Державний вищий навчальний заклад

«Прикарпатський національний університет імені Василя Стефаника»

Факультет математики та інформатики

Кафедра комп'ютерних наук та інформаційних систем

КУРСОВА РОБОТА

Тема: "Розробка навчальної гри з використанням інтегрованого середовища Unity 2D"

Виконали
студенти 3 курсу групи КН-3
Напряму підготовки "Комп'ютерні науки"
Предко Олександр _____
Фуштей Владислав _____

Керівник
кандидат технічних наук, доцент
Ровінський Віктор Анатолійович

_____

Національна шкала: _____

Університетська шкала: _____

Оцінка ECTS: _____

Члени комісії:

_____

(підпис) (прізвище та ініціали)

_____

(підпис) (прізвище та ініціали)

_____

(підпис) (прізвище та ініціали)

м. Івано-Франківськ
2020 рік

ЗМІСТ

ВСТУП

В наш час неможливо уявити повсякденне життя без постійного потоку інформації. Для задоволення цих потреб використовуються сучасні засоби мультимедіа: телебачення, Інтернет, тощо.

Мультимедіа — комбінування різних форм представлення інформації на одному носієві. Мультимедіа може бути грубо класифікована як лінійна і нелінійна. Аналогом лінійного способу подання може бути кіно. Людина, що переглядає даний документ жодним чином не може вплинути на його зміст. Нелінійний спосіб подання інформації дозволяє людині брати участь у поданні інформації, взаємодіючи якимось чином із засобом відображення мультимедійних даних. Участь людини в даному процесі також називається «інтерактивністю». Такий спосіб взаємодії людини й комп'ютера найбільш повно представлений у категоріях комп'ютерних відео ігор.

Відеогра — це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відеопристрою. Електронні пристрої, які використовуються для того щоб грати, називаються ігровим платформами. Наприклад, до таких платформ належать персональний комп'ютер та гральна консоль. Пристрій введення, який використовується для керування грою, називається ігровим контролером. Це може бути, наприклад, джойстик, клавіатура та мишка, геймпад або сенсорний екран. Комп’ютерні відеоігри ігри набули небувалої популярності за останні роки та посіли почесне місце на ринку розваг та дозвілля. Як результат, бурний розвиток індустрії розробки комп‘ютерних ігор. Для задоволення потреб розробників було створено величезну кількості інструментальних засобів та продовжується розробка нових. Загальна ціль всіх інструментальних засобів - покращення розробки, використання передових технологій обробки графіки, фізики, забезпечення кросплатформенності розроблених проектів[1].

Метою даної роботи є створення ігрової платформи на Unity на основі неї написання невеликої демонстраційної гри, яка може залучити потенційних покупців в стрімко розширюється ринку комп'ютерних розваг. Так само створена платформа є основою для створення наступних програмних продуктів значно скоротивши час їх створення на 70%.

Тема курсової роботи "Розробка навчальної гри з використанням інтегрованого середовища Unity 3D" на даний момент є актуальною в силу того, що Unity все більше отримує користувачів, викликає непідробний інтерес у тисяч розробників ігор по всьому світу. Тому наша курсова робота має високу актуальність і може використовуватися як посібник з розробки з використанням платформи Unity. Дана платформа націлена на ринок казуальних розважальних програм створюваних розробниками, а не багатомільйонних програмних колосів з гігантськими бюджетами, тому вона привертає безліч інді-розробників, які можуть скористатися її можливостями. Дана курсова робота зводить над Unity розгорнуту високорівневу платформу (фреймворк) і надає багато зручні механізми взаємодії з "залізом" за допомогою зрозумілих функцій. Безліч рутинних операцій у фреймворку робиться автоматично без участі програміста. Unity це ігровий движок, що дозволяє створювати гри під більшість – це популярних платформ. За допомогою даного движка розробляються ігри, які запускаються на персональних комп'ютерах (які працюють під Windows, MacOS, Linux), на смартфонах і планшетах

(iOS, Android, Windows Phone), на), на смартфонах і планшетах (iOS, Android, Windows Phone), на ігрових консолях (PS, Xbox), на смартфонах і планшетах (iOS, Android, Windows Phone), на, Wii).

1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Мультимедійні ігри

В сфері мультимедіа не останнє місце займає таке явище, як мультимедійні комп'ютерні ігри. Мультимедійні ігри — такі ігри, у яких гравець взаємодіє з віртуальним середовищем, побудованих комп'ютером. Стан віртуального середовища передається гравцеві за допомогою різних способів передачі інформації (аудіальний, візуальний, тактильний). Наразі всі комп'ютерні ігри відносяться до мультимедійних ігор. В такий тип ігор можна грати як в поодинці на локальному комп'ютері або приставці, так і з іншими гравцями через локальну чи глобальну мережу. У 2011 році відеоігри були офіційно визнані видом мистецтва урядом США та Національним фондом мистецтв США[2].

1.2 Ігрова індустрія

Перша комп‘ютерна гра «Зоряні війни» вийшла у світ 1962 року. Її завдання полягало в тому, щоб відбити астероїди і напади ворожих космічних кораблів. Згодом було створено багато інших ігор. А з поширенням у 1970– 1980 роках потужніших комп‘ютерів електронних ігор побільшало: пригодницькі ігри, ігри-головоломки, стратегічні ігри та ігри «екшн». Багато ігор імітують різні види спорту, як от хокей на льоду чи гольф. Чимало з них здобули високу оцінку громадськості,оскільки вони дуже цікаві й допомагають у навчанні[1].

Стає дедалі популярнішим онлайновий вид комп‘ютерної гри. Її персонажами керує не комп‘ютер, а гравці, як і через Інтернет одночасно беруть участь у грі. Їх можуть бути тисячі. Популярність таких забав пояснюється можливістю поспілкуватися з іншими. Гравці ―розмовляють одні з одними і відчувають себе частиною всесвітньої родини. В Україні щороку зростає кількість людей, що купують комп‘ютерні ігри. Якщо для гравців це просто забавка, то для розробників, виробників та розповсюджувачів — досить вигідний бізнес. Світовий ринок комп‘ютерних ігор оцінюють в сотні мільярдів доларів. На Заході один ліцензійний ігровий диск коштує $40–50. Популярну гру можуть продати накладом від мільйона до кількох десятків мільйонів примірників. Не дивно, що в розробку гри там можуть легко вкласти кілька мільйонів доларів. Ця індустрія приносить великі прибутки і державній скарбниці. Комп‘ютерні ігри стали вже й елементом політики. Парламенти західних країн дискутують щодо законодавчого обмеження насильства в комп‘ютерних іграх або ж стимулювання виробництва ігор як такого[3].

1.3 Класифікація комп’ютерних ігор

Комп'ютерні ігри в основному класифікуються за жанрами, а також за кількістю гравців. Внаслідок того, що критерії приналежності гри до того чи іншого жанру не визначені однозначно, класифікація ігор недостатньо систематизована, і в різних джерелах дані про жанр конкретного проекту можуть розрізнятися. Проте, існує консенсус, до якого прийшли розробники ігор, і приналежність гри до одного з основних жанрів майже завжди можна визначити однозначно. Ці найбільш популярні жанри (які об'єднують в собі безліч піджанрів) перераховані нижче. Існують ігри з елементами кількох жанрів, які можуть належати кожному з них (наприклад, серія Grand Theft Auto, Космічні Рейнджери, Rome: Total War і багато інших). Такі проекти зараховують або до одного з жанрів, який в грі є основним, або відразу до всіх, присутніх в грі, якщо вони в

рівній мірі становлять геймплей проекту. В основі сучасних розподілів відеоігор на жанри лежить вид активності, який найчастіше здійснює гравець в іграх даного жанру. Так відеоігри в загальному можуть поділятися на ігри руху, планування і сюжету або спілкування, дії та контролю. В багатьох класифікаціях визначення жанру відбувається за кількома осями. Наприклад, за двома осями сюжет — свобода 14 дії, або трьома абстракція — симуляція — свобода[4]. Проте найчастіше використовуваною класифікацією, хоч і не прийнятою усіма, жанри з якої зустрічаються в більшості існуючих, є наведена нижче, яка виключає осі або багаторівневі поділи[5]:

Action

В іграх такого жанру необхідно використовувати рефлекси та швидкість реакції для подолання ігрових обставин. Це один із базових жанрів і водночас найпоширеніший. Як правило екшн-ігри пов'язані із агресивними діями щодо противників і/або оточення. Персонаж гравця повинен битися, стріляти, переслідувати ціль чи самому уникати переслідування. Стосовно екшн-ігор, де наявні значні елементи пригодницьких ігор, застосовується термін Action adventure. З-поміж них часто виділяється напрям аркадних ігор, ігровий процес яких вирізняється простотою та легкістю освоєння.

Стратегія

Сенс стратегічних ігор полягає в плануванні дій та виробленні певної стратегії для досягнення якоїсь конкретної мети, наприклад, перемоги у військовій операції. Гравець керує не одним персонажем, а цілим підрозділом, підприємством чи навіть всесвітом. Відповідно до реалізації ігрового часу, стратегічні відеоігри поділяються на два основних різновиди: Покрокові стратегічні ігри — гравець та його противник здійснюють дії один за одним, покроково, маючи змогу за один ігровий хід виконати певну кількість операцій. Приклади: Heroes of Might and Magic III, Цивілізація.

Стратегічні ігри в реальному часу — і гравець і противник виконують свої дії одночасно, проте часто часто масштаб часу відрізняється від реального. Наприклад, будівництво триває кілька секунд, а ігрова година складає кілька хвилин реального часу. Приклади: StarCraft, Age of Empires III. Tower Defense — похідний жанр, в якому гравець керує оборонними баштами аби не пропустити хвилі ворогів. MOBA — похідний жанр, в якому гравець керує одним персонажем з метою захистити свою базу від ворогів та знищити ворожу.

Рольова гра

Гравець асоціюється з конкретним персонажем або лідером команди, які діють відповідно до правил своїх ролей. Наприклад, лицар не може того, що чарівник, кожна роль має свої особливості, а часом від неї залежить і розвиток сюжету. Мета ігрового процесу полягає у виконанні різноманітних завдань (квестів) для розвитку одного персонажа або групи. Можливі варіанти дій залежать від обраного образу персонажа, попередньо визначеного чи формованого самим гравцем. Рольові відеоігри часто поєднуються з іншими жанрами, утворюючи Action-RPG, тактичні рольові ігри, MUD-и і т.п.. Особливим випадком рольової гри є MMORPG, багатокористувацькі рольові онлайн-ігри, в яких живі гравці взаємодіють одне з одним у віртуальному світі через мережу Інтернет, що визначає специфіку ігрового процесу. Іноді рольові відеоігри поділяються за дизайном і побудовою сюжету. Так існує умовний поділ на рольові ігри західного зразка та східного.

Симулятор

В широкому розумінні всі ігри є симуляторами. У вужчому значенні це відеоігри, призначені для складання уявлення про дійсність за допомогою відображення певних реальних явищ та властивостей у віртуальному середовищі. Існує чимало піджанрів, як технічні (управління складними технічними пристроями, авіаційною технікою та інші, наприклад гра Іл-2 Штурмовик), аркадні (відрізняються від аркад наявністю спрощеної фізичної моделі. Наприклад, X-Wing), спортивні, економічні, побачень та інші.

Пригоди

В пригодницьких відеоіграх гравець керує ігровим персонажем, який рухається по сюжету та виконує зумовлені сценарієм завдання, покладаючись на свою уважність та логіку, здійснює пошуки підказок і вирішує загадки. Всередині жанру виділяються основні піджанри: інтерактивна література, інтерактивні фільми та візуальні романи. Часто за аналогією до пригодницьких фільмів пригодницькими називаються ті відеоігри, сюжет яких динамічно розгортається, насичений яскравими подіями, швидкою зміною обстановки, а персонажі проявляють кмітливість та сміливість. Приклади: серія про Індіану Джонса, Disney's Aladdin in Nasira's Revenge, Syberia.

Інші різновиди

Настільна гра — електронні реалізації настільних ігор. Серед них існують як реалізації класичних ігор (шахи, преферанс), так і специфічних як Magic: The Gathering. Головоломки — вимагають від гравця вирішення логічних завдань, передбачення можливих ситуацій. Приклади: Тетріс, Angry Birds. Games with a Purpose — програми, за допомогою яких люди використовують свої знання для вирішення проблем (передусім у наукових дослідженнях), при цьому граючись.

1.4 Кіберспорт

Кіберспорт — спортивні змагання з відеоігор. Історія електронного спорту почалася з гри Quake, яка мала режим мережевої гри через LAN або інтернет. Завдяки популярності гри Doom, в 1997 р. в США з'явилася перша ліга електронного спорту — Cyberathlete Professional League (CPL). Від цього року з'явилися багато нових ліг із кіберспорту. Змагання з кіберспорту, у тому числі і міжнародні, проводяться по всьому світі. Найзначнішим з них є турнір World Cyber Games, організований подібно до Олімпійських ігор. WCG були започатковані в 2000 році в Південній Кореї, і з того часу проводиться щороку, у тому числі і в інших країнах. У Південній Кореї кіберспорт отримав найбільший розвиток, отже там навіть існують телевізійні канали, які транслюють змагання з електронного спорту. Великі змагання проводяться в спеціальних місцях, де публіка може спостерігати за гравцями, що сидять за комп'ютерами, а хід змагань відстежувати на великому екрані, де транслюється ігровий процес. У Південній Кореї, через велике число глядачів, подібні змагання проводять на стадіонах[6].

Менш масштабні змагання проводяться в комп'ютерних клубах. Крім того, змагання можуть проводитися через інтернет. Гра через інтернет володіє деякими недоліками. У різних гравців можуть бути різні затримки передачі інформації через глобальну мережу в зв'язку з її неоднорідністю. Під час гри через інтернет складно виявити шахрайство партнерів. У контрасті, під час гри через локальну мережу всі гравці присутні в одному приміщенні під наглядом організаторів змагання, тому шахраювати

набагато важче. Локальна мережа зводить нанівець і проблему затримок, оскільки має достатню і однакову для всіх пропускну здатність. На великих змаганнях призовий фонд може сягати 1 000 000 доларів або більше. Найбільший виграш за історію кіберспортивних змагань виграли команда Na`Vi, яка перемогла у фіналі чемпіонату The international з дисципліни Dota 2, отримавши 1000000$. Призові фонди спонсорують компанії, пов'язані з виробництвом комплектуючих і периферії для комп'ютерів. Для кіберспорту підходять такі жанри відеоігор, як шутери від першої особи, стратегії реального часу і спортивні симулятори як найбільш видовищні і динамічні. Поточні популярні професійні відеоігри включають:Counter-Strike, FIFA, Halo 2, Heroes of Newerth, League of Legends, osu!, Point Blank, Quake, Starcraft, Unreal Tournament, Warcraft, World of Tanks.

1.5 Розробка комп'ютерних ігор

Розробка відеогри має низку послідовних етапів, загалом їх є три: розробка програмного (джерельного) коду, розробка контенту (малюнки, моделі, музика) та розробка ігрових механік. Їм передує проектування (пре-продакшну) — генерування геймдизайнером ідей щодо майбутньої гри, вибір жанру, тематики, особливостей ігрового процесу, розробка сценарію та образів персонажів з оточенням. Менеджер координує дії різних людей, залучених до розробки, складає план їхньої роботи, встановлює терміни її виконання, планує витрати.

Готова гра в свою чергу має пройти низку етапів, в ході яких потрапляє до гравців і підтримує інтерес до себе. Індустрія відеоігор включає у себе багато людей з різними професіями та ролями: програмістів, які відповідають за технічні можливості гри, художників, моделювальників та аніматорів, які створюють графічний контент, композиторів та звукорежисерів, які створюють звукове оформлення та музичний супровід, який нерідко видається окремим накладом. За успішне завершення роботи над проектом відповідають продюсери. Відеоігри, які розробляються незалежними розробниками чи аматорами називаються інді-іграми. Такі ігри нерідко створюються за допомогою спеціальних програм, які які істотно полегшують (або навіть ліквідують) процес розробки коду або графіки, наприклад, як RPG Maker.

Написання коду

Оскільки відеогра є комп'ютерною програмою, її робота, технічні можливості, контент та ігровий процес, забезпечується програмним кодом. Розробка гри включає ті ж етапи, що і розробка програмного забезпечення, але передбачає більше роботи над контентом і створення ігрових механік. Сучасні ігри здебільшого засновані на готових програмних модулях — ігрових рушіях, де вже реалізовані базові функції, здатні зв'язувати воєдино графіку, звук, об'єкти і їх рухи. Щоб налаштувати рушій для реалізації конкретного задуму програмісти доопрацьовують його, додаючи потрібні функції. Існують як вільні ігрові рушії, доступні будь-кому, так і ті, що вимагають отримання ліцензії на їх використання. Крім того рушії різняться за ліцензіями. Для незалежних розробників їх використання може бути значно дешевшим. Деякі рушії розраховані на створення ігор конкретного жанру, інші — універсальні. Не всі рушії можуть забезпечити однакові можливості та рівень графіки. Частина рушіїв дозволяють створювати ігри для різних платформ, так Unreal Development Kit підтримує розробку інтерактивних творів для PC, Xbox 360, PlayStation 3, Wii та Android. Деякі ігри

створюються в спеціальних програмах, які вже мають початкові ресурси, дії, та не вимагають знання мов програмування. Прикладами таких програм є Game Maker, Construct, RPG Maker.

Розробка контенту

Відеогра передбачає створення графіки, звуків та текстів. Концепт-арти виконуються на папері або комп'ютері, зазвичай в кількох варіантах. На основі концепт-артів художниками затверджуються і створюються моделі персонажів, предмети та декорації. Для цього художники працюють в програмах, призначених для роботи із графікою. Щоб моделі рухалися, вони анімуються в інших спеціальних програмах. Створюються різні набори рухів, які відтворюватимуться залежно від конкретних дій гравця з допомогою програмного коду. У випадку двовимірної графіки це набори спрайтів, де кожна картинка є окремими кадром. Для реалізації реалістичних рухів чи емоцій може застосовуватися захоплення руху живих акторів. Після фіксування руху датчиками вони переносяться на комп'ютерного персонажа, як людина або чудовисько. Серед них деякі додають реалістичності, як відкидання тіней, заломлення світла, постріли і вибухи. Інші позначають стани і дії персонажа, які визначають стиль виконання гри.

Деякі ігри цілком виконуються в стилі коміксу або кіноплівки. За реалізацію картинки і звуку відповідають графічний і звуковий рушії. Для звукового оформлення пишеться музика і відбувається озвучування персонажів. Крім того для повноцінного звукового оформлення потрібні ефекти, як кроки, звуки пострілів. Вони можуть обиратися з вільних бібліотек чи записуватися окремо. Декотрі композитори спеціалізуються на створенні музики до ігор. Музика може виконуватися цілими професійними оркестрами, мати пісенний супровід. Діалоги персонажів часто озвучуються спеціальними акторами на студіях озвучування. Часом ігри містять відеовставки, створені в програмах двовимірної чи тривимірної анімації.

Розробка ігрової механіки

Ігрова механіка визначає насиченість ігрового процесу, правила, за якими грається відеогра. Основою механіки є ігрові об'єкти, такі як персонажі, об'єкти, з якими вони можуть маніпулювати, декорації. Частиною ігрової механіки є управління, якими чином гравець керує персонажем та ігровим світом. Наприклад, як задається напрям руху, як активізується взаємодія з віртуальними предметами. Крім того на етапі розробки механіки створюється користувацький інтерфейс, який інформує гравця і дозволяє взаємодіяти зі світом гри[4].

В новітніх іграх світ часто моделюється так, щоб не мати чітких поділів на локації. Дизайнер рівнів розміщує готові об'єкти в ігровому світі та продумує їх рухи. Компонування рівнів визначає наскільки цікавою буде гра, які можливості будуть у гравця вирішити конкретну ситуацію. За взаємодію об'єктів, яка відбувається без контролю гравця, відповідає фізичний рушій. До прикладу, він реалізує закони інерції, гравітацію, поведінку рідин, властивості предметів. Штучний інтелект (ШІ) відповідає за поведінку персонажів, як вони реагуватимуть на дії гравця. Багато подій в грі відбуваються за скриптами. Самі події придумують сценаристи, а скрипти реалізують програмісти.

Тестування

Після завершення праці над кодом, контентом і механікою, за яких гра може функціонувати, відбувається її доопрацювання. Гра, не зібрана до кінця, але в яку можливо грати, називається альфа-версією. Вона може містити значні помилки і недопрацювання, як відсутність певних можливостей, музики або об'єктів. Виявленням проблем займаються тестери, котрі грають в цю гру, намагаючись сповна скористатися всіма доступними можливостями в ній. На пізнішому етапі виходить бета-версія, до тестування якої можуть залучатися і потенційні покупці гри. В бета-версії відбувається подальший пошук помилок, перевірка коректності взаємодії об'єктів ігрового світу, управління. Можливі внесення змін в оформлення, зміна ігрового балансу, тощо.

2. АНАЛІЗ ІНСТРУМЕНТІВ ТА СЕРЕДОВИЩА ДЛЯ РОЗРОБКИ

2.1 Дослідження існуючих інструментальних засобів

Більшість інструментальних засобів для розробки ігор можна поділити на 3 групи:

1. Фреймворк - програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. Фреймворк відрізняється від поняття бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підпрограм близькою функціональності, не впливаючи на архітектуру програмного продукту і не накладаючи на неї ніяких обмежень. У той час як фреймворк диктує правила побудови архітектури додатку на початковому етапі розробки поведінка за умовчанням, каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог.

2. Ігровий рушій - центральний програмний компонент комп'ютерних та відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі. Він забезпечує основні технології, спрощує розробку і часто дає грі можливість запускатися на декількох платформах, таких як ігрові консолі та настільні операційні системи, наприклад, GNU / Linux, Mac OS X і Microsoft Windows. Основну функціональність зазвичай забезпечує ігровий движок, що включає движок рендеринга («визуализатор»), фізичний движок, звук, систему скриптів, анімацію, штучний інтелект, мережевий код, управління пам'яттю і багатопоточність. Часто на процесі розробки можна заощадити за рахунок повторного використання одного ігрового движка для створення безлічі різних ігор[3].

3. Конструктор ігор - програма, яка об'єднує в собі ігровий рушій і інтегроване середовище розробки, і, як правило, включає в себе редактор рівнів, що працює за принципом WYSIWYG. Такі програми значно спрощує 25 процес розробки ігор, роблячи його доступним любителям-непрограмістам, і можуть бути використані в початковому навчанні програмуванню[1]. В роботі ми будемо досліджувати різноманітних представників всіх цих группи проте будемо приділяти більшу увагу на інструментам які мали реліз протягом останніх 5 років, оскільки світ мультимедіа та ігрової індустрії дуже швидко оновлюється та прогресує, то набір інструментів 5-ти річної давності виявится неактуальним в поточних реаліях .

2.2 Середовище для розробки гри

Unity — кросплатформний інструмент для розробки дво- та тривимірних застосунків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на консолях Wii, PlayStation 3 і XBox 360.Є можливість створювати інтернет-застосунки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player[2].

Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL. 30 Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація). Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти

(моделі), так і порожні ігрові об'єкти – тобто ті які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. У об'єктів з видимої геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить їх модель їх видимою. Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта.

Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів. При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна - буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріплюється до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, анімацію також можна створити попередньо в 3D редакторі та імпортувати разом з моделлю, а потім розбити на файли. У Unity вбудована підтримка мережі.Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі. Має вбудований генератор ландшафтів[4].

Переваги Unity: IDE. Поєднання редактора сцен (в комплексі загального редактора) з редактором ігрових об'єктів і редакторів скриптів. Додатково додаються генератори дерев і террейнов. Покращені можливості скриптинга, а саме на відміну від вищевказаного движка, в Unity доступні три мови: JavaScript, C # і різновид Python's Boo; кросплатформеність як уже згадувалося вище, підтримуються – Windows, Mac OS, Wii, iPhone, iPod, iPad, Android, PS3 і XBox), на смартфонах і планшетах (iOS, Android, Windows Phone), на 360, не всі з яких, звичайно, доступні у безкоштовній ліцензії. Ну і веб-плагін, звичайно, забувати не варто. Сучасний рівень графіки, здатний конкурувати з іншими двигунами. Unity, безумовно, програє Unreal Engine за кількістю реалізованих можливостей. Однак Unity володіє такими можливостями, як deferred 25 освітлення, стандартний набір постпроцессингових ефектів, SSAO. Гідним чином опрацьоване фізичний движок. Масштабованість і продуктивність. Більшу частину простих процесів движок обробляє на чудовому рівні. Запуск будь-якої програми на Unity в веб-плагін. Невисока ціна за повну ліцензійну версію для великого веб-розробника.

Недоліки Unity: закритість коду. Неможливість отримання вихідних кодів движка навіть за ліцензією; неможливість доповнення фізики движка сторонніми можливостями. Ви не зможете додати в движок сторонню фізику, або SpeedTree. Реальні мінуси складно визначити з першого погляду. Движок продуктивний, стабільний і легкий в застосуванні. У більшості нечисленних команд розробників комп'ютерних ігор основною проблемою часто ставав саме движок. Досить складно писати з нуля єдиному програмісту в команді. Потрібен повноцінний безкоштовний движок, і потрібен він відразу, програміст починає шукати безкоштовні вирішенні (Ogre, Irrlicht). Ці двигуни не так вже й погані (Torchlight написаний на Ogre), але вони складні в освоєнні і вимагають не одного програміста, а цілої команди. Звичайно, можна звернутися до наборів типу GameMaker, але серйозну гру з його допомогою зібрати складно. Що стосується Unity, то в його випадку є вже завершений пайплайн, готовий рендерер, зібрані фізику, аудіо та мережеве взаємодія, багатомовність.

Праворуч розташовані інспектори префабів (заготовки об'єктів), зліва – це об'єкти, розташовані в поточній схемі). У процесі перегляду рівня можна зупинити його, і переглянути поточний стан об'єктів.

3. РОЗРОБКА СЦЕНИ В СЕРЕДОВИЩІ UNITY 3D

3.1 Збірка сцен

Gameobjects і Components. Ігрові об'єкти (GameObjects) - це найважливіші об'єкти в Unity. Дуже важливо розуміти, що таке GameObject і як його використовувати. Кожен об'єкт в грі - це GameObject. Однак, GameObject нічого не роблять самі по собі. Вони вимагають спеціальної настройки, перш ніж стати персонажами, предметами оточенням або спеціальними ефектами. GameObject є контейнерами. Порожня коробка, яка може містити всередині різні елементи, такі як острів з запеченими тінями або фізично коректний автомобіль. Щоб дійсно зрозуміти GameObject, потрібно зрозуміти їх складові, який називаються компонентами (Components). Залежно від того, який ми хочемо створити об'єкт, ми будемо додавати різні комбінації компонентів до GameObject. Відкриймо будь-яку сцену Unity, створімо новий GameObject (використовуючи Shift-Control-N в Windows або Shift-Command-N в Mac), виберемо його і поглянемо в інспектор (Inspector)[5].

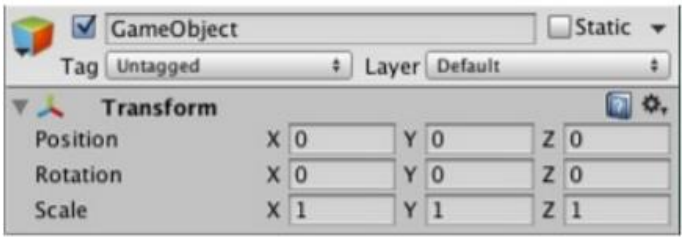

Рисунок 3.1 - Інспектор порожнього GameObject

Звернімо увагу, що навіть порожні GameObject мають ім'я, тег (Tag), і шар (Layer). Кожен GameObject також містить компонент Transform. У Unity неможливо створити GameObject без компонента Transform. Компонент Transform - один з найважливіших компонентів, так як всі властивості GameObject пов'язані з трансформаціями використовують цей компонент. Він визначає положення, обертання і масштаб GameObject в ігровому світі / вікні Scene. Якщо GameObject не матиме компонента Transform, він буде не більше ніж деякою інформацією в пам'яті комп'ютера. Він не зможе ефективно існувати в ігровому світі. 37 Компонент Transform важливий для всіх GameObject, тому він є у кожного GameObject. Але GameObject можуть містити й інші компоненти.

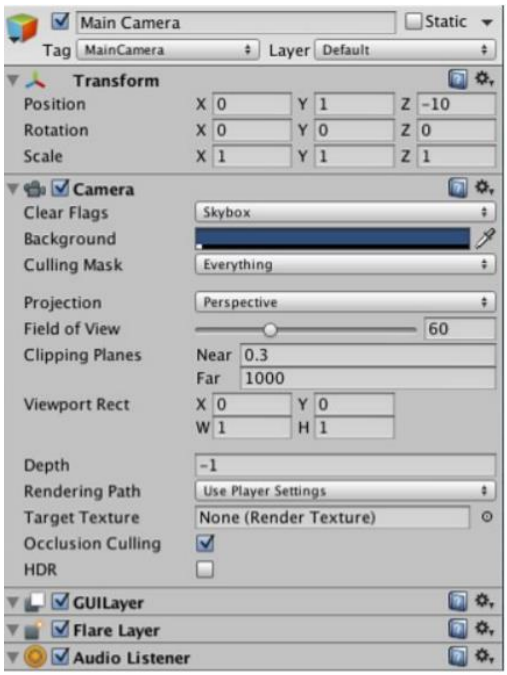

Рисунок 3.2 - GameObject під назвою Main Camera, який додається в кожну нову сцену за замовчуванням

Поглянувши на GameObject Main Camera, ми можемо побачити, що він містить різні колекції компонентів. Особливо, компонент Camera, GUI Layer, Flare Layer, і Audio Listener. Всі ці компоненти надають додаткову функціональність GameObject. Без них не було б кому займатися рендерингом ігрової графіки для граючої людини! Тверді тіла, колайдери, частинки, аудіо - це все різні компоненти (або комбінації компонентів), які можуть бути додані до будь-якого GameObject. Однією з чудових особливостей компонентів є гнучкість. При підключенні компонента до ігрового об'єкту, існують різні значення або властивості (Properties) в компоненті, які можуть змінюватися в редакторі при створенні гри або через скрипти в запущеній грі. Є два основних типи властивостей: значення (Values) і посилання (References). 38 Подивимося на рис 4.3., це порожній Ігровий Об'єкт з компонентом Audio Source. Всі параметри компонента Audio Source в Інспектора виставлені за замовчуванням.

Рисунок 3.3 - порожній Ігровий Об'єкт з компонентом Audio Source

Компонент містить одну властивість-посилання і сім властивостей значень. Audio Clip - це властивість-посилання. Коли це аудіо джерело починає грати, він буде намагатися програти файл, на який посилається властивість Audio Clip. Якщо такого посилання не виявиться, то виникне помилка, так як ніяке аудіо НЕ буде програно. Ми повинні призначити файл в інспектор. Всі інші властивості після Audio Clip це властивості-значення. Вони можуть бути відрегульовані прямо в інспектор. Властивості значення після Audio Clip - це все перемикачі, числові значення і випадаюче меню, але властивості-значення можуть також бути текстовими рядками, кольорами, кривими і іншими типами.

3.2 Використання вікна

Scene View Вікно Scene View - це інтерактивна пісочниця. Будемо використовувати Scene View для вибору і розташування оточень, гравця, камери, ворогів, і всіх інших ігрових об'єктів. Управління та маніпулювання об'єктами з використанням вікна Scene View є однією з найбільш важливих функцій Unity, тому важливо вміти робити це швидко[6].

Вікно Scene має набір навігаційних елементів управління, які допомогають орієнтуватися в ньому швидко і ефективно. Для переміщення по сцені можна використовувати клавіші зі стрілками. Клавіші вгору і вниз переміщують камеру вперед і назад в тому напрямку, в яке вона дивиться. Клавіші вліво або вправо повертають камеру в сторони. При використанні стрілок, затиснимо клавішу Shift для прискореного переміщення. Якщо вибрати гейм об'єкт в ієрархії, потім помістити миш над Scene View, і натиснути Shift + F, вид концентрується на вибраному об'єкті. Ця можливість називається кадрувати виділене (frame selection).

Рисунок 3.4 - Елементи фокусування

Переміщення, обертання по орбіті і масштабування - ключові операції навігації у вікні Scene View, тому Unity надає кілька альтернативних шляхів їх виконання для максимальної зручності.

Позиціонування ігрових об'єктів

Під час створення гри, я повинен розмістити багато різних об'єктів у ігровому світі. Для цього будуть використані інструменти трансформацій в панелі інструментів для переміщення, обертання і масштабування окремих об'єктів. Кожен інструмент має відповідне Гизмо, яке з'являється навколо виділеного ігрового об'єкта у вікні Scene. Можна використовувати миш і маніпулювати будь якою віссю Гизмо для зміни компонента Transform ігрового об'єкта, або можна ввести значення безпосередньо в числові поля компонента в інспекторі. Кожен з трьох режимів може бути обраний гарячими клавішами - W для переміщення, E для обертання і R для масштабування.

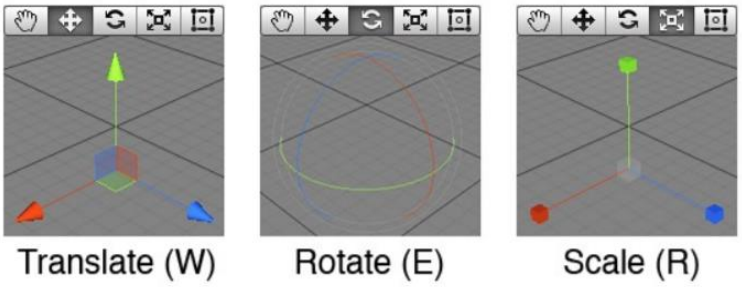

Рисунок 3.5 - Переміщення, обертання і масштабування

Пошук в Scene

При роботі з великими і складними сценами буде корисний пошук певних об'єктів. Використовуючи можливість Search в Unity, можна відфільтрувати об'єкт або групу об'єктів. Можна шукати Ассет по імені, по типу компонента, і, в деяких випадках, по мітках Ассет. Можна вибрати режим пошуку з випадаючого меню Search. У вікнах Scene і Hierarchy є поле пошуку, що дозволяє фільтрувати об'єкти по імені. Так як ці вікна, по суті, просто два варіанти відображення одних і тих же об'єктів, будь-якій пошуковій запит буде дублюватися в обох полях пошуку і застосовуватися однаково до обох вікон. Звернемо увагу, що коли пошук активний, обидва вікна трохи відрізняються: вікно Scene буде показувати об'єкти, які не пройшли по фільтру сірим, а у вікні Hierarchy будуть відображатися тільки ті об'єкти, імена яких відповідають пошуковому запиту.

Невеликий хрестик праворуч від поля пошуку видаляє пошукової запит і повертає вікно до нормального стану. В меню зліва від поля можна вибрати фільтр пошуку - по імені об'єкта, за його типу, або обом цим параметрам.

Префаб (Prefabs)

Досить зручно працювати з GameObject в сцені, додаючи компоненти і змінюючи їх значення на потрібні у інспекторі. Однак, це може створити ряд проблем в таких

випадках, коли працюємо над створенням NPC, об'єктом або предметом, який часто зустрічається в сцені. Звичайно, можна просто скопіювати ці об'єкти для створення дублікатів, але всі вони будуть редагуватися незалежно один від одного. В Unity можна створювати префаб. Це особливий тип Ассета, що дозволяє зберігати весь GameObject з усіма компонентами і значеннями властивостей. Префаб виступає в ролі шаблону для створення екземплярів зберігаючого об'єкта в сцені. Будь-які зміни в префабі негайно відображаються і на всіх його примірниках, при цьому можна перевизначати компоненти і настройки для кожного екземпляру окремо. Коли перетягуємо файловий Ассет (наприклад, меш) в сцену, буде створений новий екземпляр такого об'єкта і всі такі екземпляри зміняться при зміні оригінального Ассета. Однак, хоч його поведінка і схожа, але Ассет - це не префаб, так що не вийде додати до нього компоненти або використовувати будь-які інші описані нижче властивості префабов. Можна створити префаб, вибравши Asset> Create Prefab і перетягнувши об'єкт зі сцени в "Blender" порожній"Blender" префаб, що з'явився в проекті. Після чого можна створювати екземпляри префаба просто перетягуючи його з вікна Project на сцену. Імена об'єктів реалізованих префабом, будуть підсвічуватися синім у вікні Hierarchy (імена звичайних об'єктів мають чорний колір). Як уже згадувалося вище, зміни в префаб автоматично застосуються до всіх її екземплярів, однак можна змінювати і окремі екземпляри. Це корисно наприклад в разі, коли бажаємо створити кілька схожих NPC, але з зовнішніми відмінностями, щоб додати реалістичності. Щоб було чітко видно, що властивість в екземплярі префаб змінено, воно показується в інспекторі жирним шрифтом (якщо до примірника префаба доданий абсолютно новий компонент, то всі його властивості будуть написані жирним шрифтом).

Джерела світла

Джерела світла є невід'ємною частиною кожної сцени. У той час як меши і текстури визначають форму і зовнішній вигляд сцени, джерела світла визначають колір і атмосферу вашого 3D оточення. Організація їх одночасної роботи вимагає невеликої практики, але результат може бути приголомшливим.

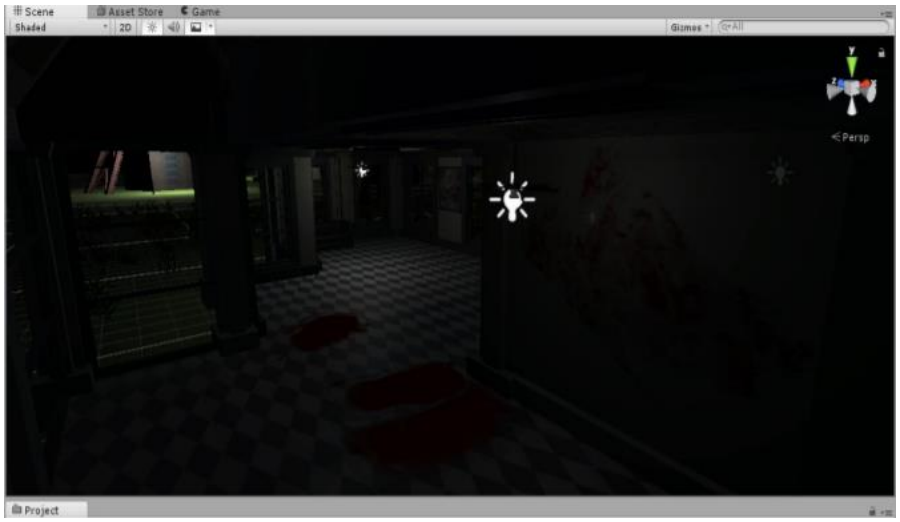

Рисунок 3.7 - Проста компоновка двох джерел світла

Джерела світла можуть бути додані в сцену з використанням меню GameObject-> Create Other. Після того, як джерело світла буде додано до сцени, можна керувати ним як будь-яким іншим GameObject. Також, можна додати компонент Light до будь-якого виділеного об'єкту використовуючи Component-> Rendering-> Light. Компонент Light має багато різних властивостей, які можна змінювати в інспектора (Inspector).

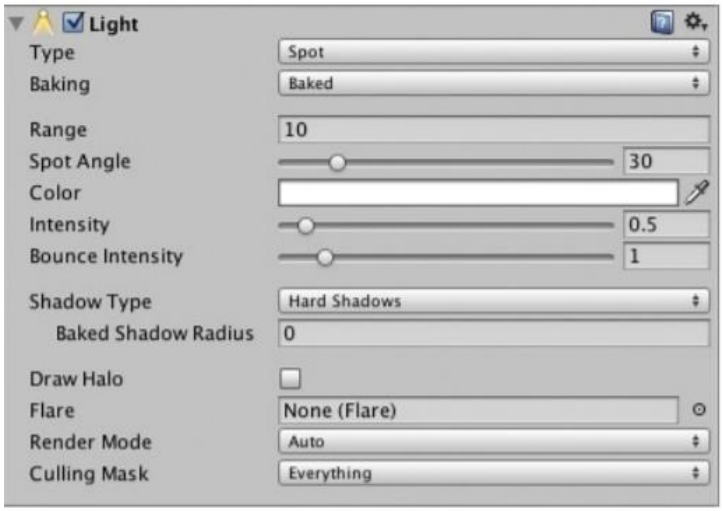

Рисунок 3.8 - Властивості компонента Light в інспектора

Просто змінюючи властивість Color (колір) джерела світла, можна додати сцені зовсім інший настрій.

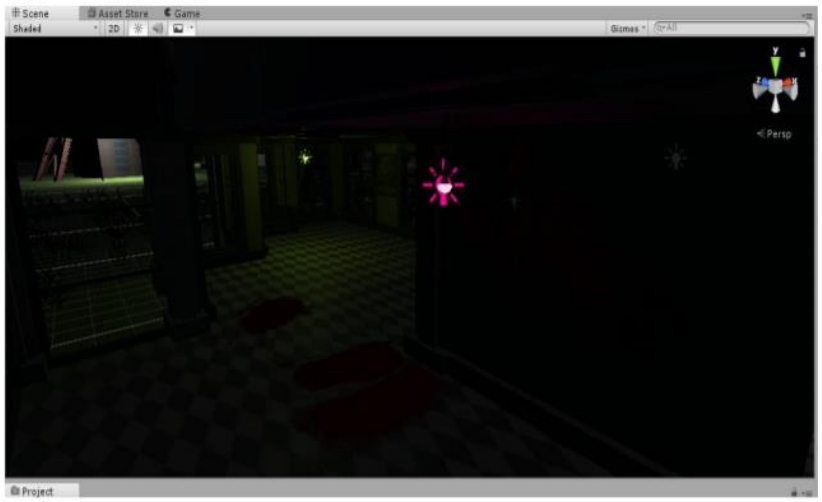

Рисунок 3.9 - Похмурі джерела світла

3.3 Реалізація прототипу на Unity

Unity studio – це окрема середа розробки з набором вбудованих редакторів та інструментів для відладки гри. Є можливість запустити гру а самій студії, слідкувати за змінними , емулювати дії гравця, тощо.

Робота з графічними ресурсами

Щоб додати графічні ресурси варто лише приєднати їх до проекту у спеціальному менеджері проекту. Студія автоматично розцінює графічні файли як текстури, проте якщо це набір спрайтів потрібно лише змінити властивість об‘єкту на спрайт. Також присутній редактор спрайтів, де можна нарізати спрайти автоматично, чи вручну, задати центр для кожного спрайту, тощо.

Робота з фізикою та колізією

В Unity э декілька вбудованих рушіїв для 2D та 3D та декілька способів взаємодії з ними. Перш за все потрібно додати компонент Collider до об‘єкту гри, тип Collider буде визначати тип рушія, для прототипу використаємо Physics 2D Collider. Цей колайдер буде використовуватись для визначення претинута доторкання об‘єктів з іншими колайдерами, додаткові налаштування triggers дозволяє обмету проходити через інші об‘єкти проте так само викликати подію накладання, напроти Cinematic objects буде зупиняти об‘єкти що доторкаються до нього , проте не будуть викликати подію накладання якщо будуть рухатись самі[1].

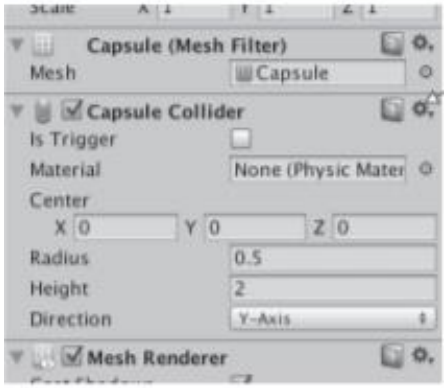

Рис 3.10 Налаштування Collider в Unity

Ще однією важливою деталлю налаштування колізії є рівні(Layers) можна назначити певним об‘єктам певні рівні. Об’єкти між різними рівнями не взаємодіють, до того ж можна налаштувати колізію між об‘єктами на одному рівні. Таким чином для нашого прототипу буде доцільно вимкнути взаємодію куль між собою, кулі та гравця та ворогів один з одним.

Робота з логікою гри

Робота з логікою гри проходить за допомогою написання скриптів та додаванню ції скриптів до сцени чи обраного об‘єкту. Головна концепція скриптів це використання певних явищ чи методів з періодичним викликом. Таким чином цей скрипт можна приєднати до декількох об‘єктів та декілька скриптів до одного. Скажімо скрипт ,що під час кожного виклику методу Update перевіряє чи не знизився показник здоров‘я цілі до 0, і якщо так то знищує його. Такий скрипт можна додати як до об‘єкту гравця так і до об‘єктів ворогів і навіть перешкод що можуть руйнуватись. Скрипти пишуться на мові C# в MS Visual Studio з якою інтегрується проект створений в Unity studio. Таким чином стає доступним весь арсенал відладки Visual Studio при роботі з грою.

Організація взаємодії з гравцем

В Unity існує два методи, призначення яких оновлювати дані, Fixedupdate та Update. Різниця в тому, що Fixedupdate викликається зі сталою періодичністю в приблизно 2 мілісекунди і використовується для обробки фізики, зміни швидкості, тощо. Тоді як Update викликається кожного разу перед оновленням фрейму, іноді період цього виклику більше 2 мілісекунд іноді менше. Для обробки сигналів від гравця використовується саме методі Update, адже він після отримання сигналу. В цьому методі ми маємо доступ до об‘єкту input, з його допомогою дізнаємося про стан клавіатури та очікувати натискання якоїсь певної клавіші. Або натомість використати більш гнучке

рішення звернувшись до властивості input під назвою Axes – своєрідний рівень абстракції до якого можна прив‘язати логіку гри. Таким чином незалежно від того, буде використовуватися клавіатура, джойстик чи навіть тачскрін як пристрій вводу, гра буде коректно працювати. Настроїти об‘єкти Axes можна в спеціальному менеджері, задати ім‘я, та сигнали за замовчуванням[3].

Робота з логікою гри та взаємодія з гравцем

Вся логіка гри зібрана в методі update що викликається декілька разів за секунду. Обробка реакції гравця також реалізується в цьому методі. Використовуючи методи об‘єкту фізики ми можемо легко перевірити чи зіштовхнулись, наклались 2 об’єкта, задати їм швидкість тощо. Також ми маємо обробити реакцію на натискання клавіш переміщення, пострілу та реакцію камери на переміщення гравця. Для обробки складних дій , аби зберігати читабельність коду, доцільно виносити їх в окремі функції та викликати в методі update або навіть створювати цілі класи для обробки складної логіки, виносити їх в окремі файли і підключити в html документі на початку.

Скрипти з відповідними поясненнями

SceneManagerr.cs

Реалізує логіку переключення між сценами

```
public class SceneManagerr : MonoBehaviour
{
    public void LoadNextScene()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void LoadScene(int index)
    {
        SceneManager.LoadScene(index);
    }

    public void Quit()
    {
        Application.Quit();
    }
}
```

MultGameManager.cs

Скрипт, відповідаючий за логіку гри “Множення”.

```
public class MultGameManager : MonoBehaviour
{
    public int rightAttempts=0;
    public int falseAttempts=0;
    public float timerVal = 60;

    public int firstNum = 1;
    public int secondNum = 1;
    public int result=0;
    public int guessedResult = 0;

    public InputField resultInputField;

    private MultUIManager uIManager;

    void Start()
    {
        uIManager = FindObjectOfType<MultUIManager>();
        GenerateNewTask();
    }
    void Update()
    {
        timerVal -= Time.deltaTime;

        if (Input.GetKeyDown(KeyCode.Return))
        {
            SubmitResult();
            resultInputField.text = "";
            resultInputField.Select();
            resultInputField.ActivateInputField();
        }

        if (timerVal <= 0)
        {
```

```
            uIManager.GameOver();
        }
    }

    private void GenerateNewTask(){
        firstNum = Random.Range(0, 11);
        secondNum = Random.Range(0, 11);
        result = firstNum * secondNum;
    }

    private void SubmitResult()
    {
        guessedResult = System.Convert.ToInt32(resultInputField.text);

        if (guessedResult == result){
            rightAttempts++;
        }
        else{
            falseAttempts++;
        }
        GenerateNewTask();
    }
}
```

MultUIManager.cs

Скрипт, відповідіючий за логіку UI компонентів гри “Множення”.

```
using UnityEngine.UI;

public class MultUIManager : MonoBehaviour
{
    public Text rightAttemptsText;
    public Text falseAttemptsText;
    public Text timerText;

    public Text firstNumText;
    public Text secondNumText;


    public GameObject mainCanvas;
    public GameObject gameOverCanvas;

    private MultGameManager gameManager;

    public void Start(){
        gameManager = FindObjectOfType<MultGameManager>();
    }

    public void Update(){
        rightAttemptsText.text = "Вірно: " + gameManager.rightAttempts.ToString();
        falseAttemptsText.text = "Невірно: " + gameManager.falseAttempts.ToString();
        timerText.text = "Час: " + ((int)gameManager.timerVal).ToString();

        firstNumText.text = gameManager.firstNum.ToString();
        secondNumText.text = gameManager.secondNum.ToString();
    }

    public void GameOver(){
        mainCanvas.SetActive(false);
        gameOverCanvas.SetActive(true);
    }}
```

TypingGameManager.cs

Скрипт, відповідаючий за логіку гри “Швидкість набору”.

```
using UnityEngine;

public class TypingGameManager : MonoBehaviour
{
    public int rightAttempts = 0;
    public float timerVal = 60;

    public List<string> words;

    public string buffer = "";

    void Start(){
        GenerateNewWord();
    }

    void Update(){
        timerVal -= Time.deltaTime;
        PrecessInput();

        if (timerVal <= 0)
        {
            FindObjectOfType<TypingUIManager>().GameOver();
        }
    }

    private void PrecessInput(){
        foreach (char ch in Input.inputString)
        {
            if (ch == buffer[0]){
                buffer = buffer.Remove(0, 1);
            }

            if (buffer.Equals("")){
                GenerateNewWord();
                rightAttempts++;
            }}
       }

    private void GenerateNewWord(){
        buffer = words[Random.Range(0, words.Count)];
    }
}
```

TypeUIManager.cs

Скрипт, відповідіючий за логіку UI компонентів гри “Швидкість набору”.

```
using UnityEngine.UI;

public class TypingUIManager : MonoBehaviour{

    public Text rightAttemptsText;
    public Text bufferText;
    public Text timerText;

    private TypingGameManager gameManager;

    public GameObject mainCanvas;
    public GameObject gameOverCanvas;
```

```
    void Start(){
        gameManager = FindObjectOfType<TypingGameManager>();
    }

    void Update(){
        rightAttemptsText.text = "Вірно: " + gameManager.rightAttempts.ToString();
        bufferText.text = gameManager.buffer;
        timerText.text = "Час: " + ((int)gameManager.timerVal).ToString();
    }

    public void GameOver(){
        mainCanvas.SetActive(false);
        gameOverCanvas.SetActive(true);
    }
}
```

ClockGameManager.cs

Скрипт, відповідаючий за логіку гри “Вчимося користуватися годинником”.

```
public class MyTime
{
    public int Hour { get; set; }
    public int Minute { get; set; }

    public MyTime() { }

    public MyTime SetHour(int hour)
    {
        Hour = hour;
        return this;
    }

    public MyTime SetMinute(int minute)
    {
        Minute = minute;
        return this;
    }

    public override string ToString()
    {
        return Hour.ToString("00") + ":" + Minute.ToString("00");
    }
}

public class ClockGameManager : MonoBehaviour
{
    public static int MINUTES_ON_CLOCK = 60;
    public static int HOURS_ON_CLOCK = 12;

    public Transform minutesHand;
    public Transform hoursHand;

    public int rightAttempts = 0;
    public float timerVal = 60;
    private bool checkOneTime = true;

    private MyTime timeToGet, currentTime;
    private ClockUIManager uIManager;

    void Start()
    {
        uIManager = FindObjectOfType<ClockUIManager>();
```

```
        currentTime = new
MyTime().SetHour(GetClockByAngle(System.Math.Abs(hoursHand.rotation.eulerAngles.z -
360), HOURS_ON_CLOCK))

.SetMinute(GetClockByAngle(System.Math.Abs(minutesHand.rotation.eulerAngles.z - 360),
MINUTES_ON_CLOCK));
        GenerateNewTask();
    }

    public void GenerateNewTask()
    {
        timeToGet = GenerateTime();
        uIManager.SetTimeText(timeToGet);
        uIManager.SetNextButtonActive(false);
        checkOneTime = true;
    }

    void Update()
    {
        timerVal -= Time.deltaTime;


currentTime.SetHour(GetClockByAngle(System.Math.Abs(hoursHand.rotation.eulerAngles.z -
360), HOURS_ON_CLOCK))

.SetMinute(GetClockByAngle(System.Math.Abs(minutesHand.rotation.eulerAngles.z - 360),
MINUTES_ON_CLOCK));

        print(currentTime);

        if (currentTime.Hour == timeToGet.Hour && currentTime.Minute ==
timeToGet.Minute &&checkOneTime)
        {
            uIManager.SetNextButtonActive(true);
            checkOneTime = false;
            rightAttempts++;
        }

        if (timerVal <= 0) {
            uIManager.GameOver();
        }
    }

    private MyTime GenerateTime()
    {
        return new MyTime().SetHour(Random.Range(0, HOURS_ON_CLOCK +
1)).SetMinute(Random.Range(0, MINUTES_ON_CLOCK));
    }

    public int GetClockByAngle(float angle, int bas)
    {
        return Mathf.FloorToInt(GetPercent(angle, 360) * bas);
    }

    public float GetPercent(float amount, float from)
    {
        return amount / from;
    }
}
```

ClockUIManager.cs

Скрипт, відповідіючий за логіку UI компонентів гри “Вчимося користуватися годинником”.

```
using UnityEngine.UI;

public class ClockUIManager : MonoBehaviour
{
    public Text timeToGetText;
    public GameObject nextBtn;

    public Text rightAttemptsText;
    public Text timerText;

    public GameObject mainCanvas;
    public GameObject gameOverCanvas;

    private ClockGameManager gameManager;
    void Start(){
        gameManager = FindObjectOfType<ClockGameManager>();
    }
    void Update(){
        rightAttemptsText.text = "Рахунок: " + gameManager.rightAttempts.ToString();
        timerText.text = "Час: " + ((int)gameManager.timerVal).ToString();
    }

    public void SetTimeText(MyTime time){
        timeToGetText.text = time.Hour.ToString("00");
        timeToGetText.text += ":"+time.Minute.ToString("00");
    }

    public void SetNextButtonActive(bool state)
    {
        nextBtn.SetActive(state);
    }

    public void GameOver()
    {
        mainCanvas.SetActive(false);
        gameOverCanvas.SetActive(true);
    }}
```

Скрипт, відповідаючий за обертання стрілок годинника

```
public class ClockHand : MonoBehaviour {
    public KeyCode keyCode = KeyCode.H;

    void Update()
    {
        if (Input.GetKey(keyCode))
        {
            var mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

            Vector2 direction = new Vector2(
                mousePosition.x * 150 - transform.position.x,
                mousePosition.y * 150 - transform.position.y
                );
            transform.up = direction;
        }
    }}
```

4. СТРУКТУРА ГРИ

Гра запускається зі стартового(головного) меню (рис. 4.1).

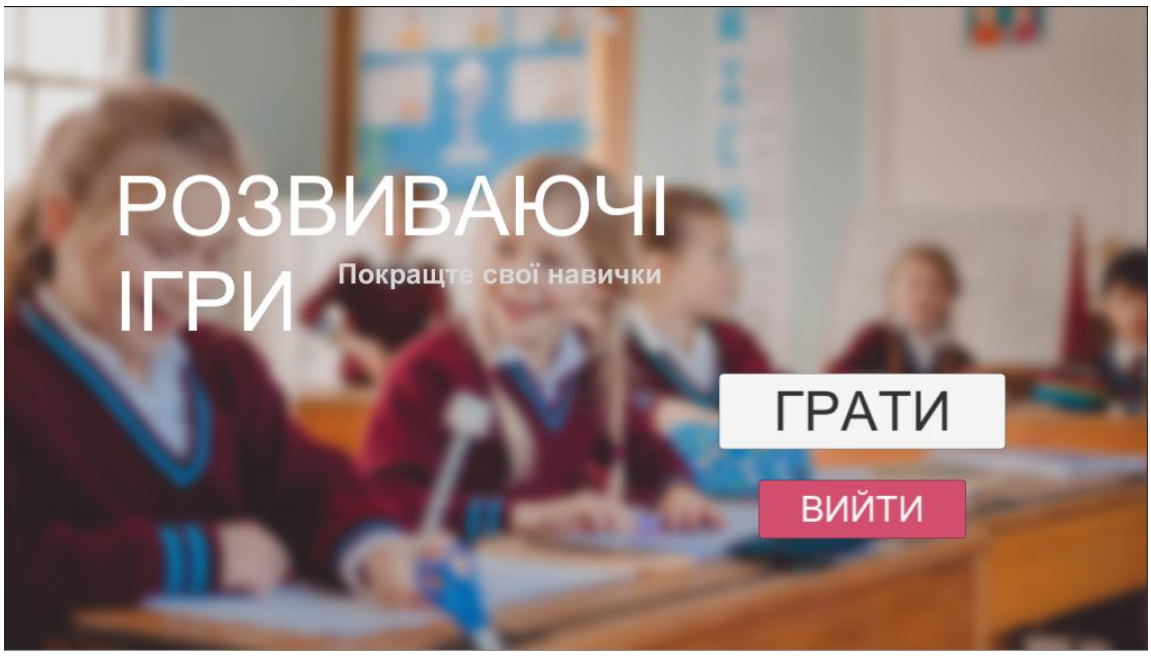

Рисунок 4.1 – Стартове меню

З початкового вікна можна почати гру або вийти з гри.

При виборі пункту меню «Грати» ми попадаємо в меню з іграми(рис. 4.2).

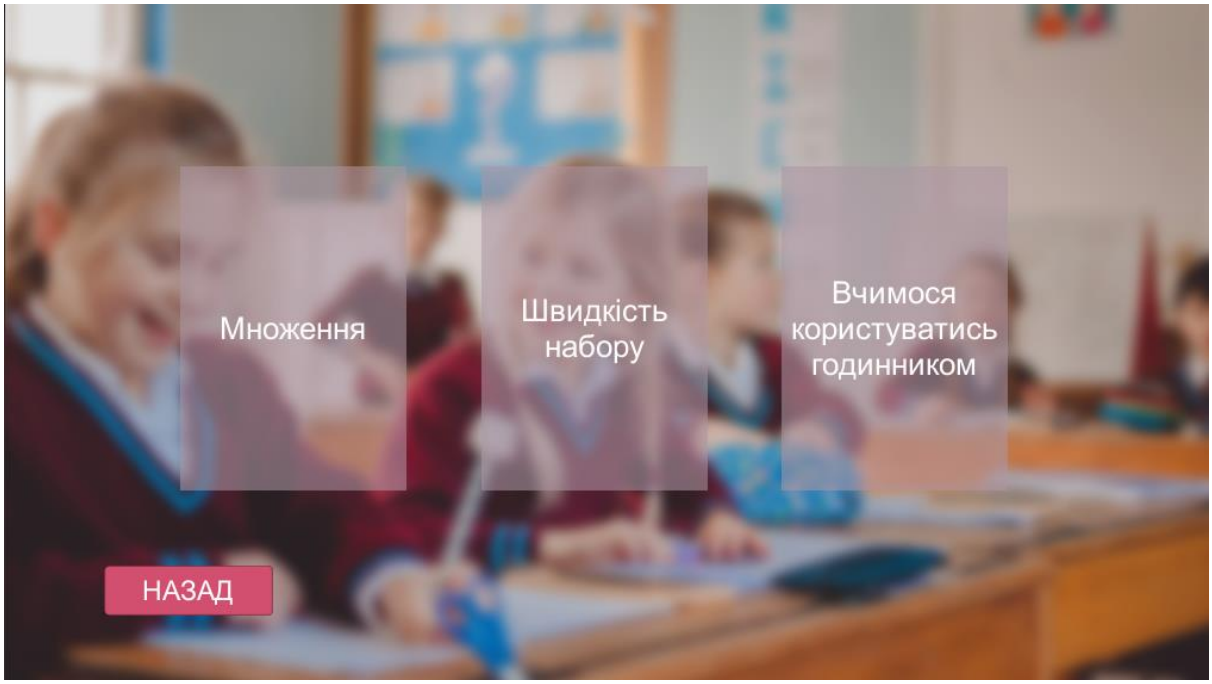

Рисунок 4.2 – Меню ігор

Вибравши бажану гру ви попадете на сцену, що відповідає за цю гру .

Ви можете попасти в одну з трьох таких ігор:

В цій грі можна навчитись множити відповідаючи на поставлену задачу. Кількість правильних відповідей записується та гра триває допоки таймер не дійде до 0.

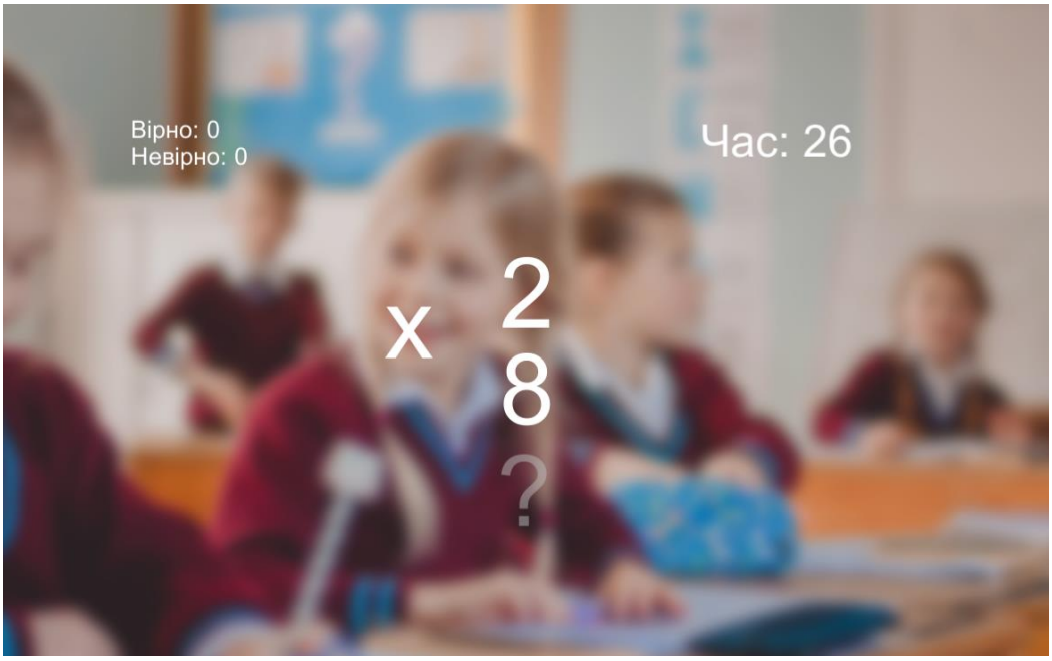

Рисунок 4.3 – Гра “Множення”

В наступній грі ви тренуєте швидкість свого набору за рахунок введення різних слів. Кількість правильних відповідей записується та гра триває допоки таймер не дійде до 0.

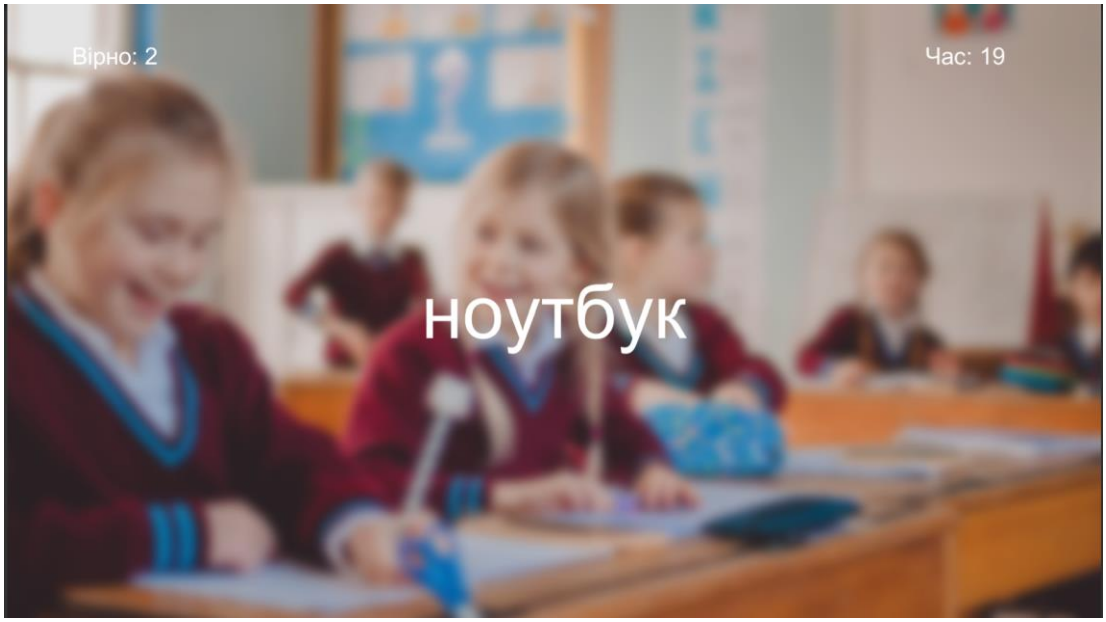

Рисунок 4.4 – Гра “Швидкість набору”

В останній грі ви навчитесь налаштовувати годинник, вивчаючи те, як він працює, порівнюючи його із цифровим аналогом.

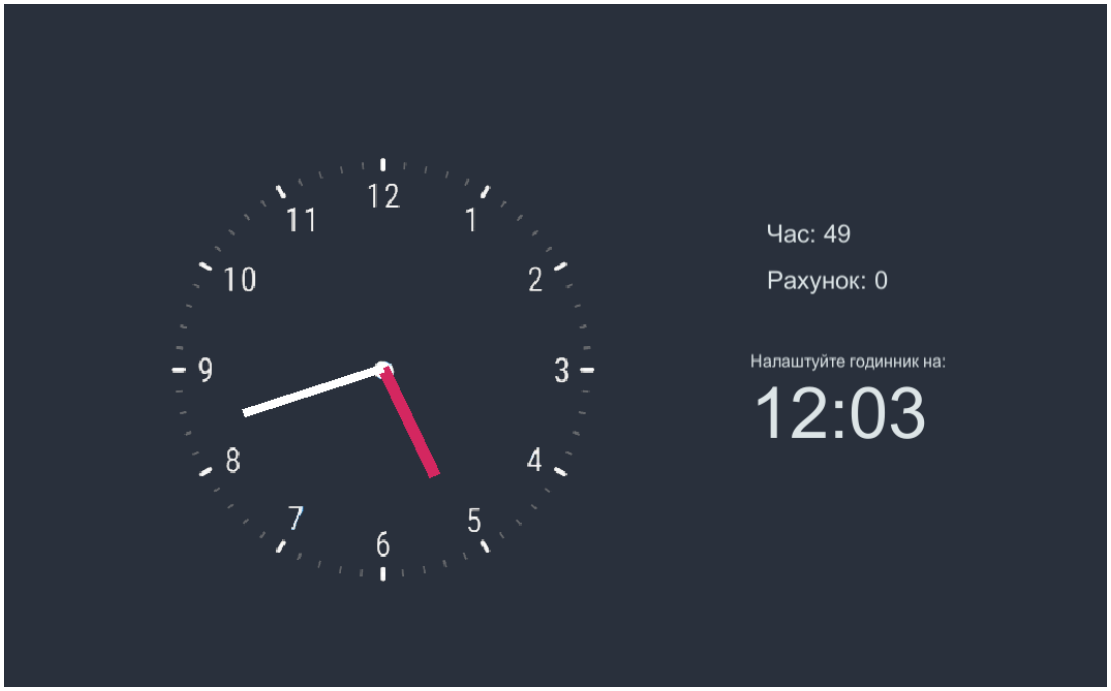

Рисунок 4.5 – Гра “Вчимося користуватися годинником”

При завершенні часу, що виділяється на гру, на екрані появиться наступний текст з кнопкою, що позволить повернутись у меню.

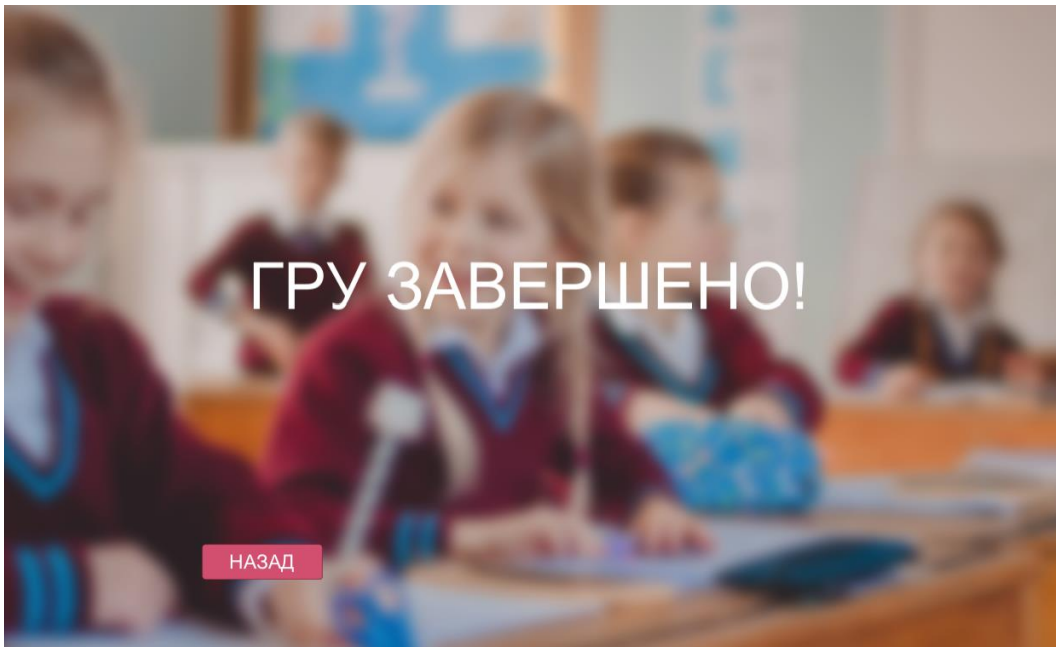

Рисунок 4.6 – Зображення на екрані при завершенні гри

ВИСНОВКИ

Як видно з зазначених матеріалів за останні роки відеоігри зайняли чималу нішу на ринку розваг та дозвілля. А в деяких країнах навіть досягли статусу спорту та мистецтва. Компаній з розробки ігор стає дедалі більше. До того ж з'явилося таке поняття як інді-розробник та інді-студія – незалежні розробники що власноруч, без підтримки великих компаній, створюють свої ігри. Як наслідок все більше людей починають захоплюватись розРобкою відеоігор. В свою чергу з'явився чималий попит на інструментальні засоби що полегшують розробку, покращують графіку, оптимізують фізику тощо. Проте різні засоби мають специфічні сфери використання залежно від типу, жанру, платформи тощо. А отже потреба в їх аналізі та класифікації виглядає досить актуальною, завдяки моїм дослідженням розробник зможе зберегти час та більш доцільно обрати потрібні йому інструментальні засоби.

Unity – ігровий рушій, зібравший в собі фізичний, графічний рушій, великий набір інструментів та широкий функціонал. Гарний баланс міл кількістю написаного коду та використання вбудованого інструментарію для налаштування елементів гри. Зручна візуалізація продукту та чудові можливості відладки як коду так і поведінки гри в цілому. Великі можливості для повторного використання коду та вбудована оптимізація графіки, фізики. Unity виявився найпотужнішим засобом для розробки гри серед розглянутих, використаний для аналізу прототип виглядав надто легким для використання представленного функціоналу. Найкраще Unity покаже себе в розробці великих та середніх проектів командою чи поодиноким розробником.

Була створена навчальна гра "DevGames" для реалізації якої виконано аналіз середовища Unity. За допомогою цього середовища було створено об'єкти власноруч для використання їх у проекті. Результатом розробки стала гра на основі Unity 2D, яка містить різні ігри для навчання та саморозвитку гравців.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Большаков Д. І. 3D моделювання. / Д. І. Большаков - Техатека, 2011. - 34 с.

2. Бочков М. Д. Основи 3D-моделювання. / М. Д. Бочков - СПб. : Пітер, 2003. - 106 с.

3. Гембейк Е. В. Ігрова індустрія. Створення ігор. / Е. В. Гембейк - М. : 3DNs, 2007. - 412 с.

4. Дацко М. А. Моделювання складних об'єктів. / М. А. Дацко - М. : Максімас, 2015. - 111 с.

5. Хокінг Д. М. Unity в дії. Мультиплатформенна розробка на. / Д. М. Хокінг, 2016. - 336 с.