math_2023_017

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
кафедра математичного моделювання

Створення ігрового 3D додатку у середовищі Unity

Кваліфікаційна робота

Рівень вищої освіти – другий (магістерський)

Виконав:
студент 6 курсу, 607 групи
Длубік Владислав Володимирович
Керівник:
доцент, канд.фіз.-мат. наук
Івасюк Г.П.

До захисту допущено
на засіданні кафедри
протокол № 9 від 5 грудня 2023 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2023

Анотація

Розроблено ігровий 3D додаток для розвитку аналітичних здібностей користувачів, а саме школярів та студентів, для знаходження оптимального шляху проходження рівнів, корекції рішень на основі аналізу минулих спроб.

Застосунок призначений для використання у процесі навчання та розвитку вмінь різної спрямованості.

Додаток написано за допомогою мови програмування C# та з використано платформу для розробки Unity 3D.

Ключові слова:

додаток, подія, програмне забезпечення, гра, користувач, розвиток, класи та об’єкти, sprite, scene, script.

Анотація

A 3D gaming application has been developed to develop the analytical skills of users, namely schoolchildren and students, to find the optimal way to pass levels, correct decisions based on the analysis of past attempts.

The application is intended to be used in the process of learning and developing skills of various kinds.

The application is written in C# programming language and uses the Unity 3D development platform.

Keywords:

application, event, software, game, user, development, classes and objects, sprite, scene, script.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ В.В.Длубік

Зміст

Вступ

Актуальність роботи. На сьогоднішній день передові технології переплітаються з нашим повсякденним життям, проникаючи практично у кожен аспект. Від телефонів та персональних комп'ютерів до різноманітних гаджетів, ці пристрої не лише полегшують наше життя, а й допомагають нам розвиватися. Однак зараз через неймовірну кількість контенту, виникає ситуація, де ігри не дають корисного навантаження, а є лише способом проведення дозвілля.

У сучасному світі, насиченому стресом та інформаційним перевантаженням, ігри головоломки виявляються актуальними, оскільки вони надають можливість відпочити і розслабити мозок, сприяючи відсотковій зосередженості та відпочинку від повсякденних турбот. Головоломки стають ефективним інструментом для розвитку когнітивних функцій, таких як логіка, спостережливість та творче мислення, що робить їх актуальними у контексті підтримки інтелектуального зростання.

З розвитком технологій і популярністю мобільних платформ, ігри головоломки стають доступними для широкого кола користувачів, дозволяючи займатися цікавим та корисним відпочинком в будь-якому місці та часі, що підсилює їхню актуальність в наш час.

Ці ігри можуть стати не лише відмінним засобом розваги, але і ефективним інструментом для стимулювання мислення, розвитку творчих навичок та підвищення рівня інтелектуальної активності.

Мета. Метою кваліфікаційної роботи, є розробка додатку який допоможе користувачам у розвитку розумових здібностей, зацікавить та допоможе покращити свої когнітивні вміння. Реалізувати застосунок шляхом використання мови програмування C# та з використано платформу для розробки Unity 3D.

Комп’ютерні ігри це – одна з тих причин, чому підлітки обожнюють проводити час за екраном. Відеоігри ні в якому разі не можна назвати безглуздим заняттям, оскільки вони навчають своїх гравців адаптовуватися до умов гри та її правил, освоїти навички персонажа, досягти нових рекордів, своєчасно реагувати на несподіванки, намагатися випередити свого суперника. Водночас в іграх гравець ототожнює себе з персонажем гри, тому його перемоги, такі привабливі, а поразки такі болючі. І звісно ж діти, чий розум якраз налаштований на швидке сприйняття нової інформації, найкраще справляються з цілями ігор, які створені для всебічного розвитку кожного, хто має бажання в них зіграти.

Однак завжди потрібно пам'ятати, що крім розваг у кожного є свої обов’язки, і найважливіший з них це безсумнівно навчання. На мою думку, для переважної більшості дітей причиною їх невдач та промахів у навчанні є не нестача розумових здібностей, а те, що вони знаходять навчання як нудне і нікому не потрібне заняття, яке лише забирає їх час. Проте це не так, відеоігри та навчальна програма мають багато спільного. Відрізняється лише спосіб подачі завдання. Під час навчання, нам дають конкретну задачу та точний, алгоритм її вирішення, як наприклад в математиці. Для більшості це малоприємна, монотонна робота, яка не викликає великого захвату, особливо у малечі. Відеоігри в свою чергу дають простір для фантазії, хоча якщо задуматись то кожна гра це просто ще одна поставлена задача, яку потрібно виконати, тільки з невимушеним поданням. Якщо почати розглядати дану проблему як програміст то, навчання в школах більше схоже на програмування імперативним методом, тобто покроково, прямолінійно, точно, а ігри в свою чергу більше схожі на програмування декларативним методом, тобто ми вказуємо лише бажаний результат або ж ціль, а яким шляхом буде досягнута ця ціль вибір кожного гравця. Під час гри дитина намагається досягнути найкращого результату, дотримуючись правил поставлених грою та

враховуючи всі обмеження. На мій погляд, це одна з найголовніших навичок в житті людини. Всі ми повинні вирішувати поставлені перед нами задачі.

Кваліфікаційна робота присвячена створенню 3D гри, яка б зацікавила дітей, та допомогла їм ставити перед собою цілі та досягати їх із задоволенням. Оскільки діти все більше часу проводять граючи відеоігри, треба намагатися зробити так, щоб таке заняття приносило якомога більше користі. А що може бути краще, ніж проводити свій вільний час із задоволенням та користю?

Розділ 1. Опис середовища Unity

1.1 Ігровий рушій Unity 3D

Unity – це багатоплатформовий інструмент для розробки 2D і 3D ігор, а також додатків, що працює на операційних системах OS X та Windows. Він розроблений компанією Unity Technologies у 2005 році і з того часу не припиняє розвиватись. Програма дає змогу використовувати велику кількість функціональних можливостей. Безперечно, він став лідером індустрії, і як тільки з'являється нова ігрова / графічна технологія, розробники негайно реалізують її в Unity. Крім розробки сінглплеєрних ігор для PC, шляхом підключення експортерів можна перенести ігри під інші ОС, консолі і мобільні технології (за експорт доведеться доплатити 1500 доларів за кожну платформу: iOS, Android, BlackBerry). Плюс до цього утворилася ціла індустрія, що працює над створенням доповнень і розширень ігрового двигуна, серед них є як спеціалізовані серверні рішення для Unity (eg Photon - повноцінний ігровий сервер), так і засоби для розробки користувальницького інтерфейсу (NGUI), конструктори, призначені для створення ігор визначених жанрів (e.g. Playmaker).

У самого редактора Unity є порти під OS X і Windows, при цьому спочатку він був призначений для OS X. В Unity включена підтримка DirectX 11, що відкриває твоїм додаткам дорогу в світи Windows 8 і Windows Phone 8. Зовсім недавно вийшла чергова версія Unity під номером 4.2, в якій з'явилася підтримка останньої на даний момент OpenGL ES 3.0, поки цими засобами володіють тільки топові Android-смартфони. Движок Unity особливо цінний за низький поріг входження для початківців користувачів, завдяки цьому, а також тому, що інді-версія безкоштовна, навколо движка організувалося величезне співтовариство. Низький поріг входження є результатом грамотного дизайну програми: багато речей можна виконати за допомогою різних редакторів, не написавши при цьому ні строчки коду (якщо що, код пишеться

на JavaScript, C #, Boo). Вихідний код на C / C ++ закритий, але це у зв'язку з розширеною компонентної структурою движка не створює ніяких перешкод [1].

У Unity є дві основних переваги над іншими передовими інструментами для розробки ігор: надзвичайно продуктивний візуальний робочий процес і потужна підтримка між платформами

Ігровий рушій Unity дуже популярний серед розробників відеоігор, особливо серед початківців. Незважаючи на те, що двигун орієнтований на тривимірні додатки, він підтримує роботу з двомірної графікою.

Unity виявляється надзвичайно корисною у розробці проектів, які потребують постійного удосконалення чи модифікації з плином часу. Однією з важливих переваг є можливість коригувати об'єкти прямо в редакторі та переміщати їх по сцені, навіть під час активного використання гри. Unity дозволяє налаштовувати сам редактор за допомогою сценаріїв, що робить процес розробки більш гнучким та ефективним.

Це лише частина переваг Unity. Інтуїтивне налаштування робочого простору, простий і функціональний інтерфейс, безліч варіантів розробки додатків і гнучкість системи роблять Unity лідером серед багатоплатформових двигунів. Додатково, Unity підтримує фізику твердих тіл і тканин, а також фізику типу Ragdoll (ганчіркова лялька).

У редакторі також існує система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, масштабу та батьківського об'єкта. Скрипти у самому редакторі кріпляться до об'єктів у вигляді окремих компонентів. У Unity вбудована підтримка мережі. Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі. Також, важливою є можливість підлаштовувати редактор під конкретного користувача, тому кожен зможе налаштувати свою робочу зону саме так, як він того бажає [2].

Що виходить з вищеописаного, двигун ідеально підходить для починаючих розробників. Unity дозволяє швидко створювати об'єкти, розповсюджувати та зв’язувати їх, створювати нехитруючий центр, створювати власний вміст та вміщувати магазинні асоціації. Так як двигун має величезну аудиторію користувачів, знайти рішення будь-якої проблеми не складає труднощів - ком'юнити з радістю допомагають починаючому розробнику. Є велика кількість офіційних та користувацьких блогів та навчальних курсів які надають усі необхідні знання.

Великі студії також знаходять у цьому ігровому рушії свої переваги. Його можливості переміщення дозволяють створити масштабні ігри близькі до AAA-класу. За прикладами далеко ходити не потрібно: Pillars of Eternity, Firewatch, Inside, Superhot - усі ці проекти, створені на Unity. Будь то дорогою сюжетно-орієнтований проект або розрахована на багато користувачів браузерна гра на Unity3D, розробники в будь-якому випадку отримають потужний і гнучкий інструментарій для створення максимально якісного продукту.

1.2 Особливості взаємодії з Unity

Unity має багато переваг, які роблять з нього чудовий засіб розробки додатків та ігор, але також необхідно згадати про його недоліки.

Не всі компоненти редактора однаково добре працюють в 3D і 2D. Зокрема, вбудована система освітлення не дозволяє двомірним об'єктів служити повноцінним перешкодою для світла і відкидати тіні на інші об'єкти. До всіх об'єктів прикріплений однаковий матеріал з однаковим шейдером. Якщо куб є повноцінним перешкодою для світла, то чотирикутник відкидає тінь тільки від тих джерел, що не лежать в одній площині з ним, а спрайт і зовсім ігнорується системою розрахунку тіней. У тривимірному просторі це обумовлено фізичним розрахунком освітлюваних областей, так як чотирикутник, на відміну від куба, не має товщини, проте в двомірному просторі об'ємні фігури представлені плоскими 2D об'єктами, тому необхідно забезпечити можливість взаємодії зі світлом двомірних об'єктів, ідентичну взаємодії тривимірних в 3D просторі [1].

Наступним несподіваним недоліком є використання (prefabs) так званих шаблонних екземплярів. Хоча вони ніби додають гнучкості до візуального створення інтерактивних об’єктів, але редагування таких шаблонів іноді на диво важко реалізувати. Зміни у шаблонах іноді даються не з першого разу, це в основному відбувається тільки на початку відладки префабу, але цілком може виникнути така проблема і посеред творчого процесу. Іноді, змінюючи шаблон, потрібно брати до уваги те як ці зміни вплинуть на вже готову сцену і чи будуть вони сумісні з іншими шаблонами (prefabs).

Ще одним недоліком є те що в Unity в разі виникнення технічних проблем може не зберегтися сценарій останньої сцени, тому потрібно уважно слідкувати за тим щоб все було збережено і про всяк випадок мати резервні копії вашого проекту чи гри. Наступним неприємним недоліком виявилось те, що можливості підключення зовнішніх бібліотек просто немає, Unity не

підтримує посилання на зовнішні бібліотеки коду. Через це замість того щоб просто передати посилання на одну загальнодоступну папку, всі маніпуляції із збереженням проекту та підгрузкою потрібних бібліотек доведеться виконувати вручну. Відсутність єдиної папки з бібліотеками ускладнює колективне використання функціоналу різними проектами. Цієї незручності можна уникнути, раціонально застосовуючи системи контролю версій, але готове рішення даної проблеми в Unity відсутнє. Хоча це могло б дати набагато більше функціональних можливостей для розробки софту на базі даного інструменту.

Unity дає змогу створювати ігри та додатки, переважно хорошої, товарної якості. Проте при роботі з Unity можуть виникнути певні труднощі. В основному, це пов'язано з недостатнім досвідом як при роботі з даним ігровим рушієм, так і з самим програмуванням в цілому. Переважно Unity описують як простий набір готових програмних компонентів, для використання яких програмування або взагалі не потрібно, або потрібно лише частково. Це вкрай неправильна позиція, оскільки без програмування не вдасться створити продвинутый продукт, а тому він навряд чи зможе зацікавити як користувачів так і інвесторів [3].

1.3 Початок роботи з Unity

Починати потрібно з того, що для роботи з Unity вам необхідно встановити цей ігровий рушій на ваш комп'ютер. Краще всього це зробити з офіційного сайту Unity (https://unity3d.com/ru/get-unity/download). Після цього запускаєте Unity, краще це робити за допомогою Unity Hub, цей інструмент дозволяє вибрати версію двигуна для роботи.

Інтерфейс в Unity ділиться на декілька частин: вкладка Scene, вкладка Game, панель інструментів, вкладка Hierarchy, панель Inspector, вкладка Project та Console. У кожної частини є власне призначення, при цьому вони відіграють важливу роль в циклі створення гри:

- Перегляд файлів виконується на вкладці Project.
- Покладені в тривимірну сцену об’єкти переглядаються на вкладці Scene.
- Панель інструментів надає елементи керування сценою.
- Змінювати взаємозв’язки між об’єктами можна методом пересування їх на вкладку Hierarchy.
- Панель Inspector відображає інформацію про виділені об’єкти, в тому числі і про зв’язаний з ними код.
- Тестувати отримані результати можна на вкладці Game, одночасно переглядаючи повідомлення про помилки на вкладці Console.

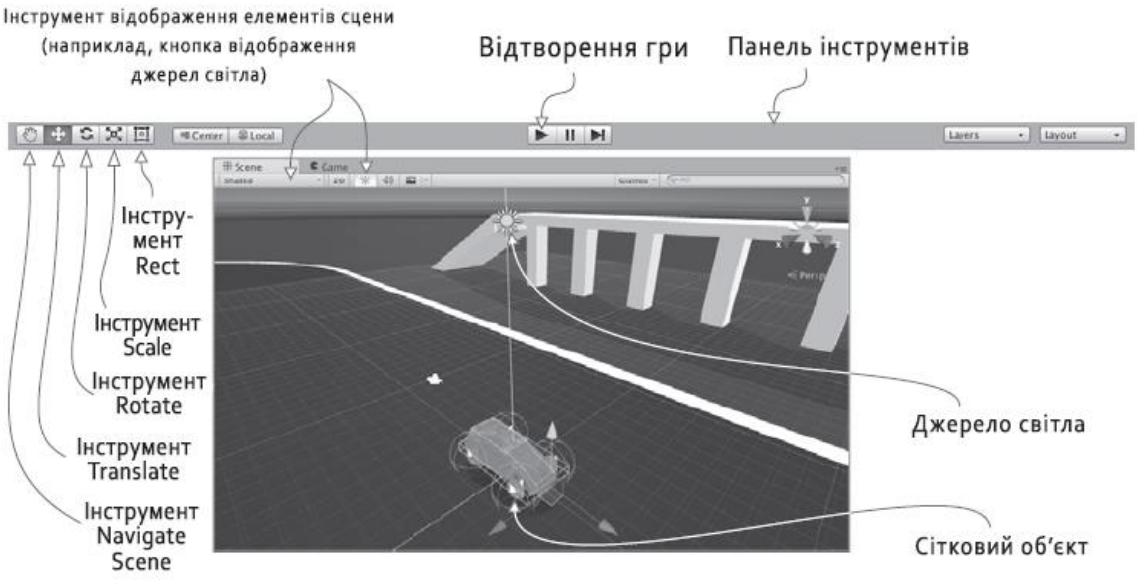
Ця компоновка пропонується за замовчуванням; всі доступні представлення вміщені на вкладки, які можна переміщувати, можна змінювати їх розмір і фіксувати в різних частинах екрану. Звісно ви можете поекспериментувати з вибором компоновки, але поки ви не розібрались в призначенні кожного елемента інтерфейсу, варіант компоновки запропонований за замовчуванням, є оптимальним.

Найбільш помітною частиною інтерфейсу є розміщена по центру вкладка Scene. Саме тут можна бачити, як виглядає ігровий світ, і переміщувати об’єкти в сцені. Сіткові об’єкти в сцені виглядають так, як і потрібно, у вигляді сіток. Також можна побачити і низку інших об’єктів, що

відображаються різноманітними значками і кольоровими лініями. Це камери, джерела світла, джерела звуку, області зіткнень і т.п. Зрозуміло, зображення, що спостерігається, відрізняється від того, що буде видно в процесі гри, - можна розглядати сцену, не обмежуючись ігровою інтерпретацією.

Ігрова інтерпретація відображається не на окремій частині екрану, а на вкладці Game, розташованій поряд з вкладкою Scene (перехід з вкладки на вкладку здійснюється з допомогою кнопок у верхньому лівому кутку області відображення). В інтерфейсі реалізовані також і інші компоненти, сконструйовані подібним чином; для зміни вмісту, що вони відображають, достатньо перейти на іншу вкладку. Після запуску гри починає відображатися ігрова інтерпретація, тобто немає необхідності переходити на вкладку Game – перемикання здійснюється автоматично. В режимі відтворення гри можна повернутися на вкладку Scene для перегляду об’єктів. Це вкрай корисна можливість, що дозволяє зрозуміти, як виглядає гра, що відтворюється, зсередини і що в ній відбувається. Подібний інструмент налагодження відсутній в більшості ігрових двигунів.

Для запуску гри достатньо клацнути на кнопку Play, що розташована над вкладкою Scene. Вся верхня частина інтерфейсу зайнята так званою панеллю інструментів, і кнопка Play знаходиться якраз в центрі цієї панелі. Малюнок 1.1 отриманий відсіканням решти частин інтерфейсу редактора і ілюструє тільки панель інструментів з розташованими під нею вкладками Scene та Game.

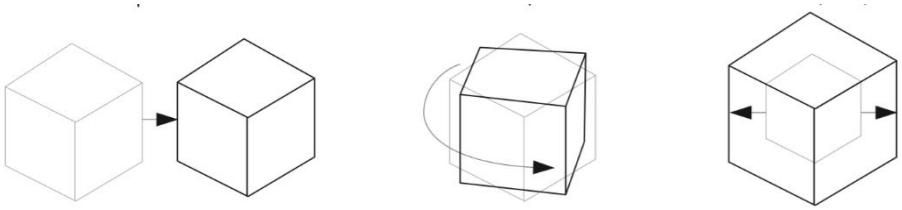


Мал. 1.1 Обрізаний варіант редактора, що демонструє тільки панель інструментів та вкладки Scene та Game

На лівій стороні панелі інструментів розміщені кнопки навігації і перетворення об’єктів. Вони дозволяють розглядати сцену з різних сторін і переміщувати об’єкти. Перегляд сцен і переміщення об’єктів – дві основні дії, що виконуються у візуальному редакторі Unity. Праву сторону панелі інструментів займають випадаючі меню з переліком компоновок і шарів. Оскільки інтерфейс Unity є гнучкою системою, з можливістю налаштування, існує меню Layouts, що дозволяє переходити від одного варіанту компоновки до іншого. Меню Layers відноситься до розширених функціональних можливостей.

Навігація по сцені здійснюється в основному за допомогою миші і набору клавіш-модифікаторів, що впливають на результат маніпуляцій мишею. Трьома головними операціями є переміщення (move), поворот (orbit), і масштабування (zoom). В основному вони зводяться до клацань і перетягування з допомогою натиснутих клавіш Alt (або Option на комп’ютерах Mac) і Ctrl. Для повного розуміння механіки слід трохи попрактикуватись. Перетворення об'єктів також відбувається за допомогою цих трьох операцій. Більш того, кожному типу навігації відповідає власне перетворення:

перенесення(translate), поворот(rotate) і зміна розмірів(scale). Малюнок нижче демонструє ці перетворення на прикладі куба.



Мал. 1.2 Застосування трьох варіантів перетворення: перенесення, повороту і зміни розміру.

(більш світлі лінії означають початкове положення об'єктів)

Після виділення об'єкта сцени з'являється можливість рухати його (або, якщо брати більш точний термін, переносити), обертати і вказувати його розмір [1]. Якщо поглянути з цієї точки зору на процес навігації по сцені, то переміщення означає процес переносу камери, поворот відповідає повороту камери, а масштабування – зміни розмірів камери. Перехід між цими операціями здійснюється не тільки кнопками панелі інструментів, але й натисканням клавіш W, E та R. При переході в режим перетворення у виділеного об'єкта з'являються кольорові стрілки або кола. Це габаритний контейнер перетворення (transform gizmo), перетягуванням якого змінюється вигляд об'єкта. Поряд із кнопками перетворення знаходиться ще одна кнопка. Це кнопка інструмента Rect, що дозволяє перейти до роботи з двовимірною графікою поєднуючи в собі операції переносу, повороту і зміни розмірів. В тривимірному просторі за кожну з цих операцій відповідає свій інструмент, але в двовимірному просторі вони об'єднані, оскільки стає менше на один вимір. В Unity існує також набір клавіатурних комбінацій для пришвидшення виконання різноманітних операцій.

По ліву сторону екрану знаходиться вкладка Hierarchy, тоді як праву частину заповнила панель Inspector. Вкладка Hierarchy містить список всіх присутніх в сцені об'єктів у вигляді деревоподібної структури, гілки якої вкладені одна в одну відповідно до ієрархічних зв'язків між об'єктами. В

загальному, ця вкладка пропонує можливість виділення об’єктів за іменем, позбавляючи від необхідності шукати потрібний об’єкт в сцені, щоби виділити його кліком. Ієрархічні зв’язки об’єднують об’єкти один з одним. Візуально це оформлено у вигляді папок, при цьому є можливість переміщувати одночасно цілі групи об’єктів.

На панелі Inspector відображаються дані виділеного в поточний момент об’єкта. Більша частина інформації, що відображається, являється списком компонентів, причому можна додавати нові компоненти об’єкти або видаляти існуючі. Всі об’єкти гри містять принаймні один компонент – Transform, - тому на панелі Inspector завжди буде видно хоча б відомості про положення і орієнтацію виділеного об’єкта. У багатьох об’єктів є цілі списки компонентів, включаючи зв’язані з цими об’єктами сценарії.

В нижній частині екрану знаходяться вкладки Project і Console. В даному випадку ми бачимо таку ж організацію елементів інтерфейсу, як і у вкладках Scene і View. На вкладці Project відображається всі ресурси (графічні елементи і т. п.) проекту. В лівій частині цієї вкладки видно список папок проекту; при виділенні папки справа з’являються файли, що знаходяться в ній. Взагалі, це такий же список, як і у Hierarchy вкладці, але якщо ця вкладка показує перелік об’єктів сцени, то на вкладці Project знаходяться файли, не включені в жодну конкретну сцену (в тому числі і самі файли сцен – збережена сцена з’являється у вкладці Project). Ця вкладка дублює вміст папки Assets на диску, але в загальному випадку не рекомендується здійснювати видалення чи переміщення файлів безпосередньо з Assets. Рекомендується здійснювати ці операції на вкладці Project, а про синхронізацію з папкою Assets потурбується Unity.

Вкладка Console являє собою місце, де виводяться сповіщення, пов’язані з кодом. Іноді це спеціально вставлені розробником в програму повідомлення налагоджувача, іноді це повідомлення Unity, що з’являються при виявленні помилок в написаному розробником сценарії.

1.4. Запуск коду в Unity: компоненти сценарію

Виконання коду в Unity завжди починається з файлів, пов'язаних з об'єктом сцени. В кінцевому результаті це все частина згаданої нами раніше системи компонентів: об'єкти гри будуються як набори компонентів, і кожен такий набір може містити в собі сценарій. В термінології Unity файли з кодом прийнято називати сценаріями. Даний термін використовується в тому ж значенні, що і у випадку, коли в браузері запускається JavaScript: код виконується всередині рушія гри Unity, а не з'являється після компіляції у вигляді самостійного фрагменту. Сценарії в Unity більше нагадують індивідуальні класи ООП.

В Unity компонентами об'єктів виступають тільки ті сценарії, що успадковуються від класу MonoBehaviour – базового класу компонентів-сценаріїв. Цей клас визначає спосіб приєднання компонентів до об'єктів гри. Успадкування від цього класу дає низку методів, що автоматично додаються, які можна перевизначати. Це метод Start(), що викликається при активації об'єкта (яка наступає після загрузки рівня, в якому знаходиться об'єкт), та метод Update(). Відповідно, написаний код запускається, якщо він знаходиться в цих попередньо встановлених методах. Метод Update() викликається в кожному кадрі. Кадром (frame) називається один прохід зацикленого коду гри. Майже всі відеоігри (не тільки в Unity, а й взагалі) будуються навколо основного циклу гри. Тобто код циклічно виконується весь час, поки запущена гра. Кожен цикл включає в себе промальовування екрану, звідки, власне, і виник термін «кадр» (аналогічно до набору статичних кадрів у фільмі).

Щоб створити сценарій, необхідно вибрати команду C# Script в меню Create, доступ до якого можна отримати відкривши меню Assets (в меню GameObjects та в меню Assets є варіанти команди Create, але це різні команди) або клацнувши правою кнопкою миші в будь-якій точці вкладки Project. Після

введення назви нового сценарію, файл можна пересунути на довільний об’єкт сцени.

1.5 Графічні ресурси

Графічним ресурсом називається окрема одиниця візуальної інформації (зазвичай - файл), яку використовує гра. Це всеосяжна збірна назва для всього візуального вмісту; до графічних ресурсів відносять файли зображень, тривимірні моделі і т. п. Насправді. Це всього окремий випадок ресурсу, який, як відомо, являє собою будь-який файл, що використовується грою (наприклад, сценарій). Всі вони Unity знаходяться в одній папці Assets. В табл. 1 перераховані та описані п’ять основних видів графічних ресурсів, які застосовуються при створенні ігор.

Таблиця 1. *Основні типи графічних ресурсів*

Тип	Визначення типу
Двовимірне зображення	Плоскі зображення. В реальному світі їм відповідають картини та фотографії.
Тривимірна модель	Віртуальні тривимірні об’єкти (майже синонім до «сіткових об’єктів»). В реальному світі їм відповідають скульптури.
Матеріал	Пакет інформації, що визначає властивості поверхні будь-якого об’єкта, до якого приєднаний матеріал. До таких властивостей відносять світло, блиск і навіть невеликі нерівності.
Анімація	Пакет інформації, що визначає рух зв’язаного з ним об’єкта. Існують як і завчасно створені деталізовані послідовності, так і код, що вираховує положення об’єктів «на льоту».
Система частинок	Механізм створення великої кількості невеликих рухомих об’єктів і керування ними. Дозволяє створювати різноманітні візуальні ефекти, такі як вогонь, дим і бризки води

Створення графіки для нової гри в загальному випадку починається з двовимірних зображень або тривимірних моделей, оскільки ці ресурси формують базу для всього іншого. Зрозуміло, що двовимірні зображення слугують основою для двовимірної графіки, в той час як тривимірні моделі – для тривимірної. Точніше, двовимірні зображення являються плоскими

картинками. Модель – це тривимірний віртуальний об’єкт. Згаданий вище термін «сітковий об’єкт» є майже синонімом до тривимірної моделі, але відноситься виключно до геометрії тривимірних об’єктів (з’єднані між собою лінії та фігури), в той час як термін «модель» набуває більш загального значення і часто означає і інші атрибути об’єкта [1].

Наступні два типи ресурсів у списку – це матеріали та анімація. На відміну від двовимірних зображень і тривимірних моделей вони не існують самі по собі. Зображення і моделі легко уявити завдяки їх аналогам з реального світу. Першим відповідають картини, другим – скульптури. Матеріали та анімації не мають прямих асоціацій з реальними об’єктами. Якщо провести аналогію з мистецтвом, матеріал можна уявити як речовину (глина, бронза, мармур і т. п.), з якого створюється скульптура. Схожим чином анімація являє собою прив’язаний до видимого об’єкта абстрактний шар інформації. Оскільки рухи, як описує анімація, можна задати незалежно від об’єкта, то їх можна використовувати, поєднуючи і комбінуючи з іншими об’єктами. В якості конкретного прикладу можна розглянути персонажа, що рухається по сцені. Його положення в кожен момент визначається кодом гри (сценаріями руху). Але детальні рухи ніг, що ступають по землі, рухи рук, що розмахуються і повороти стегон являються відтворювані в циклі анімаційні послідовності, які і є графічним ресурсом. Останній графічний ресурс, що згадується в табл. 1, це система частинок.

Системою частинок (particle system) називається механізм створення великої кількості рухомих об’єктів і керування ними. Зазвичай ці об’єкти мають маленький розмір – саме тому вони називаються частинками, хоча це не є обов’язковою умовою. Системні частинки слугують для створення візуальних ефектів, як вогонь, дим, або водяні бризки. Роль частинок (тобто окремих об’єктів, що контролюються системою) може відігравати будь-який сітковий об’єкт, але для більшості ефектів достатньо квадрата, який відтворює зображення (наприклад, іскри полум’я або згустку диму).

В основному створення графіки для гри виконується у зовнішніх програмах, які ніяк не зв’язані з Unity. В Unity можна генерувати тільки матеріали і системи частинок. Для створення тривимірних моделей і анімацій застосовуються найрізноманітніші графічні редактори. Отримані у зовнішній програмі тривимірні моделі потім зберігаються як графічні ресурси, тобто імпортуються в Unity. Головною задачею розробника, що працює в Unity є побудова сцени, в якій фігурують двовимірні зображення, тривимірні моделі, матеріали і системи частинок. Іноді існуватиме потреба користуватися вже готовими графічними ресурсами, імпортуючи їх в Unity, але можуть складатися ситуації (особливо з системою частинок), коли графічний ресурс необхідно створювати з нуля.

Створення ігрової графіки – величезна предметна область. Беручи до уваги вище сказане, людина, що займається програмуванням ігор, повинна розуміти, як Unity працює з графічними ресурсами, і, можливо, навіть вміти створювати їх грубі замінники; їх ще називають програмістською графікою (programmer art) і пізніше (вже в готовій грі) замінюють потрібними графічними ресурсами. Майже кожен програміст, хоча б раз, пробував себе у створені графіки для додатків або ігор. Насправді – це не така легка справа як здається. Створення ігрової графіки потребує значних графічних ресурсів, з якими потрібно вміти працювати на пристойному рівні, для досягнення бажаного ефекту. Для початківців найкращим варіантом буде залучення спеціаліста або використання безкоштовної графіки, тобто графіки яка є у відкритому доступі і доступна для використання. Звісно завжди є можливість самому створити потрібну вам графіку. Кількість графічних ілюстраторів вражає, але за використання деяких доведеться заплатити, що може бути перешкодою для початківців.

Розділ 2. Створення гри головоломки

2.1. Постановка задачі

Створити 3D гру головоломку для школярів та студентів. Метою гри є розробка алгоритмів оптимального шляху проходження рівня, розвиток уважності, координації рухів, просторової уяви та цілеспрямованості.

Для виконання поставленої задачі необхідно:

- створити дві основні сцени – Menu та End;
- створити сцени гри – Game;
- створити меню та інші форми для навігації по грі;
- використати у грі аудіо систему;
- створити об’єкти, необхідні для ігрового процесу: шаблон (префаб) для створення екземплярів кубів, об’єкт, діями якого буде керувати гравець (Player) тобто жовтий куб;
- у сценаріях створених об’єктів прописати логіку гри та описати рух динамічних об’єктів; для гравця створити горизонтальний рух, логіку у разі взаємодії з різними кубами;
- для сцени Menu створити можливість запуску гри при натисканні на певну клавішу;
- розробити виклик паузи, під час якої ігровий час “заморожується”, також можна вийти з гри, або повернутись в головне меню.
- створити дві основні кнопки «R» для перезапуску поточного рівня та «Q» для переходу на інший рівень.

2.2. Алгоритм реалізації

Гра складається з двох основних сцен – Menu та End, які відображають головне меню та кінець гри відповідно. Основний процес гри відбувається в окремо створених сценах від 1 рівня до крайнього рівня.

Інтерфейс гри написаний англійською мовою.

2.2.1 Створення сцени Menu

Першим, що побачить користувач при запуску гри, буде сцена Menu. Всі елементи даної сцени належать до сімейства UI (user interface). Спочатку було додано базовий елемент Canvas, а всі наступні елементи будуть його нащадками.

Було створено Text (назва гри) з використанням спеціального шрифту та задіяним Component Shadow для надання об’ємності тексту.

Для створення кожної із трьох кнопок головного меню – Розпочати, Налаштування та Вихід – використано елемент Button.

За допомогою Photoshop було створено відповідні картинки (image), для головних кнопок меню (див. мал. 2.1).

Щоб додати кнопці інтерактивності, їй надано здатність змінювати розмір при різних, виконуваних з нею, маніпуляціях: наведенні курсора виконується (Animation) анімація зміни розміру, при натисканні на кнопку лунає звук кліка, а також у функцію кнопки OnClick() додано ряд певних дій, що будуть виконуватися, залежно від вибраної кнопки, при її натисканні.

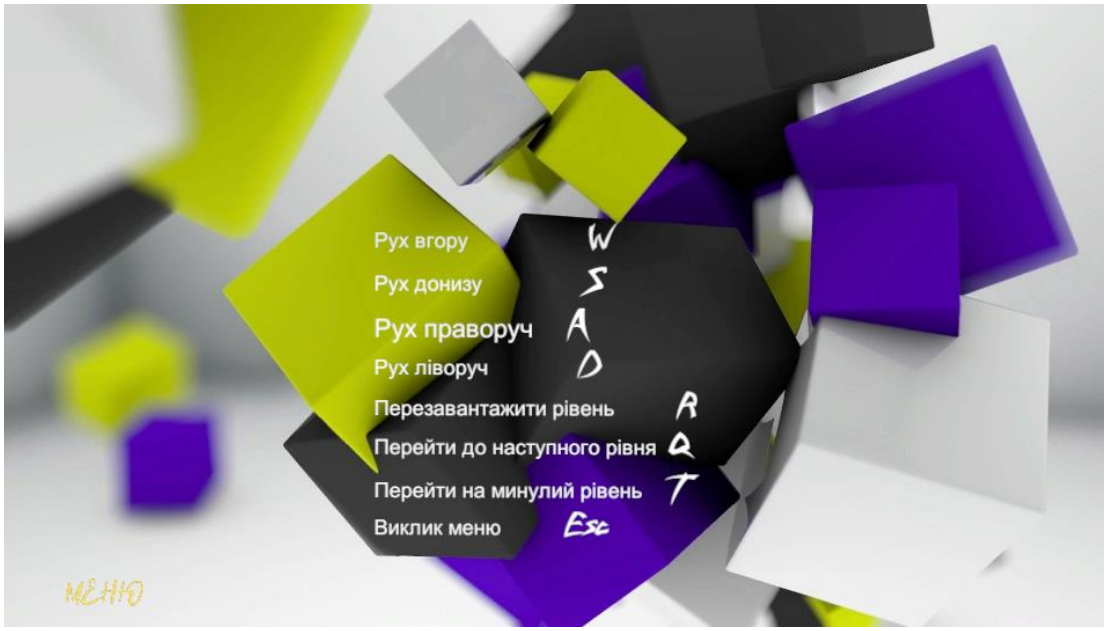
Мал. 2.1 Головне меню

При натиску кнопки Розпочати викликається функція LoadScene об’єкта керування сценами – SceneManager. Дана функція отримує як параметр індекс сцени, і завантажує її. В даному випадку вона завантажує сцену LVL1(детальніше додаток 2).

Подія кнопки Вихід - це завершення роботи програми. Це здійснено за допомогою функції Quit() об’єкта Application.

До кнопки Options додано функцію SetActive(bool isActive), яка викликається двічі. Перший раз для панелі стартових кнопок (Головного меню) для її деактивації (SetActive(false)), другий раз – для відкриття панелі меню опцій (SetActive(true)). Меню Options містить опис функцій, які будуть виконані при натисканні на певні кнопки.

Окрім того в меню Options містяться елементи Button з певним Normal Color та Highlighted Color і Pressed Color, що використовуються для відображення назви опції. Також внизу меню розміщена кнопка Back, яка також має свою анімацію, закриває меню опцій і відкриває попереднє - головне меню(див. мал. 2.2).



Мал. 2.2 Меню Опції

Керування аудіо на сцені Menu здійснює базовий елемент Canvas. Він містить компоненти типу AudioSource, які зберігають музику та звуки відповідно. Для джерела музики активні властивості PlayOnAwake (відтворення при створенні об’єкта).

2.2.2 Створення сцен Game

Сцена Game містить ігрове поле у вигляді ромба. На сцені розміщено три кнопки для керувати рівнями: «R» для перезапуску поточного рівня та «Q» для переходу на новий рівень, «T» для переходу на минулий рівень. По центру нашого ігрового поля розміщено жовтий куб(Player), а також певні перешкоди у вигляді інших кубів. Гравцю потрібно пройти перешкоди і дійти до виходу з ромба, який може бути розташований на будь-якій грані ромба, що автоматично запустить наступний рівень.

Гравець (Player) пересувається по полю гри горизонтально, методом ковзання по ромбу. Куби-перешкоди бувають кількох типів, а саме:

- статичний куб-перешкода (*GreyCube*),
- куб, що рухається відповідно гравцю(GreenCube),
- куб, що рухається перпендикулярно до гравця (*PinkCube*),
- ворог, що рухається від PointA до PointB (*movecube*),
- ворог, що рухається від PointA до заданої кінцевої точки PointX (*movecube1*),
- сфера, що рухається по заданій траєкторії (*PinkCube*),
- прямокутний паралелепіпед, що рухається вертикально(*guillotine)*
- куби які активуються натисканням кнопки на ігровій сцені.
- куб, який рухається за певною траєкторією.

Кольори кубів можуть змінюватись, для різноманіття ігрового процесу.

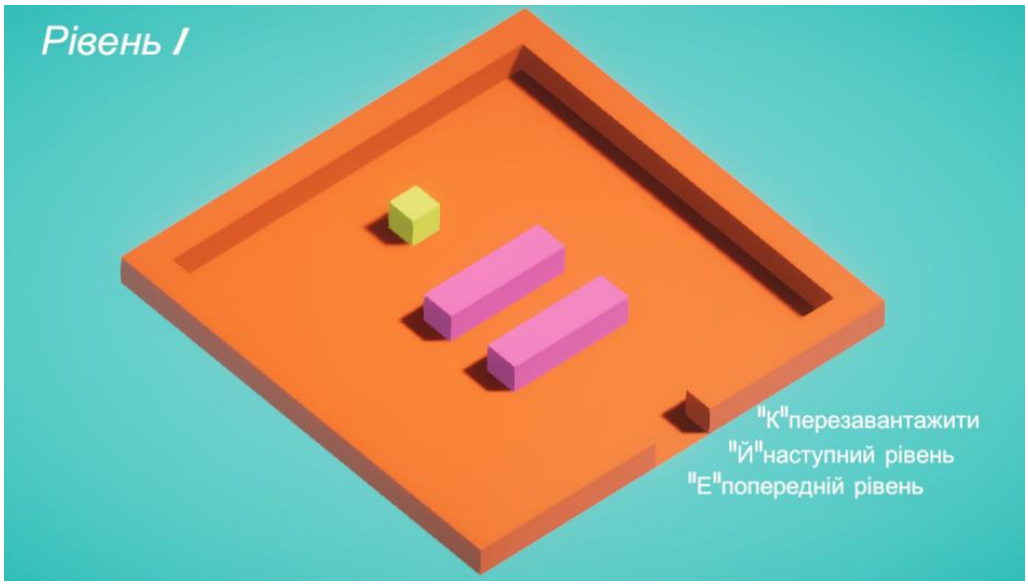
Оскільки вони будуть створюватися у великій кількості, було створено шаблони (prefab), для кожного куба. За допомогою шаблонів набагато простіше створювати сцену, оскільки не потрібно дублювати кожен об'єкт в ручну, а можна просто перетягнути (prefab) з відповідної папки Assets>Prefab. Деяким об’єктам в Unity є можливість задати тег. Це достатньо зручно коли об’єкт рухається по сцені, і інші об’єкти мають можливість перевірити, з чим

вони зіткнулись. Наприклад гравець має тег Player. Для переходу на інший рівень, гравець з тегом Player торкається невидимого куба з тегом Finish.

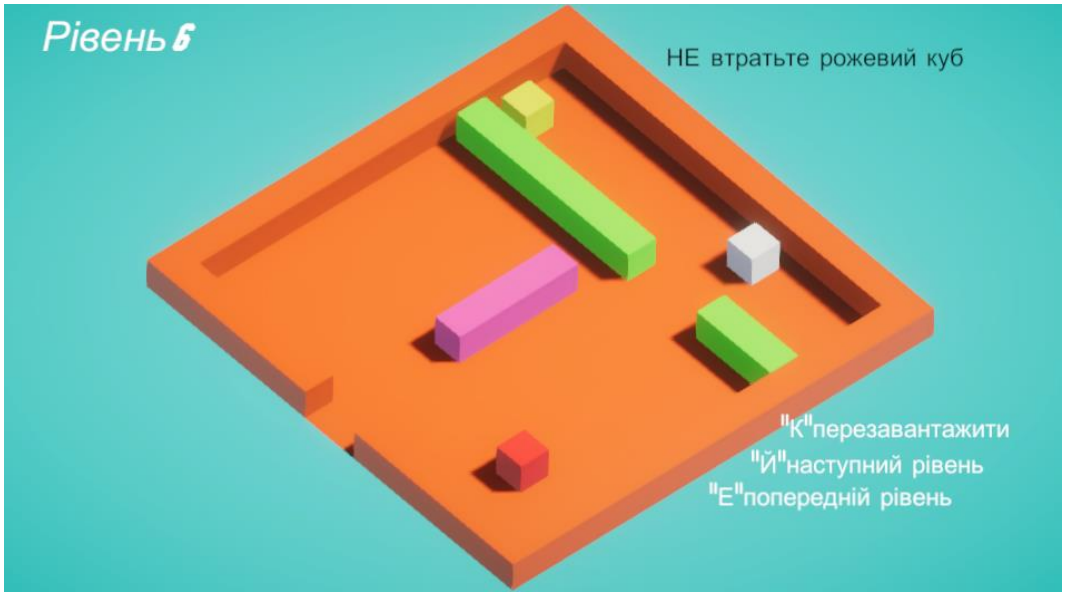
Гравець - це жовтий куб, власне, який рухається вгору і вниз. При зіткненні з різними кубами відбуваються певні дії. Наприклад, умовно зелений куб може переміщувати жовтий куб по сцені, що допомагає оминути перепони.

Ігровому об’єкту Player додано два компоненти: Rigidbody3D для надання і контролю рухом об’єкта через сценарій, а також BoxCollider3D для обробки зіткнень з кубами (детальніше додаток 1). Кубам в свою чергу, також надано компонент BoxCollider3D. У разі зіткнення двох елементів, що містять компонент BoxCollider3D, відбуваються певні дії, з урахуванням, того який саме куб з яким зіткнувся. В свою чергу невидимий куб-фініш має компонент BoxCollider3D, але з активною у ньому властивістю Is Trigger. Це вказує на те, що об’єкт є не твердим тілом, через який можуть проходити інші об’єкти. У такому випадку спрацює функція OnTriggerEnter().

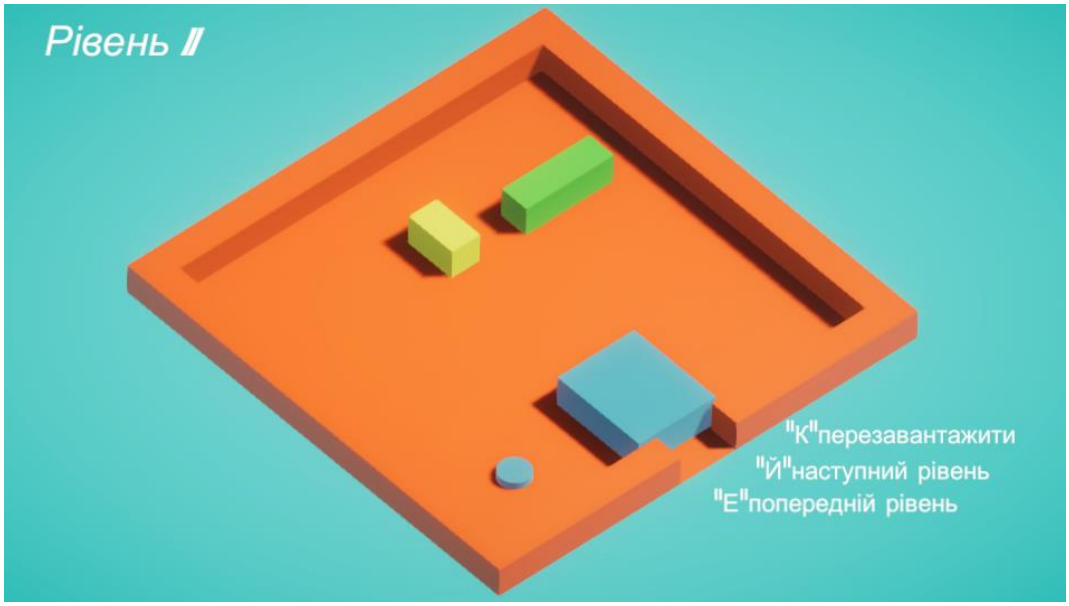
Під час запуску сцени гри з’являється, відповідно, сама ігрова сцена LVL1 та повідомлення для керування рівнями в нижньому правому кутку, також на деяких рівнях є підказки у верхньому правому кутку (див. мал. 2.3-2.9).



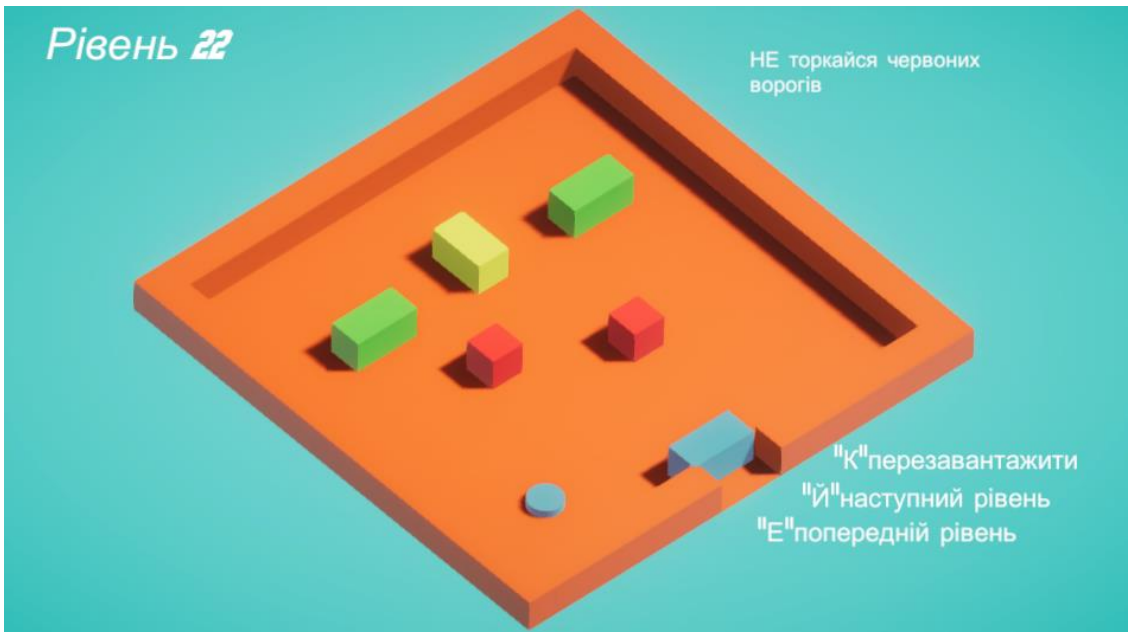
Мал. 2.3Початок гри LVL1



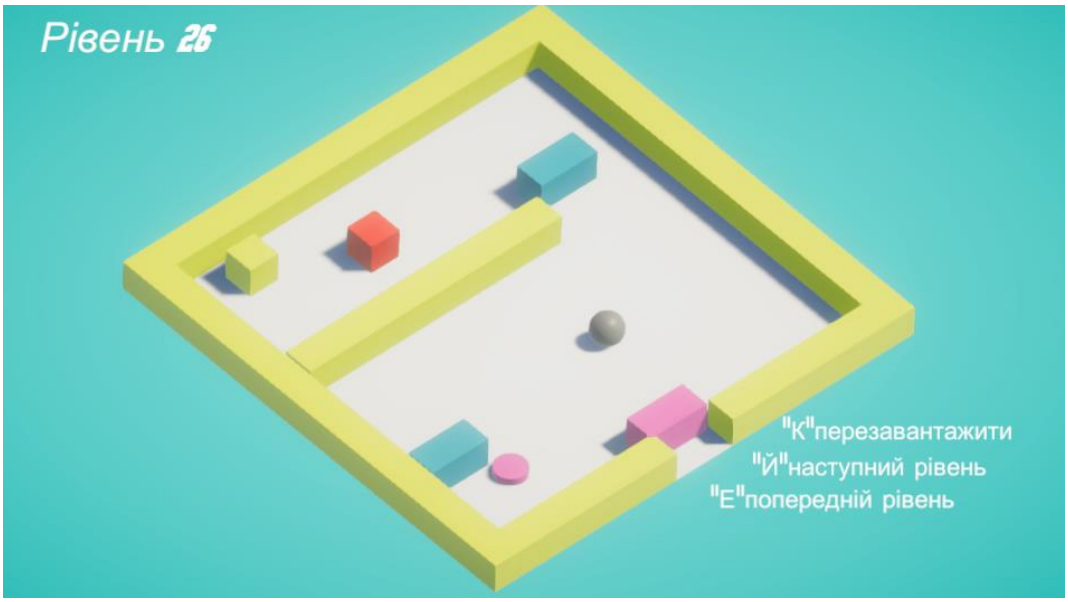
Мал. 2.4 LVL 6



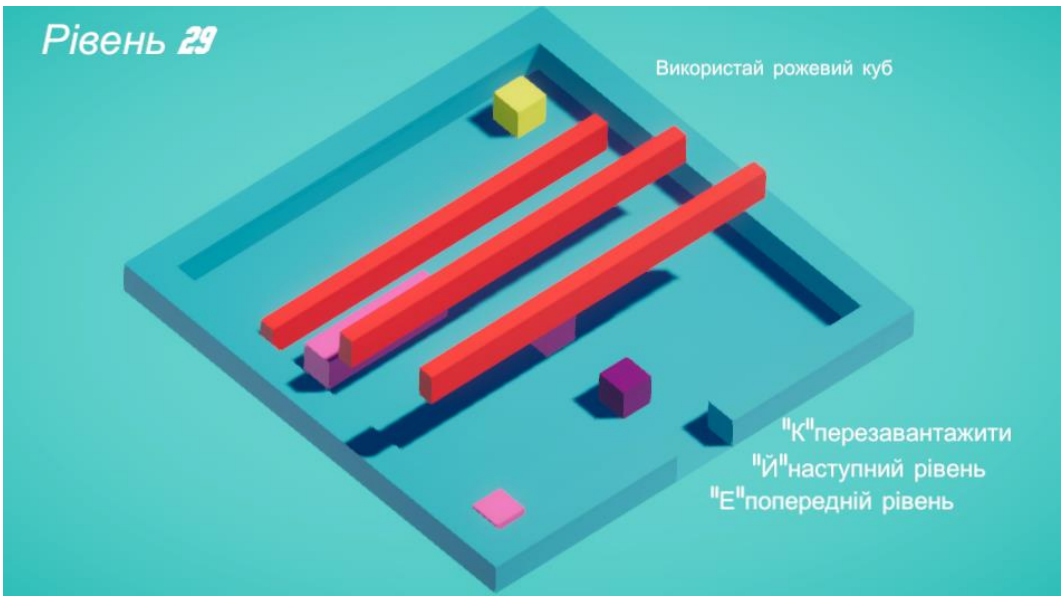
Мал. 2.5 Нова механіка кнопки LVL11



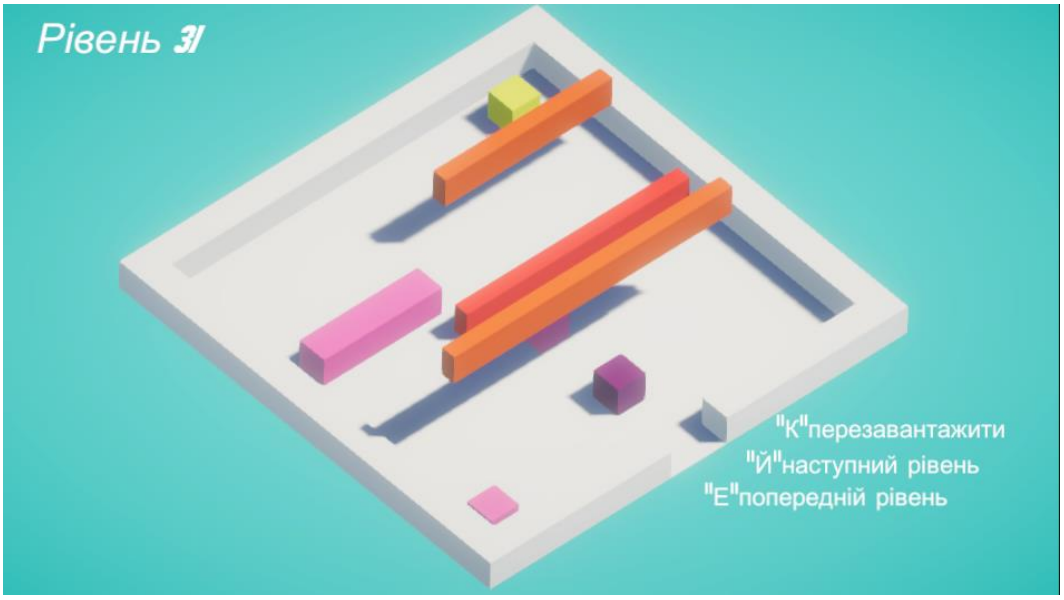
Мал. 2.6 Нова механіка вороги LVL22



Мал. 2.7 нова механіка куля LVL26



Мал. 2.8 нова механіка гільйотина (guillotine) LVL29



Мал. 2.9 Кінець гри LVL31

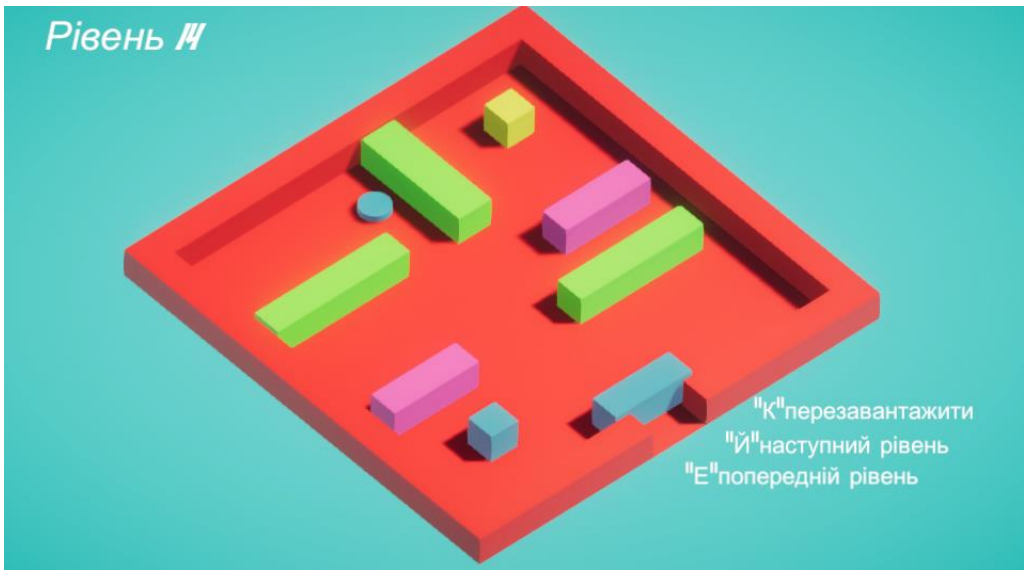
Відповідно при натисканні на клавішу Розпочати, ігровий процес запускається. В той же час музика на задньому фоні лунає завдяки музичному плеєру MusicPlayer (детальніше додаток 5). Компонент AudioSource відтворює звук та зациклює його за допомогою активних у ньому властивостей PlayOnAwake та Loop.

Після початку гри гравець починає рухатись за допомогою клавіш «W» вгору і «S» вниз. Перешкоди, або куби-перешкоди керуються гравцем за допомогою клавіш «A» вліво і «D» вправо, це незвичайна розстановка клавіш, яка додає більше цікавості в гру. У параметрах кожного куба вказано з якою швидкістю він може рухатись і по яких осях «X», «Y», «Z» відповідно. Як тільки гравець почне рух, його метою є пройти всі перешкоди, які трапляються йому на шляху, та досягнути куба з тегом Finish(детальніше додаток 1). Він має компонент BoxCollider3D, з активною у ньому властивістю Is Trigger. Оскільки куб-фініш взаємодіє з Player, то він має його якось розпізнати, і в цьому випадку допоможе функція :

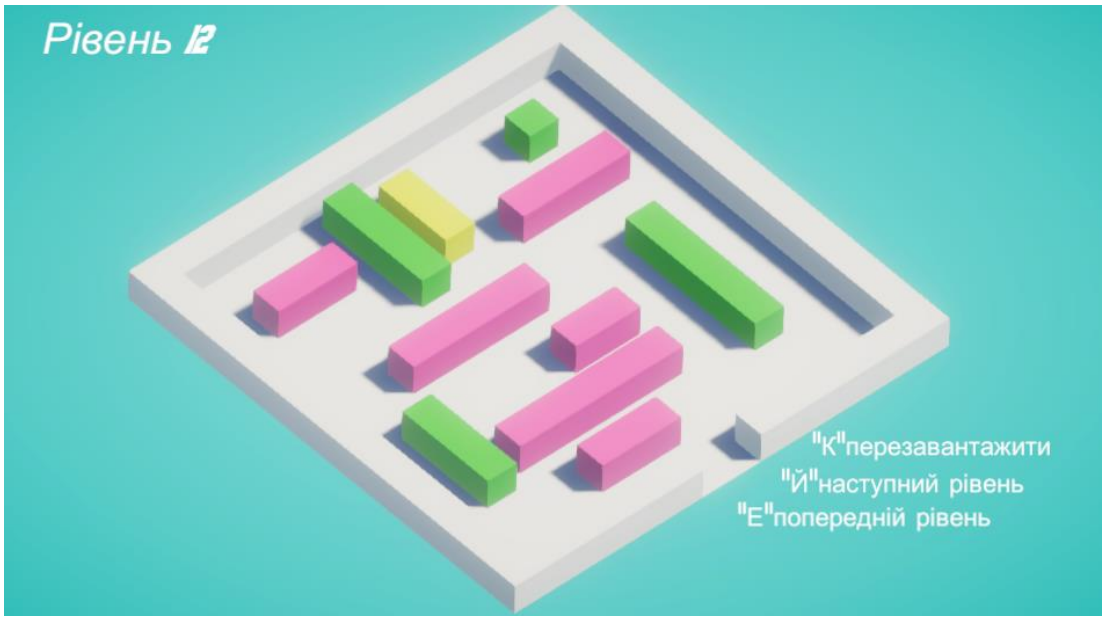
`if` (this.CompareTag("Player") && other.CompareTag("Finish")),

яка описує дію після дотику гравця з тегом (“Player”) та куба з тегом("Finish")). Після зіткнення нашого гравця з цим невидимим кубом буде активована функція OnTriggerEnter(), яка вмикає перехід на інший рівень.

Чималу роль відіграє ще одна механіка, яка додає різноманіття в ігровий процес, а саме механіка кнопок. Вона працює за таким принципом: при натисканні кубом на кнопку перемикаються два стани кубів, які прив’язані до цієї кнопки. Перший стан - це firstGroup Material normal, вказує на масив об’єктів першої групи, secondGroupe Material transparent вказує на масив об’єктів другої групи. Спочатку всі куби, що прив’язані до кнопки є в першому стані, через них не можна пройти поки не буде натиснута хоча б одна кнопка на сцені. При натисканні кнопки кубом відбувається перехід об’єкта з першої групи в другу. Також відбувається взаємодія між кнопками, в разі якої при натисканні на одну кнопку перша група кубів, що до неї прив’язані стає невидимою (secondGroupe) і через них можна проходити, а інша група кубів, що прив’язана до другої кнопки стає активною і переходить в першу групу (firstGroup) (детальніше додаток 4) (див. мал. 3-3.3).



Мал. 3 Ігровий процес

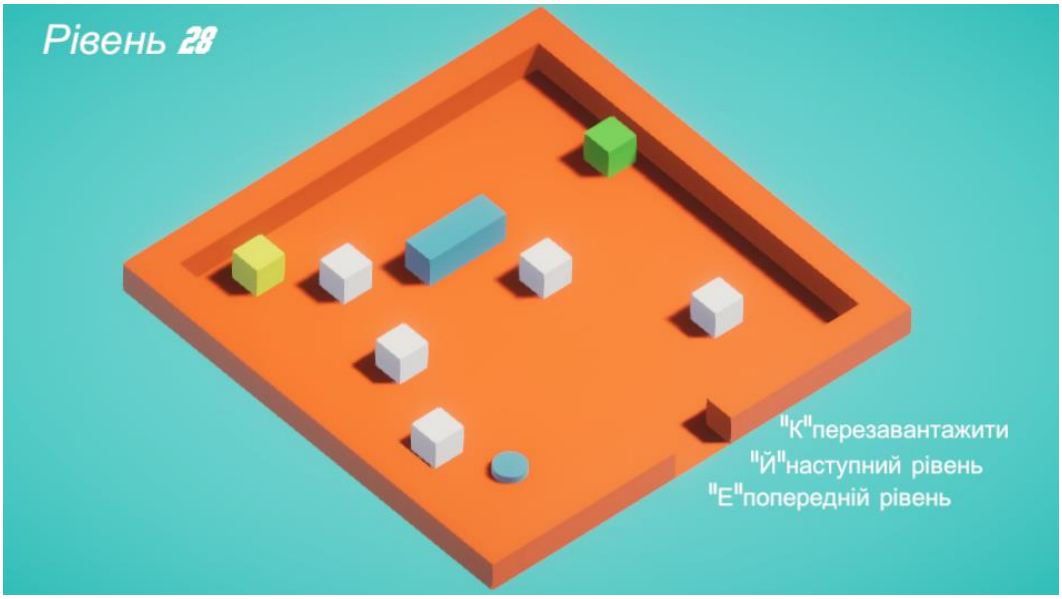


Мал. 3.1 Ігровий процес

Існує ще одна чудова механіка реалізована у грі - це механіка ворогів. Вороги у грі представленні кубами, сферами та перешкодами які рухаються вертикально вгору і відповідно вниз в певному циклі. Кожна перешкода має відповідну невидиму точку початку руху Point0 та наступну точку Point1.(детальніше додаток 7) Рух може як відбуватися циклічно так і бути від Point0 до Pointn.(детальніше додаток 8)

Слід відмітити, що для кожного типу ворогів було прописано відповідні скрипти. Перші вороги яких зустрічає гравець представляють собою куби червоного кольору, що рухаються з початку запуску сцени, при торканні до такого ворога, за допомогою скрипта, рівень починається з початку.

Кількість ворогів їх розташування, швидкість і кількість точок від яких і до яких прямує ворог, можна налаштувати відповідно як через скрипт так і через редактор у самому юніті.Ось ілюстрація ігрового процесу де зображено декілька ворогів, у кожного своя траєкторія руху, швидкість, та напрямок(див. мал.3.2).



Мал. 3.2 механіка ворогів

Також у грі реалізована можливість паузи(Pause). Вона вмикається при натисканні кнопки Esc, після чого ігровий час зупиняється. Це відбувається завдяки функції (Time.timeScale = 0f). Змінюючи значення цієї функції, для симуляції bullet-time ефекту, можна регулювати швидкість протікання часу(детальніше додаток 3). При значенні 1 час проходить так само швидко, як і в режимі реального часу. При значенні 0.5 час сповільнюється на половину; при значенні 2 час прискорюється в 2 рази. Зрозуміло, що при значенні 0 час зупиняється. В меню паузи присутні декілька кнопок, які дозволяють маніпулювати ігровим процесом. Перш за все – це кнопка Розпочати, яка дозволяє продовжити гру, також можна знову натиснути на кнопку Esc для продовження ігрового процесу. При цьому значення (Time.timeScale = 1f). Наступною є кнопка Меню, яка повертає нас в сцену (Menu) головне меню. Також присутня кнопка Вихід, при натисканні на яку відбувається вихід з гри.

Мал. 3.3 Меню паузи

2.3. Опис гри

При запуску додатку з’являється головне меню гри, що складається з трьох кнопок: Розпочати, Налаштування та Вихід.

При натисканні на кнопку Налаштування відкривається однойменне меню. У ньому є можна ознайомитись з деякими ігровими атрибутами та можливостями, наприклад, виклик меню паузи. Для виходу з меню Options(опії) до головного меню необхідно натиснути кнопку Back(назад), що знаходиться внизу.

Кнопка Вихід завершує роботу гри.

Кнопка Розпочати запускає ігровий процес. Після чого відкриється сама гра.

Одразу з’явиться ігрове поле та можна починати сам ігровий процес. У верхній лівій частині екрану розміщено лічильник рівнів ,наприклад, спочатку це рівень 1 (LVL1). Також в правому нижньому кутку є підказки для керування рівнями, а саме «R» для перезапуску поточного рівня, «Q» для переходу на наступний рівень та «T» для переходу на попередній рівень.

Гравець (Player) пересувається по полю гри горизонтально, методом ковзання по ромбу. Куби-перешкоди бувають кількох типів, а саме: звичайна (GreyCube), куб, що рухається так само як гравець(GreenCube), куб, що рухається перпендикулярно до гравця (PinkCube), куби-вороги, що рухається від PointA до заданої кінцевої точки PointX (*movecube1*), сфери, що рухається по заданій траєкторії (*PinkCube*), прямокутний паралелепіпед, що рухається вертикально(*guillotine)*, куби які активуються натисканням кнопки на ігровій сцені, а також куби, які рухаються за певними траєкторіями. Кольори кубів можуть змінюватись, для різноманіття ігрового процесу. Гравець повинен проявити уважність і вибирати оптимальний алгоритм проходження рівня, у випадку якщо гравець помилився він завжди може перезапустити рівень спочатку і почати продумувати іншу стратегію.

Завдання гравця – це пройти всі рівні гри. Це допоможе з користю провести час та розвине у гравця критичне мислення, просторову уяву, вміння вирішувати головоломки, та приймати нестандартні рішення. Основна мета цієї гри – це прагнення досягти більшого, перевершити себе.

Під час ігрового процесу є можливість призупинити гру, натиснувши кнопку Esc. Тоді час в грі “заморозиться” та з’явиться меню паузи. До цього меню входять наступні кнопки:

- Розпочати, для продовження припиненої гри (дану дію можна також здійснити знову натиснувши Esc в припиненній грі).
- Меню, для повернення до головного меню.
- Вихід, для виходу з гри.

Висновки

Під час виконання кваліфікаційної роботи для досягнення мети опрацьовано такі завдання:

- досліджено сучасні підходи, методи та способи для створення ігрового додатку;
- досліджено проблематику і потребу у продукті;
- обрано технології розробки Unity3d та проведено її дослідження;
- засвоєно теоретичні та практичні навички у гейм розробці;
- проаналізовано необхідність та можливість використання додатку;
- створено додаток «Puzzle Cube» для розвитку інтелектуальних здібностей користувачів;
- тестування та аналіз розробленого додатку;

Гра розвиває критичне мислення, просторову уяву, вміння вирішувати головоломки та приймати нестандартні рішення, також вчить цілеспрямованості у досягненні поставлених цілей, сприяє інтелектуальному розвитку користувача.

У розробленого додатку «Puzzle Cube» існує можливості розширення та модернізації існуючого функціоналу шляхом додавання нових механік, розширення рівнів, зміни точок зору користувача, додавання кооперативу, зміни масштабу рівні, існує можливість розширювати та зменшувати 3d об’єкти тощо.

Список літератури

1. What is C#? [Електронний ресурс] https://docs.microsoft.com/en-us/dotnet/csharp/.
2. C# Programming Guide. [Електронний ресурс] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/.
3. Unity Documentation. [Електронний ресурс] https://docs.unity3d.com/Manual/index.html.
4. Learning C# Programming for Unity. [Електронний ресурс] https://learn.unity.com/course/programming.
5. Unity in Action: Multiplatform Game Development in C# by Joe Hocking.
6. Unity Game Development Essentials by Will Goldstone.
7. Official Unity Scripting API Reference [Електронний ресурс] http://docs.unity3d.com/ScriptReference/.
8. Unity Game Development Tutorial for Beginners. [Електронний ресурс] https://www.udemy.com/course/unitycourse/.
9. Посібник користувача Unity[Електроний ресурс]: https://docs.unity3d.com/Manual/index.html
10. Про створеня платформера на Unity[Електроний ресурс]: https://habr.com/ru/company/microsoft/blog/236125/
11. Офіційний скриптовий довідник[Електроний ресурс]: http://docs.unity3d.com/ScriptReference/
12. Ігровий рушій Unity[Електроний ресурс]: https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8)

Додатки

Додаток 1

(Player.cs**)**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; //для перехода на інший рівень

public class Player : MonoBehaviour
{
    [SerializeField] KeyCode keyOne;
    [SerializeField] KeyCode keyTwo;
    [SerializeField] Vector3 moveDirection;

    private void FixedUpdate() // керування
    {
        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);//
перезагружаємо поточну сцену

SFXManeger.sfxInstance.Audio.PlayOneShot(SFXManeger.sfxInstance.Click)
;////////////////////// для звука клавіш
        }
        if (Input.GetKey(KeyCode.Q))
        {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);// переходемо до наступного рівня

SFXManeger.sfxInstance.Audio.PlayOneShot(SFXManeger.sfxInstance.Click)
;//////////////////////////
        }
        if (Input.GetKey(KeyCode.T))
        {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex -
1);// переходемо до попереднього рівня
```

```
SFXManeger.sfxInstance.Audio.PlayOneShot(SFXManeger.sfxInstance.Click)
;//////////////////////////
        }


    }

    private void OnTriggerEnter(Collider other)
    {
        if (this.CompareTag("Player") &&
other.CompareTag("movecube"))// якщо гравець торкається рухливого куба
        {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);//
перезагружаємо поточну сцену
        }
        if (this.CompareTag("Player") && other.CompareTag("Finish"))//
якщо гравець торкається фінішу
        {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex +
1);// загружаємо наступну сцену
        }
        if (this.CompareTag("Cube") && other.CompareTag("Cube") )//
якщо куб торкається куба
        {
            foreach(Activator button in
FindObjectsOfType<Activator>())
            {
                button.canPush = false;
            }
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (this.CompareTag("Cube") && other.CompareTag("Cube") )//
якщо гравець торкається кубу
        {
            foreach (Activator button in
FindObjectsOfType<Activator>())
            {
                button.canPush = true;
            }
        }
    }
}
```

Додаток 2

(MainMenu.cs**)**

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

public class MainMenu : MonoBehaviour {
    public void PlayGame()//для того щоб почати гру, сцени будуть йти
по списку
    {

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    public void QuitGame()//для того щоб вийти з гри
    {
        Debug.Log("Quit");
        Application.Quit();//ця дія буде закривати гру
    }
    public void LoadMenu()//для того щоб повернутись в головне меню
    {
        Debug.Log("Load");
        Time.timeScale = 1f;// щоб уникнути бага в грі ніби там
уповільненого часу в 0f
        SceneManager.LoadScene("Menu");// буде повертатись в головне
меню

    }
}
```

Додаток 3

(Pause_menu.cs**)**

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


public class Pause_menu : MonoBehaviour
{
    public static bool GameIsPause = false;// при старті дія
GameIsPause false (гра не на паузі)

    public GameObject pauseMenuUI;

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))// при натисканні на
Escape заходить і виходить з меню паузи
        {
            if (GameIsPause)
            {
                Resume();// якщо гра на паузі то ми маємо продовжити
гру
            }
            else
            {
                Pause();//в іншому випадку пауза
            }
        }
    }

    public void Resume()//продовжувати гру
    {
        pauseMenuUI.SetActive(false);// дозволяє визивати меню паузи
коли ми продовжуємо гру false меню паузи не відображається
        Time.timeScale = 1f;//швидкість гри відновлено
        GameIsPause = false;// гра не на паузі продовжуємо грати)
    }

    void Pause()
    {
        pauseMenuUI.SetActive(true);// дозволяє визивати меню паузи
true тобто під час паузи меню повинне з’явитися
        Time.timeScale = 0f;//дія для того щоб зупинити весь процес
гри уповільнює часу в 0f
```

```
        GameIsPause = true;
    }

    public void LoadMenu()//для того щоб повернутись в головне меню
    {
        Debug.Log("Load");
        Time.timeScale = 1f;// щоб уникнути бага в грі ніби там
уповільненого часу в 0f
        SceneManager.LoadScene("Menu");// буде повертатись в головне
меню

    }

    public void QuitGame()//для того щоб вийти з гри
    {
        Debug.Log("Quit");
        Application.Quit();//ця дія буде закривати гру
    }
}
```

Додаток 4

(Activator.cs**)**

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

public class Activator : MonoBehaviour
{
    public GameObject[] firstGroup; // вказую масив об'єктів в першу
групу
    public GameObject[] secondGroup; // вказую масив об'єктів в другу
групу
    public Activator button;
    public Material normal;
    public Material transparent;
    public bool canPush; // добавив нову зміну для регулювання кнопок
і блоків

    private void OnTriggerEnter(Collider other)
    {
        if (canPush)
        {
            if (other.CompareTag("Cube") ||
other.CompareTag("Player"))//якщо кнопки торкнувся звичайний куб або
гравець
            {
                foreach (GameObject first in firstGroup)// для кожного
об'єкта в 1 групі(normal)
                {
                    first.GetComponent<Renderer>().material = normal;
                    first.GetComponent<Collider>().isTrigger =
false;// через блоки не можна буде проходити
                }
                foreach (GameObject second in secondGroup)// для
кожного об'єкта в 2 групі(transparent)
                {
                    second.GetComponent<Renderer>().material =
transparent;
                    second.GetComponent<Collider>().isTrigger =
true;// через блоки можна буде проходити
                }
                GetComponent<Renderer>().material = transparent;//
матеріал кнопки робимо прозрачним а матеріал другої обичним
                button.GetComponent<Renderer>().material = normal;
                button.canPush = true;
            }
        }
    }
```

Додаток 5

(SFXManeger.cs)

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

public class SFXManeger : MonoBehaviour
{
    public AudioSource Audio;

    public AudioClip Click;

    public static SFXManeger sfxInstance;

    private void Awake()
    {
        if (sfxInstance != null && sfxInstance != this)
        {
            Destroy(this.gameObject);
            return;
        }

        sfxInstance = this;
        DontDestroyOnLoad(this);
    }
}
```

Додаток 6

(DgScript.cs)

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

public class BgScript : MonoBehaviour
{
    public static BgScript BgInstance;

    private void Awake()// музика грає у фоновому режимі і коли закінчиться
почнеться спочатку
    {
        if(BgInstance != null && BgInstance !=this)
        {
            Destroy(this.gameObject);
            return;
        }

        BgInstance = this;
        DontDestroyOnLoad(this);
    }
}
```

Додаток 7

(MoveBetweenPoints.cs**)**

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



public class MoveBetweenPoints : MonoBehaviour
{
    public Transform pointA; // Об'єкт PointA
    public Transform pointB; // Об'єкт PointB
    public float speed = 2.0f; // Швидкість руху

    private Transform currentTarget; // Поточна ціль для руху

    private void Start()
    {
        currentTarget = pointA; // Початкова ціль - PointA
    }

    private void Update()
    {
        float step = speed * Time.deltaTime;
        transform.position = Vector3.MoveTowards(transform.position,
currentTarget.position, step);

        if (transform.position == currentTarget.position)
        {
            // Якщо досягнута поточна ціль, змінюємо ціль на інший
об'єкт
            if (currentTarget == pointA)
            {
                currentTarget = pointB;
            }
            else
            {
                currentTarget = pointA;
            }
        }
    }
}
```

Додаток 8

(MoveABCD.cs**)**

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

public class MoveABCD : MonoBehaviour

{
    public Transform[] points; // Масив точок, в які рухаємося
    public float speed = 2.0f; // Швидкість руху

    private int currentTargetIndex = 0; // Індекс поточної цілі

    private void Start()
    {
        if (points.Length > 0)
        {
            currentTargetIndex = 0; // Починаємо з першої точки
            UpdateTargetPosition(); // Викликаємо метод для оновлення
поточної цілі
        }
    }

    private void Update()
    {
        float step = speed * Time.deltaTime; // Визначення кроку руху
        transform.position = Vector3.MoveTowards(transform.position,
points[currentTargetIndex].position, step); // Рух до поточної цілі

        if (transform.position == points[currentTargetIndex].position)
        {
            currentTargetIndex++; // Зміна цілі на наступну
            if (currentTargetIndex >= points.Length)
            {
                currentTargetIndex = 0; // Починаємо знову з першої точки,
якщо досягнута остання
            }
            UpdateTargetPosition(); // Викликаємо метод для оновлення
поточної цілі
        }
    }

    private void UpdateTargetPosition()
    {
        // Оновлюємо поточну ціль для руху
        currentTargetIndex = Mathf.Clamp(currentTargetIndex, 0, points.Length
- 1); // Обмеження індексу у межах діапазону точок
    }
}
```