

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Пояснювальна записка

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:
«Технології реалізації n - вимірних кубічних об'єктів в іграх-
головоломках»

Виконав студент групи КН-19-1
СМУЛЬСЬКИЙ Данило Зореславович

Керівник роботи:
СУГОНЯК Інна Іванівна

Рецензент:
КУЗЬМЕНКО Олександр Вікторович

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Зав. кафедри комп'ютерних
наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: СМУЛЬСЬКИЙ Данило Зореславович

Керівник роботи: СУГОНЯК Інна Іванівна

Тема роботи: «Технології реалізації n - вимірних кубічних об'єктів в іграх-головоломках»,

затверджена наказом закладу вищої освіти від «05» січня 2023 р., №10/с

Термін здачі закінченої роботи: «22» червня 2023 р.

Вихідні дані роботи: Предмет дослідження: Гра головоломка. Об'єкт дослідження: Механіка, геймплей, елементи гри головоломки.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Опрацювання літературних джерел.	16 березня 2023 – 12 квітня 2023	Виконано
2	Проектування структури гри	14 квітня 2023 – 1 травня 2023	Виконано
3	Написання програмного коду	2 травня 2023 – 17 травня 2023	Виконано
4	Візуалізація гри (створення текстур)	19 травня 2023 – 25 травня 2023	Виконано
5	Створення рівнів	26 травня 2023 – 2 червня 2023	Виконано
6	Тестування	3 червня 2023 – 14 червня 2023	Виконано
7	Оформлення пояснювальної записки	15 червня 2023 – 24 червня 2023	Виконано

Здобувач вищої освіти Данило СМУЛЬСЬКИЙ
Керівник Інна СУГОНЯК

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу гри головоломки та пояснювальної записки. У роботі визначено основні завдання на розробку гри.

Метою даної роботи, по темі "Технологіям реалізації n-вимірних кубічних об'єктів в іграх-головоломках", є дослідження та розробка ефективних методів реалізації кубічних об'єктів в іграх-головоломках. Основне зусилля спрямоване на створення ігрової головоломки, заснованої на використанні кубів та розробленої з використанням ігрового рушія Unity. Основним завданням є створення інтерактивного ігрового середовища, де гравець взаємодіє з кубічними об'єктами та вирішує логічні завдання.

Крім того, були проведені тестування гри, а результати тестів були наведені в записці пояснення. Виявлено та усунено декілька багів.

Пояснювальна записка має 5 сторінок додатку.

Список використаних джерел має 6 найменувань і займає 1 сторінку.

В роботі наведено 26 рисунків.

Загальний обсяг роботи складає 65 сторінок.

КЛЮЧОВІ СЛОВА: UNITY, LVL, ГРА, ГОЛОВОЛОМКА, КУБ.

КН.КР.Б. – 122 – 23 - ПЗ

					КН.КР.Б. – 122 – 23 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Технології реалізації n - вимірних кубічних об'єктів в іграх-головоломках Пояснювальна записка	Літ.	Арк.	Аркушів	
Розроб.		Д.З.СМУЛЬСЬКИЙ							
Керівник		І.І. СУГОНЯК					4	48	
Рецензент		О.В.КУЗЬМЕНКО				Житомирська політехніка, група КН-19-1			
Н. Контр.									
Зав.каф.		М.С.ГРАФ							

ABSTRACT

The bachelor's thesis consists of a software complex of a puzzle game and an explanatory note. The thesis identifies the main tasks for game development.

The aim of this work, entitled "Technologies for Implementing n-Dimensional Cubic Objects in Puzzle Games," is to investigate and develop effective methods for implementing cubic objects in puzzle games. The main focus is on creating a puzzle game based on the use of cubes, developed using the Unity game engine. The primary objective is to create an interactive gaming environment where players interact with cubic objects and solve logical challenges.

Additionally, game testing was conducted, and the test results were documented in the explanatory note. Several bugs were identified and fixed.

The explanatory note includes a 5-page appendix.

The reference list contains 6 sources and spans 1 page.

The thesis includes 21 illustrations.

The total volume of the work is 48 pages.

KEYWORDS: Unity, level, game, puzzle, cube.

					<i>КН.КР.Б. – 122 – 23 - ПЗ</i>	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. КОНЦЕПЦІЯ ТА ДИЗАЙН КВАЛІФІКАЦІЙНОЇ РОБОТИ	10
1.1 Концепт документ.....	10
1.2 Вимоги до функціональних характеристик	11
1.3 Обґрунтування вибору інструментальних засобів.....	12
1.4 Аналіз аналогів програмного продукту.....	15
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ	22
2.1 Розробка архітектури ігрового додатку	22
2.2 Діаграма прецедентів використання та діаграма класів	26
2.3 Проектування та реалізація алгоритмів роботи системи	37
Висновки до другого розділу	39
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЙОГО ТЕСТУВАННЯ	40
3.1 Основні етапи реалізації гри	40
3.2 Створення ігрового середовища та інтерфейсу користувача	42
3.3 Опис роботи з програмним додатком та тестування.....	50
Висновки до третього розділу:	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТКИ.....	61

Перелік умовних скорочень

U3D – Unity3D.

LVL – Level

ПР – Початкова позиція

РВ – Рух вгору

РН – Рух вниз

РП – Рух вправо

РВ – Рух вліво

ПЗ – Початкова позиція

ФІШ – Фініш

VS – Visual Studio

					<i>КН.КР.Б. – 122 – 23 - ПЗ</i>	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Уявіть захоплюючу гру, яка захоплює ваш розум і спонукає вас до використання логічного мислення та стратегічного планування. Гра, яку ми пропонуємо, це головоломка - захоплюючий світ, де ви будете використовувати свої розумові можливості для вирішення складних завдань.

У цій грі кожен куб має свою власну колірну палітру, а ваше завдання головним кубом дібратися до кінця рівня, щоб перейти на наступний.

Ви будете вдосконалювати свої навички вирішення проблеми, розвивати логічне мислення та покращувати свою здатність до абстрактного мислення.

Гра головоломка також пропонує вам можливість розважитися та розслабитися. Вона займається вашим розумом, спонукаючи до концентрації і відмовляючись від повсюдних турбот. Ви можете використовувати її як засіб для розумового відпочинку, занурившись у світ викликів та творчості.

Головоломка - це не просто гра, це спосіб розвинути свої розумові здібності та навчитися новим стратегіям розв'язання проблеми. Вона тренує ваше терпіння, стимулює творче мислення та вчить вас знаходити незвичайні шляхи вирішення завдань.

Гра стане твоїм надійним партнером у розвитку розуму та вивченні нових стратегій. Вона запропонує тобі відпочинок від повсякденних справ, де ти зможеш розслабитися, а водночас активувати свій розум. Ти зануришся у захоплюючий світ гри, де кожне завдання принесе тобі задоволення від досягнення успіху.

Метою випускної роботи є проектування та розробка ігрової головоломки з кубами на рушії U3D. Головна мета полягає в створенні захоплюючого геймплею, який вимагатиме від гравця логічного мислення, стратегічного планування та розв'язання складних головоломок. У процесі

					<i>КН.КР.Б. – 122 – 23 - ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

розробки гри, будуть створені різноманітні рівні складності з унікальними головоломками, де гравець повинен буде маніпулювати кубами та знаходити оптимальні шляхи для їх переміщення, щоб основним кубом дібратися до кінця рівня та перейти на наступний. Кожен рівень буде вимагати розумових зусиль і творчого підходу до пошуку рішень.

Метою такого проекту є створення цікавої та захоплюючої гри, яка надихне гравців на розвиток логічного мислення, просторової уяви та креативного підходу до вирішення проблем. Результатом роботи буде повноцінний програмний продукт, який буде доступний для гравців і надихатиме їх на розвиток розумових навичок.

Ця випускна робота спрямована на поєднання технічних навичок у розробці ігрових модулів зі змістовим аспектом гри головоломки, що розвиватиме розумові здібності гравців.

Об'єктом дослідження є гра головоломка з використанням кубів.

Предметом дослідження є використання рушія U3D, для розробки гри головоломки з використанням кубів.

Розроблена гра може бути використана для розваги та відпочинку, розвитку креативності та творчості.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. КОНЦЕПЦІЯ ТА ДИЗАЙН КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Концепт документ

Концепт документ — це документ, який містить опис та концепцію для розробки проекту. Він є основою для розуміння та комунікації між розробниками, дизайнерами, виконавцями та іншими учасниками процесу.

Концепт документ детально описує цілі, ідеї, основні механіки, функціональність, характеристики та загальну концепцію проекту. Він є своєрідним планом, який визначає напрям та основні принципи роботи над проектом.

Назва гри: Labyrinths of the Mind.

Жанр: Логічна гра, головоломка.

Цільова аудиторія: любителі головоломок та логічних завдань.

Ціль гри: Головним кубом з початкової позиції (ПЗ) дістатися до фінішу, тобто завершити рівень та перейти на наступний.

Механіка гри: Гравець рухає двома видами кубів лише вгору (РВ) та вниз (РН), також присутні куби, які рухаються вліво (РВ) та вправо (РП). Також присутні перешкоди, які не рухаються, та заважають гравцю дістатися до фінішу. За допомогою вищевказаних кубів гравцю потрібно пройти рівень.

Умови перемоги: Гравець перемагає, якщо основним кубом дістається до фінішу.

Основні особливості гри: Легкий для вивчення геймплей, інтуїтивно зрозуміле меню. Привабливий дизайн ігрового поля, та кубів.

Інтерфейс користувача:

Головне меню: Меню з кнопками для навігації.

Нова гра: Розпочати нову гру.

Список рівнів: Можливість перейти на будь-який рівень.

Пауза: Можливість поставити гру на паузу.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Вихід: Вийти з гри.

Ігрове поле: Поле розташоване на екрані. Залежно від рівня по полю розкидані різноманітні куби, за допомогою яких гравець може проходити рівень.

Екран перемоги: Після проходження останнього рівня відображається екран, на якому гравця вітають з проходженням гри, після чого його направляють у головне меню, де він може обрати будь-який рівень та спробувати знайти інше рішення для нього.

Геймплей та логіка: Гравець може здійснити хід вгору, вниз, вліво, або вправо. При здійсненні ходу всі куби на рівні (окрім перешкоди) рухаються у вибраному напрямку.

Платформи: Персональні комп'ютери (PC)

1.2 Вимоги до функціональних характеристик

У цьому підрозділі будуть розглянуті вимоги до функціональних характеристик нашої головоломки з кубами. Головоломка з кубами є інтерактивною грою, в якій гравець взаємодіє з кубічними об'єктами та вирішує різноманітні логічні завдання.

Вимоги до функціональних характеристик визначають необхідні можливості та функції, які мають бути реалізовані у продукті чи системі. Вони описують те, що продукт має виконувати. Це визначає основні функції та операції, які мають бути доступні у продукті.

Вони допомагають визначити основні функції та можливості, які мають бути реалізовані у грі головоломці, щоб задовольнити потреби та очікування користувачів, що будуть користуватися проектом.

В ході виконання кваліфікаційної роботи потрібно розробити головоломку, яка включає в себе наступне:

1. Запуск самої головоломки;
2. Головне меню
3. Можливість почати нову гру

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Можливість переглянути список рівнів
5. Можливість вийти з гри.
6. Поставити гру на паузу
7. Можливість вийти в головне меню

Головоломки з кубами мають потенціал стати найпопулярнішим жанром серед гравців. Вони сприяють розвитку мозкової активності, логічного мислення, уваги та просторової орієнтації. Такі ігри можуть бути цікаві не тільки для дітей, але й для дорослих, які шукають виклики та розваги для свого розуму.

Гра такого жанру відкриває унікальні можливості для розвитку мозкових навичок та когнітивних здібностей. Граючи в такі ігри, гравці вдосконалюють своє логічне мислення, просторову уяву та аналітичні здібності. Вирішення складних головоломок з кубами вимагає від гравця концентрації уваги, терпіння, стратегічного планування та вирішення проблем.

Ці ігри є не тільки чудовим способом розваги, але й ефективним інструментом для навчання. Вони можуть використовуватися в освітніх програмах для розвитку творчого та критичного мислення, вирішення проблем, співробітництва та комунікації. Вони є чудовим варіантом для тренування розумових навичок у школах, університетах та навчальних центрах.

Крім того, головоломки з кубами можуть стати цікавою формою розваг для всієї родини. Вони сприяють спільному проведенню часу, взаємодії та співпраці між батьками та дітьми. Вирішення головоломок разом стимулює командну роботу, розвиває навички обміну ідей та стратегій, а також підтримує сімейні зв'язки.

Захоплюючі, зухвалі та ментально стимулюючі, головоломки привертають увагу широкого кола гравців. Інтерес до цих ігор постійно зростає, а їхній потенціал ще далеко не вичерпаний. Розробка нових та

оригінальних головоломок з кубами відкриває безмежні можливості для розвитку цього жанру та задоволення потреб сучасних гравців.

Головоломки пропонують нескінченну різноманітність комбінацій та рішень, що робить їх захоплюючими. Гравці можуть насолоджуватися процесом розгадування складних завдань та досягненням нових рівнів. Крім того, головоломки сприяють розвитку творчого мислення, стратегічного планування та побудови логічних ланцюжків.

Однією з особливостей головоломок є їхня доступність і портативність. Вони можуть бути грою, яку можна взяти з собою у подорож, грати у будь-який час та у будь-якому місці. Завдяки цьому вони стають чудовим способом провести час і розважитися в очікуванні, в дорозі або в перервах між справами.

Однією з головних вимог є візуальне подання кубічних об'єктів. Гра повинна забезпечувати високоякісне та реалістичне відображення кубів, використовуючи 3D-моделі, текстури та різноманітні візуальні ефекти. Це допоможе створити привабливу графіку для користувачів.

Іншою важливою вимогою є інтерактивність гри. Гравець повинен мати можливість взаємодіяти із кубами. Це вимагає розробки зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє гравцеві легко керувати кубами та виконувати необхідні дії.

Рівні складності є ще однією важливою характеристикою. Головоломка повинна збільшувати рівні складності, починаючи з простих завдань для новачків і складних головоломок для досвідчених гравців. Це дозволить гравцям відчувати прогрес та виклик у грі, забезпечуючи тривалий інтерес.

Логічні завдання є основою головоломки із кубами. Гра повинна пропонувати гравцям різноманітні логічні завдання, які вимагатимуть аналітичного мислення та вирішення проблем.

Графічний інтерфейс користувача також має важливу роль. Гра повинна мати зручний та привабливий графічний інтерфейс, який дозволяє гравцеві легко переходити, взаємодіяти з елементами гри та отримувати необхідну інформацію. Інтерфейс має бути зрозумілим та інтуїтивно зрозумілим, спрощуючи взаємодію гравця з головоломкою.

Гра-головоломка з кубами є інтерактивною програмою, розробленою з використанням ігрового двигуна U3D. Мета гри полягає в тому, щоб запропонувати гравцям захоплюючі та логічні завдання, що базуються на використанні кубічних об'єктів. Вона є інноваційним ігровим додатком, який приносить захоплюючий досвід та виклик для любителів головоломок та пазлів. Розроблено з використанням передових технологій та принципів дизайну ігор.

Основна мета гри полягає в тому, щоб захоплююче провести гравця через серію рівнів, у кожному з яких вирішити складні завдання, використовуючи кубічні об'єкти. Гравець має проявити свою логічну думку, креативність та стратегічне мислення для досягнення успіху.

У грі представлені різні типи кубів з різними властивостями та функціями. Куби можуть рухатися певними маршрутами. Гравець повинен вміло використовувати ці особливості, щоб подолати перешкоди та вирішити головоломки.

У грі реалізовано кілька рівнів складності. Починаючи з простих головоломок, гравці поступово переходять до складніших завдань, які потребують ретельного планування та стратегії.

Один з основних аспектів гри – це фізична модель, яка відображає реалістичні рухи та взаємодії кубів. Куби можуть бути переміщені та з'єднані один з одним, що дозволяє гравцям створювати різноманітні конфігурації та вирішувати головоломки.

Графічний компонент гри також заслуговує на особливу увагу. Візуальне представлення кубів та ігрового середовища виконано з високою якістю та деталізацією.

Інтерфейс гри інтуїтивно зрозумілий та забезпечує зручне управління. Гравці можуть взаємодіяти з кубами за допомогою клавіш на клавіатурі. Також є додаткові функції, такі як підказки.

Важливим аспектом гри є система рівнів та прогресії. Гравці можуть проходити рівні послідовно, відкриваючи нові головоломки та виклики у міру свого прогресу. Або вибрати рівень, що їх цікавить, через відповідне меню.

Насамкінець, гра-головоломка з кубами пропонує захоплюючу та інтерактивну ігрову пригоду, яка поєднує в собі логічні завдання, фізичну модель та привабливу графіку. Унікальна комбінація різноманітних рівнів складності та інтуїтивного інтерфейсу користувача робить гру привабливою і доступною для любителів головоломок різних вікових груп. Завдяки своєму потенціалу для розвитку мозкової активності, логічного мислення та просторової орієнтації головоломки з кубами можуть стати популярним жанром серед гравців. Така гра пропонує не тільки дітям, але й дорослим захоплюючу розвагу та виклик для розуму.

1.3 Обґрунтування вибору інструментальних засобів

U3D [1] є одним з найпопулярніших і найпотужніших ігрових рушіїв на ринку, зазначимо його переваги:

– Кросплатформенність: U3D дозволяє розробляти ігри та програми, які можуть працювати на різних платформах, таких як Windows, macOS, iOS, Android, Xbox, PlayStation та багатьох інших. Це значно спрощує портування та розповсюдження проєктів.

– Великі можливості: U3D пропонує багатий інструментарій та функціональність для розробки ігор. Включає графічний рушій, фізичний рушій, підтримку анімації, аудіо, штучний інтелект, мережева взаємодія,

редактор сцен і багато іншого. Це дозволяє розробникам створювати ігри різних жанрів та складності.

– Візуальний редактор U3D пропонує зручний візуальний редактор, який дозволяє розробляти ігрові сцени, компоненти, анімації та інтерфейси без необхідності писати велику кількість коду. Це особливо корисно для художників та дизайнерів, які можуть легко вносити зміни та тестувати різні ідеї.

– Велика спільнота та ресурси: U3D має величезну спільноту розробників, які активно обмінюються досвідом, знаннями та ресурсами. Це дозволяє отримувати підтримку, знаходити відповіді на питання, вивчати нові техніки та використовувати готові рішення, що суттєво прискорює розробку та покращує якість проектів.

– Безліч плагінів та активів: U3D підтримує розширення функціональності за допомогою плагінів та активів, які надають різні готові рішення, інструменти та ефекти. Це дозволяє заощадити час та зусилля при розробці, а також додати якісні ефекти та функціональність у проекти.

– Легкість вивчення: U3D має відносно просту мову програмування (C#) та інтуїтивно зрозумілий інтерфейс, що робить його доступним для новачків. Є безліч навчальних матеріалів, документації та онлайн-курсів, які допоможуть розробникам-початківцям швидко освоїти основи і приступити до створення своїх проектів.

– Інтеграція з іншими інструментами: U3D легко інтегрується з іншими інструментами та програмним забезпеченням. Наприклад, можна використовувати спеціалізовані програми для створення моделей, анімацій чи звукових ефектів, а потім імпортувати їх у U3D для подальшої роботи.

– Підтримка віртуальної та доповненої реальності: U3D надає потужні інструменти для розробки віртуальної та доповненої реальності. Це відкриває двері для створення захоплюючих та інноваційних ігрових та візуальних проектів, які використовують останні технології.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

– Мультиплатформна розробка: U3D дозволяє розробляти ігри та програми, які можуть запускатися на різних платформах з мінімальними змінами. Це значно спрощує процес портування проектів різні пристрої, оскільки більшість коду і ресурсів можна перевикористовувати. Це економить час та ресурси розробників.

– Можливість створення комерційно успішних проектів U3D має доведену історію успішних комерційних проектів. Багато відомих ігор, включаючи хіти на мобільних платформах, комп'ютерах та консолях, були розроблені з використанням U3D. Двигун надає інструменти та можливості для створення проектів високої якості, які можуть залучити широку аудиторію та забезпечити комерційний успіх.

Visual Studio (VS) – це інтегроване середовище розробки, розроблене компанією Microsoft. Вона надає розробникам потужні інструменти та функціональності для створення різноманітних програм, включаючи веб-додатки, настільні програми, мобільні програми та ігри [5]. Ось деякі характеристики середовища розробки Visual Studio:

– Потужний редактор коду: VS надає високофункціональний редактор коду з функціями автозавершення, підсвічуванням синтаксису, швидким переходом до визначення, рефакторинг коду та іншими можливостями. Це дозволяє розробникам ефективно писати та редагувати код, підвищуючи продуктивність та прискорюючи розробку.

– Налагодження та профілювання: VS має потужні інструменти налагодження, що дозволяють розробникам знаходити та виправляти помилки в коді. Він підтримує покрокове виконання, точки зупинки, перегляд значень змінних та інші функції налагодження. Крім того, VS надає можливість профілювання програм для оптимізації продуктивності та виявлення вузьких місць.

– Керування проектом та версіонування: VS пропонує інструменти для управління проектами, включаючи системи контролю

версій, такі як Git та Team Foundation Version Control (TFVC). Це дозволяє розробникам ефективно співпрацювати над проектами, відстежувати зміни, керувати гілками коду та виконувати злиття змін.

– Інтеграція з платформами та сервісами: VS інтегрується з різними платформами та сервісами, полегшуючи розробку програм. Це включає інтеграцію з хмарними платформами Microsoft Azure, інструментами розробки мобільних програм Xamarin, веб-фреймворком ASP.NET та іншими сервісами Microsoft.

– Розширюваність: VS має високий рівень розширюваності. Розробники можуть використовувати безліч плагінів і розширень, створених спільнотою розробників або Microsoft, щоб додати додаткові функціональні можливості та інтегрувати інструменти сторонніх виробників.

1.4 Аналіз аналогів програмного продукту

Q.U.B.E. (Quick Understanding of Block Extrusion) - головоломка від першої особи, розроблена та видана компанією Toxic Games. випущена у січні 2012 року. Вона доступна на різних платформах, включаючи Microsoft Windows, MacOS, Linux, PlayStation 3, PlayStation 4, Xbox One, Wii U, iOS та Android. Геймплей гри полягає у вирішенні головоломок з використанням спеціальних рукавичок, які дозволяють взаємодіяти з кубами та змінювати їх властивості. Гравець має долати перешкоди, активувати механізми та досягати фінішу рівня, використовуючи куби різних кольорів та властивостей.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

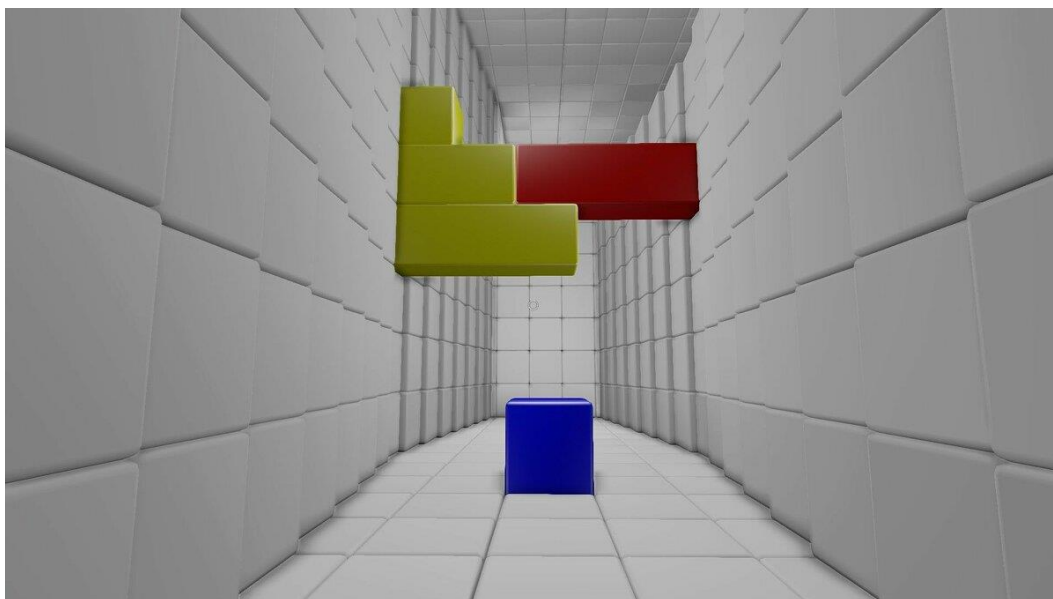


Рис.1.1. Q.U.B.E.

Cubiques - головоломка, розроблена та видана компанією Dilmer Valecillos. Гра була випущена у листопаді 2016 року. Вона доступна на різних платформах, включаючи iOS, Android, Windows, macOS та Linux. Геймплей Cubiques заснований на вирішенні тривимірних головоломок, де гравець має керувати основним кубом, переміщуючи його за рівнем і взаємодіючи з іншими кубами, щоб досягти мети та перейти на наступний рівень. Гра пропонує різноманітні головоломки та викликають логічні завдання, а також естетично приємний мінімалістичний дизайн.

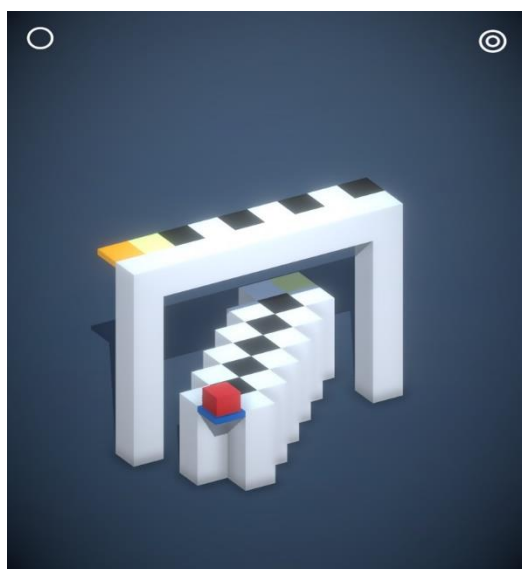


Рис.1.2. Cubiques

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Cubot: The Complexity of Simplicity - головоломка, розроблена та видана компанією Nicolas Pierre-Loti-Viaud. Гра була випущена у жовтні 2014 року. Вона доступна на різних платформах, включаючи iOS, Android, Windows, macOS та Linux. Геймплей Cubot: The Complexity of Simplicity заснований на вирішенні тривимірних головоломок, де гравець має управляти маленьким роботом Cubot, переміщуючи його за рівнем і знаходячи оптимальний шлях до виходу. Мета гри – перемістити Cubot на кожному рівні до виходу, використовуючи доступні переміщення та повороти. Гра пропонує різноманітні рівні складності та завдання, а також стильний графічний дизайн та звукове оформлення.

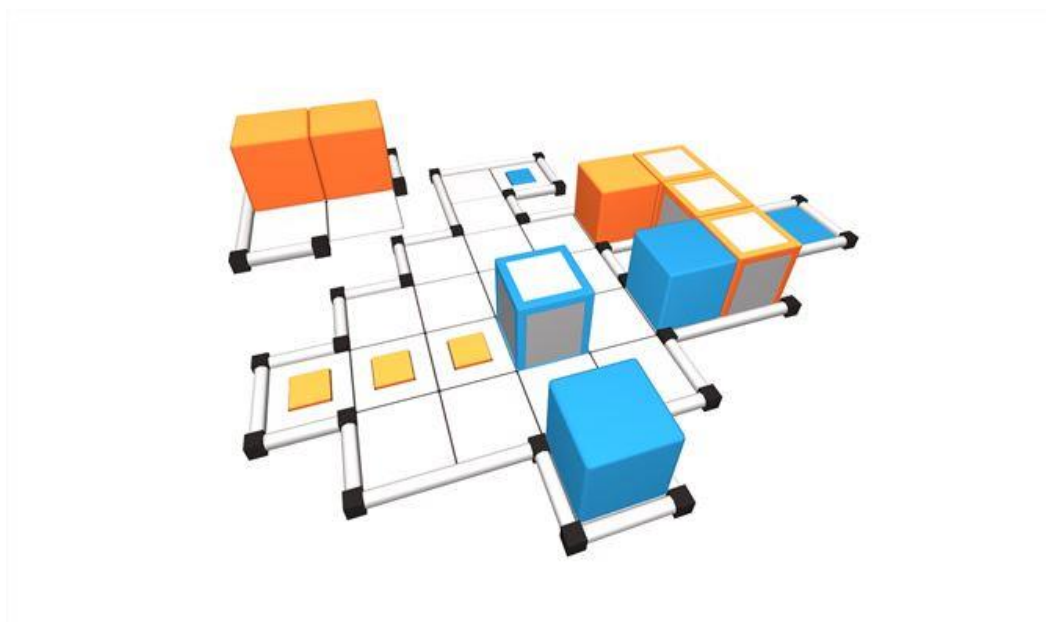


Рис.1.3.Cubot: The Complexity of Simplicity

Проаналізувавши ігри: Q.U.B.E., Cubiques і Cubot: The Complexity of Simplicity, було виділено те, що основна увага ігор припадає на взаємодію з кубами, оскільки це ключовий елемент кваліфікаційної роботи. У кожній грі з вищеписаних куби відіграють головну роль у вирішенні головоломок на проходження рівнів.

Під час розробки гри, головною метою було створення захоплюючих головоломок, які вимагатимуть від гравця логічного мислення та

планування. Приділено увагу створенню інтуїтивно зрозумілому інтерфейсу, як самих рівнів, так і головного меню.

Висновки до першого розділу

У цьому розділі було сформовано концепцію гри, та дизайн кваліфікаційного проекту.

У першій частині були описані основні функції та цілі продукту, що розробляється, а також його потенційна аудиторія та переваги.

У другій частині були проаналізовані аналоги програмного продукту, такі як Q.U.B.E., Cubiques та Cubot: The Complexity of Simplicity. Було зазначено, що у всіх цих іграх основна увага приділяється взаємодії з кубами та вирішенню головоломок. Однак цільовий продукт матиме свої унікальні особливості та інноваційні елементи.

У третій частині було підбито підсумки аналізу. Було наголошено, що вибрані ігри пропонують цікаві головоломки та викликають логічні завдання. Однак цільовий продукт матиме свої унікальні особливості, які дозволять йому виділитися на тлі аналогів.

Таким чином, аналіз дозволив визначити напрямок розробки програмного продукту, а також виявити його конкурентні переваги. Цільовий продукт пропонуватиме захоплюючі головоломки, що вимагають логічного мислення, з інтуїтивно зрозумілим інтерфейсом та унікальними особливостями, які зроблять його привабливим для цільової аудиторії.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

2.1 Розробка архітектури ігрового додатку

Розробка архітектури ігрового додатку - це процес створення структури та організації компонентів, систем та взаємодії між ними в рамках ігрового застосування. Архітектура ігрового застосування визначає його основні компоненти, модулі, логіку і спосіб взаємодії між ними.

Мета розробки архітектури ігрового додатку полягає у створенні ефективної та масштабованої структури, яка забезпечує зручність розробки, підтримки та розширення ігрового додатку. Добре спроектована архітектура допомагає знизити складність розробки, полегшити супровід коду, покращити модульність та повторне використання компонентів.

Розробка архітектури ігрового додатку також допомагає визначити основні функціональні блоки програми, включаючи графічний інтерфейс, ігрову логіку, фізичну симуляцію, управління ресурсами, взаємодію з користувачем та інші аспекти. Вона дозволяє розробникам мати загальне уявлення про структуру програми, спрощує командну роботу та покращує розуміння системи загалом.

У результаті, розробка архітектури ігрового додатку допомагає забезпечити його якісне функціонування, ефективність, надійність і гнучкість. Добре спроектована архітектура ігрового застосування відіграє важливу роль у досягненні цілей проекту та створенні задовільного ігрового досвіду для користувачів.

Розробка архітектури ігрового додатка гри «Labyrinths of the Mind» включає кілька ключових компонентів. Ось загальна структура та основні елементи, які слід врахувати при розробці такої програми:

Ігрове поле:

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Ігрове поле є основним середовищем, у якому розгортається ігровий процес. Воно є площиною з розмірами 1.5, 1, 1.5. Щоб створити візуальне враження, на ігрове поле накладаються текстури, що надають йому унікального вигляду.

Стіни ігрового поля:

Стіни ігрового поля служать для обмеження руху кубів та запобігання їх виходу за межі рівня. Складаються вони із кубів, які щільно розташовані навколо ігрового поля.

Фініш:

Фініш – це мета гравця, яку він має досягти, використовуючи куб із тегом "Player". Фініш знаходиться у певному місці на ігровому полі та є ключовим елементом для успішного завершення рівня.

Типи кубів:

У грі присутні різні типи кубів, кожен з яких має свої особливості та рух:

Головний куб: Цей куб може рухатися вгору та вниз. Гравцю необхідно керувати головним кубом, щоб подолати перепони та дістатися фінішу.

Другий куб: Цей куб рухається вліво і вправо. Він є додатковим інструментом для гравця, допомагаючи йому навігувати та долати складності на рівні.

Допоміжний куб: Допоміжний куб може рухатися вгору та вниз. Він надає додаткову підтримку гравцеві, допомагаючи йому подолати певні перешкоди чи досягти певних місць.

Перешкода: Перешкоди є нерухомими кубами, які гравець повинен враховувати при плануванні свого шляху до фінішу. Гравець повинен обходити перешкоди та знаходити шляхи навколо них.

Меню паузи:

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

У грі є функція паузи, що дозволяє гравцеві призупинити ігровий процес у будь-який момент. Паузу можна активувати, натиснувши клавішу "P". Щоб відновити гру, гравець також має натиснути клавішу "P".

Зміна рівня:

Гра пропонує можливість змінити поточний рівень. Якщо гравець не зацікавлений у поточному рівні, він може перейти до наступного, натиснувши кнопку "N". За винятком останнього рівня, гравець може перемикатися між рівнями, щоб випробувати різні виклики та перемогти на кожному з них. Кнопка N активна тільки на проміжних рівнях, а на 10-му рівні вона недоступна, так як це останній рівень гри.

Початок рівня заново:

Гравець має можливість почати рівень знову, натиснувши кнопку "R". Це корисно, якщо гравець зіткнувся з труднощами або зробив помилку і хоче спробувати ще раз, не повертаючись до головного меню.

Вихід у головне меню:

Якщо гравець хоче вийти з поточного рівня чи паузи та повернутися до головного меню, він може натиснути клавішу Esc. При натисканні Esc перед гравцем з'являється сцена із запитом підтвердження виходу в головне меню. Гравець може вибрати «Так», щоб вийти в головне меню, або «Ні», щоб продовжити грати на поточному рівні.

Вітальне вікно:

Після успішного проходження 10-го рівня гравцеві відображається вітальне вікно, де він отримує похвалу за свої досягнення та успішне завершення гри. Після цього гравець повертається до головного меню, де може почати нову гру або вибрати інші доступні опції.

Головне меню:

Головне меню являє собою стартову точку для гравця та надає кілька варіантів дій:

					<i>КН.КР.Б. – 122 – 23 - ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

"Нова гра": При виборі цієї опції гравець переходить на перший рівень і починає гру з самого початку.

"Список рівнів": При виборі цієї опції гравець переходить на сцену зі списком рівнів. На цій сцені відображається 10 різних фотографій із компонентом "Button" для кожного рівня. При натисканні на кнопку відповідного рівня гравець переноситься на цей рівень та починає гру з вибраного моменту.

"Вихід": При виборі цієї опції гравець виходить із гри.

Список рівнів також містить кнопку "Назад", яка дозволяє гравцеві повернутися зі списку рівнів до головного меню, якщо він захоче змінити своє рішення або вибрати іншу опцію.

Таким чином, розробка архітектури ігрового додатку включає ігрове поле, стіни, різні типи кубів з текстурями та певними рухами, фініш, меню паузи, зміну рівнів, можливість розпочати рівень заново, вихід у головне меню з підтвердженням, вітальне вікно та головне меню з опціями "Нова гра", "Список рівнів" та "Вихід". Ці елементи створюють цікавий та захоплюючий ігровий досвід для гравця.

Проаналізувавши архітектуру ігрового додатку створимо діаграму розгортання (Deployment Diagram).

Діаграма розгортання в UML використовується для моделювання апаратного та програмного забезпечення, яке складається з корпусу та корпусу із закритими компонентами фізичних пристроїв (рис.2.1).

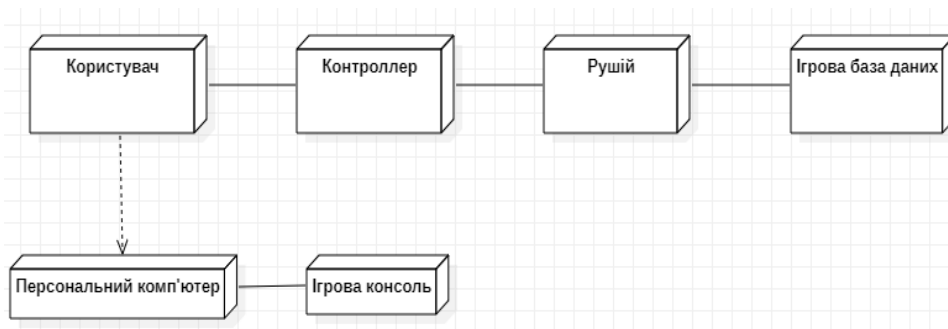


Рис.2.1. Діаграма розгортання (Deployment Diagram)

Створивши діаграму, опишемо її:

Користувач: Це вузол, який представляє гравця або користувача гри. Він взаємодіє з ігровим процесом через контролер.

Ігровий ПК: Це вузол, що представляє ігровий комп'ютер, на якому гра запускається. Він відповідає за розгортання та забезпечення функціонування ігрового ПЗ.

Контролер: Це компонент, який обробляє введення від користувача і передає їх в ігровий двигун. Він пов'язаний з користувачем та ігровим ПК.

Ігрова Консоль: Це вузол, що представляє ігрову консоль, яка може бути використана як альтернативний пристрій для гри. Вона відповідає за відображення графіки гри. Пов'язана з персональним комп'ютером.

Рушій: Це компонент, який виконує ігрову логіку, включаючи фізику, штучний інтелект та керування ігровими об'єктами. Він пов'язаний з контролером та ігровою базою даних.

Ігрова база даних: Це компонент, який зберігає ігрову інформацію, таку як рівні, досягнення, збережені дані та іншу ігрову статистику. Вона пов'язана з рушієм.

2.2 Діаграма прецедентів використання та діаграма класів

Use Case Diagram (діаграма прецедентів використання) - це тип діаграми в мові моделювання UML (Unified Modeling Language), який є графічним уявленням взаємодії між акторами (користувачами або зовнішніми системами) і системою, ілюструючи функціональні можливості, які надає система[7].

Основна мета використання діаграми прецедентів використання полягає у моделюванні та візуалізації взаємодій між користувачами та системою, ідентифікації функціональних вимог та основних сценаріїв використання системи. Дана діаграма є інструментом для аналізу та розуміння функціональності системи з погляду користувачів.

Переваги використання діаграми прецедентів використання включають:

Розуміння вимог користувачів: Діаграма прецедентів використання допомагає розробникам та аналітикам зрозуміти вимоги та очікування користувачів, а також визначити основні функції, які має надавати система.

Візуалізація взаємодії: Діаграма прецедентів використання надає графічне уявлення взаємодії між акторами та системою, що допомагає краще зрозуміти процеси та потоки інформації в системі.

Визначення меж системи: Діаграма прецедентів використання допомагає визначити межі системи, виявити взаємодію із зовнішніми акторами та системами, а також ідентифікувати основні інтерфейси та точки взаємодії.

Ідентифікація основних сценаріїв використання: Діаграма прецедентів використання дозволяє ідентифікувати та описати основні сценарії використання системи, що допомагає краще зрозуміти потоки дій та процеси, що виконуються в системі.

Визначення пріоритетів: Діаграма прецедентів використання може використовуватись для визначення пріоритетів функціональності системи, ідентифікації основних та найважливіших прецедентів використання.

Загалом використання діаграми прецедентів використання дозволяє розробникам та аналітикам краще зрозуміти потреби користувачів, визначити функціональні вимоги та специфікації системи, а також полегшує комунікацію та узгодження між учасниками проекту.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

Створимо Use Case Diagram для нашої гри (рис.2.2). Та наведемо опис самої діаграми.

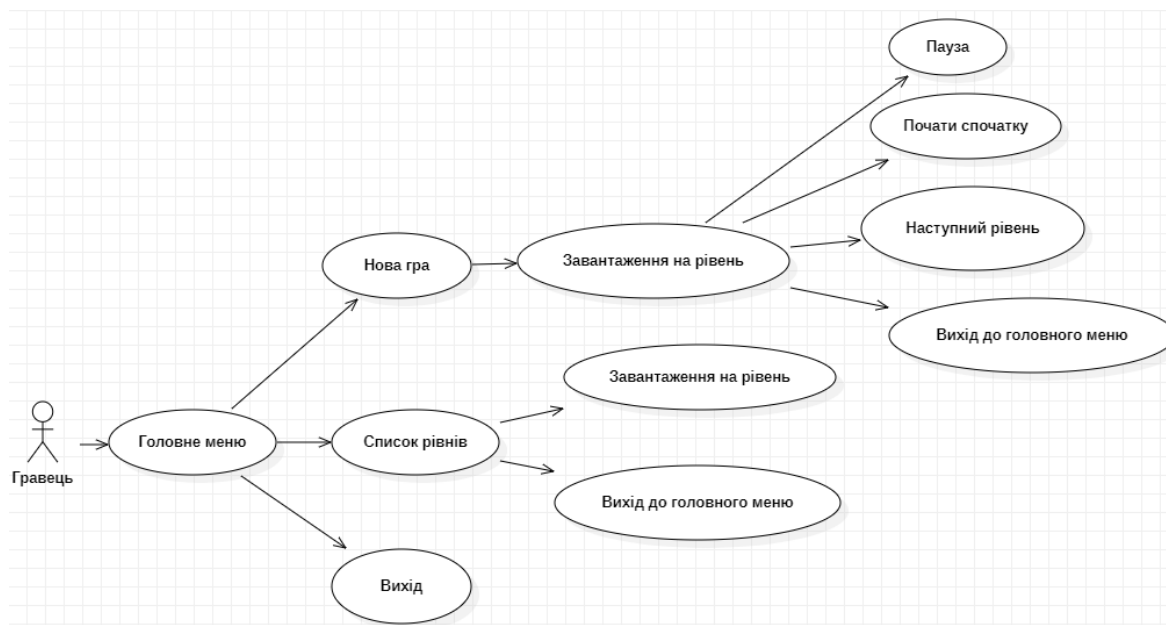


Рис.2.2. Use Case Diagram

Гравець є актором, який взаємодіє із ігровим додатком. Головне меню є одним з основних функціональних елементів гри та надає три варіанти дій гравцю.

Нова гра: Цей варіант дозволяє гравцеві розпочати нову гру. Після вибору цієї опції відбувається завантаження на рівень, де гравцеві надаються додаткові варіанти дій.

Пауза: Гравець може поставити гру на паузу, щоб тимчасово призупинити проходження рівня.

Почати спочатку: Гравець може почати рівень знову, скинувши прогрес і повернувшись у початковий стан.

Перейти на наступний рівень: Після успішного завершення поточного рівня гравець може перейти на наступний рівень.

Повернутися до головного меню: Гравець може повернутися до головного меню, залишаючи поточний рівень.

Список рівнів: Ця опція дозволяє гравцеві переглянути список доступних рівнів та вибрати один із них для завантаження.

Завантаження на рівень: Гравець може вибрати рівень зі списку та завантажити його для початку гри.

Вихід у головне меню: Гравець може повернутися до головного меню, скасовуючи вибір рівня.

Вихід: Цей варіант дозволяє гравцеві вийти з програми.

Вся ця послідовність взаємодій між гравцем та ігровим програмою представлена на діаграмі Use Case, яка допомагає зрозуміти основні функціональні можливості програми та послідовність дій, доступних для гравця.

Діаграма класів є однією з ключових діаграм в об'єктно-орієнтованому програмуванні та використовується для візуалізації структури класів у системі чи програмі. Вона дозволяє уявити класи, їх атрибути та методи, а також зв'язки між класами.

Основна мета діаграми класів - описати структуру класів та їх взаємозв'язку у системі, що забезпечує розуміння архітектури та дизайну програми.

Виокремимо декілька переваг використання діаграми класів:

Візуалізація структури класів: Діаграма класів дозволяє легко уявити класи та їх взаємодію у системі. Вона дозволяє розробникам виявляти зв'язки між класами, ідентифікувати атрибути та методи кожного класу.

Розуміння зв'язків: Діаграма класів дозволяє зрозуміти зв'язки та залежність між класами. Це допомагає розробникам краще організувати код та забезпечити чисту архітектуру.

Проектування та моделювання: Діаграма класів використовується на етапі проектування системи, щоб визначити класи, їх атрибути та методи, а також їхню взаємодію. Вона є основою створення структури коду і допомагає розробникам зрозуміти, як різні компоненти системи взаємодіють друг з одним.

Комунікація та документація: Діаграма класів є ефективним засобом комунікації між розробниками, архітекторами та іншими учасниками проекту. Вона дозволяє легко обговорювати та візуалізувати структуру класів, що допомагає покращити розуміння системи та забезпечити єдине уявлення.

Аналіз та модифікація: Діаграма класів може використовуватись для аналізу та модифікації існуючої системи. Вона дозволяє ідентифікувати залежності між класами, виявити слабкі місця та потенційні проблеми в архітектурі, а також планувати зміни у системі.

Загалом діаграма класів є потужним інструментом для аналізу, проектування та візуалізації об'єктно-орієнтованих систем. Вона допомагає розробникам зрозуміти структуру та взаємозв'язки класів, що сприяє розробці чистого та гнучкого коду.

Ознайомившись із загальними поняттями діаграми класів, та основними її перевагами, створимо діаграму класів нашої гри (рис 2.3)

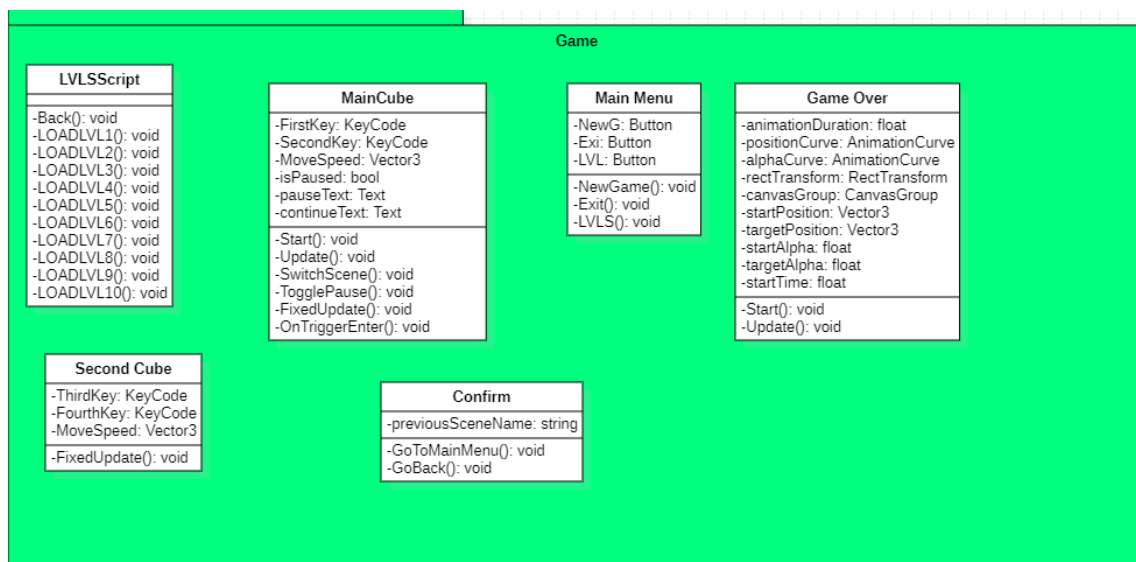


Рис.2.3. Створення діаграми класів.

Створивши саму діаграму класів потрібно описати. Детальний опис діаграми корисний по декільком причинам:

Забезпечення розуміння: Детальний опис діаграми допомагає розробникам та іншим зацікавленим особам повністю зрозуміти, як

працює система чи програмне забезпечення. Воно розкриває деталі кожного класу, його полів та методів, а також зв'язки між ними.

Комунікація: Детальний опис діаграми дозволяє комунікувати та обмінюватися інформацією про систему з іншими членами команди розробки або заінтересованими сторонами. Воно може бути допоміжним засобом під час обговорення та розуміння кодової бази.

Документація: Детальний опис діаграми може бути важливою частиною документації проекту. Воно допомагає задокументувати функціональність, взаємодію та особливості кожного класу, що полегшує розуміння та підтримку системи у майбутньому.

Аналіз та налагодження: При виникненні проблем або помилок у коді детальний опис діаграми може бути корисним інструментом для аналізу та налагодження. Воно дозволяє швидко визначити місце виникнення проблеми та зрозуміти, які компоненти системи можуть бути порушені.

Рефакторинг та модифікація: При необхідності внести зміни до системи або провести рефакторинг коду, детальний опис діаграми може бути допоміжним посібником. Воно допомагає зрозуміти вплив змін на інші компоненти системи та оцінити можливі наслідки.

В цілому, детальний опис діаграми є важливим інструментом для розуміння, комунікації та документування системи, а також для аналізу, налагодження та модифікації кодової бази. Воно допомагає розробникам та зацікавленим сторонам отримати повне уявлення про систему та її компоненти.

Перейдемо до самого опису діаграми:

Клас MainCube представляє головний та другорядні куби у грі. Він відповідає за рух головного куба та другорядного, управління паузою та перемикання рівнів.

Поля:

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

FirstKey: KeyCode: Клавiша, яка спричиняє позитивний рух куба.

SecondKey: KeyCode: Клавiша, яка спричиняє негативний рух куба.

MoveSpeed: Vector3: Вектор швидкостi руху куба.

isPaused: bool: Прапор, який вказує, чи гра на паузі.

pauseText: Text: Текстовий об'єкт, що відображає повідомлення про паузу.

continueText: Text: Текстовий об'єкт, що відображає повідомлення про продовження гри.

Методи:

Start(): Виконується при запуску гри та ініціалізує початкові значення.

Update(): Викликається кожен кадр та обробляє введення користувача для керування грою та паузою.

SwitchScene(): Завантажує сцену Confirm і зберігає попередню сцену.

TogglePause(): Перемикає режим паузи у грі.

FixedUpdate(): Викликається з постійною частотою та обробляє рух куба на основі натиснутих клавiш.

OnTriggerEnter(Collider other): Викликається при взаємодії з іншими колайдерами та обробляє перехід на наступний рівень при зіткненні з об'єктом Finish.

Клас SecondCube представляє допоміжний куб у грі. Він відповідає за рух другого куба при натисканні певних кнопок.

Поля:

ThirdKey: KeyCode: Клавiша, яка спричиняє позитивний рух куба.

FourthKey: KeyCode: Клавiша, яка спричиняє негативний рух куба.

MoveSpeed: Vector3: Вектор швидкостi руху куба.

Методи:

FixedUpdate(): Викликається з постійною частотою та обробляє рух куба на основі натиснутих клавiш.

Клас LVLSScript містить методи для завантаження різних рівнів гри. Він відповідає за перехід між різними рівнями.

Методи:

Back(): Завантаження головного меню гри.

LOADLVL1(): Завантажує "Рівень 1".

LOADLVL2(): Завантажує "Рівень 2".

LOADLVL3(): Завантажує "Рівень 3".

LOADLVL4(): Завантажує "Рівень 4".

LOADLVL5(): Завантажує "Рівень 5".

LOADLVL6(): Завантажує "Рівень 6".

LOADLVL7(): Завантажує "Рівень 7".

LOADLVL8(): Завантажує "Рівень 8".

LOADLVL9(): Завантажує "Рівень 9".

LOADLVL10(): Завантажує "Рівень 10".

Клас MainMenu представляє головне меню гри. Він відповідає за обробку натискань кнопок у головному меню.

Поля:

NewG: Button: Клавiша для початку нової гри.

Exi: Button: Клавiша для виходу з гри.

LVL: Button: Кнопка, щоб відкрити меню вибору рівня.

Методи:

NewGame(): Завантажує сцену "Рівень 1" для початку нової гри.

Exit(): Виходить із гри.

LVLS(): Завантажує сцену "LVLS", щоб відкрити меню вибору рівня.

Клас GameOver відображає екран Game Over після завершення гри. Він обробляє анімацію та перехід до головного меню.

Поля:

animationDuration: float: Тривалість анімації Game Over.

positionCurve: AnimationCurve: Крива анімації для позиції.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

alphaCurve: AnimationCurve: Крива анімації для прозорості.

rectTransform: RectTransform: Компонент, який відповідає за позиціонування елементів інтерфейсу.

canvasGroup: CanvasGroup: Компонент, який управляє прозорістю елементів інтерфейсу.

startPosition: Vector3: Початкова позиція елементів інтерфейсу.

targetPosition: Vector3: Цільова позиція елементів інтерфейсу.

startAlpha: float: Початкове прозорість елементів інтерфейсу.

targetAlpha: float: Цільове значення прозорості елементів інтерфейсу.

startTime: float: Час початку анімації.

Методи:

Start(): Виконується при запуску гри та ініціалізує початкові значення.

Update(): Викликається кожен кадр та обробляє анімацію Game Over.

GoToMainMenu(): Завантажує сцену головного меню після завершення анімації Game Over.

Клас Confirm відображає екран підтвердження під час виходу з гри або перемикання рівнів. Він опрацьовує вибір користувача.

Поля:

previousSceneName: string: Назва попередньої сцени, збереженої з MainCube класу.

Методи:

GoToMainMenu(): Завантажує головне меню гри.

GoBack(): Повертає до попередньої сцени, збереженої з MainCube.

Створивши діаграму класів та детально її проаналізувавши, потрібно ще створити зв'язки для самої діаграми (рис 2.4), оскільки зв'язки в діаграмі відіграють доволі важливу роль, приведемо декілька прикладів, для чого вони потрібні та чому вони настільки важливі.

Візуалізація взаємодії: Зв'язки в діаграмі допомагають візуалізувати взаємодію та комунікацію між різними елементами системи, такими як

класи, об'єкти чи компоненти. Вони показують, як компоненти взаємодіють один з одним і передають дані чи викликають методи.

Визначення залежностей: Зв'язки у діаграмі допомагають визначити залежність між різними елементами системи. Вони показують, що один елемент залежить від іншого для виконання певних операцій або отримання даних. Залежності можуть бути спрямованими або двосторонніми, і їхнє розуміння допомагає розробникам враховувати ці залежності при проектуванні та зміні системи.

Відстеження впливу змін: Зв'язки в діаграмі допомагають відстежувати вплив змін на інші елементи системи. При внесенні змін до одного елемента системи можна визначити, які інші елементи можуть бути порушені та вимагають відповідних модифікацій. Це допомагає скоротити потенційні проблеми та полегшує розуміння впливу змін.

Загалом встановлення зв'язків у діаграмі важливе для візуалізації взаємодії, розуміння потоку управління, визначення залежностей, відстеження впливу змін, а також для документації та комунікації в рамках проекту. Вони допомагають створити повне уявлення про систему та покращити її розуміння та розробку.

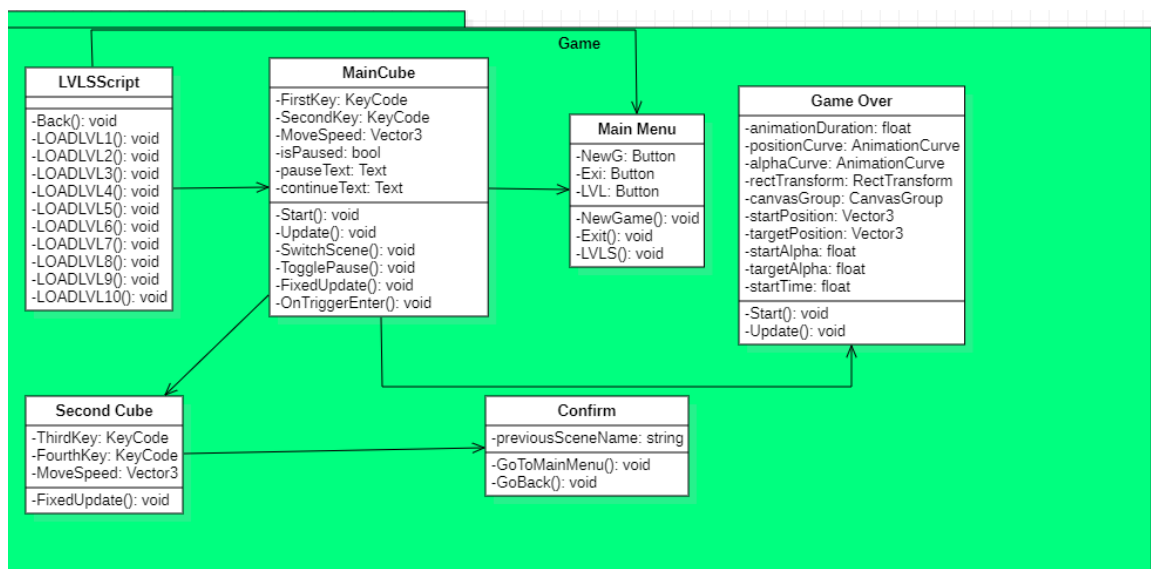


Рис.2.4 Оновлена діаграма класів зі зв'язками.

Встановивши зв'язки в діаграмі потрібно описати їх між собою. Чому саме вони зв'язані і для чого.

LVLSScript та MainCube:

LVLSScript містить методи від LOADLVL1() до LOADLVL10(), які служать для завантаження різних рівнів гри. MainCube як основний ігровий об'єкт може вимагати завантаження рівня за допомогою цих методів.

MainCube та SecondCube:

Обидва класи, MainCube та SecondCube, мають змінні MoveSpeed, які визначають швидкість руху об'єктів у грі. Це дозволяє їм взаємодіяти та обробляти рух об'єктів з використанням подібних параметрів швидкості.

MainCube та MainMenu:

MainMenu містить кнопки NewG, Exi та LVL, а також відповідні методи NewGame(), Exit() та LVLS(). MainCube може взаємодіяти з MainMenu, викликаючи ці методи, наприклад для запуску нової гри, виходу з гри або відкриття меню вибору рівнів.

GameOver та MainCube:

GameOver має змінні та методи, що відповідають за анімацію та положення екрану Game Over. MainCube містить методи Start() та Update(). Зв'язок між GameOver та MainCube дозволяє GameOver використовувати методи Start() та Update() MainCube для обробки початку та оновлення екрану Game Over.

Confirm та MainMenu:

Confirm містить змінну previousSceneName, а також методи GoToMainMenu() та GoBack(). MainMenu має кнопки NewG, Exi та LVL, а також відповідні методи NewGame(), Exit() та LVLS(). Зв'язок між Confirm та MainMenu дозволяє Confirm викликати методи NewGame() та Exit() MainMenu, а також виконувати навігацію назад або перехід до головного меню за допомогою методів GoToMainMenu() та GoBack().

Встановлені зв'язки відображають взаємодію між класами та надають доступ до функціональності, яку один клас пропонує іншому. Зв'язки дозволяють класам співпрацювати, обмінюватися даними та викликати методи один одного, що у результаті допомагає реалізувати логіку гри чи програми.

2.3 Проектування та реалізація алгоритмів роботи системи

Проектування та реалізація алгоритмів роботи системи відіграють важливу роль у розробці програмного забезпечення. Алгоритми є послідовністю інструкцій або операцій, які визначають логіку роботи системи. Вони описують, як система має виконувати певні завдання чи вирішувати певні проблеми. В нашому випадку алгоритмом роботи системи буде блок-схема. Наведемо декілька переваг блок-схеми:

Ясність та зрозумілість: Блок-схема представляє алгоритм у графічному вигляді, що робить його більш зрозумілим та легким для сприйняття. Вона дозволяє краще уявити послідовність операцій, умови, цикли та інші елементи алгоритму, що допомагає розробникам та зацікавленим сторонам краще зрозуміти його логіку та функціональність.

Структурованість: Блок-схема допомагає організувати алгоритм структуровану форму. Блоки різної форми і стрілки, що з'єднують їх, відображають послідовність операцій, умови і цикли. Це допомагає розробникам легше аналізувати та відстежувати потік виконання алгоритму.

Легкість візуальної комунікації: Блок-схема надає зручний спосіб для комунікації та обговорення алгоритму між розробниками та зацікавленими сторонами. Візуальне подання алгоритму допомагає краще пояснити його логіку та структуру, а також дозволяє виявити потенційні проблеми чи покращення.

Документація та налагодження: Блок-схема є важливим засобом документації алгоритму. Вона дозволяє зберегти та уявити алгоритм у ясній і структурованій формі, що спрощує його розуміння та подальшу підтримку.

Блок-схема також може бути використана для налагодження алгоритму, дозволяючи досліджувати його потік виконання та виявляти можливі помилки чи недоліки.

Підтримка аналізу та оптимізації: Блок-схема може бути корисною при аналізі та оптимізації алгоритму. Візуальне уявлення дозволяє виявляти вузькі місця, неефективні операції чи зайві кроки. Це допомагає розробникам знаходити способи покращити алгоритм, збільшити його продуктивність чи скоротити час виконання.

Загалом блок-схема надає зрозумілий, структурований та візуальний підхід до опису алгоритмів, що допомагає розробникам, командам та зацікавленим сторонам краще зрозуміти, обговорити, документувати та оптимізувати роботу системи.

Проаналізувавши переваги та саму суть блок-схеми. Створимо блок-схему, що в загальному відповідає нашій грі (рис.2.5)

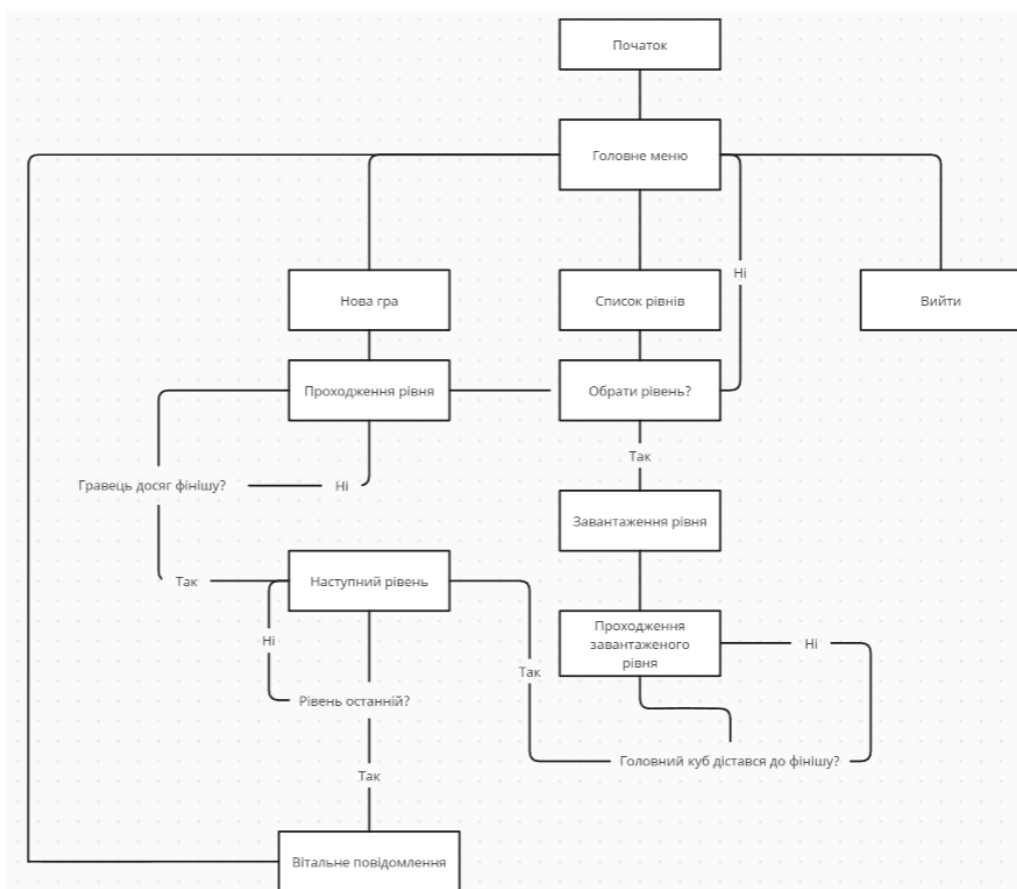


Рис.2.5. Загальна блок-схема роботи нашої гри.

Висновки до другого розділу

У другому розділі описано програмне забезпечення, за допомогою якого ми створювали діаграми, що відносяться до нашої гри. Детально розглянуто на описано архітектуру ігрового додатку. Ознайомлено з діаграмою прецедентів використання. Виокремлені її переваги, та створено власну діаграму до нашої гри. Створено діграму класів, яка є невід’ємною складовою нашого проекту.

Розглянуто важливість встановлення зв'язків у діаграмі класів для візуалізації взаємодії, розуміння потоку управління та комунікації в рамках проекту. Були наведені приклади зв'язків між різними класами у грі.

Також було обговорено переваги використання блок-схеми для проектування та реалізації алгоритмів роботи системи. Блок-схема надає зрозуміле та структуроване візуальне подання алгоритму, сприяє його аналізу, документації, комунікації та оптимізації.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЙОГО ТЕСТУВАННЯ

3.1 Основні етапи реалізації гри

У даному розділі буде детально описано скрипт, що накладений на головний куб нашої гри [2]. Буде створено загальний огляд функцій та методів.

```
private void Start()
{
    pauseText.enabled = false;
    continueText.enabled = false;
}
```

Start(): Цей метод викликається під час запуску скрипта і використовується для ініціалізації початкового стану. У цьому випадку він відключає тексти паузи та продовження гри.

```
private void Update()
{
    if (Input.GetKeyDown(KeyCode.P))
    {
        TogglePause();
    }

    if (Input.GetKeyDown(KeyCode.Escape))
    {
        SwitchScene();
    }
}
```

Update(): Цей метод викликається кожен кадр і використовується для оновлення стану гри. У цьому випадку він перевіряє натискання клавіш "P" та "Escape" для виклику відповідних функцій.

```
private void SwitchScene()
{
    PlayerPrefs.SetString("PreviousScene", SceneManager.GetActiveScene().name);
    SceneManager.LoadScene("Confirm");
}
```

SwitchScene(): Ця функція викликається при натисканні клавіші Escape і використовується для перемикавання на сцену Confirm. Вона зберігає ім'я поточної активної сцени PlayerPrefs і завантажує сцену "Confirm" за допомогою методу LoadScene() з класу SceneManager.

```
private void TogglePause()
{
    isPaused = !isPaused;
}
```

```

    if (isPaused)
    {
        Time.timeScale = 0f;
        pauseText.enabled = true;
        continueText.enabled = true;
    }
    else
    {
        Time.timeScale = 1f;
        pauseText.enabled = false;
        continueText.enabled = false;
    }
}

```

TogglePause(): Ця функція викликається при натисканні клавіші "P" і використовується для перемикавання стану паузи гри. Вона змінює значення змінної isPaused, встановлює відповідну тимчасову шкалу гри (Time.timeScale) та включає/вимикає тексти паузи та продовження гри.

```

private void FixedUpdate()
{
    if (!isPaused)
    {
        if (Input.GetKey(FirstKey))
        {
            GetComponent<Rigidbody>().velocity += MoveSpeed;
        }
        if (Input.GetKey(SecondKey))
        {
            GetComponent<Rigidbody>().velocity -= MoveSpeed;
        }
        if (Input.GetKey(KeyCode.R))
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }

        if (Input.GetKey(KeyCode.N))
        {
            if (SceneManager.GetActiveScene().name == "Рівень 10")
            {
                return;
            }
            else
            {
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
            }
        }
    }
}

```

FixedUpdate(): Цей метод викликається постійною частотою і використовується для виконання фізичних обчислень. У цьому випадку він перевіряє стан паузи та обробляє натискання клавіш FirstKey, SecondKey, "R" і "N". Залежно від натиснутої клавіші, метод змінює швидкість руху куба, перезавантажує поточну сцену або завантажує наступну сцену (за винятком сцени "Рівень 10").

```
private void OnTriggerEnter(Collider other)
{
    if (this.CompareTag("Player") && other.CompareTag("Finish"))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
}
```

OnTriggerEnter(Collider other): Цей метод викликається, коли колайдер об'єкта спрацьовує тригерну подію зіткнення. В даному випадку він перевіряє, чи є поточний об'єкт з тегом "Player" і зіткнувся з об'єктом, що має тег "Finish". Якщо умова виконується, метод завантажує наступну сцену.

3.2 Створення ігрового середовища та інтерфейсу користувача

Ігрове поле в U3D створюється шляхом розміщення та розташування об'єктів у сцені. Зазвичай, ігрове поле складається з різних елементів, таких як земля, стіни, об'єкти оточення, персонажі та інші ігрові об'єкти [6].

Створення ігрового поля включає в себе наступні аспекти:

Створення об'єктів: Створення та наштування ігрових об'єктів, які становитимуть ігрове поле.

Розташування об'єктів: Розміщення створених об'єктів в сцені, визначаючи їхню позицію, поворот і масштаб. Використовуюючи редактор сцени Unity для переміщення та обертання об'єктів.

Компоненти та скрипти: Прикріплення різних компонентів до об'єктів ігрового поля, щоб додати їм функціональність.

Декорування та налаштування: Додавання додаткових елементів для покращення зовнішнього вигляду та функціональності ігрового поля.

Тестування: Після створення ігрового поля важливо провести тестування, щоб переконатися, що все працює правильно.

Загальна ідея полягає в тому, щоб використовувати можливості Unity для створення та налаштування об'єктів, їх розміщення у сцені та додавання

необхідної функціональності та візуальних ефектів, щоб створити бажане ігрове поле [3].

Перш за все створюємо ігрове поле за допомогою об'єкту "Plane", якому встановлюється довільний розмір. В нашому випадку розмір об'єкту буде наступним: $X - 1.5$, $Y - 1$, $Z - 1.5$, та вже нанесені на нього текстури. (рис.2.1)

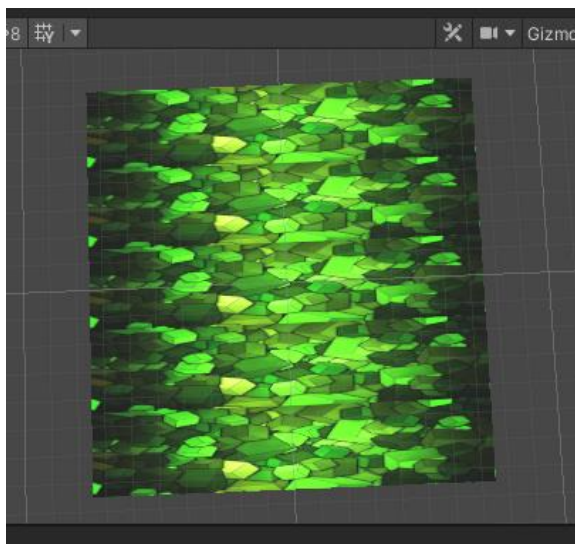


Рис.3.1. Створення ігрового поля.

Після чого, потрібно створити обмеження для кубів, щоб під час переміщення по ігровому полю, вони не вилітали за межі поля. В нашому випадку стіни буде створено з 5 розтягнутих кубів, на які вже нанесено текстури. Також залишаємо місце для фінішу. (рис.2.2)

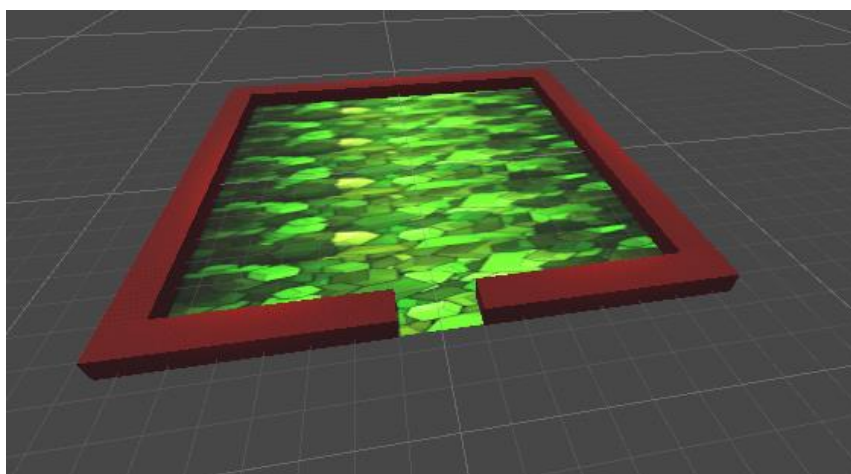


Рис.3.2. Створення стін до ігрового поля.

Загальне ігрове поле доволі просте, та вже готове. Залишається лише наповнити його головним, та другорядними об'єктами, та додати вже сам фініш і перевірку до нього.

Створивши ігрове поле нам тепер потрібно наповнити його ігровими об'єктами. Як було вказано раніше, в нашій грі буде наявно 4 типи кубів.

- Головний куб (РВ, РН)
- Допоміжний куб (РП, РЛ)
- Перешкода
- Другорядний куб (РВ, РН)

Головний куб (MainCube) представляє основного персонажа гри. У нашому проєкті головний куб є ігровим об'єктом, який керуватиме гравець у процесі гри. Саме цим кубом гравцю потрібно дістатися до фінішу, задля того, щоб перейти на наступний рівень. (рис 2.3).

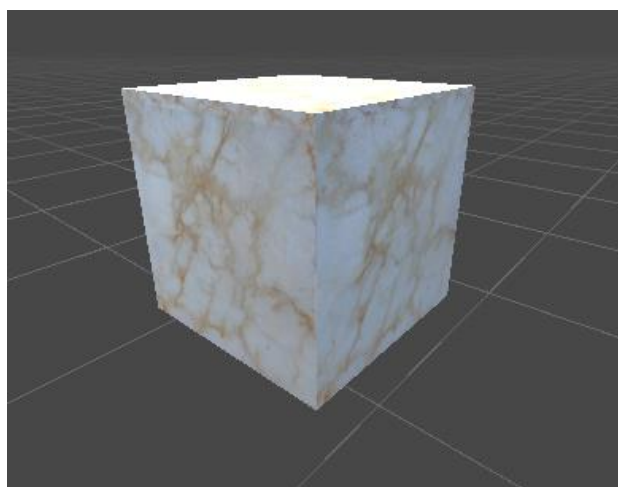


Рис.3.3. Головний куб

Створивши головний куб, тепер потрібно подбати про створення допоміжного куба (рис 2.4), другорядного куба (рис 2.5), та самої перешкоди (рис 2.6), що буде заважати гравцю дістатися до фінішу.

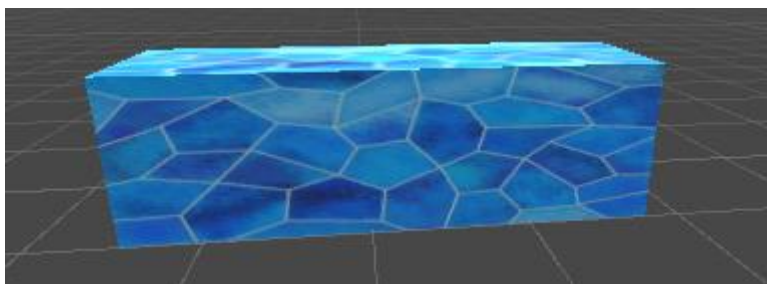


Рис.3.4. Допоміжний куб

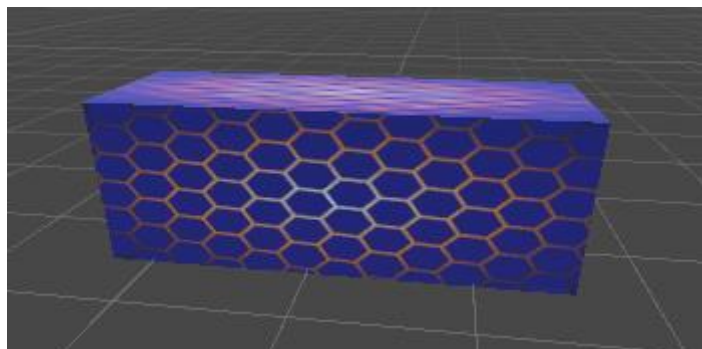


Рис.3.5. Другорядний куб

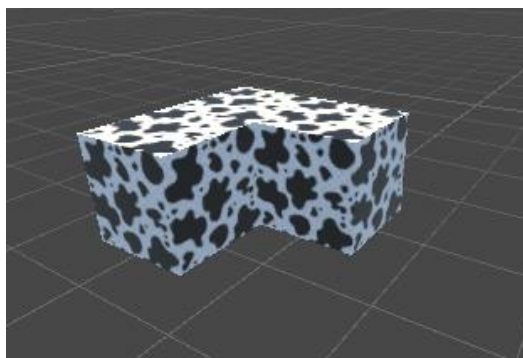


Рис. 3.6. Приклад перешкоди

Другорядний куб та допоміжний куб є додатковими ігровими об'єктами, які будуть взаємодіяти з головним кубом і оточенням у нашій грі. Куби при рухах можуть взаємодіяти з головним кубом, зміщуючи його по ігровому полю, допомагаючи гравцю дістатися до фінішу, щоб пройти рівень та перейти на наступний. Куби з'являються в певних місцях на ігровому полі. Місце їх появлення задані заздалегідь.

Перешкода також являється додатковим ігровим об'єктом. Вона взагалі не рухається та не зсувається, можна сказати, що це статичний куб, який заважає гравцю дістатися до фінішу.

Створивши всі ігрові об'єкти, та додавши їх в префаби, щоб в наступних сценах не створювати все з нуля, ми в результаті маємо готове ігрове поле та ігрові об'єкти. Також сформуємо перший рівень (рис 2.7), що є доволі простим, та знайомить гравця з основною механікою переміщення ігрових об'єктів [4].

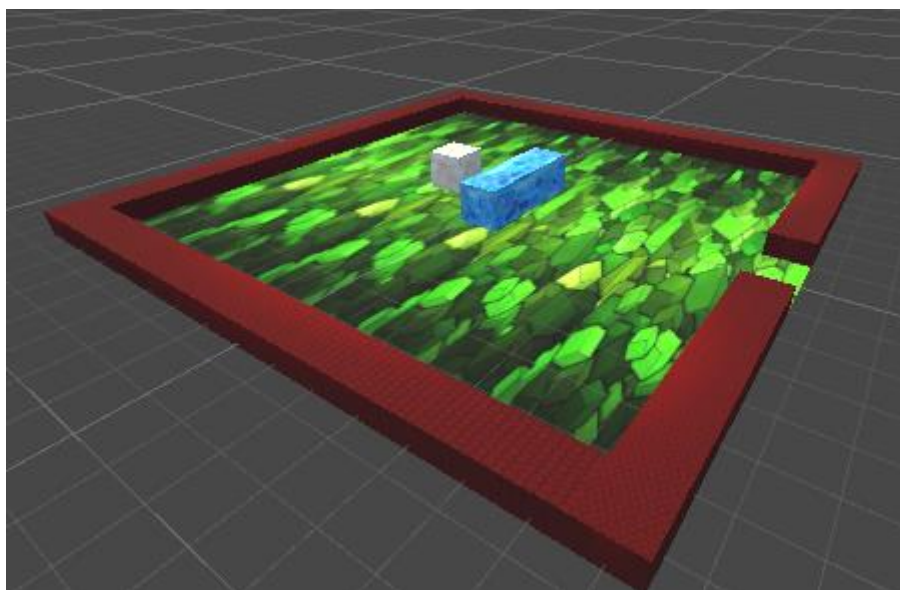


Рис.3.7. Створення рівня

Інтерфейс користувача (UI) у загальному значенні являє собою засіб взаємодії між користувачем та комп'ютерною програмою чи системою. Він забезпечує користувачеві зручний та інтуїтивно зрозумілий спосіб контролю та управління програмним забезпеченням, відображення інформації та отримання зворотного зв'язку.

Рішення про вибір кольорів інтерфейсу користувача залежить від багатьох факторів, включаючи тему гри, цільову аудиторію, стиль та настрій.

Найважливіше при виборі колірної оформлення інтерфейсу користувача врахувати такі фактори:

1. Цільова аудиторія Гра може бути призначена для різних вікових груп або типів гравців. Можливо, колірна схема має бути адаптована під переваги та очікування цільової аудиторії.

2. Тема гри: Якщо гра має певну тематику, колірна схема може бути пов'язана із цією темою. Наприклад, гра з фантастичними елементами може мати яскраві та незвичайні кольори, тоді як гра з серйозним сюжетом може віддавати перевагу більш приглушеним і нейтральним кольорам.

3. Контрастність та читання: Важливо, щоб кольори, що використовуються в інтерфейсі, забезпечували хорошу контрастність та читаність. Це дозволяє користувачам легко розрізняти елементи інтерфейсу, текст та візуальні елементи.

4. Емоційна реакція: Кольори можуть викликати емоційні реакції користувачів. Наприклад, яскраві та насичені кольори можуть створювати відчуття енергії та веселощів, тоді як спокійні та приглушені кольори можуть викликати почуття розслабленості чи серйозності.

Ознайомившись з загальними факторами при виборі колірної оформлення інтерфейсу було сформовано загальну колірну палітру для створення інтерфейсу користувача. Далі буде описано процес створення інтерфейсу. Перш за все потрібно створити сцену з головним меню (рис 2.7) в якому буде можливість пересуватися по грі, також додати задній фон для нашого ігрового поля, щоб не залишати його по замовчуванню (рис 2.8). Під час створення було обрано основою темно-синій колір.

Темно-синій колір часто асоціюється зі спокоєм та розслабленим станом. Він має заспокійливий ефект на глядачів та може допомогти створити атмосферу комфорту та умиротворення.

Дослідження психології кольору показують, що синій колір може знижувати рівень стресу, знижувати пульс та тиск, а також сприяти релаксації. Він також може викликати почуття прохолоди та свіжості.



Рис.3.8 Головне меню.

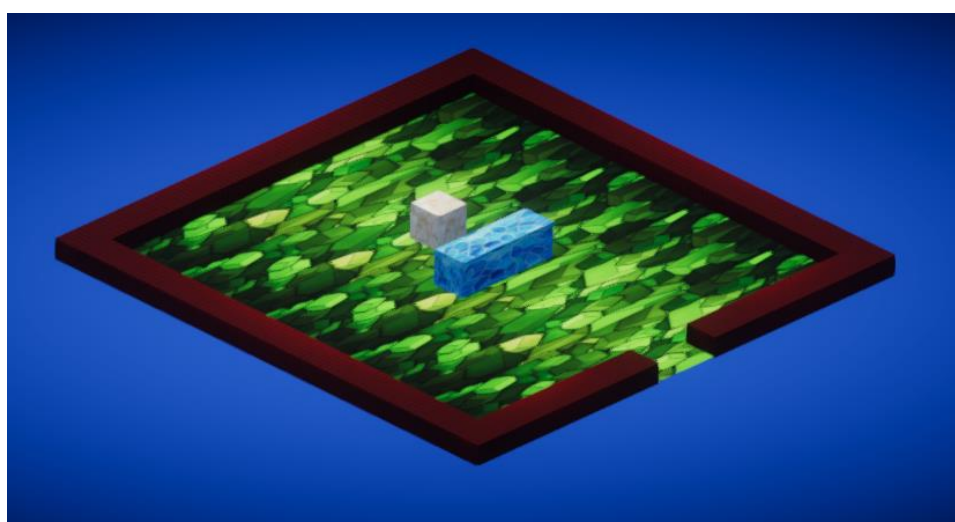


Рис.3.9 Загальний рівень

Також потрібно додати текстові надписи (рис 2.9) на самому рівні, щоб користувач розумів на якому рівні він знаходиться, та підказки таких механік як: перехід на наступний рівень, та початок рівня спочатку.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

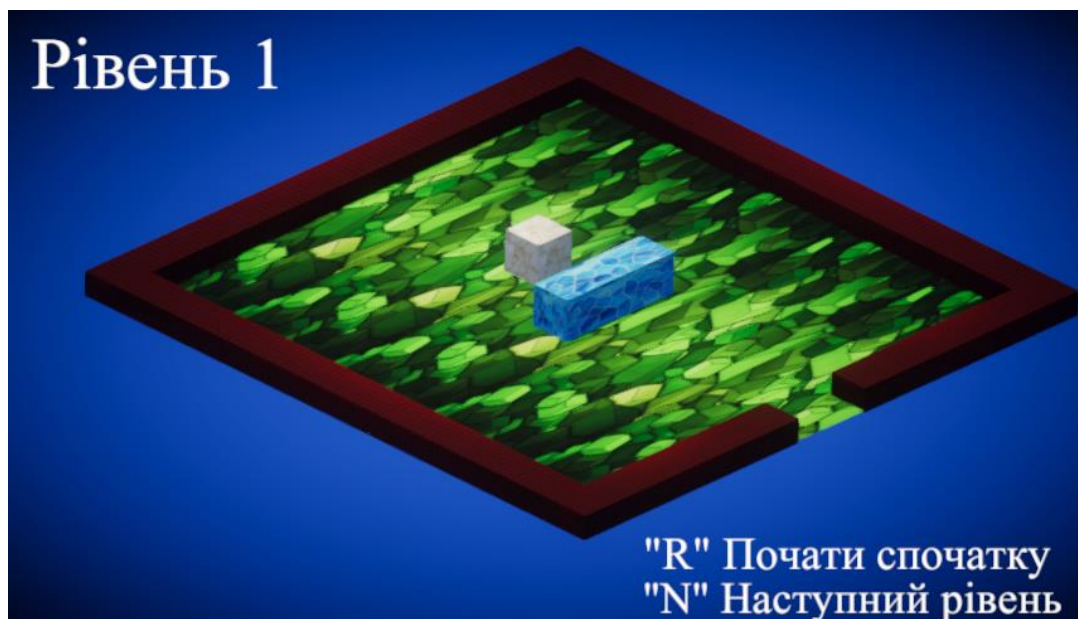


Рис.3.10. Оновлений вигляд

3.3 Опис роботи з програмним додатком та тестування

Після запуску гри, першою сценою є головне меню (рис 3.1), в якому наявні три елементи керування, що мають вигляд кнопок.

Перша з них «Нова гра» переносить гравця на перший рівень головоломки. Друга «Список рівнів» на список, з якого гравець може обрати будь-який рівень з наявних, або ж повернутися назад до головного меню. Третя ж закриває гру.

Кнопки також мають свій колір, залежно від дії. Звичайний колір кнопок – зелений, під час наведення на них курсора, вони стають рожеві, після натискання – блакитні.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49



Рис.3.11. Головне меню.

Перед початком нової гри, перейдемо у «Список рівнів» (рис.3.2), та перевіримо загальну їх кількість і можливість повернутися до головного меню нашого продукту. Список рівнів відображається за допомогою рисунків з нумерацією кожного рівня. На кожен рисунок додано об'єкт «Button», та прописаний рядок коду, за допомогою якого здійснюється переміщення гравця, після кліку мишкою на будь-який рівень.



Рис.3.12. Список рівнів

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Перевіливши список рівнів, та загальну їх кількість, ми можемо завантажити будь-який з представлених рівнів. Повернемося до головного меню, та натиснемо на «Нова гра». Після натискання кнопки, гравця переміщає на перший рівень гри (рис.3.3). Де вже можна починати гру, переміщуючи куби за допомогою клавіш.

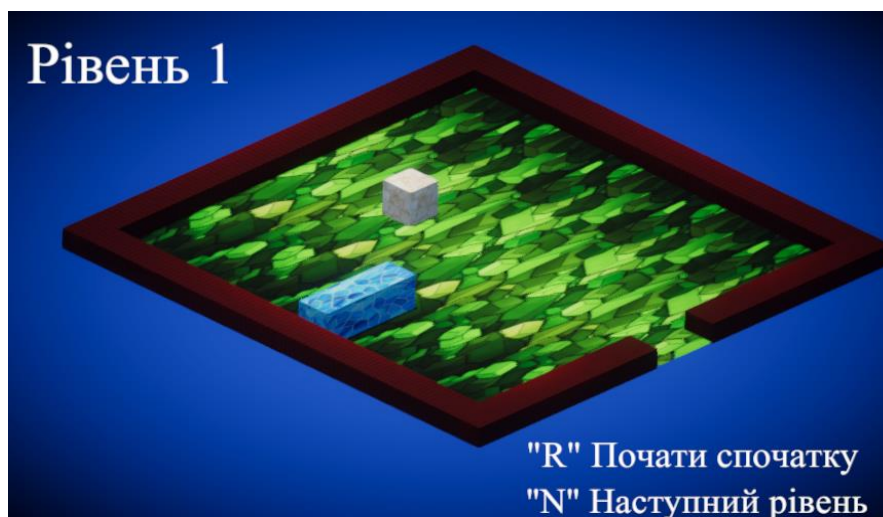


Рис.3.13 Перший рівень.

У грі також присутні можливості поставити її на паузу (рис.3.4.). Для того, щоб поставили гру на паузу, гравцю потрібно натиснути на кнопку «Р» на клавіатурі. Після чого з'являється відповідне повідомлення, що гра призупинена. Для відновлення стану гри потрібно ще раз натиснути на ту ж саму кнопку.



Рис.3.14 Статус паузи

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

Після відновлення стану гри, ми можемо продовжити проходження рівня. Також наявні підказки стосовно клавіші «R», та «N», спочатку перевіримо можливість почати рівень спочатку (рис 3.5), для цього розкидаємо куби, та натиснемо на клавішу «R».



Рис.3.15. Результат натиску «R»

Після перевірки можливості почати спочатку бачимо другу підказку, що надає можливість гравцю перейти до наступного рівня (рис 3.6), якщо цей рівень йому не до вподоби, або він занадто складний, а бажання побачити наступний рівень присутнє.

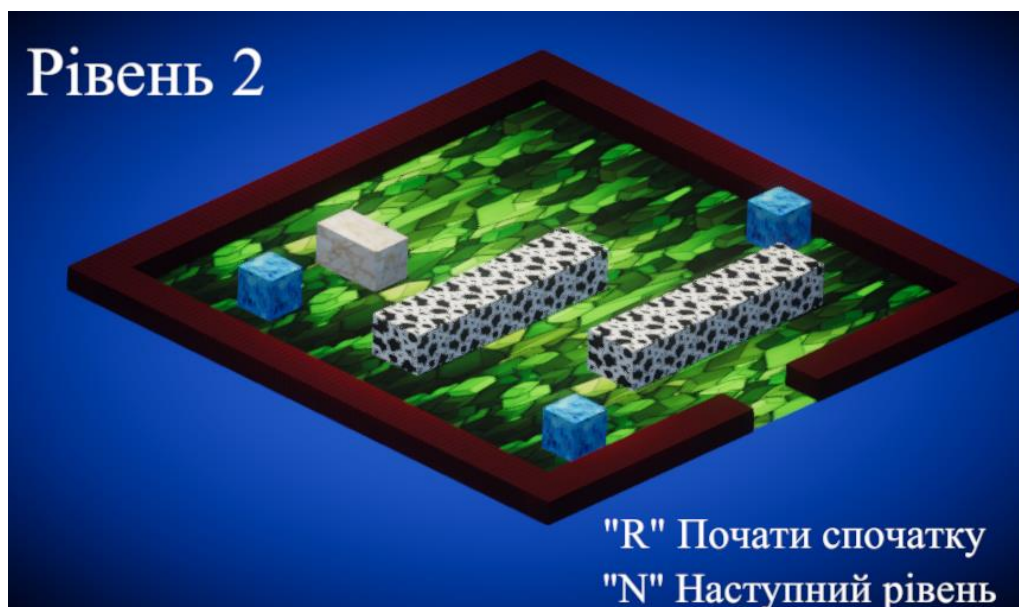


Рис.3.16. Перехід на наступний рівень

Також наявна можливість перейти до головного меню прямо з самого рівня, для цього гравцю потрібно натиснути «Esc». Після чого з'явиться

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

меню підтвердження (рис 3.7). Натиснувши кнопку «Так» гравця переносить до головного меню, натиснувши «Ні», він повертається до проходження рівня.

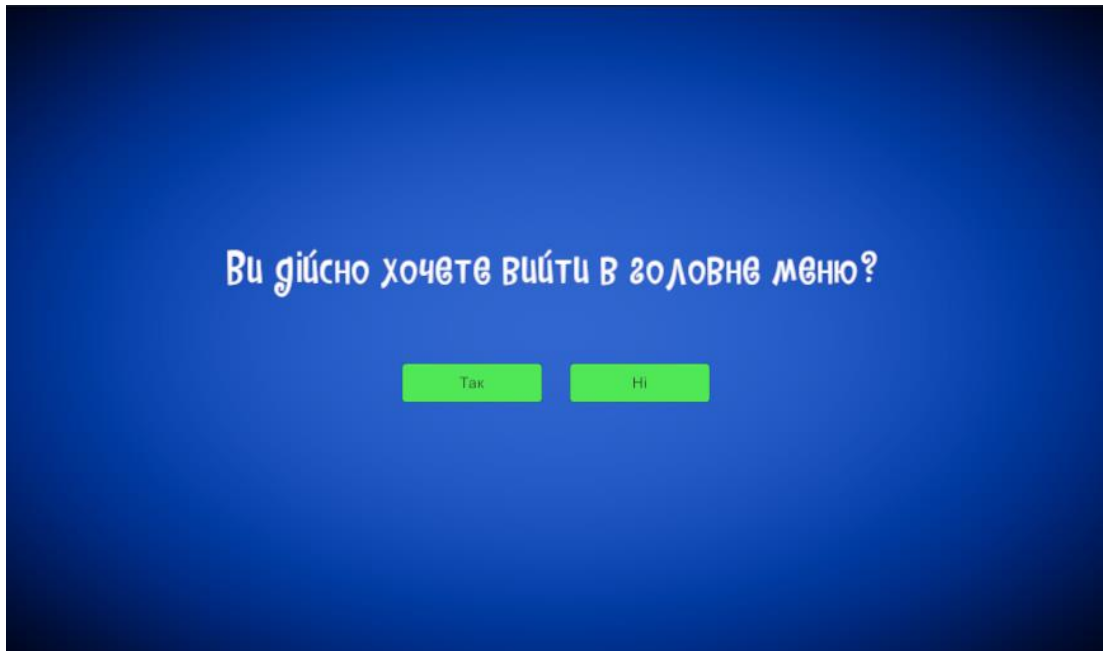


Рис.3.17. Сцена підтвердження

Також після проходження останнього рівня перед гравцем (де відсутня можливість перейти на наступний рівень) з'являється вітальне повідомлення, про успішне проходження гри (рис 3.8). Після чого гравця переносить у головне меню, де він може почати гру заново, або ж обрати будь-який рівень для пошуку інших способів розв'язку головоломки.



Рис.3.18. Вітальне повідомлення

Тестування - це процес, що включає проведення різних тестів для перевірки функцій, якості та надійності програмного додатку. Мета тестування полягає у виявленні помилок та дефектів, щоб забезпечити високу якість та безперебійну роботу програми.

Існує декілька технік тест дизайну, для тестування програмного додатку буде використано техніку «Cause/Effect» (Причина/Наслідок).

Кнопка «Нова гра»

Передумови: Гравець повинен знаходитись у головному меню

Кроки: Натискання кнопки «Нова гра»

Очікуваний результат: Перехід до першого рівня

Фактичний результат: Перехід до першого рівня

Кнопка «Список рівнів»

Передумови: Гравець повинен знаходитись у головному меню

Кроки: Натискання кнопки «Список рівнів»

Очікуваний результат: Перехід до загального списку рівнів

Фактичний результат: Перехід до загального списку рівнів

Кнопка «Назад»

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Передумови: Гравець повинен знаходитись у списку рівнів

Кроки: Натискання кнопки «Назад», для повернення до головного меню

Очікуваний результат: Повернення до головного меню

Фактичний результат: Повернення до головного меню

Кнопка «Вихід»

Передумови: Гравець повинен знаходитись у головному меню

Кроки: Натискання кнопки «Вихід»

Очікуваний результат: Гравець виходить зі гри

Фактичний результат: Гравець виходить зі гри

Перевірка повернення до головного меню

Передумови: Гравець повинен знаходитись на одному з рівнів.

Кроки: Натискання кнопки «Esc»

Очікуваний результат: Перед гравцем з'являється підтвердження, чи дійсно він хоче вийти до головного меню.

Фактичний результат: Перед гравцем з'являється підтвердження, чи дійсно він хоче вийти до головного меню.

Натискання кнопки «Так»

Передумови: Гравець повинен отримати підтвердження про вихід до головного меню

Кроки: Натискання кнопки «Так»

Очікуваний результат: Гравця повертає до головного меню

Фактичний результат: Гравця повертає до головного меню

Натискання кнопки «Ні»

Передумови: Гравець повинен отримати підтвердження про вихід до головного меню

Кроки: Натискання кнопки «Ні»

Очікуваний результат: Гравця повертає до проходження рівня

Фактичний результат: Гравця повертає до проходження рівня

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

Під час тестування програмного забезпечення було виявлено та усунуено декілька багів:

1. Під час запуску рівня, куби, що знаходилися на ігровому полі зміщувалися вправу сторону, без натискання кнопок переміщення. Причиною цього багу був невеликий нахил ігрового поля в праву сторону, через що всі об'єкти (окрім перешкоди) також зміщувалися.

2. Головний та другорядні куби перелітали через нерухому перешкоду.

Причиною цього багу була відсутність заморозки кубів по Y. Через що вони перелітали перешкоду.

3. Якщо другорядний куб досягав фінішу, гравця переносило на наступний рівень.

Цей баг був зумовлений тим, що під час створення кубів, другорядний куб брали з префабу головного, після чого вже накладали текстури. Не змінивши при цьому тег. У другорядного куба так само як і в головного був тег «Player», тому гра вважала, що головний куб дійшов до фінішу і переносила гравця на наступний рівень.

Висновки до третього розділу

У третьому розділі було докладно описано, як здійснюється робота з програмним додатком. Було розглянуто головне меню гри, яке містить три основні кнопки. Описано дії, які виконуються під час натискання на кожну з цих кнопок. Описано процес створення ігрового поля, ігрових об'єктів, користувацького інтерфейсу. Також описано основні етапи реалізації гри.

Також було описано можливість переходу до наступного рівня, якщо поточний рівень не сподобався гравцю або виявився надто складним. Це забезпечує гнучкість та вибір гравцем у тому, який рівень проходити.

Крім того, була представлена функція Esc, яка дозволяє гравцеві повернутися в головне меню прямо з рівня. При натисканні на "Esc" з'являється підтвердження про вихід у головне меню, і гравець може

вибрати, чи він підтверджує цю дію чи продовжує проходження поточного рівня.

Для перевірки роботи програмного забезпечення було використано техніку "Cause/Effect". Були створені тестові сценарії для перевірки кожної кнопки та функції. Очікувані результати були зіставлені з фактичними результатами. У процесі тестування було виявлено та усунуто деякі помилки, такі як зміщення кубів на ігровому полі без натискання кнопок переміщення, проходження кубами через нерухомі перешкоди та проходження рівня за допомогою другорядного куба.

					<i>КН.КР.Б. – 122 – 23 - ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було створено гру «Labyrinths of the Mind»

Була представлена загальна інформація про проект, його цілі та завдання. Описано гру-головоломку, основні механіки та вимоги до розробки програмного застосування. Було проведено аналіз існуючих аналогів та визначено особливості, які роблять цю гру унікальною. В результаті цього розділу було отримано ключові вимоги та напрямки для подальшої розробки.

Представлена розробка програми. Була описана архітектура програми та її основні компоненти, включаючи ігровий рушій, графічний інтерфейс користувача та логіку гри. Були представлені діаграми класів та послідовності для наочного подання взаємодії компонентів. Також було розглянуто особливості реалізації рівнів та управління ігровим процесом.

Описано роботу з програмним додатком та його тестування. Були представлені основні елементи інтерфейсу, такі як головне меню, кнопки навігації та підтвердження дій. Були описані можливості переходу між рівнями та виходу з гри. Також було використано техніку тест-дизайну "Cause/Effect" для перевірки функціональності програми. Було виявлено та усунуено помилки, пов'язані з переміщенням кубів та проходженням через перешкоди.

Загалом, даний проект є завершеною розробкою гри-головоломки. Тестування дозволило виявити та виправити помилки, забезпечуючи високу якість та надійність програми.

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Unity in Action: Multiplatform Game Development in C#" автора Джоноффа Маннинга (Jon Manning).
2. "Beginning C# Game Programming" автора Харри Чесера (Harrison Ferrone).
3. "Unity in Action: Multiplatform Game Development in C#" автора Джошуа Стріттера.
4. "Unity Game Development Cookbook" автора Квінна Беррі.
5. "C# 7.0 in a Nutshell: The Definitive Reference" авторів Джозефа та Бенджаміна Альбахарі.
6. "Створення гри головоломки: Від ідеї до реалізації" авторства Олександра Мельникова.
7. "UML з прикладами. Основи моделювання програмного забезпечення" від Андрія Лесів та Ігоря Поліщу

					КН.КР.Б. – 122 – 23 - ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Лістинг MainCube.cs

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

public class MainCube : MonoBehaviour
{
    [SerializeField] KeyCode FirstKey;
    [SerializeField] KeyCode SecondKey;
    [SerializeField] Vector3 MoveSpeed;

    private bool isPaused = false;
    public Text pauseText, continueText;

    private void Start()
    {
        pauseText.enabled = false;
        continueText.enabled = false;
    }

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.P))
        {
            TogglePause();
        }

        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SwitchScene();
        }
    }

    private void SwitchScene()
    {
        PlayerPrefs.SetString("PreviousScene", SceneManager.GetActiveScene().name);
        SceneManager.LoadScene("Confirm");
    }

    private void TogglePause()
    {
        isPaused = !isPaused;

        if (isPaused)
        {
            Time.timeScale = 0f;
            pauseText.enabled = true;
            continueText.enabled = true;
        }
        else
        {
            Time.timeScale = 1f;
            pauseText.enabled = false;
            continueText.enabled = false;
        }
    }

    private void FixedUpdate()
    {

```

```

        if (!isPaused)
        {
            if (Input.GetKey(FirstKey))
            {
                GetComponent<Rigidbody>().velocity += MoveSpeed;
            }
            if (Input.GetKey(SecondKey))
            {
                GetComponent<Rigidbody>().velocity -= MoveSpeed;
            }
            if (Input.GetKey(KeyCode.R))
            {
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
            }

            if (Input.GetKey(KeyCode.N))
            {
                if (SceneManager.GetActiveScene().name == "Рівень 10")
                {
                    return;
                }
                else
                {
                    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
                }
            }
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        if (this.CompareTag("Player") && other.CompareTag("Finish"))
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        }
    }
}

```

Лістинг SecondCube.cs

```

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

public class SecondCube : MonoBehaviour
{
    [SerializeField] KeyCode ThirdKey;
    [SerializeField] KeyCode FourthKey;
    [SerializeField] Vector3 MoveSpeed;

    private void FixedUpdate()
    {
        if (Input.GetKey(ThirdKey))
        {
            GetComponent<Rigidbody>().velocity += MoveSpeed;
        }
        if (Input.GetKey(FourthKey))
        {
            GetComponent<Rigidbody>().velocity -= MoveSpeed;
        }
    }
}

```

Лістинг MainMenu.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public Button NewG;
    public Button Exi;
    public Button LVL;

    public void NewGame()
    {
        SceneManager.LoadScene("Рівень 1");
    }

    public void Exit()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }

    public void LVLS()
    {
        SceneManager.LoadScene("LVLS");
    }
}
```

Лістинг GameOver.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameOver : MonoBehaviour
{
    public float animationDuration = 3f;
    public AnimationCurve positionCurve;
    public AnimationCurve alphaCurve;

    private RectTransform rectTransform;
    private CanvasGroup canvasGroup;
    private Vector3 startPosition;
    private Vector3 targetPosition;
    private float startAlpha;
    private float targetAlpha;
    private float startTime;

    private void Start()
    {
        rectTransform = GetComponent<RectTransform>();
        canvasGroup = GetComponent<CanvasGroup>();

        startPosition = new Vector3(0f, -200f, 0f);
        targetPosition = new Vector3(0f, 200f, 0f);
    }
}
```

```

        startAlpha = 0f;
        targetAlpha = 1f;

        rectTransform.anchoredPosition = startPosition;
        canvasGroup.alpha = startAlpha;

        startTime = Time.time;
    }

    private void Update()
    {
        float elapsedTime = Time.time - startTime;

        float positionProgress = Mathf.Clamp01(elapsedTime / animationDuration);
        rectTransform.anchoredPosition = Vector3.Lerp(startPosition, targetPosition,
        positionCurve.Evaluate(positionProgress));

        float alphaProgress = Mathf.Clamp01(elapsedTime / animationDuration);
        canvasGroup.alpha = Mathf.Lerp(startAlpha, targetAlpha,
        alphaCurve.Evaluate(alphaProgress));

        if (elapsedTime >= animationDuration)
        {
            SceneManager.LoadScene("Main Menu");
        }
    }
}

```

Лістинг LVLSScript.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LVLSScript : MonoBehaviour
{
    public void Back()
    {
        SceneManager.LoadScene("Main Menu");
    }

    public void LOADLVL1()
    {
        SceneManager.LoadScene("Рівень 1");
    }

    public void LOADLVL2()
    {
        SceneManager.LoadScene("Рівень 2");
    }
    public void LOADLVL3()
    {
        SceneManager.LoadScene("Рівень 3");
    }
    public void LOADLVL4()
    {

```

```

        SceneManager.LoadScene("Рівень 4");
    }
    public void LOADLVL5()
    {
        SceneManager.LoadScene("Рівень 5");
    }
    public void LOADLVL6()
    {
        SceneManager.LoadScene("Рівень 6");
    }
    public void LOADLVL7()
    {
        SceneManager.LoadScene("Рівень 7");
    }
    public void LOADLVL8()
    {
        SceneManager.LoadScene("Рівень 8");
    }
    public void LOADLVL9()
    {
        SceneManager.LoadScene("Рівень 9");
    }
    public void LOADLVL10()
    {
        SceneManager.LoadScene("Рівень 10");
    }
}

```

Лістинг Confirm.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Confirm : MonoBehaviour
{
    private string previousSceneName;

    private void Start()
    {
        previousSceneName = PlayerPrefs.GetString("PreviousScene", "Main Menu");
    }

    public void GoToMainMenu()
    {
        SceneManager.LoadScene("Main Menu");
    }

    public void GoBack()
    {
        SceneManager.LoadScene(previousSceneName);
    }
}

```