МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАНН

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Ігровий мобільний додаток "Combat dots"»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-52 Кузьменко Владислав Віталійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «__» _____2019 р.

Науковий керівник _____ ___к.т.н., доц., Федотова Н. А.__
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ _____Шифрін Д. М._____
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

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

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2019 р.

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

Кузьменку Владиславу Віталійовичу

1 Тема роботи *Ігровий мобільний додаток "Combat dots"*

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

затверджені наказом по університету від « 17 » травня 2019 р. № 0834-III

2 Строк подання студентом роботи «3» червня 2019 р.
3 Вхідні дані до роботи
Існуючі правила гри, моделі Android-додатків
4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
вступ, аналіз предметної області Android-додатків, моделювання процесу створення Android-додатку, моделювання варіантів використання Android-додатку, розробка ігрових об'єктів, розробка звукового супроводження, розробка програмного сегменту ігрової механіки, розробка скриптової імітації діяльності різнорівневого штучного інтелекту, висновки, список використаної літератури

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Зображення логотипів Android-додатків, що є аналогами «Combat Dots», зображення контекстної діаграми А-0 та її декомпозицій, діаграма варіантів використання Android-додатку, зображення сцен Android-додатка «Combat Dots», зображення стартових позицій для віртуального супротивника на різних рівнях складності.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Федотова Н.А.*		
Моделювання процесу створення ігрового Android-додатку «Combat Dots»	*Федотова Н.А.*		
Створення ігрового Android-додатку «Combat Dots»	*Федотова Н.А.*		

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Ознайомлення з поточним станом ігрових Android-додатків у Play Market	01.04.19-02.04.19	
2.	Ідентифікація мети та задач	03.04.19-03.04.19	
3.	Аналіз існуючих правил гри	04.04.19-09.04.19	
4.	Визначення апаратного забезпечення	10.04.19-10.04.19	
5.	Визначення програмного забезпечення	10.04.19-10.04.19	
6.	Реалізація методів основного функціоналу додатку	10.04.19-20.04.19	
7.	Реалізація методів ігрової механіки для двох режимів гри	20.04.19-03.05.19	
8.	Реалізація методів скриптової імітації діяльності різнорівневого штучного інтелекту.	29.05.19-16.05.19	
9.	Аналіз результатів	17.05.19-28.05.19	
10.	Оформлення документації	29.05.19-02.06.19	

Студент _____ Кузьменко В.В.
(підпис)

Керівник роботи _____ ктн. доц. Федотова Н.А..
(підпис)

РЕФЕРАТ

Тема роботи: «Ігровий мобільний додаток "Combat dots"».

Пояснювальна записка містить розділи: «Вступ», «Аналіз предметної області Android-додатків», «Постановка задачі та методи дослідження Android-додатків», «Проектування Android-додатку», «Розробка ігрового Android-додатка», «Висновок», «Список літератури», та три додатки. Вона включає 72 сторінки, 8 таблиць, 29 ілюстрацій, та 21 інтернет-джерело.

В розділі «Аналіз предметної області Android-додатків» був виконаний опис проекту створення Android-додатків, огляд існуючих аналогів Android-додатку, було обрано апаратне та програмне забезпечення та зроблено висновки щодо виконаної роботи.

В розділі «Постановка задачі та методи дослідження Android-додатків» було проаналізовано існуючі правила гри, обрано методи реалізації майбутнього Android-додатку та проведене планування робіт щодо проектування та реалізації майбутнього Android-додатку.

В розділі «Проектування Android-додатку» були змодельовані такі діаграми як: контекстна діаграма IDEF0 процесу створення ігрового Android-додатку та діаграма варіантів використання.

В розділі «Розробка ігрового Android-додатка» описується процес створення ігрового Android-додатку «Combat Dots», а саме: розробка файлів локалізації, розробка спрайтів, розробка звукового супроводження та розробка програмного сегменту додатку.

Результатом роботи над проектом є apk-файл готового ігрового Android-додатку «Combat Dots».

Ключові слова: мобільний ігровий Android-додаток, логіко-стратегічна гра, Unity3D, скриптова імітація діяльності різнорівневого штучного інтелекту.

ЗМІСТ

ВСТУП 7

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ANDROID-ДОДАТКІВ 10

1.1 Опис проекту створення Android-додатку 10

1.2 Огляд існуючих аналогів Android-додатку 11

1.3 Вибір програмного забезпечення для розробки Android-додатку 15

1.4 Висновки до розділу 1 18

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ ANDROID-ДОДАТКІВ 19

2.1 Правила гри Android-додатку 19

2.2 Вибір методів реалізації Android-додатку 20

2.3 Планування робіт проектування та реалізації Android-додатку 21

3 ПРОЕКТУВАННЯ ANDROID-ДОДАТКУ 22

3.1 Діаграми нотації IDEF0 по процесу створення Android-додатку 22

3.2 Діаграма варіантів використання Android-додатку 33

4 РОЗРОБКА ІГРОВОГО ANDROID-ДОДАТКА 36

4.1 Попередня підготовка перед початком розробки Android-додатка 36

4.2 Розробка сцен Android-додатка 38

4.3 Програмна реалізація Android-додатка 46

4.4 Готовий продукт 55

ВИСНОВОК 56

СПИСОК ЛІТЕРАТУРИ 57

ДОДАТОК А 59
ДОДАТОК Б. 63
ДОДАТОК С 71

ВСТУП

Метою даного проекту є створення логіко-стратегічного Android-додатка «Combat Dots» на платформі Android.

Об’єктом дослідження є процес розробки Android-додатка згідно створеного сценарію та існуючих правил.

Предметом дослідження є модель процесу реалізації режиму гри для двох гравців на одному смартфоні та реалізація одиночного режиму гри проти скриптової імітації діяльності різнорівневого штучного інтелекту.

Для досягнення поставленої мети необхідно реалізувати наступні задачі дослідження:

- дослідити предметну область, проаналізувати існуючі правила гри та сформувати технічне завдання;
- обрати оптимальні методи розробки і програмне забезпечення, та на їх базі реалізувати основний функціонал додатку та ігрове поле;
- реалізувати імітацію діяльності штучного інтелекту за допомогою скриптів-налаштувань для віртуального супротивника живому гравцю.

Для програмної реалізації ігрового поля було вирішено створити матрицю відповідної розмірності. Для реалізації захоплення точок вводиться поняття стану комірок. Серед основних станів, що будуть найчастіше використовуватись у програмній частині можна виділити:

- «neutral» – відповідає нічийній комірці;
- «first» – відповідає захопленій комірці першого гравця;
- «second» – відповідає захопленій комірці другого гравця;
- «maybe first» – відповідає можливості захоплення комірки першим гравцем;
- «maybe second» – відповідає можливості захоплення комірки другим гравцем.

Спочатку програмно реалізовується двовимірний квадратний масив стандартного ігрового поля розмірності 9x9. Далі необхідно визначити кожну можливу позицію рівну стану «neutral». Саме так програмно будуть позначені тимчасово нічийні комірки на ігровому полі, що відповідають нейтральній території та прозорому кольору у візуальній реалізації.

Для визначення можливості зробити хід у обрану комірку буде використовуватись динамічна функція, що шукає на полі точки конкретного гравця (того, чий зараз хід) та вираховує комірки, у які зараз є можливість зробити свій хід і динамічно присвоює їм програмне значення рівне стану «maybe first» або «maybe second» (в залежності від того чий зараз хід) та візуально відображає ці комірки кольором конкретного гравця з параметром непрозорості рівним 25% від початкового кольору.

Після того як гравець зробить свій хід, то обрана позиція у програмному сегменті видаляється з масиву можливих ходів, отримує значення відповідного гравця і присвоєне йому візуальне відображення, а масив можливих ходів втрачає своє кольорове забарвлення через те, що їх значення знову буде рівне стану «neutral» після виконання динамічної функції завершення та передачі ходу.

У Android-додатку буде присутня скриптова імітація діяльності різнорівневого штучного інтелекту для віртуального супротивника реальному гравцю у режимі одиночної гри. Кожен наступний рівень складності матиме більш складну формулу прорахунку обраного варіанту ходу ніж попередній, а також додаткові комірки на старті, щоб збільшити складність проходження одиночного режиму для реального гравця.

Після запуску додатка користувачу буде виведена на екран інформація про використаний у розробці рушій Unity3D, що відповідає вимогам використання безкоштовної версії даного рушія у створенні некомерційних проектів. Потім користувачу на екран виводиться сцена вибору мови: «Англійська», «Російська» та «Українська». Далі користувач буде переправлений у «Головне меню» гри. У «Головному меню» на вибір представлені п’ять кнопок-обробників подій: «Одиночна гра», «Гра на двох», «Налаштування», «Допомога» та «Вихід».

У обробнику події «Одиночна гра» буде представлена гра у двох можливих режимах проти імітованої за допомогою спеціально налаштованих скриптів діяльності п’ятирівневого штучного інтелекту.

У обробнику події «Гра на двох» буде представлена можливість грати проти реального опонента на одному телефоні у двох можливих режимах.

У обробнику події «Налаштування» буде представлена можливість змінити тему (темна та світла), змінити мову та керувати налаштуваннями звуку.

У обробнику події «Довідка» будуть наведені актуальні правила гри та ілюстрації до них.

У обробнику події «Вихід» буде представлена можливість завершити роботу з додатком.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ANDROID-ДОДАТКІВ

1.1 Опис проекту створення Android-додатку

З розвитком сучасних технологій, що дозволяють створювати персональні мобільні пристрої і різні гаджети, ринок корпорацій отримав потужний стимул до покращення власних мереж. Телефони грають важливу роль в повсякденній роботі: з їх допомогою зчитують файли, перевіряють електронну пошту та друкують документи за допомогою мережевого принтера. Станом на 2019 рік майже у кожної людини є свій смартфон. Близько 100% вірогідність того, що його мобільна операційна система це Android або iOS. Це пояснюється тим, що Google та Apple займають тотально-домінуючу позицію на ринку мобільних телефонів. Якщо проаналізувати додатки, що найчастіше використовуються людьми, то це або додатки соціальних мереж або ігрові додатки. Через це на ринку поступово сформувався окремий сегмент – мобільні додатки [1].

Специфіка даного сегмента в тому, що розробка Android-додатків повинна проводитися з урахуванням особливостей мобільних пристроїв: відмінностями інтерфейсу, розміром екрану, сенсорним управлінням. Якщо інтерфейс розрахований на тактильне управління, то необхідно збільшити розміри відображуваних елементів, адже якщо вони будуть занадто малі, то натиснути на них буде важко.

Завдяки тому, що ринок пропонує величезне різноманіття версій мобільних пристроїв, кожна з них вимагає певного виду платформи для роботи з додатками.

Ігри для мобільних телефонів були широко поширені задовго до того, як iPhone і Android стали боротися за цей сегмент ринку. Однак поява нових пристроїв і ідеологій помітно змінили картину. Мобільні ігри перестали бути прерогативою дітей. Тепер навіть дорослі можуть розслабитися у спокійних іграх або навпаки напрягти свій мозок у логічних та стратегічних. У газетах друкуються історії про

успішних розробників, що заробили свій капітал на ринку мобільних ігор. Відомі компанії прагнуть залучити на свій бік кращих розробників.

1.2 Огляд існуючих аналогів Android-додатку

У світі існує дуже велика кількість IT-компаній, цільова діяльність яких спрямована на розроблення сучасних та високоякісних мобільних додатків, суттєву частину яких складають логічні або стратегічні ігри. Серед них є такі «гіганти» у розробці мобільних ігор як Nevasoft, Alawar, HeroCraft та Rovio Stars.

1.2.1 Android-додаток «ТОЧКИ ОНЛАЙН»

Згідно опису у Play Market – це логічна настільна гра на клітинному папері. Замість листочка і ручки у грі використовується ваш смартфон (рис.1.1).

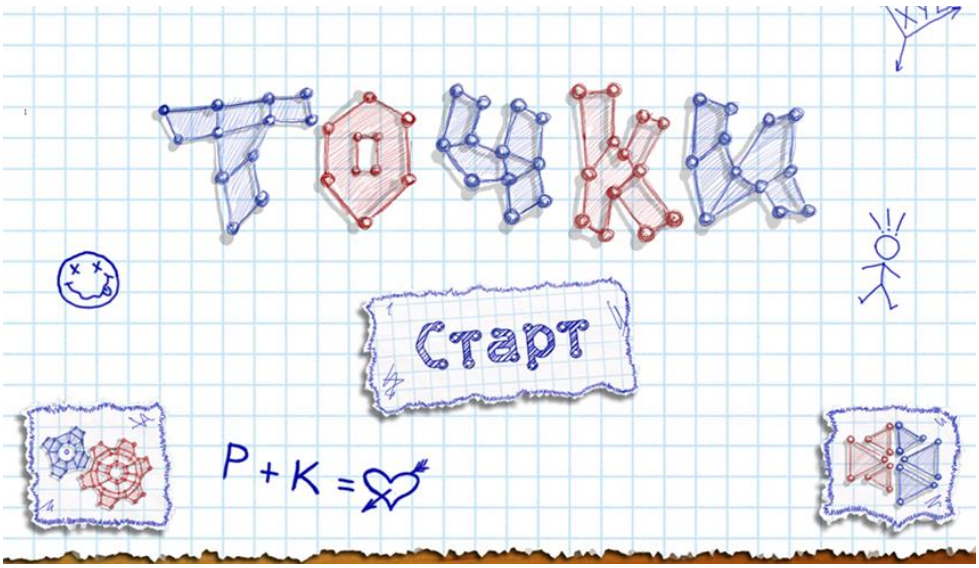


Рисунок 1.1 – Android-додаток «ТОЧКИ ОНЛАЙН»

Є можливість грати із суперниками з усього світу завдяки он-лайн мультиплееру. Також є режим для тренування в іграх зі штучним інтелектом. Мета гри - оточити якомога більше точок противника. Для цього суперники по черзі

ставлять точки на перетині ліній листа в клітку, кожен своїм кольором. Оточення повинно бути побудовано так, щоб відстань між точками становила трохи більше однієї клітини - по горизонталі, вертикалі або по діагоналі. Партія закінчується, коли не залишилося вільних місць, за взаємною згодою гравців, або коли один з гравців здався.

У грі відсутня локалізація. Із доступних мов лише Російська. Також у грі відсутній редактор рівнів та самі рівні гри [4].

1.2.2 Android-додаток «СУДОКУ»

Судоку – це логічна гра-задача. Вашою метою є розрахунок наявності чисел від 1 до 9 у кожному рядку, стовпчику та квадрату 3х3. Згідно опису у Play Market цей Android-додаток має рівні складності, що підходять і для початківців і для досвідчених гравців (рис.1.2).

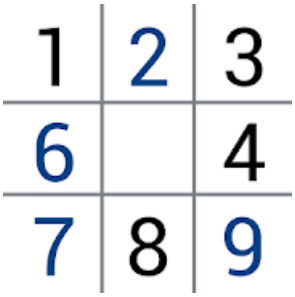


Рисунок 1.2 – Android-додаток «СУДОКУ»

У грі відсутня локалізація, а із доступних мов лише Російська. Також у грі відсутній редактор для створення власних судоку та немає можливості грати вдвох [6].

1.2.3 Android-додаток «SEA BATTLE 2»

Морський бій 2 – це звичайний «Морський бій», але з новими можливостями і розширеним арсеналом. Згідно статистики Play Market мільйони людей по всьому світу грають в цю гру. У вашому розпорядженні кораблі, літаки, підводні човни, міни, радари і не тільки. Для перемоги необхідно розставити на полі бою свої кораблі та

наносити удари по полю ворога. Серед нововведень ви можете використати арсенал, намагаючись потопити кораблі супротивника. У грі наявний мультиплеєр, тому ви можете змагатися з суперниками з усього світу через Інтернет в режимі реального часу (рис.1.3).

Рисунок 1.3 – Android-додаток «SEA BATTLE 2»

Ця гра є кроссплатформенною, тобто користувачі Персональних комп'ютерів можуть змагатися з користувачами смартфонів. Морський бій - гра з графікою у зошитовому стилі. У грі доступні дві мови: російська та англійська [5].

1.2.4 Android-додаток «1LINE – ONE-STROKE PUZZLE GAME»

Головоломка для тренування ваших розумових здібностей, в якій ви малюєте одну ламану лінію і намагаєтесь повторити заданий малюнок. Ця гра створена для посилення інтелектуальних здібностей гравця (рис.1.4).

Рисунок 1.4 – Android-додаток «1LINE – ONE-STROKE PUZZLE GAME»

На перший погляд гра здається легкою, але після проходження усе більшої кількості рівнів, вони досягнуть такої складності, що гравцю доведеться витрачати години на проходження одного рівня. У грі відсутня локалізація. Єдина із доступних мов – це англійська. Немає можливості створювати нові рівні. Також відсутня можливість грати проти іншого гравця та штучний інтелект у рівнях [8].

1.2.5 Android-додаток «САПЕР НА РУССКОМ»

«Сапер на русском» для Android це класичний сапер (Minesweeper). Особливості даної версії «Сапера» – це можливість налаштовувати розмір клітинок і поля для зручності і під ваш розмір вашої руки. Гра автоматично налаштовується під розширення вашого екрану. У грі присутні 3 рівня складності. Також ви можете зберегти гру і продовжити у будь-який зручний для вас час (рис.1.5).

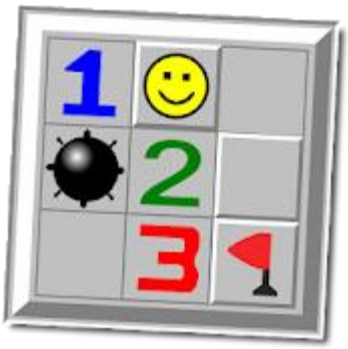


Рисунок 1.5 – Android-додаток «САПЕР НА РУССКОМ»

У грі відсутня локалізація. Серед доступних мов лише російська. Зовсім немає звукового супроводження, лише вібровідклик [7].

Проаналізувавши аналоги було складено порівняльну таблицю продуктів цих компаній. Порівняння відбувалося за такими критеріями:

1. Наявність головного меню,
2. Наявність різних рівнів для гри,
3. Наявність скриптів, що імітують різнорівневий штучний інтелект,
4. Можливість редагувати ігрове поле,
5. Можливість грати вдвох,

6. Локалізація,
7. Звукове супроводження.

Таблиця 1 – Порівняння аналогів

	Точки онлайн	Судоку	Sea Battle 2	1Line – one-stroke puzzle game	Сапер на русском	**Combat Dots (мій додаток)**
1	+	-	+	+	-	+
2	-	+	+	+	+	+
3	+	-	+	-	-	+
4	-	-	-	-	-	-
5	+	-	+	-	-	+
6	-	-	+	-	-	+
7	+	+	+	+	-	+

1.3 Вибір програмного забезпечення для розробки Android-додатку

Для програмної реалізації мобільного додатка існує велика кількість мов програмування. Для додатків, що позиціонують себе як ігрові вірним рішенням буде використовувати рушії гри такі як Unity3D або UE4. Для створення об’єктів, що будуть використовуватися у додатках доцільно буде використовувати такі засоби як Adobe Photoshop або Adobe Illustrator для двомірної растрової і векторної графіки відповідно та 3ds Max для створення повноцінних тривимірних об’єктів.

Unity3D – багатоплатформовий інструмент, що використовується для створення дво- та тривимірних додатків. Основою рушія гри є мови програмування

C# та JavaScript. За допомогою Drag&Drop інтерфейсу, на якому базується рушій Unity можна у режимі реального часу програмувати та тестувати створюване програмне забезпечення.

C# – крос-платформна об'єктно-орієнтована мова програмування. Для реалізації скриптів у рушії Unity3D має використовувати наслідування від класу функцій Unity – «Mono Behavior».

JavaScript – слабко типізована динамічна мова програмування сценаріїв. У більшості використовується у веб-додатках для реалізації дизайну та бекенду (Node js). Після адаптації до рушію Unity версію цієї мови програмування було названо UnityScript. Є менш популярною ніж C# у створенні скриптів.

Adobe Photoshop – графічний редактор від компанії Adobe. Представляє собою найзручніший засіб для роботи з елементами растрової графіки.

Adobe Illustrator – графічний редактор від компанії Adobe. Представляє собою засіб для маніпуляції та редагування векторних об’єктів.

3ds Max – графічний редактор тривимірних об’єктів. Має використовуватись при створенні тривимірних додатків.

Unreal Engine 4 – ігровий рушій, що базується на створенні тривимірних ігрових додатків. Основою рушія гри є мова програмування C++. Є можливість портувати гру у якості веб-додатка, що буде працювати на сучасному HTML5 та спеціально налаштованому під UE4 JavaScript.

Аналізуючи підходяще для використання програмне забезпечення було складено порівняльні таблиці можливостей схожих за своєю суттю засобів. Порівняння рушіїв гри відбувалося за такими критеріями:

1. Основна мова програмування,
2. Наявність зручного інтерфейсу для створення, редагування та тестування,
3. Наявність безкоштовної версії,
4. Крос-платформенність,

Таблиця 2 – Порівняння рушіїв гри

	Unity3D	Unreal Engine 4
1	C# / JavaScript	C++ / JavaScript
2	Drag&Drop інтерфейс	Огляд сцени без можливості редагування або тестування одночасно
3	При доході менше 200000$ на рік	Присутня безкоштовна версія
4	Будь-яка з платформ	Windows, Mac OS, Android, iOS

Порівняння графічних редакторів відбувалося за такими критеріями:

1. Спеціалізація,
2. Можливість роботи з різними форматами не власної спеціалізації,
3. Наявність безкоштовної версії,
4. Крос-платформенність отриманих об’єктів,

Таблиця 3 – Порівняння графічних редакторів

	Adobe Photoshop	Adobe Illustrator	3ds Max
1	Растрова графіка	Векторна графіка	Тривимірні об’єкти
2	Сучасні версії дозволяють працювати з тривимірними об’єктами	Робота з тривимірними об’єктами лише у рамках векторного двомірного зображення	Відсутня робота з двомірною графікою
3	Безкоштовна версія на 1 місяць	Безкоштовна версія на 1 місяць	Студентська ліцензія
4	Windows, Mac OS, Linux, Android, iOS	Windows, Mac OS, Linux, Android, iOS	Windows, Mac OS, Linux

Проект буде створено за допомогою ігрового рушія Unity3D на мові програмування C#. Даний рушій було обрано через те, що в ньому використовується контейнерно-компонентний підхід на базисі функціонального програмування, в рамках якого розробник створює об'єкти-контейнери і до них додає різні об’єкти-компоненти та інші об'єкти-контейнери.

Іншою перевагою Unity3d є кросс-платформеність, тобто одне і теж програмне забезпечення (Desktop-версія або мобільний додаток), що було створене на рушію Unity з мінімальними змінами може бути перенесене на різні платформи. Це величезний плюс, який скорочує час на розробку проекту в кілька разів.

Для створення растрових графічних зображень, що будуть портуватися у рушій доцільно використовувати Adobe Photoshop.

1.4 Висновки до розділу 1

Мобільні додатки представляють собою майбутнє сфери інформаційних технологій. З кожним роком науково-технічний прогрес не перестає дивувати. Потужність та обчислювальна здатність сучасних мобільних пристроїв наздоганяє та іноді навіть переганяє комп’ютерну техніку та ноутбуки, а загальна компактність не залишає жодного шансу нерухомим ПК.

У 2018-2019 роках найбільші IT-компанії вже портували свою флагманську продукцію у версії для мобільних додатків. Adobe та Autodesk вже оголосили монополію у сфері мобільного дизайну та проектування. Тепер займатися фото та відео-монтажем і моделюванням можна і у мобільних версіях відповідних програм.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ ANDROID-ДОДАТКІВ

2.1. Правила гри Android-додатку

1. У якості ігрового поля буде використана обмежена ділянка 9x9 (81 доступна позиція)

2. Перший гравець починає з однією стартовою точкою, що знаходиться у центрі його власної першої лінії нападу. Другий гравець починає з двома точками, що знаходяться зверху та знизу від центральної точки його власної першої лінії нападу задля компенсації втрати ініціативності його ходів.

3. Коштовна важливість ліній для кожного гравця визначається її віддаленістю від стартової лінії конкретного гравця. Так як поле 9x9, тому і ліній підрахунку фінальних балів також дев’ять. Наприклад лінія, з якої стартує перший гравець принесе йому по 1 балу за кожну захоплену позицію, але ця ж сама лінія принесе по 9 балів за кожну захоплену позицію другому гравцю, так як вона віддалена від його стартової на 7 ліній (перша лінія стартова, дев’ята лінія остання та сім між ними).

4. Захоплювати позицію можна лише в тому разі, якщо вона ще не захоплена другим гравцем та є прилеглою до вже захопленої вами.

5. Якщо якісь незахоплені позиції повністю ізольовані одним гравцем від іншого, тобто він ніяк не зможе їх захопити, то вони автоматично захоплюються не витрачаючи хід гравцем, що ізолював їх.

6. Перемога віддається тому гравцю, який набрав більше балів після розподілу усіх 81 позицій або у випадку рівної кількості балів перемога віддається другому гравцю.

7. У режимі "Тактика" на початку гри випадково генеруються перешкоди, що перекривають шлях та не дають захоплювати комірки. Перешкоди є дзеркальними, а тому збалансованими.

8. Також в цьому режимі віртуальний супротивник буде ігнорувати будь-які перешкоди. Це збільшує складність гри та потребує від вас оптимальних стратегічних рішень.

9. Перемога віддається так само як і у стандартному режимі, але гра може завершитись достроково, якщо на полі залишаться позиції, що неможливо захопити через перешкоди.

Також було сформоване та затверджене технічне завдання, яке було розміщене у додатку А.

2.2. Вибір методів реалізації Android-додатку

Основою гри є квадратна матриця (двовимірний масив) розмірністю 9x9.

Методом прорахунку можливого ходу буде наявність вже захопленої сусідньої клітинки поряд з місцем нападу. Програмно – це порівняння посилань двох комірок шляхом додавання або віднімання одиниці та перевірка їх стану, що має бути рівним значенню «neutral».

Методом закінчення гри буде перевірка наповнення матриці на відповідність стану комірок «first» або «second».

Метод автоматичного заповнення ізольованих комірок буде створений за допомогою логіки нечітких множин.

Нечітка множина – спроба математично формалізувати нечітку інформацію з метою її використати при побудові математичної моделі складної системи.

Спочатку відбувається аналіз всіх нічийних комірок на даному ігровому полі. Після цього циклічно видаляються всі нічийні комірки, що прилягають до комірок

супротивника або до тих нічийних комірок, що також прилягають до комірок супротивника. Дія циклу завершується після того, як кількість проаналізованих комірок стане статичною, тобто перестане зменшуватись. Саме ці комірки будуть автоматично захоплені.

Результати проекту можуть бути використані у діяльності різних App-Dev компаній. Особливо їх потребують компанії, чиї реалізації проектів базуються на матричній математиці та функціонально-динамічній обробці багатовимірних масивів інформації при функціональному програмуванні.

2.3 Планування робіт проектування та реалізації Android-додатку

Завдяки сформованій меті та визначенню переліка задач, було підібрано необхідний інструментарій для виконання майбутніх дій. Наступним кроком є планування робіт по виконанню цього проекту. Були розроблені діаграма Ганта для конкретного розподілення часових та людських ресурсів згідно між запланованими роботами, структурна декомпозиція робіт (WBS) для відстеження ланцюгів виконуваних робіт, організаційна структура проекту (OBS) для відстеження виконавців робіт зазначених у WBS та матриця відповідальності для узгодження структури виконання робіт, які наведені у Додатку Б. Також були прораховані можливості виникнення ризиків та описані методи для їх уникнення або усунення.

3. ПРОЕКТУВАННЯ ANDROID-ДОДАТКУ

Після проведення аналізу предметної області, визначення мети та задач проекту, формування списку вимог до функціоналу та вибору засобів реалізації необхідно перейти до проектування Android-додатку.

Проектування – це процес планування та вирішення задач для створення раціональної програмної реалізації.

3.1 Діаграми нотації IDEF0 по процесу створення Android-додатку

Процес проектування Android-додатку доцільно розпочати з побудови контекстної діаграми А-0. Така діаграма містить у собі стисле та доцільне пояснення кожного кроку реалізації, що дає змогу оцінити правильність обраного шляху виконання. Головними елементами даної діаграми є: вхідні дані, вихідні дані, управління та механізми. Зазвичай вказують точку зору (виконавця робіт), предметну область та ціль виконання.

Після проведеного аналізу відносно головних елементів контекстної діаграми «Створення Android-додатка "Combat Dots"», було сформовано перелік даних:

- Вхідні дані: завдання на створення гри.
- Вихідні дані: реліз apk-версії готової гри.
- Управління: вимоги до гри, обмеження ПЗ, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Контекстна діаграма А-0, що була розроблена за допомогою програмного забезпечення Business Process Design Tool на основі цих даних представлена на рис.3.1.

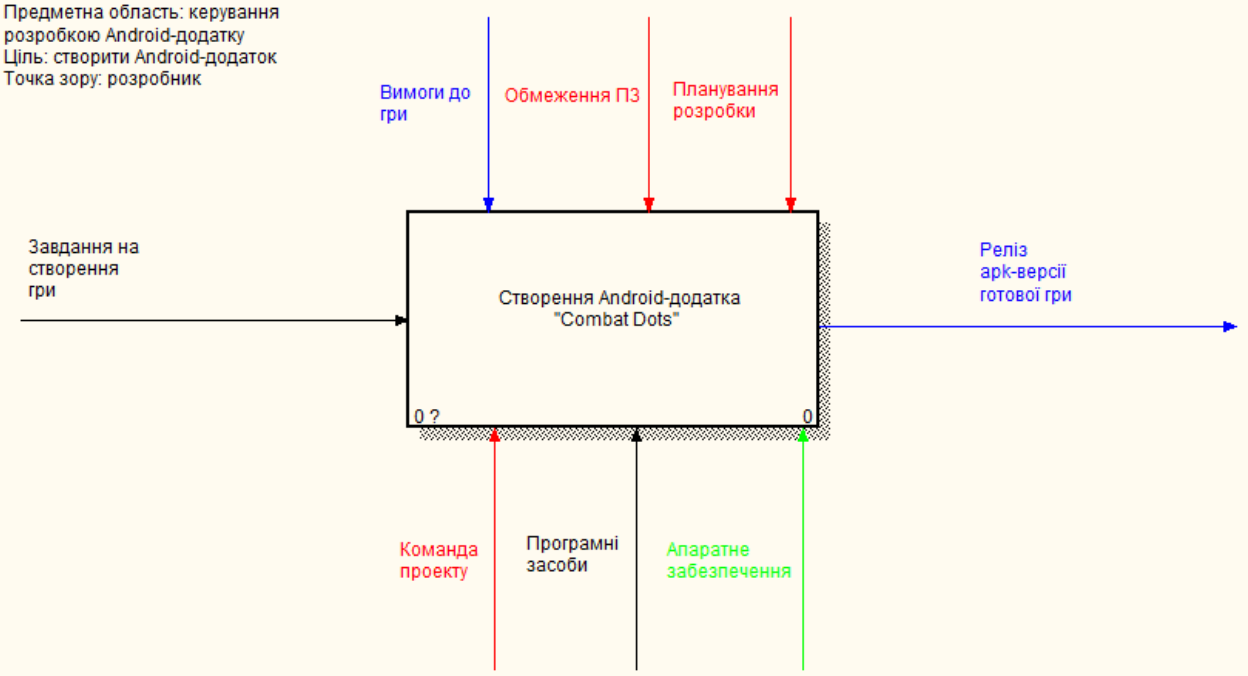


Рисунок 3.1 – Контекстна діаграма A-0

Оскільки така діаграма містить у собі лише загальну інформацію щодо системи, то виникає необхідність у виконанні декомпозиції.

Декомпозиція – це розподіл складного об’єкту, системи чи задачі на складові частини або елементи.

Даний процес дозволить детальніше розглянути логіку послідовності виконання робіт для реалізації запланованого продукту.

Діаграма A-0 була розбита на три підрівні, а саме: реалізація ігрового поля, реалізація імітації дій штучного інтелекту за допомогою скриптів і тестування та відладка. Це обумовлено тим, що процес створення Android-додатку поділяється на три етапи:

- Перший етап: реалізація ігрової механіки, а саме реалізація ігрової зони, реалізація комірок, реалізація пошуку можливих ходів та реалізація методу автоматичного захоплення.
- Другий етап: реалізація імітації гри штучного інтелекту п’яти різних рівнів за допомогою скриптів.
- Третій етап: процес тестування гри та процес відладки у випадку знайдених помилок.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: завдання на створення гри.
- Вихідні дані: файли з налаштованою механікою гри, модель імітації штучного інтелекту.
- Управління: вимоги до гри, обмеження ПЗ, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри, модель імітації штучного інтелекту.
- Вихідні дані: файли альфа-версії гри.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі третього етапу були сформовані такі дані:

- Вхідні дані: файли альфа-версії гри.
- Вихідні дані: реліз apk-версії готової гри.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд діаграми А-0 після декомпозиції на 3 етапи наведений на рис.3.2

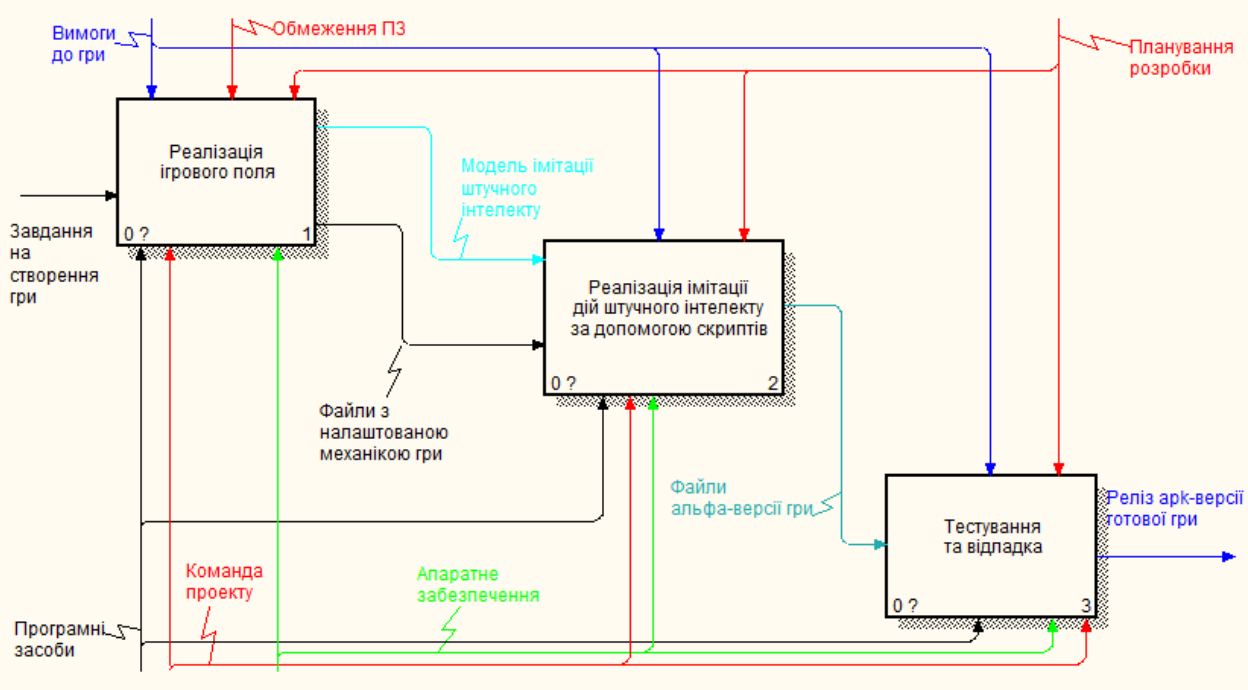


Рисунок 3.2 – Декомпозиція контекстної діаграми A-0

Після декомпозиції контекстної діаграми на три процеси була виконана декомпозиція кожного з них на певний перелік робіт за умови, що результат попередньої є вхідними даними для наступних.

Процес «Реалізація ігрового поля» був розбитий на п’ять підрівнів, а саме: генерація та реалізація ігрової зони, генерація та реалізація ігрових точок, реалізація методу пошуку можливих ходів, реалізація методу автоматичного захоплення та реалізація методів режиму «Тактика». Це обумовлено тим, що процес створення ігрового поля поділяється на п’ять етапів:

- Перший етап: на базисі ігрової механіки реалізувати обмежену ігрову зону на ігровому полі.
- Другий етап: реалізація комірок та можливості їх захоплення на вже створеній ігровій зоні.
- Третій етап: реалізація методу пошуку можливих ходів на ігровому полі для допомоги гравцям та для обчислень штучного інтелекту.
- Четвертий етап: реалізація методу автоматичного захоплення ігрових комірок на ігровому полі.

- П’ятий етап: реалізація методів режиму гри «Тактика», у якому присутні перешкоди на ігровому полі.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: завдання на створення гри.
- Вихідні дані: файли ігрової зони.
- Управління: вимоги до гри, обмеження ПЗ, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: файли ігрової зони.
- Вихідні дані: файли комірок.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі третього етапу були сформовані такі дані:

- Вхідні дані: файли комірок.
- Вихідні дані: файли методів пошуку можливих ходів.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі четвертого етапу були сформовані такі дані:

- Вхідні дані: файли методів пошуку можливих ходів.
- Вихідні дані: файли методу автозахоплення.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі п’ятого етапу були сформовані такі дані:

- Вхідні дані: файли методу автозахоплення.
- Вихідні дані: файли з налаштованою механікою гри та модель імітації штучного інтелекту.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд процесу «Реалізація ігрового поля» після декомпозиції на 5 етапів наведений на рис.3.3

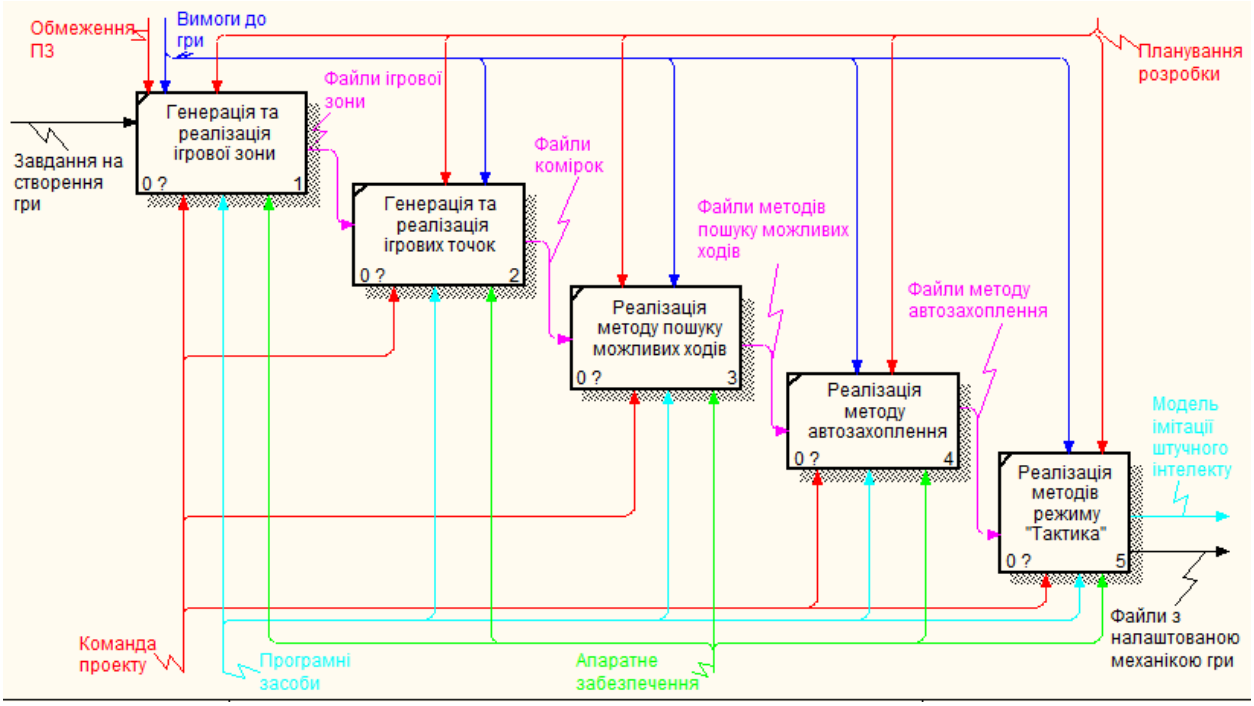


Рисунок 3.3 – Декомпозиція процесу «Реалізація ігрового поля»

Процес «Реалізація імітації дій штучного інтелекту за допомогою скриптів» був розбитий на два підрівні, а саме: реалізація рівнів складності і реалізація збірки штучного інтелекту та реалізація сцени для вибору одного із них. Це обумовлено тим, що реалізації імітації дій штучного інтелекту за допомогою скриптів поділяється на два етапи:

- Перший етап: реалізуються п’ять рівнів імітації діяльності штучного інтелекту.
- Другий етап: реалізація збірки всіх рівнів складності та реалізація сцени для цього.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри та модель імітації штучного інтелекту.

- Вихідні дані: файли простого рівня складності, файли середнього рівня складності, файли легкого рівня складності, файли важкого рівня складності та файли дуже важкого рівня складності.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: файли простого рівня складності, файли середнього рівня складності, файли легкого рівня складності, файли важкого рівня складності та файли дуже важкого рівня складності.
- Вихідні дані: файли альфа-версії гри.
- Управління: планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд процесу «Реалізація імітації дій штучного інтелекту за допомогою скриптів» після декомпозиції на 2 етапи наведений на рис.3.4

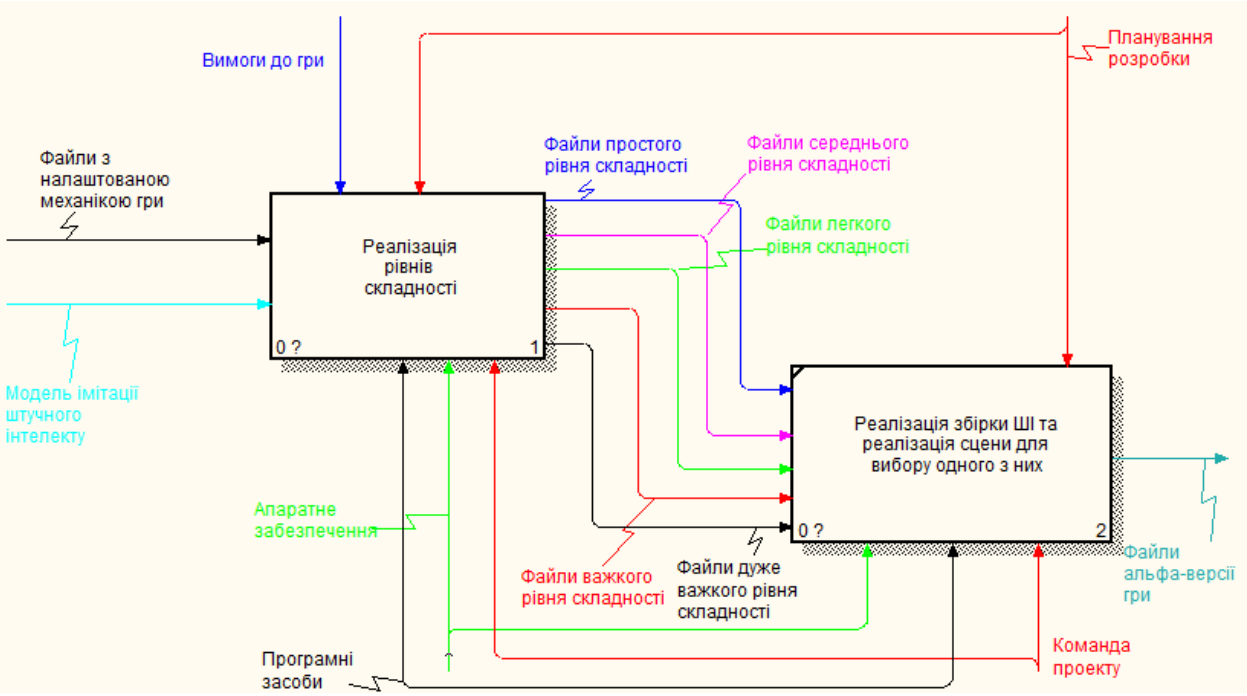


Рисунок 3.4 – Декомпозиція процесу «Реалізація штучного інтелекту»

Процес «Реалізація рівнів складності» був розбитий на п’ять підрівнів, а саме: реалізація простого рівня складності, реалізація легкого рівня складності, реалізація середнього рівня складності, реалізація важкого рівня складності, реалізація дуже важкого рівня складності. Це обумовлено тим, що реалізація рівнів складності поділяється на п’ять етапів:

- Перший етап: реалізується простий рівень складності.
- Другий етап: реалізується легкий рівень складності, що використовує напрацювання простого рівня складності.
- Третій етап: реалізується середній рівень складності, що використовує напрацювання попередніх рівнів складності рівнів складності та опирається на модель імітації діяльності штучного інтелекту.
- Четвертий етап: реалізується важкий рівень складності, що використовує напрацювання середнього рівня складності та модернізує їх.
- П’ятий етап: реалізується дуже важкий рівень складності, що використовує напрацювання важкого рівня складності та модернізує їх.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри.
- Вихідні дані: файли простого рівня складності.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри та файли простого рівня складності.
- Вихідні дані: файли легкого рівня складності.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі третього етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою, файли легкого рівня складності та модель імітації штучного інтелекту.

- Вихідні дані: файли середнього рівня складності.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі четвертого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри, файли середнього рівня складності та модель імітації штучного інтелекту.
- Вихідні дані: файли важкого рівня.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі п’ятого етапу були сформовані такі дані:

- Вхідні дані: файли з налаштованою механікою гри, файли важкого рівня складності та модель імітації штучного інтелекту.
- Вихідні дані: файли дуже важкого рівня складності.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд процесу «Реалізація рівнів складності» після декомпозиції на 5 етапів наведений на рис.3.5

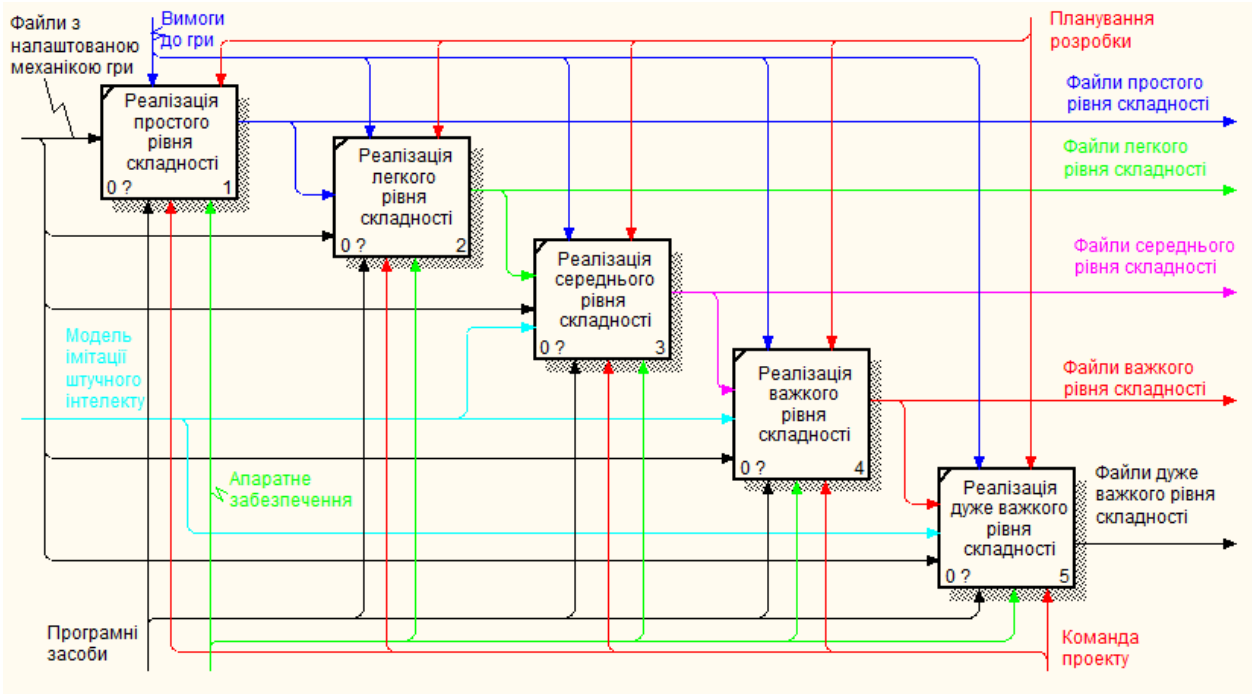


Рисунок 3.5 – Декомпозиція процесу «Реалізація рівнів складності»

Процес «Тестування та відладка» був розбитий на два підрівні, а саме: тестування та відладка. Це обумовлено тим, що процес пошуку помилок та їх виправлення поділяється на два етапи:

- Перший етап: пошук помилок.
- Другий етап: виправлення знайдених помилок, якщо вони є.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: файли альфа-версії гри.
- Вихідні дані: звіт про помилки, реліз apk-версії гри.
- Управління: вимоги до гри, планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: звіт про помилки.
- Вихідні дані: виправлений код.
- Управління: планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд процесу «Тестування та відладка» після декомпозиції на 2 етапи наведений на рис.3.6

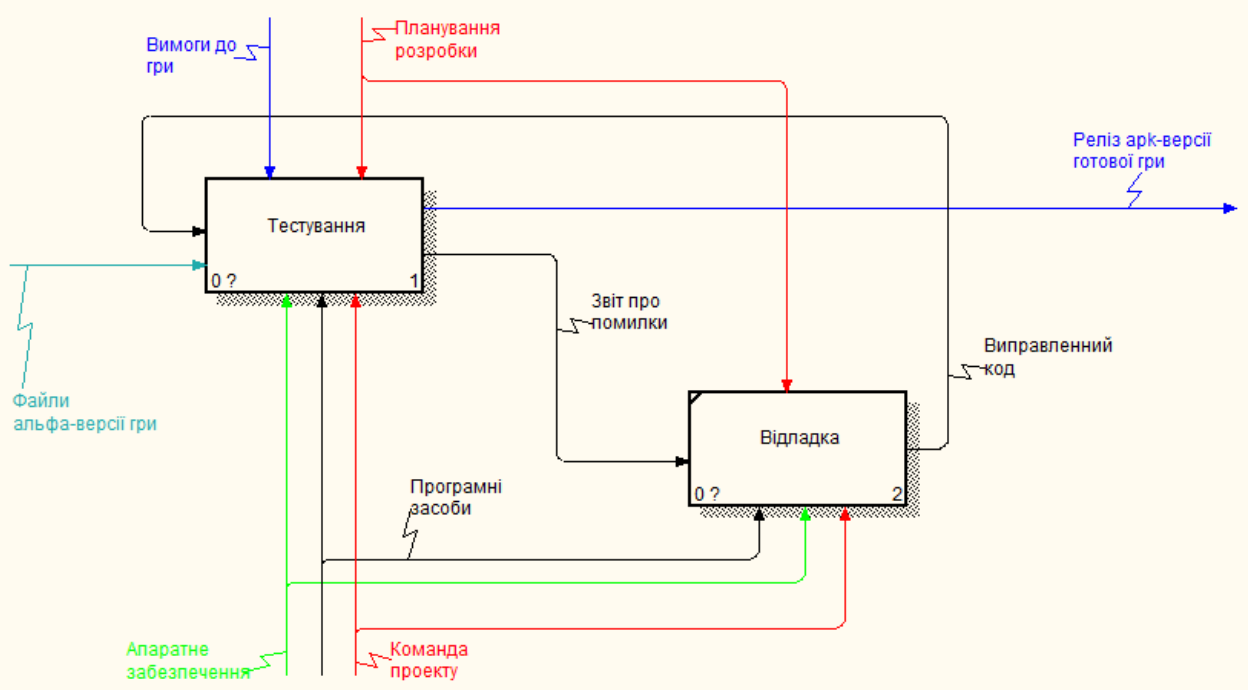


Рисунок 3.6 – Декомпозиція процесу «Тестування та відладка»

Процес «Тестування» був розбитий на три підрівні, а саме: тестування функціоналу продукту, тестування режиму гри на двох та тестування одиночного режиму гри. Це обумовлено тим, що процес тестування поділяється на два етапи:

- Перший етап: перевірка на правильність роботи функціоналу гри.
- Другий етап: перевірка на правильність роботи режиму гри на двох.
- Третій етап: перевірка на правильність роботи одиночного режиму гри.

При описі першого етапу були сформовані такі дані:

- Вхідні дані: виправлений код, файли альфа-версії гри.
- Вихідні дані: відмітка у тест-кейсі про проходження тестування.
- Управління: планування розробки.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі другого етапу були сформовані такі дані:

- Вхідні дані: відмітка у тест-кейсі про проходження тестування.

- Вихідні дані: відмітка у тест-кейсі про проходження тестування.
- Управління: планування розробки, вимоги до гри.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

При описі третього етапу були сформовані такі дані:

- Вхідні дані: відмітка у тест-кейсі про проходження тестування.
- Вихідні дані: звіт про помилки, реліз apk-версії гри.
- Управління: планування розробки, вимоги до гри.
- Механізми: команда проекту, програмні засоби, апаратне забезпечення.

Вигляд процесу «Тестування» після декомпозиції на 3 етапи наведений на рис.3.6

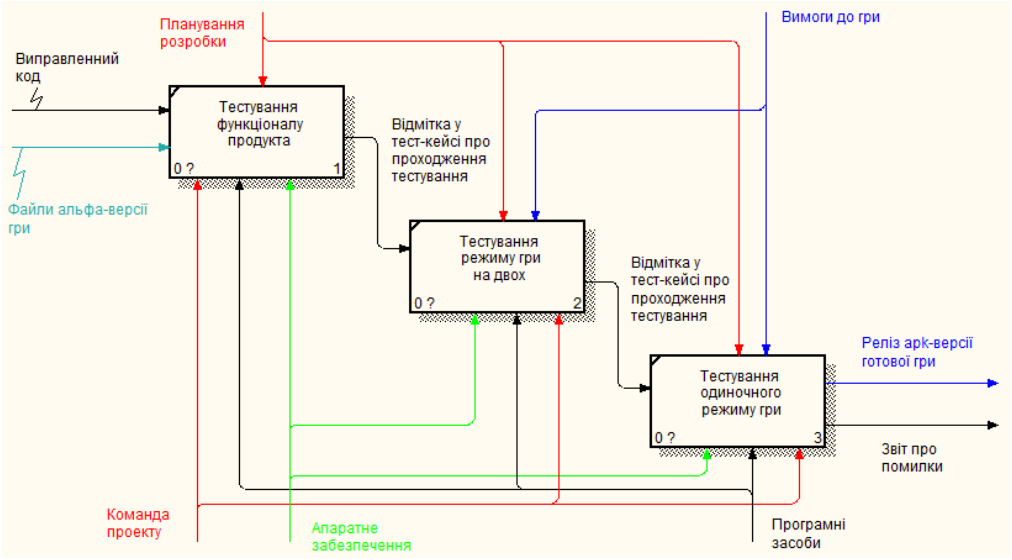


Рисунок 3.7 – Декомпозиція процесу «Тестування»

3.2 Діаграма варіантів використання Android-додатку

Наступним етапом проектування продукту є розробка діаграми варіантів використання (Use Case Diagram). Ця діаграма відображає залежність між актором

(групою акторів) та варіантами, як продукт може використовуватись. У ній можна відстежити, які саме дії актору призведуть до виконання одних чи інших процесів.

Для розробки діаграми Use Case був визначений один актор – Гравець. Саме він обирає те, що буде відбуватись у грі. Інші актори відсутні, тому що гра не передбачає багатокористувальницького режиму гри через мережу або інтернет та не передбачає зв'язок запитів з базою даних для завантаження або прийняття результатів гри.

Після того, як був визначений актор, що буде взаємодіяти з системою, необхідно сформувати перелік варіантів використання. Вони відповідають вимогам зазначеним у Додатку А.

Варіанти використання ігрового Android-додатку:

- Змінити налаштувань додатку;
- Отримати довідку згідно правил гри;
- Завершення роботи додатку;
- Проходження одиночної гри;
- Проходження гри на двох.

В свою чергу деякі з можливих варіантів використання мають власне продовження.

Варіанти використання «Проходження гри на двох» та «Проходження одиночної гри» мають два процеси-розвитку:

- Класичний режим гри;
- Режим гри «Тактика».

Кожен з цих варіантів призведе до процесу «Показ результату гри».

Варіант використання «Зміна налаштувань додатку» має три варіанти використання:

- Обрати мову;
- Змінити налаштування звуку;
- Змінити тему.

На основі сформованих даних про актора та всіх можливих варіантів використання ігрового Android-додатку була розроблена діаграма варіантів використання (рис.3.8).

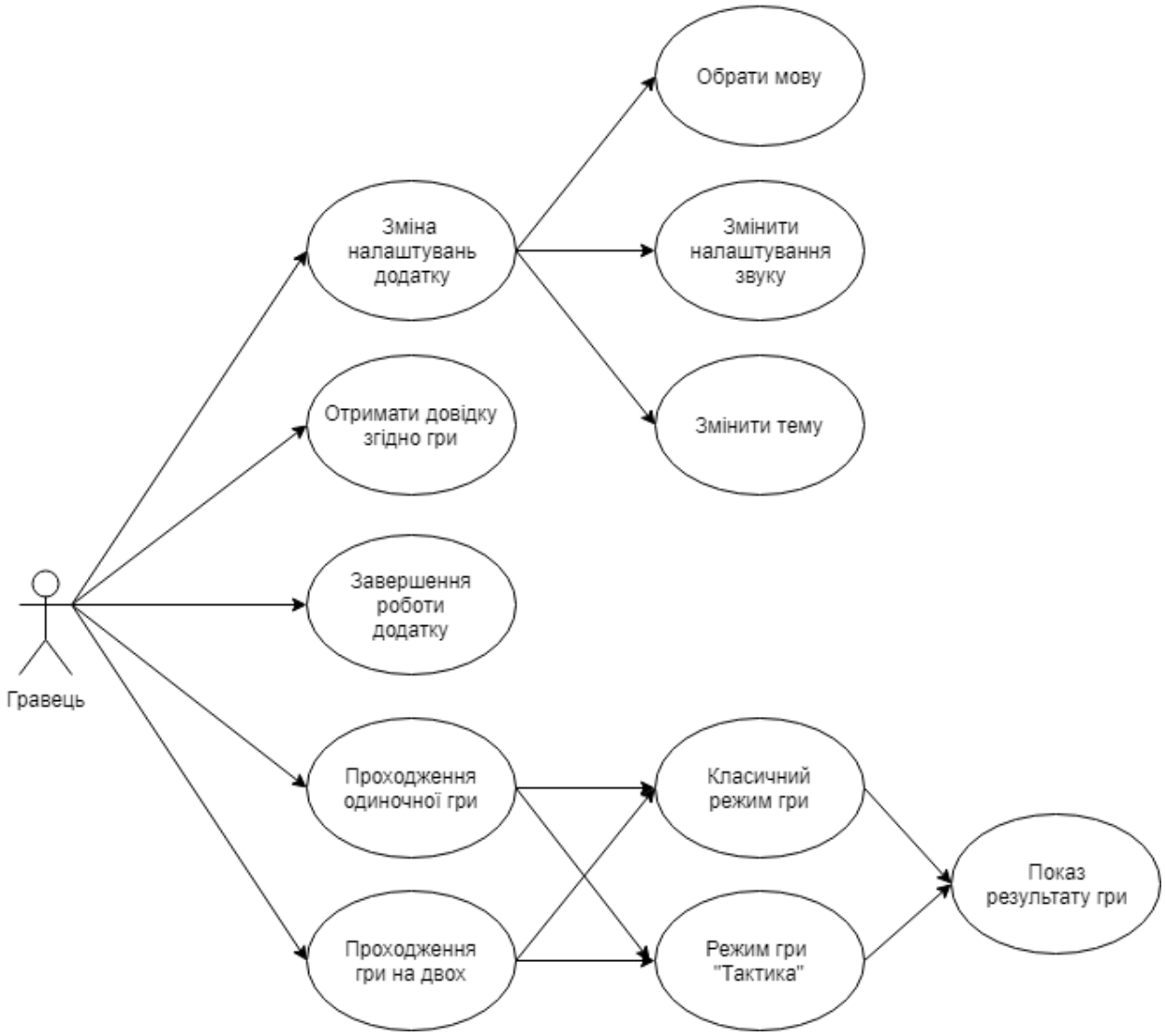


Рисунок 3.8 – Діаграма варіантів використання Android-додатку

4 РОЗРОБКА ІГРОВОГО ANDROID-ДОДАТКА

4.1 Попередня підготовка перед початком розробки Android-додатка

4.1.1 Розробка файлів локалізації

Перед початком розробки самого Android-додатку необхідно підготувати всі компоненти, що будуть використовуватись при розробці або при збірці проекту. Одним із таких компонентів є файли локалізації. Вони представляють собою масиви даних, що містять у собі слова, словосполучення або навіть речення, що використовуються у користувальницькому інтерфейсі. Зазвичай на кожну мову відводиться по одному окремому файлу в якому і будуть міститися абсолютно всі рядкові, що відповідають обраній мові.

У моєму Android-додатку будуть присутні файли локалізації для трьох мов: англійська, російська та українська. Ці файли є ідентичними за своєю структурою і відрізняються лише перекладом на ту чи іншу мову. Структура такого файлу представляє собою два масиви: масив правил та масив заголовків. У масиві правил містяться актуальні правила, у масиві заголовків містяться заголовки кожної з створених у майбутньому сцен.

4.1.2 Розробка спрайтів та текстур

Для покращення візуального сприйняття майбутнього проекту необхідно розробити доцільні текстури та спрайти. Текстури будуть слугувати для бекграунду кожної зі сцен, а спрайти для відображення функціональних подій або у якості кнопок користувальницького інтерфейсу. Кольорова гамма текстури та деяких спрайтів буде налаштовуватись не тільки під світлу, а й під темну тему.

Для світлого ігрового поля буде використовуватись зображення білого квадратного паперу у клітинку розміром 10 на 10 клітинок. Для темного ігрового поля

буде використовуватись зображення світлого ігрового поля, але після опрацювання модифікатором «Інверсія» у програмі Adobe Photoshop CS6. Також зображення світлого ігрового поля буде використовуватись і у якості постійної текстури після деяких модернізацій. Для текстурування бекграунду світлої теми передбачено зменшення яскравості та контрасту на 25%, а для темної теми на 50%. Після опрацювання дана текстура методом 10-кратного множення розміщується на задньому плані кожної сцени без зміни розміру.

Для майбутніх ігрових об’єктів створюються колоподібні фігури чотирьох кольорів: червоного та синього для світлої теми і жовтого та блакитного для темної теми. Також кожна з новоутворених фігур копіюється та отримує значення непрозорості 50%. Це необхідно для функціоналу відображення можливих ходів.

Для відображення перешкод на ігровому полі створюються горизонтальні та вертикальні прямокутники чорного кольору для світлої теми та білого кольору для темної теми.

Для елементів, що будуть слугувати елементами користувальницького інтерфейсу створюються унікальні зображення, які мають підходити за змістом до тієї події, на функціонал якої вони будуть перенаправляти.

4.1.3 Робота зі звуковим супроводженням Android-додатку

Для Android-додатку передбачений лише один звук – це звучання бульбашки, що лопається. Даний звуковий ефект є короткотривалим і буде використовуватись у якості звукового супроводження виконаного ходу на ігровому полі.

4.2 Розробка сцен Android-додатка

4.2.1 Розробка стартової сцени

Першою сценою, що буде зустрічати гравця – буде сцена вибору мови гри (рис.4.1). На цій сцені використовуються 3 спрайти-кнопки, що представляють собою сферичні прапори трьох країн: України, Росії та Сполучених Штатів Америки. Прапор США використовується для позначення англійської мови тому, що американський варіант англійської мови більш поширений у світі ніж британський.

Кожен з прапорів буде встановлювати у Android-додатку вибрану мову та завантажувати рядкові літерали з відповідного файлу локалізації. Після цього гравець буде переправлений у сцену головного меню.

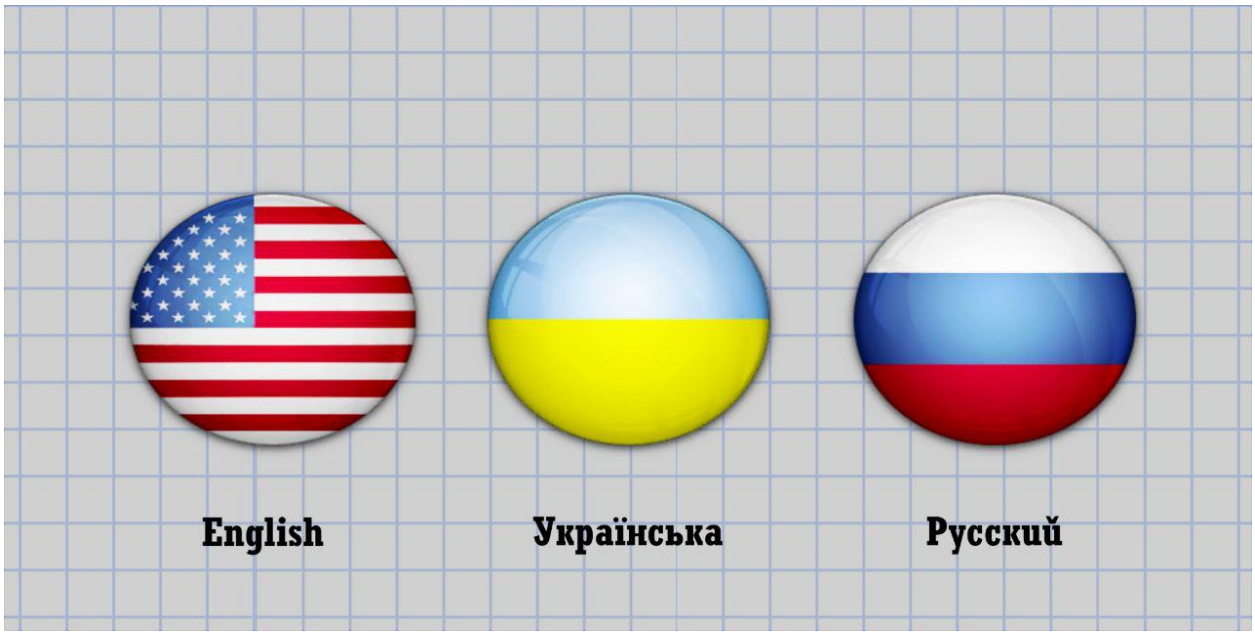


Рисунок 4.1 – Стартова сцена

4.2.2 Розробка сцени головного меню

Після вибору мови гри гравець перенаправляється у головне меню (рис.4.2). На цій сцені розміщуються п’ять спрайтів-кнопок та п’ять елементів тексту. Кожен спрайт після події-натискання повинен буде перенаправляти гравця на іншу сцену:

спрайти одиночної гри та гри на двох перенаправлятимуть у сцену вибору режиму гри, спрайт налаштувань перенаправлятиме на сцену налаштувань гри, спрайт довідки перенаправлятиме у сцену довідки, спрайт виходу дозволить закінчити роботу з додатком.

Чотири елементи тексту розміщуються під відповідними спрайтами та є спочатку порожніми. Один елемент тексту розміщується зверху і є виконує роль заголовка. Після завантаження сцени у текстові елементи буде завантажуватись рядкове оформлення згідно вибраної мови.

Рисунок 4.2 – Сцена головного меню

4.2.3 Розробка сцени Довідки

Після натискання на спрайт довідки у сцені головного меню гравець перенаправляється у сцену Довідки (рис.4.3). На даній сцені розміщується 2 елементи тексту та 6 спрайтів з яких 3 спрайти будуть виконувати функцію кнопок, а 3 спрайти будуть виконувати функцію простих зображень.

Один елемент тексту виконує роль заголовка сцени, другий є великою текстовою областю, в яку будуть завантажуватись цілі речення, що містять у собі правила.

Спрайти-кнопки на даній сцені виконують роль стрілок. Дві стрілки відповідають за перегортання вперед-назад сторінки правил. Одна стрілка відповідає за переправлення у головне меню.

Статичні спрайти на цій сцені є частиною декількох сторінок правил тому і будуть з’являтись та зникати при перегортанні правил гри.

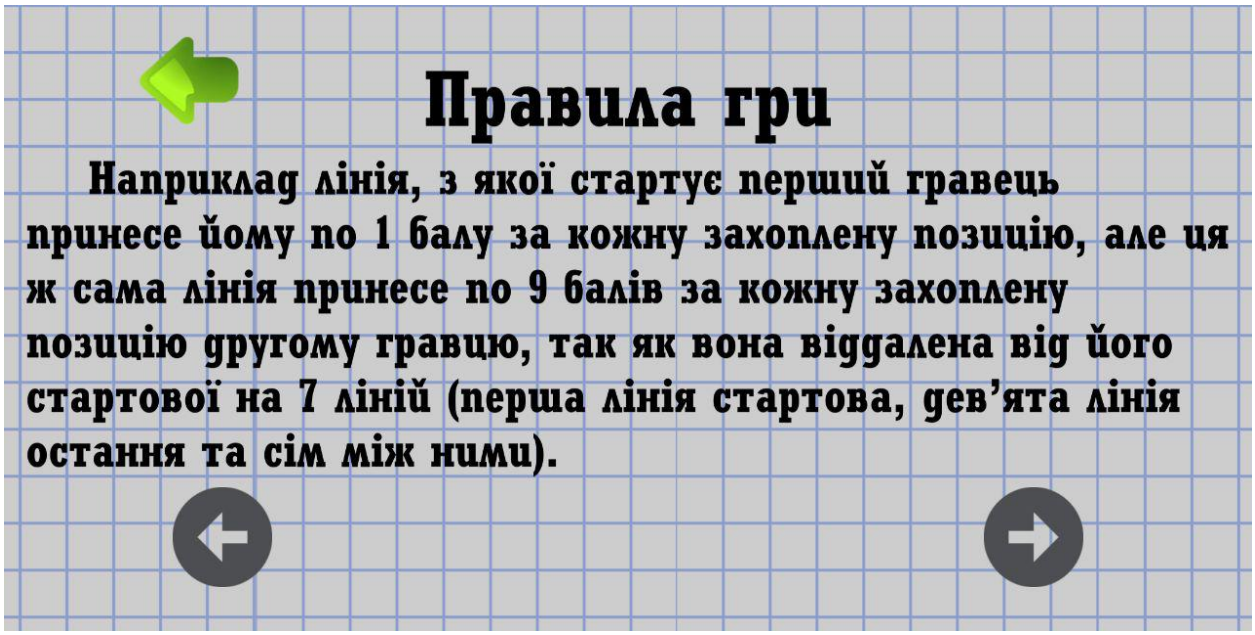


Рисунок 4.3 – Сцена довідки

4.2.4 Розробка сцени вибору режиму гри

Після вибору одиночної або багатокористувальницької гри гравець буде перенаправлений у сцену вибору режиму гри (рис.4.4). Дана сцена містить у собі два спрайти-кнопки та три текстових об’єкти. Натиснення на кнопку повинно визначати в який саме режим буде грати гравець. Одна з кнопок визначить за гравцем стандартний режим гри, інша – режим «Тактика». Один з текстових об’єктів виконує роль заголовка сцени, інші два є підписом про призначення спрайтів.

Рисунок 4.4 – Сцена вибору режиму гри

4.2.5 Розробка сцени вибору рівня складності

У разі якщо гравець обрав одиночну гру у головному меню та зробив свій вибір стосовно режиму гри, то він буде переправлений у сцену вибору рівня складності (рис.4.5). На цій сцені знаходяться 6 функціональних спрайтів-кнопок та 5 текстових об’єктів.

Один зі спрайтів – це стрілка повернення у головне меню, інші спрайти це кнопки вибору рівня складності гри. Текстові об’єкти виконують роль підписів біля відповідних зображень, що відповідають різним рівням складності гри.

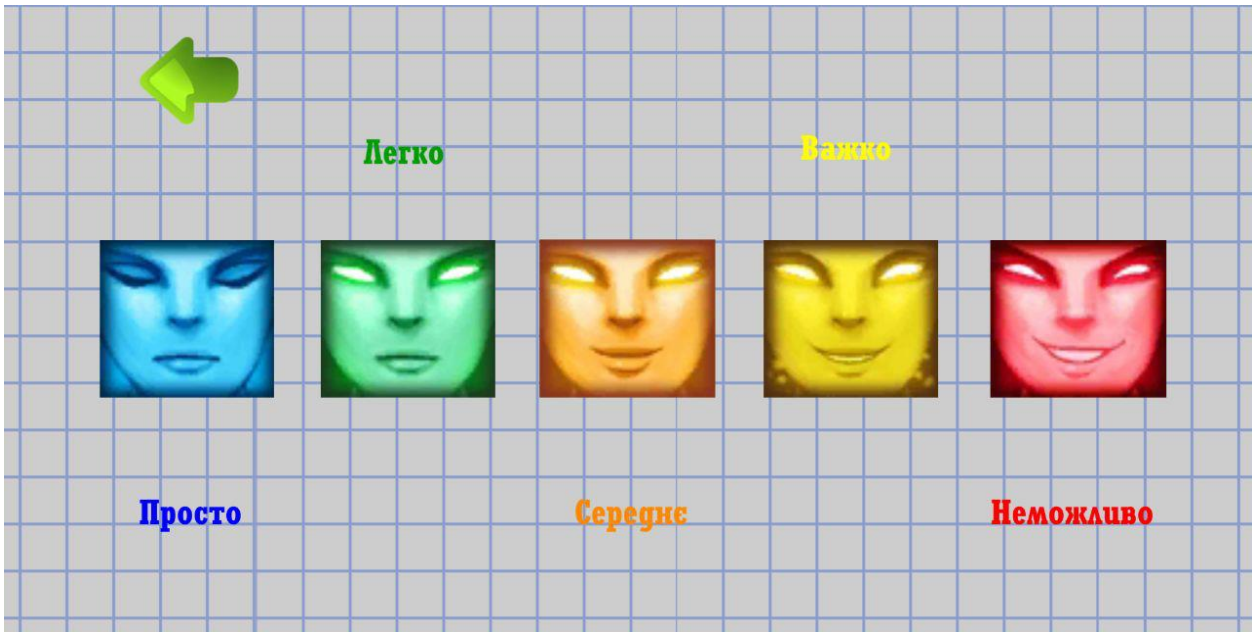


Рисунок 4.5 – Сцена вибору рівня складності

4.2.6 Розробка сцени налаштувань

На сцену налаштувань (рис.4.6) можна потрапити якщо вибрати відповідну кнопку у головному меню. Дана сцена дозволяє змінювати тему, мову та вмикати або вимикати звук. Для виконання функціоналу що описаний вище було прийняте рішення додати дев’ять спрайтів-кнопок та одну панель користувальницького інтерфейсу.

За функцію зміни параметрів звуку відповідатимуть два спрайти, що змінюють один-одного після кліку.

За функцію зміни теми відповідають зменшені зображення ігрового поля. У випадку обраної світлої теми у налаштуваннях буде зображене мініатюрне поле темної теми і навпаки. Дані зображення змінюють один-одного після кліку так само як і спрайти налаштувань звуку.

За функцію зміни мови відповідає спрайт-кнопка, після кліку на яку з’являтиметься панель на якій будуть доступні всі три мови. Після вибору однієї з мов панель зникає. Сцена з відкритою панеллю мов зображена на рисунку 4.7.

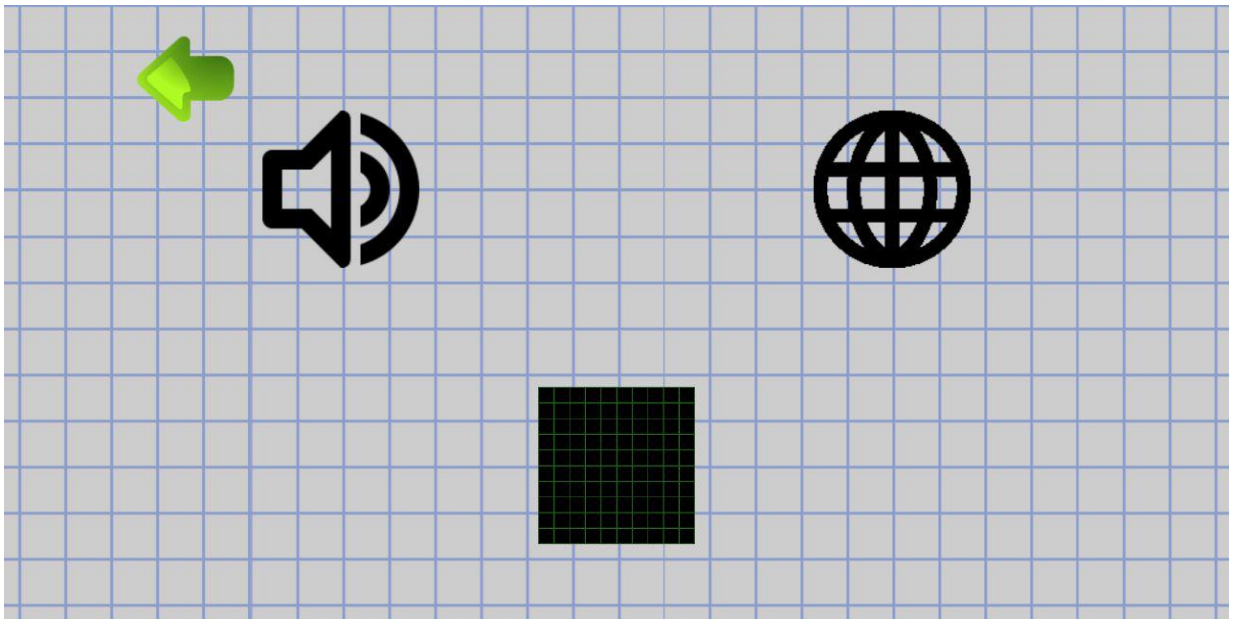


Рисунок 4.6 – Сцена налаштувань

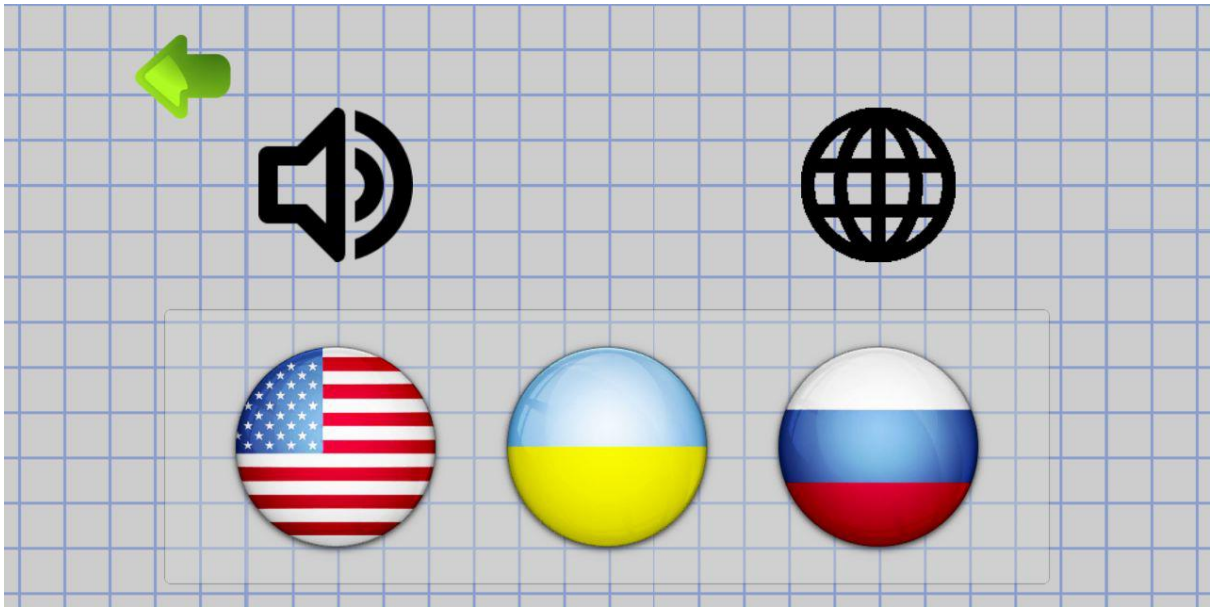


Рисунок 4.7 – Сцена налаштувань з панеллю мов

4.2.7 Розробка сцени ігрового поля

На сцену ігрового поля необхідно додати 16 спрайтів, 3 об’єкти тексту та одну панель користувальницького інтерфейсу. Один спрайт-кнопка виконує роль стрілки, що дозволяє повернутися до головного меню. Чотирнадцять спрайтів задіяні у візуальному відображенні ігрової зони: вісім спрайтів, що відповідають за відображення точок (чотири кольори стандартні та чотири кольори з непрозорістю

50%), чотири спрайти відповідають за відображення перешкод (два спрайти для світлої теми та два спрайти для темної), два спрайти представляють собою цілу ігрову зону (один варіант для світлої теми та один для темної).

Два текстових об’єкти відображають кількість очок, що має кожний гравець на даний момент. Кожен з них перефарбовується у теперішній колір відповідного гравця. Сцена ігрового поля у початковому стані з перешкодами зазначена на рисунку 4.8.

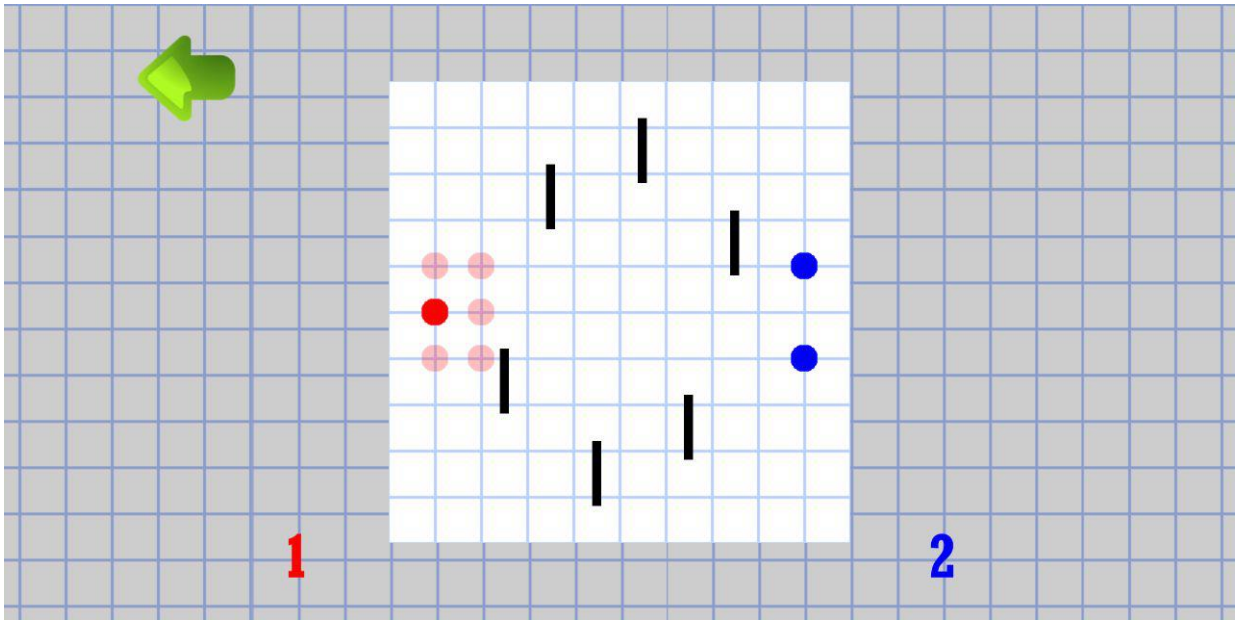


Рисунок 4.8 – Сцена ігрового поля

Панель користувальницького інтерфейсу виконує роль післяматчевого результату. Вона містить у собі один спрайт та один текстовий об’єкт в якому зазначається гравець, що переміг. Сцена ігрового поля з показаною післяматчевою панеллю зображена на рисунку 4.9.

Рисунок 4.9 – Сцена ігрового поля з післяматчевою панеллю

4.2.8 Сцена вибору кількості перешкод

У разі якщо гравець на сцені вибору режиму гри обрав режим «Тактика», то він буде переправлений на сцену вибору кількості перешкод для цього режиму (рис.4.10). На даній сцені міститься 3 спрайти-кнопки та один текстовий об’єкт що виконує роль заголовка.

Рисунок 4.10 – Сцена вибору кількості перешкод

4.3 Програмна реалізація Android-додатка

4.3.1 Скрипт Sound.cs

Даний скрипт є класом Sound, що містить у собі дані про звуковий файл, а саме: назву, кількість повторів відтворення, гучність та подачу. Об’єкти класу Sound використовуються у скрипті AudioManager.cs. Виконує роль класу, а тому не пов’язується з жодним об’єктом на сценах.

4.3.2 Скрипт AudioManager.cs

Даний скрипт визначає кількість усіх звуків, що були створені за допомогою класу Sound та дозволяє викликати їх у інших скриптах відтворюючи звукове супроводження. Використовується у сценах одиночної гри та гри на двох, так як лише у цих сценах присутнє звукове супроводження. Пов’язується з зображеннями комірок, що мають параметр непрозорості 50%.

4.3.3 Скрипт soundChange.cs

Даний скрипт відповідає за вмикання або вимкнення звуку у Android-додатку. Використовується у сцені налаштувань та пов’язується з зображеннями, що відповідають за налаштування звуку. Результат виконання цього скрипту є глобальним для всього додатку.

4.3.4 Скрипт backToMenu.cs

Даний скрипт дозволяє повернутись у головне меню. Виконання цього скрипту додається до всіх спрайтів, що повинні повертати гравця у головне меню. Використовується у сценах налаштувань, довідки, вибору рівня складності, одичної гри та гри на двох. Пов’язується з зображенням, що повинно повертати гравця у головне меню.

4.3.5 Скрипти uaRules.cs, ruRules.cs та enRules.cs

Дані скрипти містять у собі два масиви локалізованих до відповідної мови рядкових літералів та слугують для завантаження необхідних рядків на вибраній мові до сцен Android-додатку. Ці скрипти додаються на всі сцени Android-додатка та пов’язуються з головною камерою.

4.3.6 Скрипт helpArrows.cs

Даний скрипт призначений для роботи зі сценою довідки. В ньому відбувається завантаження інформації про вибрану гравцем мову. Після цього на сцену з файлів локалізації завантажуються правила. Пов’язується з зображеннями, що відповідають за перегортання сторінок правил і зчитує події-кліки з них.

4.3.7 Скрипт btnChange.cs

Даний скрипт керує поняттям «стану» скрипта helpArrows.cs. Він слідкує за тим, яке правило в даний момент відображене на сцені і передає це значення у головний скрипт. Якщо стан сцени правил рівний лівій або правій кінцевій частині сторінок, то завдяки цьому блокуються та зникають кнопки перегортання у відповідну сторону. Використовується на сцені довідки і повязується з кнопками, що відповідають за перегортання сторінок правил.

4.3.8 Скрипт btnScale.cs

Даний скрипт використовується для збільшення на 15% розміру кнопок при натисненні на них. Також скрипт зменшує розмір кнопки до початкового стану, якщо натискання припинилось. Використовується на початковій сцені, сцені довідки, сцені налаштувань, сцені вибору рівня складності та сценах ігрового поля. Пов’язується з об’єктами, що необхідно виділяти при події кліку на них.

4.3.9 Скрипт cameraColor.cs

Даний скрипт виконує зміну кольору бекграунду камери та текстуру бекграудуну сцен відповідно до обраної теми. Використовується на камерах усіх сцен.

4.3.10 Скрипт mainStart.cs

Даний скрипт виконує першочерговий вибір мови у додатку. Використовується у першій сцені і присвоює додатку мову в залежності від того, на яку кнопку натисне гравець.

4.3.11 Скрипт settTheme.cs

Даний скрипт виконує перевірку теперішньої та зміну на вибрану гравцем тему. Використовується на всіх сценах додатку.

4.3.12 Скрипт modeMenu.cs

В цьому скрипті так само як і у helpArrows.cs спочатку завантажується інформація про обрану мову, а після цього з файлів локалізації завантажуються необхідні рядкові літерали. Скрипт використовується у сцені вибору режиму гри.

4.3.13 Скрипт levelsText.cs

Даний скрипт завантажує інформацію про вибрану мову, потім отримує рядкові літерали з файлів локалізації і відображає їх на сцені вибору рівня складності.

4.3.14 Скрипт langMenu.cs

В цьому скрипті так само як і у helpArrows.cs спочатку завантажується інформація про обрану мову, а після цього з файлів локалізації завантажуються необхідні рядкові літерали. Скрипт використовується у сцені головного меню.

4.3.15 Скрипт modeCount.cs

Даний скрипт завантажує рядкові літерали з файлів локалізації згідно вибраної гравцем мови. Використовується на сцені вибору кількості перешкод для режиму «Тактика».

4.3.16 Скрипт diffSet.cs

Даний скрипт є обробником подій для кнопок рівнів складності скриптової імітації діяльності штучного інтелекту та використовується на сцені вибору рівня складності. Після цього завантажує сцену одиночної гри.

4.3.17 Скрипти langChange.cs та langSet.cs

Дані скрипти використовуються у якості обробників подій на сцені довідки для зміни теперішньої мови на іншу для всього Android-додатку. Пов’язуються з додатковою панеллю мов.

4.3.18 Скрипт fontTheme.cs

Даний скрипт відповідає за колір шрифту текстового об’єкту, що відповідає за кількість очок кожного гравця. Колір залежить від обраної теми та конкретного гравця. Використовуються на сценах одиночної гри та гри на двох.

4.3.19 Скрипт themeChange.cs

Даний скрипт відповідає за переключення теми у Android-додатку. Використовується у сцені налаштувань та пов’язується з зображеннями, що відповідають за зміну теми. Результат виконання цього скрипту є глобальним для всього додатку.

4.3.20 Скрипти transMenu.cs, transMode.cs, transModeCount.cs, transModeMulti.cs та transModeMultiCount.cs

Дані скрипти є сприптами-сполученнями, що пов’язують між собою декілька сцен, тобто кожен зі скриптів забезпечує перехід з однієї сцени на іншу.

4.3.21 Скрипт gameSingle.cs

Даний скрипт відповідає за всю механіку гри проти скриптової імітації різнорівневого штучного інтелекту у двох ігрових режимах. Використовується у сцені одиночної гри. Пов’язується з ігровим полем.

Простий рівень складності

Віртуальний гравець простого рівня складності кожен раз буде обирати випадковий хід з усіх доступних йому. Стартові позиції для віртуального гравця цього рівня складності є стандартними (рис.4.11).

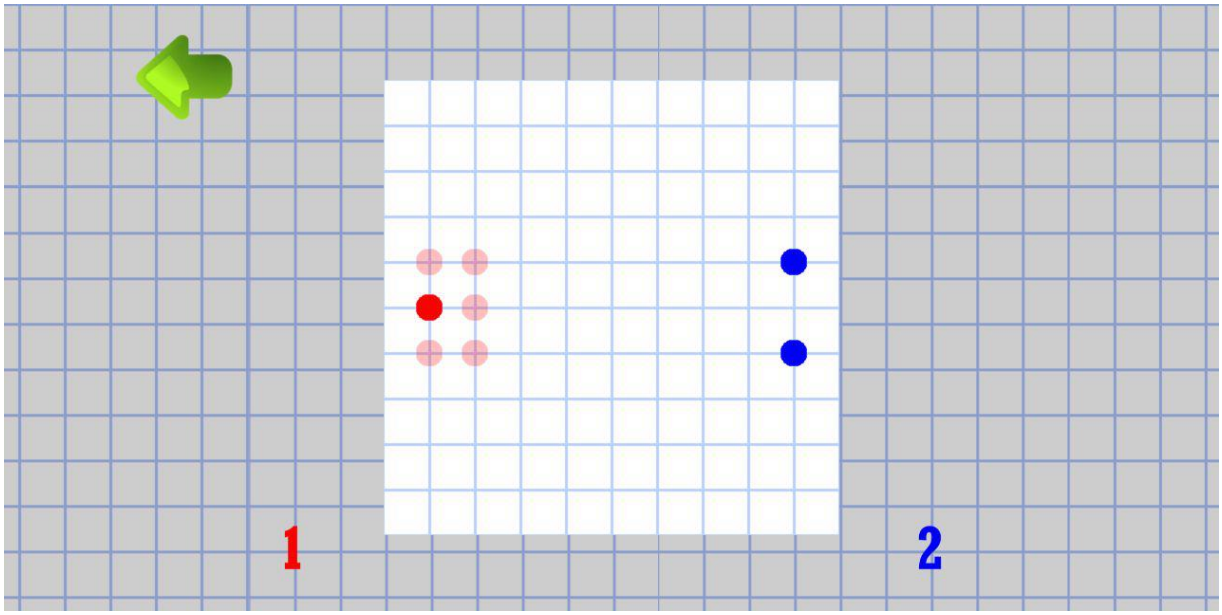


Рисунок 4.11 – Сцена з початковими комірками віртуального гравця простого рівня складності

Легкий рівень складності

Віртуальний гравець легкого рівня складності обирає два ходи:

- Обирає випадковий хід з усіх доступних для нього;
- Обирає хід, що принесе найбільшу кількість балів в даний момент.

Після цього обирає випадково один із цих двох ходів. Починає гру з однією додатковою коміркою на старті (рис.4.12).

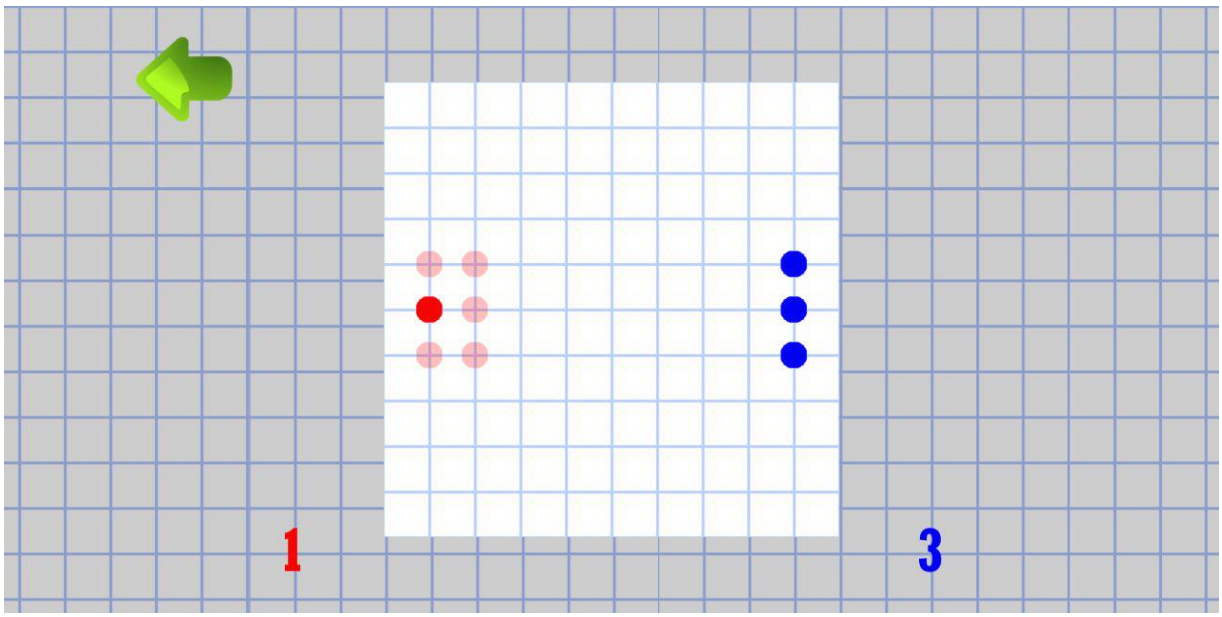


Рисунок 4.12 – Сцена з початковими комірками віртуального гравця легкого рівня складності

Середній рівень складності

Віртуальний гравець середнього рівня складності спочатку аналізує усі доступні йому можливі ходи і додає їх у окремий список. Після цього почергово програмно будує віртуальні ігрові поля та на кожному окремому полі додає можливий хід як вже зроблений. Після цього проводить аналіз кожного віртуального ігрового поля на функцію автозахоплення позицій і записує поточну кількість балів, що має віртуальний гравець в даний момент. Далі аналізуються всі можливі ходи віртуального гравця на кожному віртуальному ігровому полі. Після цього будується ще одне поле віртуальності, на якому проводиться той самий аналіз для прорахунку ситуації на ігровому полі через один хід. Після цього необхідно опрацювати список всіх записаних можливих ходів на звичайному ігровому полі. Спочатку аналізується позиція на її місце розташування перед трьома вже захопленими цим гравцем або на рекурсивне розташування перед трьома позиціями, що є захопленими або

розташовуються перед позиціями схожими позиціями. Якщо позиція проаналізована як так, то вона отримує менший пріоритет на додавання до ігрового поля. Далі позиції аналізуються по кількості балів, що вони принесуть за два ходи з урахуванням методу автозахоплення. Пріоритет отримує та позиція, що принесла найбільшу кількість балів та дає найбільшу кількість балів конкретно в даний хід. Стартова позиція ідентична легкому рівню складності (рис.4.13).

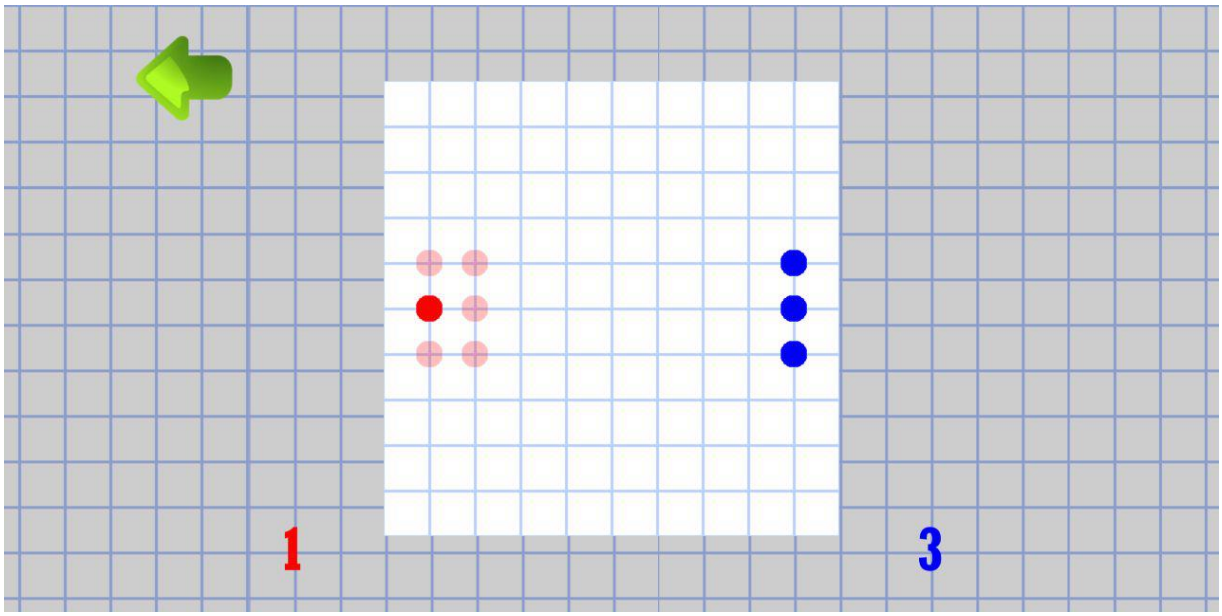


Рисунок 4.13 – Сцена з початковими комірками віртуального гравця середнього рівня складності

Важкий рівень складності

Алгоритм віртуальний гравця важкого рівня складності відповідає алгоритму гравця середнього рівня складності, але має більше захоплених позицій на старті (рис.4.14).

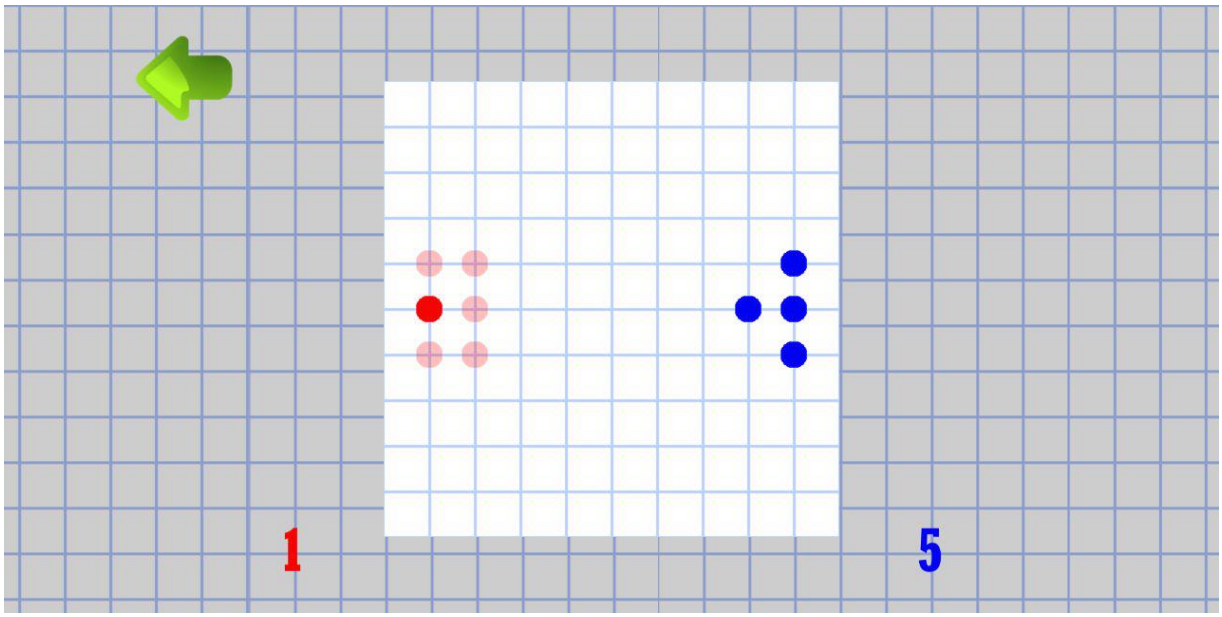


Рисунок 4.14 – Сцена з початковими комірками віртуального гравця важкого рівня складності

Дуже важкий рівень складності

Алгоритм віртуальний гравця дуже важкого рівня складності відповідає алгоритму гравців середнього та важкого рівнів складності складності, але має більше захоплених позицій на старті (рис.4.15).

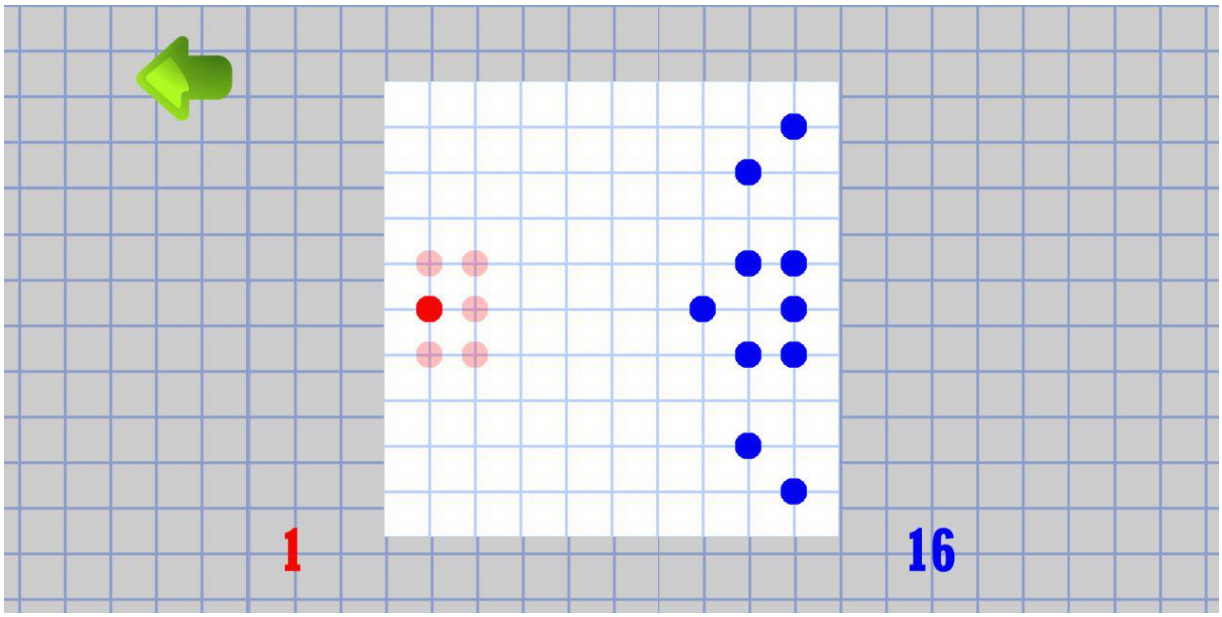


Рисунок 4.15 – Сцена з початковими комірками віртуального гравця важкого рівня складності

4.3.22 Скрипт singleClick.cs

Даний скрипт є функцією, що отримує з ігрового поля координати ходу живого гравця та передає їх на опрацювання у gameSingle.cs.

4.3.23 Скрипт gameMulti.cs

Даний скрипт відповідає за всю механіку гри у мультиплеєрі у двох ігрових режимах. Використовується у сцені гри на двох. Пов’язується з ігровим полем.

4.3.24 Скрипт turnClick.cs

Даний скрипт є функцією, що отримує з ігрового поля координати ходу кожного з гравців та передає їх на опрацювання у gameMulti.cs.

4.4 Готовий продукт

Готовий Android-додаток представляє собою скомпільовані створені сцени та скрипти у проекті з використанням різних версій JDK та SDK для підтримки більшості смартфонів з різними Android-версіями.

Функції реалізації методу автозахоплення на ігровому полі знаходяться у додатку С.

Всі інші файли готового проекту знаходяться на ресурсі GitHub за посиланням https://github.com/KuzmenkoVladyslav/Combat-Dots

Файли програмного коду знаходяться за посиланням: Assets/Scripts/.

Файли локалізації знаходяться за посиланням: Assets/Text/.

Файли спрайтів знаходяться за посиланням: Assets/Sprites/.

Файли текстур знаходяться за посиланням: Assets/Texture/.

Файли звукового супроводження знаходяться за посиланням: Assets/Sound/.

Файли сцен знаходяться за посиланням: Assets/Scenes/.

Файли шрифтів знаходяться за посиланням: Assets/Fonts/.

Файл готового Android-додатку знаходиться на посиланням Game/.

ВИСНОВОК

Під час виконання дипломного проекту був проведений аналіз предметної області. У результаті була виявлена абсолютна унікальність майбутнього проекту (ідея проекту та реалізація є цілком і повністю результатом мозкового штурму його автора та не є плагіатом жодного з нині існуючого програмного забезпечення або реальної гри).

В аналітичній частині дипломного проекту були описані вимоги до розробки ігрового Android-додатка і обґрунтований вибір програмного забезпечення для реалізації завдання.

Завдяки аналізу засобів реалізації було обрано певний набір інструментарію для виконання проекту, а саме – мова програмування C#, рушій гри Unity3D та редактор растрової графіки Adobe Photoshop CS6.

Для планування робіт були створені діаграма Ганта, структурна декомпозиція робіт (WBS), організаційна структура проекту (OBS), матриця відповідальності та проведений аналіз ризиків.

Для передбачення помилок при створенні Android-додатку були побудовані контекстна діаграма IDEF0 та діаграма варіантів використання.

Процес реалізації Android-додатку був розділений на 3 етапи:

- процес попередньої підготовки перед розробкою Android-додатка в якому розроблялися файли локалізації, спрайти та звукове супроводження;
- процес будування сцен майбутнього Android-додатку;
- процес реалізації програмного коду та пов’язання скриптів і сцен за допомогою ігрового рушія.

СПИСОК ЛІТЕРАТУРИ

1. Актуальность мобильных приложений, особенности их создания. [Електронний ресурс] – Режим доступу до ресурсу: http://blognat.ru/aktualnost-mobilnykh-prilozhenijj-osobennosti-ikh-sozdaniya/.

2. Ігри для мобільного ANDROID – ОСОБЛИВА ШТУКА [Електронний ресурс] / http://easy-code.com.ua – Режим доступу до ресурсу: http://easy-code.com.ua/2014/11/igri-dlya-mobilnogo-android-osobliva-shtuka/.

3. Gartner. iOS и Android занимают уже 99,9% рынка мобильных ОС. [Електронний ресурс] / Gartner – Режим доступу до ресурсу: https://www.ixbt.com/news/2018/02/24/ios-android-99-9.html.

4. BYRIL. Точки Онлайн [Електронний ресурс] / BYRIL – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.byril.dots.

5. BYRIL. Морской бой 2 [Електронний ресурс] / BYRIL – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.byril.seabattle2.

6. Easybrain. Судоку [Електронний ресурс] / Easybrain – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.easybrain.sudoku.android.

7. Evgeny Karavashkin. Сапёр на русском [Електронний ресурс] / Evgeny Karavashkin – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=Draziw.Button.Mines.

8. MagicAnt.Inc. 1LINE - one-stroke puzzle game [Електронний ресурс] / MagicAnt.Inc – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.gamestart.oneline.

9. Moodle ЧДУ ім. П. Могили. Moodle ЧДУ ім. П. Могили [Електронний ресурс] / Moodle ЧДУ ім. П. Могили // Moodle ЧДУ ім. П. Могили. – 2014. – Режим доступу до ресурсу: http://moodle.chdu.edu.ua/mod/resource/view.php?id=64700.

10. Нечітка_множина [Електронний ресурс] // Wikipedia. – 2013. – Режим доступу до ресурсу:

https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D0%B0_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%B0.

11. Arrays in Unity3D [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://docs.unity3d.com/ru/530/ScriptReference/Array.html.

12. Создание и уничтожение игровых объектов (GameObjects) [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://docs.unity3d.com/ru/current/Manual/CreateDestroyObjects.html.

13. ScriptingConcepts [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.unity3d.com/ru/current/Manual/ScriptingConcepts.html.

14. Штучний інтелект [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.

15. Object.Instantiate [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Object.Instantiate.html.

16. Color [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Color.html.

17. AndroidDevice [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/Android.AndroidDevice.html.

18. SDK Android Studio [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://developer.android.com/studio/index.html.

19. Oracle JDK for Android [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.oracle.com/technetwork/java/javase/downloads/index.html.

20. ArrayList in C# [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/4.3.php.

21. ІМА 2019. ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА / МАТЕРІАЛИ та програма НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ / – Суми: Сумський державний університет, 2019. – С. 128.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інформаційної системи підтримки діяльності сервісного центру
«Ігровий мобільний додаток “Combat Dots”»

Суми 2019
Формування вимог до моделей і інформаційної технології ігрового мобільного додатка “Combat Dots”

1. ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ ANDROID-ДОДАТКУ

Призначення Android-додатку

Ігровий мобільний додаток призначений для розваг у вільний час та для інтелектуального та стратегічного розвитку людського мозку.

Мета створення Android-додатку

Реалізувати власну ідею логіко-стратегічного мобільного додатку для перенесення її місця дії з паперу на телефон та для популізації розумних ігор серед людей.

Цільова аудиторія Android-додатку

Ігровий мобільний додаток є і розважальним і розвиваючим, тому самостійно користуватися ним може будь-хто, кому вже виповнилося 7 років та діти від 3 до 7 років у присутності дорослих.

2. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вимоги до функціональних можливостей Android-додатку

Гра розрахована на людей, що хочуть під час п’ятихвилинки підтримувати мізки у тонусі та позмагатися з другом або штучним інтелектом у розумовому поєдинку.

Продукт повинен мати наступний функціонал:

Ігровий функціонал

Чотири режими гри:

- Одиночна гра проти скриптової імітації діяльності різнорівневого штучного інтелекту у стандартному режимі;
- Одиночна гра проти скриптової імітації діяльності різнорівневого штучного інтелекту у режимі гри «Тактика»;
- Гра на двох у стандартному режимі;
- Гра на двох у режимі гри «Тактика».

Графічний та звуковий функціонали

- вибір світлої або темної теми;
- вибір мови
- відключення звуку.

Інтерфейс

- одиночна гра;
- гра на двох
- налаштування
- допомога
- вихід

Вимоги до вхідних та вихідних даних

Вхідними даними програми є ігрові налаштування, які обирає користувач, а саме обрана мова, обрана тема, обрані налаштування звуку та обраний режим гри.

Вихідними даними є гра згідно обраних налаштувань.

Вимоги до програмного та апаратного забезпечення

Для функціонування додатку необхідна операційна система Android 5.0 або новіша версія.

3. СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ ANROID-ДОДАТКА4

Детальна послідовність етапів створення гри наведено в табл. А.1

Таблиця А.1 – Етапи створення гри

№	Склад і зміст робіт	Строк розробки (в робочих днях)
1	Створення растрових об’єктів для додатка	5
2	Програмування математичної матриці для програмного сегменту додатка	10
3	Пов’язання растрових об’єктів з програмним сегментом додатку	15
4	Реалізація інтерфейсу	10
5	Тестування	5

ДОДАТОК Б.
ПЛАНУВАННЯ РОБІТ

Планування змісту структури робіт IT-проекту (WBS). Структурна декомпозиція робіт (work breakdown structure, WBS) - у менеджменті та системотехніці є орієнтованою на повне завершення проекту та детермінованою декомпозицією проекту на менші частки. Є ключовою часткою робіт по проекту, яка організовує командну робот. Структурна декомпозиція робіт проекту представлена на рис. Б1.

Організаційна структура проекту (OBS). Організаційна структура проекту (OBS) – це графічне відображення учасників проекту та їхніх відповідальних осіб, що були залучені до реалізації проекту. На верхньому рівні OBS проекту знаходиться керівник та команда управління проектом; на наступному рівні – виконавці. Останнім рівнем OBS-структури є відповідальні особи виконавців.

OBS структура представлена на рисунку Б.2.

Побудова матриці відповідальності. Матриця відповідальності (Responsibility Assignment Matrix) забезпечує опис і узгодження структури відповідальності за виконання пакетів робіт. Вона являє собою форму опису розподілу відповідальності за реалізацію робіт проекту із зазначенням ролі кожного з виконавців.

На рисунку Б.3 показано матрицю відповідальності проекту.

Побудова календарного графіка виконання IT-проекту. Діаграма Ганта - це вид діаграми, що використовується для планування і контролю виконання проекту. Такий мережевий графік присутній практично у всіх СУ проектами. На діаграмі

відображаються завдання і стадії проекту з урахуванням їх часу виконання. Завдання на діаграмі можуть бути залежними один від одного або незалежними.

Для побудови діаграми Ганта використовуємо програму MS Office Project 2013 року.

На рисунку Б.4 показана діаграма Ганта.

Управління ризиками. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків.

Визначимо ризики:

- R1 – зміна вимог при реалізації проекту;
- R2 – невчасне виконання етапів проекту;
- R3 – виявлення помилок на етапі завершення;
- R4 – виникнення несправностей в апаратному забезпеченні;
- R5 – непередбачувані життєві обставини розробника.

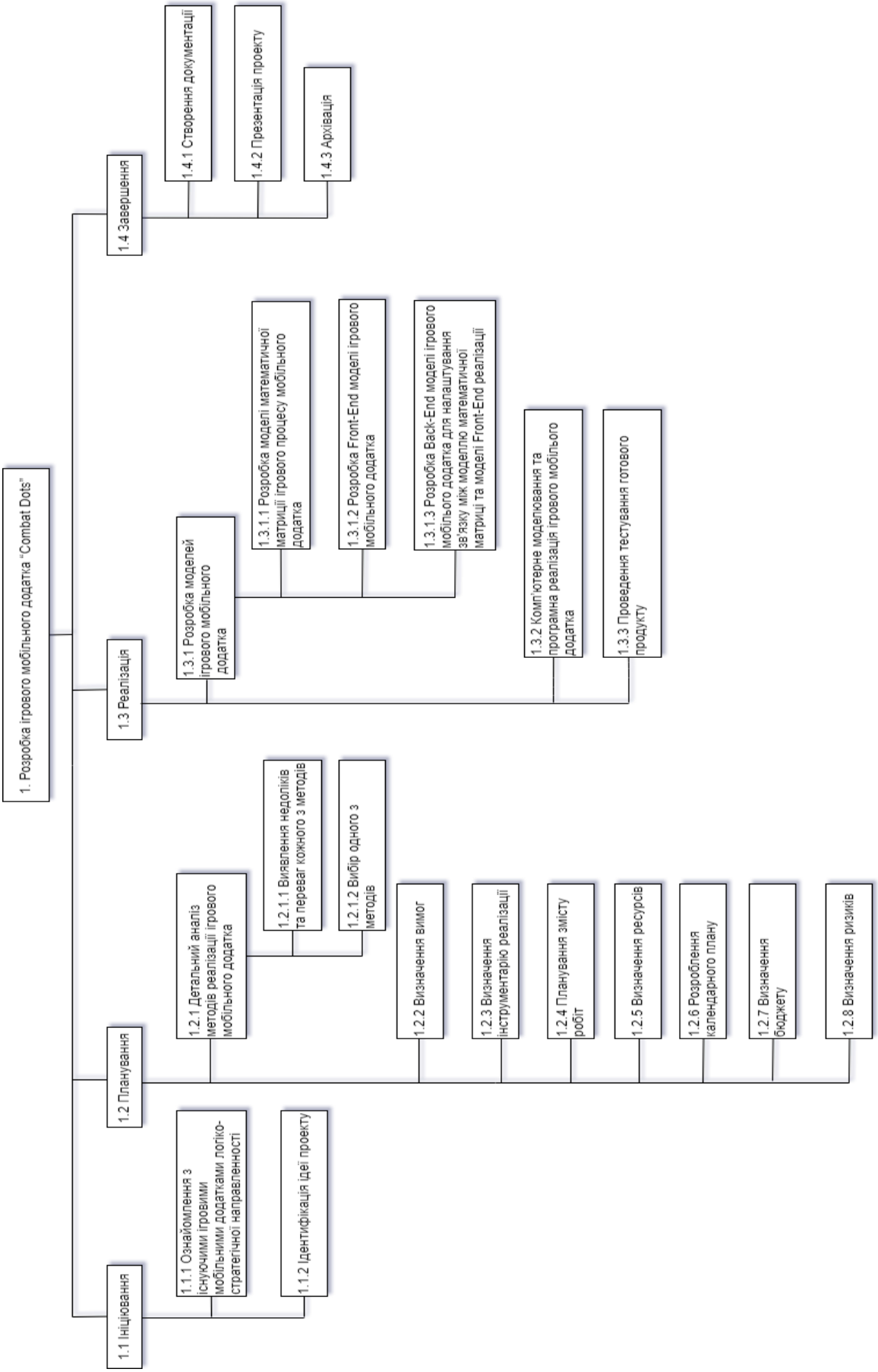


Рисунок Б.1 – WBS структура проекту

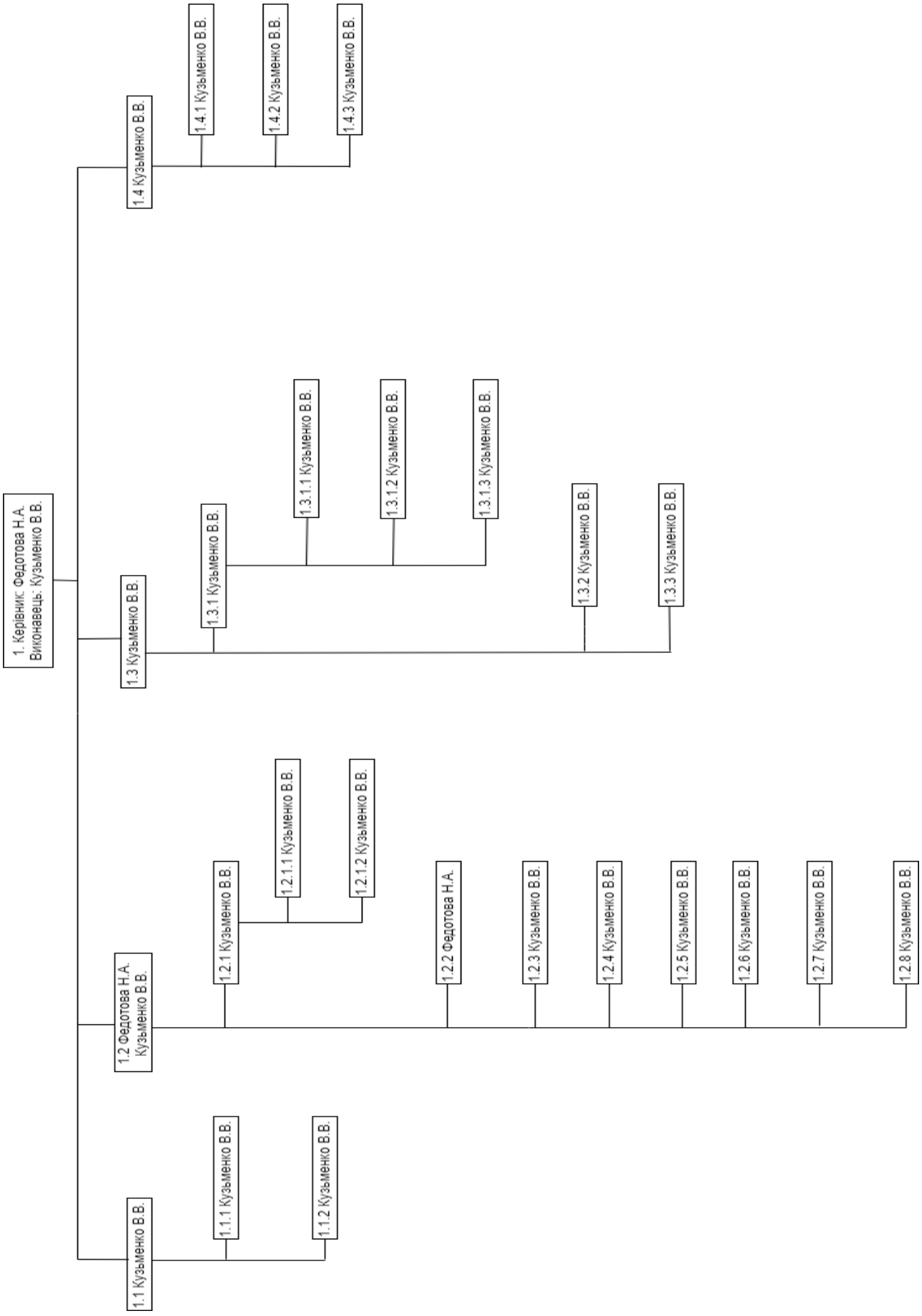
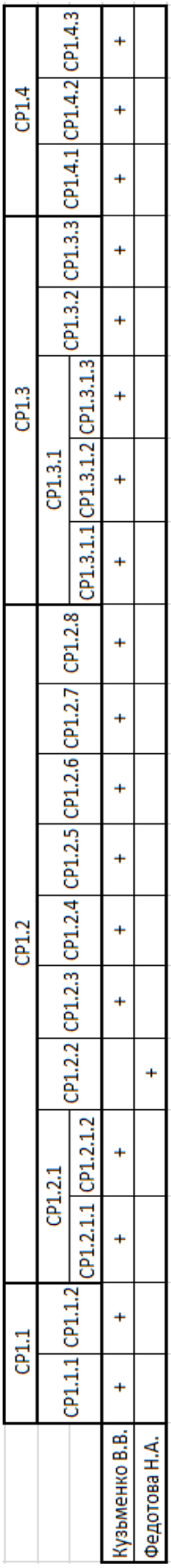


Рисунок Б.2 – OBS структура



	CP1.1		CP1.2								CP1.3				CP1.4		
	CP1.1.1	CP1.1.2	CP1.2.1		CP1.2.2	CP1.2.3	CP1.2.4	CP1.2.5	CP1.2.6	CP1.2.7	CP1.2.8	CP1.3.1			CP1.3.2	CP1.3.3	CP1.4.1	CP1.4.2	CP1.4.3
			CP1.2.1.1	CP1.2.1.2								CP1.3.1.1	CP1.3.1.2	CP1.3.1.3					
Кузьменко В.В.	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+
Федотова Н.А.					+														

Рисунок Б.3 – Матриця відповідальності

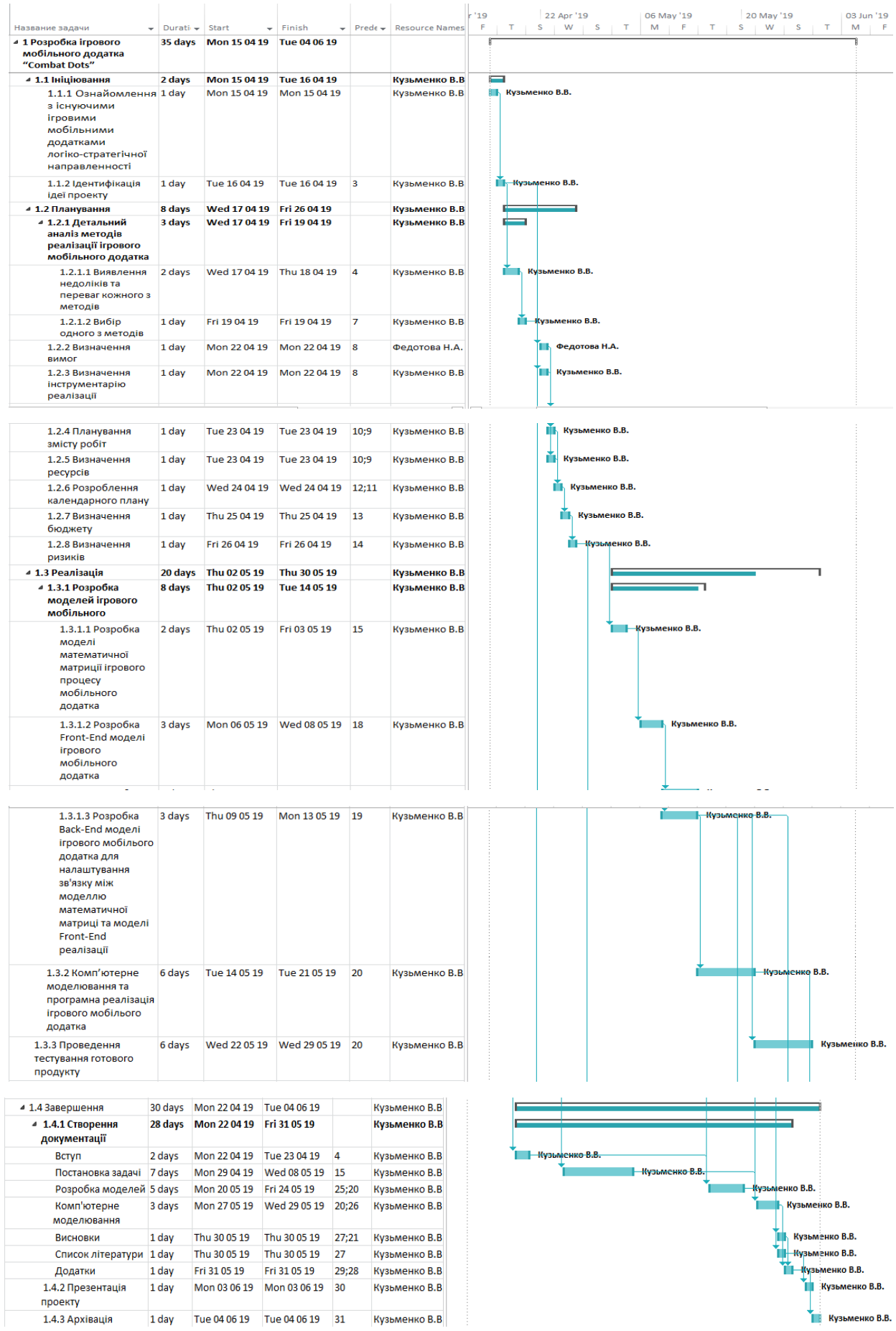


Рисунок Б.4 – Діаграма Ганта

Ймовірність виникнення ризиків

Таблиця Б.1 – Ймовірності виникнення ризиків

Ймовірність виникнення	R1	R2	R3	R4	R5
Дуже низька	+				
Низька		+	+		+
Середня					
Висока				+	
Дуже висока					

Таблиця Б.2 – Значимість впливу ризиків

Значимість впливу	R1	R2	R3	R4	R5
мінімальна					
низька					+
середня					
висока	+	+		+	
максимальна			+		

Таблиця Б.3 – Втрати при виникненні ризиків

Втрати			R3	R4	
		R2			
	R1				
					R5
	Ймовірність				

Виходячи з цього, було визначено чотири критичних ризики, такі як:

- R1 – зміна вимог при реалізації проекту;
- R2 – невчасне виконання етапів проекту;
- R3 – виявлення помилок на етапі завершення;
- R4 – виникнення несправностей в апаратному забезпеченні.

Першого ризику «зміна вимог при реалізації проекту» можна уникнути, заздалегідь визначивши вимоги з керівником. Другого ризику «невчасне виконання етапів проекту» можна уникнути, якщо заздалегідь розподілити свій час та ресурси. Третій ризик «виявлення помилок на етапі завершення» найкритичніший, так як у разі його виникнення на виправлення помилок піде багато часу. Четвертий ризик

«виникнення несправностей в апаратному забезпеченні» може привести до некоректної роботи системи.

Формування бюджету проекту. Останнім етапом планування проекту – є етап розподіл бюджету даного проекту. Були визначені особи, які брали участь в даному проекті та між якими необхідно розподілити бюджет:

- розробник;
- керівник.

У таблиці Б.4 приведений бюджет на витрачення заробітної плати.

Таблиця Б.4 – Розподілення бюджету

Посада	За 1 год/грн	Робочих годин	Всього
Розробник	375	200	75000
Керівник	400	3,5	1400

Бюджет заробітної плати складає *76 400 грн*.

ДОДАТОК С

Реалізація методу автозаповнення у файлах механіки гри на прикладі функцій для першого гравця.

Наступна функція викликається після того як один із користувачів зробить свій хід:

```
private void Checking_first()
   {
// змінна, що підраховує кількість нічийних комірок, що не є прилеглими до комірок іншого гравця
      checking_fir = 0;
// змінна, що підраховує кількість нічийних комірок, що не є прилеглими до комірок іншого гравця, але в наступній ітерації
      new_checking_fir = 0;
// всі нічийні комірки на ігровому полі отримують новий стан «auto-first» завдяки якому і відбувається перевірка
      for (int i = 0; i < gameField.GetLength(0); i++)
      {
         for (int j = 0; j < gameField.GetLength(1); j++)
         {
            if (gameField[i, j].Equals("neutral"))
            {
               gameField[i, j] = "auto-first";
             }
         }
      }
// результат виклику фукнції Auto_First() записується у зміну
      checking_fir = Auto_First();
// наступний результат виклику фукнції Auto_First() записується у зміну
      new_checking_fir = Auto_First();
// перевірка на рівність цих двох змінних, якщо змінні рівні, то цикл завершується, якщо ні, то у першу змінну записується результат другої змінної, а у другу змінну знову викликається функція Auto_First()
      while (checking_fir != new_checking_fir) {
         checking_fir = new_checking_fir;
         new_checking_fir = Auto_First();
      }
//після виходу з циклу всі комірки, що зберегли стан «auto-first» отримують стан належності гравцю
      for (int i = 0; i < gameField.GetLength(0); i++)
      {
         for (int j = 0; j < gameField.GetLength(1); j++)
         {
            if (gameField[i, j].Equals("auto-first"))
            {
               gameField[i, j] = "pre-first";
            }
         }
      }
```

```
//викликається функція, що відображає дані комірки належними гравцю
      Create();
   }
```

Функція, результат якої записувався у змінні попередньої функції:

```
private int Auto_First()
   {
//змінна-лічильник кількості комірок зі станом «auto-first»
      testing_auto_first = 0;
//перевірка комірок зі станом «auto-first» на прилеглість до комірок зі станом «second» або «neutral»
      for (int i = 0; i < gameField.GetLength(0); i++)
      {
         for (int j = 0; j < gameField.GetLength(1); j++)
         {
            if (gameField[i, j].Equals("auto-first"))
            {
               if (i > 0 && gameField[i - 1, j].Equals("second") || i > 0 && gameField[i - 1, j].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (j > 0 && gameField[i, j - 1].Equals("second") || j > 0 && gameField[i, j - 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (i > 0 && j > 0 && gameField[i - 1, j - 1].Equals("second") || i > 0 && j > 0 && gameField[i - 1, j - 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (j < 8 && gameField[i, j + 1].Equals("second") || j < 8 &&  gameField[i, j + 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (i < 8 && gameField[i + 1, j].Equals("second") || i < 8 && gameField[i + 1, j].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (i < 8 && j < 8 && gameField[i + 1, j + 1].Equals("second") || i < 8 && j < 8 && gameField[i + 1, j + 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (i < 8 && j > 0 && gameField[i + 1, j - 1].Equals("second") || i < 8 && j > 0 && gameField[i + 1, j - 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }

               if (i > 0 && j < 8 && gameField[i - 1, j + 1].Equals("second") || i > 0 && j < 8 && gameField[i - 1, j + 1].Equals("neutral"))
               {
                  gameField[i, j] = "neutral";
               }
            }
         }
      }
//підрахунок кількості комірок зі станом «auto-first», що залишилися після перевірки на прилеглість
      for (int i = 0; i < gameField.GetLength(0); i++)
      {
         for (int j = 0; j < gameField.GetLength(1); j++)
         {
            if (gameField[i, j].Equals("auto-first"))
            {
               testing_auto_first++;
            }
         }
      }

      return testing_auto_first;
   }
```