку, до якої входить: графічний дизайн гри, програмування скриптової частини додатку, створення дизайну інтерфейсу та головного меню, додавання звукового складнику та збірка ігрового додатку в .aab файл. Весь проект було реалізовано за допомогою мови програмування C# .NET, ігрового рушію Unity та середовища програмування Visual Studio. Також для створення дизайну використовувалися програми Adobe Illustrator та Adobe Photoshop, а для створення звукового складника – FL Studio.

Останнім кроком у досягненні мети проекту було проведення тестування та перевірка працездатності ігрового додатку. Після позитивних результатів ігрового додатку «Expulsion of Invaders» завантажено його на цифровий ринок Play Market. Після чого було створено супровідну документацію у виді пояснювальної записки дипломної роботи.

СПИСОК ЛІТЕРАТУРИ

1. Ігрова індустрія за 2020 рік у цифрах та фактах. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://itc.ua/news/igrovaya-industriya-za-2020-god-v-czifrah-i-faktah-infografika-gameindustry-biz/

2. Відеогра/Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%B5%D0%BE%D0%B3%D1%80%D0%B0

3. Класифікація комп'ютерних ігор/Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80

4. Дослідження впливу комп'ютерних ігор на розвиток дітей. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://stud.com.ua/127079/psihologiya/doslidzhennya_vplivu_kompyuternih_igor_rozvitok_ditey

5. Аркада (жанр)/Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу:URL:https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D0%B0_(%D0%B6%D0%B0%D0%BD%D1%80)

6. Кількість активних Android-гаджетів у світі. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://itc.ua/news/kolichestvo-aktivnyh-ustrojstv-android-v-mire-prevysilo-3-milliarda/

7. В Україні встановлено мінімальну зарплату і прожитковий мінімум на 2022 рік/Головне управління Держпраці у Київській області. [Електронний ресурс] – Режим доступу до ресурсу: URL:http://kiev.dsp.gov.ua/novyny/v-ukraini-zrosly-minimalna-zarplata-i-prozhytkovyi-minimum/

8. Найкращі мобільні телефони середнього класу. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://www.androidsis.com/uk/%D0%BD%D0%B0%D0%B9%D0%BA%D1%80%D0%B0%D1%89%D1%96-%D0%BC%D0%BE%D0%B1%D1%96%D0%BB%D1%8C%D0%BD%D1%96-%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%B8-%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%8C%D0%BE%D0%B3%D0%BE-%D0%BA%D0%BB%D0%B0%D1%81%D1%83/

9. Unity (ігровий рушій)/Вікіпедія. URL:https://ru.wikipedia.org/wiki/Unity_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)

10. Топ кращих ігор жанру аркада/Google Play. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://play.google.com/store/apps/collection/cluster?clp=ogoXCAkSC0dBTUVfQVJDQURFKgIIB1ICCAE%3D:S:ANO1ljLjoNQ&gsr=ChqiChcICRILR0FNRV9BUkNBREUqAggHUgIIAQ%3D%3D:S:ANO1ljIj13g&hl=ru&gl=US

11. Мобільна гра «Duet»/Google Play. URL:https://play.google.com/store/apps/details?id=com.kumobius.android.duet&hl=ru&gl=US

12. Мобільна гра «Hyperforma»/Google Play. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://play.google.com/store/apps/details?id=com.herocraft.game.hyperforma.cyber.hack.arcade

13. Мобільна гра «High Risers»/Google Play. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://play.google.com/store/apps/details?id=com.kumobius.android.highrisers&hl=ru&gl=US

14. What is IDEF - Definition, Methods, and Benefits/Edraw. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://www.edrawsoft.com/what-is-idef.html

15. Моделювання процесу управління стратегічною гнучкістю підприємства/Електронний журнал «Ефективна економіка». [Електронний ресурс] – Режим доступу до ресурсу: URL:http://www.economy.nayka.com.ua/?op=1&z=1729#:~:text=IDEF0%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C%20%D1%81%D0%BA%D0%BB%D0%B0%D0%B4%D0%B0%D1%94%D1%82%D1%8C%D1%81%D1%8F%20%D0%B7,%D1%82%D0%B0%20%D0%B0%D1%81%D0%BE%D1%86%D1%96%D0%B9%D0%BE%D0%B2%D0%B0%D0%BD%D1%96%20%D0%B7%20%D0%BD%D0%B8%D0%BC%D0%B8%20%D0%B2%D1%96%D0%B4%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%BD%D1%8F.

16. Розробка UML діаграми варіантів використання/StudFiles. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://studfile.net/preview/5200239/page:6/

17. Блок-схема/Вікіпедія. [Електронний ресурс] – Режим доступу до ресурсу: URL:https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0

18. Репозиторій ігрового додатку «Expulsion of Invaders»/GitHub [Електронний ресурс] – Режим доступу до ресурсу: URL:https://github.com/AlexanderSmasher/Expulsion-Of-Invaders

ДОДАТОК А. Технічне завдання

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Технічне завдання
на розробку ігрового додатку

«Expulsion of Invaders»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Федотова Н.А.

Студент групи ІТ-81

_____ Слободяник О.В.

2022

1 ПРИЗНАЧЕННЯ І МЕТА ІГРОВОГО ДОДАТКУ

1.1 Призначення ігрового додатку

Ігровий додаток призначений для задоволення розважальних потреб користувача завдяки захопливому та інтенсивному ігровому процесу.

1.2 Мета створення ігрового додатку

Метою даного дослідження є розробка ігрового додатку під назвою «Expulsion of Invaders» жанру двовимірна аркада для мобільної платформи Android на цікаву та актуальну тематику на базі ігрового рушію Unity.

1.3 Цільова аудиторія

Цільовою аудиторією даного ігрового додатку є користувачі віком від 8 років, які мають мобільні пристрої з операційною системою Android 11.0 і вище та підключенням до мережі інтернет.

2 ВИМОГИ ПРОЕКТУ

2.1 Вимоги до мобільного додатку в цілому

2.1.1 Вимоги до функціоналу

Оскільки ігровий додаток розроблюється на базі ігрового рушію Unity, вся скриптова частина повинна створюватися за допомогою мови програмування C#. Також збірка гри повинна бути реалізована у форматі .aab, так як потенційною платформою ігрового додатку є Android.

2.1.2 Вимоги до ігрового процесу

Жанром гри є двовимірна аркада, тому геймплей повинен бути простим та цікавим. Наявність рейтингу серед гравців є головною опцією ігрового додатку, бо наявність рейтингу надає користувачам змагального інтересу до відеогри.

2.1.3 Вимоги до дизайну

Дизайн ігрового додатку повинен дотримуватися одного стилю та однієї тематики. Через те що відеогра реалізується у двовимірному просторі, замість тривимірних моделей будуть використовуватися спрайти. Також інтерфейс матиме власний дизайн згідно з правилами UI та UX.

2.1.4 Вимоги до ресурсів

Усі використанні спрайти, елементи інтерфейсу, бекграунди, звуки, тощо повинні мати дозвіл на комерційне використання, в іншому випадку вони повинні бути зробленими самостійно.

2.1.5 Вимоги до апаратного забезпечення

Для стабільної роботи ігрового додатку з налаштуваннями високого рівня графіки користувач повинен мати операційну систему Android 11 і вище, приблизно 100 мб вільної пам’яті, 4 гб оперативної пам’яті або більше, процесор Mediatek Helio G80 схожої потужності або новіше.

2.1.6 Вимоги до релізу ігрового додатку

Ігровий додаток повинен мати назву «Expulsion of Invaders» та бути реалізованим на цифровому ринку Play Market. Також відеогра повинна включати в себе рекламну інтеграцію Google та рейтинг серед гравців Google Play.

2.2 Структура мобільного додатку

2.2.1 Структура ігрового процесу

Ігровий додаток повинен мати наступні правила:

– гравець повинен керувати лише одним персонажем або ігровим об’єктом;

– сцена гри повинна мати нескінченний процес доти, доки гравець не втратить умовні одиниці життя;

– на ігровому полі повинні з’являтися перешкоди та умовні монети, які будуть впливати на поточний рейтинг гравця за одну ігрову сесію;

– складність гри повинна змінюватися з часом;

– повинен присутній баланс між гравцем та супротивниками або перешкодами.

2.2.2 Структура головного меню

Головне меню повинно складатися з таких елементів як:

– кнопка «Грати»;

– кнопка «Налаштування»;

– кнопка «Рейтинг»;

– кнопка «Вихід».

2.2.3 Структура звукових ресурсів

Усі звукові ресурси повинні бути оброблені за такими критеріями:

– наднизькі частоти (<100 гц) повинні бути у монофонічній панорамі;

– частоти менші за 20 гц та більші за 20 000 гц повинні вирізатися еквалайзером;

– звук повинен бути скомпресований до -0db задля уникнення кліпування та перепаду гучності між різними звуками;

– усі резонанси частот повинні бути прибрані.

2.2.4 Структура графічних ресурсів

Так як ігровий додаток має двовимірну реалізацію, графічні ресурси повинні складатися:

– спрайти гри повинні мати високу якість та дозвіл;

– файли спрайтів та анімація повинні зберігатися у форматі .psd, тобто у файлі програмного забезпечення Adobe Photoshop;

– усі файли спрайтів повинні мати однакову корекцію кольору.

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ ІГРОВОГО ДОДАТКУ

Детальний опис етапів і строків розробки та реалізації ігрового додатку наведено в таблиці А.1.

Таблиця А.1 – Етапи створення ігрового додатку

№	Склад і зміст робіт	Строк розробки
1	Створення ідеї	4 дні
2	Планування робіт	2 дні
3	Розробка дизайну	3 дні
4	Створення спрайтів та анімації	7 днів
5	Написання ігрового процесу гри	7 днів
6	Створення аудіо супроводу	3 дні
7	Створення головного меню	2 дні
8	Збір усіх елементів ігрового додатку	2 дні
9	Інтеграція реклами та створення рейтингу	2 дні
	Загальна тривалість робіт	32 дні

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Фінальним кроком у розробці мобільного ігрового додатку є завантаження його на цифровий ринок. Так як платформа розробки відеогри це Android, то і завантажувати його треба на Play Market. Для цього треба створити аккаунт у застосунку Google Developer, зібрати проект в один файл .apk та завантажити його у цифровий ринок Play Market.

ДОДАТОК Б.
Планування робіт

Основною метою даного дослідження є розробка ігрового додатку під назвою «Expulsion of Invaders» жанру двовимірної аркади для мобільної платформи Android на цікаву або актуальну тематику на базі ігрового рушію Unity.

Продукт призначений для розваг на мобільному пристрої, користувачі якого зможуть зануритися у цікавий ігровий процес. Також вони матимуть можливість зберегти свій рекорд серед інших гравців.

Розробка повинна завершитися до кінця 4-го курсу, тобто до червня місяця. Даний ігровий додаток буде поширюватися у мережі інтернет через цифровий ринок Play Market. Щоб проект був успішним та мав великі показники завантажень, треба на концептуальному етапі правильно визначити мету ігрового додатку за допомогою SMART-методу. Результати деталізації методом SMART показані у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific	Розробка ігрового додатку під назвою «Expulsion of Invaders» жанру двовимірної аркади для мобільної платформи «Android» на цікаву або актуальну тематику на базі ігрового рушію Unity
Measurable	Набраний максимальний рекорд
Achievable	**Січень – Лютий:** планування архітектури (ТЗ, вибір стеку технологій, опис моделей даних). **Лютий – Травень:** розробка ігрового додатку (написання скриптів, малювання спрайтів та анімації, звуко-інженерія тощо) **Травень – Червень:** тестування та виправлення багів
Relevant	Для розвитку ігрової індустрії та власних навичок у розробці ігрових додатків
Time-framed	до кінця 4 курсу (червень 2022р.).

Планування змісту робіт

Структура розбивки робіт (WBS) – це візуальна, ієрархічна та орієнтована на виконання декомпозиція проекту, яку треба провести для виділення та досягнення цілей проекту. Усі етапи роботи над проектом викладені в структурній діаграмі розбивки робіт, що робить її важливим інструментом планування проекту.

Менеджери проектів використовують програмне забезпечення для керування проектами, щоб скласти та виконати структуру розбивки робіт. У поєднанні з діаграмою Ганта, яка включає рівні WBS та ієрархію завдань, програмне забезпечення для керування проектами може бути особливо ефективним для планування проектів.

WBS структура включає в себе такі елементи, як:

– завдання – номер завдання, ідентифікатор, назва та його опис;

– виконавець завдання – відповідальний за вчасне виконання завдання;

– залежність завдань – поєднання декількох завдань разом, якщо їх виконання залежить від виконання інших;

– дата початку та завершення завдання – визначає дедлайни завдання та проекту в цілому;

– тривалість – оцінює час, який треба витратити на виконання завдання;

– оцінка роботи – фактична кількість часу роботи, потрібної для виконання завдання (об’єднання всіх ресурсних годин разом, якщо працюєте паралельно);

– статус завдання – на якому етапі знаходиться завдання в даний момент часу;

– діаграма Ганта – візуалізація WBS із завданнями, представленими графічно з часовою шкалою.

На рисунку Б.1 зображено WBS діаграму, яка була створена згідно з діаграмою Ганта та нашим технічним завданням.

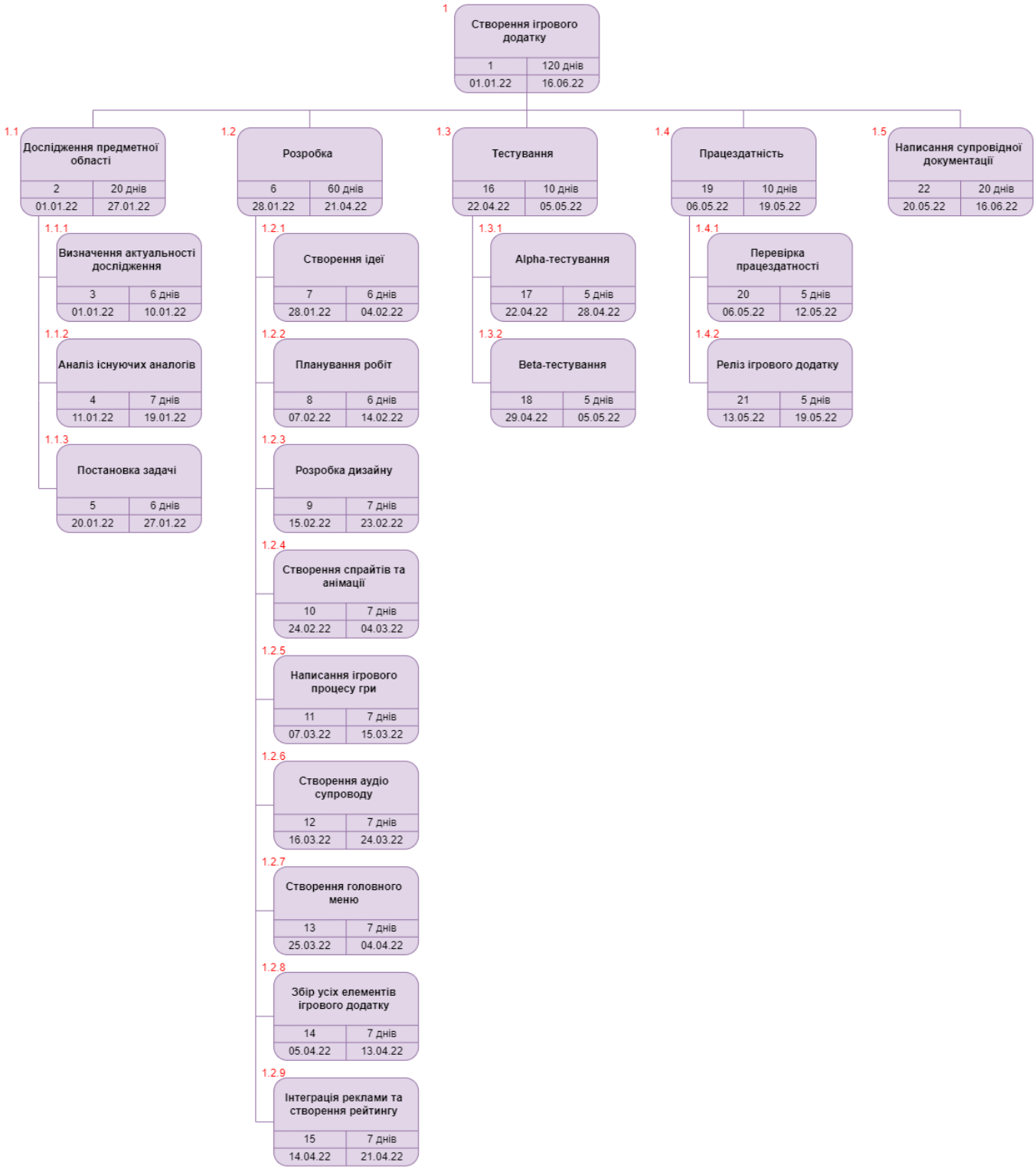


Рисунок Б.1 – WBS структура робіт проекту

Планування структури виконавців

Організаційна структура розбивки (OBS) – це структура організації проекту у вигляді ієрархічної моделі для визначення відповідальності, підзвітності, управління та затвердження всіх дозволених обсягів робіт.

Структура розбивки організації об’єднує подібні завдання проекту та поєднує їх із структурою організації. Організаційна структура розбивки використовується у визначенні відповідальності за управління проектом та виконавців тих чи інших завдань. Також вона забезпечує організаційну перспективу проекту, а не вказує на конкретні завдання, які треба виконати. Ієрархічна структура розбивки організації дає змогу для керування інформацією про проект на вищих рівнях.

Щоб розробити структуру розподілу організації:

– зобразити всю організацію як ієрархію;

– визначте всі відділи та проектні групи;

– укажіть функціональні групи (де розподіляється вартість роботи, яку виконує користувач) і групи схвалення (хто схвалює роботу, яку виконує користувач, і будь-які затвердження часу відпустки) для кожного учасника проекту.

На рисунку Б.2 зображено OBS діаграму, яка була створена згідно з діаграмою Ганта та нашим технічним завданням.

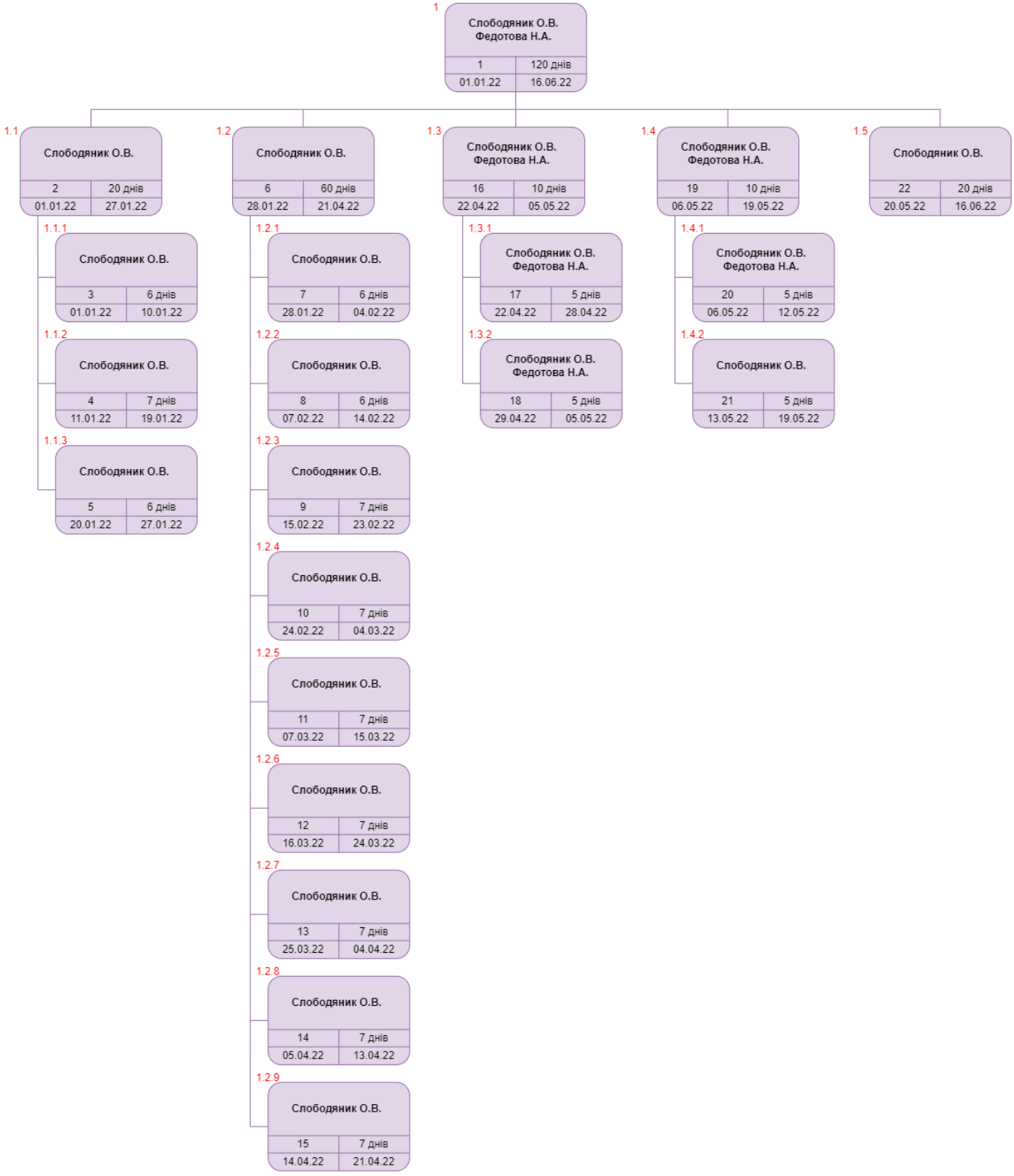


Рисунок Б.2 – OBS структура робіт проекту

Ім’я виконавці ігрового додатку та їх роль наведено в таблиці Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	**Ім’я**	**Проектна роль**
Розробник	Слободяник О.В.	Виконує розробку проекту
Проектувальник	Слободяник О.В.	Створює геймплей та програмну частину ігрового додатку
Дизайнер	Слободяник О.В.	Виконує дизайн усіх ресурсів ігрового додатку
Звуковий-інженер	Слободяник О.В.	Виконує зведення та мастеринг звукового супроводу проекту
Тестувальник	Слободяник О.В.	Виконує тестування та надання баг-репорту ігрового додатку
Керівник проекту	Федотова Н.А.	Формує технічне завдання на розробку ігрового додатку

Діаграма Ганта

Діаграма Ганта — це інструмент управління проектами, який допомагає планувати та складати розклад проектів будь-яких розмірів. Основними елементами діаграми Ганта є завдання проекту та їх часові шкали, які представленні у вигляді горизонтальної гістограми. Вона показує дати початку та завершення завдань та проекту в цілому. Це корисно для моніторингу виконання завдань у будь-якому проекті. Оскільки етап виконання завдання виконаний у вигляді стовпчастої діаграми, ви можете швидко перевірити прогрес та побачити:

– візуальне відображення всього проекту;

– терміни та терміни виконання всіх завдань;

– відносини та залежності між різними видами діяльності;

– фази проекту.

На рисунку Б.3 зображено діаграму Ганта, яка була створена згідно з нашим технічним завданням.

	Сд	Название задачи	Длите	Начало	Окончани
1	**1**	**Створення ігрового додатку**	**120 дней**	**Сб 01.01.22**	**Чт 16.06.22**
2	**1.1**	**Дослідження предметної області**	**20 дней**	**Сб 01.01.22**	**Чт 27.01.22**
3	1.1.1	Визначення актуальності дослідження	6 дней	Пн 03.01.22	Пн 10.01.22
4	1.1.2	Аналіз існуючих аналогів	7 дней	Вт 11.01.22	Ср 19.01.22
5	1.1.3	Постановка задачі	6 дней	Чт 20.01.22	Чт 27.01.22
6	**1.2**	**Розробка**	**60 дней**	**Пт 28.01.22**	**Чт 21.04.22**
7	1.2.1	Створення ідеї	6 дней	Пт 28.01.22	Пт 04.02.22
8	1.2.2	Планування робіт	6 дней	Пн 07.02.22	Пн 14.02.22
9	1.2.3	Розробка дизайну	7 дней	Вт 15.02.22	Ср 23.02.22
10	1.2.4	Створення спрайтів та анімації	7 дней	Чт 24.02.22	Пт 04.03.22
11	1.2.5	Написання ігрового процесу гри	7 дней	Пн 07.03.22	Вт 15.03.22
12	1.2.6	Створення аудіо супроводу	7 дней	Ср 16.03.22	Чт 24.03.22
13	1.2.7	Створення головного меню	7 дней	Пт 25.03.22	Пн 04.04.22
14	1.2.8	Збір усіх елементів ігрового додатку	7 дней	Вт 05.04.22	Ср 13.04.22
15	1.2.9	Інтеграція реклами та створення рейтингу	6 дней	Чт 14.04.22	Чт 21.04.22
16	**1.3**	**Тестування**	**10 дней**	**Пт 22.04.22**	**Чт 05.05.22**
17	1.3.1	Alpha-тестування	5 дней	Пт 22.04.22	Чт 28.04.22
18	1.3.2	Beta-тестування	5 дней	Пт 29.04.22	Чт 05.05.22
19	**1.4**	**Працездатність**	**10 дней**	**Пт 06.05.22**	**Чт 19.05.22**
20	1.4.1	Перевірка працездатності	5 дней	Пт 06.05.22	Чт 12.05.22
21	1.4.2	Реліз ігрового додатку	5 дней	Пт 13.05.22	Чт 19.05.22
22	1.5	Написання супровідної дукоментації	20 дней	Пт 20.05.22	Чт 16.06.22

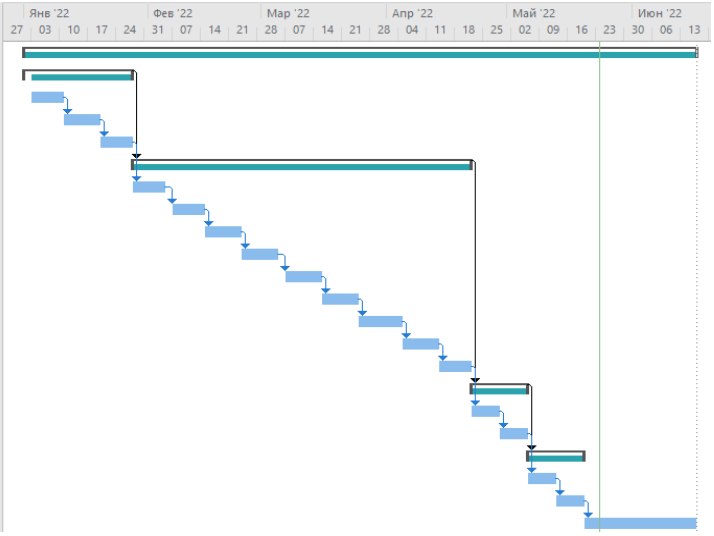


Рисунок Б.3 – Календарний графік проекту

Управління ризиками проекту

Під час виконання якісної та швидкої оцінки ризиків треба визначити ризики, які повинні бути усунені першочергово. Залежно від рівня впливу та ймовірності ризику – реагування буде відповідне. Наступним кроком є проведення кількісного та якісного оцінювання ризиків. Таке оцінювання можуть проводити одночасно або окремо один від одного, що залежить від рівня забезпечення проекту або самих потреб його. У таблиці Б.3 представлено шкалу для класифікації ризиків за впливом та ймовірністю впливу на ігровий додаток.

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю та величиною впливу

Оцінка	**Ймовірність**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для зменшення негативних впливів ризиків та ймовірності їх виникнення на проект виконують планування реагування на них. До нього входить визначення ефективності розробки ігрового додатку та оцінка наслідків впливу на нього. Оцінювання ризиків відбувається за параметрами, що описані в таблиці Б.3. У результаті планування реагування отримано матрицю ймовірності виникнення ризиків та впливу них на ігровий додаток, що зображена на рисунку Б.4. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

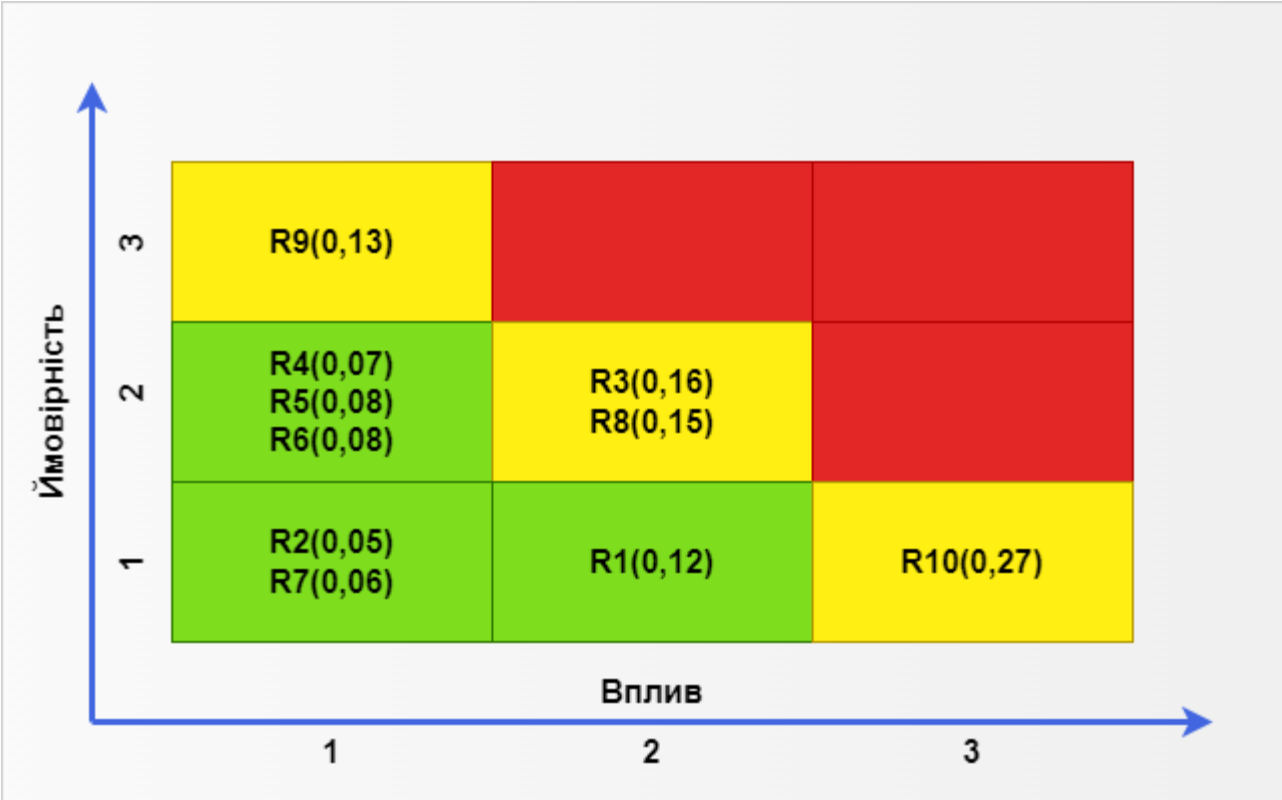


Рисунок Б.4 – Матриця ймовірності

Класифікація ризиків за рівнем, відповідно до отриманих даних, представлена у таблиці Б.4. У таблиці Б.5 описано ризики, ймовірність їх, ранг та стратегії реагування на кожний з них.

Таблиця Б.4 – Шкала оцінювання за рівнем ризику.

№	**Назва**	**Межі**	**Ризики, які входять (номера)**
1	Прийнятні	0,005≤R≤0,12	1, 2, 4, 5, 6, 7
2	Виправдані	0,12<R<0,36	3, 8, 9, 10
3	Недопустимі	0,36≤R≤0,76	-

Таблиця Б.5 – Матриця ймовірності та впливу згідно проекту

ID	Статус ризику	Опис	Ймовірність	Вплив	Ранг	План А	Тип стратегії реагування	План Б
1	Відкритий	Баги інтерфейсу	Малий	Малий	0,12	Створювати динамічний інтерфейс під будь який дозвіл екрану	Зменшення	Створити інтерфейс під найпопулярніші дозволи екрану
2	Відкритий	Нестабільність програмного забезпечення	Дуже малий	Середній	0,05	Протестувати програмне забезпечення	Зменшення	Перестановити програмне забезпечення
3	Відкритий	Баги у скриптовій частині	Дуже малий	Дуже великий	0,16	Дотримуватися принципу SOLID	Ухилення	Провести рефакторинг коду
4	Відкритий	Авторські права на ресурсах ігрового додатку	Дуже малий	Дуже великий	0,07	Перевірити ресурси на авторство	Ухилення	Знайти альтернативні ресурси, або створити власні
5	Відкритий	Неправильна компресія звукових ресурсів	Дуже малий	Малий	0,08	Налаштувати мульти-бендову компресію	Зменшення	Лімітувати звук до 0db
6	Відкритий	Часте внесення змін у ТЗ	Дуже малий	Малий	0,08	Визначити точну мету та потреби проекту	Зменшення	Перевизначити технічне завдання
7	Відкритий	Низька кваліфікація розробника	Дуже малий	Великий	0,06	Знати усі стеки технологій які будуть використовуватися	Зменшення	Знайти аналог технологіям
8	Відкритий	Неоптимальний розподіл часу	Малий	Середній	0,15	Розробити діаграму Ганта та визначити терміни	Ухилення	Перевизначити строки різних етапів на інші терміни
9	Відкритий	Поява альтернативного продукту	Дуже великий	Дуже малий	0,13	Провести дослідження аналогів та слідкувати за трендами	Передача	
10	Відкритий	Баги на етапі тестування	Дуже великий	Малий	0,27		Ухилення	Провести рефакторинг коду

ДОДАТОК В.
Програмний код

Весь програмний код, метадані та додаткові плагіни проекту знаходяться за посиланням: https://github.com/AlexanderSmasher/Expulsion-Of-Invaders

Далі наведено найголовніші скрипти ігрового додатку.

PlayerController.cs

```
using System;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    [SerializeField] private float Speed;
    [SerializeField] private GameObject ExplosionPrefab;
    [SerializeField] private GameObject DamagePrefab;
    [SerializeField] private float HP;
    [SerializeField] private HPBarUI HPBar;
    [SerializeField] private MarkCounter Mark;
    [SerializeField] private GameController Controller;
    [SerializeField] private AdMobController AMC;

    private float XMin;
    private float XMax;
    private float YMin;
    private float YMax;
    private float Damage;
    private float BarSize = 1f;
    private int TryCount;

    private Vector2 Padding() => new Vector2(GetComponent<BoxCollider2D>().size.x / 2,
GetComponent<BoxCollider2D>().size.y / 2);
    private void FindBounds()
    {
        Camera mainCamera = Camera.main;
        XMin = mainCamera.ViewportToWorldPoint(new Vector3(0, 0, 0)).x + Padding().x;
        XMax = mainCamera.ViewportToWorldPoint(new Vector3(1, 0, 0)).x - Padding().x;
        YMin = mainCamera.ViewportToWorldPoint(new Vector3(0, 0, 0)).y + Padding().y;
        YMax = mainCamera.ViewportToWorldPoint(new Vector3(0, 1, 0)).y - Padding().y;
    }
    private void DamageHPBar()
    {
        if (HP > 0)
        {
            HP -= 1;
```

```
            BarSize -= Damage;
            HPBar.SetHP(BarSize);
        }
    }
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "EnemysBullet")
        {
            DamageHPBar();
            Destroy(collision.gameObject);
            GameObject damage = Instantiate(DamagePrefab, collision.transform.position,
Quaternion.identity);
            Destroy(damage, 0.6f);
            if (HP <= 0)
            {
                TryCount++;
                PlayerPrefs.SetInt("SaveTry", TryCount);
                Destroy(gameObject);
                GameObject explosion = Instantiate(ExplosionPrefab, transform.position,
Quaternion.identity);
                Destroy(explosion, 0.6f);
                if (TryCount % 3 == 0)
                {
                    Controller.ShowAd();
                    AMC.ShowAd();
                }
                else
                    Controller.GameOver();
            }
        }
        if (collision.tag == "Mark")
        {
            Destroy(collision.gameObject);
            Mark.AddMark();
        }
    }
    private void Start()
    {
        AMC.InterAd.OnAdClosed += HandleOnAdClosed;
        FindBounds();
        Damage = BarSize / HP;
        TryCount = PlayerPrefs.GetInt("SaveTry");
    }
    private void Update()
    {
        if (Input.touchCount == 1)
            Move();
    }
    private void Move()
    {
        Vector2 startTouchPos = Camera.main.ScreenToWorldPoint(Input.GetTouch(0).position);
        if (GetComponent<BoxCollider2D>().bounds.Contains(startTouchPos))
        {
```

```
            float offsetPosX = Mathf.Clamp(startTouchPos.x, XMin,XMax);
            float offsetPosY = Mathf.Clamp(startTouchPos.y, YMin, YMax);
            transform.position = new Vector2(offsetPosX, offsetPosY);
        }
    }
    public void HandleOnAdClosed(object sender, EventArgs args) => Controller.GameOver();
}
```

ShotController.cs

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

public class ShotController : MonoBehaviour
{
    [SerializeField] private GameObject PlayerShot;
    [SerializeField] private Transform Cannon1;
    [SerializeField] private Transform Cannon2;
    [SerializeField] private List<GameObject> Flash;
    [SerializeField] private float ShootDelay;

    private void Fire()
    {
        Instantiate(PlayerShot, Cannon1.position, Quaternion.identity);
        Flash[0].SetActive(true);
        Instantiate(PlayerShot, Cannon2.position, Quaternion.identity);
        Flash[1].SetActive(true);
    }
    private IEnumerator Shoot()
    {
        while (true)
        {
            yield return new WaitForSeconds(ShootDelay);
            Fire();
            yield return new WaitForSeconds(0.08f);
            Flash[0].SetActive(false);
            Flash[1].SetActive(false);
        }
    }
    private void Start() => StartCoroutine(Shoot());
    private void Awake()
    {
        Flash[0].SetActive(false);
        Flash[1].SetActive(false);
    }
}
```

EnemyController.cs

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

```
using UnityEngine;

public class EnemyController : MonoBehaviour
{
    [SerializeField] private Enemy EnemyType;
    [SerializeField] private GameObject EnemyShotPrefab;
    [SerializeField] private List<Transform> EnemyCannon;
    [SerializeField] private List<GameObject> EnemyFlash;
    [SerializeField] private GameObject EnemyExplosionPrefab;
    [SerializeField] private GameObject EnemyDamagePrefab;
    [SerializeField] private float EnemyShootDelay;
    [SerializeField] private float EnemySpeed;
    [SerializeField] private float HP;
    [SerializeField] private HPBarController HPBar;
    [SerializeField] private GameObject MarkPrefab;

    private float Damage;
    private float BarSize = 1f;
    private bool EnemyIsReload = true;

    private void Fire(Enemy type)
    {
        if (type == Enemy.MI8)
        {
            Instantiate(EnemyShotPrefab, EnemyCannon[0].position, Quaternion.identity);
            EnemyFlash[0].SetActive(true);
        }
        else if (type == Enemy.TU22)
        {
            if (EnemyIsReload)
            {
                Instantiate(EnemyShotPrefab, EnemyCannon[0].position, Quaternion.identity);
                Instantiate(EnemyShotPrefab, EnemyCannon[1].position, Quaternion.identity);
                EnemyFlash[0].SetActive(true);
                EnemyFlash[1].SetActive(true);
                EnemyIsReload = false;
            }
            else
            {
                Instantiate(EnemyShotPrefab, EnemyCannon[2].position, Quaternion.identity);
                Instantiate(EnemyShotPrefab, EnemyCannon[3].position, Quaternion.identity);
                EnemyFlash[2].SetActive(true);
                EnemyFlash[3].SetActive(true);
                EnemyIsReload = true;
            }
        }
        else if (type == Enemy.SU30)
        {
            if (EnemyIsReload)
            {
                Instantiate(EnemyShotPrefab, EnemyCannon[0].position, Quaternion.identity);
                EnemyFlash[0].SetActive(true);
                EnemyIsReload = false;
```

```
        }
        else
        {
            Instantiate(EnemyShotPrefab, EnemyCannon[1].position, Quaternion.identity);
            EnemyFlash[1].SetActive(true);
            EnemyIsReload = true;
        }
    }
    else if (type == Enemy.SU35)
    {
        if (EnemyIsReload)
        {
            Instantiate(EnemyShotPrefab, EnemyCannon[0].position, Quaternion.identity);
            Instantiate(EnemyShotPrefab, EnemyCannon[1].position, Quaternion.identity);
            EnemyFlash[0].SetActive(true);
            EnemyFlash[1].SetActive(true);
            EnemyIsReload = false;
        }
        else
        {
            Instantiate(EnemyShotPrefab, EnemyCannon[2].position, Quaternion.identity);
            EnemyFlash[2].SetActive(true);
            EnemyIsReload = true;
        }
    }
}
private IEnumerator Shoot(Enemy type)
{
    if (type == Enemy.MI8)
    {
        while (true)
        {
            yield return new WaitForSeconds(EnemyShootDelay);
            Fire(EnemyType);
            yield return new WaitForSeconds(0.08f);
            EnemyFlash[0].SetActive(false);
        }
    }
    if (type == Enemy.SU35)
    {
        while (true)
        {
            yield return new WaitForSeconds(EnemyShootDelay);
            Fire(EnemyType);
            yield return new WaitForSeconds(0.08f);
            EnemyFlash[0].SetActive(false);
            EnemyFlash[1].SetActive(false);
            EnemyFlash[2].SetActive(false);
        }
    }
    if (type == Enemy.SU30)
    {
        while (true)
```

```
            {
                yield return new WaitForSeconds(EnemyShootDelay);
                Fire(EnemyType);
                yield return new WaitForSeconds(0.08f);
                EnemyFlash[0].SetActive(false);
                EnemyFlash[1].SetActive(false);
            }
        }
        if (type == Enemy.TU22)
        {
            while (true)
            {
                yield return new WaitForSeconds(EnemyShootDelay);
                Fire(EnemyType);
                yield return new WaitForSeconds(0.08f);
                EnemyFlash[0].SetActive(false);
                EnemyFlash[1].SetActive(false);
                EnemyFlash[2].SetActive(false);
                EnemyFlash[3].SetActive(false);
            }
        }
    }
    private void Move() => transform.Translate(Vector2.down * EnemySpeed * Time.deltaTime);
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "PlayersBullet")
        {
            DamageHPBar();
            Destroy(collision.gameObject);
            GameObject damage = Instantiate(EnemyDamagePrefab, collision.transform.position,
Quaternion.identity);
            Destroy(damage, 0.6f);
            if (HP <= 0)
            {
                Instantiate(MarkPrefab, transform.position, Quaternion.identity);
                Destroy(gameObject);
                GameObject explosion = Instantiate(EnemyExplosionPrefab, transform.position,
Quaternion.identity);
                Destroy(explosion, 0.6f);
            }
        }
    }
    private void DamageHPBar()
    {
        if (HP > 0)
        {
            HP -= 1;
            BarSize -= Damage;
            HPBar.SetHP(BarSize);
        }
    }
    private void Start()
    {
```

```
        for (int i = 0; i < EnemyFlash.Count; i++)
            EnemyFlash[i].SetActive(false);
        StartCoroutine(Shoot(EnemyType));
        Damage = BarSize / HP;
    }
    private void Update() => Move();
}
```