РЕФЕРАТ

Пояснювальна записка до дипломної роботи «3D гра на мові програмування C# з використанням Unity» містить: 55 сторінок, 36 рисунків, 5 таблиці, 4 блок схеми 9 використаних джерел, 1 додаток.

У першому розділі проаналізовано предметну область та визначені вимоги до додатку.

Другий розділ складається з огляду засобів розробки та технологій, побудови діаграм процесів, проектування таблиці бази даних.

Третій розділ містить опис розробки додатку та опис функціоналу його сегментів, розроблено інструкцію користувача.

Мета дипломної роботи – розроблення гри на мові програмування C# у середовищі розробки Unity

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

C#– об’єктно-орієнтована мова програмування.

Unity – кросплатформове інтегроване середовище розробки (IDE) для створення ігор і додатків.

Жанр гри: Головоломка - Головоломки є популярним жанром у світі відеоігор. Вони вимагають від гравця логічного мислення, розв'язування складних завдань та використання стратегій для досягнення поставленої мети. Головоломки можуть мати різноманітні форми, такі як складні лабіринти, розміщення об'єктів у певному порядку, головоломки на базі фізики тощо.

ЗМІСТ

ВСТУП

У сучасному світі розробка ігор є однією з найпопулярніших та найзахоплюючіших галузей програмування. Ігри завойовують серця мільйонів гравців по всьому світу, надаючи їм можливість поринути у віртуальні світи та взаємодіяти з уявними персонажами та об'єктами. З кожним роком ігрова індустрія стає все більш розвиненою та технологічною, пропонуючи неймовірні візуальні та аудіо ефекти, реалістичну графіку та захоплюючі сценарії.

Одним з найважливіших аспектів розробки сучасних ігор є використання тривимірної (3D) графіки, яка надає гравцям ще більш глибокий та реалістичний досвід. Для створення 3D ігор розробники користуються різноманітними інструментами та технологіями. Однак, одним з найпопулярніших середовищ розробки є Unity.

Unity - це інтегроване середовище розробки, яке надає безліч інструментів та можливостей для створення як 2D, так і 3D ігор. Воно підтримує різні платформи, включаючи комп'ютери, мобільні пристрої, консолі та віртуальну реальність. Unity володіє потужним редактором, який дозволяє розробникам створювати складні ігрові сцени, налаштовувати фізику об'єктів, програмувати штучний інтелект та створювати ефектні візуальні ефекти.

Одним з ключових інструментів, що використовуються в Unity для програмування ігрової логіки, є мова програмування C#.

C# - надає розробникам широкий функціонал та можливості для створення складних інтерактивних систем та геймплею в 3D іграх. C# є об'єктно-орієнтованою мовою програмування, яка дозволяє ефективно керувати об'єктами, взаємодіяти з графічними ресурсами, створювати складну логіку гри, реалізовувати штучний інтелект персонажів та багато іншого.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Головоломкові ігри у жанрі 3D графіки є популярним жанром серед геймерів. Для виробників і розробників ігор важливо розуміти особливості та популярність головоломкових ігор у 3D форматі. Нижче наведено детальний аналіз предметної області, який включає огляд існуючих головоломкових ігор у 3D форматі, визначення основних особливостей головоломкових ігор та їх популярності серед гравців, а також дослідження ринкових тенденцій і попиту на головоломкові ігри.

Огляд існуючих головоломкових ігор у 3D форматі:

У сучасному геймінгу існує значна кількість головоломкових ігор, які використовують 3D графіку для створення унікального геймплею та візуального досвіду. Нижче наведено огляд деяких існуючих головоломкових ігор у 3D форматі, що демонструють різноманітність жанрів та механік гри.

1. "Portal" (Valve Corporation, 2007) "Portal" є однією з найвідоміших головоломкових ігор, яка поєднує графіку у 3D форматі зі складними фізичними головоломками. Гравці керують героєм, використовуючи порталів з різних точок простору, щоб вирішувати завдання та пересуватися в складних лабіринтах.
2. "The Witness" (Thekla, Inc., 2016) "The Witness" - це головоломкова гра, яка пропонує гравцям розв'язувати логічні головоломки на головоломних островах. Гра використовує стилізовану 3D графіку та різноманітні механіки, такі як розміщення ліній, кольорові шаблони та інші складні елементи, що розкриваються по ходу гри.
3. "Monument Valley" (ustwo games, 2014) "Monument Valley" - це головоломкова гра з чарівною 3D графікою, яка поєднує елементи оптичної ілюзії з

головоломками, що базуються на перспективі. Гравці проводять головоломного персонажа через магічні структури, змінюючи їх форму та розташування, щоб вирішувати завдання.

4. "The Talos Principle" (Croteam, 2014) "The Talos Principle" - це головоломкова гра з великими відкритими світами, що використовує фотореалістичну 3D графіку. Гравці розв'язують логічні головоломки, використовуючи різноманітні механіки, такі як переміщення об'єктів, активування пристроїв та взаємодія зі світом навколо себе.

Визначення основних особливостей головоломкових ігор та їх популярності серед гравців:

Головоломкові ігри є окремим жанром в геймінгу, які привертають увагу гравців своєю викликливістю, логічними завданнями та потужними стимулами для розвитку креативного та аналітичного мислення. Основні особливості головоломкових ігор включають наступне:

1. Логічні головоломки: Головоломкові ігри пропонують гравцям різноманітні логічні завдання, які потребують аналізу, планування та розв'язування проблем. Ці завдання можуть включати розміщення об'єктів, вирішення математичних або логічних задач, розгадування шифрів тощо.
2. Виклик і складність: Головоломкові ігри часто відомі своєю високою складністю, яка ставить перед гравцем виклик і спонукає до вдосконалення навичок та стратегій. Важкі головоломки стимулюють гравців докладати зусиль та розвивати нові підходи для їх розв'язання.
3. Креативність і інновації: Головоломкові ігри часто пропонують унікальні механіки та концепції, які вимагають креативного мислення та нестандартних рішень. Вони можуть включати в себе використання фізики, оптичних ілюзій, графічних ефектів та інших неочікуваних елементів.
4. Задоволення від розв'язання: Основна мета головоломкових ігор полягає в розв'язанні головоломок і досягненні успіху. Коли гравець знаходить правильне рішення або пройшов складний рівень, він відчуває задоволення від досягнення цілі та подолання виклику.

5. Популярність серед гравців: Головоломкові ігри мають широку аудиторію серед гравців, оскільки вони пропонують цікаві та викликливі геймплейні виклики. Багато гравців віддають перевагу головоломкам, оскільки вони сприяють розвитку критичного мислення, покращують здатність до аналізу та прийняття рішень, а також надають можливість провести час у захоплюючому та інтелектуально стимулюючому середовищі.

Аналіз популярності головоломкових ігор серед гравців вказує на стійкий інтерес до цього жанру. Зростаюча кількість головоломкових ігор, які випускаються на ринок, свідчить про збільшення попиту на них. Гравці виявляють інтерес до складних головоломок, які вимагають від них креативності, логічного мислення та вирішення проблем. Крім того, головоломкові ігри часто приваблюють гравців своєю розважальністю та здатністю випробовувати свої здібності та навички.

Враховуючи ці особливості і популярність головоломкових ігор, розробка 3D головоломкової гри на платформі Unity може бути захоплюючою та успішною ідеєю.

Дослідження ринкових тенденцій і попиту на головоломкові ігри:

Дослідження ринкових тенденцій і попиту на головоломкові ігри є важливим етапом перед розробкою 3D головоломкової гри на платформі Unity. Воно дозволяє зрозуміти потреби та вимоги цільової аудиторії, а також здобути інформацію про конкурентні головоломкові ігри на ринку. Основні аспекти дослідження включають наступне:

1. Аналіз ринку головоломкових ігор: Дослідження популярних ігрових платформ та магазинів, де пропонуються головоломкові ігри, таких як Steam, App Store, Google Play і т.д. Визначення кількості головоломкових ігор на ринку та їхньої розподілу за жанрами та платформами.
2. Аналіз конкуренції: Вивчення існуючих головоломкових ігор у 3D форматі, їхніх особливостей, геймплею та популярності серед гравців. Виявлення недоліків та прогалин на ринку, які можуть бути використані для створення унікальної гри.
3. Цільова аудиторія: Визначення характеристик цільової аудиторії для головоломкових ігор. Вивчення їхніх інтересів, вподобань, вікових груп,

демографічних характеристик та геймерських звичок. Це допоможе сформувати геймплей та дизайн, які будуть приваблювати цільову аудиторію.

4. Популярні тренди: Виявлення актуальних трендів у головоломкових іграх, які привертають увагу гравців. Це можуть бути нові механіки геймплею, використання віртуальної реальності або розширеної реальності, мультиплеєрні режими тощо. Врахування таких трендів допоможе створити сучасну та привабливу гру.
5. Маркетингові можливості: Вивчення можливостей для просування та монетизації головоломкової гри. Аналіз рекламних кампаній, спонсорських угод та інших маркетингових стратегій, які можуть підвищити популярність та дохідність гри.

Дослідження ринкових тенденцій і попиту на головоломкові ігри надає важливу інформацію для успішного планування та розробки 3D головоломкової гри. Врахування вимог аудиторії та розуміння конкурентного середовища допоможуть створити захоплюючу та вдалий продукт, який зможе привернути увагу гравців.

1.2 Опис предметної області

Для розробки головоломкових ігор у 3D графіці в середовищі розробки Unity і мові програмування C# існує ряд інструментів та ресурсів, які можуть бути використані для ефективної реалізації проекту. Огляд і доступ до цих ресурсів може значно сприяти розробці ігрового додатку.

1. Розширення Unity: Unity має широкий вибір розширень, які можуть полегшити розробку головоломкових ігор. Наприклад, "Playmaker" - візуальний інструмент для створення ігрової логіки без необхідності в програмуванні, "ProBuilder" - для швидкого моделювання геометрії прямо в Unity, "DOTween" - для анімації об'єктів та багато інших.
2. Онлайн-ресурси: Існують численні онлайн-ресурси, де можна знайти безкоштовні та платні готові моделі, текстури, звукові ефекти та інші ресурси, що можуть бути використані для створення головоломок у 3D графіці.

Наприклад, Unity Asset Store, TurboSquid, Sketchfab тощо.

3. Документація та туторіали: Unity надає широкий набір документації, туторіалів, відеоуроків та форумів, які допомагають розробникам отримати необхідні знання і навички для реалізації головоломкових ігор. Офіційна документація Unity, Unity Learn, YouTube канали та спільноти розробників є цінним джерелом інформації.
4. Інструменти розробки: Unity має вбудовані інструменти для розробки ігор, такі як інспектор об'єктів, графічний редактор анімацій, система частинок, фізичний двигун та інші. Використання цих інструментів дозволяє розробникам створювати складні головоломки з різноманітними механіками та ефектами.
5. Мова програмування C#: Unity використовує мову програмування C# для розробки ігрової логіки та взаємодії об'єктів. Знання C# дозволяє розробникам створювати складні логічні завдання, маніпулювати об'єктами та впроваджувати особливості головоломкових ігор.

Аналіз інформаційного забезпечення предметної області допомагає розробникам краще розуміти доступні ресурси та інструменти для реалізації головоломкових ігор у 3D графіці. Знання про існуючі можливості дозволяє ефективно використовувати ці ресурси та створювати захоплюючі та якісні ігрові додатки.

Вивчення різних методів реалізації головоломок у 3D графіці

Для успішної реалізації головоломок у 3D графіці існує ряд методів, які дозволяють створити цікаві та викликаючі грати виклики головоломки. Детальний аналіз цих методів допомагає зрозуміти їх переваги та недоліки та вибрати найбільш підходящий підхід для даної дипломної роботи. Нижче представлені деякі з розглянутих методів:

1. Модель фізики: Використання фізичної моделі дозволяє симулювати реалістичну поведінку об'єктів, їх рух та взаємодію. Застосування фізики дозволяє створити головоломку зі складними механіками, де об'єкти взаємодіють один з одним згідно з фізичними законами.
2. Логічне програмування: Використання логічного програмування дозволяє створити складні логічні взаємодії та правила для головоломки. Цей підхід

дозволяє контролювати послідовність подій та умови вирішення головоломки.

3. Анімація та рух: Використання анімації та руху об'єктів дозволяє створити живу та динамічну головоломку. Це може включати анімовані об'єкти, рухомі платформи або механізми, які гравцеві потрібно керувати для досягнення цілей головоломки.
4. Використання шейдерів: Використання спеціальних шейдерів дозволяє створити візуальні ефекти та покращити реалізм головоломки. Шейдери дозволяють контролювати освітлення, тіні, текстури та інші аспекти візуального представлення головоломки.
5. Графічні ефекти: Додавання різних графічних ефектів, таких як частинки, блискітки, туман та інші, допомагає підвищити естетичний вигляд головоломки та створити привабливу графіку для гравців.

Проведення детального дослідження цих методів дозволить вибрати найбільш ефективні та підходящі рішення для реалізації головоломок у 3D графіці у контекст.

Аналіз наявних інструментів та технологій для створення графіки, фізики та логіки головоломкової гри:

У даному розділі буде проведений аналіз наявних інструментів та технологій, що можуть бути використані для розробки графіки, фізики та логіки головоломкової гри у 3D форматі.

1. Інструменти для розробки графіки:
    - Unity: популярне середовище розробки, яке надає широкі можливості для створення графіки, включаючи моделювання об'єктів, текстурування, освітлення та анімацію.
    - Unreal Engine: потужний інструмент для розробки графіки, який надає високу якість візуалізації, фотореалістичність та широкий функціонал для створення головоломкових ефектів.
2. Інструменти для фізичної моделювання:
    - PhysX: фізичний двигун, який дозволяє реалістично моделювати фізичну взаємодію об'єктів, таку як колізії, рух та сили.
    - Bullet Physics: відкритий фізичний двигун, який надає широкий спектр

фізичних ефектів та можливостей для моделювання реального руху об'єктів.

3. Інструменти для розробки логіки головоломок:
    - Unity Playmaker: візуальний інструмент для розробки логіки гри за допомогою графічного інтерфейсу. Цей інструмент дозволяє швидко створювати складну логіку головоломок без програмування.
    - Unreal Blueprints: візуальний скриптовий інструмент, який дозволяє створювати логіку гри шляхом з'єднання блоків. Цей інструмент підходить для швидкого прототипування та розробки головоломкової логіки.

Огляд наявних ресурсів, документації та публікацій

У цьому розділі буде проведений огляд наявних ресурсів, документації та публікацій, які можуть бути корисними для розробки головоломкової гри. Важливо провести дослідження та зібрати відповідні джерела інформації, які допоможуть у вирішенні поставлених завдань.

1. Онлайн-ресурси:
    - Форуми та спільноти розробників ігор: дослідження форумів та спільнот, пов'язаних з розробкою ігор, може дати доступ до цінної інформації, досвіду та порад від інших розробників.
    - Офіційні веб-сайти інструментів та технологій: перегляд документації, статей, відеоуроків та прикладів на офіційних веб-сайтах різних інструментів та технологій може забезпечити глибоке розуміння їх можливостей та використання.
2. Навчальні матеріали:
    - Книги та публікації: вивчення книг та наукових публікацій, присвячених розробці ігор, графіки, фізики та логіки, може допомогти отримати теоретичні знання та підходи, які можуть бути застосовані у розробці головоломкової гри.
    - Онлайн-курси та відеоуроки: використання онлайн-курсів та відеоуроків, присвячених розробці ігор, може допомогти вивчити практичні навички

та засвоїти конкретні техніки розробки головоломкової гри.

3. Наукові дослідження:
    - Академічні статті та конференції: огляд академічних досліджень та статей, присвячених графіці, фізиці та логіці головоломкових ігор, може дати доступ до нових ідей, підходів та технологій, що використовуються у даній області.

Загальний аналіз цих ресурсів та джерел інформації допоможе зрозуміти поточний стан предметної області та вибрати найбільш корисні та релевантні ресурси для подальшої розробки головоломкової гри.

1.3 Постановка задачі

Мета дипломної роботи полягає в розробці 3D гри у жанрі головоломка на мові програмування C# у середовищі розробки Unity. Головна мета полягає у створенні захоплюючої головоломкової гри, яка надасть гравцям цікаві виклики та задачі, а також забезпечить візуально привабливе середовище.

Для досягнення цієї мети поставлені такі основні цілі:

1. Розробка геймплею головоломкової гри: розробити цікаві та різноманітні рівні з головоломками, які вимагатимуть від гравця логічного мислення та розв'язання складних завдань.
2. Створення 3D графіки: розробити візуально привабливе середовище гри з використанням 3D графіки, яке буде сприяти поглибленню іммерсії гравця.
3. Реалізація фізичної моделі: розробити фізичну модель, що дозволить гравцям взаємодіяти з об'єктами гри та розв'язувати головоломки
4. Імплементація логіки гри: розробити логіку гри, включаючи управління головоломками

Загальна мета та цілі дипломної роботи визначають основний напрямок роботи та визначають основні завдання, які необхідно виконати для успішного виконання проекту.

Формулювання конкретних задач, які потрібно вирішити для досягнення поставленої мети:

Для досягнення поставленої мети розробки 3D гри у жанрі головоломка на мові

програмування C# у середовищі розробки Unity, необхідно вирішити наступні конкретні задачі:

1. Дослідити різні типи головоломок у 3D графіці: Ознайомитись з різними видами головоломок, такими як лабіринти, головоломки з рухомими блоками, головоломки з фізикою тощо, та обрати підходящі для реалізації у власній грі.
2. Розробити геймплей: Створити систему рівнів та головоломок з належною складністю та прогресивністю. Розробити механіку взаємодії з об'єктами гри.
3. Вивчити інструменти та технології Unity: Поглиблено вивчити функціонал та можливості середовища розробки Unity, включаючи роботу з 3D моделями, анімаціями, фізикою, освітленням та іншими елементами гри.
4. Реалізувати графічну частину гри: Створити 3D моделі об'єктів гри, дизайн рівнів, стилістику та ефекти візуалізації для забезпечення привабливості гри.
5. Реалізувати фізичну модель: Розробити систему фізики, яка забезпечить реалістичне поведінка об'єктів у грі, включаючи рух, колізії, гравітацію тощо.
6. Реалізувати логіку гри: Розробити логіку головоломок, включаючи управління гравцем, перевірку розв'язання головоломки, систему очків та досягнень.
7. Провести тестування та оптимізацію: Перевірити та виправити можливі помилки, провести тестування гри на різних пристроях та оптимізувати її продуктивність для забезпечення гладкого та стабільного геймплею.
8. Документування та підготовка проекту: Створити документацію, яка описує архітектуру, функціонал та інструкції по використанню гри. Підготувати проект для захисту дипломної роботи.

Ці задачі визначають основні кроки, які потрібно виконати для успішної реалізації дипломної роботи і досягнення поставленої мети.

Висновок: У даному розділі було проведено аналіз основних вимог до гри, таких як геймплей, рівні складності, механіки гри, графіка та звуковий супровід. Було визначено, що цільовою платформою для розробки гри буде середовище розробки Unity, оскільки воно надає необхідні інструменти та можливості для створення 3D гри у жанрі головоломки.

Також було розглянуто вибір засобів розробки, де було вирішено використовувати

мову програмування C# у поєднанні з інтегрованою середою розробки Microsoft Visual Studio. Цей вибір обумовлений підтримкою Unity для мови C# та зручним інтерфейсом роботи з Microsoft Visual Studio.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Розробка структури системи

Гра повинна виконувати наступні функції:

- Рух червоного куба вперед та назад за допомогою натискання заданих клавіш на клавіатурі.

- Рух зеленого куба вліво та вправо за допомогою натискання заданих клавіш на клавіатурі відмінних від клавіш управління червоного куба.

- Рух рожевого куба разом з червоним кубом вперед та назад за допомогою натискання заданих клавіш на клавіатурі.

- Перехід на наступний рівень при досяганні фінішу червоним кубом.

- Рух червоного куба вліво та вправо за умови що зелений куб штовхає його в один із напрямків.

- Неможливість зрушити з місця білий куб.

- Можливість натискання кнопки червоним, зеленим та рожевим кубом.

- Зміна стану жовтого куба з непрозорого на прозорий та навпаки при натисканні кнопки.

- Зміна стану кнопки з непрозорого на прозорий за умови що кнопка була натиснута.

- Зміна стану кнопки з прозорого на непрозорий за умови що була натиснута інша кнопка.

- Неможливість натиснути кнопку, якщо в прозорому жовтому кубі знаходиться інший куб.

- Зміна гучності музики в головному меню у вкладці Settings.

- Можливість зайти на будь-який рівень, за умови проходження попереднього в головному меню у вкладці Levels.

- Збереження прогресу проходження рівнів після перезавантаження гри.

- Можливість анулювати прогрес проходження рівнів натисканням кнопки Reset.

Основна ціль гри полягає у тому, щоб червоний куб дійшов до фінішу. Червоним кубом можна керувати за допомогою натискання клавіш на клавіатурі. Клавіші дозволяють йому рухатися вперед та назад. На його шляху перешкоди у вигляді інших кубів з якими можна взаємодіяти за допомогою натисканням клавіш на клавіатурі або взаємодією з іншими об’єктами на рівні. Коли червоний куб досягає фінішу гравець переходить на наступний рівень. Поступово на рівнях будуть з’являтися нові перешкоди та механіки.

Усього на у грі є 5 різних кубів та кнопка:

-Червоний куб – основний куб (гравець).

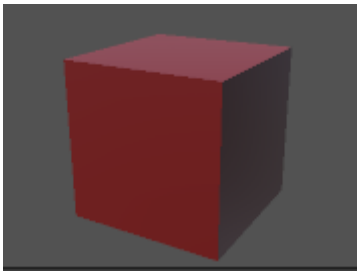


Рисунок 2.1 – червоний куб

-Зелений куб – перешкода, яка рухається натисканням клавіш на клавіатурі.

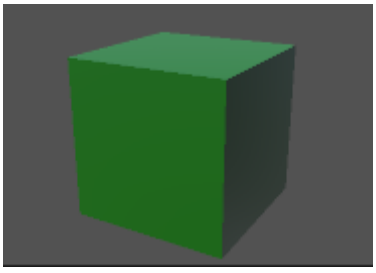


Рисунок 2.2 – зелений куб

-Рожевий куб – перешкода, яка рухається разом х червоним кубом натисканням клавіш на клавіатурі.

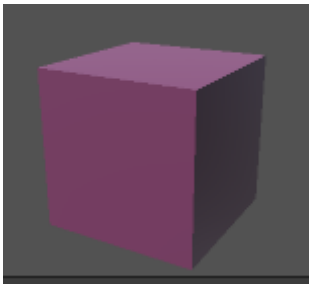


Рисунок 2.3 – рожевий куб

-Білий куб – перешкода яка не рухається.

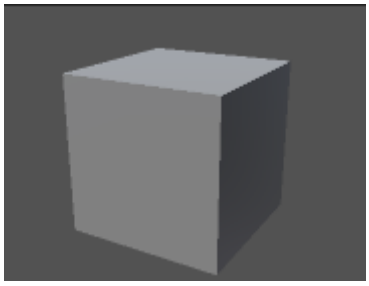


Рисунок 2.4 – білий куб

-Жовтий куб – перешкода яка може бути прозорою або непрозорою.

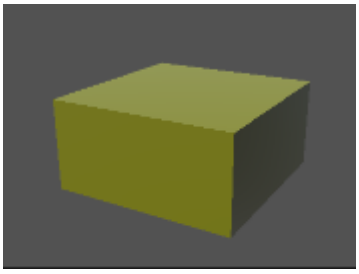
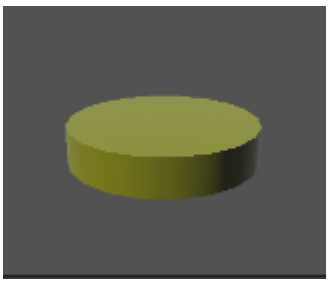


Рисунок 2.5 жовтий непрозорий та прозорий куб

-Кнопка – змінює стан жовтого куба з непрозорого на непрозорий та навпаки. Після натискання змінює свій стан на непрозорий та не може бути натиснута.



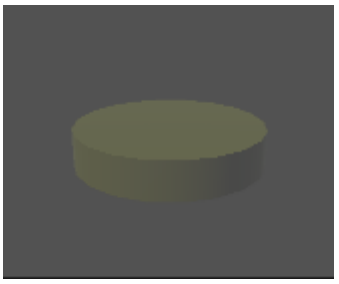
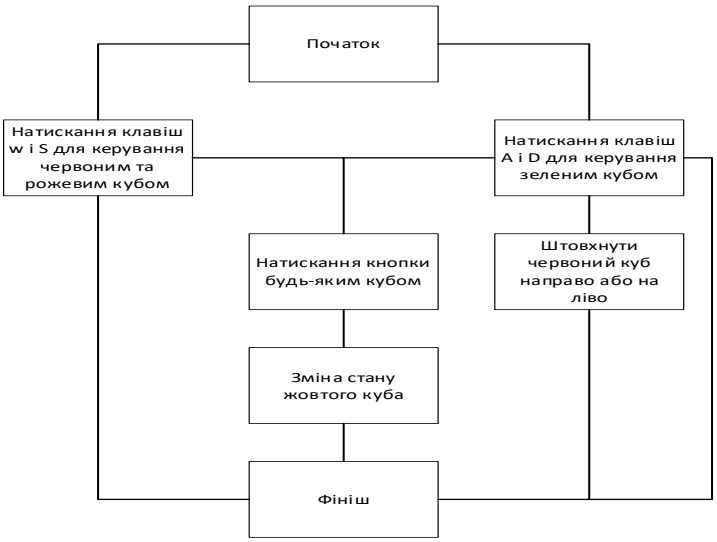


Рисунок 2.6 – непрозора та прозора кнопка

Блок схема 2.1 – Взаємодія об’єктів



Головне меню гри має 4 кнопки з якими можна взаємодіяти натискання лівої клавіші миші:

-Кнопка Play – запускає перший рівень гри.

-Кнопка Levels – вибір рівня на який хочеш потрапити. Нові рівні відкриваються за умови проходження попереднього. Натискання кнопки Reset анулює прогрес, а натискання кнопки Back повертає на головне меню.

-Кнопка Settings – можливість налаштування гучності музики. Натискання кнопки Back повертає на головне меню.

-Кнопка Quit – вихід із гри.

Рисунок. 2.7 – головне меню гри

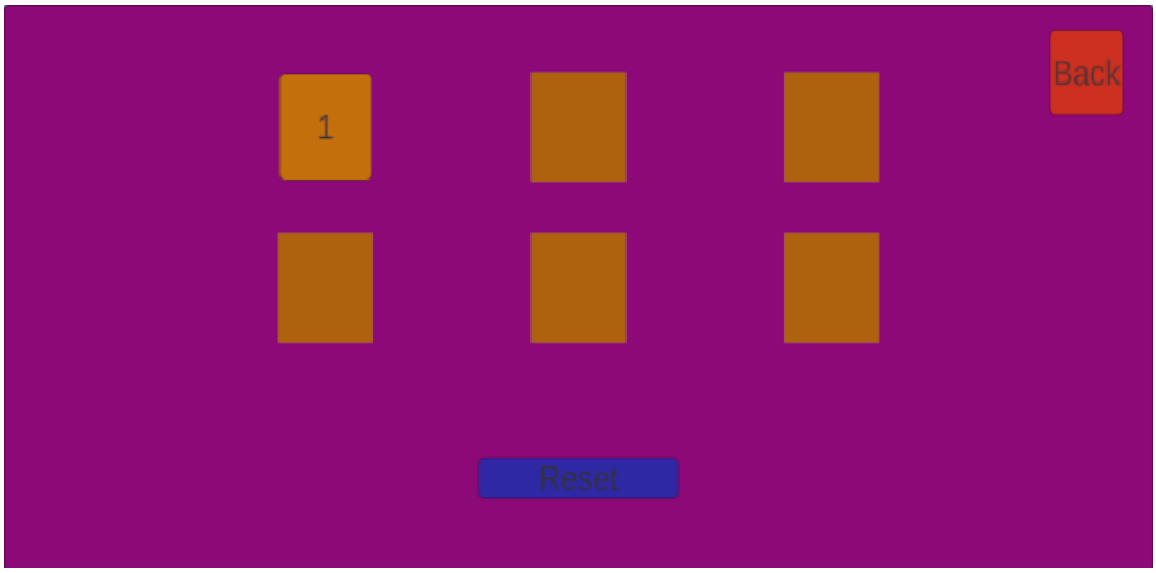


Рисунок 2.8 – меню вибору рівня (Levels)

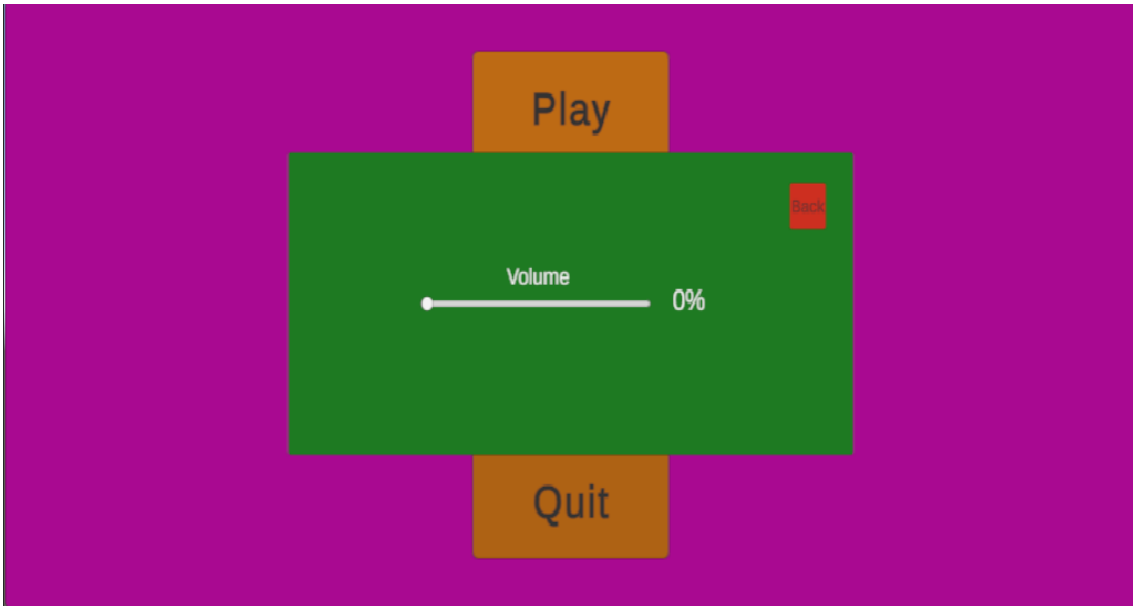
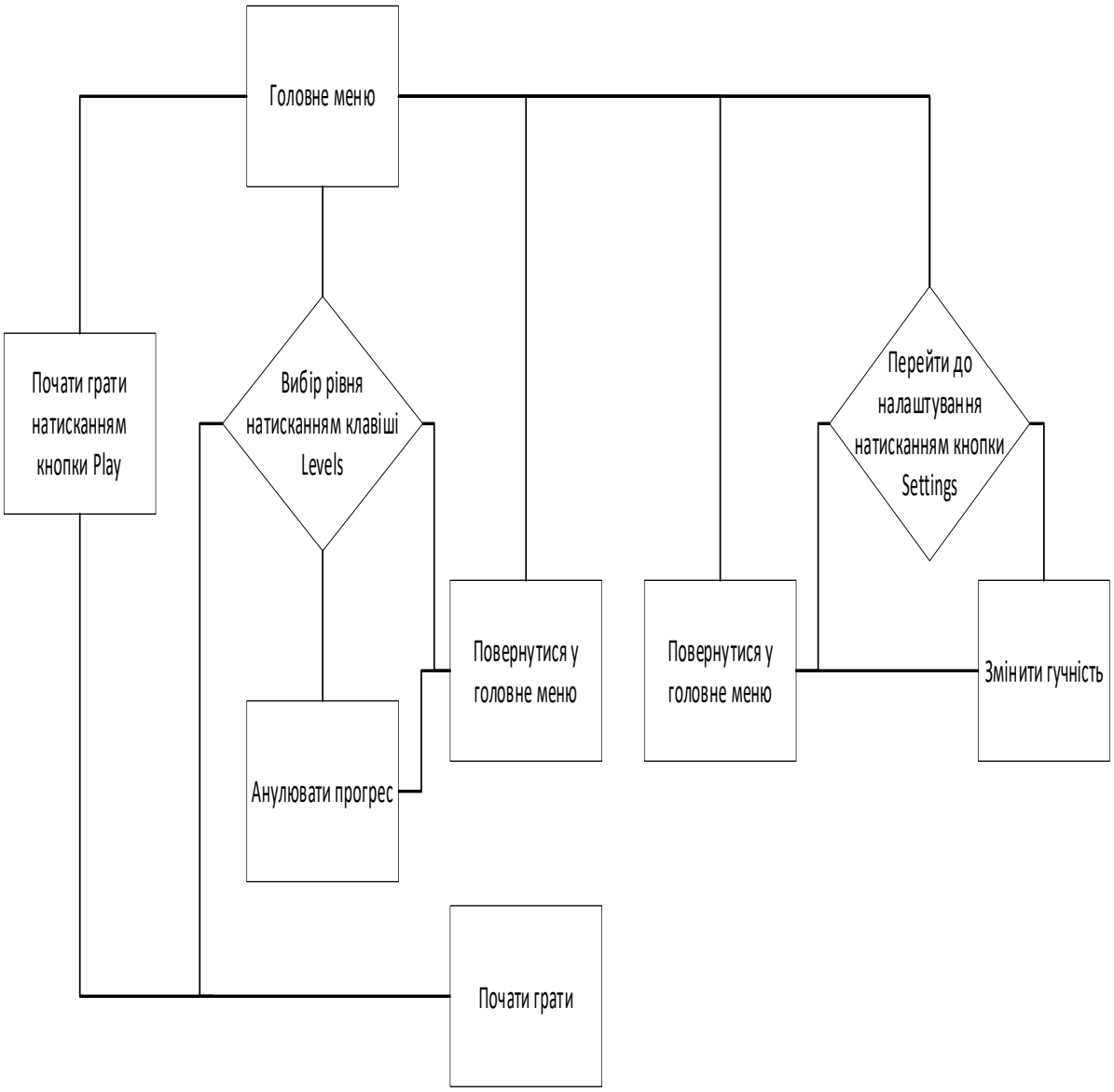


Рисунок 2.9 – налаштування гучності музики

Блок схема 2.2 – головне меню



Таблиці опису основних функцій:

Табл.2.1 - Таблиця опису скрипта – Player.cs

Назва функції	**Параметри**	**Призначення**
FixedUpdate	()	Ця частина кода відповідає за обробку руху об’єкта гравця відповідно до введених клавіш та управління переходами між сценами

OnTriggerEnter	(Collider other)	Вона відповідає за обробку подій, які виникають при зіткненні жовтого та іншого куба у грі
OnTriggerExit	(Collider other)	Вона відповідає за обробку подій, які виникають при виході куба з зони жовтого куба у грі

Табл.2.2 - Таблиця опису скрипта –Scenes.cs

Назва функції	**Параметри**	**Призначення**
Awake	()	використовується для підписки на подію завантаження сцени та встановлення обробника цієї події
Start	()	Вона встановлює значення змінної level за допомогою PlayerPrefs.GetInt("level", level), що дозволяє отримати збережене значення рівня гри з попередніх запусків.
OnSceneLoaded	(Scene scene, LoadSceneMode mode)	У тілі функції, змінна currentSceneIndex отримує індекс поточно завантаженої сцени. Потім перевіряється умова, де порівнюється значення змінної level з поточним індексом сцени. Якщо значення level менше

		поточного індексу, то значення level оновлюється на поточний індекс сцени, і це нове значення зберігається за допомогою PlayerPrefs.SetInt("level", level).
ChangeScenes	(int numberScenes)	Функція ChangeScenes перевіряє переданий параметр numberScenes, який є індексом сцени в грі. Якщо індекс відповідає існуючій сцені, функція завантажує цю сцену.
ResetProgress	()	Функція ResetProgress видаляє збережені дані про прогрес гри і повертає його в початковий стан

Табл.2.3 - Таблиця опису скрипта –Activator.cs

Назва функції	**Параметри**	**Призначення**
OnTriggerEnter	(Collider other)	Функція OnTriggerEnter(Collider other) перевіряє, чи можна натиснути на активатор (button), а потім перевіряє, чи об'єкт, який потрапив в тригер, має тег

		"Cube" або "Player". Якщо умови виконуються, функція змінює матеріали та колайдери об'єктів в групах та активаторі.

Табл.2.4 - Таблиця опису скрипта –Sound.cs

Назва функції	**Параметри**	**Призначення**
Awake	()	Функція Awake() перевіряє збережений параметр гучності звуку в PlayerPrefs. Якщо параметр існує, вона встановлює гучність звуку та оновлює слайдер. Якщо параметр не існує, вона встановлює значення за замовчуванням і зберігає його.
LateUpdate	()	Функція LateUpdate() оновлює гучність звуку на основі положення слайдера. Вона перевіряє наявність слайдера, оновлює гучність та відображає відсоткове значення у текстовому полі.

Табл.2.5 - Таблиця опису скрипта –BCInstance.cs

Назва функції	**Параметри**	**Призначення**
Awake	()	Функція Awake() перевіряє наявність об'єкта з тегом

		createTag. Якщо об'єкт існує, поточний об'єкт знищується. Якщо об'єкт не знайдений, поточному об'єкту присвоюється тег createTag, і він залишається активним під час зміни сцен.

2.2 Вибір засобів розробки

Ігровий рушій Unity. Після проведення аналізу різних ігрових рушіїв, я прийшов до висновку, що для розробки мого дипломного проекту найбільш відповідним варіантом буде використання ігрового рушія Unity. Unity - це потужна та популярна платформа розробки ігор, яка надає розширені можливості для створення 3D графіки та реалістичного геймплею.

Однією з переваг Unity є його підтримка графічних бібліотек OpenGL та DirectX, що дозволяє використовувати різноманітні графічні можливості для створення візуально привабливих ігор. Крім того, Unity має зручний інтерфейс з використанням методу Drag&Drop, що дозволяє швидко та легко налаштовувати ігрові об'єкти та їх взаємодію прямо в редакторі.

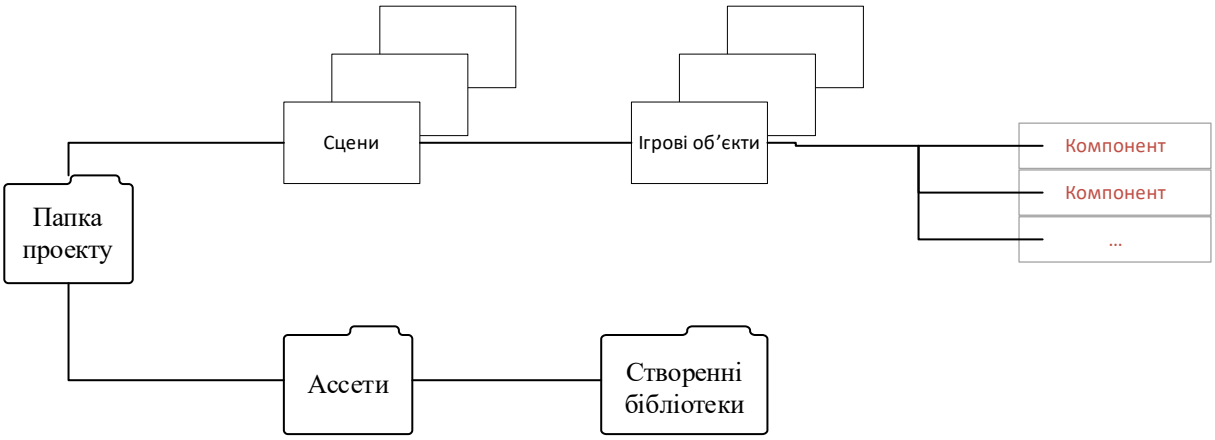
У структурі Unity кожна гра складається з окремих сцен - файлів, що містять ігрові об'єкти, налаштування та скрипти. Об'єкти в Unity можуть мати різні компоненти, які визначають їх поведінку та взаємодію з гравцем. Важливими компонентами є Transform, що зберігає інформацію про положення та розмір об'єкту, та Mesh Renderer, який відповідає за відображення моделі об'єкта на екрані.

Окрім графіки, Unity надає можливості для редагування шейдерів та створення анімацій. Також, Unity має власний формат unityassets, який дозволяє запаковувати моделі, скрипти, матеріали та звуки в один файл для зручності передачі та обміну ресурсами між розробниками. На основі цього формату функціонує Unity Asset Store,

де розробники можуть публікувати та продавати свої ассети.

Unity підтримує різні мови програмування, зокрема C# та JavaScript, для написання скриптів та розробки логіки гри. Крім того, Unity має спеціальну надбудову Shader Lab, що дозволяє використовувати декілька шейдерних мов для створення реалістичної графіки.

З урахуванням всіх цих факторів та можливостей, я обрав Unity як основний інструмент для розробки моєї дипломної роботи у жанрі 3D головоломки з використанням мови програмування C# у середовищі Unity. Цей вибір надає мені широкі можливості для створення цікавого геймплею, реалістичної графіки та звукового супроводу, а також забезпечує підтримку для моєї цільової платформи розробки.



Блок схема 2.3 – Структура проекту на Unity

Мова програмування C# є універсальною об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft спільно з платформою .NET. Вона має синтаксис, схожий на мови з роду C, такі як C++ та Java. C# застосовується для розробки різноманітного програмного забезпечення, включаючи веб-сайти, веб-додатки, офісні програми, десктопні та мобільні додатки, ігри та інші проекти. Важливо зазначити, що мова програмування C# підтримується ігровим рушієм Unity.

C# має кілька переваг, зокрема:

- Компонентно-орієнтований підхід до програмування, що сприяє гнучкості та повторному використанню коду.

- Мова прагне до справжньої об'єктно-орієнтованості, де кожен об'єкт є сутністю.
- Уніфікована система типізації.
- Більш висока безпека коду порівняно з C++.
- Розширені можливості подійно-орієнтованого програмування.

Проте, C# також має деякі недоліки, такі як:

- Відносно низька продуктивність.
- Обмежена кількість концептуальних інновацій.
- Складний синтаксис.

Отже, мова програмування C# є потужним інструментом для розробки програмного забезпечення, зокрема в галузі ігрової розробки, проте вона має свої переваги та недоліки, які слід враховувати при виборі її для проекту.

Microsoft Visual Studio є одним з основних інтегрованих середовищ розробки (IDE), які можуть бути використані для створення 3D ігор у жанрі головоломка на мові програмування C# в середовищі розробки Unity.

Visual Studio надає потужні інструменти для розробки гри, включаючи редактор коду з автодоповненням, підсвічуванням синтаксису та функціями рефакторингу. Ви можете легко писати, відлагоджувати та тестувати свій код на мові програмування C# безпосередньо у Visual Studio.

Окрім цього, Visual Studio має розширену інтеграцію з Unity. Ви можете підключити Visual Studio до свого проекту Unity і використовувати його функціональність для створення скриптів, компіляції та налагодження гри. Він дозволяє зручно працювати з об'єктами та компонентами Unity, а також забезпечує візуальний редактор для швидкого розробки інтерфейсу користувача.

Visual Studio також підтримує систему контролю версій, що дозволяє вам відстежувати зміни в коді та співпрацювати з іншими розробниками. Ви можете легко спільно працювати над своїм проектом і керувати версіями вашого програмного забезпечення безпосередньо у Visual Studio.

Висновок: другий розділ дипломної роботи зосереджений на розробці структури системи для головоломкової гри, виборі цільової платформи та засобів

розробки, а також реалізації механіки управління та взаємодії об'єктів. Цей розділ є важливим кроком у створенні гри та підготовці до наступних етапів розробки та тестування.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 План розробки гри

Для реалізації гри був створений план розробки гри:

- Створення ігрового поля
- Створення ігрових об’єктів
- Додавання механіки управління та взаємодії об’єктів
- Додавання більшої кількості рівнів
- Створення головного меню
- Реалізація музики
- Реалізація збереження прогресу пройдених рівнів та можливість його анулювати.

3.2 Створення ігрового поля

Для створення ігрового поля було додано 6 об’єктів Cube. Перший використовується як платформа, а інші 5 – стіни.

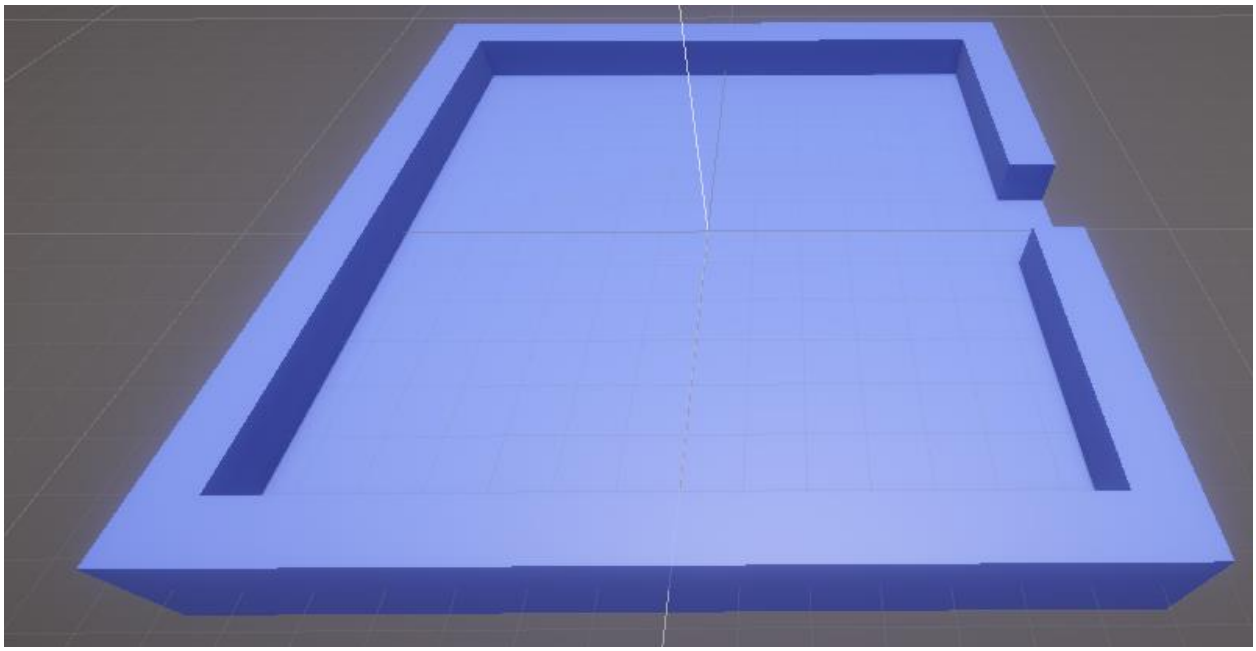


Рисунок 3.1 – ігрове поле

Для надання їм блакитного кольору був створений матеріал, якому я надав відповідний колір.

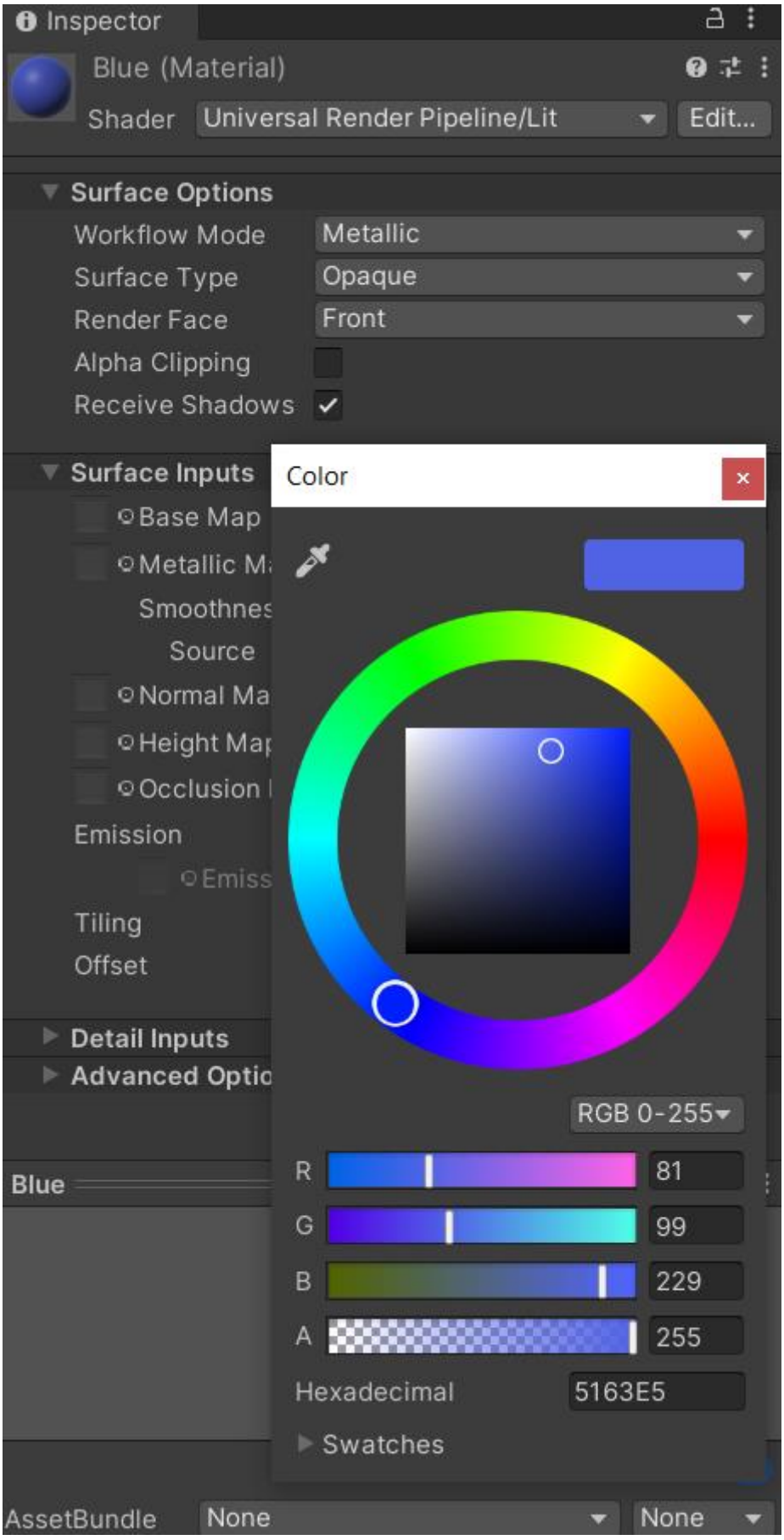


Рисунок 3.2 – матеріал синього кольору

3.3 Створення ігрових об’єктів

На цьому етапі було створено 5 кубів (рисунок 2.1 – 2.5), кнопка(рисунок 2.6) та матеріали (рисунок 3.3 – 3.8) для кожного об’єкт. Кожен куб має своє призначення та може взаємодіяти з іншими об’єктами.

Рисунок 3.3- матеріал зеленого куба

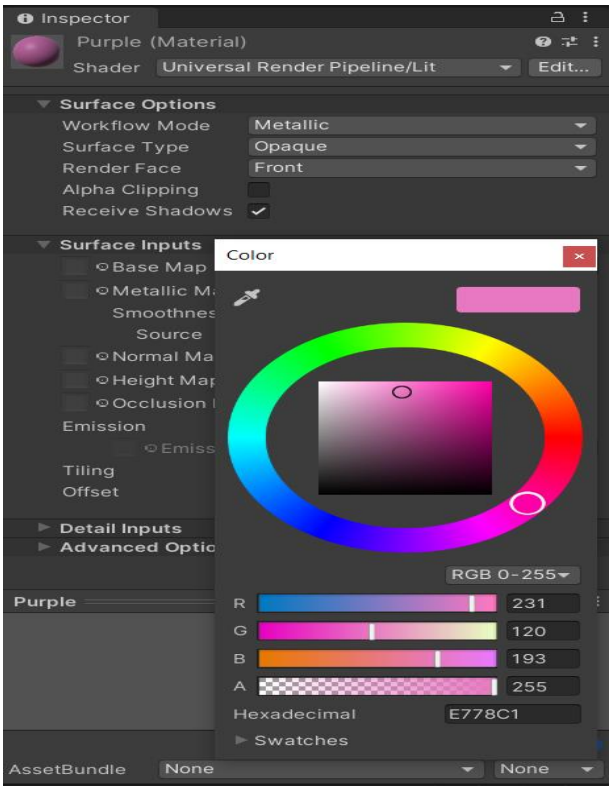
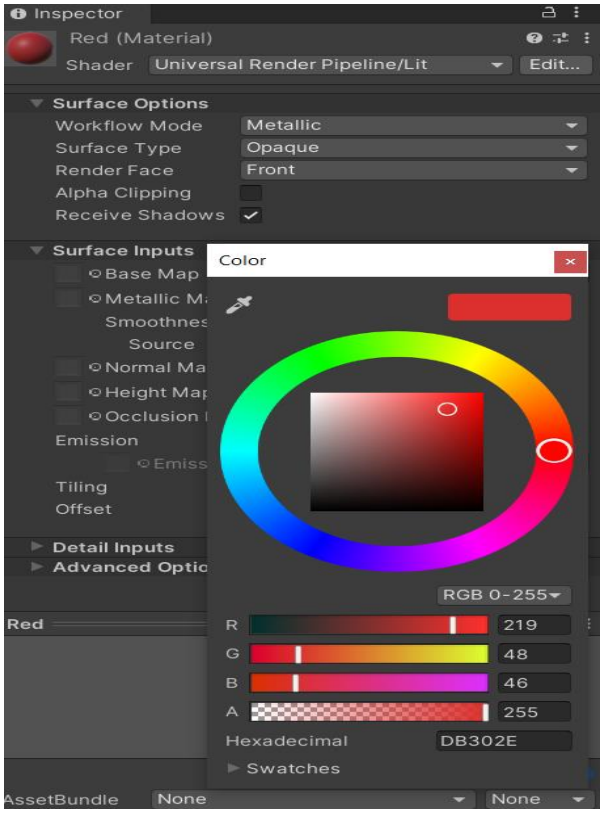


Рисунок 3.4 – матеріал рожевого куба



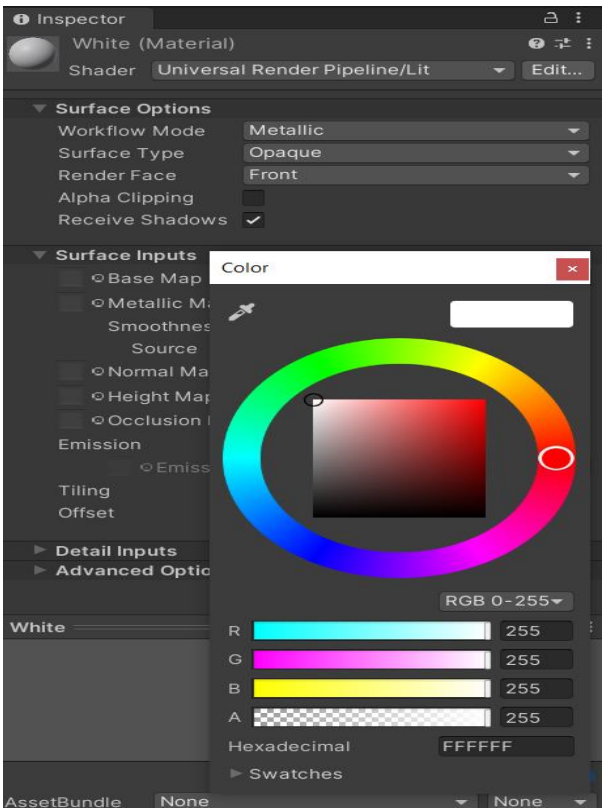
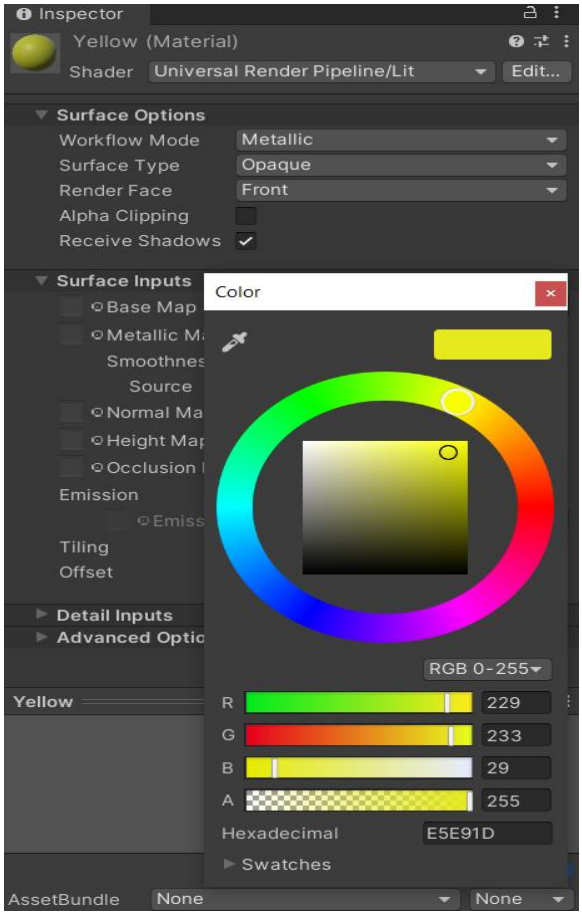


Рисунок 3.5 – матеріал червоного куба Рисунок 3.6 – матеріал білого куба



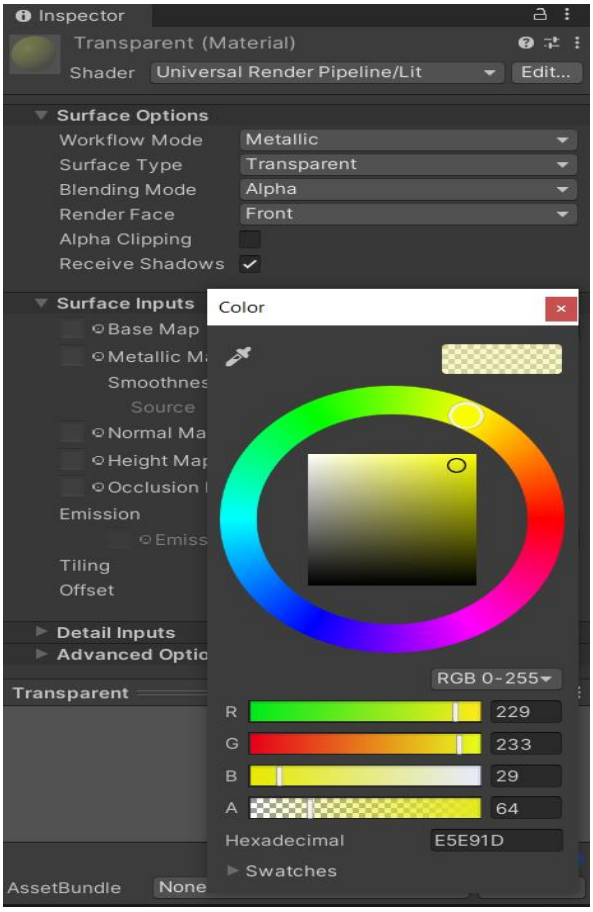


Рисунок 3.7 – матеріал жовтий Рисунок 3.8 – матеріал прозорий

3.3 Додавання механіки управління та взаємодії об’єктів

Для керування кубами було вирішено використовувати кнопки W, S, A і D. Також були додані кнопки для оновлення рівня (R), для переходу на наступний рівень (Q) та для переходу у головне меню (Escape).

Для реалізації управлянні був написаний скрипт, що став компонентом для червоного, зеленого, рожевого кубів

```
private void FixedUpdate()
{
    if (Input.GetKey(keyOne))
    {
        GetComponent<Rigidbody>().velocity += moveDirection;
    }
    if (Input.GetKey(keyTwo))
    {
        GetComponent<Rigidbody>().velocity -= moveDirection;
    }
    if (Input.GetKey(KeyCode.R))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
    if (Input.GetKey(KeyCode.Q))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    if (Input.GetKey(KeyCode.Escape))
    {
        SceneManager.LoadScene(0);
    }
}
```

Рисунок 3.9 – Частина кода скрипту Player.cs

Для того щоб кубами керувати у компоненті Player.cs потрібно встановити бажані клавіщі.

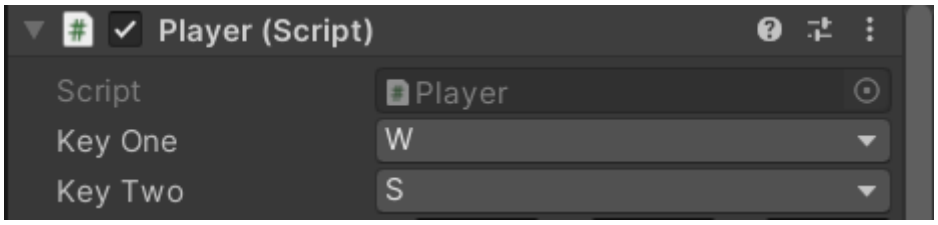


Рисунок 3.10 – компонент Player.cs

Для того щоб зелений куб неможливо було штовхнути вперед або назад в параметрі Freeze Position компонента Rigidbody у інспекторі потрібно було поставити галочку навпроти букви «Z», тим самим я заблокував можливість змінювати позицію по координаті «Z»

Рисунок 3.11 – компонент Rigidbody зеленого куба

Для рожевого куба потрібно зробити теж саме, але замість координати «Z» потрібно заморозити координату «X».

Рисунок 3.12 – компонент Rigidbody рожевого куба

Оскільки білий та жовтий куб повинен бути нерухомим для нього потрібно заморозити координати «X» та «Z».

Рисунок 3.13 - компонент Rigidbody білого та жовтого куба

Для кнопки був створений скрипт Activator.cs, який за допомогою функції OnTriggerEnter(Collider other) перевіряє чи є дотик між кнопкою з іншим об’єктом якому у інспекторі Unity був привласнений тег Cube або Player. Якщо ця умова виконується то матеріал жовтого куба змінюється з непрозорого на прозорий або навпаки.

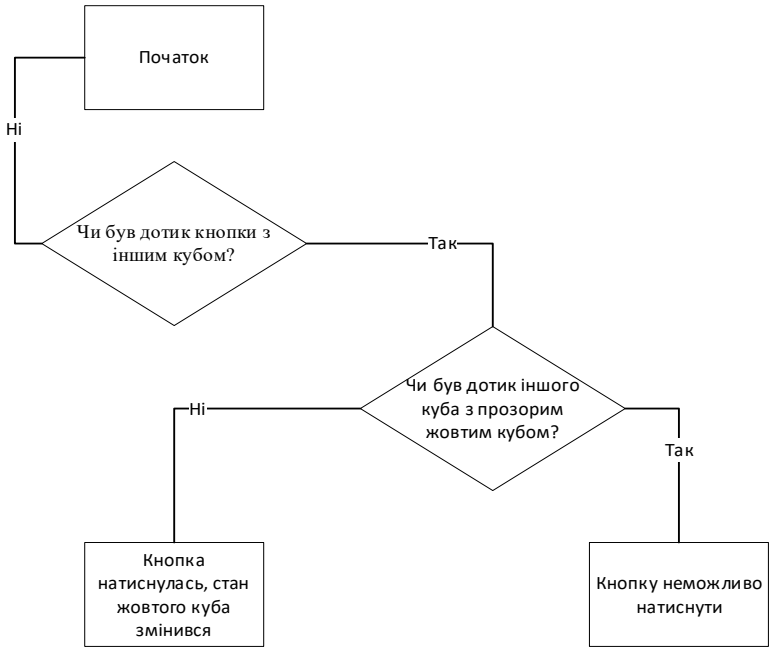
```
private void OnTriggerEnter(Collider other)
{
    if (canPush)
    {
        if (other.CompareTag("Cube") || other.CompareTag("Player"))
        {
            foreach (GameObject first in firstGroup)
            {
                first.GetComponent<Renderer>().material = normal;
                first.GetComponent<Collider>().isTrigger = false;
            }
            foreach (GameObject second in secondGroup)
            {
                second.GetComponent<Renderer>().material = transparent;
                second.GetComponent<Collider>().isTrigger = true;
            }
            GetComponent<Renderer>().material = transparent;
            button.GetComponent<Renderer>().material = normal;
            button.canPush = true;
        }
    }
}
```

Рисунок 3.14 – частина коду Activator.cs

У процесі гри ми можемо зіштовхнутися з тим, що в момент натискання кнопки один із кубів знаходиться в прозорому жовтому кубі і тоді він опиниться у жовтому кубі і не зможе рухатися. Щоб цього не трапилося була створена змінна canPush. У скрипті Player.cs було створено дві функції для присвоєння змінній canPush стану false або true. За допомогою функції OnTriggerEnter(Collider other) ми робимо перевірку чи є дотик жовтого прозорого куба з іншим і за умови якщо він є, змінній canPush надається значення false. Коли куб виходить із прозорого жовтого куба функція OnTriggerExit(Collider other) присвоює змінній canPush значення false.

```
private void OnTriggerEnter(Collider other)
{
    if (this.CompareTag("Player") && other.CompareTag("Finish"))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    if (this.CompareTag("Cube") || this.CompareTag("Cube") && other.CompareTag("Cube"))
    {
        foreach(Activator button in FindObjectsOfType<Activator>())
        {
            button.canPush = false;
        }
    }
}
private void OnTriggerExit(Collider other)
{
    if(this.CompareTag("Cube") || this.CompareTag("Cube") && other.CompareTag("Cube"))
    {
        foreach (Activator button in FindObjectsOfType<Activator>())
        {
            button.canPush = true;
        }
    }
```

Рисунок 3.15 – частина коду Player.cs



Блок схема 3.1 – алгоритм кнопки

Залишилося лише зробити перехід на наступний рівень. Для його реалізації була створена невидима стіна с тегом Finish. У скрипті player.cs функція OnTriggerEnter(Collider other) перевіряє дотик об'єкта якому привласнений тег Player з об’єктом якому привласнений тег Finish. Якщо дотик є, то відбувається перехід на наступний рівень.

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

Рисунок 3.16 – частина скрипта Player.cs

3.4 Додавання більшої кількості рівнів

Тепер коли ми маємо основні механіки потрібно зробити більше рівнів, для цього потрібно створити нову сцену натиснувши File – New Scene.

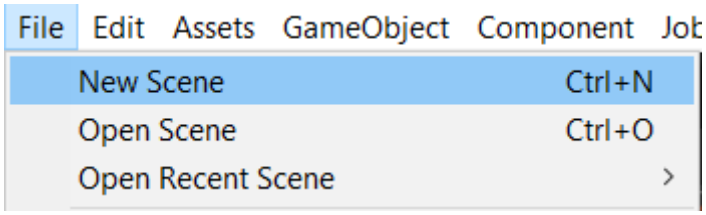


Рисунок 3.17 – створення нової сцени

Після створення нової сцени залишилося лише розмістити на ній ігрове поле та ігрові об’єкти.

3.5 Створення головного меню

Для головного меню була створена нова сцена на якій я розмістив об’єкт Plane та 4 об’єкта Button

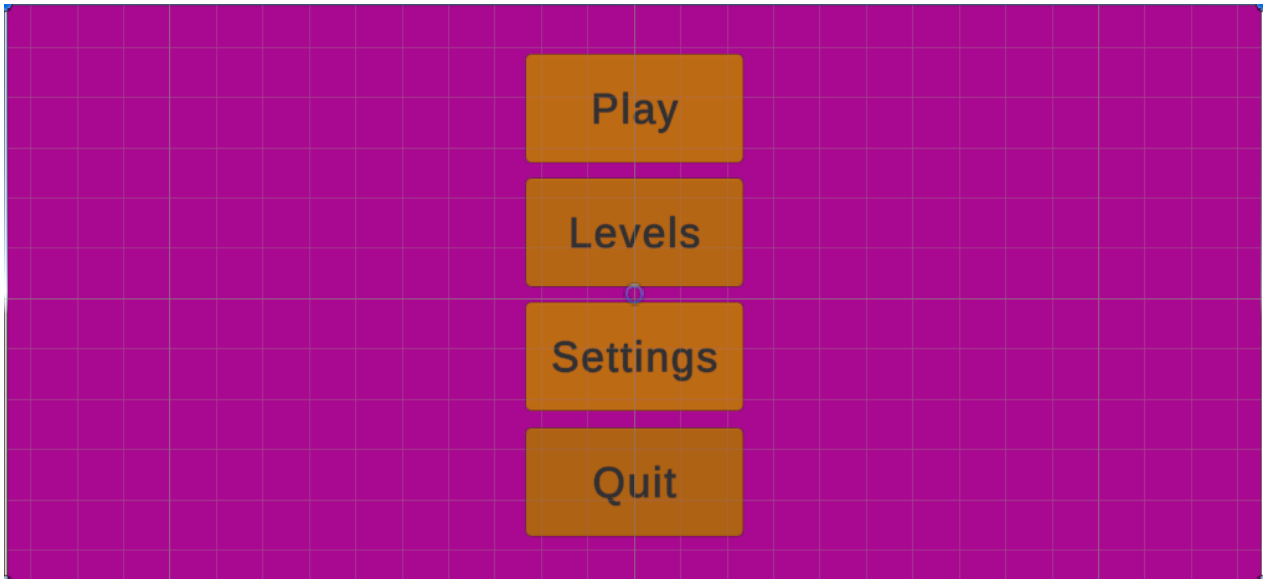


Рисунок 3.18 – головне меню

Для меню Level потрібно створити ще один об’єкт Plane, кнопку яка буде відповідати за повернення у головне меню, стільки кнопок скільки у нас рівнів та кнопку Reset яка буде анулювати прогрес рівнів. Усі кнопки повинні бути дочірніми до об’єкта Plane.

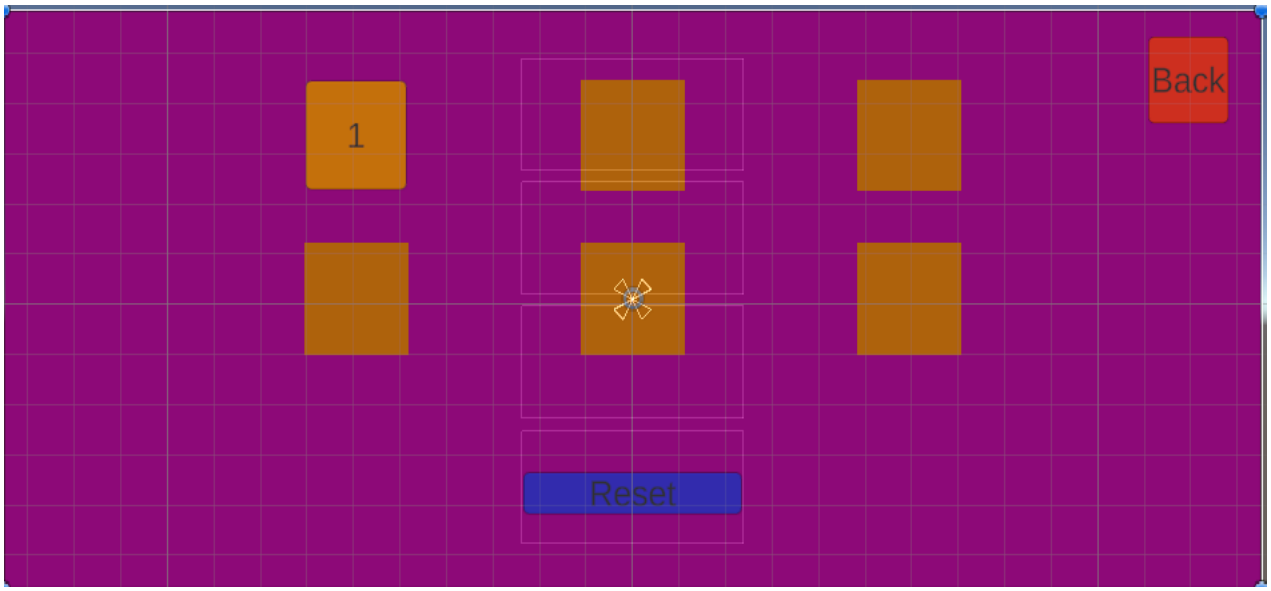


Рисунок 3.19 – меню Levels

Після створення меню level його потрібно сховати знявши галочку в інспекторі.

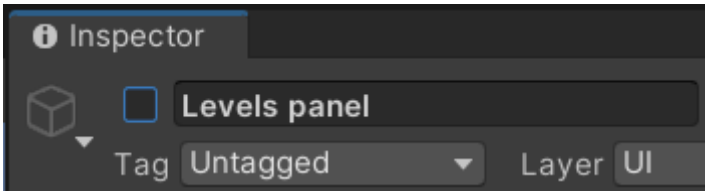


Рисунок 3.20 – інспектор об’єкта Plane

Для того щоб відкрити меню levels в інспекторі кнопки levels в параметрі OnClick налаштовуємо цю дію.

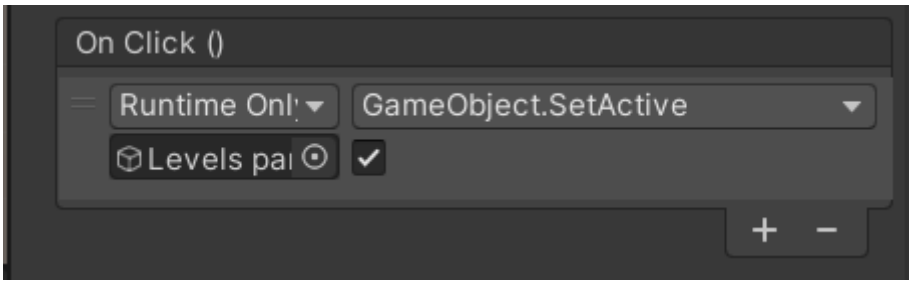


Рисунок 3.21 – інспектор кнопки Levels

Для закриття меню levels в інспекторі кнопки back робимо тіж самі налаштування але галочку потрібно вимкнути.

Рисунок 3.22 – інспектор кнопки Back

Тепер напишемо скрипт для переходу на обраний нами рівень в меню Levels.

```
public void ChangeScenes(int numberScenes)
{
    if (numberScenes >= 0 && numberScenes < SceneManager.sceneCountInBuildSettings)
    {
        SceneManager.LoadScene(numberScenes);
    }
    else
    {
        Debug.LogWarning("Scene with index " + numberScenes + " does not exist.");
    }
}
```

Рисунок 3.23 – частина коду скрипта Scenes.cs

Після написання скрипту створюємо пустий об’єкт і додаємо на нього цей скрипт. На кнопках рівнів в інспекторі параметра OnClick робимо такі налаштування де 1 – це індекс сцени бажаного рівня. Для кожної кнопки змінюємо це значення в залежності від того, на який рівень ми хочемо потрапити.

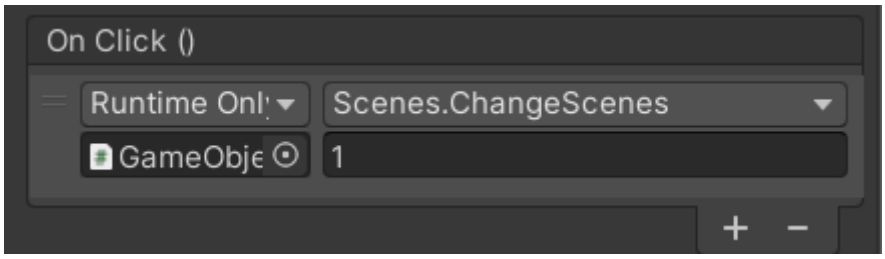


Рисунок 3.24 – інспектор кнопки вибору рівня

Тепер робимо теж саме для меню Settings, але додаємо лише одну кнопку, об’єкт Skider та Text.

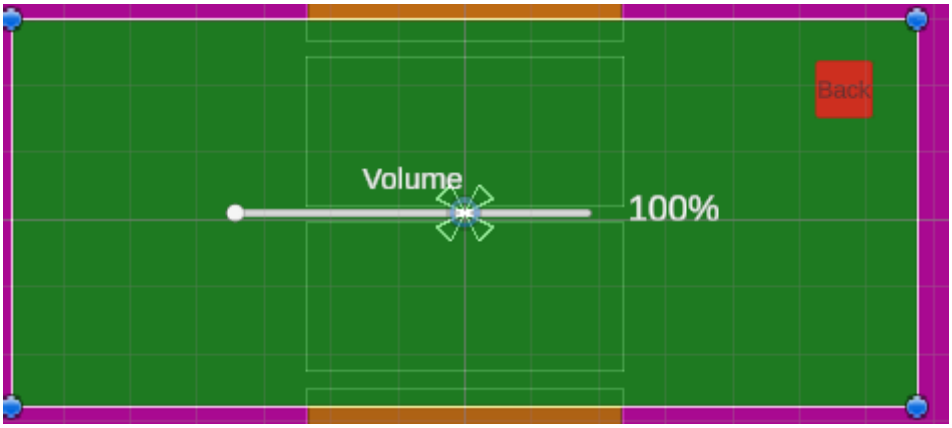


Рисунок 3.25 – меню Settings

3.6 Реалізація музики

Для цього об'єкт Audio Source. В його інспекторі до компонента Audio Source додаємо бажану музику.

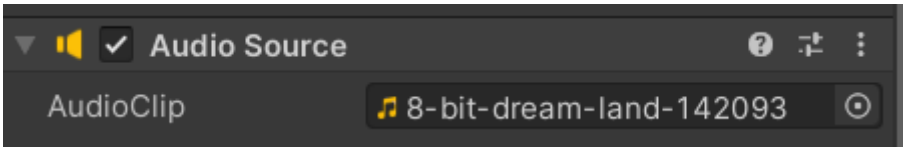


Рисунок 3.26 – інспектор об’єкта Audio Source

Створимо два скрипта Sound.cs та BCInstance.

Перший буде відповідати за те щоб за допомогою слайдера можливо було налаштовувати гучність музики, а цифри в об’єкті Text змінювались відповідно до рівня гучності музики.

Другий скрипт відповідає за те щоб музика не зупинялась при переході на інші рівні та продовжувала грати не починаючи спочатку.

3.7 Реалізація збереження прогресу пройдених рівнів та можливість його анулювати.

Збереження прогресу буде реалізовано за допомогою PlayerPrefs. Для цього потрібно створити змінну level якій буде призначатися індекс сцени. Оскільки індекси сцен відповідають номерам рівнів, значення level буде дорівнювати номеру рівня, а PlayerPrefs буде запам’ятовувати це значення.

Для того щоб неможливо було потрапити на рівень через меню Levels якщо не був пройден попередній, я створив об’єкти image та розмістив їх поверх кнопок. Ці об’єкти будуть зникати коли гравець пройде попередній рівень.

```
private void Start()
{
    level = PlayerPrefs.GetInt("level", level);
    for (int i = 0; i < level; i++)
    {
        closedLevel[i].SetActive(false);
    }
}
```

Рисунок 3.27 – частина коду скрипта Scenes.cs

Висновок: У цій дипломній роботі була розроблена 3D гра у жанрі головоломка за допомогою мови програмування C# у середовищі розробки Unity. Робота складається з двох основних розділів, кожен з яких розглядає важливі аспекти розробки та реалізації гри.

У першому розділі було проведено аналіз вимог до гри, включаючи геймплей, рівні складності, механіки гри, графіку та звуковий супровід. Також було визначено цільову платформу та обрано засоби розробки, зокрема мову програмування C# та середовище розробки Unity, що є оптимальними для реалізації головоломкової гри.

У другому розділі була розроблена структура системи гри, включаючи компоненти, модулі та їх взаємозв'язки. Було досліджено механіку управління та взаємодії об'єктів у грі, розроблено логіку гри та впроваджено фізичну модель об'єктів. Також

було розглянуто використання Microsoft Visual Studio для зручної розробки та налагодження програмного коду.

В результаті проведених досліджень та розробок була створена функціональна та цікава 3D гра у жанрі головоломка. Реалізація геометричних загадок, механік гри та взаємодії об'єктів дозволяє гравцям відчути виклик та задоволення від розв'язування складних головоломок.