МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-06

Дипломний проект

здобувача освіти денної форми навчання
РП.06.18.000.ДП

ПОПОВА ДЕНИСА СЕРГІЙОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітня програма: «**Розробка програмного забезпечення»**

Група: **4РП-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter.

Проектний матеріал складається з пояснювальної записки на 76 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Попов Д.С.)

Керівник _____ (Джабраілов Д.В.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кунуп Т.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «____»_____ 2023 р. Протокол ДКК № 1

Оцінка ДКК 5 (відмінно)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНАХТ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____ 2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти *Попов Денис Сергійович*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter*

затверджена наказом по коледжу від "17" жовтня 2023р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) 09.06.2023

3. Вихідні данні до проекту (роботи)

1. Використання ігрового двигуна Unity;

2. Використання принципів модульної розробки ігор;

3. Використання мови програмування C# та бібліотек Unity для розробки гри;

4. Реалізація логіки ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter;

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

Опис предметної області. Огляд існуючих рішень у ігровій індустрії. Загальний технічний опис вирішення задачі дипломного проекту. Огляд інструментів та програмних рішень для виконання дипломного проекту. Процес створення макету 2D-гри у жанрі top down shooter. Проектування роботи логіки для ігрових об'єктів та елементів інтерфейсу гри. Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter.

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

Особливості ігрового жанру top down shooter; Особливості ігрового двигуна Unity; Етапи створення макету 2D-гри в жанрі top down shooter; Принцип роботи модулів логіки ігрових об'єктів проекту; Принцип роботи модулів роботи елементів інтерфейсу проекту; Етапи розробки логіки для ігрових об'єктів та елементів інтерфейсу; Скріншоти результатів розробки проекту;

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	*Джабраілов Д.В.*		
розділ			
2.Екон. частина	*Копайгородська Т.Г.*		
3.Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачі завдання 24.04.2023

Керівник *Джабраілов Д.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	*Вступ. Постановка мети та задач проектування*	*17.05.2023*	Виконав
2.	*Аналіз предметної галузі*	*18.05.2023*	Виконав
3.	*Огляд інструментів та програмних рішень для*	*19.05.2023*	Виконав
	виконання дипломного проекту		
4.	*Створення макету 2D-гри у жанрі top down shooter*	*23.05.2023*	Виконав
5.	*Проектування роботи логіки для ігрових об'єктів та*	*25.05.2023*	Виконав
	елементів інтерфейсу гри		
6.	*Реалізація логіки для ігрових об'єктів та елементів*	*28.05.2023*	Виконав
	інтерфейсу гри		
7.	*Тестування роботи створених модулів*	*31.05.2023*	Виконав
8.	*Робота над Економічною частиною*	*2.06.2023*	Виконав
9.	*Робота над Охороною праці*	*4.06.2023*	Виконав
10.	*Підготовка матеріалів до захисту*	*6.06.2023*	Виконав
11.	*Попередній малий захист*	*12.06.2023*	Виконав
12.	*Проведення захисту дипломного проекту*	*19.06.2023*	Виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 06. 18 000. 00 ДП	Дипломний проект		
А4			РП 06. 18 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 06. 18 000. 00 ДП		
Розробив		Попов Д.С.			Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри в жанрі top down shooter	Літ.	Аркуш 5 / Аркушів 76
Перевірив		Джабраілов Д.В.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4РП-06	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Актуальність теми. Для повноцінної роботи людини у суспільстві їй необхідно задовольняти основі потреби, як їжа, сон та відпочинок. Такий відпочинок може приймати різні форми і одною із таких форм є гра в цифрові ігри.

Комп'ютерні ігри пройшли великий шлях, від нішевої галузі IT-ринку, до великого кластеру останнього та створення цілої індустрії. В наш час цифрові ігри мають велику жанрову, структурну, апаратну та цільову диференціацію, але одна з основних функцій ігор залишається незмінною – комфортне проведення часу під час відпочинку.

Процес створення таких цифрових продуктів має певні складові, якість виконання котрих напряму впливає на якість відпочинку людини. Темою даного дипломного проектування є «Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри в жанрі top down shooter» і саме реалізація цих складових має важливий вклад у створення якісного цифрового ігрового продукту.

Крім того, актуальність теми дипломного проектування зумовлена і тим, що процес створення зазначених в темі складових є типовим для ігор різних жанрів, отже робота по реалізації цих елементів – це вклад у відточення процесів розробки цифрових ігор інших жанрів. Реалізація зазначених в темі складових, дасть змогу краще зрозуміти процес розробки цифрових ігор, а також створити шаблони та модулі для подальшої розробки інших проектів такого типу.

Також, виконання теми дипломного проекту допоможе більш детально дослідити принципи розробки цифрових ігор за допомогою актуальних програмних рішень та технологій. Такий підхід дозволить у подальшому створювати рішення на сучасній технологічній основі, не відриваючись від технологій розробки ігор.

Основною метою роботи є реалізація важливих складових будь-якого ігрового продукту, як логіка ігрових об'єктів та елементів інтерфейсу.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Опис предметної області

Тема дипломного проектування включає в собі складові предметної області розробки цифрових ігор. Така область включає в собі ряд складових:

- Визначення жанрів цифрових ігор;
- Визначення технологій для розробки цифрових ігор;
- Визначення основних складових ігрових механік цифрової гри;
- Визначення із способами реалізації цих складових.

Жанри в ігровій індустрії представлені дуже величезним спектром, який має своє коріння від перших ігор для ігрових автоматів, та ще далі, до «забав» програмістів зі створення ігор на осцилографах. В наші дні жанри поділяють в залежності від платформи, на якій буде виконуватись гра, а також основних ігрових механік та зовнішнього виду ігрового процесу. Обрання ігрового жанру грає велику роль не тільки в плані зовнішніх проявів, але й у питанні використання тих, чи інших ігрових механік.

У випадку із темою дипломного проектування, тема чітко визначає жанр, що дає змогу вибрати елементи ігрової логіки об’єктів та елементів інтерфейсу. У класичному розумінні 2D top down shooter-у очікуються такі основні елементи оточення, як рівень із перспективою «зверху-вниз», об’єкти «аптечки», об’єкти «бронювання», об’єкти «набоїв», об’єкти «ворогів», а також активні зональні об’єкти, які б могли виконувати послуги для обслуговування гравця. Те, які об’єкти використовувати, залежить від задуму гри.

Визначення технологій для розробки, в основному полягає у знаходженні відповідного ігрового двигуна, а також інструментів розробки. До їх вибору можна підходити із декількох сторін, але самим важливим є обрання ігрового двигуна для гри. Раніше це було великою проблемою, адже доступних для використання пересічним користувачем ігрових двигунів майже не було. В наш

час є вибір із рішень різного виду ліцензії та технологій.

Визначення попередніх питань дає змогу визначитись із тим, що можна реалізувати, опираючись на обраний жанр та технології. Ігровими механіками називають елементи ігрового процесу до якого залучається гравець – яким чином, та для чого гравець виконує ті чи інші дії і є ігровими механіками.

В загальному розумінні, визначення із способом реалізації полягає у архітектурі ігрового проекту. У значній мірі це диктується ігровим двигуном, але в рамках його архітектури можна будувати проекти, із використанням принципів модульної розробки, коли ви зможете використовувати елементи одної гри у розробці іншої гри. Такий підхід допомагає в подальшому зменшити витрати часу та ресурсів на створення нової цифрової гри. В наш час саме модульний підхід є основним під час розробки ігрових проектів, тому часто можна знайти в коді гри елементи попередніх проектів розробників.

1.2 Огляд існуючих рішень у ігровій індустрії

Для жанру 2D top down shooter є класичній набір ігрових об’єктів з якими можна взаємодіяти. Як було зазначено вище, такими об’єктами частіше за все є такі, що допомагають гравцю далі рухатись по ігровому процесу.

Рисунок 1.1. 2D top down shooter

Основним та незмінним способом для реалізації таких об’єктів є відображення їх у ігровому просторі. Частіше за все це робиться в одному і тому самому стилі та кольоровій гамі. Це робиться для того, щоби гравець запам’ятав цей шаблон та за необхідністю знав зо саме йому шукати. Також, вважається вірним виконувати такі елементи у стилістиці того «ресурсу» на який сприяє такий об’єкт. Тобто, якщо це «аптечка», що відновлює показники здоров’я, то вона має бути або у виді червоного плюсу, або такого кольору, як показник здоров’я гравця.

Принцип взаємодії дуже простий. Частіше за все, в цифрових іграх, для взаємодії із таким об’єктом достатньо «стати» на нього, тобто об’єкт гравця має перетнутись із об’єктом для взаємодії. Гра автоматично виконає поповнення необхідного «ресурсу» гравця. Після взаємодії, частіше за все, такі об’єкти видаляються з ігрового поля. Приклади таких об’єктів, з якими можна взаємодіяти можна переглянути на рисунках 1.2, 1.3, 1.4:

Рисунок 1.2. Приклад реалізації об’єкту для взаємодії із гравцем №1

Рисунок 1.3. Приклад реалізації об’єкту для взаємодії із гравцем №2

Окремо стоять такі об’єкти, які не видаляються із гри та дають змогу гравцю виконувати безлімітне, або багаторазове користання. Наприклад, такими об’єктами можуть бути «ремонтні станції», або «станції медичної допомоги». Також, таким чином можна реалізовувати ігрові магазини, для поповнення ігрових ресурсів.

Рисунок 1.4. Приклад реалізації об’єктів для взаємодії із гравцем №3

Спосіб взаємодії із таким об’єктом частіше за все не відрізняються від одноразових – гравцю достатньо перетнути такий об’єкт і він почне працювати. Якщо це магазин, то відкриється панель для ігрових покупок, якщо це зона захвату, то почнеться захват. Приклад такого об’єкту можна побачити на рисунку 1.5:

Рисунок 1.5. Приклад об’єкту для багаторазової взаємодії

Нестандартні підходи до реалізації таких об’єктів зумовлені геймдизайнерським задумом. Інколи, необхідно натиснути якусь кнопку, або мати який-небудь інший елемент із гравцем, або необхідна взаємодія спочатку із іншим об’єктом.

Щодо елементів інтерфейсу та взаємодії із ними частіше за все, в жанрі 2D top down shooter гравцю відображають основні «ресурси» для виконання ігрового процесу та його показник «здоров’я». Ці елементи можуть виконуватись, як в виді букв та чисел, так і в інших проявах, як панелі, повзунки, прогресбари.

Часто взаємодія із ними доволі обмежена. Гра сама виконує вилучення показника «здоров’я», або набоїв. Більш складний ігровий процес може викликати потребу від розробника відображати текстові повідомлення, панелі для взаємодії та кнопки. В класичному же розумінні жанру, гравцю достатньо тільки показник «здоров’я» та набоїв.

1. 3 Загальній технічний опис вирішення задачі дипломного проекту

Для вирішення поставлених задач, необхідно виконати розробку та імплементацію логіки взаємодії для ігрових об’єктів а також елементів інтерфейсу.

Керуючись принципами розробки розглянутими раніше, а також задумом ігрового проекту, можна виділити декілька об’єктів із якими буде виконано систему взаємодії. Такими будуть об’єкти для ремонту техніки гравця (рисунок 1.7), а також охолодження його зброї (рисунок 1.8). Об’єктами для довгого, або багаторазового використання будуть виступати пункти ремонту (рисунок 1.9) та охолодження зброї (рисунок 1.10).

Оскільки проект знаходиться на початковій стадії розробки, візуальна реалізація таких елементів буде виконана за допомогою складних примітивів, ігровим двигуном.

Програмна частина буде реалізована за допомогою внутрішніх скриптів ігрового двигуна (рисунок 1.6), написаних на мові програмування C#.

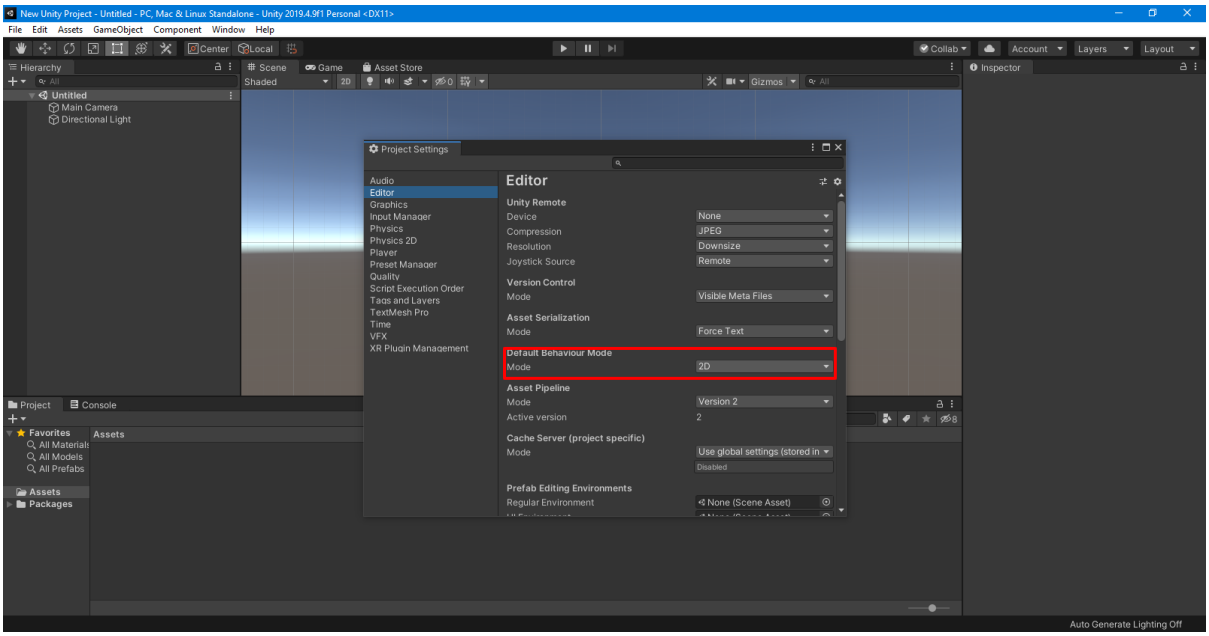


Рисунок 1.6. Приклад вікна програмування

Схема взаємодії із об’єктами буде виконуватись у вигляді перетину

об’єкту та автоматичної взаємодії із ним.

Слід зазначити, що ці об’єкти будуть нести в своєму візуальному виконанні кольорові схеми для швидкого визначення приналежності до відповідного ігрового «ресурсу». Наприклад, все що відноситься до ремонту техніки гравця має сірий колір, який буде використовуватись у інтерфейсі для відображення стану техніки.

Окрім того необхідно врахувати необхідність використання принципів модульної розробки гри, таким чином, щоби створені об’єкти та код для них можна було використовувати у майбутніх розробках. Також, це дозволить опираючись на код створювати, за потребою, нові об’єкти, що будуть виконувати інші задачі. Реалізація таких об’єктів буде значно полегшена, оскільки необхідно буде внести лише невеликі зміни у код.

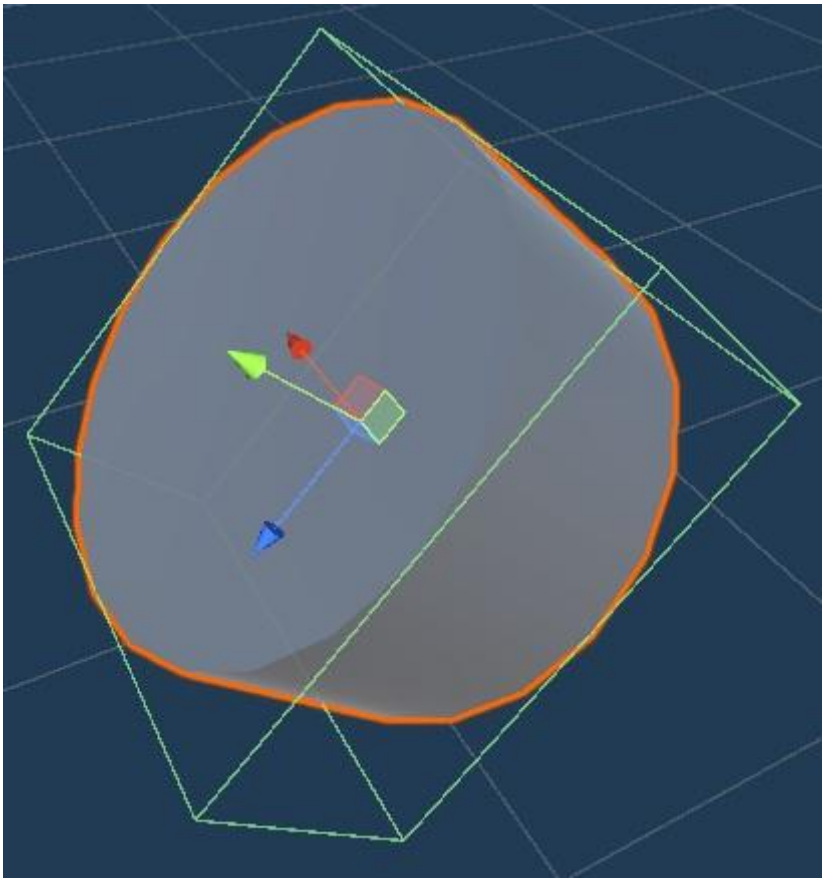


Рисунок 1.7. Приклад візуального виконання об’єкту для ремонту техніки гравця

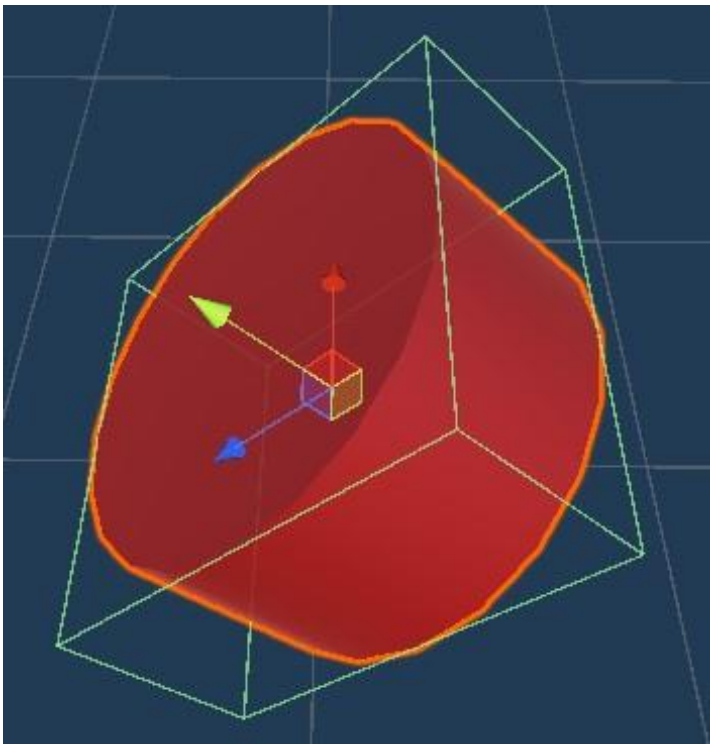


Рисунок 1.8. Приклад візуального виконання об’єкту для охолодження зброї гравця

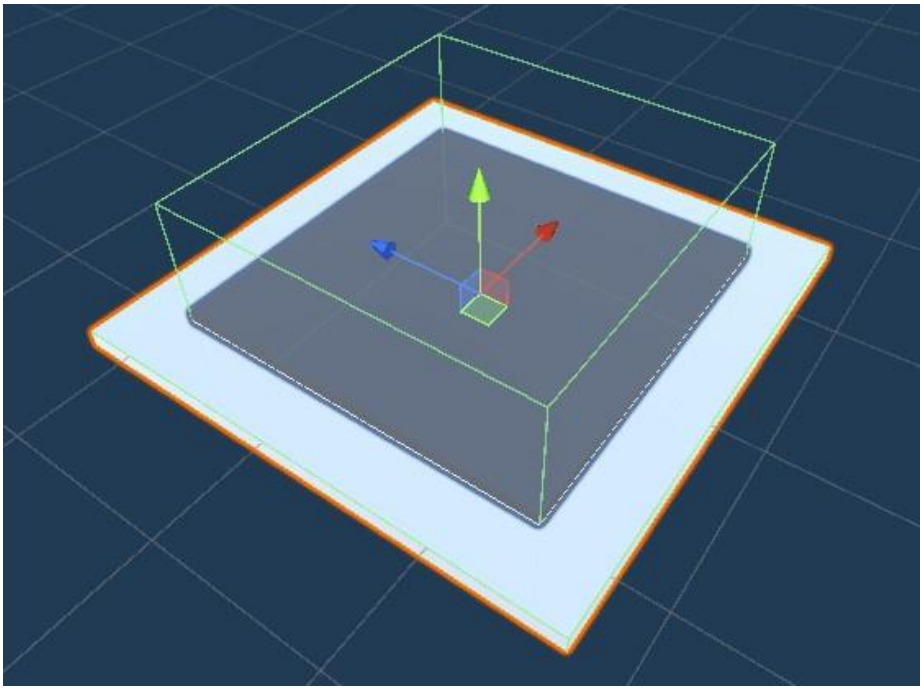


Рисунок 1.9. Приклад візуального виконання станції для багаторазового ремонту техніки гравця

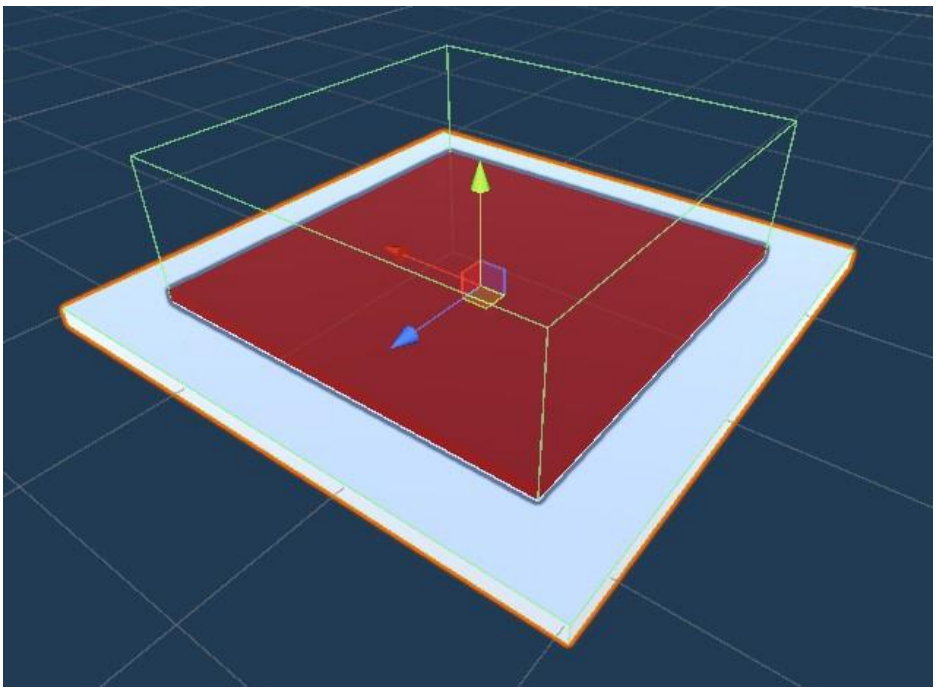


Рисунок 1.10. Приклад візуального виконання станції для багаторазового охолодження зброї гравця

Щодо виконання інтерфейсу, то робота із ним буде основана на обраному ігровому двигуні за допомогою його інструментів та принципів програмування на мові C#. Візуально інтерфейс гравця буде відображати два основних параметри гравця – технічний стан його техніки та перегрів зброї гравця. Один показник буде зменшуватись в залежності від отриманих ушкоджень, а другий навпаки збільшуватись під час використання зброї. Приклади вигляду цих елементів можна побачити на рисунку 1.11 та рисунку 1.12:

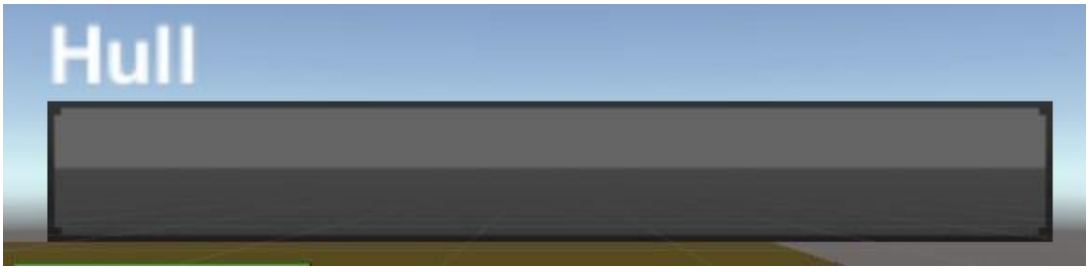


Рисунок 1.11. Приклад показника, який відповідає за технічний стан техніки гравця

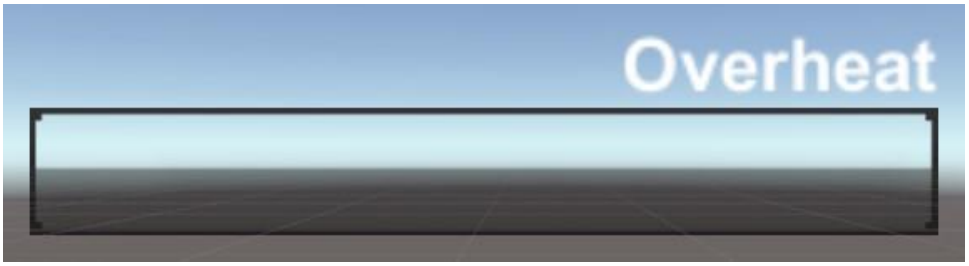


Рисунок 1.12. Приклад показника, який відповідає за перегрів зброї гравця

Окремо слід зазначити, що під час реалізації проекту, на його початкових етапах, у елементах інтерфейсу буде використовуватись англійська мова. Це зумовлено в першу чергу полегшенням процесу розробки через зменшення маніпуляцій із зміною мови, а також правилами локалізації ігор у основних платформах розповсюдження цифрових ігор. Ці правила говорять про обов'язкову присутність англійської мови у інтерфейсу гри. Тому, звичайно, ігри створюють спочатку на англійській мові, а потім виконують локалізацію на інші цільові мови.

1.4 Огляд інструментів та програмних рішень

Для виконання завдання дипломного проектування в першу чергу було обрано ігровий двигун, який буде лежати в основі гри. Ігрові двигуні є набором програмних функцій, методів, класів та бібліотек, що разом формують умови для створення цифрової гри. Раніше ігрові двигуни ділилися на два види: ігровий двигун та фізичний двигун. Перший відповідав за ігрові механіки, способи та особливості взаємодії між ігровими елементами, а також ігрову логіку. Фізичний двигун відповідав за симуляцію фізики в ігрових проектах. В ті часи, часто було можливо сказати, що гра А була створена на двигуні гри В. Із розвитком інструментів та технологій розробки цифрових ігор, ігрові двигуни еволюціонували і в наш час поєднують у собі ігрові механіки та фізичну взаємодію об'єктів на сцені.

Ігрові двигуни розповсюджуються за ліцензією, яка залежить від двигуна, або умов його використання. Деякі ігрові двигуни взагалі не розповсюджуються, інші мають відкритий код, або безкоштовну основу. Для виконання цілей дипломного проектування буде доцільно обрати один із умовно безкоштовних двигунів, яких в наш час достатньо.

Unity – це ігровий двигун, який надає сучасні технології в сфері ігрової індустрії у зрозумілій середі розробки для розробників буд-якого рівня знань та вмінь. Крім того, цей двигун відразу дає можливість створювати проекти буд-якої складності із використанням всього доступного в Unity інструментарію. Поєднує в собі, як програмну частину ігрових двигунів, так і власну фізичну модель, яку можна зручно налаштувати в залежності від ваших потреб. Крім того, Unity дає змогу створювати цифрові ігри у 2D та 3D графіці, а також комбінувати ці види візуалізації, розміщуючи 3D графіку ну 2D-просторі. Мова програмування, що використовується у двигуні – C#. На рисунку 1.13 можна бачити логотип ігрового двигуна:

Рисунок 1.13. Логотип ігрового двигуна Unity

Одним із плюсів обраного ігрового двигуна полягає в умовах його ліцензування. Так, можна використовувати персональну версію двигуна, або відразу отримати корпоративну цілком безкоштовно. Безкоштовне

використання двигуна залишається таким до тих пір, доки кількість річного прибутку не перевищить 100000$ та 200000$ відповідно до версій двигуна. Також, Unity залишається безкоштовним для цілей освіти на науки. На даний час на ігровому двигуні Unity створено велику кількість успішних ігор, як Pillars of Eternity (рисунок 1.14), Pathfinder: Kingmaker (рисунок 1.15), Escape from Tarkov (рисунок 1.16).

1.4.1 Pillars of Eternity

Це рольова відеогра, розроблена Obsidian Entertainment під видавництвом Paradox Interactive. Гра була випущена для платформ Microsoft Windows, OS X і Linux 26 березня 2015 року.

Рисунок 1.14. Гра на Unity Pillars of Eternity

У серії ігор Pillars of Eternity гравці беруть на себе роль персонажа, яких у ігровому всесвіті називають «Спостерігачами» (The Watcher) — особи, здатної бачити душі і взаємодіяти з ними, переглядати спогади померлих або маніпулювати душами живих — це працює за механікою партійних рольових відеоігор з активною паузою. Подібно до своїх духовних попередників, Baldur's Gate, Icewind Dale і Planescape: Torment — це ізометрична гра, де огляд ведеться

з фіксованої точки на 3D-моделі на двовимірних пререндерених фонах

Нова гра починається зі створення персонажа, причому вибір, який робиться щодо раси, походження, статистики та класу персонажа (окрім зовнішнього вигляду), впливає на те, які вибори можна зробити в діалогах з неігровими персонажами або інтерактивними об'єктами. Кожен клас з одинадцяти доступних отримує переваги від певних характеристик та має набір унікальних здібностей, який можна використовувати в битвах. На додаток до них персонаж може використовувати п'ять навичок — Скритність, Атлетика, Історія, Механіка і Виживання — які надають певні переваги в різних ситуаціях, наприклад взлом закритих контейнерів або отримання відповідних бонусів під час відпочинку партії. Класи персонажів і ігрова механіка мають деякі поверхневі схожості із системою Dungeons & Dragons, але взагалом є власною системою, створеною командою Obsidian. Персонажі підвищують свій рівень після отримання достатньої кількості очок досвіду, які видаються тільки за виконання квестів — битви з ворогами не приносить жодних очок досвіду, тобто гра пропагандує ненасильницький підхід у проходженні сюжетної лінії.

1.4.2 Pathfinder: Kingmaker

Pathfinder: Kingmaker - пригодницька рольова гра, події якої розгортаються в фентезійному всесвіті відомої серії настільних ігор Pathfinder.

Рисунок 1.15. Гра на Unity Pathfinder: Kingmaker

В грі є цікавий сюжет. Гравцеві належить зіграти за відважного шукача пригод, мета якого - вижити в світі, повному магії і зла. Гравець може обрати собі персонажа з чотирнадцяти унікальних класів. Не дивлячись на те, що гра розрахована на одного гравця, можна вибрати собі компаньйонів (доступно 11 видів). Карта світу величезна. Кількість різних завдань в грі обчислюється сотнями.

Гра цікава своєю ігровою механікою та свободою дій. Персонажа можна озброїти потужними заклинаннями, надати йому безліч умінь та здібностей, наділити різними рисами, максимально підлаштувавши його під свої побажання і стиль гри. Крім того, гра дозволяє створити і розбудувати своє власне королівство.

1.4.3 Escape from Tarkov

Багатокористувацька рольова відеогра від першої особи, розроблена компанією Battlestate Games під керівництвом Микити Буянова, що поєднує в собі жанри FPS, бойового симулятора і RPG з MMO елементами.

Рисунок 1.16. Гра на Unity Escape from Tarkov

Події гри розгортаються у всесвіті, створеному Микитою Буяновим у 2009 році і раніше використаному у грі Contract Wars. Місце дії — вигадане місто Тарков в Норвінскій області.

Локація гри — місто і його околиці — ізольована від зовнішнього світу миротворчими силами ООН і збройними силами після політичної кризи, викликаної незаконною діяльністю транснаціональної корпорації TerraGroup. У місті — активний збройний конфлікт між двома приватними військовими підрозділами: USEC (представляють інтереси TerraGroup) та BEAR

Розробники втілили реалістичну бойову модель з достовірною фізикою і балістикою: перегрів, клин та поступове зношування зброї, реалістичні процеси перезарядки, прицілу і пострілу. Збережено реалістичні погодні умови, зміну дня та ночі

Запроваджено новий системний модуль — симулятор бою в екстремальних умовах, що дає повний контроль над рухами персонажа. На стан персонажа впливають зневоднення, нервове виснаження, хвороби і травми.

Для покращення фізичного стану гравцям доведеться використовувати медикаменти та різноманітні речі побуту. У грі реалізовано реалістичну

економічну систему, механіки торгівлі, обміну та курсу валют

Щоб виконати завдання дипломного проектування, буде використовуватись безкоштовна персональна версія ігрового двигуна Unity.

1.4.4 Середовище розробки Visual Studio

Visual Studio 2017 також буде використовуватись під час розробки проекту для дипломного проектування, як середа розробки для програмної частини цифрової гри. Visual Studio буде використовуватись як редактор коду та оглядач коду проекту. Безпосередньо для роботи із Unity, буде використовуватись модуль мови програмування C#, оскільки саме ця мова використовується ігровим двигуном.

Microsoft Visual Studio— один з продуктів компанії Майкрософт, який

представляє з себе інтегровану середу розробки програмного забезпечення і ряд

інших інструментальних засобів. Даний продукт дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в не керованому, так і керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудовуваних інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду, додавання нових наборів

інструментів, наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів циклу розробки програмного забезпечення. Загальний вигляд вікна, яке

відображається при завантаженні Visual Studio, початок роботи та створення нового проекту (рисунок 1.17)

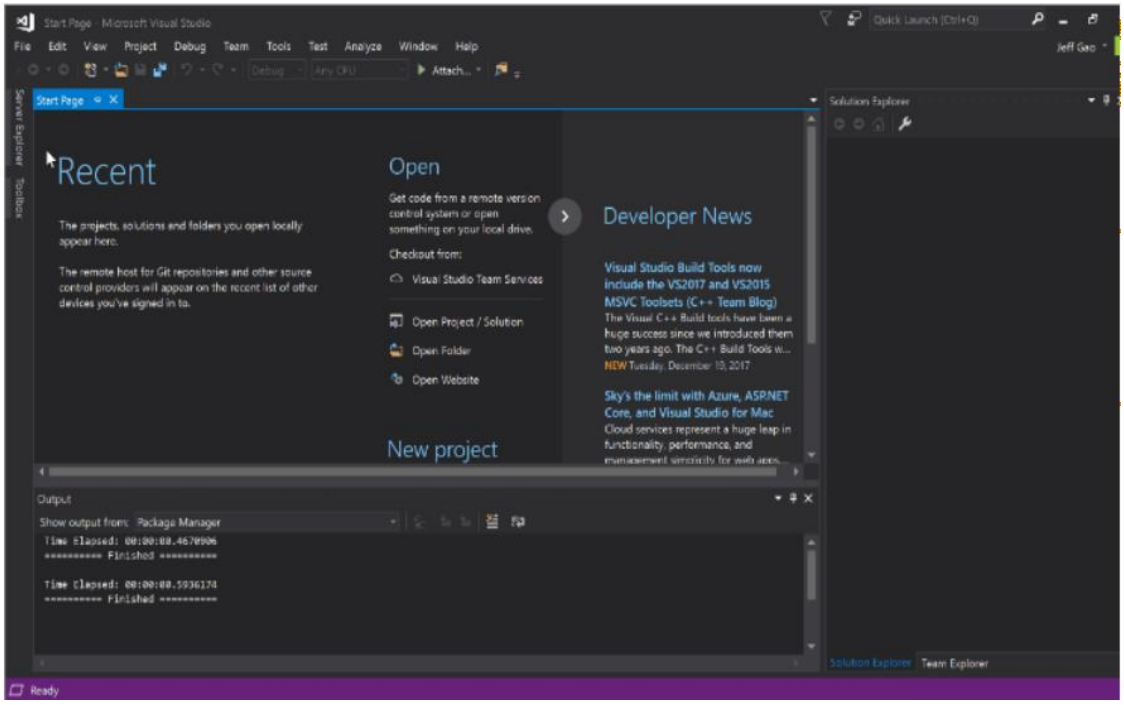


Рисунок 1.17. Visual Studio

З точки зору ліцензії Visual Studio також має гнучку систему тарифів та планів використання. Visual Studio залишається безкоштовним для самостійного не комерційного використання, а також використання у цілях освіти та науки. Безкоштовна версія не відрізняється по функціоналу від платної.

1.5 Процес створення макету 2D-гри у жанрі top down shooter

Реалізація логіки для ігрових об’єктів та елементів інтерфейсу передбачає основу, на якій будуть виконуватись ці роботи. Такою основою є цифрова 2D-гра, в жанрі top down shooter. Створення макету цієї гри починається з розробки концепту ігрового процесу. Оскільки нам відомий жанр та стиль візуалізації, то можна розробити невеликий дизайндокумент проекту.

Ігровий процес буде представляє собою переміщення по рівням гри, пошук ігрових ресурсів, та «перестрілки» із об’єктами ворогів. За виконання тих чи інших дій гравцю нараховуються ігрові бали. Приклад того, як може виглядати ігровий процес можна побачити на рисунку 1.18:

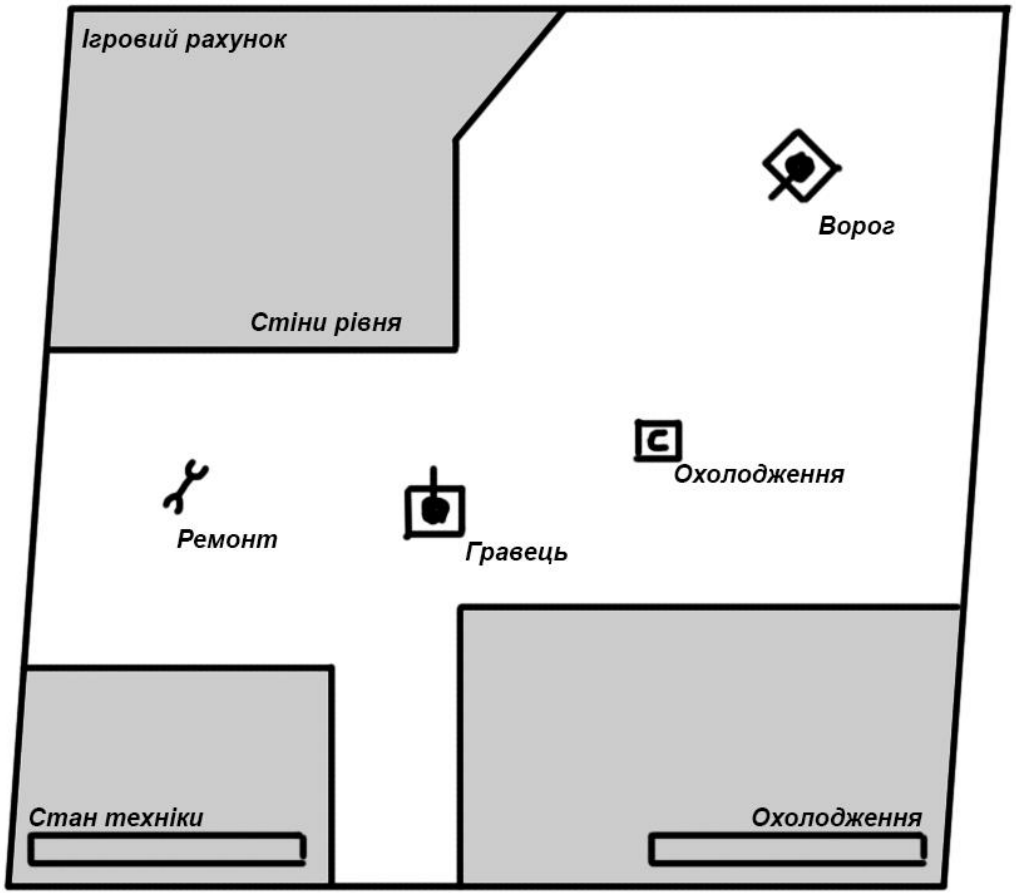


Рисунок 1.18. Дизайн-іллюстрація до проекту 2D-гри в жанрі top town shooter

Після підготовки дизайн-документу по проекту, потрібно розробити всі необхідні елементи та об’єкти, що потрібні для ігрового процесу. Завдяки потужним можливостям двигуна Unity, більшу частину таких об’єктів можна створити в самому Unity. Якість таких об’єктів буде невеликою, але для створення гри на ранніх етапах цього достатньо. З того, що неможливо зробити в самому двигуні лише спрайти для інтерфейсу. Після створення всього необхідного, ці об’єкти імпортуються в проект.

Наступник кроком є створення ігрової поверхні, та рівню (рисунок 1.19). Не дивлячись на те, що гра виконується в 2D-режимі, цілком допустиму

реалізовувати її із використанням 3D-графіки. Рівень створюється за допомогою примітивів: земля – площиною, а рівень кубами із різними параметрами деформації. Такий підхід дозволяє швидко перейти до розробки коду самої гри, залишивши роботу із візуальною складовою для людей, що за це відповідають, або на наступні етапи розробки.

Рисунок 1.19. Рівень із розміщеною землею та границями «коридорів» рівню»

Після створення елементів рівню, необхідно створити об’єкт гравця. Так само, як і попередні елементи, на початкових етапах роботи із проектом не обов’язково створювати повноцінну модель, або спрайт гравця, адже для реалізації основних ігрових механік буде достатньо примітиву. Тим більше, що рівні взаємодії виконуються із використанням потужностей Unity. Об’єкт гравця можна бачити на рисунку 1.20:

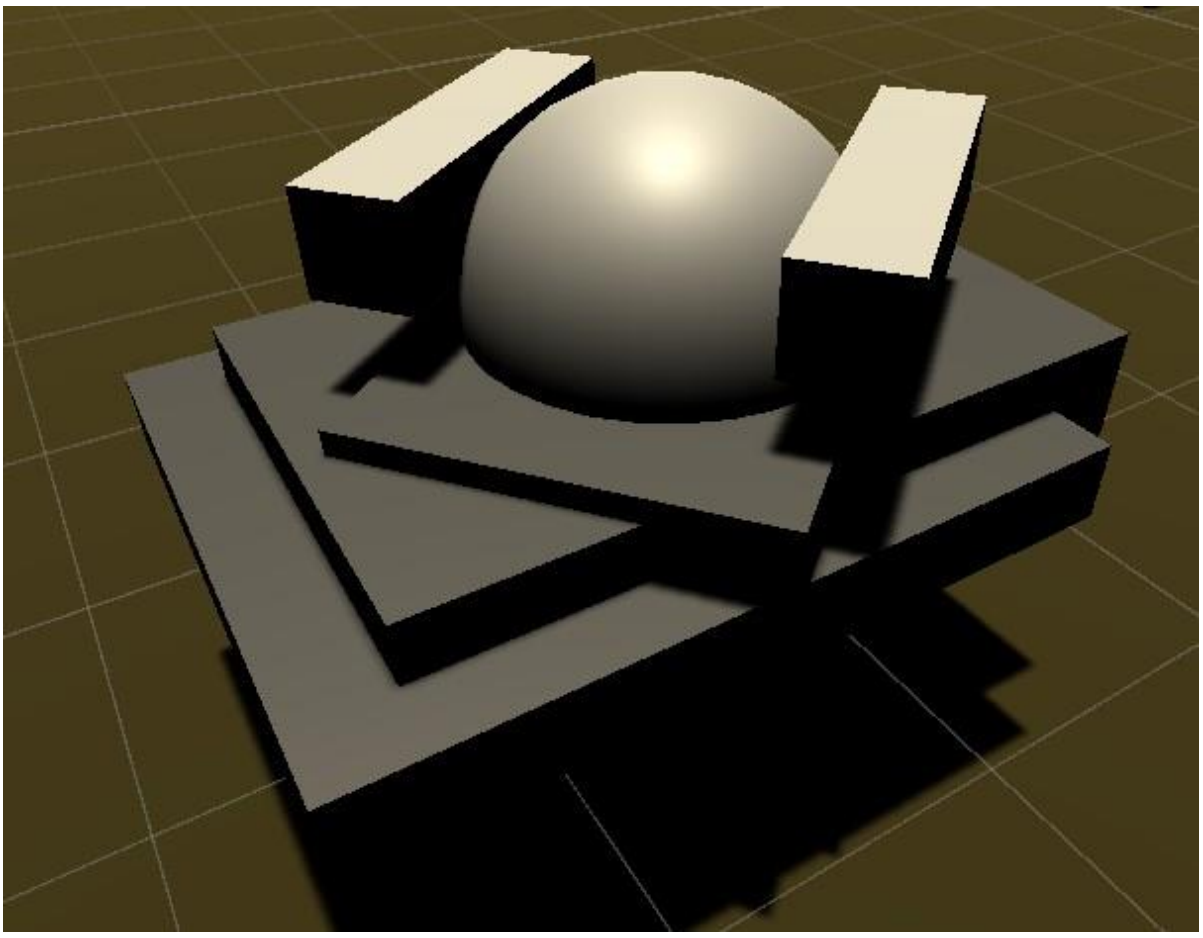


Рисунок 1.20. Об’єкт гравця зібраний із примітивів Unity

Наступним кроком в реалізації проекту буде написання коду для управління об’єктом гравця. В Unity передбачено багато різних видів роботи із фізикою та управління об’єктами. В даному випадку, для переміщення гравця по рівню треба окремо реалізувати рух техніки по рівню вліво-вправо, а також верх-вниз. І окремо управління баштою техніки. Перед цим, спочатку необхідно закріпити камеру над гравцем, для того щоб забезпечити реалізацію жанрової особливості, а також для коректності отримання даних прицілювання мишею. Також необхідно перемкнути камеру в режим проекції Ortthographic – це є обов’язковим для того, щоби отримувати вірні параметри положення миші на екрані, а також виключити деформацію об’єктів на рівні. Алгоритм роботи коду обробки вводу для руху гравця можна побачити на рисунку 1.21.

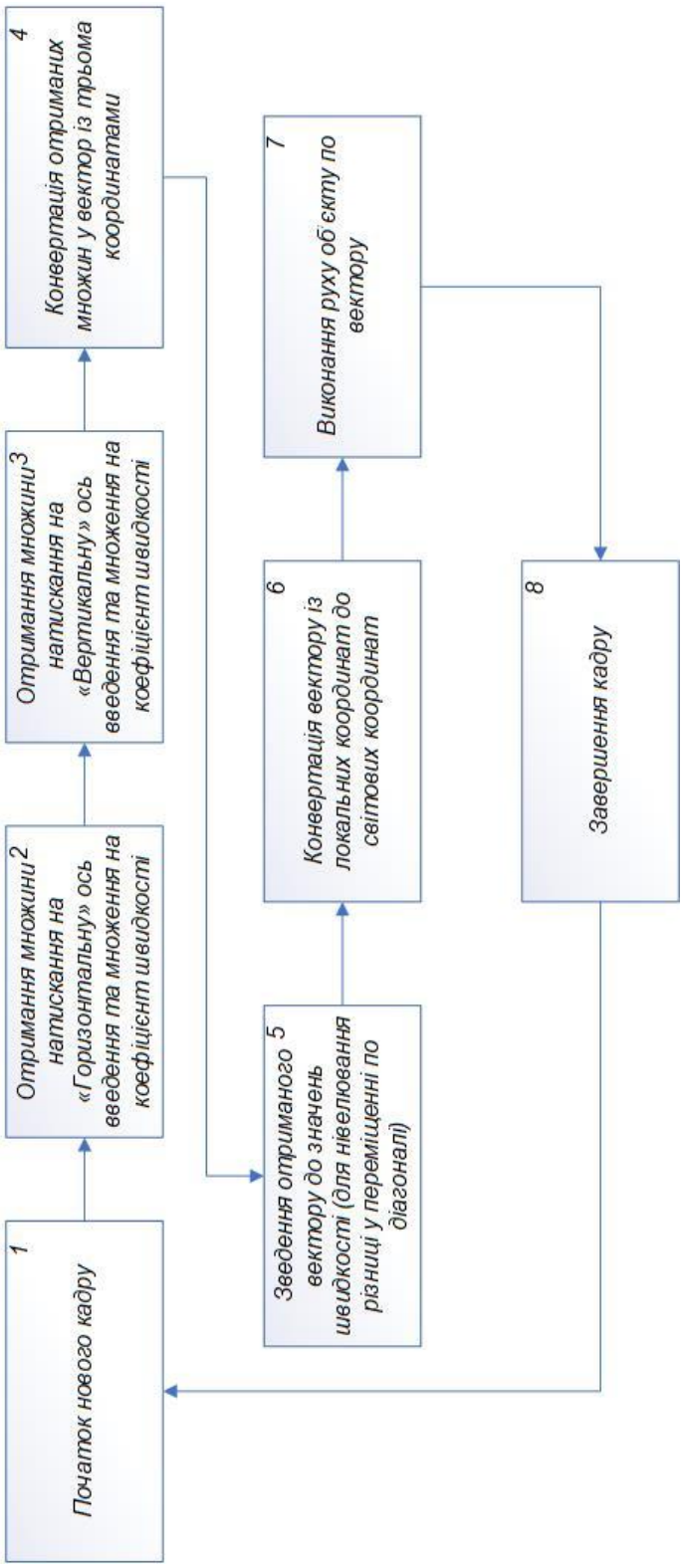


Рисунок 1.21. Алгоритм роботи коду переміщення об’єкту гравця

Його робота складається з покадрового зчитування натискання кнопок руху, множення цих значень на коефіцієнт швидкості, перетворення отриманих величин у вектор із трьома координатами, нормалізації цього вектору та

виконання руху об’єкту гравця. Із новим кадром процес повторюється.

За дещо схожим принципом працює код для реалізації прицілювання мишею та повороту башти гравця. Спочатку отримуються координати миші на екрані гравця, та заносяться у вектор із трьома координатами. Далі виконується трансляція цього вектору із «поверхні камери» на світові координати. Від отриманого вектору віднімають вектор позиції гравця для отримання вектору напряму між векторами. Ці дані перетворюємо у градуси та виконуємо поворот об’єкту башти гравця на даний градус по осі Y, адже саме ця ось є перпендикулярною до поверхні рівню. Алгоритм роботи цього коду можна побачити на рисунку 1.22:

Також, необхідно реалізувати постріли гравця. Оскільки у об’єкта гравця є 2 гармати, таким чином и постріли необхідно реалізувати роздільно. Принцип роботи обох гармат однаковий, відрізняється лише візуально. З точки зору двигуна постріли реалізуються за допомогою променів, які генеруються у точці положення гармати та проецирується по осі Z яка спрямована вперед гармати. Далі луч повертає координату в якій був перетин фізичного об’єкту, а також за необхідністю цей луч може повернути дані про цей об’єкт. Візуальна реалізація цього процесу виконується за допомогою компоненту LineRenderer який відображає лінії за заданими точками, якими у випадку імітації роботи лазеру буде стартова та кінцеві точки променю.

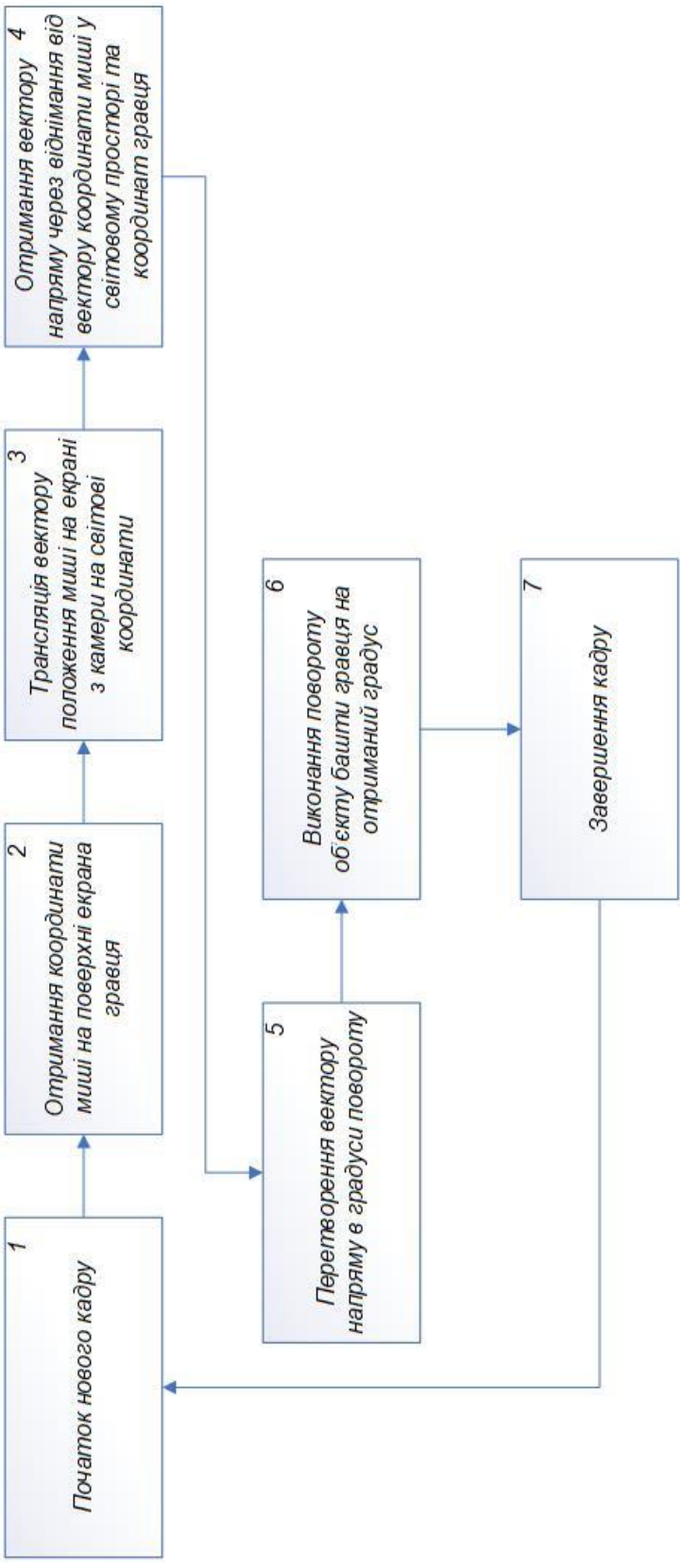


Рисунок 1.22. Алгоритм роботи коду повороту об’єкту башти гравця та прицілювання

Код реалізує визначені вище положення, покадрово перевіряючи натискання на ліву, або праву кнопки миші, які відповідають за різні гармати. При натисканні на одну з кнопок відповідна гармата створює промінь за координатами об’єкту гармати та проецирується по осі Z. Наступним кроком є отримання об’єкту з яким був перетин – якщо він був. Вмикається LineRenderer, який малює лінію. Якщо натискання на мишу не відбувається LineRenderer вимикається. Алгоритм роботи коду можна побачити на рисунку 1.23:

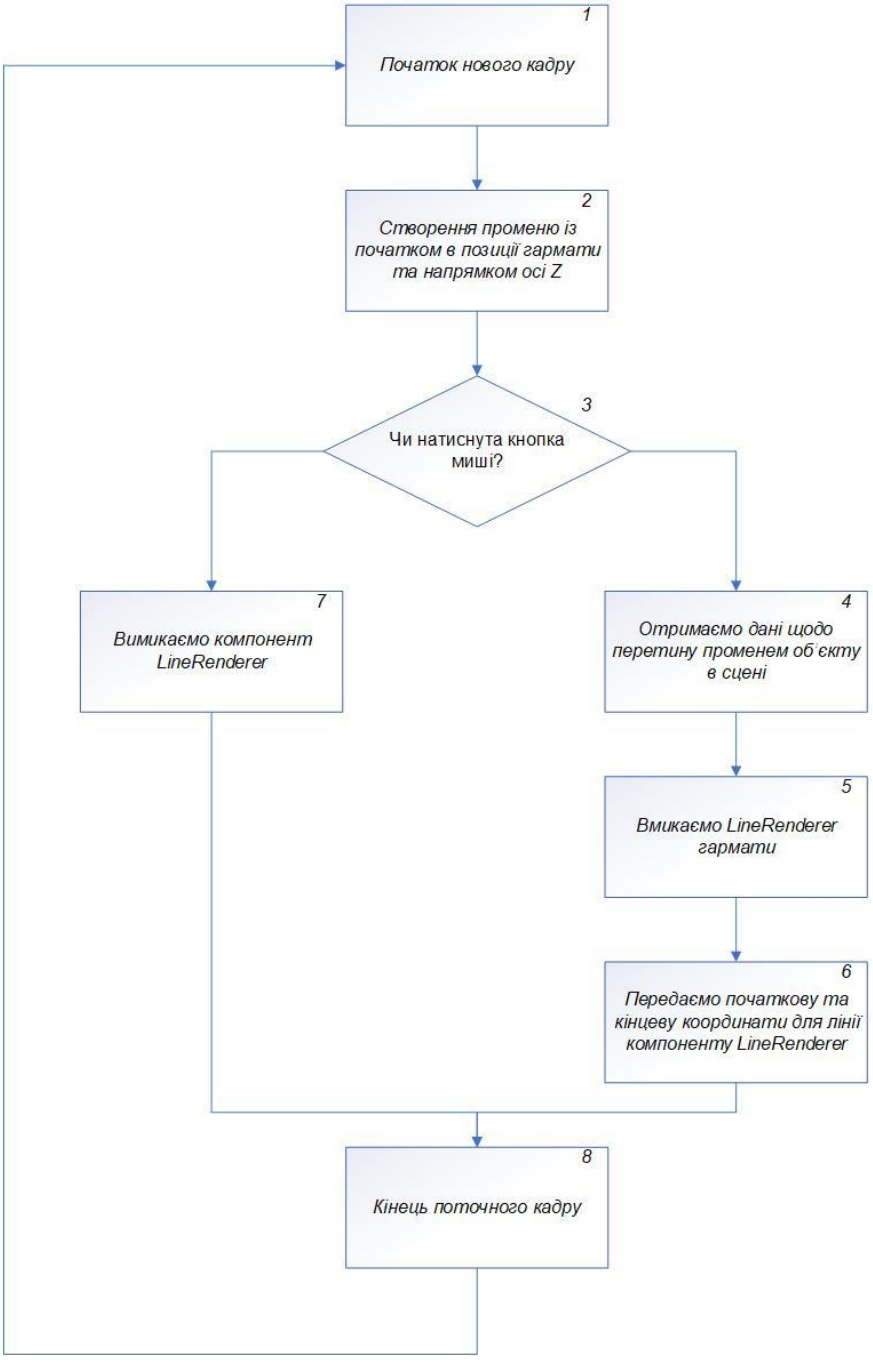


Рисунок 1.23. Алгоритм роботи коду пострілу для гравця

Виконання зазначених вище етапів створення початкового проекту дає можливість розпочати роботу над виконанням основного завдання дипломного проектування. На основі створеного коду та елементів проекту можна розпочати модифікацію елементів взаємодії із оточенням. Також наявність об’єкту гравця дасть можливість пов’язати його із інтерфейсом та камерою гравця. Ігрові ресурси, якими буде користуватись гравець, також можна створювати та тестувати на роботи вже маючі створені вище елементи проекту.

1.6 Проектування роботи логіки для ігрових об’єктів та елементів інтерфейсу гри

Метою розробки є створення логіки для ігрових об’єктів та елементів інтерфейсу гри. Розробка починається із постановки задачі та проектування роботи даних елементів, їх взаємодії та взаємозв’язків.

Логіка ігрових об’єктів передбачає створення умов, при яких об’єкти на ігровій сцені будуть взаємодіяти з гравцем або оточенням, ігровими ресурсами гравця та інтерфейсу.

В свою чергу інтерфейс має також виводити на екран основну інформацію про ігрові ресурси гравця, його параметрів, а також швидко та вірно реагувати на зміну цих даних.

Окремо треба передбачити принцип модульності в розробці, а також можливість внесення додаткових ігрових механік в існуючу систему без необхідності її повної перебудови. Інакше кажучи – потрібно спроектувати роботу логіки ігрових об’єктів та елементів інтерфейсу таким чином, щоби додавання нових елементів ігрових механік відбувалося без додаткових робіт.

Оскільки логіка та ігровий інтерфейс працюють із однаковими даними та мають прямий взаємозв’язок, є доцільним створити між цими частинами гри тісний маршрут роботи з точки зору коду.

Наступник кроком є опрацювання того, з чого будуть складатись ці елементи гри, які функції вони будуть виконувати та на які блоки будуть

поділені. Спочатку розділимо ігрову логіку і інтерфейс на дві окремі частини і почнемо роботу саме з логіки ігрових об’єктів. Оскільки всі ігрові елементи взаємодіють один із одним тим, чи іншим способом, необхідно створити код для об’єкту «Ігрового менеджеру».

Цей об’єкт буде виконувати роль посередника між елементами ігрової логіки, а також ігровим інтерфейсом та ігровою логікою. Головна мета роботи «Ігрового менеджеру» полягає в контролюванні ігрового процесу та своєчасній централізованій передачі даних до елементів, що їх потребують. Таким чином «Ігровий менеджер» буде зберігати в собі основні параметри гри та ігрові ресурси гравця. Всередині цього об’єкту необхідно створити методи, що дають змогу записувати чи зчитувати ці параметри. В свою чергу «Ігровий менеджер» повинен мати доступ до об’єктів ігрової логіки, оскільки саме він впливає на їх роботу, опираючись на параметри, що зберігає. Функційну схему зв’язку та взаємодії «Ігрового менеджеру» та інших ігрових об’єктів можна переглянути на рисунку 1.24.

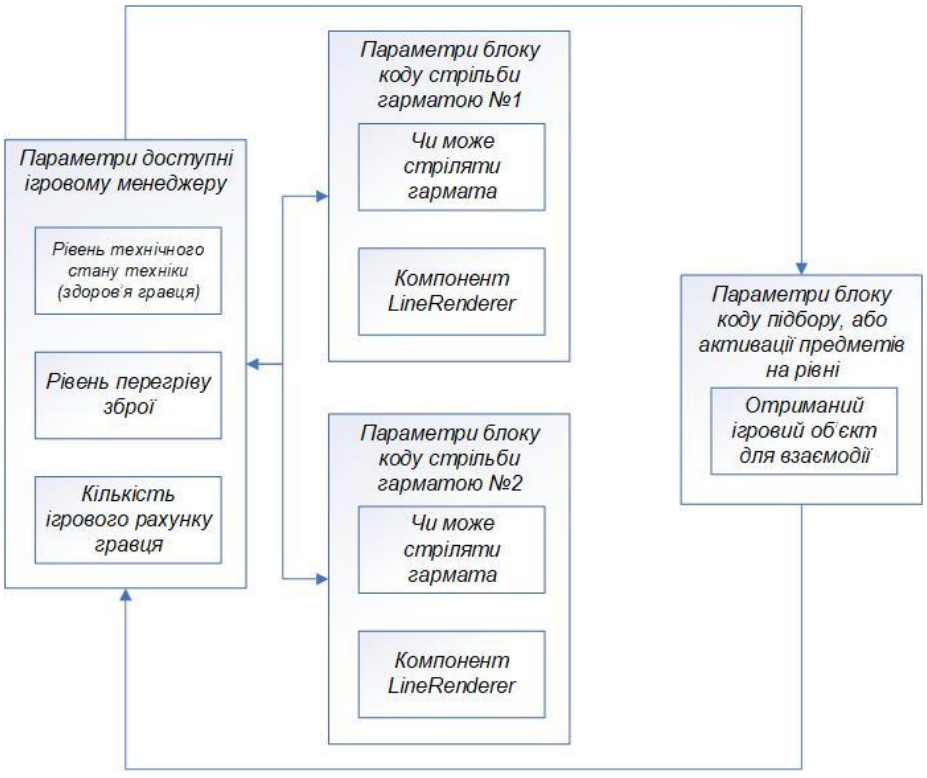


Рисунок 1.24. Функційна схема зв’язку та взаємодії «Ігрового менеджеру» та інших ігрових об’єктів

З точки зору виконання завдання дипломного проекту, в схемі зображеній на рисунку 1.24 нас цікавить робота Блоку коду підбору, або активації предметів на рівні. Саме цей об’єкт та його робота із Ігровим менеджером будуть реалізовувати логіку ігрових об’єктів.

Функційна схема взаємодії між «Ігровим менеджером» та Блоком коду підбору, або активації предметів на рівні зображена на рисунку 1.25. З цієї схеми можна бачити, що якщо під час пересування гравця по рівню він починає взаємодіяти із іншим ігровим об’єктом на рівні, то під час цієї взаємодії, отримується інформація щодо того, якого типу цей об’єкт. Потім необхідно порівняти тип об’єкту із переліком дій пов’язаних із типом об’єкту та якщо є збіг, тоді необхідно звернутись до «Ігрового менеджеру» для виконання необхідної дії. Тип об’єкту визначається за його «Тегом». Теги – це заздалегідь створені «імена» типів ігрових об’єктів в Unity. Можна використовувати, як теги створені розробниками ігрового двигуна, так і створити свої. Оскільки в нашій грі є 4 різні види ігрових об’єктів для взаємодії, отже нам необхідно і 4 тега. По 2 на об’єкти що підвищують ігровий ресурс гравця, а також по 2 для точок багаторазового відновлення ігрових ресурсів гравця.

Після виконання необхідних дій у «Ігровому менеджері», виконуюча середа повертає контроль до Блоку коду підбору, або активації предметів на рівні. У випадку, якщо об’єкт був одноразовим, то його знищують, інакше, цей крок ігнорується. Після чого гра переходить далі по розрахункам в кадрі. Окремо, слід зазначити, що в разі, якщо при взаємодії з якоїсь причини збігу по тегам не було виявлено, виконуюча середа також переходить до виконання наступних дій в кадрі. За схожим принципом реалізується взаємодія «Ігрового менеджера» з іншим кодом, який відповідає за логіку ігрових об’єктів.

Рисунок 1.25. Функційна схема взаємодії між «Ігровим менеджером» та Блоком коду підбору, або активації предметів на рівні

Наступним кроком буде проектування роботи взаємодії коду та інтерфейсу. Так само, як із проблемою логіки ігрових об’єктів, нам необхідно

буде створити «Менеджер ігрового інтерфейсу», який буде посередником між інтерфейсом гри та безпосередньо кодом самої гри. Взаємодія між ними більш проста, на відміну від «Ігрового менеджеру» та об’єктами на рівні, оскільки вся робота складається у створенні зв’язку між елементами інтерфейсу та кодом. Функціональну схему роботи «Менеджеру ігрового інтерфейсу» та елементів інтерфейсу можна побачити на рисунку 1.26:

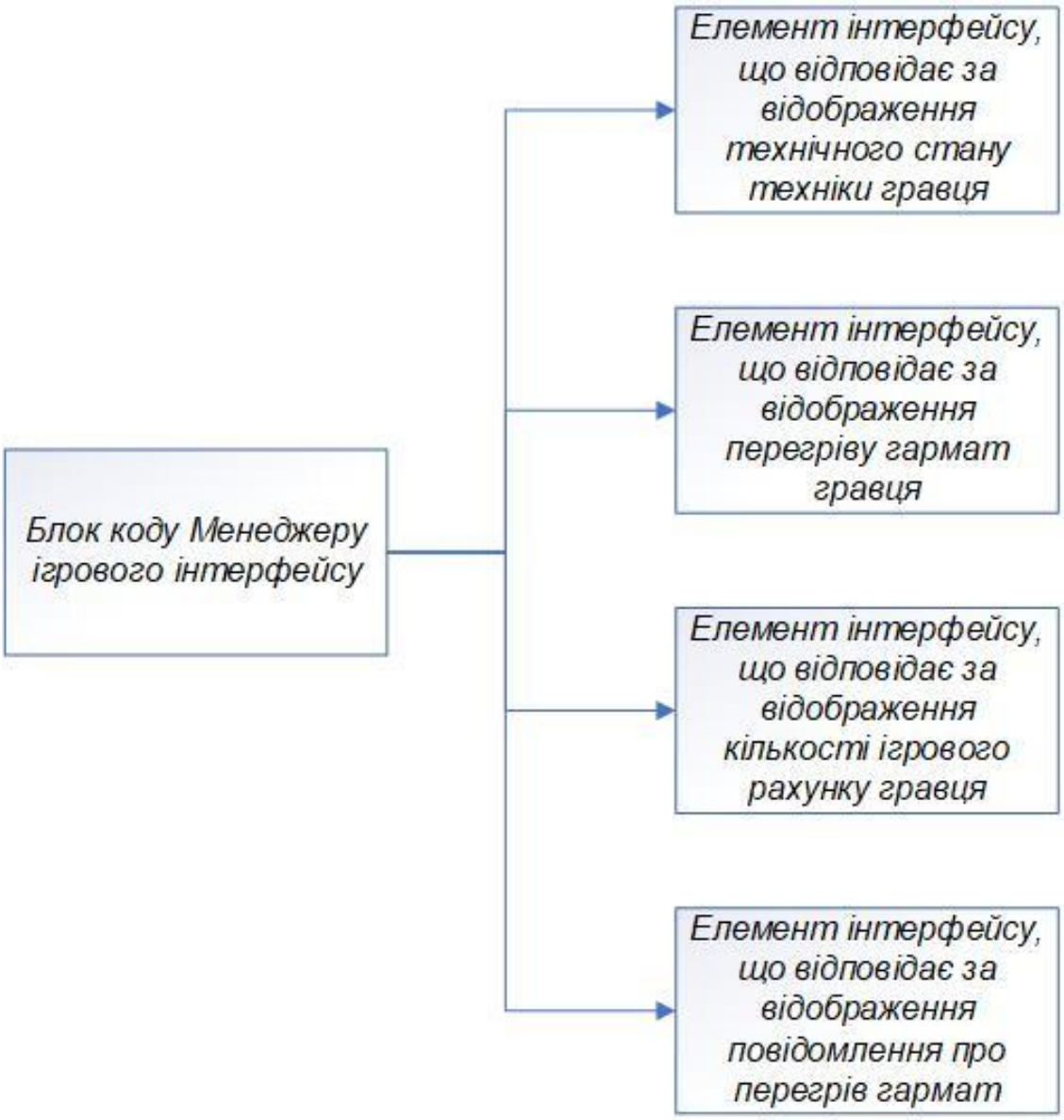


Рисунок 1.26. Функціональна схема роботи «Менеджеру ігрового інтерфейсу» та елементів інтерфейсу

Останнім кроком в проектуванні буде передбачення взаємодії, між «Ігровим менеджером» та «Менеджером ігрового інтерфейсу». Як вже було

зазначено вище, це потрібно зробити через те, що ігрові ресурси знаходяться у сфері відповідальності «Ігрового менеджеру». Окрім того, блоки коду гри можуть змінювати ці ресурси і щоби не розмножувати зв’язки в проекті, доцільно централізовано обмінюватись даними через «Ігровий менеджер». Функціональну схему взаємодії логіки ігрових об’єктів та ігрового інтерфейсу можна побачити на рисунку 1.27:

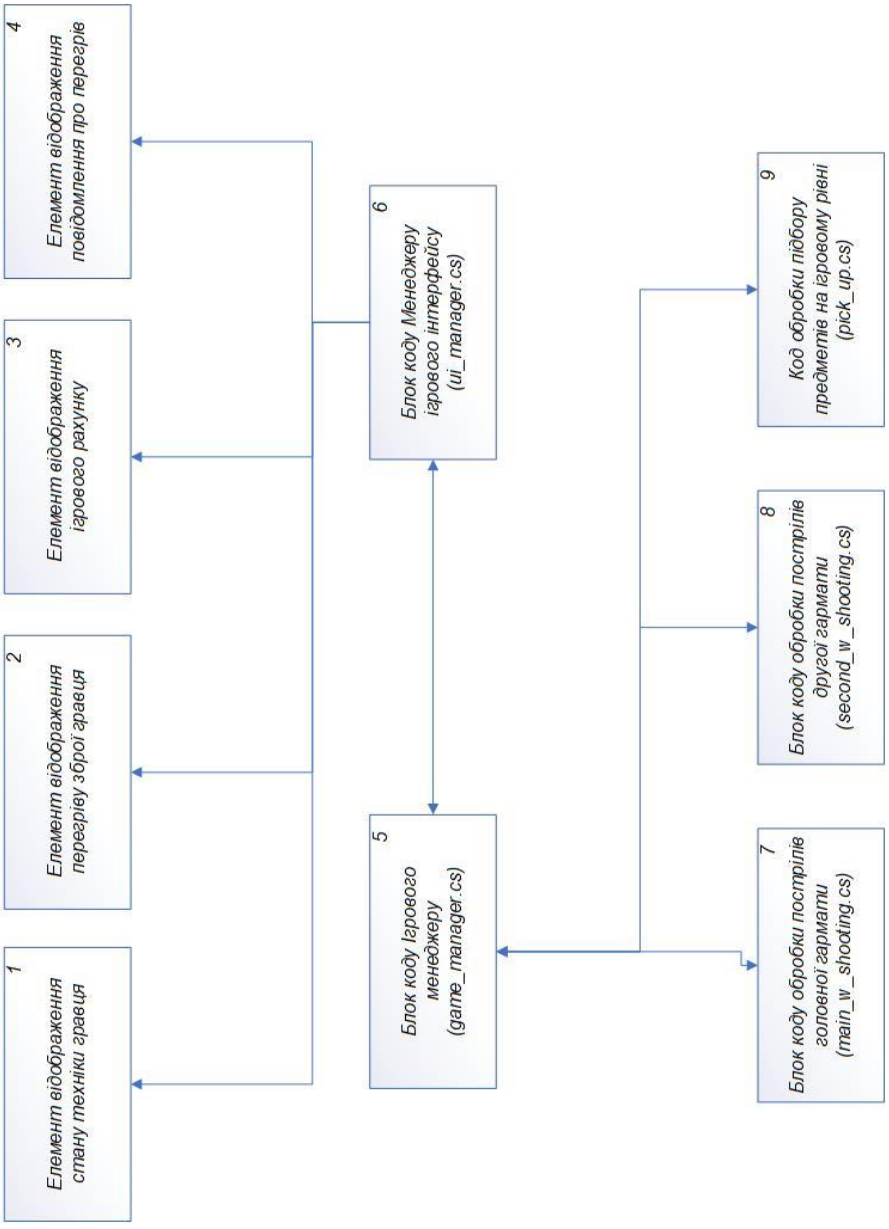


Рисунок 1.27. Функціональна схема взаємодії логіки ігрових об’єктів та ігрового інтерфейсу

1.7 Реалізація логіки для ігрових об’єктів та елементів інтерфейсу гри

Реалізація спроектованих блоків та коду в основному полягає в роботи із редактором ігрового двигуна Unity, загальний вигляд якого можна побачити на рисунку 1.28:

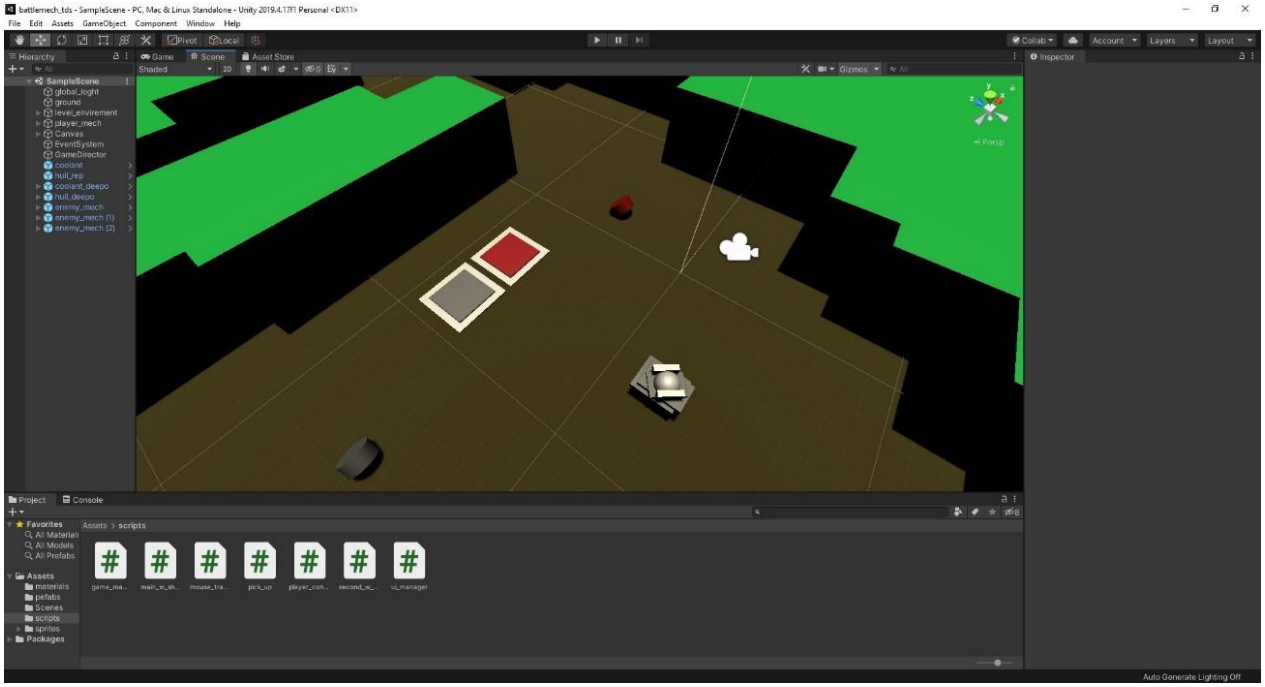


Рисунок 1.28. Загальний вид редактора ігрового двигуна Unity

В першу чергу реалізовуємо об’єкти із котрими будемо працювати. Цей процес в Unity полягає в створені примітивів через розділ редактора Hierarchy (рисунок 1.29). В цьому розділі відображаються всі ігрові об’єкти, що є на поточній сцені. Такими об’єктами можуть бути примітиви, пусті об’єкті, які нічного в середині не мають, або особливі об’єкти, як елементи освітлення, звуку, відео, інтерфейсу та інші.

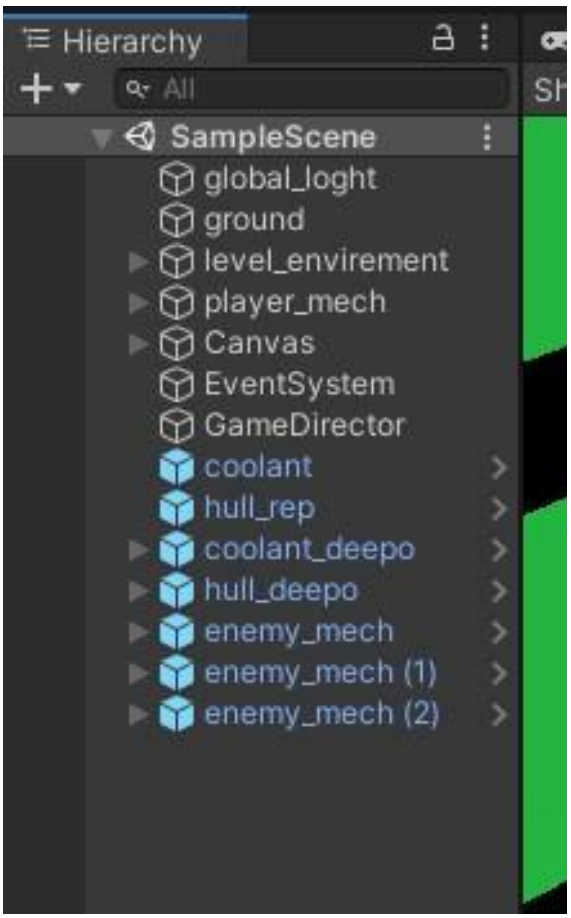


Рисунок 1.29. Загальний вид розділу Hierarchy редактору ігрового двигуна Unity

Сценою в Unity називають рівень, в якому знаходяться ігрові об’єкти, логіка та інші складові сцени. Слід зазначити, що сцени можуть перемикатись одна на одну. Їх може бути велика кількість і кожна з них, може мати свою унікальну структуру, свої ігрові об’єкти та використовувати свої скрипти. Сцена сама по собі представляє собою файл, в якому зберігається текстова інформація про зміст сцени. Коли ігровий двигун завантажує сцену, він зчитує цю інформацію та відтворює її своїми силами.

Нам для реалізації логіки ігрових об’єктів буде достатньо створити примітиви – їх описання було вище. Ці об’єкти будуть слугувати для ремонту техніки, охолодження гармат гравця, а також набору ігрового рахунку. Після створення примітиву, він буде розміщений на сцені, як показано на рисунку 1.30:

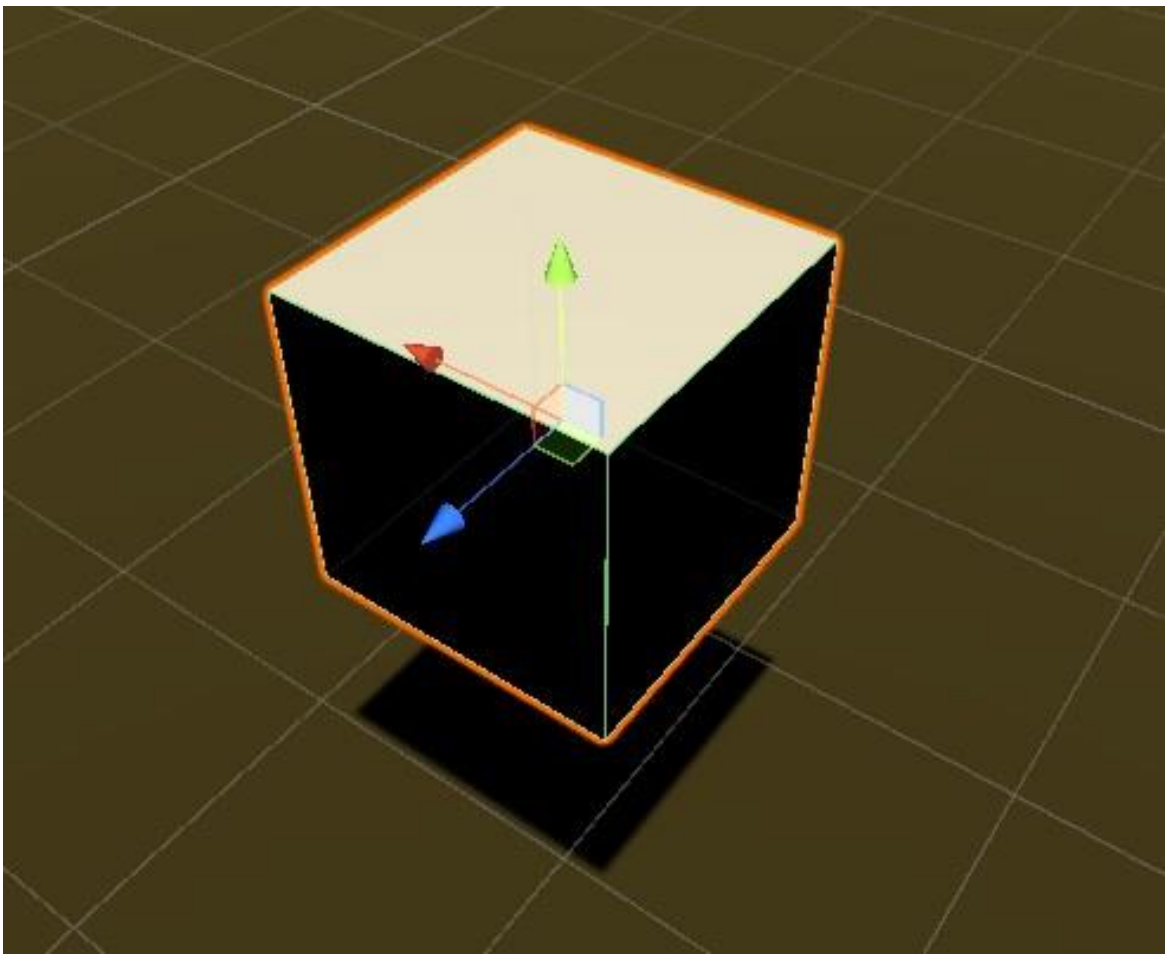


Рисунок 1.30. Примітив створений через розділ Hierarchy

та розміщений на сцені.

Після того як об’єкт з’явиться на сцені, він буде розміщений в просторі ближче до середини акценту камери редактора. Тому, об’єкту необхідно буде дати конкретні координати розміщення в просторі. Це можна зробити через розділ Inspector, загальний вид якого можна побачити на рисунку 1.31.

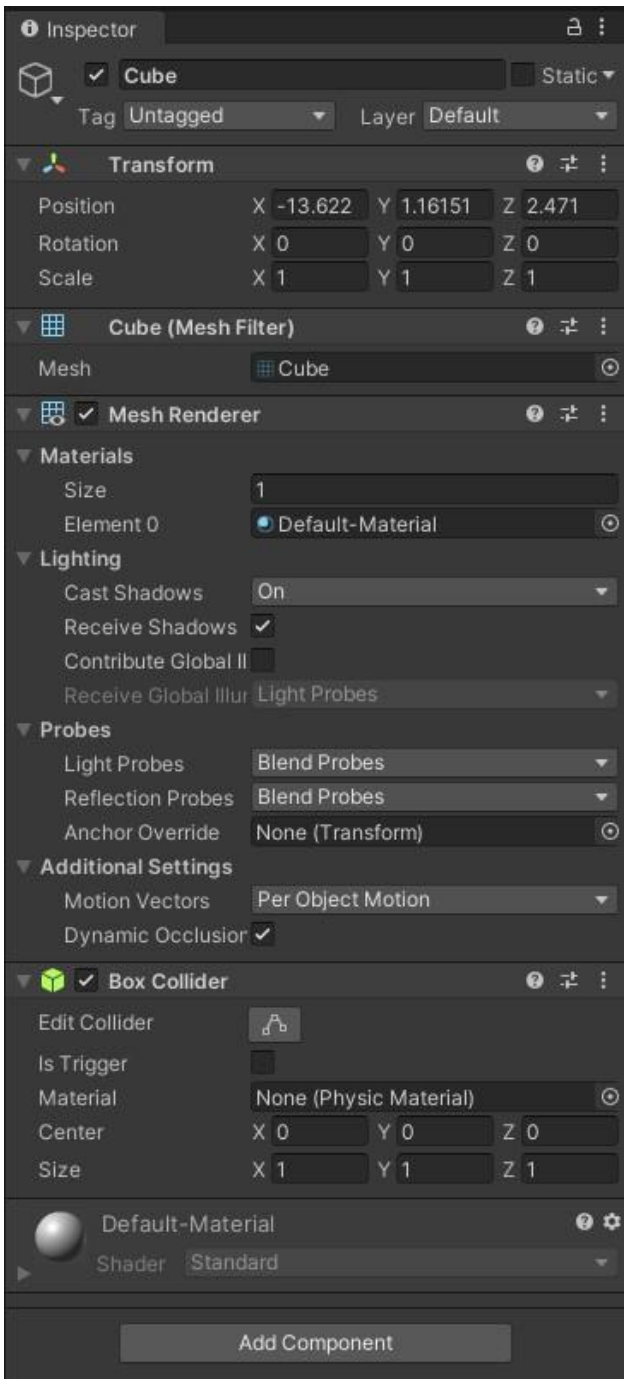


Рисунок 1.31. Загальний вид розділу Inspector редактора ігрового двигуна Unity

Inspector також є одним із головних розділів двигуна, оскільки саме в ньому можна додавати до об’єктів нові компоненти. Компонентами в Unity називають властивості, що мають ігрові об’єкти на сцені. Є такі компоненти, що існують у об’єктів за замовченням, як компонент Transform, що зберігає положення об’єкту в просторі ігрової сцени, його поворот та масштабування.

Також, щойно створений об’єкт має типові компоненти притаманні кубу – Меш Кубу, Рендерер Кубу, Колайдер Кубу а також матеріал за замовченням. Всі ці компоненти можна редагувати, видаляти а також додавати нові за допомогою кнопки Add Component.

Ще одним важливим розділом редактора ігрового двигуна Unity є Project, загальний вид якого можна побачити на рисунку 1.32. Цей розділ відповідає за навігацію по проекту, а саме його папкам та файлам. Саме тут можна створювати спеціалізовані під Unity файли без використання зовнішнього програмного забезпечення.

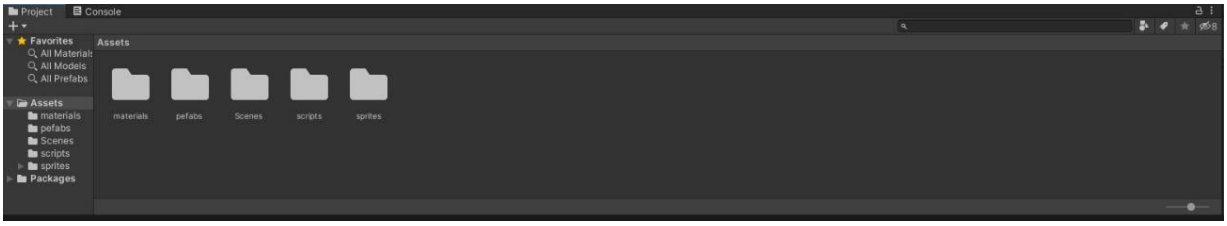


Рисунок 1.32. Загальний вид розділу Project редактора ігрового двигуна Unity

Тут важливо створювати відповідні папки для відповідних файлів. Наприклад, для створення об’єкту для ремонту техніки гравця, нам необхідно виконати його в кольоровій гамі, в якій цей ресурс відображається у ігровому інтерфейсі. Змінити колір об’єкту можна за допомогою Material, або матеріалу – це файли, що імітують той чи інший матеріал, його колір та оптичні властивості. Створивши папку materials ми правою кнопкою викликаємо контекстне меню та обираємо Create, а там Material. Буде створено матеріал, якому ми даємо ім’я, після чого редагуємо. Додавши матеріал до компоненту Mesh Renderer ми змінюємо колір примітиву.

Також, через те що наш об’єкт повинен взаємодіяти із гравцем, то необхідно змінити властивість компоненту Box Collider, поставивши позначку в пункті Is Trigger. Це зробить об’єкт «безтілесним», але коли інший ігровий об’єкт із «тілом» перетне його, то спрацює тригер, який можна буде використати у подальшому.

Коли робота зі створення основних об’єктів для взаємодії завершена, для того щоби кожен раз не повторювати роботу, треба створити «префаби» об’єктів. Префаб – це окремий вид об’єкта, який створюється у папці проекту. У подальшому такий об’єкт можна використовувати багаторазово та кожний екземпляр буде існувати окремо один від одного. Якщо після розміщення всіх префабів з’явиться потреба ввести зміну до префабів, то буде достатньо змінити той, що знаходиться в папці проекту і зміни в ньому автоматично будуть додані до його варіантів на сцені. На рисунку 1.33 можна побачити створені для проекту префаби:

Рисунок 1.33. Створені для проекту префаби

Після створення всіх об’єктів для взаємодії, необхідно створити ігровий інтерфейс. Це можна зробити так само, як об’єкти через розділ Hierarchy, створивши там об’єкт Canvas. Це нова модель роботи з інтерфейсом в Unity за допомогою «холстів». Всередині цих холстів ми можемо створювати спеціалізовані об’єкти інтерфейсу та редагувати їх безпосередньо на сцені.

Створивши всі об’єкти на сцені, а також ігровий інтерфейс, потрібно розпочати роботу із кодом. В Unity код для гри пишеться за допомогою невеличких скриптів, що виконують запрограмований сценарій. Ці скрипти створюються в папці проекту через розділ Project, а потім додаються до ігрових об’єктів, як компоненти. Скрипт написаний та доданий до проекту не працює до тих пір, поки він не буде доданий до об’єкту на сцені. Далі скрипт отримує доступ до компонентів об’єкту. Подальший зв’язок із іншими ігровими об’єктами реалізовується за допомогою програмування, через спеціальні команди скриптів Unity.

Оскільки перед нами стоїть задача створити модульну структуру скриптів, треба відразу розробити код таким чином, щоби він міг працювати і в інших проектах. Таким чином, використовуючи спроектовані вище функційні схеми, можна створити код скриптів, враховуючи всі вказані на схемах зв’язки. Зв’язок між скриптами та об’єктами в Unity виконується за допомогою створення змінних з інструкцією SerializedField. Це дає змогу вивести змінну у розділ Inspector та задавати їх значення, або, якщо як тип такої змінної ми вказували ігровий об’єкт – передати в цю змінну через Inspector цільовий об’єкт. Таким самим чином, ми можемо передавати скрипти, вказавши їх, як тип для змінних із інструкцією SerializedField.

Наприклад, скрипт для «Ігрового менеджера» буде мати в собі посилання на всі скрипти, які задіяні у виконанні ігрової логіки. Всередині цього скрипту, будуть створені методи для отримання значень, та відправки значень, оскільки всі ігрові ресурси гравця зберігатимуться саме в «Ігровому менеджері». Також, в цьому скрипті виконуються методи для керування можливістю вести вогонь з гармат та передача статусу до них.

```
public void incr_player_overheat(float value)
{
    player_overheat += value;
    if(player_overheat > 100)
    {
        player_overheat = 100;
        Debug.Log("OVERHEATED!");
        StartCoroutine(overheated_core());
    }
}
```

Наприклад код методу, наведений вище викликається під час пострілу із

гармат. Скрипт пострілу, звертається до «Ігрового менеджеру» та викликає метод перегріву. Після чого «Ігровий менеджер» виконує збільшення показника перегріву. Слід зазначити, що гра перевіряє значення перегріву та в разі, якщо перегрів досяг значення 100, тоді викликається метод, який не дозволяє використовувати гармати. Нижче, його код:

```
public IEnumerator overheated_core()
{
    _main_w_shooting.main_w_overheated = true;
    _second_w_shooting.second_w_overheated = true;
    _main_w_shooting.clean_line();
    _second_w_shooting.clean_line();
    _ui_manager.show_overheat_line();
    yield return new WaitForSeconds(5f);
    Debug.Log("WAited");
    _main_w_shooting.main_w_overheated = false;
    _second_w_shooting.second_w_overheated = false;
    _ui_manager.hide_overheat_line();
}
```

Цей код методу спочатку звертається до гармат та переключає їх змінну перегріву main_w_overheated та second_w_overheated в положення true, що не дає в скрипту гармат виконувати постріли. Після цього, «Ігровий менеджер» примусово припиняє відображення лучів пострілів для обох гармат. Далі «Ігровий менеджер» звертається до «Менеджеру ігрового інтерфейсу» та викликає метод відображення повідомлення про перегрів. Після цього метод перегріву у «Ігровому менеджері» встає на очікування та передає управління грою далі. Після проходження 5 секунд, гра повертає управління назад до цього

метода, який перемикає змінні перегріву гармат назад у положення false, що дає змогу знову стріляти.

Так само і зі скриптом «Менеджеру ігрового інтерфейсу» – він буде вмістити в собі посилання на всі елементи ігрового інтерфейсу. В цьому скрипті також виконуються зміни цих елементів та їх вмісту. Відображення, або виконання команди на приховання елементу з інтерфейсу. Наприклад знизу показано код методу що виставляє поточне значення міцності корпусу гравця:

```
public void set_hull_value(float new_value)
{
    if(new_value <= hull_slider.maxValue)
    {
        hull_slider.value = new_value;
    }
}
```

Тут метод спочатку отримує нове поточне значення міцності корпусу гравця. Після чого виконується перевірка значення на граничні величини. Якщо отримане значення менше, або дорівнює максимальному значенню міцності корпусу, тоді отримане значення відображається на ігровому інтерфейсі. Це друга, запобіжна перевірка, оскільки даний метод викликається із «Ігрового менеджера» який також контролює вихід за границі розмірів значень, тому значення, яке приходить в цей метод майже гарантовано буде виконувати умову запису.

Єдиним, що відрізняє ці два скрипти – це відсутність зв’язку від «Менеджеру ігрового інтерфейсу» до «Ігрового менеджеру». Тільки останній може звертатись до першого, а не навпаки. Саме «Ігровий менеджер» вирішує, коли виконувати зміни у інтерфейсі та яким чином. Втім виконання зміни елементів є прерогативою «Менеджера ігрового інтерфейсу».

Є сенс окремо зупинитись на скрипті, який виконую підбір предметів на рівні, або використання «станцій» відновлення – скрипт pick_up.cs. Цей скрипт має 2 частини, які відповідають типам зустрічаємих на рівні об’єктів.

```
private void OnTriggerEnter(Collider other)
{
    GameObject entered_pick_up = other.gameObject;
    if(entered_pick_up.tag == "hull_rep")
    {
        _game_manager.incr_player_hull(25f);
        Destroy(entered_pick_up);
        _game_manager.incr_player_score(10);
    }
    else if(entered_pick_up.tag == "coolant")
    {
        _game_manager.decr_player_overheat(25f);
        Destroy(entered_pick_up);
        _game_manager.incr_player_score(10);
    }
}
```

Код події що представлений вище спрацьовує лише тоді, коли гравець колайдером своєї техніки пересікає буд-який колайдер на рівні. В цьому разі, із колайдера, що був перетнутий береться інформація про ігровий об’єкт. Якщо тег цього об’єкта відповідає тегу ремонту, або охолодження, тоді виконуються відповідні дії, після чого об’єкт знищується, а гравцю нараховуються ігровий рахунок.

```
private void OnTriggerStay(Collider other)
{
    GameObject entered_pick_up = other.gameObject;
    if ((entered_pick_up.tag == "hull_deepo") && (_game_manager._take_player_hull() < 100))
    {
        _game_manager.incr_player_hull(0.2f);
        _game_manager.incr_player_score(1);
    }
    else if ((entered_pick_up.tag == "coolant_deepo") && (_game_manager._take_player_overheat() > 0))
    {
        _game_manager.decr_player_overheat(0.4f);
        _game_manager.incr_player_score(1);
    }
}
```

Код події приведений вище також із скрипта pick_up.cs. Він виконує схожий процес, але зі «станціями» поповнення. Тобто об’єктами, які не знищуються під час використання. Втім, є різниця у виклику. Ця подія викликається кожний кадр, коли колайдер гравця знаходиться в іншому калайдері. Таким чино, ставши на станцію спрацьовує подія. Знову зчитується ігровий об’єкт із колайдеру, звіряються теги и якщо вони співпадають та ігровий ресурс не є максимальним, або мінімальним (у разі із перегрівом), тоді виконуються відповідні дії поповнення, та нарахування ігрового рахунку.

Окремо слід зазначити кінцеву структуру проекту, який має вид, як на рисунку 1.34:

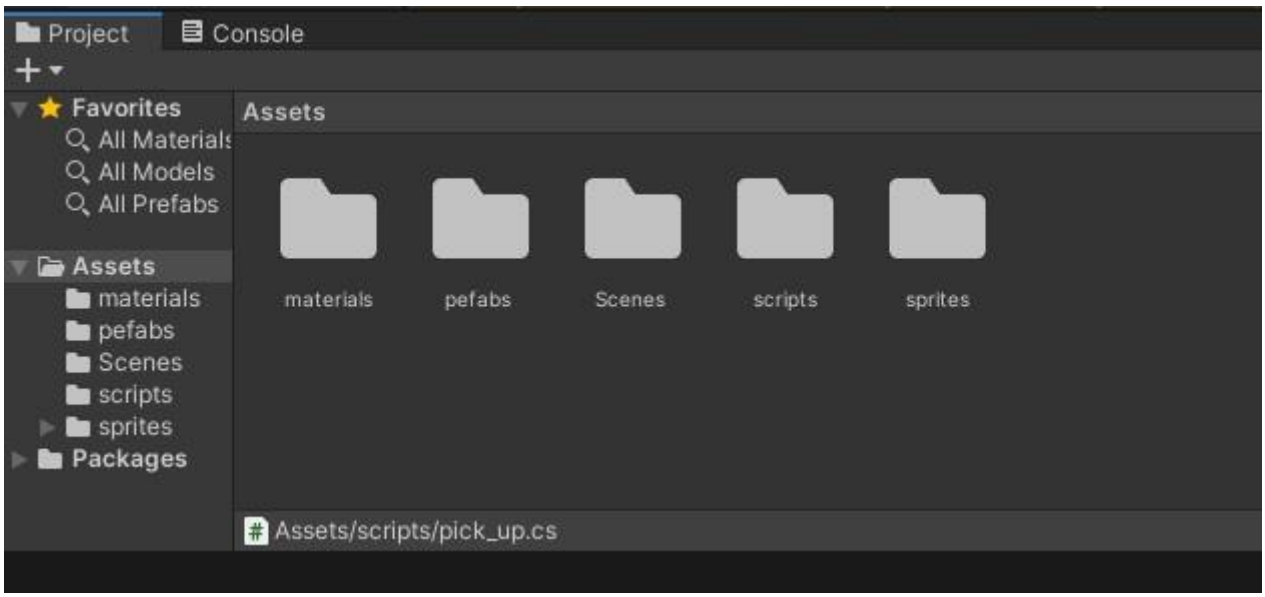


Рисунок 1.34. Структура проекту гри в Unity

Unity у своїй структурі має системні файли ігрового двигуна, а всі файли, які були створені під час роботи над проектом розміщаються у каталозі Asset. Тут же були створені цільові каталоги для робочих файлів та скриптів. Матеріали створювались та розміщувались у папці materials, а скрипти в паці scripts.

Файли використовуються через панель Inspector середи розробки Unity. Наприклад, використання скриптів виконується через перетаскування файлу скрипту у поле скрипта ігрового об’єкта, як зображено на рисунку 1.35:

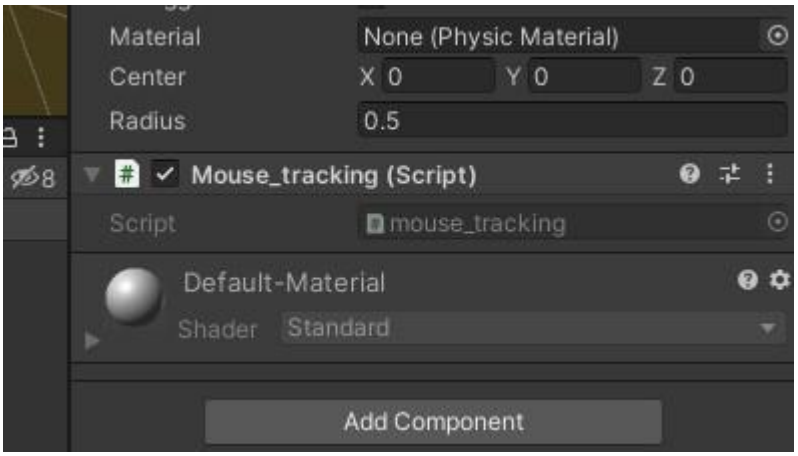


Рисунок 1.35. Приклад закріпленого скріпта за ігровим об’єктом

2. ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

Метою дипломного проектування було реалізувати логіку для ігрових об’єктів та елементів інтерфейсу 2D-гри в жанрі top down shooter, створення механізму взаємодії між ігровими елементами на сцені гри та елементами інтерфейсу, взаємодію між гравцем та оточенням.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

Таблиця 2.1 - Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП імітаційного моделювання	7800 – 8800

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі таблиці 2.2

Таблиця.2.2 Каталог норми часу

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 262 х 0,8 = 209,6 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (таблиця 2.3.); Кн – поправочний коефіцієнт, що враховує ступінь новизни (таблиця 2.4.); Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (таблиця 2.5.).

Таблиця 2.3. - Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11

РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Та*L_1*К_н=209,6*0,12*0,8= 20,12 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н=209,6*0,11*0,8 = 18,44 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=209,6*0,61*0,8*0,8= 81.83 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}=3$ (стр), розробка ТП $N_{тп}=9$(стр), розробка робочого проекту $N_{рп}=14$ (стр), пояснювальна записка відповідно $N_{пз}=36$ (стр)

Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Pтз}$=20,12	$T_{кк}$=0,7*$N_{тз}$= 0,7*3=2,1	$T_{нк}$=0,15*$N_{тз}$=0,15*3=0,45
2.Розробка ТП	$T_{Pтп}$=18,44	$T_{кк}$=0,7*$N_{тп}$=0,7*9=6,3	$T_{нк}$=0,15*$N_{тп}$=0,15*9=1.4
3.Розробка РП	$T_{Pрп}$= 81.83	$T_{кк}$=0,7*$N_{рп}$=0,7*14=9,8	$T_{нк}$=0,15*$N_{рп}$=0,15*14=2,1
4.Розробка ПЗ	$T_{пз}$=1,5**$N_{пз}$= 1,5*36 =54	$T_{кк}$=0,7*$N_{тз}$=0,7*36=25,2	$T_{нк}$=0,15*$N_{пз}$=0,15*36 =5,4
Усього, в т.ч.:	227,19		
- на розробку	$\Sigma T_р$=174,39		
- контроль керівника		$\Sigma T_{кк}$=43,4	
- нормоконтроль			$\Sigma T_{нк}$=9,4

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.7 - Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	174,39	50.00	8719,50
2.Контроль керівника	43,4	60,10	2608,34
3.Нормоконтроль	9,4	60,10	564,94
Усього	-	-	ΣЗо= 11892,78

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8:

Таблиця 2.8.- Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	70	3.0	210,0
Разом	-	-	-	$В_{м\ і}$ =210,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*210=21,00
Усього				$В_м=В_{мі}+В_{тр\_з}$= 231,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. - Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	231.10	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	11892,78	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1189,28	$Зд = 0{,}1 \times Зо$ = 11892,78*0,1
4.Відрахування до єдиного фонду соціального внеску	2878,05	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (11892,78+1189,28)
5. Накладні витрати	4757,11	$Внак. = 0{,}4 \times Зо$ =0,4 * 11892,78
6. Повна собівартість	20948,32	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 231.10+11892,78+1189,28+2878,05+ 4757,11

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100=(20948,32*10)/100=2094,83$ грн (2.4)

Де р – плановий рівень рентабельності (10-15%). Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_п+П = 20948,32+2094,83 = 23043,15$ грн; (2.5)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р=Ц_о+ПДВ = 23043,15 +23043,15 *0.2 = 27651,78$ грн; (2.6)

3 ОХОРОНА ПРАЦІ

3.1 Вступ

Україна приділяє велику увагу питанням охорони життя і здоров'я своїх громадян, створенню безпечних умов праці роботодавцями, керівниками установ, організацій, проте кількість нещасних випадків, що трапляються на виробництві, залишається дуже великою.

Поліпшення умов та охорона праці стає одним з важливих напрямків матеріального та культурного рівня життя народу, а це, у свою чергу, сприяє зростанню якості та продуктивності праці, підвищенню соціально-економічних показників виробництва, зменшенню коштів на витрати від травматизму, професійних захворювань і аварій. Дипломним проектом передбачена розробка гри жанру шутер, тому в даному розділі розглянемо питання створення безпечних умов праці для користувача ПК.

3.2 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК при розробці даного програмного комплексу

В процесі роботи на користувачів ПК можуть мати вплив наступні небезпечні та шкідливі фактори:

- Невідповідність параметрів мікроклімату нормам;
- Недостатній рівень освітленості;
- Ураження електрострумом;
- Статична електрика;

3.3 Гігієнічні вимоги до виробничого середовища

Потрібно враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Конкретні показники

зазначених санітарних норм викладені в Державних санітарих правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98.

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп'ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп'ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу.

У разі надмірного шуму чи вібрації технічного обладнання, роботодавець повинен забезпечити працівників антивібраційними килимками.

Виробничі приміщення необхідно обладнати аптечками першої медичної допомоги.

3.4 Вимоги до організації робочого місця працівника

Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам. Розмір одного робочого місця має становити не менше 6 квадратних метрів. При необхідності, суміжні робочі місця співробітників, що працюють з комп'ютером, слід розділити перегородками висотою до 2 метрів. При визначенні достатнього розміру приміщення і робочого місця на одну особу необхідно додатково враховувати шафи, сейфи, тумби або інші предмети меблів чи обладнання, які знаходяться в кімнаті. На столі працівника можливо розмістити допоміжні для роботи пристрої (принтери, колонки, сканери), а також місця для зберігання документів, за умови, що це не обмежуватиме видимість екрану і не заважатиме працівнику. Робочий стілець співробітника має бути підйомно-поворотним, легко регульованим за висотою та забезпечувати належну підтримку та зручне положення спини і хребта особи. Щодня необхідно проводити вологе прибирання приміщення, та очищати робоче місце та безпосередньо монітор комп'ютера від запиленості.

На підприємстві забороняється: проводити ремонт та технічне обслуговування комп’ютера за робочим місцем працівника; самочинно ремонтувати або намагатись здійснити технічне налагодження комп’ютера без залучення компетентних спеціалістів; складувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи; використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану; працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою.

Допускати до роботи осіб, які не пройшли затверджений на підприємстві курс охорони праці для роботи з комп’ютером, не дозволяється. Переглянути візуалізацію вимог до організації робочого місця працівників можна на рисунку 3.1:

Рисунок 3.1. Візуалізація вимог до організації робочого місця працівника

3.5 Електробезпека

Вимоги електробезпеки у приміщеннях, де встановлені електронно-обчислювальні машини і персональні комп'ютери (далі — ЕОМ) відображені у *ДНАОП 0.00-1.31-99*. Відповідно до цього нормативного документу під час проектування систем електропостачання, монтажу основного електрообладнання та електричного освітлення будівель та приміщень для ЕОМ необхідно дотримуватись вимог Правил влаштування електроустановок *(ПВЕ), ГОСТ 12.1.006-84, ГОСТ 12.1.019-79, ГОСТ 12.1.030-81, ГОСТ 12.1.045-84, ПТЕ, ПБЕ,* ВСН 59-88 "Электрооборудованиежилых и общественных зданий. Нормы

проектирования", СН 357-77 "Инструкция по проектированию силового осветительного оборудования промишленныхпредприятий", Правил пожежної безпеки в Україні та інших нормативних документів, що стосуються штучного освітлення і електротехнічних пристроїв, а також вимог нормативно-технічної експлуатаційної документації заводу-виробника.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів і прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення

Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам ДНАОП 0.00-1.21-98 "Правила безпечної експлуатації електроустановок споживачів". Заземлені конструкції, що знаходяться у приміщеннях (батереї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), мають бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

Є неприпустимими:

— експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;

— застосування саморобних продовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;

— користування пошкодженими розетками, розгалужувальними та з’єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;

3.6 Пожежна безпека

Джерелами займання можуть бути електричні іскри, дуги, коротке замикання, струмові перевантаження, перегріті опірні поверхні, несправність обладнання. Окислювачем звичайно служить кисень. Але потужність і тривалість дії цих джерел займання порівняно малі, тому горіння, як правило, не розвивається. Виникнення пожежі в електронних пристроях можливо, якщо використовуються спалимі і важкоспалимі матеріали і вироби.

Забезпечення пожежної безпеки на підприємствах здійснюється наступними основними компонентами виробництва:

- технічною системою, яка передбачає надійність обладнання, використання безпечних технологій, визначає обсяг вибухопожежонебезпечних речовин, проектні рішення, впровадження систем виявлення та гасіння пожеж тощо;

- персоналом, його підготовкою, забезпеченням регламентами і правилами роботи;

- системою управління.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани - ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

Для гасіння пожеж на початкових стадіях широко застосовуються вогнегасники. У виробничих приміщеннях це головним чином вуглекислотні вогнегасники, достоїнством яких є висока ефективність гасіння пожежі, збереження електричного устаткування. Розташовують вогнегасники на видних місцях, на висоті не більше як 1,5 м від полу.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1,35 м від полу.

Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис «Запасний вихід». План евакуації вивішується на видному місці у основного виходу із приміщення.

ВИСНОВКИ

Метою дипломного проектування було реалізувати логіку для ігрових об’єктів та елементів інтерфейсу 2D-гри в жанрі top down shooter. Створення механізму взаємодії між ігровими елементами на сцені гри та елементами інтерфейсу. Взаємодію між гравцем та оточенням.

Для виконання поставленої задачі необхідно було створити макет 2D-гри в жанрі top down shooter. Для цієї цілі було використано ігровий двигун Unity, що має дуже широкий та зрозумілий функціонал, а також безкоштовну ліцензію для цілей освіти, науки та лімітованої комерції. Написання коду проводилось через середу розробки Visual Studio 2017, що також має вільну ліцензію для не комерційного використання.

Виконання роботи розпочалось з аналізу питання, пошуку необхідних даних, постановки задачі, проектування самої гри, а також майбутніх ігрових об’єктів, елементів інтерфейсу. Також було проведено проектування взаємодії кожного елементу гри, їх взаємозв’язків та принципів обміну даними.

Під час проектування було закладено базис для подальшого розвитку цифрової гри. Реалізовано менеджери ігрового процесу та ігрового інтерфейсу, які виконують важливу роль у розмежуванні візуальної частини гри та її скриптів. Практична робота була проведена згідно із спроектованими функційними схемами та алгоритмами.

В результаті було створено макет 2D-гри в жанрі top down shooter, в якому був реалізований основний ігровий функціонал, елементи інтерфейсу, ігрові об’єкти. Гравець може переміщуватись по рівню, виконувати постріли із гармат, як разом, так і роздільно. Взаємодіяти із оточенням, використовувати активізуємі об’єкти та підбирати одноразові об’єкті. Між ігровими об’єктами та інтерфейсом була реалізована система передачі даних.

Окрім виконання поставленого завдання, завдяки використання принципів модульної розробки, була закладена основа для майбутнього

удосконалення ігрових елементів, взаємодії між ігровими об’єктами, елементами інтерфейсу. Гнучкі можливості обраної мови програмування дають змогу в подальшому додавати до гри нові елементи, моделі та ігрові механіки, а закладені модульні принципи – швидко та якісно імплементувати новий ігровий код.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hocking J. Unity in Action: Multiplatform Game Development in C# with Unity 5. Shelter Island, New York: Manning Publications. 352 p.
2. Unity User Manual: [Website]. URL: https://docs.unity3d.com/Manual/index.html (viewed on: 21.05.2023).
3. Unity: [Website]. URL: https://unity.com/ru (viewed on: 21.05.2023).
4. C# docs: [Website]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (viewed on: 21.05.2023).

ДОДАТОК А Лістинг коду програми

Скрипт player_controller.cs

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

public class player_controler : MonoBehaviour
{
	CharacterController _character_controller;
	public GameObject player_mech_body;
	public float player_speed = 0.15f;

	void Start()
	{
		_character_controller = GetComponent<CharacterController>();
		if(player_mech_body == null)
		{
			Debug.LogError("NO PLAYER MECH BODY VARIABLE!");
		}
	}

	private void FixedUpdate()
	{
		float deltaX = Input.GetAxis("Horizontal") * player_speed;
		float deltaZ = Input.GetAxis("Vertical") * player_speed;
		Vector3 movement = new Vector3(deltaX, 0, deltaZ);
		movement = Vector3.ClampMagnitude(movement, player_speed);
		movement = transform.TransformDirection(movement);
		_character_controller.Move(movement);
	}
}
```

Скрипт mouse_tracking.cs

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

public class mouse_tracking : MonoBehaviour
{
	void Update()
	{
```

```
            Vector3 mouse_position = Input.mousePosition;
            mouse_position =
            Camera.main.ScreenToWorldPoint(Input.mousePosition);
            Vector3 rotation = mouse_position - transform.position;
            float rotY = Mathf.Atan2(rotation.x, rotation.z) * Mathf.Rad2Deg;
            transform.rotation = Quaternion.Euler(0, rotY, 0);
        }
}
```

Скрипт main_w_shooting.cs

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

public class main_w_shooting : MonoBehaviour
{
    [SerializeField] game_manager _game_manager;
    LineRenderer _lineRenderer;
    public bool main_w_overheated = false;

    void Start()
    {
        _lineRenderer = GetComponent<LineRenderer>();
    }

    void Update()
    {
        Ray _ray = new Ray(transform.position, transform.forward);
        if(!main_w_overheated)
        {
            if (Input.GetMouseButton(0))
            {
                RaycastHit _ray_hit_info;
                Physics.Raycast(_ray, out _ray_hit_info);
                _lineRenderer.enabled = true;
                _lineRenderer.SetPosition(0, transform.position);
                _lineRenderer.SetPosition(1, _ray_hit_info.point);
                _game_manager.incr_player_overheat(0.01f);
            }
            else
            {
                _game_manager.decr_player_overheat(0.005f);
                clean_line();
            }
        }
```

```
    }

    public void clean_line()
    {
        _lineRenderer.enabled = false;
    }
}
```

Скрипт second_w_shooting.cs

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

public class second_w_shooting : MonoBehaviour
{
    [SerializeField] game_manager _game_manager;
    LineRenderer _lineRenderer;
    public bool second_w_overheated = false;

    void Start()
    {
        _lineRenderer = GetComponent<LineRenderer>();
    }

    void Update()
    {
        Ray _ray = new Ray(transform.position, transform.forward);
        if(!second_w_overheated)
        {
            if (Input.GetMouseButton(1))
            {
                RaycastHit _ray_hit_info;
                Physics.Raycast(_ray, out _ray_hit_info);
                _lineRenderer.enabled = true;
                _lineRenderer.SetPosition(0, transform.position);
                _lineRenderer.SetPosition(1, _ray_hit_info.point);
                _game_manager.incr_player_overheat(0.01f);
            }
            else
            {
                _game_manager.decr_player_overheat(0.005f);
                clean_line();
            }
        }
    }
```

```
    public void clean_line()
    {
        _lineRenderer.enabled = false;
    }
}
```

Скрипт ui_manager.cs

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

public class ui_manager : MonoBehaviour
{
    [SerializeField] Slider hull_slider;
    [SerializeField] Slider overheat_slider;
    [SerializeField] Text overheated_core_line;
    [SerializeField] Text score_line;

    public void set_hull_max_value(float new_value)
    {
        hull_slider.maxValue = new_value;
    }

    public void set_hull_value(float new_value)
    {
        if(new_value <= hull_slider.maxValue)
        {
            hull_slider.value = new_value;
        }
    }

    public void set_overheat_max_value(float new_value)
    {
        overheat_slider.maxValue = new_value;
    }

    public void set_overheat_value(float new_value)
    {
        if (new_value <= overheat_slider.maxValue)
        {
            overheat_slider.value = new_value;
        }
    }
```

```
	public void show_overheat_line()
	{
		overheated_core_line.text = "!OVERHEATED CORE!";
	}

	public void hide_overheat_line()
	{
		overheated_core_line.text = "";
	}

	public void set_score(int value)
	{
		score_line.text = "Score: " + value;
	}
}
```

Скрипт game_manager.cs

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

public class game_manager : MonoBehaviour
{
	[SerializeField] ui_manager _ui_manager;
	[SerializeField] main_w_shooting _main_w_shooting;
	[SerializeField] second_w_shooting _second_w_shooting;
	[SerializeField] GameObject player_mech;
	public float player_overheat = 0;
	public float player_hull = 100;
	public int player_score = 0;

	public void incr_player_overheat(float value)
	{
		player_overheat += value;
		if(player_overheat > 100)
		{
			player_overheat = 100;
			Debug.Log("OVERHEATED!");
			StartCoroutine(overheated_core());
		}
	}

	public float _take_player_overheat()
	{
```

```
        return player_overheat;
}

public float _take_player_hull()
{
        return player_hull;
}

public void decr_player_overheat(float value)
{
        player_overheat -= value;
        if(player_overheat < 0)
        {
                player_overheat = 0;
        }
}

public void incr_player_hull(float value)
{
        player_hull += value;
        if(player_hull > 100)
        {
                player_hull = 100;
        }
}

public void decr_player_hull(float value)
{
        player_hull -= value;
        if(player_hull <= 0)
        {
                Destroy(player_mech);
        }
}

public void incr_player_score(int value)
{
        player_score += value;
        _ui_manager.set_score(player_score);
}

void Update()
{
        if(Input.GetKeyUp(KeyCode.Space))
```

```
            {
                decr_player_hull(20f);
            }
            _ui_manager.set_hull_value(player_hull);
            _ui_manager.set_overheat_value(player_overheat);
        }

        public IEnumerator overheated_core()
        {
            _main_w_shooting.main_w_overheated = true;
            _second_w_shooting.second_w_overheated = true;
            _main_w_shooting.clean_line();
            _second_w_shooting.clean_line();
            _ui_manager.show_overheat_line();
            yield return new WaitForSeconds(5f);
            Debug.Log("WAited");
            _main_w_shooting.main_w_overheated = false;
            _second_w_shooting.second_w_overheated = false;
            _ui_manager.hide_overheat_line();
        }
}
```

Скрипт pick_up.cs

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

public class pick_up : MonoBehaviour
{
    [SerializeField] game_manager _game_manager;
    private void OnTriggerEnter(Collider other)
    {
        GameObject entered_pick_up = other.gameObject;
        if(entered_pick_up.tag == "hull_rep")
        {
            _game_manager.incr_player_hull(25f);
            Destroy(entered_pick_up);
            _game_manager.incr_player_score(10);
        }
        else if(entered_pick_up.tag == "coolant")
        {
            _game_manager.decr_player_overheat(25f);
            Destroy(entered_pick_up);
            _game_manager.incr_player_score(10);
        }
```

```
        }

        private void OnTriggerStay(Collider other)
        {
            GameObject entered_pick_up = other.gameObject;
            if ((entered_pick_up.tag == "hull_deepo") &&
            (_game_manager._take_player_hull() < 100))
            {
                _game_manager.incr_player_hull(0.2f);
                _game_manager.incr_player_score(1);
            }
            else if ((entered_pick_up.tag == "coolant_deepo") &&
            (_game_manager._take_player_overheat() > 0))
            {
                _game_manager.decr_player_overheat(0.4f);
                _game_manager.incr_player_score(1);
            }
        }
    }
```

ДОДАТОК Б Слайди мультимедійної презентації

Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter

Дипломний проект студента групи 4РП-06: Попова Дениса Сергійовича

Керівник: Джабраілов Д.В.

Особливості ігрового жанру top down shooter

Ігровий жанр top down shooter має велику історію та з'явився одним із перших в ігровій індустрії, поклавши основу для майбутніх шутерів.

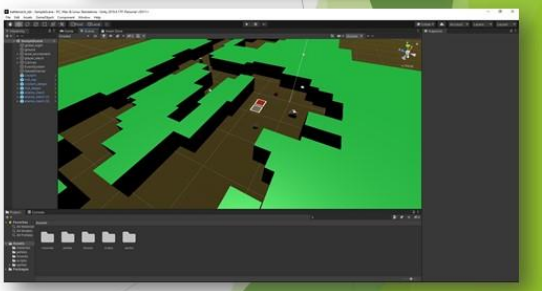
До особлививостей ігрового жанру відносять:

- Велику інтенсивність ігрового процесу;
- Гнучкі вимоги до навичок гравця;
- Гнучкі візуальні можливості;
- Ігри цього жанру реалізуються, як в 2D так і 3D;
- Орієнтація камери зверху-вниз;
- Вільне переміщення по рівню;
- Взаємодія з оточенням;

Особливості ігрового двигуна Unity

Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі, який почав свій шлях у 2005-му році та продовжує розвиватись!

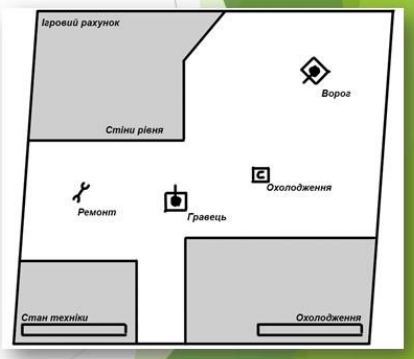
- Має зручну візуальну середу розробки;
- Має можливість міжплатформенної розробки ігор;
- Підтримує модульну систему компонентів;
- Використання мови програмування C# та його можливостей;
- Велике ком'юніті розробників;
- Зрозуміла та вичерпна документація, безліч курсів у інтернеті та літературі;
- Гнучка система ліцензування ігрового двигуна;
- Безплатні та платні ассети для проектів будь-якого рівня.



Етапи створення макету 2D-гри в жанрі top down shooter

Макет гри є необхідним для виконання завдання дипломного проектування. Макетом гри вважається проект гри, виконаний на Unity, для якого будуть реалізовуватись задачі дипломного проектування. Є наступні етапи створення та розробки макету 2D-гри:

1. Розробка концепту ігрового процесу;
2. Виготовлення спрайтів для гри;
3. Виготовлення матеріалів для гри;
4. Створення рівня за допомогою інструментів Unity;
5. Створення та розміщення ігрових об'єктів, об'єкта гравця;
6. Створення інтерфейсу



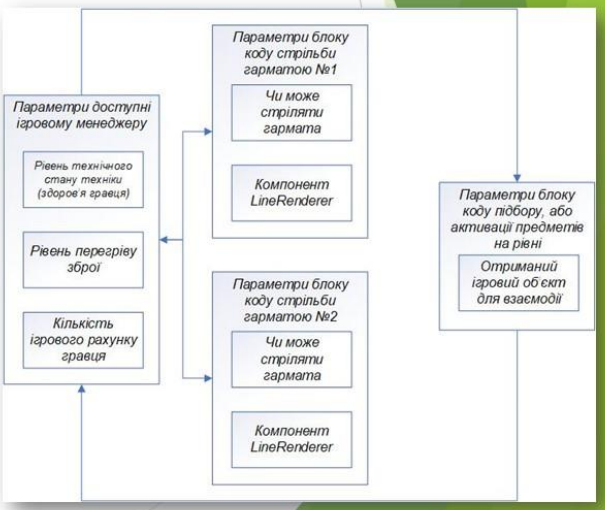
Принцип роботи модулів логіки ігрових об’єктів проекту

Ігровими об’єктами в іграх називають об’єкти, з якими гравець може взаємодіяти. Такими можуть бути об’єкти, що дають ресурси, вороги, або точки збереження.

В даному проекті, згідно концепту реалізовуються об’єкти для відновлення ігрових ресурсів гравця.

Для роботи і взаємодії таких об’єктів використовується «Ігровий менеджер», що керує взаємодією ігрових об’єктів, гравця та інтерфейсу.

На функційній схемі зображені модулі ігрових об’єктів та їх елементи, з якими взаємодіє «Ігровий менеджер».



Принцип роботи модулів логіки ігрових об’єктів проекту

«Ігровий менеджер» має доступ до компонентів ігрових об’єктів та завдяки існуючих зв’язків із ними може керувати їх поведінкою, або створювати зворотній зв’язок від ігрових об’єктів до гравця.

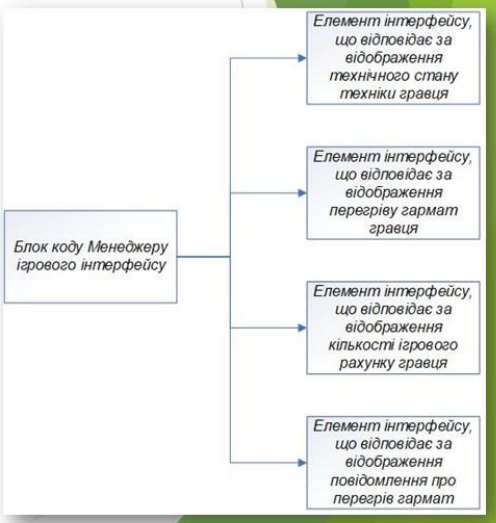
Наприклад код, що реалізує використання одноразових ігрових об’єктів для відновлення ресурсів гравця, при контакті об’єкту гравця із ігровим об’єктом на рівні зчитує його тег. Якщо тег є одним із тих, що активують ту, чи іншу дію, викликається «Ігровий менеджер», що виконує цільову дію.

Таким самим чином виконується буд-яка взаємодія об’єктів у грі завдяки посереднику «Ігровому менеджеру».

Принцип роботи модулів роботи елементів інтерфейсу проекту

Елементи інтерфейсу гри існують окремо від ігрової сцени. Для взаємодії із ними реалізовано «Менеджер ігрового інтерфейсу», і тільки він має доступ до елементів інтерфейсу.

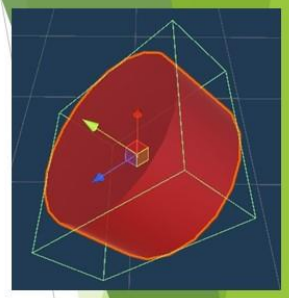
У разі потреби, «Ігровий менеджер» звертається до «Менеджеру ігрового інтерфейсу» та передає йому команду на зміну того чи іншого елементу інтерфейсу. Після чого саме «Менеджер ігрового інтерфейсу» змінює елементи інтерфейсу.



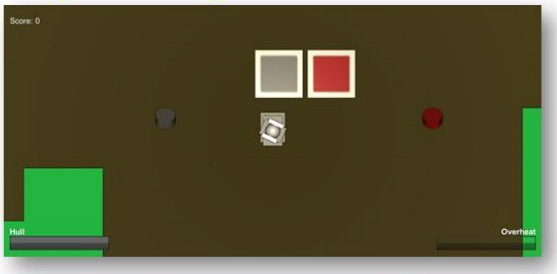
Етапи розробки логіки для ігрових об'єктів та елементів інтерфейсу

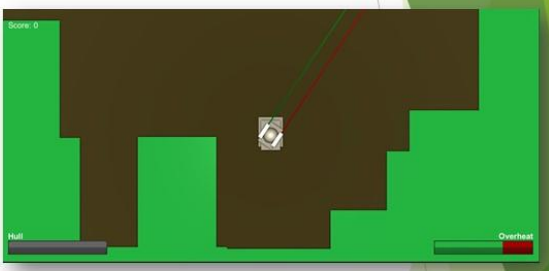
Після проектування роботи всіх модулів, процес розробки зводиться до декількох етапів:

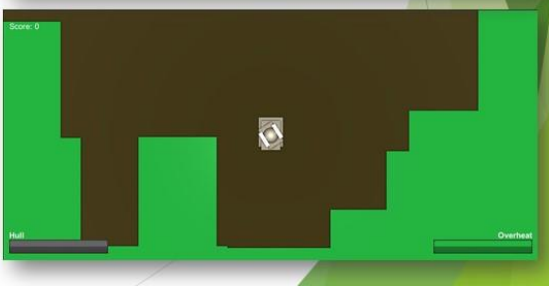
1. Написання коду скриптів визначених модулів;
2. Додавання скриптів до відповідних ігрових об'єктів;
3. Створення зв'язків між ігровими об'єктами та їх компонентами;
4. Відладка та тестування роботи.



Скріншоти результатів розробки проекту







Дякую за Вашу увагу!

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015519272

Дата перевірки:
09.06.2023 08:12:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2023 08:14:49 EEST

ID користувача:
100011688

Назва документа: 4РП-06 Попов Д.С

Кількість сторінок: 56 Кількість слів: 8411 Кількість символів: 61382 Розмір файлу: 2.83 MB ID файлу: 1015173754

17.7%
Схожість

Найбільша схожість: 4.87% з Інтернет-джерелом (http://elartu.tntu.edu.ua/bitstream/lib/36792/1/mag2021_StashukV.pd.

17.7% Джерела з Інтернету | 629 ..... Сторінка 58

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 24

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти
відділення комп'ютерних систем

Попова Дениса Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність 121 " Інженерія програмного забезпечення "

Освітня програма «Розробка програмного забезпечення»

Керівник дипломного проекту (роботи) Джабраілов Дмитро Володимирович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter*

Обсяг розрахунково-пояснювальної записки 76 сторінок

Обсяг графічної (презентаційної) частини 10 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню
Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для ігрової індустрії та присвячена питанням реалізації логіки для ігрових об'єктів та ігрового дизайну.

б) характеристика виконання кожного розділу дипломного проекту (роботи)
Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі Проведено аналіз існуючих ігрових аналогів реалізації елементів завдання проекту, розроблена ігрова концепція. Спроектовано та реалізовано макет гри для виконання завдання, реалізовані ігрові елементи, впроваджено ігрову логіку для об'єктів на рівні, елементів інтерфейсу.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на відмінному рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детальний опис логіки ігрових об’єктів;
2. Проведено проектування та реалізація макету гри для виконання завдань проектування;
3. Присутній опис основних модулів.

д) основні недоліки дипломного проекту (роботи)

1. Ігровий проект має обмежений функціонал з точки зору взаємодії із ворогами на рівні.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім’я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв’язку, к.ф.н., доцент кафедри КБ та ТЗІ, пом.декану факультету інформаційних технологій та кібербезпеки

Підпис:

«16» червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Попова Дениса Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: 121 "Інженерія програмного забезпечення"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 76 сторінок. У пояснювальній записці виконано опис предметної області, інструменти розробки для виконання теми проектування. Проведено проектування та реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри в жанрі top down shooter. Графічна частина складається з 10 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Попов Д.С. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Попов Д.С. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Попов Д.С. мав змогу самостійно приймати рішення з реалізації логіки ігрових об'єктів та елементів інтерфейсу, показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою Visual Studio а також в повній мірі використовувати можливості ігрового двигуна Unity.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Джабраілов Дмитро Володимирович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач комісії комп'ютерних технологій та програмної інженерії

Підпис

«09» 06 2023 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Попов Денис Сергійович,
здобувач освіти гр. 4РП-06, та

Джабраілов Дмитро Володимирович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту молодшого спеціаліста на тему:

«Реалізація логіки для ігрових об'єктів та елементів інтерфейсу 2D-гри у жанрі top down shooter» (автор роботи – Попов Д.С., керівник роботи – Джабраілов Д.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Попов Д.С. /

Керівник _____ / Джабраілов Д.В./

«09» 06 2024 р.