МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ І. С. Скарга-Бандурова
«____»_____ 2020 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Ігровий додаток Vegetable Samurai для ОС Android

Освітній ступінь "бакалавр"
Спеціальність 122 Комп'ютерні науки
(шифр і назва спеціальності)

Керівник проекту: _____ (підпис) М.Є. Щербакова (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О. Критська (ініціали, прізвище)

Здобувач вищої освіти: _____ (підпис) А.І. Пронька (ініціали, прізвище)

Група: КН-16д

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ:
Т.в.о. завідувача кафедри КНІ
_____ С.О.Сафонова
«_____» _____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Проньки Андрія Івановича
(прізвище, ім'я, по батькові)

1. Тема роботи Ігровий додаток Vegetable Samurai для ОС Android

керівник проекту (роботи) Щербакова М.Є., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "30" 04 2020 р. № 77/15.15
2. Термін подання здобувачем вищої освіти роботи 12.06.2020
3. Вихідні дані до роботи ОС Android, мова програмування C#, ігровий рушій Unity

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз та постановка задачі
2) Вибір засобів розробки
3) Розробка ігрового додатку Vegetable Samurai для ОС Android
4) Охорона праці

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Я.О., ст.викл.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка заходів по безпечному веденню робіт	27.04.20-29.04.20	
2	Пошук літературних джерел	30.04.20-01.05.20	
3	Розробка технічного завдання	02.05.20-15.05.20	
4	Розробка ігрового додатку	16.05.20-06.06.20	
5	Оформлення пояснювальної записки	07.06.20-12.06.20	

Здобувач вищої освіти _____ А.І. Пронька
(підпис) (ініціали, прізвище)

Керівник _____ М.Є. Щербакова

Р Е Ф Е Р А Т

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 76 с., 43 рис., 2 табл., 31 бібліографічних джерел, 2 додатки.

Об'єкт розробки: ігровий додаток Vegetable Samurai для ОС Android.

Мета роботи: розробка ігрового додатку Vegetable Samurai для ОС Android.

В проекті виконано:

1. Аналіз відео ігор.
2. Вибір платформи і програмних засобів реалізації додатку.
3. Визначення функціоналу додатку.
4. Розробка ігрового додатку Vegetable Samurai для ОС Android.

Практичне значення, галузь застосування роботи: розважання, мобільні пристрої.

МОБІЛЬНА ПЛАТФОРМА, C#, ВІДЕО ІГРИ, UNITY, АНДРОЇД-ДОДАТОК

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А, м. Сєвєродонецьк, 93400

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1 Відеоігри 9
1.2 Мобільні платформи 12
1.3 Постановка задачі 14
1.4 Висновки до розділу 1 16
2 ВИБІР ЗАСОБІВ РОЗРОБКИ 17
2.1 Операційна система Android 17
2.2 Android Studio 20
2.3 Ігровий рушій Unity 22
2.4 Мова програмування C# 23
2.5 Висновки до розділу 2 25
3 РОЗРОБКА ІГРОВОГО ДОДАТКУ VEGETABLE SAMURAI ДЛЯ ОС ANDROID 26
3.1 Загальна будова додатку 26
3.2 Будова та функціонал сцени PreLoader 28
3.3 Будова та функціонал сцени Menu 30
3.4 Будова та функціонал сцени Game 34
3.5 Висновки до розділу 3 43
4 ОХОРОНА ПРАЦІ 44
4.1 Загальні питання з охорони праці 44
4.1.1 Правові та організаційні основи охорони праці 44
4.1.2 Організаційно-технічні заходи з безпеки праці 45
4.2 Аналіз стану умов праці 46
4.2.1 Вимоги до приміщень 46
4.2.2 Вимоги до організації місця праці 47
4.3 Виробнича санітарія 48
4.4 Гігієнічні вимоги до параметрів виробничого середовища 48

4.4.1 Освітлення 48
4.4.2 Вентилювання 51
4.4.3 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі) 51
4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій 53
4.6 Висновки до розділу 4 54
ВИСНОВКИ 55
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 56
Додаток А Лістинг програмного коду 60
Додаток Б Комп'ютерна презентація 72

ВСТУП

Сучасні відеоігри є одним з найпопулярніших та найефективніших видів розваг, а також, на мою думку, формою мистецтва.

Через швидкі технологічні прориви, що відбулися в кінці двадцятого та двадцять першому століттях, історія мобільних ігор досить коротка. Найбільш ранньою популярною грою вважається варіант тетрісу 1994 року на пристрої Hagenuk MT-2000.

У 1997 році Nokia запустила гру Snake. Гра була попередньо встановлена на більшості мобільних пристроїв, що випускалися компанією Nokia. Вона все ще залишається найбільш поширеною грою і знаходиться в приблизно 350 мільйонах мобільних пристроїв у всьому світі.

У 1999 році запуск платформи I-mode від NTT DoCoMo призвів до першої інстанції комерціалізованих та завантажуваних мобільних ігор. Це відбулося в Японії. До 2000 року мобільні ігри, які можна завантажити, стали доступними через різні платформи та в країнах, де були доступні телефони та сучасні телекомунікаційні мережі. До 2003 року в Японії стали доступними ряд ігор, включаючи головоломки та віртуальні ігри з домашніми тваринами.

Хоча різноманітні мобільні ігри були доступні та широко розповсюджені, справжній бум стався 2008 року, коли Apple Inc запустила магазин додатків iOS. Різні компанії змогли почати розробляти складніші ігри у вигляді додатків та продавати широкій аудиторії. Це викликало конкуренцію в ігровій індустрії, що привабило більше компаній залучитися до розробки цікавіших ігор в різних жанрах.

Сьогодні розробляються найрізноманітніші мобільні ігри. Ігри сумісні як з iOS, так і з платформою Android та доступні через Google Play та магазин додатків iOS. Багато сучасних пристроїв продаються з вже попередньо встановленими іграми. Деякі ігри можна завантажити та грати безкоштовно, деякі – на основі передплати.

Мене зацікавила тема розробки відеогри за низкою причин: у мене великий досвід в гранні відеоігор, тому я добре розумію, які саме чинники є найбільш важливими для розваги гравця; розробка відеоігор надає розробнику майже повну свободу в плані реалізації проекту, мабуть, стосовно всіх етапів розробки, будь то вибір платформи, середовища розробки, мови програмування, графічного дизайну, ігрового процесу, способу монетизації тощо; реалізація мінімальних функцій відеоігор досить невелика, а потенціал для подальшого покращення майже невичерпний.

В дипломному проекті передбачається розробити ігровий додаток Vegetable Samurai для ОС Android. Гральний процес складається з наступних елементів: на початку гри знизу екрану вверх або по діагоналі вилітають овочі, гравець повинен торкаючись до екрану проводити лінію через овочі, розрізаючи їх навпіл, до того як вони вилетять за межі екрана. Якщо овочі вилітають за межі екрану нерозрізаними, у гравця віднімається одна з трьох одиниць помилки; якщо гравець втрачає всі три одиниці помилки – гра завершується; якщо гравець набрав більше очок ніж коли-небудь раніше – встановлюється новий рекорд очок. Іноді замість овочу вилітає бомба, якщо гравець її розрізає, вона вибухає, і гравець втрачає одиницю помилки; тому, щоб набрати більше очок, йому потрібно бути уважним, а не просто бездумно водити по екрану. Чим більше гравець розрізає овочів без втрати одиниць помилки, тим швидше вилітають овочі. Перед початком гри гравець може вибрати один з трьох рівнів складності, що визначає початкову швидкість овочів.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Відеоігри

Відеогра – це електронна гра, яка передбачає взаємодію з користувальницьким інтерфейсом для створення візуального зворотного зв’язку на дво- або тривимірному пристроях відображення відео, таких як сенсорний екран, гарнітура віртуальної реальності, та монітор або телевізор. Починаючи з 1980-х років, відеоігри стають все більш важливою частиною індустрії розваг, і чи є вони також формою мистецтва – це питання спірне.

Розробка та авторство відеоігор, як і будь-якої іншої форми розваг, часто є міждисциплінарною сферою. Розробники відеоігор, як зазвичай називають співробітників цієї галузі, це насамперед програмісти та графічні дизайнери. Протягом багатьох років це визначення розширювалося, включаючи майже всі професії, які потрібні в створенні будь-яких кіно- або телевізійних програм, включаючи звукових дизайнерів, музикантів, та інших техніків; а також професії, характерні для відеоігор, наприклад дизайнера ігор. За координацію розробників відповідають виробники ігор.

У перші часи цієї галузі частіше одна людина виконувала всі ролі розробки відеоігри. Оскільки платформи стали складнішими та потужнішими за типом матеріалу, який вони можуть представити, знадобилися більші команди для створення всього мистецтва, програмування, кінематографії та ін. Це не означає, що вік ігор, розроблених однією людиною, минув, оскільки це все ще іноді зустрічається на ринках казуальних ігор та портативних комп'ютерів, де невеликі ігри переважають через технічні обмеження, такі як обмежена оперативна пам'ять або відсутність спеціалізованих можливостей 3D рендерингу на цільовій платформі (наприклад, деякі КПК).

З появою стандартних операційних систем для мобільних пристроїв, таких як iOS та Android та пристроїв з більшою продуктивністю обладнання, мобільні ігри стали важливою платформою. Ці ігри можуть використовувати унікальні

функції мобільних пристроїв, які не завжди можна знайти на інших платформах, такі як система глобального позиціонування та цифрові камери для підтримки ігрового процесу розширеної реальності. Мобільні ігри також призвели до розвитку мікротранзакцій як ефективної моделі доходу для казуальних ігор.

Варто зазначити, що в дипломному проекті передбачається розробка гри, яку можна класифікувати як казуальна. Казуальні (від англ. “casual” – повсякденний, несерйозний) ігри отримали свою назву від простоти, доступності, простого для розуміння ігрового процесу та набору правил. Крім того, казуальні ігри часто підтримують можливість починати і зупиняти гру на вимогу. Казуальні ігри як формат існували задовго до введення терміну і включають такі відеоігри, як Solitaire або Minesweeper, які зазвичай можна знайти попередньо встановленими в багатьох версіях операційної системи Microsoft Windows. Прикладами жанрів цієї категорії є match three, hidden object, time management, puzzle або багато ігор у стилі tower defense. Казуальні ігри, як правило, доступні через магазини додатків та інтернет-магазини, такі як Google Play та App Store, або надаються для безкоштовної гри через веб-портали, такі як Newgrounds. Хоча казуальні ігри найчастіше граються на персональних комп’ютерах, телефонах або планшетах, їх можна також знайти у багатьох службах завантаження он-лайн консольних систем (наприклад, PlayStation Network, WiiWare або Xbox Live).

Основними особливостями казуальних ігор є:

– веселий, простий геймплей, який легко зрозуміти;

– простий користувальницький інтерфейс, керований інтерфейсом натискання та переведення мобільного телефону або однією кнопкою миші;

– короткі сеанси, тому гру можна грати під час перерв на роботі, під час пересування громадським транспортом або під час очікування в черзі де завгодно;

– часто знайомі візуальні елементи, наприклад, гральні карти або сітка об'єктів Match 3.

Культура відеоігор – це нова світова медіакультура, що формується навколо відеоігор. Набуваючи все більшої популярності, відеоігри значно вплинули на популярну культуру. Культура відеоігор також еволюціонувала з часом рука об

руку з інтернет-культурою та зростаючою популярністю мобільних ігор. Багато людей, які грають у відеоігри, ідентифікують себе як геймери, що може означати кого завгодно, від того, хто полюбляє ігри, до того, хто захоплюється ними. У міру того, як відеоігри стають все більш соціальними з багатокористувацькими та он-лайн можливостями, геймери опиняються частиною зростаючих соціальних мереж. Ігри можуть бути як розвагами, так і змаганнями, оскільки нова тенденція, відома як електронні види спорту, стає все більш широко прийнятою. У 2010-х роках відеоігри та дискусії про тенденції та теми відеоігор можна побачити в соціальних медіа, політиці, телебаченні, кіно та музиці.

В залежності від рішень гравця відеоігри можуть мати як позитивний вплив, так і негативний. До потенційних факторів позитивного впливу можна віднести можливе покращення самопочуття, розвиток пізнавальних навичок та навичок роботи в команді, а також мотивація фізичних навантажень.

SDT (Self-determination theory – теорія самодетермінації) – теорія про психологічний підхід до розуміння людської мотивації, особистості і психологічного благополуччя, зокрема, докладну проблематику внутрішньої і зовнішньої мотивації.[1] SDT забезпечує основу для розуміння ефектів грання в відеоігри; самопочуття, навички вирішення проблем, навички поведінки в групі, фізичне навантаження. Ці фактори можна виміряти, щоб визначити вплив відеоігор на людей.

Можливість створити ідеальний образ власного "Я" та отримати декілька варіантів зміни цього образу дає відчуття задоволення [2]. Ця тема має багато суперечок; точно невідомо, чи є ця свобода корисною для особистості чи згубною.

Пізнавальні навички можна посилити за допомогою повторення головоломок, ігор на пам'ять, орієнтації в просторі та контролю уваги. Більшість відеоігор надають можливості використовувати ці навички з можливістю спробувати кілька разів навіть після невдачі. Це дозволяє гравцеві вчитися на помилках і повністю розуміти, як і чому може працювати рішення проблеми.

Розвиток он-лайн ігор створив можливість спілкуватися та працювати разом, щоб виконати певне завдання. Вміння працювати групою в грі добре відповідає роботі в колективі в реальності.

З впровадженням Wii Fit та VR (Virtual Reality – віртуальна реальність) популярність відеоігор з елементами фізичного навантаження зростає, що дозволяє гравцям відчувати більш активну, а не сидячу гру. Мобільні додатки намагаються розширити цю концепцію впровадженням елементів доповненої реальності, як в грі Pokémon Go, що передбачає ходьбу для прогресу в грі.

1.2 Мобільні платформи

Мобільна операційна система - це операційна система для мобільних телефонів, планшетів, смарт-годинників, ПК 2-в-1 (які можна перевести в режим ноутбука або від'єднати і працювати в режимі планшетного ПК) або інших мобільних пристроїв. Хоча такі комп’ютери, як типові ноутбуки, є "мобільними", операційні системи, які зазвичай використовуються на них, не вважаються мобільними, оскільки вони спочатку були розроблені для настільних комп'ютерів, які в минулому не мали або не потребували певних мобільних функцій. Ця відмінність стає розмитою в деяких нових операційних системах, які є гібридами, створеними для обох цілей.

Мобільні операційні системи поєднують в собі функції операційної системи персонального комп’ютера з іншими функціями, корисними для мобільного чи портативного використання, і, як правило, включають вбудований бездротовий модем та SIM-лоток для телефонії та передачі даних. Першого кварталу 2018 року було продано понад 383 мільйони смартфонів, на яких 86,2 відсотка працює на Android та 12,9 відсотка під керуванням iOS [3]. Сам Android є популярнішим за популярну настільну операційну систему Windows, і загалом використання смартфонів (навіть без планшетів) переважає використання настільних комп’ютерів.

Мобільні пристрої з можливостями мобільного зв’язку (наприклад, смартфони) містять дві мобільні операційні системи – основна програмна платформа, орієнтована на користувачів, доповнюється другою власною операційною системою реального часу, що керує радіо та іншим обладнанням. Дослідження показали, що ці низькорівневі системи можуть містити ряд недоліків безпеки, що дозволяє шкідливим базовим станціям отримувати високий рівень контролю над мобільним пристроєм [4].

Мобільні операційні системи мають більшість використання з 2017 року (вимірюється веб-використанням); навіть за умови, що під керуванням їх використовуються лише смартфони (крім планшетів мобільні операційні системи все одно мають більшість використання). Таким чином, традиційні настільні ОС тепер є меншиною використовуваних видів ОС.

В даний час найпопулярнішими мобільними операційними системами є Android та iOS. За даними Statcounter Global Stats від 04.05.2020 ці дві операційні системи займають 99,47% світового ринку операційних систем для мобільних пристроїв, тому звернемо свою увагу тільки на ці дві платформи.

Android – це мобільна операційна система на основі модифікованого ядра Linux, розроблена Google. Базова система є відкритим кодом (і лише ядра є копілефт), але додатки та драйвери, що забезпечують функціональність, все більше стають пропрієтарними [5]. Окрім того, що Android є найпоширенішою операційною системою для смартфонів, вона також є найпопулярнішою операційною системою для комп'ютерів загального призначення (категорія, що включає настільні комп'ютери та мобільні пристрої), хоча Android не є популярною операційною системою для звичайних (настільних) персональних комп'ютерів (ПК). Незважаючи на те, що операційна система Android є безкоштовним та відкритим програмним забезпеченням у продаваних пристроях, значна частина програмного забезпечення, що постачається разом із нею (включаючи програми Google та програмне забезпечення встановлене виробником), є пропрієтарним програмним забезпеченням.

Випуски Android до 2.0 (1.0, 1.5, 1.6) використовувались виключно на мобільних телефонах. Випуски Android 2.x в основному використовувались для мобільних телефонів, але також для деяких планшетів, Android 3.0 був випуском, орієнтованим на планшет, і офіційно не працює на мобільних телефонах, а сумісність телефонів і планшетів об’єднана з Android 4.0. Поточна версія Android – Android 10, випущена 3 вересня 2019 року.

iOS (раніше названа iPhone OS) була створена компанією Apple Inc. Ця операційна система є другою найпоширенішою на смартфонах в усьому світі, але найбільш прибутковою через агресивну цінову конкуренцію між виробниками пристроїв на базі Android. Операційна система iOS є пропрієтарною, і побудована на основі відкритої операційної системи Darwin. IPhone, iPod Touch, iPad, та Apple TV другого чи третього покоління використовують iOS, який походить від macOS.

Нативні додатки сторонніх розробників офіційно не підтримувались до виходу iPhone OS 2.0 11 липня 2008 року. До цього єдиним способом встановлення сторонніх додатків був "джейлбрейк" (сленг, від англ. дослівно jailbreak – втеча з в'язниці). В останні роки джейлбрейк-сцена кардинально змінилася через постійні зусилля Apple щодо захисту своєї операційної системи та запобігання несанкціонованим модифікаціям. В даний час джейлбрейки останніх ітерацій iOS вимагають повторного джейлбрейкінгу пристрою при кожному завантаженні, а недоліки захисної системи для джейлбрейків стає все важче знайти та використовувати.

В даний час всі пристрої iOS розробляються компанією Apple і виробляються Foxconn або іншими партнерами Apple.

1.3 Постановка задачі

Завданням дипломного проекту є створення ігрового додатку для платформи Android. Для цього потрібно реалізувати зручний та легко зрозумілий користувацький інтерфейс, а також захоплюючий ігровий процес.

В користувацькому інтерфейсі потрібно реалізувати головне меню, інтерфейс ігрового процесу, меню призупинення гри та меню завершення гри.

Головне меню повинно містити наступні елементи:

– кнопку початку гри;

– кнопки виключення або включення музики або звуків.

Інтерфейс ігрового процесу повинен містити наступні елементи:

– кнопку призупинення ігри та виклику меню призупинення;

– кількість набраних очок;

– індикатор наявних одиниць помилок.

Меню призупинення гри повинно містити наступні елементи:

– кнопку переходу до головного меню;

– кнопку відновлення ігрового процесу;

– кнопки виключення або включення музики або звуків.

Меню завершення гри повинно містити наступні елементи:

– кнопку початку нової гри;

– кнопку переходу до головного меню;

– кнопки виключення або включення музики або звуків.

Ігровий процесу має реалізувати наступні функції:

– політ овочів знизу екрану вверх прямо або по діагоналі з послідуючим падінням;

– можливість розрізати овочі через проведення через них лінії дотиком до екрану;

– зменшення наявних одиниць помилок, якщо овоч падає за межі екрану нерозрізаним;

– підрахунок очок, які гравець отримує, розрізаючи овочі;

– прискорення польоту овочів після кожного розрізаного овочу до того, як гравець втратить одиницю помилки; тоді швидкість польоту овочів встановлюється такою, як на початку гри;

– завершення гри, якщо гравець втрачає всі три одиниці помилок.

Ігровий процес повинен реалізувати наступні функції:

– поява та переміщення різноманітних овочів або бомб;

– розрізання овочів або бомб при проведенні по ним лінії;

– втрата одиниці помилки при розрізі бомби, якщо овоч падає за межі екрану нерозрізаним;

– програш при втраті 3 одиниць помилок;

– зменшення інтервалу появи овочів та бомб при кожному розрізанні овочу;

– відновлення інтервалу появи овочів та бомб до початкового значення при втраті одиниці помилки;

– збільшення значення очок при розрізанні овочів та оновлення рекордного значення очок;

– розріз всіх овочів без збільшення значення очок при втраті одиниці помилок;

– можливість нажати на паузу;

– можливість виключити або включити музику або звуки.

1.4 Висновки до розділу 1

В даному розділі був виконаний аналіз сучасних відеоігор та мобільних платформ, а також виконана постановка завдання.

За результатами аналізу стану відеоігор в сучасному суспільстві було з’ясовано, що з моменту їх створення вони надалі неспинно набирають популярності і впливу на життя людей.

Розглянувши наявні мобільні платформи ми зрозуміли, що для ігор єдиними значущими платформами є Android та iOS, з двох Android виявилася набагато більш популярною в Україні, що відображується навіть в учбовій програмі.

В третьому підпункті було виконано детальну постановку завдання; створені вимоги до функціоналу створеного додатку дозволять краще підібрати засоби розробки.

2 ВИБІР ЗАСОБІВ РОЗРОБКИ

2.1 Операційна система Android

Для розробки дипломного проекту я обрав операційну систему Android, тому що на ній працює мій смартфон, а також, я вважаю, варто взяти до уваги, що згідно з даними Statcounter Global Stats від 04.05.2020 в Україні Android займає 82.09% ринку, а iOS тільки 17.46%. Розглянемо характеристику операційної системи Android.

Android – це операційна система для мобільних пристроїв, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, розробленого головним чином для мобільних пристроїв із сенсорним екраном, таких як смартфони та планшети. Android розробляється консорціумом розробників, відомим як Open Handset Alliance, з яких головним учасником та комерційним маркетологом є Google.[6]

Основними особливостям, що відрізняють Android від інших мобільних ОС, є унікальності таких елементів, як: користувацький інтерфейс, додатки та управління пам’яттю.

За замовчуванням користувацький інтерфейс Android в основному заснований на прямих маніпуляціях, використовуючи сенсорний ввід, який відносно відповідає дійсним діям, наприклад, проведення пальцем, натискання пальцем, зведення та розведення двома пальцями для маніпулювання екранними об'єктами, а також включає віртуальну клавіатуру [7]. Ігрові контролери та повнорозмірні фізичні клавіатури підтримуються через Bluetooth або USB. Реакція на взаємодію користувача розроблена бути негайною і забезпечує плавний інтерфейс, який часто використовує вібраційні можливості пристрою, щоб забезпечити користувачеві швидкий зворотний зв'язок. Внутрішнє апаратне забезпечення, наприклад, акселерометри, гіроскопи, та датчики наближення використовуються деякими програмами для реагування на додаткові дії користувача, наприклад, регулювання екрана від портретного до пейзажного

залежно від орієнтації пристрою, або дозволяти користувачеві керувати транспортним засобом в гоночній грі, повертаючи пристрій, імітуючи управління кермом.

Додатки ("apps"), що розширюють функціональність пристроїв, створюються за допомогою Android SDK (Software Development Kit – набір для розробки програмного забезпечення) [8], та, часто, мови програмування Java. Java може поєднуватися з C або C++, разом з вибором нестандартних виконуючих середовищ, які дозволяють покращити підтримку C++. Мова програмування Go також підтримується, хоча з обмеженим набором API (Application Programming Interface – інтерфейс програмування додатків). У травні 2017 року Google оголосила про підтримку розробки додатків для Android на мові програмування Kotlin.

SDK включає значний набір інструментів розробки, включаючи відлагоджувач, бібліотеки програмного забезпечення, емулятор мобільних телефонів на основі QEMU, документацію, зразок коду, та навчальні посібники. Спочатку IDE (Integrated Development Environment – інтегроване середовище розробки), що підтримувалося Google, було Eclipse, використовуючи плагін Android Development Tools (ADT). У грудні 2014 року компанія Google випустила Android Studio, заснований на IntelliJ IDEA, як основний IDE для розробки програм Android. Доступні й інші інструменти розробки, включаючи NDK (Native Development Kit – нативний набір розробки), для додатків чи розширень на C або C++, Google App Inventor – візуальне середовище для програмістів-початківців та набір різноманітних фреймворків для міжплатформних мобільних веб-додатків. У січні 2014 року Google оприлюднила фреймворк, заснований на Apache Cordova, для перенесення веб-додатків Chrome HTML 5 на Android, вбудований в оболонку нативної програми. Окрім того, Google у 2014 році придбав Firebase, який надає корисні інструменти для розробників програм та веб-сайтів.

Оскільки пристрої Android зазвичай працюють від акумуляторів, Android спроектований управляти процесами, щоб мінімізувати споживання енергії. Якщо програма не використовується, система призупиняє її роботу, так що, хоча вона

доступна для негайного використання, а не закрита, вона не використовує енергію акумулятора чи ресурси процесора. Android керує програмами, які зберігаються в пам'яті, автоматично: при недостатньому обсязі пам’яті система почне непомітно і автоматично закривати неактивні процеси, починаючи з тих, які були неактивними протягом найдовшого часу.

Також варто звернути увагу на особливості апаратного забезпечення, з яким працює Android. Основною апаратною платформою для Android є ARM (архітектури ARMv7 та ARMv8-A), архітектури x86 та x86-64 також офіційно підтримуються в пізніших версіях Android. Неофіційний проект Android-x86 забезпечив підтримку архітектур x86 до початку офіційної підтримки. Архітектури ARMv5TE та MIPS32 / 64 також історично підтримувалися, але були видалені в пізніших версіях Android. З 2012 року почали з’являтися пристрої Android з процесорами Intel, включаючи телефони та планшети. Отримуючи підтримку 64-розрядних платформ, Android спочатку було запущено на 64-бітних x86, а потім на ARM64. Починаючи з Android 5.0 "Lollipop", крім 32-бітових варіантів підтримуються 64-розрядні варіанти всіх платформ.

Вимоги до мінімальної кількості оперативної пам’яті для пристроїв під управлінням Android 7.1 коливаються на практиці від 2 Гб для найкращого обладнання, до 1 Гб для найпоширенішого, до абсолютного мінімуму 512 Мб для найнижчого 32-розрядного смартфона. Рекомендація для Android 4.4 - мати принаймні 512 Мбайт оперативної пам’яті, тоді як для пристроїв з низькою оперативною пам’яттю необхідний мінімальний обсяг – 340 Мб, який не включає пам'ять, виділену на різні апаратні компоненти, такі як baseband processor (це прилад, що керує усіма радіо функціями). Для Android 4.4 потрібен 32-розрядний архітектурний процесор ARMv7, MIPS або x86 (останні два через неофіційні порти) разом із GPU (Graphics Processing Unit – графічний процесор) сумісним з OpenGL ES 2.0. Android підтримує OpenGL ES 1.1, 2.0, 3.0, 3.1 та останню основну версію, 3.2 та з Android 7.0 Vulkan (а для деяких пристроїв доступна версія 1.1). Деякі додатки можуть вимагати обов’язково певної версії OpenGL ES, а для запуску таких програм потрібне відповідне обладнання GPU.

Пристрої Android містять багато додаткових апаратних компонентів, включаючи камери, GPS (Global Positioning System – система глобального позиціонування), датчики орієнтації, спеціальні ігрові контролери, акселерометри, гіроскопи, барометри, магнітометри, датчики наближення, датчики тиску, термометри та сенсорні екрани. Деякі апаратні компоненти не є обов’язковими, але вони стали стандартними для певних класів пристроїв, наприклад смартфонів, і за їх наявності застосовуються додаткові вимоги. Раніше більше обладнання було обов’язковим, але з часом деякі вимоги були послаблені або усунені взагалі. Наприклад, оскільки Android розроблявся спочатку як ОС телефону, то мікрофони вважалися обов’язковим обладнанням, але з часом функція телефону стала необов'язковою. Раніше Android вимагав підтримку камери автофокусування, потім вимоги були послаблені до обов’язкової підтримки камери з фіксованим фокусом, якщо вона взагалі присутня, оскільки вимога підтримки камери була повністю скасована, коли Android почали використовувати на телеприставках.

Окрім роботи на смартфонах та планшетах, кілька постачальників технічного обладнання додають можливість запускати Android на звичайному ПК із клавіатурою та мишкою. На додаток до їх доступності на комерційно доступному обладнанні, у проекті Android-x86 вільно доступні аналогічні апаратні версії Android, включаючи налаштований Android 4.4. Використовуючи емулятор Android, що входить до Android SDK, або сторонні емулятори, Android також може працювати ненативно в архітектурах x86.

2.2 Android Studio

Android Studio пропонується для створення додатків на офіційному сайті, також саме з цим середовищем розробки відбувалася робота під час навчального процесу, тому я вважаю варто розглянути Android Studio як першу опцію для розробки ігрового додатку.

Android Studio є офіційним IDE для операційної системи Android від Google, побудованим на основі програмного забезпечення IntelliJ IDEA від JetBrains, та розробленого спеціально для розробки програмного забезпечення для Android [9]. Воно доступне для завантаження для операційних систем на базі Windows, macOS та Linux. Android Studio замінило Eclipse ADT (Android Development Tools – інструменти розробки для Android) як основне IDE для розробки нативних додатків для Android.

Про Android Studio було оголошено 16 травня 2013 року на конференції Google I/O. Воно було на стадії попереднього доступу огляду починаючи з версії 0.1 в травні 2013 року, потім вступило у бета-стадію починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна збірка була випущена в грудні 2014 року, починаючи з версії 1.0.

7 травня 2019 року Kotlin замінила Java як бажану мову Google для розробки додатків для Android. Java все ще підтримується, як і C++.

У поточній стабільній версії надаються такі функції:

– підтримка побудови на основі Gradle;

– рефакторинг для Android та швидкі виправлення;

– lint інструменти для отримання даних про продуктивність, зручність використання, сумісність версій та інші проблеми;

– інтеграція ProGuard та можливості підписання програм;

– майстри на основі шаблонів для створення загальних дизайнів та компонентів Android;

– багатий редактор макетів, що дозволяє користувачам перетягувати компоненти інтерфейсу користувача, можливість перегляду макетів на кількох конфігураціях екрана;

– підтримка створення додатків Android Wear;

– вбудована підтримка платформи Google Cloud, яка дозволяє інтегруватися з Firebase Cloud Messaging (раніше "Google Cloud Messaging") та Google App Engine.

– AVD (Android Virtual Device – віртуальний пристрій Android; емулятор) для запуску та налагодження додатків в Android Studio.

Android Studio підтримує всі ті ж мови програмування IntelliJ (і CLion), наприклад Java або C++, та більше з розширеннями, такими як Go; також Android Studio 3.0 або новішої версії підтримує Kotlin та "всі функції мови Java 7 та підмножину функцій мови Java 8, які залежать від версії платформи". Зовнішні проекти підтримують деякі функції Java 9. Хоча IntelliJ заявляє, що Android Studio побудоване на підтримці всіх випущених версій Java та Java 12, незрозуміло, на якому рівні Android Studio підтримує версії Java до Java 12 (у документації згадується часткова підтримка Java 8). Принаймні деякі нові мовні функції до Java 12 можна використовувати в Android.

2.3 Ігровий рушій Unity

Створювати гру було вирішено, використовуючи ігровий рушій Unity, який дозволяє переносити створений проект на численну кількість платформ на вибір, включаючи Android.

Unity - це міжплатформенний ігровий рушій, розроблений Unity Technologies, вперше оголошений та випущений у червні 2005 року на конференції компанії Apple Inc. “The Apple Worldwide Developers Conference” як ігровий рушій Mac OS X. Станом на 2018 рік, Unity був розширений для підтримки понад 25 платформ. Він може бути використаний для створення найрізноманітніших ігор: тривимірних, двовимірних, віртуальної реальності, з доповненою реальністю, симуляторів та інших [10]. Unity іноді використовують у галузях, що не входять до відеоігор, таких як кіно, автомобілебудування, архітектура, інженерія та будівництво.

Unity надає користувачам можливість створювати ігри як у 2D, так і в 3D. Він пропонує первинний API для скриптів в C#, як для редактора Unity у вигляді плагінів, так і для самих ігор, а також функцію взаємодії елементів, перетягуючи їх один на одного. До того, як C# була основною мовою програмування для Unity,

він підтримував Boo, яку було видалено з випуском Unity 5, та версією JavaScript під назвою UnityScript, яка була визначена застарілою в серпні 2017 року після виходу Unity 2017.1, на користь C#.

У межах 2D ігор Unity дозволяє імпорт спрайтів та передовий рендеринг 2D середовища. Для 3D-ігор він дозволяє визначати компресію текстур, виконання методу текстурування mipmaps, та налаштування роздільності екрану для всіх платформ, які підтримуються Unity [11], та забезпечує підтримку bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture та full-screen post-processing effects.

Станом на 2018 рік Unity використовувався для створення приблизно половини нових мобільних ігор на ринку та 60 відсотків всіх програм доповненої реальності та віртуальної реальності.

2.4 Мова програмування C#

Для скриптингу гри буде використовуватися мова програмування C#.

C# – це універсальна, багатопарадигмова мова програмування, основними характеристиками якої є сильна типізація, лексична область видимості, а також наступні парадигми програмування: декларативне програмування, функціональне, узагальнене, об’єктно-орієнтоване, та компонентно-орієнтоване. Вона була розроблена близько 2000 року Microsoft в рамках своєї ініціативи .NET, а згодом затверджена як міжнародний стандарт Ecma (ECMA-334) та ISO (ISO / IEC 23270: 2018). Mono – назва вільного проекту з відкритим кодом з розробки компілятора та виконуючого середовища мови. C# – одна з мов програмування, розроблена для CLI (Common Language Infrastructure – узагальнена мовна інфраструктура).

Стандарт Ecma перераховує наступні основні цілі дизайну для мови C# [12]:

– мова призначена як проста, сучасна, об'єктно-орієнтована мова програмування загального призначення;

– мова та її реалізація повинні забезпечувати підтримку принципів розробки програмних забезпечень, таких як перевірку на сильну типізацію, array bounds checking, виявлення спроб використання неініціалізованих змінних та автоматичне збирання сміття. Важливі ефективність та довговічність програмного забезпечення, а також продуктивність програміста;

– мова призначена для використання при розробці програмних компонентів, придатних для розміщення в розподілених середовищах;

– переносність дуже важлива для вихідного коду та програмістів, особливо тих, хто вже знайомий з C і C++;

– підтримка інтернаціоналізації дуже важлива;

– C# призначена для написання додатків як для розміщених, так і для вбудованих систем, починаючи від дуже великих, які використовують складні операційні системи, і закінчуючи дуже маленькими, що мають спеціальні функції;

– хоча програми C# повинні бути економічними щодо потреб у пам'яті та потужності обробки, мова не повинна конкурувати безпосередньо за продуктивністю та розміром з C або мовою асемблера.

Основний синтаксис мови C# схожий з іншими мовами стилю C, такими як C, C++ та Java.

Розглянемо деякі особливості C#, які відрізняють її від C, C++ та Java, а саме:

– переносність. За задумом C# – це мова програмування, яка найбільш безпосередньо відображає основну CLI. Більшість її вбудованих типів відповідають типам значень, реалізованим у рамках CLI. Однак мовна специфікація не визначає вимог до генерації коду компілятора: тобто, вона не вказує, що компілятор C# повинен орієнтуватися на CLR (Common Language Runtime – загальномовне виконуюче середовище) або генерувати CIL (Common Intermediate Language – узагальнена проміжна мова) або генерувати будь-який інший конкретний формат. Теоретично компілятор C# може генерувати машинний код, як традиційні компілятори C++ або Fortran;

– типізація. C# підтримує декларації змінних без умови з сильною типізацією з ключовим словом var та декларацію масивів з ключовим словом new [], за яким слідує ініціалізатор колекції. C# підтримує суворий булевий тип даних – bool. Оператори, що приймають умови, такі як while та if, потребують вираження типу, який реалізує оператор true, наприклад булівський тип;

– доступ до пам’яті. У C# індикатори адрес пам'яті можуть використовуватися лише в блоках, спеціально позначених як небезпечні, а програми з небезпечним кодом потребують відповідних дозволів для запуску. В основному доступ до об'єктів здійснюється через безпечні посилання на об'єкти, які завжди або вказують на "живий" об'єкт, або мають чітко визначене нульове значення; неможливо отримати посилання на «мертвий» об’єкт (той, який було визначено як сміття), або на випадковий блок пам'яті.

2.5 Висновки до розділу 2

В даному розділі було розглянуто обрану платформу та програмне забезпечення для розробки ігрового додатку

Розглянувши платформу Android було з’ясовано основні особливості операційної системи та технічного забезпечення, з якими вона працює.

Обираючи програмне забезпечення, було розглянуто Android Studio та Unity як найбільш популярні опції при розробці відеоігор. Unity має значну перевагу, адже це в першу чергу ігровий рушій, і він має набагато більше можливостей, створених спеціально для розробки ігор, порівняно с Android Studio.

Тож, спираючись на поставлену задачу та відому інформацію, були обрані наступні засоби розробки: Android, Unity, C#.

3 РОЗРОБКА ІГРОВОГО ДОДАТКУ VEGETABLE SAMURAI ДЛЯ ОС ANDROID

3.1 Загальна будова додатку

В ігровому рушії Unity головним об’єктом в ієрархії об’єктів є сцени (рисунок 3.1).

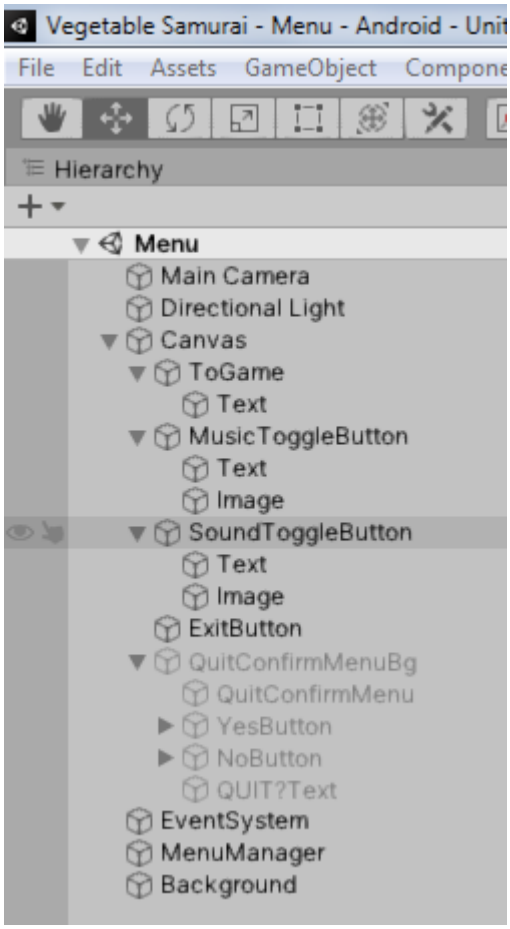

Рисунок 3.1 – Ієрархія об’єктів сцени Menu

В свою чергу сцени вміщають ігрові об’єкти, які можуть містити безліч різноманітних функціональних компонентів (рисунок 3.2): позиція на сцені, камера, джерело світла, джерело звуку, слухач звуку, скрипт, зображення, кнопка, полотно, текст, слід курсору, тощо [13].

Розроблений додаток має 3 сцени, поділені на рівні: 0, 1, та 2.

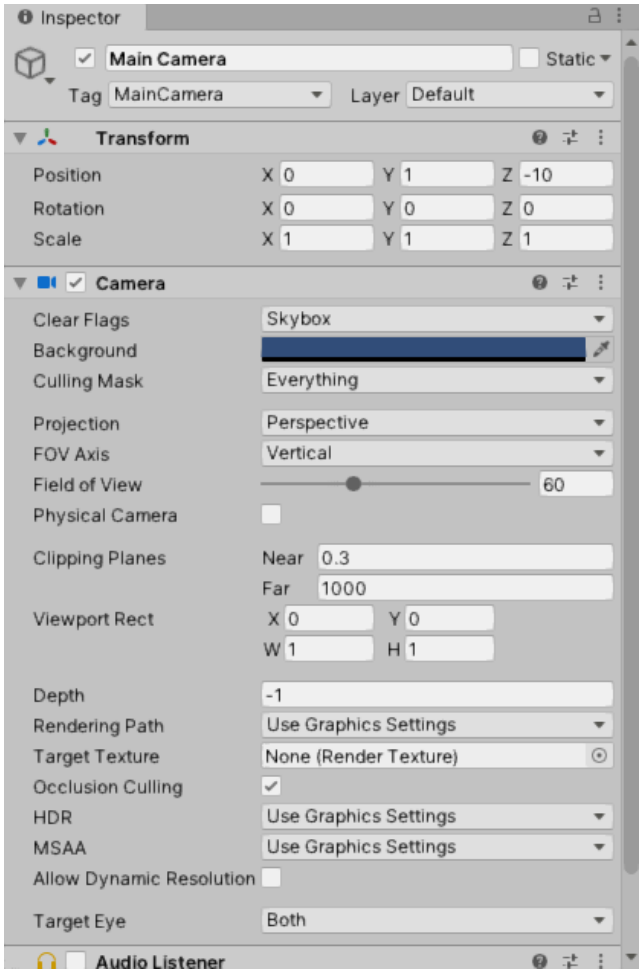

Рисунок 3.2 – Елементи об’єкту Main Camera

Перехід по сценам відбувається за допомогою скриптів автоматично або при натисканні кнопки [14]. При запуску додатку викликається сцена PreLoader з рівнем 0, яка виконує скрипт, що закінчується переходом до сцени Menu з рівнем 1 (рисунок 3.3), яка має кнопки для виходу з додатку або переходу до сцени Game, яка має кнопки, щоб перейти до попередньої сцени (рисунок 3.4).

```
private void Start()
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
        if (PlayerPrefs.GetInt("CrossedC0") == 1)
            source.mute = true;
        if (PlayerPrefs.GetInt("CrossedC1") == 1)
            SoundMuted = true;
        SceneManager.LoadScene("Menu");
    }
```

Рисунок 3.3 – Код старту сцени PreLoader

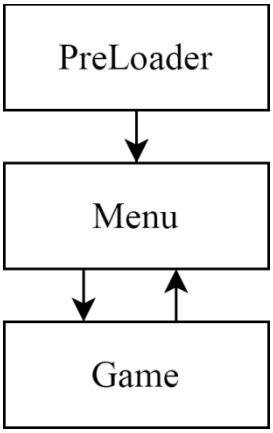

Рисунок 3.4 – Схема переходу по сценам

3.2 Будова та функціонал сцени PreLoader

Сцена PreLoader має всього один ігровий об’єкт SoundManager (рисунок 3.5).

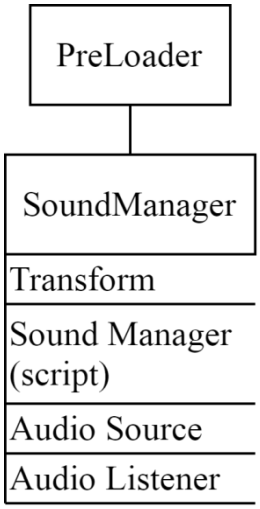

Рисунок 3.5 – Схема структури сцени PreLoader

Розглянемо функціонал елементів об’єкту SoundManager.

Елемент Audio Source є джерелом музики (рисунок 3.6).

Audio Listener – потрібен для сприйняття джерел звуку у просторі та їх програвання.

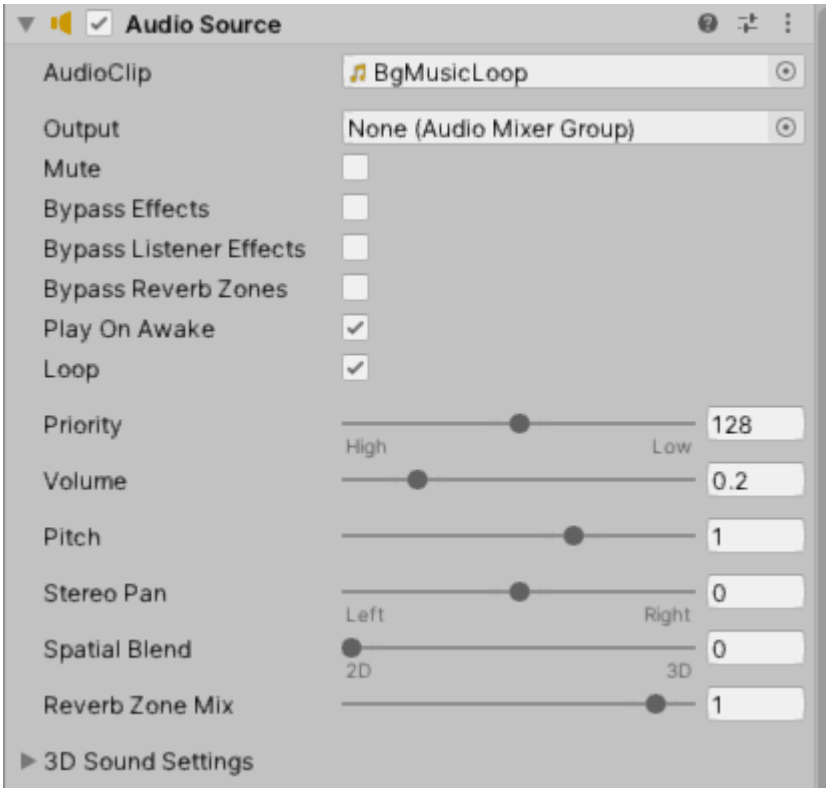

Рисунок 3.6 – Елемент Audio Source

Скрипт Sound Manager виконує багато функцій. При першому завантаженні додатку він створює копію об’єкту SoundManager, яка не буде знищуватися при переходжені між сценами як звичайні об’єкти [15].

Створення копії об’єкту SoundManager важливо, тому що даний об’єкт відповідає за звукові функції, які будуть використовуватися в обох інших сценах, в тому числі при переході між ними, а також функцію виходу з додатку за допомогою кнопок в головному меню. До звукових функцій скрипту відносяться робота музики, звуків (рисунок 3.7), кнопок включення або виключення музики або звуків.

```
    public void PlaySound(int soundIndex)
        {
            if (SoundMuted == false)
    AudioSource.PlayClipAtPoint(allSounds[soundIndex],transform.pos
ition);
            else return;
        }
```

Рисунок 3.7 – Код програвання звуку

3.3 Будова та функціонал сцени Menu

Сцена Menu має набагато більш об’єктів, аніж попередня сцена (рисунок 3.8).

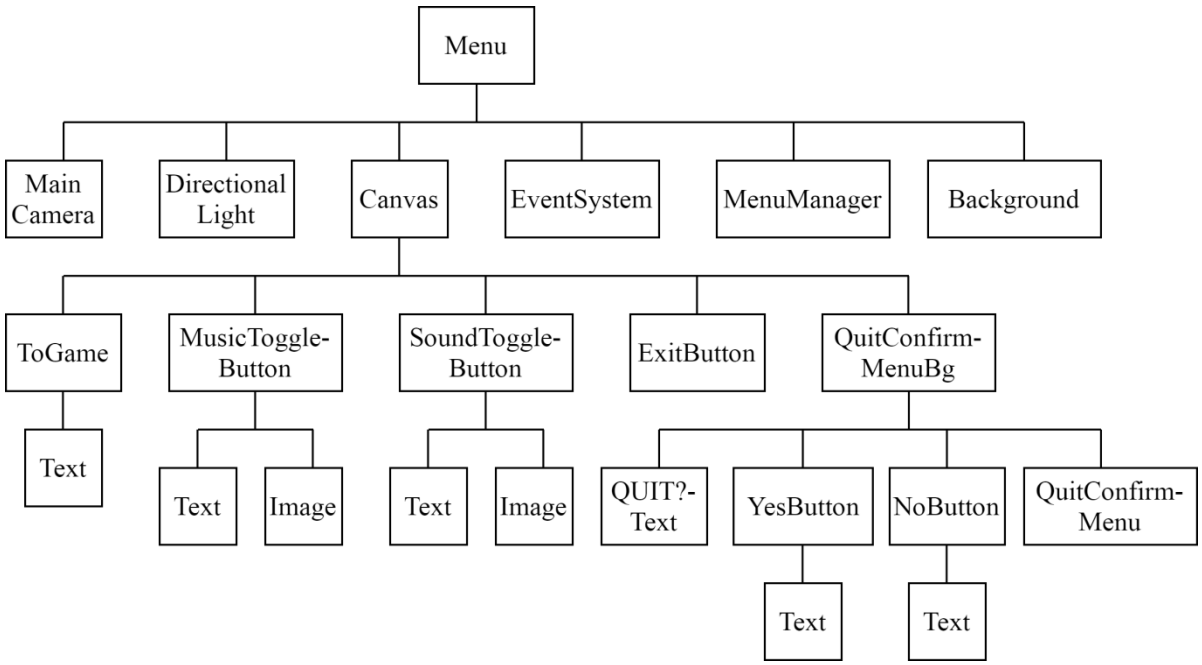

Рисунок 3.8 – Схема структури об’єктів сцени Menu

Розглянемо функціонал об’єктів даної сцени.

Об’єкт Main Camera визначає параметри камери, яка відповідає за сприйняття і програвання зображення.

Directional Lights – визначає параметри освітлення, що впливає на візуальні якості зображення.

Event System – відповідає за деякі параметри взаємодії користувача з додатком, як то кількість можливих дій в секунду або функціонування деяких кнопок, як enter або submit [16].

Menu Manager – містить скрипт Menu Manager.

Background – задає зображення на задньому плані.

Canvas, QuitConfirmMenuBg, QuitConfirmMenu – містять зображення для вікон меню та визначають ієрархію вікон меню.

ToGame – викликає функцію для переходу до сцени Game (рисунок 3.9).

```
IEnumerator ToGameCoroutine()
{
    SoundManager.Instance.PlaySound(0);
    yield return new WaitForSeconds(0.261f);
    SceneManager.LoadScene("Game");
}
public void ToGame()
{
    StartCoroutine(ToGameCoroutine());
}
```

Рисунок 3.9 – Код переходу до сцени Game

ExitButton – викликає меню підтвердження виходу з додатку (рисунок 3.10).

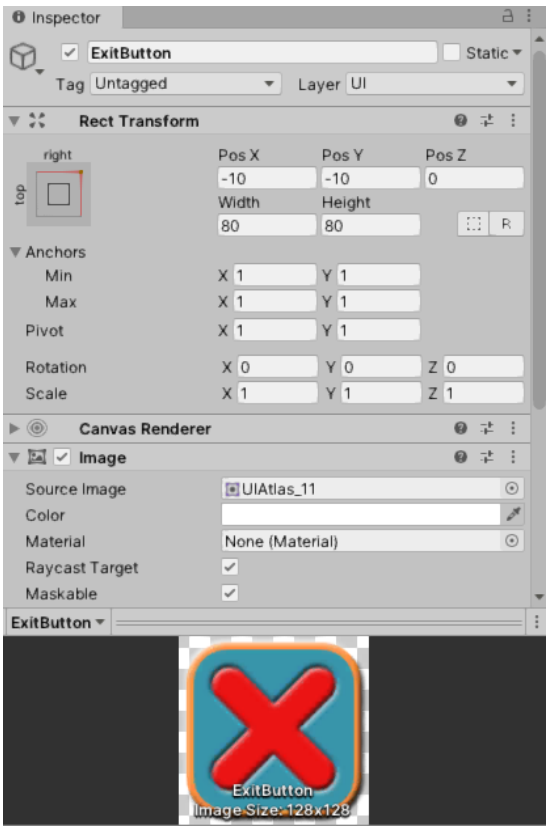

Рисунок 3.10 – Елемент ExitButton

YesButton – виконує функцію виходу з додатку.

NoButton – закриває меню підтвердження виходу з додатку.

Text, QUIT?Text – відповідають за параметри тексту.

Image – містить зображення закреслених кіл, які з’являються над кнопками музики або звуків, якщо музика або звуки були вимкнуті.

MusicToggleButton – вимикає або вмикає музику.

SoundToggleButton – вимикає або вмикає звуки: натискання кнопки, розрізання овочу або бомби, втрати одиниці помилок (рисунок 3.11).

```
public void ToggleGameSoundsButton() {
      SoundManager.Instance.ToggleGameSounds();
      if (CrossedCircles[1].enabled == false) {
         CrossedCircles[1].enabled = true;
         PlayerPrefs.SetInt("CrossedC1", 1);
      }
      else {
         CrossedCircles[1].enabled = false;
         PlayerPrefs.SetInt("CrossedC1", 0);
      } }
```

Рисунок 3.11 – Код вимикання або вмикання звуків

При завантаженні сцени Menu ми отримуємо можливість перейти до сцени Game, натиснувши кнопку “PLAY” (рисунок 3.12), виключити або включити музику або звуки (рисунок 3.13), або викликати меню підтвердження виходу з додатку та вийти з додатку (рисунок 3.14).

Рисунок 3.12 – Вигляд головного меню

Рисунок 3.13 – Вигляд кнопок музики та звуків, якщо музику та звуки було вимкнуто

Рисунок 3.14 – Вигляд меню підтвердження виходу з додатку

Скрипт Menu Manager відповідає за наступні функції:

– перехід до сцени Game при натисканні кнопки “PLAY”;

– вимкнення або ввімкнення музики (рисунок 3.15) або звуків при натисканні на відповідну кнопку;

```
public void ToggleBgMusicButton() {
      SoundManager.Instance.ToggleBgMusic();
      if (CrossedCircles[0].enabled == false) {
         CrossedCircles[0].enabled = true;
         PlayerPrefs.SetInt("CrossedC0", 1);
      }
      else {
         CrossedCircles[0].enabled = false;
         PlayerPrefs.SetInt("CrossedC0", 0);
      } }
```

Рисунок 3.15 – Код вимикання або вмикання музики

– відкривання або закривання меню підтвердження виходу з додатку (рисунок 3.16);

```
public void XExitButton()
{
   ExitConfirmMenu.SetActive(true);
}
```

Рисунок 3.16 – Код відкриття меню підтвердження виходу з додатку

– вихід з додатку при натисканні кнопки “YES” в меню підтвердження виходу з додатку (рисунок 3.17).

```
public void YesExitButton()
   {
      SoundManager.Instance.QuitApp();
   }
```

Рисунок 3.17 – Код виходу з додатку

3.4 Будова та функціонал сцени Game

Сцена Game має найбільшу кількість об’єктів відносно попередніх сцен (рисунок 3.18).

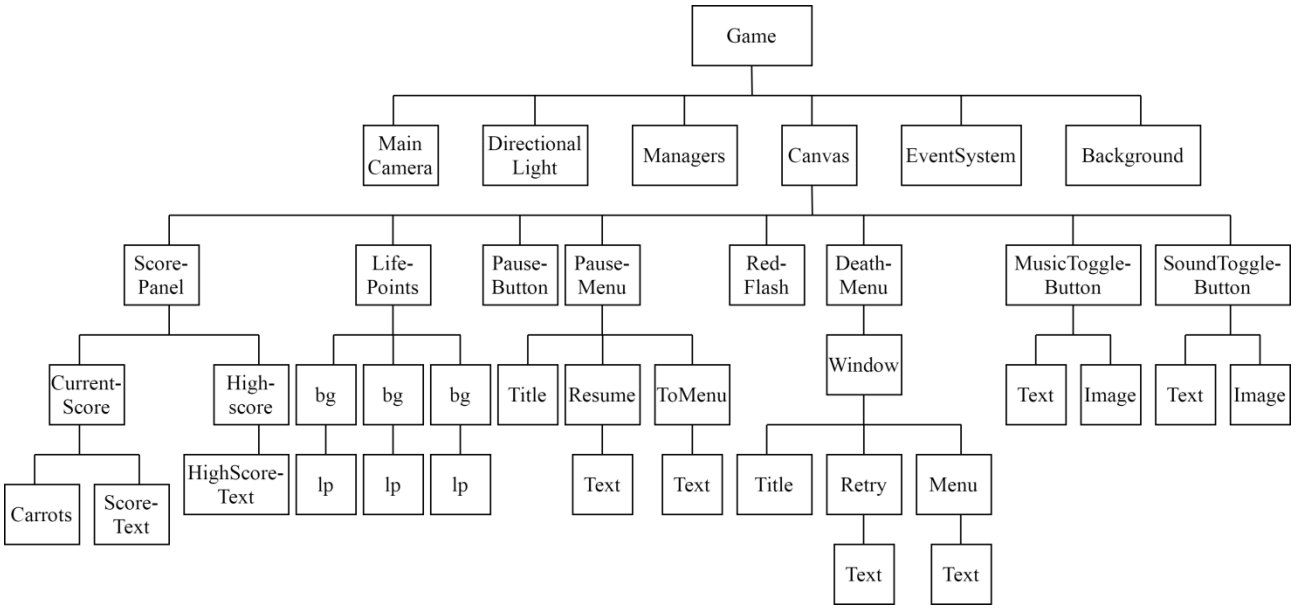

Рисунок 3.18 – Схема структури об’єктів сцени Game

Розглянемо функціонал об’єктів даної сцени.

Об’єкти Main Camera, Directional Light, Canvas, EventSystem, Background, MusicToggleButton, SoundToggleButton, Text, Image відповідають об’єктам з однаковим ім’ям зі сцени Menu та виконують однакові функції.

ScorePanel, CurrentScore, Highscore, LifePoints, PauseMenu, DeathMenu, Window – містять зображення для вікон меню та визначають ієрархію вікон меню.

Carrots – містить зображення розрізаної моркви поруч з індикатором очок.

ScoreText – містить текст очок, який змінюється по ходу гри.

HighScoreText – містить текст максимальної кількості очок, набраних на протязі однієї гри.

lp – містить зображення індикаторів наявних одиниць помилок.

bg – містить зображення індикаторів втрачених одиниць помилок.

PauseButton – кнопка паузи або продовження гри (рисунок 3.19).

```
public void PauseGame() {
        SoundManager.Instance.PlaySound(0);
        pauseMenu.SetActive(!pauseMenu.activeSelf);
        MusicToggleButton.SetActive(!MusicToggleButton.activeSelf);
        SoundToggleButton.SetActive(!SoundToggleButton.activeSelf);
 if (PlayerPrefs.GetInt("CrossedC0") == 1) CrossedCircles[0].enabled = true;
 if (PlayerPrefs.GetInt("CrossedC1") == 1) CrossedCircles[1].enabled = true;
        isPaused = pauseMenu.activeSelf;
        Time.timeScale = (Time.timeScale == 0) ? 1 : 0 ;
        SoundManager.Instance.PlaySound(0); }
```

Рисунок 3.19 – Код паузи гри

Title – містить текст титулів меню паузи та програшу.

Resume – кнопка продовження гри.

ToMenu, Menu – кнопка переходу до сцени Menu.

RedFlash – містить напівпрозоре полотно червоного кольору, яке тимчасово на одну мить з’являється при втраті одиниці помилки.

Retry – кнопка початку нової гри.

При завантаженні сцени Game одразу починається нова гра (рисунок 3.20).

Рисунок 3.20 – Початок нової гри

Задача гравця - розрізати овочі (рисунок 3.21), проводячи по ним лінію, та уникати бомб (рисунок 3.22).

```
public void Slice()
    {
        if(isSliced) return;
        if (verticalVelocity < 0.5f) verticalVelocity = 0.5f;
        speed = speed * 0.5f;
        isSliced = true;
        SoundManager.Instance.PlaySound(1);
        GameManager.Instance.IncrementScore(1);
        GameManager.Instance.IncrementDifficulty(0.01f);
    }
```

Рисунок 3.21 – Код розрізання овочу

При розрізанні овочів зменшується інтервал появи овочів та бомб, а при втраті одиниці помилки інтервал стає таким, як на початку гри, тобто гра має “гнучку” систему складності.

Рисунок 3.22 – Розрізання овочу, бомба, слід після проведення по екрану

Якщо овоч падає за межі екрану нерозрізаним або гравець розрізає бомбу (рисунок 3.23), він втрачає одиницю помилки, всі овочі на екрані розрізаються без збільшення очок, та інтервал появи овочів та бомб встановлюється таким, як на початку гри (рисунок 3.24).

```
public void Slice()
    {
        if(isSliced) return;
        if (verticalVelocity < 0.5f)
            verticalVelocity = 0.5f;
        speed = speed * 0.5f;
        isSliced = true;
        SoundManager.Instance.PlaySound(2);
        GameManager.Instance.LoseLP();
    }
```

Рисунок 3.23 – Код розрізання бомби

Рисунок 3.24 – Розрізана бомба, втрачена одиниця помилок, овочі розрізані без збільшення очок

Гравець має можливість натиснути на паузу, в меню паузи вимкнути або ввімкнути музику або звуки, перейти до сцени Menu (рисунок 3.25) або продовжити гру (рисунок 3.26).

```
IEnumerator ToMenuCoroutine(){
        Time.timeScale = 1;
        SoundManager.Instance.PlaySound(0);
        yield return new WaitForSeconds(0.261f);
        SceneManager.LoadScene("Menu");
    }
public void ToMenu()
    {
        StartCoroutine(ToMenuCoroutine());
    }
```

Рисунок 3.25 – Код переходу до сцени Menu

Рисунок 3.26 – Меню паузи

При програші в меню програшу можна також вимкнути або ввімкнути музику або звуки, перейти до сцени Menu, а також почати нову гру (рисунок 3.27)

Рисунок 3.27 – Меню програшу

При створенні об’єкту овочу (рисунок 3.28) обирається початковий спрайт – зображення об’єкту, відповідне до використаного префабу – попередньо створеного об’єкту, з якого в майбутньому будуть створюватися копії [17].

```
    private Vegetable GetVegetable()
        {
            Vegetable v = veggies.Find(x => !x.IsActive);
            if(v == null) {
                v  =  Instantiate(vegetablePrefab[Random.Range(0,8)])
.GetComponent<Vegetable>();
                veggies.Add(v);
            }
            return v;
        }
```

Рисунок 3.28 – Код створення об’єкту овочу

При розрізанні овочу або бомби спрайт змінюється згідно масиву спрайтів, створеного із зображення, поділеного за фреймами (рисунок 3.29), тим самим створюється анімація [18].

Рисунок 3.29 – Приклад зображення для поділу на фрейми

Також для поліпшення різноманіття руху об’єктів при їх створенні об’єктам надається випадкова швидкість крутіння.

В сцені Game використовуються скрипти GameManager, Vegetable та Bomb.

Скрипт Vegetable відповідає за наступні функції:

– запуск овочу: встановлення початкової позиції об’єкту, вектору руху, швидкості руху, фактору гравітації, швидкості крутіння, присвоєння спрайту;

– розріз овочу: зміна швидкості польоту для надання візуального ефекту взаємодії з об’єктом при розрізі, програвання звуку розрізу, збільшення очок, зменшення інтервалу появи овочів та бомб;

– розріз овочу без збільшення очок (рисунок 3.30): використовується при втраті одиниці помилки щоб створити більш цікавий візуальний ефект позбавлення нерозрізаних овочів;

```
public void NoPointsSlice()
    {
        if(isSliced)
            return;
        if (verticalVelocity < 0.5f)
            verticalVelocity = 0.5f;
        speed = speed * 0.5f;
        isSliced = true;
        SoundManager.Instance.PlaySound(1);
    }
```

Рисунок 3.30 – Код розрізу овочу без збільшення очок

– оновлення об’єкту овочу кожний фрейм: зміна позиції об’єкту для створення ефекту польоту, зміна спрайту при розрізі для ефекту анімації, втрата одиниці помилок та знищення об’єкту при падінні за межі екрану.

Скрипт Bomb майже повністю однаковий зі скриптом Vegetable, основними відмінностями є:

– при розрізі бомби не збільшуються очки, програється звук вибуху, втрачається одиниця помилок;

– при падінні не втрачається одиниця помилок.

Для взаємодії з об’єктами овочів та бомб дотиком використовуються координати дотику в просторі та координати елементу об’єкту Box Collider 2D (рисунок 3.31) – квадрат, що оточує об’єкт [19].

Також для розрізання об’єкту недостатньо простого натискання, потрібно провести через коллайдер щонайменш коротку лінію, розмір якої визначається виміром останньої та теперішньої позиції дотику в просторі.

Скрипт GameManager є найбільшим скриптом проекту та відповідає за наступні функції:

– встановлення мінімального часу натиску, потрібного для створення сліду проведення;

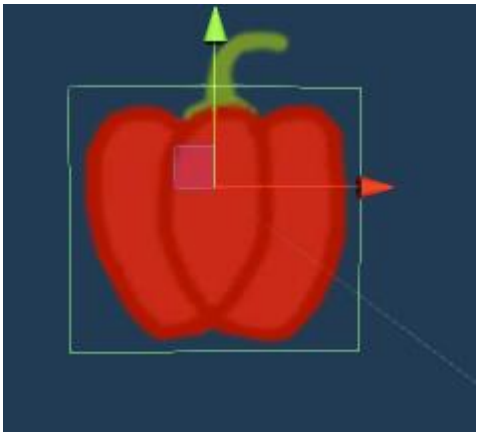

Рисунок 3.31 – Зелений квадрат – візуальний вигляд елементу Box Collider 2D

– початок нової гри: встановлення до початкових значень інтервалу появи нових об’єктів овочів та бомб, очок, одиниць помилок, видалення всіх старих овочів та бомб;

– створення нових об’єктів овочів або бомб (рисунок 3.32) в певному інтервалі: ініціалізація об’єкту, присвоєння відповідного префабу – попередньо створеного об’єкту, занесення до списку;

```
    private Bomb GetBomb(){
            Bomb b = bombs.Find(x => !x.IsActive);
            if(b == null)
            {
                b = Instantiate(bombPrefab[0]).
GetComponent<Bomb>();
                bombs.Add(b);
            }
            return b;
        }
```

Рисунок 3.32 – Код створення об’єкту бомби

– зменшення інтервалу появи овочів та бомб для підвищення складності гри;

– втрата одиниці помилок: виключення зображення одиниці помилок, розріз овочів без збільшення очок, встановлення інтервалу появи овочів та бомб рівному початковому, червоний спалах для візуальної індикації, програш при втраті всіх трьох одиниць помилок;

– пауза гри;

– перехід до сцени Menu;

– підрахунок очок на протязі гри та присвоєння значення очок значенню найвищих очок в разі встановлення нового рекорду (рисунок 3.33);

```
public void IncrementScore(int scoreAmount) {
        score += scoreAmount;
        scoreText.text = score.ToString();
        if (score > highscore)
        {
            highscore = score;
          highscoreText.text = "BEST: " + highscore.ToString();
            PlayerPrefs.SetInt("Score",highscore);
        }
    }
```

Рисунок 3.33 – Код зміни індикаторів очок

– оновлення при кожному фрейму: запуск овочів та бомб, виявлення взаємодії між координатами проведеної лінії та координат коллайдерів об’єктів овочів та бомб.

3.5 Висновки до розділу 3

Було розроблено ігровий додаток Vegetable Samurai для ОС Android.

Користувацький інтерфейс має контрастний дизайн без зайвих деталей.

Ігровий процес виконує функції польоту овочів та бомб, їх розрізання, підрахунок очок, втрата одиниць помилок та програш після втрати трьох одиниць помилок. Зовнішній вигляд та розмір овочів визначається попередньо створеним набором об’єктів овочів. Слід після проведення по екрану, крутіння овочів та бомб при польоті, зміна їх швидкості та спрайтів при розрізі - всі ці функції створюють ефекти анімації. Звуки при взаємодії з елементами меню, розрізанні овочів або бомб, а також музика доповнюють анімацію для покращення відгуку гри на дії гравця.

4 ОХОРОНА ПРАЦІ

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» [20] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

4.1.1 Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

Державна політика в галузі охорони праці визначається відповідно до Конституції України Верховною Радою України і спрямована на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням. Відповідно до статті 3 Закону України «Про охорону праці» [20] (далі – Закону) законодавство про охорону праці складається з Закону, Кодексу законів про працю України [21], Закону України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності" [22] та прийнятих відповідно до них нормативно-правових актів, норм міжнародного договору (ратифіковані Конвенції і Рекомендації МОТ, директиви Європейської Ради).

4.1.2 Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [23].

Обов'язковими вимогами враховане наступне:

– ознайомлення з правилами безпеки праці, одержання відповідних інструктажів засвідчується у журналі інструктажів.

– перед допуском до самостійної роботи кожен працівник має право на навчання з питань охорони праці і роботодавець зобов'язаний, і проводить таке навчання у вигляді двох інструктажів з питань охорони праці:

1) вступного, який проводять працівники служби охорони праці об'єкта господарювання з усіма працівниками, яких приймають на роботу незалежно від їхньої освіти та стажу роботи за програмою, в якій подають загальні питання охорони праці із врахуванням її особливостей на об'єкті господарювання;

2) первинного, який проводять керівники структурних підрозділів на місці праці з кожним працівником до початку їхньої роботи на цьому робочому місці.

Проходження працівником цих інструктажів з питань охорони праці підтверджується записами у відповідних журналах обліку інструктажів і скріплюється підписами осіб, які проводили інструктажі та осіб, які отримали інструктажі.

3) Повторний (не рідше одного разу в 6 місяців);

4) Позаплановий (при зміні правил охорони праці);

5) Поточний (проводять з працівниками перед виконанням робіт, на яких оформляється наряд-допуск).

– обов'язкові організаційні заходи перед початком, під час і після завершення роботи повинні включати перевірку (візуально) наявності і справності електрообладнання та його заземлення, а під час виконання роботи вимогу «не

залишати без нагляду обладнання, яке працює». Після закінчення роботи - вимагається прибирання робочого місця, відключення всіх електроприладів від електромережі.

4.2 Аналіз стану умов праці

Робота над створенням системи оптимізації запитів до бази даних проходитиме в кімнаті квартири. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	3,9
Ширина, м	3,4
Висота, м	2,6
Площа, $м^2$	13,26
Об'єм, $м^3$	34,476

Згідно з ДСН 3.3.6.042-99 [24] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним згідно статті Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 [25] і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	730	680 ÷800
Висота простору для ніг, мм	710	не менше 600
Ширина простору для ніг, мм	590	не менше 500
Глибина простору для ніг, мм	670	не менше 650
Висота поверхні сидіння, мм	480	400 ÷500
Ширина сидіння, мм	420	не менше 400
Глибина сидіння, мм	410	не менше 400
Висота поверхні спинки, мм	580	не менше 300
Ширина опорної поверхні спинки, мм	510	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	430	400
Відстань від очей до екрану дисплея, мм	780	700 ÷800

Приміщення кабінету має об’єм 34,476 м^3, площу – 13,26 м^2.

Температура в приміщенні протягом року коливається у межах 19–24°С, відносна вологість – близько 50%. Система вентилювання приміщення – природна неорганізована, а опалення – централізоване.

Розміщення вікон забезпечує природне освітлення з коефіцієнтом природного освітлення не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою чотирьох люмінесцентних ламп, забезпечує рівень освітленості не менше 200 Лк.

За ступенем пожежної безпеки приміщення належить до категорії В.

4.3 Виробнича санітарія

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00-7.15-18 [28] «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Освітлення

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Освітленість приміщення має велике значення при роботі на ПЕОМ. Вона багато в чому визначається колірною і мережевий обстановкою. Для зменшеного поглинання світла стеля і стіни вище панелей (1,5-1,7м.). Якщо вони не облицьовані звукопоглинальним матеріалом, фарбуються білою водоемульсійною фарбою (коефіцієнт відбиття повинен бути не менше 0,7). Для забарвлення стіни панелей рекомендується віддавати перевагу світлим фарбам.

Природне освітлення, коли робочі місця з ПЕОМ розташовуються в один ряд по довжині приміщення на відстані 0,8 - 1,0 м від стіни з віконними прорізами, і екрани знаходяться перпендикулярно цієї стіни. Основний потік природного світла при цій повинен бути зліва. Не допускається спрямування основного світлового потоку природного світла праворуч, ззаду і спереду працює на ПЕОМ. Оптимальна відстань очей до екрана відео монітора повинна становити 60-70 см, допустиме не менше 50 см. Розглядати інформацію ближче 50 см не рекомендується.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2018 [27]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДСанПІН 3.3.2.007-98 [25] і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b = (1/8 \div 1/10) \cdot S_n \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

S_n = a · b = 3,9 · 3,4 = 13,26м^2

$S_{вік}$ = 1/8 · 13,26 ≈ 1,66м^2

Приймаємо 1 вікно площею S = 1,66м^2.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для

створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 13,26 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 1;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 13{,}26 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 1} \approx 2$$

Приймаємо освітлювальну установку, яка складається з двох світильників, який складаються з 2-х люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4.2 Вентилювання

Здійснюється провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.4.3 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі)

Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з \cdot R_з \cdot \eta_п}, \quad (4.3)$$

де $R_з$ – опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

$R_п$ – опір опори, яке з'єднує заземлювачі, Ом;

n – кількість заземлювачів;

$\eta_з$ – коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; $\eta_з$ = 0,7

$\eta_п$ – коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; $\eta_п$ = 0,5;

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(ln \frac{2 \cdot l}{d} + \frac{1}{2} ln \frac{4 \cdot t + l}{4 \cdot t - l} \right) \quad (4.4)$$

де ρ - питомий опір ґрунту, залежить від типу ґрунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м.

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3} \cdot \left(ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} ln \frac{4 \cdot 2 + 3}{4 \cdot 2 - 3} \right) = 110, \text{Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_n = \frac{\rho}{2\pi \cdot L} \cdot ln \frac{2 \cdot L^2}{b \cdot t_1} \quad (4.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі:

$П_{буд.}$ = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t_1 - глибина заземлення від рівня землі, м; t_1 = 0,5 м.

$$R_n = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot ln \frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99, \text{Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з} \quad (4.6)$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{ззп} < 4$ Ом.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

Усунення шуму в приміщенні є однією з найскладніших проблем, оскільки джерела шуму різноманітні й потребують комплексу заходів технічного, організаційного і медичного характеру на всіх стадіях проектування, будівництва, експлуатації машин і устаткування.

На місці праці можливі аварійні ситуації техногенного характеру та загрози природного характеру, що можуть перерости у надзвичайні ситуації. Серед надзвичайних ситуацій техногенного характеру домінують пожежі та вибухи, а серед небезпек природного характеру — аномальні гідрометеорологічні явища та медико-біологічні загрози.

Загальні вимоги електробезпеки повинні відповідати ГОСТ 12.1.045-84[31]. Для захисту від уражень електричним струмом використовують захисне заземлення. Воно повинно захищати людей від уражень електричним струмом у випадку дотику до металевих неструмопровідних частин, які можуть опинитись під напругою внаслідок пошкодження ізоляції, це досягається з’єднанням металевих частин електроустановок з землею, або її еквівалентом.

4.6 Висновки до розділу 4

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної магістерської роботи було розробити ігровий додаток Vegetable Samurai для ОС Android, і як результат був створений ігровий додаток.

Аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для умов праці з використанням персонального комп'ютера на якому буде розроблятися ігровий додаток.

ВИСНОВКИ

У межах дипломного проекту виконано розробку ігрового додатку Vegetable Samurai для ОС Android.

Аналіз сучасних відеоігор допоміг обрати жанр та тип розроблюваної гри.

З розглянутих платформ було обрано ОС Android через її популярність та доступність апаратних та програмних засобів розробки.

Було поставлено задачу, в якій визначено головні вимоги до користувацького інтерфейсу та механіки ігрового процесу.

Ознайомлення з характеристикою та особливостями ОС Android допомогло краще зрозуміти завдання технічної реалізації та підбору програмного забезпечення розробки додатку.

Основним програмним забезпеченням для створення гри було обрано ігровий рушій Unity. Спрямованість Unity на розробку відеоігор дозволила швидше розробити ігровий додаток.

Популярність мови C# гарантувала численну кількість якісної літератури, що полегшило завдання ознайомлення з мовою програмування та вирішення помилок при написанні коду додатку.

Основні особливості створеного додатку:

– контрастний інтерфейс без зайвих деталей;

– графічні та звукові реакції на взаємодію користувача з грою;

– “гнучка” складність гри – складність збільшується, чим довше гравець вдало розрізає овочі та зменшується, коли він втрачає одиницю помилок;

– мобільність – можливість швидко вийти з гри та повернутися до неї.

Таким чином, було проаналізовано предметну область, поставлено задачі, обрано засоби розробки та створено додаток, повністю відповідний поставленим задачам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Richard M. Ryan, Edward L. Deci. “Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being” (PDF). [Електронний ресурс] // University of Rochester. URL: https://selfdeterminationtheory.org/SDT/documents/2000_RyanDeci_SDT.pdf (дата звернення: 30.04.2020)

2. Jamie Madigan. "The Psychology of Video Game Avatars". [Електронний ресурс] // The Psychology of Video Games. URL: http://www.psychologyofgames.com/2013/11/the-psychology-of-video-game-avatars/ (дата звернення: 30.04.2020)

3. "Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018". [Електронний ресурс] // Gartner, Inc. Gartner. URL: https://web.archive.org/web/20180829072934/https://www.gartner.com/newsroom/id/3876865 (дата звернення: 4.05.2020)

4. Thom Holwerda. The second operating system hiding in every mobile phone. [Електронний ресурс] // OSnews. URL: https://www.osnews.com/story/27416/the-second-operating-system-hiding-in-every-mobile-phone/ (дата звернення: 5.05.2020)

5. Ron Amadeo. “Google’s iron grip on Android: Controlling open source by any means necessary”. [Електронний ресурс] // Ars Technica. URL: https://arstechnica.com/gadgets/2018/07/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/ (дата звернення: 5.05.2020)

6. Bertel King. “Is Android Really Open Source? And Does It Even Matter?”. [Електронний ресурс] // MakeUseOf. URL: https://www.makeuseof.com/tag/android-really-open-source-matter/ (дата звернення: 6.05.2020)

7. "Touch Devices". [Електронний ресурс] // Android Open Source Project. URL: https://web.archive.org/web/20120125061950/http://source.android.com/tech/input/touct-devices.html (дата звернення: 7.05.2020)

8. Adam Sinicki. "How to install the Android SDK (Software Development Kit)". [Електронний ресурс] // Android Authority. URL: https://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/ (дата звернення: 8.05.2020)

9. Xavier Ducrohet, Tor Norbye, Katherine Chou. "Android Studio: An IDE built for Android". [Електронний ресурс] // Android Developers Blog. Google. URL: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html (дата звернення: 8.05.2020)

10. Samuel Axon. “Unity at 10: For better—or worse—game development has never been easier”. [Електронний ресурс] // Ars Technica. URL: https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/ (дата звернення: 8.05.2020)

11. "Unity – Multiplatform". [Електронний ресурс] // Unity. Unity Technologies. URL: https://unity.com/features/multiplatform (дата звернення: 11.05.2020)

12. C# Language Specification (PDF). [Електронний ресурс] // Ecma International. URL: https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf (дата звернення: 11.05.2020)

13. Piotr Korzuszek. “7 Ways to Keep Your Unity Project Organized”. [Електронний ресурс] // The Knights of Unity. URL: https://blog.theknightsofunity.com/7-ways-keep-unity-project-organized/ (дата звернення: 16.05.2020)

14. “Changing Scenes in Unity 3D”. [Електронний ресурс] // Studytonight. URL: https://www.studytonight.com/game-development-in-2D/changing-scene (дата звернення: 16.05.2020)

15. Christian Engvall. “DontDestroyOnLoad tutorial”. [Електронний ресурс] // Honkbark Studios. URL: https://honkbarkstudios.com/developer-blog/dontdestroyonload-tutorial/ (дата звернення: 18.05.2020)

16. DyadichenkoGA. “Работа с EventSystem в Unity. Базовые вещи в работе с UI”. [Електронний ресурс] // Хабр. URL: https://habr.com/ru/post/359106/ (дата звернення: 18.05.2020)

17. “Mastering the basics of Unity: Understanding prefabs”. [Електронний ресурс] // Pluralsight. URL: https://www.pluralsight.com/blog/tutorials/mastering-basics-unity-understanding-prefabs (дата звернення: 20.05.2020)

18. John Horton. “Simple 2D Sprite-sheet animations in Unity”. [Електронний ресурс] // Game Code School. URL: http://gamecodeschool.com/unity/simple-2d-sprite-sheet-animations-in-unity/ (дата звернення: 20.05.2020)

19. Romeo Violini. “Unity Collision Detection 2D Everything You Need To Know + Examples”. [Електронний ресурс] // Gamedevelopertips. URL: http://gamedevelopertips.com/unity-collision-detection-2d/ (дата звернення: 20.05.2020)

20. Закон України "Про охорону праці". Вводиться в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669. - Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/2694-12

21. Кодекс законів про працю України. Затверджується Законом № 322-VIII від 10.12.71 ВВР, 1971. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/322-08

22. Закон України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності". Наказ від 21 грудня 2000 року N 2180-III. Режим доступу: www. URL: https://dnaop.com/html/2065/doc-zakon-ukrajini-pro-zagalynoobovjazkove-derzhavne-socialynestrahuvannya-vid-neshhasnogo-vipadku-na-virobnictvi-ta-profesijnogo-z

23. Про затвердження Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.12-05). Наказ від 26.01.2005 №15. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0231-05

24. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. Постанова N 42 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

25. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. Затверджено Постановою Головного державного санітарного лікаря України 10 грудня 1998 р. N 7. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

26. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом. Наказ від 1 липня 2016 року N 204. Режим доступу: www. URL: http://epicentre.co.ua/dstu/doc28522.html

27. ДБН В.2.5-28:2018 «Природне і штучне освітлення». Режим доступу: www. URL: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf

28. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за № 508/31960. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

29. ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». Наказ від 15.06.2016 №158. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0158858-16

30. Закон України «Про охорону навколишнього природного середовища» . Вводиться в дію Постановою ВР № 1268-XII від 26.06.91, ВВР, 1991, № 41, ст.547. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/1264-12

31. ГОСТ 12.1.045-84 Електростатичні поля, припустимі рівні на робочих місцях і вимоги до проведення контролю. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48138

Додаток А

Лістинг програмного коду

GameManager.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5   using UnityEngine.SceneManagement;
6
7   public class GameManager : MonoBehaviour
8   {
9   public static GameManager Instance{set;get;}
10
11  private const float REQUIRED_SLICEFORCE = 600.0f;
12
13  public GameObject[] vegetablePrefab;
14  public GameObject[] bombPrefab;
15  public GameObject RedFlash;
16  public float deltaSpawn = 1.0f; // швидкість появи овочів
17  public float DeltaTrailTime = 0.8f;
18
19  private bool isPaused;
20  private List<Vegetable> veggies = new List<Vegetable>();
21  private List<Bomb> bombs = new List<Bomb>();
22  private float lastSpawn;
23  private float lastBombSpawn;
24  private Vector3 lastMousePos;
25  private Collider2D[] veggiesCols;
26  private Collider2D[] bombsCols;
27  private float LastTrailTime;
28  private TrailRenderer tr;
29
30
31  // користувацький інтерфейс
32  private int score;
33  private int highscore;
34  private int lifepoint;
35  public Text scoreText;
36  public Text highscoreText;
37  public Image[] lifepoints;
38  public GameObject pauseMenu;
39  public GameObject deathMenu;
40  public GameObject MusicToggleButton;
41  public GameObject SoundToggleButton;
42  public Image[] CrossedCircles;
43
44  IEnumerator ToMenuCoroutine()
45  {
```

```
46  Time.timeScale = 1;
47  SoundManager.Instance.PlaySound(0);
48  yield return new WaitForSeconds(0.261f);
49  SceneManager.LoadScene("Menu");
50  }
51
52  IEnumerator RedFlashCoroutine()
53  {
54  RedFlash.SetActive(true);
55  SoundManager.Instance.PlaySound(3);
56  yield return new WaitForSeconds(0.1f);
57  RedFlash.SetActive(false);
58  }
59
60  IEnumerator DeathMenuCoroutine()
61  {
62  yield return new WaitForSeconds(0.1f);
63  deathMenu.SetActive(true);
64  MusicToggleButton.SetActive(!MusicToggleButton.activeSelf);
65  SoundToggleButton.SetActive(!SoundToggleButton.activeSelf);
66  if (PlayerPrefs.GetInt("CrossedC0") == 1)
67  CrossedCircles[0].enabled = true;
68  if (PlayerPrefs.GetInt("CrossedC1") == 1)
69  CrossedCircles[1].enabled = true;
70  }
71
72  private void Awake()
73  {
74  Instance = this;
75  }
76
77  private void Start()
78  {
79  tr = GetComponent<TrailRenderer>();
80  if (PlayerPrefs.GetInt("CrossedC0") == 1)
81  CrossedCircles[0].enabled = true;
82  if (PlayerPrefs.GetInt("CrossedC1") == 1)
83  CrossedCircles[1].enabled = true;
84  veggiesCols = new Collider2D[0];
85  bombsCols = new Collider2D[0];
86  NewGame();
87  }
88
89  public void NewGame()
90  {
91  deltaSpawn = 1.0f;
92  score = 0;
93  lifepoint = 3;
94  pauseMenu.SetActive(false);
95  scoreText.text = score.ToString();
96  highscore = PlayerPrefs.GetInt("Score", 0);
97  highscoreText.text = "BEST: " + highscore.ToString();
```

```
98  Time.timeScale = 1;
99  isPaused = false;
100
101 foreach (Image i in lifepoints)
102 i.enabled = true;
103
104 foreach (Vegetable v in veggies)
105 if (v != null)
106 Destroy(v.gameObject);
107 veggies.Clear();
108
109 foreach (Bomb b in bombs)
110 Destroy(b.gameObject);
111 bombs.Clear();
112
113 deathMenu.SetActive(false);
114 }
115
116 public void Retry()
117 {
118 MusicToggleButton.SetActive(!MusicToggleButton.activeSelf);
119 SoundToggleButton.SetActive(!SoundToggleButton.activeSelf);
120 NewGame();
121 }
122
123 private void Update()
124 {
125 if (isPaused)
126 return;
127
128 if(Time.time - lastSpawn > deltaSpawn)
129 {
130 Vegetable v = GetVegetable();
131 float randomX = Random.Range(-1.65f,1.65f);
132 v.LaunchVegatable(Random.Range(1.85f,2.75f),randomX,-randomX);
133 lastSpawn = Time.time;
134 }
135
136 if(Time.time - lastBombSpawn > deltaSpawn*2+2.321f)
137 {
138 Bomb b = GetBomb();
139 float randomX = Random.Range(-1.65f,1.65f);
140 b.LaunchBomb(Random.Range(1.85f,2.75f),randomX,-randomX);
141 lastBombSpawn = Time.time;
142 }
143
144 if(Input.GetMouseButton(0))
145 {
146 Vector3 pos =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
147 pos.z = -1;
148
```

```
149 if(Time.time - LastTrailTime > DeltaTrailTime)
150 {
151 tr.transform.position = new Vector3 (pos.x,pos.y,pos.z);
152 tr.enabled=true;
153 }
154
155 Collider2D[] thisFramesVeggie = Physics2D.OverlapPointAll(new
Vector2(pos.x,pos.y), LayerMask.GetMask("Vegetable"));
156 Collider2D[] thisFramesBomb = Physics2D.OverlapPointAll(new
Vector2(pos.x,pos.y), LayerMask.GetMask("Bomb"));
157
158 if ((Input.mousePosition - lastMousePos).sqrMagnitude >
REQUIRED_SLICEFORCE)
159 {
160 foreach (Collider2D c2 in thisFramesVeggie)
161 {
162 for (int i = 0; i < veggiesCols.Length; i++)
163 {
164 if (c2 == veggiesCols[i])
165 {
166 c2.GetComponent<Vegetable>().Slice();
167 }
168 }
169 }
170
171 foreach (Collider2D b2 in thisFramesBomb)
172 {
173 for (int i = 0; i < bombsCols.Length; i++)
174 {
175 if (b2 == bombsCols[i])
176 {
177 b2.GetComponent<Bomb>().Slice();
178 }
179 }
180 }
181 }
182
183 lastMousePos = Input.mousePosition;
184 veggiesCols = thisFramesVeggie;
185 bombsCols = thisFramesBomb;
186 }
187
188 if(Input.GetMouseButtonUp(0))
189 {
190 tr.enabled=false;
191 LastTrailTime = Time.time;
192 }
193 }
194
195 private Vegetable GetVegetable()
196 {
197 Vegetable v = veggies.Find(x => !x.IsActive);
```

```
198 if(v == null)
199 {
200 v = Instantiate( vegetablePrefab[Random.Range(0,8)])
.GetComponent<Vegetable>();
201 veggies.Add(v);
202 }
203 return v;
204 }
205
206 private Bomb GetBomb()
207 {
208 Bomb b = bombs.Find(x => !x.IsActive);
209 if(b == null)
210 {
211 b = Instantiate(bombPrefab[0]).GetComponent<Bomb>();
212 bombs.Add(b);
213 }
214 return b;
215 }
216
217 public void IncrementScore(int scoreAmount) {
218 score += scoreAmount;
219 scoreText.text = score.ToString();
220 if (score > highscore)
221 {
222 highscore = score;
223 highscoreText.text = "BEST: " + highscore.ToString();
224 PlayerPrefs.SetInt("Score",highscore);
225 }
226 }
227
228 public void IncrementDifficulty(float IncrNum)
229 {
230 if (deltaSpawn>0.5f)
231 deltaSpawn -= IncrNum;
232 }
233
234 public void LoseLP()
235 {
236 if (lifepoint == 0)
237 return;
238
239 StartCoroutine(RedFlashCoroutine());
240 deltaSpawn = 1.0f;
241 lifepoint--;
242 lifepoints[lifepoint].enabled = false;
243
244 if (lifepoint == 0)
245 {
246 foreach (Vegetable v in veggies)
247 if (v != null)
248 v.NoPointsSlice();
```

```
249 Death();
250 return;
251 }
252
253 foreach (Vegetable v in veggies)
254 if (v != null)
255 v.NoPointsSlice();
256 }
257
258 public void Death()
259 {
260 isPaused = true;
261 StartCoroutine(DeathMenuCoroutine());
262 }
263
264 public void PauseGame() {
265 SoundManager.Instance.PlaySound(0);
266 pauseMenu.SetActive(!pauseMenu.activeSelf);
267 MusicToggleButton.SetActive(!MusicToggleButton.activeSelf);
268 SoundToggleButton.SetActive(!SoundToggleButton.activeSelf);
269 if (PlayerPrefs.GetInt("CrossedC0") == 1)
CrossedCircles[0].enabled = true;
270 if (PlayerPrefs.GetInt("CrossedC1") == 1)
CrossedCircles[1].enabled = true;
271 isPaused = pauseMenu.activeSelf;
272 Time.timeScale = (Time.timeScale == 0) ? 1 : 0 ;
273 SoundManager.Instance.PlaySound(0); }
274
275 public void ToMenu()
276 {
277 StartCoroutine(ToMenuCoroutine());
278 }
279
280 public void ClickSound()
281 {
282 SoundManager.Instance.PlaySound(0);
283 }
284
285 public void ToggleBgMusicButton()
286 {
287 SoundManager.Instance.ToggleBgMusic();
288 if (CrossedCircles[0].enabled == false)
289 {
290 CrossedCircles[0].enabled = true;
291 PlayerPrefs.SetInt("CrossedC0", 1);
292 }
293 else
294 {
295 CrossedCircles[0].enabled = false;
296 PlayerPrefs.SetInt("CrossedC0", 0);
297 }
298 }
```

```
299
300 public void ToggleGameSoundsButton()
301 {
302 SoundManager.Instance.ToggleGameSounds();
303 if (CrossedCircles[1].enabled == false)
304 {
305 CrossedCircles[1].enabled = true;
306 PlayerPrefs.SetInt("CrossedC1", 1);
307 }
308 else
309 {
310 CrossedCircles[1].enabled = false;
311 PlayerPrefs.SetInt("CrossedC1", 0);
312 }
313 }
314 }
```

Vegetable.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class Vegetable : MonoBehaviour
7  {
8      private const float GRAVITY = 2.0f;
9
10     public bool IsActive { set; get; }
11     public SpriteRenderer sRenderer;
12     public GameManager GameManager;
13
14     private float verticalVelocity;
15     private float speed;
16     private bool isSliced = false;
17
18     public Sprite[] sprites;
19     private int spriteIndex;
20     private float lastSpriteUpdate;
21     private float spriteUpdateDelta = 0.1f; // швидкість анімації
   при розрізі
22     private float rotationSpeed;
23
24     public void LaunchVegatable(float verticalVelocity, float
   xSpeed, float xStart)
25     {
26         IsActive = true;
27         speed = xSpeed;
28         this.verticalVelocity = verticalVelocity;
29         transform.position = new Vector3(xStart, 0, 0);
30         rotationSpeed = Random.Range(-360, 360);
31         isSliced = false;
```

```
32          spriteIndex = 0;
33          sRenderer.sprite = sprites[spriteIndex];
34      }
35
36      private void Update()
37      {
38          if (!IsActive)
39              return;
40
41          verticalVelocity -= GRAVITY * Time.deltaTime;
42          transform.position += new Vector3(speed, verticalVelocity,
    0) * Time.deltaTime;
43          transform.Rotate(new Vector3(0, 0, rotationSpeed) *
    Time.deltaTime);
44
45          if (isSliced)
46          {
47              if (spriteIndex != sprites.Length-1 && Time.time -
    lastSpriteUpdate > spriteUpdateDelta)
48              {
49                  lastSpriteUpdate = Time.time;
50                  spriteIndex++;
51                  sRenderer.sprite = sprites[spriteIndex];
52              }
53          }
54
55          if (transform.position.y < -1)
56          {
57              IsActive = false;
58              if (!isSliced)
59                  GameManager.Instance.LoseLP();
60              Destroy(this.gameObject);
61          }
62      }
63
64      public void Slice()
65      {
66          if(isSliced) return;
67          if (verticalVelocity < 0.5f) verticalVelocity = 0.5f;
68          speed = speed * 0.5f;
69          isSliced = true;
70          SoundManager.Instance.PlaySound(1);
71          GameManager.Instance.IncrementScore(1);
72          GameManager.Instance.IncrementDifficulty(0.01f);
73      }
74
75      public void NoPointsSlice()
76      {
77          if(isSliced)
78              return;
79          if (verticalVelocity < 0.5f)
80              verticalVelocity = 0.5f;
```

```
81          speed = speed * 0.5f;
82          isSliced = true;
83          SoundManager.Instance.PlaySound(1);
84      }
85  }
```

Bomb.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Bomb : MonoBehaviour
6  {
7      private const float GRAVITY = 2.0f;
8
9      public bool IsActive { set; get; }
10     public SpriteRenderer sRenderer;
11     public GameManager GameManager;
12
13     private float verticalVelocity;
14     private float speed;
15     private bool isSliced = false;
16
17     public Sprite[] sprites;
18     private int spriteIndex;
19     private float lastSpriteUpdate;
20     private float spriteUpdateDelta = 0.1f; // швидкість анімації
   при розрізі
21     private float rotationSpeed;
22
23     public void LaunchBomb(float verticalVelocity, float xSpeed,
   float xStart)
24     {
25         IsActive = true;
26         speed = xSpeed;
27         this.verticalVelocity = verticalVelocity;
28         transform.position = new Vector3(xStart, 0, 0);
29         rotationSpeed = Random.Range(-360, 360);
30         isSliced = false;
31         spriteIndex = 0;
32         sRenderer.sprite = sprites[spriteIndex];
33     }
34
35     private void Update()
36     {
37         if (!IsActive)
38             return;
39
40         verticalVelocity -= GRAVITY * Time.deltaTime;
41         transform.position += new Vector3(speed, verticalVelocity,
   0) * Time.deltaTime;
```

```
42          transform.Rotate(new Vector3(0, 0, rotationSpeed) *
  Time.deltaTime);
43
44          if (isSliced)
45          {
46              if (spriteIndex != sprites.Length-1 && Time.time -
  lastSpriteUpdate > spriteUpdateDelta)
47              {
48                  lastSpriteUpdate = Time.time;
49                  spriteIndex++;
50                  sRenderer.sprite = sprites[spriteIndex];
51              }
52          }
53
54          if (transform.position.y < -1)
55          {
56              IsActive = false;
57          }
58      }
59
60      public void Slice()
61      {
62          if(isSliced) return;
63          if (verticalVelocity < 0.5f)
64              verticalVelocity = 0.5f;
65          speed = speed * 0.5f;
66          isSliced = true;
67          SoundManager.Instance.PlaySound(2);
68          GameManager.Instance.LoseLP();
69      }
70  }
```

MenuManager.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  public class MenuManager : MonoBehaviour
8  {
9      public Image[] CrossedCircles;
10     public GameObject ExitConfirmMenu;
11
12     private void Start()
13     {
14         Time.timeScale = 1;
15         if (PlayerPrefs.GetInt("CrossedC0") == 1)
16             CrossedCircles[0].enabled = true;
17         if (PlayerPrefs.GetInt("CrossedC1") == 1)
18             CrossedCircles[1].enabled = true;
```

```
19    }
20    public void ToGame()
21    {
22       StartCoroutine(ToGameCoroutine());
23    }
24
25    IEnumerator ToGameCoroutine()
26    {
27       SoundManager.Instance.PlaySound(0);
28       yield return new WaitForSeconds(0.261f);
29       SceneManager.LoadScene("Game");
30    }
31
32    public void ToggleBgMusicButton() {
33       SoundManager.Instance.ToggleBgMusic();
34       if (CrossedCircles[0].enabled == false) {
35          CrossedCircles[0].enabled = true;
36          PlayerPrefs.SetInt("CrossedC0", 1);
37       }
38       else {
39          CrossedCircles[0].enabled = false;
40          PlayerPrefs.SetInt("CrossedC0", 0);
41       } }
42
43    public void ToggleGameSoundsButton() {
44       SoundManager.Instance.ToggleGameSounds();
45       if (CrossedCircles[1].enabled == false) {
46          CrossedCircles[1].enabled = true;
47          PlayerPrefs.SetInt("CrossedC1", 1);
48       }
49       else {
50          CrossedCircles[1].enabled = false;
51          PlayerPrefs.SetInt("CrossedC1", 0);
52       } }
53
54    public void XExitButton()
55    {
56       ExitConfirmMenu.SetActive(true);
57    }
58
59    public void NoExitButton()
60    {
61       ExitConfirmMenu.SetActive(false);
62    }
63
64    public void YesExitButton()
65    {
66       SoundManager.Instance.QuitApp();
67    }
68
69    public void ClickSound()
70     {
```

```
71          SoundManager.Instance.PlaySound(0);
72      }
73  }
```

SoundManager.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class SoundManager : MonoBehaviour
7  {
8      public static SoundManager Instance{set;get;}
9
10     public AudioSource source;
11     public AudioClip[] allSounds;
12     private bool SoundMuted;
13
14     private void Start()
15     {
16         Instance = this;
17         DontDestroyOnLoad(gameObject);
18         if (PlayerPrefs.GetInt("CrossedC0") == 1)
19             source.mute = true;
20         if (PlayerPrefs.GetInt("CrossedC1") == 1)
21             SoundMuted = true;
22         SceneManager.LoadScene("Menu");
23     }
24
25     public void PlaySound(int soundIndex)
26     {
27         if (SoundMuted == false)
   AudioSource.PlayClipAtPoint(allSounds[soundIndex],transform.positi
   on);
28         else return;
29     }
30     public void ToggleBgMusic()
31     {
32         source.mute = !source.mute;
33     }
34     public void ToggleGameSounds()
35     {
36         if (SoundMuted == false)
37             SoundMuted = true;
38         else SoundMuted = false;
39     }
40     public void QuitApp()
41     {
42         Application.Quit();
43     }
44 }
```

Додаток Б

Комп’ютерна презентація

Рисунок Б.1 – Титульний слайд

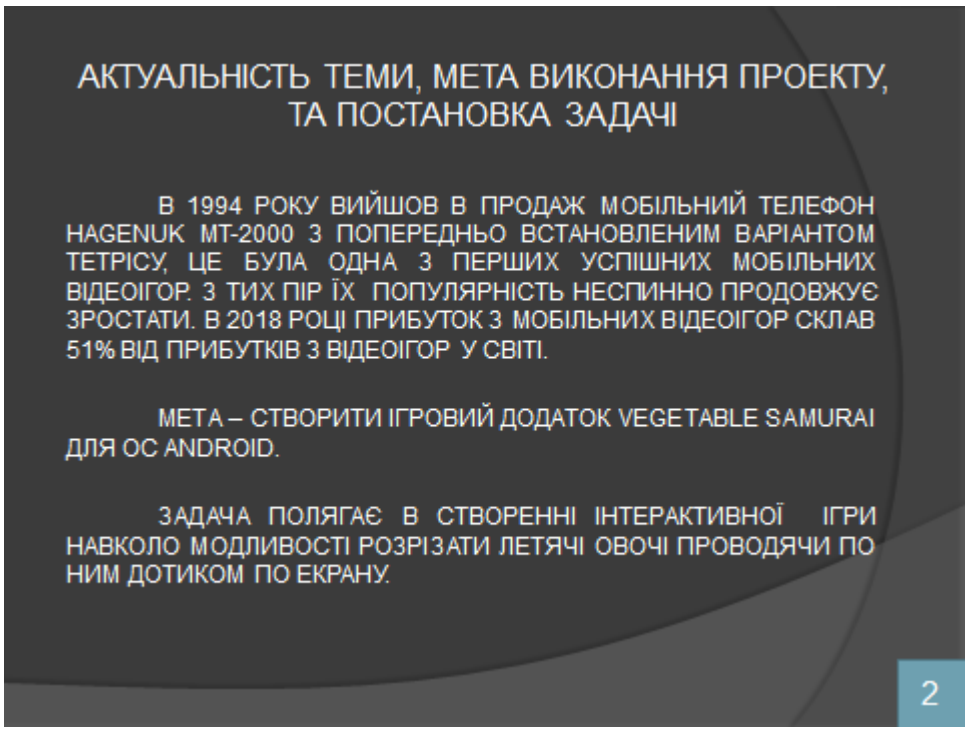

Рисунок Б.2 – Актуальність теми, мета виконання проекту, та постановка задачі

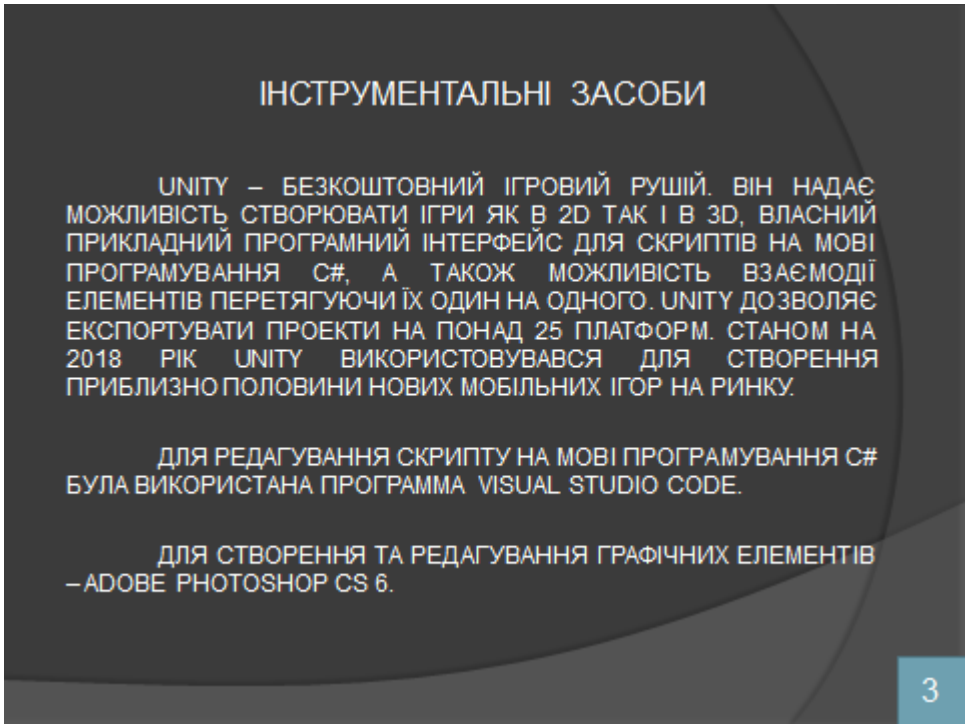

Рисунок Б.3 – Інструментальні засоби

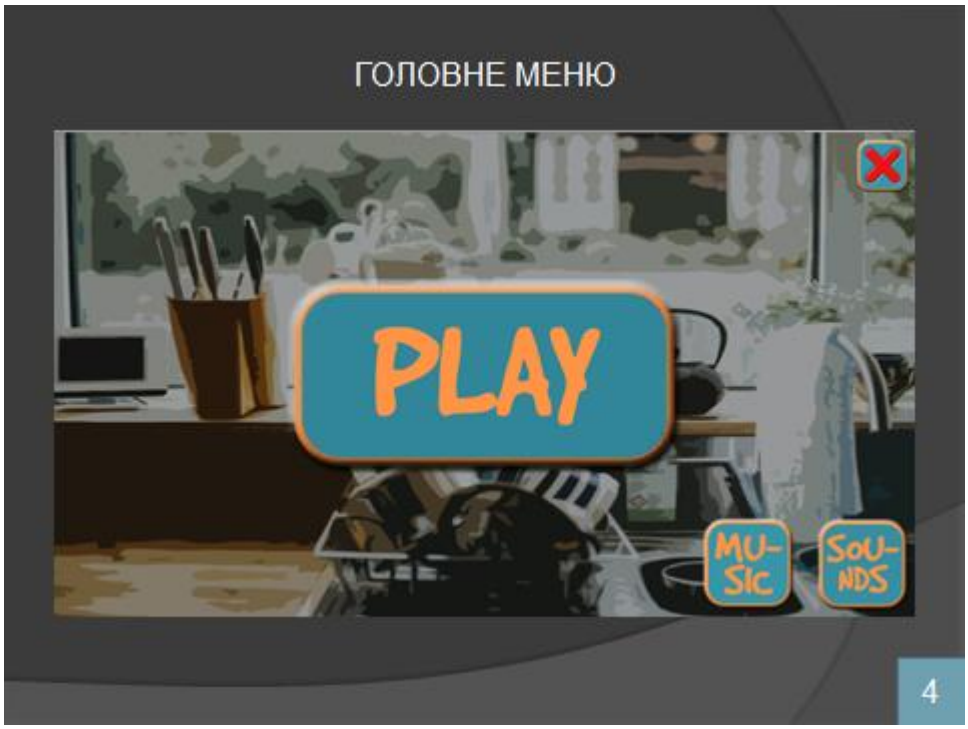

Рисунок Б.4 – Головне меню

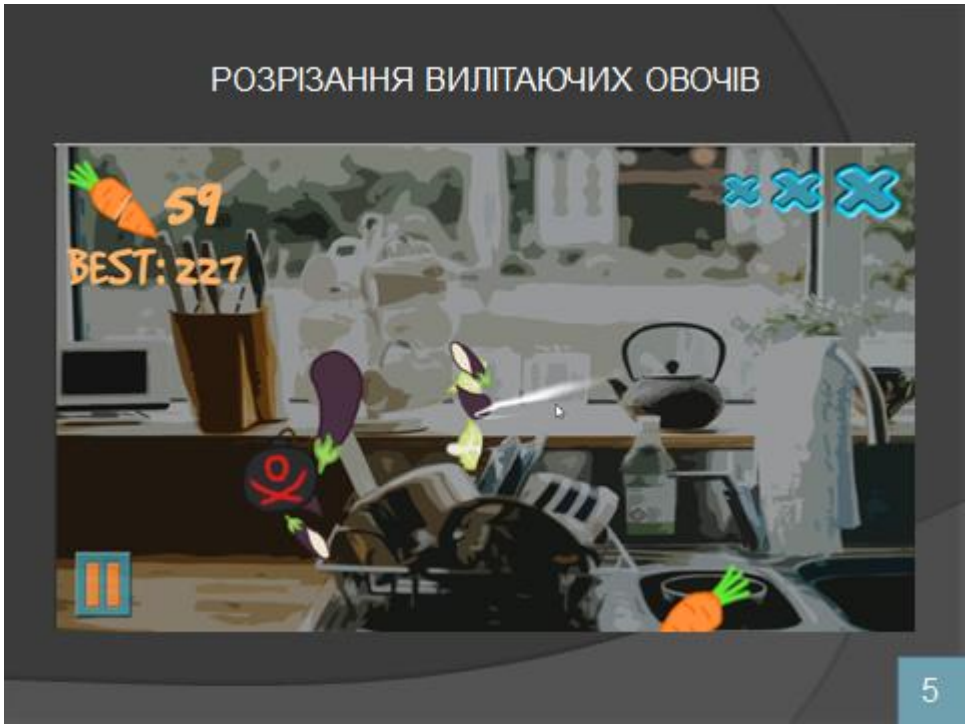

Рисунок Б.5 – Розрізання вилітаючих овочів

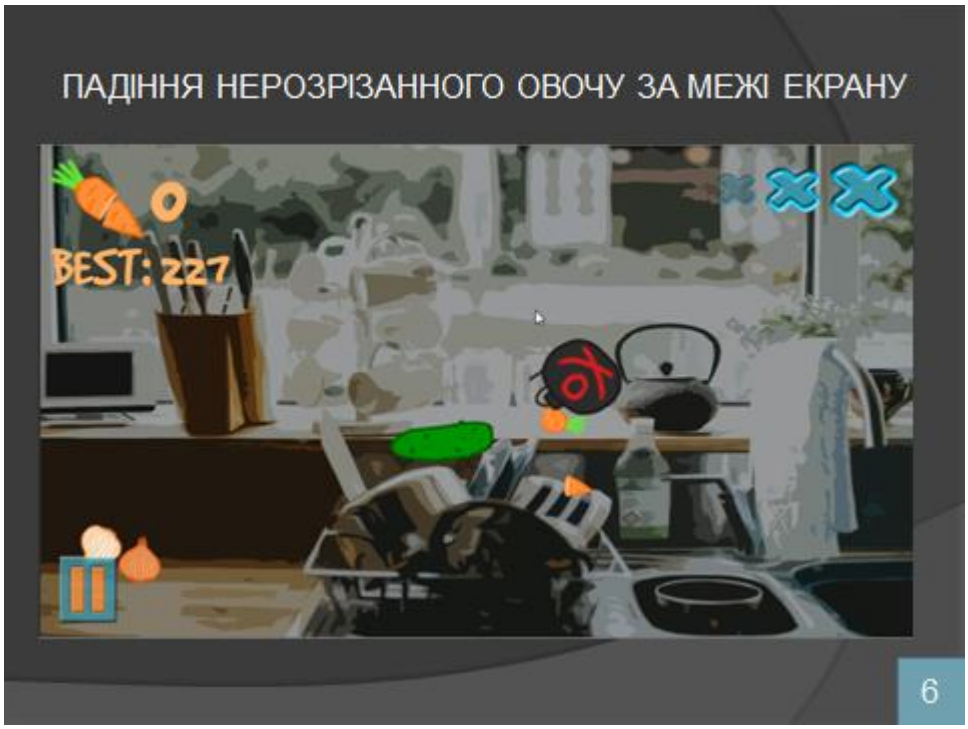

Рисунок Б.6 – Падіння нерозрізаного овочу за межі екрану

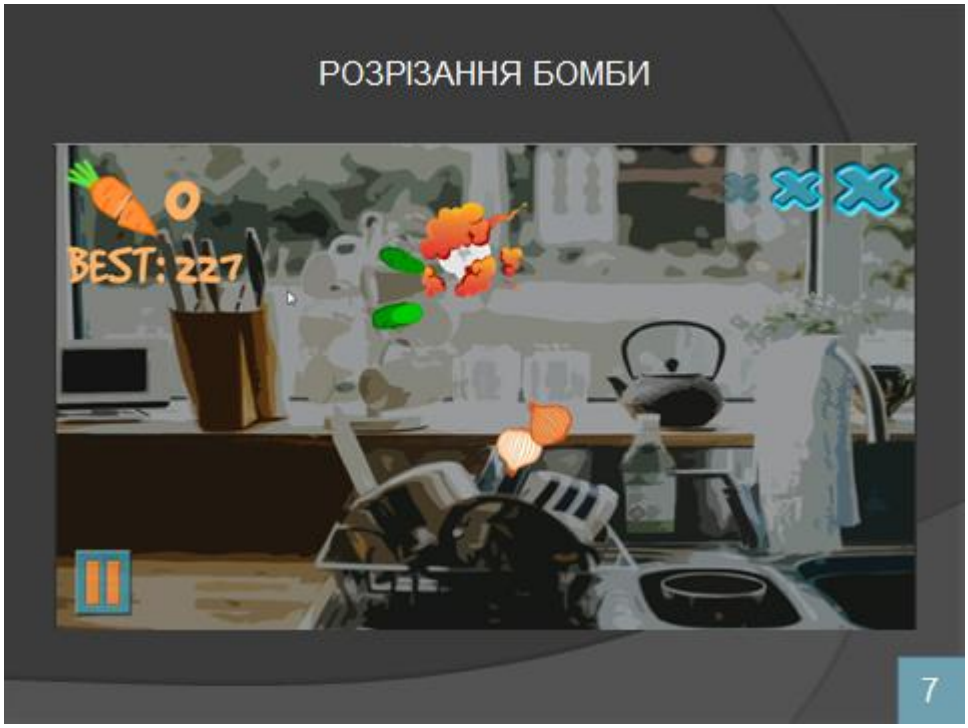

Рисунок Б.7 – Розрізання бомби

Рисунок Б.8 – Меню паузи

Рисунок Б.9 – Меню програшу

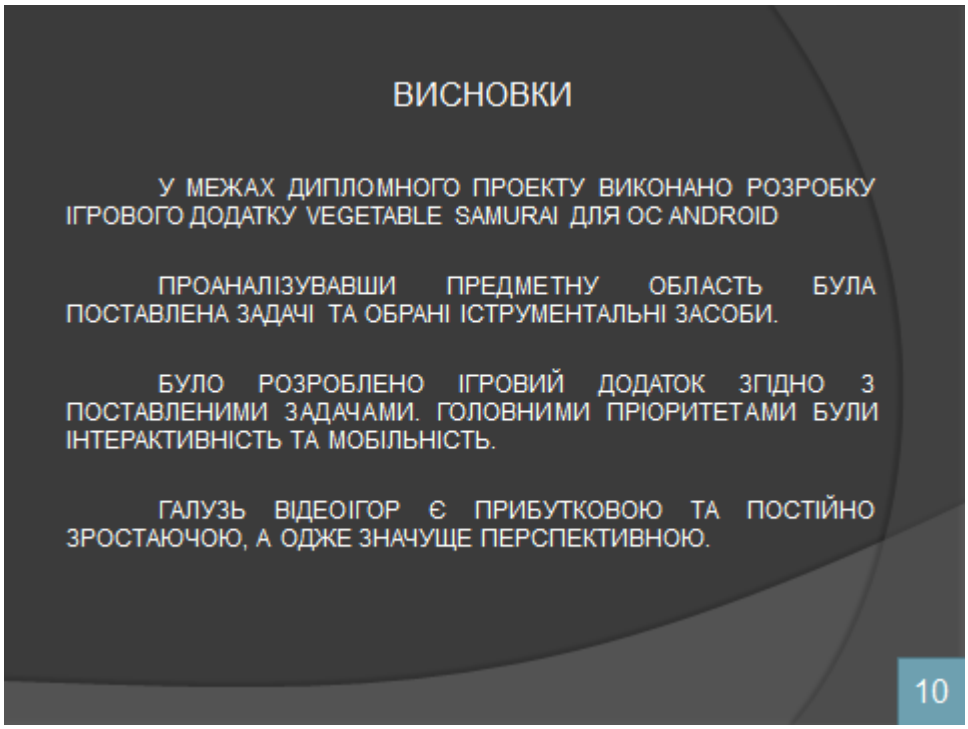

Рисунок Б.10 – Висновки