ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**РОЗРОБКА МОБІЛЬНОЇ ГРИ З СЕТІНГОМ В СТИЛІ ПОДОРОЖІ ЗАКАРПАТТЯМ ДЛЯ ANDROID МОВОЮ C# З ВИКОРИСТАННЯМ РУШІЯ UNITY**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій КУЛІШ
(підпис)

Виконав: здобувач вищої освіти групи ППЗ-51

Андрій КУЛІШ

Керівник: Тимур ДОВЖЕНКО
к.т.н, доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кулішу Андрію Андрійовичу

1. Тема кваліфікаційної роботи: «Розробка мобільної гри з сетингом в стилі подорожі закарпаттям для android мовою c# з використанням рушія Unity»
керівник кваліфікаційної роботи к.т.н, доцент кафедри ІПЗ Тимур ДОВЖЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: опис основних етапів створення гри, вибір і налаштування інструментів, розробка механік гри та графічного інтерфейсу, тестування та оптимізація продуктивності коду, підготовка технічної документації

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Теоретична частина.
2. Аналіз та особливості казуальних ігор.
3. Обгрунтування вибору технологій.
4. Розробка гри.
5. Реалізація гри.

6. Тестування гри.

5. Перелік графічного матеріалу: *презентація*

1. Діаграма прецедентів.

2. Схема структури системи.

3. Схема бази даних.

4. Діаграма класів.

5. Блок-схема алгоритму основної механіки гри.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.24	
2	Аналіз та дослідження існуючих аналогів	26.02.24-01.03.24	
3	Аналіз та вибір інструментів для розробки додатку	01.03.24-14.03.24	
4	Проектування додатку	15.03.24-20.03.24	
5	Реалізація додатку	20.03.24-04.04.24	
6	Тестування додатку	04.04.24-15.04.24	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Андрій КУЛІШ
(підпис)

Керівник
кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 63 стор., 2 табл., 20 рис., 10 джерел.

Мета роботи - покращення геймплею мобільної гри жанру гонки за рахунок використання сетінгу в стилі подорожі Закарпаттям для Android мовою C# з використанням рушія Unity.

Об'єкт дослідження - геймплеєм мобільної гри жанру гонки з сетінгом в стилі подорожі.

Предмет дослідження - мобільна гра жанру гонки з сетінгом в стилі подорожі.

Короткий зміст роботи: Було проаналізовано популярні гоночні ігри, визначено ключові механіки, такі як детальна графіка, реалістична фізика, вибір авто, різноманітні траси. Все це враховано у розробці гри. Описано концепт гри з історичними і культурними аспектами Закарпаття, сценаріїми та трасами, що підсилюють унікальність ігрового світу. Сформульовано технічні вимоги: оптимізація для різних мобільних пристроїв, інтуїтивний інтерфейс, системи керування. Реалізовано основні механіки, створено ігрові локації та авто, відображаючи культуру Закарпаття, що забезпечує зручність управління. Завершено розробку і проведено тестування, виявлено та усунуто помилки за допомогою сценаріїв і ручних методів, забезпечено стабільність роботи ігри та високу якість кінцевого продукту.

КЛЮЧОВІ СЛОВА: гра, Unity, C#, Android, iOS, геймплей, Asset, фреймворк, GIT, SQLite, GameDev, 3D, 2D

ЗМІСТ

ВСТУП

Зростання популярності ігор за останні роки ставить перед розробниками завдання забезпечити постійне покращення геймплею для задоволення високих вимог геймерської аудиторії. На фоні конкуренції в індустрії ігор важливо надати гравцям непередбачуваність та віртуальну реалістичність через вдосконалену генерацію ігрових предметів.

У сучасному світі комп'ютерні ігри здатні не лише надати емоційний відпочинок, але й стати засобом розвитку та навчання. Гра спрямована на забезпечення захоплюючого та цікавого геймплею, в якому гравці можуть розвивати своє стратегічне мислення та приймати важливі рішення.

Дипломна робота передбачає вивчення методів розробки відеоігрових програмних продуктів, аналіз науково-технічної літератури з питань, пов'язаних з програмним забезпеченням для розробки відеоігор, а також використання офіційної документації Unity, Microsoft Visual Studio та мови програмування C#. Велика увага приділена розробці логіки гри, реалізації графічного інтерфейсу та тестуванню.

Мета роботи - покращення геймплею мобільної гри жанру гонки за рахунок використання сетінгу в стилі подорожі Закарпаттям для Android мовою C# з використанням рушія Unity.

Для досягнення цієї мети в роботі необхідно вирішити такі завдання:

Проаналізувати мобільні ігри в жанрі гонок та визначити основні особливості та механіки, які реалізуться в них.

Розробити концепт гри, де будуть описані сюжет, ігрові механіки, візуальний стиль, а також ідеї для інтеграції культурних та історичних елементів Закарпаття.

Розробити технічне завдання, що включає вимоги до гри, опис функціональності та графічних ресурсів.

Спроектувати та реалізувати механіки гри, системи квестів, ігрові локації, ігрові персонажі, а також інтерфейс користувача для гри жанру гонок з використанням сетінгу в стилі подорожі Закарпаттям.

Провести тестування гри, виявити та виправити помилки, забезпечити стабільність роботи гри.

Об’єкт дослідження - геймплеєм мобільної гри жанру гонки з сетінгом в стилі подорожі.

Предмет дослідження - мобільна гра жанру гонки з сетінгом в стилі подорожі.

Апробація результатів:

1. Куліш А.А. Розробка застосунку для тестування rest-запитів зовнішніх систем на мові java / Аверічев І.М., Куліш А.А. // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24.04.24 – подано до друку, ДУІКТ

2. Куліш А.А. Розробка мобільної гри для android мовою C# / Аверічев І.М., Куліш А.А. // IV Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15.05.24 – подано до друку, ДУІКТ

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Казуальні ігри та їх роль у сучасному житті

У динамічному світі мобільних ігор розуміння цільової аудиторії та жанру є ключовим для створення ігор, які резонують із гравцями. Проаналізуємо поточні тенденції та демографічні показники мобільних ігор. Розуміння ландшафту: щоб створити гру, яка виділяється, важливо зрозуміти існуючий ландшафт мобільних ігор. Це передбачає вивчення сучасних тенденцій, популярних жанрів і демографічних уподобань. Звіти ігрової індустрії Newzoo та Statista, пропонують цінну інформацію про тенденції.

Демографічне розповсюдження мобільних ігор є широким і охоплює різні вікові групи, стать і географічне розташування. Такі ігри, як «Candy Crush» і «PUBG Mobile», показали, що привабливість мобільних ігор виходить за рамки традицій. До нових тенденцій відносять ігри доповненої реальності (AR), «Pokémon GO», тому зростання популярності казуальних і гіперказуальних ігор змінили ландшафт.

Аналіз тенденцій допомагає зрозуміти, чим зараз займаються гравці. Розглянемо виявлення недостатньо обслуговуваних ніш або недостатньо використаних механізмів.

Аналіз прогалин: шукайте прогалини на ринку. Чи є жанри чи ігрові механізми, які популярні на інших платформах, але недостатньо представлені в мобільних іграх? Наприклад, сюжетні ігри чи певні типи стратегічних ігор можуть бути менш поширеними, але мати потенційну аудиторію.

Інноваційна механіка. Потрібно досліджувати недостатньо використовувані механізми, які можуть привнести новий погляд на мобільні ігри. Такі механізми, як управління на основі жестів або інтерактивне оповідання, можуть запропонувати нові враження для мобільних геймерів.

Культурні та регіональні переваги. Різні регіони мають різні переваги. Наприклад, стратегічні ігри можуть бути більш популярними в деяких частинах Азії, тоді як західна аудиторія може віддавати перевагу насиченим екшном пригодам. Пристосування ігор до цих уподобань може задіяти певні ринки.

Створення персонажа гравця. Визначення ідеального гравця. Створіть детальну персону свого ідеального гравця. Ця особа має включати вік, стать, ігрові вподобання, використання пристрою та психографічні елементи, як-от мотивація для гри.

Мотивації та вподобання. Важливо розуміти, що мотивує ваших гравців. Вони прагнуть до втечі, соціальної взаємодії чи виклику? Узгодження дизайну гри з цими мотивами може значно підвищити її привабливість.

Цикл зворотного зв'язку. Щоб удосконалити свою особистість гравця потрібно використовувать опитування, прослуховування соціальних мереж і бета-тестування відгуків. Безпосереднє спілкування з цільовою аудиторією може дати цінну інформацію про її вподобання та очікування.

Також потрібно розробить привабливу основну концепцію. Створення видатної мобільної гри на сьогоднішньому насиченому ринку потребує переконливої основної концепції. Цей крок передбачає розробку унікальної торгової пропозиції, зосередження на захоплюючому ігровому процесі та прототипування вашої ідеї до досконалості.

Розробка унікальної торгової пропозиції (USP) Ідентифікація вашого USP: Визначте, що відрізняє вашу гру від конкурентів. Це може бути інноваційна ігрова механіка, унікальний візуальний стиль або захоплююча розповідь. Такі ігри, як «Monument Valley», досягли успіху, запропонувавши унікальний візуальний досвід у поєднанні з механікою головоломки.

Використання тенденцій і прогалин. Потрібно використовувати інформацію з аналізу ринку, щоб визначити тенденції та прогалини. Гра, яка поєднує популярні елементи з чимось новим, може привернути увагу аудиторії. Наприклад, інтеграція технології AR у гру-головоломку може бути потенційним USP.

Вирівнювання цільової аудиторії: переконайтеся, що ваш USP резонує з вашою цільовою аудиторією. Якщо ваш гравець віддає перевагу коротким, захоплюючим ігровим сесіям, USP, орієнтований на швидкий і корисний ігровий процес, буде ефективним.

Зосередження на простому, захоплюючому циклі ядра гри. Простота і глибина: основний ігровий цикл має бути простим для розуміння, але забезпечувати глибину та прогрес. Цей баланс утримує гравців зацікавленими з часом. Розглядаючи такі ігри, як «Clash Royale», яка пропонує просту механіку, увагу привертають глибокі стратегічні можливості.

Системи винагороди та прогресу. Потрібно включати систему винагород за прогрес. Це може включати вміст, який можна розблокувати, підвищення рівня гравця або досягнення в грі. Такі системи заохочують продовжувати гру та інвестиції в гру.

Збалансування викликів і доступності. Потрібно переконайтеся, що гра доступна для вашої цільової демографічної групи, але водночас є складною. Цей баланс має вирішальне значення для підтримки інтересу та задоволення гравців. Створення прототипу та ітерація вашої основної концепції.

Швидке створення прототипів: розробка швидких прототипів, щоб перевіряти та вдосконалювати основну ігрову механіку. Цей процес допомагає визначити, що працює, а що ні, на початку циклу розробки.

Збір відгуків. Для повторення гри потрібно використовувати відгуки гравців під час сеансів тестування. Розуміння реакції та вподобань гравців є неоціненним для вдосконалення вашої основної концепції.

Ітеративна розробка. Використання ітеративного підходу до розробки гри означає постійне вдосконалення та покращення гри на основі тестування та відгуків, гарантуючи, що кінцевий продукт буде веселим і привабливим.

Дизайн для мобільної гри. Розробка дизайну для мобільних ігор має вирішальне значення для того, щоб гра не лише охопила широку аудиторію, але й забезпечувала приємний і доступний досвід. Цей крок зосереджений на оптимізації сенсорного керування, інтерфейсу користувача/UX для маленьких

екранів, можливості відтворення сеансу та розгляді функціональності в автономному режимі.

Пріоритет інтуїтивного сенсорного керування. Простота – головне. Розробіть сенсорні елементи керування, які легко зрозуміти та використовувати. Такі ігри, як «Fruit Ninja», вирізняються тим, що використовують просту інтуїтивно зрозумілу та приємну механіку проведення.

Чуйний і точний: переконайтеся, що елементи керування оперативні та точні. Гравці повинні відчувати контроль над грою, і будь-яке відставання або неточність можуть розчарувати.

Інтеграція підручника. Плавно інтегруйте навчальні посібники в ігровий процес, щоб навчити керувати, не перевантажуючи гравця. Поступове знайомство з механікою може покращити процес навчання. UI/UX, оптимізований для маленьких екранів.

Чіткий і лаконічний інтерфейс користувача. Користувальницький інтерфейс має бути зрозумілим і незавантаженим, а важлива інформація має бути легкодоступною. Потрібно продумати про розміщення кнопок і інформації, щоб вони були легко доступними та читабельними.

Адаптивний дизайн. Розробляйте елементи інтерфейсу користувача, які адаптуються до різних розмірів і орієнтацій екрана. Це забезпечує стабільну роботу на різних пристроях.

Тестування користувача. Для отримання відгуків про UI/UX потрібно проводити сеанси тестування користувачів. Цей відгук є безцінним для внесення ітераційних покращень у дизайн.

Забезпечення відтворення короткої сесії. Швидкі та задовільні заняття: Створюйте короткі, але корисні ігрові сесії. Такі ігри, як «Clash of Clans», пропонують швидкі ігрові сесії з чіткими цілями та винагородами, що ідеально підходить для мобільних ігор.

Мікропрогресії: потрібно впровадити систему мікропрогресій, яка винагороджує гравців навіть за короткі ігрові сесії. Це може включати щоденні виклики, міні-квести або короткострокові цілі.

Функція паузи та відновлення. Дозвольте гравцям легко призупиняти та відновлювати гру. Це важливо в мобільних іграх, де переривання є звичайним явищем. Функціональність в автономному режимі та низьке використання даних.

Відтворення в автономному режимі. Спробуйте запропонувати варіанти гри в режимі офлайн. Це збільшує доступність для гравців без постійного доступу до Інтернету та може надати вашій грі конкурентну перевагу.

Оптимізація використання даних. Оптимізуйте свою гру, щоб мінімізувати використання даних. Це особливо важливо для гравців у регіонах з обмеженими або дорогими тарифними планами.

Збалансування елементів онлайн і офлайн. Якщо у вашій грі є онлайн-компоненти, збалансуйте їх офлайн-функціями. Це гарантує, що гра залишається зручною та приємною за різних сценаріїв.

Створіть стійку стратегію монетизації. У сфері мобільних ігор розробка стабільної стратегії монетизації так само важлива, як і створення захоплюючої гри. Цей крок стосується вибору моделі монетизації, яка відповідає дизайну вашої гри та аудиторії, впровадження етичних методів монетизації та пропозиції цінних ігрових предметів або досвіду.

Вибір моделі монетизації. Потрібно розуміння різних моделей. Ознайомтеся з різними моделями монетизації, такими як freemium, pay-to-play і формати з підтримкою реклами. Кожна з них має свої сильні сторони та підходить для різних типів ігор і аудиторії.

Узгодження з ігровим процесом і аудиторією: виберіть модель, яка доповнює ваш основний ігровий процес і цільову аудиторію. Наприклад, звичайні ігри можуть добре працювати з моделями на основі реклами, тоді як більш глибокі ігри можуть отримати користь від покупок у програмі або підписок.

Дослідження ринку. Проведення дослідження ринку, дає можливість зрозуміти, які стратегії монетизації добре працюють у вибраному жанрі та для цільової демографічної групи.

Впровадження етичних методів монетизації. Ненав’язлива реклама. Якщо обирається монетизація на основі реклами, потрібно переконатися, що реклама

ненав’язлива та не заважає ігровому процесу. Оголошення на основі винагороди, де гравці вирішують переглядати рекламу в обмін на винагороди в грі, є популярним вибором.

Чесні покупки в додатку. Для покупок у додатку пропонуються предмети, які додають цінності грі, не створюючи сценарію «плати за перемогу». Цей підхід поважає досвід гравця та заохочує чесну гру.

Варіанти підписки. Якщо розглядається модель підписки, пропонуються відчутні переваги, тобто ексклюзивний вміст або регулярні оновлення, які виправдовують поточні витрати.

Цінні ігрові предмети або досвід. Покращення прогресу гравця підрозуміває внутрішньоігрові предмети або досвід, які позитивно сприяють прогресу гравця. Це можуть бути косметичні предмети, заощадження часу або спеціальні режими гри.

Збалансована пропозиція. Потрібно переконатися, що покупки в грі збалансовані та не дають несправедливої переваги гравцям, які платять. Мета полягає в тому, щоб покращити досвід, а не затьмарити основну механіку ігрового процесу.

Безперервне значення. Щоб покупки в грі залишалися привабливими, потрібно забезпечувати постійну цінність за допомогою оновлень або нового вмісту. Такий підхід може сприяти базі лояльних гравців і стабільному потоку доходу.

Надання пріоритету якості та ефективності. Щоб досягти успіху на конкурентному ринку мобільних ігор, важливо віддавати перевагу якості та продуктивності. Цей крок зосереджується на важливості ретельного тестування, оптимізації для різних пристроїв і використання відгуків гравців для постійного вдосконалення гри.

Інвестиції в ретельне тестування та виправлення помилок. Комплексне тестування: запровадьте ретельний процес тестування, який охоплює всі аспекти гри, включаючи механізми ігрового процесу, інтерфейс користувача та серверні системи. Це допомагає виявити та вирішити проблеми до оприлюднення.

Виправлення помилок. Потрібно виділяти значні ресурси для виправлення помилок. Гра без помилок на момент запуску може значно покращити утримання та задоволення гравців. Такі інструменти, як програмне забезпечення для відстеження помилок, можуть спростити цей процес.

Бета-тестування. Щоб отримати уявлення про те, як гра працює в реальних сценаріях потрібно постійно проводить бета-тестування з реальними користувачами. Це може виявити несподівані помилки або області, які потрібно покращити.

Оптимізація графіки та продуктивності. Переконайтеся, що графіку гри оптимізовано для мобільних пристроїв. Це передбачає створення ресурсів, які добре виглядатимуть на невеликих екранах, і налаштування точності графіки для різних апаратних можливостей.

Продуктивність на різних пристроях. Тестування та оптимізація продуктивності гри на багатьох мобільних пристроях, включаючи старіші моделі. Це гарантує, що гра доступна ширшій аудиторії.

Баланс візуальних ефектів і продуктивності. Потрібно знайти правильний баланс між приголомшливими візуальними ефектами та плавною продуктивністю. Гра, яка виглядає чудово, але працює погано, може погіршити взаємодію з користувачем.

Збір відгуків гравців і періодичні оновлення. Канали зворотного зв'язку. Створіть канали для гравців, щоб надавати відгуки, наприклад ігрові інструменти, форуми або платформи соціальних мереж. Цей відгук є безцінним для розуміння потреб і вподобань гравців.

Ітерації на основі даних. Щоб направляти оновлення та покращення, використовуйте дані гравців і аналітику,. Цей підхід гарантує, що зміни узгоджуються з поведінкою та вподобаннями гравців.

Постійне вдосконалення. Зобов’язуйтесь регулярно оновлювати та ітерації на основі відгуків гравців і аналізу даних. Це не тільки робить гру свіжою та захоплюючою, але й демонструє прагнення до якості та задоволення гравців.

Впроваджуйте ефективний маркетинг і залучайте користувачів. Ефективні

маркетингові стратегії та стратегії залучення користувачів є критично важливими для успіху вашої мобільної гри. Цей крок зосереджений на оптимізації списків у магазині додатків, залученні соціальних мереж і впливових людей, а також розгляді платних кампаній для ширшого охоплення та збільшення кількості завантажень.

Оптимізація вашого лістингу App Store. Оптимізація ключових слів. Використовуйте релевантні ключові слова в списку додатків, щоб покращити видимість у результатах пошуку. Такі інструменти, як App Annie або Sensor Tower, можуть допомогти визначити ефективні ключові слова.

Захоплюючі візуальні ефекти. Додайте захоплюючі знімки екрана та попередній перегляд відео, які демонструють найкращі аспекти вашої гри. Візуальні елементи відіграють вирішальну роль у залученні потенційних гравців.

Переконливі описи: пишіть чіткі, захоплюючі описи, які підкреслюють унікальні особливості та переваги вашої гри. Добре складений опис може значно вплинути на рішення щодо завантаження.

Використання соціальних медіа та маркетингу впливу. Залучення до соціальних медіа: потрібно створити сильну присутність на таких платформах, як Facebook, Twitter та Instagram. Діліться цікавим вмістом, оновленнями та спілкуйтеся зі своєю аудиторією, щоб створити спільноту навколо вашої гри.

Партнерство з інфлюенсерами: співпрацюйте з інфлюенсерами, які резонують із вашою цільовою аудиторією. Інфлюенсери можуть забезпечити довіру та охопити ширшу аудиторію через такі платформи, як YouTube і Twitch.

Створення спільноти. Створюйте спільноту навколо гри, взаємодіючи з гравцями, організовуючи заходи та заохочуючи вміст, створений користувачами.

Розгляд платних кампаній із залучення користувачів. Цільова реклама: інвестуйте в цільові рекламні кампанії на таких платформах, як Google Ads або Facebook Ads. Ці кампанії можна адаптувати для охоплення певних демографічних груп та інтересів, що максимізує їхню ефективність.

Аналіз рентабельності інвестицій: відстежуйте рентабельність інвестицій (ROI) своїх кампаній, щоб зрозуміти їх ефективність. Налаштуйте стратегії на основі даних про ефективність, щоб оптимізувати свої витрати.

Перехресне просування та партнерство: досліджуйте можливості перехресного просування з іншими іграми чи програмами. Партнерство може бути економічно ефективним способом охоплення нової аудиторії.

Розвивайте спільноту та залучайте гравців. Розвиток активної спільноти та постійна залученість гравців є критично важливими для довгострокового успіху мобільної гри. Цей останній крок підкреслює важливість регулярного оновлення вмісту, сприяння взаємодії спільноти та чуйного реагування на відгуки гравців.

Регулярно публікуйте новий вміст, виклики та події. Свіжий вміст: Постійно вводьте нові рівні, персонажів або сюжетні лінії, щоб підтримувати інтерес гравців. Регулярні оновлення можуть знову розпалити хвилювання та дати гравцям причини повернутися.

Захоплюючі виклики та події: впроваджуйте обмежені за часом виклики або спеціальні події, щоб створити відчуття терміновості та хвилювання. Такі ігри, як «Fortnite», демонструють, як події можуть значно підвищити зацікавленість.

Винагороджувальний прогрес: переконайтеся, що новий вміст пропонує як завдання, так і винагороди, сприяючи відчуттю прогресу та досягнення гравця. Створення активної спільноти за допомогою функцій у грі Соціальна інтеграція: додайте такі соціальні функції, як чат у грі, списки друзів або гільдії, щоб стимулювати взаємодію гравців. Це може сформувати почуття спільності та приналежності.

Елементи змагання: запровадьте таблиці лідерів або змагальні режими, які заохочують дружнє змагання. Це може збільшити залучення гравців і поставити цілі, до яких варто прагнути.

Події спільноти: проводьте внутрішньоігрові спільноти чи конкурси, які заохочують гравців до участі та співпраці. Реагування на відгуки гравців і впровадження покращень.

Активне слухання: регулярно відстежуйте відгуки гравців і відповідайте на них на форумах, у соціальних мережах і в ігрових каналах. Розуміння перспектив гравців є ключовим для постійного вдосконалення.

Впровадження змін: будьте готові вносити зміни на основі відгуків гравців. Коригування, спрямовані на загальні проблеми, можуть значно підвищити задоволеність гравців.

Прозоре спілкування: підтримуйте відкрите та прозоре спілкування зі своєю базою гравців. Інформування гравців про майбутні зміни та врахування їхніх відгуків сприяє довірі та лояльності.

У світі мобільних ігор, що постійно розвивається, успіх залежить від всебічного підходу, який охоплює розуміння вашої аудиторії, розробку унікального ігрового процесу, пріоритетність ігрових можливостей, впровадження сталої монетизації, забезпечення якісної продуктивності, залучення до ефективного маркетингу та сприяння міцній спільноті.

Застосовуючи ці принципи, розробники можуть створювати ігри, які не тільки захоплюють гравців, але й витримують випробування часом на конкурентному ринку. Ключ до успішної мобільної гри полягає в безперервних інноваціях, чуйному реагуванні на потреби гравців і невпинній концентрації на якості та взаємодії.

1.2 Потреба у розробці мобільної казуальної гри

Ринок мобільних ігор розвивається надзвичайно швидко, і є багато досліджень, які підтверджують думку, що це довгострокова тенденція. Згідно з дослідженням Statista, дохід від ігор на смартфонах у 2023 році оцінювався у понад 63 мільярди доларів у всьому світі, тоді як дохід від ігор для планшетів досяг 13,7 мільярдів доларів. Прогнози показують, що дохід індустрії мобільних ігор продовжить зростати, перевищивши позначку в 100 мільярдів доларів до 2025 року. У секторі мобільних ігор великі видавці та розробники ігор, такі як EA, Activision Blizzard і Rovio, активно працюють у цьому секторі протягом тривалого часу та заробляють мільйони на таких хітах, як Angry Birds або The Sims Mobile.

Є також нові компанії, які розглядають мобільний бум як можливість вийти на ринок за нижчою ціною, ніж бюджет, необхідний для консольних і ПК. Незалежно від того, вирішите ви розробити щось унікальне чи копію популярної гри, вам потрібно зрозуміти, скільки це коштуватиме, перш ніж почати. Тут, у Room 8 Studio, ми постійно шукаємо креативні ідеї для розробки наступного ігрового хіта та любимо співпрацювати з партнерами, які поділяють те саме бачення. Хоча для успіху недостатньо мати чудову ідею, для її створення потрібна відмінна команда.

Спираючись на великий досвід створення ігор з нуля, існує декілька підходів. Для створення мобільних ігор розглядають основні етапи процесу та можливі витрати. Оцінка вартості розробки мобільної гри враховує кілька факторів.

Платформа: iOS, Android, Windows або всі. технології. Найпоширенішими варіантами є: Unreal Engine і Unity. Незважаючи на те, що Unreal має різноманітні та досить недорогі ресурси та сценарії як частину своєї бібліотеки, пропонує чудовий контроль процесів та можливості оптимізації, цей движок все ще орієнтований на розробку ПК та консолей. З іншого боку, Unity є повністю кросплатформною, набагато більш гнучкою та пропонує більш широкий спектр інструментів, включаючи спеціальні інструменти для iOS та Android, що робить її найкращою для мобільної розробки. Зосередимось на Unity як на нашій основній технології.

Складність. Щоб дізнатися, скільки коштує створення гри, спочатку потрібно оцінити її потенційну складність. Ця цілісна концепція включає низку важливих аспектів: жанр, режим (однокористувацький, багатокористувацький або соціальний), цільові ринки та потреби локалізації. Крім того, слід включити витрати на підтримку після запуску, щоб покривати регулярні оновлення та додавання нового вмісту (наприклад, рівні гри, нові персонажі, скіни тощо), які також слід планувати заздалегідь. Ігровий дизайн і історія. Те, чи сподобається гравцям гра чи ні, зрештою залежить як від дизайну, так і від сюжетної лінії, тому це важливі частини проекту. Тут потрібно визначити особливості ігрового

процесу, персонажів і середовище, а також дизайн UI/UX. Для правильної оцінки витрат на розробку ігрових додатків, знадобиться підтримка різних спеціалістів — художників, дизайнерів ігор тощо. Крім того, найефективніші ігри завжди практично без помилок і високопродуктивні на широкому діапазоні пристроїв, виходячи з цільової аудиторії. Щоб гра була успішною потрібно завжди враховувати її вартість.

В цілому мобільні ігри поділяються на декілька категорій. Ця класифікація допомогає оцінити діапазон витрат на розробку мобільних ігор.

Розглянемо міні та гіперказуальні ігри. Такі ігри візуально прості та зрозумілі. Гравцям пропонується простий досвід із короткими (або міні-сесіями), а потім вони можуть легко відключитися.

Міні- та гіперказуальні ігри, як правило, містять спрощену, але повторювану механіку та значну увагу приділяють постійним винагородам і звичній поведінці. Гіперказуальні ігри потрібно створювати надзвичайно швидко та в безперервному ланцюжку одна за одною — одна гра вийшла, дві гри у виробництві. Виходячи з швидкоплинної тенденції, час для виходу на ринок однієї гіперказуальної гри є вирішальним. Завдяки такому економічно ефективному підходу гіперказуальна гра може коштувати від 25 000 до 50 000 доларів США.

Казуальні ігри використовують спрощену основну механіку, але їх геймплей складніший, ніж міні- чи гіперказуальні ігри. Вони повинні мати добре збалансовану економіку, що дає можливість додавати додаткові ігрові функції, такі як збір ігрової валюти, розблокування нових персонажів, скінів, рівнів або навіть мета-ігор. Найпопулярнішими казуальними жанрами є головоломки (особливо ті, що базуються на механіці «Три в ряд»), аркади, платформери та прості симулятори (наприклад, ігри з управлінням часом і містом). Згідно з прогнозами Deconstructor of Fun 2024, ігри-головоломки стануть найприбутковішим жанром на мобільних пристроях цього року. Загалом у казуальних іграх на головоломки припадає трохи більше 50% усіх доходів у 2023 році.

Розробка повного циклу казуальної мобільної гри може коштувати від 400 000 до 1 000 000 доларів або навіть більше, залежно від обсягу та складності. У нижній точці цього діапазону знаходяться такі ігри, як проста гра в слова — One Word, тоді як платформер або гра з управлінням часом, наприклад, Manage the Stars, будуть на вищому рівні.

У казуальних іграх також слід враховувати витрати на живу підтримку операцій, яка не включена в початковий бюджет розробки, але є важливою для цих ігор.

1.3 Як саме створюється захоплююча мобільна гра

Розробка захоплюючої гри для мобільних пристроїв включає кілька ключових етапів, спрямованих на створення привабливого ігрового досвіду для користувачів. Основна мета - надати гравцям можливість весело провести час за допомогою гри. Для досягнення цієї мети необхідно виконати ряд важливих завдань.

Перший етап включає аналіз існуючих мобільних ігор, які користуються популярністю серед гравців. Такий аналіз є критично важливим для створення гри, яка буде привабливою та цікавою для користувачів. Визначення основних характеристик успішних ігор допоможе виявити елементи, які слід враховувати при розробці нової гри.

На наступному етапі необхідно визначити, які потреби користувачів має задовольнити гра. Розробка концепції включає врахування елементів геймдизайну та створення геймплею, який відповідає вимогам та забезпечує захоплюючий ігровий досвід для користувачів. Це може включати визначення жанру гри, розробку основної механіки, сюжету та візуального стилю.

Реалізація мобільної гри є наступним важливим етапом. Використання мови програмування C# та ігрового рушія Unity, дозволяє створювати високоякісні ігри. Реалізація включає створення всіх необхідних компонентів гри, забезпечення взаємодії. Важливо також забезпечити, щоб гра була функціональною та

привабливою для користувачів, відповідала очікуванням гравців та працювала стабільно на різних пристроях.

Тестування та виправлення помилок є останнім, але не менш важливим етапом. Це дозволить забезпечити високу якість кінцевого продукту та задовольнити вимоги користувачів.

Таким чином, виконання цих етапів дозволить створити захоплюючу та якісну мобільну гру, яка відповідатиме потребам і очікуванням користувачів та забезпечить високу якість кінцевого продукту.

1.4 Чому саме казуальна гра у жанрі гонки?

Створення штучних перешкод у грі є ключовим елементом для забезпечення захоплюючого геймплею. Ці перешкоди можуть мати різну природу: від важкодоступних об'єктів до складних лабіринтів чи ворожих персонажів. Їх присутність створює виклики та труднощі для гравця, підвищуючи емоційне напруження та стимулюючи активну участь.

Процес створення штучних перешкод вимагає ретельного планування та дизайну. Кожна перешкода повинна бути добре продумана, щоб забезпечити баланс між складністю та досяжністю. Важливо враховувати рівень навичок гравця та його здатність адаптуватися до нових викликів. Перешкоди не повинні бути занадто складними або, навпаки, надто легкими; оптимальний баланс складності забезпечує захоплюючий ігровий досвід.

Штучні перешкоди дозволяють розширити межі геймплею та зробити його більш різноманітним. Вони можуть бути представлені у вигляді великих стін, маленьких кущів або інших об'єктів, які додають динаміки і різноманітності в гру. Це робить ігровий процес багатогранним і цікавим для гравців.

Перешкоди також впливають на взаємодію гравця з іншими елементами гри. Вони додають глибини та складності до геймплею, що робить гру більш цікавою та захоплюючою. Не лише наявність перешкод, але й їх розташування, взаємодія з іншими елементами гри і логіка їхнього подолання є важливими аспектами при створенні гри.

Загалом, створення штучних перешкод є невід'ємною частиною процесу розробки гри. Вони додають необхідні виклики, що утримують увагу гравця та стимулюють його продовжувати грати. Правильно спроектовані перешкоди можуть значно покращити ігровий досвід, роблячи його більш захоплюючим і цікавим для різноманітної аудиторії.

1.5 Роль створення штучних перешкод в грі

Створення штучних перешкод у грі є ключовим елементом для забезпечення захоплюючого геймплею. Ці перешкоди можуть мати різну природу: від важкодоступних об'єктів до складних лабіринтів чи ворожих персонажів. Їх присутність створює виклики та труднощі для гравця, підвищуючи емоційне напруження та стимулюючи активну участь.

Процес створення штучних перешкод вимагає ретельного планування та дизайну. Кожна перешкода повинна бути добре продумана, щоб забезпечити баланс між складністю та досяжністю. Важливо враховувати рівень навичок гравця та його здатність адаптуватися до нових викликів. Перешкоди не повинні бути занадто складними або, навпаки, надто легкими; оптимальний баланс складності забезпечує захоплюючий ігровий досвід.

Штучні перешкоди дозволяють розширити межі геймплею та зробити його більш різноманітним. Вони можуть бути представлені у вигляді великих стін, маленьких кущів або інших об'єктів, які додають динаміки і різноманітності в гру. Це робить ігровий процес багатогранним і цікавим для гравців.

Перешкоди також впливають на взаємодію гравця з іншими елементами гри. Вони додають глибини та складності до геймплею, що робить гру більш цікавою та захоплюючою. Не лише наявність перешкод, але й їх розташування, взаємодія з іншими елементами гри і логіка їхнього подолання є важливими аспектами при створенні гри.

Загалом, створення штучних перешкод є невід'ємною частиною процесу розробки гри. Вони додають необхідні виклики, що утримують увагу гравця та стимулюють його продовжувати грати. Ефективно розроблені перешкоди можуть

значно поліпшити ігровий процес, зробивши його більш цікавим і захоплюючим для різних типів гравців.

2. АНАЛІЗ ТА ОСОБЛИВОСТІ КАЗУАЛЬНИХ ІГОР

2.1 Аналіз казуальних ігор для Android та IPhone пристроїв

На платформах Android та iPhone існує велика кількість різноманітних казуальних ігор, доступних для завантаження. Деякі з найпопулярніших площадок для отримання цих ігор включають:

1. **Google Play та App Store**: Ці платформи є основними магазинами додатків для Android та iPhone пристроїв відповідно. Вони мають великий вибір казуальних ігор різних жанрів та категорій, доступних для безкоштовного або платного завантаження.
2. **Steam**: Хоча Steam відомий своїми комп'ютерними іграми, він також пропонує широкий вибір казуальних ігор для Android пристроїв через додаток Steam Link.
3. **itch.io**: Це онлайн-магазин, де розробники можуть розміщувати свої ігри, включаючи казуальні, незалежно від їхньої популярності чи жанру.
4. **Kongregate та Game Jolt**: Ці платформи спеціалізуються на браузерних іграх, включаючи казуальні, і надають можливість грати в них онлайн або завантажувати на мобільні пристрої.

Переваги казуальних ігор включають:

Доступність: вони легко доступні і здатні працювати на широкому спектрі пристроїв, включаючи мобільні телефони та планшети.

Простий геймплей: казуальні ігри зазвичай мають простий інтерфейс та геймплей, що дозволяє користувачам швидко вступити у гру без необхідності вивчення складних правил.

Незважаючи на це, є деякі недоліки казуальних ігор:

Обмежена глибина геймплею: вони часто не мають складного сюжету або глибокої механіки гри, що може призвести до того, що гравці швидко втрачають інтерес.

Монетизація: деякі казуальні ігри використовують агресивну модель монетизації, що може включати в себе часті рекламні відео, платні додаткові функції або мікротранзакції, що робить гру менш приємною для користувачів.

Не дивлячись на недоліки, казуальні ігри залишаються популярними серед широкої аудиторії гравців, які шукають прості та легкі у використанні ігри для своїх мобільних пристроїв.

2.2 Аналіз механік казуальних ігор

Аналіз механік казуальних ігор включає вивчення основних принципів геймплею, які роблять гру привабливою та захоплюючою для гравців. Деякі з основних механік, які зазвичай зустрічаються в казуальних іграх, включають:

Простий контроль: Казуальні ігри зазвичай мають простий та інтуїтивно зрозумілий контроль, що дозволяє гравцям легко взаємодіяти з грою без значного навантаження на їхні моторні навички чи реакцію.

Короткі рівні або завдання: Багато казуальних ігор використовують короткі рівні або завдання, які можна швидко завершити. Це дозволяє гравцям грати в гру на короткі періоди часу без необхідності витрачати багато часу на одну гру.

Прості головоломки та завдання: Казуальні ігри зазвичай мають прості головоломки або завдання, які не вимагають великої кількості розумових зусиль для їх вирішення. Це дозволяє гравцям насолоджуватися грою без стресу чи надмірного напруження.

Механіка "легко вчитися, важко опанувати": Багато казуальних ігор мають просту механіку гри, але водночас можуть мати складніші рівні або завдання, які вимагають від гравців стратегічного мислення та добре обдуманих дій.

Системи нагородження: Ще однією поширеною механікою казуальних ігор є системи нагородження, такі як отримання бонусів, досягнень або нових рівнів. Ці системи нагородження стимулюють гравців до подальшої гри та досягнення нових цілей.

Аналіз механік казуальних ігор допомагає розробникам краще зрозуміти, що робить гру привабливою для гравців та як можна поліпшити їхній досвід гри. Це дозволяє створювати більш захоплюючі та задовольняючі ігри для широкої аудиторії гравців.

2.3 Аналіз звукової частини ігор

Звукова частина гри є ключовим елементом, який значно впливає на загальний досвід гравця. Аналіз звукової складової казуальних ігор включає дослідження наступних аспектів:

Музикальний супровід: Музика в грі має ключове значення для формування атмосфери та настрою. Вона може бути енергійною та захопливою під час екшн-сцен, спокійною та релаксуючою під час менших моментів або напруженою та драматичною в рішальних моментах гри.

Звукові ефекти: Звукові ефекти використовуються для підсилення дій та подій у грі. Наприклад, звук вибуху, ковзання або падіння може допомогти гравцеві краще зрозуміти події на екрані та глибше зануритися в ігровий процес.

Голосові команди та діалоги: У деяких казуальних іграх використовуються голосові команди або діалоги для надання інструкцій гравцеві або розповіді історії. Якщо діалоги виконані якісно та відповідають характеру гри, вони можуть значно підвищити іммерсивність ігрового процесу.

Атмосфера та настрій: Звукова частина гри може створювати особливу атмосферу та передавати певний настрій. Наприклад, звуки природи, містичні мелодії або наркотичні ритми можуть допомогти створити відповідний фон для гри відповідного жанру.

Оптимізація звуку: Важливо враховувати, що звукова частина гри повинна бути оптимізована для різних пристроїв та звукових систем. Звук повинен відтворюватися якісно навіть на пристроях з невеликими динаміками або слабкими звуковими картами.

Аналіз звукової частини казуальних ігор допомагає розробникам створити ігри, які не тільки візуально привабливі, але й звучать захопливо та зацікавлюючe, що підвищує загальний рівень задоволення від гри.

2.4 Аналіз графіки та эффектів

Аналіз графіки та ефектів в казуальних іграх є ключовим етапом у розробці, оскільки вони значно впливають на загальний вигляд та враження від гри. Важливі аспекти аналізу включають:

Стиль графіки: Казуальні ігри можуть мати різноманітний стиль графіки, від анімаційних ілюстрацій до реалістичних 3D моделей. Вибір стилю графіки повинен відповідати тематиці гри та приваблювати цільову аудиторію.

Дизайн персонажів та об'єктів: Деталізація та оригінальність дизайну персонажів та об'єктів гри важлива для створення візуально привабливого середовища. Чіткі та яскраві образи можуть привернути увагу гравців та створити запам'ятовуваність гри.

Анімація: Плавність та реалістичність анімації персонажів та об'єктів гри додає іммерсивності та життєвості ігровому світу. Добре виконана анімація може підвищити залученість гравців та зробити гру більш захоплюючою.

Візуальні ефекти: Використання спеціальних візуальних ефектів, таких як світлові, часткові, плоскість і туман, може значно підвищити естетичний вигляд гри та створити атмосферу.

Оптимізація графіки: Важливо, щоб графіка була оптимізована для різних пристроїв та платформ, оскільки користувачі грають на різних пристроях з різними характеристиками. Ефективне використання ресурсів гарантує плавну гру без затримок чи зависань.

Аналіз графіки та ефектів допомагає розробникам створити візуально привабливу та естетичну гру, яка привертає увагу та задовольняє користувачів.

2.4 Концепту та геймплей казуальних ігор

Концепція та геймплей казуальних ігор є ключовими складовими успішної гри, оскільки вони визначають її унікальність, привабливість та відчуття задоволення від гри. Основні аспекти аналізу концепції та геймплею включають:

Оригінальність ідеї: Успішна казуальна гра часто базується на унікальній та цікавій ідеї, яка відрізняється від інших ігор на ринку. Важливо, щоб концепція була привабливою та захоплюючою для цільової аудиторії.

Простота та доступність: Казуальні ігри мають простий та легкий для зрозуміння геймплей, що дозволяє користувачам легко вникнути в гру без значних зусиль чи попереднього досвіду в іграх.

Зв'язок з тематикою: Концепція та геймплей повинні гармонійно поєднуватися з тематикою гри. Це створює атмосферу, яка відповідає цілям та амбіціям розробників, а також привертає увагу цільової аудиторії.

Елементи гейміфікації: Включення елементів гейміфікації, таких як досягнення, лідерські таблиці, бонуси та винагороди, може зробити гру більш захоплюючою та стимулюючою для гравців.

Поступове ускладнення: Хороші казуальні ігри часто мають механізм поступового ускладнення, що дозволяє гравцеві поступово розвиватися та вдосконалювати свої навички, зберігаючи при цьому високий рівень інтересу та мотивації.

Аналіз концепції та геймплею допомагає розробникам створити захоплюючу та цікаву гру, яка привертає увагу гравців та забезпечує їм задоволення від гри.

2.5 Приклади казуальних ігор:

Рисунок 2.1 – Знімок екрану гри “super MARIO BROS.”

Super Mario Bros. – це легендарна відеогра, яка була створена компанією Nintendo та випущена в 1985 році. Гра була розроблена Шіґеру Міядзакі та Такасі Тезука, і стала першою частиною серії ігор про Маріо, який став символом для Nintendo. Належить до жанру платформерів, що означає: гравець керує персонажем, який пересувається по рівням, стрибає через препятствія та бореться з ворожими персонажами. Головним героєм гри є водопровідник на ім'я Маріо, який повинен врятувати принцесу Піч з полону лиходія Боузера, схожого на черепаху.

Super Mario Bros. внесла значний вклад у світ відеоігор. Вона встановила нові стандарти для ігор та стала однією з найуспішніших ігор усіх часів. Гра вперше використовувала горизонтальний скролінг, що дозволяло гравцям досліджувати великі рівні. Також вона вперше використовувала "труби" як спосіб переходу між рівнями, що стало одним з найвпізнаваніших елементів серії.

Завдяки успіху на ринку Super Mario Bros вплинула на індустрію відеоігор. Вона допомогла підняти компанію Nintendo на вершину успіху та стала символом їхньої консолі NES. Гра також дала початок великій серії ігор про Маріо, яка стала однією з найвідоміших та найуспішніших серій в історії відеоігор.

Рисунок 2.1 – Знімок екрану гри “Fruit Ninja”

Fruit Ninja – це аркадна відеогра, розроблена студією Halfbrick Studios та випущена в 2010 році. Гра доступна на різних платформах, включаючи iOS, Android, Windows Phone та інші. Основна механіка гри полягає в тому, щоб різати фрукти, які постійно з'являються на екрані, за допомогою перетягування пальцем по сенсорному екрану. Гравцю належить уникати різати бомби, які також можуть з'явитися серед фруктів, оскільки це призведе до завершення гри. Гравець отримує бали за кожен розрізаний фрукт і має можливість використовувати різні бонуси та покращення. Fruit Ninja стала дуже популярною завдяки своїй простій, але дуже веселій геймплейній механіці. Гра отримала позитивні відгуки за графіку, звуковий супровід та загальну геймплейну концепцію. Вона також стала однією з найуспішніших мобільних ігор, з більш ніж мільярдом завантажень з різних магазинів додатків. Fruit Ninja відкрила новий жанр "фруктових аркад", який став популярним серед гравців усіх вікових груп. Гра також стала прикладом успішного використання сенсорного екрану мобільних пристроїв у геймплейні, що відкрило шлях для подібних ігор у майбутньому.

3. ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

3.1 Прегляд мов програмування та ігрових двигунів

При розробці казуальних ігор, використання відповідних мов програмування та ігрових рушіїв є надзвичайно важливим. Вибір правильних інструментів впливає на швидкість і ефективність розробки, а також на якість готової гри.

Серед найпопулярніших ігрових рушіїв для створення мобільних ігор особливо вирізняється Unity, який підтримує як 2D, так і 3D розробку. Він має інтуїтивно зрозумілий інтерфейс і багатий набір інструментів для роботи з графікою, анімацією, звуком та іншими аспектами гри. Крім того, Unity використовується для створення ігор на різних платформах, включаючи Android, iOS, Windows та інші.

Однією з головних мов програмування на Unity є C#. Це потужна мова програмування, яка надає розробникам великі можливості для створення ефективних і продуктивних ігор. Вона має зрозумілий синтаксис і багату бібліотеку, що робить її легкою у вивченні та використанні. Завдяки своїй потужності та гнучкості, C# дозволяє швидко та ефективно створювати як прості, так і складні ігри.

Серед інших ігрових рушіїв варто відзначити Unreal Engine, який дозволяє створювати високоякісні 3D ігри з великою кількістю візуальних ефектів. Він підтримує мову програмування C++ і має потужну систему візуалізації, що робить його відмінним вибором для створення складних ігор. Проте Unreal Engine може бути складним у використанні для новачків і надмірним для простих казуальних ігор.

Розглянемо мову програмування Java, яка використовується для розробки ігор на платформі Android та є популярною. Вона надає велику кількість фреймворків, бібліотек та інструментів для розробки 2D і 3D ігор. Однак, порівняно з іншими мовами, Java може бути менш продуктивною для високопродуктивних ігор. Swift, що використовується для розробки ігор на платформі iOS, відзначається високою швидкодією та великою кількістю інструментів для розробки графіки та аудіо. Проте, обмежена підтримка на інших платформах робить її менш універсальною. Kotlin, що також застосовується для розробки ігор на Android, має схожий синтаксис з Java, але є більш продуктивним та гнучким.

Вибір ігрового рушія та мови програмування залежить від конкретних завдань і потреб розробника, а також його кваліфікації та досвіду. Наприклад, для

розробки мобільної гри під Android з використанням C# Unity є оптимальним вибором. Він добре підходить для мобільних платформ, підтримує мову C# та має інтуїтивний інтерфейс, що значно спрощує процес розробки.

Таким чином, вибір правильного ігрового рушія та мови програмування є важливим кроком у процесі створення казуальних ігор. Це дозволяє забезпечити ефективну та якісну розробку, що, в свою чергу, впливає на успіх і популярність готового продукту серед користувачів.

3.2 Використання Unity з C# - кращий вибір

Використання Unity з мовою програмування C# є одним з найкращих виборів для розробки ігор, зокрема казуальних. Ось кілька причин, чому ця комбінація є оптимальною:

Потужність Unity: Unity є одним з найпопулярніших і потужних ігрових движків на ринку. Він надає розробникам велику кількість інструментів для створення якісних ігор з різноманітними функціями та ефектами.

Кросплатформеність: Unity дозволяє легко розгортати гру на різних платформах, включаючи Android, iOS, ПК, консолі та веб. Це дозволяє досягти більшої аудиторії гравців та максимізувати прибуток від продажів.

Мова програмування C#: C# є потужною та ефективною мовою програмування, яка використовується для розробки ігор у Unity. Вона має зручний синтаксис та широкі можливості, що робить її привабливим вибором як для початківців, так і для досвідчених розробників.

Широкі можливості: Використання Unity з C# дозволяє реалізувати різноманітні ідеї та концепції гри. Ви можете створити як 2D, так і 3D ігри з різноманітним геймплеєм та механіками, використовуючи широкий спектр доступних інструментів та ресурсів.

Підтримка та спільнота: Unity володіє великою та активною спільнотою розробників, що надає підтримку, навчальні матеріали та різноманітні ресурси для вдосконалення навичок у створенні ігор.

Отже, використання Unity з мовою програмування C# є кращим вибором для створення казуальних ігор завдяки потужності движка, кросплатформеності, ефективності мови програмування та широким можливостям для реалізації різноманітних ідей та концепцій гри.

3.3 Blender для створення об'єктів

Blender - це потужний і багатофункціональний графічний редактор, який використовується для створення об'єктів у грі. Він є важливим інструментом у розробці казуальних ігор, особливо коли йдеться про створення 3D-графіки. Наведемо кілька причин, чому використання Blender є кращим вибором для створення об'єктів у порівнянні з іншими графічними програмами:

Безкоштовність: Blender є повністю безкоштовним програмним забезпеченням з відкритим вихідним кодом, що робить його доступним для широкого кола користувачів.

Різноманітність функцій: Blender охоплює широкий набір інструментів для моделювання, текстурування, анімації та рендерингу, що дозволяє створювати різноманітні та складні об'єкти для ігор.

Співпраця з Unity: Blender ідеально інтегрується з Unity, що дозволяє легко і швидко імпортувати створені об'єкти у вашу гру. Це полегшує процес розробки і дозволяє зосередитися на створенні якісного контенту.

Активна спільнота: у Blender є велика та активна спільнота користувачів, яка надає підтримку, навчальні матеріали та поради щодо використання програми. Це дозволяє швидко засвоювати нові функції та розвивати свої навички у галузі 3D-моделювання.

Загалом, використання Blender для створення об'єктів у казуальних іграх є вигідним вибором завдяки його безкоштовності, різноманітності функцій, співпраці з Unity та активній спільноті користувачів.

3.4 Кінцеві технології розробки

Переличимо та розглянемо технології розробки: Unity, Git, GitLab, IntelliJ IDEA, Dbeaver. На рис. 3.1 представлено технології розробки.

Рисунок 3.1 – Технології розробки

Unity - це потужний ігровий рушій, який дозволяє розробляти 2D та 3D ігри для різних платформ, включаючи мобільні пристрої, ПК та консолі. Він забезпечує широкий набір інструментів для створення інтерактивного контенту, таких як фізика, анімація, рендеринг, аудіо, та багато іншого. Unity використовує мову програмування C#, що дозволяє легко писати скрипти для управління поведінкою об'єктів у грі.

Git - це система контролю версій, яка дозволяє розробникам відслідковувати зміни в коді, працювати над проектами в команді та зберігати історію змін. Git забезпечує ефективне злиття коду, гілкування та підтримку різних версій проекту. Це допомагає запобігти втраті даних і дозволяє легко керувати різними версіями програмного забезпечення.

GitLab - це веб-сервіс для управління репозиторіями Git, який надає інструменти для безперервної інтеграції та безперервної доставки (CI/CD). GitLab дозволяє командам співпрацювати над кодом, відслідковувати помилки, автоматизувати процеси розгортання та багато іншого. Він забезпечує надійне середовище для керування проектами та підтримки розробки програмного забезпечення.

IntelliJ IDEA - це інтегроване середовище розробки (IDE), яке підтримує різні мови програмування, включаючи Java, Kotlin, Groovy та інші. IntelliJ IDEA забезпечує широкий набір функцій для підвищення продуктивності розробників, таких як автодоповнення коду, рефакторинг, інтеграція з системами контролю версій, інструменти для тестування та налагодження. Це допомагає розробникам писати якісний код швидше та ефективніше.

DBeaver - це універсальний інструмент для роботи з базами даних, який підтримує більшість популярних СУБД, включаючи MySQL, PostgreSQL, SQLite, Oracle та інші. DBeaver забезпечує зручний графічний інтерфейс для адміністрування баз даних, виконання SQL-запитів, візуалізації даних та аналізу структури баз даних. Він допомагає розробникам та адміністраторам баз даних ефективно керувати даними та виконувати різноманітні операції з базами даних.

4. РОЗРОБКА ГРИ

4.1 Визначення критеріїв та плану розробки гри

Визначення жанру гри

Перед тим як приступити до розробки сценарію гри, важливо чітко визначити її жанр. Варіантом є аркада, головоломка, симулятор або будь-який інший тип

казуальної гри. Обрання жанру допоможе сконцентруватися на певній тематиці, механіці гри та інших важливих аспектах.

Визначення мети гри

Залежно від жанру, потрібно визначити основну мету гри. Гравці можуть збирати бонуси, розв'язувати головоломки. Чітко визначена мета допоможе структурувати сценарій та забезпечить напрямок для подальшої розробки.

Створення головного героя

Головний герой є ключовим елементом гри. Персонаж повинен мати унікальні характеристики, які зроблять його цікавим та запам'ятовуваним для гравців. Варто розробити бекграунд героя, його мотивацію та особливості.

Сюжет

Сюжет має бути простим, проте він буде захоплювати увагу гравців. Використовуйте діалоги та події, щоб розкрити характери персонажів і просунути сюжет. Сюжет може включати в себе завдання, місії, несподівані повороти подій та розвиток персонажів.

Ігрова механіка

Ігрова механіка повинна відповідати жанру та меті гри. Це можуть бути різні види взаємодії, такі як збори предметів, вирішення головоломок, бойові дії тощо. Важливо визначити ігрові механіки на початковому етапі, оскільки це значно спростить подальшу розробку.

Рівні гри

Рівні повинні бути цікавими та захоплюючими, відповідати меті гри та ігровій механіці. Рівні мають бути прогресивними, тобто складність буде зростати з кожним новим рівнем. Важливо забезпечити баланс між складністю та цікавістю, щоб гравці не втратили інтерес.

Додаткові елементи гри

Додавання різних бонусів, можливості виконувати завдання від інших персонажів або змінювати зовнішність головного героя зробить гру більш цікавою та захоплюючою. Ці елементи можуть стимулювати гравців повертатися до гри та продовжувати грати.

Тестування гри

Після створення сценарію необхідно протестувати гру, щоб переконатися, що вона цікава та захоплює гравців. Виправлення помилок, додавання нових елементів та покращення ігрового процесу допоможуть зробити гру кращою. Тестування також допоможе зібрати відгуки від гравців та внести необхідні зміни. Створення сценарію казуальної гри є важливим етапом, який впливає на успішність гри. Правильне визначення жанру, мети, головного героя, сюжету, ігрової механіки та рівнів допоможе створити захоплюючу гру. Тестування та покращення гри на основі відгуків дозволить досягти високої якості та забезпечити інтерес гравців.

4.2 Взаємодія з користувачем

Сенсорний екран відіграє ключову роль у мобільних іграх, оскільки він дозволяє гравцям взаємодіяти з грою безпосередньо через жести і дотики. Це надає спосіб керування, який є більш прямим і інтуїтивним порівняно з традиційними методами, такими як клавіатура та миша. Сенсорні екрани відкривають нові можливості для гравців, дозволяючи їм відчути гру на більш глибокому рівні і зробити процес керування більш природним та захоплюючим.

Одним із найбільш поширених способів взаємодії з сенсорним екраном є дотик, або тапання. Цей жест дозволяє гравцям вибирати об'єкти, запускати дії та взаємодіяти з інтерфейсом гри. Наприклад, в іграх жанру "Match-3", гравці можуть переміщувати елементи на екрані, просто торкаючись їх і перетягуючи на потрібне місце. Це створює відчуття прямої взаємодії з ігровим світом і робить геймплей більш захоплюючим.

Іншим важливим жестом є свайп, який використовується для переміщення об'єктів або прокручування екрану. У "runner" іграх, наприклад, гравці можуть уникати перешкод, виконуючи свайпи вліво або вправо, а також стрибати, натискаючи на екран або проводячи пальцем вгору. Це дозволяє гравцям легко керувати персонажем, використовуючи природні рухи.

Мультитач жести, такі як щипок для зменшення і розведення пальців для збільшення, також стали стандартом в сучасних мобільних іграх. Ці жести дозволяють гравцям змінювати масштаб і обертати об'єкти, що особливо корисно в стратегіях і симуляторах, де потрібно детально розглядати ігровий світ або керувати великою кількістю об'єктів. Наприклад, в іграх жанру "стратегія", гравці можуть використовувати мультитач жести для зручного огляду та керування своїми військами та ресурсами.

Сенсорний екран також відкриває можливості для нових жанрів ігор, які раніше не були можливими на інших платформах. Наприклад, інтерактивні історії та головоломки часто використовують специфічні сенсорні жести для взаємодії з сюжетом і середовищем гри. Це дозволяє гравцям відчувати себе частиною історії, впливаючи на її розвиток через свої дії.

Однак, взаємодія з сенсорним екраном має свої виклики. Точність керування може бути проблемою, особливо в іграх, де потрібна висока точність. Наприклад, в іграх, де потрібно точно націлитися або швидко реагувати на зміни, сенсорний екран може бути менш зручним, ніж традиційні методи. Також відбитки пальців на екрані можуть знижувати якість зображення і створювати перешкоди для точного керування. Це особливо актуально для ігор з багатим візуальним дизайном і деталізованою графікою.

Звук відіграє важливу роль у взаємодії гри з користувачем, надаючи додаткові шари емоцій та інформації. Звукові ефекти можуть повідомляти гравця про важливі події в грі, такі як зміна рівня, отримання повідомлень або попередження про небезпеку. Наприклад, у шутерах звуки пострілів і вибухів не лише створюють атмосферу, але й допомагають гравцеві орієнтуватися в просторі, визначаючи напрямок і відстань до джерела звуку.

Музика також має значний вплив на ігровий досвід. Вона може створювати певний настрій і підсилювати емоції гравця. Наприклад, напружена музика в хорор-іграх може викликати відчуття тривоги і страху, тоді як мелодійна музика в пригодницьких іграх може створювати атмосферу захоплення і дослідження.

Музика допомагає гравцям глибше зануритися в ігровий світ і переживати його події більш емоційно.

Голосове керування є ще однією цікавою можливістю для взаємодії з грою. Деякі ігри використовують голосові команди для керування персонажем або іншими ігровими об'єктами. Це може бути особливо корисно в ситуаціях, коли гравець не може використовувати сенсорний екран, наприклад, під час активних фізичних занять або у випадках, коли руки зайняті. Голосові команди можуть забезпечувати більш інтерактивний досвід і дозволяти гравцям відчувати себе більш зануреними у гру.

Звук має свої переваги та виклики. Він допомагає занурити гравця у світ гри, роблячи досвід більш реалістичним і захоплюючим. Звукові сигнали можуть допомагати гравцеві в орієнтації та виконанні завдань, надаючи важливу інформацію без необхідності постійно дивитися на екран. Однак, звук може бути неефективним у шумному середовищі або якщо гравець вимушений грати без звуку. Якість звуку також може залежати від апаратних можливостей пристрою, що може обмежувати його ефективність у деяких випадках.

Взаємодія гри з користувачем за допомогою сенсорного екрану та звуку значно підвищує інтуїтивність та іммерсивність геймплею. Сенсорний екран забезпечує простий та прямий спосіб керування, тоді як звук допомагає створити атмосферу та покращити взаємодію з грою. Разом ці елементи створюють більш захоплюючий та інтерактивний досвід для гравців, роблячи мобільні ігри одними з найбільш інноваційних та захоплюючих форм розваг.

4.3 Розробка архітектури гри

Розробка архітектури гри починається з визначення основних компонентів, які будуть складати гру. Цей етап включає кілька ключових елементів:

По-перше, геймплей, який визначає основні механіки та логіку гри, відображає, як гравці взаємодіють з грою. По-друге, графіка, яка включає відображення всіх візуальних елементів гри. Звук також є важливим компонентом, що включає музичний супровід, звукові ефекти та голосові

озвучки. Фізика відповідає за реалістичну поведінку об'єктів у грі, а штучний інтелект (ШІ) створює логіку поведінки неігрових персонажів, додаючи інтерактивність і виклики для гравця. Для багатокористувацьких ігор важливим є також мережевий геймплей, що забезпечує взаємодію між гравцями через мережу.

Визначення взаємодії цих компонентів та їх залежностей є наступним критичним кроком. Один з ефективних підходів до розробки архітектури гри – це розбивка гри на окремі модулі або компоненти, які можна розробляти окремо і згодом інтегрувати. Такий підхід дозволяє одночасно працювати над різними частинами гри та зменшує ризик конфліктів між розробниками.

Використання шаблонів проектування є ще одним важливим аспектом. Шаблони проектування надають універсальні рішення для стандартних проблем, з якими стикаються під час розробки програмного забезпечення.

Вони дозволяють швидко та ефективно вирішувати типові проблеми та зменшувати кількість помилок. Наприклад, шаблон Singleton забезпечує існування лише одного екземпляра класу, тоді як Factory Method використовується для створення об'єктів без прямого визначення їх класу. Шаблон Observer дозволяє об'єктам слідкувати та реагувати на зміни в інших об'єктах, а Decorator додає нову функціональність до існуючих об'єктів. Adapter дозволяє використовувати існуючий клас з іншим інтерфейсом.

Шаблони проектування поділяються на кілька категорій: шаблони створення об'єктів (Creational Patterns), структури (Structural Patterns) та поведінки (Behavioral Patterns). Кожна категорія вирішує певні проблеми, пов'язані зі створенням, організацією та співпрацею між об'єктами.

Патерн "Сервіс-орієнтована архітектура" (SOA) є одним з найбільш популярних у розробці ігор. Він розбиває код на складові, де кожен сервіс має конкретно своє завдання, і взаємодіє з іншими сервісами через чітко визначені інтерфейси. Використання SOA допомагає розділити функціональність гри на легко управляючі та модульні компоненти, що сприяє легкості розробки, підтримки та тестування коду.

Додатково, у Unity широко використовуються популярні патерни та архітектури, такі як Zenject для інверсії керування (IoC) та архітектура Entity-Component-System (ECS), яка забезпечує високу продуктивність та масштабованість. Використання цих підходів дозволяє створювати гнучкі, добре організовані та легкі в підтримці ігри.

Отже, розробка архітектури гри є важливим етапом, який визначає організацію та взаємодію компонентів гри. Використання модульного підходу та шаблонів проектування допомагає забезпечити ефективну організацію коду, гнучкість та легкість підтримки, що дозволяє створювати масштабовані та якісні ігри.

4.4 Обмежений бюджет гри

Розробка ігор з обмеженими фінансовими ресурсами створює для розробників багато викликів. Проте, ці труднощі можна подолати за допомогою креативних рішень і правильного підходу.

Графіки та анімації

Однією з найбільших витрат у процесі створення гри є розробка якісної графіки та анімації. Виготовлення моделей, ассетів та анімацій вимагає дорогого програмного забезпечення, потужного обладнання та оплати праці фахівців. Наймання досвідчених художників та розробників може потребувати значних фінансових вкладень, що створює труднощі при недоліку бюджеті.

Маркетинг та реклама

Ефективне просування гри є критично важливим для її успіху на ринку. Проте, обмежений бюджет часто не дозволяє розробникам інвестувати у маркетинг та рекламу.

Обмеженість фінансів

Обмежений бюджет часто змушує знижувати якість гри, що може вплинути на графіку, звук та інші важливі аспекти. Крім того, недостатні фінансові ресурси обмежують можливості найму талановитих розробників та художників, що також може вплинути на кінцевий продукт.

Затримки в розробці

Створення гри з обмеженим бюджетом часто займає більше часу та зусиль, що може призвести до затримок у запуску гри та втрати потенційних прибутків. Тривалий час розробки також може негативно вплинути на моральний стан команди, що може ще більше затягнути процес.

Необхідність творчих рішень

Обмежений бюджет вимагає від розробників креативних та інноваційних рішень для зменшення витрат. Це може включати використання безкоштовного або відкритого програмного забезпечення, залучення фрілансерів для виконання окремих завдань та фокус на грі.

Рішення для зменшення витрат

Використання безкоштовного або відкритого програмного забезпечення

Для створення графічних елементів можна використовувати безкоштовні програми, такі як GIMP або Krita. Крім того, є безкоштовні бібліотеки звуків та музики, які можна використовувати для створення аудіоефектів та музики для гри.

Фокус на основних аспектах гри

Замість того, щоб витрачати ресурси на складні графічні елементи, варто сконцентруватися на розробці геймплею та ігрової механіки. Це дозволить створити гру, яка буде цікавою для гравців.

Привернення фрілансерів

Залучення фахівців-фрілансерів може бути ефективним способом зменшення витрат. Фрілансери можуть працювати над окремими аспектами проекту, що дозволяє знизити загальні витрати.

Планування проекту

Розробка плану проекту з урахуванням обмеженого бюджету має вирішальне значення. Контроль витрат через проектну документацію або спеціалізоване програмне забезпечення допоможе запобігти перевищенню встановленого бюджету.

Використання стратегій зниження ризиків

Виконання тестування та перевірки на всіх етапах розробки допоможе запобігти непередбачуваним проблемам. Це дасть змогу своєчасно виявляти та виправляти помилки, що знизить ризик затримок і додаткових витрат.

Зосередження на якості

Незважаючи на обмежені фінанси, важливо підтримувати високу якість продукту. Зосередьтеся на розробці простих, але якісних ігрових механік та графічних елементів замість складних і дорогих рішень. Це дозволить створити гру, яка буде цікавою та привабливою для гравців.

4.5 Вибір між двовимірною та тривимірною графікою у розробці гри

Рішення між використанням двовимірної та тривимірної графіки є важливим етапом у розробці гри. Підходи мають свої переваги та недоліки.

Переваги та недоліки двовимірної графіки

Менші вимоги до ресурсів

Двовимірна графіка вимагає менше ресурсів. Це особливо важливо для мобільних платформ та старих комп'ютерів. Зменшені ресурсні вимоги також знижують вартість розробки, оскільки немає потреби у складних тривимірних моделях та анімаціях.

Оптимізація

Створення двовимірної гри є менш складним процесом у порівнянні з розробкою тривимірної. Відсутність необхідності працювати з об'ємними об'єктами та складними світловими ефектами дозволяє швидше завершити проект. Це дає змогу розробникам зосередитися на інших важливих аспектах гри, таких як сюжет, геймплей, музика та звукові ефекти.

Унікальний стиль та атмосфера

Двовимірні ігри можуть мати унікальний візуальний стиль. Багато гравців цінують такі ігри за їх ностальгічну привабливість, що може створити особливий зв'язок з аудиторією. Візуальна привабливість двовимірних ігор дозволяє їм виділятися на ринку.

Переваги та недоліки тривимірної графіки

Реалізм та іммерсивність

Тривимірна графіка дозволяє гравцям досліджувати об'єкти з різних кутів, підвищуючи відчуття присутності та реалістичності. Реалістичне відображення об'єктів створює більш захоплюючий геймплей, значно підвищуючи загальне задоволення від гри. Такий підхід особливо підходить для жанрів, які потребують детального візуального представлення, таких як симулятори або шутери від першої особи.

Високі вимоги до ресурсів

Тривимірні ігри вимагають значно більше ресурсів, що може стати проблемою для менш потужних комп'ютерів або користувачів з обмеженим доступом до ресурсів. Це може обмежити потенційну аудиторію гри. Крім того, створення тривимірної графіки потребує більше ресурсів та спеціальних знань, що може збільшити витрати на розробку.

Складність розробки

Створення тривимірних моделей та ефектів освітлення є складним процесом, що вимагає більше часу та зусиль. Для цього потрібні висококваліфіковані фахівці, що збільшує бюджет проекту. Однак цей підходить для проектів, які потребують високого рівня деталізації та реалізму.

Тип гри та її мета

Для деяких жанрів, таких як шутери, тривимірна графіка може бути ефективною. Водночас інші жанри, як-от платформери або ретро-ігри, краще підходять для двовимірної графіки. Важливо враховувати мету гри: якщо основна мета полягає у створенні гри зі спрощеною графікою, то двовимірна гра може бути кращим варіантом.

Цільова аудиторія

Очікування гравців відіграють важливу роль у виборі графіки. Деякі гравці віддають перевагу двовимірним іграм за їх стиль та ностальгію, інші - тривимірним за їх реалістичність та іммерсивність. Важливо розуміти, чого саме очікує ваша цільова аудиторія від гри, яку ви розробляєте.

Таким чином, вибір між двовимірною та тривимірною графікою залежить від вимог і цілей розробника, а також від типу гри та очікувань цільової аудиторії.

4.6 Вимоги до гри

Функціональні вимоги

Гравець повинен мати можливість запускати нову гру.

В головному меню повинна бути кнопка "Нова гра", яка дозволяє гравцеві розпочати нову гру з початку. При натисканні на цю кнопку, гра повинна очищати всі попередні дані прогресу і починати з першої локації.

Гравець повинен мати можливість виходу з гри.

В меню гри повинна бути кнопка "Вихід", яка дозволяє гравцеві закрити гру. При натисканні цієї кнопки, гра повинна коректно зберегти прогрес гравця та завершити роботу програми.

Гравець повинен мати можливість керувати машиною (рухатися, взаємодіяти з об'єктами, виконувати дії).

Гравець повинен мати доступ до інтуїтивного інтерфейсу керування машиною, включаючи сенсорні кнопки або жестові управління для руху, прискорення, гальмування та виконання спеціальних дій (наприклад, використання бонусів або взаємодії з об'єктами на дорозі).

Машина повинна мати характеристики (швидкість, здоров'я), які можуть змінюватися в залежності від дій гравця.

Кожна машина повинна мати базові параметри, такі як максимальна швидкість, рівень здоров'я, витривалість. Ці параметри можуть змінюватися в залежності від того, як гравець проходить гру, збирає бонуси або зіштовхується з перешкодами.

Гра повинна містити різні локації, натхненні реальними місцями Закарпаття.

Локації повинні бути різноманітними та деталізованими, відтворюючи атмосферу реальних місць Закарпаття. Кожна локація повинна мати унікальний ландшафт, архітектуру та інші характерні риси.

Локації повинні містити різні перешкоди та виклики для гравця.

На кожній локації повинні бути перешкоди (наприклад, ями, каміння, вибоїни) та виклики (наприклад, складні маршрути, суперники), які потребують від гравця навичок для їх подолання.

Гра повинна мати фонову музику, яка змінюється в залежності від локації та ситуації.

Фонова музика повинна відповідати тематиці та атмосфері кожної локації. Також музика може змінюватися в залежності від інтенсивності ігрових подій, таких як гонки або екшн-сцени.

Гра повинна мати звукові ефекти для взаємодій з об'єктами.

При взаємодії з об'єктами (наприклад, збір бонусів, зіштовхування з перешкодами) повинні відтворюватися відповідні звукові ефекти, що підсилюють ігровий досвід.

Нефункціональні вимоги

Гра повинна завантажуватися не більше ніж за 5 секунд на сучасних пристроях Android. Всі дії гравця (натискання кнопок, рух персонажа) повинні мати відгук не більше ніж за 100 мілісекунд.

Це забезпечить швидкий і плавний ігровий процес без значних затримок, покращуючи користувацький досвід.

Гра повинна коректно відображатися на екранах з різною роздільною здатністю та співвідношенням сторін (від 16:9 до 21:9).

Інтерфейс та графіка гри повинні адаптуватися до різних розмірів та співвідношень екранів, щоб гра виглядала привабливо і зручно незалежно від пристрою.

Особисті дані користувачів (наприклад, прогрес гри) повинні бути захищені та зберігатися у безпечному вигляді.

Всі особисті дані користувачів повинні зберігатися з використанням сучасних методів шифрування та захисту, щоб забезпечити конфіденційність і безпеку.

Гра повинна мати вбудовану документацію або підказки, що пояснюють основні механіки та управління.

Для нових гравців повинна бути доступна інструкція або інтерактивні підказки, які пояснюють як керувати машиною, використовувати бонуси та проходити локації.

Гра повинна підтримувати пристрої з версіями Android не старішими за Android 7.0 (Nougat).

Це забезпечить сумісність гри з більшістю сучасних пристроїв, покриваючи значну частину цільової аудиторії.

Розробник повинен забезпечувати регулярні оновлення гри для виправлення помилок та додавання нового контенту.

Регулярні оновлення повинні виправляти виявлені помилки, покращувати продуктивність гри та додавати нові локації, машини, перешкоди і інші елементи, щоб підтримувати інтерес гравців.

5. РЕАЛІЗАЦІЯ ГРИ

5.1 Характеристики об'єкта

Налаштування характеристик об'єкта в Unity є ключовим аспектом у процесі розробки ігор, який визначає, як об'єкт виглядає, рухається і взаємодіє з іншими елементами гри. Цей процес вимагає глибокого розуміння різних компонентів і параметрів, які можуть бути застосовані до об'єкта, щоб забезпечити його коректну роботу і бажану поведінку в грі.

При створенні об'єкта в Unity кожен GameObject починається з базового компонента Transform, який визначає позицію, обертання і масштаб об'єкта в просторі. Transform є основою для всіх інших налаштувань, оскільки саме він визначає, де знаходиться об'єкт і як він орієнтований у світі гри. Налаштування Transform включає точне визначення координат об'єкта, його кутів обертання та

розмірів. Цей компонент є фундаментальним, оскільки кожен об'єкт у грі повинен мати свою унікальну позицію та розмір, щоб правильно взаємодіяти з іншими об'єктами і виглядати відповідно до задуму розробників.

Для відображення об'єкта на екрані використовується компонент Mesh Renderer разом з Mesh Filter. Mesh Filter визначає форму об'єкта, тоді як Mesh Renderer відповідає за відображення цієї форми, застосовуючи до неї матеріали і текстури. Матеріали визначають поверхневі властивості об'єкта, такі як колір, відбивна здатність, прозорість та інші оптичні характеристики.

Вибір правильного матеріалу є важливим кроком у створенні візуальної привабливості об'єкта, оскільки він впливає на те, як об'єкт виглядатиме під різними умовами освітлення і як він взаємодіятиме з іншими об'єктами та навколишнім середовищем.

Щоб надати об'єкту фізичні властивості, до нього додається компонент Rigidbody. Цей компонент дозволяє об'єкту підпадати під дію фізичних законів, таких як гравітація, сила тертя і інерція. Налаштування Rigidbody включає визначення маси об'єкта, його опору повітрю, можливості обертання і реакції на сили та імпульси.

Завдяки Rigidbody об'єкти можуть реалістично взаємодіяти з іншими фізичними об'єктами в грі, наприклад, при зіткненнях або під впливом сили вибуху. Це додає глибини та реалістичності геймплею, роблячи взаємодії між об'єктами більш правдоподібними і цікавими для гравців. На рис. 5.1 представлено знімок панелі налаштування об'єкта в Unity.

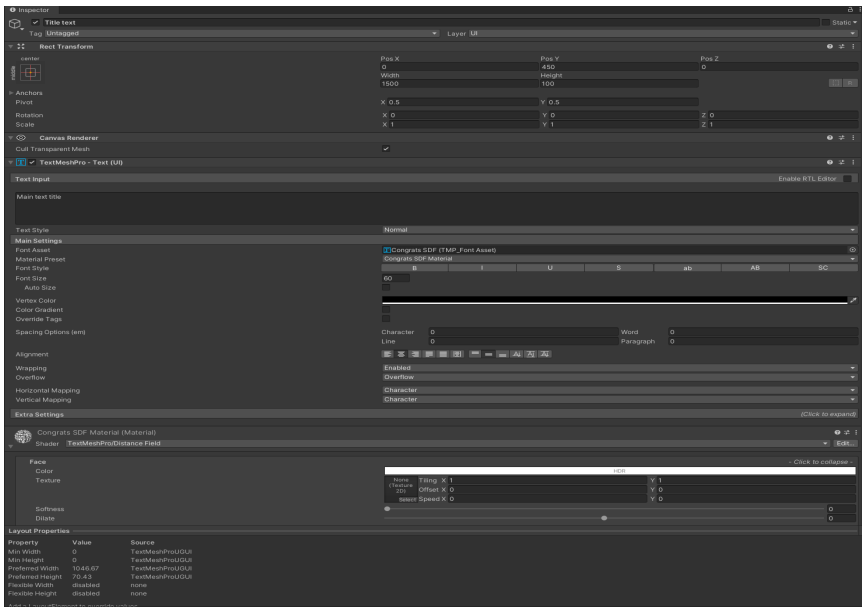


Рисунок 5.1 – Знімок панелі налаштування об'єкта в Unity

Компонент Collider визначає невидимі межі об'єкта, які використовуються для виявлення зіткнень з іншими об'єктами. Collider може мати різні форми, такі як Box Collider, Sphere Collider або Mesh Collider, кожна з яких підходить для різних типів об'єктів. Наприклад, Box Collider є оптимальним для об'єктів з прямокутною формою, тоді як Sphere Collider краще підходить для сферичних об'єктів. Важливо налаштувати розміри і форму Collider відповідно до форми об'єкта, щоб забезпечити точне виявлення зіткнень і коректну взаємодію з іншими об'єктами. Анімація об'єктів є частиною налаштування характеристик, оскільки дозволяє створювати динамічні рухи і переходи. Для цього використовується компонент Animator, який дозволяє створювати анімаційні кліпи і встановлювати їх на об'єкти. Animator дозволяє налаштовувати складні анімаційні переходи між різними станами об'єкта, такими як рух, атака, стрибок і т.д.

Це додає життєвості об'єктам, роблячи їх поведінку більш реалістичною і цікавою для гравців. Завдяки анімації можна створювати вражаючі візуальні ефекти, які підкреслюють важливі моменти геймплею і привертають увагу гравців до ключових подій у грі.

Для створення більш складних взаємодій між об'єктами і гравцем використовуються скрипти. Скрипти дозволяють визначати поведінку об'єктів, їх

реакцію на дії гравця і взаємодію з іншими об'єктами. За допомогою скриптів можна реалізувати різноманітні механіки, такі як керування рухом об'єкта, виявлення зіткнень, обробка введення від гравця і багато іншого. Написання скриптів вимагає знань мови програмування C# і розуміння основ програмування в Unity. Правильно написані скрипти допомагають створювати багатофункціональні ігрові об'єкти, які можуть реагувати на зміну умов і забезпечувати більш глибокий ігровий досвід.

Оптимізація характеристик об'єктів - це важливо, оскільки вона впливає на загальну продуктивність гри. Це включає налаштування рівня деталізації (Level of Detail, LOD), управління кількістю трикутників у моделі, використання відповідних текстурних форматів і оптимізацію анімаційних кліпів. Оптимізація допомагає знизити навантаження на систему, забезпечуючи плавний геймплей і високу якість графіки. Важливо враховувати, що занадто висока деталізація і складність об'єктів можуть призводити до зниження продуктивності, тому необхідно знайти баланс між якістю і продуктивністю.

LOD дозволяє змінювати деталі моделі в залежності від відстані до камери, що допомагає зменшити навантаження на систему при відображенні об'єктів, що знаходяться далеко від камери. Використання відповідних текстурних форматів і розмірів дозволяє знизити навантаження на пам'ять і графічний процесор. Компресія текстур і оптимізація розмірів файлів допомагає зменшити використання ресурсів, забезпечуючи більш плавний геймплей.

Таким чином, налаштування характеристик об'єкта в Unity є складним і важливим процесом, що включає вибір і налаштування компонентів, визначення фізичних і візуальних параметрів, створення анімацій і скриптів, а також оптимізацію для забезпечення високої продуктивності. Кожен з цих аспектів впливає на загальну якість і реалістичність гри, роблячи її більш захоплюючою і привабливою для гравців. Правильне налаштування характеристик допомагає створювати ігрові світи, які забезпечують чудовий досвід і дарують незабутні враження.

5.2 Фізика об'єктів у Unity

Фізика об'єктів у Unity є ключовим аспектом при створенні реалістичного та захоплюючого геймплею. Unity надає потужні інструменти для симуляції фізичних властивостей об'єктів у грі, що дозволяє створювати ефект реальності та взаємодії між об'єктами у віртуальному світі.

Однією з ключових особливостей фізики у Unity є можливість використання реалістичних параметрів, таких як маса, сила тяжіння, тертя та інші. Реалістична симуляція руху об'єктів дозволяє створювати ефекти, які легко відтворюються у реальному світі, такі як падіння, зіткнення та розбивання об'єктів.

Крім того, Unity пропонує широкий спектр колайдерів та фізичних матеріалів, які дозволяють точно визначати області взаємодії між об'єктами та їхні властивості при зіткненні. Це робить можливим створення складних фізичних сценаріїв, таких як симуляція руйнування будівель або реалістичне поведінка транспортних засобів.

Фізика об'єктів у Unity також підтримує анімацію об'єктів на основі фізичних законів, що дозволяє створювати плавні та реалістичні рухи об'єктів у грі. Це додає до геймплею більшу динаміку та іммерсивність, підвищуючи загальний рівень залучення гравця до віртуального світу.

Загалом, фізика об'єктів у Unity відкриває широкі можливості для реалістичного та захоплюючого геймплею, дозволяючи розробникам створювати ігри з високим рівнем іммерсії та взаємодії.

5.3 Типи колайдерів у Unity

У Unity існують різноманітні типи колайдерів, які використовуються для визначення областей взаємодії між об'єктами у грі. Основні типи колайдерів включають:

Box Collider (Колайдер прямокутника): Цей тип колайдеру використовується для визначення областей у формі прямокутника. Він часто використовується для об'єктів з простою геометрією, таких як стіни, підлоги та інші прямокутні об'єкти.

Sphere Collider (Колайдер сфери): Використовується для визначення областей у формі сфери. Він зазвичай використовується для об'єктів, які мають круглу або сферичну форму, наприклад, м'ячі або камені.

Capsule Collider (Колайдер капсули): Цей тип колайдеру використовується для визначення областей у формі капсули, що є комбінацією циліндра та двох половин сфери. Він часто використовується для моделювання об'єктів зі складною формою, наприклад, персонажів зі складною геометрією.

Mesh Collider (Колайдер меша): Використовується для визначення областей взаємодії на основі мешу об'єкта. Він найбільш гнучкий серед усіх типів колайдерів і дозволяє точно визначати області взаємодії для об'єктів будь-якої форми та складності.

Terrain Collider (Колайдер ландшафту): Використовується для взаємодії з тереном у грі. Цей тип колайдеру дозволяє об'єктам взаємодіяти з поверхнею терену, що створює реалістичний ефект контакту з землею.

Кожен з цих типів колайдерів має свої унікальні властивості та використання, які можна налаштовувати для відповідності конкретним потребам у грі. Використання відповідних типів колайдерів допомагає забезпечити точну та ефективну взаємодію між об'єктами у вашій грі. На рис. 5.2 представлено знімок панелі колайдеру в Unity.

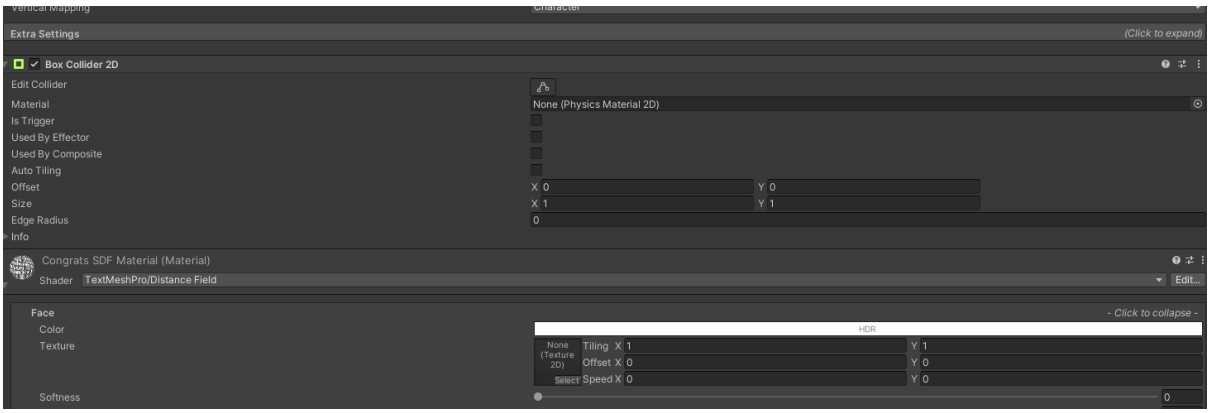


Рисунок 5.2 – Панелі колайдеру в Unity

5.4 Ігрові анімації

Ігрові анімації відіграють важливу роль у створенні живого та захоплюючого ігрового досвіду. На рис. 5.3 представлено знімок панелі анімації в Unity.

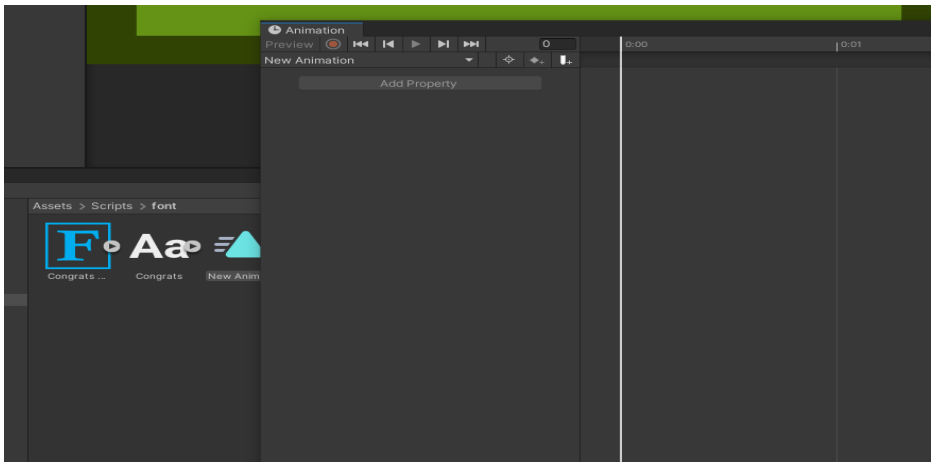


Рисунок 5.3 – Панелі анімації в Unity

Наведемо аспекти, які слід враховувати при роботі:

Персонажі та об'єкти: Анімації можуть надати життя персонажам та об'єктам у грі, роблячи їх рухи реалістичними та виразними. Це може включати анімацію ходьби, бігу, стрибків, атак, рухів ворогів та багато іншого.

Переходи між станами: Ігрові анімації допомагають плавно перемикатися між різними станами персонажів або об'єктів, такими як стояння, рух, атака, відпочинок тощо. Плавні переходи додають реалістичності та покращують іммерсію гравця.

Взаємодія з оточенням: Ігрові анімації можуть відтворювати реакції персонажів на оточуюче середовище, такі як реакція на зіткнення з перешкодами, зміни погодних умов або інтеракція з іншими об'єктами.

Ефекти та спеціальні прийоми: Ігрові анімації можуть використовуватися для створення різноманітних ефектів та спеціальних прийомів, таких як вибухи, магічні закляття, перетворення об'єктів тощо. Ці ефекти додають екшену та візуального блиску до гри.

Оптимізація та продуктивність: При роботі з ігровими анімаціями важливо враховувати оптимізацію та продуктивність. Ефективне використання анімаційних ресурсів може покращити продуктивність гри та забезпечити плавну роботу на різних пристроях. На рис. 5.4 представлено знімок панелі анімації в Unity.

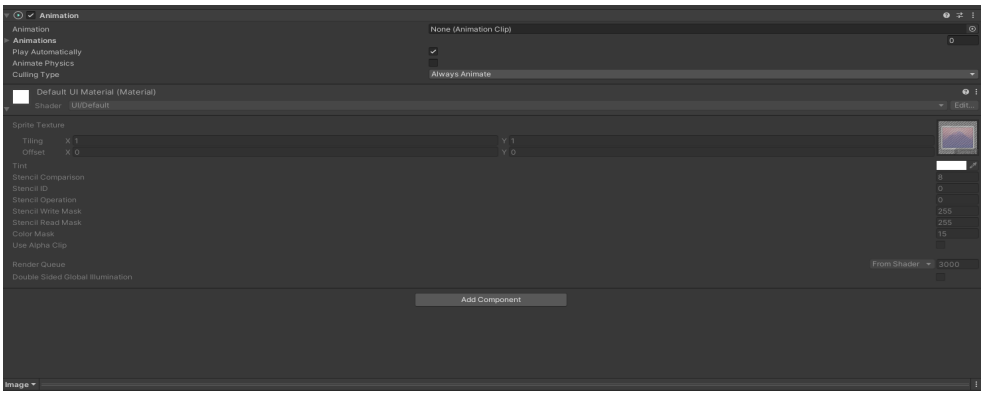


Рисунок 5.4 – Панель анімації в Unity

Інструменти для розробки: Unity та інші ігрові двигуни надають широкий набір інструментів для створення ігрових анімацій. Ці інструменти дозволяють створювати, редагувати та керувати анімаціями у зручний спосіб.

Загалом, ігрові анімації є важливим елементом розробки ігор, який додає реалізму, емоційної зв'язку та відчуття живості до геймплею. Їх правильне використання може підвищити якість та привабливість вашої гри для гравців.

5.5 Оптимізація проекту у Unity

Оптимізація проекту у Unity починається з розуміння середовища ігрового рушія та його можливостей. Unity надає розробникам безліч вбудованих аналізаторів та інструментів для аналізу продуктивності проекту. Основним компонентом для аналізу є Profiler, який дозволяє відстежувати продуктивність гри в режимі реального часу.

Profiler у Unity – це потужний інструмент, який допомагає розробникам виявити вузькі місця у продуктивності проекту. Profiler відстежує використання процесора, GPU, пам'яті та інших ресурсів у різних аспектах гри. Це включає обробку фізики, рендеринг, анімації, обробку скриптів та багато іншого.

На рис. 5.5 представлено панель навантаження CPU/GPU в Unity.

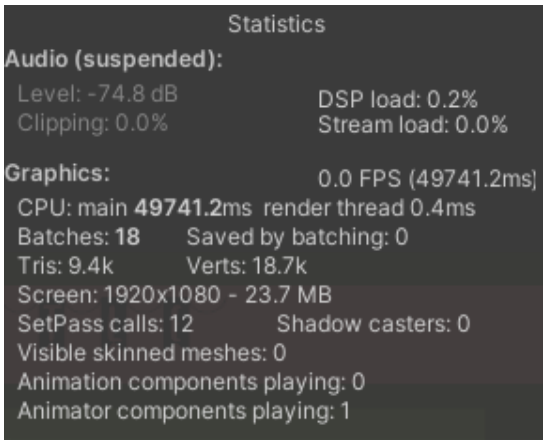


Рисунок 5.5 – Панель навантаження CPU/GPU в Unity

Основні функції Profiler:

CPU Usage: Дозволяє відстежувати використання процесора різними частинами гри, включаючи обробку фізики, рендеринг, анімації та інші обчислення.

GPU Usage: Відстежує використання графічного процесора, включаючи час, витрачений на рендеринг кадрів, обробку текстур та шейдерів.

Memory: Аналізує використання пам'яті у грі, включаючи розподіл пам'яті між різними об'єктами та ресурсами.

Rendering: Дозволяє відстежувати процес рендерингу, включаючи кількість викликів рендеру, використання текстур та шейдерів.

Physics: Аналізує обробку фізики у грі, включаючи зіткнення, обробку колайдерів та інші фізичні обчислення.

Scripts: Відстежує виконання скриптів, включаючи час, витрачений на виклики методів Update, FixedUpdate та інших функцій.

Основні підходи до оптимізації:

1. Зменшення кількості полігонів

Зменшення кількості полігонів у 3D-моделях допомагає знизити навантаження на GPU. Використання моделей з низькою полігональністю та застосування методів оптимізації, таких як LOD (Level of Detail), дозволяє зберегти високу продуктивність без втрати якості візуального представлення.

2. Оптимізація текстур

Використання текстур з низькою роздільною здатністю та застосування компресії текстур дозволяє знизити використання пам'яті та підвищити продуктивність. Варто також уникати зайвих текстур та використовувати текстурні атласи, щоб зменшити кількість викликів до GPU.

3. Оптимізація скриптів

Оптимізація скриптів включає зменшення кількості викликів методів Update та FixedUpdate, використання кешування для часто викликаних методів та змінних, а також уникання зайвих обчислень. Варто також звертати увагу на використання корутин та асинхронних операцій для розподілу навантаження.

4. Зниження кількості викликів рендеру

Зниження кількості викликів рендеру (draw calls) допомагає знизити навантаження на GPU. Використання методів об'єднання об'єктів (batching) та динамічного об'єднання дозволяє зменшити кількість викликів рендеру.

5. Використання оптимізованих шейдерів

Використання оптимізованих шейдерів дозволяє знизити навантаження на GPU. Варто уникати складних шейдерів та використовувати простіші

альтернативи, де це можливо. Також важливо знижувати кількість обробки пікселів та уникати зайвих операцій у шейдерах.

На рис. 5.6 представлено панель збірки проекту в Unity.

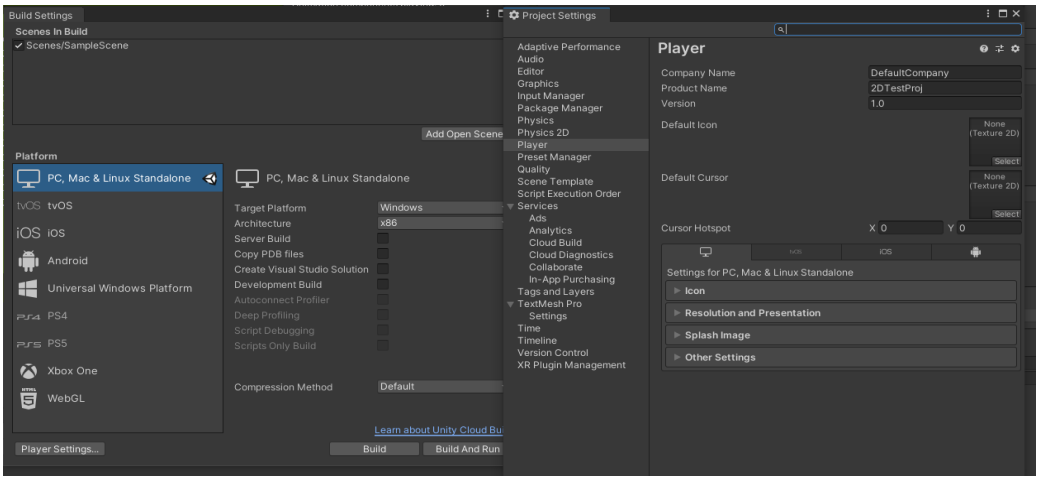


Рисунок 5.6 – Панель збірки проекту в Unity

Використання інструментів аналізу продуктивності

Окрім Profiler, Unity надає інші інструменти для аналізу продуктивності, такі як Frame Debugger, Memory Profiler та Physics Debugger. Ці інструменти допомагають детально аналізувати різні аспекти продуктивності гри та виявляти проблеми, які можуть впливати на загальну продуктивність.

Оптимізація проекту у Unity є важливим етапом розробки, що дозволяє забезпечити плавний та реалістичний геймплей. Використання Profiler та інших інструментів аналізу продуктивності допомагає виявити вузькі місця у проекті та застосувати ефективні методи оптимізації. Завдяки цьому можна досягти високої продуктивності та забезпечити гравцям захоплюючий ігровий досвід.

5.6 Базова архітектура проекту

Unity - це потужний ігровий рушій, який використовується для створення двовимірних і тривимірних ігор та інтерактивного контенту.

На рис. 5.7 представлено компонентно-орієнтована архітектура проекту.

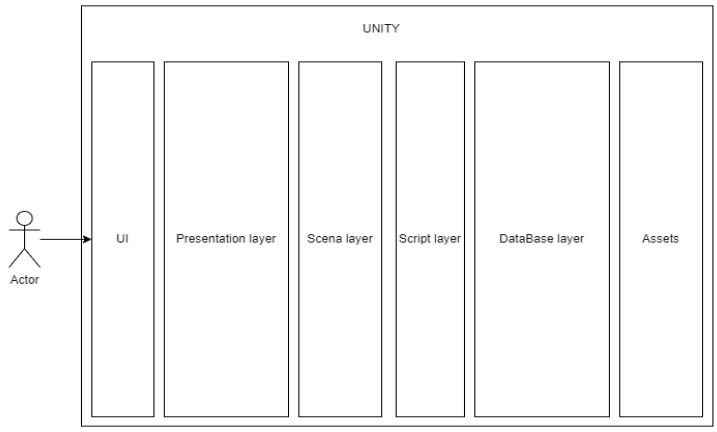


Рисунок 5.7 – Компонентно-орієнтована архітектура проекту

Ось основні аспекти його роботи зсередини:

Основні компоненти Unity:

Робоче середовище (Unity Editor): Unity Editor - це інтегроване середовище розробки (IDE), яке включає в себе всі необхідні інструменти для створення ігор. Воно дозволяє розробникам імпортувати графічні ресурси, створювати сцени, налаштовувати фізику та взаємодію об'єктів, а також писати код.

Сцени та об'єкти: Ігровий процес у Unity організовується в сценах. Сцена - це контейнер для всіх об'єктів, які взаємодіють один з одним. Об'єкти у сцені називаються GameObjects. Кожен GameObject може мати компоненти, які визначають його поведінку, вигляд та інші властивості.

Компоненти: Компоненти - це модулі, які додаються до GameObjects для надання їм певних властивостей та функціональності. Наприклад, компонент Transform визначає позицію, обертання та масштаб об'єкта, а компонент Rigidbody додає фізичні властивості, такі як маса та гравітація.

Скрипти: Поведінка об'єктів визначається за допомогою скриптів, написаних на мові програмування C#. Скрипти прив'язуються до GameObjects як компоненти і дозволяють розробникам створювати складні алгоритми взаємодії та логіку гри.

Фізичний рушій: Unity використовує фізичний рушій для обробки фізичних взаємодій між об'єктами. Це включає в себе зіткнення, сили, гравітацію та інші фізичні ефекти.

Рендеринг: Unity має потужну систему рендерингу, яка обробляє відображення графіки. Вона підтримує різні шейдери, освітлення та інші ефекти, що дозволяє створювати високоякісні візуальні елементи.

Архітектура проектів у Unity:

Для проектів у Unity підходить кілька типів архітектур, залежно від масштабу та вимог гри. Представимо декілька популярних підходів:

Компонентно-орієнтована архітектура: Unity за своєю природою є компонентно-орієнтованим. Це означає, що функціональність гри розподіляється між різними компонентами, які прив'язуються до GameObjects. Цей підхід дозволяє легко змінювати та розширювати гру, додаючи або змінюючи компоненти без впливу на інші частини системи.

Модель-Вид-Контролер (MVC): MVC - це шаблон проектування, який розділяє логіку програми на три основні частини: модель (дані та логіка), вид (користувацький інтерфейс) та контролер (взаємодія між моделлю та видом). Використання MVC допомагає організувати код, роблячи його більш структурованим та легшим для підтримки.

Модель-Вид-Видова Модель (MVVM): MVVM схожий на MVC, але додає ще один рівень абстракції - видову модель, яка служить посередником між моделлю та видом. Цей підхід корисний для складних проектів з великою кількістю взаємодій між користувачем та даними.

Система-ентити-компоненти (ECS): ECS - це архітектурний підхід, який зосереджується на відокремленні даних (систем) від їх поведінки (компонентів). Це дозволяє більш ефективно управляти великою кількістю об'єктів у грі, особливо коли йдеться про оптимізацію продуктивності.

Компонентно-орієнтовану архітектуру для розробки проекту, оскільки вона природньо вписується в структуру Unity, який побудований на основі GameObjects з різноманітними компонентами. Цей підхід забезпечує максимальне використання можливостей рушія, що значно спрощує процес розробки.

Компонентно-орієнтована архітектура відрізняється високою гнучкістю та масштабованістю, дозволяючи легко додавати нові функціональні можливості або змінювати існуючі компоненти без значних зусиль.

Це робить проект адаптивним до змін та розширень, що є важливим у динамічному середовищі розробки ігор. Крім того, така архітектура спрощує процес тестування та налагодження, оскільки кожен компонент можна тестувати окремо, що сприяє швидкому виявленню та виправленню помилок. Завдяки цим перевагам компонентно-орієнтована архітектура є оптимальним вибором для створення складних та якісних ігрових проектів.

5.7 Схема бази-даних

Unity зберігає дані у кількох різних форматах та місцях залежно від типу даних та їх призначення. Наведемо основні способи зберігання даних у Unity:

1. Сцени (Scenes)

Сцени в Unity є основними контейнерами для зберігання даних про об'єкти в грі. Сцена містить інформацію про всі GameObjects, їх компоненти та властивості. Файли сцен мають розширення .unity і зберігаються в папці проекту. Сцени можна завантажувати та вивантажувати динамічно під час виконання гри.

2. Prefab-файли

Prefabs - це шаблони об'єктів, які можна використовувати для створення екземплярів GameObjects з однаковими властивостями та компонентами. Вони зберігаються у файлах з розширенням .prefab. Prefabs дозволяють легко повторно використовувати та змінювати об'єкти в різних сценах.

3. ScriptableObject

ScriptableObject - це спеціальний клас Unity, що дозволяє зберігати дані в редакторі та використовувати їх у грі. Вони зберігаються як окремі файли у

форматі .asset. ScriptableObject ідеально підходять для зберігання налаштувань, конфігурацій або будь-яких інших даних, які потрібно використовувати у багатьох місцях проекту.

4. PlayerPrefs

PlayerPrefs - це проста система збереження, яка використовується для зберігання невеликих обсягів даних, таких як налаштування гри або прогрес гравця. Дані зберігаються у вигляді ключ-значення і можуть бути записані та прочитані з локального сховища пристрою. PlayerPrefs підходить для зберігання примітивних типів даних, таких як числа та рядки.

5. Файли

Unity дозволяє зберігати дані у власних файлах за допомогою стандартних методів вводу-виводу (I/O). Це може бути корисно для зберігання великих обсягів даних або складних структур даних, таких як JSON або XML. Такі файли можуть зберігатися в локальному сховищі пристрою або в хмарі.

6. Бази даних

Для більш складних або великих проектів можна використовувати зовнішні бази даних для зберігання даних. Це може бути SQLite для локальних баз даних або зовнішні серверні бази даних, такі як MySQL або Firebase, для мережевих ігор. Використання баз даних дозволяє ефективно управляти великими обсягами даних та забезпечує високий рівень доступності та надійності.

7. AssetBundles

AssetBundles - це пакети активів, які можуть бути завантажені динамічно під час виконання гри. Вони дозволяють зберігати та завантажувати ресурси, такі як текстури, моделі та аудіофайли, у вигляді окремих файлів. AssetBundles корисні для зменшення розміру початкового завантаження гри та динамічного завантаження контенту під час гри.

Таким чином, Unity забезпечує різноманітні способи зберігання даних, що дозволяє розробникам вибирати найоптимальніший метод залежно від специфіки їх проекту та вимог.

В роботі використано PlayerPrefs для зберігання даних у грі. Цей метод підходить для збереження невеликих обсягів даних, таких як налаштування гри або прогрес гравця, завдяки простій системі збереження у форматі ключ-значення.

Однак, для більш масштабного проекту та для зберігання більш складних структур даних, потрібно перейти до використання бази даних SQLite.

На рис. 5.8 представлено схема бази даних для казуальної гри гонки.

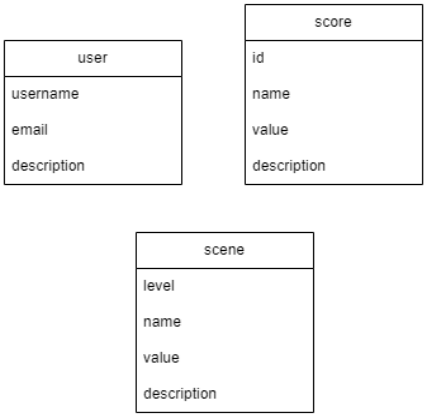


Рисунок 5.8 – Схема бази даних для казуальної гри гонки

Використання SQLite дозволить ефективно управляти великими обсягами даних, забезпечить кращу організацію інформації та підвищить продуктивність гри, особливо при роботі з великим обсягом даних або складними запитами. Перехід до SQLite надасть більшу гнучкість та надійність у зберіганні даних.

5.8 Взаємодія класів з Unity та діаграма класів

MonoBehaviour

Більшість класів у Unity успадковуються від базового класу MonoBehaviour. Цей клас дозволяє створювати компоненти, які можна прикріплювати до об'єктів у грі (GameObjects). Використання MonoBehaviour забезпечує доступ до важливих функцій і подій Unity, таких як ініціалізація об'єктів, обробка кадрів, фізичні обчислення тощо.

GameObjects і компоненти

У Unity всі об'єкти в сцені представлені як GameObject. Кожен GameObject може мати один або більше компонентів, які додають йому функціональність. Класи взаємодіють з цими компонентами, щоб створювати поведінку об'єктів у грі. Наприклад, класи можуть обробляти ввід від користувача, змінювати властивості об'єктів або викликати певні дії.

Сцени

Сцени в Unity є контейнерами для GameObjects. Класи взаємодіють з об'єктами у сцені через їх компоненти, додаючи, змінюючи або видаляючи їх у процесі виконання гри. Це включає створення нових об'єктів, завантаження або перемикання сцен, а також маніпуляцію існуючими об'єктами.

ScriptableObject

ScriptableObject – це спеціальний клас, який використовується для зберігання даних і конфігурацій, які можна легко використовувати та редагувати в Unity Editor. Це дозволяє зберігати дані поза об'єктами і забезпечує легкий доступ до них, що спрощує управління налаштуваннями гри та конфігураціями.

Інші взаємодії

Класи можуть взаємодіяти з різними системами Unity, такими як фізика, анімація, аудіо та інші. Наприклад, класи можуть використовувати фізичні компоненти для взаємодії з об'єктами, керувати анімаціями персонажів або обробляти звукові ефекти. Це забезпечує комплексну інтеграцію різних аспектів гри в єдину систему.

На рис. 5.9 представлено діаграма класів та взаємодія з Unity.

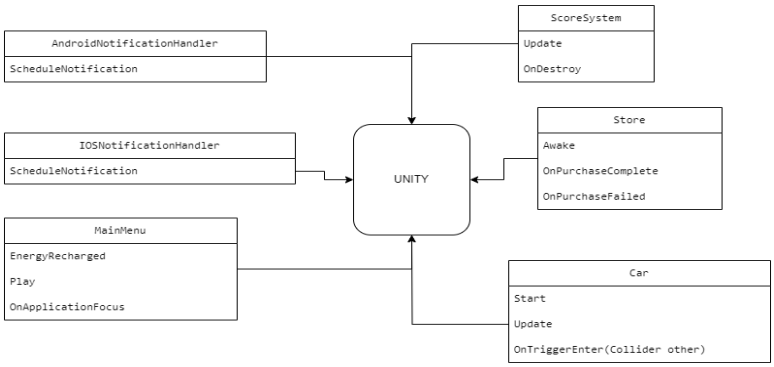


Рисунок 5.9 – Діаграма класів та взаємодія з Unity

Взаємодія класів з Unity забезпечує гнучкість та ефективність при розробці ігор, дозволяючи розробникам створювати складні ігрові механіки та взаємодії, які забезпечують захоплюючий ігровий досвід.

5.9 Алгоритми основної механіки гри-гонки

Представимо алгоритм основної механіки гри-гонки

1. Старт гонки

Алгоритм старту гонки включає підготовку гравця до початку руху та забезпечення початкового положення:

Встановити позицію гравця на стартовій лінії.

Запустити зворотний відлік (3, 2, 1, Старт).

Відкрити можливість керування автомобілем для гравця.

2. Управління автомобілем

Алгоритм управління автомобілем дозволяє гравцеві керувати транспортним засобом за допомогою сенсорного екрану:

Обробка введення від гравця (натискання на екран, нахил пристрою, тощо).

Зміна напрямку руху автомобіля відповідно до введення.

Регулювання швидкості автомобіля на основі натискання (прискорення або гальмування).

Застосування фізичних сил до автомобіля (тягар, тертя, тощо) для створення реалістичної поведінки.

3. Набір очок

Алгоритм набору очок відповідає за підрахунок та відображення очок, які гравець заробляє під час руху по трасі:

Підрахунок очок на основі пройденої відстані.

Додавання бонусних очок за збирання спеціальних предметів на трасі.

Відображення поточного рахунку на екрані.

4. Колізії та зіткнення

Алгоритм обробки колізій та зіткнень відповідає за взаємодію автомобіля гравця з об'єктами на трасі:

Виявлення колізій з перешкодами на трасі.

Віднімання очок або закінчення гри у випадку зіткнення з перешкодою.

Реалізація фізичних реакцій (зупинка автомобіля, відтворення звукових ефектів тощо) при зіткненні.

5. Перемога та програш

Алгоритм визначення перемоги та програшу визначає результат гри на основі дій гравця:

Виявлення зіткнень з перешкодами.

Закінчення гри при зіткненні з перешкодою та відображення повідомлення про програш.

Відображення підсумкового рахунку після закінчення гри.

Представлені алгоритми забезпечують основну механіку соло-гри гонки, створюючи захоплюючий та реалістичний ігровий процес. На рис. 5.9.1 представлено Блок-схема алгоритму основної механіки.

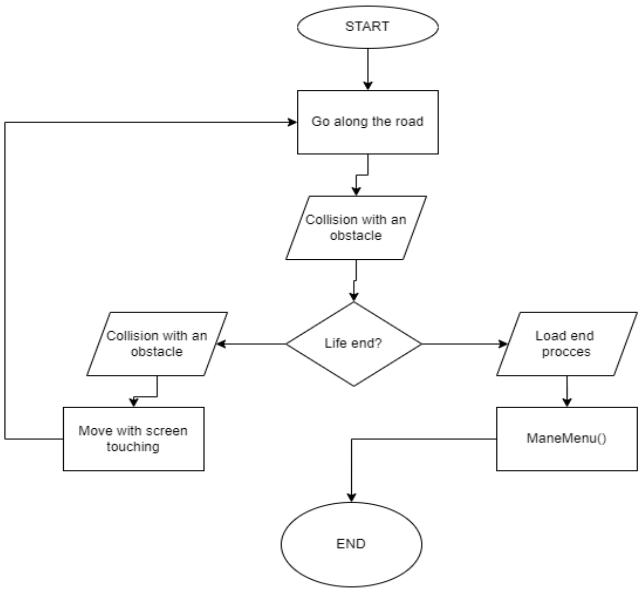


Рисунок 5.9.1 – Блок-схема алгоритму основної механіки

5.10 Код основних функцій гри

Клас NotificationHandler відповідає за налаштування та відправку локальних повідомлень на Android-пристроях. Він використовує Unity.Notifications.Android для створення та реєстрації каналу повідомлень, а також для планування відправлення повідомлення.

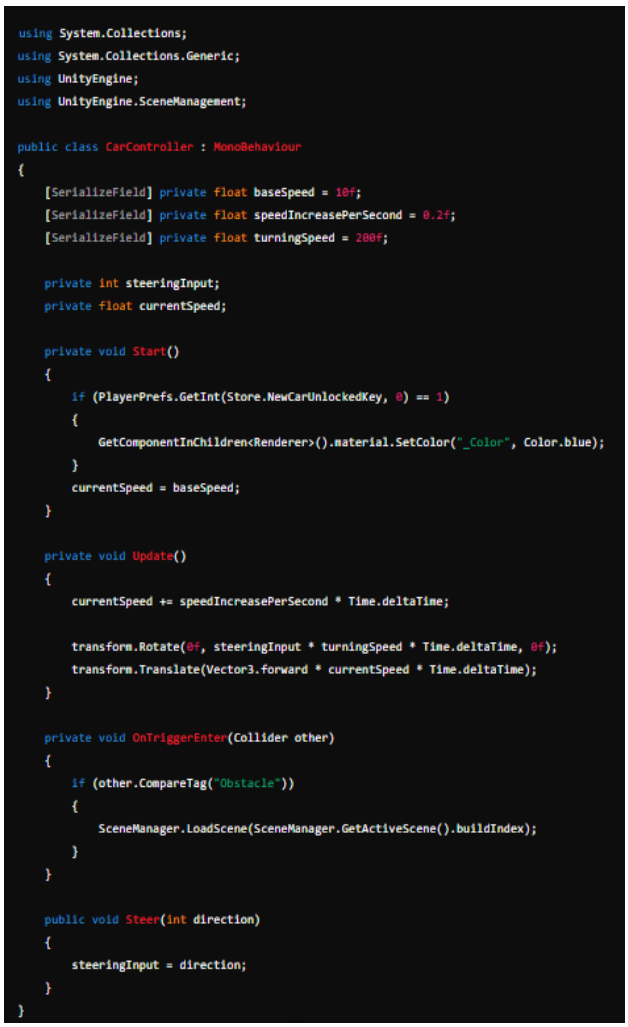
Метод ScheduleNotification приймає параметр DateTime, який визначає час, коли повідомлення буде відправлене. На рис. 5.9.2 представлено код класу - NotificationHandler.

```
using System;
using System.Collections.Generic;
using UnityEngine;
#if UNITY_ANDROID
using Unity.Notifications.Android;
#endif

public class NotificationHandler : MonoBehaviour
{
#if UNITY_ANDROID
    private const string NotificationChannelId = "notification_channel";

    public void ScheduleNotification(DateTime notificationTime)
    {
        var channel = new AndroidNotificationChannel
        {
            Id = NotificationChannelId,
            Name = "Notification Channel",
            Description = "Channel for game notifications",
            Importance = Importance.Default
        };

        AndroidNotificationCenter.RegisterNotificationChannel(channel);

        var notification = new AndroidNotification
        {
            Title = "Energy Recharged!",
            Text = "Your energy has recharged, come back to play again!",
            SmallIcon = "default",
            LargeIcon = "default",
            FireTime = notificationTime
        };

        AndroidNotificationCenter.SendNotification(notification, NotificationChannelId);
    }
#endif
}
```

Рисунок 5.9.2 – Код класу - NotificationHandler

Клас CarController відповідає за керування автомобілем у грі. Він обробляє рух автомобіля вперед, збільшення швидкості з часом, а також повороти в залежності від вводу гравця. У разі зіткнення з об'єктом, який має тег "Obstacle", гра перезавантажує поточну сцену.

Метод Steer встановлює напрямок керування автомобілем на основі вводу гравця. На рис. 5.9.3 представлено код класу - CarController.



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

public class CarController : MonoBehaviour
{
    [SerializeField] private float baseSpeed = 10f;
    [SerializeField] private float speedIncreasePerSecond = 0.2f;
    [SerializeField] private float turningSpeed = 200f;

    private int steeringInput;
    private float currentSpeed;

    private void Start()
    {
        if (PlayerPrefs.GetInt(Store.NewCarUnlockedKey, 0) == 1)
        {
            GetComponentInChildren<Renderer>().material.SetColor("_Color", Color.blue);
        }
        currentSpeed = baseSpeed;
    }

    private void Update()
    {
        currentSpeed += speedIncreasePerSecond * Time.deltaTime;

        transform.Rotate(0f, steeringInput * turningSpeed * Time.deltaTime, 0f);
        transform.Translate(Vector3.forward * currentSpeed * Time.deltaTime);
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Obstacle"))
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        }
    }

    public void Steer(int direction)
    {
        steeringInput = direction;
    }
}
```

Рисунок 5.9.3 – Код класу - CarController

Клас iOSNotificationHandler відповідає за налаштування та відправку локальних повідомлень на iOS-пристроях. Він використовує Unity.Notifications.iOS для створення та налаштування повідомлень.

Метод ScheduleNotification приймає параметр int minutes, який визначає час у хвилинах, через який буде відправлене повідомлення. Повідомлення включає заголовок, підзаголовок, тіло повідомлення та параметри відображення.

На рис. 5.9.4 представлено код класу - iOSNotificationHandler.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
#if UNITY_IOS
using Unity.Notifications.iOS;
#endif

public class iOSNotificationHandler : MonoBehaviour
{
#if UNITY_IOS
    public void ScheduleNotification(int minutes)
    {
        var notification = new iOSNotification
        {
            Title = "Energy Recharged",
            Subtitle = "Your energy has been recharged",
            Body = "Your energy has recharged, come back to play again!",
            ShowInForeground = true,
            ForegroundPresentationOption =
                PresentationOption.Alert | PresentationOption.Sound,
            CategoryIdentifier = "category_a",
            ThreadIdentifier = "thread1",
            Trigger = new iOSNotificationTimeIntervalTrigger
            {
                TimeInterval = new System.TimeSpan(0, minutes, 0),
                Repeats = false
            }
        };

        iOSNotificationCenter.ScheduleNotification(notification);
    }
#endif
}
```

Рисунок 5.9.4 – Код класу - iOSNotificationHandler

Клас MainMenu відповідає за управління головним меню гри, включаючи відображення рахунку, рівня енергії та обробку початку гри. Він перевіряє та відновлює енергію гравця при поверненні в додаток, планує відправлення повідомлень про відновлення енергії на Android та iOS платформах, і перезавантажує сцену гри при натисканні кнопки "Play". На рис. 5.9.5 представлено код класу - MainMenu перша частина.

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

public class MainMenu : MonoBehaviour
{
    [SerializeField] private TMP_Text highScoreText;
    [SerializeField] private TMP_Text energyText;
    [SerializeField] private Button playButton;
    [SerializeField] private AndroidNotificationHandler androidNotificationHandler;
    [SerializeField] private iOSNotificationHandler iosNotificationHandler;
    [SerializeField] private int maxEnergy;
    [SerializeField] private int energyRechargeDuration;

    private int energy;

    private const string EnergyKey = "Energy";
    private const string EnergyReadyKey = "EnergyReady";

    private void Start()
    {
        OnApplicationFocus(true);
    }
```

Рисунок 5.9.5 – Код класу - MainMenu перша частина

На рис. 5.9.6 представлено код класу - MainMenu друга частина.

```
    private void OnApplicationFocus(bool hasFocus)
    {
        if (!hasFocus) return;

        CancelInvoke();

        int highScore = PlayerPrefs.GetInt(ScoreSystem.HighScoreKey, 0);
        highScoreText.text = $"High Score: {highScore}";

        energy = PlayerPrefs.GetInt(EnergyKey, maxEnergy);

        if (energy == 0)
        {
            string energyReadyString = PlayerPrefs.GetString(EnergyReadyKey, string.Empty)
            if (string.IsNullOrEmpty(energyReadyString)) return;

            DateTime energyReady = DateTime.Parse(energyReadyString);
            if (DateTime.Now > energyReady)
            {
                energy = maxEnergy;
                PlayerPrefs.SetInt(EnergyKey, energy);
            }
            else
            {
                playButton.interactable = false;
                Invoke(nameof(EnergyRecharged), (energyReady - DateTime.Now).Seconds);
            }
        }

        energyText.text = $"Play ({energy})";
    }
```

Рисунок 5.9.6 – Код класу - MainMenu друга частина

На рис. 5.9.7 представлено код класу - MainMenu третя частина.

```
    private void EnergyRecharged()
    {
        playButton.interactable = true;
        energy = maxEnergy;
        PlayerPrefs.SetInt(EnergyKey, energy);
        energyText.text = $"Play ({energy})";
    }

    public void Play()
    {
        if (energy < 1) return;

        energy--;
        PlayerPrefs.SetInt(EnergyKey, energy);

        if (energy == 0)
        {
            DateTime energyReady = DateTime.Now.AddMinutes(energyRechargeDuration);
            PlayerPrefs.SetString(EnergyReadyKey, energyReady.ToString());

#if UNITY_ANDROID
            androidNotificationHandler.ScheduleNotification(energyReady);
#elif UNITY_IOS
            iosNotificationHandler.ScheduleNotification(energyRechargeDuration);
#endif
        }

        SceneManager.LoadScene(1);
    }
}
```

Рисунок 5.9.7 – Код класу - MainMenu третя частина

Клас ScoreSystem відповідає за обчислення та відображення рахунку гравця у грі. Він оновлює рахунок на кожному кадрі відповідно до часу, що минув, і множника рахунку. Після завершення гри, якщо поточний рахунок перевищує високий рахунок, він зберігає новий високий рахунок у PlayerPrefs. На рис. 5.9.8 представлено код класу - ScoreSystem.

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

public class ScoreSystem : MonoBehaviour
{
    [SerializeField] private TMP_Text scoreText;
    [SerializeField] private float scoreMultiplier;

    public const string HighScoreKey = "HighScore";

    private float score;

    // Update is called once per frame
    void Update()
    {
        score += Time.deltaTime * scoreMultiplier;
        scoreText.text = Mathf.FloorToInt(score).ToString();
    }

    private void OnDestroy()
    {
        int currentHighScore = PlayerPrefs.GetInt(HighScoreKey, 0);
        if (score > currentHighScore)
        {
            PlayerPrefs.SetInt(HighScoreKey, Mathf.FloorToInt(score));
        }
    }
}
```

Рисунок 5.9.8 – Код класу - ScoreSystem

Клас Store відповідає за обробку покупок у грі. Він використовує UnityEngine.Purchasing для управління покупками та відновленням покупок на iOS. У методі Awake перевіряється, чи платформа є iPhone, і якщо ні, кнопка відновлення покупок приховується.

Метод OnPurchaseComplete зберігає стан розблокованого автомобіля в PlayerPrefs, коли покупка завершена. Метод OnPurchaseFailed відображає повідомлення в журналі про невдалу покупку з причиною невдачі.

На рис. 5.9.9 представлено код класу - Store.

```
using UnityEngine;
using UnityEngine.Purchasing;

public class Store : MonoBehaviour
{
    [SerializeField] private GameObject restoreButton;

    private const string NewCarId = "com.gamedevtv.simpledriving.newcar";
    public const string NewCarUnlockedKey = "NewCarUnlocked";

    private void Awake()
    {
        if (Application.platform != RuntimePlatform.IPhonePlayer)
        {
            restoreButton.SetActive(false);
        }
    }

    public void OnPurchaseComplete(Product product)
    {
        if (product.definition.id == NewCarId)
        {
            PlayerPrefs.SetInt(NewCarUnlockedKey, 1);
        }
    }

    public void OnPurchaseFailed(Product product,
                                 PurchaseFailureReason reason)
    {
        Debug.LogWarning($"Failed to purchase product {product.definition.id} " +
                         $"because {reason}");
    }
}
```

Рисунок 5.9.9 – Код класу - Store

5.11 Рисунки тестової моделі

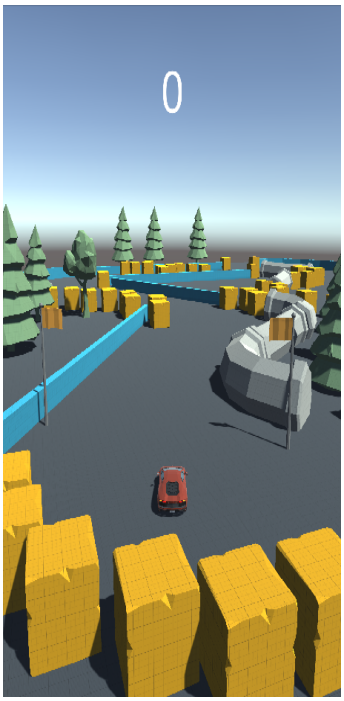


Рисунок 5.19 – Рисунок старту гри

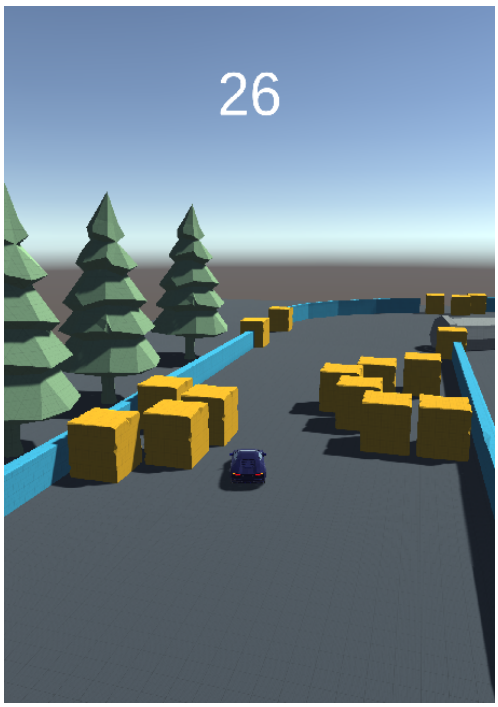


Рисунок 5.20 – Рисунок в момент гри

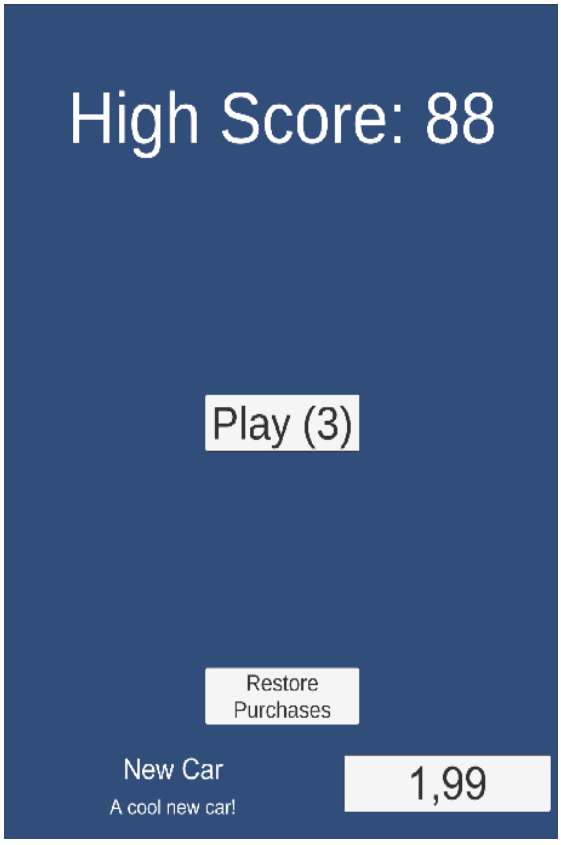


Рисунок 5.21 – Рисунок головного меню

6. ТЕСТУВАННЯ

6.1 Тестування ігор: Загальний огляд

Тестування ігор - це процес перевірки відеоігор на наявність помилок, дефектів і недоліків перед їх випуском на ринок. Основна мета тестування - забезпечити високу якість гри, зробити її цікавою, стабільною та безпомилковою для кінцевих користувачів. Тестування ігор включає різні види перевірок, кожна з яких має свої завдання і методики.

Основні етапи тестування ігор

Планування тестування

На цьому етапі створюється план тестування, який включає визначення цілей, підходів, ресурсів і графіку тестування. Планування також передбачає розробку тестових сценаріїв і підготовку необхідного обладнання та програмного забезпечення.

Функціональне тестування

Мета цього виду тестування - перевірити основну функціональність гри. Тестери перевіряють, чи всі ігрові механіки працюють належним чином, чи гравець може виконувати всі дії, передбачені дизайном гри, чи немає критичних помилок, що заважають проходженню гри.

Тестування продуктивності

Цей тип тестування оцінює, наскільки стабільно гра працює на різних пристроях і в різних умовах. Перевіряється швидкість завантаження, кількість кадрів за секунду (FPS), час відгуку гри на дії користувача та інші параметри продуктивності.

Тестування на сумісність

Мета цього тестування - переконатися, що гра коректно працює на різних пристроях, операційних системах, з різними конфігураціями апаратного забезпечення та роздільною здатністю екранів.

Тестування локалізації

Цей тип тестування перевіряє правильність перекладу гри на різні мови, відповідність культурним і регіональним особливостям. Тестери перевіряють текстові та голосові елементи, щоб уникнути помилок і неточностей в перекладі.

Регресійне тестування

Проводиться після внесення змін або виправлення помилок, щоб переконатися, що новий код не вплинув негативно на вже протестовані функції. Це дозволяє уникнути появи нових дефектів у вже стабільних частинах гри.

Бета-тестування

На цьому етапі гра доступна для обмеженої кількості користувачів (бета-тестерів), які грають в гру та надають зворотний зв'язок про її роботу, виявляють помилки та недоліки, що могли бути пропущені на попередніх етапах.

6.2 Тестові сценарії

Тестовий сценарій - це детальний план дій, який використовується для перевірки конкретного аспекту програмного забезпечення, у даному випадку, гри. Тестовий сценарій описує умови, вхідні дані, кроки, які необхідно виконати, очікувані результати та фактичні результати тестування. Основна мета тестового сценарію - забезпечити систематичне і всебічне тестування програмного забезпечення, щоб виявити можливі дефекти або помилки.

Тестовий сценарій для казуальної гри-гонки з перешкодами

Опис гри

Казуальна гра-гонки, де гравець керує автомобілем, що їде по трасах. З часом швидкість автомобіля збільшується, і на дорозі з'являються різні перешкоди, яких потрібно уникати.

Тестовий сценарій 1: Запуск нової гри

Назва сценарію: Запуск нової гри

Мета: Перевірити, чи можна успішно запустити нову гру з головного меню.

Попередні умови: Гравець знаходиться в головному меню гри.

Кроки:

Натиснути кнопку "Нова гра" в головному меню.

Переконатися, що починається нова гра, і гравець бачить стартовий екран траси.

Очікуваний результат: Нова гра починається без помилок, і гравець бачить стартовий екран траси.

Тестовий сценарій 2: Вихід з гри

Назва сценарію: Вихід з гри

Мета: Перевірити, чи можна успішно вийти з гри з головного меню.

Попередні умови: Гравець знаходиться в головному меню гри.

Кроки:

Натиснути кнопку "Вихід" в головному меню.

Переконатися, що гра завершує роботу без помилок і повертається до робочого столу пристрою.

Очікуваний результат: Гра завершує роботу без помилок, і пристрій повертається до робочого столу.

Тестовий сценарій 3: Керування машиною

Назва сценарію: Керування машиною

Мета: Перевірити, чи працює управління машиною під час гри.

Попередні умови: Гравець знаходиться на трасі в режимі гри.

Кроки:

Використовувати сенсорне керування для руху машини вліво і вправо.

Переконатися, що машина реагує на команди без затримок.

Очікуваний результат: Машина рухається вліво і вправо відповідно до дій гравця без затримок.

Тестовий сценарій 4: Зіткнення з перешкодами

Назва сценарію: Зіткнення з перешкодами

Мета: Перевірити реакцію гри на зіткнення машини з перешкодами.

Попередні умови: Гравець знаходиться на трасі в режимі гри.

Кроки:

Вести машину так, щоб вона зіткнулася з перешкодою на трасі.

Переконатися, що при зіткненні гра реагує правильно (наприклад, зменшується рівень здоров'я машини або гра закінчується).

Очікуваний результат: Гра правильно реагує на зіткнення з перешкодою (наприклад, зменшення здоров'я або завершення гри).

Тестовий сценарій 5: Збільшення швидкості з часом

Назва сценарію: Збільшення швидкості з часом

Мета: Перевірити, чи збільшується швидкість машини з часом під час гри.

Попередні умови: Гравець знаходиться на трасі в режимі гри.

Кроки:

Продовжувати грати в гру протягом певного часу (наприклад, 5 хвилин).

Спостерігати за швидкістю машини протягом цього часу.

Очікуваний результат: Швидкість машини поступово збільшується з часом.

Тестовий сценарій 6: Відображення різних локацій

Назва сценарію: Відображення різних локацій

Мета: Перевірити, чи правильно відображаються різні локації у грі.

Попередні умови: Гравець пройшов кілька рівнів гри.

Кроки:

Пройти перший рівень гри.

Переконатися, що після проходження першого рівня гра переходить до наступної локації.

Перевірити, чи відображення нової локації коректне.

Очікуваний результат: Після проходження рівня гра переходить до нової локації, яка правильно відображається.

Тестовий сценарій 7: Фонова музика та звукові ефекти

Назва сценарію: Фонова музика та звукові ефекти

Мета: Перевірити, чи правильно відтворюється фонова музика та звукові ефекти у грі.

Попередні умови: Гравець знаходиться на трасі в режимі гри.

Кроки:

Почати нову гру і спостерігати за фоновою музикою.

Взаємодіяти з різними об'єктами на трасі (наприклад, збирати бонуси, зіштовхуватися з перешкодами).

Переконатися, що звукові ефекти відповідають діям гравця.

Очікуваний результат: Фонова музика змінюється в залежності від локації, а звукові ефекти відтворюються коректно при взаємодії з об'єктами.

ВИСНОВКИ

Було проведено аналіз існуючих мобільних ігор у жанрі гонок, що дозволило визначити ключові особливості та механіки, які користуються популярністю серед гравців. Виявлено, що успішні ігри характеризуються високим рівнем деталізації графіки, реалістичною фізикою руху транспортних засобів, можливістю кастомізації автомобілів, різноманітністю трас і сценаріїв, а також інтеграцією соціальних функцій, таких як мультиплеєрні режими та таблиці лідерів. Ці елементи були враховані при розробці концепту нашої гри.

Розроблено детальний концепт гри, в якому описані сюжет, ігрові механіки, візуальний стиль, а також інтеграція культурних та історичних елементів Закарпаття. Сюжет гри відображає культурні та історичні аспекти регіону, а ігрові механіки включають різноманітні завдання та квести, що пов'язані з подорожами по Закарпаттю. Візуальний стиль гри створений з урахуванням характерних архітектурних та природних елементів регіону, що додає унікальності ігровому середовищу.

Створено технічне завдання, яке включає детальний опис вимог до гри, функціональності та графічних ресурсів. Вимоги включають оптимізацію гри для різних мобільних пристроїв, забезпечення плавної роботи навіть на менш потужних девайсах, а також розробку інтуїтивно зрозумілого інтерфейсу користувача. Опис функціональності охоплює системи керування транспортними засобами, реалізацію квестів та місій, а також інтеграцію культурних елементів Закарпаття в ігровий процес.

Спроектовано та реалізовано основні механіки гри, включаючи системи квестів, ігрові локації, персонажів та інтерфейс користувача. Використання сетингу подорожі Закарпаттям дозволило створити унікальні ігрові локації, що відображають реальні місця регіону. Ігрові персонажі розроблені з урахуванням місцевої культури та історії, що додає автентичності грі. Інтерфейс користувача забезпечує зручне керування та навігацію по грі, що підвищує загальний комфорт для гравців.

Після завершення розробки проведено ретельне тестування гри для виявлення та виправлення помилок. Використано як автоматизовані, так і ручні методи тестування, що дозволило знайти і виправити баги та забезпечити стабільність роботи гри на різних пристроях. Завдяки цьому вдалося досягти високої якості кінцевого продукту, який відповідає очікуванням користувачів та забезпечує позитивний ігровий досвід.

ПЕРЕЛІК ПОСИЛАНЬ

1. Top 10 game engines, 2022. [Електронний ресурс] – Ресурс доступу: https://ulab.sumdu.edu.ua/uk/10-najkrashhih-igrovih-rushiiv

2. Getting Started with Unity: Colliders and UnityScript, 2019. [Електронний ресурс] – Ресурс доступу: https://coderlessons.com/articles/veb-razrabotkaarticles/nachalo-raboty-s-unity-kollaidery-i-unityscript

3. Mobile game development market in Ukraine, 2021. [Електронний ресурс] – Ресурс доступу: https://it.comments.ua/ua/article/games/rinok-rozrobkimobilnih-igor-v-ukraini-vbivati-chas-uzhe-ne-tak-veselo-yak-ranishe690462.html

4. Офіційний сайт гри “Katana Zero” [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.katanazero.com

5. Endless runner [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: http://jerrymomoda.com/analysis-endless-runners/

6. Game Engine [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://en.wikipedia.org/wiki/Game_engine

7. Казуальні ігри [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.prostranstvo.media/uk/kazualni-igry-vdavana-prostota-na- mezhi-vysokogo-mystecztva/

8. Офіційний сайт рушія Unity [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://docs.unity3d.com

9. Офіційний сайт рушія Unreal Engine [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.unrealengine.com/

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА МОБІЛЬНОЇ ГРИ З СЕТІНГОМ В СТИЛІ ПОДОРОЖІ ЗАКАРПАТТЯМ ДЛЯ ANDROID МОВОЮ C# З ВИКОРИСТАННЯМ РУШІЯ UNITY

Виконав студент 5 курсу
групи ППЗ-51
Куліш Андрій Андрійович
Керівник роботи
К.т.н, доцент кафедри ІПЗ Довженко Теймур Павлович

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

●**Мета роботи -** покращення геймплею мобільної гри жанру гонки за рахунок використання сетінгу в стилі подорожі Закарпаттям для Android мовою C# з використанням рушія Unity.

●**Об’єкт дослідження -** геймплей мобільної гри жанру гонки з сетінгом в стилі подорожі.

●**Предмет дослідження -** мобільна гра жанру гонки з сетінгом в стилі подорожі.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати мобільні ігри в жанрі гонок та визначити основні особливості та механіки, які реалізуться в них.
2. Розробити концепт гри, де будуть описані сюжет, ігрові механіки, візуальний стиль, а також ідеї для інтеграції культурних та історичних елементів Закарпаття.
3. Розробити технічне завдання, що включає вимоги до гри, опис функціональності та графічних ресурсів.
4. Спроектувати та реалізувати механіки гри, системи квестів, ігрові локації, ігрові персонажі, а також інтерфейс користувача для гри жанру гонок з використанням сетінгу в стилі подорожі Закарпаттям.
5. Провести тестування гри, виявити та виправити помилки.

3

АНАЛІЗ АНАЛОГІВ

	Need for Speed: NL	Real Racing 3	Simple Driving
Підтримка iOS та Android	+	+	+
Внутрішньоігрові покупки	+	-	+
Система накопичення енергії	-	+	+
Українська тематика	-	-	+
Нескінченна гра	-	-	+
Інтеграція з Google Auth	+	-	-
Підтримка API App Store/Apple Store	+	+	+

4

КОНЦЕПТ ГРИ

- Гравець керує автомобілем, доторкаючись до екрану, щоб змінювати його траєкторію, об'їжджати перешкоди та збирати бонуси.
- Ігровий рівень постійно рухається вперед, якщо гравець не встигає уникати перешкод або виїжджає за межі екрану, він програє.
- Після збору бонусів гравець отримує очки.
- Швидкість руху рівня та частота появи перешкод збільшуються з часом, що підвищує складність гри.
- Якщо гравець врізається у перешкоду, він програє, його прогрес втрачається і гра починається з початку.
- Чим далі гравець просунувся у грі і чим більше бонусів він зібрав, тим вищий його рекорд.
- Гравець може змагатися з іншими гравцями у встановленні рекорду або покращувати власні результати.

5

ВИМОГИ ДО ІГРОВОГО КОНТЕНТУ

Функціональні вимоги

1. Гравець повинен мати можливість запускати нову гру з головного меню.
2. Гравець повинен мати можливість виходу з гри через головне меню.
3. Гравець повинен мати можливість керувати машиною (рухатися, взаємодіяти з об'єктами, виконувати дії) за допомогою сенсорного екрану.
4. Машина повинна мати змінні характеристики (швидкість, здоров'я), які можуть змінюватися в залежності від дій гравця.
5. Локації повинні містити різні перешкоди та виклики для гравця.
6. Гра повинна мати фонову музику, яка змінюється в залежності від локації та ситуації.

Нефункціональні вимоги

1. Гра повинна завантажуватися не більше ніж за 5 секунд на сучасних пристроях Android та iOS. Всі дії гравця (натискання кнопок, рух персонажа) повинні мати відгук не більше ніж за 100 мілісекунд.
2. Гра повинна коректно відображатися на екранах з різною роздільною здатністю та співвідношенням сторін (від 16:9 до 21:9).
3. Особисті дані користувачів (наприклад, прогрес гри) повинні бути захищені та зберігатися у безпечному вигляді.
4. Гра повинна мати вбудовану документацію або підказки, що пояснюють основні механіки та управління.
5. Гра повинна підтримувати пристрої з версіями Android не старішими за Android 7.0 (Nougat) та iOS не старішими за iOS 12.
6. Розробник повинен забезпечувати регулярні оновлення гри для виправлення помилок та додавання нового контенту.
7. Гра повинна містити різні локації, натхненні реальними місцями Закарпаття.
8. Гра повинна мати звукові ефекти для взаємодій з об'єктами.

6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

DBeaver

Unity

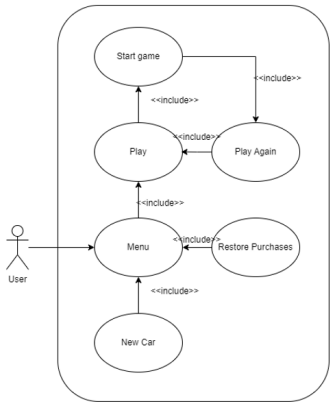
GitLab

git

IDEA

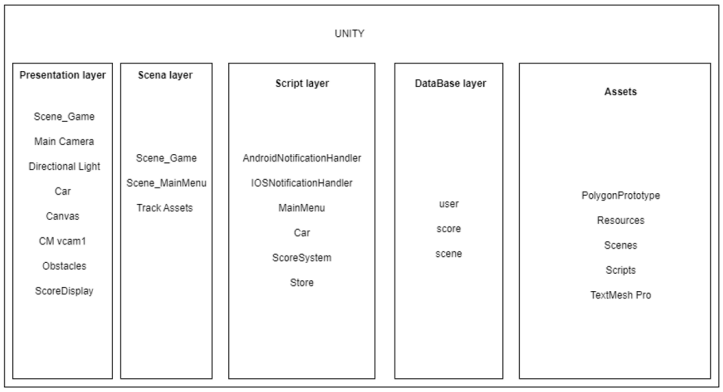
7

ДІАГРАМА ПРЕЦЕДЕНТІВ



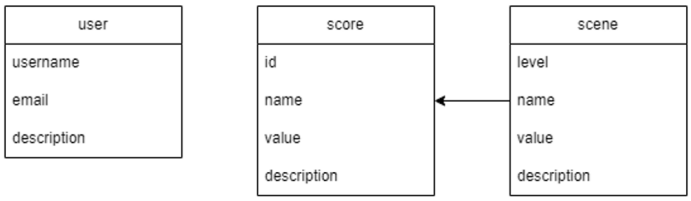
8

СХЕМА СТРУКТУРИ СИСТЕМИ



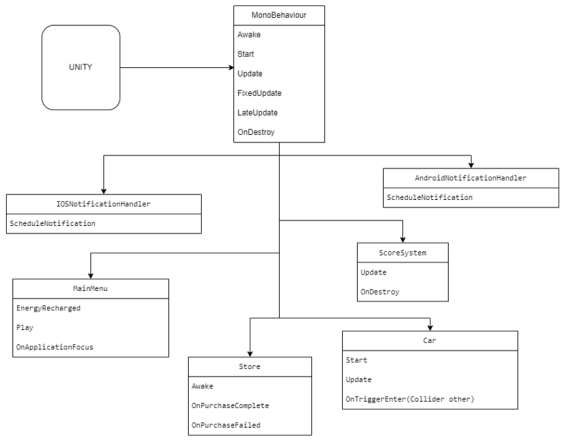
9

СХЕМА БАЗИ ДАНИХ



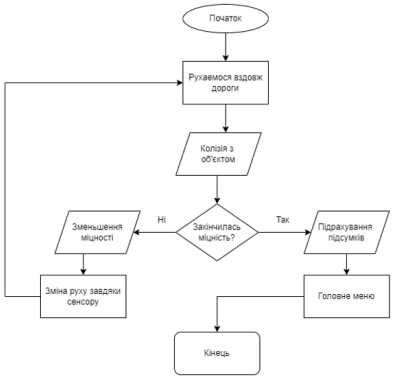
10

ДІАГРАМА КЛАСІВ



11

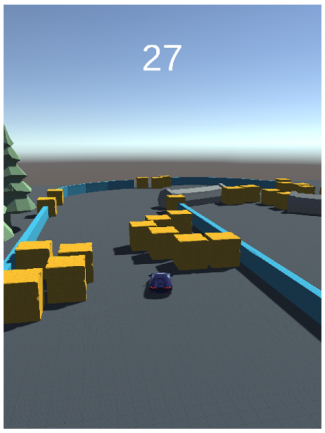
БЛОК-СХЕМА АЛГОРИТМУ ОСНОВНОЇ МЕХАНІКИ ГРИ



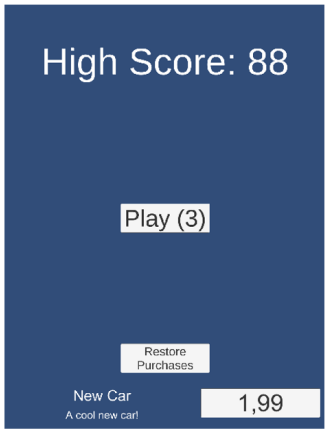
12

ЕКРАННІ ФОРМИ

Перша локація



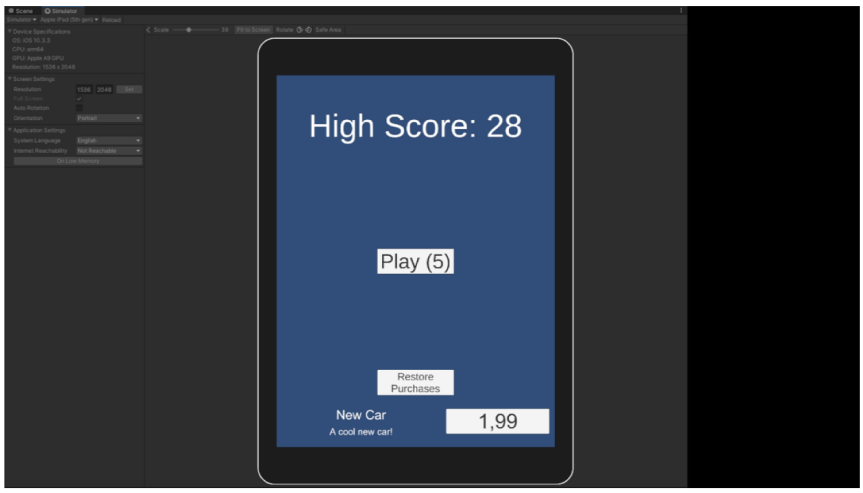
Нова машинка



Головне меню

13

ВІДЕО ГРИ



14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Куліш А.А., Аверічев І.М. Розробка застосунку для тестування rest-запитів зовнішніх систем на мові java. Застосування програмного забезпечення в ІКТ. Всеукраїнська науково-технічна конференція. Збірник тез. 24.04.24, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 91-92
2. Куліш А.А., Аверічев І.М. Розробка мобільної гри для android мовою C#. Сучасні інтелектуальні інформаційні технології в науці та освіті. IV Всеукраїнська науково-практична конференція. Збірник тез. 15.05.24, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. подано до друку.

ВИСНОВКИ

1. Проаналізовано мобільні ігри в жанрі гонок, визначено основні особливості та механіки.
2. Розроблено концепт гри з описом сюжету, ігрових механік, візуального стилю та інтеграції елементів Закарпаття.
3. Сформульовано технічне завдання, яке включає вимоги до гри, опис функціональності та графічних ресурсів.
4. Спроектовано та реалізовано механіки гри, системи перешкод, ігрові локації, машинки та інтерфейс користувача.
5. Проведено тестування гри, виявлено та виправлено помилки.

ДЯКУЮ ЗА УВАГУ!