Міністерство освіти і науки України
Український державний університет науки і технологій

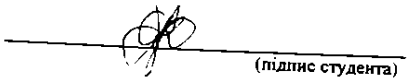
Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

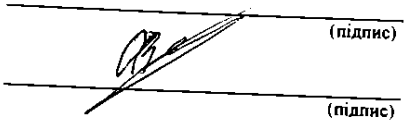
Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Розробка мобільного 3-D додатку X-Racer. Інтерфейс користувача та тестування»
за освітньою програмою «**12 Інженерія програмного забезпечення**»
зі спеціальності: «**121 Інженерія програмного забезпечення**»
Виконав: студент групи «ПЗ1912»


(підпис студента) /Кирило СКОРИК/
(Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) /доц. Вадим АНДРЮЩЕНКО/
(посада, Ім'я ПРІЗВИЩЕ)


Нормоконтролер: (підпис) /Світлана ВОЛКОВА/
(посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент 
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis
master

on the topic: « Development of the X-Racer mobile 3-D application. User interface and testing »

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ1912: ___ _____ /Kyrylo SKORYK/
(посада) (підпис)

Scientific Supervisor: ___ _____ /Vadym ANDRIUSHCHENKO/
(посада) (підпис)

Normative controller: ___ _____ / Svitlana VOLKOVA/
(посада) (підпис)

Supervisors:
Economic part ___ _____ / Mykola GNENNIYN/
(посада) (підпис)

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ грудня 202_ р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Бакалавр_____

студенту Скорик Кирило Петрович_____

1. Тема дипломної роботи: Розробка мобільного додатку X-Racer.Візуальна частина та взаємодія з UI

   Керівник роботи: Андрющенко Вадим Олександрович

   затверджені наказом 1209 ст від 07.12. 2022 року

2. Строк подання студентом роботи ___.___. 202_ року

3. Вихідні дані до дипломної роботи: _____.

4. Зміст пояснювальної записки (перелік питань до розробки):
   4.1. Аналітична частина: _____;
   4.2. Основна частина: _____;
5. Перелік демонстраційного матеріалу:
   5.1. презентація;
   5.2. демонстраційне відео.

6. Консультанти:

Розділ	Консультант	Завдання видав	Завдання прийняв
Техніко-економічні розрахунки	Микола ГНЕНИЙ	Вадим АНДРЮЩЕНКО	Кирило СКОРИК

КАЛЕНДАРНИЙ ПЛАН

№	азва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	15.01.23 – 16.01.23	1%
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	16.01.23 – 20.02.23	10%
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	20.02.23 – 19.03.23	20%
4	Постановка задачі	19.03.23 – 23.03.23	30%
5	Виконання досліджень інструментів для розробки мобільного додатку	23.03.23 – 20.04.23	60%
6	Оформлення тез доповідей	20.04.23 – 23.04.23	70%
7	Оформлення пояснювальної записки	23.04.23 – 15.06.23	90%
8	Розробка демонстраційних матеріалів	15.06.23 – 19.06.23	100%
9	Подання кваліфікаційної роботи до кафедри	23.06.23	
10	хист кваліфікаційної роботи на засіданні Екзаменаційної комісії	28.06.23	

Студент _____ /Кирило СКОРИК/

Керівник роботи _____ /Вадим АНДРЮЩЕНКО/

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра 121 с., 53 рис. , 3 табл., 15 джерел.

Об’єктом дослідження є розробка мобільних ігор. Предметом дослідження є створення візуального супроводження та його налагодження.

Метою дослідження в контексті даної роботи є виявлення підходящої для прогнозування архітектури нейронних мереж, її реалізація за допомогою бібліотек Numpy та TensorFlow для Python, як найбільш популярних фреймворків для побудови передбачуваних моделей, оцінка точності прогнозів моделі та підбір оптимальних параметрів.

Методи дослідження: використання інструменту Blender для створення 3D та 2D моделей. Подальша інтеграція та налагодження створених моделей у середовищи двигуна Unity

Пояснювальна записка складається зі вступу, п'яти розділів, висновків, бібліографічного списку та тексту програми.

Вступ описує суть, мету та актуальність роботи (1 сторінка).

Перший розділ: аналіз мобільних гоночних ігор, огляд інтсрумента Blender та двигуна Unity (25 сторінок).

Другий розділ: збір вимог для створення мобільного додатку (3 сторінки).

Третій розділ: проектування обь’єктів (4 сторінки).

Четвертий розділ: розробка програми (23 сторінки).

П’ятий розділ: розробка програми (2 сторінки).

ЗМІСТ

ВСТУП

Останніми роками мобільні ігри стрімко зросли і стали помітним прибутковим сегментом ігрової індустрії. Перегони особливо приваблюють гравців своїм адреналіновим досвідом і захоплюючим ігровим досвідом. Однак створення візуально приголомшливих гоночних ігор для мобільних платформ потребує глибокого розуміння інструментів і методів, задіяних у розробці високоякісної графіки та анімації. Ігровий движок Unity та популярне програмне забезпечення для 3D-моделювання Blender забезпечують потужну комбінацію для вражаючих візуальних ефектів у мобільних гоночних програмах.

Основна мета – вивчити та задокументувати процес створення та тестування візуальних ресурсів для мобільної гоночної програми X-Racer за допомогою 3D-моделювання,на основі ігрового движка Unity та Blender. Дослідження спрямоване на вивчення робочих процесів, методологій і найкращих практик, залучених до тестування та ефективної інтеграції цих інструментів для розробки візуально привабливих гоночних ігор на мобільних платформах. Аналізуючи процес розробки та вирішуючи проблеми, що виникли під час впровадження, ця дипломна робота має на меті надати практичні поради розробникам ігор, які хочуть покращити візуальні ефекти своїх мобільних гоночних ігор.

Ця робота присвячена створенню візуальних ресурсів спеціально для мобільних гоночних програм. Він забезпечує глибокий погляд на використання ігрового движка Unity як основної платформи розробки та Blender як основного інструменту 3D-моделювання. Дослідження охоплюють усі аспекти процесу розробки зовнішнього вигляду, включаючи моделювання, накладання текстур, освітлення, ефекти, дизайн інтерфейсу користувача, тестування та оптимізацію. Однак він не охоплює розширені теми програмування, крім створення візуальних ресурсів. Дослідження в першу чергу спрямоване на розробників початкового та середнього рівня, зацікавлених у створенні візуально привабливих мобільних гоночних ігор.

РОЗДІЛ 1
ЗБІР ТА АНАЛІЗ ВИМОГ

1.1 Мобільні програми для перегонів

Поява смартфонів і поширення мобільних пристроїв призвели до значного розширення індустрії мобільних ігор. Із зростанням доступності та доступності смартфонів ігри стали широко поширеною діяльністю, доступною для мільйонів користувачів у всьому світі. Зручність мобільних ігор дозволяє гравцям насолоджуватися грою в будь-який час і в будь-якому місці, що призвело до її величезної популярності.

Рисунок 1.1 Аналіз доходів з ігор згідно Newzoo

Оперуючись на дані з сайту Newzoo(Рисунок 1.1 Аналіз доходів з ігор згідно Newzoo) які провели дослідження згідно доходів з ігор для різних типів пристроїв показують величезний вплив і популярність мобільних ігор. Доходи від мобільних ігор незмінно перевищують решту ігрової індустрії, включаючи ігри для консолей і комп’ютерів. Ці статистичні дані підкреслюють величезний потенціал ринку та величезну базу гравців, яких залучили мобільні ігри.

Зростання мобільних ігор можна пояснити кількома тенденціями. Модель

«freemium», у якій ігри пропонуються безкоштовно, але включають покупки в програмі або рекламу, стала загальною стратегією монетизації. Такий підхід дозволяє розробникам охоплювати ширшу аудиторію та отримувати дохід за рахунок додаткових покупок у самій грі. Крім того, соціальна інтеграція та функції для кількох гравців стали невід'ємними функціями, які полегшують взаємодію та взаємодію між гравцями.

Смартфони зробили революцію в грі, запропонувавши високопродуктивне апаратне забезпечення, розширені графічні можливості та інтуїтивно зрозумілі сенсорні інтерфейси. Широка доступність смартфонів за різними цінами демократизувала ігри, зробивши їх доступними для людей з будь-яким соціально-економічним статусом. Зручність і портативність мобільних пристроїв дозволяють гравцям грати в будь-який час і в будь-якому місці, перетворюючи ігри зі стаціонарної діяльності на мобільний спосіб життя.

Магазини програм, такі як Apple App Store і Google Play Store, відіграють ключову роль у доступності та розповсюдженні мобільних ігор. Ці платформи забезпечують централізований ринок, де розробники можуть демонструвати та поширювати свої ігри серед глобальної аудиторії. Магазин додатків керує процесом транзакцій, надаючи користувачам безперешкодний досвід пошуку, завантаження та оновлення ігор. Легкий доступ до величезної бібліотеки ігор сприяє зростанню індустрії мобільних ігор.

Перегони характеризуються транспортними засобами та змаганнями на швидкість. Зазвичай гравець керує транспортним засобом, таким як мотоцикл або автомобіль, і змагається з суперниками запрограмованими розробниками гри, або з іншими гравцями. Основний ігровий процес полягає в тому, щоб завершити гонки і досягти найшвидшого кола, часто з різними цілями, такими як перемога в чемпіонатах або розблокування нових транспортних засобів і трас. Приплив адреналіну в гоночних іграх створює ефект занурення, який приваблює як звичайних, так і завзятих гравців.

Існує багато різних типів гоночних ігор, кожен з яких пропонує різний ігровий досвід. Перегони в аркадному стилі підкреслюють швидку дію і просте управління, а також можуть включати перебільшену фізику і бонуси. Ці типи ігор роблять акцент на розвагах та азарті, а не на реалістичності. З іншого боку, гоночні симулятори мають на меті забезпечити більш достовірний досвід, відтворюючи реальну фізику та керування автомобілем. Вони орієнтовані на гравців, які хочуть отримати більш реалістичний і складний досвід перегонів. Багатокористувацькі ігри також дозволяють гравцям змагатися один з одним онлайн, заохочуючи соціальну взаємодію та інтенсивну конкуренцію. У своїй роботі я вибрав аркадну тему розробки. Вона була обрана через легкість в управлінні та гарний ігровий опит для користувачів.

Гоночні ігри мають різноманітні механіки та функції, які покращують ігровий процес. Наприклад, транспортні засоби можна робити краще за допомогою різних схем фарбування, наклейок та покращення характеристик. Вони також пропонують різноманітні траси та середовища, від міських до бездоріжжя, що додає різноманітності та виклику ігровому процесу. Крім того, гоночні ігри часто використовують систему прогресії, коли нові моделі автомобілів, траси та режими гри розблоковуються в міру проходження гри. Таблиці лідерів і хронометражі заохочують гравців прагнути до кращого часу проходження кола і змагатися за найвищі позиції з іншими гравцями.

Захоплюючий ігровий досвід в гоночних іграх важливий для того, щоб привернути увагу гравців і утримати їх у грі. Для цього потрібно розробляти траси з цікавим плануванням, додавати складні перешкоди і балансувати між майстерним водінням і прийняттям стратегічних рішень. Це також вимагає ефективної передачі швидкості та азарту перегонів візуально та на слух і занурення гравця у швидкісний екшн. Ефекти навколишнього середовища, такі як погода та зміна дня і ночі, також сприяють зануренню, роблячи перегони

більш реалістичними та захоплюючими.

Візуальні елементи відіграють важливу роль в успіху гоночних ігор, оскільки вони забезпечують захопливий і цікавий досвід для гравця. Візуальні елементи гоночних ігор включають графіку, анімацію, оточення, транспортні засоби, спецефекти та дизайн користувацького інтерфейсу. Разом ці елементи створюють візуальну ідентичність гри, підвищують реалістичність, надають відчуття швидкості та забезпечують естетично приємний досвід. Візуальна якість гоночної гри має значний вплив на залученість і задоволеність гравців, а тому є ключовим елементом у розробці ігор.

Гоночні ігри вимагають високоякісної графіки та анімації, щоб забезпечити реалістичне оточення, деталізовані моделі транспортних засобів і плавний рух. Реалістична графіка посилює занурення в гру, оскільки гравець відчуває себе у світі перегонів. Детальні моделі транспортних засобів, точне фізичне моделювання та реалістична анімація додають грі достовірності та покращують загальний досвід. Плавні та плавні анімації реалістично переміщують транспортні засоби та об'єкти в грі, збільшуючи відчуття швидкості та забезпечуючи візуально приємне видовище для гравця.

Візуальні елементи в гоночних іграх мають не лише естетичне значення, але й відіграють функціональну роль в ігровій механіці. Чіткий, візуально виразний дизайн траси та відповідні знаки і маркери допомагають гравцеві ефективно пересуватися по гоночній трасі. Візуальний зворотний зв'язок, такий як сліди від ковзання шин, іскри та димові ефекти, надають гравцям важливу інформацію про характеристики автомобіля, пошкодження та небезпеку зіткнення. Візуальні підказки відіграють важливу роль у передачі важливої ігрової інформації, допомагаючи гравцям приймати рішення за долі секунди та вдосконалювати свої гоночні навички.

Багато гоночних ігор встановили нові стандарти візуальної досконалості в ігровій індустрії. Ці ігри розширюють межі з точки зору графічної достовірності,

реалістичної фізики та уваги до деталей. Наприклад, такі ігри, як Forza Horizon і Gran Turismo, широко відомі своїми приголомшливими візуальними ефектами, високо деталізованими моделями автомобілів і візуально виразним оточенням. Візуальний успіх цих ігор підвищив очікування гравців від жанру і заохотив розробників постійно вдосконалювати візуальний дизайн і розширювати технічні межі платформи.

Гоночні ігри для мобільних пристроїв пройшли довгий шлях розвитку з часів зародження мобільних ігор. Спочатку, через технічні обмеження, гоночні ігри для мобільних пристроїв мали просту графіку, обмежене управління та базову ігрову механіку. Ці ігри нагадували ігри з прокруткою зверху вниз або вбік, з простим дизайном траси та мінімальними можливостями кастомізації транспортного засобу. Але з розвитком мобільних технологій гоночні ігри значно еволюціонували, пропонуючи більш захоплюючий та візуально приголомшливий досвід.

Удосконалення мобільного обладнання та графічних процесорів відіграло важливу роль в еволюції мобільних гоночних ігор. Потужніші процесори, вдосконалені графічні процесори та більший обсяг оперативної пам'яті дозволили розробникам створювати візуально приголомшливі гоночні ігри, які конкурують з іграми на консолях та ПК. Дисплеї з високою роздільною здатністю, яскрава передача кольору та вища частота оновлення екрану сприяють візуальній достовірності мобільних гоночних ігор, покращуючи загальне занурення та ігровий досвід гравця.

Мобільні гоночні ігри з часом еволюціонували з точки зору ігрового процесу та візуальної достовірності. Розробники впроваджують реалістичні фізичні симуляції, більш складний дизайн трас і вдосконалені системи штучного інтелекту, щоб забезпечити захоплюючий ігровий досвід з ефектом присутності. Впровадження сенсорного і нахильного управління та сумісність із зовнішніми контролерами також покращили ігровий процес, підвищивши точність і

швидкість реагування на керування транспортним засобом.

Візуальні ефекти мобільних гоночних ігор значно покращилися. Розробники використовували складні методи освітлення і затінення, динамічні погодні системи і відображення в реальному часі, щоб створити візуально приголомшливе середовище. Текстури з високою роздільною здатністю, деталізовані моделі транспортних засобів та реалістична анімація сприяють більш захоплюючому та візуально привабливому досвіду. Крім того, просунуті візуальні ефекти, такі як розмиття руху, системи частинок і динамічні кути камери, підсилюють відчуття швидкості та азарту, надаючи гравцям більш захоплюючий досвід перегонів.

Існує багато прикладів успішних мобільних гоночних ігор, які розширюють межі візуальних інновацій у цьому жанрі. Такі ігри, як Asphalt, Real Racing та CSR Racing демонструють еволюцію візуальних елементів у мобільних гоночних іграх. Ці ігри вирізняються винятковою увагою до деталей, реалістичною фізикою та бездоганною графікою. Від тонко деталізованих автомобілів до красивого оточення - ці ігри підняли планку візуальної якості мобільних гоночних ігор і надихнули інших розробників прагнути до подібної візуальної досконалості.

1.2 Ігровий движок Unity

Unity став одним з найпопулярніших і найпоширеніших рушіїв для розробки ігор в індустрії. Він пропонує широкий спектр інструментів і можливостей, які роблять його ідеальним для розробки мобільних гоночних додатків. Цей підрозділ знайомить з ігровим рушієм Unity та висвітлює його ключові особливості та переваги.

Unity відомий своїми крос-платформними можливостями, що дозволяє розробникам створювати ігри, які можна розповсюджувати на різних

платформах, включаючи iOS, Android, Windows та macOS. Ця крос-платформна підтримка позбавляє розробників необхідності писати окремі кодові бази для кожної платформи, що значно скорочує час і зусилля на розробку.

Unity надає багатий і потужний набір функцій, що дозволяє розробникам створювати високо інтерактивні та візуально приголомшливі мобільні гоночні ігри, пропонуючи інструменти для 3D-моделювання, анімації, симуляції фізики, інтеграції звуку, спец ефектів і багато іншого. Крім того, Unity Asset Store надає велику бібліотеку готових до використання ресурсів, плагінів та скриптів для прискорення розробки та покращення функціональності гри.

Unity має процвітаючу спільноту розробників та безліч ресурсів, включаючи вичерпну документацію, навчальні посібники, форуми та онлайн-спільноту. Ця комплексна система підтримки дозволяє розробникам швидко знаходити відповіді на запитання, вивчати нові техніки та вирішувати проблеми. Активна спільнота також заохочує до співпраці та обміну знаннями, що полегшує розробникам Unity розвиток і навчання.

Unity має інтуїтивно зрозумілий, дружній інтерфейс, доступний для розробників усіх рівнів. Візуальний редактор забезпечує візуальний та інтерактивний процес розробки, що дозволяє користувачам створювати сцени, маніпулювати ресурсами та реалізовувати ігрові механіки без необхідності глибоких знань з кодування. Функція перетягування та візуальна система сценаріїв Unity під назвою Playmaker полегшує створення прототипів та ітерації над ігровими функціями.

Однією з сильних сторін Unity є можливості тестування та ітерацій у реальному часі. Розробники можуть вносити зміни до своїх ігор і бачити результати миттєво, без необхідності тривалого процесу збірки. Це значно спрощує ітеративний процес розробки, дозволяючи розробникам швидко змінювати та покращувати ігрову механіку, візуальні ефекти та загальний ігровий досвід.

Unity надає набір функцій та інструментів, спеціально розроблених для задоволення потреб розробки мобільних гоночних додатків.

Крос-платформність Unity дозволяє розробникам створювати мобільні гоночні ігри, які можна розповсюджувати на різних платформах, включаючи iOS та Android. Ця функція усуває необхідність створювати окремі кодові бази для кожної платформи, заощаджуючи час і зусилля розробників. Це дозволяє іграм охопити ширшу аудиторію та максимізувати потенційне охоплення ринку.

Unity надає розширені можливості рендерингу графіки, які сприяють підвищенню візуальної якості мобільних гоночних ігор. Підтримуються такі функції, як динамічне освітлення, тіні в реальному часі, віддзеркалення та ефекти пост обробки, що дозволяють розробникам створювати візуально приголомшливі гоночні середовища. Ці функції підвищують загальну реалістичність, занурення та візуальну привабливість гри, забезпечуючи захоплюючий візуальний досвід для гравця.

Вбудований фізичний рушій Unity забезпечує реалістичну симуляцію фізики у мобільних гоночних іграх. Ця функція важлива для створення точної динаміки транспортних засобів, зіткнень та механіки керування. Розробники можуть використовувати фізичні компоненти Unity, такі як тверді тіла, колайдери та з'єднання, щоб досягти реалістичного та чутливого ігрового процесу. Фізичне моделювання робить перегони більш правдоподібними, захоплюючими та цікавими для гравця.

Unity пропонує надійну систему анімації, яка дозволяє розробникам мобільних гоночних ігор створювати і контролювати динамічну та реалістичну анімацію. Розробники можуть анімувати та оживляти транспортні засоби, персонажів та елементи навколишнього середовища, щоб підвищити візуальну привабливість своїх ігор, а інструменти анімації Unity надають такі можливості, як анімація ключових кадрів, зворотна кінематика, дерева накладання та події анімації, що допомагають розробникам створювати плавні, реалістичні анімації

та дозволяють розробникам створювати плавні, реалістичні рухи.

Конвеєр ресурсів Unity спрощує імпорт, керування та оптимізацію ресурсів для розробки мобільних гоночних ігор. Розробники можуть легко імпортувати ігрові ресурси, такі як 3D-моделі, текстури та аудіофайли, до редактора Unity, а також ефективно редагувати та керувати ними. Конвеєр ресурсів Unity також включає функції оптимізації ресурсів, які дозволяють розробникам зменшити розмір файлів, оптимізувати текстури та завантаження ресурсів, щоб забезпечити оптимальну продуктивність на мобільних пристроях.

Unity підтримує C# як основну мову програмування, надаючи розробникам потужне та гнучке середовище для написання сценаріїв. Це дозволяє розробникам реалізовувати такі функції, як ігрові механіки, системи штучного інтелекту та користувацькі інтерфейси в мобільних гоночних іграх. C# дозволяє розробникам використовувати такі функції, як об'єктно-орієнтоване програмування, багаті бібліотеки та легка інтеграція з Unity API для написання чистого та ефективного коду.

Unity надає функції та методи оптимізації, спеціально розроблені для мобільних платформ. Вона надає інструменти для оптимізації продуктивності гри, зменшення використання пам'яті та збільшення частоти кадрів на мобільних пристроях. Він також дозволяє задавати різні налаштування та конфігурації для кожної мобільної платформи, забезпечуючи сумісність та оптимальну продуктивність на різних пристроях.

Редактор місцевості Unity's Terrain Editor - це потужний інструмент, який дозволяє розробникам створювати та формувати місцевість для гоночного середовища. Елементи рельєфу, такі як пагорби, гори, долини та скелі, можна формувати та малювати. Розробники можуть змінювати висоту рельєфу, застосовувати текстури та додавати листя, щоб створювати реалістичні та візуально приємні ландшафти. Редактор місцевості також підтримує розміщення елементів навколишнього середовища, таких як скелі, дерева і будівлі, що додає

загальну атмосферу гоночного середовища.

Проектування плану траси є важливою частиною створення гоночного середовища, і Unity надає ряд інструментів, які допомагають у розробці плану траси. Розробники можуть використовувати вбудовані інструменти редактора Unity для створення доріг та дорожніх сплайнів, які визначають план треку. Ці інструменти точно розміщують криві, прямі ділянки та перепади висот, дозволяючи розробникам створювати треки, які забезпечують складний та захоплюючий досвід перегонів. Вони також надають можливість додавати елементи, пов'язані з трасою, такі як контрольні пункти та стартові позиції.

Щоб зробити середовище перегонів більш привабливим, в Unity можна інтегрувати об'єкти навколишнього середовища. До таких ресурсів належать 3D-моделі будівель, рослинності, реквізиту та інших елементів середовища, а конвеєр імпорту ресурсів Unity дозволяє легко інтегрувати ці ресурси у гру. Розробники можуть імпортувати ресурси та розміщувати їх у середовищі перегонів, щоб забезпечити належне масштабування, вирівнювання та позиціонування. Розміщення потрібних ресурсів у середовищі може підвищити реалістичність та занурення у перегони.

Візуальні ефекти відіграють важливу роль у покращенні атмосфери та візуальної привабливості гоночного середовища, і Unity пропонує розробникам широкий спектр інструментів та можливостей для створення візуальних ефектів. Наприклад, розробники можуть використовувати систему частинок Unity для створення погодних ефектів, таких як пісок, іскри, дим або дощ і сніг. Ефекти освітлення, такі як динамічні тіні та віддзеркалення, також можуть бути застосовані для додавання глибини та реалістичності оточенню. Ретельно додаючи візуальні ефекти, розробники можуть створювати привабливі та візуально привабливі гоночні середовища.

Для мобільних гоночних ігор важливо створювати візуально насичене середовище, зберігаючи при цьому оптимальну продуктивність, і Unity надає

інструменти та методи для оптимізації продуктивності гри. Розробники можуть оптимізувати продуктивність периферійних ресурсів за допомогою таких методів, як LOD (рівень деталізації) для зменшення складності віддалених об'єктів, застосування оклюзії для запобігання створенню небажаних елементів, а також оптимізація розміру та стиснення текстур. Ці оптимізації забезпечують плавний ігровий процес та ефективне використання системних ресурсів на мобільних пристроях.

Для створення реалістичних моделей транспортних засобів використовуються програмні пакети для 3D моделювання, такі як Blender та Autodesk Maya. Ці програмні пакети надають ряд інструментів та методів моделювання для створення точної геометрії транспортного засобу. Кузови, шасі, колеса та інші компоненти можуть бути змодельовані відповідно до бажаного дизайну та специфікацій. Необхідно дотримуватися правильних пропорцій, масштабу та деталізації, щоб забезпечити достовірне відображення транспортного засобу.

Для оптимальної роботи на мобільних пристроях, модель транспортного засобу повинна бути оптимізована. Це включає зменшення кількості полігонів, видалення непотрібних деталей та оптимізацію текстур. Моделі з великою кількістю полігонів можуть спричинити проблеми з продуктивністю, особливо на мобільних пристроях низького класу. Оптимізуючи моделі, розробники можуть досягти балансу між візуальною якістю та продуктивністю.

Unity підтримує різні формати файлів для імпорту 3D моделей, зокрема FBX, OBJ та Collada . Розробники можуть імпортувати моделі транспортних засобів з програмного забезпечення для моделювання у будь-якому з цих форматів та імпортувати їх до конвеєра ресурсів Unity; опції імпорту Unity дозволяють налаштувати масштаб, точки прив'язки та інші властивості імпортованої моделі. Важливо переконатися, що імпортована модель зберігає свої пропорції та вирівнювання, щоб полегшити точне фізичне моделювання.

Фізичний рушій Unity забезпечує реалістичне моделювання фізики транспортних засобів для гоночних ігор. Розробники можуть використовувати фізичні компоненти Unity, такі як Rigidbody, Colliders та Joints, щоб налаштувати фізичне моделювання транспортних засобів. Компонент Rigidbody моделює фізичні властивості транспортного засобу, такі як маса, тертя та швидкість. Колайдери визначають межі транспортного засобу і дозволяють йому взаємодіяти з навколишнім середовищем та іншими об'єктами. З'єднання дозволяють частинам автомобіля, таким як підвіска та колеса, кріпитися та рухатися.

Unity надає спеціальний компонент під назвою Wheel Collider для імітації поведінки коліс транспортного засобу. Розробники можуть додати Wheel Collider до моделі колеса транспортного засобу і налаштувати такі параметри, як маса, радіус, підвіска, тертя тощо. Wheel Collider взаємодіє з рельєфом місцевості і застосовує сили і крутний момент для імітації руху колеса, стиснення підвіски і шини. Він імітує рух коліс, стиснення підвіски та зчеплення шин з дорогою. Правильна конфігурація Wheel Collider та налаштування підвіски мають важливе значення для реалістичної та чутливої поведінки автомобіля.

Точне налаштування параметрів фізики автомобіля є важливим кроком у створенні приємних вражень від перегонів. Розробники можуть регулювати такі параметри, як вага, опір, жорсткість підвіски, тертя шин, точність рульового управління та гальмівна сила, щоб досягти бажаної поведінки автомобіля. Баланс між реалістичністю та точністю гри вимагає багаторазового тестування та доопрацювання. Розробники можуть точно налаштувати ці параметри на основі відгуків гравців, щоб забезпечити реалістичність руху транспортних засобів та захоплюючий досвід перегонів.

Дизайн користувацького інтерфейсу та інтерактивність відіграють важливу роль у мобільних гоночних іграх, надаючи гравцям інтуїтивно зрозумілі елементи керування, життєво важливу інформацію та захоплюючий ігровий

досвід. Unity надає гнучкий та інтуїтивно зрозумілий інтерфейс для проектування інтерфейсів користувача (UI). Розробники можуть створювати візуально привабливі та функціональні елементи інтерфейсу користувача за допомогою системи інтерфейсу користувача Unity, яка включає такі компоненти, як кнопки, повзунки, текстові елементи та панелі. Елементи інтерфейсу можуть бути розміщені та оформлені відповідно до візуальної теми гри, таким чином надаючи гравцям інтуїтивно зрозумілий макет. Розробники можуть використовувати полотно інтерфейсу та компоненти макета Unity для створення адаптивних елементів інтерфейсу, які підлаштовуються під різні розміри та орієнтацію екрану.

Елементи керування є невід'ємною частиною ігрового процесу в мобільних гоночних іграх, і Unity пропонує цілий ряд методів введення для мобільних пристроїв, включаючи сенсорні кнопки, кнопки нахилу та віртуальні кнопки. Використовуючи систему введення Unity, розробники можуть перехоплювати жести дотику, такі як торкання, перетягування та свайп, і перетворювати їх на конкретні ігрові дії. Вони також можуть використовувати нахил пристрою для керування кермом або використовувати віртуальні кнопки для керування такими діями, як прискорення, гальмування або прискорення. Ці елементи керування мають бути інтуїтивно зрозумілими та чуйними, щоб забезпечити безперебійний ігровий процес.

Інформаційні дисплеї (HUD) є важливим компонентом мобільних гоночних ігор, які надають інформацію в реальному часі під час гри, і Unity дозволяє розробникам створювати елементи HUD, використовуючи компоненти інтерфейсу, такі як текст, зображення та індикатори прогресу. Елементи HUD можуть відображати таку важливу інформацію, як швидкість, час проходження кола, позиція, залишкове прискорення, міні-мапа. Важливо, щоб дизайн HUD був чітким, лаконічним і ненав'язливим, щоб не заважати гравцеві бачити оточення перегонів.

Unity може надавати гравцеві зворотній зв'язок та сповіщення за допомогою візуальних та звукових підказок. Візуальні підказки включають такі індикатори, як прискорення, зіткнення, контрольні точки та завершення кола. Звукові підказки можуть покращити ігровий процес, надаючи зворотній зв'язок, наприклад, звуки двигуна автомобіля, вереск шин та звуки аварії. Ці механізми зворотного зв'язку допомагають гравцеві зануритися в ігровий світ і надають важливу інформацію та попередження для покращення результатів у перегонах.

Система користувацького інтерфейсу Unity дозволяє розробникам створювати екрани меню для запуску, паузи, налаштувань та виходу з гри. Вони можуть створювати візуально привабливі меню з кнопками, повзунками та опціями для керування налаштуваннями гри, параметрами звуку та іншими функціями кастомізації. Крім того, функції управління сценами Unity забезпечують плавні переходи між різними ігровими режимами, рівнями та меню. Добре продумана система меню та потік проходження гри сприяють плавному та приємному ігровому процесу.

Unity дозволяє реалізовувати внутрішньо ігрові події та винагороди, щоб підвищити взаємодію та залученість гравців. Ви можете створювати такі події, як випробування на час, турніри та спеціальні завдання, щоб урізноманітнити гру та зробити її більш захоплюючою. Нагороди, такі як нові транспортні засоби, кастомізації та розблокування треків, заохочують гравців до прогресу та досягнення певних цілей. Скриптові можливості Unity дозволяють розробникам впроваджувати тригери подій, відстежувати прогрес та системи нагород для створення цікавого та динамічного ігрового процесу.

Дизайн користувацького інтерфейсу та інтерактивність відіграють важливу роль у мобільних гоночних іграх, надаючи гравцям інтуїтивно зрозумілі елементи керування, життєво важливу інформацію та захоплюючий ігровий досвід. Unity надає гнучкий та інтуїтивно зрозумілий інтерфейс для проектування інтерфейсів користувача (UI). Розробники можуть створювати

візуально привабливі та функціональні елементи інтерфейсу користувача за допомогою системи інтерфейсу користувача Unity, яка включає такі компоненти, як кнопки, повзунки, текстові елементи та панелі. Елементи інтерфейсу можуть бути розміщені та оформлені відповідно до візуальної теми гри, таким чином надаючи гравцям інтуїтивно зрозумілий макет. Розробники можуть використовувати полотно інтерфейсу та компоненти макета Unity для створення адаптивних елементів інтерфейсу, які підлаштовуються під різні розміри та орієнтацію екрану.

Елементи керування є невід'ємною частиною ігрового процесу в мобільних гоночних іграх, і Unity пропонує цілий ряд методів введення для мобільних пристроїв, включаючи сенсорні кнопки, кнопки нахилу та віртуальні кнопки. Використовуючи систему введення Unity, розробники можуть перехоплювати жести дотику, такі як торкання, перетягування та свайп, і перетворювати їх на конкретні ігрові дії. Вони також можуть використовувати нахил пристрою для керування кермом або використовувати віртуальні кнопки для керування такими діями, як прискорення, гальмування або прискорення. Ці елементи керування мають бути інтуїтивно зрозумілими та чуйними, щоб забезпечити безперебійний ігровий процес.

Інформаційні дисплеї (HUD) є важливим компонентом мобільних гоночних ігор, які надають інформацію в реальному часі під час гри, і Unity дозволяє розробникам створювати елементи HUD, використовуючи компоненти інтерфейсу, такі як текст, зображення та індикатори прогресу. Елементи HUD можуть відображати таку важливу інформацію, як швидкість, час проходження кола, позиція, залишкове прискорення, міні-мапа. Важливо, щоб дизайн HUD був чітким, лаконічним і ненав'язливим, щоб не заважати гравцеві бачити оточення перегонів.

Unity може надавати гравцеві зворотній зв'язок та сповіщення за допомогою візуальних та звукових підказок. Візуальні підказки включають такі індикатори,

як прискорення, зіткнення, контрольні точки та завершення кола. Звукові підказки можуть покращити ігровий процес, надаючи зворотній зв'язок, наприклад, звуки двигуна автомобіля, вереск шин та звуки аварії. Ці механізми зворотного зв'язку допомагають гравцеві зануритися в ігровий світ і надають важливу інформацію та попередження для покращення результатів у перегонах.

Система користувацького інтерфейсу Unity дозволяє розробникам створювати екрани меню для запуску, паузи, налаштувань та виходу з гри. Вони можуть створювати візуально привабливі меню з кнопками, повзунками та опціями для керування налаштуваннями гри, параметрами звуку та іншими функціями кастомізації. Крім того, функції управління сценами Unity забезпечують плавні переходи між різними ігровими режимами, рівнями та меню. Добре продумана система меню та потік проходження гри сприяють плавному та приємному ігровому процесу.

Unity дозволяє реалізовувати внутрішньо ігрові події та винагороди, щоб підвищити взаємодію та залученість гравців. Ви можете створювати такі події, як випробування на час, турніри та спеціальні завдання, щоб урізноманітнити гру та зробити її більш захоплюючою. Нагороди, такі як нові транспортні засоби, кастомізації та розблокування треків, заохочують гравців до прогресу та досягнення певних цілей. Скриптові можливості Unity дозволяють розробникам впроваджувати тригери подій, відстежувати прогрес та системи нагород для створення цікавого та динамічного ігрового процесу.

Система користувацького інтерфейсу Unity, елементи керування, HUD, механізми зворотного зв'язку, система меню, ігрові події та нагороди дозволяють розробникам створювати захопливі та інтерактивні користувацькі інтерфейси в мобільних гоночних іграх. Добре продуманий користувацький інтерфейс та інтуїтивно зрозуміла взаємодія покращує ігровий процес і робить гру доступною та цікавою для гравців.

Дизайн користувацького інтерфейсу та інтерактивність відіграють важливу

роль у мобільних гоночних іграх, надаючи гравцям інтуїтивно зрозумілі елементи керування, життєво важливу інформацію та захоплюючий ігровий досвід. Unity надає гнучкий та інтуїтивно зрозумілий інтерфейс для проектування інтерфейсів користувача (UI). Розробники можуть створювати візуально привабливі та функціональні елементи інтерфейсу користувача за допомогою системи інтерфейсу користувача Unity, яка включає такі компоненти, як кнопки, повзунки, текстові елементи та панелі. Елементи інтерфейсу можуть бути розміщені та оформлені відповідно до візуальної теми гри, таким чином надаючи гравцям інтуїтивно зрозумілий макет. Розробники можуть використовувати полотно інтерфейсу та компоненти макета Unity для створення адаптивних елементів інтерфейсу, які підлаштовуються під різні розміри та орієнтацію екрану.

Елементи керування є невід'ємною частиною ігрового процесу в мобільних гоночних іграх, і Unity пропонує цілий ряд методів введення для мобільних пристроїв, включаючи сенсорні кнопки, кнопки нахилу та віртуальні кнопки. Використовуючи систему введення Unity, розробники можуть перехоплювати жести дотику, такі як торкання, перетягування та свайп, і перетворювати їх на конкретні ігрові дії. Вони також можуть використовувати нахил пристрою для керування кермом або використовувати віртуальні кнопки для керування такими діями, як прискорення, гальмування або прискорення. Ці елементи керування мають бути інтуїтивно зрозумілими та чуйними, щоб забезпечити безперебійний ігровий процес.

Інформаційні дисплеї (HUD) є важливим компонентом мобільних гоночних ігор, які надають інформацію в реальному часі під час гри, і Unity дозволяє розробникам створювати елементи HUD, використовуючи компоненти інтерфейсу, такі як текст, зображення та індикатори прогресу. Елементи HUD можуть відображати таку важливу інформацію, як швидкість, час проходження кола, позиція, залишкове прискорення, міні-мапа. Важливо, щоб дизайн HUD був

чітким, лаконічним і ненав'язливим, щоб не заважати гравцеві бачити оточення перегонів.

Unity може надавати гравцеві зворотній зв'язок та сповіщення за допомогою візуальних та звукових підказок. Візуальні підказки включають такі індикатори, як прискорення, зіткнення, контрольні точки та завершення кола. Звукові підказки можуть покращити ігровий процес, надаючи зворотній зв'язок, наприклад, звуки двигуна автомобіля, вереск шин та звуки аварії. Ці механізми зворотного зв'язку допомагають гравцеві зануритися в ігровий світ і надають важливу інформацію та попередження для покращення результатів у перегонах.

Система користувацького інтерфейсу Unity дозволяє розробникам створювати екрани меню для запуску, паузи, налаштувань та виходу з гри. Вони можуть створювати візуально привабливі меню з кнопками, повзунками та опціями для керування налаштуваннями гри, параметрами звуку та іншими функціями кастомізації. Крім того, функції управління сценами Unity забезпечують плавні переходи між різними ігровими режимами, рівнями та меню. Добре продумана система меню та потік проходження гри сприяють плавному та приємному ігровому процесу.

1.3 Blender для 3D-моделювання

Blender - це потужне та універсальне програмне забезпечення для 3D-моделювання з відкритим вихідним кодом, яке широко використовується в індустрії розробки ігор, пропонуючи широкий спектр інструментів та можливостей для створення 3D-моделей, анімації та візуальних ефектів. Цей розділ знайомить з Blender'ом, висвітлює його інтерфейс, ключові можливості та придатність для створення ресурсів для мобільних гоночних ігор.

Після запуску Blender'a користувачеві відкривається унікальний інтерфейс,

який можна налаштовувати. Цей інтерфейс складається з різних панелей, панелей інструментів та вікон, які надають доступ до різних функцій та опцій редагування. Основними компонентами інтерфейсу Blender'a є

Область перегляду: центральна область, де користувачі можуть переглядати та маніпулювати 3D моделями, сценами та анімаціями.

Панелі інструментів та панелі: розташовані по обидва боки від області перегляду і надають доступ до інструментів моделювання, властивостей, налаштувань та опцій.

Редактор властивостей: панель, на якій відображаються певні властивості та налаштування, пов'язані з вибраним об'єктом або елементом.

Часова шкала: дозволяє керувати і контролювати анімацію та ключові кадри.

Структура: відображає огляд усіх об'єктів, матеріалів та компонентів у сцені.

Інтерфейс Blender'a добре налаштовується, дозволяючи користувачам розташовувати та організовувати панелі відповідно до власних вподобань та робочих процесів. Він також надає повний набір комбінацій клавіш і гарячих клавіш, щоб полегшити навігацію та взаємодію з програмою.

Blender має широкий спектр можливостей, які роблять його ідеальним для 3D-моделювання та анімації.

Blender надає потужний набір інструментів моделювання для створення та редагування 3D моделей. Ці інструменти включають екструзію, масштабування, зняття фасок, розділення поверхонь тощо, а також забезпечують точний контроль над геометрією та маніпуляціями з формою.

Blender пропонує просунуті інструменти UV-мапінгу, які дозволяють відкривати 3D-моделі та створювати UV-карти для мапування текстур. Підтримується цілий ряд технік малювання текстур і варіантів створення матеріалів, що дозволяє користувачам додавати колір, текстуру і деталізацію поверхонь до своїх моделей.

Функції оснащення та анімації Blender'a дозволяють користувачам

створювати скелетні структури, визначати рухи суглобів та анімувати 3D-моделі. Потужна система арматури, зважені інструменти малювання та ключові можливості кадрування дозволяють користувачам створювати реалістичні рухи та анімацію.

Візуальні ефекти Blender включає широкий спектр інструментів для створення візуальних ефектів, зокрема систему частинок, динамічне моделювання (включаючи тканину, рідину та фізику) і потужну систему композиції на основі вузлів. Ці можливості дозволяють користувачам створювати реалістичні симуляції ігрових ресурсів та привабливі візуальні ефекти.

Blender підтримує написання сценаріїв на мові Python, яка надає широкий спектр можливостей для кастомізації та автоматизації. Користувачі можуть писати скрипти для автоматизації повторюваних завдань, створювати власні інструменти та розширювати можливості Blender'a відповідно до конкретних вимог.

Універсальність та широкий спектр можливостей Blender'a роблять його цінним інструментом для створення ресурсів для мобільних гоночних ігор. Конкретні причини, чому Blender є актуальним для розробки мобільних гоночних ігор, включають

3D-моделювання: потужні інструменти моделювання Blender'a дозволяють розробникам створювати детальні та точні 3D-моделі транспортних засобів, треків, оточення та інших ігрових елементів, необхідних для мобільних гоночних ігор. Гнучкість Blender'a дозволяє створювати різноманітні дизайни транспортних засобів та варіанти кастомізації. UV-візуалізація та текстурування: Можливості UV-рендерингу Blender'a дозволяють розробникам відкривати та текстурувати 3D-моделі, забезпечуючи високоякісні текстури та реалістичну деталізацію поверхонь транспортних засобів та оточення. Це підвищує візуальну привабливість та реалістичність гри.

Можливості оснащення та анімації Blender'a дозволяють розробникам створювати реалістичні рухи транспортних засобів та анімації, такі як динаміка підвіски, обертання коліс та керування кермом. Це надає мобільним перегонам реалістичності та захопливості.

Інструменти візуальних ефектів Blender'a дозволяють створювати динамічні та візуально привабливі ефекти, такі як дим, полум'я, прискорення, дрейф та ефекти на основі частинок для зіткнень. Ці ефекти можуть посилити загальне захоплення та візуальний вплив гоночної гри.

Blender підтримує різноманітні формати файлів, які зазвичай використовуються у розробці ігор, зокрема FBX та OBJ, які сумісні з Unity та іншими ігровими рушіями. Це дозволяє легко інтегрувати ресурси Blender'a у конвеєр розробки гри.

Інтерфейс та навігація Blender'a є важливими елементами для ефективної роботи з програмою. У цьому розділі детально описано компоненти інтерфейсу та елементів керування навігацією у Blender'і. Інтерфейс дозволяє вам переміщатися у тривимірному робочому просторі, отримувати доступ до інструментів та панелей, а також безперешкодно взаємодіяти з вашими моделями.

Компоненти інтерфейсу

Інтерфейс Blender'a складається з різних компонентів, які надають користувачам доступ до різних функцій та опцій редагування. Розуміння цих компонентів має вирішальне значення для ефективної навігації та робочого процесу. Основними компонентами інтерфейсу є

Область перегляду -центральна область вікна Blender'a, де користувачі можуть переглядати та керувати своїми 3D-моделями. Доступні різні режими перегляду, такі як каркасний, суцільний і візуалізація, що дозволяють користувачеві візуалізувати модель з різних точок зору.

Панелі інструментів та панелі навколо області перегляду містять широкий

спектр інструментів, опцій та налаштувань, пов'язаних з різними аспектами моделювання, анімації та візуалізації. Користувачі можуть налаштовувати розташування та видимість цих панелей відповідно до своїх уподобань у робочому процесі.

Редактор властивостей - це універсальна панель, яка відображає контекстно-залежні налаштування та властивості на основі вибраного об'єкта або режиму. Вона надає доступ до низки опцій для зміни атрибутів об'єктів, матеріалів, текстур тощо.

Панель Outliner надає огляд усіх об'єктів, колекцій та інших компонентів у сцені. Вона дозволяє ефективно організовувати, вибирати та керувати об'єктами, а також полегшує роботу зі складними сценами.

Ефективна навігація у тривимірному робочому просторі має важливе значення для точного моделювання та керування сценою, і Blender має різноманітні елементи керування навігацією.

Користувач може обертатися навколо сцени, утримуючи центральну кнопку миші та переміщуючи мишу. Це дозволяє обертати модель на 360° і розглядати її під різними кутами.

Панорамування дозволяє користувачеві переміщати вид по горизонталі або вертикалі, утримуючи клавішу Shift і перетягуючи середню кнопку миші.

Масштабування дозволяє користувачеві збільшувати або зменшувати масштаб моделі. Це можна зробити, прокручуючи коліщатко миші або використовуючи клавіші + і -. Масштабування також можна виконати, утримуючи клавішу Ctrl і перетягуючи середню кнопку миші.

Blender надає візуальні пристрої, такі як пристрої навігації по видовому екрану та віджети масштабування, які надають альтернативні способи навігації по 3D-візуальному екрану. Ці пристрої надають інтерактивні елементи керування, які дозволяють користувачеві клацати та перетягувати для переміщення, масштабування та обертання видошукача.

Blender надає набір навігаційних ярликів для швидкого перемикання між різними видами або для доступу до певних точок зору чи кутів камери.

Тривимірне моделювання є важливою частиною створення ресурсів для мобільних гоночних ігор, і Blender пропонує потужний набір інструментів та технік для створення детальних та реалістичних тривимірних моделей. У цьому розділі описано основні техніки 3D-моделювання у Blender'і, які дозволяють користувачам точно ліпити, маніпулювати та покращувати свої моделі.

Полігональне моделювання є широко вживаною технікою тривимірного моделювання, і Blender надає потужні інструменти для роботи з полігонами. У цій техніці моделі створюються та редагуються шляхом маніпулювання окремими полігонами та вершинами; Blender пропонує декілька інструментів для редагування полігонів, зокрема

Витискання дублює та розтягує вибрані грані, ребра або кути вздовж заданої осі. Це корисно для створення проекцій, додавання деталей і створення складних фігур.

Масштабування дозволяє змінювати розмір вибраних компонентів рівномірно або вздовж певних осей. Це корисно для коригування пропорцій моделі або створення варіацій розмірів.

Зняття фасок - це процес округлення або скошування країв виділених полігонів. Це додає моделі плавності та реалістичності, уникаючи гострих країв і створюючи реалістичні переходи.

Модифікатор Поверхні Розбиття у Blender'і може розбивати існуючі полігони на частини для створення більш гладких поверхонь. Це покращує геометрію та загальну якість моделі.

Blender має широкі можливості редагування сітей, які дозволяють користувачам змінювати та покращувати структуру своїх 3D-моделей. Ці інструменти включають

Редагування ребер та кутів: У Blender можна точно маніпулювати ребрами та

кутами, дозволяючи користувачам змінювати форму, вирівнювати або маніпулювати окремими частинами моделі. Окремі ребра і кути можна вибирати, переміщати, обертати і масштабувати для детальних налаштувань.

Вирізання петель створює нові ребра на вибраних полігонах і забезпечує більш точний контроль над петлями країв. Інструмент "Вирізання та прокрутка" дозволяє інтерактивно прокручувати новостворену кромку вздовж існуючої топології, а також налаштовувати обтікання і щільність контуру кромки.

Мостовий контур створює і з'єднує міст між двома або більше контурами. Він корисний для з'єднання окремих частин моделі або заповнення проміжків.

Інструмент "Ніж" дозволяє вручну вирізати багатокутники, малюючи лінії на поверхні. Це дає вам можливість створювати складні розрізи і відокремлювати частини моделі.

Система модифікаторів Blender'а забезпечує неруйнівний спосіб модифікації моделей шляхом застосування різноманітних маніпуляцій та ефектів; деякі з найпоширеніших модифікаторів, що використовуються у тривимірному моделюванні Blender'ом, є такими

Модифікатор Дзеркало (Mirror) дозволяє створювати симетричні моделі шляхом дзеркального відображення геометрії по визначених осях. Це спрощує процес моделювання та забезпечує узгодженість.

Модифікатор Subdivision Surface (Поверхня Розбиття) розбиває сіть на частини для створення більш гладких поверхонь. Це особливо корисно для органічних форм і високоякісних поверхонь.

Модифікатори Solidify (Затвердіння) видавлюють грані та краї моделі і додають їй товщини. Це корисно для створення тривимірних об'єктів або додавання товщини тонким поверхням.

Масив: повторює та впорядковує вибрані компоненти (наприклад, групи об'єктів або вершини) відповідно до заданого шаблону. Корисний для створення повторюваних патернів, наприклад, рівномірно розташованих об'єктів.

Blender має потужний інструмент скульптування, який дозволяє користувачам додавати дрібні деталі та органічні форми до своїх моделей. Скульптування пропонує більш художній та інтуїтивний підхід до моделювання, дозволяючи користувачам формувати та покращувати поверхню своїх моделей. або вилучення деталей та різноманітні пензлі для скульптування. Ретопологія.

Ретопологія - це процес створення нової, оптимізованої топології на основі існуючої моделі. У Blender'i є інструменти ретопології, такі як Remesh та модифікатор Shrinkwrap, які дозволяють створювати чисті, оптимізовані топології, придатні для анімації та розробки ігор. Можна створювати чисті та оптимізовані топології, придатні для анімації та розробки ігор. Ретопологія необхідна для зменшення кількості полігонів, підвищення продуктивності та забезпечення правильної деформації під час анімації.

UV-відображення - це процес вирівнювання 3D-моделі до 2D-площини і створення UV-карти, яка визначає, як текстури наносяться на поверхні моделі. Blender пропонує різноманітні методи та інструменти UV-відображення, що дозволяють досягти точного і акуратного розгортання:

Smart UV Project (Розумний UV-проект): Smart UV Project (Розумний UV-проект): Інструмент Smart UV Project у Blender'i автоматично генерує УФ-координати для вибраних поверхонь. Це забезпечує швидкий та ефективний спосіб виконання базового мапування ультрафіолетових променів.

Користувачі можуть вручну розгортати 3D-модель, створюючи шви по краях, де вони хочуть розділити UV-карту; інструмент Blender's Unwrap (Розгортання) дозволяє користувачам позначати шви і розгортати модель на основі цих швів, надаючи їм повний контроль над UV-макетом, надаючи користувачеві повний контроль над макетом.

У Blender'i є спеціальна робоча область UV-редагування з інструментами та панелями, призначеними для маніпулювання та редагування UV-мап; UV-острівці можна масштабувати, обертати та переміщувати, а відстань між

острівцями можна регулювати для збільшення роздільної здатності текстури.

Після завершення UV-маніпулювання, користувач може перейти до застосування текстур до поверхонь моделі, на які нанесено UV-мапу. Blender пропонує наступні варіанти текстур

Користувачі можуть імпортувати файли зображень або створювати текстури у режимі малювання текстур у Blender'i. Ці текстури зображень можуть бути призначені до певних острівців UV-світла або застосовані безпосередньо до поверхонь моделі.

У Blender'i є декілька генераторів процедурних текстур, таких як генератор шуму, шахової дошки та градієнта, які дозволяють користувачам створювати текстури програмно. Процедурні текстури можна змішувати та маніпулювати ними для досягнення різних ефектів та патернів.

Вузловий редактор матеріалів у Blender'i дозволяє створювати складні та деталізовані матеріали, комбінуючи різні вузли текстур та налаштовуючи їхні властивості. Це дозволяє створювати візуально багаті та реалістичні матеріали для ігрових ресурсів.

Висновок до пункту 1.

Цей розділ закладає основу для розуміння важливості створення візуальних ресурсів у мобільних гоночних додатках шляхом відстеження тенденцій в індустрії мобільних ігор, розуміння особливостей гоночних ігор, важливості візуальних ефектів у цьому жанрі та еволюції мобільних гоночних ігор.

РОЗДІЛ 2
ЗОВНІШНЕ ПРОЕКТУВАННЯ

Функціональне призначення – налагодження взаємодії між користувачем та інтерфейсом гри, створення зворотного зв’язку між грою та користувачем.

Експлуатаційне призначення- підвищення швидкості роботи користувача та спрощення взаємодії з програмою гри.

Функціональні вимоги:

- Відстеження натискань користувача на екран
- Навігація у всьому меню ігри
- Зміна кольору машин
- Передання команд від кнопок до коду гри
- Показ реклами за допомогою Add Mob
- Реалізація нагород за перегляд реклами
- Налаштування якості графіки
- Створення 3D моделей
- Створення ландшафту карт
- Створення інтерфейсу користувача
- Розробка технічних характеристик машин.

Таблиця вихідних даних.

Найменування	Суть	Формат	Область допустимих значень
Tap	Натискання на екран	Змінна типу bool	bool
Axelerometr	Значення кута повороту екрана	Вектор напрямку руху	Vector3

Через велику кількість однакових по типу об’єктів із різними назвами було обране рішення занести до таблиці тільки основні типи об’єктів не вдаючись у подробиці кожного з них.

Таблиця вхідних даних.

Найменування	Суть	Формат	Область допустимих значень
Button	Кнопка	Об’єкт Button	Tap, Rect Transform, Canvas Render, Script
Canvas	Абстрактний простір, в якому проводиться налаштування та малювання UI	Об’єкт Canvas	Rect Transform, Canvas, Script
Image	Зображення	Об’єкт Image	Tap, Source image, Rect Transform, Canvas Render, Script
Plane	Панель для фону	Об’єкт Plane	Rect Transform, Source image, Material
3D Model	3D модель	3D об’єкт	Transform, Mesh Render, Sprite Render,

			Script
Lights	Випромінювач світла	Пустий об’єкт	Transform, Lens Flare

Опис зовнішнього інформаційного середовища. Данні типу Тар вводятся за допомогою дотиків до екрану мобільного пристрою. Данні типу Axelerometr вводятся через нахил екрану мобільного пристрою. Навігація по різним меню гри, взаємодія з гаражем , закриття перегляду реклами, взаємодія з магазином реалізована через застосування кнопок та зображень. Які у свою чергу розташовані на об`єкті типу Canvas. Управління машиною реалізоване за допомогою Axelerometr, об’єктів типу Button та Image. Для коректної роботи програми на мобільному пристрої необхідна наявність операційної системи Android версії Android 6 або вище.

Діаграму прецедентів(Рисунок 2.1 Діаграма прецедентів) побудовано на основі можливостей користувача у застосуванні UI.

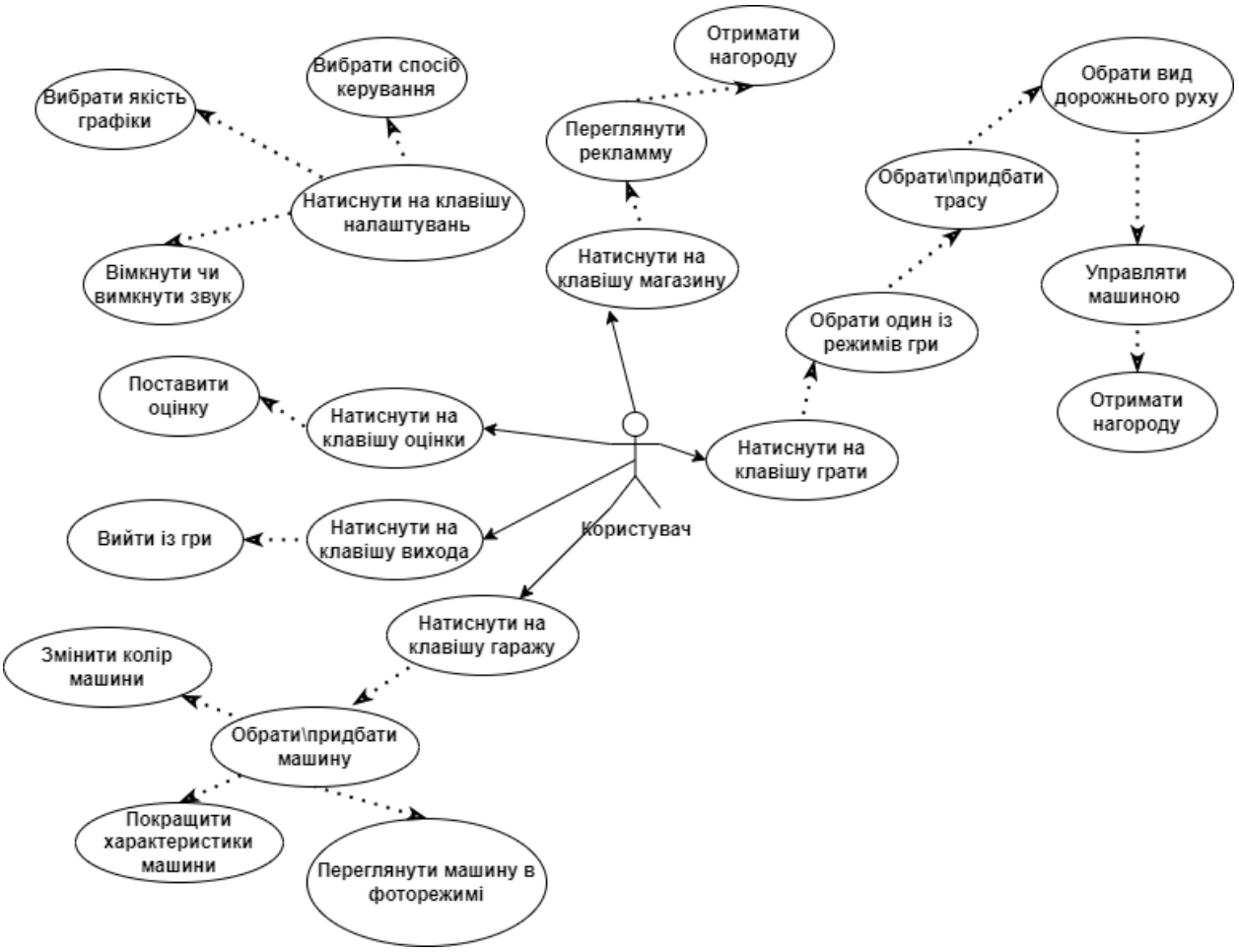


Рисунок 2.1 Діаграма прецедентів

Висновок до пункту 2.

В цьому розділі було зібрані та визначені вимоги та призначення розроблюваного продукту. Для наглядної демонстрації була створена діаграма прецедентів.

РОЗДІЛ 3
ВНУТРІШНЕ ПРОЕКТУВАННЯ

Для моделювання словника системи треба виконати аналіз зовнішніх специфікацій системи. Були ідентифіковані такі об’єкти: Зображення, Клавіша, Полотно та Панель, 3D модель, Світло.

Встановлені обов’язки об’єктів:

- Зображення – відображення зображення, надання можливості бути ціллю для променів.
- Клавіша – при натисканні на клавішу відображення ефекту затемнення, виклик інших команд та виклик програвання звукового супроводу натиску на клавішу.
- Панель – відображення фону.
- Полотно – надання можливості роботи всіх елементів UI, змінення масштабу інтерфейсу користувача згідно різних масштабів екрану пристроїв, створення променів навігації при натисканні на екран.
- 3D модель – відображення 3D моделей із можливістю прикріплення додаткових сприптів до них.
- Світло – створення освітлення.

Таблиця об’єктів

Сутність	Атрибути	Методи
Зображення	Tap – можливість прийняття навігаційних променів; Source image – джерело зображення; Rect Transform –	Відображення зображення та тексту.

	місцезнаходження на сцені; Canvas Render – елемент для зв'язування з Полотном; Script – можливість прикріплювати скріпти.	
Клавіша	Тар можливість прийняття навігаційних променів; Rect Transform– місцезнаходження на сцені; Canvas Render– елемент для зв'язування з Полотном; Script– можливість прикріплювати скріпти.	Відображення ефекту натискання, програвання звуку натискання, виклик подій в інших скриптах
Панель	Rect Transform – місцезнаходження на сцені; Source image – джерело зображення; Material -– джерело	Відображення фонового зображення

	матеріалу.	
Полотно	Rect Transform – місцезнаходження на сцені; Canvas – надання статусу полотна Script- можливість прикріплювати скріпти	Масштабування, відстеження дотиків
3D модель	Transform – місцезнаходження на сцені; Mesh Render – зчитування полотна 3D моделі; Sprite Render – зчитування джерела 3D моделі Script	Відображення 3D моделі. Взаємодія з приєднаними скриптами
Світло	Transform– місцезнаходження на сцені; Lens Flare- генерація світла.	Створення освітлення

Множини об’єктів працюючих разом для досягнення необхідного результату:

- Зображення, Клавіша- зв’язування клавіші із зображенням для їх зручного використання у побудові UI.
- Полотно, Зображення- відображення зображення на UI.
- Полотно, Клавіша - відображення клавіші на UI.

- Полотно, Панель- відображення панелі на UI.
- Світло, 3D Модель – прикріплення освічення до 3D моделі для зручності побудови дизайну гри

Набір обов’язків для об’єктів виходять із методів та атрибутів об’єктів.

Моделювання непрограмних сутностей для даної системи

виконувати не потрібно, оскільки особливості апаратної частини не

використовується в роботі системи, всі інші сутності вже представлено у вигляді класів.

Наслідування в цьому проекті присутнє із класом який дозволяє інтегрувати код C# в двигун Unity, усі класи наслідують цей скріпти об’єктів.

Таблиця зв’язків.

Об’єкт який зв’язується	Об’єкт з яким зв’язуються	Тип зв’язку
Панель	Полотно	Залежність
Клавіша	Полотно	Залежність
Зображення	Полотно	Залежність
Світло	3D Модель	Асоціація
Зображення	Клавіша	Асоціація

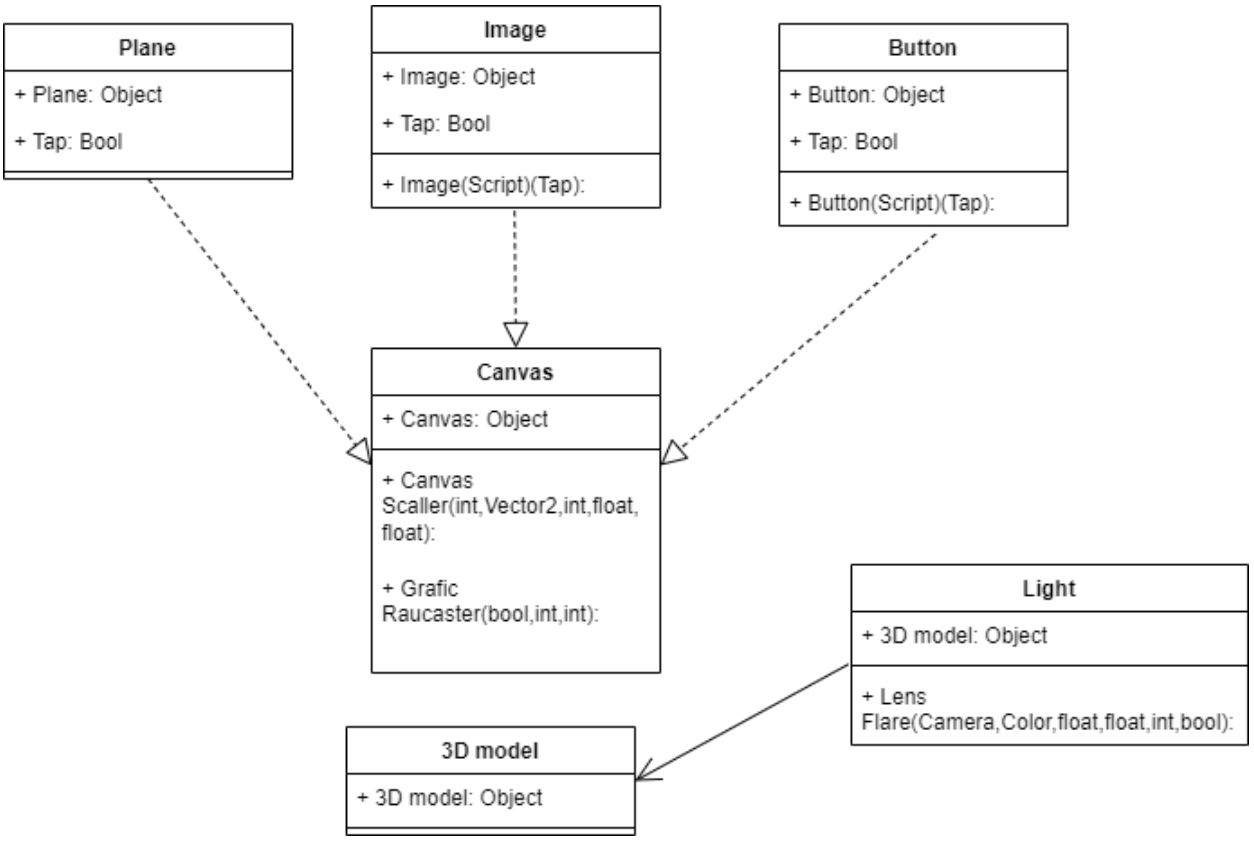


Рисунок 3.1 Таблиця об’єктів

Висновок до пункту 3.

В цьому розділі були визначені та спроектовані основні об’єкти розроблюваного продукту. Для наглядної демонстрації взаємодії об’єктів була створена діаграма прецедентів.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМИ
4.1 Розробка 3D моделей у Blender

Створення 3D моделей було вирішено почати із моделей авто. Бо саме транспортний засіб являє собою головний об’єкт для гоночних ігор. Так як машина це складна структура яка включає в себе багато компонентів, було вирішено розділити створення моделей авто на багато компонентів. Це рішення допоможе у майбутньому створювати анімації руху для кожної частини автомобіля окремо. Що у результаті приведе до більшої реалістичності.

Рисунок 4.1.1 Дзеркало переднього виду

Рисунок 4.1.2 Лобове дзеркало

Дзеркало переднього виду(Рисунок 4.1.1 Дзеркало переднього виду) та лобове дзеркало(Рисунок 4.1.2 Лобове дзеркало) було виконано суцільно одним матеріалом для всієї моделі. До матеріалу дзеркал було додано ефект блиску на сонці для надання об’єму та реалістичності.

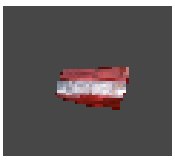


Рисунок 4.1.3 Фари заднього виду

Фари заднього виду(Рисунок 4.1.3 Фари заднього виду) були зроблені окремо через те що після інтеграції до Unity до кожної фари потрібно буде додати елемент освічення. Для фар переднього виду світло буде приєднане до дзеркал

переднього виду.

Рисунок 4.1.4 Номерний знак

Для номерного знаку(Рисунок 4.1.4 Номерний знак) було використано декілька матеріалів. А саме матеріал блискучого металу для того щоб вибити напис у низу таблички. Синій та білий матеріали це матеріал використаний для створення фар із зміненим кольором.

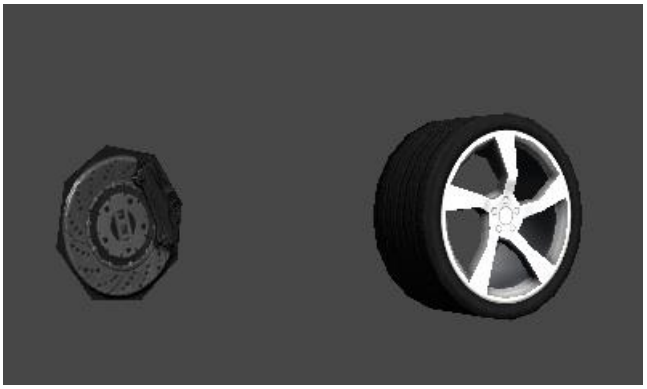


Рисунок 4.1.5 Диски та колесо

У дисках та колесах(Рисунок 4.1.5 Диски та колесо) було використано принцип скульптування для створення маленьких деталей завдяки яким колеса будуть виглядати більш натуральними а не схожими на цільну модель. Також був використаний модифікатор дзеркала для віддзеркалення однієї частини колеса та придання колесу точних ліній. На модель шини було нанесено зображення ліній проти ковзання. Через те що колесо та диски були зроблені окремо у розробника виникає можливість зробити для них різну анімацію переміщення. Щоб колесо та диски оберталися у різному темпі.

Обшивка корпусу машини(Рисунок 4.1.6 Обшивка машини) була зроблена з урахуванням окремих деталей такі як двері капот та бампер. Вона була виконана із одного матеріалу використовуючи ефект металу для створення блиску при

освіченні.

Рисунок 4.1.6 Обшивка машини

Рисунок 4.1.7 Каркас машини

Каркас машини(Рисунок 4.1.7 Каркас машини) було створено розраховуючи на приєднання до нього усіх інших частин авто. Хоч він і виглядає як набір різних деталей проте каркас представляє собою цільну модель.

Далі я хочу продемонструвати об'єднану модель машини.(Рисунок 4.1.8 Об’єднана модель машини)

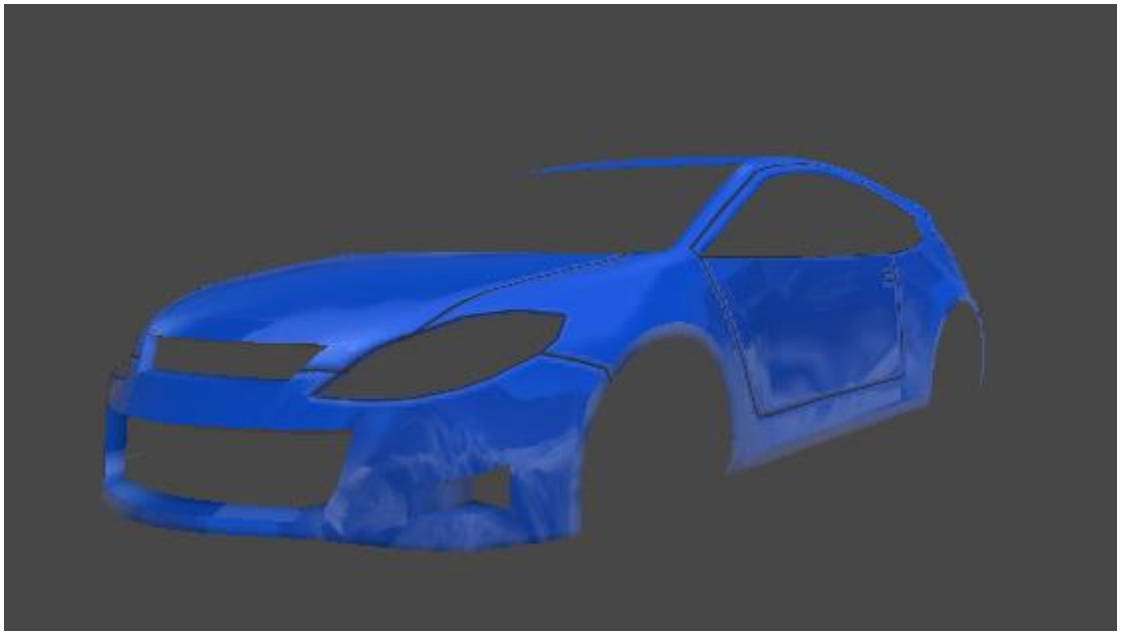


Рисунок 4.1.8 Об’єднана модель машини

Щоб не описувати кожну модель окрема надалі буду демонструвати збірні моделі машин.

Друга модель машини(Рисунок 4.1.9 Передній план модель 2)(Рисунок 4.1.10 Задній план модель 2) побудована на основі Toyota Supra.Третя модель(Рисунок 4.1.11 Передній план модель 3)(Рисунок 4.1.12 Задній план модель 3) була зроблена на основі машини Dodge Viper. Для четвертої моделі(Рисунок 4.1.13 Передній план модель 4)(Рисунок 4.1.14 Задній план модель 4) був обраний Cadillac Escalade. П’ята модель(Рисунок 4.1.15 Передній план модель 5)(Рисунок 4.1.16 Задній план модель 5) створена на основі Range Rover Evoque.

Рисунок 4.1.9 Передній план модель 2

Рисунок 4.1.10 Задній план модель 2

Рисунок 4.1.11 Передній план модель 3

Рисунок 4.1.12 Задній план модель 3

Рисунок 4.1.13 Передній план модель 4

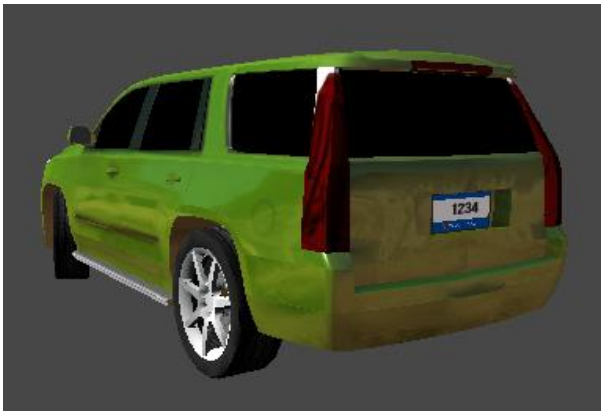


Рисунок 4.1.14 Задній план модель 4

Рисунок 4.1.15 Передній план модель 5

Рисунок 4.1.16 Задній план модель 5

Всі моделі транспорту були представлені тож надалі продемонструю створені моделі для гаражу. Це місце в якому буде проводитися вибір, придбання, та покращення авто.

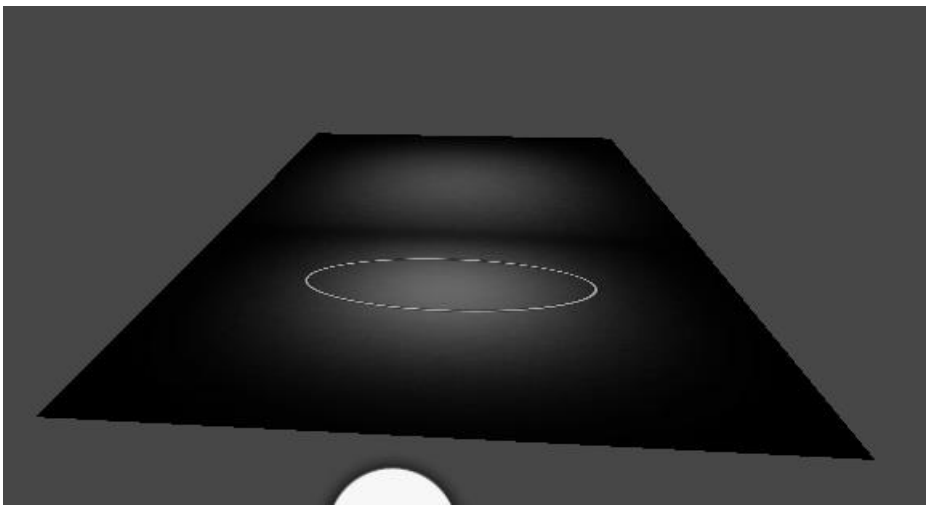


Рисунок 4.1.17 Модель підлоги гаражу

Першою показаною моделлю виступає підлога гаража(Рисунок 4.1.17 Модель підлоги гаражу). Поодаль від центру було намальовано овал. Він буде підкреслювати моделі авто, котрі на ньому розташовані. Це було зроблено щоб моделі авто не зливалися з фоном підлоги. Ефект освічених ділянок на підлозі було зроблено за допомогою освітлення кольору підлоги. Це зробить ефект

світло більш привабливим для користувачів.

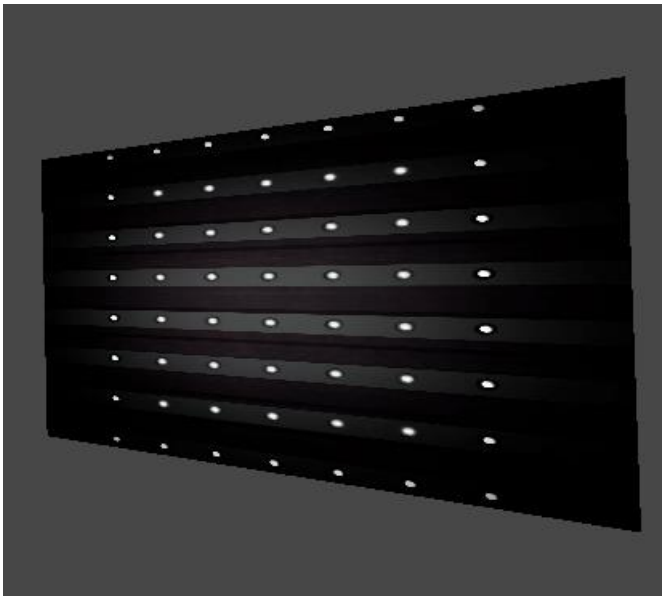


Рисунок 4.1.18 Модель стелі гаражу

Стеля(Рисунок 4.1.18 Модель стелі гаражу) була зроблена у тому ж темному стилі що й підлога. Також до стелі додане імітування ламп освітлення.

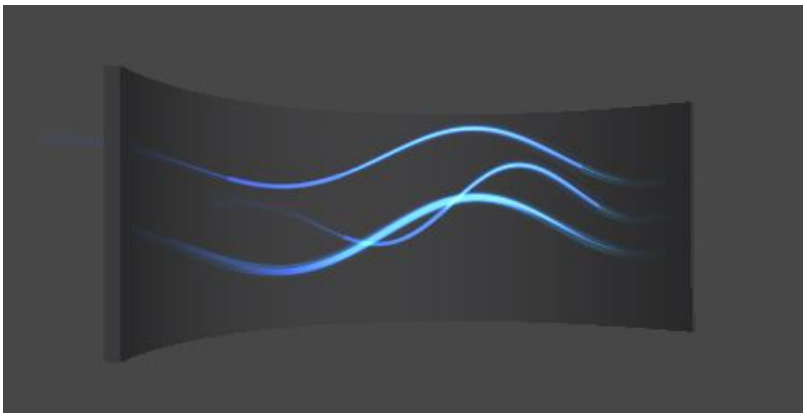


Рисунок 4.1.19 Модель фону гаражу

Фон гаражу(Рисунок 4.1.19 Модель фону гаражу) також зроблено щоб підкреслити авто перед ним. Була додана анімація плаваючих ліній щоб складена композиція не виглядала однотонно.

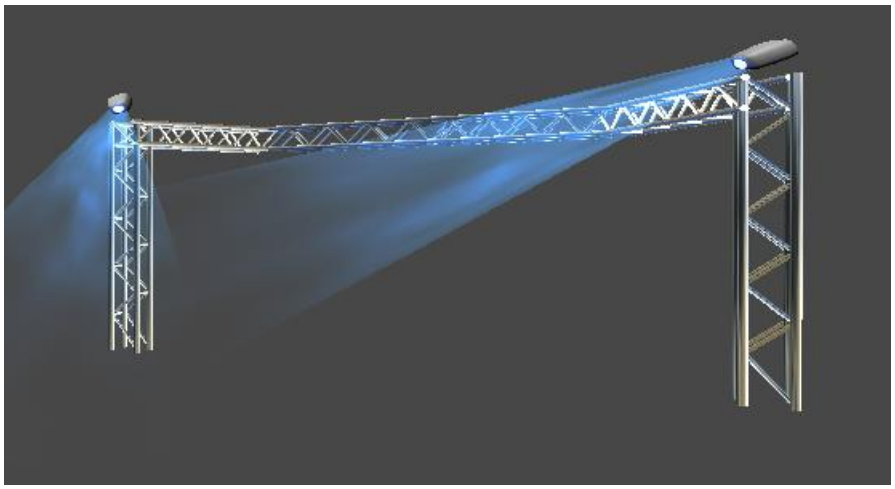


Рисунок 4.1.20 Модель балок та ліхтарів

Балки та ліхтарі(Рисунок 4.1.20 Модель балок та ліхтарів) створені для додання антуражу до гаража.

4.2 Створення 2D моделей у Blender

Я розбив 2D моделі за типами для зручності їх представлення. Типи моделей були встановлені згідно їх розміру та одноманітності стилю. Зображення для екрана завантаження(Рисунок 4.2.1 Фонове зображення) та полоса загрузки(Рисунок 4.2.2 Полоса завантаження) це перше що гравець бачить при відкритті мобільного додатку, тому саме вони задають подальший стиль всього інтерфейсу. Для гри були обрані яскраві проте спокійні неонові тона фіолетового зеленого та синього.

Рисунок 4.2.1 Фонове зображення

Рисунок 4.2.2 Полоса завантаження

Як і меню загрузки клавіші(Рисунок 4.2.3 Клавіша округла)(Рисунок 4.2.4 Клавіша прямокутна) вирішено було виконати в єдиному неновому стилі. Також на них було додано ефект освічення.

Рисунок 4.2.3 Клавіша округла

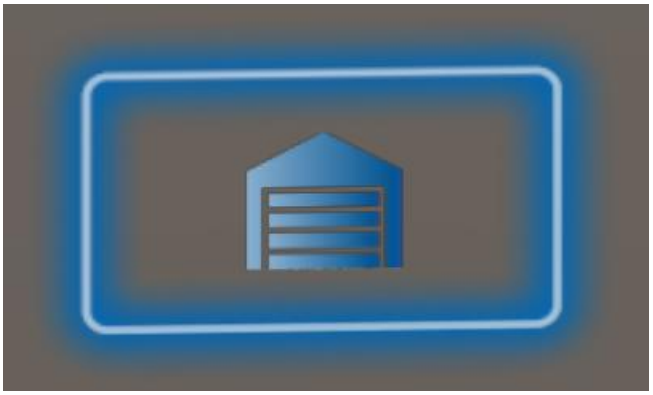


Рисунок 4.2.4 Клавіша прямокутна

Клавіші для придбання покращення транспорту(Рисунок 4.2.5 Модель клавіші покращення) було виконано наслідуючи вид реальних частин машини. Ці зображення помістили на синій фон для контрастності.

Рисунок 4.2.5 Модель клавіші покращення

Для головної валюти гри(Рисунок 4.2.6 Модель валюти) було обрано таймер який рухається. Це зображення відображає головну суть ігор про перегони, а

саме швидкість і час.

Рисунок 4.2.6 Модель валюти

Клавіші для рівнів(Рисунок 4.2.7 Модель клавіші рівня) було вирішено стилізувати під машини на різних трасах для надання динамічності. Також на кожній із них було написана суть рівня.

Рисунок 4.2.7 Модель клавіші рівня

Клавіші вибору карти були зроблені за схожим стилем.(Рисунок 4.2.8 Модель клавіші карти)

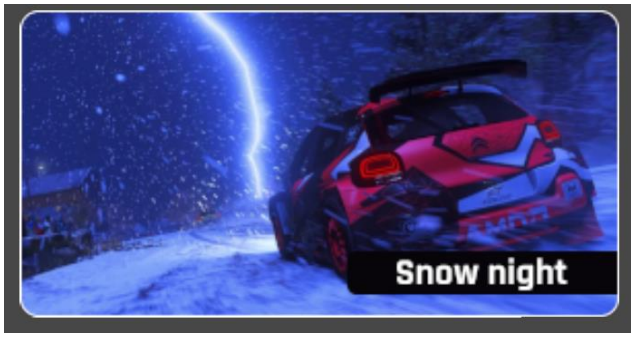


Рисунок 4.2.8 Модель клавіші карти

Для керування були використані клавіші із дизайном педалей(Рисунок 4.2.10 Клавіша газу) та неонова клавіша прискорення(Рисунок 4.2.9 Клавіша прискорення).

Рисунок 4.2.9 Клавіша прискорення

Рисунок 4.2.10 Клавіша газу

4.3 Інтеграція моделей до Unity

Після створення моделей наступним кроком буде експорт їх у формат, який можна буде легко імпортувати і використовувати в Unity.

Blender підтримує різноманітні формати файлів для експорту ресурсів, деякі з яких підходять для Unity. Найпоширенішими форматами файлів для експорту ресурсів з Blender до Unity є FBX (Filmbox): FBX - це універсальний формат файлів, який підтримує геометрію, матеріали, анімацію та інші дані ресурсів. Він широко підтримується Unity і забезпечує повну інтеграцію між Blender та Unity; при експорті до FBX необхідно обов'язково додати всі необхідні дані, такі як текстури, UV та анімацію.

OBJ (Wavefront Objects): OBJ - це широко підтримуваний формат файлів, який фокусується в першу чергу на геометрії і підходить для імпорту статичних 3D-моделей з Blender в Unity. Однак, OBJ файли не підтримують розширені можливості, такі як матеріали та анімації, тому можуть знадобитися додаткові кроки при налаштуванні матеріалів та анімацій в Unity. Цей формат я використав при імпортуванні моделей машин.

Перед експортом ресурсів з Blender важливо очистити сцену та оптимізувати

модель для рендерингу в реальному часі в Unity. Для цього слід

видалити об'єкти, освітлення, камери та елементи, які не потрібні у фінальній сцені в Unity. Це зменшує розмір файлу та спрощує ієрархію ресурсів.

Щоб експортувати ресурси з Blender для використання у Unity, потрібно виконати такі кроки

Виділити об'єкт або мережу, яку треба експортувати.

Вибрати експортувати з меню Файл (File) і вибрати відповідний формат файлу.

Далі треба обрати шлях експорту та ім'я файлу і натиснути кнопку "Експортувати", щоб зберегти ресурси у потрібне місце.

Після експорту ресурсів з Blender відкриваймо проект Unity і імпортуймо експортовані файли до редактора Unity. Щоб імпортувати ресурси треба створити папку у проекті Unity для зберігання імпортованих ресурсів. Перетягнути експортовані файли з File Explorer до редактора Unity, зокрема до створеної папки.

Unity автоматично імпортує ресурси, включно з геометрією, матеріалами та текстурами, до проекту. Такі параметри, як масштаб імпорту, матеріали та анімації, можна налаштувати у вікні Інспектора Unity.

4.4 Створення повноцінних меню та трас в Unity

Після інтеграції моделей до Unity можна перейти до створення меню та трас із усіма клавішами та об’єктами. Розпочнемо із меню завантаження. Зображення машини із назвою гри було накладено на чорний фон та відцентровано(Рисунок 4.4.1 Меню завантаження). Полоса завантаження поміщена під зображення.

Рисунок 4.4.1 Меню завантаження

На стартовому меню гри(Рисунок 4.4.2 Стартове меню) розташовані всі елементи навігації. У верхньому лівому кутку розташовані клавіші переходу до меню: виходу із гри, оцінки додатку та налаштувань. У верхньому правому кутку розміщений лічильник зібраної валюти та клавіша яка відображає поповнення валюти. При натисканні на лічильник користувач перейде до меню перегляду відео за валюту. По центру знизу розташована клавіша переходу до меню вибору

Рисунок 4.4.2 Стартове меню

рівня. Вона розташована так щоб першою привертати погляд користувача. Поряд із нею з лівої сторони меню перегляду відео за валюту а з правої меню гаража.

Кнопка виходу із додатку зроблена червоним кольором(Рисунок 4.4.3 Меню виходу) бо червоний символізує небезпеку. Аналогічну цьому клавіша яка дозволяє залишитися зелена.

Рисунок 4.4.3 Меню виходу

Оцінка гри зображена(Рисунок 4.4.4 Меню оцінки) зірками як у більшості популярних мобільних ігор. Натиснувши на зірку користувач звоже оцінити гру.

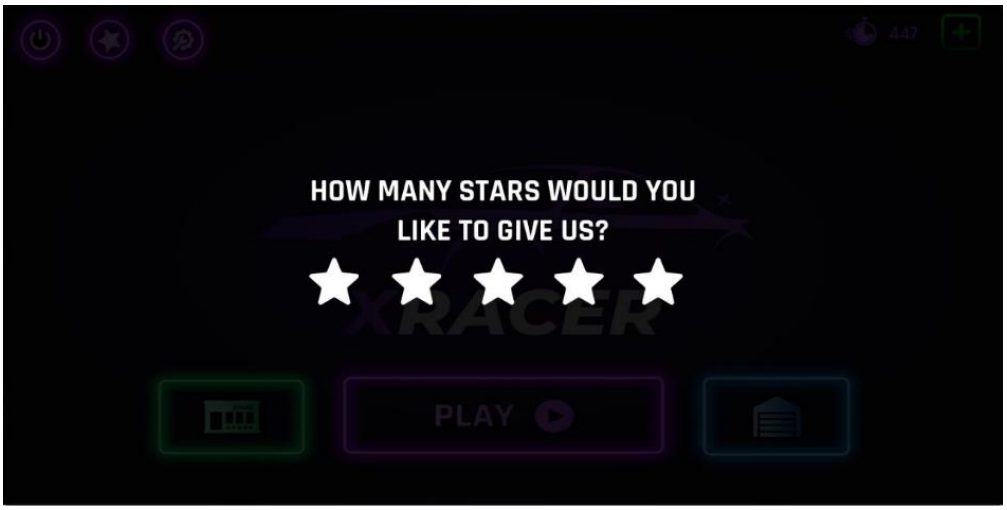


Рисунок 4.4.4 Меню оцінки

Налаштування(Рисунок 4.4.5 Меню налаштування) створено із застосуванням повзунків для зручності користування.

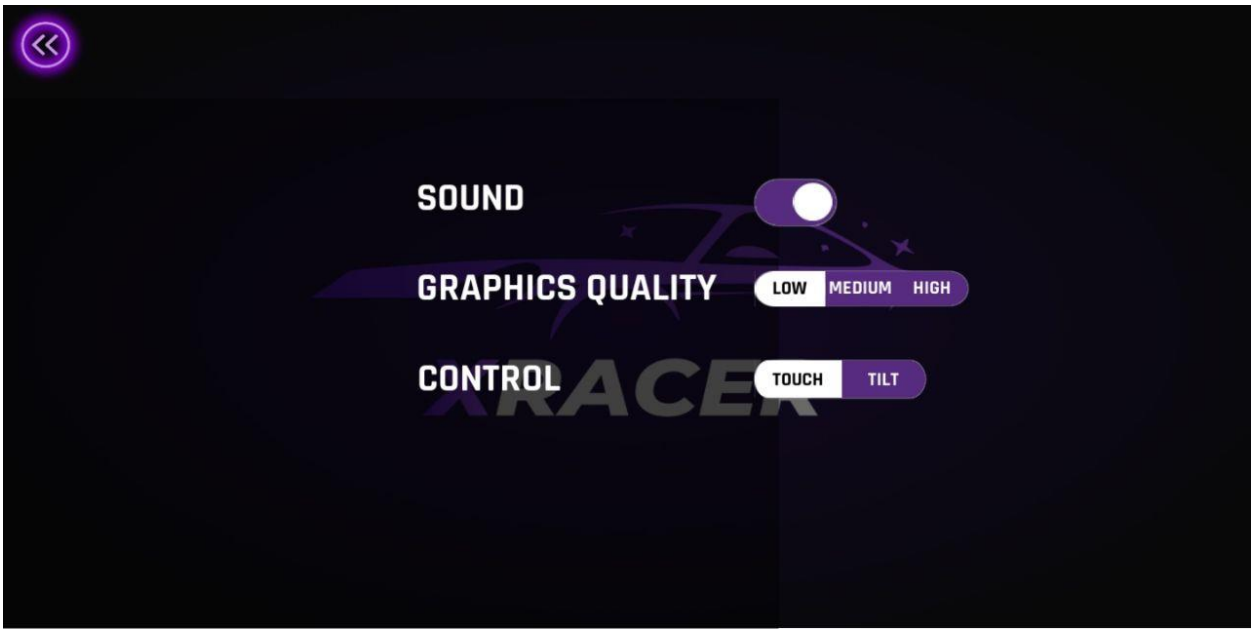


Рисунок 4.4.5 Меню налаштування

В гаражі(Рисунок 4.4.6 Меню гаража) користувач може побачити зібраний гараж із 3D моделей. Також наглядно видно характеристики авто. Посередині знизу розташована клавіша вибору авто. Зліва клавіша покращення а з правої сторони клавіша фарбування.

Рисунок 4.4.6 Меню гаража

Відкривши покращення(Рисунок 4.4.7 Покращення авто) користувач побачить клавіші які відповідають за покращення окремих частин авто. Ці клавіші було вирішено розташувати на сірому фоні із кнопкою закриття праворуч. Покращивши авто користувач побачить заповнення характеристик та збільшення кількості зірок на елементі авто.

Рисунок 4.4.7 Покращення авто

Фарбування(Рисунок 4.4.8 Фарбування авто) авто представить гравцеві палітрою кольорів на аналогічному спливаючому меню як із покращенням.

Рисунок 4.4.8 Фарбування авто

В меню перегляду відео за валюту(Рисунок 4.4.8 Меню перегляду відео за валюту) перед гравцем предстане вибір із нагород за перегляд.

Рисунок 4.4.8 Меню перегляду відео за валюту

У меню вибору рівня(Рисунок 4.4.9 Меню вибору рівня) недоступні рівні зробив сірими із повішеним замком. Рівні щільно розташовані один до одного щоб створити уявлення цілісності.

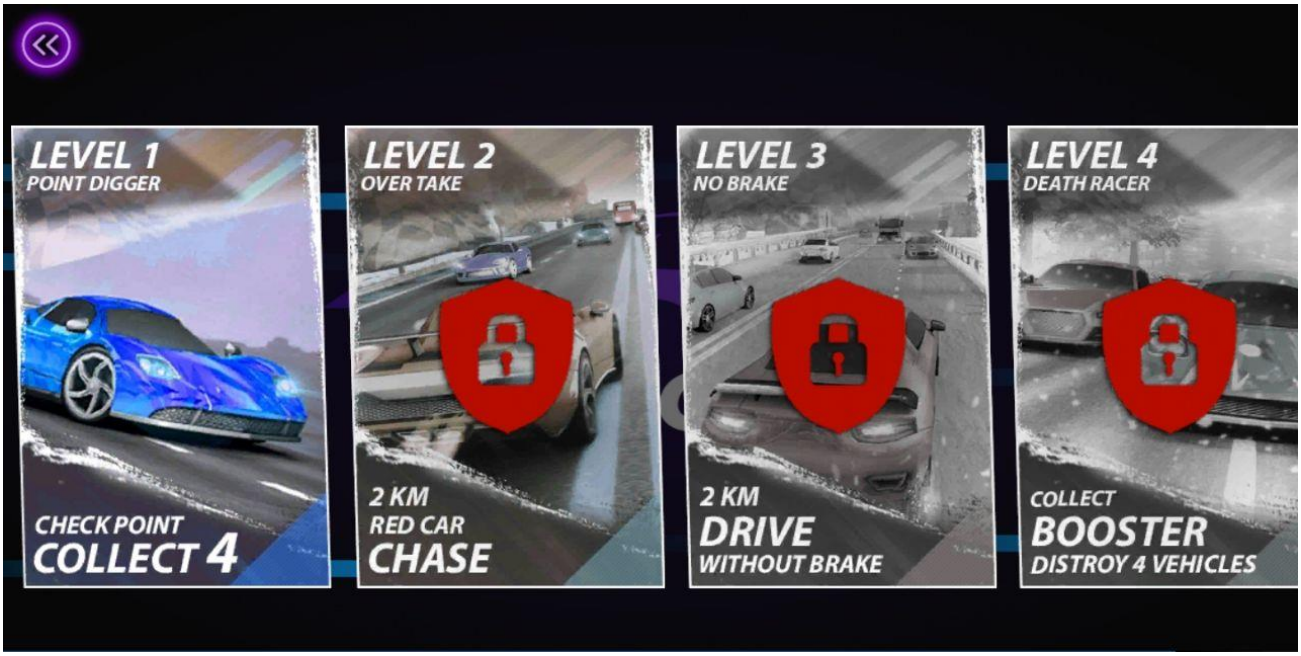


Рисунок 4.4.9 Меню вибору рівня

Для придання свіжості зображення до меню вибору руху(Рисунок 4.4.10 Меню вибору руху) було вирішено додати сині смуги.

Рисунок 4.4.10 Меню вибору руху

Аналогічно із вибором рівня у меню вибору трас(Рисунок 4.4.11 Меню вибору руху) було додано елемент замку. Для трас які треба придбати. Також для кожної траси додана ціна. Із підвищенням ціни за трасу підвищується її рівень складності який виражається у дорожньому покритті та кількості машин.

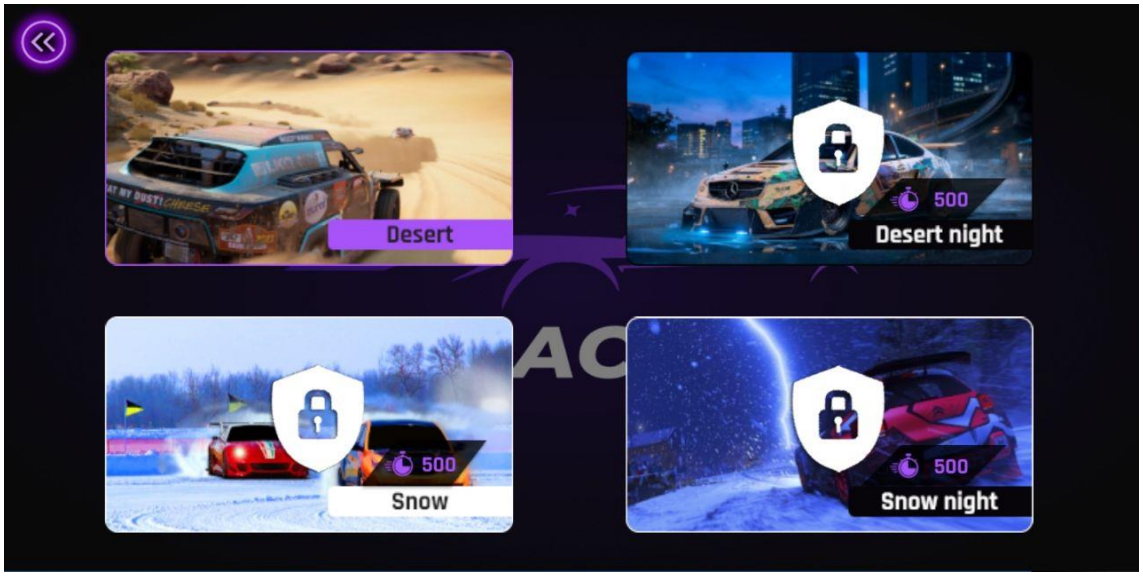


Рисунок 4.4.11 Меню вибору руху

На ігровому HUD(Рисунок 4.4.12 Ігровий HUD) розташовані такі елементи як:керуючі клавіші зліва для маневрування, праворуч розташовані педалі гальма та газу для прискорення та уповільнення, також поруч розташована клавіша прискорення, над нею помістилися лічильники очок та пройденої дистанції, зверху посередині міститься шкала прискорення, ліворуч від неї знаходиться лічильник кількості життів кнопка переходу до меню паузи та кнопка зміни виду, знизу посередині розташований рекламний банер.

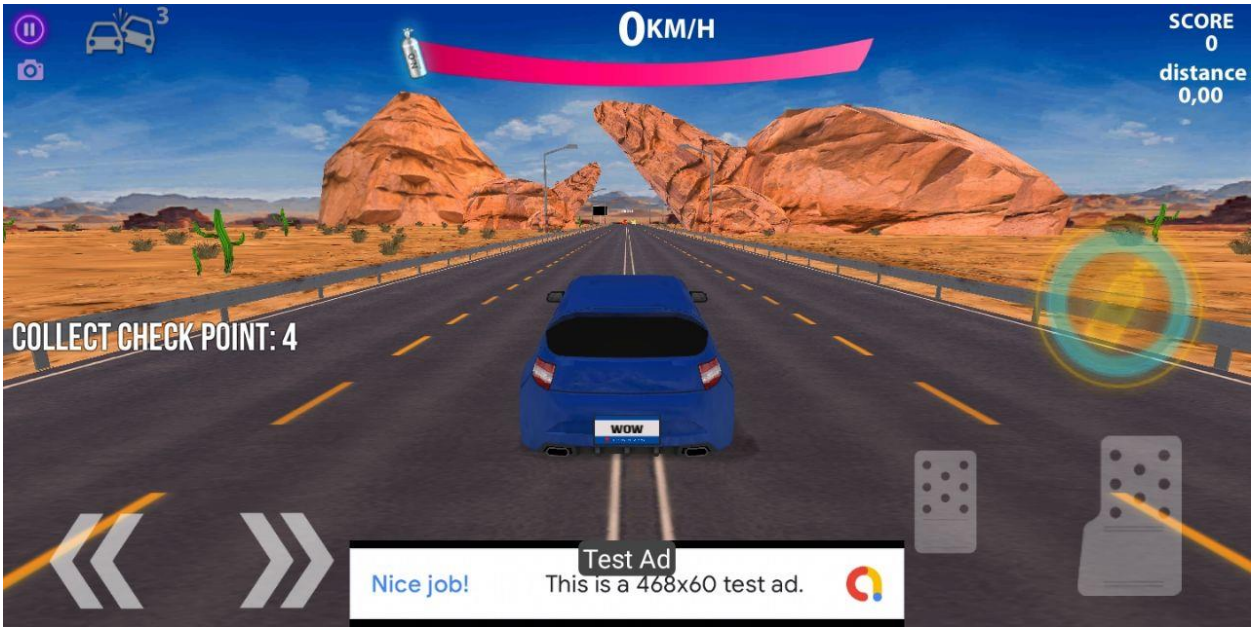


Рисунок 4.4.12 Ігровий HUD

В меню виграшу\програшу(Рисунок 4.4.13 Меню виграшу\програшу) розташовані лічильник зібраних очок, та лічильники що підраховують кількість зароблені валюти. Також своє місце мають клавіші навігації.

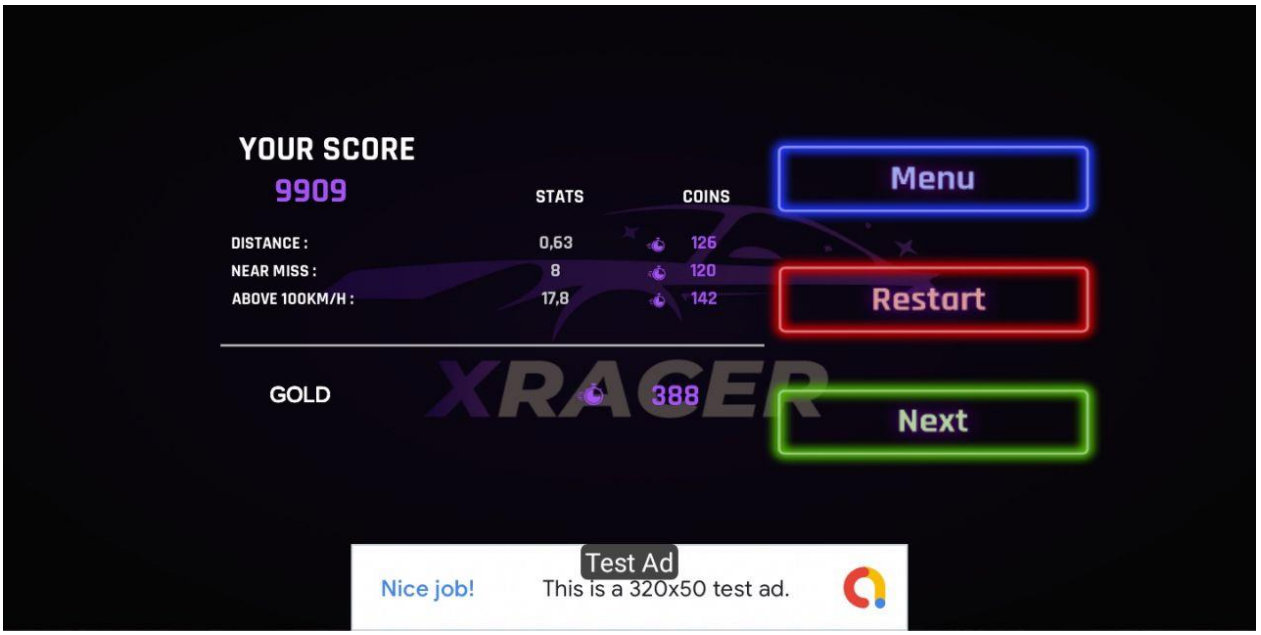


Рисунок 4.4.13 Меню виграшу\програшу

Також слід не забувати поставити в кожному із меню кнопки навігації для зручного маневрування між різними меню.

Для створення трас(Рисунок 4.4.14 Траса пустелі в день) була використана велика кількість різних 3D моделей у ролі фону та неба виступають об’єкти типу Plane. Кущі на околицях дороги використовують ефект паралаксу для надання 3D об’єму хоча самі являють собою 2D зображення.

Рисунок 4.4.14 Траса пустелі в день

4.5 Налагодження роботи меню

Меню створені та закріплені проте вони не будуть працювати без прикріпленої до них кодової частини. Загалом всі клавіши мають зв'язок через єдиний клас. Цей клас вирішує яку саме меню треба відкрити. Саме до нього надсилають запити всі кнопки навігації.

```
public void GarageButton() //open garage
{
    playClickSound();
    if (!mainPanel.activeInHierarchy)
    {
        bool isShow = isUnlockCarAvailable();
        if (isShow)
            Invoke("openUnlockCarPanel", 2f);
    }
    mainPanel.SetActive(false);
    carsSelectionPanel.SetActive(true);
    maincamera.SetActive(false);
    cars[carcounter].SetActive(true);
    carSelectionCamera.SetActive(true);
    currentCarStatus();
    checkUnlockCarStatus();
    CarUpgradation.instance.setCurrentUpgradateUI(cars[carcounter].GetComponent<CarStats>().data, carcounter);
    PlayerPrefs.SetInt("unlockcar", 1);
    changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
    print("Open Grage");
}
```

Рисунок 4.5.1 Функція відкриття гаражу

Наприклад функція відкриття гаражу(Рисунок 4.5.1 Функція відкриття гаражу) знаходиться в цьому класі, проте завдяки двигуну Unity я маю можливість прикріпити її до об'єкту кнопки. Також у самій кнопці з'єдную функцію із трігером OnClick. Це дозволить визвати функцію відкриття меню гаража натискаючи на кнопку. Код кнопки закриття теж знаходиться в класі меню. Хоча з однієї сторони це дуже велике нагромадження коду. Але не зважаючи на це, такий метод кодування дозволяє не плодити сутності та мати всю навігацію в одному місці.

```
public void GarageBacktoMainButton() //close garage
{
    playClickSound();
    mainPanel.SetActive(true);
    carsSelectionPanel.SetActive(false);
    maincamera.SetActive(true);
    carSelectionCamera.SetActive(false);
}
```

Рисунок 4.5.2 Функція повертання до меню

Викликаючи функцію відкриття гаражу(Рисунок 4.5.2 Функція повертання до меню) програма робить інші меню неактивними. Із початку роботи програми вона створює всі сутності та об’єкти але робить їх неактивними. Цей крок дозволяє економити час при загрузці різних елементів. Бо пропадає потреба в їх загрузці вони вже є на сцені з самого початку. Код робить їх активними та неактивними коли це потрібно, що призупиняє їх роботу та візуально убирає від спостерігача.

```
public void NextCarButton() //show next car
{
    playClickSound();
    colorPanel.SetActive(false);
    upgradePanel.SetActive(false);
    cannotApplyColor.SetActive(false);
    OrbitCam.ins.OnChange();
    nextBtn.GetComponent<Button>().enabled = false;
    PrevBtn.GetComponent<Button>().enabled = false;

    Invoke("OnSecond", 0.1f);

    if (carcounter == cars.Length - 1)
        carcounter = 0;
    else
        carcounter++;
    MoveCars(carcounter);
    currentCarStatus();
    if (PlayerPrefs.GetInt("unlockcar") == 1)
    {
        //unlockCarPanel.SetActive(true);
        _adInterstatial.ShowAd();
        PlayerPrefs.SetInt("unlockcar", 0);
    }
    changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
}
```

Рисунок 4.5.3 Функція перегляду машин

Завдяки функції перегляду машин(Рисунок 4.5.3 Функція перегляду машин) користувач може дивитися на різні типи доступних авто. Як видно із коду використовуєтеся карутина для відстеження руху машини. Це зроблено для того щоб машини плавно заміняли один одного при їх заміні. Для переміщення машини(Рисунок 4.5.4 Функція переміщення машини) по осі координат були використані вектори.

```
private void MoveCars(int target)
{
    _allCars.transform.localPosition = new Vector3(target, _allCars.transform.localPosition.y, _allCars.transform.localPosition.z);
}
```

Рисунок 4.5.4 Функція переміщення машини

Завдяки функції currentCarStatus(Рисунок 4.5.5 Функція статусу машини) відображаються дані авто якщо вона куплена. А якщо ні відображаєтеся ціна

```
void currentCarStatus()
{
    carNameLabel.text = getCarName();
    if (carcounter == 0)
    {
        carPriceLabel.text = "";
        unlockButton.SetActive(false);
        upgradeBtn.SetActive(true);
        paintBtn.SetActive(true);
        selectBtn.SetActive(true);
    }
    else
    {
        if (PlayerPrefs.GetInt("CAR" + carcounter) != 1)
        { //locked car
            carPriceLabel.text = "BUY FOR $" + getCarPrice(carcounter - 1).ToString();
            unlockButton.SetActive(true);
            upgradeBtn.SetActive(false);
            paintBtn.SetActive(false);
            selectBtn.SetActive(false);
        }
        else
        { //unlocked car
            unlockButton.SetActive(false);
            carPriceLabel.text = "";
            upgradeBtn.SetActive(true);
            paintBtn.SetActive(true);
            selectBtn.SetActive(true);
        }
    }
    if (cars[carcounter].GetComponent<CarStats>().data.canChangeColor)
        CarUpgradation.instance.changeCarColor(0, cars[carcounter].GetComponent<CarStats>().data);

    CarUpgradation.instance.setCurrentUpgradateUI(cars[carcounter].GetComponent<CarStats>().data, carcounter);
}
```

Рисунок 4.5.5 Функція статусу машини

Завдяки інтегрованому ассету DOTween створена плавна анімація появлення та зникання меню покращень(Рисунок 4.5.6 Функція появлення меню покращення).

```
public void UpgradeCarButton()
{
    playClickSound();
    upgradePanel.SetActive(true);
    upgraderAnim.GetComponent<DOTweenAnimation>().DORestart();
    cannotApplyColor.SetActive(false);
}
```

Рисунок 4.5.6 Функція появлення меню покращення

Висновок до пункту 4.

В цьому розділі були розроблені та налагоджені 3D та 2D моделі розроблюваного продукту. Для наглядної демонстрації були використані зображення розроблюваних елементів.

РОЗДІЛ 5
ТЕСТУВАННЯ ПРОГРАМИ

5.1 Тестування чорною скринькою

Тестування чорною скринькою буде проводитися за допомогою тест кейсів. Воно буде перевіряти працездатність функціоналу програми за різних умов.

Таблиця тест кейсів.

Опис тесту	Шаги відтворення	Очікувані результати	Статус
Відкриття головного меню гри	1.Тапнути на значок гри на робочому столі 2. Зверніути увагу на екран.	Перед гравцем відкривається головний екран гри після запуску програми.	Пройдено
Вихід із гри.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку вимкнення; 3. Тапнути на кнопку "Yes".	Гравець успішно виходить із гри після натискання на кнопку "Yes".	Пройдено
Повернення до головного меню.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку вимкнення; 3. Тапнути на кнопку "No".	Гравець успішно повертається до головного меню після натискання на кнопку "No".	Пройдено

Перехід до налаштувань .	1. Запустити гру "XRacer"; 2. Тапнути на кнопку "гайковий ключ" .	Гравцю відкривається меню налаштувань гри після натискання на кнопку налаштувань (гайковий ключ).	Пройдено
Увімкнення/вимкнення звуку.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку налаштувань "гайковий ключ"; 3. Тапнути на тумблер "Sound"; 4. Звернути увагу, що звук вимкнувся; 5. Повторно тапнути на тумблер "Sound"; 6. Звернути увагу, що звук увімкнувся.	Гравець регулює увімкнення/вимкнення звуку в грі за допомогою тапа на тумблер "Sound" в налаштуваннях.	Пройдено
Регулювання графіка гри.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку налаштувань "гайковий ключ";	Гравець регулює якість графіки у грі за допомогою тапа на кнопки "Graphics Quality" у налаштуваннях.	Пройдено

	3. Вибрати 1 із 3 варіантів графіки: Low/Medium/High.		
Зміна управління у грі.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку налаштувань "гайковий ключ"; 3. Вибрати 1 з 2 типів управління: Touch/Tilt (Auto).	Гравець змінює керування автомобілем з кнопкового на автоматичний (за допомогою повороту пристрою) завдяки тапам на кнопки "Control" в налаштуваннях.	Пройдено
Повернення до головного меню.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку налаштувань "гайковий ключ"; 3. Тапнути на кнопку "Назад" (стрілочки вліво).	Гравець повертається у головне меню гри після натискання на кнопку "Назад" (стрілочки вліво)	Пройдено
Беззвучний режим.	1.Вимкнути звук на пристрої; 2. Включити звук у грі; 3. Звернути увагу на	Коли на пристрої ввімкнено безучний режим, а у грі перемикач звуку увімкнено, ви не почуєте звукові ефекти та музику у грі	Пройдено

	відсутність звуку в грі.	через відключений звук на пристрої.	
Перехід до розділу "Магазин" через кнопку "+"	1. Запустити гру "XRacer"; 2. Тапнути на кнопку "+".	Гравцеві відкривається сторінка магазину після натискання на кнопку "+".	Пройдено
Перехід до розділу "Магазин" через кнопку "Store"	1. Запустити гру "XRacer"; 2. Тапнути на кнопку "Store".	Гравцю відкривається сторінка магазину після натискання на кнопку "Store".	Пройдено
Купівля № 1 (300 монет)	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на піктограму перегляду реклами (кнопка play); 4. Зачекати 7 секунд реклами; 5. Тапнути на закриття екрана; 6. Звернути увагу на внутрішньоігровий баланс.	Баланс гравця поповнюється 300 монет після перегляду реклами.	Пройдено

Покупка №2 (600 монет)	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на піктограму перегляду реклами (кнопка play); 4. Зачекати 7 секунд реклами; 5. Тапнути на закриття екрана; 6. Звернути увагу на внутрішньоігровий баланс.	Баланс гравця поповнюється на 600 монет після перегляду реклами.	Пройдено
Купівля № 3 (900 монет)	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на піктограму перегляду реклами (кнопка play); 4. Зачекати 7 секунд реклами; 5. Тапнути на закриття екрана;	Баланс гравця поповнюється на 900 монет після перегляду реклами.	Пройдено

	6. Звернути увагу на внутрішньоігровий баланс.		
Купівля № 4 (1200 монет)	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на піктограму перегляду реклами (кнопка play); 4. Зачекати 7 секунд реклами; 5. Тапнути на закриття екрана; 6. Звернути увагу на внутрішньоігровий баланс.	Баланс гравця поповнюється на 1200 монет після перегляду реклами.	Пройдено
Купівля бандла № 5 (1500 монет)	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на піктограму перегляду реклами (кнопка play);	Баланс гравця поповнюється на 1500 монет після перегляду реклами.	Пройдено

	4. Зачекати 7 секунд реклами; 5. Тапнути на закриття екрана; 6. Звернути увагу на внутрішньоігровий баланс.		
Вихід із магазину.	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Тапнути на кнопку "Назад" (стрілочки вліво)	Гравець успішно повертається в меню гри після натискання на кнопку "Назад"	Пройдено
Повторне придбання після успішного перегляду реклами	1. Запустити гру "XRacer"; 2. Перейти до розділу "Магазин"; 3. Успішно здобути будь-яку нагороду за перегляд реклами; 4. Повторно тапнути на цей же піктограму перегляду реклами.	Відображається інформер "Ads are not ready!" щоб гравець не зміг накручувати собі баланс за допомогою безперервного перегляду реклами.	Пройдено

Перехід у гараж.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок).	Гравець переходить на екран гаража.	Пройдено
Вихід із гаража.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Тапнути на кнопку "Назад" (стрілки вліво).	Гравець успішно повертається до головного меню гри.	Пройдено
Свайп автомобілів у гаражі.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Тапнути на кнопки вліво/вправо; 4. Звернути увагу на вітрину.	Гравець успішно гортає карусель автомобілів на вітрині.	Пройдено
Купівля автомобіля.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок);	Куплена машина успішно додається в гараж, а з рахунку гравця списується сума, що дорівнює ціні автомобіля.	Пройдено

	3. Дістати до машини з кнопкою "Buy for..."; 4. Тапнути на цю кнопку.		
Стайлінг автомобіля.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Вибрати своє авто, яке хочемо стилізувати; 4. Тапнути на кнопку стайлінгу (болнчик); 5. Вибрати потрібний колір; 6. Тапнути на закриття екрана стайлінгу (стрілки вниз)	Гравець успішно застосовує обраний колір до даного автомобіля.	Пройдено
Купівля нового кольору.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Вибрати своє авто, яке	Гравець успішно купує новий колір для автомобіля.	Пройдено

	хочемо стилізувати; 4. Тапнути на кнопку стайлінгу (болнчик); 5. Вибрати потрібний платний колір; 6. Тапнути на кнопку "Buy Color".		
Детейлінг автомобіля.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Вибрати своє авто, яке хочемо апгрейдити;4. Тапнути на кнопку дітейлінгу (гайковий ключ); 5. Тапнути на необхідну характеристику;	Гравець успішно апгрейдить обрану деталь автомобіля.	Пройдено
Фоторежим.	1. Запустити гру "XRacer";	Перед гравцем залишиться тільки вітрина з автомобілем, який можна крутити і	Пройдено

	2. Тапнути на кнопку гаража (будиночок); 3. Тапнути на кнопку фоторежиму (фотоапарат).	зробити гарний скріншот.	
Вихід із фоторежиму	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Тапнути на кнопку фоторежиму (фотоапарат); 4. Повторити п.3	Гравець повертається до головного меню гри після повторного натискання на кнопку фоторежиму.	Пройдено
Купівля автомобіля без грошей.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Дістати до машини з кнопкою "Buy for..."; 4. Тапнути на цю кнопку.	Гравця переводять на сторінку магазину, де він може придбати внутрішньоігрову валюту.	Пройдено
Купівля нового кольору без грошей.	1. Запустити гру "XRacer";	Якщо у Гравця не вистачає коштів, йому	Пройдено

	2. Тапнути на кнопку гаража (будиночок); 3. Вибрати своє авто, яке хочемо стилізувати; 4. Тапнути на кнопку стайлінгу (болнчик); 5. Вибрати потрібний платний колір; 6. Тапнути на кнопку "Buy Color".	відображається курсор, який вказує на його баланс.	
Купівля нової деталі без грошей.	1. Запустити гру "XRacer"; 2. Тапнути на кнопку гаража (будиночок); 3. Вибрати своє авто, яке хочемо апгрейдити; 4. Тапнути на кнопку дітейлінгу (гайковий ключ); 5. Вибрати потрібну деталь.	Якщо у Гравця не вистачає коштів, йому відображається курсор, який вказує на його баланс.	Пройдено

Висновок до пункту 5.

В цьому розділі розроблений продукт був протестований. Для наглядної демонстрації була створена таблиця кейсів.

РОЗДІЛ 6
ВИСНОВКИ

Отже, це дослідження було присвячене візуальному створенню мобільних гоночних додатків з використанням ігрового рушія Unity та Blender для 3D-моделювання. Під час дослідження було проаналізовано різні аспекти мобільної ігрової індустрії, характеристики гоночних ігор та важливість візуальних елементів для покращення ігрового досвіду. Також було досліджено еволюцію мобільних гоночних ігор, яка показала, як розвиток технологій та дизайну сформував цей жанр.

Ігровий рушій Unity зарекомендував себе як потужний та універсальний інструмент для розробки мобільних гоночних додатків. Я описав його особливості, призначені для мобільних платформ, включаючи оптимізовану продуктивність, крос-платформну сумісність та надійні інструменти розробки. Він також детально зупинився на можливостях Unity щодо створення середовища та треків, підкресливши, наскільки легко створювати реалістичні та захоплюючі гоночні середовища.

Робота також надала вичерпну інформацію про моделювання транспортних засобів та фізичне моделювання в рушії Unity. Від дизайну та моделювання транспортних засобів до реалізації реалістичної фізики, вона продемонструвала, що Unity дозволяє створювати захопливі та динамічні перегони.

Крім того, було детально обговорено використання Blender'a для 3D-моделювання, розглянуто інтуїтивно зрозумілий інтерфейс Blender'a, інструменти навігації, базові методи полігонального моделювання, редагування сітки, скульптування та ретопології.

Нарешті, стаття пояснює, як легко інтегрувати ресурси між Blender'ом та Unity. Дотримуючись рекомендованої процедури експорту та оптимізуючи ресурси для рендерингу в реальному часі, розробники можуть ефективно

переносити 3D-моделі, текстури та анімацію з Blender в Unity для безперешкодного робочого процесу.

Таким чином, поєднання Unity та Blender надає потужний набір інструментів для створення візуально приголомшливих мобільних гоночних додатків. Ми сподіваємося, що ця стаття дала вам повне уявлення про процеси, пов'язані з конвеєром візуальної розробки, від створення концепції та моделювання ресурсів у Blender'і до їхньої реалізації в Unity для створення захоплюючого гоночного досвіду.

Використовуючи можливості цих двох стандартних інструментів, розробники ігор можуть розкрити свій творчий потенціал і втілити в життя захопливі мобільні гоночні ігри. У майбутньому ігровий рушій Unity та Blender будуть розвиватися, відкриваючи нові можливості для візуального дизайну з великим потенціалом для розширення меж мобільних ігор.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Manning, J., Buttfield-Addison, P. (2017). Mobile Game Development with Unity: Build Once, Deploy Anywhere. Сполучені Штати Америки: O'Reilly Media.

2. Hocking, J. (2015). Unity in Action: Multiplatform Game Development in C#. Сполучені Штати Америки: Manning Publications Company.

3. Manning, J., Nugent, T., Buttfield-Addison, P. (2023). Unity Development Cookbook: Real-Time Solutions from Game Development to AI. (n.p.): O'Reilly Media.

4. Simonds, B. (2013). Blender Master Class: A Hands-on Guide to Modeling, Sculpting, Materials, and Rendering. Сполучені Штати Америки: No Starch Press.

5. Doran, J. P. (2014). Unity Game Development Blueprints. Велика Британія: Packt Publishing.

6. Hess, R. (2013). Blender Foundations: The Essential Guide to Learning Blender 2.5. Нідерланди: CRC Press.

7. Goldstone, W. (2011). Unity 3.x Game Development Essentials: Game Development with C# and Javascript. Велика Британія: Packt Publishing.

8. Fisher, G. (2014). Blender 3D Basics: Beginner's Guide. Велика Британія: Packt Publishing, Limited.

9. Hocking, J. (2015). Unity in Action: Multiplatform Game Development in C#. Сполучені Штати Америки: Manning Publications Company.

10. Simonds, B. (2013). Blender Master Class: A Hands-on Guide to Modeling, Sculpting, Materials, and Rendering. Сполучені Штати Америки: No Starch Press.

11. Geig, M. (2018). Unity 2018 Game Development in 24 Hours, Sams Teach Yourself. Велика Британія: Pearson Education."Unity Virtual Reality Projects" by Jonathan Linowes

12. Villar, O. (2014). Learning Blender: A Hands-On Guide to Creating 3D Animated Characters. Велика Британія: Pearson Education.

13. Castilhos Melo, F., Somma, V. (2017). Blender 3D Printing by Example.:

Learn to Use Blender's Modeling Tools for 3D Printing by Creating 4 Projects. Велика Британія: Packt Publishing.

14. Van Gumster, J. (2020). Blender For Dummies. Велика Британія: Wiley. "Unity in Action: Multiplatform Game Development in Unity 5" by Joe Hocking

15. Fisher, G. (2014). Blender 3D Basics: Beginner's Guide. Велика Британія: Packt Publishing, Limited.

ТЕКСТ ПРОГРАМИ

MenuScript.cs

```
using UnityEngine;
using System.Collections;
using System;
using UnityEngine.UI;
using DG.Tweening;
using UnityEngine.Analytics;
using GoogleMobileAds.Api;
[RequireComponent(typeof(AdInterstatial))]

public class MenuScript : MonoBehaviour //contain main menu all panels functionality
{
    [SerializeField] private Button _selectOneWay, _selectTwoWays;
    [SerializeField] private GameObject NoPlateButton;
    [SerializeField] private AdInterstatial _adInterstatial;
    [SerializeField] private GameObject selectBtn;
    [SerializeField] private GameObject _allCars;
    [SerializeField] private AdBanner _adBanner;

    public static int x = 1;
    #region Variables declaration

    [Space(5)]
    [Header("Main Menu")]
    public GameObject mainPanel, topBar;
    public Text RewminigTimeReward;
    public GameObject getFreeCashPanel;
    public GameObject maincamera;

    [Space(5)]
    [Header("Rewards")]
    public GameObject firstTimeCashSprite;
    public GameObject dailyBonusPanel, dailyRewardSubPanel;
    public Image[] dailyRewardBoundary;
    public Button[] dailyRewardDays;
    public Color dayEnabled, dayDisabled;
    public Text dailyRewardUserName;

    [Space(5)]
    [Header("Links")]
    public LeaderboardManager leader;
    public string privacyLink;
    public GameObject shareWarning;

    [Space(5)]
    [Header("Setting/Quit")]
    public float soundBar;
    public GameObject quitgamePanel, quitAnimPanel;
```

```
    public GameObject settingsSprite, settingAnimSprite, Optionpanal, CreditsPanal, AboutPanal,
UpdatePanal;



    [Space(5)]
    [Header("Inapp Buttons")]
    public Button fullUnlockButon;

    public GameObject freePradoAd, freeImpossibleAd;

    public GameObject bundle, Gold, Denerio, freeCash, MoveBtns, youtubeAd, YoutubeText,
facebookAd, facebookText;

    [Space(5)]
    [Header("Level Selection")]
    public GameObject challangesSelection, RewardgesSelection;
    public GameObject descriptionBox;
    public GameObject[] LevelLocks, RewardLevelLocks, RewardedStar;
    public Button[] levelParent, RewardlevelParent;
    public Text LevelsDescText;
    public Button LevelNextButton, RewardLevelNextButton;
    public ScrollRect LevelSlider;
    float lastUnlocked;
    public LevelsData _description, DailyEventDes;

    [Space(5)]
    [Header("Environment Selection")]
    public GameObject envSelectionPanel, onewayImage;
    public GameObject[] envLocks, highWayLocks, highwayUnlock;
    public Text currencyEnv, DineroCurrency, garageCurrency, shopCurrency;
    public Text[] highwayUnlockText;

    [Space(5)]
    [Header("HighWay Selection")]
    public GameObject highwaySelection;
    public GameObject[] highwayLocks;
    [Space(5)]
    [Header("Mode Selection")]
    public GameObject levelModeSelection;
    public GameObject trafficModeSelection;
    //public GameObject gameModeSelection;
    public Text bestScoreOneWay;
    public Text bestScoreTwoWay;
    public Text bestScoreTimeTrail;
    //public Text currencyMode;

    [Space(5)]
    [Header("Car Selection")]
```

```
public GameObject nextBtn;
public GameObject PrevBtn;

public GameObject carsSelectionPanel;
public GameObject unlockCarBtn;
public GameObject[] cars;
public Vector3[] carStartPos, carPos;
public Text carNameLabel;
public Text carPriceLabel;
public GameObject upgradeBtn;
public GameObject paintBtn;
public GameObject cannotApplyColor;
public Text customizePriceLabel;
//public Image lockSprite;
//public Image lockSpriteSub;
public GameObject carSelectionCamera;
public GameObject colorPanel, colorAnim;
public GameObject upgradePanel, upgraderAnim;
public GameObject unlockButton;
public static int selectedCar = 0;
public static int carcounter = 0;
int colorNumber;
bool upgradeBool, customizeBool;
float errortimer = 2;

public GameObject purchaseCashSprite;
[Header("Static variables")]
public static bool isLoadingAd;
public static int menuLoaded;
public static bool check, isShare;
public static string userName;
public static bool day;
static bool first = true;
public static int totalLevels, currentevent;
bool thisOnce = true;
public static MenuScript instance;
public AudioSource LoadingSound;
public GameObject NotiDailyEvent, SettingNoti, UpdateNoti, DailyEventTime;
#endregion
void Awake()
{
    //PlayerPrefs.DeleteAll ();
    //   PlayerPrefs.SetInt("Currency",500000);
    //  PlayerPrefs.SetInt("combo", 1000);
    instance = this;
    QualitySettings.vSyncCount = 0;
    QualitySettings.SetQualityLevel(0);
    Application.targetFrameRate = 30;
```

```
    }
    void Start()
    {
        _adBanner.DisableBanner();
        isLoadingAd = true; //show AD on loading panel
        Time.timeScale = 1f;
        PlayerPrefs.SetInt("LevelCompleted" + 0, 1);
        totalLevels = levelParent.Length;

        for (int t = 0; t < levelParent.Length; t++)
        {
            PlayerPrefs.SetInt("LevelCompleted" + t, 1); //all level unlock
            if (PlayerPrefs.GetInt("LevelCompleted" + t) == 1)
            {
                levelParent[t].GetComponent<RectTransform>().localScale = new Vector3(1.1f, 1.1f,
1.1f);
            }
        }
        if (!PlayerPrefs.HasKey("colorIdex"))
        {
            PlayerPrefs.SetInt("colorIdex", 1);
            PlayerPrefs.SetInt("CarIdex", 0);
            PlayerPrefs.SetInt("currentEvnt" + 1, 1);
        }
        if (!PlayerPrefs.HasKey("SettingNotification"))
        {
            SettingNoti.SetActive(true);
            //UpdateNoti.SetActive(true);
        }
        else
        {
            //SettingNoti.SetActive(false);
            //UpdateNoti.SetActive(false);
        }
        //initializers
        initializeLevelPanel();
        initializeGarage();
        initializeSetting();
        checkJoiningReward();
        initializeTrafficMode();
        CheckLevelLocks();
        dayCheck();
        setIAPButtons();
        showMegaPack();
        detectAndroidVersion();
        PlayerPrefs.SetInt("GameOverAd", 1); //reset Fail Panel Ad count
        for (int i = 0; i < cars.Length; i++)
        {
```

```
            carStartPos[i] = cars[i].transform.eulerAngles;
            carPos[i] = cars[i].transform.position;
        }
        for (int i = 1; i < 6; i++)
        {
            if (PlayerPrefs.GetInt("currentEvnt" + i) == 2)
            {
                RewardlevelParent[i - 1].GetComponent<Button>().interactable = false;
                NotiDailyEvent.SetActive(false);
                for (int s = 0; s < 5; s++)
                    RewardedStar[i - 1].transform.GetChild(s).GetComponent<Image>().color =
Color.yellow;
            }
            else if (PlayerPrefs.GetInt("currentEvnt" + i) == 1)
            {
                RewardlevelParent[i - 1].GetComponent<Button>().interactable = true;
                NotiDailyEvent.SetActive(true);
                print("event false");
            }


        }

        if (PlayerPrefs.GetInt("GotoLevels", 0) == 1)
        {
            SelectLevelsMode();
            PlayerPrefs.SetInt("GotoLevels", 0);
        }

        AudioListener.volume = PlayerPrefs.GetFloat("SOUND");
        int a = Convert.ToInt32(PlayerPrefs.GetFloat("SOUND"));
        ValueChangedSlider(a);


    }
    void Update()
    {

        int Hours, Minuts, Second;
        Minuts = System.DateTime.Now.Minute; Second = System.DateTime.Now.Second;
        Hours = PlayerPrefs.GetInt("RemainingTime") - PlayerPrefs.GetInt("CurrentSpintTime");
        Second = 60 - Second;
        if (Minuts == 60)
            Hours -= 1;
        Minuts = 60 - Minuts;
        RewminigTimeReward.text = (Hours + " : " + Minuts + " : " + Second.ToString());

        garageCurrency.text = PlayerPrefs.GetInt("Currency").ToString();
        currencyEnv.text = PlayerPrefs.GetInt("Currency").ToString();
```

```
        shopCurrency.text = PlayerPrefs.GetInt("Currency").ToString();
        if (PlayerPrefs.GetInt("changeAdd") == 1)
        {
            freePradoAd.SetActive(false);
            freeImpossibleAd.SetActive(true);
        }
        if (PlayerPrefs.GetInt("changeAdd1") == 1)
        {
            freePradoAd.SetActive(false);
            freeImpossibleAd.SetActive(false);
            MoveBtns.GetComponent<Animator>().enabled = true;
        }

        if (PlayerPrefs.GetInt("AD") == 1)
        {
            youtubeAd.GetComponent<Button>().enabled = false;
            YoutubeText.SetActive(true);
        }
        if (PlayerPrefs.GetInt("AD1") == 1)
        {
            facebookAd.GetComponent<Button>().enabled = false;
            facebookText.SetActive(true);
        }

    }
    void detectAndroidVersion()
    {
        //detect current android api level //Change
        //int apiLevel =
int.Parse(SystemInfo.operatingSystem.Substring(SystemInfo.operatingSystem.IndexOf("-") + 1,
3));
        //if (apiLevel <= 23)
        //{
        //    isShare = true;
        //}
        //else
        //{
        //    isShare = false;
        //}
    }
    void showMegaPack()
    {
        menuLoaded++;
        if (menuLoaded == 2 && !PlayerPrefs.GetInt("UNLOCKFULLGAME").Equals(1))
        {
            openMegaPack();
        }
    }
    #region Rewards
```

```
#region joining reward
public void checkJoiningReward() //show joining panel
{
   if (PlayerPrefs.GetInt("FIRSTTIME") != 1) //joining reward
      firstTimeCashSprite.SetActive(true);
}
public void ClaimCashButton() //get joining reward
{
   PlayerPrefs.SetInt("Currency", PlayerPrefs.GetInt("Currency") + 500);
   PlayerPrefs.SetInt("FIRSTTIME", 1);
   firstTimeCashSprite.SetActive(false);
}
#endregion

#region daily reward

public void OpenDailyRewardPanel()
{
   dailyBonusPanel.SetActive(true);
}

public void dayCheck() //check wether next day or not
{
   if (PlayerPrefs.HasKey("PlayDate"))
   { //if already played
      string stringDate = PlayerPrefs.GetString("PlayDate");
      DateTime oldDate = Convert.ToDateTime(stringDate); //saved date
      DateTime newDate = System.DateTime.Now; //current date
      TimeSpan difference = newDate.Subtract(oldDate);
      if (difference.Days == 1)
      { //if playing next day: give daily reward

         dailyBonusPanel.SetActive(true);
         int currentDay = PlayerPrefs.GetInt("BONUSDAY");
         if (currentDay != 7)
         {
            // Debug.Log (currentDay + "Current");
            for (int i = 0; i < dailyRewardDays.Length; i++)
            { //deactivate all btns except current
               //dailyRewardBoundary[i].color = dayDisabled;
               dailyRewardDays[i].interactable = false;
            }
            //print("day reward Name" + dailyRewardBoundary[currentDay -
1].transform.GetChild(0).gameObject.name);
            //dailyRewardBoundary[currentDay -
1].transform.GetChild(0).gameObject.SetActive(false);
            //dailyRewardBoundary[currentDay - 1].color = dayEnabled;
            dailyRewardDays[currentDay - 1].interactable = true;
            string newStringDate = Convert.ToString(newDate);
```

```
                PlayerPrefs.SetString("PlayDate", newStringDate);
            }
            else
            {
                Debug.Log("Relajfhsdhf");
                string newStringDate = Convert.ToString(System.DateTime.Now);
                PlayerPrefs.SetString("PlayDate", newStringDate);
                dailyBonusPanel.SetActive(true);
                for (int i = 1; i < dailyRewardDays.Length; i++)
                { //deactivate all btns except first
                    //dailyRewardBoundary[i].color = dayDisabled;
                    dailyRewardDays[i].interactable = false;
                }
                //dailyRewardBoundary[0].color = dayEnabled;
                dailyRewardDays[0].interactable = true;
                //print("day reward Name 2 " +
dailyRewardBoundary[0].transform.GetChild(0).gameObject.name);
                //dailyRewardBoundary[0].transform.GetChild(0).gameObject.SetActive(false);
            }
        }
        else if (difference.Days > 1)
        { //if not playing next day: reset playerprefs
            dailyBonusPanel.SetActive(true);
            for (int i = 1; i < dailyRewardDays.Length; i++)
            { //deactivate all btns except first
                //dailyRewardBoundary[i].color = dayDisabled;
                dailyRewardDays[i].interactable = false;
            }
            //dailyRewardBoundary[0].color = dayEnabled;
            dailyRewardDays[0].interactable = true;
            //print("day reward Name 1 " +
dailyRewardBoundary[0].transform.GetChild(0).gameObject.name);
            //dailyRewardBoundary[0].transform.GetChild(0).gameObject.SetActive(false);
            string newStringDate = Convert.ToString(newDate);
            PlayerPrefs.SetString("PlayDate", newStringDate);
        }
        else if (difference.Days < 1)
            return;
    }
    else
    { //playing ist time
        string newStringDate = Convert.ToString(System.DateTime.Now);
        PlayerPrefs.SetString("PlayDate", newStringDate);
        dailyBonusPanel.SetActive(true);
        for (int i = 1; i < dailyRewardDays.Length; i++)
        { //deactivate all btns except first
            //dailyRewardBoundary[i].color = dayDisabled;
            dailyRewardDays[i].interactable = false;
        }
```

```
            //dailyRewardBoundary[0].color = dayEnabled;
            dailyRewardDays[0].interactable = true;
            //print("day reward Name 0 " +
dailyRewardBoundary[0].transform.GetChild(0).gameObject.name);
            //dailyRewardBoundary[0].transform.GetChild(0).gameObject.SetActive(false);
        }
    }
    int dailyRewardValue(int index) //get daily reward value
    {
        int[] rewardValue = new int[] { 50, 100, 200, 500, 800, 1000 };
        return rewardValue[index];
    }
    public void DailyBonus(int day) //daily bonus btn clicked
    {
        int reward = dailyRewardValue(day - 1);
        PlayerPrefs.SetInt("Currency", PlayerPrefs.GetInt("Currency") + reward);
        PlayerPrefs.SetInt("BONUSDAY", day + 1);
        if (userName != null)
        {
            dailyRewardUserName.text = "Hey! " + userName;
        }
        dailyRewardDays[day - 1].interactable = false;
        dailyRewardSubPanel.SetActive(true);
    }
    public void CloseDailyRewardPanel()
    {
        playClickSound();
        dailyRewardSubPanel.SetActive(false);
        dailyBonusPanel.SetActive(false);
    }

    #endregion
    #endregion

    #region external links
    public void RateUsButton()
    {
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.door.modren.car.traffic.
apps&hl=en");
    }
    public void MoreGamesButton()
    {
        playClickSound();
        Application.OpenURL("https://play.google.com/store/apps/developer?id=Door+to+apps");
    }
    public void PrivacyPolicyLink()
    {
```

```
        playClickSound();
        Application.OpenURL(privacyLink);
    }
    public void LINKS(string LINK)
    {
        playClickSound();
        Application.OpenURL(LINK);
    }
    #endregion
    #region cash panel methods
    public void GetFreeCashButton() //open cash panel
    {
        playClickSound();
        getFreeCashPanel.SetActive(true);
        //#if !UNITY_EDITOR
        //            if(Ads_Manager.Instance)
        //                    Ads_Manager.Instance.ShowSmallAdmobBanner();
        //#endif
    }
    public void CloseButtonGetFreeCash() //close cash panel
    {
        playClickSound();
        getFreeCashPanel.SetActive(false);

    }
    #endregion
    #region traffic mode selection methods
    void initializeTrafficMode()
    {
        //bestScoreOneWay.text = PlayerPrefs.GetInt("bestScoreOneWay").ToString();
        //bestScoreTwoWay.text = PlayerPrefs.GetInt("bestScoreTwoWay").ToString();
        //bestScoreTimeTrail.text = PlayerPrefs.GetInt("bestScoreTimeAttack").ToString();
    }
    public void selectFreeTrafficMode(int index) //one way:0, two way:1, time trail:2
    {
        PlayerPrefs.SetInt("SelectedModeIndex", index);
        //topBar.SetActive(false);
        //descriptionBox.SetActive(true);
        //LevelNextButton.gameObject.SetActive(true);
        Application.LoadLevel("LoadingScreen");  //8
                                    //#if !UNITY_EDITOR
                                    //      if (index == 0)
                                    //      {
                                    //      GameAnalytics.instance.CustomEventAnalytics("One
Way",1);
                                    //       FBAManager.Instance.CoustomEventFAB("One Way", 1);
                                    //      }else
                                    //      {
                                    //      GameAnalytics.instance.CustomEventAnalytics("Two
```

```
Way",2);
                                    //        FBAManager.Instance.CoustomEventFAB("Two Way", 2);
                                    //      }

    //          if(Ads_Manager.Instance)
    //                  Ads_Manager.Instance.HideSmallAdmobBanner ();
    //#endif

    //print("Trrafic Mode " + index);
  }
  public void RaceBtn()
  {

    Invoke("LoadScene", 0.5f);
  }
  void LoadScene()
  {
    Application.LoadLevel("LoadingScreen");  //8
  }
  //    public void selectLevelTrafficMode(int index) //one way:0, two way:1
  //    {
  //          PlayerPrefs.SetInt ("SelectedModeIndex", index);

  //          Application.LoadLevel ("LoadingScreen"); //8
  //#if !UNITY_EDITOR
  //          if(Ads_Manager.Instance)
  //                  Ads_Manager.Instance.HideSmallAdmobBanner ();
  //#endif
  //    }
  public void BackFromLevelTraffic()
  {
    playClickSound();

    envSelectionPanel.SetActive(true);
    levelModeSelection.SetActive(false);
    //#if !UNITY_EDITOR
    //          if (Ads_Manager.Instance) {
    //                  Ads_Manager.Instance.HideSmallAdmobBanner ();
    //          }
    //#endif

  }
  public void BackFromTrafficeFree()
  {
    playClickSound();
    if (descriptionBox.activeSelf)
    {
      descriptionBox.SetActive(false);
      LevelNextButton.gameObject.SetActive(false);
```

```
    }
    else
    {
        envSelectionPanel.SetActive(true);
        trafficModeSelection.SetActive(false);
        envAdFlag = false;
    }
}
#endregion
#region purchase/shop panel methods
public void PurchaseCashButton()
{
    playClickSound();
    purchaseCashSprite.SetActive(true);
    //#if !UNITY_EDITOR
    //            if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.ShowSmallAdmobBanner();
    //#endif
}
public void PurchaseCashBackButton()
{
    playClickSound();
    purchaseCashSprite.SetActive(false);
    //#if !UNITY_EDITOR
    //                    if(Ads_Manager.Instance)
    //                            Ads_Manager.Instance.HideSmallAdmobBanner ();
    //#endif
}
#endregion
#region garage methods

void initializeGarage()
{
    if (PlayerPrefs.HasKey("LASTSELECTEDCAR"))
    { //active current car
        selectedCar = PlayerPrefs.GetInt("LASTSELECTEDCAR");
        carcounter = selectedCar;
    }
    else
    { //by default first car
        selectedCar = 0;
        carcounter = 0;
    }

    PlayerPrefs.SetInt("CAR0", 1); //first car always unlocked
}
public void GarageButton() //open garage
{
    playClickSound();
```

```
        if (!mainPanel.activeInHierarchy)
        {
            bool isShow = isUnlockCarAvailable();
            if (isShow)
                Invoke("openUnlockCarPanel", 2f);
        }
        mainPanel.SetActive(false);
        carsSelectionPanel.SetActive(true);
        maincamera.SetActive(false);
        cars[carcounter].SetActive(true);
        carSelectionCamera.SetActive(true);
        currentCarStatus();
        checkUnlockCarStatus();

CarUpgradation.instance.setCurrentUpgradateUI(cars[carcounter].GetComponent<CarStats>().data,
carcounter);
        PlayerPrefs.SetInt("unlockcar", 1);
        changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
        print("Open Grage");
    }
    public void GarageBacktoMainButton() //close garage
    {
        playClickSound();
        mainPanel.SetActive(true);
        carsSelectionPanel.SetActive(false);
        maincamera.SetActive(true);
        carSelectionCamera.SetActive(false);
    }
    public void SelectCarButton() //select current car
    {
        playClickSound();

        carsSelectionPanel.SetActive(false);
        carSelectionCamera.SetActive(false);
        maincamera.SetActive(true);
        CancelInvoke("openUnlockCarPanel");
        selectedCar = carcounter;
        PlayerPrefs.SetInt("LASTSELECTEDCAR", selectedCar);
        modeAdFlag = false;
        //                  currentCarStatus ();
        Invoke("modeAds", 5);
    }
    public void NextCarButton() //show next car
    {
        playClickSound();
        colorPanel.SetActive(false);
        upgradePanel.SetActive(false);
        cannotApplyColor.SetActive(false);
        OrbitCam.ins.OnChange();
```

```
    nextBtn.GetComponent<Button>().enabled = false;
    PrevBtn.GetComponent<Button>().enabled = false;

    Invoke("OnSecond", 0.1f);

    if (carcounter == cars.Length - 1)
        carcounter = 0;
    else
        carcounter++;
    MoveCars(carcounter);
    currentCarStatus();
    if (PlayerPrefs.GetInt("unlockcar") == 1)
    {
        //unlockCarPanel.SetActive(true);
        _adInterstatial.ShowAd();
        PlayerPrefs.SetInt("unlockcar", 0);
    }
    changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
}
void delyCarActive()
{
    for (int i = 0; i < cars.Length; i++)
    {
        cars[i].SetActive(false);
    }
    cars[carcounter].SetActive(true);
}
public void PreviousCarButton() //show previous car
{
    playClickSound();
    colorPanel.SetActive(false);
    upgradePanel.SetActive(false);
    cannotApplyColor.SetActive(false);

    OrbitCam.ins.OnChangeValue();
    nextBtn.GetComponent<Button>().enabled = false;
    PrevBtn.GetComponent<Button>().enabled = false;
    //  CarUpgradation.instance.check();
    Invoke("OnSecondRotate", 0.1f);


    if (carcounter == 0)
        carcounter = cars.Length - 1;
    else
    {
        carcounter--;

    }
    MoveCars(carcounter);
```

```
        currentCarStatus();
        if (PlayerPrefs.GetInt("unlockcar") == 1)
        {
            //unlockCarPanel.SetActive(true);
            _adInterstatial.ShowAd();
            PlayerPrefs.SetInt("unlockcar", 0);
        }
        changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
    }
    private void MoveCars(int target)
    {
        _allCars.transform.localPosition = new Vector3(target, _allCars.transform.localPosition.y,
_allCars.transform.localPosition.z);
    }
    string getCarName()
    {
        string[] carName = new string[] { "MINICA DF", "ROVER MX12", "Lamborghini Glardo",
"Chevrolet Camaro", "Bugati Veyron" };
        return carName[carcounter];
    }
    int getCarPrice(int index)
    {
        int[] carPrice = new int[] { 8000, 12000, 20000, 30000, 45000 };
        return carPrice[index];
    }
    public void UnlockCar()
    {
        int carPrice = getCarPrice(carcounter - 1);
        if (PlayerPrefs.GetInt("Currency") >= carPrice)
        {
            PlayerPrefs.SetInt("Currency", PlayerPrefs.GetInt("Currency") - carPrice);
            PlayerPrefs.SetInt("CAR" + carcounter, 1);
            checkCarAchi();
            currentCarStatus();
        }
        else
        {
            purchaseCashSprite.SetActive(true);
        }
    }
    void currentCarStatus()
    {
        carNameLabel.text = getCarName();
        if (carcounter == 0)
        {
            carPriceLabel.text = "";
            unlockButton.SetActive(false);
            upgradeBtn.SetActive(true);
            paintBtn.SetActive(true);
```

```
            selectBtn.SetActive(true);
        }
        else
        {
            if (PlayerPrefs.GetInt("CAR" + carcounter) != 1)
            { //locked car
                carPriceLabel.text = "BUY FOR $" + getCarPrice(carcounter - 1).ToString();
                unlockButton.SetActive(true);
                upgradeBtn.SetActive(false);
                paintBtn.SetActive(false);
                selectBtn.SetActive(false);
            }
            else
            { //unlocked car
                unlockButton.SetActive(false);
                carPriceLabel.text = "";
                upgradeBtn.SetActive(true);
                paintBtn.SetActive(true);
                selectBtn.SetActive(true);
            }
        }
        if (cars[carcounter].GetComponent<CarStats>().data.canChangeColor)
            CarUpgradation.instance.changeCarColor(0,
cars[carcounter].GetComponent<CarStats>().data);


CarUpgradation.instance.setCurrentUpgradateUI(cars[carcounter].GetComponent<CarStats>().data,
carcounter);
    }



    public void UpgradeCarButton()
    {
        playClickSound();
        upgradePanel.SetActive(true);
        upgraderAnim.GetComponent<DOTweenAnimation>().DORestart();
        cannotApplyColor.SetActive(false);
    }

    public void closeUpgradeCarButton()
    {
        playClickSound();
        upgradePanel.SetActive(false);
    }
    public void paintCarButton()
    {
        playClickSound();
        if (cars[carcounter].GetComponent<CarStats>().data.canChangeColor)
```

```
        {

CarUpgradation.instance.setCurrentCarColorUI(cars[carcounter].GetComponent<CarStats>().data);
            colorPanel.SetActive(true);
            colorAnim.GetComponent<DOTweenAnimation>().DORestart();
            cannotApplyColor.SetActive(false);
        }
        else
        {
            StartCoroutine(noColorWarning());
        }
        changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
        CarUpgradation.instance.check();
    }
    public void closePaintPanel()
    {
        playClickSound();
        colorPanel.SetActive(false);
    }
    int ColorIndx = 0;
    public void changeCarColor(int index)
    {
        playClickSound();
        CarUpgradation.instance.ClickColor(index,
cars[carcounter].GetComponent<CarStats>().data);
        ColorIndx = index;
    }
    public void ClickBuyCarColor()
    {
        playClickSound();
        CarUpgradation.instance.purchaseColor(ColorIndx,
cars[carcounter].GetComponent<CarStats>().data, carcounter);
    }
    public void changeCarColor(int index, int caridex)
    {
        CarUpgradation.instance.purchaseColor(index, cars[caridex].GetComponent<CarStats>().data,
carcounter);
    }
    IEnumerator noColorWarning()
    {
        cannotApplyColor.SetActive(true);
        yield return new WaitForSeconds(2f);
        cannotApplyColor.SetActive(false);
    }

    public void turbineUpgradation()
    {

CarUpgradation.instance.turbineUpgradation(cars[carcounter].GetComponent<CarStats>().data);
```

```
    }
    public void pistonUpgradation()
    {

CarUpgradation.instance.pistonsUpgradation(cars[carcounter].GetComponent<CarStats>().data);
    }
    public void tuningUpgradation()
    {

CarUpgradation.instance.tuningUpgradation(cars[carcounter].GetComponent<CarStats>().data);
    }
    public void gearboxUpgradation()
    {

CarUpgradation.instance.gearboxUpgradation(cars[carcounter].GetComponent<CarStats>().data);
    }
    public void shockUpgradation()
    {

CarUpgradation.instance.shocksUpgradation(cars[carcounter].GetComponent<CarStats>().data);
    }
    public void wheelUpgradation()
    {

CarUpgradation.instance.wheelsUpgradation(cars[carcounter].GetComponent<CarStats>().data);
    }
    #endregion
    #region setting panel methods
    void initializeSetting()
    {
        if (PlayerPrefs.GetInt("FIRSTTIME") != 1)
        { //joining reward
            AudioListener.volume = 1;
            soundBar = 1;
            PlayerPrefs.SetFloat("SOUND", 1f);
        }
        else
        {
            AudioListener.volume = PlayerPrefs.GetFloat("SOUND");
            soundBar = PlayerPrefs.GetFloat("SOUND");
        }
    }
    public void SettingsButton()
    {
        playClickSound();
        settingsSprite.SetActive(true);
        settingAdFlag = false;
        Invoke("settingQuitAds", 4);
```

```
    //#if !UNITY_EDITOR
    //          if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.ShowSmallAdmobBanner();
    //#endif
}
public void SettingsBackButton()
{
    playClickSound();
    CancelInvoke("settingQuitAds");
    settingsSprite.SetActive(false);
    //#if !UNITY_EDITOR
    //          if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.HideSmallAdmobBanner();
    //#endif
}
public GameObject Sound_On, Sound_off;
public void ValueChangedSlider(int VALUE)
{
    soundBar = VALUE;
    PlayerPrefs.SetFloat("SOUND", VALUE);
    AudioListener.volume = VALUE;

    if (VALUE == 1)
    {
        Sound_off.GetComponent<Image>().color = new Color(1f, 1f, 1f, 0f);
        //Sound_off.transform.GetChild(0).GetComponent<Text>().color = Color.white;
        //Sound_On.transform.GetChild(0).GetComponent<Text>().color = Color.black;
        Sound_On.GetComponent<Image>().color = new Color(1f, 1f, 1f, 1f);
    }
    if (VALUE == 0)
    {
        Sound_off.GetComponent<Image>().color = new Color(1f, 1f, 1f, 1f);
        Sound_On.GetComponent<Image>().color = new Color(1f, 1f, 1f, 0f);
        //Sound_off.transform.GetChild(0).GetComponent<Text>().color = Color.black;
        //Sound_On.transform.GetChild(0).GetComponent<Text>().color = Color.white;
    }
}
#endregion
#region quit panel methods
public void QuitGameYesButton()
{
    playClickSound();
    Application.Quit();
}
public void QuitGameNoButton()
{
    playClickSound();
    mainPanel.SetActive(true);
    CancelInvoke("settingQuitAds");
```

```
    quitgamePanel.SetActive(false);
    //#if !UNITY_EDITOR
    //          if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.HideSmallAdmobBanner();
    //#endif
}

public void QuitButton()
{
    playClickSound();
    mainPanel.SetActive(false);
    quitgamePanel.SetActive(true);
    //      quitAnimPanel.GetComponent<DOTweenAnimation>().DORestart();
    settingAdFlag = false;
    Invoke("settingQuitAds", 4);
    //#if !UNITY_EDITOR
    //          if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.ShowSmallAdmobBanner();
    //#endif
}
#endregion
#region environment selection panel
public void SelectFreeMode() //open environment selection
{
    playClickSound();
    LevelManager.levelNumber = 0;
    LevelChallange._level = 0;
    PlayerPrefs.SetInt("RespawnExtra", 1);
    mainPanel.SetActive(false);
    highwaySelection.SetActive(true);
    RemoveHighwayLocks();
    //#if !UNITY_EDITOR
    //          if(Ads_Manager.Instance)
    //                    Ads_Manager.Instance.HideSmallAdmobBanner();
    //#endif
}
public void backFromLongDrive() //back from environment selection
{
    highwaySelection.SetActive(false);
    mainPanel.SetActive(true);
}
public void backFromEnvironment() //back from environment selection
{
    playClickSound();

    print(" other levels");
    highwaySelection.SetActive(false);
    envSelectionPanel.SetActive(false);
    if (PlayerPrefs.GetInt("EnvetToEnv") == 2)
```

```
    {
        RewardgesSelection.gameObject.SetActive(true);
        challangesSelection.SetActive(false);
    }
    else
    {
        RewardgesSelection.gameObject.SetActive(false);
        challangesSelection.SetActive(true);
    }
    modeAdFlag = false;
    Invoke("modeAds", 5);
    //#if !UNITY_EDITOR
    //                  if (Ads_Manager.Instance)
    //                          Ads_Manager.Instance.ShowSmallAdmobBanner ();
    //#endif

}
public void backFromHighway()
{
    playClickSound();
    highwaySelection.SetActive(false);
    envSelectionPanel.SetActive(false);
    if (PlayerPrefs.GetInt("EnvetToEnv") == 2)
    {
        RewardgesSelection.gameObject.SetActive(true);
        challangesSelection.SetActive(false);
    }
    else
    {
        RewardgesSelection.gameObject.SetActive(false);
        challangesSelection.SetActive(true);
    }
    //mainPanel.SetActive(true);
    modeAdFlag = false;
    Invoke("modeAds", 5);
}
public void BackToMainEnvSelection()
{
    playClickSound();
    mainPanel.SetActive(true);
    envSelectionPanel.SetActive(false);
}
public void selectEnvironment(int index)
{
    //playClickSound ();


    LoadingScript.leveltoload = index;
    if (LevelManager.levelNumber == 10)
```

```
{
   onewayImage.GetComponent<Button>().interactable = false;
}
else
{
   onewayImage.GetComponent<Button>().interactable = true;
}

if (LevelManager.levelNumber == 2)
   _selectTwoWays.interactable = false;
if (LevelManager.levelNumber == 13)
{
   _selectOneWay.interactable = false;
   _selectTwoWays.interactable = true;
}

if (index != 0)

{ //ist env unlocked by default

   if (PlayerPrefs.GetInt("ENV" + index) != 1)
   { //locked environment
      if (PlayerPrefs.GetInt("Currency") >= 500)
      { //purchase environment

         PlayerPrefs.SetInt("Currency", PlayerPrefs.GetInt("Currency") - 500);
         PlayerPrefs.SetInt("ENV" + index, 1);
         RemoveEnvLocks(); //unlock purchased environment
         if (!PlayerPrefs.HasKey("firstEnvBought"))
         {
            PlayerPrefs.SetInt("firstEnvBought", 1);
            LeaderboardManager.currentAchiement = 17;
            leader.revealAcheivements();
         }
      }
      else
      { //not enough coins: open cash panel
         PurchaseCashButton();
      }
   }

   else
   { //environment already unlocked
      envSelectionPanel.SetActive(false);
      if (PlayerPrefs.GetInt("FristPlay") == 1)
      {
         trafficModeSelection.SetActive(true);
         //  print("here");
      }
```

```
            //#if !UNITY_EDITOR
            //                if (Ads_Manager.Instance)
            //
        Ads_Manager.Instance.ShowSmallAdmobBanner ();
            //#endif
        }
    }
    else
    { //snow: bydefault unlocked
        envSelectionPanel.SetActive(false);
        if (PlayerPrefs.GetInt("FristPlay") == 1)
        {
            trafficModeSelection.SetActive(true);
            //  print("here0");
        }



        //#if !UNITY_EDITOR
        //                                  if (Ads_Manager.Instance)
        //                                          Ads_Manager.Instance.ShowSmallAdmobBanner ();
        //#endif
    }
    if (index == 0 || index == 2)
    {
        day = true;
    }
    else
        day = false;

    //   print("Envirement" + index);
    //#if !UNITY_EDITOR
    //        GameAnalytics.instance.CustomEventAnalytics("Envirement", index);
    //         FBAManager.Instance.CoustomEventFAB("Envirement", index);
    //#endif
    PlayerPrefs.SetInt("FristPlay", 1);
}
public void selectHighway(int index)
{
    playClickSound();
    LoadingScript.leveltoload = index;
    if (index != 0)
    { //ist env unlocked by default
        if (PlayerPrefs.GetInt("ENV" + index) != 1)
```

```
        { //locked environment
            if (PlayerPrefs.GetInt("Currency") >= 500)
            { //purchase environment
                PlayerPrefs.SetInt("Currency", PlayerPrefs.GetInt("Currency") - 500);
                PlayerPrefs.SetInt("ENV" + index, 1);
                RemoveHighwayLocks(); //unlock purchased environment
                if (!PlayerPrefs.HasKey("firstEnvBought"))
                {
                    PlayerPrefs.SetInt("firstEnvBought", 1);
                    //                                          LeaderboardManager.ach42 = true;
                    LeaderboardManager.currentAchiement = 17;
                    leader.revealAcheivements();
                }
            }
            else
            { //not enough coins: open cash panel
                PurchaseCashButton();
            }
        }
        else
        { //environment already unlocked
            highwaySelection.SetActive(false);
            trafficModeSelection.SetActive(true);
            //#if !UNITY_EDITOR
            //                      if(Ads_Manager.Instance)
            //                              Ads_Manager.Instance.ShowSmallAdmobBanner();
            //#endif
        }
    }
    else
    { //snow: bydefault unlocked
        highwaySelection.SetActive(false);
        trafficModeSelection.SetActive(true);
        //#if !UNITY_EDITOR
        //                  if(Ads_Manager.Instance)
        //                          Ads_Manager.Instance.ShowSmallAdmobBanner();
        //#endif
    }
    if (index == 0 || index == 2)
    {
        day = true;
    }
    else
        day = false;
    HR_GamePlayHandler.highwayNumber = index + 1;
}
public void RemoveEnvLocks() //environments locked status
{
    if (PlayerPrefs.GetInt("ENV1") == 1)
```

```
            envLocks[0].SetActive(false);
        if (PlayerPrefs.GetInt("ENV2") == 1)
            envLocks[1].SetActive(false);
        if (PlayerPrefs.GetInt("ENV3") == 1)
            envLocks[2].SetActive(false);

    }

    public void RemoveHighwayLocks() //highway locked status
    {
        highwayUnlockText[0].text = PlayerPrefs.GetFloat("HighWay1Stats").ToString("F2") + "/" +
_description.highwayDistance[0] + "KM";
        if (PlayerPrefs.GetInt("ENV1") == 1 || PlayerPrefs.GetFloat("HighWay1Stats") >=
_description.highwayDistance[0])
        {
            highWayLocks[0].SetActive(false);
            highwayUnlock[0].SetActive(true);
            highwayUnlockText[1].text = PlayerPrefs.GetFloat("HighWay2Stats").ToString("F2") + "/"
+ _description.highwayDistance[1] + "KM";
        }
        else
        {
            highWayLocks[0].SetActive(true);
            highwayUnlock[0].SetActive(false);
        }
        if (PlayerPrefs.GetInt("ENV2") == 1 || PlayerPrefs.GetFloat("HighWay2Stats") >=
_description.highwayDistance[1])
        {
            highWayLocks[1].SetActive(false);
            highwayUnlock[1].SetActive(true);
            highwayUnlockText[2].text = PlayerPrefs.GetFloat("HighWay3Stats").ToString("F2") + "/"
+ _description.highwayDistance[2] + "KM";
        }
        else
        {
            highWayLocks[1].SetActive(true);
            highwayUnlock[1].SetActive(false);
        }
        if (PlayerPrefs.GetInt("ENV3") == 1 || PlayerPrefs.GetFloat("HighWay3Stats") >=
_description.highwayDistance[2])
        {
            highWayLocks[2].SetActive(false);
            highwayUnlock[2].SetActive(true);
            highwayUnlockText[3].text = PlayerPrefs.GetFloat("HighWay4Stats").ToString("F2") + "/"
+ _description.highwayDistance[3] + "KM";
        }
        else
        {
            highWayLocks[2].SetActive(true);
```

```
        highwayUnlock[2].SetActive(false);
    }

}
public void GameModeSelectionBackButton() //back from mode selection
{
    playClickSound();
}
#endregion
#region Level selection methods
void initializeLevelPanel()
{
    if (check)
    { //open level selection panel
        mainPanel.SetActive(false);
        challangesSelection.SetActive(true);
        check = false;
    }
}
string getLevelDescription(int index)
{
    return _description.levelsDescription[index];
}
string getEventDescription(int index)
{
    return DailyEventDes.levelsDescription[index];
}

public void SelectLevelsMode() //open levels selection panel
{

    if (PlayerPrefs.GetInt("FristPlay", 0) == 1)
    {
        playClickSound();
        challangesSelection.SetActive(true);
        mainPanel.SetActive(false);
    }
    else
    {
        LoadingSound.Play();
        trafficModeSelection.SetActive(false);
        SelectLevel(0);
        selectEnvironment(0);
        PlayerPrefs.SetInt("SelectedModeIndex", 0);

        LoadScene();
    }
    //    changeCarColor(PlayerPrefs.GetInt("colorIdex"), PlayerPrefs.GetInt("CarIdex"));
    //    print("Open Campaign (Levels)");
```

```
        //#if !UNITY_EDITOR
        //        GameAnalytics.instance.CustomEventAnalytics("Open Campaign (Levels)");
        //        FBAManager.Instance.CoustomEventFAB("Open Campaign (Levels)");
        //#endif
        //#if !UNITY_EDITOR
        //        if (Ads_Manager.Instance) {
        //                        Ads_Manager.Instance.ShowSmallAdmobBanner ();
        //            }
        //#endif
    }
    public void SelectRewardMode() //open levels selection panel RewardgesSelection
    {
        playClickSound();
        RewardgesSelection.SetActive(true);
        descriptionBox.SetActive(false);
        RewardLevelNextButton.gameObject.SetActive(false);
        //    print("Open Daily Event ");
#if !UNITY_EDITOR
  //      GameAnalytics.instance.CustomEventAnalytics("Open Daily Event");
  //       FBAManager.Instance.CoustomEventFAB("Open Daily Event");
#endif
        //#if !UNITY_EDITOR
        //        if (Ads_Manager.Instance)
        //        {
        //            Ads_Manager.Instance.ShowSmallAdmobBanner();
        //        }
        //#endif
    }
    public void RewarModeBackButton() //back from levels panel
    {
        playClickSound();
        RewardgesSelection.SetActive(false);
        mainPanel.SetActive(true);
        //#if !UNITY_EDITOR
        //        if (Ads_Manager.Instance)
        //        {
        //            Ads_Manager.Instance.HideSmallAdmobBanner();
        //        }
        //#endif
        modeAdFlag = false;
        Invoke("modeAds", 5);

    }
    public void LevelsModeBackButton() //back from levels panel
    {
        playClickSound();


        challangesSelection.SetActive(false);
```

```
    RewardgesSelection.SetActive(false);
    mainPanel.SetActive(true);
    //#if !UNITY_EDITOR
    //            if (Ads_Manager.Instance) {
    //                        Ads_Manager.Instance.HideSmallAdmobBanner ();
    //            }
    //#endif
    modeAdFlag = false;
    Invoke("modeAds", 5);

}
public void LevelNext()
{
    playClickSound();
    RewardgesSelection.SetActive(false);
    challangesSelection.SetActive(false);
    envSelectionPanel.SetActive(true);
    RemoveEnvLocks();
    PlayerPrefs.SetInt("RespawnExtra", 2);
}
public void CheckLevelLocks() //activate unlocked levels
{
    for (int i = 0; i < LevelLocks.Length; i++)
    {
        if (PlayerPrefs.GetInt("LevelCompleted" + i) == 1)
        { //unlcoked
            LevelLocks[i].SetActive(false);
            levelParent[i].interactable = true;
        }
        else
        { //locked
            LevelLocks[i].SetActive(true);
            levelParent[i].interactable = false;
        }
    }


}
public void CheckRewardLocks() //activate unlocked levels
{
    int RewadHours = 0;
    int Hours = System.DateTime.Now.Hour; //current date
    int Days = System.DateTime.Now.Day;
    if (!PlayerPrefs.HasKey("StarttHour"))
    {
        PlayerPrefs.SetInt("StarttHour", Hours);
        PlayerPrefs.SetInt("StartDay", Days);
    }
    if (PlayerPrefs.GetInt("StartDay") == Days)
```

```
        {
            PlayerPrefs.SetInt("CurrentSpintTime", PlayerPrefs.GetInt("CurrentSpintTime") + (Hours -
PlayerPrefs.GetInt("StarttHour")));
        }
        else
        {
            int MissDays = Days - PlayerPrefs.GetInt("StartDay");
            if (MissDays == 1)
            {
                int spintHours = Hours + 24 - PlayerPrefs.GetInt("StarttHour");
                PlayerPrefs.SetInt("SpintTime", PlayerPrefs.GetInt("SpintTime") + spintHours);
                PlayerPrefs.SetInt("CurrentSpintTime", PlayerPrefs.GetInt("SpintTime"));
                PlayerPrefs.SetInt("StarttHour", Hours);

                PlayerPrefs.SetInt("StartDay", Days);

            }
            else
            {
                int spintHours = Hours + MissDays * 24;
                PlayerPrefs.SetInt("SpintTime", PlayerPrefs.GetInt("SpintTime") + spintHours);
                PlayerPrefs.SetInt("CurrentSpintTime", PlayerPrefs.GetInt("SpintTime"));
                PlayerPrefs.SetInt("StarttHour", Hours);
                PlayerPrefs.SetInt("StartDay", Days);
            }

        }

        Debug.Log("Today Tart Time == " + PlayerPrefs.GetInt("StarttHour") + "Time SpintTotal = "
+ PlayerPrefs.GetInt("CurrentSpintTime"));
        if (12 >= PlayerPrefs.GetInt("CurrentSpintTime"))
        {
            RewadHours = 0;
            PlayerPrefs.SetInt("RemainingTime", 11);
        }
        else if (24 >= PlayerPrefs.GetInt("CurrentSpintTime"))
        {
            RewadHours = 1;
            PlayerPrefs.SetInt("RemainingTime", 24);
        }
        else if (36 >= PlayerPrefs.GetInt("CurrentSpintTime"))
        {
            RewadHours = 2; PlayerPrefs.SetInt("RemainingTime", 36);
        }
        else if (48 >= PlayerPrefs.GetInt("CurrentSpintTime"))
        {
            RewadHours = 3; PlayerPrefs.SetInt("RemainingTime", 48);
        }
        else if (60 >= PlayerPrefs.GetInt("CurrentSpintTime"))
```

```
        {
            RewadHours = 4;
            PlayerPrefs.DeleteKey("SpintTime");
            PlayerPrefs.DeleteKey("CurrentSpintTime");
        }
        print("remaining time = " + (PlayerPrefs.GetInt("CurrentSpintTime") -
PlayerPrefs.GetInt("RemainingTime")));
        RewminigTimeReward.text = (PlayerPrefs.GetInt("CurrentSpintTime") -
PlayerPrefs.GetInt("RemainingTime")) + ":0:20".ToString();
        for (int i = 0; i < RewardLevelLocks.Length; i++)
        {
            if (RewadHours == i)
            { //unlcoked
                RewardLevelLocks[i].SetActive(false);
                currentevent = i;
                int index = i + 1;
                if (PlayerPrefs.GetInt("currentEvnt" + index) == 1)
                    DailyEventTime.SetActive(false);
                else
                    DailyEventTime.SetActive(true);
            }
            else
            { //locked

                RewardLevelLocks[i].SetActive(true);
                RewardlevelParent[i].GetComponent<Button>().interactable = true;

            }
        }
    }

    public void SelectRewardLevel(int i)
    {
        playClickSound();
        print(i + " =i = reward= " + currentevent);
        if (i == currentevent)
        {
            PlayerPrefs.SetInt("currentEvnt" + i, 1);
            // descriptionBox.SetActive(true);
            LevelsDescText.text = getEventDescription(i);
            LevelManager.levelNumber = 0;
            LevelChallange._level = 0;
            LevelChallange.DailyEvent = i + 1;
            LevelNext();
            PlayerPrefs.SetInt("EnvetToEnv", 2);
            //RewardLevelNextButton.gameObject.SetActive(true);
            HR_GamePlayHandler.highwayNumber = 0;
        }
    }
```

```
public void SelectLevel(int i)
{
    playClickSound();
    //descriptionBox.SetActive (true);
    LevelsDescText.text = getLevelDescription(i);
    //if (i <= PlayerPrefs.GetInt ("LevelCompleted") + 1)
    if (PlayerPrefs.GetInt("LevelCompleted" + i) == 1)
    {
        LevelChallange.DailyEvent = 0;
        LevelManager.levelNumber = i + 1;
        LevelChallange._level = i + 1;
        LevelNextButton.gameObject.SetActive(true);

    }
    else
    {
        LevelNextButton.gameObject.SetActive(false);
    }
    HR_GamePlayHandler.highwayNumber = 0;
    PlayerPrefs.SetInt("EnvetToEnv", 1);
    LevelNext();
}
#endregion
#region in-house ads
public GameObject[] envAd;
bool settingAdFlag = false, modeAdFlag = false, envAdFlag = false;
void settingQuitAds()
{
    if (settingsSprite.activeInHierarchy)
    {
        if (settingAdFlag)
        {
            settingAdFlag = false;

        }
        else
        {
            settingAdFlag = true;
        }
        Invoke("settingQuitAds", 4);
    }
    else if (quitgamePanel.activeInHierarchy)
    {
        if (settingAdFlag)
        {
            settingAdFlag = false;
        }
        else
        {
```

```
            settingAdFlag = true;
        }
        Invoke("settingQuitAds", 4);
    }
}
void modeAds()
{
    if (mainPanel.activeInHierarchy)
    {
        if (modeAdFlag)
        {
            modeAdFlag = false;
        }
        else
        {
            modeAdFlag = true;

        }
        Invoke("modeAds", 5);
    }
}
void environmentAd()
{
    if (envSelectionPanel.activeInHierarchy)
    {
        if (envAdFlag)
        {
            envAdFlag = false;
            envAd[0].SetActive(true);
            envAd[1].SetActive(false);
        }
        else
        {
            envAdFlag = true;
            envAd[0].SetActive(false);
            envAd[1].SetActive(true);
        }
        Invoke("environmentAd", 5);
    }
}
public void pradoAd()
{
    playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.doit.fun.games.offroad.r
ally.truck.apps&hl=en");
}
public void ImpossibleTracksAd()
{
```

```
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.impossible.tracks.real.race.games&referrer=utm_source=modern_car");
    }
    public void waterStuntAd()
    {
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.zact.water.park.stuntman.game.free&referrer=utm_source=modern_car");
    }
    public void mainADDClick()
    {
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.den.crazy.speed.curve.traffic.race&referrer=utm_source=modern_car");
    }
    public void mainADDClick2()
    {
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.zact.bicycle.endless.rider.game.free&referrer=utm_source=modern_car");
    }
    public void Ad1()
    {
        playClickSound();

Application.OpenURL("https://play.google.com/store/apps/details?id=com.door.modern.car.parking.games&referrer=utm_source=modern_car");
    }
    #endregion
    #region set inapp buttons
    public void unlockCarIap()
    {
        unlockCarBtn.SetActive(false);
        currentCarStatus();
    }
    bool isUnlockCarAvailable()
    {
        for (int i = 1; i <= cars.Length; i++)
        { //show unlock car btn if any car is unlocked
            if (PlayerPrefs.GetInt("CAR" + i) != 1)
            {
                return true;
            }
        }
```

```
        return false;
    }
    void checkUnlockCarStatus() //inapp button
    {
        for (int i = 1; i <= cars.Length; i++)
        { //show unlock car btn if any car is unlocked
            if (PlayerPrefs.GetInt("CAR" + i) != 1)
            {
                return;
            }
        }
        unlockCarBtn.SetActive(false);

    }
    public void setIAPButtons()
    {
        if (PlayerPrefs.GetInt("UNLOCKFULLGAME").Equals(1))
            fullUnlockButon.interactable = false;
        //if (PlayerPrefs.GetInt("ADSUNLOCK").Equals(1)) revome ad
    }
    #endregion
    #region extrass
    public void CreditsButton()
    {
        Application.LoadLevel(7);
    }



    void checkCarAchi()
    {
        if (!PlayerPrefs.HasKey("firstCarBought"))
        {
            PlayerPrefs.SetInt("firstCarBought", 1);
            PlayerPrefs.SetInt("firstFiveBought", 1);
            //                          LeaderboardManager.ach41 = true;
            LeaderboardManager.currentAchiement = 18;
            leader.revealAcheivements();
        }
        else
        {
            PlayerPrefs.SetInt("firstFiveBought", PlayerPrefs.GetInt("firstFiveBought") + 1);
            if (PlayerPrefs.GetInt("firstFiveBought") == 5 &&
!PlayerPrefs.HasKey("firstFiveClaimed"))
            {
                PlayerPrefs.SetInt("firstFiveClaimed", 1);
                //                                  LeaderboardManager.ach44 = true;
                LeaderboardManager.currentAchiement = 19;
                leader.revealAcheivements();
```

```
        }
    }
}
#endregion
#region IAP offer panel
public GameObject megaPack, unlockCarPanel;
void openMegaPack() //OPEN PANEL
{
    //megaPack.SetActive(true);
    _adInterstatial.ShowAd();
}
public void closeMegaPack() //CLOSE PANEL
{
    playClickSound();
    megaPack.SetActive(false);
}
void openUnlockCarPanel() //open panel
{
    unlockCarPanel.SetActive(true);
}
public void closeUnlockCarPanel() //close panel
{
    playClickSound();
    unlockCarPanel.SetActive(false);
}
public void closePanelsOnReward()
{
    if (megaPack.activeInHierarchy)
    {
        megaPack.SetActive(false);
    }
    if (unlockCarPanel.activeInHierarchy)
    {
        unlockCarPanel.SetActive(false);
    }
}
#endregion
#region sounds
public AudioSource clickSound;
public void playClickSound()
{
    clickSound.Play();
}
#endregion
#region rate panel methods
public GameObject ratePanel, dummyColor, rateBtn, thankuBtn, rateWarning;
public GameObject[] stars;
public void showRatePanel()
{
```

```
        ratePanel.SetActive(true);
        setStarColors(-1);
    }

    public void starClicked(int starNumber) //star button clicked
    {
        setStarColors(starNumber);
    }
    void setStarColors(int starNumber) //set rate panel
    {
        foreach (GameObject star in stars)
        {
            star.GetComponent<Image>().color = dummyColor.GetComponent<Image>().color;
        }
        if (starNumber >= 0)
        { //star button clicked
            for (int i = 0; i <= starNumber; i++)
            {
                stars[i].GetComponent<Image>().color = Color.yellow;
            }
            if (starNumber >= 3)
            {
                rateBtn.SetActive(true);
                thankuBtn.SetActive(false);
            }
            else
            {
                rateBtn.SetActive(false);
                thankuBtn.SetActive(true);
            }
        }
        else
        { //star panel enabled
            rateBtn.SetActive(false);
            thankuBtn.SetActive(false);
        }
    }
    public void rateClicked()
    {
        //if (Application.internetReachability != NetworkReachability.NotReachable) {
        //
Application.OpenURL("https://play.google.com/store/apps/details?id=com.PlayfireRoyale.XRacer
ExtremeCarRacingPro");
        //    PlayerPrefs.SetInt("Rate", 1);
        //    Invoke("closeRatePanel", 0.2f);
        //}
        //else {
        //    rateWarning.SetActive(true);
        //    Invoke("closeRatePanel", 1.5f);
```

```
    //}
    closeRatePanel();
}
public void thankuClicked()
{
    PlayerPrefs.SetInt("Rate", 1);
    closeRatePanel();
}
public void closeRatePanel()
{
    if (rateWarning.activeInHierarchy)
    {
        rateWarning.SetActive(false);
    }
    ratePanel.SetActive(false);
}

#endregion

void OnSecondRotate()
{
    OrbitCam.ins.OnChangeValue();
    Invoke("activeBtns", 0.1f);
}
void OnSecond()
{
    OrbitCam.ins.OnChange();
    Invoke("activeBtns", 0.1f);
}

void activeBtns()
{
    nextBtn.GetComponent<Button>().enabled = true;
    PrevBtn.GetComponent<Button>().enabled = true;
}
public void settingOption(GameObject current)
{
    Optionpanal.SetActive(false);
    CreditsPanal.SetActive(false);
    AboutPanal.SetActive(false);
    UpdatePanal.SetActive(false);
    current.SetActive(true);
    if (UpdatePanal.activeSelf)
        PlayerPrefs.SetInt("SettingNotification", 1);
}

public void InappSetting(GameObject current)
{
    bundle.SetActive(false);
```

```
        Gold.SetActive(false);
        Denerio.SetActive(false);
        freeCash.SetActive(false);
        current.SetActive(true);

    }

}
```