

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:
«Розробка 3D гри в жанрі аркада на базі Unity»

Виконав студент групи КН-19-2
ЛІТОВЧЕНКО Михайло Андрійович

Керівник роботи:
КОРОТУН Ольга Володимирівна

Рецензент:
СУГОНЯК Інна Іванівна

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **ЛІТОВЧЕНКО Михайло Андрійович**

Керівник роботи: **КОРОТУН Ольга Володимирівна**

Тема роботи: «Розробка 3D гри в жанрі аркада на базі Unity»,

затверджена наказом закладу вищої освіти від «05» січня 2023 р., №10/с

Термін здачі закінченої роботи: «10» червня 2023 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

...

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Аналіз аґалогів програмного продукту. Визначення вимог до додатку. Вибір інструментальних засобів.	16 січня 2023 – 15 лютого 2023	Виконано
2	Проектування та реалізація алгоритмів системи	16 лютого 2023 – 1 березня 2023	Виконано
3	Написання програмного коду	1 березня 2023 – 2 травня 2023	Виконано
4	Тестування системи	2 травня 2023 – 11 травня 2023	Виконано
5	Оформлення пояснювальної записки	11 травня 2023 – 29 травня 2023	
6	Попередній захист	30 травня 2023	Виконано

Здобувач вищої освіти **Михайло ЛІТОВЧЕНКО**
Керівник **Ольга КОРОТУН**

РЕФЕРАТ

Випускна робота бакалавра складається з 3D гри в жанрі аркада та пояснювальної записки. Пояснювальна записка до випускної роботи містить 75 сторінки, 41 ілюстрацій, 8 додатків та 17 використаних джерел.

Метою роботи є розробка 3D гри в жанрі аркада, завдання якої є покращення рефлексів.

В роботі визначено основні завдання на розробку 3D гри, проаналізовано додатки аналоги, визначено основний функціонал та вимоги до системи. Розглянуто популярні аркадні ігри та обрано актуальний жанр гри, спроектовано та реалізовано моделі локацій, автомобілів. Програмний продукт був протестований.

КЛЮЧОВІ СЛОВА: UNITY, АРКАДА, VS CODE.

					КН.КР.Б – 122 – 23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		М.А. Літовченко			Розробка 3D гри в жанрі аркада на базі Unity Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		О.В. Коротун					3	75
Рецензент		І.І. Сугоняк				Житомирська політехніка, група КН-19-2		
Н. контр.								
Зав. каф.		М.С. ГРАФ						

ABSTRACT

The bachelor thesis consists of a 3D game in the arcade genre and an explanatory note. The explanatory note to the final thesis contains 75 pages, 41 illustrations, 8 appendix and 17 used sources.

The purpose of the work is to develop a 3D game in the arcade genre, the task of which is to improve reflexes.

In the work, the main tasks for the development of a 3D game were defined, analog applications were analyzed, the main functionality and system requirements were defined. Popular arcade games were considered, and a relevant game genre was chosen, models of locations and cars were designed and implemented. The software product has been tested.

KEY WORDS: UNITY, ARCADE, VS CODE.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface (інтерфейс програмування додатків)

UI - User Interface (користувацький інтерфейс)

FPS - Frames Per Second (кадри на секунду)

JS – JavaScript мова програмування, призначена для використання в браузері з метою привнесення до веб-сторінки ефектності.;

HTML – HyperText Markup Language «мова гіпертекстової розмітки» — стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері;

VS code – редактор вихідного коду, розроблений Microsoft для Windows, Linux та MacOS.

E-brake - Emergency Brake (екстрений гальмо) - ручне гальмо, що використовується для керування задніми колесами та сприяє дрефту.

Angle - Drift Angle (кут дрефту) - кут між напрямом руху автомобіля і напрямом, в якому спрямовані задні колеса під час дрефту.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ 3D ГРИ НА UNITY	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів популярних аркадних ігор	12
1.3. Вибір архітектури для створення	20
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	29
Висновки до першого розділу	30
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ 3D ГРИ	31
2.1. Визначення варіантів використання системи	31
2.2. Алгоритми роботи системи	32
2.3. Структура компонентів системи	34
2.4. Реалізація динамічної камери	39
Висновки до другого розділу	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З 3D ГРОЮ	44
3.1 Порядок встановлення та налаштування гри	44
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з грою	45
3.3. Тестування роботи 3D гри	51
Висновки до третього розділу	57
ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ	61

ВСТУП

Актуальність теми

Аркадні ігри - це комп'ютерні ігри, що почали з'являтися у 1970-х роках в аркадах (ігрових залах), де гравці могли грати в ігри, кидаючи монети в автомати. У сучасний час, аркадні ігри є популярним видом комп'ютерних ігор, що доступні для гри на комп'ютерах, консолях та мобільних пристроях.

Аркадні ігри є досить популярними ігровим жанром, що має багато шанувальників по всьому світу. Незважаючи на те, що цей жанр виник ще в 70-х роках минулого століття, він не втратив своєї актуальності й досі продовжує бути популярним серед гравців.

Аркадні ігри пропонують прості, але динамічні геймплеї, що дозволяє гравцям швидко зануритися в гру. Багато ігор цього жанру мають легкість і доступність в гравців будь-якого рівня складності, а також вони зазвичай не потребують великих зусиль для їх вивчення.

З ростом популярності онлайн-ігор та мобільних платформ, аркадні ігри стали ще більш актуальними, оскільки вони добре пристосовуються до різних форматів ігор та дозволяють гравцям грати в них на будь-якому пристрої, звідки б вони не знаходилися.

Також варто відзначити, що аркадні ігри мають великий потенціал для інновацій та експериментів в геймдизайні, що дозволяє розробникам створювати нові та унікальні ігрові ідеї та механіки.

Отже, можна сказати, що тема аркадних ігор є досить актуальною та цікавою в сучасному світі геймінгу.

Метою випускної роботи є розробка 3D гри в жанрі аркада на базі Unity. Мета розробки аркадних ігор може бути різною, в залежності від цілей розробника та аудиторії, для якої призначена гра. Основні мети - це розвага, конкуренція, розвиток навичок, пізнавальний аспект, реклама. В даній кваліфікаційній роботі описується випадок розвитку навичок набутих під час навчання.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

Встановлена мета обумовлює наступні завдання, основними етапами постановки завдань при створенні аркадної гри можуть бути:

- визначення цільової аудиторії, наприклад, це може бути гра для дітей, дорослих або професійних гравців;
- визначення концепції гри;
- визначення функцій гри;
- визначення технічних обмежень;
- розробка графічного дизайну та звукового оформлення;
- тестування та оптимізація.

Об'єктом дослідження є розробка 3D гри в жанрі аркада. Такі ігри впливають на людський мозок, когнітивні та емоційні функції, а також механіки гри та їхні взаємовідносини з гравцями.

Предметом дослідження є розробка 3D гри в жанрі аркада на базі Unity. Необхідно описати механіку зазначеної гри, такі як геймплей, рівні складності, правила гри, ігрові механіки, інтерфейс користувача, а також вплив гри на гравців. Дослідження можуть включати аналіз дизайну гри, дослідження впливу гри на емоційний стан гравців, їхню когнітивну та рухову функції, а також вивчення соціальної взаємодії між гравцями у грі.

Також сюди же можна віднести взаємодію між гравцями та різними видами ігрових систем, включаючи механіки досягнення результатів та системи винагород. Дослідження можуть відповісти на питання про те, які фактори впливають на поведінку гравців в грі, які аспекти гри є найбільш привабливими для гравців, і які ігрові механіки можуть бути використані для покращення досвіду гри та задоволення гравців.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ 3D ГРИ НА UNITY

1.1. Постановка задачі

Задача полягає у розробці аркадної гри, яка має бути захоплюючою та цікавою для гравців. Головною метою гри є надання гравцям конкретної цілі, яку вони зможуть досягти через геймплей. Гра також повинна бути достатньо складною, щоб стимулювати гравців до подолання викликів та поліпшення своїх навичок.

Одним із важливих аспектів гри є візуальна графіка, яка повинна бути привабливою та залучати увагу гравців. Гра також потребує відповідного звукового супроводження, щоб створити іммерсивну атмосферу та підкреслити дії гравців.

При створенні гри необхідно визначити жанр гри, наприклад, аркадний дріфт, та розробити відповідну механіку гри та геймплей. Це може включати керування автомобілем, змагання з іншими гравцями або комп'ютером, виконання трюків або завдань на різних рівнях складності.

Гра також може містити різних персонажів або автомобілів, кожен з яких має свої унікальні можливості та характеристики. Наприклад, деякі автомобілі можуть бути швидкими, але менш стійкими, тоді як інші можуть мати кращу контрольованість при дріфті.

Для стимулювання гравців та підвищення інтересу до гри, можна створити систему балів та нагород. Гравці можуть отримувати бали за досягнення певних цілей, успішне виконання завдань або перемоги над іншими гравцями. Нагороди можуть включати відкриття нових рівнів, отримання нових автомобілів або кастомізацію існуючих.

Розробка інтерфейсу гри є ще одним важливим аспектом проекту. Інтерфейс повинен бути інтуїтивно зрозумілим та зручним для користувачів. Він повинен містити необхідну інформацію, таку як показники прогресу, керування, налаштування гри тощо.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Крім того, гра повинна бути розроблена з урахуванням підтримки різних платформ та пристроїв. Ігровий движок Unity 3D, який був використаний у проекті, дозволяє легко перенести гру на різні операційні системи, такі як Windows, macOS, Android, iOS та інші.

Загалом, основна мета полягає у створенні цікавої та захоплюючої гри, яка зможе привернути увагу гравців та забезпечити їм максимальний рівень задоволення від процесу гри. Для досягнення цієї мети необхідно розробити відповідну механіку гри, привабливу візуальну графіку та звукове супроводження, а також систему балів та нагород. Також важливими аспектами є розробка зручного інтерфейсу та підтримка гри на різних платформах та пристроях.

Для досягнення цього мети необхідно вирішити наступні **завдання**:

1. Розробити концепцію гри і визначити її жанр (наприклад, платформер, стрілялка, гонки тощо).
2. Створити геймплей, який буде цікавим та викликає у гравців бажання грати в гру.
3. Розробити привабливу графіку та звуковий супровід.
4. Протестувати гру на різних пристроях і в різних умовах для забезпечення стабільної роботи.
5. Оптимізувати гру для забезпечення максимальної продуктивності та ефективності.
6. Розробити систему керування та інтерфейс гри, які будуть інтуїтивно зрозумілі та прості для гравців.
7. Придумати механіку гри, яка буде допомагати гравцям збирати бонуси та отримувати нові рівні.
8. Підготувати гру до випуску, розробити маркетингову стратегію та провести промо-кампанію для привернення уваги гравців.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів популярних аркадних ігор

Аркадні ігри - це комп'ютерні ігри, які часто мають простий геймплей та вимагають від гравця швидкості реакції та координації рухів. Ось кілька аналогів аркадних ігор:

1. Tetris - гра, в якій гравець повинен складати фігурки з різних блоків, щоб заповнити рядки на екрані. (рис. 1.1)

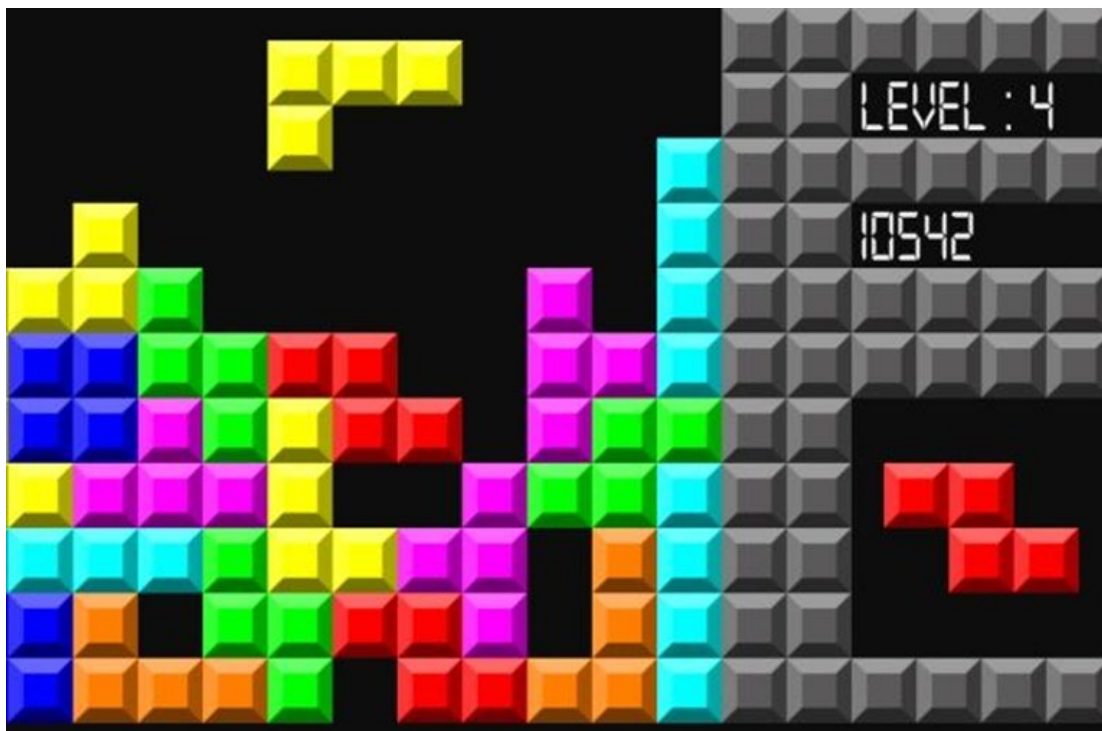


Рис. 1.1 Гра Tetris

2. Вітаю у світі Рас-Ман - найвідомішої аркадної гри усіх часів! У цій захоплюючій пригоді ти береш на себе роль круглого героя, який відправляється в унікальний лабіринт, повний незабутніх пригод та небезпек.

Завдання твого Пак-мена - збирати маленькі, блискучі кульки, розкидані по всьому лабіринту. Здається просто, але це не так просто, як здається! Ворожі персонажі, відомі як "привиди", стрімко полюють на тебе. Твоя винахідливість та швидкість будуть твоїми найкращими друзями в цій небезпечній грі.

Кожен з привидів має свою унікальну особистість та поведінку. Зрозуміти їх рухи та приховані маніпуляції - це ключ до твоєї перемоги. Ти повинен робити все можливе, щоб уникнути їх уваги, бо вони роблять все можливе, щоб спіймати тебе!

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Але не бійся, у лабіринті також розкидані особливі кульки, які роблять тебе тимчасово непомітним для привидів. Використовуй цей шанс, щоб повернутися проти своїх ворогів і навіть з'їдати їх на кілька драгоцінних секунд! Але пам'ятай, що це лише тимчасова безпека, і привиди знову будуть на тебе полювати. (рис. 1.2)



Рис. 1.2 Гра Pac-Man

3. Space Invaders - гра, де гравець керує космічним кораблем, який повинен знищувати прибульців, що наближаються до нього згори екрана. (рис. 1.3)

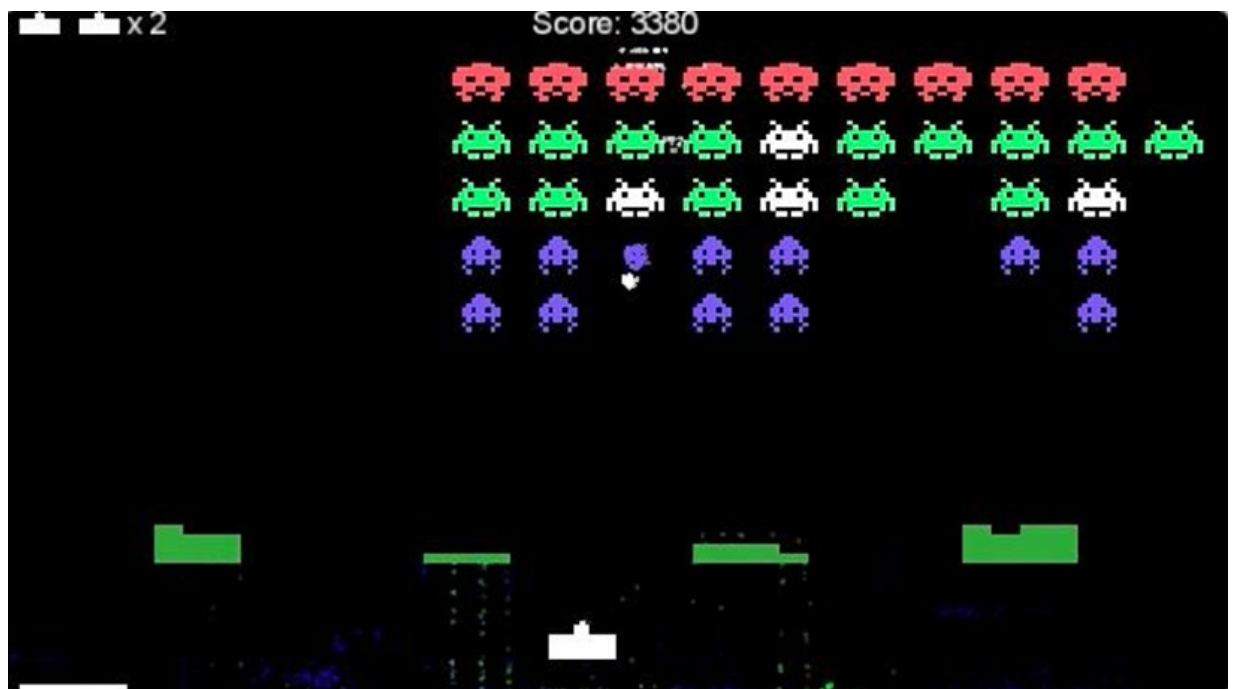


Рис. 1.3 Гра Space Invaders

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

4. Donkey Kong - гра, де гравець керує персонажем на ім'я Маріо, який повинен долати перешкоди, щоб врятувати свою кохану від злого горилли на ім'я Донкі Конг. (рис. 1.4)



Рис. 1.4 Гра Donkey Kong

5. "Among Us" - гра, де гравці приймають ролі членів екіпажу космічного корабля, один з яких є імпостором. Це соціальна дедуктивна гра, де гравці повинні співпрацювати, щоб виявити та вигнати імпостора, або ж імпостор повинен вбити всіх інших гравців, щоб перемогти. (рис. 1.5)

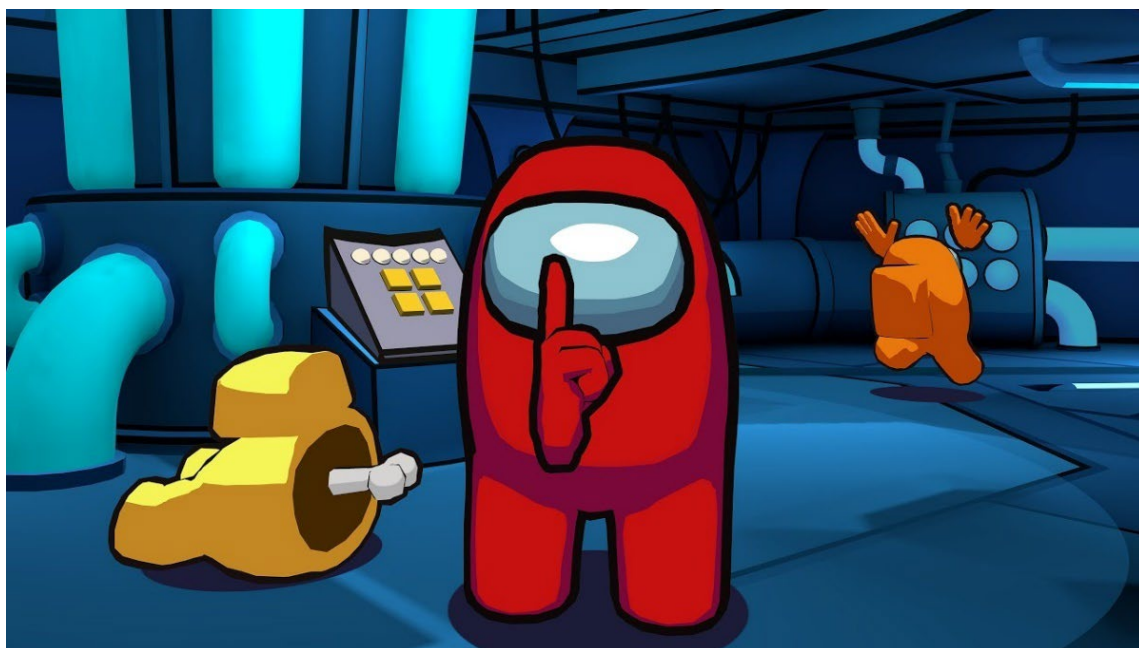


Рис. 1.5 Гра Among Us

6. "Animal Crossing: New Horizons" - симулятор життя, де гравці переїжджають на безлюдний острів і взаємодіють з антропоморфними тваринами. Гра пропонує можливість будувати і прикрашати свій острів, рибалити, збирати ресурси та розвивати своє селище. (рис. 1.6)



Рис. 1.6 Гра Animal Crossing: New Horizons

7. "Minecraft Dungeons" - аркадна гра в стилі "дункельвд" в універсумі Minecraft. Гравці прокладають свій шлях крізь лабіринти, борючись з монстрами, збираючи знаряддя та отримуючи нові навички, щоб перемогти злих босів. (рис. 1.7)

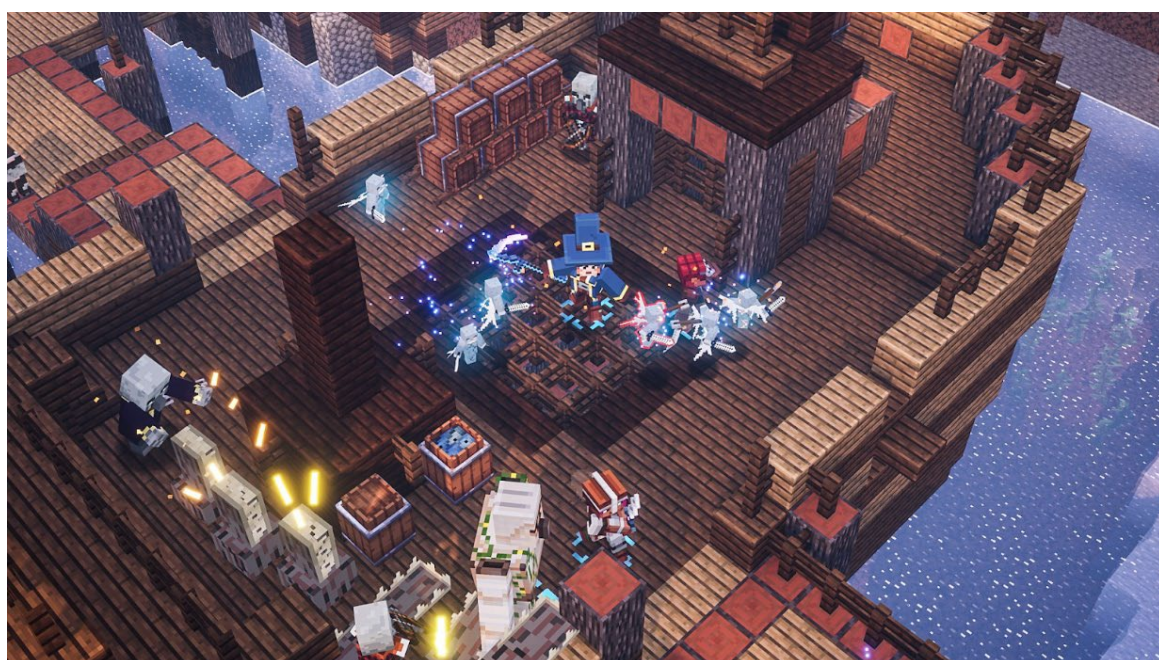


Рис. 1.7 Гра Minecraft Dungeons

8. "Hades" - рогалик, де гравці беруть на себе роль сина Громадянина і мають прокладати шлях через підземелля в пекло, зустрічаючи різні божества та міфічні істоти. Гра поєднує в собі швидкі бойові сцени, розширювання навичок та наративний аспект. (рис. 1.8)

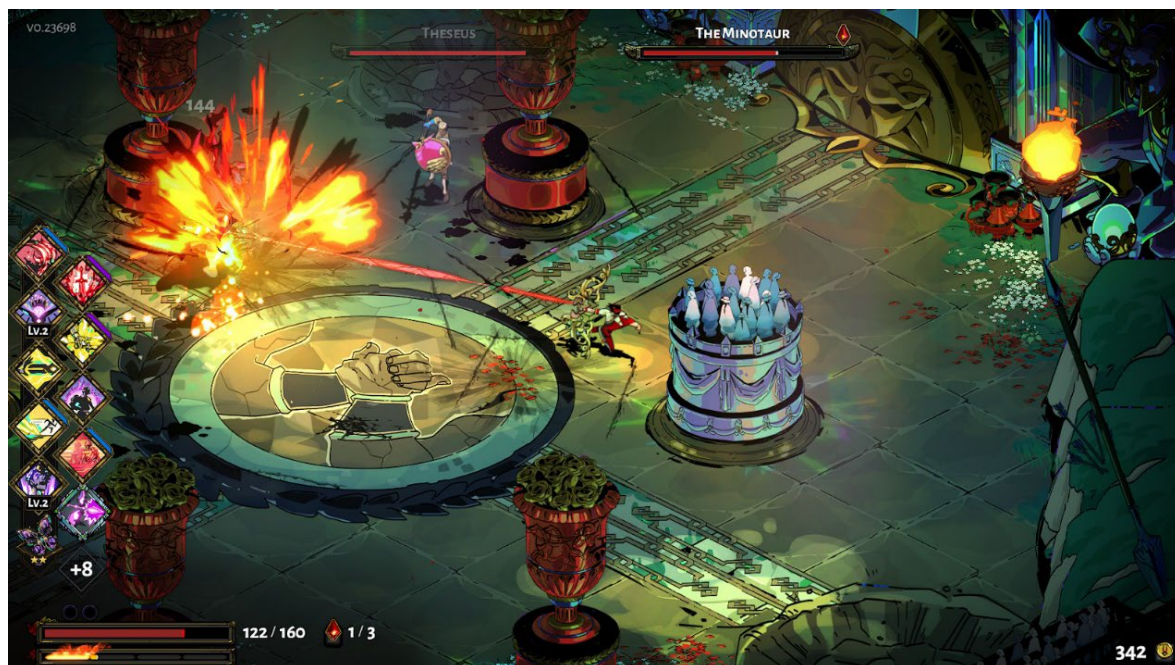


Рис. 1.8 Гра Minecraft Dungeons

9. "Crash Bandicoot 4: It's About Time" - платформер, який є прямим продовженням оригінальної трилогії Crash Bandicoot. Гравцям належить керувати Крешем та його союзниками, подолуючи різні перешкоди та босів, щоб зберегти світ від злих сил. (рис. 1.9)



Рис. 1.9 Гра Crash Bandicoot 4: It's About Time

10. Sonic & All-Stars Racing Transformed: Ця гра, доступна на різних платформах, пропонує гравцям гонятися в якості персонажів з всесвіту Соніка та інших Sega. Гоночні траси можуть трансформуватися з поверхні на повітря та воду, що додає більше різноманіття до геймплею. (рис. 1.10)



Рис. 1.10 Гра Sonic & All-Stars Racing Transformed

11. Crash Team Racing Nitro-Fueled: Ця гра, доступна на різних платформах, включається в список популярних гоночних аркадних ігор. Вона є

переосмисленням оригінальної гри Crash Team Racing з покращеною графікою та багатим вибором гоночних трас, персонажів та режимів гри. (рис. 1.11)



Рис. 1.11 Гра Crash Team Racing Nitro-Fueled

12. Burnout Paradise Remastered: Ця гра доступна на різних платформах, включаючи PlayStation, Xbox і ПК. Вона пропонує відкритий світ, де гравці можуть свободно ганятися по місту, виконувати руйнівні трюки та брати участь в гоночних подіях. (рис. 1.12)



Рис. 1.12 Гра Burnout Paradise Remastered

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

13. Forza Horizon 4: Це гоночна аркадна гра, доступна на Xbox і ПК. Гравці можуть насолоджуватися відкритим світом з відтворенням Великобританії, де вони змагаються в різних типах гонок, виконують трюки та виконують завдання. (рис. 1.13)



Рис. 1.13 Гра Forza Horizon 4

14. Mario Kart 8 Deluxe: Це гра на Nintendo Switch, в якій гравці змагаються на швидких трасах у світі Маріо. Вона пропонує різноманітні гоночні траси, виклики та режими гри, а також можливість використовувати різні зброї та предмети для переваги над суперниками. (рис.1.14)

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.14 Гра Mario Kart 8 Deluxe

15. Ласкаво просимо у світ адреналіну та швидкості, де нічні вулиці стають полем битви для найбільш безкомпромісних та захоплюючих гоночних змагань. Need for Speed: Underground - це яскравий представник аркадних гоночних ігор, який вразить тебе своєю інтенсивністю та незабутніми емоціями.

Need for Speed: Underground віддає перевагу кастомізації автомобілів. Ти зможеш створити свій унікальний болід, який відображатиме твій стиль та особистість. Від зовнішнього вигляду, змінюючи кузов, обвіси, фари та кольори, до технічних характеристик, покращуючи двигун, підвіску та гальмівну систему - можливості безмежні. Ти відчуєш себе справжнім автомобільним дизайнером і механіком, який творить шедеври на колесах. (рис. 1.15)



Рис. 1.15 Гра Need for Speed: Underground

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

Ці ігри можуть бути знайдені на багатьох платформах, таких як консолі, комп'ютери та мобільні пристрої.

1.3. Вибір архітектури для створення

Для створення аркадних ігор можна використовувати різні архітектури, які залежать від специфіки самої гри та технічних можливостей розробника. Архітектура гри визначає структуру програмного забезпечення, його компоненти та взаємодію між ними.

Одним з поширених підходів є архітектура "**Component-Based**". В цьому підході гра будується з використанням компонентів, які представляють окремі функціональні частини. Кожен компонент відповідає за певний аспект гри, такий як рух персонажа, взаємодія з об'єктами, штучний інтелект тощо. Ці компоненти можуть бути повторно використані в різних частинах гри, що спрощує розробку та розширення.

Іншим підходом є архітектура "**Event-Driven**". В цьому випадку гра базується на взаємодії подій. Компоненти гри відправляють та слухають події, і реагують на них відповідно. Наприклад, коли гравець натискає кнопку стріляння, гра відправляє подію "стріляння", і відповідний компонент обробляє цю подію для виконання відповідної дії.

Ще однією архітектурою, яку можна використовувати для аркадних ігор, є "**Game Loop**". Цей підхід використовує цикл оновлення гри, де на кожній ітерації оновлюються стани об'єктів, обробляються введення гравця та відображається графіка. Цей цикл безперервно повторюється протягом гри, забезпечуючи її плавну роботу та відгук на дії гравця.

Крім цього, в залежності від масштабу та складності гри, можуть використовуватися інші архітектурні підходи, такі як "**Entity-Component-System**" (ECS), "**Model-View-Controller**" (MVC) тощо. Вибір архітектури

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

залежить від конкретних потреб проекту та зручності для розробки та підтримки гри.

Важливо враховувати, що архітектура гри повинна бути гнучкою та масштабованою, щоб забезпечити легкість утримання та розширення функціональності. Також варто враховувати вимоги до продуктивності та оптимізації, особливо для аркадних ігор, які часто вимагають швидкої реакції та плавного геймплею.

Ось кілька популярних архітектур, які використовуються для створення аркадних ігор:

Unity - це потужний інтегрований середовище розробки ігор, яке надає розробникам широкі можливості для створення захоплюючих ігрових досвідів. Основні особливості Unity включають:

Мультиплатформна розробка: **Unity** дозволяє розробникам створювати ігри для різних платформ, включаючи ПК, консолі, мобільні пристрої, віртуальну та доповнену реальність. Це дає змогу зайняти широку аудиторію гравців і забезпечити кроссплатформову сумісність.

Високоякісна графіка та візуалізація: **Unity** надає потужні інструменти для створення реалістичних світів. Завдяки різноманітним шейдерам, текстурам, освітленню, ефектам частинок та пост-процесингу, розробники можуть досягти вражаючої графіки та візуальних ефектів у своїх іграх.

Розширені інструменти редагування: **Unity** має вбудований редактор, який дозволяє розробникам змінювати та налаштовувати світи, персонажів, анімацію та інші елементи гри. Редактор також підтримує широкий спектр імпорту ресурсів і дозволяє візуально створювати сцени та рівні гри.

Фізика та зіткнення: **Unity** має розвинену систему фізики, яка дозволяє створювати реалістичні поведінки об'єктів в грі. Розробники можуть

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

встановлювати фізичні властивості об'єктів, моделювати зіткнення, сили та рухи, що додає реалізму ігровому середовищу.

Можливості програмування: **Unity** має вбудовану підтримку різних мов програмування, зокрема **C#, JavaScript** та **Boo**. Це дає розробникам велику свободу вибору технології для реалізації логіки гри та інтерактивності.

Велика спільнота розробників: **Unity** має широку спільноту розробників, яка активно співпрацює і допомагає один одному з розв'язанням проблем, надає поради та підтримку. Існує багато форумів, блогів, посібників та відеоуроків, що полегшують навчання та розвиток у галузі розробки ігор з використанням **Unity**. Додатково до основних особливостей, **Unity** також має інші цікаві фішки, такі як:

Вбудована система анімації, що дозволяє створювати складні рухи та анімації персонажів.

Інтеграція зі сторонніми програмами та сервісами, які допомагають забезпечити аналітику, рекламу, соціальну ігрову мережу та інші функціональності.

Можливість розробки віртуальної та доповненої реальності, що дозволяє створювати іммерсивні ігрові досвіди для **VR** та **AR** пристроїв.

Інструменти для створення мережевої гри та мультиплеєра, що дозволяють розробникам створювати онлайн-ігри зі змаганнями або спільною грою.

Загалом, **Unity** надає розробникам широкий функціонал та потужні інструменти для створення різноманітних ігрових проєктів, забезпечуючи їм креативну свободу та можливість зробити гру захопливою та вражаючою для гравців.

Unity використовуються відомими компаніями, такими як **Electronic Arts, Microsoft, Ubisoft, Blizzard Entertainment** та інші. **Unity** також

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

використовується для розробки навчальних програм та симуляцій у різних галузях, включаючи медицину, авіацію, архітектуру та інші (рис. 1.6).

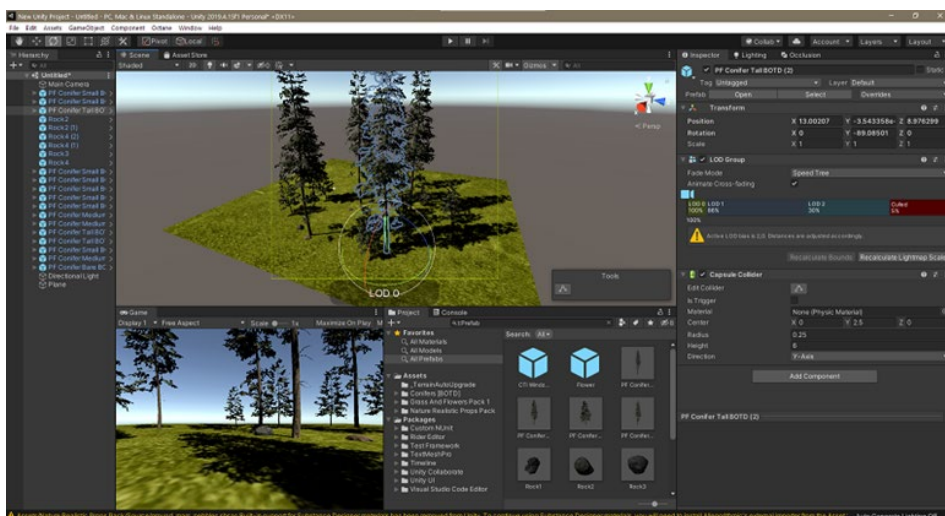


Рис. 1.16 Ігровий движок Unity

Unreal Engine - це потужний інструментарій для розробки ігор, симуляцій та візуалізацій, створений компанією **Epic Games**. Цей ігровий двигун володіє великою популярністю та широким спектром функціональності. Основні особливості **Unreal Engine** включають:

Високоякісна графіка та візуалізація: **Unreal Engine** володіє потужними інструментами для створення захоплюючих і вражаючих графічних світів. Двигун підтримує фотореалістичну графіку, динамічне освітлення, тіні та ефекти частинок, що дозволяє розробникам створювати вражаючі візуальні ефекти.

Розширені інструменти редагування: **Unreal Engine** надає розробникам вбудований редактор, який дозволяє змінювати та налаштовувати світи, персонажів, анімацію та інші елементи гри. Редактор дозволяє візуально створювати складні сцени та мапи, а також налаштовувати логіку гри.

Фізика та зіткнення: **Unreal Engine** пропонує розвинену систему фізики та зіткнень, що дозволяє створювати реалістичні поведінки об'єктів у грі. Розробники можуть встановлювати фізичні властивості об'єктів, моделювати реалістичні зіткнення та фізичні ефекти.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

Мультиплатформний розробка: Unreal Engine дозволяє розробникам створювати ігри для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Це дає можливість досягти широкої аудиторії гравців та забезпечити кроссплатформову сумісність.

Велика спільнота розробників: **Unreal Engine** має активну та дружню спільноту розробників, яка надає підтримку, допомогу та обмін досвідом. Існує багато ресурсів, таких як форуми, блоги, вебінари та відеоуроки, де розробники можуть отримати поради та вирішити проблеми.

Поза основними особливостями, **Unreal Engine** також має додаткові цікаві фішки, такі як:

Blueprints: **Unreal Engine** пропонує систему **Blueprints**, яка дозволяє розробникам створювати складну логіку гри без необхідності програмування. Це візуальний інтерфейс, який дозволяє створювати скрипти та взаємодію об'єктів за допомогою графічних блоків.

Marketplace: У **Unreal Engine** є вбудований **Marketplace**, де розробники можуть придбати готові активи, такі як моделі персонажів, текстури, звукові ефекти тощо. Це допомагає економити час та ресурси при розробці гри.

Live Link: **Unreal Engine** підтримує функцію **Live Link**, яка дозволяє розробникам підключати рухові захоплення та інші зовнішні пристрої, щоб створювати реалістичну анімацію персонажів.

Завдяки своїм особливостям та функціональності, **Unreal Engine** є популярним вибором для розробників, які прагнуть створити вражаючі, графічно захоплюючі та високоякісні ігрові проекти.

Unreal Engine використовуються відомими компаніями, такими як **Epic Games, Ubisoft, Electronic Arts, Square Enix** та інші (рис. 1.7).

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.17 Ігровий движок Unreal Engine

Phaser: Phaser - це безкоштовний ігровий двигун з відкритим вихідним кодом, що дозволяє розробляти ігри для веб-браузерів та мобільних пристроїв. Phaser створений на базі **HTML5**, **JavaScript** та **Canvas** технологій.

Phaser - це безкоштовний ігровий двигун з відкритим вихідним кодом, спеціально розроблений для створення ігор, які працюють у веб-браузерах та на мобільних пристроях. Цей двигун базується на HTML5, JavaScript та Canvas технологіях, що забезпечує високу доступність та переносимість проектів.

Перерахуємо основні особливості Phaser.

Різноманітність типів ігор: **Phaser** надає підтримку для різних жанрів ігор, включаючи платформери, стрілянини, головоломки, ігри зі змінним видом та багато інших. Це дозволяє розробникам втілювати свої творчі ідеї у різноманітних проектах.

Можливості анімації: **Phaser** надає потужні інструменти для створення анімаційних ефектів у грі. Це включає підтримку кадрів анімації, спрайтів (графічних об'єктів) та таймлайнів, що дає можливість створювати живі та динамічні сцени.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

Фізика та зіткнення: **Phaser** має вбудовану систему фізики, що дозволяє реалістично моделювати поведінку об'єктів у грі. Розробники можуть встановлювати фізичні властивості об'єктів, забезпечувати правильні зіткнення та фізичні реакції.

Легкість використання: **Phaser** має дружній та зрозумілий інтерфейс користувача, що дозволяє легко почати розробку ігор. Існує документація, приклади коду та навчальні ресурси, які сприяють швидкому освоєнню двигуна.

Підтримка мобільних пристроїв: **Phaser** надає підтримку для мобільних пристроїв і може працювати на різних платформах, включаючи iOS та Android. Це дозволяє розробникам створювати ігри, які можна грати на смартфонах і планшетах.

Велика спільнота розробників: **Phaser** має активну та допоміжну спільноту розробників, яка надає підтримку та спільний досвід. Існують форуми, блоги, соціальні медіа групи та онлайн-комунітети, де можна обмінюватись ідеями, задавати питання та отримувати відповіді.

Додаткові цікаві фішки **Phaser**.

Таймери та анімаційні події: **Phaser** має вбудовану систему таймерів та анімаційних подій, яка дозволяє контролювати хронологію та послідовність подій у грі.

Аудіо підтримка: **Phaser** надає можливість відтворювати звукові ефекти та музику в грі. Розробники можуть налаштовувати гучність, змішування аудіо та застосовувати ефекти звуку.

Розширення та плагіни: **Phaser** має розширювану архітектуру, яка дозволяє розробникам створювати власні плагіни та розширення для додаткового функціоналу.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Phaser є популярним вибором для розробників, які бажають створювати ігри у веб-середовищі з високою якістю та ефективністю. Його гнучкість, доступність та підтримка спільноти роблять його ідеальним варіантом для початківців та досвідчених розробників ігор.

Phaser використовуються розробники з усього світу, щоб створювати різні типи ігор для веб-браузерів та мобільних пристроїв. **Phaser** також використовується в навчальних програмах та для створення інтерактивних веб-сайтів (рис. 1.8).



Рис. 1.18 Ігровий движок Phaser

Construct - це безкоштовний візуальний ігровий двигун, розроблений з метою спростити процес створення **2D-ігор** для веб-браузерів та мобільних пристроїв. Його основна перевага полягає в можливості створення ігор без необхідності програмування, завдяки візуальному інтерфейсу та логічним блокам.

Основні особливості **Construct** включають наступне.

Візуальне програмування: **Construct** надає візуальний інтерфейс для створення ігор шляхом використання логічних блоків. Розробники можуть використовувати графічний редактор для створення об'єктів, додавання поведінок та встановлення взаємодії між ними без потреби в кодуванні.

Шаблони ігор: **Construct** постачається з більш ніж 100 готовими шаблонами ігор, які різноманітні за жанрами та механіками. Це дозволяє швидко почати роботу над грою, вибравши відповідний шаблон та внесення необхідних змін.

Вбудовані ефекти: **Construct** має вбудовану систему ефектів, що спрощує додавання візуальних покращень до ігор. Завдяки цьому, розробники можуть швидко застосовувати ефекти, такі як тіні, блиск, розмиття та багато інших, без необхідності у глибокому розумінні графічного програмування.

Підтримка мобільних пристроїв: **Construct** працює на різних платформах, включаючи **iOS** та **Android**, що дозволяє розробникам створювати ігри, які можна запускати на мобільних пристроях. Це забезпечує гнучкість та доступність для аудиторії, яка грає на смартфонах та планшетах.

Легкість використання: **Construct** пропонує інтуїтивно зрозумілий інтерфейс користувача, що спрощує процес розробки ігор. Новачки без попереднього досвіду у програмуванні можуть швидко освоїти інструменти та почати створювати свої власні ігри.

Підтримка інтернету: **Construct** дозволяє легко створювати ігри з мультиплеєром та взаємодією з інтернетом. Розробники можуть реалізовувати онлайн-функціонал, такий як спільна гра з іншими гравцями, збереження прогресу в хмарних сервісах та обмін даними через мережу.

Construct є привабливим вибором для тих, хто бажає швидко створити **2D-ігри** без програмування. Він дозволяє розробникам зосередитись на творчому процесі та швидко реалізувати свої ідеї в грі, використовуючи візуальні інструменти та готові шаблони ігор.

Construct використовуються розробники з усього світу, щоб створювати різні типи ігор для веб-браузерів та мобільних пристроїв. Також особливістю

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

Construct можна вважати те, що він використовується в навчальних програмах та для створення інтерактивних веб-сайтів.(рис. 1.9)

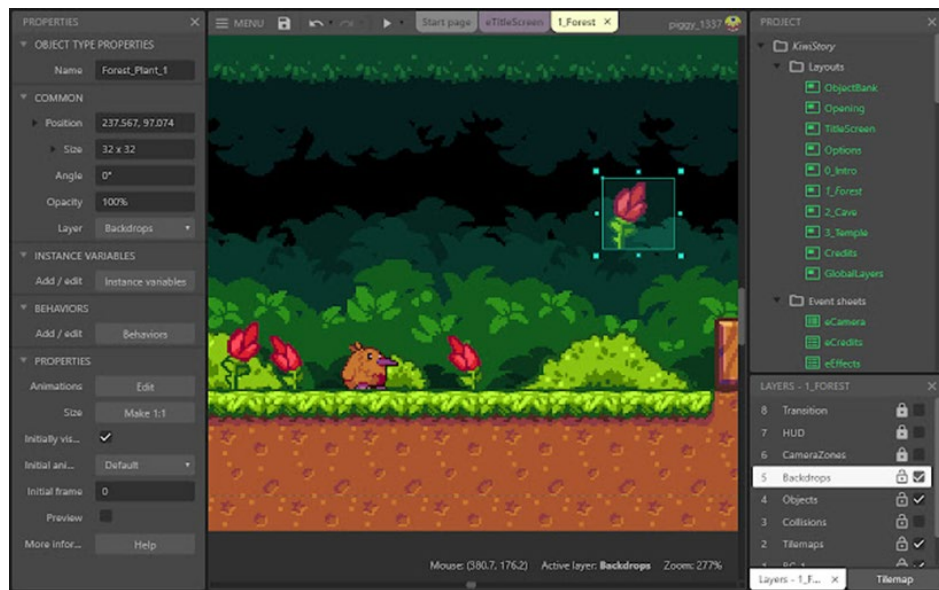


Рис. 1.19 Ігровий двигун Construct

Ці архітектури дозволяють розробникам створювати аркадні ігри з різним рівнем складності та використовувати різні технології для досягнення бажаних результатів.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки гри обрано мову програмування C#, а саме фреймворк VS Code.

Visual Studio Code — редактор коду від компанії Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб- та хмарних додатків. Має ряд переваг.

Розширюваність: VS Code має потужну систему розширень, що дозволяє розширити його функціональність для відповіді на різні потреби розробників. Можна встановлювати розширення для підтримки різних мов програмування, фреймворків, інструментів версіонування коду, відладки та багато іншого.

Легкість використання: VS Code має інтуїтивний інтерфейс та просту навігацію. Він швидко запускається та потребує мінімальних вимог до

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		29

системних ресурсів. Інтеграція з Git також дуже зручна, що дозволяє ефективно працювати з версіонуванням коду.

Підтримка різних мов програмування: VS Code підтримує багато мов програмування, включаючи популярні мови, такі як JavaScript, Python, Java, C#, PHP і багато інших. Він надає підказки, автодоповнення, перевірку синтаксису та багато іншого для полегшення розробки.

Відладка: VS Code має вбудовану підтримку відлагодження, яка дозволяє налагоджувати код прямо у редакторі. Можна ставити точки останова, відстежувати значення змінних та крокувати через код для виявлення та виправлення помилок.

Комунітет та екосистема: VS Code має велику та активну спільноту розробників. Це означає, що розробник може легко знайти допомогу, документацію, підручники та розширення, які додатково полегшують роботу з редактором. Крім того, популярні фреймворки та інструмент

Висновки до першого розділу

В першому розділі було поставлено технічне завдання, проаналізовані існуючі аналоги, такі як: Tetris, Pac-Man, Space Invaders, Galaga, Donkey Kong. На основі аналізу було зроблено висновки, які аркадні ігри є найбільш популярними.

Для реалізації було вирішено використати такі інструментальні засоби: VS Code, ігровий движок Unity, мову стилів C#, наведено їх короткий опис.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ 3D ГРИ

2.1. Визначення варіантів використання системи

Діаграма варіантів використання (Use Case Diagram) - це графічний інструмент моделювання, який демонструє функціональність системи та взаємодію акторів (користувачів або зовнішніх систем) з цією системою. Вона дозволяє визначити основні варіанти використання системи та їх залежності.

Основними елементами діаграми варіантів використання є актори і варіанти використання.

Діаграма варіантів використання (Use Case Diagram) є графічним інструментом, який дозволяє моделювати взаємодію між акторами і системою в рамках конкретного програмного продукту або системи. Основними елементами діаграми варіантів використання є актори і варіанти використання.

Актори представляють ролі або зовнішні сутності, які взаємодіють з системою. Актори можуть бути реальними користувачами, іншими системами, зовнішніми сервісами або навіть іншими частинами системи. Вони зображуються у вигляді піктограм людей або інших об'єктів, що відповідають їх ролі або типу.

Варіанти використання, також відомі як "сценарії", описують окремі функціональні можливості або сценарії взаємодії між акторами та системою. Кожен варіант використання має назву, яка конкретизує дію або функціональність, яку він представляє. Вони допомагають уявити, як система повинна відповідати на взаємодію з акторами.

Діаграма варіантів використання дає загальний огляд взаємодії між акторами і системою, і може бути використана як засіб комунікації між розробниками, дизайнерами та зацікавленими сторонами проекту. Вона допомагає визначити функціональні вимоги до системи та встановити контекст, в якому вона працює.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграма варіантів використання може бути використана на ранніх етапах розробки для уточнення вимог до системи, а також на пізніших етапах для валідації функціональності та перевірки, чи враховано всі потреби акторів.

Загалом, діаграма варіантів використання є потужним інструментом для моделювання взаємодії між акторами і системою, що допомагає зрозуміти функціональність системи та встановити правильний контекст розробки.

Діаграма варіантів використання допомагає визначити потреби користувачів та основні функції системи. Вона також використовується для комунікації між розробниками та зацікавленими сторонами для уточнення вимог до системи. (рис. 2.1)

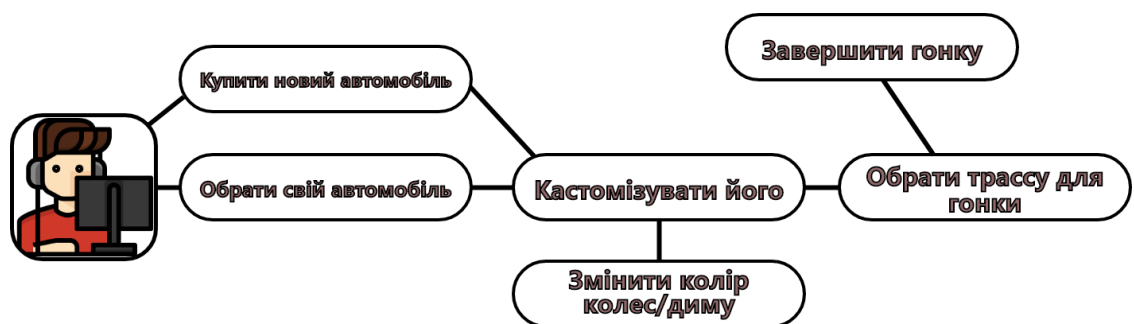


Рис. 2.1 Діаграма варіантів використання

У цьому прикладі гравець та опоненти є акторами, які взаємодіють з системою гри. Варіанти використання описують основні дії, які може виконувати користувач, такі як грати гоночний забіг, кастомізувати автомобіль, вибирати траси та інші.

2.2. Алгоритми роботи системи

При розробці структури файлів системи для аркадної 3D гри про дріфт можна використовувати наступну організацію:

Папка **"Assets"**: Це основна папка, в якій зберігаються всі активи гри, такі як моделі, текстури, звуки, анімації та інші ресурси.

Ця папка може бути поділена на декілька підпапок для кращої організації, наприклад:

"Models" - зберігаються 3D моделі автомобілів, оточення та інші об'єкти.

"Textures" можна розмістити текстури для моделей та інші графічні ресурси.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

"**Audio**" - додаються звукові ефекти, музика та діалогові файли.

"**Animations**" можна зберігати анімації персонажів, автомобілів та інших об'єктів.

Папка "**Scripts**" - місце, де зберігаються скрипти, написані на мові програмування, такі як **C#** або **JavaScript**. Ці скрипти використовуються для реалізації логіки гри, управління персонажем, фізики, штучного інтелекту та інших аспектів.

Папка "**Scenes**" - знаходяться файли сцен, які визначають окремі рівні або локації гри. Кожна сцена містить інформацію про розташування об'єктів, освітлення, камеру та інші параметри.

Папка "**UI**" або "**GUI**" - знаходяться файли, пов'язані з інтерфейсом користувача, такі як меню, кнопки, інформаційні панелі тощо. Ці файли можуть бути збережені у вигляді префабів (prefabs) для зручності використання.

Папка "**Scripts/Managers**" можна розмістити скрипти-менеджери, які керують різними аспектами гри, наприклад, управлінням рівнями, аудіо, ресурсами та іншими глобальними функціями.

Папка "**Scripts/Controllers**" - скрипти-контролери, які відповідають за управління персонажем, автомобілем або іншими об'єктами гри.

Папка "**Settings**" можна зберігати файли налаштувань гри, такі як налаштування графіки, звуку, управління тощо.

Папка "**Documentation**" - документація про гру, включаючи довідкові файли, описи ресурсів та іншу важливу інформацію.

Варто зазначити, що це лише загальна структура, і вона може бути змінена або розширена залежно від потреб проекту та особливостей розробки, але рекомендується дотримуватися організованої структури файлів, щоб забезпечити легку навігацію та підтримку проекту.

2.3. Структура компонентів системи

Оскільки розроблена гра є аркадними гонками доцільно показати керування машиною та описати, де за що відповідає.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

На рис. 2.2 продемонстровано як відбувається налаштування колес.

```
public CarWheels carWheels;

[System.Serializable]
public class CarWheels
{
    public ConnectWheel wheels;
    public WheelSetting setting;
}

[System.Serializable]
public class ConnectWheel
{
    public bool frontWheelDrive = true;
    public Transform frontRight;
    public Transform frontLeft;

    public bool backWheelDrive = true;
    public Transform backRight;
    public Transform backLeft;
}

[System.Serializable]
public class WheelSetting
{
    public float Radius = 0.4f;
    public float Weight = 1000.0f;
    public float Distance = 0.2f;
}
```

Рис. 2.2 Налаштування колес

Публічне поле "carWheels" типу "CarWheels" - це поле представляє весь набір коліс автомобіля і використовується для доступу до них з інших класів.

Клас "CarWheels" - клас є серіалізованим (підтримує збереження та завантаження в редакторі Unity) і містить два поля: "wheels" (типу "ConnectWheel") і "setting" (типу "WheelSetting").

Клас "ConnectWheel" - це клас є серіалізованим і вміщує інформацію про кожне колесо автомобіля. Він складається з наступних полів:

"frontWheelDrive": Логічна змінна, яка вказує, чи є передні колеса приводом автомобіля.

"frontRight" і "frontLeft": Об'єкти Transform, які представляють праве та ліве переднє колесо відповідно.

"backWheelDrive": Логічна змінна, яка вказує, чи є задні колеса приводом автомобіля.

"backRight" і "backLeft": Об'єкти Transform, які представляють праве та ліве заднє колесо відповідно.

Клас "WheelSetting" - клас також серіалізований і містить налаштування для кожного колеса автомобіля. Він містить такі поля:

"Radius" - радіус колеса.

"Weight" - вага колеса.

"Distance" - відстань між передніми та задніми колесами.

```
1 using UnityEngine;
2 using System.Collections;
3 using System.Collections.Generic;
4
5 public class AIVehicle : MonoBehaviour
6 {
7
8     public float carRestTime = 5.0f;
9
10    public float forwardSpeed = 1.0f;
11    public float steerSpeed = 1.0f;
12    public float nextNodeDistance = 5;
13
14    public Transform frontPoint;
15
16    [HideInInspector]
17    public Transform currentNode, nextNode, wrongNode;
18    [HideInInspector]
19    public int currentLap = 0;
20
21    [HideInInspector]
22    public float playerCurrentTime, playerBestTime, playerLastTime = 0.0f;
23    [HideInInspector]
24    public float AIAccel, AISteer = 0.0f;
25    [HideInInspector]
26    public bool AIBrake = false;
27
28    private VehicleControl vehicleControl;
29
30    private bool goNextNode = true;
31    private bool getLap = false;
32
33    private int carPreviousNodes = 0;
34    private float targetAngle;
35    private float restTimer = 0.0f;
36
37    void Start()
38    {
39        restTimer = carRestTime;
40        currentLap = 0;
41        vehicleControl = transform.GetComponent<VehicleControl>();
42        currentNode = nextNode;
43    }
44
45    void Update()
46    {
47        if (GameUI.manage.gameStarted)
48            playerCurrentTime += Time.deltaTime;
```

Рис. 2.3 Скрипт для керування штучного інтелекту автомобіля в грі

Ця частина коду є скриптом який існує для керування штучного інтелекту автомобіля в грі.

Ось деякі з основних функціональностей, що виконує цей код.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Визначення змінних: Код містить декілька змінних, таких як швидкість руху вперед (**forwardSpeed**), швидкість повороту (**steerSpeed**), відстань до наступної точки (**nextNodeDistance**) та інші, які контролюють поведінку автомобіля.

Ініціалізація: в методі **Start()** виконується початкова налаштування автомобіля, встановлення початкових значень змінних і отримання початкової точки (**currentNode**).

```
[System.Serializable]
public class MenuGUI
{
    public Text GameScore;
    public Text VehicleName;
    public Text VehiclePrice;

    public Slider VehicleSpeed;
    public Slider VehicleBraking;
    public Slider VehicleNitro;

    public Slider sensitivity;

    public Toggle audio;
    public Toggle music;
    public Toggle vibrateToggle;
    public Toggle ButtonMode, AccelMode;

    public Image wheelColor, smokeColor;
    public Image loadingBar;

    public GameObject loading;
    public GameObject customizeVehicle;
    public GameObject buyNewVehicle;
}

[System.Serializable]
public class MenuPanels
{
    public GameObject MainMenu;
    public GameObject SelectVehicle;
    public GameObject SelectLevel;
    public GameObject EnoughMoney;
    public GameObject Settings;
}
```

Рис. 2.4 Частина коду яка відповідає за функціонал головного меню

Оновлення: в методі **Update()** код виконується на кожному кадрі гри. Він відслідковує час гравця (**playerCurrentTime**) і виконує функцію **AICarControl()** для керування автомобілем.

Функція **AICarControl()** - функція містить основну логіку керування автомобілем штучного інтелекту. Вона відповідає за рух автомобіля, його повороти, швидкість і зупинку.

Пропонуємо розглянути досить цікаву частину коду, яка відповідає за функціонал головного меню, що буде розглянуто згодом. Цей код представляє два класи: **MenuGUI** і **MenuPanels**. Опишемо кожен з них окремо.

Клас **MenuGUI**.

Text GameScore, **Text VehicleName** та **Text VehiclePrice** - це поля для відображення тексту. Вони використовуються для відображення інформації про гру, назви транспортного засобу та його ціни.

Slider VehicleSpeed, **Slider VehicleBraking** та **Slider VehicleNitro** - це ползунки, що використовуються для відображення значень швидкості, гальмування та рівня нітро для транспортного засобу.

Slider sensitivity - це ползунок, що використовується для налаштування чутливості.

Toggle audio, **Toggle music** та **Toggle vibrateToggle** - це перемикачі, що використовуються для управління звуком, музикою та вібрацією в грі.

Toggle ButtonMode, **AccelMode** - це перемикачі, які використовуються для вибору режиму керування, наприклад, кнопочового або акселерометрового.

Image wheelColor та **Image smokeColor** - це зображення, які можуть використовуватися для відображення кольору коліс та диму в грі.

Image loadingBar - це зображення, яке може використовуватися для відображення прогресу завантаження.

GameObject loading, **GameObject customizeVehicle** та **GameObject buyNewVehicle** - це гральні об'єкти (game objects), які використовуються для різних елементів головного меню, таких як екран завантаження, налаштування транспортного засобу та екран покупки нового транспортного засобу.

Клас **MenuPanels**.

GameObject MainMenu, **GameObject SelectVehicle**, **GameObject SelectLevel**, **GameObject EnoughMoney** та **GameObject Settings** - це гральні

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

об'єкти, які використовуються для різних екранів та панелей меню гри, таких як головне меню, вибір транспортного засобу, вибір рівня, екран недостатньо коштів і налаштування.

Загалом, цей код представляє собою класи, що містять публічні поля для доступу до різноманітних елементів інтерфейсу користувача (UI) та гральних об'єктів, що використовуються в меню гри.

2.4. Реалізація динамічної камери

Камера в грі відіграє важливу роль, оскільки вона забезпечує гравцю зорову перспективу та взаємодію з гральним світом. Вона відповідає за створення візуального представлення грального середовища, показує гравцеві те, що відбувається навколо нього і дозволяє керувати поглядом гравця.

Налаштування камери в грі є важливим аспектом розробки, оскільки воно впливає на загальний досвід гравця. Деякі основні аспекти налаштування камери включають:

Позиція: Камера повинна бути розташована відповідно до потреб гри. Вона може бути розташована зі спостережної точки зверху, ззаду або з перспективи гравця. Вірно підібрана позиція камери дозволяє гравцеві зручно спостерігати за гральним середовищем і отримувати необхідну інформацію.

Кут огляду: Кут огляду камери визначає, яку частину грального світу вона показує гравцю. Це може бути широкий кут огляду для показу більшої області або вузький кут для деталізованого перегляду. Важливо забезпечити достатньо широкий огляд для гравця, але при цьому не допустити зайвого перекриття графіки або втрати деталей.

Зум: Зум камери визначає ступінь збільшення або зменшення візуального масштабу грального світу. Зум може використовуватися для показу деталей або для покращення зручності гравця. Правильно налаштований зум дозволяє гравцеві краще бачити деталі гри або концентруватися на певних об'єктах.

Рух камери: Деякі гри використовують рух камери для створення ефекту реалізму або для зміни перспективи гравця. Це може включати панораму,

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

повороти або зміщення камери. Рух камери може бути синхронізований з рухом гравця або контролюватися окремо.

Ефекти та фільтри: Камера може використовувати різні ефекти та фільтри для поліпшення візуального враження гри. Це можуть бути ефекти освітлення, туману, кольірних фільтрів або пост-процесингові ефекти. Вони додають атмосферності і настрою до грального світу.

Налаштування камери в грі вимагає балансу між *функціональністю* та *естетикою*. Вона повинна забезпечувати комфортну гру і забезпечувати необхідну інформацію гравцю. Правильно налаштована камера допомагає створити занурення в гру та забезпечує більш задоволений гравець досвід.

Тому варто показати та описати її (рис. 2.3).

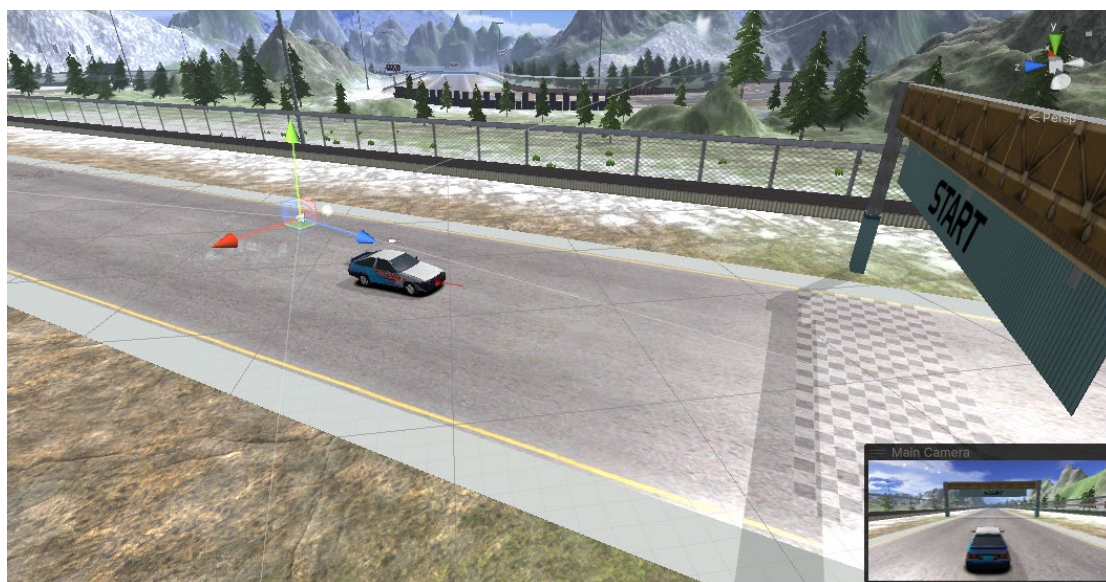


Рис. 2.4 Вигляд в конструкторі та вигляд від лиця гравця

На рис. 2.3 можна переглянути локацію та авто гравця, за яким слідкує камера, вона повністю повторює рухи гравця, та слідкую за ним. Тепер розглянемо сам код камери.

```

public float smooth = 0.3f;

public float distance = 5.0f;
public float height = 1.0f;
public float angle = 20;

public LayerMask lineOfSightMask = 0;

[HideInInspector]
public Transform target;
[HideInInspector]
public List<Transform> cameraSwitchView;

private bool farCameraView = false;

private Vector3 farCameraPosition;
private Vector3 velocity = Vector3.zero;

private float Xsmooth;
private float farDistance = 0.0f;
private float zAngleAmount = 0.0f;
private float timeScale = 0.0f;
private float currentDistance;

private int Switch = -1;

void Awake()
{
    manage = this;
    farCameraPosition = transform.position;
}

void Start()
{
    farCameraView = true;
    farCameraPosition = (AIControl.manage.firstAINode.GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode.position)
        + new Vector3(Random.Range(-5.0f, 5.0f), Random.Range(5.0f, 10.0f), Random.Range(-5.0f, 5.0f));
}

```

Рис. 2.5 Налаштування камери

У блоці коду в методі LateUpdate() камера налаштовується в залежності від поточного режиму (Switch) і положення автомобіля гравця (target):

Якщо Switch дорівнює 1, камера слідує за автомобілем гравця.

Якщо Switch менше, ніж кількість попередньо встановлених позицій камери (cameraSwitchView), камера перемикається на одну з попередньо встановлених позицій.

В інших випадках камера переходить в режим дальнього огляду і слідує за автомобілем, що знаходиться далеко від гравця.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

У методі Awake(), присвоюється посилання поточного класу VehicleCamera статичної змінної manage, щоб до нього можна було звернутись з інших скриптів.

У методі Start() ініціалізуються початкові значення для режиму дальнього огляду камери. Визначається позиція (farCameraPosition), на якій камера буде слідкувати за автомобілем, що знаходиться далеко від гравця.

У методі LateUpdate(), в залежності від режиму (Switch), налаштовується положення та орієнтація камери:

Якщо режим рівнен -1, камера слідує за автомобілем гравця, повертаючись і переміщаючись навколо нього.

Якщо режим менше встановлених позицій камери (cameraSwitchView), камера перемикається на задану позицію та орієнтацію.

В інших випадках, коли камера знаходиться в режимі дальнього огляду, вона слідує за автомобілем, що знаходиться далеко від гравця, зберігаючи певну відстань і обмежуючи поле зору.

Метод CameraSwitch() дозволяє переключати камеру між різними режимами (Switch). При кожному виклику методу CameraSwitch() значення Switch збільшується на 1, і якщо у нього більше кількості попередньо встановлених позицій камери, Switch знову встановлюється на -1 (режим слідування для автомобіля гравця).

Метод AccelerationAngle() визначає кут повороту камери по осі Z на основі даних акселерометра пристрою.

Метод AdjustLineOfSight() визначає, є чи перекриття між камерою і цільовою точкою, і повертає відстань до перешкоди, якщо воно є. Якщо перешкоди немає, повертається поточне відстань currentDistance.

Висновки до другого розділу

В цьому розділі побудовано діаграму варіантів використання (Use Case Diagram) та алгоритми роботи системи є важливими інструментами при розробці аркадних 3D ігор про дріфт. Діаграма варіантів використання допомагає

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

визначити основні функціональні можливості системи та взаємодію з користувачами. Варіанти використання описують окремі дії та сценарії взаємодії з системою.

Ще одним важливим аспектом є налаштування динамічної камери в грі. Динамічна камера забезпечує гравцю зорову перспективу на гральний світ та може покращити занурення гравця в гру. Налаштування динамічної камери може включати визначення точок огляду, керування зумом або налаштування кутів огляду в залежності від руху гравця або подій в грі.

Загалом, налаштування камери і колесами в аркадних 3D іграх про дріфт дозволяють створити більш реалістичну та захоплюючу гральну досвід. Вони допомагають створити занурення гравця в гру, забезпечують контроль і відчуття повноти над рухом транспортного засобу та дозволяють гравцям налаштувати гру під свої потреби та уподобання.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З 3D ГРОЮ

3.1 Порядок встановлення та налаштування гри

Розробка 3D Drift на рушії Unity 3D дозволяє створити гру з оптимальною оптимізацією та можливістю перенесення на різні операційні системи. Unity 3D є потужним інструментом для створення ігор, який підтримує різні платформи, такі як Windows, macOS, Android, iOS та багато інших. Це означає, що розробники можуть створювати гру в середовищі Unity 3D, а потім легко переносити її на цільову операційну систему, в залежності від потреб і вимог ринку.

Операційні системи, на яких можна запустити гру, повинні відповідати мінімальним вимогам, визначеним для Windows-версії гри.

1. процесор з тактовою частотою 2.4 ГГц або вище;
2. 4 ОЗУ;
3. 1 ГБ вільного місця на жорсткому диску;
4. Графічний пристрій з підтримкою Direct X9 (бажано Direct X11).

Ці вимоги гарантують, що гра запуститься на комп'ютерах зі зазначеними характеристиками та забезпечить комфортний геймплей.

За допомогою Unity 3D можна забезпечити стабільну частоту кадрів у грі. Зазвичай рекомендовано досягати частоти кадрів 30fps для систем, конфігурація яких не старіше 4-5 років, що забезпечить комфортну геймплейну досвід. На більшості сучасних систем можна досягти частоти кадрів 60fps, що вважається більш плавним і зручним для гри.

Отже, завдяки використанню Unity 3D та оптимальній оптимізації системи, 3D Drift може працювати на різних операційних системах та запускатись на системах з мінімальними вимогами. Гравці зможуть насолоджуватись грою зі стабільною частотою кадрів, що дозволяє отримати задоволення від геймплею навіть на середньо потужних комп'ютерах або мобільних пристроях.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з грою

Після завантаження користувач опиняється у головному меню (рис. 3.1).

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 3.1 Головне меню

Тут наявні три елементи керування, що представлені у вигляді кнопок: «Select Vehicle (Вибір автомобіля)», «Налаштування (Іконка з права зверху)» та «Exit (Вихід)». Перша з них переносить гравця у підменю вибору автомобіля, кнопка «Налаштування» - у підменю налаштувань (рис. 3.2), кнопка «Вихід» виконує вихід з гри.

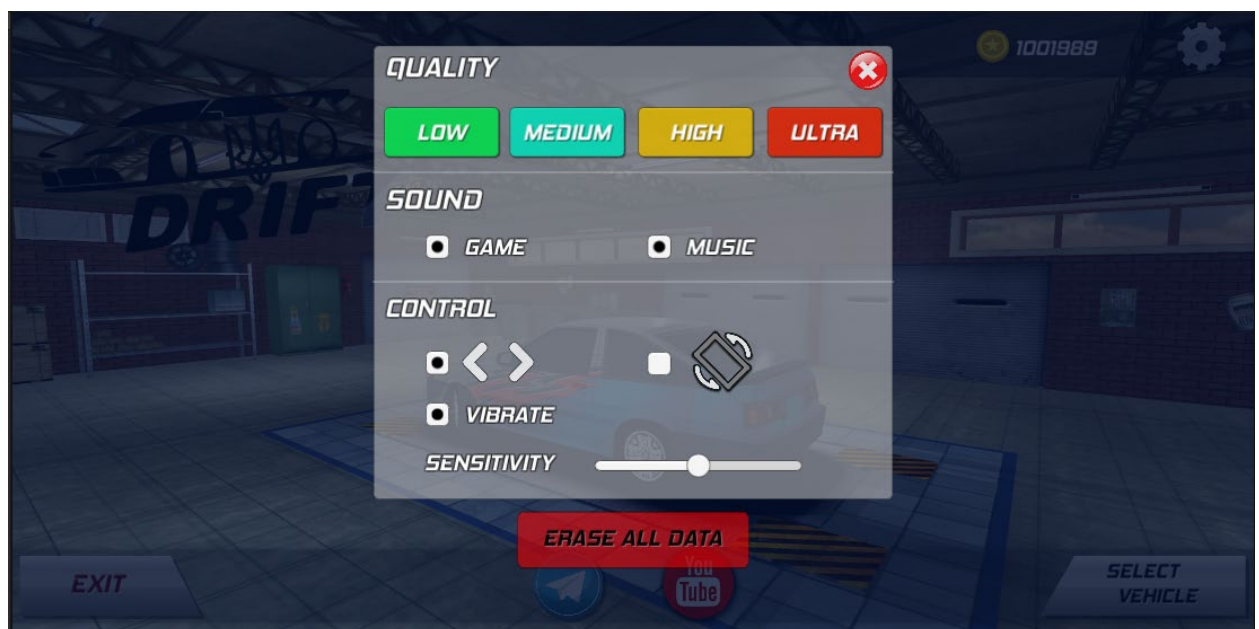


Рис. 3.2 Сторінка налаштування

Після переходу у підменю налаштувань (рис. 3.2) користувачеві будуть представлені налаштування чотирьох видів: ігрові, графічні, звукові та налаштування керування. Всі чотири доступні відразу в меню, кожен пункт

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

складається з назви налаштування та елементів керування, що надає гравцеві управління цим налаштуванням.

Під меню вибору автомобіля (рис. 3.3) складається зі списку з дванадцяти автомобілів, які можна переглянути. Кожному авто можна змінити колір дисків та колір диму з під колес. За необхідності присутня можливість виходу в ігрове меню.



Рис. 3.3 Меню вибору авто

Після успішного вибору автомобіля, гравець отримує захоплюючу можливість обрати карти, на яких він зможе випробувати свій обраний транспортний засіб. Гра пропонує 5 захоплюючих мап, кожна з яких має свої унікальні особливості та атмосферу (рис. 3.4).

Перша карта вражає гравців своїми горами, вкритими зеленим лісом та скелястими утвореннями. Цей трек надає чудову можливість насолодитися приголомшливими пейзажами та провести час серед природи.

Друга карта відправить гравців у зимову атмосферу, де всюди панує сніг та льодяники. Гладка поверхня дороги та сніжні замети створять виклик для керування автомобілем і додадуть додаткову порцію адреналіну.

Третя карта перенесе гравців на пустелю зі скелями та каньйонами. Цей трек запропонує неймовірну атмосферу пригоди та ризику, де кожна крива та вибір шляху важливі для успішного проходження гонки.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Четверта карта - нічний трек, заповнений містичними відтінками та підсвіченими елементами. Гравці можуть насолоджуватися та відчувати магію нічного дрифту, коли відблиски вогнів та тіні надають грі таємничості.

Кожна карта має свою власну атмосферу, яка поглине гравців і занурює їх у неповторний світ гри. Вибір місця для гонки стає додатковим викликом, оскільки кожна карта вимагає своєї унікальної стратегії та навичок водіння.



Рис. 3.4 Маню вибору мапи

Після того, як гравець обрав бажану карту, настає момент, коли він з нетерпінням натискає кнопку "Старт", готуючись до захоплюючої пригоди. Від цього моменту починається процес завантаження рівня, який підготує гравця до неймовірного досвіду гонок.

Під час завантаження рівня гравець може відчувати зростаючу напругу і адреналін, оскільки кожен біт інформації, що поступає в його систему, приносить його ближче до миттєвого початку гри. Завантаження рівня може супроводжуватися захоплюючими візуальними ефектами або анімаціями, що поглинають гравця ще більше в світ гри.

Цей короткий момент очікування перед початком гри виступає як своєрідна підготовка до пригоди, коли серце гравця б'ється швидше, а руки трохи

непевні. Він вже уявляє себе на трасі, готовий підкорити кожен поворот і випробувати свої навички керування в повному обсязі.

Таким чином, кнопка "Старт" стає важливим переходом, що відокремлює гравця від очікування і відкриває двері в захоплюючий світ гри, де його навички, реакція та стратегія будуть протестовані на максимум.

(рис. 3.5.).

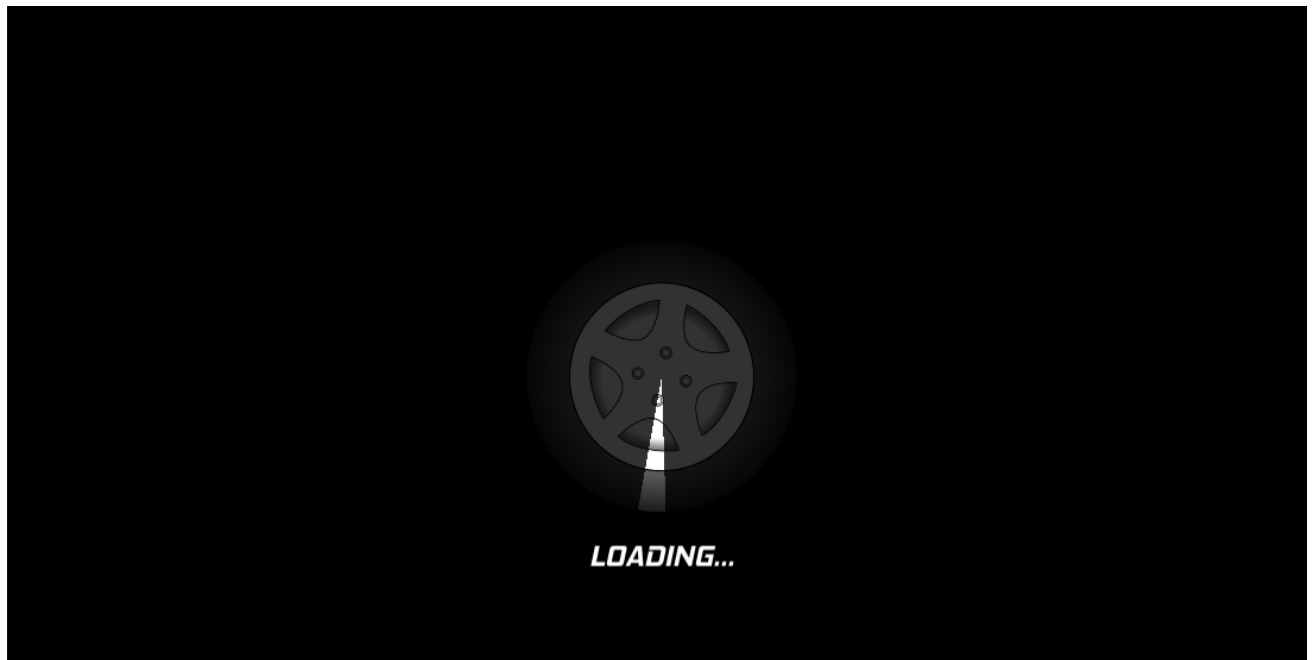


Рис. 3.5 Завантаження рівню

Після завантаження рівня і початку гри у гравця є можливість відкрити меню паузи (рис. 3.6.). Головне меню складається з 3 пунктів: “Resume”, “Restart” та “Main Menu”.

“Resume”- поверне гравця до гри з того моменту, коли була натиснута пауза.

“Restart”- почне рівень з самого початку.

“Main Menu”- перекине гравця до головного меню.

Також в меню “Паузи” праворуч зверху присутні дві іконки, які відповідають за гучність гри, та гучність музики в грі.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

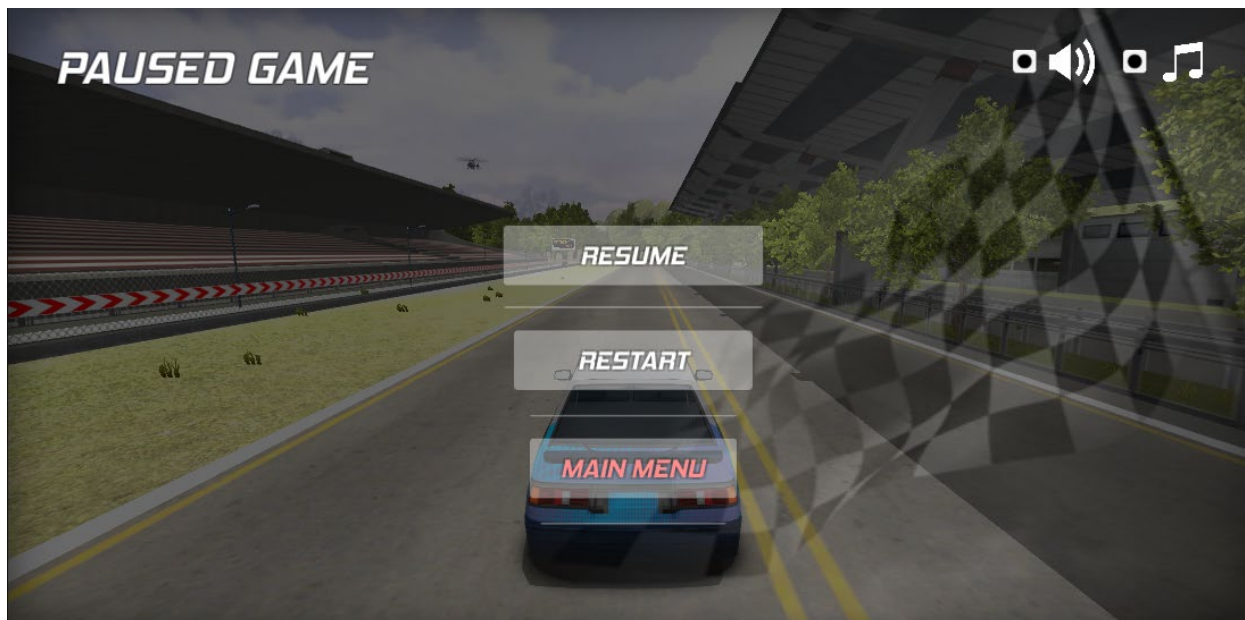


Рис. 3.6 Меню паузи в грі

Вікно кастомізації є одним з захоплюючих елементів гри, де гравець має можливість налаштувати вигляд свого автомобіля за своїми власними уподобаннями. У цьому вікні доступні різні опції для зміни зовнішнього вигляду автомобіля, зокрема колір дисків та диму.

Кастомізація кольору дисків дозволяє гравцеві вибрати бажаний колір для дисків свого автомобіля. Це може бути широкий спектр варіантів, включаючи яскраві та насичені кольори, класичні відтінки. Гравець може експериментувати з різними комбінаціями кольорів дисків, щоб створити унікальний і виразний вигляд свого автомобіля.

Крім того, гравець має можливість налаштувати колір диму, що виходить з під коліс автомобіля. Це дозволяє персоналізувати стиль гри і створити особливий ефект під час дрифту або швидкої їзди. Гравець може обрати класичний білий дим, який нагадує професійних гонщиків, або експериментувати з іншими кольорами, такими як синій, зелений, червоний, щоб створити яскравий та незабутній ефект.

Завдяки вікну кастомізації гравець має можливість виразити свою творчість та створити унікальний вигляд свого автомобіля, що відображає його стиль та особистість. Ця функція додає більше гнучкості та індивідуальності в

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

гру, дозволяючи гравцеві відчути себе справжнім творцем свого власного автомобіля в ігровому світі. (рис. 3.7).



Рис. 3.7 Вікно кастомізації

При купівлі автомобіля гравцю спливає вікно, яке інформує його про купівлю авто, гравець має можливість відмовитись, або купити його (рис. 3.8.).

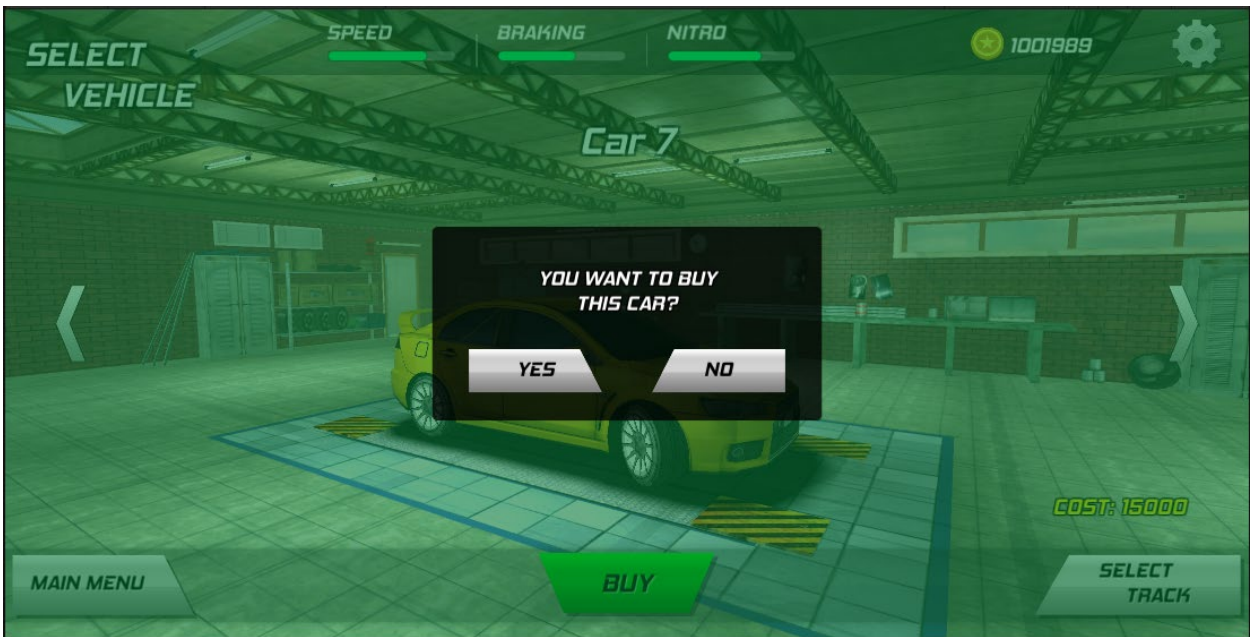


Рис. 3.8 Вікно, яке інформує про купівлю авто

3.3. Тестування роботи 3D гри

Перевірка на працездатність є невід'ємною частиною процесу розробки програмного коду системи, включаючи ігрові програми. Ігрове тестування виявляється дещо відмінним від тестування інтернет-ресурсів, мобільних

додатків та інших продуктів, оскільки вимагає специфічних підходів та уваги до деталей.

Під час ігрового тестування особлива увага зосереджується на перевірці функціональності, геймплею та загального враження від гри. Тестувальники активно спілкуються з грою, перевіряють різні можливості та сценарії, аналізують реакцію гравців та взаємодію з ігровим світом. Вони перевіряють, чи працюють усі функції гри належним чином, чи відповідають їхні результати очікуванням, і чи є які-небудь неполадки або помилки.

Під час ігрового тестування також враховуються аспекти, які впливають на іммерсію гравців. Це може включати перевірку візуальних ефектів, звукового супроводу, рівня деталізації графіки та забезпечення плавності гри. Тестувальники також перевіряють рівні складності, баланс гри та взаємодію з іншими гравцями або штучним інтелектом.

У процесі ігрового тестування використовуються спеціальні інструменти та програми, які допомагають виявляти та відстежувати помилки, а також збирати дані для подальшого аналізу. Результати тестування допомагають розробникам виправити помилки, поліпшити функціональність та забезпечити кращий геймплей для користувачів.

Загалом, ігрове тестування відіграє важливу роль у забезпеченні якості та задоволення від гри для гравців. Воно допомагає перевірити функціональність, взаємодію та іммерсію гри, а також виявити та виправити помилки та недоліки.

Тестування ігрових систем містить такі основні види:

- тестування функціональності;
- тестування контенту;
- тестування геймплею;
- користувацьке тестування.

Тестування функціональності. Нижче наведено тест-кейси роботи головного меню.

Кнопка «Settings»:

Передумови: гравець повинен знаходитись у головному меню;

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Кроки: натискання кнопки «Settings»;

Очікуваний результат: перехід до вікна з налаштуваннями;

Фактичний результат: перехід до вікна з налаштуваннями.

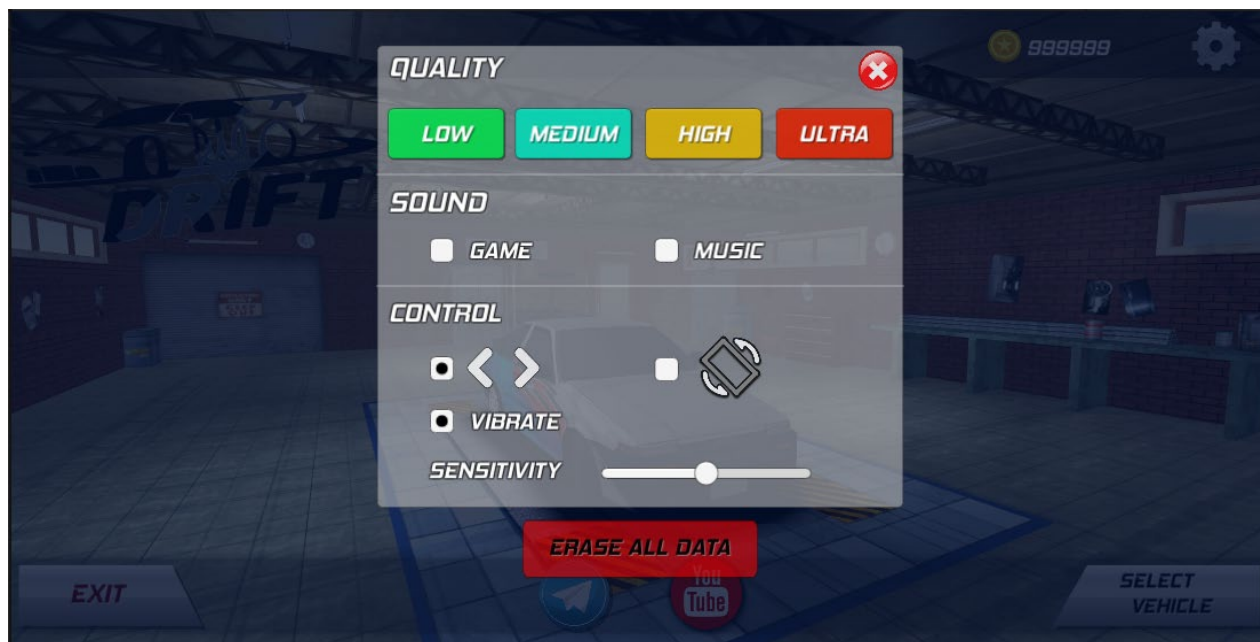


Рис. 3.9 Вікно з налаштуваннями

Кнопка «Select Vehicle»:

Передумови: Гравець повинен знаходитись у головному меню;

Кроки: Натискання кнопки «Play»;

Очікуваний результат: перехід до списку авто.

Фактичний результат: перехід до списку авто;



Рис. 3.10 Вікно з списком авто

Кнопки «Low/Medium/High/Ultra»:

Передумови: Гравець повинен знаходитись у головному меню;

Кроки: натискання кнопок «Low/Medium/High/Ultra» для зміни якості гри;

Очікуваний результат: покращення або погіршення якості картинки, відстані деталізації об'єктів;

Фактичний результат: покращення або погіршення якості картинки, відстані деталізації об'єктів.

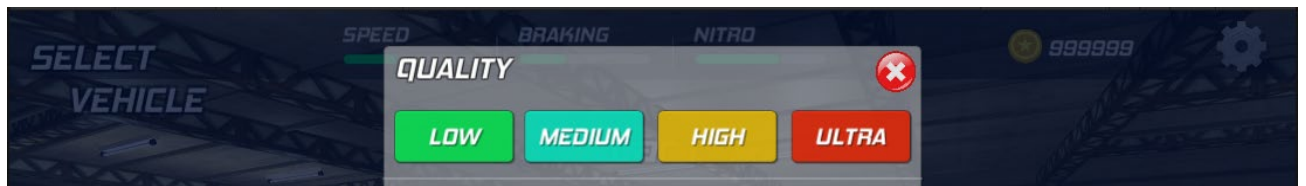


Рис. 3.11 Вікно з налаштуванням графіки



Рис. 3.12 Вигляд гри з рівнем якості “Low”

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 3.13 Вигляд гри з рівнем якості “Ultra”

Кнопка «Select Track»:

Передумови: гравець повинен знаходитись у меню «Select Vehicle»;

Кроки: натискання кнопки «Select Track» для вибору траси;

Очікуваний результат: перехід до списку ігрових мап та вибору мапи яка буде підсвічуватись коли гравець обере її;

Фактичний результат: перехід до списку ігрових мап та вибір мапи, яка підсвічується коли ми обираємо її.



Рис. 3.14 Вибір мапи (обрана мапа 2)

Кнопка «Start»:

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

Передумови: гравець повинен знаходитись у меню «Select Track»;

Кроки: натискання кнопки «Start» для завантаження рівня;

Очікуваний результат: перехід у екран завантаження;

Фактичний результат: перехід у екран завантаження.

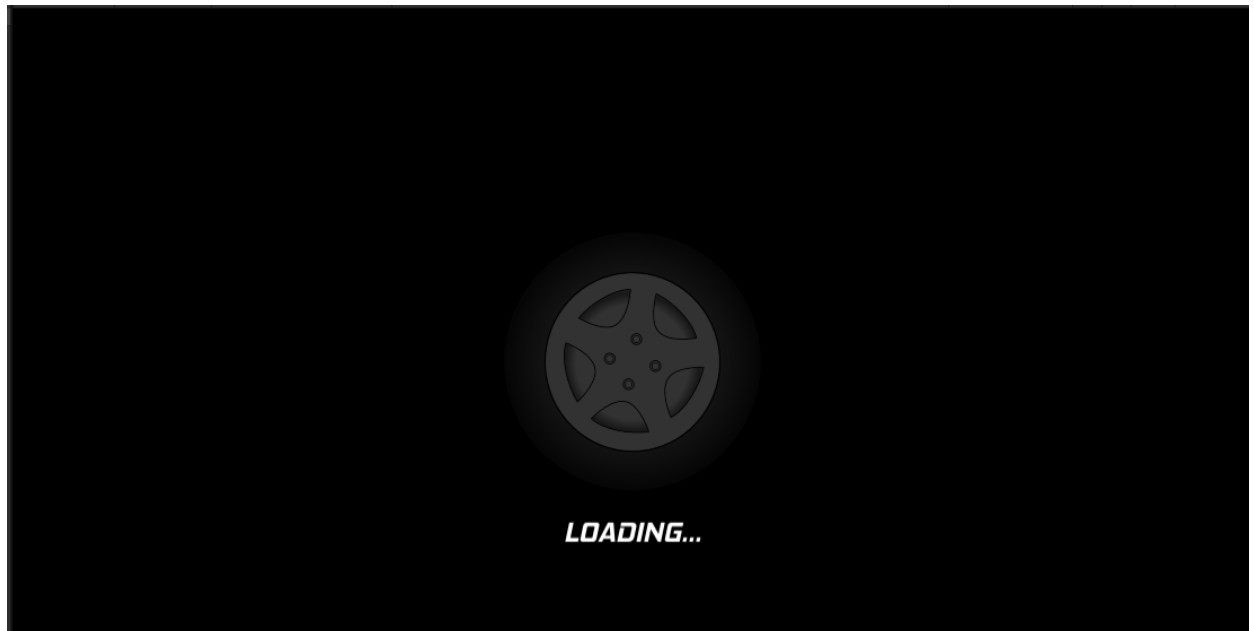


Рис. 3.15 Завантаження рівня

Кнопка «Erase All Date» у головному меню:

Передумови: гравець повинен знаходитись у головному меню;

Кроки: натискання кнопки «Erase All Date»;

Очікуваний результат: Спливаюче вікно «Game Rest» на кнопки підтвердження «Yes» та «No», при натисненні кнопки «Yes» відбувається скидання прогресу до нуля, «No» закриває спливаюче вікно «Game Rest»;

Фактичний результат: Спливаюче вікно «Game Rest» на кнопки підтвердження «Yes» та «No», при натисненні кнопки «Yes» - скидання прогресу до нуля, «No» - закривання спливаючого вікна «Game Rest».

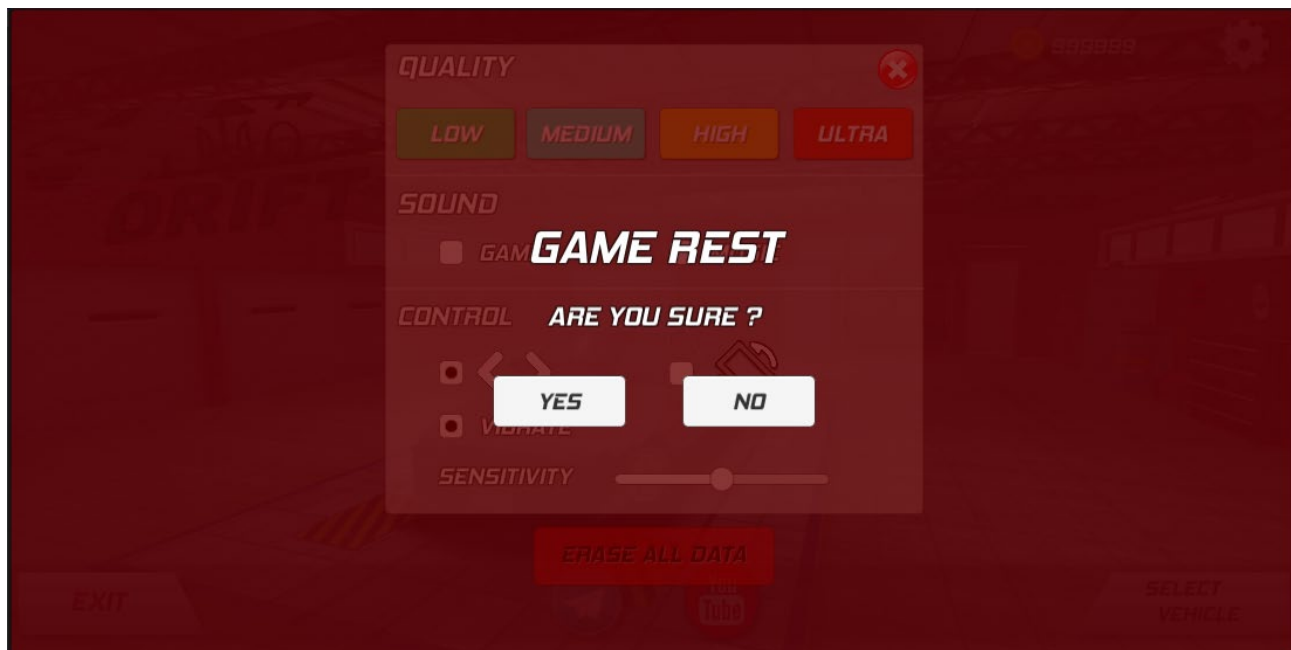


Рис. 3.16 Меню «Game Rest»

Тестування геймплею.

Проведено комплексне тестування геймплею, зокрема основної функціональності, яка включає керування автомобілем на різних швидкостях, взаємодію з різними ділянками траси та виконання дрифтів, відомих як "занос".

Під час тестування було ретельно перевірено роботу керування автомобілем на різних швидкостях, перевірено його реакцію та оборотність на поворотах та прямих ділянках. Кожна частина траси була пройдена з метою виявлення можливих неполадок або незручностей у керуванні, наприклад, неправильної поведінки автомобіля на виїзді з повороту або непропорційного кута дрейфу.

Окрема увага була приділена дослідженню дрифтів, які є важливою складовою геймплею гри. Було перевірено можливість автомобіля зайти в дрифт, його стійкість та контрольованість під час дрейфу. Реалістичність та відчуття занурення в процес гри були оцінені шляхом тестування реакції автомобіля на рухи гравця та взаємодії з фізикою середовища.

Після детального тестування доступного функціоналу гри було встановлено, що недоліки або помилки в роботі гри не були виявлені. Геймплей відповідає очікуванням і передбачуваному функціоналу, а керування

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

автомобілем та його динаміка показують високу якість та відповідність задуманому геймплею.

Висновки до третього розділу

В цьому розділі розроблено дизайн гри, моделей авто, та локацій. Завдяки проектуванню діаграм визначили основні функції гравця. На основі якої зрозумілі можливості гравця, а саме: можливість кастомізування, придбання авто, вибору локації для тестування автомобіля, функціонал заносу, покращення або погіршення якості гри для слабких або кращих девайсів, скидання прогресу для початку все з нуля.

Проведено комплексне тестування гри, під час якого виявлені та виправлені незначні помилки. Після виправлень, гра функціонує бездоганно та без перебоїв, повністю задовольняючи всі вимоги її функціоналу. Загальний стан гри після тестування відповідає вимогам і вона готова до використання без будь-яких проблем.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи спроектовано та реалізовано “3D гру на базі Unity 3D”. Під час роботи над цим проектом вирішено ряд завдань:

У першому розділі було встановлено технічні вимоги та проведено аналіз існуючих аналогів аркадних ігор, таких як Tetris, Pac-Man, Space Invaders, Galaga та Donkey Kong. Цей аналіз дозволив визначити найбільш популярні аркадні ігри. Для реалізації проекту було вирішено використати такі інструменти як VS Code (інтегроване середовище розробки), ігровий движок Unity, а також мову програмування C#, яка використовується для розробки логіки і функціональності гри.

У другому розділі було побудовано діаграму варіантів використання (Use Case Diagram) та розроблено алгоритми роботи системи. Ці інструменти є важливими при розробці аркадних 3D ігор про дріфт, оскільки вони допомагають визначити основні функціональні можливості системи та взаємодію з користувачами.

Додатково, було приділено увагу налаштуванню динамічної камери в грі. Динамічна камера є важливим елементом, який забезпечує гравцю зорову перспективу на гральний світ та покращує занурення гравця в гру.

У третьому розділі було розроблено дизайн гри, включаючи моделі автомобілів та локацій. Було проведено проектування діаграм, яке визначило основні функції гравця, такі як можливість кастомізації, придбання автомобілів, вибір локацій для тестування автомобіля, функціонал заносу, а також можливість покращення або погіршення якості гри в залежності від можливостей пристрою. Також була реалізована функція скидання прогресу гри для початку з початку.

У заключному розділі було проведено тестування гри, виявлено та виправлено незначні помилки. Гра в цілому визнана придатною до використання та відповідає своєму функціоналу.

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chris Dickinson. Unity Game Optimization. 2019. 398 p.
2. Chris Dickinson. Unity Game Optimization: Optimize, Monetize, and Customize Your Unity Game. 2017. 338 p.
3. Francesco Sapio. Unity UI Cookbook. 2018. 284 p.
4. Harrison Ferrone. Learning C# by Developing Games with Unity. 2019. 506 p.
5. Joe Hocking. Unity in Action: Multiplatform Game Development in C#. 2018. 527 p.
6. Joe Hocking. Unity in Action: Multiplatform Game Development in C#, Second Edition. 2018. 504 p.
7. Jon Manning. Unity Game Development Cookbook: Essentials for Every Game. 2019. 406 p.
8. Jonathan Linowes. Unity Virtual Reality Projects. 2015. 272 p.
9. Ray Barrera. Unity Artificial Intelligence Programming. 2015. 232 p.
10. Simon Jackson. Mastering Unity 2D Game Development. 2014. 384 p.
11. Tanya X. Short, Tarn Adams. Procedural Generation in Game Design. 2017. 232 p.
12. Офіційна документація Unity 3D [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unity3d.com/ru/530/Manual/UnityManual.html>.
13. Офіційний сайт платформи VS Code [Електронний ресурс] – Режим доступу до ресурсу: <https://visualstudio.microsoft.com/ru/>.
14. Unity Learn Офіційний освітній ресурс Unity з безкоштовними курсами та навчальним матеріалом [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.unity.com/>.
15. Unity3d.College Сайт з навчальними матеріалами, статтями, підручниками та відеоуроками з Unity. [Електронний ресурс] – Режим доступу до ресурсу <https://unity3d.college/>
16. Catlike Coding Сайт, що пропонує серії уроків, які розкривають певні аспекти роботи з Unity. [Електронний ресурс] - Режим доступу до ресурсу: <https://catlikecoding.com/unity/tutorials/>

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

- 17.Unity Patterns Сайт, що описує різні архітектурні патерни та підходи до розробки в Unity. [Електронний ресурс] - Режим доступу до ресурсу:
<https://unitypatterns.com/>
- 18.Ray Wenderlich Веб-сайт з багатьма статтями та відеоуроками з Unity та різних аспектів розробки ігор. [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.raywenderlich.com/unity>
- 19.Unity Technologies Blog Офіційний блог Unity з оновленнями, статтями та порадами від експертів. [Електронний ресурс] - Режим доступу до ресурсу:
<https://blogs.unity3d.com/>
- 20.Udemy Онлайн-платформа з курсами та навчальними ресурсами з розробки ігор у Unity. [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.udemy.com/courses/game-development/unity/>.
21. Unity Ninja Форум та ресурс для обміну знаннями, створений спільнотою розробників Unity. [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.unityninja.net/>

ДОДАТКИ

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Код AIVehicle.cs

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

public class AIVehicle : MonoBehaviour
{
    public float carRestTime = 5.0f;

    public float forwardSpeed = 1.0f;
    public float steerSpeed = 1.0f;
    public float nextNodeDistance = 5;

    public Transform frontPoint;

    [HideInInspector]
    public Transform currentNode, nextNode, wrongNode;
    [HideInInspector]
    public int currentLap = 0;

    [HideInInspector]
    public float playerCurrentTime, playerBestTime, playerLastTime = 0.0f;
    [HideInInspector]
    public float AIAccel, AISteer = 0.0f;
    [HideInInspector]
    public bool AIBrake = false;

    private VehicleControl vehicleControl;

    private bool goNextNode = true;
    private bool getLap = false;

    private int carPreviousNodes = 0;
    private float targetAngle;
    private float restTimeer = 0.0f;

    void Start()
    {
        restTimeer = carRestTime;
        currentLap = 0;
        vehicleControl = transform.GetComponent<VehicleControl>();
        currentNode = nextNode;
    }

    void Update()
    {
        if (GameUI.manage.gameStarted)
            playerCurrentTime += Time.deltaTime;

        AICarControl();
    }
}
```

Код який відповідає за керування рухом

```
void AICarControl()
{
    if (AIControl.manage.firstAINode == wrongNode && !getLap)
```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            currentLap++;
            playerLastTime = playerCurrentTime;

            if (currentLap == 1) GameUI.manage.gameFinished = true;

            if (playerBestTime == 0.0f || playerBestTime > playerCurrentTime)
            playerBestTime = playerCurrentTime;

            playerCurrentTime = 0.0f;
            getLap = true;
        }
        else if (AIControl.manage.startPoint != currentNode)
        {
            getLap = false;
        }

        //////////////////////////////////////
        //////////////////////////////////////

        Vector3 CurrentNodeForward = currentNode.TransformDirection(Vector3.forward);
        Vector3 CurrentNodetoOther = currentNode.position - frontPoint.position;

        Vector3 NextNodeForward = nextNode.TransformDirection(Vector3.forward);
        Vector3 NextNodetoOther = nextNode.position - frontPoint.position;

        carPreviousNodes = Mathf.Clamp(carPreviousNodes, 0, 6);

        if (Mathf.Abs(Quaternion.Dot(nextNode.rotation, frontPoint.rotation)) < 0.5f &&
!GameUI.manage.gameFinished)
            GameUI.manage.carWrongWay = true;
        else
            GameUI.manage.carWrongWay = false;

        if (nextNode)
        {
            if (vehicleControl.vehicleMode == VehicleMode.AICar)
            {
                if (Vector3.Distance(frontPoint.position, nextNode.position) <
nextNodeDistance && nextNode != currentNode)
                {
                    currentNode = nextNode;
                    wrongNode = nextNode;
                    goNextNode = true;
                }
            }
            else if (vehicleControl.vehicleMode == VehicleMode.Player)
            {
                if (Vector3.Dot(NextNodeForward, NextNodetoOther) < 0.0f)
                {
                    carPreviousNodes--;
                    currentNode = nextNode;

                    if (wrongNode != null && wrongNode.GetComponent<AINode>().nextNode ==
nextNode) wrongNode = nextNode;

                    goNextNode = true;
                }
            }
        }
    }

```

```

        else if (Vector3.Dot(CurrentNodeForward, CurrentNodeToOther) > 0.0f &&
currentNode.GetComponent<AINode>().previousNode != currentNode)
        {
            carPreviousNodes++;
            currentNode = currentNode.GetComponent<AINode>().previousNode;

            if (carPreviousNodes == 5)
            {
                carPreviousNodes = 0;
                transform.GetComponent<Rigidbody>().Sleep();
                transform.rotation = wrongNode.rotation;
                transform.position = wrongNode.position + Vector3.up;

                currentNode = wrongNode;
            }

            goNextNode = true;
        }
    }

    if (Vector3.Distance(frontPoint.position, nextNode.position) >
nextNodeDistance * 5)
    {
        carPreviousNodes = 0;
        transform.GetComponent<Rigidbody>().Sleep();
        transform.rotation = wrongNode.rotation;
        transform.position = wrongNode.position + Vector3.up;

        currentNode = wrongNode;
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

GameUI.manage.carBrakeWarning = false;

if (currentNode.GetComponent<AINode>())
{
    AINode Nodescript = currentNode.GetComponent<AINode>();

    if (Mathf.Abs(AISteer) > 0.0f && vehicleControl.speed > 70.0f &&
Nodescript.nodeSetting.brakeing)
    {
        AIAccel = -Mathf.Abs(AISteer);
        GameUI.manage.carBrakeWarning = true;
    }
    else
    {
        AIAccel = 0.2f;
    }

    if (goNextNode)
    {
        nextNode = Nodescript.nextNode;
        goNextNode = false;
    }
}

var relativeTarget = transform.InverseTransformPoint(nextNode.position);

targetAngle = Mathf.Atan2(relativeTarget.x, relativeTarget.z);
targetAngle *= Mathf.Rad2Deg;
targetAngle = Mathf.Clamp(targetAngle, -65, 65);

AISteer = Mathf.SmoothStep(AISteer, targetAngle / 60, steerSpeed / 3.0f);

```

```

    }

}

```

Код який відповідає за форматування часу

```

using UnityEngine;
using System.Collections;
using System;

public class FormatSecondsScript : MonoBehaviour
{
    public string FormatSeconds(float elapsed)
    {
        int d = (int)(elapsed * 100.0f);
        int minutes = d / (60 * 100);
        int seconds = (d % (60 * 100)) / 100;
        int hundredths = d % 100;
        return String.Format("{0:00}:{1:00}.{2:00}", minutes, seconds, hundredths);
    }
}

```

Код який відповідає за керування інтерфейсом

```

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

public class GameUI : MonoBehaviour
{
    public static GameUI manage;

    public Panels panels;
    public CarUI carUI;
    public Sounds sounds;

    [HideInInspector]
    public int totalDriftCoins;
    [HideInInspector]
    public int earnCoins, totalEarnCoins;
    [HideInInspector]
    public bool canDrift;
    [HideInInspector]
    public int driftAmount = 0;
    [HideInInspector]
    public float penaltyTime = 0.0f;
    [HideInInspector]
    public bool gameStarted, gamePaused, gameRest, gameFinished = false;
    [HideInInspector]
    public int racePrize, totalPrize = 0;

    private int currentLevelNumber = 0;

    private float menuLoadTime = 0.0f;
    private AsyncOperation sceneLoadingOperation = null;

    private AIVehicle AIVehicleScript;
    private float timerDrift = 1.0f;
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[HideInInspector]
public bool carPenalty=false;
[HideInInspector]
public bool carWrongWay = false;
[HideInInspector]
public bool carBrakeWarning = false;

private int gearst = 0;
private float thisAngle = -150;

private float startTimer = 1.0f;
private int startCont = 3;

[System.Serializable]
public class Panels
{

    public GameStart gameStart;
    public Gameplay gamePlay;
    public GamePuased gamePuased;
    public GameFinish gameFinish;

    [System.Serializable]
    public class GameStart
    {
        public GameObject root;
        public Text startTimeUI;

        public GameObject loading;
        public Image loadingBar;

        public Animator FadeBackGround;
    }

    [System.Serializable]
    public class Gameplay
    {
        public GameObject root;
        public GameObject buttonsUI, accelUI;
        public Image wrongWay;
        public Image driftWheel;
        public Image brakeWarning;
        public Text currentTime, bestTime;
        public Text driftCoins;
        public Text driftText;
        public Text driftXText;
        public Text penaltyText;
    }

    [System.Serializable]
    public class GamePuased
    {
        public GameObject root;
        public Toggle audioToggle;
        public Toggle musicToggle;
    }

    [System.Serializable]
    public class GameFinish
    {
        public GameObject root;
        public Text yourTime, penaltyTime, bestTime;
        public Text racePrize, driftPrize;
        public Text totalPrize;
        public StarClass stars;
    }

    [System.Serializable]
    public class StarClass

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        public float Star1Time, Star2Time, Star3Time;
        public Image Star1, Star2, Star3;
    }

[System.Serializable]
public class CarUI
{
    public Image tachometerNeedle;
    public Image barShiftGUI;

    public Text speedText;
    public Text GearText;
}

[System.Serializable]
public class Sounds
{
    public AudioSource music;
    public AudioClip countDown, countStart;
}

```

Код який відповідає за функціонал гри, а саме “Початок таймеру на початку рівня, заносом, завантаження сцени”

```

void Awake()
{

    manage = this;
    AudioListener.pause = false;
    Time.timeScale = 1.0f;

    if (PlayerPrefs.GetFloat("BestTime" + Application.loadedLevel.ToString()) != 0.0f)
        panels.gamePlay.bestTime.text = "Best: " +
        FormatSeconds(PlayerPrefs.GetFloat("BestTime" + Application.loadedLevel.ToString()));

    panels.gamePuased.audioToggle.isOn = (PlayerPrefs.GetInt("AudioActive") == 0) ?
true : false;
    panels.gamePuased.musicToggle.isOn = (PlayerPrefs.GetInt("MusicActive") == 0) ?
true : false;

    AudioListener.volume = (PlayerPrefs.GetInt("AudioActive") == 0) ? 1.0f : 0.0f;
    sounds.music.mute = (PlayerPrefs.GetInt("MusicActive") == 0) ? false : true;

}

void Start()
{
    if (AIControl.manage.controlMode == ControlMode.Mobile)
    {
        switch (PlayerPrefs.GetString("ControlMode"))
        {
            case "":
                panels.gamePlay.buttonsUI.SetActive(true);
                break;
            case "Buttons":

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        panels.gamePlay.buttonsUI.SetActive(true);
        break;
    case "Accel":
        panels.gamePlay.accelUI.SetActive(true);
        break;
    }
}

void Update()
{
    StartingGameTimer();

    ShowCarUI();

    if (sceneLoadingOperation != null)
    {
        panels.gameStart.loadingBar.fillAmount =
Mathf.MoveTowards(panels.gameStart.loadingBar.fillAmount, sceneLoadingOperation.progress
+ 0.2f, Time.deltaTime * 0.5f);

        if (panels.gameStart.loadingBar.fillAmount > sceneLoadingOperation.progress)
            sceneLoadingOperation.allowSceneActivation = true;
    }

    if (carPenalty) { penaltyTime += Time.deltaTime;
panels.gamePlay.penaltyText.color = Color.red; } else { panels.gamePlay.penaltyText.color
= Color.white; }

    if (carWrongWay) { panels.gamePlay.wrongWay.gameObject.SetActive(true); } else {
panels.gamePlay.wrongWay.gameObject.SetActive(false); }

    if (carBrakeWarning) { panels.gamePlay.brakeWarning.color = new Color(1, 1, 1,
1); } else { panels.gamePlay.brakeWarning.color = new Color(1, 1, 1, 0.2f); }

    panels.gamePlay.currentTime.text = "Time: " +
FormatSeconds(AIControl.CurrentVehicle.AIVehicle.playerCurrentTime);
    panels.gamePlay.bestTime.text = "Best: " +
FormatSeconds(PlayerPrefs.GetFloat("BestTime" + Application.loadedLevel));
    panels.gamePlay.penaltyText.text = "Penalty: " +
FormatSeconds(penaltyTime).ToString();

    panels.gamePlay.driftCoins.text = "Drift Coins: " + totalDriftCoins.ToString();

    if (!gameFinished && !carWrongWay)
    {
        if (timerDrift == 0)
            canDrift = true;
        else
            timerDrift = Mathf.MoveTowards(timerDrift, 0.0f, Time.deltaTime);

        if ((driftAmount / 100.0f) > 1.0f)
        {
            timerDrift = 1.0f;
            earnCoins = (driftAmount - 100);

            panels.gamePlay.driftWheel.fillAmount = 1.0f;
            panels.gamePlay.driftWheel.rectTransform.Rotate(0, 0, -500.0f *
Time.deltaTime);
            panels.gamePlay.driftText.text = (driftAmount - 100).ToString();

            panels.gamePlay.driftXText.gameObject.SetActive(true);

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (earnCoins > 300)
        {
            if (panels.gamePlay.driftXText.text == "2X")
            {
                panels.gamePlay.driftXText.GetComponent<Animator>().Play(0);
                panels.gamePlay.driftXText.text = "3X";
            }
            totalEarnCoins = earnCoins*3;
        }
        else if (earnCoins > 150)
        {
            if (panels.gamePlay.driftXText.text == "1X")
            {
                panels.gamePlay.driftXText.GetComponent<Animator>().Play(0);
                panels.gamePlay.driftXText.text = "2X";
            }
            totalEarnCoins = earnCoins*2;
        }
        else if (earnCoins > 0)
        {
            panels.gamePlay.driftXText.text = "1X";
            totalEarnCoins = earnCoins;
        }
    }
    else if (canDrift)
    {
        if (panels.gamePlay.driftWheel.fillAmount == 1)
        {
            totalDriftCoins += totalEarnCoins;
            totalEarnCoins = 0;
            earnCoins = 0;
        }

        panels.gamePlay.driftWheel.fillAmount = driftAmount / 100.0f;
        panels.gamePlay.driftWheel.rectTransform.rotation = Quaternion.identity;
        panels.gamePlay.driftText.text = "";
        panels.gamePlay.driftXText.gameObject.SetActive(false);
    }
}
else
{
    driftAmount = 0;
    panels.gamePlay.driftWheel.fillAmount = 0.0f;
    panels.gamePlay.driftWheel.rectTransform.rotation = Quaternion.identity;
    panels.gamePlay.driftText.text = "";
    panels.gamePlay.driftXText.gameObject.SetActive(false);

    if (gameFinished)
    {
        if (panels.gameFinish.root.activeSelf == false)
        {

            AIVehicleScript = AIControl.CurrentVehicle.AIVehicle;

            panels.gameFinish.yourTime.text = "Current: " +
            FormatSeconds(AIVehicleScript.playerBestTime);
            panels.gameFinish.penaltyTime.text = "Penalty: " +
            FormatSeconds(penaltyTime);

            if (AIVehicleScript.playerBestTime < PlayerPrefs.GetFloat("BestTime"
+ Application.loadedLevel) || PlayerPrefs.GetFloat("BestTime" + Application.loadedLevel)
== 0)
            {
                PlayerPrefs.SetFloat("BestTime" + Application.loadedLevel,
AIVehicleScript.playerBestTime);
            }
        }
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        panels.gameFinish.bestTime.text = "Best: " +
FormatSeconds(AIVehicleScript.playerBestTime);
    }
    else
    {
        panels.gameFinish.bestTime.text = "Best: " +
FormatSeconds(PlayerPrefs.GetFloat("BestTime" + Application.loadedLevel));
    }

    if (AIVehicleScript.playerBestTime <
(panels.gameFinish.stars.Star3Time-penaltyTime))
    {

        panels.gameFinish.stars.Star1.color = Color.white;
        panels.gameFinish.stars.Star2.color = Color.white;
        panels.gameFinish.stars.Star3.color = Color.white;

        racePrize = (int)((Application.loadedLevel / 5.0f) * 3000.0f);
        PlayerPrefs.SetInt("LevelStar" + Application.loadedLevel, 3);

    }
    else if (AIVehicleScript.playerBestTime <
(panels.gameFinish.stars.Star2Time - penaltyTime))
    {

        panels.gameFinish.stars.Star1.color = Color.white;
        panels.gameFinish.stars.Star2.color = Color.white;

        racePrize = (int)((Application.loadedLevel / 5.0f) * 2000.0f);
        PlayerPrefs.SetInt("LevelStar" + Application.loadedLevel, 2);

    }
    else if (AIVehicleScript.playerBestTime <(
panels.gameFinish.stars.Star1Time - penaltyTime))
    {

        panels.gameFinish.stars.Star1.color = Color.white;

        racePrize = (int)((Application.loadedLevel / 5.0f) * 1500.0f);
        PlayerPrefs.SetInt("LevelStar" + Application.loadedLevel, 1);
    }
    else
    {
        PlayerPrefs.SetInt("LevelStar" + Application.loadedLevel, 0);
        racePrize = (int)((Application.loadedLevel / 5.0f) * 1000.0f);
    }

    totalPrize = (totalDriftCoins + racePrize);

    panels.gameFinish.racePrize.text = "Track: " + racePrize.ToString();
    panels.gameFinish.driftPrize.text = "Drift Coins: " +
totalDriftCoins.ToString();
    panels.gameFinish.totalPrize.text = "Total: " +
totalPrize.ToString();

    panels.gamePlay.root.gameObject.SetActive(false);
    panels.gamePuased.root.gameObject.SetActive(false);
    panels.gameFinish.root.gameObject.SetActive(true);

    PlayerPrefs.SetInt("GameScore", PlayerPrefs.GetInt("GameScore") +
totalPrize);

    if (PlayerPrefs.GetInt("CurrentLevelUnlocked") <=
Application.loadedLevel)
        PlayerPrefs.SetInt("CurrentLevelUnlocked",
Application.loadedLevel);

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		



Код який відповідає за налаштування камери її позиціонування та поведінки

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

public class VehicleCamera : MonoBehaviour
{
    public static VehicleCamera manage;

    public float smooth = 0.3f;

    public float distance = 5.0f;
    public float height = 1.0f;
    public float angle = 20;

    public LayerMask lineOfSightMask = 0;

    [HideInInspector]
    public Transform target;
    [HideInInspector]
    public List<Transform> cameraSwitchView;

    private bool farCameraView = false;

    private Vector3 farCameraPosition;
    private Vector3 velocity = Vector3.zero;

    private float Xsmooth;
    private float farDistance = 0.0f;
    private float zAngleAmount = 0.0f;
    private float timeScale = 0.0f;
    private float currentDistance;

    private int Switch = -1;

    void Awake()
    {
        manage = this;
        farCameraPosition = transform.position;
    }

    void Start()
    {
        farCameraView = true;
        farCameraPosition = (AIControl.manage.firstAINode.GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
```

```

        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode
        .GetComponent<AINode>().nextNode.position)
        + new Vector3(Random.Range(-5.0f, 5.0f), Random.Range(5.0f, 10.0f),
Random.Range(-5.0f, 5.0f));
    }

    void LateUpdate()
    {

        if (GameUI.manage.gameFinished)
            Switch = 4;

        farDistance = Vector3.Distance(farCameraPosition, target.position);
        if (farDistance < 100.0f && farCameraView) farCameraView = false;

        //transform.GetComponent<Blur>().enabled = GameUI.manage.gamePaused ? true :
false;

        // add MotionBlur effect to camera
        //if (AIControl.CurrentVehicle.shifting || GameUI.manage.driftAmount > 25)
        //transform.GetComponent<MotionBlur>().blurAmount =
Mathf.Lerp(transform.GetComponent<MotionBlur>().blurAmount, 0.5f, Time.deltaTime * 5);
        //else
        //transform.GetComponent<MotionBlur>().blurAmount =
Mathf.Lerp(transform.GetComponent<MotionBlur>().blurAmount, 0.0f, Time.deltaTime);

        if (Switch == -1)
        {

            RenderSettings.flareStrength = 0.3f;

            GetComponent<Camera>().fieldOfView =
Mathf.Clamp(AIControl.CurrentVehicle.speed / 10.0f + 60.0f, 60, 90.0f);

            currentDistance = distance;
            float yAngle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
target.eulerAngles.y, ref velocity.y, smooth);

            float xAngle = Mathf.SmoothDampAngle(transform.eulerAngles.x,
target.eulerAngles.x + (angle), ref velocity.x, smooth);

            // Look at the target
            transform.eulerAngles = new Vector3(xAngle, yAngle, AccelerationAngle());

            Xsmooth = Mathf.Lerp(Xsmooth, velocity.y, Time.deltaTime * 10.0f);
            var direction = transform.rotation * -new Vector3(-Xsmooth / 300.0f, 0, 1);
            var targetDistance = AdjustLineOfSight(target.position + new Vector3(0,
height, 0), direction);

            transform.position = target.position + new Vector3(0, height, 0) + direction
* targetDistance;

        }
        else if (Switch < AIControl.CurrentVehicle.cameraView.cameraSwitchView.Count)
        {

            RenderSettings.flareStrength = 0.3f;
            GetComponent<Camera>().fieldOfView = 60;

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        transform.position = cameraSwitchView[Switch].position;
        transform.rotation = Quaternion.Lerp(transform.rotation,
cameraSwitchView[Switch].rotation, Time.deltaTime * 5.0f);

    }
    else {

        if (farDistance > 120.0f && !farCameraView)
        {
            farCameraPosition = (AIControl.CurrentVehicle.AIVehicle.nextNode
                .GetComponent<AINode>().nextNode
                .GetComponent<AINode>().nextNode
                .GetComponent<AINode>().nextNode
                .GetComponent<AINode>().nextNode
                .GetComponent<AINode>().nextNode
                .GetComponent<AINode>().nextNode.position)
                + new Vector3(Random.Range(-5.0f, 5.0f), Random.Range(10.0f, 15.0f),
Random.Range(-5.0f, 5.0f));

            farCameraView = true;

        }

        RenderSettings.flareStrength = 0.0f;

        GetComponent<Camera>().fieldOfView = Mathf.Clamp(50.0f - (farDistance /
2.0f), 10.0f, 120.0f);

        var newRotation = Quaternion.LookRotation(target.position -
transform.position);

        transform.position = farCameraPosition;
        transform.rotation = Quaternion.Slerp(transform.rotation, newRotation,
Time.deltaTime * 15);
    }

}

public void CameraSwitch()
{
    Switch++;
    if (Switch > AIControl.CurrentVehicle.cameraView.cameraSwitchView.Count) { Switch
= -1; }
}

////////////////////////////////////
////////////////////////////////////

private float AccelerationAngle()
{
    zAngleAmount = Mathf.Clamp(zAngleAmount, -45.0f, 45.0f);
    zAngleAmount = Mathf.Lerp(zAngleAmount, Input.acceleration.x * -70.0f,
Time.deltaTime * 2.0f);
    return zAngleAmount;
}

float AdjustLineOfSight(Vector3 target, Vector3 direction)
{
    RaycastHit hit;

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (Physics.Raycast(target, direction, out hit, currentDistance,
lineOfSightMask.value))
            return hit.distance;
        else
            return currentDistance;
    }
}

```

VehicleControl.cs (Частина коду яка відповідає за налаштування авто, світла фар при гальмуванні та задньому ходу, звуки авто і тд)

```

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

public class VehicleControl : MonoBehaviour
{
    public VehicleMode vehicleMode = VehicleMode.Player;

    // Wheels Setting //////////////////////////////////////

    public CarWheels carWheels;

    [System.Serializable]
    public class CarWheels
    {
        public ConnectWheel wheels;
        public WheelSetting setting;
    }

    [System.Serializable]
    public class ConnectWheel
    {
        public bool frontWheelDrive = true;
        public Transform frontRight;
        public Transform frontLeft;

        public bool backWheelDrive = true;
        public Transform backRight;
        public Transform backLeft;
    }

    [System.Serializable]
    public class WheelSetting
    {
        public float Radius = 0.4f;
        public float Weight = 1000.0f;
        public float Distance = 0.2f;
    }

    // Lights Setting //////////////////////////////////////

    public CarLights carLights;
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[System.Serializable]
public class CarLights
{
    public Light[] brakeLights;
    public Light[] reverseLights;
}

// Car sounds //////////////////////////////////////

public CarSounds carSounds;

[System.Serializable]
public class CarSounds
{
    public AudioSource IdleEngine, LowEngine, HighEngine;

    public AudioSource nitro;
    public AudioSource switchGear;
    public AudioMixerGroup brakeAudioMixer;
}

// Car Particle //////////////////////////////////////

public CarParticles carParticles;

[System.Serializable]
public class CarParticles
{
    public Material brakeParticleMaterial;
    public GameObject brakeParticlePrefab;
    public ParticleSystem shiftParticle1, shiftParticle2;
    private GameObject[] wheelParticle = new GameObject[4];
}

// Car Engine Setting //////////////////////////////////////

public CarSetting carSetting;

[System.Serializable]
public class CarSetting
{
    public bool showNormalGizmos = false;
    public Transform carSteer;
    public HitGround[] hitGround;

    public float springs = 25000.0f;
    public float dampers = 1500.0f;

    public float carPower = 120f;
    public float shiftPower = 150f;
    public float brakePower = 8000f;

    public Vector3 shiftCentre = new Vector3(0.0f, -0.8f, 0.0f);

    public float maxSteerAngle = 25.0f;

    public float shiftDownRPM = 1500.0f;
    public float shiftUpRPM = 2500.0f;
    public float idleRPM = 500.0f;

    public float stiffness = 2.0f;

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        public bool automaticGear = true;

        public float[] gears = { -10f, 9f, 6f, 4.5f, 3f, 2.5f };

        public float LimitBackwardSpeed = 60.0f;
        public float LimitForwardSpeed = 220.0f;

    }

    [System.Serializable]
    public class HitGround
    {

        public string tag = "street";
        public bool grounded = false;
        public AudioClip brakeSound;
        public AudioClip groundSound;
        public Color brakeColor;

    }

    // Camera Setting //////////////////////////////////////

    public CameraView cameraView;

    [System.Serializable]
    public class CameraView
    {
        public List<Transform> cameraSwitchView;

        public float distance = 8.0f;
        public float height = 2.5f;
        public float angle = 8;

    }

```

RunButton.cs

```

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class RunButton : MonoBehaviour
{
    public KeyCode keyPress;
    void Update()
    {
        if (Input.GetKeyDown(keyPress))
            transform.GetComponent<Button>().onClick.Invoke();
    }
}

```

					КН.КР.Б – 122 – 23 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		