Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Сеня Данила Дмитровича*
(ПІБ)

академічної групи *121-20ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення ігрової локації на базі середовища Unreal Engine*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Мещеряков. Л.І.*			
розділів:				
спеціальний	*проф. Мещеряков. Л.І.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст. преп. Мартиненко А.А*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ Сень.Д.Д
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20ск-1* *Сеня.Д.Д*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення ігрової локації на базі середовища Unreal Engine*

затверджена наказом ректора НТУ «ДП» від *18.05.202. №268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *проф. Мещеряков. Л.І.*
(підпис) (посада, прізвище, ініціали

Завдання прийняв до виконання _____ *Сень Д.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*
Термін подання кваліфікаційної роботи до ЕК: *13.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 99 с., 25 рис., 3 дод., 20 джерел.

Об'єкт розробки: програмне забезпечення для створення ігрової локації на базі середовища Unreal Engine.

Мета кваліфікаційної роботи: розробити програмне забезпечення для створення ігрової локації з використанням середовища Unreal Engine.

Структура та обсяг. Дипломна робота складається з вступу, трьох розділів та висновків.

У вступі проведено аналіз сучасного стану проблеми, уточнено постановку завдання, визначено мету кваліфікаційної роботи та галузь її застосування, обґрунтовано актуальність теми.

У першому розділі проведено дослідження предметної галузі та існуючих рішень, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання.

У другому розділі обрано платформу для розробки, виконано проектування програми та її розробку, наведено опис алгоритму та структури функціонування системи, визначено вхідні та вихідні дані, наведено характеристики параметрів технічних засобів та описано роботу програми.

У економічному розділі визначено трудомісткість розробленого програмного продукту, проведено підрахунок вартості роботи зі створення застосунку та розраховано час на його реалізацію.

У висновках проаналізовані отримані результати роботи.

Практичне значення полягає у розробці програмного забезпечення для створення ігрової локації на базі середовища Unreal Engine. Що демонструє етапи розробки сцен в ігрових рушіях, та розробку до них спеціалізованих додатків, для подальшої інтеграції в мультимедійні проекти.

Актуальність програмного продукту визначається великою популярністю комп'ютерних ігор та зростаючою важливістю комп'ютерної 3D-графіки останні роки. Розробка програмного забезпечення, що дозволяє створювати ігрові локації з використанням Unreal Engine, є актуальною та відповідає потребам розвитку галузі комп'ютерних ігор та візуального мистецтва.

Список ключових слів: ШЕЙДЕР, ІГРОВИЙ ДОДАТОК, МОДЕЛЮВАННЯ, ІГРОВИЙ ДВИГУН, 3D ГРАФІКА, UNREAL ENGINE, ІГРОВА ЛОКАЦІЯ.

ABSTRACT

Explanatory Note: 99 pages, 25 figures, 3 appendices, 20 references.

Development Object: Software for creating a game location based on the Unreal Engine environment.

Objective of the Qualification Work: To develop software for creating a game location using the Unreal Engine environment.

Structure and Scope: The diploma work consists of an introduction, three chapters, and conclusions.

The introduction analyzes the current state of the problem, clarifies the task, defines the objective of the qualification work and its application area, and justifies the relevance of the topic.

In the first chapter, research is conducted on the subject area and existing solutions, the relevance of the task and the purpose of the development are determined, and the task is formulated.

In the second chapter, a development platform is selected, program design and development are performed, an algorithm and the structure of the system's functioning are described, input and output data are defined, technical parameters are provided, and the program's operation is described.

In the economic chapter, the complexity of the developed software product is determined, the cost of application development work is calculated, and the time for its implementation is estimated.

The conclusions analyze the obtained results of the work.

The practical significance lies in the development of software for creating a game location based on the Unreal Engine environment, demonstrating the stages of scene development in game environments and the development of specialized applications for them, for further integration into multimedia projects.

The relevance of the software product is determined by the high popularity of computer games and the increasing importance of computer 3D graphics in recent years. The development of software that allows creating game locations using Unreal Engine is relevant and meets the needs of the computer game industry and visual arts.

List of keywords: SHADER, GAME ADD-ON, MODELING, GAME ENGINE, 3D GRAPHICS, UNREAL ENGINE, GAME LOCATION.

.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

OS -	операційна система;
GPU -	графічний процесор;
CPU -	центральний процесор;
ZIP -	формат архівації файлів і стиснення даних бе втрат;
Shader -	це програмний код, для обробки графіки у комп'ютерних програмах;
Skinning	- процес прив’язування скелету до 3D-моделі;
Highpoly	- високополігональна сітка 3D-моделі;
Lowpoly	- низькополігональна сітка 3D-моделі;
3D -	тривимірний.

ВСТУП

Тематика даної роботи пов'язана з розробкою ігрової локації на базі Unreal Engine 5 і її візуалізацією за допомогою шейдерів, а також використанням Unreal Engine 5 для створення різноманітних мультимедійних проектів.

Метою цього дослідження є вивчення необхідного інструментарію для створення ігрових локацій та розробки шейдерів на базі Unreal Engine 5 з метою досягнення високої якості візуалізації. Крім того, метою є розкриття можливостей Unreal Engine 5 як потужного інструменту для створення різноманітних мультимедійних проектів.

У сучасному світі розвиток ігрової індустрії супроводжується швидким розширенням технологій візуалізації, що дозволяє створювати вражаючі та реалістичні ігрові світи. Unreal Engine 5, як один з найпотужніших рушіїв для розробки ігор, надає широкі можливості для створення деталізованих та захоплюючих ігрових локацій.

Розробка ігрових локацій є складним завданням, що вимагає розуміння різних аспектів, включаючи геймплей, дизайн рівнів, моделювання, освітлення та візуалізацію. Однак, успішна реалізація вимагає також глибокого розуміння інструментарію рушія гри та можливостей, які він пропонує.

Одним з ключових аспектів розробки ігрових локацій є використання шейдерів - програмних модулів, які керують графічним процесором і забезпечують реалістичне відображення матеріалів, освітлення та спеціальних ефектів.

Мета цього дослідження полягає в детальному вивченні можливостей Unreal Engine 5 для створення ігрових локацій та розробці шейдерів, а також у досягненні високої якості візуалізації. Продовженням роботи буде розробка та реалізація конкретної ігрової локації з використанням розроблених шейдерів та демонстрація їхньої ефективності на практиці.

Розробка ігрових локацій та розробка шейдерів - це лише частина можливостей, які надає Unreal Engine 5. Цей двигун є потужним інструментом для створення різноманітних мультимедійних проектів, включаючи віртуальну реальність, архітектурну візуалізацію, тренінгові симуляції, відеоінсталяції, мультимедійні презентації та інші проекти.

Unreal Engine 5 надає розробникам потужні інструменти для моделювання складних та деталізованих об'єктів, створення фотореалістичних матеріалів, налаштування освітлення та створення різноманітних спеціальних ефектів. Завдяки своїй високоякісній графічній обробці, Unreal Engine 5 дозволяє створювати вражаючі та імерсійні мультимедійні проекти, які привертають увагу та захоплюють глядачів.

Крім того, Unreal Engine 5 має вбудовані інструменти для фізичного моделювання, анімації персонажів та штучного інтелекту, що дозволяє створювати живі та реалістичні світи із взаємодіючими об'єктами та персонажами. Він також підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність, що розширює його можливості та забезпечує гнучкість у розробці та розгортанні мультимедійних проектів.

Завдяки широкому спектру функціональності та потужним інструментам, Unreal Engine 5 відкриває безліч можливостей для творчості та інновацій у сфері розробки мультимедійних проектів. Він дозволяє розробникам втілювати свої уявлення і створювати захоплюючі та вражаючі віртуальні світи, що взаємодіють з глядачем та створюють неповторні емоції та враження.

Основною метою даної роботи є детальне вивчення інструментарію Unreal Engine 5 для розробки ігрових локацій та розробки шейдерів. Дослідження включатиме аналіз функціональних можливостей рушія гри, вивчення процесу створення ігрових локацій та розробки шейдерів з використанням Unreal Engine 5, а також практичну реалізацію конкретної ігрової локації з використанням розроблених шейдерів.

Результатом виконання даної роботи буде готова ігрова локація з високоякісною візуалізацією, що дозволить дослідникам та розробникам використовувати цей досвід для подальшого удосконалення графічних можливостей у галузі ігрової розробки. Крім того, результати дослідження можуть бути корисні у створенні інших мультимедійних проектів, що використовують Unreal Engine 5.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Тривимірна (3D) графіка є важливим розділом комп'ютерної графіки, який дозволяє створювати зображення, відео та віртуальні персонажі з об'ємними образами за допомогою моделювання об'єктів у трьох вимірах. Основна мета 3D-моделювання полягає в розробці візуально об'ємних образів об'єктів. Моделі можуть бути створені як автоматично, так і вручну.

Процес ручного моделювання для 3D комп'ютерної графіки схожий на пластичне мистецтво, де виконавець створює геометричні дані об'єкт. Також існує можливість фізичного створення 3D-моделей за допомогою пристроїв для 3D-друку, які формують модель пошарово з тривимірним матеріалом.

У процесі створення 3D-моделі необхідно дотримуватися певних правил та інструкцій, щоб забезпечити точне та своєчасне виконання завдання. Цей правильний порядок дій називається пайплайном, який включає в себе стандарти та оптимізує час і ресурси на кожному етапі створення моделі. Він вирішує різноманітні технічні та художні завдання, такі як вибір стилю моделі, обмеження полігонів, наявність карт нерівностей, методи текстурингу, роздільна здатність текстур, анімація моделі та інші.

В ігровій індустрії процес розробки 3D-сцени починається з блокування та закінчується готовою моделлю всередині проекту. Одним з найважливіших питань, що виникають при створенні ігрового пайплайну, є вибір між стилізацією та реалізмом, оскільки вони визначають загальний вигляд та атмосферу гри.

Розуміння загальних відомостей з предметної галузі 3D графіки є важливим кроком у розробці ігрових сцен та використанні комп'ютерної графіки. Ці знання надають основи для подальшого дослідження та розвитку цієї захоплюючої галузі, яка має великий потенціал у сучасному світі розваг та віртуальної реальності.

Основним завданням реалізму є імітування життя. Такий об'єкт не обов'язково існує у реальності, але він має виглядати реально. Реалізм нашого часу досить сильно

відрізняється від того, що було 20 років тому. На прикладі ігор 1993 року та 2023, можна побачити, як візуальна та технічна якість покращилась (рис.1.1) та (рис.1.2).

Рис. 1.1. Приклад графіки в іграх 1993 року

Рис. 1.2. Приклад графіки в іграх 2024 року

Прикладами підвищення якості деталей в моделях є їх кількість, збільшена за рахунок PBR, якого не існувало ще 10 років тому.

Основні принципи реалізму полягають в тому, щоб найбільш точно передавати суть об'єктів з найбільшою кількістю деталей. Зазвичай реалізм використовують у медіа. В ігровій індустрії реалістична графіка з'явилась нещодавно завдяки модернізації ПК, які набули здатність обробляти необхідну для реалістичної моделі кількість полігонів. Див. рис. 1.3.

Рис. 1.3. Приклад розвитку реалізму

Створення реалістичного оточення в іграх є складним та цікавим процесом, який включає кілька етапів і вимагає уваги до деталей та творчого підходу. Для досягнення реалізму в ігрових оточеннях використовуються певні кроки та технології.

- Вибір концепту та створення дошки референсів:

Перш за все, необхідно зрозуміти, яке оточення потрібно створити. Здійснюється пошук основного концепту та збір додаткових референсів. Концепт-арт та дошка референсів використовуємо для визначення стилю, кольорів, архітектурних деталей та інших елементів оточення. Цей етап допомагає встановити основні параметри та візуальний напрямок роботи.

Приклад концепт-арту для 3D-художника наведено на рис. 1.4.

Рис. 1.4. Приклад концепту

- Блокаут:

На початковому етапі виконується блокування сцени або рівня. Це означає створення грубої форми об'єктів та їх розташування в просторі, щоб визначити композицію та розміщення елементів. Блок-моделі можуть бути простими геометричними формами, які є підказкою для подальшого деталізування.

- Скульптинг (або створення highpoly сітки):

На цьому етапі створюється та деталізується уся модель без обмежень за кількістю полігонів, сітка моделі настільки щільна, що можна ліпити все, що завгодно, як зі шматка пластиліну. Високополігональна сітка потрібна лише для запікання деталей на lowpoly. Приклад highpoly моделі наведено на рис. 1.5.

- Ретопологія (або створення lowpoly сітки):

Завдання ретопології - знайти ідеальний баланс між виразністю та простотою моделі. Для цього кожний елемент моделі повинен мати функціональне завдання: впливати на силует або виправляти відблиск.

На цьому етапі створюється та сама модель, що буде завантажуватись в ігровий двигун. Приклад такої моделі наведено на рис. 1.6.

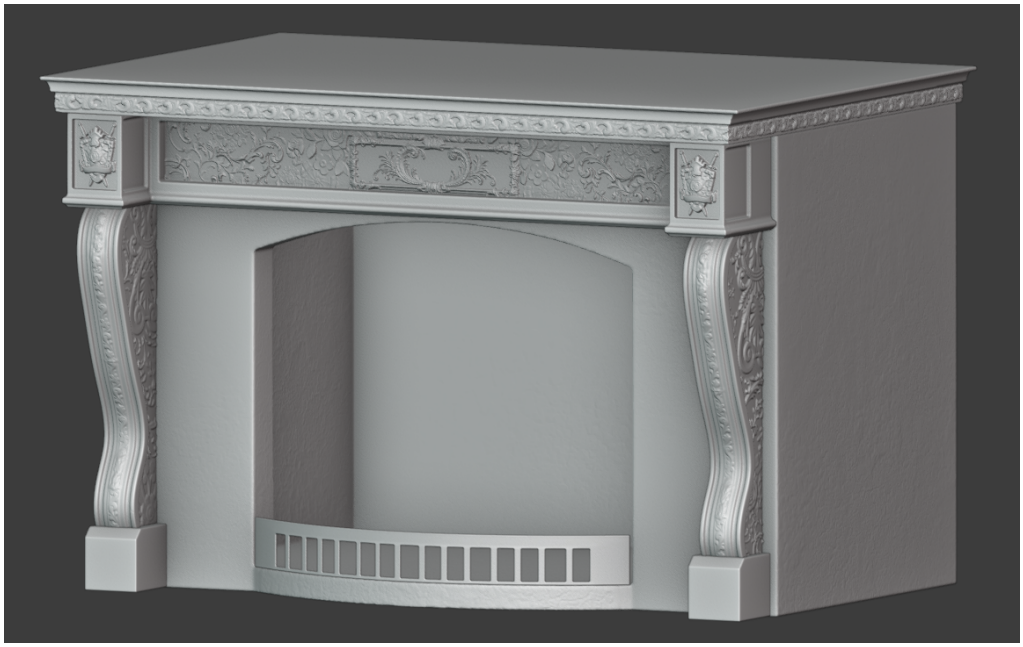

Рис 1.5. Приклад високополігонального ассету

- Розгортка:

На цьому етапі розгортається lowpoly модель на площину для того, щоб запечені карти та текстури, які являються простими плоскими картинками, могли застосуватися до об'ємної моделі. Приклад розгортки наведено на рис 1.6.

- Baking maps:

Етап перенесення деталізації з високополігональної сітки на низькополігональну. На цьому етапі запікають карту нормалей, що імітує високу деталізацію на оптимізованих низькополігональних моделях. Також з високополігональної моделі запікають карту тіней та ID-карту, яка допомагає при текстуруванні. Приклад карт наведено на рис. 1.6.

- Створення текстур:

Етап фарбування низькополігональної моделі. Існує кілька технологій візуалізації картинки зі своїми вимогами до текстур. Наприклад у старих іграх малювали карту кольору з вшитим у неї світлом і тінями, і карту відблисків, але в сучасних проектах ААА використовується фізично коректний рендер — PBR.

Приклад текстур та моделей наведено на рис. 1.7.

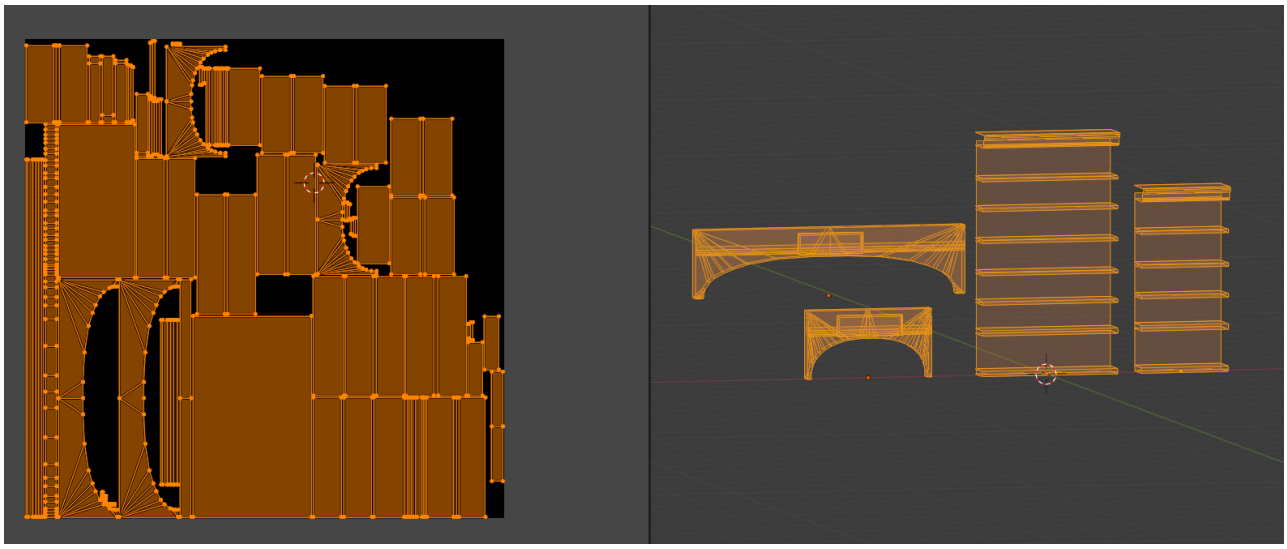

Рис. 1.6. Приклад ретопології та розгортки ассетів

- Імпорт у двигун:

На етапі імпорту в двигун основною задачею є налаштування моделі, текстур та анімацій, щоб усе працювало як потрібно і об’єкт виглядав відповідно концепту. Приклад наведено на рис. 1.8.

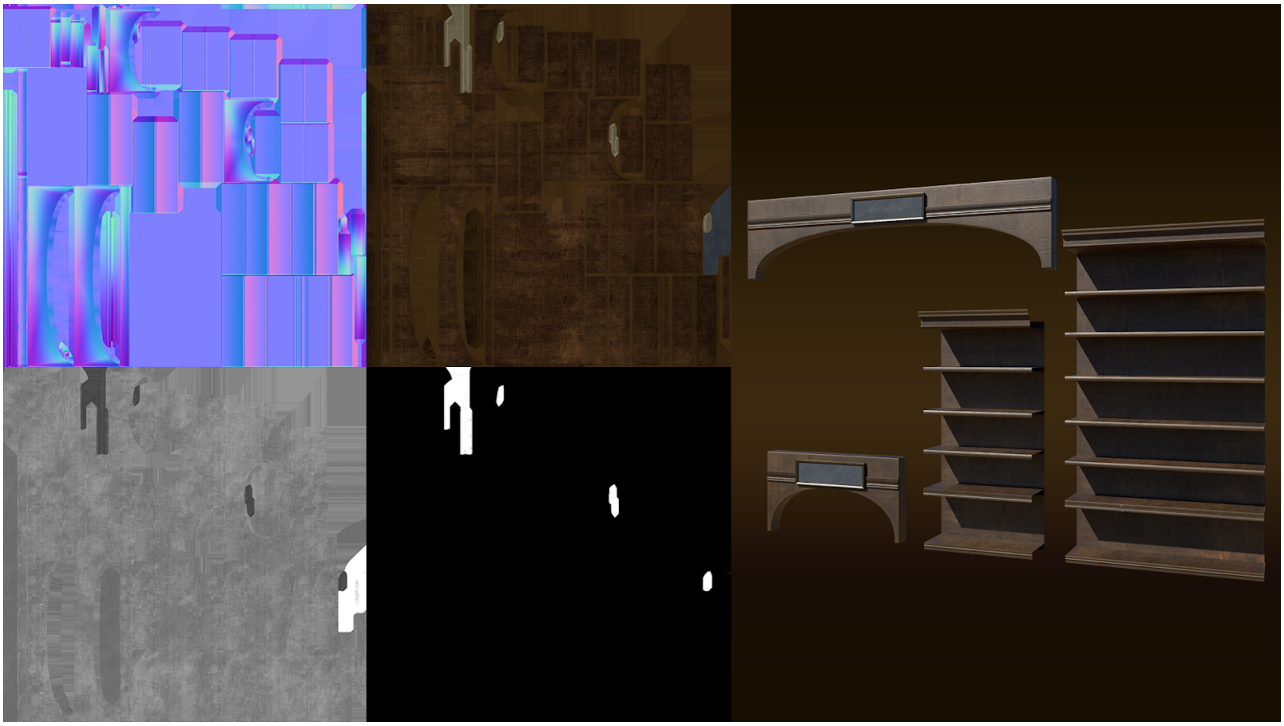

Рис 1.7. Приклад текстурних карт

Рис 1.8. Фінальний вигляд моделей в двигуні

Змінюються програми та технічні вимоги, проте порядок етапів зазвичай залишається незмінним. Правильно виконані етапи дають на виході якісну та оптимізовану під двигун 3D-модель.

1.2. Призначення розробки та галузь застосування

На сьогодні 3D-графіка використовується в багатьох галузях:

- У медичній промисловості використовуються докладні моделі органів, що формуються на основі багатьох 2D-зрізів отриманих зображень, отриманих з МРТ або КТ.
- У науковій сфері використовуються високодеталізовані моделі хімічних сполук.
- У галузі архітектури 3D-моделі використовуються для візуалізації та демонстрації проектованих будівель та ландшафтів замість традиційних фізичних архітектурних моделей.
- Археологічна спільнота створює 3D-моделі культурної спадщини з метою дослідження та візуалізації.
- Кіноіндустрія використовує 3D-моделі для створення анімаційних та

реальних фільмів, включаючи персонажів та об'єкти.

- Галузь відеоігор використовує 3D-моделі як активи для комп'ютерних і відеоігор, що дозволяє створювати високоякісні та реалістичні ігри, розширюючи можливості та свободу користувачів.

Завдяки поліпшенню графіки у відеоіграх стає легше переносити отриманий досвід у реальне життя. Набагато простіше співпрацювати з персонажем гри, якщо він майже неминуче нагадує людину, а не абстрактні полігональні моделі з нанесеними текстурами обличчя.

Одним з прикладів користі відеоігрового досвіду є поліпшення прийняття рішень. Дослідження показують, що шутери, такі як "Counter Strike", розвивають "низькорівневе сприйняття" та реакцію на непередбачувані об'єкти, спонукаючи гравця вживати відповідних дій. З іншого боку, ігри жанру "інтерактивне кіно", як наприклад "Beyond: Two Souls", у яких прийняття рішень впливає на фінал гри, добре моделюють ситуації, де необхідно швидко зробити важливий вибір, що визначає долю персонажа та неперсонажних героїв, розвиваючи розуміння причинно-наслідкових зв'язків.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2023 р;

– завдання на кваліфікаційну роботу на тему “Розробка програмного забезпечення ігрової локації на базі середовища Unreal Engine”

1.4. Постановка завдання

Завданням даної роботи є розробка ігрової локації на базі середовища Unreal Engine5 та створення візуального стилю проекту за допомогою 3D-графіки. В результаті необхідно спроектувати та розробити ігрову локацію для платформи Windows та за допомогою 3D-графіки створити оточення, та розробити шейдери для використання у проекті.

Поставлена задача буде досягнута при виконанні таких умов:

- вивчення предметної галузі;
- моделінг, анімація та інтегрування оточення в двигун Unreal Engine 5.
- Розробка фізично коректних шейдерів для реалістичної передачі світла, тіней, та кольорів оточення.
- Налаштування освітлення у сцені, з використанням новітньої системи Lumen.
- Створення PBR матеріалів з урахуванням особливостей розроблених шейдерів, та розміщення освітлення.

1.5. Вимоги до програми або програмного виробу

1.5.1.Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

- підтримка PBR текстур;
- фізично коректна робота зі світлом та тінями;
- наявність демонстраційної ігрової локації створеної на базі розроблених шейдерів;

1.5.2.Вимоги до інформаційної безпеки

Для забезпечення інформаційної безпеки користувача повинні бути забезпечені такі умови:

- своєчасне встановлення оновлень;
- відсутність необхідності користувачу реєструватися та створювати акаунт всередині проектів;

1.5.3.Вимоги до складу та параметрів технічних засобів

Відповідні характеристики необхідні для роботи додатку на Windows :

- OS: Windows 7 (SP1+) або вище
- CPU: Intel Core i3/i5/i7 1.8 GHz CPU dual-core. AMD 2.0 GHz dual-core;
- GPU: NVIDIA GeForce 260 / Radeon HD 4000 Series / Intel HD Graphics 4000;
- DirectX**:** Версії 10 або вище;
- накопичувач: 512мб;
- оперативна пам’ять: 4 Гб.

1.5.4.Вимоги до інформаційної та програмної сумісності

Додаток створений на основі ігрового двигуна Unreal Engine та призначений для використання на операційній системі Windows, а мовою програмування був обораний C++.

Додаток не потребує особливої процедури встановлення, а просто розпаковується з zip-архіву. Оновлення відбувається шляхом перекачування архіву з мережі інтернет.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Загальні відомості з предметної галузі

Результатом даної кваліфікаційної роботи має бути ігрова локація яка використовує розроблений шейдер, розроблений в двигуні Unreal Engine, та візуальний стиль проекту, створений за допомогою 3D-графіки.

Проект повинен продемонструвати можливості розробленого шейдеру та можливості його використання при створенні ігрових рівнів на базі Unreal Engine 5, та його роботу з системою Lumen яка максимально реалістично відтворює вплив світла на об'єкти

2.2. Опис застосованих математичних методів

При створенні фізично коректних шейдерів використовуються різні математичні методи, які дозволяють моделювати реалістичну поведінку світла та матеріалів. Деякі з цих методів включають:

- Модель Брансфорда-Нікалсона (англ. Lambertian Reflectance Model): Ця модель описує дифузне відбивання світла від матеріалів з матовою поверхнею. Вона базується на законі Ламберта, який стверджує, що інтенсивність відбитого світла пропорційна косинусу кута між нормаллю поверхні та напрямом світла. Вона вираховується за формулою:$I_D = \mathbf{L} \cdot \mathbf{N} C I_L$, де $\mathbf{I_D}$ - інтенсивність дифузно відбитого світла (яскравість поверхні), **C** позначає колір, а $\mathbf{I_L}$ - інтенсивність світла що падає.
- Модель Фонга (англ. Phong Reflection Model): Ця модель додає до дифузного відбивання такі компоненти, як відбитий блиск (specular highlight) та затінення (shading). Вона використовує вектори нормалі, напряму світла та спостереження для розрахунку інтенсивності відбитого світла. Вона вираховується за формулою: $I = K_a I_a + K_d(\vec{n}, \vec{l}) + K_s(\vec{n}, \vec{h})^p$, де $\vec{n}$ - вектор нормалі до площі у точці, $\vec{l}$ –

падаючий промінь (напрямок на джерело світла), $\vec{h}$ – відбитий промінь (напрямок променя ідеально відбивається від поверхні), $\vec{h} = 2(\vec{l} * \vec{n})\vec{n} - \vec{l}$, K_a – коефіцієнт фонового освітлення, K_s – коефіцієнт відблисків, K_d – коефіцієнт розсіяної освітленості.

- Модель Бруфера-Фрінеля (англ. Fresnel Reflectance Model): Ця модель враховує явище відбиття світла на межі двох середовищ з різними показниками заломлення. Вона використовує формулу Фреснеля для обчислення коефіцієнта відбиття в залежності від кута падіння світла та показників заломлення матеріалу та середовища.
- Модель Мікросфери (англ. Microfacet Model): Ця модель описує поверхню матеріалу як сукупність мікроскопічних фасеток (мікросфер), кожна з яких має свою нормаль. Вона використовує статистичні розподіли нормалей фасеток для розрахунку відбитого світла та блисків на поверхні.

$k_{\text{spec}} = \frac{DFG}{4(V \cdot N)(N \cdot L)}$, де D — коефіцієнт розподілу Бекмана, як зазначено вище, а F — терм Френеля, а G — геометричний термін загасання, що описує самозатінення за рахунок мікрограней, і має вигляд:

$G = \min\left(1, \frac{2(H \cdot N)(V \cdot N)}{V \cdot H}, \frac{2(H \cdot N)(L \cdot N)}{V \cdot H}\right)$ У цих формулах V - вектор камери або ока, H - півкутний вектор, L - вектор до джерела світла і N - нормальний вектор, а α - кут між H і N.

- Модель Бі-Торранса (англ. Bidirectional Reflectance Distribution Function, BRDF): Ця модель описує поведінку світла при відбиванні від поверхні матеріалу в будь-якому напрямку. Вона використовує BRDF-функцію, яка враховує кут падіння світла, кут спостереження та характеристики матеріалу для обчислення інтенсивності відбитого світла. Вона вираховується за формулою: $f_{\mathrm{r}}(\omega_{\mathrm{i}}, \omega_{\mathrm{r}}) = \frac{\mathrm{d}L_{\mathrm{r}}(\omega_{\mathrm{r}})}{\mathrm{d}E_{\mathrm{i}}(\omega_{\mathrm{i}})} = \frac{1}{L_{\mathrm{i}}(\omega_{\mathrm{i}})\cos\theta_{\mathrm{i}}}\frac{\mathrm{d}L_{\mathrm{r}}(\omega_{\mathrm{r}})}{\mathrm{d}\omega_{\mathrm{i}}}$, де L - сяйво, або потужність на одиницю твердого кута в напрямку променя на одиницю проектованої площі-перпендикулярно променю, E- випромінювання, або потужність на одиницю площі поверхні, і θ_{i} - кут

між ω_i поверхнева нормаль $\mathbf{n}$. Індекс i вказує на падаюче світло, тоді як індекс r вказує на відбите світло.

Ці математичні методи, разом з іншими техніками, дозволяють створювати шейдери, що відтворюють різні фізичні явища, такі як дифузне та відбите світло, прозорість, відблиски, преломлення, розсіяне світло та інші, забезпечуючи більш реалістичний вигляд об'єктів у візуалізації ігрового світу.

Під час створення 3D-моделей для проекту використовувався математичний метод тріангуляції. Тріангуляцією називається процес розбиття полігональної області зі складною конфігурацією на трикутники. Будь-яку площину можна розбити на трикутники, а трикутник в свою чергу є найпростішим полігоном, вершини якого однозначно задають грань. Алгоритми тріангуляції використовують у багатьох процедурах машинної графіки. Наприклад, формування поверхонь, фарбування, видалення невидимих частин. В ігровій індустрії тріангуляція 3D-моделі є важливим етапом перед експортом у двигун, так як тріангуляція моделі забезпечує коректне відображення геометрії, текстур та бликів безпосередньо у грі.

2.3. Опис використаних технологій та мов програмування

Substance Painter - це потужний інтуїтивний інструмент для текстурування 3D-моделей, який дозволяє художникам створювати реалістичні текстури зі складними матеріалами та деталізацією. Він має широкий набір інструментів для розфарбування, створення масок, генерації бамп-мап, додавання глянцю, металічності та інших ефектів. Substance Painter підтримує PBR (Physically Based Rendering) та може інтегрувати в інші програмами для подальшої роботи з текстурами.

Marmoset Toolbag - це програмне забезпечення для попереднього перегляду та рендерингу 3D-моделей, що дозволяє швидко та ефективно створювати візуалізації. Воно має інтуїтивний інтерфейс, широкий спектр матеріалів та освітлення, підтримку PBR, а також інструменти для налаштування камери, розміщення світла, створення анімаційних петель та багато іншого. Marmoset Toolbag є потужним інструментом для швидкого та якісного візуалізації 3D-моделей.

3DS MAX - це професійний пакет програмного забезпечення для

3D-моделювання, анімації та візуалізації. Він надає широкий набір інструментів для створення складних 3D-моделей, реалістичних анімацій, візуалізації та спеціальних ефектів. 3DS MAX підтримує різноманітні формати файлів, має потужні інструменти для моделювання та анімації, включаючи каркасне моделювання, скульптуру, динаміку тіл, рендеринг та багато іншого. Це популярне програмне забезпечення серед візуалізаторів, архітекторів, ігрових розробників та інших професіоналів 3D-індустрії.

ZBrush - це програмне забезпечення для цифрової скульптури та моделювання 3D-об'єктів. Його використовують для створення високодеталізованих 3D-моделей з використанням різних інструментів, таких як кисті, маски, вирівнювання, деталізація та інші. ZBrush має потужний інтерфейс та широкі можливості для створення реалістичних 3D-моделей, включаючи складні текстури, бамп-мапи, дисплейсмент-мапи та інші ефекти. ZBrush є популярним у застосуванні в індустрії відеоігор, фільмів та анімації.

Quixel - це компанія та серія інструментів для створення фотореалістичних матеріалів та текстур. Quixel Suite має інструменти для створення матеріалів з високою деталізацією, включаючи дисплейсмент-мапи, бамп-мапи, глянцевість та інші характеристики. Quixel Mixer дозволяє створювати і змішувати матеріали з використанням шарів та масок. Quixel Bridge надає доступ до бібліотеки фотореалістичних матеріалів та моделей. Quixel є потужним інструментом для швидкого та реалістичного створення матеріалів та текстур.

C++ - це потужна мова програмування загального призначення, яка використовується для розробки різноманітних програм і систем. Вона володіє високою продуктивністю, низьким рівнем доступу до апаратного забезпечення та багатим набором функцій. C++ дозволяє розробникам створювати швидкі, ефективні та масштабовані програми, включаючи графічні ігри, розширення двигунів, вбудовані системи, драйвери та інше. Вона широко використовується в індустрії розробки програмного забезпечення та особливо в галузі комп'ютерної графіки.

Unreal Engine 5 (UE5) є потужним інтегрованим середовищем розробки, яке використовується для створення вражаючих відеоігор, симуляцій, віртуальної реальності та інших інтерактивних візуальних проектів. UE5 пропонує ряд

інноваційних функцій, які покращують реалістичність та продуктивність розробки.

Одна з найбільш вражаючих нововведень в UE5 - це система Lumen. Lumen є глобальною системою освітлення в реальному часі, яка використовує глобальну інформацію про освітлення для створення реалістичного освітлення та тіней в ігровому середовищі. Вона автоматично виявляє джерела світла, розсіює освітлення від поверхонь та створює вражаючі ефекти освітлення, такі як реалістичні тіні, відбиття та м'яке освітлення.

Ще одним інноваційним компонентом Unreal Engine 5 є система Nanite. Nanite є технологією відображення масштабованих деталей, яка дозволяє розробникам працювати з неймовірно деталізованими моделями без необхідності упрощення геометрії або використання звичайних технік LOD (рівні деталізації). Вона використовує високоефективну компресію та розподілену розрахункову систему для обробки величезних обсягів деталей і дозволяє розробникам створювати реалістичні та деталізовані світи.

2.4. Опис структури системи та алгоритмів її функціонування

Структура системи базується на архітектурі Unreal Engine 5 та включає різні компоненти, які співпрацюють між собою для створення ігрової локації. Основними компонентами є:

- Графічний двигун Unreal Engine 5: Він забезпечує загальну основу для розробки ігрової локації, включаючи рендеринг, фізику, освітлення та інші важливі функції.

- Локаційні елементи: Вони складаються з об'єктів, текстур, матеріалів та інших ресурсів, які використовуються для створення віртуальної ігрової локації.

- Шейдери: Вони використовуються для обробки графічних ефектів, освітлення та матеріалів у візуалізації локації. Приклад роботи шейдерів наведено на рис 2.4.1.

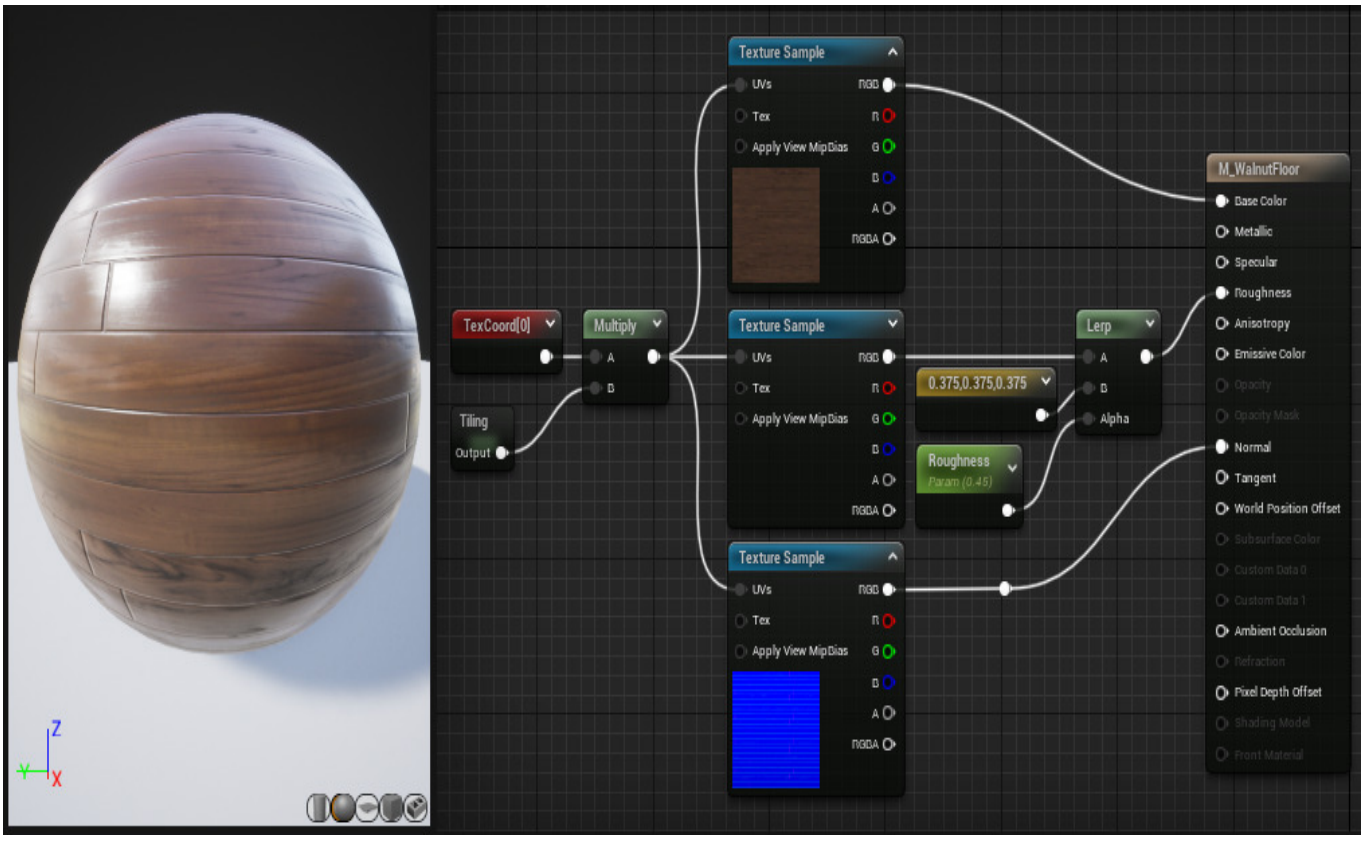

Рис. 2.1.1 - Візуалізація роботи шейдера з картами текстур у середовищі

Для досягнення бажаного візуального ефекту та функціональності ігрової локації використовуються різні алгоритми. Нижче наведені основні алгоритми, що використовуються у проекті:

- Алгоритм рендерингу: Unreal Engine 5 використовує потужні алгоритми рендерингу, такі як алгоритми трасування променів та методи прихованої поверхні, для створення реалістичного зображення ігрової локації.

- Алгоритми освітлення: Використовуються різні алгоритми освітлення, такі як амбієнтне освітлення, тінювання, динамічне освітлення тощо, для створення віртуального освітлення у локації.

- Алгоритми матеріалів та шейдерів: Для створення реалістичних матеріалів та ефектів використовуються різні алгоритми шейдерів, такі як алгоритми текстурного зображення, бамп-мапування, паралаксу тощо. Приклад алгоритму шейдерів наведено на рис 2.4.2.

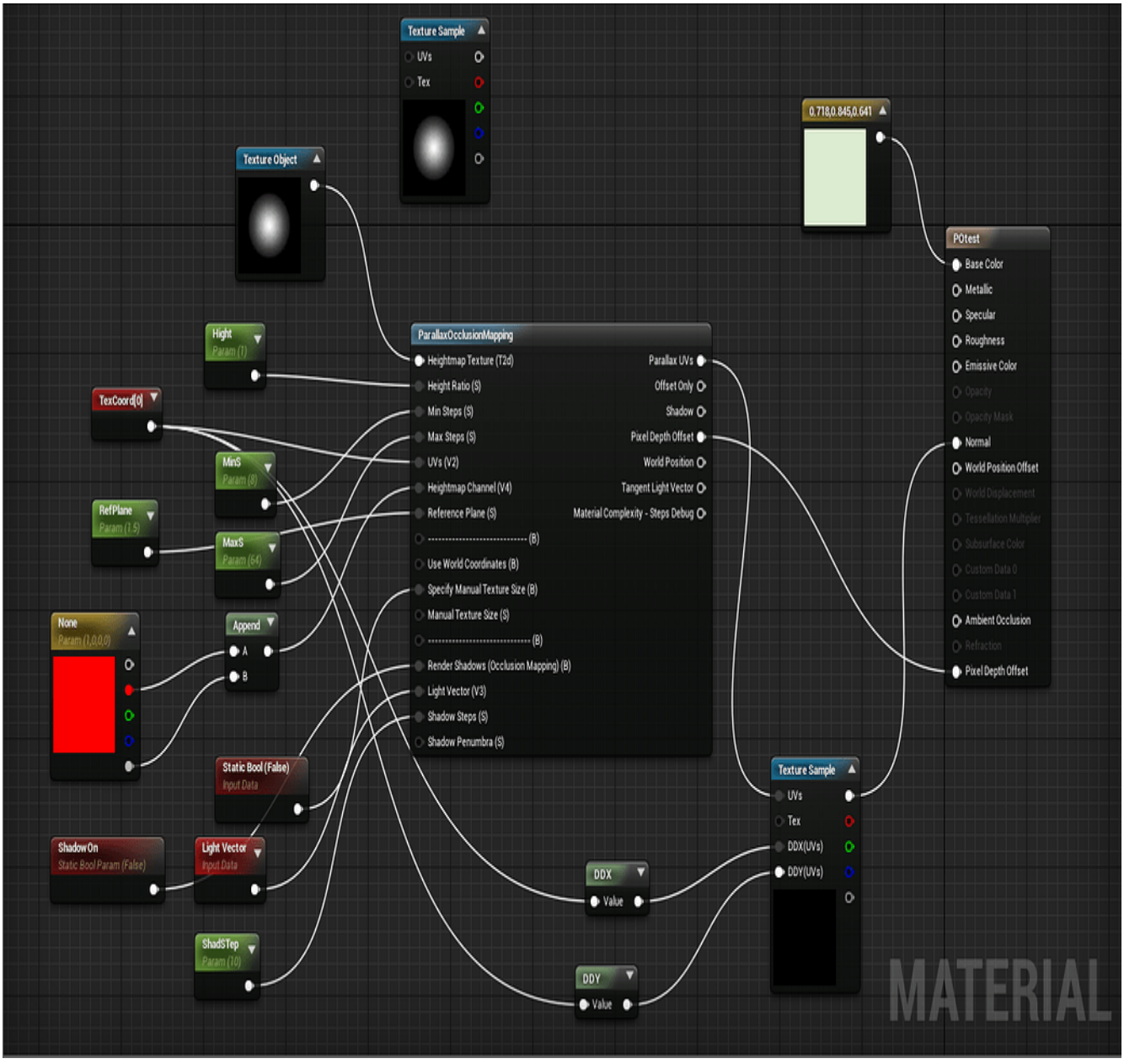

Рис. 2.1.2 - Алгоритм роботи шейдеру паралакса

Компоненти системи взаємодіють між собою для створення ігрової локації. Наприклад, шейдери використовуються для обробки матеріалів та освітлення, щоб забезпечити бажану візуалізацію. Локаційні елементи використовують ресурси графічного двигуна для відтворення об'єктів та текстур. Ця взаємодія компонентів сприяє створенню реалістичної ігрової локації.

При проектуванні локації та створенні ассетів для неї використовується велика кількість різноманітної інформації, та виконуються розрахунки необхідних ресурсів для її створення. Для демонстрації системи створення 3D моделі була розроблена IDEF0 діаграма .Див. рис. 2.1.3.

Рис 2.1.3 – Модель IDEF0 розробки 3d моделі

Варіант діаграми IDEF1 із детальним розкриттям поетапності розробки 3D моделі для даного проекту представлений на рис. 2.1.4.

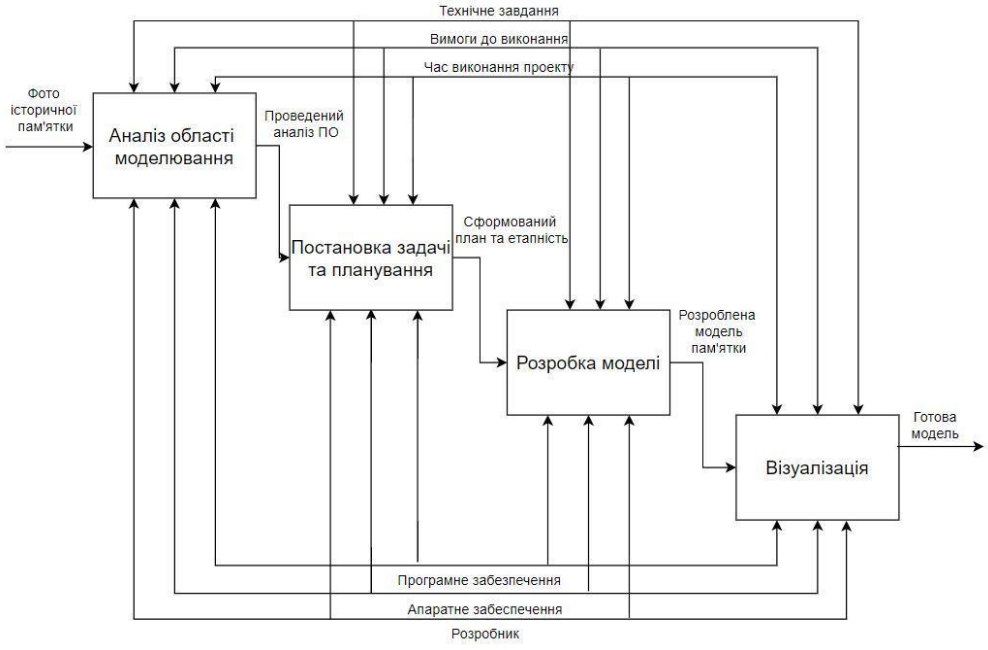

Рис 2.1.4 – Модель IDEF1 розробки 3D моделі

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Даний програмний додаток приймає ввід користувача для управління матеріалами в середині середовища розробки. В якості вихідних даних програма повертає графічне відображення змін. Демонстративна локація лише відображає вже налаштовані розробником параметри матеріалів.

2.6. Опис розробленої системи

2.6.1.Вимоги до функціональних характеристик

Для розробки була використана електронно-обчислювальна машина(ноутбук) з нижченаведеними характеристиками:

- OS: Windows 10
- CPU: Intel Core i5-7300HQ (2.50 ГГц)
- GPU: NVIDIA GeForce GTX 1050Ti
- Memory: SSD 1 TB
- RAM: 16 GB.
- Screen: 1920x1080

2.6.2.Використані програмні засоби

Autodesk 3DS MAX, часто називається просто 3DS MAX, - це програмне забезпечення для комп'ютерної 3D-графіки, яке використовується для моделювання, рендерингу, анімації та створення візуальних ефектів.

Користувачі 3DS MAX створюють віртуальні сцени, в яких розташовують та редагують об'єкти, освітлення, камери та матеріали. Програма пропонує потужні інструменти моделювання, які дозволяють створювати складні 3D-моделі об'єктів, будівель, персонажів тощо. За допомогою 3DS MAX також можна створювати анімацію, задавати рух об'єктів та персонажів, створювати реалістичні візуальні ефекти, такі як вогонь, вода, дим та вибухи.

3DS MAX використовує графічну структуру з вузлами, де об'єкти, матеріали, освітлення та ефекти з'єднані між собою. Кожен вузол має свої параметри та налаштування, що дозволяє користувачам контролювати вигляд та поведінку об'єктів. Інтерфейс 3DS MAX надає зручний спосіб відображення та редагування цих вузлових мереж. Приклад інтерфейсу Autodesk 3DS MAX наведено на рис. 2.2.1.

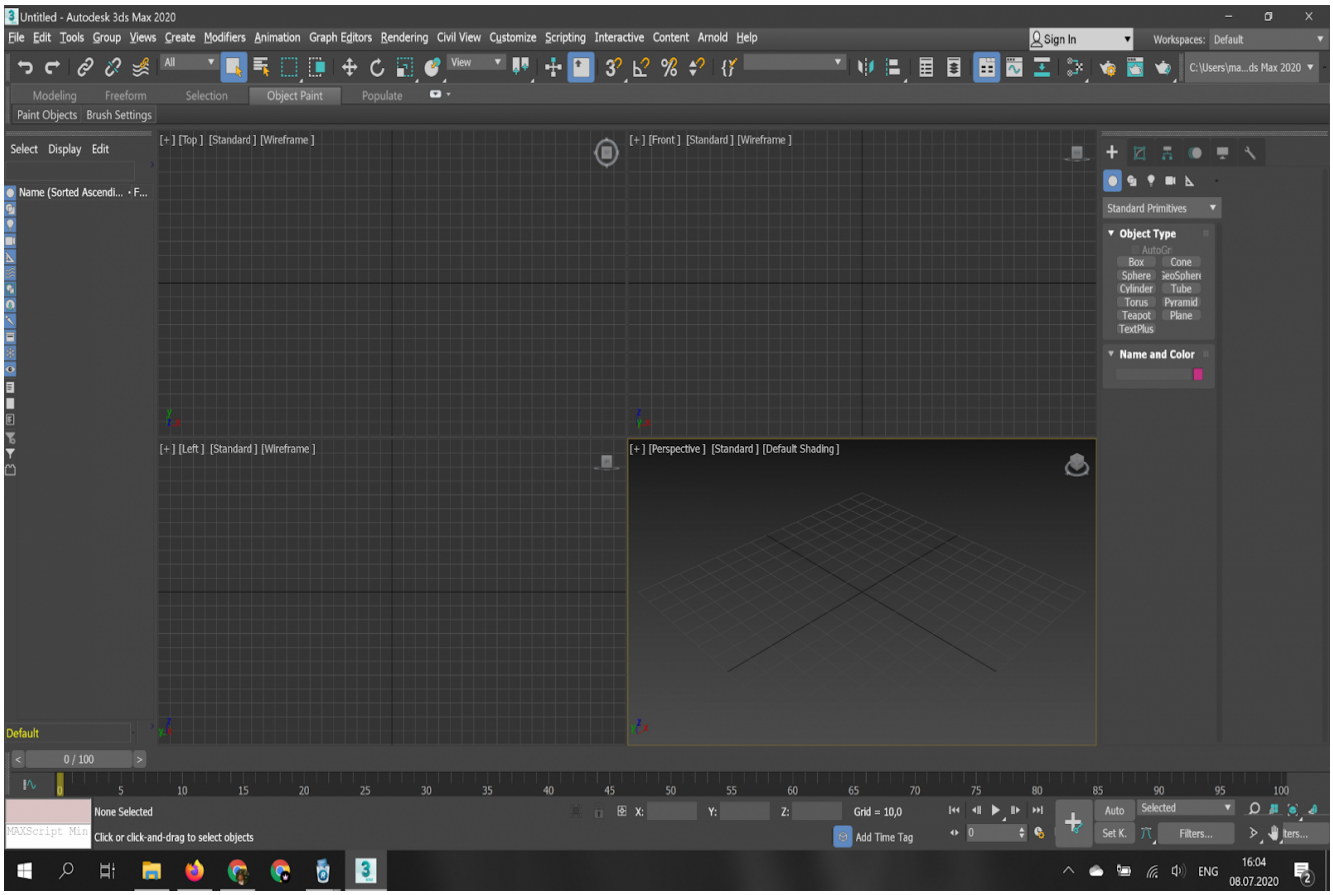

Рис. 2.2.1. - Інтерфейс програми 3DS MAX.

Pixologic ZBrush є програмним інструментом для цифрової скульптури, що об'єднує можливості 3D/2,5D моделювання, текстурування та фарбування. Використовуючи запатентовану технологію "піксол", це програмне забезпечення зберігає детальну інформацію про освітлення, колір, матеріал, орієнтацію та глибину кожної точки, яка складає об'єкти на екрані. Одне з головних відмінностей ZBrush від інших пакетів моделювання полягає в тому, що воно дозволяє схожий на традиційне ліплення процес моделювання. З високою роздільною здатністю, що може перевищувати 40 мільйонів полігонів, ZBrush використовується компаніями, такими як ILM, Weta Digital, Epic Games і Electronic Arts, для створення моделей, які використовуються в фільмах, іграх та анімації. Однією з функцій ZBrush є динамічні рівні роздільної здатності, що дозволяють скульпторам вносити глобальні або локальні зміни до своїх моделей. Після цього деталі сітки можуть бути експортовані у вигляді карт нормалей для використання на низькополігональних версіях тих самих

моделей.

Приклад інтерфейсу ZBrush наведено на рис. 2.2.2.

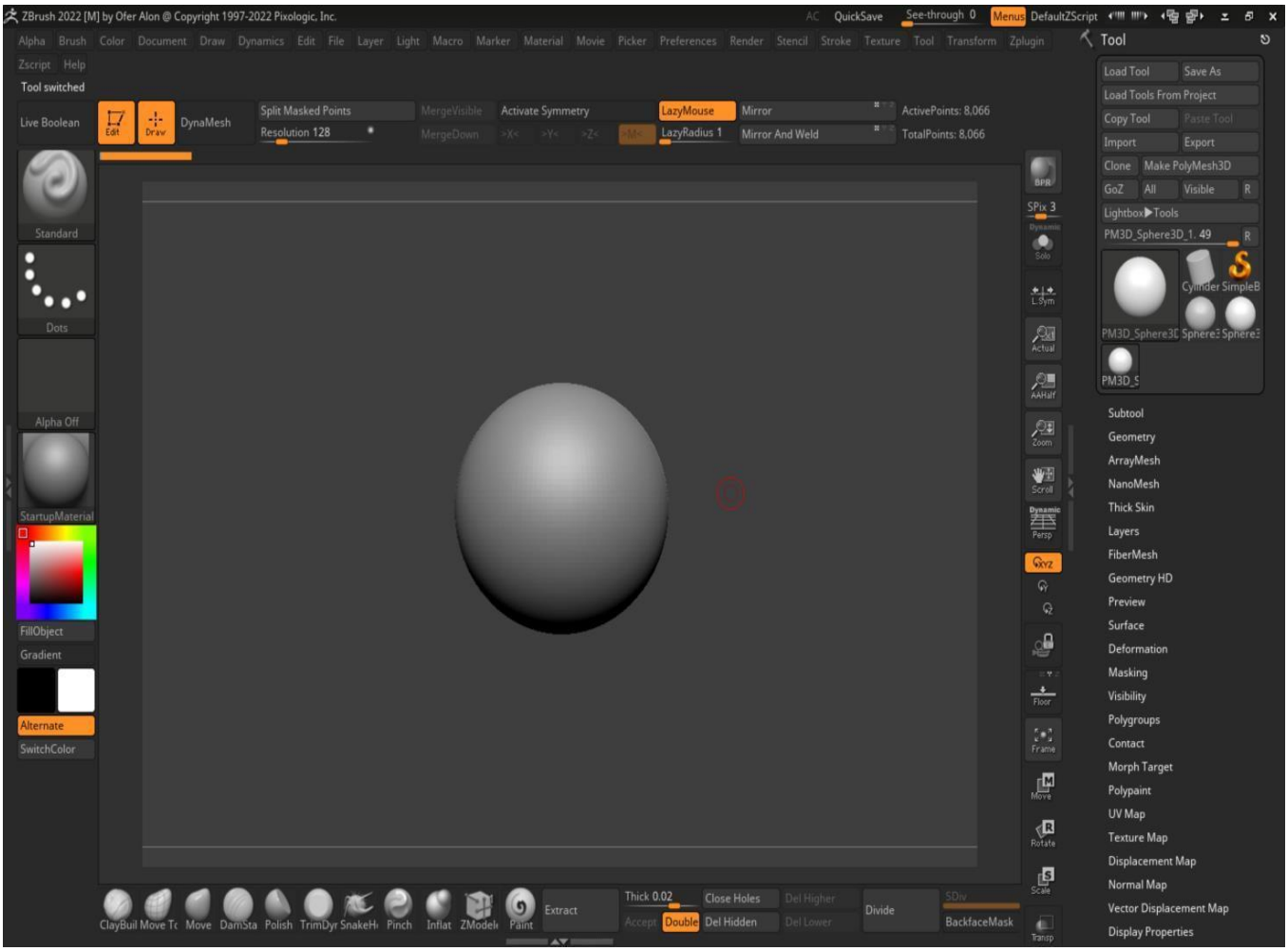

Рис. 2.2.2. - Інтерфейс програми ZBrush.

Substance Painter є надзвичайно потужним інструментом для створення 3D-малюнків. Цей інструмент можна порівняти з цифровим малюванням Adobe Photoshop, але для 3D-моделей. Основне призначення Substance Painter - створення текстур для моделей. Його розширені інструменти маскування та процедурного текстурування дозволяють створювати текстури, які важче досягти в програмах з обмеженими можливостями, таких як Photoshop.

Цей програмний засіб має кілька корисних функцій, таких як підтримка випікання текстур у роздільності 8k, робочі процеси PBR матеріалів у реальному часі та можливість збереження попередніх налаштувань матеріалів. Substance Painter також дозволяє малювати безпосередньо на 3D-моделі або на 2D-картах у вікні

перегляду 3D.

Програма Substance Painter відображає та експортує всі текстури у форматі PBR, що дозволяє імпортувати їх безпосередньо у ігровий двигун з однаковим результатом. Інтерфейс Substance Painter є дуже зручним та інтуїтивно зрозумілим.наведено на рис. 2.2.3.

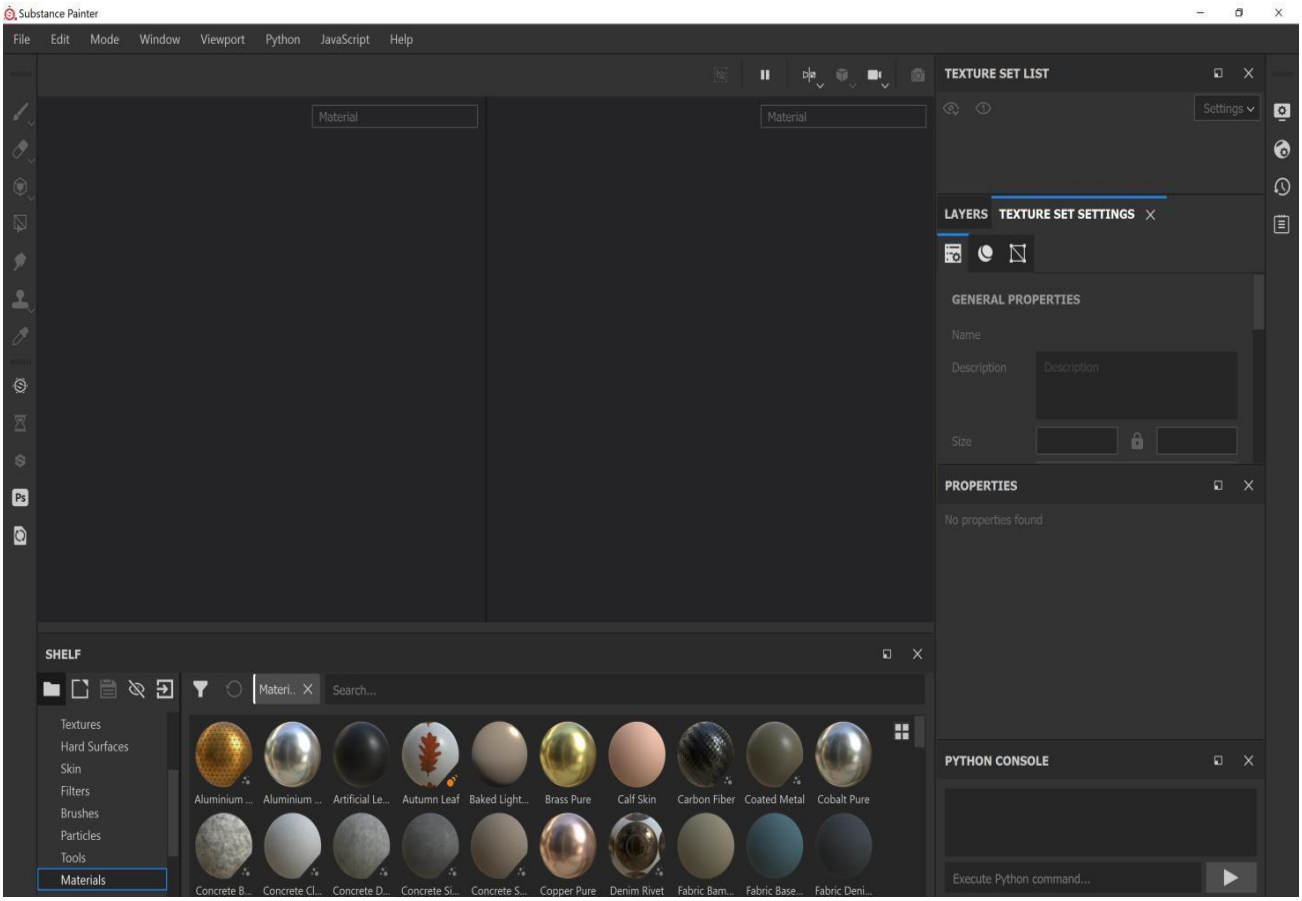

Рис. 2.2.3. - Інтерфейс програми Substance Painter.

Headus UVLayout — це програма для створення та редагування координат UV-текстур для 3D-полісіток і поверхонь. Унікальний підхід UVLayout, який використовується професіоналами в індустрії ігор та візуальних ефектів, любителями всіх типів і студентами, дає художникам текстури інструменти, необхідні для створення високоякісних UV-текстур із низьким викривленням за значно менший час, ніж традиційними методами.

Приклад інтерфейсу Headus UVLayout наведено на рис. 2.2.4.

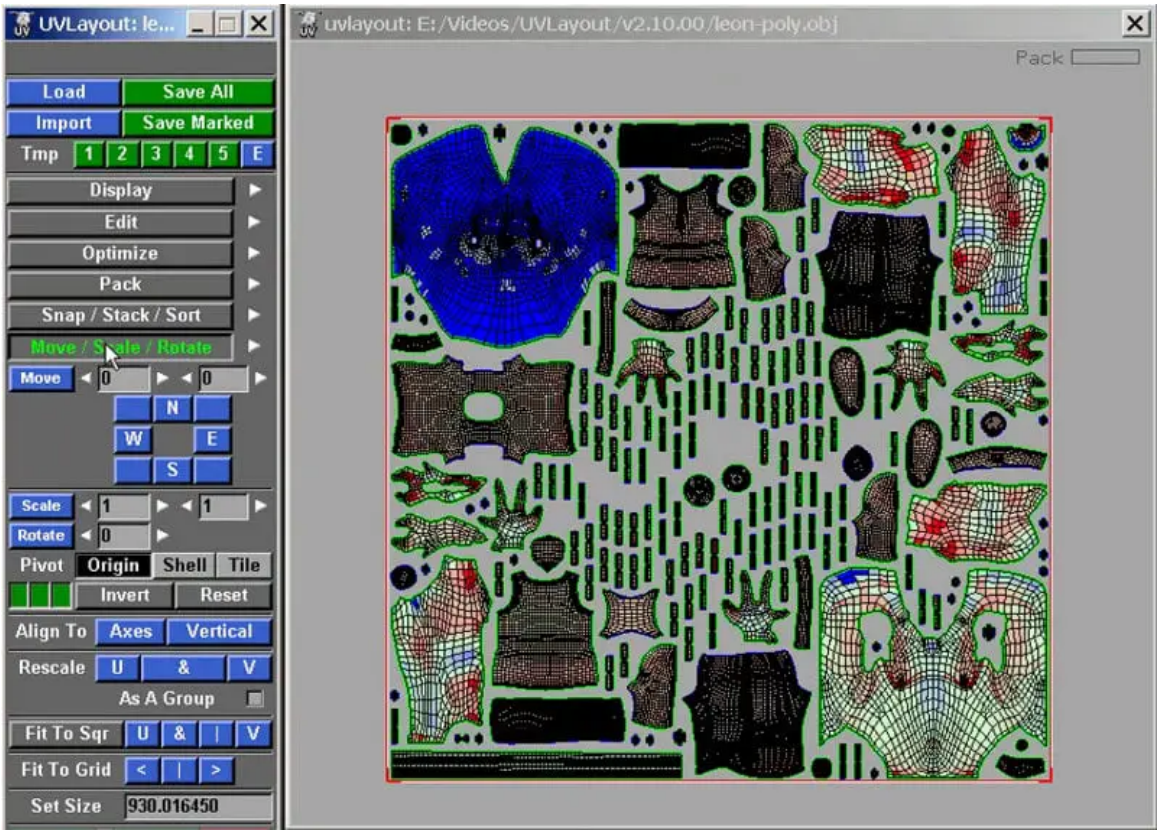

Рис. 2.2.4. - Інтерфейс програми Headus UVLayout.

Marmoset Tollbag 4 - це програмний інструмент для художників 3D, що вирішує різноманітні художні завдання, включаючи рендеринг, текстурні роботи, налаштування сцени та випікання карт (наприклад, normal map, ID map і т.д.).

Інструмент Toolbag забезпечує фізичну візуалізацію в реальному часі та освітлення, базоване на зображеннях, що дозволяє досягти високої якості зображення. Для швидкого випікання Toolbag використовує потужність вашого графічного процесора (відеокарти). При внесенні змін до локальних ділянок вашої сітки, він автоматично перевипікає лише змінену область, що призводить до миттєвого оновлення попереднього результату. Завдяки потужному механізму візуалізації GPU Toolbag може обробляти сітки навіть в дуже високій роздільній здатності і містити мільйони полігонів при використанні сучасних відеокарт. Важливо враховувати, що продуктивність Toolbag залежить від конкретного графічного процесора, який використовується. Інтерфейс Marmoset Toolbag.наведено на рис. 2.2.5.

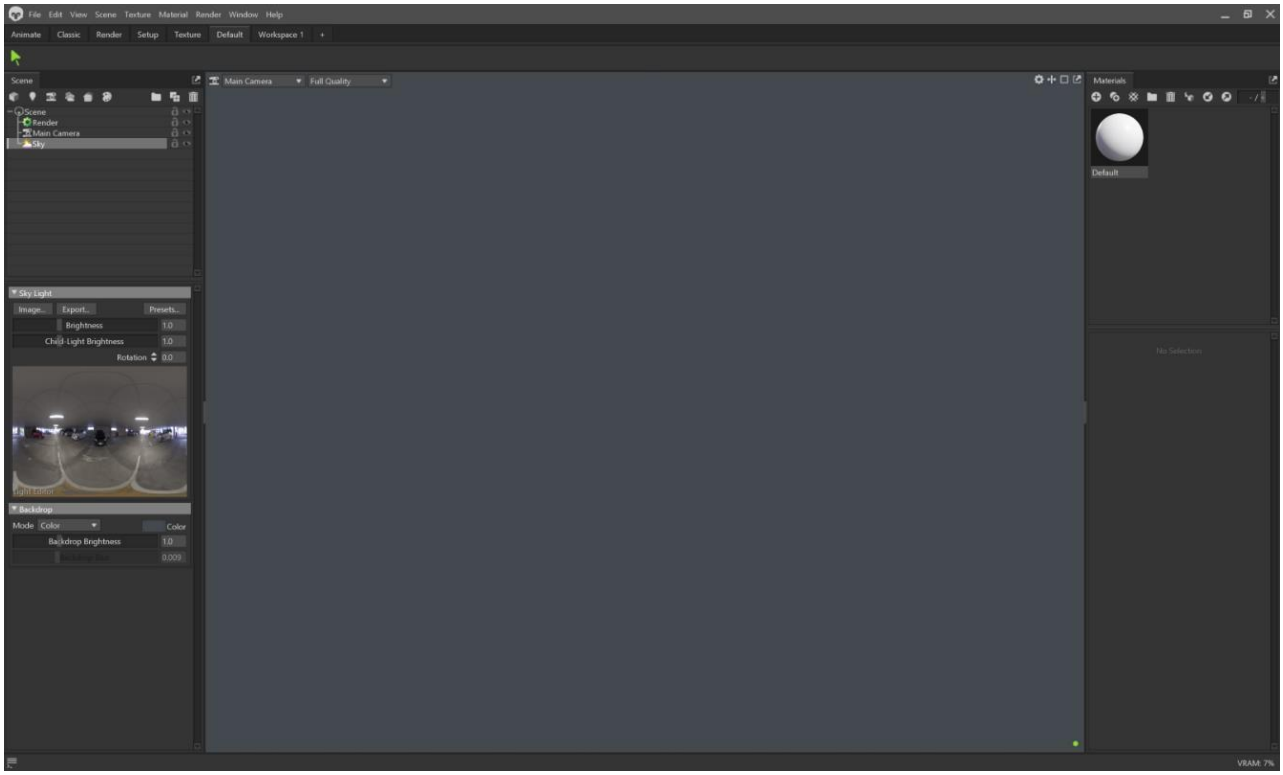

Рис. 2.2.5. - Інтерфейс програми Marmoset Toolbag.

Unreal Engine 5 є високопродуктивним і передовим інструментарієм для створення 3D-сцен та візуалізації, який визначає нові стандарти в галузі реалістичного моделювання та іммерсивних досвідів. Цей двигун розроблений з урахуванням найсучасніших графічних технологій, що дозволяє створювати вражаючі, фотореалістичні віртуальні світи, здатні перенести глядача відразу в середину захоплюючих ігрових пригод або дозволити детально дослідити архітектурні проекти до найдрібніших деталей.

Unreal Engine 5 пропонує розширений набір інструментів для рендерингу, освітлення, фізики та анімації, що дозволяє розробникам створювати неперевершені візуальні ефекти та динамічні інтерактивні сцени. Завдяки використанню передових технологій, таких як рейтрейсинг, динамічне освітлення Global Illumination, фотореалістичні матеріали та ефекти частинок, Unreal Engine 5 дозволяє досягти неймовірного рівня деталізації та реалізму візуального відтворення.

Окрім того, Unreal Engine 5 надає зручний інтерфейс розробки та потужні редактори для моделювання об'єктів, створення матеріалів, анімації та налаштування фізики. Цей інтуїтивно зрозумілий інтерфейс дозволяє розробникам зосередитися на

творчому процесі і швидко досягати бажаних результатів. Інтерфейс Unreal Engine 5 наведено на рис. 2.2.6.

Рис. 2.2.6. - Інтерфейс програми Unreal Engine 5.

Unreal Engine 5 також підтримує широкий спектр платформ, включаючи персональні комп'ютери, ігрові консолі та мобільні пристрої, що дозволяє розробникам створювати ігри для різних аудиторій. Двигун також має можливість експортування готових проектів у форматі, що підтримується різними ігровими двигунами, забезпечуючи максимальну гнучкість та сумісність при інтеграції з іншими інструментами та технологіями розробки.

2.6.3.Виклик та завантаження програми

Для запуску проекту необхідно завантажити архів з додатком із мережі Інтернет та розархівувати через будь-який архіватор. Після розпаковки архіву необхідно просто

запустити виконуваний файл у папці програми - додаткової інсталяції не потрібно. Розпакований вигляд архіву наведено на рис. 2.6.3.1.

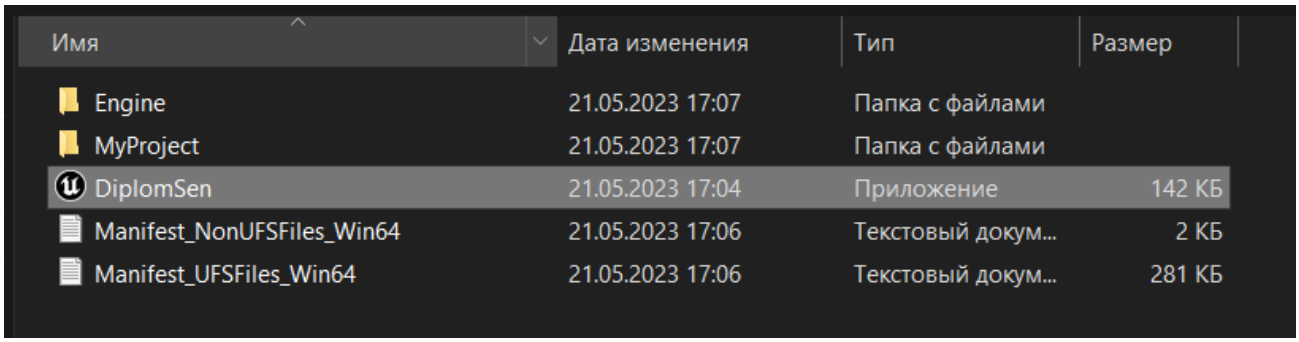

Рис. 2.3 - Приклад папки з додатком після розпакування архіву.

2.6.4.Опис інтерфейсу користувача

Рис. 2.4.1 - Рівень Бібліотека 1.

Рис. 2.4.2 - Рівень Бібліотека 2.

Рис. 2.4.3 - Рівень Бібліотека 3.

Рис. 2.4.4 - Рівень Бібліотека 4.

Рис. 2.4.5 - Рівень Бібліотека 5.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2500;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 157 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ – 17 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу

творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.
Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q – передбачуване число операторів (2500); C – коефіцієнт складності програми (1,4);

p – коефіцієнт кореляції програми в ході її розробки (0,06).

$$Q = 2500 \cdot 1{,}4 \cdot (1 + 0{,}06) = 3710;$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75\ldots85) \cdot K}, \text{ людино-годин} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,1);

$$t_u = 3710 \cdot 1{,}2/(85 \cdot 1{,}4) = 37{,}4, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20\ldots25) \cdot K}, \qquad (3.4)$$

$$t_a = 3710/(20 \cdot 1{,}4) = 132{,}5, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot K}, \quad (3.5)$$

$$t_n = 3710/(25 \cdot 1{,}4\ ) = 106, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

− за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4\ldots5)\cdot K}, \quad (3.6)$$

$$t_{\text{отл}} = 3710/(5 \cdot 1{,}4\ ) = 530, \text{ людино-годин,}$$

− за умови комплексного налагодження завдання:

$$t^{\text{к}}_{\text{отл}} = 1{,}2 \cdot t_{\text{отл}}; \quad (3.7)$$

$$t^{\text{к}}_{\text{отл}} = 1{,}2 \cdot 530 = 636, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де *tдр* – трудомісткість підготовки матеріалів і рукопису;

$$t_{\partial} = \frac{Q}{(15\ldots20)\cdot K}; \quad (3.9)$$

$$t_{\partial} = 3710/(20 \cdot 1{,}4\ ) = 132{,}5, \text{ людино-годин,}$$

де *tдо* – трудомісткість редагування, печатки й оформлення документації;

$$t_{\partial o} = 0{,}75 \cdot t_{\partial p}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 132{,}5 = 99{,}4, \text{ людино-годин.}$$

$$t_{д} = 132{,}5 + 99{,}4 = 231{,}9, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 37{,}4 + 132{,}5 + 106 + 530 + 231{,}9 = 1088, \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 1088

людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$КПО = ЗЗП + ЗМВ \text{ , грн,} \qquad (3.11)$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$ЗЗП = t \cdot СПР \text{ , грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;
$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 112 грн/год, то отримаємо:

$$З_{зп} = 1088 \cdot 112 = 121856, \text{ грн.}$$

Вартість машинного часу $З_{МВ,}$ необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$ЗМВ = tотл \cdot СМ \text{ , грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;
$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 530 \cdot 16 = 8480 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 121856 + 8480 = 130336 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ , мес.} \qquad (3.14)$$

де B_k – число виконавців;
$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).
Витрати на створення програмного продукту:
T = 1088/(1 · 176) = 6,2 міс.

Висновки. Додаток має вартість 130336 грн. Ймовірний очікуваний час розробки – 6,2 місяці при стандартному 40-годинному робочому тижні і 178- годинному робочому місяці. Цей термін пов’язаний з кількістю операторів і включає в себе час для дослідження та розробку алгоритму розв’язання задачі, розробку дизайну і створення документації. На розробку додатку буде витрачено 1088 людино-годин.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено програмне забезпечення для створення ігрових локацій з використанням середовища Unreal Engine. Головною метою цього програмного забезпечення було розширення можливостей роботи з графікою у руші Unreal Engine та демонстрація створення локацій з можливістю подальшої інтеграції їх у мультимедійні та ігрові проекти.

Програма була реалізована для операційної системи Windows, яка є основною платформою для цільової аудиторії продукту. Розробка додатку здійснювалась з використанням мови програмування C++ та ігрового двигуна Unreal Engine. Для створення візуальної частини додатку були використані такі сучасні програми для 3D-моделювання, як 3DS MAX, ZBrush, Headus UVLayout, Marmoset Toolbag та Substance Painter.

Особливу увагу в розробленому програмному забезпеченні було приділено шейдерам - спеціальним програмним кодам, які керують процесом обробки графічних об'єктів та їх візуалізацією. Шейдери забезпечують реалістичність графіки шляхом використання складних алгоритмів освітлення, тіней, текстур та спеціальних ефектів. Розроблені шейдери в програмному забезпеченні забезпечують широкий спектр візуальних покращень і дозволяють досягти високої якості графіки у створюваних локаціях.

Розроблене програмне забезпечення дозволяє розширити можливості роботи з графікою у рушії та використовувати створені локації в різноманітних мультимедійних та ігрових проектах. Результати економічного аналізу свідчать про прийнятну трудомісткість та вартість розробки програмного забезпечення.

В “Економічному розділі” дипломної роботи були проведені розрахунки для визначення трудомісткості розробленого додатку (1088 людино-годин), вартості роботи по його створенню (130336 грн) та приблизного часу, витраченого на розробку (6,2 місяці).

Розроблений додаток відповідає поставленим цілям та завданням дипломної роботи і може бути використаний у подальших дослідженнях та реалізації ігрових та мультимедійних проектів, що базуються на Unreal Engine.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is 3D Modeling & What's It Used For?. URL: https://conceptartempire.com/what-is-3d-modeling/ (дата звернення: 21.04.2022).
2. Vaughan W. Digital Modeling. New Riders Pub, 2017. 410 p.
3. What Is 3D Modeling?. URL: https://www.lifewire.com/what-is-3d- modeling-2164 (дата звернення: 22.04.2022).
4. Chopine A. 3D Art Essentials: The Fundamentals of 3D Modeling, Texturing, and Animation. Routledge, 2017. 288 p.
5. 3D virtual reality models help yield better surgical outcomes: Innovative technology improves visualization of patient anatomy, study finds. URL: https://www.sciencedaily.com/releases/2019/09/190918131457.htm (дата звернення 24.04.2022).
6. 3D Modelling Pipeline. URL:https://medium.com/@homicidalnacho/3d-modelling-pipeline-bd9be7dba136 (дата звернення: 25.04.2022)
7. What is Mixamo and How Can it be Used in Games. URL: https://cgobsession.com/what-is-mixamo-and-how-can-it-be-used-in-games/ (дата звернення 25.04.2022).
8. Albahari J. C++ 10 in a Nutshell: The Definitive Reference. O`Reilly Media, 2022. 1058 p.
9. C#/.NET History Lesson.
URL: http://jameskovacs.com/2007/09/07/cnet-history-lesson/ (дата звернення 26.04.2022).
10. Unity architecture. URL: https://docs.unity3d.com/Manual/unity-architecture.html (дата звернення 28.04.2022).
11. Lavieri E. Getting Started with UE5: A Beginner's Guide to 2D and 3D game development, 3rd Edition. Packt Publishing, 2018. 336 p.

12. Unreal Engine 3 brings very expensive dev tools at a very low price. URL: https://arstechnica.com/information-technology/2010/09/unity-3-brings-very-expensive-dev-tools-at-a-very-low-price/ (дата звернення 29.04.2022).
13. Robert C. Martin’s Principle Collection. URL: http://principles-wiki.net/collections:robert_c._martin_s_principle_collection (дата звернення 29.04.2022).
14. Tickoo S. Autodesk 3Ds Max: A Comprehensive Guide. CADCIM Technologies, 2021. 674 p.
15. ZBrush Features URL: Pixologic.com (дата звернення 1.05.2022).
16. What is Substance Painter. URL: https://conceptartempire.com/what-is-substance-painter (дата звернення 1.05.2022)
17. Headus UVLayout. URL: https://www.uvlayout.com/ (дата звернення 1.05.2022)
18. Ultimate Guide to Marvellous Designer with Camille Kleinman. URL: https://discover.therookies.co/2019/09/13/ultimate-guide-to-marvellous-designer-with-camille-kleinman/ (дата звернення 4.05.2022).
19. Jira. URL: Atlassian.com (дата звернення 4.05.2022).
20. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи” / Укладачі О.Г.Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг PlayerBase.cs:.

```
#include "Kismet/KismetMathLibrary.h"
#include "TriggerInterface.h"
#include "InteractableBase.h"
#include "DrawDebugHelpers.h"
#include "Kismet/GameplayStatics.h"
#include "Camera/PlayerCameraManager.h"
#include "PlayerControllerBase.h"
#include "MainUIBase.h"
#include "TriggerComponent.h"
#include "Camera/CameraComponent.h"
#include "Components/CapsuleComponent.h"

// Sets default values
APlayerBase::APlayerBase()
{
	// Set this character to call Tick() every frame.  You can turn this off to improve performance if you don't need it.
	PrimaryActorTick.bCanEverTick = true;

	Camera = CreateDefaultSubobject<UCameraComponent>(TEXT("Camera"));
	Camera->SetupAttachment(RootComponent);
}

// Called when the game starts or when spawned
void APlayerBase::BeginPlay()
{
	Super::BeginPlay();

	InitialCapsuleHeight = GetCapsuleComponent()->GetUnscaledCapsuleHalfHeight();
	GetCharacterMovement()->MaxWalkSpeed = WalkSpeed;
}

void APlayerBase::MovementForward(float AxisValue)
{
	FVector ForwardVector = UKismetMathLibrary::GetForwardVector(GetControlRotation());
	AddMovementInput(ForwardVector, AxisValue);

	if(!GetWorldTimerManager().IsTimerActive(FootstepTimer))
```

```
		GetWorldTimerManager().SetTimer(FootstepTimer, this,
&APlayerBase::HandleFootsteps, 1.f);
}

void APlayerBase::MovementRight(float AxisValue)
{
	FVector RightVector = UKismetMathLibrary::GetRightVector(GetControlRotation());
	AddMovementInput(RightVector, AxisValue);

	if (!GetWorldTimerManager().IsTimerActive(FootstepTimer))
		GetWorldTimerManager().SetTimer(FootstepTimer, this,
&APlayerBase::HandleFootsteps, 1.f);
}

void APlayerBase::HoverInteraction(float DeltaTime)
{
	FVector ForwardVector =
UKismetMathLibrary::GetForwardVector(GetControlRotation());
	FVector Distance = ForwardVector * InteractDistance;
	FCollisionQueryParams QueryParams =
FCollisionQueryParams(FName(TEXT("Interaction Actor")), false, this);
	FHitResult Hit;
	FVector CameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(),
0)->GetCameraLocation();
	APlayerControllerBase* PlayerController =
Cast<APlayerControllerBase>(GetController());

	//Line trace for interactable objects
	if (GetWorld()->LineTraceSingleByChannel(Hit, CameraLocation, CameraLocation +
Distance, ECC_Visibility, QueryParams))
	{
		if (ensure(Hit.GetActor()) && Cast<AInteractableBase>(Hit.GetActor()))
		{
			InteractHover = Cast<AInteractableBase>(Hit.GetActor());

PlayerController->MainUI->PlayInteractAnim(EUMGSequencePlayMode::Forward);
		}
		else
		{
			InteractHover = nullptr;

PlayerController->MainUI->PlayInteractAnim(EUMGSequencePlayMode::Reverse);
		}
	}
	else
	{
```

```
			InteractHover = nullptr;

PlayerController->MainUI->PlayInteractAnim(EUMGSequencePlayMode::Reverse);
		}
}

void APlayerBase::Interact()
{
		if (IsValid(InteractHover) && InteractHover->bCanInteract)
		{
			UTriggerComponent* TriggerComponent =
InteractHover->FindComponentByClass<UTriggerComponent>();
			if (TriggerComponent != nullptr)
			{
				TriggerComponent->TriggerActors();
			}

		}
}

// Called every frame
void APlayerBase::Tick(float DeltaTime)
{
		Super::Tick(DeltaTime);

		//Get actors in front of player that are interactable
		HoverInteraction(DeltaTime);

		HandleCrouching(DeltaTime);
}

// Called to bind functionality to input
void APlayerBase::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
		Super::SetupPlayerInputComponent(PlayerInputComponent);

		PlayerInputComponent->BindAxis("MoveForward", this,
&APlayerBase::MovementForward);
		PlayerInputComponent->BindAxis("MoveRight", this,
&APlayerBase::MovementRight);

		PlayerInputComponent->BindAxis("LookUp", this,
&APawn::AddControllerPitchInput);
		PlayerInputComponent->BindAxis("LookRight", this,
&APawn::AddControllerYawInput);
```

```
	PlayerInputComponent->BindAction("Interact", IE_Pressed, this,
&APlayerBase::Interact);

	PlayerInputComponent->BindAction("Crouch", IE_Pressed, this,
&APlayerBase::StartCrouch);
	PlayerInputComponent->BindAction("Crouch", IE_Released, this,
&APlayerBase::StopCrouch);

	PlayerInputComponent->BindAction("Sprint", IE_Pressed, this,
&APlayerBase::Sprint);
	PlayerInputComponent->BindAction("Sprint", IE_Released, this,
&APlayerBase::StopSprint);

}

void APlayerBase::HandleCrouching(float DeltaTime)
{
	UCharacterMovementComponent* MoveComponent = GetCharacterMovement();
	FHitResult Hit;
	float SmoothCrouch = GetCapsuleComponent()->GetUnscaledCapsuleHalfHeight();
	if (bIsCrouching && GetMovementComponent()->IsMovingOnGround())
	{
		SmoothCrouch = FMath::FInterpTo(SmoothCrouch,
MoveComponent->CrouchedHalfHeight, DeltaTime, CrouchSpeed);
		MoveComponent->MaxWalkSpeed =
MoveComponent->MaxWalkSpeedCrouched;

		GetCapsuleComponent()->SetCapsuleHalfHeight(SmoothCrouch);

	}
	else if (GetCapsuleComponent()->GetUnscaledCapsuleHalfHeight() !=
InitialCapsuleHeight && !GetWorld()->LineTraceSingleByChannel(Hit, GetActorLocation(),
GetActorLocation() + (GetActorLocation().UpVector * InitialCapsuleHeight), ECC_Visibility,
FCollisionQueryParams(NAME_None, false, this)))
	{
		SmoothCrouch = FMath::FInterpTo(SmoothCrouch, InitialCapsuleHeight,
DeltaTime, CrouchSpeed);

		GetCapsuleComponent()->SetCapsuleHalfHeight(SmoothCrouch);

		if(bIsSprinting)
			MoveComponent->MaxWalkSpeed = SprintSpeed;
		else
			MoveComponent->MaxWalkSpeed = WalkSpeed;
	}
}
```

```
void APlayerBase::HandleFootsteps()
{
	UE_LOG(LogTemp, Log, TEXT("Test"));
}
```

Лістинг CoreMinimal.cpp:.

```
#include "GameFramework/Character.h"
#include "GameFramework/CharacterMovementComponent.h"
#include "PlayerBase.generated.h"

class ITriggerInterface;
class AInteractableBase;
class UCameraComponent;

UCLASS()
class SPOOKYGAME_API APlayerBase : public ACharacter
{
	GENERATED_BODY()
public:
	// Sets default values for this character's properties
	APlayerBase();

	// Called every frame
	virtual void Tick(float DeltaTime) override;

	// Called to bind functionality to input
	virtual void SetupPlayerInputComponent(class UInputComponent*
PlayerInputComponent) override;

	//Distance that the player can interact with objects
	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly)
	float InteractDistance = 500.f;

	//Camera component
	UPROPERTY(VisibleAnywhere, BlueprintReadWrite)
	UCameraComponent* Camera;

	//Speed to crouch and uncrouch
	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Movement")
	float CrouchSpeed = 5.f;

	//Player walking speed (Copied from CharacterMovement Component)
	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Movement")
```

```
        float WalkSpeed = 250.f;

        //Player running speed
        UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Movement")
        float SprintSpeed = 350.f;

        //Store interact actor that the player is currently looking at
        UPROPERTY(VisibleAnywhere, BlueprintReadOnly)
        AInteractableBase* InteractHover = nullptr;

protected:
        // Called when the game starts or when spawned
        virtual void BeginPlay() override;
private:
        //Control player movement forward and backward
        void MovementForward(float AxisValue);

        //Control player movement right and left
        void MovementRight(float AxisValue);

        void HoverInteraction(float DeltaTime);

        void Interact();

        //Change player move state to sprinting
        void Sprint() { GetCharacterMovement()->MaxWalkSpeed = SprintSpeed; bIsSprinting
= true; }

        //Change player move state to running
        void StopSprint() { GetCharacterMovement()->MaxWalkSpeed = WalkSpeed;
bIsSprinting = false; }

        //Called to begin crouching
        void StartCrouch() { bIsCrouching = true; }

        //Called when player no longer wants to crouch
        void StopCrouch() { bIsCrouching = false; }

        //Called every tick to handle smooth crouching
        void HandleCrouching(float DeltaTime);

        //Store initial capsulehalfheight
        float InitialCapsuleHeight;

        //Handle triggering footstep sounds and screen shake
        UFUNCTION()
```

```
	void HandleFootsteps();

	//Handle footstep sound delay
	FTimerHandle FootstepTimer;

	//Store interact actor that player is interacting with
	AInteractableBase* CurrentInteractActor = nullptr;

	//When true, player wants to crouch
	bool bIsCrouching;

	//When true, player wants to sprint
	bool bIsSprinting;
};
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/PlayerController.h"
#include "PlayerControllerBase.generated.h"

class UMainUIBase;

/**
 *
 */
UCLASS()
class SPOOKYGAME_API APlayerControllerBase : public APlayerController
{
	GENERATED_BODY()

public:
	UPROPERTY(EditAnywhere, BlueprintReadWrite)
	UMainUIBase* MainUI;

};
// Fill out your copyright notice in the Description page of Project Settings.


#include "TriggerComponent.h"

#include "TriggerInterface.h"
#include "InteractableBase.h"

// Sets default values for this component's properties
```

```
UTriggerComponent::UTriggerComponent()
{
	// Set this component to be initialized when the game starts, and to be ticked every frame.  You can turn these features
	// off to improve performance if you don't need them.
	PrimaryComponentTick.bCanEverTick = true;

	// ...
}


// Called when the game starts
void UTriggerComponent::BeginPlay()
{
	Super::BeginPlay();

	// ...

}


// Called every frame
void UTriggerComponent::TickComponent(float DeltaTime, ELevelTick TickType, FActorComponentTickFunction* ThisTickFunction)
{
	Super::TickComponent(DeltaTime, TickType, ThisTickFunction);

	// ...
}

void UTriggerComponent::TriggerActors()
{
	for (AInteractableBase* Actor : ActorsToTrigger)
	{
		//Call trigger function for all actors in array if they implement the trigger interface
		if (IsValid(Actor))
		{
			ITriggerInterface* TriggerActor = Cast<ITriggerInterface>(Actor);
			TriggerActor->Execute_OnTrigger(Actor);
		}
	}
}

// Fill out your copyright notice in the Description page of Project Settings.

#pragma once
```

```
#include "CoreMinimal.h"
#include "Components/ActorComponent.h"
#include "TriggerComponent.generated.h"

class AInteractableBase;

UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent) )
class SPOOKYGAME_API UTriggerComponent : public UActorComponent
{
	GENERATED_BODY()

public:
	// Sets default values for this component's properties
	UTriggerComponent();

protected:
	// Called when the game starts
	virtual void BeginPlay() override;

public:
	// Called every frame
	virtual void TickComponent(float DeltaTime, ELevelTick TickType,
FActorComponentTickFunction* ThisTickFunction) override;

	//TArray of actors to trigger when TriggerActors() function is called
	UPROPERTY(EditAnywhere, BlueprintReadOnly)
	TArray<AInteractableBase*> ActorsToTrigger;

	//Function to be called to trigger actors in ActorsToTrigger array
	UFUNCTION(BlueprintCallable)
	void TriggerActors();


};
```

Лістинг GameUI.cpp:.

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

  <!-- Epic Games, Inc. UE4 Visualizers -->

  <!-- FString visualizer -->
  <Type Name="FString">
```

```
    <DisplayString Condition="Data.ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="Data.ArrayNum &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="Data.ArrayMax &lt; Data.ArrayNum">Invalid</DisplayString>
    <DisplayString Condition="Data.ArrayMax &gt;=
Data.ArrayNum">{Data.AllocatorInstance.Data,su}</DisplayString>
    <StringView Condition="Data.ArrayMax &gt;=
Data.ArrayNum">Data.AllocatorInstance.Data,su</StringView>
  </Type>

  <!-- TStringView default visualizer -->
  <Type Name="TStringView&lt;*&gt;">
    <DisplayString Condition="Size == 0">Empty</DisplayString>
    <DisplayString Condition="Size &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="Size &gt; 0">{DataPtr,[Size]}</DisplayString>
    <StringView Condition="Size &gt; 0">DataPtr,[Size]</StringView>
  </Type>

  <!-- TStringView<WIDECHAR> visualizer -->
  <Type Name="TStringView&lt;WIDECHAR&gt;">
    <DisplayString Condition="Size == 0">Empty</DisplayString>
    <DisplayString Condition="Size &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="Size &gt; 0">{DataPtr,[Size]su}</DisplayString>
    <StringView Condition="Size &gt; 0">DataPtr,[Size]su</StringView>
  </Type>

  <!-- TStringView<ANSICHAR> visualizer -->
  <Type Name="TStringView&lt;ANSICHAR&gt;">
    <DisplayString Condition="Size == 0">Empty</DisplayString>
    <DisplayString Condition="Size &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="Size &gt; 0">{DataPtr,[Size]s}</DisplayString>
    <StringView Condition="Size &gt; 0">DataPtr,[Size]s</StringView>
  </Type>

  <Type Name="TStringBuilderBase&lt;*&gt;">
    <DisplayString Condition="Base == CurPos">Empty</DisplayString>
    <DisplayString Condition="sizeof($T1) == 1">{Base,[CurPos-Base]s}</DisplayString>
    <DisplayString>{Base,[CurPos-Base]su}</DisplayString>
    <StringView Condition="sizeof($T1) == 1">Base,[CurPos-Base]s</StringView>
    <StringView>Base,[CurPos-Base]su</StringView>
  </Type>

  <Type Name="FGuid">
    <DisplayString>{{{A,Xb}-{(unsigned __int16)(B >> 16),Xb}-{(unsigned
__int16)B,Xb}-{(unsigned __int16)(C >> 16),Xb}-{(unsigned
__int16)C,Xb}{D,Xb}}}</DisplayString>
    <Expand>
```

```
    <Item Name="A">A,X</Item>
    <Item Name="B">B,X</Item>
    <Item Name="C">C,X</Item>
    <Item Name="D">D,X</Item>
   </Expand>
  </Type>

  <!-- FText visualizer -->
  <Type Name="FStringTableEntry">
   <DisplayString>{*DisplayString.Object}</DisplayString>
  </Type>
  <Type Name="FTextHistory_StringTableEntry::FStringTableReferenceData">
   <DisplayString Condition="StringTableEntry.Object != 0 &amp;amp;&amp;amp;
StringTableEntry.WeakReferenceCount.ReferenceController->SharedReferenceCount &gt;
0">{*StringTableEntry.Object}</DisplayString>
   <DisplayString Condition="StringTableEntry.Object == 0 ||
StringTableEntry.WeakReferenceCount.ReferenceController->SharedReferenceCount ==
0">&lt;MISSING STRING TABLE ENTRY&gt;</DisplayString>
  </Type>
  <Type Name="FTextHistory_StringTableEntry">
   <DisplayString>{*StringTableReferenceData.Object}</DisplayString>
  </Type>
  <Type Name="TLocalizedTextData&lt;*&gt;">
   <DisplayString>{*LocalizedString.Object}</DisplayString>
  </Type>
  <Type Name="TGeneratedTextData&lt;*&gt;">
   <DisplayString
Condition="LocalizedString.Object">{*LocalizedString.Object}</DisplayString>
   <DisplayString Condition="!LocalizedString.Object">{DisplayString}</DisplayString>
  </Type>
  <Type Name="TIndirectTextData&lt;*&gt;">
   <DisplayString>{History}</DisplayString>
  </Type>
  <Type Name="FText">
   <DisplayString>{*TextData.Object}</DisplayString>
  </Type>

  <!-- FName visualizer -->

  <Type Name="FName">
   <DisplayString Condition="Number">{ComparisonIndex}_{Number - 1}</DisplayString>
   <DisplayString>{ComparisonIndex}</DisplayString>
  </Type>

  <Type Name="FName" Priority="High">
   <DisplayString Condition="Number">{DisplayIndex}_{Number - 1}</DisplayString>
```

```
    <DisplayString>{DisplayIndex}</DisplayString>
  </Type>

  <Type Name="FMinimalName">
    <DisplayString Condition="Number">{Index}_{Number - 1}</DisplayString>
    <DisplayString>{Index}</DisplayString>
  </Type>

  <Type Name="FNameEntryId">
    <DisplayString Condition="Value &amp; FNameDebugVisualizer::UnusedMask">Illegal
name (block index out of range)</DisplayString>
    <DisplayString Condition="!GNameBlocksDebug[Value &gt;&gt;
FNameDebugVisualizer::OffsetBits]">Illegal name (null block)</DisplayString>

    <DisplayString>{(FNameEntry&amp;)GNameBlocksDebug[Value &gt;&gt;
FNameDebugVisualizer::OffsetBits][FNameDebugVisualizer::EntryStride * (Value &amp;
FNameDebugVisualizer::OffsetMask)],sb}</DisplayString>
  </Type>

  <Type Name="FNameEntry">
    <DisplayString Condition="Header.Len &gt; FNameDebugVisualizer::MaxLength">Illegal
name (length > NAME_SIZE)</DisplayString>
    <DisplayString
Condition="Header.bIsWide">{WideName,[Header.Len]su}</DisplayString>
    <DisplayString>{AnsiName,[Header.Len]s}</DisplayString>
  </Type>

  <!-- FStatNameAndInfo -->
  <Type Name="FStatNameAndInfo">
    <DisplayString>{(EStatOperation::Type)((NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatOperation::Shift)) &amp; EStatOperation::Mask),en}
{(EStatDataType::Type)((NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatDataType::Shift)) &amp; EStatDataType::Mask),en} {NameAndInfo}</DisplayString>
    <Expand>
      <Item Name="[Name]">NameAndInfo</Item>
      <Item Name="[StatOperation]">(EStatOperation::Type)((NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatOperation::Shift)) &amp; EStatOperation::Mask)</Item>
      <Item Name="[StatDataType]">(EStatDataType::Type)((NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatDataType::Shift)) &amp; EStatDataType::Mask)</Item>
      <Item Name="[IsCycle]">((NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatMetaFlags::Shift)) &amp; EStatMetaFlags::IsCycle) == EStatMetaFlags::IsCycle</Item>
      <Item Name="[IsMemory]">((NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatMetaFlags::Shift)) &amp; EStatMetaFlags::IsMemory) ==
EStatMetaFlags::IsMemory</Item>
      <Item Name="[IsPackedCCAndDuration]">((NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatMetaFlags::Shift)) &amp;
```

```
EStatMetaFlags::IsPackedCCAndDuration) ==
EStatMetaFlags::IsPackedCCAndDuration</Item>
     <Item Name="[ShouldClearEveryFrame]">((NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatMetaFlags::Shift)) &amp;
EStatMetaFlags::ShouldClearEveryFrame) ==
EStatMetaFlags::ShouldClearEveryFrame</Item>
   </Expand>
  </Type>

  <!-- FStatMessage without DebugStatData-->
  <Type Name="FStatMessage">
   <DisplayString>{NameAndInfo}</DisplayString>
  </Type>

  <!-- FStatMessage with DebugStatData -->
  <Type Name="FStatMessage" Priority="High">
   <!--ST_None     -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp; EStatDataType::Mask) ==
EStatDataType::ST_None">
     {{NoneType NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_int64 && !IsPackedCCAndDuration && !IsCycle -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp; EStatDataType::Mask) ==
EStatDataType::ST_int64 &amp;&amp; ((NameAndInfo.NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatMetaFlags::Shift)) &amp;
EStatMetaFlags::IsPackedCCAndDuration) != EStatMetaFlags::IsPackedCCAndDuration
&amp;&amp; ((NameAndInfo.NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatMetaFlags::Shift)) &amp; EStatMetaFlags::IsCycle) != EStatMetaFlags::IsCycle">
     {{Int64={DebugStatData.Cycles} NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_int64 && !IsPackedCCAndDuration && IsCycle -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp; EStatDataType::Mask) ==
EStatDataType::ST_int64 &amp;&amp; ((NameAndInfo.NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatMetaFlags::Shift)) &amp;
EStatMetaFlags::IsPackedCCAndDuration) != EStatMetaFlags::IsPackedCCAndDuration
&amp;&amp; ((NameAndInfo.NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatMetaFlags::Shift)) &amp; EStatMetaFlags::IsCycle) == EStatMetaFlags::IsCycle">
     {{Cycles={DebugStatData.Cycles} NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_int64 && IsPackedCCAndDuration && IsCycle -->
```

```
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp;amp; EStatDataType::Mask) ==
EStatDataType::ST_int64 &amp;amp;&amp;amp; ((NameAndInfo.NameAndInfo.Number >>
(EStatAllFields::StartShift + EStatMetaFlags::Shift)) &amp;amp;
EStatMetaFlags::IsPackedCCAndDuration) == EStatMetaFlags::IsPackedCCAndDuration
&amp;amp;&amp;amp; ((NameAndInfo.NameAndInfo.Number >> (EStatAllFields::StartShift +
EStatMetaFlags::Shift)) &amp;amp; EStatMetaFlags::IsCycle) == EStatMetaFlags::IsCycle">

{{Count={DebugStatData.CCAndDuration[0]},Cycles={DebugStatData.CCAndDuration[1]}
NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_double -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp;amp; EStatDataType::Mask) ==
EStatDataType::ST_double">
     {{Float={DebugStatData.Float} NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_FName -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp;amp; EStatDataType::Mask) ==
EStatDataType::ST_FName">
     {{Name={(FNameEntryId&amp;amp;)DebugStatData.Cycles}
NameAndInfo={NameAndInfo}}}
   </DisplayString>

   <!--ST_Ptr -->
   <DisplayString Condition="(EStatDataType::Type)((NameAndInfo.NameAndInfo.Number
>> (EStatAllFields::StartShift + EStatDataType::Shift)) &amp;amp; EStatDataType::Mask) ==
EStatDataType::ST_Ptr">
     {{Ptr={DebugStatData.Ptr} NameAndInfo={NameAndInfo}}}
   </DisplayString>
  </Type>

  <!-- FAllocationInfo -->
  <!--
                uint64 OldPtr;
                uint64 Ptr;
                int64 Size;
                FName EncodedCallstack;
                uint32 SequenceTag;
                EMemoryOperation Op; Alloc=1, Free=2, Realloc=3
                bool bHasBrokenCallstack;
        -->
  <Type Name="FAllocationInfo">
```

```
    <!-- Alloc -->
    <DisplayString Condition="Op == 1" >
      {{A SeqTag={SequenceTag} Ptr={Ptr} Size={Size} Callstack={EncodedCallstack}
bHasBrokenCallstack={bHasBrokenCallstack}}}
    </DisplayString>

    <!-- Free -->
    <DisplayString Condition="Op == 2" >
      {{F SeqTag={SequenceTag} Ptr={Ptr} bHasBrokenCallstack={bHasBrokenCallstack}}}
    </DisplayString>

    <!-- Realloc -->
    <DisplayString Condition="Op == 3" >
      {{R SeqTag={SequenceTag} OldPtr={OldPtr} Ptr={Ptr} NewSize={Size}
Callstack={EncodedCallstack} bHasBrokenCallstack={bHasBrokenCallstack}}}
    </DisplayString>
  </Type>

  <Type Name="FThreadSafeCounter">
    <DisplayString>{Counter}</DisplayString>
  </Type>

  <Type Name="FThreadSafeBool">
    <DisplayString Condition="Counter==0">False</DisplayString>
    <DisplayString Condition="Counter==1">True</DisplayString>
  </Type>

  <!-- FTimespan visualizer -->
  <Type Name="FTimespan">
    <DisplayString>Ticks = {Ticks}</DisplayString>
    <Expand>
      <Item Name="Total Milliseconds">Ticks / ETimespan::TicksPerMillisecond</Item>
      <Item Name="Total Seconds">Ticks / ETimespan::TicksPerSecond</Item>
      <Item Name="Total Minutes">Ticks / ETimespan::TicksPerMinute</Item>
      <Item Name="Total Hours">Ticks / ETimespan::TicksPerHour</Item>
      <Item Name="Total Days">Ticks / ETimespan::TicksPerDay</Item>
    </Expand>
  </Type>

  <Type Name="FAsyncPackageData">
    <DisplayString>ExportCount={ExportCount}, ExportBundleCount={ExportBundleCount},
ImportedPackages={ImportedAsyncPackages.ArrayNum}</DisplayString>
  </Type>

  <Type Name="FEventLoadNode2">
    <DisplayString Condition="bDone.Element">Done {Spec-&amp;gt;Func,na}</DisplayString>
```

```
    <DisplayString Condition="bFired.Element">Fired {Spec-&gt;Func,na}</DisplayString>
    <DisplayString>Queued (BarrierCount={BarrierCount.Element})
{Spec-&gt;Func,na}</DisplayString>
  </Type>

  <Type Name="FExportObject">
    <DisplayString Condition="Object != 0">{Object-&gt;NamePrivate}
({Object-&gt;ObjectFlags,x})</DisplayString>
    <DisplayString Condition="!bFiltered &amp;&amp; !bExportLoadFailed">Null
(Create)</DisplayString>
    <DisplayString Condition="bFiltered">Null (Filtered)</DisplayString>
    <DisplayString>Null (Failed)</DisplayString>
  </Type>

  <Type Name="FAsyncPackageDesc2">
    <DisplayString Condition="CustomPackageId.Id !=
CustomPackageId.InvalidId">{CustomPackageName} ({CustomPackageId.Id,X})
{DiskPackageName} ({DiskPackageId.Id,X})</DisplayString>
    <DisplayString>{DiskPackageName} ({DiskPackageId.Id,X})</DisplayString>
  </Type>

  <Type Name="FAsyncPackage2">
    <DisplayString>{Desc}</DisplayString>
  </Type>

  <Type Name="FPackageId">
    <DisplayString Condition="Id != InvalidId">{Id,X}</DisplayString>
    <DisplayString>Null</DisplayString>
  </Type>

  <Type Name="FPackageIndex">
    <DisplayString Condition="Index &lt; 0">ImportIndex={-Index-1}</DisplayString>
    <DisplayString Condition="Index &gt; 0">ExportIndex={Index-1}</DisplayString>
    <DisplayString>Null</DisplayString>
  </Type>

  <Type Name="FPackageObjectIndex">
    <DisplayString Condition="TypeAndId != Invalid">{(EType)(TypeAndId >> TypeShift)}
{TypeAndId &amp; IndexMask,X}</DisplayString>
    <DisplayString>Null</DisplayString>
  </Type>

  <Type Name="TPackageStoreEntryCArrayView&lt;*&gt;">
    <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
    <DisplayString>Num={ArrayNum}, Data[0]={($T1*)((uint8*)this +
OffsetToDataFromThis)}</DisplayString>
```

```
    <Expand>
      <ArrayItems Condition="ArrayNum &gt;= 0">
        <Size>ArrayNum</Size>
        <ValuePointer>($T1*)((char*)this + OffsetToDataFromThis)</ValuePointer>
      </ArrayItems>
    </Expand>
  </Type>

  <Type Name="FVector">
    <DisplayString>{{X={X} Y={Y} Z={Z}}}</DisplayString>
    <Expand HideRawView="true">
      <Item Name="X">X</Item>
      <Item Name="Y">Y</Item>
      <Item Name="Z">Z</Item>
      <Item Name="||V||&#xB2;">X*X + Y*Y + Z*Z</Item>
    </Expand>
  </Type>

  <Type Name="FQuat">
    <DisplayString>{{X={X} Y={Y} Z={Z} W={W}}}</DisplayString>
    <Expand HideRawView="true">
      <Item Name="X">X</Item>
      <Item Name="Y">Y</Item>
      <Item Name="Z">Z</Item>
      <Item Name="W">W</Item>
      <Item Name="||Q||&#xB2;">X*X + Y*Y + Z*Z + W*W</Item>
    </Expand>
  </Type>

  <Type Name="FRotator">
    <DisplayString>{{Pitch={Pitch} Yaw={Yaw} Roll={Roll}}}</DisplayString>
    <Expand HideRawView="true">
      <Item Name="Pitch (Y)">Pitch</Item>
      <Item Name="Yaw (Z)">Yaw</Item>
      <Item Name="Roll (X)">Roll</Item>
    </Expand>
  </Type>

  <Type Name="FTransform">
    <DisplayString>{{Translation={{X={Translation.m128_f32[0]}
Y={Translation.m128_f32[1]} Z={Translation.m128_f32[2]}}}
Rotation={{X={Rotation.m128_f32[0]} Y={Rotation.m128_f32[1]}
Z={Rotation.m128_f32[2]} W={Rotation.m128_f32[3]}}}}}</DisplayString>
    <Expand>
      <Synthetic Name="Translation">
        <DisplayString>{{X={Translation.m128_f32[0]} Y={Translation.m128_f32[1]}
```

```
Z={Translation.m128_f32[2]}}}</DisplayString>
      <Expand>
        <Item Name="X">Translation.m128_f32[0]</Item>
        <Item Name="Y">Translation.m128_f32[1]</Item>
        <Item Name="Z">Translation.m128_f32[2]</Item>
        <Item Name="||V||&#xB2;">Translation.m128_f32[0]*Translation.m128_f32[0] +
Translation.m128_f32[1]*Translation.m128_f32[1] +
Translation.m128_f32[2]*Translation.m128_f32[2]</Item>
      </Expand>
    </Synthetic>
    <Synthetic Name="Rotation">
      <DisplayString>{{X={Rotation.m128_f32[0]} Y={Rotation.m128_f32[1]}
Z={Rotation.m128_f32[2]} W={Rotation.m128_f32[3]}}}</DisplayString>
      <Expand>
        <Item Name="X">Rotation.m128_f32[0]</Item>
        <Item Name="Y">Rotation.m128_f32[1]</Item>
        <Item Name="Z">Rotation.m128_f32[2]</Item>
        <Item Name="W">Rotation.m128_f32[3]</Item>
        <Item Name="||Q||&#xB2;">Rotation.m128_f32[0]*Rotation.m128_f32[0] +
Rotation.m128_f32[1]*Rotation.m128_f32[1] + Rotation.m128_f32[2]*Rotation.m128_f32[2]
+ Rotation.m128_f32[3]*Rotation.m128_f32[3]</Item>
      </Expand>
    </Synthetic>
    <Synthetic Name="Scale3D">
      <DisplayString>{{X={Scale3D.m128_f32[0]} Y={Scale3D.m128_f32[1]}
Z={Scale3D.m128_f32[2]}}}</DisplayString>
      <Expand>
        <Item Name="X">Scale3D.m128_f32[0]</Item>
        <Item Name="Y">Scale3D.m128_f32[1]</Item>
        <Item Name="Z">Scale3D.m128_f32[2]</Item>
      </Expand>
    </Synthetic>
  </Expand>
 </Type>

 <Type Name="FPlane">
  <DisplayString>{{X={X} Y={Y} Z={Z} W={W}}}</DisplayString>
  <Expand HideRawView="true">
    <Item Name="X">X</Item>
    <Item Name="Y">Y</Item>
    <Item Name="Z">Z</Item>
    <Item Name="W">W</Item>
  </Expand>
 </Type>

 <!-- TEnumAsByte visualizer -->
```

```
  <Type Name="TEnumAsByte&lt;*&gt;">
    <DisplayString>{($T1)Value}</DisplayString>
  </Type>

  <!-- UObjectBase visualizer -->
  <Type Name="UObjectBase">
    <DisplayString>(Name={NamePrivate})</DisplayString>
  </Type>

  <!-- FFieldClass visualizer -->
  <Type Name="FFieldClass">
    <DisplayString>(Name={Name})</DisplayString>
  </Type>

  <!-- FFieldVariant visualizer -->
  <Type Name="FFieldVariant">
    <DisplayString Condition="bIsUObject==0">{Container.Field}</DisplayString>
    <DisplayString Condition="bIsUObject==1">{Container.Object}</DisplayString>
    <Expand>
      <Item Name="Field" Condition="bIsUObject==0">Container.Field</Item>
      <Item Name="Object" Condition="bIsUObject==1">Container.Object</Item>
    </Expand>
  </Type>

  <!-- FField visualizer -->
  <Type Name="FField">
    <DisplayString>(Name={NamePrivate})</DisplayString>
  </Type>

  <!-- FChunkedFixedUObjectArray visualizer -->
  <Type Name="FChunkedFixedUObjectArray">
    <DisplayString Condition="NumElements == 0">Empty</DisplayString>
    <DisplayString Condition="NumElements &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="NumElements &gt; 0">NumElements={NumElements},
NumChunks={NumChunks}, {NumElementsPerChunk}</DisplayString>

    <Expand>
      <IndexListItems Condition="NumElements &gt; 0">
        <Size>NumElements</Size>
        <ValueNode>
          Objects[$i / NumElementsPerChunk][$i % NumElementsPerChunk]
        </ValueNode>
      </IndexListItems>
    </Expand>
  </Type>
```

```
 <!-- TArray<*,TFixedAllocator<*> > visualizer -->
 <Type Name="TArray&lt;*,TFixedAllocator&lt;*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;=
ArrayNum">Num={ArrayNum}</DisplayString>
   <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
       <Size>ArrayNum</Size>
       <ValuePointer>(TArray&lt;$T1,TFixedAllocator&lt;$T2&gt;
&gt;::ElementType*)AllocatorInstance.InlineData</ValuePointer>
     </ArrayItems>
   </Expand>
 </Type>

 <!-- TArray<*,TInlineAllocator<*,*> > visualizer -->
 <Type Name="TArray&lt;*,TInlineAllocator&lt;*,*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;=
ArrayNum">Num={ArrayNum}</DisplayString>
   <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
       <Size>ArrayNum</Size>
       <ValuePointer Condition="AllocatorInstance.SecondaryData.Data ==
0">(TArray&lt;$T1,TInlineAllocator&lt;$T2,$T3&gt;
&gt;::ElementType*)AllocatorInstance.InlineData</ValuePointer>
       <ValuePointer Condition="AllocatorInstance.SecondaryData.Data !=
0">(TArray&lt;$T1,TInlineAllocator&lt;$T2,$T3&gt;
&gt;::ElementType*)AllocatorInstance.SecondaryData.Data</ValuePointer>
     </ArrayItems>
   </Expand>
 </Type>

 <!-- TArray<*,TMemoryImageAllocator<*>> visualizer -->
 <Type Name="TArray&lt;*,TMemoryImageAllocator&lt;*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;=
ArrayNum">Num={ArrayNum}</DisplayString>
   <DisplayString>Ptr={AllocatorInstance.Data.Ptr}</DisplayString>
   <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
```

```
      <Size>ArrayNum</Size>
      <ValuePointer Condition="(AllocatorInstance.Data.OffsetFromThis &amp; 1) !=
0">(TArray&lt;$T1,TMemoryImageAllocator&lt;$T2&gt; &gt;::ElementType*)(
(char*)&amp;AllocatorInstance.Data + (AllocatorInstance.Data.OffsetFromThis >>
1))</ValuePointer>
      <ValuePointer Condition="(AllocatorInstance.Data.OffsetFromThis &amp; 1) ==
0">(TArray&lt;$T1,TMemoryImageAllocator&lt;$T2&gt;
&gt;::ElementType*)AllocatorInstance.Data.Ptr</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>

  <!-- TArray visualizer -->
  <Type Name="TArray&lt;*,*&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;=
ArrayNum">Num={ArrayNum}</DisplayString>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>

<ValuePointer>(TArray&lt;$T1,$T2&gt;::ElementType*)AllocatorInstance.Data</ValuePointe
r>
    </ArrayItems>
   </Expand>
  </Type>

  <!-- TArray<char> visualizers -->
  <Type Name="TArray&lt;char,*&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum}
{(char*)AllocatorInstance.Data,[ArrayNum]s}</DisplayString>
   <StringView Condition="ArrayMax &gt;=
ArrayNum">(char*)AllocatorInstance.Data,[ArrayNum]s</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer>(char*)AllocatorInstance.Data</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>
  <Type Name="TArray&lt;char,TFixedAllocator&lt;*&gt;&gt;">
```

```
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum}
{(char*)AllocatorInstance.InlineData,[ArrayNum]s}</DisplayString>
   <StringView Condition="ArrayMax &gt;=
ArrayNum">(char*)AllocatorInstance.InlineData,[ArrayNum]s</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer>(char*)AllocatorInstance.InlineData</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>
  <Type Name="TArray&lt;char,TInlineAllocator&lt;*,*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data == 0">Num={ArrayNum}
{(char*)AllocatorInstance.InlineData,[ArrayNum]s}</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data != 0">Num={ArrayNum}
{(char*)AllocatorInstance.SecondaryData.Data,[ArrayNum]s}</DisplayString>
   <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data ==
0">(char*)AllocatorInstance.InlineData,[ArrayNum]s</StringView>
   <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data !=
0">(char*)AllocatorInstance.SecondaryData.Data,[ArrayNum]s</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer Condition="AllocatorInstance.SecondaryData.Data ==
0">(char*)AllocatorInstance.InlineData</ValuePointer>
     <ValuePointer Condition="AllocatorInstance.SecondaryData.Data !=
0">(char*)AllocatorInstance.SecondaryData.Data</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>

  <!-- TArray<unsigned char> visualizers -->
  <Type Name="TArray&lt;unsigned char,*&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
```

```
    <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum} {(unsigned
char*)AllocatorInstance.Data,[ArrayNum]s}</DisplayString>
    <StringView Condition="ArrayMax &gt;= ArrayNum">(unsigned
char*)AllocatorInstance.Data,[ArrayNum]s</StringView>
    <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
      <Size>ArrayNum</Size>
      <ValuePointer>(unsigned char*)AllocatorInstance.Data</ValuePointer>
     </ArrayItems>
    </Expand>
  </Type>
  <Type Name="TArray&lt;unsigned char,TFixedAllocator&lt;*&gt;&gt;">
    <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
    <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum} {(unsigned
char*)AllocatorInstance.InlineData,[ArrayNum]s}</DisplayString>
    <StringView Condition="ArrayMax &gt;= ArrayNum">(unsigned
char*)AllocatorInstance.InlineData,[ArrayNum]s</StringView>
    <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
      <Size>ArrayNum</Size>
      <ValuePointer>(unsigned char*)AllocatorInstance.InlineData</ValuePointer>
     </ArrayItems>
    </Expand>
  </Type>
  <Type Name="TArray&lt;unsigned char,TInlineAllocator&lt;*,*&gt;&gt;">
    <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
    <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
    <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data == 0">Num={ArrayNum} {(unsigned
char*)AllocatorInstance.InlineData,[ArrayNum]s}</DisplayString>
    <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data != 0">Num={ArrayNum} {(unsigned
char*)AllocatorInstance.SecondaryData.Data,[ArrayNum]s}</DisplayString>
    <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data == 0">(unsigned
char*)AllocatorInstance.InlineData,[ArrayNum]s</StringView>
    <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data != 0">(unsigned
char*)AllocatorInstance.SecondaryData.Data,[ArrayNum]s</StringView>
    <Expand>
     <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
      <Size>ArrayNum</Size>
      <ValuePointer Condition="AllocatorInstance.SecondaryData.Data == 0">(unsigned
```

```
char*)AllocatorInstance.InlineData</ValuePointer>
      <ValuePointer Condition="AllocatorInstance.SecondaryData.Data != 0">(unsigned
char*)AllocatorInstance.SecondaryData.Data</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>

  <!-- TArray<wchar_t> visualizers -->
  <Type Name="TArray&lt;wchar_t,*&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum}
{(wchar_t*)AllocatorInstance.Data,[ArrayNum]su}</DisplayString>
   <StringView Condition="ArrayMax &gt;=
ArrayNum">(wchar_t*)AllocatorInstance.Data,[ArrayNum]su</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer>(wchar_t*)AllocatorInstance.Data</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>
  <Type Name="TArray&lt;wchar_t,TFixedAllocator&lt;*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum">Num={ArrayNum}
{(wchar_t*)AllocatorInstance.InlineData,[ArrayNum]su}</DisplayString>
   <StringView Condition="ArrayMax &gt;=
ArrayNum">(wchar_t*)AllocatorInstance.InlineData,[ArrayNum]su</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer>(wchar_t*)AllocatorInstance.InlineData</ValuePointer>
    </ArrayItems>
   </Expand>
  </Type>
  <Type Name="TArray&lt;wchar_t,TInlineAllocator&lt;*,*&gt;&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &lt; ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data == 0">Num={ArrayNum}
{(wchar_t*)AllocatorInstance.InlineData,[ArrayNum]su}</DisplayString>
   <DisplayString Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
```

```
AllocatorInstance.SecondaryData.Data != 0">Num={ArrayNum}
{(wchar_t*)AllocatorInstance.SecondaryData.Data,[ArrayNum]su}</DisplayString>
   <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data ==
0">(wchar_t*)AllocatorInstance.InlineData,[ArrayNum]su</StringView>
   <StringView Condition="ArrayMax &gt;= ArrayNum &amp;&amp;
AllocatorInstance.SecondaryData.Data !=
0">(wchar_t*)AllocatorInstance.SecondaryData.Data,[ArrayNum]su</StringView>
   <Expand>
    <ArrayItems Condition="ArrayNum &lt;= ArrayMax">
     <Size>ArrayNum</Size>
     <ValuePointer Condition="AllocatorInstance.SecondaryData.Data ==
0">(wchar_t*)AllocatorInstance.InlineData</ValuePointer>
     <ValuePointer Condition="AllocatorInstance.SecondaryData.Data !=
0">(wchar_t*)AllocatorInstance.SecondaryData.Data</ValuePointer>
    </ArrayItems>
   </Expand>
 </Type>

 <!-- TArrayView visualizer -->
 <Type Name="TArrayView&lt;*&gt;">
   <DisplayString Condition="ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="ArrayNum &gt; 0">Num={ArrayNum}</DisplayString>
   <Expand>
    <ArrayItems Condition="ArrayNum &gt; 0">
     <Size>ArrayNum</Size>
     <ValuePointer>DataPtr</ValuePointer>
    </ArrayItems>
   </Expand>
 </Type>

 <!-- TIndirectArray visualizer -->
 <Type Name="TIndirectArray&lt;*,*&gt;">
   <DisplayString Condition="Array.ArrayNum == 0">Empty</DisplayString>
   <DisplayString Condition="Array.ArrayNum &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="Array.ArrayMax &lt;
Array.ArrayNum">Invalid</DisplayString>
   <DisplayString Condition="Array.ArrayMax &gt;=
Array.ArrayNum">Num={Array.ArrayNum}</DisplayString>
   <Expand>
    <IndexListItems Condition="Array.ArrayNum &lt;= Array.ArrayMax">
     <Size>Array.ArrayNum</Size>

<ValueNode>*((TIndirectArray&lt;$T1,$T2&gt;::ElementType**)Array.AllocatorInstance.Da
ta)[$i]</ValueNode>
```

```
    </IndexListItems>
   </Expand>
  </Type>

  <!-- TChunkedArray visualizer -->
  <Type Name="TChunkedArray&lt;*,*&gt;">
   <DisplayString Condition="NumElements == 0">Empty</DisplayString>
   <DisplayString Condition="NumElements &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="NumElements &gt; 0">NumElements={NumElements},
NumChunks={Chunks.Array.ArrayNum}, {NumElementsPerChunk}</DisplayString>

   <Expand>
    <IndexListItems Condition="NumElements &gt; 0">
     <Size>NumElements</Size>
     <ValueNode>
      *(
      *(

(TChunkedArray&lt;$T1,$T2&gt;::ElementType**)Chunks.Array.AllocatorInstance.Data +
($i / NumElementsPerChunk)
       ) + ($i % NumElementsPerChunk)
       )
     </ValueNode>
    </IndexListItems>
   </Expand>
  </Type>

  <!-- TSparseArray visualizer -->
  <Type Name="TSparseArray&lt;*,*&gt;">
   <DisplayString Condition="(Data.ArrayNum - NumFreeIndices) &lt;=
0">Empty</DisplayString>
   <DisplayString Condition="Data.ArrayNum &lt;=
Data.ArrayMax">Num={Data.ArrayNum - NumFreeIndices}</DisplayString>
   <Expand>
    <IndexListItems Condition="Data.ArrayNum &gt; 0 &amp;&amp; Data.ArrayNum &lt;=
Data.ArrayMax">
     <Size>Data.ArrayNum</Size>
     <ValueNode Condition="(AllocationFlags.AllocatorInstance.SecondaryData.Data != 0)
&amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocationFlags.AllocatorInstance.SecondaryData.Data)[$i/3
2]&gt;&gt;($i%32) &amp; 1) !=
0">*reinterpret_cast&lt;ElementType*&gt;(reinterpret_cast&lt;FElementOrFreeListLink*&gt
;(Data.AllocatorInstance.Data) + $i)</ValueNode>
     <ValueNode Condition="(AllocationFlags.AllocatorInstance.SecondaryData.Data == 0)
&amp;&amp; (reinterpret_cast&lt;uint32*&gt;(AllocationFlags.AllocatorInstance.InlineData
)[$i/32]&gt;&gt;($i%32) &amp; 1) !=
```

```
0">*reinterpret_cast&lt;ElementType*&gt;(reinterpret_cast&lt;FElementOrFreeListLink*&gt
;(Data.AllocatorInstance.Data) + $i)</ValueNode>
      <ValueNode Condition="(AllocationFlags.AllocatorInstance.SecondaryData.Data != 0)
&amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocationFlags.AllocatorInstance.SecondaryData.Data)[$i/3
2]&gt;&gt;($i%32) &amp; 1) == 0">"Invalid",sb</ValueNode>
      <ValueNode Condition="(AllocationFlags.AllocatorInstance.SecondaryData.Data == 0)
&amp;&amp; (reinterpret_cast&lt;uint32*&gt;(AllocationFlags.AllocatorInstance.InlineData
)[$i/32]&gt;&gt;($i%32) &amp; 1) == 0">"Invalid",sb</ValueNode>
    </IndexListItems>
   </Expand>
  </Type>

  <!-- TBitArray visualizer -->
  <Type Name="TBitArray&lt;*&gt;">
   <DisplayString Condition="NumBits == 0">Empty</DisplayString>
   <DisplayString Condition="NumBits &lt; 0">Invalid</DisplayString>
   <DisplayString Condition="NumBits &gt; 0">NumBits={NumBits},
MaxBits={MaxBits}</DisplayString>
   <Expand>
    <IndexListItems Condition="NumBits &gt; 0">
     <Size>NumBits</Size>
     <ValueNode Condition="(AllocatorInstance.SecondaryData.Data != 0) &amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocatorInstance.SecondaryData.Data
)[$i/32]&gt;&gt;($i%32) &amp; 1) != 0">1</ValueNode>
     <ValueNode Condition="(AllocatorInstance.SecondaryData.Data == 0) &amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocatorInstance.InlineData
)[$i/32]&gt;&gt;($i%32) &amp; 1) != 0">1</ValueNode>
     <ValueNode Condition="(AllocatorInstance.SecondaryData.Data != 0) &amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocatorInstance.SecondaryData.Data
)[$i/32]&gt;&gt;($i%32) &amp; 1) == 0">0</ValueNode>
     <ValueNode Condition="(AllocatorInstance.SecondaryData.Data == 0) &amp;&amp;
(reinterpret_cast&lt;uint32*&gt;(AllocatorInstance.InlineData
)[$i/32]&gt;&gt;($i%32) &amp; 1) == 0">0</ValueNode>
    </IndexListItems>
   </Expand>
  </Type>

  <!-- TRingBuffer visualizer -->
  <Type Name="TRingBuffer&lt;*,*&gt;">
   <DisplayString Condition="AfterBack == Front">Empty</DisplayString>
   <DisplayString Condition="AfterBack - Front &lt;= IndexMask+1">Num={AfterBack -
Front}</DisplayString>
   <DisplayString Condition="AfterBack - Front &gt;
IndexMask+1">Invalid</DisplayString>
   <Expand>
```

```
    <IndexListItems Condition="AfterBack - Front &lt;= IndexMask+1">
      <Size>AfterBack - Front</Size>
      <ValueNode>AllocationData[(Front + $i) &amp; IndexMask]</ValueNode>
    </IndexListItems>
  </Expand>
 </Type>

 <!-- TPair visualizer -->
 <Type Name="TPair&lt;*,*&gt;">
  <DisplayString>({Key}, {Value})</DisplayString>
 </Type>

 <!-- TSharedPtr visualizer -->
 <Type Name="TSharedPtr&lt;*,*&gt;">
  <DisplayString Condition="Object == 0">Null</DisplayString>
  <DisplayString Condition="Object != 0">Ptr={(void*)Object},
SharedRefs={SharedReferenceCount.ReferenceController->SharedReferenceCount},
WeakRefs={SharedReferenceCount.ReferenceController->WeakReferenceCount},
Object={*Object}</DisplayString>
  <Expand>
    <Item Condition="Object != 0"
Name="[SharedReferenceCount]">SharedReferenceCount.ReferenceController->SharedRefer
enceCount</Item>
    <Item Condition="Object != 0"
Name="[WeakReferenceCount]">SharedReferenceCount.ReferenceController->WeakReferen
ceCount</Item>
    <Item Condition="Object != 0" Name="[Ptr]">(void*)Object</Item>
    <ExpandedItem Condition="Object != 0">*Object</ExpandedItem>
  </Expand>
 </Type>

 <!-- TSharedRef visualizer -->
 <Type Name="TSharedRef&lt;*,*&gt;">
  <DisplayString Condition="Object != 0">Ptr={(void*)Object},
SharedRefs={SharedReferenceCount.ReferenceController->SharedReferenceCount},
WeakRefs={SharedReferenceCount.ReferenceController->WeakReferenceCount},
Object={*Object}</DisplayString>
  <Expand>
    <Item Condition="Object != 0"
Name="[SharedReferenceCount]">SharedReferenceCount.ReferenceController->SharedRefer
enceCount</Item>
    <Item Condition="Object != 0"
Name="[WeakReferenceCount]">SharedReferenceCount.ReferenceController->WeakReferen
ceCount</Item>
    <Item Condition="Object != 0" Name="[Ptr]">(void*)Object</Item>
    <ExpandedItem Condition="Object != 0">*Object</ExpandedItem>
```

```
    </Expand>
  </Type>

  <!-- TWeakPtr visualizer -->
  <Type Name="TWeakPtr&lt;*,*&gt;">
    <DisplayString Condition="Object == 0">Null</DisplayString>
    <DisplayString
Condition="WeakReferenceCount.ReferenceController->SharedReferenceCount == 0">Object
has been destroyed</DisplayString>
    <DisplayString Condition="Object != 0">Ptr={(void*)Object},
SharedRefs={WeakReferenceCount.ReferenceController->SharedReferenceCount},
WeakRefs={WeakReferenceCount.ReferenceController->WeakReferenceCount},
Object={*Object}</DisplayString>
    <Expand>
      <Item Condition="Object != 0"
Name="[SharedReferenceCount]">WeakReferenceCount.ReferenceController->SharedRefere
nceCount</Item>
      <Item Condition="Object != 0"
Name="[WeakReferenceCount]">WeakReferenceCount.ReferenceController->WeakReferenc
eCount</Item>
      <Item Condition="Object != 0 &amp;amp;&amp;amp;
WeakReferenceCount.ReferenceController->SharedReferenceCount > 0"
Name="[Ptr]">(void*)Object</Item>
      <ExpandedItem Condition="Object != 0 &amp;amp;&amp;amp;
WeakReferenceCount.ReferenceController->SharedReferenceCount >
0">*Object</ExpandedItem>
    </Expand>
  </Type>

  <!-- TMemoryImagePtr visualizer -->
  <Type Name="TMemoryImagePtr&lt;*&gt;">
    <DisplayString Condition="Ptr == 0">Null</DisplayString>
    <DisplayString Condition="Ptr != 0"></DisplayString>
    <Expand>
      <Item Condition="Ptr != 0 &amp;amp;&amp;amp; (OffsetFromThis &amp;amp; 1) != 0"
Name="Object">($T1*)( (char*)this + (OffsetFromThis >> 1))</Item>
      <Item Condition="Ptr != 0 &amp;amp;&amp;amp; (OffsetFromThis &amp;amp; 1) == 0"
Name="Object">($T1*)Ptr</Item>
    </Expand>
  </Type>

  <Type Name="FHashedNameDebugString">
    <DisplayString Condition="String.Ptr == 0">Empty</DisplayString>
    <DisplayString Condition="(String.OffsetFromThis &amp;amp; 1) != 0">{(char*)this +
(String.OffsetFromThis >> 1)}</DisplayString>
    <DisplayString>{String.Ptr}</DisplayString>
```

```
  </Type>

  <!-- TMapBase visualizer -->
  <Type Name="TMapBase&lt;*,*,*,*&gt;">
    <DisplayString>{Pairs}</DisplayString>
    <Expand>
      <ExpandedItem>Pairs</ExpandedItem>
    </Expand>
  </Type>

  <!-- TSet visualizer -->
  <Type Name="TSet&lt;*,*,*&gt;">
    <DisplayString Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices &lt;=
0">Empty</DisplayString>
    <DisplayString Condition="Elements.Data.ArrayNum &lt;=
Elements.Data.ArrayMax">Num={Elements.Data.ArrayNum -
Elements.NumFreeIndices}</DisplayString>
    <Expand>
      <CustomListItems Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices
&gt; 0 &amp;&amp; Elements.Data.ArrayNum &lt;= Elements.Data.ArrayMax">
        <Variable Name="Index" InitialValue="0" />
        <Size>Elements.Data.ArrayNum - Elements.NumFreeIndices</Size>
        <Loop>
          <If Condition="
                ((Elements.AllocationFlags.AllocatorInstance.SecondaryData.Data != 0)
&amp;&amp;
((reinterpret_cast&lt;uint32*&gt;(Elements.AllocationFlags.AllocatorInstance.SecondaryData.
Data)[Index/32]&gt;&gt;(Index%32)) &amp; 1) != 0)
             || ((Elements.AllocationFlags.AllocatorInstance.SecondaryData.Data == 0)
&amp;&amp;
((reinterpret_cast&lt;uint32*&gt;(Elements.AllocationFlags.AllocatorInstance.InlineData
)[Index/32]&gt;&gt;(Index%32)) &amp; 1) != 0)
          ">
            <Item>((TSetElement &lt;$T1&gt;
*)Elements.Data.AllocatorInstance.Data)[Index].Value</Item>
          </If>
          <Exec>++Index</Exec>
        </Loop>
      </CustomListItems>
    </Expand>
  </Type>

  <!-- TSet<*,*,TInlineSetAllocator<*>> visualizer -->
  <Type Name="TSet&lt;*,*,TInlineSetAllocator&lt;*&gt;&gt;">
    <DisplayString Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices &lt;=
0">Empty</DisplayString>
```

```
   <DisplayString Condition="Elements.Data.ArrayNum &lt;=
Elements.Data.ArrayMax">Num={Elements.Data.ArrayNum -
Elements.NumFreeIndices}</DisplayString>
   <Expand>
    <CustomListItems Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices
&gt; 0 &amp;&amp; Elements.Data.ArrayNum &lt;= Elements.Data.ArrayMax">
      <Variable Name="Index" InitialValue="0" />
      <Size>Elements.Data.ArrayNum - Elements.NumFreeIndices</Size>
      <Loop>
       <If Condition="
            ((Elements.AllocationFlags.AllocatorInstance.SecondaryData.Data != 0)
&amp;&amp;
((reinterpret_cast&lt;uint32*&gt;(Elements.AllocationFlags.AllocatorInstance.SecondaryData.
Data)[Index/32]&gt;&gt;(Index%32)) &amp; 1) != 0)
          || ((Elements.AllocationFlags.AllocatorInstance.SecondaryData.Data == 0)
&amp;&amp;
((reinterpret_cast&lt;uint32*&gt;(Elements.AllocationFlags.AllocatorInstance.InlineData
)[Index/32]&gt;&gt;(Index%32)) &amp; 1) != 0)
        ">
        <If Condition="Elements.Data.AllocatorInstance.SecondaryData.Data == 0">
         <Item>((TSetElement&lt;$T1&gt;
*)Elements.Data.AllocatorInstance.InlineData)[Index].Value</Item>
        </If>
        <If Condition="Elements.Data.AllocatorInstance.SecondaryData.Data != 0">
         <Item>((TSetElement&lt;$T1&gt;
*)Elements.Data.AllocatorInstance.SecondaryData.Data)[Index].Value</Item>
        </If>
       </If>
       <Exec>++Index</Exec>
      </Loop>
    </CustomListItems>
   </Expand>
  </Type>

  <!-- TSet<*,*,TFixedSetAllocator<*>> visualizer -->
  <Type Name="TSet&lt;*,*,TFixedSetAllocator&lt;*&gt;&gt;">
   <DisplayString Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices &lt;=
0">Empty</DisplayString>
   <DisplayString Condition="Elements.Data.ArrayNum &lt;=
Elements.Data.ArrayMax">Num={Elements.Data.ArrayNum -
Elements.NumFreeIndices}</DisplayString>
   <Expand>
    <CustomListItems Condition="Elements.Data.ArrayNum - Elements.NumFreeIndices
&gt; 0 &amp;&amp; Elements.Data.ArrayNum &lt;= Elements.Data.ArrayMax">
      <Variable Name="Index" InitialValue="0" />
      <Size>Elements.Data.ArrayNum - Elements.NumFreeIndices</Size>
```

```
      <Loop>
        <If
Condition="((reinterpret_cast&lt;uint32*&gt;(Elements.AllocationFlags.AllocatorInstance.Inli
neData)[Index/32]&gt;&gt;(Index%32)) &amp; 1) != 0">
          <Item>((TSetElement&lt;$T1&gt;
*)Elements.Data.AllocatorInstance.InlineData)[Index].Value</Item>
        </If>
        <Exec>++Index</Exec>
      </Loop>
    </CustomListItems>
  </Expand>
 </Type>

 <!-- FWeakObjectPtr visualizer -->
 <Type Name="FWeakObjectPtr">
  <DisplayString Condition="ObjectSerialNumber &lt; 1">nullptr</DisplayString>
  <DisplayString Condition="GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].SerialNumber !=
ObjectSerialNumber">STALE</DisplayString>
  <DisplayString>{GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].Object}</DisplayString>
  <Expand>
   <ExpandedItem>GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].Object</ExpandedItem>
  </Expand>
 </Type>

 <!-- TWeakObjectPtr<*> visualizer -->
 <Type Name="TWeakObjectPtr&lt;*&gt;">
  <DisplayString Condition="ObjectSerialNumber &lt; 1">nullptr</DisplayString>
  <DisplayString Condition="GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].SerialNumber !=
ObjectSerialNumber">STALE</DisplayString>
  <DisplayString>{($T1*)GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].Object}</DisplayString>
  <Expand>
   <ExpandedItem>($T1*)GObjectArrayForDebugVisualizers->Objects[ObjectIndex /
65536][ObjectIndex % 65536].Object</ExpandedItem>
  </Expand>
 </Type>

 <!-- FSubobjectPtr visualizer -->
 <Type Name="FSubobjectPtr">
  <DisplayString>{Object}</DisplayString>
 </Type>
```

```
<!-- FVertexID visualizer -->
<Type Name="FVertexID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- FVertexInstanceID visualizer -->
<Type Name="FVertexInstanceID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- FEdgeID visualizer -->
<Type Name="FEdgeID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- FPolygonID visualizer -->
<Type Name="FPolygonID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- FPolygonGroupID visualizer -->
<Type Name="FPolygonGroupID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- FTriangleID visualizer -->
<Type Name="FTriangleID">
  <DisplayString>{IDValue}</DisplayString>
</Type>

<!-- TOptional visualizer -->
<Type Name="TOptional&lt;*&gt;">
  <DisplayString Condition="!bIsSet">Unset</DisplayString>
  <DisplayString Condition="bIsSet">Set: {{{*($T1*)&amp;Value}}}</DisplayString>
  <Expand>
    <ExpandedItem Condition="bIsSet">*($T1*)&amp;Value</ExpandedItem>
  </Expand>
</Type>

<!-- TUnion visualizer -->
<Type Name="TUnion&lt;*&gt;">
  <DisplayString>TUnion{{{CurrentSubtypeIndex}}}</DisplayString>
  <Expand>
    <Item Name="[TUnion.CurrentSubtypeIndex]">CurrentSubtypeIndex</Item>
    <ExpandedItem
Condition="CurrentSubtypeIndex==0">*($T1*)&amp;Values.A</ExpandedItem>
```

```
    <ExpandedItem
Condition="CurrentSubtypeIndex==1">*($T2*)&amp;amp;Values.B</ExpandedItem>
    <ExpandedItem
Condition="CurrentSubtypeIndex==2">*($T3*)&amp;amp;Values.C</ExpandedItem>
    <ExpandedItem
Condition="CurrentSubtypeIndex==3">*($T4*)&amp;amp;Values.D</ExpandedItem>
    <ExpandedItem
Condition="CurrentSubtypeIndex==4">*($T5*)&amp;amp;Values.E</ExpandedItem>
    <ExpandedItem
Condition="CurrentSubtypeIndex==5">*($T6*)&amp;amp;Values.F</ExpandedItem>
   </Expand>
  </Type>

  <!-- TInlineValue visualizer -->
  <Type Name="TInlineValue&lt;*&gt;">
   <DisplayString Condition="!bIsValid">Null</DisplayString>
   <DisplayString Condition="bIsValid &amp;amp;&amp;amp;
bInline">{{{*($T1*)&amp;amp;Data}}}</DisplayString>
   <DisplayString Condition="bIsValid &amp;amp;&amp;amp;
!bInline">{{{**(($T1**)&amp;amp;Data)}}}</DisplayString>
   <Expand>
    <ExpandedItem Condition="bIsValid &amp;amp;&amp;amp;
bInline">*($T1*)&amp;amp;Data</ExpandedItem>
    <ExpandedItem Condition="bIsValid &amp;amp;&amp;amp;
!bInline">**(($T1**)&amp;amp;Data)</ExpandedItem>
   </Expand>
  </Type>

  <!-- TFunction visualizer -->
  <Type Name="UE4Function_Private::TDebugHelper&lt;*&gt;">
   <DisplayString>{*Ptr}</DisplayString>
   <Expand>
    <Item Name="[Lambda]">*Ptr</Item>
   </Expand>
  </Type>
  <Type Name="TFunctionRef&lt;*&gt;">
   <DisplayString Condition="Callable">{DebugPtrStorage}</DisplayString>
   <DisplayString Condition="!Callable">Unset</DisplayString>
   <Expand>
    <ExpandedItem Condition="Callable">DebugPtrStorage</ExpandedItem>
   </Expand>
  </Type>
  <Type Name="TFunction&lt;*&gt;">
   <AlternativeType Name="TUniqueFunction&lt;*&gt;"></AlternativeType>
   <DisplayString Condition="Callable != 0">{DebugPtrStorage}</DisplayString>
   <DisplayString Condition="Callable == 0">Unset</DisplayString>
```

```
    <Expand>
      <ExpandedItem Condition="Callable != 0">DebugPtrStorage</ExpandedItem>
    </Expand>
  </Type>

  <!-- FGameplayTagContainer visualizer -->
  <Type Name="FGameplayTagContainer">
    <DisplayString Condition="GameplayTags.ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="GameplayTags.ArrayNum ==
1">Tag={*((FName*)(GameplayTags.AllocatorInstance.Data))}</DisplayString>
    <DisplayString Condition="GameplayTags.ArrayNum >
1">Num={GameplayTags.ArrayNum}</DisplayString>
    <Expand>
      <ArrayItems Condition="GameplayTags.ArrayNum &lt;= GameplayTags.ArrayMax">
        <Size>GameplayTags.ArrayNum</Size>
        <ValuePointer>((FName*)(GameplayTags.AllocatorInstance.Data))</ValuePointer>
      </ArrayItems>
    </Expand>
  </Type>

  <!-- FActorRepList visualizer -->
  <Type Name="FActorRepList">
    <DisplayString >({Num}/{Max} {RefCount})</DisplayString>
    <Expand>
      <Item Name="[RefCount]">RefCount</Item>
      <Item Name="[Num]">Num</Item>
      <Item Name="[Max]">Max</Item>
      <ArrayItems>
        <Size>Num</Size>
        <ValuePointer>((FActorRepListType*)(Data))</ValuePointer>
      </ArrayItems>
    </Expand>
  </Type>

  <!-- TTuple visualizer -->
  <Type Name="TTuple&lt;&gt;">
    <DisplayString>{{}}</DisplayString>
    <Expand>
    </Expand>
  </Type>
  <Type Name="TTuple&lt;*&gt;">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,1&gt;*)this).Value}
}}</DisplayString>
    <Expand>
      <Item
```

```
Name="[0]">((UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,1&gt;*)this)->Value</Item>
    </Expand>
  </Type>
  <Type Name="TTuple&lt;*,*&gt;">
    <DisplayString>{{{Key},{Value}}}</DisplayString>
    <Expand>
      <Item Name="[0:Key]">Key</Item>
      <Item Name="[1:Value]">Value</Item>
    </Expand>
  </Type>
  <Type Name="TTuple&lt;*,*,*&gt;">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,3&gt;*)this).Value},
{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,3&gt;*)this).Value},{(*(UE4Tuple_Pri
vate::TTupleBaseElement&lt;$T3,2,3&gt;*)this).Value}}}</DisplayString>
    <Expand>
      <Item
Name="[0]">((UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,3&gt;*)this)->Value</Item>
      <Item
Name="[1]">((UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,3&gt;*)this)->Value</Item>
      <Item
Name="[2]">((UE4Tuple_Private::TTupleBaseElement&lt;$T3,2,3&gt;*)this)->Value</Item>
    </Expand>
  </Type>
  <Type Name="TTuple&lt;*,*,*,*&gt;">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,4&gt;*)this).Value},
{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,4&gt;*)this).Value},{(*(UE4Tuple_Pri
vate::TTupleBaseElement&lt;$T3,2,4&gt;*)this).Value},{(*(UE4Tuple_Private::TTupleBaseE
lement&lt;$T4,3,4&gt;*)this).Value}}}</DisplayString>
    <Expand>
      <Item
Name="[0]">((UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,4&gt;*)this)->Value</Item>
      <Item
Name="[1]">((UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,4&gt;*)this)->Value</Item>
      <Item
Name="[2]">((UE4Tuple_Private::TTupleBaseElement&lt;$T3,2,4&gt;*)this)->Value</Item>
      <Item
Name="[3]">((UE4Tuple_Private::TTupleBaseElement&lt;$T4,3,4&gt;*)this)->Value</Item>
    </Expand>
  </Type>
  <Type Name="TTuple&lt;*,*,*,*,*&gt;">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,5&gt;*)this).Value},
{(*(UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,5&gt;*)this).Value},{(*(UE4Tuple_Pri
vate::TTupleBaseElement&lt;$T3,2,5&gt;*)this).Value},{(*(UE4Tuple_Private::TTupleBaseE
```

lement<$T4,3,5>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T5,4,5>*)this).Value}}}</DisplayString>
<Expand>
<Item Name="[0]">((UE4Tuple_Private::TTupleBaseElement<$T1,0,5>*)this)->Value</Item>
<Item Name="[1]">((UE4Tuple_Private::TTupleBaseElement<$T2,1,5>*)this)->Value</Item>
<Item Name="[2]">((UE4Tuple_Private::TTupleBaseElement<$T3,2,5>*)this)->Value</Item>
<Item Name="[3]">((UE4Tuple_Private::TTupleBaseElement<$T4,3,5>*)this)->Value</Item>
<Item Name="[4]">((UE4Tuple_Private::TTupleBaseElement<$T5,4,5>*)this)->Value</Item>
</Expand>
</Type>
<Type Name="TTuple<*,*,*,*,*,*>">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement<$T1,0,6>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T2,1,6>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T3,2,6>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T4,3,6>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T5,4,6>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T6,5,6>*)this).Value}}}</DisplayString>
<Expand>
<Item Name="[0]">((UE4Tuple_Private::TTupleBaseElement<$T1,0,6>*)this)->Value</Item>
<Item Name="[1]">((UE4Tuple_Private::TTupleBaseElement<$T2,1,6>*)this)->Value</Item>
<Item Name="[2]">((UE4Tuple_Private::TTupleBaseElement<$T3,2,6>*)this)->Value</Item>
<Item Name="[3]">((UE4Tuple_Private::TTupleBaseElement<$T4,3,6>*)this)->Value</Item>
<Item Name="[4]">((UE4Tuple_Private::TTupleBaseElement<$T5,4,6>*)this)->Value</Item>
<Item Name="[5]">((UE4Tuple_Private::TTupleBaseElement<$T6,5,6>*)this)->Value</Item>
</Expand>
</Type>
<Type Name="TTuple<*,*,*,*,*,*,*>">

<DisplayString>{{{(*(UE4Tuple_Private::TTupleBaseElement<$T1,0,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T2,1,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T3,2,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T4,3,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T5,4,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T6,5,7>*)this).Value},{(*(UE4Tuple_Private::TTupleBaseElement<$T7,6,7>*)this).Value}}}</DisplayString>

```
    <Expand>
      <Item
Name="[0]">((UE4Tuple_Private::TTupleBaseElement&lt;$T1,0,7&gt;*)this)->Value</Item>
      <Item
Name="[1]">((UE4Tuple_Private::TTupleBaseElement&lt;$T2,1,7&gt;*)this)->Value</Item>
      <Item
Name="[2]">((UE4Tuple_Private::TTupleBaseElement&lt;$T3,2,7&gt;*)this)->Value</Item>
      <Item
Name="[3]">((UE4Tuple_Private::TTupleBaseElement&lt;$T4,3,7&gt;*)this)->Value</Item>
      <Item
Name="[4]">((UE4Tuple_Private::TTupleBaseElement&lt;$T5,4,7&gt;*)this)->Value</Item>
      <Item
Name="[5]">((UE4Tuple_Private::TTupleBaseElement&lt;$T6,5,7&gt;*)this)->Value</Item>
      <Item
Name="[6]">((UE4Tuple_Private::TTupleBaseElement&lt;$T7,6,7&gt;*)this)->Value</Item>
    </Expand>
  </Type>

  <!-- TDelegateBase visualizer -->
  <Type Name="TBaseStaticDelegateInstance&lt;*&gt;">
    <DisplayString>{StaticFuncPtr}</DisplayString>
    <Expand>
      <Item Name="[StaticFuncPtr]">StaticFuncPtr</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TBaseFunctorDelegateInstance&lt;*&gt;">
    <DisplayString>{Functor}</DisplayString>
    <Expand>
      <Item Name="[Functor]">Functor</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TBaseRawMethodDelegateInstance&lt;*,*,*&gt;">
    <DisplayString>{UserObject}</DisplayString>
    <Expand>
      <Item Name="[UserObject]">UserObject</Item>
      <Item Name="[MethodPtr]">MethodPtr</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TBaseSPMethodDelegateInstance&lt;*,*,*&gt;">
    <DisplayString>{UserObject}</DisplayString>
```

```
    <Expand>
      <Item Name="[UserObject]">UserObject</Item>
      <Item Name="[MethodPtr]">MethodPtr</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TBaseUObjectMethodDelegateInstance&lt;*,*,*&gt;">
    <DisplayString>{UserObject}</DisplayString>
    <Expand>
      <Item Name="[UserObject]">UserObject</Item>
      <Item Name="[MethodPtr]">MethodPtr</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TBaseUFunctionDelegateInstance&lt;*,*&gt;">
    <DisplayString>{UserObjectPtr}</DisplayString>
    <Expand>
      <Item Name="[UserObject]">UserObjectPtr</Item>
      <Item Name="[FunctionName]">FunctionName</Item>
      <Item Name="[Payload]">Payload</Item>
      <Item Name="[Handle]">Handle</Item>
    </Expand>
  </Type>
  <Type Name="TDelegateBase&lt;*&gt;" Priority="MediumLow">
    <DisplayString Condition="DelegateSize == 0">Unbound</DisplayString>
    <DisplayString Condition="DelegateSize &lt;=
2">{*((IDelegateInstance*)(DelegateAllocator.InlineData))}</DisplayString>
    <DisplayString Condition="DelegateSize &gt;
2">{*((IDelegateInstance*)(DelegateAllocator.SecondaryData.Data))}</DisplayString>
    <Expand>
      <ExpandedItem Condition="DelegateSize == 0">DelegateAllocator</ExpandedItem>
      <ExpandedItem Condition="DelegateSize &lt;=
2">*((IDelegateInstance*)(DelegateAllocator.InlineData))</ExpandedItem>
      <ExpandedItem Condition="DelegateSize &gt;
2">*((IDelegateInstance*)(DelegateAllocator.SecondaryData.Data))</ExpandedItem>
    </Expand>
  </Type>
  <Type Name="TDelegateBase&lt;*&gt;">
    <DisplayString Condition="DelegateSize == 0">Unbound</DisplayString>
    <DisplayString Condition="DelegateSize !=
0">{*((IDelegateInstance*)(DelegateAllocator.Data))}</DisplayString>
    <Expand>
      <ExpandedItem Condition="DelegateSize == 0">DelegateAllocator</ExpandedItem>
      <ExpandedItem Condition="DelegateSize !=
```

```
0">*((IDelegateInstance*)(DelegateAllocator.Data))</ExpandedItem>
    </Expand>
  </Type>
  <Type Name="TMulticastDelegate&lt;*,*&gt;" Priority="MediumLow">
    <DisplayString Condition="InvocationList.ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="InvocationList.ArrayNum !=
0">Num={InvocationList.ArrayNum}</DisplayString>
    <Expand>
      <ArrayItems>
        <Size>InvocationList.ArrayNum</Size>
        <ValuePointer Condition="InvocationList.AllocatorInstance.SecondaryData.Data ==
0">(TDelegateBase&lt;$T2&gt;*)(InvocationList.AllocatorInstance.InlineData)</ValuePointe
r>
        <ValuePointer Condition="InvocationList.AllocatorInstance.SecondaryData.Data !=
0">(TDelegateBase&lt;$T2&gt;*)(InvocationList.AllocatorInstance.SecondaryData.Data)</Va
luePointer>
      </ArrayItems>
    </Expand>
  </Type>
  <Type Name="TMulticastDelegate&lt;*,*&gt;">
    <DisplayString Condition="InvocationList.ArrayNum == 0">Empty</DisplayString>
    <DisplayString Condition="InvocationList.ArrayNum !=
0">Num={InvocationList.ArrayNum}</DisplayString>
    <Expand>
      <ArrayItems>
        <Size>InvocationList.ArrayNum</Size>

<ValuePointer>(TDelegateBase&lt;$T2&gt;*)(InvocationList.AllocatorInstance.Data)</Value
Pointer>
      </ArrayItems>
    </Expand>
  </Type>

  <!-- TRange visualizer -->
  <Type Name="TAtomic&lt;*&gt;">
    <DisplayString>{Element}</DisplayString>
    <Expand>
      <ExpandedItem>Element</ExpandedItem>
    </Expand>
  </Type>

  <!-- TRange visualizer -->
  <Type Name="TRange&lt;*&gt;">
    <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Open
&amp;&amp; UpperBound.Type.Value == ERangeBoundTypes::Open">[-&#8734;,
+&#8734;]</DisplayString>
```

```
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Open
&amp;&amp; UpperBound.Type.Value == ERangeBoundTypes::Inclusive">[-&#8734;,
{UpperBound.Value}]</DisplayString>
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Open
&amp;&amp; UpperBound.Type.Value == ERangeBoundTypes::Exclusive">[-&#8734;,
{UpperBound.Value})</DisplayString>

   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Inclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Open">[{LowerBound.Value}, +&#8734;)</DisplayString>
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Inclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Inclusive">[{LowerBound.Value},
{UpperBound.Value}]</DisplayString>
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Inclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Exclusive">[{LowerBound.Value},
{UpperBound.Value})</DisplayString>

   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Exclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Open">({LowerBound.Value}, +&#8734;]</DisplayString>
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Exclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Inclusive">({LowerBound.Value},
{UpperBound.Value}]</DisplayString>
   <DisplayString Condition="LowerBound.Type.Value == ERangeBoundTypes::Exclusive
&amp;&amp; UpperBound.Type.Value ==
ERangeBoundTypes::Exclusive">({LowerBound.Value},
{UpperBound.Value})</DisplayString>
  </Type>

  <!-- FFrameNumber visualizer -->
  <Type Name="FFrameNumber">
   <DisplayString>{Value}</DisplayString>
  </Type>

  <!-- FFrameTime visualizer -->
  <Type Name="FFrameTime">
   <DisplayString Condition="SubFrame == 0.0">{FrameNumber}</DisplayString>
   <DisplayString Condition="SubFrame !=
0.0">{(double)(FrameNumber.Value)+SubFrame}</DisplayString>
  </Type>

  <!-- FRHICommandList visualizer -->
  <Type Name="FRHICommandBase">
```

```
  <DisplayString>{{ RHI Command -> { this->__vfptr[0] } }}</DisplayString>
 </Type>

 <Type Name="FRHICommandList">
  <Expand>
   <LinkedListItems>
    <HeadPointer>Root</HeadPointer>
    <NextPointer>Next</NextPointer>
    <ValueNode>this</ValueNode>
   </LinkedListItems>
  </Expand>
 </Type>

        <!-- TVariant visualizer -->
        <Type Name="TVariant&lt;*&gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
                <Expand>
                        <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;Storage</Item>
                        <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>
        <Type Name="TVariant&lt;*,*&gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
                <Expand>
                        <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;Storage</Item>
                        <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>
        <Type Name="TVariant&lt;*,*,*&gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
```

```
            <Expand>
                <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;amp;Storage</Item>
                <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                <Item Name="TypeIndex">TypeIndex</Item>
            </Expand>
        </Type>
        <Type Name="TVariant&amp;lt;*,*,*,*&amp;gt;">
            <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
3">{*($T4*)&amp;amp;Storage}</DisplayString>
            <DisplayString>Uninitialized</DisplayString>
            <Expand>
                <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;amp;Storage</Item>
                <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                <Item Name="Storage" Condition="TypeIndex ==
3">*($T4*)&amp;amp;Storage</Item>
                <Item Name="TypeIndex">TypeIndex</Item>
            </Expand>
        </Type>
        <Type Name="TVariant&amp;lt;*,*,*,*,*&amp;gt;">
            <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
3">{*($T4*)&amp;amp;Storage}</DisplayString>
            <DisplayString Condition="TypeIndex ==
4">{*($T5*)&amp;amp;Storage}</DisplayString>
            <DisplayString>Uninitialized</DisplayString>
            <Expand>
                <Item Name="Storage" Condition="TypeIndex ==
```

```
0">*($T1*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
3">*($T4*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
4">*($T5*)&amp;amp;Storage</Item>
                    <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>
        <Type Name="TVariant&lt;*,*,*,*,*,*&gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
3">{*($T4*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
4">{*($T5*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
5">{*($T6*)&amp;amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
                <Expand>
                    <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
3">*($T4*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
4">*($T5*)&amp;amp;Storage</Item>
                    <Item Name="Storage" Condition="TypeIndex ==
5">*($T6*)&amp;amp;Storage</Item>
                    <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>
        <Type Name="TVariant&lt;*,*,*,*,*,*,*&gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
```

```
1">{*($T2*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
3">{*($T4*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
4">{*($T5*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
5">{*($T6*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
6">{*($T7*)&amp;amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
                <Expand>
                        <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
3">*($T4*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
4">*($T5*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
5">*($T6*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
6">*($T7*)&amp;amp;Storage</Item>
                        <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>
        <Type Name="TVariant&amp;lt;*,*,*,*,*,*,*,*&amp;gt;">
                <DisplayString Condition="TypeIndex ==
0">{*($T1*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
1">{*($T2*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
2">{*($T3*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
3">{*($T4*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
4">{*($T5*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
5">{*($T6*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
6">{*($T7*)&amp;amp;Storage}</DisplayString>
                <DisplayString Condition="TypeIndex ==
```

```
7">{*($T8*)&amp;amp;Storage}</DisplayString>
                <DisplayString>Uninitialized</DisplayString>
                <Expand>
                        <Item Name="Storage" Condition="TypeIndex ==
0">*($T1*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
1">*($T2*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
2">*($T3*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
3">*($T4*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
4">*($T5*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
5">*($T6*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
6">*($T7*)&amp;amp;Storage</Item>
                        <Item Name="Storage" Condition="TypeIndex ==
7">*($T8*)&amp;amp;Storage</Item>
                        <Item Name="TypeIndex">TypeIndex</Item>
                </Expand>
        </Type>

 <!--
 *
 * Chaos Visualizers
 *
 -->

 <!-- 2D TVector vizualizer -->
 <Type Name="Chaos::TVector&amp;lt;int,2&amp;gt;">
   <DisplayString>{{{X}, {Y}}}</DisplayString>
 </Type>
 <Type Name="Chaos::TVector&amp;lt;float,2&amp;gt;">
   <DisplayString>{{{X}, {Y}}}</DisplayString>
 </Type>

 <!-- 3D TVector vizualizer -->
 <Type Name="Chaos::TVector&amp;lt;*,3&amp;gt;">
   <DisplayString>{{{X}, {Y}, {Z}}}</DisplayString>
 </Type>

 <!-- 3D TRotation vizualizer -->
 <Type Name="Chaos::TRotation&amp;lt;*,3&amp;gt;">
   <DisplayString>{{{X}, {Y}, {Z}, {W}}}</DisplayString>
 </Type>
```

```
 <!-- 3D TRigidTransform vizualizer -->
 <Type Name="Chaos::TRigidTransform&lt;*,3&gt;">
  <Expand>
   <Item Name="Rotation">(Chaos::TRotation&lt;$T1,3&gt;&amp;)Rotation</Item>
   <Item Name="Translation">(Chaos::TVector&lt;$T1,3&gt;&amp;)Translation</Item>
   <Item Name="Scale3D">(Chaos::TVector&lt;$T1,3&gt;&amp;)Scale3D</Item>
  </Expand>
 </Type>

 <!-- 3x3 PMatrix visualizer. Just show diagonal elements in preview line. -->
 <Type Name="Chaos::PMatrix&lt;*,3,3&gt;">
  <DisplayString>{{{M[0][0]}, ..., {M[1][1]}, ..., {M[2][2]}}}</DisplayString>
 </Type>

 <!-- Chaos Particle Handle Vizualizer. Displays the elements from the SoA for the particle
represented by the handle. -->
 <Type Name="Chaos::TGeometryParticleHandleImp&lt;*,*,*&gt;">
  <!-- non-transient handle -->
  <DisplayString Condition="$T3 == 1">&lt;{HandleIdx}&gt; @
{((Chaos::TVector&lt;$T1,$T2&gt;*)GeometryParticles->MX.AllocatorInstance.Data)[Particl
eIdx]} ({Type})</DisplayString>
  <!-- transient handle -->
  <DisplayString Condition="$T3 ==
0">&lt;T{((Chaos::TGeometryParticleHandleImp&lt;$T1,$T2,1&gt;**)GeometryParticles->
MGeometryParticleHandle.AllocatorInstance.Data)[ParticleIdx]->HandleIdx}&gt; @
{((Chaos::TVector&lt;$T1,$T2&gt;*)GeometryParticles->MX.AllocatorInstance.Data)[Particl
eIdx]} ({Type})</DisplayString>

  <Expand>
   <Item Condition="Type &gt;= 0" Name="Type">Type</Item>
   <Item Condition="Type &gt;= 0 &amp;&amp; $T3==1"
Name="HandleIdx">HandleIdx</Item>
   <Item Condition="Type &gt;= 0"
Name="UniqueIdx">((int32*)GeometryParticles->MUniqueIdx.AllocatorInstance.Data)[Parti
cleIdx]</Item>
   <Item Condition="Type &gt;= 0"
Name="ParticleID">((int32*)GeometryParticles->MParticleIDs.AllocatorInstance.Data)[Parti
cleIdx]</Item>
   <Item Condition="Type &gt;= 0" Optional="true"
Name="DebugName">((FName*)GeometryParticles->MDebugName.AllocatorInstance.Data)
[ParticleIdx]</Item>
   <Item Condition="Type &gt;= 0"
Name="X">((Chaos::TVector&lt;$T1,$T2&gt;*)GeometryParticles->MX.AllocatorInstance.D
ata)[ParticleIdx]</Item>
   <Item Condition="Type &gt;= 0"
```

Name="R">((Chaos::TRotation<$T1,$T2>*)GeometryParticles->MR.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 0" Name="Geometry">((Chaos::TSerializablePtr<Chaos::FImplicitObject>*)GeometryParticles->MGeometry.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 0" Name="DynamicGeometry">((Chaos::TSerializablePtr<Chaos::FImplicitObject>*)GeometryParticles->MDynamicGeometry.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 0" Name="GeometryParticle">((Chaos::TGeometryParticle<$T1,$T2>**)GeometryParticles->MGeometryParticle.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 0" Name="ShapesArray">((Chaos::FShapesArray*)GeometryParticles->MShapesArray.AllocatorInstance.Data)[ParticleIdx]</Item>

<Item Condition="Type >= 1" Name="V">((Chaos::TVector<$T1,$T2>*)KinematicGeometryParticles->MV.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 1" Name="W">((Chaos::TVector<$T1,$T2>*)KinematicGeometryParticles->MW.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 1" Name="KinematicTarget">((Chaos::TKinematicTarget<$T1,$T2>*)KinematicGeometryParticles->KinematicTargets.AllocatorInstance.Data)[ParticleIdx]</Item>

<Item Condition="Type >= 2" Name="VSmooth">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MVSmooth.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="WSmooth">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MWSmooth.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="F">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MF.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="T">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MT.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="LinearImpulse">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MLinearImpulse.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="AngularImpulse">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MAngularImpulse.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="M">((float*)PBDRigidParticles->MM.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2"

Name="InvM">((float*)PBDRigidParticles->MInvM.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="I">((Chaos::PMatrix<$T1,$T2,3>*)PBDRigidParticles->MI.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="InvI">((Chaos::PMatrix<$T1,$T2,3>*)PBDRigidParticles->MInvI.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="CenterOfMass">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MCenterOfMass.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="RotationOfMass">((Chaos::TRotation<$T1,$T2>*)PBDRigidParticles->MRotationOfMass.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="CollisionParticles">((TUniquePtr<Chaos::TBVHParticles<$T1,$T2>,TDefaultDelete<Chaos::TBVHParticles<$T1,$T2> > >*)PBDRigidParticles->MCollisionParticles.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="CollisionGroup">((int*)PBDRigidParticles->MCollisionGroup.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="CollisionConstraintFlags">((int*)PBDRigidParticles->MCollisionConstraintFlags.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="Island">((int*)PBDRigidParticles->MIsland.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="Disabled">((bool*)PBDRigidParticles->MDisabled.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="ToBeRemovedOnFracture">((bool*)PBDRigidParticles->MToBeRemovedOnFracture.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="ObjectState">((enum Chaos::EObjectStateType*)PBDRigidParticles->MObjectState.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="P">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MP.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="Q">((Chaos::TRotation<$T1,$T2>*)PBDRigidParticles->MQ.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="PreV">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MPreV.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2"

Name="PreW">((Chaos::TVector<$T1,$T2>*)PBDRigidParticles->MPreW.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="GravityEnabled">((bool*)PBDRigidParticles->MGravityEnabled.AllocatorInstance.Data)[ParticleIdx]</Item>
<Item Condition="Type >= 2" Name="CCDEnabled">((bool*)PBDRigidParticles->bCCDEnabled.AllocatorInstance.Data)[ParticleIdx]</Item>
</Expand>
</Type>

<!-- Chaos Generic Particle Handle Vizualizer. Displays as a particle handle. -->
<Type Name="Chaos::TGenericParticleHandle<*,*>">
<DisplayString><{Imp.Handle->HandleIdx}> @ {((Chaos::TVector<$T1,$T2>*)Imp.Handle->GeometryParticles->MX.AllocatorInstance.Data)[Imp.Handle->ParticleIdx]} ({Imp.Handle->Type})</DisplayString>
<Expand>
<ExpandedItem>Imp.Handle</ExpandedItem>
</Expand>
</Type>

<!-- Chaos Kinematic Target Vizualizer. -->
<Type Name="Chaos::TKinematicTarget<*,*>">
<DisplayString>{Mode}</DisplayString>
<Expand>
<Item Condition="Mode > 0" Name="Position">Target.Translation</Item>
<Item Condition="Mode > 0" Name="Rotation">Target.Rotation</Item>
</Expand>
</Type>

<!-- Chaos Dense matrix visualizer. -->
<Type Name="Chaos::TDenseMatrix<*>">
<Expand>
<ArrayItems>
<Rank>2</Rank>
<Size>$i == 0 ? NRows : NCols</Size>
<ValuePointer>M</ValuePointer>
</ArrayItems>
</Expand>
</Type>

<!--
Particles SoA vizualizer. This works by displaying the persistent handle for the particle which uses the above vizualizer.
We are lucky we can do this - I don't think there's an easy way to vizualize SoAs in object order in natvis otherwise.

```
  -->
  <Type Name="Chaos::TGeometryParticlesImp&lt;*,*,*&gt;">
    <DisplayString Condition="MSize &lt;= 0">Empty</DisplayString>
    <DisplayString Condition="MSize &gt; 0">Size={MSize}</DisplayString>
    <Expand>
      <IndexListItems Condition="MSize &gt; 0">
        <Size>MSize</Size>
        <ValueNode>

((Chaos::TGeometryParticleHandleImp&lt;$T1,$T2,1&gt;**)MGeometryParticleHandle.Allo
catorInstance.Data)[$i]
        </ValueNode>
      </IndexListItems>
    </Expand>
  </Type>

  <!-- Chaos Joint Handle Vizualizer. -->
  <Type Name="Chaos::FPBDJointConstraintHandle">
    <DisplayString>{ConstraintIndex}:
{(((Chaos::FPBDJointState*)ConstraintContainer->ConstraintStates.AllocatorInstance.Data)[
ConstraintIndex]).Level}</DisplayString>
    <Expand>
      <Item Name="ConstraintIndex">ConstraintIndex</Item>
      <Item
Name="ConstraintParticles">((Chaos::FPBDJointConstraints::FParticlePair*)ConstraintConta
iner->ConstraintParticles.AllocatorInstance.Data)[ConstraintIndex]</Item>
      <Item
Name="ConstraintSettings">((Chaos::FPBDJointSettings*)ConstraintContainer->ConstraintS
ettings.AllocatorInstance.Data)[ConstraintIndex]</Item>
      <Item
Name="ConstraintState">((Chaos::FPBDJointState*)ConstraintContainer->ConstraintStates.A
llocatorInstance.Data)[ConstraintIndex]</Item>
    </Expand>
  </Type>

  <!-- Chaos Collision Handle Vizualizer. -->
  <Type Name="Chaos::FPBDCollisionConstraintHandle">
    <DisplayString>{ConstraintIndex}</DisplayString>
    <Expand>
      <Item Name="ConstraintIndex">ConstraintIndex</Item>
      <Item
Name="Constraint">((Chaos::FRigidBodyPointContactConstraint*)ConstraintContainer->Con
straints.SinglePointConstraints.AllocatorInstance.Data)[ConstraintIndex]</Item>
    </Expand>
  </Type>
```

```
  <!-- Chaos Convex StructureData Vizualizer -->
  <Type Name="Chaos::FConvexStructureData">
    <Expand>
      <Item Name="IndexType">IndexType</Item>
      <Item Condition="IndexType == 1" Name="Data">Data.DataS</Item>
      <Item Condition="IndexType == 2" Name="Data">Data.DataM</Item>
      <Item Condition="IndexType == 3" Name="Data">Data.DataL</Item>
    </Expand>
  </Type>

  <!-- FIoChunkId visualizer -->
  <Type Name="FIoChunkId">
    <!-- Id is 12 bytes long and the object is considered Invalid if all values are 0 -->
    <DisplayString Condition="reinterpret_cast&lt;uint32*&gt;(Id)[0] == 0 &amp;&amp;
reinterpret_cast&lt;uint32*&gt;(Id)[1] == 0 &amp;&amp;
reinterpret_cast&lt;uint32*&gt;(Id)[2] == 0">Invalid</DisplayString>
    <Expand>
      <Item Name="ChunkId">*(uint64*)&amp;Id[0]</Item>
      <Item Name="ChunkIndex">*(uint16*)&amp;Id[8]</Item>
      <Item Name="EIoChunkType">*(EIoChunkType*)&amp;Id[11]</Item>
    </Expand>
  </Type>
  <!--
  * Niagara Visualizers
  *
  -->
  <!-- FNiagaraVariable visualizer -->
  <Type Name="FNiagaraVariableBase">
    <DisplayString>Name={Name},
Type={((FNiagaraTypeDefinition*)(FNiagaraTypeRegistry::RegisteredTypes.AllocatorInstanc
e.InlineData))[TypeDefHandle.RegisteredTypeIndex].ClassStructOrEnum->NamePrivate}</D
isplayString>
  </Type>

  <Type Name="UNiagaraNodeFunctionCall">
    <DisplayString>Name={NamePrivate},
FunctionName={FunctionDisplayName}</DisplayString>
  </Type>

</AutoVisualizer>
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу на кваліфікаційну роботу бакалавра
на тему:
"Розробка програмного забезпечення ігрової локації на базі середовища Unreal Engine"
студента групи 121-20ск-1 Сеня Данила Дмитровича

Керівник економічного розділу
доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В
ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Сень.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Сень.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Сень.rar	Архів містить коди і і скомпільовану програму
Презентація	
Сень.pptx	Презентація кваліфікаційної роботи