МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

_____

к.т.н, доц. Михелєв І.Л.

"____" _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: Розробка мобільної гри «Last Chance»

Виконав: студент групи

_____*Горелко Михайло Сергійовчи*
(підпис)

Керівник роботи:

____к.т.н., доцент_____
(посада, науковий ступінь вчене звання)

_____ *Михелєв І.Л.*
(підпис)

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"
Гарант освітньої програми
_____
к.т.н, доц. Гайда А.Ю.
"____" _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Горелку Михайлу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка мобільної гри«LastChanse»

Керівник роботи Михелєв Ігор Леонідович

Затверджені наказом ректора № 243-уч від "18" березня 2024 року.

2. Термін подання роботи: 20 червня 2024 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів)
_____
_____
_____

5. Перелік презентаційних матеріалів_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Михелєв І.Л.		
2	Михелєв І.Л.		
3	Михелєв І.Л.		
4	Гурець Н.В.		

7. Дата видачі завдання 05 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Студент _____ Горелко М.С
(підпис) (ПІБ)

Керівник роботи _____ Михелєв І.Л.
(підпис) (ПІБ)

АНОТАЦІЯ

У кваліфікаційній роботі на тему "Розробка мобільної гри «Last Chance»" досліджено процес створення головоломки, де гравець повинен обертати лабіринт для витягнення всіх кубиків, окрім сірого. Основною метою роботи було здобуття навичок проектування та реалізації комп'ютерної гри з використанням об'єктно-орієнтованого програмування в середовищі Unity. Процес розробки включав аналіз ігрового жанру та аналогів, об’єктно-орієнтований аналіз та проектування, написання програмного коду, впровадження бази даних, тестування та оптимізацію гри. Результатом роботи стала створена мобільна гра, яка поєднує в собі логіку, стратегію та інтуїцію, сприяючи розвитку інтелектуальних здібностей гравців.

ANNOTATIONS

The qualification work titled "Development of the Mobile Game 'Last Chance'" explores the process of creating a puzzle game where the player must rotate a labyrinth to extract all cubes except the gray one. The main goal of the work was to gain skills in designing and implementing a computer game using object-oriented programming in the Unity environment. The development process included analyzing the game genre and analogs, conducting object-oriented analysis and design, writing the program code, implementing a database, testing, and optimizing the game. The result of the work is a mobile game that combines logic, strategy, and intuition, contributing to the development of players' intellectual abilities.

ЗМІСТ

ВСТУП

З кожним роком кількість, популярність та якість комп'ютерних ігор зростає. Попит на ігри також зростає, завдяки чому ігрова індустрія виходить на новий рівень.

Метою дослідження є здобуття навичок проектування та реалізації комп’ютерної гри.

Об'єктом дослідження є методи та засоби середовища об'єктно-орієнтованого програмування Visual Studio в Unity, яке дозволяє створювати ігри за парадигмою об'єктно-орієнтованого підходу.

Предметом курсового проектування є створення ігрового програмного продукту – головоломки «Last Chanse» засобами мови C#.

Завданням дипломного проєкту є створення гри «Last Chanse» за допомогою ігрового движка Unity.

Сенс моєї гри полягає в тому, що гравець повинен обертаючи лабіринт витягти з нього усі кубики окрім сірого.

У розділі «Дослідження та аналіз вимог до створення гри «Last Chanse»» встановлюються вимоги до створюваної гри та до засобів її створення.

У розділі «Розробка комп’ютерної гри «Last Chanse»» визначається алгоритм роботи гри та функції, які вона буде виконувати.

У розділі «Тестування» вказується як користуватися грою та наводяться дії, що вона може виконувати.

У розділі «Охорона праці» здійснено аналіз умов, гігієни праці та пожежної безпеки виробничого приміщення.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ВИМОГ ДО СТВОРЕННЯ ГРИ «LAST CHANSE»

1.1 Дослідження гральних рушіїв

На даний момент є багато ігрових движків різного призначення, але серед них найбільш популярними є: Unity; GameMaker Studio; Construct 3; Godot Engine; Cocos2D, Phaser.

Unity є одним з найбільш популярних рушіїв гри для розробки 2D-ігор. Його функціональні можливості охоплюють весь спектр розробки ігор, від проектування та моделювання 3D-об'єктів до програмування складних систем фізики та штучного інтелекту. Unity пропонує широкий спектр інструментів та функцій для розробки ігор, включаючи фізику, анімацію та програмування. Він також має багато додаткових плагінів, які дозволяють розширювати його можливості. Unity підтримує розробку ігор з використанням різних мов програмування, включаючи C ++, C # та JavaScript. Unity також дозволяє розробникам імпортувати різні формати графіки та звуків, що дозволяє їм використовувати зовнішні ресурси для своїх ігор.

GameMaker Studio є потужним рушієм гри, спеціалізованим на створенні 2D-ігор. Його основною перевагою є візуальний інтерфейс, який дозволяє створювати графічні об'єкти, задавати їх поведінку та розміщувати їх у сцені без необхідності написання складного коду. GameMaker Studio має вбудовану систему фізики та дозволяє легко створювати різноманітні ефекти, анімацію та штучний інтелект. Він підтримує багато платформ, включаючи ПК, мобільні пристрої, консолі та веб. GameMaker Studio також має активну спільноту розробників та широкий вибір ресурсів для навчання та підтримки.

Construct 3 - це графічний рушій для створення 2D ігор, який використовує веб-технології. Він має простий та зрозумілий інтерфейс, що робить його доступним для початківців та досвідчених розробників. Рушій дозволяє створювати ігри

без програмування за допомогою системи поведінки та вбудованої бібліотеки готових компонентів. Основними перевагами Construct 3 є легкість використання та швидкість розробки. Він має багато готових компонентів, таких як рух, фізика, анімація, звукові ефекти та інші, що значно спрощує процес створення гри. Крім того, Construct 3 має вбудовану систему поведінки, яка дозволяє легко керувати об'єктами в грі. Construct 3 підтримує експорт готових ігор на різні платформи, такі як iOS, Android, Windows, Mac OS X та Linux. Рушій також підтримує інтеграцію з різними інструментами, такими як Photoshop, GIMP та TexturePacker.Godot Engine: Godot Engine є відкритим джерелом рушія гри, який дозволяє розробникам створювати 2D-ігри безкоштовно. Рушій має вбудовану систему фізики та підтримує складні системи руху та анімації. Godot Engine також підтримує розробку ігор з використанням різних мов програмування, включаючи C ++, C #, Python та GDScript. Крім того, рушій має вбудовану систему редактора, що дозволяє розробникам створювати графіку та звук, редагувати сценарії та налаштовувати системи фізики.

Cocos2D є безкоштовним, відкритим рушієм для створення 2D ігор для мобільних пристроїв та ПК. Він підтримує багато мов програмування, таких як C++, JavaScript, Lua та Swift, що робить його досить гнучким рушієм для розробки ігор. Основними перевагами Cocos2D є легкість використання, швидкість розробки та гнучкість. Він має багато готових компонентів, таких як рух, кільцеві елементи, анімація та ефекти, що значно спрощує створення гри. Крім того, Cocos2D має вбудовану підтримку фізики, що дозволяє створювати реалістичні фізичні ефекти. Cocos2D також має багато документації та активну спільноту розробників, що робить його легким у використанні для початківців та досвідчених розробників. Крім того, рушій підтримує розробку ігор для різних платформ, таких як iOS, Android, Windows, Mac OS X та Linux.

Phaser - це безкоштовний ігровий рушій, розроблений для створення браузерних ігор з використанням JavaScript. Phaser пропонує зручний інтерфейс для розробки ігор з графікою в стилі 2D, а також надійне підтримує сучасні браузери.

Основні можливості Phaser:

– підтримка різноманітних фізичних ефектів, таких як динаміка колізій, вплив гравітації і масштабування;

– можливість роботи з аудіо та анімацією, зокрема з використанням спрайтів (графічних об'єктів, які містять декілька кадрів з анімацією);

– підтримка клавіатури та миші, а також тачскрінів для розробки мобільних ігор;

– широкі можливості для роботи з текстом та шрифтами;

– підтримка WebGL для покращення графічної якості ігор.

Phaser також пропонує велику кількість документації та прикладів використання, що дозволяє швидко розібратися з функціональністю рушія та створити гру. Рушій підтримується відкритою спільнотою, яка активно розвивається та додає нові функції і підтримку для різних платформ.

Для створення гри я обрав Unity, адже він більш простий та вимагає менше часу на розробку гри. Також Unity має велику бібліотеку безкоштовних асетів.

1.2 Аналіз ігрового жанру та існуючих аналогів

Я створюю гру для дипломного проєкту жанру головоломка. Жанр комп'ютерних ігор, що називається головоломкою полягає у вирішенні різних логічних завдань, які вимагають від гравців використання логіки, стратегії та інтуїції. Жанр головоломки є одним з найпопулярніших ігрових жанрів та має величезну аудиторію по всьому світу.

Основною перевагою жанру головоломки є його різноманітність та можливість використання різних механік та рішень завдань. Головоломки можуть бути простими, такими як складання пазлів або збір певних об’єктів, або ж складними, такими як розв’язування математичних задач, головоломок з лабіринтами або кри-

птографічних головоломок. Жанр головоломки часто використовується у навчальних іграх та прикладних програмах, де гравець може вивчати нові концепції та знання, вирішуючи різні завдання.

Ще однією перевагою жанру головоломки є його доступність для гравців різного віку та рівня навичок. Більшість головоломок можна вирішити без необхідності глибокого знання гри, що робить їх доступними для всіх. Також жанр головоломки є відмінним засобом для тренування та розвитку мозку, підвищення креативності та логічного мислення.

Одним з недоліків жанру головоломки є те, що він може бути надто складним для деяких гравців, що може зіпсувати їх враження від гри. Іншим недоліком жанру головоломки є те, що він може бути менш привабливим для гравців, які шукають більш динамічну та екшн-орієнтовану гру. Оскільки жанр головоломки зазвичай вимагає від гравця більш спокійного та розміреного підходу, він може бути менш привабливим для тих, хто шукає більш емоційне та захоплююче геймплей.

Проте, незважаючи на ці недоліки, жанр головоломки все ще залишається дуже популярним серед гравців. Багато гравців вважають, що вирішення складних головоломок є дуже задовільним та стимулюючим досвідом. Більшість головоломок також мають дуже високу ігрову реіграбельність, що означає, що гравці можуть повертатися до гри знову та знову, щоб вирішувати різні завдання та досягати кращих результатів.

Крім того, жанр головоломки є досить унікальним, оскільки він може бути успішно поєднаний з іншими жанрами гри, створюючи нові та цікаві гібриди. Наприклад, головоломки можуть бути використані в якості елементів гри в пригодницьких, рольових або стратегічних іграх.

Аналогами до створюваної гри є: Tetris; Candy Crush; Braid.

Tetris (рис. 1) - це головоломка, створена в 1984 році російським програмістом Алексеєм Пажитновим. Гра отримала свою назву від слова "тетроміно", яке описує форму фігур, що складаються з чотирьох квадратів.

У грі Tetris гравець повинен управляти фігурами з різними формами, які падають зверху до низу екрану, і розміщувати їх так, щоб утворювалися повні рядки без прогалин. Кожна фігура складається з чотирьох квадратів, які можна повертати та переміщувати горизонтально зліва направо.

Гра має різні рівні складності, і з кожним новим рівнем швидкість падіння фігур збільшується, що робить гру все складнішою. Гра закінчується, коли фігури досягають верхньої частини екрану, і більше не можуть бути розміщені.

Одна з головних принад гри Tetris полягає в її простоті. Вона дуже легко зрозуміла, і навіть новачок може швидко зрозуміти, як грати. Одночасно, гра є дуже захопливою, і досить важкою на вищих рівнях складності, що робить її досить варіативною та реіграбельною.

Tetris став однією з найуспішніших та найпопулярніших головоломок у світі, і був випущений на більш ніж 65 різних платформах. Гра входить до культурної спадщини та стала символом епохи комп'ютерних ігор.

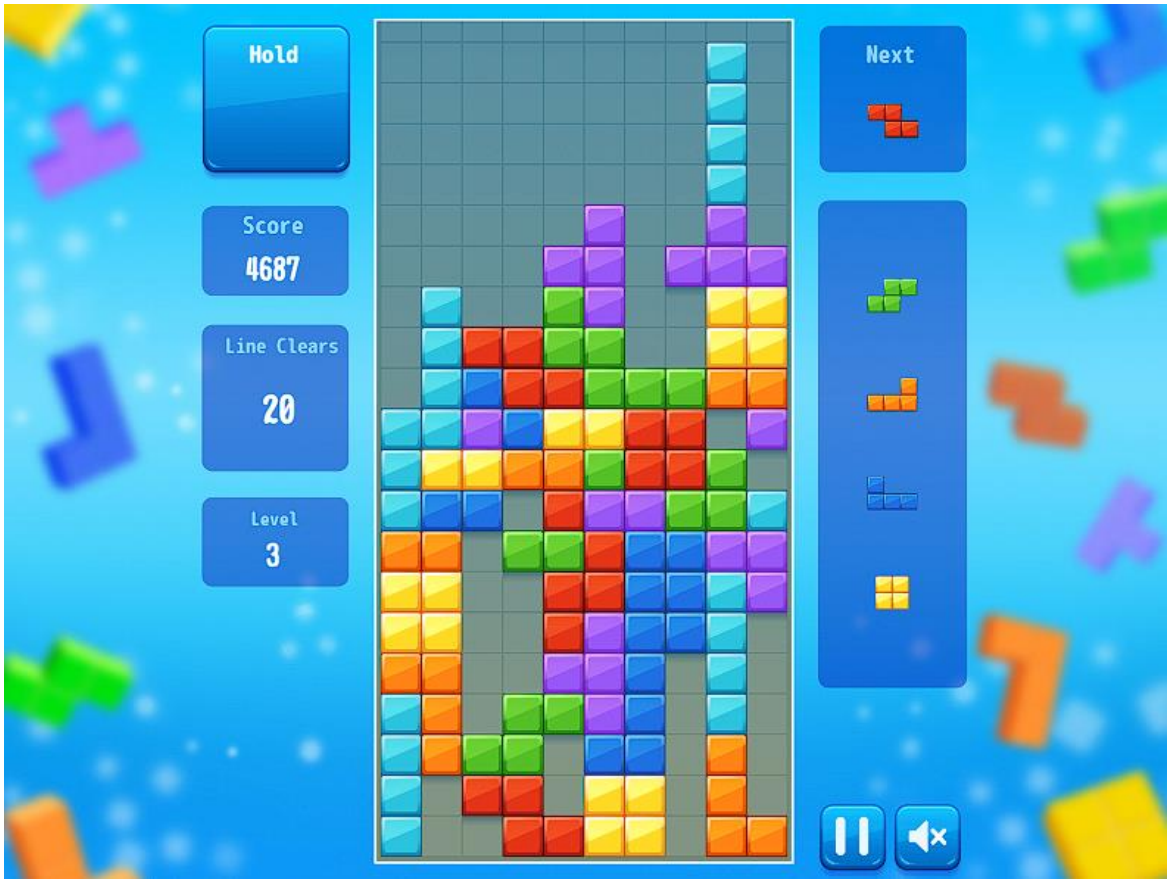

Рисунок 1 Сучасна інтерпретація гри Tetris

Candy Crush (рис. 2) - це одна з найпопулярніших головоломок у світі, розроблена компанією King та випущена в 2012 році. Гра проста в освоєнні та має привабливу графіку та звукове супроводження.

У грі гравець повинен збирати комбінації з трьох або більше однакових цукерок, які розташовані на ігровому полі. Комбінування цукерок дозволяє отримувати бонуси, які допомагають ускладнювати гру для опонента або допомагають самому гравцеві.

Гра має різноманітність рівнів та завдань, що дозволяє гравцю отримувати нові виклики та не набридати. У грі також є можливість грати з друзями на соціальних мережах або ділитися своїми результатами.

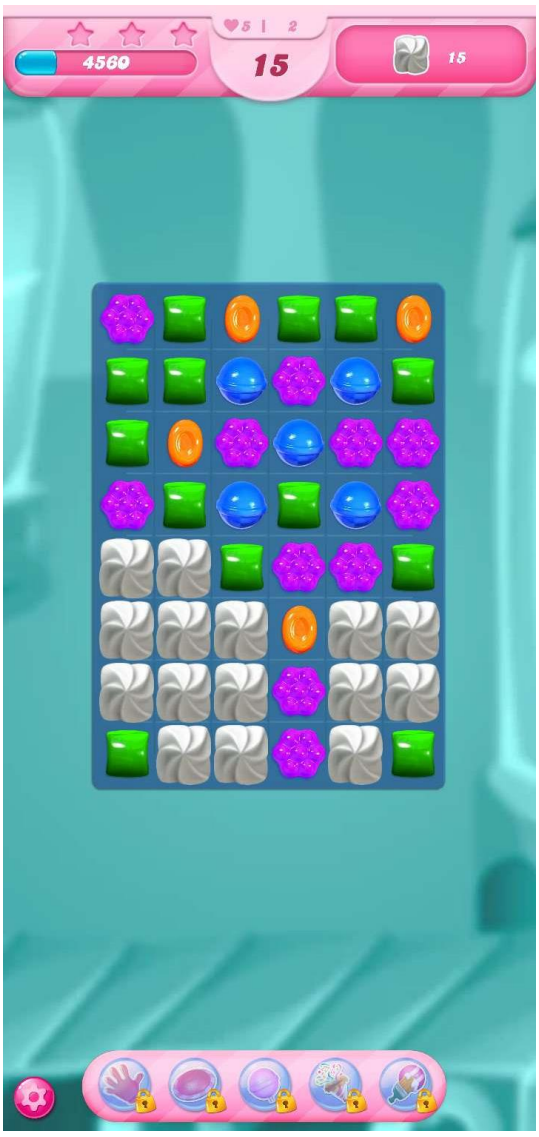

Рисунок 2 Candy Crush

Одна з головних принад гри Candy Crush полягає в її простоті та доступності для всіх вікових груп. Це робить її дуже популярною серед багатьох людей, а особливо серед тих, хто не є досвідченими гравцями комп'ютерних ігор.

Гра вважається високодохідною для розробника King, що сприяло появі безлічі схожих ігор в цьому жанрі. Водночас, гра Candy Crush залишається однією з найпопулярніших та найвідоміших головоломок у світі.

Braid (рис. 3) - це інноваційна головоломкова гра, створена в 2008 році дизайнером і програмістом Джонатаном Блоу. Гра має незвичний геймплей, у якому гравець має використовувати часові петлі для вирішення головоломок і проходження рівнів.

Головний герой гри - Тім, який мандрує світом, щоб знайти свою кохану принцесу. Щоб допомогти йому, гравець повинен пересуватися по етапам, використовуючи різні часові петлі. Наприклад, можна перемотати час назад, щоб змінити результат події або використати затримку часу, щоб зупинити рух об'єктів на екрані.

Рисунок 3 Braid

Гра має велику кількість головоломок, які стають все складнішими і вимагають більшої кмітливості з боку гравця. Гра також має чудову графіку в стилі ручної рисовки та мелодійну музику, що доповнює загадкову та містичну атмосферу гри.

Braid отримала високі оцінки від критиків і стала однією з найуспішніших ігор в жанрі головоломок. Гра не тільки пропонує цікаві головоломки, але й стимулює гравців мислити інноваційно та змушує дивитися на вирішення проблем з іншого кута зору.

1.3 Визначення вимог

Створюється гра жанру головоломка. Сенс гри полягає у тому, що гравець повинен витягти 3 кольорових кубика з лабіринту не випустивши сірий кубик з нього. На усі кубики діє гравітація, щоб їх витягнути необхідно повернути лабіринт у певний бік. Кубики обертаються разом з лабіринтом. Окрім основного виходу в рівнях можуть бути додаткові виходи, через які може пройти кубик відповідного кольору. Гравець має можливість кастомізувати кубики збираючи монети під час проходження рівнів та обмінюючи їх на скіни. Для приємної взаємодії з функціями гри меню повинно бути приємно оформлене.

Тож для створення гри було поставлено наступні вимоги:

- створити меню;
- створити ігровий світ;
- створити 4 кубики та 3 виходи;
- створити паузу;
- додати монети та магазин;
- написати скрипт обертання світу, скрипт виходів, кнопки, монет та зміни скінів;
- додати обробку звуків та фонову музику.

Після виконання всіх перерахованих вимог ми отримаємо об'єктно-орієнтовану модель гри.

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОЇ ГРИ «LAST CHANSE»

2.1 Об’єктно-орієнтований аналіз

Моя гра належить до жанру головоломки, що включає дві групи рухомих об'єктів - ігрове поле, яке може рухатися на 90 градусів вліво або вправо, та кубики, які рухаються лише вздовж осі Y. Щоб гравець міг керувати персонажем, я розробив скрипт, який відповідає за цю функцію. Кожна сучасна гра має інтерфейс, який забезпечує зручність для гравця. Інтерфейс повинен бути яскравим, гармонійним, інтуїтивно зрозумілим та зручним.

Для досягнення мети було вирішено використовувати об'єктно-орієнтований підхід, що дозволить уникнути проблем проектування, характерних для процедурного підходу. Об'єктно-орієнтований підхід дозволяє створити чітку структуру коду, що робить його більш читабельним та зручним для модифікацій.

У Unity є базовий клас MonoBehavior, який є батьківським класом всіх класів. Використання цього класу дозволяє використовувати всі його вбудовані методи у дочірніх класах. У цьому дипломному проєкті всі скрипти наслідували саме цей клас.

Алгоритм для об'єктно-орієнтованої програми гри можна представити у наступному вигляді:

– запуск гри;
– вибір налаштувань;
– вибір рівня або продовжити з поточного рівня;
– при зміні положення ігрового поля кубики починають падати в низ;
– якщо вивести сірій кубик за межі поля то гравець програє;
– якщо будь-який кубик торкається монети, то вона збирається;
– якщо з верху від виходу певного кольору знаходиться кубик також кольору то вихід втрачає свої фізичні властивості;

– при натисненні на жовтої кнопки на ігровому полі втрачає фізичні властивості жовта стінка і через неї можуть пройти всі куби;

– кожен рівень повинен мати кнопку паузи;

– в паузі повинно бути три кнопки, перша це вихід до головного меню, друга це перезапуск рівня, а третя це продовжити грати;

– в головному меню повинен бути магазин текстур кубиків.

Моя гра працює по схемі, наведеній нижче (рис. 4).

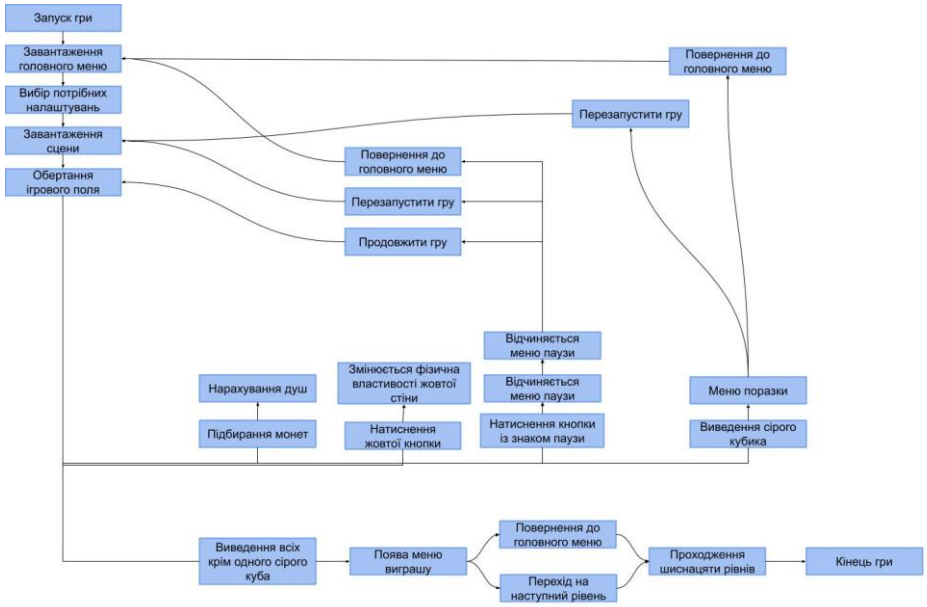

Рисунок 4 Схема роботи гри

Модель системи можна побачити на трасі подій (рис. 5)

Рисунок 5 Траса подій

2.2 Об’єктно-орієнтоване проектування

Для початку потрібно було знайти необхідний матеріал, а саме: спрайти, звуки, музику. Для початку я вирішив створити спрайти кубиків, стін, кнопки, виходів та головного меню. Спрайти стін розставлялись за допомогою інструменту Tilemap. Цей інструмент створює сітку потрібної розмірності по якій виставляються спрайти. Завдяки цьому спрайти виставляються швидко, без стиків. Сітка повинна бути трохи менше самих спрайтів, щоб не залишалось пусте місце між ними. Тайлу стіни я додав BoxCollider та Rigidbody2D для обробки зіткнень.

На рисунку 6 зображені діаграми класів.

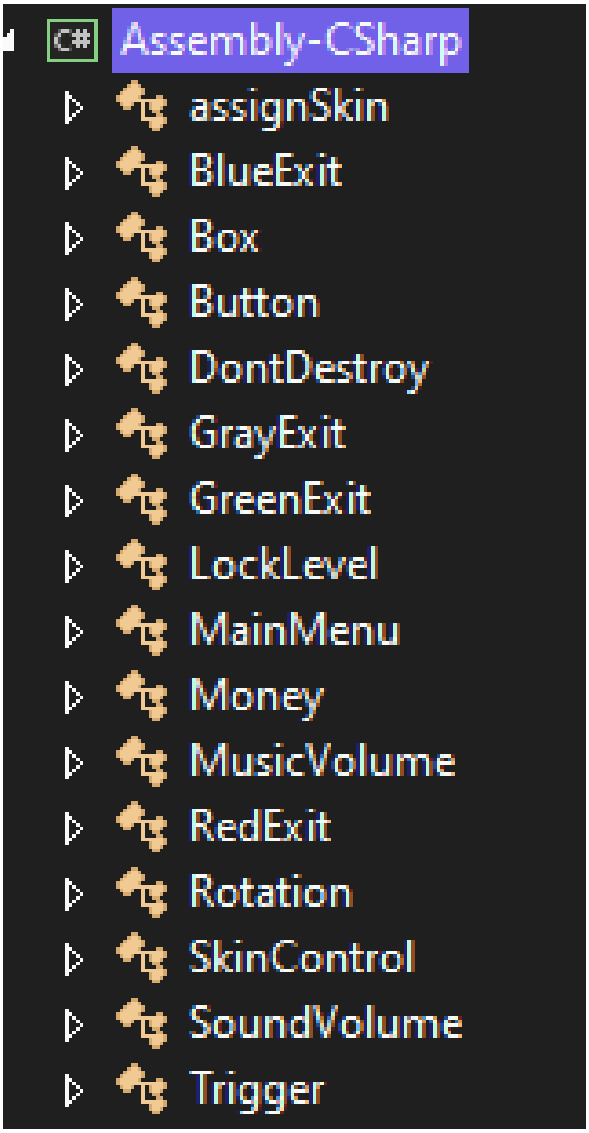

Рисунок 6 Діаграма класів

На рисунку 7 Зображені усі методи та атрибути класів.

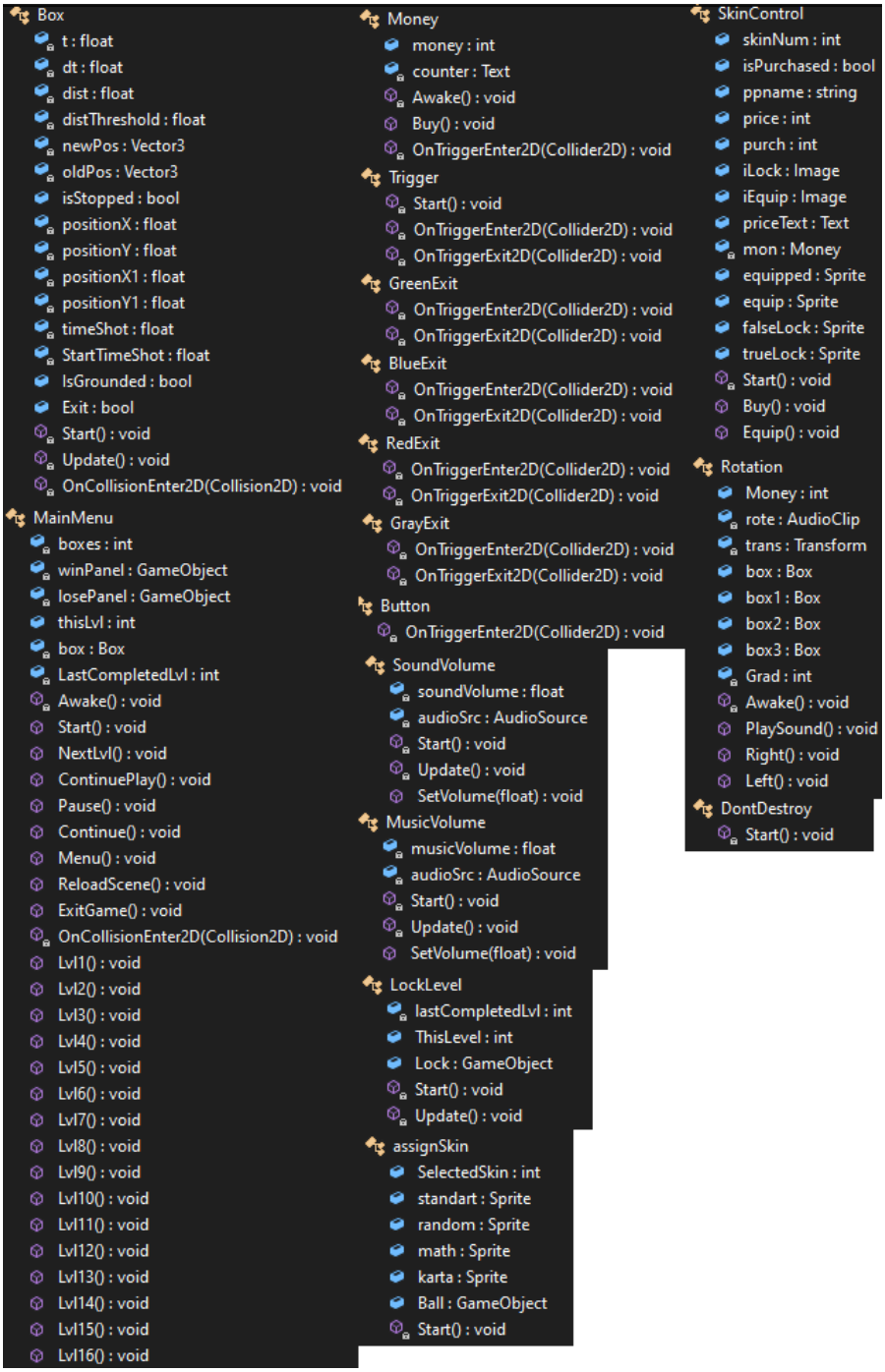

Рисунок 7 Методи та атрибути класів

AssignSkin:

– Start() – перевіряє, чи вже збережено вибір гравцем певного скину та встановлює його для головного героя за допомогою GetComponent<SpriteRenderer>().sprite.

BlueExit:

– OnTriggerEnter2D(Collider2D collision) – викликається, коли колайдер об'єкту потрапляє в тригер-колайдер. Якщо тег об'єкту-зіткнення дорівнює "Blue", то вимикається компонент BoxCollider2D, до якого додано цей скрипт;

– OnTriggerExit2D(Collider2D collision) – викликається, коли колайдер об'єкту виходить з тригер-колайдеру. Якщо тег об'єкту-зіткнення дорівнює "Blue", то у компонента BoxCollider2D знову включається фізика.

Box:

– Start() – ця функція викликається в момент створення екземпляру класу і ініціалізує змінну oldPos, яка зберігає початкову позицію об'єкту;

– Update() – ця функція викликається кожен кадр. Вона рахує різницю між поточною позицією об'єкту та попередньою, використовуючи вектор newPos та змінні dist і distThreshold. Якщо різниця менша за distThreshold, то значення змінної isStopped встановлюється в true. Якщо змінна Exit рівна true, то також встановлюється значення isStopped в true;

– OnCollisionEnter2D(Collision2D collision) – ця функція викликається, коли об'єкт зіштовхується з іншим об'єктом. У цій функції значення змінної isStopped встановлюється в true.

DontDestroy:

– Start() – робить так щоб при зміні сценфи об’єкт не знищувався.

GrayExit:

– OnTriggerEnter2D(Collider2D collision) – викликається, коли колайдер об'єкту потрапляє в тригер-колайдер. Якщо тег об'єкту-зіткнення дорівнює "Gray", то вимикається компонент BoxCollider2D, до якого додано цей скрипт;

– OnTriggerExit2D(Collider2D collision) – викликається, коли колайдер об'єкту виходить з тригер-колайдеру. Якщо тег об'єкту-зіткнення дорівнює "Gray", то у компонента BoxCollider2D знову включається фізика.

GreenExit:

– OnTriggerEnter2D(Collider2D collision) – викликається, коли колайдер об'єкту потрапляє в тригер-колайдер. Якщо тег об'єкту-зіткнення дорівнює "Green", то вимикається компонент BoxCollider2D, до якого додано цей скрипт;

– OnTriggerExit2D(Collider2D collision) – викликається, коли колайдер об'єкту виходить з тригер-колайдеру. Якщо тег об'єкту-зіткнення дорівнює "Green", то у компонента BoxCollider2D знову включається фізика.

LockLevel:

– Start () – при запуску встановлює значення змінної lastCompletedLvl як 1, якщо це значення відсутнє в PlayerPrefs. Якщо ж ключ зі значенням існує, то воно зчитується з PlayerPrefs і встановлюється як значення lastCompletedLvl;

– Update () – перевіряє чи поточний рівень (ThisLevel) менше або дорівнює останньому завершеному рівню (lastCompletedLvl). Якщо це так, то блокуючий знак Lock вимикається (SetActive (false)). Якщо ThisLevel більше lastCompletedLvl, то Lock залишається увімкненим (блокованим).

MainMenu:

– Awake () – перевіряє, чи вперше гравець запускає гру, якщо так, то встановлює LastCompletedLvl на 1 і FirstLaunch на 1 в PlayerPrefs;

– Start () – встановлює значення LastCompletedLvl збережене в PlayerPrefs або 1, якщо такого значення не існує;

– NextLvl () – збільшує значення LastCompletedLvl на 1, зберігає це значення в PlayerPrefs та завантажує наступний рівень;

– ContinuePlay () – завантажує наступний рівень, якщо гравець не закінчив жодного рівня, або наступний рівень після останнього, який гравець успішно пройшов. Зберігає це значення в PlayerPrefs;

– Pause () – зупиняє час гри;

– Continue () – відновлює час гри;

– Menu () – завантажує головне меню гри, і збільшує значення LastCompletedLvl на 1, якщо гравець пройшов поточний рівень;

– ReloadScene () – перезавантажує поточний рівень;

– ExitGame () – закриває гру;

– OnCollisionEnter2D () – реагує на зіткнення з об'єктом, збільшує значення boxes, видаляє ящик, якщо він виходить за межі камери і відображає відповідні панелі перемоги або поразки, в залежності від того, чи гравець зібрав достатню кількість ящиків;

– Lvl1 ()– завантажує відповідний рівень;

– Lvl2 ()– завантажує відповідний рівень;

– Lvl3 ()– завантажує відповідний рівень;

– Lvl4 ()– завантажує відповідний рівень;

– Lvl5 ()– завантажує відповідний рівень;

– Lvl6 ()– завантажує відповідний рівень;

– Lvl7 ()– завантажує відповідний рівень;

– Lvl8 ()– завантажує відповідний рівень;

– Lvl9 ()– завантажує відповідний рівень;

– Lvl10 ()– завантажує відповідний рівень;

– Lvl11 ()– завантажує відповідний рівень;

– Lvl12 ()– завантажує відповідний рівень;

– Lvl13 ()– завантажує відповідний рівень;

– Lvl14 ()– завантажує відповідний рівень;

– Lvl15 ()– завантажує відповідний рівень;

– Lvl16 ()– завантажує відповідний рівень.

Money:

– Awake() – ця функція викликається коли гра починається і встановлює початкову кількість грошей, якщо значення збережене у PlayerPrefs;

– Buy() – ця функція викликається, коли гравець натискає кнопку купівлі. Вона збільшує кількість грошей, оновлює їх значення на екрані та зберігає нове значення у PlayerPrefs;

– OnTriggerEnter2D(Collider2D collision) – ця функція викликається, коли гравець перетинає межі тригера. Вона збільшує кількість грошей, оновлює їх значення на екрані та зберігає нове значення у PlayerPrefs.

MusicVolume:

– Start() – встановлює змінну audioSrc на початку роботи скрипту;

– Update() – кожен кадр встановлює гучність музики відповідно до значення musicVolume, яке зберігається в PlayerPrefs, якщо таке значення є;

– SetVolume(float vol) – зберігає значення гучності музики vol в PlayerPrefs та оновлює значення musicVolume відповідно до цього значення.

RedExit:

– OnTriggerEnter2D(Collider2D collision) – викликається, коли колайдер об'єкту потрапляє в тригер-колайдер. Якщо тег об'єкту-зіткнення дорівнює "Red", то вимикається компонент BoxCollider2D, до якого додано цей скрипт;

– OnTriggerExit2D(Collider2D collision) – викликається, коли колайдер об'єкту виходить з тригер-колайдеру. Якщо тег об'єкту-зіткнення дорівнює "Red", то у компонента BoxCollider2D знову включається фізика.

Rotation:

– Awake() – це метод, який викликається один раз під час створення об'єкта. Він перевіряє, чи є збережений значення грошей у PlayerPrefs та встановлює початкове значення грошей;

– PlaySound() – цей метод програє звуковий кліп, який призначений для відтворення звуку під час обертання об'єкта;

– Right() – цей метод повертає об'єкт вправо на 90 градусів, якщо всі ігрові об'єкти зупинені;

– Left() – цей метод повертає об'єкт вліво на 90 градусів, якщо всі ігрові об'єкти зупинені.

SkinControl:

– Start() – запускається один раз при старті гри і перевіряє, чи є предмет купленим. Якщо так, то блокує зображення замка та відображає зображення екіпіровки, якщо предмет не є екіпірованим, або навпаки, якщо предмет екіпірований;

– Buy() – перевіряє, чи грошей гравця достатньо для купівлі предмета. Якщо так, віднімає вартість предмета від грошей гравця, купує предмет, блокує зображення замка, відображає зображення екіпіровки та зберігає стан предмета у PlayerPrefs. Якщо предмет вже куплений, то екіпірує його;

– Equip() – екіпірує предмет, якщо він куплений.

SoundVolume:

– Start() – встановлює змінну audioSrc на початку роботи скрипту;

– Update() – кожен кадр встановлює гучність звуків відповідно до значення soundVolume, яке зберігається в PlayerPrefs, якщо таке значення є;

– SetVolume(float vol) – зберігає значення гучності музики vol в PlayerPrefs та оновлює значення soundVolume відповідно до цього значення.

Trigger:

– Start() – метод, який викликається один раз при запуску скрипту. Встановлює колайдер на початку вимкненим, щоб об'єкт не був активним відразу після запуску;

– OnTriggerEnter2D(Collider2D collision) – метод, який викликається, коли якийсь об'єкт потрапляє в тригер. У виконанні цього методу колайдер вмикається;

– OnTriggerExit2D(Collider2D collision) – метод, який викликається, коли об'єкт виходить з тригера. У виконанні цього методу колайдер вимикається.

2.3 Об‘єктно-орієнтоване програмування

Фрагмент програмного коду представлений у додатку В

2.4 Впровадження бази даних та мета використання бази даних

Використання бази даних у проєкті комп'ютерної гри «Last Chanse» дозволяє ефективно зберігати та управляти ігровими даними, такими як результати гравців, рівні, налаштування та інші важливі параметри. Основні цілі використання бази даних включають:

1. Збереження прогресу гри: База даних зберігає інформацію про поточний стан гри, що дозволяє гравцям продовжити гру з того місця, де вони зупинилися.

2. Збереження налаштувань гри: Налаштування гри, такі як рівень складності, звукові налаштування та інші параметри, зберігаються у базі даних.

3. Зберігання результатів гри: База даних дозволяє зберігати результати гравців, що дозволяє організувати таблицю рекордів та підвищити мотивацію гравців досягати кращих результатів.

2.5 Вибір СУБД

Для реалізації бази даних була обрана система управління базами даних (СУБД) SQLite (рис 8).

Рисунок 8 СУБД SQLite

Основними причинами вибору саме цієї СУБД стали:

1. Легковажність та простота використання: SQLite є компактною та не потребує окремого серверного програмного забезпечення, що спрощує її інтеграцію у невеликі проекти (рис 9).

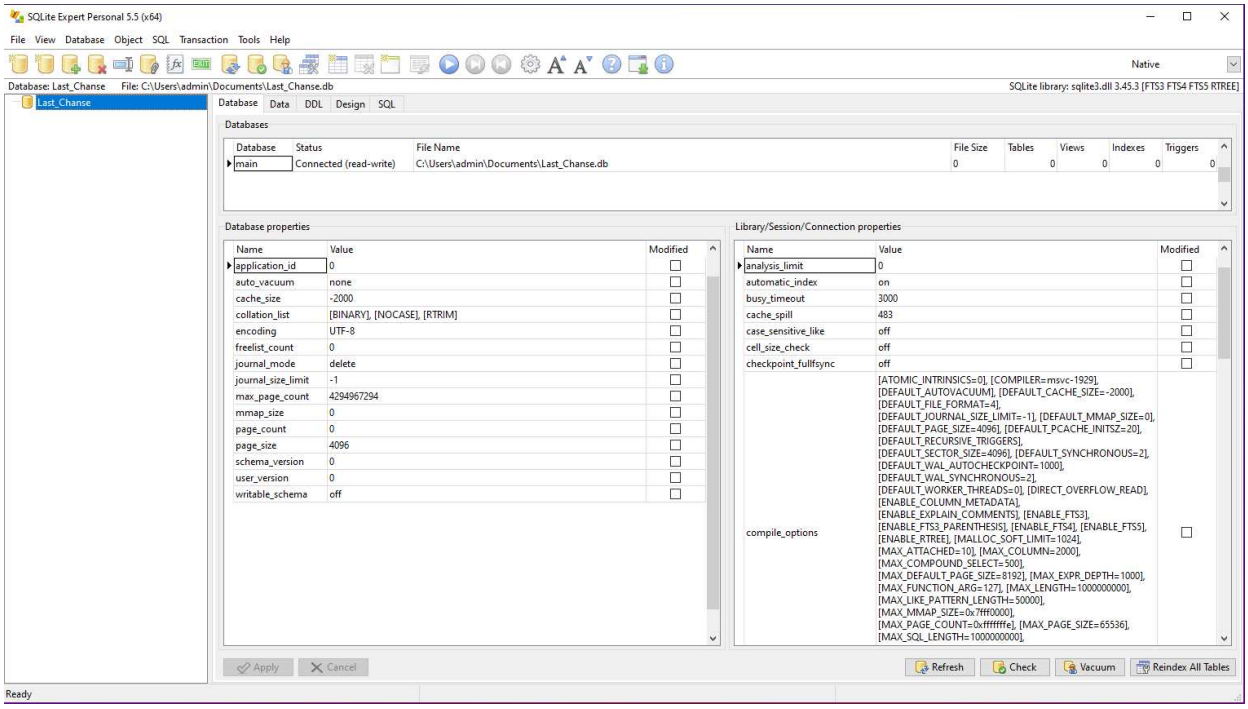

Рисунок 9 Інтерфейс SQLite

2. Переносність: База даних зберігається у вигляді одного файлу, що робить її легкою для переносу та резервного копіювання.

3. Продуктивність: SQLite забезпечує достатню продуктивність для більшості потреб невеликих та середніх проектів, включаючи зберігання даних гри.

2.6 Структура та логічна схема бази даних

База даних для гри «Last Chanse» складається з наступних двох основних таблиць: `GameProgress` та `GameSettings` (рис.10). Ці таблиці зберігають інформацію про прогрес гравця та налаштування гри.

Таблиця GameProgress:

- `id` (INTEGER, PRIMARY KEY, AUTOINCREMENT): унікальний ідентифікатор запису.
- `level` (INTEGER): поточний рівень гравця.
- `score` (INTEGER): поточний рахунок гравця.
- `timestamp` (TEXT): час збереження прогресу.

Таблиця GameSettings:

- `id` (INTEGER, PRIMARY KEY, AUTOINCREMENT): унікальний ідентифікатор запису.

- `volume` (INTEGER): рівень гучності.

- `difficulty` (TEXT): рівень складності (наприклад, "легкий", "середній", "важкий").

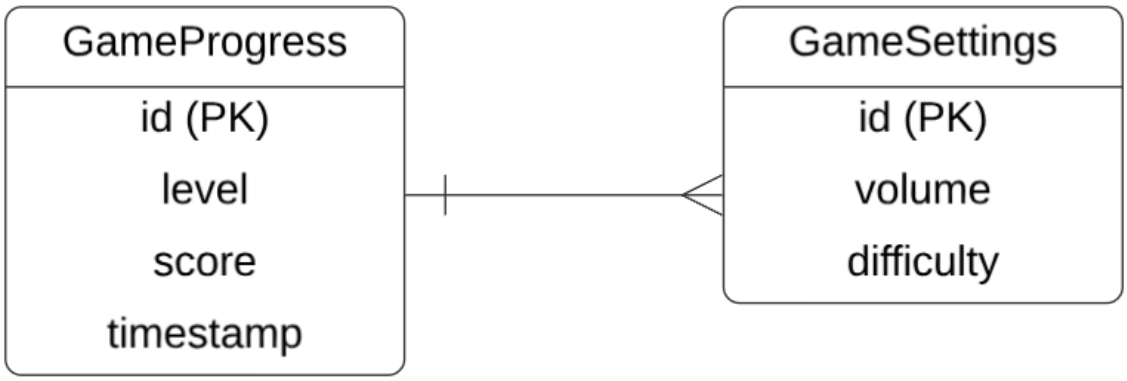

Рисунок 10 Логічна блок схема

2.7 Реалізація операцій CRUD

Для роботи з базою даних у проєкті реалізовані основні операції CRUD (Create, Read, Update, Delete).

Фрагмент програмного коду представлений у додатку В

2.8 Приклади використання бази даних у грі

Для більш повного розуміння того, як база даних інтегрується у гру «Last Chanse», розглянемо кілька прикладів її використання:

1. Збереження ігрового прогресу:

Після кожного завершеного рівня або досягнення важливого результату, прогрес гравця (рівень та рахунок) зберігається у таблиці GameProgress, що дозволяє гравцю продовжити гру з того ж місця при наступному запуску.

2. Завантаження ігрового прогресу:

При запуску гри, останній збережений прогрес завантажується з бази даних, що дозволяє гравцю продовжити гру з того місця, де він зупинився.

3. Збереження налаштувань гри:

Гравець може змінювати налаштування гри, такі як рівень гучності або складності. Ці зміни зберігаються у базі даних і завантажуються при наступному запуску гри.

4. Оновлення налаштувань гри:

Гравець може в будь-який момент змінити налаштування гри, і ці зміни будуть збережені у базі даних для використання у майбутніх сесіях гри.

Таким чином, впровадження бази даних у проєкт «Last Chanse» дозволяє забезпечити зручне та ефективне управління ігровими даними, зберігання прогресу та налаштувань, а також загальне покращення ігрового досвіду.

РОЗДІЛ 3.РЕАЛІЗАЦІЯ ГРИ

Щоб запустити гру потрібно натиснути на іконку (рис. 8).

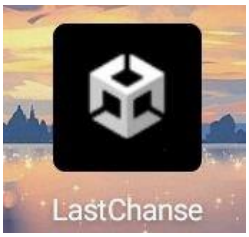

Рисунок 11 Іконка гри

Після завантаження гри відкривається головне меню (рис. 9).

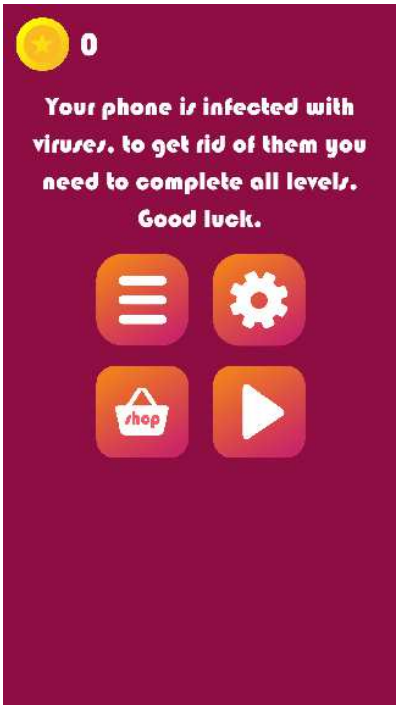

Рисунок 12 Головне меню

Головне меню містить наступні кнопки:

– список рівнів (рис. 10);
– налаштування (рис. 11);
– магазин (рис. 12);
– продовжити з останнього рівня (рис. 13).

Рисунок 13 Список рівнів

Рисунок 14 Налаштування

Рисунок 15 Магазин

Рисунок 16 Продовжити з останнього рівня

Список рівнів (рис. 17, 18) зроблений таким чином, що на одній панелі відображається по 8 рівнів, панелі перелистуються.

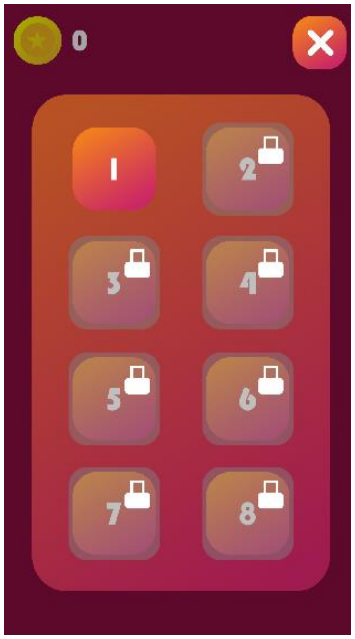

Рисунок 17 Перша панель списку рівнів

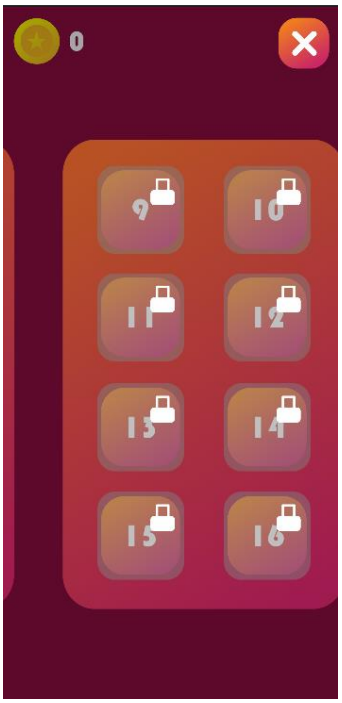

Рисунок 18 Друга сторінка списку рівнів

В меню налаштувань (рис. 19) можна змінити гучність звуків та музики.

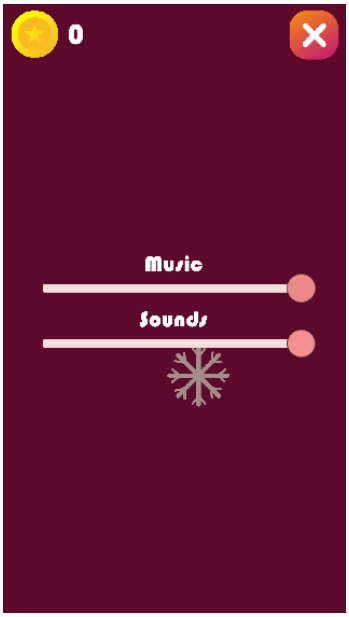

Рисунок 19 Меню налаштувань

В магазині (рис. 20) гравець може купити будь-який скін для кубиків, якщо йому вистачає на нього коштів. Для цього потрібно натиснути кнопку «Equip» (рис. 21) під відповідним скіном.

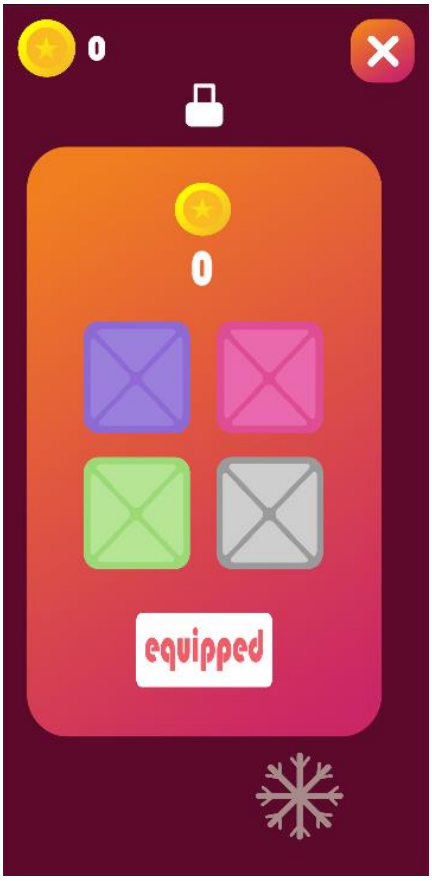

Рисунок 20 Магазин

Рисунок 21 Кнопка «Equip»

В кожному вікні, що відкривається є кнопка, що закриває його (рис. 19).

Рисунок 22 Кнопка закриття вікна

Після першого запуску гри при натисканні кнопки продовження гри за замовченням відкривається перший рівень (рис. 23).

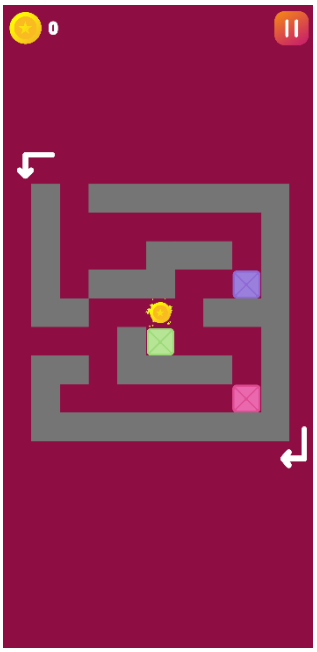

Рисунок 23 Перший рівень

Гравець може обертати світ натискаючи на праву чи ліву частину екрану. В який бік обернеться лабіринт вказують відповідні стрілки на його кутах. Гравець може поставити паузу під час гри (рис. 24) за допомогою кнопки паузи (рис. 25).

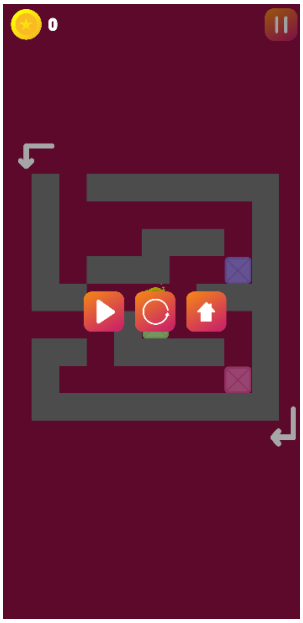

Рисунок 24 Меню паузи

Рисунок 25 Кнопка паузи

Меню паузи містить наступні кнопки:

– продовжити (рис. 26);

– перезапуск (рис. 27);

– вихід у головне меню (рис. 28).

Рисунок 26 Кнопка продовжити

Рисунок 27 Кнопка перезапуску

Рисунок 28 Кнопка виходу у головне меню

Якщо гравець проходить рівень відкривається меню перемоги (рис. 29).

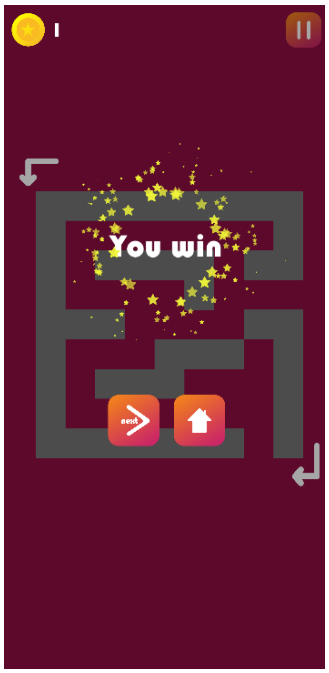

Рисунок 29 Меню перемоги

Меню перемоги містить наступні кнопки:

– перехід на наступний рівень (рис. 30);

– вихід у головне меню.

Рисунок 30 Кнопка переходу на наступний рівень

Якщо сірий кубик покидає ігрове поле з’являється меню програшу (рис. 31).

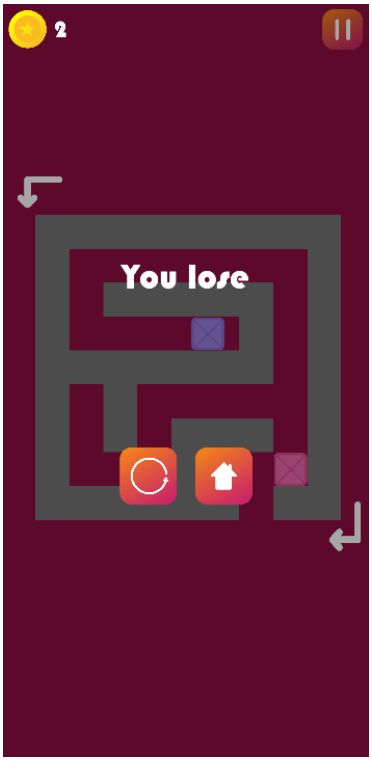

Рисунок 31 Меню програшу

Меню перемоги містить наступні кнопки:

– перезапуску;

– вихід у головне меню.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці

При аналізі умов праці необхідно визначити фактори, які можуть негативно впливати на здоров'я та продуктивність розробників і спробувати їх зменшити або уникнути. Наприклад, для зменшення нервово-емоційного напруження можна використовувати практики, такі як медитація або йога, або збільшувати відпочинкові перерви.

Для боротьби з монотонністю праці можна використовувати різноманітні інструменти та розваги, або впроваджувати різноманітні проєкти, які можуть змінюватись залежно від потреб компанії.

Освітлення та вентиляція також важливі для здоров'я розробників. Недостатня вентиляція може спричинити концентрацію шкідливих речовин в повітрі, що може вплинути на здоров'я. Щодо освітлення, важливо забезпечити достатній рівень природного та штучного світла.

Невідповідність ергономічних показників робочого місця може призвести до зниження продуктивності та проблем зі здоров'ям. Розробники можуть розглянути можливість встановлення ергономічного обладнання та регулювання висоти стільців та столів.

Для забезпечення належних умов праці важливо дотримуватися вимог з охорони праці та здоров'я працівників, проводити регулярні медичні огляди, забезпечувати навчання з охорони праці та безпеки, і організовувати робочий процес.

Враховуючи все вищезазначене, при створенні ігор необхідно пам'ятати про умови праці розробників, щоб забезпечити їх ефективність та здоров'я. Необхідно дотримуватися принципу балансу між роботою та відпочинком, забезпечувати належні умови освітлення та вентиляції, розміщення обладнання та ергономіку робочого місця.

Крім того, важливо забезпечувати навчання та інструктажі з охорони праці та безпеки, а також проводити регулярні медичні огляди працівників. Застосування практик, які допомагають зняти стрес та зберегти мотивацію, таких як медитація або йога, та розваги на робочому місці можуть покращити ефективність роботи та здоров'я працівників.

4.2 Гігієна праці та виробнича санітарія

Один з найважливіших аспектів гігієни праці - дотримання правил особистої гігієни працівників та забезпечення належної санітарії робочих місць. Працівники повинні дотримуватись правил щодо чистоти рук, а робочі місця повинні бути регулярно очищені від забруднень.

Крім того, використання безпечних матеріалів та обладнання є дуже важливим. При створенні ігор, розробники повинні уважно розглядати питання безпеки та здоров'я працівників та забезпечити дотримання правил безпеки.

Якщо виникають питання щодо безпеки на робочому місці, працівники мають право звернутися до відповідних органів, а роботодавці повинні бути готові відповідати на запити та виконувати необхідні заходи щодо усунення виявлених проблем та запобігання їх повторенню в майбутньому.

Також важливо зазначити, що дотримання вимог гігієни праці та виробничої санітарії сприяє підвищенню продуктивності працівників та зниженню ризику виникнення травм та захворювань пов'язаних з роботою. Забезпечення безпеки та здоров'я працівників є однією з ключових вимог для досягнення успіху та стабільного розвитку будь-якого підприємства.

Навчання працівників щодо вимог гігієни праці та виробничої санітарії є також важливим елементом забезпечення безпеки праці. Розробники ігор повинні забезпечити навчання працівників щодо правил безпеки на робочому місці та надати їм необхідні інструкції щодо використання обладнання та матеріалів.

Крім того, розробники повинні вести постійний моніторинг за дотриманням вимог гігієни праці та виробничої санітарії на підприємстві. Це дозволяє вчасно виявляти можливі проблеми та запобігати їх розвитку. Постійний моніторинг є важливим елементом програми забезпечення безпеки праці, яка повинна бути введена в дію на кожному підприємстві, що займається розробкою ігор.

Узагалі, безпека та здоров'я працівників є необхідним елементом успішної роботи будь-якого підприємства, зокрема тих, що займаються розробкою ігор. Забезпечення безпеки та здоров'я працівників повинно бути пріоритетним завданням для розробників ігор, а також роботодавців. Загалом опікування над здоров’ям робітників приносить більше користі, ніж зайвих зусиль, оскільки забезпечення здоров'я та безпеки працівників не тільки зменшує ризик виникнення травм та захворювань, але також підвищує їх продуктивність та задоволеність роботою. Безпечне та здорове робоче середовище є ключовим фактором у забезпеченні стійкого розвитку підприємства та підвищенні його конкурентоспроможності на ринку.

У кінцевому результаті, створення безпечного та здорового робочого середовища є спільною відповідальністю роботодавців, працівників та відповідних органів. Роботодавці повинні забезпечувати належні умови праці та виробничої санітарії, надавати необхідне навчання та підтримку працівникам щодо дотримання правил безпеки. Проте, працівники також мають відповідальність за свою особисту гігієну та дотримання правил безпеки на робочому місці.

Тому, щоб забезпечити безпечне та здорове робоче середовище, необхідно планувати та реалізовувати програми забезпечення безпеки праці, проводити періодичні навчання та медичні огляди, забезпечувати належну виробничу санітарію та застосовувати безпечні технології та обладнання. Всі ці заходи допоможуть зберегти здоров'я та безпеку працівників, підвищити їх ефективність та задоволеність роботою, а також забезпечити стійкий розвиток підприємства та його успіх на ринку.

4.3 Пожежна безпека виробничого приміщення

Пожежна безпека є надзвичайно важливою складовою охорони життя та здоров'я людей у будь-якому приміщенні, включаючи офісні приміщення. Забезпечення пожежної безпеки в офісі передбачає ряд заходів та процедур, що повинні бути дотримані всіма працівниками.

Одним із перших кроків у забезпеченні пожежної безпеки є перевірка приміщення на наявність вогненебезпечних матеріалів та їхнє відповідне зберігання. Такі матеріали повинні зберігатися в спеціально обладнаних зонах з відповідними попереджувальними знаками.

Не слід забувати про регулярну перевірку та технічне обслуговування систем пожежної безпеки. Це включає перевірку та заміну батарей в димових та вуглекислотних детекторах, регулярну перевірку електричних приладів та системи пожежного сповіщення. Щорічна перевірка системи пожежної безпеки повинна бути проведена фахівцями-пожежниками.

Для забезпечення швидкого виходу з приміщення у разі пожежі необхідно регулярно проводити навчання працівників правилам поведінки під час пожежі, а також забезпечити наявність видимої та зрозумілої системи евакуації, зокрема, планів евакуації та попередньо визначених маршрутів.

У разі виникнення пожежі необхідно вміти швидко та правильно реагувати на ситуацію. Для цього повинні бути встановлені засоби пожежогасіння, такі як вогнегасники, та працівники повинні знати, як їх використовувати. Також варто забезпечити регулярну перевірку та технічне обслуговування засобів пожежогасіння та інших систем пожежної безпеки.

Нарешті, у разі виникнення пожежі важливо швидко повідомити про це пожежну охорону, а також викликати швидку допомогу у разі потреби. Необхідно також дотримуватися всіх інших вимог, що стосуються пожежної безпеки, щоб забезпечити безпеку всіх працівників та відвідувачів приміщення.

4.4 Виконання розрахункової частини

Аналіз робочого місця (рис. 32) та порівняння його з існуючими санітарно-гігієнічними нормами (табл. 1).

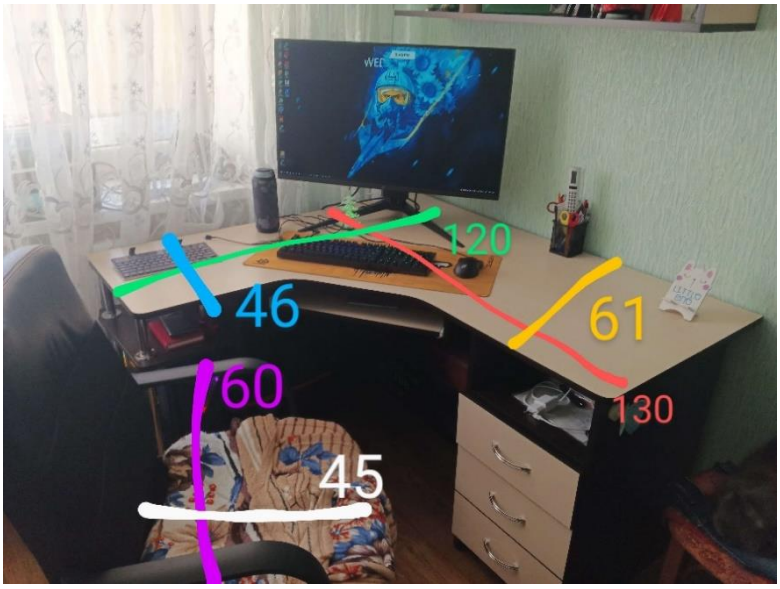

Рисунок 32 Робоче місце

Таблиця 1 Оцінка параметрів робочого місця нормам

Фактор виробничого середовища	Санітарно-гігієнічні норми	Існуючі показники	Висновок
Температура	19-25ºС	23 ºС	В нормі
Відносна вологість	40-60%	-	-
Освітленість робочої поверхні	Не менше 300Лк	-	-
Відстань від очей до монітора	600-700мм	620	В нормі

Висота столу	680-720	760	Зависокий
Ширина столу	900	610	Замалий
Довжина столу	1600	1300	Замалий
Висота стільця	400-500	480	В нормі
Розміри сидіння стільця	380-400	550	Завеликий
Площа на одного працівника	6м^2 (мінімально 4,5м^2)	6,5 м^2	В нормі

Розрахунок необхідної потужності кондиціонеру для даного приміщення.

$Q_x = Q_З + Q_О + Q_Р = 546 + 337{,}2 + 0{,}1 = 883{,}3$Вт

$Q_З = S * h * q = 6{,}5 * 2{,}4 * 35 = 546$кВт

$S = a * b = 2{,}5 * 2{,}6 = 6{,}5$м^2

$Q_О = 0{,}3 * p + 1 * Q_{ok} = 0{,}3 * 1123 + 0{,}3 = 337{,}2$кВт

$Q_Р = n_ч * q_ч + n_ж * q_ж = 1 * 0{,}1 = 0{,}1$кВт

Отже для достатнього кондиціонування приміщення підходить кондиціонер моделі Веко ВР 207 С потужністю 1,9кВт

Розрахунок необхідної кількість світильників, кожен світильник має по 2 лампочки.

$E = 500$ лк; $К_з = 1{,}7$; $S = 2{,}5 * 2{,}6 = 6{,}5$м^2; $Z = 1{,}1$ $Ф = 2000$ лм; $n = 2; 5$

U=64/100=0,64

$$N = \frac{E * К_з * S * Z}{Ф * U * n} = \frac{500 * 1{,}7 * 240 * 1{,}1}{2000 * 0{,}64 * 2} = 8{,}8 => 9$$

$$i = \frac{A * B}{h * (A + B)} = \frac{6{,}5}{2{,}4 * 5{,}1} = 0{,}53$$

$h = H - h_p - h_c = 2{,}4 - 0{,}76 - 0{,}2 = 1{,}44$м

Отже приміщення повинно мати 9 світильників по 2 лампочки, які розміщені на висоті 1,44м.

Для пожежної безпеки даного офісного приміщення достатньо мати 1 вогнегасник ВВПА-400.

ВИСНОВОК

Під час виконання дипломної роботи була розроблена гра «Last Chanse». Під час створення гри використовувалися класи, спадкування, вбудовані функції Unity. При розробці гри я отримав практичні навички у роботі з класами та їх взаємозв'язками.

Під час розробки гри були враховані всі, поставлені раніше, вимоги. Для розробки гри було використано наступні компоненти:

- колайдери;
- спрайти;
- скрипти;
- компонент RigidBody2D;
- Particle System;
- Slider;
- Button.

Колайдер описує розміри об'єкта, а RigidBody2D завдяки колайдеру може обробляти зіткнення з іншими ігровими об'єктами. Particle System дозволяє створювати гарні спецефекти. За допомогою компонента Slider регулюється гучність музики та звуків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ігнатенко, М. О. Розробка ігор на Unity: від ідеї до реалізації. — Київ: Наукова думка, 2020. — 456 с.
2. Петренко, О. В. Об'єктно-орієнтоване програмування на C#. — Харків: Видавництво "Ранок", 2021. — 384 с.
3. Коваленко, Ю. І. Вступ до програмування на C# для розробників ігор. — Львів: ПАІС, 2019. — 512 с.
4. Мельник, І. М. Використання Unity для створення 2D ігор. — Одеса: ОНУ, 2022. — 304 с.
5. Соколовський, В. П. Програмування ігор на платформі .NET і C#. — Дніпро: ДНУ, 2023. — 422 с.
6. Федоренко, А. С. Розробка ігрових додатків на Unity з використанням C#. — Запоріжжя: ЗНУ, 2021. — 360 с.
7. [Електронний ресурс] — Unity Documentation. — Режим доступу: https://docs.unity3d.com/
8. [Електронний ресурс] — Офіційний сайт мови програмування C#. — Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
9. [Електронний ресурс] — Habrahabr: Статті про розробку ігор. — Режим доступу: https://habr.com/ru/hub/gamedev/
10. [Електронний ресурс] — CyberForum: Форум розробників. — Режим доступу: https://www.cyberforum.ru/
11. Гнатюк, М. В. Використання LINQ в програмуванні на C#. — Полтава: ПНТУ, 2020. — 384 с.

ДОДАТОК Б – ІНСТРУКЦІЯ КОРИСТУВАЧА

Основні правила гри

1. Мета гри:
    - Витягнути з лабіринту всі кольорові кубики, не витягнувши сірий кубик.
    - Кубики піддаються дії гравітації, і їх потрібно витягувати шляхом обертання лабіринту.
2. Керування:
    - Обертання лабіринту здійснюється натисканням на праву або ліву частину екрану.
    - Напрямок обертання позначено стрілками на кутах лабіринту.
    - У кожному рівні є пауза, яку можна активувати натисканням відповідної кнопки.
3. Гра:
    - Після запуску гри ви потрапляєте у головне меню, яке містить кнопки для переходу до списку рівнів, налаштувань, магазину і продовження з останнього рівня.
    - Спочатку відкривається перший рівень, де гравець вчиться основним принципам керування.
    - Гра розділена на рівні, кожен з яких ускладнюється з новими механіками і викликами.
4. Кубики і виходи:
    - Лабіринт має кілька кубиків різних кольорів і сірий кубик.
    - Гравець повинен повертати лабіринт, щоб кольорові кубики потрапляли у відповідні виходи.
    - Сірий кубик не повинен покинути лабіринт.
5. Монети та магазин:
    - Під час гри гравці можуть збирати монети, які випадають з рівнів.
    - Монети можна обміняти в магазині на скіни для кубиків.

- Щоб купити скіни, потрібно натиснути кнопку "Equip" під відповідним скином.

Інтерфейс гри

1. Головне меню:
    - Список рівнів: дозволяє переглядати доступні рівні і вибирати їх для гри.
    - Налаштування: дозволяє змінювати гучність звуків та музики.
    - Магазин: дає можливість купувати скіни для кубиків.
    - Продовжити: продовжує гру з останнього пройденого рівня.
2. Меню паузи:
    - Продовжити: повернення до гри.
    - Перезапуск: перезапускає поточний рівень.
    - Вихід у головне меню: вихід до головного меню гри.
3. Меню перемоги:
    - Перехід на наступний рівень.
    - Вихід у головне меню.
4. Меню програшу:
    - Перезапуск рівня.
    - Вихід у головне меню.

Технічні деталі

- Гра створена на рушії Unity з використанням мови програмування C#.
- Використовується об'єктно-орієнтований підхід для створення чіткої структури коду та зручності в модифікаціях.

Системні вимоги

- Платформи: iOS, Android
- Мінімальні вимоги:
    - iOS 10.0 або новіше.
    - Android 5.0 або новіше.

ДОДАТОК В - ФРАГМЕНТ ПРОГРАМНОГО КОДУ

2.3 Об‘єктно-орієнтоване програмування

Скрипт Rotation:

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

public class Rotation : MonoBehaviour
{
    public int Money;

    [SerializeField] AudioClip rote;

    private Transform trans;
    public Box box;
    public Box box1;
    public Box box2;
    public Box box3;
    private int Grad = 0;

    void Awake()
    {
        if (PlayerPrefs.HasKey("Money"))
        {
            Money = PlayerPrefs.GetInt("Money");
        }
        PlayerPrefs.SetInt("Money", Money);
```

```
        }
        public void PlaySound()
        {
            GetComponent<AudioSource>().PlayOneShot(rote);
        }
        public void Right()
        {
            if (box.isStopped && box1.isStopped && box2.isStopped && box3.isStopped)
            {
                Grad -= 90;
                transform.rotation = Quaternion.Euler(0, 0, Grad);
                PlaySound();
            }
        }
        public void Left()
        {
            if (box.isStopped && box1.isStopped && box2.isStopped && box3.isStopped)
            {
                Grad += 90;
                transform.rotation = Quaternion.Euler(0, 0, Grad);
                PlaySound();
            }
        }
    }
```

Скрипт SkinControl:

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

```
using UnityEngine;
using UnityEngine.UI;

public class SkinControl : MonoBehaviour
{

    public int skinNum;
    public bool isPurchased;
    public string ppname;
    public int price;
    public int purch;
    public Image iLock;
    public Image iEquip;
    public Text priceText;
    Money mon;

    public Sprite equipped;
    public Sprite equip;
    public Sprite falseLock;
    public Sprite trueLock;

    private void Start()
    {
        mon = FindObjectOfType<Money>();
        purch = PlayerPrefs.GetInt(ppname, 0);
        if(purch == 0)
        {
            priceText.text = price.ToString();
            iLock.GetComponent<Image>().sprite = trueLock;
```

```
        if(PlayerPrefs.GetInt("Selected") == skinNum)
        {
            iEquip.GetComponent<Image>().sprite = equipped;
        }
        else
        {
            iEquip.GetComponent<Image>().sprite = equip;
        }
    }
    else
    {
        iLock.GetComponent<Image>().sprite = falseLock;
        isPurchased = true;
        priceText.text = price.ToString();
        if (PlayerPrefs.GetInt("Selected") == skinNum)
        {
            iEquip.GetComponent<Image>().sprite = equipped;
        }
        else
        {
            iEquip.GetComponent<Image>().sprite = equip;
        }
    }
}
public void Buy()
{
    if(mon.money >= price && isPurchased == false)
    {
        PlayerPrefs.SetInt("Money", mon.money -= price);
```

```
            mon.Buy();
            PlayerPrefs.SetInt(ppname, 1);
            iLock.GetComponent<Image>().sprite = falseLock;
            priceText.text = "0";
            price = 0;
        }
        else if(isPurchased)
        {
            PlayerPrefs.SetInt("Selected", skinNum);
            iEquip.GetComponent<Image>().sprite = equipped;
        }
    }
    public void Equip()
    {
        if (isPurchased)
        {
            PlayerPrefs.SetInt("Selected", skinNum);
            iEquip.GetComponent<Image>().sprite = equipped;
        }
    }
}
```

2.7 Реалізація операцій CRUD

Для роботи з базою даних у проєкті реалізовані основні операції CRUD (Create, Read, Update, Delete).

Збереження прогресу гри:

```
using System.Data.SQLite;
using UnityEngine;

public class DatabaseManager : MonoBehaviour
{
    private string dbPath;

    void Start()
    {
        dbPath = Application.dataPath + "/game.db";
        CreateDatabase();
    }

    void CreateDatabase()
    {
        if (!System.IO.File.Exists(dbPath))
        {
            SQLiteConnection.CreateFile(dbPath);
            using (var connection = new SQLiteConnection("Data Source=" + dbPath))
            {
                connection.Open();
                string sql = @"
                    CREATE TABLE GameProgress (
```

```
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    level INTEGER,
                    score INTEGER,
                    timestamp TEXT
                );

                CREATE TABLE GameSettings (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    volume INTEGER,
                    difficulty TEXT
                );";

                using (var command = new SQLiteCommand(sql, connection))
                {
                    command.ExecuteNonQuery();
                }
            }
        }
    }

    public void SaveGameProgress(int level, int score)
    {
        using (var connection = new SQLiteConnection("Data Source=" + dbPath))
        {
            connection.Open();
            string sql = "INSERT INTO GameProgress (level, score, timestamp) VALUES (@level, @score, @timestamp)";
            using (var command = new SQLiteCommand(sql, connection))
            {
```

```
                command.Parameters.AddWithValue("@level", level);
                command.Parameters.AddWithValue("@score", score);
                command.Parameters.AddWithValue("@timestamp",
System.DateTime.Now.ToString());
                command.ExecuteNonQuery();
            }
        }
    }

    public void SaveGameSettings(int volume, string difficulty)
    {
        using (var connection = new SQLiteConnection("Data Source=" + dbPath))
        {
            connection.Open();
            string sql = "INSERT INTO GameSettings (volume, difficulty) VALUES
(@volume, @difficulty)";
            using (var command = new SQLiteCommand(sql, connection))
            {
                command.Parameters.AddWithValue("@volume", volume);
                command.Parameters.AddWithValue("@difficulty", difficulty);
                command.ExecuteNonQuery();
            }
        }
    }
}
```