

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій

(повне найменування факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Розробка комп'ютерної гри мовою C# за допомогою

(назва теми)

середовища Unity

Виконав(ла): студент(ка) 4 курсу, групи КНТ 510 сп

Спеціальності 123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Комп'ютерна інженерія»

КНИРИК Є. О.

(ПРИЗВИЩЕ та ініціали)

Керівник КАСЬЯН К.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент КАЧАН О.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

«17» травня 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КНИРИКА Єгора Олександровича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної гри мовою C# за допомогою середовища Unity

керівник проєкту (роботи) к. т. н., доцент, КАСЬЯН Костянтин Миколайович

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від «24» березня 2023 року № 71

2. Строк подання студентом проєкту (роботи) 17 травня 2023 року

3. Вихідні дані до проєкту (роботи) розробка комп'ютерної 3D гри «The - Ball» у жанрі платформер за допомогою мови C# та середовища Unity

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметної області;

2) Огляд ігрових рушіїв;

3) Розробка гри.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів плакатів)

Інтерфейс гри (A3);

Інтерфейс інсталятора (A3);

Ігровий процес (A3);

Діаграма використання комп'ютерної гри (A3).

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	КАСЬЯН К. М., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання «01» березня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Робота з джерелами інформації	01.03.2023 р.	
2	Аналіз ігрових рушіїв, виділення недоліків та переваг	14.03.2023 р	
3	Розробка інтерфейсу гри	25.03.2023 р.	
4	Розробка рівнів	20.04.2023 р.	
5	Створення локалізації гри	22.04.2023 р.	
6	Тестування гри	27.04.2023 р	
7	Створення інсталятора	29.04.2023 р.	
8	Оформлення отриманих результатів у ПЗ	01.05.2023 р.	
9	Оформлення графічного матеріалу	10.05.2023 р.	
10	Оформлення допоміжного матеріалу	17.05.2023 р	

Студент (ка)



(підпис)

Єгор КНИРИК

(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)



(підпис)

Костянтин КАСЬЯН

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 79 с., 50 рис., 6 табл., 11 джерел.

3D, C#, GAME, GODOT ENGINE, UNITY, UNREAL ENGINE, ІГРОВИЙ РУШІЙ

Мета роботи – розробка комп'ютерної гри мовою C# за допомогою середовища Unity.

Об'єкт розробки – процес розробки комп'ютерної гри мовою C# за допомогою середовища Unity.

Пояснювальна записка складається з 3 розділів. Перший розділ присвячено опису та аналізу предметної області розробки, а саме: класифікації комп'ютерних ігор, етапи розробки комп'ютерних ігор.

Другий розділ присвячено огляду ігрових рушіїв а саме: Unity, Unreal Engine, Godot Engine.

У третьому розділі описується процес розробки гри, а саме: проектування, розробка інтерфейсу гри, розробка префабів та 3D моделей, розробка рівнів, локалізація гри, збірка гри, тестування, створення інсталятора.

ЗМІСТ

Скорочення та умовні позначки	7
Вступ.....	8
1 Опис предметної області	9
2 Огляд ігрових рушіїв.....	17
2.1 Unreal Engine.....	17
2.2 Godot Engine.....	18
2.3 Unity	19
3 Розробка гри.....	20
3.1 Проектування.....	20
3.2 Інтерфейс Unity. Налаштування проекту.....	21
3.3 Розробка інтерфейсу гри	25
3.4 Розробка префабів (Prefabs) та 3D моделей	37
3.5 Розробка рівнів	52
3.6 Локалізація гри	59
3.7 Збірка гри	63
3.8 Тестування	65
3.9 Створення інсталятора.....	67
Висновки	69
Перелік джерел посилання	70
Додаток А.....	71

Плакати:

Плакат 1 - Інтерфейс гри

Плакат 2 - Інтерфейс інсталятора

Плакат 3 - Ігровий процес

Плакат 4 - Діаграма використання комп'ютерної гри

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

VR - (Virtual Reality) Віртуальна реальність

IDE - (Integrated Development Environment) Інтегроване середовище розробки

2D - (2-Dimensional) Двовимірний

2.5D - (2.5-Dimensional) 2,5 - мірний

3D - (3-Dimensional) Тршовимірний

GUI - (Graphic User Interface) Графічний інтерфейс користувача

UI - (User Interface) Інтерфейс користувача

ВСТУП

Комп'ютерні ігри займають значну частину сучасної розважальної культури. Індустрія відеоігор, хоч і з'явилася відносно недавно, близько 40 років тому, пройшла вражаючий розвиток і стала великою та прибутковою галуззю з щорічними доходами в сотні мільярдів доларів. Зростання популярності віртуальних розваг можливе завдяки широкому поширенню комп'ютерних технологій, зокрема мережі Інтернет.

Цифрове поширення відеоігор сприяло можливості меншим і незалежним розробникам займатися створенням ігор без значних витрат на фізичні копії. Ринок комп'ютерних ігор зростає з кожним роком, і завдяки широкому спектру жанрів та різноманіттю ігор, кожен може знайти щось на свій смак. Від казуальних ігор до важких стратегій у реальному часі та онлайн ігор - вибір залишається за користувачем.

Варто відзначити, що в сучасному світі розробка комп'ютерних ігор не обмежується лише розважальними іграми. Відеоігри стали цінним інструментом для навчання та використовуються для досягнення дослідницьких цілей. Навчальні ігри створюються з метою передачі знань і навичок у вигляді цікавого та захоплюючого досвіду. Вони можуть бути використані у навчальних закладах або тренінгових програмах.

Таким чином, комп'ютерні ігри не тільки надають розважальний відпочинок, але й використовуються як ефективний інструмент для навчання та розвитку інтелектуальних здібностей. Їх роль у сучасному світі стає все більш ваговою і відкриває нові можливості для використання в різних сферах життя.

Метою дипломної роботи є розробка 3D гри «The - Ball» у жанрі платформер за допомогою мови C# та середовища Unity.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Комп'ютерні ігри - це програми, які створюються для розваги та розвитку, або для навчання інших функцій, які пов'язані з роботою на комп'ютері.

Комп'ютерні ігри можна класифікувати за багатьма параметрами:

- жанр;
- платформа;
- кількість гравців;
- вікова категорія.

Жанр комп'ютерної гри - це категорія, до якої можна віднести гру залежно від її характеристик і особливостей геймплею. Жанр визначає тематику, механіку гри, цільову аудиторію та інші аспекти гри. Нижче наведені деякі жанри комп'ютерних ігор.

Екшен (Action). Сутність екшенів відображає їхню назву – у перекладі з англійської вона означає «дію», і тут гравець знаходиться в самому центрі дії та керує ним. Здебільшого воно пов'язане з фізичними випробуваннями, які потрібно подолати. До цього жанру належали найперші відеоігри, наприклад, Galaga та Donkey Kong. Зазвичай в екшенах дуже легко розпочати гру, тому вони досі залишаються найпопулярнішими відеоіграми [1].

Платформери отримали свою назву завдяки тому, що в процесі гри її головний персонаж взаємодіє з платформами – зазвичай бігає, стрибає чи падає. Різноманітних платформерів дуже багато, найвідомішим з них, мабуть, можна назвати серію Super Mario Bros, а Donkey Kong був одним із найперших [1].

Стратегія (Strategy). Геймплей цього жанру сягає своїм корінням в традиційні настільні стратегічні ігри. Часто в цих іграх вам доступний цілий світ і всі його ресурси, а щоб впоратися з випробуваннями, гравцеві потрібно ретельно продумувати стратегію та тактику [1].

Згодом значна частина жанру стратегії перейшла від покрокових систем до реального часу, слідуючи бажанням гравців [1].

Рольова гра (RPG) – мабуть, другий за популярністю жанр. У ньому часто зустрічаються середньовічні або фентезійні сеттинги, що пов'язано з походженням жанру, який веде свою історію від Dungeons & Dragons та інших настільних ігор. Але рольові ігри фантастичної тематики, наприклад Mass Effect, Fallout і більшість частин Final Fantasy, також не обділені увагою хардкорних ролевиків і внесли в жанр свій унікальний внесок. Крім того, на жанр вплинули і культурні відмінності – зокрема багато геймерів ділять RPG на WRPG (західні) і JRPG (японські). Нарешті, однією із знакових фішок жанру є вибір гравця, що впливає на результат гри, тобто у багатьох RPG є кілька альтернативних кінцівок. А тепер розглянемо різні види рольових ігор [1].

Спортивна гра (Sports). Спортивні ігри, як і слідує з назви, симулюють різні види спорту, наприклад, гольф, баскетбол, звичайний та американський футбол. Зустрічаються також олімпійські види спорту типу лижних гонок, і навіть барний спорт на кшталт більярду або дартс. Зазвичай суперники управляються комп'ютером, хоча можна змагатися з іншими гравцями. І якщо ми говоримо про спорт, самі по собі відеоігри теж можна вважати спортом [1].

Симулятор (Simulator). Представники жанру симуляторів мають одну спільну рису – всі вони створені для симуляції реальних чи вигаданих систем, ситуацій чи подій [1].

Пригода (Adventure). Пригодницькі ігри часто діляться за стилем геймплею, а не сюжетом або контентом. І незважаючи на те, що з розвитком технологій розробники отримали безліч можливостей щодо реалізації сюжетів у цьому жанрі, за своєю суттю ці ігри недалеко відійшли від своїх текстових предків. Зазвичай у квестах гравці взаємодіють з іншими персонажами та навколишнім світом для вирішення головоломок із підказками, просуваючись таким чином за сюжетом. Крім періодичних міні-ігор, в квестах рідко зустрічаються екшен - елементи. Тому серед мейнстрімних геймерів жанр не такий популярний. Найуспішнішим квестом усіх часів вважається Myst. Він вийшов у 1993 році, який багато хто вважає роком розквіту жанру, і став грою, що найбільше продається для ПК усіх часів. Крім того, він відкрив світові епоху ігор на компакт-дисках [1].

Перегони (Racing) Одні з найпопулярніших ігор цієї категорії – гоночні симулятори серій Forza, Grand Turismo та Need for Speed, сюди ж ми відносимо класику ігрових автоматів типу Pole Position. Тут гравці беруть участь у гоночних заїздах проти суперників чи тимчасово [1].

Головоломка (Puzzle) Головоломки, або логічні ігри зазвичай граються на одному екрані або ігровому полі, а для проходження потрібно вирішувати певні інтелектуальні завдання.

Навчальна гра (Educational). У таких іграх проста механіка застосовується для навчання різних предметів та навичок, наприклад, математики або набору тексту на клавіатурі, тому вони часто входять до багатьох списків кращих відеоігор для дітей [1].

Оборона веж (Tower Defense). В обороні веж гравці повинні відображати хвилі ворогів, що керуються комп'ютером (часто званих «крипами»). Конкретні здібності веж і крипів у різних іграх можуть відрізнятися, але зазвичай є кілька веж з різними здібностями, наприклад, уповільненням або отруєнням крипів. Знищивши певну кількість крипів, гравець отримує ресурси на купівлю нових веж або апгрейд існуючих, наприклад, збільшення втрат або дальності [1].

Казуальні ігри засновані на найпростішій механіці і чудово підходять для короткої швидкоплинної гри. Завдяки розвитку мобільних ігор останні кілька років ми спостерігаємо вибухове зростання популярності цього жанру. Для утримання цікавих гравців розробники часто використовують нагороди та ретельно продумані рівні, оскільки геймплей у цьому жанрі зазвичай досить одноманітний [1].

Представлені вище жанри не можна назвати повними. Часто у ігор змішаний тип жанрів, а також останнім часом почали з'являтися ігри власних жанрів, які можна віднести як до одного з представлених тут жанрів, так і до самостійного, окремо від них.

Також ігри можна поділити за кількістю гравців на два види:

- однокористувацькі;
- багатокористувацькі.

За візуальним уявленням комп'ютерні ігри можна розділити на наступні види:

- 2D ігри;
- 2.5D ігри;
- 3D ігри;
- VR-ігри.

2D ігри - це ігри, які відображаються у двох вимірах, на площині, наприклад, Super Mario Bros.

2.5D ігри - це ігри, які відображаються в 2D-просторі, але мають 3D-елементи.

3D ігри - це ігри, які повністю відображаються в 3D-просторі.

VR-ігри - це ігри, які використовують технології віртуальної реальності, щоб створити іммерсивний досвід гри, наприклад, Beat Saber.

Комп'ютерні ігри можна розділити за типом платформи на наступні категорії:

- консольні;
- комп'ютерні;
- браузерні;
- мобільні ігри;
- віртуальна реальність;
- аркадні.

Вікові рейтинги – це системи які гарантують, що розважальний контент, такий як ігри, фільми, телевізійні шоу чи мобільні додатки чітко позначені мінімальними віковими рекомендаціями на основі контенту, який вони містять. Такі вікові рейтинги надають рекомендації споживачам, зокрема батькам, в процесі прийняття рішення про покупку того чи іншого продукту для власної дитини [2].

Відеоігри насолоджуються дуже різноманітні групи гравців по всій Європі. Діти та дорослі, чоловіки та жінки, усі вони регулярно грають в ігри на

призначений для цього ігровій консолі, на персональному комп'ютері чи мобільному пристрої, типу смартфона чи планшета [2].

В той час як більшість ігор підходять для гравців різного віку, частина з них доступні для дітей старшого віку або підлітків [2].

Рейтинг PEGI бере до уваги вікову придатність гри, а не рівень складності. Гра з рейтингом PEGI 3 не матиме невідповідного контенту, але інколи може бути важкою для засвоєння молодшими дітьми. І навпаки, є ігри PEGI 18, в які дуже легко грати, але вони містять елементи, які роблять їх неприйнятними для молодшої аудиторії. PEGI використовується та визнається у всій Європі та має потужну підтримку Європейської комісії. Він розглядається як зразок європейської гармонізації у галузі захисту дітей [2].

PEGI 3. Зміст ігор з рейтингом PEGI 3 вважається придатним для всіх вікових груп. У грі не повинно бути звуків чи картинок, які можуть налякати маленьких дітей. Прийнятна дуже легка форма насильства (в комічному контексті чи подитячому). Лайка повинна бути відсутня у будь-якій формі [3]. Зображення PEGI 3 представлено на рисунку 1.1.



Рисунок 1.1 – Рейтинг PEGI 3

PEGI 7. До цієї категорії слід віднести ігровий контент зі сценами або звуками, які можуть лякати дітей молодшого віку. Дуже легкі форми насильства (приховане, не детальне або нереалістичне насильство) є прийнятними для гри з рейтингом PEGI 7 [3]. Зображення PEGI 7 представлено на рисунку 1.2.



Рисунок 1.2 – Рейтинг PEGI 7

PEGI 12. Відеоігри, які демонструють насильство трохи більш графічного характеру щодо фантастичних персонажів або нереалістичне насильство над людськими персонажами, потрапляють до цієї вікової категорії. Можуть бути присутніми сексуальні натяки чи сексуальне позування, тоді як будь-яка лайка в цій категорії повинна бути м'якою [3]. Зображення PEGI 12 представлено на рисунку 1.3.



Рисунок 1.3 – Рейтинг PEGI 12

PEGI 16. Цей рейтинг застосовується, коли зображення насильства (або сексуальної активності) досягає рівня, який виглядає так само, як воно могло б виглядати у реальному житті. Використання лайки в іграх з рейтингом PEGI 16 може бути більш екстремальним, тоді як вживання тютюну, алкоголю або незаконних наркотиків також може бути присутнім [3]. Зображення PEGI 16 представлено на рисунку 1.4.



Рисунок 1.4 – Рейтинг PEGI 16

PEGI 18. Класифікація «для дорослих» застосовується, коли рівень насильства досягає стадії, коли воно стає зображенням грубого насильства, очевидно безмотивного вбивства або насильства щодо беззахисних персонажів. Гламуризація вживання незаконних наркотиків, імітація азартних ігор та явна сексуальна активність також повинні належати до цієї вікової категорії [3]. Зображення PEGI 18 представлено на рисунку 1.5.



Рисунок 1.5 – Рейтинг PEGI 18

Створення комп'ютерних ігор - це складний та багатоетапний процес, який може включати наступні етапи:

- концепція і проектування;
- розробка;
- тестування та налагодження;
- реліз та підтримка.

Далі наведено більш детальний огляд пунктів.

Етап концепції і проектування включає в себе: визначення цільової аудиторії, вибір платформи та жанру гри, визначення візуального стилю, вибір програмного забезпечення для розробки гри, визначення ідеї, цілі та концепції гри. Кожна гра починається з ідеї. Чим оригінальніше буде ідея, тим більше шансів, що гра сподобається користувачам.

Етап розробки включає в себе: створення моделей персонажів, об'єктів та оточення гри, розробка текстур, матеріалів та освітлення, програмування логіки гри, включаючи рух, фізику, штучний інтелект та геймплейні механіки, створення звукових ефектів та музики, розробка інтерфейсу користувача та системи управління.

Етап тестування та налагодження включає в себе: виявлення та виправлення помилок, багів та недоліків гри, проведення тестування гри на різних платформах та з різними конфігураціями обладнання, балансування геймплею та оптимізація продуктивності гри.

Етап реліз та підтримка включає в себе: підготовку гри до публікації на різних платформах, розробка маркетингових матеріалів та просування гри, випуск гри на ринок та реклама, підтримку гри, включаючи виправлення помилок, випуск оновлень та взаємодію з гравцями.

Ігровий рушій - це програмне забезпечення, яке надає розробникам інструменти і середовище для створення, редагування і реалізації відеоігор. Він забезпечує необхідні функціональні можливості для роботи з графікою, фізикою, звуком, та іншими аспектами гри.

Ігрові рушії включають в себе набір інструментів і функціональності, які дозволяють розробникам створювати ігри безпосередньо в середовищі рушія. Вони забезпечують такі можливості, як візуальне редагування сцен, програмування логіки гри, імпорт графіки та звукових ефектів, управління ресурсами гри, тестування і налагодження гри та інше.

Найпопулярніші ігрові рушії, такі як Unity, Unreal Engine, Godot та інші, надають розробникам широкий набір інструментів і можливостей для створення як

невеликих інді-ігор, так і великих AAA-проектів. Вони підтримують різні платформи, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність.

З використанням ігрових рушіїв розробники можуть ефективно створювати гри, зосереджуючись на творчості та ігровому дизайні, оскільки багато технічних аспектів уже вирішено в рамках рушія. Це дозволяє зберегти час і зусилля, що робить ігрові рушії потужним інструментом для розробки відеоігор.

2 ОГЛЯД ІГРОВИХ РУШІВ

2.1 Unreal Engine

Unreal Engine - багатоплатформовий ігровий рушій, який розроблений компанією Epic Games. Перша версія була реалізована в 1998 році. Остання версія Unreal Engine 5.2, була випущена в 2023 році. Unreal Engine використовується для розробки великих та складних ігор з відкритим світом. Ігровий рушій Unreal Engine доволі складний в використанні.

Втім, новачкам також тут знайдеться місце. Все завдяки середовищу візуального програмування Blueprints, яке вважається найпотужнішим за можливостями. З його допомогою можна створити повноцінну гру, не написавши жодного рядка коду. Як доказ Epic Games зробила так основу популярної королівської битви Fortnite. Втім, для охочих поекспериментувати є скрипти на C++ [4].

Ігровий рушій є безкоштовним, але якщо доходи гри перевищують 1000000\$ за квартал, 5% віддається компанії Epic Games [5].

Переваги:

- крос - платформеність.
- висока функціональність;
- велике ком'юніті;
- підтримка графічного реактора для створення алгоритмів;

Недоліки:

- підтримується лише одна мова;
- непростий в освоєнні.
- умовно-безкоштовний;
- високі системні вимоги;

2.2 Godot Engine

Godot - це безкоштовний, відкритий ігровий движок з візуальною системою редагування і скриптовою мовою програмування, що базується на Python-подібній мові з назвою GDScript. Двигун забезпечує багато різних можливостей для створення 2D та 3D ігор, включаючи фізику, анімацію, звук та відео.

Godot є повністю безплатним, з відкритим кодом під ліцензією MIT. Без будь-яких платежів, відрахувань, і тому подібного. Ігри користувачів є їхніми, до останнього рядка коду рушія. Розвиток Godot є цілком незалежним і керованим спільнотою, що дозволяє користувачам формувати свій рушій відповідно до їхніх очікувань. Це підтримується некомерційною організацією Software Freedom Conservancy [6].

Переваги:

- безкоштовний та відкритий код, що дає можливість модифікувати та доповнювати ігровий движок під свої потреби;
- підтримка мов програмування, таких як C#, C++ та GDScript, що дає змогу вибрати найбільш зручний варіант для конкретного проекту;
- зручний та простий використовувати інтерфейс користувача, який дозволяє швидко створювати прототипи ігор та редагувати графічні ефекти;
- наявність багатофункціональних інструментів для розробки, таких як редактор анімації, фізичний редактор, система частинок та інші;

- висока продуктивність, що дозволяє створювати графічно складні проекти з великою кількістю об'єктів та спецефектів.

Недоліки:

- не така велика кількість додатків та плагінів, порівняно з іншими ігровими движками.

- не дуже велика кількість матеріалів для вивчення, хоча з'являється все більше та більше навчальних ресурсів та книг.

2.3 Unity

Unity: один з найпопулярніших ігрових движків, що використовується для розробки ігор на різних платформах.

У Unity дуже простий інтерфейс, завдяки якому багато об'єктів можна «викидати» в робочу зону за допомогою курсора. Усі проекти в Unity складаються із «сцен», кожен з яких вдасться редагувати окремо, що розширює простір для творчості. Рушій підтримує написання скриптів на C#, а для новачків підійде середовище візуального програмування Bolt [4].

Unity також має Asset Store - це онлайн-магазин, що пропонує розробникам ігор, які використовують Unity, широкий вибір ресурсів, таких як готові моделі 3D-об'єктів, текстури, звукові ефекти, плагіни, скрипти та інші корисні компоненти. Цей магазин надає можливість економити час і зусилля на створення компонентів самостійно, а також розширювати можливості своєї гри за допомогою сторонніх ресурсів.

При персональній ліцензії платити розробникам двигуна не потрібно, якщо оборот або обсяг залучених інвестицій не вище 100 тис. \$ за останні 12 місяців [7].

З переваг можна виділити наступні:

- велике ком'юніті;
- висока функціональність;

- зручність в роботі і компіляції;
- мультиплатформеність;
- велика кількість бібліотек.

Недоліками середовища є:

- умовно – безкоштовний
- закритий вихідний код;

3 РОЗРОБКА ГРИ

3.1 Проектування

Комп'ютерна гра, що розробляється, має розважальних характер. Мета гри – пройти всі рівні.

3D Гра «The-Ball» відноситься до жанру платформер. Основні ігрові механіки пов'язані з платформами: падаючі платформи, рухомі платформи, обертаючі платформи. Також до ігрових механік входять: телепорт, шипи.

Для розробки комп'ютерної гри, як і для будь-якого іншого продукту, одну з найважливіших ролей відіграє визначення цільової аудиторії продукту. Коли визначена цільова аудиторія, розробнику набагато легше визначити багато аспектів продукту, що розробляється. Гра орієнтована здебільшого на дитячу аудиторію віком від 7 до 14 років. Складність рівнів занадто проста для дорослої людини, але обмежувальний контент в грі відсутній.

Комп'ютерна гра розробляється для операційних систем: Windows 10, Windows 11.

3.2 Інтерфейс Unity. Налаштування проекту

Інтерфейс Unity розділений на кілька основних блоків: вкладка Scene, вкладка Game, панель інструментів, вкладка Hierarchy, панель Inspector, вкладки Project і Console. Інтерфейс Unity наведено на рисунку 3.1. Також за необхідності можна додати деякі вкладки наприклад Lighting ті інші. У якості інтегрованої середовища розробки Unity при встановленні завантажує та обирає IDE Visual Studio .

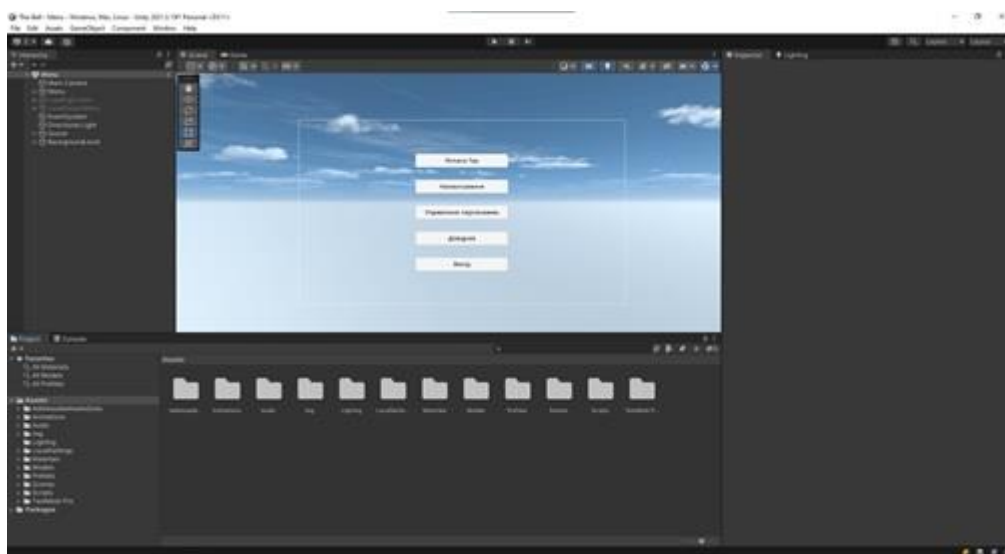


Рисунок 3.1 – Інтерфейс Unity

Вкладка Scene дозволяє візуально переміщатися сценою і редагувати її. Вигляд сцени може відображати тривимірну або двовимірну перспективу залежно від типу проекту.

Вкладка Game - це область, де можна переглядати та тестувати гру в реальному часі. Вона містить відображення гри з точки зору гравця.

Панель інструментів забезпечує доступ до найважливіших робочих функцій.

Вкладка Hierarchy містить у собі список всіх об'єктів, що належать до сцени.

Панель Inspector відображає детальну інформацію про вибраний об'єкт. Панель містить налаштування та компоненти, що відносяться до обраного об'єкту.

Вкладка Project відображає список усіх ресурсів, що використовуються в грі, таких як текстури, звуки, моделі та інші файли.

Вкладка Console відображає налагоджувальну інформацію, таку як повідомлення про помилки, попередження та інші повідомлення.

Перш за все, необхідно додати в проект усі, створені раніше та необхідні для подальшої розробки файли. Для зручності навігації створюємо необхідні папки. Основна файлова система без вкладених папок наведена на рисунку 3.2.

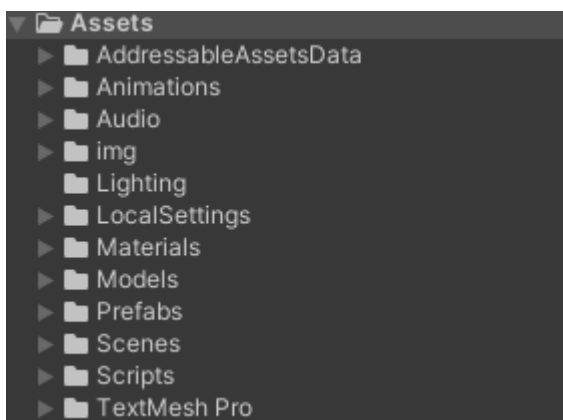


Рисунок 3.2– Файлова система Assets

В директорії Scripts містяться скрипти з описом всіх класів. Папка Scenes містить у собі усі сцени гри. В папці Animations містяться анімації для кнопок в меню. В папці Audio містяться звуки, які застосовуються в грі, такі як музика в меню. В папці img містяться зображення та іконки. В папці Lighting міститься файл налаштування освітлення, що застосовуються в грі. В директорії LocalSettings знаходяться таблиці та файли для локалізації гри. В папці Materials зберігаються матеріали для 3D моделей. В директорії Prefabs містяться всі префаби. В директорії Models зберігаються усі 3D моделі. Директорії Addressable Assets Data та TextMesh Pro створені є допоміжними в них містяться файли необхідні для роботи деяких встановлених пакетів.

Вся інтерактивність та ігровий процес у Unity будуються на основі трьох фундаментальних блоків: об'єкти GameObject, компоненти та змінні.

Будь-який об'єкт у грі є `GameObject`, будь то персонажі, джерела світла, спецефекти, декорації та все інше.

Ігрові об'єкти самі по собі не мають деякої поведінки. Для того щоб об'єкт почав працювати, ігровому об'єкту потрібні різні атрибути, що додаються за допомогою компонентів.

Компоненти (`Component`) визначають поведінку ігрових об'єктів, до яких вони прикріплені, і керують ними.

Компоненти мають ряд властивостей або змінних, які можна налаштувати у вікні `Inspector` редактора Unity або за допомогою скрипту.

У Unity є безліч найрізноманітніших компонентів, але невдовзі можете зіткнутися з необхідністю створити власний компонент реалізації своїх алгоритмів. Це можна зробити за допомогою скриптів, в яких описано вашу власну ігрову логіку та поведінку об'єктів, і які потім прикріплюються до ігрових об'єктів як компоненти. Кожен скрипт пов'язується із внутрішніми механізмами Unity шляхом реалізації класу, похідного від вбудованого класу `MonoBehaviour`.

Компоненти на основі скриптів дозволяють запускати ігрові події, перевіряти об'єкт на предмет зіткнень, застосовувати фізичні властивості, програмувати реакцію на керування користувача та багато іншого. Блок-схема життєвого циклу скрипту наведена на рисунку 3.3.

`Scene` - це середовище, в якому розгортається дія гри. В Unity можна створювати багато сцен, які можна об'єднувати в одну гру. Кожна сцена може містити різні об'єкти, які взаємодіють між собою.

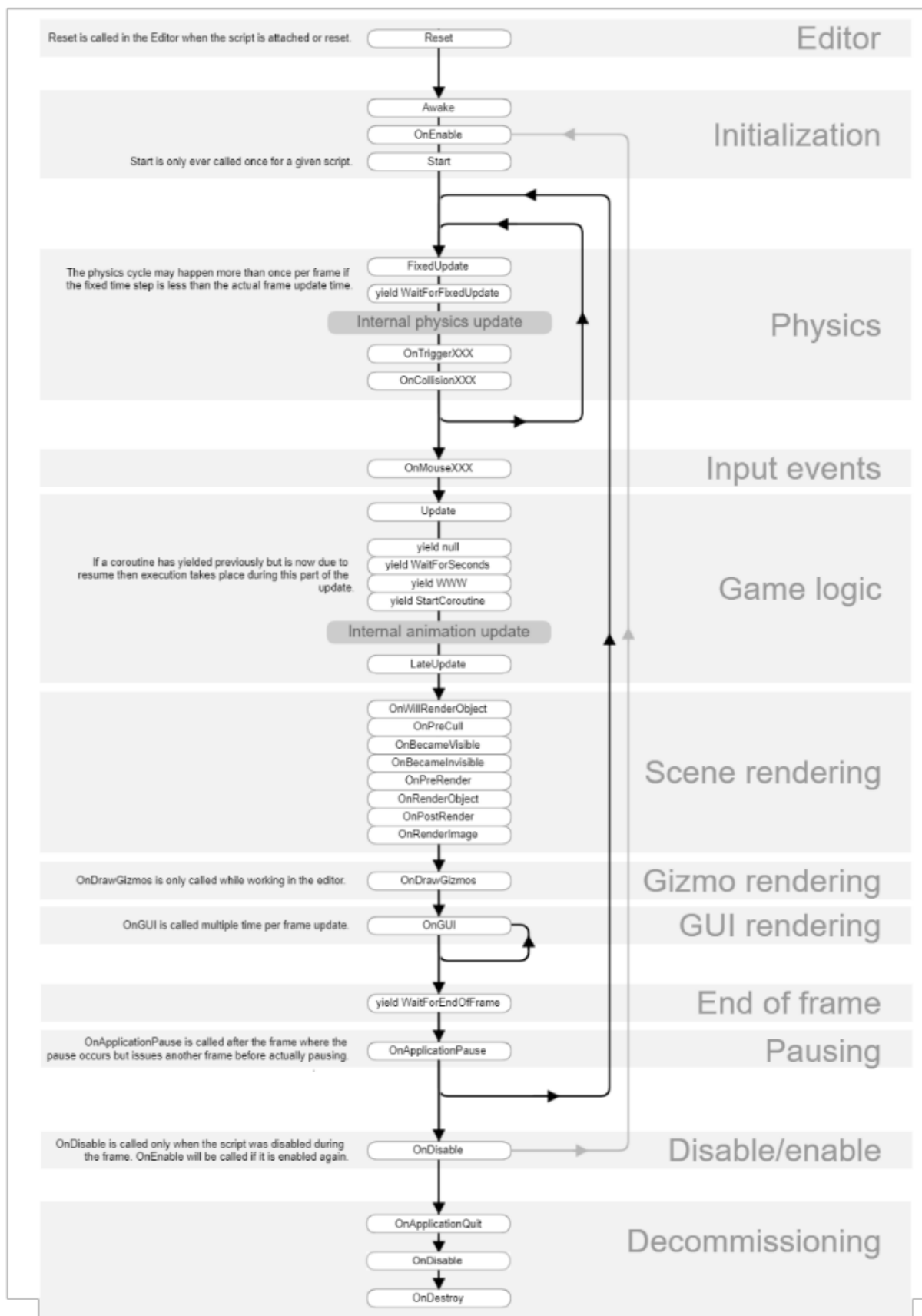


Рисунок 3.3 – Блок-схема життєвого циклу скрипту

3.3 Розробка інтерфейсу гри

Для розробки інтерфейсу гри використовувалися наступні елементи Unity UI:

- canvas;
- buttons;
- image;
- panel;
- slider;
- dropdown;
- text;
- toggle;
- scrollbar.

Canvas - це основний елемент для створення графічного інтерфейсу користувача (GUI) в Unity. Він використовується для створення контейнера, в якому можна розміщувати інші елементи, такі як кнопки, текстові поля та інші елементи GUI.

Buttons - елемент Button дозволяє створювати кнопки в Unity, які можна налаштувати таким чином, щоб вона виконувала дії, коли на неї натискають.

Image - елемент Image дозволяє відображати зображення в Unity. Можна налаштувати зображення, щоб воно було відображене зі змінним розміром, кольором, прозорістю та іншими властивостями.

Panel - це елемент, який дозволяє створювати панелі в Unity. Панелі можна використовувати для групування інших елементів, таких як кнопки, текстові поля та інші компоненти Unity.

Slider - це елемент, який дозволяє користувачеві вибирати значення з певного діапазону, переміщуючи повзунок між мінімальним і максимальним значеннями.

Dropdown - це елемент, який дозволяє створювати випадаючі списки в Unity.

Text - це елемент, який дозволяє додавати текстові елементи на екран в Unity.

Toggle - це елемент, який дозволяє створювати перемикачі в Unity.

Scrollbar - це елемент, який дозволяє додавати прокрутку на екран в Unity.

Main Camera - це елемент, який використовується для створення камери в Unity. Камера відповідає за те, що гравець бачить на екрані.

Інтерфейс гри складається з наступних вікон: головного меню, меню вибору рівня, меню налаштувань, меню паузи, довідника, меню управління персонажем. Всі крім меню паузи знаходяться на одній сцені.

Головне меню дозволяє перейти в меню вибору рівня, меню налаштувань, довідник, меню управління персонажем, а також дає можливість вийти з гри. Головне меню представлено на рисунку 3.4.

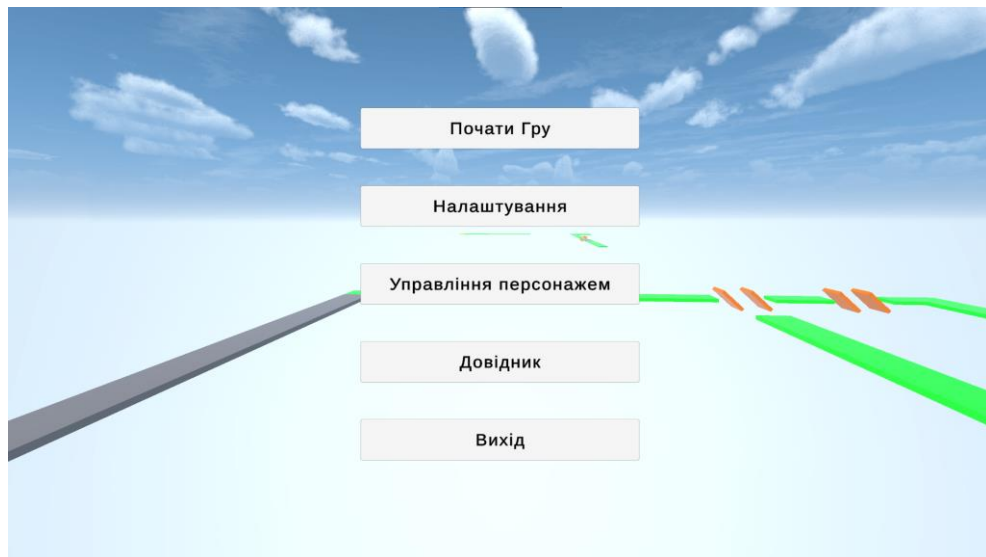


Рисунок 3.4 – Головне меню

Обертання камери в сцені реалізує скрипт, який наведено в лістингу 3.1.

Лістинг 3.1 - Скрипт обертання камери

```
using UnityEngine;

public class CameraOrbit : MonoBehaviour
{
    [SerializeField]
    private Transform target;

    private float distance = 40f;
    private float speed = 0.1f;

    void Update()
```

```

    {
        float currentDistance = Vector3.Distance(transform.position,
target.position);
        float angle = Time.time * speed;

        Vector3 newPosition = new Vector3(
            target.position.x + Mathf.Cos(angle) * distance,
            transform.position.y,
            target.position.z + Mathf.Sin(angle) * distance
        );

        transform.position = newPosition;
        transform.LookAt(target);
    }
}

```

Увімкнення відображення курсору в головному меню реалізовано за допомогою скрипту наведеного в лістингу 3.2.

Лістинг 3.2 - Скрипт увімкнення відображення курсу

```

using UnityEngine;

public class EnableCursor : MonoBehaviour
{
    private void Awake()
    {
        Cursor.lockState = CursorLockMode.None;
    }
}

```

Вихід з гри реалізовано за допомогою скрипту наведено в лістингу 3.3.

Лістинг 3.3 - Скрипт виходу з гри

```

using UnityEngine;

public class ExitGame : MonoBehaviour
{
    public void Exit()
    {
        Application.Quit();
    }
}

```

Перехід між різними частинами інтерфейсу здійснюється за допомогою функції `GameObject.SetActive`, яка спрацьовує при натисканні кнопки. Приклад представлено на рисунку 3.5.

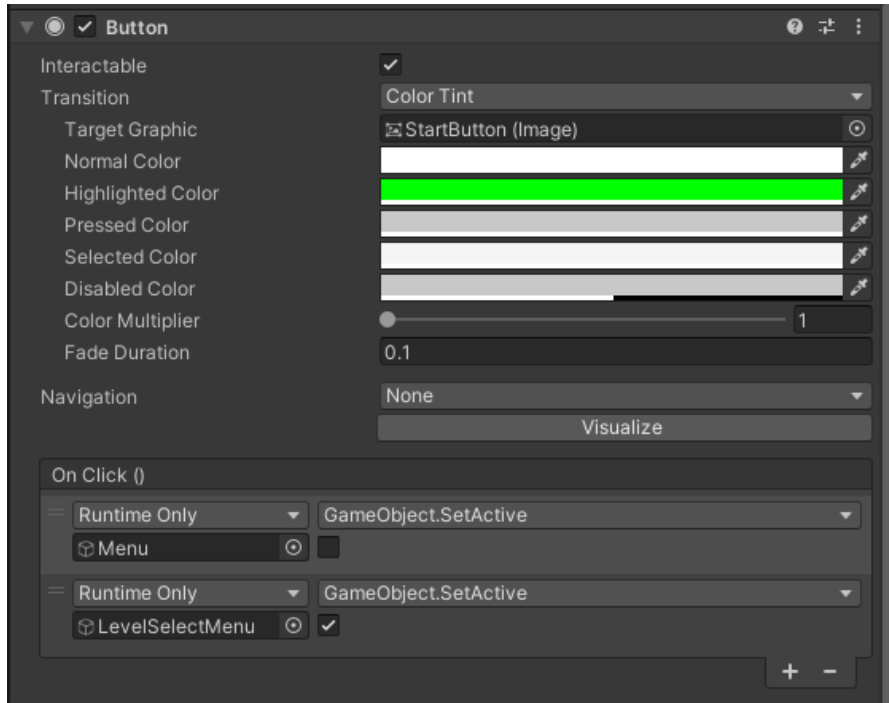


Рисунок 3.5 – Налаштування кнопки «Почати Гру»

Меню вибору рівня відповідає за відображення всіх рівнів, завантаження обраного рівня. Меню вибору рівня представлено на рисунку 3.6.



Рисунок 3.6 – Меню вибору рівня

Скрипт кнопки рівня наведено в лістингу 3.4.

Лістинг 3.4 - Скрипт кнопки рівня

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

public class LevelButton : MonoBehaviour
{
    [SerializeField]
    private Slider slidebar;

    [SerializeField]
    private GameObject loadingScreen;

    [SerializeField]
    private int SceneIndex;

    public void OpenScene()
    {
        loadingScreen.SetActive(true);
        StartCoroutine(LoadAsync());
    }

    IEnumerator LoadAsync()
    {
        AsyncOperation asyncLoad =
SceneManager.LoadSceneAsync(SceneIndex);
        asyncLoad.allowSceneActivation = false;

        while (!asyncLoad.isDone)
        {
            slidebar.value = asyncLoad.progress;

            if (asyncLoad.progress >= .9f &&
!asyncLoad.allowSceneActivation)
            {
                yield return new WaitForSeconds(1);
                asyncLoad.allowSceneActivation = true;
            }

            yield return null;
        }
    }
}
```

Скрипт відкриття рівнів наведено в лістингу 3.5.

Лістинг 3.5 – Скрипт відкриття рівнів

```
using UnityEngine;
using UnityEngine.UI;

public class LevelSelector : MonoBehaviour
{
    [SerializeField]
    private Button[] LevelButtons;

    private SaveData Level;

    private void Awake()
    {
        Level = SaveManager.LoadLevel();

        for (int i = 0; i < LevelButtons.Length; i++)
        {
            if(i > Level.SaveLevel)
                LevelButtons[i].interactable = false;
        }
    }
}
```

Збереження ігрового прогресу відбувається шляхом запису у файл класу SaveData, в якому знаходиться змінна типу int, яка зберігає номер рівня. Клас SaveData наведено в лістингу 3.6.

Лістинг 3.6 – Клас SaveData

```
using System;

[Serializable]
public class SaveData
{
    public int SaveLevel;
}
```

Менеджер збережень відповідає за запис в файл та зчитування з файлу класу SaveData. Скрипт менеджера збережень наведено в лістингу 3.7.

Лістинг 3.7 – Скрипт менеджер збережень

```

using UnityEngine;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

public static class SaveManager
{
    public static void SaveLevel(SaveData data)
    {
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Create(Application.persistentDataPath
+ "/MySaveData.dat");
        bf.Serialize(file, data);
        file.Close();
    }

    public static SaveData LoadLevel()
    {
        if (File.Exists(Application.persistentDataPath +
"/MySaveData.dat"))
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream file
=File.Open(Application.persistentDataPath + "/MySaveData.dat",
 FileMode.Open);
            SaveData data = (SaveData)bf.Deserialize(file);
            file.Close();
            return data;
        }
        else
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream file =
File.Create(Application.persistentDataPath + "/MySaveData.dat");
            SaveData data = new SaveData();
            data.SaveLevel = 0;
            bf.Serialize(file, data);
            file.Close();
            return data;
        }
    }
}

```

Меню налаштувань відповідає за налаштування: роздільної здатності, повноекранний режим, графіки, гучності музики, гучності звукових ефектів, чутливості камери, вибір мови. Меню налаштувань представлено на рисунку 3.7.

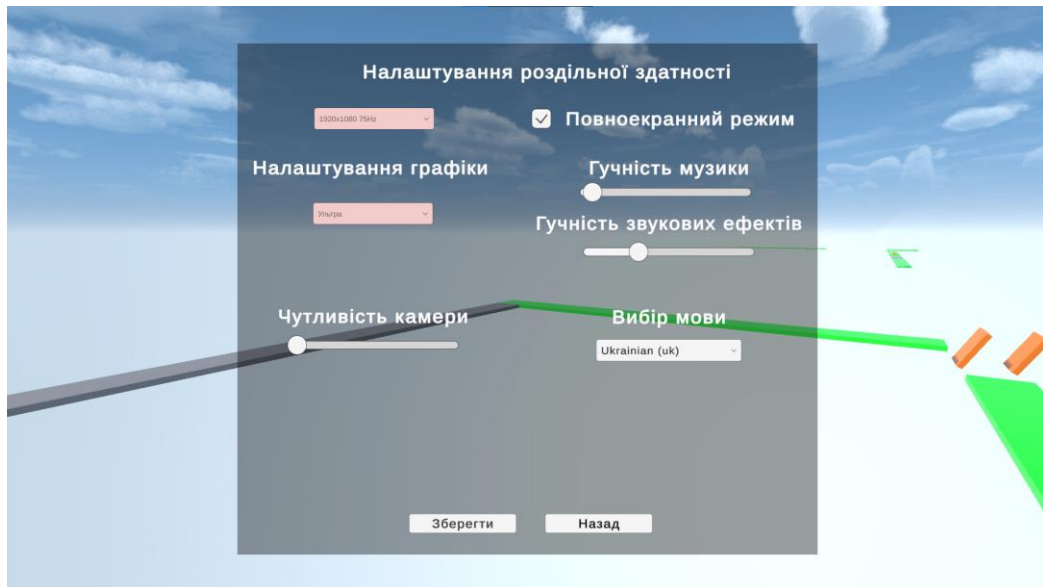


Рисунок 3.7 – Меню налаштувань

Скрипт, який реалізує функціонал налаштувань роздільної здатності, повноекранного режиму, графіки наведений в додатку А.1

Функціонал зміни мови реалізує скрипт наведений у лістингу 3.8.

Лістинг 3.8 – Скрипт зміни мови

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Localization.Settings;
using TMPro;

public class LocaleDropdown : MonoBehaviour
{
    public TMP_Dropdown dropdown;

    IEnumerator Start()
    {
        yield return LocalizationSettings.InitializationOperation;

        var options = new List<TMP_Dropdown.OptionData>();
        int selected = 0;
        for (int i = 0; i <
LocalizationSettings.AvailableLocales.Locales.Count; ++i)
        {
            var locale =
LocalizationSettings.AvailableLocales.Locales[i];
            if (LocalizationSettings.SelectedLocale == locale)
                selected = i;
            options.Add(new TMP_Dropdown.OptionData(locale.name));
        }
    }
}
```

```

    }
    dropdown.options = options;

    dropdown.value = selected;
    dropdown.onValueChanged.AddListener(LocaleSelected);
}

static void LocaleSelected(int index)
{
    LocalizationSettings.SelectedLocale =
LocalizationSettings.AvailableLocales.Locales[index];
}
}

```

Функціонал налаштувань чутливості камери виконує скрипт наведений в лістингу 3.9.

Лістинг 3.9 – Скрипт налаштування чутливості камери

```

using UnityEngine;
using UnityEngine.UI;

public class SensivitySettings : MonoBehaviour
{
    [SerializeField]
    private Slider SensivitySlider;

    void Start()
    {
        ChangeSensivity(PlayerPrefs.GetFloat("Sensivity", 1));
    }

    public void ChangeSensivity(float sensivity)
    {
        SensivitySlider.value = sensivity;
        PlayerPrefs.SetFloat("Sensivity", sensivity);
    }
}

```

Функціонал налаштувань: гучності музики, гучності звукових ефектів виконує скрипт наведений в лістингу 3.10.

Лістинг 3.10 - Скрипт налаштування гучності музики та звукових ефектів

```
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.UI;

public class VolumeSettings : MonoBehaviour
{
    [SerializeField]
    private AudioMixer Mixer;

    [SerializeField]
    private Slider MusicSlider, SoundEffectsSlider;

    void Start()
    {
        ChangeMusicVolume(PlayerPrefs.GetFloat("MusicVolume", 1));
        ChangeSoundEffectsVolume(PlayerPrefs.GetFloat("SoundEffectsVolume",
1));
    }

    public void ChangeMusicVolume(float volume)
    {
        MusicSlider.value = volume;
        Mixer.SetFloat("MusicVolume", Mathf.Log10(volume) * 20);
        PlayerPrefs.SetFloat("MusicVolume", volume);
    }

    public void ChangeSoundEffectsVolume(float volume)
    {
        SoundEffectsSlider.value = volume;
        Mixer.SetFloat("SoundEffectsVolume", Mathf.Log10(volume) * 20);
        PlayerPrefs.SetFloat("SoundEffectsVolume", volume);
    }
}
```

Завантаження нової сцени знищує всі об'єкти поточної сцени. Для того щоб запобігти знищенню об'єкту в якому знаходиться музика та звукові ефекти використовується скрипт наведений в лістингу 3.11.

Лістинг 3.11 – Скрипт збереження об'єкту

```
using UnityEngine;

public class DontDestroy : MonoBehaviour
{
    void Awake()
    {
        GameObject[] objs = GameObject.FindGameObjectsWithTag("Music");

        if (objs.Length > 1)
        {
            Destroy(this.gameObject);
        }

        DontDestroyOnLoad(this.gameObject);
    }
}
```

Меню паузи можна викликати лише під час ігрового процесу. Головне призначення меню паузи призупиняти ігровий процес та надавати наступні можливості: перезапуск рівня, завантаження головного меню, продовжити ігровий процес, налаштування гучності музики та звукових ефектів. Меню паузи представлено на рисунку 3.8. Основний функціонал виконує скрипт наведений в додатку A2.

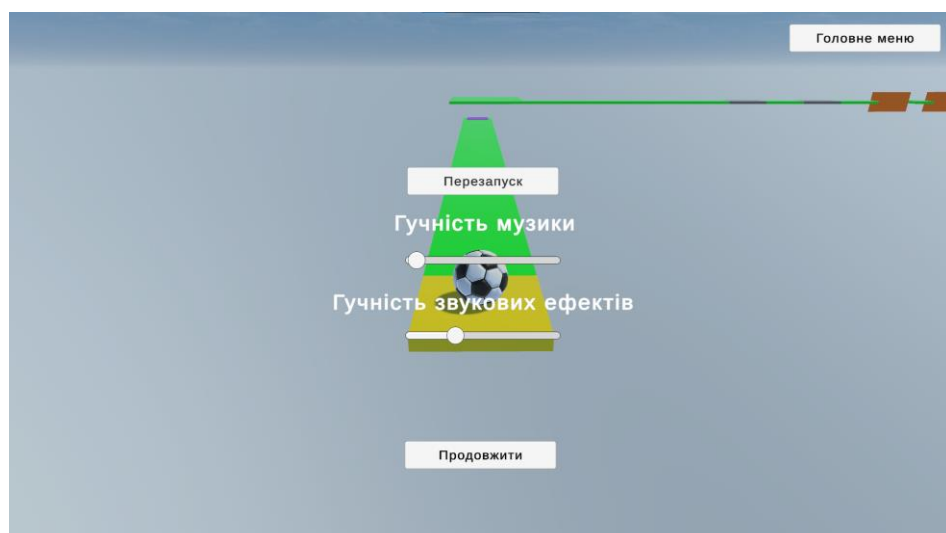


Рисунок 3.8 – Меню паузи

Налаштування гучності музики та звукових ефектів в меню паузи виконує скрипт наведений в лістингу 3.12.

Лістинг 3.12 – Скрипт налаштування музики та звукових ефектів

```
using UnityEngine;
using UnityEngine.Audio;

public class LoadSoundSettings : MonoBehaviour
{
    [SerializeField]
    private AudioMixer Mixer;

    void Start()
    {
        ChangeMusicVolume(PlayerPrefs.GetFloat("MusicVolume", 1));
        ChangeSoundEffectsVolume(PlayerPrefs.GetFloat("SoundEffectsVolume", 1));
    }

    public void ChangeMusicVolume(float volume)
    {
        Mixer.SetFloat("MusicVolume", Mathf.Log10(volume) * 20);
    }

    public void ChangeSoundEffectsVolume(float volume)
    {
        Mixer.SetFloat("SoundEffectsVolume", Mathf.Log10(volume) * 20);
    }
}
```

Меню управління персонажем відповідає за відображення способу керування ігровим персонажем. Зовнішній вигляд представлено на рисунку 3.9.

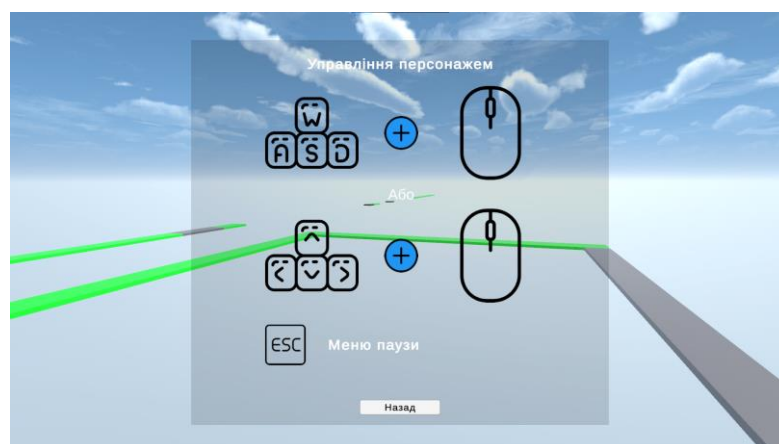


Рисунок 3.9 – Меню управління персонажем

Довідник відповідає за надання інформації стосовно ігрового процесу. Зовнішній вигляд довідника представлено на рисунку 3.10.

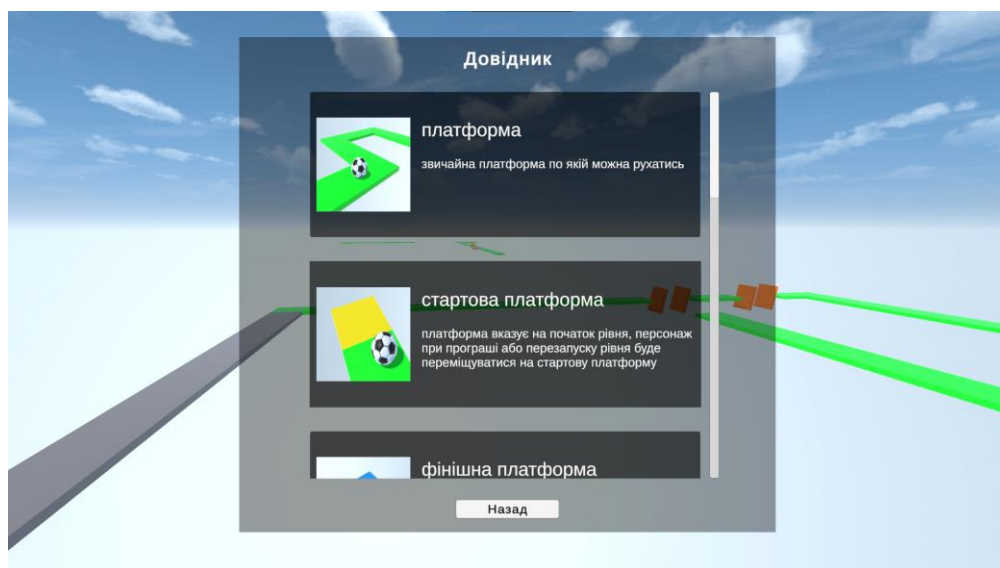


Рисунок 3.10 – Довідник

3.4 Розробка префабів (Prefabs) та 3D моделей

Blender - це потужний безкоштовний 3D-редактор, що дозволяє створювати та редагувати 3D - моделі, анімації, візуалізації. Цей програмний продукт має відкритий вихідний код та широкий спектр функцій та інструментів для створення професійних 3D - моделей.

За допомогою Blender розроблені наступні 3D моделі: Box, Maze, QR_Code, Spike, Football. Всі інші 3D моделі розроблені в Unity.

3D модель Box представлена на рисунку 3.11.

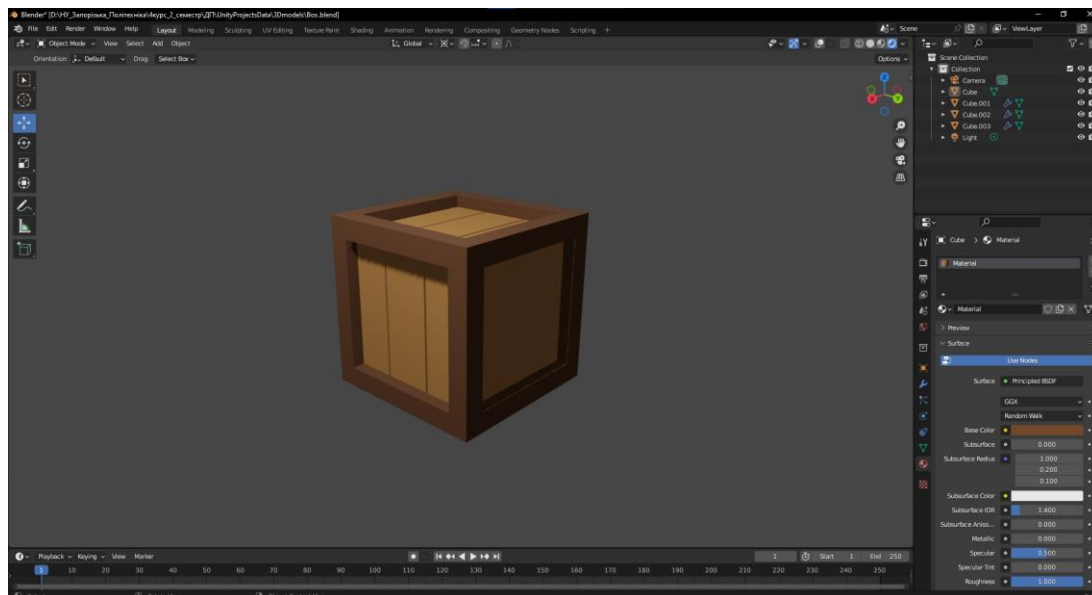


Рисунок 3.11 – 3D модель Box

3D модель Maze представлена на рисунку 3.12.

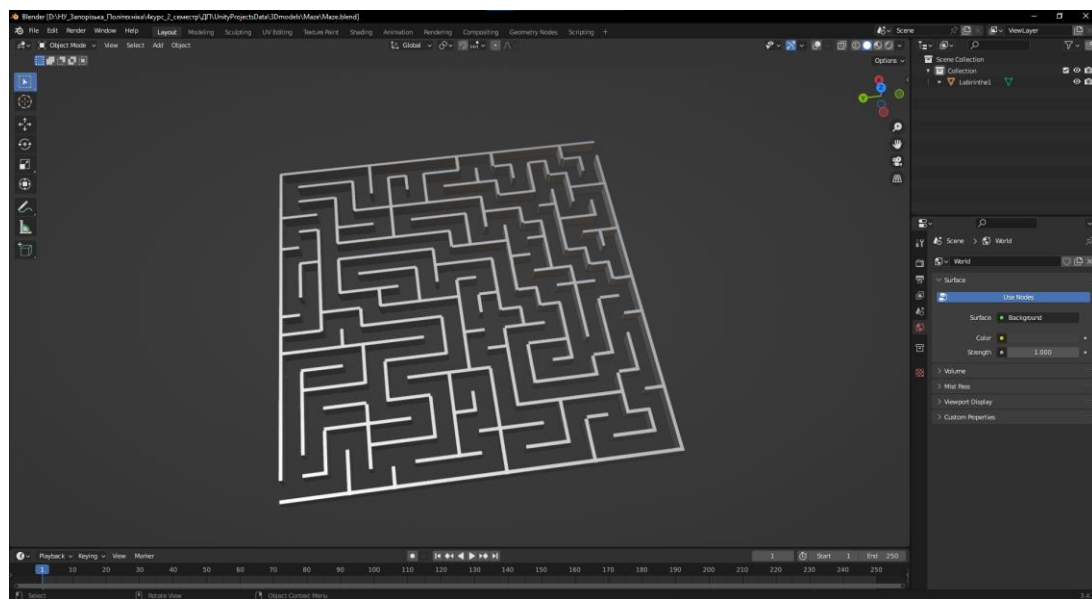


Рисунок 3.12 – 3D модель Maze

3D модель QR_Code представлена на рисунку 3.13.

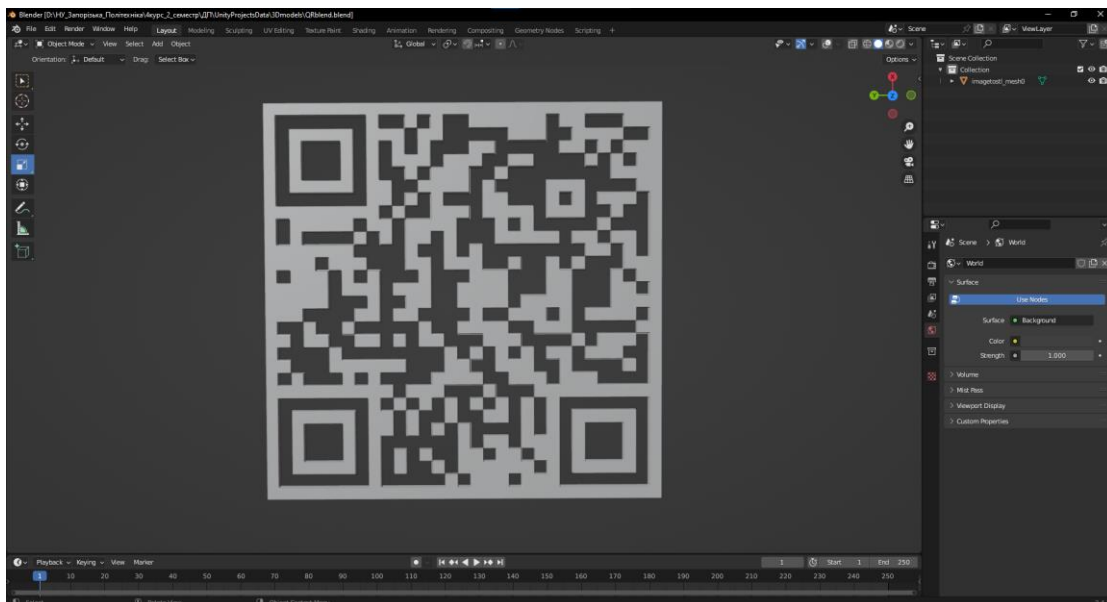


Рисунок 3.13 – 3D модель QR_Code

3D модель Spike представлена на рисунку 3.14.

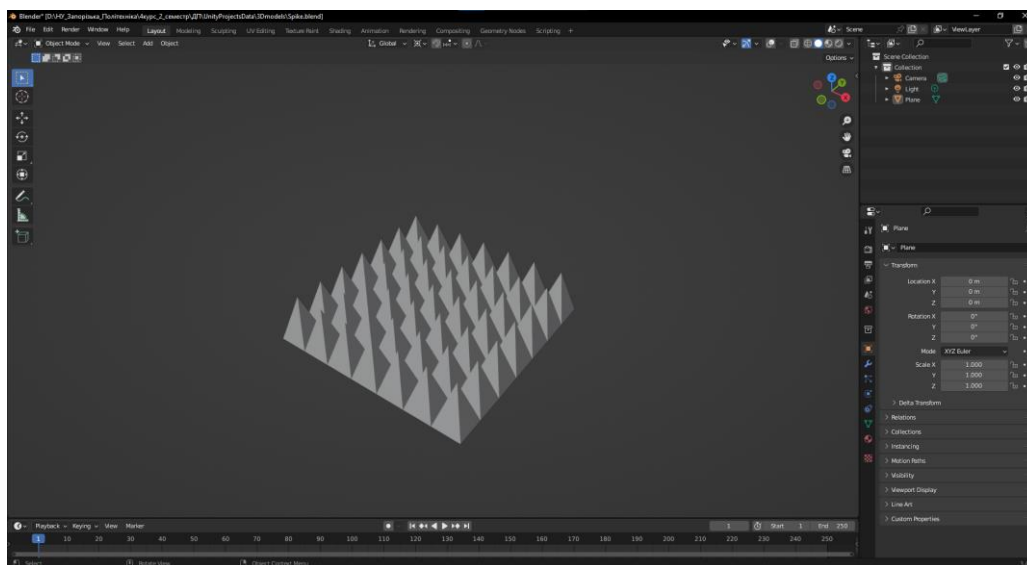


Рисунок 3.14 – 3D модель Spike

3D модель Football представлена на рисунку 3.15.

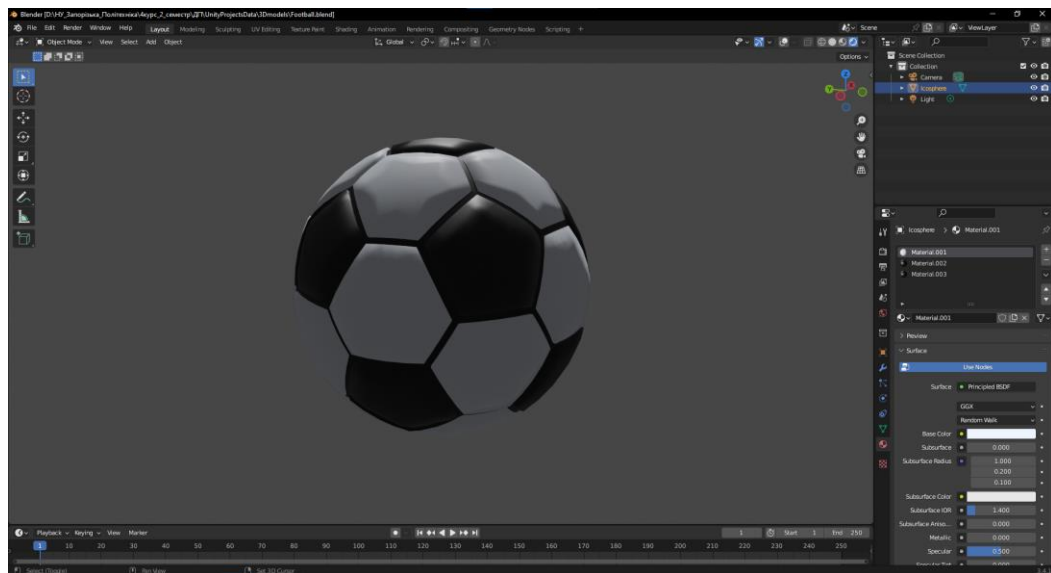


Рисунок 3.15 – 3D модель Football

В Unity Material - це компонент, який використовується для відображення візуальних елементів у сцені. Кожен Material містить набір властивостей, які визначають вигляд об'єкта, на який він накладається.

Material може містити інформацію про текстури, колір, прозорість, відбивання світла, тіні та інші ефекти. Кожен Material може бути застосований до одного або більше об'єктів, які мають різні форми та розміри.

У Unity є вбудований редактор Material, який дозволяє створювати нові Material, змінювати їх властивості та налаштовувати параметри відображення. Крім того, можна використовувати готові Material з Asset Store або створювати власні Material з використанням програм зовнішньої графіки, таких як Photoshop або GIMP. На рисунку 3.16 представлено основні матеріали, які представляють із себе різні кольори.

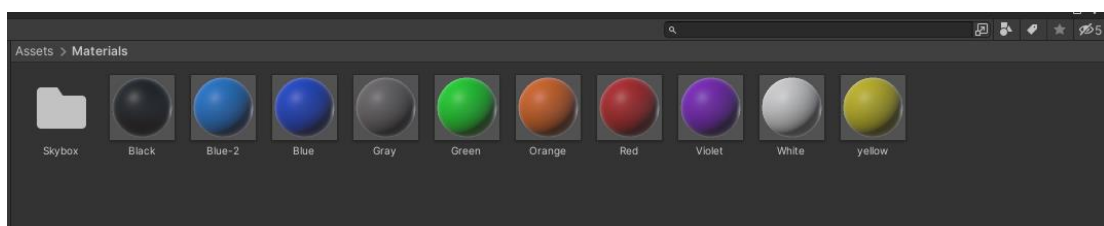


Рисунок 3.16 – Матеріали

Skybox являє собою 6-сторонній куб, який промальовується позаду всієї іншої графіки в грі. У Unity Skybox використовується для створення ефекту безмежного простору навколо сцени, який складається з панорамних зображень.

Skybox може бути створений за допомогою вбудованого редактора Unity або завантажений з Asset Store. В якості Skybox можуть використовуватися текстури або зображення з високою роздільною здатністю, які містять відображення хмар, сонця, місяця, зірок та інші деталі

Skybox відображається як сферична оболонка, яка оточує всю сцену. При переміщенні камери Skybox змінює своє положення та орієнтацію, щоб створити ілюзію руху камери в космосі або на поверхні Землі.

Для всіх сцен окрім сцени «EndGame» був створений Skybox, який складається з 6 зображень. Skybox представлений на рисунку 3.17.

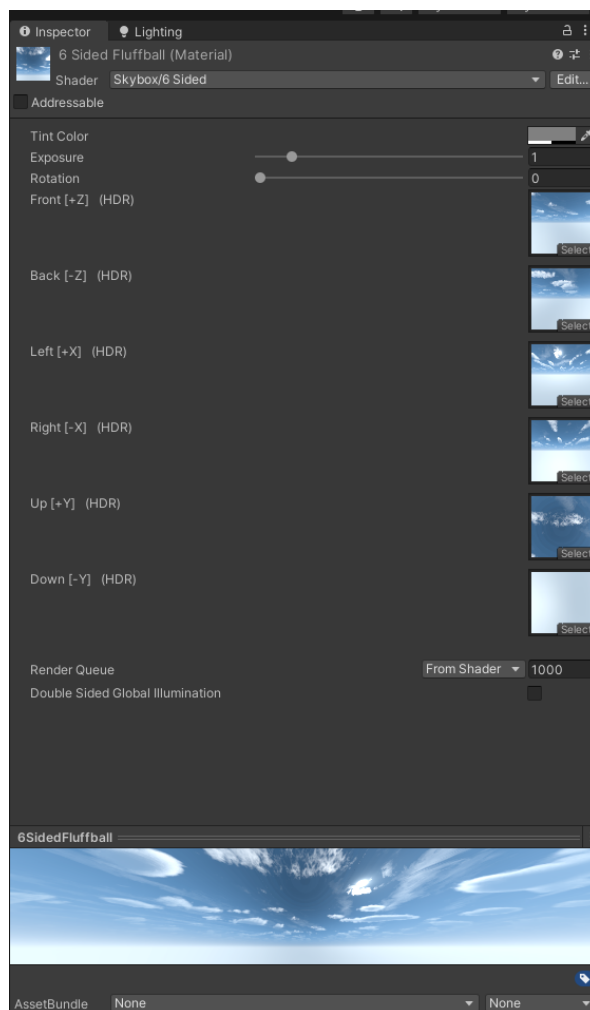


Рисунок 3.17 – Матеріал Skybox

Prefab - це об'єкт, що містить усю необхідну інформацію про об'єкт: модель, матеріали, компоненти, позицію, розмір, поворот тощо. Prefab може містити як один, так і декілька ігрових об'єктів.

Один з головних функціональних принципів Prefab в Unity полягає в тому, що ви можете створювати кілька копій одного і того ж Prefab об'єкту і розміщувати їх на сцені. Це дозволяє ефективно використовувати Prefab об'єкти як шаблони для створення множини однотипних об'єктів, наприклад, дерев, кущів, меблів.

Крім того, використання Prefab об'єктів може зменшити час розробки та спростити процес роботи з об'єктами в Unity, оскільки ви можете редагувати всі екземпляри Prefab об'єкту одночасно, а будь-які зміни, які ви робите на самому Prefab об'єкті, автоматично відображаються в усіх його екземплярах.

Для розробки префабів використовувалися наступні компоненти ігрових об'єктів: `Unity:Transform`, `Box Collider`, `Sphere Collider`, `Mesh Collider`, `Rigidbody`, `AudioSource`, `Mesh Renderer`.

Компонент `Transform` (трансформація) використовується для зберігання значень позиції, обертання, розмірів та стану успадкування `GameObject`'а, тому він дуже важливий. До `GameObject` завжди доданий компонент `Transform` - його неможливо видалити або створити `GameObject` без нього.

Компонент `Sphere Collider` - це колайдер, який представляє собою сферичну область. Він використовується для виявлення зіткнень з об'єктами, які мають форму сфери.

Компонент `Box Collider` - це колайдер, який представляє собою прямокутний паралелепіпед або коробку. Він використовується для виявлення зіткнень з об'єктами, які мають форму прямокутної коробки або ящика.

Компонент `Mesh Collider` - це колайдер, який використовується для виявлення зіткнень з об'єктами, які мають складну форму. Він може бути створений з власноруч створеної моделі або з імпортованої моделі.

Компонент `Rigidbody` - це основний компонент, що включає фізичну поведінку для об'єкта. З прикріпленням `Rigidbody` об'єкт негайно почне реагувати на

гравітацію. Якщо додано один або кілька компонентів Collider, то при колізіях (зіткненнях) об'єкт пересуватиметься.

Компонент Audio Source - відповідає за відтворення звукових ефектів.

Компонент Mesh Renderer - відповідає за візуалізацію 3D-моделі на екрані.

Основні префаби для створення рівнів: Box, FallPlatform, FinishPlatform, LongPlatform, MainPlayer, MovingPlatform, Platform1x1, Platform1x2, Platform1x3, Platform1x4, Platform2x2, Platform3x3, Platform4x4, Platform10x10, PlayerDeath, RotatePlatform, Spike, StartPlatform, Teleport.

Префаб MainPlayer складається з 3D моделі, MainCamera та 4 скриптів. 3D модель представлена на рисунку 3.18. Скрипт, який відповідає за управління камерою представлено в додатку А.3



Рисунок 3.18 – 3D модель префабу MainPlayer

Функцію управління головним персонажем виконує скрипт представлений в лістингу 3.13

Лістинг 3.13 – Скрипт управління персонажем

```
using UnityEngine;
```

```
public class PlayerController : MonoBehaviour
{
    [SerializeField]
    private float speed = 10;
    [SerializeField]
    private Camera followCamera;
```

```

private Rigidbody rb;

void Start()
{
    rb = GetComponent<Rigidbody>();
}

void FixedUpdate()
{
    Move();
}

private void Move()
{
    float h = Input.GetAxis("Horizontal");
    float v = Input.GetAxis("Vertical");
    Vector3 movement = Quaternion.Euler(0,
followCamera.transform.eulerAngles.y, 0) * new Vector3(h, 0, v);
    rb.AddForce(movement * speed);
}
}

```

Функціонал смерті персонажа виконує скрипт наведений в лістингу 3.14.

Лістинг 3.14 – Скрипт смерті персонажа

```

using UnityEngine;

public class PlayerDeath : MonoBehaviour
{
    [SerializeField]
    private GameObject DeahtBody;

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "DeathZone")
        {
            Death();
        }
    }

    private void Death()
    {
        Destroy(gameObject);
        Instantiate(DeahtBody, transform.position,
transform.rotation);
    }
}

```

Функціонал перезавантаження рівня при падінні виконує скрипт наведений в лістингу 3.15.

Лістинг 3.15 – Скрипт перезавантаження рівня при падінні

```
using UnityEngine.SceneManagement;
using UnityEngine;

public class RestartLevelOnFall : MonoBehaviour
{
    [SerializeField]
    private float _fallLimit = -30f;

    private void FixedUpdate()
    {
        if (transform.position.y < _fallLimit)
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }
    }
}
```

Префаб Вох складається з 3D моделі та 1 скрипту. 3D модель представлена на рисунку 3.19. Скрипт наведений в лістингу 3.16.

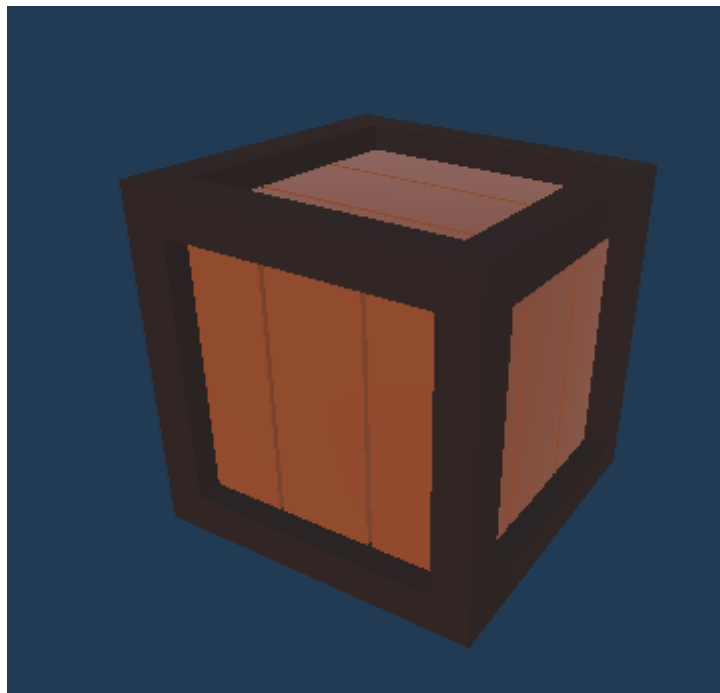


Рисунок 3.19 – 3D модель префабу Вох

Лістинг 3.16 – Скрипт видалення об'єкту при падінні

```
using UnityEngine;

public class DeleteOnFall : MonoBehaviour
{
    private float _fallLimit = -30f;

    private void FixedUpdate()
    {
        if (transform.position.y < _fallLimit)
        {
            Destroy(gameObject, 10);
        }
    }
}
```

Префаб FallPlatform складається з 3D моделі та 1 скрипту. 3D модель представлена на рисунку 3.20. Скрипт наведений в лістингу 3.17.

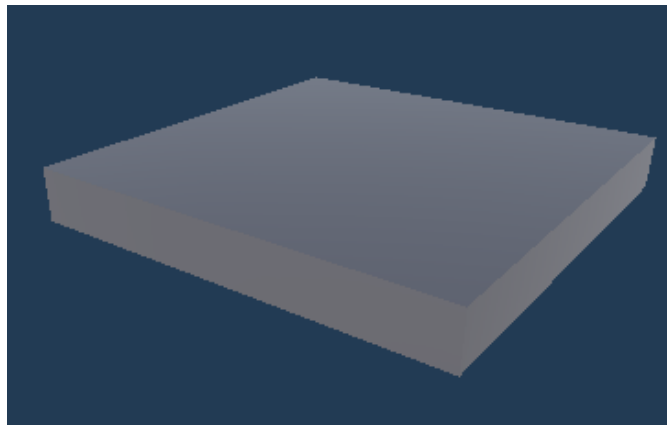


Рисунок 3.20 – 3D модель префабу FallPlatform

Лістинг 3.17 – Скрипт падаючої платформи

```
using System.Collections;
using UnityEngine;

public class FallPlatform : MonoBehaviour
{
    private bool isFalling = false;
    private float downSpeed = 0;
    private float fallDelay = 0.6f;

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "Player")
```

```

        {
            StartCoroutine(FallAfterDelay());
        }
    }

    private void FixedUpdate()
    {
        if (isFalling)
        {
            downSpeed += Time.deltaTime / 20;
            transform.position = new Vector3(transform.position.x,
transform.position.y - downSpeed, transform.position.z);
        }
    }

    IEnumerator FallAfterDelay()
    {
        yield return new WaitForSeconds(fallDelay);
        isFalling = true;
        Destroy(gameObject, 10);
    }
}

```

Префаб RotatePlatform складається з 3D моделі та 1 скрипту. 3D модель представлена на рисунку 3.21. Скрипт наведений в лістингу 3.18.

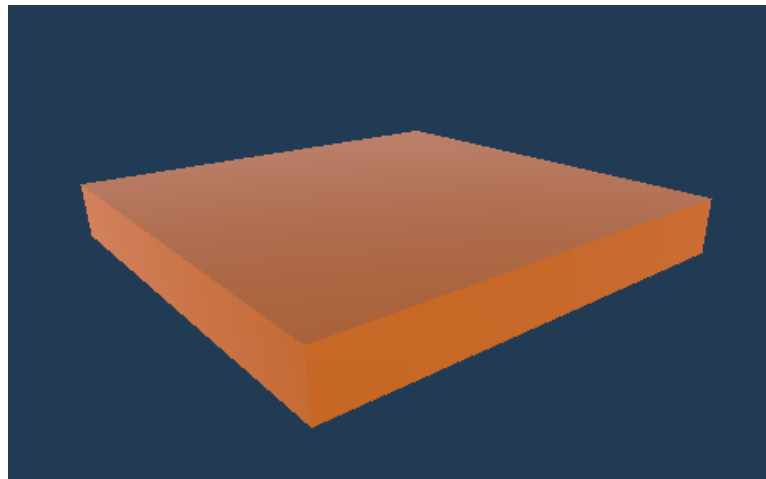


Рисунок 3.21 – 3D модель префабу RotatePlatform

Лістинг 3.18 – Скрипт обертаючої платформи

```

using System.Collections;
using UnityEngine;

public class RotatePlatform : MonoBehaviour
{
    private float rotationSpeed = 1f;
    private float currentAngle = 0f;
    private float rotationDirection = 1f;
    private bool isRotating = true;
    private float delayAfterFullRotation = 2f;

    private void FixedUpdate()
    {
        if (isRotating)
        {
            Rotation();
        }
    }

    public void Rotation()
    {
        currentAngle += rotationSpeed * rotationDirection;
        transform.rotation *= Quaternion.AngleAxis(rotationSpeed *
rotationDirection, Vector3.left);

        if (currentAngle >= 360f)
        {
            isRotating = false;
            currentAngle = 0f;
        }
    }

    StartCoroutine(WaitAndStartRotation(delayAfterFullRotation));
}

IEnumerator WaitAndStartRotation(float delay)
{
    yield return new WaitForSeconds(delay);
    isRotating = true;
}
}

```

Префаб Teleport складається з 3D моделі та 1 скрипту. 3D модель представлена на рисунку 3.22. Скрипт наведений в лістингу 3.19.

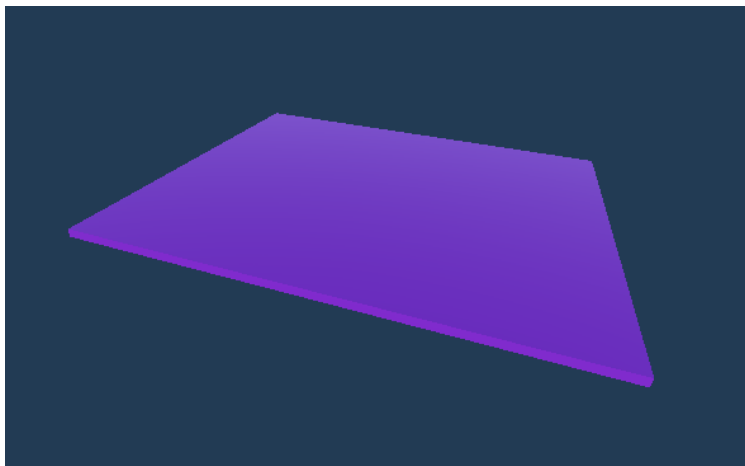


Рисунок 3.22 – 3D модель префабу Teleport

Лістинг 3.19 – Скрипт телепорту

```
using UnityEngine;

public class TeleportPlatform : MonoBehaviour
{
    [SerializeField]
    private Transform playerTransform, destination;
    [SerializeField]
    private GameObject player;
    [SerializeField]
    private AudioSource soundEffect;

    private void Awake()
    {
        soundEffect = gameObject.GetComponent<AudioSource>();
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            soundEffect.PlayOneShot(soundEffect.clip, 1);
            player.SetActive(false);
            playerTransform.position = destination.position;
            player.SetActive(true);
        }
    }
}
```

Префаб MovingPlatform, складається з 3D моделі, 2 порожніх ігрових об'єктів, які відповідають за координати переміщення та 1 скрипту. 3D модель представлена на рисунку 3.23. Скрипт наведений в додатку А.4

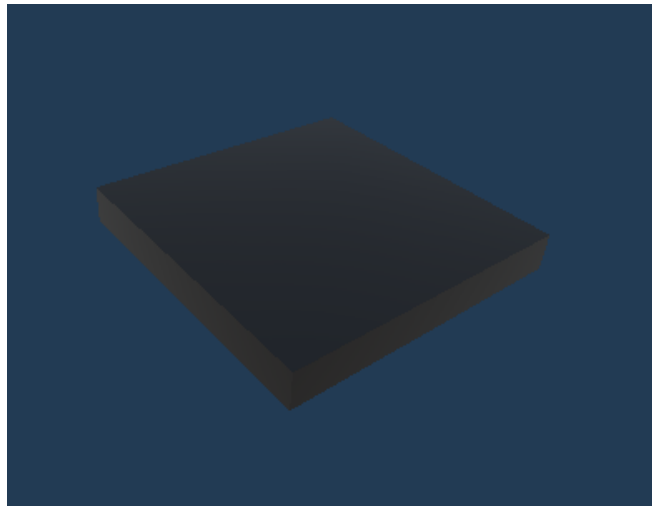


Рисунок 3.23 – 3D модель префабу MovingPlatform

Префаб Platform1x1 складається з 3D моделі відображеної на рисунку 3.24.

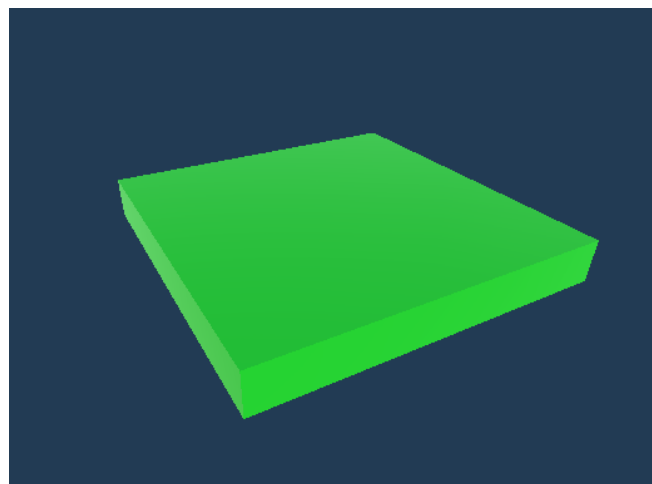


Рисунок 3.24 – 3D модель префабу Platform1x1

Префаби: Platform1x2, Platform1x3, Platform1x4, Platform2x2, Platform3x3, Platform4x4, Platform10x10, LongPlatform відрізняються від Platform1x1 лише розміром.

На рисунку 3.25 представлені 3D моделі, з яких складається префаб PlayerDeath.

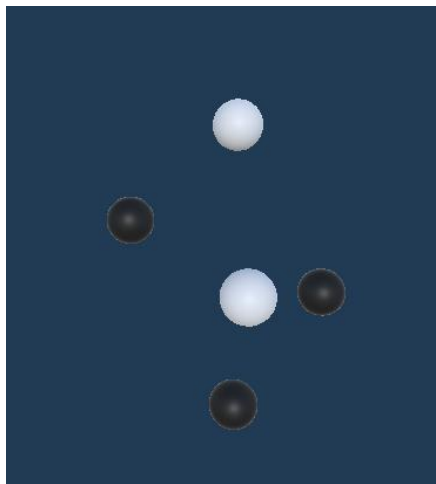


Рисунок 3.25 – 3D моделі префабу PlayerDeath

Префаб FinishPlatform складається з 3D моделі відображеної на рисунку 3.26.

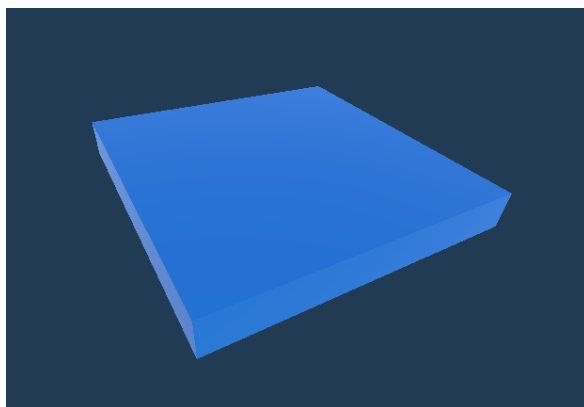


Рисунок 3.26 – 3D модель префабу FinishPlatform

Префаб StartPlatform складається з 3D моделі відображеної на рисунку 3.27.

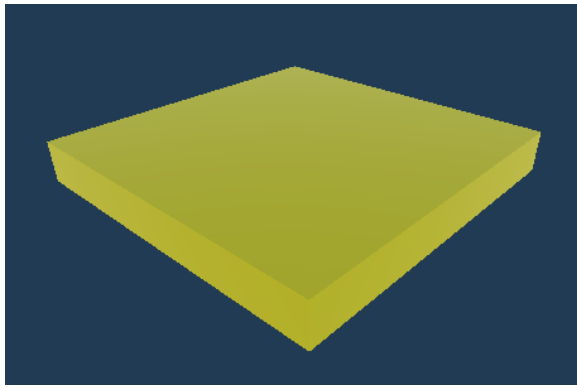


Рисунок 3.27 – 3D модель префабу StartPlatform

Префаб Spike складається з 3D моделі, яка представлена на рисунку 3.28.

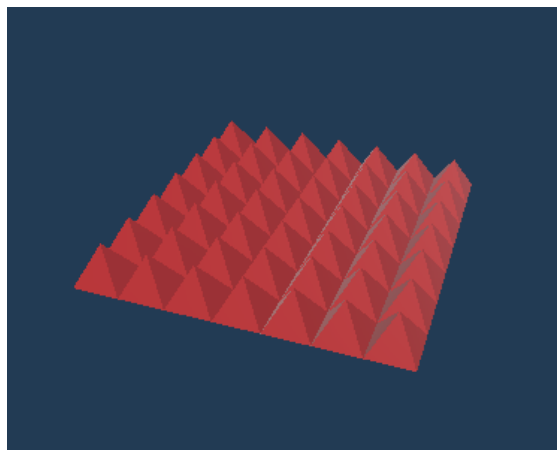


Рисунок 3.28 – 3D модель префабу Spike

3.5 Розробка рівнів

Для гри розроблено 10 рівнів. Кожен рівень створений на окремій сцені. Для побудови рівнів використовувалися префаби. На рисунках 3.29 – 3.38 представлені створені рівні з 1 по 10.

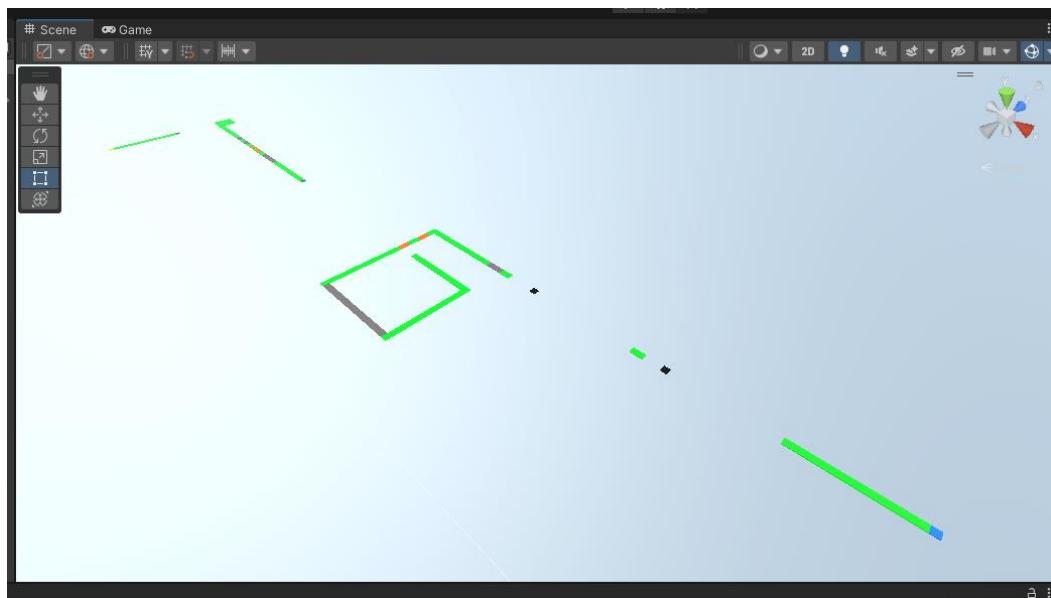


Рисунок 3.29 – Рівень 1

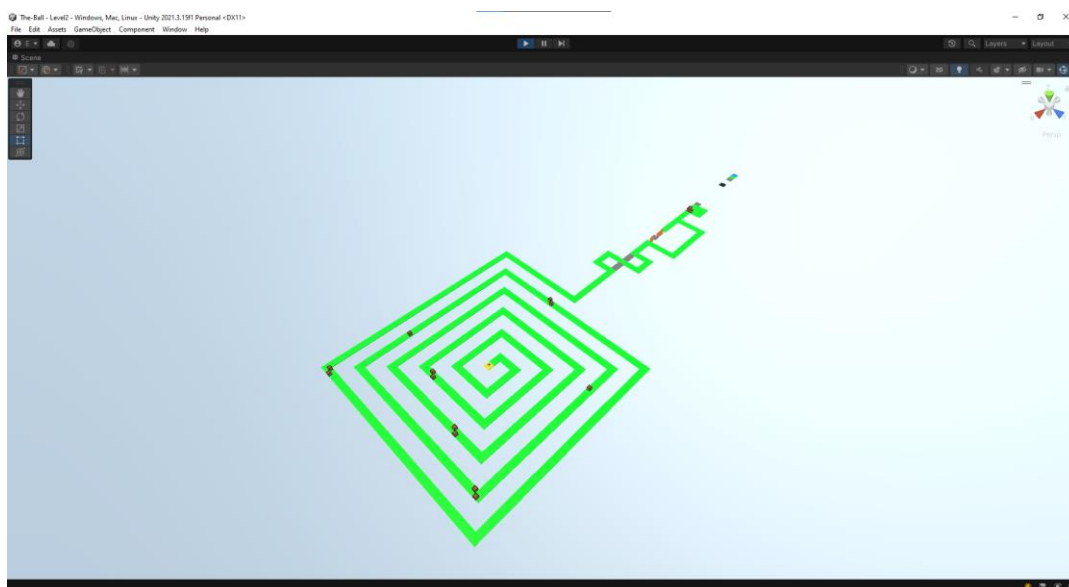


Рисунок 3.30 – Рівень 2

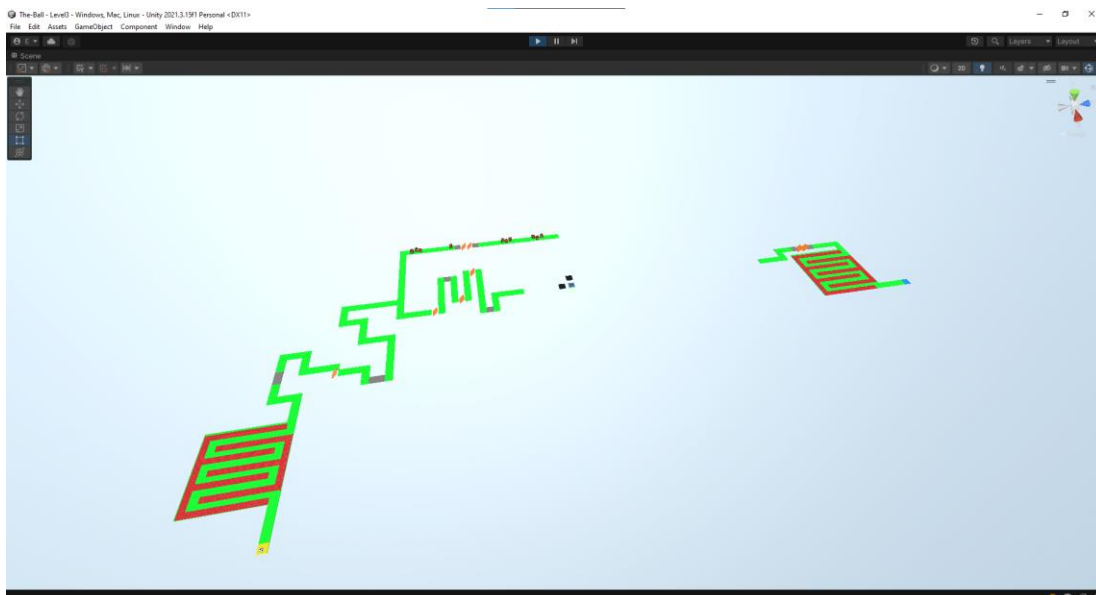


Рисунок 3.31 – Рівень 3

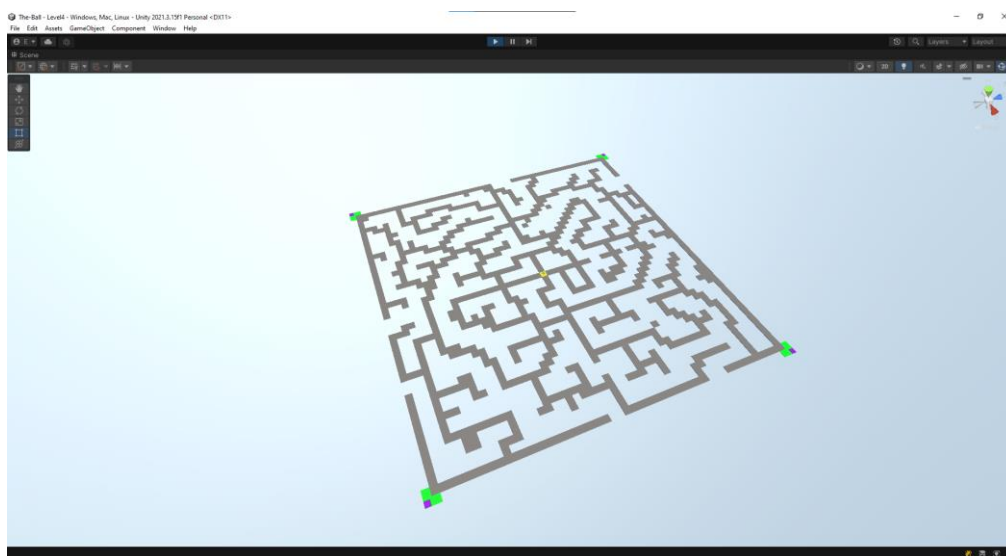


Рисунок 3.32 – Рівень 4

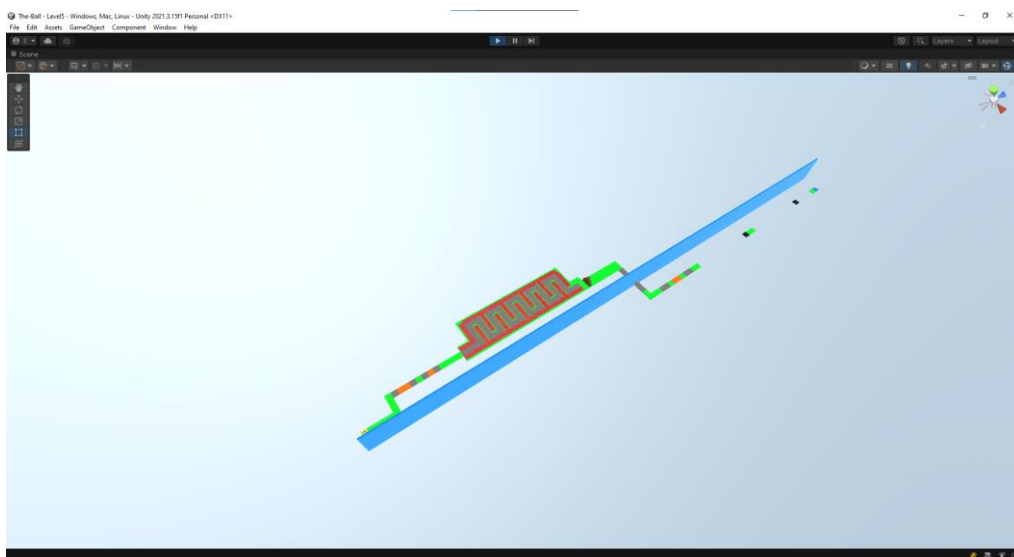


Рисунок 3.33 – Рівень 5

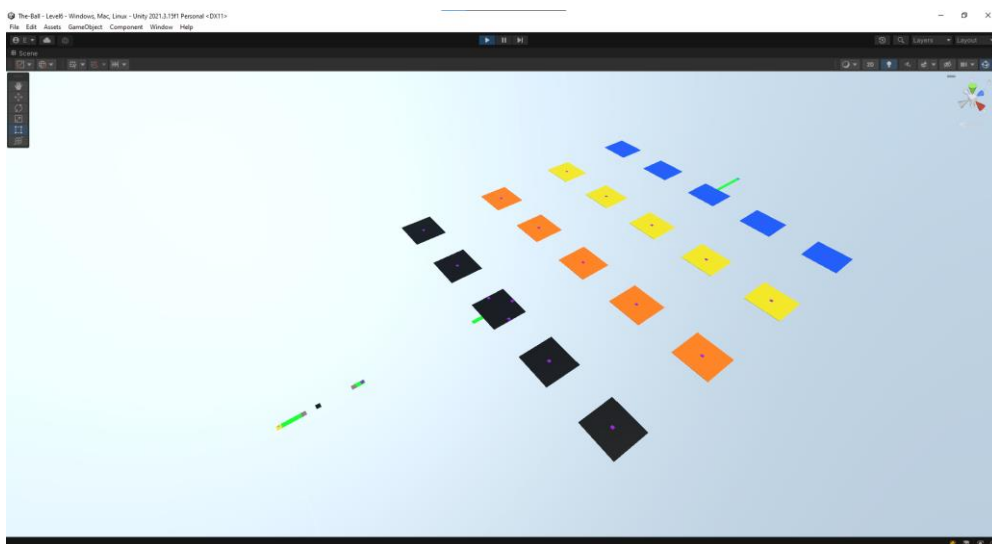


Рисунок 3.34 – Рівень 6

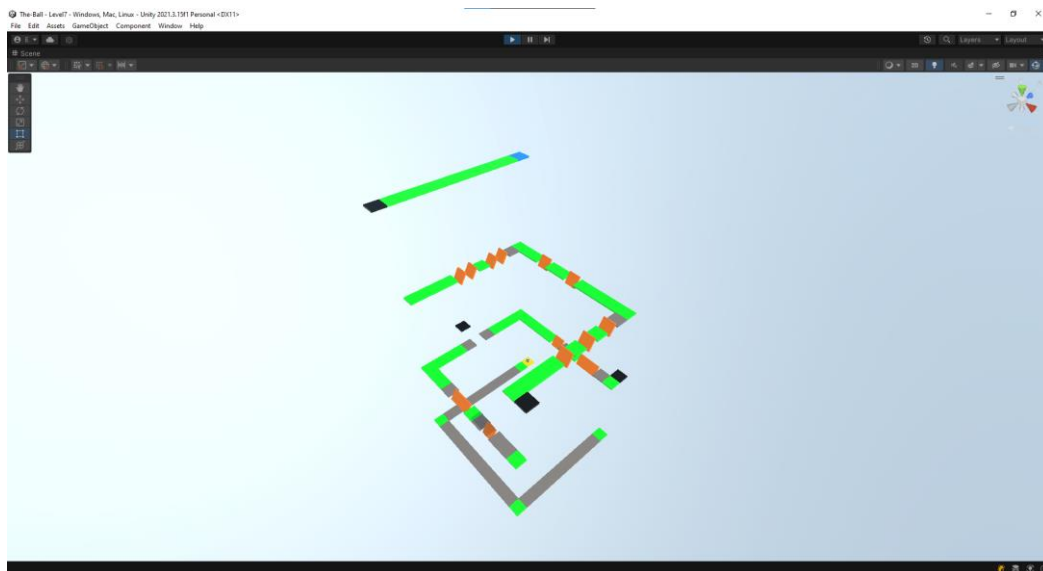


Рисунок 3.35 – Рівень 7

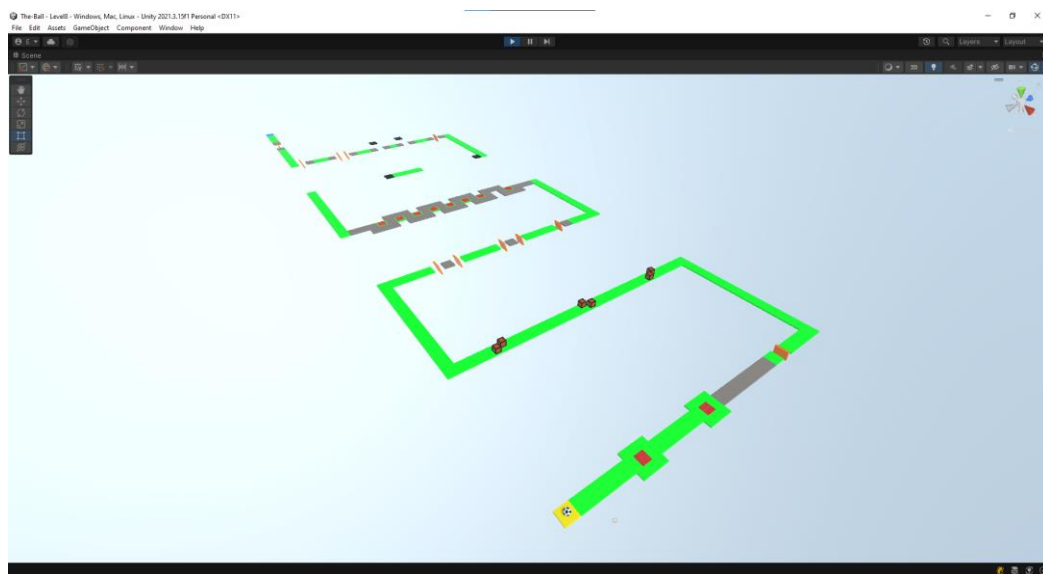


Рисунок 3.36 – Рівень 8

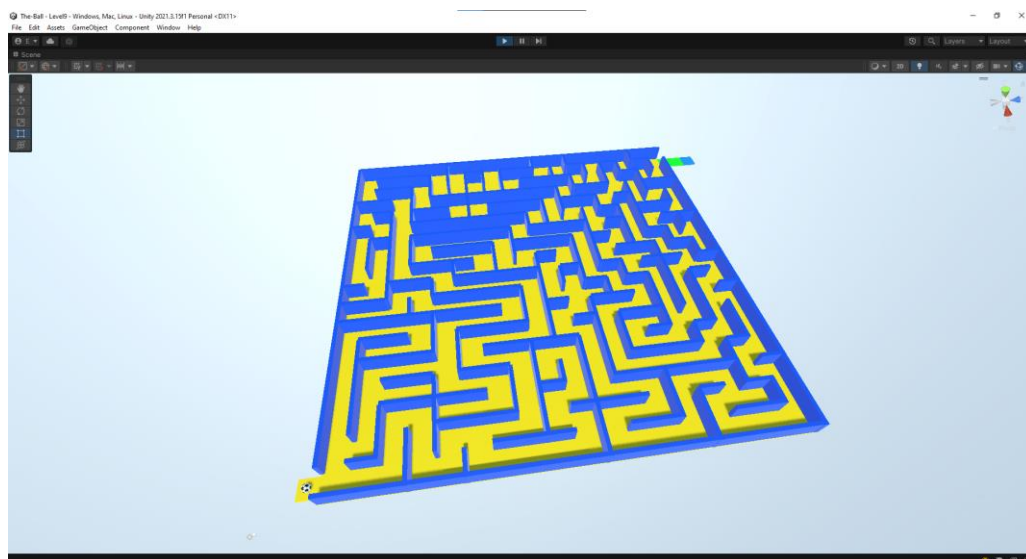


Рисунок 3.37 – Рівень 9

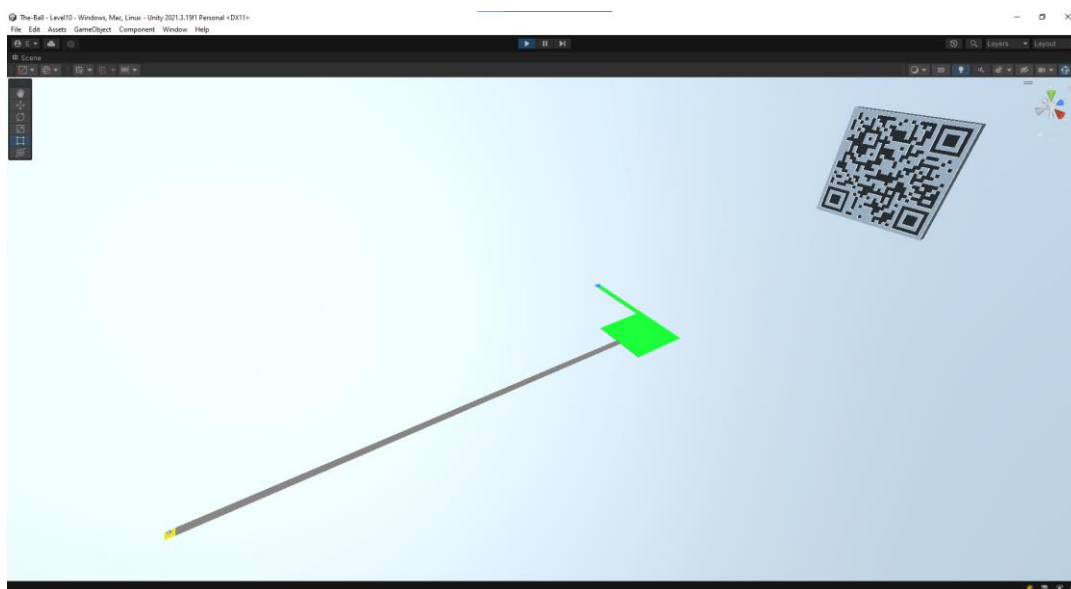


Рисунок 3.38 – Рівень 10

Після проходження останнього рівня завантажиться сцена «EndGame», в якій знаходиться текстове повідомлення і кнопка завантаження головного меню. Вигляд сцени «EndGame» представлено на рисунку 3.39

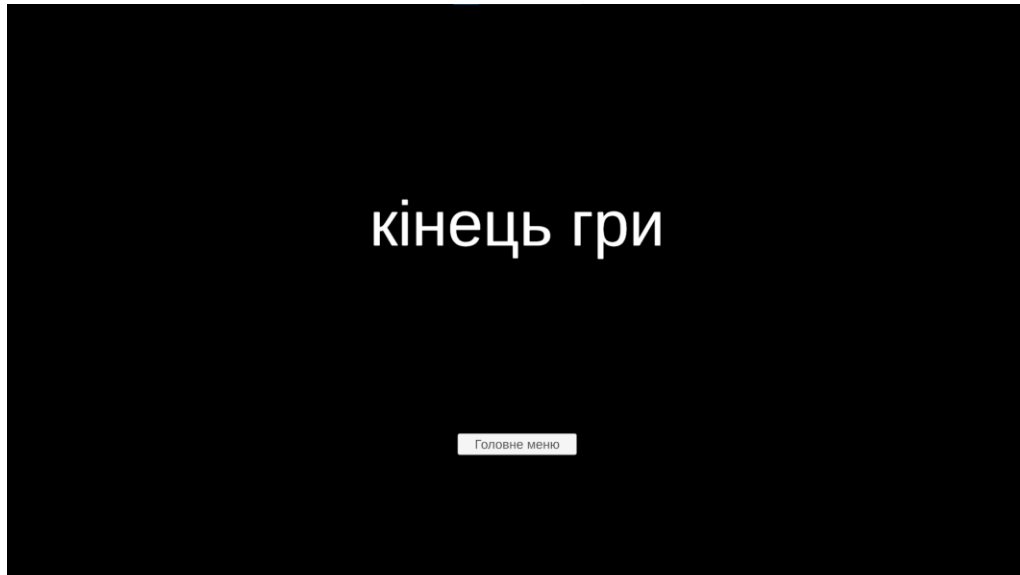


Рисунок 3.39 – Сцена «EndGame»

Функціонал кнопки реалізує скрипт наведений в лістингу 3.20

Лістинг 3.20 – Скрипт переходу в головне меню

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class GoToMainMenu : MonoBehaviour
{
    public void LoadMenu()
    {
        SceneManager.LoadScene(0);
    }
}
```

3.6 Локалізація гри

Для локалізації гри встановлено пакет Unity Localization package.

Unity Localization package - це набір інструментів, які дозволяють розробникам додавати підтримку багатомовності до своїх проектів Unity.

Пакет локалізації включає такі функції, як:

- локалізація тексту;
- локалізація ресурсів;
- інструменти попереднього перегляду та тестування.

В налаштуваннях проекту в розділі Localization додані 2 мови українська та англійська. Також налаштована мова за замовчуванням українська. Налаштування локалізації в проекті представлено на рисунку 3.40.

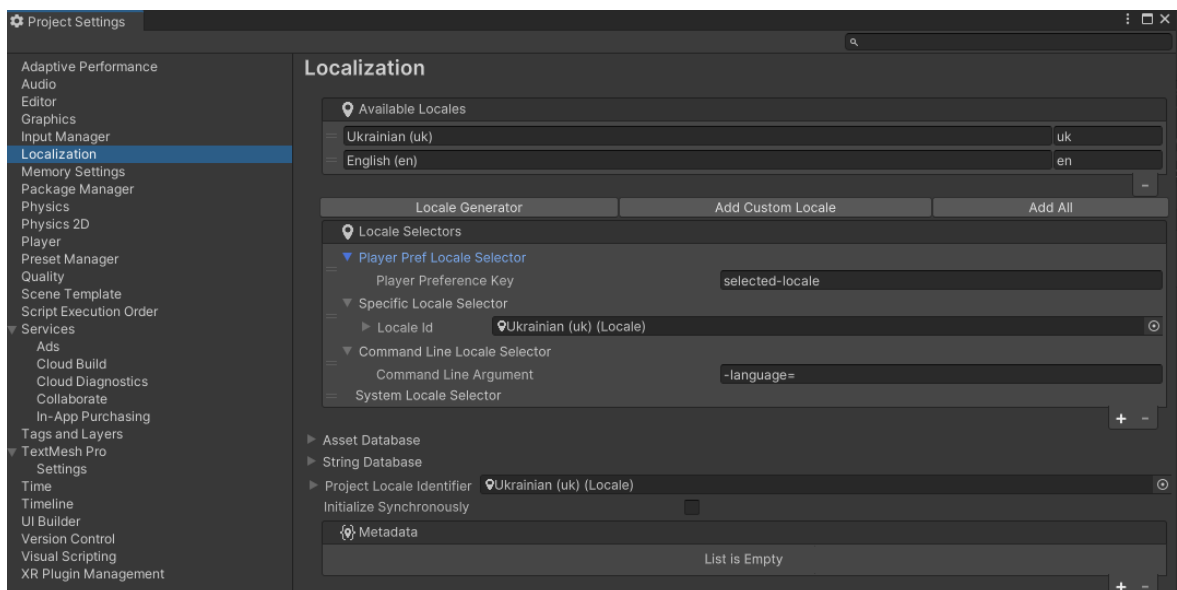


Рисунок 3.40 – Налаштування локалізації в проекті

Для локалізації основного тексту інтерфейсу гри створена таблиця UI Text, яка наведена в таблиці 3.1.

Таблиця 3.1 – Вміст таблиці UI Text

Key	Ukrainian (uk)	English (en)
StartBtnScene0	Почати Гру	Start Game
SettingsBtnScene0	Налаштування	Settings
ExitBtnScene0	Вихід	Exit Game
LevelSelectHeaderTextScene0	Вибір Рівня	Select Level
BackBtn	Назад	Back
ResolutionSettingsTextScene0	Налаштування роздільної здатності	Resolution settings
FullScreenModeTextScene0	Повноекранний режим	Full screen mode
GraphicsSettingsTextScene0	Налаштування графіки	Graphics settings
MusicSettingsTextScene0	Гучність музики	Mucis volume
SoundEffectsSettingsTextScene0	Гучність звукових ефектів	Volume of sound effects
CameraSensitivityTextScene0	Чутливість камери	Camera sensitivity
LanguageSettingsTextScene0	Вибір мови	Language selection
SaveBtnScene0	Зберегти	Save
LoadingText	Завантаження	Loading
RestartBtn	Перезапуск	Restart
ContinueBtn	Продовжити	Continue
EndGameTextSceneLast	кінець гри	The End
MainMenu	Головне меню	Main menu
CharacterControlBtn	Управління персонажем	Character control
CatalogBtn	Довідник	Manual
PauseMenu	Меню паузи	Pause Menu
ControlCharacterOr	Або	Or

Для локалізації частини інтерфейсу довідник створена таблиця Manual Text, яка представлена в таблиці 3.2.

Таблиця 3.2 – Вміст таблиці Manual Text

Key	Ukrainian (uk)	English (en)
FallingplatformHeader	падаюча платформа	Falling platform
FallingplatformText	платформа падає через декілька секунд після доторкання персонажа до неї	The platform falls a few seconds after the character touches it
BoxHeader	ящик	Box
BoxText	ящик представляє собою перешкоду яку можна рухати	The box is an obstacle that can be moved
PlatformHeader	платформа	Platform
PlatformText	звичайна платформа по якій можна рухатись	An ordinary platform on which you can move
FinishPlatformHeader	фінішна платформа	Finish platform
FinishPlatformText	платформа вказує на закінчення рівня при доторканні персонажа закінчує поточний рівень та запускає наступний	The platform indicates the end of the level. Touching a character ends the current level and starts the next one
StartPlatformHeader	стартова платформа	Start platform
StartPlatformText	платформа вказує на початок рівня, персонаж при програші або перезапуску рівня буде переміщуватися на стартову платформу	The platform indicates the beginning of the level. The character when losing or restarting the level will be moved to the starting platform
SpikeHeader	шипи	Spikes
SpikeText	при доторканні персонажа до шипів рівень перезапускається	When the character touches the spikes, the level restarts
MovingPlatformHeader	рухома платформа	Moving platform
MovingPlatformText	рухома платформа переміщується від однієї точки до іншої, при досягненні будь якої точки очікує декілька секунд	A moving platform moves from one point to another. When reaching any point, it waits for a few seconds
TeleportHeader	телепорт	Teleport
TeleportText	телепорт переміщує персонажа до іншої частини рівня	Teleport moves the character to another part of the level
RotatingPlatformHeader	обертова платформа	Rotating platform
RotatingPlatformText	обертові платформи може бути двох типів ті що обертаються горизонтально та ті що обертаються вертикально, платформа робить оберт 360 градусів і після цього очікує декілька секунд	Rotating platforms can be of two types: those that rotate horizontally and those that rotate vertically. The platform rotates 360 degrees and then waits for a few seconds

Після створення таблиць локалізацій до кожного об'єкт тексту у проекті додано компонент Localize String Event.

Localize String Event - це компонент пакету Unity Localisation, який дозволяє замінювати рядки тексту, що відображаються в грі, залежно від вибраної мови користувача.

Цей компонент дозволяє пов'язати ключ локалізації з відображуваним текстом, що дозволяє розмістити всі необхідні рядки тексту у файлах локалізації. Коли гравець змінює мову в налаштуваннях гри, Localize String Event змінює текст, відображений на екрані, відповідно до вибраної мови. Приклад налаштованого компоненту наведений на рисунку 3.41.

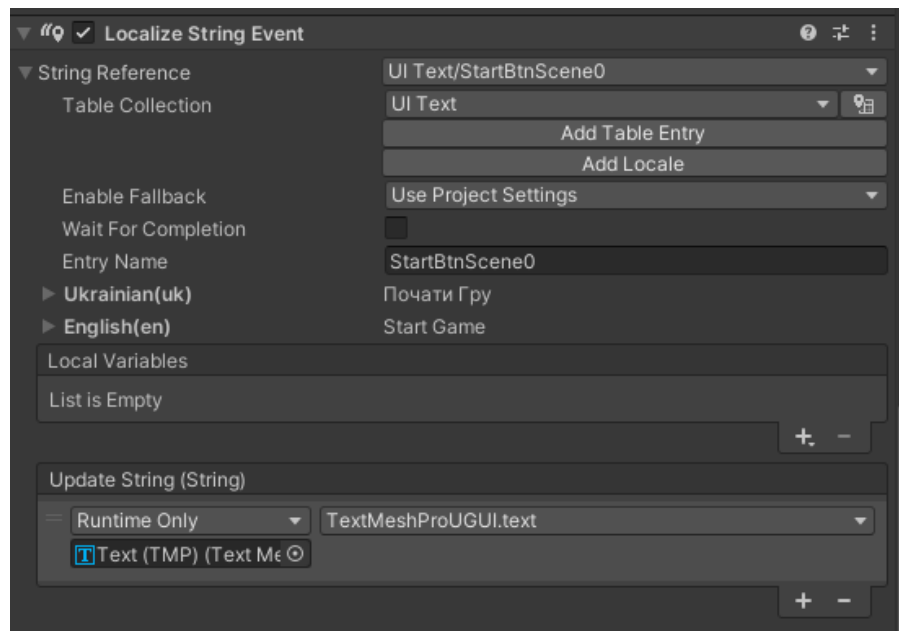


Рисунок 3.41 – Налаштування компоненту Localize String Event

3.7 Збірка гри

Збірка гри - це процес перетворення розробленої гри з проекту Unity в готовий до виконання файл. Під час збірки гри Unity компілює весь скриптовий код, обробляє ресурси (зображення, звуки, моделі тощо) і об'єднує їх у готовий продукт. Готовий виконавчий файл можна запустити на різних платформах в залежності від обраної цільової платформи, наприклад: Windows, MacOS, iOS, Android тощо.

В Build Settings додані усі сцени гри та обрана цільова платформа Windows. Налаштування Build Settings наведено на рисунку 3.42.

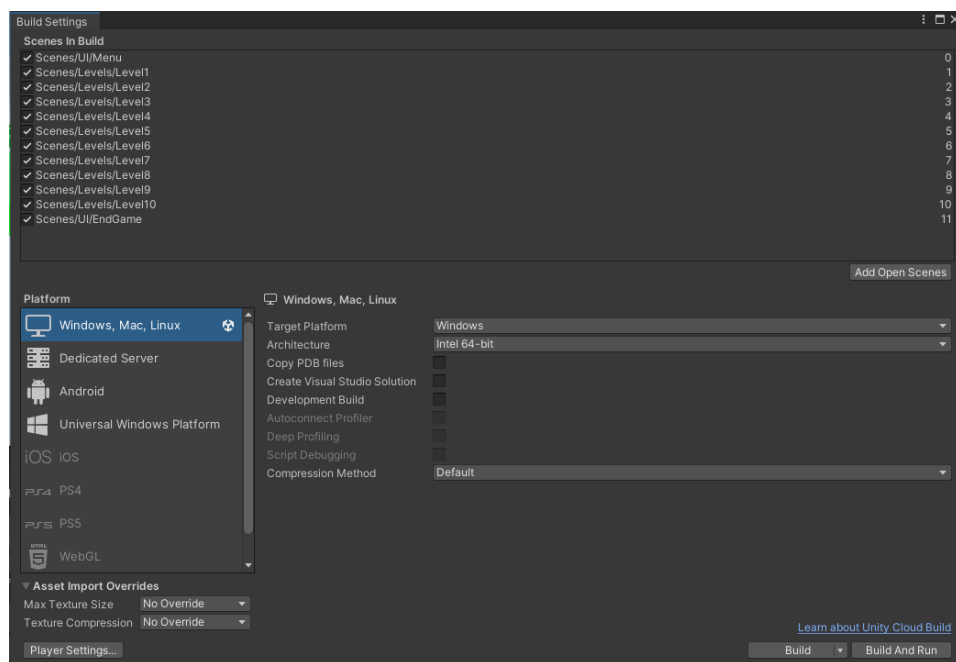


Рисунок 3.42 – Налаштування Build Settings

Процес збірки гри наведено на рисунку 3.43.

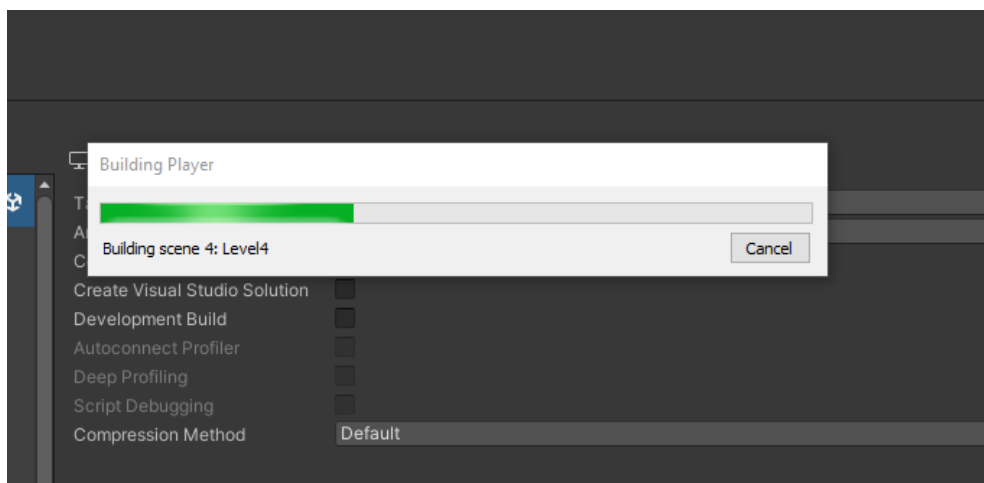


Рисунок 3.43 – Процесс збірки гри

Після процесу збірки гри отримаємо всі необхідні файли для роботи гри та виконавчий файл гри. Результат збірки гри наведено на рисунку 3.44.

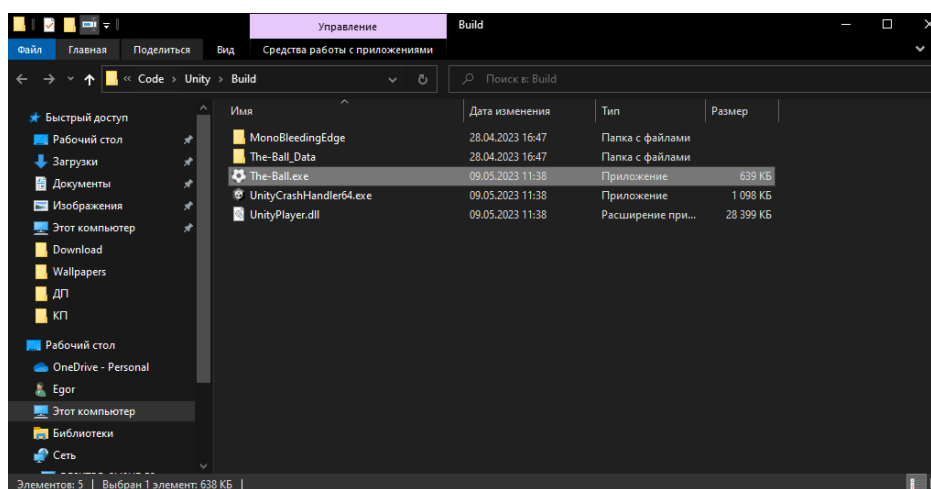


Рисунок 3.44 – Результат збірки гри

3.8 Тестування

Тестування гри дозволяє виявити та виправити проблеми, покращити якість та взаємодію користувача з грою.

Після розробки гра може містити значні помилки, як відсутність певних можливостей, музики або об'єктів. Виявленням проблем займаються тестувальники, котрі грають в гру, намагаючись сповна скористатися всіма доступними можливостями в ній.

Для тестування гри використовувались наступні тести:

- функціональне тестування: перевірка правильності роботи всіх функцій гри, включаючи головний геймплей, керування, взаємодію з об'єктами, та інші геймплейні аспекти.;

- тестування графіки та аудіо: оцінка якості матеріалів, 3D моделей, анімацій, а також звукових ефектів, музики;

- тестування інтерфейсу користувача: перевірка зручності та ефективності використання інтерфейсу гри, включаючи меню, налаштування, підказки та інші елементи.;

- тестування сумісності: перевірка сумісності гри з різними операційними системами, апаратними конфігураціями, графічними картами та іншими аспектами апаратного забезпечення;

- тестування продуктивності: оцінка продуктивності гри на різних системах, оцінка кадрової частоти.

В таблицях 3.3 – 3.6 представлено конфігурації обладнання на якому виконувалося тестування.

Таблиця 3.3 – Конфігурація комп'ютера

Характеристика	Значення
Процесор	AMD Athlon II X2 250 3.0 GHz
Оперативна пам'ять	4 ГБ DDR3
Відеокарта	GeForce GT 240 512MB GDDR3
Операційна система	Windows 10 64-bit

Таблиця 3.4 – Конфігурація комп'ютера

Характеристика	Значення
Процесор	AMD Ryzen 5 5500 3.6GHz/16MB
Оперативна пам'ять	16 ГБ DDR4
Відеокарта	Asus GeForce GTX 1050 Ti ROG Strix 4GB GDDR5 (128bit)
Операційна система	Windows 10 64-bit

Таблиця 3.5 – Конфігурація ноутбуку

Характеристика	Значення
Процесор	AMD Ryzen 3 5300U (2.6 - 3.8 ГГц)
Оперативна пам'ять	8 ГБ DDR4
Відеокарта	Інтегрована в процесор Radeon RX Vega 6 512MB
Операційна система	Windows 11 64-bit

Таблиця 3.6 – Конфігурація комп'ютера

Характеристика	Значення
Процесор	AMD Ryzen 5 4500 3.6GHz/8MB
Оперативна пам'ять	16 ГБ DDR4
Відеокарта	Sapphire Radeon RX 550 Pulse 4GB GDDR5 (128bit)
Операційна система	Windows 10 64-bit

3.9 Створення інсталятора

Inno Setup - це безкоштовна програма з відкритим вихідним кодом для створення програмних інсталяторів для Windows. За допомогою майстра створення інсталятора створений скрипт наведений в лістингу 3.21.

Лістинг 3.21 – Скрипт для створення інсталятора

```
[Setup]
AppId={{72B704E4-B385-4E4E-B551-D0F48E7E5842}}
AppName=The-Ball
AppVersion=1.0
AppPublisher=KnyrykCompany, Inc.
AppPublisherURL=https://github.com/Knirik-Yegor/Build-The-Ball
AppSupportURL=https://github.com/Knirik-Yegor/Build-The-Ball
AppUpdatesURL=https://github.com/Knirik-Yegor/Build-The-Ball
DefaultDirName={autopf}\The-Ball
DefaultGroupName=The-Ball
AllowNoIcons=yes

OutputDir=D:\Code\Unity
OutputBaseFilename=SetupThe-Ball
Compression=lzma
SolidCompression=yes
WizardStyle=modern

[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"
Name: "ukrainian"; MessagesFile: "compiler:Languages\Ukrainian.isl"

[Tasks]
```

```

Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}";
GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]
Source: "D:\Code\Unity\Build\The-Ball.exe"; DestDir: "{app}"; Flags:
ignoreversion
Source: "D:\Code\Unity\Build\*"; DestDir: "{app}"; Flags:
ignoreversion recursesubdirs createallsubdirs

[Icons]
Name: "{group}\The-Ball"; Filename: "{app}\The-Ball.exe"
Name: "{autodesktop}\The-Ball"; Filename: "{app}\The-Ball.exe";
Tasks: desktopicon

[Run]
Filename: "{app}\The-Ball.exe"; Description: "{cm:LaunchProgram,The-
Ball}"; Flags: nowait postinstall skipifsilent

```

Після запуску на виконання наведеного вище скрипту в Inno Setup отримаємо виконуваний файл, який представлено на рисунку 3.45.

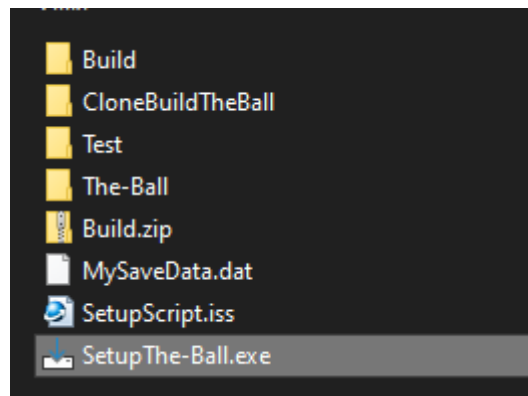


Рисунок 3.45 – Створення інсталятора

ВИСНОВКИ

В даному бакалаврському проекті була розроблена комп'ютерна 3D гра «The-Ball» у жанрі платформер.

Під час виконання бакалаврського проекту виконано:

- аналіз предметної області та визначені основні поняття;
- огляд існуючих ігрових рушіїв;
- проектування гри;
- розробка інтерфейсу гри;
- розробка рівнів гри;
- локалізація гри;
- збірка гри;
- тестування гри;
- створення інсталятора;

Таким чином, можна констатувати, що поставлена мета бакалаврського проекту досягнута. Гра має потенціал і надалі може удосконалюватись, в майбутніх оновленнях можна покращити: готовий функціонал, додати більше різноманітних рівнів, нових ігрових механік.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Усі жанри комп'ютерних ігор URL: <http://uaplay.com.ua/usi-zhanry-pc-ihor/> (дата звернення: 05.03.2023)
2. PEGI Вікові рейтинги URL: <http://pegi.info/uk/page/pegi-age-ratings> (дата звернення: 07.03.2023)
3. PEGI RATING URL: <http://pegi.info/uk/what-do-the-labels-mean> (дата звернення: 07.03.2023)
4. Добірка ігрових рушіїв для розробників URL: <https://dev.ua/news/game-engines-list-1657124713> (дата звернення: 15.03.2023)
5. Unreal Engine URL: <https://www.unrealengine.com/en-US/download> (дата звернення: 15.03.2023)
6. Документація Godot розробників URL: <http://docs.godotengine.org/uk/stable/about/introduction.html> (дата звернення: 16.03.2023)
7. Unity URL: <https://unity.com/pricing#plans-student-and-hobbyist> (дата звернення: 16.03.2023)
8. Blender URL: <https://www.blender.org> (дата звернення: 04.04.2023)
9. Unity Localization package URL: <https://docs.unity3d.com/Packages/com.unity.localization@1.3/manual/> (дата звернення: 22.04.2023)
10. Inno Setup URL: <https://jrsoftware.org/isdl.php> (дата звернення: 29.04.2023)
11. Документація Unity URL: <https://docs.unity3d.com/Manual/index.html> (дата звернення: 01.04.2023)

ДОДАТОК А

Додаток А.1 – Скрипт налаштувань роздільної здатності, повноекранного режиму, графіки

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

public class SettingsMenu : MonoBehaviour
{
    [SerializeField]
    private TMP_Dropdown resolutionDropdown;

    [SerializeField]
    private TMP_Dropdown qualityDropdown;

    [SerializeField]
    private Toggle fullscreenToggle;

    Resolution[] resolutions;

    void Start()
    {
        resolutionDropdown.ClearOptions();
        List<string> options = new List<string>();
        resolutions = Screen.resolutions;
        int currentResolutionIndex = 0;

        for (int i = 0; i < resolutions.Length; i++)
        {
            string option = resolutions[i].width + "x" +
resolutions[i].height + " " + resolutions[i].refreshRate + "Hz";
            options.Add(option);
            if (resolutions[i].width ==
Screen.currentResolution.width && resolutions[i].height ==
Screen.currentResolution.height)
            {
                currentResolutionIndex = i;
            }
        }

        resolutionDropdown.AddOptions(options);
        resolutionDropdown.RefreshShownValue();
        LoadSettings(currentResolutionIndex);
    }
}
```

```

public void SetFullscreen(bool isFullscreen)
{
    Screen.fullScreen = isFullscreen;
    fullscreenToggle.isOn = isFullscreen;
}

public void SetResolution(int resolutionIndex)
{
    Resolution resolution = resolutions[resolutionIndex];
    Screen.SetResolution(resolution.width, resolution.height,
Screen.fullScreen);
}

public void SetQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
}

public void SaveSettings()
{
    PlayerPrefs.SetInt("QualitySettingPreference",
qualityDropdown.value);
    PlayerPrefs.SetInt("ResolutionPreference",
resolutionDropdown.value);
    PlayerPrefs.SetInt("FullscreenPreference",
System.Convert.ToInt32(Screen.fullScreen));
    PlayerPrefs.Save();
}

public void LoadSettings(int currentResolutionIndex)
{
    if (PlayerPrefs.HasKey("QualitySettingPreference"))
    {
        qualityDropdown.value =
PlayerPrefs.GetInt("QualitySettingPreference");
    }
    else
    {
        qualityDropdown.value = 3;
    }

    if (PlayerPrefs.HasKey("ResolutionPreference"))
    {
        resolutionDropdown.value =
PlayerPrefs.GetInt("ResolutionPreference");
    }
    else
    {
        resolutionDropdown.value = currentResolutionIndex;
    }

    if (PlayerPrefs.HasKey("FullscreenPreference"))
    {

```

```

        Screen.fullScreen =
System.Convert.ToBoolean(PlayerPrefs.GetInt("FullscreenPreference"))
;
        fullscreenToggle.isOn =
System.Convert.ToBoolean(PlayerPrefs.GetInt("FullscreenPreference"))
;
    }
    else
    {
        Screen.fullScreen = true;
        fullscreenToggle.isOn = true;
    }
}
}

```

Додаток А.2 – Скрипт меню паузи

```

using UnityEngine;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{
    [SerializeField]
    private GameObject pauseGameMenu;

    private bool PauseGame;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (PauseGame)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseGameMenu.SetActive(false);
        Time.timeScale = 1f;
        PauseGame = false;
        Cursor.lockState = CursorLockMode.Locked;
    }

    public void Pause()
    {

```

```

        pauseGameMenu.SetActive(true);
        Time.timeScale = 0f;
        PauseGame = true;
        Cursor.lockState = CursorLockMode.None;
    }

    public void LoadMenu()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene(0);
    }

    public void Restart()
    {
        Time.timeScale = 1f;

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    public void SetPauseGame(bool pause)
    {
        PauseGame = pause;
    }

    public bool GetPauseGame()
    {
        return PauseGame;
    }
}

```

Додаток А.3 – Скрипт контролер камери

```

using UnityEngine;
using UnityEngine.SceneManagement;

public class CameraController : MonoBehaviour
{
    [SerializeField]
    private PauseMenu pauseMenu;

    [SerializeField]
    private Transform Target;

    [SerializeField]
    private float maxDistance = 20.0f;
    [SerializeField]
    private float minDistance = 5.0f;

    [SerializeField]
    private float smoothTime = 0.2f;

```

```

[SerializeField]
private float distanceFromTarget = 10.0f;

private float mouseSensitivity;

private float rotationY;
private float rotationX;

private Vector3 currentRotation;
private Vector3 smoothVelocity = Vector3.zero;

private float mouseX;
private float mouseY;
private float mouseScrollWheel;

private float timeToRestart = 3f;

void Start()
{
    mouseSensitivity = PlayerPrefs.GetFloat("Sensitivity", 1);
    Cursor.lockState = CursorLockMode.Locked;
}

void Update()
{
    if (Target != null)
    {
        Rotation();
    }
    else
    {
        timeToRestart -= Time.deltaTime;

        if (timeToRestart <= 0f)
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }
    }
}

private void Rotation()
{
    if (!pauseMenu.GetPauseGame())
    {
        mouseX = Input.GetAxis("Mouse X") * mouseSensitivity;
        mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity;
        mouseScrollWheel = Input.GetAxis("Mouse ScrollWheel");
        distanceFromTarget += mouseScrollWheel * 10.0f;

        distanceFromTarget = Mathf.Clamp(distanceFromTarget,
minDistance, maxDistance);
    }
}

```

```

        rotationX += mouseY;
        rotationY += mouseX;

        rotationX = Mathf.Clamp(rotationX, -10, 90);

        transform.localEulerAngles = currentRotation;
        Vector3 nextRotation = new Vector3(rotationX,
rotationY);
        currentRotation = Vector3.SmoothDamp(currentRotation,
nextRotation, ref smoothVelocity, smoothTime);
        transform.position = Target.position - transform.forward
* distanceFromTarget;
    }
}
}

```

Додаток А.4 – Скрипт рухомої платформи

```

using System.Collections;
using UnityEngine;

public class MovingPlatform : MonoBehaviour
{
    [SerializeField]
    private float speed;

    [SerializeField]
    private Transform startPoint, endPoint;

    [SerializeField]
    private float changeDirectionDelay;

    private Transform destinationTarget, departTarget;
    private float startTime;
    private float journeyLength;
    private bool isWaiting;

    void Start()
    {
        departTarget = startPoint;
        destinationTarget = endPoint;

        startTime = Time.time;
        journeyLength = Vector3.Distance(departTarget.position,
destinationTarget.position);
    }

    void FixedUpdate()
    {
        Move();
    }
}

```

```

private void Move()
{
    if (!isWaiting)
    {
        if (Vector3.Distance(transform.position,
destinationTarget.position) > 0.01f)
        {
            float distCovered = (Time.time - startTime) * speed;

            float fractionOfJourney = distCovered /
journeyLength;

            transform.position =
Vector3.Lerp(departTarget.position, destinationTarget.position,
fractionOfJourney);
        }
        else
        {
            isWaiting = true;
            StartCoroutine(changeDelay());
        }
    }
}

void ChangeDestination()
{
    if (departTarget == endPoint && destinationTarget ==
startPoint)
    {
        departTarget = startPoint;
        destinationTarget = endPoint;
    }
    else
    {
        departTarget = endPoint;
        destinationTarget = startPoint;
    }
}

IEnumerator changeDelay()
{
    yield return new WaitForSeconds(changeDirectionDelay);
    ChangeDestination();
    startTime = Time.time;
    journeyLength = Vector3.Distance(departTarget.position,
destinationTarget.position);
    isWaiting = false;
}

void OnTriggerEnter(Collider other)
{

```

```

        if (other.gameObject.tag == "Player" || other.gameObject.tag
== "Box")
        {
            other.transform.parent = transform;

        }

    }

    void OnTriggerExit(Collider other)
    {
        if (other.gameObject.tag == "Player" || other.gameObject.tag
== "Box")
        {
            other.transform.parent = null;

        }

    }

}

```

Додаток А.5 – Скрипт фінішної платформи

```

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

public class TriggerLoadLevel : MonoBehaviour
{
    [SerializeField]
    private Slider slidebar;

    [SerializeField]
    private GameObject loadingScreen;

    [SerializeField]
    private AudioSource soundEffect;

    private bool Loading = false;

    private void Awake()
    {
        soundEffect = gameObject.GetComponent<AudioSource>();
    }

    private void OnTriggerEnter(Collider col)
    {
        if(col.tag == "Player")
        {
            soundEffect.PlayOneShot(soundEffect.clip, 1);
            Debug.Log(SceneManager.sceneCountInBuildSettings-1);
            SaveData data = new SaveData();

```

```

        data = SaveManager.LoadLevel();
        if (data.SaveLevel <
SceneManager.GetActiveScene().buildIndex)
        {
            data.SaveLevel =
SceneManager.GetActiveScene().buildIndex;
            SaveManager.SaveLevel(data);
        }

        if (!Loading)
        {
            loadingScreen.SetActive(true);

StartCoroutine(LoadAsync(SceneManager.GetActiveScene().buildIndex +
1));
        }
    }
}

IEnumerator LoadAsync(int SceneIndex)
{
    Loading = true;
    AsyncOperation asyncLoad =
SceneManager.LoadSceneAsync(SceneIndex);
    asyncLoad.allowSceneActivation = false;

    while (!asyncLoad.isDone)
    {
        scrollbar.value = asyncLoad.progress;

        if (asyncLoad.progress >= .9f &&
!asyncLoad.allowSceneActivation)
        {
            yield return new WaitForSeconds(1);
            asyncLoad.allowSceneActivation = true;
        }

        yield return null;
    }
}
}

```

Інтерфейс гри

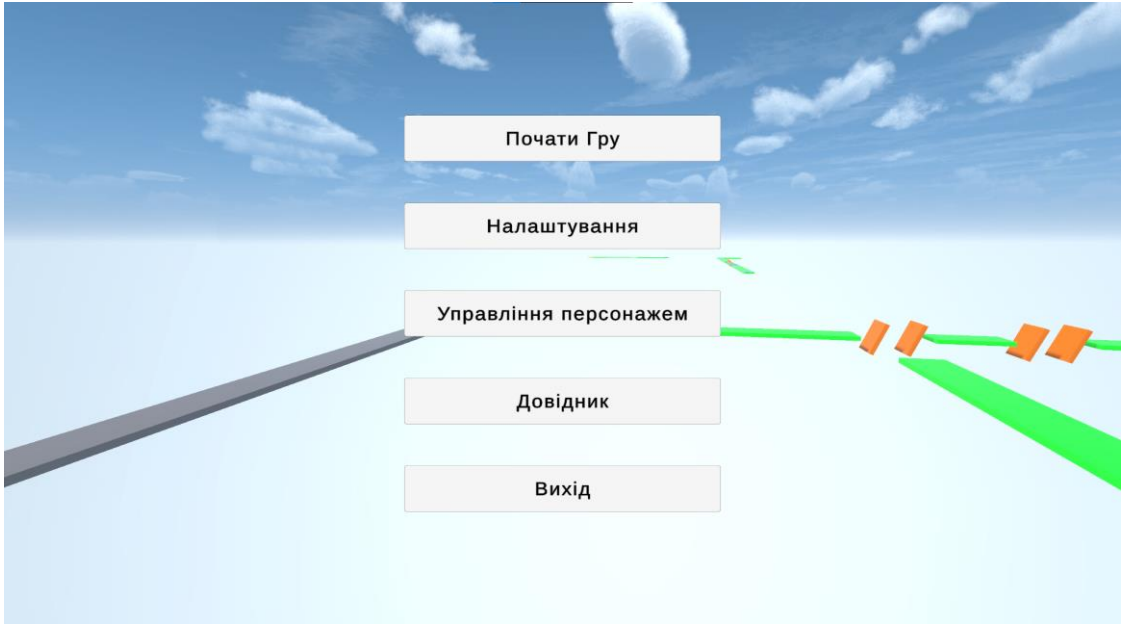


Рисунок 1 - Головне меню

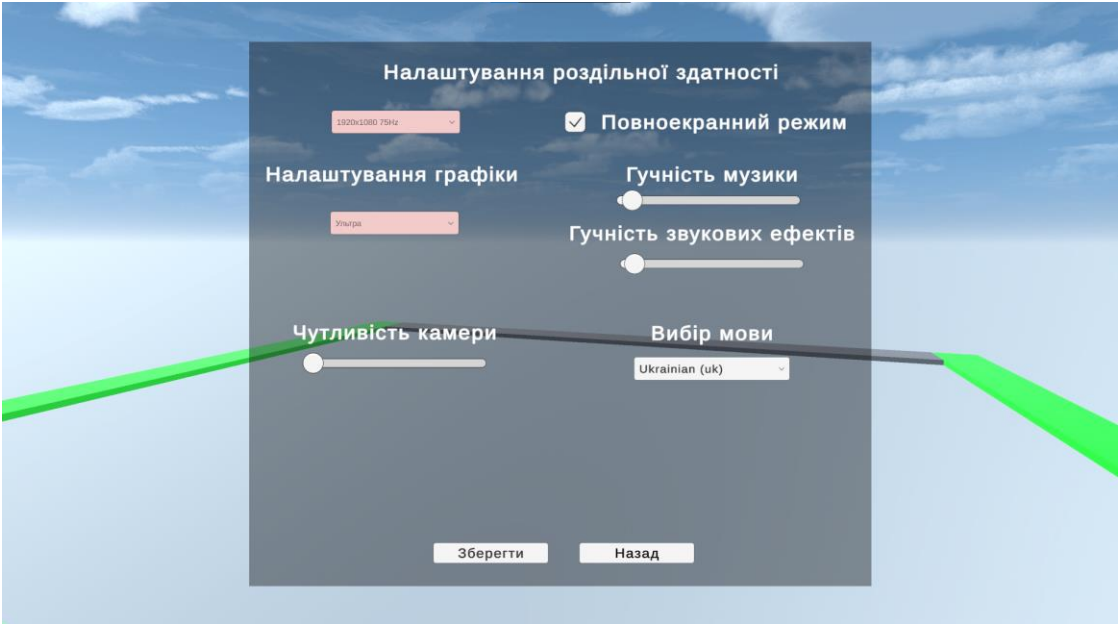


Рисунок 2 - Меню налаштувань



Рисунок 3 - Меню вибору рівня

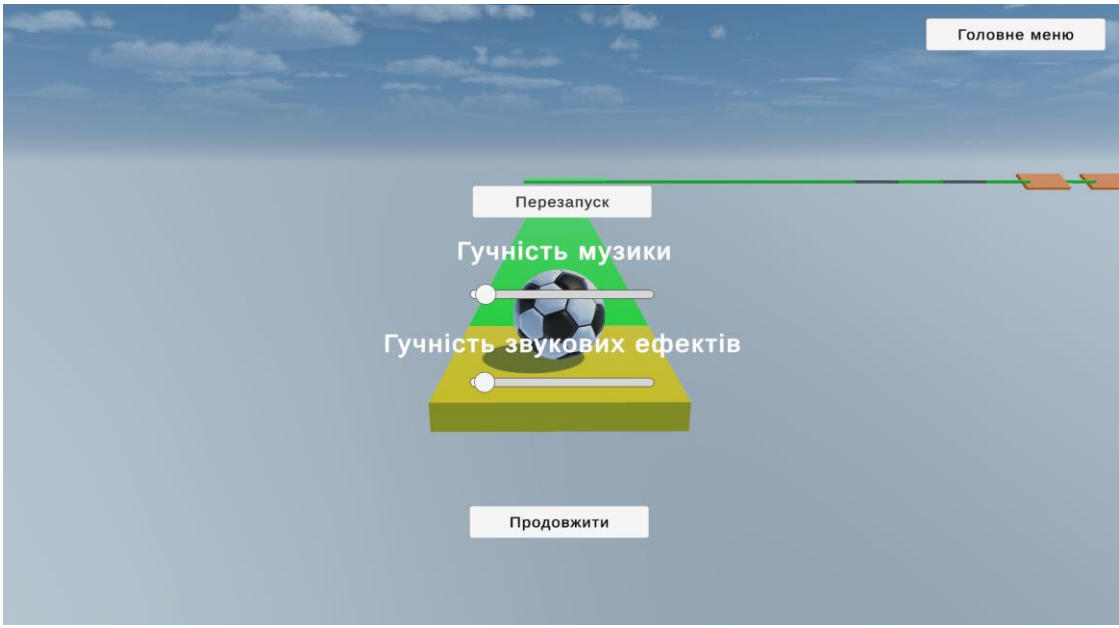


Рисунок 4 - Меню паузи

					13.02070849.00034 ПЛ1				
					Розробка комп'ютерної гри мовою С# за допомогою середовища Unity				
					Інтерфейс гри				
Зам.	Лист.	№ докум.	Підп.	Дата	НУ "Запорізька політехніка" КНТ 510 - СП				
Розроб.	Книрик С.О.								
Перевір.	Касьян К.М.								
Т.контр.									
					Лист 1Листів 1				
Н.контр.	Щербак Н.В.				Інтерфейс гри				
Затвердж.	Кудерметов Р.К.								

Інтерфейс інсталятора

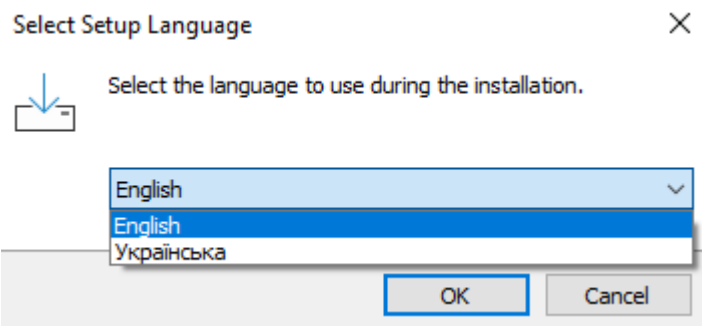


Рисунок 1 - Вибір мови інсталятора

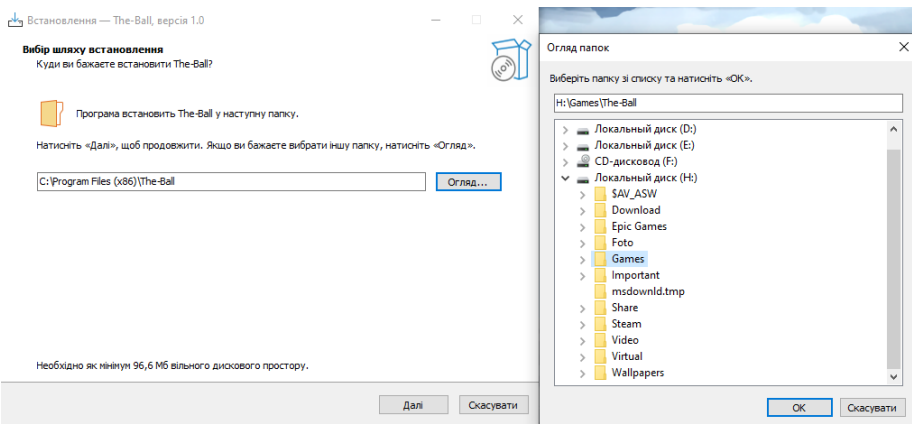


Рисунок 2 - Вибір шляху встановлення

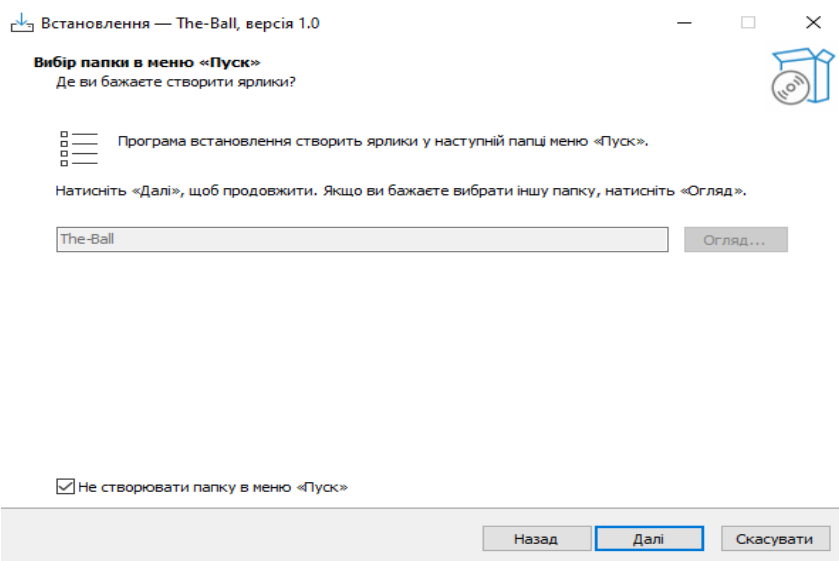


Рисунок 3 - Вибір папки в меню «Пуск»

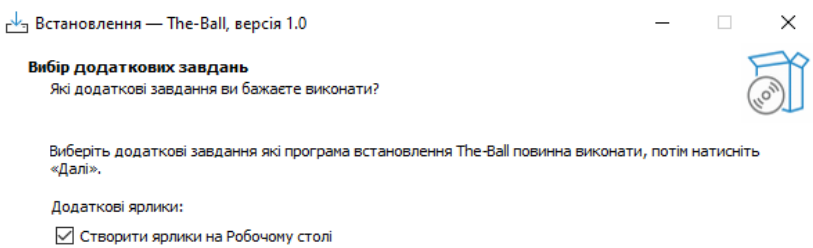


Рисунок 4 - Вибір додаткових завдань

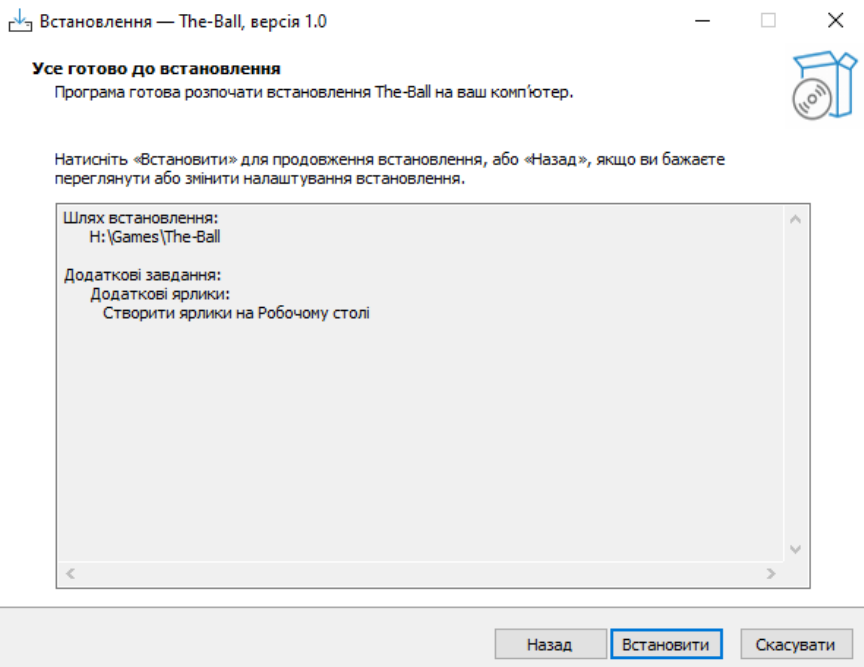


Рисунок 5 - Перегляд налаштувань

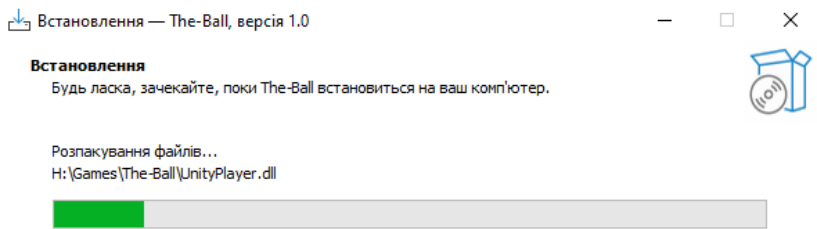


Рисунок 6 - Встановлення

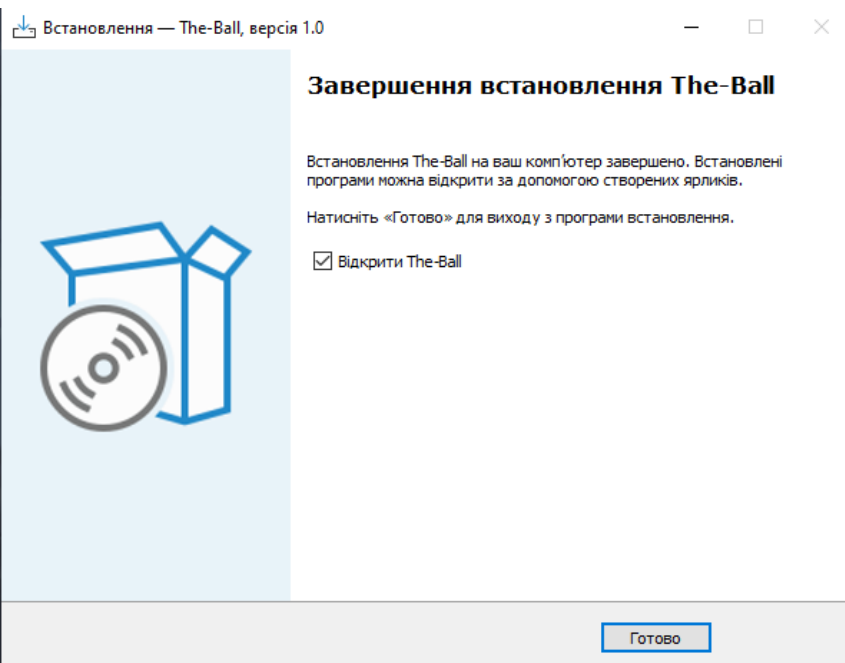


Рисунок 7 - Завершення встановлення

					13.02070849.00034 ПЛ2				
					Розробка комп'ютерної гри мовою C# за допомогою середовища Unity				
					Інтерфейс інсталятора				
					ІНУ "Запорізька політехніка" КНТ 510 - СП				

Ігровий процес

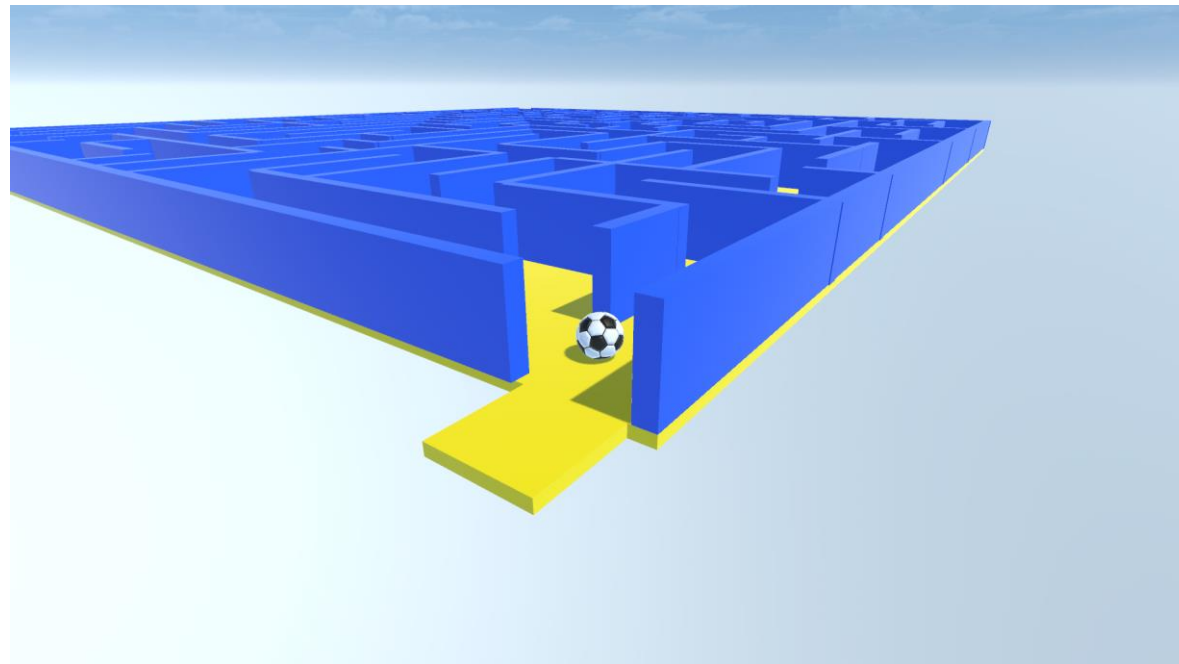


Рисунок 1 - Рівень 9

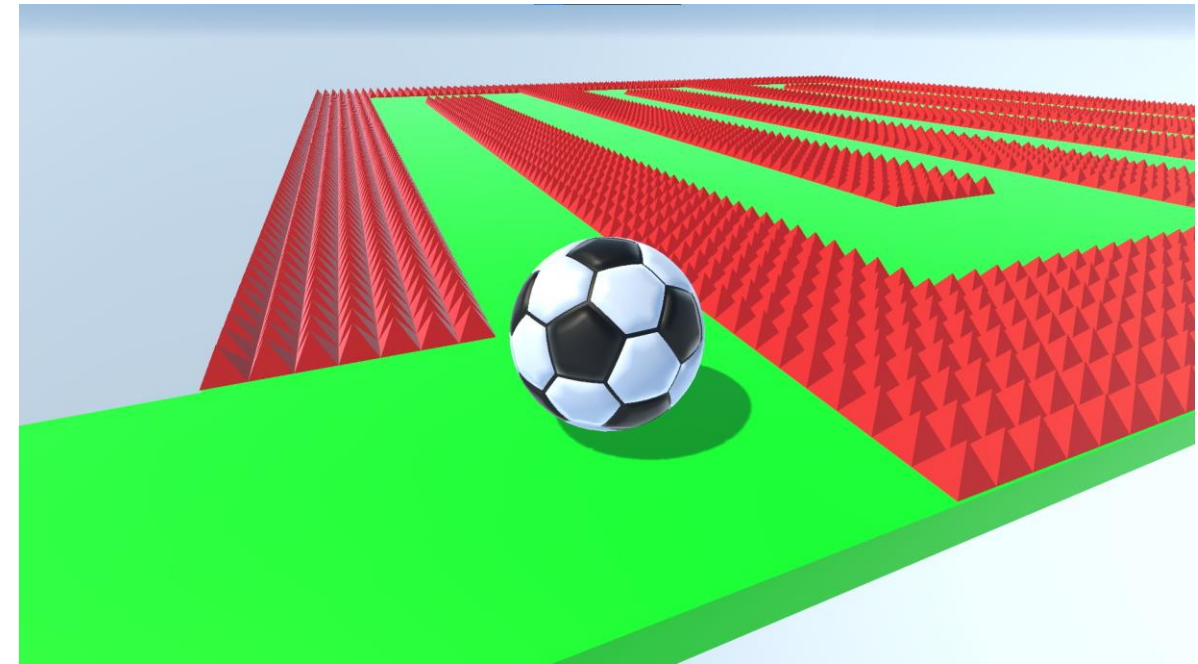


Рисунок 2 - Рівень 3

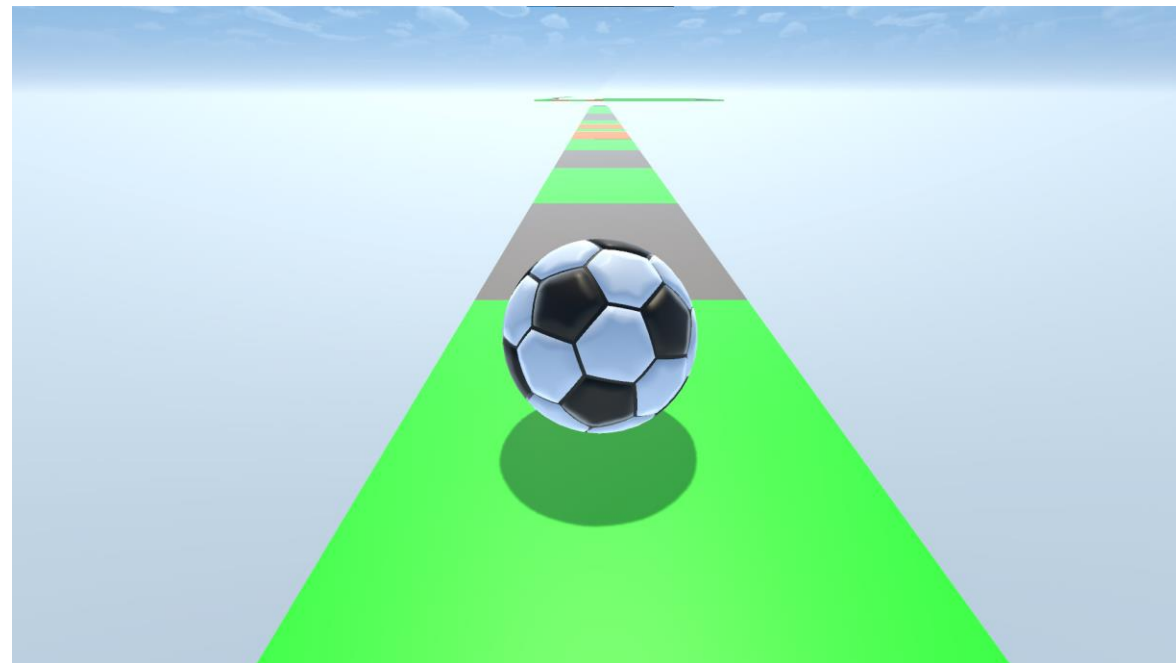


Рисунок 3 - Рівень 1

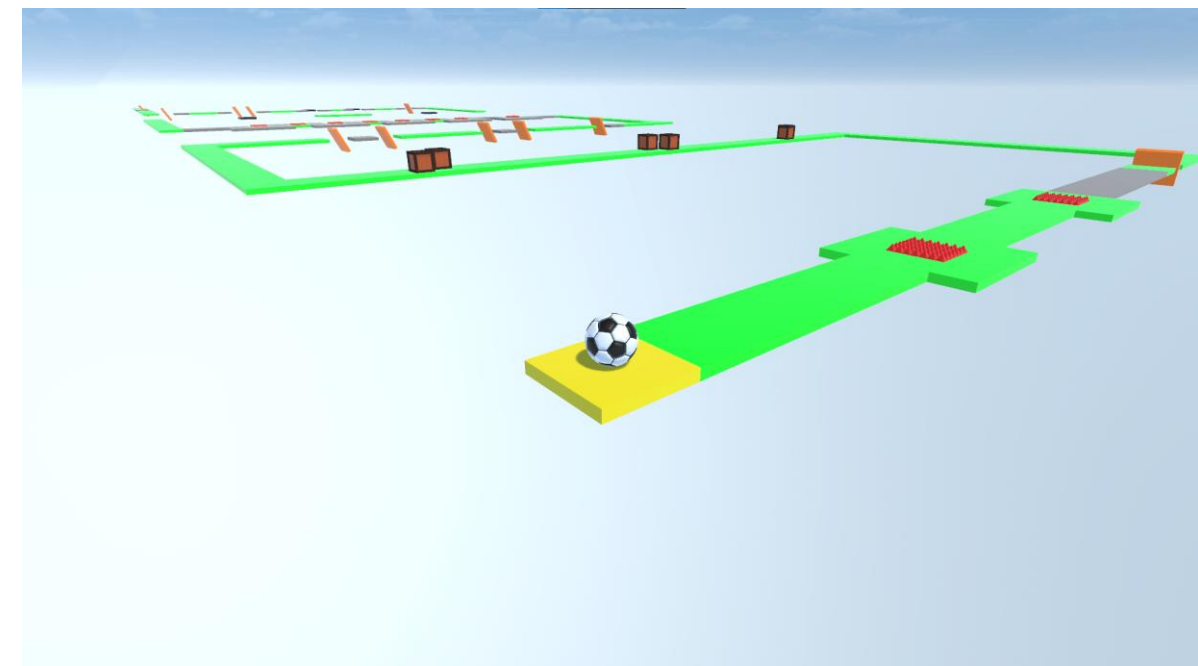


Рисунок 4 - Рівень 8

					13.02070849.00034 ПЛЗ				
					Розробка комп'ютерної гри мовою C# за допомогою середовища Unity	Лист.		Маса	Масшт.
Зам.	Лист.	№ докум.	Підп.	Дата					
Розроб.	Книрик Є.О.								
Перевір.	Касьян К.М.								
Т.контр.									
					Ігровий процес	Лист 1		Листів 1	
Н.контр.	Щербак Н.В.					НУ "Запорізька політехніка"			
Затвердж.	Кудерметов Р.К.					КНТ 510 - СП			

Діаграма використання комп'ютерної гри



					13.02070849.00034 ПЛ4				
					Розробка комп'ютерної гри мовою С# за допомогою середовища Unity				
Зам.	Лист.	№ докум.	Підп.	Дата					
Розроб.		Книрик С.О.			Діаграма використання комп'ютерної гри				
Перевір.		Касьян К.М.							
Т.контр.									
					НУ "Запорізька політехніка" КНТ 510 - СП				
Н.контр.		Щербак Н.В.							
Затвердж.		Кудерметов Р.К.							