НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи

на ступінь вищої освіти бакалавр
на тему: «**РОЗРОБКА ГРИ «SHOOT AND RUN» ЖАНР ШУТЕР МОВОЮ C#»**

Виконав: студент 4 курсу, групи ПД– 43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Косенко А.П.
(прізвище та ініціали)
Керівник Дібрівний О.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення
О.В. Негоденко

“_____” _____2022 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Косенку Андрію Павловичу

(прізвище, ім‘я, по батькові)

1. Тема роботи: «Розробка гри «Shoot and run» жанр шутер мовою C#»
Керівник роботи доктор філософії Дібрівний О.А.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “18” лютого 2022 року № .
2. Строк подання студентом роботи 03.06.2022.
3. Вихідні дані до роботи:
   3.1. Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо розробки ігор;
   3.2. Офіційна документація Unity;
   3.3. Офіційна документація Rider;
   3.4. Науково-технічна література;
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Аналіз предметної області
   4.2. Огляд та аналіз існуючих іструментів для створення гри.

4.3. Створення та реалізація гри.

4.4. Представлення розробки гри

4.5. Висновки

5. Перелік графічного матеріалу.
   5.1. Титульний слайд
   5.2. Мета, об’єкт та предмет дослідження
   5.3. Актуальність роботи
   5.4. Аналіз аналогів
   5.5. Порівняння з аналогами
   5.6. Технічне завдання
   5.7. Програмні засоби реалізації
   5.8. Інструменти використані для реалізації
   5.9. Апробація результатів дослідження
   5.10. Висновки

6. Дата видачі завдання: 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.2022	Виконано
2	Дослідження аналогів та актуальності додатку	20.04.2022	Виконано
3	Аналіз та вибір інструментів для розробки додатку	21.04.2022 – 22.04.2022	Виконано
4	Проектування та реалізація	23.04.2022 – 27.04.2022	Виконано
5	Висновки, оформлення роботи	27.04.2022 – 30.04.2022	Виконано
6	Розробка демонстраційних матеріалів	30.04.2022	Виконано
7	Попередній захист роботи	24.05.2022	
8	Здача роботи	03.06.2022	

Студент _____ Косенко А.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 65 с., 27рис., 11 джерел.

Ключові слова: ігровий двигун, три в ряд, Unity, комп’ютерна гра, жанр шутер, відеогра.Ключові слова:

Об’єкт дослідження – процес розробки відео ігор з використанням двигуна Unity.

Предмет дослідження – розробка комп’ютерної гри жанру шутер на двигуні Unity.

Мета роботи – вдосконалення існуючих механік гри жанру шутер.

Наукова новизна – створення нових та перетворення відомих ігрових механік, притаманних для ігор жанра шутер; створення інтуїтивного та привабливого дизайну гри, що зачепить гравця.

У дипломному проекті був проведений аналіз ринку комп’ютерних ігор та додатків-аналогів. Проаналізовано та виявлено переваги і недоліки програмних інструментів розробки. Проаналізовано особливості розробки комп’ютерних ігор жанрі шутер.

Комп’ютерну гру було створено за допомоги ігрового двигуна Unity. Ігрова логіка написана у середовищі розробки Rider на мові програмування C#. В якості платформи було обрано операційну систему Windows. Для створення елементів графічного дизайну та прототипування гри було використано Blender.

Дана комп’ютерна гра в жанрі шутер може слугувати як і швидким способом ненадовго відволіктися, так і повноцінною грою, яка забезпечить швидке занурення у стан потоку та дозволить отримувати задоволення від подолання поставлених завдань.

Галузь використання – розваги.

ЗМІСТ

ВСТУП

Існують багато жанрів в тому числі як стрілялки(шутер). Відеогра один із способів відпочити від тяжкого дня або вбити час. Існують багато жанрів в тому числі як стрілялки або ігри в жанрі шутер. Інакше кажучи, відпочити та рзоважитись, більше того це навіть корисно . Ігрові розваги, у свою чергу, дають можливість в умовах подібного мислення вирішити насущу проблему. Оскільки такі ігри розробляються на ПК, вони мають системні вимоги до ОС та комплектуючих які б дозволяли запускати гру. Але в моїй грі зможе пограти кожен не зважаючи на системні вимоги, бо вони будуть не високими. В шутерах максимально легко і комфортно грати. Короткі сесії та динамічний прогрес забезпечують швидке занурення у стан потоку та дозволяють отримувати задоволення вже за декілька хвилин.

Шутерні ігри безсумнівно стали одним із трендів останніх років. Якщо подивитися на топи які проводили дослідники, то можна побачити, що серед 10 найкращих жанрів за кількістю завантажень –2 належать до шутерних ігор.

Об’єкт дослідження – процес розробки відео ігор з використанням двигуна Unity.

Предмет дослідження – розробка комп’ютерної гри жанру шутер на двигуні Unity.

Метою даної роботи є розробка відео гри в жанрі шутер.

Наукова новизна проекту – створення нових та перетворення відомих ігрових механік, притаманних для ігор жанра шутер; створення інтуїтивного та привабливого дизайну гри, та основних механік гри.

У дипломному проекті був проведений аналіз шутерних ігор та додатків-аналогів. Проаналізовано та виявлено переваги і недоліки

програмних інструментів розробки. Проаналізовано особливості розробки мобільних ігор жанра шутер.

Гра розроблена в ігровому двигуні Unity, програмний код написаний в середовищі розробки Rider на мові програмування C#. Для створення елементів графічного дизайну та прототипування гри було використано Blender.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GPU - graphics processing unit

CPU - central processing unit

ПК – персональний комп’ютер

ОС – операційна система

P2E - Play-to-Earn

UE4 – Unreal Engine 4

MVC - model view controler

CPV - cost per view

API - application programming interface

RPG - role-playing game

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз загальних характеристик ігор

Сьогодні в It-індустрії одним з найпопулярнішим напрямком є Game development.

Game development це процес розробки гри під певну платформу. Це може бути гра для ПК, для консолей, для смартфонів, для VR-шоломів і т.д.

Комп'ютерна відеогра – це програма, що служить для організації ігрового процесу, яка передбачає взаємодію з користувацьким інтерфейсом або пристроєм введення, наприклад джойстиком, контролером, клавіатурою або пристроєм для визначення руху, для створення візуального зворотного зв'язку. Цей відгук відображається на пристрої відображення відео, як-от телевізор, монітор, сенсорний екран або гарнітура віртуальної реальності. Відеоігри часто доповнюються звуковим зворотним зв'язком, який передається через динаміки чи навушники, а іноді й іншими типами зворотного зв'язку, включаючи тактильні технології.

Комп'ютерні ігри часто створюються на основі сторонніх джерел. Так, наприклад, комп'ютерні ігри як «METRO 2033» та «S.T.A.L.K.E.R», створені українською компанією 4A Games, та GSC Game world були створені за мотивами твору Дмитро Глуховський «METRO 2033» та Аркадій Стругацький, Борис Стругацький «Пикник на обочине». В 2015 році провели опитування в США Entertainment Software Association, де дізналися, що одним з найпопулярнішим жанром вважаються саме шутер. Далі приводжу таблицю найпопулярнішими жанри.

Таблиця 1.1 – Популярності відеоігор

Жанр	**популярність**
Екшн	28,2 %
Шутер	**21,7 %**
Спортивні	13,3 %
рольові	9,5 %
Файтинг	6 %
перегони	5,2 %
Стратаегії	4,1 %

Жанр відеогри використовується для класифікації відеоігор відповідно до інтерактивних ігрових дій гравця.

Шутер це один з видів жанру, який вимагає від гравця боротися з суперником шляхом стрілянини. Також шутер можна ще поділити, як от першої (CS GO, Wolfenstein the new order, DOOM) так і от третьої особи (Макс Пейн). Існують різновиди як тактичні, в яких ігровий персонаж діє у складі команди (CS GO, Tom Clancy's Rainbow Six® Siege), аркади (Alien Shooter), стелс-екшн, метою якого є приховані дії для виконання завдань, без прямого знищення противників (серія Hitman). Щодо ігор, де основою ігрового процесу є знищення великих кількостей ворогів, а сама стрілянина в цьому переважає над тактикою і влучністю, застосовується термін Shoot 'em up (R-Type, Touhou, Contra).

За деякими популярними іграми проводяться змагання різних видів масштабності, від регіональних до світових, які називаються кіберспортом. Деякі країни зараховують кіберспорт до офіційного виду спорту, та, наприклад, уряд США у 2011 році визнав комп'ютерні ігри окремим видом мистецтва, поряд із театром та кіно. Зазвичай змагання проводяться в спортивних симуляторах, шутерах або в стратегіях реального часу. Дисциплінами кіберспорту є конкретні ігри. На даний момент дисциплін дуже багато, але найпопулярнішими в плані шутера вважаються "Counter-Strike: Global Offensive", саме з нього проводяться наймасштабніші змагання, на яких збираються цілі стадіони глядачів.

Український гравець Simple цієї ж команди у грі CS:GO визнаний другий рік поспіль найкращім гравцем у світі.

Комп'ютерні ігри мають настільки істотний вплив на сучасне суспільство, що вже можуть виступати в якості навчального матеріалу або дозволяють використовувати гравців в науково-дослідних цілях. Таким чином, в деяких європейських навчальних закладах почали використовувати відомі ігри для

навчання. У школах Швеції гра Minecraft є обов'язковою частиною програми з 2013 року. У Австралії гру використовують під час уроків природознавства, США - вивчення історії.

Використовують методику лише у молодших класах, старшим учням гра не настільки цікава. На початку уроку діти закріплюють вивчений матеріал, а потім виконують завдання у Minecraft. Наприклад, шукають скриню, а щоб її відкрити, вирішують приклади та завдання. Якщо рішення правильне, скриня відкривається, а в ньому — сувій із новим завданням.

Відеоігри можуть бути дуже корисні в тому сенсі, що вони створюють ефект симуляції дії, але при цьому не несуть будь-якої очевидної небезпеки. У дослідженні 2006 року, проведеному Бавельє і дослідником К. Шоном Гріном, дев'ять осіб, які не грали в ігри, грали в Medal of Honor: Allied Assault по одній годині на день протягом 10 днів, а вісім людей, які не грали в ігри, грали тетрис протягом того ж періоду часу. Потренувавшись із військовим стрільцем менше двох тижнів, неігрові гравці змогли покращити свої результати у трьох тестах зорової уваги — навички, життєво важливої для таких занять, як читання та водіння.

Також Шон Грін вирішив дізнатися, як ігри впливають на нашу здатність до прийняття рішень. Його метою було з'ясувати, чи можуть ігри, що вимагають від нас уваги до дрібних деталей, підвищувати нашу здатність до сприйняття і тим самим покращувати прийняті рішення.

Група молодих людей без досвіду комп'ютерних ігор награла 50 годин у стрілялку від першої особи. Другій групі запропонували неквапливу стратегію. Виявилося, що шутери, які вимагають уваги до об'єктів, які несподівано виникають і зникають на периферії поля зору, значно покращують так зване «низькорівневе сприйняття» і справді прискорюють прийняття рішень. Відмінні новини для фанатів «Halo» та «Call of Duty».

1.1.1. Класифікації ігор

Відеоігри класифікуються за багатьма критеріями наприклад:

- Ігровими платформами
- Операційними системами
- Жанрами кількістю гравців
- Візуальною складовою, стилістикою.
- Сюжетна гра

1.1.2. Класифікація за кількістю гравців

Усі ігри можливо розділити на одиночні та мультиплеєрні. Дивлячись на те яке саме буде розроблена гра, буде зрозуміло її орієнтир.

Одиночна гра – вид гри, яка розрахована на участь однієї людини. Зазвичай це сюжетна гра де, гравцю протистоїть штучний інтелект, а його метою є проходження певного рівня локації, де для цього гравцеві потрібно накоплювати ресурси для покупки зброї, якщо це гра в жанрі шутер або прокачувати навички. Часто ці цілі комбінуються.

Іншим видом ігор є мультиплеєрні. Цей режим гри пристосований до гри більше ніж однієї людини. Якщо це гра по типу CS GO то зазвичай до 10 гравців одночасно по локальній мережі або інтернету. В багатьох іграх одиночна гра та мультиплеєр комбінуються.

Різновидом мультиплеєра є масові онлайн ігри. Це ігри, які використовують лише підключення до інтернету. У більшості таких ігор відсутня одиночна гра. Найчастіше цей спосіб використовується в таких жанрах як настільні та рольові ігри .Серед них розрізняють також ігри, що запускаються в браузері і не вимагають скачування та установки додатку на комп'ютер.

1.1.3. Класифікація за ОС

Ігри можна також класифікувати за підтримуваними операційними системами. Зазвичай ігрові студії обирають певну ОС, а інколи з часом більше. Це залежить від фінансових становищ. Тому загалом підтримують обмежену кількість операційних систем, що є перспективними або навіть обов'язковими для максимальної доступності гри користувачам. Підтримувані ОС можуть бути трьох форм-факторів: комп'ютерні, консольні і мобільні. До комп'ютерних належать Windows, Linux та MacOS. До консольних належать XboxOS та PlaystationOS. У мобільних ОС існує велика кількість різних за популярністю, направленістю та брендами ОС, але загалом більшість не є релевантними у питанні ігор або є сумісними з Android-додатками.

iOS - це закрита мобільна операційна система, створена та розроблена компанією Apple Inc. виключно для її обладнання. Вона має другу за величиною базу смартфонів у світі - 19% ринку операційних систем, але найбільший прибуток через агресивну цінову конкуренцію між виробниками на базі Android. Пристрої iPad та Apple TV також працювали на цій ОС до введення нових окремих у 2019 році. Це основа для інших операційних систем компанії Apple Inc, таких як iPadOS, tvOS та watchOS.

Android - це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, розробленого головним чином для мобільних пристроїв із сенсорним екраном. Наразі використовується також для тв-приставок, автомобільних систем, а іноді і ноутбуків. Android розробляється консорціумом розробників, відомим як Open Handset Alliance і комерційно спонсорується Google. Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java. Програми для Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik. Google пропонує для вільного завантаження інструментарій для розробки. Для Android був розроблений окремий формат інсталяційних пакетів .apk. У березні 2017 року ОС Android стала найпопулярнішою ОС, з якої

виходили в інтернет, конкуруючи навіть з Windows. Взагалі на ринку мобільних пристроїв Android займає впевнене перше місце по кількості базі смартфонів - 78%.

1.1.4. Класифікація за жанрам

Внаслідок того, що в іграх критерії приналежності до певного жанру гри чи іншого жанру не завжди зрозуміла визначені однозначно, класифікація комп'ютерних 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. За основний критерій сучасного розподілу жанрів відеоігор беремо вид активності, який найбільш часто здійснюються в іграх цього жанру. Так ігри в загальному можуть ділитися на три великі групи: ігри дії, ігри контролю та ігри інформації (табл. 1.1).

Ігри інформації: Як зрозуміло з назви, головне в іграх цієї групи - отримання інформації в усіх її проявах. В таких іграх іноді присутній і планування, і динаміка, але інформація в них найважливіше.

Ігри дії: Головне в іграх цієї групи - рухи, які необхідний здійснювати керуючи якимось тілом або технічним засобом.

Ігри контролю: Група «ігри контролю» складається з тих ігор, головна суть яких - планування подій і управління для досягнення переваги в подальшому.

Існують ігри, у яких присутні елементи декількох жанрів. У цьому випадку гру зараховують або до одного з жанрів, який є основним, або до декількох, що найбільше виділяються у грі.

Безсумнівно, під впливом розвитку індустрії жанрова відмінність ігор також безперервно еволюціонує. Представлена тут класифікація жанрів не є повною і може доповнюватися та змінюватися з часом. З'являються як нові піджанри, так і цілі окремі напрямки. Одним з таких напрямків є казуальні ігри.

Казуальна гра – це відеогра, яка не потребує великих зусиль для ігрового процесу і призначена для широкого кола користувачів. Вона не вимагає від користувача будь-яких особливих навичок або інструкції для ігрового процесу.

Найчастіше час на проходження таких ігор невеликий, тому вони добре підходять для тих, хто не може приділяти грі багато часу. В них грають від випадку до випадку, мимохідь, найчастіше - щоб якось «вбити» час. Казуальні відрізняються простими правилами та мають прощений інтерфейс. Багато подібних ігор мають також яскраву і привабливу графіку з простою колірною схемою і мінімум тексту. В основному застосовується 2D-дизайн гри. Механіки не вимагають особливих зусиль і майже завжди є нескінченно-зацикленими. В них можна грати нескінченну кількість часу, що призводить до звикання.

В даний час для визначення таких ігор не має чітких меж і до казуальних відносять ігри, дистрибутиви яких відносно невеликі розміром, є безкоштовними і поширюються так, щоб їх було легко завантажити і відразу розпочати ігровий процес.

Часто казуальні ігри обговорюються як бізнес-модель, а не повноцінний жанр мобільних ігор, так як такі ігри дешеві в розробці та легкі в дистрибуції.

1.1.5. Класифікація за системою монетизації

Одна із головних причин чому існує ігрова індустрія це гроші. Але яким чином заробляють ігрові компанії?

Один важливий критерій це можливість отримати прибуток для своєї компанії. Тому бувають декілька способів заробляти собі на хліб.

За системою монетизації ігри бувають:

• Buy-to-play – платить за копію гри один раз, отримуючи повний обсяг контенту. Можливе придбання нових доповнень;

• Free-to-play – ігрового контенту надається безкоштовно, якщо гравець хоче отримати більше – платить;

• з періодичної підпискою – мати доступ до гри, потрібно регулярно оплачувати підписку (раз на місяць або рік). При цьому, поновлення та доповнення до таких ігор, частіше за все, доступні безкоштовно;

• безкоштовні з внутрішньоігровими покупками – ігри умовно безкоштовні, але, зазвичай, щоб мати можливість на рівних змагатися з іншими гравцями,

доводиться платити. У деяких подібних іграх внутрішньоігрові покупки настільки необхідні, що їх часом називають Pay-To-Win (плати, щоб перемагати);

• безкоштовні з внутрішньоігровою рекламою – в іграх можуть показуватися рекламні оголошення 3 типів: відео із винагородою (коли ці рекламні ролики проглядаються, гравець нагороджується внутрішньоігровими нагородами), банерна реклама (реклама, що знаходиться зазвичай у нижній частині екрана користувача), міжсторінкові оголошення (рекламні оголошення, що відображаються між ігровими сеансами).

1.2. Дослідження ігрового ринку

Більше ніж 30 років існують комп'ютерні ігри, за цей час ігрова індустрія дуже швидко розвинулась. Але з'явився досить могутній конкурент, це ринок мобільних ігор який ще швидше процвітає. За останні пару років ми бачили, як мобільні ігри б'ють рекорди та заробляють мільярди. Зростаюче проникнення смартфонів і технологічний прогрес із дедалі більшим впровадженням нових технологій для розробки ігор є основними факторами розвитку індустрії мобільних ігор.

Але комп'ютерна індустрія не сидить на одному міть і також продовжує швидкими темпами розвиватися. Так на ігровому ринку працюють такі компанії: Microsoft Corporation (Редмонд, Вашингтон, США), Nintendo Co., Ltd (Кіото, Японія), Rovio Entertainment Corporation (Еспоо, Фінляндія), Nvidia Corporation (Каліфорнія, США), Valve Corporation (Вашингтон, США), PlayJam Ltd (Лондон, Сполучене Королівство), Electronic Arts Inc (Каліфорнія, США), Sony Group Corporation (Токіо, Японія), Bandai Namco Holdings Inc (Токіо, Японія), Activision Blizzard, Inc (Каліфорнія , США) та інші гравці.

Ключові гравці на ринку зосереджені на покращеннях та покращеннях, щоб надати своїм користувачам гри преміум-класу. Для цього вони працюють над апаратним забезпеченням, пропускною спроможністю програмного забезпечення та швидкістю мобільного інтернету. Більше того, багато ключових гравців зосереджено на придбаннях. Наприклад, у вересні 2019 року

Microsoft придбала компанію з виробництва відеоігор ZeniMax за 7,5 млрд. доларів США. Очікується, що ці кроки, зроблені великими компаніями, сприятимуть зростанню ринку протягом прогнозованого періоду

Індустрія відеоігор процвітає у всьому світі, незважаючи на пандемію. Світова економіка зіткнулася з безпрецедентними економічними потрясіннями через карантин. Проте люди практикували соціальне дистанціювання і занурилися у цифровий світ, щоб відволіктися від жахливої ситуації, спричиненої вірусом. Згідно з повідомленнями, індійці проводили близько 7-8 годин на тиждень, граючи в онлайн-ігри, і їхня тривалість збільшилася на 53% під час блокування. Отже, ігрова промисловість процвітала, а доходи різко зросли.

1.3. Дослідження аналогів гри в жанрі шутер

Counter-Strike: Global Offensive - це розрахований на багато користувачів онлайн-шутер від першої особи. Дата виходу КС ДО – серпень 2012 року. Минуло вже понад чотири роки, але гра залишається популярною і сьогодні: різні турніри та змагання збирають багатомільйонну аудиторію з різних країн.(Див. рис.1.1)

Основна ідея гри – проведення контртерористичної операції у окремій локації. В операції беруть участь терористи та спецназ. Терористи повинні підірвати бомбу, утримати заручників або переклацати спецназ, а спецназ не повинен дати терористам здійснити задумане.

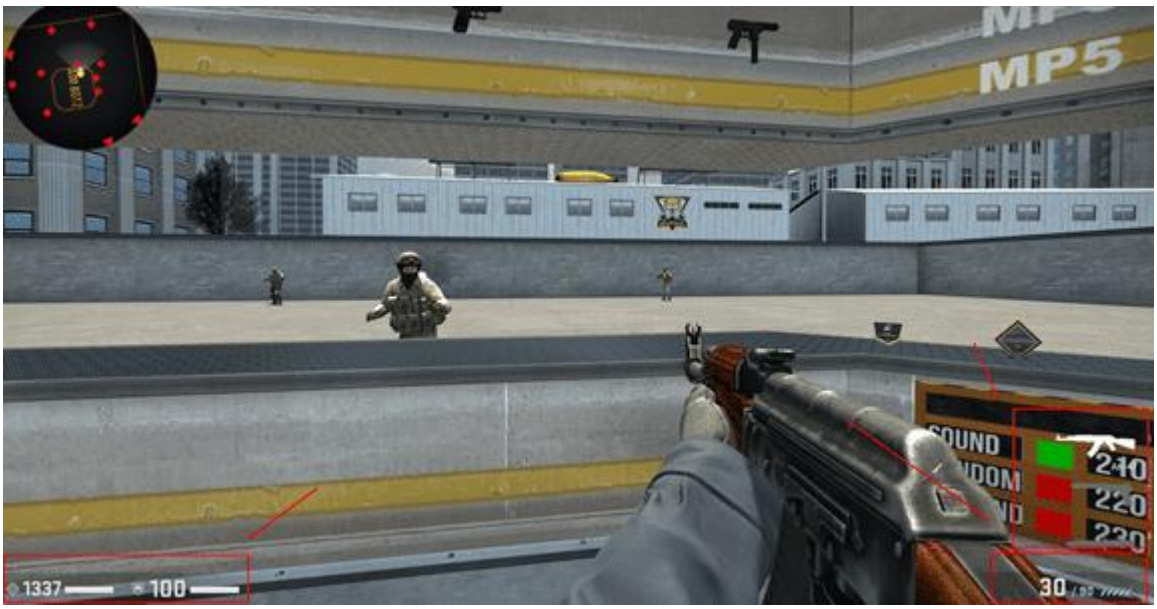


Рисунок 1.1 – Вигляд гри CS GO

Бойові дії відбуваються на картах, де кожна з команд має виконати певне завдання. Щоб її виконати, доведеться постаратися – інші гравці заважатимуть усіма силами, стрілятимуть з-за кожного кута, кидатимуть гранати та заходитимуть з флангів та атакуватимуть у спину.

Будь-який шутер не обходиться без гарних гармат, і CS:GO не виняток. В арсеналі терористів та спецназу зброя на будь-який смак: пістолети, пістолети-кулемети, дробовики, кулемети, автоматичні та снайперські гвинтівки. Найпопулярніші – AK-47, M4A4, M4A4-S, AWP, Scar 20. Зазначимо, що кожен гравець повинен навчитися точно стріляти зі всіх видів зброї – не завжди є можливість купити улюблену гармату. Плюси та причини популярності

- Основні переваги CS: GO:
- Створений зрозумілий інтерфейс

Чудова реалізація звуківЗ першої версії Counter-Strike завоювала любов серед гравців, яка лише посилювалася з кожною новою частиною гри.

Основні мінуси CS: GO, з якими стикаються гравці.

- Механіка стрільби застаріла
- Не має механіки прицілювання
- Погана реалізація механіки руху

DOOM Eternal це ураганним шутером від першої особи, в якому гравцеві доведеться багато бігати, багато стрибати, багато стріляти і іноді вирішувати прості головоломки. Щодо DOOM 2016 геймплей у новій частині особливо не

змінився, проте деякі нові ігрові механіки стають помітними практично одразу.

Рисунок 1.2 – Вигляд гри Doom Eternal

Сюжет DOOM Eternal і справді досить простий - на нашу рідну планету напали демони і головний герой гри, як справжній джентльмен, вирішив за неї заступитися. Насамперед гравцеві доведеться провчити трьох жерців, які керують усім цим демонічним неподобством. Здаватися без бою жерці не мають наміру і тому спершу нам доведеться повозитися з натовпом демонів рангом нижче.

У перервах між бійками гравець дізнаватиметься цікаві подробиці про інопланетне вторгнення та їх дійових осіб, а для ще більшого занурення у всесвіт ігри розробники DOOM Eternal щедро привласнили своє дітище різноманітними записками-притчами, пропускати які не рекомендується. Зрозуміло, якщо гравцеві це все справді цікаво.

Отже, головний герой цієї гри став настільки крутим, що практично скасував гравітацію і тепер може парити в повітрі акі Домінік Торетто — відтепер гравцеві доступні подвійні стрибки та подвійні ривки. З подвійними стрибками ми знайомі ще з часів попередньої частини, а ось на ривках слід зупинитись детальніше.

По-перше, ними можна (і потрібно) користуватися в повітрі, щоб досягти «важкодоступних» місць, яких у грі просто хоч греблю гати — без комбінацій

стрибків і ривків пройти локацію далі часом буває неможливо.

По-друге, ривки можна використовувати в бою для ухилення від атак супротивників - швидко виносити своє тільце з-під вогню можна в будь-якому з чотирьох напрямків.

Щоб ще більше урізноманітнити способи переміщення по локаціях, розробники придумали стрімкі стіни з особливою поверхнею, до якої Кат Рока прилипає як магніт. Відштовхуватися і дертися по таких стінах теж можна.

Основні переваги DOOM Eternal:

- Реалізовано різноманітність зброї та монстрів;
- Чудово реалізована механіка стрільби;
- Чудово реалізовано механіка руху.

Основні мінуси:

- Дуже перевантажений інтерфейс та візуалізація гри;
- Платформінг не підходить шутеру;
- Погана реалізація RPG механік.

Таблиця 1.2 – Зведені результати характеристик

Показник	Counter-Strike: Global Offensive	DOOM Eternal
Платформа	Windows	Windows, PS4, Switch
Сюжет	нема	має
Одиночна	ні	так
Розрахована на багато користувачів	так	ні
Супер здібності RPG	немає	має
Тактична	так	ні
Динамічна	так	ні
Стабільність продукту	так	так

1.4. Опис ігрових механік

– Інтерфейс UI

Один із найбільш недооцінених, але вкрай важливих елементів розробки. Він визначає, як користувач буде взаємодіяти з основними системами гри, і не тільки передає гравцеві інформацію про персонажа та ігровий світ, але й формує певну модель його поведінки. В шутерах інтерфейс передає лише ігрову інформацію, тобто скільки ще є боєзапасу, абос скільки залишилось хітпоінту. Керування ігровим аватаром

В будь-який грі курування своїм аватаром здійснюється через маніпуляцію клавіатурою або мишкою, якщо це комп’ютерна гра, або через геймпад. Клавіатуру ми використовуємо для того, щоб ігровий аватар міг переміщатися з однієї осі координат до іншої. Також ми можемо підбирати предмети, або стріляти зі своєї зброї.

– Використання ігрового об’єкту «зброя»

Одним з найголовніших механіки шутерної гри це зброя.

Вона нам необхідна для того щоб знищувати умовного ворога в грі. Але для того щоб ми могли її використати потрібно до неї боєзапас, тобто якщо у нас закінчується боєзапас то стріляти ми не можемо. Також в механіки зброя є максимальна кількість боєзапасу та поточна кількість тобто максимальна ємкість ражка зброї. Після того як закінчується поточна кількість боєзапасу то відбувається процес перезарядки зброї.

1.5. Постановка завдань дослідження

Під час розробки концепції програми було визначено, що перш ніж проектувати гру, потрібно вибирати конкретні інструменти та засоби реалізації, потрібно дослідити сферу рушіїв для створення проектів та визначити які патерн технології потрібно використати

Тому для ефективності побудови додатку, були поставлені завдання на дослідження певних технологій та інструментів, що дозволять реалізувати розроблену концепцію гри, а саме:

- Дослідити функціональність рушія
- Дослідити інструменти побудови десктопних додатків за допомогою мови програмування C#
- Дослідити патерн технології
- Дослідити структуру та функціональність

2. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ІСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ГРИ

2.1. Огляд двигуна Unity

Разом зі створенням перших ігор програмісти дійшли до того що кожна гра містить загальні компоненти, навіть попри відмінність апаратних платформ. А перші ігри мали місце на гральних автоматах розміром із холодильник.

Загальна для ігор функціональність - графічні рішення, ігрові механіки, розрахунок фізики та інше - стала виділятися в окремі бібліотеки, але, щоб бути "ігровим рушієм" було ще далеко. Багато в чому це було пов'язано із серйозною відмінністю програмно-апаратних платформ та невизначеності у самих іграх. Адже жанри та типи ігор ще потрібно було винайти, при тому, що багато перших ігор були текстовими. Власне, саме для ранніх адвенчур та платформерів і стали виникати ігрові движки, особливо з розвитком графіки – гарним прикладом можна назвати Adventure Game Interpreter (AGI). При розробці King's Quest у далекому 1984 році, програмісти Sierra On-Line зіткнулися з незручністю низькорівневої розробки такої складної та перспективної за графіком у ті часи гри – і розробили набір рішень, яким і став AGI. Усього на ньому було випущено 14 різних ігор за 5 років на 7 різних платформах, тому поняття "кросплатформенність" було важливим вже тоді.

Проте, движки на той час рідко виходили межі початкової компанії-розробника і, зазвичай, були досить вузькоспеціалізованими під конкретний жанр гри.

Для того щоб розробити свою відеогру потрібно для початку обрати рушій. Цим рушієм буде Unity так як основна його перевага те що вона одна із небагатьох являється безкоштовною, та одна із простіших для використання.

Unity – міжплатформне середовище розробки комп'ютерних ігор. Unity дозволяє створювати програми, що працюють під більш ніж 20 різними операційними системами, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші. Випуск Unity відбувся у 2005 році і з того часу триває постійний розвиток.

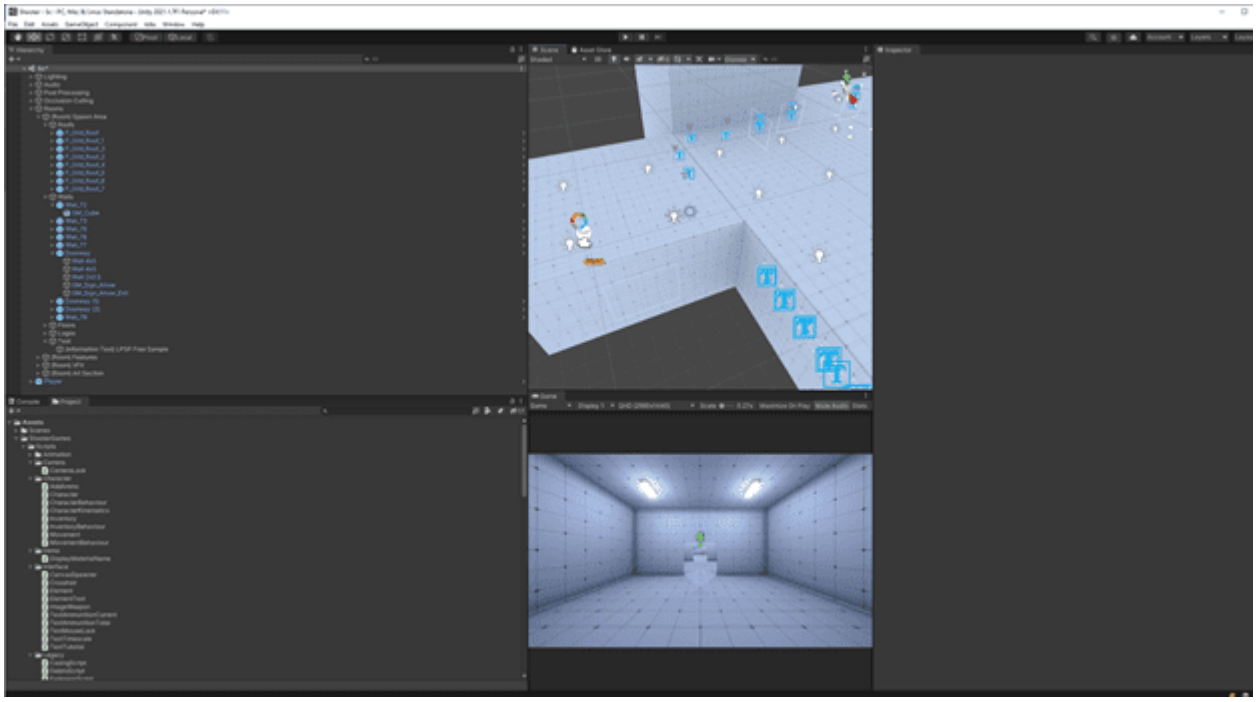


Рисунок 2.1 – Вигляд рушій юніті

Основними перевагами Unity є наявність візуального середовища розробки, міжплатформної підтримки та модульної системи компонентів.

До недоліків відносять поява складнощів при роботі з багатокомпонентними схемами та утруднення при підключенні зовнішніх бібліотек.

Редактор Unity має простий Drag&Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Двигун підтримує дві скриптові мови: C#, JavaScript (модифікація). Раніше була підтримка Boo (діалект Python), але його забрали в 5-й версії. Розрахунки фізики здійснює фізичний двигун PhysX від NVIDIA.

Проект в Unity ділиться на сцени (уровни) — окремі файли, що містять свої ігрові світи зі своїм набором об’єктів, сценаріїв та настроїв. Сцени можуть містити в собі як, власно, об'єкти (моделі), так і пусті ігрові об'єкти — об'єкти, які не мають моделі («пустишки»). Об’єкти, у своїй черзі містять набори компонентів, з якими й взаємодіють скрипти. Також у об’єкта є назва (в Unity допускається наявність двох і більше об’єктів з одиничними назвами), може бути тег (метка) і шар, на якому він повинен відображатися. Так, у будь-якого об’єкта на сцені обов’язково присутній компонент Transform — він зберігає в собі координати місця розташування, повороту та розмірів об’єкта по всьому трьом осям. У об’єктах із видимою геометричною формою також за умовчанням присутня компонент Mesh Renderer, який виконує модель об’єкта видимої.

К об’єктам можна застосовувати коллізії (в Unity т. н. коллайдери — коллайдер), яких існує кілька типів.

Також Unity підтримує физику твёрдих тел і тканини, а також фізику типу Ragdoll (тряпична кукла). В редакторе є система спадкування об'єктів; дочерні об'єкти будуть повторювати всі зміни позицій, повороту та масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

При імпорті текстури в Unity можна сгенерувати альфа-канал, mip-уровни, normal-map, light-map, карту відражений, проте безпосередньо на моделі текстури прикріпити неможливо — буде створений матеріал, який буде призначений для шейдера, а потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання та редагування шейдерів. Редактор Unity має компонент для створення анімації, а також анімацію можна попередньо створити в 3D-редакторі та імпортувати разом із моделлю, а потім розбити файли.

Unity 3D підтримує систему Level Of Detail (сокр. LOD), суть якої полягає в тому, що на дальньому зростанні від гравця високодеталізовані моделі замінюються на менш деталізовані, і наоборот, а також система відбракування оклюзії, суть якої в тому, що у об'єктах, не потрапляючи в поле зору камери не візуалізується геометрию та коллізію, що знімає нагрузку на центральний процесор і дозволяє оптимізувати проект. При компіляції проекту створюється виконуваний (.exe) файл-ігри (для Windows), а в окремій папці — дані гри (включаючи всі ігрові програми та динамічні підключаються бібліотеки).

Движок підтримує безліч популярних форматів. Моделі, звуки, текстури, матеріали, скрипти можна запаковувати у форматі .unityassets і передавати іншим розробникам або викладати у вільний доступ. Цей формат використовується у внутрішньому магазині Unity Asset Store, у якому розробники можуть безкоштовно та за гроші розміщувати в загальному доступі різні елементи, необхідні при створенні ігор. Щоб використовувати Unity Asset Store, необхідно мати обліковий запис розробника Unity. Unity має всі необхідні компоненти для створення мультиплеера. Також можна використовувати відповідний спосіб контролю версії. Як приклад, Tortoise SVN або Source Gear.

В Unity входить Unity Asset Server — інструмент для спільної розробки на

базі Unity, який є доповненням, додає версії контролю та ряд інших серверних рішень.

2.2. Огляд Ігрового двигуна Unreal Engine 4

Unreal Engine 4 Unreal Engine 4 - популярний і широко використовуваний ігровий двигун, розроблений Epic Games.

Він використовується в багатьох сучасних AAA іграх, таких як власний шутер Fortnite у жанрі королівської битви від Epic чи інших популярних ігор, таких як Rocket League від Psyonix.

Це дозволяє розробляти на кількох платформах від ПК до консолей, таких як PS4, Xbox One та Nintendo Switch. Це одна причина, через яку він так широко використовується, через його гнучкість для роботи між цими різними платформами.

Досвідченіші програмісти можуть використовувати мову C++ для створення власних сценаріїв, що запускаються в ігровому движку. Більше розробників-аматорів можуть використовувати його дуже потужні характеристики, які в основному є готовими кодами блоків, які ви можете додати у свої об'єкти для взаємодії. Він має потужний матеріал та інструменти анімації для художників, які дозволяють швидко створювати складні сцени. Налаштування деяких з цих функцій спочатку може бути складним, але наведено кілька прикладів, де можна просто змінити параметри, поки не створюєте те, що шукаєте. Він використовує широко розповсюджений робочий процес PBR для своїх матеріалів та візуалізації. Це у поєднанні з динамічним або запеченим у тінях та освітленні дозволяє отримати неймовірний фотореалістичний контент, який все ще працює у режимі реального часу.

Функція Blueprints, яку я згадував раніше, дозволяє створювати прості скрипти, що взаємодіють один з одним. Все це використовує візуальний інтерфейс, тому навіть якщо ви ніколи не кодували життя, кілька коротких підручників допоможуть вам на шляху до робочої гри. Є також вбудовані інструменти, що значно прощають створення віртуальної або додаткової реальності.

Високу популярність віртуальної реальності в останні роки неможливо ігнорувати, тому завжди матиме доступ до руху корисного автомобіля, що легко допоможе вам створити контент для цих платформ, зокрема Oculus Rift та HTC Vive.

Unreal також є інструменти, що дозволяють легко створити свій ландшафт. Ви можете малювати своїми власними матеріалами за допомогою миші та змішувати їх разом, щоб, наприклад, трава вільно перетікала в бетонну дорогу.

Потім можна використовувати сітки дерев і трав і намалювати цілі ліси або ділянки квітів. Ви можете створити повні сцени у відкритому повітрі за лічені хвилини, і вони оптимізовані для запуску у вашій грі, якщо ви використовуєте миші рослини, надані за допомогою Unreal Engine.Огляд існуючих компонентів та функцій у Unity3D

З плюсів даного ігрового двигуна можна відмітити декілька:

Універсальність. UE4 можна використовувати і для PlayStation, і для Switch, і для ПК, – буквально для чого завгодно – це дає потрібну гнучкість при розробці гри;

- велика бібліотека ассетів;
- простота входу через Blueprints. Щоправда, для професійної роботи все одно доведеться вивчити C++;
- хороша задокументованість та підтримуваність, активна спільнота з купою opensource-інструментів;
- можливість безкоштовного використання для маленьких проектів.

Серед мінусів двигуна можна вказати декілька недоліків:

- високий поріг входження для розробників ігор. Без огляду на те, що Blueprint спрощує розробку, все ж для простої людини цей ігровий двигун буде являтися занадто складним через дуже широкий функціонал, що надає Unreal Engine 4 для розробників;
- багато розробників скаржаться на мізерну та застарілу документацію з вузькоспеціалізованих тем;
- двигун орієнтований на складні ігри з сучасною 3D графікою.

З 3 березня 2015 року Unreal Engine 4 став безкоштовним, але якщо щоквартальний прибуток компанії розробника перевищує 3000 доларів, то

необхідно передавати компанії Epic Games 5% від прибутку з продажів гри.

2.3. Огляд існуючих компонентів та функцій у Unity3D

2.3.1. Опис функцій Unity

Скрипти маніпулюють змінними за допомогою функцій. Існує ряд функцій, які автоматично запускаються всередині Unity.

Awake викликається лише один раз, коли створюється екземпляр GameObject з цим компонентом. Якщо GameObject неактивний, він не буде викликаний, поки не стане активним. Однак Awake викликається, навіть якщо об’єкт GameObject активний, але компонент не ввімкнено (з маленьким прапорцем біля його імені). Ви можете використовувати Awake для ініціалізації всіх змінних, яким потрібно призначити значення.

Start – як і Awake, Start буде викликатися, якщо GameObject активний, але лише якщо компонент увімкнено.

Update викликається один раз на кадр. Сюди ви розміщуєте код для визначення логіки, яка працює безперервно, як-от анімація, AI та інші частини гри, які потрібно постійно оновлювати.

FixedUpdate – це коли ви хочете занятися фізикою. FixedUpdate є фіксоване оновлення

LateUpdate — це функція, схожа на Update, але LateUpdate викликається в кінці кадру. Unity перегляне всі об’єкти гри, знайде всі оновлення та викличе LateUpdates. Це добре для таких речей, як камера. Скажімо, ви хочете перемістити персонажа у своїй грі. А потім він стикається з іншим персонажем і опиняється в іншому положенні. Якщо ми перемістимо камеру одночасно з персонажем, буде похитнутися, і камера не буде там, де їй потрібно. Отже, по суті, це другий цикл, який дуже зручний.

Всі функції мають свій порядок виконання(див. Рис. 2.2)

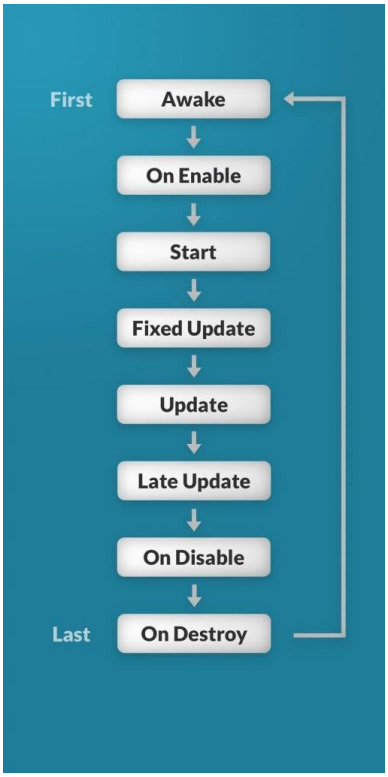


Рисунок 2.2 – Візуалізація порядку виконання функцій Unity

2.3.2. Опис компонентів Unity

–Опис компоненту Rigidbody

Один із найголовніших компонентів в юніті це Rigidbody

Rigidbody вмикає ваші GameObjects діяти під контролем фізики. Rigidbody може отримувати сили та крутний момент, щоб змусити ваші об’єкти рухатися реалістично. Будь-який ігровий об’єкт повинен містити Rigidbody, щоб на нього впливала гравітація, діяти під додатковими силами за допомогою сценаріїв або взаємодіяти з іншими об’єктами через фізичний механізм NVIDIA PhysX.

Rigidbody дозволяють вашим GameObjects діяти під контролем фізичного движка. Це відкриває шлях до такої поведінки, як реалістичні зіткнення та різноманітні типи суглобів. Маніпулювання вашими GameObjects шляхом додавання сил до Rigidbody створює зовсім інше відчуття та вигляд, ніж налаштування компонента трансформації безпосередньо. Загалом, ви не повинні маніпулювати Rigidbody і Transform одного і того самого GameObject - тільки одного або іншого.

Найбільша відмінність між маніпулюванням Transform і Rigidbody полягає у використанні сили Rigidbody можуть сприймати сили та крутний момент, а

трансформатори — ні. Трансформації можна переводити та обертати, але це не те саме, що використовувати фізику. Додавання сил/крутного моменту до Rigidbody фактично змінить положення об’єкта та обертання компонента Transform. Ось чому ви повинні використовувати тільки те чи інше. Зміна трансформації під час використання фізики може спричинити проблеми зі зіткненнями та іншими обчисленнями.

Rigidbody повинні бути явно додані до вашого GameObject, перш ніж на них вплине фізичний движок. Ви можете додати Rigidbody до вибраного об’єкта за допомогою меню Components->Physics->Rigidbody. Тепер ваш об’єкт готовий до фізики; він впаде під силу тяжіння і може отримувати сили за допомогою сценаріїв, але вам може знадобитися додати коллайдер, щоб змусити його вести себе саме так, як ви хочете.(див. рис. 2.3)

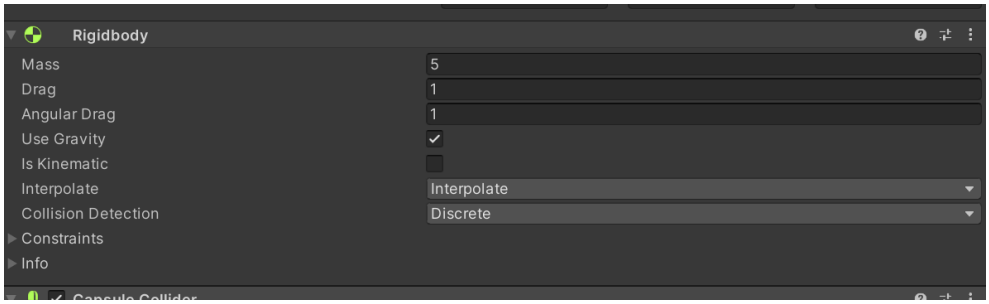


Рисунок 2.3 – вигляд компоненту Rigidbody

– Опис компоненту Collider

Коллайдери — це ще один різновид компонентів, які необхідно додати поряд із Rigidbody, щоб дозволити виникнути зіткнення. Якщо два твердих тіла стикаються один з одним, фізичний механізм не розрахує зіткнення, якщо до обох об’єктів також не приєднано коллайдер. Тверді тіла без коллайдерів будуть просто проходити один крізь одного під час фізичного моделювання. Коллайдери визначають фізичні межі Rigidbody

– Опис компонента Canvas

Canvas — це область, всередині якої мають бути всі елементи інтерфейсу користувача. Canvas — це ігровий об’єкт із компонентом Canvas, і всі елементи інтерфейсу користувача мають бути дочірніми для такого Canvas.

Створення нового елемента інтерфейсу користувача, наприклад зображення за допомогою меню GameObject > UI > Image, автоматично створює Canvas, якщо в сцені ще немає Canvas. Елемент інтерфейсу користувача

створюється як дочірній для цього Canvas.

Область Canvas відображається у вигляді прямокутника в режимі перегляду сцени. Це дозволяє легко розташовувати елементи інтерфейсу користувача, не потребуючи постійного перегляду Game View.

Canvas використовує об’єкт EventSystem, щоб допомогти системі обміну повідомленнями.

Елементи інтерфейсу користувача на Canvas малюються в тому ж порядку, в якому вони з’являються в ієрархії. Першою малюється перша дитина, наступною – друга і так далі. Якщо два елементи інтерфейсу інтерфейсу перекриваються, останній з’явиться поверх попереднього. Щоб змінити, який елемент відображатиметься поверх інших елементів, просто змініть порядок елементів в ієрархії, перетягнувши їх. Порядком також можна керувати за допомогою сценаріїв за допомогою цих методів у компоненті Transform: SetAsFirstSibling, SetAsLastSibling і SetSiblingIndex.

Також потрібно розуміти, що є певна взаємодія між ігровими об’єктами які находяться на сцені та компонентів(див. рис. 2.4)

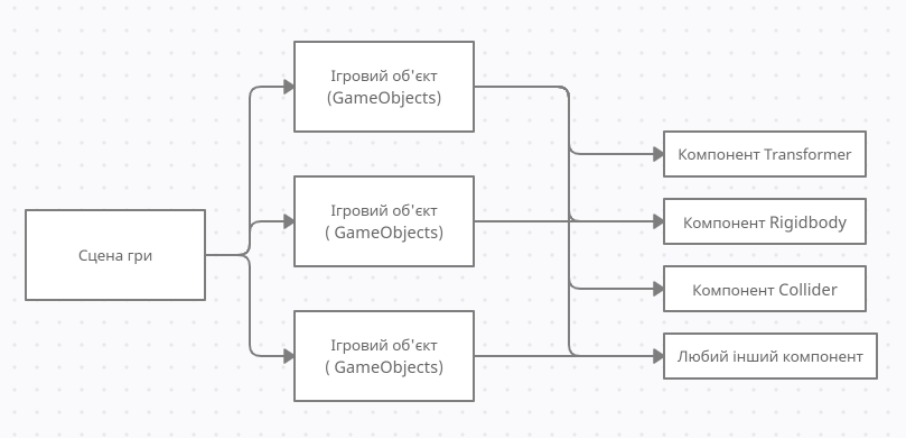


Рисунок 2.4 – Демонстрація залежності та зв'язку між GameObject та компонент

– Опис скриптингу

Скрипти повідомляють нашим GameObjects, як поводитися; це сценарії та компоненти, прикріплені до GameObjects, і те, як вони взаємодіють один з одним, створюють ваш ігровий процес. Написання сценаріїв у Unity відрізняється від чистого програмування. Якщо ви займалися чистим програмуванням, наприклад, створили працюючу програму, ви повинні

розуміти, що в Unity вам не потрібно створювати код, який запускає програму, тому що Unity робить це за вас. Натомість ви зосереджуєтеся на ігровому процесі у своїх сценаріях. Більше того в юніті є своя ієрахія компонентів де скріпти також вважаються компонентом. (див. рис. 2.5)

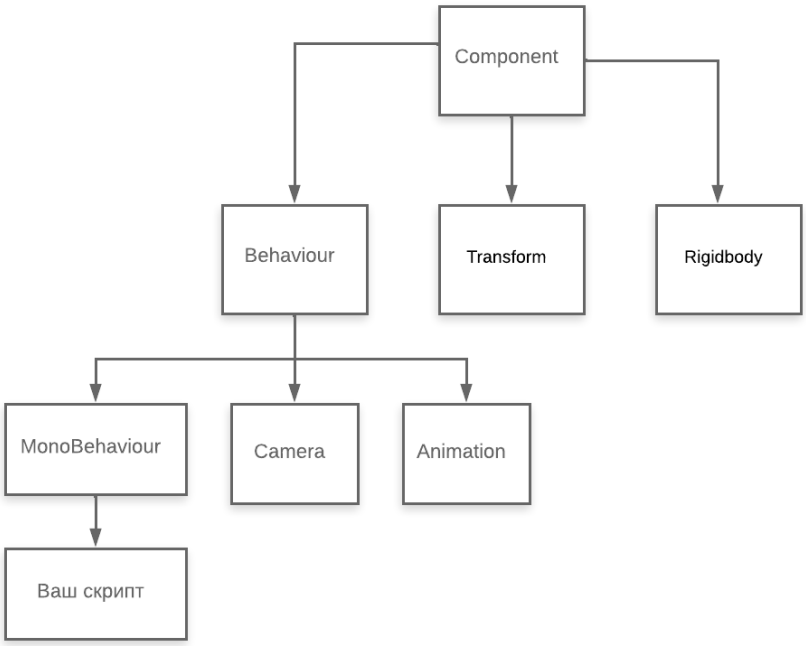


Рисунок 2.5 – Демонстрація ієрархії компонент

2.4. Огляд та переваги інтегрованої среди розробки Rider та C#

– Опис мови C#

На сьогоднішній момент мова програмування C# одна з найпотужніших мов, що швидко розвиваються і затребуваних в IT-галузі. Зараз на ньому пишуться різні програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

C# вже не молода мова і, як і вся платформа .NET, вже пройшов великий шлях. Перша версія мови вийшла разом із релізом Microsoft Visual Studio .NET у лютому 2002 року. Поточною версією мови є версія C# 10.0, яка вийшла 8 листопада 2021 разом із релізом .NET 6.

C# є мовою із Сі-подібним синтаксисом і близький у цьому відношенні до C++ та Java. Тому якщо ви знайомі з однією з цих мов, то оволодіти C# буде легше.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java та C++. Наприклад, C# підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і додатків, що розширюються. І C# продовжує активно розвиватися і з кожною новою версією з'являється все більше цікавих функціональностей.

Нова інтегрована среда розробки (Integrated Development Environment — IDE) JetBrains дозволяє створювати додатки для Windows, веб-приклади та мобільні додатки, як і Microsoft Visual Studio. Але, на відміну від Visual Studio, Rider є кросс-платформною середою, її можна використовувати під Windows, OS X і Linux (тогда як Visual Studio — тільки під Windows). Хоча в JetBrains визнаються, що версія для Linux поки не протестована.

– Опис та переваги Rider

JetBrains вже має продуктову лінійку IDE для розробки практично під усі сучасні популярні технологічні стеки. І з появою Rider список підтримуваних компанією технологій можна рахувати повним

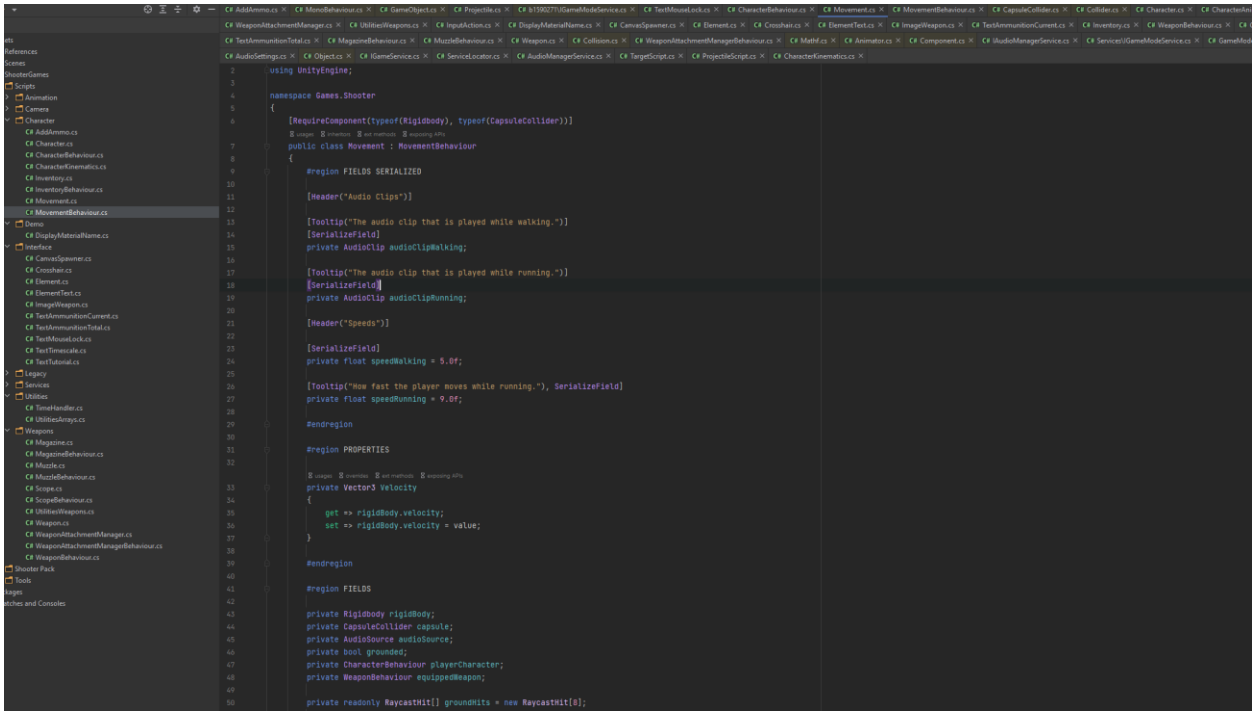


Рисунок 2.6 – Вигляд середовище розробки Rider

– Переваги і недоліки Rider

"Проект Rider звільняє від залежності на Visual Studio і знімає дуже значне

обмеження - крос-платформність платформної розробки під .NET підтримує екосистему, яку створює Microsoft, і не має платних конкурентів, розповіли в компанії.

У блозі на сайті JetBrains описано п'ять основних функцій Rider, які вже є в новому проекті: "розумна" навігація, "розумні" функції редагування коду, перевірка на наявність помилок (з підказками), рефакторинг та декомпілятор. А до переваг середовища, окрім крос-платформенності, розробники відносять: підтримку серед виконання .NET і Mono, підтримку DNX і можливість налагодження проектів .NET і Mono. Згодом розробник обіцяє додати можливість налагодження DNX та підтримку CoreCLR.

Rider побудована на платформі IntelliJ, як і багато інших продуктів JetBrains (IntelliJ IDEA, WebStorm, PHPStorm, DataGrip та ін). Платформа IntelliJ включає наступні компоненти: віртуальну файлову систему, модуль інтерфейсу користувача, текстовий редактор, підтримку синтаксису, компоненти для роботи з кодом, функцію контролю версій, відладчик і тестувальник. Код IntelliJ є відкритим. Він розповсюджується під ліцензією Apache 2.0.

Другим важливим компонентом Rider є ReSharper - розширення для Visual Studio, що надає практично всі функції IDE для розробки C#, XAML, Razor, ASP, JavaScript, TypeScript та інших мовах. Розробником ReSharper також є JetBrains.

Що не вистачає у Rider?

1. Робота з шаблонами для проектів

Йдеться про можливість зберегти проект як шаблонний. Дуже допомагає під час створення "Шаблон мікросервісів"

2. Утиліти для роботи з Microsoft SQL Server

Порівняння схем DB та даних у різних SQL серверах – дуже полонений інструмент. А в деяких моментах просто незамінний.

3. Live unit-тестування – відмінний та дуже зручний інструмент, який дозволяє створювати Unit-тести дуже швидко. Live unit-тестування тримає систему гарячої, тобто тести проганяються в реальному часі, що дуже зручно.

4. Дуже звик до Package Management Console, якої також немає в Rider

5. Спеціальна вставка для об'єктів типу JSON і XML, яка автоматично перетворює на CSharp-класи.

(а точніше, її відсутність)", - повідомив CNews керівник проекту в JetBrains Шкредов Сергій. Для деяких стеків розробки (таких як ASP.NET) проект Rider є прямим конкурентом Visual Studio.

2.5. Огляд та переваги Blender

Blender – це безкоштовний пакет для створення 3D з відкритим кодом, який підтримує практично всі аспекти розробки 3D. Завдяки потужній базі можливостей моделювання є також надійне текстурування, оснащення, анімація, освітлення та безліч інших інструментів для повного створення 3D. Це програмне забезпечення відмінно підходить, якщо ви хочете мати справу тільки зі статичними моделями або поринути у світ анімації.

Як передісторія скажу, що програмне забезпечення Blender було розроблено на базі Blender Foundation, некомерційної організації, створеної в 2002 році. У 2007 році було створено дочірній інститут Blender Institute, в якому зараз знаходиться фонд, який став базою для подальшого розвитку та творчих проектів.

Незважаючи на те, що Blender безкоштовний, він доступний і цінний для широкого кола користувачів, від любителів-початківців до професійних аніматорів. Навіть НАСА використовує її для багатьох своїх публічних моделей! Оскільки він постійно вдосконалюється просунутими користувачами, він може бути деякою кривою навчання для повного любителя.

В основі Blender лежить доступність: надання людям творчої сили, щоб вони могли будувати все, що спаде їм на думку. Для тих, хто хоче створювати власні моделі для 3D-друку, це безцінний інструмент Як ви, можливо, вже зрозуміли, однією з найкращих переваг Blender є те, що він є абсолютно безкоштовним!

Blender був випущений під ліцензією GNU General Public License, яка дозволяє людям: використовувати Blender для будь-якихцілей,поширювати блендер,вивчити, як працює Blender, і змінити його, та розповсюджувати змінені версії Blender.

!

3. СТВОРЕННЯ ТА РЕАЛІЗАЦІЯ ГРИ

3.1. Планування розробки проекту

Шаблон MVC

Для того, щоб розробити та мати можливість розвивати гру буде використаний шаблоний патерн MVC.

MVC (Model-View-Controller) - це шаблон для розробки програмного забезпечення, який зазвичай використовується для реалізації інтерфейсів, даних і керуючої логіки. Він підкреслює поділ між бізнес-логікою програмного забезпечення та відображенням. Цей «поділ обов'язків» забезпечує кращий поділ праці та поліпшення технічного обслуговування. Деякі інші шаблони проектування засновані на MVC, наприклад MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter) та MVW (Model-View-Whatever).

Три частини шаблону проектування програмного забезпечення MVC можна описати таким чином(див. рис. 3.1)

1. Model: керує даними та бізнес-логікою.
2. View: керує макетом та відображенням.
3. Controller: спрямовує команди до моделі та частин виду.

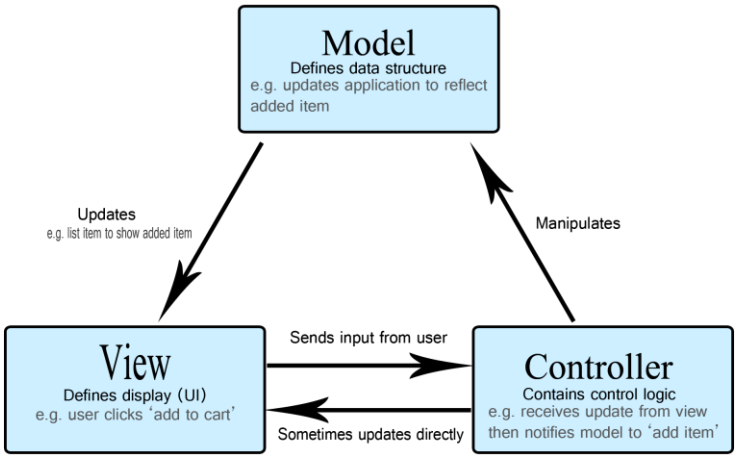


Рисунок 3.1 – Вигляд патерної моделі MVC

– Model (Модель)

Модель визначає, які дані має містити програма. Якщо стан цих даних змінюється, модель зазвичай повідомляє уявлення (тому відображення може змінюватися в міру необхідності) і іноді контролер (якщо для управління оновленим поданням потрібна інша логіка).

Повертаючись до нашого додатка зі списком покупок, модель вказуватиме, які дані мають містити елементи списку — товар, ціна тощо — і які елементи списку вже є.

–View (Вид)

Подання визначає, як відображаються дані програми.

У нашому додатку зі списком покупок представлення визначатиме, як список буде представлений користувачеві, і отримувати дані для відображення моделі.

–Controller (Контролер)

Контролер містить логіку, яка оновлює модель та/або подання у відповідь на введення даних від користувачів програми.

Так, наприклад, наш список покупок може мати форми введення та кнопки, які дозволяють нам додавати або видаляти елементи. Ці дії вимагають оновлення моделі, тому вхідні дані відправляються контролеру, який потім відповідним чином маніпулює моделлю, яка відправляє оновлені дані в представлення.

Однак ви також можете просто оновити представлення, щоб відображати дані в іншому форматі, наприклад, змінити порядок елементів на алфавітний або від найнижчої до найвищої ціни. У цьому випадку контролер може обробляти це безпосередньо без необхідності оновлення моделі.

Принцип SOLID

Для того щоб код був досить зрозумілим буде використано принцип SOLID

SOLID - один із найпопулярніших наборів принципів проектування при розробці об'єктно-орієнтованого програмного забезпечення. Це мнемонічна абревіатура для наступних п'яти принципів дизайну(див.рис.3.2):

- Принцип єдиної відповідальності
- Відкритий/Закритий Принцип
- Принцип заміни Лісків
- Принцип поділу інтерфейсу
- Інверсія залежності

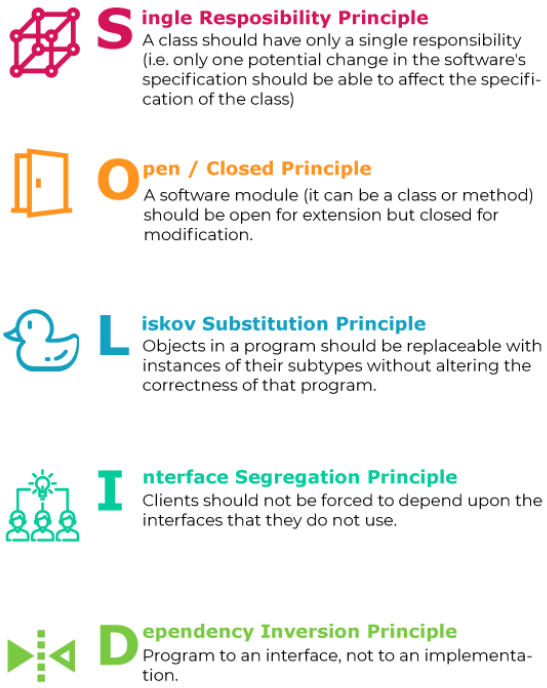


Рисунок 3.2 – SOLID принцип

Аргумент на користь принципу єдиної відповідальності щодо простий: він спрощує реалізацію вашого програмного забезпечення та запобігає несподіваним побічним ефектам майбутніх змін.

Чим довше розробляється проект тим швидше за все, що вимоги згодом можуть змінюватися. Кожен із них також змінює відповідальність як мінімум одного класу. Чим більше обов'язків вашого класу, тим частіше вам потрібно його міняти. Якщо ваш клас реалізує кілька обов'язків, вони більше не незалежні.

Вам потрібно змінити свій клас, як тільки зміниться один з його обов'язків. Це, очевидно, частіше, ніж вам треба було б міняти його, якби він мав лише один обов'язок.

Це може бути незначним, але це також впливає на всі класи або компоненти, які залежать від зміненого класу. Залежно від вашої зміни вам може знадобитися оновити залежність або перекомпілювати залежні класи, навіть якщо вони не стосуються вашої зміни безпосередньо. Вони використовують лише один з інших обов'язків, реалізованих вашим класом, але вам все одно потрібно їх оновити.

Зрештою, вам потрібно частіше змінювати свій клас, а кожна зміна складніша, має більше побічних ефектів і потребує набагато більше роботи, ніж слід. Таким чином, краще уникнути цих проблем, переконавшись, що кожен клас має лише один обов'язок.

Принцип єдиної відповідальності дає ще одну істотну перевагу. Класи, програмні компоненти та мікросервіси, які мають лише один обов'язок, набагато простіше пояснити, зрозуміти та впровадити, ніж ті, які надають рішення для всього. Це зменшує кількість помилок, підвищує швидкість розробки і робить ваше життя як розробника програмного забезпечення набагато простіше. Отже, принцип єдиної відповідальності є важливим правилом, що дозволяє зробити ваш код більш зрозумілим, але не використовуйте його як біблію програмування. Керуйтеся здоровим глуздом під час розробки коду. Немає сенсу мати кілька класів, що містять лише одну функцію.

3.2. Опис програми та її алгоритми

Функції системи представлені UML діаграма прецедентів(Див. рис. 3.3)

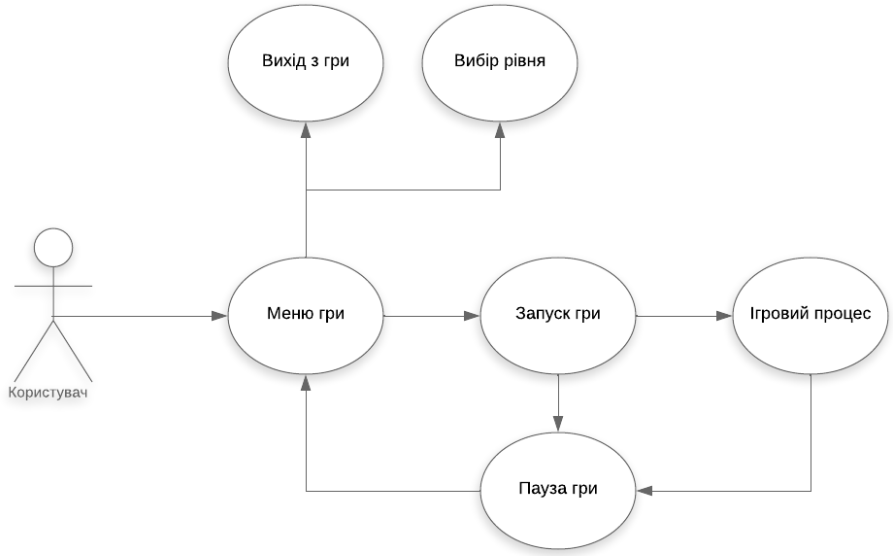


Рисунок 3.3 - UML діаграма прецедентів

Ця діаграма зображує, що користувач може:

– Меню гри, це коли після запуску самої гри побадаємо на стартове меню

– Вихід з гри, тобто користувач будучи в головній меню може вийти з гри та припинити тим самим роботу програми.

– Вибір рівня в грі, означає що можна обирати з якого рівня поинати гру

– Запуск гри, тобто старт гри, загрузка локаціїї з якої починає гравець

– Пауза гри, надається гравцю для того щоб міг призупинити гру

– Ігрвий процес, означає, що гравець може робити все що дозволено в грі, тобто грати.

Загальний алгоритм гри являє собою послідовність того, як користується користувач. Гравцеві перед тим як почати гру може наприклади одразу вийти з гри, або обрати рівень гри. Також користувач може одразу розпочати гру. Ігровий пароцес являє собою усе що може робити в грі, тобто стриляти, знищувати ворога, та дістатитися наступного рівня. Під час гри гравець може поставити гру на паузу, і далі зявится окно де гравець може війти з гри або продовжити гру тобто зняти з паузи.(Див. рис. 3.4)

Рисунок 3.4 - Алгоритм роботи програми

В процесі розробки програми потрібно читко бачити звязок між всіма класама, інакше кажучи ієрархія класів (Див. рис. 3.5)

Ієрархія класів, також звана таксономією класів, є групою пов'язаних класів, які пов'язані за допомогою успадкування для виконання аналогічних дій. Верхньою частиною ієрархії може бути один базовий клас, з якого отримані всі інші класи, що знаходяться нижче за нього, або ієрархія може мати кілька базових класів, функціональні можливості яких об'єднуються пізніше в один або кілька похідних класів. Відносини між класами можуть бути проілюстровані як дерева, і кожне менше дерево в рамках великої таксономії може розглядатися як ієрархія.

Рисунок 3.5 – Ієрархія класів

В програмі є основні класи таки як

- CharacterBehaviour цей клас являється класом батьківськм, який відовідає за функціональність персонажа
- Character клас який вже являється потомком, тобто приймає дані які знаходяться в класі батька
- Movement – клас який відповідає за рух та анімацію персонажа,

являється також класом потомком

- MovementBehaviour - клас являється батьківським класом відповідає за рух та анімацію персонажа,
- WeaponBehaviour клас являється батьківським, відповідає за логіку зброї та її анімацію
- Weapon клас який вже являється потомком відповідає за логіку зброї та її анімацію
- MagazineBehaviour клас являється батьківським, відповідає за логіку магазина до зброї.
- Magazine клас який вже являється потомком, відповідає за логіку магазина до зброї.

3.3. Реалізація гри

Для реалізації гри спочатку створюємо проект в Unity3D. Переходимо на вкладку Projects далі обераємо в правому кутку вкладку New project(Див. рис. 3.1)

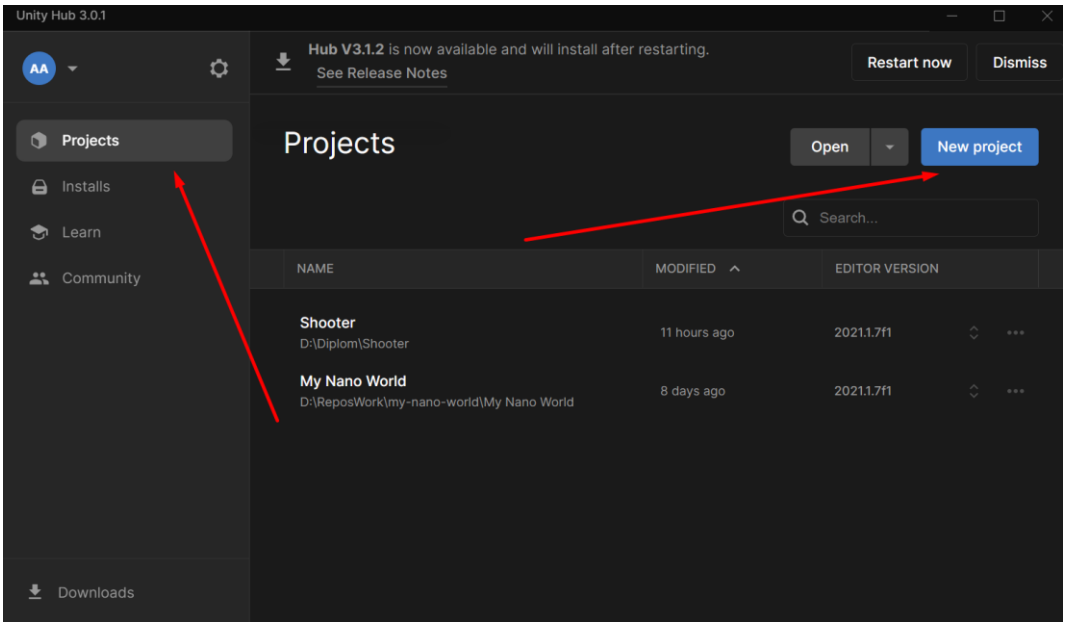


Рисунок 3.1 – Створення проекту

Далі буде запропоновано обирати який саме проект створити, тобто це може бути як 2D розуміючи, якщо планується робити гри в 2D форматі. Але для мого проекту потрібно саме 3D, тому обираємо, тобто натискаємо на вкладку 3D. Після цього у лівому нижньому вікні є тестове поле під назвою Project name, тут

потрібно обрати ім’я проекту. Під ним є вже вкладка Create project, далі буде генеруватися проект це займає декілька хвилин. (Див. рис. 3.2)

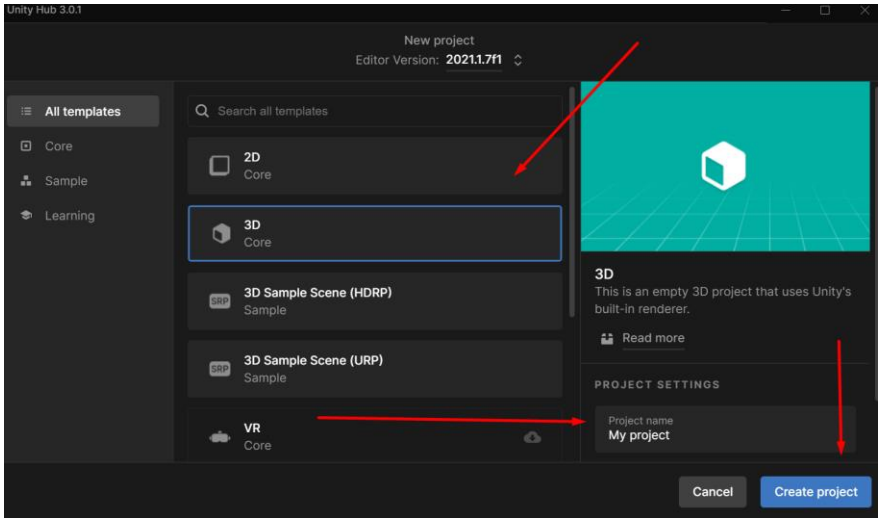


Рисунок 3.2 – Настроювання проекту

Коли проект запустився то бачимо, як великий іструментарій юніті.

Спочатку юніті потрібно трихи налаштувати, а саме підключити до рушія інтегровану среду розробки Rider. На верхній з лівого боку панельки обираємо Edit -> Preferences. Далі з’являється вікно Preferences де далі потрібно обрати External Tools де в першому рядку обираємо який редактор хочемо використовувати, в цьому випадку буде Rider. (Див. рис. 3.3)

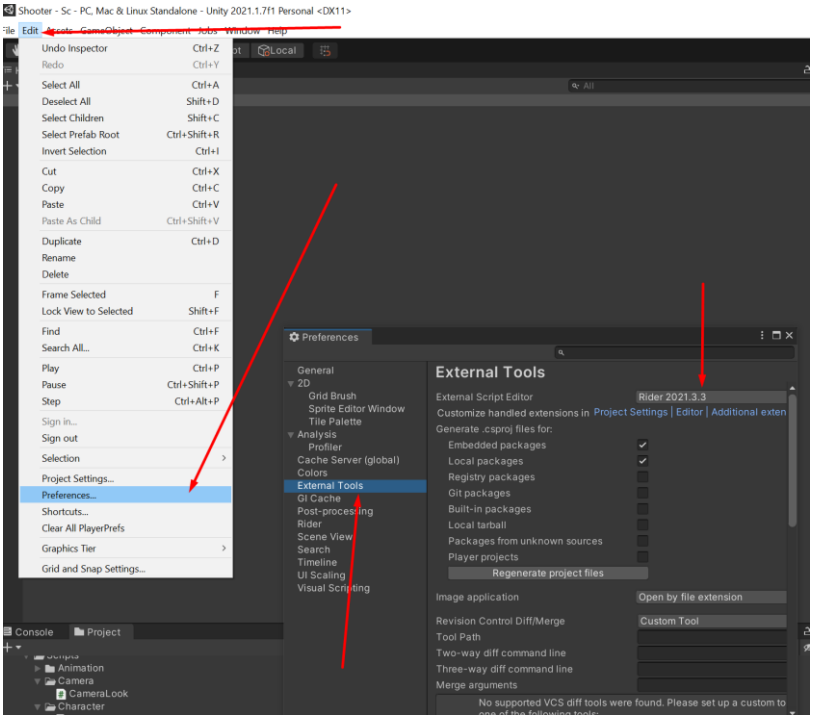


Рисунок 3.3 – Підключення до среди розробки Rider

У мого проєкту використовуються ігровий ассет який було створено у Blendery.

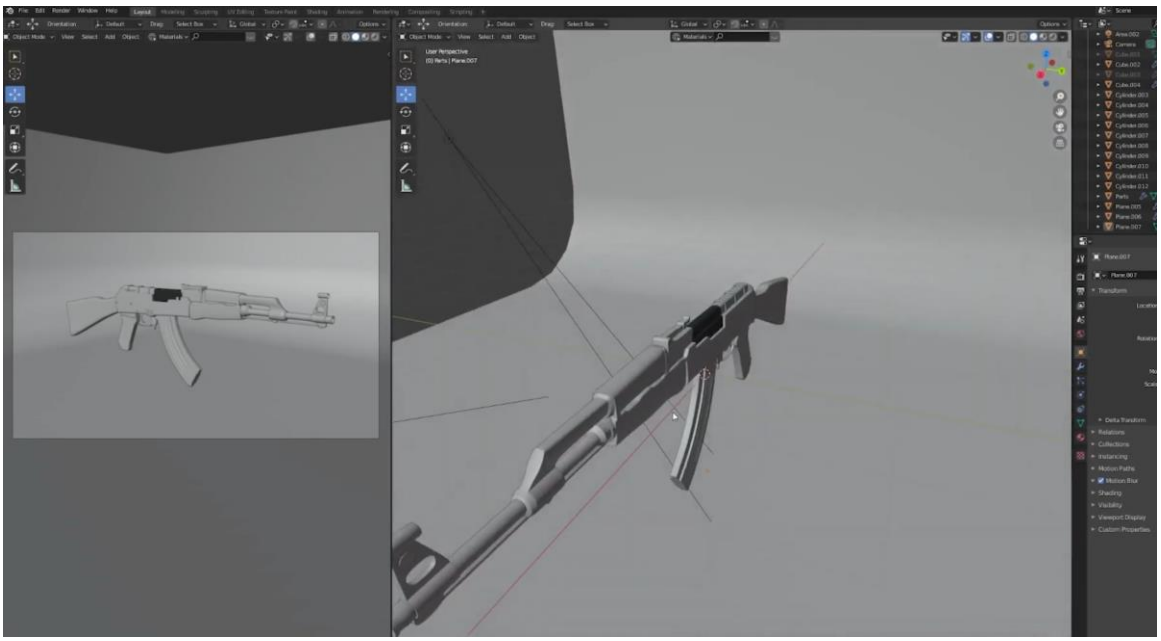


Рисунок 3.4 – Вигляд Blender

Після не великої підготовки, настав час писати скрипти. Для того щоб писати треба спочатку створити файл, потрібно в нижньому кутку екрану де знаходяться автоматично створені файли натиснуту в цьому області ПКМ, далі C# Script, після чого створиться файл скрипта.

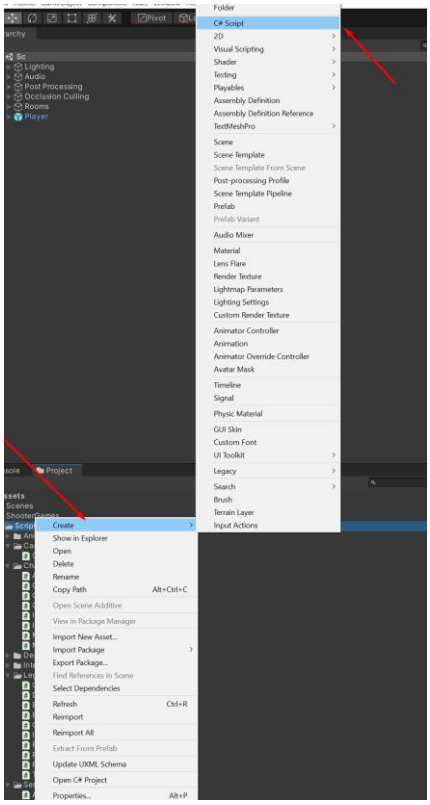


Рисунок 3.5– Створення скриптів.

Тепер розглянемо деклька скриптів більш детально. Наприклад клас Movement а саме функцію MoveCharacter. private void MoveCharacter()

```
{

    Vector2 frameInput = playerCharacter.GetInputMovement();
    var movement = new Vector3(frameInput.x, 0.0f, frameInput.y);

    if(playerCharacter.IsRunning())
        movement *= speedRunning;
    else
    {
        movement *= speedWalking;
    }

    movement = transform.TransformDirection(movement);


    Velocity = new Vector3(movement.x, 0.0f, movement.z);
}
```

Ця функція відповідає саме за рух нашого підконтрольному гравцю ігрового аватара. В цій функції використовуєть Vector2 де ініцілізується під назвою поле frameInput де далі вже привласнюємо об’єкт нашого ігового аватара. Vector2 потрібен для того щоб ми могли відслідковувати поточні координати нашого

ігрового об’єкта «X» та «Y»

Далі бачимо поле movement який вже потрібний для саме для руху ігрового об’єкта, але так як гра являється 3D то привласнюємо нову координату.

Також в методі використовується оператор if та else, в них бачимо такий рядок playerCharacter.IsRunning(), це означає що умова являється така, якщо ігровий об’єкт у стані ходьби то ми мусимо брати поле movement *= speedWalking, тобто звичайна ходьба. А якщо біжимо то movement *= speedRunning;

Після того як виконалось одне із умов, далі ми вокиристовуємо transform та Velocity це для того щоб будо відчуття невеликого спротиву коли вже об’єкт почав рухатися.

Тепер розглянимо інший клас та метод, який віповідає за стрілбу, клас називається Weapon, а метод Fire.

```
public override void Fire(float spreadMultiplier = 1.0f)
{
   if (muzzleBehaviour == null)
      return;

   if (playerCamera == null)
      return;

   Transform muzzleSocket = muzzleBehaviour.GetSocket();

   const string stateName = "Fire";
   animator.Play(stateName, 0, 0.0f);

   ammunitionCurrent = Mathf.Clamp(ammunitionCurrent - 1, 0, magazineBehaviour.GetAmmunitionCurrent());
   last++;

   muzzleBehaviour.Effect();

   Quaternion rotation = Quaternion.LookRotation(playerCamera.forward * 1000.0f - muzzleSocket.position);


   if (Physics.Raycast(new Ray(playerCamera.position, playerCamera.forward),
      out RaycastHit hit, maximumDistance, mask))
      rotation = Quaternion.LookRotation(hit.point - muzzleSocket.position);
```

```
        GameObject projectile = Instantiate(prefabProjectile, muzzleSocket.position, rotation);

        projectile.GetComponent<Rigidbody>().velocity = projectile.transform.forward * projectileImpulse;
    }
```

Спочатку цього метода є пара умов, тобто 2 оператора if.

Перший оператор відпоідає за те щоб у гравця точно була зброя, якщо такої зброї не має то далі вже код не буде виконуватися, бо сенсу немає, якщо нема зброї.

Наступна умова це перевірка на те що камера гравця точно висить на об’єкті бо без неї, інакше ми насправді не маємо можливості виконувати трасування.

Transform muzzleSocket – це звідки буде починатися постріли.

Наступних рядках створюємо стрінгове поле для того щоб зберігати необхідне слово, або інакше кажучі ключ, для того щоб спрацювала анімація.

ammunitionCurrent це поле яке зберігає певну кількість боєзапасу у нашому магазині. Коли стриляємо нам потрібно позбутися одного патрону.

У наступних рядках виконується ефекти.

Quaternion rotation – цей рядок визначає обертання, в яке ми хочемо вистрілити наш снаряд.

У наступних рядках іде поява самої кулі.

Після того як описали скрипти та методи потрібно для своєго об’єкта або інакше кажучи префаба через драген дроп прикрипити наш скрипт, та добавити інші необхідні компоненти.

Необхідно обов'язково приєднати таки компоненти як Rigibody, це компонент відповідає за фізику ігрових об’єктів.

Також потрібно обрати компонент Capsule Collider, це також необхідний об’єкт без якого неможливо наприклад стояти на поверхності землі.

Наступний компонент Audio Source – цей компонент необхідний для відтворення звуків.

Нуступні компоненти вже являються створені сркрипти, в них потрібно трихи налаштувати. (Див. Рис. 3.7)

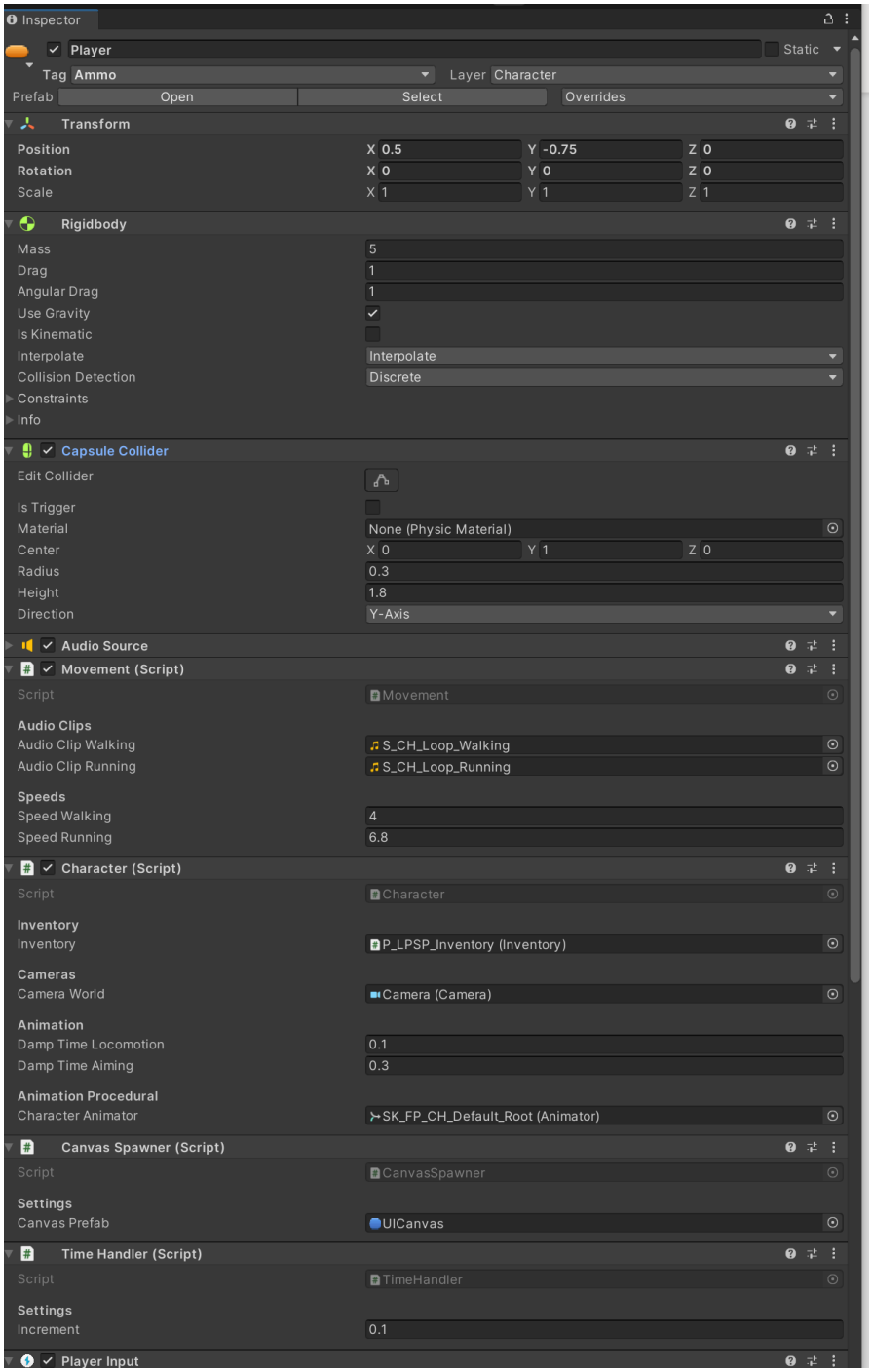


Рисунок 3.6– Вигляд інспектора префаба Player.

Тепер розгянимо як зробити в грі інтерфейс, для цього ми повині створити Canvas. Для того щоб створити канвас потрібно на полі сцені клацнути ПКМ.

Далі обрати Ui потім Canvas.(Див. Рис. 3.8)

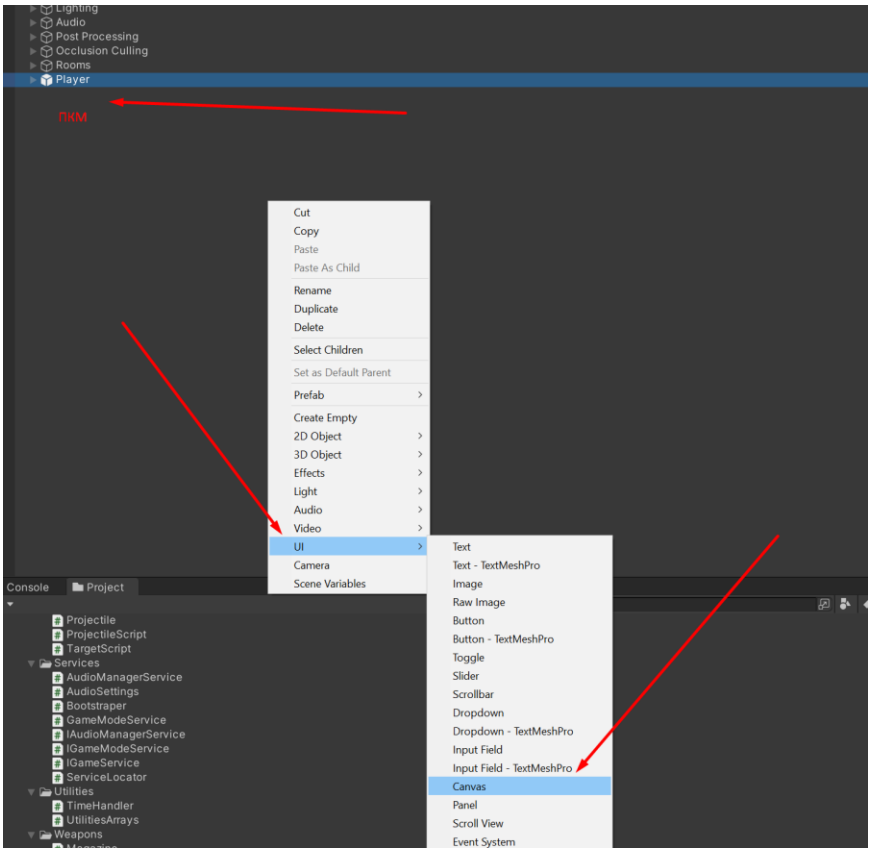


Рисунок 3.7 – Створення Canvas

Після створення канваса ми робимо з нього префаб тобто драген дропеємо до папки проекту. Після чого відкриваємо прифаб канвас та налаштовуємо.

В іграх, як шутер повинен бути максимально простий та зручний інтерфейс, який не буде заважати гравцеві поринати в гру. Основними елементами які потрібно зробити, це відображати кількість боєзапасу, та перехрестя, щоб гравець міг розуміти як куди саме полетить куля.

Також треба не величкий туторіал зробити для новеньких гравців, щоб могли розуміти яка клавіша, та за що відповідає.

Але просто показувати, що в тебе боєзапас не достатньо потрібно, щоб коли гравець робив постріл потрібно, щоб інтерфейс міг відновлюватися кожного разу як буде зролебний вистріл та показував атульну інформацію.(Див.рис.3.9)

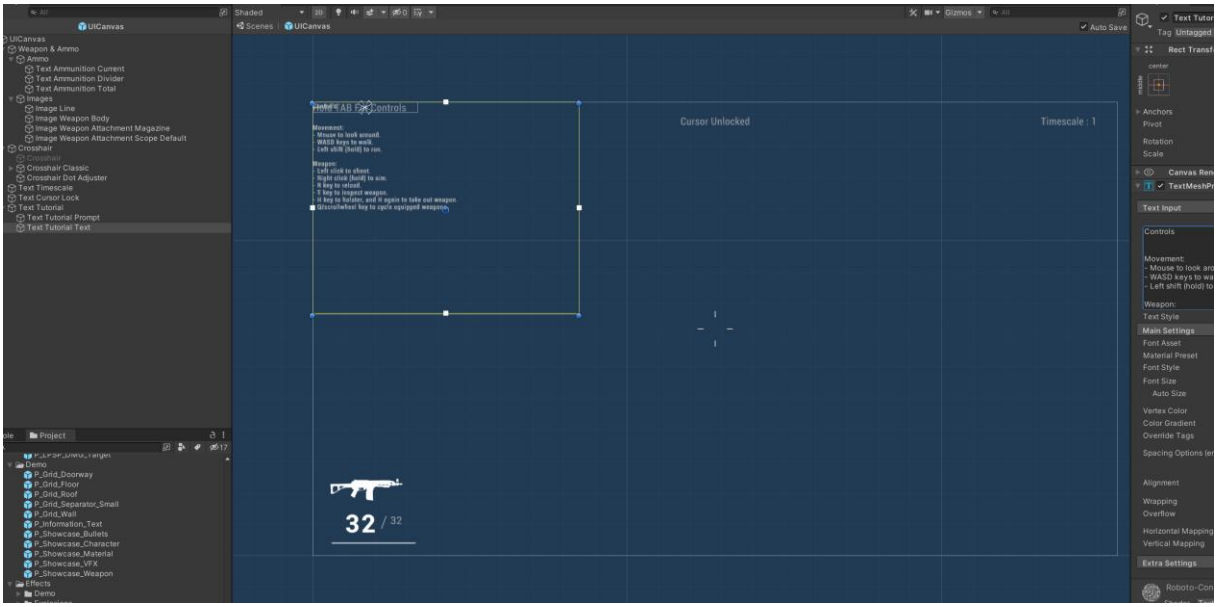


Рисунок 3.8 – Вигляд реалізації інтерфейсу.

4. Представлення розробки гри

4.1. Скріншоти гри

Після того як творили основні елементи гри можна спокіно демонстувати.

Рисунок 4.1 – Меню гри.

Рисунок 4.2 – Вибір рівня.

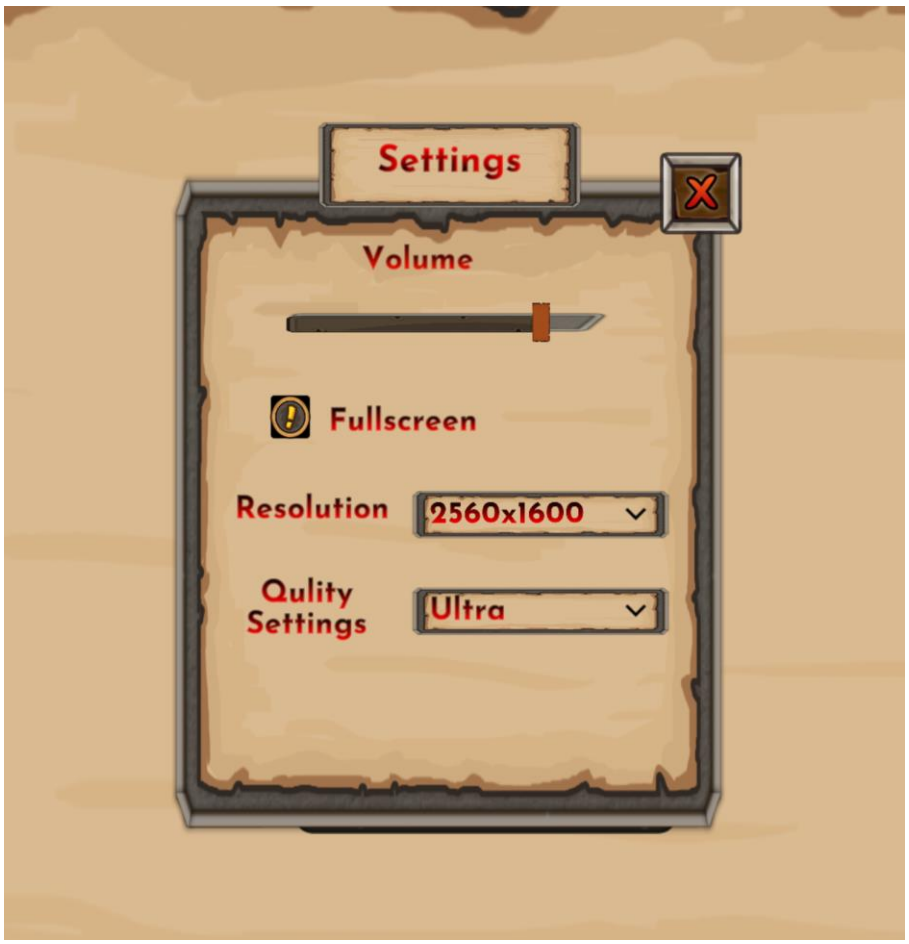


Рисунок 4.3– Налаштування графіки.

Рисунок 4.4 – Ігровий процес.

ВИСНОВКИ

Дана робота була спрямована на проектування та реалізацію гри «SHOOT AND RUN» у жанрі шутер. Було обгрунтовано актуальність розробленої гри та її наукову новизнушляхом аналізу використання рушія Unity3D. Було досліджено типові ігри у жанрі шутер, де виявили, що для розробки гри використовують різноманітні рушії, але досить рідко хто робив шутер саме на Unity.

1. Було розроблено гру shoot and run , де було реалізовані основні механіки гри, де гравець може рухатися в любому напрямку, використовувати свою зброю, поповнювати боєзапас, знищувати ворогів.

Було вибрано інструменти для розробки гри, такі як Unity3 та Rider обгрунтовано їх перевагу та ефективність над іншими аналогами.

Для того щоб розробити успішний продукт, було проаналізовано різні ігри у своєму жанрі, та створено набір вимог у вигляді UML діаграми.

В ході розробки гри було спроектовано та реалізовано MVC алгоритм, де було читко виділено принцип її роботи.

2. Описано програмні засоби та інструменти, котрі було застосовано для розробки програмного забезпечення. Виявлено що середовище розробки Rider є найбільш зручним для виконання поставлених задач.

3. Високу якість реалізації коду завдяки використання абстрктного класудозволило досить ефективно та досить гнучно реалізовувати гру.

4. Розроблена система інтерфейсу UI дозволило читко бачити необхідну гравцеві інформації, яка потрібна для розуміння, як далі надходити.

5. Роботу було апробовано серед висококваліфікованих спеціалістів в сфері Game development, у апробації взяли участь 3 осіб з рівнем знань та досвідом від Middle до Senior. Було проаналізовано та виявлено ряд переваг та недоліків системи. Також було розроблено план покращень та розширень механік гри.Виявлено, що основнимиперевагами гри є:

- Простота керування персонажем;

- Стабільність гри та відсутність багів або критичних помилок;
- Продуманий користувацький інтерфейс;

Результати дослідження бакалаврської роботи апробовані на всеукраїнських науково-технічних конференціях: "Застосування програмного забезпечення в інфокомунікаційних технологіях

ПЕРЕЛІК ПОСИЛАНЬ

1. Research shows playing first-person shooters improves learning abilities, cognitive function – [Електронний ресурс] – Режим доступу: https://www.polygon.com/2013/1/30/3932876/research-playing-first-person-shooters-improves-learning-abilities-cognitive-function

2. Жанри відеоігор – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%96%D0%B0%D0%BD%D1%80%D0%B8_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80

3. Reasons Video Games Are Good For Your Health – [Електронний ресурс] – Режим доступу: https://www.forbes.com/sites/jordanshapiro/2013/11/27/4-reasons-video-games-are-good-for-your-health-according-to-american-psychological-association/?sh=3452e7003a00

4. Metanit – [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/tutorial/1.1.php

5. Unity User Manual – [Електронний ресурс] – Режим доступ https://docs.unity3d.com/2021.1/Documentation/Manual/index.html

6. The First 5 Principles of Object Oriented Design – [Електронний ресурс] – Режим доступ https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

7. Douglas Crockford. JSON: The Fat-Free Alternative to XML, 2009 – 55 p.

8. MVC: Model, View, Controller – [Електронний ресурс] – Режим доступ https://www.codecademy.com/article/mvc

9. Unity Documentation – [Електронний ресурс] – Режим доступ https://docs.unity3d.com/ScriptReference

10. Doom Eternal Wiki Guide – [Електронний ресурс] – Режим доступ https://www.ign.com/wikis/doom-eternal/

11. Counter-Strike: Global Offensive (for PC) – [Електронний ресурс] – Режим доступ https://www.pcmag.com/reviews/counter-strike-global-offensive-for-pc

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ГРИ «SHOOT AND RUN» ЖАНР ШУТЕР МОВОЮ C#

Виконав студент 4 курсу

групи ПД-43
Косенко Андрій Павлович
Керівник роботи

Доктор філософії Дібрівний О.А.

Київ – 2022

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – вдосконалення існуючих механік гри жанру шутер.

Об’єкт дослідження – процес розробки відеоігор з використанням двигуна Unity

Предмет дослідження – розробка комп’ютерної гри жанру шутер на двигуні Unity.

Порівняння з аналогами

Назва продукту	Переваги	Недоліки
CS GO	1. Реалізація стрільба від стегна 2. Реалізація звуків 3. Реалізація механіки тихий ходьби	1. Реалізація механіка стрільби 2. Реалізація механіки прицілювання 3. Реалізація механіки бігу
DOOM Eternel	1. Реалізація різноманітність зброї; 2. Реалізація механіка стрільби; 3. Реалізація механіка руху	1. Реалізація меню гри; 2. Реалізація звуків; 3. Реалізація присідання.

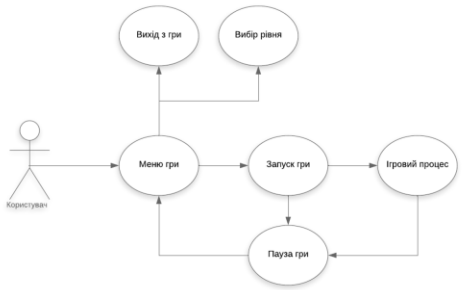
ТЕХНІЧНІ ЗАВДАННЯ

1.Розробити гру на двигуні Unity3D мовою C#
2.Розробити механіку ходьби персонажу
3.Розробити механіку бігу персонажу
4.Розрозробити механіку прицілювання
5.Розробити механіку стрільби від стегна
6.Розробити меню гри
7.Розробити налаштування гри

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

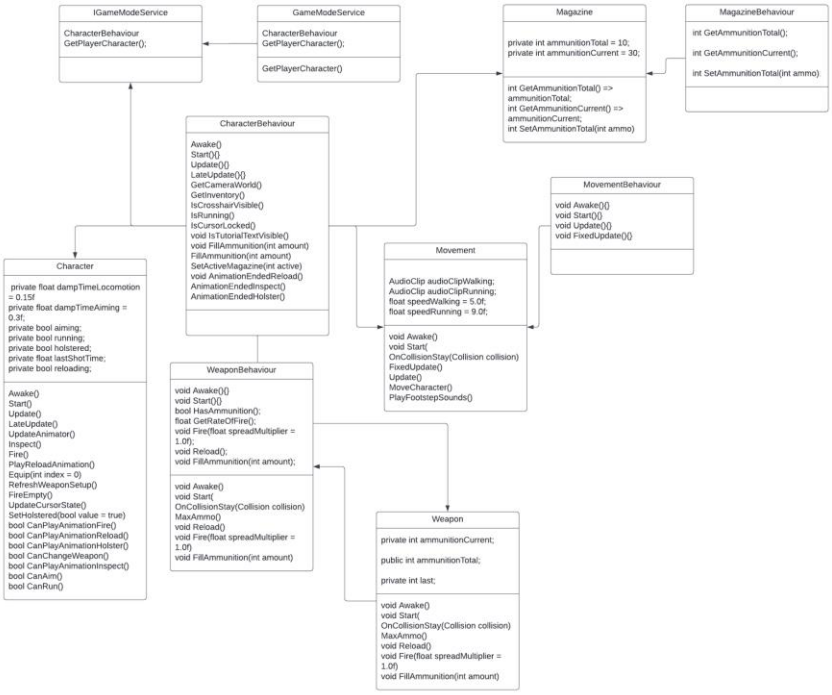
- *Unity3D* - міжплатформне середовище розробки комп'ютерних ігор. Unity дозволяє створювати програми, що працюють під більш ніж 20 різними операційними системами, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші.
- *Rider* - інтегрована среда розробки (Integrated Development Environment — IDE) JetBrains дозволяє створювати додатки для Windows, веб-приклади та мобільні додатки, як і Microsoft Visual Studio.
- *Blender* - це вільне та відкрите програмне забезпечення, яке використовується багатьма художниками графіки та VFX, щоб донести свою уяву до візуального реалістичного світу на екрани в 2D та 3D комп'ютерній графіці.

Діаграма прецедентів



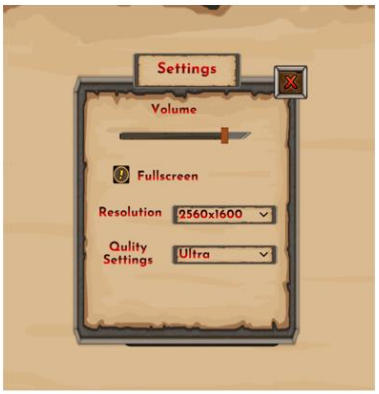
Діаграма алгоритмів

Діаграма класів



Приклад роботи проекту

Приклад роботи проекту



Приклад роботи проекту

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Косенко А.П. Дослідження ігорового жанру шутер: Матеріали науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез.\- 20.04.2022, ДУТ, м.Київ, ст 106-107

ВИСНОВКИ

1. Обґрунтовано актуальність гри та порівняно з іншими аналогами гри певного жанру.
2. Спроектовані та дослдіжені основні вимоги до гри.
3. Описані програмні засоби та інструменти, котрі було застосовано для розробки програмного забезпечення.
4 . Розроблено гру та ключові її механіки.

ДЯКУЮ ЗА УВАГУ!