Міністерство освіти і науки України

Університет митної справи та фінансів

Факультет інноваційних технологій

Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему :«Ігровий додаток в жанрі «Симулятор Далекобійника» на основі рушія Unreal Engine»

Виконав: студент групи ІПЗ21-2

Спеціальність 121 «Інженерія програмного забезпечення»

Петренко Вадим Олегович

(прізвище та ініціали)

Керівник к.т.н., доц. Ульяновська Ю. В

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Дніпровський державний технічний університет

(місце роботи)

завідувач кафедри програмного забезпечення систем

(посада)

к.т.н., доцент Жульковський О.О.

(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Петренко В.О. Розробка ігрового додатку в жанрі Симулятор Далекобійника на основі рушія Unreal Engine.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2025.

Об'єктом дослідження є процес розробки ігрового програмного забезпечення.

Предмет дослідження – створення ігрового додатку в жанрі симулятор далекобійника із використанням рушія Unreal Engine.

Мета кваліфікаційної роботи полягає в моделюванні професійної діяльності далекобійника щляхом розробки ігрового симуляційного програмного забезпечення що поєднує інноваційні підходи до фізичного моделювання, геймплейної логіки та 3D-візуалізації на основі сучасного ігрового рушія Unreal Engine.

У роботі проведено аналіз сучасних ігрових рушіїв та обґрунтовано вибір Unreal Engine як платформи для реалізації симулятора. Розглянуто основні принципи створення 3D-ігрового світу, розробки механіки керування транспортом, взаємодії з навколишнім середовищем та впровадження сценаріїв геймплею. Особливу увагу приділено створенню користувацького інтерфейсу, оптимізації продуктивності та використанню візуальної системи скриптів Blueprints для реалізації ігрової логіки та інтерактивних елементів.

Практична цінність дипломної роботи полягає у створенні базового ігрового додатку, що може слугувати основою для подальшого розширення функціоналу або використання як навчального прикладу в галузі геймдеву.

Ключові слова: Unreal Engine, симулятор, ігровий додаток, розробка ігор, 3D-моделювання, геймплей, логістика, Blueprints.

ABSTRACT

Petrenko V.O. Development of a Truck Driving Simulator Game Application Based on the Unreal Engine.

Diploma thesis (project) for obtaining the degree of Bachelor in in speciality 121 «Software Engineering» – University of Customs and Finance, Dnipro, 2025.

The object of the study is the process of developing game software.

The subject of the study is the creation of a truck driving simulator game application using the Unreal Engine.

The aim of the qualification project is to simulate the professional activity of a truck driver by developing a game-based simulation software that integrates innovative approaches to physical modeling, gameplay logic, and 3D visualization using the modern game engine Unreal Engine.

The thesis includes an analysis of modern game engines and substantiates the choice of Unreal Engine as the development platform for the simulator. It explores the main principles of creating a 3D game world, developing vehicle control mechanics, interactions with the environment, and implementing gameplay scenarios. Special attention is paid to the creation of the user interface, performance optimization, and the use of Unreal Engine's Blueprint visual scripting system for implementing game logic and interactions.

The practical value of this work lies in the development of a basic game application that can serve as a foundation for further feature expansion or as an educational example in the field of game development.

Keywords: Unreal Engine, simulator, game application, game development, 3D modeling, gameplay, logistics, Blueprints.

ЗМІСТ

ВСТУП

Сучасна індустрія відеоігор є однією з найдинамічніших галузей цифрової економіки, що активно розвивається завдяки постійному вдосконаленню технологій, зростаючому попиту серед користувачів та розширенню сфер застосування ігор — від розваг до навчання та моделювання професійної діяльності. У 2024 році загальний дохід галузі сягнув $177,9 мільярда, а до 2027 року прогнозується його зростання до понад $213 мільярдів. Однією з ключових тенденцій є розвиток жанру симуляторів — ігор, що дозволяють моделювати реальні процеси у віртуальному середовищі.

Симулятори далекобійника, такі як Euro Truck Simulator 2 чи Truck & Logistics Simulator, стабільно утримують високі позиції в рейтингах продажів завдяки поєднанню ігрової складової з елементами логістики, реалістичного керування транспортом та взаємодії з навколишнім середовищем. Проте багато сучасних симуляторів мають обмежений функціонал або застарілу графіку, що знижує загальний рівень занурення.

У зв'язку з цим, створення нового ігрового продукту в жанрі симулятору далекобійника з використанням сучасних можливостей рушія Unreal Engine 5, таких як Nanite, Lumen, Chaos Vehicle та Blueprint-візуальне програмування, є актуальним і перспективним напрямом. Unreal Engine дозволяє створювати візуально насичене середовище, детальну фізику руху та розширювану логіку, що робить його ідеальним інструментом для реалізації ігор цього типу.

Мета кваліфікаційної роботи полягає в моделюванні професійної діяльності далекобійника щляхом розробки ігрового симуляційного програмного забезпечення що поєднує інноваційні підходи до фізичного моделювання, геймплейної логіки та 3D-візуалізації на основі сучасного ігрового рушія Unreal Engine.

Методи дослідження: огляд і порівняльний аналіз ігрових рушіїв (аналітичний метод), моделювання 3D-об'єктів у Blender (практичний метод), створення текстур у Substance 3D Painter, візуальне програмування за допомогою

Blueprints, тестування продуктивності та взаємодії елементів гри в середовищі Unreal Engine.

У відповідності до поставленої мети, в кваліфікаційній роботі були визначені наступні завдання:

1) Проаналізувати жанрові особливості та існуючі рішення у сфері симуляторів вантажного транспорту.

2) Обґрунтувати вибір рушія та інструментів розробки.

3) Реалізувати базову ігрову сцену з відкритим світом та трафіком.

4) Створити фізично коректну модель транспортного засобу.

5) Реалізувати інтерфейс, меню, HUD та систему підказок.

6) Провести тестування продуктивності та якості симуляції.

Об'єктом дослідження є процес розробки ігрового програмного забезпечення.

Предметом дослідження є створення ігрового додатку в жанрі симулятора далекобійника з використанням рушія Unreal Engine.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел з 16 найменувань. Загальний обсяг роботи — 63 сторінок, містить 30 рисунків і 3 таблиці.

Практичне значення результатів полягає у створенні базової версії симулятора, яка може бути розширена до повноцінного комерційного продукту.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

Ігрова індустрія активно розвивається і стабільно демонструє стрімке зростання. Відеоігри стали невід'ємною частиною культури та інструментом не лише для розваг, але й для навчання, тренування навичок, психологічного розвантаження тощо. Особливу нішу на ринку займають симулятори – жанр, що дозволяє моделювати реальні процеси або професії у віртуальному середовищі. Одним з популярних піджанрів симуляторів є ігри, що імітують професію водія вантажного транспорту – так звані симулятори далекобійника.

Актуальність розробки власного симулятора далекобійника на базі рушія Unreal Engine зумовлена:

- високим інтересом користувачів до реалістичних симуляторів;
- потребою у створенні вітчизняного продукту, здатного задовольнити сучасні вимоги гравців;
- розширеними можливостями рушія Unreal Engine для створення 3D-середовищ, фізичних моделей та інтерактивної логіки;
- навчальною та дослідницькою цінністю проєкту у межах підготовки бакалавра за спеціальністю 121 «Інженерія програмного забезпечення».

Глобальна ігрова індустрія у 2025 році залишається однією з найбільш динамічно зростаючих сфер цифрових технологій. Незважаючи на незначне зниження доходів у 2024 році, загальний обсяг ринку відеоігор становив $177,9 мільярда. Прогнозується, що до 2027 року цей показник зросте до $213,3 мільярда, а кількість активних гравців сягне 3,8 мільярда осіб [1].

Одним з популярних напрямів у сучасному геймдеві є ігри в жанрі симуляторів, зокрема симулятори професійної діяльності. Вони забезпечують гравцям ефект присутності, високий рівень занурення та можливість взаємодіяти з віртуальним світом у наближених до реальності умовах. Симулятори далекобійника, такі як Euro Truck Simulator 2 та American Truck Simulator,

користуються незмінною популярністю протягом багатьох років, демонструючи високі показники продажів та активної аудиторії.

У свою чергу, рушій Unreal Engine залишається однією з провідних платформ для розробки 3D-ігор завдяки своїй продуктивності, високій якості візуалізації, широкому інструментарію та активній підтримці спільноти. За даними звіту Video Game Insights, у 2024 році понад 28% нових ігор у Steam було створено з використанням Unreal Engine, що свідчить про високий попит на дану технологію серед розробників [2].

Отже, розробка ігрового застосунку в жанрі симулятор далекобійника з використанням рушія Unreal Engine є актуальною, оскільки поєднує в собі:

- Стабільне зростання глобального ігрового ринку;
- Популярність жанру симуляторів як засобу моделювання реального досвіду;
- Технологічну перевагу Unreal Engine у створенні складних інтерактивних 3D-систем.

1.2 Аналіз жанрових характеристик симуляторів

Симулятори як жанр комп'ютерних ігор відіграють важливу роль у сучасній гейм-індустрії. Вони моделюють реальні або умовні процеси, даючи змогу користувачам імітувати професійну або життєву діяльність, що робить даний жанр не лише розважальним, але й освітнім та тренувальним інструментом.

Жанр симуляторів поділяється на декілька категорій: транспортні симулятори (водіння, авіаперельоти), економічні симулятори, симулятори виживання, будівництва, побуту тощо. Найбільшу нішу на ринку посідають саме транспортні симулятори, серед яких особливо виділяється піджанр симуляторів вантажних перевезень (далекобійників). Їх популярність зумовлена тим, що вони поєднують керування складною технікою, дотримання дорожніх правил, логістику та планування маршрутів. Такі ігри, як Euro Truck Simulator 2 та Truck

& Logistics Simulator, продовжують займати високі позиції в рейтингах продажів і мають активну геймерську спільноту [3].

Інтерес до даного жанру обумовлений також соціальними та культурними чинниками: гравців приваблює можливість спробувати себе в новій професії, не виходячи з дому. Крім того, симулятори сприяють розвитку уваги, просторового мислення, планування та моторики.

Водночас, незважаючи на достатню кількість існуючих проєктів, багато з них мають застарілу графіку, обмежений функціонал або слабку фізичну модель. Сучасні тенденції вимагають реалістичної візуалізації, достовірної фізики керування транспортом, інтерактивного середовища та відкритого світу. Це створює передумови для розробки нового ігрового продукту, здатного задовольнити ці потреби.

Технічні можливості рушія Unreal Engine надають широкі перспективи для реалізації подібного функціоналу. Завдяки підтримці фізичного моделювання, систем освітлення та розширеного керування, він є доцільною платформою для створення симуляторів нового покоління.

Процес розробки ігрового застосунку в жанрі симуляторів має ряд специфічних вимог, що відрізняють його від інших жанрів, таких як шутери або платформери. Основні функціональні особливості полягають у наступному:

1. Реалістична фізика — ключовий компонент жанру. Необхідно моделювати вагу транспортного засобу, силу тертя, гальмівний шлях, відчуття керма тощо.

2. Ігрові меню — наявність головного меню, меню налаштувань, меню паузи з можливістю збереження гри, виходу та перезапуску, що забезпечує зручність взаємодії користувача з ігровим застосунком.

3. Підтримка відкритого світу — безперервність руху без завантаження між локаціями є важливою складовою досвіду гравця.

У межах кваліфікаційної роботи ставиться завдання створити ігровий додаток у жанрі симулятору далекобійника з використанням рушія Unreal Engine. Основні функціональні компоненти, які мають бути реалізовані:

- базову фізичну модель руху вантажного транспорту;
- інтерактивне керування;
- головне меню та меню паузи;
- наявність дорожнього трафіку (NPC, окрім користувача);
- користувацький інтерфейс;
- відкритий світ з різноманітними локаціями.

Реалізація зазначених функцій дасть змогу створити якісний симулятор, що відповідатиме сучасним вимогам і матиме потенціал для подальшого розвитку.

1.3 Аналіз існуючих аналогів у жанрі симуляторів

На сучасному ринку відеоігор існує успішних ігрових застосунків у жанрі симуляторів далекобійників. Аналіз їхніх функціональних можливостей дозволяє виявити сильні й слабкі сторони існуючих продуктів, а також сформувати орієнтири для розробки власного застосунку.

Euro Truck Simulator 2 (рис. 1.1) — найвідоміший симулятор далекобійника, який занурює гравця у світ міжнародних вантажних перевезень. У грі реалізовано реалістичну фізику керування вантажівками, автентичні інтер'єри кабін, динамічну зміну погоди та доби, а також детально змодельовані європейські дороги й міста. Гравець може брати замовлення на перевезення різноманітних вантажів, обираючи оптимальні маршрути, враховуючи витрати на паливо, час доставки та стан транспортного засобу. У грі реалізовано зміну доби, просту погодну систему, розвинуту економіку та систему модифікацій. Попри значні переваги, такі як деталізована дорожня інфраструктура, реалізм фізики та активна спільнота, проєкт має і недоліки. Зокрема, рушій гри вже є застарілим і обмежує графічні можливості, а інтерактивність з навколишнім світом є обмеженою. Також відчувається нестача сценарного прогресу, що знижує мотивацію гравця на тривалий термін [4].

Рисунок 1.1 – Ігровий процес гри Euro Truck Simulator 2

Truck & Logistics Simulator (рис. 1.2) робить акцент на повному логістичному процесі — від початкового завантаження товарів до їхньої доставки за вказаною адресою. Гравець бере участь у всіх етапах роботи: самостійно керує навантажувачем або автонавантажувачем для завантаження палет, після чого перевозить вантаж на вантажівці, мікроавтобусі чи іншому відповідному транспорті. У процесі гри потрібно враховувати тип вантажу, способи його закріплення, вагу, балансування та маршрут, що робить ігровий процес максимально наближеним до реальної логістичної діяльності. Серед переваг можна відзначити розмаїття техніки, деталізацію логістичних процесів і механіки завантаження. Водночас графіка поступається конкурентам, відкритий світ обмежений, а фізика руху транспорту — нестабільна. Гра також має проблеми з оптимізацією, особливо на ПК-платформі. Варто також зазначити, що розробники регулярно випускають оновлення, які усувають помилки та додають нову техніку. Однак, незважаючи на оновлення, проблеми з оптимізацією залишаються актуальними. Через це навіть на потужних системах можуть спостерігатися фризи та падіння частоти кадрів. [5].

Рисунок 1.2 – Ігровий процес гри Truck & Logistics Simulator

Ще один цікавий представник жанру — On The Road: Truck Simulator (рис. 1.3), розроблений компанією Toxtronyx Interactive та випущений у 2021 році. Гра використовує рушій Unreal Engine 4 і орієнтована на реалістичне моделювання вантажоперевезень територією Німеччини. Unreal Engine забезпечує якісне візуальне оформлення, динамічні погодні умови та реалістичну географію. Разом з тим, проект страждає на нестачу контенту, має часті технічні збої та недопрацьовану геймплейну логіку. Відчувається брак сценарної мотивації, що обмежує тривалість залучення гравця [6].

Рисунок 1.3 – Ігровий процес гри On The Road

У таблиці 1.1 нижче наведено коротке порівняння трьох популярних симуляторів – Euro Truck Simulator 2, Truck & Logistics Simulator та On The Road. Порівнювались такі параметри, як графіка, фізика, наявність відкритого світу, економічні елементи та основні геймплейні можливості.

Таблиця 1.1

Порівняльна характеристика існуючих ігрових симуляторів

Назва гри	Рушій	Графіка	Фізика	Відкритий світ	Економіка	Геймплейні механіки
ETS2	Prism3D	Середня	Реалістична	Так	Так	Перевезення вантажів
Truck & Logistics	Unity	Середня	Варіюється	Частково	Так	Завантаження/доставка
On The Road	Unreal Engine 4	Висока	Посередня	Так	Частково	Перевезення, Погода

1.4 Визначення вимог до ігрового застосунку

Після аналізу жанрових характеристик симуляторів і вивчення актуальних прикладів ігор у цій сфері були сформульовані вимоги до створюваного ігрового застосунку. Ці вимоги охоплюють не лише функціональні можливості майбутнього програмного продукту, а й технічні особливості, загальну архітектуру системи, а також враховують очікування з боку кінцевого користувача. У процесі розробки було взято до уваги необхідність створення збалансованого, продуктивного й розширюваного ігрового середовища, здатного забезпечити глибоке занурення користувача у симуляцію професії далекобійника.

Цільовим призначенням проєкту є створення інтерактивного віртуального середовища, в якому користувач виконує роль водія вантажного транспорту. Проєкт моделює ключові аспекти цієї професії: планування маршруту, доставку вантажів, контроль за витратами пального, дотримання правил дорожнього руху та взаємодію з динамічним трафіком. Гравець повинен адаптуватися до змін у

дорожній ситуації, виконувати логістичні завдання та досягати певних цілей, що забезпечує не лише розважальну складову, а й навчальний ефект.

Ігровий застосунок орієнтований переважно на розважальний сегмент споживачів, однак він також може бути використаний у навчальному процесі — наприклад, у рамках професійної підготовки водіїв великовантажного транспорту або як інструмент популяризації логістичних професій. Цільовою аудиторією проєкту є користувачі віком від 16 років, які мають персональні комп'ютери з актуальними технічними характеристиками, необхідними для запуску ресурсомістких ігор з високим рівнем деталізації.

Структурно ігровий застосунок реалізується у вигляді клієнтського додатку, що включає в собі такі основні підсистеми: рушій Unreal Engine 5, на якому реалізується графіка, логіка та фізика гри; модуль управління вантажним транспортом з підтримкою параметрів керування, гальмування та ваги навантаження; система моделювання дорожнього трафіку, яка забезпечує інтерактивну взаємодію з іншими транспортними засобами; а також інтерфейс користувача, який поєднує аудіо-візуальні ефекти з функціональними панелями керування.

Основний функціонал включає модуль реалістичного керування вантажівкою, що імітує поведінку транспортного засобу з урахуванням швидкості, ваги вантажу, витрати пального, стану дороги тощо. Користувацький інтерфейс охоплює головне меню, меню паузи та елементи HUD, які відображають актуальну ігрову інформацію. Додатково передбачена система підказок, яка допомагає гравцю орієнтуватися в інтерфейсі та реагувати на події під час сесії. Рух інших транспортних засобів реалізований через систему трафіку, яка базується на наперед заданих маршрутах та сценаріях поведінки NPC.

До нефункціональних вимог відноситься забезпечення стабільної продуктивності на комп'ютерах середнього рівня, оптимізація ігрового процесу для плавної роботи навіть при високій деталізації графіки, а також підтримка розширюваності. У майбутньому застосунок має дозволяти додавання нового

контенту — карт, транспортних засобів, механік, сценаріїв. Інтерфейс користувача повинен залишатися інтуїтивно зрозумілим та зручним для користувача незалежно від досвіду в подібних іграх.

Вимоги до технічного забезпечення:

- Операційна система: Windows 10 або новіша
- ЦП: Intel Core i5-10400F / AMD Ryzen 5 3600
- ОЗП: від 8 ГБ (рекомендовано 16 ГБ)
- Відеокарта: NVIDIA GTX 1660 / AMD RX 580 або новіші
- Місце на диску: мінімум 20 ГБ

Архітектура програмного забезпечення базується на клієнтській моделі. В якості рушія використовується Unreal Engine 5, який забезпечує підтримку графічних технологій нового покоління, системи фізичного моделювання Chaos, освітлення Lumen та інші можливості. Логіка гри реалізується переважно засобами Blueprints, однак у складних випадках допускається використання C++. Для забезпечення сумісності використовуються DirectX 12, Windows SDK, а також набір офіційних плагінів UE5, які відповідають за фізику, AI, UI-компоненти тощо.

Користувацький інтерфейс включає стандартну ігрову навігацію — головне меню, меню паузи, налаштування параметрів графіки, звуку, керування, а також інформаційну панель HUD. Окрему роль відіграє система сповіщень та контекстних підказок, які оновлюються в реальному часі залежно від дій гравця. Вся система побудована з урахуванням потреб майбутнього розширення і забезпечення стабільної роботи незалежно від конфігурації апаратного забезпечення.

Застосування такого підходу до структурування вимог, модульності та архітектури гри дозволяє досягти цілісності програмного продукту, спрощує підтримку та майбутню модернізацію системи, а також формує надійне підґрунтя для успішної реалізації кваліфікаційної атестаційної роботи.

1.5 Висновок до першого розділу

У результаті дослідження предметної області встановлено актуальність теми розробки симулятора далекобійника на базі рушія Unreal Engine. Було з'ясовано, що жанр симуляторів, зокрема транспортних, займає вагоме місце в ігровій індустрії, демонструючи стійкий попит серед користувачів. Аналіз сучасних ігрових застосунків у цій ніші дозволив визначити ключові переваги та недоліки існуючих рішень, що у свою чергу дало змогу окреслити основні напрями вдосконалення.

Було проведено порівняльний аналіз існуючих аналогів, таких як Euro Truck Simulator 2, Truck & Logistics Simulator, On The Road: Truck Simulator. Це дозволило виокремити найкращі практики реалізації механік керування, побудови ігрового світу, системи трафіку, інтерфейсу, а також виявити недоліки, які варто врахувати під час розробки власного проєкту. Особливу увагу було приділено функціональним можливостям таких продуктів і їх впливу на досвід користувача.

Особливу увагу приділено технічним вимогам до майбутнього застосунку, таким як реалізація реалістичної фізичної моделі руху, інтерактивного керування, системи ігрових меню, NPC-дорожнього трафіку, підтримки відкритого світу та високоякісної графіки. Аналіз рушія Unreal Engine підтвердив його доцільність як технологічної бази для створення симулятора нового покоління, що відповідає сучасним стандартам візуалізації та геймплею.

Таким чином, у межах розділу було сформульовано концептуальні засади розробки ігрового застосунку, визначено функціональні та технічні вимоги до нього, а також окреслено завдання подальшого дослідження і проєктування.

РОЗДІЛ 2. АНАЛІЗ ТА ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ ГРИ

2.1 Вибір інструментів розробки та програмного середовища

Розробка сучасного ігрового застосунку потребує використання цілого комплексу інструментів, що охоплюють створення візуального контенту, програмну реалізацію логіки, управління ресурсами та підтримку життєвого циклу проєкту. У межах цього проєкту було сформовано набір програмних засобів, що забезпечують ефективну, гнучку та масштабовану розробку симулятора далекобійника.

2.1.1 Порівняння ігрових рушіїв для розробки симуляторів

Одним із найважливіших рішень при створенні ігрового продукту є вибір рушія, на основі якого реалізовуватиметься проєкт. Ігровий рушій визначає не лише технічні можливості майбутньої гри, а й рівень графіки, реалістичність фізики, швидкість розробки, підтримку масштабування, інтеграцію інтерфейсів, а також гнучкість і стабільність в майбутньому. Особливо актуальним це є для ігор жанру симулятор, де важливо забезпечити високий ступінь правдоподібності у взаємодії об'єктів, точне фізичне моделювання, плавність рухів, реалістичне освітлення та велику відкриту ігрову карту.

Unreal Engine 5, розроблений компанією Epic Games, підтримує найсучасніші технології графіки та симуляції. Однією з його основних переваг є Nanite — система віртуалізованої геометрії, яка дає змогу використовувати складні моделі без помітного навантаження на систему. Крім цього, система Lumen забезпечує глобальне освітлення та відображення в реальному часі, що суттєво покращує візуальне сприйняття гри. Вбудована фізика Chaos дозволяє реалістично моделювати поведінку транспортних засобів, взаємодію з середовищем, деформації та зіткнення. Також рушій має систему візуального програмування Blueprints, що дозволяє створювати ігрову логіку без написання

коду, що значно полегшує процес розробки. Unreal Engine підтримує всі сучасні платформи, у тому числі Windows, Android, iOS, VR/AR та ігрові консолі. Використання рушія є безкоштовним до досягнення $1 млн доходу [7].

Unity — багатоплатформовий рушій, що широко використовується для розробки мобільних, 2D- та 3D-ігор. Він має простий інтерфейс, широку документацію, активну спільноту та багатий магазин ресурсів Unity Asset Store. Однак при створенні симуляторів з великими відкритими світами рушій демонструє гіршу оптимізацію та графіку у порівнянні з UE5. Реалістичну фізику можливо реалізувати, але вона вимагає інтеграції сторонніх бібліотек або детального налаштування. Unity не має вбудованої системи візуального скриптингу (тільки через сторонні плагіни), тому основна логіка реалізується через C#. Хоча рушій добре підходить для інді-проєктів і мобільних додатків, для складного симулятора він менш зручний [8].

Godot є відкритим рушієм із відкритим кодом і повною відсутністю ліцензійних обмежень. Він має власну мову GDScript, підтримує 2D- і 3D-графіку, швидко завантажується та зручний для невеликих ігор. Для проєктів типу симулятора вантажного транспорту він поки не є оптимальним вибором, оскільки не забезпечує достатньої якості графіки, розширеної фізики або гнучкої роботи з великими сценами. Також відсутність розвиненої інфраструктури для реалістичного освітлення, транспортних систем і складної поведінки AI обмежує його використання у подібних задачах [9].

CryEngine — рушій із високою якістю візуалізації, який раніше використовувався у великих проєктах, таких як Crysis. Підтримує PBR-графіку, погодні ефекти, візуально насичені ландшафти, однак має високий поріг входу, складну структуру і слабку інтеграцію з інструментами візуального програмування. Через це він рідше використовується незалежними розробниками. Хоча CryEngine теоретично підходить для симуляторів, його використання потребує значних ресурсів та часу на адаптацію, що не відповідає цілям даної дипломної роботи[10].

Попри те, що кожен із розглянутих рушіїв має свої сильні та слабкі сторони, вибір конкретної платформи залежить від завдань проєкту, рівня очікуваної реалістичності, наявних ресурсів і досвіду розробника. Для ігор у жанрі симуляторів важливою є не лише якість графіки, а й можливість обробки великої кількості об'єктів, детальної фізики транспорту, підтримки погодних умов, відкритого світу та оптимізації.

У таблиці 2.1 нижче відображено ключові характеристики рушіїв, що враховуються при виборі технологічної основи для симуляторів: якість графіки, фізика, оптимізація великих сцен, простота розробки, підтримка візуального скриптингу, документація та умови ліцензування.

Таблиця 2.1

Порівняльна характеристика рушіїв для створення симуляторів

Параметр	Unreal Engine 5	Unity	Godot	CryEngine
Графіка	Дуже висока	Середня	Низька	Висока
Фізика	Реалістична та стабільна	Обмежена	Базова	Реалістична
Простота освоєння	Складна для новачків	Зручна для початківців	Проста	Складна
Ліцензування	Безкоштовно до $1M	Безкоштовно / Pro	Повністю безкоштовний	Умовно безкоштовний
Підтримка Blueprints	Так	Ні	Ні	Ні
Оптимізація великих сцен	Висока	Задовільна	Низька	Висока
Підтримка VR/AR	Повна	Повна	Часткова	Повна

Проаналізовані рушії демонструють, що Unreal Engine 5 задовольняє вимоги до розробки симулятора далекобійника з високим рівнем графіки, реалістичною фізикою, відкритим світом та підтримкою масштабування. Тому було обрано Unreal Engine як базову технологічну платформу.

2.1.2 Інструменти для моделювання та дизайну контенту

Візуальна складова відіграє ключову роль у сприйнятті симулятора користувачем. Для створення реалістичного середовища, об'єктів, транспорту та інтерфейсів використовуються спеціалізовані інструменти, які дозволяють розробнику гнучко моделювати, текстурувати й оформлювати 3D- та 2D-елементи гри. У цьому проєкті були обрані три основні інструменти: Blender, Adobe Photoshop та Substance 3D Painter, які доповнюють один одного на різних етапах виробництва контенту.

Blender є потужним безкоштовним програмним забезпеченням з відкритим кодом, яке забезпечує повний цикл створення 3D-моделей — від базового полігонального моделювання до скульптингу, UV-розгортки, анімації та рендерингу. Він підтримує експорт у формати FBX, OBJ, GLTF, які легко інтегруються в Unreal Engine. Blender має зручні інструменти для створення як статичних об'єктів (вантажівки, будівлі, дорожня інфраструктура), так і анімованих (наприклад, об’єктів, що рухаються в трафіку). Завдяки плагінам та скриптам на Python, Blender може бути кастомізований для конкретних потреб ігрової розробки, а також інтегрований із іншими пакетами, зокрема Substance та Photoshop [11].

Adobe Photoshop використовується на етапі створення текстур, інтерфейсів та підготовки допоміжних матеріалів. За допомогою цього редактора розробляються такі елементи, як кнопки, іконки, HUD (Head-Up Display), екрани меню, а також ручна корекція текстур. Photoshop дозволяє працювати з шарами, масками, альфа-каналами та нормалями, що особливо корисно при створенні матеріалів для UE5. Крім того, можливість прямого експорту в TGA, PNG або PSD формат спрощує передачу графіки до рушія. Photoshop також часто використовується для створення концепт-арту, розмітки об'єктів та підготовки карт освітлення чи прозорості [12].

Substance 3D Painter (розробка Adobe) є галузевим стандартом у сфері процедурного текстурування 3D-об’єктів. Він дозволяє наносити реалістичні

матеріали, мікродеталі, бруд, потертості, металевість, подряпини та інші ефекти на моделі, імпортовані з Blender. Substance підтримує PBR-матеріали (Physically Based Rendering), що повністю відповідають системі рендерингу Unreal Engine. Крім цього, Painter дозволяє створювати текстурні комплекти з картами кольору, нормалей, шорсткості, металевості, емісії тощо, які експортуються безпосередньо у форматах, готових до використання в рушії. Це дає змогу значно скоротити час між створенням моделі та її інтеграцією у гру, забезпечуючи при цьому високий рівень фотореалістичності [13].

Загалом, комбінація Blender для 3D-моделювання, Photoshop для інтерфейсів, і Substance Painter для процедурного текстурування дозволяє створювати візуально насичений, деталізований ігровий світ, який повністю відповідає вимогам жанру симуляторів. Кожен із цих інструментів виконує свою унікальну функцію в пайплайні розробки контенту, а їх взаємна інтеграція забезпечує ефективність і гнучкість у роботі над ігровими активами.

2.2 Методи реалізації ключових компонентів гри

У процесі реалізації ігрового застосунку були створені базові функціональні компоненти, необхідні для забезпечення ігрового процесу симулятора. Реалізація виконувалась у середовищі Unreal Engine 5 із використанням візуального програмування Blueprints. Розробка охоплювала три ключові напрямки: керування вантажівкою, моделювання дорожнього трафіку та створення елементарного користувацького інтерфейсу.

2.2.1 Система керування вантажівкою

Для реалізації фізики транспортного засобу використовується система Chaos Vehicle — модуль рушія Unreal Engine, який надає набір інструментів для створення транспортних засобів з реалістичною поведінкою. Chaos Vehicle підтримує багатоколісні конфігурації, облік маси, інерції, гальмівної сили,

крутного моменту, зчеплення з дорогою та динаміки підвіски. На відміну від звичайної ієрархії об'єктів, Chaos Vehicle використовує компонент ChaosWheeledVehicleMovement, що дозволяє керувати параметрами кожного колеса окремо.[14]

Керування транспортом реалізовується через систему подій вводу. Команди, пов’язані з акселерацією, гальмуванням, поворотами, перемиканням передач або вмиканням реверсу, обробляються Blueprint-логікою. У Blueprint-редакторі створюються змінні для напрямку руху, гальма, керма, які через вузли (nodes) передаються до компонентів транспортного засобу. Для візуального сприйняття застосовуються окремі камери з можливістю перемикання між видами. Також додається логіка для вмикання світлових сигналів, звукового сигналу та інших ефектів.

2.2.2 Система дорожнього трафіку

Система трафіку реалізована із застосуванням AI-контролерів (AI Controllers) — спеціальних класів у Unreal Engine, які відповідають за автономну поведінку неігрових персонажів. AI-контролер зв’язується з транспортним засобом і виконує логіку його переміщення. У даному випадку AI-контролер керує рухом NPC-автомобілів вздовж заданої траєкторії, яка побудована за допомогою Spline-лінії.[14]

Реалізація трафіку передбачає створення Blueprint-класу автомобіля NPC, до якого прикріплюється сплайн-компонент. У межах Blueprint прописується цикл руху автомобіля по сплайну з певною швидкістю. За допомогою таймерів або подій можна реалізувати зміну напрямку, зупинки або реакцію на перешкоди (через виявлення об’єктів за допомогою ліній трасування — Line Trace). Для оптимізації системи передбачено, що NPC-автомобілі активуються лише поблизу до гравця (за допомогою тригерів або LOD). Такий підхід дозволяє моделювати умовно живе середовище з можливістю подальшого ускладнення логіки трафіку.

2.2.3 Користувацький інтерфейс

Графічний інтерфейс користувача в Unreal Engine реалізується за допомогою UMG (Unreal Motion Graphics) — вбудованого фреймворку для створення UI-елементів у редакторі. UMG надає інструменти для створення кнопок, текстових полів, слайдерів, іконок, панелей, а також логіки їхньої взаємодії з гравцем.

Для реалізації інтерфейсу створюються Widget Blueprints, у яких розміщуються графічні елементи (UI компоненти) на сцені. Кожен віджет має власну логіку, що пов'язується з ігровими подіями. Наприклад, у головному меню реалізуються кнопки "Почати гру", "Продовжити", "Вихід", які викликають відповідні події в Blueprint: завантаження рівня, збереження/відновлення гри або закриття застосунку. Меню паузи прив'язується до натискання клавіші Esc, після чого активується відповідний віджет.

Ігрова панель (HUD) реалізовується як окремий віджет, який прикріплюється до гравця або камери. Вона містить поля для виведення швидкості, стану пального, повідомлень, контекстних підказок тощо. Всі змінні оновлюються динамічно в процесі гри через Blueprint-зв'язки з геймплейними подіями. Передбачена можливість масштабування інтерфейсу під різні екрани завдяки використанню Canvas Panel та властивостей автоматичної адаптації (SizeBox, Scale Box).[14]

2.3 Висновок до другого розділу

У другому розділі було розглянуто та обґрунтовано вибір програмних засобів, технологій і підходів, що лягли в основу реалізації ігрового симулятора вантажного транспорту. Проведено порівняльний аналіз сучасних ігрових рушіїв, у результаті якого обрано Unreal Engine 5 як платформу, що найкраще відповідає вимогам до графіки, фізичного моделювання та гнучкості розробки. Його інструменти, такі як Chaos Vehicle, Blueprints, Nanite і Lumen, забезпечують

широкі можливості для створення ігрових середовищ із високим ступенем реалістичності та інтерактивності.

Окрему увагу приділено підбору інструментів для створення візуального контенту. Програмне забезпечення Blender, Adobe Photoshop і Substance 3D Painter дозволило реалізувати повний цикл 3D-моделювання, текстурування та оформлення графіки з подальшою інтеграцією в ігровий рушій.

Крім того, в межах цього розділу проаналізовано ключові методи реалізації базових систем гри, зокрема системи керування вантажівкою, дорожнього трафіку та інтерфейсу користувача. Визначено архітектурні підходи, описано принципи використання Chaos Vehicle для моделювання транспорту, AI-контролерів та сплайнів для створення NPC-трафіку, а також засоби UMG для побудови ігрового меню та HUD.

У результаті проведеного аналізу було сформовано технічно обґрунтовану основу для подальшої реалізації ігрового функціоналу відповідно до поставленої мети кваліфікаційної роботи. Обрані методи, засоби та рішення дозволяють забезпечити стабільну, гнучку та візуально привабливу реалізацію симулятора у середовищі Unreal Engine 5.

РОЗДІЛ 3. РОЗРОБКА ІГРОВОГО ЗАСТОСУНКУ У ЖАНРІ СИМУЛЯТОРА ДАЛЕКОБІЙНИКА

У цьому розділі розглянуто етапи практичної реалізації ігрового застосунку у жанрі симулятора далекобійника з використанням рушія Unreal Engine 5. Реалізація охоплює створення ігрової сцени, налаштування керованого транспортного засобу, побудову системи трафіку, розробку інтерфейсу користувача, впровадження базової логіки доставки вантажів та оптимізацію проєкту.

3.1 Створення 3D-моделей для ігрового середовища

Однією з ключових складових розробки ігрового застосунку є створення якісних 3D-об’єктів, що формують візуальне наповнення сцени. Від їхньої якості, оптимізації та відповідності стилю гри залежить сприйняття всього віртуального світу. Моделі мають бути не лише візуально привабливими, а й ефективними з технічного погляду — з оптимальною кількістю полігонів, правильною топологією та адаптованими текстурами для реального часу.

Основна частина моделювання виконувалась у середовищі Blender, яке дозволяє створювати повний цикл об’єкта — від базової геометрії до UV-розгортки. Подальша обробка, зокрема текстурування, здійснювалась у Substance 3D Painter, який дає змогу наносити фотореалістичні матеріали, ефекти потертості, бруду, металевих поверхонь тощо. Це дозволяє суттєво підвищити рівень деталізації навіть для моделей з невеликою кількістю полігонів.

Процес створення моделей поділяється на кілька послідовних етапів. По-перше, відбувається побудова геометрії — створення основної форми об’єкта шляхом полігонального моделювання. На цьому етапі визначаються пропорції, габарити, форма й рівень деталізації об’єкта відповідно до його призначення. Після цього виконується топологічна оптимізація: усуваються зайві вершини, об'єднуються співпадаючі точки, нормалізуються нормалі поверхонь, що

забезпечує стабільність при освітленні та шейдингу. Наступним кроком є створення UV-розгортки, яка дозволяє правильно нанести текстури. Це важливий етап, що впливає на якість відображення матеріалів у рушії. Розгортка виконується вручну з урахуванням логіки швів, масштабу та уніфікації островів. Далі модель експортується у форматі .FBX, який є стандартом для подальшого використання в інших середовищах, зокрема в Unreal Engine. Встановлюються параметри експорту: збереження нормалей, UV-координат, трансформацій тощо. Після цього відбувається етап текстурування у Substance 3D Painter. Використовуються текстурні карти: Base Color, Roughness, Normal Map, Metallic, Ambient Occlusion. При необхідності також застосовується запікання деталей з high-poly моделей, що дозволяє перенести мікродеталі на low-poly об'єкт. Заключним кроком є імпорт до рушія Unreal Engine, де модель отримує фінальне оформлення: застосування матеріалів, налаштування колізій, створення LOD, інтеграція у сцену. Детальніше цей етап описано в наступному підрозділі.

Під час створення моделей було враховано принципи оптимізації: полігональне навантаження кожного об'єкта зведено до мінімально необхідного рівня. Це забезпечило стабільну продуктивність гри навіть у випадках розміщення великої кількості об'єктів одночасно в сцені.

3.1.1 Моделювання об'єкта «Дорожній конус» у Blender

Для демонстрації процесу створення ігрової моделі було обрано простий об'єкт — дорожній конус, який часто зустрічається в іграх, пов'язаних з транспортом. Об'єкт має чітку форму, невелику кількість полігонів і водночас дозволяє відпрацювати базові техніки 3D-моделювання.

Процес розпочинається зі створення нової сцени в Blender. Для створення основи дорожнього конуса використано примітив Cone (вкладка Add → Mesh → Cone). У вікні параметрів (нижній лівий кут) було встановлено такі значення:

- Vertices (кількість сегментів основи): 24
- Radius1 (радіус основи): 0.3 м

- Radius2: 0 м (верхівка)
- Depth (висота): 0.6 м

Далі було додано цилиндричну основу (Add → Mesh → Cylinder), яка виконує функцію підставки. Вона має радіус дещо більший, ніж у основи конуса, та незначну висоту — приблизно 0.05 м. Обидва об’єкти об'єднуються в один (Ctrl + J), після чого переходять у редагування (Edit Mode) для подальшої обробки (рис. 3.1).

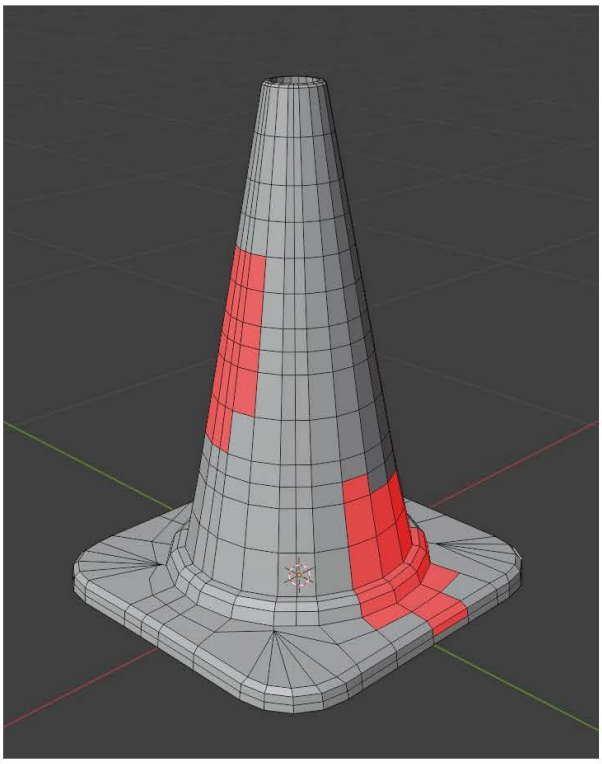

Рисунок 3.1 – Базова форма дорожнього конуса

Для досягнення правильного відображення світла і згладжених країв застосовується модифікатор Bevel (вкладка Modifiers → Add Modifier → Bevel). У налаштуваннях встановлюється невелике значення Width (0.01–0.015 м), а також 2–3 сегменти згладжування. Це дозволяє уникнути різких переходів у візуалізації при освітленні, що особливо помітно в ігровому рушії.

Далі виконується перевірка та очищення геометрії:

1) Merge by Distance (M → Merge by Distance) — об’єднання близьких вершин, щоб прибрати дублікати.

2) Shade Smooth — згладження поверхні (правий клік по об’єкту → Shade Smooth).

3) Normal Orientation — увімкнення відображення нормалей (Overlay → Face Orientation), що дозволяє переконатися у правильності напрямку поверхонь.

У разі виявлення проблем із нормалями (наприклад, частина полігонів має зворотне направлення), застосовується функція Recalculate Normals (в Edit Mode: Ctrl + N) (рис. 3.2).

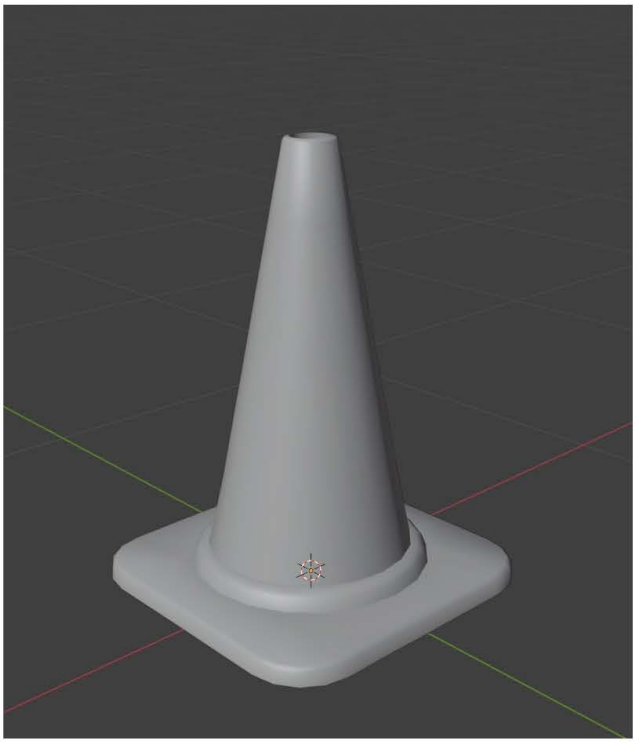

Рисунок 3.2 – Модифікований об’єкт із застосованим Bevel та перевіреними нормалями

Після очищення геометрії виконується об’єднання всіх трансформацій (Ctrl + A → Apply All Transforms), щоб переконатися, що масштаб і положення об’єкта коректно збережені перед експортом. Також проводиться фінальна перевірка на наявність внутрішніх полігонів, зайвих граней і неузгодженої топології.

Об’єкт зберігається в окремому .blend-файлі для подальшої UV-розгортки та текстурування (рис. 3.3).

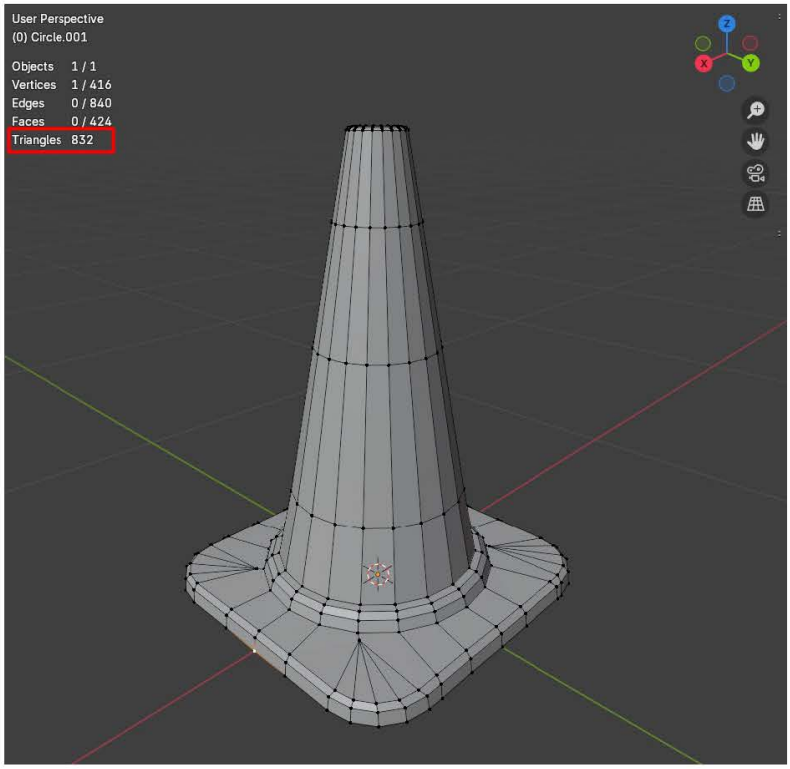

Рисунок 3.3 – Готова геометрія конуса перед UV-розгорткою

3.1.2 Створення high-poly моделі для запікання нормалей

Для досягнення більшої реалістичності поверхні та глибини деталей на low-poly моделі у подальшому етапі текстурування, часто створюється окрема високополігональна (high-poly) версія об’єкта. Вона не використовується безпосередньо в грі, але служить основою для запікання карт нормалей, висоти та амбієнт-оклюжену, що дозволяє передати дрібні деталі на оптимізованій моделі без збільшення геометрії.

Створення hi-poly моделі починається з дублювання основної геометрії (Shift + D → Enter), після чого нова копія розміщується на новому шарі або зберігається як окремий об’єкт. Для уникнення подальших плутанин бажано перейменувати копію, наприклад, на "Cone_High" (рис. 3.4).

До високополігональної моделі застосовуються модифікатори:

1) Bevel — для згладження усіх крайових ребер. Значення Width і кількість сегментів збільшуються для візуальної деталізації;

2) Subdivision Surface — додає додаткові рівні підрозділення геометрії. Для моделей, що запікатимуться, рекомендується 2–3 рівні (Levels Viewport: 2–3).

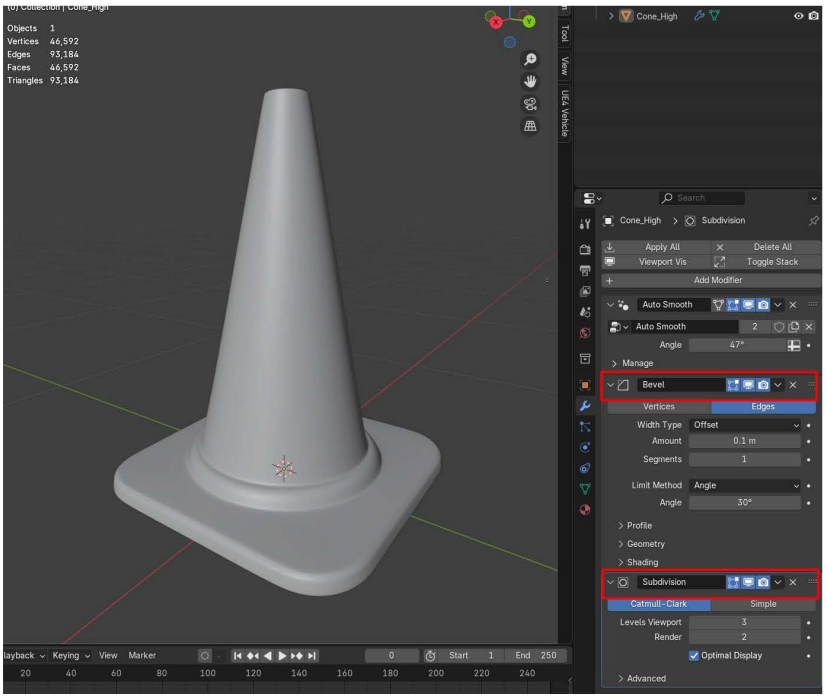

Рисунок 3.4 – Модель з активованими модифікаторами Bevel та Subdivision Surface

Для створення природного вигляду поверхні, на високополігональній моделі можна вручну додати дрібні нерівності, сколи, вм’ятини, подряпини тощо за допомогою інструментів Sculpt Mode:

- Draw — для додавання форм поверх рельєфу;
- Crease — для підсилення контурів;
- Smooth — для пом’якшення переходів;
- Blob / Clay Strips — для формування нерівностей.

Такі деталі не додають полігонів до фінальної low-poly моделі, але в подальшому будуть перенесені через карту нормалей, запечену в Substance 3D Painter (рис. 3.5).

Рисунок 3.5 – High-poly версія з доданими елементами скульптингу

В результаті цього етапу отримуємо дві версії однієї моделі:

- Low-poly — оптимізована модель для використання в рушії;
- High-poly — детальна модель для генерації карт нормалей.

Обидві версії зберігаються в окремих файлах для подальшого імпорту до Substance 3D Painter та запікання.

3.1.3 UV-розгортка та підготовка моделі до експорту

Після завершення етапів створення low-poly та high-poly версій об’єкта наступним кроком є формування UV-розгортки. Цей процес дає змогу «розгорнути» тривимірну геометрію моделі у двовимірному просторі, щоб у подальшому на неї можна було коректно накласти текстури. Від правильності розгортки залежить якість візуального відображення матеріалів, а також точність запікання нормалей, амбієнт-оклюжену та інших мап у Substance 3D Painter.

На початку роботи модель очищується від автоматично згенерованих або застарілих UV-даних. У режимі редагування (Edit Mode) виділяється вся геометрія об’єкта, після чого виконується команда Clear Seam, що скидає раніше встановлені шви. Далі вручну встановлюються нові шви (seams), які визначають межі розгортки. Вони задають, де саме буде розрізано поверхню моделі для перенесення її у двовимірну площину.

У випадку дорожнього конуса шви були розміщені у таких логічно обґрунтованих місцях: вздовж вертикальної поверхні конуса, щоб «розгорнути» основну частину; по краю круглої основи, а також по периметру нижньої циліндричної підставки. Така схема дає змогу отримати чисту та непрекривану UV-сітку, яка легко масштабується і дає хороший результат при текстуруванні (рис. 3.6).

Після розмітки швів виконується команда UV Unwrap, що створює саму розгортку. Далі в UV Editor відбувається оптимізація — масштабування, вирівнювання та розміщення елементів у межах координатного простору 0–1. Важливо забезпечити рівномірне щільне пакування UV-островів для ефективного використання текстурного простору.. Особлива увага приділяється тому, щоб уникнути перекриттів і надмірної деформації.

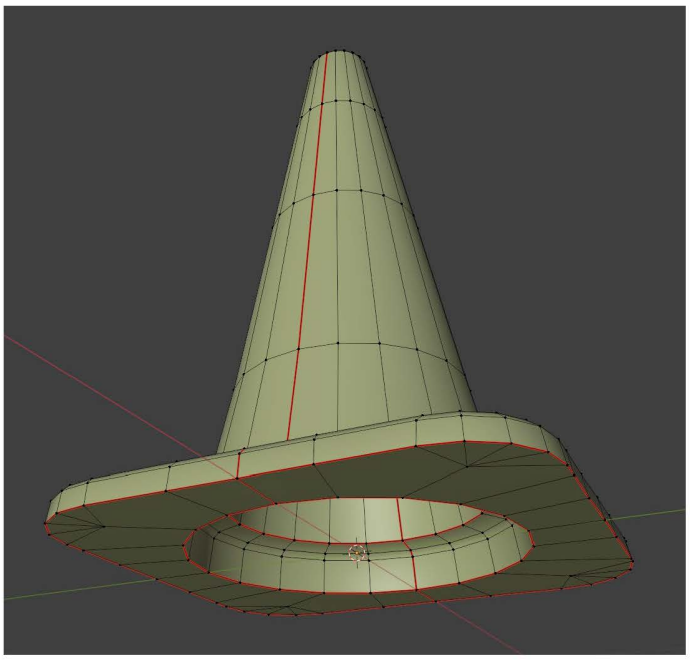

Рисунок 3.6 – Встановлення швів (Seams) для UV-розгортки

Після визначення швів об'єкт повністю виділяється (клавіша A в режимі редагування), після чого виконується команда Unwrap через меню UV (U → Unwrap). У вікні UV Editor автоматично створюється сітка UV-островів — частин поверхні моделі, розкладених у двовимірному просторі. Розгортку можна відредагувати вручну: змінити положення островів, відмасштабувати або обернути для досягнення оптимального заповнення текстурного простору (рис. 3.7).

У процесі перевірки важливо переконатися у відсутності перекриттів UV-островів, рівномірності масштабів елементів та достатній площі для важливих зон (наприклад, ділянок із логотипами або складною деталізацією). Для візуального контролю може бути застосована тестова сітка (Checker Texture) або команда Pack Islands, яка автоматично оптимізує розміщення островів в межах текстурного простору.

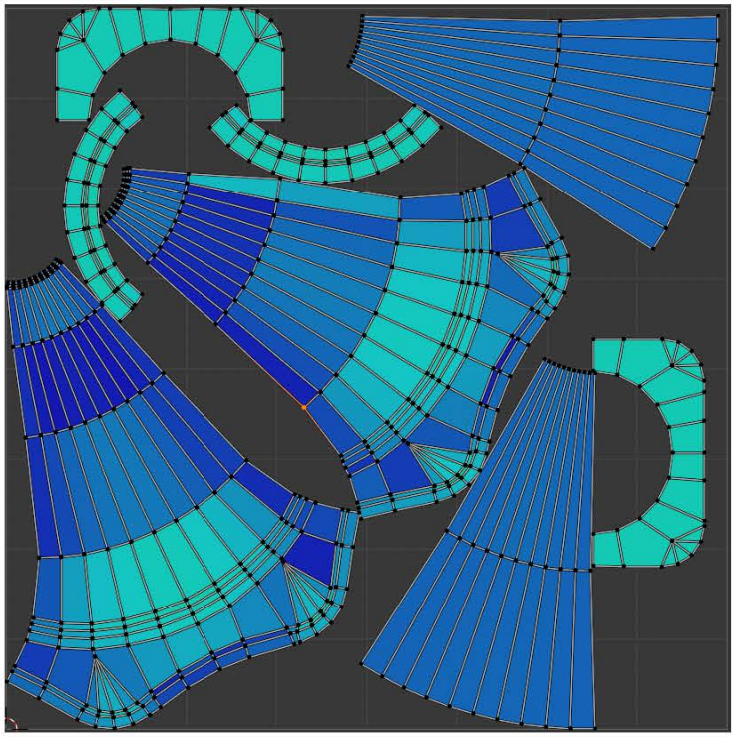

Рисунок 3.7 – Готова UV-розгортка об’єкта

Після завершення розгортки модель експортується у формат .FBX, який є найбільш сумісним як із текстурними редакторами, так і з ігровими рушіями. У

вікні експорту обираються лише активні об'єкти (опція Selected Objects), активується параметр Apply Transform, що зберігає масштаб, позицію та обертання об'єкта, а також встановлюється тип згладжування Smoothing: Face. Обов'язково переконуємося, що експорт включає UV-координати та нормалі.

У результаті формується два окремі файли: модель із низькою кількістю полігонів зберігається під назвою TraffConLP.fbx, а деталізована high-poly версія — як TraffConHP.fbx (рис. 3.8). Обидві моделі готові до імпорту в Substance 3D Painter, де буде виконане запікання нормалей і створення фінальних текстур на основі високодеталізованої геометрії.

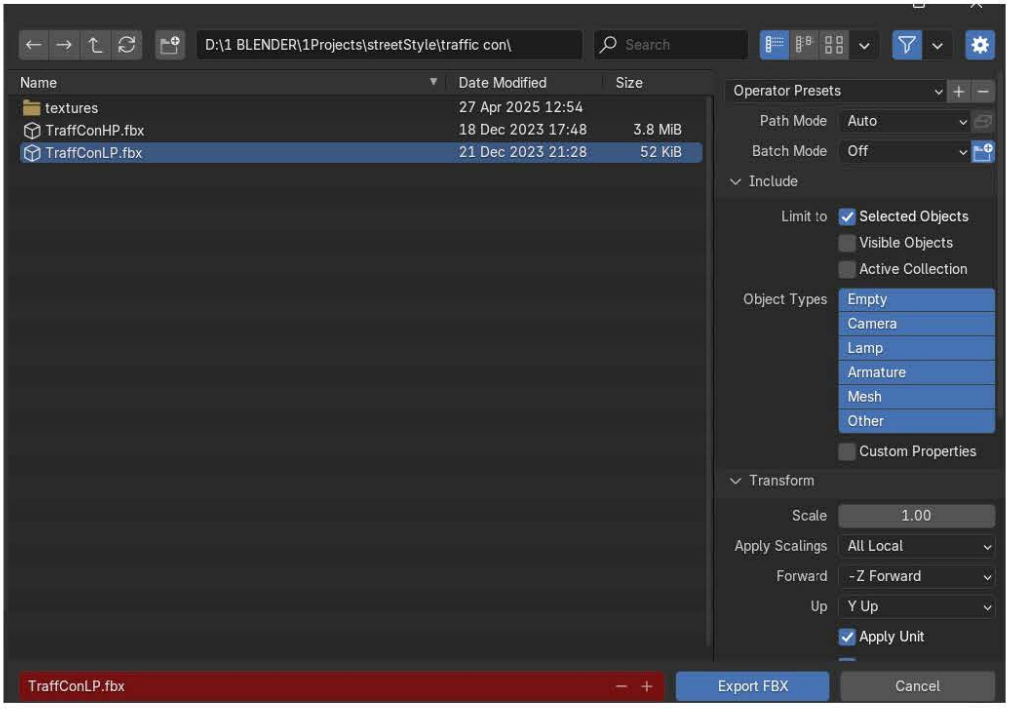

Рисунок 3.8 – Параметри експорту low-poly моделі у форматі FBX

3.1.4 Імпорт моделі до Substance 3D Painter та створення текстур

Після завершення процесів моделювання та створення UV-розгортки наступним етапом стало перенесення моделі до середовища текстурування — Substance 3D Painter. Саме в цьому середовищі здійснюється запікання високополігональних деталей на low-poly модель, а також створення фотореалістичних текстур за технологією PBR (Physically Based Rendering).

На першому етапі роботи в Substance Painter створюється новий проєкт (File → New) (рис. 3.9). У діалоговому вікні обирається шаблон PBR – Metallic Roughness — це найбільш універсальний підхід для сучасних ігрових рушіїв, зокрема Unreal Engine 5. Далі вказується шлях до low-poly моделі у форматі .FBX (у нашому випадку Cone_Low.fbx). Особливу увагу слід приділити налаштуванню нормалей у полі Normal Map Format. Необхідно вибрати варіант DirectX, оскільки Unreal Engine інтерпретує нормалі саме у цьому форматі. У разі вибору OpenGL текстури можуть візуально відображатися некоректно після імпорту в рушій.

Після імпорту виконується автоматичне розпізнавання UV-островів, перевіряється їх якість та заповнення простору. Якщо UV-сітка має перекриття або некоректно масштабовані елементи, це може негативно вплинути на якість запікання. Тому перед переходом до наступного етапу бажано переконатися, що усі частини об'єкта відображаються рівномірно, а текстурний простір заповнено ефективно.

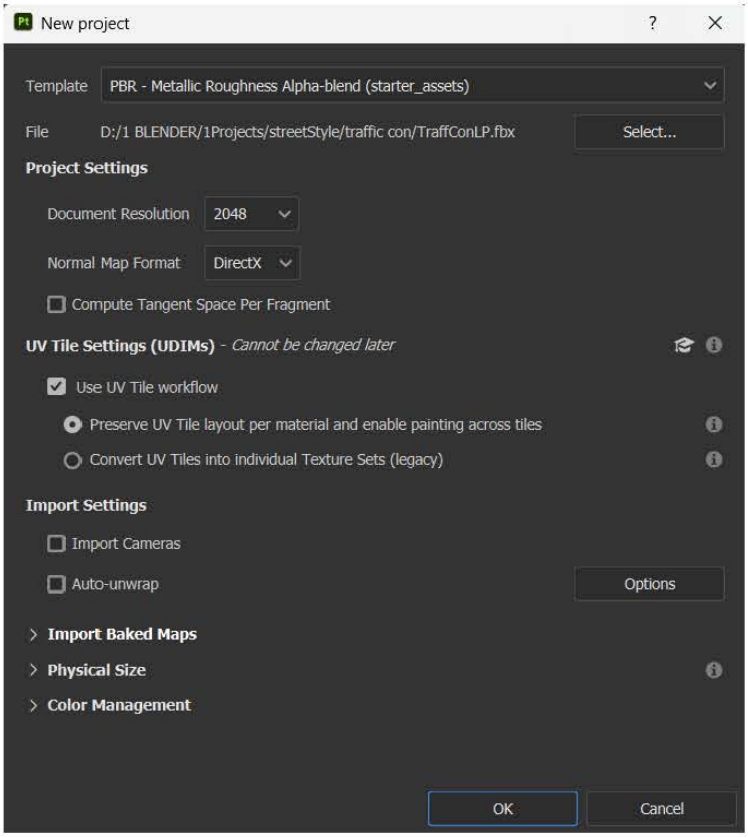

Рисунок 3.9 – Імпорт low-poly моделі в Substance 3D Painter

Для перенесення дрібних геометричних деталей з high-poly моделі (Cone_High.fbx) на low-poly виконується процедура запікання карт (Baking Mesh Maps). Цей процес дозволяє генерувати карти нормалей, навколишнього затінення (AO), висоти, кривизни (Curvature), ID та інших атрибутів, які потім використовуються при створенні текстур.

Запікання виконується через меню Texture Set Settings → Bake Mesh Maps. У параметрах вказується шлях до high-poly моделі, а також визначається тип використовуваних карт (рис. 3.10). З-поміж найбільш критичних:

- Normal Map — для збереження рельєфу поверхні;
- Ambient Occlusion — для симуляції глибини та затінення;
- Curvature — для підсилення країв і стиків;
- Position / Thickness — для додаткових ефектів в матеріалах.

Важливо правильно налаштувати відстань сканування (Max Frontal / Rear Distance) для коректного перекриття low- та high-poly поверхонь. Після запуску процесу (кнопка Bake Selected Textures) виконується обчислення, після чого запечені карти зберігаються всередині проєкту та використовуються автоматично.

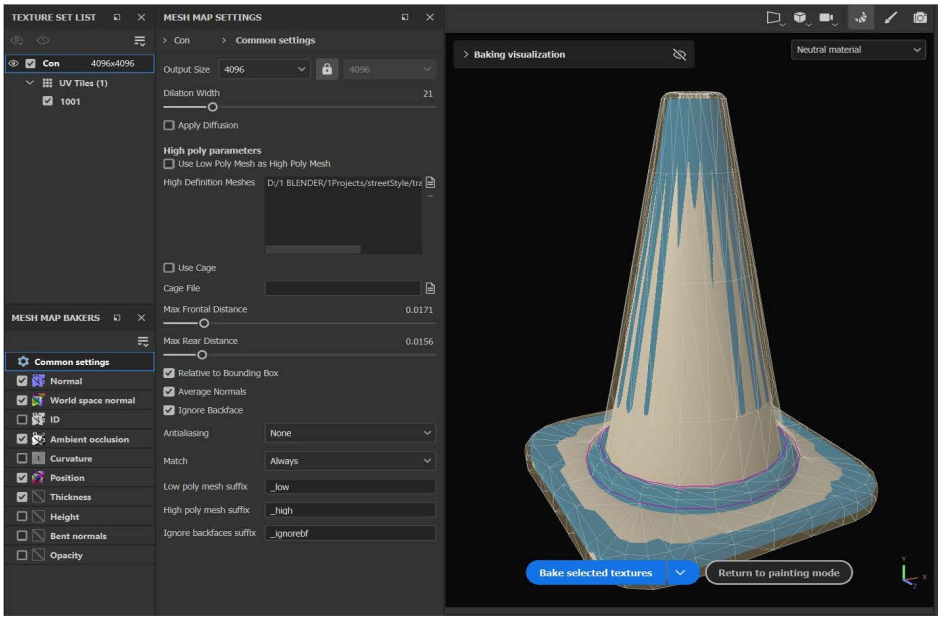

Рисунок 3.10 – Налаштування параметрів запікання та підключення high-poly моделі

Після завершення запікання переходять безпосередньо до створення матеріалу. У випадку дорожнього конуса, текстурування здійснювалося шляхом застосування Smart Materials, що імітують типові матеріали — пластик, гума, фарба тощо. Для основної частини об'єкта був використаний червоно-помаранчевий пластиковий матеріал із середнім рівнем глянцю, на який вручну додано маску з потертостями. Це дозволяє досягти ефекту старіння, який виглядає реалістично у поєднанні з картою нормалей.

Окремим шаром створено білу смугу, яка наноситься за допомогою Paint Layer та альфа-маски. Для гумової основи застосовано темний матовий матеріал із легким ефектом запиленості. Також додано grunge-текстури з карти Curvature, що підсилюють крайові зони.

У підсумку створено повноцінний матеріал, який виглядає реалістично, має глибину та чітко виражені мікродеталі (рис. 3.11). Завдяки використанню PBR-підходу, ці властивості будуть однаково коректно відображатися в будь-якому рушії, який підтримує фізично коректне освітлення.

Рисунок 3.11 – Приклад нанесення матеріалів і текстур у Substance Painter

Після завершення текстурування обирається пункт File → Export Textures, де налаштовується формат експорту. Для Unreal Engine 5 доцільно використовувати пресет Unreal Engine 4 Packed, який включає всі необхідні карти: Base Color, Normal, OcclusionRoughnessMetallic (в одному файлі), Height тощо.

Текстури експортуються у форматі .PNG або .TGA з роздільною здатністю 2K. Після експорту вони будуть готові для підключення до матеріалу у рушії.

3.1.5 Узагальнення процесу створення 3D-моделей для ігор

Описаний процес моделювання об’єкта на прикладі дорожнього конуса відображає типовий робочий підхід, який застосовується при створенні 3D-ресурсів у сучасній ігровій індустрії. Він охоплює повний цикл підготовки ігрових моделей — від формування базової геометрії до експорту готових текстур для інтеграції у рушій.

Основна мета полягає в тому, щоб поєднати реалістичний зовнішній вигляд об'єкта з оптимізованою геометрією, що не перевантажує ігрову сцену. Для цього й використовується комбінація low-poly моделі, яка застосовується безпосередньо у грі, та high-poly версії, з якої запікаються деталі поверхні. Такий підхід дозволяє значно покращити візуальне сприйняття без шкоди для продуктивності.

Робота з UV-розгорткою та запіканням карт у Substance 3D Painter є стандартною практикою в професійному виробництві ігор. Автоматизовані інструменти, такі як Smart Materials і Grunge-текстури, дозволяють швидко додавати сліди зношення, бруду, потертостей, що робить об'єкти більш природними й правдоподібними. У сукупності це забезпечує високий рівень якості навіть для допоміжних елементів сцени.

Завдяки модульності підходу описаний метод може застосовуватись для створення широкого спектра ігрових активів: від дрібних об’єктів типу дорожніх знаків до складних транспортних засобів чи будівель.

3.2 Створення проєкту та моделювання світу гри

Після завершення підготовки 3D-активів у зовнішніх програмних засобах наступним етапом є створення проєкту в ігровому рушії Unreal Engine 5 та початок побудови повноцінного ігрового середовища, у якому буде реалізовано основну логіку симулятора. Цей етап охоплює ініціалізацію структури проєкту, імпорт раніше створених моделей, організацію рівня та створення ігрової сцени з дорогами, інфраструктурою й допоміжними об’єктами.

Процес починається з ініціалізації нового проєкту в UE5, після чого відбувається імпорт підготовлених моделей, таких як дорожній конус, дорожні знаки, бар'єри та інші декоративні або функціональні елементи. Завершальним етапом цього блоку є побудова базової карти: формування рельєфу, розміщення об'єктів, встановлення освітлення та візуальних ефектів.

3.2.1 Створення нового проєкту в Unreal Engine 5

Розробка починається з ініціалізації нового проєкту в Unreal Engine. Після запуску рушія в головному меню обирається шаблон Games → Blank, що дозволяє створити проєкт з нуля, без попередньо визначених механік (рис. 3.12).

У параметрах проєкту встановлюються такі опції:

- Blueprint як основна мова програмування;
- Starter Content: Yes — для базових матеріалів, текстур і моделей;
- Ray Tracing: Disabled — щоб уникнути навантаження на систему;
- Target Platform: Desktop/Console;
- Graphics: Scalable

Проєкт зберігається під відповідною назвою TruckSimulator. Після створення проєкту автоматично відкривається початкова сцена (Level), у якій можна працювати над побудовою світу.

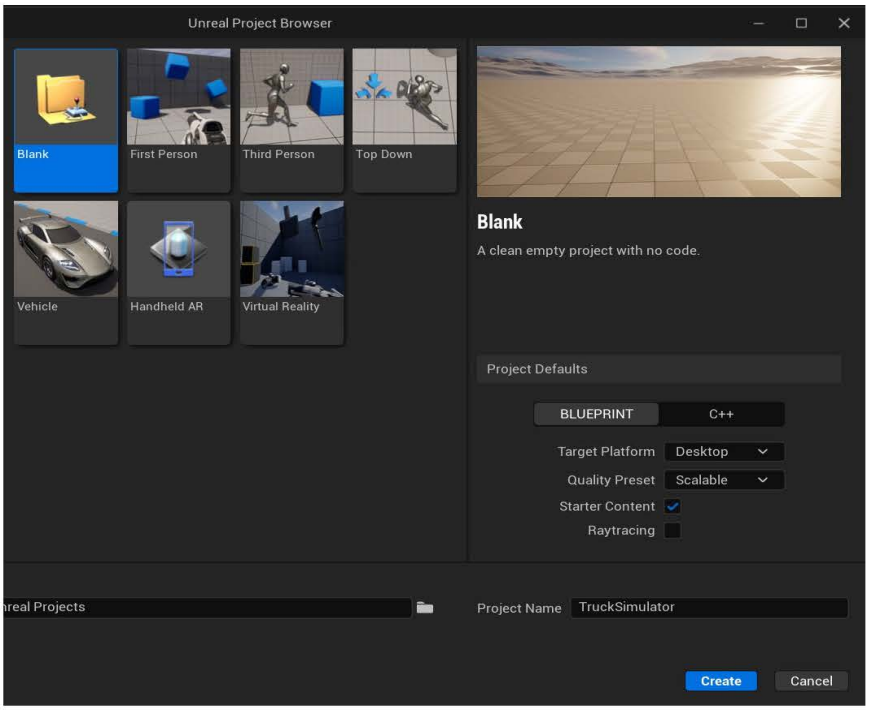

Рисунок 3.12 – Вікно створення нового проєкту в Unreal Engine 5

Далі середовище налаштовується під роботу: очищуються зайві об'єкти з шаблону, додається Player Start, камера, освітлення та готується простір для розміщення ігрових об'єктів.

3.2.2 Імпорт та налаштування моделей у проєкті

Після створення проєкту в Unreal Engine 5 наступним етапом є імпорт зовнішніх моделей, створених у Blender і текстурованих у Substance 3D Painter. На цьому етапі важливо не лише правильно додати файли у рушій, але й виконати необхідні налаштування — застосувати матеріали, задати колізії, структуру рівнів деталізації (LOD) та інтегрувати об'єкти в робочий процес.
Для демонстрації цього етапу було обрано модель дорожнього конуса із минулого підрозділу.

Імпорт відбувається через стандартне вікно: File → Import Into Level. У якості формату використовується .FBX, що був підготовлений на етапі моделювання та текстурування.

Під час імпорту встановлюються такі ключові параметри:

- Import Mesh – увімкнено (імпорт сітки);
- Import Normals and Tangents – увімкнено;
- Normal Import Method – DirectX (відповідає картам із Substance Painter);
- Import Textures / Materials – за потреби, якщо текстури вже запаковані;
- Combine Meshes – вимкнено (щоб кожен об'єкт залишався окремим);
- Transform / Scale – 1.0 (масштаб об'єкта має бути відповідним до гри).

Після імпорту у Content Browser створюється новий Material, до якого додаються карти: Base Color, Normal Map, Roughness, Metallic, AO (якщо використовується Packed). Вони підключаються до відповідних вузлів у матеріальному графі. Після збереження матеріал призначається імпортованій сітці.

Для взаємодії об'єкта з фізичним середовищем необхідно створити або призначити Collision. У редакторі Static Mesh (Asset Editor) відкривається вкладка Collision → Add Box Simplified Collision або Add Convex Collision. У випадку з дорожнім конусом застосовується спрощена капсульна або циліндрична колізія, що забезпечує точність і водночас економить ресурси.

Щоб покращити продуктивність у великих сценах, кожна модель повинна мати рівні деталізації (Level of Detail). У редакторі Static Mesh у вкладці LOD Settings обирається Auto Generate LODs, де рушій автоматично створює 2–3 варіанти об'єкта з меншою кількістю полігонів. Це дозволяє зменшувати навантаження при віддаленні об'єкта від камери.

Після всіх налаштувань модель готова до розміщення на ігровій сцені (рис. 3.13). У вікні Viewport об'єкт перетягується з Content Browser, позиціонується відносно рельєфу, додається до колекції (Folder) відповідно до призначення (декоративний, технічний, блокуючий об'єкт тощо). Об'єкт автоматично успадковує властивості освітлення, тіней, взаємодії з гравцем та може бути використаний у подальших механіках гри (наприклад, перевірка зіткнень, взаємодія зі світлом тощо).

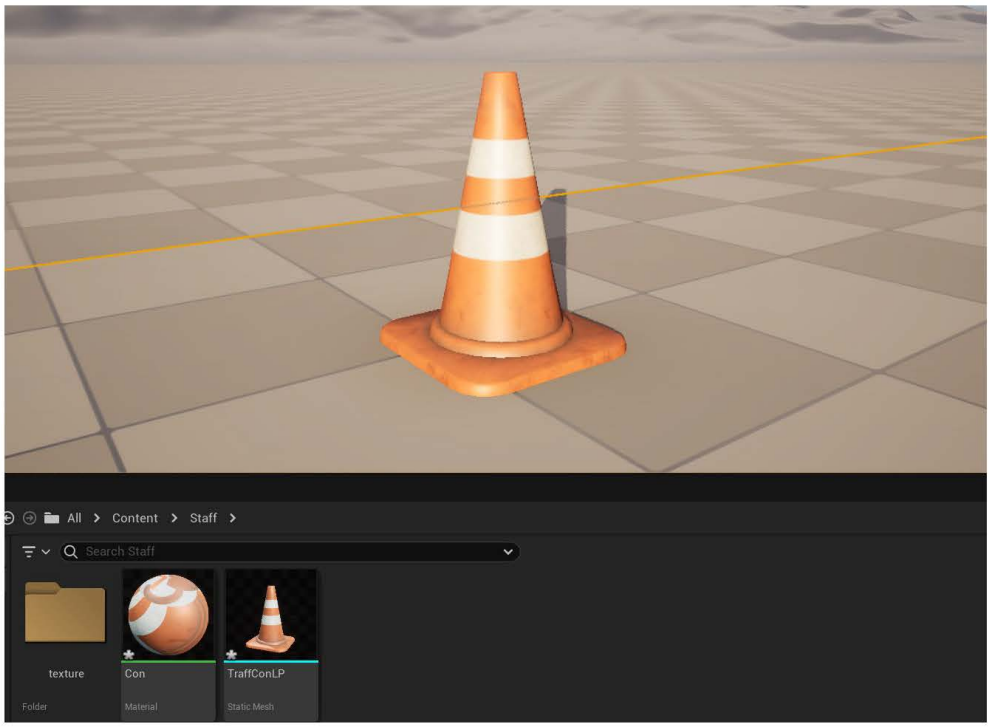

Рисунок 3.13 – Повністю налаштований об'єкт в Unreal Engine 5

3.2.3 Побудова ігрової сцени та налаштування освітлення

Після налаштування моделей у проєкті наступним кроком є побудова ігрової сцени — просторового середовища, в якому відбувається взаємодія гравця з ігровим світом. Створення карти передбачає компонування рельєфу, розміщення об’єктів, налаштування освітлення, тіней та атмосфери. Від правильності реалізації цього етапу залежить не лише візуальна складова гри, а й її технічна оптимізація.

Для формування основи ігрового світу в Unreal Engine 5 використовується інструмент Landscape, що дозволяє створити деталізовану поверхню — рівнини, пагорби, схили. У вікні Modes обирається вкладка Landscape, після чого задаються розміри сітки та резолюція. Створення здійснюється натисканням кнопки Create.

Після генерації ландшафту на нього накладається багатошаровий матеріал, що поєднує текстури трави, ґрунту, асфальту тощо. Розміщення шарів здійснюється вручну за допомогою пензлів у режимі Sculpt та Paint.

Імпортовані моделі (наприклад, дорожні конуси, бар'єри, знаки) (рис. 3.14) перетягуються з Content Browser безпосередньо у вікно Viewport. Об'єкти групуються у відповідні папки (Folders) в Outliner, залежно від їхнього призначення. Для масштабності сцени об'єкти копіюються, обертаються, масштабуються та адаптуються до рельєфу вручну або із застосуванням Align to Landscape.

Рисунок 3.14 – Процес створення середовища в Unreal Engine 5

Особлива увага приділяється правильному розміщенню декорацій, які впливають на геймплей — перешкод, орієнтирів, інфраструктурних елементів. Вони також можуть бути інтегровані у Blueprint Actors, якщо потребують інтерактивної поведінки.

Для забезпечення візуальної привабливості та реалістичності в сцену додаються базові елементи освітлення:

1) Directional Light – основне джерело світла, що імітує сонце;

2) Sky Atmosphere – симулює атмосферу та кольорові зміни неба;

3) Exponential Height Fog – створює ефект глибини та туману на віддалених об'єктах;

4) Sky Light – забезпечує м'яке розсіяне освітлення;

5) Post Process Volume – контролює ефекти освітлення та інші.

Після виконання всіх необхідних етапів — створення рельєфу, імпорту та налаштування моделей, розміщення об’єктів, додавання освітлення та атмосфери — було сформовано базову ігрову сцену (рис. 3.15), що відображає характерні риси середовища для симулятора далекобійника. Створений простір включає дорожню інфраструктуру, технічні зони та декорації, які забезпечують як візуальну повноту, так і функціональну основу для подальшого розвитку ігрового процесу.

Рисунок 3.15 – Готова сцена з рельєфом, освітленням та розміщеними моделями

3.3 Реалізація керованого транспортного засобу

Одним із ключових елементів симулятора далекобійника є реалізація повноцінної керованої вантажівки. На цьому етапі відбувається імпорт 3D-моделі вантажного автомобіля та його компонентів, створення базового батьківського класу транспортного засобу, налаштування підключення коліс, анімації, а також створення повноцінного Blueprint-актора, з яким взаємодіятиме гравець.

3.3.1 Імпорт вантажівки та компонентів

Модель вантажного автомобіля імпортується у проєкт у форматі .FBX. При цьому шасі (основа кузова) та колеса імпортуються окремими об'єктами. Такий підхід дозволяє гнучко керувати трансформаціями, підвіскою та анімацією обертання коліс у процесі гри. Також при імпорті важливо переконатись, що центр координат моделей (pivot point) розташований коректно — для шасі це зазвичай нижній центр, для коліс — геометричний центр обертання.

Моделі після імпорту конвертуються у Static Mesh (колеса) та Skeletal Mesh (рис. 3.16) і зберігаються в окремій папці /Vehicles/Truck/.

Рисунок 3.16 – Skeletal Mesh вантажівки

3.3.2 Створення базового класу транспортного засобу

Для забезпечення можливості повторного використання логіки транспортних засобів було створено батьківський клас Blueprint з назвою BP_VehicleMaster (рис. 3.17). Цей клас заснований на ChaosWheeledVehicle що дозволяє реалізувати реалістичну поведінку транспорту.

У BP_VehicleMaster налаштовуються:

- базові змінні (макс. швидкість, сила гальмування, маса);
- структура компонентів (Mesh, SpringArms, Camera, Audio, Input Mapping);
- загальні функції керування (рух, поворот, гальмо, задній хід);
- події старту, зупинки, оновлення швидкості тощо.

Цей клас не містить конкретної геометрії чи моделей, а лише логіку і базову структуру, яка наслідується всіма транспортними засобами у грі.

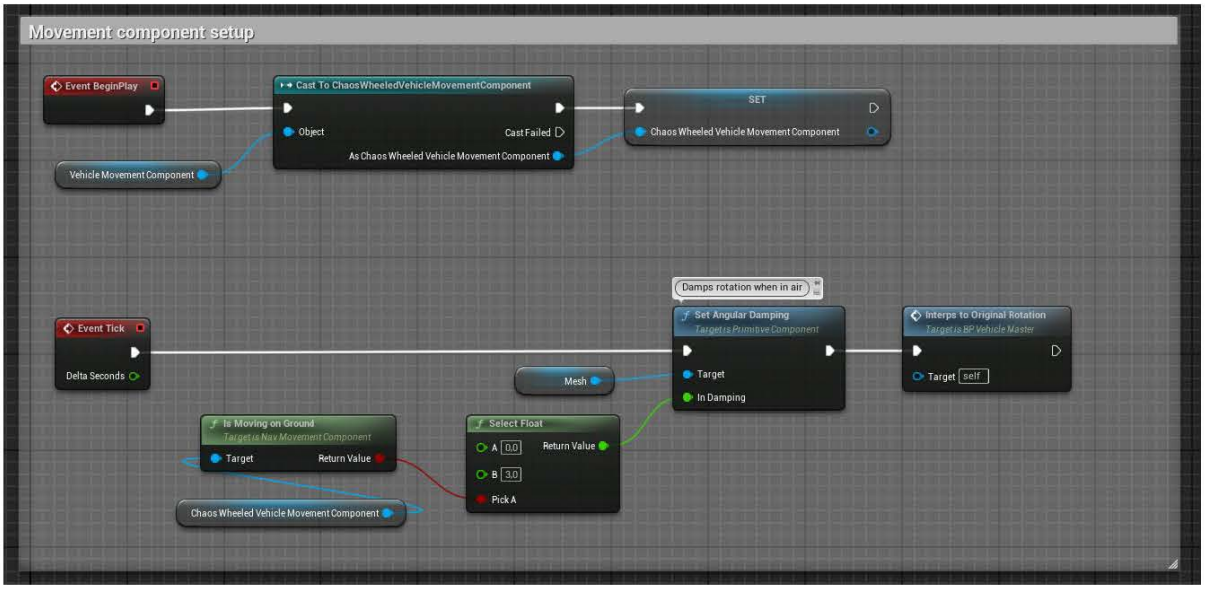

Рисунок 3.17 – Частина налаштуваннь в BP_VehicleMaster

3.3.3 Створення блюпрінта вантажівки

На основі BP_VehicleMaster створюється конкретний блюпрінт вантажівки BP_Truck (рис. 3.18). У нього додається модель кузова, яка прикріплюється до головного Mesh-компонента. Окремо налаштовуються чотири колеса, для кожного з яких визначається своє положення, функція повороту (для передніх) або привід (для задніх). У властивостях кожного колеса задаються радіус, жорсткість підвіски, сила зчеплення, кут повороту та інші фізичні параметри.

Кожне колесо підключається окремо з вказанням типу (FrontLeft, FrontRight, RearLeft, RearRight тощо) та відповідним розташуванням на осі.

Налаштовуються такі параметри:

1) Wheel Radius, Suspension Max Raise/Drop, Friction, Steering Angle;

2) Use Auto Width — вимкнено для точного контролю;

3) Handbrake Axis — для задніх коліс.

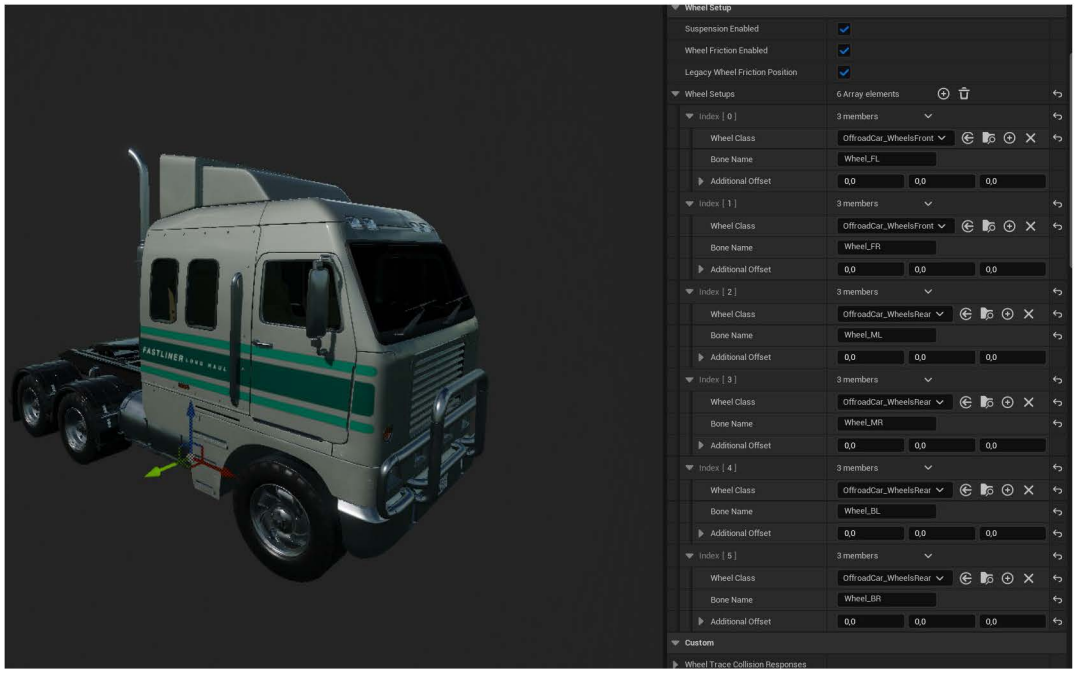

Рисунок 3.18 – Налаштування коліс в BP_Truck

3.3.4 Анімація обертання коліс

Щоб забезпечити візуальну достовірність руху транспортного засобу, необхідно налаштувати систему анімації обертання коліс. Це реалізується за допомогою Animation Blueprint (рис. 3.19), прив’язаного до Skeletal Mesh моделі вантажівки. У ньому конфігурується анімаційний граф (AnimGraph), який визначає, як саме мають обертатися колеса у відповідь на фізичну симуляцію в грі.

У якості базової поза-моделі використовується вузол Mesh Space Ref Pose, який подає опорну позу скелета. Далі додається вузол Wheel Controller, що відповідає за передачу обертання та кута повороту на колеса на основі даних від компонента ChaosWheeledVehicleMovementComponent.

Щоб дані коректно перетворювались у локальні координати моделі, після Wheel Controller використовується вузол Component To Local, а результат передається до вихідного вузла Output Pose, який виводить фінальну позу для рендерингу.

Цей ланцюг дає змогу колісним об'єктам автоматично анімуватись під час руху транспортного засобу, обертатись при прискоренні, сповільнюватись при гальмуванні, а також змінювати кут при повороті.

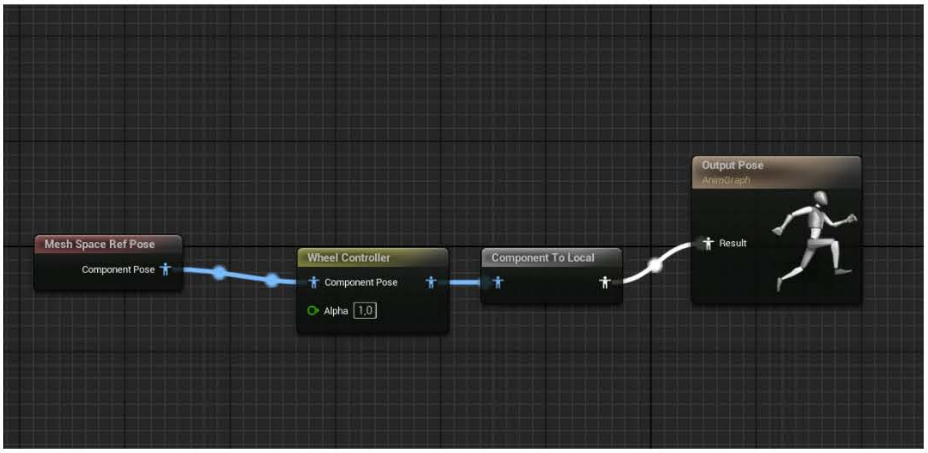

Рисунок 3.19 – Налаштування анімаційного графа для обертання коліс у Anim Blueprint

3.3.5 Призначення гравця та встановлення ігрового режиму

Після завершення етапу створення транспортного засобу, підключення фізики, коліс та анімаційної логіки необхідно зробити так, щоб гравець за замовчуванням керував саме реалізованою вантажівкою. Для цього створюється ігровий режим (Game Mode), в якому визначається стандартний клас гравця, тобто дефолтний Pawn (рис. 3.20).

У папці проєкту /Blueprints/GameModes/ створюється новий Blueprint клас, що наслідує від GameModeBase. Йому задається ім’я TruckSimulatorGame. Після створення, у параметрах GameMode встановлюється клас гравця (Default Pawn Class) — BP_Truck, тобто та сама вантажівка, яка була створена раніше у якості керованого транспортного засобу.

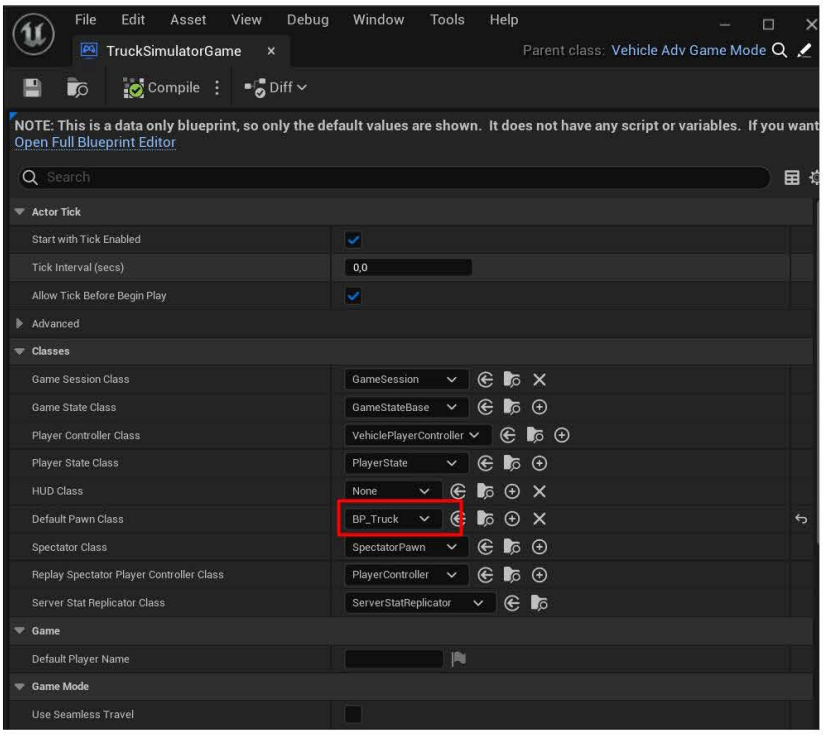

Рисунок 3.20 – Встановлення BP_Truck як дефолтного Pawn у Blueprint TruckSimulatorGame

Наступним кроком є активація цього режиму гри для поточної сцени. У вкладці World Settings у полі GameMode Override вибирається TruckSimulatorGame (рис. 3.21). Завдяки цьому, при запуску гри на будь-якій мапі, де активний цей Game Mode, гравець одразу буде спавнитися у вантажівці та матиме змогу почати керування без потреби вручну виставляти об’єкт у сцену.

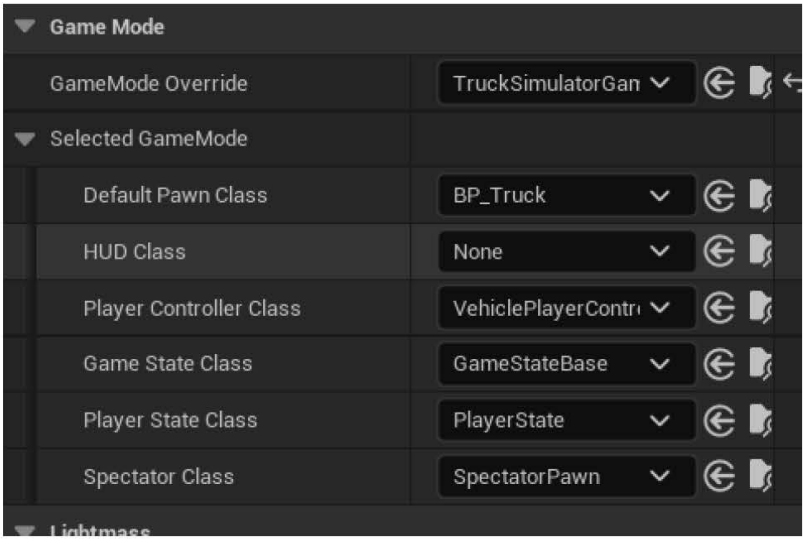

Рисунок 3.21 – Призначення ігрового режиму TruckSimulatorGame

Цей крок дозволяє повністю зв'язати ігрову логіку, Blueprint транспортного засобу та сцену. Таким чином, взаємодія з гравцем починається одразу після запуску гри, а реалізований функціонал BP_Truck стає основою подальшого розвитку геймплею.

3.4 Структура та призначення меню в ігровому застосунку

Меню є важливою частиною взаємодії користувача з будь-яким ігровим застосунком. Воно виконує функцію зв'язувального інтерфейсу між гравцем та ігровими механіками, надаючи можливість керування запуском, налаштуванням, збереженням, виходом та іншими аспектами гри. Окрім функціонального призначення, меню формує перше враження від проєкту та задає загальний стиль інтерфейсу.

В іграх жанру симуляторів, де важливе занурення у процес та зручність керування, якісно побудоване меню є критично важливим. Через нього користувач отримує доступ до початку гри, конфігурації параметрів (наприклад, гучність, управління, графіка), довідки, кредитів та можливості завершити сеанс. Кожне з меню реалізується за допомогою інструментів UMG (Unreal Motion Graphics), що дозволяють створювати адаптивні та інтерактивні інтерфейси без написання коду, використовуючи візуальний редактор.

Одним із базових елементів взаємодії користувача з грою є головне меню, яке відображається при запуску застосунку. Воно виконує функції навігації, налаштування параметрів гри, перегляду інформації про авторів та виходу з гри. Головне меню реалізовано за допомогою системи UMG (Unreal Motion Graphics) — вбудованого інструменту для створення інтерфейсів в Unreal Engine 5.

Процес розробки меню розпочинаєть зі створення Widget Blueprint з назвою WBP_MainMenu. В його структурі використовується базовий Canvas Panel для фіксованого розміщення елементів інтерфейсу. На панель накладається фон — зображення, яке створює атмосферу гри, а також додано логотип і назва — "VP Truck Simulator".

Для розміщення кнопок навігації створено окремий Vertical Box, у якому компонуються чотири кнопки: Play, Settings, Credits та Quit. Кожна з них реалізована як окремий елемент типу Button, із вкладеним текстовим компонентом. Для візуального оформлення використано стилізовану рамку Border, а також фонову панель з кольоровим контуром (Border_Colour та Box_Outline).

У подальшому, для кожної кнопки призначаються відповідні функції у Blueprint-логіці:

- Play – завантаження ігрового рівня (наприклад, Open Level (MainMap)),
- Settings – відкриття віджету з параметрами (гучність, графіка),
- Credits – виклик екрану з інформацією про розробників,
- Quit – завершення гри (Quit Game).

Також у Graph Blueprint було реалізовано логіку прив'язки головного меню до гравця. Для цього у проєкті створювався окремий рівень, який слугує стартовим меню гри. У його Level Blueprint налаштовано виклик команди Create Widget (WBP_MainMenu) та Add to Viewport, що дозволяє автоматично додавати меню до екрану при запуску (рис. 3.22). Додатково активується Set Show Mouse Cursor, що дає змогу керувати інтерфейсом мишею, та блокується ввід з клавіатури до моменту переходу до геймплейного рівня.

Рисунок 3.22 – Інтерфейс головного меню гри

Функціональність кнопок реалізовано у графічному редакторі Blueprints. Для кожної кнопки передбачено подію On Button Clicked, яка ініціює відповідну логіку. Наприклад, при натисканні кнопки Play (рис. 3.23) створюється віджет завантажувального екрана (Create LOADING SCREEN Widget), який додається до в'юпорту. Після короткої затримки (Delay), виконується команда Open Level, що переносить гравця до обраного рівня . Паралельно виконується налаштування режиму вводу (Set Input Mode Game Only) та приховується курсор миші, що готує інтерфейс до повноцінного геймплею.

Кнопка Credits спочатку прибирає поточне меню з екрана (Remove from Parent), після чого створює та виводить новий віджет з інформацією про автора гри (Create CREDITS Widget). Аналогічний підхід реалізовано для кнопки Settings, яка відкриває окремий інтерфейс налаштувань гри, створюючи новий відповідний віджет (SETTINGS Widget) і додаючи його на екран. Нарешті, кнопка Quit безпосередньо викликає функцію Quit Game, яка завершує виконання програми. Для цього використовується вбудований вузол Unreal Engine, що забезпечує коректне завершення роботи застосунку незалежно від платформи.

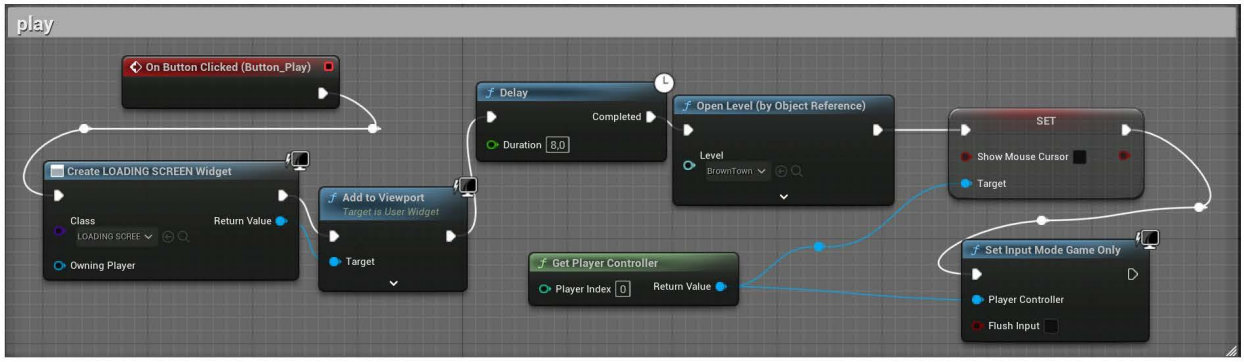

Рисунок 3.23 – Реалізація кнопки Play

За аналогічним принципом було реалізовано й інші елементи інтерфейсу — меню налаштувань, екран кредитів та вікна повідомлень. Кожне з них створювалося як окремий Widget Blueprint із відповідною структурою та логікою взаємодії, що викликаються з головного або паузного меню. Усі віджети мають

уніфікований стиль оформлення, що відповідає загальній візуальній тематиці гри. Для покращення користувацького досвіду було реалізовано анімації появи та зникнення елементів. Вся логіка роботи інтерфейсу побудована на подієво-орієнтованому підході, що забезпечує гнучкість і масштабованість. Завдяки цьому в подальшому легко додавати нові елементи або змінювати наявні без необхідності повної перебудови системи.

3.5 Реалізація трафіку та системи маршруту

Одним із ключових компонентів симулятора є імітація дорожнього руху, а також визначення стартової та кінцевої точок маршруту гравця. Така система дозволяє створити ігровий процес із чіткою логікою: старт, рух по маршруту, фініш та завершення ігрової сесії. Реалізація цього функціоналу включає декілька складових: створення автономних автомобілів (AI), налаштування їхніх маршрутів, розміщення точки появи гравця (Player Start) та інтерактивної фінішної зони, що завершує гру.

Використання сплайнів для руху NPC дозволяє точно контролювати траєкторії автомобілів, створюючи реалістичну імітацію трафіку без складної навігації. Розміщення гравця за допомогою компонента Player Start забезпечує правильну ініціалізацію транспортного засобу після запуску гри.

3.5.1 Створення трафіку та маршруту руху

Для симуляції дорожнього руху у проєкті було реалізовано систему автономних транспортних засобів (NPC), а також ігровий маршрут, яким має слідувати гравець. Обидва ці елементи базуються на використанні Spline-компонентів, що забезпечують гнучке, контрольоване та передбачуване переміщення об’єктів у межах заданої траєкторії. Завдяки цьому вдається створити реалістичний ефект дорожнього трафіку, який не перевантажує рушій та забезпечує плавність руху.

Першим кроком було створення окремого Blueprint-класу транспортного засобу NPC з назвою AI_CAR (рис. 3.24). Цей клас побудований на базі ChaosWheeledVehicle, що надає можливість моделювати складну фізику автомобіля — з урахуванням ваги, гальмівного шляху, зчеплення з дорогою, крутного моменту та інших параметрів. Для керування рухом автомобіля використовується спеціальний AI Controller, який відповідає за автономну поведінку, включаючи прийняття рішень, виконання маневрів і взаємодію з оточенням.

Клас AI_CAR містить Spline-компонент, який визначає маршрут руху NPC. У Graph Blueprint реалізовано алгоритм, за яким транспорт автоматично слідує сплайном із заданою швидкістю, імітуючи трафік.

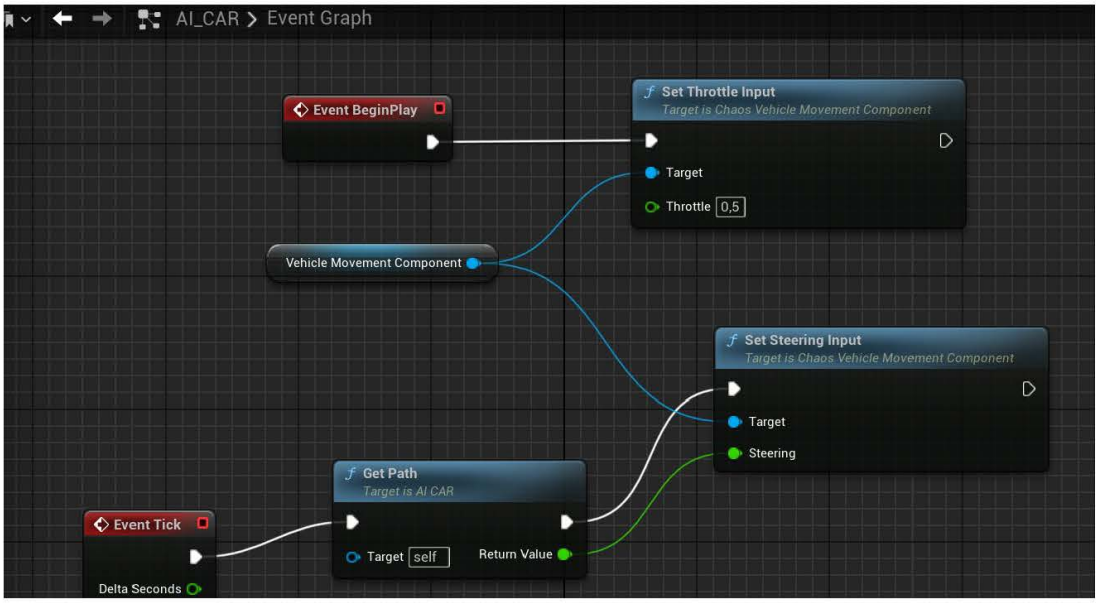

Рисунок 3.24 – Частина логіки AI_CAR

До структури AI_CAR додається компонент Spline (рис. 3.25), який задає маршрут руху автомобіля. У Graph Blueprint реалізовано цикл, за яким NPC-автомобіль послідовно переміщується вздовж координат сплайну із заданою швидкістю. Для більшої реалістичності також можуть додаватися умови для зупинок, зміни напрямку або коригування швидкості залежно від обставин (наприклад, ситуацій на перехрестях).

Рисунок 3.25 – AI_CAR з логікою руху по сплайну

3.5.2 Початкова та кінцева точки маршруту

Для реалізації цілісного ігрового процесу в симуляторі вантажних перевезень необхідно визначити чіткий маршрут гравця — початок, шлях руху та точку завершення. Це дозволяє сформувати логічну структуру сесії гри: від моменту появи гравця на мапі до фінального екрану з результатом.

Поява гравця у світі відбувається за допомогою стандартного компонента Player Start, який розміщується на початку маршруту. Саме в цій точці і відбувається спавн гравця при запуску гри. У GameMode з назвою TruckSimulatorGame зазначено, що за замовчуванням використовується клас BP_Truck, що автоматично зв'язує точку старту з транспортним засобом гравця. Завдяки цьому, при запуску гри гравець одразу потрапляє у кабіну вантажівки на початку маршруту, без потреби ручного розміщення об’єкта (рис. 3.26).

Рисунок 3.26 – Початкова точка гравця (Player Start)

Кінцева точка маршруту розташовується у визначеному місці сцени, що символізує завершення перевезення. У цій зоні встановлюється Trigger Box, який реагує на вхід транспортного засобу гравця. При активації тригера спрацьовує Blueprint-логіка, що викликає окремий Widget Blueprint — WBP_FinishScreen (рис. 3.27). На цьому інтерфейсі виводиться повідомлення з подякою за участь у грі, а також пропонується вибір: рестарт гри (Open Level) або вихід (Quit Game).

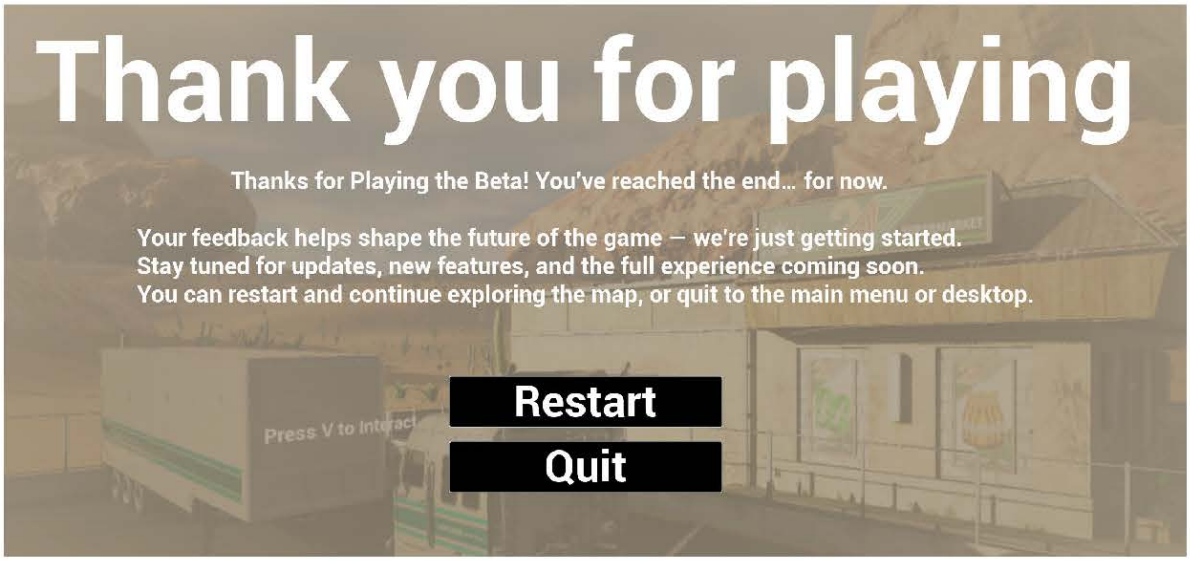

Рисунок 3.27 – Інтерфейс WBP_FinishScreen

3.6 Фази та ітерації життєвого циклу програмного забезпечення

Проєктування і розробка програмного забезпечення, особливо такого складного продукту як симулятор, передбачає проходження чітко визначених етапів. У межах даного проєкту використано інкрементно-ітераційну модель життєвого циклу, яка поєднує послідовність фаз класичних моделей з перевагами циклічного вдосконалення, властивого підходам гнучкої розробки.

Першою фазою розробки став аналіз вимог. На цьому етапі було зібрано та систематизовано інформацію про специфіку жанру симуляторів, типові очікування гравців, приклади успішних аналогів. У результаті сформовано технічне завдання, що включало функціональні, нефункціональні, апаратні та архітектурні вимоги до системи.

Реалізація ігрового застосунку здійснювалася поетапно — через серію ітерацій, кожна з яких завершувалась формуванням працездатного інкременту. У першій ітерації реалізовано базове керування вантажівкою та налаштування фізики руху. У наступних ітераціях поступово додавались елементи навігації, карта, дорожній трафік, головне меню та HUD. Особливу увагу було приділено реалізації сценаріїв доставки вантажу — від отримання замовлення до отримання винагороди.

Кожен інкремент підлягав тестуванню на предмет працездатності функцій, взаємодії між модулями та коректного відображення ігрової інформації. Виявлені помилки виправлялись до переходу до наступної ітерації, що дозволяло підтримувати стабільність проєкту на всіх етапах.

На завершальному етапі виконано оптимізацію продуктивності, зокрема впроваджено механізми LOD (зменшення деталізації віддалених об'єктів), оптимізовано використання ресурсів і зменшено розмір ігрового білду. Було також підготовлено інсталяційний пакет для демонстрації.

Загальна структура життєвого циклу, реалізована в проєкті, подана у таблиці 3.1 У ній зазначено основні фази, їхній зміст та результати, яких очікувалося досягти на кожному етапі.

Таблиця 3.1

Життєвий цикл розробки проєкту

№	Фаза / Ітерація	Зміст робіт	Очікуваний результат
1	Аналіз вимог	Визначення цільової аудиторії, функціоналу, обмежень, системних вимог	Сформульоване технічне завдання і структура проєкту
2	Проєктування	Побудова архітектури, моделювання логіки (IDEF0, UML), проектування UI	Схема системи, сценарії використання, структури класів
3	Ітерація 1: Базовий геймплей	Реалізація фізики керування вантажівкою, базового управління камерою	Прототип керованого транспорту
4	Ітерація 2: Навігація й маршрути	Додавання карти, навігаційних точок, обмежень руху	Робоча система орієнтації гравця у світі
5	Ітерація 3: Трафік і NPC	Реалізація дорожнього трафіку та взаємодії з іншими авто	Симуляція живого середовища з NPC
6	Ітерація 4: Інтерфейс та меню	Головне меню, меню паузи, HUD, підказки, повідомлення	Повноцінний UI з підтримкою взаємодії
7	Тестування	Перевірка взаємодії між компонентами, усунення помилок, юзабіліті-тест	Стабільна версія з мінімальною кількістю помилок
8	Оптимізація	LOD, стрімінг рівнів, зменшення розміру проєкту, профілювання	Підвищення FPS, стабільна робота на ПК середнього рівня
9	Завершення і супровід	Підготовка до демонстрації, документація, оновлення, можливе масштабування	Фінальний білд і потенціал для подальшого розвитку

3.7 Висновок до розділу 3

У цьому розділі було детально розглянуто процес реалізації ключових компонентів ігрового застосунку в жанрі симулятор далекобійника на основі рушія Unreal Engine 5. Послідовно були описані всі етапи створення проєкту: від моделювання та текстурування об'єктів у сторонніх редакторах до безпосередньої інтеграції моделей у рушій та побудови повноцінного ігрового світу.

Розроблено ігрову сцену з основним середовищем, підготовлено вантажівку як головний транспортний засіб гравця, реалізовано систему керування з використанням Chaos Vehicle, а також створено анімаційний граф для обертання коліс. Було налаштовано базові інтерфейси взаємодії: головне меню, меню паузи та фінальний екран, що забезпечують логіку запуску та завершення гри. Окрему увагу приділено реалізації елементів дорожнього трафіку — створено NPC-автомобілі з автономною поведінкою, налаштовано маршрути руху та систему спавну.

Завдяки використанню інструментів Unreal Engine та зовнішніх програмних засобів вдалося досягти функціональної цілісності гри. Кожен компонент — візуальний, ігровий чи технічний — був інтегрований у загальну структуру проєкту з дотриманням принципів гнучкості, повторного використання і оптимізації. Отримані результати є основою для подальшого розвитку гри: додавання нових місій, розширення карти, ускладнення логіки AI, системи збереження прогресу тощо.

ВИСНОВОК

Кваліфікаційна робота була присвячена розробці 3D-ігрового застосунку в жанрі симулятор далекобійника, що є актуальним напрямом у контексті розвитку сучасної ігрової індустрії. Метою роботи було створення базової версії симулятора з реалістичною фізикою керування транспортом, інтерактивним середовищем та візуально якісним ігровим світом. Такий підхід спрямований на покращення користувацького досвіду та демонстрацію можливостей використання рушія Unreal Engine 5 у створенні симуляційного контенту.

У процесі роботи було проведено огляд предметної області, що включав аналіз сучасних підходів до розробки симуляторів, визначення особливостей жанру, порівняння ігрових рушіїв та інструментів моделювання. Особливу увагу приділено характеристикам таких рушіїв, як Unity, Unreal Engine, Godot та CryEngine, в результаті чого було обґрунтовано вибір Unreal Engine 5 як оптимальної платформи для реалізації проєкту завдяки його розширеному функціоналу, реалістичній фізиці та підтримці візуального програмування через Blueprints.

У ході реалізації проєкту використано сучасні засоби створення ігрового контенту, серед яких Blender для 3D-моделювання, Substance 3D Painter для текстурування та Adobe Photoshop для підготовки інтерфейсних і допоміжних елементів. Побудовано ігрове середовище, створено керований транспортний засіб на основі Chaos Vehicle, розроблено логіку обертання коліс, імплементовано інтерфейс користувача, що включає головне меню, меню паузи та екран завершення гри. Також реалізовано систему трафіку на основі AI-контролерів та маршрут гравця з інтерактивною стартовою і кінцевою точками.

Важливим аспектом роботи стала інтеграція всіх компонентів у єдину ігрову сцену з дотриманням принципів модульності, гнучкості та оптимізації. Було реалізовано сценарій гри, який включає запуск, навігацію по ігровій карті, взаємодію з трафіком та завершення маршруту. Система інтерфейсів розроблена з урахуванням сучасних UX-підходів, що забезпечує зручність для гравця.

Результати тестування прототипу показали, що всі основні функції працюють стабільно, інтерфейс коректно реагує на дії користувача, а фізика керування вантажівкою створює реалістичне відчуття симуляції. Таким чином, поставлені в роботі завдання були досягнуті в повному обсязі, а розроблений проєкт демонструє ефективність застосування сучасних інструментів і технологій для створення якісного ігрового продукту.

Отримані результати можуть бути використані для подальшого розвитку гри: додавання відкритого світу, системи місій, мультиплеєра, логістичних завдань або поглиблення симуляційної складової. Крім того, проєкт може бути корисним як приклад у навчальному процесі або як основа для комерційного застосунку у сфері ігрової розробки.

Також важливою частиною стало опрацювання логіки завершення гри. Завдяки використанню сплайнів, тригерів і екрану з подякою було реалізовано базовий сценарій проходження маршруту. Це надає грі завершеності та логічної структури, що особливо важливо для користувацького досвіду.

Проєкт має потенціал до розширення: можна додати механіку витрат пального, систему замовлень, економіку, погодні умови, цикл дня і ночі, багатокористувацький режим або навігацію з картами. Реалізація таких компонентів перетворить гру з базової симуляції у повноцінний продукт, здатний конкурувати на сучасному ринку.

Таким чином, виконана кваліфікаційна робота не лише відповідає вимогам до бакалаврського рівня підготовки, але й демонструє практичну реалізацію принципів розробки ігор у сучасному середовищі. Вона підтверджує здатність автора застосовувати набуті знання в області програмування, 3D-моделювання, дизайну інтерфейсів і роботи з ігровими рушіями для створення реального прикладного програмного продукту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Newzoo. Global Games Market Update Q1 2025 [Електронний ресурс] – Режим доступу: https://newzoo.com/resources/blog/global-games-market-update-q1-2025

2. VGInsights. The Big Game Engine Report of 2025 [Електронний ресурс] – Режим доступу: https://vginsights.com/assets/reports/The_Big_Game_Engines_Report_of_2025.pdf

3. VGInsights. Truck Simulation Games: Popularity and Market Overview 2024–2025 [Електронний ресурс] – Режим доступу: https://vginsights.com

4. Euro Truck Simulator 2. Офіційний сайт [Електронний ресурс] – Режим доступу: https://eurotrucksimulator2.com/

5. Truck & Logistics Simulator. Simula Games [Електронний ресурс] – Режим доступу: https://simulagames.com/

6. On The Road: Truck Simulator. Aerosoft [Електронний ресурс] – Режим доступу: https://www.aerosoft.com/en/shop/move/truck-traffic/1774/on-the-road-truck-simulator

7. Unreal Engine. Офіційний сайт [Електронний ресурс] – Режим доступу: https://www.unrealengine.com

8. Unity. Офіційний сайт [Електронний ресурс] – Режим доступу: https://unity.com

9. Godot Engine. Офіційний сайт [Електронний ресурс] – Режим доступу: https://godotengine.org

10. CryEngine. Офіційний сайт [Електронний ресурс] – Режим доступу: https://www.cryengine.com

11. Blender. Офіційний сайт [Електронний ресурс] – Режим доступу: https://www.blender.org

12. Adobe Photoshop [Електронний ресурс] – Режим доступу: https://www.adobe.com/products/photoshop.html

13. Adobe Substance 3D Painter [Електронний ресурс] – Режим доступу: https://substance3d.adobe.com

14. Unreal Engine 5.5 Documentation [Електронний ресурс] – Режим доступу: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation

15. Wright T. Blender 3D: Noob to Pro. A comprehensive guide to 3D modeling, animation, and rendering, 2022. 510 p.

16. Wilkins J. The Complete Guide to Blender Graphics: Computer Modeling & Animation, CRC Press, 2021. 512 p.