ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»
Навчально-науковий інститут денної освіти
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(*підпис*)

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ІГРОВОГО ДОДАТКУ В СЕРЕДОВИЩІ UNITY 2D»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Шморгун Богдан Олегович
_____ «___»_____2023 р.
(підпис)

Науковий керівник к.ф.-м.н., доцент, Черненко Оксана Олексіївна
_____ «___»_____2023 р.
(підпис)

Рецензент

ПОЛТАВА 2023 р.

ЗМІСТ

АНОТАЦІЯ

Дана дипломна робота присвячена розробці симулятора гонок у вигляді гри на базі ігрового двигуна Unity та мовою програмування C#. Для поставленої мети було проведено аналіз жанру розбору ігор, проаналізовано досягнення представників даного жанру та дослідження різних середовищ розробки ігор та ігрових рухів. Результатом роботи є прототип двовимірної платформерної гри, у якій зустрічаються графічні, звукові та програмні елементи для роботи гри. У процесі розробки було розглянуто питання програмування руху гонщика, поведінки машин на трасі та управління камерою. Розроблено систему рівнів та прогресу гравця. Для пропозицій були використані інструменти, біржові ігрові двигуни Unity. Весь процес розробки проводився з використанням сучасних ескізів та концепцій у галузі розробки ігор. Результат роботи може бути використаний для постійного розвитку та удосконалення ігор у жанрі симуляторів перегонів. Для розробки алгоритму використовувалось середовище програмування Visual Studio.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Debug - це налагоджувач, комп'ютерна програма, яка використовується для тестування та виправлення помилок в інших програмах**.**

IDE - Integrated Development Environment - комплексне програмне рішення для розробки програмного забезпечення.

UML — Уніфікована мова програмування — уніфікована мова моделювання, що використовується в парадигмі об’єктно-орієнтованого програмування.

Актив – це графіка, звуковий супровід або компонент сценарію. Вони прилипають до предметів і є важливою частиною гри. Векторне зображення — це зображення, що складається з геометричних об’єктів (ліній, кіл, кривих тощо), що описуються математичними рівняннями та графічними примітивами.

ОС – це операційна система.

ПЗ – програмне забезпечення.

ПП – програмним продуктом.

Префаби — це набори готових ігрових об’єктів і компонентів, які використовуються кілька разів у грі.

Растрове зображення — це зображення, що складається з ряду пікселів, кожен піксель якого має певний колір.

Рендеринг, комп’ютерна візуалізація – це процес отримання зображення на основі моделі за допомогою комп’ютерної програми.

Сценарій — це невелика програма, яка містить ряд дій, призначених для автоматичного виконання завдання.

Спрайти - це двовимірні зображення, які використовуються в комп'ютерній графіці. У більшості випадків це растрове зображення, яке вільно переміщується по екрану.

ВСТУП

У XXI столітті - столітті інформаційного суспільства, люди отримують величезну кількість інформації, яку необхідно аналізувати та структурувати. Для подальшого використання отриманої інформації необхідно надати зберігання. Сьогодні мобільні телефони можуть допомогти вирішити цю проблему. Кінцеві пристрої, такі як смартфони, планшети, електронні книги та ноутбуки. Однак згадані портативні пристрої використовуються не тільки для роботи, але і для відпочинку. Через велике інформаційне навантаження заходи доводилося часто змінювати.

На допомогу у вирішенні цієї проблеми створено мобільні ігрові програми, що не несуть важкого смислового навантаження, тим самим сприяючи відпочинку людини та переключенню уваги з однієї діяльності на іншу. Виходячи з вищевикладеного, можемо зробити висновок, що ігри є одним з основних продуктів на ринку інформаційних продуктів та послуг, що дозволяє розробникам розширити коло вироблених додатків.

Також слід зазначити, що ігрові додатки роблять великий внесок у розвиток інформаційних технологій, наприклад, у розвиток програмного та апаратного забезпечення. Завдяки зростаючим вимогам користувача до якості ігор збільшуються вимоги до продуктивності пристроїв, що в свою чергу спричиняє розробку нових технологій. Саме тому розробка ігрових програм на сьогодні є найбільш затребуваною серед розробників мобільних додатків.

Мета дослідження – створення програмного продукту.

Задачі дослідження:

– аналіз предметної області;

– проектування концепту ігрового застосунку;

– вибір інструментарію, методів реалізації та тестування програмного продукту.

Об’єктом дослідження є процес аналізу мобільного геймінгу у жанрі симулятор гонок, вивчення їх ігрових механік.

Предметом дослідження є технології: ігровий рушій Unity, двовимірний графічний редактор Adobe Photoshop, 3D графічний редактор Blender, мова програмування C#.

Для досягнення мети та вирішення завдання реалізації мобільного ігрового додатку необхідно здійснити:

- опис предметного середовища;
- огляд наявних аналогів;
- постановку задачі;
- опис архітектури рішення, що розроблюється;
- опис коду та інтерфейсу програми;
- тестування

РОЗДІЛ 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Обґрунтування вибору середовищ розробки

1.1.1. Вибір ігрового рушія

Ігровий рушій — це центральна програмна частина комп'ютерної або мобільної відеоігри або іншої інтерактивної програми, яка обробляє графіку в реальному часі. Він надає основну технологію, спрощує розробку та дозволяє вашій грі працювати на різних платформах, таких як: ігрові консолі та настільні операційні системи.

Починаючи розробку продукту, виникає питання, в якому середовищі буде створюватися ігровий додаток. Ігрових движків для створення ігор безліч і з кожним роком їх кількість зростає. Деякі дозволяють розробляти 2D-ігри (движки, такі як CONSTRUCT 2, Corona тощо), інші дозволяють розробляти 3D (Unreal Engine, Unity, Godot). Деякі програмні продукти вимагають щомісячної або річної підписки, але доступні безкоштовні рішення. Деякі навіть дозволяють створювати кросплатформні ігри. Це означає, що коли ви створюєте продукт на одній платформі, ви можете перенести його на іншу платформу. Ігри, які створюють власний ігровий движок з таких причин, як небажання покладатися на технології інших компаній, бажання оптимізувати робочий процес ігор, що розробляються, або планування впровадження нових технологій, які не підтримуються існуючими технологіями.

Розглянемо середовище розробки ігрових програм, яке: Unreal Engine, Unity, Godot.

Unreal Engine був розроблений Epic Games. Unreal Engine дозволяє створювати ігри для більшості операційних систем, консолей і мобільних платформ. Завдяки підтримці різних систем відтворення графіки, підтримці різних систем відтворення звуку та можливостям мережевих ігор. За допомогою Unreal Engine можна створювати різноманітні ігри.

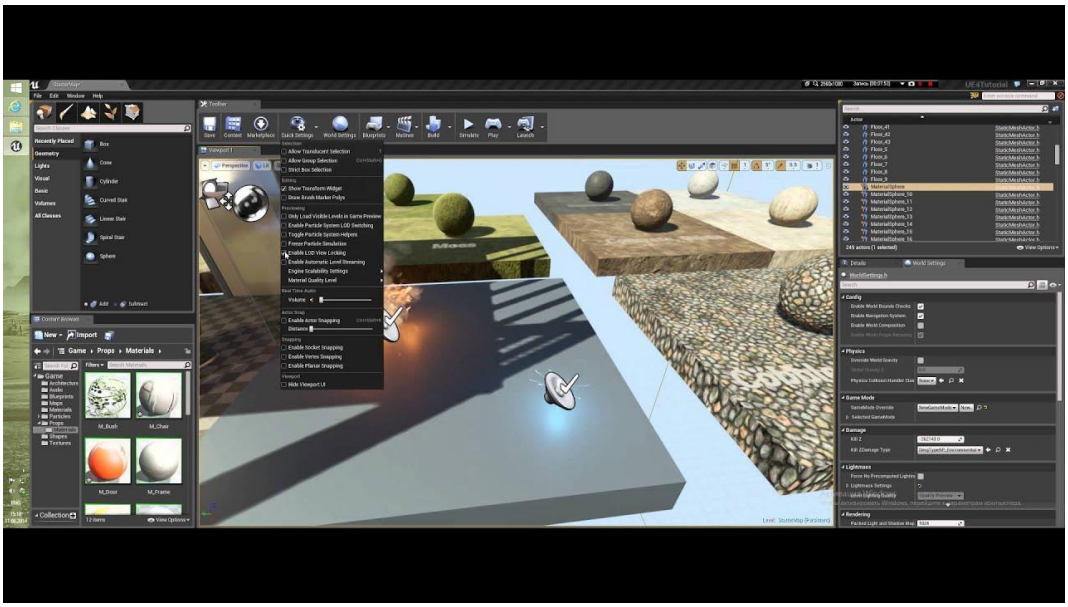

Рисунок 1.1 – Інтерфейс редактора Unreal Engine

Unity — це ігровий рушій, який допомагає створювати 2D і 3D ігри для ПК, Android, IOS, ігрових консолей і пристроїв VR. Він використовує підхід, орієнтований на компоненти, коли розробники створюють ігрові об’єкти (наприклад, головного героя) і додають до них різні компоненти (наприклад, візуальні ефекти персонажа та способи керування ними). Завдяки зручному інтерфейсу перетягування та функціональному графічному редактору цей движок дозволяє створювати карти, розміщувати об’єкти в реальному часі та негайно тестувати отримані результати.

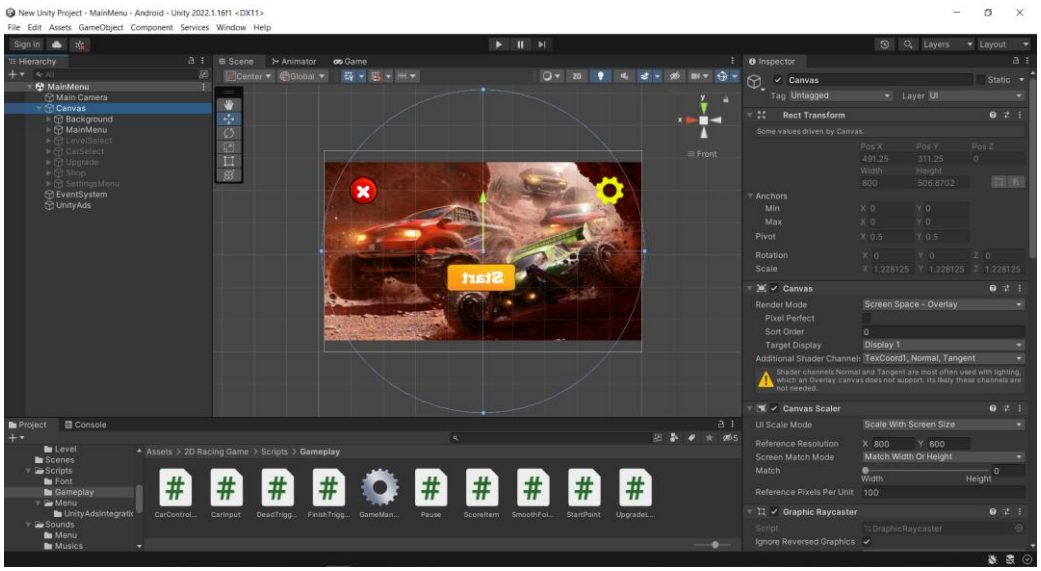

Рисунок 1.2 – Інтерфейс редактора Unity

Godot Engine — це універсальний кросплатформний ігровий рушій для створення 2D і 3D ігор за допомогою єдиного інтерфейсу. Він надає повний набір загальних інструментів, щоб користувачі могли зосередитися на створенні ігор. Ігри можна експортувати одним клацанням миші на різні платформи, включаючи основні настільні платформи (Linux, macOS, Windows), мобільні платформи (Android, iOS), веб-платформи (HTML5) і консолі.

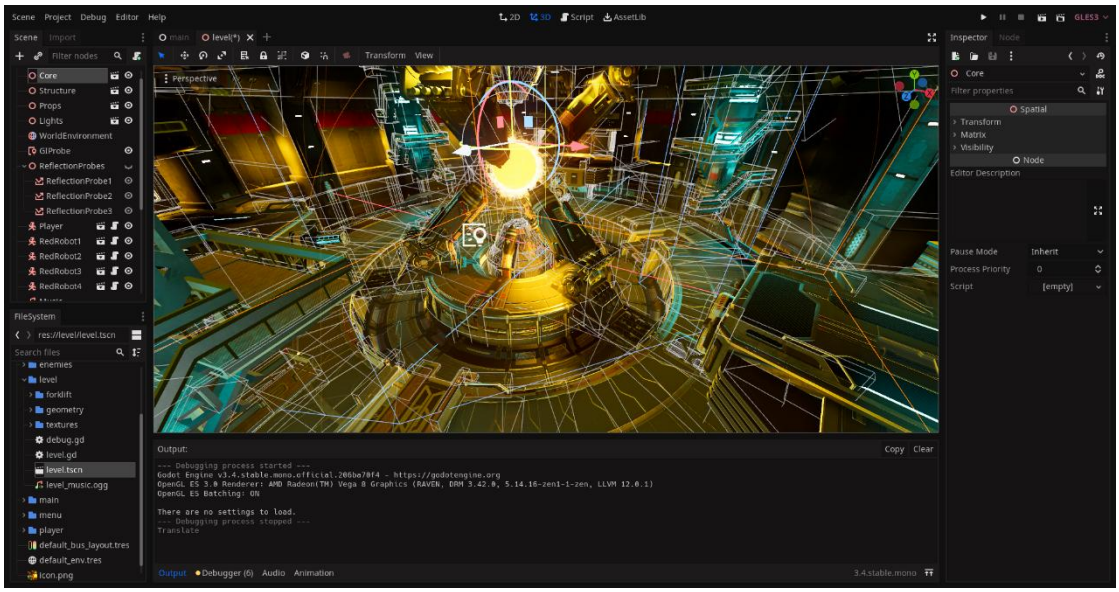

Рисунок 1.3 – Інтерфейс редактора Godot

Godot є повністю безкоштовним і відкритим вихідним кодом за безкоштовною ліцензією MIT. Ніяких зобов'язань, жодних комісій не потрібно. Розробка в Godot є повністю незалежною та керованою спільнотою, що дозволяє користувачам створювати механізм відповідно до власних очікувань. Unreal Engine непростий у вивченні та вимагає багато часу та ресурсів для роботи, тому ми зупиняємось між Godot Engine та Unity при виборі середовища розробки. Godot має багато переваг і є абсолютно безкоштовним і доступним інтерфейсом. Godot — це ігровий движок, який відносно простий у вивченні та використанні. З даним рушієм впораються як досвідчені розробники, так і початківці ентузіасти. Вбудована мова програмування GDScript має майже такий же синтаксис, як Python, і її легко

вивчити. Бар’єр входу також низький завдяки інтуїтивно зрозумілому редактору та можливості інтеграції з плагінами та програмами сторонніх розробників. Якщо подивитись на Unity, то ситуація зовсім інша. Щоб створювати ігри за допомогою ігрового движка Unity, потрібно вміти програмувати на C#. Це дуже потужна мова програмування, яка мала достатньо можливостей для виконання певних функцій. Це не одна з мов програмування низького рівня, але це не заважає вам впоратися з цим завданням. Зваживши всі плюси і мінуси обох двигунів, я вирішив вибрати Unity. Це дало загальні поняття, вправи з використання інтерфейсу користувача та подальший розвиток ігрового додатку.

1.1.2. Вибір середовища для розробки 2D графіки

Графічний редактор — це прикладна програма (пакет програм), яка дозволяє користувачам створювати та редагувати зображення на екрані свого комп’ютера та зберігати їх у форматах графічних файлів. Існує два типи графічних редакторів: ті, що спеціалізуються на створенні растрових зображень, і ті, які створені для роботи з векторними зображеннями. Для аналізу графічних редакторів було обрано три популярні програми.

Adobe Photoshop, Illustrator, GIMP. Adobe Photoshop — універсальний редактор для роботи з фотографіями (растрові зображення та деякі векторні інструменти). Цей програмний продукт працює на ПК з мобільними версіями операційних систем macOS, Windows, iOS та Android. Графічні редактори використовуються в комерційних цілях (телебачення, кіно, реклама, ігри, ретуш тощо). Adobe Illustrator — це універсальне векторне графічне середовище, яке містить неперевершений набір інструментів для створення логотипів, значків, малюнків та ілюстрацій для друку, Інтернету, відео та мобільних пристроїв.

GIMP — це кросплатформний редактор зображень, доступний для Linux, macOS, Windows та інших операційних систем. Це безкоштовне програмне забезпечення, яке дозволяє користувачам змінювати вихідний код і поширювати свої зміни.

Таблиця 1.1

Порівняльна таблиця графічних редакторів

Критерії	Програмне забезпечення		
	Photoshop	Illustrator	Gimp
Локалізація (українська мова)	Присутня	Присутня	Присутня
Вартість продукту	Безкоштовний пробний період	Безкоштовний пробний період	Безкоштовна
Можливість роботи з векторними зображеннями	Обмежена	Присутня	Присутня
Можливість роботи з растровими зображеннями	Присутня	Обмежена	Обмежена
Можливість постобробки зображення	Присутня	Обмежена	Обмежена

Після аналізу трьох графічних редакторів (табл. 1.1) було обрано Adobe Photoshop. Гарне програмне забезпечення для графічного дизайну, редагування зображень і ретуші фотографій. Програма має широкий набір функцій і спеціалізується на обробці растрових зображень, які використовуються ілюстраторами для створення графіки.

1.1.3. Вибір середовища для розробки 3D моделей

Кваліфікаційна робота потребує середовища для моделювання 3D-графіки. Серед багатьох програм для 3D-дизайну слід вибрати правильну програму, яка відповідає декільком потребам.

Підтримка імпорту моделей в Unity, низькі вимоги до характеристик ПК, зручний і зрозумілий інтерфейс. Серед наявних програмних продуктів, які можуть підійти під вимоги до проєкту, можна виділити Blender та 3ds Max.

3ds Max — тривимірний графічний редактор, повнофункціональний професійний додаток для створення та редагування об’єктів і створення візуалізацій, розроблений Autodesk. Містить найсучасніші інструменти для архітекторів, дизайнерів, художників і мультимедійних професіоналів. Працює з операційними системами Microsoft Windows і Mac OS.

3ds Max використовується для візуалізації моделей будівель, комп’ютерних ігор, 3D-мультфільмів, рекламних роликів тощо. За допомогою цього редактора було створено багато анімованих моделей для фільмів.

Blender — безкоштовний програмний продукт для створення та редагування 3D-графіки. Програма поширюється на всіх основних платформах, використовує відкритий вихідний код і є абсолютно безкоштовною для бажаючих. Є й українська версія.

Ці особливості роблять його надзвичайно популярним як серед початківців, так і серед справжніх професіоналів модельного бізнесу. Дизайнери часто вибирають Blender як основний робочий інструмент для великих серйозних проектів.

Таблиця 1.2

Порівняльна таблиця ПЗ для тривимірної графіки

Критерії	Програмне забезпечення	
	Blender	3ds Max
Системні характеристики	Середні	Високі
Локалізація (українська мова)	Присутня	Немає
Вартість продукту	Безкоштовний	Платний
Сумісність з Unity	Присутня	Присутня

Швидкість рендерингу	Швидша, порівнюючи з 3ds Max	Повільніша, порівнюючи з Blender

Кожен редактор 3D-графіки має власні властивості та унікальний набір властивостей. Бізнес-інструменти не завжди кращі за безкоштовні, але навпаки.

Проаналізувавши вимоги, потреби та доцільність, ми створюємо модель середовища та маршрут за допомогою Blender, програмного забезпечення для створення та редагування тривимірної графіки (табл. 1.2). Blender є досить потужним 3D-редактором і знаходиться в стадії активної розробки. Чудова альтернатива вже дорогим програмам і ідеально справляється з поставленим завданням.

1.1.4. Висновки

Враховуючи вищезазначені програми та їхні переваги, найкращим поштовхом для створення продукту є використання Unity з мовою програмування C#.

Програмний код розробляється в середовищі Visual Studio. Adobe Photoshop використовується для створення 2D-графіки, тоді як Blender і Unity Asset Store використовуються для 3D-сцени та моделі автомобіля. Короткий порівняльний аналіз обраного програмного забезпечення наведено в таблиці 1.3.

Таблиця 1.3

Порівняльний аналіз програмних продуктів для розробки мобільного додатку

Категорія	**Варіанти програм**	**Висновок**
Ігровий рушій	Unity, Godot, Unreal Engine	З Unity ви можете розробляти ігри без спеціальних знань. Завдяки зручному інтерфейсу перетягування цей механізм дозволяє малювати карти та розміщувати об’єкти в реальному часі. Наявність бібліотеки активів і плагінів, які можуть значно прискорити процес розробки гри

Мова програмування	C#, C++, JavaScript	Ігровий рушій визначив мову програмування (C#)
Середовище розробки програмного коду	Microsoft Visual Studio, VS Code	Visual Studio — це IDE для роботи з мовою програмування C#. IntelliSense та навігація кодом дозволяють легко переміщатися між сценаріями, а Debug допомагає швидко знаходити проблеми.
Створення 2D графіки	Adobe Photoshop, Gimp, illustrator	Photoshop - найпопулярніший редактор для двовимірної графіки. За допомогою його інструментів ви можете редагувати наявні зображення, застосовувати фільтри або малювати з нуля. Підтримуються численні формати, які використовує Unity.
Створення 3D моделей	Blender, 3ds Max	Blender — це безкоштовна програма, яка містить основні інструменти, необхідні для створення 3D-моделей, і не потребує великих ресурсів комп’ютера.
Додаткові ресурси	Unity Asset Store	Unity Asset Store — це постійно зростаюча бібліотека активів. Доступна велика різноманітність ресурсів, від текстур, анімацій і моделей до готових зразків проектів, навчальних посібників і розширень редактора.

1.2. Опис предметного середовища

Unity – кросплатформенний рушій розробки комп'ютерних ігор. Unity дозволяє створювати програми, що працюють під більш ніж 20 різними операційними системами, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші. Як правило, Unity надає безліч функціональних можливостей, що дозволяють їх задіяти в різних іграх, в які входять

моделювання фізичних середовищ, карти нормалей, динамічні тіні та багато іншого. На відміну від багатьох ігрових рушіїв, у Unity є дві основні переваги:

наявність величезної бібліотеки ассетів та плагінів та кросплатформенна підтримка.

Перший фактор включає не тільки інструментарій візуального моделювання, а й інтегроване середовище, ланцюжок складання, що спрямоване на підвищення продуктивності розробників, зокрема етапів створення прототипів та тестування. Кросплатформна підтримка означає не лише розташування (встановлення на персональних комп'ютерах, мобільних пристроях, консолях тощо), а й наявність засобів розробки (інтегровані середовища доступні для Windows і macOS).

Недоліками є: відсутність підтримки Unity для підключення до зовнішніх бібліотек, робота, яку програмісти повинні налаштовувати окремо, робота, яка ускладнює командну роботу, робота в багатокомпонентних схемах і труднощі з підключенням зовнішніх бібліотек.

1.3. Огляд існуючих платформ

Наявність операційної системи (ОС) є головною особливістю, яка відрізняє смартфон від звичайних мобільних телефонів. Операційна система часто є вирішальним фактором при виборі конкретної моделі телефону або пристрою.

Найпопулярнішими операційними системами для мобільних пристроїв сьогодні є Android та iOS.

Android – це мобільна операційна система, яка існує вже більше 15 років. Користувачі впізнають цю платформу як основну операційну систему для мобільних телефонів і планшетів у всьому світі. Крім того, існують інші операційні системи, які підтримують програми Android, як-от ОС Chrome і Windows 11.

На даний момент Android є найпопулярнішою операційною системою в світі. За оцінками, він працює на 2,5 мільярдах активних пристроїв у всьому світі та понад 3 мільярдах користувачів, або приблизно 39% від загального населення світу. Це значно випереджає iOS від Apple, другу за популярністю операційну систему у світі. Переваги розробки мобільних додатків під ОС Android:

– швидка публікація та менше обмежень;

– більше користувачів і завантажень;

– ефективна монетизація за допомогою реклам.

Недоліки розробки мобільних додатків під ОС Android:

– довший час розробки;

– нижчий середній дохід;

– більше можливих помилок.

iOS є однією з найпопулярніших мобільних операційних систем, розроблених і створених Apple Inc. Пристрій iOS — це електронний гаджет, який працює на iOS. Пристрої Apple iOS включають: iPad, iPod Touch, iPhone. iOS є другою за популярністю операційною системою після Android. Протягом багатьох років пристрої Android та iOS вели жорстку конкуренцію за частку ринку.

Переглянемо переваги розробки мобільних додатків під iOS:

– вищий середній дохід;

– швидший розвиток;

– менше можливих помилок.

Недоліками ж є:

– у два рази менше користувачів;

– тривала публікація з багатьма обмеженнями;

– необхідність придбання спеціального обладнання.

В App Store (платформа для пошуку та завантаженню додатків для iOS пристроїв), щоб мати змогу публікувати додатки потрібно зробити грошовий внесок у розмірі 99$. Реєстрація фізичної особи, яка в майбутньому викладатиме додатки у вільний доступ, складає від 3 тижнів. Далі розробника чекають довгі та не зовсім прості процеси.

Ігри також неможливо опублікувати безкоштовно на платформі Android. Таким чином, розробник повинен сплатити одноразову комісію розробнику в

розмірі 25 доларів США. Однак у Google набагато менше вимог до програм, які він публікує, ніж у Apple.

Варіанти монетизації також важливі. Це набагато простіше для пристроїв під управлінням ОС Android. Враховуючи кількість пристроїв під цією платформою, можна з упевненістю сказати, що реклама приносить більший дохід.

Розробка додатків для Android та iOS має плюси та мінуси, залежно від цілей, які хоче переслідувати розробник, цільової аудиторії та варіантів монетизації.

У кваліфікаційній роботі вирішальними стали пункти: «тривала публікація з багатьма обмеженнями» та «необхідність придбання спеціального обладнання». Так як, на останньому етапі розробки кваліфікаційна робота повинна бути кінцевим робочим проектом, який буде можливим завантажити у будь-яку крамницю для того, щоб користувачі могли мати вільний та легкий доступ для завантаження додатку на свій прилад, потрібно визначитись з операційною системою мобільного пристрою.

Розглянувши і проаналізувавши дві найбільш популярні платформи для розробки мобільних додатків, можна зробити висновок, що Android є найбільш підходящою операційною системою для створення програм з наступних причин: − посідає перше місце за поширеністю на ринку мобільних пристроїв; − дозволяє завантажувати програми з інших джерел, крім офіційного магазину програм; − обліковий запис розробника має найнижчу вартість.

1.4. Огляд наявних аналогів

На перших етапах розробки власного ігрового додатку необхідно проаналізувати схожі мобільні додатки їх механіки, цілі, дизайн. До уваги були взяті наступні додатки:

– сцена Level

– сцена Vehicle

– сцена завантаженої локації по якій їздить гравець

iRacing - це гоночна гра в жанрі симулятора, яка дозволяє зануритися в атмосферу подорожі різними локаціями по складній місцевості. У цій класичній 2D-грі є все: просте управління, ігровий принцип «згори донизу», мультяшний дизайн, класичний саундтрек.

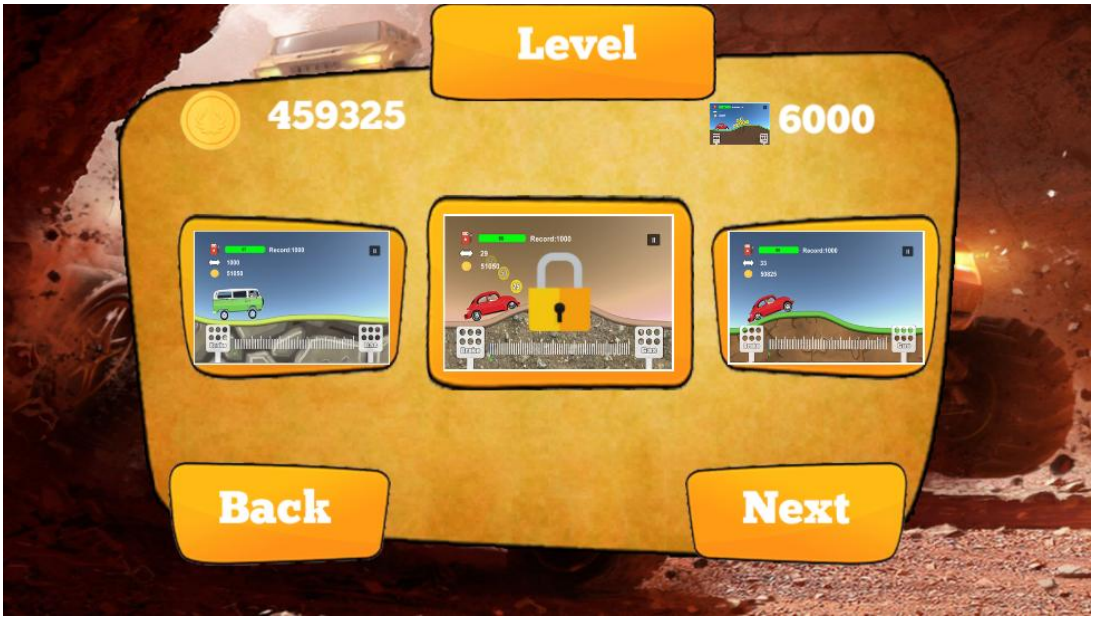

Рисунок 1.4 – Процес гри iRacing в сцені Level

iRacing – це гра, в якій гравцю доведеться їздити по складній поверхності на різних локаціях. В процесі гри потрібно доїхати до фінішу збираючи на своєму шляху монетки для купівлі нових локацій і машин. Також потрібно збирати пальне щоб продовжити свій шлях в захоплюючій грі.

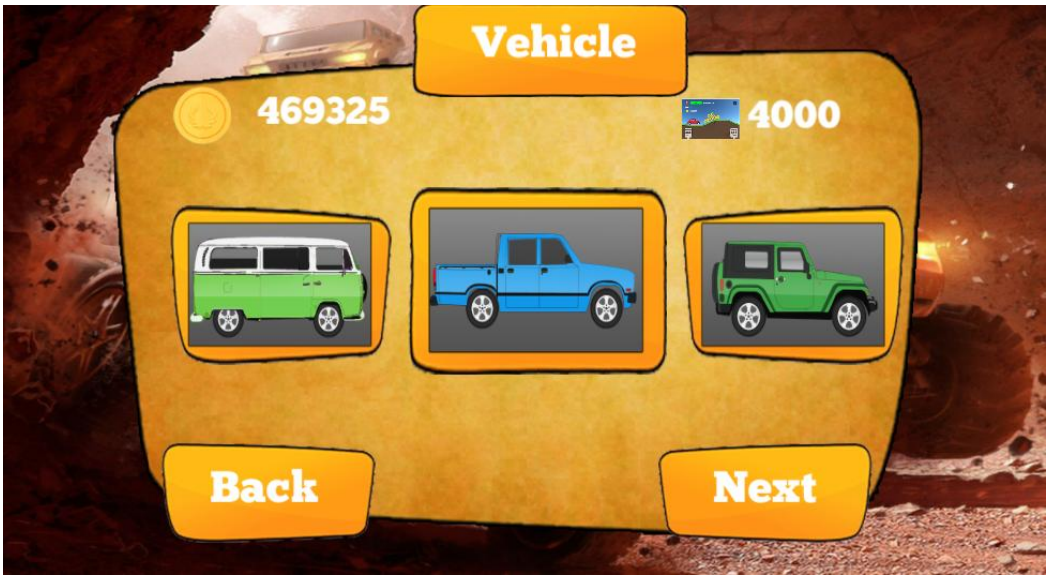

Рисунок 1.5 – Процес гри в вибір машини в меню Vehicle

iRacing містить усі види автомобілів, кожна з яких ефективна для певної локації. Гравець може купувати автомобілі магазині і обирати улюблену в гаражі, досліджувати локації милуючись краєвидами і відкривати для себе нові краєвиди.

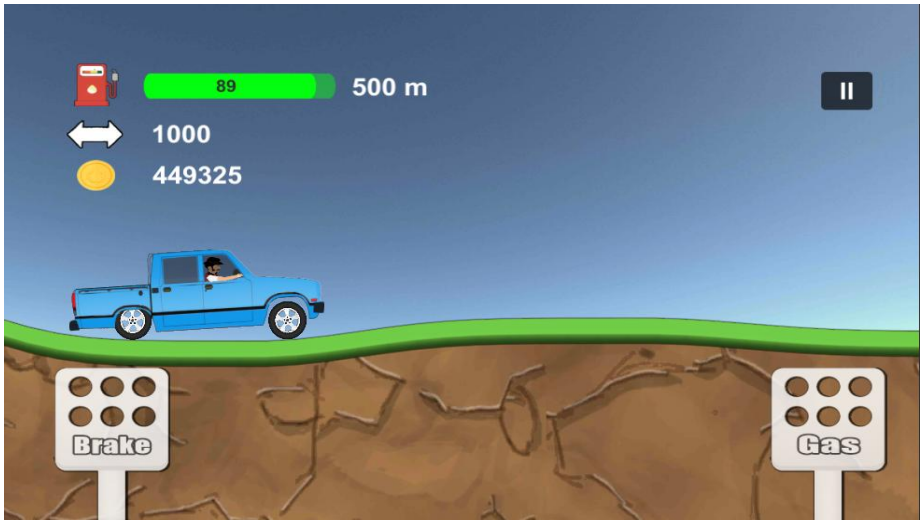

Рисунок 1.6 – Процес гри на завантаженій локації

Висновок до розділу 1

У першому розділі розглядався вибір ігрового рушія, середовища розробки 2D і 3D моделей, тематичні середовища, перевірялися існуючі платформи розробки та доступні аналоги.

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області

Жанр гонок досить популярний у сфері мобільного геймінгу. Загальні механіки успішно реалізуються на кожній з платформ, але спосіб реалізації може відрізнятися. Розглянувши популярних представників ігор жанру симулятор перегонів, були обрані основні та допоміжні механіки гри, стилі зовнішнього вигляду та предметів оточення для майбутнього додатку. Також обрано вигляд камери типу TopDown для спрощення ігрового процесу, адже гравцю так буде зручніше контролювати оточення та опонентів.

Комп'ютерні симулятори є гонками найпопулярніших жанрів серед гравців усього світу. Вони дозволяють користувачам відчути від керування автомобілем на швидкості, відчути адреналін, який відбувається в процесі і навіть розвиває навички водіння. Жанр симулятора перегонів з'явився ще у 70-х роках минулого століття та з технічною порцією значного розвитку.

Однією з особливостей комп'ютерних ігор-симуляторів є реалістична дія руху автомобіля. Гравці отримують можливість набирати параметри автомобіля, брати участь у різних чемпіонатах та змаганнях, а також розвивати свої навички в різних умовах. Симулятори гонок використовують різні фізичні моделі, які відтворюють деталізовані характеристики автомобілів та ландшафти, що робить ігровий процес максимально реалістичним.

Однією з найпопулярніших серій гонок симуляторів є Gran Turismo, випущена в 1997 році. Ця гра стала неймовірно реалістичною геймплею та неймовірною графікою. Gran Turismo зібрала представників найпопулярніших симуляторів по всьому світу і досі залишається одним із найпопулярніших симуляторів.

Ще одним із популярних симуляторів гонок є Forza Motorsport. Ця серія була створена компанією Turn 10 Studios та з'явилася ексклюзивною ігровою консолею Xbox. Forza Motorsport також була визнана гравцями завдяки своїй реалістичності та чудовій графіці.

Симулятори гонок також широко використовуються у професійному спорті, де вони допомагають гонщикам тренуватися та розвивати навички. У симуляторах гонок використовуються точні моделі автомобілів та треків, що дозволяє гонщикам зануритися повністю з головою в цей режим.

2.2. Постановка задачі

В результаті аналізу теми були сформульовані цілі кваліфікаційної роботи. Я розробив мобільний гоночний симулятор «iRacing», щоб гравці могли відчути емоції симуляторних ігор, прискорити рефлекси та покращити координацію рухів, а також відпочити.

Для досягнення мети кваліфікаційної роботи був визначений перелік основних завдань:

– виконати аналіз предметної області та огляд наявних аналогів;

– визначити інструментарій розробки;

– спроектувати функціонал додатку;

– розробити дизайн ігрового продукту;

– розробити механіки;

– налаштувати сцени та розробити ігрові рівні;

– протестувати ігровий додаток.

Мобільний додаток повинен забезпечувати корисне проведення часу в грі та мати можливість покращувати рефлекси в грі.

2.3. Розробка архітектури ігрового додатку

Створення ігрового додатку дуже трудомісткий і складний процес. Даний процес об’єднує у собі величезну кількість елементів у єдину програму, призначену для того, щоб розважати геймера.

Для гри необхідно реалізувати модуль, який відповідає за ігрову логіку і фізику, який буде управляти взаємодією ігрових об'єктів на сцені, перемикати сцени в залежності від обраного пункту меню, завантажувати карти рівнів і так далі.

Розгорнутий опис функціональних вимог здійснюється на етапі проектування програми. Для того, щоб деталізувати вимоги, необхідно виділити процеси, що відбуваються в заданій предметній області. Опис таких процесів на UML виконується у вигляді прецедентів (рис. 2.1). Прецеденти є сценарієм або варіантом використання ігрового програми при взаємодії із зовнішнім середовищем. За допомогою прецедентів описується функціонування гри з точки зору користувача, який називається в UML актором (Гравець). Актор є будь-якою зовнішньою по відношенню до модельованої системі сутністю: людина, штучний інтелект, яка взаємодіє з системою і використовує її функціональні можливості для досягнення певної мети.

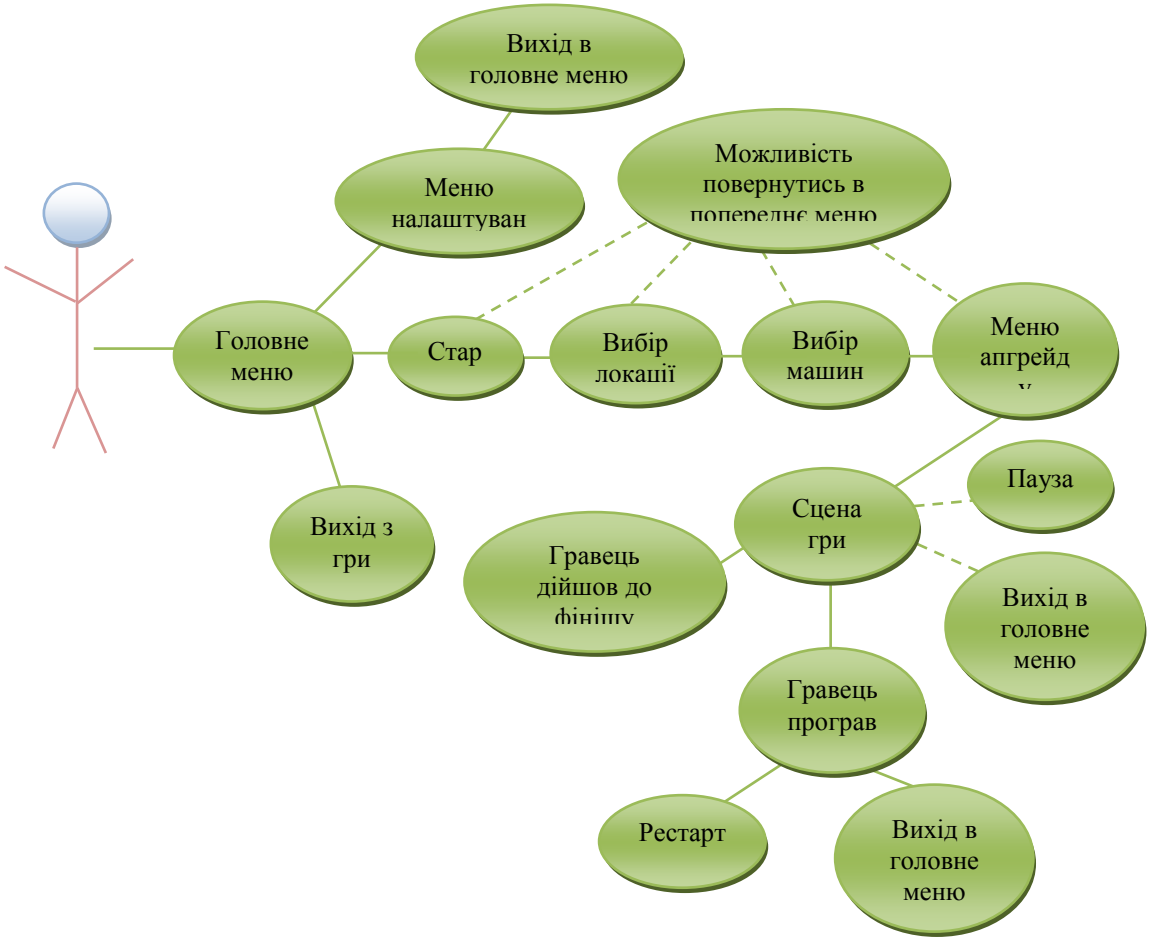

Рисунок 2.1 – Діаграма прецедентів

2.4. Game Loop

Ігровий цикл — це система, яка послідовно викликає модулі в правильному порядку. Ігровий процес оформлений у вигляді схем, реалізований у скриптах і підключений до MainCamera.

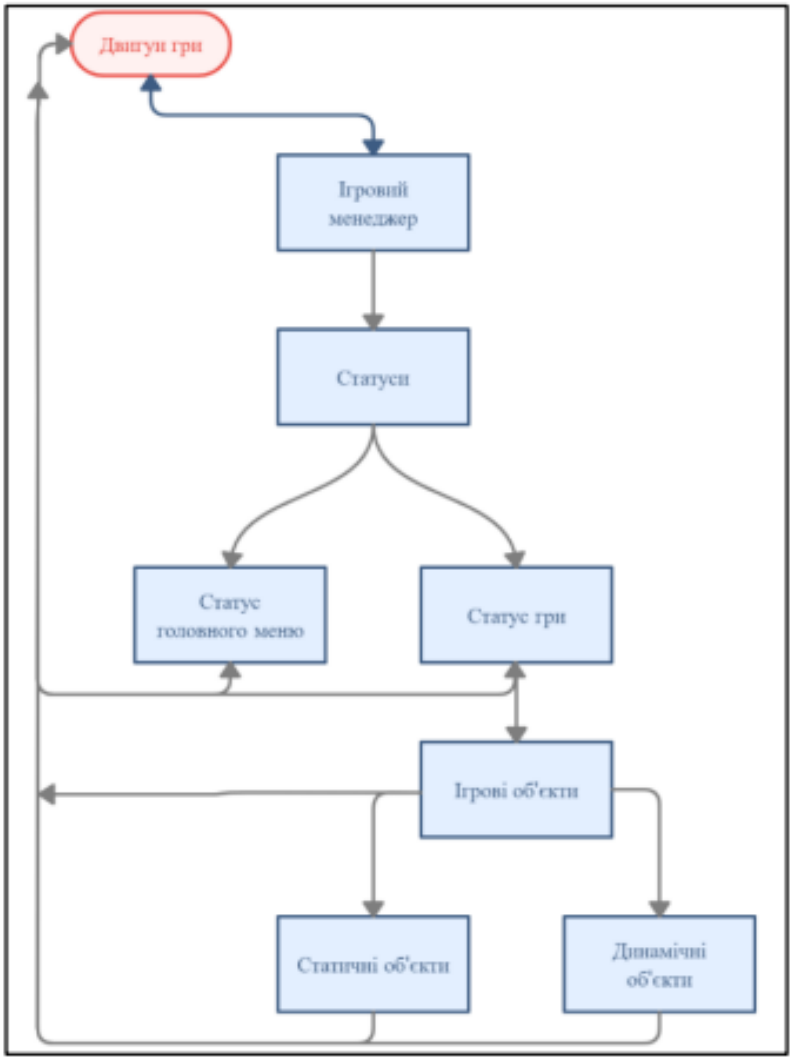

Рисунок 2.2 – Діаграма iRacing

2.5. Проектування інтерфейсу гри

В результаті аналізу предметної області та вимог, був розроблений інтерфейс ігрового додатка. Кожне меню або ігровий режим є окремим станом, який має свій клас, призначений для його перемикання. Основне ігрове поле (рис. 2.3) містить ігрові об'єкти логіки представлення. На екрані присутні: шкала запасу пального, відстань (яку проїхав гравець), кількість монет зібраних гравцем за весь час, меню

паузи, дві педалі для керування транспортним засобом, візуально представлена карта зі складним рельєфом на якому розташовані монети і каністри з бензином які потрібно підбирати.

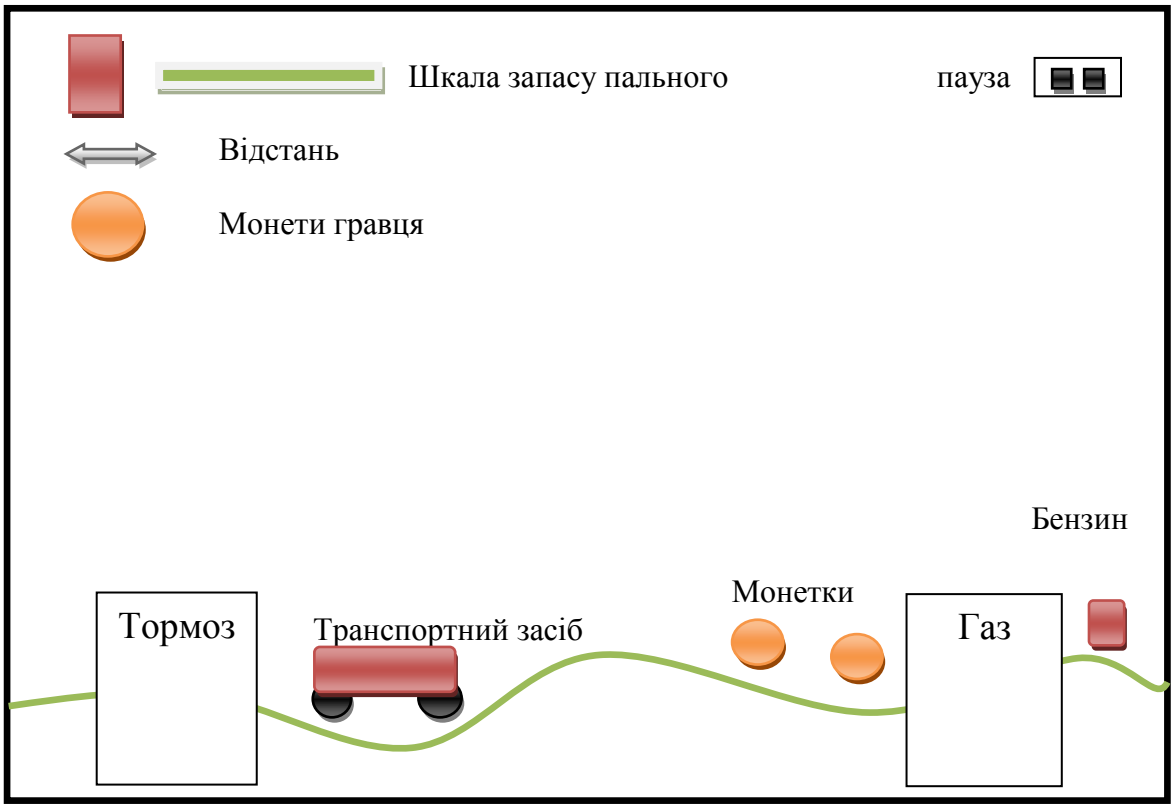

Рисунок 2.3 – Прототип інтерфейсу ігрової сцени

2.6. Система меню

Кожна гра починається з головного меню. Дана система дозволяє гравцеві запускати гру, виходити з програми. Для її проектування вкрай зручно використати діаграму станів.

Головне меню складається з:

– Старт

– Вибір локацій

– Вибір машин

– Меню апгрейду машини

– Завантаження сцени з грою

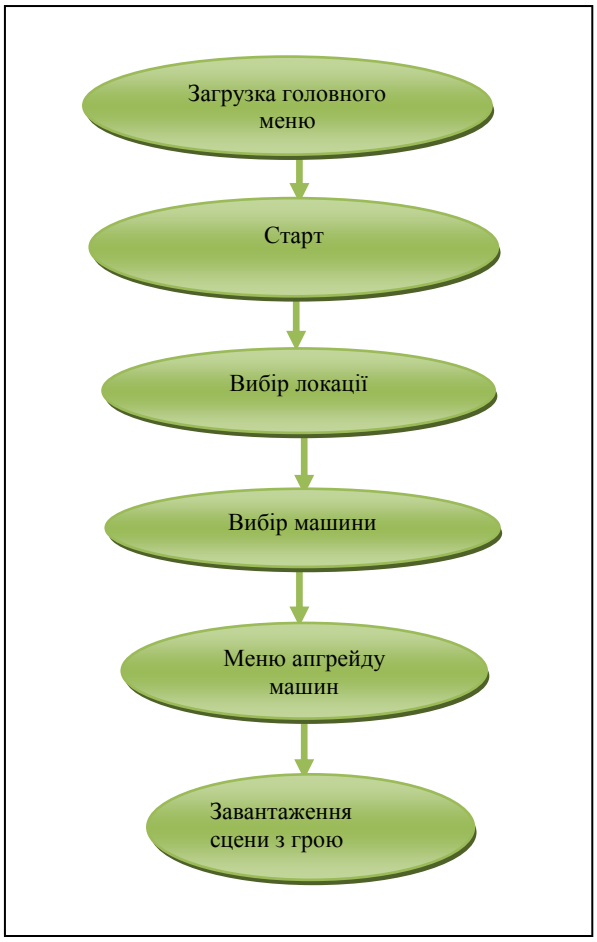

Рисунок 2.4 – Діаграма станів – Старт гри

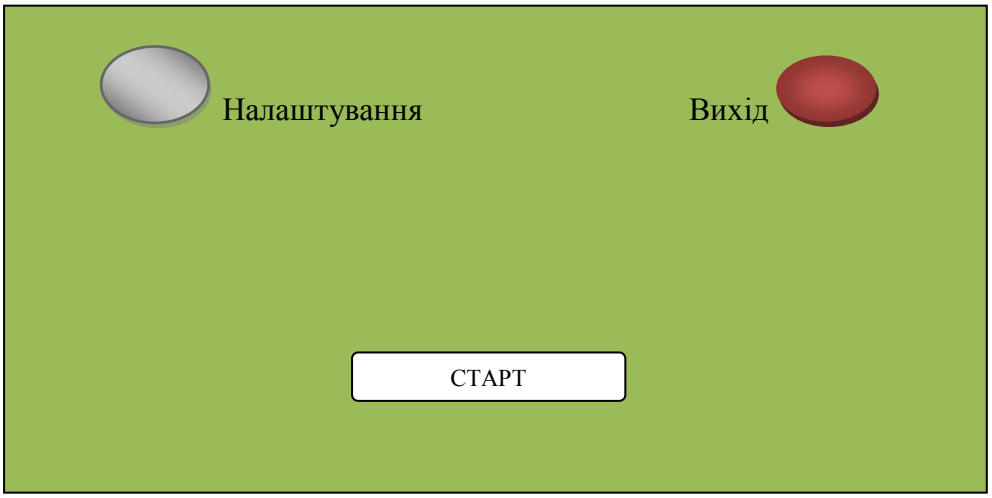

Рисунок 2.5 – Прототип головного меню гри

Висновок до розділу 2

У другому розділі було проведено аналіз предметної області, підготовлено постановку задачі, виконано розробку архітектури ігрового додатку, яка включає: ігровий цикл, проектування прототипів інтерфейсів додатку.

РОЗДІЛ 3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Засоби розробки містять технічну документацію для таких середовищ: Unity, Microsoft Visual Studio, Adobe Photoshop і Blender призначені для роботи з інтерфейсами і бібліотеками, які використовуються при створенні ігрових продуктів. Публікації та інтернет-статті про відеохостинг YouTube.

3.2. Вимоги до технічного та програмного забезпечення

Для створення ігрового застосунку впроваджено наступні вимоги до програмного забезпечення:

– Unity – ігровий рушій;

– C# – мова програмування;

– Microsoft Visual Studio – середовище програмування;

– Unity Asset Store – бібліотека моделей та плагінів;

– Adobe Photoshop – створення двовимірної графіки;

– Blender – створення тривимірної графіки.

Вимоги до технічного забезпечення пристрою для розробки:

– Процесор: Intel Pentium Silver N 5030

– Відеокарта: Intel UHD Graphics 605

– Оперативна пам'ять: 8 ГБ

– Операційна система: Windows 10, 64-розрядна

Вимоги до технічного забезпечення мобільного пристрою:

– Процесор: 4-ядерний

– Оперативна пам'ять: 2 ГБ

– Місце на диску: 100 МБ

– Операційна система: Android 6.0

3.3. Опис програмної реалізації

3.3.1. Створення меню

Створення меню в Unity займає трохи часу, оскільки нам потрібно створити іншу сцену під назвою MainMenu. Сцена MainMenu містить налаштування камери, саме головне меню та інші параметри гонки (вибір автомобіля та траси), звуковий супровід і сценарій RaceInfoManager, який буде пояснено пізніше.

Рисунок 3.1 – Сцена головного меню

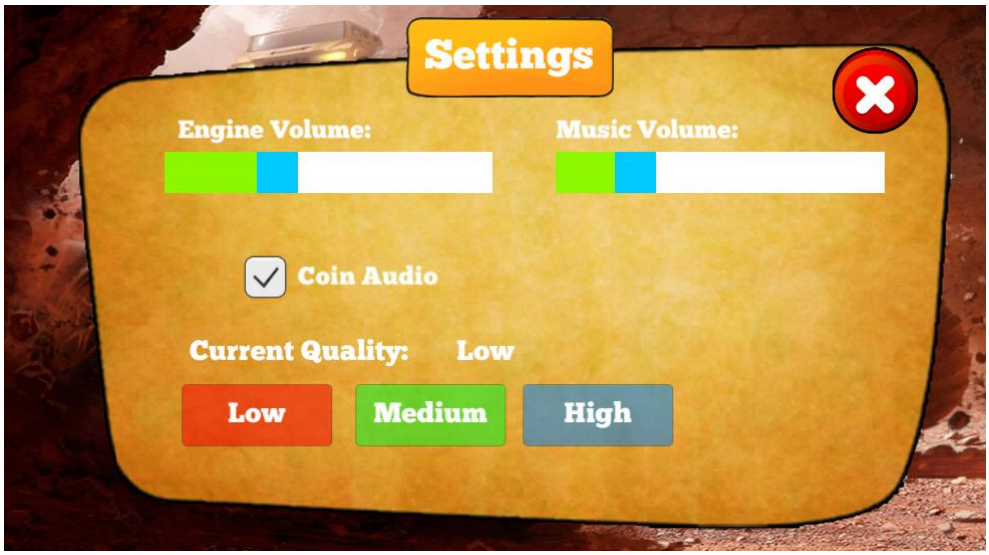

Рисунок 3.2 – Меню налаштувань

Рисунок 3.3 – Меню придбання і вибору локацій

Рисунок 3.4 – магазин для придбання і вибору машини

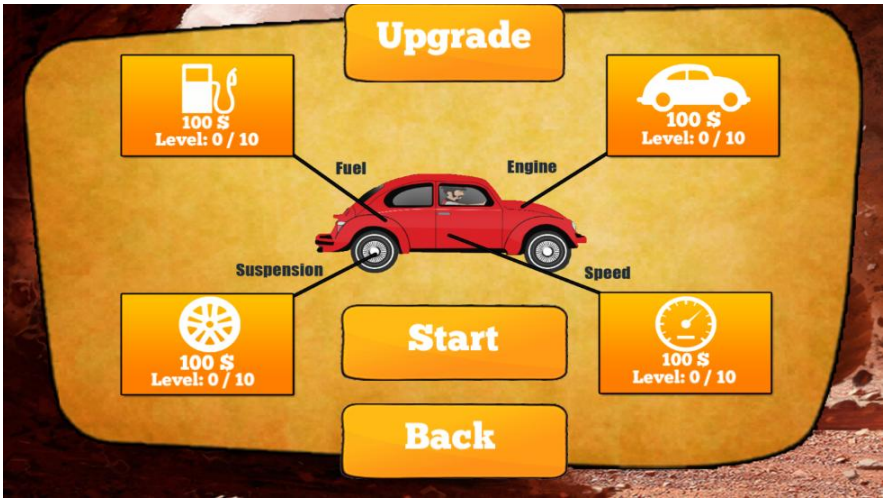

Рисунок 3.5 – меню апгрейду

Після вибору автомобілю, та треку, натискаємо кнопку «START» у меню Race Setup, після чого автоматично включиться наступна сцена.

Скрипт MainMenu має такі елементи:

– вибір траси;

– вибір авто;

– зображення відповідних елементів.

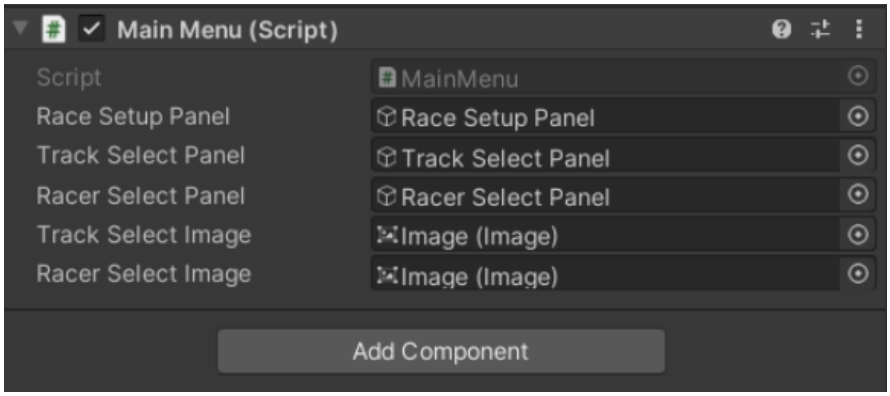

Рисунок 3.6 – MainMenu

3.3.2. Моделювання тестового треку

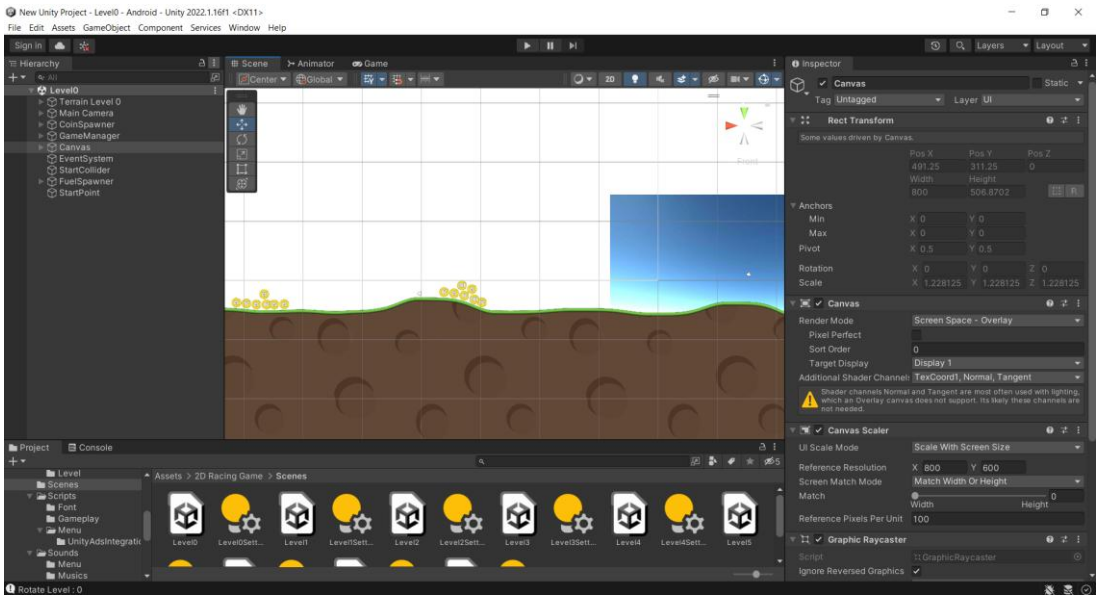

Рисунок 3.7 – Сцена рівня

Дана сцена розроблена у початковому форматі. Трек, на якому розміщені стартова позиція на якій і з’являється машина гравця, далі гравцю потрібно проїхати складний шлях, бо рельєф не простий і доїхати до фінішу.

3.3.3. Інтерфейс та органи керування

На рисунку 3.8 продемонстровано інтерфейс органів керування, а також ігрової статистики (кількість монет, запас бензину і пройдена відстань гравцем).

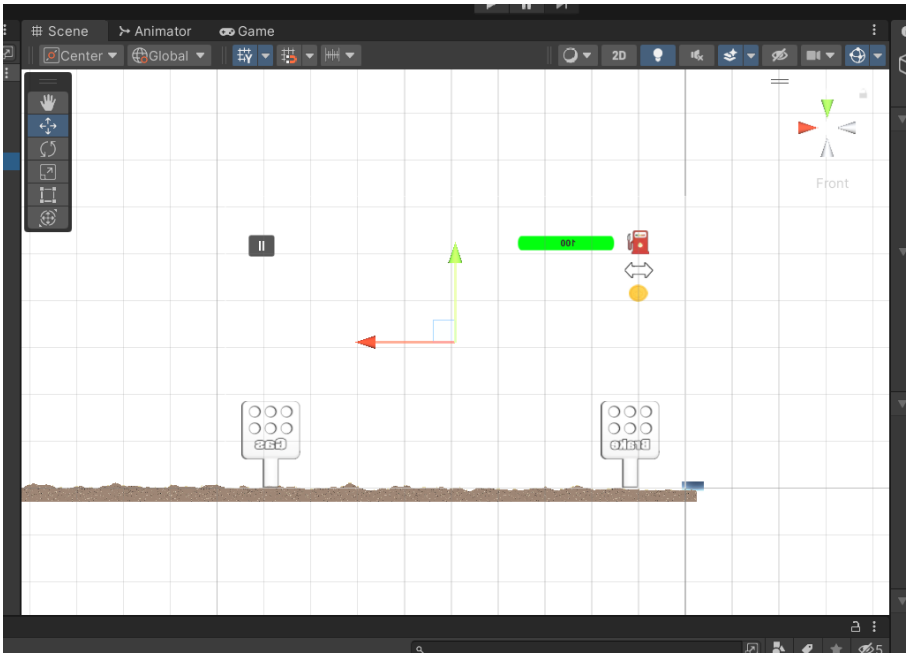

Рисунок 3.8 – Інтерфейс ігрової сцени

Даний інтерфейс не буде працювати без відповідних скриптів, тому було створено UI Manager, який обробляє всю статистику (рис. 3.8).

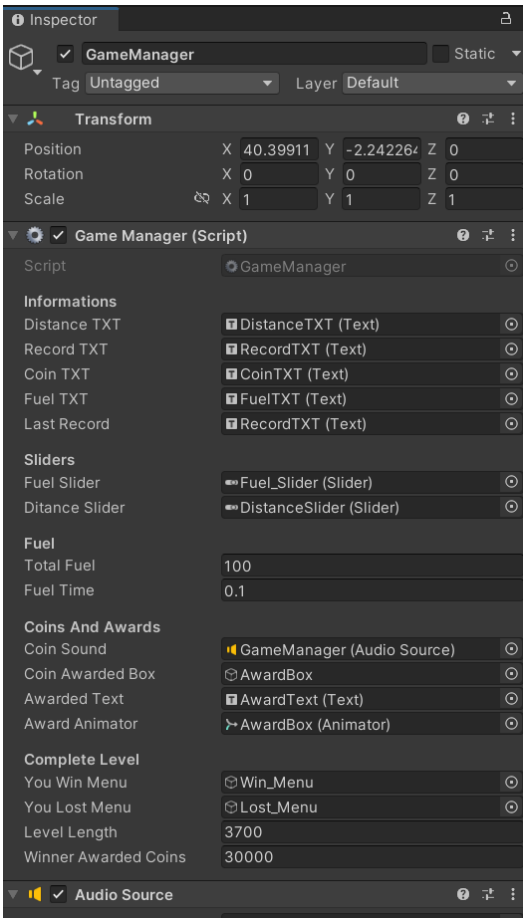

Рисунок 3.9 – Робота скрипту UI Manager

Також на зображеному рисунку інтерфейсу ігрової сцени містяться елементи керування автомобілем, і кнопка «Pause». Ці органи керування входять до безкоштовного пакету ассетів Simple Input у магазині Unity AssetStore.

3.3.4. Розробка CarController

Щодо управління автомобілем, написання та реалізація скрипту CarController була трудомісткою та зайняла досить багато часу. Префаб моделі авто (рис. 3.10) включає тривимірну колізію, аудіо супроводження (звуки двигуна, зіткнення з монетами і каністрою), а також грязь, який формується під час пробуксовки.

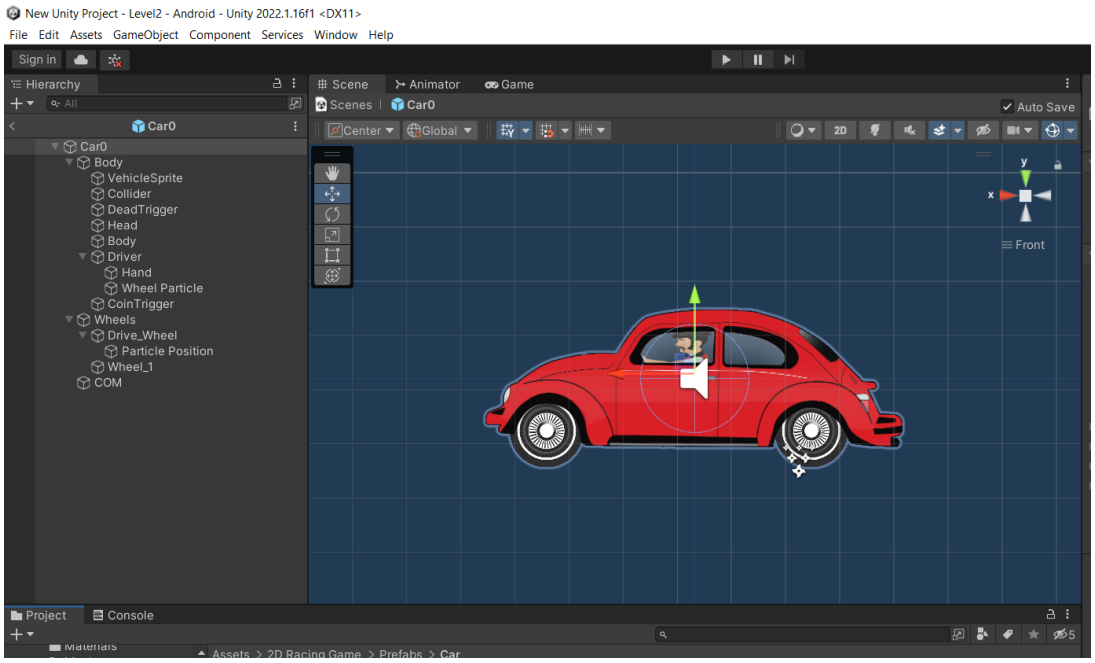

Рисунок 3.10 – Префаб автівки

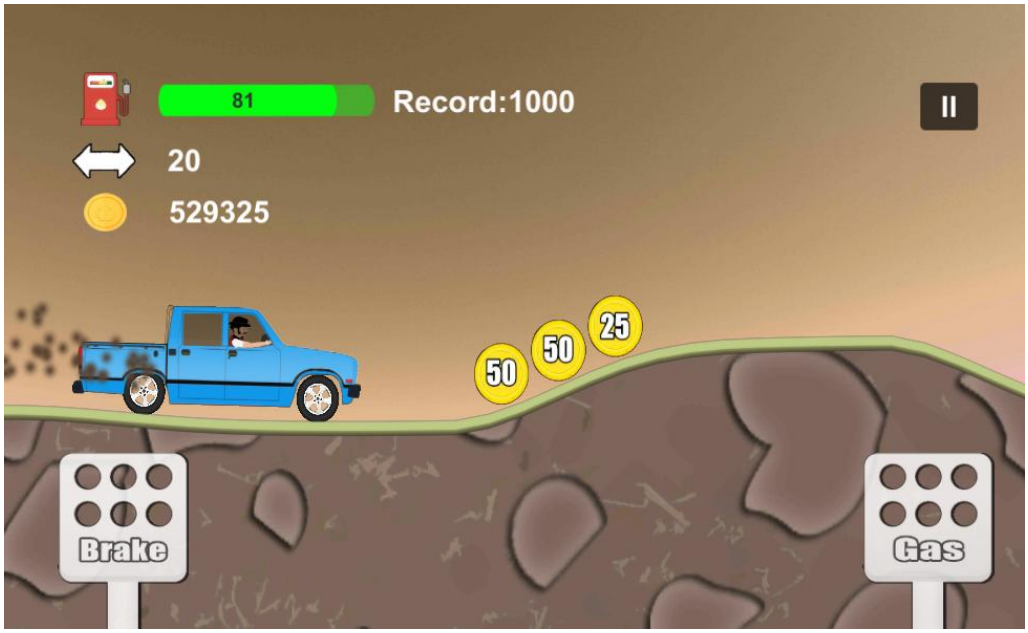

Рисунок 3.11 – Реалізація диму з коліс

Цей скрипт містить звичайний код для контролера автомобіля. У коді автомобільний об’єкт просто рухається відповідно до заданих введень. Контролер під’єднаний до органів керування екранних кнопок: газ і гальмо.

Функції скрипту:

– Реалізація керування автомобілем;

– Налаштування параметрами автомобіля (максимальна швидкість, пробуксовка, швидкість заднім ходом, рівень гравітації і так далі);

– Налаштування оберту коліс навколо своєї осі (колеса змінюють свій кут, коли вони змінюють напрямок);

– Контроль та налаштування (налаштування базової максимальної швидкості, прискорення та максимальний кут повороту;

– Аудіо супроводження автомобіля.

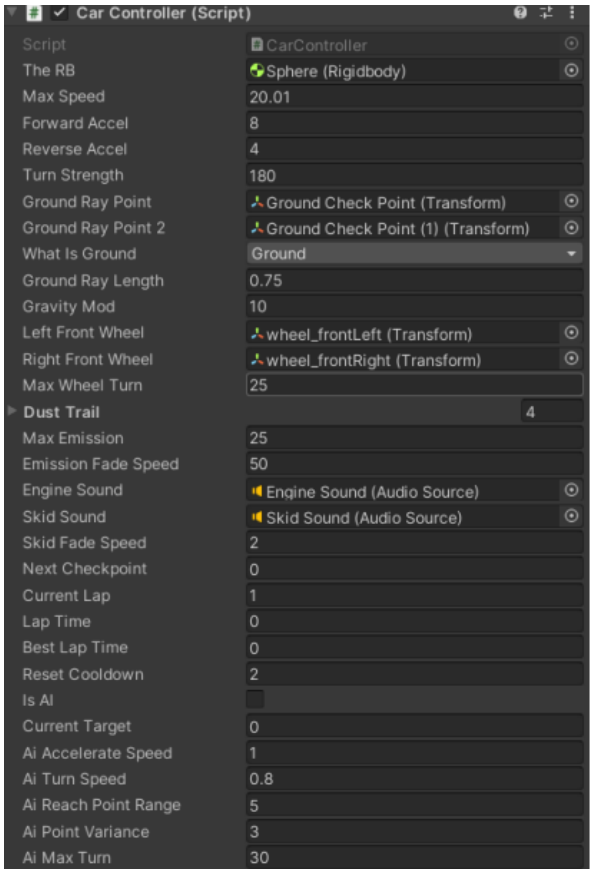

Рисунок 3.12 – Скрипт у інспекторі Unity

3.3.5. Розробка CameraController

Так як ігровий проект iRacing має вид збоку, потрібно налаштувати вид камери, під час проходження різних рівнів.

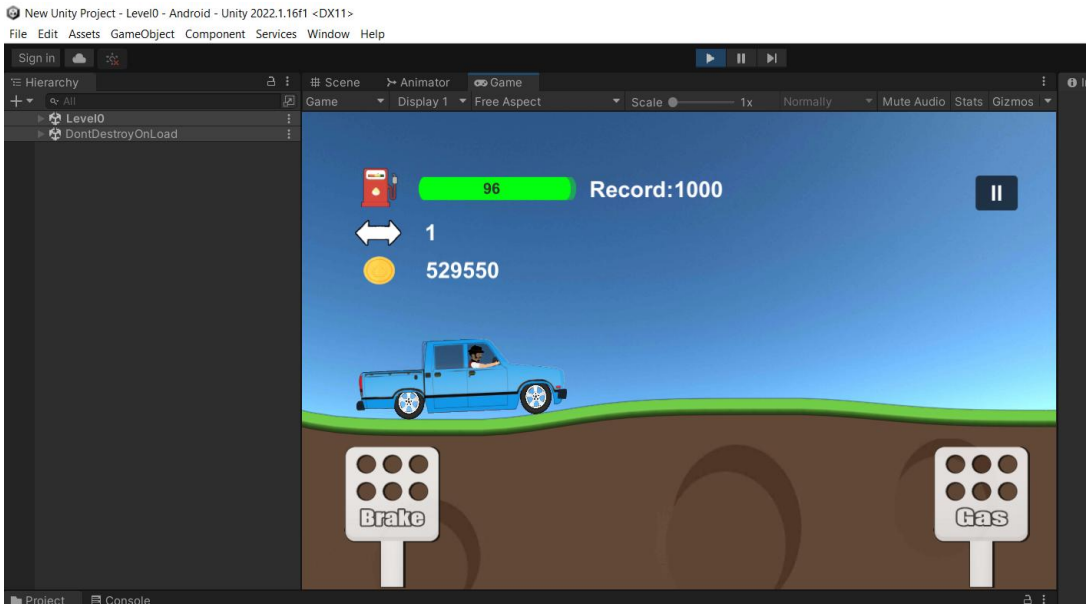

Рисунок 3.13 – Вигляд ігрової сцени зі сторони гравця

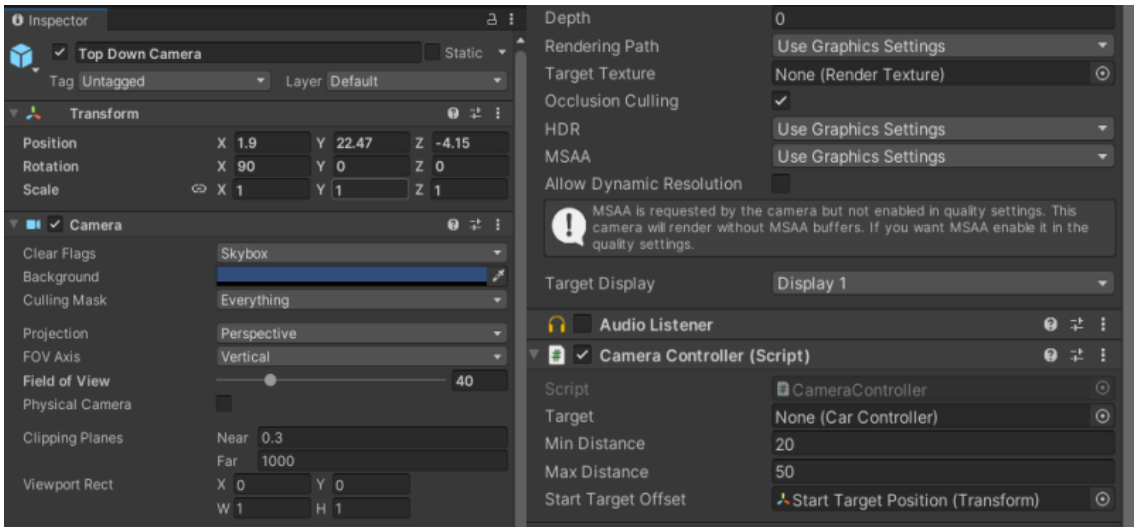

Рисунок 3.14 – Налаштування камери у інспекторі

3.3.6. Розробка MusicManager

Для відтворення аудіо у вашому проекті потрібен MusicManager. Це шум двигуна, вереск шин, зіткнення і музика в головному меню. Використаний аудіофайл створено за ліцензією. Автор асоціює їх як вільні для використання в особистих або комерційних проектах. Ці звуки було вибрано з Asset Store.

Unity Audio Mixer дозволяє змішувати різні джерела звуку, застосовувати ефекти та виконувати мастеринг. У вікні показано аудіомікшери, які в основному

складаються з деревовидної структури груп аудіомікшерів. Група аудіомікшера — це сигнальний ланцюг, який може застосовувати ослаблення гучності та корекцію висоти. Ви можете використовувати його для вставки ефектів, які обробляють звуковий сигнал, і змінювати параметри ефекту. Існує також механізм надсилання/отримання для передачі результатів з однієї шини на іншу.

Рисунок 3.15 – AudioMixer

3.3.7. Розробка графічного оформлення

Слід зазначити, що основні етапи реалізації проекту – розробка графічного дизайну та розробка самого проекту – відбувалися паралельно. Це означає, що для написання програмного коду потрібні готові анімації та інші речі, а стандартних примітивів Unity достатньо для перевірки функціональності самого коду. Такий спосіб реалізації проекту значно збільшує швидкість роботи.

Для цього завдання було використано таке програмне забезпечення: Adobe Photoshop, Blender, Unity Asset Store. Меню ігрового проекту, растрові зображення (спрайти) іконок інтерфейсу створені в Adobe Photoshop.

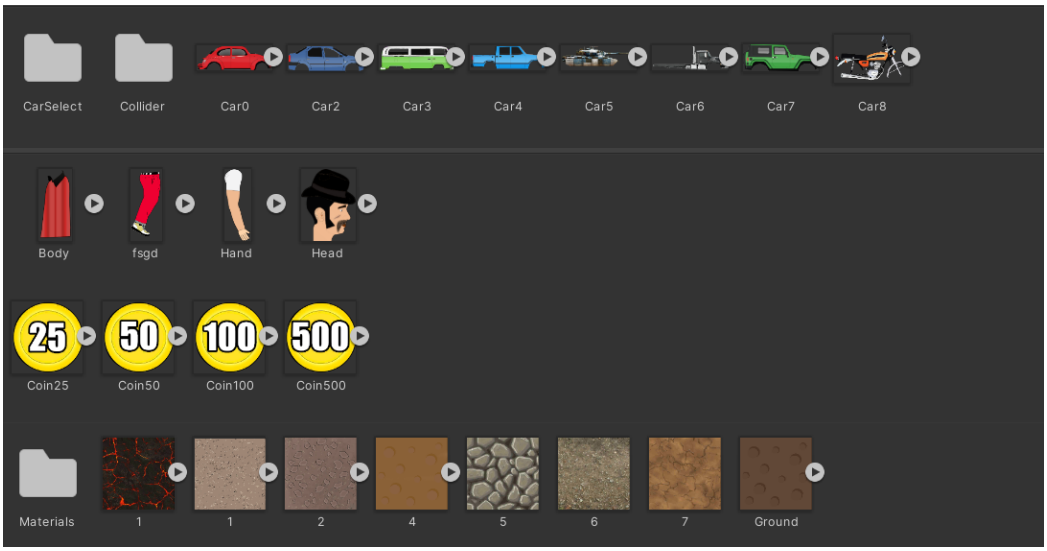

Рисунок 3.16 – Спрайти в Adobe Photoshop

Рисунок 3.17 – Логотип ігрового застосунку «iRacing»

Висновок до розділу 3

Під час роботи над проектом кваліфікаційної роботи були пояснені інструменти розробки, які використовувалися під час розробки проекту, технічні та програмні вимоги, а також процес впровадження ігрових додатків за допомогою засобів розробки.

РОЗДІЛ 4 ТЕСТУВАННЯ

4.1. Функціональне тестування

Функціональне тестування – це вид тестування, спрямований на перевірку відповідності функціональних вимог програмного забезпечення його фактичним характеристикам. Основне завдання функціонального тестування — переконатися, що розроблений вами програмний продукт має всі необхідні функціональні можливості. У таблиці 4.1 наведені результати функціональних випробувань.

Таблиця 4.1

Результати функціонального тестування

№	Назва тесту	Очікуваний результат	Отриманий результат	Тест пройдений?
1	Відображення головного меню гри	На екрані відображається головне меню гри з кнопками «Play» та «Quit»	На екрані відобразилось головне меню гри з кнопками «Play» та «Quit»	+
2	Робота кнопки «Start»	При натиску на кнопку «Start» очікуємо перехід у наступне меню	При натиску на кнопку «Start» перейшло у наступне меню	+
3	Робота кнопки «Quit»	При натиску на кнопку «Quit» очікуємо вихід з гри	При натиску на кнопку «Quit» вийшли з гри	+
4	Відображення меню «Level»	На екрані відображається головне меню гри з вікнами вибору локацій	На екрані відобразилось головне меню гри з вікнами вибору локацій	+
5	Робота меню «Vehicle»	На екрані відображається головне меню гри з вікнами вибору транспортних засобів	На екрані відображається головне меню гри з вікнами вибору транспортних засобів	+
6	Робота меню «Upgrade»	На екрані відображається головне меню гри з вікнами покращення транспортних	На екрані відображається головне меню гри з вікнами	+

		засобів	покращення транспортних засобів	
7	Робота кнопки «START»	При натиску на кнопку «START» очікуємо завантаження рівня гри	При натиску на кнопку «START» завантажується рівень гри	+
8	Робота MusicMixer	Ігровий проект супроводжується аудіо треками у головному меню та під час гри	Ігровий проект супроводжується аудіо треками у головному меню та під час гри	+
9	Робота SimpleInput	При натиску педалей керування автомобілем виконується переміщення машини по карті	При натиску на педалей керування автомобілем виконується переміщення машини по карті	+
11	Робота RaceManager	При закінченні гонки очікується фініш	При закінченні гонки очікується фініш	+

4.2. Usability Test

Тестування — це метод перевірки, який визначає зручність використання, зрозумілість і привабливість продукту для користувачів. Здійснюється за допомогою залучення користувачів з метою проходження тесту, з метою отримання додаткової інформації та висновків від тестувальників. Учасникам тестування необхідно виконати такі задачі:

– почати гру;

– зупинити гру;

– продовжити гру;

– обрати локацію;

– обрати машину;

– зробити апгрейд обраною користувачем машини.

Висновок до розділу 4

Тестери успішно виконали всі завдання без проблем. Респонденти відзначили простоту використання, гарний дизайн і складність проходження. Пройшов тест на юзабіліті.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було вивчено сучасні інформаційні технології навчання, їх вплив на ігрову індустрію. Показані можливості ігрового рушія Unity у розробці ігор.

Додаток Unity, що є професійним ігровим рушієм, використовується у створенні відеоігор для різних платформ. Це інструмент, яким щодня користуються досвідчені розробники, а також один із найдоступніших інструментів для початківців.

При написанні коду використано мову C#. Ця мова дозволяє легко увійти до розробки гри. Мова C# актуальна на сьогоднішній день, тому що поєднує найкращі ідеї сучасних мов програмування Java, C++, Visual Basic і т. д. Через велику різноманітність синтаксичних конструкцій та можливості працювати з платформою Unity.

Для створення 2D графіки застосовано програмне забезпечення Adobe Photoshop, яке головним чином призначене для створення растрової графіки. Особливості Adobe Photoshop полягають у багатому інструментарії для операції створення і обробки зображень, високій якості обробки графічних зображень, зручності й простоті в експлуатації, широких можливостях до автоматизації обробки растрових зображень.

Програмне забезпечення Blender використовувалось для моделювання 3D графіки. Його особливостями пакету є малий розмір, висока швидкість рендерингу, наявність безкоштовної версії.

Таким чином, можна зробити висновок, що використовуючи можливості вищеназваних програм, можна створювати ігрові додатки розважального характеру для різних платформ.

У ході виконання кваліфікаційної роботи бакалавра було реалізовано ігровий застосунок в жанрі симулятора гонок під мобільну систему Android на рушію Unity. Для вирішення поставленої мети було виконано такі завдання:

– проведено аналіз актуальності теми;

– виконано аналіз існуючих аналогів;

– виконано огляд актуальних програмних середовищ для реалізації додатку;

– спроектовано ігровий додаток;

– реалізовано ігровий додаток;

– проведено тестування ігрового застосунку.

– були описані тези на семінарі комп’ютерні науки.

Дана мобільна гра може слугувати як і швидким способом ненадовго відволіктися, так і повноцінною грою, яка забезпечує занурення у свій світ та відволіктися від проблем сьогодення. В грі ви зможете покращувати різні транспортні засоби, від легкової машини до важкого танку. Тут вас чекають різні локації, та непростий рельєф. Проїжджаючи по складним локаціям потрібно подумати де можна розігнатись, а де слід пригальмувати. Під час проходження гри вас буде супроводжувати мелодійна музика. А в кінці вашого непростого шляху вас буде очікувати фініш.

СПИСОК ЛІТЕРАТУРИ

1. Unity Growth Facts [Електронний ресурс] – Режим доступу до ресурсу: https://unity3d.com/ru/public-relations.
2. Genshin Impact works its magic to become biggest global launch of a Chinese game ever [Електронний ресурс] – Режим доступу до ресурсу: https://www.scmp.com/tech/apps-social/article/3103522/genshin-impact-works-its-magic-become-biggest-global-launch.
3. What's new in Unity 2021.1.2 [Електронний ресурс] – Режим доступу до ресурсу: https://unity3d.com/unity/whats-new/2021.1.2.
4. Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C# / Jeremy Gibson Bond., 2019 – (Addison-Wesley).
5. Unity Asset Store [Електронний ресурс] - https://assetstore.unity.com/.
6. Unity Store [Електронний ресурс] – Режим доступу до ресурсу: https://store.unity3d.com/.
7. 1. Дацко М. А. Моделювання складних об'єктів. / М. А. Дацко. – М. : Максімас, 2015. – 111 с.
8. 2. Крейтон, Р.Х. Основи розробки ігор у Unity / Р.Х. Крейтон. – Packt Publishing, 2010, – 83 с.
9. Хокінг Д. М. Unity в дії. Мультиплатформенна розробка на практиці. / Д. М.Хокінг, 2016. – 336 с.
10. Joseph Hocking — Unity in Action. Multiplatform game development in C# with Unity 5, 2015.
11. Alan Thorn — How to Cheat in Unity 5: Tips and Tricks for Game Development, 2015.
12. Kenny Lammers — Unity Shaders and Effects Cookbook, 2013.

ДОДАТОК

Скрипт для керування і руху машини

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class araba : MonoBehaviour
{
    public WheelJoint2D arka_tekerlek;
    public WheelJoint2D on_telerlek;
    public float hiz;
    private float hareket;
    private float rot_ilk;
    private float rot_mevcut;
    public float rot_son;
    private float rot_ilk_t;
    private float rot_mevcut_t;
    public float rot_son_t;
    public int toplam_altin;
    public Text altin_texti;
    public Image yakit_resmi;
    public float tomplam_yakit = 94;
    public float yakit_azalma_hiz;
    private void Start()
    {
        rot_mevcut = transform.rotation.eulerAngles.z;
        rot_ilk = rot_mevcut;
        rot_son = rot_mevcut;
        rot_mevcut_t = transform.rotation.eulerAngles.z;
        rot_ilk_t = rot_mevcut_t;
        rot_son_t = rot_mevcut_t;
    }

    // Update is called once per frame
    void Update()
    {
        float v = Input.GetAxis("Vertical");
        hareket = hiz * v;
        Takla();
        TersTakla();
        RectTransform asd = yakit_resmi.GetComponent<RectTransform>();
        asd.sizeDelta = new Vector2(asd.sizeDelta.x - Time.deltaTime * yakit_azalma_hiz,
asd.sizeDelta.y);
    }

    private void FixedUpdate()
    {
        if(hareket == 0)
        {
            arka_tekerlek.useMotor = false;
            on_telerlek.useMotor = false;
        }
        else
        {
            arka_tekerlek.useMotor = true;
            on_telerlek.useMotor = true;
            JointMotor2D motore = new JointMotor2D();
            motore.motorSpeed = hareket;
            motore.maxMotorTorque = 10000;
            arka_tekerlek.motor = motore;
            on_telerlek.motor = motore;
```

```
        }
    }

    void Takla()
    {
        rot_mevcut = transform.rotation.eulerAngles.z;
        if(rot_son < rot_mevcut)
        {
            rot_ilk = rot_mevcut;
        }
        else if(rot_son > rot_mevcut && rot_son - rot_mevcut > 100)
        {
            rot_ilk = rot_mevcut;
        }
        if (rot_ilk - rot_mevcut > 300)
        {
            Debug.Log("Takla");
            rot_ilk = rot_mevcut;
        }
        rot_son = rot_mevcut;
    }

    void TersTakla()
    {
        rot_mevcut_t = transform.rotation.eulerAngles.z;
        if (rot_son_t > rot_mevcut_t)
        {
            rot_ilk_t = rot_mevcut_t;
        }
        else if (rot_son_t < rot_mevcut_t && rot_mevcut_t - rot_son_t > 100)
        {
            rot_ilk_t = rot_mevcut_t;
        }
        if (rot_mevcut_t - rot_ilk_t > 300)
        {
            Debug.Log("Ters Takla");
            rot_ilk_t = rot_mevcut_t;
        }
        rot_son_t = rot_mevcut_t;
    }
     void OnTriggerEnter2D(Collider2D coll)
    {
        if (coll.gameObject.tag == "altin")
        {
            int miktar = coll.GetComponent<altinn>().miktar;
            toplam_altin += miktar;
            GameObject.Destroy(coll.gameObject);
            altin_texti.text = toplam_altin.ToString();
        }
        if (coll.gameObject.tag == "yakit")
        {
            RectTransform asd = yakit_resmi.GetComponent<RectTransform>();
            asd.sizeDelta = new Vector2(tomplam_yakit, asd.sizeDelta.y);
            GameObject.Destroy(coll.gameObject);
        }
    }
}
```

Скрипт для збору монет

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

public class altinn : MonoBehaviour
{
    public int miktar;
}


void OnTriggerEnter2D(Collider2D coll)
    {
        if (coll.gameObject.tag == "altin")
        {
            int miktar = coll.GetComponent<altinn>().miktar;
            toplam_altin += miktar;
            GameObject.Destroy(coll.gameObject);
            altin_texti.text = toplam_altin.ToString();
        }
    }
using UnityEngine;
using UnityEngine.UI;

public class MoneyCounter : MonoBehaviour
{
    private Text txt;

    private void Awake()
    {
        txt = GetComponent<Text>();
    }
    private void Update()
    {
        txt.text = SaveManager.instance.money + "$";
    }
}
using UnityEngine;

public class MoneyAdd : MonoBehaviour
{
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            SaveManager.instance.money += 100;
            SaveManager.instance.Save();
        }
        else if (Input.GetKeyDown(KeyCode.F))
        {
            SaveManager.instance.money -= 100;
            SaveManager.instance.Save();
        }
    }
}
```

Скрипт для палива

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

public class airtime : MonoBehaviour
{
    float sayac = 0;
    float combo = 0;
    public bool on_air = true;

    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if(sayac > 1f)
        {
            on_air = true;
            combo++;
            sayac = 0;
            Debug.Log("Havada: " + combo);
        }
        sayac += Time.deltaTime;
    }

    void OnCollisionEnter2D(Collision2D coll)
    {
        if (coll.gameObject.tag == "Player" || coll.gameObject.tag == "teker")
        {
            combo = 0;
            sayac = 0;
            on_air = false;
        }
    }
    void OnCollisionStay2D(Collision2D coll)
    {
        if (coll.gameObject.tag == "Player" || coll.gameObject.tag == "teker")
        {
            combo = 0;
            sayac = 0;
            on_air = false;
        }
    }
}
```

Скрипт головного меню гри

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    public void QuitGame()
    {
        Debug.Log("QUIT"); Application.Quit();}}
```

Меню паузи гри

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class PauseMenu : MonoBehaviour
{
    public bool PauseGame;
    public GameObject pauseGameMenu;
    private void Update()
    {
        if(Input.GetKeyDown(KeyCode.Escape))
        {
            if(PauseGame)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseGameMenu.SetActive(false);
        Time.timeScale = 1f;
        PauseGame = false;
    }

    public void Pause()
    {
        pauseGameMenu.SetActive(true);
        Time.timeScale = 0f;
        PauseGame = true;
    }

    public void LoadMenu()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene("Menu");
    }
}
```

Скрипт налаштування

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;
public class OptionMenu : MonoBehaviour
{
    public AudioMixer audioMixer;

    public void SetVolume(float volume)
    {
        audioMixer.SetFloat("volume", Mathf.Log10(volume) * 20);
    }
    public void SetQuality(int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }

    public void Sound()
    {
        AudioListener.pause = !AudioListener.pause;
    }
}
```

Скрипт для покупок в магазині гри

```
using UnityEngine;
using UnityEngine.PlayerLoop;
using UnityEngine.UI;

public class CarSelection : MonoBehaviour
{
    [Header ("Navigation Buttons")]
    [SerializeField] private Button previousButton;
    [SerializeField] private Button nextButton;

    [Header("Play/Buy Buttons")]
    [SerializeField] private Button play;
    [SerializeField] private Button buy;
    [SerializeField] private Text priceText;

    [Header("Car Attributes")]
    [SerializeField] private int[] carPrices;
    private int currentCar;

    [Header("Sound")]
    [SerializeField] private AudioClip purchase;
    private AudioSource source;

    private void Start()
    {
        source = GetComponent<AudioSource>();
        currentCar = SaveManager.instance.currentCar;
        SelectCar(currentCar);
    }

    private void SelectCar(int _index)
    {
        for (int i = 0; i < transform.childCount; i++)
            transform.GetChild(i).gameObject.SetActive(i == _index);

        UpdateUI();
    }
```

```
    private void UpdateUI()
    {
        //If current car unlocked show the play button
        if (SaveManager.instance.carsUnlocked[currentCar])
        {
            play.gameObject.SetActive(true);
            buy.gameObject.SetActive(false);
        }
        //If not show the buy button and set the price
        else
        {
            play.gameObject.SetActive(false);
            buy.gameObject.SetActive(true);
            priceText.text = carPrices[currentCar] + "$";
        }
    }

    private void Update()
    {
        //Check if we have enough money
        if (buy.gameObject.activeInHierarchy)
            buy.interactable = (SaveManager.instance.money >= carPrices[currentCar]);
    }

    public void ChangeCar(int _change)
    {
        currentCar += _change;

        if (currentCar > transform.childCount - 1)
            currentCar = 0;
        else if (currentCar < 0)
            currentCar = transform.childCount - 1;

        SaveManager.instance.currentCar = currentCar;
        SaveManager.instance.Save();
        SelectCar(currentCar);
    }
    public void BuyCar()
    {
        SaveManager.instance.money -= carPrices[currentCar];
        SaveManager.instance.carsUnlocked[currentCar] = true;
        SaveManager.instance.Save();
        source.PlayOneShot(purchase);
        UpdateUI();
    }
       }
```

Скрипт збереження покупок

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using TMPro;
using UnityEngine;

public class SaveManager : MonoBehaviour
{
    public static SaveManager instance { get; private set; }

    //What we want to save
    public int currentCar;
    public int money;
    public bool[] carsUnlocked = new bool[6] { true, false, false, false, false, false};
```

```
    private void Awake()
    {
        if (instance != null && instance != this)
            Destroy(gameObject);
        else
            instance = this;

        DontDestroyOnLoad(gameObject);
        Load();
    }

    public void Load()
    {
        if (File.Exists(Application.persistentDataPath + "/playerInfo.dat"))
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream file = File.Open(Application.persistentDataPath + "/playerInfo.dat",
FileMode.Open);
            PlayerData_Storage data = (PlayerData_Storage)bf.Deserialize(file);

            money = data.money;
            currentCar = data.currentCar;
            carsUnlocked = data.carsUnlocked;

            if (data.carsUnlocked ==  null)
                carsUnlocked = new bool[6] { true, false, false, false, false, false };

            file.Close();
        }
    }

    public void Save()
    {
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Create(Application.persistentDataPath + "/playerInfo.dat");
        PlayerData_Storage data = new PlayerData_Storage();

        data.money = money;
        data.currentCar = currentCar;
        data.carsUnlocked = carsUnlocked;

        bf.Serialize(file, data);
        file.Close();
    }
}

[Serializable]
class PlayerData_Storage
{
    public int currentCar;
    public int money;
    public bool[] carsUnlocked;
}
```