МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ
на здобуття освітнього ступеня бакалавра

на тему: «**Розробка ігрового застосунку в жанрі "три в ряд" під мобільну систему Android на рушію Unity»**

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

Керівник: *викладач-стажист*
Гаврильчик Леонід Сергійович

Рецензент: *викладач-стажист*
Місай Володимир Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

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

Протокол № ____ від « ___ » _____ 2022 р.

Острог, 2022

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*

Кафедра: *економіко-математичного моделювання та інформаційних технологій*

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА

«___» _____20__ р.

З А В Д А Н Н Я

на кваліфікаційну роботу/проєкт студента

Ковбасюка Олександра Михайловича

1. Тема роботи Розробка ігрового застосунку в жанрі "три в ряд" під мобільну систему Android на рушію Unity
керівник роботи/проєкту Гаврильчик Леонід Сергійович, викладач-стажист

Затверджено наказом ректора НаУОА від 29 жовтня 2021 року № 110.

2. Термін здачі студентом закінченої роботи/проєкту: 03 червня 2022 року.

3. Вихідні дані до роботи/проєкту: постановка задачі, аналіз аналогів, вихідні дані.
4. Перелік завдань, які належить виконати: опис предметного середовища; огляд наявних аналогів; постановка задачі; опис архітектури рішення, що розробляється; опис коду та інтерфейсу додатку; тестування.
5. Перелік графічного матеріалу: діаграма прецедентів функціональних вимог, діаграма станів ігрового циклу, діаграма станів системи меню.

6. Консультанти розділів роботи/проєкту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Гаврильчик Л.С.	01.12.2021р.	01.12.2021р.
2	Гаврильчик Л.С.	01.12.2021р.	01.12.2021р.
3	Гаврильчик Л.С.	01.12.2021р.	01.12.2021р.

7. Дата видачі завдання: 01.12.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Затвердження теми роботи/проєкту	до 01.11.2021 р.	
2	Постановка завдання	до 01.12. 2021 р.	
3	Розробка архітектури та загальної структури системи	до 01.02.2022 р.	
4	Розробка структур окремих підсистем	до 01.03. 2022 р.	
5	Програмна реалізація системи	до 01.05.2022 р.	
6	Попередній захист кваліфікаційної роботи/проєкту	до 01.06.2022р.	

Студент: _____ Олександр КОВБАСЮК

Керівник кваліфікаційної роботи/проєкту: _____ Леонід ГАВРИЛЬЧИК

АНОТАЦІЯ
кваліфікаційної роботи/проєкту
на здобуття освітнього ступеня бакалавра

Тема: Розробка ігрового застосунку в жанрі "три в ряд" під мобільну систему Android на рушію Unity

Автор: Ковбасюк Олександр Михайлович

Науковий керівник: Гаврильчик Леонід Сергійович

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *56 с., 20 рис., 3 табл., 3 додатки, 14 джерел.*

Ключові слова: ігровий застосунок, Unity, 2D

Короткий зміст праці:
Кваліфікаційна робота/проєкт на тему: «Розробка ігрового застосунку в жанрі "три в ряд" під мобільну систему Android на рушію Unity» присвячена розробці 2D гри з можливістю користування на платформі Android. Актуальність обраної для кваліфікаційної роботи/проєкту теми пов’язана з великою популярністю мобільних ігрових додатків даного жанру серед Android користувачів. У процесі роботи використано такі сучасні засоби як: ігровий рушій Unity, мова програмування C#, інтегроване середовище розробки Microsoft Visual Studio, графічний редактор Adobe Photoshop.

Qualification thesis on the topic: "Development of a game application in the genre of "Match 3" for the Android mobile system on the Unity engine" is devoted to the development of 2D game with the possibility of use on the Android platform. The relevance of the topic selected for qualification work is associated with the great popularity of mobile game applications of this genre among users. In the process of work such modern tools as: game engine Unity, C# programming language, integrated development environment Microsoft Visual Studio, graphic editor Adobe Photoshop were used.

_____

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Debug – відлагоджувач, комп'ютерна програма, яка використовується для тестування і виправлення вад інших програм.

IDE – інтегроване середовище розробки — комплексне програмне рішення для розробки програмного забезпечення.

Ассет – це компоненти, які є графікою, звуковим супроводом або скриптами. Вони прикріплюються до об'єктів і становлять важливу частину гри.

ОС – операційна система.

ПЗ – програмне забезпечення.

ПП – програмний продукт.

ШІ – штучний інтелект.

Растрове зображення – це зображення, що являє собою набір пікселів, кожний із яких має певний колір.

Скрипт – це невелика програма, яка містить послідовність дій, створених для автоматичного виконання завдання.

ВСТУП

За останні кілька років індустрія мобільних ігор перебуває на піку своєї популярності та прибутковості – а верхівки поки що не видно. Стрімкий розвиток смартфонів, які за потужністю вже наступають на п'яти ноутбуків, призвело до збільшення кількості та асортименту мобільних ігор.

Справжнім зародженням мобільного геймінгу можна вважати іграшку Snake, яка вийшла в 1997 для телефонів Nokia. Улюблена мільйонами «Змійка» мала елементарну концепцію і виявилася дуже успішною, заклавши основи для подальшого розвитку.

Через два роки Японія стала першою країною, яка перевела мобільні ігри на комерційну основу. Вона створила платформу NTT DoCoMo на базі технології i-mode, яка, своєю чергою, була еволюцією технології WAP. Японська новинка швидко поширилася Азією, а потім і по всьому світу. Одночасно з розвитком телефонів удосконалювалися ігри до них. Прикладом такої еволюції можна вважати компанію Namco, яка випустила в 2003 файтинг, в якому користувачі могли створити свого персонажа за допомогою камери, призначити йому характеристики сили і швидкості. Потім це фото можна було отруїти другові на телефон та викликати його на поєдинок. Через деякий час Panasonic випустила гру, в якій потрібно було годувати віртуального вихованця за допомогою фотографій їжі. Однак по-справжньому ринок мобільних ігор змінився в 2008 році, коли Apple запустила свій магазин додатків – App Store. Це здійснило «справжню революцію» у процесі створення та розповсюдження подібних забав.

Значне зростання мобільних ігор привернуло до себе увагу багатьох рекламодавців, оскільки вони дають можливість легко дістатися мільйонів користувачів. Через високий попит на умовно-безкоштовні ігри, були створені рекламні платформи, такі як Appnext, які дозволяють розробникам контактувати один з одним, щоб рекламувати свої проекти в інших іграх. Це стало справжньою золотою жилою для багатьох розробників та рекламодавців.

Мобільна ігрова індустрія змінюється зі швидкістю зростання начинки смартфонів протягом 10 років. Немає жодних сумнівів, що вона продовжить

зростати та еволюціонувати за допомогою нових технологій. А розвиток розширеної і віртуальної реальності може зовсім кардинально її трансформувати.

Мета дослідження – створення програмного продукту.

Задачі дослідження:

— аналіз предметної області;

— проектування концепту концепту ігрового застосунку;

— вибір інструментарію, методів реалізації та тестування ПП.

Об’єктом дослідження виконання роботи є процес аналізу мобільного геймінгу у жанрі три в ряд, вивчення їх ігрових механік.

Предметом дослідження є технології: ігровий рушій Unity, двовимірний графічний редактор Adobe Photoshop, мова програмування C#.

Для досягнення мети та вирішення завдання реалізації мобільного ігрового додатку необхідно здійснити:

1) опис предметного середовища (функціональної моделі, процесу діяльності);

2) огляд наявних аналогів;

3) постановку задачі;

4) опис архітектури рішення, що розробляється;

5) опис коду та інтерфейсу програми;

6) тестування.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Обґрунтування вибору середовища розробки

1.1.1. Вибір ігрового рушія

Ігровий рушій (game engine) — це середовище розробки програмного забезпечення, яке також називають «ігровим двигуном» або «ігровою структурою» з налаштуваннями та конфігураціями, які оптимізують і спрощують розробку відеоігор на різних мовах програмування. Ігровий механізм може включати у собі механізм візуалізації 2D або 3D графіки, сумісний з різними форматами імпорту, фізичний механізм, який імітує реальні дії, штучний інтелект (Artificial intelligence), який автоматично реагує на дії гравця, звуковий механізм, який керує звуковими ефектами, двигун анімації та безліч інших функцій.

Перед початком розробки будь-якого продукту насамперед потрібно обрати середовище у якому будемо працювати. Для розробки мобільних ігрових додатків існує велика кількість ігрових рушіїв. Щороку їхня кількість тільки зростає. Відмінність між ними полягає у тому, що жоден з них не зможе повністю задовільнити наші потреби. Частина з них дозволяє розробляти 2D ігри, інша частина 3D. Проте існує ще одна вагома різниця між ігровими двигунами – це їх доступність, адже існують як безкоштовні двигуни так і платні.

Проте ми будемо робити свій вибір між трьома найпопулярнішими і найбільшими ігровими двигунами – UNREAL ENGINE, Unity та Solar2D. Розглянемо їх детальніше.

Unreal Engine - один з найпопулярніших двигунів на сьогодні. У зв'язку з використанням C++ має найбільший спектр можливостей і, зокрема, свою візуальну систему програмування — Blueprint. Має потужне ком'юніті, багато відеоуроків, вже готових ассетів і часто використовується як при розробці AAA-ігор, так і невеликих проектів.

UI Підтримує більшість відомих платформ: Microsoft Windows, Linux, Mac OS та Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation

3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube, Nintendo Switch і т.д., в iOS та Android.

Починаючи з версії 4.0 присутній потужний редактор ШІ, редактор для створення кат-сцен і підтримка DirectX 12. В цілому, UI дозволяє досягти справді вражаючої картинки. У графічному плані - це один з найпотужніших двигунів з усіх існуючих.

З березня 2015 року рушій став повністю безкоштовним за умови, що прибуток від проектів, створених на основі двигуна не перевищує $3000 за квартал. Після перевищення потрібно буде відраховувати Epic Games 5% прибутку від продажу гри.

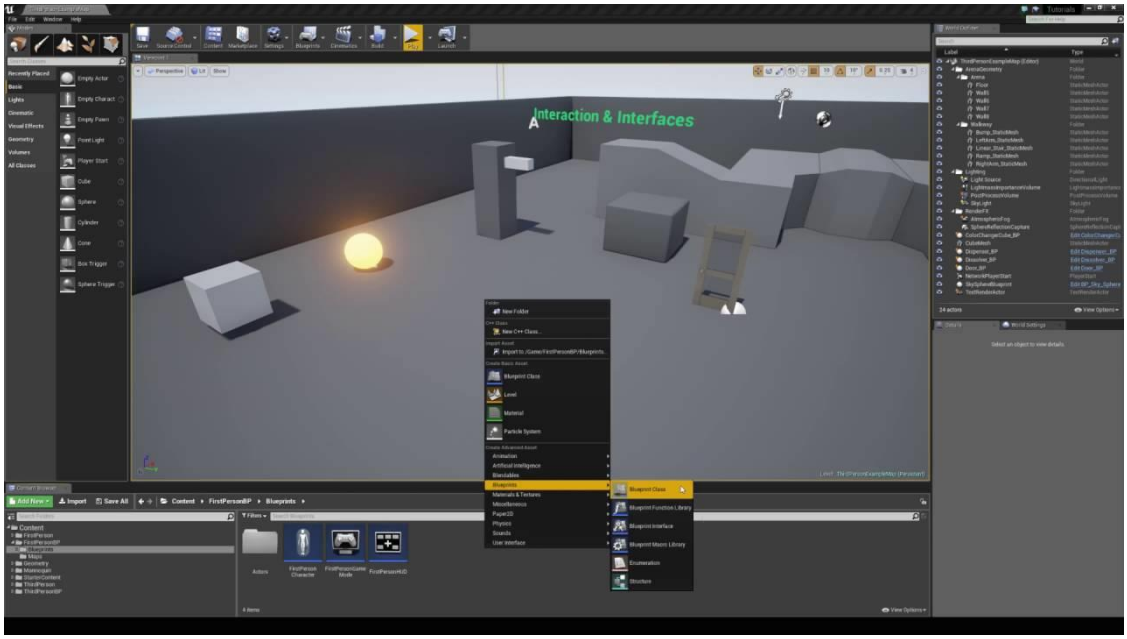

Рис. 1.1 – Інтерфейс редактора Unreal Engine

Unity – використовується повсюдно і будучи мультиплатформним, підходить під широкий спектр завдань, хоча графічно дещо поступається Unreal. Дозволяє працювати над 2D та 3D іграми, створюючи проекти під Windows, OS X, PlayStation 4, XBox, Windows Phone, Android, Apple iOS та Linux, у тому числі і під Wii, Xbox 360, Xbox One, Nintendo Switch. Є можливість створювати програми для запуску в браузерах за допомогою спеціального модуля Unity (Unity Web Player), що підключається, а також за допомогою реалізації технології WebGL.

Програми, створені за допомогою Unity, підтримують DirectX та OpenGL. Двигун використовується як розробниками AAA-ігор, так і Indie-студіями. Є власний Asset store, сильна та активна ком'юніті та вражаюча кількість документації та відео-уроків.

В наявності движка простий Drag&Drop інтерфейс, що складається з різних вікон і дозволяє робити налагодження гри прямо в редакторі. Двигун підтримує скриптові мови C# та JavaScript. Усі розрахунки фізики виробляються за допомогою NVIDIA PhysX.

Ліцензія Unity Personal є безкоштовною, однак, якщо дохід вашої компанії становить більше 100 000 $ на рік або якщо вам вдалося залучити на розробку більше 100 000 $, ви не маєте права використовувати Unity Personal. Можна буде скористатися версією Unity Plus для компаній, що заробляють до 200 000 $ на рік, або Unity Pro - вона не накладає жодних обмежень щодо доходу.

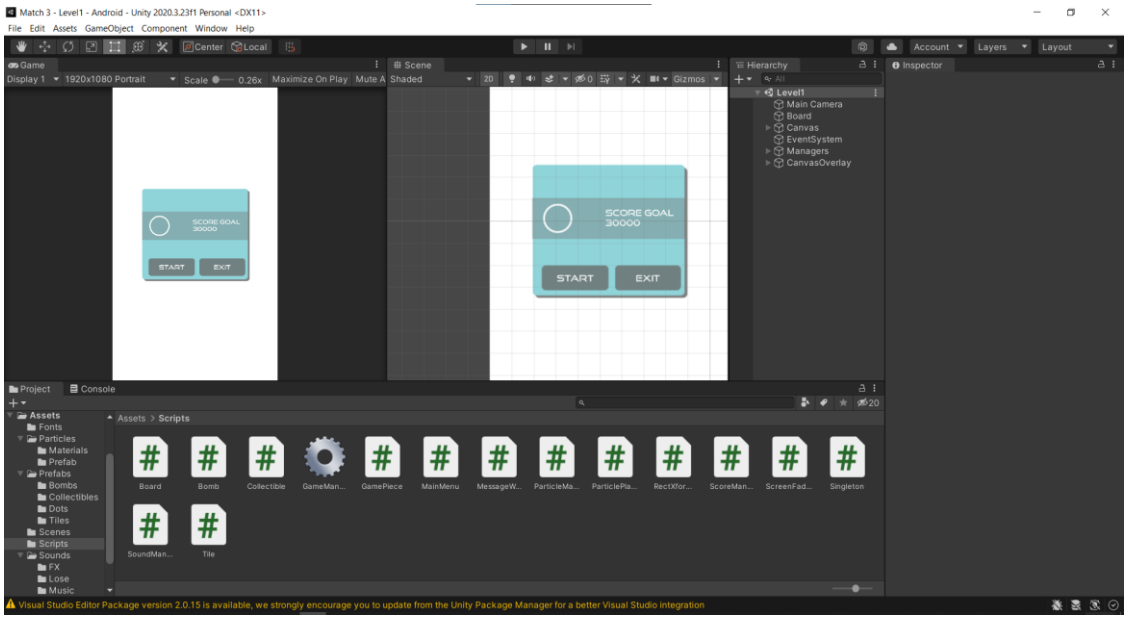

Рис. 1.2 – Інтерфейс редактора Unity

Solar2D - це безкоштовний, з відкритим кодом, платформний набір для розробки програмного забезпечення, спочатку розроблений компанією Corona Labs Inc., а тепер підтримується Владом Щербаном. Solar2D має повноцінний симулятор і найпростішу debug консоль. Також є дуже зручна функція Live Build – тестування програми одночасно на кількох пристроях та різних платформах у режимі реального часу. Для роботи цієї функції необхідно, щоб комп'ютер та

пристрій були підключені до однієї локальної мережі. При розробці програм використовується мова Lua – за ідеологією та реалізацією мова найближча до JavaScript.

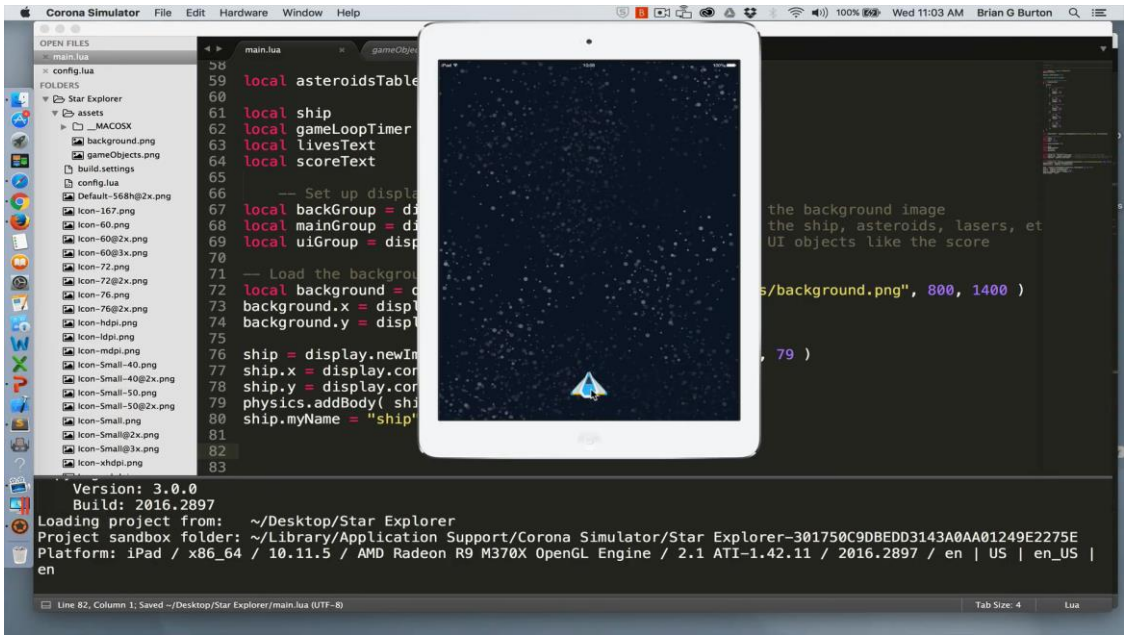

Рис. 1.3 – Інтерфейс редактора Solar2D

Щодо вибору середовища розробки, проведемо коротке порівняння між Unity, Unreal Engine та Solar2D.

Якщо спуститися на рівень коду, Unity виграє тим, що це C#, писати на якому легше. Unity має величезну спільноту, і на YouTube можна знайти багато інструкцій, тому навіть без навичок програмування за допомогою цього движка можна реалізувати щось нескладне.

Unreal Engine відмінно підходить для швидкого прототипування, великих ігор, має відкритий код, але для роботи потрібні знання в області C++. Великою перевагою є можливість створити повноцінну гру практично без коду.

У Unity трохи нижчі системні вимоги, сам двигун і проекти на ньому займають менше місця на диску.

Всі порівнювальні рушії здатні видавати приблизно однакову графіку. Спочатку вона краща в Unreal Engine, але все залежить від досвіду розробника.

З іншого боку, при створенні невеликих 2D та 2,5D-ігор, Unity – найкращий вибір, особливо коли йдеться про продукт із сенсорним інтерфейсом. Зворотний бік – закритий код Unity і без Bolt (інструмент віртуального програмування)

потрібно навчитися програмувати. Але навчання відносно просте через безліч безкоштовних та платних курсів.

Стосовно Solar2D, у зв'язку з відсутністю у даному рушію візуального редактора для когось робота з цим двигуном може бути трохи складніша, аніж у вище перерахованих рушіях. Solar2D створений виключно під розробку 2D програм. Написання програмного коду відбувається на мові програмування Lua, яка дуже схожа до Java Script.

Як бачимо, все залежить від того, який проект нам потрібно зробити. Якщо це мобільна гра у 2D або 2,5D – ми явно оберемо Unity через простоту. Unreal Engine не створювався для 2D-ігор і реалізація проекту додасть непотрібної складності. Але якщо планується файтинг, гонка, серйозний шутер - краще вибрати Unreal Engine.

Провівши детальний аналіз та взявши до уваги всі плюси та мінуси порівнюваних ігрових рушіїв, було прийнято рішення працювати з Unity. Адже, даний ігровий рушій ми детально розглядали під час проходження курсу «Розробки ігрових додатків на Unity», це дало загальні поняття та практику користування інтерфейсом та подальшу розробку ігрового додатку.

1.1.2. Вибір середовища для розробки 2D графіки

Графічний редактор — це тип комп’ютерної програми, яка дозволяє користувачеві редагувати та змінювати графічні зображення різними методами.

Ці типи програм-редакторів зазвичай поділяються на два різних типи: редактори растрової графіки та редактори векторної графіки. Відмінності між цими двома типами зазвичай впливають на типи додатків, для яких використовується програма, і те, як остаточно створені зображення відображаються на екрані або під час друку. Програма графічного редактора може бути безкоштовною але лише з кількома функціями або дорогим професійним програмним забезпеченням для графічного мистецтва з десятками інструментів і функцій.

Для аналізу графічних редакторів було обрано дві популярні програми: Adobe Photoshop та Adobe illustrator.

Adobe Photoshop – це графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe.

Серед переваг Adobe Photoshop можна виділити наступні:

— висока якість обробки графічних зображень;

— зручність і простота в експлуатації;

— великі можливості, які дозволяють виконувати будь-які операції створення і обробки зображень;

— широкі можливості автоматизації обробки растрових зображень, які базуються на використанні сценаріїв;

— сучасний механізм роботи з кольоровими профілями, які допускають їх втілення в файли зображень з метою автоматичної корекції кольорових параметрів при виводі на друк для різних пристроїв;

— великий набір команд фільтрації, за допомогою яких можна створювати найрізноманітніші художні ефекти.

Adobe Illustrator — професійний графічний редактор для створення та редагування векторної графіки від компанії Adobe.

Серед переваг Adobe Illustrator можна виділити настпні:

— у ньому є кілька готових інструментів, які надають нам різноманітні форми;

— ми можете зробити банер будь-якого розміру, починаючи від маленької точки до великого згинання;

— імпорт, редагування та модифікація зображень можна проводити на одній платформі;

— це зручно, якщо він використовується одночасно з іншим продуктом Adobe.

Порівняльний аналіз програмних продуктів для розробки мобільного додатку виконано у таблиці нижче.

Порівняльна таблиця графічних редакторів

Критерії	Програмне забезпечення	
	Photoshop	Illustrator
Вартість продукту	Безкоштовний пробний період	Безкоштовний пробний період
Локалізація (українська мова)	Присутня	Присутня
Можливість постобробки зображення	Присутня	Обмежена
Можливість роботи з растровими зображеннями	Присутня	Обмежена
Можливість роботи з векторними зображеннями	Обмежена	Присутня

1.1.3. Висновки

Взявши до уваги переваги вище описаних програм найкращим рушієм для створення мобільної гри є використання Unity із застосуванням мови програмування C#.

Розробка програмного коду буде відбуватися у середовищі Microsoft Visual Studio. Для створення 2D графіки буде використана програма Adobe Photoshop.

Категорія	**Варіанти програм**	**Висновок**
Ігровий рушій	Unity, Solar2D, Unreal Engine	При створенні невеликих 2D та 2,5D-ігор, Unity – найкращий вибір, особливо коли йдеться про продукт із сенсорним інтерфейсом.
Мова програмування	C#, C++, Lua	Вибравши ігровий рушій Unity він визначив C# як основну мову програмування для розробки коду.

Середовище розробки програмного коду	Microsoft Visual Studio,VS Code	Visual Studio – це IDE для роботи з мовою програмування C#. Використання IntelliSense та навігації по коду спрощує переміщення по скриптам.
Створення2D графіки	Adobe Photoshop, Adobe illustrator	Photoshop є найпопулярнішим редактором для 2D графіки. У ньому підтримується велика кількість форматів, які використовуються в Unity.
Додаткові ресурси	Freesound (https://freesound.org/)	Freesound — це спільна база даних звуків з ліцензією Creative Commons.

1.2. Опис предметного середовища

Unity – міжплатформне середовище розробки ігор. Цей ігровий рушій дозволяє створювати програми, що працюють більш ніж на 20 різних операційних системах, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші. Випуск Unity відбувся у 2005 році і з того часу триває постійний розвиток.

Основними перевагами Unity є наявність візуального середовища розробки, міжплатформної підтримки та модульної системи компонентів. До недоліків відносять поява складнощів при роботі з багатокомпонентними схемами та складнощами при підключенні зовнішніх бібліотек.

На Unity написані тисячі ігор, додатків та симуляцій, які охоплюють безліч платформ та жанрів. При цьому Unity використовується як великими розробниками, так і незалежними студіями. об'єкти в реальному часі і відразу ж тестувати отриманий результат.

Для того щоб писати гру у ігровому двигуні Unity 3D, можна не збирати команду, а розробляти її самому. Також потрібно вміти програмувати на мові C#. Це дуже потужна мова програмування, її функціонала більше ніж достатньо для виконання тих, або інших функцій. Хоча він не відноситься до низькорівневих мов програмування, це не заважає йому справлятися із своєю задачею.

1.3. Огляд існуючих платформ

Головною особливістю, яка відрізняє смартфон від звичайного мобільного телефону є наявність операційної системи у першому.

Операційних систем для смартфонів існує не так багато, їх всього три:

— Android;

— iOS;

— Windows Phone.

Зрозуміло, що вони багато в чому відрізняються: мають свої особливості, переваги та недоліки, у кожної своя частка ринку. Проте ми розглянемо дві найпопулярніші операційні системи на даний момент: Android та iOS.

Android – це мобільна операційна система, яка існує вже майже 15 років. операційна система для смартфонів, планшетів та нетбуків. Google придбала розробника програмного забезпечення Android inc. у 2005 році. Операційна система Android заснована на модифікованому ядрі Linux.. Крім того, існують інші операційні системи, які підтримують програми Android, зокрема ОС Chrome і Windows 11.

Android на сьогоднішній день є найпопулярнішою операційною системою у світі. За оцінками, він працює на 2,5 мільярдах активних пристроїв по всьому світу з понад 3 мільярдами користувачів — або приблизно 39% усього населення планети. Це значно перевершує iOS від Apple, яка є другою за популярністю операційною системою в усьому світі.

Переваги розробки мобільних додатків під ОС Android:

— число користувачів пристроїв на Андроїд величезна;

— мобільний додаток дозволить розширити аудиторію за допомогою просування через магазин Google Play;

— більшість користувачів вже підключили карти або інші засоби оплати до Google Play;

— при розробці Android-додатка ми отримуємо більше свободи для обробки даних користувачів. Google пред'являє менше вимог до складання політики конфіденційності, ніж Apple.

— платформа Google Play менш вимоглива до оформлення додатка, ніж App Store.

Недоліки розробки мобільних додатків під ОС Android:

— довший час розробки;

— нижчий середній дохід;

— більше можливих помилок;

— набагато більше видів пристроїв, під які потрібно пристосувати ігровий додаток.

Apple iOS – це операційна система, яка існує вже більше десяти років. По суті, її назва перекладається як "операційна система для iPhone", але сьогодні вона встановлюється і на різні планшети iPad-и.

Переглянемо переваги розробки мобільних додатків під iOS:

— платоспроможність клієнтів Apple. Наприклад, частка завантаження платних додатків для iPhone - 9% (проти 4% у Андроїд);

— зручність розробки і тестування. Набагато менше видів пристроїв, під які потрібно пристосувати її. Пристрої на iOS робить тільки Apple;

— менше можливих помилок.

Недоліками ж є:

— у два рази менше користувачів;

— тривала публікація з багатьма обмеженнями;

— необхідність придбання спеціального обладнання.

Розробка додатків як для Android так і iOS має свої переваги та нюанси залежно від цілей, аудиторії та варіантів монетизації, які розробник збираєтеся реалізувати.

У нашому проекті вирішальним кроком стане пункт: «Тривала публікація з багатьма обмеженнями». Так як, на останньому етапі розробки, кваліфікаційна робота повинна бути кінцевим робочим ігровим додатком, який можна буде завантажити у крамницю Google Play для того, щоб користувачі могли мати вільний та легкий доступ для завантаження додатку на свій мобільний пристрій Android.

Також ключовим аспектом у розробці ігрового додатку є монетизації, яка набагато простіша для смартфонів під керуванням операційної системи Android, та беручи до уваги кількість пристроїв які працюють на даній платформі, можна спрогнозувати, що реклама принесе більший дохід.

Оскільки, тема кваліфікаційної роботи – «Розробка ігрового застосунку головоломки у стилі «три в ряд» під мобільну систему Android на рушію Unity», додаток будемо розробляти для пристроїв під керуванням ОС Android.

1.4. Огляд наявних аналогів

На початковому етапі розробки ігрового додатку необхідно проаналізувати схожі мобільні додатки: їх механіки, цілі, дизайн. До уваги були взяті наступні додатки:

— Match 3 Candy

— Juice Pop Mania

— Fruits Forest : Rainbow Apple

Match 3 Candy – це захоплююча, весела, непередбачувана та складна гра. Розробник нас запевняє, що даний ігровий додаток нам сподобається якщо ми не любителі ігор в жанрі "Три в ряд". У цій класичній 2D-грі є все: просте управління, привабливий ігровий вигляд, різноманітність рівнів.

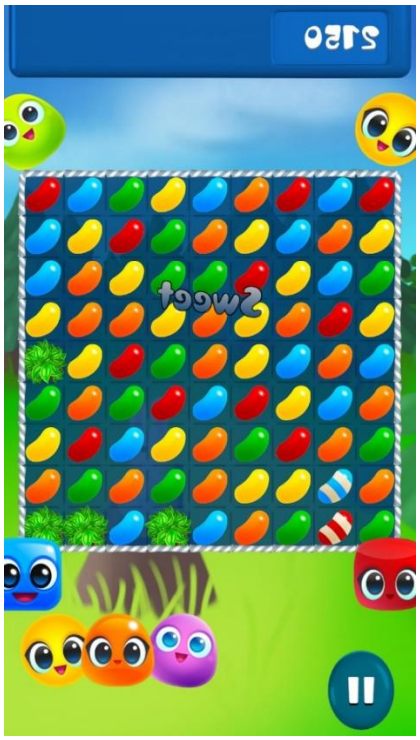

Рис.1.4 – Ігровий процес гри в Match 3 Candy

Juice Pop Mania - ця гра є абсолютно безкоштовною у даному жанрі ігор. У даному ігровому застосунку присутні сотні ігрових рівнів, цікава та приємна графіка, спокійна музика.

Рис. 1.5. – Ігровий процес гри в Juice Pop Mania

Fruits Forest досить цікава гра-головоломка з приємною ігровою графікою та музикою. Особливості цього ігрового додатку полягають у тому, що тут немає обмеження «життів» і можна грати досхочу, проте це є також і мінусом даної гри – до неї швидко звикаєш через відсутніх тих самих «життів». До наступних позитивних особливостей застосунку можна віднести наступні два пункти: можливість грати без підключення до Інтернету та його малий розмір, всього 20 мб.

Рис. 1.6 – Процес гри в Fruits Forest

Жанр ігор-головоломок «Три в ряд» є досить популярним у сфері мобільних ігор. Загальні механіки успішно реалізуються успішно на кожній з платформ, але спосіб реалізації може відрізнятися.

Розглянувши популярні ігрові застосунки представників вище вказаного жанру були обрані основні та допоміжні механіки гри, стилі зовнішнього вигляду та предметів оточення для майбутнього ігрового застосунку.

Висновок до розділу 1

У першому розділі виконано вибір ігрового рушія, середовища для розробки 2D моделей, здійснено опис предметного середовища, а також оглянуто існуючі платформи для розробки і наявні аналоги. Після огляду наявних аналогів були обрані основні та допоміжні механіки гри, стилі зовнішнього вигляду та предметів оточення для майбутнього ігрового застосунку.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області

Три в ряд – тип ігрової механіки казуальних головоломок, який здобув величезну популярність за рахунок своєї простоти та гнучкості. Суть ігор, що належать до цього типу, зводиться до пересування фішок по ігровому полю та складання ланцюжків або лінійок із трьох і більше елементів. Простір, на якому розташовуються фішки, є орнаментом з квадратів або, що значно рідше, шестикутників. Гексагональна структура поля складніша, як з позиції гравця, так і з позиції розробника, з чим, мабуть, пов'язана її невелика популярність.

Загальний прибуток від всіх ігор-головоломок збільшується щороку. Відомо, що ігри даного жанру принесли 4,9 млрд. доларів у першій половині 2021 року, що на 56,8% більше, ніж минулого року. Оскільки ігри жанру «три в ряд» є найпопулярнішими серед ігор-головоломок, це основне джерело доходу всіх компаній-розробників.

Ігровий додаток повинен містити наступні механіки:

— підрахунок набраних очок;

— відкриття наступного рівня після успішного проходження попереднього.

2.2. Постановка задачі

У результаті аналізу предметної області було сформульовано мету кваліфікаційної роботи: розробка ігрового застосунку-головоломки «Combo3». Метою даного застосунку є веселе проведення часу та покращення уваги, логічного мислення гравців.

Для досягнення мети кваліфікаційної роботи було визначено перелік основних завдань:

— виконати аналіз предметної області, та огляд наявних аналогів;

— визначити інструментарій розробки;

— спроектувати функціонал додатку;

— розробити дизайн ігрового продукту;

— розробити механіки;

— налаштувати сцени та розробити ігрові рівні;

— протестувати ігровий додаток.

Мобільний додаток повинен забезпечувати веселе та корисне проведення часу в грі.

2.3. Розробка архітектури ігрового додатку

Створення ігрового застосунку дуже трудомісткий і складний процес. В ході цього процесу величезна кількість елементів об'єднується в програму призначену для того, щоб гравець міг цікаво провести вільний час.

Для нашого ігрового застосунку необхідно реалізувати модуль, який відповідає за ігрову логіку і фізику, який буде керувати взаємодією ігрових об'єктів на сцені, перемикати сцени в залежності від обраного пункту меню, завантажувати рівні і так далі.

Розгорнутий опис функціональних вимог здійснюється на етапі проектування програми. Для того щоб деталізувати вимоги, необхідно виділити процеси, що відбуваються в заданій предметній області. Опис таких процесів на UML виконується у вигляді прецедентів (Рис. Діаграма прецедентів). Прецеденти є сценарієм або варіантом використання ігрового програми при взаємодії із зовнішнім середовищем. За допомогою прецедентів описується функціонування гри з точки зору користувача, який називається в UML актором (Гравець). Актор взаємодіє з системою і використовує її функціональні можливості для досягнення певної мети.

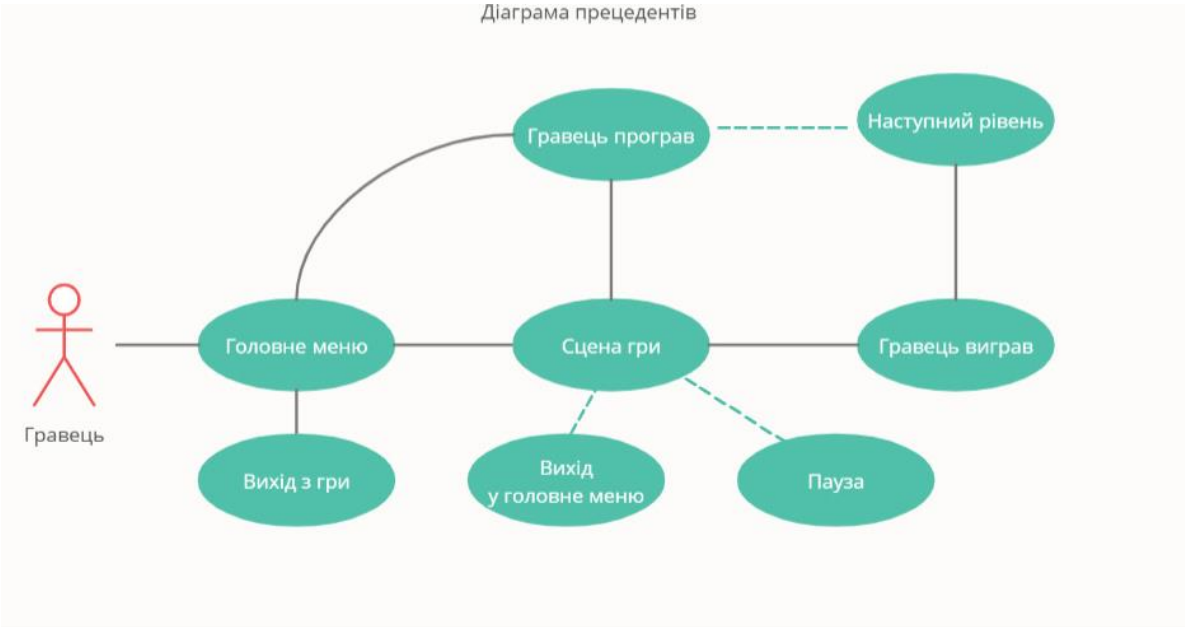

Рис. 2.1 – Діаграма прецедентів

2.4. Проектування інтерфейсу гри

В результаті аналізу предметної області та вимог, був розроблений інтерфейс ігрового додатка. Кожне меню або ігровий режим є окремим станом, який має свій клас, призначений для його перемикання.

Основне ігрове поле містить ігрові об'єкти логіки представлення. На екрані також присутні лічильники набраних очок, поточний рівень на якому знаходиться гравець, показник кількості ходів, що залишилися у гравця.

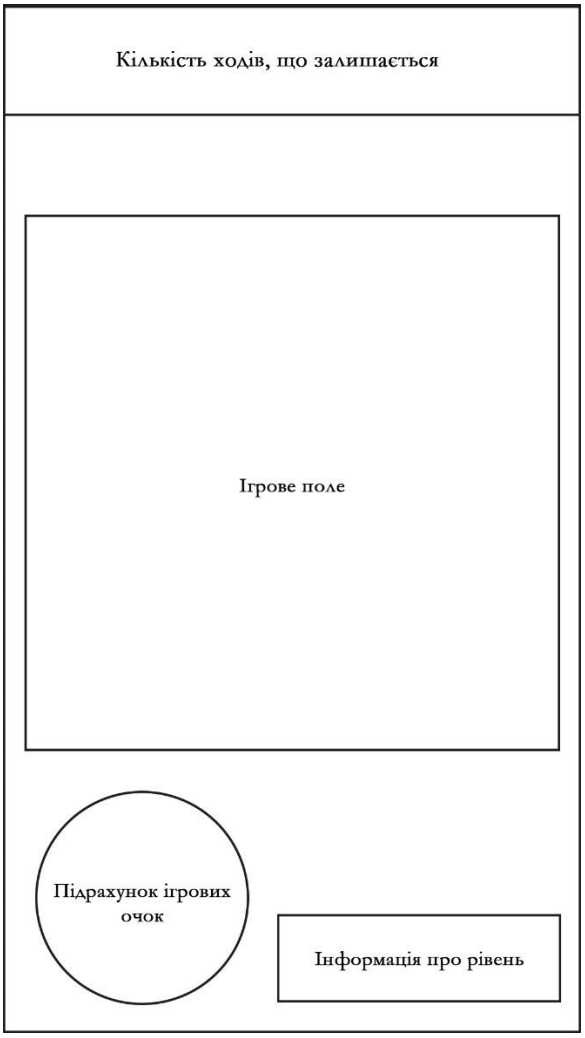

Рис. 2.2 – Прототип інтерфейсу ігрової сцени

2.5. Система меню

Будь-який ігровий застосунок має своє меню. Дана система дозволяє гравцеві запускати гру, виходити з програми. Для її проектування вкрай зручно використати діаграму станів.

Головне меню складається з:

— Кнопка «Грати»

— Кнопка «Вихід»

Рис. 2.3 – Діаграма станів – Старт гри

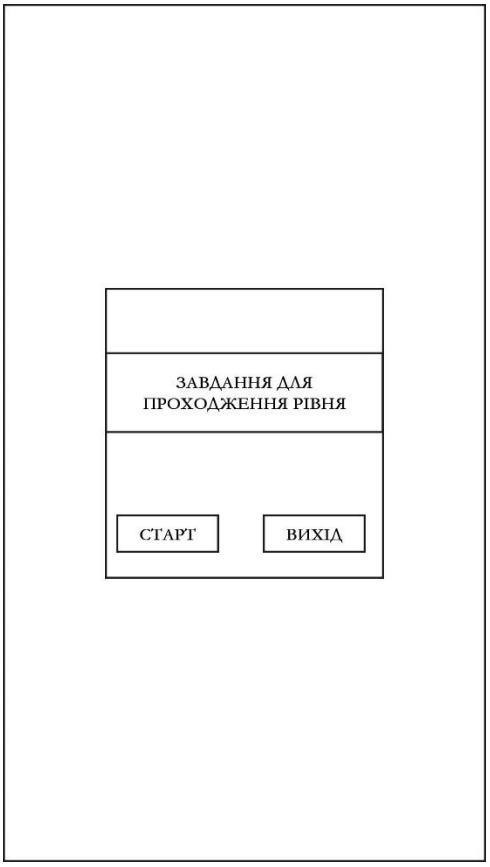

Рис. 2.4 – Прототип меню

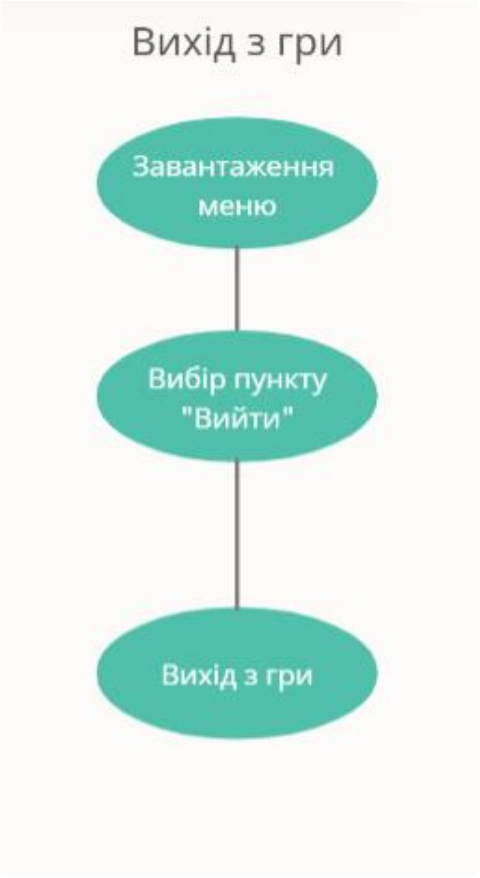

Рис. 2.5 – Діаграма станів – Вихід з гри

Висновок до розділу 2

У другому розділі було підготовлено постановку задачі, виконано розробку архітектури ігрового додатку. Також в результаті аналізу предметної області та вимог, був розроблений прототип інтерфейсу ігрового поля та головного меню додатку. Кожне меню або ігровий режим є окремим станом, який має свій клас, призначений для його перемикання.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Засоби розробки включають технічну документацію таких середовищ: Microsoft Visual Studio, Adobe Photoshop та Unity, – для роботи з інтерфейсом і бібліотеками, які використовуються при створенні ігрового продукту, публікації на відеохостингу YouTube, а також статті в Інтернеті.

3.2. Вимоги до технічного та програмного забезпечення

Для створення ігрового застосунку впроваджено наступні вимоги до програмного забезпечення:

— C# – мова програмування;
— Microsoft Visual Studio – середовище програмування;
— Unity – ігровий рушій;
— Freesound – база даних звуків з ліцензією Creative Commons;
— Adobe Photoshop – створення двовимірної графіки;

Вимоги до технічного забезпечення пристрою для розробки:

— Процесор: Intel Core i5-4310
— Відеокарта: NVIDIA® Quadro K2100M 2 ГБ
— Оперативна пам'ять: 16 ГБ
— Операційна система: Windows 10 64-розрядна

Вимоги до технічного забезпечення мобільного пристрою:

— Процесор: 4-ядерний
— Оперативна пам'ять: 2 ГБ
— Місце на диску: 23 МБ
— Операційна система: Android 5.0 і вище

3.3. Опис системної реалізації

3.3.1. Створення ігрового поля

Щоб створити ігрове поле у Unity для гри жанру «три в ряд» потрібно створити окрему сцену, під назвою Board.

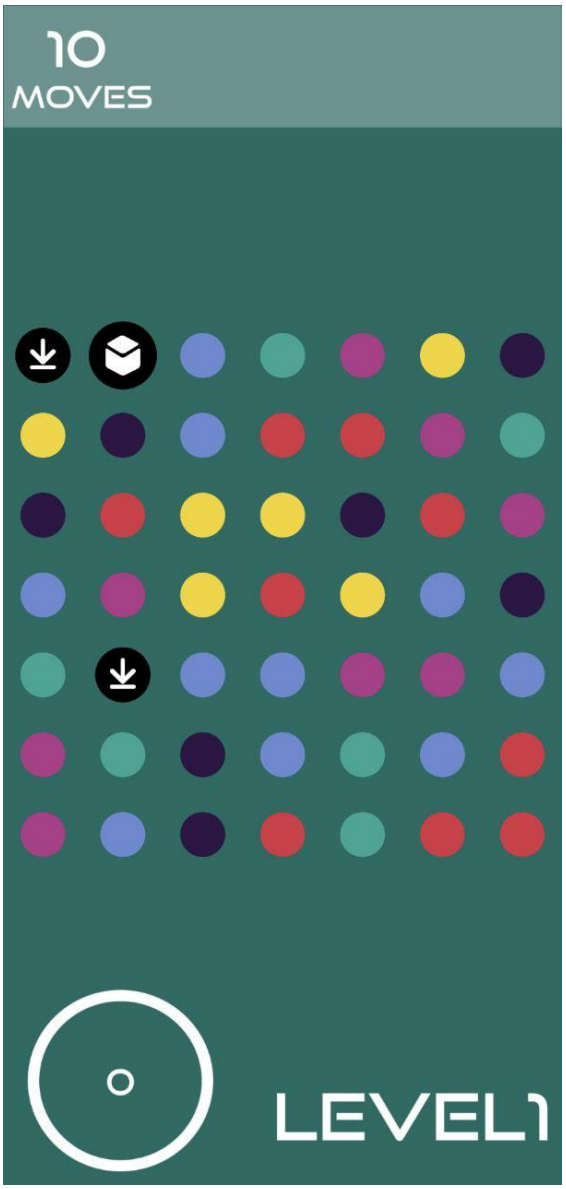

Рис. 3.1 – Ігрове поле

В загальному скрипт Board містить наступні елементи:

— Розміри ігрового поля;

— Ігрових елементів:

— Ігрових елементів у вигляді бомб:

— Допоміжні ігрові елементи;

— Початкове розміщення ігрових елементів на полі (для створення нових рівнів);

— Налаштування швидкості переміщення ігрових елементів на ігровому полі.

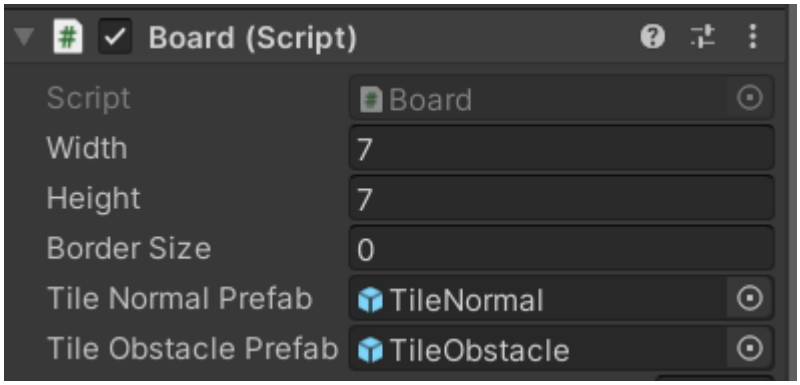

Рис. 3.2 – Board

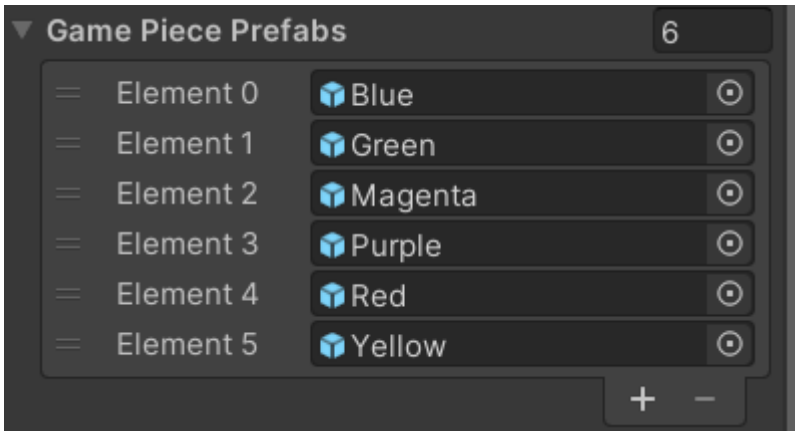

Рис. 3.3 – Скрипт Board, основні ігрові елементи

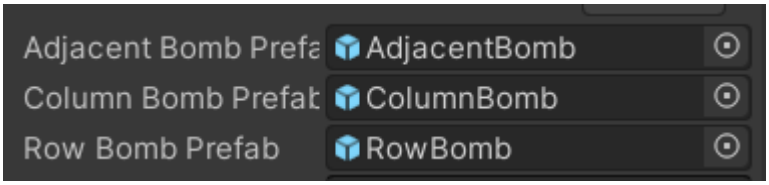

Рис. 3.4 – Скрипт Board, основні ігрові елементи у вигляді бомб.

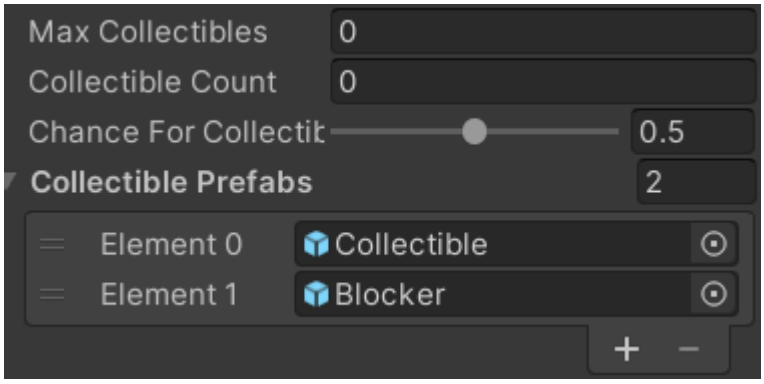

Рис. 3.5 – Скрипт Board, налаштування допоміжних ігрових елементів

Рис. 3.6 – Скрипт Board, налаштування швидкості переміщення ігрових об’єктів

3.3.2. Система ігрових очок

Для того щоб зробити нашу гру цікавішою та азартнішою м розробили систему здобуття очок. Для наших гравців було додано можливість здобувати ігрові очки, для цього до всіх елементів ігрового поля було підключено скрипт GamePiece. Частина коду у цьому скрипті дає можливість надати ігровим елементам собівартість. За знищення звичайних ігрового елемента ми отримуємо 20 очок, за знищення бомби – 50 очок, за знищення допоміжних елементів – 100 та 150 очок, в залежності від виду елементу.

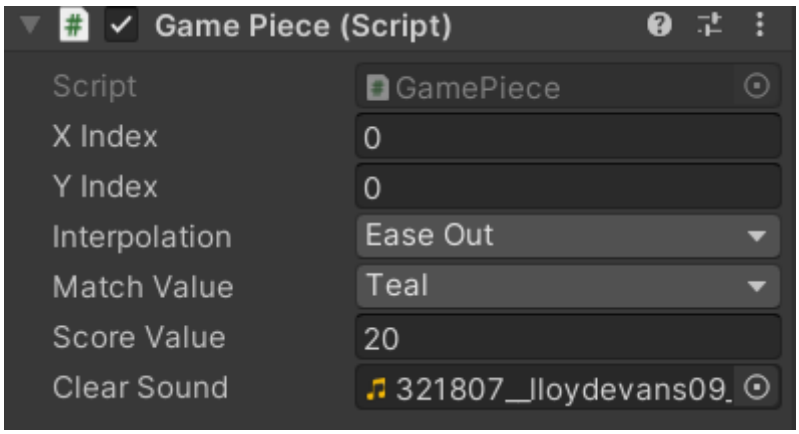

Рис. 3.7 – Скрипт Game Piece

Також для підрахунку, відображення і контролю набраних очок було створено сцену ScoreBorder.

Рис. 3.8 – Підрахунок набраних очок

3.3.3. Створення анімацій

Основною частиною будь-якої сучасної гри є анімації. У наш ігровий додаток було додано два види анімацій: анімація очищення ігрового поля та анімація для кращого відображення бомб на ігровому полі.

Для додавання анімацій було створено сцену ParticleManager з однойменним скриптом.

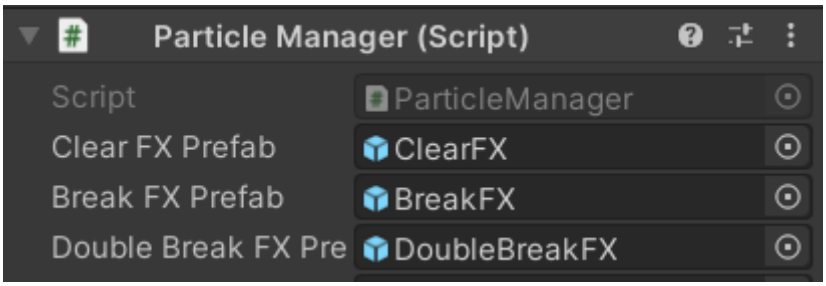

Рис. 3.9 – Скрипт ParticleManager

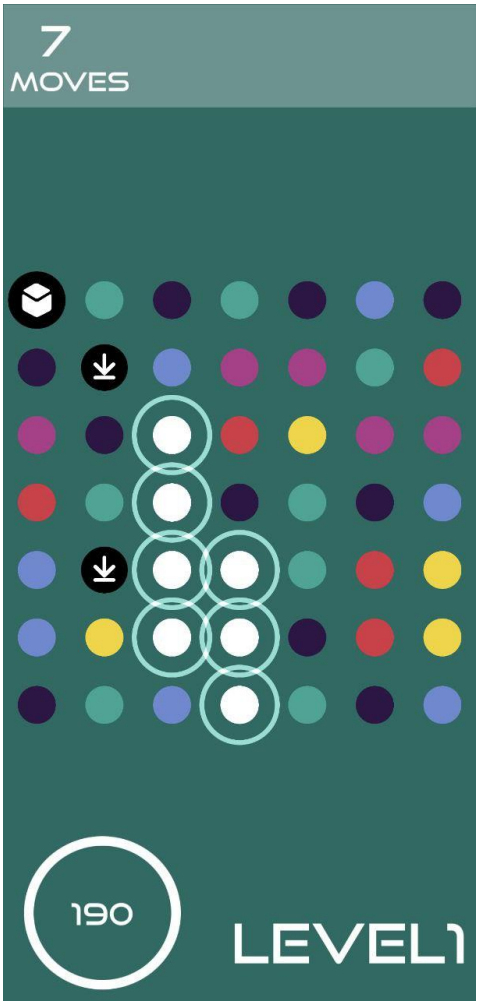

Рис. 3.10 – Анімація очищення ігрового поля

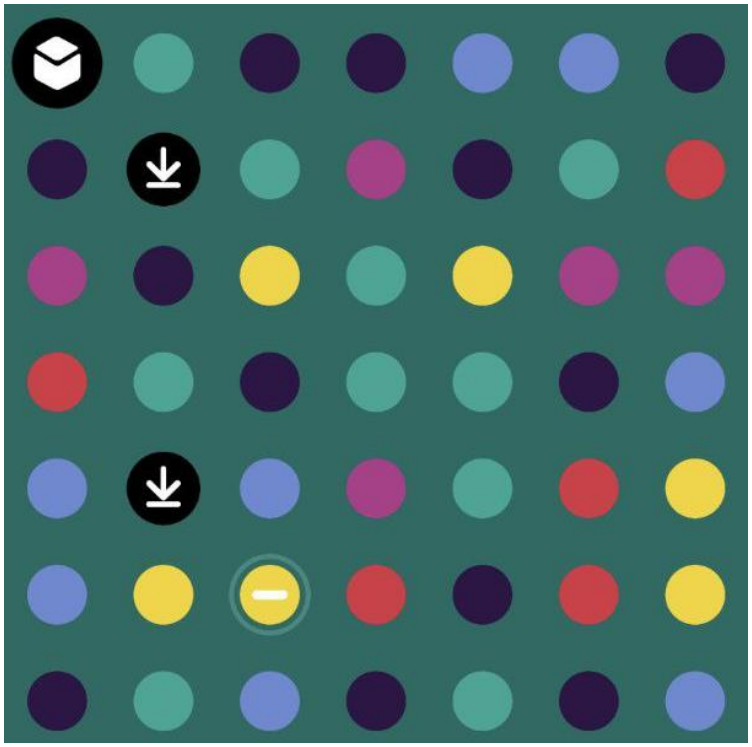

Рис. 3.11 – Анімація для бомб

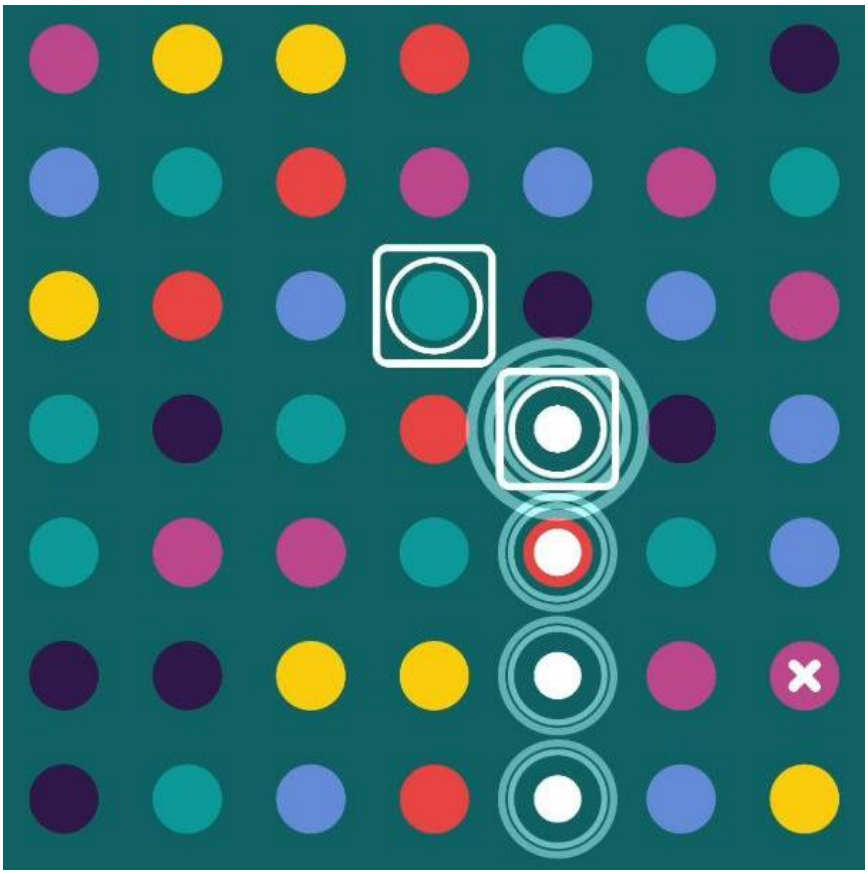

Рис. 3.12 – Анімація знищення допоміжного ігрового елементу

3.3.4. Створення меню

Щоб створити меню в Unity потрібно небагато часу, для цього потрібно створити окрему сцену, під назвою MainMenu.

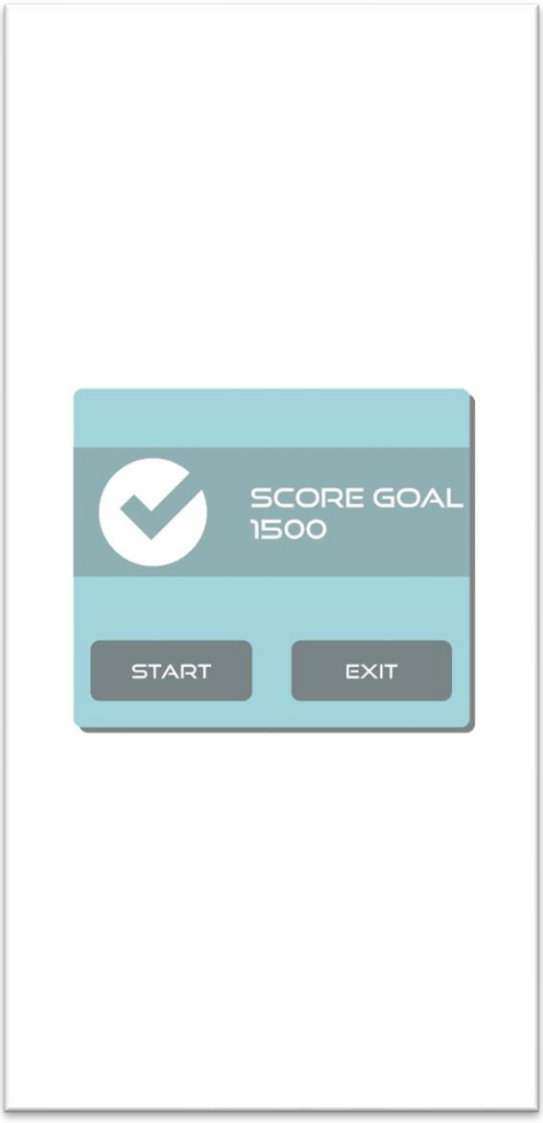

Рис. 3.13 – Сцена головного меню

Для кращого візуального ефекту появи меню до нього було підключено скрипт RectXformMover. Він призначений для щоб зробити наше меню випадаючим, тобто при запуску гри меню переміщується зверху до центру екрану, після натискання кнопки «Старт» меню зміщується від центру вниз за ігрову площину.

Також при програші чи виграші будь-якого ігрового рівня з’являється відповідне меню, яке інформує гравця про програш/виграш.

Рис. 3.14 – Інформування гравця про перемогу

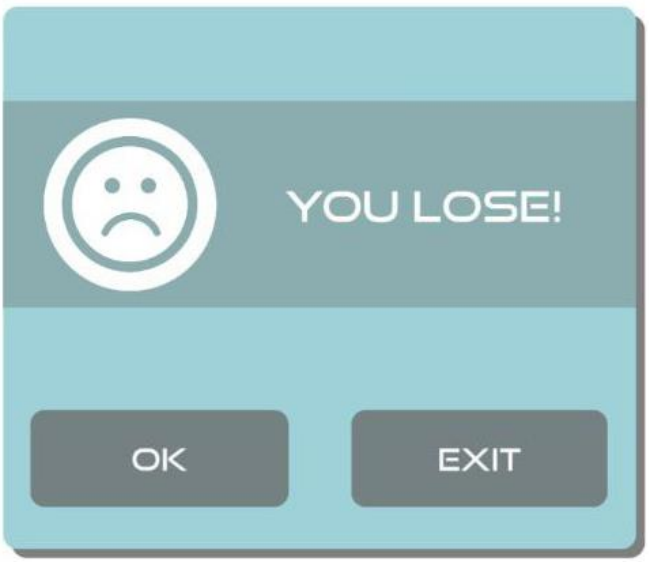

Рис. 3.15 – Інформування гравця про поразку

3.3.5. Додавання музики та звукових ефектів

До нашого ігрового додатку ми додали спеціальну сцену з однойменним скрипом SoundManager. Він потрібен, щоб відтворювати аудіо у проекті. Це звук очищення ігрового поля, вибух бомб, звукове супроводження перемоги та програшу гравця, а також музика у головному меню та під час процесу. Використані аудіо були взяті з сайту Freesound – бази звуків з ліцензією Creative Commons.

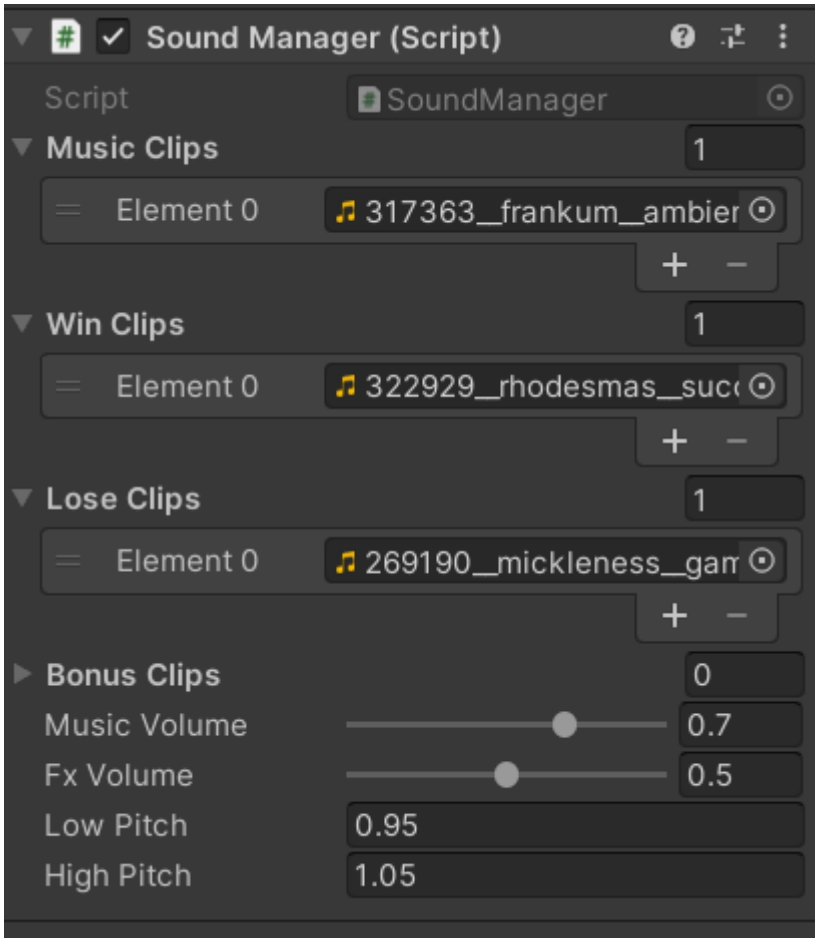

Рис. 3.16 – Скрипт SoundManager

3.3.6. Додавання інформаційної панелі

Задля кращої взаємодії гравця з нашим ігровим додатком було додано дві інформаційних панелі: прозора та напівпрозора, які відповідно знаходяться у нижній та верхній частинах екрану.

У верхній частині екрану розміщується поле у якому знаходиться інформація про кількість ходів, які залишилися у гравця.

Нижнє поле гри дає можливість гравцеві спостерігати за наявною кількістю очок, про яку ми вже згадували раніше, а також інформує його про те на якому рівні він зараз перебуває.

Рис. 3.17 – Верхнє інформаційне поле

Рис. 3.18 – Нижнє інформаційне поле

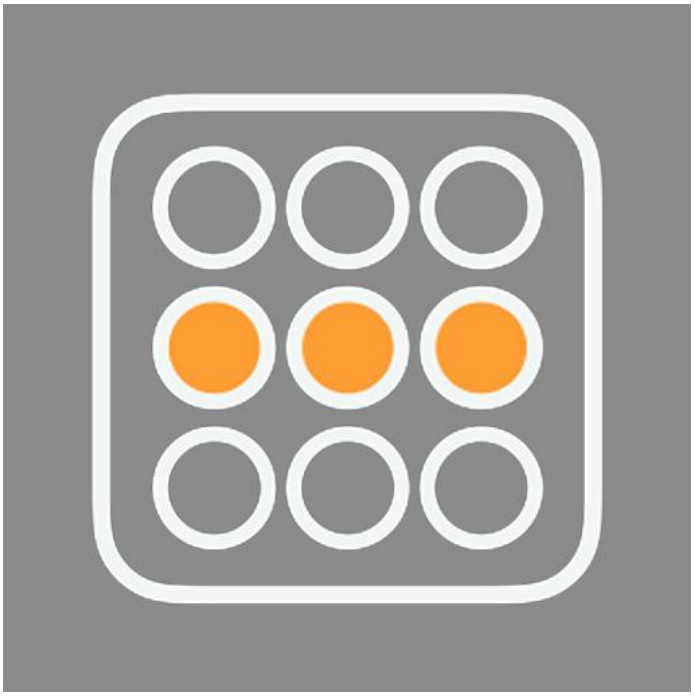

Рис. 3.19 – Іконка ігрового додатку

3.3.7. Додавання стороннього шрифту

Щоб підкреслити стиль гри було замінено основний стиль за замовчуванням Arial на сторонній безкоштовний шрифт Good Times Regular від Typodermic Fonts Inc.. Посилання на ліцензію про використання даного шрифти знаходиться у пункті «Список використаних джерел».

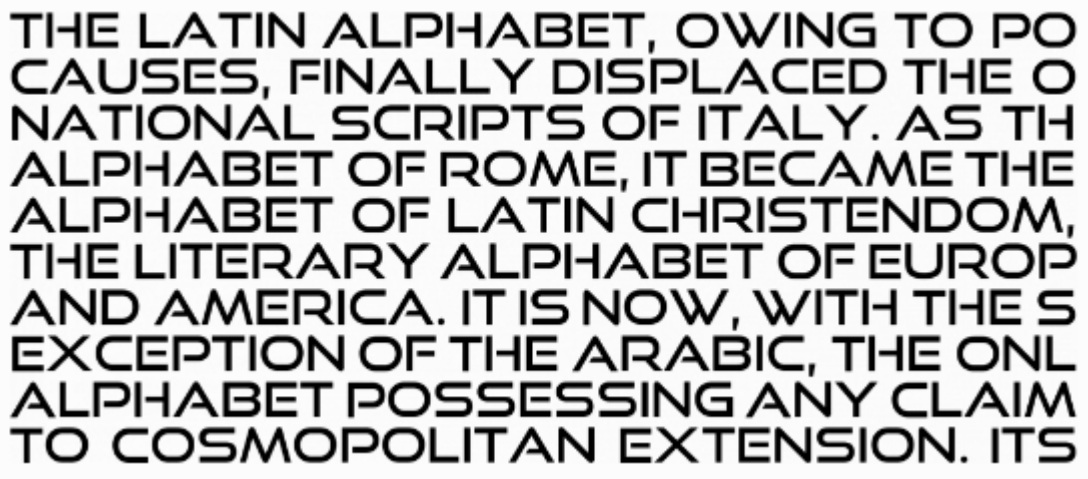

Рис. 3.20 – Шрифт Good Times Regular

Висновок до розділу 3

У процесі роботи над кваліфікаційним проектом було затверджено засоби розробки, які будуть використовуватись під час розробки проекту, вимоги до технічного та програмного забезпечення, також було описано процес реалізації ігрового застосунку за допомогою засобів розробки.

РОЗДІЛ 4
ТЕСТУВАННЯ

4.1. Функціональне тестування

Мета функціонального тестування – виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог. Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв'язку функціональних задач.

За результатам функціонального тестування було створено таблицю з результатами.

№	Назва тесту	Очікуваний результат	Отриманий результат	Тест пройдений?
1	Відображення головного меню гри	На екрані відображається головне меню гри з кнопками «Start» та «Exit»	На екрані відобразилось головне меню гри з кнопками «Start» та «Exit»	+
2	Робота кнопки «Start»	При натиску на кнопку «Start» очікуємо на початок гри	При натиску на кнопку «Start» розпочалася гра	+
3	Робота кнопки «Exit»	При натиску на кнопку «Exit» очікуємо вихід з гри	При натиску на кнопку «Exit» вийшли з гри	+
4	Робота ігрового поля. Переміщення ігрових елементів	При перетягуванні ігрових елементів вони змінюються між собою місцями.	При перетягуванні ігрових елементів вони змінюються між собою місцями.	+
5	Перевірка працездатності	При переміщенні трьох однакових	При переміщенні трьох однакових	+

	логіки гри. Зникнення трьох однакових елементів	ігрових елементів в один ряд вони зникають	ігрових елементів в один ряд вони зникли	
6	Перевірка працездатності логіки гри. Зникнення чотирьох однакових елементів	При переміщенні чотирьох однакових ігрових елементів у один ряд вони зникають	При переміщенні чотирьох однакових ігрових елементів у один ряд вони зникли	+
7	Перевірка працездатності логіки гри. Створення бомб	При переміщенні чотирьох і більше однакових ігрових елементів у один ряд вони зникають і на їх місці з’являються бомби.	При переміщенні чотирьох і більше однакових ігрових елементів у один ряд вони зникають і на їх місці з’являються бомби.	+
8	Перевірка працездатності логіки гри. Робота бомб	При переміщенні трьох і більше однакових за кольором ігрових елементів, у тому числі бомби, в один ряд вони бомба спрацьовує	При переміщенні трьох і більше однакових за кольором ігрових елементів, у тому числі бомби, в один ряд вони бомба спрацьовула	+
9	Перевірка працездатності «ScoreManager»	При знищенні ігрових елементів нараховуються очки	При знищенні ігрових елементів нараховуються очки	+
10	Перевірка працездатності анімацій	При знищенні основних та допоміжних	При знищенні основних та допоміжних	+

		елементів спрацьовує анімація	елементів спрацьовує анімація	
11	Перевірка працездатності «SoundManager»	При вході у гру включається музика, яка супроводжує гравця під час гри. Поразка/перемога гравця супроводжується різними звуковими сигналами. Очищення ігрового поля супроводжується відповідним звуковим сигналом	При вході у гру включається музика, яка супроводжує гравця під час гри. Поразка/перемога гравця супроводжується різними звуковими сигналами. Очищення ігрового поля супроводжується відповідним звуковим сигналом	+
12	Перевірка працездатності меню після поразки гравця	Після поразки з’являється меню із кнопками «OK» та «EXIT», яке інформує гравця про поразку	Після поразки з’являється меню із кнопками «OK» та «EXIT», яке інформує гравця про поразку	+
13	Перевірка працездатності меню після перемоги гравця	Після перемоги з’являється меню із кнопками «NEXT» та «EXIT», яке інформує гравця про перемогу та дозволяє перейти на новий рівень	Після перемоги з’являється меню із кнопками «NEXT» та «EXIT», яке інформує гравця про перемогу та дозволяє перейти на новий рівень	+

4.2. Usability Test

Usability Test – це спосіб тестування користувачами, яке проводиться з метою оцінити зручність, інтуїтивність, корисність і задоволеність від використання продукту для кінцевого користувача.

Учасникам тестування необхідно виконати такі задачі:

— почати гру;

— програти рівень

— повторити проходження рівня;

— виграти рівень;

— вийти з гри.

Респонденти відмітили простоту і зрозумілість інтерфейсу та ігрової механіки гри, не звичайне оформлення та середню важкість проходження даного ігрового застосунку. Всі задачі були виконані тестувальниками успішно та без проблем. Usability test пройдено успішно.

Висновок до розділу 4

Usability test пройдено успішно. Респонденти відмітили простоту і зрозумілість інтерфейсу та ігрової механіки гри, не звичайне оформлення та середню важкість проходження даного ігрового застосунку. Всі задачі, які були поставлені перед тестувальниками виконані успішно.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було досліджено особливості створення відеоігор з двовимірною графікою за допомогою ігрового рушія Unity та показано його можливості у розробці ігор.

Додаток Unity, що є професійним ігровим рушієм, використовується у створенні відеоігор для різних платформ. Найбільша превага даного рушія полягає у тому, що він є одним найдоступнішим інструментом для початківців.

Написання коду здійснювалося на мові програмування C#. Ця мова дозволяє легко увійти до розробки гри. C# залишається досить актуальною на сьогоднішній день . Ще однією особливістю даною мови програмування можна вважати її велику різноманітність синтаксичних конструкцій та можливості працювати з платформою Unity.

Під час створення графіки було застосовано програмне забезпечення Adobe Photoshop, адже саме це ПЗ є найпопулярнішим редактором для 2D графіки. У ньому підтримується велика кількість форматів, які використовуються в Unity.

Таким чином, можна зробити висновок, що використовуючи можливості вищеназваних програм, можна створювати ігрові додатки розважального характеру для різних платформ.

У ході виконання кваліфікаційної роботи бакалавра було реалізовано ігровий застосунок жанру «три в ряд» під мобільну систему Android на рушію Unity.

Для вирішення поставленої мети було виконано такі завдання:

— виконано аналіз актуальності теми;

— проведено аналіз аналогічних ігор;

— описано концепцію гри;

— зроблено огляд програмних середовищ для реалізації ігрового додатку;

— реалізовано ігровий додаток;

— проведено тестування ігрового застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Joseph Hocking — Unity in Action. Multiplatform game development in C# with Unity, 2016
2. Alan Thorn – Learn Unity For 2D Game Development, 2013
3. Alan Thorn – Mastering Unity Scripting, 2015
4. Sue Blackman – Unity for Absolute Beginners, 2014
5. Ron Penton – Beginning C# Game Programming, 2016
6. Enrico Bounanno – Functional Programming in C#: How to write better C# code, 2017
7. Офіційна документація Unity [Електронний ресурс] – https://docs.unity3d.com/Manual/index.html
8. Офіційний форум Unity по скриптах [Електронний ресурс] – https://forum.unity.com/forums/scripting.12/
9. Офіційний форум Unity по 2D [Електронний ресурс] – https://forum.unity.com/forums/2d.53/
10. Офіційна документація по C# [Електронний ресурс] – https://docs.microsoft.com/ru-ru/dotnet/csharp/
11. Good Times Regular from Good Times by Typodermic Fonts Inc.(License) [Електронний ресурс] – https://www.fontspring.com/fonts/typodermic/good-times/good-times-regular#license
12. Unity & Unity3d & Android & ADS. Match 3 Puzzle. Урок 2 - Tile, GM, Board [Електронний ресурс] – https://www.youtube.com/watch?v=mhi-VsOPl5g&t=791s&ab_channel=Udeck%E2%80%93%D0%A7%D0%B5%D1%80%D1%82%D0%BE%D0%B3%D0%B3%D0%B5%D0%B9%D0%BC%D0%B4%D0%B5%D0%B2%D0%B0
13. Colanderp Unity3D|Programming|Gaming [Електронний ресурс] – https://www.youtube.com/c/Colanderp
14. START MENU in Unity [Електронний ресурс] – https://www.youtube.com/watch?v=zc8ac_qUXQY&ab_channel=Brackeys

ДОДАТКИ

Скрипт GameManager:

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

public class GameManager : Singleton<GameManager>
{

	public int movesLeft = 30;
	public int scoreGoal = 10000;
	public ScreenFader screenFader;
	public Text levelNameText;
	public Text movesLeftText;

	Board m_board;

	bool m_isReadyToBegin = false;
	bool m_isGameOver = false;
	bool m_isWinner = false;
	bool m_isReadyToReload = false;

	public MessageWindow messageWindow;

	public Sprite loseIcon;
	public Sprite winIcon;
	public Sprite goalIcon;


	void Start()
	{
		m_board = GameObject.FindObjectOfType<Board>().GetComponent<Board>();

		Scene scene = SceneManager.GetActiveScene();

		if (levelNameText != null)
		{
			levelNameText.text = scene.name;
		}

		UpdateMoves();
		StartCoroutine("ExecuteGameLoop");
	}

	public void UpdateMoves()
	{
		if (movesLeftText != null)
		{
			movesLeftText.text = movesLeft.ToString();

		}
	}

	IEnumerator ExecuteGameLoop()
	{
		yield return StartCoroutine("StartGameRoutine");
		yield return StartCoroutine("PlayGameRoutine");
		yield return StartCoroutine("EndGameRoutine");
	}

	public void BeginGame()
```

```
	{
		m_isReadyToBegin = true;

	}

	IEnumerator StartGameRoutine()
	{
		if (messageWindow != null)
		{
			messageWindow.GetComponent<RectXformMover>().MoveOn();
			messageWindow.ShowMessage(goalIcon, "score goal\n" +
scoreGoal.ToString(), "start");
		}

		while (!m_isReadyToBegin)
		{
			yield return null;
		}


		if (screenFader != null)
		{
			screenFader.FadeOff();
		}

		yield return new WaitForSeconds(0.5f);

		if (m_board != null)
		{
			m_board.SetupBoard();
		}
	}

	IEnumerator PlayGameRoutine()
	{
		while (!m_isGameOver)
		{
			if (ScoreManager.Instance != null)
			{
				if (ScoreManager.Instance.CurrentScore >= scoreGoal)
				{
					m_isGameOver = true;
					m_isWinner = true;
				}
			}

			if (movesLeft == 0)
			{
				m_isGameOver = true;
				m_isWinner = false;
			}

			yield return null;
		}
	}

	IEnumerator EndGameRoutine()
	{
		m_isReadyToReload = false;



		if (m_isWinner)
		{
			if (messageWindow != null)
			{
```

```
                        messageWindow.GetComponent<RectXformMover>().MoveOn();
                        messageWindow.ShowMessage(winIcon, "YOU WIN!", "NEXT");
                }
            }
            else
            {
                if (messageWindow != null)
                {
                        messageWindow.GetComponent<RectXformMover>().MoveOn();
                        messageWindow.ShowMessage(loseIcon, "YOU LOSE!", "OK");
                }
            }

            yield return new WaitForSeconds(1f);

            if (screenFader != null)
            {
                screenFader.FadeOn();
            }

            while (!m_isReadyToReload)
            {
                yield return null;
            }

            SceneManager.LoadScene(SceneManager.GetActiveScene().name);

        }

        public void ReloadScene()
        {
            m_isReadyToReload = true;
        }
}
```

Скрипт GamePiece:

```
using UnityEngine;
using System.Collections;

public enum MatchValue
{
        Yellow,
        Blue,
        Magenta,
        Indigo,
        Green,
        Teal,
        Red,
        Cyan,
        Wild,
        None
}


public class GamePiece : MonoBehaviour
{

        public int xIndex;
        public int yIndex;
```

```
	Board board;

	bool isMoving = false;

	public InterpType interpolation = InterpType.SmootherStep;

	public enum InterpType
	{
		Linear,
		EaseOut,
		EaseIn,
		SmoothStep,
		SmootherStep
	};

	public MatchValue matchValue;

	public int scoreValue = 20;

	public AudioClip clearSound;


	void Start()
	{

	}

	void Update()
	{

	}

	public void Init(Board board)
	{
		this.board = board;
	}

	public void SetCoord(int x, int y)
	{
		xIndex = x;
		yIndex = y;
	}

	public void Move(int destX, int destY, float timeToMove)
	{

		if (!isMoving)
		{

			StartCoroutine(MoveRoutine(new Vector3(destX, destY, 0),
timeToMove));
		}
	}


	IEnumerator MoveRoutine(Vector3 destination, float timeToMove)
	{
		Vector3 startPosition = transform.position;

		bool reachedDestination = false;

		float elapsedTime = 0f;

		isMoving = true;

		while (!reachedDestination)
```

```
		{
			if (Vector3.Distance(transform.position, destination) < 0.01f)
			{

				reachedDestination = true;

				if (board != null)
				{
					board.PlaceGamePiece(this, (int)destination.x,
(int)destination.y);

				}

				break;
			}
			elapsedTime += Time.deltaTime;

			float t = Mathf.Clamp(elapsedTime / timeToMove, 0f, 1f);

			switch (interpolation)
			{
				case InterpType.Linear:
					break;
				case InterpType.EaseOut:
					t = Mathf.Sin(t * Mathf.PI * 0.5f);
					break;
				case InterpType.EaseIn:
					t = 1 - Mathf.Cos(t * Mathf.PI * 0.5f);
					break;
				case InterpType.SmoothStep:
					t = t * t * (3 - 2 * t);
					break;
				case InterpType.SmootherStep:
					t = t * t * t * (t * (t * 6 - 15) + 10);
					break;
			}

			transform.position = Vector3.Lerp(startPosition, destination, t);

			yield return null;
		}

		isMoving = false;


	}

	public void ChangeColor(GamePiece pieceToMatch)
	{
		SpriteRenderer rendererToChange = GetComponent<SpriteRenderer>();

		if (pieceToMatch != null)
		{
			SpriteRenderer rendererToMatch =
pieceToMatch.GetComponent<SpriteRenderer>();

			if (rendererToMatch != null && rendererToChange != null)
			{
				rendererToChange.color = rendererToMatch.color;
			}

			matchValue = pieceToMatch.matchValue;
		}

	}
```

```
	public void ScorePoints(int multiplier = 1, int bonus = 0)
	{
		if (ScoreManager.Instance != null)
		{
			ScoreManager.Instance.AddScore(scoreValue * multiplier + bonus);
		}

		if (SoundManager.Instance != null)
		{
			SoundManager.Instance.PlayClipAtPoint(clearSound, Vector3.zero,
SoundManager.Instance.fxVolume);
		}
	}

}
```

Скрипт SoundManager:

```
using UnityEngine;
using System.Collections;

public class SoundManager : Singleton<SoundManager>
{
    public AudioClip[] musicClips;
    public AudioClip[] winClips;
    public AudioClip[] loseClips;
    public AudioClip[] bonusClips;

    [Range(0, 1)]
    public float musicVolume = 0.5f;

    [Range(0, 1)]
    public float fxVolume = 1.0f;

    public float lowPitch = 0.95f;
    public float highPitch = 1.05f;

    void Start()
    {
        PlayRandomMusic();
    }

    public AudioSource PlayClipAtPoint(AudioClip clip, Vector3 position, float
volume = 1f)
    {
        if (clip != null)
        {
            GameObject go = new GameObject("SoundFX" + clip.name);
            go.transform.position = position;

            AudioSource source = go.AddComponent<AudioSource>();
            source.clip = clip;

            float randomPitch = Random.Range(lowPitch, highPitch);
            source.pitch = randomPitch;

            source.volume = volume;

            source.Play();
            Destroy(go, clip.length);
            return source;
```

```
        }

        return null;
    }

    public AudioSource PlayRandom(AudioClip[] clips, Vector3 position, float volume
= 1f)
    {
        if (clips != null)
        {
            if (clips.Length != 0)
            {
                int randomIndex = Random.Range(0, clips.Length);

                if (clips[randomIndex] != null)
                {
                    AudioSource source = PlayClipAtPoint(clips[randomIndex],
position, volume);
                    return source;
                }
            }
        }
        return null;
    }

    public void PlayRandomMusic()
    {
        PlayRandom(musicClips, Vector3.zero, musicVolume);

    }

    public void PlayWinSound()
    {
        PlayRandom(winClips, Vector3.zero, fxVolume);
    }

    public void PlayLoseSound()
    {
        PlayRandom(loseClips, Vector3.zero, fxVolume * 0.5f);
    }

    public void PlayBonusSound()
    {
        PlayRandom(bonusClips, Vector3.zero, fxVolume);
    }

}
```

Скрипт SreenFader:

```
using System.Collections;

using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(MaskableGraphic))]
public class ScreenFader : MonoBehaviour
{
    public float solidAlpha = 1f;
    public float clearAlpha = 0f;
```

```
    public float delay = 0f;
    public float timeToFade = 1f;

    private MaskableGraphic graphic;

    private void Start() => graphic = GetComponent<MaskableGraphic>();

    private IEnumerator FadeRoutine(float alpha)
    {
        yield return new WaitForSeconds(delay);

        graphic.CrossFadeAlpha(alpha, timeToFade, true);
    }

    public void FadeOn() => StartCoroutine(FadeRoutine(solidAlpha));

    public void FadeOff() => StartCoroutine(FadeRoutine(clearAlpha));
}
```