ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ

УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка ігрової локації на базі платформи Unreal Engine 4»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Радіонова Ярослава Андрійовича

Науковий керівник
кандидат фізико-математичних наук _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем Спеціальність 122 «Комп'ютерні науки»

Затверджую

Зав. кафедри _____Пурський О.І.

«12» грудня 2022р.

Завдання

на випускну кваліфікаційну роботу студенту

Радіонову Ярославу Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи

«Розробка ігрової локації на базі платформи Unreal Engine 4»

Затверджена наказом ректора від *«09 » грудня 2022 р. № 3332*

2. Строк здачі студентом закінченої роботи *30 травня 2023 року*
3. Цільова установка та вихідні дані до роботи

Мета роботи: розробка графічної складової ігрового рівня та її демонстрація .

Об'єкт дослідження: процес створення 3D графіки для ігрового рівня.

Предмет дослідження: технології в системі ігрового рушія та програм для створення 3D графіки.

4. Перелік графічного матеріалу

_____

_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
2	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
3	Філімонова т.О.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ ПРОЦЕСУ СТВОРЕННЯ 3D ГРАФІКИ

1.1 Особливості побудови моделей, їхня будова та етапи створення

1.2 Теорія створення текстур та матеріалів для моделей. PBR текстури

1.3 Загальні відомості про технології рендерінгу та освітлення у ігровому рушії Unreal Engine 5

РОЗДІЛ 2. ОСНОВНА ТЕОРІЯ СТВОРЕННЯ ІГРОВОГО РІВНЯ З ТОЧКИ ЗОРУ ВІЗУАЛЬНОЇ СКЛАДОВОЇ

2.1 «Блокаут» та планування побудови рівня

2.2 Етапи створення локації

2.3 Постобробка

РОЗДІЛ 3. РОЗРОБКА 3D ГРАФІКИ ТА ЇЇ ВИКОРИСТАННЯ У ІГРОВОМУ РУШІЇ UNREAL ENGINE 5

3.1 Особливості створення моделей у пакеті 3D моделювання Blender

3.2 Особливості роботи в програмах для текстурування Substance Painter

3.3 Особливості роботи у ігровому рушії Unreal Engine 5

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

Додаток А

Додаток Б

Додаток В

Додаток Г

Додаток Д

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1.Аналітичне дослідження процесу створення 3D графіки*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2.Основна теорія створення ігрового рівня з точки зору візуальної складової*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3.Розробка 3D графіки та її використання у ігровому рушії Unreal Engine 5*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 - 01.06.2023*	*31.05.2023 - 01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи

Пурський О.І.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Радіонов Я.А.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи

Випускна кваліфікаційна робота «Розробка ігрової локації на базі платформи Unreal Engine 4» присвячена створенню ігрового локації на базі рушія Unreal Engine 5 шляхом розробки і додавання 3D моделей у середовище ігрового рівня. В результаті створена композиція, що складається з 3D моделей і яку у вигляді ігрової локації можна використати в подальшому у ігровому продукті (комп'ютерній грі).

Випускна кваліфікаційна робота відповідає всім вимогам до випускних кваліфікаційних робіт. Всі поставлені завдання виконані. Випускна кваліфікаційна робота може бути допущена до захисту.

Керівник випускної кваліфікаційної роботи _____ 30.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми _____ Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У випускній кваліфікаційній роботі було теоретично описано основні методи, принципи створення 3D моделей та формування композиції у вигляді ігрової локації на базі рушія Unreal Engine 5. Був створений комплекс 3D моделей, які були використані у якості об’єктів ігрового рівня, а також було створене середовище всередині ігрового рушія для їхнього розміщення та демонстрації.

Ключові слова: Рушій, меш, модель, об’єкт, текстура, карта, нормаль, рендер, геймплей.

Anotation

In the graduation qualification work, the main methods, principles of creating 3D models and forming a composition in the form of a game location based on Unreal Engine 5 were theoretically described. A complex of 3D models was created, which were used as objects of the game level, and an environment was also created inside the game engine to host and display them.

Keywords: driver, mesh, model, object, texture, map, normal, render, gameplay.

ЗМІСТ

ВСТУП

Випускна кваліфікаційна робота є практичним застосуванням знань про створення 3D графіки, реалізації композиції у вигляді «ігрового рівня» - збірці наявних моделей у середовищі ігрового рушія, яку можна використовувати як елемент загального продукту, такого як, наприклад, комп'ютерна гра.

Актуальність і практична значущість обраної теми дипломної роботи формує реалії сьогодення: комп'ютерні ігри стали невід'ємною частиною індустрії розваг, капіталізація яких росте з кожним днем все більше і більше, обганяючи навіть дорогі у виробництві фільми та серіали.

Особливістю вирішення поставленого питання є розв'язання проблематики обмеженості потужностей більшості машин, якими користуються гравці: не кожен комп'ютер здатний відтворювати ігровий рівень якісно і швидко, оскільки на це впливає фінансове становище гравця – чим дорожче комп'ютер, тим складніші задачі він здатний вирішувати. Моя задача, як спеціаліста, створити графіку для ігрового рівня таким чином, щоб був збережений баланс між продуктивністю та якістю зображення.

Метою роботи є розробка графічної складової ігрового рівня та її демонстрація у відповідному для цього графічному середовищі.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Підібрати відповідні матеріали, на які потрібно опиратися при створенні ігрового рівня;
2. Пропрацювати основний концепт ігрового рівня, продумати місцеположення тих чи інших об'єктів у сцені;
3. Створити основний ландшафт рівня (оскільки сцена буде на відкритій місцевості);
4. Створити моделі, якими буде наповнена сцена;
5. Створити текстури для них;

6. Пропрацювати освітлення у сцені, основні елементи керування для демонстрації та налаштувати постобробку;
7. Зібрати усі наявні елементи в одну готову композицію.

Об’єктом дослідження є процес створення 3D графіки для ігрового рівня.

Предметом дослідження є технології в системі ігрового рушія та програм для створення 3D графіки. З **методів дослідження** варто зазначити програмний пакет 3D моделювання Blender, програма для текстурування Substance Painter, програма для створення текстурних карт Marmoset Toolbag, а також ігровий рушій Unreal Engine 5 (ПРИМІТКА: при роботі була змінена версія з 4 на 5, далі у роботі буде використовуватися назва «Unreal Engine 5»). Усі перелічені програми мають інтегровані методи досліджень, які допомагають у вирішенні поставленого питання.

Головним представником нетрадиційних джерел інформації є відеохостинг YouTube. Наявні відеоматеріали по темі моделювання 3D моделей дозволяють швидко і ефективно адаптувати наявні знання під практичне застосування, що значно скорочує час виконання роботи. Також серед відеоматеріалів були використані онлайн курси з 3D моделювання, які дозволяють отримати більш глибоку та вузько направлену інформацію стосовно вирішення деяких специфічних проблем, а також які дозволяють підвищити ефективність роботи.

Практичне значення. Отриманий результат може бути використаний у створенні такого продукту, як комп’ютерна гра, а також може бути доданий до професійного портфоліо.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 29 найменувань, додатків і містить 46 сторінки основного тексту, і 12 рисунків.

РОЗДІЛ 1.
АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ ПРОЦЕСУ СТВОРЕННЯ 3D ГРАФІКИ

1.1. Особливості побудови моделей, їхня будова та етапи створення

Аналітичне дослідження процесу створення 3D-графіки являє собою комплексний аналіз процесу створення тривимірної графіки з використанням комп'ютерів та супутніх з ними технологій. Цей процес зазвичай включає кілька етапів, на кшталт моделювання, текстурування, освітлення та рендерингу, пропоную розібратися у сутності кожного з них.

Моделювання - це процес створення 3D-моделі об'єкта, що включає в себе створення форми, деталей та поверхонь шляхом створення математичної моделі (геометрії). Процес моделювання відбувається за допомогою спеціальних програм, таких як Blender, 3ds Max, Maya або Zbrush.

Текстуруванням називають процес додавання текстур до поверхонь об'єкта. Текстури використовуються при створенні «матеріалів» - наборі параметрів, які відповідають за зовнішній вигляд. Цей процес можна виконати за допомогою програм, на кшталт Photoshop, Substance Painter або Quixel Mixer.

Освітлення являється імітацією присутності реального світла у сцені, де знаходиться 3D-модель. Зазвичай це виконується за допомогою різних типів освітлення, таких як точкове, напрямлене, амбієнтне та кілька інших. В залежності від програми, в якій будується освітлення, може бути доступним до використання як «швидке» світло, яке економить ресурси комп’ютера, так і фізично коректне світло .

Процесом створення остаточного зображення 3D-моделі називають рендеринг. Фактично це збір усіх компонентів візуальної складової моделі і відтворення їх у єдиній візуальній структурі. Рендеринг можна виконати за

допомогою спеціальних програм, таких як Arnold, V-Ray, Redshift або Octane Render.

Результат описаних вище етапів можна використовувати різних галузях, таких як відеоігри, анімація, архітектурне моделювання, медична візуалізація, індустріальний дизайн та інші. Наприклад, 3D-графіка використовується для створення візуалізацій медичних досліджень та процедур, що допомагають лікарям краще розуміти та діагностувати хвороби. У архітектурному моделюванні 3D-графіка використовується для створення візуалізацій будівель та інтер'єрів, що допомагає архітекторам та дизайнерам краще розуміти та презентувати свої проєкти.

Аналітичне дослідження процесу створення 3D-графіки допомагає розуміти складність та технічні вимоги для її створення і обслуговування, а також виявляти можливі проблеми та вдосконалювати методи створення. Дослідження можуть виявити, що певний етап створення є дуже трудомістким і потребує вдосконалення методів або використання нових технологій.

У робочих умовах в індустрії 3D процес створення графіки (а саме моделі) включає наступні кроки:

- Визначення мети моделювання: перед початком необхідно визначити її мету та призначення для підбору методів, технологій і засобів, які виконають поставлену задачу ефективніше . Це може бути створення прототипу виробу, ігрової моделі, рекламного рендеру, і тд.
- Створення скетчу: першим кроком є створення концептуального скетчу або малюнку об'єкта, який має бути створений. Цей етап може бути виконаний на папері, в графічному редакторі або у спеціальному програмному забезпеченні для концептуального малювання. Опираючись на нього, художник матиме розуміння того, як саме має бути створена модель.
- Вибір програмного забезпечення: для створення 3D-моделі потрібна спеціальна програма. Кожна з програм на ринку 3D графіки має свої

переваги та недоліки, тому вибір програмного забезпечення залежить від конкретної мети моделювання та індивідуальних потреб користувача або компанії.

- Створення базової форми: за допомогою наявних в програмі інструментів створюється форма, яка стане основою для майбутньої моделі. Неправильна побудова базової форми може порушити усі подальші етапи, тому художник має уважно помітити усі особливості форми об’єкту, який він хоче моделювати.
- Додавання деталей: наступним етапом є додавання деталей до базової форми. Це може включати додавання текстур, кольорів, освітлення, тіней та інших ефектів, також як і детальне моделювання різних частин об'єкта.
- Збереження та експорт: після завершення процесу створення, її необхідно зберегти в форматі, який може бути використаний для подальшого використання, такий як OBJ, FBX або STL. Це стане у нагоді, якщо модель потрібно використати в сторонній програмі.

Системи створення 3D графіки необхідні у створенні різноманітних проєктів, які використовують комп'ютерну графіку. Ці системи забезпечують потрібні інструменти для створення 3D моделей.

Роль та місце досліджуваної системи залежать від конкретного використання. Наприклад, у веб-дизайні та відеоіграх важливо мати систему з високою продуктивністю, щоб забезпечити швидку обробку графіки та відгук на дії користувача. У архітектурному проєктуванні та інженерії важливо мати систему з точними вимірами та можливістю моделювання реалістичних будівель та машин. Окрім того, вони можуть мати інтерфейси користувача з різним рівнем складності, які дають можливість працювати з системою без необхідності знати програмування або математику, що лежить в основі роботи системи. У загальному розумінні, досліджувана система визначає рівень доступності та можливостей для створення реалістичної 3D графіки.

Сучасний стан галузі 3D є дуже розвиненим та має безліч переваг. Проте, є кілька проблем, які виникають при розробці та використанні 3D графіки.

Одна з основних проблем - це складність та вимогливість до обладнання та програмного забезпечення. Для створення та відтворення сучасних 3D моделей необхідні потужні комп'ютери та спеціальні програми. Це може бути високо коштовним та недосяжним для багатьох користувачів та компаній. Іншою проблемою є нестача кваліфікованих фахівців. Для створення 3D графіки необхідна кваліфікація та досвід у багатьох областях, таких як моделювання, текстурування, освітлення, анімація та рендеринг, а також ерудованість та обізнаність у сутності процесів, явищ та об’єктів, які притаманні майбутній моделі або набору моделей.

До цього можна додати набір проблем, які виникають при навчанні майбутнього спеціаліста: попри наявність в Інтернеті великої кількості інформації на тему 3-D графіки, враховуючи комплексність знань, які необхідні для якісної роботи, не вистачає структурованого підходу до навчання. Цю проблему частково вирішує наявність приватних навчальних курсів, але єдиного структурованого підходу до навчання не існує. Також з цим пов’язана проблема відсутності централізованого дослідження сучасних підходів до створення графіки та технологій: інформація не встигає структуруватися і для її вивчення спеціалістами витрачається час, за який інформація може втратити актуальність. Єдине місце, де технології вивчаються в актуальний для них час – це приватні компанії, адже від цього напряму залежить ефективність їхніх робітників, що в свою чергу впливає на прибуток.

Паралельною проблемою є забезпечення реалістичної взаємодії між об'єктами та взаємодії з ними, такої як, наприклад, зіткнення згідно із фізичними законами. Це пов’язано або з недосконалістю систем, які відповідають за це, або з геометрією об’єкту, адже різна її будова одного і того самого об’єкту буде по різному на нього впливати.

Ще однією проблемою є необхідність використання великої кількості даних для роботи з графікою. Об'єм даних, які необхідні для моделювання та створення візуально складних об'єктів, може бути дуже великим. Це може викликати проблеми з обробкою та зберіганням даних, особливо при роботі зі складними проєктами.

Недостатня швидкість та продуктивність під час відтворення 3D зображень, особливо в режимі реального часу, є наріжним каменем проблем в роботі з графікою, особливо в іграх. Для відтворення високоякісних 3D зображень необхідна велика кількість обчислень та графічних обробок, що може бути вимогливим для більшості комп’ютерного обладнання, якого у багатьох немає.

Існує багато провідних компаній, які спеціалізуються на розвитку та вирішенні проблем 3D-графіки. Деякі з найбільш відомих з них на сьогоднішній день:

1. NVIDIA - ця компанія є одним з провідних виробників графічних процесорів (GPU), які використовуються для створення 3D-графіки. NVIDIA також пропонує різноманітне програмне забезпечення для розробки та оптимізації графіки.
2. AMD - як і NVIDIA, AMD виробляє графічні процесори для комп'ютерів, що дозволяє збільшити продуктивність 3D-графіки. Компанія також пропонує різноманітне програмне забезпечення для розробки та оптимізації графіки.
3. Autodesk - ця компанія є провідним розробником програмного забезпечення для проєктування та моделювання 3D-об'єктів. Деякі з їх відомих програм включають AutoCAD, 3ds Max та Maya.
4. Unity Technologies - компанія спеціалізується на створенні ігрових рушіїв для розробки 3D-ігор та інших візуальних додатків.
5. Epic Games - компанія розробляє один з найпопулярніших ігрових рушіїв - Unreal Engine, який дозволяє створювати реалістичну 3D-графіку для ігор та візуальних додатків.

Серед провідних вчених і спеціалістів у галузі 3D-графіки, випускників провідних університетів та компаній зі світу:

1. Ед Кетмел (Ed Catmull) - засновник та президент Pixar Animation Studios та Walt Disney Animation Studios.
2. Джон Лассетер (John Lasseter) - керівник креативного відділу Pixar Animation Studios та Walt Disney Animation Studios.
3. Ерік Хейнс (Eric Haines) - відомий автор та консультант з рендерингу та комп'ютерної графіки.
4. Крейг Рейнольдс (Craig Reynolds) - провідний науковець у галузі комп'ютерної анімації.

1.2. Теорія створення текстур та матеріалів для моделей. PBR текстури

Текстури - це зображення, які використовуються на поверхнях 3D-моделей для імітації фізичних властивостей об’єкту [2]. Для створення текстур можна використовувати різні програми, такі як Adobe Photoshop, Substance Painter, або будь-який інший програмний пакет для 3D моделювання.

PBR (Physically Based Rendering) текстури - це особливий тип текстур, який використовується для створення матеріалів з реалістичною поведінкою світла [10]. Це означає, що матеріали, створені з використанням PBR-текстур, виглядають більш реалістично в різних умовах освітлення.

Для створення PBR-текстур, зазвичай використовуються спеціальні програми, такі як Substance Designer або Quixel Mixer. Ці програми дозволяють створювати текстури, які включають інформацію про різні фізичні властивості матеріалу, такі як здатність до відбивання світла, розсіювання світла та інші.

Взаємодія 3D-моделей та текстур відбувається за допомогою матеріалів. Матеріалом називають набір параметрів, що описують зовнішній вигляд поверхні 3D-моделі, такі як колір, текстура, транспарентність (прозорість), рефлексія та інші. Матеріал може бути створений шляхом комбінування різних функцій,

генераторів та тому подібних інструментів, які використовують текстуру як початковий матеріал, а в фіналі видавати унікальний результат.

Для створення текстур в 3D-графіці використовуються різні види карт, які містять додаткову інформацію про поверхню об'єкта. Основні карти, які використовуються для створення текстур, включають:

1. Карта колірної текстури (Diffuse map) - ця карта містить основний колір поверхні об'єкта. Вона використовується для відображення різних текстур, таких як шкіра, дерево, камінь та інші [9].
2. Карта нормалей (Normal map) – є різновидом рельєфної карти. Вони являють собою особливий тип текстури, який дозволяє додавати деталі поверхні, такі як нерівності, подряпини та інші об’єкти, до моделі, яка вловлює світло так, ніби вони представлені справжньою геометрією [11]. Тобто ця карта дозволяє створити ілюзію наявності об’єкта на поверхні.
3. Карта відбивності (Specular map) – у ній наявна інформація про ступінь відбивності поверхні. Використовується для створення блискучих ефектів на поверхні об'єкта, таких як метал, глянець або скло [12].
4. Карта грубості (Roughness map) - містить інформацію про ступінь гладкості поверхні. Вона вказує комп’ютеру, де відображати поверхню з більшим блиском, або навпаки більш матове відображення. Це дозволяє імітувати, наприклад, плями від жиру, пил або сліди на скляних поверхнях, і не тільки [10].
5. Карта освітлення (Light map) – у цій карті є інформація про освітлення на поверхні об'єкта. Її використовують для створення ефекту тіней та більш реалістичної візуалізації освітлення [1].

Існує багато інших карт, які теж є важливими, проте згадаю я про них пізніше.

1.3. Загальні відомості про технології рендерінгу та освітлення у ігровому рушії Unreal Engine 5

Враховуючи, що для кінцевого продукту нам потрібна лише візуальна частина, варто не тільки опустити відомості про загальний функціонал ігрового рушію, який використовується як платформа для демонстрації, а також висвітлити такі складові як освітлення та рендеринг, бо саме вони впливають безпосередньо на загальний вигляд сцени рівню.

Основні технології рендерінгу в Unreal Engine 5 включають:

1. Nanite — це система віртуалізованої геометрії Unreal Engine 5, яка використовує новий внутрішній формат сітки та технологію візуалізації для відтворення деталей у масштабі пікселів і великої кількості об’єктів. Він розумно працює лише над тими деталями, які можна сприйняти гравцем, і не більше [21]. Ця система дозволяє відображати великі та деталізовані об’єкти з мільйонами полігонів без зайвого навантаження на систему та комп’ютер.
2. Lumen — нова система глобального освітлення, яка дозволяє динамічно підлаштовувати світло від джерел освітлення. Lumen використовує кілька методів трасування променів для вирішення проблеми глобального освітлення та відбиття світла від об’єктів. За замовчуванням використовує програмне трасування променів через поля відстані зі знаком (функція орієнтованої відстані), але може досягти вищої якості, якщо ввімкнено апаратне трасування променів [19].
3. Temporal Super Resolution — це система підвищення дискретизації, вбудована в Unreal Engine 5. Підвищення дискретизації — це метод, який широко використовується в багатьох середовищах, включаючи фотографію, аудіо, графічний дизайн та образотворче мистецтво, а тепер і розробку ігор. Підвищена дискретизація бере медіафайл, наприклад зображення, і збільшує роздільну здатність до десяти разів без шкоди для якості [29].

4. Virtual Shadow Maps — технологія, яка дозволяє підвищити якість тіней в грі, зменшуючи використання пам'яті . Вона використовується для створення більш точних тіней та зменшення артефактів, пов'язаних з обрізанням тіней [26].
5. Niagara — система частинок, яка дозволяє створювати складні ефекти, такі як вогонь, дим та воду. Niagara підтримує високу деталізацію та може забезпечити широкий діапазон налаштувань для створення різних ефектів [14].
6. Chaos — новий фізичний рушій, який дозволяє створювати складні фізичні ефекти, такі як руйнування та деформації, що додатково підвищує рівень реалізму в грі [13].

Основні технології освітлення в Unreal Engine 5 включають:

1. Lumen — як вже згадувалося, це нова система глобального освітлення, яка дозволяє динамічно підлаштовувати світло від джерел освітлення .
2. Reflections — Unreal Engine 5 підтримує підрахунок відбиття світла на поверхнях, що дозволяє створювати реалістичні відображення дзеркальних та металевих поверхонь [23].
3. Shadow Maps — технологія, яка дозволяє розраховувати тіні від джерел світла на поверхнях, що дає можливість створювати реалістичні тіні в грі.
4. Lightmass — це рішення, яке використовується для підрахунку індиректного освітлення в грі, що дає можливість створювати більш реалістичні освітлення та тіні [18].
5. Volumetric Fog — це технологія, яка дозволяє створювати атмосферні ефекти, такі як туман та густа атмосфера [27].

Варто розуміти, що 3D розвивається далі, і кожна нова ітерація розробки технологій дозволяє реалізувати все більш амбіційні та складні задачі. Ми не можемо вважати, що технології будуть постійно актуальними і нам потрібно адаптуватися до нових реалій, які встановлюють головні гіганти індустрії графіки

та відеоігор. Перелічені вище аспекти цих технологій можуть застаріти через якийсь час, коли з’являться більш нові функції.

Про амбітність додам наступне: вже зараз наявні технології здатні створювати вражаючу гіперреалістичну графіку для кіно та реклами, але також вони розвиваються для реалізації ідей щодо створення комп’ютерних ігор.

РОЗДІЛ 2.
ОСНОВНА ТЕОРІЯ СТВОРЕННЯ ІГРОВОГО РІВНЯ З ТОЧКИ ЗОРУ ВІЗУАЛЬНОЇ СКЛАДОВОЇ

2.1. «Блокаут» та планування побудови рівня

Як і будь-який якісно продуманий продукт, комп'ютерна гра починається з планування. У випадку області і предмету мого дослідження мене більше цікавить аспект, пов'язаний з графікою. Планування ігрових рівнів технікою «блок-аут» може забезпечити більш ефективну оптимізацію гри, покращення її продуктивності та зниження навантаження на обчислювальну машину [28].

Одним з методів планування рівнів технікою блок-аут є побудова прототипу рівня, який відображає головні елементи та локації гри, але має меншу кількість деталей та текстур. Прототип можна створити шляхом використання простих геометричних форм, таких як куби, циліндри та сфери, які відображають головні елементи рівня, такі як стіни, сходи та інші об'єкти.

Особливістю ігрового рушія, який я використав у своїй випускній кваліфікаційній роботі, є те, що для блок-ауту непотрібно створювати базові фігури у сторонній програмі: в Unreal Engine 5 доступна функція додавання геометричних фігур безпосередньо у рушії. Також варто зауважити, що для блок-ауту непотрібно витрачати час на більш точне створення моделі, адже першочергово при виконанні даної техніки головне – це ефективність і швидкість візуалізації концепту рівня. Фігури можуть «провалюватися» одне в одну, мати не зовсім точну форму, і тд.

Після створення прототипу рівня, розробник може застосувати блок-аут для зниження рівня деталізації елементів рівня, які знаходяться на задньому плані або не потрібні для головної дії гри. Наприклад, стіни, які знаходяться далеко від головної дії, можна замінити менш деталізованими моделями, що знизить кількість полігонів, які потрібно відобразити на екрані.

Блок-аут також може бути використаний для оптимізації роботи ігрового рушія. Наприклад, він може допомогти в зниженні часу завантаження рівнів та покращенні продуктивності гри шляхом зниження кількості полігонів, які потрібно відобразити на екрані. Це може зменшити навантаження на процесор та відеокарту, що дозволить ігровому рушій працювати більш ефективно та покращити продуктивність гри. У моїй роботі була використана технологія Nanite, яка дозволяє змінювати геометрію в залежності від відстані до об’єкту. Ця технологія може бути ефективно скомбінована з технікою блок-ауту.

Щоб ефективно використовувати цю техніку, розробники повинні мати глибокі знання про ігровий рушій, який вони використовують, а також знати, як ефективно розподіляти деталізацію на різних ділянках рівня. Розробники повинні також мати розуміння того, як різні елементи гри взаємодіють між собою.

Одним з найважливіших аспектів розробки ігрових рівнів є забезпечення балансу між графікою та продуктивністю гри. Використання блок-ауту дозволяє розробникам знайти баланс між цими аспектами, щоб забезпечити високу якість графіки та продуктивність гри. Також це може допомогти знизити час розробки рівнів, оскільки це дає розробникам не створювати фінальний варіант графіки одразу, що може зекономити сили і кошти на розробку, а також стає доступна можливість швидше експериментувати з різними варіантами дизайну рівнів.

2.2. Етапи створення локації

Локація у комп'ютерній грі - це визначена область віртуального світу гри, яка має свої власні характеристики та атмосферу. Локація може бути будь-якого розміру та форми - від маленького кімнатного приміщення до великої місцевості або світу.

У багатьох іграх кожна локація має свій власний дизайн, інтерактивні елементи та завдання для гравців, що можуть бути пов'язані з головною історією гри або бічними завданнями. Локації можуть бути статичними, тобто незмінними

протягом всієї гри, або динамічними, коли вони змінюються з часом або в залежності від дій гравця.

В цілому, локації додають глибину та різноманітність ігровому процесу, дозволяючи гравцям досліджувати різні області віртуального світу, знайомитися з персонажами та виконувати різноманітні завдання.

Локації можуть мати велике значення для геймплею, оскільки вони можуть впливати на взаємодію гравця з оточенням та іншими персонажами. Наприклад, локація може мати важливу роль у вирішенні головоломок або битв з ворогами, або вона може стати джерелом ресурсів або нагород для гравця.

Крім того, локації можуть бути використані як інструмент для налаштування настрою гри та створення певної атмосфери, а також як спосіб доносити історію гри та її сюжет. Кожен предмет, об’єкт та композиція можуть наштовхнути гравця на роздуми про деталі сюжету або підказати йому, які дії потрібно зробити у контексті історії.

Ігрові локації можуть бути розроблені як статичні (наприклад, як локації у традиційних RPG), так і динамічні (наприклад, як у відкритих світах, таких як Grand Theft Auto або Minecraft). У динамічних локаціях зміна оточення може відбуватися у залежності від дій гравця або випадкових факторів, що дозволяє створювати більш іммерсивний та непередбачуваний геймплей.

Створення локації у комп'ютерних іграх доволі складний та багатоетапний процес, який включає в себе різноманітні етапи, починаючи від концептуалізації та проєктування, і закінчуючи імплементацією в саму гру. Розберемо етапи створення локації:

1. Концептуалізація та проєктування: на цьому етапі розробники визначають загальну концепцію та цілі локації. Вони вирішують, яким чином локація буде інтегрована в головну історію гри, який тип локації буде створено (наприклад, пустеля, ліс, місто), який рівень складності вона матиме, які завдання будуть доступні для гравців, і т.д.

Концептуалізацією називають процес визначення ідеї гри та її концепції. Розробники збираються, щоб обговорити основні елементи гри, включаючи історію, персонажів, геймплей та графіку. На цьому етапі можуть бути створені ескізи концепції, щоб візуалізувати, як гра має виглядати та працювати. Розробники можуть також з'ясувати, яку аудиторію гри вони мають на увазі та які її потреби та вимоги.

Після концептуалізації, розробники переходять до проєктування гри. Це включає в себе розробку детального плану для реалізації концепції, включаючи визначення функціональності, дизайну та механік гри. Розробники можуть використовувати різні інструменти, такі як діаграми потоків даних та схеми взаємодії, щоб допомогти у вирішенні технічних питань та визначенні деталей гри.

Після того, як проєктування гри завершено, можуть бути створені детальні плани для різних аспектів гри, таких як програмування, дизайн графіки, звуковий дизайн та інші. Розробники також можуть проводити додаткові тести для перевірки різних аспектів гри та забезпечення того, що вона працює якнайкраще.

2. Створення прототипу: розробники створюють грубий прототип локації, щоб протестувати його на виконання основних вимог до локації. Це допомагає їм вирішити проблеми та узгодити деталі, що дозволяє зекономити час та зусилля на наступних етапах.

Створення прототипу є наступним етапом після концептуалізації та проєктування. Прототип - це тестова версія гри, яка допомагає розробникам перевірити різні аспекти гри, такі як механіки гри, графіку та функціонал.

Першим кроком в створенні прототипу є визначення того, який тип прототипу потрібний. Наприклад, можна створити прототип на папері, цифровий прототип, або використати спеціальне програмне забезпечення для створення прототипу.

Після визначення типу прототипу, розробники можуть розпочати роботу над його створенням. Перш за все, вони визначають механіку гри, яку вони хочуть

перевірити, та створюють просту версію цієї механіки. До прикладу, якщо ціль - перевірити, як гравець взаємодіє з різними предметами у грі, розробники можуть створити прототип, де гравець може взаємодіяти з кількома предметами.

Далі розробники можуть додати інші елементи до прототипу, такі як функціонал, історія гри, графіка тощо. Найважливішим завданням на цьому етапі є забезпечення того, щоб прототип був достатньо простим, щоб його можна було швидко тестувати та змінювати.

Розробники можуть провести тестування з гравцями та отримати повернення щодо того, що працює та що потребує покращення. Це допомагає розробникам виявити проблеми та вдосконалити їх, перш ніж перейти до наступних етапів розробки. У разі необхідності, розробники можуть повернутися до етапу проєктування та внести зміни до концепції гри.

3. Розробка зовнішнього вигляду: тут розробники створюють зовнішній вигляд локації, включаючи дизайн та візуальні ефекти. Це включає в себе визначення палітри кольорів, моделей об'єктів, текстур, освітлення та інших елементів, що дозволяють створити атмосферу та настрій локації.

При створенні зовнішнього вигляду, розробники використовують спеціальні програмні засоби, такі як 3D-моделювання, графічні редактори та інші інструменти. При цьому вони керуються концепцією гри, яка була розроблена на попередньому етапі.

Перш ніж розпочати створення зовнішнього вигляду гри, розробники визначають її стиль та атмосферу. Наприклад, гра може мати ретро стиль, футуристичний стиль, реалістичний стиль тощо. Розробники також визначають кольорову гаму, освітлення, текстури, ефекти та інші деталі, щоб донести задуману атмосферу гри до гравців. Особливу увагу потрібно приділяти зовнішньому вигляду у разі, якщо стоїть задача створити локацію у реалістичному стилі, бо тоді потрібно врахувати особливості будови, структури та зовнішності об’єктів, які будуть взяті за основу моделей у локації. Наприклад, не варто

використовувати насичені кольори там, де їх не може бути, адже це виглядає неприродно, і може спотворити враження про гру.

4. Створення об'єктів: на цьому етапі створюються об'єкти, які знаходяться в локації, такі як будівлі, рослини, тварини та інші елементи. Розробники використовують 3D-моделювання та інші програми для створення цих об'єктів та їх анімації.
5. Розміщення об'єктів та ландшафту: на даному етапі розробники створюють ландшафт та розміщують об'єкти на локації. Це включає в себе вирішення питання про те, які об'єкти будуть розміщені в якому місці, як вони взаємодіятимуть між собою та як будуть впливати на гру та гравців.

Починаючи з розміщення ландшафту, розробники створюють терен, який включає в себе гори, долини, рівнини та інші природні форми рельєфу. Вони використовують спеціальні програми, такі як World Machine, для створення висотних карт та генерації рельєфу. Після цього, розробники можуть додавати деталі до терену, такі як камені, дерева та інші об’єкти навколишнього середовища. В рушії Unreal Engine 5 наявна своя власна система створення ландшафту, що дозволяє суттєво скоротити витрати часу на його створення та додавання до локації.

Після створення ландшафту, розробники починають розміщувати об'єкти в грі. Вони використовують програмні засоби, такі як Unity, Unreal Engine та інші, для розміщення об'єктів у 3D-просторі.

Розміщення об'єктів може також включати створення штучного освітлення, яке створює атмосферу та підсилює настрій гри. Розробники можуть додавати різноманітні джерела світла, такі як ліхтарі, свічки, факели, які роблять гру більш реалістичною та живою.

Крім розміщення об'єктів, розробники можуть створювати штучних персонажів, які рухаються та взаємодіють з гравцем. Ці персонажі можуть мати власний штучний інтелект та реагувати на дії гравця, створюючи більш реалістичне відчуття життя у грі.

У процесі розміщення об'єктів та ландшафту, розробники повинні також враховувати оптимізацію гри, щоб забезпечити її плавну роботу та запобігти зависанням. Вони повинні розробляти локацію таким чином, щоб вона працювала на різних пристроях з різними характеристиками, такими як швидкість процесора та обсяг оперативної пам'яті.

6. Тестування: розробники проводять тестування локації, щоб виявити та виправити помилки, покращити геймплей та забезпечити найкращий досвід для гравців.

Першим кроком тестування є внутрішнє тестування, де розробники перевіряють гру власноруч, використовуючи різні сценарії гри та різні методи гри. Вони також можуть використовувати спеціальні програмні засоби для виявлення технічних помилок, таких як програмні збої, помилки відображення та інші проблеми з функціональністю.

Другим кроком тестування є бета-тестування, де гравці з усього світу можуть випробувати гру та надати зворотний зв'язок розробникам. Розробники можуть включити спеціальні інструменти для збору даних, щоб визначити проблеми та вдосконалити гру на основі відгуків гравців.

7. Оптимізація: спеціалісти працюють над оптимізацією локації для забезпечення максимальної продуктивності та забезпечення того, що гра працює ефективно на різних пристроях та платформах.

Оптимізація - це процес покращення продуктивності гри шляхом зменшення обсягу ресурсів, необхідних для її функціонування. Оптимізація є важливим етапом у створенні комп'ютерних ігор, оскільки вона дозволяє підвищити продуктивність та забезпечити гладке функціонування гри на різних платформах.

Оптимізація може включати в себе наступні кроки:

- Пошук та виправлення помилок в коді - це може включати в себе виправлення погано написаного коду, видалення зайвого коду,

усунення непотрібних функцій та використання більш ефективних алгоритмів.

- Використання оптимізованих ресурсів - використання оптимізованих ресурсів може значно підвищити продуктивність гри. Наприклад, використання текстур меншого розміру, зменшення кількості полігонів у моделях персонажів та об'єктів, використання оптимізованих шейдерів та ін.
- Регулювання налаштувань графіки та звуку - це може включати в собі зменшення деталізації графіки, зменшення кількості частинок, використання меншої кількості ефектів та вимкнення функцій, які не впливають на геймплей. Також може бути важливим регулювання налаштувань звуку, які можуть впливати на продуктивність.

Після виконання всіх попередніх етапів розробки локації, необхідно провести оптимізацію, щоб забезпечити плавний та безперебійний геймплей, зменшити час завантаження локації та забезпечити максимальну продуктивність гри, наприклад:

- Використання мінімальної кількості полігонів: чим менше полігонів містить локація, тим швидше вона буде завантажуватися та працювати.
- Використання LOD (Level of Detail): LOD - це техніка, яка дозволяє зменшити деталізацію моделі, коли вона знаходиться на відстані від гравця. Наприклад, деталізацію можна зменшувати при наближенні до об'єкту, що дозволяє зменшити навантаження на процесор.
- Використання текстур з низькою роздільною здатністю: дозволяє зменшити вагу файлів та зменшити час завантаження.
- Використання пакування текстур: пакування текстур дозволяє зменшити кількість викликів до графічної пам'яті та зменшити час завантаження локації.

- Використання зручної геометрії: використання геометрії зручної форми дозволяє зменшити кількість полігонів та прискорити процес рендерингу.
- Використання алгоритмів керування об'єктами: керування об'єктами дозволяє динамічно змінювати кількість об'єктів на екрані, що дозволяє зменшити навантаження на процесор.

8. Імплементація: на даному етапі локація стає частинкою гри та готова для використання гравцями. Вона може бути додана в гру після відпрацювання всіх деталей та врахування всіх нюансів.

Імплементацією називають процес реалізації плану проєктування в робочій програмі. Під час імплементації, програміст виконує написання коду та інтеграцію всіх елементів гри, таких як графіка, звук, логіка гри, фізика та інші.

Під час імплементації, розробники використовують мови програмування, такі як C++, Java або Python, залежно від обраної платформи. Розробники реалізовують головну логіку гри та взаємодію з користувачем, яка включає в себе управління персонажами, обробку введення користувача, управління камерою та інші функції.

Після написання коду розробники проводять тестування програми, щоб переконатися, що вона працює належним чином та відповідає вимогам проєкту. Якщо виявляться будь-які помилки або проблеми, розробники виправляють їх та повторюють тестування, задіявши компетентних у цьому питанні співробітників. Також можливий варіант, який використовується останнім часом в індустрії відеоігор: для тестування все частіше залучають звичайних гравців шляхом надання доступу до версії гри.

2.3. Постобробка

Постобробка в іграх дозволяє підвищити рівень реалізму та візуальної якості гри. Після створення геометрії моделі, текстур та освітлення, постобробка є останнім етапом в процесі створення візуальних ефектів [22].

Основна мета постобробки в іграх полягає у тому, щоб додати деталізацію, реалізм та глибину до геометрії, текстур та освітлення в грі. Це може включати налаштування кольорів, контрасту та яскравості, розмиття та ефекти тіней, рельєфність та текстурні ефекти, покращення освітлення та створення ефектів спеціальних візуальних ефектів.

Процес створення постобробки може включати такі етапи, як аналіз сцени гри, вибір відповідного ефекту, застосування ефекту до зображення та налаштування параметрів для досягнення бажаного результату. Цей процес може бути дуже трудомістким і вимагати багато часу та зусиль, але результат може значно покращити візуальний досвід гравця в грі.

Взагалі, постобробка в іграх є важливим елементом в створенні реалістичного та якісного геймплею. Вона дозволяє розробникам створити більш деталізовані та реалістичні світи в іграх, що може забезпечити більш глибокий занурення гравця в гру.

Постобробка може бути також використана для вирішення певних технічних проблем, таких як проблеми з текстурами, освітленням та зображенням в цілому. Вона може допомогти розробникам покращити роздільну здатність зображення, знизити рівень шуму та покращити рівень деталізації візуальних ефектів.

Постобробка в іграх може бути реалізована за допомогою різноманітних технік та програмних засобів. Найпоширеніші техніки постобробки включають:

- Освітлювальна постобробка - це процес налаштування освітлення у грі для створення більш реалістичної атмосфери. Цей процес може бути досягнутий за допомогою різних програмних засобів для редагування освітлення, таких як Unity, Unreal Engine 5, і тд.

- Ефекти постобробки - це процес додавання різноманітних візуальних ефектів до гри, таких як дим, вогонь, вода та інші. Він може бути досягнутий за допомогою різних програмних засобів для створення візуальних ефектів.
- Композитинг - це об'єднання різних елементів гри, таких як текстури, освітлення та візуальні ефекти, для створення остаточного візуального ефекту. Цей процес може бути досягнутий за допомогою різних програмних засобів для композитингу, таких як Nuke, Adobe After Effects, або внутрішніх засобів для композітингу, наприклад як у Blender.

Створення гри як продукту – неймовірно складний і комплексний процес. Для реалізації перелічених вище принципів і методів розробки інвестори фінансують команди розробників все більше і більше. Тим же командам, у яких амбіції поменше, або яким не пощастило мати спонсорів, мають творчо підходити до розробки, враховуючи баланс між бюджетом і правильною реалізацією вищевказаних принципів, методів та способів розробки.

РОЗДІЛ 3.
РОЗРОБКА 3D ГРАФІКИ ТА ЇЇ ВИКОРИСТАННЯ У ІГРОВОМУ РУШІЇ UNREAL ENGINE 5

3.1. Особливості створення моделей у пакеті 3D моделювання Blender

Blender - це безкоштовна програма для створення 3D-моделей, анімації, візуалізації та багатьох інших 3D-застосувань. Основні можливості Blender включають моделювання, текстурування, анімацію, рендеринг, складання, композитинг, створення ігор і багато іншого.

Одна з найзручніших функцій інтерфейсу програми – це створення власного робочого простору: робочі вікна можна створювати і видаляти в будь-якій послідовності і кількості. Можна змінювати розмір та конфігурацію, а також створювати власні вкладки робочих просторів (наприклад, якщо в одному вікні у вас є бажання створювати модель, а в іншому робити текстури для моделі, в програмі вже є за замовчуванням налаштовані для цього вкладки, але ви можете створити і свої власні). Вигляд інтерфейсу програми разом з моделлю, над якою велася робота, можна побачити у Додатку А.

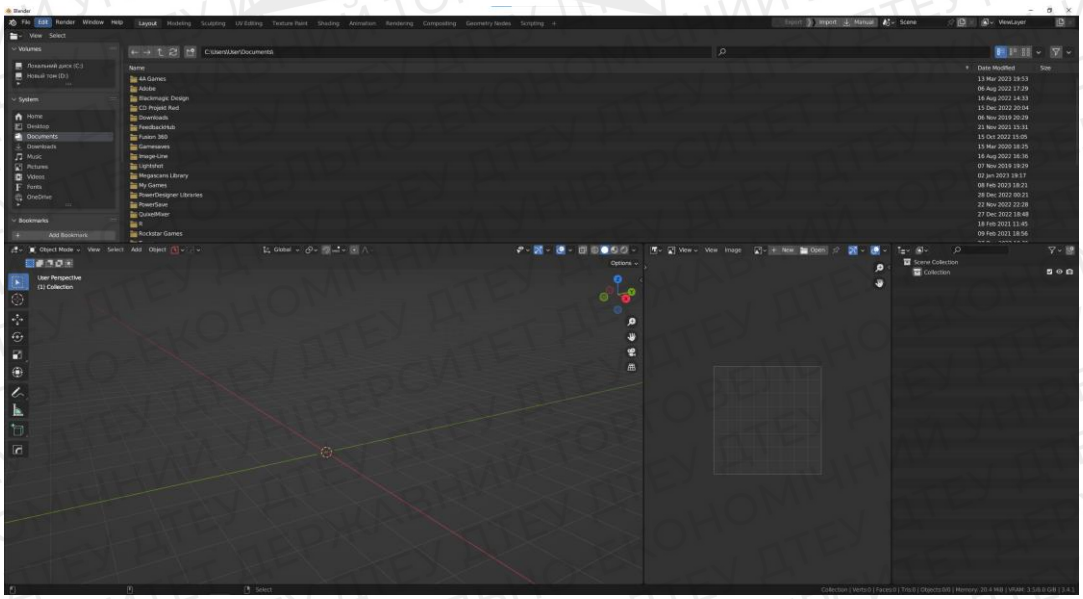


Рис. 3.1. Інтерфейс програми та його можливості

Наша задача в контексті випускної кваліфікаційної роботи – розібрати особливості моделювання в цій програмі. Тому я опущу подробиці роботи з іншими її функціями, і перейду безпосередньо до моделювання.

У Blender існує кілька різних методів моделювання, включаючи моделювання за допомогою мешів, нурбсів, площин і кривих.

Моделювання за допомогою мешів (mesh modeling) полягає в створенні 3D-моделів з мережі точок, з'єднаних лініями (ребрами) та поверхнями (гранями). Цей метод дозволяє створювати детальні 3D-моделі з великою кількістю точок. Він використовується найчастіше.

Рис. 3.2 Моделювання за допомогою мешів

Нурбси (NURBS) - це математичні поверхні, які дозволяють створювати складні форми з використанням більш простих форм, таких як круги, квадрати та еліпси. Цей метод дозволяє створювати гладкі форми з мінімальною кількістю точок, але використовується рідко.

Площини (Plane modeling) - це метод моделювання, який полягає в створенні 3D-об'єктів з площин, які з'єднані між собою. Цей метод дозволяє швидко створювати прості форми, такі як стіни, підлоги та столи.

Криві (Curve modeling) - це метод моделювання, який полягає в створенні 3D-об'єктів з кривих ліній, які з'єднані між собою. Цей метод дозволяє створювати складні форми, такі як скручені трубки, залізничні колії та інші складні форми.

У Blender існує велика кількість інструментів та режимів, що дозволяють користувачам створювати складні 3D-моделі та анімації. Ось декілька основних режимів у Blender:

- Режим об'єктів: у цьому режимі можна створювати, переміщувати, масштабувати та обертати 3D-об'єкти у просторі [5].
- Режим редагування: у ньому ви отримуєте доступ до структури об'єкту (геометрії) і здатні проводити різного роду маніпуляції для зміни його форми за допомогою інструментів [5].
- Режим скульптингу: тут можна детально проробити деталі 3D-моделі, змінюючи її форму та текстуру. Це метод, який імітує реальну роботу з глиною [5].

Одним з найголовніших інструментів моделювання являється ручне моделювання – тобто прямий вплив на геометрію об’єкта за допомогою інструментів. Розробник отримує до них доступ при переході до «Режиму редагування».

До переліку цих інструментів входять:

1. 3D Cursor – інструмент, який дозволяє маніпулювати всім, що є у програмі. Це точка, яку виставляє розробник для задавання точки маніпуляції, наприклад, на місці Курсору при потребі з’являється будь яка фігура або об’єкт [3].
2. Extrude – «видавлювання», тобто копіювання точок, ребер або граней і створення нових точок, ребер або граней по заданій формі [4].
3. Move, Rotate, Scale – переміщення, поворот та масштабування елементів відповідно.

4. Inset – створювання геометрії всередині грані в тій площині, що і сама грань (зазвичай робиться всередину).
5. Bevel – створення фаски або скоса.
6. Loop Cut – створення геометрії методом закритого або незакритого кільця. Для адекватної реалізації цього інструменту у грані, в якій і створюється «луп», має бути 4 кути. Якщо грані мають між собою однакову кількість кутів і вона дорівнює 4, і грані замикаються, то круг буде замкнутий. Якщо ж ні, то круг або буде не замкнений, або ребро не сформується взагалі (в такому випадку інструмент буде генерувати замість ребер точки, якими буде розділяти навпіл вже наявні ребра) [8].
7. Knife – дозволяє створювати ребра і точки у довільному порядку методом, схожий на «різання ножем».
8. Poly Build – створення об’єкту з нуля по точках.
9. Spin – створення копій геометрії по кругу відносно опорної точки.

Окремою особливістю Blender являються модифікатори – це функції, які впливають на об’єкт особливим чином [5]. Вони розділяються на декілька груп:

1. Modify – це група модифікаторів, яка дозволяє переносити інформацію з одного об’єкта на інший, або змінювати інформацію про затінення і напрями нормалей у об’єкту.
2. Generate – це група модифікаторів, яка дозволяє з наявного об’єкту генерувати щось інше. Наприклад, модифікатор Subdivision Surface підрозділяє поверхню і заокруглює простір між точками, які були підрозділені, таким чином створюючи більше округлу поверхню.
3. Deform – це модифікатори, які деформують об’єкт. Наприклад, модифікатор Bend дозволяє буквально гнути об’єкт за заданими параметрами, а модифікатор Shrinkwrap дозволяє огортати один об’єкт іншим об’єктом.

4. Physics – група модифікаторів, які генерують фізичні процеси відносно об’єкту.

На рисунку нижче ви можете побачити весь перелік модифікаторів.

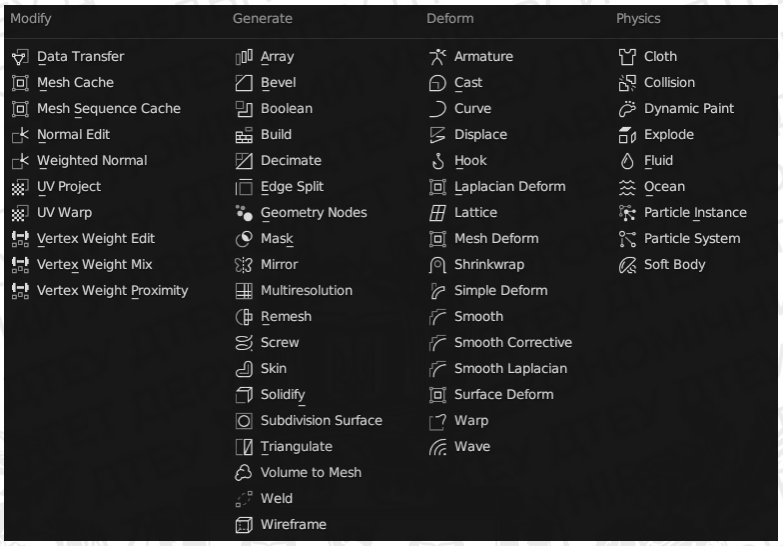


Рис. 3.3. Перелік усіх модифікаторів

Blender є відкритим програмним забезпеченням, що означає, що він доступний безкоштовно та його код може бути змінений та модифікований користувачами. Деякі плагіни можуть повністю поміняти підхід до роботи, наприклад, для так званого Hard Surface моделювання (моделювання об’єктів, які створені людиною) використовується цілий набір плагінів: Box Cutter, MESHMachine, Machine Tools, і так далі.

У своїй випускній кваліфікаційній роботі при моделюванні усіх об’єктів у сцені я використовував в основному метод моделювання мешами та площинами. Також для деталізації я використовував метод скульптингу, але варто зазначити, що для цього я користувався іншою програмою, тож у цьому розділі я розкажу лише про перші два згадані методи моделювання.

При визначенні об’єкту, який ми маємо моделювати, варто розрахувати кількість полігонів або трикутників, які ми можемо виділити на цей об’єкт. Це дозволить нам у майбутньому зекономити ресурси комп’ютера на стадії додавання об’єкту на рівень. Якщо об’єкт простий, то можна задати його форму наявними фігурами, для цього в основному використовується Куб, Квадрат, Циліндр, Коло. Якщо мені потрібна фігура з іншою кількістю кутів, ніж зазвичай,

я можу поміняти цю кількість або при створенні фігури (наприклад циліндр з 32 вершинами, а не з 24), або ж вручну за допомогою вищевказаних у цьому розділі інструментів.

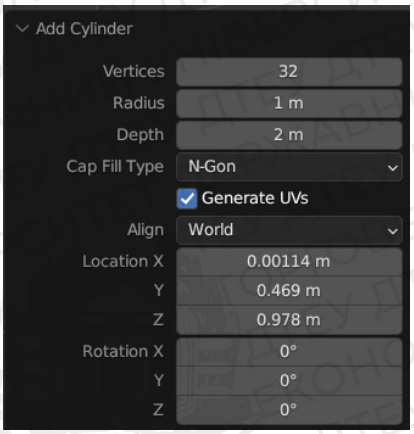


Рис. 3.4. Додавання меша

Далі я за допомогою інструментів змінюю форму об’єкта, притримуючись наступних принципів: по-перше, об’єкт має мати форму, яка вказана у матеріалах, або так званих «референсах» - фото, відео, схеми або креслення реально існуючого об’єкту в об’єктивній реальності. Вигляд програми для роботи з референсами PureRef можна побачити у Додатку Г.

По-друге, модель має бути оптимізована: якщо точка, ребро або грань при видаленні не змінює форму об’єкта, то вони зайві і мають бути видалені. По-третє, геометрія об’єкту має складатися з трикутників, або квадратів чи прямокутників. Це дає змогу більш зручно змінювати об’єкт під поставлені задачі, а також дозволяє контролювати шейдінг. Шейдінг (англ. shading) - це процес створення візуальної реалістичності насиченості зображень в комп'ютерній графіці. Це означає, що шейдінг використовується для створення візуального ефекту текстури, що надає об'єкту на зображенні вигляду реального предмета. Якщо геометрія була побудована неправильно, і має багатокутники, або неправильно з’єднані точки у площині, то шейдінг буде виглядати некоректно, що може вплинути на кінцевий результат.

У разі якщо об’єкт більш складний, я можу комбінувати між собою фігури, правильно їх з’єднувати і змінювати на свій розсуд. Таким чином я можу

створити складну форму з простих форм. На цьому етапі потрібно постійно перевіряти об’єкт на такі показники:

1. Кількість полігонів (трикутників) – враховуючи попередні розрахунки, які були проведені перед створенням об’єкту, ми маємо постійно контролювати кількість геометрії, яку ми на нього виділили. Потрібно старатися не виходити за поставлені рамки.
2. Зайві точки, ребра, грані – вони можуть викликати різного роду помилки, які впливають на кінцевий результат роботи. Або вони можуть вплинути на швидкість і складність роботи над об’єктом [6].
3. Зшиття – це процес перевірки геометрії на те, щоб усі точки були між собою правильно з’єднані. Через людський фактор, точки можуть між собою не «зшитися» і вони просто накладаються одна на одну. Це складно помітити і таке явище може викликати проблеми з роботою над об’єктом у майбутньому. Наприклад, комп’ютер не зможе зрозуміти, до якої групи точок, або до якої грані належить точка, і при формуванні UV розгортки розтягне грань таким чином, що кінцевий її вигляд буде некоректним.
4. Показники розміру, масштабу, повороту та координат. Якщо здійснювати маніпуляцію з об’єктом в об’єктному режимі, то при масштабуванні та повороті мешу зміняться відповідні показники. Якщо візуально це не принесе ніяких проблем з роботою, то при накладанні модифікаторів або експорті та роботі над мешем в іншій програмі можуть виникнути неприємності: об’єкт може стати завеликим або замалим від початкових розмірів, або повернутий не так, як треба. Для того, щоб уникнути цього, потрібно постійно слідкувати за цим показником, і в разі помилки затвердити дані показники як стандартні. Тоді комп’ютер не буде змінювати об’єкт і залишить все так, як і планувалося розробником.

5. «Півот поінт» - це точка, відносно якої відбуваються усі маніпуляції над мешем. У нього може не бути геометрії, але півот поінт буде завжди. Без нього не може існувати об’єкту. Положення півот поінту виставляє сам розробник: це може бути точка геометричної середини об’єкту, центру мас або довільна точка, яку виставляє розробник за допомогою 3D курсору. У разі спрацювання людського фактору може відбутися ситуація, коли півот поінт виставлений неправильно і, наприклад, замість того, щоб розмістити об’єкт відносно підлоги на рівні, об’єкт провалюється «крізь текстури».
6. Порядок модифікаторів: при неправильному порядку модифікаторів результат може бути спотворений та відображатися некоректно. Потрібно знати властивості модифікаторів і їхню сутність, для того, щоб знати, в якій послідовності їх потрібно ставити.
7. Топологія: геометрія об’єкту має відповідати тому, що задумав розробник. Якщо розробник хоче круглу вазу, то для цього потрібно побудувати округлу геометрію і слідкувати, щоб його дії не порушили топологію, наприклад, у даному випадку, кривизну і округлість об’єкту. Також неправильно побудована топологія може викликати проблеми із затіненнями і реагуванням на джерела світла.
8. Полярність полігонів. Кожен полігон має 2 сторони: позитивну та негативну. Позитивна сторона відмальовується відеокартою, негативна - ні. Це робиться для економії ресурсів комп’ютера, бо, з практичної точки зору нікому не потрібно бачити те, що знаходиться всередині об’єкту. Якщо розробник допустив помилку, і не відслідкував полярність, може виникнути ситуація, що замість текстури ми будемо бачити дірку у об’єкті, або в гірших випадках повністю прозорий об’єкт.
9. Налаштування «смуз груп» (англ. Smooth Group) – це процес виставлення правильного затінення на об’єкті. При моделюванні

немає сенсу робити максимально реалістичну геометрію там, де можна обманути око людини (наприклад, зробити пласку поверхню заокругленою). Ось декілька методів налаштування смуз груп:

1) Shade Smooth – виставлення 1 смуз групи на весь об’єкт. Зазвичай не використовується, адже тоді це змушує об’єкт виглядати дуже нереалістично.
2) Shade Auto Smooth – виставлення різних смуз груп за кутом нахилу полігонів відносно одне одного, наприклад: якщо була задана команда «Shade Auto Smooth by 30 degrees», то програма виставить різне затінення між полігонами, кут нахилу яких відносно одне одного є більше 30 градусів. Також можна комбінувати цей метод з іншим інструментом – «Mark Sharp». Використавши цей інструмент на ребрі, ви задаєте правило, що це ребро буде жорстким і 1 смуз групи між 2 полігонами, до яких це ребро належить, не буде.

Рис. 3.5 Результати роботи Shade Smooth (ліворуч) та Shade Auto Smooth (праворуч)

10. UV-розгортка: для того, щоб у майбутньому обрані текстури відображалися коректно, потрібно перевірити, чи правильно була розгорнута модель. Шви розгортки (місця, де поверхня розривається)

мають бути між площинами, які явно знаходяться під немалим кутом (якщо візуально кут між площинами невеликий, то в деяких випадках там не ставлять шов). Потрібно уникати розміщення шва на пласких поверхнях, бо при створенні текстури на поверхні буде видно цей шов. Також потрібно розгортати об’єкт таким чином, щоб утворені після цього частини не були розтягнуті та відділені повністю (потрібно уникати ситуацій, коли в одному UV-острівці знаходиться шматок іншого острівця, це призводить до рваного вигляду текстури у майбутньому). Приклад UV-розгортки можна побачити на Рисунку 3.6 та у Додатку Б. Також потрібно випрямляти округлі острівці для уникнення ефекту «драбини» на текстурі у майбутньому (якщо UV-розгортка не забезпечує оптимальне розміщення текселів текстури на поверхні моделі, це може привести до низької роздільної здатності, яка збільшує розмір текселів і викликає ефект драбини на текстурі) та оптимізації UV-простору.

Рис. 3.6 UV-розгортка об'єкту «Full_Can_LP»

11. Texel density - це поняття, що використовується в комп'ютерній графіці для позначення "текстурного елемента". Тексель (texel) вимірюється в пікселях на текстурі. Оскільки текстури можуть мати

різну роздільну здатність (кількість пікселів на одиницю довжини або площі), то розмір текселів може також варіюватися.

Наприклад, якщо текстура має роздільну здатність 1024x1024 пікселів, то кількість текселів у текстурі буде 1024 x 1024 = 1,048,576. Текстури в комп'ютерній графіці зазвичай складаються з пікселів, але відмінність полягає в тому, що тексель - це піксель на текстурі, а не на екрані. Для досягнення естетичного результату потрібно враховувати цей параметр для досягнення 2 цілей:

a. Збільшення якості текстури на об’єкті;
b. Досягнення приблизно однакового текселю в композиції (наборі декількох об’єктів у сцені). Це потрібно для уникнення «мильного» ефекту: коли на великих об’єктах стоїть занадто маленька текстура, і по відношенню до іншого об’єкту великий об’єкт виглядає гірше.

При створенні моделі потрібно враховувати як мінімум 2 вимоги, які завжди стоять перед розробником: перше – це естетичність об’єкту, об’єкт має виглядати згідно контексту, в якому він знаходиться (реалістичне середовище чи стилізоване). Друге – це оптимізація моделі для економії ресурсів. У цій постановці задачі виникає дилема: для того, щоб зробити модель естетичнішою, вона має мати більш цікаві, не пласкі форми, тобто вимагає більшої кількості геометрії, і в такому випадку модель може потребувати сотні тисяч полігонів. Але з іншого боку нам потрібно економити ресурси, бо таких моделей у сцені може бути десятки, якщо не сотні.

У такому випадку в індустрії придумали алгоритм, який дозволяє створити естетичну модель без використання величезної кількості геометрії. Для цього використовуються згадані у розділі 1.2 PBR текстури. Конкретно питання великої кількості полігонів вирішує карта нормалей, або ж Normal Map. Для її створення ми маємо створити 2 моделі: Low Poly та High Poly.

Фактично ці моделі однакові за своєю основною формою, але вони мають відмінності. High Poly – це модель, яка створюється згідно ідеї і задуму розробника, з максимально можливою деталізацією, але великою кількістю геометрії. Low Poly – це модель, яка основною формою повторює High Poly, але при цьому має набагато меншу кількість полігонів і, відповідно, не має того рівня деталізації, яким володіє перша модель.

Рис. 3.7. Low Poly (ліворуч) та High Poly (праворуч)

В залежності від поставленої задачі, а також особливостей будови об’єкту, деякі деталі можуть бути відтворені геометрією на Low Poly для досягнення більш естетичного результату, зберігаючи баланс між економією ресурсів та зовнішнього вигляду моделі.

Після того, як 2 моделі були створені, потрібно пересвідчитися, що вони мають однакову основну форму. Для цього зазвичай потрібно:

- Виставити півот-поінти у однакових місцях;
- Перемістити об’єкти одночасно на початок координат у програмі для 3D моделювання;
- При правильному розміщенні та за умови, що об’єкти повторюють форму одне одного, має утворитися так званий «Z-fight» - явище, при якому поверхні полігонів обох об’єктів перетинаються між собою, що

спричиняє візуальний ефект, який схожий на білий шум. Це відбувається через те, що відео карта при розміщенні полігонів у однакових координатах не розуміє, який з двох полігонів потрібно підмальовувати першим;

- У разі якщо Z-fight не спостерігається, то потрібно перевірити, чи усі частини об’єктів перетинаються між собою та відредагувати їхнє положення або за допомогою алгоритмів, або вручну.

Варто зауважити, що практиці неможливо абсолютно точно витримати однакову основну форму, тому невелике відхилення допустиме.

Далі або у самому Blender або у сторонній програмі розробник починає етап «Запікання» - це процес перенесення інформації з High Poly на Low Poly, створюючи відповідні карти на основі UV-розгортки Low Poly. Вигляд роботи у сторонній програмі для запікання Marmoset Toolbag можна побачити у Додатку В. Запікання включає в себе такі етапи, як:

- Створення текстури - визначення матеріалів, текстур та освітлення на поверхнях 3D моделі.
- Обчислення світла - розрахунок того, як світло взаємодіє з поверхнями моделі.
- Збереження текстури - створення текстурного файлу з попередньо обчисленою інформацією про освітлення, колір та матеріали на поверхнях 3D моделі.

Для того, щоб обчислити інформацію про перепади висот на об’єкті, і при цьому не утворити помилки від неточного повторення форми High Poly відносно Low Poly, використовується так званий «Cage» - область, яка утворюється по напрямку нормалей відносно поверхні Low Poly моделі і захоплює усі деталі High Poly, які власне у цей Cage потрапили. Після того, як текстури були створені, вони можуть бути використані під час рендерингу, що дозволяє ефективно відображати освітлення та матеріали на поверхнях 3D моделі без потреби повторно обчислювати їх для кожного кадру.

Blender довів, що робота над графікою може бути проста та ефективна, а також те, що безкоштовний продукт може бути конкурентноздібним.

3.2. Особливості роботи в програмах для текстурування Substance Painter

Після створення текстурних карт та підготовки основного об'єкту (Low Poly) до роботи, можна переходити власне до створення текстур. Для роботи я обрав програму Substance Painter від компанії Adobe.

Substance Painter є програмою для розфарбування та текстурування 3D-моделей. Ця програма має декілька потужних можливостей, які дозволяють користувачам легко створювати фотореалістичні текстури та матеріали для своїх 3D-моделей. Ось деякі з можливостей, які доступні у Substance Painter:

- Підтримка PBR - Substance Painter підтримує фізично правильний рендеринг (PBR), що дозволяє користувачам створювати текстури та матеріали, які виглядають реалістично незалежно від умов освітлення.
- Smart Materials – у програмі є колекція матеріалів Smart Materials, які дозволяють користувачам швидко та легко створювати реалістичні матеріали з високою якістю.
- Texture Baking - Substance Painter має вбудований режим запікання, що дозволяє користувачам створювати текстури з низькою роздільною здатністю з високої роздільної здатності моделі.
- Підтримка різних форматів - програма підтримує різні формати текстур, включаючи PSD, PNG, TIFF, TGA та інші.
- Розширені інструменти малювання - Substance Painter має розширені інструменти малювання, що дозволяють користувачам легко створювати складні шари, малюнки та градієнти.
- Реалістичне освітлення - Substance Painter має інструменти для створення реалістичного освітлення на текстурах та матеріалах, що

дозволяє користувачам більш точно відтворювати світло на їх 3D-моделі.

- Можливість експорту - програма має можливість експортувати текстури та матеріали в різні формати, що дозволяє користувачам використовувати їх у різних програмах та на різних платформах.
- Підтримка 8K-роздільності - Substance Painter підтримує роботу з текстурами та матеріалами у роздільності 8K, що дозволяє користувачам створювати дуже деталізовані та реалістичні текстури для своїх 3D-моделей.

У цьому розділі я опишу процес створення текстур у цій програмі. Для початку нам потрібно створити проєкт, в якому ми будемо працювати. Для цього потрібно:

- Завантажити модель, над якою буде вестися робота;
- Обрати розмір текстури (1024, 2048, 4096, і тд);
- Обрати режим відображення нормалей: Direct X чи Opengl;
- Завантажити наявні ресурси: текстурні карти, текстури і тд.

Принцип роботи програми схожий на принцип роботи Photoshop: матеріали створюються «шарами», а самі шари можна змінювати за допомогою різноманітних функцій. Під час роботи над об’єктом для зручності додані можливості повертати його навколо своєї осі, переміщати камеру відносно нього, наближатися та віддалятися, а також змінювати положення світла у сцені. Робочий процес над текстуруванням моделі можна побачити на Рисунку 3.8 та у Додатку Д.

Також для розуміння процесу текстурування у Viewport (вікні, де відбувається відображення об’єкту) можна змінювати режими відображення. Це дозволяє бачити не тільки усю комбінацію налаштованих шарів, а і кожен окремий канал: наприклад режим Mask дозволяє бачити лише інтенсивність нанесення маски на об’єкт, а режим Base Color – тільки колір об’єкту. При створенні текстур може виникнути необхідність поміняти їх розмір:це можна

зробити, не створюючи проєкт заново – наявні шари під лаштуються під заданий розмір текстур.

Рис. 3.8 Робочий процес у Substance Painter

У програмі є так званий Shelf – це бібліотека матеріалів, пензлів, текстур, та інших ресурсів, які можна між собою комбінувати для досягнення унікального результату. До інструментів Substance Painter належать також генератори текстур, які дозволяють автоматично створювати різноманітні текстури, такі як дерево, метал, шкіра, камінь та інші (їх можна знайти у Shelf).

Якість текстури напряму залежить від обраного розміру, який підбирається згідно текселя у сцені. Якщо на великому об’єкті буде замала текстура, то загальний її вигляд буде «мильним», а окремі текселі будуть виглядати неестетично через ефект «драбини».

Після завершення роботи над текстурами розробник має експортувати їх у вигляді файлів для подальшого використання у роботі. Для цього він має обрати так званий «пресет» - набір параметрів та правил, за якими буде формуватися набір текстур. Кожна програма може потребувати свій власний набір необхідних каналів для коректного відображення текстури: наприклад, для Unreal Engine 5 достатньо каналів Base Color, AORM (Ambient Occlusion, Roughness, Metallic; ця карта містить декілька карт в собі одночасно, для їх розділення на окремі карти

використовуються канали RGB відповідно порядку назв), Normal, та Emissive, а для інших програм може бути більше. Окрім цього, в разі, якщо потрібно використати специфічний канал, його можна додати до списку на імпорт. Для зручності можна створити свій власний пресет, який буде включати в себе найрізноманітніші канали, які тільки потрібно.

Дана програма дає зрозуміти, що створення текстур може бути не тільки ефективним, але і зручним. Це дозволяє не обмежувати себе і реалізовувати найбільш унікальні ідеї.

3.3. Особливості роботи у ігровому рушії Unreal Engine 5

В контексті випускної кваліфікаційної роботи я сфокусуюся на особливостях роботи у рушії з графікою. Потрібно розуміти, що функціонал рушія не обмежується лише розміщенням 3D моделей у сцені, і у ньому можна працювати з програмуванням об’єктів, сценаріїв і тд. Є можливість працювати з реалістичною фізикою, яка буде впливати на поведінку «акторів» у сцені, а також при необхідності на наявні об’єкти, також доступна можливість працювати з різними типами анімації та рендером.

Для початку роботи над реалізацією ігрового рівня потрібно створити проєкт. У програмі є різного роду «пресети», які дозволяють зекономити час на підготовку до роботи: кожним пре сет має свої об’єкти, акторів, освітлення, тощо. Ось декілька прикладів:

- Ігри: найголовніша вкладка містить у собі декілька основних пресетів для створення різноманітних типів гри – First Person (гра від 1 обличчя), Third Person (гра від 3 обличчя), Top Down (вид зверху вниз), Vehicle (набір функцій, який дозволяє зекономити час для створення транспорту в грі), Handheld AR (створення доповненої реальності), VR (гра у стилі Віртуальної реальності), а також Blank –

це пустий рівень, в якому виставлені лише налаштування за замовчуванням (саме такий пресет обрав я).

- Film/Video & Live events: це вкладка, де знаходяться пресети для створення фільмів, відео та контенту для живих зустрічей та прямих трансляцій.
- Architecture: це вкладка для пресетів, які дозволяють візуалізувати архітектурні або інтер'єрні проєкти.
- Automotive Product Design & Manufacturing: це вкладка для пресетів, які розроблені для демонстрації автомобільної техніки та будь-якого продуктового дизайну.

Після того, як був обраний пресет, потрібно розмістити основні Level Instances – це структури, які дозволяють контролювати процеси на рівні, або формують цей самий рівень. Ось основні з них:

- Landscape – за допомогою Режиму редагування ландшафту у рушії можна створити ландшафт, не створюючи для цього окрему модель в окремій програмі. За допомогою наявних інструментів, таких як Sculpt, Erase, Smooth, Flatten та інших можна зручно редагувати наявний ландшафт під поставлені задачі. Це дозволяє суттєво зекономити час при роботі над рівнем, а також відредагувати окремі ділянки ландшафту в разі непередбачуваних змін структури рівня [17].
- BP Sky Sphere в UE5 - це готовий до використання компонент для створення реалістичного неба та атмосферного освітлення у проєкті. Він містить в собі всі необхідні матеріали, текстури та налаштування, які дозволяють створити реалістичний ефект неба та освітлення без необхідності вручну створювати всі необхідні матеріали та текстури. BP Sky Sphere дозволяє легко створювати різні ефекти неба та освітлення, такі схід та захід сонця, хмарні дні та ночі, різні кольорові палітри тощо.
- Directional Light - це джерело освітлення в ігровому рушії Unreal Engine 5, яке дозволяє створювати реалістичне освітлення у вашому проєкті. Він

імітує світло, яке походить від сонця та віддаленого джерела, і може бути використаний для створення денного та нічного часу. Наявна можливість налаштувати його колір, інтенсивність, напрямок, властивості та тіні, які він кидає на об'єкти в сцені [15].

- Exponential Height Fog - це ефект, який відображає туман у ігровому проєкті. Є можливість налаштувати його властивості, такі як густота туману, колір туману, висота туману та інші, для створення різних ефектів [16].
- Post Process Volume - це об'єкт в Unreal Engine 5, який дозволяє застосовувати різноманітні ефекти пост-процесу. Цей об'єкт можна використовувати для додавання різних ефектів, таких як корекція кольору, блюр, кольоровий градієнт, кольорові фільтри, а також для налаштування різних параметрів освітлення і тіней в грі. Post Process Volume можна розмістити в будь-якому місці гри або іншого проєкту, і він буде застосовуватися до всіх об'єктів, які перебувають в зоні дії об'єкта [22].
- Sky Atmosphere - це нова функція в Unreal Engine 5, яка дозволяє створювати реалістичне небо. SkyAtmosphere дозволяє детально налаштувати кольорову палітру, насиченість, нахил і скупченість хмар, що дає можливість створювати різноманітні реалістичні погодні умови, такі як ясний сонячний день, хмарний день, сутінки, заходи сонця і т.д. Він також використовує більш ефективну модель розсіяння світла, що дозволяє зменшити обсяги розрахунків та забезпечити високу продуктивність [24].
- Sky Light - це один з основних інструментів освітлення в Unreal Engine, який дозволяє створювати реалістичне небесне освітлення. Він доповнює інші елементи освітлення, такі як Directional Light і Point Light, дозволяючи змінювати колір, насиченість та інтенсивність небесного освітлення в залежності від погодних умов та часу доби. SkyLight працює шляхом відбивання світла від неба та інших об'єктів в ньому, таких як хмари, туман та інші. SkyLight також дозволяє створювати динамічне освітлення, яке змінюється в залежності від часу доби та погодних умов [25].

- Spherical Reflection Capture - це інструмент в Unreal Engine, який дозволяє створювати реалістичні відображення об'єктів на поверхнях, що відображають світло. Він використовується для створення реалістичної взаємодії світла з поверхнями в реальному часі. Spherical Reflection Capture працює шляхом створення кубової карти зображення з точки зору кулі, яка відображає світло на поверхні в реальному часі. Ця карта зображення зберігається у вигляді текстури, яку можна використовувати для створення реалістичних відображень на поверхнях [23].

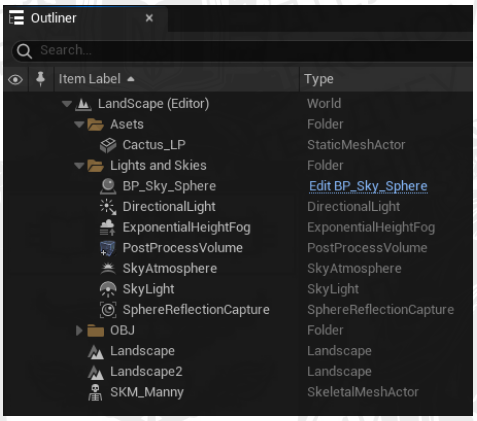


Рис. 3.9. Level Instances

Коли усі необхідні Level Instances були розміщені на рівні та налаштовані розробником, то залишається тільки додати створені раніше 3D моделі у сцену. Перед цим для полегшення задачі розробник може скористатися внутрішніми можливостями рушія для блок-ауту. Після того, як розробника влаштовує виставлена ним композиція, він виставляє замість них вже створені об’єкти, які і будуть основою фінального продукту – ігрового рівня.

Перед безпосереднім додаванням об’єкта на рівень розробник бачить вікно, у якому можна виставити параметри, згідно з якими об’єкт буде доданий. В ці параметри входять декілька десятків пунктів, особливо нас цікавлять наступні:

- Build Nanite – включення функції Nanite.
- Generate Missing Collision – якщо колізію (структури, які використовуються для визначення того, як об'єкти повинні реагувати на зіткнення, наприклад, чи вони повинні взаємодіяти, зіткнутися і

зупинитися) не створив розробник, рушій автоматично створить цю колізію самостійно.

- Normal import method – вибір даних, який визначає, як саме рушій має розуміти нормалі об’єкту.
- Transform – набір параметрів, який визначає, з якими змінами у масштабі та повороті об’єкт буде імпортуватися у проєкт.
- Material import method – параметр, який визначає, чи створювати матеріал з тих даних, що рушій бачить у файлі, чи ні. Зазвичай вказується, щоб рушій не створював матеріал, бо текстури для об’єкту створюються окремими файлами.

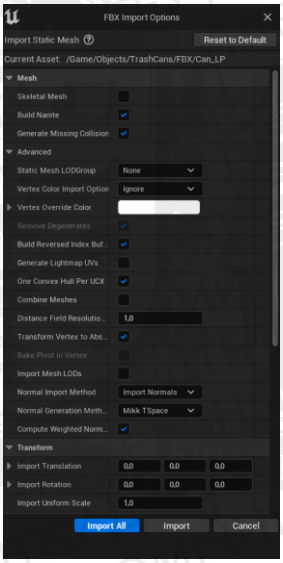


Рис. 3. 10. Вікно імпорту об'єкта у сцену

Імпортувавши об’єкт до рушія, ми вже можемо ним користуватися, але для досягнення фінального результату у вигляді повноцінної моделі нам потрібно додати текстури. Імпортувавши текстури, розробник не може просто так їх назначити на об’єкт, для цього потрібно створити матеріал.

Вся проблема полягає у тому, що кожен об’єкт, якщо він не розміщений декілька разів у сцені, потребує окремого матеріалу, а створювати заново кожен матеріал буде контрпродуктивно. Саме тому розробники створюють так звані «Майстер матеріали» – це матеріали, які створюються один раз під конкретний набір об’єктів, для яких підійде дана конфігурація матеріалу. Вона створюється за

допомогою «нод» – це структури, які замінюють строчки програмного коду візуальною складовою, і дозволяють зручно налаштовувати взаємодію текстур між собою. Для стандартного матеріалу у моєму проєкті використовується наступна схема: є головна нода матеріалу, в якій міститься набір «Входів» – функції, які приймають дані з текстур і виконують свою особливу задачу (наприклад, вхід Base Color відповідає за те, які дані показувати у вигляді кольору, Roughness відповідає за те, в яких місцях буде відображено більш гладку або грубу поверхню, і тд). В неї входять ноди текстур, в які вже власне можна додати або текстури за замовчуванням, або «пробки» - пусті текстури, які дають програмі зрозуміти, що нода не пуста (в інакшому випадку буде виникати помилка). У кожної з цих нод є різні виходи у вигляді каналів: RGB (повний набір кольорів), окремо R, G, та B (червоний, зелений та синій відповідно), A (канал прозорості) та RGBA (усі канали в одному). В залежності від задачі розробник може підключати виходи до входів, комбінуючи різні порядки підключення та досягаючи таким чином різних результатів.

Наприклад, замість того, щоб використовувати 3 різні текстури для Ambient Occlusion, Roughness та Metallic, можна використати текстуру, яку ми отримали у Substance Painter (AORM, див. розділ 3.2.) та підключити виходи окремих каналів R, G та B відповідно.

Для регулювання правильності накладення текстури на об’єкт, використовується комбінація нод «TexCoord»(визначає і передає текстурні координати), «UV»(працює з UV-розгорткою об’єкту) та «Multiply»(приймає 2 аргументи та множить їх), і результат перемноження 2 перших нод у третій останній передається на вхід «UVs» у нодах текстур.

Після того, як матеріал був налаштований згідно задуму розробника, він зберігається і береться за основу для створення Material Instance – матеріал, який вже призначається для окремого об’єкту. В цьому матеріалі вже можна додати необхідні текстури та регулювати параметри для їхнього правильного накладення на об’єкт, після чого матеріал зберігається і накладається на об’єкт.

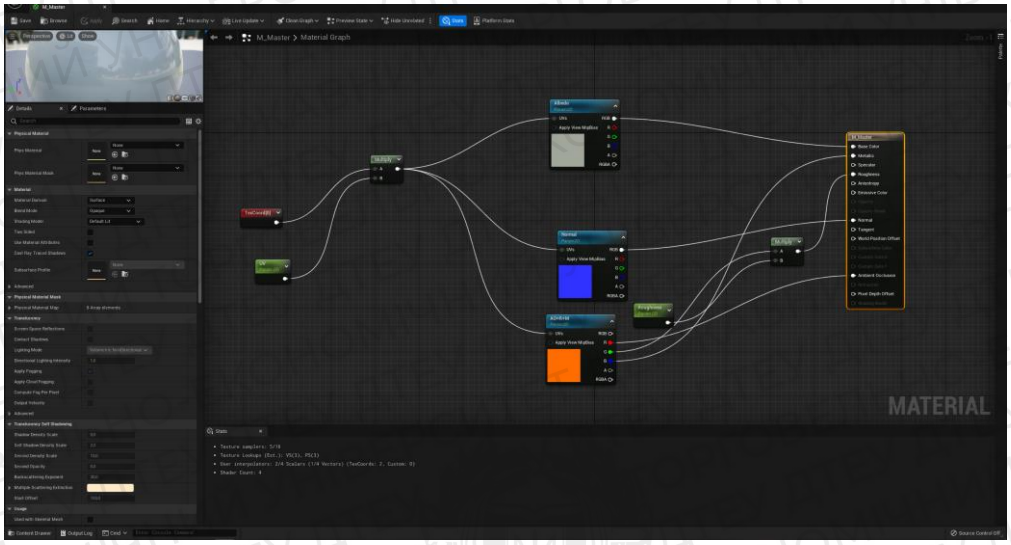


Рис. 3.11 Меню налаштування Master Material

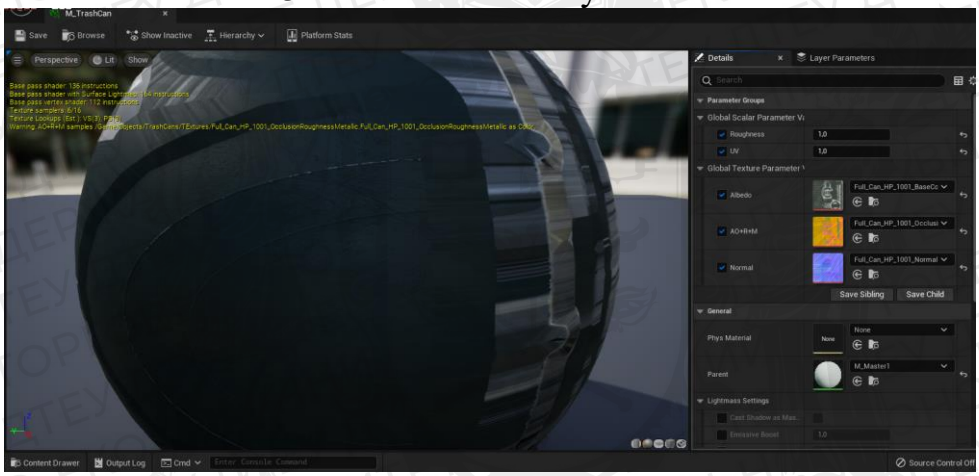


Рис. 3.12. Меню налаштування Material Instance

Під час роботи над сценою функціоналу польоту камери розробника у просторі може бути недостатньо, тому попри фокус на графічну складову доведеться описати і створити деяку структуру, пов’язану вже з програмуванням – Ігрового персонажа.

Персонаж буде примітивним, але ефективним з точки зору поставленої задачі: оцінка масштабу та розміщення об’єктів у сцені відносно поля зору Ігрового Персонажа, а також надання можливості демонстрації фінального продукту. Його модель складається з моделі, яка була додана у стартовий контент рушія (Manny або Quinn) та камери, яка розміщена у зоні голови моделі для імітації виду від першого обличчя. Також використовуються алгоритми, які були вбудовані у модель за допомогою BluePrints – аналог нодів, який відповідає за повноцінне програмування, але шляхом візуалізації програмного коду у вигляді

візуальних структур та зв’язків. Персонаж може бігати та стрибати, а також повертати камеру туди, куди вкаже гравець.

Рослинність на рівні буде створена шляхом окремого режиму Foliage – це режим, який дозволяє дублювати об’єкти і зручно розміщувати їх за допомогою віртуальних пензлів і не тільки. Це спрощує роботу розробнику, адже кожен кущ або кактус у сцені не потрібно розміщувати вручну. Достатньо лише завантажити в проєкт моделі та текстури рослин і обрати зручний спосіб розміщення.

Після розміщення усіх об’єктів у сцені, накладення текстур та налаштування усіх процесів освітлення та пост обробки розробник перевіряє сцену на наявність «артефактів» – помилок відображення, які псують фінальний вигляд об’єкта.

У висновку можна зазначити що сучасні технології рушія дозволяють комфортно і без зайвих зусиль створювати продукт, який з кожною новою ітерацією версії рушія буде мати менше компромісів при створенні ігор.

ВИСНОВКИ

1. Мета та завдання випускної кваліфікаційної роботи була досягнута: на базі ігрового рушія Unreal Engine 5 була створена ігрова локація, яка може бути використана у подальшому, як ігровий рівень. Всі заплановані до моделювання моделі були створені і розміщені на локації згідно із задумом.
2. Були досліджені технології в системі ігрового рушія Unreal Engine 5 та програм для створення 3D графіки. Використані методи дослідження показали, що сучасні технології 3D моделювання можуть бути простими та ефективними для поставленої задачі.
3. Створена локація утримала баланс між економією ресурсів та естетичною складовою: рівень не впливає критично на роботу комп'ютера і відтворюється без проблем. Моделі на локації адекватно реагують на віртуальне освітлення і виглядають згідно задуму.
4. Рекомендації: рівень можна розвивати з естетичної точки зору далі, тобто додавати нові об'єкти, які будуть доповнювати композицію. Особливо відчутна потреба в цьому в контексті дрібних деталей оточення та рослинності. Також варто продумати нові шляхи оптимізації, окрім використаних раніше технологій рушія та оптимізації геометрії об'єктів. Для цього необхідно перевіряти оновлення рушія до нових версій, у яких можуть додати новий функціонал, який власне і реалізує ці ідеї. Локація може бути використана у майбутньому як частина повноцінного ігрового рівня у комп'ютерній грі, але для цього потрібно поєднати локацію з іншими подібними елементами і реалізувати ігровий потенціал шляхом впровадження ігрових систем у вигляді програмного коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія: Light Map [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Lightmap
2. Вікіпедія: Текстура [Електронний ресурс]. - Режим доступу: https://uk.wikipedia.org/wiki/Текстура_(тривимірна_графіка)
3. Документація Blender: 3D Cursor [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/editors/3dview/controls/pivot_point/3d_cursor.html
4. Документація Blender: Extrude [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/extrude.html
5. Документація Blender: Introduction [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/modeling/modifiers/introduction.html
6. Документація Blender: Merge [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/merge.html
7. Документація Blender: Modeling Modes [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/modeling/meshes/introduction.html#modeling-modes
8. Документація Blender: Structure [Електронний ресурс]. - Режим доступу: https://docs.blender.org/manual/en/latest/modeling/meshes/structure.html
9. Документація CryEngine: Diffuse Maps [Електронний ресурс]. - Режим доступу: https://docs.cryengine.com/display/SDKDOC2/DiffuseMaps
10. Документація Substance Painter: THE PBR GUIDE - PART 2 [Електронний ресурс]. - Режим доступу: https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-2
11. Документація Unity: Normal map (Bump mapping) [Електронний ресурс]. – Режим

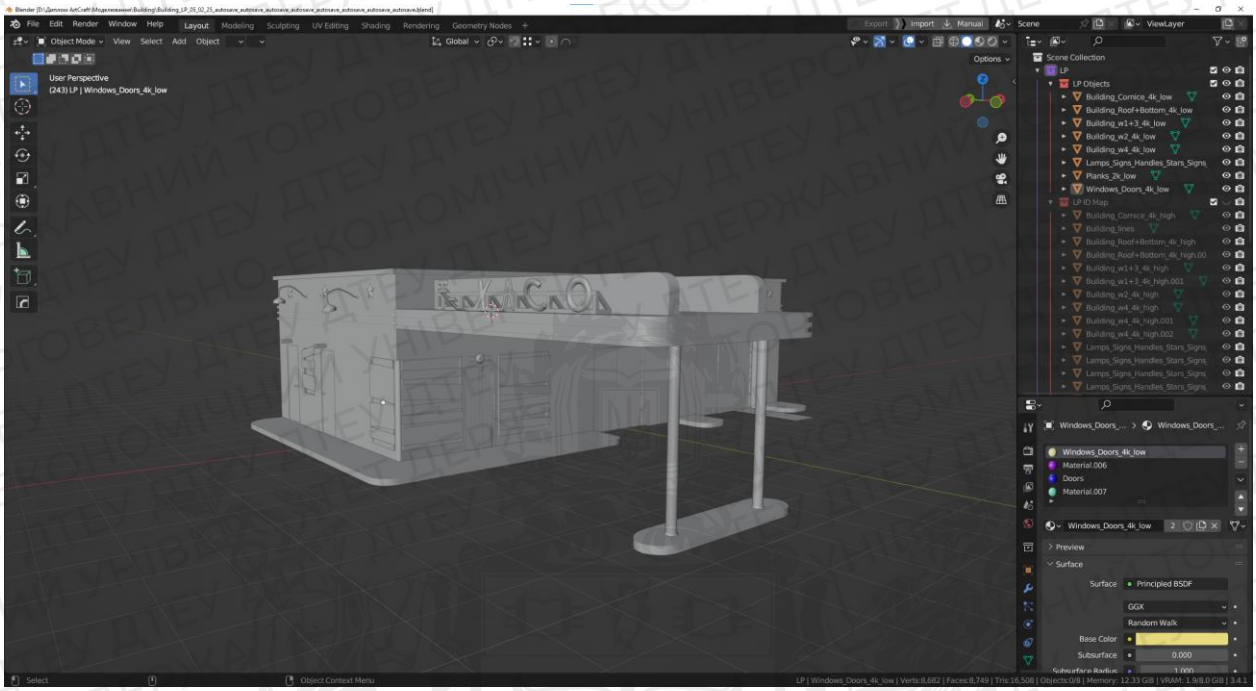
доступу:https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html

12. Документація Unity: Specular [Електронний ресурс]. - Режим доступу: https://docs.unity3d.com/Manual/shader-NormalSpecular.html
13. Документація Unreal Engine 5: Chaos Destruction [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/chaos-destruction-in-unreal-engine/
14. Документація Unreal Engine 5: Creating Visual Effects [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/creating-visual-effects-in-niagara-for-unreal-engine/
15. Документація Unreal Engine 5: Directional Lights [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/directional-lights-in-unreal-engine/
16. Документація Unreal Engine 5: Exponential Height Fog [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/exponential-height-fog-in-unreal-engine/
17. Документація Unreal Engine 5: Landscape Technical Guide [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/landscape-technical-guide-in-unreal-engine/
18. Документація Unreal Engine 5: Lightmass Basics [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/lightmass-basics-in-unreal-engine/
19. Документація Unreal Engine 5: Lumen Technical Details [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/lumen-technical-details-in-unreal-engine/
20. Документація Unreal Engine 5: Materials [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.1/en-US/unreal-engine-materials/

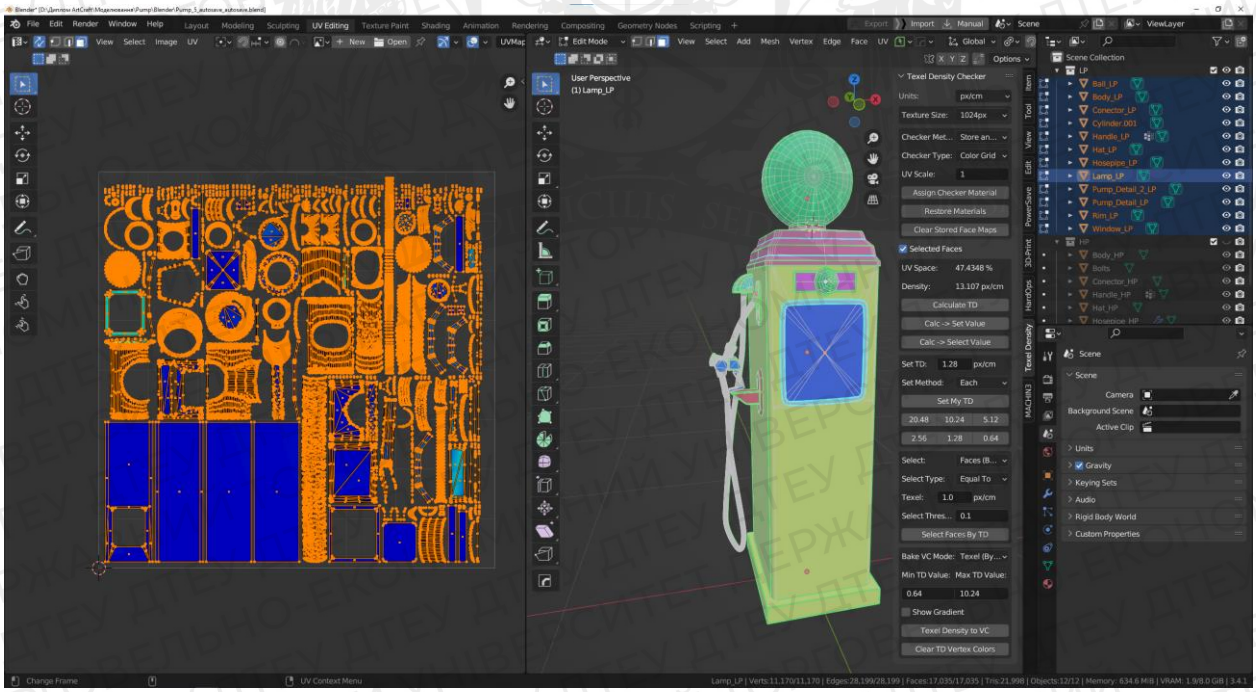
21. Документація Unreal Engine 5: Nanite Virtualized Geometry [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/
22. Документація Unreal Engine 5: Post Process Effects [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/post-process-effects-in-unreal-engine/
23. Документація Unreal Engine 5: Reflections Capture in Unreal Engine [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/reflections-captures-in-unreal-engine/
24. Документація Unreal Engine 5: Sky Atmosphere Component [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/sky-atmosphere-component-in-unreal-engine/
25. Документація Unreal Engine 5: Sky Lights [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.1/en-US/sky-lights-in-unreal-engine/
26. Документація Unreal Engine 5: Virtual Shadow Maps [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/virtual-shadow-maps-in-unreal-engine/
27. Документація Unreal Engine 5: Volumetric Fog [Електронний ресурс]. - Режим доступу: https://docs.unrealengine.com/5.0/en-US/volumetric-fog-in-unreal-engine/
28. Стаття «The Level Design Book: BlockOut» [Електронний ресурс]. - Режим доступу: https://book.leveldesignbook.com/process/blockout
29. Стаття «What is Temporal Super Resolution?» [Електронний ресурс]. - Режим доступу: https://cghero.com/glossary/what-is-temporal-super-resolution

ДОДАТКИ

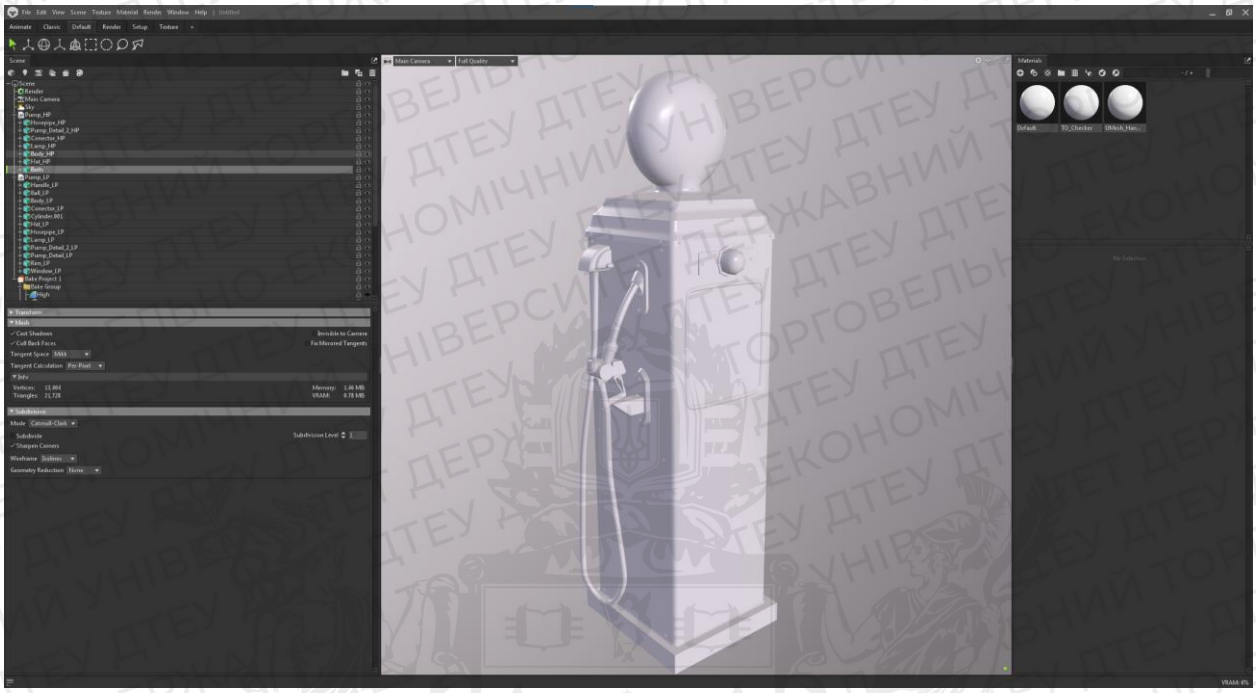
Додаток А



Додаток Б



Додаток В



Додаток Г

Додаток Д