ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ІГРОВИЙ ДОДАТОК ДЛЯ ТЕЛЕФОНУ «AIR ATTACK»

зі спеціальності 122 Комп’ютерні науки освітня
програма «Комп’ютерні науки» ступеня магістра

Виконавець роботи Береговий Ростислав Романович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник доцент, к.ф.-м.н., Черненко Оксана Олексіївна
_____«____»_____ 202_ р.
(підпис)

Рецензент

ПОЛТАВА 2024

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 202__ р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему Ігровий додаток для телефону «Air Attack»
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Береговий Ростислав Романович
Затверджена наказом ректора № 206-Н від «27» жовтня 2023 р.
Термін подання студентом роботи «___»_____ 202__ р.
Вихідні дані до кваліфікаційної роботи: публікації з теми, системи дистанційного навчання.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ЗМІСТ
ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Мобільний додаток
2.1.1 Види мобільних додатків
2.1.2 Типи мобільних додатків
2.1.3 Особливості мобільних додатків
2.2 Мобільна гра
2.2.1 Жанри мобільних ігор
2.3 Системи, середовища програмування, середовища для розробки програмного забезпечення
2.4 Алгоритм роботи гри-конкурента «1945 Air Force»
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1 Правила ігрового додатку
3.2 Проектування ігрового додатку
3.3 Алгоритм роботи ігрового додатку
3.4 Стиль графіки
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
3.1 Розробка графічних елементів
3.2 Програмна реалізація ігрового додатку
3.3 Опис процесу гри
ВИСНОВКИ
СПИСОК ЛІТЕРАТУРИ
ДОДАТОК А. ДИСК З МАТЕРІАЛАМИ

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Ольховська О.В.		
Інформаційний огляд	Ольховська О.В.		
Теоретична частина	Ольховська О.В.		
Практична частина	Ольховська О.В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 202___ р.

Здобувач вищої освіти _____ Береговий Ростислав Романович

(підпис)

Науковий керівник _____ доцент, к.ф.-м.н., Черненко Оксана Олексіївна

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

Результати захисту кваліфікаційної роботи
Кваліфікаційна робота оцінена на
_____
(балів, оцінка за національною шкалою, оцінка за ECTS)
Протокол засідання ЕК № ____ від «____» _____ 2023 р.
Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

Погоджено
Науковий керівник _____
к.ф.-м.н., Оксана ЧЕРНЕНКО
«____» _____ 202__ р.

План

кваліфікаційної роботи на тему
Проектування чат-бота студента ПУЕТ
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Береговий Ростислав Романович
ЗМІСТ

Здобувач вищої освіти _____ Ростислав БЕРЕГОВИЙ
«____» _____ 202__ р.

АНОТАЦІЯ

Записка: 50 с., 23 рис., 17 джерел.

Мета роботи – проектування та реалізація ігрового додатку для телефону «Air Attack».

Об’єкт розробки – алгоритм роботи ігрового додатку для телефону «Air Attack».

Методи дослідження та інформаційне забезпечення – середовище візуальної розробки програм Visual Studio Code та об’єктно-орієнтована мова програмування C#.

Результати дослідження. Зроблено огляд відомого програмного забезпечення для створення ігрового додатку для телефона, виявлено позитивні та негативні сторони. Розроблено алгоритм ігрового додатку для телефону «Air Attack», побудовано його блоксхему.

Рекомендації щодо використання результатів дослідження. Програмний продукт, що реалізує ігровий додаток для телефону «Air Attack».

ЗМІСТ

ВСТУП

У сучасному світі мобільні додатки стають все більш популярними та відомими серед користувачів різних вікових груп. Вони не лише розважають, але й пропонують нові форми інтерактивного проведення часу. Одним з яскравих представників таких додатків є гра «Air Attack.

У цій магістерській роботі ми збираємося проаналізувати та спроектувати основні аспекти цієї гри, включаючи її ігровий процес, графічне оформлення, механіку управління та рівні складності. Ми також розглянемо історію розвитку цього жанру ігор, вплив технологічних інновацій на розвиток мобільних геймінгових додатків.

Цей проект дозволить нам глибше зануритися в світ мобільних ігор, вивчити їх розробку та проектування, а також оцінити потенціал цього сегмента ринку в контексті сучасної індустрії розваг.

Мета магістерської роботи – проектування та реалізація ігрового додатку для телефону «Air Attack».

Об'єктом розробки є алгоритм роботи ігрового додатку для телефону «Air Attack».

Предмет розробки — ігровий додаток для телефону «Air Attack».

Перелік використаних методів полягає в застосуванні граматик, мов, що задаються граматиками.

Магістерська робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі розглянуто типи мобільних ігор, системи, середовища програмування, середовища для розробки програмного забезпечення, алгоритм роботи гри-конкурента. У третьому розділі представлено правила ігрового додатку, проектування ігрового додатку, алгоритм роботи ігрового додатку, стиль графіки. У четвертому розділі представлено розробка графічних елементів, програмна реалізація ігрового додатку, опис процесу гри.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Головним завданням магістерської роботи з фаху – є створення елементів мобільної гри, програмування елементів програмного забезпечення мобільної гри «Air Attack»

Завдання магістерської роботи з фаху:

- Розглянути матеріал з теми «Проектування ігрового додатку для телефону «Air Attack» ».
- Розробити елементи ігрового додатку для телефону. А саме розробити моделі літаків. Розробити карту гри, а саме: поля, ліси, степи, водойма. Розробити елементи інтерфейсу, а саме: головне меню гри, меню паузи та меню наступного рівня.
- Запрограмувати елементи ігрового додатку для телефону. Програмування елементів має відбуватися у застосунках, а саме: для реалізації інтерфейсу, літаків, карти – Photoshop та Illustrator; для розробки ігрового процесу – Unity.
- Переглянути результати створення елементів ігрового додатку.

У процесі створення програмного забезпечення для ігрового додатку для телефону, необхідно врахувати наступні важливі елементи ігрового процесу:

- Після знищення ворожого літака має з’являтися монети для допомоги гравцю.
- У літака має бути 3 життi.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Мобільний додаток

Мобільний додаток (мобільна аплікація) – це програмне забезпечення, розроблене для роботи на мобільних пристроях, таких як смартфони, планшети або смарт-годинники. Ці додатки дозволяють користувачам виконувати різноманітні завдання, включаючи комунікацію, роботу, розваги, навчання, управління особистими фінансами та багато іншого.

2.1.1 Види мобільних додатків

За цільовою аудиторією:

- Корпоративні: Призначені для внутрішнього використання у компаніях.
- Споживчі: Розроблені для кінцевих користувачів, спрямовані на розваги, соціальні мережі тощо.

За платформою:

- iOS: Для пристроїв Apple.
- Android: Для смартфонів і планшетів на базі операційної системи Android.
- Гібридні: Розроблені для роботи на різних платформах.

За функціональністю:

- Ігрові: Зорієнтовані на розваги та геймінг.
- Продуктивність: Допомагають у плануванні, роботі та організації.
- Спілкування: Включають месенджери, соціальні мережі та інструменти для спілкування.

2.1.2 Типи мобільних додатків

Нативні додатки:

- Розроблені спеціально для однієї конкретної платформи, використовуючи мови програмування, такі як Swift або Java.
- Мають високу продуктивність і доступ до всіх функцій пристрою.

Веб-додатки:

- Запускаються у веб-браузері на мобільному пристрої.
- Не потребують встановлення через магазин додатків, але можуть мати обмежений доступ до функцій пристрою.

Гібридні додатки:

- Комбінують елементи нативних та веб-додатків.
- Написані на мовах, таких як HTML, CSS і JavaScript, і збираються в аплікації за допомогою фреймворків, таких як Ionic або React Native.

2.1.3 Особливості мобільних додатків

- Адаптивний дизайн: Мобільні додатки мають бути адаптованими до різних розмірів екранів та орієнтацій пристроїв.
- Безпека: Захист особистої інформації користувачів є пріоритетом для розробників.
- Інтуїтивний інтерфейс: Додатки повинні мати зрозумілий і легкий у використанні інтерфейс для забезпечення зручності користувачів.
- Офлайн доступ: Деякі додатки можуть працювати в режимі офлайн, забезпечуючи користувачам доступ до контенту навіть без підключення до інтернету.
- Оновлення: Регулярні оновлення для виправлення помилок, вдосконалення функціонала та забезпечення сумісності з новими версіями операційних систем. [2]

2.2 Мобільна гра

Мобільна гра — це така гра, в яку можна грати в русі, використовуючи свій смартфон. У ній, як і в комп'ютерній грі, є свої правила та цілі.

Мобільний геймінг — це завжди про невеликий дисплей, доступність в будь-який час, більш часті і короткі, в порівнянні з PC, ігрові сесії. Знаючи ці особливості, розробники продумують геймплей, складність і тривалість рівнів, створюють фішки для повернення користувача в гру.

2.2.1 Жанри мобільних ігор

Основні жанри мобільних ігор:

- Казуальні ігри. Казуальні ігри (casual games) орієнтовані на широку аудиторію і не вимагають від гравця значних зусиль або часу. Вони зазвичай мають простий геймплей та інтуїтивно зрозумілий інтерфейс. Приклади: Candy Crush Saga, Angry Birds, Fruit Ninja.
- Пазли. Пазли фокусуються на вирішенні логічних задач і головоломок. Ці ігри часто вимагають від гравців аналітичного мислення та стратегічного планування. Приклади: Tetris, Monument Valley, 2048.
- Екшн. Екшн-ігри насичені швидкими і динамічними подіями. Вони часто вимагають від гравців рефлексів і координації. Приклади: Temple Run, Subway Surfers, PUBG Mobile.
- Пригодницькі ігри. Пригодницькі ігри пропонують гравцям вивчати світи, вирішувати загадки і брати участь у захоплюючих історіях. Приклади: The Room, Life is Strange, Broken Sword.
- Рольові ігри . Рольові ігри дозволяють гравцям створювати та розвивати персонажів, занурюватися у фантастичні світи та виконувати

різноманітні квести. Приклади: Genshin Impact, Final Fantasy, The Elder Scrolls: Blades.

- Стратегії. Стратегічні ігри вимагають від гравців планування і прийняття рішень для досягнення довгострокових цілей. Вони можуть бути як покроковими, так і в реальному часі. Приклади: Clash of Clans, Civilization VI, Plague Inc.
- Симулятори. Симулятори імітують реальні або вигадані процеси. Вони можуть охоплювати широкий спектр тем, від управління містом до життя персонажів. Приклади: The Sims Mobile, SimCity BuildIt, Flight Pilot Simulator.
- Спортивні ігри. Спортивні ігри дозволяють гравцям брати участь у віртуальних спортивних змаганнях, таких як футбол, баскетбол, гонки тощо. Приклади: FIFA Mobile, NBA Live Mobile, Asphalt 9: Legends.
- Карткові і настільні ігри. Ці ігри перенесли популярні карткові та настільні ігри у цифровий формат. Вони часто включають як класичні ігри, так і нові розробки. Приклади: Hearthstone, Solitaire, Monopoly.
- Музичні і ритмічні ігри. Музичні ігри фокусуються на взаємодії з музикою та ритмом. Гравці повинні натискати на екран або виконувати інші дії у відповідності з ритмом музики. Приклади: Cytus, Beat Saber, Guitar Hero.

2.3 Системи, середовища програмування, середовища для розробки програмного забезпечення

Unity – це одна з найпопулярніших і найпотужніших платформ для розробки ігор і інтерактивного контенту. Вона надає розробникам інструменти для створення 2D, 3D, віртуальної реальності (VR) та доповненої реальності (AR) додатків. Завдяки своїй універсальності та

потужним можливостям, Unity використовується як інді-розробниками, так і великими студіями для створення ігор та інтерактивних додатків.

Основні характеристики Unity:

- Кросплатформеність: Unity дозволяє розробникам створювати додатки для різних платформ, включаючи мобільні пристрої (iOS, Android), комп'ютери (Windows, Mac, Linux), консолі (PlayStation, Xbox, Nintendo Switch), а також платформи для VR та AR (Oculus, HTC Vive, HoloLens).
- Інтуїтивно зрозумілий інтерфейс: Unity має зручний графічний інтерфейс користувача, який дозволяє легко створювати ігрові сцени, об'єкти та налаштовувати їхні властивості.
- Потужний движок: Unity забезпечує високу продуктивність і гнучкість завдяки своєму движку, який підтримує фізику, анімацію, рендеринг, аудіо та інші важливі аспекти розробки ігор.
- Підтримка C#: Для написання скриптів в Unity використовується мова програмування C#, що робить процес розробки більш структурованим і зрозумілим.
- Unity Asset Store: Unity має власний магазин активів, де розробники можуть купувати або завантажувати безкоштовні моделі, текстури, звукові ефекти, скрипти та інші ресурси, що значно прискорює процес розробки.
- Компонентна система: Unity використовує компонентну систему, де об'єкти сцени (GameObjects) можуть мати різні компоненти (скрипти, фізичні властивості, анімації тощо), що робить їх гнучкими та легко налаштовуваними.
- Підтримка VR і AR: Unity надає інструменти для розробки віртуальної та доповненої реальності, що робить його популярним вибором для створення контенту в цих сферах.

Основні застосування Unity:

- Розробка ігор: Unity використовується для створення ігор різних жанрів, від простих 2D-головоломок до складних 3D-екшенів і симуляторів.
- Інтерактивні додатки: Unity застосовується для створення інтерактивних додатків для освіти, тренінгів, архітектури та інших галузей.
- VR та AR контент: Завдяки підтримці VR і AR, Unity використовується для створення занурюючих віртуальних середовищ та доповненої реальності для різних пристроїв.
- Симуляції: Unity застосовується для розробки симуляцій, що використовуються в науці, інженерії, медицині та інших сферах.

Переваги Unity:

- Широка спільнота розробників: Unity має велику спільноту, що надає підтримку, ресурси та навчальні матеріали для новачків і професіоналів.
- Часті оновлення та підтримка: Unity постійно оновлюється, додаючи нові функції та покращуючи існуючі.
- Інтеграція з іншими інструментами: Unity добре інтегрується з іншими інструментами розробки, такими як Blender, Photoshop, Visual Studio та іншими. [3]

Unity Hub – це офіційний інструмент від Unity Technologies, створений для управління встановленнями Unity Editor, проектами та ліцензіями. Він надає зручний інтерфейс для завантаження та встановлення різних версій Unity, створення та відкриття проектів, а також налаштування параметрів розробки.

Основні функції Unity Hub:

- Управління версіями Unity: Unity Hub дозволяє легко встановлювати, оновлювати та видаляти різні версії Unity Editor. Це особливо корисно для розробників, які працюють над різними проектами, що потребують різних версій Unity.

- Створення та управління проектами: За допомогою Unity Hub можна створювати нові проекти або відкривати існуючі. Він також надає доступ до останніх проектів, що були відкриті, що дозволяє швидко продовжити роботу.
- Ліцензування: Unity Hub дозволяє керувати ліцензіями, зокрема активувати нові ліцензії або оновлювати наявні. Це включає як безкоштовні, так і платні версії.
- Модульні установки: Unity Hub дозволяє додавати або видаляти модулі для різних платформ (наприклад, Android, iOS, WebGL), що дає можливість налаштовувати середовище розробки відповідно до потреб проекту.
- Доступ до навчальних матеріалів: Unity Hub надає доступ до офіційних навчальних матеріалів та курсів Unity Learn, що допомагає розробникам вивчати нові можливості та техніки роботи з Unity.

Переваги використання Unity Hub:

- Централізоване управління: Unity Hub дозволяє управляти всіма аспектами розробки в одному місці, що спрощує процес розробки і знижує ймовірність помилок.
- Легка зміна версій: Можливість легко змінювати версії Unity дозволяє розробникам швидко адаптуватися до вимог різних проектів.
- Оптимізація процесу розробки: Unity Hub автоматизує багато процесів, що дозволяє зосередитися на розробці самого проекту, а не на налаштуванні середовища.

Як встановити та використовувати Unity Hub:

1. Завантаження та встановлення: Завантажити Unity Hub можна з офіційного сайту Unity. Після завантаження, слідуйте інструкціям для встановлення на вашій операційній системі (Windows, macOS, Linux).

2. Створення облікового запису: Після встановлення необхідно увійти або створити обліковий запис Unity. Це дозволить вам керувати ліцензіями та отримувати доступ до додаткових ресурсів.
3. Встановлення версій Unity Editor: В Unity Hub перейдіть до вкладки «Installs» і натисніть «Add» для вибору та встановлення потрібної версії Unity Editor. Можна вибрати додаткові модулі для конкретних платформ.
4. Створення проекту:Перейдіть до вкладки «Projects» і натисніть «New» для створення нового проекту. Виберіть версію Unity та шаблон проекту, щоб розпочати роботу. [4]

2.4 Алгоритм роботи гри-конкурента «1945 Air Force»

«1945 Air Force» – це аркадна стрілялка, де гравець керує літаком, стріляє ворожі літаки, збирає бонуси та уникає ворожих снарядів. Ось узагальнений алгоритм роботи гри:

1. Запуск гри
   1.1. Ініціалізація системи:
      1.1.1. Завантаження ресурсів (графіка, звуки, анімації).
      1.1.2. Ініціалізація ігрового двигуна.
      1.1.3. Завантаження даних профілю гравця (досягнення, високі результати, налаштування).
   1.2. Відображення головного меню:
      1.2.1. Відображення кнопок: "Грати", "Налаштування", "Досягнення", "Магазин".
      1.2.2. Очікування взаємодії з користувачем.
2. Початок гри
   2.1. Старт рівня:
      2.1.1. Завантаження даних поточного рівня (розташування ворогів, складність, типи ворогів).

2.1.2. Ініціалізація положення гравця та ворожих літаків.

2.2.Ігровий цикл:

2.2.1. Поки гравець не програв або не пройшов рівень:

2.2.1.1. Обробка введення: Отримання даних з сенсорного екрану (керування літаком).

2.2.1.2. Оновлення стану гри:

2.2.1.2.1. Рух літака гравця відповідно до введення.

2.2.1.2.2. Автоматична стрільба з літака гравця.

2.2.1.2.3. Генерація та рух ворожих літаків та їх снарядів.

2.2.1.2.4. Перевірка зіткнень (між гравцем і ворогами, між снарядами і літаками).

2.2.1.2.5. Збір бонусів та оновлення статусу гравця (збільшення життя, покращення зброї тощо).

2.2.1.3. Рендеринг:

2.2.1.3.1. Відображення гравця, ворогів, снарядів та фону.

2.2.1.3.2. Оновлення інтерфейсу (очки, життя, бонуси).

3. Перемога чи поразка

3.1.Перевірка умов завершення рівня:

3.1.1. Якщо всі вороги знищені або гравець досяг кінцевої точки рівня, рівень завершено.

3.1.2. Якщо літак гравця знищено, гра закінчується поразкою.

3.2.Обробка завершення рівня:

3.2.1. Якщо гравець виграв:

3.2.1.1. Показати екран завершення рівня з отриманими очками, бонусами.

3.2.1.2. Пропозиція перейти до наступного рівня або повернутися до головного меню.

3.2.2. Якщо гравець програв:

3.2.2.1. Показати екран поразки з опціями перезапуску рівня або повернення до головного меню.

4. Поза ігровий цикл

4.1. Магазин:

4.1.1. Відображення доступних для покупки предметів (нові літаки, покращення зброї тощо).

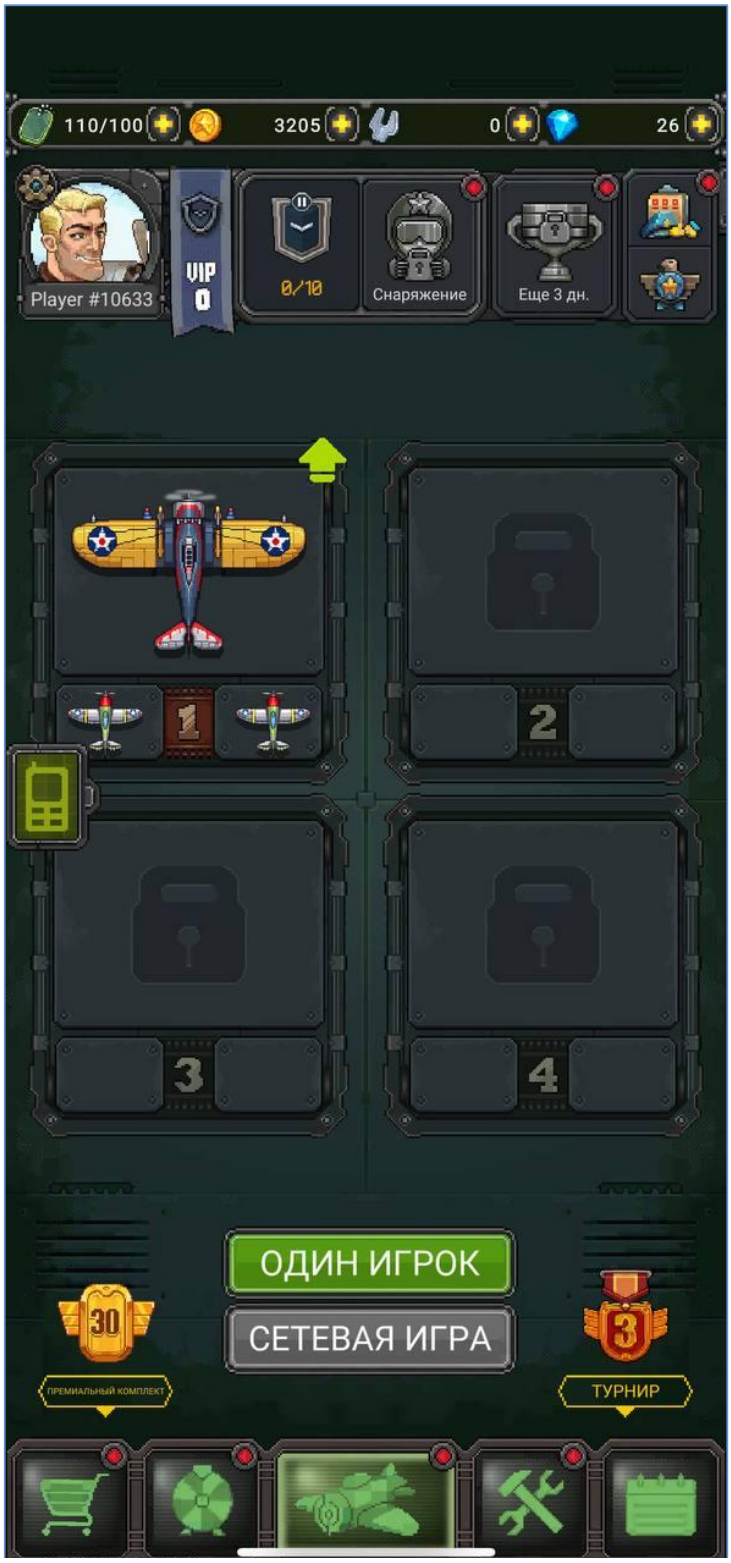
4.1.2. Обробка покупок за внутрішньо ігрову валюту або реальні гроші.

4.2. Досягнення та таблиця лідерів:

4.2.1. Відображення досягнень гравця та порівняння з іншими гравцями в таблиці лідерів.

4.3. Налаштування:

4.3.1. Зміна ігрових параметрів (гучність звуку, складність, управління тощо).



2.4.1 – Головне меню

На головному меню ми бачимо:

- Вибір подальших дій: магазин, ангар, вибір рівнів, календар нагород.
- Вибір режиму гри: один гравець або мережева гра.

- Вибір літака, котрий доступний гравцю.
- Елемент інтерфейсу, котрий показує валюту гравцю: жетони, монети, кристали.
- Профіль гравця, натиснувши на котрий, можна побачити досягнення гравця.

2.4.2 – Меню вибору рівня

На Рисунку 2.4.2 гравець має можливість обрати рівень, який він пройшов, або має пройти. Пройдений рівень підсвічується зеленим, ще не пройдений обводиться білим кругом, не доступний рівень має іконку замку. Після кожного пройденого рівня, над його іконкою з’являються зірки, що показують на яку оцінку пройдено рівень.

Гравець також має змогу обрати різні режими гри, а саме просто проходження рівнів або захист об’єктів від атак ворога.

Після збирання достатньої кількості зірок, гравець має можливість отримати контейнер, в якому міститься бонуси до гри.

2.4.3 – Ігровий процес

На Рисунку 2.4.3 гравець бачить ігровий процес та має безпосередню участь в ньому, а саме:

- Веде керування літаком.
- Контролює вогонь літака.

- Вражає ворожі цілі.
- Ухиляється від ворожих снарядів.
- Збирає бонуси.
- Використовує бонуси.

2.4.4 – Наявність ворожих снарядів у ігровому процесі

На Рисунку 2.4.4 гравець бачить ворожі снаряди, а саме червоні кілько подібні сфери червоного кольору, котрі наносять шкоду літаку гравця.

2.4.5 – Наявність бонусів під час ігрового процесу

На Рисунку 2.4.5 гравець бачить наявність бонусів під час ігрового процесу. Ці бонуси допомагають гравцю в проходженні рівня та виконані задачі. Основними бонусами є:

- Збільшення броні.
- Збільшення шкоди.
- Збільшення радіусу пострілів.

2.4.6 – Меню магазину

На Рисунку 2.4.6 гравець бачить меню магазину, в якому може придбати нові літаки або комплекти з бонусами для ігрового процесу.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Правила ігрового додатку

У грі про літаки гравець має управляти своїм літаком, щоб уникати зіткнень із ворогами, знищувати супротивників і збирати бонуси.

Основна мета гри — знищити усі ворожі літаки, зберігаючи життя свого літака. Літак можна переміщувати в межах ігрового поля в усіх напрямках за допомогою сенсорного екрану, джойстика або нахилу пристрою. Постріли здійснюються постійно, не зважаючи на дії гравця.

Якщо літак зіштовхується з ворогом або ворожим набоєм гравець втрачає одне життя. Кожного разу, коли снаряд гравця влучає у ворога, з ворожого літака випадають монети.

Простір для гри обмежений екраном, і літак не може виходити за межі цього простору. Якщо гравець досягає нових рівнів, складність гри зростає: вороги стають швидшими та складнішими.

Гра завершується, якщо у гравця закінчуються всі життя. У разі успішного завершення всіх рівнів гравець перемагає.

Заборонено використовувати сторонні засоби чи модифікації для отримання переваг, і такі дії можуть анулювати досягнення гравця.

3.2 Проектування ігрового додатку

Проектування ігрового додатку для телефону почалось з розробки діаграми роботи інтерфейсу (Рисунок 3.2.1). На цій діаграмі ми можемо бачити, що гравця зустрічає «Стартове меню», в якому є можливість перейти до основних елементів інтерфейсу.

«Налаштування» - відповідають за стандартні налаштування гри, а саме: регулювання звуку, вибір мови.

«Карта рівнів» - відповідає за можливість гравця обирати рівень, який він хоче проходити, або який йому відкритий.

«Ангар літаків» - відповідає за елемент інтерфейсу в якому гравець може обрати свій літак або модифікувати його.

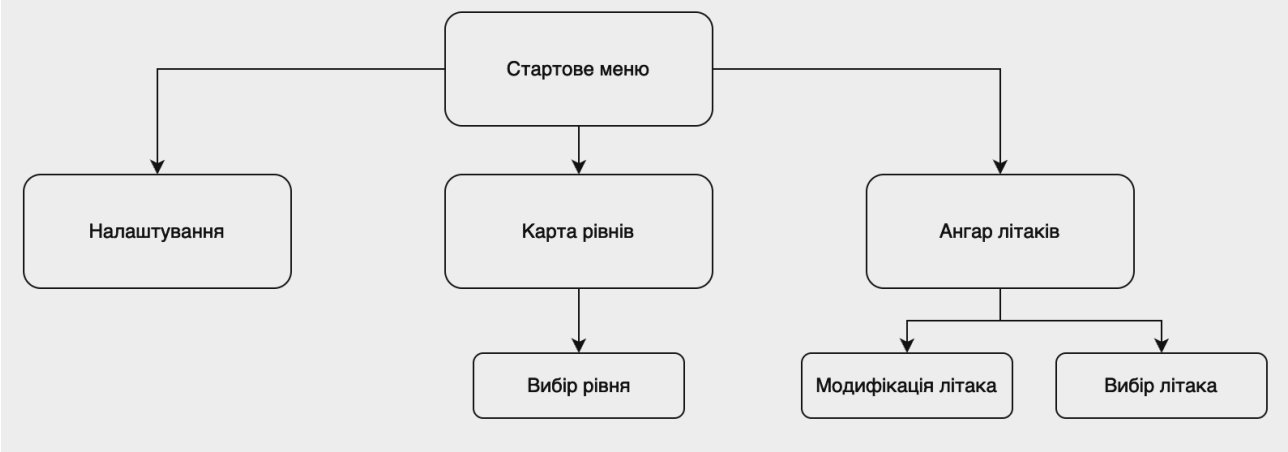


Рисунок 3.2.1 – Діаграма роботи меню

Наступним кроком в проектуванні ігрового додатку був пошук та аналіз моделей літаків, а також розробка самих моделей. Першим літаком був Міг-29. (Рисунок 3.2.2)

Рисунок 3.2.2 – Модель літака Міг-29

На моделі літака можна побачити позначки жовто-синього кольору, що означають українську символіку. Окрім символіки для помітки приналежності літака до України, літак має окрас «піксель» для маскування в повітрі. Ця модель безпосередньо використовується в ігровому процесі.

Рисунок 3.2.3 – Модель літака СУ-27

На Рисунку 3.2.3 зображена модель літака СУ-27. На основі цієї моделі розробляється модель літака для гри. Для команди гравця на модель літака буде наноситись окрас української авіації, як у прикладі з Міг-29, та додавання позначок української авіації. Для ворожої команди на модель літака буде наноситись окрас ворожої команди та позначки ворожої команди.

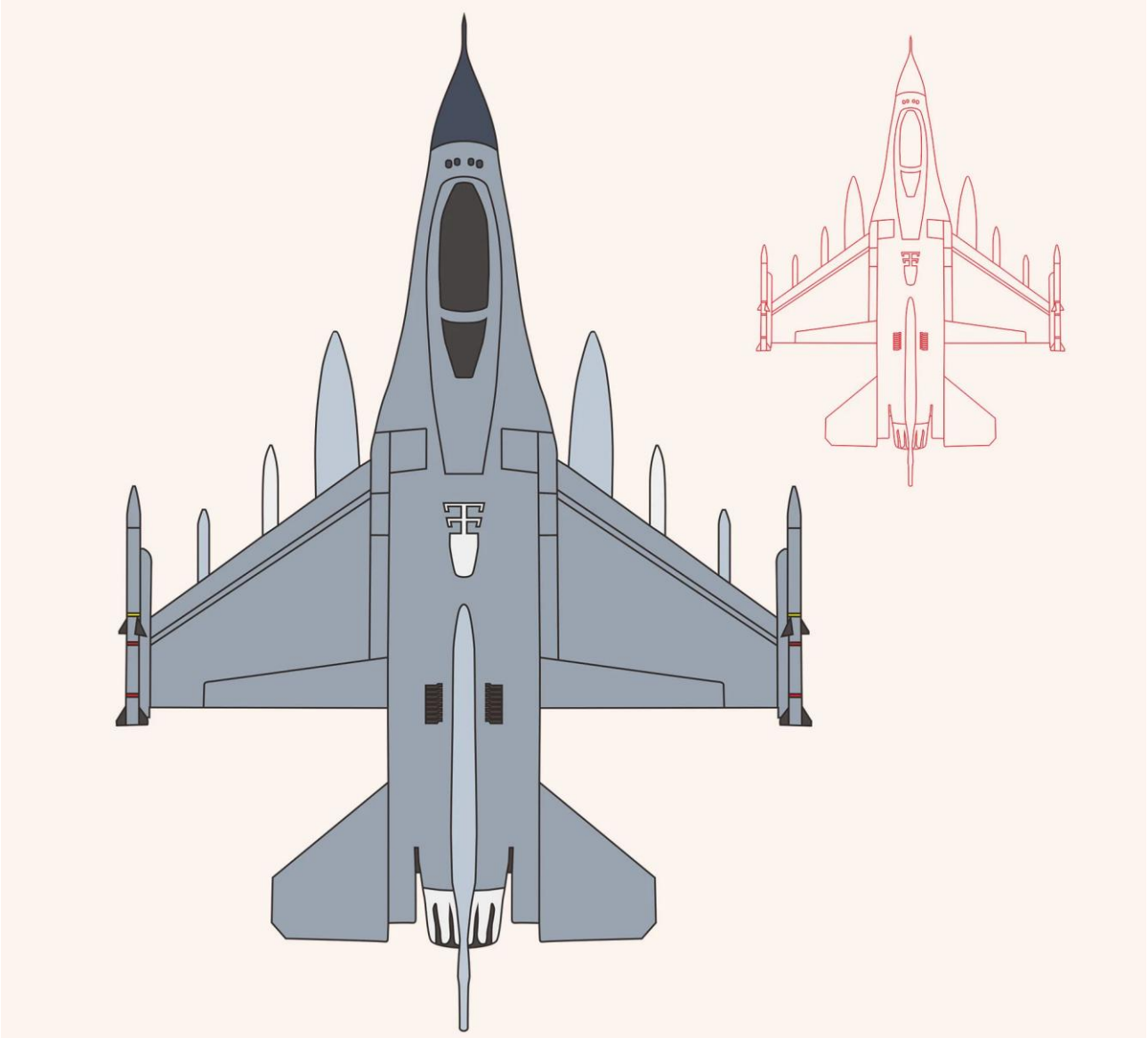


Рисунок 3.2.4 – Модель літака Ф-16

Останнім кроком пошуку та аналізу моделей літаків є модель літака Ф-16. (Рисунок 3.2.4)

Перед розробкою гри, було створено блок-схему роботи гри. (Рисунок 3.2.5)

Рисунок 3.2.5 – Блок-схема

На блок-схемі видно принцип роботи програми. Після запуску гри гравець потрапляю в меню, після чого обирає рівень та починає виконувати завдання рівня. Якщо завдання виконано невдало, то гравець відправляється до проходження рівня знову. Якщо завдання виконано вдало, гравець отримує результат рівня та має можливість або завершити гру, або обрати наступний рівень.

3.3 Алгоритм роботи ігрового додатку

У ігровому додатку процес починається з ініціалізації. Завантажуються всі необхідні ресурси, включаючи графіку літака, фонові зображення та звуки. Створюється карта, яка окреслює межі гри та демонструє декорації. Далі ініціалізуються основні об'єкти: літак гравця, вороги та бонуси. У цей момент ми визначаємо правила, наприклад, скільки життів у гравця, його початкові точки та наскільки важкою буде гра.

Гра працює через базовий цикл. У цьому циклі обробляється вхід від гравця. Режим введення може бути через сенсорний екран, джойстик або гіроскоп. Це визначає рух літака гравця. Літак стріляє постійно, незалежно від дій гравця, допоки гра не зупиниться.

Стан гри постійно оновлюється. Вороги та перешкоди рухаються відповідно до сценарію рівня. Відбувається перевірка зіткнень між об’єктами, такими як літак гравця та вороги, або між снарядами та ворогами. Нові вороги та бонуси з’являються випадковим чином, а залежно від рівня складність гри зростає.

Результати гри розраховуються в режимі реального часу. Якщо літак гравця зіштовхується з ворогами чи перешкодами, кількість його життів зменшується. За знищення ворогів або збір бонусів гравець отримує монети. Гра завершується, якщо у гравця закінчуються життя або досягається задана мета, а саме знищення всіх ворогів на рівні. Рухи літака гравця, суперників, снарядів і спеціальних здібностей показуються на екрані в кожному кадрі відео. Дисплей показує подробиці про рахунок, кількість життів і етап. У грі також використовуються такі звуки, як постріли, бум-бум і музика, щоб переконатися, що грати в гру цікаво.

3.4 Стиль графіки

Стиль графіки для ігрового додатку обирався серед основних:

- Піксельна графіка.

- 2D-графіка.
- Векторна графіка.
- Ретро-футуризм.
- Готика або темний фентезі-стиль.

Піксельна графіка — форма цифрового зображення, створеного на комп'ютері за допомогою растрового графічного редактора, де зображення редагується на рівні пікселів (точок), а роздільна здатність зображення настільки мала, що окремі пікселі чітко видно. [6]

Переваги:

- Один з найбільш простих у вивченні стилів комп'ютерного мистецтва.
- Природний вибір на обмежених палітрах і наднизьких розширеннях, де важливий кожен піксель.
- Потребує мало пам'яті за рахунок застосування палітрових форматів з невеликою кількістю кольорів.
- Навіть при дуже поганій передачі кольору піксельний малюнок не втрачає виразности.
- Добре виглядає на екранах із чіткими межами пікселів.

Недоліки:

- В епоху hicolor-моніторів і відеопроцесорів з апаратним альфа-змішуванням виразніше виглядають інші стилі.
- Погано переносить автоматичне масштабування (при зміні розширення картинку потрібно перемальовувати).
- На неякісних моніторах може мерехтіти.

Комп'ютерна 2D-графіка — головним чином складається з двовимірних моделей (таких як геометричні 2D-моделі, текст і цифрові зображення) і методів, визначених для них. Комп'ютерна 2D-графіка використовується, головним чином, в програмах, які були розроблені на базі традиційних

технологій друку і малювання, таких як книгодрукування, картографія, креслення, реклама, тощо. У цих додатках двовимірне зображення — не просто представлення реального об'єкта, але і незалежний культурний експонат з власним семантичним значенням; тому двовимірні моделі є кращими, так як вони дають більш безпосередній контроль над зображенням, ніж комп'ютерна 3D-графіка. 2D-графічні моделі можуть об'єднувати геометричні моделі (векторна графіка), цифрові зображення (растрова графіка), набраний текст (з певним змістом, стилем шрифту і розміром, кольором, положенням і орієнтацією), математичні функції, рівняння і інше. Ці компоненти можуть змінюватися і управлятися двовимірними геометричними перетвореннями, такими як паралельне перенесення, обертання, масштабування. У об'єктно-орієнтованій графіці зображення побічно описано об'єктом, наділеним методом самостійного рендеринга — процедура, яка призначає колір на пікселі зображення довільним алгоритмом. Складні моделі можуть бути побудовані об'єднанням більш простих об'єктів в парадигмах об'єктно-орієнтованого програмування. [7]

Ве́кторна гра́фіка (також геометричне моделювання або об'єктно-орієнтована графіка) — створення зображення в комп'ютерній графіці з сукупності геометричних примітивів — точок, ліній, кривих, полігонів, тобто об'єктів, які можна описати математичними виразами. [8]

Переваги:

- Розмір файла, який займає описова частина, не залежить від реальної величини об'єкта, що дозволяє, використовуючи мінімальну кількість інформації, описати достатньо великий об'єкт файлом мінімального розміру.
- У зв'язку з тим, що інформація про об'єкт зберігається в описовій формі, можна нескінченно збільшити графічний примітив, наприклад, дугу кола, і вона залишиться гладкою. З іншого боку,

якщо крива представлена у вигляді ламаної лінії, збільшення покаже, що крива не є гладкою.

- Параметри об'єктів зберігаються і можуть бути легко змінені. Також це означає, що переміщення, масштабування, обертання та інше, не погіршує якості малюнка. Більш того, зазвичай вказують розміри в апаратно-незалежних одиницях, які ведуть до найкращої растеризації на растрових приладах.
- При збільшенні або зменшенні об'єктів товщина ліній може бути задана постійною величиною, незалежно від реального контуру.

Недоліки:

- Не кожен об'єкт може бути легко зображений у векторному вигляді — для того, щоб зображення було подібним до оригіналу може знадобитися дуже велика кількість об'єктів з високою складністю, що негативно впливає на кількість пам'яті, яку займатиме зображення та час для його відтворення.

Ретро футуризм показує погляд на майбутнє через призму минулого, відображаючи усе в сучасному науково-фантастичному вигляді. Ця тенденція була популярна в 2019 році й залишиться актуальною ще в 2020 році. Напрямок ретро футуризму дозволив створити нові жанри і надати дизайну відчуття ностальгії. Такий дизайн подає зовсім інший вид візуальної естетики, яку шукають сучасні покоління. [9]

Те́мне фе́нтезі — піджанр художніх творів який включає в себе елементи жахів та готики, і дія якої відбувається в антуражі традиційного фентезі. Традиційне темне фентезі побудовано за класичними фентезійним канонами, але на відміну від класичного фентезі, де основу сюжету складають ідея боротьби Добра зі Злом, в темному фентезі Зло вже перемогло, і його прояви сприймаються як належне. Події темного фентезі відбуваються в суворому, жорстокому і похмурому світі, де герої, переважно, як мінімум, неоднозначні з позиції моралі і ведуть боротьбу

лише за те, щоб світ остаточно не перетворився на пекло. Основною ідеєю темного фентезі можна назвати боротьбу «меншого» Зла з «великим», абсолютним Злом. Автори книг у цьому жанрі буквально шокують читача виписаною з особливим старанням атмосферою гнітючого мороку та повної безнадії. При цьому, на думку Джона Клюта, до жанру не відносяться відверто фантасмагоричні твори про вигадану «потойбічну нечисту силу» як-то упирі, перевертні, окультна маячня тощо. [10]

З урахуванням усієї вище перерахованої інформації, вибір у стилі графіків зупинився на піксельній графіці.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

3.1 Розробка графічних елементів

Для створення графічних елементів гри було використано програму Photoshop. Стиль усіх елементів – піксельна графіка.

Процес створення літака починається з базового корпусу. Спочатку формується центральна частина – фюзеляж, зображений як овальна фігура з червоними та фіолетовими акцентами. Потім додаються крила, які розширюють основну форму. Їх розміщують по боках, використовуючи окремі шари, щоб забезпечити зручність редагування. (Рисунок 3.1.1)

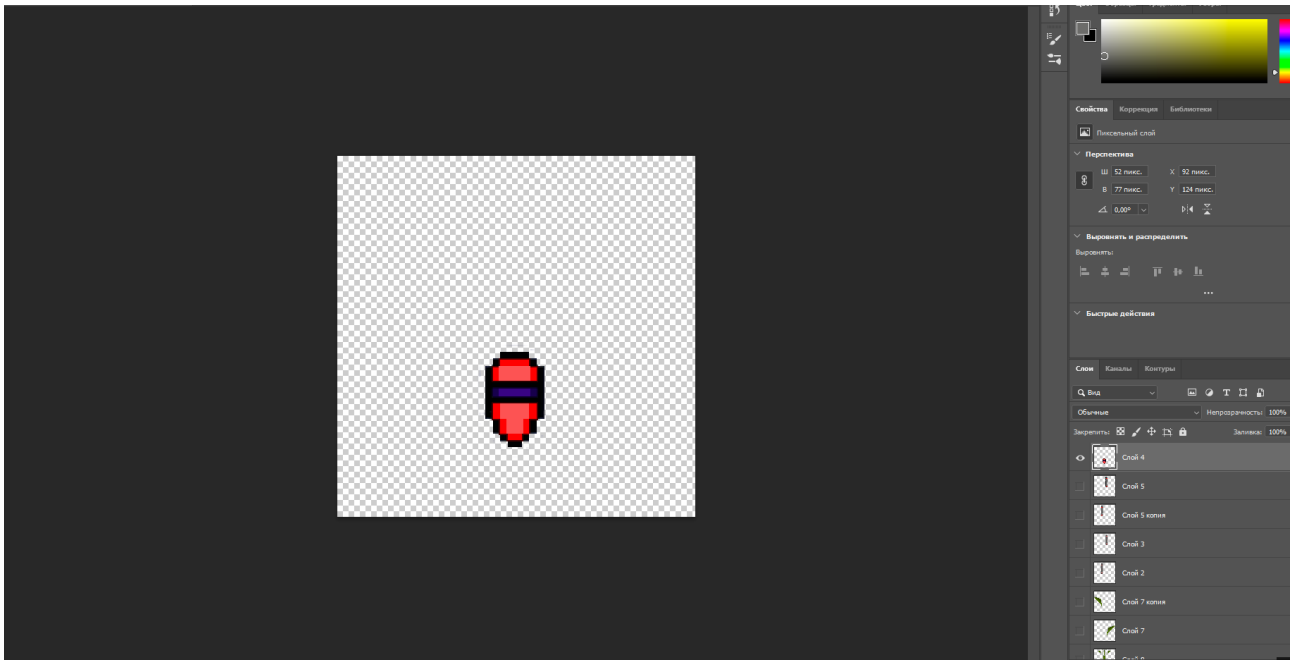


Рисунок 3.1.1

Після цього додаються двигуни чи інші елементи – сірі вертикальні частини зверху літака. Вони розташовані симетрично, і для їхнього створення використовуються прості прямокутні форми. Усе це накладається поверх базового корпусу. (Рисунок 3.1.2)

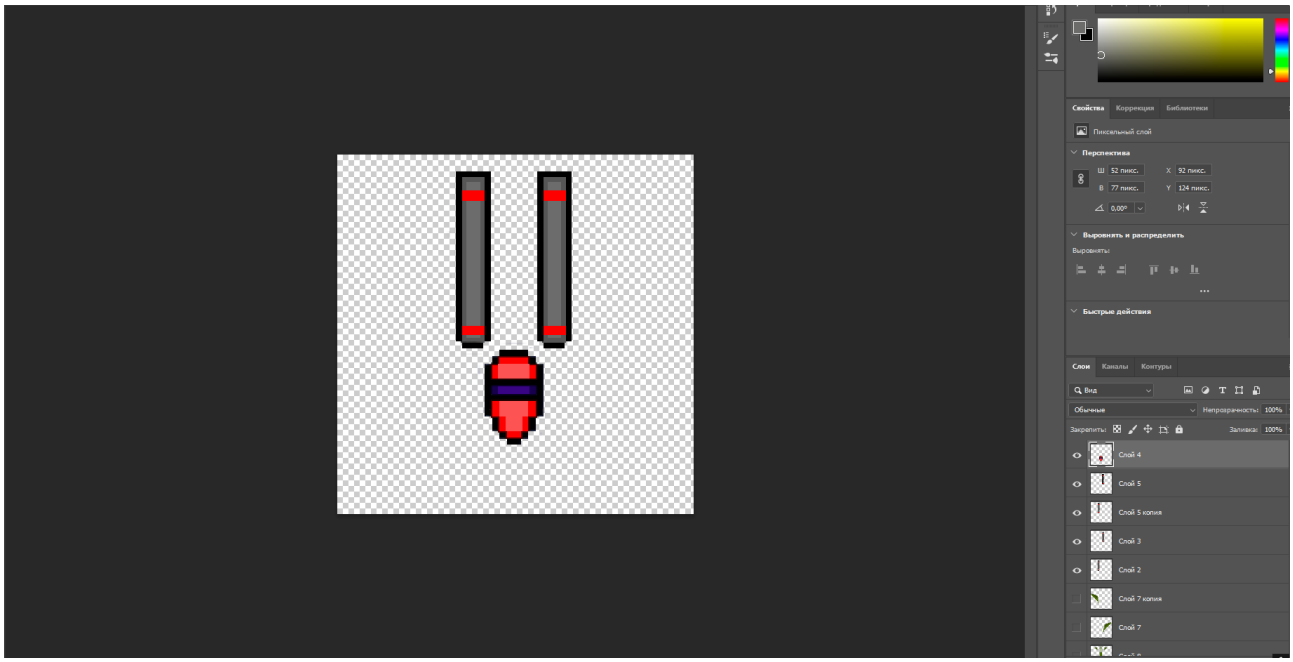


Рисунок 3.1.2

На завершальному етапі додаються дрібні деталі, такі як контури, смуги чи інші декоративні елементи, щоб підкреслити форму літака. Для збереження прозорості фон лишається порожнім, що дозволяє легко інтегрувати літак у гру або інший проєкт. Уся робота виконується на різних шарах, що дозволяє гнучко змінювати вигляд літака на будь-якому етапі. (Рисунок 3.1.3)

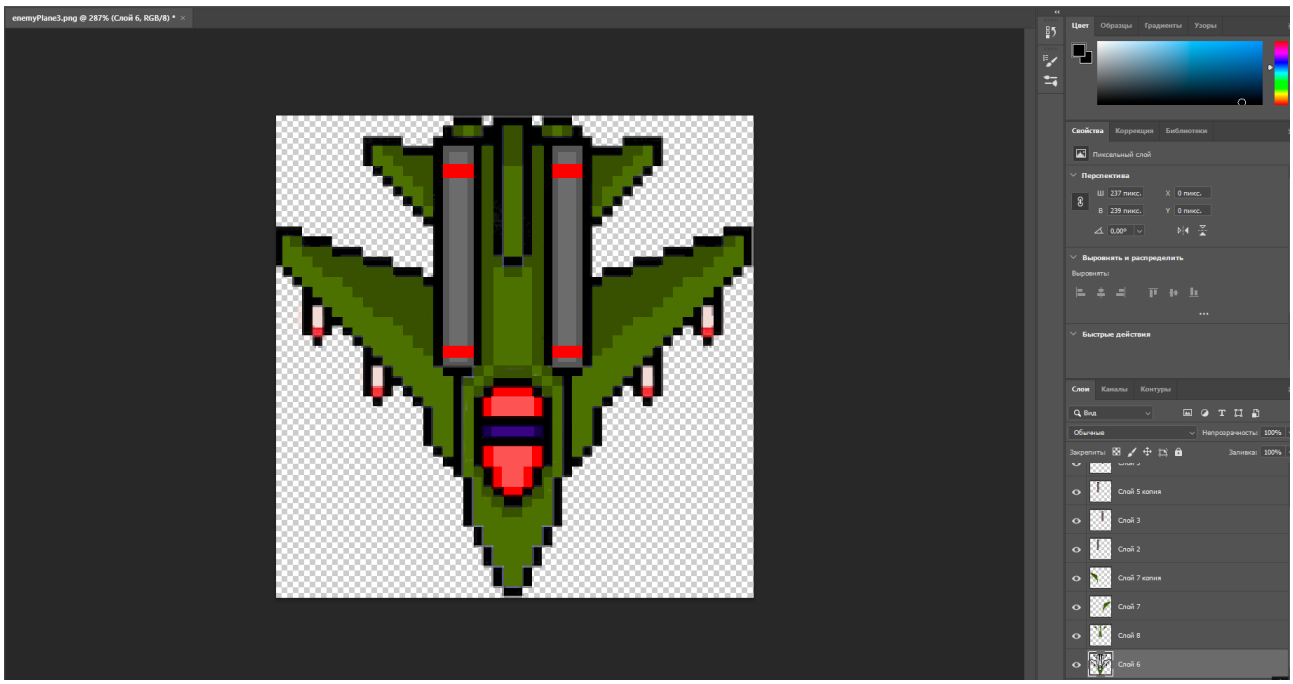


Рисунок 3.1.3

3.2 Програмна реалізація ігрового додатку

Програмна реалізація ігрового додатку розпочинається із налаштування середовища програмування, а саме налаштування проекту в Unity. (Рисунок 3.2.1)

Наступним кроком в реалізації є додавання графічних об’єктів розробленних в Photoshop. Усі елементи реалізовані у стилі піксельних картинок. Із розробленних графічних об’єктів є: літак гравця, три види літаків противника, набої гравця, набої противника, ефект влучання, ефект вибуху літака.(Рисунок 3.2.1 та Рисунок 3.2.1.1)

Рисунок 3.2.1 – Моделі літаков

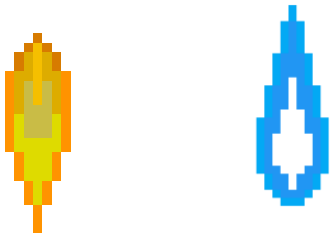


Рисунок 3.2.1.1 – Моделі набоїв

Наступний крок після додавання графічних об’єктів є створення заднього фону та можливість руху його під час руху літака. Було реалізовано за допомогою строки коду “meshRender.material.mainTextureOffset += new Vector2(0, speed * Time.deltaTime);”. meshRender — це посилання на компонент MeshRenderer об'єкта в Unity, який відповідає за відображення 3D-моделі

за допомогою матеріалу. Матеріал цього об'єкта можна отримати через властивість material, і ця властивість дає доступ до текстур і шейдерів, які використовуються для рендерингу. В даному випадку, з допомогою mainTextureOffset змінюється зміщення основної текстури на об'єкті. Це значення задається через тип Vector2, який містить два числа — зміщення по осі X та Y. В mainTextureOffset по суті визначається, як саме текстура накладається на поверхню об'єкта. Коли додається нове значення до mainTextureOffset, воно змінює позицію текстури. Новий Vector2(0, speed * Time.deltaTime) задає зміщення текстури. Перше значення — це 0, що означає, що текстура не рухатиметься по осі X, а лише по осі Y. Друге значення — це speed * Time.deltaTime. Тут speed визначає швидкість руху текстури по вертикалі, а Time.deltaTime відповідає за час між кадрами, що дозволяє робити зміщення незалежним від частоти кадрів. Time.deltaTime гарантує, що текстура буде рухатися з постійною швидкістю, незалежно від того, скільки кадрів на секунду рендериться в грі. Загалом, цей код змінює позицію текстури по осі Y, створюючи ефект її руху, і робить цей рух плавним, незалежно від частоти кадрів.

Наступним кроком є додавання літака гравця та надання можливості керувати літаком. Літак додається новим об’єктом за допомогою перетягування моделі до списку об’єктів. Далі створюється скрипт PlayerScript.cs, в якому створюється можливість керування. (Рисунок 3.2.2)

```
float deltaX = Input.GetAxis("Horizontal") * Time.deltaTime * speed;
float deltaY = Input.GetAxis("Vertical") * Time.deltaTime * speed;

float newXpos = Mathf.Clamp(transform.position.x + deltaX,minX,maxX);
float newYpos = Mathf.Clamp(transform.position.y + deltaY,minY,maxY);

transform.position = new Vector2(newXpos, newYpos);
```

Рисунок 3.2.2 – Реалізація керування літаком

У першому рядку оголошується змінна deltaX, яка буде містити зміну позиції об'єкта по горизонтальній осі X. Метод Input.GetAxis("Horizontal") отримує вхід користувача по горизонтальній осі (за замовчуванням це клавіші A/D або стрілки вліво/вправо на клавіатурі). Це значення змінюється від -1 до 1, де -1 — рух вліво, а 1 — вправо. Множення на Time.deltaTime робить так, щоб зміщення було незалежним від частоти кадрів, забезпечуючи однакову швидкість переміщення на всіх пристроях. Далі множимо на змінну speed, яка визначає швидкість переміщення об'єкта. Результат обчислення цього виразу зберігається в змінній deltaX.

У другому рядку оголошується змінна deltaY, яка буде містити зміщення об'єкта по вертикальній осі Y. Метод Input.GetAxis("Vertical") отримує вхід користувача по вертикальній осі (за замовчуванням це клавіші W/S або стрілки вгору/вниз на клавіатурі), що також повертає значення від -1 до 1. Множення на Time.deltaTime дозволяє забезпечити плавне переміщення незалежно від частоти кадрів. Після цього множиться на змінну speed, щоб контролювати швидкість руху. Результат цього виразу зберігається в змінній deltaY.

У третьому рядку оголошується змінна newXpos, яка міститиме нову позицію об'єкта по осі X після переміщення. transform.position.x отримує поточну координату об'єкта по осі X, до якої додається зміщення deltaX, обчислене в попередньому рядку. Однак ми використовуємо функцію Mathf.Clamp, щоб обмежити нову позицію в межах заданого діапазону — від minX до maxX. Це означає, що якщо нове значення координати X виходить за ці межі, воно буде "прищеплене" до найближчої допустимої межі.

У четвертому рядку оголошується змінна newXpos, яка міститиме нову позицію об'єкта по осі X після переміщення. transform.position.x отримує поточну координату об'єкта по осі X, до якої додається зміщення

deltaX, обчислене в попередньому рядку. Однак ми використовуємо функцію Mathf.Clamp, щоб обмежити нову позицію в межах заданого діапазону — від minX до maxX. Це означає, що якщо нове значення координати X виходить за ці межі, воно буде "прищеплене" до найближчої допустимої межі.

В останньому рядку Ми присвоюємо нове значення позиції об'єкта. Створюємо новий об'єкт типу Vector2, який містить обмежені значення координат по осях X та Y — це newXpos і newYpos. Цей новий Vector2 призначається властивості transform.position, що відповідає за позицію об'єкта в 2D-просторі. Завдяки цьому об'єкт переміщається до нової, обмеженої позиції, враховуючи введення користувача і обмеження на координати. Таким чином, цей код дозволяє переміщати об'єкт за допомогою введення користувача (стрілок або клавіш WASD) з обмеженням на координати, щоб об'єкт не виходив за задані межі.

Наступнип кроком в розробці ігрового додатку є створення можливості стріляти для гравця. Для реалізіції цього елементу було додано модель набоїв та створено новий скрипт Shooting.cs.(Рисунок 3.2.3) Також надано фізику польоту для набоїв за допомогою скрипту Bullet.cs. (Рисунок 3.2.4)

```
void Update()
{

}

void Fire(){
    Instantiate(playerBullet, spawnPoint1.position, Quaternion.identity);
    Instantiate(playerBullet, spawnPoint2.position, Quaternion.identity);
}

IEnumerator Shoot(){
    while(true){
        yield return new WaitForSeconds(bulletSpawnTime);
        Fire();
    }
}
```

Рисунок 3.2.3 – Реалізація можливсті стріляти

У цьому коді реалізовано стрільбу для гравця, використовуючи два методи — Fire() і Shoot().

Метод Fire() призначений для створення куль. Спочатку оголошується метод Fire(), який не приймає параметрів і не повертає значень. У ньому викликаються два методи Instantiate(). Перший викликається для створення кулі в точці спавну spawnPoint1, передаючи позицію spawnPoint1.position та обертання Quaternion.identity, що означає відсутність обертання. Це створює першу кулю. Далі аналогічний виклик Instantiate() відбувається для точки спавну spawnPoint2, де створюється друга куля, також без обертання. Таким чином, метод Fire() створює дві кулі в двох точках спавну одночасно.

Метод Shoot() є корутиною, тобто методом, який виконується по черзі через кілька кадрів, а не все за один раз. У ньому створюється нескінченний цикл while(true), який постійно викликає метод Fire(), забезпечуючи регулярне створення куль. Кожен цикл чекає певний час між пострілами завдяки yield return new WaitForSeconds(bulletSpawnTime). В bulletSpawnTime зберігається час затримки між пострілами, який визначає, як часто будуть створюватися нові кулі. Після кожної паузи викликається метод Fire(), що спричиняє появу двох нових куль. Цей цикл продовжується нескінченно, поки не буде зупинено корутину.

У результаті, метод Shoot() дозволяє постійно стріляти через задані інтервали часу, створюючи кулі з двох точок спавну, поки корутина працює.

```
// Update is called once per frame
void Update()
{
    transform.Translate(Vector2.up * speed * Time.deltaTime);
}
```

Рисунок 3.2.4 – Реалізація польоту набоя

На рисунку 3.2.4 метод Translate використовується для переміщення об'єкта в Unity. Спочатку, transform — це компонент Transform**,** який зберігає позицію об'єкта в просторі. Через цей компонент змінюється позиція об'єкта. Метод Translate() змінює позицію об'єкта на певну відстань, яку визначає вектор, переданий як параметр.

Vector2.up — це вектор, який вказує в напрямку вверх по осі Y в 2D-просторі, і має значення (0, 1). Коли цей вектор множиться на speed, результат — це зміщення, яке пропорційне швидкості. speed визначає, як швидко буде переміщуватися об'єкт у цьому напрямку. Тобто, чим більше значення speed, тим швидше об'єкт рухатиметься вгору.

Time.deltaTime множить це зміщення, щоб забезпечити рух, який не залежить від частоти кадрів. Time.deltaTime — це час, що минув з моменту останнього кадру, і це гарантує, що рух об'єкта буде однаковим на різних пристроях, навіть якщо частота кадрів варіюється.

В результаті, цей код переміщає об'єкт вгору з швидкістю, визначеною змінною speed, при цьому рух є плавним і незалежним від FPS.

Наступним кроком є створення ворожих літаків. Для реалізації цього кроку було додано модель ворожих літаків та створено скрипт EnemyScript.cs. (Рисунок 3.2.5)

```
void Update()
{
    transform.Translate(Vector2.down * speed * Time.deltaTime);
}

private void OnTriggerEnter2D(Collider2D collision){
    if(collision.tag=="PlayerBullet"){
        Destroy(gameObject);
    }


}

void EnemyFire(){

    for(int i = 0; i < gunPoint.Length; i++){
        Instantiate(enemyBullet, gunPoint[i].position, Quaternion.identity);
    }
}
IEnumerator EnemyShooting(){
    while(true){
        yield return new WaitForSeconds(enemyBulletSpawnTime);
        EnemyFire();
    }

}
```

Рисунок 3.2.5 – Реалізація скрипту ворожих літаків

У цьому коді реалізовано взаємодію ворога з кулею гравця, а також механізм стрільби ворога.

Метод OnTriggerEnter2D(Collider2D collision) викликається автоматично, коли об'єкт, до якого прикріплений цей скрипт, вступає в колізію з іншим об'єктом. Параметр collision містить інформацію про об'єкт, з яким відбулася колізія. У цьому методі перевіряється, чи тег об'єкта, з яким сталося зіткнення, є "PlayerBullet". Якщо це так, то викликається метод Destroy(gameObject), який знищує об'єкт ворога, тобто, коли ворог потрапляє в кулю гравця, він знищується.

Метод EnemyFire() відповідає за стрільбу ворога. Він проходить по масиву точок стрільби gunPoint за допомогою циклу for. Для кожної точки стрільби викликається метод Instantiate(), який створює префаб кулі ворога (enemyBullet) на позиції відповідної точки стрільби. Позиція визначається через gunPoint[i].position, а обертання встановлюється в Quaternion.identity, тобто без обертання.

Метод EnemyShooting() — це корутина, яка виконується безперервно. Вона містить нескінченний цикл while(true), що дозволяє ворогу постійно стріляти. У циклі кожен раз, після затримки, яка задається через WaitForSeconds(enemyBulletSpawnTime), викликається метод EnemyFire(), що і генерує кулі. Це дозволяє ворогу постійно стріляти з певним інтервалом, поки корутина працює.

Таким чином, коли ворог вступає в зіткнення з кулею гравця, він знищується. Паралельно, якщо корутина EnemyShooting() запущена, ворог буде безперервно стріляти з кількох точок, створюючи кулі з інтервалом, визначеним у enemyBulletSpawnTime.

Для реалізації випадкового появлення ворожих літаків було створено скрипт Spawner.cs. (Рисунок 3.2.6)

```
IEnumerator EnemySpawner(){
    while (true){
    yield return new WaitForSeconds(respawnTime);
    SpawnEnemy();
    }
}

void SpawnEnemy(){
    int randomValue = Random.Range(0,enemy.Length);
    int randomXPos = Random.Range(-2, 2);
    Instantiate(enemy[randomValue], new Vector2(randomXPos, transform.position.y), Quaternion.identity);
}
```

Рисунок 3.2.6 – Реалізація випадкового появлення літаків

Цей код містить дві основні частини: корутину для постійного спавну ворогів і метод для фактичного створення ворогів у випадкових місцях. Розглянемо кожен елемент коду детально.

Метод EnemySpawner() є корутиною, яка запускає нескінченний цикл, що відповідає за створення ворогів через певні інтервали часу. Це дозволяє ворогам з'являтися через задані проміжки часу без блокування виконання інших частин гри. У циклі використовується yield return new WaitForSeconds(respawnTime), щоб зробити паузу між кожним спавном. Кількість часу між спавнами визначається значенням змінної respawnTime,

яка може бути визначена в іншому місці коду або в редакторі Unity. Після кожної затримки викликається метод SpawnEnemy(), який відповідає за фактичний спавн ворога.

Метод SpawnEnemy() відповідає за створення ворогів. Перш за все, він генерує випадкове значення для вибору ворога з масиву enemy. Це відбувається за допомогою Random.Range(0, enemy.Length), що дає випадковий індекс для вибору одного з ворогів, які зберігаються в масиві enemy.

Наступним кроком є вибір випадкової позиції по осі X для спавну ворога. За допомогою Random.Range(-2, 2) генерується випадкове значення в діапазоні від -2 до 2. Це визначає, де саме на осі X з'явиться ворог (в межах цього діапазону, тобто на невеликій відстані від центру екрану).

Після цього використовується Instantiate(), щоб створити ворога в новій позиції. Позиція визначається як new Vector2(randomXPos, transform.position.y), де randomXPos — випадкова координата по осі X, а transform.position.y — це поточна координата по осі Y, тобто ворог з'явиться на тій самій висоті, де знаходиться об'єкт, до якого прикріплений цей скрипт. Встановлено обертання Quaternion.identity, що означає відсутність обертання у створеного об'єкта (ворога).

Таким чином, цей код дозволяє створювати ворогів випадковим чином через визначений інтервал часу, вибираючи випадкові позиції на осі X в межах від -2 до 2 та на тій самій висоті, де знаходиться об'єкт, до якого прикріплений цей скрипт.

Також було додатково створено скрипт Shredder.cs. Основна його задача полягала у знищенні об’єктів що покинули зону мапи, включно із літаками та набоями.

3.3 Опис процесу гри

На Рисунку 3.3.1 представлено екран початку гри з назвою "AIR ATTACK". У центрі знаходяться кнопки "START" (почати гру) та "EXIT" (вийти).

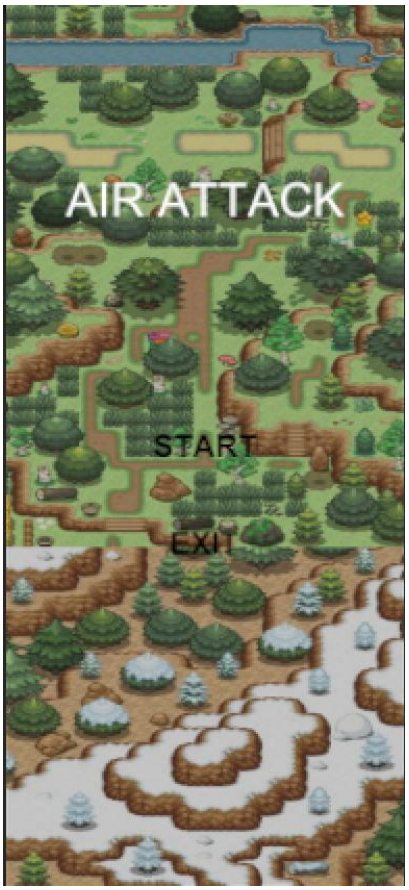


Рисунок 3.3.1 – Початковий екран

На Рисунку 3.3.2 гра в активному процесі. Видно кілька літаків: один керується гравцем, а інші є супротивниками. На екрані розташовані золоті монети, які можна збирати. У нижній частині видно шкалу, що може позначати рівень здоров'я та кількість зібраних монет.

Рисунок 3.3.2 – Активний процес гри

На Рисунку 3.3.3 гра перебуває в режимі паузи, про що свідчить великий напис "PAUSE". Під ним розташовані кнопки "RESUME" (продовжити гру) та "EXIT" (вийти з гри).

Рисунок 3.3.3 – Меню паузи

На Рисунку 3.3.4 напис "Game Over" у верхній частині екрану, що свідчить про закінчення гри. Під написом "Game Over" є кнопки з текстом "Try Again" (спробувати знову) і "Exit" (вийти), які дозволяють гравцеві вирішити, чи почати нову гру, чи завершити. Гравець програв у грі, і тепер йому пропонується вибір подальших дій.

Рисунок 3.3.4 – Меню програшу

ВИСНОВКИ

У ході виконання магістерської роботи на тему "Мобільна гра Air Attack" було спроектовано ігровий додаток для мобільних пристроїв, який відповідає сучасним вимогам індустрії мобільних ігор. У процесі роботи були виконані наступні етапи:

- Аналіз ідеї та концепції гри: Було проведено детальний аналіз ринку мобільних ігор, визначено популярні жанри та механіки, що дозволило створити концепцію гри.
- Ознайомлено з мовою програмування C# та середовище Unity для створення ігрової логіки, а також інтегровано графіку та звук для покращення ігрового досвіду.

Результатом роботи є спроектована та розроблена мобільна гра «Air Attack», яка буде володіти захоплюючим геймплеєм та привабливим дизайном. Проект дозволив поглибити знання в області розробки ігор, навички програмування та роботи з графічними та звуковими ресурсами.

У майбутньому планується розробити нові рівні, покращити інтерфейс та додати нових противників, а також розширити аудиторію гравців шляхом активного маркетингу та реклами.

СПИСОК ЛІТЕРАТУРИ

1. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. К.: Держстандарт, 2015. 37 с.
2. Мобільні додатки [Електронний ресурс]. – Режим доступу: URL: http://surl.li/ublzh – Назва з екрану.
3. Unity [Електронний ресурс]. – Режим доступу: URL: https://unity.com/ – Назва з екрану.
4. Unity Hub [Електронний ресурс]. – Режим доступу: URL: https://unity.com/unity-hub – Назва з екрану.
5. Бабій М.С. Теорія програмування: Навчальний посібник [Електронний ресурс] / М.С. Бабій, О.П. Чекалов.– Суми: Вид-во СумДУ, 2009. – 181 с.
6. Піксельна графіка [Електронний ресурс]. – Режим доступу: URL: http://shorturl.at/W0tjJ – Назва з екрану.
7. Ком’ютерна 2D графіка [Електронний ресурс]. – Режим доступу: URL: http://shorturl.at/aw5lG – Назва з екрану.
8. Векторна графіка [Електронний ресурс]. – Режим доступу: URL: http://shorturl.at/zV05O – Назва з екрану.
9. Ретро футуризм [Електронний ресурс]. – Режим доступу: URL: http://shorturl.at/F3w1X – Назва з екрану.
10. Темне фентезі [Електронний ресурс]. – Режим доступу: URL: http://shorturl.at/38fKQ – Назва з екрану.
11. Нікітченко М.С. Теоретичні основи програмування: Навчальний посібник [Електронний ресурс] / М.С. Нікітченко. – Київ: КНУ ім. Т.Г. Шевченка, 2009. – 200 с. – Режим доступу: http://ttp.unicyb.kiev.ua/doc/TOP.pdf.

12. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
13. ДСТУ 8302:2015. БІБЛІОГРАФІЧНЕ ПОСИЛАННЯ. Загальні положення та правила складання / Нац. стандарт України. Вид. офіц. [Уведено вперше; чинний від 01.07.2016]. Київ : ДП «УкрНДНЦ», 2016. 17 с. (Інформація та документація). – З внесеними поправками.
14. ДСТУ ГОСТ 7.1:2006 Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання / Нац. стандарт України. Вид. офіц. – [Чинний від 2007-07-01]. Київ : Держспоживстандарт України, 2007. 47 с.
15. Стандарт вищої освіти зі спеціальності 122 «Комп’ютерні науки» для другого (магістерського) рівня вищої освіти (наказ МОН України № 393 від 28.04.2022 р.).
16. Освітньо-професійна програма «Комп’ютерні науки». Ступінь − магістр / Полтавський університет економіки та торгівлі. 2022. 17 с.
17. Положення про кваліфікаційну роботу (проєкт) (ДПСЯ М 9-8.5.-19-05-22), Полтавський університет економіки та торгівлі. 2022. 30 с

ДОДАТОК А. ДИСК З МАТЕРІАЛАМИ