Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(рівень вищої освіти)
на тему:
"Розроблення мобільної гри "Drag Racing" засобами Unity"

Виконав: студент **2** курсу, групи **ІСТС-21**
Спеціальності **126** -"Інформаційні системи та технології"
(шифр і назва напряму підготовки, спеціальності)
Масляк А. О.
(прізвище та ініціали)
Керівник Капран І. Д., Борецька І. Б.
(прізвище та ініціали)
Рецензент Дончук М. В.
(прізвище та ініціали)

Львів – 2023 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут *деревообробних та комп'ютерних технологій і дизайну*
Кафедра *інформаційних систем та комп'ютерного моделювання*
Рівень вищої освіти *перший (бакалаврський)*
Спеціальність *126 – "Інформаційні системи та технології"*
(шифр і назва)

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ІСКМ
_____ Сторожук О. Л.
"12" 06 2023 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Масляку Артему Оскаровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «*Розроблення мобільної гри "Drag Racing" засобами Unity*»
керівник проекту (роботи) *Капран Ігор Дмитрович, магістр,*
Борецька Ірина Богданівна, доцент, кандидат технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від"*21*"*листопада* *2022* р.№ *С-521*
2. Термін подання студентом проекту (роботи) *12 червня 2023 року*
3. Вихідні дані до проекту (роботи) *Аналіз шляхів вирішення поставлених задач та переваг і недоліків аналогів мобільної гри, огляд алгоритмів та програмних засобів для розроблення мобільної гри «Drag Racing» під операційну систему Android засобами Unity.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *Вступ. Розділ 1. Стан проблемної області.*
Розділ 2. Інформаційне та математичне забезпечення.
Розділ 3. Програмне та технічне забезпечення.
Висновки. Список використаної літератури. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайди для доповіді (підготовка матеріалу для доповіді загальним обсягом 10-12 слайдів)

6. Дата видачі завдання *25 листопада 2022 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Системний аналіз стану проблемної області. Огляд літературних джерел згідно досліджуваної теми. Збір потрібних матеріалів. Формування функціональних вимог та постановка задачі проекту.	*03. 02. 2023 р. 14. 02. 2023 р.*	*Виконано*
2.	Огляд сучасного стану проблемної області. Оформлення першого розділу пояснювальної записки.	*14. 02. 2023 р. 28. 02. 2023 р.*	*Виконано*
3.	Написання другого розділу. Аналіз інформаційного та математичного забезпечення.	*28. 02. 2023 р. 16. 03. 2023 р.*	*Виконано*
4.	Оформлення третього розділу пояснювальної записки. Програмна реалізація	*16. 03. 2023 р. 27. 03. 2023 р.*	*Виконано*
5.	Оформлення третього розділу пояснювальної записки. Формування апаратного забезпечення.	*27. 03. 2023 р. 03. 04. 2023 р.*	*Виконано*
6.	Оформлення пояснювальної записки та здача на рецензування.	*03. 04. 2023 р. 12. 06. 2023 р.*	*Виконано*

Студент _____ *Масляк А. О.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Капран І. Д.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Борецька І. Б.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 42 сторінок пояснювальної записки, 32 рисунків, 1 таблиця, 5 додатків, 15 джерел.

Дипломна робота присвячена проектуванню та розробці мобільної гри «Drag Racing» для пристроїв на операційній системі Android. Під час виконання дипломної роботи було проаналізовано основні тенденції у розробці мобільних застосунків, розглянуто аналоги даної гри, виявлено їх недоліки, на основі яких створено нову мобільну гру згідно із заявленим технічним завданням. Було розроблено графічне забезпечення гри, структуру гри, та саму гру. Проведене тестування розробленої мобільної гри, а також були розробленні інструкції щодо використання гри. Для розроблення гри використано багатоплатформовий інструмент Unity та об'єктно-орієнтовану мову програмування C#.

Ключові слова: мобільна гра, моделювання, програмування, проектування, C#, Unity, Android.

ABSTRACT

Thesis contains 42 pages of explanatory note, 32 drawings, 1 table, 5 appendices, 15 sources.

Thesis is devoted to the design and development of the mobile game "Drag Racing" for devices running the Android operating system. During the thesis the main trends in the development of mobile applications were analyzed, analogues of this game were considered, their shortcomings were identified, on the basis of which the game was created in accordance with the stated technical task. Graphics of the game, the structure of the game, and the game itself were developed. Testing of the developed mobile game was conducted, as well as instructions on the use of the game were developed. The multi-platform tool Unity3D and object-oriented programming language C # were used to develop the game.

Keywords: mobile game, modeling, programming, designing, C#, Unity, Android.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити та програмно реалізувати на основі комп'ютерних технологій мобільну гру «Drag Racing» в жанрі перегони під операційну систему Android. Для розроблення даної гри використати багатоплатформовий інструмент та рушій Unity. Логіку мобільної гри реалізувати за допомогою мови програмування C#. Для досягнення поставленої мети потрібно вирішити наступні задачі:

1) Аналіз існуючих аналогів та вибір способу реалізації;

2) Розроблення структури мобільної гри;

3) Розроблення ігрового застосунку;

4) Тестування проекту.

За результатами проведеної роботи скласти пояснювальну записку, яка крім теоретичної частини має містити в собі детальне пояснення виконання кожного кроку завдання.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМНОЇ ОБЛАСТІ 9
1.1. Статистика ринку мобільних ігор 9
1.2. Огляд аналогів мобільної гри 14
РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 16
2.1. Огляд ігрового рушія Unity 16
2.2 Редактор Unity – програмний інструмент 18
2.3. Unity Simulator 22
2.4. Мова програмування C# 23
РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 25
3.1. Концепція мобільної гри «Drag Racing» 25
3.2. Розроблення головного меню гри 26
3.3. Створення ігрової сцени 30
ВИСНОВКИ 42
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43
ДОДАТКИ 45

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Android - операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux.

Unity - багатоплатформовий інструмент для розробки відеоігор і застосунків та рушій, на якому вони працюють.

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

IDFA - унікальний рекламний ідентифікатор для пристроїв Apple.

KPI (англ. key performance indicators) - фінансова та нефінансова система оцінки, яка допомагає організації визначити ступінь досягнення стратегічних цілей.

Спрайт (англ. sprite) - графічний об'єкт у комп'ютерній графіці.

ВСТУП

Розроблення мобільних застосунків це дуже перспективне і вигідне заняття. Одним із найцікавіших напрямків є мобільні ігри. Однак, ігри - це досить складні програми, а враховуючи, що на ринку існує як мінімум дві мобільні платформи, на які варто орієнтуватися, складність зростає вдвічі. У наші дні існує багато ігрових мобільних застосунків. Двовимірні, тривимірні, віртуальної реальності та інші. На перший погляд, вже зроблено все що потрібно і можна було б просто створювати власну гру. Однак за статистикою близько половини з топ 100 мобільних ігор зроблено на власних рушіях. Мобільні ігри займають досить серйозну нішу в сучасній індустрії відео-розваг.

Актуальність теми: Сьогодні на ринку мобільної розробки можна спостерігати парадоксальну ситуацію: технології випереджають думку. Програмісти та дизайнери отримали у своє розпорядження широкий набір інструментів та технологій, на базі яких можна генерувати безліч ідей для потенційно успішних мобільних застосунків. Не випадково особливої популярності набуває такий формат розваг як ігри, який об'єднує IT-фахівців для спільної роботи. Ринку потрібні нові концепції в даній сфері.

Об’єкт дослідження: процес розроблення мобільного, двомірного ігрового застосунку у жанрі перегони за допомогою Unity для пристроїв на операційній системі Android.

Предмет дослідження: методи, засоби, прийоми та інформаційні технології розроблення ігрового ПЗ для пристроїв на базі операційної системи Android.

Метою дипломної роботи є розробити та вдосконалити на основі аналізу аналогів даної гри мобільний застосунок згідно із заявленим технічним завданням.

Результатом роботи є apk-файл готової мобільної гри «Drag Racing» для пристроїв на операційній системі Android.

РОЗДІЛ 1. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Статистика ринку мобільних ігор

Ринок мобільних ігор постійно змінюється. Якщо ви являєтеся розробником мобільних ігор, видавцем, маркетологом або інвестором, вам необхідно бути в курсі тенденцій ринку мобільних ігор і розуміти, куди рухається галузь. Це єдиний спосіб приймати зважені рішення, отримуючи цінну інформацію про поточний стан індустрії мобільних ігор та її майбутнє. Крім того, корисною буде інформація про мобільних гравців, статистику монетизації , залучення користувачів і найважливіші KPI мобільних ігор.

У попередньому році (2022) ми спостерігали корекцію ринку мобільних ігор після двох років зростання, яке відбулося в результаті карантину. Глобальна рецесія, зміни IDFA та бурхливий період на китайському ринку ігор також вплинули на падіння світового ринку ігор.

Однак падіння після карантину було очікуваним, тому ми з оптимізмом дивимося в майбутнє ринку мобільних ігор, незважаючи на економічний спад. Крім того, навіть незважаючи на падіння ринку, мобільний зв’язок є найуспішнішим і найприбутковішим сектором промисловості.

Глобальний ринок ігор, включаючи мобільні та інші платформи, заробив понад 184 мільярди доларів у 2022 році. Це на -4,3% менше порівняно з минулим роком. Очікується, що до 2025 року він зросте до 211 мільярдів доларів (Newzoo). У Північній Америці ринок ігор впав на -5,1% порівняно з минулим роком до 48,4 мільярда доларів у 2022 році, а в Європі він впав на -3,5% порівняно з минулим роком до 32,9 мільярда доларів. Ринок ігор в Азіатсько-Тихоокеанському регіоні досяг $87,9 млрд із падінням на -5,6% порівняно з минулим роком (рис. 1.1).

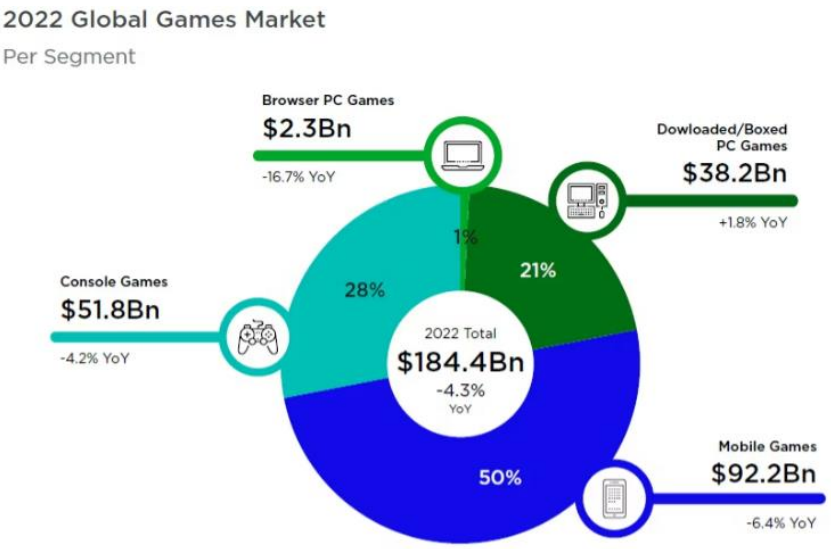

Рисунок 1.1. Розмір ринку і дохід від мобільних ігор

Із 184 мільярдів доларів, отриманих у 2022 році всіма типами ігор, частка ринку мобільних ігор становить 50%, або 92 мільярди доларів. Консольні ігри мали 28% доходу, комп'ютерні ігри – 21%, а браузерні комп'ютерні ігри – лише 1%.

Дохід від мобільних ігор у 2022 році склав 92 мільярди доларів, що на -6,4% менше порівняно з минулим роком. Незважаючи на зниження доходу, спостерігався сплеск завантажень мобільних ігор. У 2022 році їх кількість досягла майже 90 мільярдів, що на 6 мільярдів більше, ніж у 2021 році. Основним драйвером завантажень були гіперказуальні ігри. Statista прогнозує, що до 2027 року ринок мобільних ігор зросте до 419 мільярдів доларів. Середній дохід на користувача (ARPU) для мобільних ігор досягне $164 у 2023 році.

У 2022 році користувачі в Індонезії, Бразилії, Саудівській Аравії, Сінгапурі та Південній Кореї перевищили 5 годин на день у мобільних додатках та іграх. Трійку лідерів мобільних ринків у 2022 році за кількістю завантажень склали Китай, Індія та США. Трьома найбільшими ринками мобільного зв'язку у 2022 році за споживчими витратами були Китай, США та Японія. 1419 додатків та ігор заробили понад 10 мільйонів доларів у 2022 році. 224 перевищили 100 мільйонів доларів, а 10 перевищили 1 мільярд доларів.

Станом на 2023 рік у магазинах додатків доступно майже 700 000 ігор – близько 500 000 у Google Play і 200 000 в Apple App Store.

Наш обширний список статистики мобільних ігор був би неповним без провідних жанрів і піджанрів. Дані жанри і піджанри були лідерами у 2022 році за споживчими витратами та завантаженнями (рис. 1.2).

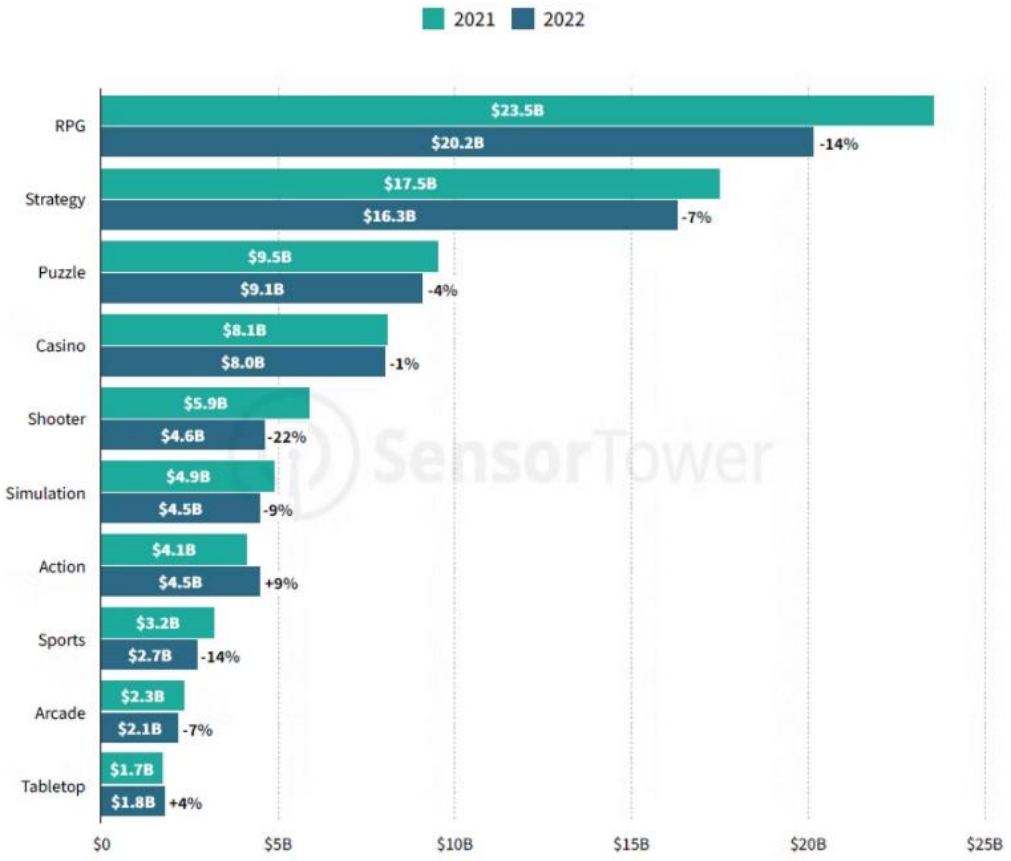

Рисунок 1.2. Найкращі жанри та піджанри мобільних ігор

За даними SensorTower, найкращими жанрами у 2022 році у всьому світі за валовим доходом були:

1. RPG (20,2 мільярда доларів);
2. Стратегія ($16,3 млрд);
3. Puzzle ($9,1 млрд);
4. Казино (8 мільярдів доларів);
5. Шутер ($4,6 млрд);
6. Симуляція (4,5 мільярда доларів);
7. Екшн ($4,5 млрд);
8. Спорт (2,7 мільярда доларів);
9. Arcade ($2,1 млрд);
10. Настільний ($1,7 млрд).

Найкращими піджанрами мобільних ігор, які продовжують розвиватися, незважаючи на світові тенденції були:

- Найпопулярніші піджанри мобільних ігор за споживчими витратами : Puzzle – Word, Match – Merge, Party – Party Royale, RPG – Roguelike ARPG і Simulation – Creative Sandbox.
- Якщо ми подивимося на завантаження, найпопулярнішими піджанрами мобільних ігор є RPG – MMORPG, Match – Match3, Match – Connect Tiles, Match – Merge та RPG – Roguelike ARPG.
- Найпопулярніші піджанри мобільних ігор за кількістю витрачених годин : симулятори – Creative Sandbox, стратегії – стратегія 4x, матч – Match3, настільний пасьянс і головоломка – слово.

Згідно з дослідженням data.ai, найкращими іграми MAU, у які, швидше за все, гратимуть різні вікові групи зображені на рис. 1.3. Загалом молоді гравці люблять стрілянину та ігри-симулятори, тоді як старші геймери віддають перевагу іграм у казино та головоломкам.

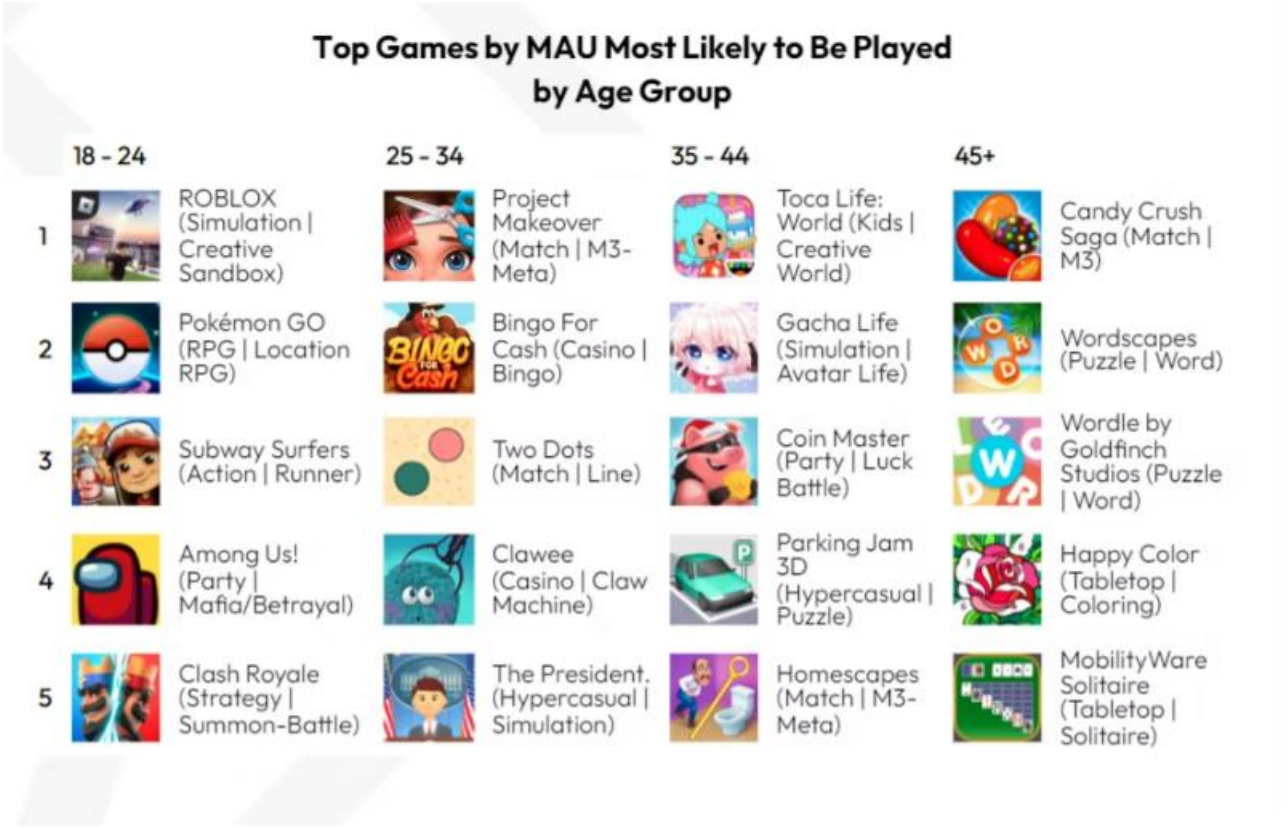

Рисунок 1.3. Найкращі ігри за віковою групою

Судячи з цієї статистики мобільних ігор, очевидно, що ринок мобільних ігор зазнав невеликого спаду в 2022 році після буму в 2020 і 2021 роках. Однак ця корекція ринку була очікуваною, і ми впевнені, що ринок мобільних ігор стабілізується та продовжить зростати у 2023 році та в наступні роки.

Розробникам ігор і маркетологам потрібно змінити свої стратегії, щоб зосередитися на покращенні взаємодії з користувачами та зробити свої ігри максимально привабливими – це ключ до успішної монетизації.

Крім того, стратегії залучення користувачів постійно розвиваються, і зараз як ніколи важливо націлюватися на правильну аудиторію за допомогою правильних оголошень. Зміни IDFA суттєво вплинули на ринок мобільних ігор, який також продовжуватиме розвиватися до 2023 року.

Ми спостерігали зростання NFT та блокчейн-ігор , які мають потенціал назавжди змінити ринок мобільних ігор. Крім того, розмови про метавсесвіт змусили всіх задуматися, як це вплине на ігри.

У будь-якому випадку, буде цікаво спостерігати, як ринок мобільних ігор буде змінюватися та розвиватися в майбутньому.

1.2. Огляд аналогів мобільної гри

Незважаючи на всі незручності ігор у жанрі перегони для мобільних пристроїв, їх вже існує незліченна кількість - влаштувати перегони можна практично на будь якому виді транспорту, починаючи від мотоциклів та аквабайків і закінчуючи космічними кораблями.

Need For Speed Undercover

Старі добрі гоночні часи повернулися знову: машина їде по трасі, яка лише злегка змінює напрямок, повернути на дорогу, що примикає, зовсім не представляється можливості. Зате можна йти в дрифт, сповільнювати перебіг часу, купувати нові машини, займатися їх тюнінгом. Динамічний та захоплюючий ігровий процес забезпечений. Головний плюс NFSU приголомшливий рівень графіки. На тлі інших ігор здається, що EA просто переступила межу можливого. До NFSU такої графіки не було ні в кого (рис. 1.4).

Рисунок 1.4. Мобільна гра Need For Speed Undercover

Fastlane Street Racing

Це перші нормальні перегони в магазині застосунків. Усі попередні були з суттєвими недоліками. Дуже складні, але від цього не менш цікаві перегони. Багато машин, щоправда зовсім нереальні, хоч і нагадують зовнішнім виглядом деякі дуже дорогі спорткари. Проте, всі машини суттєво відрізняються одна від одної.

В основному режимі гри - аркадному, ви як звичайно ганяєтеся з опонентами різними трасами, які відкриваються в міру проходження гри. У другому режимі, опонентом стаєте самі ви: треба проїхати трасу не зачіпаючи огорож, встигнути проїхати потрібну кількість кіл і наздогнати опонента.

На самому початку гри необхідно потерпіти буквально пару годин, звикнути в управлінні, перейнятися духом гри і кілька тижнів захоплюючих перегонів вам забезпечено. Для бажаючих потестувати гру безкоштовно є лайт версія (рис. 1.5).

Рисунок 1.5. Мобільна гра Fastlane Street Racing

Firemint Realracing

Найсвіжіша гра у трійці. Назва гри зовсім не випадкова, компанія розробник намагалася отримати максимум реалізму від мобільного телефону. На противагу реалізму в грі є можливості включити віртуального помічника, який допомагатиме пригальмовувати в потрібний момент, що дуже корисно для новачків. Усі заїзди відбуваються на кільцевих трасах. Автомобілі розділені на кілька класів і, як це було у випадку з Fastlane RealRacing, схожі на реально існуючі моделі, але називаються інакше.

RealRacing не випадково потрапив на останнє місце. Як гонки вона просто прекрасна, але захоплюючою та різноманітною назвати її навряд чи вдасться. Для прискіпливих користувачів є навіть вид з кабіни з кермом і приладами. Треба також відзначити, що це єдині перегони зі списку, які розвиваються - у грі з'явився ще один четвертий клас автомобілів (рис. 1.6).

Рисунок 1.6. Мобільна гра Firemint Realracing

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Огляд ігрового рушія Unity

Очевидно, за всіма параметрами лідирує ігровий рушій Unity. Не тільки в мобільній ігровій розробці (де він займає понад 50% ринку), а й на Steam для ПК (рис. 2.1).

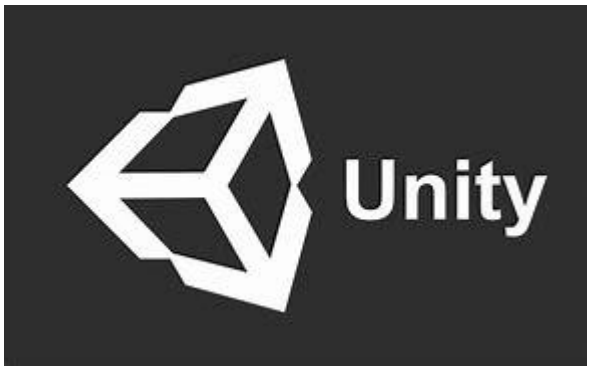

Рисунок 2.1. Багатоплатформовий інструмент для розробки та рушій Unity

Це видно із діаграми нижче (рис. 2.2.), на якій показана популярність ігрових рушіїв в Steam за роками. Unity дійсно є домінуючим рушієм, починаючи з 2016 року.

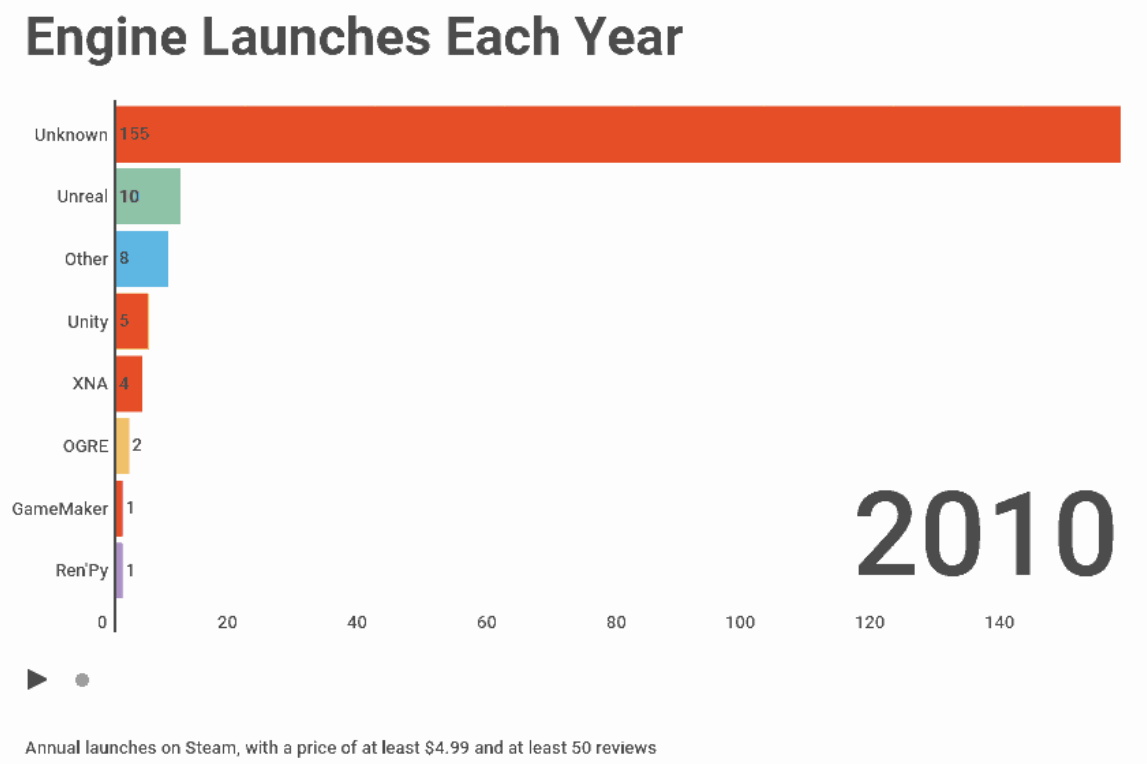

Рисунок 2.2. Діаграма популярності ігрових рушіїв

Майже ті самі результати зображені на рис. 2.3., але у вигляді кругової діаграми:

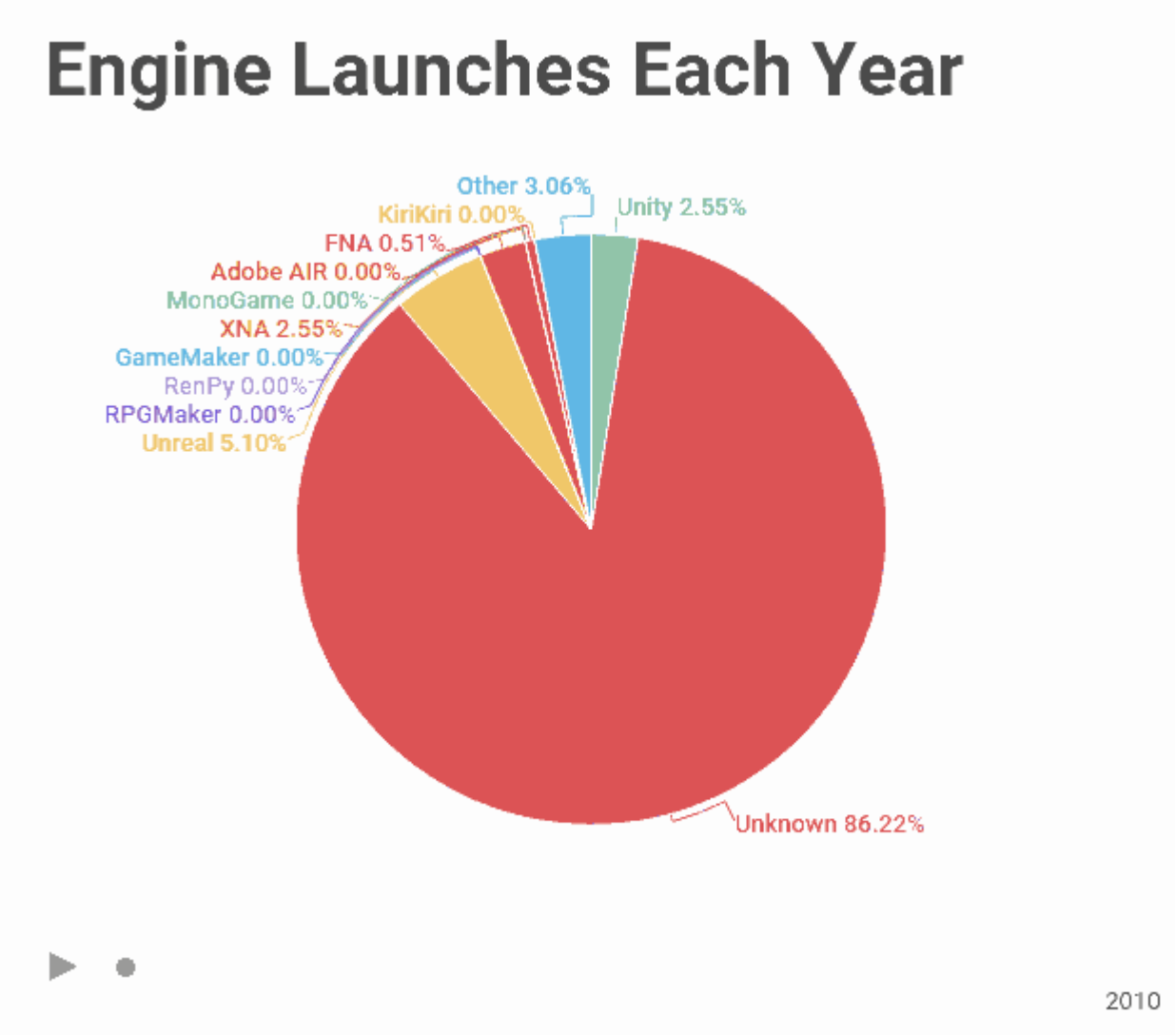

Рисунок 2.3. Кругова діаграма популярності ігрових рушіїв

Unity (Unity Technologies). Вперше випущений у 2005 році як рушій тільки для Mac OS X, тепер Unity підтримується і Windows, і Linux, а також відомий своєю кроссплатформенністю на всі мобільні, консольні, ПК-платформи і навіть AR/VR. Відрізняється потужним ком'юніті та великою кількістю навчальних матеріалів, а також наявністю величезної бібліотеки ассетів та плагінів, за допомогою яких можна значно прискорити процес ігрової розробки. Це найпопулярніший двигун серед розробників ігор, причому як серед інді розробників, так і великих студій. Рушій зосереджений на ідеї «доступності»: у нього досить низький поріг входу, його легко освоїти, він безкоштовний для незалежних розробників.

Так, Unity досить простий в освоєнні, але якщо ви хочете створювати щось складніше примітивних платформерів, то вам знадобиться гарне знання мови програмування C# для написання скриптів та об'єктів і подальшого впровадження їх у гру. Також потрібно розуміти, що Unity - вже досить старий рушій, тому в ньому є свої особливості та артефакти, а разом з тим - порядна повільність і необхідність допрацьовувати деякі інструменти самотужки. Наприклад, ігри, які використовують uNet для роботи в мережі, незабаром мають підтримувати інфраструктуру самостійно, оскільки підтримка цього інструменту поступово припиняється.

2.2 Редактор Unity – програмний інструмент

Редактор Unity - програмний інструмент, який використовується для створення 2D та 3D ігор, застосунків. Графічна та програмна частина редактора періодично оновлюються розробниками, але загальні риси графічного інтерфейсу користувача, функціонал та підхід до розробки зберігаються (рис. 2.4).

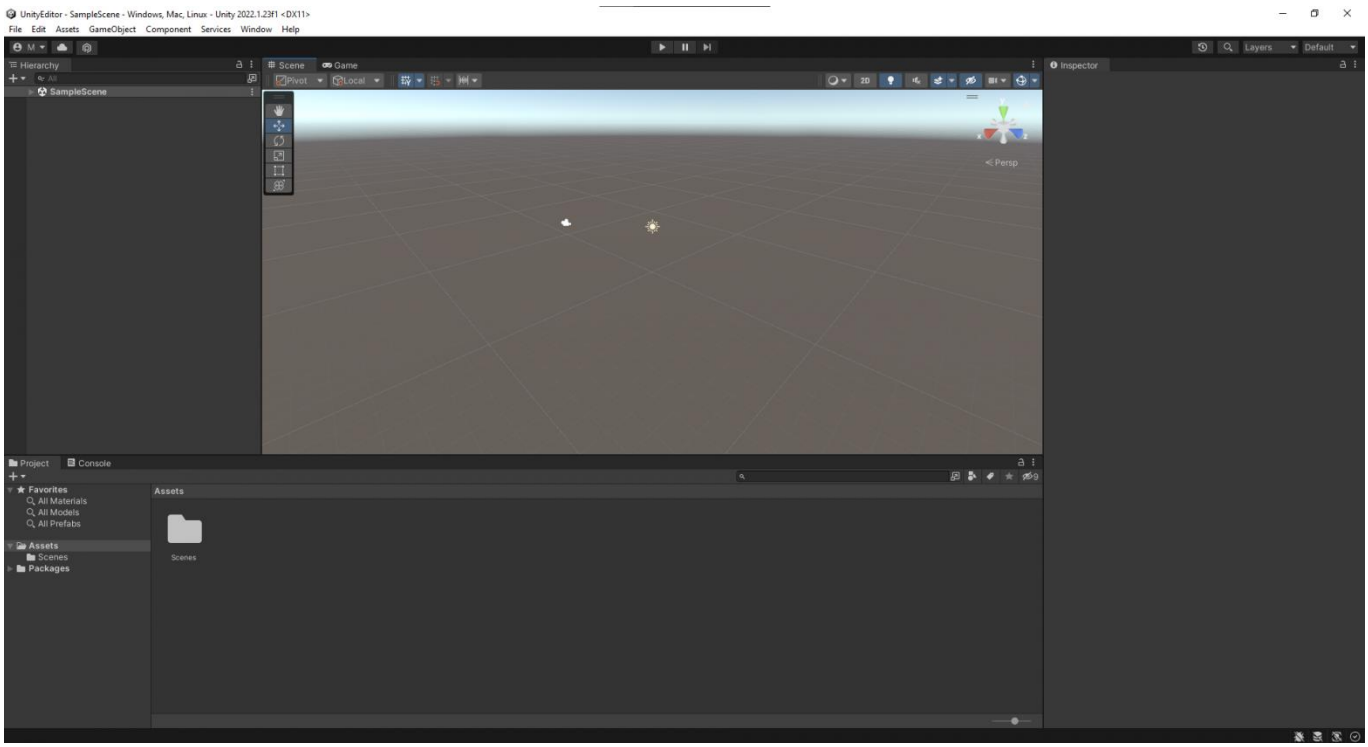

Рисунок 2.4. Редактор Unity версія 2022.1.23

У вікні Scene відображається сцена - віртуальний світ, що створюється користувачем. Користувач може вибирати, переміщати та редагувати об'єкти на

сцені. На сцену додаються ігрові об'єкти, такі як персонажі, світло, ефекти, камери, декорації (рис. 2.5).

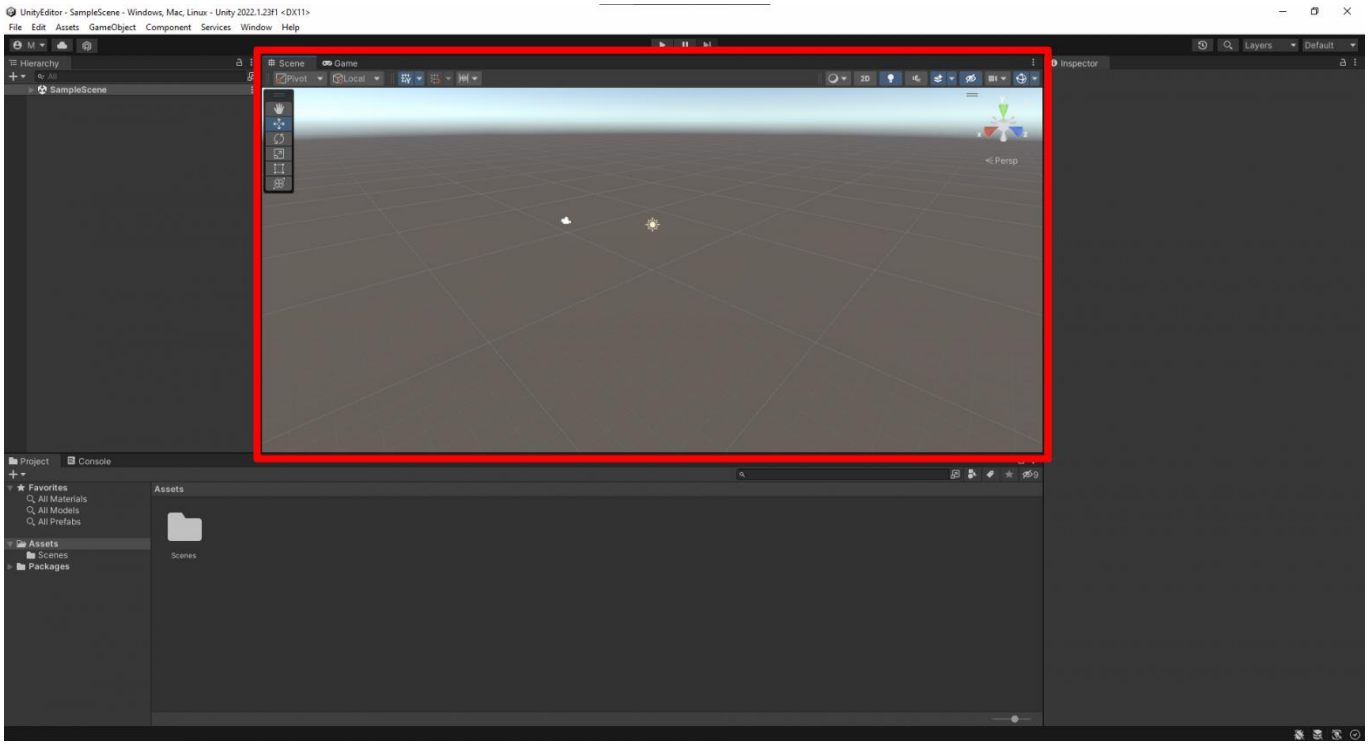

Рисунок 2.5. Вікно "Scene"

Для керування камерою редактора та зміни виду існує кілька методів:

1. Управління стрілками клавіатури;
2. Управління мишею та клавіатурою;

У вікні "Project" відображаються всі файли та папки, які використовуються у проекті. Через вікно "Project" користувач може створювати, редагувати, відкривати, перейменовувати, копіювати та видаляти файли (рис. 2.6).

У вікні Hierarchy відображаються всі об'єкти, розташовані на сцені. Через вікно "Hierarchy" користувач може вибирати, видаляти, копіювати, перейменовувати, сортувати та об'єднувати в групи об'єкти на сцені (рис. 2.7).

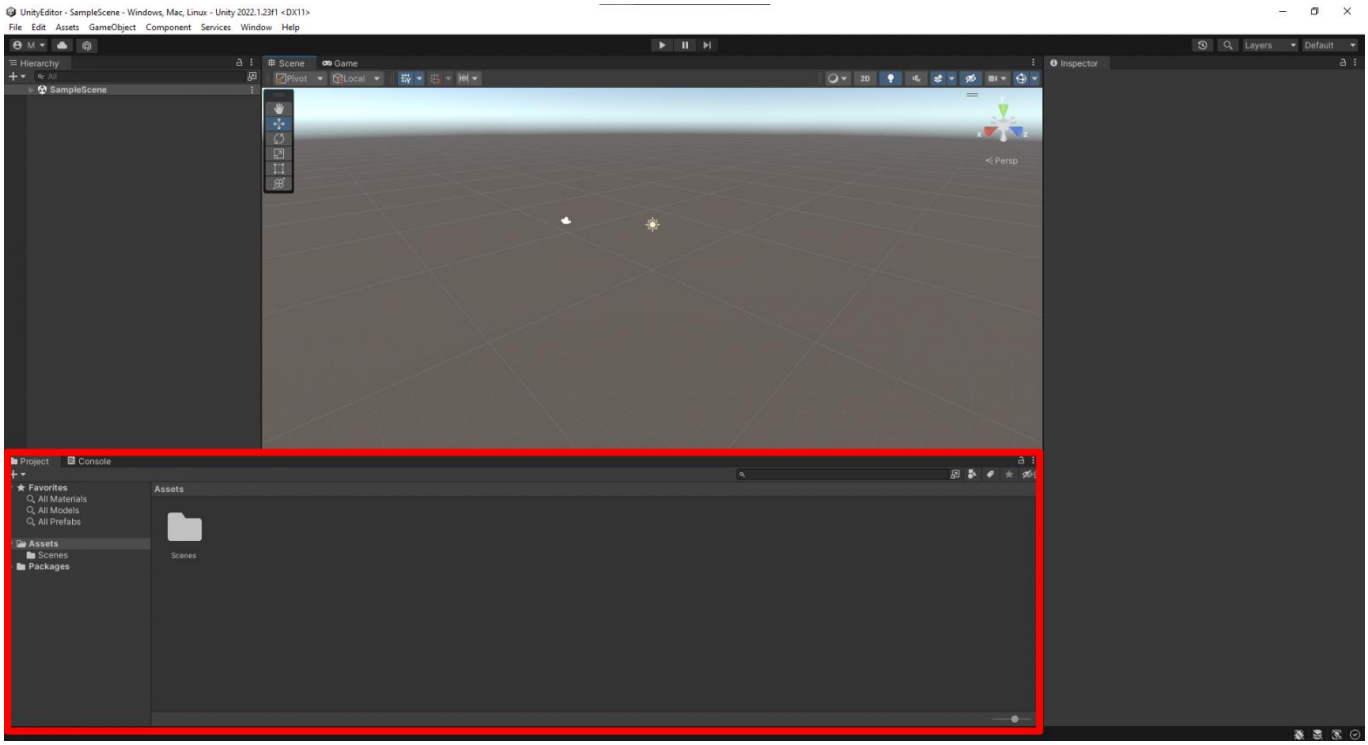

Рисунок 2.6. Вікно "Project"

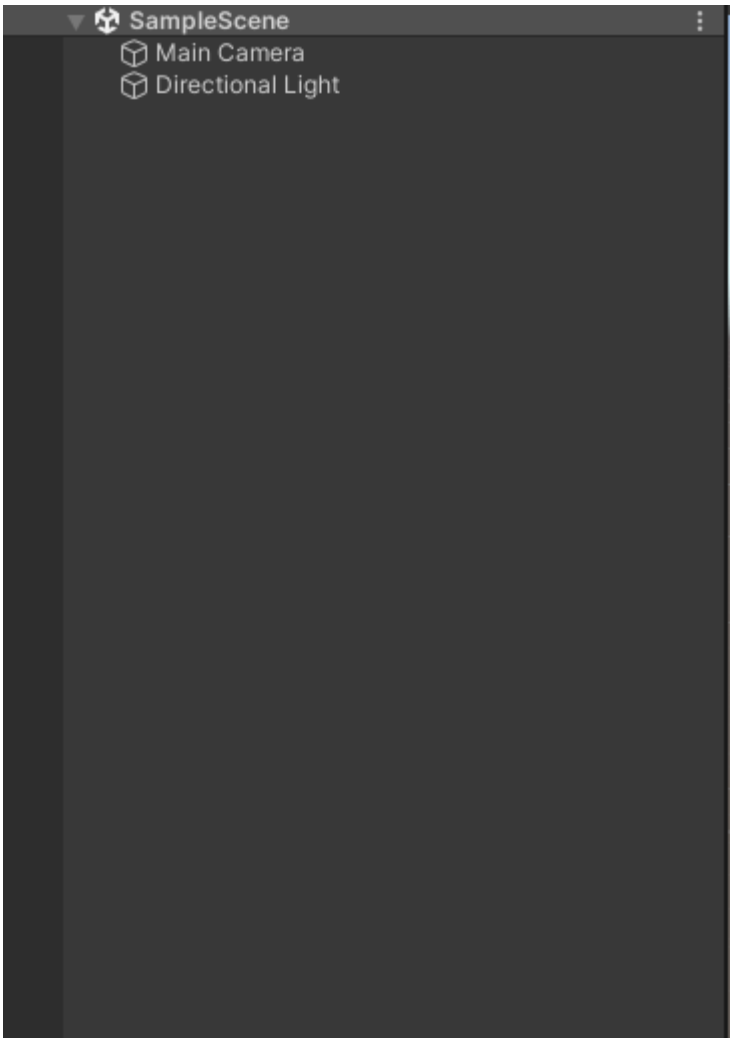

Рисунок 2.7. Вікно "Hierarchy"

У вікні Inspector відображаються всі властивості вибраного об'єкта. Користувач може переглядати та редагувати параметри об'єктів на сцені, компонентів, матеріалів та файлів у проекті (рис. 2.8).

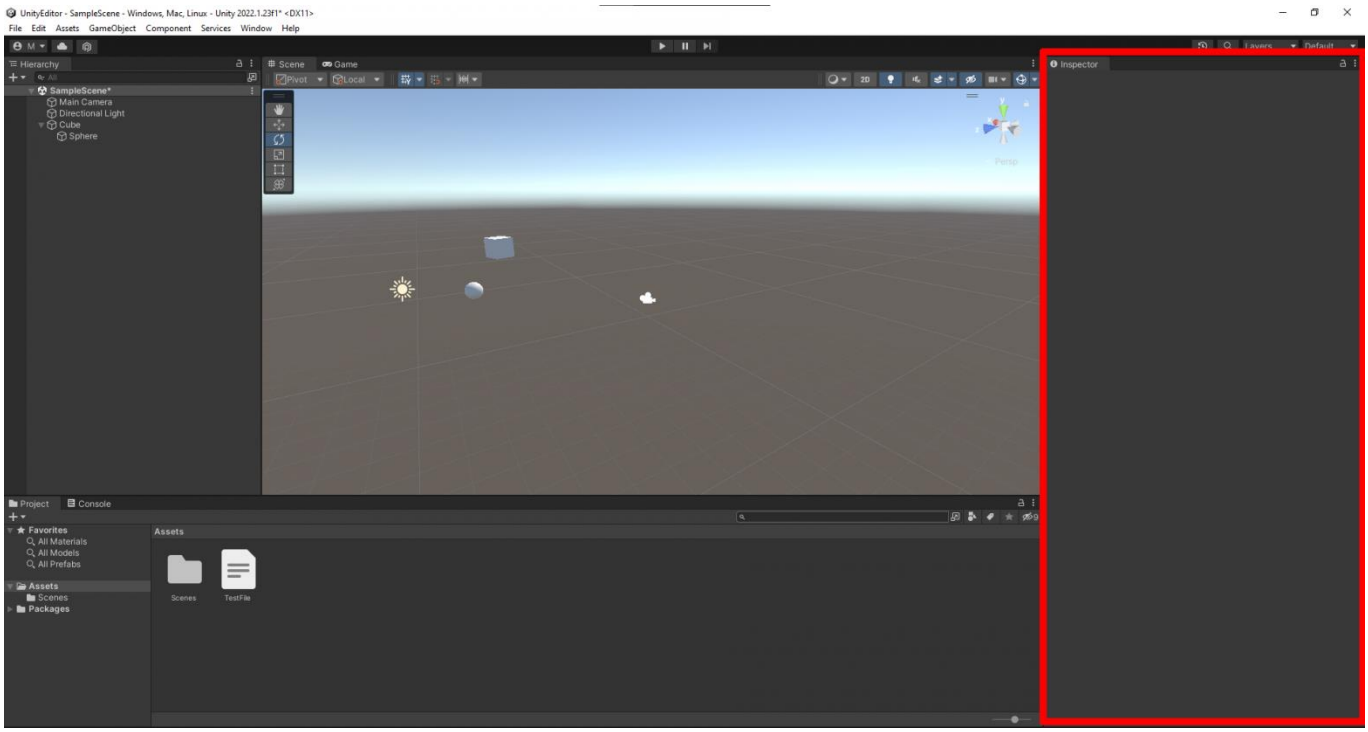

Рисунок 2.8. Вікно "Inspector"

На рисунку 2.8. вікно порожнє. Для відображення властивостей потрібно вибрати об'єкт. Вибрано раніше доданий до проекту текстовий документ "TestFile.txt". У вікні "Inspector" з'явилася інформація про файл та його вміст (рис. 2.9).

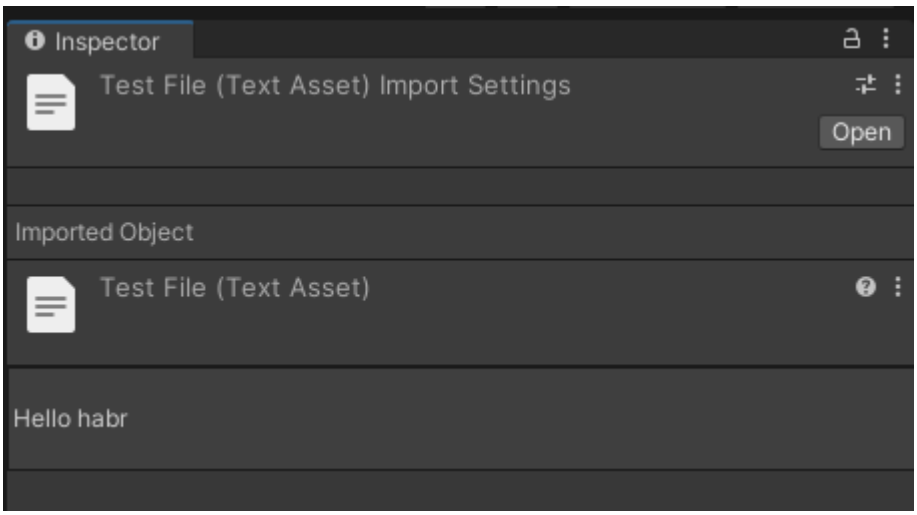

Рисунок 2.9. Інформація про файл TestFile.txt

Редактор Unity має можливість гнучкого налаштування інтерфейсу. Користувач може змінювати розміри, орієнтацію та положення вікон, прикріплювати їх до панелей, створювати вкладки. Користувач може зберегти і завантажити розташування і параметри вікон за допомогою списку "Select editor layout", що знаходиться в правому верхньому кутку редактора.

2.3. Unity Simulator

Симулятор - одна із стандартних функцій Unity про яку новачки, як правило, дізнаються далеко не відразу. Найчастіше це взагалі відбувається випадково, коли процес розробки вже налагоджений і розбиратися з чимось новим немає ніякого бажання: з цієї причини багато розробників не використовують дану функцію. І дарма, адже вона може суттєво полегшити роботу над проектом.

Незважаючи на те, що основна мета симулятора - демонстрація того, як гра виглядатиме на сучасних пристроях з урахуванням їх різноманітних вирізів, його функціонал цим не обмежується.

Спочатку потрібно додати симулятор, як одне з активних вікон редактора. Для цього у верхньому меню йдемо шляхом: Window/General/Device Simulator (рис. 2.10).

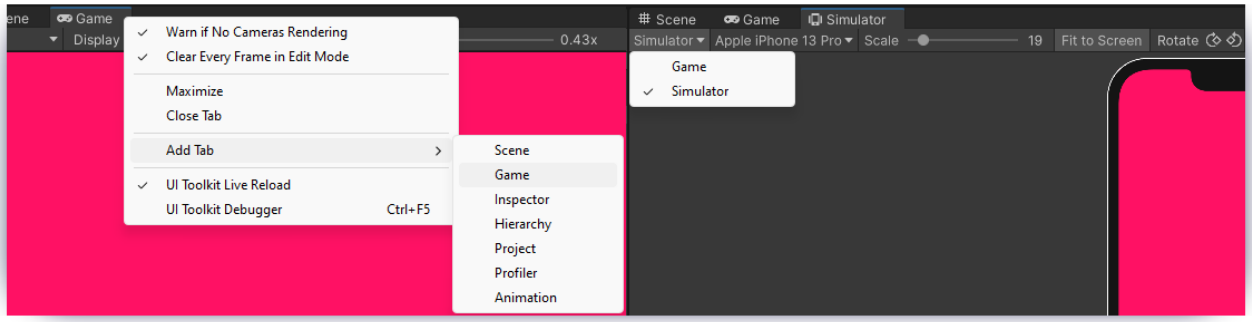

Рисунок 2.10. Додавання симулятора до редактора Unity

Налаштувань не так вже і багато, тому швидко розглянемо кожне:

- Цільовий пристрій - вибираємо пристрій, який нас цікавить і йдемо далі;
- Scale — збільшення або зменшення вигляду;
- FitToScreen - підігнати Scale за розміром екрана;
- Rotate – повернути пристрій;

- Play Mode – Focused/Unfocused. Вибираємо першу, якщо хочемо, щоб при старті гра запускалася у вікні симулятора; друге (за замовчуванням) у вікні Game;
- Control Panel – корисність цієї вкладки важко переоцінити. Вона відкриває панель налаштувань симульованого пристрою, в якій можна змінити мову системи та встановити доступ до мережі.

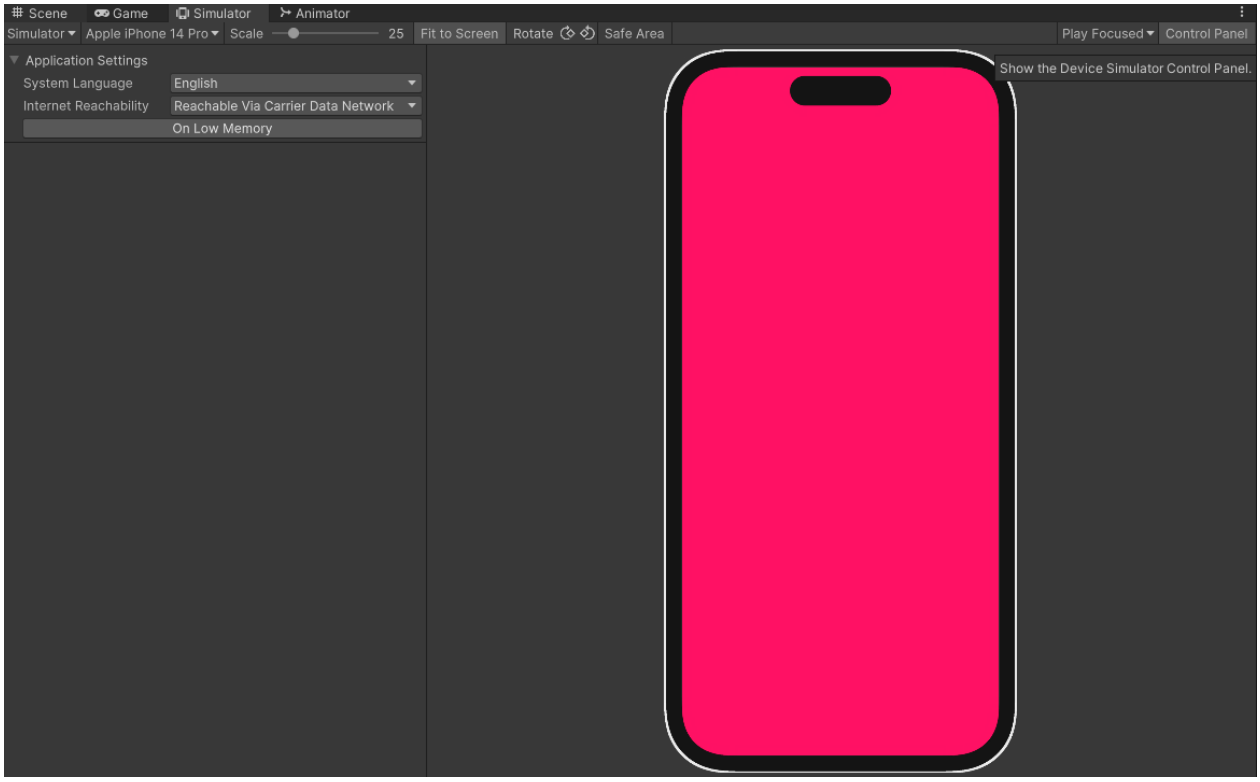

Рисунок 2.11. Інтерфейс симулятора Unity Simulator

2.4. Мова програмування C#

На ринку є десятки популярних мов програмування, і мережа переповнена статтями для новачків. C# - потужна мова з купою можливостей. Мова досить сучасна, всі найпопулярніші фічі, які є на ринку в ній або є, або плануються. Синтаксис простіший ніж у Java, важчий ніж у сучасних мовах на зразок Котліна. Код при компіляції перетворюється на проміжну мову, яку виконує JIT-компілятор на клієнтській машині - у цьому є свої переваги, але такий принцип роботи не залишає шансів писати настільки ж швидкий код, як на плюсах чи голанзі. Але C# все ще досить швидкий у своїй категорії. Якщо просто сісти і писати нехитрий код,

дуже довго не впиратимешся в якісь несподівані проблеми у продуктивності самої мови.

Мова не супер швидка - але є величезний простір для оптимізації. Структури - щоб зберігати дані на стеку, АПІ збирача сміття, щоб оптимізувати його роботу в конкретних кейсах, є unsafe - щоб попрацювати з покажчиками та оптимізувати якийсь ботлнек. Багаті інструменти для паралелізму. Є інтринсики для оптимізації під конкретні процесори.

У C# потужна підтримка ООП - тут будь-який код лежить у класах, повний супорт успадкування, а тепер і множинного успадкування через дефолтну реалізацію інтерфейсів. Купа різних модифікаторів доступу, можливість розділяти відповідальність модулів на рівні збирання. Типізація строга статична - все що можна виконується на етапі компіляції, все що не можна - метаінфа про типи їде в рантайм, і може бути опрацьована там. Сам дизайн мови ніби спеціально зроблений так, щоб опрацьовувати архітектуру. У C# дуже легко пояснити компілятору, які частини системи роблять це, а якісь те, і хто з них про кого знає.

Набагато менше мова C# підтримує функціональну парадигму. Але й тут є чіткий рух уперед. Найпоширеніший у дотнеті спосіб роботи з колекціями – LINQ вирази – виконаний повністю у функціональному стилі.

Але основну проблему мови це лише посилює. Для сучасної мови програмування C# дуже багатослівна. Вона має нові функціональні фічі, але величезна кількість необхідних синтаксичних надбудов не дозволить їх використовувати так само витончено, як у якомусь OCaml. Тут що далі, то гірше. Страшний вантаж зворотної сумісності легко розрулюється творцями мови технічно, але завжди виливається в ще одне обтяження синтаксису. Код мінімально робочої програми на C# - мова буквально відмовляється звільняти мене від написання очевидного.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Концепція мобільної гри «Drag Racing»

Концепція і дизайн мобільної гри полягає в тому, що автомобіль повинен проїхати, якумога більшу відстань. При цьому на дорозі динамічно з'являються перешкоди, які необхідно оминати повз інакше «гра закінчиться». Також потрібно стежити за шкалою палива і збирати на дорозі паливні баки, після використання всього палива гра теж зупиняється (рис. 3.1).

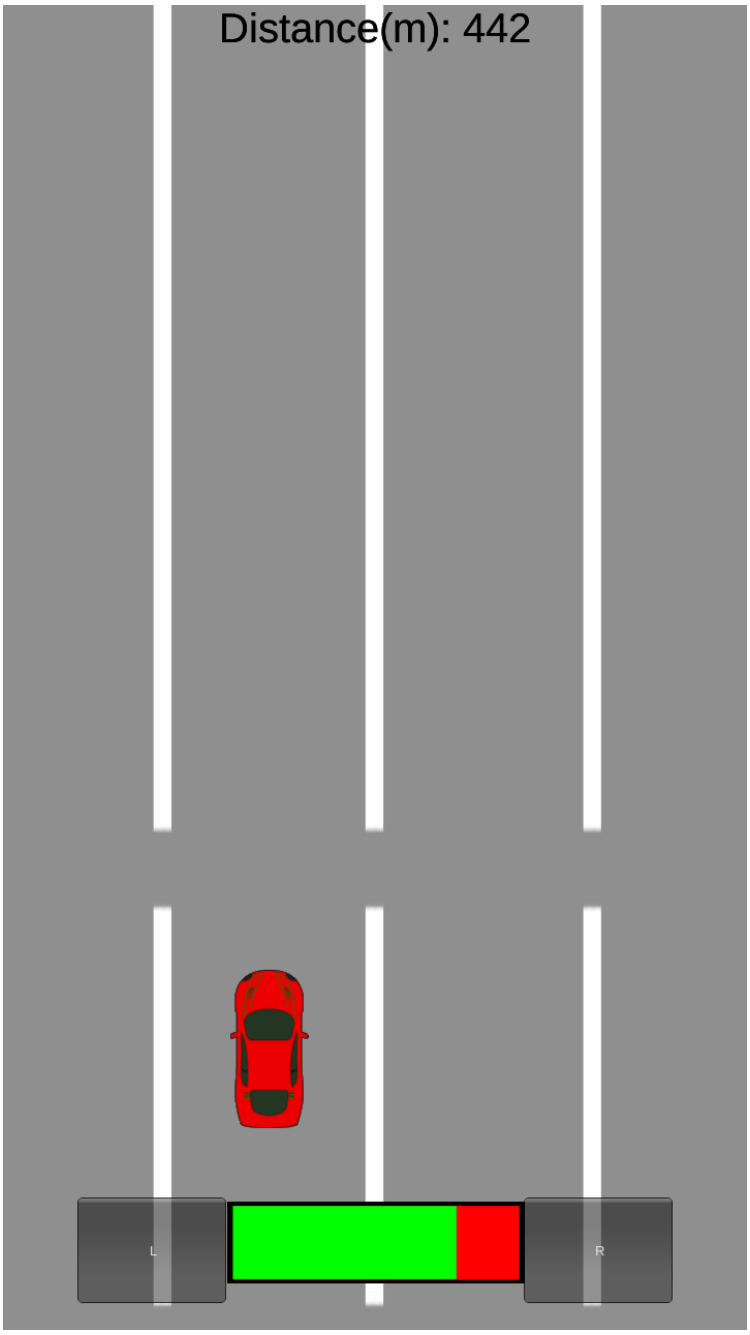

Рисунок 3.1. Дизайн мобільної гри «Drag Racing»

3.2. Розроблення головного меню гри

Гра складається з двох сцен: головне меню та сама ігрова сцена. Де "menu" це головне меню, а "1" це ігрова сцена (рис. 3.2).

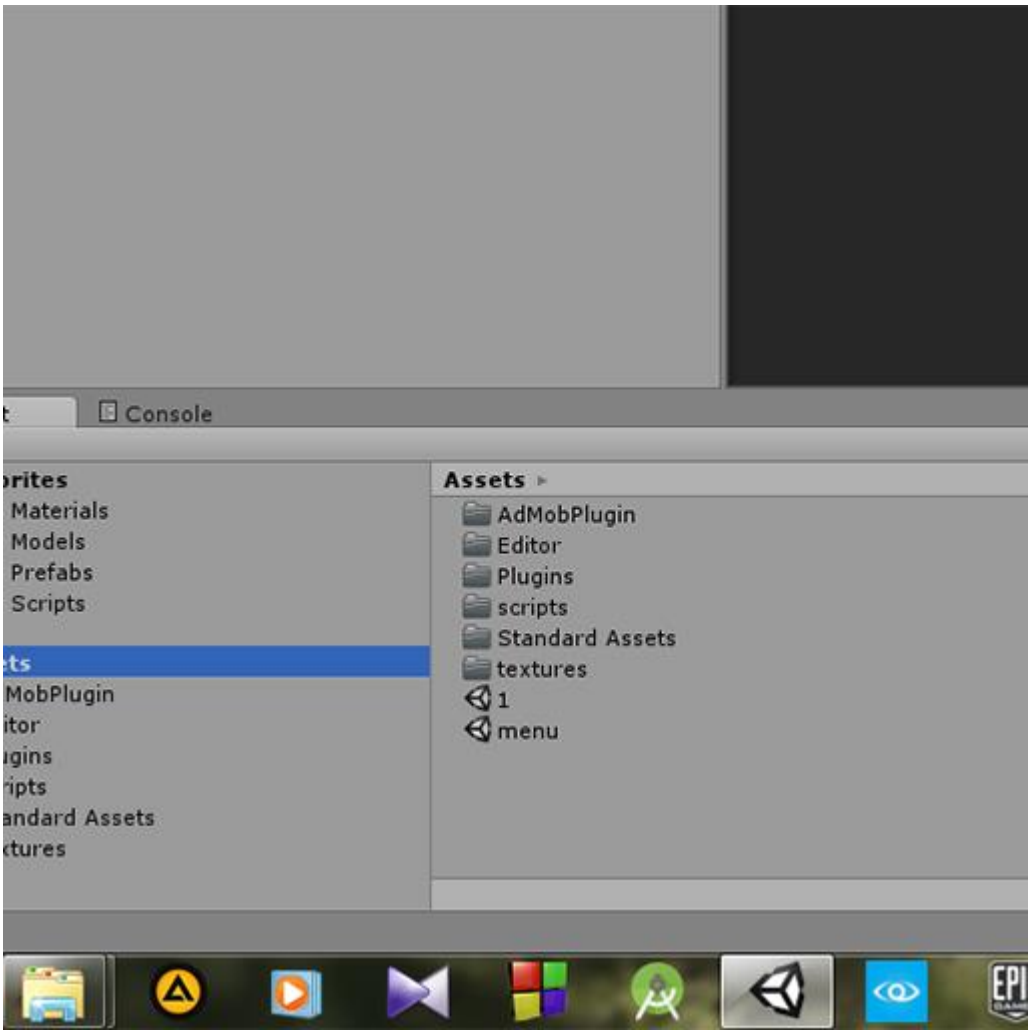

Рисунок 3.2. Структура проекту гри

Для створення простого головного меню гри нам знадобиться елемент управління GUI, який є стандартним у Unity (рис. 3.3).

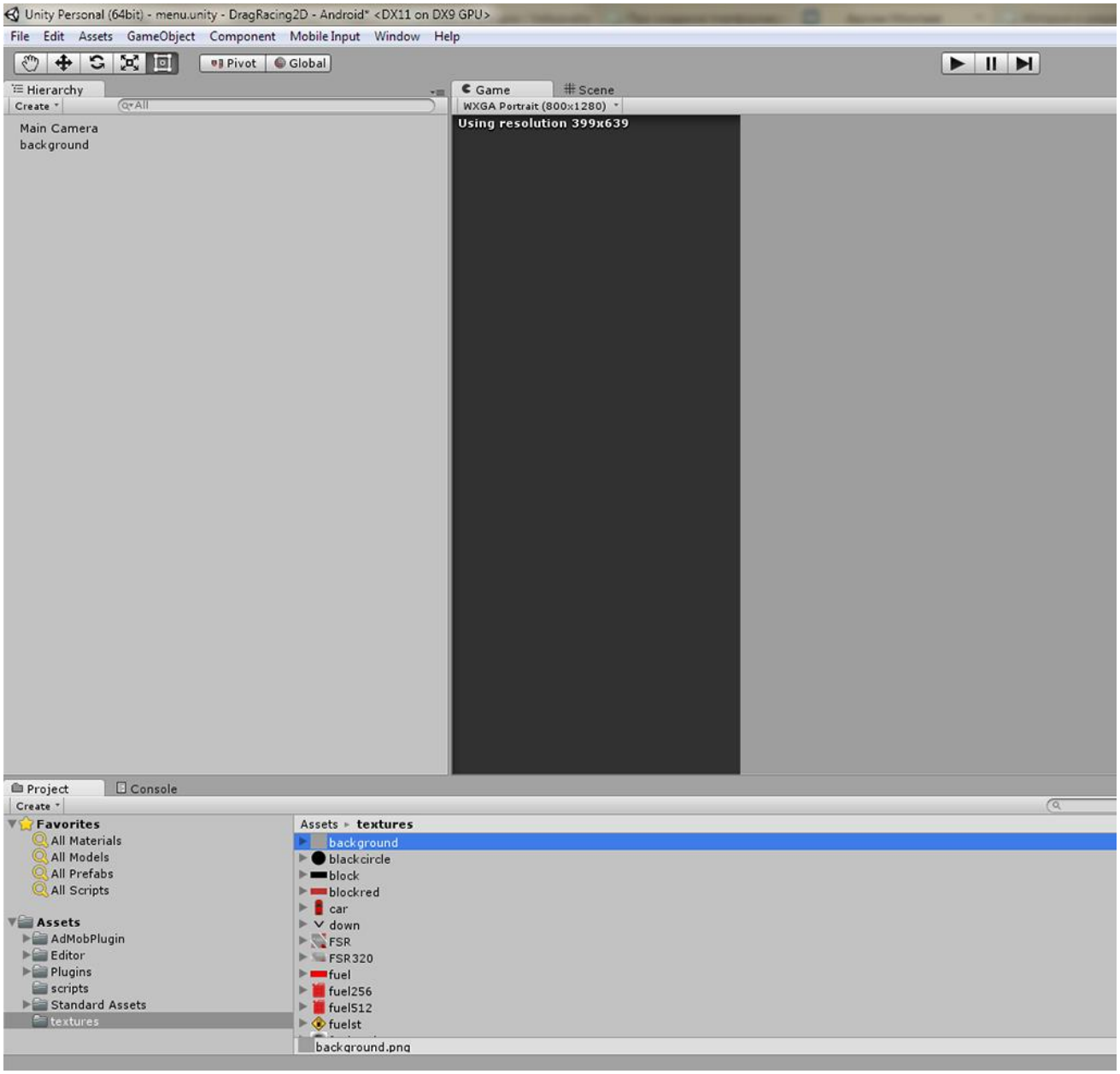

Рисунок 3.3. Елемент управління GUI

В якості фону для головного меню гри ми використали спрайт з ім'ям «background», який заповнений сірим кольором. При потребі можна вибрати будь який колір (рис. 3.4).

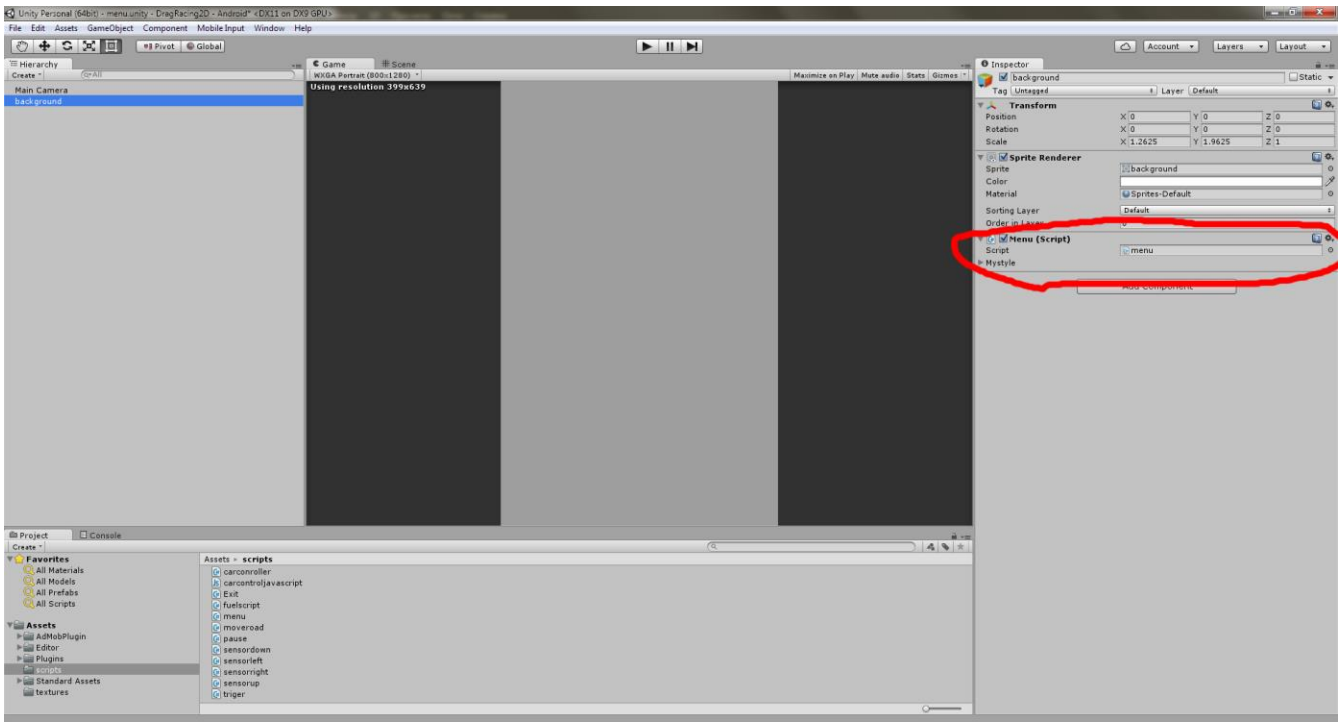

Рисунок 3.4. Вибір фону для головного меню гри

Далі створюємо скрипт «menu.cs» (клацаємо правою кнопкою->вибираємо Create-> C# Script) і прикріплюємо його на background.

Вміст скрипту:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

public class menu : MonoBehaviour {

	public GUIStyle mystyle; //оголошується для того щоб змінювати зображення
GUI компонентів(шрифт, розмір і т.д.)
	string score; //змінна для зберігання пройденої дистанції
	void Start ()
	{
		StreamReader scoredata = new StreamReader
(Application.persistentDataPath + "/score.gd"); //створення файлової змінної
		score = scoredata.ReadLine (); //зчитування рядка
		scoredata.Close (); //закривання файлової змінної
	}


	void Update () {

	}
	void OnGUI(){
		GUI.Box (new Rect (Screen.width*0.15f, Screen.height*0.8f,
Screen.width*0.7f, Screen.height*0.1f), "MAX DISTANCE:"+score,mystyle); //створюємо
невелике вікно для відображення пройденної відстані
		if (GUI.Button (new Rect (Screen.width*0.15f, Screen.height*0.25f,
Screen.width*0.7f, Screen.height*0.1f), "Start game",mystyle)) //створюємо кнопку для
запуску ігрової сцени
		{
			Application.LoadLevel(1);//Завантаження ігрової сцени
		}
		if (GUI.Button (new Rect (Screen.width*0.15f, Screen.height*0.4f,
Screen.width*0.7f, Screen.height*0.1f), "Exit",mystyle)) //створюємо кнопку для
виходу з гри
		{
			Application.Quit();//Вихід з гри
		}
	}
}
```

В результаті виконання вмісту скрипта має вийти приблизно ось так:

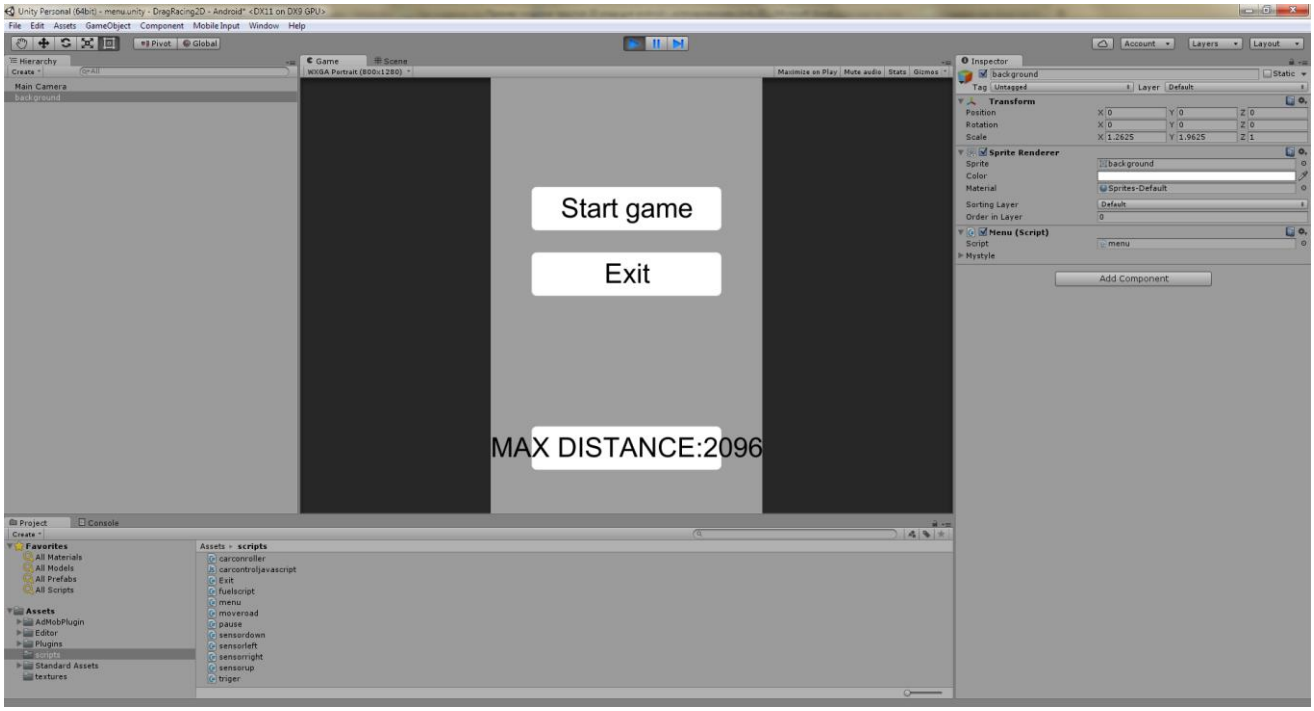

Рисунок 3.5. Реалізація скрипта «menu.cs»

Шрифт, колір та розмір GUI елементів можна змінити за допомогою MyStyle.

Рисунок 3.6. Головне меню мобільної гри «Drag Racing»

3.3. Створення ігрової сцени

Основними елементами на ігровій сцені є дорога, автомобіль та шкала палива.

1. *Дорога:*

Зважаючи на те, що гонка є нескінченною і завершується лише тоді, коли автомобіль наштовхнеться на перешкоду або закінчиться бензин, то дорога є рухомим об’єктом. Тобто автомобіль може переміщатися вліво або вправо, а ілюзія руху створюється за допомогою рухомої дороги (рис. 3.7).

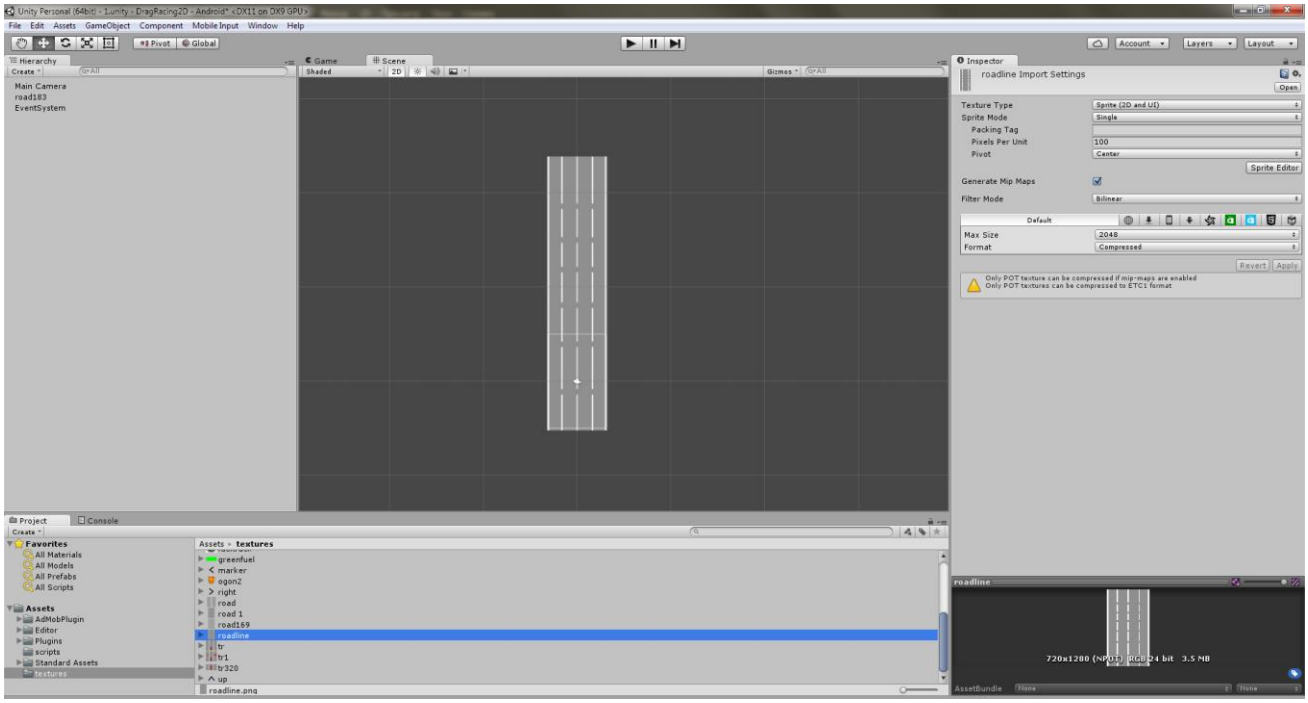

Рисунок 3.7. Створення основного елементу гри – дороги

Переносимо спрайт із дорогою на ігрову сцену та налаштовуємо за розмірами камери (рис. 3. 8).

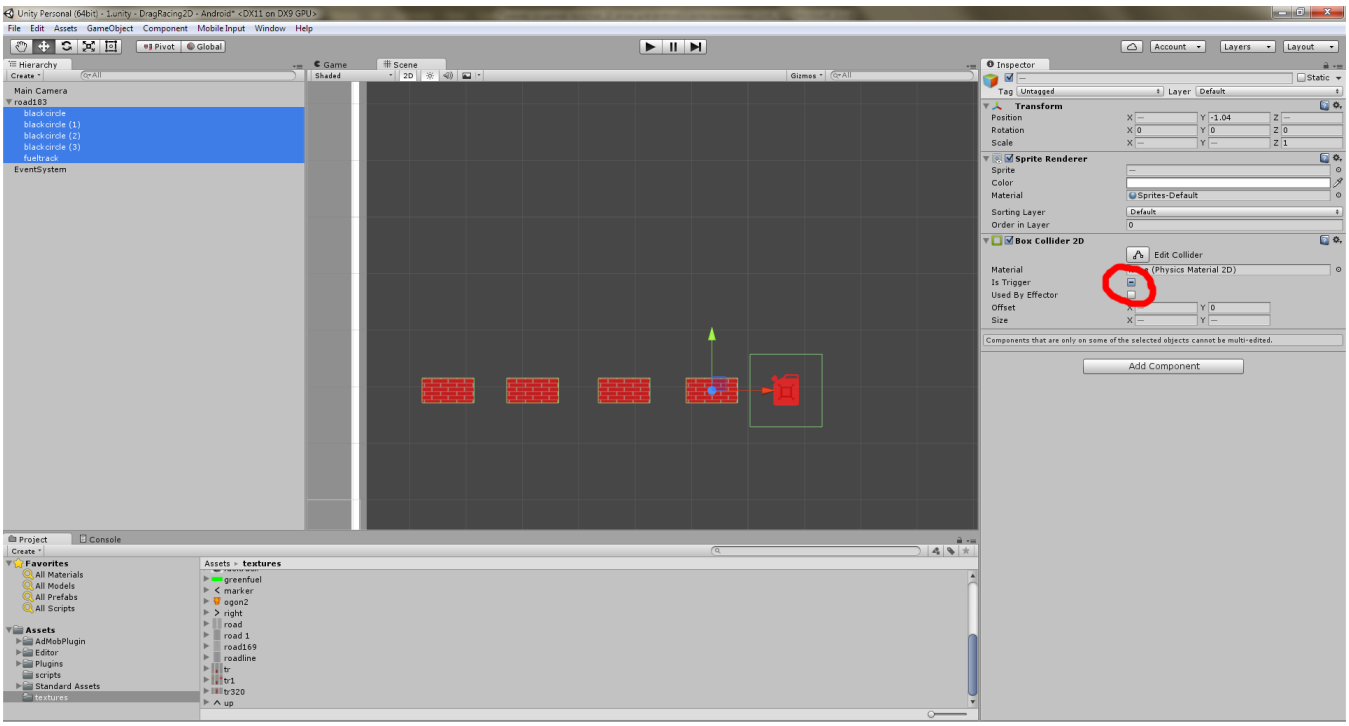

Рисунок 3.8. Налаштування параметрів дороги за розмірами камери

Потім додаємо як дочірні об'єкти всередину дороги 4 блоки з перешкодами, паливний бак і не забуваємо додати до них Box Collider 2D. Ще треба відзначити Is Triger для перетину з автомобілем (рис. 3.9).

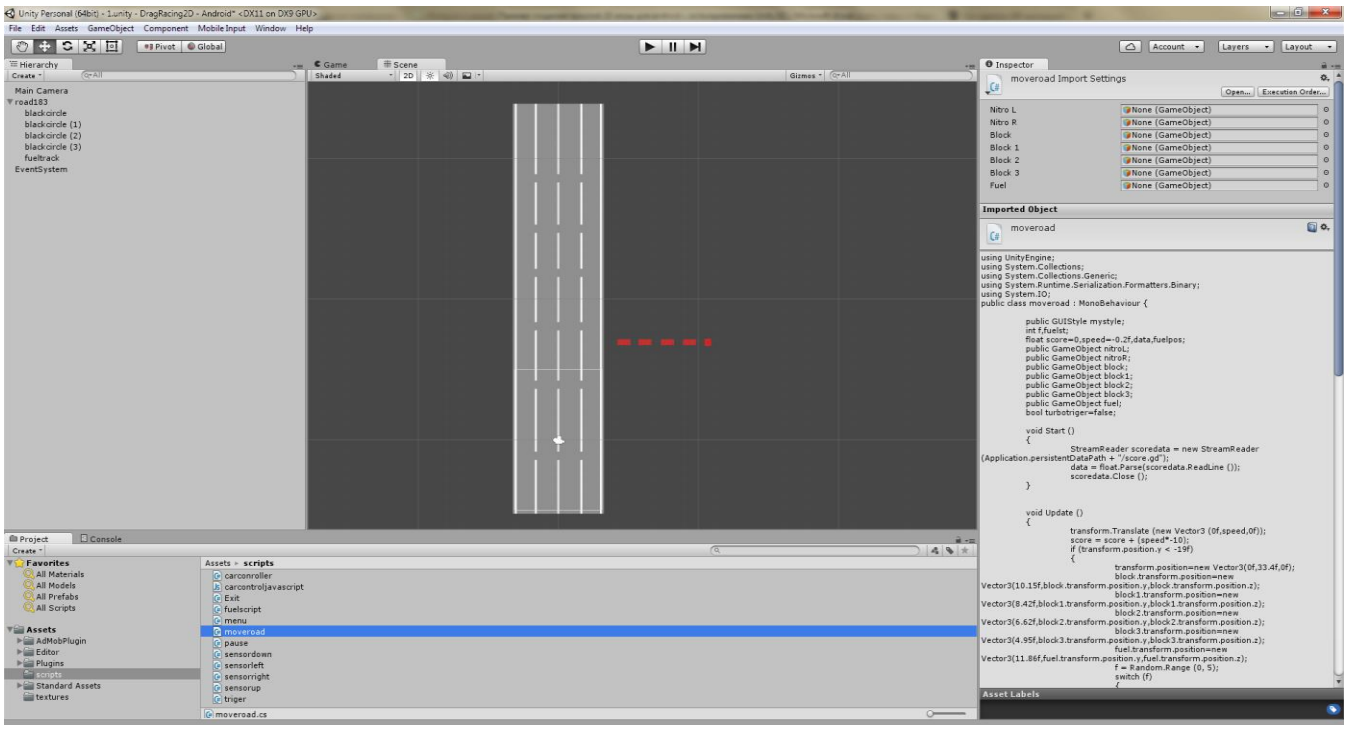

Рисунок 3.9. Додавання на дорогу перешкод та паливного бака

Тепер створюємо скрипт moveroad.cs та завантажуємо його на нашу дорогу. Додаємо до нього наступний код:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
public class moveroad : MonoBehaviour {

        public GUIStyle mystyle;//створення стилю
        int f,fuelst;
        float score=0,speed=-0.2f,data,fuelpos;// змінні для зберігання відстані,
швидкості і рекорду
        public GameObject block;// ігровий об'єкт для розміщення блоку
        public GameObject block1;
        public GameObject block2;
        public GameObject block3;
        public GameObject fuel;
        bool turbotriger=false;

        void Start ()
        {
                StreamReader scoredata = new StreamReader
(Application.persistentDataPath + "/score.gd");
                data = float.Parse(scoredata.ReadLine ());//зчитування з файлу
інформації про рекорд
                scoredata.Close ();
        }


        void Update ()
        {
                transform.Translate (new Vector3 (0f,speed,0f));//рух дороги із
заданою вище швидкістю
                score = score + (speed*-10);// підрахунок відстані
                if (transform.position.y < -19f) // якщо дорога виходить за межі
камери то вона "телепортується" догори
                {
                        transform.position=new Vector3(0f,33.4f,0f);//нова позиція
дороги
                        block.transform.position=new
Vector3(10.15f,block.transform.position.y,block.transform.position.z);
                        block1.transform.position=new
Vector3(8.42f,block1.transform.position.y,block1.transform.position.z);
                        block2.transform.position=new
Vector3(6.62f,block2.transform.position.y,block2.transform.position.z);
                        block3.transform.position=new
Vector3(4.95f,block3.transform.position.y,block3.transform.position.z);
                        fuel.transform.position=new
Vector3(11.86f,fuel.transform.position.y,fuel.transform.position.z);
                       //приховування за межі камери всіх перешкод(блоків)
                        f = Random.Range (0, 5);//випадкова поява на дорозі 1-го з
4-х блоків або каністи з паливом
                        switch (f)
                        {
                        case 0:block.transform.position=new
Vector3(2.40f,block.transform.position.y,block.transform.position.z); break;
                        case 1:block1.transform.position=new
Vector3(0.90f,block1.transform.position.y,block1.transform.position.z); break;
                        case 2:block2.transform.position=new Vector3(-
0.80f,block2.transform.position.y,block2.transform.position.z); break;
                        case 3:block3.transform.position=new Vector3(-
2.35f,block3.transform.position.y,block3.transform.position.z); break;
                        case 4:
                                fuelst=Random.Range(0,4);
                                if(fuelst==0){fuelpos=2.40f;}
                                if(fuelst==1){fuelpos=0.90f;}
                                if(fuelst==2){fuelpos=-0.80f;}
                                if(fuelst==3){fuelpos=-2.35f;}
                                fuel.transform.position=new
Vector3(fuelpos,fuel.transform.position.y,fuel.transform.position.z);
                                break;
                        }
                        if (score>data)// якщо поточна пройдена відстань перевищує
те що записано у файлі рекорду то йде оновлення даних
                        {
                                StreamWriter scoredata=new
StreamWriter(Application.persistentDataPath + "/score.gd");//створюємо файлову змінну
для зберігання пройденої відстані
                                scoredata.WriteLine(score);//записуємо нове
значення у файл
                                scoredata.Close();//закриваємо файлову змінну
                        }
                }

        }
        void OnGUI(){
                GUI.Box (new Rect (0, 0, Screen.width, Screen.height*0.05f),
"Distance(m): " + score,mystyle);//створюємо вікно для підрахунку відстані
        }
}
```

Повинно вийти приблизно так, як показано на рис. 3.10. Якщо все так і залишити то після того, як дорога дійде до кінця, лишиться порожній простір і це буде продовжуватися по колу, дорога зникатиме і з’являтиметься знову.

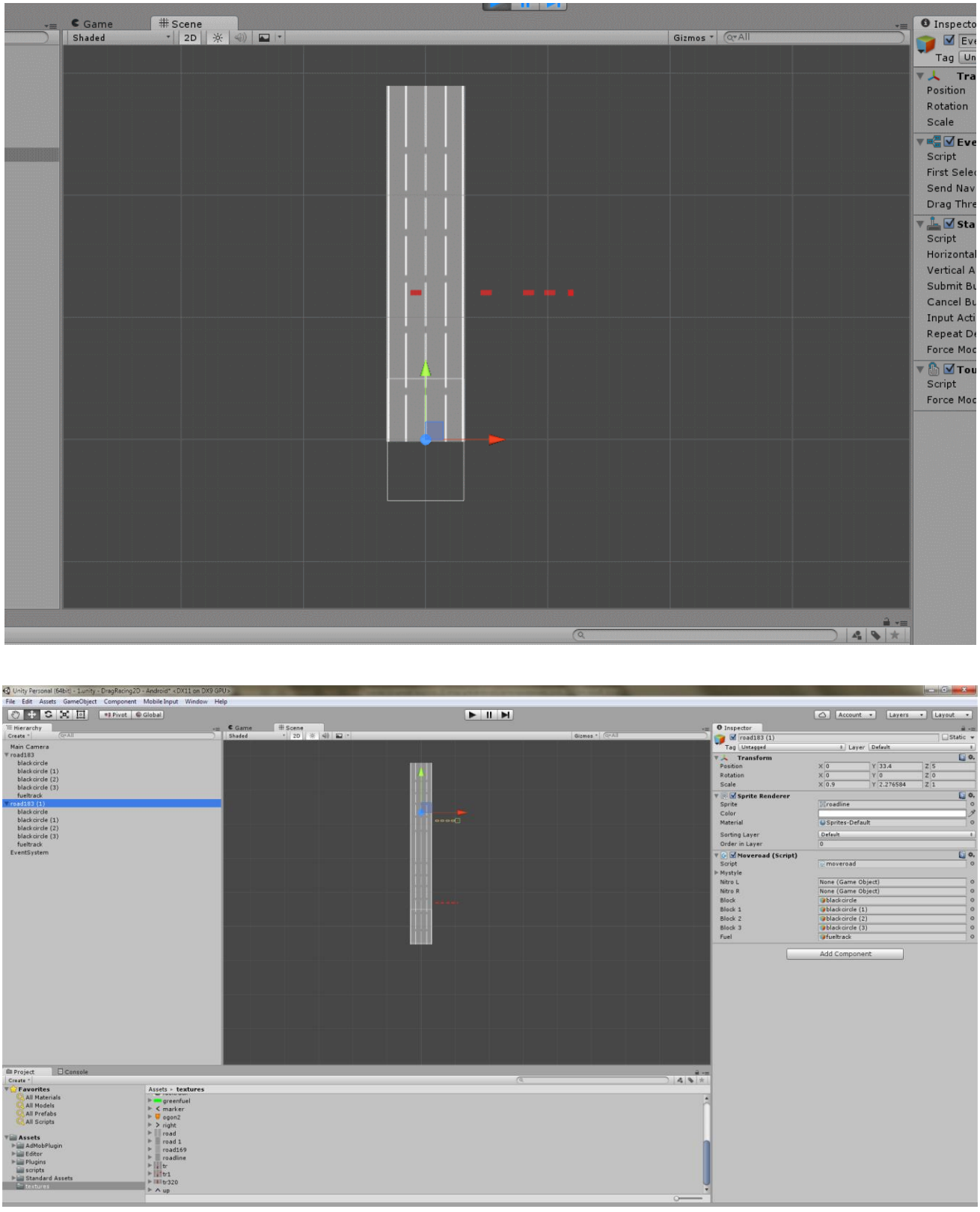

Рисунок 3.10. Реалізація скрипта moveroad.cs

Щоб вирішити цю проблему, треба створити дублікат вже готової дороги і трохи змінити скрипт (рис. 3.11).

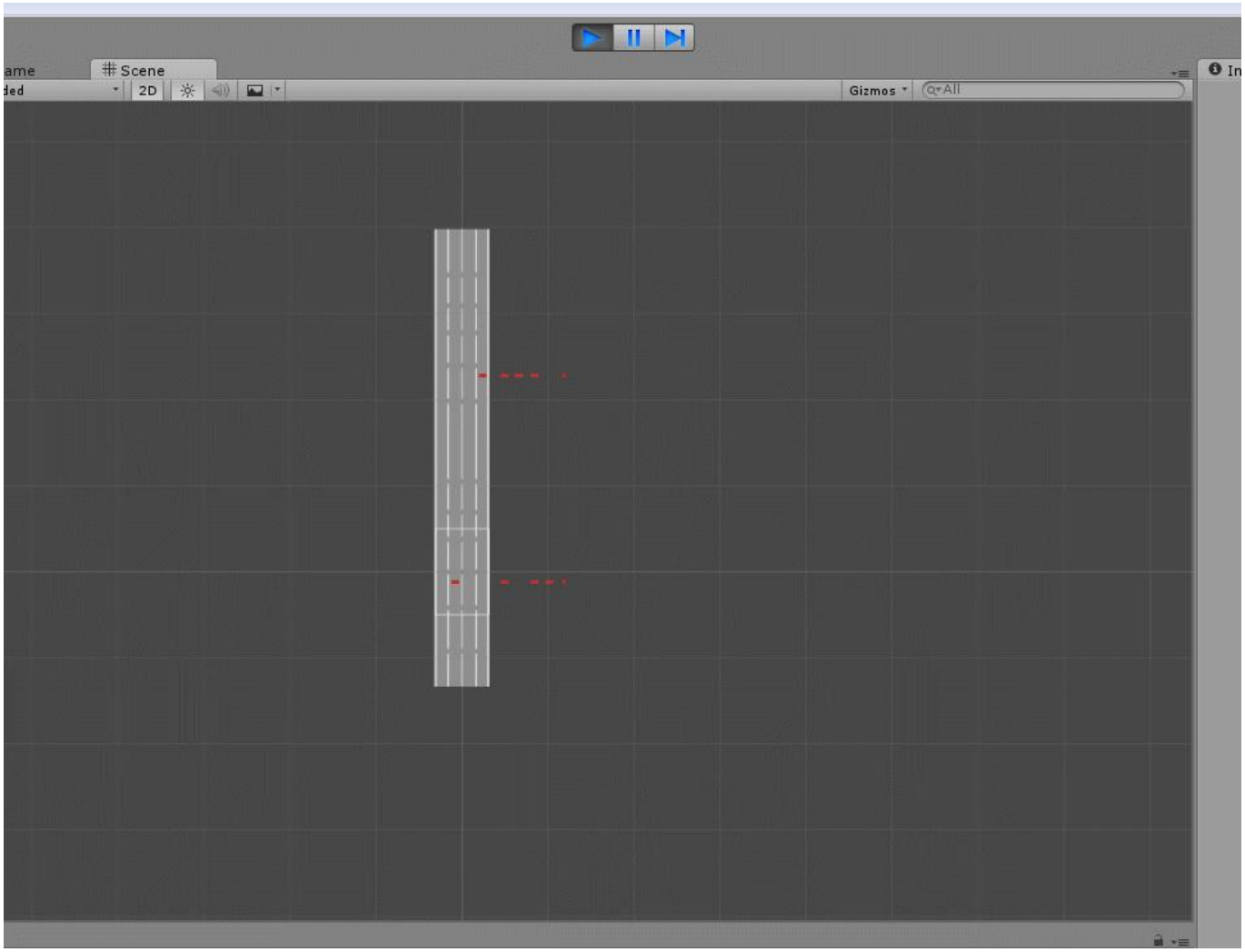

Рисунок 3.11. Основний елемент ігрової сцени (дорога)

2. Автомобіль:

Перетягуємо спрайт автомобіля на сцену та встановлюємо його у будь-яке місце на дорозі (рис. 3. 12). Потім створюємо скрипт carcontroller.cs та прикріплюємо його на автомобіль. Тепер автомобіль може рухатися.

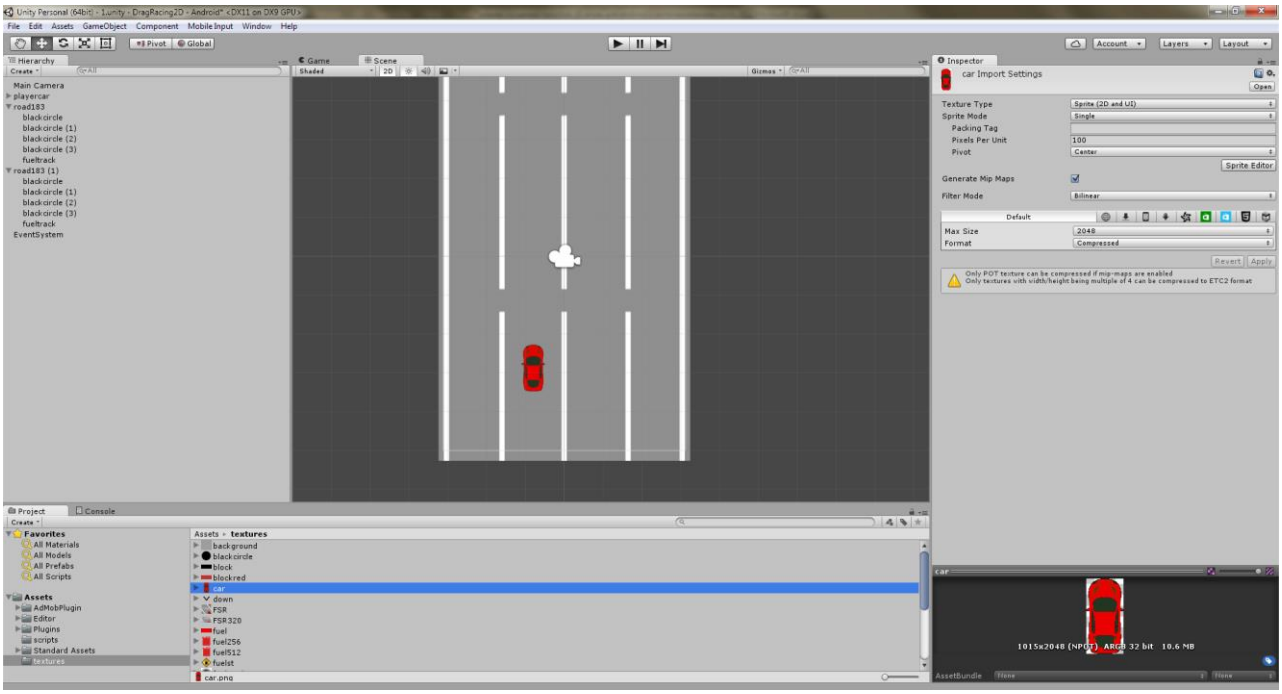

Рисунок 3.12. Створення елементу ігрової сцени – автомобіль

Вміст carcontroller.cs:

```
using UnityEngine;
using System.Collections;
using UnityStandardAssets.CrossPlatformInput;

public class carconroller : MonoBehaviour
{

        void Start ()
        {

        }


        public void Update ()
        {
                if (transform.rotation.z !=0) //перевірка на зіткнення автомобіля і
перешкоди, при зіткнені відбувається завантаження меню
                {
                        Application.LoadLevel (0);
                }

        }
        }
        public void OnGUI()
        {
                if (GUI.RepeatButton (new Rect (Screen.width*0.1f,
Screen.height*0.9f, Screen.width*0.2f, Screen.height*0.08f), "L")) //створюємо кнопку
для руху вліво
                {
                        if (transform.position.x > -2.4f)
                        {
                                transform.Translate (new Vector3 (-0.05f, 0f, 0f));
                        }
                }

                if (GUI.RepeatButton (new Rect (Screen.width*0.7f,
Screen.height*0.9f, Screen.width*0.2f, Screen.height*0.08f), "R")) //створюємо кнопку
для руху вправо
                {
                        if (transform.position.x < 2.4f)
                        {
                                transform.Translate (new Vector3 (0.05f, 0f, 0f));
                        }
                }
        }

}
```

3.Шкала палива:

Для створення шкали палива знадобиться два спрайти одинакових розмірів, але різних кольорів. Один буде зеленого кольору, а інший червоного. І зробити один із них дочірнім. В нашому випадку дочірнім спрайтом буде зелений (рис. 3. 13).

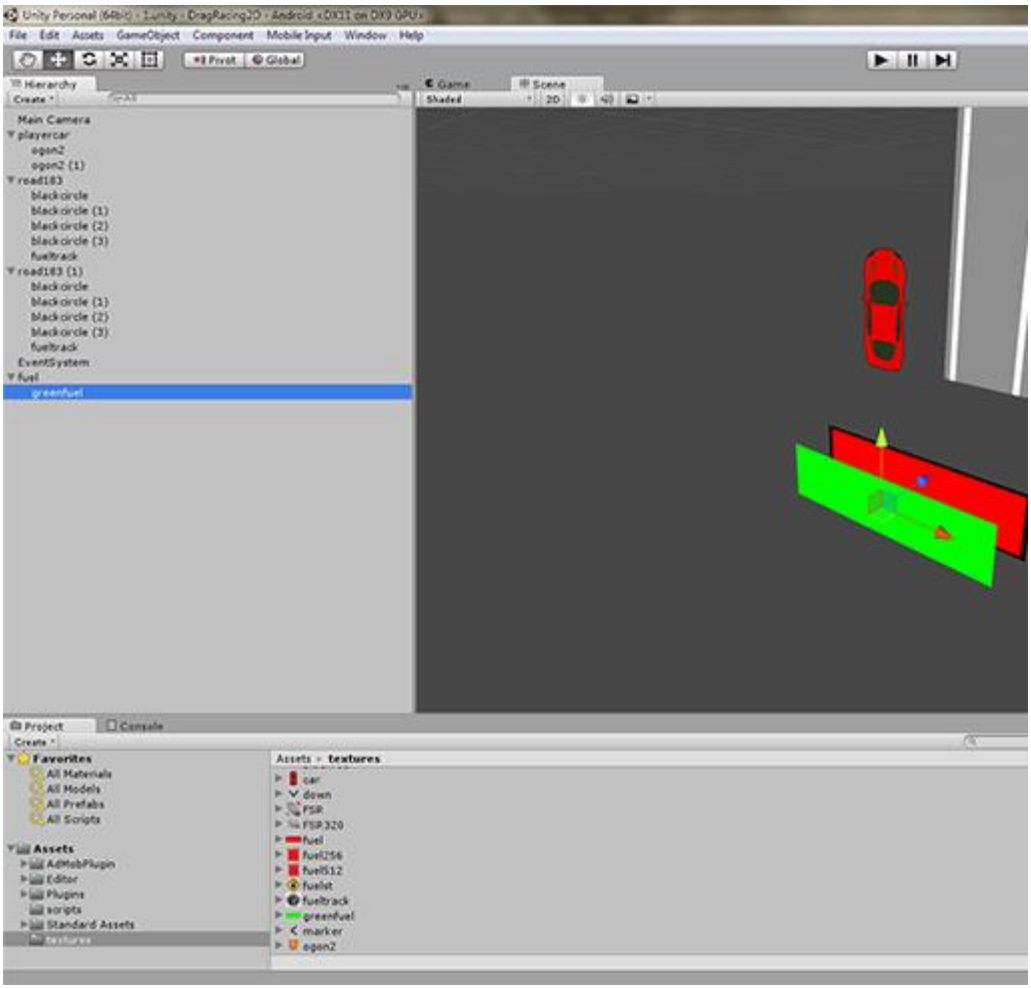

Рисунок 3.13. Створення ігрового елементу сцени – шкали палива

Далі створюємо скрипт fuelscript.cs, прикріплюємо його на fuel і додаємо в нього код:

```
using UnityEngine;
using System.Collections;

public class fuelscript : MonoBehaviour {

        public GameObject fuelall;
        float mytimer=100f;// задання плаваючого числа
        // Use this for initialization
        void Start ()
        {

        }
        void Update ()
        {
                mytimer = 100f;
                mytimer -= Time.deltaTime;//зміна числа з часом
                if (mytimer/mytimer==1f) //перевірка на період часу в 1 секунду
                {
                        fuelall.transform.position=new
Vector3(fuelall.transform.position.x-
0.0011f,fuelall.transform.position.y,fuelall.transform.position.z);
                        fuelall.transform.localScale = new
Vector3(fuelall.transform.localScale.x-0.001f, 1, 1);
                       //вище йде здвиг вліво і зменшення по ширині зеленої полоски
для імітації шкали
                }
                if (fuelall.transform.localScale.x < 0) //якщо шкала зникла то
завантаження йде до головного меню
                {
                        Application.LoadLevel(0);
                }
        }
}
```

Дорога у нас це road183 та її дублікат road183(1). У її дочірній об'єкт fueltrack потрібно додати скрипт для виявлення перетину з автомобілем та заповнення палива.

Створюємо скрипт triger.cs і вішаємо його на fueltrack для обох доріг і відзначаємо як Is Triger.

Код:

```
using UnityEngine;
using System.Collections;

public class triger : MonoBehaviour {


        public GameObject fuel;//додаємо сюди greenfuel
        // Use this for initialization
        void Start () {

        }

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

        }
        void  OnTriggerEnter2D(Collider2D col)
        {
                if (col.gameObject.name == "playercar") //перевірка перетину
автомобіля і об’єкта fuel
                {
                        fuel.transform.position=new
Vector3(0,fuel.transform.position.y,fuel.transform.position.z);
                        fuel.transform.localScale = new Vector3(1, 1, 1);
                       //відновлення у об’єкта fuel стандартних значень
                }
        }


}
```

За відсутності професійного художника, з іконкою для мобільної гри довелося працювати самостійно:

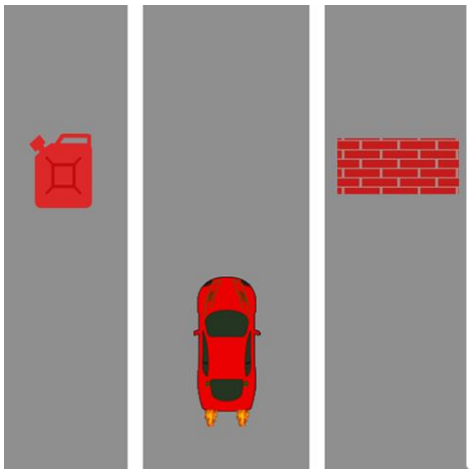

Рисунок 3.14. Іконка мобільної гри «Drag Racing»

3.4. Тестування мобільної гри «Drag Racing»

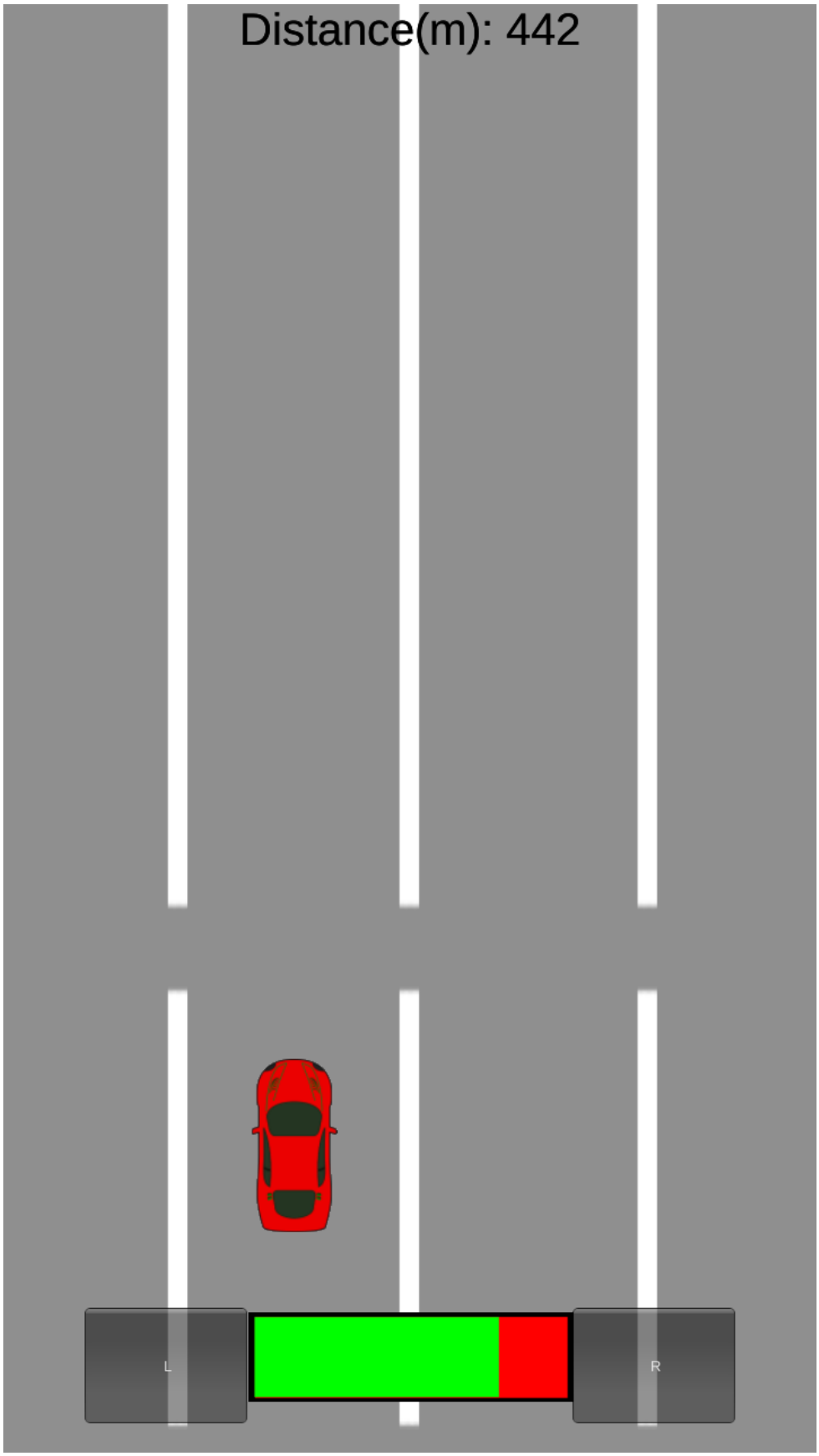

Рисунок 3.15. Тестування мобільної гри «Drag Racing»

3.5. Вимоги до програмного та апаратного забезпечення

Дана мобільна гра «Drag Racing» розроблена для мобільних телефонів, які працюють на операційній системі Android та мають в наявності сенсорний екран. Телефон має відповідати усім технічним вимогам і володіти необхідним програмним забезпеченням. Ця гра не потребує безпровідного зв’язку. Основною вимогою для успішної і безперебійної роботи даного застосунку являється наявність операційної системи Android 4. 1 та вище.

Це гра, яка відноситься до жанру нескінченних аркадних гонок. Тримайте свою машину на дорозі і уникайте перешкод на шляху, збирайте бензин, щоб ваша машина не зупинилася. Постарайся пройти якомога більше дистанції. Отже для успішної роботи мобільної гри потрібні наступні характеристики програмного та апаратного забезпечення, які наводяться в таблиці 3.1.

Таблиця 3.1 Основні характеристики технічного та системного забезпечення

Операційна система	Android 2.3.2+ (Gingerbread, API 9) та вище.
Екран	Сенсорний з діагоналлю не менше 4 дюймів
Пам'ять	Не менше 10.1 MB
Архітектура	armeabi-v7a
Оснащення	Наявність смартфону або планшету
Мова інтерфейсу	Англійська
Категорія	Симулятор
Жанр	Гонки
Вікові обмеження	Everyone

ВИСНОВКИ

Дана дипломна робота присвячена проектуванню та розробці мобільної гри «Drag Racing» для пристроїв на операційній системі Android. Під час виконання дипломної роботи було проаналізовано основні тенденції у розробці мобільних застосунків, розглянуто аналоги даної гри, виявлено їх недоліки, на основі яких створено нову мобільну гру згідно із заявленим технічним завданням. Було розроблено графічне забезпечення гри, структуру гри, та саму гру. Проведене тестування розробленої мобільної гри, а також були розробленні інструкції щодо використання гри. Для розроблення гри використано багатоплатформовий інструмент Unity та об'єктно-орієнтовану мову програмування C#.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Game Engine // Unity. – [Електронний ресурс]. – Режим доступу: http://unity3d.com (дата звернення: 05.06.2023).
2. Hocking J. Unity in Action: Multiplatform Game Development in C#. – 2nd ed. – Shelter Island: Manning Publications, 2018. – 400 p.
3. A* Pathfinding Project. – [Електронний ресурс]. – Режим доступу: http://arongranberg.com/astar/ (дата звернення: 05.06.2023).
4. Become a Developer // Brackeys. – [Електронний ресурс]. – Режим доступу: http://brackeys.com/ (дата звернення: 05.06.2023).
5. Games software revenues to reach $110 billion by 2018. – [Електронний ресурс]. – Режим доступу: http://www.gamesindustry.biz/articles/2015- 05-04-games-software-revenues-to-reachusd110-billion-by-2018-digi-capital (дата звернення: 05.06.2023).
6. Bethke, E. Game development and production [Text] / E. Bethke. – Wordware Publishing, Inc, 2003.– 415 c. – ISBN 1-55622-951-8.
7. Rogers, S. Level Up [Text] / S. Rogers.– John Wiley & Sons, Ltd, 2010. – 492 c. – ISBN 978-0-470-68867-0.
8. Unity, Source 2, Unreal Engine 4, or CryENGINE- Which Game Engine Should I Choose? – [Електронний ресурс]. – Режим доступу: http://blog.digitaltutors.com/unity-udk-cryengine-gameengine-choose/ (дата звернення: 05.06.2023).
9. Physics engines survey results. – [Електронний ресурс]. – Режим доступу: http://bulletphysics.org/wordpress/?p=88 (дата звернення: 05.06.2023).
10. Another million unity developers in the house. – [Електронний ресурс]. – Режим доступу: http://blogs.unity3d.com/2013/07/09/anothermillion-unity-developers-in-the-house (дата звернення: 05.06.2023).
11. Dickinson M. Beginning Unity 2021 Game Development: Create Amazing Games with C#, Apress, 2021. – 412 p.

12. Gibson J., Halpern B. Learning Unity 2020: Build 3D Games with Unity and C#, Packt Publishing, 2020. – 356 p.
13. Murray J. C# Game Programming Cookbook for Unity 3D, CRC Press, 2021. – 520 p.
14. Unity Technologies. Unity Manual. – [Електронний ресурс]. – Режим доступу: https://docs.unity3d.com/Manual/index.html (дата звернення: 15.04.2023).
15. Nystrom R. Game Programming Patterns, 2021. – 354 p.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного коду скрипта «menu.cs»

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

public class menu : MonoBehaviour {

	public GUIStyle mystyle;
	string score;
	void Start ()
	{
		StreamReader scoredata = new StreamReader
(Application.persistentDataPath + "/score.gd");
		score = scoredata.ReadLine ();
		scoredata.Close ();
	}


	void Update () {

	}
	void OnGUI(){
		GUI.Box (new Rect (Screen.width*0.15f, Screen.height*0.8f,
Screen.width*0.7f, Screen.height*0.1f), "MAX DISTANCE:"+score,mystyle);
		if (GUI.Button (new Rect (Screen.width*0.15f,
Screen.height*0.25f, Screen.width*0.7f, Screen.height*0.1f), "Start
game",mystyle
		{
			Application.LoadLevel(1);
		}
		if (GUI.Button (new Rect (Screen.width*0.15f,
Screen.height*0.4f, Screen.width*0.7f, Screen.height*0.1f), "Exit",mystyle))
		{
			Application.Quit();
		}
	}
}
```

ДОДАТОК Б

Лістинг програмного коду скрипта moveroad.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
public class moveroad : MonoBehaviour {

	public GUIStyle mystyle;
	int f,fuelst;
	float score=0,speed=-0.2f,data,fuelpos;
	public GameObject block;
	public GameObject block1;
	public GameObject block2;
	public GameObject block3;
	public GameObject fuel;
	bool turbotriger=false;

	void Start ()
	{
		StreamReader scoredata = new StreamReader
(Application.persistentDataPath + "/score.gd");
		data = float.Parse(scoredata.ReadLine ());
		scoredata.Close ();
	}


	void Update ()
	{
		transform.Translate (new Vector3 (0f,speed,0f));
		score = score + (speed*-10);
		if (transform.position.y < -19f)
		{
			transform.position=new Vector3(0f,33.4f,0f);
			block.transform.position=new
Vector3(10.15f,block.transform.position.y,block.transform.position.z);
			block1.transform.position=new
Vector3(8.42f,block1.transform.position.y,block1.transform.position.z);
			block2.transform.position=new
Vector3(6.62f,block2.transform.position.y,block2.transform.position.z);
			block3.transform.position=new
Vector3(4.95f,block3.transform.position.y,block3.transform.position.z);
			fuel.transform.position=new
Vector3(11.86f,fuel.transform.position.y,fuel.transform.position.z);
			f = Random.Range (0, 5);
			switch (f)
```

```
                    {
                    case 0:block.transform.position=new
Vector3(2.40f,block.transform.position.y,block.transform.position.z); break;
                    case 1:block1.transform.position=new
Vector3(0.90f,block1.transform.position.y,block1.transform.position.z);
break;
                    case 2:block2.transform.position=new Vector3(-
0.80f,block2.transform.position.y,block2.transform.position.z); break;
                    case 3:block3.transform.position=new Vector3(-
2.35f,block3.transform.position.y,block3.transform.position.z); break;
                    case 4:
                           fuelst=Random.Range(0,4);
                           if(fuelst==0){fuelpos=2.40f;}
                           if(fuelst==1){fuelpos=0.90f;}
                           if(fuelst==2){fuelpos=-0.80f;}
                           if(fuelst==3){fuelpos=-2.35f;}
                           fuel.transform.position=new
Vector3(fuelpos,fuel.transform.position.y,fuel.transform.position.z);
                           break;
                    }
                    if (score>data)
                    {
                           StreamWriter scoredata=new
StreamWriter(Application.persistentDataPath + "/score.gd");
                           scoredata.WriteLine(score);
                           scoredata.Close();
                    }
             }

       }
       void OnGUI(){
              GUI.Box (new Rect (0, 0, Screen.width, Screen.height*0.05f),
"Distance(m): " + score,mystyle);
       }
}
```

ДОДАТОК В

Лістинг програмного коду скрипта carcontroller.cs

```
using UnityEngine;
using System.Collections;
using UnityStandardAssets.CrossPlatformInput;

public class carconroller : MonoBehaviour
{

	void Start ()
	{

	}


	public void Update ()
	{
		if (transform.rotation.z !=0)
		{
			Application.LoadLevel (0);
		}

	}
	}
	public void OnGUI()
	{
		if (GUI.RepeatButton (new Rect (Screen.width*0.1f,
Screen.height*0.9f, Screen.width*0.2f, Screen.height*0.08f), "L"))
		{
			if (transform.position.x > -2.4f)
			{
				transform.Translate (new Vector3 (-0.05f, 0f,
0f));
			}
		}

		if (GUI.RepeatButton (new Rect (Screen.width*0.7f,
Screen.height*0.9f, Screen.width*0.2f, Screen.height*0.08f), "R"))
		{
			if (transform.position.x < 2.4f)
			{
				transform.Translate (new Vector3 (0.05f, 0f,
0f));
			}
		}
	}
```

```
}
```

ДОДАТОК Г

Лістинг програмного коду скрипта fuelscript.cs

```
using UnityEngine;
using System.Collections;

public class fuelscript : MonoBehaviour {

	public GameObject fuelall;
	float mytimer=100f;
	// Use this for initialization
	void Start ()
	{

	}
	void Update ()
	{
		mytimer = 100f;
		mytimer -= Time.deltaTime;
		if (mytimer/mytimer==1f)
		{
			fuelall.transform.position=new
Vector3(fuelall.transform.position.x-
0.0011f,fuelall.transform.position.y,fuelall.transform.position.z);
			fuelall.transform.localScale = new
Vector3(fuelall.transform.localScale.x-0.001f, 1, 1);

		}
		if (fuelall.transform.localScale.x < 0)
		{
			Application.LoadLevel(0);
		}
	}
}
```

ДОДАТОК Д

Лістинг програмного коду скрипта triger.cs

```
using UnityEngine;
using System.Collections;

public class triger : MonoBehaviour {


	public GameObject fuel;
	// Use this for initialization
	void Start () {

	}

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

	}
	void  OnTriggerEnter2D(Collider2D col)
	{
		if (col.gameObject.name == "playercar")
		{
			fuel.transform.position=new
Vector3(0,fuel.transform.position.y,fuel.transform.position.z);
			fuel.transform.localScale = new Vector3(1, 1, 1);

		}
	}


}
```