МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

3D-гра у жанрі аркади з інтерактивним контентом на базі UNITY

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.21810806

Студент _____ В.С. Вальковський

підпис

«__» _____ 2022 р.

Керівник канд. фіз-мат. наук, доцент _____ А.І. Воробйова

підпис

«__» _____ 2022 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедри
_____ Є. О. Давиденко
« __ » _____ 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук
Вальковському В'ячеславу Сергійовичу .
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи:

3D-гра у жанрі аркади з інтерактивним контентом на базі UNITY

Затверджена наказом по ЧНУ ім. П.Могили від « 01 » грудня 2021 р. № 314

2. Строк представлення кваліфікаційної роботи: « __ » _____ 202__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є 3D-гра у жанрі аркади з інтерактивним контентом на базі UNITY

4. Перелік питань, що підлягають розробці:

- аналіз предметної області та існуючих аналогів;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення;
- аналіз результатів розробки.

5. Перелік графічних матеріалів:

презентація.

6. Завдання до спеціальної частини

Дослідження питань охорони праці, які безпосередньо пов’язані з діяльністю розробника програмного забезпечення.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А.О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи *канд. фіз-мат. наук, доцент Воробйова Алла Іванівна*
(посада, прізвище, ім’я, по батькові)

(підпис)

Завдання прийнято до виконання

Вальковський В’ячеслав Сергійович
(прізвище, ім’я, по батькові студента)

(підпис)

Дата видачі завдання « ___ » _____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання кваліфікаційної роботи

Тема: «3D-гра у жанрі аркади з інтерактивним контентом на базі UNITY»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	03.10.2021	04.10.2021	виконано
2.	Огляд літератури за темою роботи	06.10.2021	08.10.2021	виконано
3.	Складання календарного плану КРБ	09.10.2021	10.10.2021	виконано
4.	Аналіз сучасних засобів реалізації процесу дистанційного навчання	15.10.2021	15.10.2021	виконано
5.	Розробка вимог до системи	03.11.2021	04.11.2021	виконано
6.	Аналіз ринку наявних технологій і рішень для вирішення завдань	16.11.2021	22.02.2022	виконано
7.	Проектування гри на базі UNITY	14.03.2022	30.05.2022	виконано
8.	Реалізація, тестування та налагодження застосунку	30.05.2022	10.06.2022	виконано
9.	Розробка спеціальної частини з охорони праці	10.06.2022	28.05.2022	виконано
10.	Відгук керівника КРБ	08.05.2022	20.05.2022	виконано
11.	Оформлення КРБ та презентації	21.05.2022	21.05.2022	виконано
12.	Попередній захист	15.06.2022	15.06.2022	виконано
13.	Рецензування	20.06.2022	20.06.2022	виконано
14.	Завершення оформлення КРБ та презентації	22.06.2022	22.06.2022	виконано
15.	Захист кваліфікаційної роботи	30.06.2022	30.06.2022	виконано

Розробив студент *Вальковський В'ячеслав Сергійович* _____
(прізвище, ім'я, по батькові студента) (*підпис*)

«__»_____2022 р.

Керівник роботи *канд. фіз-мат. наук, доцент Воробйова Алла Іванівна* _____
(посада, прізвище, ім'я, по батькові) (*підпис*)

«__»_____2022 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

« 3D-гра у жанрі аркади з інтерактивним контентом на базі Unity »

Студент 408 гр.: Вальковський В’ячеслав Сергійович

Керівник: канд. фіз-мат наук, доцент Воробйова Алла Іванівна

Мета: є поширення популярності розробки ігрових застосунків у жанрі «аркада» на базі Unity

Пояснювальна записка бакалаврської дипломної роботи складається з вступу, трьох розділів, висновків та додатків.

У вступі визначається актуальність теми, що приймається за мету та невеликий огляд поставленої задачі, предмет дослідження та об’єкт дослідження.

У першому розділі описується аналітична частина, тобто огляд існуючих аналогів. Визначається основна особливість ігор, завдяки чому було сформовано загальне розуміння предметної області.

У другому розділі описується процес розробки та вибір мови програмування, середовище розробки, ігровий двигун. Розробка UML-діаграм та опис інтерфейсів.

У третьому розділі демонструється проведена робота з кодування та тестування, крім того описується розробка ігрового меню та налаштувань для зручного використання.

У висновках проводиться аналіз роботи та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 51 сторінку, вона містить 4 розділи, 17 ілюстрацій, 3 таблиці, 15 джерел в переліку посилань.

Ключові слова: Unity, аркада, гонки.

ABSTRACT

of the Bachelor’s Thesis

« 3D game in the arcade genre with interactive content based on Unity»

Student 408 gr.: Vakovskyi Viacheslav

Supervisor: Associate Professor Vorobyova Alla

This work is devoted research on the development of a game application in the genre of "arcade" based on Unity.

Purpose: development of a game application in the genre of "arcade" on the Unity platform.

The explanatory note of bachelor's thesis consists of admission, three sections, conclusions and annexes.

The introduction determines the relevance of the topic taken as a goal and a small overview of the task, the subject of research and the object of study.

The first section describes the analytical part, ie an overview of existing car simulators. The main feature of the games is determined, thanks to which a general understanding of the subject area was formed.

The second section describes the development process and the choice of programming language, development environment, game engine. Development of UML diagrams and description of interfaces.

The third section demonstrates the work done on coding and testing, in addition, the development of a game menu and settings for easy use is described.

The conclusions analyze the work and the results obtained.

The bachelor's qualification work is set out on 51 pages, it contains 4 sections, 17 illustrations, 3 tables, 15 sources in the list of references.

Keywords: Unity, arcade, racing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНІ ПОЗНАКИ

ОС	–	операційна система
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ECS	–	Entity Component System
HUD	–	Heads-Up Display
CLR	–	Common Language Runtipe
UTF	–	Unity Test Framework
UI	–	User interface
UML	–	Unified Modeling Language
Геймплей	–	ігровий процес
Аркада	–	жанр відеоігор
Pool	–	особливі списки об'єктів
Action	–	дія
Components	–	компоненти
C++	–	мова програмування
C#	–	мова програмування
Entity	–	об’єкт гри, істота
Image texture	–	тестурне зображення
Racing	–	жанр відеоігор

ВСТУП

Актуальність

Індустрія комп’ютерних ігор виникла в середині 1970-х як рух ентузіастів, і за кілька десятиліть вона виросла з невеликого ринку до величезної галузі з річним прибутком у 9,5 мільярдів доларів США в 2017 році. Є як великі гравці, так і малий бізнес на ринку. , а також незалежні розробники та спільноти (наприклад, Indiegogo, Kickstarter тощо). З розвитком комп’ютерів у розробників ігор з’являється більше можливостей і свободи для творчості. Ігри пройшли довгий шлях від найпростіших двоколірних 2D ігор до сучасних 3D ігор з таким рівнем графіки, що людина, яка грає в гру, забуває, що вона в грі, і сприймає свій віртуальний світ як реальний. Також розробляються ігри з шоломами віртуальної реальності, і незабаром на ринку з’являться ігри, які дозволять гравцям повністю зануритися у віртуальний світ.

Центральним програмним компонентом комп’ютерних ігор є ігровий движок. Він забезпечує основну технологію, спрощує розробку та часто дозволяє грі запускатися на кількох платформах, наприклад на настільних ігрових консолях, таких як Linux, Mac OS X і Microsoft Windows. двигун, звук, система сценаріїв, анімація, штучний інтелект, мережевий код, імпотенція управління пам'яттю. Часто процес розробки може заощадити вам витрати на повторне використання одного ігрового движка для створення багатьох різних ігор.

Аркадна гра - гра з примітивним геймплеєм. Комп’ютерну гру називають «аркадною грою», коли вона передається безпосередньо з ігрового автомата або подібного до ігор. Наприклад, аркадні ігри включають такі жанри, як «fighting», частина жанру «racing».

Більшість аркадних ігор поширені на ігрових консолях та ігрових автоматах.Аркадні ігри включають у себе просте, інтуітивно зрозуміле управління ігровими об'єктами.

Користувачеві немає потреби налаштовувати та вивчати управління ігри, що дозволяє практично відразу приступити до ігрового процесу.

Об’єктом роботи є процес розробки ігрового застосунку у жанрі «аркада» на базі Unity.

Предметом роботи є технологія розробки ігрового застосуноку у жанрі аркади на базі Unity

Метою є поширення популярності розробки ігрових застосунків у жанрі «аркада» на базі Unity.

Таким чином, актуальне завдання розробки ігрової програми у жанрі аркада.

Завдання:

- виконати аналіз вимог та створити зовнішні специфікації;
- виконати проектування ігрового застосунку;
- реалізувати ігровий додаток;
- протестувати розроблений ігровий застосунок.

1 АНАЛІЗ ВИМОГ

1.1 Огляд предметної області

Тема комп'ютерних ігор часто опиняється в епіцентрі дискусій. Сьогодні гра – це вид мистецтва, який має достатньо галузей класифікацій.

Для реалізації своїх ідей і концепцій у відеоіграх розробник використовує різні інструменти: мови програмування, знання математики, фізики, англійської мови, системи контролю версій і ігровий движок - основу кожної гри. Зараз розберемо тему про ігрові движки.

Тому ігровий движок є одночасно інструментом для створення ігор і базовим програмним забезпеченням. Це кілька підсистем, які повинні працювати разом, щоб гра працювала. Ці підсистеми включають рендеринг, анімацію, фізику, звук, сценарії, штучний інтелект, мережевий код тощо. І одним з найпопулярніших движків є Unity - кросплатформний продукт від Unity Technologies, який дозволяє створювати програми на більш ніж 25 різних платформах. Це персональні комп'ютери (Windows, MacOS, Linux), мобільні пристрої (Android, iOS), консолі (PlayStation, Xbox, Switch тощо), VR (віртуальна реальність) та інші пристрої. Неважливо, що ви виберете – двигун задовольнить ваші потреби.

Процес створення ігор досить складний, тому часто дуже важливо отримати поради від досвідчених розробників, які добре знайомі з різними нюансами індустрії. Таку підтримку може надати спільнота Unity, оскільки з цим движком вже працюють понад два мільйони професіоналів, а кількість розробників, зацікавлених у створенні ігор, зростає з кожним днем. На офіційному веб-сайті Unity є форум спільноти, і ви можете використовувати інструмент відстеження помилок.

UnityAsset Store — ще одна функція для розробки ігор. Це сховище ресурсів, яке дозволяє не зациклюватися при створенні моделей персонажів, будівель, текстур. Комерційний майданчик також дозволяє дизайнерам, музикантам і розробникам отримувати додатковий дохід.

Основною мовою розробки на Unity є C #. Це досить легко навчитися і використовувати. Це дозволить створити власну гру за короткий час.

Ну і ще один незаперечний плюс Unity - відносно гнучкий движок, здатний створювати 2D і 3D ігри. А найприємніше в цьому движку – достатня кількість навчальних матеріалів для всіх, хто хоче поринути в ігрову індустрію.

Unity по праву можна назвати щедрим ігровим движком. Не тільки тому, що він випустив багато ігор різних жанрів для багатьох платформ, але, перш за все, для своєї безкоштовної базової версії. Звичайно, безкоштовна версія має ряд обмежень, але вона також надає всі необхідні функції для створення ігор. Unity значно зменшує грошові та трудові витрати розробників.

Движок відносно простий у використанні для початківців розробників, але він відповідає сучасним технологіям. Саме ця якість зробила Unity дуже популярною. Але навіть у великих студіях його часто використовують для створення ігор AAA. Прикладів багато.

Навіть у високопрофільованій грі Pokemon GO основна механіка гри була зроблена на Unity.

Гра була настільки популярна, що її встановили понад 100 мільйонів людей, а загальний дохід творців перевищив 440 мільйонів доларів. Unity - один з найпопулярніших двигунів, на якому розробляються понад п'ятдесят відсотків мобільних ігор. Хоча існує думка, що Unity підходить виключно для створення низькобюджетних ігор, але це абсолютно не правильно.

1.2 Огляд аналогів

Важливим етапом для розробника є проведення аналізу застосунків-аналогів. Це допомагає надихатись ідеїю, зважати на недоліки, та підвищити якість гри. Для конкурентоспроможності застосунок повинен мати не лише подібний функционал ігор-аналогів, а також власні, індивідуальні можливості для охоплення більш широкої публіки.

Retrowave

Щодо геймплею Retrowave – цілком звичайна аркадна гонка, проте за атмосферою її досить важко перевершити. Гра відправить вас у футуристичну версію 80-х років, де вам належить насолоджуватися нескінченними гонками по неоновій трасі у супроводі синтвейву/ретровейву, що викликає незрозуміле почуття ностальгії.

Таблиця 1.1 – Опис «Retrowave»

Назва	Retrowave
Архітектура	Multi-tier application
Виробник	RewindApp
Мова реалізації	C#

Кінець таблиці 1.1

Функції	1. Здатність обирати авто 2. Здатність оновлення авто 3. Здатність ставити рекорд на ділянці
Переваги	1. Не менше 10 машин. 2. Більше 5 різних Всесвітів. 3. 4 різних режими гри. 4. Унікальна ретро-футуристична атмосфера. 5. Більше 45 різних музик синтезаторів. 6. Придбати систему оновлення продуктивності автомобіля. 7. Система придбання нових кольорів та коліс. 8. Захоплюючий геймплей.
Недоліки	Відсутне геймплейне різноманіття
Веб-сайт	https://store.steampowered.com/app/1239690/Retrowave/

Slipstream

Це гоночна гра, натхненна візуальними образами, музикою, іграми та автомобілями кінця 80-х і початку 90-х. Він побудований на спеціальному ігровому движку з автентичним ретро-відчуттям та унікальною графікою. Саундтрек, заснований на впливах синти-попу та джаз-ф'южн, задає тон гонці в різноманітних екзотичних місцях з усього світу, включаючи міста, пустелі, ліси, гори та пляжі. Механіка дрейфу та ковзання додає глибину ігровому процесу водіння, а результат – складний і захоплюючий досвід.

Таблиця 1.2 – Опис «Slipstream»

Назва	Slipstream
Архітектура	Multi-tier application
Виробник	ansdor
Мова реалізації	Python, java
Функції	1. Здатність обирати авто 2. Здатність оновлення авто 3. Здатність ставити рекорд на ділянці
Переваги	1. Автентичний псевдо-3D-ігровий движок з 2D-графікою, як у дні слави аркадних гонщиків. 2. 20 різноманітних треків у різних екзотичних місцях по всьому світу і за його межами. 3. Локальний мультиплеер до 4 гравців. 4. Оригінальний саундтрек із 9 ексклюзивними піснями + ви можете додати власну музику.
Недоліки	Відсутне геймплейне різноманіття
Веб-сайт	https://store.steampowered.com/app/732810/Slipstream/

1.3 Специфікация вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням застосунку є застосування в повсякденному житті у сфері розваг та дозвілля.

Погодження, що ухвалені в програмній документації

Створення загального ПЗ та його злагодженої роботи буде використовувати допоміжні ассети та бібліотеки Unity.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 22.06.2022 р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Застосунок націлений на використання користувачем для проведення дозвілу та відпочинку.

Характеристики користувачів

Основні характеристики користувачів: наявність ПК

Загальна структура і склад системи

Основні частини програмного забезпечення: застосунок.

Загальні обмеження

Обмеженнь немає.

ФУНКЦІЇ СИСТЕМИ

Функція ставлення персональних рекордів

Опис функції

Функція отримання досягнення допомогає гравцю отримувати більше задоволення від гри.

Вхідна і вихідна інформація

Вхідна інформація – відсутня

Функціональні вимоги

Функціональні вимоги - відсутні

Опис функції

Функція привласнення гравцем іменем авто.

Функціональні вимоги

Збереження імені для гравця.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В даному ПЗ основним джерелом вхідної інформації є користувач.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги відсутні.

Вимоги до способів організації, збереження та ведення інформації

Усі данні зберігаются на жорсткому диску.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

- Вимоги до технічного забезпечення: ОС: Windows XP / 7
- Процесор: Intel Dual-Core 2.4 GHz
- Оперативна пам’ять: 3 GB ОП
- Відеокарта: NVIDIA GeForce GTX 275, GeForce GT 520, GeForce 8800GT
- DirectX: версії 9.0

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура програмної системи складається з таких компонентів: клієнтської частини.

Системне програмне забезпечення

Гра розроблена на платформі Unity з використанням мови програмування C#.

Мережне програмне забезпечення

Для створення програмного забезпечання було використано ОС Windows 10.

Мова і технологія розробки ПЗ

Розробка гри відбувалась на платформі Unity з використанням мови програмування C#.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс має задовольняти усі вимоги дизайну, він є доволі зручним у використанні .

Апаратний інтерфейс

Апаратним інтерфейсом є ПК користувача, з операційною системою Windows 10.

Програмний інтерфейс

У ході розробки було використано дві категоріх Unity Scripting API : UnityEditor (Animations, Events тощо) та UnityEngine (Analytics, Audio тощо).

Комунікаційний протокол

Відсутній.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Гра є доступною для будь-якого користувача, за умови наявності у користувача апаратного інтерфейсу.

Супроводжуваність

Гра не потребує супроводжуваності.

Переносимість

Програмне забезпечення може працювати на ОС Windows (10 версія та вище).

Продуктивність

Продуктивність роботи ПЗ залежить від характеристик ПК.

Надійність

Відсутня.

Безпека

Відсутня.

ІНШІ ВИМОГИ

Усі вимоги сформовано та описано вище, доповнення не вимагається.

Висновок до розділу 1

Під час складання розділу, були засвоєнні знання та навички стосовно розробки гри на базі Unity та написання специфікації вимог до програмного забезпечення. Таким чином, у першому розділі кваліфікаційної роботи бакалавра проаналізовано предметну область застосунку, завдяки чому визначено актуальність роботи, в основі якої лежить теза, що комп'ютерні ігри мають вплив на розвиток, а також поліпшення функціонування мозку, через що мають змогу покращувати та закріплювати здібності щодо просторового мислення, крім того й візуального уявлення стосовно тривимірних об'єктів.

В розділі, окрім опису гри, визначено специфікацію вимог до ПЗ, що розробляється. В данному підрозділі продемонстровано повний опис поведінки гри, включно з множиною функціональних та нефункціональних вимог.

2 ОГЛЯД ЕТАПІВ СТВОРЕННЯ ЗАСТОСУНКУ

2.1 Основні вимоги до технічного забезпечення

Движок Unity побудований на власному C / C ++ всередині, але має оболонку C #, з якою ви можете взаємодіяти. Отже, ви повинні бути знайомі з деякими ключовими поняттями написання скриптів на C #.

Відеоігри на базі Unity підтримують OpenGL і DirectX. Крім того, цей двигун використовує простий інтерфейс Drag & Drop. складається з різних вікон, що дозволяє вносити корективи в гру в редакторі.

Unity — хороший вибір для створення проектів середнього розміру як для ПК, так і для мобільних пристроїв. Велика кількість готових активів, включаючи скрипти, дуже допомагає. Велика спільнота також допоможе вам вирішити проблему, якщо вона виникає.

Unity надає API для доступу до засобів введення та налаштувань Android. Unity дозволяє викликати функції, написані на C/C++ безпосередньо зі скриптів (Java-функції можуть бути викликані непрямим способом). [3]

Щоб підвищити продуктивність у версії Unity для Android, динамічне написання JavaScript завжди вимкнено (якщо до кожного сценарію автоматично не застосовується #pragma strict). Це важливо пам’ятати, якщо ви завантажуєте проект на Android зі старих платформ і якщо ви стикаєтеся з помилками компіляції – проблему слід шукати в першу чергу письмово. Такі помилки зазвичай легко обробляються, якщо переконатися, що типи всіх змінних явно вказано або вказано під час ініціалізації. Хоча Unity Android підтримує текстури DXT / PVRTC / ATC, Unity розширює текстури RGB (A) під час запуску, якщо ці методи стиснення підтримуються на конкретному пристрої. Це може серйозно вплинути на продуктивність графічного процесора, тому рекомендується використовувати формат ETC. Це де-факто стандарт для Android і повинен підтримуватися всіма сучасними пристроями. З іншого боку, ETC не підтримує

альфа-канал, і іноді 16-бітові текстури RGBA будуть найкращим вибором з точки зору якості, розміру та швидкості рендеринга там, де потрібен альфа-канал. [3]

Однією з переваг Unity перед іншими інструментами є те, що він дозволяє створювати ігри з шоломами віртуальної реальності. Віртуальна реальність – високорозвинена форма комп'ютерного моделювання, яка дозволяє користувачеві зануритися у штучний світ і за допомогою спеціальних сенсорних пристроїв діяти безпосередньо в ньому, пов'язувати свої рухи з аудіовізуальними ефектами. При цьому зорове, слухове, тактильне та рухове сприйняття користувача замінюються їх комп'ютерним моделюванням. Характерними ознаками віртуальної реальності є: моделювання в реальному часі; імітація середовища з високим ступенем реалізму; здатність впливати на навколишнє середовище та мати зворотний зв'язок. [4]

Компонентно-орієнтований підхід до програмування, який підтримує гнучкість і повторне використання деяких частин програмного забезпечення;

- мова претендує на реальну об'єктну орієнтацію, тобто суть претендує на роль об'єкта;
- система письма однорідна;
- мова більше орієнтована на безпеку коду в порівнянні з C ++;
- передове програмування, орієнтоване на події.

C # також має деякі недоліки:

- продуктивність відносно низька;
- мало нових концептуальних ідей;
- складний синтаксис.

Microsoft Visual Studio — це інтегроване середовище розробки (IDE), створене Microsoft. Цей продукт використовує платформи розробки програмного забезпечення, такі як Windows API, Windows Presentation Foundation, Windows Forms і Microsoft Silverlight.

Visual Studio Tools — це безкоштовне розширення для Visual Studio. Це робить Microsoft дуже потужним інструментом кросплатформних технологій в Unity, оскільки движок не має редактора коду. Це розширення використовує налагодження коду Microsoft Visual Studio та продуктивність для створення та налагодження сценаріїв C# за допомогою потужних функцій Visual Studio.

Крім того, Visual Studio Tools має дуже глибоку інтеграцію з редактором Unity, тому розробник витрачає менше кліків для виконання простих завдань. Це забезпечує підвищення продуктивності під час розробки гри. Також можна виділити наступні сильні сторони продукту:

- Використання технології автозаповнення IntelliSense, яка записує назву функції при введенні перших літер:
- потужні можливості рефакторингу (зміна внутрішньої структури коду для полегшення розуміння, але без зміни зовнішньої поведінки самої системи);
- швидке налаштування для Unity;
- широкий вибір налаштувань робочого середовища.
- використовування технології автозаповнення IntelliSense, яка записує ім'я функції під час введення його початкових букв:

Розробка ігор для смартфонів, комп’ютерів і консолей дуже різна. Хоча б тому, що вони мають різні технічні характеристики, пристрої введення/виводу та методи розповсюдження продукції. Створити одну гру на кількох платформах відразу не вийде.

Якщо розробка гри планується для персонального комп’ютера, слід враховувати наступні моменти:

- введення з клавіатури та миші - те, до чого користувачі звикли з дитинства;

- вивести зображення на екран монітора. Моделей моніторів багато, вони відрізняються частотою зміни зображення, розміром, роздільною здатністю – це необхідно враховувати при створенні інтерфейсу гри;
- великий обсяг оперативної та відеопам'яті; ви можете дозволити собі детальні текстури, плавну анімацію, високополігональні об’єкти світу та великі карти;
- Величезна кількість відеокарт, процесорів та інших компонентів, що робить тестування ігор трудомістким процесом;
- можливість розповсюдження по-старому способу (диски) або через інтернет-магазини (найпопулярнішим на даний момент є steam).

2.2 Вибір двигуна для створення гри

Один з найпопулярніших двигунів сьогодні. Unity — це кросплатформний інструмент розробки додатків і ігор, що працює на операційних системах. Програми на базі Unity працюють на Windows, OS X, Android, Apple iOS, Linux, а також на Wii, PlayStation 3 і XBox 360.

Інтернет-додатки можна створювати як за допомогою спеціального модуля для браузера Unity, так і шляхом експериментальної реалізації всередині модуля

Програми, створені за допомогою Unity, підтримують DirectX і OpenGL. Технічні характеристики:

- Скрипти на C# та JavaScript;
- Ігровий движок, повністю інтегрований із середовищем розробки, що дозволяє тестувати гру безпосередньо в редакторі;
- Робота з ресурсами можлива за допомогою звичайного Drag & Drop;
- Система успадкування об'єктів;
- Підтримка імпорту великої кількості форматів файлів;
- Вбудований ландшафтний редактор;
- Вбудована підтримка мережі;

- Є рішення для спільної розробки - Asset Server.

Також можна використовувати зручний для користувача спосіб контролю версій. На-приклад, SVN або Source Gear.

Редактор Unity має простий інтерфейс Drag & Drop, який можна легко налаштувати. Він складається з різних вікон, тому ви можете налагодити гру прямо в редакторі.

Проект Unity розділений на сцени - окремі файли, які містять ігрові світи зі своїм набором об'єктів, сценаріїв і налаштувань. Об’єкти сцени містять набори компонентів, з якими взаємодіють сценарії.

У них також є ім'я, тег і шар, на якому він повинен з'явитися. Так, кожен об’єкт у сцені повинен мати компонент Transform - він зберігає координати розташування, ворота та розміру на всіх трьох осях. У версії движка 2019 з'явився новий тип ігрових об'єктів - сутності. За замовчуванням вони не мають компонентів і відображаються у спеціальному вікні налагоджувача сутності.

Істоти є частиною нової системи компонентів сутностей (ECS). Метою системи є підвищення продуктивності та простоти розробки.

Якщо раніше всі змінні та функції для певного об’єкта зберігалися в одному скрипті MonoBehavior, то тепер вони розділені на два окремих класи - компонент, який зберігає змінні певної істоти, і систему, яка керує необхідними істотами. Тому для переміщення використовується один сценарій, який застосовується до всіх об’єктів, які потрібно перемістити. Раніше кожен об’єкт рухався самостійно, що створювало велике навантаження на систему.

Unity також підтримує фізику за допомогою фізичних механізмів Unity Physics і Box2D. Редактор має систему успадкування об’єктів, дочірні об’єкти будуть повторювати всі зміни положення, повороту та масштабу батьківського об’єкта. Скрипти в редакторі додаються до об’єктів як окремі компоненти.

Імпортуючи текстуру в движок, ви можете створити карту відображення, але ви не можете прикріпити текстуру безпосередньо до моделі - ви створите

матеріал, до якого буде призначено шейдер, і тільки потім матеріал буде приєднано до моделі. Модель.

Редактор Unity підтримує написання та редагування шейдерів. Він також містить компонент для створення анімації, який також можна заздалегідь створити в 3D-редакторі та імпортувати разом із моделлю, а потім розділити на файли.

Система сценаріїв ігрового движка створена на моно-вільному проекті з відкритим кодом для реалізації .NET Framework. Програмісти можуть використовувати UnityScript (спеціальна мова сценаріїв, подібна до JavaScript і ECMAScript) і C #.

Таблиця 2.1 – порівняння ігрових двигунів

	Unity3D		Unreal Engine	
	Переваги	Недоліки	Переваги	Недоліки
1	Є безкоштовна версія	Безкоштовна лише для персонального використання	Масштабований	C++ досить складний для вивчення та використання
2	Є готові assets	Логотип Unity при використанні безкоштовної версії	Є готові assets	Проєкт, як і сам Unreal Engine, займає багато місця
3	Працює на багатьох платформах		Якісні навчальні матеріали	Важко розібратися в інтерфейсі
4	Велика спільнота		Велика спільнота	Вимагає значного досвіду та знань
5	Підтримує як 2D, так і 3D		Підтримує як 2D, так і 3D	

2.3 Вибір мови програмування для створення гри

Для створення сучасних комп'ютерних ігор використовуються різні мови програмування. Їх вибір залежить від обраного движка і платформи, на якій буде

запускатися гра. Наприклад, Swift підходить для розробки ігор на платформах iOS і MacOS, Java Script підходить для розробки ігор у браузері, а C++ і C# зазвичай використовуються для комп'ютерних ігор. Як мови програмування в Unity 2019 доступні такі мови: C# і JavaScript.

C# — це об'єктно-орієнтована мова програмування із безпечною системою запису для платформи .NET. Розроблено Андерсом Галесбергом, Скоттом Вілтамутом і Пітером Голдом під егідою Microsoft Research [8].

Синтаксис C# близький до C++ і Java. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класу, атрибути, події, властивості, винятки, коментарі у форматі XML.

Він бере на озброєння багато своїх попередників - C++, C#, виходячи з практики їх використання, виключаються деякі моделі, які виявилися проблематичними при розробці програмних систем, наприклад, множинне наслідування класів (на відміну від C++).

C# дуже близький до мови програмування Java. Мова Java була створена Sun Microsystems, коли глобальний розвиток Інтернету приніс проблему розподілених обчислень. Базуючись на популярній мові C++, Java виключила потенційно небезпечні речі (наприклад, покажчики без транскордонного контролю).

Для розподілених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп’ютера чи іншого інтелектуального пристрою. Java здобула значну популярність і також була ліцензована Microsoft. Але з часом Sun почала звинувачувати Microsoft у тому, що її клон Java сумісний лише з платформою Windows, що суперечить концепції незалежних від машин часу виконання та порушує ліцензійну угоду. Microsoft відмовилася виконувати прохання Sun, тому з'ясування стосунків стало судовим

позовом. Суд постановив, що позиція Sun була справедливою, і зобов'язав Microsoft відмовитися від неліцензованого використання Java.

У цій ситуації Microsoft вирішила використати свою вагу на ринку, щоб створити власну аналогію Java, мови, на якій корпорація стане повноправним власником. Ця новостворена мова називається C#. Він успадкував концепцію віртуальної машини (середовище .NET), байт-код і більшу безпеку вихідного коду програми від Java, на додаток до врахування досвіду використання Java-додатків.

Інновація C# — це можливість легше, ніж попередники, взаємодіяти з кодом, написаним іншими мовами, що важливо при створенні великих проектів.Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

C # є флагманом Microsoft, оскільки він повністю використовує нові можливості .NET. Інші мови програмування, хоча й підтримуються, вважаються успадкованими недоліками у використанні .NET.

C# був розроблений як мова програмування прикладного рівня для CLR, і тому залежить насамперед від можливостей самого CLR. Це особливо актуально для системи C#. Наявність чи відсутність певних виразних ознак мови диктується тим, чи можна перекласти певну мовну особливість у відповідні конструкції CLR [6].

2.4 Вибір фреймворку для створення гри

Фреймворк — це набір готових до використання програмних рішень, включаючи дизайн, логіку та базову функціональність системи чи підсистеми. Відповідно, програмна структура може також включати інструменти, деякі бібліотеки коду, скрипти і взагалі все, що полегшує створення та комбінування різних компонентів великого програмного забезпечення або швидке створення

готового і не обов'язково громіздкого програмного продукту. Дизайн кінцевого продукту зазвичай базується на одному API.

Unity Test Framework (UTF) є одним з найважливіших інструментів контролю якості; це система автоматизованого тестування як в редакторі, так і на підтримуваних платформах. І ця система доступна всім користувачам Unity.

Новий API Test Runner розширює гнучкість та можливості UTF, забезпечуючи максимальне покриття тестів. Ця система доступна в менеджері пакетів Unity, тому ми можемо швидше виправляти й оновлювати помилки. Крім того, це означає локальну доступність вихідного коду UTF для вас. Вивчіть вихідний код, покращуйте його під час налагодження та змінюйте його на свій смак.

Якщо ви новачок у Unity Test Framework (UTF), прочитайте вхідну документацію. Коротше кажучи, UTF дозволяє користувачам Unity тестувати свій код у режимі редагування та відтворення, а також на цільових платформах, таких як Standalone, Android, iOS та інші [7].

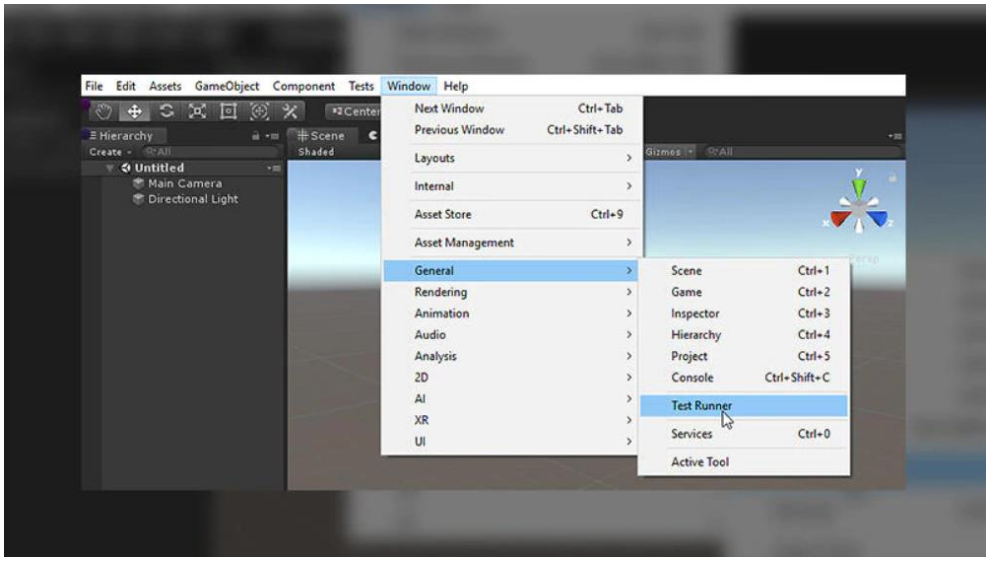

Рисунок 2.1 – Unity Test Framework

Також у двигуна є підтримка скриптів та API. Скрипти допомагають вам писати команди, які виконуватимуться грою весь час або після якихось дій гравця. API допомагає спростити написання скриптів.

2.5 Вибір програми для розробки дизайну та моделей застосунку

Для розробки гри в Unity використовуються 3D-моделі. Коли виникає необхідність створення власних моделей, було проаналізовано можливості Unity, щоб оцінити можливість створення 3D-моделей саме в ньому. Але оскільки можливості середовища Unity досить обмежені, тому було розглянуто сторонні програми, призначені для моделювання об'єктів у 3D. В результаті аналізу з великої кількості різних програм вибір зупинено на редакторі Blender. Він дозволяє легко моделювати, а також редагувати 3D-моделі, які призначені для використання в іграх, відео, рекламі.

Blender – програмний продукт, який використовується для створення 3D-візуалізації: від статичних об'єктів до редагування відео, розроблений студією NeoGeo. На сьогоднішній день перевага даного редактора в тому, що він має відкритий вихідний код і добре підходить не лише приватним особам, а й невеликим студіям, які створюють 3D-відеоролики, маючи лише Blender. Цей продукт підтримує такі інструменти, як моделювання, анімація, композинг, текстуринг та інші типи симуляцій.

Також Blender надає великі можливості для моделювання, однією з переваг цього програмного продукту є те, що в ньому зібрані можливості для моделювання різними методами: це інструменти для скульптингу, можливості роботи зі сплайнами, кривими Безье, і полігональне моделювання. Кожен із методів має свої переваги та недоліки. В результаті можна отримати модель, яку можемо в цьому ж редакторі і тектсурувати, а також поставити анімацію. Отриману 3D-модель є як використовувати для створення статичної картинки або відео в самому Blender, так і експортувати цю модель в інші програмні редактори, де вже використовувати їх для кінцевих цілей. Наприклад, у Unity для використання моделі комп'ютерної гри.

Unity підтримує різні формати для імпорту 3D-моделей, включаючи такі приватні формати програм для моделювання, як .blend (розширення Blender), .mb (розширення Maya), .c4d (розширення Cinema 4D). Крім того, із Blender можна експортувати не лише окремі об'єкти, а й цілі сцени. Для роботи у сфері 3D-моделювання в залежності від цілей може бути достатньо одного програмного продукту, але дуже часто можливостей однієї програми недостатньо для реалізації задуманого проекту. Unity - не тільки професійний потужний інструмент для створення ігор та інших додатків з використанням 3D-моделювання, але й непогано оснащена платформа для створення простих 3D-об'єктів, які бракують візуалізації будь-яких сцен. Проте складніші об'єкти найзручніше створювати у спеціальному редакторі Blender, після чого отриманий об'єкт імпортувати в Unity, хоч і процес імпортування має свої нюанси.

При експорті слід враховувати такі особливості:

– основні складності експорту об'єктів з Blender до Unity пов'язані з різницею орієнтації осей у цих програмах;

– для об'єктів, які в Unity будуть використовуватися як динамічні, важливим є правильне визначення точки Origin;

– матеріал та текстури для об'єкта не можна зберегти єдиним файлом з об'єктом;

– імпорт у Unity системи частинок, створених у Blender, неможливий безпосередньо;

– Вибір формату для експорту з Blender до Unity залежить від переваг особи, що створює модель.

Ще одна програма, на яку впав вибір – Cinema 4D. Вона є найпростішою у використанні програмою. Принаймні, так заявляють на сайті розробників. Всі основні переваги даного продукту є: Зручність у використанні, інтуїтивний інтерфейс, стабільність, інтегрована довідка, доступність програми в різних варіантах залежно від конкретних завдань.

Cinema 4D – універсальна програма для 3D-моделювання, редагування об'єктів та створення ефектів. Рендери можуть використовуватися як «рідні», так і вбудовані безпосередньо в програму за допомогою плагінів.

Основні особливості програми:

- Комбінація різних типів моделювання (полігонів, сплайнів, модифікаторів)
- Ви можете розмістити сторонні візуалізатори в основній версії програми
- Великий вибір шейдерів і файлів зображень, анімації вже створені як шаблон
- Глибоке освітлення допомагає створити реалістичне світло, відображення тощо для максимальної достовірності

Висновок до розділу 2

Під час складання розділу, були засвоєнні знання та навички стосовно розробки гри на базі Unity та проаналізовані аспекти до програмного забезпечення. Таким чином, у другому розділі кваліфікаційної роботи бакалавра проаналізовано основні вимоги Unity що до зручності користування для звичайного користувача. Також у розділі було вибрано зручний ігровий двигун, із можливих, вибрано мову програмування а також фреймворку, який є вбудованим в Unity. Також згадано про програми для розробки дизайну та моделей застосунку.

3 АРХІТЕКТУРА ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Уніфікована мова моделювання (UML) також використовується для проектування системи в програмній інженерії, яка є візуальною мовою для визначення та документування системи. Вимоги до сценаріїв, які виражають, як користувачі використовують систему, відображаються за допомогою UML. Системні обмеження також відображаються за допомогою UML.

Таким чином, багато дослідників, які працюють як інженери програмного забезпечення, публікують статті про те, як системні діаграми UML використовуються для розробки та роблять внесок у практику програмної інженерії. У нашому дослідженні SLR використовується, щоб зрозуміти, які діаграми UML популярні, чому вони використовуються та які програми є найпопулярнішими. Це стандартизований графічний формат відображення для візуалізації, специфікації, проектування та документації (програмного забезпечення) систем. Він пропонує набір стандартизованих типів діаграм, які дозволяють легко організувати складні дані, процеси та системи чітким та інтуїтивно зрозумілим способом.

UML не є процедурою чи процесом; скоріше, він надає «словник» символів, кожен із яких має певне значення. Він пропонує типи діаграм для об’єктно-орієнтованого аналізу, проектування та програмування, забезпечуючи плавний перехід від системних вимог до остаточної реалізації. Також відображається структура та поведінка системи, що дає чіткі вказівки щодо оптимізації рішення.

3.1 Діаграма станів програмного забезпечення

Діаграма стану - діаграма, яка визначає зміну стану об'єкта з часом, одна з діаграм моделювання поведінки в UML.

Елементами діаграми є:

- Круг, що вказує початковий стан;
- Круг з маленькою крапкою посередині, що вказує на кінцевий стан;
- Стрілка, що вказує на перехід.

Назва події, що викликає перехід, відзначається над/під стрілкою.

Діаграма станів описує процес зміни станів тільки одного класу або одного екземпляра конкретного класу, тобто моделює всі можливі зміни стану конкретного об’єкта. Таким чином, зміна стану об’єкта може бути викликана зовнішніми впливами з боку інших об’єктів або ззовні. Це опис реакції об’єкта на такі зовнішні впливи та використання діаграм станів.

Основна мета цієї діаграми — описати можливі послідовності станів і переходів, які разом характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма станів представляє динамічну поведінку сутностей на основі специфікації їх реакції на сприйняття певних подій. Системи, які реагують на зовнішні дії інших систем або користувачів, іноді називають реактивними. Якщо такі дії ініціюються у випадкові випадкові моменти години, то говорять про асинхронну поведінку моделі.

На рисунку 3.1 приведена загальна діаграма станів програмного забезпечення

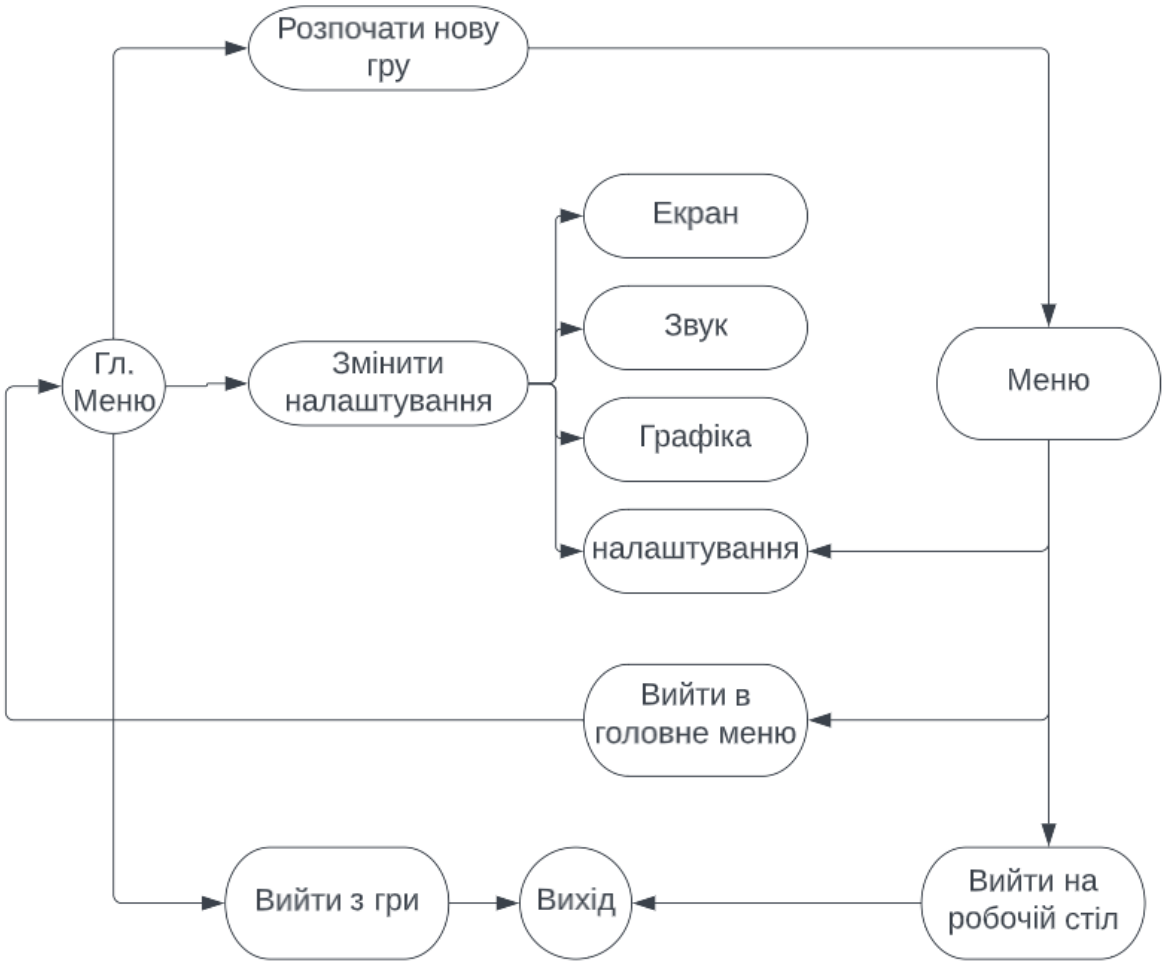

Рисунок 3.1 - Загальна діаграма станів

Розпочати нову гру- можливість почати гру з самого початку, ігноруючи будь-які файли збереження прогресу.

Змінити налаштування - можливість змін конфігурацій гри.

Екран - змінення співвідношення пікселів екрану.

Звук - налаштування гучності навколишнього середовища у грі.

Графіка - змінення конфігурації гри та якість текстур, заради зменшення впливу на систему.

Налаштування - змінення складності гри.

Вийти в головне меню - повернення користувача на головне вікно.

Прочитати відомості про автора - містить інформацію про розробників.

Меню - сторінка, де можна продовжити гру з файлу збереження та вийти на робочий стіл.

Вийти на робочий стіл - завершення сеансу у грі, та повернення користувача на головний екран Windows.

3.2 Діаграма компонентів програмного забезпечення

Діаграма компонентів, на відміну раніше розглянутих діаграм, описує особливості фізичного уявлення системи. Діаграма компонентів дозволяє визначити архітектуру системи, що розробляється, встановивши залежності між програмними компонентами, у ролі яких може виступати вихідний, бінарний та виконуваний код.

У багатьох середовищах розробки модуль чи компонент відповідає файлу. Стрілки, що з'єднують модулі, показують відносини взаємозалежності, аналогічні тим, що мають місце при компіляції вихідних текстів програм. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси та залежності між ними.

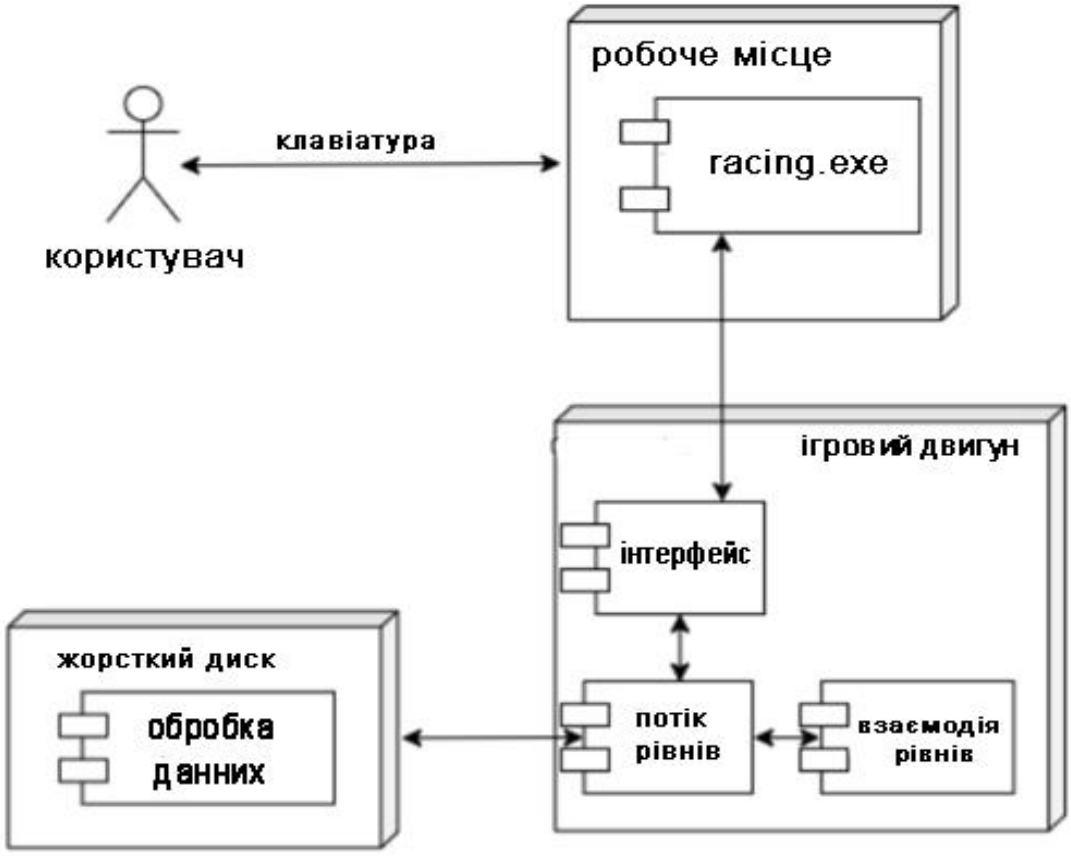

Рисунок 3.2 - Діаграма компонентів

3.3 Діаграма класів програмного забезпечення

Діаграма класів - діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язку між ними.

Це набір статичних, декларативних елементів моделі. Діаграми класів можна використовувати при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - опис існуючих і використовуваних систем. Інформація про діаграму класів відображається безпосередньо у вихідному коді програми - у більшості існуючих інструментів моделювання UML можливе генерування коду для певної мови програмування (зазвичай Java або C ++).

Таким чином, діаграма класів є кінцевим результатом і відправною точкою процесу розробки.

Клас є основним будівельним елементом літака. Ця концепція також присутня в мовах програмування GO, тобто існує відповідність між класами UML і класами програм, що є основою для автоматичного генерування програмного коду або реінжинірингу.

Кожен клас має назву, атрибути та операції. Клас на діаграмі відображається у вигляді прямокутника, розділеного на 3 області. Верхня містить назву класу, середня - опис атрибутів (властивостей), нижня - назви сервісних операцій, наданих об'єктами цього класу.

Діаграма класів є центральним елементом проектування об'єктно-орієнтованої системи. Позначення класів використовується на різних етапах проектування і створюється з різним рівнем деталізації. UML використовується не тільки для проектування, а й для документації та створення ескізів проекту.

У будь-якому об'єктно-орієнтованому процесі проектування діаграма класів є результатом, тому що вона є моделлю, найбільш близькою до реалізації, тобто коду.

Існують інструменти, що здатні перетворити діаграму класів в код такий процес називається кодогенерацією та підтримується безліччю IDE та засобів проектування.

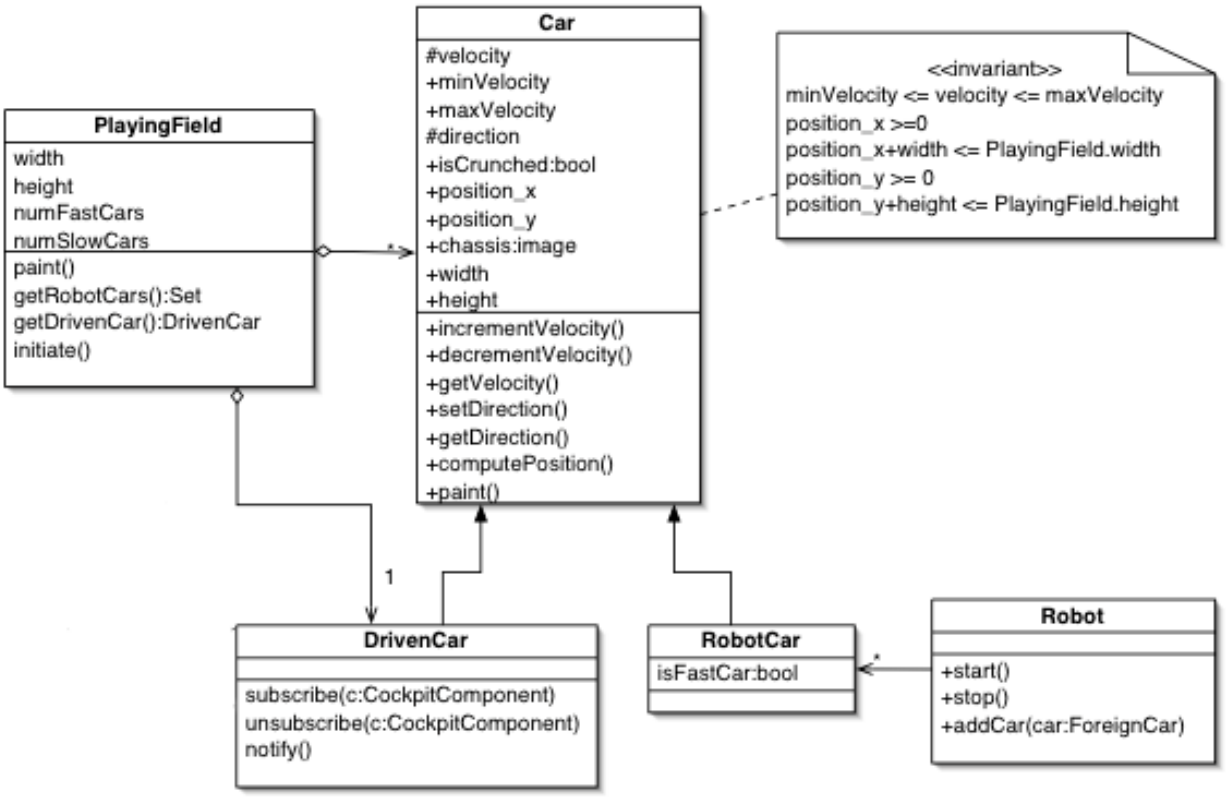

Рисунок 3.3 - Діаграма класів

Клас Car - відповідає за налаштування віртуального автомобіля гравця.

Клас PlayingField - відповідає за налаштування ігрового поля гравця.

Клас MouseSteering - відповідає за налаштування чутливості керування.

Клас DrivenCar - відповідає за налаштування взаємодії автомобіля з гравцем.

Клас RobotCar - відповідає за налаштування машин на ігровому полі яким керує штучний інтелект.

Клас Robot - відповідає за налаштування керування автомобіля роботом.

3.4 Діаграма послідовності програмного забезпечення

Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас.

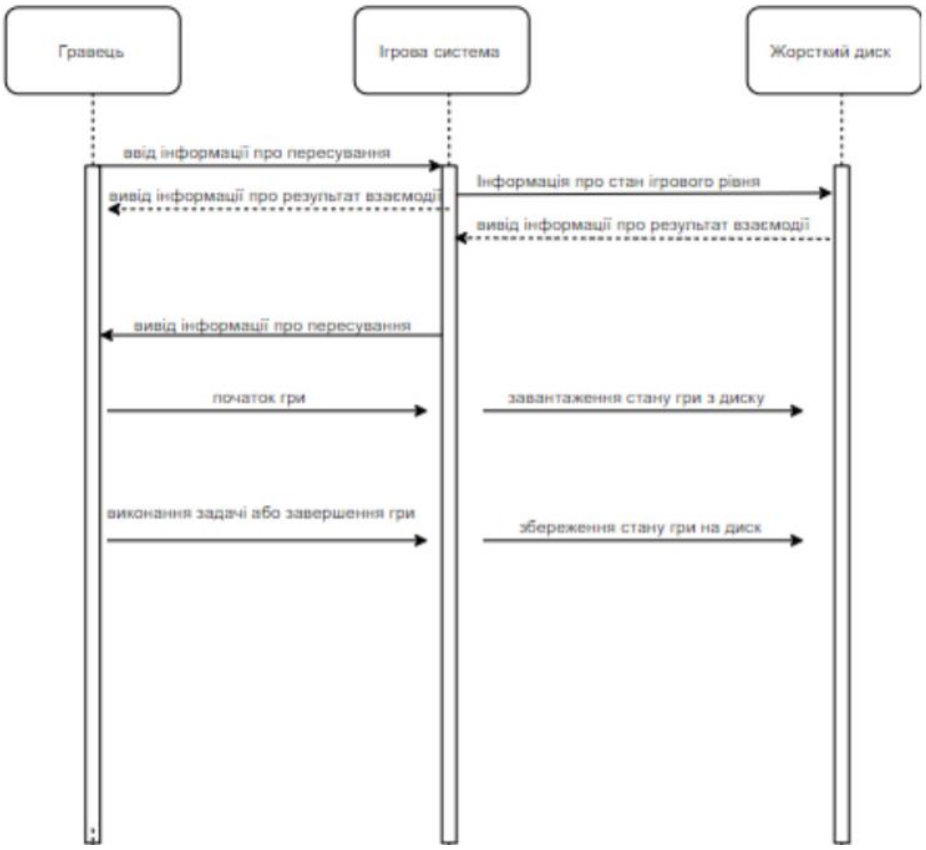

Рисунок 3.4 - Діаграма послідовності

Висновок до розділу 3

Під час складання розділу, були застосовані знання та навички стосовно розробки гри на базі Unity та проаналізовані аспекти до програмного забезпечення. Таким чином, у третьому розділі кваліфікаційної роботи бакалавра застосовано навички креслення UML-діаграм. Заради наглядного показу як буде працювати ігровий застосунок.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розробка скриптів для програмного забезпечення

Програмування скриптів – це спосіб написання інструкцій для початку гри дуже специфічною мовою. Це схоже на правила настільної гри, які пояснюють порядок ходів, визначення першого гравця та різні варіанти в певний момент часу. Різниця в тому, що підручники з настільними іграми написані для розуміння людей, тоді як для програмування потрібна мова, яку розуміє комп'ютер.

Скрипт — це програма або файл сценарію, який автоматизує завдання, яке користувач виконує вручну за допомогою інтерфейсу програми. Скрипти написані мовами сценаріїв, які відрізняються за синтаксисом, сферою дії та можливостями. Мови сценаріїв включають: AngelScript, Perl, Python, PHP, JavaScript, JScript тощо.

Можливість скриптів величезна. Наприклад:

- можливість доступу до баз даних;
- seo-скрипти, які допомагають просувати сайт за допомогою спеціальних програм автоматизації браузера;
- Лічильники відвідувань, які допомагають відстежувати статистику відвідувань;
- можливість створювати записи в гостьових книгах;
- можливість залишати коментарі до улюблених статей;
- усі cms та форуми створені на основі скриптів;
- динамічний веб-відображення;
- організація зміни частини сайту без перезавантаження всієї сторінки тощо.

Скрипти іноді використовуються для налаштування й автоматизації повторюваних завдань та керування загальними функціями комп'ютера. Прикладом такого скрипту є Autoexec.bat. Скрипти Visual Basic і DOS запускають процеси в Windows, а скрипти Apple автоматизують завдання на комп'ютерах Mac. Коли сценарії відкриваються через механізми сценаріїв, команди в сценаріях виконуються.

Unity дозволяє створювати власні компоненти за допомогою скриптів. Вони дозволяють активувати ігрові події, змінювати налаштування компонентів і будь-яким чином реагувати на дії користувача.

Unity підтримує дві мови програмування:

- C #, стандартна мова в цій галузі, схожа на Java або C ++;
- UnityScript, мова, розроблена спеціально для використання в Unity на основі JavaScript.

```
public class cameraController : MonoBehaviour {
    private GameObject Player;
    private controller RR;
    private GameObject cameralookAt,cameraPos;
    private float speed = 0;
    private float defaltFOV = 0, desiredFOV = 0;
    [Range (0, 50)] public float smothTime = 8;

    private void Start () {
        Player = GameObject.FindGameObjectWithTag ("Player");
        RR = Player.GetComponent<controller> ();
        cameralookAt = Player.transform.Find ("camera lookAt").gameObject;
        cameraPos = Player.transform.Find ("camera constraint").gameObject;
```

```
defaltFOV = Camera.main.fieldOfView;
desiredFOV = defaltFOV + 15;
    }
```

Наведено скрипт cameraController написаний на мові C#. У цьому шматку коді буде розглянуто параметри камери. Далі буде наведений приклад, як закріпити камеру за гравцем.

```
private void follow () {
    speed = RR.KPH / smothTime;
    gameObject.transform.position = Vector3.Lerp (transform.position,
cameraPos.transform.position ,  Time.deltaTime * speed);
    gameObject.transform.LookAt (cameralookAt.gameObject.transform.position);
    }

  public void engineUpgradeButton(){
    if (engineFlag) return;
    xPointer = -520;
    int[] length =
list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().P_engin
eUpgrade;
    generatedArray = new GameObject[length.Length];
    for (int i = 0; i < length.Length ; i++) {
       generateArray("engineUpgrade", engineUpgrade, length, i, enginePrice,
PlayerPrefs.GetInt((list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<cont
roller>().carName + "engineUpgrade").ToString())
        ) ;
    }

    engineFlag = true;
```

```
        disablePanel();
        exitPanelButton.SetActive(false);
            }
```

Наведено скрипт engineUpgradeButton написаний на мові C#. В даному шматку коді розглянуто параметри для модернізування двигуна автомобіля, яким керує гравець. Нижче буде розглянуто функцію покупки налаштування.

```
    private void buy(string testUpgradeString, int i, int k,int upgradeType){
        if(PlayerPrefs.GetInt("currency") >= i && upgradeType < k )
        {
            PlayerPrefs.SetInt("currency", PlayerPrefs.GetInt("currency") - i);
            applyPurchase(testUpgradeString , k);
            refreshCurrentCanvas();
        }

        if (i == 0)
        {
            applyPurchase(testUpgradeString, k);
            refreshCurrentCanvas();
        }
    }

    private void applyPurchase(string testUpgradeString ,int value){

        switch (testUpgradeString){
            case "engineUpgrade":

list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_engine
Upgrade = value;
```

```
            break;
        case "spoilerUpgrade":

list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_spoiler
Upgrade = value;
            break;
        case "nitrusUpgrade":

list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_nitrus
Upgrade = value;
            break;
        case "wheelUpgrade":

list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_wheel
Upgrade = value;
            break;
        case "colorUpgrade":

list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_colorU
pgrade = value;
            break;
        case "handlingUpgrade":
```

```
list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().I_handli
ngUpgrade = value;
            break;
        }
list.vehicles[PlayerPrefs.GetInt("pointer")].GetComponent<carModifier>().test();
```

Макроси є поширеними сценаріями. Вони працюють з графічними вікнами, кнопками та меню, створеними системою для імітації дій користувача. Вони також записують натискання клавіш для полегшення повторюваних завдань і виконують їх з меншою кількістю натискань клавіш. Кожен користувач комп'ютера використовує певні скрипти, навіть якщо вони цього не знають. Більшість користувачів використовують скрипти, які встановлюють нове програмне забезпечення. Вони крок за кроком ведуть користувачів через процес встановлення та зупиняються в місцях, де доступні різні варіанти.

В даній функції applyPurchase описується підтвердження вибору певних налаштувань. Таких як, двинуг, спойлер, вприск нітро, колір.

```
 private void moveVehicle(){

    brakeVehicle();

    if (drive == driveType.allWheelDrive){
       for (int i = 0; i < wheels.Length; i++){
          wheels[i].motorTorque = totalPower / 4;
          wheels[i].brakeTorque = brakPower;
       }
    }else if(drive == driveType.rearWheelDrive){
       wheels[2].motorTorque = totalPower / 2;
       wheels[3].motorTorque = totalPower / 2;
```

```
        for (int i = 0; i < wheels.Length; i++)
        {
            wheels[i].brakeTorque = brakPower;
        }
    }
    else{
        wheels[0].motorTorque = totalPower / 2;
        wheels[1].motorTorque = totalPower / 2;

        for (int i = 0; i < wheels.Length; i++)
        {
            wheels[i].brakeTorque = brakPower;
        }
    }
    KPH = rigidbody.velocity.magnitude * 3.6f;
}
```

Наведено скрипт moveVehicle написаний на мові C#. Ця функція відповідає за переміщення автівки по ігровому полю. Та залежить від ще одних функцій, які будуть наведені у додатку.

Настав час розібрати ефекти від автомобіля під час використання ручного тормозу, реалізації фар та слід від прискорювання.

```
private void activateLights() {
    if (IM.vertical < 0 || controller.KPH <= 1) turnLightsOn();
    else turnLightsOff();
        }

  private void chectDrift() {
    if (IM.handbrake) startEmitter();
    else stopEmitter();
        }
```

При завершенні гоночної ділянки. Буде показано кінцевий результат, по завершенню гонки.

```
public class finishTrigger : MonoBehaviour{

  public controller Controller;

  private void OnTriggerEnter(Collider other) {
    if(other.tag == "Finish")
      Controller.hasFinished = true;
  }
    }
```

4.1 Тестування програмного забезпечення

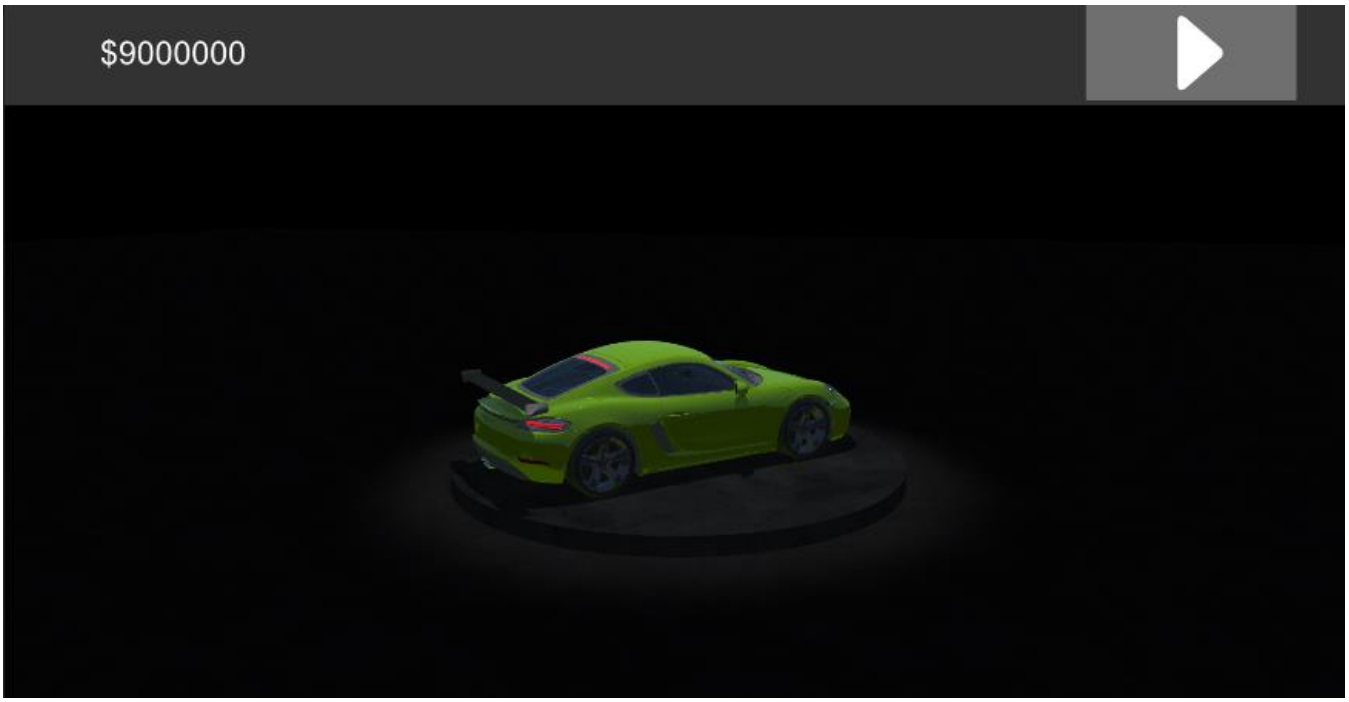

Рисунок 4.1 – Початкове головне меню

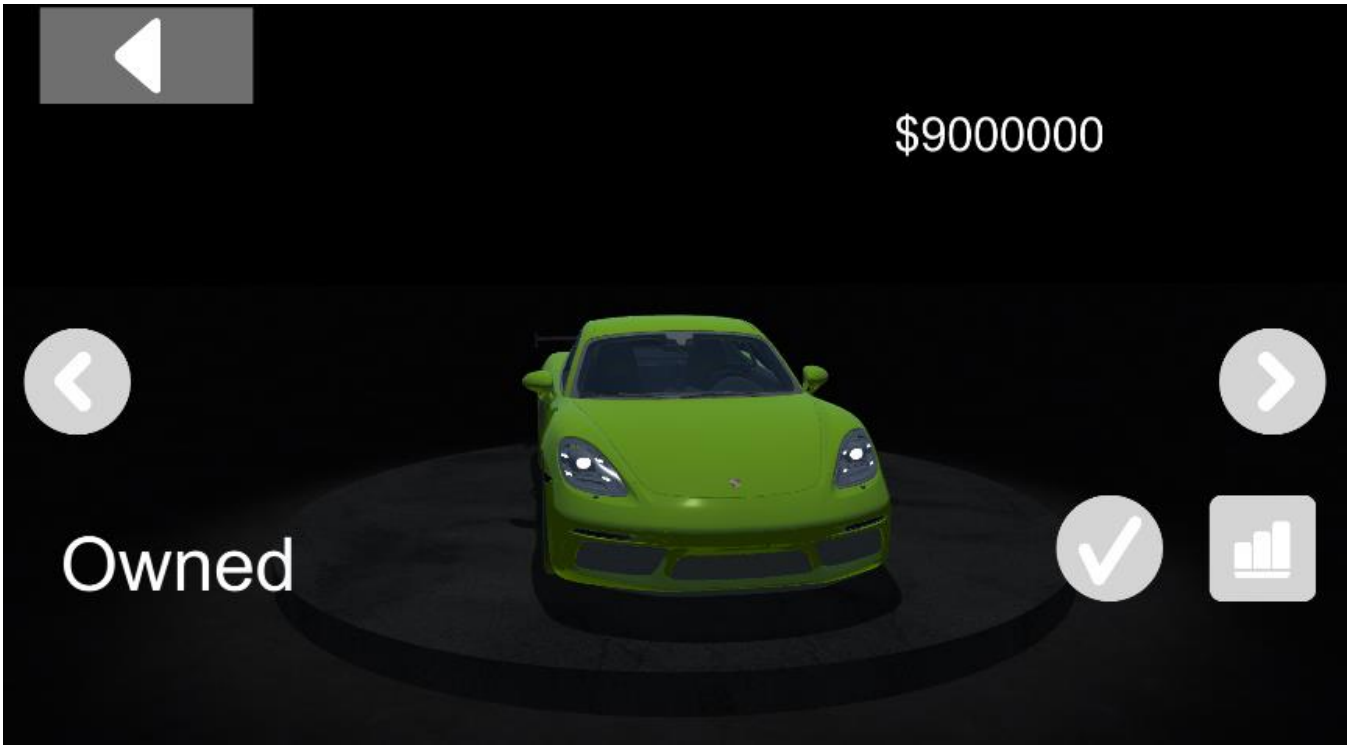

Рисунок 4.2 –Головне меню

Рисунок 4.3 – Меню покращення автомобіля

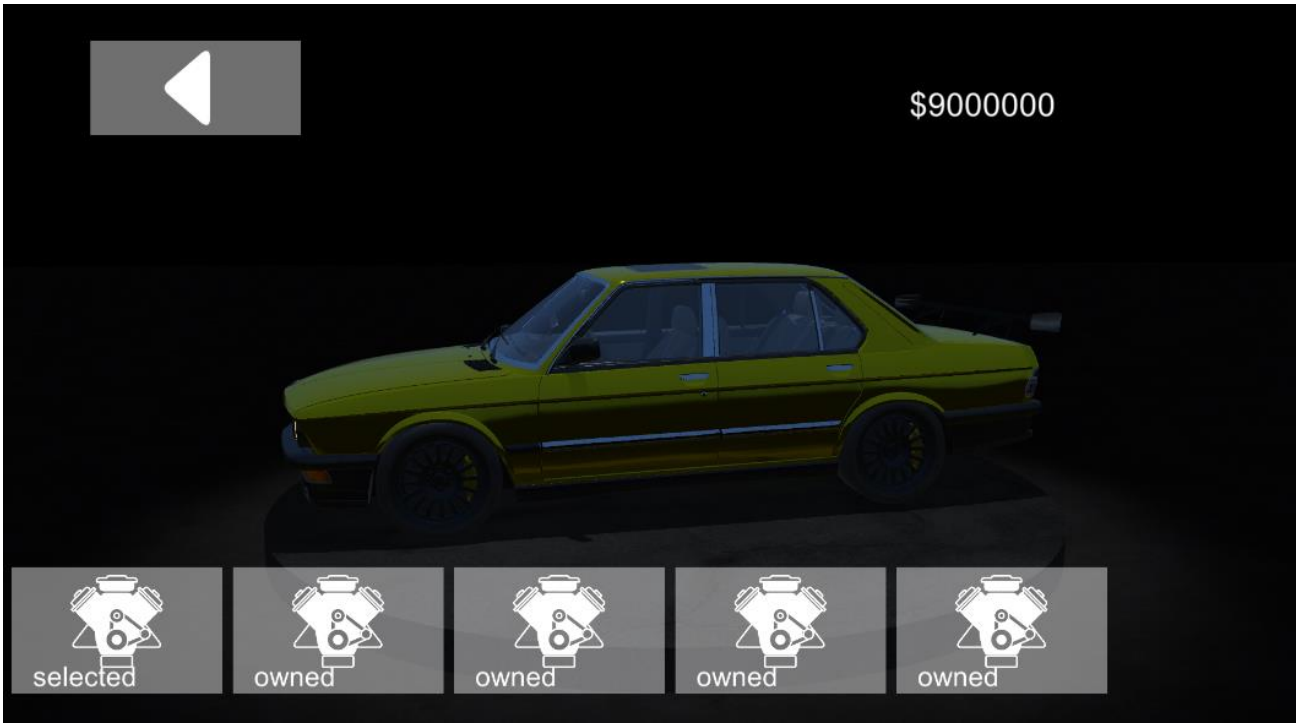

Рисунок 4.4 – Вибір двигуна

Рисунок 4.5 – Вибір нітро прискорювача

Рисунок 4.5 – Вибір спойлеру

Рисунок 4.7 – Вибір кольору

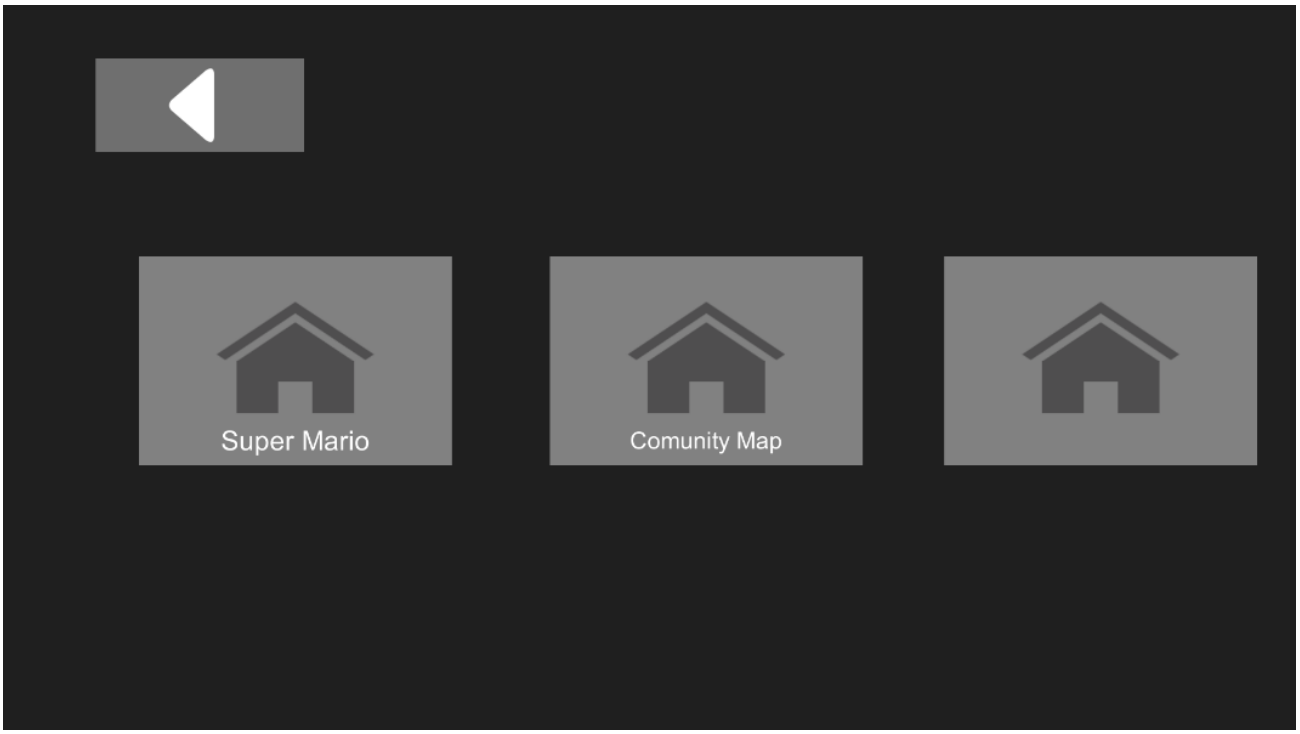

Рисунок 4.8 – Вибір траси

Рисунок 4.9 – Старт

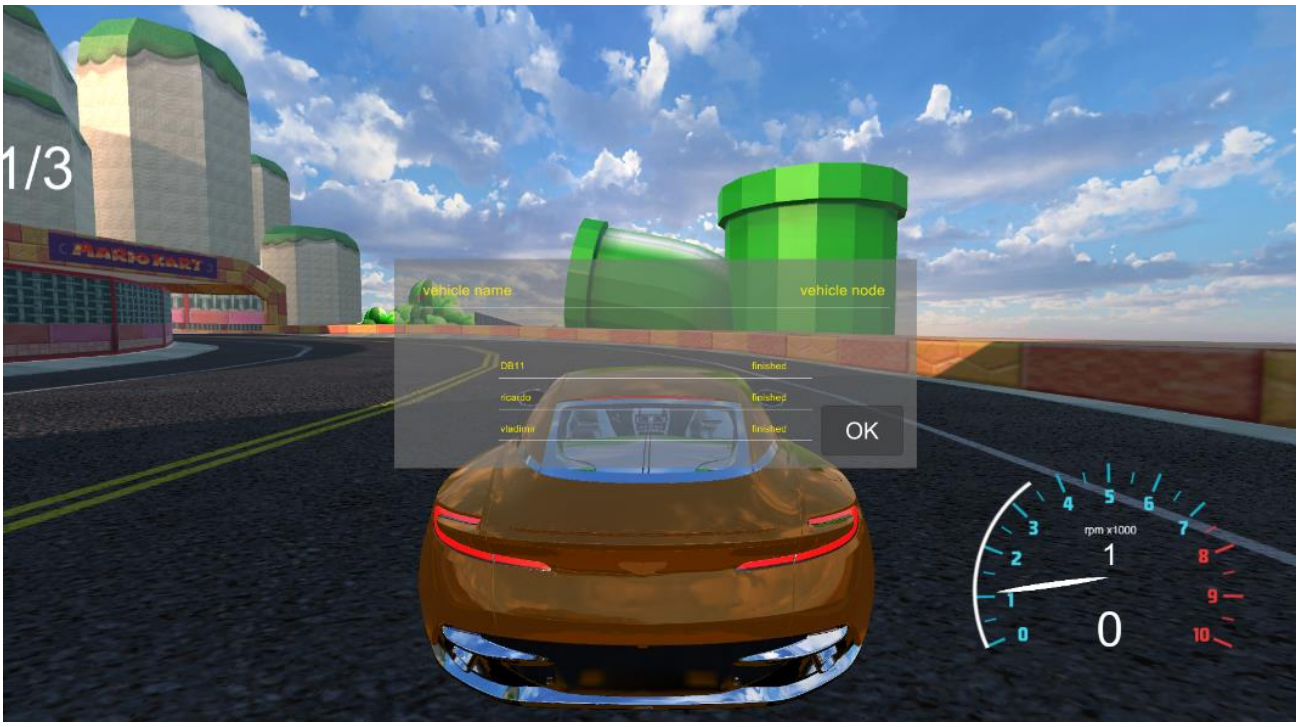

Рисунок 4.11 – Фініш

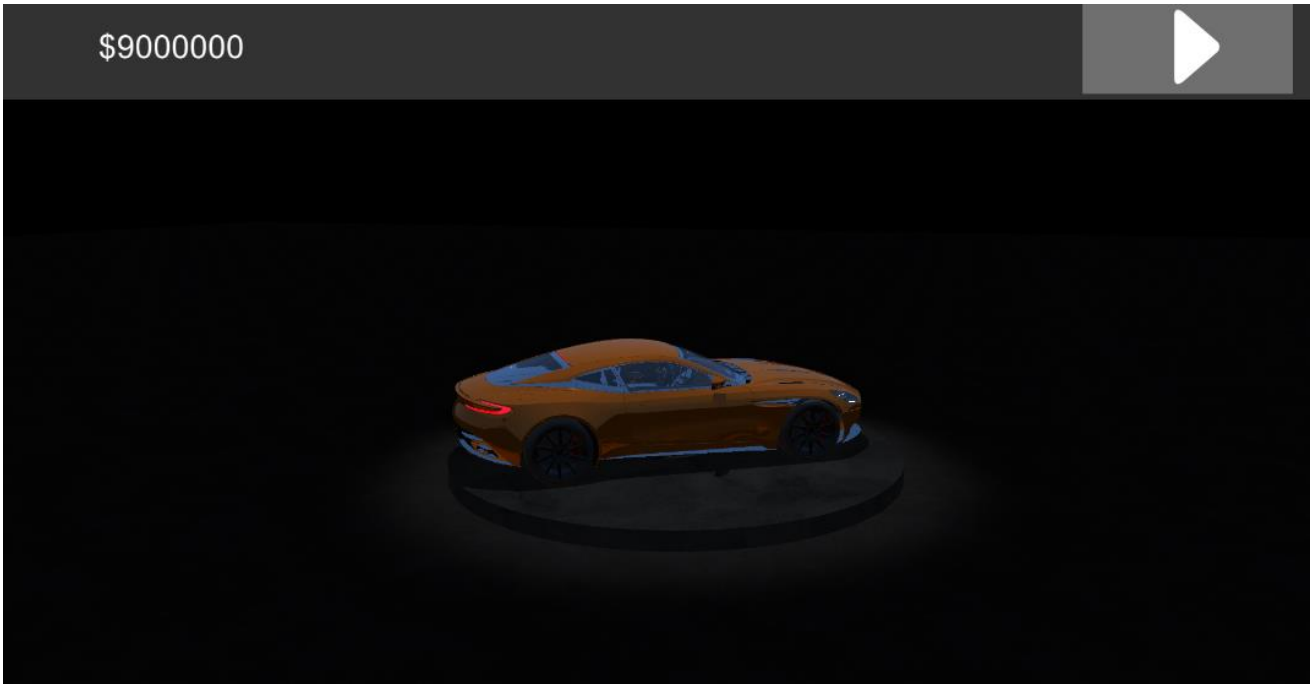

Рисунок 4.12 – Повернення на початковий екран

Висновок до розділу 4

Під час складання розділу, були застосовані знання та навички стосовно розробки гри на базі Unity та проаналізовані аспекти до програмного забезпечення. Unity дозволяє створювати свої компоненти, використовуючи скрипти. Вони дозволяють активувати ігрові події, змінювати параметри компонентів, і відповідати на дії користувача яким завгодно способом. Було продемонстровано певні та загальні скрипти, їх опис та проведення тестування програмного забезпечення.

ВИСНОВОК

Під час виконання кваліфікаційної роботи було проведено аналіз предметної області. В аналітичній частині першого розділу було описано вимоги до розробки ігрового застосунку, порівняння з конкурентами. Обґрунтування вибору програмного забезпечення для реалізації завдання.

У другому розділі, завдяки аналізу засобів реалізації, було обрано мову програмування, двигун, для майбутнього проекту, та програмне забезпечення для реалізації моделей автівок для гри. Також було проведено порівняння мов програмувань, типів ігрових двигунів.

У третьому розділі було змодельовано та продемонстровано UML - діаграми для програмного застосунку. Було розглянуто такі діаграми як: діаграма класів, діаграма компонентів, діаграма використання та діаграма послідовності.

У четвертому розділі було проведено реалізацію програмного забезпечення. Наведені приклади коду - це скрипти для ігрового застосунку, це управління та ефекти автомобіля, апгрейди автівок, налаштування камери та інше. Також було проведено тестування програмного забезпечення. Воно продемонстровано ілюстраціями.

Завдання, які були поставленні при ході виконання кваліфікаційної роботи бакалавра, були виконані у повному обсязі.

ПЕРЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 3D Adventure Game Using Unity / R. R et al. Bonfring International Journal of Software Engineering and Soft Computing. 2019. Vol. 9, no. 2. P. 16–20. URL: https://doi.org/10.9756/bijsesc.9015 (дата звернення: 10.05.2022).

2. Ahearn L. 3D game environments: Create professional 3D game worlds. Amsterdam : Elsevier/Focal Press, 2008. P. 4-23 (дата звернення: 10.05.2022).

3. Unity documentation URL: https://docs.unity3d.com/ (дата звернення 10.06.2022).

4. Розбір ігровоги двигуна Unity3d. URL: https://gamedevmania.ru/engines/unity3d. (дата звернення 10.06.2022).

5. Двигун Unity – унікальність, переваги та недоліки. URL: https://cubiq.ru/dvizhok-unity/. (дата звернення 10.06.2022).

6. Мова прогамуваня C#: Детальний розбір. URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/. (дата звернення 10.06.2022).

7. Офіційний сайт Unity 3D. URL: https://unity.com/ (дата звернення 10.06.2022).

8. Особливості підготовки 3D – об’єктів, смодельованих у Blender для імпорту в Unity 3D. URL: https://cyberleninka.ru/ (дата звернення 10.06.2022).

9. Learning C By Developing Games With Unity 3d. Packt Publishing Limited, 2013. Publishing V. G. D. Video Game Developing Difficulty Level: Journal / Notebook / Diary Gift - 6 X9 - 120 Pages - White Lined Paper - Matte Cover. Independently Published, 2020. 120 p.

10. Taylor A. G. Developing with Unity and Visual Studio. *Develop Microsoft HoloLens Apps Now*. Berkeley, CA, 2016. P. 75–90. URL: https://doi.org/10.1007/978-1-4842-2202-7_9 (дата звернення: 11.05.2022).

11. Unity 4 Fundamentals Making Games With Unity. Taylor & Francis Ltd, 2013. (дата звернення: 11.05.2022).

12. Patil P. Software for Programing Contest. *International Journal for Research in Applied Science and Engineering Technology*. 2019. Vol. 7, no. 5. P. 3712–3714. URL: https://doi.org/10.22214/ijraset.2019.5610 (дата звернення: 11.05.2022).

13. Lanzinger F. Epilogue. *3D Game Development with Unity*. Boca Raton, 2022. P. 389–390. URL: https://doi.org/10.1201/9780429328725-25 (дата звернення: 11.05.2022).

14. Romphf J. Coding Activities for Developing Games in Unity. Rosen Publishing Group, 2020. 64 p. (дата звернення: 11.05.2022).

15. Лекції по Unity. URL: https://zinref.ru/000_uchebniki/02800_logika/011_lekcii_raznie_25/941.htm (дата звернення: 10.06.2022).

ДОДАТОК А

Лістинг коду з параметрами камери

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

public class cameraController : MonoBehaviour {

    private GameObject Player;
    private controller RR;
    private GameObject cameralookAt,cameraPos;
    private float speed = 0;
    private float defaltFOV = 0, desiredFOV = 0;
    [Range (0, 50)] public float smothTime = 8;

    private void Start () {
        Player = GameObject.FindGameObjectWithTag ("Player");
        RR = Player.GetComponent<controller> ();
        cameralookAt = Player.transform.Find ("camera lookAt").gameObject;
        cameraPos = Player.transform.Find ("camera constraint").gameObject;

        defaltFOV = Camera.main.fieldOfView;
        desiredFOV = defaltFOV + 15;
    }

    private void FixedUpdate () {
        follow ();
        boostFOV ();

    }
    private void follow () {
        speed = RR.KPH / smothTime;
        gameObject.transform.position = Vector3.Lerp (transform.position,
cameraPos.transform.position ,  Time.deltaTime * speed);
        gameObject.transform.LookAt (cameralookAt.gameObject.transform.position);
    }
    private void boostFOV () {

        if (RR.nitrusFlag)
```

```
        Camera.main.fieldOfView = Mathf.Lerp (Camera.main.fieldOfView, desiredFOV, Time.deltaTime * 5);
    else
        Camera.main.fieldOfView = Mathf.Lerp (Camera.main.fieldOfView, defaltFOV, Time.deltaTime * 5);

  }

}
```

ДОДАТОК Б

Лістинг коду для меню модифікації автомобіля

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

public class carModifier : MonoBehaviour
{
  public controller controllerRef;
  [Header("engine")]
  public bool engineUpgrade;
  public int I_engineUpgrade = 0;
  public int[] P_engineUpgrade;

  [Header("spoiler")]
  public bool spoilerUpgrade;
  public int I_spoilerUpgrade = 0;
  public GameObject[] L_spoilerUpgrade;
  public int[] P_spoilerUpgrade;
```

```
[Header("nitrus")]
public bool nitrusUpgrade;
public int I_nitrusUpgrade = 0;
public int[] P_nitrusUpgrade;


[Header("color")]
public GameObject car;
public bool colorUpgrade;
public int I_colorUpgrade = 0;
public Material[] L_colorUpgrade;
public int[] P_colorUpgrade;


[Header("wheels")]
public bool wheelUpgrade;
public int I_wheelUpgrade = 0;
public GameObject[] L_wheelUpgrade;
public int[] P_wheelUpgrade;

[Header("handling")]
public bool handlingUpgrade;
public int I_handlingUpgrade = 0;
public int[] P_handlingUpgrade;

private void Start(){
```

```
        updateValues();


    }
    private void Update()
    {
        if (SceneManager.GetActiveScene().name != "awakeScene") return;
        updateValues();
    }
    public void applyUpgrades(){


        if (I_engineUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"engineUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName + "engineUpgrade").ToString(),
I_engineUpgrade);
            updatePrices(P_engineUpgrade,I_engineUpgrade);
        }
        else if (I_colorUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"colorUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName + "colorUpgrade").ToString(),
I_colorUpgrade);
            updatePrices(P_colorUpgrade, I_colorUpgrade);
        }
        else if (I_handlingUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"handlingUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName + "handlingUpgrade").ToString(),
I_handlingUpgrade);
            updatePrices(P_handlingUpgrade, I_handlingUpgrade);
        }
```

```
        else if (I_nitrusUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"nitrusUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName +
"nitrusUpgrade").ToString(),I_nitrusUpgrade);
            updatePrices(P_nitrusUpgrade, I_nitrusUpgrade);
        }
        else if (I_spoilerUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"spoilerUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName +
"spoilerUpgrade").ToString(),I_spoilerUpgrade);
            updatePrices(P_spoilerUpgrade,I_spoilerUpgrade);
        }
        else if (I_wheelUpgrade != PlayerPrefs.GetInt((controllerRef.carName +
"wheelUpgrade").ToString())) {
            PlayerPrefs.SetInt((controllerRef.carName +
"wheelUpgrade").ToString(),I_wheelUpgrade);
            updatePrices(P_wheelUpgrade, I_wheelUpgrade);
        }


    }

    public void test() {
        applyUpgrades();
        updateValues();
    }
```

```
    private void applySpoilerUpgrade(){
        if (!spoilerUpgrade) return;
        foreach (GameObject G in L_spoilerUpgrade)
        {
            G.SetActive(false);
        }
        L_spoilerUpgrade[I_spoilerUpgrade].SetActive(true);
    }
    private void applyColorUpgrade(){
        if (!colorUpgrade) return;

        car.GetComponent<MeshRenderer>().material =
L_colorUpgrade[I_colorUpgrade];
    }
    private void updateValues(){
        I_engineUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"engineUpgrade").ToString());
        I_colorUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"colorUpgrade").ToString());
        I_handlingUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"handlingUpgrade").ToString());
        I_nitrusUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"nitrusUpgrade").ToString());
```

```
        I_spoilerUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"spoilerUpgrade").ToString());
        I_wheelUpgrade = PlayerPrefs.GetInt((controllerRef.carName +
"wheelUpgrade").ToString());

        applySpoilerUpgrade();
        applyColorUpgrade();

    }


    private void updatePrices(int[] givenArray,int index){
        if (index == 0) return;
        for (int i = 0; i < givenArray.Length;i++ ) {
            if (i <= index) {
                givenArray[i] = 0;

            }
        }

    }
}
```

ДОДАТОК В

Лістинг коду для візуальних ефектів автомобіля

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

public class carEffects : MonoBehaviour
{
    public Material brakeLights;
    public AudioSource skidClip;
    public TrailRenderer[] tireMarks;
    public ParticleSystem[] smoke;
    public ParticleSystem[] nitrusSmoke;
    //public GameObject lights;
    private controller controller;
    private inputManager IM;
    private bool smokeFlag  = false , lightsFlag = false , tireMarksFlag;

    //do lights in here
    private void Start() {
        if(gameObject.tag == "AI")return;
        controller = GetComponent<controller>();
        IM = GetComponent<inputManager>();

    }

    private void FixedUpdate() {
        if(gameObject.tag == "AI")return;
```

```
        chectDrift();
        activateSmoke();
        activateLights();
    }

    private void activateSmoke(){
        if (controller.playPauseSmoke) startSmoke();
        else stopSmoke();

        if (smokeFlag)
        {
            for (int i = 0; i < smoke.Length; i++)
            {
                var emission = smoke[i].emission;
                emission.rateOverTime = ((int)controller.KPH * 10 <= 2000) ?
(int)controller.KPH * 10 : 2000;
            }
        }
    }

    public void startSmoke(){
        if(smokeFlag)return;
        for (int i = 0; i < smoke.Length; i++){
            var emission = smoke[i].emission;
            emission.rateOverTime = ((int) controller.KPH *2 >= 2000) ? (int)
controller.KPH * 2 : 2000;
            smoke[i].Play();
```

```
    }
    smokeFlag = true;

}

public void stopSmoke(){
    if(!smokeFlag) return;
    for (int i = 0; i < smoke.Length; i++){
        smoke[i].Stop();
    }
    smokeFlag = false;
}

public void startNitrusEmitter()
{
    if (controller.nitrusFlag) return;
    for (int i = 0; i < nitrusSmoke.Length; i++)
    {
        nitrusSmoke[i].Play();
    }

    controller.nitrusFlag = true;
}
public void stopNitrusEmitter()
{
    if (!controller.nitrusFlag) return;
    for (int i = 0; i < nitrusSmoke.Length; i++)
    {
```

```
            nitrusSmoke[i].Stop();
        }
        controller.nitrusFlag = false;

    }

    private void activateLights() {
        if (IM.vertical < 0 || controller.KPH <= 1) turnLightsOn();
        else turnLightsOff();
    }

    private void turnLightsOn(){
        if (lightsFlag) return;
        brakeLights.SetColor("_EmissionColor", Color.red *5);
        lightsFlag = true;
        //lights.SetActive(true);
    }

    private void turnLightsOff(){
        if (!lightsFlag) return;
        brakeLights.SetColor("_EmissionColor", Color.black);
        lightsFlag = false;
        //lights.SetActive(false);
    }

    private void chectDrift() {
        if (IM.handbrake) startEmitter();
        else stopEmitter();
```

```
    }

    private void startEmitter() {
        if (tireMarksFlag) return;
        foreach (TrailRenderer T in tireMarks) {
            T.emitting = true;
        }
        skidClip.Play();
        tireMarksFlag = true;
    }
    private void stopEmitter() {
        if (!tireMarksFlag) return;
        foreach (TrailRenderer T in tireMarks)
        {
            T.emitting = false;
        }
        skidClip.Stop();
        tireMarksFlag = false;
    }

}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

3D-ГРА У ЖАНРІ АРКАДИ З ІНТЕРАКТИВНИМ КОНТЕНТОМ НА БАЗІ UNITY

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

Спеціальність «Інженерія програмного забезпечення»
121 – КРБ.1 – 408.21810806

Студент _____ **В.С.Вальковський**
підпис
«__» _____ 2022 р.

Консультант кан.тех.наук., доцент _____ **А. О. Алексєєва**
підпис
«__» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПК – Персональний комп’ютер

МОЗ – Міністерство охорони здоров’я

ДСН – Державні санітарні норми

ДСанПІН – Державні санітарні правила та норми

МЮУ – Міністерство юстиції України

ВСТУП

У зв’язку з розвитком інформаційних технологій кожне підприємство вирішило їх впроваджувати для полегшення та автоматизації роботи за допомогою технічних пристроїв. Як приклад, можна навести використання ноутбуків та персональних комп’ютерів, які мають на меті збір, аналіз та обмін інформацією всередині та за межами підприємств або установ. Використання комп’ютера полегшує процеси на підприємствах.

Взявши до уваги переваги інтеграції інформаційних технологій за допомогою технічних пристроїв, треба ще враховувати також і недоліки їх інтеграції у робочий процес. Головним недоліком є загострення суспільного та власного здоров’я, яке потребує створення та удосконалення існуючих робочих місць, які матимуть більш сприятливі до здоров’я умови робочого місця. Також треба виділити проведення заходів спрямованих на запобігання розвитку негативних наслідків впливу персонального комп’ютера на здоров’я користувача.

Велика кількість проблем зі здоров’ям виникає через використання персонального комп’ютера або ноутбука. Наприклад, проблеми зі спиною, проблеми із зором або синдром зап’ястного каналу.

Вплив на зір дисплея залежить від його положення. Одним із негативних факторів є частота динамічного оновлення дисплею, яка при маленькій частоті оновлення супроводжується мерехтінням, яке має негативні вплив на зір. За низької частоти оновлення дисплею очні м’язи перебувають у великому навантаженні, що може викликати розвинення короткозорості або втрати зору.

1 НОРМАТИВНА БАЗА З ОХОРОНИ ПРАЦІ

1.1 Загальні положення

Дія інструкції поширюється на всі підрозділи підприємства, де виконують роботи з персональним комп'ютером (ПК).

Інструкція розроблена відповідно до Положення про розробку інструкцій з охорони праці, затвердженого наказом Держнаглядохоронпраці від 29.01.1998 № 9, Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці від 26.01.2005 № 15, Правил охорони праці під час експлуатації електронно-обчислювальних машин, затверджених наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65, Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджених постановою Головного державного санітарного лікаря України від 10.12.1998 № 7, Загальних вимог стосовно забезпечення роботодавцями охорони праці працівників, затверджених наказом Міністерства надзвичайних ситуацій України від 25.01.2012 № 67 (НПАОП 0.00-7.11-12).

За цією інструкцією працівника, який використовує персональний комп'ютер (*далі* — користувач), інструктують перед початком роботи (первинний інструктаж), а потім через кожні 6 місяців (повторний інструктаж). Результати інструктажу заносять до Журналу реєстрації інструктажів з питань охорони праці на робочому місці (у журналі має бути підпис особи, яка інструктує, та користувача).

Користувач зобов'язаний дбати про особисту безпеку і здоров`я, а також про безпеку і здоров'я довколишніх при виконанні будь-яких робіт, а також під час перебування на території підприємства.

До роботи на персональному комп'ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки.

Користувач зобов'язаний:

- виконувати правила внутрішнього трудового розпорядку;
- не допускати за своє робоче місце сторонніх осіб;
- не виконувати вказівок, які суперечать правилам охорони праці та пожежної безпеки;
- знати правила надання до медичної допомоги;
- знати розташування та вміти користуватись первинними засобами пожежогасіння;
- вміти працювати з ПК.

Основні небезпечні та шкідливі виробничі фактори, що можуть впливати на користувача:

- підвищений рівень статичної електрики;
- нерівномірність розподілу яскравості в полі зору;
- підвищена яскравість світлового зображення;
- ураження електричним струмом;
- напруга зору та уваги;
- тривалі статичні навантаження.

У приміщеннях із ПК має бути природне і штучне освітлення.

При розміщенні робочих місць необхідно унеможливити пряме засвічування екрана природним освітленням.

При природному освітленні слід передбачити наявність сонцезахисних засобів (плівка, жалюзі, штори тощо).

Світлові відблиски із клавіатури, екрана та інших частин ПК у напрямку очей користувача неприпустимі.

Основним обладнанням робочого місця є ПК або ноутбук, монітор, клавіатура, маніпулятор, робочий стіл, стілець (крісло).

При розміщенні елементів робочого місця слід враховувати:

- робочу позу користувача;
- простір для розміщення користувача;
- можливість огляду елементів робочого місця;
- можливість огляду простору поза межами робочого місця;
- можливість робити записи, розміщувати на робочому столі документацію та матеріали, які використовує користувач.

Розміщення елементів робочого місця не має заважати рухам та переміщенню для експлуатування ПК.

Монітор встановлюють так, щоб відстань від поверхні екрана до очей користувача була 600-700 мм залежно від розміру екрана.

Клавіатуру розміщують на робочому або окремому столі на відстані 100-300 мм від краю з боку користувача. Положення клавіатури та кут її нахилу залежить від побажання користувача (як правило, в межах 5-15°). Не допускати хитання клавіатури.

Конструкція робочого столу має бути такою, щоб оптимально розмістити на робочій поверхні обладнання, що використовують, з урахуванням кількості, розмірів, конструктивних особливостей і характеру його роботи.

Крісло має забезпечувати підтримування раціональної робочої пози під час виконання основних виробничих операцій та можливість зміни пози. Тип робочого крісла обирають залежно від характеру та тривалості роботи.

Раціональна поза користувача:

- ступні розташовані на підлозі або на підставці для ніг;
- стегна зорієнтовані у горизонтальній площині;
- верхні ділянки рук вертикальні;
- кут ліктьового суглоба у межах 70-90°;
- зап’ястя зігнуті під кутом не більше ніж 20°;
- нахил голови у межах 15-20°, а часті її повороти виключені.

Для забезпечення оптимальної робочої пози користувача необхідно:

- засоби праці, з якими користувач має тривалий або найбільш частий

зоровий контакт, розмістити у центрі зони зорового спостереження та моніторного поля;

– забезпечити відстань близько 500 мм між найважливішими засобами праці, з якими користувач працює найчастіше.

ПК встановлювати на рівній твердій поверхні (столі). Не дозволено встановлювати ПК та оргтехніку на хитких підставках чи на похилій поверхні.

ПК не встановлювати впритул до стіни, перегородки тощо. Не допускати загородження вентиляційних отворів ПК сторонніми предметами.

Розетка біля ПК має бути в доступному місці, щоб в аварійних випадках можна було своєчасно його відімкнути. Не рекомендовано використовувати подовжувачі.

Під час переміщення ПК, периферійних пристроїв витягти вилку живлення з розетки.

Не допускати ушкодження чи модифікування шнура живлення. Заборонено ставити важкі речі на шнур живлення, тягнути чи надмірно перегинати його, скручувати та перев’язувати шнур живлення вузлом.

ПК під’єднувати до електромережі лише за допомогою справних штепсельних з’єднань та електророзеток заводського виробництва.

Штепсельні з’єднання та електророзетки мають бути зі спеціальними контактами для під’єднання нульового захисного провідника. Їхня конструкція має забезпечувати з’єднання нульового захисного провідника раніше, ніж з’єднання фазового та нульового робочого провідників. Порядок роз’єднань при вимкненні має бути зворотнім.

Заборонено під’єднувати електрообладнання до звичайної двошнурової електромережі.

За невиконання цієї інструкції працівники несуть відповідальність згідно з чинним законодавством.

1.2 Вимоги безпеки перед початком роботи за комп’ютером

Оглянути робоче місце і навести на ньому лад; впевнитись, що на ньому немає сторонніх предмети, все обладнання і блоки ПК з’єднані з системним блоком з’єднувальними шнурами.

Перевірити надійність встановлення апаратури на робочому столі. Монітор не має стояти на краю стола. Повернути монітор так, щоб було зручно дивитися на екран - під прямим кутом (а не збоку) і трохи зверху вниз; при цьому екран має бути трохи нахиленим - нижній край ближче до користувача.

Перевірити загальний стан апаратури, справність електропроводки, з’єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана.

Вставити вилку в розетку і впевнитися, що вона міцно тримається. Заборонено вставляти і виймати вилку мокрими руками.

Відрегулювати та зафіксувати висоту крісла та зручний для користувача нахил спинки.

За потреби приєднати до комп’ютера необхідну апаратуру (принтер, сканер тощо). Усі кабелі, що з’єднують системний блок із іншими пристроями, вмикати та вимикати лише при вимкненому комп’ютері.

Відрегулювати яскравість свічення, контрастовість монітора.

Про всі виявлені несправності інформувати керівника робіт і не братися до роботи, доки їх не буде усунено.

1.3 Вимоги безпеки під час виконання роботи за комп'ютером

Під час роботи на ПК:

- стійко встановити клавіатуру на робочому столі, не допускаючи її хитання, водночас передбачити можливість її поворотів та переміщень;
- якщо в конструкції клавіатури не передбачено простору для упору долонь, клавіатуру розміщують на відстані не менше 100 мм від краю столу в оптимальній зоні моніторного поля;
- під час роботи на клавіатурі сидіти рівно, не напружуватися;
- щоб зменшити несприятливе навантаження на користувача при роботі з комп'ютерною мишею (вимушена поза, необхідність постійно контролювати якість дій), забезпечити велику вільну поверхню столу для переміщення комп'ютерної миші та зручного упору ліктьового суглоба;
- періодично при вимкненому комп'ютері прибирати пил із поверхонь апаратури спеціальними серветками.

При роботі з ПК заборонено:

- самостійно розбирати та ремонтувати системний блок (корпус ноутбука), монітор, клавіатуру, комп'ютерну мишу тощо;
- встромляти сторонні предмети до вентиляційних отворів ПК, ноутбука або монітора;
- ставити на системний блок ПК та периферійні пристрої металеві предмети, ємкості з водою (вази, горщики для квітів, склянки), оскільки через потрапляння води у середину апарата може виникнути пожежа або ураження електрострумом.

Тривалість безперервної роботи за ПК не має перевищувати 2 год. Після цього необхідно зробити 15-хвилинну перерву.

Якщо виник зоровий дискомфорт або інші неприємні відчуття, необхідно зробити коротку перерву.

Для зниження нервово-емоційного напруження, стомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно під час декількох перерв виконувати комплекс вправ.

1.4 Вимоги безпеки після виконання роботи за комп'ютером

- Зберегти інформацію.
- Вимкнути ПК, монітор чи ноутбук.
- Вимкнути стабілізатор, якщо комп'ютер під'єднаний до мережі через нього.
- Прибрати робоче місце.

1.5 Вимоги безпеки в аварійній ситуації у приміщенні

Аварійні та небезпечні ситуації під час виконання роботи на ПК можуть виникнути у разі: короткого замикання, перевантаження блоку живлення системного блоку, перегрівання, пожежі, поломки крісла тощо.

У разі виникнення аварії або ситуації, що може привести до аварії, нещасного випадку, негайно від'єднати ПК від електромережі, повідомити інцидент керівникові.

Не допускати в небезпечну зону сторонніх осіб.

Якщо стався нещасний випадок, зберегти обстановку в робочій зоні та устаткування у такому стані, в якому вони були на момент події (якщо це не загрожує життю і здоров'ю інших працівників і не призведе до більш тяжких наслідків). Поінформувати про подію керівника робіт (іншу відповідальну особу підприємства) та в подальшому керуватися його вказівками. Вжити заходів, щоб запобігти подібним випадкам у подальшому.

У разі виникнення пожежі (ознак горіння), повідомити керівнику та, за потреби, викликати оперативно-рятувальну службу за телефоном 101 або 112 (назвати адресу та місце виникнення пожежі, наявність людей, повідомити своє

прізвище) та вжити можливих заходів для евакуювання людей, гасіння (локалізації) пожежі наявними засобами пожежогасіння. Пам’ятати, що гасіння електротехнічних пристроїв, які перебувають під напругою, виконувати лише після їх попереднього від’єднання від електромережі. Гасити за допомогою вуглекислотних або порошкових вогнегасників, а в окремих випадках — сухим піском.

За потреби надати потерпілому домедичну допомогу згідно з інструкцією, що діє на підприємстві. У разі подальшого погіршення самопочуття потерпілого, не припиняючи надання домедичної допомоги, викликати за телефоном 103 швидку медичну допомогу.

Виконувати вказівки керівника робіт для ліквідування небезпеки.

1.6 Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот

Всі прилади повинні бути перевірені в органах Держстандарту. До та після вимірювань проводять акустичну або електричну калібровку вимірювальних приладів. Різниця в калібровці не повинна перевищувати 1 дБ. Порядок вимірювання рівнів звуку шумомірами та розрахунок еквівалентного рівня регламентується ДСН 3.3.6.037-99

При проведенні вимірювань мікрофон слід розташовувати на висоті 1,5 м над рівнем підлоги чи робочого майданчика (якщо робота виконується стоячи) чи на висоті і відстані 15 см від вуха людини, на яку діє шум (якщо робота виконується сидячи чи лежачи). Мікрофон повинен бути зорієнтований у напрямку максимального рівня шуму та віддалений не менш ніж на 0,5 м від оператора, який проводить вимірювання. При швидкості руху повітря більш ніж 1 м/с на місці, де проводяться виміри, мікрофон має бути захищений протиповітряним пристроєм.

При проведенні вимірювань октавних рівнів звукового тиску перемикач частотної характеристики пристрою встановлюють в положенні “фільтр”.

Октавні рівні звукового тиску вимірюють у смугах з середньогеометричними частотами 31,5 – 8000 Гц.

При проведенні вимірювань рівнів звуку та еквівалентних рівнів звуку, дБА, дБАекв. перемикач частотної характеристики пристрою встановлюють у положенні “А” (за допомогою відповідних фільтрів знижена чутливість на низьких та високих частотах) чи “Аекв”.

При проведенні вимірювань рівнів шуму та октавних рівнів звукового тиску постійного шуму перемикач часової характеристики пристрою встановлюють в положення “повільно”. Значення рівнів приймають за середніми показниками при коливанні стрілки пристрою. Значення рівнів шуму та октавних рівнів звукового тиску зчитують зі шкали пристрою з точністю до 1 дБА, дБ. Вимірювання рівнів шуму та октавних рівнів звукового тиску постійного шуму повинні бути проведені у кожній точці не менше трьох разів.

При проведенні вимірювань еквівалентних рівнів шуму, що коливаються в часі, для визначення еквівалентного (за енергією) рівня шуму перемикач часової характеристики пристрою встановлюють в положенні “повільно”. Значення рівнів шуму приймають за показниками стрілки пристрою у момент відліку.

При проведенні вимірювань максимальних рівнів імпульсного шуму перемикач часової характеристики пристрою встановлюють в положенні "імпульс". Значення рівнів приймають за максимальним показником пристрою.

Таблиця 1 – Допустимі рівні звуку

Вид трудової діяльності, робочі місця	Рівні звукового тиску в дБ
	в октавних смугах із середньогеометричними частотами, Гц

Кінець таблиці 1

	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв.
Програмісти ЕОМ	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ЕОМ та оператори комп'ютерного набору	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів ЕОМ	103	91	83	77	73	70	68	66	64	75

1.7 Норми мікроклімату для приміщень з ВТД ЕОМ та ПЕОМ

Оптимальні умови мікроклімату встановлюються для постійних робочих місць

Показники температури повітря в робочій зоні по висоті та по горизонталі, а також протягом робочої зміни не повинні виходити за межі нормованих величин оптимальної температури для даної категорії робіт.

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання (екрани і т. ін.), зовнішніх поверхонь технологічного устаткування, огороджуючих конструкцій не повинна виходити більш ніж на 2 град.С за межі оптимальних величин температури повітря для даної категорії робіт.

При виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням в кабінетах, пультах і постах керування технологічними процесами, в залах обчислювальної техніки та інших приміщеннях повинні дотримуватися оптимальні умови мікроклімату.

Таблиця 2 – Норми мікроклімату

Пора року	Категорія робіт	Температура повітря, град. С не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	легка-1а	22–24	40–60	0,1
	легка-1б	21–23	40–60	0,1
Теплий	легка-1а	23–25	40–60	0,1
	легка-1б	22–24	40–60	0,2

1.8 Рівні іонізації повітря приміщень при роботі на ВДТ ЕОМ та ПЕОМ

В даний час активно застосовується штучна іонізація повітря, яка може бути здійснена двома шляхами: подачею іонізованого повітря або установкою іонізатора безпосередньо всередині приміщення.

Отримання аероіонів в аероіонізаторі може бути здійснено різними способами, і в залежності від фізичного процесу, використовуваного для отримання аероіонів, аероіонізатори можна класифікувати наступним чином : термоелектронні (плазмові); радіоізотопні; фотоелектричні (ультрафіолетові); гідродинамічні, електричні (високовольтні, коронні) і комбіновані.

У термоелектронних аероіонізаторах, що застосовуються, в основному, для дослідницьких цілей, використовується ефект термоелектронної емісії розпечених металів при їх нагріванні до високої температури. Принцип дії радіоізотопних аероіонізаторів заснований на властивості променів радіоактивних речовин іонізувати повітря.

Такі прилади використовуються для проведення медико-біологічних спостережень. Принцип дії гідродинамічних аероіонізаторів заснований на електризації крапель води при її дробленні або розпиленні. Фотоелектричні аероіонізатори при роботі використовують іонізуючу здатність ртутно-кварцових ламп, що генерують короткохвильові ультрафіолетові промені.

Такі аероіонізатори можна використовувати для 8 іонізації повітря великих приміщень. Принцип дії електричних аероіонізаторів заснований на явищі автоелектронної емісії, що виникає при певному градієнті електричного поля випромінювача.

В даний час для підтримки аероіонного режиму в приміщеннях в основному застосовуються електричні іонізатори коронного розряду, які можуть бути наступних типів: уніполярні; біполярні; ефлювіальні; оснащені вентилятором; що працюють в режимі «темної» або «світиться» корони.

До безперечних переваг сучасних іонізаторів слід віднести порівняно невеликі габарити, легку вагу, транспортабельність, простоту обслуговування. Разом з тим, дані про продуктивність іонізатора наводяться на відстані 1 м від нього, немає можливості автоматичного регулювання емісії іонів, зарубіжні іонізатори відрізняються високої вартістю приладів.

Таблиця 3 – Рівні іонізації повітря

Рівні	Число іонів в 1 куб. см повітря	
	N+	N–
Мінімально необхідні	400	600
Оптимальні	1500–3000	3000–5000
Максимально допустимі	50000	50000

1.9 Допустимі параметри електромагнітних неіонізуючих випромінювань і електростатичного поля

Захист від електромагнітних випромінювань радіочастотного діапазону Засоби та заходи захисту від ЕМ випромінювань радіочастотного діапазону поділяються на індивідуальні та колективні. Останні можна поділити на організаційні, технічні та лікувально-профілактичні.

До організаційних заходів колективного захисту належать: 4 - розміщення об'єктів, які випромінюють ЕМП таким чином, щоб звести до мінімуму можливе опромінення людей;

- "захист часом" - перебування персоналу в зоні дії ЕМП обмежується мінімально необхідним для проведення робіт часом;

- "захист відстанню" - віддалення робочих місць на максимально допустиму відстань від джерел ЕМП;

- "захист кількістю" - потужність джерел випромінювання повинна бути мінімально необхідною; - виділення зон випромінювання ЕМП відповідними знаками безпеки;

- проведення дозиметричного контролю. Технічні засоби колективного захисту передбачають:

- екранування джерел випромінювання ЕМП;

- екранування робочих місць;

- дистанційне керування установками, до складу яких входять джерела ЕМП;

- застосування попереджувальної сигналізації.

До лікувально-профілактичних заходів колективного захисту належать: - попередній та періодичні медогляди;

- надання додаткової оплачуваної відпустки та скорочення тривалості робочої зміни;

- допуск до роботи з джерелами ЕМП осіб, вік яких становить не менше 18 років, а також таких, що не мають протипоказів за станом здоров'я.

Одним із найбільш ефективних технічних засобів захисту від ЕМ випромінювань радіочастотного діапазону, що знаходить широке застосування у промисловості, є екранування. Для екранів використовуються, головним чином, матеріали з великою електричною провідністю (мідь, латунь, алюміній та його сплави, сталь). Екрани виготовляються із металевих листів або сіток у вигляді замкнутих камер, шаф чи кожухів, що під'єднуються до системи заземлення.

Принцип дії захисних екранів базується на поглинанні енергії випромінювання матеріалом з наступним відведенням в землю, а також на відбиванні її від екрана. Захист приміщення від впливу зовнішніх ЕМП можна забезпечити шляхом оклеювання стін металізованими шпалерами та облаштування на вікнах металевих сіток

Таблиця 4 – Параметри електромагнітних неіонізуючих випромінювань

Види поля	Допустимі параметри поля		Допустима поверхнева щільність потоку енергії (інтенсивність потоку енергії), Вт/м^2
	за електричною складовою (Е), В/м	за магнітною складовою (Н), А/м	
Напруженість електромагнітного поля			
60 кГц до 3 мГц	50	5	
3 кГц до 30 мГц	20	-	
30 кГц до 50 мГц	10	0,3	
30 кГц до 300 мГц	5	-	
300 кГц до 300 гГц	-	-	10 Вт/кв. м
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру:			

Кінець таблиці 4

УФ-С (220–280 мм)			0,001
УФ-В (280–320 мм)	-	-	0,01
УФ-А (320–400 мм)			10,0
в видимій частині спектру:			
400–760 мм			10,0
в інфрачервоній частині спектру:			
0,76–10,0 мкм			35,0–70,0
Напруженість електричного поля ВДТ			20кВ/м

На практиці вкрай важливо знати не тільки параметри джерел іонізації, а й параметри аероіонного режиму в приміщенні. Головне питання - якою буде концентрація іонів в робочій зоні, і зокрема, в зоні розташування органів дихання працівника. Одним з підходів до визначення концентрації аероіонів на робочих місцях є застосування лічильників іонів.

Всі існуючі пристрої вимірювання концентрації аероіонів можна класифікувати на спектрометри, реєстратори аероіонів і власне лічильники аероіонів. Для вимірювання концентрації аероіонів в лічильниках аероіонів використовують, як правило, аспіраційний метод або метод відкритого колектора.

Висновок

Під час виконання спеціальної частини з охорони праці було досліджено умови праці людини на підприємствах та в установах, а також роботу при користуванні електронними пристроями.

Правила, викладені в нормативно-правових актах, сьогодні дуже актуальні і важливі, оскільки майже на кожному підприємстві є співробітники, які працюють з електронними пристроями. Коли людина працює за персональним комп'ютером, вона сильно втомлюється, страждає від погіршення зору, нервозності та в деяких випадках психічного здоров'я.

Сьогодні існує багато шкідливих впливів на діяльність людини, таких як шум, вібрація та інші фактори, тому кожен працівник та роботодавець повинен слідувати нормативних вимогам охорони праці, що максимально мінімізувати шкідливий вплив цих факторів на кожного працівника.

Під час діяльності людини на підприємствах необхідно дотримуватись вимог і стандартів захисту працівників в організаціях. Дотримання вимог до працівників та власників підприємств зменшить шкідливий вплив на здоров'я людини.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Порядок проведення медичних оглядів працівників певних категорій. URL: https://zakon.rada.gov.ua/laws/show/z0846-07#Text (дата звернення: 09.06.2022).

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. URL: https://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 07.06.2022).

3. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. URL: https://www.sop.com.ua/article/758-vimogi-shchodo-bezpeki-ta-zahistu-zdorovya-pratsvnikv-pd-chas-roboti-z-ekrannimi-pristroyami (дата звернення: 07.06.2022).

4. Закон України за 25 квітня 2018р №508/31960 про затвердження вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 07.06.2022).

5. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. URL: http://www.ohranatruda.in.ua/pages/5180/ (дата звернення: 07.06.2022).

6. Робота в офісі: основні санітарно-гігієнічні вимоги. URL: https://te.dsp.gov.ua/robota-v-ofisi-osnovni-sanitarno-gigiyenichni-vymogy/ (дата звернення: 06.06.2022).