МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Ігровий додаток-тренажер пам'яті»

Здобувача (ки) групи ІТ-91 Денисенка Микити Сергійовича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Микита ДЕНИСЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доц., к.т.н., доц. Світлана ВАЩЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

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

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«_____» _____2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ
Денисенку Микиті Сергійовичу

1 Тема роботи Ігровий додаток-тренажер *пам'яті*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « *29* » 05 2023 р. №0588-VI

2 Строк подання студентом роботи « 08 » 06 2023 р.

3 Вхідні дані до роботи Технічне завдання на розробку, системні вимоги, правила гри

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, аналіз предметної області, визначення актуальності дослідження, аналіз існуючих продуктів-аналогів, постановка задачі, моделювання додатку, структурно-функціональна модель, моделювання варіантів використання, розробка ігрового додатку, програмна реалізація, демонстрація роботи, висновки, список використаних джерел, додатки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Актуальність, Постановка задачі, Аналіз продуктів-аналогів, порівняння продуктів аналогів, Вимоги до реалізації, Правила гри, Зовнішня структура додатку, Моделювання роботи додатку, Діаграма варіантів використання, Засоби реалізації, Демонстрація додатку, Висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання____01.01.2023_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Визначення актуальності досліджень	05.01.2023	Виконано
2	Аналіз продуктів-аналогів	11.01.2023	Виконано
3	Проектування	07.02.2023	Виконано
4	Розробка прототипу	20.03.2023	Виконано
5	Розробка інтерфейсу та дизайну	31.03.2023	Виконано
6	Тестування додатку	30.05.2023	Виконано
7	Реліз	12.06.2023	Виконано

Студент _____ Микита ДЕНИСЕНКО
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Розробка мобільного ігрового додатку-тренажеру пам’яті».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 20 найменувань, додатків. Загальний обсяг роботи − 75 сторінок, у тому числі 36 сторінок основного тексту, 2 сторінки списку використаних джерел, 30 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці мобільного ігрового додатку для покращення пам’яті користувачів. В роботі проведено аналіз ринку мобільних відеоігор, популярності та поширення мобільних ігор та використання смартфонів, як ігрової платформи. Досліджено вплив ігор на розвиток людини, як і використання ігрових технологій у навчанні. У роботі виконано моделювання та розробка ігрового додатку. Результатом проведеної роботи є публікація гри у PlayMarket Практичне значення роботи полягає у створені і поширенню розвиваючої мобільної гри. Представлені альтернативного способу використання ігрових технологій. Та надання користувачам можливість розважатись, отримуючи користь.

Ключові слова: мобільна гра, розвиваюча гра, ігровий додаток, Unity, PlayMarket, тренування пам’яті.

ЗМІСТ

ВСТУП 5

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Визначення актуальності дослідження 7

1.2 Аналіз існуючих продуктів-аналогів 9

1.3 Постановка задачі 17

2 МОДЕЛЮВАННЯ ДОДАТКУ 19

2.1 Структурно-функціональна модель 19

2.2 Моделювання варіантів використання 21

3 РОЗРОБКА ІГРОВОГО ДОДАТКУ 23

3.1 Вибір інструментів для розробки**Ошибка! Закладка не определена.**

3.2 Програмна реалізація 23

3.3 Демонстрація та тестування 35

ВИСНОВКИ 41

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 42

ДОДАТОК А 45

ДОДАТОК Б 51

ДОДАТОК В 63

ВСТУП

На сьогоднішній день ігрова індустрія є однією з найпопулярніших сфер розваг. Ринок ігрових додатків вже став більший за ринок кіноіндустрії, із значною різницею по річному прибутку. Після ковідних обмежень 2020 року, ігри набули ще більшого поширення в результаті чого в них грає вже більше ніж 3 млрд людей [1].

Ігрова індустрія стала настільки поширеною, що проводяться світові турніри, створюються речі та сувеніри із ігровою символікою та логотипами кіберспортивних команд, а в багатьох країнах кіберспорт навіть було прийнято, як офіційний вид спорту. Завдяки цьому у відеоігор створилась дуже велика база шанувальників. Ігри стають сильним маркетинговим інструментом та зацікавлюють багато молодих людей.

З іншого боку у молодого покоління є тенденція погіршення концентрації та уваги [2]. Саме через розвиток маркетингових прийомів, які перенасичують органи сприйняття, використовуючи короткі відеоролики або токсичну рекламу, стає все складнішим зацікавити молодих людей книжками або науковою літературою.

Саме тому ідеєю цього проекту стало поєднання інструментів ігрової індустрії із процесом покращення уваги та концентрації на деталях учнівської молоді. Використання ігрових додатків з метою більшого зацікавлення людей, подаючи корисні вправи під видом гри. Спроба довести та показати, що ігри можна використовувати з розумом та подання ідеї для реформування сучасної системи освіти.

Основною метою цієї роботи є розробка ігрового мобільного додатку, призначеного для тренування пам’яті.

Для досягнення поставленої мети треба вирішення такі задачі:

- детально дослідити предметну область;
- провести аналіз існуючі ігрові додатки для тренування пам’яті;

- розробити чіткі функціональні вимоги проекту, технічне завдання на розробку гри, розподілити види робіт за календарним планом ;
- розробити прототип додатку;
- реалізувати графічне та звукове наповнення;
- провести тестування та інтегрувати рекламу;
- виконати розробку додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення актуальності дослідження

Відеоігри [3] вперше зародились ще у 1947 році [4], а почали набувати популярності вже в 1960-1970 роки. Як показав час, ця індустрія не тільки виявилась довголітньою, а ще й популярною і можливо тільки здогадуватись який ще потенціал вона має [5]. Із часів коли з’явилась перша гра, минуло багато часу, змінилось покоління. Розвиток та еволюція це невпинний механізм, рушій людства, що змушує нас розвиватись та адаптуватись. Бажання покращити власні життя, зробити їх безпечнішими властиво кожній людині. Саме завдяки цьому невпинним є і розвиток технологій, в тому числі і ігрових. Змінюються інструменти, з’являються нові технології, вигадуються нові способи їх використання.

Нажаль з розвитком технологій з’являються і супротивники тих чи інших винаходів. За останні два десятиліття актуальності набули дискусії щодо користі чи шкоди відеоігор та їх вплив на людину. Є думка, що ігри завдають шкоди як психічному так і фізичному здоров’ю людини, проте останні дослідження [6] з впливу відеоігор на людину доводять її помилковість [7]. Відеоігри безумовно можуть впливати на людину, вони можуть покращувати дрібну моторику, покращувати вміння концентруватися, поліпшувати уважність та навіть розвивати тактичне мислення [8]. Навіть батьки грають з дітьми, дітям вигадують ігрові вправи, наприклад винаходження відповідностей, чи пошук маршруту із лабіринту. Це ті самі ігри, представленні в іншій формі. Сучасні дисплеї та екрани виготовляються таким чином, що вже не наносять шкоду зору. Будь що може викликати проблеми при зловживанні або недотримані рекомендацій із використання, але це не перекриває плюси того чи іншого винаходу.

Ігри більш за все призначені для відпочинку, коротання часу та розваг. Саме через це багато людей відносяться до них, як до забавок, не сприймаючі як повноцінне проводження часу, і це право кожного, проте неможливо заперечувати що вони набули неймовірної популярності та широко розповсюдились, особливо серед молодого покоління. Вони зацікавлюють та привертають увагу [4]. Цей факт має бути взятий за основу, для подальшого аналізу та ідеї його використання.

Ігрові додатки можуть приносити користь і мета цього проекту показати альтернативне бачення їх використання [9]. Використання ігрових додатків для поліпшення різних навичок, чи взагалі, для подачі навчального матеріалу [10]. Основне призначення додатку буде, як і у більшості ігор, розважити та скоротати час користувача, проте, саме результат скороченого часу, спрямувати на створення користі для користувача.

Найдоступнішими ігровими платформами на сьогодні є смартфони, хоч це і багатофункціональні пристрої, в основі яких лежить підтримання зв’язку між людьми, це ще й девайс, який є у більшості людей і який максимально адаптований для зручного отримання інформації [11].

Саме тому програмний застосунок буде розроблюватись під смартфони, з операційною системою Android. Операційна система була обрана із міркувань популярності, доступності та зручності. Мобільна гра буде спрямована на тренування пам’яті та уважності користувача. Застосунок буде представляти собою гру, основану на двовимірній графіці та реалізовувати розвиваючі вправи, представлені у вигляді різних режимів. Розробка мобільного додатку буде виконуватись із використанням ігрового рушія Unity [12] у зв’язці із IDE Visual Studio, з використанням об’єктно орієнтованої мови програмування C#.

1.2 Аналіз існуючих продуктів-аналогів

Аналіз продуктів-аналогів невід'ємний етап розробки будь якого додатку. Лише в умовах конкуренції можна очікувати на розвиток тієї чи іншої сфери діяльності[13]. Конкуренція спонукає робити краще, зручніше, вигадувати щось нове, постійно адаптуватись до змін. Для того щоб програмний продукт вийшов якісним, необхідно проаналізувати ринок на наявність подібних проектів, визначити їх слабкі та сильні сторони, прочитати зауваження користувачів, створити порівняльну таблицю, та підбити підсумки. Такий аналіз дає змогу уникнути розповсюджених проблем, навчаючись на чужих помилках та розширити бачення предметної області.

Розвиваючі ігри або ігри жанру "Головоломка" були чи не найпершими представниками цієї індустрії. Хоч на сьогоднішній день вони програють в популярності більш казуальним та аркадним іграм, існує досить велика кількість різноманітних проектів та цікавих продуктів, аналіз яких ми і будемо проводити.

Так як ціль роботи - розробка мобільного ігрового додатку на Android, будемо аналізувати саме ринок мобільних ігор, представлених у PlayMarket.

Першим претендентом на розгляд буде ігровий додаток "Remembery" від студії "Dreamy Dingo" [14] виконаний в дуже яскравому стилі.

Рисунок 1.1 – Гра «Remembery»

Суть гри полягає в проходженні рівнів, складність яких підвищується з кожним пройденим. Рівень представляє собою поле із певною кількістю закритих карток, гравець має відкривати картки парами, та запам’ятовувати що він побачив на обраній стороні картки. Задача віднайти всі однакові пари карток.

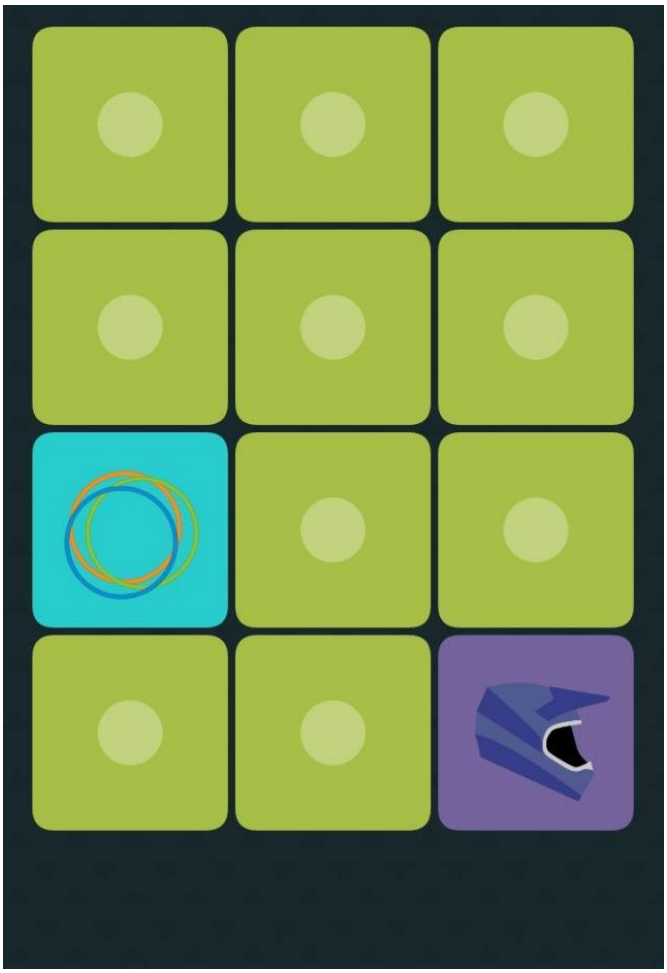


Рисунок 1.2 – Приклад завдання гри «Remembery»

З плюсів можна виділити досить яскраве графічне забезпечення та приємний звуковий супровід. Малюнки карток дуже різноманітні, виконані за окремими тематиками для кожного рівня (їжа, музикальні інструменти, геометричні фігури, тощо). Рівень складності зростає із збільшенням кількості карток та введення обмежень на кількість спроб.

Існує ще один режим гри, його сутність полягає в тому, що гравцю на декілька секунд показують положення потрібних карток, після чого картки закриваються і гравець має обрати ті, що були показані на початку.

З мінусів можна виділити не велике різноманіття режимів гри, та не дуже зручна модель відстеженню та аналізу своїх успіхів, покращення результату.

Наступним додатком на розгляд було взято проект під назвою “Elevate: Brain Training” від розробників Elevate Labs [15].

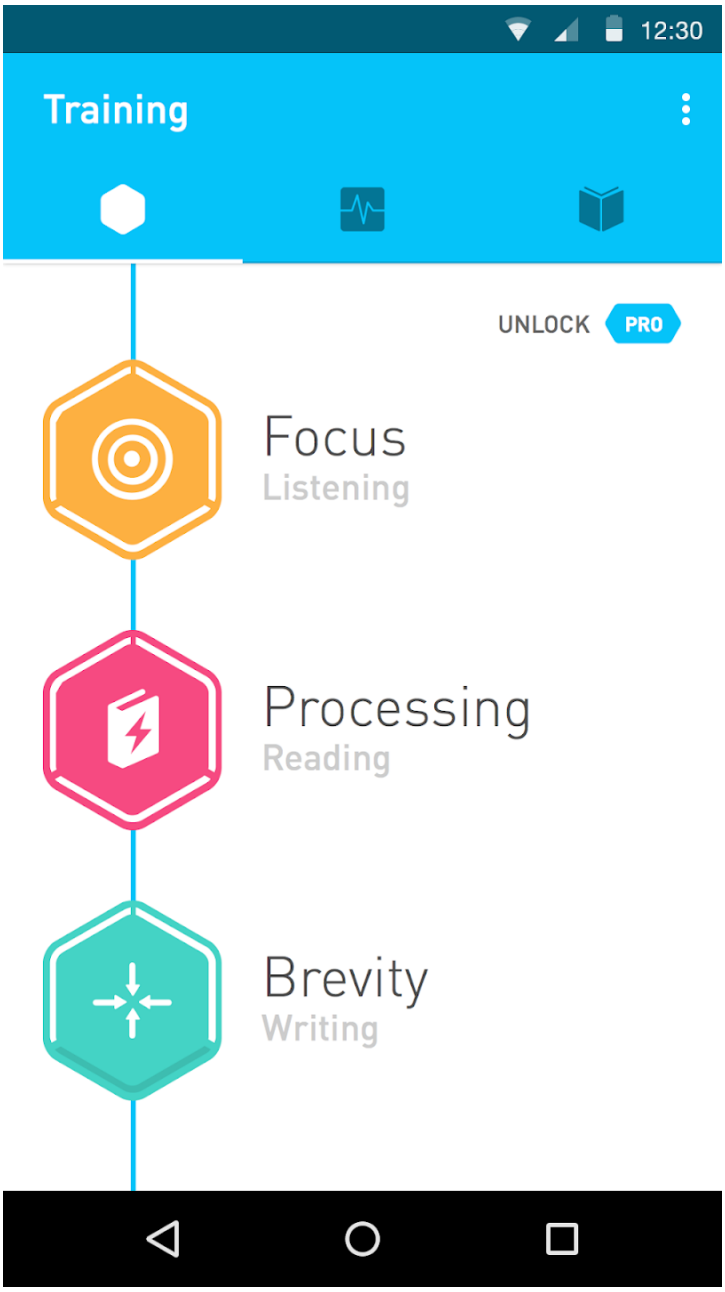


Рисунок 1.3 – Вікно гри «Elevate: Brain Training»

Ця гра виконана в більш мінімалістичному та менш яскравому стилі. Орієнтована на більш доросле, підліткове – зріле покоління. Звуковий супровід доповнює ігровий процес, інтуїтивно зрозумілими звуками вірної чи

помилкової відповіді. Даний застосунок важко назвати саме грою, скоріш саме тренажер для мозку.

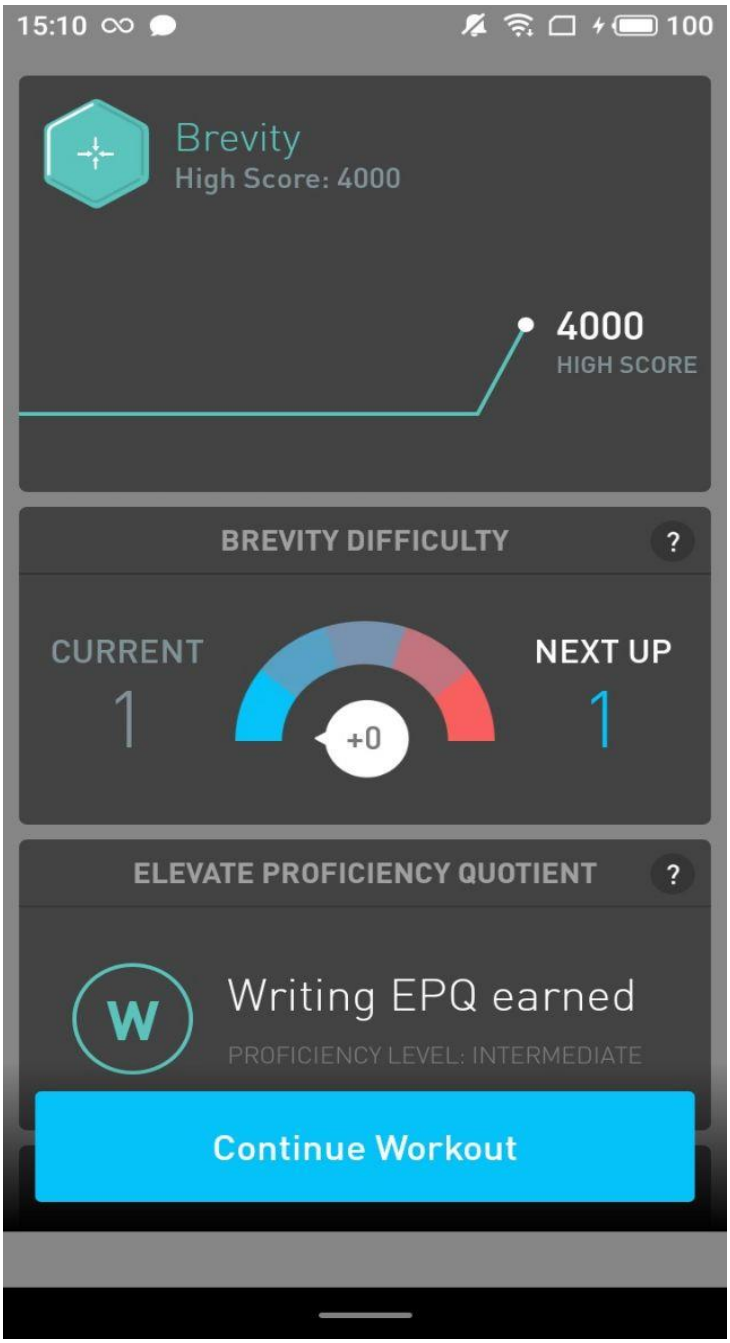


Рисунок 1.4 – Вікно гри «Elevate: Brain Training»

Процес гри представляє собою проходження декількох етапів кожного рівня, одному етапу відповідає одна вправа. Вправи змінюються на кожен

етап, проте можуть повторюються на наступному рівні, з певним ускладненням. Всього вправ достатньо велика кількість.

Так як суть гри полягає в різноплановому розвитку мозку, не всі режими гри спрямовані на покращення пам’яті.

Також слід зазначити, що на відміну від інших проектів, цей має унікальну систему монетизації. Вона реалізована за моделлю місячної підписки на застосунок і коштує достатньо багато, як для ринку України. Тому і орієнтована на зарубіжні.

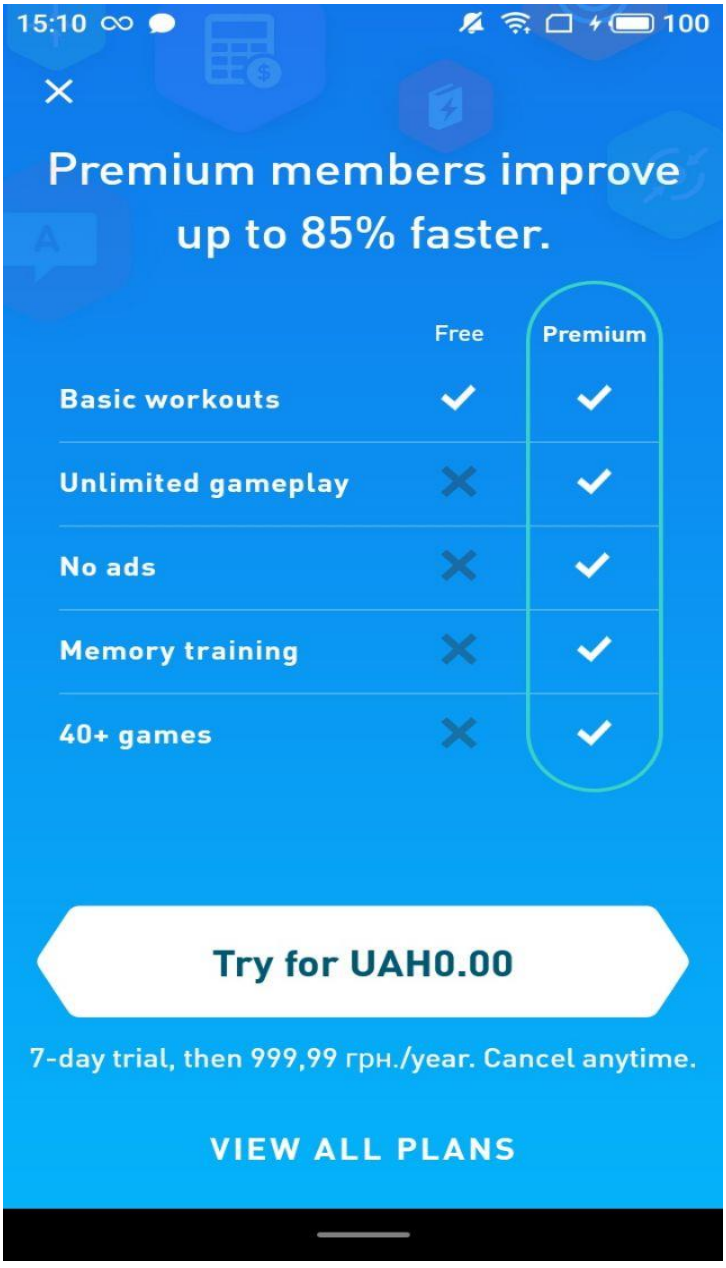


Рисунок 1.5 – Вікно гри «Elevate: Brain Training»

Ще одним додатком на розгляд було взято гру “Lights: A memory game” розроблену Lee Phillips [16].

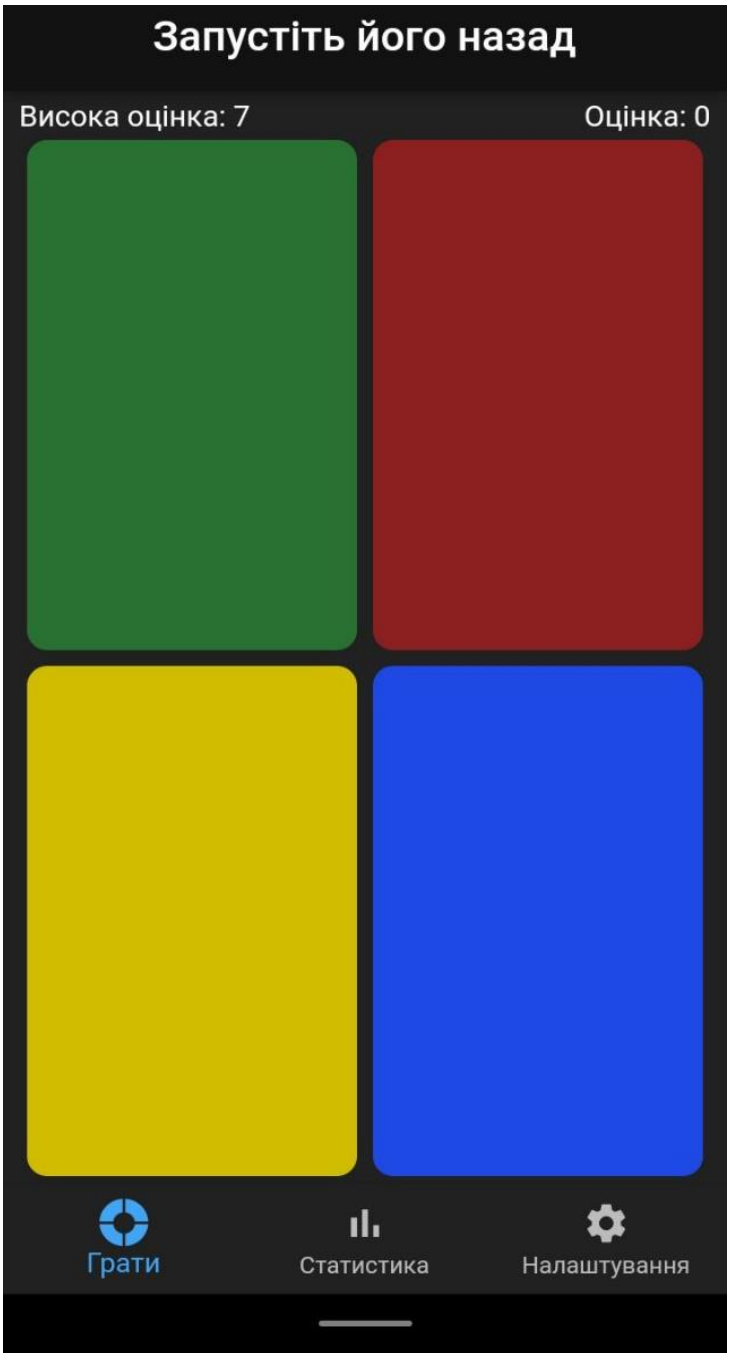


Рисунок 1.6 – Вікно гри «Lights: A memory game»

Гра виконана в дуже простому стилі та складається з однієї сцени. Гравцю представлено 4 кнопки різних кольорів, гра натискає кнопки в певній послідовності, гравець же має повторити цю послідовність. За кожною

кнопкою закріплено певний звук, звуки відрізняються тональністю, тому при натисканні послідовності, виникає певна мелодія, що відрізняється кожного разу. При успішному повторенні послідовності, вона повторюється із додаванням нових натискань до неї, що призводить до зростання складності. Задумка із звуковим супроводом, мелодією що виникає в залежності від послідовності, дуже цікава.

Із вагомих мінусів проекту можна відмітити:

- відсутність пояснень до гри, перші хвилини просто не розумієш що потрібно робити;
- відсутність різноманіття – один режим гри;
- дуже простий дизайн;
- погано прорахована прогресія складності, зазвичай пройти більше 7-8 зростань послідовності стає неможливим.

Протестувавши ці продукти аналоги та виконавши детальний їх аналіз, як і аналіз відгуків користувачів можна підбити підсумки за допомогою порівняльної таблиці, в якій будуть представлені основні виділені показники ігор, що порівнюються, та зазначена відповідність розглянутих проектів ним. Данні наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик проектів-аналогів

Характеристика	**Ігровий додаток**		
	“**Remembery**”	**“ Elevate: Brain Training ”**	“**Lights: A memory game**”
Яскравий дизайн	+	+	-
Інтуїтивно зрозумілий інтерфейс	+	+	-
Різноманітність вправ	-	+	-
Якісний звуковий супровід	+	+	+

Продовження табл. 1.1.

Збалансована прогресія складності	+	+	-
Відстеження результатів	+	+	+
Рейтинг серед гравців	+	+	-

Як видно з таблиці, розглянуті варіації ігор мають як свої переваги, так і недоліки. Тому при виконанні власної розробки будуть враховані всі позитивні сторони додатків-аналогів та реалізовані ті аспекти, які в цих аналогах є слабким елементом.

1.3 Постановка задачі

Мета цієї роботи – розробити власну розвиваючу гру “RememberIT”, спрямовану на розвиток пам’яті, концентрації та уваги користувача. Гра має бути розроблена під смартфони з операційною системою Android та бути опублікованою на PlayMarket.

Задля досягнення мети необхідно:

- виконати моделювання процесу роботи майбутньої гри;
- розробити якісну архітектуру проекту;
- розробити зручний інтерфейс та приваблюючий дизайн;
- провести тестування продукту;
- опублікувати гру.

Цільова аудиторія, ігрові режими та суть завдань описані у додатку А.

Програмний продукт має бути виконано згідно з технічним завданням на розробку (додаток А) та у відповідності до виконано планування робіт (додаток Б).

Програмна реалізація продукту здійснюється із використанням ігрового рушія Unity [17], в поєднанні із IDE Visual Studio, та використанням мови програмування C#. Звуковий та графічний супровід можуть бути як зроблені самостійно, так і запозичені, при наявності актуальної ліцензії на комерційне використання.

Проаналізувавши технічне завдання на розробку ігрового додатку, було прийнято рішення обрати наступні програмні продукти для реалізації цього проекту:

Ігровий рушій Unity – основний інструмент для розробки кроссплатформених ігор. Сучасний та актуальний рушій, використовує об’єктно орієнтовану мову програмування C#.

IDE Visual Studio – Одна з найпопулярніших IDE для використання із Unity. Має змогу виконувати дебаг коду напряму через ігровий рушій.

Figma – платформа для розробки дизайну застосунку. Сучасна та зручна платформа, із безкоштовним доступом.

Google Play Console – Консоль розробника ігор, необхідна для завантаження гри на PlayMarket.

2 МОДЕЛЮВАННЯ ДОДАТКУ

2.1 Структурно-функціональна модель

Перш за все, за для коректного моделювання слід розробити контекстну діаграму IDEF0. Це методологія функціонального моделювання, графічне представлення нотацій. Вона була розроблена для представлення та опису бізнес-процесів, проте зараз вона використовується також і при роботі із інформаційними системами. [19]

Графічне представлення реалізується за допомогою блоків(функцій) об’єднаними в одну систему, за допомогою вхідних та вихідних даних, відображених у вигляді стрілок, та контролюєму, за допомогою об’єктів та механізмів управління.

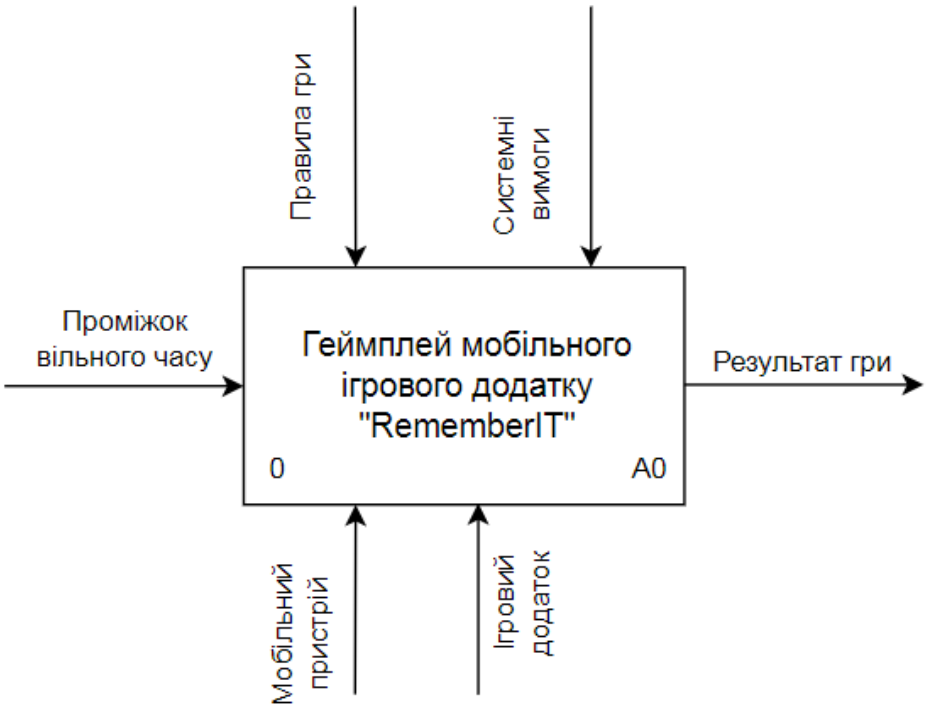


Рисунок 2.1 – Контекстна діаграма основного процесу

Основна функція ігрового додатку це надавати можливість грати, тож вона визначена як ігровий процес (геймплей). Це можливо за наявності у користувача вільного часу, тому вхідні данні визначені наступним чином. Об’єктами управління в нашому випадку є правила гри, системні вимоги до застосунку, мобільний пристрій та власне сам ігровий додаток. Вихідні данні представлені у вигляді результатів (набраного рахунку).

Наступним етапом було виконання декомпозиції контекстної діаграми. Основний процес розбивається на підпроцеси, та встановлюються залежності між ними. В результаті отримано наступну діаграму, показану на рис.2.2.

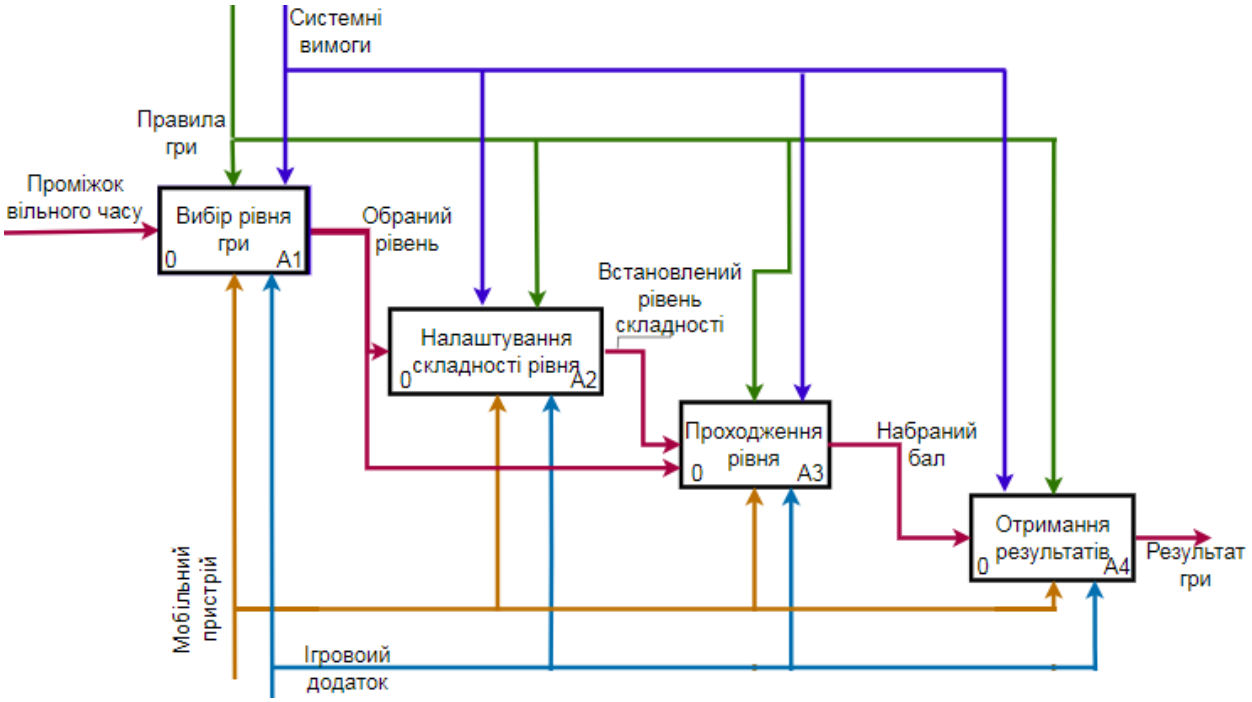


Рисунок 2.2 – Діаграма декомпозиції першого рівня

Гра починається із потрапляння в головне меню, де користувач може обрати рівень, або залишити рівень за замовчуванням, далі в залежності від обраного рівня користувач може обрати рівень складності, чи залишити його таким, що визначено за замовчуванням. Після вибору рівня та складності які впливають на ігровий процес, користувач проходить рівень. По завершенню гри результат користувача фіксується та він отримує результати гри.

2.2 Моделювання варіантів використання

Дуже важливим етапом є моделювання варіантів використання додатку. Для цього створюємо UML діаграму, що є графічним представленням варіантів використання майбутнього додатку. Діаграма (рис.2.3) складається із акторів та варіантів використання, в межах блоку (системи) - в нашому випадку це ігровий додаток “RememberIT”, актором же виступає користувач - гравець.

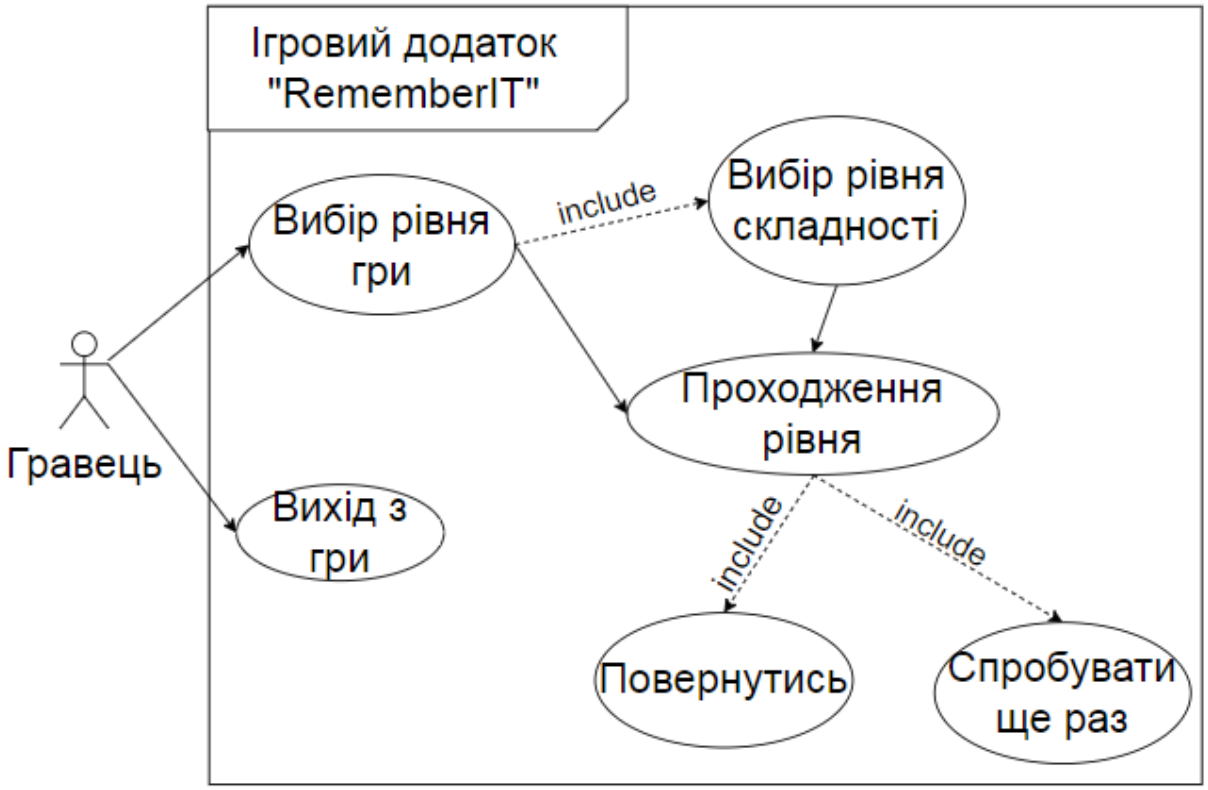


Рисунок 2.3 – Діаграма варіантів використання

Користувач має змогу вийти з гри або приступити до гри, для чого він обирає рівень, обравши рівень, користувач може вибрати рівень складності, або залишити його за замовчуванням, після чого користувач переходить до

проходження рівня. По завершенню у користувача є дві опції, спробувати ще (перезапуск рівня), або повернутись (вийти в головне меню).

3 РОЗРОБКА ІГРОВОГО ДОДАТКУ

3.1 Програмна реалізація

Перший етап розробки полягає у плануванні та створенні приблизного макету застосунка. Для цього на платформі DrawIO було реалізовано приблизний план – макет мобільного ігрового додатку “RememberIT”.

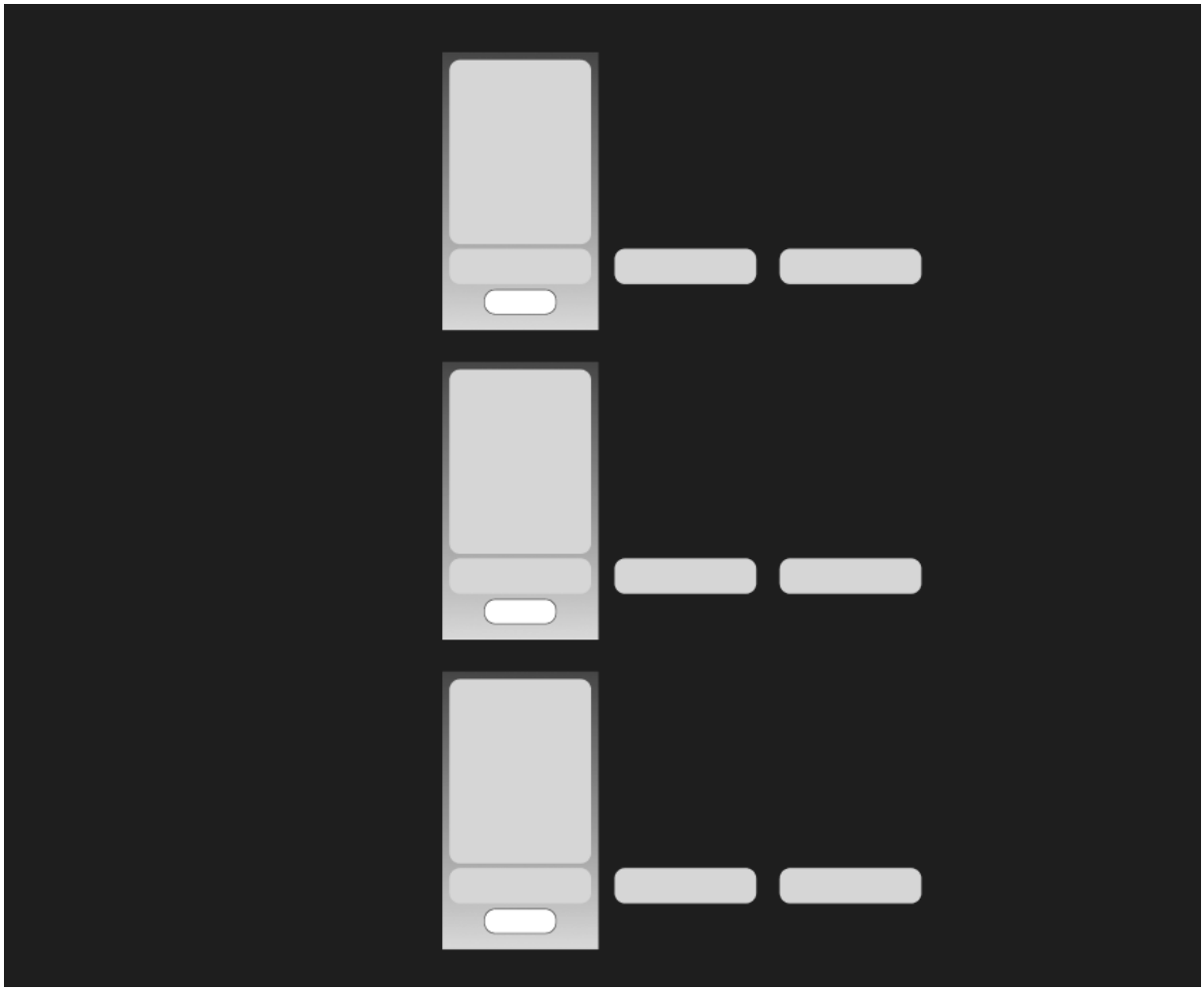


Рисунок 3.1 – Макет гри «RememberIT»

Наступним етапом було створено проект в Unity. Проект створено за звичайним 2D темплейтом, так як використання 3D графіки не планується в даному проекті.

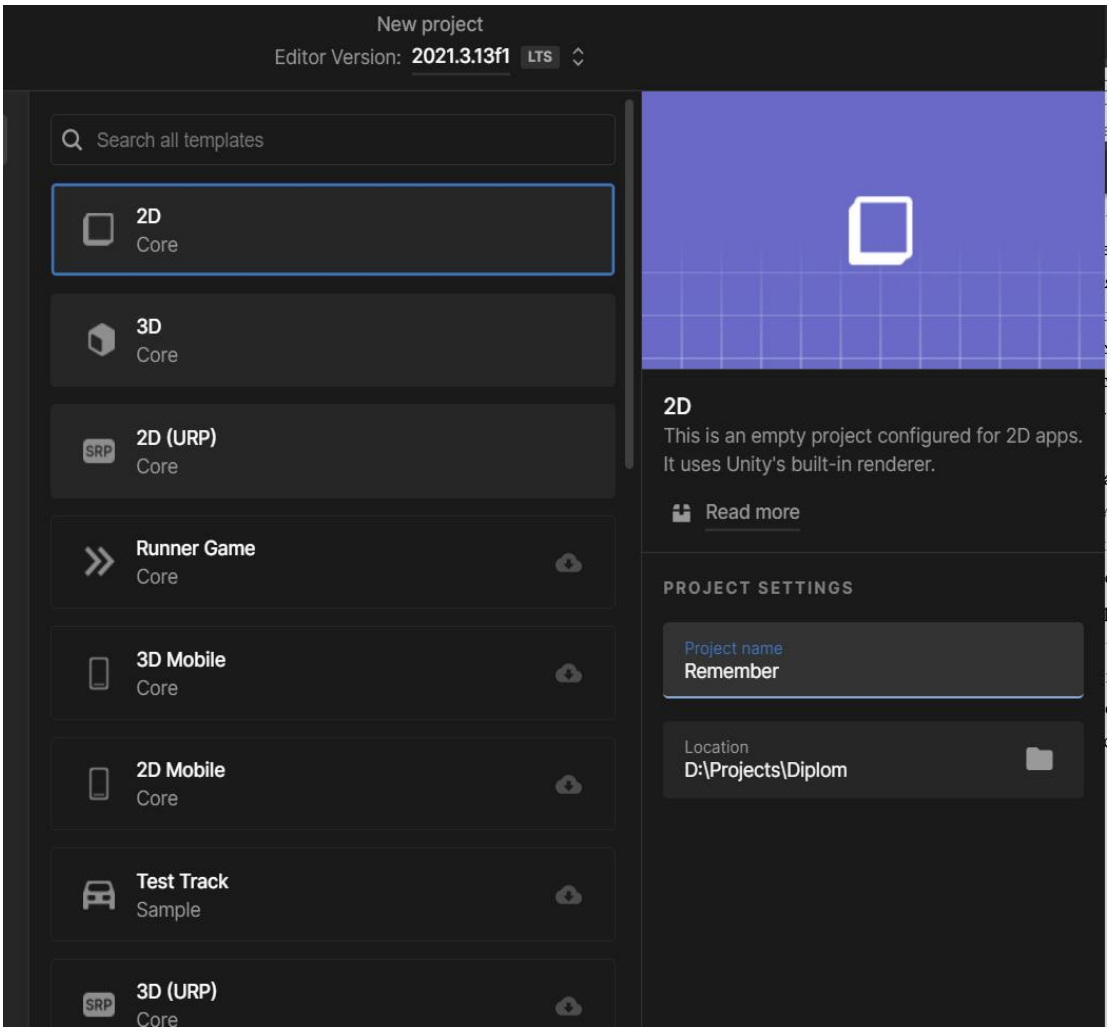


Рисунок 3.2 – Створення проекту «RememberIT»

Наступним кроком заходимо в Project – Build Settings та одразу змінюємо target platform на Android. Встановлюємо розмір вікна гри у відношення 16:9 verrtical.

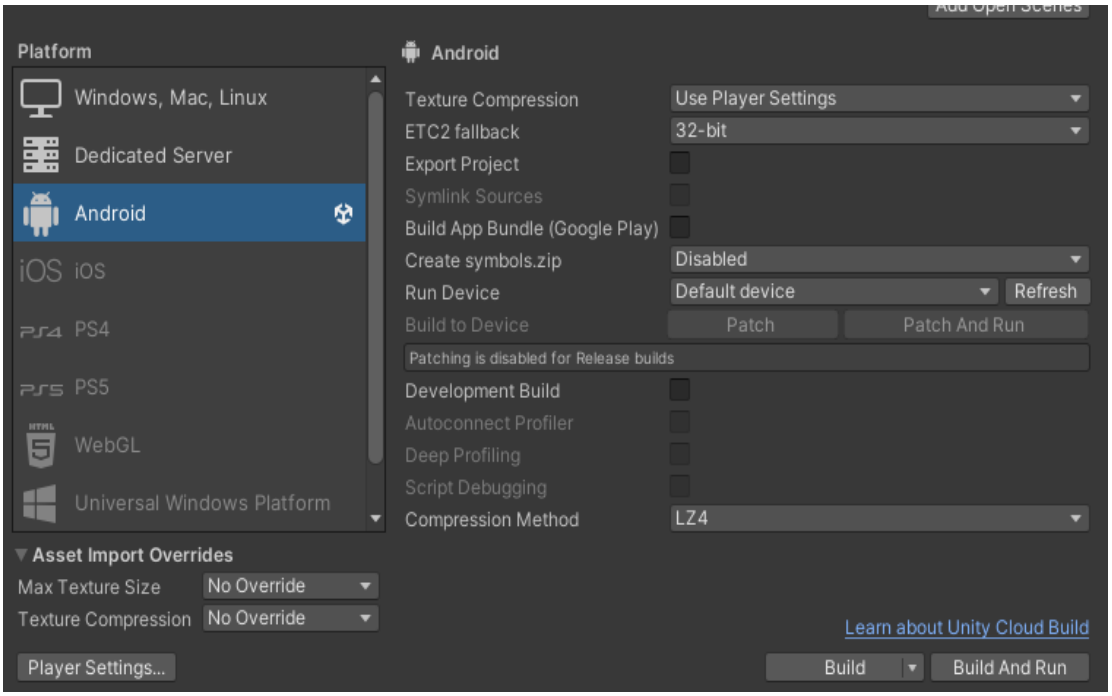


Рисунок 3.3 – Зміна платформи гри

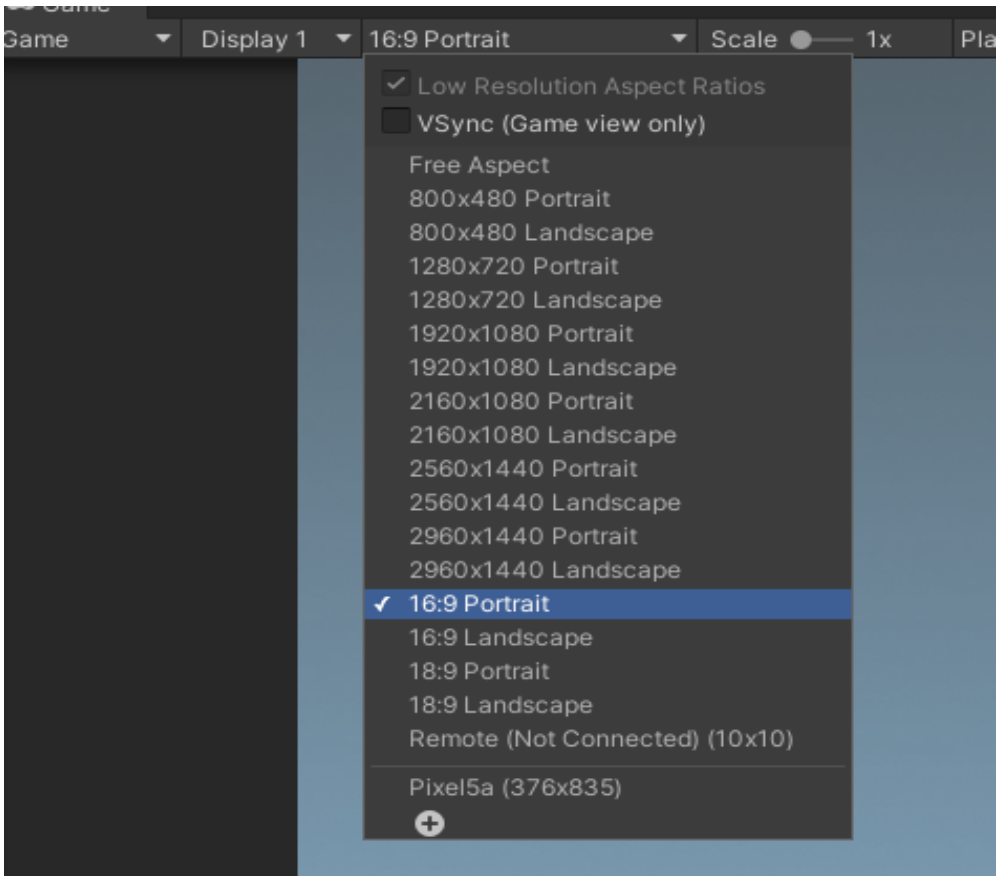


Рисунок 3.4 – Зміна орієнтації макету грального поля

Після виконання цих дій переходимо до створення прототипу гри. Створюємо необхідні сцени (рівні). На кожному рівні будуємо ігровий процес, створенням ігрових об’єктів та написання скриптів для надання їм логіки та поведінки.

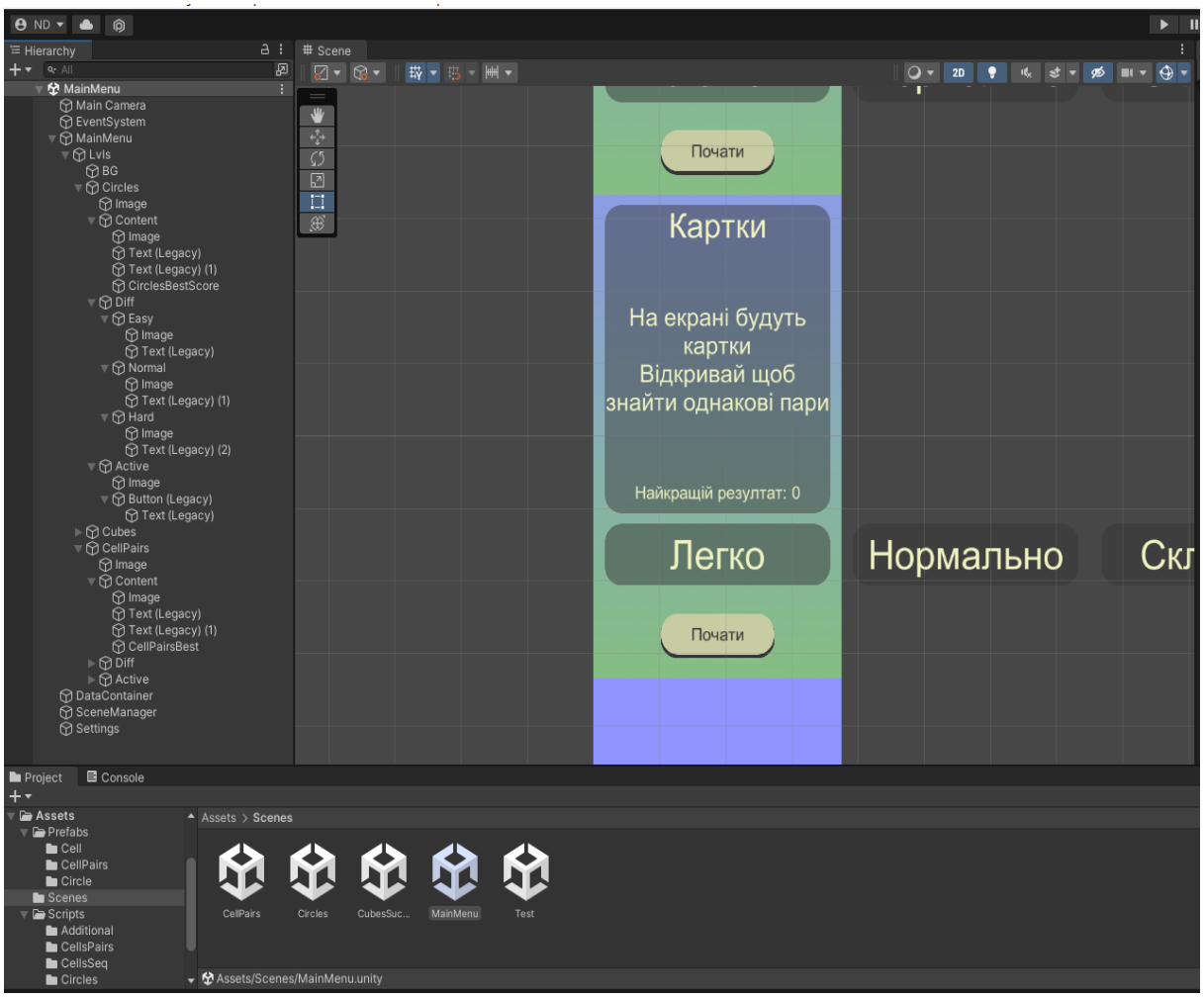


Рисунок 3.5 – Приклад наповнення сцени із головним меню

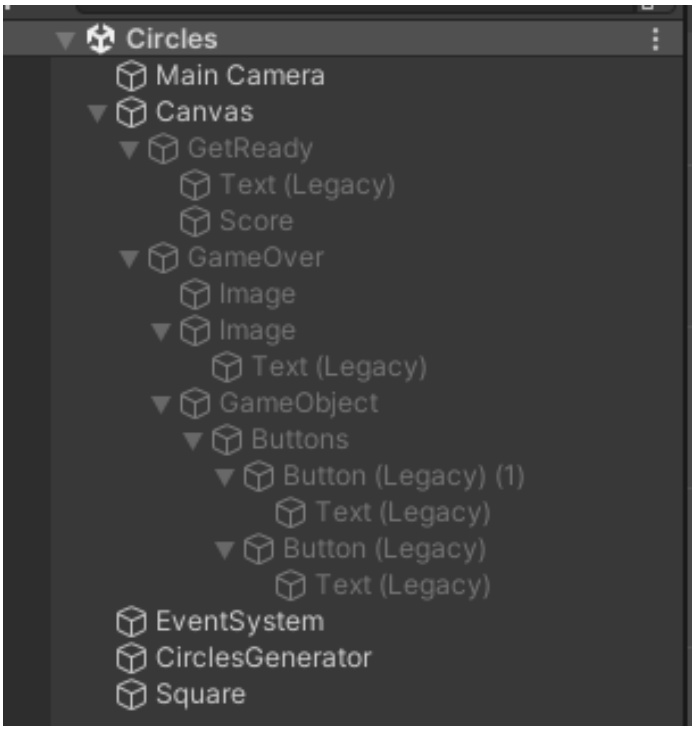


Рисунок 3.6 – Наповнення сцени «Circles»

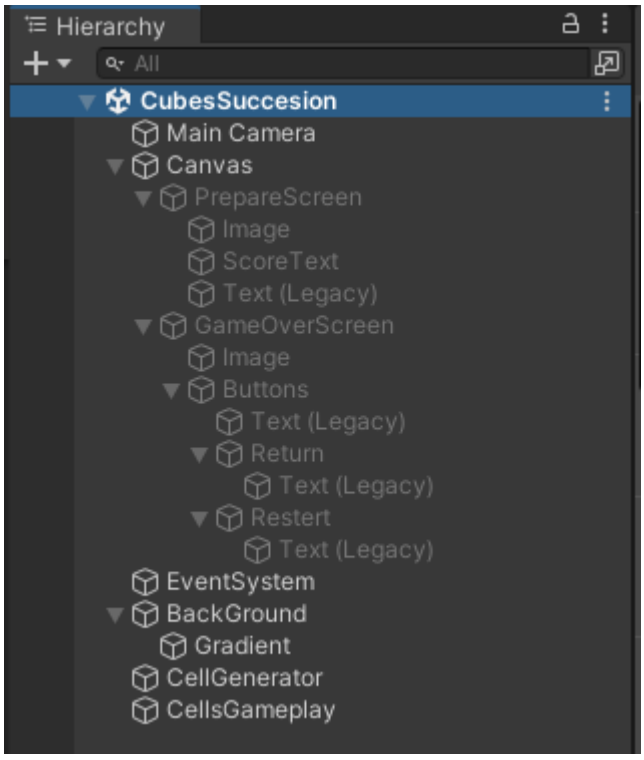


Рисунок 3.7 – Наповнення сцени «CubesSuccesion»

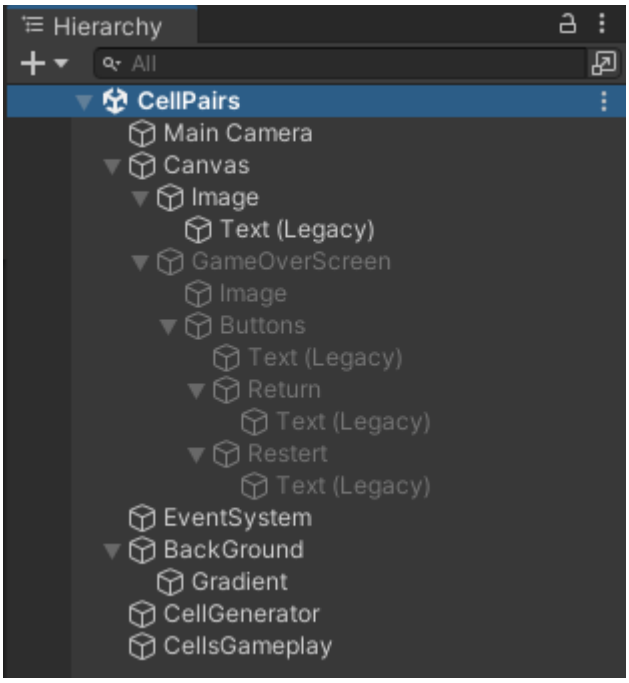


Рисунок 3.8 – Наповнення сцени «CellPairs»

Усі створені об’єкти налаштовуються за допомогою вікна компонентів, куди можна назначати як вбудовані компоненти такі як Transform(служить для управління положенням об’єкта) Sprite Renderer(для налаштування умов рендеру 2D ігрового об’єкта) Collider(для надання фізичних границь об’єкта) так і власні скрипти. Приклад на рис. 3.9.

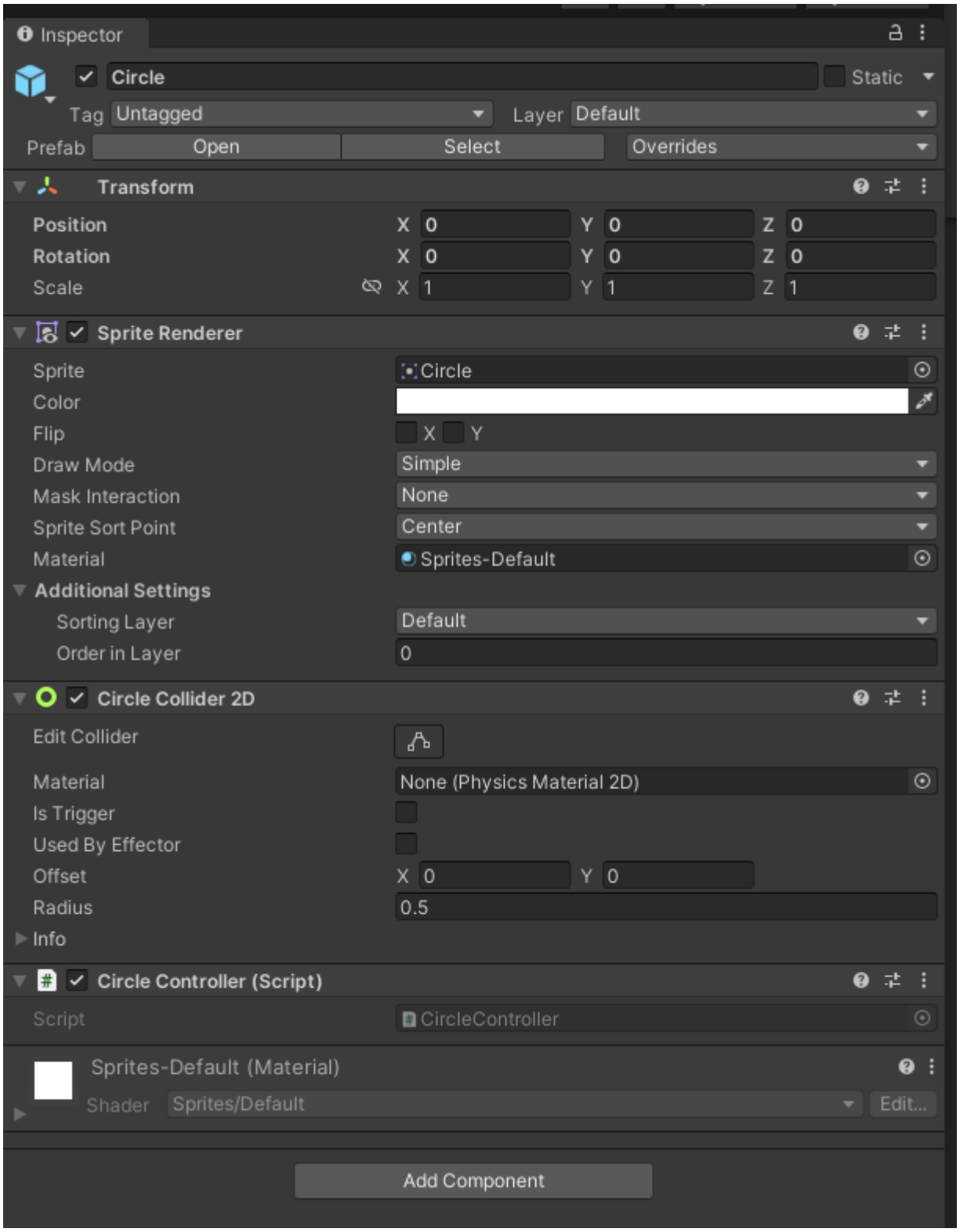


Рисунок 3.9 – Вікно інспектору об’єкта «Circle»

Коли прототип додатку створено, переходимо до створення графічного наповнення, для чого створюємо дизайн додатку на платформі Figma рис. 3.10.

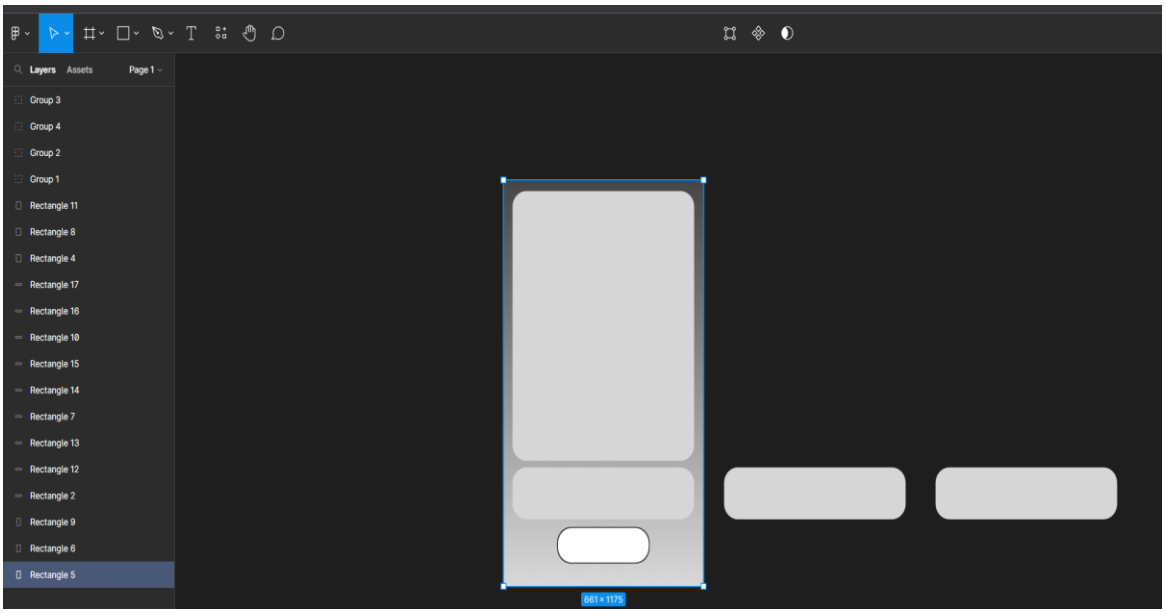


Рисунок 3.10 – Дизайн гри «RememberIT»

Експортуємо всі елементи дизайну в форматі png та імпортуємо в Unity, встановивши тип текстури на Sprite 2D.

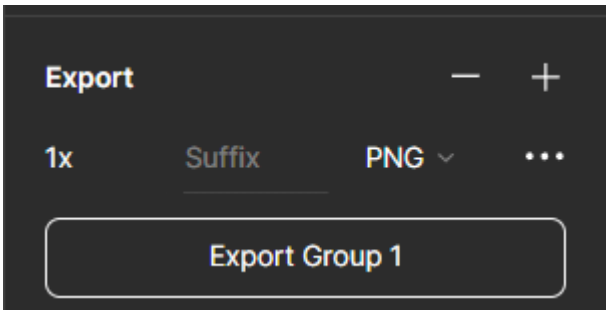


Рисунок 3.11 – Экспорт елементу UI

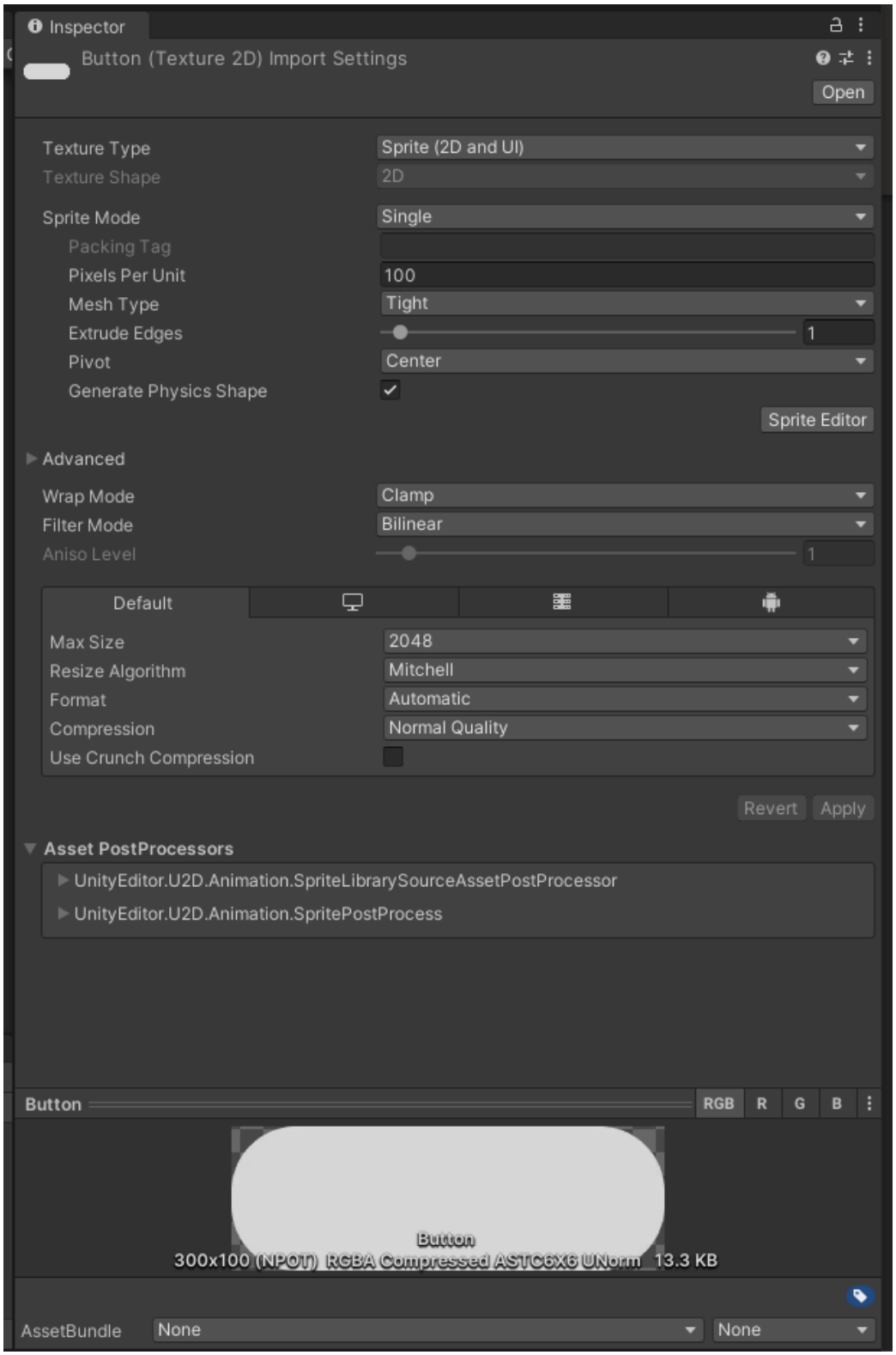


Рисунок 3.12 – Вікно інспектору імпортованого елементу UI

Завершальним етапом йде збірка проекту до apk файлу та його тестування, для цього відкриваємо File – Build Settings – Player Settings та налаштовуємо всі необхідні параметрию рис. 3.13.

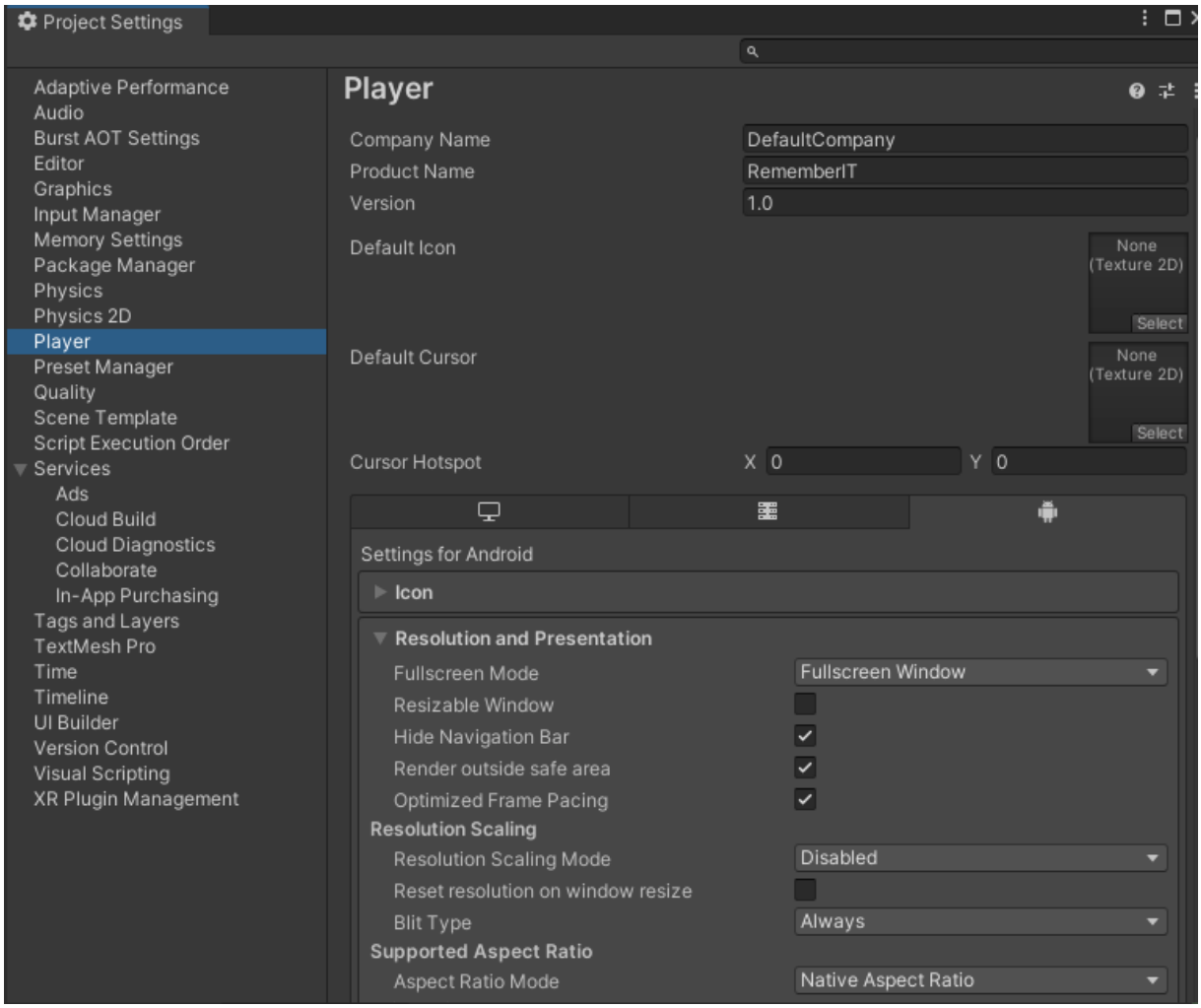


Рисунок 3.13 – Вікно налаштувань гри

Далі повертаємось до Build Settings, додаємо всі створені сцени та перевіряємо всі необхідні налаштування зборки проекту, виконуємо компіляцію проекту рис. 3.14.

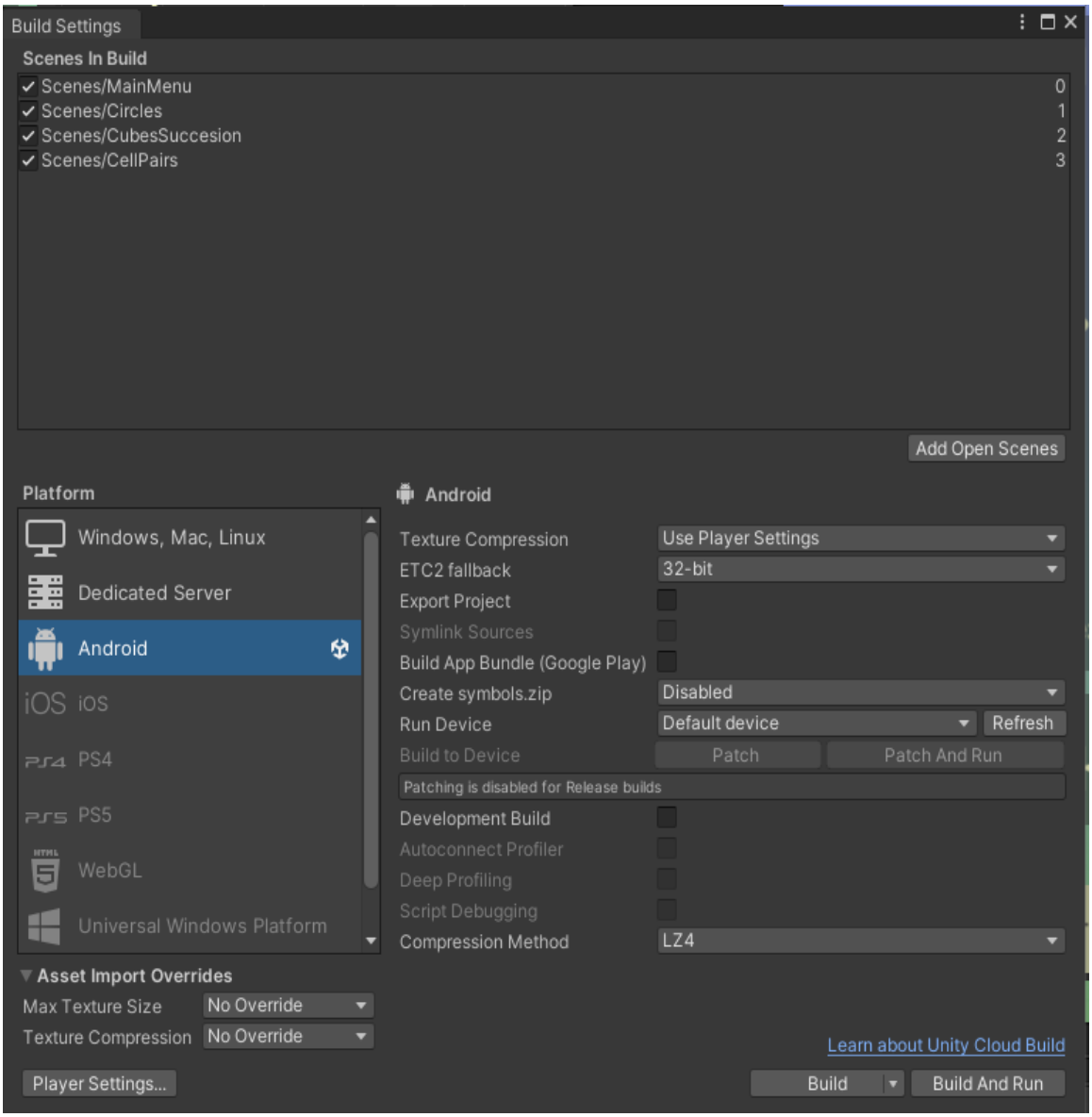


Рисунок 3.14 – Вікно налаштувань збірки проекту

Скриптова частина проекту представлена у таблиці 3.1, де зазначено кожен скрипт із його функціональним призначенням.

Таблиця 3.1 – Опис скриптів та їх алгоритмів

Назва скрипту	**Що робить**
DataContainer.cs	Служить для зберігання інформації про складність та найкращого результату за кожен рівень
PickScene.cs	Призначений для реалізації переходу між сценами за допомогою івентів
ScrollSystemF.cs	Відповідає за роботу горизонтального скролу складності кожного рівня
ScrollSystemF1.cs	Відповідає за роботу вертикального скролу вибору рівнів
SettingsSetter.cs	Служить для зручного налаштування цільового фреймрейту гри
BestScore.cs	Відповідає за відображення найкращого результату рівня з колами
CircleController.cs	Служить для контролю логіки окремого кола та зчитування натискань на нього
CirclesGameplay.cs	Контролює алгоритм роботи рівня з колами
CircleSpawner.cs	Відповідає за підготовку рівня з колами
CellController.cs	Служить для контролю логіки окремої клітинки та зчитування натискань на неї
CellGenerator.cs	Відповідає за підготовку рівня із послідовністю клітинами
CellsBestScore.cs	Відповідає за відображення найкращого результату рівня з послідовністю клітин
CellsGameplay.cs	Контролює алгоритм роботи рівня з послідовністю клітин

Продовження таблиці 3.1 – Опис скриптів та їх алгоритмів

Назва скрипту	**Що робить**
CellPairsBestScore.cs	Відповідає за відображення найкращого результату рівня з відповідністю клітин
CellPairsController.cs	Служить для контролю логіки окремої клітинки та зчитування натискань на неї(для рівню із відповідністю)
CellPairsGenerator.cs	Служить для підготовки рівню із відповідністю клітин
CellsPairsGameplay.cs	Контролює алгоритм роботи рівня із відповідністю клітин

З повним кодом кожного із скриптів можна ознайомитись в додатку В.

3.3 Демонстрація роботи

Після успішної збірки проекту слід продемонструвати роботи мобільного ігрового застосунку на смартфоні. Та протестувати усі функції. Для цього встановлюємо на смартфон отриманий apk file та запускаємо застосунок. Перша сцена що запускається – сцена головного меню. В ній можна обирати рівні свайпом вниз чи вгору, та для кожного рівня обирати рівень складності, свайпом вліво та вправо відповідно, під час зміни складності задній фон застосунку змінює колір на відповідний до складності. Також, на екрані кожного рівня є його короткий опис, та найкращій результат, що зберігається в кеші додатку при виході з нього. Доступна кнопка для запуску рівня із обраною складністю.

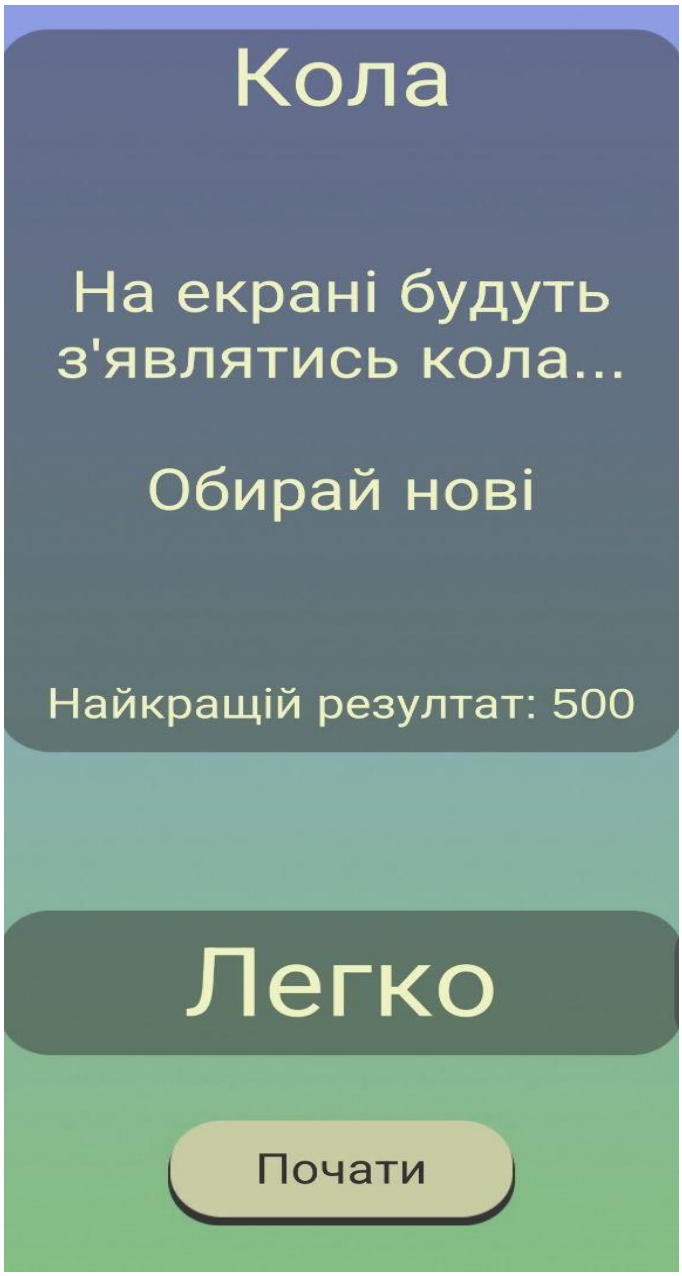


Рисунок 3.15 – Демонстрація головного меню

Рівень з колами запускається (рис 3.16), на екрані поступово з’являються нові кола, користувач обирає нові, рахунок зростає, у випадку, якщо користувач помиляється, гра закінчується, результат зберігається, та відкривається вікно із меню програшу (рис 3.17). В залежності від обраної складності змінюється розмір кіл, та їх колір.

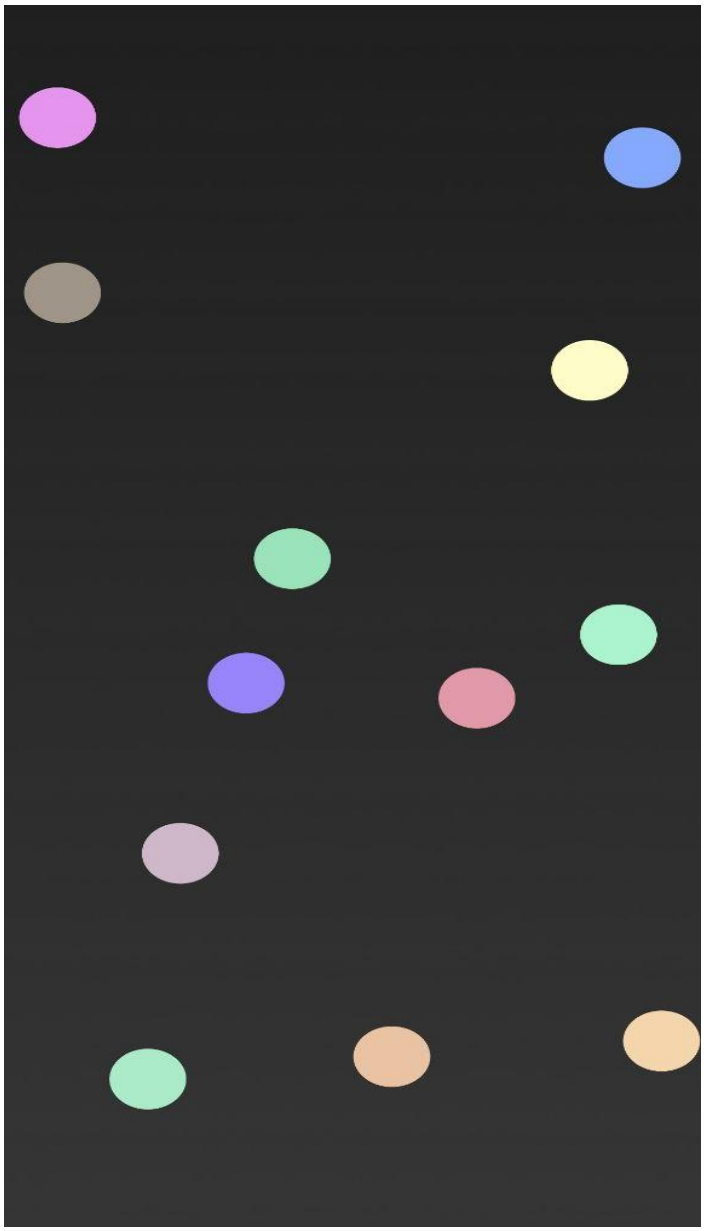


Рисунок 3.16 – Демонстрація роботи рівня із колами

Рисунок 3.17 – Демонстрація меню при програші

Рівень із послідовністю клітин реалізує наступну вправу, на екрані є поле клітин, користувачу показується послідовність натискань, яку він має повторити (рис 3.18). В залежності від обраної складності змінюється довжина послідовності та кількість клітин на екрані. Після повторення послідовності рахунок зростає, при помилковому натиску, що не відповідає послідовності гра закінчується.

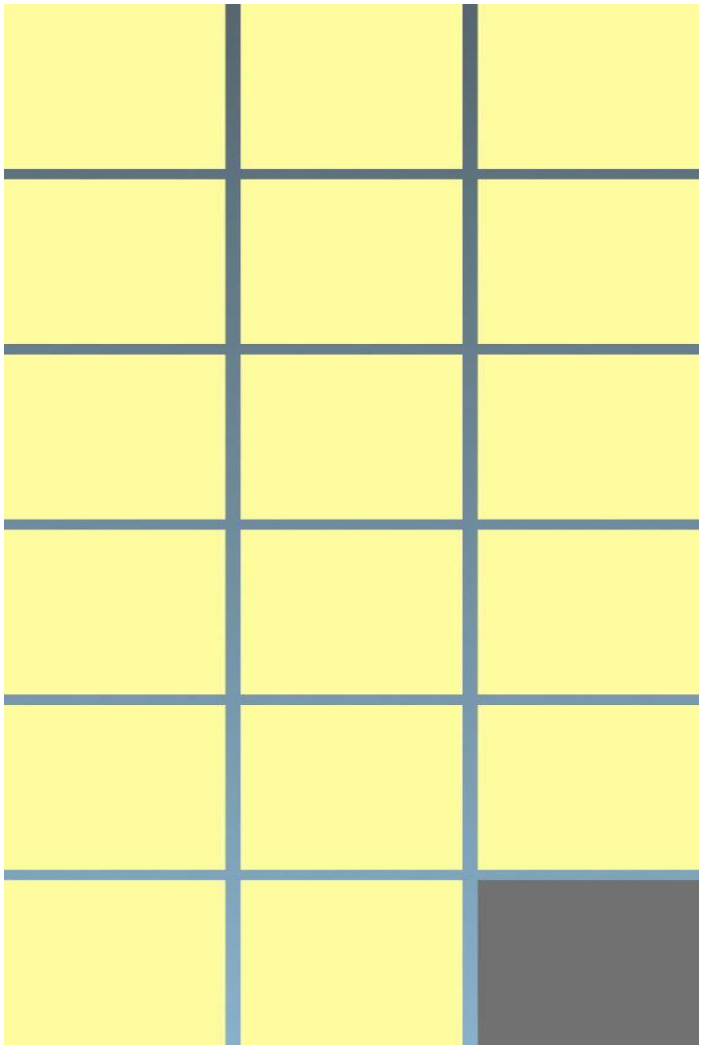


Рисунок 3.18 – Демонстрація рівню із послідовністю клітин

Наступний рівень полягає у виборі користувачем пар клітин, колір яких співпадає (рис 3.19). Напочатку у користувача закриті всі клітини, він може обирати дві, якщо колір співпадає, вони залишаються відкритими, якщо не співпадає, користувач має запам’ятати їх кольори та продовжити шукати їм пару. Гра закінчується в двох випадках, якщо користувач витратив усі спроби, що залежать від рівню складності, як і розмір поля(кількість клітин) або якщо користувач відкрив всі пари.

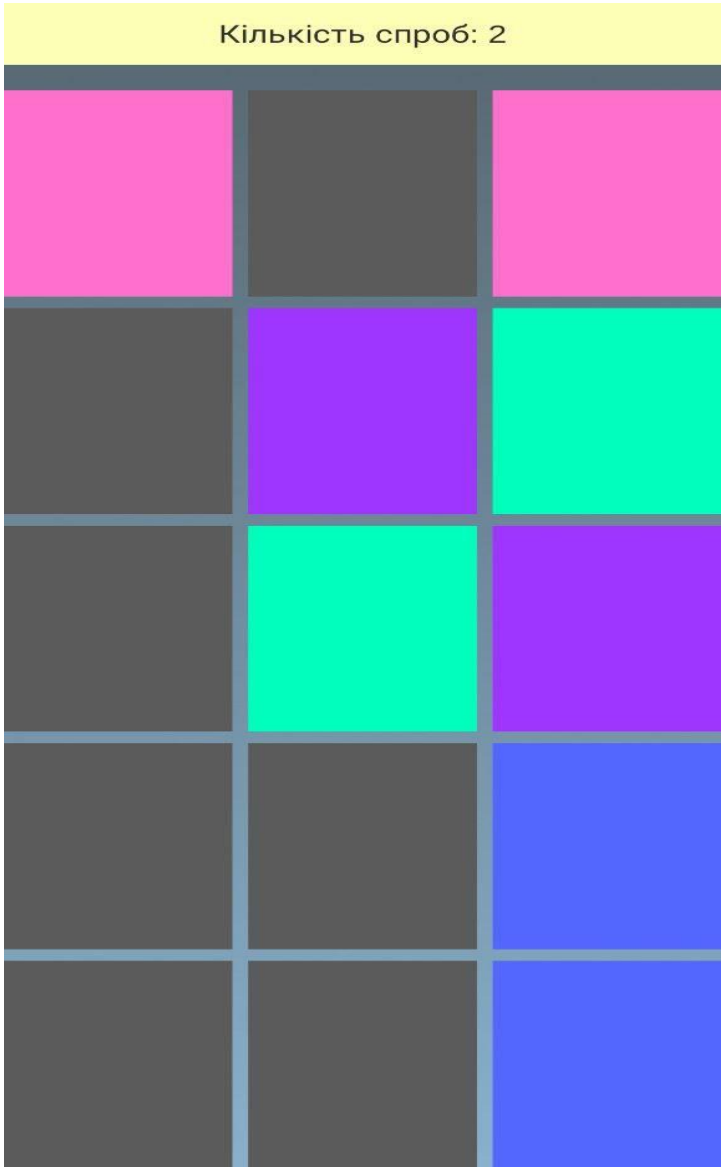


Рисунок 3.19 – Демонстрація рівню із послідовністю клітин

Гра протестована, всі функції працюють коректно. Всі функціональні вимоги, визначені в технічному завданні, реалізовано.

ВИСНОВКИ

Під час виконання роботи було проаналізовано та обґрунтовано актуальність обраної теми. Як результат виникла ідея для розробки власного ігрового мобільного додатку, спрямованого на покращення пам’яті та уваги користувачів, з метою показати альтернативний варіант використання ігрових технологій.

Було досліджено ринок мобільної ігрової індустрії на предмет проектів-аналогів, які були проаналізовані та порівнянні між собою. Характеристики, за якими виконувалося порівняння, обрано основними вимогами до майбутньої розробки.

Наступним етапом стала підготовка ідеї та постановка задачі на розробку, створено технічне завдання. Виявлена мета та ціль проекту, проведено аналіз методологією SMART. Було створено технічне завдання, що відповідає усім нормам. Проаналізовано та створено матрицю ризиків і плану реагування.

Виконано моделювання ігрового додатку у відповідності до сучасних методологій та методів моделювання. Розроблено контекстну діаграму IDEF0, виконано декомпозицію контекстної діаграми, створена UML діаграма варіантів використання.

У відповідності до технічного завдання з використанням обраних засобів реалізації виконано програмну реалізацію ігрового додатку «RemeberIT». Проведене тестування підтверджує його працездатність та наявність визначеного функціоналу. Гра завантажена на PlayMarket.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Video Game Consumer Population [Електронний ресурс] – Режим доступу до ресурсу: https://www.dfcint.com/global-video-game-consumer-population/.
2. Attention and Concentration Challenges in Children and Young People [Електронний ресурс] – Режим доступу до ресурсу: Elevate - Brain Training [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.wonder..
3. Video Game definition [Електронний ресурс] – Режим доступу до ресурсу: https://dictionary.cambridge.org/dictionary/english/video-game.
4. Video Game History [Електронний ресурс] – Режим доступу до ресурсу: https://www.history.com/topics/inventions/history-of-video-games#sources.
5. Essential Facts About the Video Game Industry [Електронний ресурс] – Режим доступу до ресурсу: https://www.theesa.com/resource/2020-essential-facts/.
6. Time spent playing video games is unlikely to impact well-being [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://royalsocietypublishing.org/doi/10.1098/rsos.220411.
7. ДОСЛІДЖЕННЯ ВПЛИВУ КОМП'ЮТЕРНИХ ІГОР НА РОЗВИТОК ДІТЕЙ [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/127079/psihologiya/doslidzhennya_vplivu_kompyuternih_igor_rozvitok_ditey.

8. Psychosocial Impacts of Mobile Game [Електронний ресурс] – Режим доступу до ресурсу: https://www.frontiersin.org/articles/10.3389/feduc.2022.843090/full.
9. Games as Research Tools [Електронний ресурс] – Режим доступу до ресурсу: https://conductscience.com/games/.
10. Application of the Educational Game [Електронний ресурс] – Режим доступу до ресурсу: https://www.frontiersin.org/articles/10.3389/feduc.2021.623793/full.
11. How Many People Own Smartphones (2023-2028) [Електронний ресурс] – Режим доступу до ресурсу: https://explodingtopics.com/blog/smartphone-stats.
12. Unity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/index.html.
13. Mobile Gaming Market [Електронний ресурс] – Режим доступу до ресурсу: https://www.psmarketresearch.com/market-analysis/mobile-gaming-market.
14. Remembery [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=ua.krou.remembery&hl=en_GB.Game 2
15. Elevate - Brain Training [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.wonder.
16. Lights: A memory game [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=us.leephillips.lights&hl=en_GB.
17. Unity Game Development Engine: A Technical Survey [Електронний

ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey.

18. The effect of monetization in the gaming industry [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/340983180_The_effect_of_monetization_in_the_gaming_industry_BA_Honours_in_Games_THESIS_Panagiotis_Markopoulos.

19. What is IDEF [Електронний ресурс] – Режим доступу до ресурсу: https://www.edrawsoft.com/what-is-idef.html.

20. Github репозітарій [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/NickDen-exe.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку мобільного додатку
«RememberIT»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Ващенко С.М.

Студент групи ІТ-91

_____ Денисенко М.С.

Суми 2023

1 Призначення й мета створення мобільного додатку

1.1 Призначення мобільного додатку

Мобільний додаток має виконувати розважальну функцію, коротати час, приносячи користь для мозку, тренуючи пам’яті та уважність.

1.2 Мета створення мобільного-додатку

Популяризація розвиваючих мобільних ігор.

1.3 Цільова аудиторія

До цільової аудиторії мобільного-додатку можна віднести практично всіх людей, які так чи інакше використовують телефон в розважальних цілях. Проте основна категорія - це діти та підлітки, щоб їх захоплення було корисним.

2 Вимоги до мобільного-додатку

2.1 Вимоги до мобільного-додатку в цілому

2.1.1 Вимоги до структури й функціонування мобільного-додатку

Мобільний-додаток має бути сумісний із операційною системою Android. Реалізація додатку має бути виконана із використанням ігрового рушія Unity, мовою програмування в якому використовується C#. Кінцевий продукт додатку має поширюватись на безкоштовній моделі “free to play”[18] та бути доступним до завантаження через PlayMarket.

2.1.2 Вимоги до дизайну

Зважаючи на цільову аудиторії, інтерфейс додатку має бути простим і інтуїтивно зрозумілим. Дизайн слід витримати в одному стилі, надаючи перевагу мінімалізму. Проект має бути реалізовано двовимірною графікою. Допускається як використанні сторонніх ассетів, спрайтів та шрифтів, за умови наявності дозволу на комерційне використання, так і власних.

2.1.3 Вимоги до ігрового процесу

Гра орієнтована на розвинення пам’яті та уважності. Вона має виконувати також і функцію коротання часу. Ігровий процес має зацікавити користувача, але не віднімати багато часу. Гра має містити різні вправи на тренування пам’яті, оформлені у вигляді різних ігрових режимів, кожен з яких має містити градацію за складністю, щоб користувачам будь якого віку та розвиненості було цікаво.

2.1.4 Вимоги до апаратного забезпечення

Щоб охопити якомога більшу аудиторію, програмний продукт має бути добре оптимізованим. Додаток має бути сумісним із більшістю популярних версій Android, на момент 2023 року. Орієнтовні апаратні вимоги: Android, версії 7.0 або новіше, до 100мб вільної пам’яті.

2.2 Структура мобільного додатку

2.2.1 Структура алгоритму додатка

Мобільний додаток має реалізовувати декілька вправ спрямованих на тренування пам’яті, представлених у вигляді ігрових режимів. Користувач зможе обирати режим гри та налаштовувати складність. По завершенню вправи, користувач бачить набраний рахунок, найкращий результат

користувача відображається у головному меню. Користувач може обрати іншу вправу або вийти зі гри.

2.2.2 Структура ігрових режимів

Очікується реалізація наступних вправ:

- Виявлення нового елементу.

На екрані з’являється коло, користувач повинен натиснути на нього, після чого на екрані з’являється нове, користувач має вірно виявляти нові кола, які з’являються з кожною ітерацією. За кожну вірну відповідь користувач отримає бали, що є його рахунком за вправу. Якщо користувач обрав коло яке вже було, гра закінчується.

Градація складності може бути виконана із використання кіл різного розміру та кольору, або одного розміру та кольору, щоб ускладнити процес.

- Виявлення відповідності

На екрані є закриті клітини, на декілька секунд вони відкриваються та користувач бачить яка фігура знаходиться під яким полем. Клітини закриваються, і користувач отримує фігуру яку потрібно знайти. Необхідно згадати під якою клітиною була відповідна фігура та відкрити її, натиснувши.

Градація складності може задіяти різну кількість клітин, колір фігур, та кількість шуканих фігур.

2.2.3 Структура інтерфейсу додатка

Для навігації проектом потрібно розробити унікальний інтерфейс, який буде простим та інтуїтивно зрозумілим. Навігація має відбуватись з головного меню додатку. Бажано уникати додаткових меню, щоб зберегти легкість навігації. Користувач зможе обирати режими гри та складність з головного меню та бачити найкращі результати з кожної вправи. Для виходу із додатку бажано задіяти системну кнопку смартфону. Інтерфейс повинен відповідати сучасним трендам в UI та UX.

3 Склад і зміст робіт зі створення мобільного додатку

Докладний опис етапів роботи зі створення мобільного-додатку наведено в таблиці А.1.

Таблиця А.1 – Етапи створення мобільного додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Написання та узгодження технічного завдання	4 дні
2	Планування та розподіл робіт	2 дні
3	Створення проекту та інтеграція системи контролю версій	1 день
4	Створення прототипу додатка із використанням примітивів	12 днів
5	Тестування прототипу	3 дні
6	Створення графічного наповнення	6 днів
7	Створення аудіо наповнення	2 дні
8	Збірка проекту та його тестування	4 дні
9	Реліз мобільного додатку	1 день
	Загальна тривалість робіт	36 днів

4 Вимоги до складу й змісту робіт із введення мобільного додатку в експлуатацію

Додаток має використатись на смартфонах із операційною системою Android та бути розміщеним на Play Market. Для цього необхідно створити аккаунт розробника Google Developer, та придбати ліцензію на публікацію. Додаток має пройти перевірку на публікацію. Також необхідно локалізувати гру під український та міжнародний ринок.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Визначення мети проекту

Кінцевим продуктом цієї роботи є ігровий мобільний додаток під операційну систему Android. Додаток має виконувати розважальну функцію, коротання часу та розвиваючу, покращуючи пам'ять та увагу користувача. Застосунок буде відноситись до розвиваючих та містити в собі перевірені вправи, що приносять користь, представлені у вигляді ігрових режимів.

Одним із необхідних та важливих етапів розробки є підготовка концептуальної ідеї та детальний її аналіз. Визначення мети проекту буде проведено за однією з найпопулярніших методологій – SMART. Результати аналізу представленні в таблиці Б1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific	Створення розвиваючої мобільної гри, для покращення пам’яті та уваги користувача. Отримання досвіду використання ігрових технологій в навчанні.
Measurable	Результатом роботи є опублікований застосунок на Play Market

Продовження табл. Б.1

Achievable	Програмна реалізація продукту здійснюється із використанням ігрового рушія Unity, в поєднанні із IDE Visual Studio, та використанням мови програмування C#. Звуковий та графічний супровід можуть бути як зроблені самостійно, так і запозичені, при наявності актуальної ліцензії на комерційне використання
Relevant	Технічне та програмне забезпечення відповідає вимогам проекту. Ліцензія розробника Play Market наявна
Time-framed	Проект має бути завершено до 1 червня 2023 року

Планування змісту структури робіт

Для того щоб проект було виконано якісно та у визначені терміни необхідна відповідна підтримка та управління проектом. Важливим є етап розбиття проекту на чіткі задачі та під задачі у поєднанні із побудовою WBS діаграми, що представляє собою візуальну, ієрархічну декомпозицію проекту, орієнтовану на виділення та досягнення цілей роботи.

Роботу слід починати з побудови чіткого переліку завдань, розбиття головної цілі на більш прості та конкретизовані. Окрім детального представлення ієрархії етапів розробки, важливим аспектом є дотримання термінів проекту. Усі роботи мають бути сплановані та розподілені за часом та кінцевими термінами. Такий підхід дає змогу мати повне уявлення про обсяг робіт, проводити аналіз поточної ситуації на певному етапі розробки та оптимізувати роботу з міркувань “тайм-менеджменту”.

Таким чином WBS структура включає в себе усі завдання на розробку, пов’язані чіткою структурою діаграми, кожне з яких має:

- назву задачі;
- ідентифікаційний послідовний номер задачі;
- час виконання задачі;
- дату завершення задачі.

Побудована WBS діаграма згідно із технічного завдання на розробку проекту “RememberIT” представлена на рисунку Б1.

Планування структури організації

Після побудови WBS діаграми, виникає необхідність реалізації структури OBS, яка відповідає за організацію визначення виконавців робіт та проектних груп, відповідальності та підпорядкування. Така структура допомагає визначити чітке розподілення обов’язків між учасниками проекту та виконувати управління проектом більш ефективно.

Для успішної реалізації OBS діаграми необхідно:

- Зобразити ієрархічно всі етапи проекту
- Визначити всіх учасників, виконавців проекту та проектних груп
- Розподілити зони відповідальності та етапи розробки
- Зазначити фіксовані терміни на кожен етап розробки

Побудована OBS діаграма проекту “RememberIT” представлена на рисунку Б2.

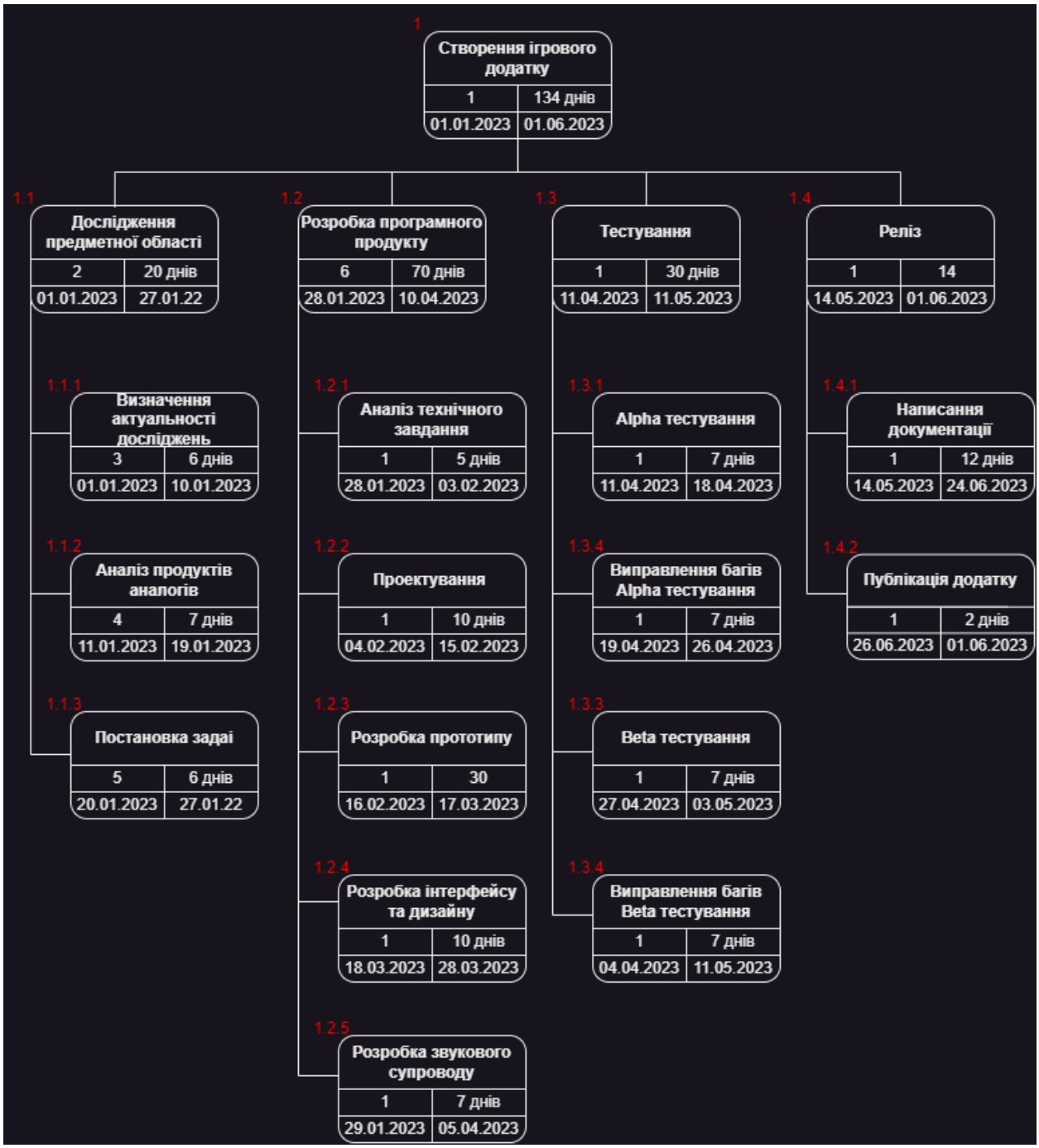


Рисунок Б1 – WBS структура робіт проекту

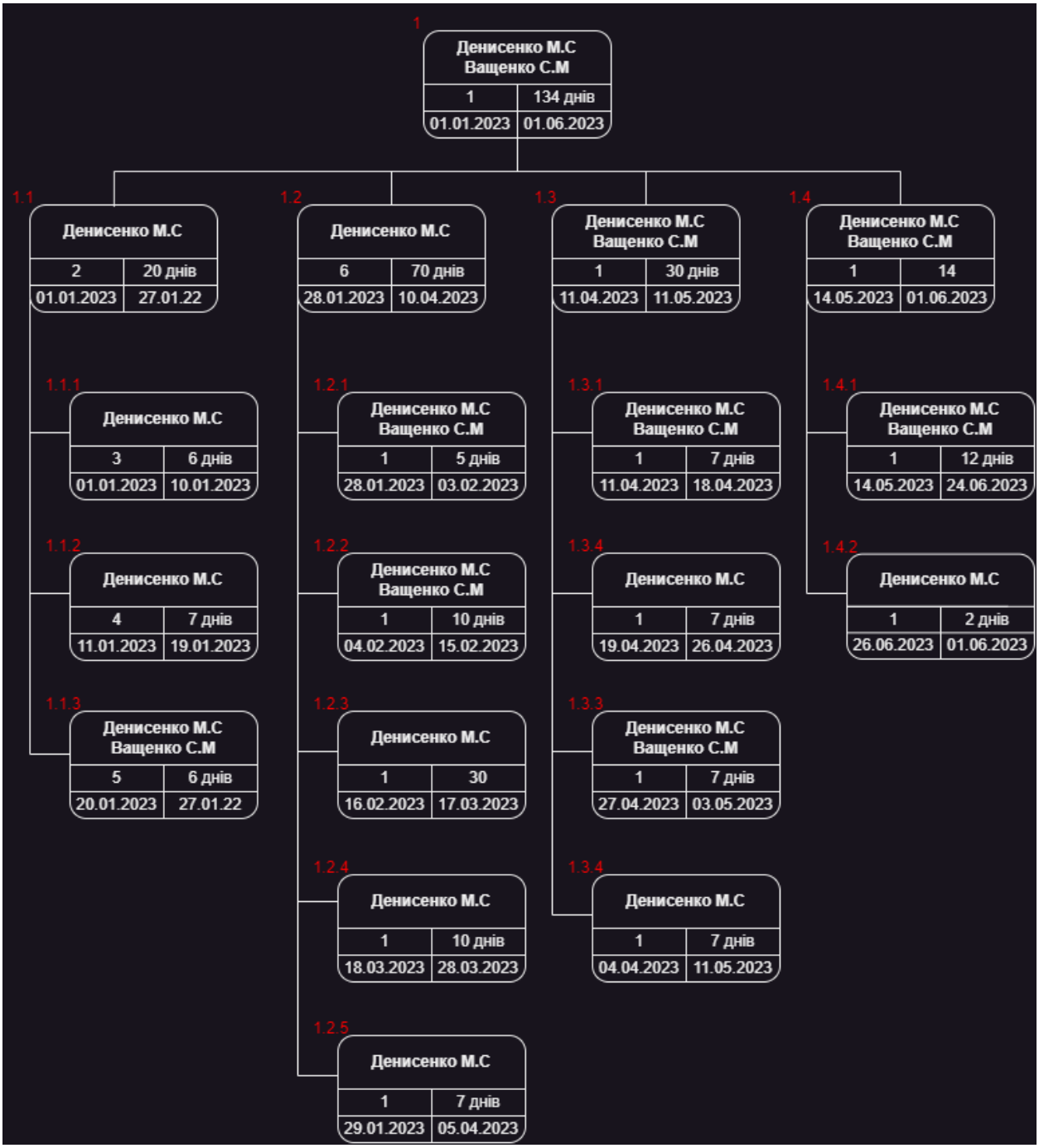


Рисунок Б2 – OBS структура проекту

Визначені виконавці із зонами відповідальності зазначені в таблиці Б2.

Таблиця Б2 – Виконавці проекту

Роль	**Ім’я**	**Зона відповідальності**
Проектувальник	Денисенко М.С	Проектування архітектури та алгоритму проекту
Розробник	Денисенко М.С	Розробка проекту та реалізація механік, написання скриптів та збірка проекту
Графічний дизайнер	Денисенко М.С	Створення або забезпечення графічного дизайну та інтерфейсу
Звуковий дизайнер	Денисенко М.С	Створення або забезпечення звукового супроводу проекту
Тестувальник	Денисенко М.С Ващенко С.М	Тестування проекту, створення баг-репорту, виправлення багів, за їх наявності
Керівник проекту	Ващенко С.М	Ведення проекту, управління робочими групами, надання технічного завдання, прийняття оперативних рішень

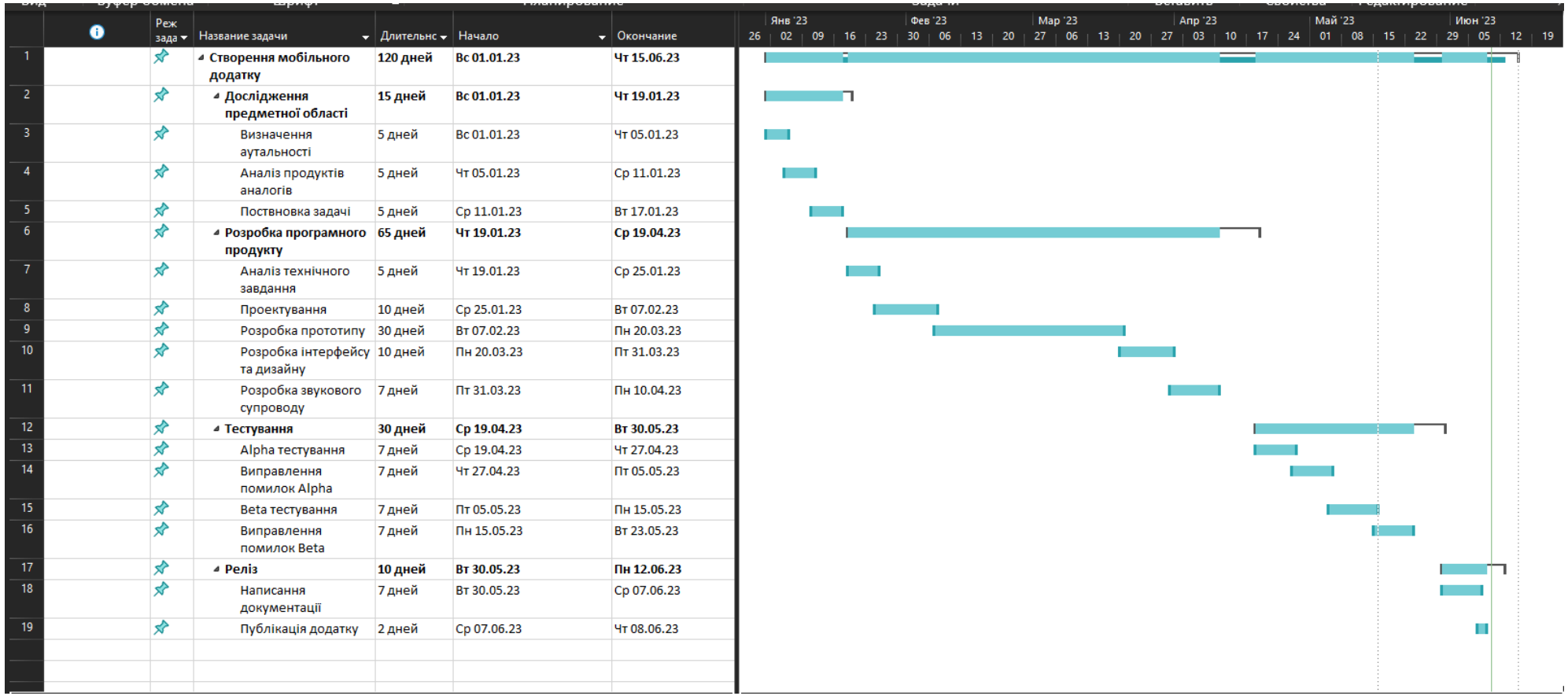
Діаграма Ганта

Наступним етапом є створення та побудова календарного плану реалізації проекту. Незалежно від галузі, одним з найпопулярніших форматів є діаграма Ганта. Діаграма Ганта представляє собою професійний інструмент для ведення проекту, використовується для планування та розподілу етапів за

часовими межами, визначення термінів та представлення цієї інформації в зручному для сприйняття виді. На цій діаграмі можливо відстежувати просування робіт над проектом, які представленні у вигляді горизонтальних ліній, накладених на ієрархію всього проекту з оглядом на терміни по кожній з них.

Створення діаграми Ганта будемо виконувати в одній з найпоширеніших програм для управління проектами – MS Project.

Діаграма Ганта проекту “RememberIT” представлена на рисунку Б3.



	Реж зада	Название задачи	Длительнс	Начало	Окончание
1		**Створення мобільного додатку**	**120 дней**	**Вс 01.01.23**	**Чт 15.06.23**
2		**Дослідження предметної області**	**15 дней**	**Вс 01.01.23**	**Чт 19.01.23**
3		Визначення аутальності	5 дней	Вс 01.01.23	Чт 05.01.23
4		Аналіз продуктів аналогів	5 дней	Чт 05.01.23	Ср 11.01.23
5		Постановка задачі	5 дней	Ср 11.01.23	Вт 17.01.23
6		**Розробка програмного продукту**	**65 дней**	**Чт 19.01.23**	**Ср 19.04.23**
7		Аналіз технічного завдання	5 дней	Чт 19.01.23	Ср 25.01.23
8		Проектування	10 дней	Ср 25.01.23	Вт 07.02.23
9		Розробка прототипу	30 дней	Вт 07.02.23	Пн 20.03.23
10		Розробка інтерфейсу та дизайну	10 дней	Пн 20.03.23	Пт 31.03.23
11		Розробка звукового супроводу	7 дней	Пт 31.03.23	Пн 10.04.23
12		**Тестування**	**30 дней**	**Ср 19.04.23**	**Вт 30.05.23**
13		Alpha тестування	7 дней	Ср 19.04.23	Чт 27.04.23
14		Виправлення помилок Alpha	7 дней	Чт 27.04.23	Пт 05.05.23
15		Beta тестування	7 дней	Пт 05.05.23	Пн 15.05.23
16		Виправлення помилок Beta	7 дней	Пн 15.05.23	Вт 23.05.23
17		**Реліз**	**10 дней**	**Вт 30.05.23**	**Пн 12.06.23**
18		Написання документації	7 дней	Вт 30.05.23	Ср 07.06.23
19		Публікація додатку	2 дней	Ср 07.06.23	Чт 08.06.23

Рисунок Б3 – Діаграма Ганта

Аналіз ризиків

Жоден проект не може існувати без повноцінної підготовки, включаючи і аналіз ризиків. Ця важлива частина допомагає уникнути непередбачуваних обставин або ж зменшити їх вплив на виконання проекту.

Під час аналізу ризиків необхідно прорахувати усі можливі випадки, що можуть зашкодити реалізації проекту. Отримані ризики, слід проаналізувати окремо і в порівнянні з іншими для того щоб визначити пріоритетність та критичність кожного з них. Ризики що мають високу вірогідність або серйозний вплив на розробку мають бути в пріоритеті до усунення. Реагування на них має проводитись якнайшвидше. Для цього необхідно розробити таблицю ризиків, які будуть аналізуватись за шкалою оцінювання.

Шкалу оцінювання представлено в таблиці Б3.

Таблиця Б.3 – Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	**Ймовірність виникнення**	**Вплив ризику**
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

Таблиця Б4 – Матриця ймовірності виникнення ризику

	1	2	3
1	R2		R1,R9
2	R3	R5	R6,R7
3	R10	R4	R8

Таблиця Б5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики
1	Прийнятні	1≤R≤3	2
2	Виправдані	3≤R≤4	1,3,5,9,10
3	Неприпустимі	6≤R≤9	4,6,7,8

Таблиця Б6 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
R1	Відкритий	Збій або вихід із ладу технічного забезпечення	Низька	Високий	4	Регулярне тестування обладнання та своєчасне обслуговування	Попередження	Ремонт вийшовшого з ладу обладнання або купівля нового
R2	Відкритий	Збій або закінчення ліцензії програмного забезпечення	Низька	Низький	2	Регулярна перевірка програмного забезпечення та термінів дії ліцензії	Попередження	Подовження дії ліцензії програмного забезпечення
R3	Відкритий	Поява альтернативного продукту	Середня	Низька	3	Моніторинг ринку продуктів аналогів.	Пом’якшення	Індивідуалізація власного проекту
R4	Відкритий	Нечітке завдання на розробку	Середня	Високий	5	Детальне обговорення завдання на розробку із замовником	Попередження	Проведення додаткових конференцій із замовником для уточнення технічного завдання
R5	Відкритий	Часте внесення змін до технічного завдання	Середня	Середній	4	Проведення регулярних зустрічей із замовником для демонстрації результату. Чітке розуміння мети проекту.	Перенесення	Проведення додаткових конференцій із замовником для внесення змін до технічного завдання

Продовження таблиці Б6 - Оцінка ймовірності виникнення, величини витрат та індексу ризику

R6	Відкритий	Помилки проектування	Середня	Високий	5	Побудова чіткої та масштабованої архітектури на етапі проектування. Виростання архітектурних патернів	Перенесення	Проведення рефакторингу проекту, оптимізації архітектурних рішень
R7	Відкритий	Втрата даних, ісходного коду.	Середня	Високий	5	Інтеграція систем для контролю версій на початковому етапі розробці	Попередження	Регулярне створення бекапів.
R8	Відкритий	Виникнення багів під час тестування	Висока	Високий	6	Своєчасне проведення тестувань, та дотримання термінів всіх етапів розробки	Прийняття	Виправлення помилок, перенос релізу, реалізація оновлень.
R9	Відкритий	Неоптимальний розподіл часу	Низька	Висока	4	Виконання повного обсягу робіт з аналізу проекту та його менеджмент	Попередження	Понад нормована робота, перенос релізу, реалізація оновлень
R10	Відкритий	Конфлікт авторських прав	Низька	Висока	4	Використання асетів лише за наявності ліцензії на комерційне використання	Попередження	Розробка власних асетів або найм спеціалістів під замовлення

ДОДАТОК В

Програмний код

DataContainer.cs

```
using UnityEngine;
using UnityEngine.UI;

public class DataContainer : MonoBehaviour
{
    [SerializeField] ScrollSystemF circles, cells, cellPairs;
    private static int CirclesDiff, CellDiff, CellPairsDiff;

    private static float MaxCirclesScore, MaxCellsScore, MaxCellPairsScore;

    private void Start()
    {
        MaxCirclesScore = PlayerPrefs.GetFloat("MaxCirclesScore", 0);
        MaxCellsScore = PlayerPrefs.GetFloat("MaxCellsScore", 0);
        MaxCellPairsScore = PlayerPrefs.GetFloat("MaxCellPairsScore", 0);
    }

    public void SetCirclesDiff()
    {
        CirclesDiff = circles.GetCurrnetDiff();
    }

    public void SetCellsDiff()
    {
        CellDiff = cells.GetCurrnetDiff();
    }

    public void SetCellPairsDiff()
    {
        CellPairsDiff = cellPairs.GetCurrnetDiff();
    }

    public static void SetCirclesBest(float score)
    {
        if (score > MaxCirclesScore)
            MaxCirclesScore = score;
        PlayerPrefs.SetFloat("MaxCirclesScore", MaxCirclesScore);
    }

    public static void SetCellsBest(float score)
    {
        if (score > MaxCellsScore)
            MaxCellsScore = score;
        PlayerPrefs.SetFloat("MaxCellsScore", MaxCellsScore);
    }

    public static void SetCellPairsBest(float score)
    {
        if (score > MaxCellPairsScore)
            MaxCellPairsScore = score;
        PlayerPrefs.SetFloat("MaxCellPairsScore", MaxCellPairsScore);
    }

    public static float GetCellsBest()
    {
        if(MaxCellsScore>0)
            Debug.Log("DC " + MaxCellsScore);
        return MaxCellsScore;
    }

    public static float GetCellPairsBest()
    {
        return MaxCellPairsScore;
    }

    public static float GetCirclesBest()
    {
        return MaxCirclesScore;
    }
```

```
    public static int GetCirclesDifficult()
    {
        return CirclesDiff;
    }
    public static int GetCellDifficult()
    {
        return CellDiff;
    }
    public static int GetCellPairsDifficult()
    {
        return CellPairsDiff;
    }

    private void OnApplicationQuit()
    {
        PlayerPrefs.Save();
    }
}
```

PickScene.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class PickScene : MonoBehaviour
{
    public void ReloadScene(int neededSceneIndex)
    {
        SceneManager.UnloadScene(neededSceneIndex);
        SceneManager.LoadScene(neededSceneIndex);
    }

    public void LoadScene(int neededSceneIndex)
    {
        SceneManager.LoadScene(neededSceneIndex);
    }
}
```

ScrollSystemF.cs

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

public class ScrollSystemF : MonoBehaviour, IDragHandler, IEndDragHandler
{
    [SerializeField] private int SceneID;
    [SerializeField] private Image page;
    [SerializeField] List<Color> colors;
    private Vector3 panelLocation;
    public float percentThreshold = 0.2f;
    public float easing = 0.5f;
    public int totalPages = 3;
    private int currentPage = 1;

    void Start()
    {
        panelLocation = transform.position;
    }

    public void OnDrag(PointerEventData data)
    {
        panelLocation.y = transform.position.y;
        float difference = data.pressPosition.x - data.position.x;
        transform.position = panelLocation - new Vector3(difference, 0, 0);
    }

    public void OnEndDrag(PointerEventData data)
    {
        float percentage = (data.pressPosition.x - data.position.x) / Screen.width;
        if (Mathf.Abs(percentage) >= percentThreshold)
        {
            Vector3 newLocation = panelLocation;
            if (percentage > 0 && currentPage < totalPages)
            {
                currentPage++;
```

```
                newLocation += new Vector3(-Screen.width, 0, 0);
            }
            else if (percentage < 0 && currentPage > 1)
            {
                currentPage--;
                newLocation += new Vector3(Screen.width, 0, 0);
            }

            newLocation.y = transform.position.y;
            StartCoroutine(SmoothMove(transform.position, newLocation, easing));
            panelLocation = newLocation;
        }
        else
        {
            panelLocation.y = transform.position.y;
            StartCoroutine(SmoothMove(transform.position, panelLocation, easing));
        }
    }

    IEnumerator SmoothMove(Vector3 startpos, Vector3 endpos, float seconds)
    {
        float t = 0f;
        while (t <= 1.0)
        {
            t += Time.smoothDeltaTime / seconds;
            transform.position = Vector3.Lerp(startpos, endpos, Mathf.SmoothStep(0f, 1f, t));
            yield return null;
            page.color = colors[currentPage - 1];
        }
    }

    public int GetCurrnetDiff()
    {
        return currentPage;
    }
}
```

ScrollSystemF1.cs

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

public class ScrollSystemF1 : MonoBehaviour, IDragHandler, IEndDragHandler
{
    private Vector3 panelLocation;
    public float percentThreshold = 0.2f;
    public float easing = 0.5f;
    public int totalPages = 3;
    private int currentPage = 1;

    void Start()
    {
        panelLocation = transform.position;
    }

    public void OnDrag(PointerEventData data)
    {
        float difference = data.pressPosition.y - data.position.y;
        transform.position = panelLocation - new Vector3(0, difference, 0);
    }

    public void OnEndDrag(PointerEventData data)
    {
        float percentage = (data.pressPosition.y - data.position.y) / Screen.height;
        if (Mathf.Abs(percentage) >= percentThreshold)
        {
            Vector3 newLocation = panelLocation;
            if (percentage < 0 && currentPage < totalPages)
            {
                currentPage++;
                newLocation += new Vector3(0, +Screen.height, 0);
            }
            else if (percentage > 0 && currentPage > 1)
            {
                currentPage--;
                newLocation += new Vector3(0, -Screen.height, 0);
            }
```

```
            StartCoroutine(SmoothMove(transform.position, newLocation, easing));
            panelLocation = newLocation;
        }
        else
        {

            StartCoroutine(SmoothMove(transform.position, panelLocation, easing));
        }
    }

    IEnumerator SmoothMove(Vector3 startpos, Vector3 endpos, float seconds)
    {
        float t = 0f;
        while (t <= 1.0)
        {
            t += Time.smoothDeltaTime / seconds;
            transform.position = Vector3.Lerp(startpos, endpos, Mathf.SmoothStep(0f, 1f, t));
            yield return null;
        }
    }
}
```

SettingsSetter.cs

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

public class SettingsSetter : MonoBehaviour
{
    [SerializeField] private int TargetFrameRate;
    void Start()
    {
        Application.targetFrameRate = TargetFrameRate;
    }
}
```

BestScore.cs

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

public class BestScore : MonoBehaviour
{
    [SerializeField] private Text bestScoreText;

    void Update()
    {
        bestScoreText.text = "Найкращій результат: " + DataContainer.GetCirclesBest();
    }
}
```

CircleController.cs

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

public class CircleController : MonoBehaviour
{
    CirclesGameplay cg;

    void Start()
    {
        cg = FindObjectOfType<CirclesGameplay>();
    }

    private void OnMouseDown()
    {
        cg.OnObjectClicked(this.gameObject);

    }
}
```

CirclesGameplay.cs

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

public class CirclesGameplay : MonoBehaviour
{
    [SerializeField] private GameObject GetReadyScreen;
    [SerializeField] private GameObject GameOverScreen;
    [SerializeField] private Text scoreTextObject;
    public float scoreIncrement = 100;
    private float score = 0;
    public bool gameOver = false;
    private int dificult;

    private List<GameObject> Circles = new List<GameObject>();
    private int currentIndex = 0;
    public void SetCircle(GameObject circle)
    {
        Circles.Add(circle);
    }

    public void HideCircles(int dif)
    {
        foreach (GameObject circle in Circles)
        {
            circle.SetActive(false);
        }

        dificult = dif;
        if (dif == 2)
        {
            scoreIncrement *= 1.25f;
        }
        else if (dif == 3)
        {
            scoreIncrement *= 1.5f;
        }
    }

    private void ActivateCurrentObject()
    {
        if (currentIndex < Circles.Count && !gameOver)
        {
            Circles[currentIndex].SetActive(true);
            GetReadyScreen.SetActive(false);
        }
    }

    public void OnObjectClicked(GameObject clickedObject)
    {
        if (clickedObject == Circles[currentIndex])
        {
            currentIndex++;
            if (currentIndex < Circles.Count)
            {
                Circles[currentIndex].SetActive(true);
                StartCoroutine(GetReadyDelay());
            }
            else
            {
                gameOver = true;
                DataContainer.SetCirclesBest(score);
                GameOverScreen.SetActive(true);
            }
            score += scoreIncrement;
            UpdateScoreText();
        }
        else
        {
            DataContainer.SetCirclesBest(score);
            gameOver = true;
            GameOverScreen.SetActive(true);
```

```
        }
    }

    private void UpdateScoreText()
    {
        scoreTextObject.text = "рахунок: " + score;
    }

    IEnumerator StartDelay()
    {
        GetReadyScreen.SetActive(true);
        yield return new WaitForSeconds(1.5f);
        ActivateCurrentObject();
        UpdateScoreText();
        GetReadyScreen.SetActive(false);
        yield return null;
    }

    void Start()
    {
        StartCoroutine(StartDelay());
    }

    IEnumerator GetReadyDelay()
    {
        GetReadyScreen.SetActive(true);
        yield return new WaitForSeconds(1f);
        GetReadyScreen.SetActive(false);
    }
}
```

CirclesSpawner.cs

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

public class CircleSpawner : MonoBehaviour
{
    [SerializeField] private GameObject circlePrefab;
    [SerializeField] private float minSize = 0.5f;
    [SerializeField] private float maxSize = 1.5f;
    [SerializeField] private int circleCount = 30;
    [SerializeField] private int maxSpawnAttempts = 10;

    private CirclesGameplay cg;
    private float cameraHeight;
    private float cameraWidth;
    int difficultLevel;
    private Color fixedCol;

    void Start()
    {
        difficultLevel = DataContainer.GetCirclesDifficult();
        if (difficultLevel > 2)
        {
            fixedCol = GetRandomBrightColor();
        }
        cg = this.GetComponent<CirclesGameplay>();
        CalculateCameraBounds();
        SpawnCircles();
    }

    private void CalculateCameraBounds()
    {
        Camera mainCamera = Camera.main;
        cameraHeight = mainCamera.orthographicSize;
        cameraWidth = cameraHeight * mainCamera.aspect;
    }

    private void SpawnCircles()
    {
        int spawnAttempts = 0;
        int circlesSpawned = 0;
        float size;

        while (circlesSpawned < circleCount && spawnAttempts < maxSpawnAttempts)
```

```
        {
            if (difficultLevel < 2)
            {
                if (spawnAttempts < 30)
                {
                    size = Random.Range(minSize, maxSize);
                }
                else if (spawnAttempts > 30 && spawnAttempts < 75)
                {
                    size = Random.Range(minSize, maxSize * 0.75f);
                }
                else
                    size = 0.5f;
            }
            else
                size = maxSize * 0.5f;


            Vector2 spawnPosition = GetRandomPosition(size);
            Collider2D[] colliders = Physics2D.OverlapCircleAll(spawnPosition, size);

            if (colliders.Length < 1)
            {
                GameObject newCircle = Instantiate(circlePrefab, spawnPosition,
Quaternion.identity);
                newCircle.transform.localScale = new Vector3(size, size, 1f);
                if (difficultLevel < 3)
                {
                    newCircle.GetComponent<SpriteRenderer>().color = GetRandomBrightColor();
                }
                else
                    newCircle.GetComponent<SpriteRenderer>().color = fixedCol;
                cg.SetCircle(newCircle);
                spawnAttempts = 0;
                circlesSpawned++;
            }
            else
            {
                spawnAttempts++;

                if (spawnAttempts >= maxSpawnAttempts)
                {
                    Debug.Log("Failed to spawn all circles within the maximum spawn attempts.");
                }
            }
        }
        cg.HideCircles(difficultLevel);
    }

    private Color GetRandomBrightColor()
    {
        float r = Random.Range(0.5f, 1f);
        float g = Random.Range(0.5f, 1f);
        float b = Random.Range(0.5f, 1f);
        return new Color(r, g, b);
    }

    private Vector2 GetRandomPosition(float size)
    {
        float spawnSize = size / 2;
        float xMin = -cameraWidth + spawnSize;
        float xMax = cameraWidth - spawnSize;
        float yMin = -cameraHeight + spawnSize;
        float yMax = cameraHeight - spawnSize;

        float x = Random.Range(xMin, xMax);
        float y = Random.Range(yMin, yMax);

        return new Vector2(x, y);
    }
}
```

CellController.cs

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

```
public class CellController : MonoBehaviour
{
    private CellsGameplay cellsGameplay;

    void Start()
    {
        cellsGameplay = FindObjectOfType<CellsGameplay>();
    }

    private void OnMouseDown()
    {
        cellsGameplay.CheckClickedObj(this.gameObject);
    }
}
```

CellGenerator.cs

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

public class CellGenerator : MonoBehaviour
{
    [SerializeField] CellsGameplay cg;
    private int difficultLevel;

    public GameObject squarePrefab;
    public float padding = 0.1f;
    private Camera mainCamera;

    private void Start()
    {
        mainCamera = Camera.main;
        difficultLevel = DataContainer.GetCellDifficult();
        if (difficultLevel == 1)
        {
            SpawnSquares(3, 4);
        }
        else if (difficultLevel == 2)
        {
            SpawnSquares(3, 5);
        }
        else
        {
            SpawnSquares(3, 6);
        }
    }

    private void SpawnSquares(int squaresInRow, int squaresInColumn)
    {
        float screenWidth = mainCamera.orthographicSize * 2f * mainCamera.aspect;
        float screenHeight = mainCamera.orthographicSize * 2f;

        SpriteRenderer squareRenderer = squarePrefab.GetComponent<SpriteRenderer>();
        float squareWidth = squareRenderer.bounds.size.x;
        float squareHeight = squareRenderer.bounds.size.y;

        float totalPaddingX = (squaresInRow - 1) * padding;
        float totalPaddingY = (squaresInColumn - 1) * padding;

        float squareSizeX = (screenWidth - totalPaddingX) / squaresInRow;
        float squareSizeY = (screenHeight - totalPaddingY) / squaresInColumn;

        float startX = -screenWidth / 2f + squareSizeX / 2f;
        float startY = screenHeight / 2f - squareSizeY / 2f;

        for (int row = 0; row < squaresInColumn; row++)
        {
            for (int column = 0; column < squaresInRow; column++)
            {
                float xPos = startX + column * (squareSizeX + padding);
                float yPos = startY - row * (squareSizeY + padding);
                float zPos = -5f;

                GameObject temp  = Instantiate(squarePrefab, new Vector3(xPos, yPos, zPos),
Quaternion.identity);
                temp.transform.localScale = new Vector3(squareSizeX, squareSizeY, 1f);
                cg.AddCell(temp);
```

```
            }
        }
        cg.CellsAreDone(difficultLevel);
    }
}
```

CellGameplay.cs

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

public class CellsGameplay : MonoBehaviour
{
    private List<GameObject> Cells = new List<GameObject>();
    private List<GameObject> SequenceCells = new List<GameObject>();
    private int currentSquareIndex = 0;
    private bool isGameOver = false;
    private bool GameFlow = false;

    private Color defColor;
    [SerializeField] private Color clickedColor;
    private int SequentiallyAmount = 4;
    [SerializeField]private float delay;
    private bool isChangingColor = false;

    int amount;

    [SerializeField] GameObject PrepScreen;
    [SerializeField] GameObject GameOverScreen;
    [SerializeField] Text currentScore;
    private float score;
    [SerializeField] private float scoreIncrement;

    public void AddCell(GameObject cell)
    {
        Cells.Add(cell);
    }

    public void CellsAreDone(int dif)
    {
        amount = Cells.Count;
        defColor = Cells[0].GetComponent<SpriteRenderer>().color;
        if (dif == 2)
        {
            SequentiallyAmount = 5;
            scoreIncrement *= 1.25f;
        }
        else if (dif == 3)
        {
            SequentiallyAmount = 6;
            scoreIncrement *= 1.5f;
        }
        StartCoroutine(DelayPrepareScreen());
    }

    private IEnumerator ClickedObjDelay(SpriteRenderer sr)
    {
        sr.color = clickedColor;
        yield return new WaitForSeconds(0.2f);
        sr.color = defColor;
    }

    private void GameOver()
    {
        GameFlow = false;
        isGameOver = true;
        DataContainer.SetCellsBest(score);
        GameOverScreen.SetActive(true);
    }

    public void CheckClickedObj(GameObject obj)
    {
        if (GameFlow)
        {
            StartCoroutine(ClickedObjDelay(obj.GetComponent<SpriteRenderer>()));
```

```
            if (obj == SequenceCells[currentSquareIndex])
            {
                score += scoreIncrement;
                UpdateScoreText();
                currentSquareIndex++;

                if (currentSquareIndex >= SequenceCells.Count)
                {
                    GameFlow = false;
                    SequenceCells.Clear();
                    currentSquareIndex = 0;
                    StartCoroutine(DelayPrepareScreen());
                }
            }
            else
                GameOver();
        }
    }

    IEnumerator DelayPrepareScreen()
    {
        PrepScreen.SetActive(true);
        yield return new WaitForSeconds(1f);
        PrepScreen.SetActive(false);
        StartCoroutine(ShowSequentially());
    }

    private int  GetRandomIndex()
    {
        int id = Random.Range(0, amount);
        return id;
    }

    IEnumerator ShowSequentially()
    {
        int changeIndex;
        int changedObjs = 0;
        while (changedObjs != SequentiallyAmount)
        {
            changeIndex = GetRandomIndex();
            isChangingColor = true;
            SequenceCells.Add(Cells[changeIndex]);
            Cells[changeIndex].GetComponent<SpriteRenderer>().color = clickedColor;

            yield return new WaitForSeconds(delay);


            Cells[changeIndex].GetComponent<SpriteRenderer>().color = defColor;
            changedObjs++;
            isChangingColor = false;
        }
        GameFlow = true;
    }

    private void UpdateScoreText()
    {
        currentScore.text = "рахунок: " + score;
    }
}
```

CellPairsController.cs

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

public class CellPairsController : MonoBehaviour
{
    private CellsPairsGameplay cellsPairsGameplay;
    [SerializeField]private Color innerColor;
    [SerializeField]bool opened = false;
    private Color defCol;

    public void SetInnerColor(Color col)
    {
        innerColor = col;
    }
```

```
    void Start()
    {
        cellsPairsGameplay = FindObjectOfType<CellsPairsGameplay>();
        defCol = this.GetComponent<SpriteRenderer>().color;
    }
    private void OnMouseDown()
    {
        if (!opened)
        {
            Debug.Log("Tap");

            this.gameObject.GetComponent<SpriteRenderer>().color = innerColor;
            cellsPairsGameplay.CellOpened(this.gameObject);
        }
    }

    public void SetCellOpenedState()
    {
        opened = true;
    }

    public void LockCell(bool state)
    {
        Debug.Log("Lock called " + state);
        if (state == false)
        {
            StartCoroutine(ReturnCellColor());
        }
        opened = state;
    }

    public Color GetCellColor()
    {
        return innerColor;
    }

    IEnumerator ReturnCellColor()
    {
        yield return new WaitForSeconds(0.5f);
        this.gameObject.GetComponent<SpriteRenderer>().color = defCol;
    }
}
```

CellPairsGenerator.cs

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

public class CellPairsGenerator : MonoBehaviour
{
    [SerializeField] private CellsPairsGameplay cellPairsGameplay;
    [SerializeField] private List<Color> colors;
    [SerializeField] private float topOffset;

    private List<GameObject> Cells = new List<GameObject>();
    int cellsAmount;
    private int difficultLevel;

    public GameObject squarePrefab;
    public float padding = 0.1f;
    private Camera mainCamera;

    private void Start()
    {
        mainCamera = Camera.main;
        difficultLevel = DataContainer.GetCellPairsDifficult();
        if (difficultLevel == 1)
        {
            SpawnSquares(3, 4);
        }
        else if (difficultLevel == 2)
        {
            SpawnSquares(3, 5);
        }
        else
        {
            SpawnSquares(3, 6);
```

```
        }
    }

    private void SpawnSquares(int squaresInRow, int squaresInColumn)
    {
        float screenWidth = mainCamera.orthographicSize * 2f * mainCamera.aspect;
        float screenHeight = mainCamera.orthographicSize * 2f;

        SpriteRenderer squareRenderer = squarePrefab.GetComponent<SpriteRenderer>();
        float squareWidth = squareRenderer.bounds.size.x;
        float squareHeight = squareRenderer.bounds.size.y;

        float totalPaddingX = (squaresInRow - 1) * padding;
        float totalPaddingY = (squaresInColumn - 1) * padding;

        float squareSizeX = (screenWidth - totalPaddingX) / squaresInRow;
        float squareSizeY = (screenHeight - totalPaddingY - topOffset) / squaresInColumn;

        float startY = screenHeight / 2f - squareSizeY / 2f - topOffset;

        float startX = -screenWidth / 2f + squareSizeX / 2f;

        for (int row = 0; row < squaresInColumn; row++)
        {
            for (int column = 0; column < squaresInRow; column++)
            {
                float xPos = startX + column * (squareSizeX + padding);
                float yPos = startY - row * (squareSizeY + padding);
                float zPos = -5f;

                GameObject temp = Instantiate(squarePrefab, new Vector3(xPos, yPos, zPos),
Quaternion.identity);
                temp.transform.localScale = new Vector3(squareSizeX, squareSizeY, 1f);
                Cells.Add(temp);
            }
        }

        cellsAmount = Cells.Count;
        cellPairsGameplay.CellsAreDone(difficultLevel, cellsAmount);
        RandomizeCells();
    }

    private void RandomizeCells()
    {
        for (int i = 0; i < cellsAmount / 2; i++)
        {
            Color col = colors[i];
            int id = Random.Range(0, Cells.Count);
            Cells[id].GetComponent<CellPairsController>().SetInnerColor(col);
            Cells.RemoveAt(id);

            id = Random.Range(0, Cells.Count);
            Cells[id].GetComponent<CellPairsController>().SetInnerColor(col);
            Cells.RemoveAt(id);
        }
    }
}
```

CellPairsGameplay.cs

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

public class CellsPairsGameplay : MonoBehaviour
{
    [SerializeField] GameObject GameOverScreen;
    [SerializeField] Text tryAmountText;
    private GameObject cell1, cell2;
    private int openedCellsAmount = 0;
    private int LockedCellPairsAmount = 0;
    private int targetLockedCellPairs;
    private int difficult;
    private int amountOfTries;

    public void CellsAreDone(int dif, int amount)
```

```
    {
        difficult = dif;
        amountOfTries = amount/2+dif;
        targetLockedCellPairs = amount;
        tryAmountText.text = "Кількість спроб: " + amountOfTries;
    }

    public void CellOpened(GameObject cell)
    {
        cell.GetComponent<CellPairsController>().SetCellOpenedState();

        if (openedCellsAmount == 0)
        {
            cell1 = cell;
            openedCellsAmount++;
        }
        else
        {
            cell2 = cell;
            if (cell1.GetComponent<CellPairsController>().GetCellColor() ==
cell2.GetComponent<CellPairsController>().GetCellColor())
            {
                cell1.GetComponent<CellPairsController>().LockCell(true);
                cell2.GetComponent<CellPairsController>().LockCell(true);

                LockedCellPairsAmount++;
                if (LockedCellPairsAmount >= targetLockedCellPairs/2)
                {
                    GameOver();
                }
            }
            else
            {
                amountOfTries--;
                tryAmountText.text = "Кількість спроб: " + amountOfTries;
                if (amountOfTries == 0)
                {
                    GameOver();
                }
                cell1.GetComponent<CellPairsController>().LockCell(false);
                cell2.GetComponent<CellPairsController>().LockCell(false);
            }
            openedCellsAmount = 0;
        }
    }

    private void GameOver()
    {
        DataContainer.SetCellPairsBest(openedCellsAmount*1000*difficult);
        GameOverScreen.SetActive(true);
    }
}
```