Прикарпатський національний університет імені Василя Стефаника
Фізико-технічний факультет
Кафедра комп'ютерної інженерії та електроніки

Ригін Юрій Вікторович
Yurii Ryhin

УДК 004:42

Спеціальність 123 «Комп'ютерна інженерія»

Кваліфікаційна робота
на здобуття освітнього ступеня бакалавра

Створення графічних моделей на основі рушію Unreal Engine
Creation of graphic models based on Unreal Engine

Науковий керівник:
асистент Тарас БЕНЬКО
Рецензент:
зав. каф. матер. і нов. тех., д. ф-м.н., професор Володимир КОЦЮБИНСЬКИЙ.

Івано-Франківськ
2024

Формат	Поз.	Позначення	Найменування	К-ть	Прим.
А4			Пояснювальна записка	103	

Змн.	Арк.	№ докум.	Підпис	Дата	123.КІ-41.13		
Розробив		Ригін Ю.В.			Специфікація	Літ.	Арк. 2 / Аркушів 1
Перевірив		Бенько Т.Г.					
Н. Контр.							
Затвердив							

АНОТАЦІЯ

Дана кваліфікаційна робота присвячена вивченню процесу створення графічних моделей з використанням рушія Unreal Engine. Робота розглядає основні етапи моделювання в середовищі Unreal Engine для створення реалістичних об'єктів.

У роботі детально описані інструменти й можливості Unreal Engine для роботи з геометрією, текстурами та налаштуванням візуалізації. Розглянуто створення моделей в режимі моделювання та в окремому продукті Metahuman, заточеному на створення людей для подальшого використання в Unreal Engine .

Практична частина роботи демонструє створення кількох повноцінних моделей різної складності, з детальним поясненням кожного етапу. Проаналізовано переваги та недоліки використання Unreal Engine для графічного моделювання в порівнянні з іншими інструментами.

Результати проведеного дослідження можуть бути корисними для спеціалістів у галузі 3D графіки та візуалізації, розробників відеоігор та візуальних ефектів, а також для студентів відповідних спеціальностей.

Ключові слова: Unreal Engine, графічне моделювання, 3D моделі, текстурування, візуалізація.

ABSTRACT

This qualification work is devoted to the study of the process of creating graphic models using the Unreal Engine engine. The work considers the main stages of modeling in the Unreal Engine environment for creating realistic objects.

The work describes in detail the tools and capabilities of Unreal Engine for working with geometry, textures and rendering settings. Considered the creation of models in simulation mode and in the separate product Metahuman, which is focused on creating people for further use in Unreal Engine.

The practical part of the work demonstrates the creation of several full-fledged models of varying complexity, with a detailed explanation of each stage. The advantages and disadvantages of using Unreal Engine for graphic modeling compared to other tools are analyzed.

The results of the research can be useful for specialists in the field of 3D graphics and visualization, developers of video games and visual effects, as well as for students of relevant specialties.

Keywords: Unreal Engine, graphic modeling, 3D models, texturing, visualization.

ПЕРЕЛІК ОСНОВНИХ СКОРОЧЕНЬ

BSP (Binary Space Partition) - метод рекурсивного поділу евклідового простору площинами;

DMX (Digital Multiplex) – стандарт, що описує метод цифрової передачі даних між контролерами та світловим обладнанням, а також додатковим обладнанням

FBX (FilmBox) – популярний формат 3D-файлів;

LOD (Levels of Detail) - прийом у програмуванні тривимірної графіки, що полягає у створенні кількох варіантів одного об'єкта з різними ступенями деталізації, які перемикаються в залежності від видалення об'єкта від віртуальної камери

MR (mixed reality, змішана реальність) - це інноваційна технологія, яка об’єднує віртуальну і доповнену реальність;

NPC (non-player character) - персонаж в комп'ютерних і настільних рольових іграх, керований програмою або майстром, в останньому випадку іноді може називатися ма́йстровим персонажем ;

UE (Unreal Engine) – ігровий рушій, який розробляється та підтримується компанією Epic Games;

UMG (Unreal Motion Graphics) - візуальний інструмент для створення елементів інтерфейсу користувача, таких як меню, прозорий дисплей або інший інтерфейс, який відображається поверх всього іншого зображення;

VR (virtual reality, VR) — різновид реальності в формі тотожності матеріального й ідеального, що створюється та існує завдяки іншій реальності ;

ПК — персональний комп’ютер;

ШІ (штучний інтелект) – здатність машин виконувати завдання, які зазвичай вимагають людського інтелекту.

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«Створення графічних моделей на основі рушію Unreal Engine»

					123.КІ-41.13			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробив		Ригін Ю.В.			. Пояснювальна записка	*Літ.*	*Арк.*	*Аркушів*
Перевірив		Бенько Т.Г.					7	103
Н. Контр.								
Затвердив								

ЗМІСТ

ВСТУП

У сучасному світі цифрових технологій графічні моделі стали невід'ємною частиною багатьох галузей, включаючи кіноіндустрію, ігрову індустрію, архітектуру, медицину, автомобілебудування та віртуальну реальність. Одним із найпотужніших інструментів для створення високоякісних графічних моделей є Unreal Engine - популярний рушій розробки ігор, який забезпечує широкі можливості для візуалізації та інтерактивності.

Unreal Engine від компанії Epic Games спочатку розроблявся для створення відеоігор, проте з часом його можливості значно розширилися. Сьогодні він використовується не тільки для розробки ігор, але й для створення фотореалістичних візуалізацій, анімацій та симуляцій в різних галузях. Завдяки своїм потужним інструментам та функціональним можливостям, Unreal Engine дозволяє художникам та розробникам створювати складні графічні моделі з високою точністю та деталізацією.

Особливої актуальності набуло освоєння цифрових технологій загалом і графічного моделювання зокрема, в контексті дистанційної роботи, яка стала надзвичайно поширеною через глобальну пандемію COVID-19 та триваючі військові конфлікти, зокрема війну в Україні. Перехід на дистанційну роботу змінив підходи до співпраці та реалізації проектів, зробивши онлайн-інструменти, такі як Unreal Engine, необхідними для ефективного виконання завдань.

Ще одним важливим фактором, чому я вибрав для дослідження саме цей рушій, а не інший, стало те, що упродовж останніх років серед великих ігрових студій помітна тенденція до відмови від рушіїв власного виробництва. І переходять вони на рушій Unreal Engine.

Unreal Engine все більше і більше наближається до того, щоб стати галузевим стандартом. Epic Games хоче створити потужний рушій з дуже багатим функціоналом та високою зручністю завдяки Blueprints і відкритому коду.

Метою цієї роботи є дослідження процесу створення графічних моделей за допомогою рушія Unreal Engine. У роботі буде розглянуто основні етапи цього

процесу, також буде проаналізовано ключові особливості Unreal Engine, які роблять його незамінним інструментом для розробників та художників.

В результаті досліджень та практичного освоєння можливостей даного рушія будуть створені графічні моделі, які в подальшому можна буде використовувати в розробці ігор або інших візуалізаціях.

Ця робота стане корисною для студентів, фахівців та всіх, хто цікавиться графічним дизайном та хоче дізнатися більше про можливості та застосування Unreal Engine у створенні графічних моделей. Сподіваюсь, що представлена інформація допоможе глибше зрозуміти процеси, що стоять за створенням високоякісних візуалізацій, та сприятиме подальшому розвитку навичок у цій галузі, особливо в умовах дистанційної роботи.

РОЗДІЛ 1. РОЗВИТОК РУШІЮ UNREAL ENGINE

Unreal Engine - серія ігрових рушіїв 3D-комп'ютерної графіки, розроблених Epic Games, вперше представлених у 1998 році у шутері від першої особи Unreal. Спочатку розроблений для ПК-шутерів від першої особи, згодом він використовувався в різноманітних жанрах ігор і був прийнятий в інших галузях, особливо в кіно та телеіндустрії. Unreal Engine написаний на C++ і має високий ступінь переносимості, підтримуючи широкий спектр настільних, мобільних, консольних і віртуальних платформ.

Останнє покоління, Unreal Engine 5, було запущено в квітні 2022 року. Його вихідний код доступний на GitHub, а комерційне використання надається на основі моделі роялті, при цьому Epic стягує 5% від доходу понад 1 мільйон доларів США, який не поширюється на ігри, опубліковані в Epic Games Store. Epic включила в рушій функції від придбаних компаній, таких як Quixel, чому сприяли доходи Fortnite.

У 2014 році Unreal Engine був названий Книгою рекордів Гіннеса «найуспішнішим рушієм для відеоігор».

1.1 Основні покоління рушію

1.1.1 Перше покоління (Unreal Engine)

Рушій Unreal Engine першого покоління був розроблений Тімом Суїні, засновником Epic Games. Створивши інструменти редагування для своїх умовно-безкоштовних ігор ZZT (1991) і Jill of the Jungle (1992), Суїні почав писати механізм у 1995 році для створення гри, яка згодом стала відомим шутером від першої особи як Unreal. Після багатьох років розробки, вона дебютувала з випуском гри в 1998 році, хоча VicroPose і Legend Entertainment мали доступ до технології набагато раніше, ліцензувавши її в 1996 році. Відповідно до інтерв'ю Суїні написав 90 відсотків коду двигуна, включаючи графіку, інструменти та мережеву систему.

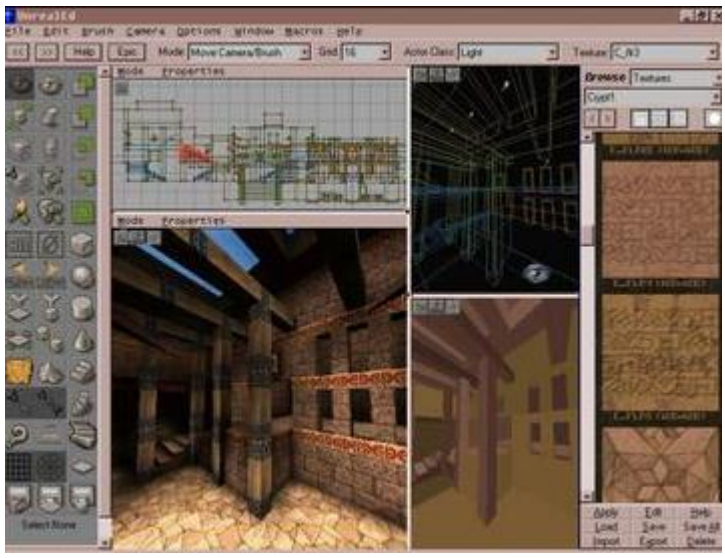

Рис.1.1 – Знімок екрана першої версії UnrealEd, що відображає графічний інтерфейс користувача, написаний на Visual Basic.

Спочатку механізм повністю покладався на програмний рендеринг, тобто графічні обчислення оброблялися центральним процесором. Однак з часом вдалося скористатися можливостями виділених графічних карт, зосередившись на API Glide, спеціально розробленому для прискорювачів 3dfx. Хоча OpenGL і Direct3D підтримувалися, вони повідомили про нижчу продуктивність порівняно з Glide через недолік у керуванні текстурами на той час. Суїні особливо критикував якість драйверів OpenGL для споживчого обладнання, описуючи їх як «надзвичайно проблематичні, з помилками та неперевірені», і назвав код у реалізації «страшним» на відміну від простішої та чистішої підтримки для Direct3D. Що стосується аудіо, Epic використовував Galaxy Sound System, програмне забезпечення, створене на мові асемблера, яке інтегрувало технології EAX і Aureal та дозволяло використовувати музику трекера, що давало розробникам рівнів гнучкість у тому, як відтворювати саундтрек гри на певну точку на картах. Стів Полдж, автор плагіна Reaper Bots для Quake, запрограмував систему штучного інтелекту на основі знань, які він отримав у свого попереднього роботодавця IBM, розробляючи протоколи маршрутизатора.

За словами Суїні, найскладнішою частиною рушія для програмування був рендерер; йому довелося кілька разів переписувати його основний алгоритм під час розробки. Він знайшов інфраструктуру, що з'єднує всі підсистеми, менш

«гламурною». Незважаючи на те, що він вимагав значних особистих зусиль, він сказав, що рушій був його улюбленим проектом в Epic, додавши: «Написання першого Unreal Engine було 3,5-річним, широким туром по сотням унікальних тем у програмному забезпеченні та було неймовірно повчальним». Серед його особливостей були - виявлення зіткнень, кольорове освітлення та обмежена форма фільтрації текстури. Він також інтегрував редактор рівнів, UnrealEd, який мав підтримку операцій з конструктивною геометрією в реальному часі ще в 1996 році, дозволяючи картографам змінювати макет рівня на льоту. Незважаючи на те, що Unreal було розроблено, щоб конкурувати з id Software (розробником Doom і Quake), співзасновник Джон Кармак похвалив гру за використання 16-бітного кольору та зауважив, що в ній реалізовані візуальні ефекти, такі як об'ємний туман.

Unreal була відома своїми графічними інноваціями, але ще багато аспектів гри були невідшліфованими, геймери скаржились на її високі системні вимоги та проблеми з онлайн-грою. Epic звернула увагу на ці моменти під час розробки Unreal Tournament, включивши кілька вдосконалень у рушій, спрямованих на оптимізацію продуктивності на машинах нижчого класу та покращення мережевого коду, а також удосконалення штучного інтелекту для ботів для відображення координації в ігрових режимах типу командних змагань, таких як Capture the Flag. Спочатку запланована як пакет розширення для Unreal, гра також постачалася з покращеною якістю зображення завдяки підтримці алгоритму стиснення S3TC, дозволяючи створювати 24-бітові текстури високої роздільної здатності без шкоди для продуктивності. На додаток до того, що він був доступний для Windows, Linux, Mac і Unix, рушій було перенесено через Unreal Tournament на PlayStation2 і, за допомогою Secret Level, на Dreamcast.

Рис.1.2 – Harry Potter and the Sorcerer's Stone для ПК на рушію Unreal Tournament.

Наприкінці 1999 року газета The New York Times зазначила, що було шістнадцять зовнішніх проектів з використанням технології Epic, включаючи Deus Ex, The Wheel of Time і Duke Nukem Forever, останній з яких спочатку базувався на рушію Quake II. На відміну від id Software, чий движковий бізнес пропонував лише вихідний код, Epic надавав підтримку ліцензіатам і збирався разом з їхніми лідерами, щоб обговорити вдосконалення своєї системи розробки ігор, яка всередині отримала назву Unreal Tech Advisory Group. Хоча виробництво коштувало близько 3 мільйонів доларів США, а ліцензії — до 350 000 доларів США, Epic надала гравцям можливість модифікувати свої ігри за допомогою UnrealEd і мови сценаріїв під назвою UnrealScript, даючи поштовх створенню спільноти ентузіастів навколо ігрового рушія, створеного для розширення на кілька поколінь ігор.

1.1.2 Друге покоління (Unreal Engine 2)

У жовтні 1998 року IGN повідомила на основі інтерв'ю з афілійованою компанією Voodoo Extreme, що Суїні досліджує свій рушій наступного покоління. Розробка почалася роком пізніше, друга версія дебютувала в 2002 році з America's Army, безкоштовним багатокористувацьким шутером, розробленим армією США, як пристрій вербування. Невдовзі після цього Epic випустив Unreal Championship на Xbox, одну з перших ігор, яка використовує Microsoft Xbox Live.

Незважаючи на те, що це покоління базується на своєму попереднику, це покоління побачило помітний прогрес у термінах візуалізації, а також нові вдосконалення набору інструментів. Здатність запускати рівні майже в 100 разів детальніші, ніж ті, які є в Unreal, рушій інтегрував різноманітні функції, включаючи інструмент кінематографічного редагування, системи частинок, плагіни експорту для 3D Studio Max і Maya, а також скелетну анімацію - система вперше була продемонстрована у версії Unreal Tournament для PlayStation 2. Крім того, інтерфейс користувача для UnrealEd був переписаний на C++ з використанням інструментарію wxWidgets, який був «найкращим доступним» на той час.

Рис.1.3 – Killing Floor на рушію Unreal Engine 2.

Epic використовував фізичний рушій Karma, стороннє програмне забезпечення від британської студії Math Engine, щоб керувати фізичними симуляціями, такими як зіткнення гравців ragdoll і довільну динаміку твердого тіла. У Unreal Tournament 2004 було успішно реалізовано ігровий процес на транспортних засобах, що дозволило широкомасштабні бої. Незважаючи на те, що Unreal Tournament 2003 підтримував фізику автомобіля через рушій Karma, як продемонструвала тестова карта з «поспішно сконструйованим транспортним засобом», лише Psyonix створив модифікацію базового коду Epic, і гра отримали повністю кодовані транспортні засоби. Вражені їхніми зусиллями, Epic вирішила включити його до свого наступника як офіційний ігровий режим під назвою

Onslaught, найнявши Psyonix як підрядника. Пізніше Psyonix розробить Rocket League, перш ніж її придбає Epic у 2019 році.

Спеціалізована версія UE2 під назвою UE2X була розроблена для Unreal Championship 2: The Liandri Conflict на оригінальній платформі Xbox із оптимізацією, специфічною для цієї консолі. У березні 2011 року Ubisoft Montreal показало, що UE2 успішно працює на Nintendo 3DS через Tom Clancy's Splinter Cell 3D.

1.1.3 Третє покоління (Unreal Engine 3)

Скріншоти Unreal Engine 3 були представлені в липні 2004 року, на той момент рушій уже розроблявся понад 18 місяців. Рушій базувався на першому поколінні, але містив нові функції. Основні архітектурні рішення, видимі програмістам: об'єктно-орієнтований дизайн, підхід до сценаріїв, керованих даними, і досить модульний підхід до підсистем все залишилися з Unreal Engine 1. Але ті частини гри, які дійсно видимі для геймерів – рендерер, система фізики, звукова система та інструменти – усе це стало новим та значно потужнішим. На відміну від Unreal Engine 2, який все ще підтримував конвеєр із фіксованими функціями, Unreal Engine 3 був розроблений для використання переваг повністю програмованого апаратного забезпечення шейдерів. Усі розрахунки освітлення та тіні виконувалися на піксель, а не на вершину. Що стосується рендеринга, Unreal Engine 3 забезпечив підтримку гамма-коректного рендерера з високим динамічним діапазоном. Першими іграми, випущеними з використанням Unreal Engine 3, були Gears of War для Xbox 360 і RoboBlitz для Windows, обидві випущені 7 листопада 2006 року.

Спочатку Unreal Engine 3 підтримував лише платформи Windows, PlayStation 3 і Xbox 360, тоді як iOS (вперше продемонстрована в Epic Citadel) і Android були додані пізніше в 2010 році, причому Infinity Blade стала першою грою для iOS, а Dungeon Defenders - першою грою для Android. У 2011 році було оголошено, що рушій підтримуватиме Adobe Flash Player 11 через API з апаратним прискоренням Stage 3D і що він використовується у двох іграх Wii U – Batman: Arkham City та Aliens: Colonial Marines. У 2013 році Epic об'єдналася з Mozilla, щоб вивести

Unreal Engine 3 у web; використовуючи підмову asm.js і компілятор Emscripten, вони змогли перенести рушій за чотири дні.

За час існування UE3 було впроваджено значні оновлення, включаючи покращене середовище, що руйнується, динаміку м'якого тіла, симуляція великого натовпу, функціональність iOS, інтеграцію Steamworks, рішення для глобального освітлення в реальному часі, і стереоскопічне 3D на Xbox 360 через TriOviz for Games Technology. Підтримка DirectX 11 була продемонстрована в демо-версії Samaritan, яка була оприлюднена на Конференції розробників ігор у 2011 році та створена компанією Epic Games у тісному партнерстві з Nvidia, та інженерами, працюючими по всій країні, щоб просувати графіку реального часу до нового високого рівня.

Рис.1.4 – Скріншот з демо-версії Samaritan.

1.1.4 Четверте покоління (Unreal Engine 4)

У серпні 2005 року Марк Рейн, віце-президент Epic Games, повідомив, що Unreal Engine 4 розроблявся протягом двох років. Говорячи в інтерв'ю на початку 2008 року, Суїні заявив, що він був фактично єдиною людиною, яка працювала над рушієм, хоча він підтвердив, що його відділ досліджень і розробок почне розширюватися пізніше того ж року, розробляючи рушій паралельно з Unreal Engine 3.

Epic представила UE4 обмеженій кількості учасників на Конференції розробників ігор 2012, а відео рушія, яке демонстрував технічний художник Алан Уіллард, було оприлюднено 7 червня 2012 року через GameTrailers TV. Однією з основних функцій, запланованих для UE4, було глобальне освітлення в режимі

реального часу за допомогою трасування воксельного конуса, що усуває попередньо обчислене освітлення. Однак ця функція під назвою Sparse Voxel Octree Global Illumination (SVOGI) і продемонстрована в демонстрації Elemental була замінена подібним, але менш дорогим алгоритмом через проблеми з продуктивністю. UE4 також включає нову систему візуальних сценаріїв «Blueprints» (наступник «Kismet» UE3), яка дозволяє швидко розвивати логіку гри без використання коду, що призводить до зменшення розриву між технічними художниками, дизайнерами , та програмістами.

Рис.1.5 – Інтерактивна візуалізація архітектури, розроблена на рушію Unreal Engine 4 (2015)

19 березня 2014 року на Конференції розробників ігор (GDC) компанія Epic Games випустила Unreal Engine 4 за новою моделлю ліцензування. За місячну підписку в розмірі 19 доларів США розробники отримували доступ до повної версії механізму, включаючи вихідний код C++, який можна було завантажити через GitHub. За будь-який випущений продукт стягувався роялті у розмірі 5% від валового доходу. Першою грою, випущеною з використанням Unreal Engine 4, була Daylight, розроблена з раннім доступом до рушію і випущена 29 квітня 2014 року.

4 вересня 2014 року Epic безкоштовно випустила Unreal Engine 4 для шкіл і університетів, включаючи особисті копії для студентів, які навчаються на акредитованих програмах з розробки відеоігор, інформатики, мистецтва, архітектури, моделювання та візуалізації. Epic відкрила Unreal Engine Marketplace для придбання ігрових активів. 19 лютого 2015 року Epic запустила Unreal Dev Grants, фонд розвитку розміром 5 мільйонів доларів, метою якого є надання грантів творчим проектам з використанням Unreal Engine 4.

У березні 2015 року Epic безкоштовно випустила Unreal Engine 4 разом із усіма майбутніми оновленнями для всіх користувачів. Натомість Epic встановила вибірковий графік роялті, вимагаючи 5% доходу за продукти, які приносять понад 3000 доларів на квартал. Суїні заявив, що коли вони перейшли на модель підписки в 2014 році, використання Unreal зросло в 10 разів і завдяки багатьом меншим розробникам, і вважав, що вони отримають ще більше використання завдяки цій новій схемі ціноутворення.

Намагаючись залучити розробників Unreal Engine, Oculus VR оголосив у жовтні 2016 року, що буде платити роялті за всі назви Oculus Rift на базі Unreal, опубліковані в їх магазині, у розмірі до перших 5 мільйонів доларів валового доходу від гри.

Щоб підготуватися до випуску свого безкоштовного режиму “Королівська битка” у Fortnite у вересні 2017 року, Epic довелося внести ряд модифікацій в Unreal Engine, які допомогли їй обробляти велику кількість (до 100) підключень до одного сервера. зберігаючи при цьому високу пропускну здатність і покращуючи рендеринг великого відкритого внутрішньоігрового світу.

З відкриттям Epic Games Store у грудні 2018 року Epic перестав стягувати комісію в розмірі 5% з ігор, які використовують Unreal Engine і випущені через Epic Games Stores, поглинаючи ці витрати як частину базової 12%-ї вартості Epic на покриття інших витрат.

Починаючи з 13 травня 2020 року та ретроактивно до 1 січня 2020 року сума звільнення від роялті збільшена до 1 000 000 доларів США валового доходу за весь період життя продукту.

Unreal Engine 4 офіційно підтримував такі платформи станом на серпень 2021 (версія 4.27): Windows, macOS, Linux, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, PlayStation 5, Xbox Series X/S, Stadia, Magic Leap, HTC Vive, Oculus, PlayStation VR, OSVR, Samsung Gear VR, і HoloLens 2. Раніше він ще підтримував Google Daydream і HTML5.

1.1.5 П'яте покоління (Unreal Engine 5)

13 травня 2020 року було представлено Unreal Engine 5, який підтримує всі існуючі системи, включаючи консолі наступного покоління PlayStation 5 і Xbox Series X/S. Робота над рушієм почалася приблизно за два роки до його оголошення. Він був випущений у ранньому доступі 26 травня 2021 року та офіційно запущений для розробників 5 квітня 2022 року.

Однією з його головних особливостей є Nanite, механізм, який дозволяє імпортувати в ігри високодеталізований вихідний фотографічний матеріал. Технологія віртуалізованої геометрії Nanite дозволяє Epic скористатися перевагами минулого придбання Quixel, найбільшої у світі бібліотеки фотограмметрії станом на 2019 рік. Мета Unreal Engine 5 полягала в тому, щоб якомога легше розробникам створювати детальні ігрові світи. без необхідності витрачати зайвий час на створення нових детальних активів. Nanite може імпортувати майже будь-яке інше тривимірне представлення об’єктів і середовищ, включаючи Zbrush і CAD-моделі, що дає змогу використовувати ресурси плівкової якості. Nanite автоматично обробляє рівні деталізації (LOD) цих імпортованих об’єктів відповідно до цільової платформи та відстані малювання, завдання, яке художник мав би виконати інакше.

Lumen - ще один компонент, який описується як «повністю динамічне рішення для глобального освітлення, яке миттєво реагує на зміни сцени та світла». Lumen позбавляє художників і розробників необхідності створювати світлову карту для певної сцени, але натомість розраховує світлові відбиття та тіні на льоту, таким чином дозволяючи поведінку джерел світла в реальному часі.

Віртуальні карти тіней - це ще один компонент, доданий в Unreal Engine 5, який описується як «новий метод відображення тіней, який використовується для

забезпечення узгодженого затінення високої роздільної здатності, який працює з ресурсами кіноякості та великими, динамічно освітленими відкритими світами». Віртуальні карти тіней відрізняються від звичайної реалізації карти тіней своєю надзвичайно високою роздільною здатністю, більш детальними тінями та відсутністю тіней, що з'являються та зникають, що можна знайти в більш поширеній техніці карт тіней через каскади тіней.

Додаткові компоненти включають Niagara для динаміки рідини та частинок і Chaos для фізичного механізму.

З потенційно десятками мільярдів багатокутників, присутніх на одному екрані з роздільною здатністю 4K, Epic також розробила Unreal Engine 5, щоб скористатися перевагами майбутніх високошвидкісних рішень для зберігання даних із консольним обладнанням наступного покоління, яке використовуватиме поєднання оперативної пам'яті та спеціальних твердотільних накопичувачів. Epic тісно співпрацювала з Sony в оптимізації Unreal Engine 5 для PlayStation 5, працюючи над архітектурою зберігання консолі. Щоб продемонструвати легкість створення деталізованого світу з мінімальними зусиллями, у травні 2020 року було представлено рушій, де було продемонстровано демо під назвою «Lumen in the Land of Nanite», запущене на PlayStation 5, яке було створено здебільшого за допомогою ресурсів бібліотеки Quixel та використання Nanite, Lumen та інших компонентів Unreal Engine 5 для створення фотореалістичної печери, яку можна досліджувати. Epic підтвердила, що Unreal Engine 5 також буде повністю підтримуватися на Xbox Series X, але під час анонсу увагу було зосереджено на PlayStation 5 як результат їхньої роботи з Sony у попередні роки.

Рис.1.6 – Печерна система в демонстрації Unreal Engine 5 «Lumen in the Land of Nanite»

Epic планує використовувати Fortnite як тестовий стенд для Unreal Engine 5, щоб продемонструвати, що цей рушій може зробити для індустрії, з грою, яка буде використовувати Unreal Engine 5. Режим Fortnite's Battle Royale отримав візуальні покращення через Unreal Engine 5.1 із запуском глави 4 4-го грудня 2022 року. The Matrix Awakens, додатковий досвід перед випуском The Matrix Resurrections, був розроблений Epic як подальша демонстрація Unreal Engine 5 разом з іншими технологіями, які вони придбали протягом 2020 і 2021 років, включаючи MetaHuman Creator, розроблений та інтегрований в Unreal Engine 5 за допомогою технологій від 3Lateral, Cubic Motion і Quixel.

MetaHuman Creator — це проект, заснований на технології трьох компаній, придбаних Epic — 3Lateral, Cubic Motion і Quixel, щоб дозволити розробникам швидко створювати реалістичних людських персонажів, які потім можна експортувати для використання в Unreal. Завдяки партнерству з Cesium Epic планує запропонувати безкоштовний плагін для надання 3D геопросторових даних для користувачів Unreal, дозволяючи їм відтворювати будь-яку частину нанесеної на карту поверхні Землі. Epic включатиме RealityCapture, продукт, який вона придбала після придбання Capturing Reality, який може генерувати 3D-моделі будь-якого об’єкта з колекції фотографій, зроблених під різними кутами, та різні

інструменти проміжного програмного забезпечення, запропоновані Epic Game Tools.

Unreal Engine 5 зберігає поточну модель роялті, коли розробники повертають Epic Games 5% валового доходу, хоча ця комісія не стягується за продажі, здійснені через Epic Games Store. Крім того, Epic разом із Unreal Engine 5 оголосила, що вони не братимуть жодної комісії з ігор, які використовують будь-яку версію Unreal Engine, за перші 1 мільйон доларів США валового доходу заднім числом до 1 січня 2020 року. Epic заявила про ліцензування робочого місця на Unreal Engine, починаючи з квітня 2024 року, для його використання в неігрових програмах, наприклад у кіно- та телевізійному виробництві, якщо їх дохід перевищує 1 мільйон доларів США, причому кожне робоче місце коштує 1850 доларів США на рік.

У березні 2024 року Epic Games запустили Project Titan, спільний арт-джем для створення безкоштовного зразка проекту відкритого світу для Unreal Engine.

1.2 Основні терміни Unreal Engine

Розглянемо найпоширеніші терміни, які використовуються під час роботи з Unreal Engine. Інколи без їх розуміння буде важко зрозуміти про, що йде мова.

Проект (Project) - містить увесь вміст гри. Він містить декілька папок на диску, наприклад, креслення та матеріали. Папки можуть називатися та впорядковуватися в проекті як завгодно. Панель Content Browser у Unreal Editor показує ту саму структуру каталогів, що й у папці Project на диску.

З кожним проектом пов’язаний файл .uproject. Файл .uproject дає змогу створити, відкрити чи зберегти проект. Можна створити будь-яку кількість різних проектів і працювати над ними паралельно.

Unreal Engine має шаблони, які є відправною точкою для різних типів проектів. Шаблони містять попередньо створені ресурси, такі як манекени, які вже налаштовані для введення даних від гравця та анімації, або зразки сцен для дослідження освітлення та стилізованого візуалізації.

Окрім існуючих шаблонів Unreal Engine, можна перетворити будь-який

проект на спеціальний шаблон, щоб повторно використовувати його ресурси та конфігурацію за потреби.

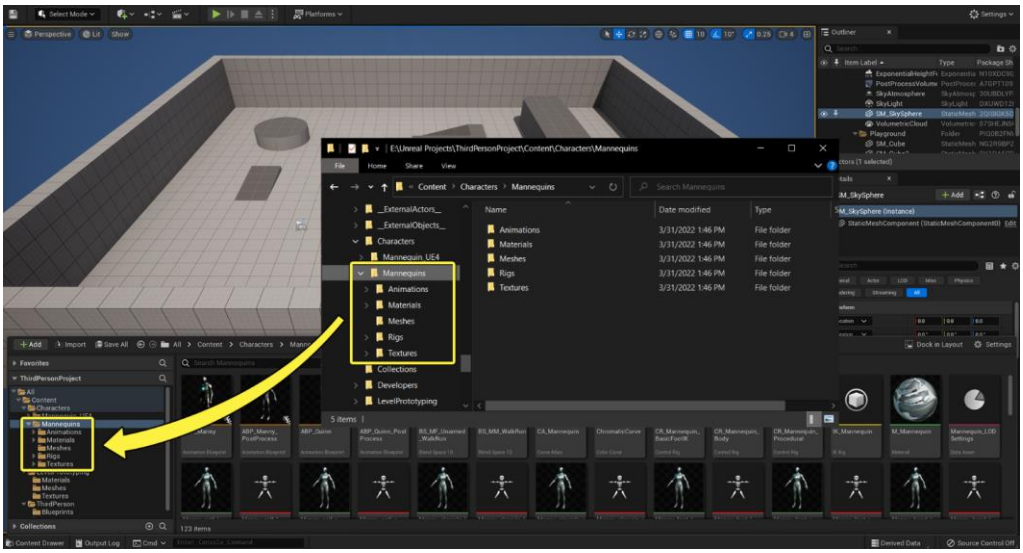

Рис.1.7 – Панель Content Browser

Система візуальних сценаріїв Blueprint - це повна система сценаріїв ігрового процесу, яка використовує інтерфейс на основі вузлів для створення елементів ігрового процесу в Unreal Editor. Як і в багатьох поширених мовах сценаріїв, він використовується для визначення об’єктно-орієнтованих класів або об’єктів у механізмі. Об’єкти, визначені за допомогою Blueprint, у розмовній мові часто називають «кресленнями» або планами.

Ця система є надзвичайно гнучкою та потужною, оскільки надає дизайнерам можливість використовувати практично повний спектр концепцій та інструментів, які зазвичай доступні лише програмістам. Крім того, спеціальна розмітка Blueprint, доступна в C++ реалізації Unreal Engine, дозволяє програмістам створювати базові системи, які можуть бути розширені дизайнерами.

Об'єкт (Object) - є найпростішим класом у Unreal Engine, іншими словами, об’єкти діють як будівельні блоки та містять багато основних функцій для Активів (Assets). Майже все в Unreal Engine успадковує (або отримує певну функціональність) від об’єкта.

У C++ UObject є базовим класом усіх об’єктів; він реалізує такі функції, як збірка сміття, підтримка метаданих (UProperty) для надання змінних у Unreal Editor та серіалізація для завантаження та збереження.

Клас (Class) - визначає поведінку та властивості конкретного актора чи об’єкта в Unreal Engine. Класи є ієрархічними, тобто клас успадковує інформацію від свого батьківського класу (тобто класу, від якого він був похідним або «підкласом»), і передає цю інформацію своїм нащадкам. Класи можна створювати в коді C++ або в Blueprints.

Клас Blueprint, який часто скорочують як Blueprint, — це Актив, який дозволяє творцям контенту легко додавати функціональні можливості до існуючих класів ігрового процесу. Креслення (Blueprints) створюються в Unreal Editor візуально, а не шляхом введення коду, і зберігаються як активи в пакеті вмісту. Вони, по суті, визначають новий клас або тип Актора, який потім можна розмістити на картах як екземпляри, що поводяться як будь-який інший тип Актора.

Кожен клас ігрового процесу (gameplay class) в Unreal Engine складається з файлу заголовка класу (.h) і вихідного файлу класу (.cpp). Заголовок класу містить оголошення класу та його членів, таких як змінні та функції, тоді як вихідний файл класу є місцем, де функціональні можливості класу визначаються шляхом реалізації функцій, які належать до класу.

Класи в Unreal Engine мають стандартизовану схему іменування, щоб миттєво розуміти, що це за клас, просто подивившись на першу літеру або префікс.

Префікси для ігрових класів:

префікс A - поширюється на базовий клас ігрових об’єктів, які створюються. Це Актори, і їх можна створити безпосередньо у світі.

префікс U - розширює базовий клас усіх ігрових об’єктів. Вони не можуть бути безпосередньо представлені у світі; вони повинні належати Актору. Зазвичай це такі об’єкти, як компоненти.

Актор (Actor) - це будь-який об’єкт, який можна розмістити на рівні, наприклад камера, статична сітка або місце початку гравця. Актори підтримують тривимірні перетворення, такі як переміщення, обертання та масштабування. Їх можна створювати (породжувати) і знищувати за допомогою ігрового коду (C++ або Blueprints).

У C++ AActor є базовим класом усіх акторів.

Акторів можна розглядати, в певному сенсі, як контейнери, які містять спеціальні типи об’єктів, які називаються компонентами. Різні типи компонентів можна використовувати для керування тим, як актори рухаються, як вони відображаються тощо. Іншою основною функцією акторів є реплікація властивостей і викликів функцій у мережі під час гри.

Компоненти пов’язуються з актором, який їх містить, коли вони створюються.

Кілька основних типів компонентів:

- UActorComponent: це базовий компонент. Його можна включити як частину актора. Він може поставити галочку, якщо ви цього хочете. Компоненти ActorComponents пов’язані з конкретним Актором, але не існують у жодному конкретному місці у світі. Зазвичай вони використовуються для концептуальних функцій, як-от ШІ або інтерпретація введених гравцями даних.
- USceneComponent: - це ActorComponents, які мають трансформації. Трансформація — це позиція у світі, визначена розташуванням, обертанням і масштабом. SceneComponents можна приєднувати один до одного в ієрархічний спосіб. Розташування, обертання та масштаб актора беруться з SceneComponent, який знаходиться в корені ієрархії.
- UPrimitiveComponent: - це SceneComponents, які мають певне графічне представлення (наприклад, сітка або система частинок). Багато цікавих налаштувань фізики та зіткнень саме тут.

Актори підтримують наявність ієрархії SceneComponents. Кожен Актор також має властивість RootComponent, яка визначає, який Компонент діє як кореневий для Актора. Самі актори не мають трансформацій, а отже, не мають місць, поворотів чи масштабів. Натомість вони покладаються на перетворення своїх компонентів; точніше, їх кореневий компонент. Якщо цей компонент є SceneComponent, він надає інформацію про трансформацію для актора. В іншому випадку Актор не матиме трансформації. Інші приєднані Компоненти мають трансформацію відносно Компонента, до якого вони приєднані.

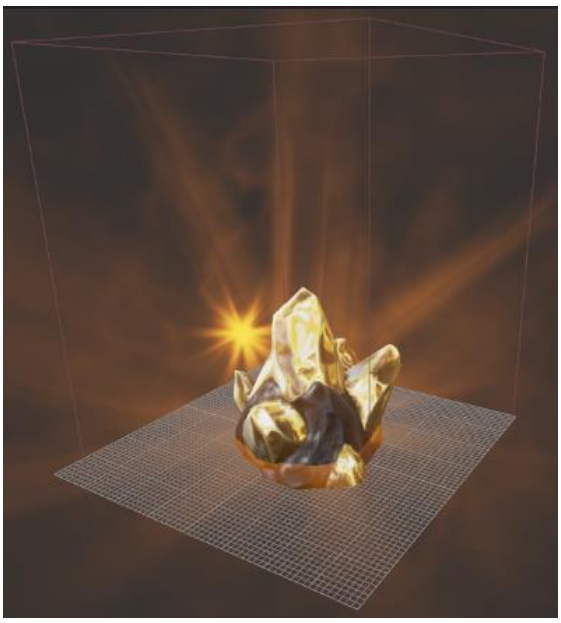

Рис.1.8 – Приклад Актора

Ієрархія даного Актора може виглядати наступним чином:

- Root - SceneComponent: базовий компонент сцени для встановлення базового розташування актора у світі.

- StaticMeshComponent: сітка, що представляє золоту руду.

- ParticleSystemComponent: випромінювач блискучих частинок, прикріплений до золотої руди.

- AudioComponent: циклічний металевий звуковий випромінювач, прикріплений до золотої руди.

- BoxComponent: вікно зіткнення, яке буде використовуватися як тригер для події перекриття для збирання золота.

Кастинг (Casting) - це дія, під час якої Актор певного класу намагається поводитися з ним так, ніби він належить до іншого класу. Кастинг може бути успішним або невдалим. Якщо кастинг вдасться, отримується доступ до функціональних можливостей класу.

Наприклад, при створенні гри, у якій є кілька типів об’єктів, які можуть по-різному впливати на персонажа гравця. Одним із таких об’єктів є Вогонь, який з часом зменшує здоров’я гравця. Коли гравець збігається з будь-яким об’єктом на Рівні, можна застосувати цей об’єкт до Вогню, щоб спробувати отримати доступ до його функції "пошкодити здоров'ю гравця".

Якщо спроба вдалась - тобто якщо гравець стоїть у вогні — здоров’я гравця починає зменшуватися.

Якщо спроба невдала - тобто, якщо гравець стоїть у будь-якому іншому типі Об’єкту - це не вплине на його здоров’я.

Кастинг відрізняється від простої перевірки, чи належить Актор до даного класу, яка повертатиме двійкову відповідь (так чи ні), але не дозволить взаємодіяти з будь-якою конкретною функціональністю цього класу.

Компонент (Component) - це частина функціональності, яку можна додати до Актора.

Коли додається компонент до Актора, актор може використовувати функціональні можливості, які надає компонент. Наприклад:

- Spot Light Component - змусить Актора випромінювати світло, як прожектор;
- Rotating Movement Component - змусить Актора обертатися;
- Audio Component - надасть Акторові можливість відтворювати звуки.

Компоненти мають бути прикріплені до Актора й не можуть існувати самі по собі.

Додаючи Компоненти до Актора, об’єднуються фрагменти, які складають Актора як єдине ціле. Наприклад, автомобіль (Актор) складається з коліс, керма, кузова, ліхтарів тощо (Компоненти). Подібним чином актор, який представляє персонажа гравця, може мати індивідуальні компоненти для скелетної сітки (візуального представлення персонажа), камери, яка слідує за персонажем, і контролера, який отримує дані від гравця.

Після об’єднання різних компонентів, які складають Актора, його можна розмістити на своєму Рівні, навіть не надаючи жодного сценарію Blueprint (або коду C++), який диктує, як Актор має функціонувати. Згідно з наведеним вище прикладом, автомобіль (Актор) може існувати як об’єкт у світі (Рівень) без драйвера (Blueprint або C++ коду), який вказує йому, що робити. Потім код Blueprint або C++ можна використовувати для доступу до кожного з компонентів автомобіля окремо (наприклад, якщо натиснути педаль газу Component, логіка Blueprint може змусити автомобіль пришвидшитися).

Пішаки (Pawn) - є підкласом Актора і служать як ігровий аватар або персона (наприклад, персонажі в грі). Пішаками може керувати гравець або штучний інтелект гри. Пішак - це фізичне представлення гравця або штучного інтелекту у світі. Це означає не лише те, що пішак визначає, як візуально виглядає гравець або штучний інтелект, але й те, як він взаємодіє зі світом у плані зіткнень та інших фізичних взаємодій. За певних обставин це може ввести в оману, оскільки деякі типи ігор можуть не мати видимої сітки або аватара гравця в грі. Незважаючи на це, пішак все ще представляє фізичне розташування, обертання тощо гравця чи сутності в грі.

За замовчуванням між контролерами та пішаками існує взаємозв'язок один до одного; тобто кожен контролер контролює лише одного пішака в будь-який момент часу. Крім того, пішаки, створені під час гри, не перебувають автоматично під контролем.

Персонаж (Character) - це підклас Пішака Актора, який призначений для використання як персонаж гравця. Підклас «Персонаж» включає налаштування зіткнення, прив'язки введення для двоногого руху та додатковий код для керованого гравцем руху. Він може виконувати основні рухи, подібні до людських, він може плавно відтворювати рухи в мережі та має деякі функції, пов'язані з анімацією.

Контролер гравця (Player Controller) - приймає дані гравця та перетворює їх на взаємодію в грі. У кожній грі є принаймні один контролер гравця. Контролер гравця часто має пішака або персонажа як репрезентацію гравця в грі. Тобто контролер гравця - це інтерфейс між пішаком і людиною-гравцем, який ним керує.

Контролер гравця також є основною точкою взаємодії з мережею для багатокористувацьких ігор. Під час гри для кількох гравців сервер має один екземпляр контролера гравця для кожного гравця в грі, оскільки він повинен мати можливість здійснювати виклики мережевих функцій для кожного гравця. Кожен клієнт має лише контролер гравця, який відповідає його гравцеві, і може використовувати лише його контролер гравця для зв'язку із сервером.

Під час налаштування контролераа гравця слід продумати які функції

повинні бути в PlayerController, а які — у вашому пішаку. Можна обробити всі введення в пішаку, особливо для менш складних випадків. Однак, якщо у є складніші потреби, як-от кілька гравців в одному ігровому клієнті, або можливість динамічно змінювати персонажів під час виконання, то краще обробляти введення в PlayerController. У цьому випадку PlayerController вирішує, що робити, а потім віддає пішаку команди (наприклад, «почати присідати», «стрибати»).

Крім того, у деяких випадках необхідно додати обробку вводу або інші функції до PlayerController. PlayerController зберігається протягом усієї гри, а пішак може бути тимчасовим. Наприклад, у стилі гри Deathmatch гравець може померти та відродитися, тож він отримає нового пішака, але PlayerController залишиться тим самим. У цьому прикладі, якщо зберігати рахунок на пішаку гравця, рахунок буде скинуто, але якщо рахунок зберігається на PlayerController, цього не буде.

Пов’язаний клас C++ - PlayerController.

Контролер ШІ (AI Controller). Подібно до того, як контролер гравця має пішака як представлення гравця в грі, контролер штучного інтелекту має пішака для представлення неігрового персонажа (NPC) у грі. За замовчуванням пішаки та персонажі отримають базовий контролер штучного інтелекту, якщо тільки ними не володіє контролер гравця або не буде наказано не створювати контролер штучного інтелекту для себе.

У той час як PlayerController покладається на гравця-людину для прийняття рішень про те, що робити, AIController більше зосереджений на реагуванні на вхідні дані з середовища та ігрового світу. Робота AIController полягає в тому, щоб спостерігати за навколишнім світом, приймати рішення та відповідним чином реагувати без прямого введення з боку людини-гравця.

Пов’язаний клас C++ - це AIController.

Стан гравця (Player State) – це стан учасника гри, наприклад гравець-людина або бот, який імітує гравця. Штучний інтелект без гравців, який існує як частина ігрового світу, не має стану гравця.

Наприклад стан гравця може містити:

- ім'я;
- поточний рівень;
- здоров'я;
- рахунок;
- чи вони зараз несуть прапор у грі Capture the Flag.

Для багатокористувацьких ігор стани гравців для всіх гравців існують на всіх машинах і можуть копіювати дані з сервера на клієнта для підтримки синхронізації. Це відрізняється від контролера гравця, який існуватиме лише на машині гравця, якого він представляє.

Пов’язаний клас C++ - PlayerState.

Режим гри (Game Mode) - встановлює правила гри, у яку грають. Ці правила можуть включати:
- кількість присутніх гравців і глядачів, а також максимальна дозволена кількість гравців і глядачів;
- як гравці приєднуються до гри;
- чи можна призупинити гру;
- будь-яка поведінка, характерна для гри, наприклад умови виграшу.

Можна встановити ігровий режим за замовчуванням у налаштуваннях проекту та змінити його для різних рівнів. Незалежно від того, як це буде реалізовано, можна мати лише один ігровий режим для кожного рівня.

У грі для кількох гравців ігровий режим існує лише на сервері, а правила копіюються (надсилаються) кожному з підключених клієнтів.

Наразі існує два базові класи, які зазвичай використовуються для режимів гри. Версія двигуна 4.14 представляє AGameModeBase, який є базовим класом для всіх ігрових режимів і є спрощеною та вдосконаленою версією класичного AGameMode. AGameMode, базовий клас ігрового режиму до версії 4.14, все ще існує та функціонує, як і раніше, але тепер є дочірнім класом AGameModeBase. AGameMode більше підходить для стандартних типів ігор, таких як багатокористувацькі шутери, завдяки реалізації концепції стану матчу.

AGameModeBase — це новий ігровий режим за замовчуванням, який включається в нові проекти коду завдяки його простоті та ефективності.

Стан гри (Game State) – це контейнер, у якому зберігається інформація, яку потрібно відтворити для кожного клієнта в грі. Простіше кажучи, це «Стан гри» для всіх підключених. Концептуально ігровий стан має керувати інформацією, яка має бути відома всім підключеним клієнтам і є специфічною для ігрового режиму, але не є специфічною для будь-якого окремого гравця.

Приклади того, що може містити стан гри:

- інформація про рахунок гри;
- скільки триває гра (включаючи час роботи до приєднання локального гравця);
- скільки персонажів AI створити залежно від кількості гравців у світі.

Для багатокористувацьких ігор існує один локальний екземпляр стану гри на комп'ютері кожного гравця. Локальні екземпляри стану гри отримують оновлену інформацію з екземпляра стану гри на сервері.

Стан гри не є найкращим місцем для відстеження конкретних речей гравця, наприклад, скільки очок один конкретний гравець набрав для команди в матчі Capture The Flag, тому що це може більш чітко оброблятися Player State. Загалом, GameState має відстежувати властивості, які змінюються під час гри та є актуальними та видимими для всіх. Хоча ігровий режим існує лише на сервері, стан гри існує на сервері та копіюється на всіх клієнтів, підтримуючи всі підключені машини в актуальному стані під час гри.

Пов'язаний клас C++ - GameState.

Пензель (Brush) - це актор, який описує тривимірну форму, наприклад куб або сферу. Можна розмістити пензлі на рівні, щоб визначити геометрію рівня (вони відомі як пензлі Binary Space Partition або BSP). Це корисно, наприклад, при необхідності швидко заблокувати рівень. Концептуально найкраще вважати, що Пензель заповнює та вирізає об'єми простору на рівні. Можна використовувати пензлі для створення базових форм для процесу проектування рівнів, або як постійні пристосування, або як щось тимчасове для тестування, поки художники створять остаточні сітки.

Раніше Пензлі використовувалися як основний будівельний блок у дизайні рівнів. Однак ця роль була передана статичним сітям і вбудованим інструментам моделювання, які набагато ефективніші. Однак геометричні пензлі можуть бути корисними на ранніх стадіях продукту для швидкого створення прототипів рівнів і об’єктів.

Об’єми (Volume) - це обмежені тривимірні простори, які мають різне використання залежно від доданих до них ефектів. Наприклад:

- блокувальні Об’єми (Blocking Volumes) - невидимі та використовуються, щоб Актори не проходили крізь них;
- Об’єми, що завдають болю (Pain Causing Volumes) - з часом завдають шкоди будь-якому Актору, який їх перекриває;
- тригерні Об’єми (Trigger Volumes) запрограмовані на виклик подій, коли Актор входить або виходить з них.

Рівень (Level) – це визначена область гри. Рівні містять усе, що гравець може бачити та з чим взаємодіяти, наприклад геометрію, Пішаків та Акторів. У відеоіграх зазвичай є кілька рівнів із чітко окресленими переходами між ними (наприклад, коли при подоланні фінального боса на рівні, відбувається перехід до наступного). Для інших типів інтерактивних продуктів, створених за допомогою Unreal Engine, можна використовувати різні рівні для переходу між різними видами вітрин або середовищ.

Unreal Engine зберігає кожен рівень як окремий файл .umap, тому іноді рівні ще називають Картами.

Для створення рівня потрібен наступний мінімальний список елементів:

- файл .umap, який є самим рівнем. Це як контейнер, який містить усе інше;
- середовище зі статичних сітчастих Акторів. Це можуть бути дерева, скелі, стіни або будь-які інші предмети навколишнього середовища. У деяких середовищах також використовуються інші типи Акторів, наприклад пейзажні Актори або водні Актори;
- Персонаж гравця, представлений Актором Skeletal Mesh;
- один або кілька ліхтарів різних типів;

- навколишні звуки та звукові ефекти (наприклад, звуки кроків);
- складні рівні можуть містити інші функції, такі як ефекти частинок, візуальна пост-обробка, потокове передавання рівнів тощо.

Світ (World) - це контейнер для всіх рівнів, які складають гру. Він обробляє потокову передачу рівнів і створення динамічних Акторів. Світ характеризує загальні властивості «простору», наприклад, силу тяжіння й туман, у якому розташовуються всі актори. Також може містити в собі параметри ігрового процесу, як, наприклад, ігровий режим, для якого призначений рівень.

Актив (Asset). Будь-який вміст у проекті Unreal Engine є Активом. Активи можна розглядати як будівельні блоки, які використовуються для створення гри та додатку.

Активи можуть бути різних типів, наприклад статичні сітки, матеріали, системи частинок і звукові сигнали. Деякі активи створюються за межами Unreal Engine (наприклад, в інших 3D-додатках, таких як Maya або 3ds Max). Інші активи, такі як плани, створюються безпосередньо в системі.

1.3 Інструменти та редактори

Unreal Engine надає комбінацію інструментів, редакторів і систем, які можна використовувати для створення гри чи додатку.

Інструмент — це те, що використовується для виконання певного завдання, наприклад розміщення Акторів на рівні або малювання місцевості.

Редактор — це набір інструментів, які використовуються для досягнення чогось більш складного. Наприклад, редактор рівнів дає змогу створювати рівні гри, або можна змінювати зовнішній вигляд матеріалів у редакторі матеріалів.

Система — це велика сукупність функцій, які разом створюють певний аспект гри чи додатку. Наприклад, Blueprint — це система, яка використовується для візуального сценарію елементів ігрового процесу.
Іноді системи та редактори можуть мати схожі назви. Наприклад, Редактор матеріалів використовується для редагування ресурсів Material, тоді як система Material забезпечує базову підтримку для використання Materials в Unreal Engine.

Деякі з цих інструментів і редакторів у Unreal Engine є вбудованими, тоді як

інші постачаються у формі додаткових плагінів, які можна ввімкнути або вимкнути залежно від потреб проекту.

1.3.1 Редактор рівнів (Level Editor)

Редактор рівнів — це основний редактор, у якому створюються рівні гри. Тут визначається ігровий простір для гри, додаванням різних типів Акторів і геометрії, візуальних сценаріїв Blueprints, Niagara тощо. За замовчуванням, коли створюється або відкривається проект, Unreal Engine відкриє редактор рівнів.

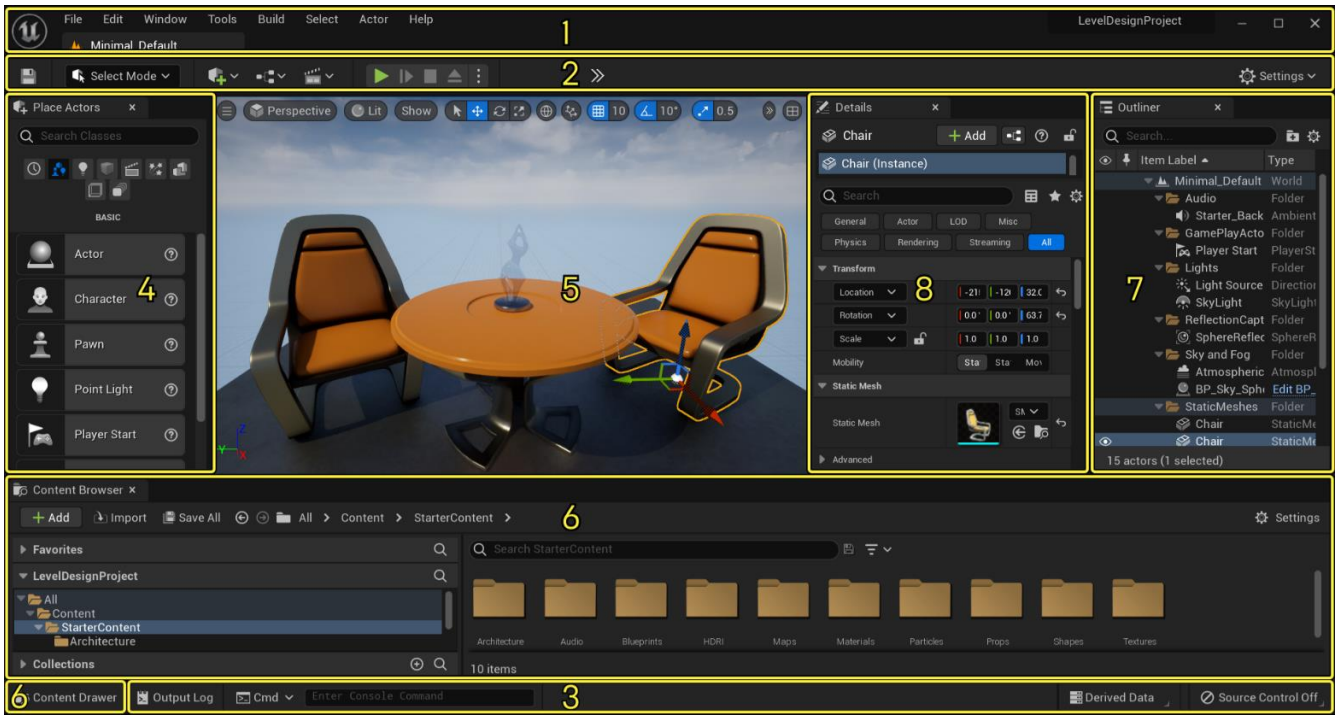

Рис.1.9 – Інтерфейс редактору рівнів за замовчуванням

1 - Панель вкладок і панель меню (Tab Bar and Menu Bar). Ця панель забезпечує доступ до загальних інструментів і команд, які використовуються під час роботи з рівнями в редакторі. Також на цій панелі відображається вкладка з назвою поточного рівня. Вкладки з інших вікон редактора можуть бути прикріплені поруч із цією вкладкою для швидкої та легкої навігації, подібно до веб-браузера. Праворуч від панелі вкладок знаходиться назва поточного проекту.

2 - Панель інструментів (Toolbar). На панелі інструментів відображається група команд, що забезпечує швидкий доступ до часто використовуваних інструментів і операцій.

3 - Нижня панель інструментів (Bottom Toolbar) - містить ярлики для командної консолі, журналу виводу та функцій похідних даних. Також відображає статус Source Control.

4 - Розміщення Акторів / Режими (Place Actor / Modes). Редактор рівнів можна перевести в різні режими редагування, щоб увімкнути спеціалізовані інтерфейси редагування та робочі процеси для редагування певних типів Акторів або геометрії.

Щоб відобразити вибір режимів, на панелі інструментів редактора рівнів треба відкрити спадне меню “Режими”.

Режими змінюють основну поведінку редактора рівнів для виконання спеціалізованих завдань, таких як переміщення та трансформація ресурсів у світі, ліплення ландшафтів, створення листя, створення геометричних пензлів і об’ємів, а також малювання на сітках. Панелі режимів містять набір інструментів, адаптованих до вибраного режиму редагування.

5 - Вікна перегляду (Viewports). Панель Viewport - це вікно у світи, які створюються в Unreal Engine. Ця панель містить набір вікон перегляду, кожне з яких можна розгорнути, щоб заповнити всю панель і надати можливість відображати світ в одному з трьох ортографічних видів (зверху, збоку, спереду) або в перспективі, що дає повний контроль над тим, що розробник бачить і як він це бачить.

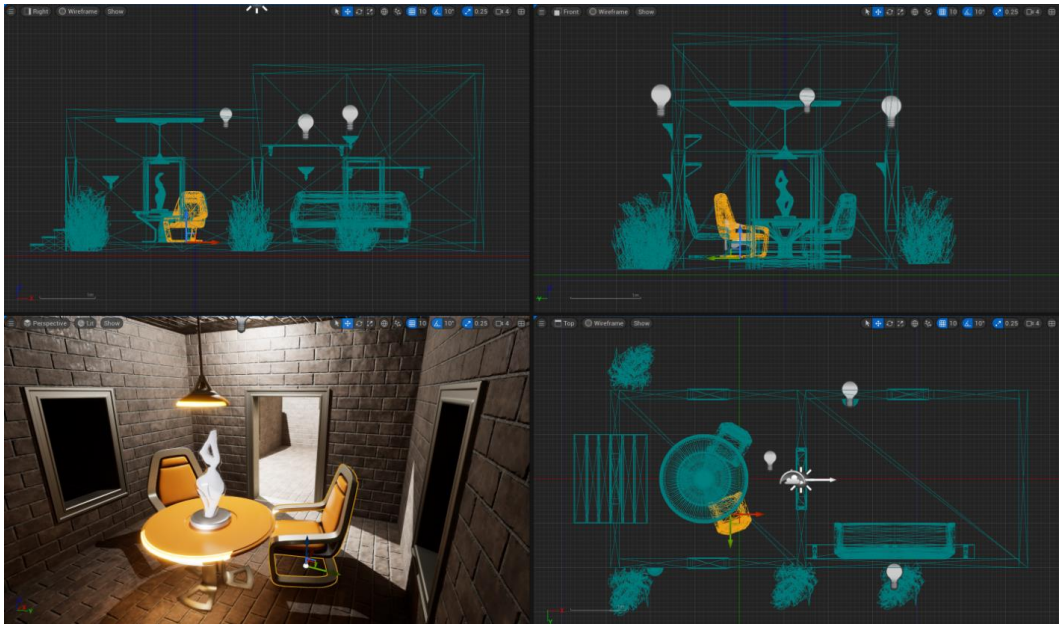

Рис.1.10– Панель Viewport

6 - Content Browser / Content Drawer. Content Browser є основною областю редактора Unreal для створення, імпорту, організації, перегляду та керування вмістом проекту Unreal. Також можна використовувати його для керування папками вмісту та виконання певних операцій з активами, наприклад:

- переглядати та взаємодіяти з усіма активами у проекті;
- знаходити об’єкти за допомогою текстового фільтра, який за бажанням можна поєднати з більш розширеною фільтрацією;
- упорядковувати ресурси в приватні, локальні або спільні колекції;
- визначати активи, які можуть містити проблеми;
- перенесити ресурси між папками вмісту або до іншого проекту.

Content Drawer - це особливий екземпляр Content Browser з дещо іншою поведінкою. Щоб відкрити його можна:

- натиснути кнопку Content Drawer на нижній панелі редактора;
- скористатися комбінацією клавіш Ctrl + пробіл (Windows) або Cmd + пробіл (macOS).

Content Drawer автоматично згортається, коли він втрачає фокус (тобто, коли відбувся клік поза ним). Щоб залишити його відкритим, треба натиснути кнопку «Закріпити в макеті». Це створює новий екземпляр Content Browser, але все ще можна відкрити новий Content Drawer.

7 - Outliner. На панелі Outliner відображаються всі Актори в межах сцени в ієрархічному перегляді дерева. Використовуючи Outliner, можна:

- вибирати і змінювати Акторів;
- шукати та фільтрувати Акторів за іменем, типом та іншими характеристиками;
- використовувати оператори розширеного пошуку для подальшого уточнення пошуку Акторів;
- налаштовувати, яку інформацію про Актора відображати.

Спадне меню «Інформація» використовується, щоб відобразити додатковий стовпець, який показує рівні, шари або назви ідентифікаторів.

8 - Деталі (Details). Панель «Деталі» містить інформацію, утиліти та функції для поточного виділення у вікні перегляду. Він містить поля редагування трансформації для переміщення, обертання та масштабування Акторів, відображає всі редаговані властивості для вибраних акторів і надає швидкий доступ до додаткових функцій редагування залежно від типу актора(ів), вибраного у вікні перегляду. Наприклад, вибрані Актори можна експортувати до FBX і конвертувати в інший сумісний тип. Деталі вибору дозволяють переглядати матеріали, використані вибраними Акторами, якщо такі є, і швидко відкривати їх для редагування.

1.3.2 Редактор статичної сітки (Static Mesh Editor)

Можна використовувати редактор статичної сітки для попереднього перегляду вигляду, зіткнення та UV-відображення, а також установлювати властивості статичних сіток і керувати ними. У редакторі статичної сітки також можна налаштувати LOD (або параметри рівня деталізації) для своїх ресурсів Static Mesh, щоб контролювати, наскільки вони прості чи деталізовані залежно від того, як і де працює гра.

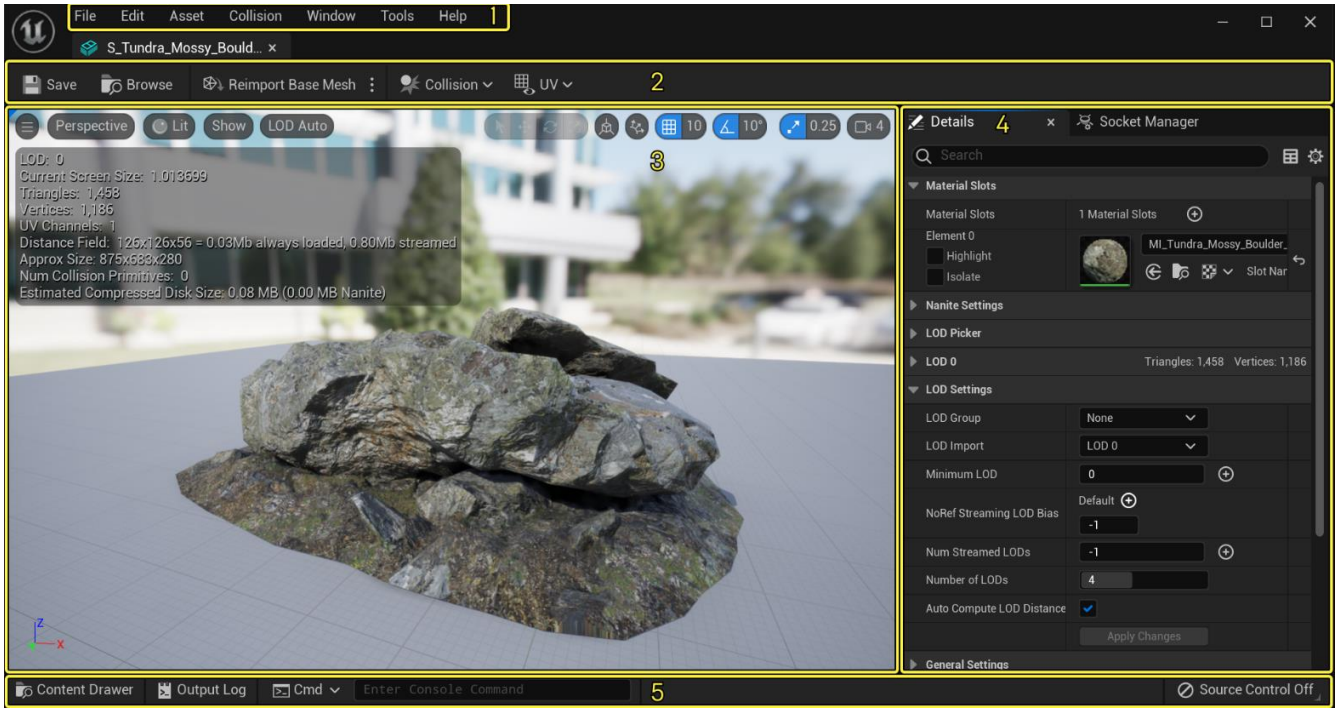

Рис.1.11 – Інтерфейс редактору статичної сітки

1 - Панель меню (Menu Bar) - забезпечує доступ до загальних інструментів і команд, які використовуються під час роботи в редакторі.

2 - Панель інструментів (Toolbar). На панелі інструментів відображається група команд, що забезпечує швидкий доступ до часто використовуваних інструментів і операцій.

3 - Панель Viewport. Панель Viewport показує рендеринг (або необов'язково каркасний) вид ресурсу Static Mesh. Це дає змогу перевіряти статичну сітку так, як вона відображатиметься в грі. Водночас Viewport містить інструменти та візуалізатори для перегляду певних даних, таких як:

- межі ресурсу Static Mesh;
- будь-яку колізійну сітку, назначену для статичної сітки;
- УФ-промені статичної сітки.

В панелі Viewport відображається набір статистичних даних або інформації про ресурс Static Mesh. У цій інформації є наступні дані:

- LOD: відображає кількість LOD (рівнів деталізації) для статичної сітки;
- поточний розмір екрану: вертикальна висота сітки, що відображається на екрані, як пропорція висоти вікна перегляду;
- трикутники: відображає кількість трикутників у статичній сітці;

- вершини: відображає кількість вершин у статичній сітці;
- УФ-канали: кількість УФ-каналів. Для відображення тіней потрібні унікальні УФ-промені, що не перекриваються;
- поле відстані: роздільна здатність і відбиток пам’яті поля відстані для цієї сітки;
- приблизний розмір: відображає приблизний розмір (довжина x ширина x висота) статичної сітки в одиницях Unreal зі шкалою 1 по всіх осях.
- кількість примітивів зіткнень.

4 - Панель деталей (Details Panel). Панель «Деталі» показує конкретні властивості, що стосуються актора статичної сітки, наприклад матеріали, застосовані до поверхні, параметри LOD і параметри зменшення сітки.

5 - Нижня панель інструментів (BottomToolbar). Містить ярлики для командної консолі, журналу виводу та функцій похідних даних. Також відображає статус Source Control

1.3.3 Редактор матеріалів (Material Editor)

Редактор матеріалів - це графовий інтерфейс на основі вузлів, який дозволяє створювати шейдери або матеріали для проекту. Тобто це місце, де створюються та редагуються матеріали. Матеріали - це ресурси, які можна застосувати до сітки, щоб контролювати її візуальний вигляд. Матеріали визначають зовнішній вигляд і властивості поверхні об’єктів у сцені, включаючи статичні та скелетні сітки, пейзажі, інтерфейс користувача та віртуальні ефекти. Наприклад, можна створити брудний матеріал і застосувати його до підлоги на даному рівні, щоб створити поверхню, яка виглядає так, ніби вона вкрита брудом.

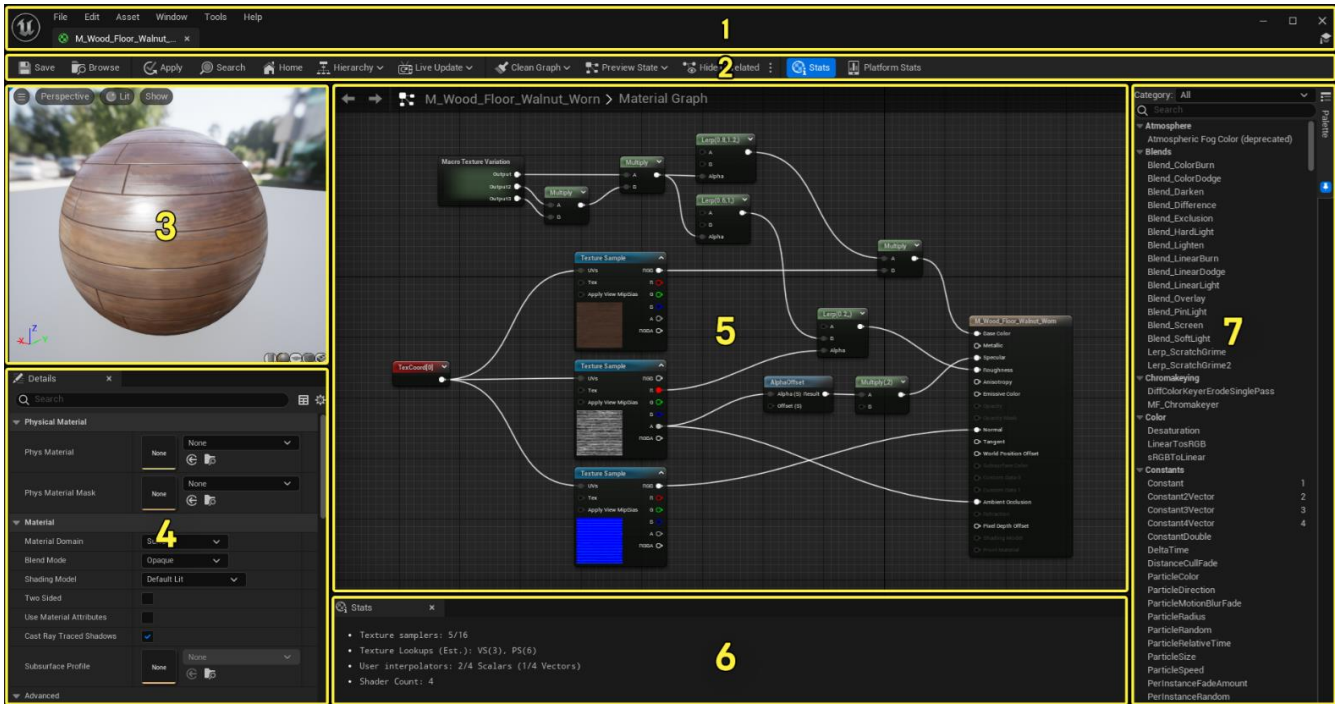

Рис.1.12 – Інтерфейс редактору матеріалів

1 - Панель меню (Menu Bar)

2- Панель інструментів (Toolbar)

Панелі меню та інструментів мають ті ж функції, що і в редакторі рівнів.

3 - Панель Viewport. На панелі Viewport відображається матеріал, який зараз редагується. Можна змінити сітку попереднього перегляду вікна перегляду за допомогою пов'язаних елементів керування панелі інструментів (п'ять кнопок форми у нижньому правому куті панелі Viewport). Щоб використати спеціальну сітку попереднього перегляду, треба вибрати Актора Static Mesh у Content Browser та клацнути піктограму цеглинки у вікні перегляду. Сітка вікна перегляду зберігається разом із матеріалом, щоб наступного разу, коли матеріал відкривався в редакторі матеріалів, він відображав ту саму сітку.

4 - Панель деталей (Details Panel). Ця панель містить вікно властивостей для всіх вибраних виразів матеріалу та вузлів функцій. Якщо вузли не вибрано, відображаються основні властивості матеріалу.

5 - Панель графів матеріалів (Material Graph Panel). Ця панель містить граф усіх виразів Матеріалу, які належать до цього Матеріалу. За замовчуванням кожен матеріал містить один базовий вузол матеріалу. Цей вузол має ряд входів, кожен з

яких пов’язаний з окремим аспектом Матеріалу, до якого можуть підключитися інші вузли Матеріалу.

6 - Панель статистики (Stats Panel). На цій панелі відображається кількість інструкцій шейдерів, які використовуються в матеріалі, а також будь-які помилки компілятора. Чим менше інструкцій, тим дешевший матеріал. Вузли вираження матеріалу, які не підключені до вузла базового матеріалу, не вносять вклад у кількість інструкцій (вартість) матеріалу.

7 - Панель палітри (Palette Panel). Панель «Палітра» містить упорядкований за категоріями список усіх вузлів матеріалу, які можна перетягнути у даний матеріал. Щоб розмістити новий вузол Material, треба перетягнути його з палітри на панель графів матеріалів. За замовчуванням палітра прихована. Треба натиснути вкладку «Палітра» у верхньому правому куті редактора матеріалів, щоб відобразити палітру. Треба клацнути піктограму Pin, щоб палітра залишалася видимою під час роботи.

Загалом для створення простого матеріалу необхідно виконати такі кроки:

1) створити новий актив Матеріалу у Content Browser;

2) відкрити редактор матеріалів;

3) розмістити матеріальні вирази та функції в графі матеріалів;

4) підключити вирази матеріалу до входів у вузлі основного матеріалу, щоб визначити атрибути матеріалу;

5) зібрати і зберегти матеріал;

6) застосувати матеріал до об’єктів на рівні.

Основна частина процесу створення матеріалу відбувається під час третього та четвертого кроків, і саме тут виконується більша частина роботи, пов’язаної з визначенням конкретного типу поверхні та додаванням творчих штрихів. Матеріальні вирази та функції є будівельними блоками Unreal Materials, кожна з яких виконує певну дію на графі матеріалів. Комбінуючи ці вузли унікальними способами, можна визначити нескінченну низку фізичних поверхонь.

Вносячи зміни в мережу Material, корисно бачити миттєвий відгук про кожну зміну в реальному часі. Редактор матеріалів пропонує три параметри в

меню оновлення в реальному часі, щоб контролювати, які елементи в редакторі матеріалів оновлюються в реальному часі - попередній перегляд матеріалу, вузли в реальному часі та попередній перегляд усіх вузлів.

Кожна з цих опцій дає можливість перемикати миттєвий відгук для іншої частини редактора матеріалів.

Попередній перегляд матеріалу – цей параметр дозволяє автоматично оновлювати будь-які зміни у вікні Viewport в реальному часі без використання кнопок «Зберегти» або «Застосувати».

Вузли в реальному часі – вузли, на які впливають функції, засновані на часі, такі як панорамування, оновлюватимуться в реальному часі.

Попередній перегляд усіх вузлів – цей параметр дозволяє перекомпілювати шейдер для кожного вузла в мережі, коли вносяться зміни. Ці зміни включають створення нових вузлів, видалення вузлів, підключення та відключення вузлів, а також зміни властивостей. Ця повторна компіляція необхідна, щоб попередній перегляд матеріалу, намальований у цьому вузлі, був актуальним. Однак перекомпіляція цих проміжних шейдерів може зайняти багато часу, особливо якщо матеріал містить велику мережу

1.3.4 Редактор планів (Blueprint Editor)

Редактор планів — це вузловий редактор графів. Це основний інструмент для створення та редагування мереж вузлів візуальних сценаріїв, які зазвичай називають планами (Blueprints). Іншими словами - це місце, де можна працювати з планами та змінювати їх. Це спеціальні ресурси, які використовуються для створення елементів ігрового процесу (наприклад, керування Актором або створення сценарію події), модифікації матеріалів або реалізації інших функцій Unreal Engine без необхідності писати код C++.

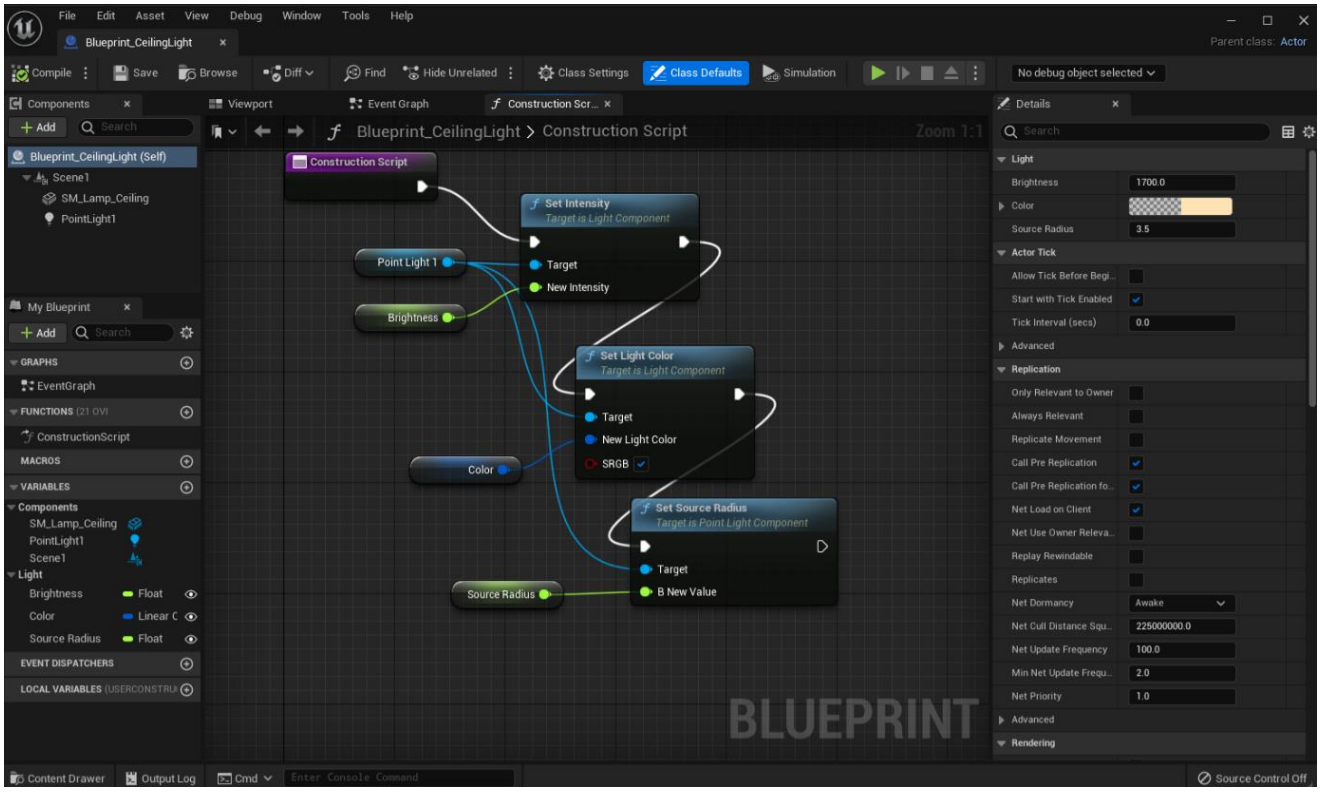

Рис.1.13– Інтерфейс редактору планів

Blueprints можуть управляти подіями на основі рівнів, керувати внутрішнім сценарієм поведінки Акторів у грі та використовуватися для керування складними анімаціями в дуже реалістичних системах ігрових персонажів. Для кожного з цих додатків Blueprint, місця, де редагується цей сценарій Blueprint, доступні інструменти, які змінюватимуться залежно від потреб. Це означає, що в Unreal Engine є кілька місць і різні способи появи редактора Blueprint. Однак, незважаючи на відмінності, Blueprint Editor може створювати та редагувати потужні візуальні сценарії для керування різними аспектами гри.

Редактор Blueprint містить контекстно-залежний дизайн, який допомагає отримати доступ до функціональних можливостей для об’єктів, які потрібні саме тоді, коли вони потрібні, а також забезпечує гнучкість у тих випадках, коли потрібно зробити щось трохи нетрадиційне.

Редактор Blueprint містить кілька інструментів і панелей, які допомагають створювати власні змінні, функції, масиви тощо. Він має різноманітні інструменти для налагодження та аналізу, вбудовані в нього, щоб допомогти швидко налагодити та покращити потік даних у проектованих мережах. У Unreal Engine

редактор Blueprint представлений у різних спеціалізованих формах залежно від типів мереж Blueprint, які редагуються

1.3.5 Blueprint Editor Level Blueprint UI

План рівня — це спеціалізований тип плану, який діє як глобальний граф подій рівня. Кожен рівень у проекті має власний проект рівня, створений за замовчуванням, який можна редагувати в редакторі Unreal, однак нові плани рівня неможливо створити через інтерфейс редактора.

Події, що стосуються рівня в цілому, або певних екземплярів Акторів на рівні, використовуються для запуску послідовностей дій у формі викликів функцій або операцій керування потоком.

Плани рівнів також надають механізм керування для потокової передачі рівня та секвенсора (Sequencer), а також для прив'язки подій до акторів, розміщених на рівні.

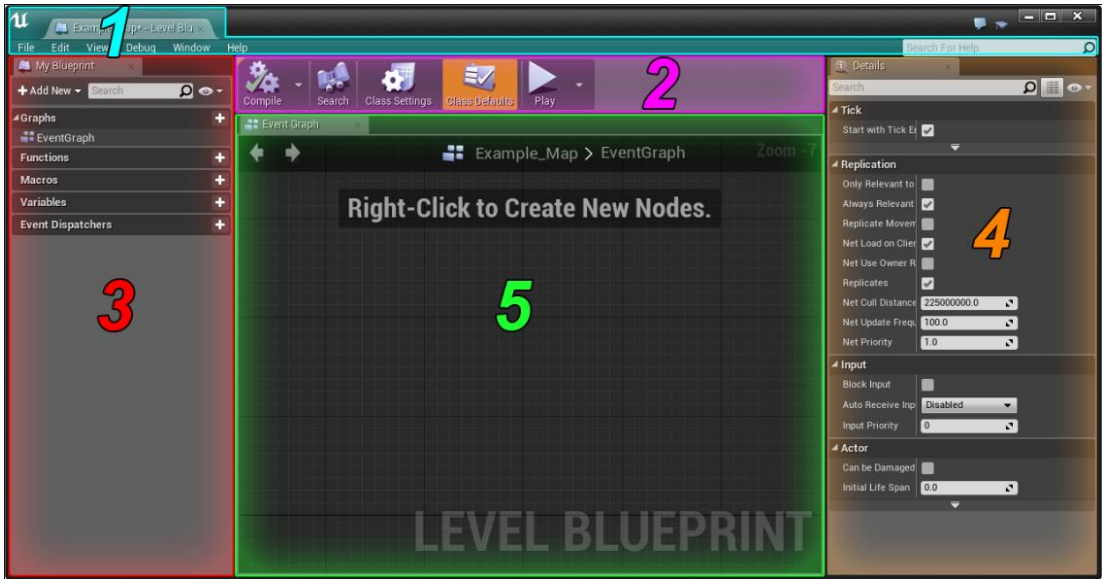

Рис.1.14 – Інтерфейс Blueprint Editor Level Blueprint UI за замовчуванням

1- Меню (Menu), 2 - Панель інструментів (Toolbar) - мають ті ж функції, що і в редакторі рівнів.

3 - Мій план (My Blueprint). На панелі My Blueprint відображається деревовидний список графів, сценаріїв, функцій, макросів тощо, що містяться в Blueprint. По суті, це схема плану, яка дозволяє легко переглядати наявні елементи плану, а також створювати нові. Різні типи планів матимуть різні типи елементів, які відображатимуться в деревному списку вкладки «My Blueprint». 4 - Деталі

(Details). Панель «Деталі» - це контекстно-залежна область, яка надає доступ для редагування властивостей вибраних елементів у редакторі планів. Він складається з панелі пошуку для швидкого доступу до певних властивостей і, як правило, містить одну або кілька категорій, що згортаються, для впорядкування включених властивостей.

На панелі «Деталі» також можна виконувати багато завдань редагування проекту, зокрема:

- процес редагування змінних Blueprint, включаючи зміну імені, типу та того, чи є змінна масивом;
- реалізовувати інтерфейси Blueprint після натискання кнопки Blueprint Props;
- додавання входів і виходів для функцій Blueprint;
- додавання події для вибраних компонентів.

5 - Редактор графів (Graph Editor). Панель редактора графів є серцем системи Blueprint. Саме тут створюються мережі вузлів і зв'язків, які визначатимуть поведінку за сценарієм. Вузли можна швидко вибрати, натиснувши на них, і змінити їх положення за допомогою перетягування.

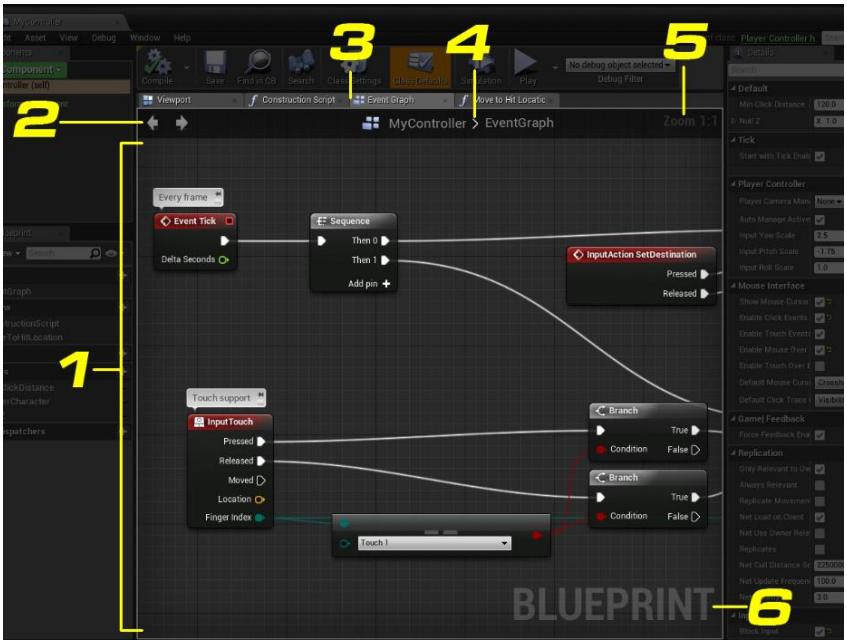

Рис.1.15 – Панель редактора графів

1 - Область графа – тут фактично розміщуються вузли;

2 - Кнопки «Вперед» і «Назад» – вони дають змогу переходити між різними графами так само, як і у веб-браузері;

3 - Область вкладок. Коли відкриваються різні графи, тут відкриваються вкладки для кожного з них, що дозволяє швидко переходити між різними графами;
4 - Breadcrumbs – показують послідовність графів і підграфів. При переході до функцій або згорнутих графів, це покаже, місце знаходження у мережі;
5 - Коефіцієнт масштабування (Zoom Factor) – просто показує поточний коефіцієнт масштабування в редакторі графів;
6 - Позначка плану (Blueprint label) – це показує тип плану, який редагується. При редагуванні інтерфейсів плану, план анімації, макроси та інші типи, ця позначка оновлюватиметься.

1.3.6 Blueprint Interface Editor UI

Інтерфейс Blueprint — це сукупність однієї або кількох функцій (тільки назва, без реалізації), які можна додати до інших Blueprints. Будь-який Blueprint, до якого додано інтерфейс, гарантовано матиме ці функції. Функціям інтерфейсу можна надати функціональність у кожному з Blueprints, який його додав. Це, по суті, схоже на концепцію інтерфейсу в загальному програмуванні, яка дозволяє використовувати кілька різних типів об’єктів, щоб усі вони мали спільний доступ і доступ до них через загальний інтерфейс. Простіше кажучи, інтерфейси Blueprint дозволяють різним Blueprint обмінюватися та надсилати дані один одному.

Інтерфейси Blueprint можуть створюватися творцями вмісту за допомогою редактора подібно до інших Blueprints, але вони мають певні обмеження, а саме:
- додавання нових змінних;
- редагування графів;
- додавання компонентів.

Blueprint Interface Editor UI за замовчування має ті ж панелі з такими ж функціями, що й Blueprint Editor Level Blueprint UI.

1.3.7 Blueprint Editor Macro Library UI

Бібліотека макросів Blueprint - це контейнер, який містить колекцію макросів або самодостатніх графів, які можна розмістити як вузли в інших Blueprints. Це може заощадити час, оскільки вони можуть зберігати типові послідовності вузлів разом із входами та виходами як для виконання, так і для передачі даних.

Макроси використовуються спільно для всіх графів, які на них посилаються, але вони автоматично розгортаються в графи, ніби вони були згорнутим вузлом під час компіляції. Це означає, що бібліотеки макросів Blueprint не потрібно компілювати. Однак зміни в макросі відображаються лише в графах, які посилаються на цей макрос, коли Blueprint, що містить ці графи, перекомпілюється.

Blueprint Editor Macro Library UI за замовчування має ті ж панелі з такими ж функціями, що й Blueprint Editor Level Blueprint UI

1.3.8 Редактор анімаційних схем (Animation Blueprint Editor)

Редактор анімаційних схем має подібні функції до редактора планів, але містить інші функції, інструменти та вікна, які допомагають створювати сценарії анімації персонажів.

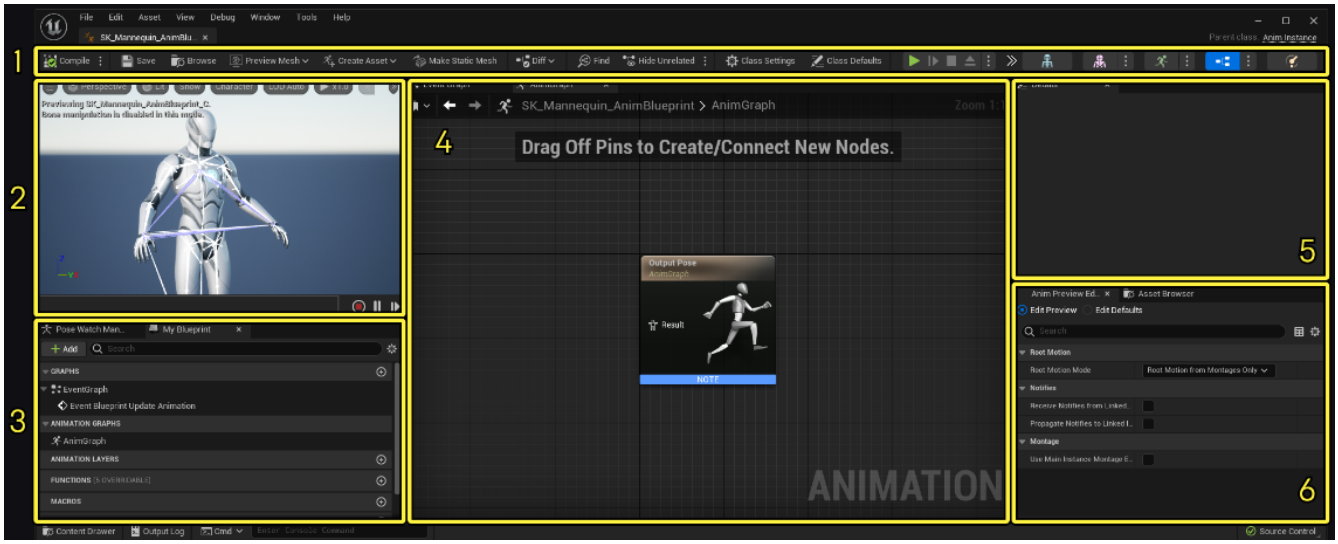

Рис.1.16 – Інтерфейс Animation Blueprint Editor за замовчуванням

1 - Панель інструментів (Toolbar), яка містить кнопки для керування схемою анімації та перемикання типу редактора. Саме за допомогою цій панелі компілюються схеми, зберігаються, знаходиться актив Animation Blueprint у Content Browser та визначаються налаштування класу та параметри класу за замовчуванням.

2 - Вікно перегляду (Viewport), де можна попередньо переглянути поведінку логіки анімаційного плану на вказаному персонажі.

3 - My Blueprint, який так само можна знайти в Blueprint Editor, містить

список графів, функцій, змінних та інших пов’язаних властивостей у Animation Blueprint.

4 - Граф (Graph), який відображає різні графи для візуальних сценаріїв у схемі анімації. Саме тут створюється логіка, яка керує персонажем під час гри. Існує три основних типи графів, кожен з яких має різні інтерфейси:

а) Граф подій (Event Graph), де створюється логіка на основі Blueprint для визначення властивостей вузла та змінних, які інформують інші області графа.

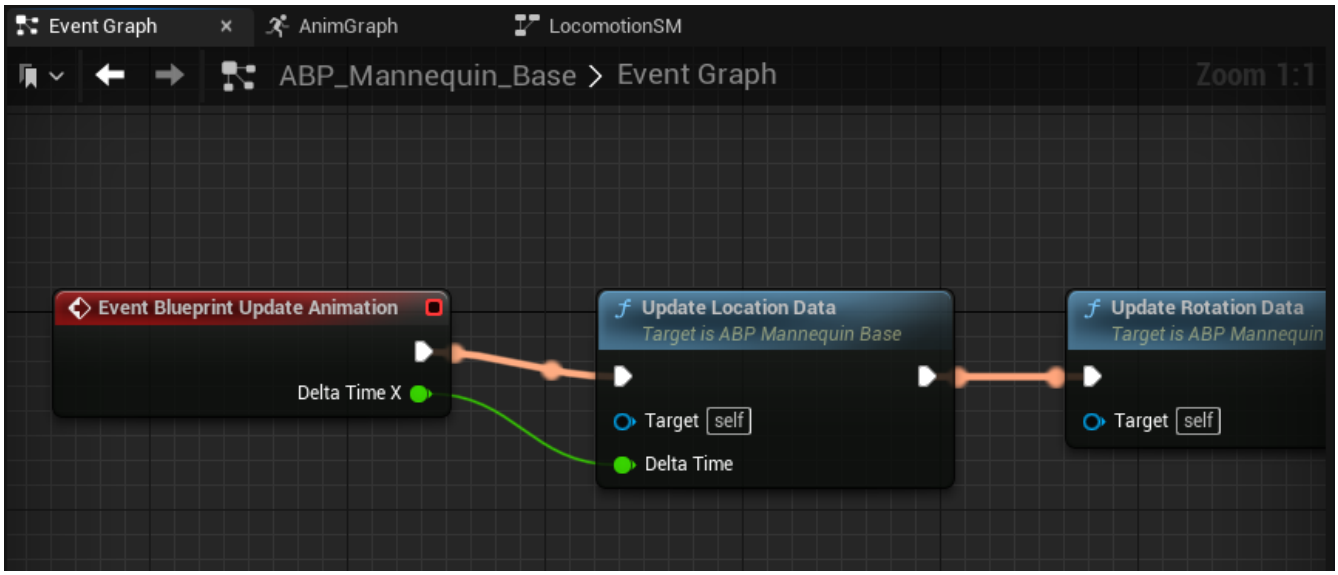

Рисунок 1.17 – Інтерфейс графу подій

б) Anim Graph, де створюється логіка на основі пози, яка оцінює остаточну позу Skeletal Mesh для поточного кадру.

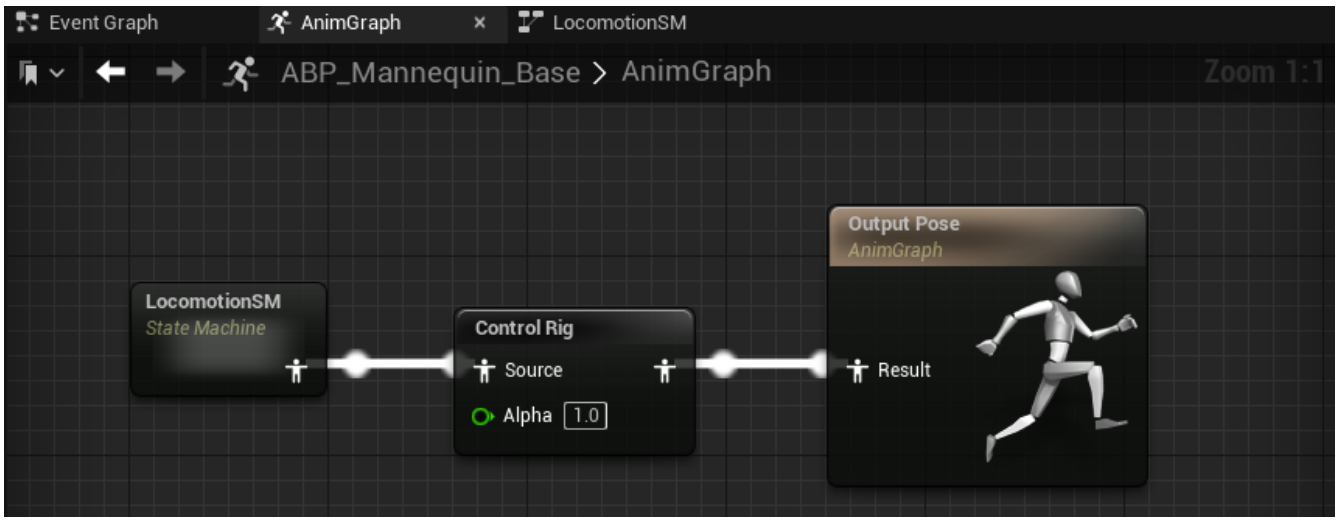

Рис.1.18 – Інтерфейс графу Anim

в) Конечні машини (State Machines), де створюється логіка на основі стану, яка зазвичай використовується для пересування. У першу чергу цей тип системи

використовується для співвіднесення анімації зі станами руху персонажів, такими як бездіяльність, ходьба, біг і стрибки. За допомогою State Machines створюються стани, визначається анімація для відтворення в цих станах і створюються різні типи переходів, щоб контролювати час переходу в інші стани. Це полегшує створення складного змішування анімації без використання надто складного Anim Graph.

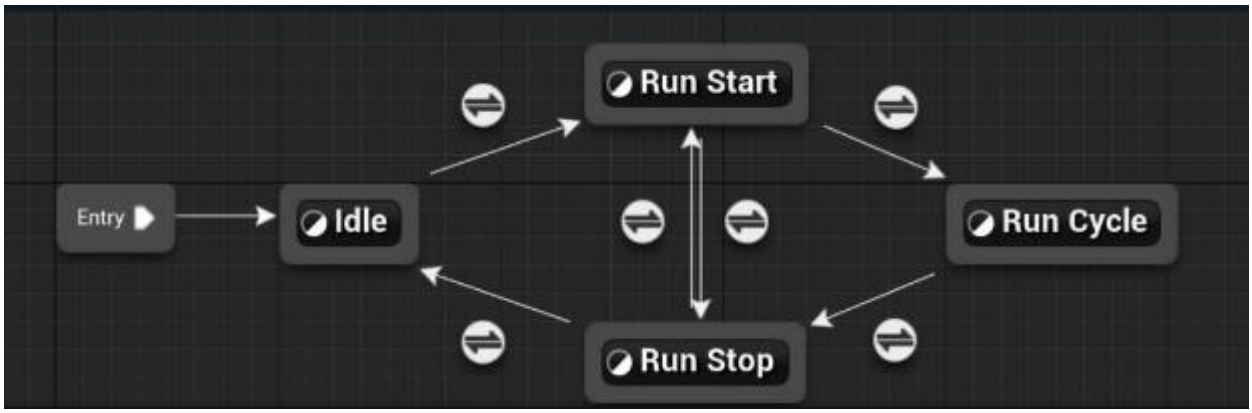

Рис.1.19 – Інтерфейс State Machines

5 - Деталі (Details), що відображає властивості вибраного елемента.

6 - Редактор попереднього перегляду Anim (Anim Preview Editor), у якому можна змінювати змінні та значення класу за замовчуванням. На окремій вкладці розміщено браузер активів, де можна переглядати та відкривати анімаційні активи, пов’язані з цим Скелетом (Skeleton).

1.3.9 Редактор активів фізики (Physics Asset Editor)

Physics Asset Editor використовується для створення активів фізики для використання зі скелетними сітками. На практиці це те, як реалізуються такі фізичні функції, як деформації та зіткнення. Вони містять набір твердих тіл і обмежень, які складають одну регдолл (ragdoll), яка не обмежується гуманоїдними регдоллами. Їх можна використовувати для будь-якої фізичної симуляції з використанням тіл і обмежень. Оскільки для Skeletal Mesh дозволено лише один фізичний ресурс, їх можна вмикати чи вимикати для багатьох компонентів Skeletal Mesh. Можна почати з нуля і створити повну установку Ragdoll або скористатися інструментами автоматизації для створення базового набору фізичних тіл і фізичних обмежень.

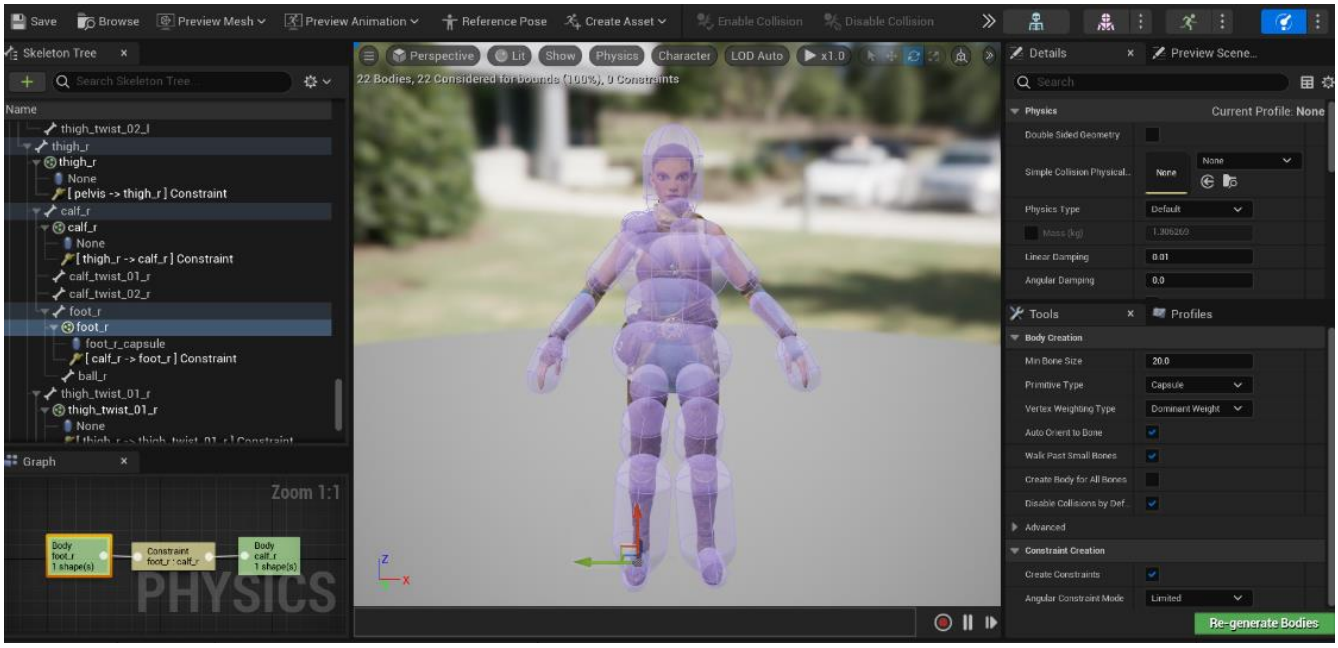

Рис.1.20 – Інтерфейс Physics Asset Editor

1.3.10 Редактор дерева поведінки (Behavior Tree Editor)

Редактор дерева поведінки — це місце, де створюються сценарії штучного інтелекту (ШІ) через систему на основі візуальних вузлів (схожу на Blueprints) для акторів на рівнях. Можна створити будь-яку кількість різних моделей поведінки для ворогів, неігрових персонажів (NPC), транспортних засобів тощо.

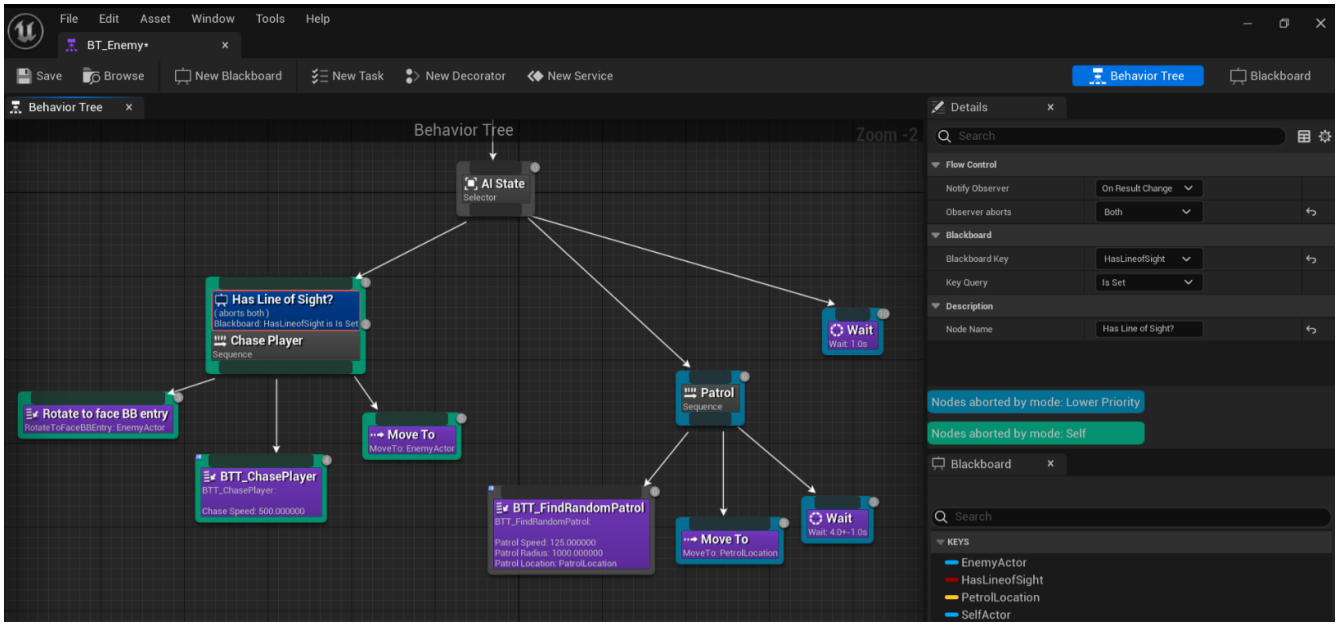

Рис.1.21 – Інтерфейс редактора дерева поведінки

Дерева поведінки створюються візуально, подібно до Blueprint, шляхом додавання та з’єднання серії вузлів, які мають певну функціональність, пов’язану з ними, до графа дерева поведінки. У той час як Дерево поведінки виконує логіку,

окремий ресурс, який називається Blackboard, використовується для зберігання інформації (так звані клавіші Blackboard), про яку дерево поведінки має знати, щоб приймати обґрунтовані рішення. Типовий робочий процес полягав би у створенні Blackboard, додаванні кількох клавіш Blackboard, а потім створенні дерева поведінки, яке використовує ресурс Blackboard.

1.3.11 Редактор скелетів (Skeleton Editor)

У Unreal Engine активи скелетів є основою для всієї анімаційної роботи зі сітками скелетів. Режим Skeleton Editor — це візуальний редактор, у якому можна знайти інструменти та властивості для внесення змін до активів скелетів.

У цьому редакторі можна маніпулювати окремими кістками та кістковими структурами, приєднувати скелетні сітчасті гнізда та переглядати будь-які криві анімації та повідомлення анімації, пов’язані з скелетом. У Skeleton Editor також можна знайти Retargeting Manager, інструмент для керування сітками, пов’язаними з поточним Skeleton Asset.

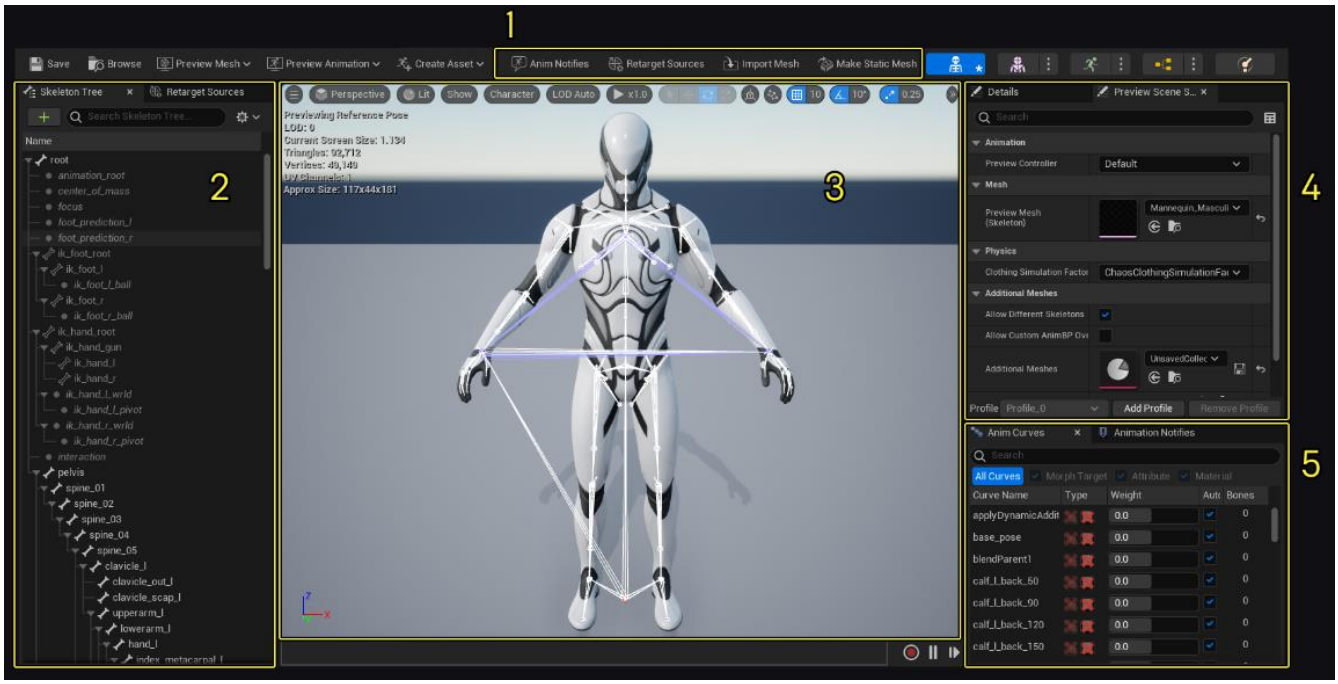

Рис.1.22 – Інтерфейс редактора скелетів

1 - Панель інструментів (Toolbar). Панель інструментів подібна до панелі інструментів у інших редакторах і вікнах Unreal Engine.

2 - Дерево скелетів / Перенацілення джерел (Skeleton Tree / Retarget Sources). На вкладці «Дерево скелетів» відображається скелетна ієрархія поточного ресурсу

скелета, що дозволяє створювати та редагувати скелетні сокети та визначати параметри, пов’язані з перенацілюванням анімації.

Інструмент Retarget Sources на панелі інструментів відкриє панель, яка зберігає та керує списком джерел перенацілювання. Джерела перенацілювання використовуються для імпорту анімацій авторських персонажів із різними пропорціями, метод називається перенацілюванням анімації.

3 - Панель Viewport. Панель Viewport дозволяє попередньо переглядати відтворення анімаційних ресурсів у вибраній скелетній сітці та надає інформацію про активи.

4 - Попередній перегляд налаштувань сцени (Preview Scene Settings / Details). На цій панелі розташовано вкладку для налаштувань сцени попереднього перегляду, за допомогою якої можна керувати параметрами попереднього перегляду, такими як вибрана анімація, застосовані скелетні сітки, освітлення вікна перегляду та налаштування постобробки.

Вкладка «Деталі» в основному використовується для зміни властивостей елементів, наприклад кісток.

5 - Криві анімації / повідомлення анімації (Anim Curves / Animation Notifies). За допомогою панелі «Криві анімації» можна створювати та керувати значеннями кривих для цілей трансформації, а також кривими атрибутів і матеріалів для вибраного скелета.

Повідомлення анімації надають програмістам анімації спосіб налаштувати події, які відбуватимуться в певні моменти під час послідовності анімації. Сповіщення зазвичай використовуються для додавання ефектів до анімації, наприклад звуків кроків під час прогулянки

1.3.12 Редактор послідовності анімації (Animation Sequence Editor)

Редактор послідовності анімації надає доступ до різноманітних орієнтованих на анімацію ресурсів, доступних для Skeletal Meshes у Unreal Engine. У редакторі послідовності анімації можна редагувати та переглядати послідовності анімації, монтажі, криві тощо.

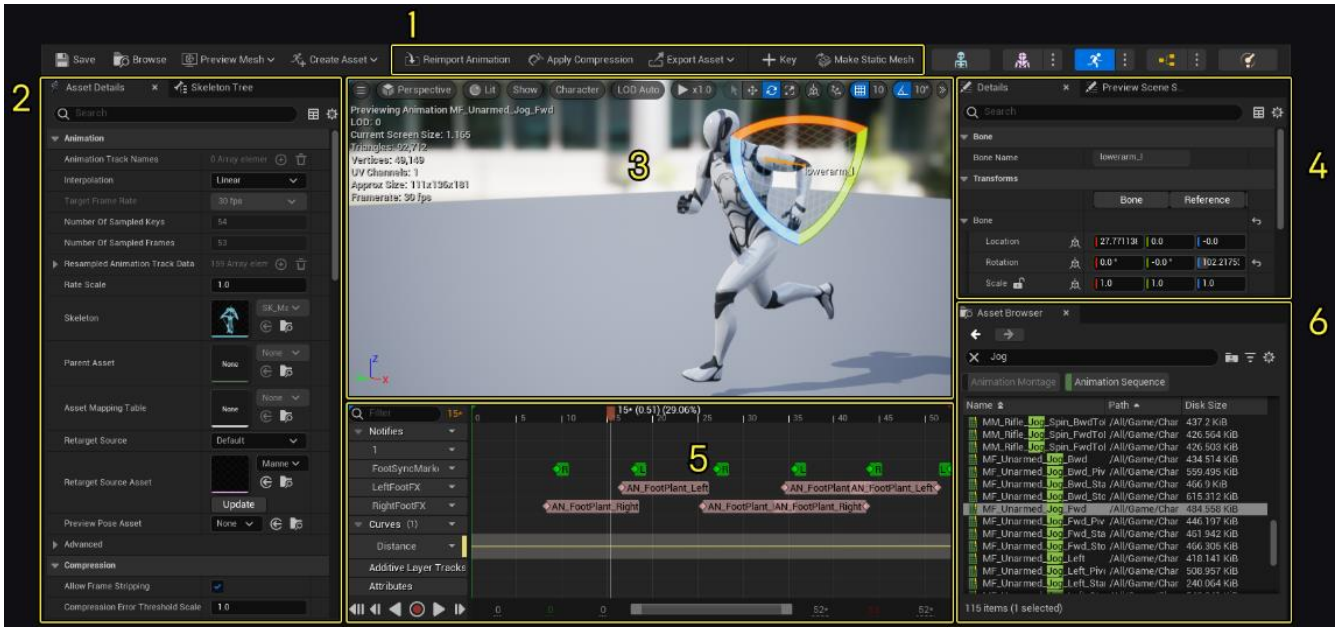

Рис.1.23 – Інтерфейс редактора послідовності анімації

1 - Панель інструментів (Toolbar)

2 - Відомості про активи / Дерево скелетів (Asset Details / Skeleton Tree). Панель «Asset Details» — це контекстно-залежний редактор властивостей, у якому редагуються та змінюються налаштування, що стосуються вибраної послідовності анімації та її активів, таких як простори змішування, монтажі анімації та сповіщення про анімацію.

3 - Панель Viewport

4 - Деталі / Попередній перегляд налаштувань сцени (Details / Preview Scene Settings)

5 - Редактор активів (Asset Editor) - це контекстне вікно, яке змінює свій інтерфейс і функції залежно від типу відкритого активу анімації. Можна відтворювати та змінювати Blend Spaces, Animation Montages тощо.

6 - Браузер активів (Asset Browser). Подібно до Content Browser, у браузері активів можна переглядати та фільтрувати анімаційні ресурси, пов’язані з поточним активом Skeleton.

Функції панелі інструментів, дерева скелетів, панелі Viewport, панелі Деталі та Попередній перегляд налаштувань сцени такі ж як і в редакторі скелетів

1.3.13 Редактор звукових сигналів (Sound Cue Editor)

Тут редагуються Sound Cues, в яких визначається поведінка відтворення аудіо в Unreal Engine 5. У цьому редакторі можна комбінувати та мікшувати кілька звукових ресурсів, щоб отримати єдиний мікшований вихід, збережений як Sound Cue.

Редактор Sound Cue — це редактор на основі вузлів, який використовується для розробки ресурсів Sound Cue. Sound Cue — це актив, який діє як контейнер для інформації про поведінку звуку. Звукові сигнали складаються із звукових вузлів, які є окремими модулями, кожен з яких впливає на дизайн звуку. Звукові вузли впорядковані у вигляді графа, який відображає зв'язки між звуковими вузлами та потік даних між ними. Звукові підказки містять граф звукового вузла, який може зберігати посилання на імпортовані аудіоресурси, відомі як звукові хвилі, або містити інструкції про те, як маніпулювати аудіо, коли воно проходить через граф.

Рис.1.24 – Інтерфейс Sound Cue Editor

1.3.14 nDisplay 3D Config Editor

nDisplay рендерить сцену Unreal Engine на кількох синхронізованих відображувальних пристроях. За допомогою редактора конфігурації nDisplay можна створити налаштування nDisplay і візуалізувати, який вміст буде відображатися на всіх пристроях відображення.

Більшість аспектів системи nDisplay визначається за допомогою єдиного активу конфігурації - nDisplay Config Asset, через nDisplay 3D Config Editor. Цей ресурс визначає комп'ютери, які складають кластерну мережу, характеристики вікон і вікон перегляду, які Unreal Engine має відобразити на кожному комп'ютері, топологію та конфігурацію пристрою відображення, частини віртуального світу, які має відображати кожне вікно перегляду, типи пристроїв введення, які будуть прийняті, тощо.

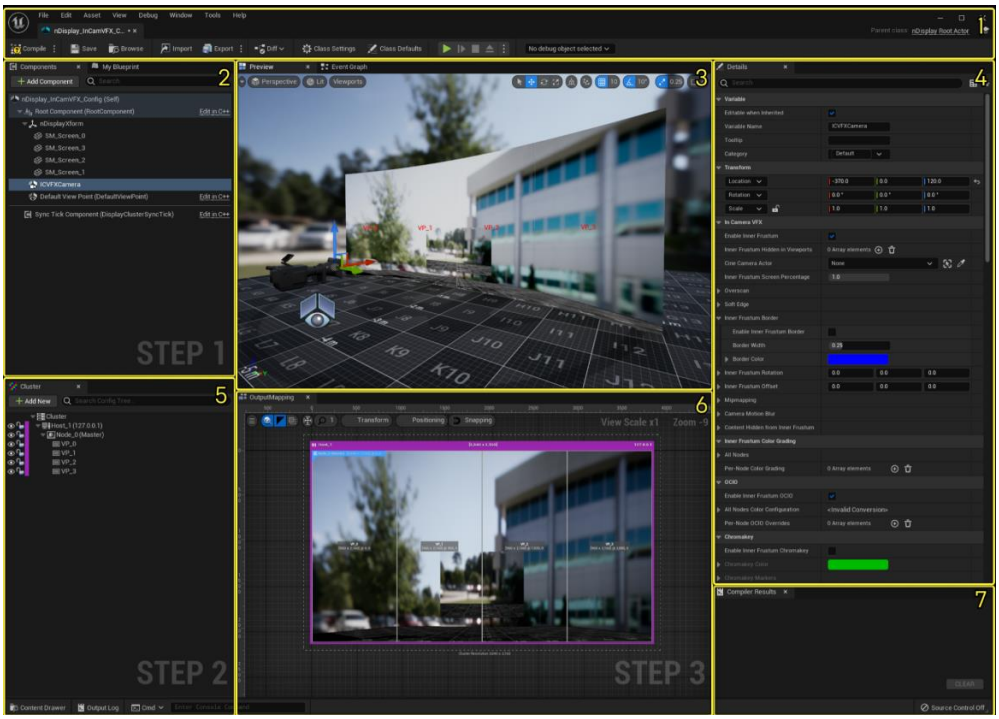

Рис.1.25 – Інтерфейс nDisplay 3D Config Editor

1 - Панель інструментів (Toolbar). Панель інструментів має більшість кнопок, що й панель інструментів у редакторі планів. Додатково є дві унікальні кнопки для nDisplay 3D Config Editor:

- Імпорт: імпортує файл конфігурації nDisplay (.ndisplay, .cfg) із локального комп'ютера;

- Експорт: експортує налаштування nDisplay у файл конфігурації (.ndisplay) на локальному комп'ютері .

2 - Компоненти (Components). Панель «Компоненти» визначає фізичний дисплей, відстеження та налаштування в камері для кластера nDisplay. Додавання компонентів до успадкованого кореневого компонента є першим кроком у проектуванні мережі nDisplay. Можна вибрати з попередньо визначеного списку

дисплеїв, трансформацій і акторів камери за замовчуванням або додати будь-які доступні компоненти UE.

3 - Попередній перегляд (Preview)

4 - Деталі (Details)

5 - Кластер (Cluster). Для кожного окремого екземпляра проектованого додатку Unreal Engine, який використовуватиметься у мережі nDisplay, потрібно визначити вузол кластера. Кожен вузол кластера є представленням екземпляра додатку та визначає ім'я хоста або IP-адресу комп'ютера, на якому запускатиметься цей екземпляр додатку. Можна налаштувати інший фізичний комп'ютер для кожного вузла кластера або мати кілька вузлів кластера, які працюють на одному хості. Кожен вузол кластера містить одне або більше вікон перегляду. Поряд із пов'язаним дисплеєм або статичними сітчастими компонентами вони визначають вікно у 3D-світі. Потім це вікно використовується для візуалізації сцени з точки зору будь-якого View Origin.

6 - Вихідне відображення (Output Mapping). Панель Output Mapping ефективно відображає вікна перегляду, визначені на панелі Cluster, у вікна програми 2D UE. На панелі Output Mapping можна:

- переглядати зв'язок між головним комп'ютером, вузлом кластера (екземпляр додатка UE) і вікном перегляду;
- візуалізувати, редагувати та відображати 2D вікна перегляду у вікні екземпляра додатку;
- перекладати і повертати вікна перегляду.

7 - Результати компілятора (Compiler Results).

1.3.15 Інші редактори

Редактор Niagara (Niagara Editor) - призначений для створення спеціальних ефектів шляхом використання повністю модульної системи ефектів частинок, що складається з окремих випромінювачів частинок для кожного ефекту. Випромінювачі можна зберегти в Content Browser для подальшого використання та служити основою для нових випромінювачів у поточних або майбутніх проектах.

Редактор інтерфейсу UMG (UMG UI Editor) - це інструмент створення візуального інтерфейсу користувача, який можна використовувати для створення елементів інтерфейсу користувача, таких як ігрові дисплеї, меню або інша графіка, пов’язана з інтерфейсом.

Редактор шрифтів (Font Editor) - використовується, щоб додавати, упорядковувати та переглядати ресурси шрифтів. Також можна визначити параметри шрифту, такі як компонування ресурсу шрифту та політику підказок (підказка шрифту — це математичний метод, який гарантує, що текст буде читабельним за будь-якого розміру дисплея).

Редактор Sequencer - дає можливість створювати кінематографічні зображення в грі за допомогою спеціального багатодоріжкового редактора. Створюючи послідовності рівнів і додаючи треки, можна визначити склад кожного треку, який визначатиме вміст сцени. Доріжки можуть складатися з таких речей, як анімація (для анімації персонажа), трансформації (переміщення предметів у сцені), аудіо (для включення музики чи звукових ефектів) тощо.

Редактор скелетних сіток (Skeletal Mesh Editor) — це місце, де можна редагувати полігональні сітки, призначати матеріали, додавати елементи одягу, регулювати рівень деталізації (LOD) і перевіряти функціональність Morph Target.

Редактор Control Rig — це набір інструментів для анімації, за допомогою яких можна монтувати та анімувати персонажів безпосередньо в системі. Використовуючи Control Rig, можна обійти потребу в установці й анімації за допомогою зовнішніх інструментів, а натомість анімувати безпосередньо в Unreal Editor. За допомогою цієї системи можна створювати та налаштовувати власні елементи керування для персонажа, анімувати в Sequencer і використовувати різноманітні інші інструменти анімації, щоб допомогти процесу анімації.

Редактор медіа (Media Editor) - використовується, щоб визначити медіа-файли або URL-адреси, які будуть як джерело медіа для відтворення в Unreal Engine. Мож визначити параметри відтворення вихідного медіафайлу, наприклад автоматичне відтворення, швидкість відтворення та повторення, але можна редагувати медіафайл безпосередньо.

Редактор бібліотеки DMX (DMX Library Editor). DMX (Digital Multiplex) - це стандарт цифрового зв’язку, який використовується в індустрії живих подій для керування різними пристроями, такими як освітлювальні прилади, лазери, димові машини, механічні пристрої та електронні рекламні щити. У редакторі бібліотеки DMX можна налаштувати ці пристрої та їхні команди

РОЗДІЛ 2. СПОСОБИ МОДЕЛЮВАННЯ В ПРОГРАМНОМУ СЕРЕДОВИЩІ UNREAL ENGINE 5

2.1 Стаціонарний режим моделювання в Unreal Engine 5

Режим редактора моделювання, також відомий як режим моделювання, надає набір інструментів для створення, ліплення та редагування 3D-геометрії безпосередньо в Unreal Engine. Ці інструменти пропонують робочі процеси для редагування топології, створення UV, призначення кількох матеріалів, редагування зіткнень і запікання текстур.

Щоб отримати доступ до режиму моделювання, треба клацнути спадне меню «Вибрати режим», а потім клацніть «Моделювання».

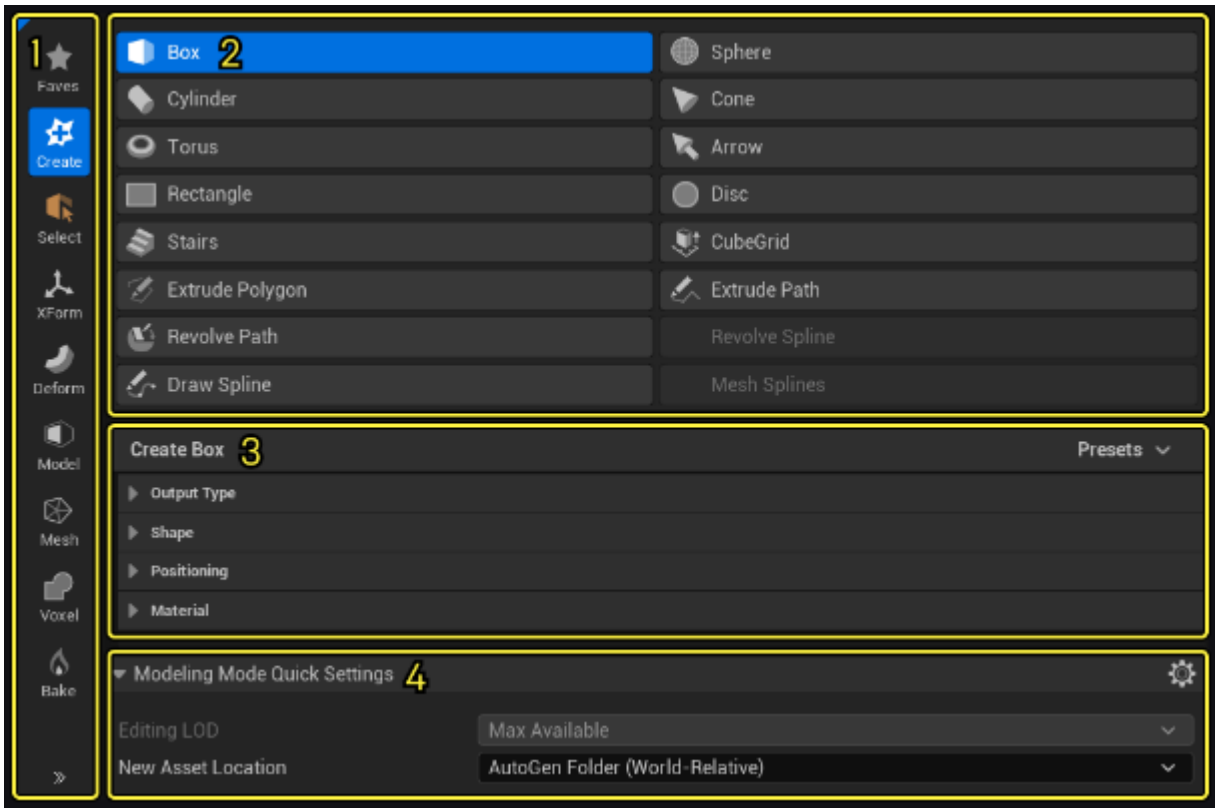

Рис.2.1 – Панелі режиму моделювання

1) Категорія інструментів. Категорії організовують групування пов’язаних інструментів:

- Створити (Create) - категорія для створення нової сітки;

- Вибрати (Select) - редагувати вибір елементів;

- Трансформувати (Transform) - категорія для налаштування розташування або представлення сітки;

- Деформувати (Deform): ліпити або деформувати сітку в цілому або в певних областях;

- Модель (Model) — категорія для детального редагування сітки;

- Сітка (Mesh) - обробка сітки для перевірки, оптимізації та редагування геометрії сітки;

- Воксель (Voxel) — категорія для перетворення сітки на вокселі для виконання об’ємних операцій перед перетворенням її назад на сітку;

- Випікання (Bake): для генерування текстури та даних кольору вершин для сіток;

- Uvs - для редагування UV-координат сітки;

- Атрибути (Attribs) - для перегляду та налаштувань вторинних властивостей сітки;

- Різне (Misc) - додаткові утиліти, такі як керування LOD та перетворення гучності.

Кожен інструмент у категорії відкриває певний набір інструментів і налаштування на панелі «Властивості інструмента».

2) Палітра інструментів. Складається з інструментів вибраної категорії.

3) Властивості інструмента. Налаштування, пов’язані з вибраним інструментом.

4) Швидкі налаштування режиму моделювання. Доступ до загальних налаштувань, таких як розташування шляху для активів, гізмо, розділ елемента та нові об’єкти сітки

2.2 Створення моделі ящика в режимі моделювання

Розглянемо процес створення моделі ящика в стаціонарному режимі моделювання Unreal Engine 5. Для створення моделі потрібно виконати серію дій.

1) Створення примітиву Box. Для початку створюємо новий проект на основі шаблону гри від першої особи і відкриваємо режим моделювання. Далі у категорії «Фігури» вибираємо «Box» та налаштовуємо параметри на панелі «Моделювання» до потрібного розміру. Клікаємо на вікно перегляду, щоб створити ящик, а потім натискаємо «Завершити» внизу екрана.

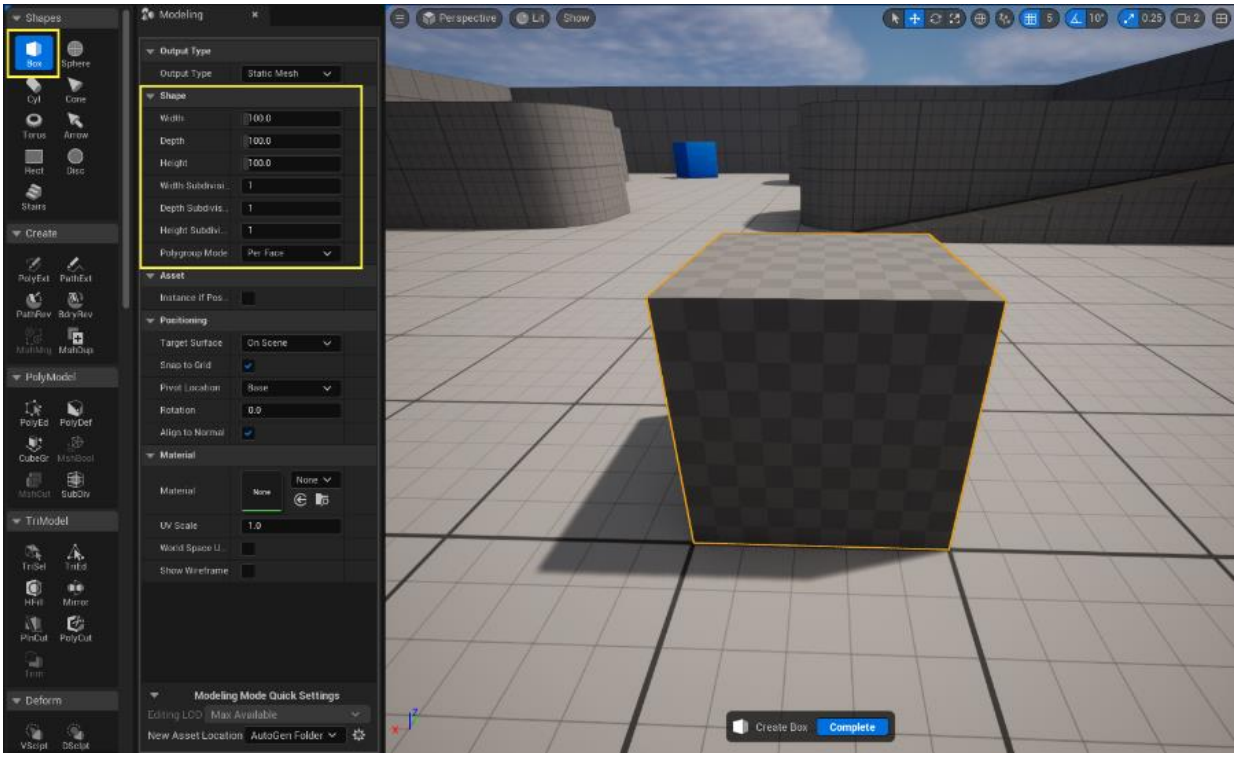

Рис.2.2 – Створення примітиву Box

2) Створення зовнішньої рами. Перетворення примітивної форми на детальну 3D-модель починається з визначення її полігруп (PolyGroups). Для цього вибираємо створений примітив у вікні перегляду і на панелі інструментів режиму натискаємо «GrpGen» в розділі «Атрибути». Це створить PolyGroups для ящика. Відкриваємо інструменти PolyEdit (вибравши PolyEd в розділі PolyModel) на панелі моделювання. Вибираємо одну з граней у вікні перегляду та натискаємо «Вставка» (Inset). Мишкою змінюємо величину вставки до необхідної і клікаємо ще раз для завершення вставки. Вибираємо створену внутрішню грань і натискаємо «Видавити» (Extrude). Видавлюємо лицьову сторону всередину, щоб створити зовнішню раму ящика. Повторюємо цей процес для кожної сторони ящика.

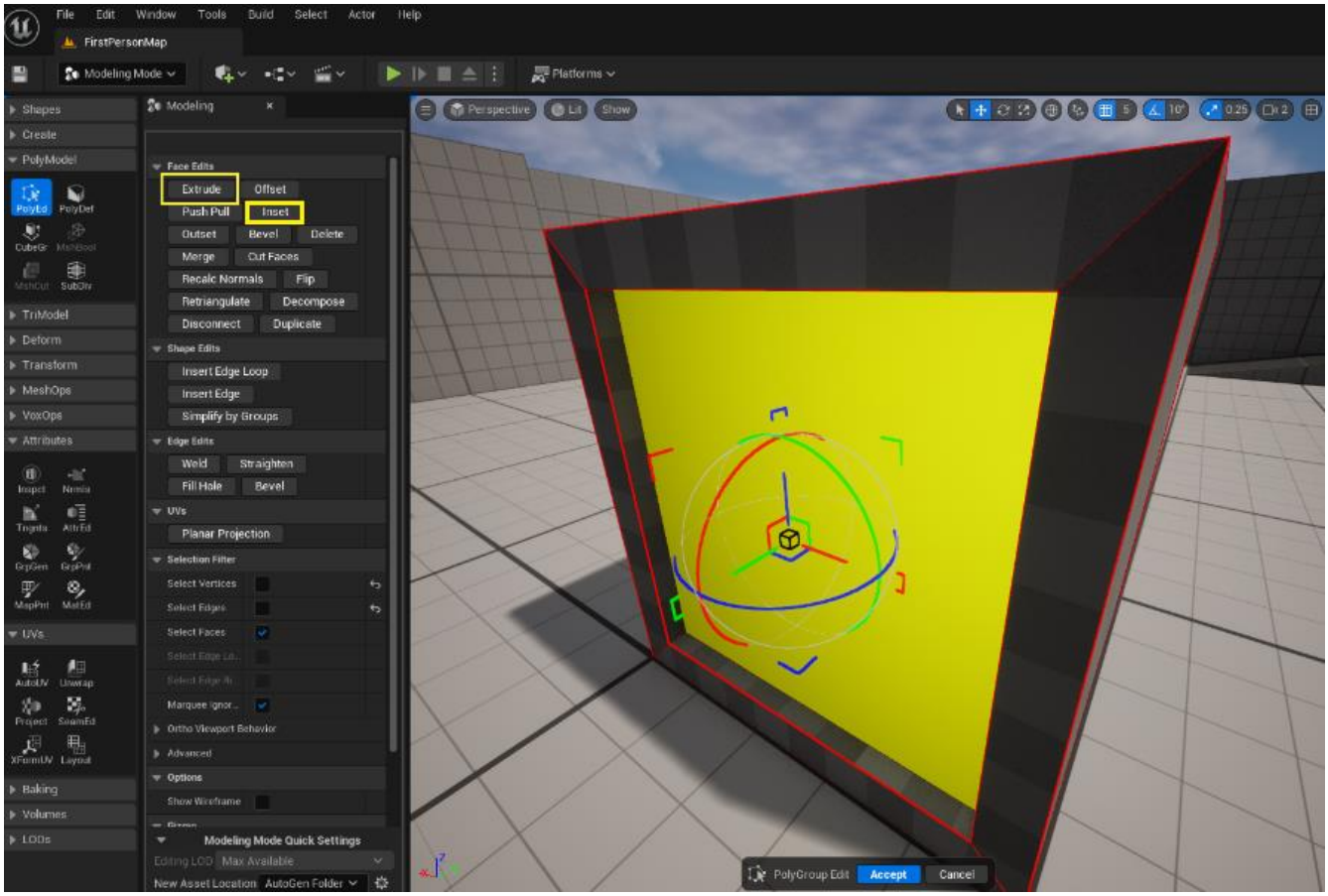

Рис.2.3 – Створення зовнішньої рами ящика

3) Додавання додаткових деталей. Далі додаємо поперечну скобу до зовнішньої рами обрешітки. В інструментах PolyEd вибираємо внутрішню грань, яку щойно видавили, і натискаємо кнопку «Розкласти» (Decompose). Це розділить лицьову сторону на складові трикутники та створить ребро від кута до кута. Вибираємо створене нове ребро. Потім утримуючи Shift, виділяємо додаткові ребра в кутах, з’єднані з новим ребром. Вибравши ребра, переходимо до розділу «Редагування ребер» і натискаємо «Скіс» (Bevel). Залишаємо параметри за замовчуванням і натискаємо кнопку «Застосувати». Це розбиває ребро і створює нову грань. Вибираємо нову лицьову сторону та витягуємо її до рівня зовнішньої рамки. Це створює дві внутрішні грані, одну на кінці екструдованої поперечної скоби, а іншу – у внутрішньому куті.

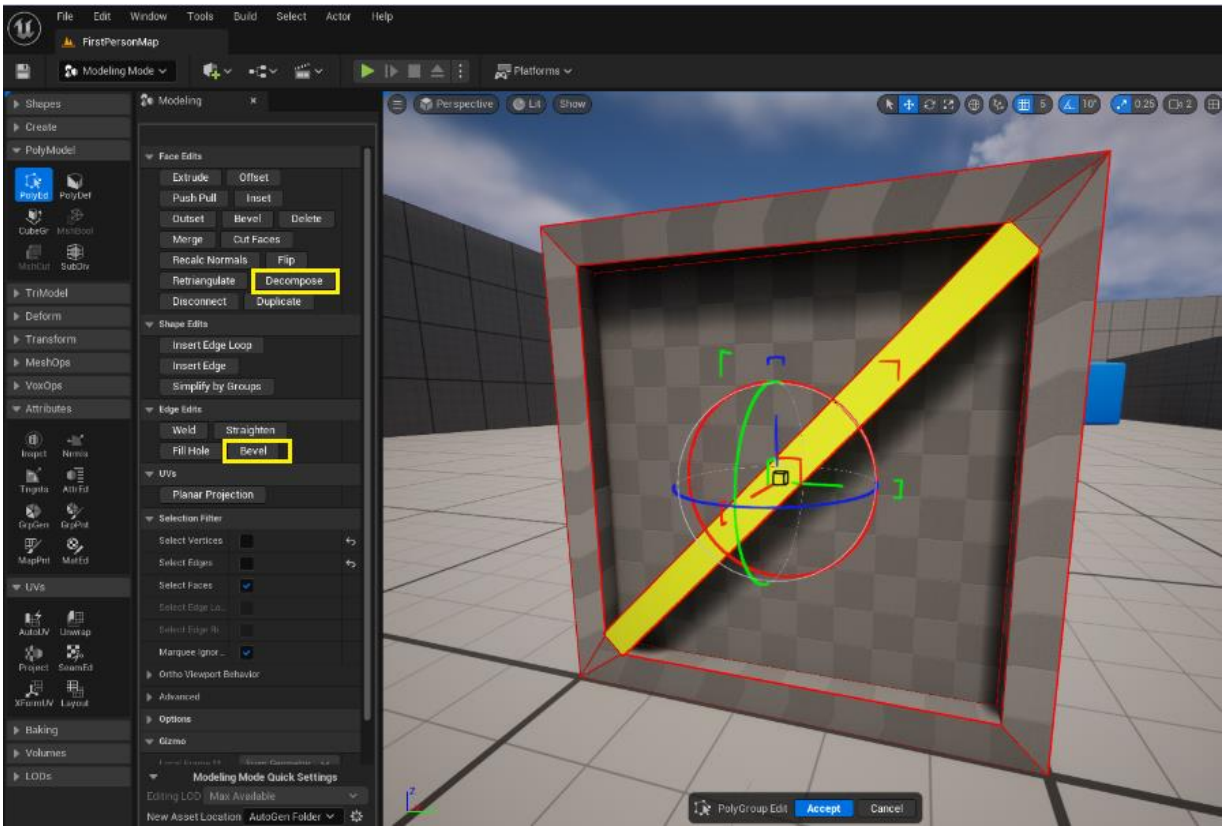

Рис.2.4 – Створення додаткових граней поперечної скоби

Переміщуємо камеру всередину ящика, вибираємо внутрішні грані та видаляємо їх.

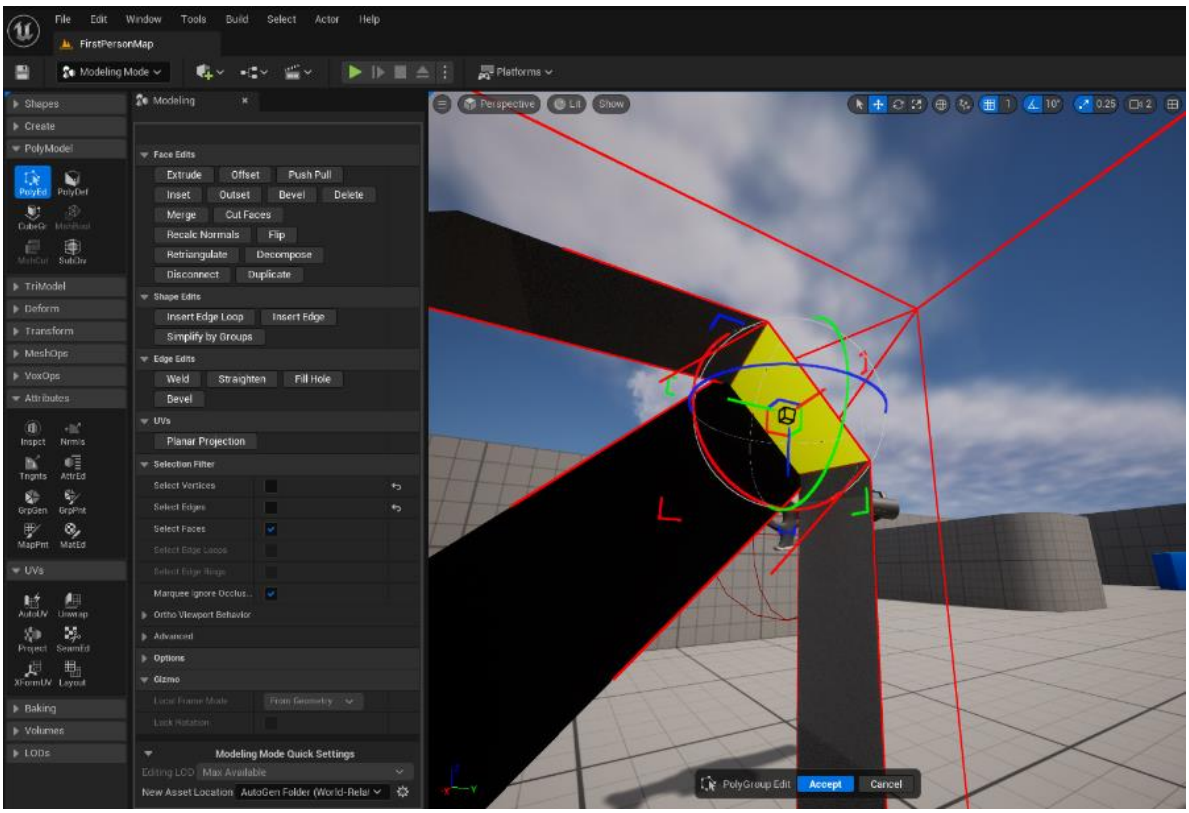

Рис.2.5 – Видалення внутрішніх граней

Переміщуємо камеру назад за межі ящика, щоб вибрати два ребра, розташовані там, де поперечна балка з'єднується з кутом. З'єднуємо ці ребра за допомогою інструменту «Зварка» (Weld). Повторюємо цей процес для іншого кінця внутрішньої рами, а потім і для кожної сторони ящика.

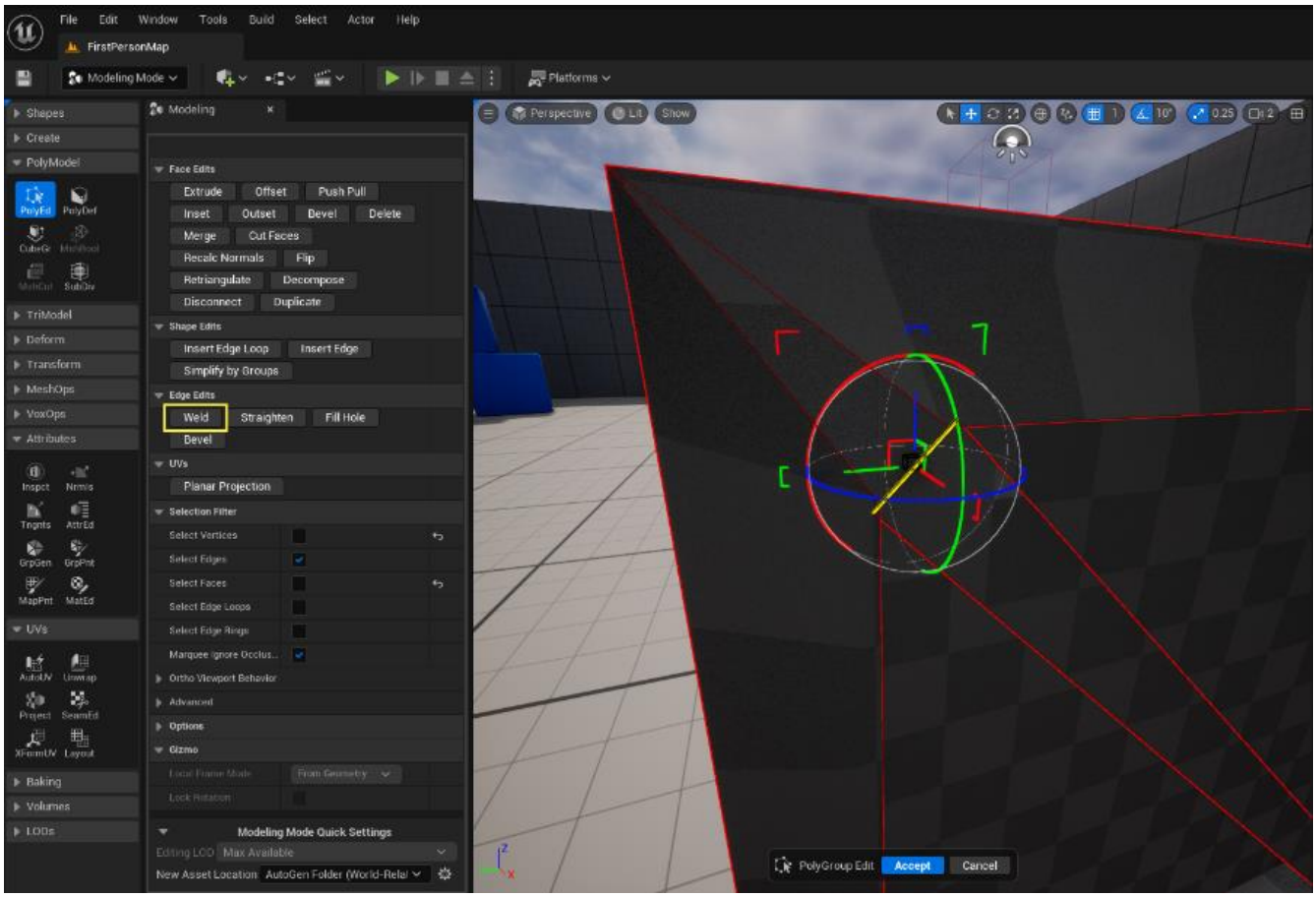

Рис.2.6 – Зварювання ребер скоби

4) Розгортання UV. Щоб текстура деревини була правильно відображена, обрешітку необхідно виконати UV розгортання. Оскільки були внесені значні зміни в оригінальну форму ящика, треба клацнути GrpGen і згенерувати нові PolyGroups для ящика. Змінюємо мінімальний розмір групи на 3, щоб створити шість нових полігруп (по одній на кожну сторону).

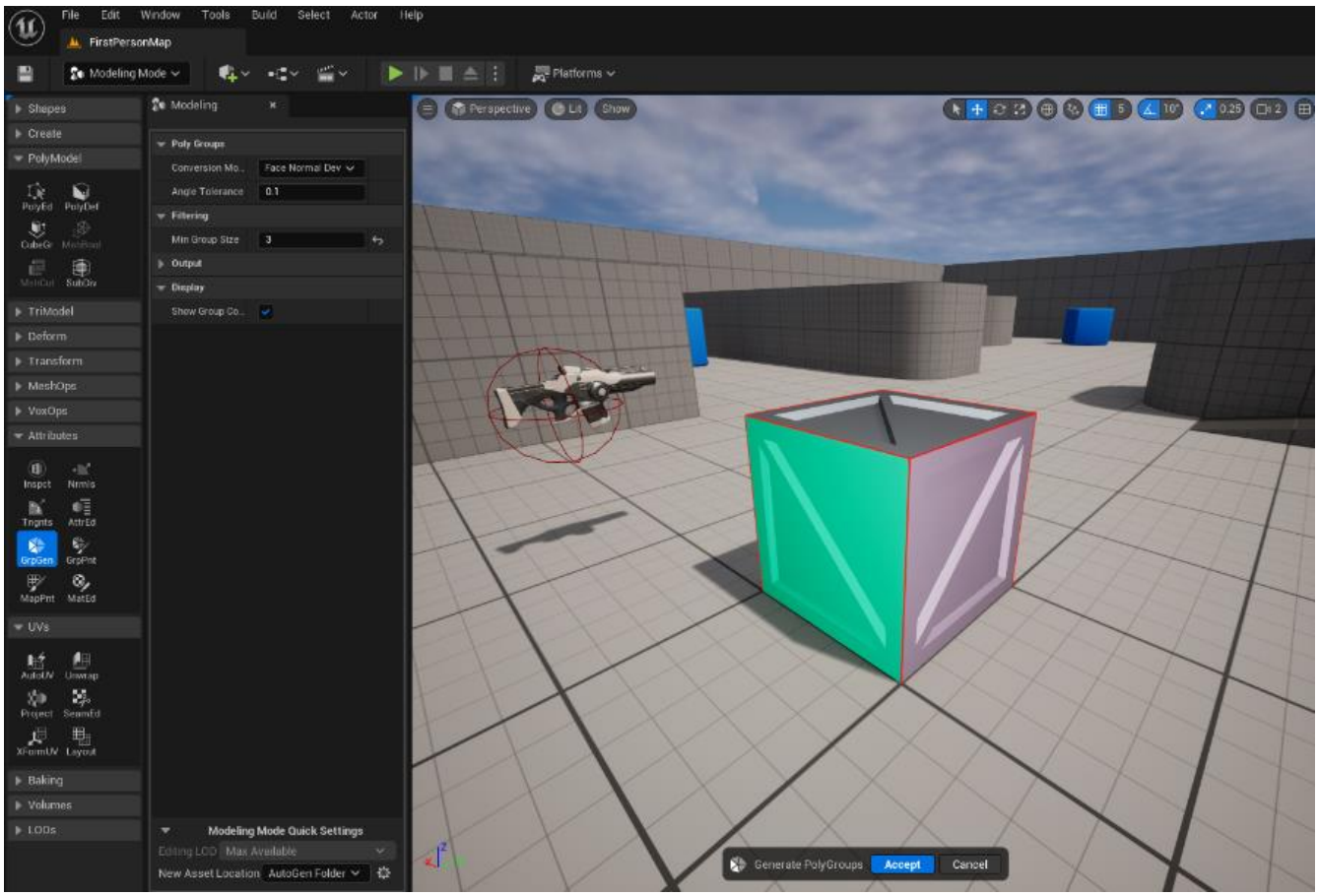

Рис.2.7 – Створення нових полігруп

Далі переходимо до розділу UVs меню та вибираємо інструмент Unwrap. Вибираємо «Полігрупи» (PolyGroups) у спадному меню «Генерація островів» (Island Generation) на панелі «Моделювання». Змінюємо значення роздільної здатності текстури на 2048. Це змінить зсув між UV-острівцями, щоб забезпечити достатню кількість пікселів між ними. Щоб переглянути згенеровану UV-карту, встановлюємо прапорець опції Enabled у розділі Preview UV Layout меню. Приймаємо зміни, щоб завершити UV-карту.

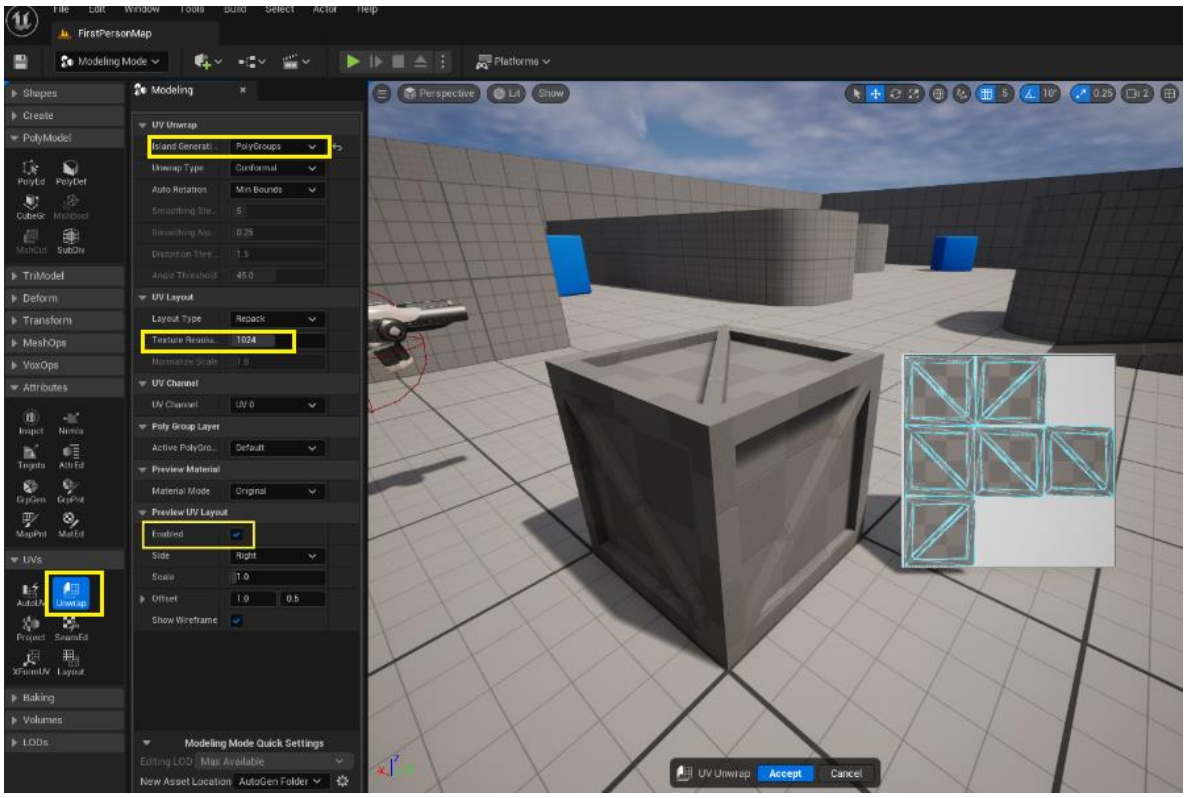

Рис.2.8 – Вікно попереднього перегляду UV-карти

Перевіряємо якість створеної UV-карти, додавши деревний матеріал до ящика. Для цього на панелі «Деталі» відкриваємо розділ «Матеріали» та застосуємо матеріал M_Wood_Pine до ящика.

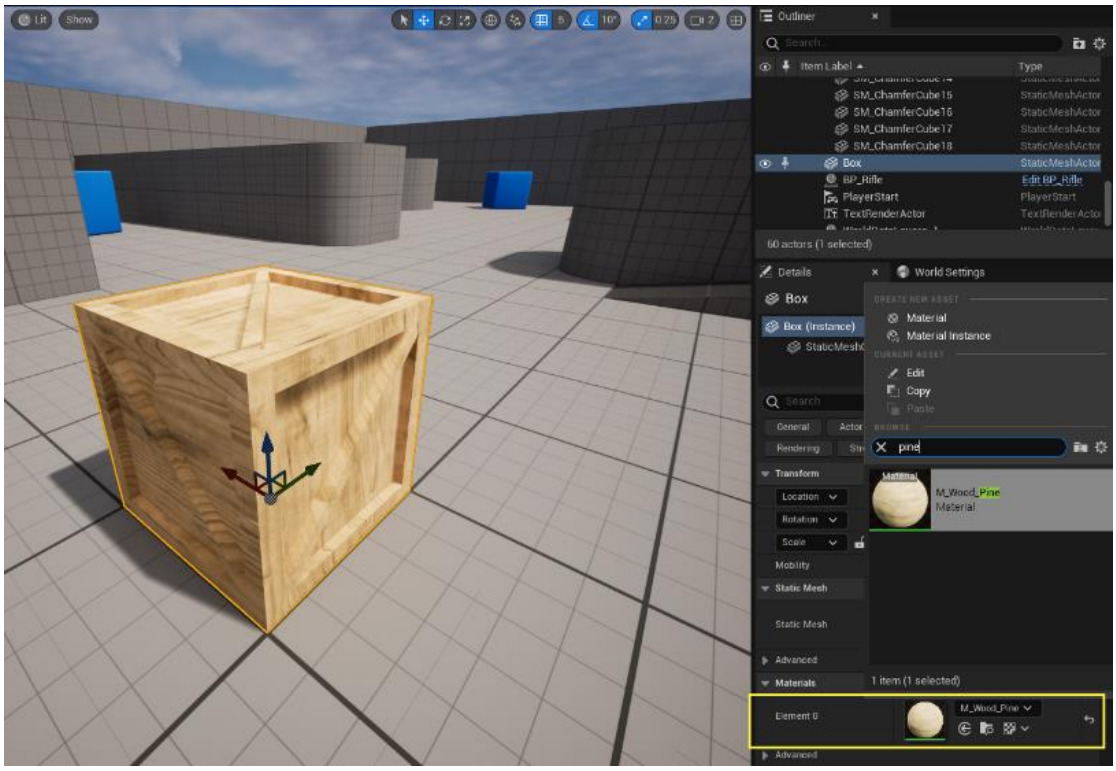

Рис.2.9 – Застосування матеріалу дерева до ящика

5) Перенесення ящика на рівень. Підготовуємо ящик для рівня, додавши спрощене зіткнення ящиків у редакторі статичної сітки та ввімкнувши симуляцію фізики. Для цього знаходимо об’єкт ящика в Content Browser і відкриваємо його в Static Mesh Editor.

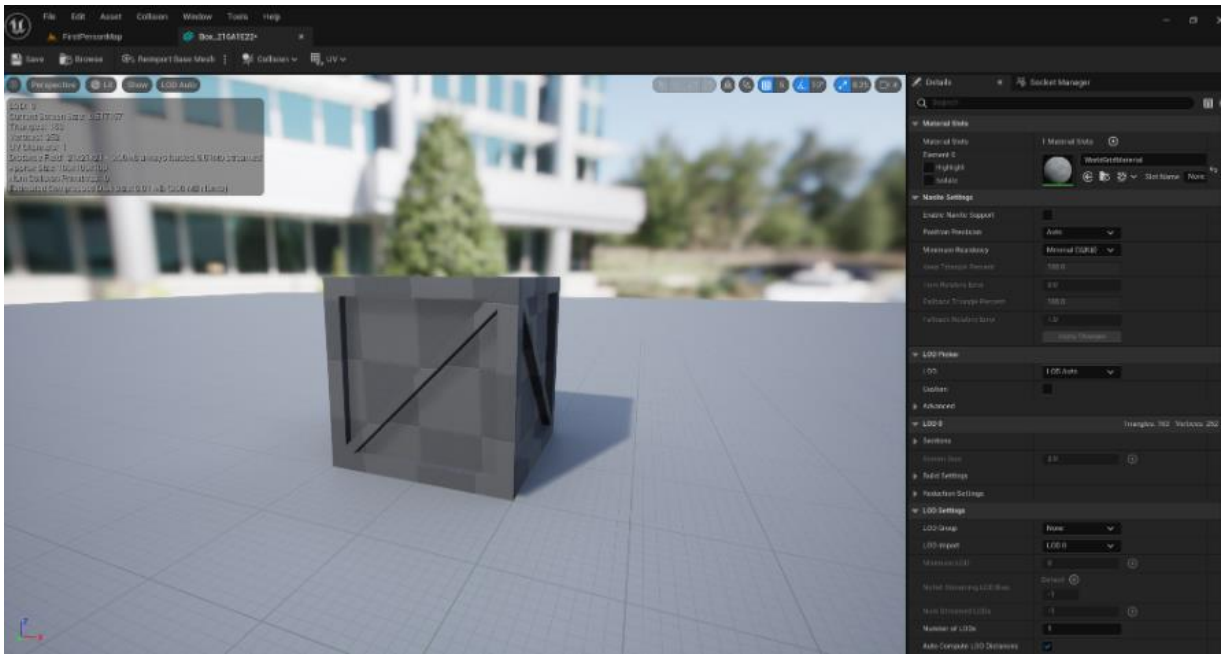

Рис.2.10 – Зіткнення ящика

Вибираємо «Додати спрощене зіткнення коробки» (Add Box Simplified Collision) в меню «Зіткнення» (Collision), щоб додати базове поле зіткнення до ящика.

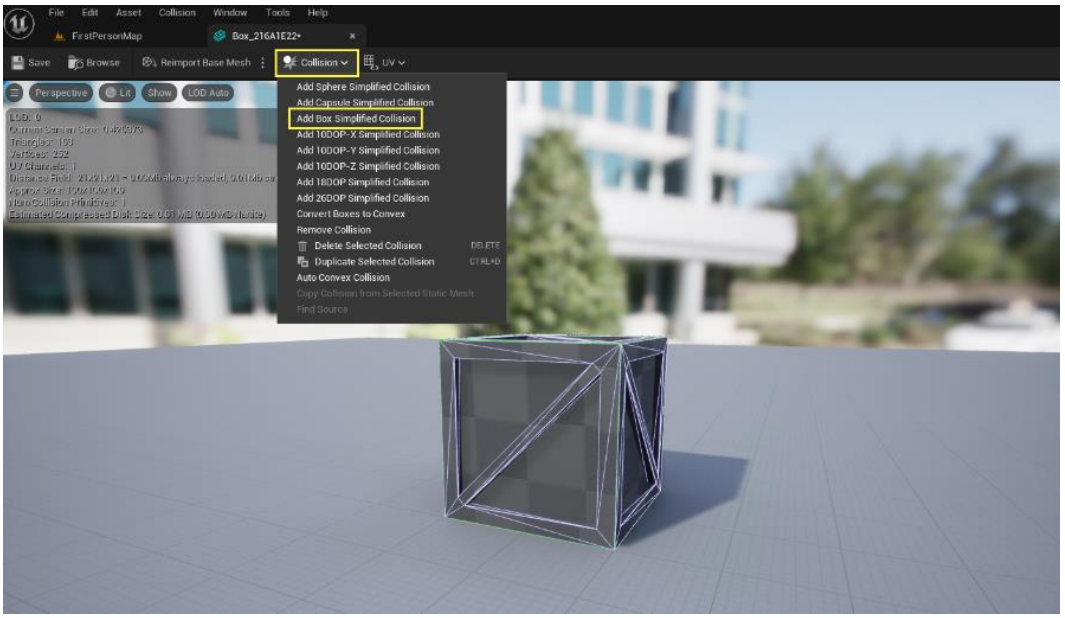

Рис.2.11 – Створений ящик в Static Mesh Editor

Вибираємо «Проект за замовчуванням» (Project Default) в розділі «Складність зіткнень» (Collision Complexity) на панелі «Деталі». Повертаємось у вікно редактора та вмикаємо Simulate Physics на панелі деталей.

Результат. Отже, виконавши серію кроків вдалося створити об’єкт дерев’яного ящика за допомогою набору інструментів режиму моделювання, повністю в редакторі Unreal Editor. Додавши деталі до моделі, UV-карту, матеріал дерева, відповідні налаштування зіткнень і ввімкнувши фізику - зробили створений ящик готовим до гри активом.

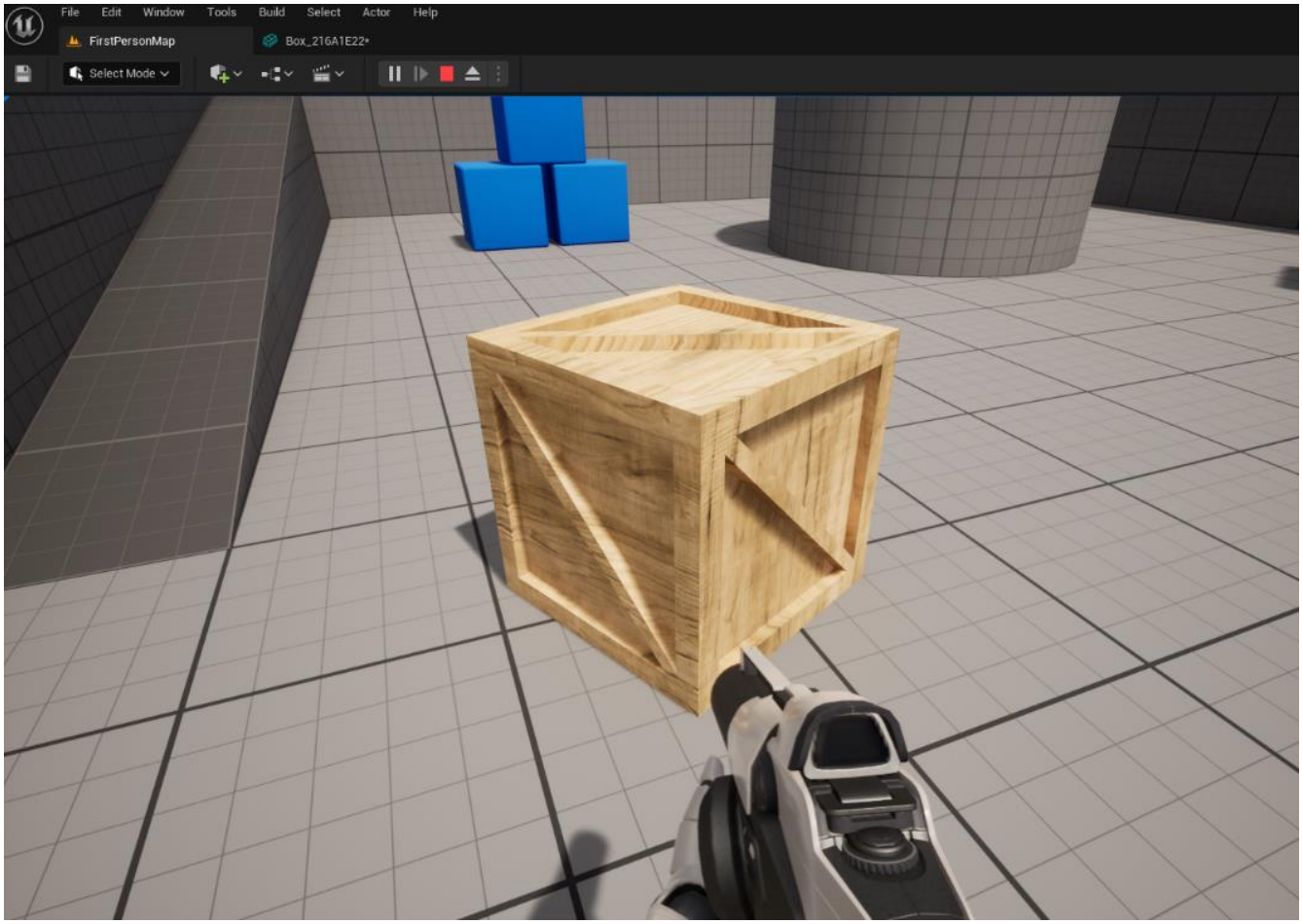

Рис.2.12 – Остаточна версія соснового ящика всередині рівня

2.3 Моделювання плану поверху в Unreal Engine

Unreal Engine часто використовують для створень цифрових зображень будівель і приміщень. Як приклад розглянемо моделювання плану поверху за допомогою режиму моделювання.

Спочатку треба створити проект, використовуючи категорію “Ігри” та шаблон “Пустий” та вибравши мову візуального програмування Unreal Engine

«Blueprints», “Desktop” в Target Platform та “Максимум” в Quality Preset і прибравши прапорці Starter Content and Raytracing. Далі в меню Outliner необхідно видалити ресурс Landscape з усіма його елементами і зберегти поточний рівень. Збережений рівень вибираємо в Maps & Modes у налаштуваннях проекту. Також в налаштуваннях проекту у розділі Rendering прибираємо прапорці для Bloom, Ambient Occlusion, Ambient Occlusion Static Fraction і Auto Exposure. Далі в налаштуваннях світу в категорії Lightmass, в розділі Lightmass Settings, знімаємо прапорець Compress Lightmass.

В панелі “Розмістити акторів”, в меню “Фігури” необхідно знайти форму площини та перетягнути її до редактора. Використовуючи Content Drawer створюємо нову папку (“План поверху”). Імпортуємо план поверху, який моделюємо, перетягнувши його в створену папку - це створить текстуру. Додаємо текстуру до попередньо створеної форми площини, перетягнувши її з браузера вмісту та опустити на площину активу.

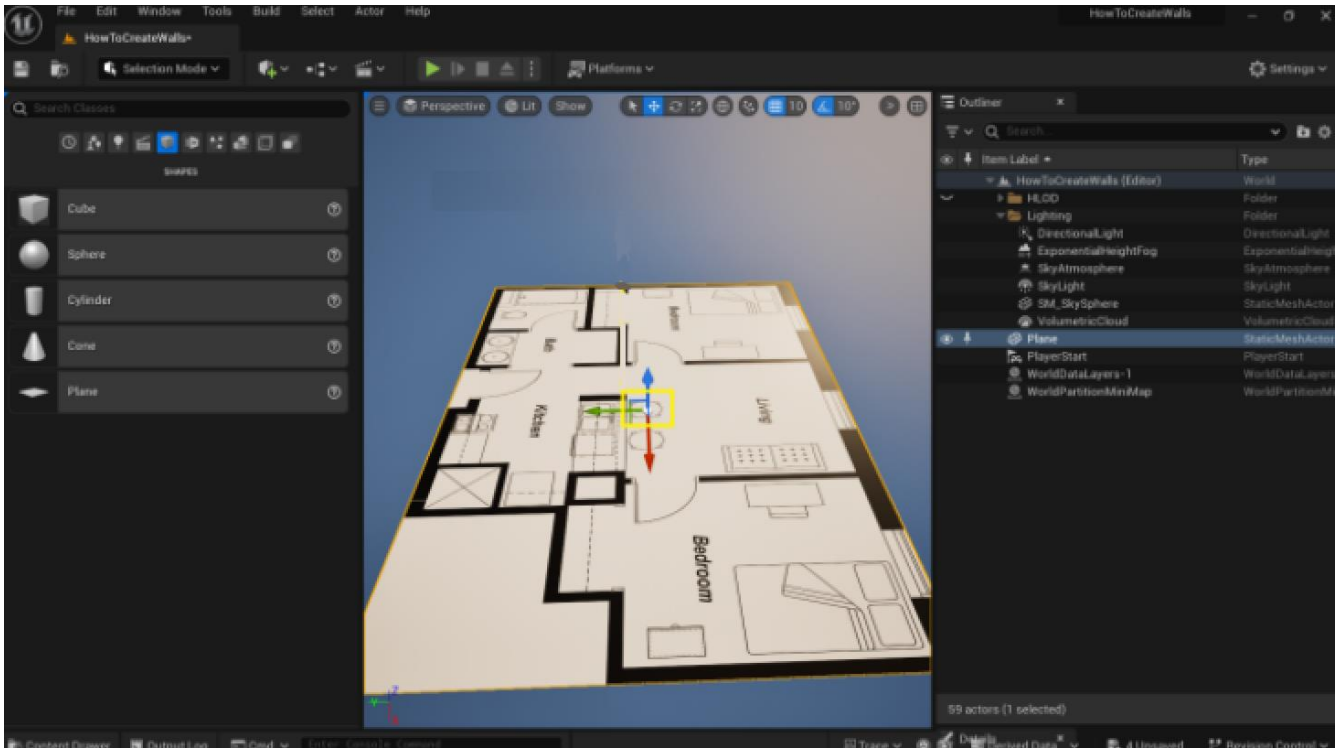

Рис.2.13 – Інтерфейс моду моделювання з імпортованим планом поверху

Активуємо вид зверху вікна перегляду рівня. Вибираємо актив в меню Outliner і фокусуємо його у вікні перегляду. Щоб план поверху відображався потрібно перейти у режим перегляду, вибрати режим без освітлення та

деактивувати сітку в меню “Show”. Далі додається форма куба і масштабується до 0.9, що є стандартною шириною дверей (можна вибирати і інші розміри). Масштабуємо площину активу, посилаючись на розмір ширини дверей. Масштабувати площину активу потрібно, поки ширина куба не співпаде з розміром дверей. Далі треба повторити цю процедуру з іншими дверима на протилежній осі.

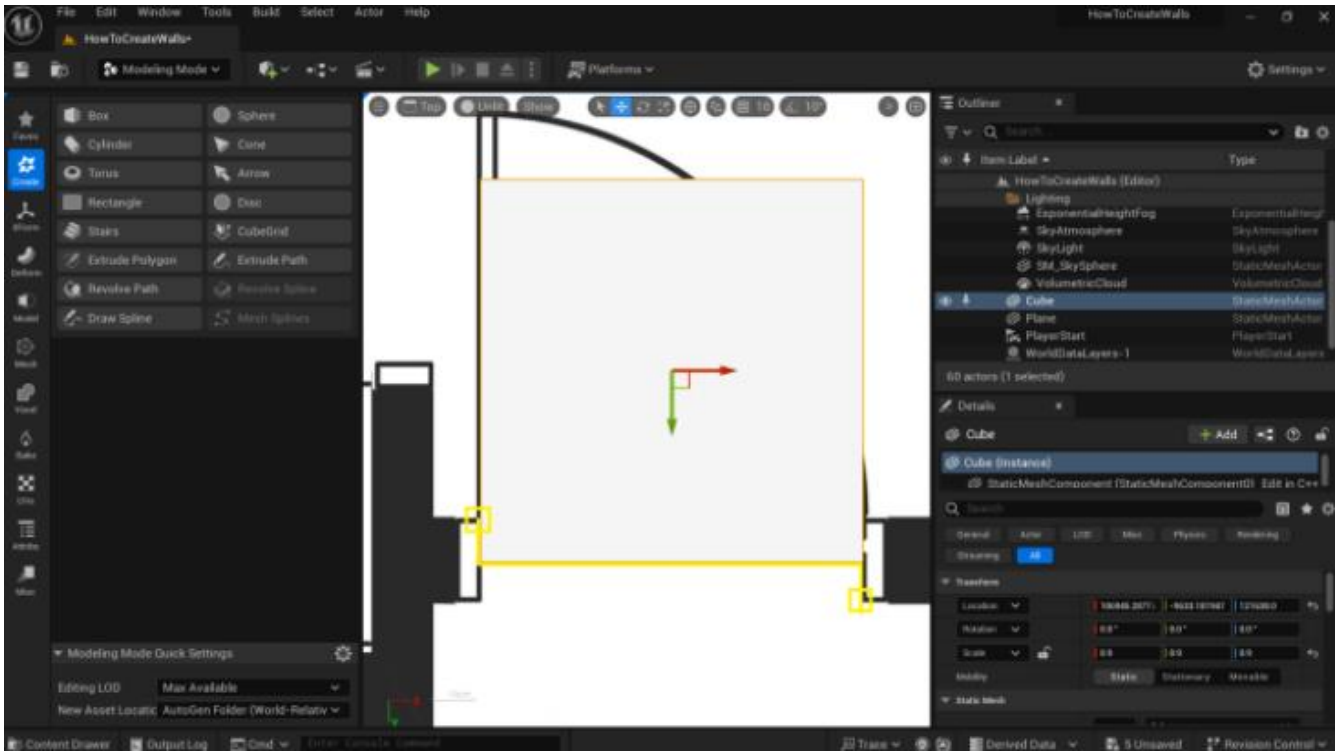

Рис.2.14 – Масштабування куба до розміру дверей

Змінюємо вікно перегляду на перспективу. Переходимо до Modeling Mode і додаємо Box в категорії Create, вибравши Per Face у властивостях PolyGroup Mode. Розміщуємо Box на межі стіни. Активуємо інструмент PolyGroups Edit в категорії Model, вибираємо грань коробки, яка не відповідає фазі стіни, і за допомогою gizmo перетягуємо її, доки об’єм не матиме такої ж товщини, як стіна на плані. Після того, як об’єм підігнаний до товщини стіни, виділяємо грань, протилежну краю, і тягнемо її, поки не знайдемо перетину з іншою стіною. Коли знайшли межу нової стіни, вибираємо властивість видавлювання (Extrude) і витягуємо, щоб завершити товщину стіни, яку знайшли. Потім знову використовуємо інструмент видавлювання, поки не досягнемо краю нової стіни. Повторюємо цю процедуру до тих пір, поки не буде завершено всі стінки, також видавлюємо грані поперечних

стінок. Щоб об’єднати дві грані, якщо вони розділені, використовується властивість Merge. Щоб завершити редагування та прийняти зміни, натискаємо кнопку «Прийняти», це створить статичну сітку з розробленою моделлю.

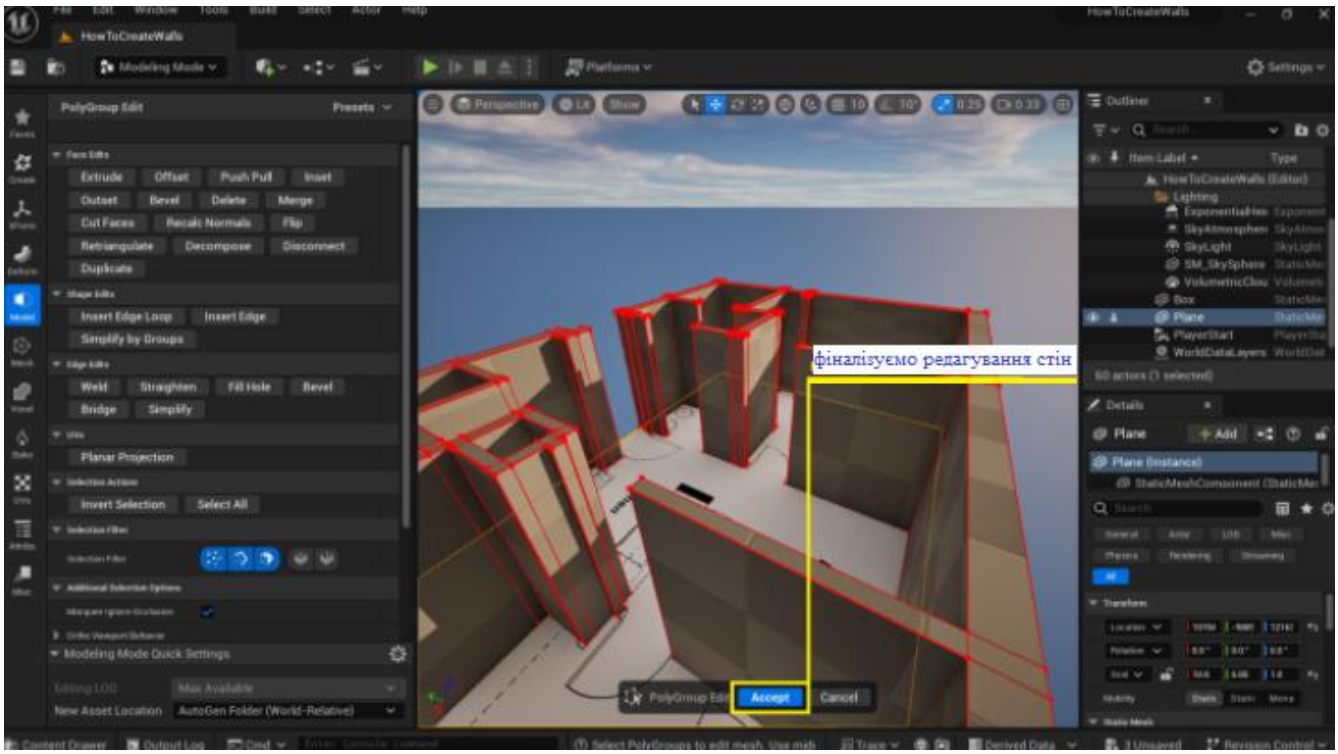

Рис.2.15 – Редагування стін

В Content Browser створюємо нову папку та називаємо її Materials, а всередині папки створюємо клас матеріалу з назвою M_Walls. В редакторі матеріалів, підключаємо створений матеріал до базового кольору (білий). Застосовуємо матеріал, перетягнувши його з Content Browser на активне вікно перегляду. Тепер можна візуалізувати у вікні перегляду стіни з застосованим матеріалом.

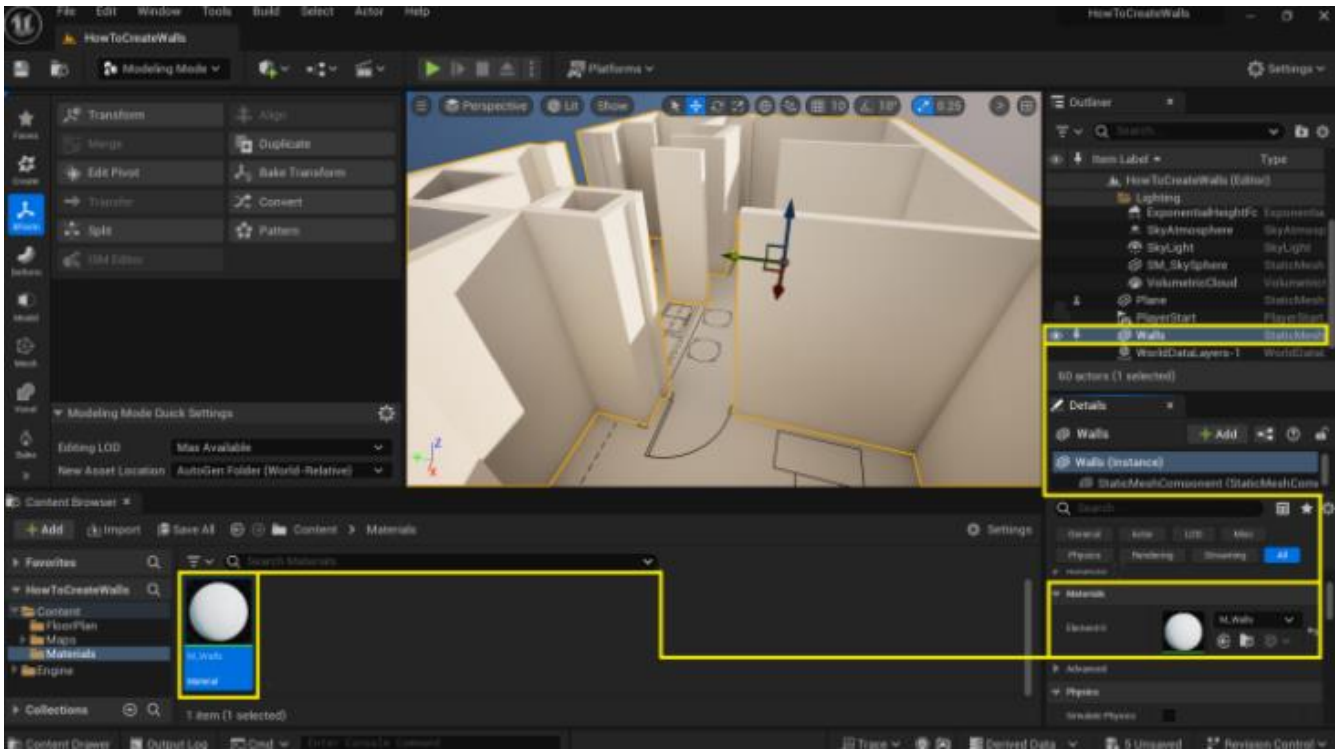

Рис.2.16 – Візуалізація стін

Щоб створити віконні отвори в стінах, вибираємо Актив у вікні перегляду або в Outliner, а потім ховаємо його. Далі знову створюємо Box, але тепер уже з розмірами вікна. Створюємо кількість блоків, що відповідає кількості вікон. Щоб розташувати ящики на відповідній висоті, можна створити напрямну коробку із середньою висотою, на якій починається вікно, яка становить приблизно 1,2 м (висота коробки становитиме 120). Потім розташовуємо прямокутники поверх еталонного вікна. Потім видаляємо опорні поля, вибираючи їх у вікні перегляду або в структурі та натискаючи клавішу Delete, а потім приховуємо статичну сітку стін.

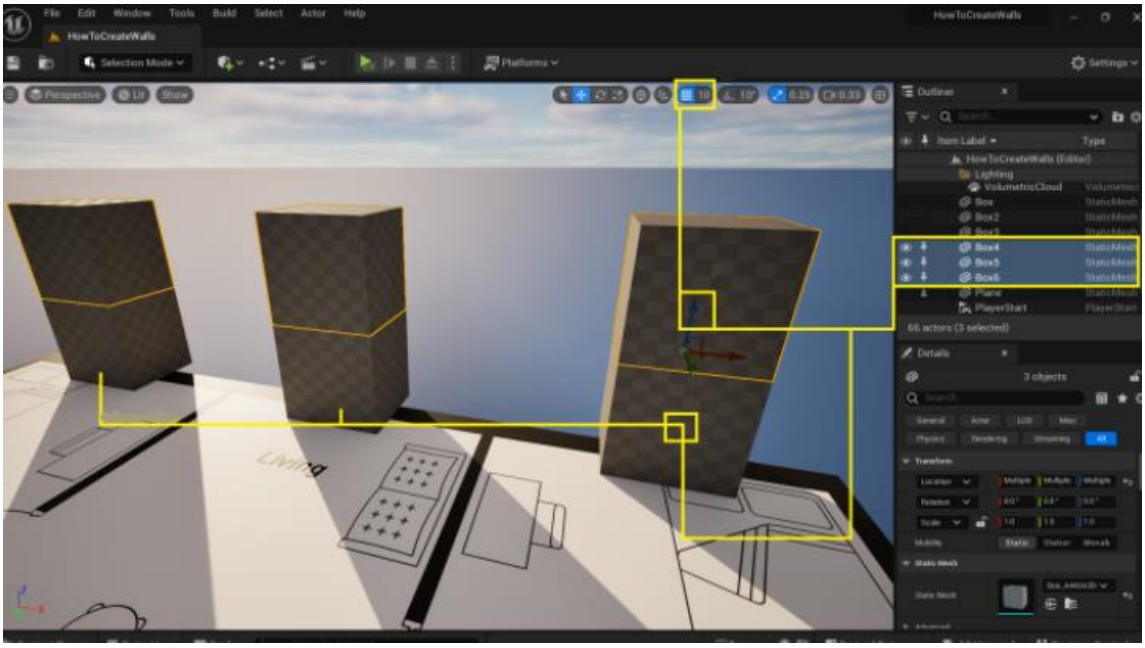

Рис.2.17 – Розміщення вікон

Щоб створити отвори, спочатку вибираємо статичну сітку стін, а потім, утримуючи клавішу CTRL, вибираємо статичну сітку коробів. Зробивши вибір, переходимо в режим моделювання до категорії моделі, і там вибираємо інструмент Boolean, у властивості операції вибираємо параметр Difference A - B, і в Write To "Firts Input Object", нарешті в Handle Inputs вибираємо Видалити вхідні дані та приймаємо зміни.

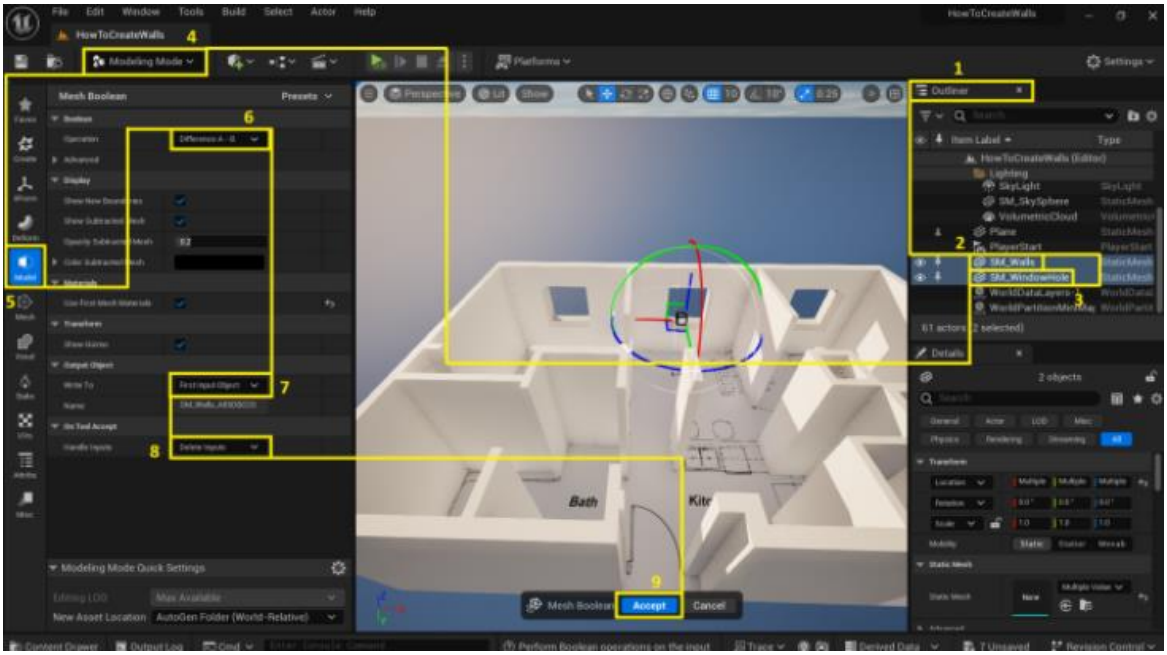

Рис.2.18 – План після створення віконних отворів

Для даху створюємо Box і розміщуємо його поверх стін у кутку, щоб зовнішні осі стіни збігалися з торцями короба. Потім за допомогою інструменту

PolyGroups Edit вибираємо грані та переміщуємо їх за допомогою gizmo, поки не покриємо відповідну поверхню. Після того, як ми створили необхідні краї для формування граней зон, використаємо інструмент Push Pull, щоб обмежити периметр даху відповідно до форми стін.

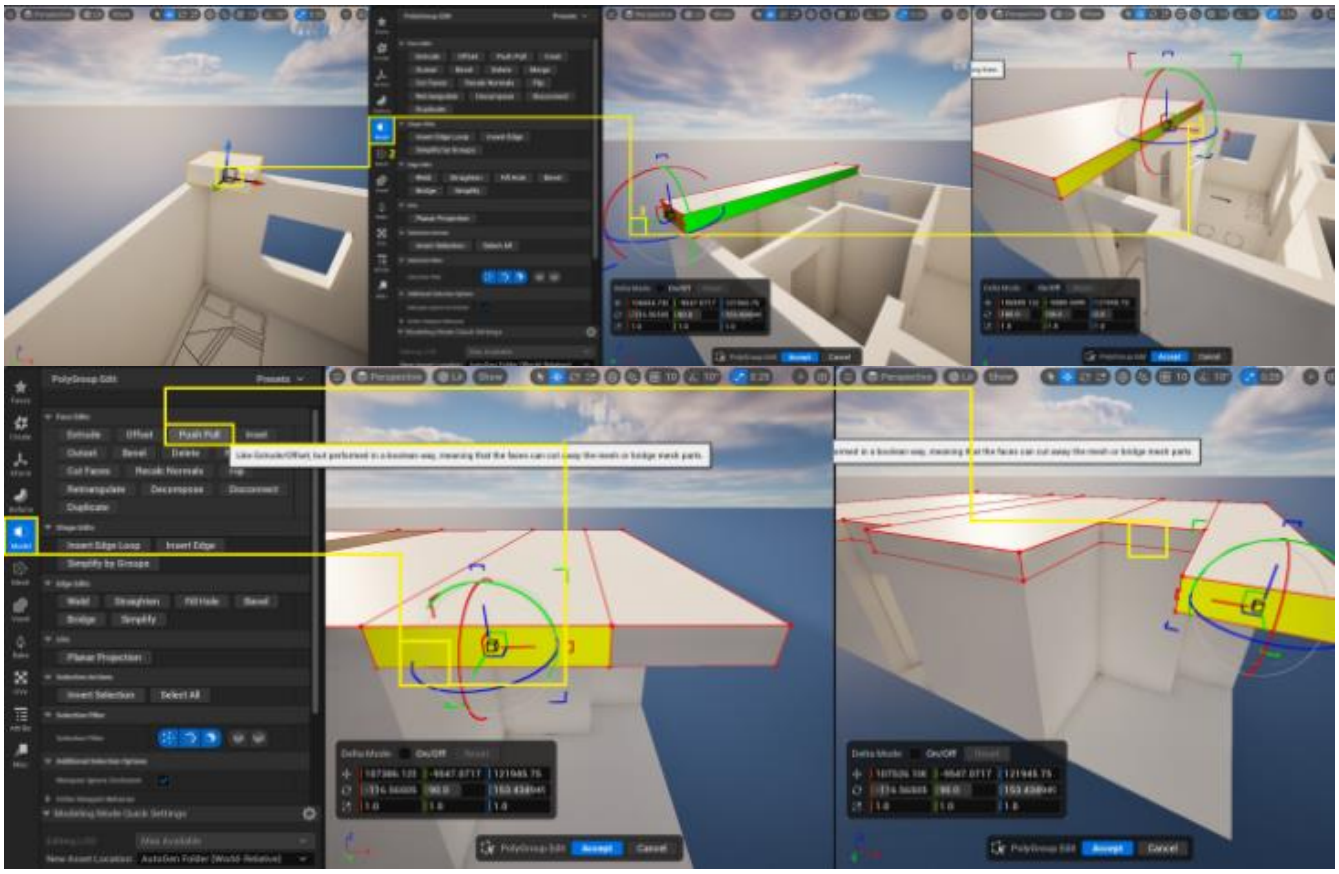

Рис.2.19 – Створення даху

Коли дах буде закінчено, вибираємо об’єкт із Outliner і переходимо до вигляду спереду, активуємо режим неосвітленого перегляду, створюємо копію об’єкта, розташовуємо даний актив прямо під стінами. Можна використати режим каркасного перегляду, щоб розмістити його правильно. Коли актив буде розміщено, це буде підлога моделі.

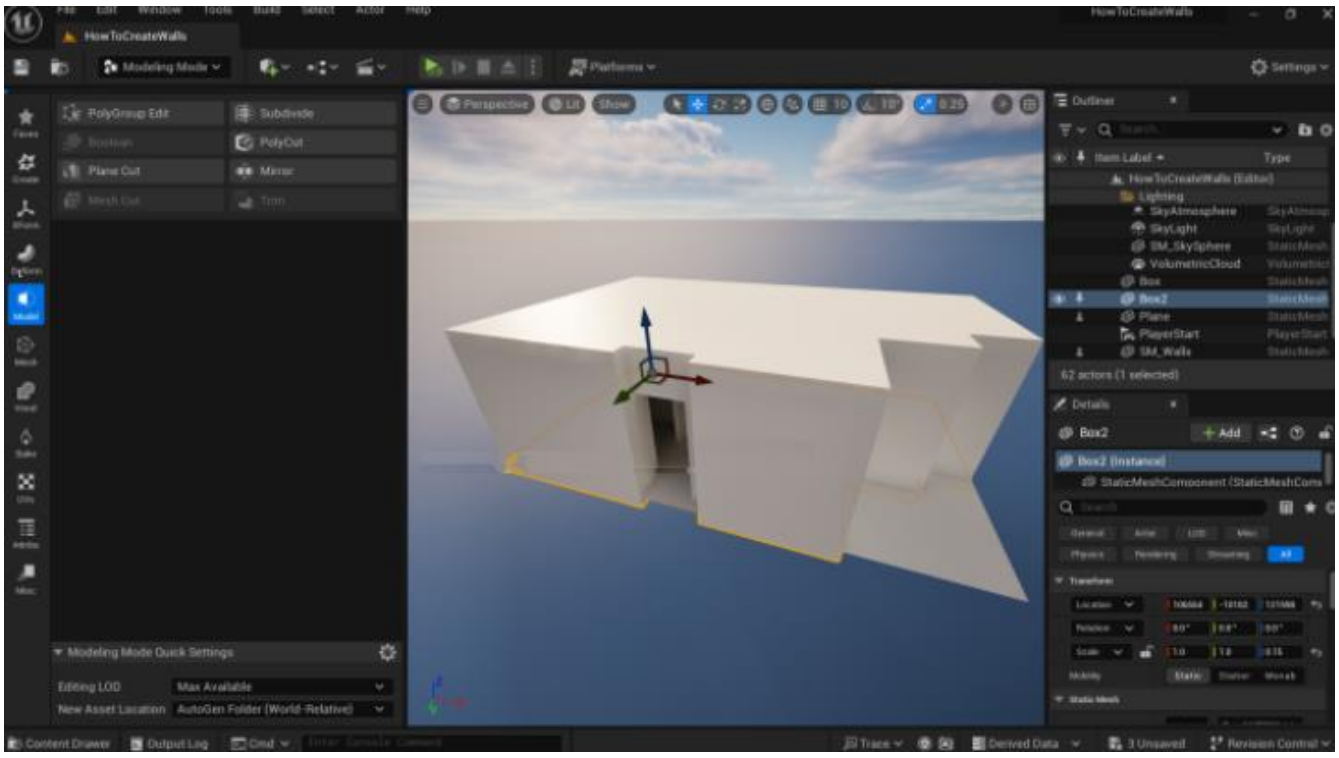

Рис.2.20– Модель плану поверху

2.4 Переваги моделювання Unreal Engine

Моделювання Unreal Engine пропонує кілька переваг, які роблять його кращим вибором для багатьох розробників і художників. Ось деякі ключові переваги використання Unreal Engine для моделювання:

- візуалізація в реальному часі: Unreal Engine надає потужну систему візуалізації в реальному часі, яка дає змогу переглядати моделі та сцени в реальному часі під час роботи над ними. Цей миттєвий зворотній зв'язок дає змогу негайно вносити зміни та повторювати проекти, що призводить до більш ефективного та спрощеного робочого процесу;

- фотореалістична графіка: Unreal Engine відомий своїми приголомшливими візуальними можливостями, що дозволяє створювати надзвичайно реалістичне середовище з ефектом занурення. Завдяки вдосконаленим системам освітлення, затінення та матеріалів можна досягти вражаючих рівнів візуальної точності, оживляючи моделі з винятковою деталізацією та реалістичністю;

- велика бібліотека активів: Unreal Engine має величезну бібліотеку готових активів, включаючи моделі, текстури, матеріали та ефекти. Ця обширна бібліотека активів надає велику кількість активів, які можуть значно пришвидшити процес

моделювання, дозволяючи більше зосередитися на творчих аспектах, а не починати з нуля;

- візуальний сценарій Blueprint: Unreal Engine пропонує потужну систему візуального сценарію під назвою Blueprint, яка дозволяє створювати інтерактивну поведінку та механізми ігрового процесу без необхідності традиційного програмування. Ця функція особливо корисна для художників і дизайнерів, які хочуть оживити свої моделі за допомогою інтерактивних елементів і функціональності;

- сумісність між платформами: Unreal Engine підтримує кілька платформ, включаючи ПК, консолі, мобільні пристрої та платформи віртуальної реальності (VR). Ця крос-платформна сумісність дає змогу створювати моделі, які можна розгортати на різних пристроях, охоплюючи ширшу аудиторію та максимізуючи потенціал;

- спільнота та підтримка: Unreal Engine має велику та активну спільноту розробників, художників та ентузіастів. Ця спільнота надає велику кількість ресурсів, навчальних посібників, форумів і активів ринку, які можуть допомогти у навчанні, усуненні несправностей і розширенні знань про можливості моделювання Unreal Engine.

Використовуючи ці переваги, Unreal Engine дозволяє художникам і розробникам створювати візуально приголомшливі та інтерактивні моделі для різноманітних додатків, включаючи ігри, віртуальну реальність, візуалізацію архітектури тощо. Його повний набір інструментів, можливості роботи в реальному часі та надійні функції роблять його універсальним і потужним вибором для моделювання та розробки ігор

РОЗДІЛ 3. СТВОРЕННЯ ГРАФІЧНИХ 3D-МОДЕЛЕЙ В СИСТЕМІ UE

3.1. Графічні 3D-моделі.

За своєю суттю 3D-моделювання - це процес створення цифрового представлення тривимірного об’єкта або середовища за допомогою спеціалізованого програмного забезпечення. Він передбачає створення віртуальних об’єктів із точною геометрією, поверхнями, текстурами та іноді анімацією для імітації їхніх аналогів у реальному світі. 3D-моделювання є основоположним у різних галузях, включаючи відеоігри, кіно та анімацію, архітектуру, дизайн продуктів і досвід віртуальної реальності.

У сфері комп’ютерної графіки тривимірні моделі будуються за допомогою багатокутників, які є плоскими поверхнями, з’єднаними вершинами, ребрами та гранями. Художники та дизайнери маніпулюють цими геометричними елементами для створення складних форм і структур, включаючи такі деталі, як колір, текстура та освітлення, щоб підвищити реалістичність і візуальну привабливість.

Процес 3D-моделювання вимагає поєднання технічних навичок, художнього бачення та уваги до деталей. Це передбачає вибір відповідних методів моделювання, використання еталонних зображень або студії концептуального мистецтва, як посібників і використання програмних засобів для маніпулювання віртуальними об’єктами у віртуальному тривимірному просторі.

Найпоширенішими методами моделювання є:

- Коробкове моделювання (Box modeling). У даній техніці художник використовує геометричну форму, як-от куб, циліндр або куля, і формує її до досягнення запланованого вигляду. Моделісти здійснюють процес у різні етапи. Вони починаються з полігональної сітки з низькою роздільною здатністю, а потім уточнюють форму. Потім вони далі розділяють сітку, забезпечуючи згладжування жорстких країв і додавання необхідних деталей. Вони повторюють процес уточнення та поділу, доки в сітці не буде достатньо полігональних деталей, які можуть передати бажану концепцію. Коробкове моделювання є одним із найпоширеніших полігональних 3D-технік і використовується в поєднанні з моделюванням країв.

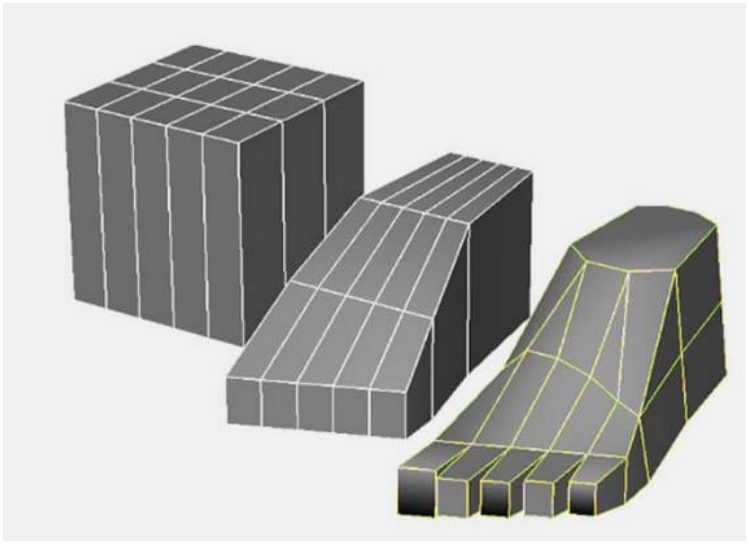

Рис.3.1 – Коробкове моделювання

- Моделювання контурів/країв. Моделювання країв є ще одним типом багатокутної техніки, хоча вона відрізняється від техніки коробки. У цьому процесі моделісти розвивають модель по частинах замість того, щоб вдосконалювати примітивну форму. Це робиться шляхом розміщення петель багатокутників уздовж контурів і заповнення проміжків, які лежать між ними. Цей процес застосовано, тому що важко завершити певні сітки за допомогою техніки моделювання коробки.

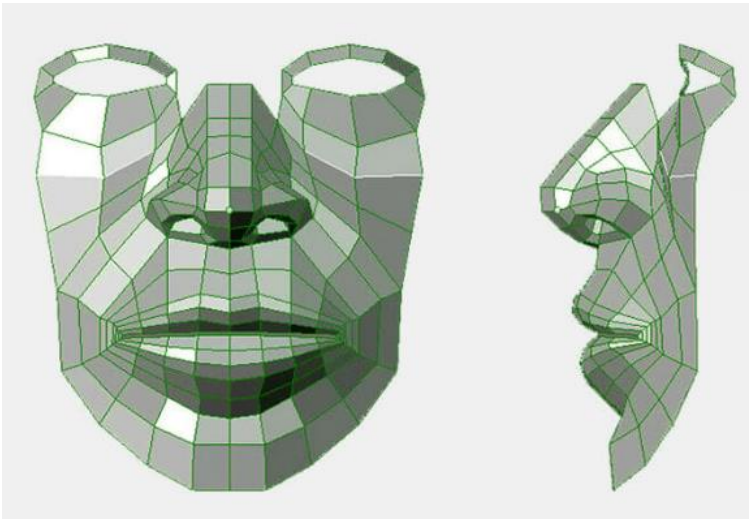

Рис.3.2 – Моделювання контурів/країв

- Сплайн/NURBS моделювання. Ця техніка моделювання Nurbs широко використовується в промисловості та автомобільній промисловості. Сітка NURBS не має ребер, граней або вершин. Ці моделі мають поверхні, які можна легко інтерпретувати. Модельєри можуть розвинути концепцію, розмістивши сітку між

сплайнами. Крива NURBS розробляється за допомогою інструменту, схожого на інструмент пера, який використовується в Adobe Photoshop або MS Paint. Модельєри малюють криві в 3D-просторі та редагують їх, переміщуючи контрольні вершини, які є серією маркерів. Криві розміщуються уздовж помітних контурів, а простір між ними автоматично інтерполюється програмами 3D-моделювання. Крім того, також можна створити криву NURBS за допомогою профільної кривої, обертаючи її навколо центральної осі. Це один із найпоширеніших методів проектування 3D-моделі, який використовується для розробки таких об'єктів, як вази, келихи та тарілки.

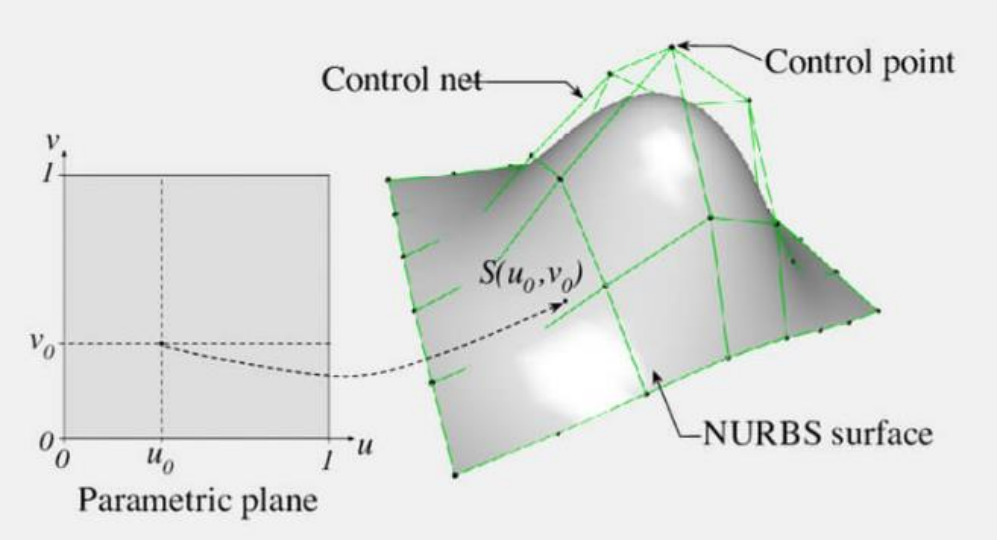

Рис.3.3 – Моделювання NURBS

- Моделювання підрозділів (Subdivision modeling). Моделювання підрозділів виконується за допомогою комбінації методів полігонального та NURBS-моделювання. У цьому гібридному процесі багатокутна модель допомагає створювати 3D-моделі, які потім перетворюються на моделі підрозділів. Художник має можливість контролювати доопрацювання 3D-моделі в певних областях. Крім того, для роботи з цими моделями можна використовувати різноманітне програмне забезпечення. Багатокутник потрібно розділити та уточнити, поки деталі не стануть чіткими. З більшим поділом поверхня стає більш гладкою.

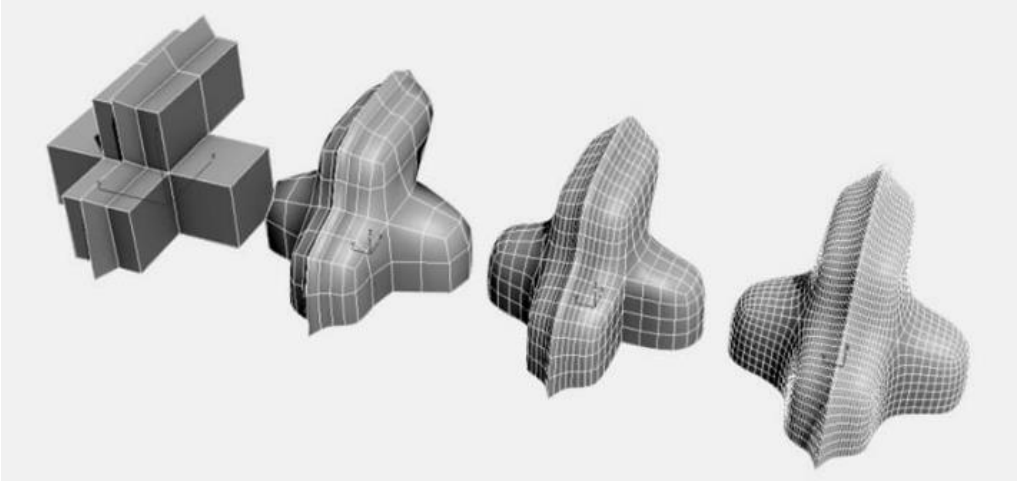

Рис.3.4 – Моделювання підрозділів

- Цифрове ліплення. Цифрова скульптура — це вид революційної технології, яка значною мірою використовує процес 3D-моделювання. Модельєрам тепер не потрібно виконувати копіткі обмеження потоку країв і топографії. Це дозволяє їм створювати різні типи 3D-моделей у спосіб, подібний до процесу ліплення з цифрової глини. Тут сітки створені органічно. Для формування моделі використовується планшетний пристрій, так само як скульптор використовує пензель на шматках глини. Завдяки цифровому ліпленню ліплення істот і персонажів вийшло на новий рівень. Модельєри можуть виконувати процес набагато швидше та з більшою ефективністю. Художники можуть працювати з сітками високої роздільної здатності, що містять мільйони багатокутників.

Рис.3.5 – Цифрове ліплення

- Модульне моделювання. Воно часто застосовуэться при моделюванні кількох подібних об’єктів, наприклад будівель. Це модульне моделювання допомагає замінювати компоненти в 3D-моделі, що навіть може доходити до

моделювання різних секцій будівлі та допомагати змінювати їх у різні способи для створення варіацій.

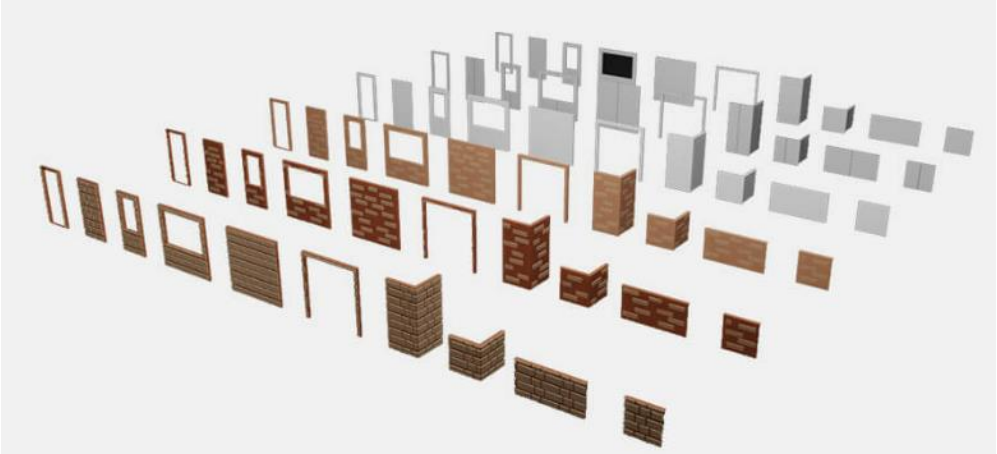

Рис.3.6 – Модульне моделювання

- Kitbashing - використовується для деталізації об’єкта, створеного за допомогою різних типів моделювання. Це тип моделювання, який починається з об’єднання набору об’єктів у більш детальні об’єкти. Ця техніка зазвичай використовується під час створення об’єктів із твердою поверхнею та допомагає дослідити, як різні деталі можуть поєднуватися ще до початку будівництва. Цей прийом є чудовим способом деталізації сцени. Використовуючи цю техніку, слід пам’ятати про три речі: співвідношення високої, середньої та низької деталізації.

Рис.3.7 – Kitbashing

- Фотограмметрія - це зовсім інший спосіб створення 3D-моделей за допомогою камери. Фотографуючи об’єкт кілька разів під кращими кутами в умовах ідеального освітлення, а потім передаючи це зображення в програму, яка інтерпретує їх і створює тривимірне представлення об’єкта, можна отримати дані

реального світу, тобто все, що створюється, наближається до реалістичності. Ця техніка є одним новим винаходом, який набув великої популярності. За допомогою дронів, наприклад, можна фотографувати всю територію та відтворювати більші споруди.

Рис.3.8 – Фотограмметрія

- Процедурне моделювання - відноситься до процесу тривимірного проектування алгоритмічного генерування моделей. На відміну від інших процесів, ці не створюються художником вручну. У цьому процесі об’єкти та сцени розробляються на основі визначених користувачем параметрів або правил. У різних пакетах моделювання середовища моделісти можуть створювати цілі ландшафти, змінюючи такі параметри, як діапазон висоти та щільність листя. Вони також можуть вибрати такі пейзажі, як прибережні райони, пустелі або альпійські райони. Техніка процедурного моделювання широко використовується в органічних об’єктах, таких як листя та дерева, де складність і варіації нескінченні. Ці моделі вкрай складно малювати від руки. Ці об’єкти можна налаштувати за допомогою різних параметрів, які можна редагувати. Наприклад, можна змінити щільність гілок, висоту стовбурів дерев, а також завитки та кути відповідно до потреб.

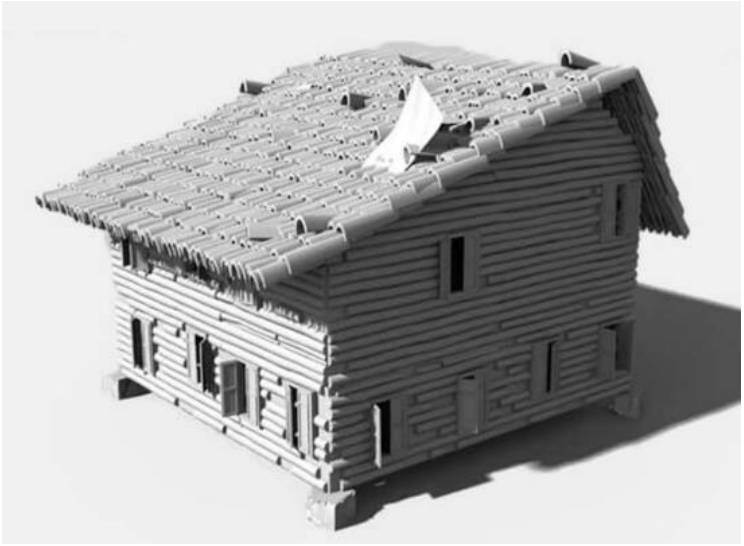

Рис.3.9 – Процедурне моделювання

- Моделювання на основі зображень. У моделюванні на основі зображень 3D-об’єкти виводяться алгоритмічно з набору 2D-зображень, які є статичними за своєю природою. Цей тип техніки використовується у випадках, коли модельєр стикається з бюджетними або часовими обмеженнями та не може розробити повністю реалізовані 3D-зображення. Це одна з найбільш часто використовуваних технік у кіноіндустрії. З роками моделювання на основі зображень стає все більш популярним в індустрії розваг.

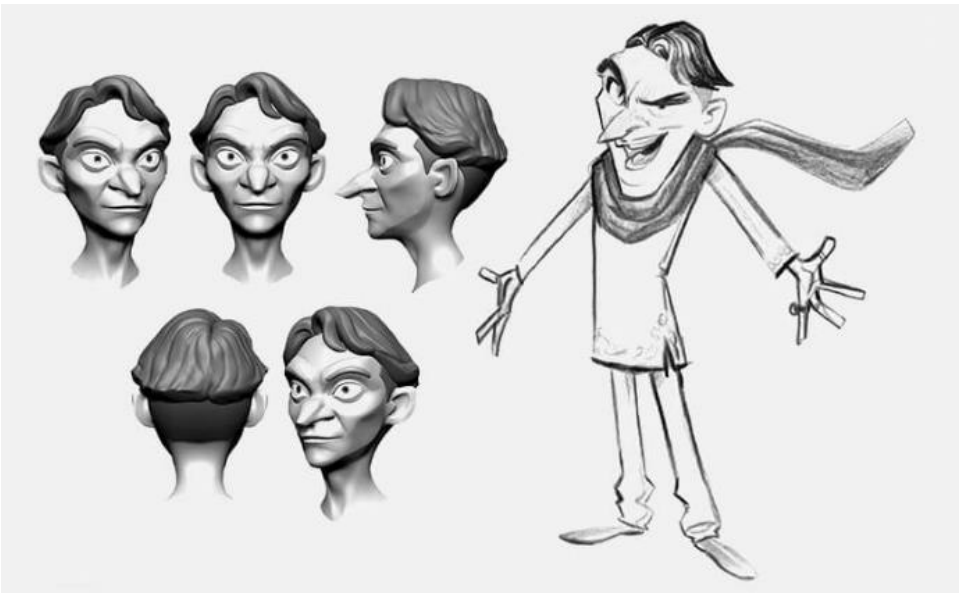

Рис.3.10 – Моделювання на основі зображень

- Моделювання поверхні (Surface modeling). Моделювання поверхні допомагає створити 3D-сплайн. Процес передбачає включення 2D-сплайна, і він відрізняється від NURBS. Цей метод в основному використовується для створення

органічних 3D-моделей у фільмах. Він пропонує достатню кількість гнучкості для модельєрів. Вони можуть легко створити тривимірне зображення з різними вимогами, використовуючи криві або поверхні.

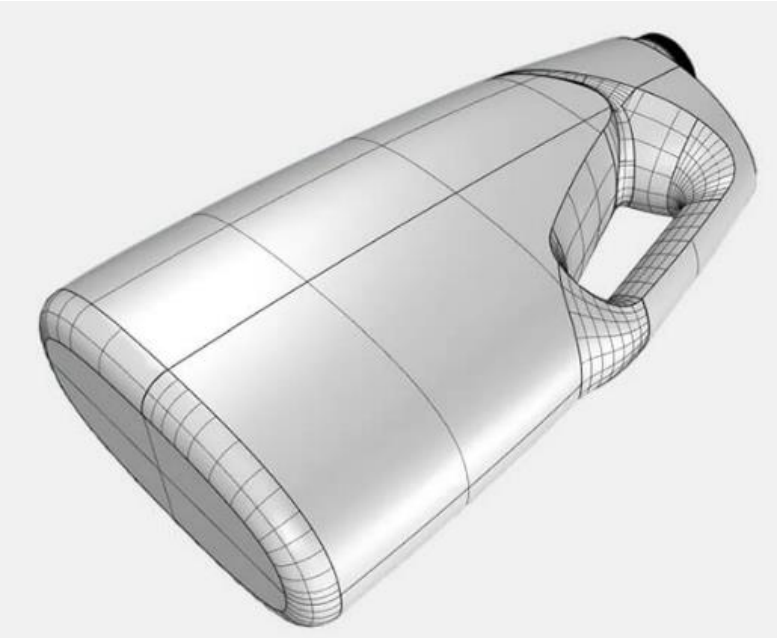

Рис.3.11 – Моделювання поверхні

- Логічне (булеве) моделювання. У цій техніці тривимірного моделювання форма об’єкта створюється за допомогою двох об’єктів. Геометрія створюється шляхом об’єднання двох об’єктів або вирізання одного з іншого. Його також можна створити з негативним простором перетину. Логічне моделювання допомагає створювати тривимірні форми, для створення яких за допомогою інших методів потрібен час. Наприклад, зігнуті та круглі форми можна комбінувати з твердими або квадратними, щоб створити нову форму. Однак ця техніка не настільки популярна в індустрії розваг.

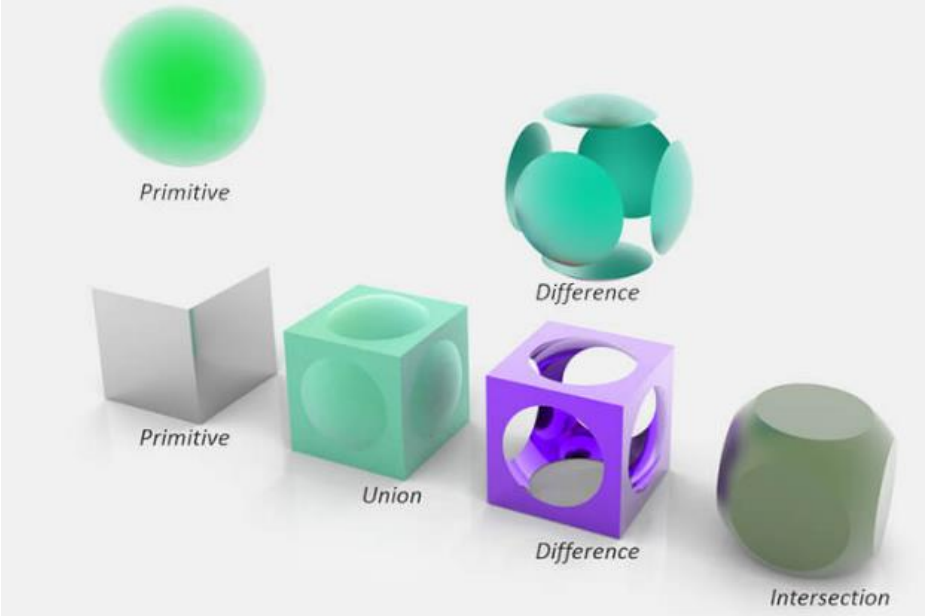

Рис.3.12 – Логічне (булеве) моделювання

- Лазерне сканування - було введено як метод 3D-моделювання завдяки вираженому прогресу цієї технології. Тут лазер використовується для сканування реального об’єкта та вимірювання його геометрії для створення 3D-моделі. Процес простий і швидкий, і не вимагає фізичного дотику до об’єктів для отримання точних розмірів. Після вимірювань можна створити цифрову модель за допомогою програмного забезпечення для 3D-зображень.

Рис.3.13 – Лазерне сканування

Отримані 3D-моделі можна вдосконалювати, оптимізувати та інтегрувати в різні програми, включаючи ігри, фільми, візуалізацію архітектури, візуалізацію

продуктів тощо. Незалежно від того, чи йдеться про створення реалістичних персонажів, великого середовища чи складного реквізиту, 3D-моделювання надає засоби для втілення уяви в цифровій сфері.

У Unreal Engine 3D-моделювання має вирішальне значення для створення віртуальних світів, персонажів, об'єктів і активів, які заповнюють ігрове середовище. Це дозволяє розробникам і художникам втілювати свої творчі ідеї в життя, проектуючи та створюючи детальний інтерактивний 3D-контент.

Unreal Engine надає ряд інструментів і функцій, спеціально розроблених для студій 3D-моделювання. До них належать потужні редактори моделювання, можливості візуалізації в реальному часі, системи матеріалів і текстур, інструменти анімації та моделювання фізики. Механізм підтримує галузеві стандартні формати файлів для імпорту та експорту 3D-моделей, дозволяючи бездоганну інтеграцію із зовнішнім програмним забезпеченням для моделювання та бібліотеками ресурсів.

Щоб ефективно використовувати 3D-моделювання в Unreal Engine, важливо розуміти основні концепції конвеєра створення вмісту рушія, такі як дизайн рівнів, управління активами та методи оптимізації. Крім того, знання матеріалів, шейдерів, освітлення та ефектів постобробки є життєво важливим для створення візуально привабливих і реалістичних сцен.

Маючи тверде розуміння 3D-моделювання в Unreal Engine, розробники та художники розкривають свій творчий потенціал і створюють захоплюючі віртуальні світи, які розширюють межі візуальної точності та інтерактивності. Це дозволяє створювати захоплюючі ігрові враження, кінематографічні послідовності, архітектурні візуалізації, програми віртуальної реальності та багато іншого.

3.2 Функції тривимірного моделювання Unreal Engine

Unreal Engine пропонує повні функції та інструменти для 3D-моделювання, що дозволяє розробникам і художникам створювати приголомшливі віртуальні середовища з високою деталізацією. Ось деякі з ключових особливостей можливостей 3D-моделювання Unreal Engine:

- потужні редактори моделювання: Unreal Engine надає інтуїтивно зрозумілі та універсальні редактори моделювання, які дозволяють користувачам створювати, змінювати та керувати 3D-геометрією. Ці редактори пропонують ряд інструментів моделювання, таких як екструзія, фаска, логічні операції тощо, що забезпечує точний контроль над створенням і модифікацією об’єктів;

- візуалізація в реальному часі: можливості відтворення в реальному часі Unreal Engine є видатною функцією. У ньому використовуються передові методи, такі як фізичне відтворення (PBR), динамічне освітлення та глобальне освітлення, що забезпечує візуально приголомшливу та реалістичну графіку. Це дозволяє художникам бачити остаточний вигляд своїх 3D-моделей у реальному часі, що робить процес ітерації більш ефективним;

- системи матеріалів і текстур: Unreal Engine надає потужну систему матеріалів і текстур, яка дозволяє художникам створювати складні та реалістичні поверхні для своїх 3D-моделей. Завдяки підтримці процедурних текстур, карт нормалей, карт висот тощо художники можуть досягти високого рівня деталізації та реалістичності своїх матеріалів;

- інструменти анімації: Unreal Engine містить надійну систему анімації, яка дозволяє створювати складні анімації персонажів, анімації об’єктів та інтерактивні послідовності. Він підтримує анімацію ключових кадрів, скелетну анімацію, змішані простори, анімації blueprints тощо, надаючи художникам гнучкість, щоб оживити свої 3D-моделі за допомогою динамічної та плавної анімації;

- симуляція фізики: Unreal Engine пропонує можливості симуляції фізики, що забезпечує реалістичну взаємодію між об’єктами та персонажами в ігровому світі. Художники можуть визначати такі фізичні властивості, як маса, сила тяжіння, тертя та поведінка при зіткненні, що сприяє зануренню та правдоподібності рухів і взаємодії 3D-моделей;

- імпорт ресурсів і сумісність: Unreal Engine підтримує широкий спектр галузевих стандартних форматів файлів для імпорту 3D-моделей, текстур і анімації. Ця сумісність дозволяє художникам використовувати свої наявні активи,

створені в зовнішньому програмному забезпеченні моделювання, і легко інтегрувати їх у середовище Unreal Engine;

Ці функції, серед іншого, роблять Unreal Engine потужним інструментом для 3D-моделювання та надають художникам і розробникам необхідні інструменти для створення візуально приголомшливих і захоплюючих вражень у своїх іграх та інтерактивних проектах

3.3 Загальна схема створення 3D-моделі в Unreal Engine

Створення 3D-моделей у Unreal Engine передбачає низку кроків, які дозволяють художникам і розробникам втілити свої ідеї в життя. Ось загальний огляд процесу:

1) Планування та концептуалізація. Перш ніж занурюватися в 3D-моделювання, важливо мати чітке бачення та концепцію моделі, яку необхідно створити. Створення ескізів ідей і збір опорних зображень можуть допомогти візуалізувати кінцевий результат;

2) Створення моделі: Unreal Engine надає кілька варіантів створення 3D-моделей. Можна почати з нуля, використовуючи вбудовані інструменти моделювання, або імпортувати вже існуючі моделі, створені у зовнішньому програмному забезпеченні, наприклад Blender, Maya або 3ds Max. Unreal Engine підтримує галузеві стандартні формати файлів, забезпечуючи бездоганну інтеграцію активів.

3) Маніпуляції з геометрією: коли є базова модель, можна вдосконалювати та маніпулювати її геометрією в Unreal Engine. Механізм пропонує ряд інструментів моделювання, таких як екструзія, масштабування, обертання та маніпулювання вершинами, щоб змінити форму та структуру моделі;

4) UV-відображення та текстурування: UV-відображення передбачає розгортання поверхні 3D-моделі на 2D-площину для підготовки її до текстурування. Unreal Engine надає надійний UV-редактор, який дозволяє ефективно створювати та змінювати UV-макети. Після UV-відображення можна застосовувати текстури та матеріали до моделі, визначаючи її зовнішній вигляд і властивості поверхні;

5) Оснащення та анімація: якщо для 3D-моделі потрібна анімація, можна створити її в системі анімації Unreal Engine. Оснащення передбачає створення скелета або серії з'єднань, які контролюють рух моделі. Після налаштування можна анімувати модель за допомогою ключових кадрів, скелетної анімації або методів процедурної анімації;

6) Імпорт та оптимізація: після створення та завершення 3D-моделі можна імпортувати її в Unreal Engine для використання у грі чи інтерактивному проекті. Можна оптимізувати геометрію, текстури та матеріали моделі під час процесу імпорту, щоб забезпечити ефективну роботу без шкоди для візуальної якості;

7) Тестування та ітерація: після імпорту, тестування та ітерація 3D-моделі в рушію має вирішальне значення. Можливості візуалізації в реальному часі Unreal Engine дозволяють побачити модель у дії та налаштувати її за потреби. Цей ітеративний процес гарантує бездоганну інтеграцію моделі в проект і відповідність бажаним візуальним і технічним стандартам

3.4 Створення ігрового персонажа

Для створення ігрового персонажа є декілька шляхів:

- створити модель в іншому редакторі, наприклад Blender чи Maya і потім імпортувати її в Unreal Engine для подальшої обробки чи анімації;
- використати вже готові активи (в суспільстві Unreal Engine велика база вже готових різноманітних моделей)
- скористатися інструментами Unreal Engine чи іншими продуктами Epic Games, зав'язаними на подальше використання в UE.

Для створення власного ігрового персонажа я скористався Metahuman Creator, програмою створення персонажів для відеоігор від Unreal Engine.

Початком для Metahuman Creator вважається 2017 рік. В цьому році розробником програми Epic Games на конференції GDC був представлений прототип майбутнього редактора. У процесі розробки прототипу Epic Games була задіяна сербська студія 3Lateral, яка вже мала досвід анімації в Devil May Cry 5 та GTA 5. Основою редактора є процедурна генерація нових осіб. Для її створення

використовувалися скани облич реальних людей: датчики зондували обличчя з точністю до дрібних пір шкіри, збираючи отриману інформацію в бази даних.

Першим кроком при створенні персонажа був вибір шаблону, з якого і почалося створення героя. Тож серед шаблонів людей різного віку, чоловіків і жінок було вибрано аватар. В Metahuman немає чіткої межі між чоловічими та жіночими персонажами. Функціонал у них загальний. Наприклад, чоловікам і жінкам будуть доступні як опції мейк-апу, так і вибір вусів з бородою.

Рис.3.14– Шаблон героя

Після вибору шаблону треба було детально попрацювати над обличчям, волоссям та тілом. Я почав з обличчя. Перш за все я попрацював з атрибутом “Блендер” - перетягнув в блендер декілька інших пресетів. І потім за допомогою курсору та перетягуванню пресетів я відредагував очі, щоки, ніс, лоб, губи та підборіддя персонажу.

Рис.3.15 – Редагування в режимі Blend

Далі настала черга попрацювати зі шкірою. Програма дозволяє змінювати колір шкіри, її контрастність, фактуру та шорсткість. Також можна додати ластовиння (вкладка freckles). За допомогою вкладки accents я додав шкірі насиченості обличчя і освітлив її.

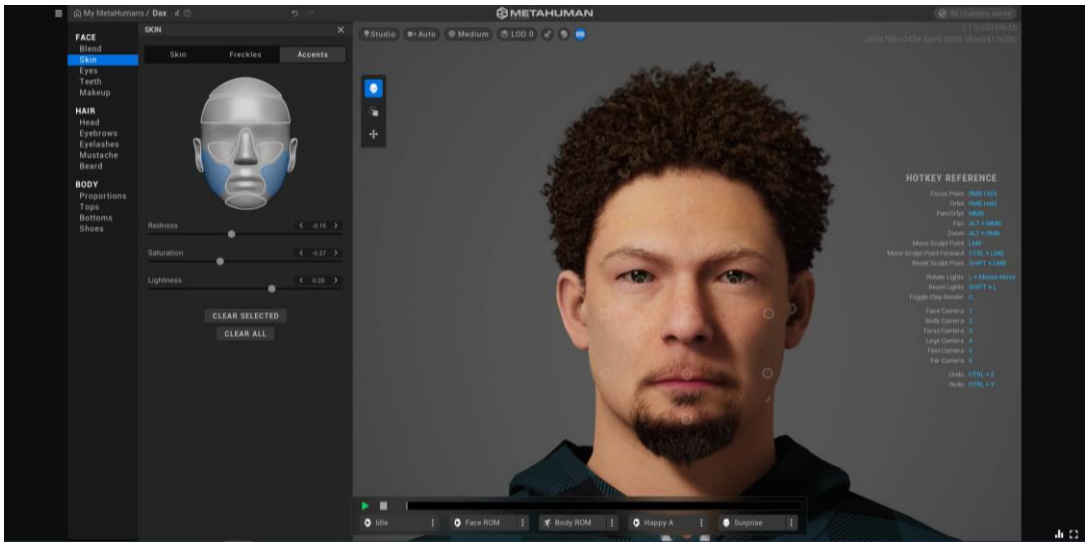

Рис.3.16 – Герой після маніпуляцій зі шкірою

Для редагування очей є атрибут Eyes. Тут можна виконувати різні маніпуляції над райдужними оболонками та склерами. Можна вибрати зовсім нові очі з наявних пресетів. Можна редагувати очі шаблону - колірний баланс очей, розмір райдужної оболонки, темряву лімба рогівки і т.д. В вкладці Склера можна редагувати відтінок очного яблука, яскравість, васкуляризацію та розташування вен. Можна навіть встановити різні параметри для різних очей - наприклад, ліве буде червоніше, ніж праве. Колір очей теж можна змінювати окремо. Таким чином,

можна добитися гетерохромії для персонажа. Я вирішив своєму герою не робити цього.

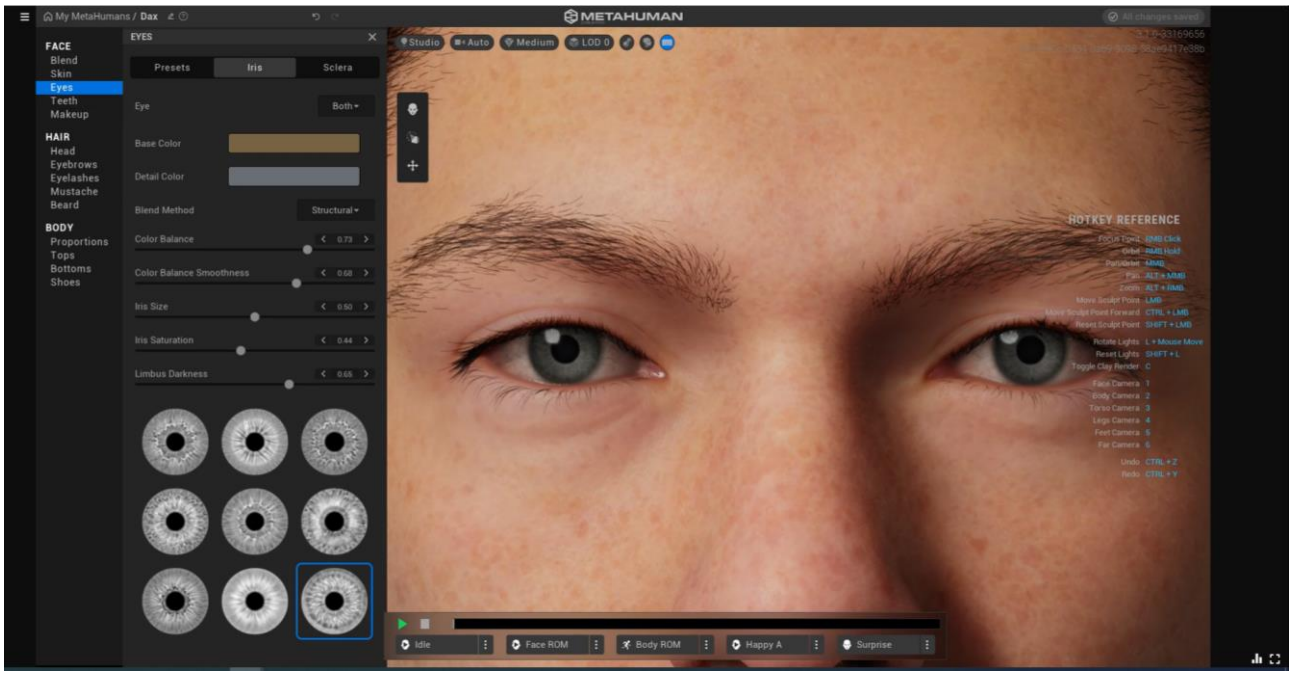

Рис.3.17 – Режим редагування очей

В розділі Teeth відбувається редагування зубів. Тут можна експериментувати з прикусом та розміром зубів. Змінювати колір язика та ясен. Краще бачити результат змін допомагає повзунок Jaw Open.

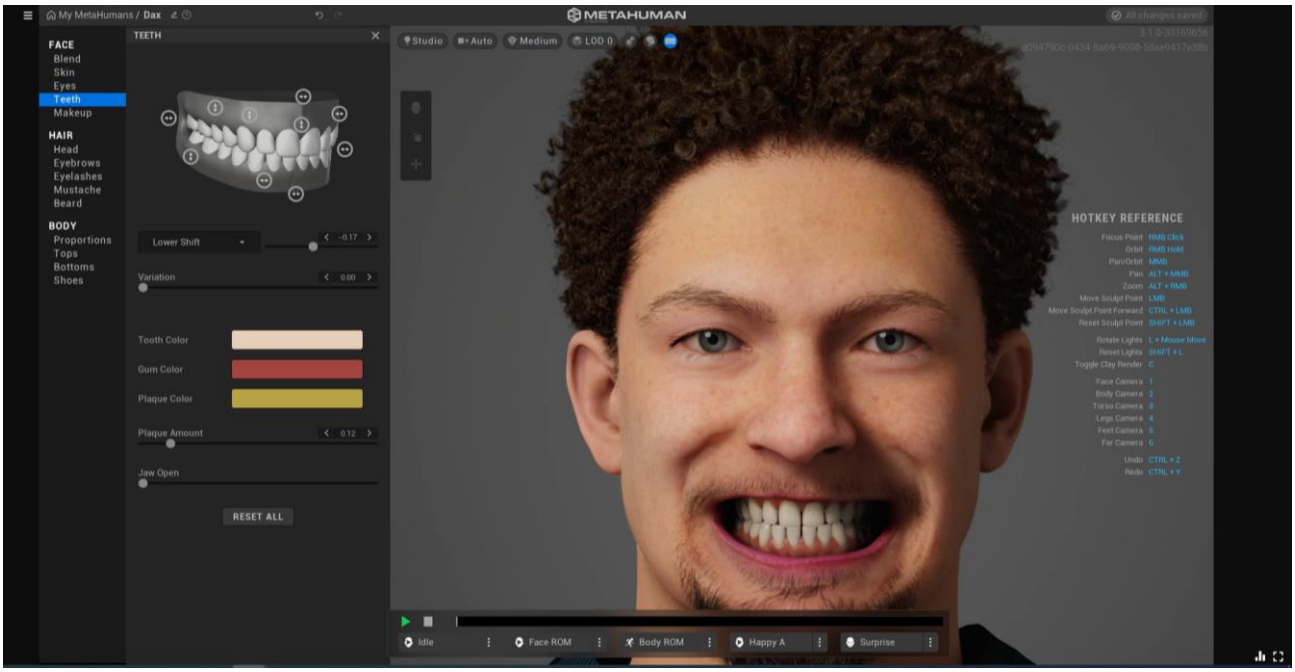

Рис.3.18 – Режим редагування зубів

Наступним кроком була робота з волоссям. В розділі з волоссям можна редагувати зачіску, брови, вії, вуса та бороду. Для кожного з цих атрибутів можна

міняти колір волосся, задавати різну їх жорсткість, густину і т.д. Також є опція «волосся з сивиною». Я вирішив зробити героя з зачіскою “ірокез”, також додав сивини.

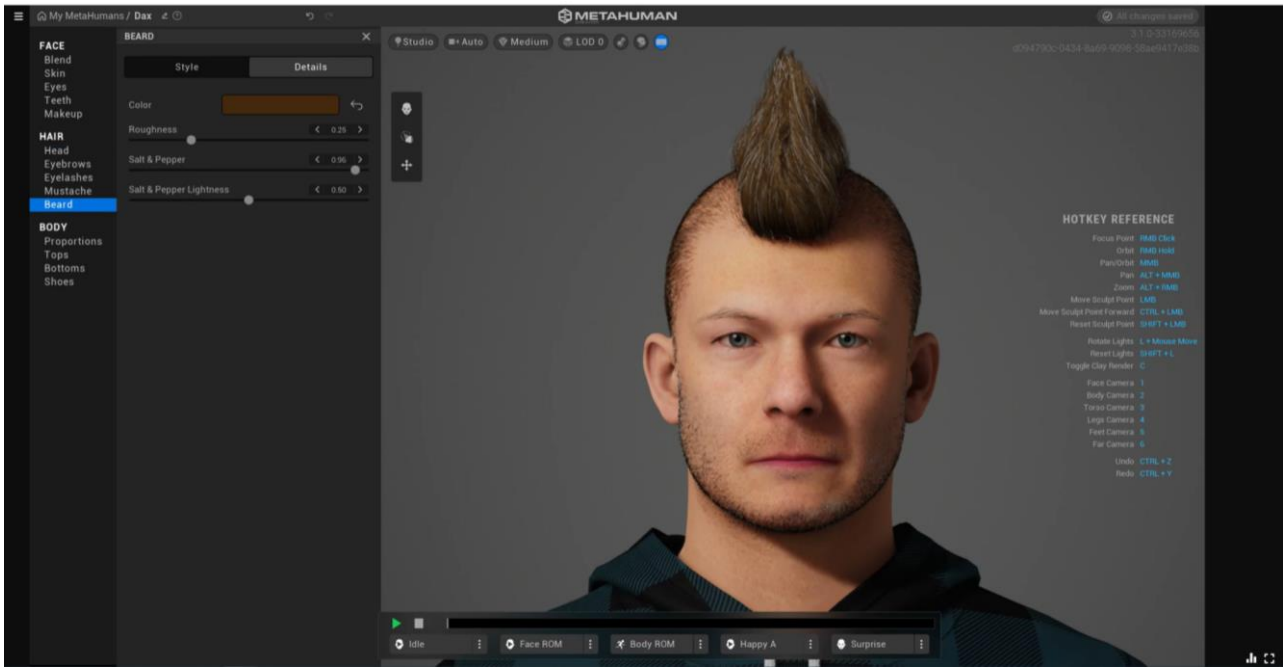

Рис.3.19 – Режим редагування волосся

Далі настала черга роботи з тілом. В режимі роботи з тілом пропрацьовується зріст, статура, пропорції голови і інших частин тіла, одяг та взуття.

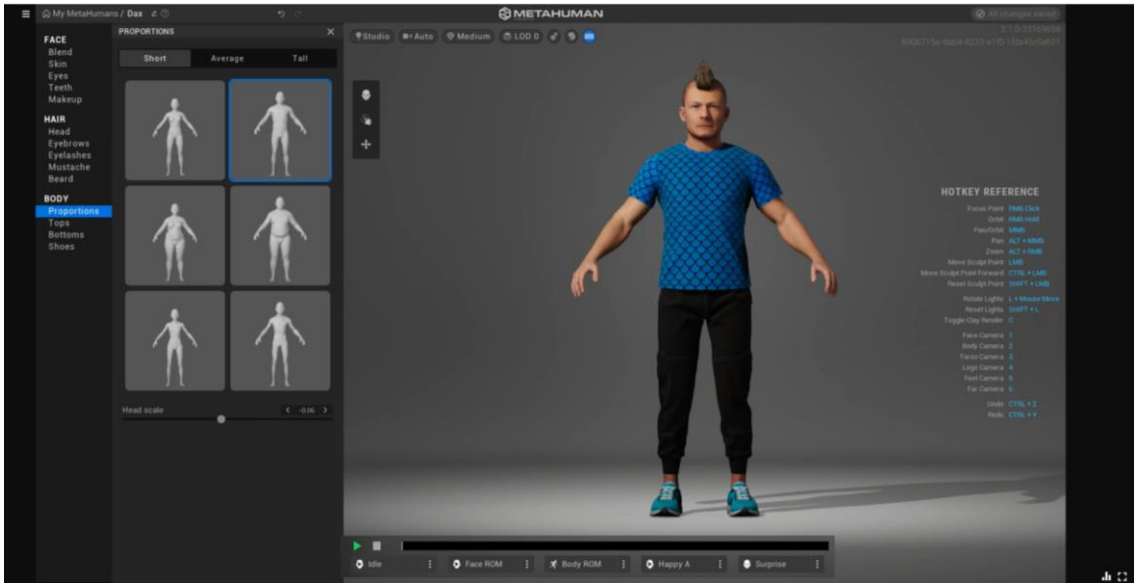

Рис.3.20 – Режим редагування тіла та одягу

Тепер, коли основа персонажу була готова, настав час пропрацювати детальніше більш тонкі риси обличчя. Для цього існують режими Move та Sculpt.

В режимі Move вносяться тонкі зміни вух, носа, губ, підборіддя, посадки глаз, посадки очей. У режимі Sculpt це робиться на ще більш тонкому та детальному рівні.

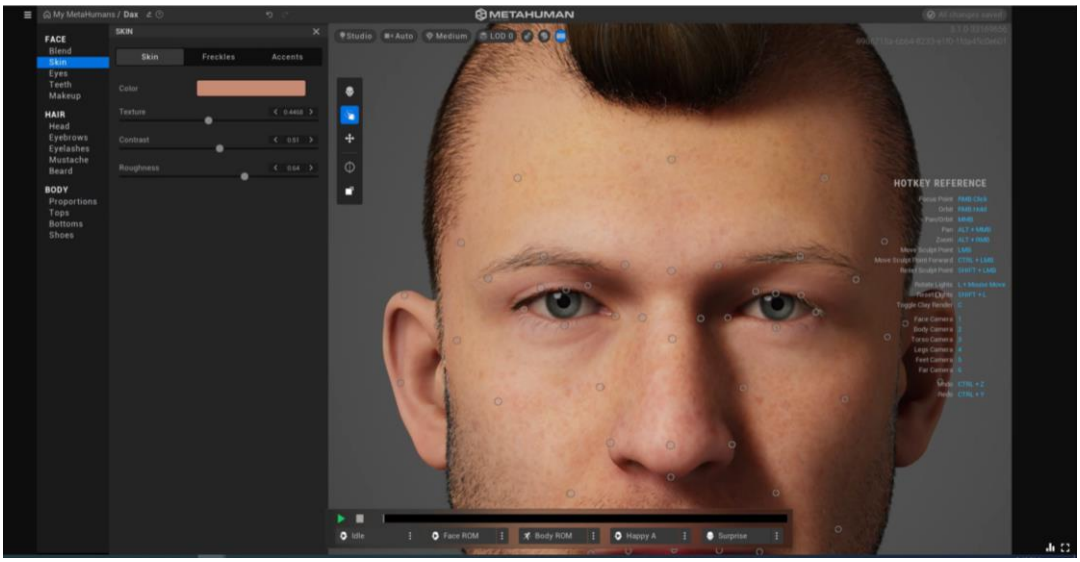

Рис.3.21 – Редагування в режимі Sculpt

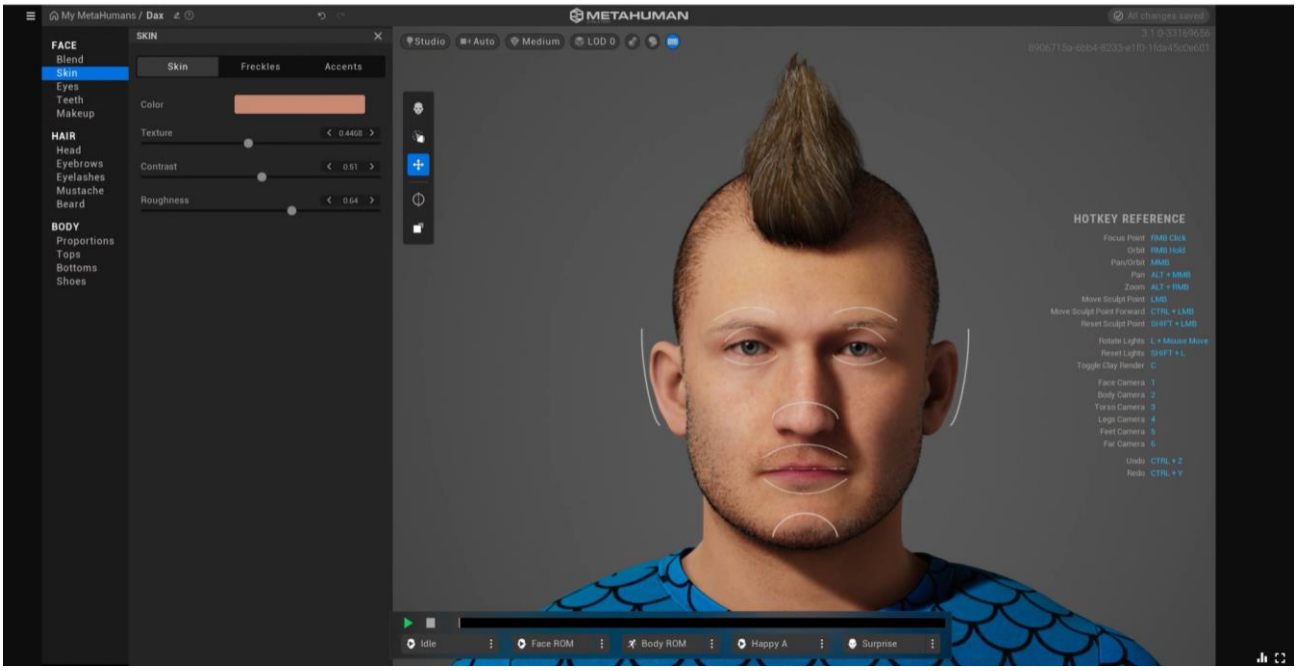

Рис.3.22 – Редагування в режимі Move

При створенні персонажа, зручною опцією є можливість розмістити його в різні умови, з різним освітлення, щоб побачити як буде виглядати персонаж загалом та його риси при різному освітленні і т.д. Це можуть бути:

- Indoor - у приміщенні;
- Silhouette - від героя залишається лише силует;
- Split – розщеплення: освітлена лише половина обличчя;
- Fireside - обличчя начебто підсвічується вогнем знизу;

- Moonligt - обличчя у місячному світлі;
- Tungsten - вольфрамовий режим;
- Portrait – портрет;
- Red lantern – червоний ліхтар;
- Rooftop - людина на даху та відповідне освітлення;
- Downtown night - у світлі вогнів великого міста;
- Underpass night – ніч у підземці.

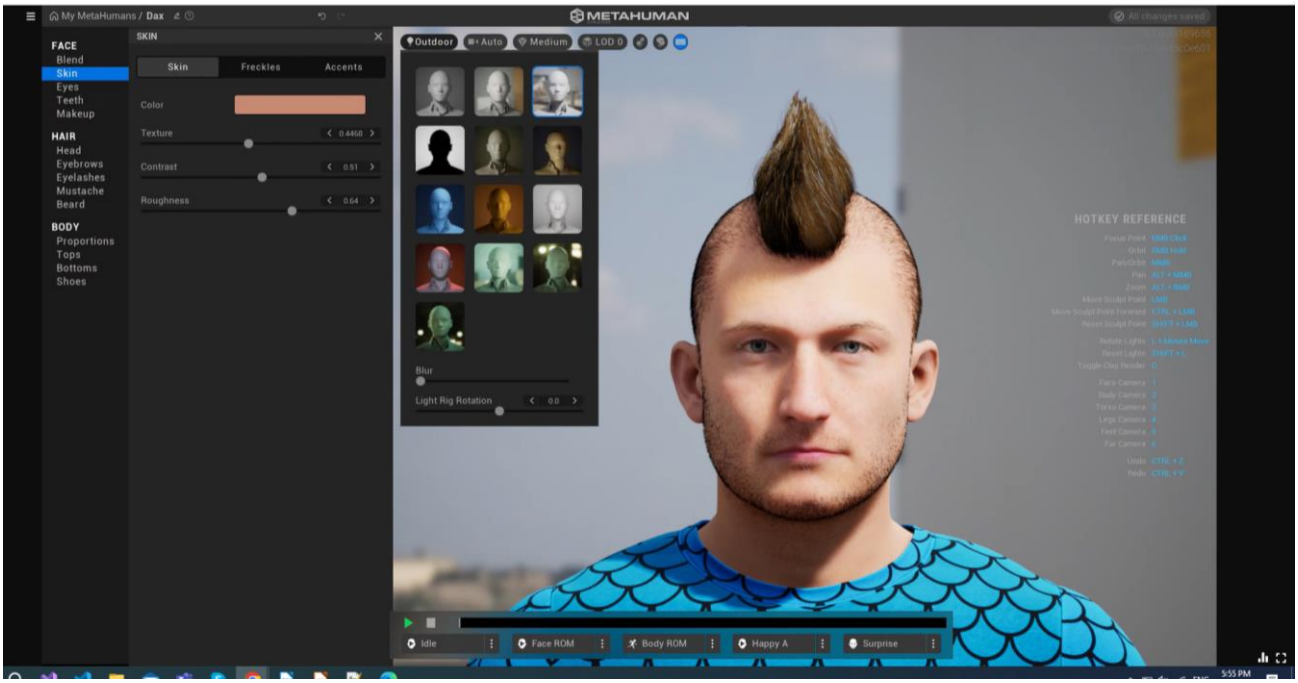

Рис.3.23 – Персонаж в режимі освітлення на вулиці

Опція Toggle Clay Material ніби підсвічує складки та зморшки обличчя. Hide Hair прибирає з обличчя весь волосяний покрив, щоб можна було максимально детально опрацьовувати його риси.

Камеру можна фокусувати на обличчі, переводити її на торс, ноги, ступні, можна брати різну крупність плану. Також можна змінювати якість візуалізації та рівень деталізації.

Рисю3.24 – Персонаж при низькому рівні деталізації

.

РОЗДІЛ 4. СФЕРИ ВИКОРИСТАННЯ UNREAL ENGINE

Завдяки новим функціям, таким як швидкий імпорт даних, blueprints, співпраця в реальному часі, розробка на багатьох платформах та інше, Unreal Engine має різні варіанти використання в багатьох галузях.

4.1 Ігрова індустрія.

Еволюція відеоігор, від 2D до 3D, а тепер ігор на основі блокчейну, є результатом нових технологій і технологічних тенденцій. Unreal Engine також представляє комплексні тенденції, технології та інструменти для виведення ігор на наступний рівень для підтримки таких великих технологій, як Metaverse.

Революційний вплив Unreal Engine на сучасні ігри незаперечний. Ігри, які використовують цю технологію, можуть створювати неймовірно реалістичну графіку, анімацію та ігровий процес, і роблять це з блискавичною швидкістю.

Це призвело до різкого збільшення кількості людей, які грають у відеоігри в усьому світі: згідно зі звітом про світовий ринок ігор Newzoo за 2019 рік, у 2018 році було 2 мільярди геймерів (проти 1 мільярда лише п’ять років тому). Сама галузь також зросла, згідно зі звітом Statista Global Digital Game Market Size Statistics for 2018 & 2022, глобальні продажі досягли 137 мільярдів доларів лише минулого року.

Unreal Engine також значно вплинув на саму розробку ігор: він дозволяє розробникам з усього світу створювати захоплюючі враження, не маючи доступу до мов програмування, таких як C++ або C#, або знання про них, натомість вони можуть зосередитися виключно на розробці контенту для їхніх проектів, одночасно використовуючи потужні можливості рушія Unreal під ними. Ігри, створені на Unreal Engine, демонструють універсальність і можливості цього надійного механізму розробки ігор. Творці можуть зосередитися на розробці захоплюючого контенту, використовуючи потужні можливості Unreal Engine, плавно інтегровані під поверхнею. Ця демократизація розробки ігор започаткувала нову еру, дозволяючи різноманітним розробникам втілювати свої бачення в життя.

Упродовж останніх років серед великих ігрових студій помітна тенденція до відмови від рушіїв власного виробництва. Наприклад, CD Projekt RED відправила в архіви REDengine, а Cyberpunk 2077: Phantom Liberty стала останнім проєктом із застосуванням цієї технології. Respawn Entertainment вирішила не використовувати Frostbite під час створення Apex Legends та дилогії Star Wars Jedi. Водночас S.T.A.L.K.E.R. 2 будується без умовного послідовника X-Ray Engine. Всі вони перейшли на Unreal Engine.

Unreal Engine все більше і більше наближається до того, щоб стати галузевим стандартом. Epic Games хоче створити потужний рушій з дуже багатим функціоналом та високою зручністю завдяки Blueprints і відкритому коду. Компанія постійно реалізовує технологічні рішення, які мають спростити та здешевити розробку. А ще - підтримує зв'язок зі своїми клієнтами.

4.2 ЗМІ та кіно

Інтеграція Unreal Engine у кіновиробництво стала революційною. Від попередньої візуалізації до постпродакшну кінематографісти використовують можливості Unreal Engine 5. Ось деякі способи використання UE5 у фільмах:

1) Створення віртуального світу. Unreal Engine відмінно справляється зі створенням захоплюючих віртуальних середовищ. Режисери можуть проектувати та візуалізувати складні ландшафти, чужі планети чи історичні події з надзвичайною деталізацією. Це не тільки економить час і ресурси в порівнянні з традиційною конструкцією наборів, але й забезпечує безпрецедентну творчу свободу. Незалежно від того, чи розгортається історія у вікторіанському особняку, храмі майя в джунглях чи витонченому інтер'єрі космічного корабля, UE5 дає можливість творцям оживити ці світи з кінематографічною точністю;

2) Змішана реальність (MR). Одне з найбільш новаторських застосувань Unreal Engine у кіновіробництві – це сфера змішаної реальності. Поєднуючи фізичні декорації та акторів із цифровими елементами в режимі реального часу, режисери можуть створювати візуально приголомшливі сцени. Ця інтеграція покращує продуктивність актора та забезпечує більш захоплюючий досвід для

глядачів. Можливості візуалізації в реальному часі UE5 роблять MR більш доступним і ефективним, відкриваючи нові можливості для оповідання.

3) Створення персонажів. Unreal Engine не обмежується створенням віртуальних середовищ; він також чудовий у створенні персонажів. UE5 підтримує створення реалістичних персонажів, як реальних, так і нереальних. Від цифрового відтворення історичних персонажів до створення вигаданих істот UE5 пропонує інструменти для проектування, анімації та інтеграції персонажів у кінематографічний наратив. Це не тільки покращує візуальну привабливість фільмів, але й дозволяє відмовитися від застарілих методів оповідання.

Вплив Unreal Engine на кіноіндустрію очевидний у зростаючому списку фільмів і телешоу, які пропонують його можливості. Фільми-блокбастери, такі як «Мандалорець» і «Вартові Галактики. 2” використовували UE5 для створення складних візуальних ефектів у реальному часі. Вони створили приголомшливі та неземні пейзажі. Крім того, такі телевізійні серіали, як «Westworld», використовували гнучкість механізму для візуалізації складних футуристичних світів і бездоганного поєднання практичних і цифрових елементів. Слід очікувати, що буде з’являтися все більше і більше фільмів, створених за допомогою цього інструменту.

4.3. Архітектура

UE допомагає змінити спосіб візуалізації архітектури. Він пропонує найбільш інноваційні та привабливі презентаційні рішення. Способи використання UE в архітектурі:

1) Використання Unreal Engine для ефективного створення нерухомих рендерів. Unreal Engine дає реалістичні зображення. І що важливо, він дозволяє створювати багато зображень набагато швидше, ніж класичні рушії, які використовувалися в архітектурі.

2) Використання Unreal Engine для інтерактивних тривимірних турів із зануренням. Серед усіх інших переваг Unreal Engine пропонує новаторське рішення - захоплюючі 3D-тури на основі потокового передавання Pixel. Ця технологія дозволяє відтворювати інтерактивну графіку в реальному часі. Його

можна передавати безпосередньо на пристрої кінцевих користувачів. Таким чином, Unreal Engine дозволяє користувачам отримувати доступ і досліджувати захоплюючі тури нерухомістю. Це можна зробити з будь-якого гаджета з підключенням до Інтернету. За допомогою цієї технології архітектори та фахівці з нерухомості можуть демонструвати свої проекти з вражаючою реалістичністю та деталізацією. Більше того, потокова передача пікселів Unreal Engine гарантує, що глядачі можуть взаємодіяти з середовищем. Вони можуть «гуляти» по віртуальній території, оглядати благоустрій, вибирати квартири, змінювати інтер'єри. Також вони можуть змінювати погоду і час доби.

3) Unreal Engine забезпечує VR-досвід. Використання Unreal Engine дозволяє оптимізувати створення VR для нерухомості та архітектури. Це тому, що Unreal Engine пропонує готові шаблони та креслення VR. Це готові фрагменти коду, які пропонують швидку відправну точку для проектів віртуальної реальності. Але справа не тільки в швидкості. Візуальні елементи Unreal Engine відомі своєю винятковою реалістичністю. Це особливо важливо, коли йдеться про створення захоплюючих реалістичних середовищ VR. Висока якість візуального виведення відрізняє Unreal від інших двигунів, таких як Unity.

4) UE дозволяє багаторазово використовувати одну сцену. Однією з головних переваг використання Unreal Engine в архітектурі є можливість використання однієї сцени для створення різних видів візуальних елементів. Наприклад, існуюча сцена Unreal Engine може слугувати основою для створення потокової передачі Pixel і VR. Ця технічна універсальність робить використання Unreal Engine в архітектурі дуже ресурсозберігаючим.

4.4 Автомобілебудування та транспорт

Надзвичайні можливості Unreal Engine вплинули на автомобільну промисловість, яка продовжує розвиватися. Деякі з провідних світових автовиробників створюють широкий спектр вражаючих 3D-досвідів у реальному часі з його допомогою.

Вони створюються як у розробці транспортних засобів, так і як інструменти продажів і маркетингу, включаючи цифрові салони, лабораторії змішаної

реальності, кінематографічні враження та онлайн-3D-конфігуратори транспортних засобів, що передаються на будь-який пристрій. Усі ці досягнення завершуються покращенням автомобільного ланцюжка створення вартості.

Автовиробники, зокрема Audi, Chevrolet, Porsche, Volkswagen, звернулися до Unreal Engine, оскільки він забезпечує найкращі фотореалістичні зображення, доступні на будь-якій платформі 3D-рендерінгу та розробки в реальному часі.

Ось деякі способи використання автовиробниками інструментів 3D-рендерінгу в реальному часі Unreal Engine:

- при проектуванні свого останнього шоу-руму цифрової віртуальної реальності в реальному часі Audi прагнула надати клієнтам максимально реалістичний досвід. Дизайнери шоу-руму використовували наявні дані CAD з Unreal Engine. За словами керівника відділу процесів, даних і технологій Audi Business Innovation, ця технологія забезпечила тіні та освітлення в режимі реального часу, які вивели фотореалістичний вигляд на новий рівень і вперше зробили автомобіль справжнім;

- Volkswagen Sweden створив ефектний конфігуратор VR для Volkswagen Arteon за допомогою Unreal Engine, який дозволяє клієнтам взаємодіяти з фотореалістичним автомобілем у режимі реального часу за допомогою гарнітури VR. Клієнти можуть повністю налаштувати свої транспортні засоби та взаємодіяти з ними за допомогою цієї системи;

- BMW і MINI звернулися до Unreal Engine для своєї лабораторії змішаної реальності, яка використовується для проектування транспортних засобів. Вони поєднують віртуальну реальність із спеціальним апаратним забезпеченням, щоб запустити роботу в реальному часі. Unreal Engine забезпечує чудові візуальні ефекти зі світлом і тінню, надаючи дизайнерам і інженерам кращий вигляд і відчуття для клієнта на ранніх етапах процесу проектування. Це краще розуміння дизайну допомагає їм легко співпрацювати, що забезпечує більш гнучку розробку.

- Porsche об’єднався з Nvidia та Unreal Engine, щоб створити «Швидкість світла», який називають кінематографічним досвідом. Надзвичайним було те, що це був перший випадок, коли автомобіль — Porsche 911 Speedster Concept — був

відтворений за допомогою інтерактивного трасування променів у реальному часі за допомогою ігрового рушія. Платформа потокової передачі в реальному часі PureWeb підтримує 3D-програми, які використовують трасування променів.

Потужність, висока якість і простота використання пояснюють, чому Unreal Engine часто обирають автовиробники, які шукають нові та інноваційні способи розробки та продажу своїх продуктів. Unreal також дозволяє маркетологам перетворювати кінцеві САПР на цифрові брошури з фотореалістичними зображеннями, які можна розмістити в будь-якому місці, а також відеовмістом можна поділитися в Інтернеті та соціальних мережах.

А для транспортних засобів, які цікавляться популярними ігровими франшизами, технологія Unreal полегшує обмін 3D-моделями CAD з ними. Ігри, створені на Unreal Engine, які використовують моделі реальних автомобілів, включають «Forza Street», «Gravel», «MotoGP» тощо.

4.5 Перспективи застосування рушію в подальших поколіннях

Оскільки Unreal Engine продовжує розвиватися та розширювати свої можливості, його потенційні можливості застосування в різних галузях безмежні. Деякі з ключових сфер, де очікуються значне зростання та поява нових інновацій у найближчі роки, включають:

- метавсесвіт і віртуальна реальність. Із зростаючим інтересом до метавсесвіту та досвіду віртуальної реальності Unreal Engine має хороші можливості стати провідною платформою для створення захоплюючих взаємопов’язаних віртуальних світів. Здатність рушія створювати фотореалістичне середовище в реальному часі та підтримувати вдосконалене апаратне забезпечення VR і AR робить його ідеальним інструментом для розробників і творців, які хочуть створювати додатки метавсесвіту нового покоління;

- симуляція та цифрові близнюки. Можливості симуляції в реальному часі Unreal Engine роблять його потужним інструментом для створення цифрових двійників – віртуальних копій об’єктів, систем або процесів реального світу. Оскільки індустрія все більше використовує технологію цифрових двійників для таких додатків, як прогнозне технічне обслуговування, оптимізація та навчання,

Unreal Engine, ймовірно, відіграватиме значну роль у створенні високоякісних інтерактивних цифрових двійників у широкому діапазоні доменів;

- співпраця в реальному часі та віддалена робота. Пандемія COVID-19, а в Україні ще й війна прискорили перехід до віддаленої роботи та співпраці, підкресливши потребу в потужних інструментах у реальному часі, які можуть сприяти безперебійному спілкуванню та спільному створенню на відстані. Здатність Unreal Engine створювати спільні інтерактивні віртуальні середовища робить його перспективною платформою для розробки рішень для співпраці та віддаленої роботи нового покоління.

- конвергенція ігрової та інших індустрій. Оскільки межі між ігровою та іншими галузями продовжують стиратися, Unreal Engine має хороші можливості, щоб служити мостом між цими світами. Універсальність і крос-платформні можливості механізму роблять його ідеальним інструментом для створення інтерактивного досвіду, який охоплює кілька сфер, від ігор і розваг до навчання та тренувань тощо.

Дивлячись у майбутнє, Unreal Engine 5 і його революційні функції, такі як Nanite, Lumen і World Partition, обіцяють розпочати нову еру фотореалістичних, масштабних і захоплюючих вражень. Зростаюче впровадження рушія в галузях промисловості, від кіно і телебачення до архітектури, автомобілебудування та інших, демонструє його неймовірну універсальність і потенціал як інструменту для інновацій і трансформації.

Оскільки світ стає все більш цифровим і взаємопов’язаним, Unreal Engine буде відігравати центральну роль у формуванні майбутнього інтерактивних розваг, дизайну та співпраці. Unreal Engine пропонує потужну, доступну платформу, яка постійно розвивається, щоб втілювати ідеї в життя, як розробників ігор, режисерів, архітекторів так і творців у будь-якій іншій галузі та розширювати межі можливого в цифровій сфері.

.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи на тему "Створення графічних моделей на основі рушію Unreal Engine" було здійснено детальне дослідження процесу розробки високоякісних графічних моделей. Основним результатом цієї роботи стало створення моделі персонажа для ігор, що дозволило на практиці оцінити можливості та інструменти Unreal Engine. Отримані результати дозволяють зробити низку важливих висновків:

1) Розробка моделі персонажа:

У процесі роботи було створено високоякісну модель персонажа для ігор, які в подальшому можна буде використовувати в розробці ігор або інших візуалізаціях. Це дозволило детально вивчити можливості Unreal Engine.

2) Високий рівень деталізації та реалістичності:

Unreal Engine продемонстрував свої потужні можливості у створенні фотореалістичних графічних моделей. Використання передових технологій значно підвищило якість візуалізації моделі персонажа, забезпечуючи високу деталізацію та реалістичність.

3) Широкий спектр можливостей для розробників та дизайнерів:

Unreal Engine пропонує великий набір інструментів та функцій, які дозволяють реалізувати найскладніші творчі проекти. Система матеріалів, інструменти для створення ефектів постобробки, а також можливості для інтеграції з іншими програмами роблять Unreal Engine універсальним рішенням для створення графічних моделей.

4) Значущість в умовах дистанційної роботи:

В умовах пандемії COVID-19 та військових конфліктів, коли дистанційна робота стала необхідністю, Unreal Engine довів свою ефективність як інструмент для віддаленої співпраці. Вбудовані функції для роботи в команді та можливості для онлайн-співпраці сприяють успішному виконанню проектів незалежно від географічного розташування учасників.

5) Інноваційні підходи в різних галузях:

Unreal Engine знайшов широке застосування не лише у геймінгу, але й у багатьох інших галузях, таких як архітектурні візуалізації, медичні симуляції, кіновиробництво та віртуальна реальність. Його використання відкриває нові горизонти для застосування графічних моделей у різних сферах діяльності.

6) Практичні результати та прикладне значення:

Виконання практичної частини роботи продемонструвало, що Unreal Engine є ефективним інструментом для створення реалістичних графічних моделей. Створена модель персонажа підтвердила практичну цінність та широкі можливості цього рушія. Отримані навички та досвід мають велике значення для подальшої професійної діяльності в галузі цифрових технологій.

7) Перспективи подальших досліджень та розвитку:

Результати даної кваліфікаційної роботи вказують на великі перспективи для подальших досліджень у сфері використання Unreal Engine. Зокрема, це стосується розробки нових методів оптимізації графічних моделей, дослідження можливостей віртуальної та доповненої реальності, а також інтеграції з іншими передовими технологіями.

На основі проведеного дослідження можна зробити висновок, що Unreal Engine є потужним інструментом для створення графічних моделей, який надає розробникам та дизайнерам широкі можливості для реалізації їх творчих задумів. Використання цього рушія сприяє підвищенню якості та реалістичності графічних моделей, а також забезпечує ефективну співпрацю в умовах дистанційної роботи.

Ця кваліфікаційна робота стала важливим етапом у розумінні та освоєнні технологій, пов'язаних із створенням графічних моделей, і впевнений, що отримані знання та навички сприятимуть подальшому професійному розвитку у цій динамічній та цікавій галузі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кривко О. Не соромно запитати: як працює Unreal Engine. URL: https://skvot.io/uk/blog/ne-soromno-zapitati-yak-pracyuye-unreal-engine (дата звернення: 12.05.2024).

2. Кознов С. Розкриття потенціалу Unreal Engine 5. Реальне застосування (туторіал для дизайнерів). URL: https://ux.pub/sergeykoznov/rozkrittia-potientsialu-unreal-engine-5-riealnie-zastosuvannia-tutorial-dlia-dizainieriv-2g57 (дата звернення: 12.05.2024).

3. Кой О. Створення персонажа у Metahuman Avatar: робота з волоссям. Гайд і поради 3D-художниці. URL: https://gamedev.dou.ua/blogs/metahuman-avatar-creation-in-morphy-vision (дата звернення: 14.05.2024).

4. Степорук Н. Unity проти Unreal Engine - який рушій обрати для гри та чому. Зважуємо всі за та проти з розробниками. URL: https://gamedev.dou.ua/articles/unity-or-unreal-engine (дата звернення: 12.05.2024).

5. Oghterop W., Venter H. Unreal Engine 5 Character Creation, Animation, and Cinematics: Create Custom 3D Assets and Bring Them to Life in Unreal Engine 5 Using MetaHuman, Lumen, and Nanite. Packt Publishing, Limited, 2022. 608 с.

6. Larson M. Unreal Engine 5 Tutorial for Beginners: Getting Started. URL: https://www.kodeco.com/31800833-unreal-engine-5-tutorial-for-beginners-getting-started (дата звернення: 12.05.2024).

7. Augusto A. How Unreal Engine is changing the automotive industry completely. URL: https://andreaugustobr.medium.com/how-unreal-engine-is-changing-the-automotive-industry-completely-2b8f4ce80460 (дата звернення: 12.05.2024).

8. Lewis K. The Transformative Role of Unreal Engine in Modern Game Creation. URL: https://www.devx.com/gaming/the-transformative-role-of-unreal-engine-in-modern-game-creation (дата звернення: 12.05.2024).

9. Meena G. Unreal Engine and its Evolution. URL: https://externlabs.com/blogs/unreal-engine-and-its-evolution (дата звернення: 12.05.2024).

10. Royal M. Unreal Engine Guide. URL: https://github.com/mikeroyal/Unreal-Engine-Guide (дата звернення: 12.05.2024).

11. Phuc D. Why Unreal Engine 5? 8 Reasons To Use Unreal Engine 5 URL: https://animost.com/ideas-inspirations/why-unreal-engine-5 (дата звернення: 12.05.2024).
12. Mat-Madajczak M. How to become a MetaHuman URL: https://wttech.blog/blog/2023/how-to-become-metahuman (дата звернення: 14.05.2024).
13. Jocque Butler. Creating a Real-Time Cinematic Character in Unreal Engine 5. URL: https://discover.therookies.co/2022/07/26/creating-a-real-time-cinematic-character-in-unreal-engine-5 (дата звернення: 12.05.2024).
14. Unreal Engine 5.3 Documentation. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-3-documentation (дата звернення: 12.05.2024).
15. MetaHuman Documentation. URL: https://dev.epicgames.com/documentation/en-us/metahuman/metahuman-documentation (дата звернення: 12.05.2024).
16. Mastering Unreal Engine 3D Modeling: A Comprehensive Guide. URL: https://polydin.com/unreal-engine-3d-modeling (дата звернення: 12.05.2024).
17. 13 Types of 3D Modeling Techniques to Choose in 2024. URL: https://professional3dservices.com/blog/3d-modeling-techniques.html (дата звернення: 14.05.2024).
18. Unreal Engine 3D Modeling: a Step-by-Step Guide. URL: https://game-ace.com/blog/unreal-engine-3d-modeling-a-step-by-step-guide (дата звернення: 12.05.2024).
19. Unreal Engine. Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Unreal_Engine (дата звернення: 12.05.2024).
20. Kim Y. Design and Performance Evaluation of Soft 3D Models using Metaball in Unreal Engine 4. Journal of Mobile Multimedia. 2022. Vol. 18_6, 1795–1810.