МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-06

Дипломний проект

здобувача освіти денної форми навчання
РП.06.05.000.ДП

ГОЛОВАЧЕНКА ЛЕОНІДА ДМИТРОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітня програма: «**Розробка програмного забезпечення»**
Група: **4РП-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку.

Проектний матеріал складається з пояснювальної записки на 76 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Головаченко Л.Д.)
Керівник _____ (Джабраілов Д.В.)

Консультанти:
з економічної частини _____ (Копайгородська Т.Г.)
з охорони праці _____ (Чорновол Н.І.)
з дотримання вимог ЄСКД _____ (Петрашова В.І.)
старший консультант _____ (Кунуп Т.В.)

До захисту допущений
Голова циклової комісії _____ (Кривченко Ю.В.)
Завідувач відділення _____ (Скорнякова О.В.)

Захист «22» 06 2023 р. Протокол ДКК № 1
Оцінка ДКК 5 (відмінно)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти *Головаченко Леонід Дмитрович*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку*

затверджена наказом по коледжу від "17" жовтня 2023р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) 09.06.2023

3. Вихідні данні до проекту (роботи)

1. Використання ігрового двигуна Unity;

2. Використання компонентів та фізичних можливостей ігрового двигуна Unity;

3. Мова програмування C#, бібліотеки ігрового двигуна Unity;

4. Реалізувати працюючу фізичну модель автомобілю в 3D-просторі для комп'ютерної гри за допомогою можливостей ігрового двигуна Unity та мови програмування C#.

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

Опис предметної області. Визначення впливу якості реалізації фізики у комп'ютерних іграх. Огляд існуючих рішень. Вибір інструментарію для реалізації фізичної моделі автомобілю у 3D-просторі. Способи реалізації фізичної моделі. Побудова алгоритму роботи скриптів фізичної моделі автомобілю. Розробка фізичної моделі автомобіля у 3D-просторі. Імплементація системи керування автомобілем. Тестування моделі. Економічна частина. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Особливості використання фізики машин у комп'ютерній ігровій індустрії; Особливості ігрового двигуна Unity; Пояснення алгоритму роботи фізичної моделі автомобілю у проекті; Етапи реалізації фізичної моделі проекту; Система керуванням автомобілем у 3D-просторі; Скріншоти результату розробки.

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	*Джабраілов Д.В.*		
розділ			
2.Екон. частина	*Копайгородська Т.Г.*		
3.Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачі завдання _____

Керівник *Джабраілов Д.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	*Вступ. Постановка мети та задач проектування*	*19.05.2023*	Виконано
2.	*Аналіз предметної галузі*	*20.05.2023*	Виконано
3.	*Огляд існуючих рішень*	*21.05.2023*	Виконано
4.	*Вибір інструментарію для реалізації фізичної моделі в*	*22.05.2023*	Виконано
	3D-просторі		
5.	*Способи реалізації фізичної моделі*	*23.05.2023*	Виконано
6.	*Побудова алгоритму роботи скриптів фізичної моделі*	*24.05.2023*	Виконано
	автомобілю		
7.	*Розробка фізичної моделі автомобілю у 3D-просторі*	*25.05.2023*	Виконано
8.	*Імплементація системи керування автомобілем*	*28.05.2023*	Виконано
9.	*Тестування фізичної моделі на працездатність*	*31.05.2023*	Виконано
10.	*Виконання Економічної частини*	*02.06.2023*	Виконано
11.	*Виконання Охорони праці*	*04.06.2023*	Виконано
12.	*Підготовка матеріалів до захисту*	*06.06.2023*	Виконано
13.	*Попередній малий захист*	*12.06.2023*	Виконано
14.	*Проведення захисту дипломного проекту*	*19.06.2023*	Виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 06. 05 000. 00 ДП	Дипломний проект		
А4			РП 06. 05 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 06. 05 000. 00 ДП		
Розробив		Головаченко Л. Д.			Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку	Літ.	Аркуш 5 / Аркушів
Перевірив		Джабраілов Д.В.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4РП-06	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Головною темою даного дипломного проекту є «Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку», і сам процес реалізації цих компонентів відіграє важливу роль у створенні високоякісного цифрового ігрового продукту.

Реалізація фізичної моделі автомобілю включає в себе вивчення та застосування фізичних законів, таких як кінематика, динаміка, механіка та інші принципи, що забезпечують правильну поведінку автомобіля в ігровому середовищі. Для досягнення високої якості фізичної моделі можуть бути використані спеціалізовані фізичні двигуни та власноручно розроблені функціональні схеми.

Окрім реалізації фізичної моделі автомобілю, важливим аспектом дипломного проекту є розробка ігрової структури та стратегій. Це включає в себе проектування рівнів, завдань, системи управління гравцем, штучного інтелекту для ігрових противників та інших компонентів, які створюють цікавий ігровий досвід.

Для досягнення успіху в розробці комп'ютерних ігор також важливо бути в курсі останніх тенденцій та технологій у галузі. Це дозволить використовувати новітні програмні рішення та технології для створення інноваційних ігрових продуктів, що відповідають сучасним вимогам та очікуванням гравців.

Отже, виконання теми дипломного проекту з фокусом на реалізацію фізичної моделі автомобілю в 3D-просторі для ігрового додатку дозволить глибше вивчити принципи розробки цифрових ігор, розвивати навички використання сучасних програмних рішень та технологій, а також створювати інноваційні ігрові рішення, відповідні сучасним тенденціям у галузі.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Опис предметної області

Тема дипломного проектування включає в собі компоненти, пов'язані з розробкою цифрових ігор. Ця галузь включає такі елементи:

- Аналіз жанрів цифрових ігор;
- Дослідження технологій для розробки цифрових ігор;
- Визначення основних компонентів геймплею цифрової гри;
- Визначення різних способів реалізації цих компонентів;

Аналіз жанрів цифрових ігор є важливим етапом дипломного проектування в галузі розробки ігор. Він передбачає детальне вивчення та оцінку різних жанрів, що дозволяє краще розуміти їх особливості та характеристики. Нижче наведено огляд деяких популярних жанрів цифрових ігор:

1. Екшн (Action): Жанр, в якому головним елементом є швидкість, рефлекси та бойові вміння гравця. Типові представники цього жанру - шутери (FPS або TPS), файтинги, платформери.

2. Рольова гра (RPG): Жанр, що зосереджений на розвитку персонажа та його участі в уявному світі. Включає розширення навичок, збору предметів та виконання квестів. Існують піджанри, такі як вестерн-рпг, фентезі-рпг, постапокаліптичні рпг та інші.

3. Пригодницька гра (Adventure): Жанр, в якому головними елементами є розв'язування головоломок, взаємодія з персонажами та дослідження ігрового світу. Включає піджанри, такі як точка-й-клацай, пригоди від першої особи, інтерактивні розповіді та інші.

4. Стратегія (Strategy): Жанр, що вимагає стратегічного мислення та планування дій. Включає в себе піджанри, такі як воєнна стратегія, виробнича стратегія, глобальна стратегія та інші.

5. Головоломки (Puzzle): Жанр, в якому гравець вирішує різноманітні головоломки, логічні завдання та складні загадки. Включає піджанри, такі як

головоломки з фізикою, математичні головоломки, логічні ігри та інші.

6. Симулятор (Simulation): Жанр, що намагається відтворити певний аспект реального життя або ситуацію. Включає піджанри, такі як авіасимулятори, автомобільні симулятори, симулятори управління містом та інші.

На Рисунку 1.1 зображені ігрові жанри.

Рисунок 1.1. Функціональна схема жанрів ігор

При реалізації дипломного проекту з фокусом на "Реалізацію фізичної моделі автомобілю", тема проекту визначає жанр гри і відкриває можливість вибрати елементи ігрової логіки та компоненти інтерфейсу. У випадку із цією темою жанр гри є Симулятор, а піджанр є автомобільний симулятор.

У процесі розробки технологій для створення ігор, основна увага приділяється вибору відповідного ігрового двигуна та набору розробницьких інструментів. Існує декілька підходів до вибору цих компонентів, проте найважливішим аспектом є вибір самого ігрового двигуна для конкретної гри. Раніше це було складною задачею, оскільки доступних ігрових двигунів, якими

можна користуватися звичайному користувачеві, було мало. Сьогодні ми маємо широкий вибір рішень з різними видами ліцензій та технологій.

Передбачення попередніх факторів дозволяє визначити можливості реалізації, враховуючи обраний жанр та технології. Поняття ігрових механік описує елементи геймплею, в які залучається гравець - які дії виконує гравець, яким способом і з якою метою вони виконуються, та являються важливими компонентами ігрового процесу. симулятори надають можливість гравцям отримати віртуальний досвід управління та взаємодії з реальними або уявними сценаріями.

В нашому випадку, симулятор автомобіля дозволить нам створити ігровий досвід, де гравець зможе відчути, як це керувати автомобілем в різних умовах та на різних типах доріг. Обравши жанр симулятора, ми зможемо сконцентруватися на розробці реалістичної фізичної моделі автомобіля, розширених можливостей керування та взаємодії з навколишнім середовищем. Ми зможемо дослідити різні аспекти автомобільної симуляції, такі як реалістична фізика руху, моделювання поведінки автомобіля на дорозі та іншими факторами, що впливають на керування автомобілем.

1.2 Визначення впливу якості реалізації фізики у комп’ютерних іграх

Якість реалізації фізики у комп'ютерних іграх має надзвичайно великий вплив на відчуття реалізму та іммерсивність гравців. Фізика в іграх стосується моделювання руху об'єктів, їх взаємодії, сили тяжіння, колізій, руйнування та інших фізичних аспектів. Якщо фізичні процеси відтворюються достовірно та реалістично, гравці мають можливість більш глибоко зануритися в гру та відчути її автентичність.

Високоякісна реалізація фізики дозволяє об'єктам у грі рухатися так, як очікувалося б у реальному світі. Наприклад, предмети можуть падати, рухатися згідно зі своєю масою та формою, взаємодіяти з іншими об'єктами з урахуванням законів фізики. Колізії між об'єктами моделюються точно, що дозволяє їм

взаємодіяти так, як очікувалося б у реальному світі, при цьому уникнення артефактів та неправдоподібних ситуацій.

Крім того, реалістична фізика дозволяє відтворювати руйнування об'єктів у грі. Наприклад, будівлі можуть розбиватися на частини з урахуванням їхньої конструкції та матеріалу (рисунок 1.2). Це створює враження реального руйнування та додає до ігрового процесу більшої глибини.

Рисунок 1.2. Приклад реалізації руйнування у грі Teardown

Врахування фізичних аспектів також важливо для досягнення балансу та геймплейної справедливості. Наприклад, сила тяжіння може впливати на рух персонажа, стрілянину або поведінку транспортних засобів. Це допомагає забезпечити реалістичність та сприятливі умови для стратегічних рішень у грі.

Крім відчуття реалізму іммерсивності, якість фізики в комп'ютерних іграх також може впливати на геймплейні можливості та стратегію гравців. Наприклад, якщо гравець має можливість використовувати фізичні закони у свою користь, це може відкрити нові стратегічні можливості та способи розв'язання завдань у грі.

Реалістична фізика також може мати вплив на мультиплеєрні взаємодії. Якщо гравці взаємодіють у спільному віртуальному просторі, точне

моделювання фізичних взаємодій може забезпечити справедливі умови для всіх учасників. Це дозволяє створити реалістичну ігрову екосистему, де фізика впливає на взаємодію гравців і створює вирішення ситуацій на основі реальних фізичних законів.

Зростання потужності обчислювальної технології дозволяє реалізувати все більш складні та реалістичні фізичні моделі у комп'ютерних іграх. Розробники використовують фізичні двигуни та алгоритми, щоб створювати вражаючі сцени, ефекти та фізичні симуляції.

Проте, важливо знати, що баланс між реалізмом фізики та геймплеєм може бути складним завданням. Деякі ігри можуть вибирати менш реалістичну фізику, щоб надати гравцям більш аркадний та експериментальний досвід. В інших випадках, особливо у симуляторах чи реалістичних шутерах, точна фізика може бути вирішальною для досягнення максимального рівня іммерсії.

Загалом, якість фізики у комп'ютерних іграх є суттєвим фактором, що визначає реалізм гри та задоволення гравців. Вдосконалення фізичного моделювання дозволяє створювати більш іммерсивні та захоплюючі враження, роблячи геймплей ще більш привабливим для широкої аудиторії.

1.3 Огляд існуючих рішень

Ігри з фізичною моделлю автомобіля надають гравцям можливість відчути реалістичне керування автомобілем, відтворюючи рухи, пов'язані з фізикою транспортного засобу. Це створює більш іммерсивний досвід і дозволяє гравцям взаємодіяти з автомобілем, як з реальним об'єктом. Ось кілька ігор, які використовують фізичну модель автомобіля:

1. "BeamNG.drive" (рисунок 1.3) використовує реалістичну фізичну модель для моделювання автомобілів. Ця модель включає детальне моделювання деформації автомобіля, зчеплення з дорогою та поведінку автомобіля в різних умовах. Основні алгоритми, використовувані для моделювання фізики автомобілів у грі, включають:

1) Колізії: Автомобілі в грі моделюються як складні тверді об'єкти, які

можуть взаємодіяти з іншими об'єктами у світі гри. Для обробки колізій використовуються алгоритми детекції колізій, такі як алгоритми опуклої оболонки чи AABB (Axis-Aligned Bounding Box), які дозволяють визначити, коли автомобіль зіткнеться з іншим об'єктом.

2) Динаміка руху: Автомобілі в грі підкоряються законам фізики, таким як закони Ньютона. Для моделювання динаміки руху використовуються чисельні методи, такі як методи Ейлера або методи Рунге-Кутта. Ці методи дозволяють визначити зміну швидкості та положення автомобіля в кожен момент часу на основі зовнішніх сил, що діють на нього.

3) Вплив зовнішніх сил: У грі враховуються різні зовнішні сили, які можуть впливати на автомобіль, наприклад, сила тяжіння, опір повітря, сили тертя та інші. Ці сили враховуються в обчисленнях руху автомобіля, що дозволяє моделювати його реалістичну поведінку на дорозі.

Усі ці алгоритми спільно працюють для створення реалістичної фізичної моделі автомобілів у грі "BeamNG.drive", що дозволяє користувачам отримати реалістичний досвід взаємодії з автомобілями та їх оточенням.

Рисунок 1.3. Приклад реалізації гри BeamNG.drive

2. У грі "Dirt Rally" (рисунок 1.4) використовуються реалістичні

моделі зчеплення з дорогою, впливу поверхонь та погодних умов на керування автомобілем, а також ефектів погоди на поведінку автомобіля. Деякі з алгоритмів та ефектів, що використовуються в грі, можуть включати:

1) Модель зчеплення з дорогою: Гра використовує складну фізичну модель для визначення зчеплення автомобіля з різними поверхнями дороги, такими як асфальт, гравій, бруд або сніг. Це включає моделювання коефіцієнта тертя, який впливає на рух автомобіля і залежить від типу поверхні та стану шин.

2) Вплив поверхонь на керування: Різні типи поверхонь в грі мають відмінні властивості, які впливають на керування автомобілем. Наприклад, на асфальті автомобіль може мати краще тримання і точніше реагувати на керування, але на бруді або снігу зчеплення може бути гірше, що призводить до ковзання та меншої контролюємості.

3) Ефекти погоди: Погодні умови в грі "Dirt Rally" можуть суттєво впливати на поведінку автомобіля. Наприклад, дощ або сніг можуть змінювати стан поверхні дороги, роблячи її слизькою та сковзкою. Вітер може також впливати на аеродинаміку автомобіля, особливо на великі швидкості.

4) Знос шин: Гра враховує знос шин під час гонок. Шини можуть зношуватись залежно від умов їх використання, таких як тип поверхні, стиль керування та тривалість гонки. Знос шин впливає на їх тримання та ефективність, що може вплинути на керування автомобілем.

Використання цих моделей та ефектів допомагає створити реалістичний гоночний досвід у грі "Dirt Rally", де гравці можуть відчути вплив різних поверхонь, погодних умов та зносу шин на керування автомобілем.

Рисунок 1.4. Приклад реалізації гри Dirt Rally

3. Серія гоночних симуляторів "Project CARS" (рисунок 1.5) використовує докладне моделювання фізики автомобілів та зовнішніх впливів для досягнення реалістичного досвіду гонок. Основні аспекти їх фізичної моделі включають:

1) Моделювання зовнішніх впливів: У грі "Project CARS" враховуються різні зовнішні впливи, які можуть впливати на поведінку автомобіля. Це включає погодні умови, такі як дощ, сніг, туман, які можуть змінювати стан поверхні дороги і зчеплення з нею. Також моделюються різні дорожні умови, включаючи нерівності, кочки, асфальтові шви, що можуть впливати на керування та стабільність автомобіля.

2) Реалістична фізична модель автомобіля: Гра використовує складну фізичну модель, яка враховує різні аспекти поведінки автомобіля. Це включає моделювання динаміки руху, зчеплення шин з поверхнею дороги, аеродинаміки, підвіски та інших факторів. Модель також враховує розподіл ваги, твердість підвіски, налаштування автомобіля та його властивості.

3) Вплив погоди: У грі "Project CARS" погодні умови мають значний вплив на гонку. Моделюються ефекти дощу, снігу, вітру, температури і

вологості, які можуть впливати на зчеплення, тертя, видимість та інші аспекти гонки. Гравці можуть відчути зміни у керуванні та поведінці автомобіля в залежності від погодних умов.

4) Моделювання пошкоджень: "Project CARS" також має систему мод

елювання пошкоджень автомобілів. Внаслідок зіткнень або контактів з перешкодами автомобілі можуть отримувати реалістичні пошкодження, що впливають на їхню функціональність та керованість.

Всі ці аспекти моделюються за допомогою реалістичних алгоритмів та фізичних принципів, що дозволяють гравцям відчути реалістичну поведінку автомобілів під час різних умов у грі "Project CARS".

Ці ігри пропонують високий рівень реалізму фізичної моделі автомобіля, дозволяючи гравцям насолоджуватися автогонками з максимальною автентичністю. Кожна з них має свої особливості та підходи до моделювання фізики автомобіля, тому можна вибрати ту, яка найбільше підходить вам за власними уподобаннями.

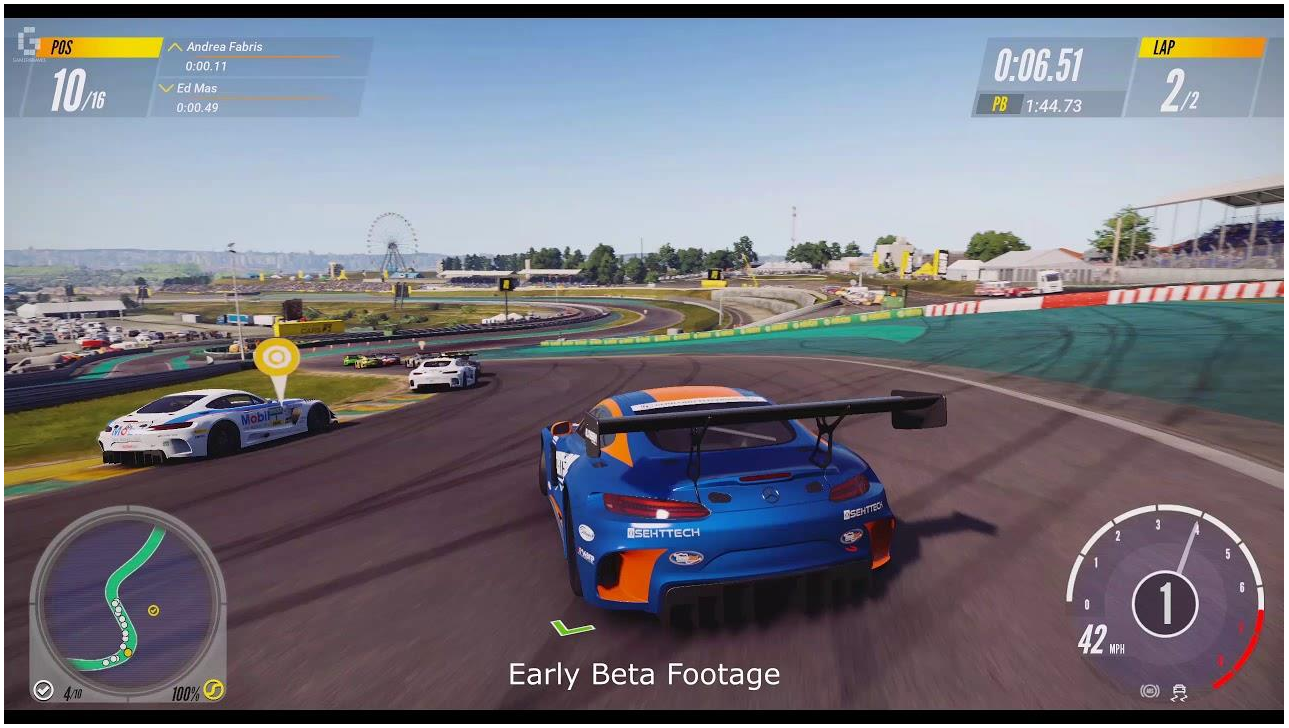


Рисунок 1.5. Приклад реалізації гри Project CARS

1.4 Вибір інструментарію для реалізації фізичної моделі автомобілю у 3D-просторі

У процесі розробки дипломного проекту був обраний ігровий двигун, який стане основою для створення гри. Ігровий двигун включає в себе набір програмних функцій, методів, класів та бібліотек, що спільно створюють умови для розробки цифрової гри. Раніше ігрові двигуни були поділені на дві категорії: ігрові двигуни, які відповідали за механіку гри, способи взаємодії між елементами та логіку гри, і фізичні двигуни, які відтворювали фізичні закони у ігрових проектах. Раніше часто казали, що гра А була розроблена на основі гри В з використанням певного ігрового двигуна. З розвитком інструментів та технологій для розробки цифрових ігор, ігрові двигуни еволюціонували і в наш час об'єднують в собі як ігрову механіку, так і фізичну взаємодію об'єктів на сцені.

На сьогоднішній день на ринку існує декілька популярних ігрових двигунів, які використовуються для розробки цифрових ігор. Ось декілька з них:

Unity є одним з найпопулярніших ігрових двигунів у галузі. Він надає широкі можливості для розробки ігор на різних платформах, включаючи комп'ютери, консолі, мобільні пристрої та віртуальну реальність.

Unreal Engine, розроблений Epic Games, є потужним ігровим двигуном, який використовується для створення якісних графічно-зорових ефектів та фотореалістичних ігрових світів. Він також підтримує різні платформи та надає доступ до додаткових інструментів для розробки.

CryEngine є іншим потужним ігровим двигуном, який відомий своїми вражаючими графічними можливостями. Він широко використовується для створення великих відкритих світів та ігрових проектів з високою деталізацією.

Godot Engine є відкритим джерелом ігровим двигуном, який набуває все більшої популярності. Він пропонує широкий спектр функцій та інструментів для розробки ігор різного жанру та масштабу.

На Рисунку 1.6 можна побачити середовище розробки популярних

ігрових двигунів на сьогоднішній день.

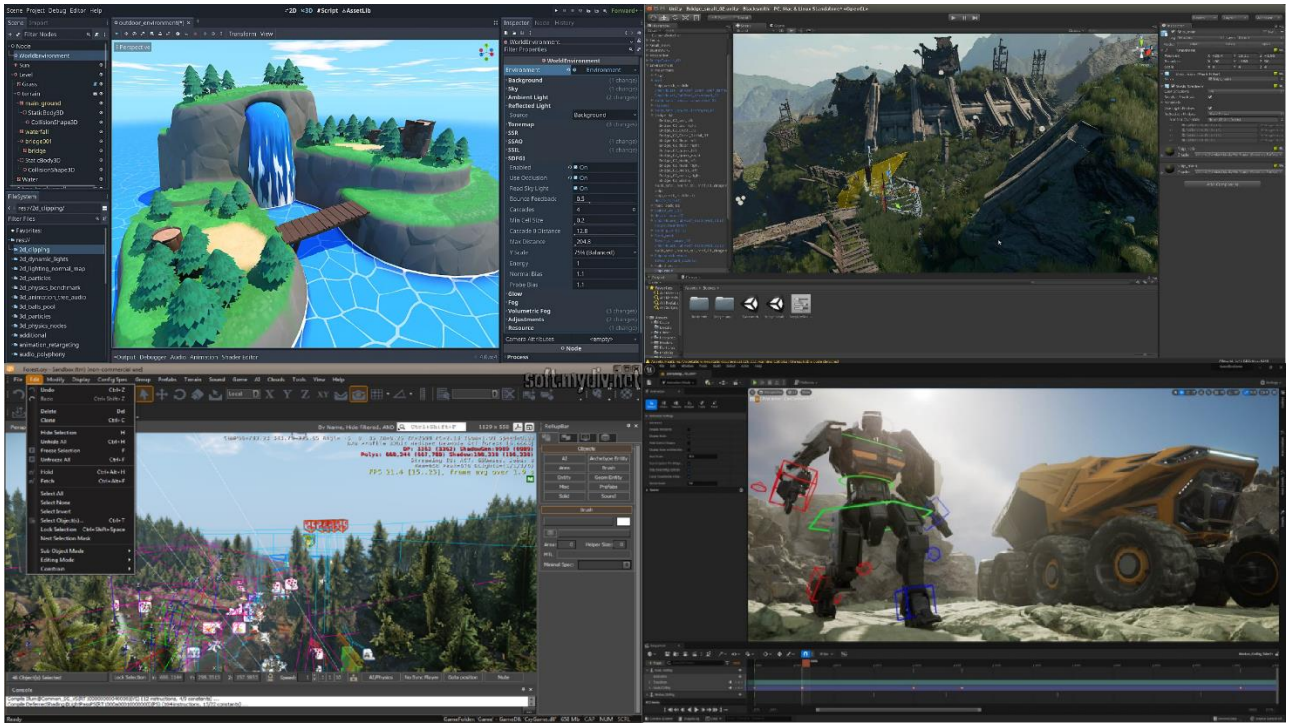


Рисунок 1.6. Середовища розробки ігрових двигунів

Для Дипломного проектування ми обмежуємося розглядом Unity та Unreal Engine з декількох причин. По-перше, обидва ці двигуни мають безкоштовні версії, що робить їх доступними навіть з обмеженим бюджетом. Деякі інші ігрові двигуни можуть вимагати плату за ліцензію або мають обмеження на використання безкоштовної версії.

По-друге, Unity та Unreal Engine є широко використовуваними та популярними двигунами у галузі розробки ігор. Це означає, що вони мають велику спільноту розробників, доступ до багатої документації, навчальних матеріалів та онлайн-ресурсів. Це дозволяє швидше навчитися використовувати ці двигуни і знайти відповіді на будь-які питання або проблеми, що виникають під час роботи.

Нарешті, враховуючи обмежений досвід роботи з ігровими двигунами, Unity та Unreal Engine є більш доступними відносно навчання та освоєння. Ці двигуни надають інтуїтивно зрозумілий інтерфейс та велику кількість вбудованих інструментів, що полегшують роботу з ними навіть новачкам у галузі геймдеву.

Unreal Engine - це потужний ігровий двигун та середовище розробки (IDE), яке використовується для створення вражаючих ігрових проектів для різних платформ, включаючи комп'ютери, консолі, мобільні пристрої та віртуальну реальність. На Рисунку 1.7 можна побачити середовище розробки ігрового двигуна Unreal Engine.

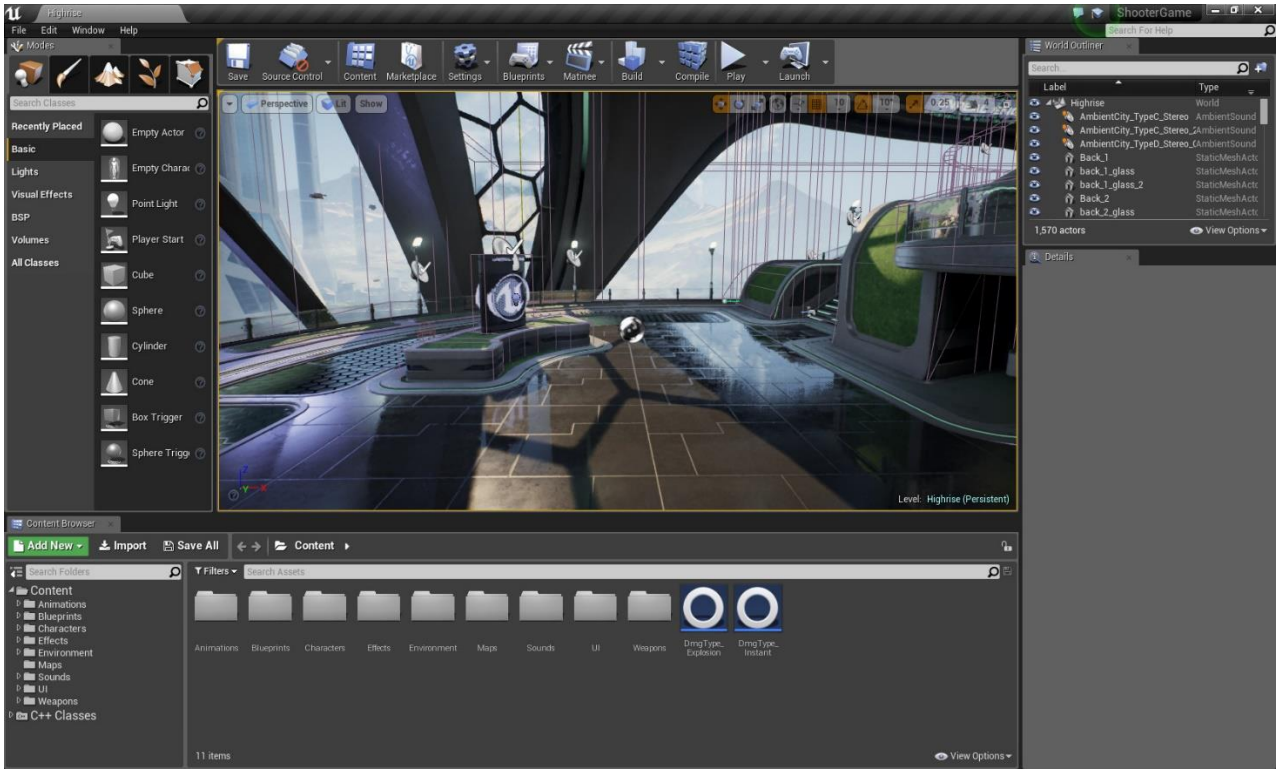


Рисунок 1.7. Середовище розробки Unreal Engine

Основні особливості Unreal Engine включають; Величезний асортимент готових компонентів, таких як моделі, текстури, анімації, ефекти світла та звуку, що допомагають створювати високоякісні графічні ефекти та реалістичні світи; Візуальний редактор Blueprints, який дозволяє створювати скрипти та логіку гри без необхідності програмування; Вбудовані інструменти для створення фізичної взаємодії об'єктів, симуляції реалістичної фізики та руху, що додають грі більшу автентичність та взаємодію з оточенням; Підтримка реалістичного освітлення та динамічних ефектів, таких як відблиски, тіні, реалістична вода та рухомі об'єкти; Можливість розгортання гри на різних платформах, включаючи Windows, macOS, Linux, Android, iOS, Xbox, PlayStation та інші.

Умови ліцензування Unreal Engine включають безкоштовну "Unreal

Engine" версію, яка дозволяє створювати та розгортати проекти безкоштовно, але зі збором роялті за дохід від продажу гри. Для платформ, таких як iOS та Android, роялті стягується, якщо дохід від гри перевищує певну межу. Крім того, Unreal Engine надає спеціальні умови ліцензування для студентів, освітніх установ та стартапів. Ось скріншот середовища розробки Unreal Engine:

Unity - це потужний ігровий двигун та середовище розробки (IDE), яке дозволяє створювати різноманітні ігрові проекти для різних платформ, таких як комп'ютери, консолі, мобільні пристрої і віртуальна реальність.

Основні особливості Unity включають: Візуальний редактор для створення і редагування сцен, об'єктів, матеріалів та анімацій; Мову програмування, такі як C# та UnityScript, для реалізації логіки гри та скриптів; Широкий спектр компонентів та систем, які допомагають в управлінні поведінкою об'єктів, фізикою, анімацією, аудіо та графікою; Підтримка інтеграції з різними зовнішніми програмами і бібліотеками; Можливість розгортання гри на різних платформах, включаючи Windows, macOS, Linux, Android, iOS, Xbox, PlayStation та інші.

Unity має кілька умов ліцензування. Існує безкоштовна версія Unity, відома як "Personal Edition", яка має обмеження на дохід розробника. Якщо розробник генерує дохід понад певну межу, він повинен перейти на платну "Plus" або "Pro" версію. Ліцензійна політика Unity також включає спеціальні умови для освітніх установ та стартапів.

Щодо фізичних можливостей Unity, двигун має вбудовану систему фізики, що дозволяє симулювати реалістичну поведінку об'єктів у грі. Unity підтримує різні види фізичних ефектів, таких як колізії, гравітація, сили, рух, розбиття об'єктів тощо. Ми можемо використовувати ці фізичні можливості для створення реалістичних ігрових сцен та взаємодії об'єктів у грі. На Рисунку 1.8 можна побачити середовище розробки ігрового двигуна Unity.

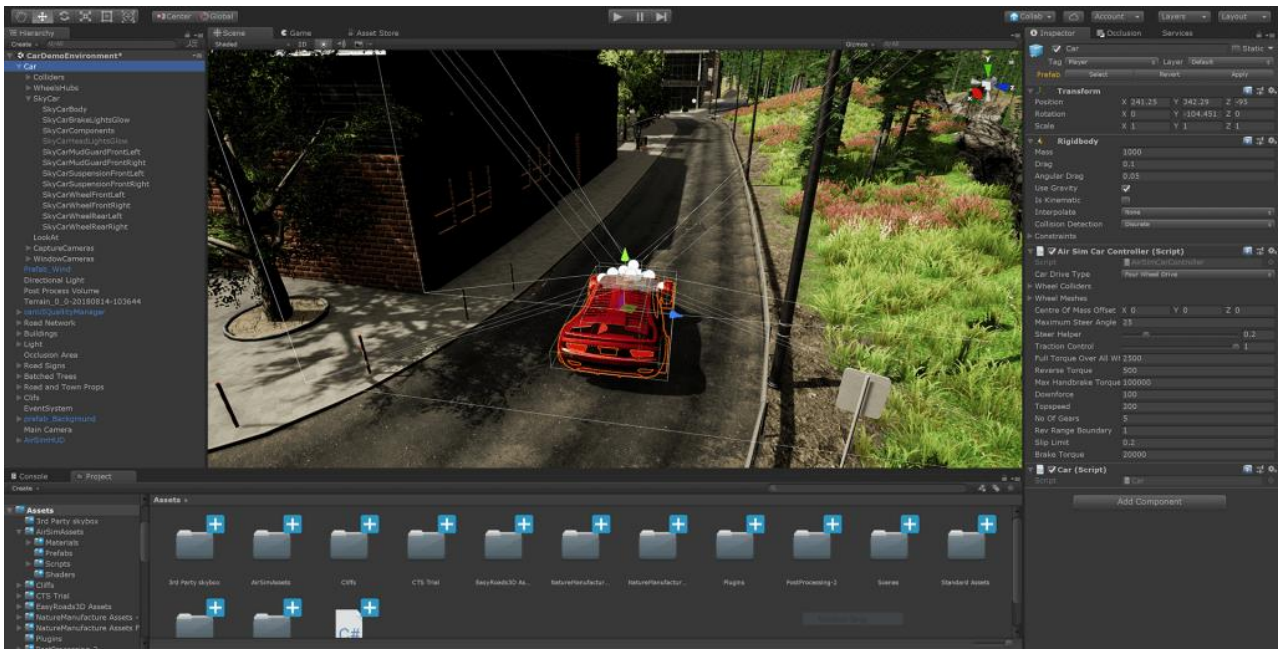


Рисунок 1.8. Середовище розробки Unity

Я обрав Unity з кількох причин. По-перше, я володію мовою програмування C#, і Unity використовує цю мову як одну з основних для розробки ігор. Це дуже зручно для мене, оскільки я розумію, як працює C# і відчуваю себе комфортно в його використанні.

По-друге, Unity має безкоштовну версію, яка дозволяє мені розробляти ігри без необхідності великих витрат на ліцензію. Це особливо важливо для мого дипломного проекту, оскільки він не комерційний за своєю природою.

Крім того, Unity має велику і активну спільноту розробників, а також велику кількість документації, онлайн-курсів та інших ресурсів, пов'язаних з розробкою ігор. Це надає мені доступ до багато інформації, яка буде корисною під час реалізації мого дипломного проекту. Unity відповідає моїм потребам і навичкам, що робить його оптимальним вибором для реалізації мого дипломного проекту.

Оскільки ми вибрали Unity для розробки нашого проекту, ми також використовуємо мову програмування C#, що означає, що наше середовище роботи над кодом - Visual Studio.

Visual Studio є інтегрованою середою розробки (IDE) від компанії Microsoft. Вона надає широкий набір інструментів та функцій, які полегшують

процес програмування, забезпечують підсвічування синтаксису, автодоповнення, налагоджувальний функціонал та багато іншого. Visual Studio має потужну систему плагінів, що дозволяє розширювати його функціональні можливості та адаптувати його під власні потреби.

Щодо ліцензування, Visual Studio пропонує кілька варіантів. Є безкоштовна версія, відома як Visual Studio Community, яка надає широкий функціонал для індивідуальних розробників та невеликих команд. Крім того, є комерційні версії, такі як Visual Studio Professional та Visual Studio Enterprise, які мають розширений функціонал та призначені для комерційного використання.

Варто зазначити, що Visual Studio Community є досить потужною та доступною платформою для розробки на C# в поєднанні з Unity. Вона дозволяє безкоштовно використовувати багато функцій, ідеально підходить для навчання та особистих проектів, включаючи наш дипломний проект.

Таким чином, використання Visual Studio разом з Unity надає нам зручне та потужне середовище для розробки, що відповідає нашим потребам і забезпечує ефективну роботу з кодом. На Рисунку 1.9 можна побачити середовище розробки Visual Studio.

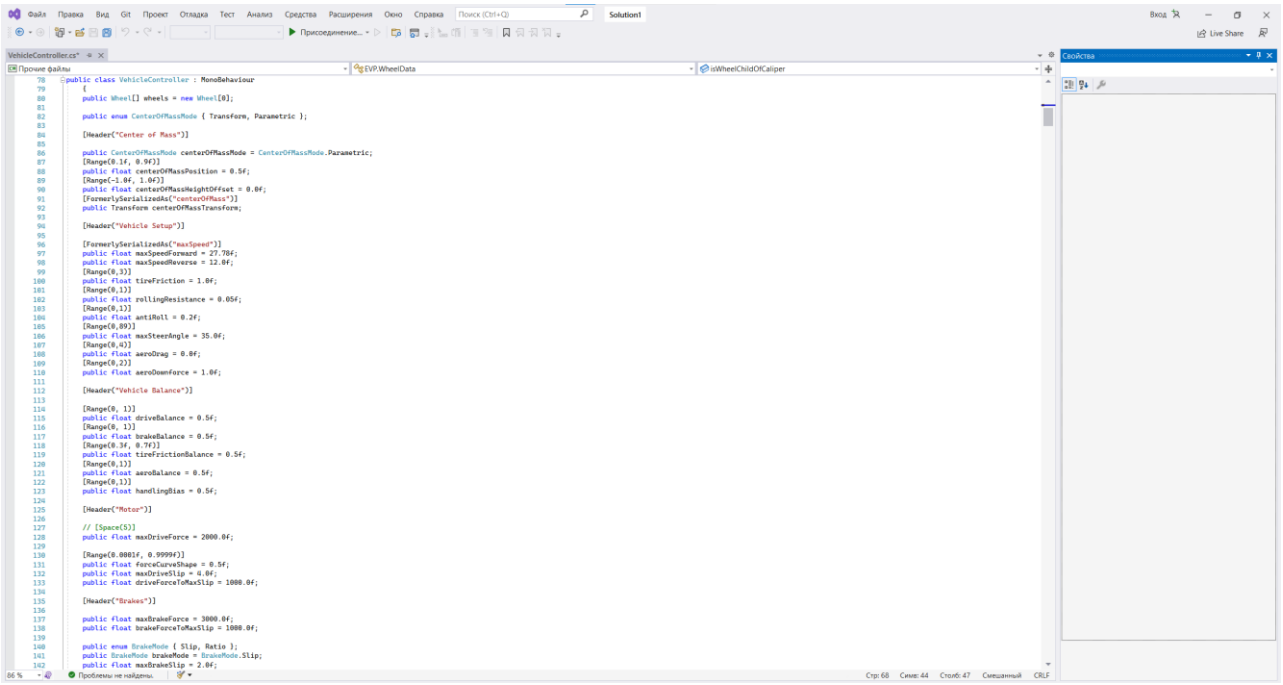


Рисунок 1.9. Середовище розробки Visual Studio

Оскільки ми вибрали Unity для розробки нашого проекту, ми маємо доступ до Unity Asset Store, який надає широкий вибір готових ресурсів, включаючи асети, скрипти, моделі, текстури, звуки та багато іншого. Цей магазин дозволяє нам швидко та зручно знайти необхідні ресурси для нашого проекту.

Unity Asset Store пропонує різні типи асетів - від безкоштовних до платних. Завдяки розширеному пошуку та фільтрам, ми маємо можливість знаходити та використовувати потрібні асети, що значно прискорює процес розробки. У контексті нашої дипломної роботи, зосередимося на виборі асетів, які стосуються графічної частини, оскільки графіка не є головною темою нашого проекту.

Після дослідження Unity Asset Store, я вже знайшов два асети, які відповідають нашим потребам у дипломній роботі. Завантаження цих асетів допоможе нам швидше розпочати реалізацію функціональності та зосередитися на основних завданнях нашого проекту. Варто підкреслити, що використання асетів з Unity Asset Store є зручним способом прискорити процес розробки та полегшити завантаження ресурсів для нашої дипломної роботи.

Перший асет "Mountain Race Tracks" є відмінним вибором для нашої гри, оскільки він надає реалістичні та деталізовані траси, на яких буде відбуватися тестування. Цей асет містить гірські ландшафти, повороти, підйоми та спуски, що додають виклику та екшену до геймплею. Ми зможемо використати ці траси як основу для нашого гоночного середовища та налаштувати їх під свої потреби. Вигляд сторінки нашого асету в Unity Asset Store на Рисунку 1.10.

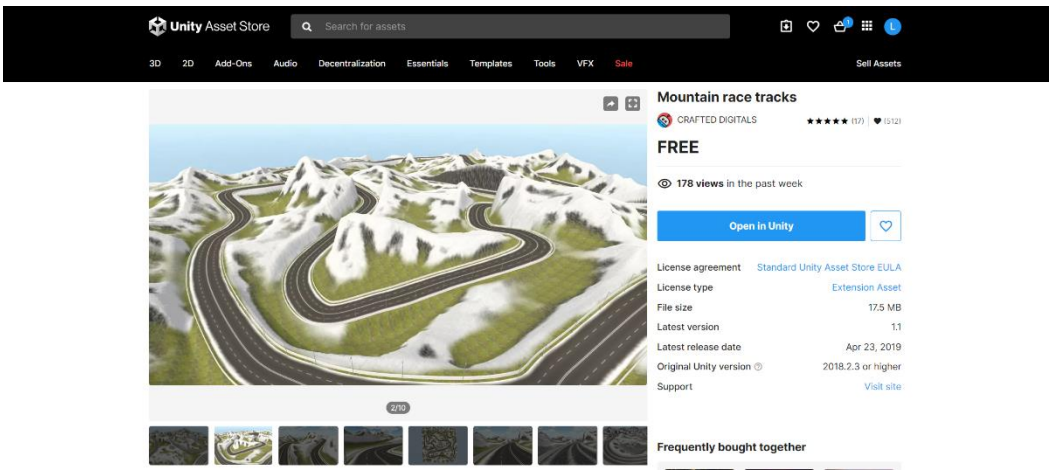


Рисунок 1.10. Сторінка асету Mountain Race Tracks

Другий асет "80's Sport Car" пропонує нам детальну модель автомобіля в стилі 80-х років. Цей асет включає в себе високоякісну 3D-модель автомобіля з усіма необхідними деталями та текстурами. Модель має реалістичну фізику руху, анімацію деталей автомобіля та можливості налаштування його характеристик. Завдяки цьому асету ми зможемо додати в нашу гру стильний та аутентичний автомобіль, який стане основою для тестування фізики. Вигляд сторінки нашого асету в Unity Asset Store на Рисунку 1.11.

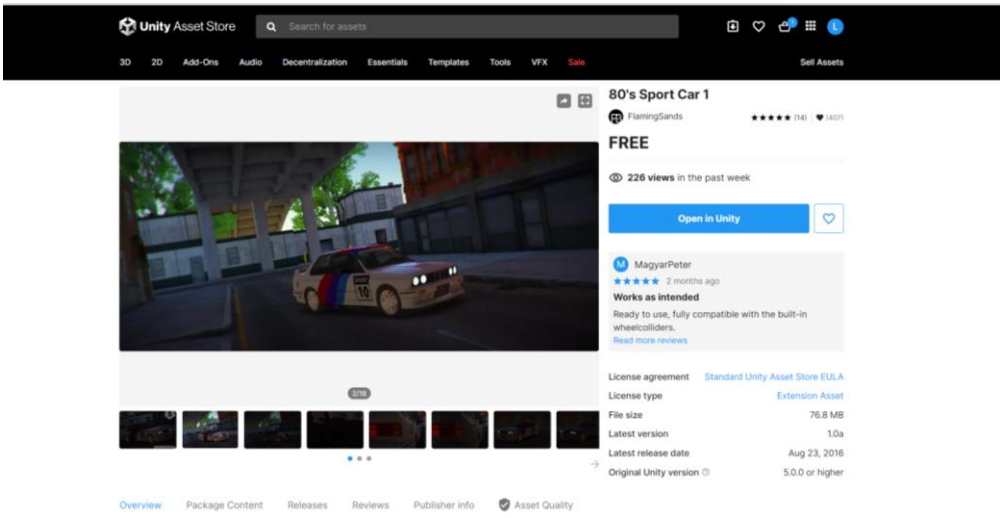


Рисунок 1.11. Сторінка асету 80's Sport Car

Використання цих асетів дозволить нам ефективно розробляти гру, використовуючи готове рішення для карти та моделі автомобіля. Ми зможемо зосередитись на реалізації гоночної механіки, управління автомобілем та інших важливих аспектів гри, що сприятиме досягненню наших цілей у дипломній роботі.

1.5 Способи реалізації фізичної моделі

У ігровій розробці існує кілька способів реалізації фізичних моделей, які дозволяють симулювати реалістичну фізику. Використання правильної фізичної моделі в іграх може покращити імерсію та відчуття взаємодії гравця з оточуючим світом. Кожен з цих підходів має свої переваги і обмеження, а їх вибір залежить від характеристик гри та бажаного рівня реалістичності.

В дискретній моделі фізики об'єкти у грі мають фіксовану позицію в просторі на кожному кроці обчислення. Це означає, що рух об'єктів моделюється шляхом зміни їхньої позиції на певну величину в певному напрямку на кожному кроці часу.

Об'єкти рухаються з фіксованою швидкістю, що означає, що кожен об'єкт має задану швидкість, яка залишається постійною протягом всього руху. Зіткнення об'єктів визначається на основі перетину їхніх геометричних форм. Наприклад, якщо два об'єкти перетинаються або перетинаються з границями один одного, відбувається зіткнення.

Після зіткнення змінюється швидкість об'єктів відповідно до заданих правил, таких як закони збереження енергії та імпульсу. Наприклад, при зіткненні двох об'єктів може відбутися відскок або зупинка одного з них в залежності від їхніх мас і швидкостей.

Цей підхід дозволяє достатньо реалістично симулювати прості фізичні ефекти, такі як рух, гравітацію, зіткнення та взаємодію об'єктів. Однак, він може бути обмеженим у моделюванні складних фізичних явищ, таких як деформація об'єктів або реалістична симуляція рідини.

Враховуйте, що це загальний опис дискретної моделі фізики, і конкретна реалізація може варіюватись залежно від ігрового двигуна або інструментів, використовуваних у розробці гри. Основна функціональна схема зображена на Рисунку 1.12:

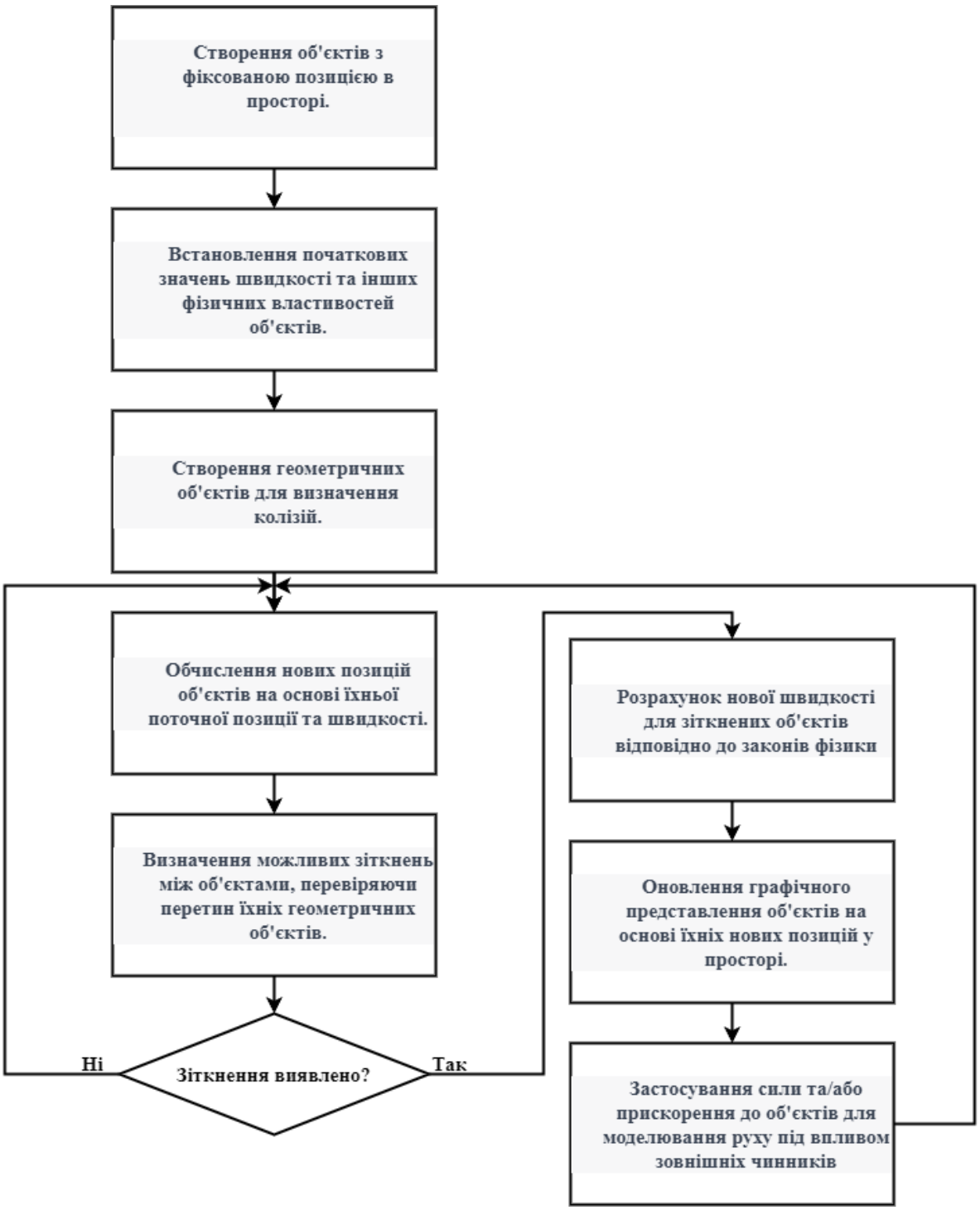


Рисунок 1.12. Функціональна схема роботи дискретної моделі фізики

М’яка тілесна модель є більш продвинутим підходом до моделювання фізики об'єктів у грі. Вона використовує фізичні принципи для моделювання об'єктів з урахуванням їхньої форми, гнучкості та матеріальних властивостей.

Цей підхід дозволяє симулювати різні типи об'єктів, які можуть бути гнучкими, деформованими або рідкими, такі як тканина, рідина або м'які тіла. Для моделювання цих об'єктів використовуються різні методи, такі як метод

скінченних елементів або метод сітки мас. Ця функціональна схема представлена у вигляді Рисунку 1.13:

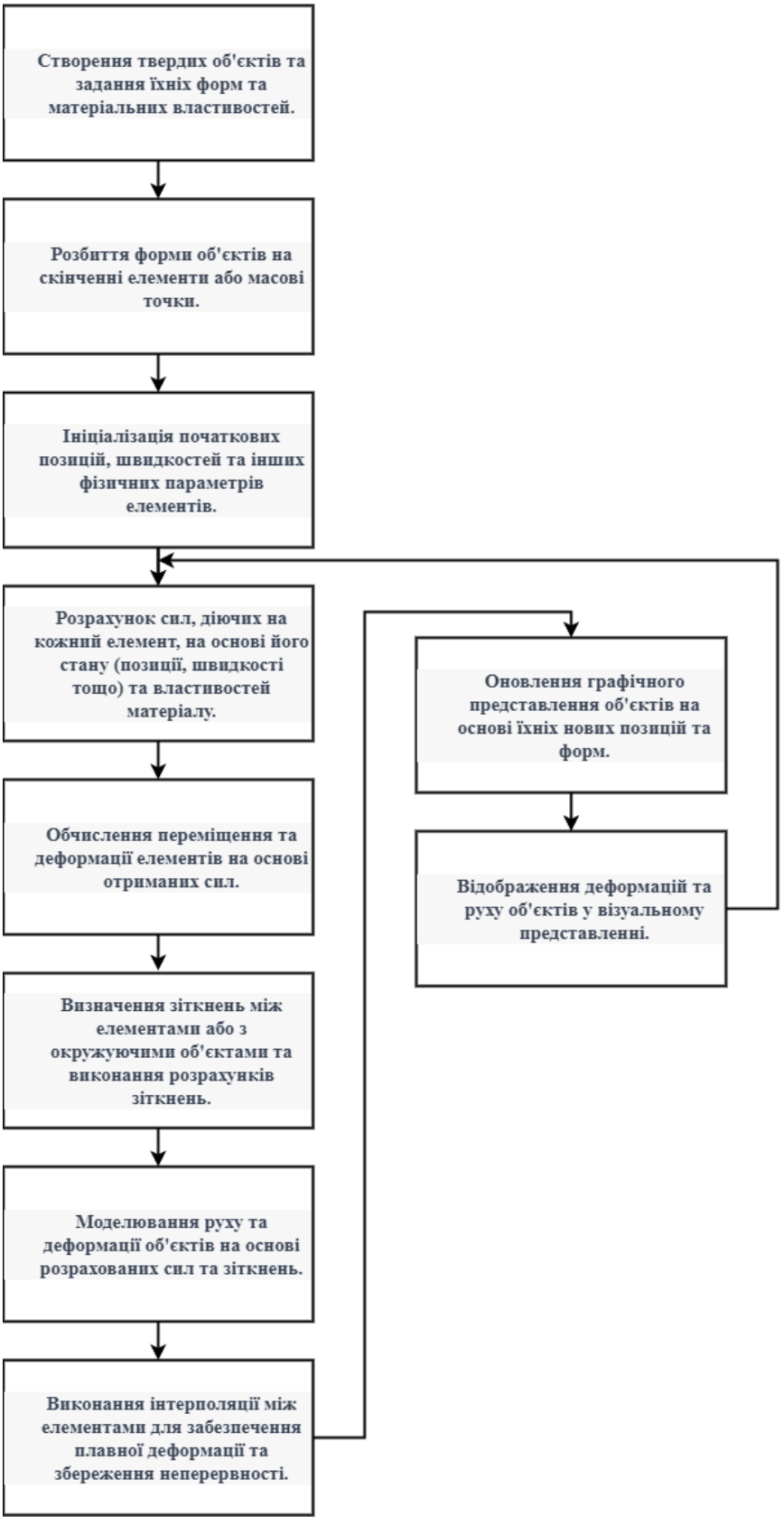


Рисунок 1.13. Функціональна схема роботи м'якої тілесної моделі фізики

М’яка тілесна модель дозволяє досягти більш реалістичних результатів у моделюванні фізики об'єктів, зокрема їхньої деформації, розтягнення та розриву. Вона знадобиться вам, якщо ваша гра вимагає докладного моделювання гнучких або деформованих об'єктів, таких як одяг персонажів, рухомих тканини, рідини або м'які тіла.

Модель частинок є ще одним підходом до симуляції фізики об'єктів у грі. Цей підхід базується на моделюванні об'єктів як системи окремих частинок, які взаємодіють між собою за допомогою фізичних законів.

Кожна частинка в цій моделі має свою масу, швидкість, положення та інші фізичні властивості. Закони фізики, зокрема закони Ньютона, застосовуються до кожної частинки для обчислення її руху та взаємодії з іншими частинками.

Симуляція моделі частинок може включати різні види взаємодій, такі як гравітація, взаємодія зовнішніх сил, зіткнення та взаємодія з поверхнею об'єктів. За допомогою цих взаємодій частинки можуть рухатися, змінювати швидкість, деформуватися та сприймати вплив оточуючого середовища.

Модель частинок зазвичай використовується для симуляції рідин, диму, волосся, вибухів, фізичних ефектів та інших подібних явищ у грі. Вона дозволяє досягти деталізованих та реалістичних ефектів, особливо у випадку комплексних та динамічних систем.

Проте, модель частинок може бути обчислювальною, оскільки вона вимагає обробки великої кількості окремих частинок та їх взаємодій. Тому для оптимізації продуктивності можуть застосовуватись різні техніки, такі як апроксимація, стиснення частинок або використання графічних прискорювачів. Функціональна схема роботи показано на Рисунку 1.14:

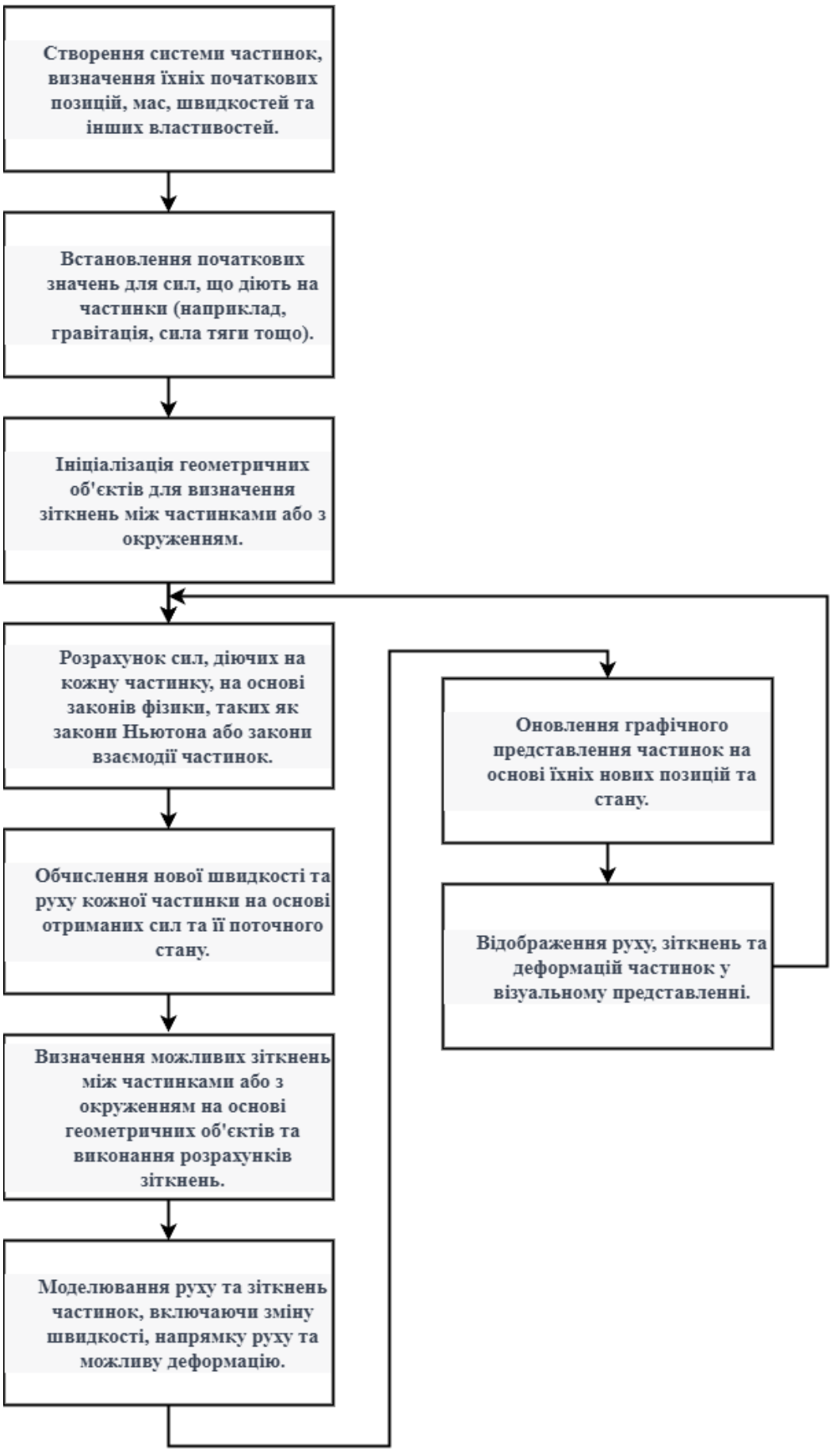


Рисунок 1.14. Функціональна схема роботи моделі частинок

Фізичні двигуни дозволяють моделювати різні аспекти фізики, такі як гравітація, зіткнення, тертя, жорсткість та деформація об'єктів. Вони враховують закони фізики та обчислюють рух об'єктів на основі цих правил.

Так, фізичні двигуни є потужним інструментом для реалізації фізики у ігрових проектах. Вони представляють собою готові бібліотеки або програмні модулі, які надають набір функцій, алгоритмів та правил для симуляції фізичних об'єктів у грі.

Використання фізичних двигунів дозволяє значно спростити процес реалізації фізики в грі. Вам не потрібно вигадувати та розробляти власні алгоритми та формули для моделювання фізичних ефектів. Замість цього ви можете скористатися готовими функціями та методами, які надаються фізичним двигуном, і просто налаштувати їх параметри для досягнення потрібних результатів.

Крім того, фізичні двигуни зазвичай оптимізовані для продуктивності та підтримують використання апаратного прискорення, такого як графічні прискорювачі, що дозволяє досягти високої швидкодії обчислень фізики у реальному часі.

Загалом, використання готових фізичних двигунів значно спрощує розробку ігрової фізики та дозволяє досягти реалістичних та деталізованих ефектів у грі.

Ця функціональна схема представлена у вигляді Рисунку 1.15:

Рисунок 1.15. Функціональна схема роботи з фізичним двигуном

1.6 Побудова алгоритму роботи скриптів фізичної моделі автомобілю

Аналіз вхідних значень керування, зокрема куту повороту, є ключовим етапом у побудові алгоритму руху автомобіля, оскільки він дозволяє враховувати фізичні закони та параметри автомобіля для досягнення бажаної поведінки автомобіля на дорозі.

Один з основних аспектів аналізу куту повороту - це його перетворення на фізичний кут повороту, що використовується в моделі автомобіля. Це можна зробити шляхом маппінгу вхідних значень куту повороту на фізичні значення, які керують рухом коліс автомобіля. Наприклад, при вхідному значенні куту повороту -1 можна встановити фізичний кут повороту, що відповідає повному лівому повороту коліс, тоді як значення 1 встановлює повний правий поворот коліс, а 0 - прямий рух.

Крім того, важливо враховувати максимальне обмеження куту повороту автомобіля. Це може бути встановлено для обмеження рівня повороту автомобіля, забезпечуючи його стабільність та безпеку. Якщо вхідне значення куту повороту перевищує максимальне обмеження, воно має бути замінене на максимально допустиме значення, щоб уникнути небажаних станів або перекручення автомобіля як показано на Рисунку 1.16.

Використання аналізованого куту повороту дозволяє обчислити інші параметри руху автомобіля. Наприклад, за допомогою куту повороту можна визначити радіус повороту автомобіля. Чим більший кут повороту, тим менший радіус повороту, що впливає на рух транспортного засобу в поворотах. Крім того, кут повороту може бути використаний для обчислення швидкості повороту автомобіля, що впливає на стійкість та реакцію автомобіля при поворотах.

Отже, аналіз вхідних значень керування, зокрема куту повороту, є важливим кроком у розробці алгоритму руху автомобіля, оскільки він дозволяє враховувати фізичні закони та параметри автомобіля для досягнення бажаної поведінки на дорозі. Цей аналіз допомагає забезпечити безпечний та

реалістичний рух автомобіля з урахуванням вхідних команд керування від водія або іншого джерела.

Рисунок 1.16. Функціональна схема роботи керування під час руху автомобіля

При аналізі прискорення в алгоритмі руху автомобіля враховуються фізичні закони та параметри автомобіля. Це дозволяє забезпечити реалістичну зміну швидкості автомобіля відповідно до вхідних значень прискорення. Алгоритм враховує обмеження щодо максимального прискорення, які впливають на здатність автомобіля до прискорення. Також враховується поточна швидкість та маса автомобіля згідно другого закону Ньютона для визначення зміни швидкості.

Аналіз прискорення уважно враховує параметри автомобіля, такі як максимальна швидкість, режим центру маси, коефіцієнти тертя та інші характеристики. Це дозволяє створити реалістичну модель руху автомобіля, де прискорення відповідає фізичним можливостям та характеристикам автомобіля.

Перетворення значення прискорення на фізичне прискорення, що використовується в моделі. Це може включати маппінг вхідних значень прискорення на фізичні значення, які керують рухом автомобіля. Наприклад, при вхідному значенні прискорення 0, фізичне прискорення може бути встановлене на нульове значення, що відповідає відсутності прискорення, а при значенні 1 - максимальному прискоренню.

За допомогою цього алгоритму, автомобіль може рухатися плавно та контрольовано згідно заданих вхідних значень прискорення, що дозволяє створити реалістичне відтворення руху автомобіля в симуляційному середовищі або в ігровому двигуні. Цю функціональну схему можна представити у вигляді Рисунку 1.17:

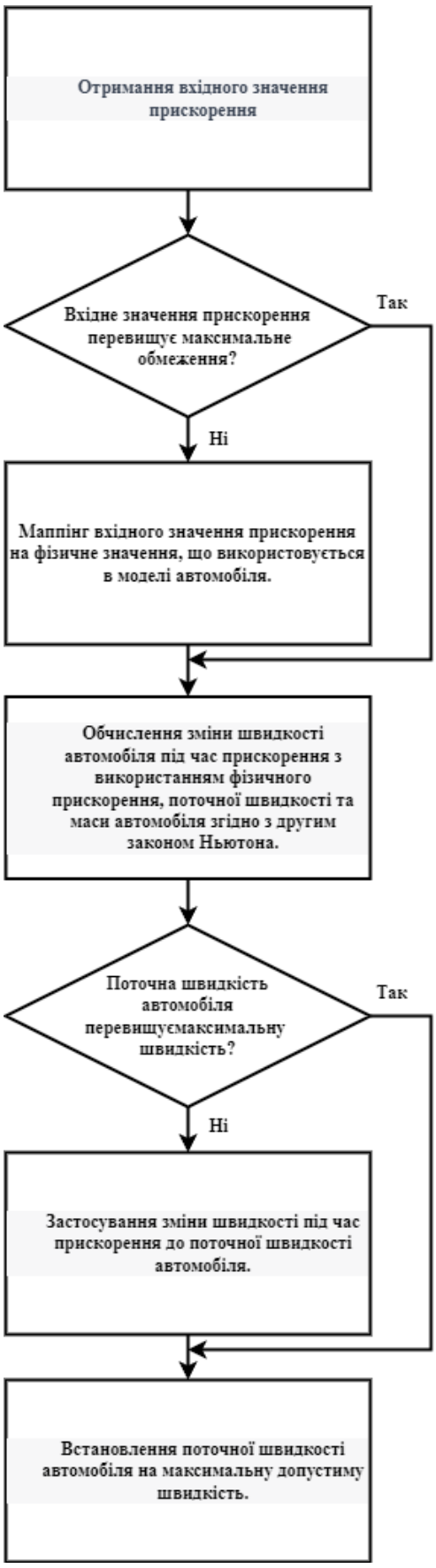


Рисунок 1.17. Функціональна схема роботи прискорення під час руху автомобіля

При аналізі гальмування в алгоритмі руху автомобіля враховуються фізичні закони та параметри автомобіля. Це дозволяє автомобілю знижувати швидкість або зупинятися відповідно до вхідних значень гальмування. Алгоритм перетворює значення гальмування на фізичне значення, яке використовується в моделі автомобіля.

Аналіз гальмування також враховує обмеження щодо максимального гальмування та мінімальної швидкості, що необхідні для забезпечення безпеки та відповідної поведінки автомобіля. Наприклад, якщо автомобіль рухається з високою швидкістю, обмеження максимального гальмування може бути застосоване для уникнення раптового зупинення та забезпечення поступового зниження швидкості.

Поточна швидкість та маса автомобіля також враховуються при аналізі гальмування. Це допомагає визначити зміну швидкості під час гальмування згідно з фізичними законами, зокрема законом збереження кінетичної енергії.

Аналіз гальмування у алгоритмі руху автомобіля дозволяє контролювати процес зниження швидкості та зупинки автомобіля відповідно до вхідних значень гальмування. Це допомагає створити реалістичну поведінку автомобіля під час гальмування та забезпечити безпеку та контрольований рух на дорозі. Основний алгоритм роботи може бути представлено на Рисунку 1.18:

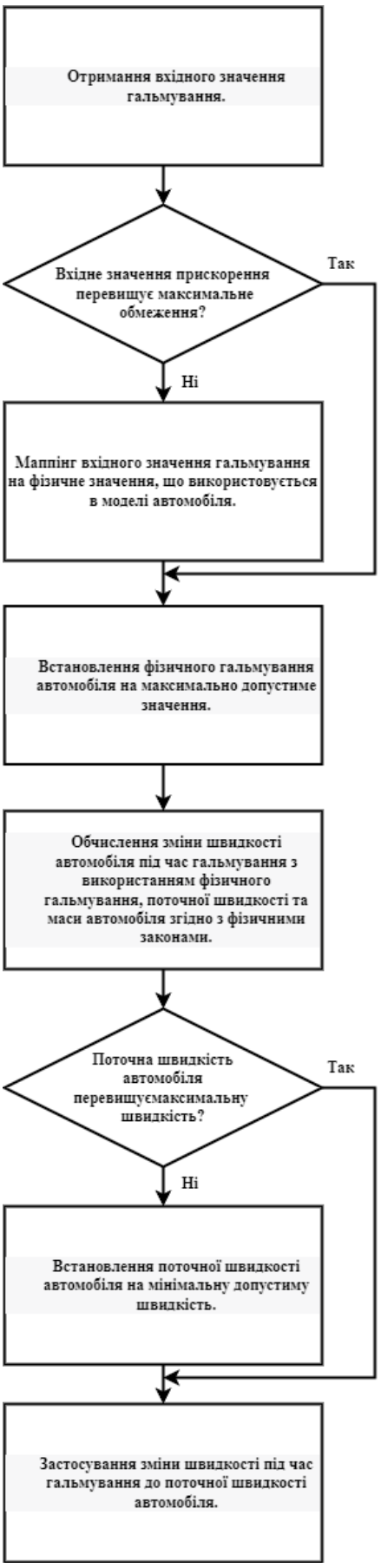


Рисунок 1.18. Функціональна схема роботи гальмування

1.7 Розробка фізичної моделі автомобіля у 3D-просторі.

Головною метою розробки є розробка логіки для ігрових об'єктів та елементів інтерфейсу гри. Цей процес розпочинається з чіткої постановки задачі і проектування роботи цих елементів, а також їх взаємодії та зв'язків між собою.

Спочатку, ми аналізуємо та визначаємо, які конкретні функції повинні виконувати ігрові об'єкти та елементи інтерфейсу. Потім ми розробляємо план роботи, визначаємо послідовність дій і необхідні кроки для досягнення мети.

На наступному етапі, ми проектуємо логіку цих елементів, визначаємо їхні можливості, поведінку та параметри. Наприклад, для ігрових об'єктів ми визначаємо, як вони рухатимуться, взаємодіятимуть з іншими об'єктами та реагуватимуть на події гри. Для елементів інтерфейсу ми проектуємо їх вигляд, функціональність та взаємодію з користувачем.

Крім того, ми враховуємо взаємозв'язки між різними елементами гри, такі як вплив одного об'єкта на інший або залежності між різними елементами інтерфейсу. Це допомагає забезпечити взаємодію та плавний хід гри для користувача.

Цей код представляє клас `VehicleController`, який відповідає за керування автомобілем в віртуальному середовищі. Він містить різноманітні функції та методи для ініціалізації автомобіля, оновлення його стану та обробки різних подій.

Основні функції та методи включають:

- `OnValidate()`: Цей метод викликається з редактора та перевіряє, що деякі параметри класу, такі як `maxDriveSlip`, `maxBrakeSlip`, `maxDriveForce`, тощо, мають невід'ємні значення.

- `OnEnable()`: Цей метод викликається при активації компонента. Він виконує різні дії, такі як пошук та збереження посилань на компоненти `Transform`, `Rigidbody` та `GroundMaterialManager`, ініціалізація даних коліс, налаштування `Rigidbody` та інше.

- `Update()`: Цей метод викликається на кожному кадрі. Він оновлює

візуальні колеса автомобіля, обчислює параметри керування, обробляє контакти зіткнення та забезпечує плавний перехід між станами тяги та гальмування.

- `FixedUpdate()`: Цей метод викликається на кожному фіксованому кроці фізичного обчислення. Він отримує вхідні значення керування, викликає методи для обчислення сил та параметрів автомобіля, прикладає сили до коліс та обробляє події зіткнення.

Цей код також містить інші допоміжні функції та методи, такі як налаштування колес, обчислення параметрів автомобіля, оновлення візуальних коліс, обробка зіткнень та багато іншого. Весь цей функціонал дозволяє створити реалістичне та гнучке керування автомобілем відповідно до заданих параметрів.

Додатково, в коді `VehicleController` можна побачити такі елементи:

- `ApplyInput()`: Цей метод приймає значення керування (керування кутом повороту, газ, гальмо, ручний гальмо) та обчислює відповідні параметри, такі як сила приводу, сила гальмування та кут повороту коліс. Він використовує ці параметри для подальшого застосування сил до коліс.

- `ApplyForcesToWheels()`: Цей метод приймає обчислені параметри (сила приводу, сила гальмування, кут повороту коліс тощо) та застосовує сили до кожного колеса автомобіля. Він враховує такі фактори, як швидкість коліс, зчеплення коліс з дорогою та розрахунок реакції на зіткнення.

- `UpdateVisualWheels()`: Цей метод оновлює позицію та обертання візуальних коліс автомобіля відповідно до позиції та обертання реальних коліс. Він забезпечує візуальну відображення руху та поворотів автомобіля.

- `UpdateDragState()`: Цей метод оновлює стан опору повітря. Він враховує різні фактори, такі як позиція та швидкість автомобіля, щоб визначити опір повітря та його вплив на рух автомобіля.

- Обробка зіткнень: В коді є можливість обробки зіткнень автомобіля. Це досягається за допомогою делегата `onImpact`, який можна налаштувати для виклику певних дій під час зіткнення автомобіля з іншим об'єктом.

Цей код має велику кількість деталей та функцій, що дозволяє точно моделювати рух та керування автомобілем. Він забезпечує реалістичну фізичну поведінку та можливості керування, що можна налаштувати для задоволення вимог конкретного додатку або гри.

`steerInput` - це вхідне значення керування кутом повороту автомобіля. Воно визначає, наскільки автомобіль повертається вліво або вправо. Значення `steerInput` зазвичай знаходиться в діапазоні від -1 до 1, де -1 відповідає максимальному повороту вліво, 0 - без повороту, а 1 - максимальний поворот вправо.

Значення `steerInput` може контролюватися за допомогою рульового колеса, джойстика або клавіші клавіатури. Використання цього значення дозволяє реалістично моделювати керування автомобілем, де повороти залежать від введення гравця.

В методі `ApplyInput()` значення `steerInput` використовується для обчислення кута повороту коліс автомобіля відповідно до введення керування. Далі цей кут повороту передається методу `ApplyForcesToWheels()`, який застосовує сили до кожного колеса для здійснення повороту автомобіля.

`throttleInput` - це вхідне значення керування прискоренням автомобіля. Воно визначає, наскільки сильно автомобіль прискорюється вперед. Значення `throttleInput` зазвичай знаходиться в діапазоні від 0 до 1, де 0 відповідає безприскоренному стану, а 1 - максимальному прискоренню.

Значення `throttleInput` може контролюватися за допомогою педалі газу, кнопки або інших контролерів. Використання цього значення дозволяє реалістично моделювати рух автомобіля, де швидкість залежить від введення гравця.

В методі `ApplyInput()` значення `throttleInput` використовується для обчислення сили приводу автомобіля. Чим більше значення `throttleInput`, тим більша сила приводу прикладається до коліс, що сприяє прискоренню автомобіля.

Далі ця сила приводу передається методу `ApplyForcesToWheels()`, який застосовує сили до коліс автомобіля, щоб забезпечити рух вперед залежно від введення прискорення.

`brakeInput` - це вхідне значення керування гальмом автомобіля. Воно визначає, наскільки сильно автомобіль гальмує або зупиняється. Значення `brakeInput` зазвичай знаходиться в діапазоні від 0 до 1, де 0 відповідає безгальмовому стану, а 1 - максимальному гальмуванню.

Значення `brakeInput` може контролюватися за допомогою педалі гальма, кнопки або інших контролерів. Використання цього значення дозволяє реалістично моделювати гальмування автомобіля, де швидкість зменшується відповідно до введення гравця.

В методі `ApplyInput()` значення `brakeInput` використовується для обчислення сили гальмування автомобіля. Чим більше значення `brakeInput`, тим більша сила гальмування прикладається до коліс, що сприяє зменшенню швидкості автомобіля.

Далі ця сила гальмування передається методу `ApplyForcesToWheels()`, який застосовує сили до коліс автомобіля, щоб забезпечити гальмування або зупинку автомобіля залежно від введення гальма.

`handbrakeInput` - це вхідне значення керування ручним гальмом автомобіля. Воно визначає, наскільки сильно ручне гальмо автомобіля застосовується. Значення `handbrakeInput` зазвичай знаходиться в діапазоні від 0 до 1, де 0 відповідає вимкненому ручному гальму, а 1 - максимальному застосуванню ручного гальма.

Ручне гальмо використовується для зупинки або фіксації автомобіля на місці. При застосуванні ручного гальма, задні колеса блокуються і утримують автомобіль у стані безрухомості або запобігають його руху назад на схилі.

У коді `handbrakeInput` використовується для визначення, чи застосовано ручне гальмо до автомобіля. Залежно від значення `handbrakeInput`, відповідні дії виконуються для ручного гальма, наприклад, блокування задніх коліс або

зменшення швидкості автомобіля.

Наступною дією буде написання коду для контролеру камери. Базовий клас `CameraMode` містить різні віртуальні методи, які можуть бути перевизначені в підкласах. Це дає можливість створювати різні камерні режими зі специфічною логікою та поведінкою.

`SetViewConfig`: Цей метод встановлює конфігурацію перегляду камери, такі як поле зору, позицію, орієнтацію тощо. Підкласи можуть використовувати цей метод для налаштування своїх внутрішніх параметрів відповідно до конкретних потреб.

`Initialize`: Цей метод викликається при ініціалізації камерного режиму. Він може використовуватись для встановлення початкових значень, налаштування параметрів та ініціалізації будь-яких необхідних компонентів.

`OnEnable`: Цей метод викликається, коли камерний режим стає активним. Він може включати в себе необхідні дії для підготовки до роботи в активному режимі, такі як підписка на події або включення необхідних компонентів.

`Update`: Цей метод викликається на кожній ітерації основного циклу оновлення програми. Він використовується для оновлення стану камери, обробки вхідних подій та виконання будь-яких інших операцій, необхідних для камерного режиму.

`OnDisable`: Цей метод викликається, коли камерний режим стає неактивним. Він може включати в себе необхідні дії для зупинки або прибирання компонентів, відписки від подій та інше.

1.8 Імплементація системи керування автомобілем

Під час реалізації проектованих блоків та коду в основному виконуються роботи з редактором ігрового двигуна Unity. Цей редактор є ключовим інструментом для створення ігрового досвіду. Він надає можливість розміщувати та налаштовувати об'єкти, встановлювати їх взаємодію та програмувати їх поведінку.

Завдяки редактору Unity, ми візуально створюємо та редагуємо сцени, де відбуватиметься гра. Вони можуть розміщувати об'єкти, такі як персонажі, предмети, терен та інші елементи гри, встановлювати їх розташування, розмір, фізичні властивості та взаємодію між ними.

Сцена в Unity - це рівень або розділ гри, де розташовуються всі ігрові об'єкти, логіка та інші компоненти. Важливо зазначити, що можна мати багато сцен і переключатись між ними. Кожна сцена може мати унікальну структуру, власні ігрові об'єкти і використовувати свої скрипти. Сцена представлена у вигляді файлу, де зберігається текстова інформація про її зміст. Коли сцена завантажується двигуном гри, він читає цю інформацію та відтворює її у своєму середовищі.

Для реалізації логіки ігрових об'єктів нам зазвичай достатньо створити прості об'єкти, які були описані раніше. Ці об'єкти можуть, наприклад, представляти засоби ремонту техніки, охолодження гармат гравця або ведення рахунку в грі. Після створення такого простого об'єкта, ми можемо розмістити його на сцені, як показано на рисунку 1.19.

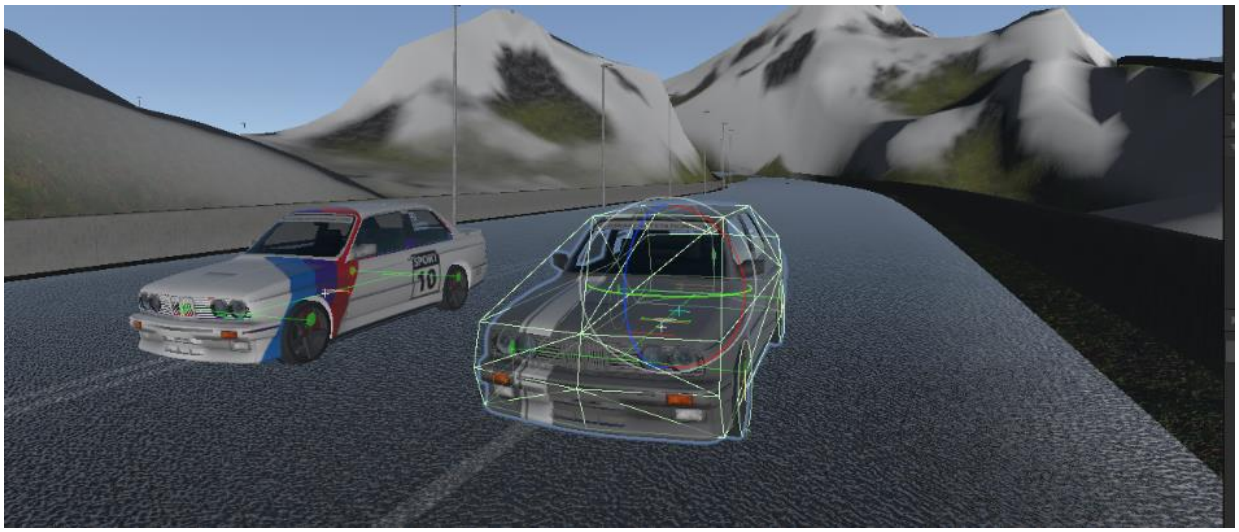


Рисунок 1.19. Об’єкти Car та DriftCar розміщені на сцені

Inspector є одним з основних розділів у двигуні Unity, оскільки саме тут ми можемо додавати нові компоненти до об'єктів. У Unity компоненти - це властивості, які присутні у ігрових об'єктів на сцені.

Деякі компоненти вже існують у об'єктах за замовчуванням, наприклад,

компонент Transform, який зберігає положення, обертання та масштаб об'єкта в просторі гри. Крім того, новостворений об'єкт має типові компоненти, які властиві кубу, наприклад, Mesh Filter, Renderer Cube, а також стандартний матеріал. Усі ці компоненти можна редагувати, видаляти або додавати нові за допомогою кнопки "Add Component". Змінювати параметри Inspector можна за допомогою вікна на рисунку 1.20:

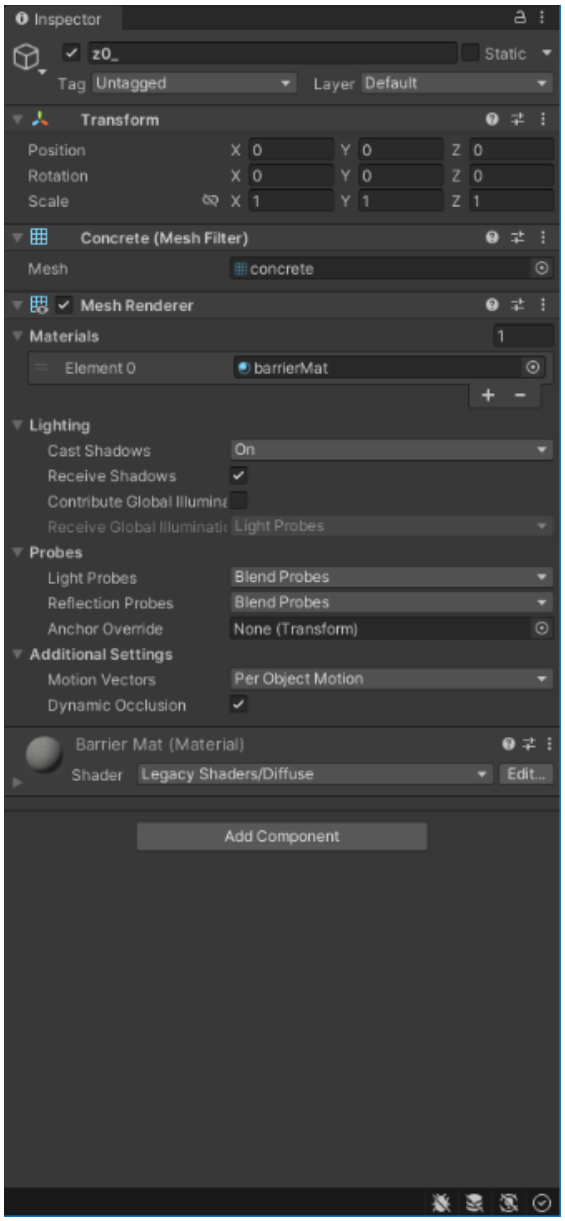


Рисунок 1.20. Параметри Inspector елемента у сцені

Після створення об'єктів, ми можемо додати до них різноманітні компоненти, які задають їхню поведінку та властивості. Наприклад, для примітивів можна додати компоненти, які визначають їхню форму, розмір і

колір. Для об'єктів освітлення можна налаштувати параметри світла, такі як його інтенсивність та кольорова температура. Додавання компонентів відбувається в розділі Inspector, де можна налаштувати їхні параметри.

Найпершим кроком є створення об'єктів, з якими ми будемо працювати. У Unity цей процес виконується шляхом створення примітивів у розділі редактора Hierarchy (див. рисунок 1.21). В цьому розділі ми можемо бачити всі ігрові об'єкти, які присутні на поточній сцені. Ці об'єкти можуть бути простими примітивами, порожніми об'єктами без вмісту всередині або спеціальними елементами, такими як освітлення, звук, відео, інтерфейс та інші.

Наприклад, ми можемо створити прямокутники, куби, кулі або інші примітиви для створення об'єктів у грі. Також ми можемо створити порожні об'єкти, які використовуються для групування інших об'єктів або для розміщення скриптів та компонентів. Крім того, Unity надає спеціальні об'єкти, які дозволяють додавати ефекти освітлення, звукові елементи, відео або елементи інтерфейсу для взаємодії з гравцем.

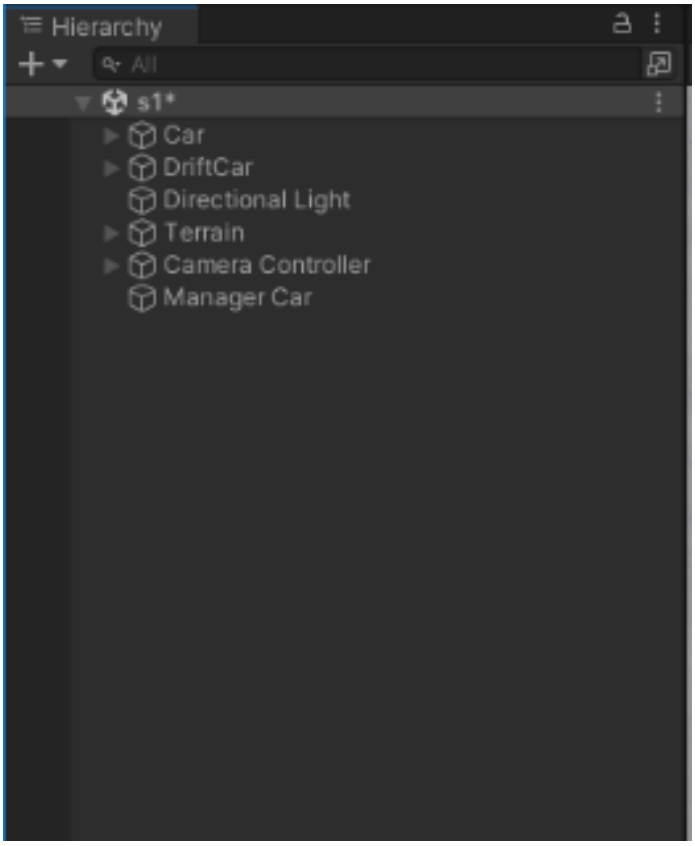


Рисунок 1.21. Вигляд розділу Hierarchy редактору ігрового двигуна

Додавання Rigidbody до автомобіля та інших об'єктів є важливим кроком в реалістичному моделюванні фізики в Unity. Rigidbody - це компонент, який надає об'єкту фізичні властивості, такі як маса, швидкість, рух і взаємодія з іншими об'єктами у сцені.

Додавання Rigidbody до автомобіля дозволяє йому реагувати на фізичні сили, такі як гравітація, та забезпечує реалістичне поведінку при зіткненнях та руху. Завдяки Rigidbody (рисунок 1.22) автомобіль може рухатися, прискорюватися, гальмувати та реагувати на удари, що додає ігровому досвіду більшу реалістичність.

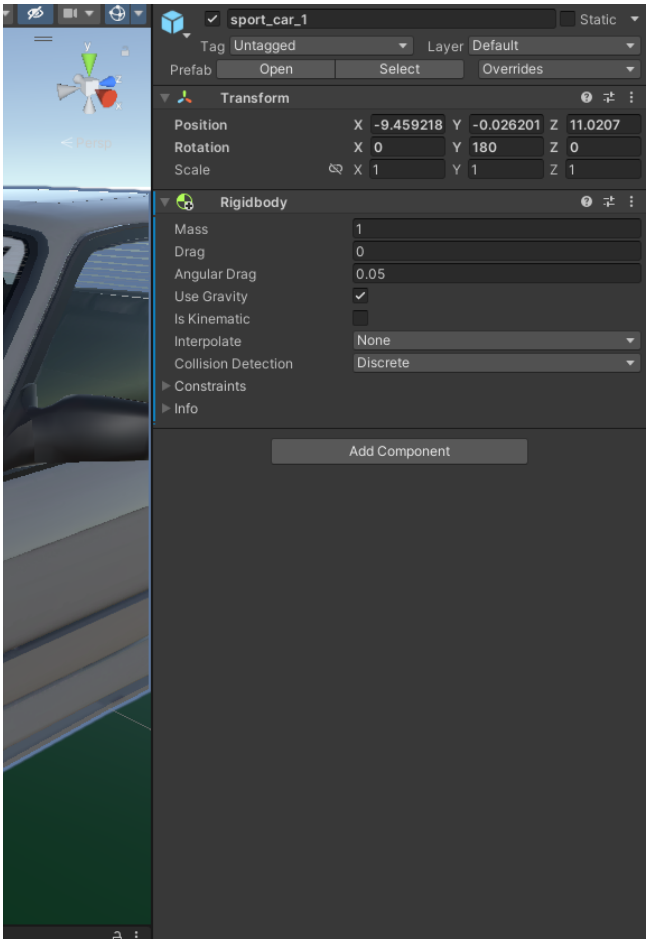


Рисунок 1.22. Додавання Rigidbody до автомобілю

Окрім того, для автомобіля та інших об'єктів можуть бути також додані інші скрипти, які контролюють їхню поведінку та взаємодію з оточуючим середовищем. Наприклад, скрипт керування автомобілем може реагувати на вхідні дані від користувача, керувати швидкістю та напрямом руху. Скрипти для зіткнень можуть визначати наслідки зіткнення об'єктів, наприклад, змінювати

їхню швидкість, спричиняти пошкодження або активувати анімацію.

Додавання скрипту VehicleController до автомобіля (рисунок 1.23) та інших об'єктів є важливим кроком в реалістичному моделюванні фізики в Unity. Скрипт VehicleController відповідає за керування поведінкою автомобіля та взаємодію з оточуючим середовищем.

У скрипті VehicleController можна використати компонент Rigidbody для надання автомобілю фізичних властивостей. Це дозволяє автомобілю реагувати на фізичні сили, такі як гравітація, та забезпечує реалістичне поведінку при зіткненнях та руху. Завдяки Rigidbody скрипт VehicleController може керувати швидкістю, прискоренням, гальмуванням та іншими аспектами руху автомобіля.

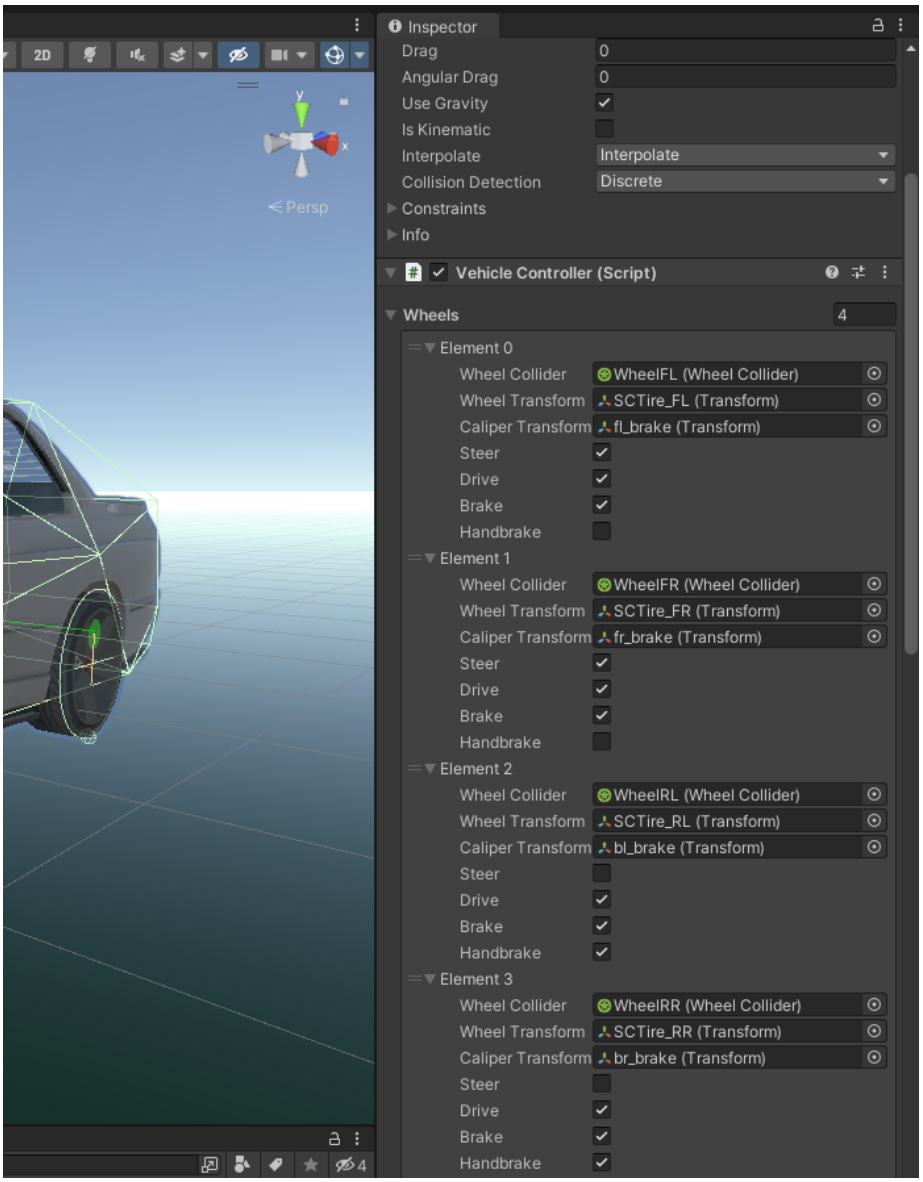


Рисунок 1.23. Додавання скрипту VehicleController до автомобілю

Крім компонента Rigidbody, в скрипт VehicleController можуть бути включені інші логіки та функціональність для керування автомобілем. Наприклад, можна включити логіку для обробки вхідних даних від користувача, щоб керувати рухом автомобіля за допомогою клавіатури або контролера. Також можуть бути включені скрипти для обробки зіткнень та взаємодії з іншими об'єктами в сцені, що дозволяють визначати наслідки зіткнень та взаємодіючих сил.

Додавання скрипту VehicleStandardInput (рисунок 1.24) до автомобіля та інших об'єктів є важливим кроком у реалістичному моделюванні фізики в Unity. Скрипт VehicleStandardInput відповідає за обробку вводу користувача та передачу вхідних значень до інших скриптів або компонентів, що керують фізичною моделлю об'єкта.

Додавання скрипту VehicleStandardInput до автомобіля дозволяє йому реагувати на вхідні дані користувача та керувати фізичним моделюванням автомобіля. Наприклад, шляхом отримання значень осей керування (таких як "steerAxis", "throttleAndBrakeAxis", тощо) скрипт VehicleStandardInput визначає керування кермом, прискоренням, гальмуванням та ручним гальмом автомобіля. Ці значення потім передаються до інших компонентів або скриптів, таких як Rigidbody або інші скрипти, які відповідають за фізичне моделювання автомобіля.

Наприклад, значення керування передаються до компонента Rigidbody автомобіля, який відповідає за фізичну поведінку об'єкта. Завдяки Rigidbody, автомобіль отримує фізичні властивості, такі як маса, швидкість, рух та взаємодію з іншими об'єктами у сцені. Це дозволяє автомобілю реагувати на сили, такі як гравітація, зіткнення з іншими об'єктами, а також рухатися, прискорюватися та гальмувати згідно з отриманими значеннями від скрипту VehicleStandardInput.

Таким чином, скрипт VehicleStandardInput співпрацює з компонентами Rigidbody та VehicleController для створення реалістичного моделювання фізики

та поведінки автомобіля у віртуальному середовищі Unity.

Рисунок 1.24. Додавання скрипту VehicleStandardInput до автомобілю

Включення компонента RigidbodyPause (рисунок 1.25) до автомобілів та інших об'єктів є складовою реалістичного моделювання фізики в Unity. RigidbodyPause є спеціальним скриптом, який додає фізичні властивості до об'єктів, таких як маса, швидкість, рух та взаємодія з іншими об'єктами у сцені.

Цей компонент дозволяє об'єктам реагувати на фізичні сили, такі як гравітація, і відтворювати реалістичне поведінку під час зіткнень та руху. За допомогою RigidbodyPause автомобіль може вставати на місце та ніякі фізичні дії не будуть впливати на автомобіль до тих пір коли ми не зупинимо роботу цього скрипта.

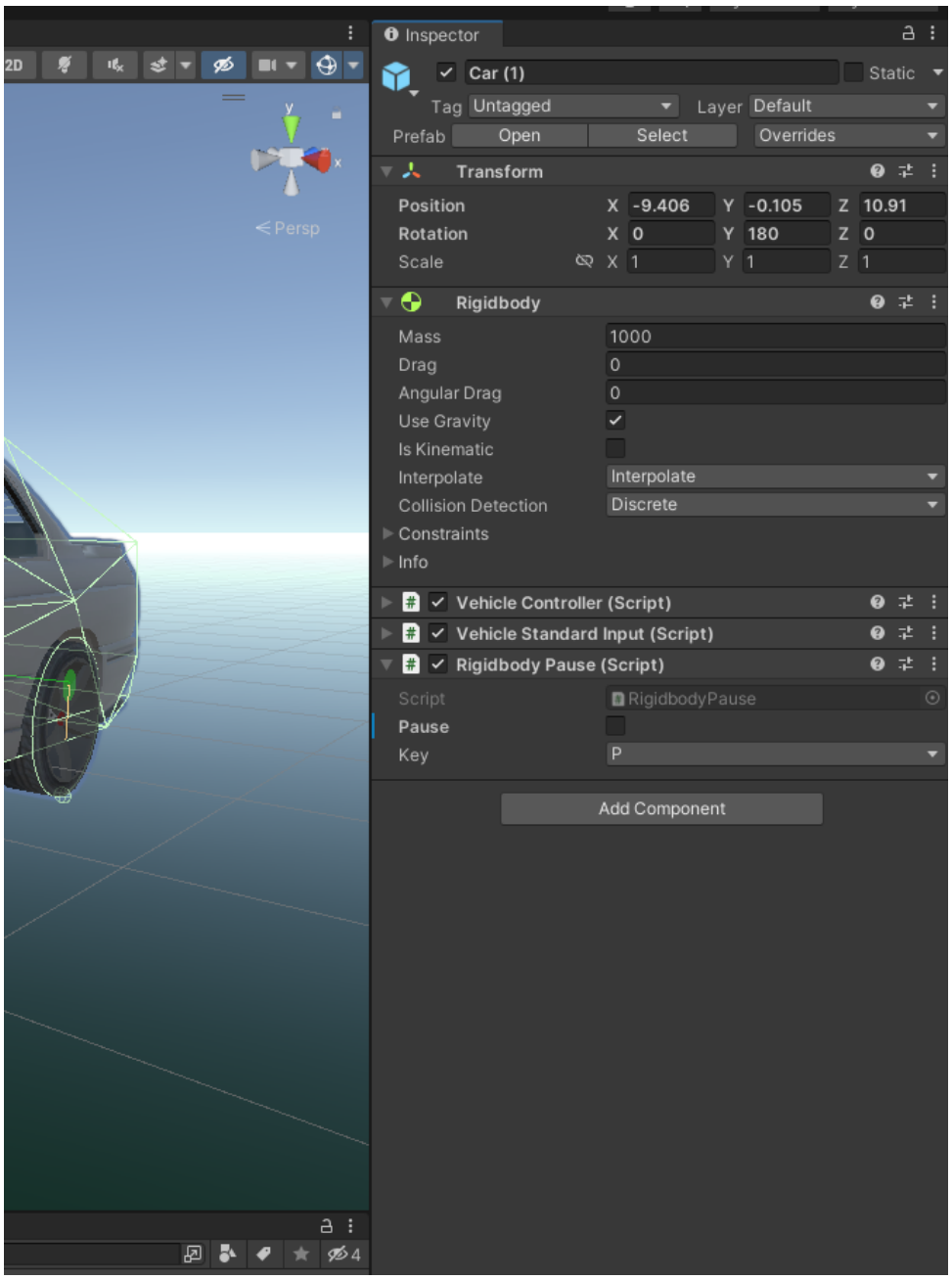


Рисунок 1.25. Додавання скрипту RigidbodyPause до автомобілю

За допомогою коду VehicleController в Unity треба додати колеса до автомобіля та керувати їх рухом(рисунок 1.26). Для початку, потрібно мати готовий модель автомобіля з встановленими колесами.

Далі потрібно налаштувати параметри кожного колеса, такі як положення, орієнтацію та з'єднання з моделлю автомобіля. Для цього треба використовувати методи і властивості класу WheelCollider.

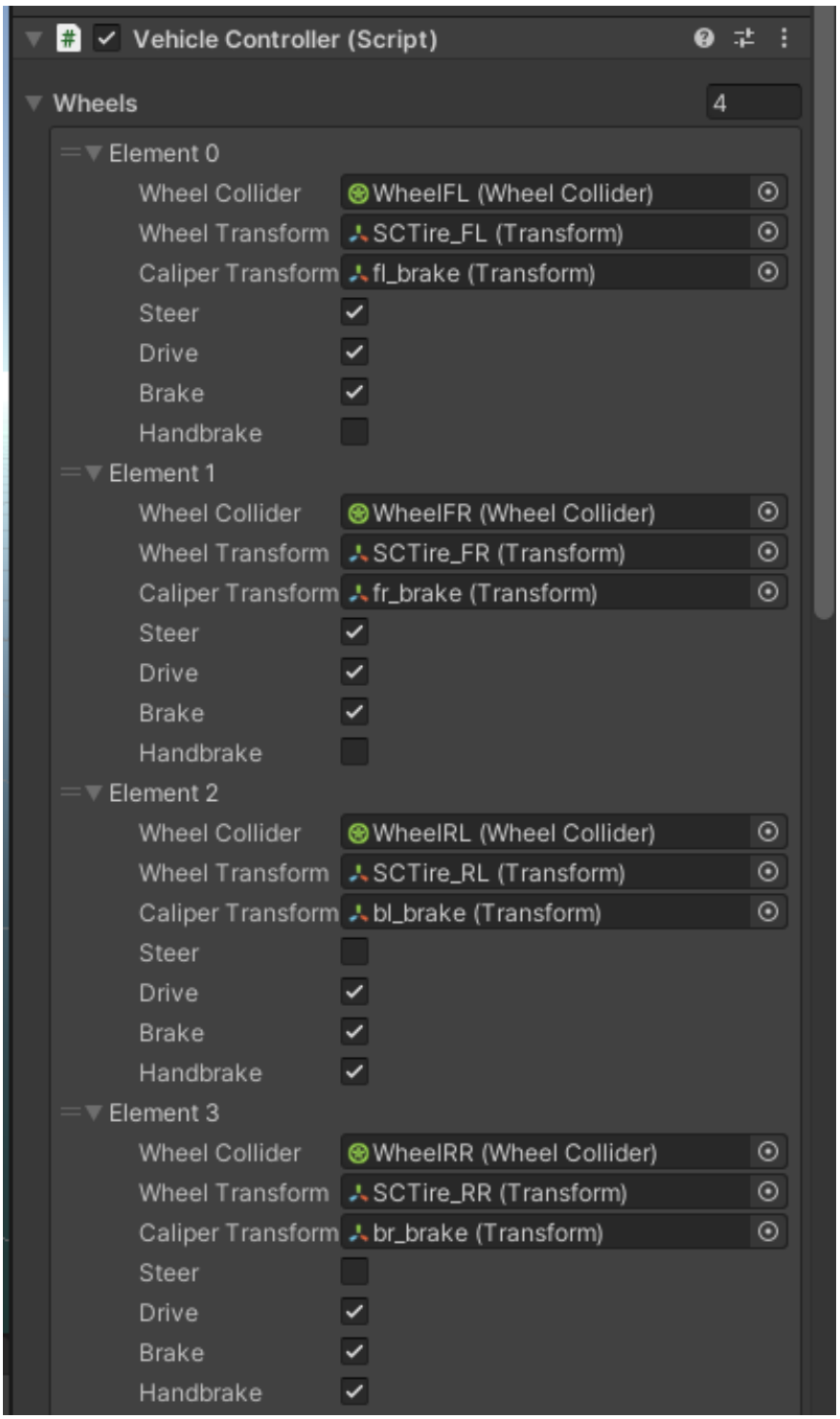


Рисунок 1.26. Налаштування колес у Автомобілі в скрипті VehicleController

В скрипті VehicleController в Unity ми можемо налаштувати різноманітні функції автомобіля, щоб забезпечити його реалістичне поведінку та керованість(рисунок 1.27). Основні функції, які можна настроїти, включають управління рухом, гальмування, обертання, підвіску та інші.

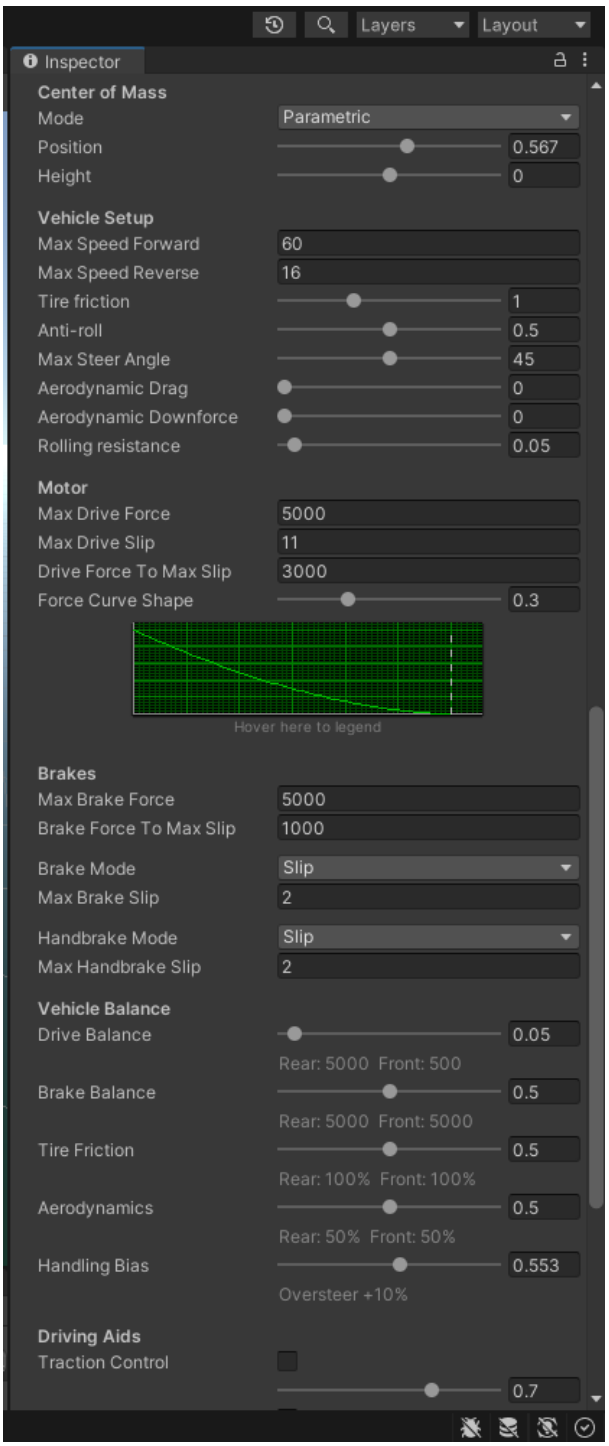


Рисунок 1.27. Налаштування функцій пересування автомобіля

В результаті, завдяки скрипту VehicleController та використанню компонента Rigidbody, автомобіль та інші об'єкти можуть мати реалістичну фізичну поведінку в середовищі Unity. Це додає більшу автентичність та імерсію до ігрового досвіду, дозволяючи автомобілю рухатися, реагувати на сили та взаємодіяти з оточуючим середовищем, що створює більш реалістичну симуляцію фізики руху.

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

В даному дипломному проекті розроблено «Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку».

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних.

Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Таблиця 2.1Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизованих розрахунків	680 – 7000
3. ПП введення інформації	1300 – 8600
4. ПП оптимізації розрахунків	7800 – 8800

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2 Трудомісткість

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): T^a = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (таблиця 2.3);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (таблиця 2.4);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (таблиця 2.4).

Таблиця 2.3 Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4 Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Таблиця 2.5 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$Т_{тз}=Т^а *L_1*К_н= 183,2 *0,12*0,7=$ 15,38 (люд/годин)

Трудомісткість розробки технічного проекту

$Т_{тп}=Т^а *L_2*К_н = 183,2$ *0,11*0,7 = 14,11(люд/годин)

Трудомісткість розробки робочого проекту

$Т_{рп}=Т^а *L_3*К_н*К_т= 183,2 *0,61*0,7*0,8= 62,58$ (люд/годин)

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}=$ 3 (стр), розробка ТП $N_{тп}=19$(стр), розробка робочого проекту $N_{рп}=14$(стр), пояснювальна записка відповідно $N_{пз}=44$ (стр).

Таблиця 2.6 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1.Розробка ТЗ	$T_{Pтз}$= 15,38	$T_{кк}=0,7*N_{тз}$ = 2,1	$T_{нк}=0,15*N_{тз}$= 0,45
2.Розробка ТП	$T_{Pтп}$= 14,11	$T_{кк}=0,7*N_{тп}$= 13,3	$T_{нк}=0,15*N_{тп}$= 2,85
3.Розробка РП	$T_{Pрп}$= 62,58	$T_{кк}=0,7*N_{рп}$= 9,8	$T_{нк}=0,15*N_{рп}$= 2,1
4.Розробка ПЗ	$T_{пз}=1,5*$ $*N_{пз}$= 54	$T_{кк}=0,7*N_{пз}$= 25,2	$T_{нк}=0,15*N_{пз}$= 5,4
Усього, в т.ч.:	ΣT=207,6		
- на розробку	$\Sigma T_р$=146,1		
- контроль керівника		$\Sigma T_{кк}$=50,7	
- нормоконтроль			$\Sigma T_{нк}$=10,8

2.3 Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.7 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	$\Sigma T_р$=146,1	40,46	591,21
2.Контроль керівника	$\Sigma T_{кк}$=50,7	80.00	4096
3.Нормоконтроль	$\Sigma T_{нк}$=10,8	80.00	756
Усього	-	-	ΣЗо = 10723,21

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПП

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	76	3	210
	Лист А1	-	-	$В_{м\,і} = 210$
Транспортні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1 = 21$
Усього				$В_м = В_{мі} + В_{тр\_з} = 231$

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	231	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10723,21	ΣЗо(див. табл. 2.7.)
3.Додаткова заробітна плата	1072,32	$Зд = 0{,}15 \times Зо = 4\,534{,}68$
4.Відрахування до єдиного фонду соціального внеску	2595	$Вє.с.в. = 0{,}22 \times (Зо + Зд) = 0{,}22*$ (10723,21+1072,32)
5. Накладні витрати	4289,28	$Внак. = 0{,}3 \times Зо = 0{,}3 * 35231{,}20 =$ 10 569,36
6. Повна собівартість	18910,81	$С_{пов} = В_м + З_о + З_д + Вє.с.в. + Внак. =$ 231,00+10723,21+1072,32+2595,00 + 4289,28

Розмір прибутку, що включен в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100 =(18910{,}81*10)/100=1891{,}08$грн

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П = 18910{,}81+1891{,}08 = 20801{,}89$ грн;

Ціна реалізації розробленого програмного продукту, становитиме:

$Ц_р=Ц_о+ПДВ = 20801{,}89 +20801{,}89 *0.2 = =24962,27$ грн;

3 ОХОРОНА ПРАЦІ

У даному розділі ДП вирішується питання охорони праці програміста на стадії реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці даного програмного комплексу

Робота з ПК супроводжується підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. На робочому місці користувача ПК виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини, підвищений рівень електромагнітних випромінювань радіочастот, висока напруга електричної мережі, статична електрика та інші.

3.2 Гігієнічні вимоги до виробничого середовища

Вивчення умов праці на робочому місці користувача ПК є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів.

3.2.1 Вимоги до приміщення

Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м^2, а об'єм не менший за 20м^3 , Також приміщення повинно мати як штучне, так і природне освітлення. Воно має бути забезпечено системою опалення, кондиціювання повітря або припливно-витяжною вентиляцією.

Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями.

3.2.2 Освітлення

Освітлення приміщення, у якому працює користувач персонального комп’ютера (програміст), використовується змішане освітлення, тобто сполучен-

ня природного й штучного освітлення. Природне освітлення - здійснюється через вікна в зовнішніх стінах будинку. Штучне освітлення - використовується при недостатньому природному освітленні й здійснюється за допомогою двох систем: загального й місцевого освітлення. Для загального освітлення приміщення, де перебуває робоче місце програміста, використовуються газорозрядні лампи типу ЛД. Нормами для даних робіт установлена необхідна освітленість робочого місця E_H=300 лк (для робіт високої точності, коли найменший розмір об'єкта розрізнення дорівнює 0,3 – 0,5 мм).

3.2.3 Шум

У робочих приміщеннях основними джерелами акустичних шумів є шуми ПЕОМ. ЕОМ є також джерелами шумів електромагнітного походження (коливання елементів електромеханічних пристроїв під впливом змінних магнітних полів). Допустимий еквівалентний рівень шуму для робочого місця оператора складає 65 дБА. Під час виконання робіт з ПК у виробничих приміщеннях значення характеристик вібрації на робочих місцях мають не перевищувати допустимі відповідно до ДСанПіН 3.3.2.007-98, ДСН 3.3.6-039-99

3.2.4 Вимоги до організації робочого місця працівника

Створення зручного робочого місця повинно бути одним з головних пріоритетів роботодавця. Створення зручного робочого місця повинно бути одним з головних пріоритетів роботодавця.

Для людина, що працює, потрібно створити санітарні умови, які б дали змогу їй плідно працювати, не перевтомлюючись та зберігати своє здоров'я. Для цього треба, щоб енергетичні витрати при праці компенсувалися відпочинком та умовами оточуючого середовища. Ці умови створюються забезпеченням для працюючого:

1) зручного робочого місця;
2) чистого повітря;
3) нормованої освітленості;
4) захисту від шуму та вібрацій;

5) захисту від дії шкідливих речовин та випромінювань;

6) робочим одягом та різними засобами індивідуального захисту;

7) побутовими приміщеннями та спеціальними службами, що призначені

8) створювати безпечні та нормальні умови праці.

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ДСаПіН 3.3.2.-007-08). Конструкція робочого місця й взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно росту працюючих для підтримки зручної пози.

Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в у припустимі рамки від 60 до 90 см

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехтливість екрана від вікон і приладів штучного освітлення, якими є лампи накалювання. Частота мерехтіння екрана $f_{мер}$=100 Гц, що відповідає умові $f_{мер}$>70 Гц.

3.2.5 Мікроклімат

Найбільш значним фактором продуктивності й безпеки праці є виробничий мікроклімат, що характеризується температурою й вологістю повітря, швидкістю його руху, а також інтенсивністю радіації, і повинен відповідати ГОСТ 12.1.005-88 і СНиП 2.04.05-86, тому значення параметру мікроклімату повинно становити: температуру повітря від 18-22 градусів Цельсію, вологість повітря від 40%-60%, та швидкість повітря від 0,1-0,2 м/с.

Для підтримки в приміщеннях нормального, що відповідає гігієнічним вимогам складу повітря, видалення з нього шкідливих газів, пару і пилу використають вентиляцію

Механічна вентиляція (кондиціонери вентилятори і т.ін.) залежно від напрямку руху повітряних потоків, може бути витяжною, нагнітаючої і нагнітаючо-витяжною. При природній вентиляції (за допомогою вікон) повітря надходить у приміщення й віддаляється з нього внаслідок різниці температур, а також під дією вітру.

3.2.6 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення відповідно до ОНТП24-86 і ПУЕ-87 відноситься до класу приміщень без підвищеної небезпеки поразки персоналу електричним струмом, оскільки відносна вологість повітря не перевищує 75%, температура не більш 35ºС, відсутні хімічно агресивні середовища. Живлення електроприладів усередині приміщення здійснюється від двухфазної мережі з заземленою нейтралю напругою 220 В і частотою 50 Гц із використанням автоматів токового захисту. У приміщенні повинна бути застосована схема заземлення.

Заземлення повинно бути зроблено за допомогою гнучкого сплетеного мідного проводу діаметром порядку 1,5 мм2. Для зменшення значень напруг дотику і відповідних їм величин струмів, при нормальному й аварійному режимах роботи устаткування необхідно виконати повторне захисне заземлення нульового проводу. Відповідно до ГОСТ-12.2.007.0-75 все устаткування (крім ЕОМ - II клас) відноситься до I класу, воно має робочу ізоляцію відповідно до вимог ГОСТ 12.1.009-76. Підключення устаткування виконане відповідно до вимог ПБЕ та ПУЕ. Додаткових заходів по електробезпечності не потрібно.

3.3 Пожежна безпека

Коли від пожежі захищаються приміщення з персональними комп’ютерами, то слід ураховувати специфіку вогнегасних речовин у вогнегасниках, які призводять під час гасіння до псування обладнання. Ці приміщення реко-

мендується оснащувати вуглекислотними вогнегасниками з урахуванням граничнодопустимої концентрації вогнегасної речовини.

Будинки, споруди, приміщення, технологічні установки повинні бути забезпечені первинними засобами пожежогасіння: вогнегасниками, ящиками з піском, покривалами з негорючого теплоізоляційного полотна, грубововняної тканини чи повсті, іншим пожежним інструментом, які використовуються для локалізації і ліквідації пожеж у початковій стадії їхнього розвитку.

Для зазначення місцезнаходження первинних засобів пожежогасіння слід установлювати відповідні знаки згідно з чинними державними стандартами. Знаки слід розміщувати на видних місцях на висоті 2-2,5 м від рівня підлоги як у середині, так і поза приміщеннями (у разі потреби).

Переносні вогнегасники повинні розміщуватися шляхом:

1) навішування на вертикальні конструкції на висоті не більше 1,5 м від рівня підлоги до нижнього торця вогнегасника і на відстані від дверей, достатній для її повного відчинення;

2) установлення в пожежні шафи пожежних кранів, або у спеціальні тумби; 3) навішування вогнегасників на кронштейни, розміщення їх у тумбах або пожежних шафах повинне забезпечувати можливість прочитання маркувальних написів на корпусі.

ВИСНОВКИ

У межах виконання моєї дипломної роботи я розробив фізику автомобіля у 3D просторі гри. Для цього були проаналізовані різні існуючі ігрові застосунки, з яких були взяті основна ідея та механіка для мого додатка.

Окремий розділ роботи присвячений фізики програмного забезпечення, що є одним з найважливіших аспектів розробки. Він дозволяє реалізувати та показати, як наш програмний продукт буде працювати після завершення розробки. Наступний розділ присвячений імплементації фізики до гри, включаючи опис основних елементів, таких як автомобіль, скрипти, префаби, які були реалізовані в грі. Також були описані основні можливості гравця.

Окремий розділ був приділений проектуванню ігрового застосунку, в результаті чого було обрано середовище розробки. З-поміж двох претендентів був обраний ігровий двигун Unity 3D. Цей двигун є легким, але має велику кількість функціоналу. Робота складалась із розробки макету проекту, а також створення відповідних скриптів фізики машини. Для реалістичної симуляції виконувались різні підходи у розробці цих фізичних систем.

Також була протестована основна ігрова механіка гри на вибраній аудиторії і отримані відгуки, що дозволило сформулювати пропозиції щодо поліпшення гри.

Під час виконання дипломного проектування я отримав значні навички в роботі з Unity 3D. Всі поставлені задачі були успішно виконані, і мета мого дипломного проектування була досягнута.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hocking J. Unity in Action: Multiplatform Game Development in C# with Unity 5. Shelter Island, New York: Manning Publications. 352 p.
2. Unity User Manual: [Website]. URL: https://docs.unity3d.com/Manual/index.html (viewed on: 21.05.2023).
3. Unity: [Website]. URL: https://unity.com/ru (viewed on: 21.05.2023).
4. C# docs: [Website]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ .
5. Unity: Mountain race tracks [Prefabs]. URL: https://assetstore.unity.com/packages/3d/environments/landscapes/mountain-race-tracks-110408
6. Unity: 80's Sport Car 1 [Prefabs]. URL: https://assetstore.unity.com/packages/3d/vehicles/land/80-s-sport-car-1-69324
7. Unity User Manual, Create a vehicle with Wheel Colliders: [Website]. URL: https://docs.unity3d.com/Manual/WheelColliderTutorial.html

ДОДАТОК А ЛІСТИНГ ОСНОВНИХ МОДУЛІВ ГРИ

Скрипт VehicleStandardInput.cs

```
void FixedUpdate (){
if (target == null) return;
float steerInput = Mathf.Clamp(Input.GetAxis(steerAxis), -1.0f, 1.0f);
float handbrakeInput = Mathf.Clamp01(Input.GetAxis(handbrakeAxis));

float forwardInput = 0.0f;
float reverseInput = 0.0f;

if (throttleAndBrakeInput == ThrottleAndBrakeInput.SeparateAxes){
forwardInput = Mathf.Clamp01(Input.GetAxis(throttleAxis));
reverseInput = Mathf.Clamp01(Input.GetAxis(brakeAxis));}
else{

forwardInput=Mathf.Clamp01(Input.GetAxis(throttleAndBrakeAxis));
reverseInput=Mathf.Clamp01(-Input.GetAxis(throttleAndBrakeAxis));}
float throttleInput = 0.0f;
float brakeInput = 0.0f;

if (continuousForwardAndReverse){
float minSpeed = 0.1f;
float minInput = 0.1f;

if (target.speed > minSpeed){
throttleInput = forwardInput;
brakeInput = reverseInput;}
else
{
if (reverseInput > minInput)
{
throttleInput = -reverseInput;
brakeInput = 0.0f;
}
 else if (forwardInput > minInput)
{
if (target.speed < -minSpeed)
{
throttleInput = 0.0f;
brakeInput = forwardInput;
```

```
}
else{
throttleInput = forwardInput;
brakeInput = 0;}}}}
else
{
bool reverse = Input.GetKey(KeyCode.LeftControl)
 || Input.GetKey(KeyCode.RightControl);

if (!reverse)
{
throttleInput = forwardInput;
brakeInput = reverseInput;
}
else
{
throttleInput = -reverseInput;
brakeInput = 0;
}
}

target.steerInput = steerInput;
target.throttleInput = throttleInput;
target.brakeInput = brakeInput;
target.handbrakeInput = handbrakeInput;

if (m_doReset)
{
target.ResetVehicle();
m_doReset = false;
}
}
```

Скрипт VehicleController.cs

```
void FixedUpdate ()
{
if (paused)
{
if (!m_singleFixedStep) return;
m_singleFixedStep = false;
}

if (!disallowRuntimeChanges)
ConfigureCenterOfMass();
```

```
throttleInput = Mathf.Clamp (throttleInput, -1.0f, +1.0f);
brakeInput = Mathf.Clamp01(brakeInput);
handbrakeInput = Mathf.Clamp01(handbrakeInput);

if (m_referenceCandidateCount > m_wheelData.Length/2)
m_referenceBody = m_referenceCandidate;

Vector3 currentVelocity = m_rigidbody.velocity;
if (m_referenceBody != null) currentVelocity -= m_referenceBody.velocity;

m_speed = Vector3.Dot(currentVelocity, m_transform.forward);
m_speedAngle = Vector3.Angle(currentVelocity, m_transform.forward) *
Mathf.Sign(Vector3.Dot(currentVelocity, m_transform.right));

float referenceDownforce =
computeExtendedTireData? (m_rigidbody.mass * Physics.gravity.magnitude)
/ m_wheelData.Length : 1.0f;

bool needUpdateVisuals =
wheelUpdateRate == UpdateRate.OnFixedUpdate && wheelPositionMode ==
PositionMode.Fast;

int groundedWheels = 0;
m_referenceCandidateCount = 0;

if (needUpdateVisuals)
ComputeSteerAngle();
foreach (WheelData wd in m_wheelData)
{
if (!disallowRuntimeChanges)
UpdateWheelCollider(wd.collider);

if (needUpdateVisuals)
UpdateSteering(wd);

UpdateSuspension(wd);
UpdateLocalFrame(wd);
UpdateGroundMaterial(wd);

ComputeTireForces(wd);
ApplyTireForces(wd);

UpdateWheelSleep(wd);
```

```
if (needUpdateVisuals)
UpdateTransform(wd, Time.deltaTime);

if (wd.grounded) groundedWheels++;

if (computeExtendedTireData)
ComputeExtendedTireData(wd, referenceDownforce);
}

float sqrVelocity = m_rigidbody.velocity.sqrMagnitude;
Vector3 normalizedVelocity = m_rigidbody.velocity.normalized;
float forwardVelocityFactor = Vector3.Dot(m_transform.forward,
normalizedVelocity);

Vector3 dragForce = -aeroDrag * sqrVelocity * normalizedVelocity;
Vector3 loadForce = -aeroDownforce * sqrVelocity * forwardVelocityFactor
* m_transform.up;

Vector3 aeroAppPoint = m_transform.TransformPoint(new Vector3(0.0f,
m_rigidbody.centerOfMass.y,
Mathf.Lerp(m_vehicleFrame.rearPosition, m_vehicleFrame.frontPosition,
aeroBalance)));

m_rigidbody.AddForceAtPosition(dragForce, aeroAppPoint);
if (groundedWheels > 0) m_rigidbody.AddForceAtPosition(loadForce,
aeroAppPoint);

if (processContacts)
HandleImpacts();
}
```

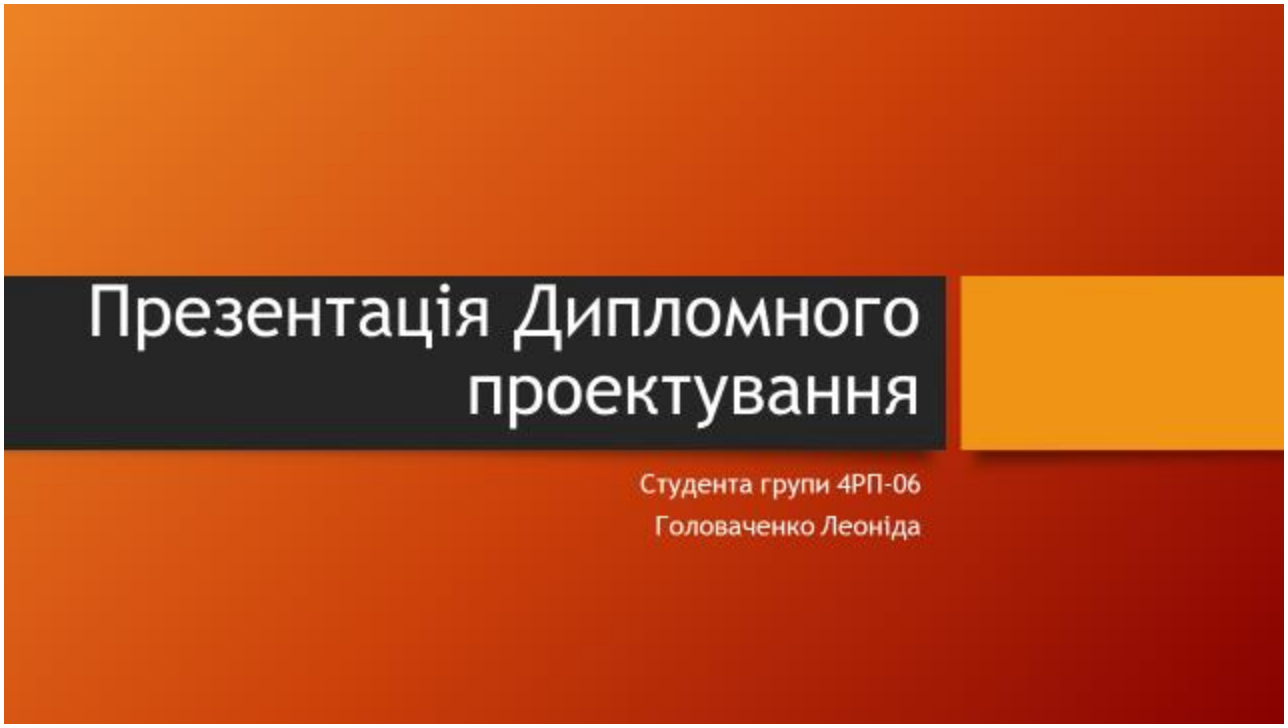
Скрипт CameraController.cs

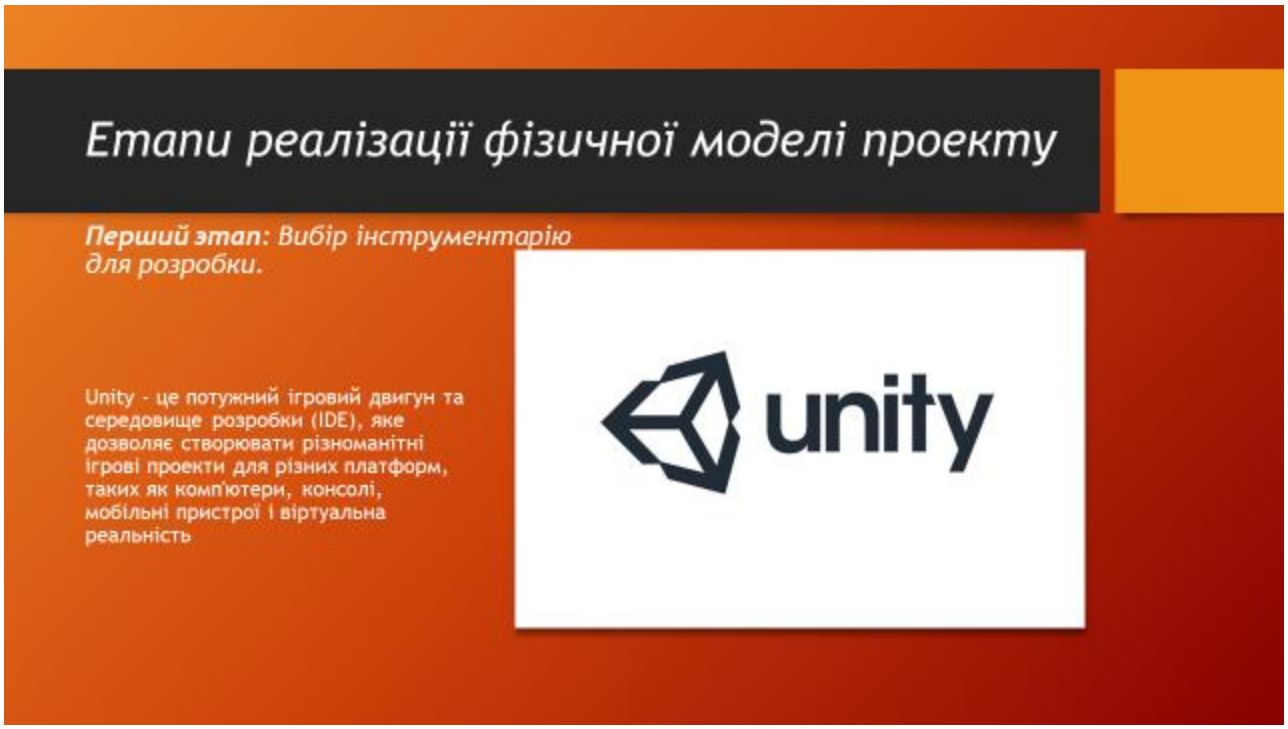
```
public class CameraFree : CameraMode
{
public float minVerticalAngle = -60.0f;
public float maxVerticalAngle = 60.0f;
public float horizontalSpeed = 4f;
public float verticalSpeed = 2f;
public float damping = 4.0f;
[Space(5)]
public bool adjustFov = true;
public float minFov = 10.0f;
public float maxFov = 60.0f;
public float fovSpeed = 20.0f;
```

```
public float fovDamping = 4.0f;
[Space(5)]
public bool adjustNearPlane = false;
public float nearPlaneAtMinFov = 1.5f;
[Space(5)]
public string horizontalAxis = "Mouse X";
public string verticalAxis = "Mouse Y";
public string fovAxis = "Mouse ScrollWheel";
[Space(5)]
public bool enableMovement = false;
public float movementSpeed = 2.0f;
public float movementDamping = 5.0f;
public string forwardAxis = "";
public string sidewaysAxis = "";
public string upwardsAxis = "";

Camera m_camera;
Vector3 m_position;
float m_fov = 0.0f;
float m_savedFov = 0.0f;
float m_savedNearPlane = 0.0f;

float m_horizontal;
float m_vertical;
public override void Initialize (Transform self)
{
m_camera = self.GetComponent<Camera>();

if (m_camera == null)
m_camera = self.GetComponentInChildren<Camera>();
}
public override void OnEnable (Transform self, Transform target, Vector3
targetOffset)
{
m_position = self.position;
Vector3 angles = self.eulerAngles;
m_horizontal = angles.y;
m_vertical = -angles.x;

if (m_camera != null)
{
m_fov = m_camera.fieldOfView;
m_savedFov = m_camera.fieldOfView;
m_savedNearPlane = m_camera.nearClipPlane;
```

```
        }
        }
        public override void Update (Transform self, Transform target, Vector3
        targetOffset, float deltaTime)
        {
        m_horizontal += GetInputForAxis(horizontalAxis) * horizontalSpeed;
        m_vertical += GetInputForAxis(verticalAxis) * verticalSpeed;
        m_vertical = Mathf.Clamp(m_vertical, minVerticalAngle, maxVerticalAngle);

        if (enableMovement)
        {
        float stepSize = movementSpeed * deltaTime;
        m_position += GetInputForAxis(forwardAxis) * stepSize * new
        Vector3(self.forward.x, 0.0f, self.forward.z).normalized;
        m_position += GetInputForAxis(sidewaysAxis) * stepSize * self.right;
        m_position += GetInputForAxis(upwardsAxis) * stepSize * self.up;
        }
        self.position = Vector3.Lerp(self.position, m_position, movementDamping *
        deltaTime);
        self.rotation = Quaternion.Slerp(self.rotation, Quaternion.Euler(-m_vertical,
        m_horizontal, 0), damping * deltaTime);
        if (m_camera != null)
        {
        m_fov -= GetInputForAxis(fovAxis) * fovSpeed;

        m_fov = Mathf.Clamp(m_fov, minFov, maxFov);
        if (adjustFov)
        m_camera.fieldOfView = Mathf.Lerp(m_camera.fieldOfView, m_fov,
        fovDamping * deltaTime);

        if (adjustNearPlane)
        {
        m_camera.nearClipPlane = Mathf.Lerp(m_savedNearPlane,
nearPlaneAtMinFov,
        Mathf.InverseLerp(maxFov, minFov, m_camera.fieldOfView));}}}
        public override void OnDisable (Transform self, Transform target, Vector3
        targetOffset)
        {
        if (m_camera != null)
        {
        m_camera.fieldOfView = m_savedFov;
        m_camera.nearClipPlane = m_savedNearPlane;}}}
```

ДОДАТОК Б СЛАЙДИ МУЛЬТИМЕДІЙНОЇ ПРЕЗЕНТАЦІЇ





Особливості ігрового двигуна Unity

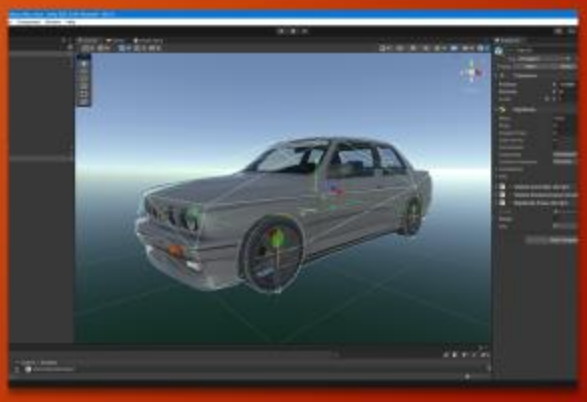
Головною превагою обирання цього двигуна були:

- Широкий вибір ресурсів
- Фізика
- Система компонентів
- Візуальний редактор
- Кросплатформеність

Етапи реалізації фізичної моделі проекту

***Другий этап**: Написання коду програми.*

На цьому етапі розробляється код для нашого автомобіля. Від цього коду і залежить як буде вести себе автомобіль



Особливості використання фізики машин у комп'ютерній ігровій індустрії

Для роботи фізики автомобіля потрібно декілька речей, а саме: Collider та Скрипт

Пояснення функіональних схем роботи фізичної моделі автомобілю у проекті

На наступних слайдах уд продеманстровано функціональні схеми мого програмного продукту.

Кут повороту, є важливим кроком у розробці алгоритму руху автомобіля, оскільки він дозволяє враховувати фізичні закони та параметри автомобіля для досягнення бажаної поведінки на дорозі.

Аналіз прискорення уважно враховує параметри автомобіля, такі як максимальна швидкість, режим центру маси, коефіцієнти тертя та інші характеристики. Це дозволяє створити реалістичну модель руху автомобіля.

При аналізі гальмування в алгоритмі руху автомобіля враховуються фізичні закони та параметри автомобіля.

Це дозволяє автомобілю знижувати швидкість або зупинятися відповідно до вхідних значень гальмування.

Етапи реалізації фізичної моделі проекту

Третій этап: Імплементація системи керування автомобілем.

Цей етап є завершальним, йдеться імплементація всього що ми розробили та додається у редактор.

Скріншоти результату розробки

Результат роботи.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Головаченко Леоніда Дмитровича

(прізвище, ім`я та по батькові)

Спеціальність 121 " Інженерія програмного забезпечення "

Освітня програма «Розробка програмного забезпечення»

Керівник дипломного проекту (роботи) Джабраілов Дмитро Володимирович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку*

Обсяг розрахунково-пояснювальної записки 76 сторінок

Обсяг графічної (презентаційної) частини 11 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною в сфері комп'ютерних ігор та присвячена питанням розробки та реалізації фізичної моделі автомобілю в комп'ютерних іграх.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуто питання загальної ігрової фізики та фізики автомобілів, проведено вибір середи розробки, представлені аналоги фізики автомобілів в іграх. Виконано проектування та розробку робочої моделі фізики автомобілю в грі та рівня для її тестування на відмінному рівні.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально розглянуті питання ігрової фізики автомобілів;
2. Обґрунтований вибір середи розробки;
3. Фізична модель застосована до якісної моделі автомобілю, що підвищує якість симуляції її фізики.

д) основні недоліки дипломного проекту (роботи)

1. Не вистачає симуляції фізики для різних видів грунтів.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента Царьов Роман Юрійович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, старший викладач кафедри комп'ютерної інженерії та інформаційних систем

Підпис:

« 16 » червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ТА ЗВ'ЯЗКУ * ВІДДІЛ КАДРІВ * 43997335 *

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Головаченко Леонід Дмитрович

(прізвище, ім`я та по батькові)

Спеціальність: 121 "Інженерія програмного забезпечення"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 78 сторінок. У пояснювальній записці виконано опис предметної області, та способи реалізації фізичних моделей в ігровій індустрії. Проведено проектування та реалізація фізичної моделі автомобіля. Графічна частина складається з 10 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Головаченко Л.Д. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Головаченко Л.Д. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Головаченко Л.Д. мав змогу самостійно приймати рішення з реалізації фізичної моделі автомобілю у 3D-просторі, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою Visual Studio, працювати у ігровому двигуні Unity.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Джабраілов Дмитро Володимирович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач комісії комп'ютерних технологій та програмної інженерії

Підпис

«09» 06 2023 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Головаченко Леонід Дмитрович,
здобувач освіти гр. 4РП-06, та

Джабраілов Дмитро Володимирович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту молодшого спеціаліста на тему:

«Реалізація фізичної моделі автомобілю в 3D-просторі для ігрового додатку (автор роботи – Головаченко Л.Д., керівник роботи – Джабраілов Д.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Головаченко Л.Д. /

Керівник _____ / Джабраілов Д.В./

«09» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015551718

Дата перевірки:
11.06.2023 22:46:57 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2023 23:06:33 EEST

ID користувача:
100011688

Назва документа: **4РП-06 Головаченко Л.Д.**

Кількість сторінок: **57** Кількість слів: **9172** Кількість символів: **69349** Розмір файлу: **10.59 MB** ID файлу: **1015203966**

8.97%
Схожість

Найбільша схожість: **2.22%** з Інтернет-джерелом (http://elcat.pnpu.edu.ua/docs/%d0%9c%d0%90%d0%a2%d0%95%d0%...

8.97% Джерела з Інтернету **544** ..... Сторінка 59

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи **33**