МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІГРОВИЙ ЗАСТОСУНОК У ЖАНРІ FIRST PERSON SHOOTER (FPS) ARENA

Спеціальність 122 «Комп'ютерні наук»

122 – КРБ – 401.2010127

Виконав студент 4-го курсу, групи 401
_____ О. В. Фоменко
«17» червня 2024 р.

Керівник: канд. фіз.-мат. наук, доцент
_____І. В. Кулаковська
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **12 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Фоменку Олексію Вікторовичу.

1. Тема кваліфікаційної роботи «ігровий застосунок у жанрі First Person Shooter (FPS) Arena».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз.-мат. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: концепт сюжету гри, 3Д-моделі ігрових об'єктів, аудіо-доріжки (музика, звуки персонажу тощо)

Очікуваний результат: ігровий застосунок у жанрі First Person Shooter (FPS) Arena

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

1) аналіз існуючих ігрових застосунків у жанрі FPS Arena: огляд відомих ігор, що вже існують на ринку, а також їхні особливості та механіка гри;

2) вивчення основ розробки на платформі Unity: для успішної розробки відіграють ключову роль знання інструментів та можливостей платформи Unity;

3) проектування геймплею і механіки гри: розробка концепції гри, включаючи управління гравця, зброю, арену та ігрові режими;

5. Перелік графічного матеріалу: 23 рисунка, презентація.

6. Завдання до спеціальної частини: «Рівень освітленості в домашньому офісі»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __канд. фіз.-мат. наук, доц. Кулаковська І. В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Фоменком О. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Ігровий застосунок у жанрі First Person Shooter (FPS) Arena_____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	15.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз сучасного стану ігрової індустрії, огляд існуючих технологій, розробка ігрового застосунку	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	Виконано

Розробив студент _____Фоменко О. В._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи _канд. фіз.-мат. наук, доц. Кулаковська І. В.___ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202__ р.

АНОТАЦІЯ

кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили
Фоменка Олексія Вікторовича
Тема: «Ігровий застосунок у жанрі First Person Shooter (FPS) Arena»

Актуальність: актуальність проекту обумовлена зростаючою популярністю ігрових технологій у різних сферах, включаючи освіту, розваги та тренування. Ігри жанру FPS популярні серед гравців і можуть використовуватися для навчання та тренувань.

Об'єкт роботи – процес розробки ігрового застосунку.

Предмет роботи – технології розробки ігрових застосунків за допомогою ігрового двигуна Unity.

Метою кваліфікаційної роботи є дослідження та реалізація різних аспектів розробки гри, включаючи системи управління гравцем, поведінку ворогів, фізичну модель гри, проектування ігрових арен, а також проведення аналізу геймплею.

Пояснювальна записка складається зі вступу, трьох розділів, висновків та додатків.

У першому розділі розглядається сучасні тренди та особливості популярних ігрових застосунків.

У другому розділі досліджено технології та інструменти для розробки ігрових застосунків.

У третьому розділі описано створення ігрового застосунку у жанрі First Person Shooter (FPS) Arena, які додаткові інструменти використовувалися та процес тестування гри.

В результаті розроблено ігровий застосунок у жанрі FPS Arena.

Кваліфікаційна робота містить 57 сторінок, 21 рисунків, _0 таблиць, _9 використаних джерел та _1 додатків.

Ключові слова: Unity, ігровий застосунок, FPS Arena, аналіз геймплею.

ABSTRACT

qualification work of a student of group 401
of BSNU named after Petro Mohyla
Oleksii Fomenko
Subject: "First Person Shooter (FPS) Arena game application"

Relevance: the relevance of the project is due to the growing popularity of game technologies in various fields, including education, entertainment and training. FPS games are popular among gamers and can be used for training and training.

The object of the work is the process of developing a game application.

The subject of the work is game application development technology using the Unity game engine.

The purpose of the qualification work is to research and implement various aspects of game development, including player control systems, enemy behavior, physical game modeling, designing game arenas, and conducting gameplay analysis.

The explanatory note consists of an introduction, three sections, conclusions and appendices.

The first section examines modern trends and features of popular game applications.

The second chapter explores technologies and tools for game application development.

The third chapter describes the creation of a First Person Shooter (FPS) Arena game application, the additional tools used and the game testing process.

As a result, a game application in the FPS Arena genre was developed.

The qualification work contains __ pages, __ figures, __ tables, __ used sources and __ appendices.

Keywords: Unity, game application, FPS Arena, gameplay analysis, game industry.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CS 2 – Counter-Strike 2

FPS – First Person Shooter

PGL – Professional Gamers League

ВСТУП

Під час роботи над кваліфікаційною роботою буде здійснений аналіз для подальшої розробки ігрового застосунку у жанрі FPS Arena на платформі Unity. Практична частина проекту була успішно реалізована, проведено значну роботу над теоретичною складовою.

Актуальність даного проекту полягає в тому, що ігрові технології набувають все більшої популярності в різних сферах, включаючи освіту, розваги та тренування. Ігрові застосунки в жанрі FPS (від англ. First-Person Shooter) особливо популярні серед гравців та використовуються як для розваг, так і для вирішення певних навчальних або тренувальних завдань. Створення такого застосунку може виявитися корисним як для навчальних закладів, так і для індивідуальних користувачів, які бажають розвивати навички в галузі розробки ігрових продуктів.

Для виконання роботи необхідно виконати наступне:

1) аналіз існуючих ігрових застосунків у жанрі FPS Arena: оглядалися відомі ігри, що вже існують на ринку, а також їхні особливості та механіка гри;

2) вивчення основ розробки на платформі Unity: для успішної розробки відіграють ключову роль знання інструментів та можливостей платформи Unity;

3) проектування геймплею і механіки гри: було розроблено концепцію гри, включаючи управління гравця, зброю, арену та ігрові режими.

Метою роботи була розробка ігрових застосунків та засвоєння процесу проектування та створення гри в жанрі FPS Arena. Також важливим аспектом було розвиток навичок роботи з платформою Unity та здобуття розуміння основ геймдизайну.

1 АНАЛІЗ СУЧАСНИХ ТРЕНДІВ У РОЗРОБЦІ ІГРОВИХ ЗАСТОСУНКІВ

Аналіз сучасних трендів у розробці ігрових застосунків у жанрі FPS дає змогу зрозуміти напрямки розвитку індустрії та визначити ключові особливості, які впливають на створення нових проектів.

1.1 Основні особливості сучасних ігрових застосунків

1. **Кросплатформеність**: завдяки розвитку хмарних технологій і платформ, таких як Google Stadia та Xbox Cloud Gaming, ігрові студії все більше звертають увагу на розробку ігор, які можна грати на різних пристроях без втрати якості.

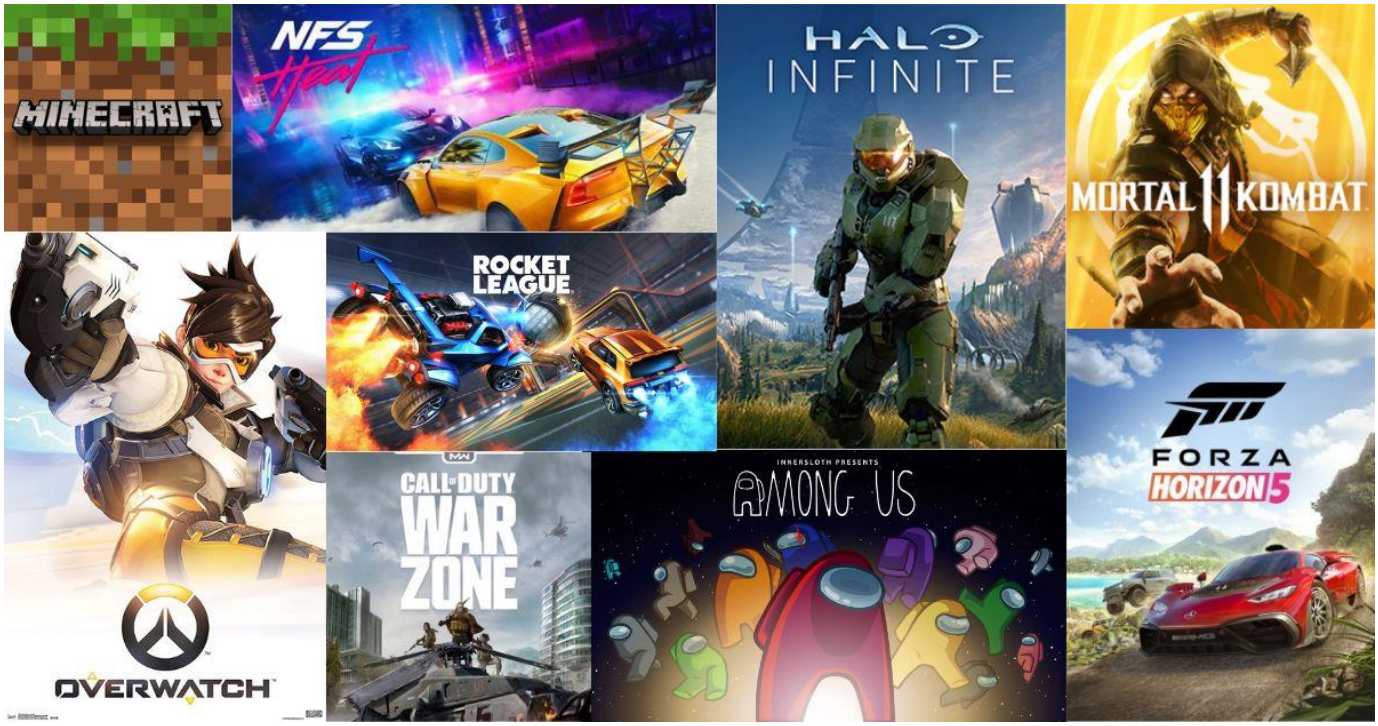


Рисунок 1.1 – Приклади кросплатформених ігор

2. **Підтримка режиму онлайн**: багатокористувацькі онлайн-ігри займають центральне місце в індустрії. Розробники активно впроваджують нові механіки гри, змагальний режим і співпрацю для залучення гравців.

3. **Постійне оновлення та додатковий контент**: випуск регулярних оновлень, нових режимів, карт та інших додаткових матеріалів стає стандартом для багатьох ігор у жанрі FPS. Це сприяє підтримці ігрового середовища та збільшенню грального активності.

Рисунок 1.2 – Гра, яка отримала нагороду року за постійну підтримку та оновлення контенту

4. **Інтеграція соціальних елементів**: соціальні функції, такі як можливість спілкування з іншими гравцями, спільні досягнення та можливість ділитися враженнями від гри, стають все більш важливими для залучення та утримання аудиторії.

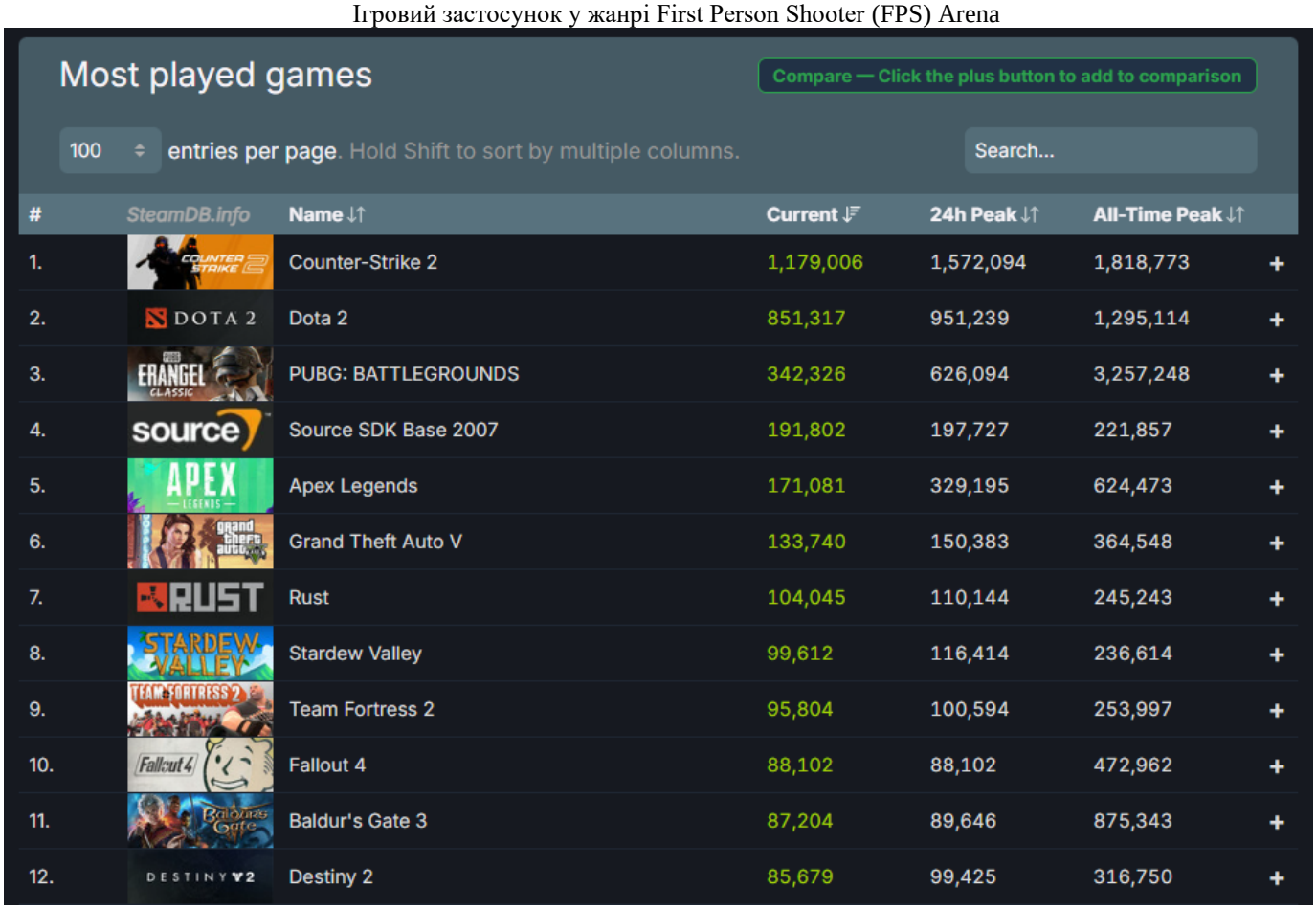
5. **Експерименти зі стилістикою та сюжетом**: розробники все частіше виходять за рамки традиційних тем та настроїв, експериментуючи з альтернативними світами, арт-стилями та сюжетами, щоб привернути увагу гравців.

Рисунок 1.3 – SCORN, незвичайним візуальним стилем гра зобов'язана роботами Ханса Гігера та Здзіслава Бексинського

1.2 Статистичні дані найпопулярніших FPS ігор

1.2.1 Гра Counter Strike 2

Однією з найпопулярніших ігор у жанрі FPS є CS 2 (Counter-Strike 2), яка демонструє найвищий офіційний онлайн серед усіх ігор як постійний так й найбільший протягом 24 годин. Також ця гра знаходиться на 3 місці серед найбільшої кількості гравців одночасно за весь час, але на першому місці також знаходиться гра у жанрі FPS[3].



#	SteamDB.info	Name	Current	24h Peak	All-Time Peak
1.		Counter-Strike 2	1,179,006	1,572,094	1,818,773
2.		Dota 2	851,317	951,239	1,295,114
3.		PUBG: BATTLEGROUNDS	342,326	626,094	3,257,248
4.		Source SDK Base 2007	191,802	197,727	221,857
5.		Apex Legends	171,081	329,195	624,473
6.		Grand Theft Auto V	133,740	150,383	364,548
7.		Rust	104,045	110,144	245,243
8.		Stardew Valley	99,612	116,414	236,614
9.		Team Fortress 2	95,804	100,594	253,997
10.		Fallout 4	88,102	88,102	472,962
11.		Baldur's Gate 3	87,204	89,646	875,343
12.		Destiny 2	85,679	99,425	316,750

Рисунок 1.4 – Рейтинг найпопулярніших ігор у Steam

CS 2 є однією з провідних ігор в кіберспорті. Турніри з великими призовими фондами, такі як Major Championships, залучають величезну кількість гравців та глядачів з усього світу. Останній такий турнір «PGL CS2 Major Copenhagen 2024» проходив з 17 по 31 березня 2024 року на Роял Арені у Копенгагені. Призовий фонд склав 1’250’000 $. Це стимулює гравців підтримувати високий рівень майстерності та постійно тренуватися, що, в свою чергу, збільшує загальний онлайн гри [11].

S, A, B, C-Tier Tournaments [edit]

Tier	G & S	Tournament	Date	Prize Pool	P#	Location	Winner	Runner-up
A-Tier		Skyesports Championship 2024	Jul 23 - 29, 2024	$200,000	8	Mumbai	TBD	TBD
C-Tier		IMAGINELEAGUE 2024	Jul 27, 2024	$8,624.20	8	João Pessoa	TBD	TBD
B-Tier		Thunderpick World Championship 2024: South American Series #2	Jul 17 - 22, 2024	$25,000	16	South America	TBD	TBD
S-Tier		Esports World Cup 2024	Jul 17 - 21, 2024	$1,000,000	15	Riyadh	TBD	TBD
C-Tier		EPIC.LAN 42	Jul 18 - 21, 2024	-		Kettering	TBD	TBD
C-Tier		Ace South American Masters Fall 2024	Jul 18 - 21, 2024		8	South America	TBD	TBD
C-Tier		Tipsport Cup Summer 2024: Online Stage	Jul 18 - 21, 2024		8	Czechia Slovakia	TBD TBD	TBD TBD
C-Tier		DACH CS Masters Season 1: Division 3	Apr 01 - Jun 23, 2024	$866.22	40	Germany	TBD	TBD
C-Tier		Dust2 Brasil Liga Season 3: Division 1	May 24 - Jun 19, 2024	$1,080.81	8	South America	TBD	TBD
C-Tier		Dust2.dk Ligaen Season 26	Apr 15 - Jun 19, 2024	$11,623.34	8	Denmark	TBD	TBD
B-Tier		Regional Clash Arena CIS	Jun 04 - 18, 2024	$60,000	16	CIS	TBD	TBD
B-Tier		Regional Clash Arena Europe	Jun 04 - 18, 2024	$60,000	16	Europe	TBD	TBD
B-Tier		Regional Clash Arena South America	Jun 04 - 18, 2024	$60,000	16	South America	TBD	TBD
S-Tier		BLAST Premier: Spring Final 2024	Jun 12 - 16, 2024	$425,000	8	London	TBD	TBD
A-Tier		FiReLEAGUE Global Finals 2023	Jun 14 - 16, 2024	$150,000	8	Buenos Aires	TBD	TBD
A-Tier		ESL Challenger at DreamHack Summer 2024	Jun 14 - 16, 2024	$100,000	8	Jönköping	TBD	TBD
B-Tier		CCT Season 2 European Series #5	Jun 03 - 16, 2024	$50,000	24	Europe	TBD	TBD
B-Tier		CCT Season 2 North American Series #1	Jun 03 - 16, 2024	$25,000	24	North America	TBD	TBD
C-Tier		DreamHack Summer 2024 BYOC	Jun 14 - 16, 2024	$6,856.11	-	Jönköping	TBD	TBD
C-Tier		Dust2 Brasil Liga Season 3	May 01 - Jun 16, 2024	$10,808.06	8	South America	TBD	TBD
C-Tier		ESEA Season 49: Main Division - North America	Apr 09 - Jun 16, 2024	$10,000	128	North America	TBD	TBD

Рисунок 1.5 – Частина найближчих турнірів по «Counter Strike 2» на 26.05.2024[4]

1.2.2 Гра Valorant

Серед багатьох ігор цього жанру значним конкурентом для великої серії ігор Counter Strike виступає гра Valorant, яка була розроблена і випущена компанією Riot Games у 2020 році. Геймплейні особливості цієї гри будуть розглянуті у наступному розділі. У цьому розглянемо статистику щодо гравців та турнірів і їх фінансування.

Valorant має значно більшу загальну кількість гравців [5], вони менш активні, проте середня кількість гравців більше саме у Valorant, а не CS 2. Причиною можуть бути більш простіша графіка та особливості геймплею.

Date	Players Count	Gain	% Gain
May 2024	17,796,957	...	...
April 2024	19,464,085	-120,606	-0.6%
March 2024	19,584,691	+1,555,597	+8.6%
February 2024	18,029,094	-289,712	-1.6%
January 2024	18,318,806	+1,198,607	+7.0%
December 2023	17,120,199	+653,828	+4.0%
November 2023	16,466,371	-810,529	-4.7%
October 2023	17,276,900	-1,222,156	-6.6%
September 2023	18,499,056	-1,395,444	-7.0%
August 2023	19,894,500	-520,251	-2.5%
July 2023	20,414,751	+347,922	+1.7%
June 2023	20,066,829	+231,720	+1.2%
May 2023	19,835,109	-264,966	-1.3%
April 2023	20,100,075	-24,180	-0.1%
March 2023	20,124,255	+1,827,859	+10.0%

Рисунок 1.6 – Кількість гравців (зареєстрованих акантів) у грі протягом місяців

Також по Valorant проводяться турнірі. Їхні призові фонди трохи менші, проте гра значно молодше свого конкурента, на цілих 8 років й це лише у порівнянні з останньої частиною серії. Кількість турнірів та їх розмах швидко збільшуються. Крім того більшість професійних гравців переходять у Valorant з CS 2 через низку різних причин серед яких неоптимізованість гри, більша ігрова статистика під час та після матчу, що значно полегшує аналіз навичок та їх покращення [12].

S, A-Tier Tournaments [edit]

Tier	$	Tournament	Date	Prize Pool	Location	P#	Winner	Runner-up
S-Tier		VCT 2024: Americas Stage 1	Apr 6 - May 12, 2024		Los Angeles	11	100 Thieves	G2 Esports
S-Tier		VCT 2024: Pacific Stage 1	Apr 6 - May 12, 2024		Seoul	11	Paper Rex	Gen.G Esports
S-Tier		VCT 2024: China Stage 1	Apr 5 - May 12, 2024		Shanghai	11	EDward Gaming	FunPlus Phoenix
S-Tier		VCT 2024: EMEA Stage 1	Apr 3 - May 12, 2024		Berlin	11	Fnatic	Team Heretics
A-Tier		VALORANT Challengers 2024 Brazil: Split 1	Feb 14 - May 2, 2024	$77,947.59	São Paulo	10	Hero Base	RED Canids
A-Tier		VALORANT Challengers 2024 Korea: Split 1	Feb 28 - Apr 19, 2024	$56,346.97	Seoul	8	LFG Portal	SLT
A-Tier		VALORANT Challengers 2024 Japan: Split 1	Feb 3 - Mar 31, 2024	$33,037.09	Nagoya	8	FENNEL	REJECT
S-Tier		VCT 2024: Masters Madrid	Mar 14 - 24, 2024	$500,000	Madrid	8	Sentinels	Gen.G Esports
S-Tier		VCT 2024: Americas Kickoff	Feb 16 - Mar 3, 2024		Los Angeles	11	Sentinels	LOUD
S-Tier		VCT 2024: China Kickoff	Feb 22 - Mar 2, 2024		Shanghai	11	EDward Gaming	FunPlus Phoenix

B-Tier Tournaments [edit]

Tier	$	Tournament	Date	Prize Pool	Location	P#	Winner	Runner-up
B-Tier		VCT 2024: Game Changers Korea Stage 1	May 20 - 22, 2024	$7,328.26	Korea	4	Mir Gaming GC	Beyond Stratos Gaming GC
B-Tier		VALORANT Challengers 2024: North America: Mid-Season Cup	May 15 - 19, 2024	$30,000	North America	12	Oxygen Esports	Turtle Troop
B-Tier		VALORANT Challengers 2024 South Asia: Split 1 - Cup 2	Apr 3 - May 19, 2024	$33,000	New Delhi	8	MLT Esports	True Rippers
B-Tier		VCT 2024: Game Changers Brazil Series 1	Apr 15 - May 16, 2024	$15,579.04	São Paulo	8	Team Liquid Brazil	MIBR GC
B-Tier		VALORANT Challengers 2024 DACH: Evolution Split 1	Feb 10 - May 4, 2024	$26,929.50	Cologne	10	CGN Esports	MOUZ
B-Tier		VCT 2024: Game Changers Japan Split 1	Apr 30 - May 2, 2024	$6,412.23	Japan	4	ZETA DIVISION GC	FENNEL Female
B-Tier		VALORANT Challengers 2024 Oceania: Split 1	Mar 25 - May 2, 2024	$20,000	Oceania	16	JiJieHao BONK	ARENA Internet Cafe
B-Tier		VALORANT Challengers 2024 Northern Europe: Polaris Split 1	Feb 1 - Apr 22, 2024	$10,664.90	United Kingdom, Nordic Countries, Ireland	8	Apeks	SweetNSour

Рисунок 1.7 – Частина найближчих турнірів по «Valorant» на 26.05.2024[4]

Обидві гри демонструють високий рівень залученості гравців та є ключовими представниками жанру FPS на сучасному ринку відеоігор. Їхня конкуренція сприяє постійному вдосконаленню та інноваціям, що вигідно для всіх учасників ринку - від гравців до розробників. Це підтверджує, що CS 2 та Valorant не лише утримують свої позиції, але й продовжують активно розвиватися, залучаючи нових гравців та створюючи насичений та конкурентний кіберспортивний ландшафт.

1.3 Огляд існуючих ігор у жанрі FPS Arena та їхніх особливостей

Жанр FPS Arena представляє собою піджанр шутерів з від першої особи, де гравець бере участь у боях на великих відкритих аренах. Особливість цього жанру полягає в інтенсивних, динамічних боях та стратегічному використанні різноманітних збройних систем та елементів оточення. Оглянемо деякі з найвідоміших ігор у цьому жанрі та їхні особливості.

1. **Quake III Arena**:

– одна з найбільш впливових ігор у жанрі;

– швидкість та динаміка гри є основними особливостями;

– чітко виокремлені режими гри, включаючи Deathmatch, Team Deathmatch, Capture the Flag тощо;

– гра акцентується на навичках гравця та стратегічному мисленні.

2. DOOM (2016):

– нова версія класичної гри DOOM приносить інтенсивні та кровопролитні бої;

– головна особливість - швидка гра з використанням широкого арсеналу руйнівної зброї та динамічних рухів.

3. Valorant:

– компетитивний FPS Arena з великим акцентом на тактичний геймплей та стратегію;

– комбінація стрілянини та використання навичок персонажів робить гру складною та цікавою;

– система еккіпажування та економіки додає стратегічних елементів до гри.

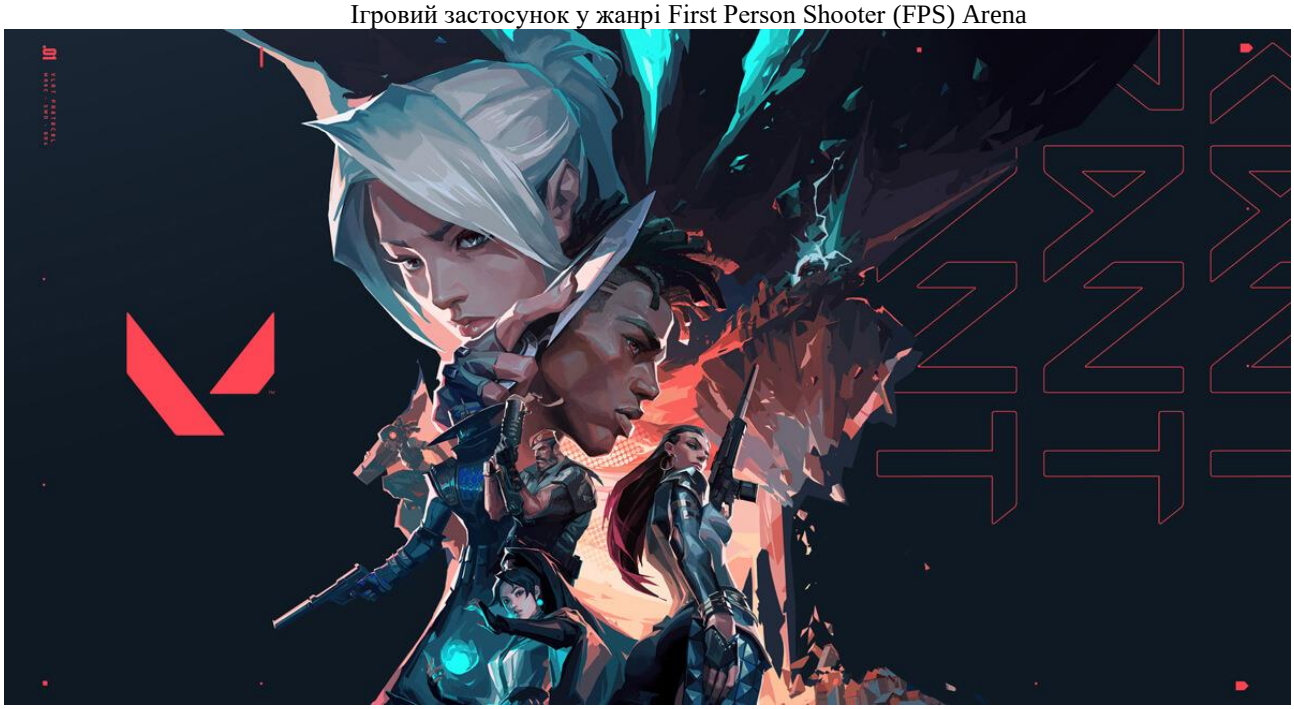


Рисунок 1.8 – Один з найпопулярніших FPS у наш час

4. **Overwatch**:

– командний FPS Arena з унікальними героями та їхніми вміннями;

Рисунок 1.9 – Усі унікальні персонажі гри Overwatch

– зосереджується на співпраці команд та стратегічному використанні ролей героїв;

– графіка в стилі коміксів та розмаїтість карт роблять гру візуально привабливою.

Кожна з цих ігор має свої особливості, але загальна риса полягає у високій динаміці, стратегічному мисленні та навичках гравця.

1.4 Постановка задачі

Метою роботи є розробка ігрового застосунку у жанрі FPS Arena на платформі Unity, що забезпечує динамічний геймплей, високий рівень графіки та інтерактивності.

Для досягнення цієї мети необхідно створити концепцію гри, включаючи основні елементи геймплею, механіки бою, карти та режими гри. Особливу увагу слід приділити дизайну персонажів та зброї, враховуючи можливості кастомізації.

Важливим етапом роботи є вибір та вивчення інструментів розробки. Необхідно вивчити можливості Unity Engine для реалізації задуманого проекту, освоїти програмування на мові C# для створення необхідних скриптів та ігрової логіки, а також оцінити додаткові інструменти та плагіни, що можуть бути корисними у процесі розробки.

Після цього слід розпочати реалізацію ігрового застосунку, створивши прототип гри з основними функціональними можливостями. Необхідно оптимізувати графіку та продуктивність гри для забезпечення плавного геймплею на різних платформах.

На етапі тестування та вдосконалення потрібно провести тестування прототипу гри для виявлення багів та недоліків, зібрати зворотний зв'язок від тестувальників та внести необхідні корективи. Важливо вдосконалити ігрову механіку та графіку на основі отриманих даних.

Завершальним етапом є підготовка технічної документації проекту, включаючи опис архітектури гри, використаних технологій та інструментів. Необхідно описати процес розробки та реалізації основних функцій гри.

Очікувані результати включають створення функціонального прототипу ігрового застосунку у жанрі FPS Arena, набуття навичок розробки ігор на платформі Unity та програмування на мові C#, розуміння основних принципів геймдизайну та розробки мультиплеєрних ігор, а також отримання зворотного зв'язку та вдосконалення гри на основі тестування. Ця робота сприятиме розвитку

практичних навичок у сфері розробки ігрових застосунків та надасть досвід створення складних інтерактивних продуктів у жанрі FPS Arena.

Висновки до розділу 1

У розділі було проведено детальний аналіз сучасних трендів у розробці ігрових застосунків у жанрі FPS. Розглянуті аспекти включали основні особливості сучасних ігор, статистичні дані найпопулярніших FPS ігор та огляд існуючих ігор у жанрі FPS Arena. Проведений аналіз вказує на те, що сучасні тренди в розробці ігрових застосунків спрямовані на створення глибоких, інтерактивних та соціально орієнтованих ігор. FPS ігри, особливо в жанрі Arena, продовжують залишатися популярними завдяки їхній динамічній механіці та постійній підтримці з боку розробників. Обидві розглянуті ігри, CS 2 та Valorant, демонструють високу активність гравців та розмах кіберспортивних турнірів, що свідчить про їхню конкурентоспроможність і актуальність у сучасному ігровому світі.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ НА UNITY

Unity — це міжплатформний ігровий двигун, розроблений компанією Unity Technologies. Вперше був представлений у червні 2005 року на Apple Worldwide Developers Conference як ексклюзивний інструмент для macOS. Unity швидко завоював популярність завдяки своїй доступності, потужним інструментам для розробки 2D і 3D ігор та можливості експорту на різні платформи, включаючи Windows, macOS, iOS, Android, консолі та інші. [9]

Сьогодні Unity використовується як незалежними розробниками, так і великими ігровими студіями для створення ігор, застосунків віртуальної реальності (VR) та доповненої реальності (AR), а також інтерактивних середовищ у різних галузях, таких як архітектура, фільми та анімація.

2.1 Ігровий двигун Unity та додаткові елементи розробки

1. **Unity Engine**: є основною платформою для розробки ігор, що надає інтегроване середовище розробки, яке дозволяє створювати графічно складні та динамічні ігри. Unity підтримує розробку як 2D, так і 3D ігор, надаючи розробникам потужні інструменти для роботи з графікою, анімацією, фізикою та штучним інтелектом. За допомогою Unity розробники можуть створювати реалістичні візуальні ефекти, складні системи взаємодії об'єктів та різноманітні ігрові механіки. Платформа також підтримує експорт проектів на різні платформи, включаючи PC, консолі, мобільні пристрої та VR/AR середовища.

2. **C# Programming Language**: для програмування на Unity основною мовою використовується C#. Це потужна мова програмування, яка надає багато можливостей для розробників. C# підтримує об'єктно-орієнтоване програмування, що дозволяє створювати складні та модульні системи. Мова має велику спільноту розробників, що сприяє активному обміну знаннями та ресурсами. Завдяки широкій підтримці бібліотек і фреймворків, C# дозволяє реалізовувати різноманітні функції та оптимізувати продуктивність ігрових додатків.

3. **Unity Asset Store**: онлайн-магазин, де розробники можуть придбати або завантажити безкоштовні ресурси або придбати їх для своїх проектів. Ассети включають моделі персонажів, текстури, звукові ефекти, скрипти, розширення та багато іншого. Використання готових ресурсів з Asset Store дозволяє значно прискорити процес розробки, зменшити витрати на створення контенту та покращити якість гри. Розробники можуть знайти як базові елементи для швидкого прототипування, так і високоякісні ассети для фінальних версій своїх проектів.

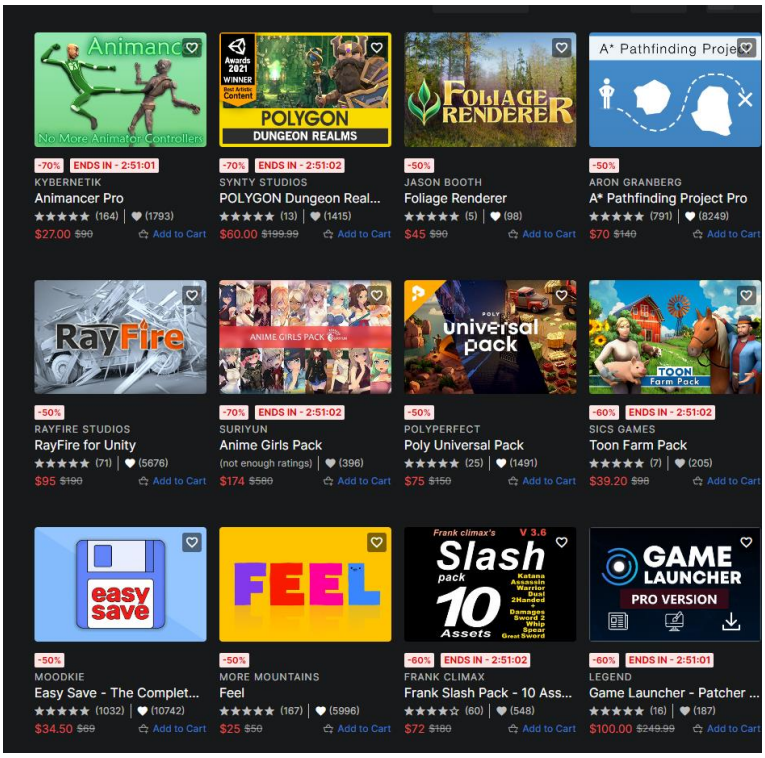


Рисунок 2.1 – Ассети у Unity Asset Store

4. **Unity Collaborate**: сервіс для спільної роботи над проектом, який дозволяє командам розробників легко працювати разом над кодом, сценами та ресурсами. Він забезпечує автоматичне керування версіями, дозволяючи зберігати та відстежувати всі зміни у проекті. Unity Collaborate робить процес співпраці простішим і ефективнішим, забезпечуючи доступ до останніх змін для всіх членів команди та знижуючи ризик конфліктів і втрати даних.

5. **Unity Analytics**: сервіс аналітики, який дозволяє відстежувати та аналізувати поведінку користувачів у грі. Він надає детальну інформацію про кількість гравців, їхні дії та взаємодію з грою, що допомагає розробникам краще розуміти свою аудиторію. Використовуючи ці дані, розробники можуть вдосконалювати ігровий процес, виправляти проблемні місця та збільшувати популярність гри. Unity Analytics підтримує різноманітні метрики та звіти, що дозволяє глибоко аналізувати всі аспекти користувацької взаємодії.

6. Unity Test Framework: цей фреймворк, який дозволяє автоматизувати тестування гри. Він забезпечує можливість створення юніт-тестів та інтеграційних тестів, що допомагає виявляти та виправляти помилки на ранніх етапах розробки. Автоматизоване тестування значно підвищує ефективність роботи команди, дозволяючи швидко перевіряти коректність функціонування різних частин гри після внесення змін. Це зменшує ризик випуску гри з критичними помилками та покращує загальну якість продукту.

Ці технології та інструменти допомагають розробникам створювати високоякісні та ефективні ігрові застосунки, забезпечуючи надійність, продуктивність та інтерактивність кінцевих продуктів [14].

2.2 Методи реалізації геймплею

Для реалізації геймплею у жанрі FPS Arena на платформі Unity потрібно розробити системи управління гравцем, поведінку ворогів та фізичну модель гри. Ось деякі методи, які можна використовувати.

2.2.1 Геймплейна система гравця

Пересування та обертання: гравець може керувати своїм персонажем за допомогою клавіатури або контролера. Для цього використовуються вбудовані функції управління Unity, такі як `Input.GetAxis`, що дозволяє легко реалізувати рух вперед, назад, вбік та обертання. Крім того, важливо враховувати плавність та швидкість руху, щоб гравець відчував повний контроль над своїм персонажем.

Стрільба та зміна зброї: гравець може взаємодіяти зі своєю зброєю та вибирати різні види зброї. Це можна реалізувати за допомогою системи введення Unity та скриптів, які керують зброєю та її ефектами. Наприклад, використання методів `Input.GetButtonDown` для виявлення натискання клавіші стрільби та відповідних скриптів для створення ефектів пострілу та відліку патронів.

Використання предметів та екіпіровка: гравець може збирати предмети, такі як аптечки або броню, та використовувати їх для покращення своїх характеристик. Це можна реалізувати через систему інтеракції та управління інвентарем. Для цього використовуються колайдери та тригери Unity, що дозволяють гравцеві взаємодіяти з предметами, а також скрипти для управління інвентарем та застосування предметів.

Рисунок 2.2 – Інвентар гравця

2.2.2 Поведінка ворогів

Штучний інтелект: вороги можуть мати програмовану поведінку, що включає рух, стрільбу, атаку та ухилення від вогню. Для цього використовуються алгоритми штучного інтелекту, такі як поведінкові дерева або алгоритм Міні-Макс.

Ці алгоритми дозволяють створювати складні моделі поведінки, які можуть адаптуватися до дій гравця, роблячи гру більш викликовою та цікавою.

Система реакції на середовище: вороги можуть реагувати на зміни у середовищі, такі як зміна рівня освітлення або наявність перешкод. Це допоможе створити більш реалістичну та цікаву гру. Реалізація такої системи можлива за допомогою датчиків та тригерів, які змінюють поведінку ворогів у залежності від умов гри.

Рисунок 2.3 – Гра, у якій вороги адаптуються під дії гравця

2.2.3 Фізична модель гри

Реалістична фізика руху та колізії: використання фізичного движка Unity дозволяє створити реалістичну модель руху та поведінки об'єктів у грі. Це включає врахування маси, сили тяжіння, тертя та інших фізичних параметрів, що впливають

на взаємодію об'єктів. Unity забезпечує зручні інструменти для налаштування фізичних властивостей, що дозволяє створювати правдоподібні рухи та колізії.

Ефекти взаємодії: для поліпшення імерсії можна використовувати ефекти взаємодії, такі як вибухи, руйнування та інші спеціальні ефекти. Це допоможе створити більш захоплюючий ігровий досвід. Для реалізації таких ефектів використовуються частинкові системи, анімації та звукові ефекти, які додають глибинності та реалістичності ігровому процесу.

2.3 Алгоритми та підходи до створення ігрових арен

При створенні ігрових арен для ігор у жанрі FPS Arena важливо враховувати динаміку гри, стратегічні можливості та комфортність гравців.

1. **Процедурне генерування.**

Застосування алгоритмів процедурного генерування дозволяє створювати різноманітні та унікальні арени кожного разу, коли гравець починає гру. Такі алгоритми можуть базуватися на шумових функціях, генетичних алгоритмах або комбінації різних підходів. Це забезпечує нескінченну варіативність ігрових просторів та підтримує інтерес гравців.

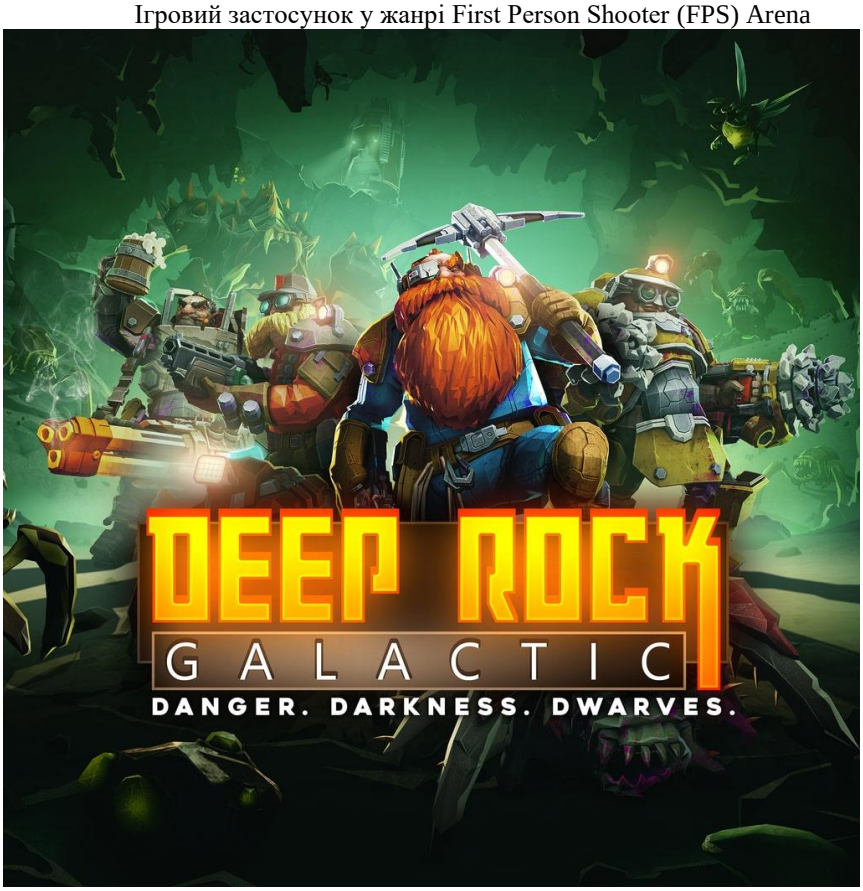


Рисунок 2.4 – Гра з процедурною генерацією кожної місії

2. **Ручне проектування.**

Ручне створення арени дозволяє розробникам контролювати кожен аспект гри і забезпечити оптимальну геймплейну ситуацію. При цьому важливо враховувати такі аспекти, як розмір, форма, розташування прихованих місць та ресурсів. Ручне проектування дозволяє досягти високого рівня деталізації та балансу в грі.

3. **Баланс геймплею.**

Важливим аспектом є збалансованість арени, щоб уникнути переваги однієї команди чи гравця над іншими. Аналіз та вдосконалення балансу можна здійснювати шляхом тестування гри та збору даних про результати матчів. Це дозволяє виявити проблемні місця та внести корективи для забезпечення чесного та рівного геймплею.

4. **Створення варіативності**.

Для забезпечення довготривалої цікавості гри можна використовувати алгоритми, які дозволяють варіювати арену під час гри або використовувати

динамічні елементи, які змінюються протягом матчу. Це може включати рухомі платформи, змінні перепони та інші елементи, що додають динаміки та непередбачуваності.

5. **Аналіз даних гравців**.

Використання аналітики гри дозволяє зрозуміти, як гравці взаємодіють з ареною та як можна покращити її дизайн. Дані про те, які області арени найчастіше використовуються, які точки зору є найбільш вигідними для стрільби та інші показники можуть допомогти вдосконалити дизайн арени. Це забезпечує підвищення комфорту та задоволення від гри для всіх гравців.

Ці підходи та алгоритми дозволяють розробникам створювати цікаві та динамічні ігрові арени у жанрі FPS Arena, які будуть цікаві для гравців та забезпечать їм захоплюючий геймплей.

Висновки до розділу 2

У цьому розділі було розглянуто основні аспекти розробки ігрових застосунків на платформі Unity, починаючи з історичної довідки про розвиток цього інструменту. Unity, будучи однією з провідних платформ для створення ігор, надає розробникам потужний набір інструментів для створення графічно складних та інтерактивних ігор.

Розглянуті були ключові компоненти, які використовуються в розробці ігор у жанрі FPS Arena. Системи управління гравцем включають в себе механіки пересування, стрільби, зміни зброї та використання предметів. Поведінка ворогів реалізується через складні алгоритми штучного інтелекту та реакцію на зміну середовища, що створює більш реалістичний та викликовий геймплей. Фізична модель гри забезпечує реалістичну взаємодію об'єктів та ефекти, які підвищують рівень занурення гравців у ігровий світ.

Також було розглянуто алгоритми та підходи до створення ігрових арен. Процедурне генерування дозволяє створювати унікальні та варіативні арени, що підтримують інтерес гравців до гри. Ручне проектування забезпечує високий рівень

деталізації та контроль над геймплейними ситуаціями. Баланс геймплею, створення варіативності та аналіз даних гравців допомагають вдосконалювати ігрові арени, роблячи їх цікавими та викликовими для гравців.

Загалом, використання Unity та його інструментів дозволяє створювати високоякісні ігрові застосунки, що задовольняють потреби як розробників, так і гравців. Отримані знання про основні аспекти розробки, поведінку гравців та ворогів, а також створення ігрових арен сприяють ефективному створенню захоплюючих ігор у жанрі FPS Arena.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІГРОВОГО ЗАСТОСУНКУ

У цьому розділі розглядається процес проектування та розробки ігрового застосунку у жанрі FPS Arena на платформі Unity. Особлива увага приділяється ключовим аспектам розробки, таким як системи управління гравцем, поведінка ворогів, фізична модель гри та проектування ігрових арен. Описуються вибрані методи та інструменти, що були використані для реалізації цих компонентів, а також аналізуються результати тестування та вдосконалення гри на основі отриманих даних. Метою цього розділу є надання детального огляду кожного етапу розробки, починаючи від концептуального проектування до остаточного тестування і налаштування гри.

3.1 Пошук та сбір ассетів і інших графічних елементів

Процес пошуку та збору ассетів і фонових зображень для гри у жанрі FPS Arena на платформі Unity складається з кількох етапів. Основні категорії ассетів включають моделі ворогів, модель гравця, ландшафт та довкілля.

На початку проекту визначаються вимоги до графічних елементів гри, такі як стиль, розмір, формат та рівень деталізації. Важливо, щоб усі ассети відповідали загальному художньому стилю гри та технічним обмеженням платформи Unity. У даному проекти був обраний примітивно-мінімалістичний стиль, який доволі часто зустрічається у проектах малих студій чи розробників-одинаків.

Рисунок 3.1 – Приклад гри з спрощеною графікою

Рисунок 3.2 – Приклад гри з реалістичною графікою

Хоча потужність Unity також дозволяє створювати доволі високополігональні текстури та моделі. Причиною такого вибору стає те, що змагатися з великими студіями у деталізації нема сенсу, тому зазвичай обирають простий, але приємний дизайн.

Unity Asset Store [8] є основним джерелом для пошуку готових ассетів. Використовуючи пошукові запити та фільтри, можна швидко знайти необхідні моделі та ресурси.

Для гри були знайдені моделі ворогів, модель гравця, а також елементи ландшафту та довкілля. Вибрані ассети повинні мати високий рейтинг та позитивні відгуки, що гарантує їх якість та сумісність з Unity.

Рисунок 3.3 – Ассети, що використовувалия у проекті

Unity Asset Store пропонує як безкоштовні, так і платні ресурси. Для даної роботи використовувалися безкоштовні ассети.

Кожен знайдений ассет проходить ретельну перевірку на відповідність вимогам проекту. Це включає тестування в демо-сценах Unity для перевірки сумісності та якості. Були наявні проблеми з правильним відображенням текстур для усіх ассетів, після невеликих змін проблему вирішили.

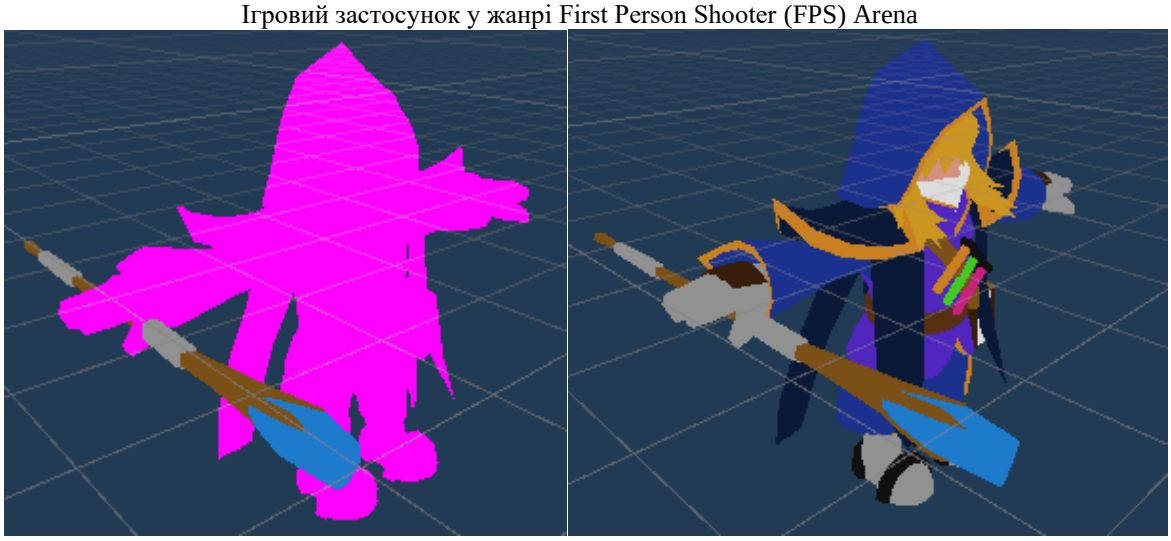


Рисунок 3.4 – Відображення текстур до та після виправлення проблеми

Фонові зображення для гри також шукалися на спеціалізованих ресурсах, таких як сайти з безкоштовними текстурами та зображеннями. Вибиралися високоякісні зображення, які доповнюють атмосферу гри.

Рисунок 3.5 – Приклади використаних зображень

3.2 Інтеграція ассетів і інших графічних елементів в ігровий застосунок

Процес інтеграції ассетів та інших графічних елементів в ігровий застосунок на платформі Unity включає кілька ключових етапів, які забезпечують належну роботу та взаємодію між різними компонентами гри.

3.2.1 Імпорт ассетів у Unity

Першим кроком є імпорт усіх необхідних моделей та текстур в проект Unity. Для цього використовуються стандартні інструменти Unity для імпорту файлів, зокрема FBX для моделей і PNG або JPEG для текстур.

Ассети розміщуються у відповідних папках проекту для зручності організації та подальшого використання [10].

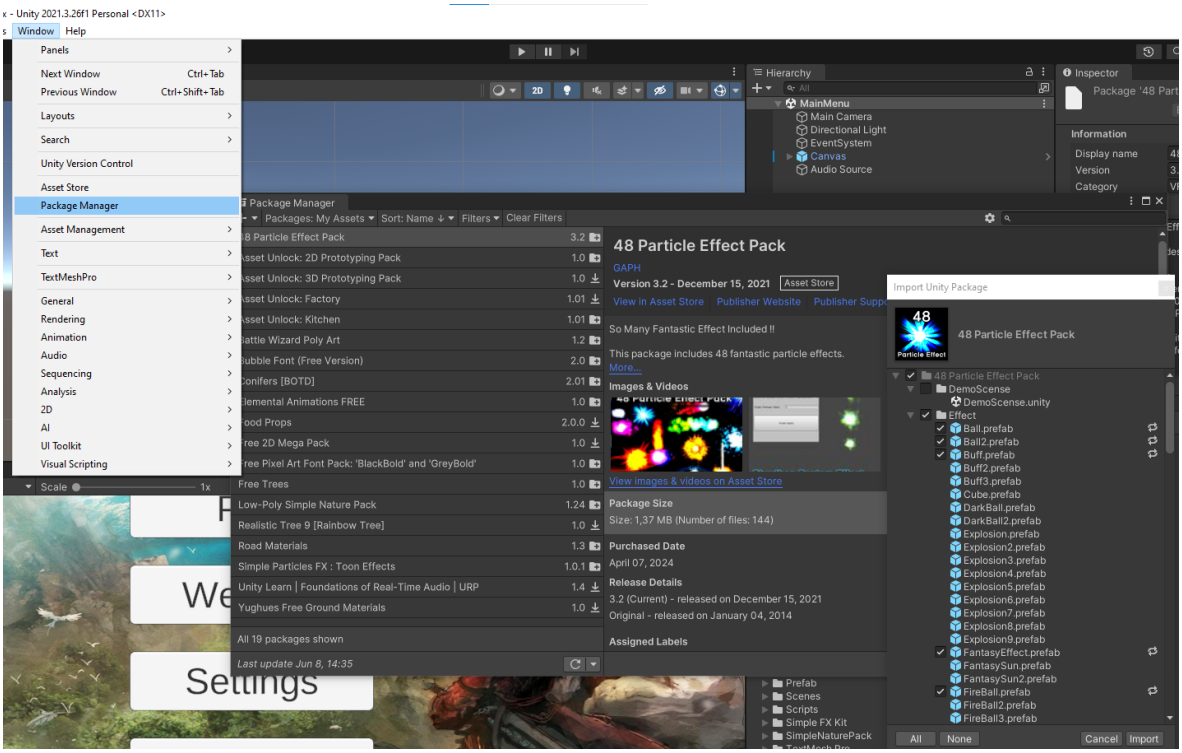


Рисунок 3.6 – Приклади імпорту ассету

3.2.2 Налаштування моделей гравця та ворогів

Створюється об'єкти гравця та ворогів, до яких додаються імпортовані моделі. Налаштовуються анімації руху, такі як ходьба, біг та стрільба. Для гравця інтегруються скрипти управління, які обробляють введення від гравця та забезпечують реалістичне переміщення. Для ворогів – скрипти, які визначають поведінку ворогів.

3.2.3 Розміщення та налаштування довкілля

Об'єкти довкілля, такі як будівлі, бар'єри, меблі та інші деталі, імпортуються та розміщуються на ландшафті відповідно до дизайну рівня. Використовуються інструменти Unity для точного позиціонування, масштабування та обертання об'єктів, щоб створити реалістичну та функціональну ігрову сцену [13].

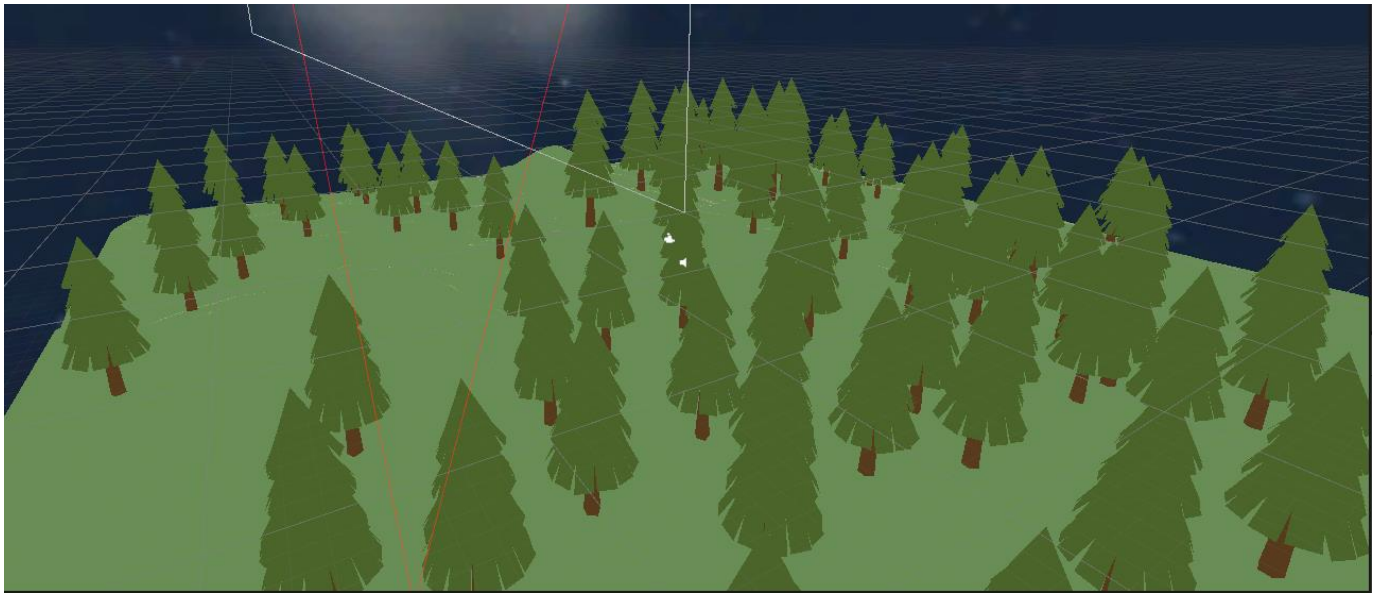


Рисунок 3.7 – Приклади створення арени

3.3 Створення скриптів

Процес створення скриптів у Unity для ігрового застосунку включає кілька ключових етапів, кожен з яких охоплює різні аспекти взаємодії гравця з грою, поведінку ворогів та загальне управління ігровим процесом. Серед необхідних скриптів – взаємодія з меню, керування персонажем, дії ворогів, їх взаємодія з гравцем та додаткові скрипти для різних ігрових механік.

3.3.1 Скрипти для взаємодії з меню

Меню головного екрану

Створюються скрипти для обробки подій меню, таких як старт гри, вихід з гри, налаштування та вибір зброї.

ButtonScript.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ButtonScript : MonoBehaviour
{
    public string sceneName = "Arena";
    public GameObject panel;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void ChangeScene(string sceneName)
    {
        PlayerInfo.SaveData();
        SceneManager.LoadScene(sceneName);
        Time.timeScale = 1;
    }

    public void Exit()
    {
        #if UNITY_EDITOR
                UnityEditor.EditorApplication.isPlaying = false;
        #endif
        Application.Quit();
    }

    public void PanelOpen()
    {
        panel.SetActive(true);
    }
    public void PanelClose()
    {
        panel.SetActive(false);
    }

    public void StaffChouse(string element)
    {
        PlayerPrefs.SetString("staffChouse", element);
    }

    public void Continue()
    {
        Time.timeScale = 1.0f;
    }

}
```

Налаштування та пауза.

Скрипти для меню налаштувань та паузи дозволяють гравцеві змінювати параметри гри, такі як звук, графічні налаштування та управління. Реалізуються функції для збереження та завантаження налаштувань з використанням PlayerPrefs або інших механізмів збереження.

Скрипти для зміни та збереження гучності музики.

VolumeSettings.cs

```
using UnityEngine;
using UnityEngine.UI;

public class VolumeSettings : MonoBehaviour
{
    [SerializeField] private AudioSource m_AudioSource;
    [SerializeField] private Slider musicSlider;
    public static float musicValue;

    private void Start()
    {
        if (PlayerPrefs.HasKey("musicVolume"))
        {
            LoadVolume();
        }
        else
        {
            SetMusicVolume();
        }
    }

    public void SetMusicVolume()
    {
        float volume = musicSlider.value;
        m_AudioSource.volume = volume;
        PlayerPrefs.SetFloat("musicVolume", volume);
    }

    private void LoadVolume()
    {
        musicSlider.value = PlayerPrefs.GetFloat("musicVolume");
        musicValue = PlayerPrefs.GetFloat("musicVolume");
        SetMusicVolume();
    }

}
```

Volume.cs

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

public class Volume : MonoBehaviour
{

    [SerializeField] private AudioSource m_AudioSource;

    // Start is called before the first frame update
```

```
    void Start()
    {
        if (PlayerPrefs.HasKey("musicVolume"))
        {
            m_AudioSource.volume = PlayerPrefs.GetFloat("musicVolume");
        }
    }
}
```

Зміна зброї.

Реалізуються методи для зміни зброї, що дозволяють гравцеві вибирати різні види зброї через меню.

WeaponControll.cs

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

public class WeaponControll : MonoBehaviour
{
    [SerializeField] private string staffElement;
    [SerializeField] private string element;
    [SerializeField] private GameObject staff;


    // Start is called before the first frame update
    void Start()
    {
        if (PlayerPrefs.HasKey("staffChouse"))
        {
            element = PlayerPrefs.GetString("staffChouse");
        }
        else
        {
            element = "frost";
        }
        LoadWeapon();
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void LoadWeapon()
    {
        if (element == staffElement)
        {
            staff.SetActive(true);
        }
        else
        {
            staff.SetActive(false);
        }
    }
}
```

3.3.2 Скрипти для керування персонажем

Пересування та обертання.

Використовуються скрипти, що обробляють введення з клавіатури або контролера для пересування та обертання персонажа. Наприклад, методи Input.GetAxis для отримання значень з осей керування. Скрипт керування гравцем винесений у Додаток А.

PlayerInfo.cs

```
using System;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using System.IO;

public class PlayerInfo : MonoBehaviour
{

    [SerializeField] GameObject character;
    public float health;
    private float maxHealth = 100;
    [SerializeField] GameObject healsBar;
    [SerializeField] TextMeshProUGUI hitpointsText;


    public static float level = 1;

    [System.Serializable]
    public class MyData
    {
        public float playerLevel = PlayerInfo.level;
    }

    public static MyData data;

    // Start is called before the first frame update
    void Start()
    {
        LoadData();
        health = maxHealth;
    }

    // Update is called once per frame
    void Update()
    {
        hitpointsText.text = "Hitpoints: " + Math.Round(health);
        healsBar.transform.localScale = new Vector3(health / 100,
healsBar.transform.localScale.y);
        if (health <= 0)
        {
            level = 1;
            SceneManager.LoadScene("MainMenu");
            SaveData();
        }
    }

    private void OnTriggerStay(Collider other)
    {
        if (other.gameObject.tag == "Enemy")
        {
```

```
            health -= 0.1f;
        }
        if (other.gameObject.tag == "Boss")
        {
            health -= 0.5f;
        }
    }

    public static void SaveData()
    {
        string json = JsonUtility.ToJson(data);
        File.WriteAllText(Application.persistentDataPath + "/data.json", json);
        Debug.Log(Application.persistentDataPath + "/data.json");
        Debug.Log("Save complite");
    }

    public void LoadData()
    {
        string path = Application.persistentDataPath + "/data.json";
        if (File.Exists(path))
        {
            string json = File.ReadAllText(path);
            data = JsonUtility.FromJson<MyData>(json);
        }
        else
        {
            Debug.LogWarning("JSON file not found!");
        }
    }

}
```

Стрільба .

Скрипти для стрільби включають створення проектилів, обробку їх траєкторії та впливу на ворогів. Використовуються методи, такі як `Instantiate` для створення нових об'єктів (наприклад, куль).

Projectile.cs

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

public class Projectile : MonoBehaviour
{
    [SerializeField] private string projectileElement;
    [SerializeField] private string element;
    [SerializeField] private GameObject projectile;
    [SerializeField] private GameObject mainProjectile;

    public float speed = 40.0f;

    private float bound = 300;

    // Start is called before the first frame update
    void Start()
    {

    }
```

```
    // Update is called once per frame
    void Update()
    {
        ProjectileType();
        transform.Translate(Vector3.forward * Time.deltaTime * speed);
        if (transform.position.z > bound || transform.position.z < -bound ||
transform.position.x > bound || transform.position.x < -bound || transform.position.y >
bound || transform.position.y < -bound)
        {
            Destroy(mainProjectile.gameObject);
        }
    }

    public void ProjectileType()
    {
        if (PlayerPrefs.HasKey("staffChouse"))
        {
            element = PlayerPrefs.GetString("staffChouse");
        }
        else
        {
            element = "frost";
        }
        LoadProjectile();
    }

    public void LoadProjectile()
    {
        if (element == projectileElement)
        {
            projectile.SetActive(true);
        }
        else
        {
            projectile.SetActive(false);
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        Destroy(mainProjectile);
    }
}
```

3.3.3 Скрипти для дій ворогів та взаємодії з ними

Створюються скрипти для управління поведінкою ворогів, переслідування, та атаки гравця. Скрипти обробляють взаємодію ворогів з гравцем, включаючи атаки та нанесення шкоди. Використовуються методи для перевірки колізій та застосування пошкоджень до гравця.

Enemy.cs

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

```
public class Enemy : MonoBehaviour
{
    [SerializeField] private int[] damage = { 3, 5, 1 };
    private GameObject player;
    public string sceneName;

    public int hitpoints = 5;
    public float speed = 3;

    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.Find("Character");
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 lookDirection = (player.transform.position -
transform.position).normalized;
        gameObject.transform.position += (lookDirection * speed * Time.deltaTime);
        if (hitpoints <= 0)
        {
            if (gameObject.tag == "Boss")
            {
                PlayerInfo.level++;
                NextLevel(sceneName);
            }
            Destroy(gameObject);
        }
    }

    public void NextLevel(string name)
    {
        SceneManager.LoadScene(name);
    }
    private void OnTriggerEnter(Collider other)
    {
        string element = PlayerPrefs.GetString("staffChouse");
        if (other.tag != "Player")
        switch (element)
        {
            case "frost":
                hitpoints = hitpoints - damage[0];
                break;
            case "earth":
                hitpoints = hitpoints - damage[1];
                break;
            case "wind":
                hitpoints = hitpoints - damage[2];
                break;
        }

    }
}
```

3.3.4 Додаткові скрипти

Скрипт для видалення снаряду.

Цей скрипт відповідає за видалення снарядів після певного часу. Це необхідно для забезпечення ефективного використання пам'яті та запобігання перенавантаження гри.

DeleteFarAway.cs

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

public class DeleteFarAway : MonoBehaviour
{

    private float bound = 500;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (transform.position.z > bound || transform.position.z < -bound ||
transform.position.x > bound || transform.position.x < -bound || transform.position.y >
bound || transform.position.y < -bound)
        {
            Destroy(gameObject);
        }
    }
}
```

Скрипт для спауну ворогів.

Цей скрипт керує створенням нових ворогів у грі. Він відповідає за спаун ворогів у певних точках, а також за динамічне збільшення складності гри.

SpawnManager.cs

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

public class SpawnManager : MonoBehaviour
{
    public GameObject enemyPrefab;
    public GameObject enemyBoss;

    private float spawnRange = 10;
    public int enemyCount;
    public int waveNumber = 1;

    // Start is called before the first frame update
    void Start()
    {
        SpawnEnamyWave(waveNumber);
    }

    // Update is called once per frame
```

```
    void Update()
    {
        enemyCount = FindObjectsOfType<Enemy>().Length;
        if (enemyCount == 0)
        {
            waveNumber++;
            SpawnEnamyWave(waveNumber);
        }
    }
    void SpawnEnamyWave(int enemiesToSpawn)
    {
        for (int i = 0; i < enemiesToSpawn * PlayerInfo.level; i++)
        {
                Instantiate(enemyPrefab, GenerateSpawnPosition(),
enemyPrefab.transform.rotation);
        }
        if (enemiesToSpawn % 5f == 0)
        {
            Instantiate(enemyBoss, GenerateSpawnPosition(),
enemyBoss.transform.rotation);
        }
    }

    private Vector3 GenerateSpawnPosition()
    {
        float spawnPosX = Random.Range(-spawnRange, spawnRange);
        float spawnPosZ = Random.Range(-spawnRange, spawnRange);

        Vector3 randomPos = new Vector3(spawnPosX, -5, spawnPosZ);
        return randomPos;
    }
}
```

Процес створення скриптів у Unity вимагає глибокого розуміння C#, знань про обробку введення користувача, управління анімаціями та фізичними властивостями об'єктів, а також навичок у обробці подій у реальному часі. Завдяки правильно написаним та оптимізованим скриптам, вдається створити функціональний, захоплюючий та стабільний ігровий застосунок.

3.4 Ітеративне проектування

Проектування ігор - це 1% натхнення
та 99% ітерацій.
– Кріс Свейн

3.4.1 Тестування сторонніми людьми

Після визначення цілей, проектування рішення та реалізації прототипу, настає етап випробування прототипу та отримання сторонньої думки про дизайн. Кожен коментар, як позитивний, так і негативний, сприяє покращенню сприйняття

гравця та вдосконаленню дизайну. Зворотний зв'язок від гравців є необхідним для удосконалення дизайну.

Наприклад, якщо гра тестувалася недостатньо, її цілі можуть бути не зовсім зрозумілими, а рівень складності може різко змінюватися. Це свідчить про те, що більшість часу в гру грали люди, які знали, як грати і долати складні ділянки, тому вони не помічали проблем, які могли б побачити звичайні тестувальники.

Таким чином, тестування та зворотний зв'язок від гравців є критично важливими для створення якісного ігрового продукту, оскільки вони дозволяють виявити недоліки та покращити ігровий процес, забезпечуючи оптимальне сприйняття гри кінцевими користувачами.

Згідно з книгою «Unity и C#. Геймдев від ідеї до реалізації.» Дж. Бонда [1], наступним колом тестувальників після команди розробки повинні бути близькі друзі. Надавши їм альфо-версію гри отримали наступний зворотній зв’язок.

Тестувальник з великим досвідом у багатьох іграх

«Перше – туторіал. Так і повинно бути що нічого не відбувається, коли я на нього натискаю? Якщо так, то навіщо взагалі потрібна ця кнопка.

Далі трохи про меню. Що damage та що speed? Вони не мають жодних значень. Там мали бути цифри, опис чи що?

І про геймплей. Благаю закріпи мишку у вікні гри, а то вона у мене відлітає на другий екран і виходить так що я можу повертатися тільки вліво (у мене екран знаходиться праворуч).»

Тестувальник з малим ігровим досвідом

«Боти з’являються дуже багато і в незручних місцях. Вони пропадають чи не з’являються. В принципі, там ще трохи небо криве, є смуга біла, а все небо темне. Дуже складно грати з wind staff, із земельним легко дуже взагалі»

Тестувальник з нормальним ігровим досвідом

«Не можу попасти по ворогу, коли стою на місці, снаряди пролітають над ним майже усі. Вороги іноді рухаються до мене спиною чи боком, не повертаються.»

Тестувальник з великим досвідом у іграх жанру FPS

«Вороги з’являються іноді всередині гравця. Часто вороги не з’являються. Не можу пройти через це рівень»

3.4.2 Виправлення знайдених проблем

Виправлення багів

Під час тестування було виявлено кілька помилок, серед яких одна критична, що суттєво заважала ігровому процесу. Проблема полягала в тому, що вороги з'являлися під ареною замість того, щоб з'являтися на ній. Через це гравець не міг їх здолати, а отже, не міг пройти рівень. Усунення цієї проблеми стало першочерговим завданням. Для вирішення цієї проблеми було внесено зміни в код файлу SpawnManager.cs, який відповідає за створення нових ворогів. Зміни полягали у коректному визначенні координат появи ворогів, що забезпечило їхнє правильне розташування на арені. Після внесення правок було проведено повторне тестування, яке підтвердило успішне вирішення проблеми: вороги тепер з'являлися на арені, що дозволяло гравцеві з ними взаємодіяти та продовжувати проходження рівня.

Інші виявлені проблеми мали переважно візуальний характер. Однією з таких проблем було неправильне положення ворогів, через що вони рухалися спиною вперед, що порушувало реалістичність ігрового процесу та знижувало загальне враження від гри. Іншою значущою проблемою були видимі стики в текстурах неба, які створювали непривабливий вигляд і порушували атмосферу гри. Для усунення цих візуальних недоліків втручання в код не знадобилося.

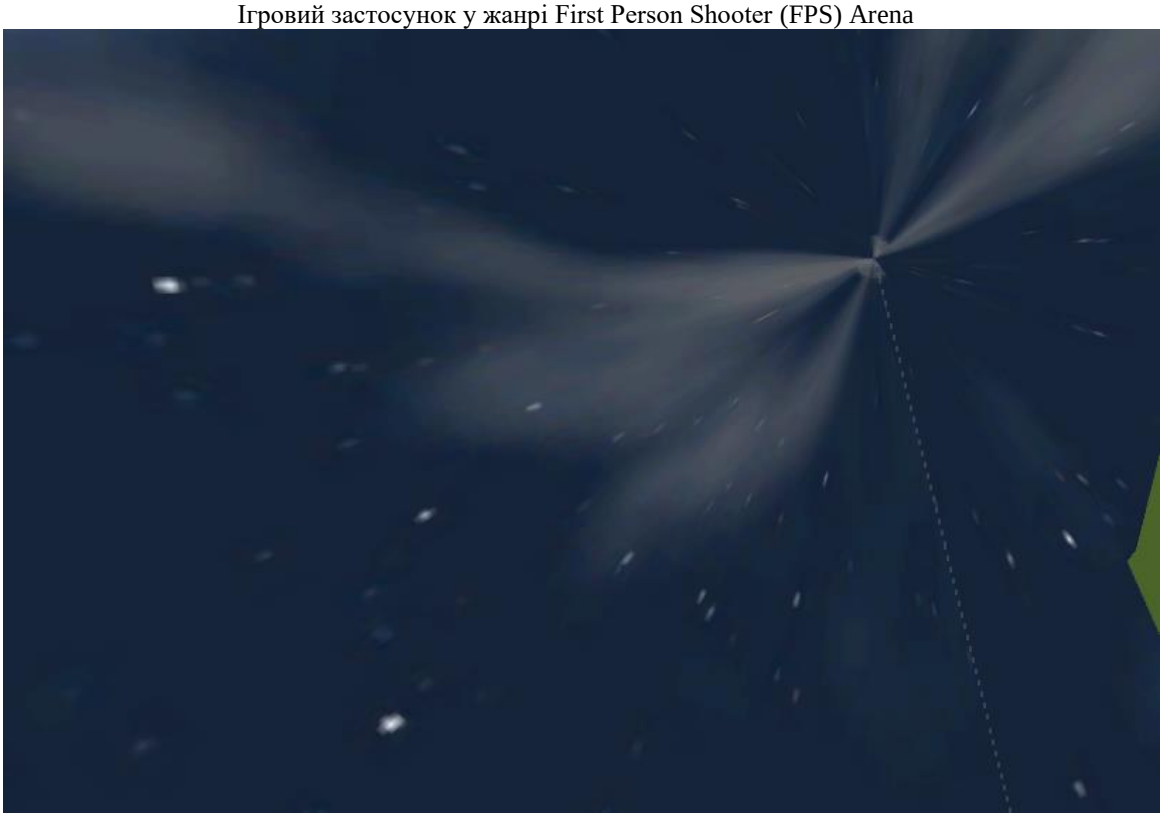


Рисунок 3.8 – Проблеми з текстурами неба

Проблему з положенням ворогів було вирішено шляхом коректування їхніх початкових орієнтацій у редакторі Unity. Це забезпечило правильний напрямок руху ворогів та покращило загальну реалістичність їхньої поведінки. Щодо стиків у текстурах неба, було використано високоякісні текстури з безшовним з'єднанням, що дозволило усунути видимі шви та покращити візуальне сприйняття ігрового середовища. Обидві ці проблеми були вирішені за допомогою налаштувань у редакторі та заміни текстур, що значно покращило загальну якість гри без необхідності змін у програмному коді.

Висновки до розділу 3

У цьому розділі детально розглянуто процес проектування та розробки ігрового застосунку у жанрі FPS Arena на платформі Unity. Проведений аналіз показав важливість ретельного планування кожного етапу розробки, починаючи від пошуку та збору ассетів до інтеграції графічних елементів та створення скриптів, які забезпечують функціональність гри.

Пошук та збір ассетів включали вибір моделей ворогів, гравця, ландшафту та довкілля. Використання якісних графічних ресурсів є ключовим для створення привабливого та реалістичного ігрового середовища. Інтеграція цих ассетів вимагала коректного налаштування їхніх параметрів у Unity для забезпечення правильного відображення та взаємодії між ними.

Процес створення скриптів був важливим етапом, що включав написання коду для взаємодії з меню, керування персонажем, дії ворогів та їхню взаємодію з гравцем. Особлива увага була приділена написанню додаткових скриптів для видалення снарядів, спауну ворогів та відображення здоров'я персонажа, що забезпечило плавний та злагоджений ігровий процес. Під час тестування були виявлені та виправлені як критичні, так і менш суттєві помилки.

Таким чином, у цьому розділі було продемонстровано комплексний підхід до проектування та розробки ігрового застосунку, включаючи всі необхідні етапи для створення якісного продукту. Ретельне планування, реалізація та тестування забезпечили успішне досягнення поставленої мети - створення функціонального та привабливого ігрового застосунку у жанрі FPS Arena.

ВИСНОВКИ

У результаті проведеної роботи була успішно реалізована розробка ігрового застосунку у жанрі FPS Arena на платформі Unity. Процес створення гри включав кілька важливих етапів: аналіз існуючих ігрових продуктів, вивчення інструментів Unity, проектування геймплею, інтеграцію графічних елементів, створення скриптів та тестування.

Перший етап, аналіз та планування, дозволив визначити ключові елементи та механіки, які потрібно було включити в розробку. Вивчення інструментів Unity і їх можливостей виявилося критично важливим для ефективної реалізації проекту. На етапі проектування та розробки було проведено пошук та збір ассетів, включаючи моделі ворогів, гравця, ландшафту та довкілля, що забезпечило візуальну привабливість гри. Інтеграція цих ассетів у Unity дозволила створити ігрове середовище, а створення скриптів для керування персонажем, дій ворогів, взаємодії з меню та додаткових функцій забезпечило функціональність гри.

Тестування та оптимізація виявили критичні та візуальні помилки, які були успішно виправлені, що значно покращило якість геймплею. Виправлення проблеми зі спауном ворогів та усунення візуальних дефектів підтвердили важливість детального тестування на кожному етапі розробки.

Метою роботи була розробка ігрового застосунку у жанрі FPS Arena, і ця мета була успішно досягнута. Проект відповідав поставленим завданням та критеріям якості, а отримані знання та навички в галузі розробки ігор на платформі Unity стали важливим внеском у подальший професійний розвиток. Загалом, проект продемонстрував комплексний підхід до розробки ігрового застосунку, включаючи ретельне планування, ефективну реалізацію та тестування. Цей досвід є цінним для майбутньої кар'єри у сфері розробки ігор, підкреслюючи необхідність уважного ставлення до кожного етапу процесу від концептуального проектування до остаточного тестування та оптимізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бонд Дж. Unity и C#. Геймдев від ідеї до реалізації. 2-ге вид., Київа, 2019. — 928 с.
2. Fullerton T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. 2008. Vol. 2, P. 496.
3. База даних Steam URL: https://steamdb.info/charts/
4. Сайт для відстеження кіберспортивних змагань Liquipedia URL: https://liquipedia.net/
5. Сайт для відстеження статистика ігор Tracker URL: https://tracker.gg/
6. Jesse Schell, Art of Game Design: A Book of Lenses. 2008. P. 489
7. Unity Learn URL: https://learn.unity.com/
8. Unity Asset Store URL: https://assetstore.unity.com/
9. Brodkin J. "How Unity3D Became a Game-Development Beast", URL: https://www.dice.com/career-advice/how-unity3d-become-a-game-development-beast
10. W. Goldstone, Unity Game Development Essentials. 2009. P. 298
11. D. Hemphill, Cybersport. 2012. P. 207
12. V. Boguslavskaya, E. Budnik, Cybersport Community: Social Structures Transformation as a Basis for Intercultural Dialogue. 2018. P. 311
13. I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. P. Baskaraca, H. Karim. 3d Modelling And Visualization Based On The Unity Game Engine – Advantages And Challenges. 2017. P. 166
14. M. Foxman. United We Stand: Platforms, Tools and Innovation With the Unity Game Engine. 2019.

ДОДАТОК А

Controller.cs

```
using System;
using System.Collections.Generic;
using UnityEngine;
#if UNITY_EDITOR
using UnityEditor;
#endif

public class Controller : MonoBehaviour
{

    public GameObject menu;
    public static Controller Instance { get; protected set; }
    public Camera MainCamera;
    public Transform CameraPosition;
    public GameObject projectilePrefab;
    public GameObject projectilePrefabPosition;
    public GameObject playerAvatar;
    private Animator anim;

    [Header("Control Settings")]
    public float MouseSensitivity = 100.0f;
    public float PlayerSpeed = 15.0f;
    public float JumpSpeed = 5.0f;

    float m_VerticalSpeed = 0.0f;
    bool m_IsPaused = false;

    float m_VerticalAngle, m_HorizontalAngle;
    public float Speed { get; private set; } = 0.0f;

    public bool LockControl { get; set; }

    public bool Grounded => m_Grounded;

    CharacterController m_CharacterController;

    bool m_Grounded;
    float m_GroundedTimer;
    float m_SpeedAtJump = 0.0f;

    void Awake()
    {
        Instance = this;
    }

    void Start()
    {
        //Cursor.lockState = CursorLockMode.Locked;
        //Cursor.visible = false;

        m_IsPaused = false;
        m_Grounded = true;

        MainCamera.transform.SetParent(CameraPosition, false);
        MainCamera.transform.localPosition = Vector3.zero;
        MainCamera.transform.localRotation = Quaternion.identity;
        m_CharacterController = GetComponent<CharacterController>();

        m_VerticalAngle = 0.0f;
```

```
        m_HorizontalAngle = transform.localEulerAngles.y;

        anim = playerAvatar.GetComponent<Animator>();
    }

    void Update()
    {
        bool wasGrounded = m_Grounded;
        bool loosedGrounding = false;

        if (!m_CharacterController.isGrounded)
        {
            if (m_Grounded)
            {
                m_GroundedTimer += Time.deltaTime;
                if (m_GroundedTimer >= 0.5f)
                {
                    loosedGrounding = true;
                    m_Grounded = false;
                }
            }
        }
        else
        {
            m_GroundedTimer = 0.0f;
            m_Grounded = true;
        }

        Speed = 0;
        Vector3 move = Vector3.zero;
        if (!LockControl)
        {
            // Jump
            if (m_Grounded && Input.GetButtonDown("Jump"))
            {
                m_VerticalSpeed = JumpSpeed;
                m_Grounded = false;
                loosedGrounding = true;
                anim.Play("JumpUp");
            }

            float actualSpeed = PlayerSpeed;

            if (loosedGrounding)
            {
                m_SpeedAtJump = actualSpeed;
            }

            // Move around with WASD
            move = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxisRaw("Vertical"));
            if (Input.GetAxisRaw("Vertical") > 0)
            {
                anim.Play("BattleWalkForward");
            }
            else if (Input.GetAxisRaw("Vertical") < 0)
            {
                anim.Play("BattleWalkBack");
            }
            else if (Input.GetAxisRaw("Horizontal") > 0)
            {
                anim.Play("BattleWalkRight");
            }
            else if (Input.GetAxisRaw("Horizontal") < 0)
```

```
            {
                anim.Play("BattleWalkLeft");
            }
            else
            {
                anim.Play("Idle01");
            }
            if (move.sqrMagnitude > 1.0f)
            {
                move.Normalize();
            }

            float usedSpeed = m_Grounded ? actualSpeed : m_SpeedAtJump;

            move = move * usedSpeed * Time.deltaTime;

            move = transform.TransformDirection(move);
            m_CharacterController.Move(move);



            // Turn player
            float turnPlayer =  Input.GetAxis("Mouse X") * MouseSensitivity;
            m_HorizontalAngle = m_HorizontalAngle + turnPlayer;

            if (m_HorizontalAngle > 360) m_HorizontalAngle -= 360.0f;
            if (m_HorizontalAngle < 0) m_HorizontalAngle += 360.0f;

            Vector3 currentAngles = transform.localEulerAngles;
            currentAngles.y = m_HorizontalAngle;
            transform.localEulerAngles = currentAngles;

            // Camera look up/down
            var turnCam = -Input.GetAxis("Mouse Y");
            turnCam = turnCam * MouseSensitivity;
            m_VerticalAngle = Mathf.Clamp(turnCam + m_VerticalAngle, -89.0f, 89.0f);
            currentAngles = CameraPosition.transform.localEulerAngles;
            currentAngles.x = m_VerticalAngle;
            CameraPosition.transform.localEulerAngles = currentAngles;

            Speed = move.magnitude / (PlayerSpeed * Time.deltaTime);
        }

        // Fall down / gravity
        m_VerticalSpeed = m_VerticalSpeed - 10.0f * Time.deltaTime;
        if (m_VerticalSpeed < -10.0f)
            m_VerticalSpeed = -10.0f; // max fall speed
        var verticalMove = new Vector3(0, m_VerticalSpeed * Time.deltaTime, 0);
        var flag = m_CharacterController.Move(verticalMove);
        if ((flag & CollisionFlags.Below) != 0)
            m_VerticalSpeed = 0;
        if (Input.GetKeyDown(KeyCode.Escape) && menu.active == false)
        {
            LockControl = true;
            menu.SetActive(true);
            Time.timeScale = 0;
        }
        else if (Input.GetKeyDown(KeyCode.Escape) && menu.active == true)
        {
            LockControl = false;
            menu.SetActive(false);
            Time.timeScale = 1;
        }
        if (menu.active == false)
```

```
        {
            LockControl = false;
        }

        if (Input.GetKeyDown(KeyCode.Mouse0))
        {
            anim.Play("Attack01");
            Instantiate(projectilePrefab, projectilePrefabPosition.transform.position,
transform.rotation);
        }
    }

    public void DisplayCursor(bool display)
    {
        m_IsPaused = display;
        Cursor.lockState = display ? CursorLockMode.None : CursorLockMode.Locked;
        Cursor.visible = display;
    }
}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

до кваліфікаційної роботи бакалавра

на тему:

ІГРОВИЙ ЗАСТОСУНОК У ЖАНРІ FIRST PERSON SHOOTER (FPS) ARENA

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401.22010127

Виконав студент 4-го курсу, групи 401
_____ О. В. Фоменко
«17» червня 2024 р.

Консультант: канд. техн. наук, доцент
_____А. О. Алексєєва
«17» червня 2024 р.

Миколаїв – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ВДТ – Візуальний дисплейний термінал

ДСанПіН – Державні Санітарні Правила та Норми

ДСТУ EN – Державні стандарти України Européen de Normalisation

ЕОМ – Електронна обчислювальна машина

ЕПТ – Електронно-променеві трубки

МОЗ – Міністерство охорони здоров'я

НПАОП – Нормативно-правові акти з охорони праці

ПК – Персональний комп’ютер

ПХБ – Поліхлоровані дифеніли

LCD – Liquid crystal display

ВСТУП

Охорона праці в умовах сучасного технологічного розвитку набуває особливої актуальності. Зокрема, при роботі з комп'ютерним обладнанням, що є невід'ємною частиною діяльності фахівців у галузі комп'ютерних технологій, питання безпеки та збереження здоров'я працівників стають надзвичайно важливими. Неправильна організація робочого місця, недотримання режиму праці та відпочинку, а також недосконале обладнання приміщень можуть призвести до серйозних негативних наслідків для здоров'я, включаючи зорову втому, м'язово-скелетні розлади та психоемоційні перевантаження.

У цьому розділі розглядаються ключові аспекти охорони праці при експлуатації комп'ютерного обладнання. Основна увага приділяється нормативно-правовим актам, які регламентують безпеку праці у сфері інформаційних технологій, а також вимогам до приміщень, обладнання та умов праці. Окремо розглядаються питання гігієни та оптимального режиму праці і відпочинку, що є критично важливими для підтримання здоров'я та продуктивності працівників.

Забезпечення належних умов праці та дотримання вимог безпеки є не лише законодавчою необхідністю, а й етичною відповідальністю роботодавця перед своїми працівниками. Таким чином, даний розділ спрямований на висвітлення важливих аспектів охорони праці, що дозволить сформувати уявлення про комплексний підхід до створення безпечного і здорового робочого середовища для спеціалістів у галузі комп'ютерних технологій.

1 НОРМАТИВНА ДОКУМЕНТАЦІЯ ЩОДО ЗАБЕЗПЕЧЕННЯ ОХОРОНИ ПРАЦІ ПІД ЧАС ВИКОРИСТАННЯ КОМП'ЮТЕРНОГО ОБЛАДНАННЯ

Експлуатація електронно-обчислювальних машин (ЕОМ) регламентується наступними нормативно-правовими актами.

Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями (затверджені наказом Мінсоцполітики від 14.02.2018 № 207), які набули чинності 18 травня 2018 року, замінивши попередні Правила охорони праці під час експлуатації електронно-обчислювальних машин № 65 від 26.03.2010 [1].

ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» (затверджені Постановою Головного державного санітарного лікаря України № 7 від 10 грудня 1998 р.) [2].

Вимоги безпеки для використання комп'ютерної техніки визначаються наступними стандартами:

- ДСТУ EN 41003:2014 «Обладнання, яке підключають до телекомунікаційних мереж та/або кабельних розподільчих систем. Додаткові вимоги щодо безпеки»;
- ДСТУ EN 60335-1:2015 «Прилади побутові та аналогічні електричні. Безпека. Частина 1. Загальні вимоги»;
- ДСТУ EN 60950-1:2015 «Обладнання інформаційних технологій. Безпека. Частина 1. Загальні вимоги»;
- ДСТУ EN 61140:2015 «Захист проти ураження електричним струмом. Загальні аспекти щодо установок та обладнання»;
- ДСТУ EN 62368-1:2017 «Обладнання аудіо-, відео-, інформаційних та комунікаційних технологій. Частина 1. Вимоги щодо безпеки».

Вимоги до влаштування комп'ютерної мережі електроживлення регулюються:

– правилами влаштування електроустановок (наприклад, Глава 1 «Загальні правила», Глава 3 «Захист і автоматика», затверджені наказом Мінпаливенерго України від 21.07.2017 № 476);

– правилами експлуатації електроустановок споживачів (затвердженими наказом Мінпаливенерго України від 25.07.2006 № 258);

– правилами безпечної експлуатації електроустановок споживачів;

Вимоги до облаштування та експлуатації комп’ютерних класів визначаються:

– правилами техніки безпеки під час навчання в кабінетах інформатики навчальних закладів загальної середньої освіти (затвердженими наказом Держпраці від 16.03.2004 № 81; НПАОП 80.0-1.12-04);

– ДСанПіН 5.5.6-167-2010 «Державні санітарні норми та правила влаштування, утримання, обладнання та організації роботи закладів, що надають послуги з комп’ютерної ігрової діяльності дітей» (затверджені наказом МОЗ від 15.12.2009 № 947).

Загальні вимоги протипожежної безпеки під час експлуатації комп'ютерної техніки регулюються:

– правилами пожежної безпеки в Україні (затвердженими наказом МВС від 30.12.2014 № 1417);

– правилами безпеки для навчальних закладів України (затвердженими наказом Міністерства освіти і науки від 15.08.2016 № 974).

2 ВИМОГИ ДО РОБОТИ З КОМП'ЮТЕРНИМ ОБЛАДНАННЯМ

Згідно з НПАОП 0.00-7.15-18, користувачам комп'ютерної техніки не потрібно присвоювати кваліфікаційну групу з електробезпеки. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.12-05), затверджене наказом Державного комітету України з нагляду за охороною праці від 26 січня 2005 р. № 15, визначає порядок допуску до роботи з комп'ютерною технікою. Відповідно до пунктів 3.1 та 3.4 цього положення, проводяться інструктажі та навчання з питань охорони праці, включаючи електробезпеку.

НПАОП 40.1-1.21-98, затверджений наказом Державного комітету України по нагляду за охороною праці від 9 січня 1998 р. № 4, регулює порядок присвоєння кваліфікаційної групи з електробезпеки на виробництві. Ці правила поширюються на працівників, які обслуговують діючі електроустановки споживачів. Технічне обслуговування включає роботи з випробування обладнання, огляд, підтягування контактних з'єднань тощо (п. 3.1 розділу III Правил технічної експлуатації електроустановок споживачів, затверджених наказом Міністерства палива та енергетики України від 25 липня 2006 р. № 258).

Електроустановки складаються з взаємопов'язаних машин, апаратів, ліній та допоміжного обладнання, призначених для виробництва, трансформації, передавання, розподілу електроенергії і перетворення її в інші види енергії (п. 1.1.3 Правил улаштування електроустановок, затверджених наказом Міністерства енергетики та вугільної промисловості України від 21 липня 2017 р. № 476).

Особи, які досягли 18 років, пройшли медичний огляд, ознайомлені з інструкцією з охорони праці та не мають протипоказань за станом здоров'я, допускаються до самостійної роботи з комп'ютером, ноутбуком, принтером, ксероксом, сканером, плазмовою панеллю, LCD-дисплеєм та іншою оргтехнікою.

Під час роботи на комп'ютері та іншій оргтехніці можуть впливати наступні небезпечні та шкідливі фактори: електрострум і випромінювання, перенапруження зору через неправильне розташування екрана. Робоче місце не повинно

знаходитися у приміщеннях без природного освітлення чи вентиляції. Кут нахилу екрана монітора має бути 10-15 градусів, відстань до екрана – 500-600 мм, а кут зору – прямим і становити 90 градусів. Для захисту від сонця слід використовувати сонцезахисні пристрої. Освітлення має бути змішаним, а в приміщенні потрібно підтримувати чистоту та порядок, регулярно провітрювати.

Про всі виявлені несправності обладнання слід негайно повідомити керівнику і припинити роботу до усунення проблеми. У випадку нещасного випадку, працівник або очевидець повинні доповісти керівнику установи і надати першу медичну допомогу. Порушення вимог охорони праці карається відповідно до чинного законодавства.

З ВИМОГИ ЩОДО РЕЖИМУ ПРАЦІ ТА ВІДПОЧИНКУ ПРИ РОБОТІ З КОМП'ЮТЕРНИМ ОБЛАДНАННЯМ

Для зменшення нервово-емоційного напруження, зорової втоми, поліпшення мозкового кровообігу та запобігання негативним наслідкам гіподинамії, варто під час деяких перерв виконувати комплекс вправ, зазначений у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007–98.

При організації роботи з використанням персональних комп'ютерів для збереження здоров'я працівників і запобігання професійним захворюванням слід передбачити регламентовані перерви для відпочинку. Режим праці та відпочинку має включати додаткові нетривалі перерви до появи ознак втоми та зниження працездатності. Основною роботою з ПК вважається така, що займає не менше 50% часу робочої зміни.

Згідно з п. 5.3 ДСанПіН 3.3.2.007-98, протягом робочого дня повинні бути передбачені:

- перерви для відпочинку та прийому їжі (обідні перерви);
- перерви для відпочинку і особистих потреб (згідно з трудовими нормами);
- додаткові перерви для окремих професій з урахуванням особливостей їх трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством і правилами внутрішнього трудового розпорядку.

Згідно з п. 5.8 ДСанПіН 3.3.2.007-98, встановлені такі режими праці та відпочинку при 8-годинній робочій зміні залежно від характеру роботи:

- для розробників програм – перерва 15 хвилин після кожної години роботи за ПК;
- для операторів ПК – перерва 15 хвилин після кожних двох годин роботи;
- для операторів комп'ютерного набору – перерва 10 хвилин після кожної години роботи за ПК.

Більшість офісних працівників підпадають під визначення розробників програм або операторів електронно-обчислювальних машин згідно з п. 5.7 ДСанПіН 3.3.2.007-98.

Якщо виробничі обставини не дозволяють дотримуватися регламентованих перерв, безперервна робота з ПК не повинна перевищувати 4 години. При 12-годинній зміні перерви повинні встановлюватися в перші 8 годин аналогічно перервам при 8-годинній зміні, а протягом останніх 4 годин – перерва 15 хвилин після кожної години роботи (п. 5.9 та п. 5.10 ДСанПіН 3.3.2.007-98).

Для зменшення негативного впливу монотонності слід чергувати операції обробки тексту та числових даних, вводу даних та редагування текстів. Для зниження нервово-емоційного напруження і зорової втоми, поліпшення мозкового кровообігу, варто під час деяких перерв виконувати комплекс вправ, зазначений у ДСанПіН 3.3.2.007-98.

У випадку хронічних скарг на зорове стомлення, попри дотримання санітарно-гігієнічних вимог, допускається індивідуальний підхід до обмеження часу роботи з ПК, зміни характеру праці та чергування з іншими видами діяльності, не пов'язаними з ПК.

4 ВИМОГИ ЩОДО ЧИСТОТИ ПОВІТРЯ НА РОБОЧИХ МІСЦЯХ

Дослідження показали, що на робочих місцях операторів ВДТ до кінця робочого дня значно зростає концентрація CO2, досягаючи 0,12-0,19% (при природному рівні в атмосферному повітрі 0,03%) .

В Німеччині було виявлено, що на робочих місцях операторів ВДТ містяться діоксини та фурани, які можуть викликати рак. Ці шкідливі сполуки входили до складу пластмас електронних плат та корпусів дисплеїв як полібромовані протипожежні речовини.

Деякі публікації зазначають, що користувачі комп'ютерів, які носять окуляри, частіше стикаються з розладами зору. Це пов'язано з тим, що для роботи за дисплеєм потрібні інші окуляри, ніж для читання, оскільки фокусна відстань при читанні складає 30 см, а для роботи за дисплеєм необхідна інша відстань .

Інші дослідження показали, що в офісах з ВДТ концентрація поліхлорованих біфенілів (ПХБ) була в кілька разів вищою, ніж в приміщеннях без ВДТ. ПХБ можуть виділятись конденсаторами та трансформаторами ВДТ. Важливо зазначити, що концентрації цих речовин рідко перевищували гранично допустимі концентрації (ГДК). Проте у більшості досліджень серед речовин, у яких було виявлено перевищення ГДК біля робочих місць з ВДТ, найчастіше називали озон, оксиди азоту та пил.

Особливу небезпеку становить підвищена концентрація озону, високотоксичного подразнювального газу, який включено до списку речовин з обмеженими максимальними значеннями концентрації на робочих місцях через його канцерогенні властивості. Для зменшення негативного впливу озону слід вимикати ВДТ, коли він не використовується. Основним заходом для запобігання впливу озону є забезпечення належної припливно-витяжної вентиляції. Для запобігання проникненню шкідливих речовин з сусідніх приміщень у приміщення з ВДТ необхідно створити надлишковий тиск [3]. Відповідно до ГОСТ 12.1.005-88, вміст озону в повітрі робочої зони не повинен перевищувати 0,1 мг/м³; оксидів азоту – 5 мг/м³; пилу – 4 мг/м³ [4][5].

5 РОЗРАХУНКИ РІВНЯ ОСВІТЛЕНОСТІ У ДОМАШНЬОМУ ОФІСІ

5.1 Опис виробничого приміщення, його геометрії, розташування в просторі та властивості інтер'єру

Приміщення домашнього офісу має прямокутну форму. Воно розташоване у десятиповерховій будівлі на четвертому поверсі. У приміщенні одне вікно, яке розташовано з виходом у двір. В оформленні інтер'єру переважають пастельні відтінки бежевого та жовтого кольору. В приміщенні один довгий стіл світло коричневого кольору, 2 стільці, один комп'ютер, принтер. Офіс працює протягом робочого тижня з 9 до 17 години:

а) довжина приміщення – 6 м;

б) ширина приміщення – 3 м;

в) висота приміщення – 2,7 м;

г) ширина вікна – 1(0,5) м;

д) висота вікна – 1,3 м;

е) кількість вікон – 1 шт.;

ж) висота верхнього краю відносно умовної робочої поверхні – 2 м;

з) відстань розрахункової точки (робочої поверхні) до зовнішньої стіни – 0,5 м.

Геометрія розрахункового приміщення, розташування еталонної робочої поверхні і протилежної будівлі наведено на *рис. 5.1 – 5.3*

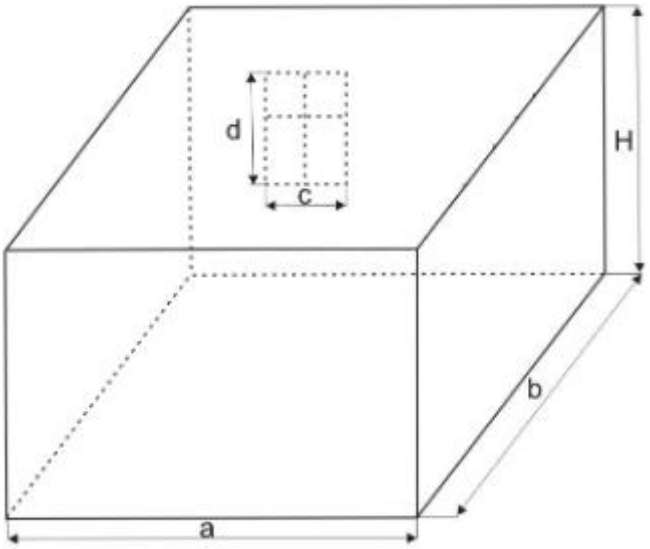


Рисунок 5.1 – Геометрія розрахункового приміщення

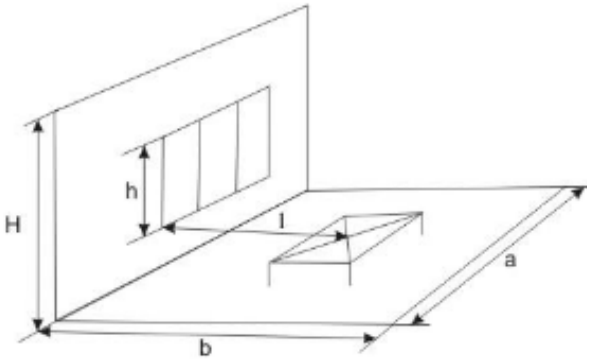


Рисунок 5.2 – Розташування робочої поверхні у виробничому приміщенні

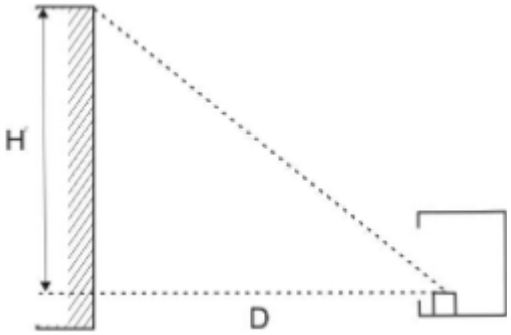


Рисунок 5.3 – Розташування протилежної будівлі

Чисельні значення розмірів зображені на рис. 5.1-5.3 наведені далі:

a – 6м;

b – 3 м;

c – 1 м;

d – 1,3 м;

h – 2 м;

l – 0,5 м;

H – 2,7 м;

H’ – 15 м;

D – 30 м.

5.2 Перевірочний розрахунок природного освітлення для виробничого приміщення.

Основним завдання при проектуванні природного освітлення є вибір типу та визначення розміщення сумарної площі світлових отворів (вікон), при яких у приміщеннях забезпечується необхідний світловий режим. Для функціонуючого приміщення доцільно виконати перевірочний розрахунок з метою визначення відповідного існуючого рівня освітленості (або площі світлових отворів) вимогам нормативних документів.

Розрахунок природного освітлення виконується в такій послідовності.

5.2.1.Значення нормованого коефіцієнта природного освітлення для III поясу світлового клімату, $e_н^{III}$ $e_н^{III}$, %. Визначається відповідно до СНіП – II – 4 – 79 в залежності від розряду зорової роботи, який залежить від найменшого розміру e_{min} e_{min} об’єкта розпізнаванням зорових робіт малої точності, при найменшому розміщенні об’єкта розпізнавання 1-5 мм. При боковому освітленні (V розряд зорової роботи),

$e_н^{III}$ $e_н^{III}$ = 1,0%

5.2.2. Коефіцієнт світлового клімату визначається відповідно для Миколаївської області, що належить до IV поясу світлового клімату:

m = 0,9

5.2.3. Коефіцієнт сонячності клімату. Визначається відповідно для світлових отворів в зовнішніх стінах будівлі, розташованих у IV поясі світлового клімату та зорієнтованих за азимутом в діапазоні 46…135 градусів:

c = 0,75

5.2.4. Нормативне значення коефіцієнту природного освітлення (КПО) до умов Миколаєва, , %

$e_н\ e_н = e_н^{III}\ e_н^{III} \times m \times c = 1 \times 0{,}9 \times 0{,}75 = 0{,}68\%$

5.2.5. Коефіцієнт запасу, що використовується при розрахунках природного освітлення. Згідно з рекомендаціями:

$К_з\ К_з = 1{,}3 \ldots 1{,}5$

Прийнято $К_з\ К_з = 1{,}4$

5.2.6. Відношення довжини приміщення (a) до його ширини (b):

$\frac{a}{b} = \frac{6}{3} = 2$

5.2.7. Відношення ширини приміщення (b) до висоти верхнього краю вікна відносно умов робочої поверхні (h) :

$\frac{b}{h} = \frac{3}{2} = 1{,}5$

5.2.8. Світлова характеристика вікна, $η_в\ η_в$:

$η_в\ η_в = f \times (\frac{a}{b}, \frac{b}{h})$, при $\frac{a}{b} = 2$ та $\frac{b}{h} = 1{,}5$

$η_в\ η_в = 8$

5.2.9. Коефіцієнт світлопропускання, τ_1. Для подвійного склопакету метало пластикових вікон:

$\tau_1 = 0{,}86$

5.2.10. Коефіцієнт, що враховує витрати світла у віконній рамі, τ_2. Для рами метало пластикових вікон:

$\tau_2 = 0{,}76$

5.2.11. Коефіцієнт, що враховує втрати світла в несучих конструкціях при боковому освітленні, τ_3.

$\tau_3 = 1$

5.2.12. Коефіцієнт, що враховує втрати світла у сонцезахисних пристроях. Для регульованих жалюзів, τ_4.

$\tau_4 = 1$

5.2.13. Загальний коефіцієнт світло пропускання, $\tau_{заг}$:

$\tau_{заг} = \tau_1 \times \tau_2 \times \tau_3 \times \tau_4 = 0{,}86 \times 0{,}76 \times 1 \times 1 = 0{,}65$

5.2.14. Коефіцієнт відбиття внутрішніх поверхонь приміщення: стелі – $\rho_{\text{стелі}}$, стін - $\rho_{\text{стін}}$, підлоги - $\rho_{\text{підлоги}}$.

- для стелі білого напівматового кольору, = 80%

- для стін світло-рожевого кольору, = 65%

- для підлоги, вкритої лінолеумом оливково-зеленого кольору, = 20%

5.2.15. Площі внутрішніх поверхонь виробничого приміщення:

$S_{\text{стелі}}$ = a×b = 6×3 = 18

$S_{\text{стін}}$ = 2×(a+b)×H = 2×(6+3)×2,7 = 48,6

$S_{\text{підлоги}}$ = a×b = 6×3 = 18

5.2.16. На основі даних розрахувати середнє значення коефіцієнта відбиття внутрішніх поверхонь приміщення:

$$\rho_{\text{сер.}} = \frac{(\rho_{\text{стелі}} \times S_{\text{стелі}}) + (\rho_{\text{стін}} \times S_{\text{стін}}) + (\rho_{\text{підлоги}} \times S_{\text{підлоги}})}{(S_{\text{стелі}} + S_{\text{стін}} + S_{\text{підлоги}}) \times 100} = \frac{((80*18)+(65*48{,}6)+(20*18))}{(18+48{,}6+18)*100} =$$

$\frac{4959}{8460}$ = = 0,586.

5.2.17. Геометричне співвідношення, що характеризує приміщення :

$\frac{a}{b}$ = 2

$\frac{b}{h}$ = 1,5

$\frac{l}{b} = \frac{0.5}{3}$ = 0,167

5.2.18. Коефіцієнт, що враховує підсилення коефіцієнту природного освітлення за рахунок відбиття світлового потоку приміщення, r_1:

$$r_1 = f\left(\rho_{\text{сер.}} \times \frac{a}{b} \times \frac{b}{h} \times \frac{l}{b}\right)$$

При $\rho_{\text{сер.}}$ = 0,586, $\frac{a}{b}$ = 2, $\frac{b}{h}$ = 1,5, $\frac{l}{b}$ = 0,167

r_1 = 2

5.2.19. Відношення відстані між протилежними будівлями, D до висоти карнизу протилежної будівлі відносно підвіконня, H':

$\frac{D}{H`} = \frac{30}{15} = 2$

5.2.20. Коефіцієнт, що враховує вплив протилежної будівлі на освітлення у приміщенні при $\frac{D}{H`} = 2$

$$К_{буд.} = 1{,}1$$

5.2.21. Знайти площу вікон, що потрібна для забезпечення необхідного рівня природного освітлення,

$$S_{в.} = \frac{e_н \times К_{з.} \times \eta_{в.} \times S_{підлоги} \times К_{буд.}}{t_{заг.} \times r_1 \times 100} = \frac{0{,}68*2*8*18*1{,}1}{0{,}65*2*100} = 1{,}657 \text{ м}^2$$

У роботі проведено розрахунки природного освітлення для домашнього офісу. Для визначення відповідностей існуючого рівня освітленості розрахунковому значенню, нам необхідно розрахувати дійсну площу вікон в приміщенні:

$$S_{в.дійсна} = n_в \times c \times d = 1 \times 1 \times 1{,}3 = 1{,}3 \text{ м}^2$$

Таким чином, порівнявши розрахунки розрахункового значення та дійсного значення, можемо зробити висновки, що у виробничому приміщенні недостатній рівень природного освітлення. Для створення сприятливих умов зорової роботи, які б виключали швидку втомлюваність очей, виникнення професійних захворювань, нещасних випадків і сприяли підвищенню продуктивності праці та якості продукції, можливі наступні зміни:

1) зміна кольору підлоги на більш світлий;
2) невелике збільшення розміру вікна;
3) зміна положення робочого місця.

ВИСНОВОК

У даній роботі було розглянуто питання охорони праці при роботі з комп’ютерним обладнанням, що є важливим аспектом забезпечення безпеки та здоров'я працівників у сучасних умовах. Аналіз нормативно-правових актів дозволив визначити основні вимоги до організації робочих місць, умов праці, а також регламенту праці та відпочинку.

Особливу увагу було приділено вимогам до приміщень з комп’ютерним обладнанням, які включають забезпечення належного освітлення, вентиляції та дотримання гігієнічних норм. Розглянуто вимоги до роботи з комп’ютерним обладнанням, зокрема щодо електробезпеки та проведення навчання з охорони праці.

Встановлено, що для підтримання високого рівня працездатності та зниження втоми необхідно дотримуватися режиму праці та відпочинку, включаючи регулярні перерви. Запропоновано рекомендації щодо виконання фізичних вправ під час перерв для зменшення негативних наслідків гіподинамії.

Аналіз досліджень показав, що забруднення повітря на робочих місцях з використанням екранних пристроїв може мати значний вплив на здоров'я працівників. Зокрема, підвищена концентрація CO_2, діоксинів, фуранів, ПХБ та озону потребує особливої уваги і застосування відповідних заходів для забезпечення безпеки.

На основі проведеного аналізу можна зробити висновок, що дотримання всіх вимог і рекомендацій з охорони праці при роботі з комп’ютерним обладнанням є ключовим фактором у забезпеченні здорових та безпечних умов праці. Це дозволить не лише запобігти професійним захворюванням, але й підвищити загальну ефективність роботи працівників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 10.05.2022).

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 10.05.2022).

3. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98 URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 10.05.2022).

4. ГОСТ 12.1.005-88.ССБП URL: http://docs.cntd.ru/document/1200003608 (дата звернення: 10.05.2022).

5. СН 4088-86. Санітарні норми мікроклімату виробничих приміщень URL: http://docs.cntd.ru/document/901710059 (дата звернення: 10.05.2022).