МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІЗУАЛІЗАЦІЇ ЕЛЕМЕНТІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ В СИСТЕМАХ ДИЗАЙНУ ІНТЕР'ЄРІВ

Здобувач вищої освіти гр. ІКм.-11 Богдан ТРУСОВ

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

В.о. завідувача кафедри
кандидат технічних наук, доцент
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«122 – Комп'ютерні науки»*

Затверджую:
в.о.зав.кафедрою_____
_____
"_____" _____ 2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Трусову Богдану Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія візуалізації елементів доповненої реальності в системах дизайну інтер'єрів*

затверджую наказом по інституту від "____" _____ 2022 р. № _____

2. Термін здачі студентом закінченого проекту (роботи)
_____
3. Вхідні данні до проекту (роботи)
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Огляд актуальної літеартури та додатків, що використовують технології доповненої реальності. 2) Порівняння основних методів розробки програмного забезпечення з використання доповненої реальності. 3)Розробка прототипу програмного забезпечення для сканування навколишнього простору та візулізації 3D моделей доповненої реальності.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз літератури та готових рішень		
2.	Постановка завдання		
3.	Вибір методик за засобів розробки		
4.	Програмна реалізація		
5.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 57 стор., 49 рис., 1 таблиця, 2 додатки, 11 літературних джерел.

Об’єкт дослідження – інформаційна технологія візуалізаціх об’єктів доповненої реальності в системах дизайну інтер’єрів

Мета роботи – розробка додатку на OS Android, Win з використанням Unreal Engine, Blueprint, ARCore для демонстрації засобу візуалізації.

Результати – проведений аналіз літератури та ринку актуальних додатків, які використовують технології доповненої реальності. Порівняно такі засоби розробки як ядро додатків доповненої реальності - ARCore та ARKit, а також рушії що можуть використовувати – Unreal Engine та Unity 3D. На основі проведеного аналізу було виявлено найбільш популярні функції додатків доповненої реальності, а саме візуалізація та взаємодія з віртуальним об’єктом. Також було обрано таргетну платформу Android. Реалізація проводилась з використанням Unreal Engine та Blueprint, що може бути нативізований у C++ код.

ДОДАТОК ANDROID, ВІЗУАЛІЗАЦІЯ ВІРТУАЛЬНОГО ОБ’ЄКТУ, ТЕХНОЛОГІЯ ДОПОВНЕНОЇ РЕАЛЬНОСТІ, UNREAL ENGINE, BLUEPRINT, 3D МОДЕЛЬ.

ЗМІСТ

ВСТУП

У сучасному світі технології розвиваються дуже швидко. Кінцевого користувача вже не здивувати корисними онлайн додатками, потоковими сервісами та подібними рішеннями, які ґрунтуються за застарілих сценаріях використання. Як наслідок було створено цілу течію рішень, що базуються на технології як доповненої так і віртуальної реальності.

Доповнена реальність є частиною змішаної реальності, тобто це будь-який віртуальний елемент, який тим чи іншим чином здатен доповнити повсякденне життя.

Зародки подібних технологій з’явилися ще у 1900 роках з винаходом стереоскопу. Цей прилад використовував функціональні можливості мозку і накладав дві різні статичні картинки на кожне око окремо, що і створювало ефект “тривимірності”, або ж “ефект присутності”. Звичайно дана технологія не йде ні в яке порівняння з сучасними аналогами, проте це відкрило простір досліджень для вчених минулого.

Першим задокументованим пристроєм пристроєм що використовував віртуальну реальність був так званий “Блакитний ящик”, або ж можелюючий політ, винайдений Едвіном Лінком. Суть була в імітації справжнього польоту, без необхідної наявності самого літака.Подібний винахід допоміг натренувати більш ніж 500 тисяч пілотів під час другої світової війни. Тобто цей випадок можна назвати першим використанням технології віртуальної реальності в освітній галузі.

Наступним великим стрибком була розробка першого реального HMD у 1970-х роках. Цей пристрій міг зчитувати рухи користувача, та візуалізувати досить прості моделі в режимі wireframe. Головним же мінусом подібного девайсу була непідйомна вага, і як наслідок його необхідно було підтримувати спеціальним тримачем біля стелі.

Як альтернативу віртуальній реальності, яка повністю змінювала світогляд, вчені висунули гіпотезу про створення доповненої реальності.

Дослідник Рональд Азум висунув гіпотезу, з чого повинна складатись подібна система:

- поєднання реального і віртуального;
- взаємодія в реальному часі;
- повноцінна тривимірність.

Слідуючи цим принципам, девайс, за допомогою якого можна було б взаємодіяти з доповненою реальністю мав би бути занадто дорогим і складним у розробці, тому технологія доповненої реальності набрала обертів уже в сучасному світі, де кожна людина має в кишені смартфон.

Проте у якій сфері можна використати подібну технологію? Прикладів цьому досить багато. По перше це сфера маркетингу. Дедалі більше розробників прагнуть створити тривимірну модель своєї продукції, для кращих продаж та розвитку бізнесу. Крім того доповнену реальність використовують у розробці мобільних ігор, а також у девайсах що націлені на полегшенні повсякденності (Google glass). У даній роботі будуть розглянуті основні засоби та методи для створення демонстраційного додатку візуалізації доповненої реальності.

1. АНАЛІЗ НАЯВНИХ ДОДАТКІВ. ПОСТАНОВКА ЗАДАЧІ

1.1 Порівняння доповненої та віртуальної реальності

Поняття віртуальної та доповненої реальності досить близькі за походженням проте мають досить багато відмінностей як у сприйнятті інформації(input), так і у відображенні її кінцевому користувачу.

Першою як течія почала розвиватись віртуальна реальність, як засіб що замінює собою навколишнє середовище. Це давало безліч переваг, проте досить сильно обмежується технологіями. Один із невід'ємних елементів віртуальної реальності є окуляри, які окрім відображення віртуальних об'єктів та звуків мають датчики керування, та положення подібного девайсу у просторі. Крім того подібний пристрій має бути приєднаним до того ж ПК чи консолі. Але навіть таке розгалуження елементів на різні пристрої не дає повноцінного комфорту: VR окуляри досить важкі, та і використання їх у повсякденних задачах майже неможливе.

Саме на основі цих можливостей та недоліків почала свій розвиток доповнена реальність. Пристрої для подібних технологій набагато менші, та практичніші за VR шоломи. Наприклад той же смартфон чи Google glass або Air glass. Крім того доповнена реальність орієнтована на те щоб покращити вже існуючу повсякденність, а не замінити її повністю. Тобто це дещо різна за призначенням технологія [2].

Розглянемо основні відмінності у технічному плані AR та VR додатків. У віртуальній реальності перш за все використовується раніше створений віртуальний простір, об'єкти взаємодії, та актори, якими здатен керувати користувач. Подібні сценарії досить ресурсозатратні для системи, проте зникає необхідність відеосканування навколишнього простору. Проте той же акселерометр та гіроскоп мають бути наявними у шоломі, задля можливості керування віртуальною камерою. Крім цього необхідні інші маніпулятори, такі як джойстики, рукавички, керма і таке інше.

Рисунок 1.1 – Приклад шолому VR із маніпуляторами

На відміну від віртуальної, доповнена реальність не потребує досить складних та незручних гаджетів, проте і має іншу менту. Найчастіше достатньо смартфону та гарно налаштованого додатку.

1.2 Сфери застосування технологій доповненої реальності

У сучасному світі ринок бізнесу не стоїть на місці. Кожен власник намагається просунути свій продукт “в топ” для отримання все більших доходів. Найбільшого ривку останнім часом зазнала технологія доповненої реальності. Найбільшою перевагою даної технології є візуалізація того, чого ще немає. Більшість компаній, які займаються розробкою інтер’єрів, чи продажу мебельних продуктів стикається з пошуком команди 3D художників для візуалізації продукту. Це необхідно щоб показати що саме в результаті отримає покупець на виході.

Рисунок 1.2 – Приклад візуалізації інтер’єру

Проте з розвитком технологій як віртуальної так і доповненої реальності необхідність у цьому цілковито зникне. Адже користувач зможе відкрити онлайн магазин, завантажити її у свій смартфон, та віртуально прикинути як зміниться його інтер’єр. Подібну технологію вже використовує такий гігант як IKEA[5].

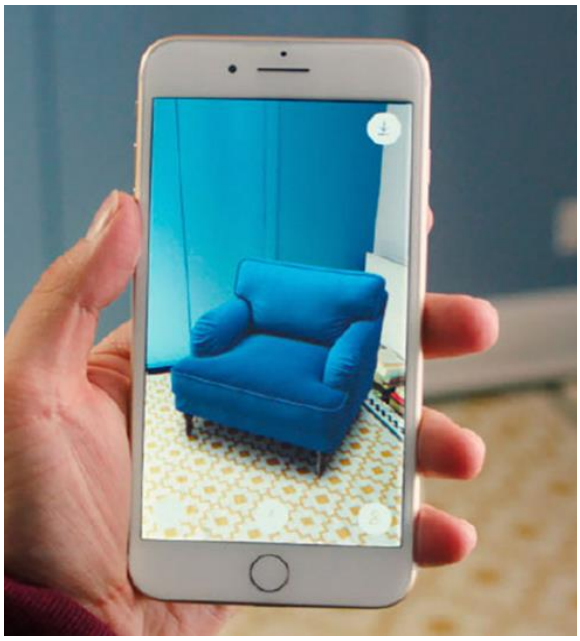


Рисунок 1.3 – AR мітка IKEA

Крім того, зі збільшення обчислювальних можливостей з’явиться можливість візуалізувати безліч об’єктів водночас. Крім візуалізації віртуальних об’єктів ще однією перевагою доповненої реальності є сканування та аналіз навколишнього світу. Подібний функціонал можна використати як для розваг (маски в соціальних мережах) так і для розробки серйозного софту. Перший та більш простий це сканування навколишнього світу для отримання реальних габаритів об’єкту. Подібні додатки досить зручні, адже результати вимірювання автоматично збираються у звіт. Крім того, деякі сучасні смартфони оснащені додатковим сканером, за допомогою якого можна перетворювати відсканована реальність і повноцінний 3D об’єкт. Так, він не буде ідеальним та матиме безліч недоліків, проте за умови якісного освітлення можна отримати чудову чорнову геометрію з високоякісними текстурами, які можна використовувати після ретопології, або як матеріал для інших об’єктів. Мінусом подібної технології є те, що вона майже не працює на смартфонах із Android OS.

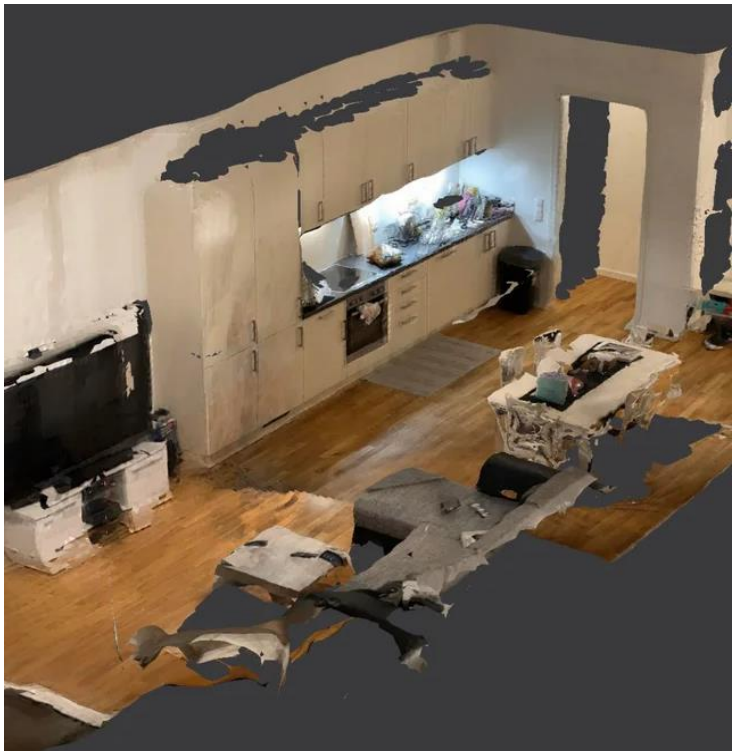


Рисунок 1.4 – Результат сканування кімнати за допомогою смартфону

1.3 Аналіз сучасних AR-додатків та виділення основних механік та можливостей

Ринок додатків для сучасних смартфонів зростає неймовірно швидко. Це дає змогу зрозуміти, чим приваблива подібна технологія, та які можливості вона надає. Перше що може прийти на думку, пов’язане з технологіями доповненої реальності це маски в тому ж Instagram чи фейсбук. Головною їх “фішкою” є те, що вони повторюють рухи обличчя користувача в реальному часі. Ця можливість з’являється завдяки таким функціям як:

- розпізнавання обличчя в реальному часі;
- захоплення типово-можливих рухів.

Тобто конкретно у цьому випадку AR функції будуються на можливостях фотозйомки та спрощеної технології motion capture. Перш за все створюється 3d модель маски, далі налаштовується базовий матеріал, за необхідності додається ефект. Наступним кроком є риггинг вже створеної моделі, тобто створення скелету на основі опорних точок, та налаштування поверхні при її деформації. Фактично, художник власноруч створює можливість переходу моделі із одного стану в інший. Наступним етапом іде прив’язка створених маркерів 3d моделі до тих, що змогла знайти система розпізнавання [3].

Наступні додатки можна згрупувати в такі категорії як перегляд раніше створених моделей у віртуальній сцені з використанням гіроскопу, та візуалізація 3d моделі поверх відеоряду. Як приклад можна розглянути веб додаток model editor. Він дає змогу завантажити модель у форматі GLTF та роздивитись її у реальному розмірі.

Рисунок 1.5 – Приклад візуалізації об’єкту за допомогою model editor

ARLOOPA – це ще один додаток, схожий за функціоналом, проте використовує дещо інші технології. Він здатен сканувати навколишнє середовище, і прив’язувати відскановані 2d зображення до раніше створених тривимірних. Подібні технології можна використати як для розваги так і в сфері маркетингу.

Рисунок 1.6 – Приклад роботи додатку ARLOOPA

Крім цих прикладів слід також згадати додатки типу AR Plan, що дає можливість за допомогою камери телефону зняти розміри елементів навколишнього середовища, та зручно зберегти отримані дані в звіт. Отже, з вище проведеного аналізу розуміємо що найпоширенішими можливостями додатків доповненої реальності є:

- сканування та трекінг оточення
- відображення створених 3д об’єктів
- шейдинг створеного 3д об’єкту
- відображення UX\UI поверх відеопотоку

На основі цих функцій можна перейти до постановки задачі.

1.4 Постановка задачі

Необхідно створити прототип додатку доповненої реальності, який матиме наступний функціонал:

1. Сканування 3D простору і знаходження вільної площини у просторі.
2. Відображення та шейдинг віртуальної моделі, згідно відсканованому простору.
3. Поворот та масштабування віртуальної моделі.

2. МЕТОДИ ТА РІШЕННЯ ПРИ СТВОРЕННІ ДОДАТКІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1 Вибір рушію для візуалізації

Основним набором інструментів, які необхідні для створення додатку є 3D рушії. Їх існує безмежна кількість, проте вибір буде здійснюватись із вбудованою підтримкою основних бібліотек для додатків доповненої реальності. Розглянемо два найпопулярніших рушії, які підтримують API ARCore та ARKit.

Unreal engine – ігровий рушій, створений компанією Epic Games. Дана програма являє собою необхідний набір інструментів для створення усіх аспектів програми, незалежно від того чи це повноцінна гра на пк чи додаток для смартфону. Перша гра, що була розроблена на цьому рушії мала спільну назву, та відносилась до жанру шутерів, що і породило міф про те що unreal вузькоспрямований рушій. Подібні речі були спростовані після опублікування документації, та надання доступу до технологій Epic Games. Крім того з виходом 5 версії рушію у 2022 році, великий відсоток інструментарію був оновлений, а також додані проривні технології: Lumen, Nanite [1].

Lumen Global Illumination - це варіант роботи зі світлом, який дещо схожий на трасування променів. Проте основну задачу, яку він вирішує це дифузне відбиття променів від освітленої поверхні. Крім того, Lumen здатен забезпечити нескінченні дифузні відбиття, в результаті чого колір елементів фінального рендеру набуває максимальної реалістичності.

Nanite Virtualized Geometry – технологія динамічного відображення геометрії за допомогою полігональних кластерів. При завантаженні моделі в рушій та підготовці її до використання відбувається аналіз поверхні, та розбиття її на блоки. При візуалізації даного об’єкту рушій сам вираховує необхідну кількість кластерів, яка необхідна моделі. Подібна технологія витісняє використання рукотворних LOD-моделей, що позитивно позначається як на швидкості розробки продукту, так і на його оптимізації.[4]

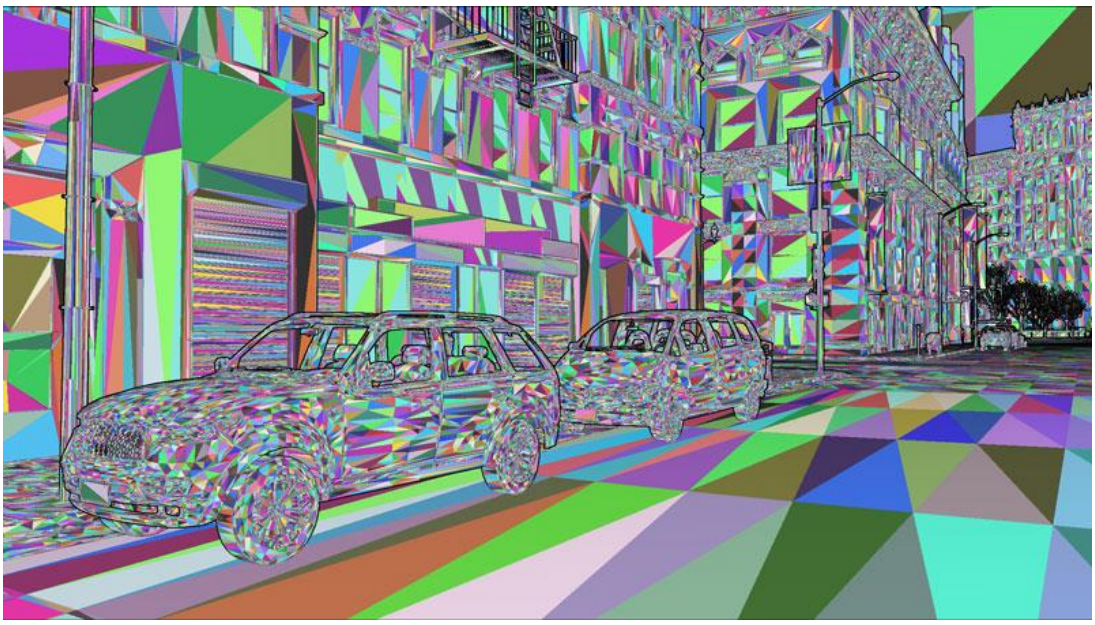


Рисунок 2.1 – Nanite

Стосовно конкурентів: чим саме цей рушій виділяється від Unity. А відмінності досить суттєві. Перш за все у кожного рушія свій список платформ, під які розробники можуть створювати свій продукт. Unreal може похизуватися такими: PC, PS4, Xbox, Android, SteamOS, iOS та Linux. Наступною перевагою даного рушію є використання технології BluePrint - програмування за допомогою нод. По суті, кожна нода має свій аналог у класичному програмуванні, наприклад той же умовний оператор, цикл, колекція і таке інше. Але при компіляції проекту всередині рушія відбувається нативізація програмного коду з віртуального середовища. Таким чином розробник працює з блюпринтом, що прискорює розробку та мінімізує шанс отримання помилки, і при цьому є можливість власноруч створювати код. Достатньо при створенні проекту вказати засіб розробки. Крім того ці два підходи можна комбінувати між собою: проект розробляється за допомогою блупринтов, а деякі специфічні функції чи класи написані за допомогою C++ та імплементовані у вигляді ноди. Приклад візуального програмування ви можете спостерігати нижче [7]:

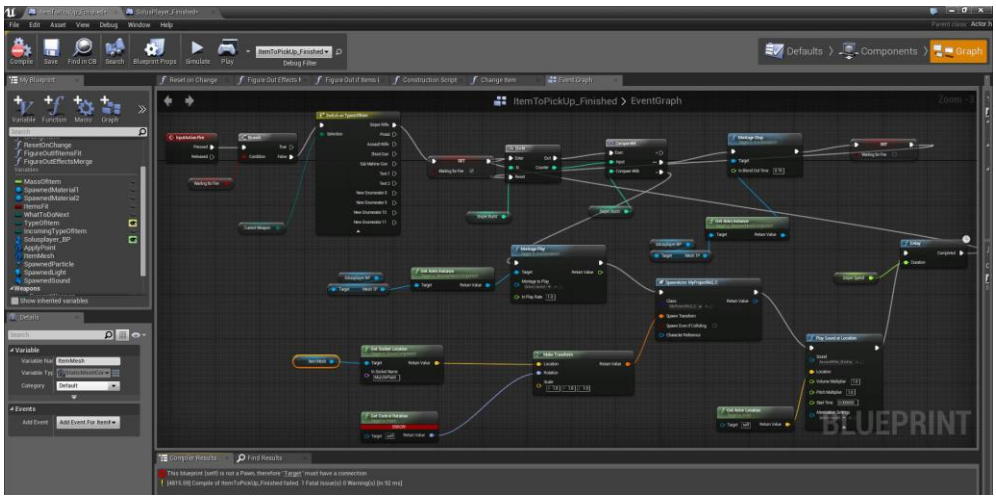


Рисунок 2.2 – Приклад використання технології BluePrint

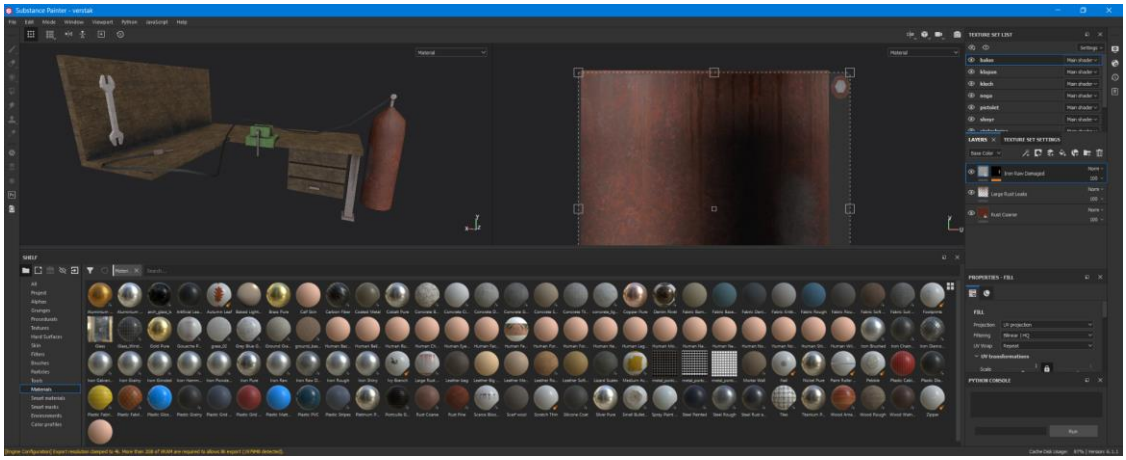


Рисунок 2.3 – Приклад текстурування у програмі Substance Painter

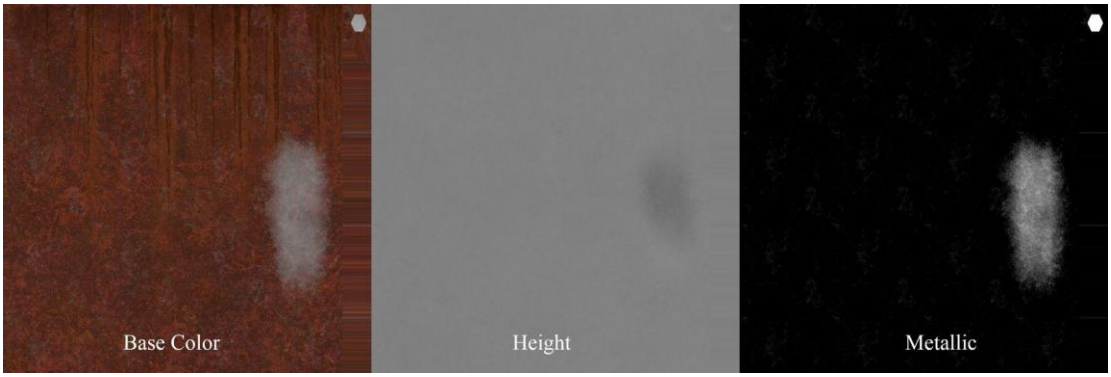


Рисунок 2.4 – PBR текстури

Вище описані інструменти є основою майже всіх пайплайнів створення тривимірної графіки. Ще однією перевагою даного рушію на відміну від конкурентів є досить гнучкий та потужний редактор матеріалів. Він підтримує

основні пайплайни роботи з 3D гарфікою та текстурами стандарту PBR, а також потребує використання Dx Normal. Крім того, асети потребують внутрішнього налаштвання в рушії лише раз. Після збереження їх можна завантажувати в пару кліків.

Рисунок 2.5 – Редактор матеріалів Unreal Engine 4

Як альтернативу, можна використовувати glTF Runtime для імпорту спеціальних AR моделей у форматі GLTF. Це суцільно прискорить інтеграцію моделі, адже подібні моделі уже містять в собі налаштовані текстури та матеріали за пайплайном metallic-roughness.

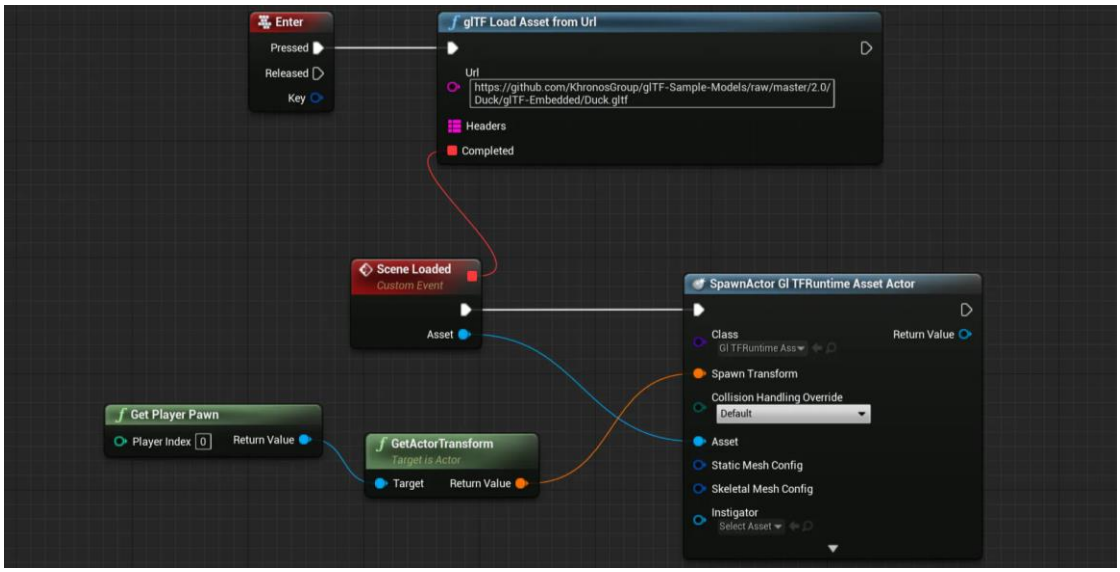


Рисунок 2.6 – Інтеграція готової AR моделі

Також слід згадати те, що вихідний код рушію відкритий, і в розробників є можливість змінювати уже готові частини рушію, а також з легкістю додавати свої. Для прикладу, було створено blueprint, що дозволяє використовувати C# замість C++. Із мінусів даного рушію є те що він потребує 4.2gb оперативної пам'яті при розрахунках освітлення та взагалі не підтримує 32-х розрядні операційні системи. Це вносить технічні обмеження як для розробників(необхідне сучасне обладнання для розробки) також для юзерів і із за цього обмеження кількість платформ для розробки різко зменшується [7].

Іншим же вибором є рушій Unity3d. Так як обидва розглянуті рушії мають вбудовану підтримку кросплатформенності, то розглянемо інші аспекти подібного софту. Перш за все Unity набагато легше свого конкурента, як в плані необхідної фізичної пам'яті ПК, так і в плані ресурсів для розробки. Для комфортної роботи йому необхідно лише до 10 гігабайтів вільного місця. В плані візуалізації Unity використовує бібліотеки DirectX 11 що робить неможливим використання технологій трасування променів. Динамічного розбиття моделей також немає, тож необхідно використовувати старий метод ручної оптимізації графіки. Подібні речі значно збільшують термін виробництва фінального продукту. Слід також зазначити чутливість цього рушію до використання та менеджменту пам'яті. При великому проектному навантаженні можуться з'явитись недоліки у вигляді статерів, тож рушій типу Unity краще використовувати у розробці легких та невибагливих додатків [11].

Налаштування для створення додатків доповненої реальності у обох варіантах дещо схоже. Перш за все необхідно налаштувати Android SDK та NDK а також пакет Java. Наступним кроком потрібне налаштування самого проекту в рушію, його збірку на ту чи іншу платформу. Слід пам'ятати що Android та IOS використовують різне ядро для проектів доповненої реальності, проте що Unity що Unreal підтримують обидва ядра розробки. Єдиний мінус Unity це його закритий код, і дещо погана оптимізація. Подібний рушій підходить для простих проектів, тоді як складні та перегружені контентом викликають зависання та в цілому погано використовують ресурси системи.

2.2 Порівняння ARCore та ARKit

Існує безліч SDK для створення додатків доповненої реальності проте порівняємо найбільші, які були створені компаніями Google та Apple. Перевагою даних наборів інструментів є те що вони:

- нативні для своєї цільової операційної системи
- ARKit чудово імплементуються в систему IOS
- розуміє всі налаштування камери, зчитування рухів та інших речей, що спрощує задачу розробникам.

ARCore починав свій шлях як чистий android sdk, і лише з часом став набором інструментів для розробки додатків доповненої реальності. По функціоналу має деякі відмінності від ARKit. Наприклад має додатковий вузол налаштування шейдерів, що цілковито сказується на фінальній картинці. При використанні даної бібліотеки з потужним рушієм типу Unreal можна отримати картинку дуже високої якості. Крім того ARCore може аналізувати освітлення у відсканованому просторі, і налаштувати освітлення віртуального об’єкту під сцену. Дана бібліотека підтримується двома платформами. Нижче наведено таблицю переваг та недоліків даних SDK [8].

Таблиця 2.1 –Порівняння SDK

ARCore		**ARKit**	
переваги	недоліки	переваги	недоліки
Покращені графічні можливості (шейдери)	Гірше захоплення руху відсканованих об’єктів	повна інтеграція в IOS	відсутнє додаткове налаштування шейдерів
Оцінка освітленості та глибини простору	Трохи гірша сумісність із IOS у порівнянні з ARKit	легка робота з usdz форматом за допомогою Reality Composer	підтримується тільки на смартфонах від Apple

Кросплатформеність		Покращене відстеження рухів та mo-cap у порівнянні з версією від google	відсутня зворотня підтримка версій

2.3 Розробка gltf та usdz моделей

Основої будь-якого додатку доповненої реальності є 3D об’єкт. Подібні файли можуть мати безліч форматів, проте основними для AR є GLB(android) та USDZ(IOS). Розробка подібних моделей поділяється на декілька етапів.

Моделювання високо полігональної моделі за допомогою будь-якого зручного 3д редактору. Наприклад 3ds max, blender, maya та інші. Подібна модель може містити більше 5 мільйонів полігонів, та передавати найдрібніші деталі 3д об’єкту. Розгортати модель на UV не потрібно, адже дана версія моделі буде використовуватись суто для запікання карт нормалей та висот.

Рисунок 2.7 – Приклад високо полігональної моделі

Наступним етапом створення AR об’єкту є моделювання низькополігональної версії, для економії ресурсів гаджету. Перш за все, якщо це можливо, прибираються модифікатори згладжування з моделі такі як TurboSmooth, MeshSmooth, Divide, залежно від 3д редактора. Це максимально зменшить час розробки. У іншому випадку можна скористатись інструментом

Retopo у програмі Zbrush і відрегулювати необхідну деталізацію. Проте, як це часто буває, результат отриманий від автоматизованих інструментів не завжди є гарної якості, тому слід перевіряти отриману топологію.

Рисунок 2.8 – Приклад низько полігональної моделі

Наступним етапом у створенні 3д моделі доповненої реальності це запікання карт нормалей і та висот з high на low. Подібний метод використовують майже у всіх пайплайнах розробки 3д додатків. Суть полягає у наступному: обидва варіанти моделі накладаються одна на одну. Із низькополігональної моделі виходять промені, і до тих пір поки не досягнуть високополігонального аналогу. Якщо відмінність нема то у даному місці карта нормалі отримає нейтрально блакитний колір, у іншому ж випадку він зміниться на жовтий чи зелений. Слід також розуміти для якої платформи розроблятиметься дана модель і обрати тип нормалі: normal_gl чи normal_dx. Різниця полягає у інвертованому зеленому каналі для версій під DirectX. Карта висот представляє собою чорно білий градієнт поверхні де темні ділянки позначають рівень нижче лоупольної моделі, а білі навпаки вище.

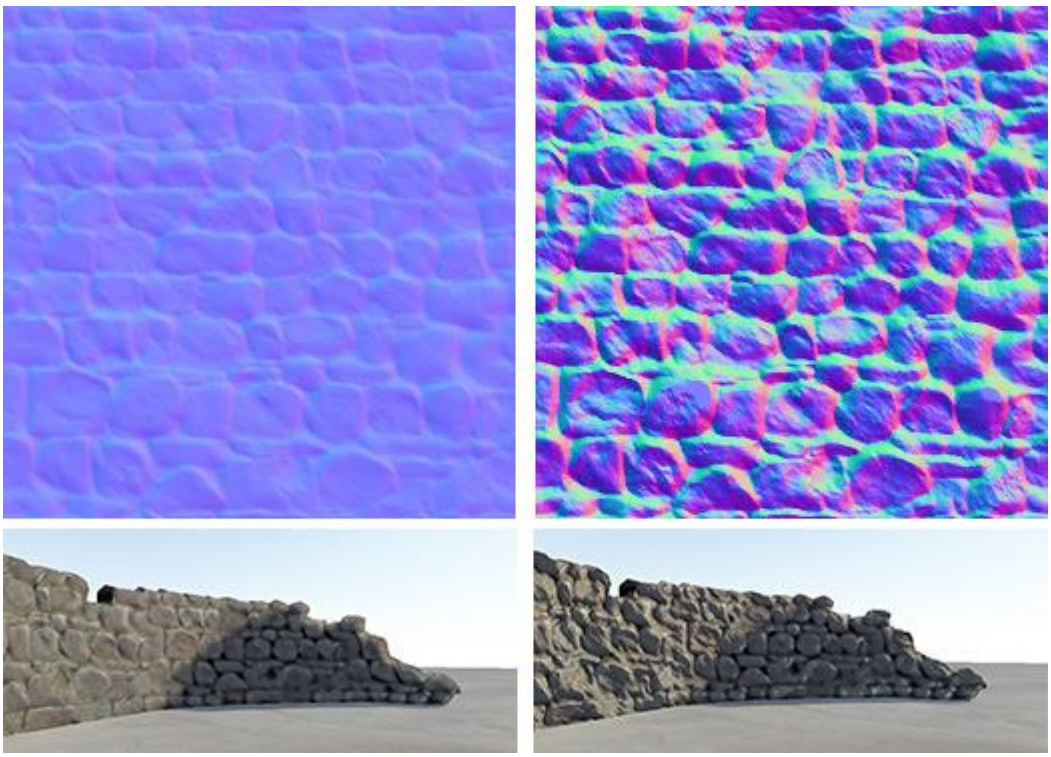


Рисунок 2.9 – Приклад роботи карти висот

Наступний етап збірки моделей може поділятись на два варіанти. Перший: створення glb або usdz моделі з використання стороннього софту. Наприклад утиліта Substance Painter або рушій візуалізації Marmoset Toolbag чи навіть Blender. Суть полягає у налаштуванні матеріалу, як він має відбивати чи заломлювати світло. Створений із металоподібних матеріалів чи діелектриків. У випадку використання цього варіанту, додатково налаштовувати щось безпосередньо у рушії Unreal чи Unity не потрібно.

У випадку ж завантаження компонентів моделі до рушію окремо, можна більш тонко налаштувати матеріали, і відразу побачити їх фінальний вигляд. Крім того цей спосіб є єдиним у випадку якщо програма передбачатиме динамічну зміну матеріалів на об’єкті.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ДЕМОНСТРАЦІЙНОГО ДОДАТКУ

3.1 Створення демонстраційного 3д об’єкту

Перш ніж переходити до роботи у самому рушії необхідно створити базову 3д модель дивану та декілька варіантів кольору до неї. Для цього використаємо 3ds max. Отримуємо результат.

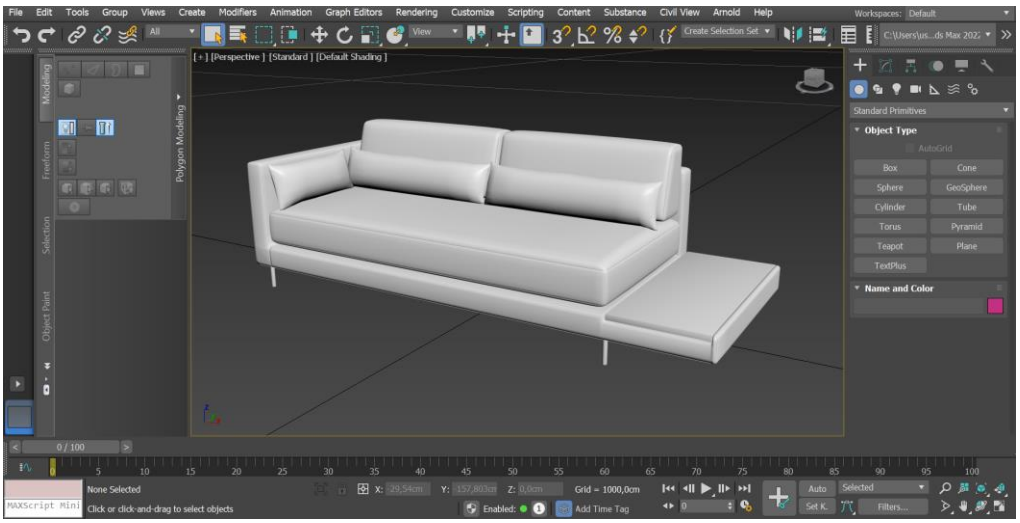


Рисунок 3.1 – створений 3D об’єкт

Наступним кроком необхідно налаштувати UV розгортку для цієї моделі. Для цього використаємо програму Rizom UV.

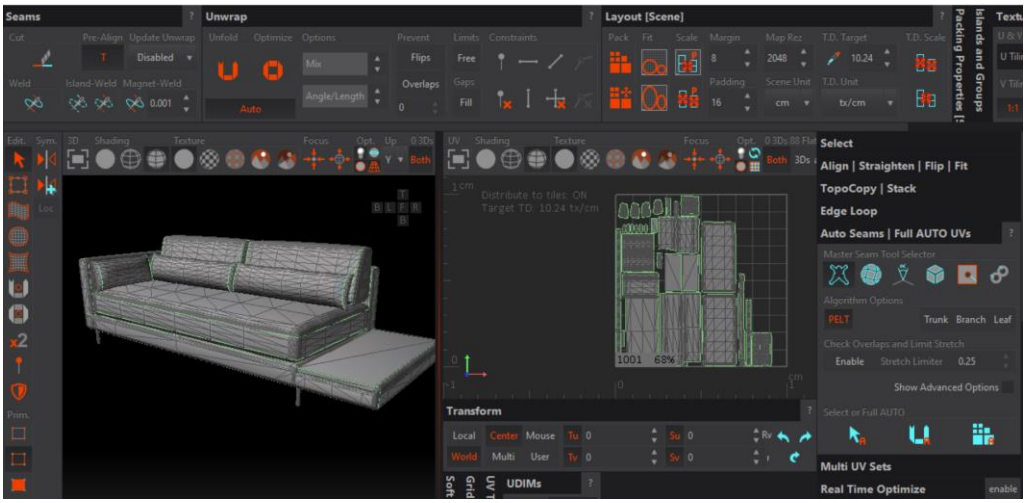


Рисунок 3.2 – розгортка

Після цього необхідно завантажити створену модель у Substance Painter та створити матеріали для кожного об’єкту моделі. Слід зауважити, що збирати модель у GLB непотрібно, адже надалі вона буде завантажена у рушій разом із текстурами окремо.

Рисунок 3.3 – налаштований об`єкт

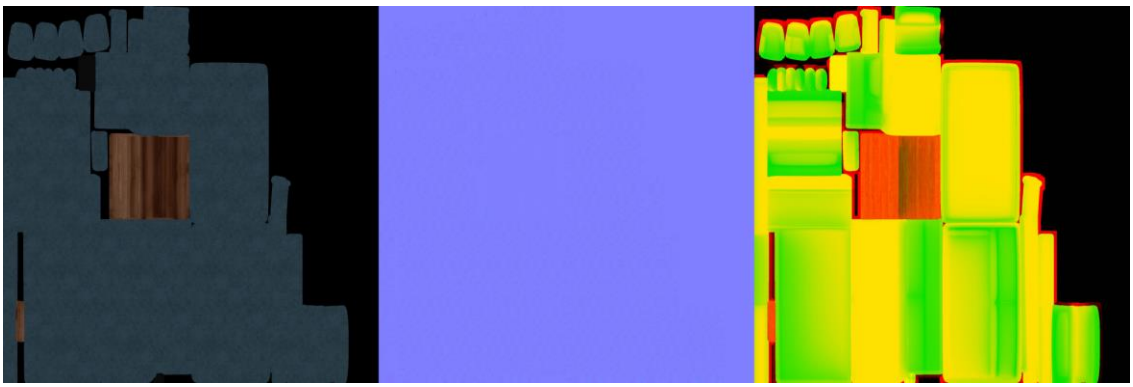


Рисунок 3.4 – отриманий набір текстур

Наступним кроком імпортуємо створений об’єкт у рушій. Для цього достатньо просто перетягти його у Content Browser.

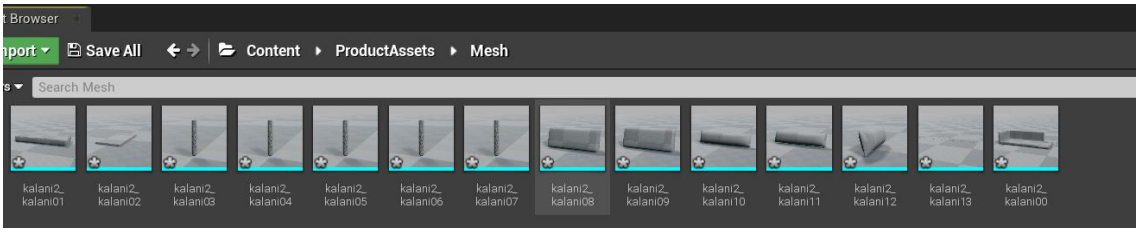


Рисунок 3.5 – браузер об’єктів

При імпорті слід поставити галку на генерацію колізії, а також на створення нового пустого слоту для матеріалу. У даному випадку буде достатньо лише одного, адже ми пекли усі текстури на єдиний UV тайл. Проте бувають випадки коли цього може бути недостатньо. Тоді для кожного підоб'єкту слід створити власний з унікальним ID [1].

Рисунок 3.6 – налаштування при імпорті мешу

Для перевірки можна клікнути на меш, якщо з ним все добре має відкритись вікно пререндеру і відобразити його з базовим матеріалом.

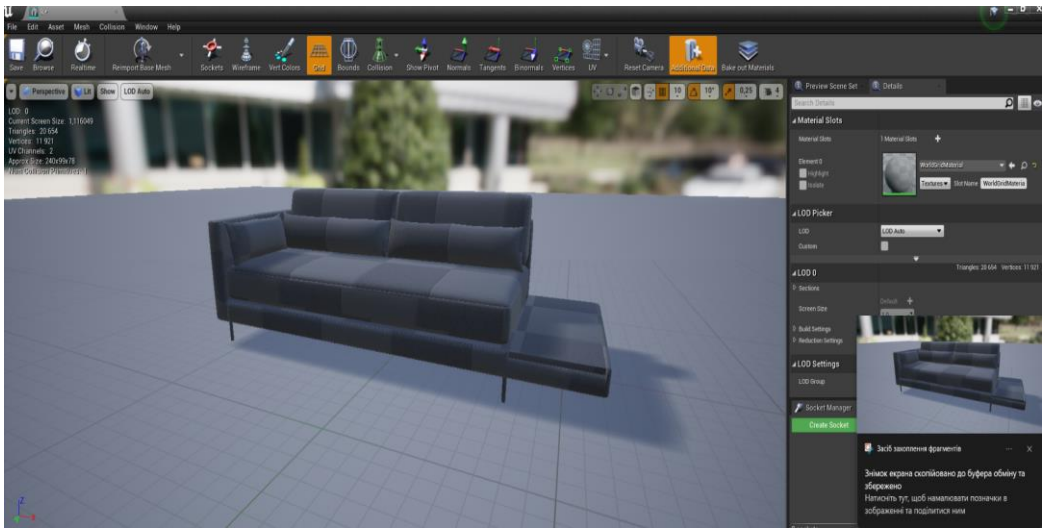


Рисунок 3.7 – меш з базовим матеріалом

Далі необхідно перейти в директорію Content\ProductAsset\Materials та завантажити набір текстур. Після цього знаходимо створений при імпорті матеріал sofa_material та клікаємо на нього. Вантажимо туди необхідні текстури та налаштовуємо матеріал. Зліва є пре-рендер тож відразу можна себе перевірити.

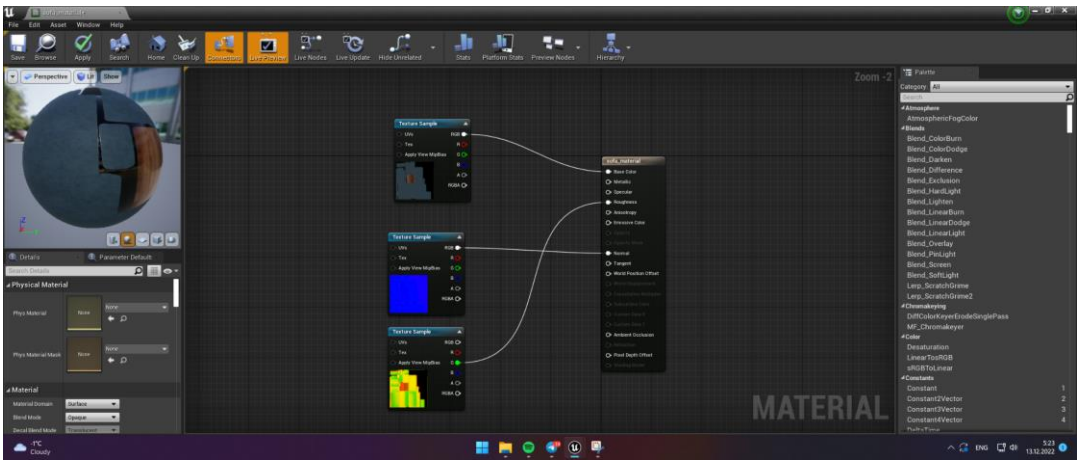


Рисунок 3.8 – налаштування матеріалу

Після подібних маніпуляцій можна відкрити створений ассет і переконатись що все коректно працює.

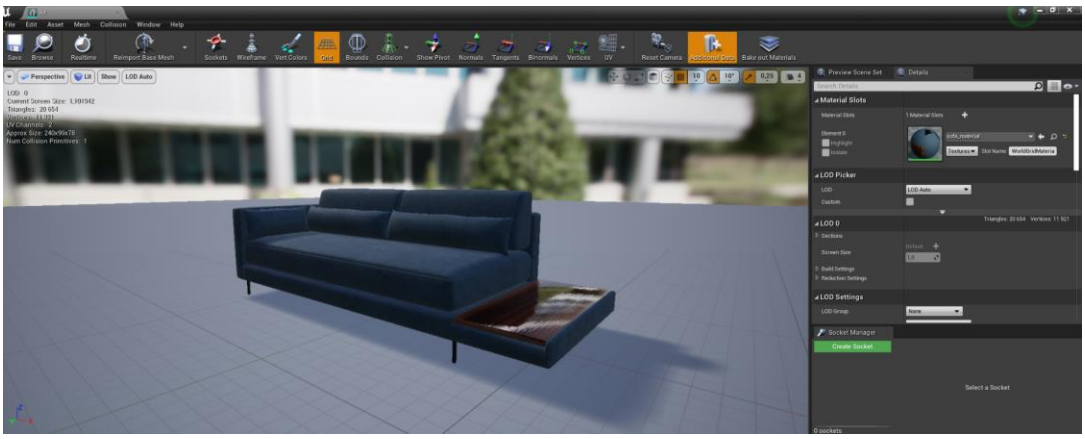


Рисунок 3.9 – налаштований ассет

3.2 Створення логіки додатку

Створюватись логіка буди за допомогою влаштованої в рушій технології BluePrint. Основних класів два : BP_ARPawn та BP_ARPlayer_Controller, а також структура ENUM_GESTURE для переліку дій користувача. Клас BP_ARPlayer досить простий. Він містить дві ноди, які відповідають за зчитування дій та їх припинення.

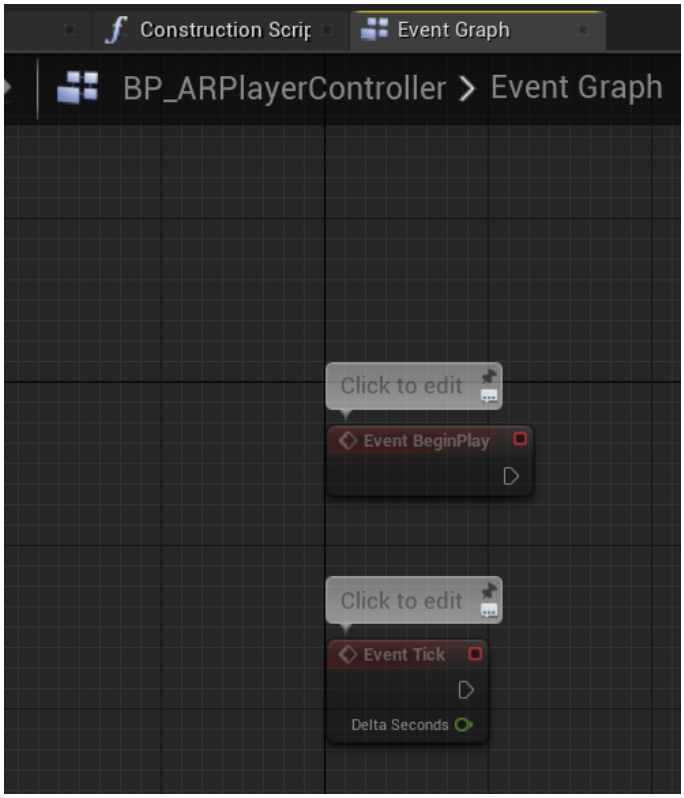


Рисунок 3.10 – клас BP_ARPlayer

У наступному класі BP_ARPawn створимо конструкцію, яка відповідатиме за:

- сканування середовища
- постановку камери у просторі та лісенер, який її оновлюватиме
- лісенер, що відповідатиме за трансформацію візуалізованої моделі
- затінення побудованої моделі

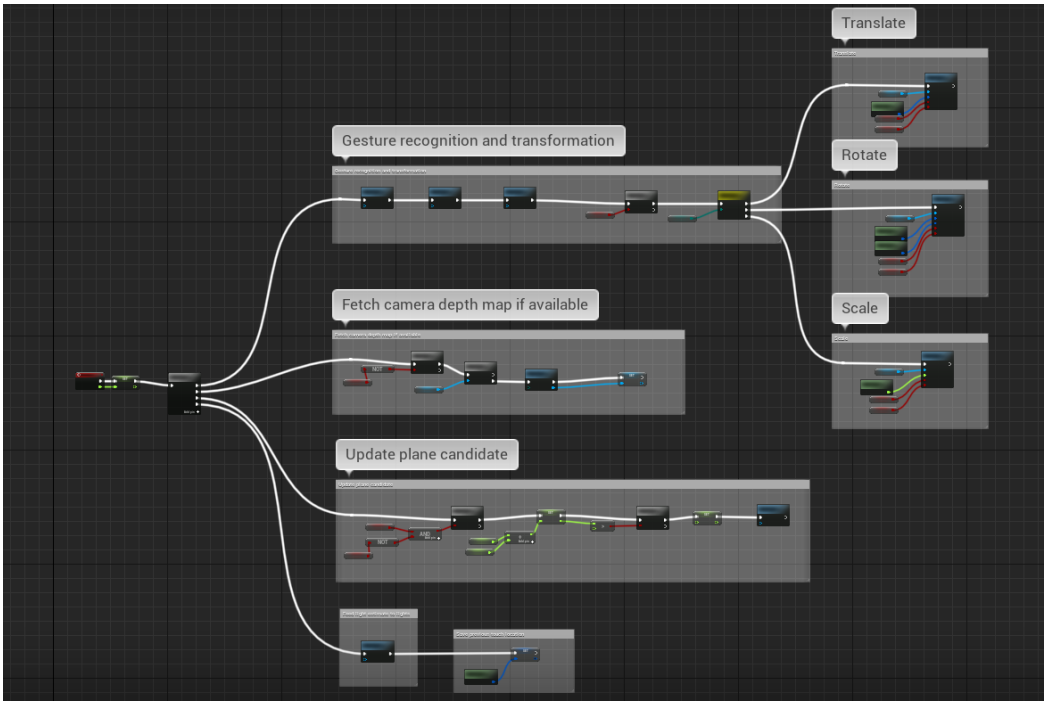


Рисунок 3.11 – загальний граф класу BP_ARPawn

Першим кроком створюємо секвенцію станів та прив’язуємо їх до лісенера(event tick). Подібна конструкція дає можливість програмі виконувати та оновлювати свої блоки паралельно один від одного.

Fetch camera depth map if available - блок класу, мета якого отримати карту глибини, яка необхідна для побудови plane(площина, що відповідатиме навколишньому середовищу, і на якій будуватиметься сам об’єкт).

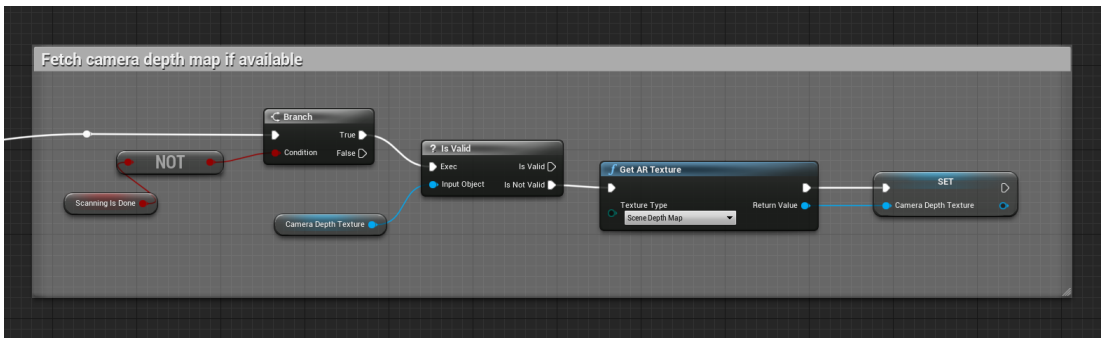


Рисунок 3.12 - Fetch camera depth map

Update plane candidate – блок класу що відповідає за знаходження площини побудови об’єкту. Якщо сканування відбулось та площину не обрано то запускається таймер в 1 секунду. Після закінчення часу обирається остання знайдена площина, в іншому разі блок повториться.

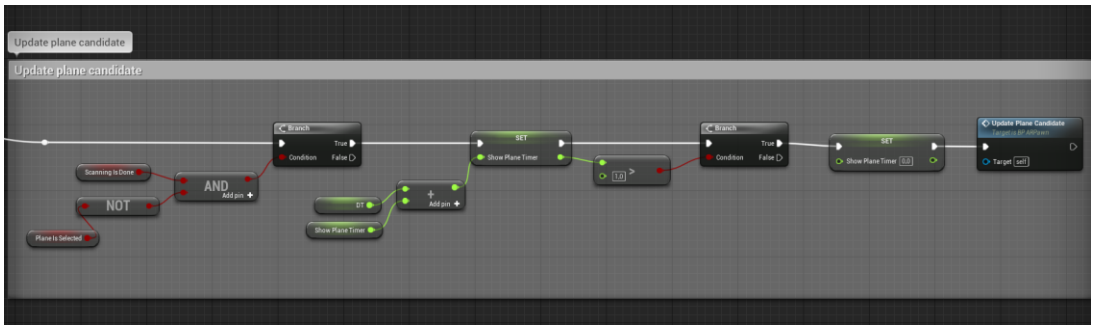


Рисунок 3.13 – Update plane candidate

Gesture recognition and transformation – відповідає за зчитування та розпізнавання жестів, які необхідні для коректної трансформації об’єкту у просторі. У разі активації скидається стан на значення default. Далі програма визначає чи це одиночний дотик чи подвійний. Одиночний дотик відповідає за спавн візуалізованого об’єкту у просторі. У цьому випадку використовується лише зчитування події touch. Подвійні дотики необхідні для обертання та скейлінгу. У першому випадку фіксується дві події: touch та finger direction. Перша визначає початок вектору який задаватиме напрям обертання, а finger direction його відслідковує. В результаті створюється оберт візуалізованого об’єкту навколо своїє осі у напрямку трансформованого на побудовану

поверхню об’єкту. Скейлінг же зчитує довжину обох рівнозначних векторів та збільшує об’єкт, поки умова буде виконуватись. За це відповідає нода Set Placeable Scale.

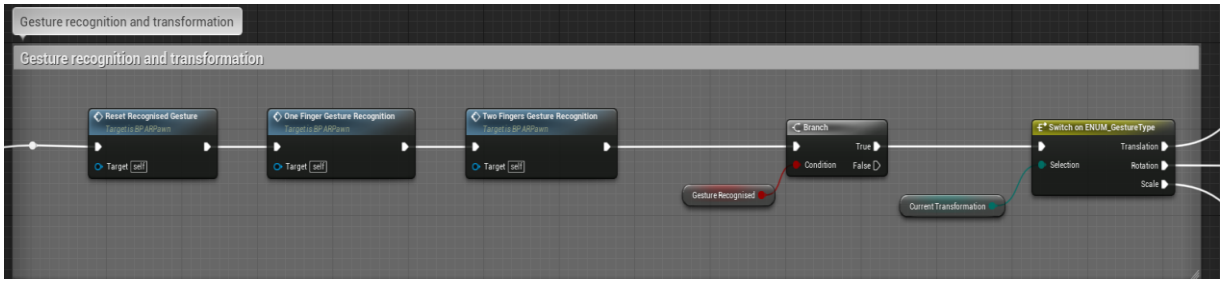


Рисунок 3.14 - Gesture recognition and transformation

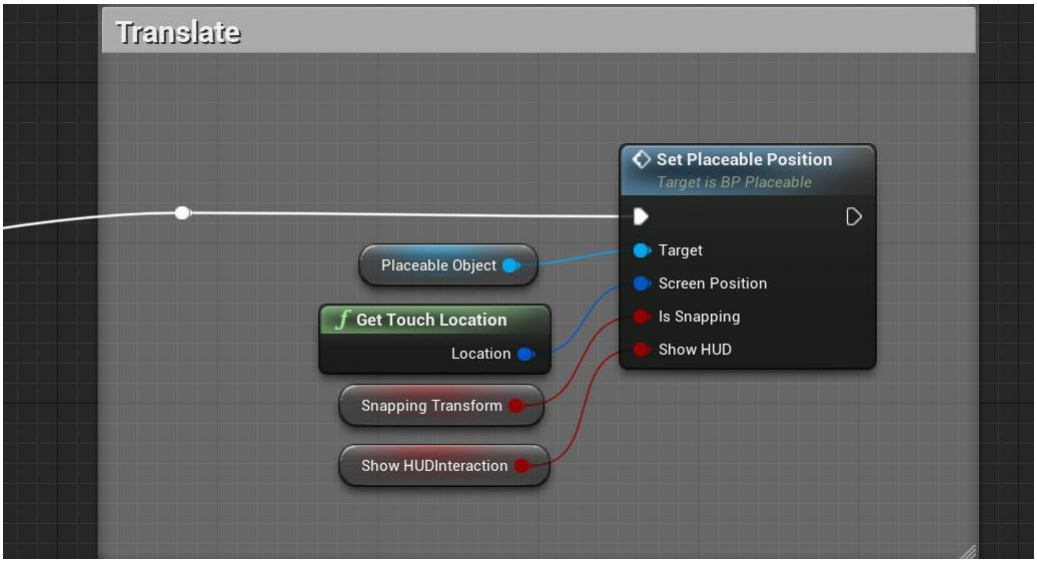


Рисунок 3.15 - модуль створення об’єкту

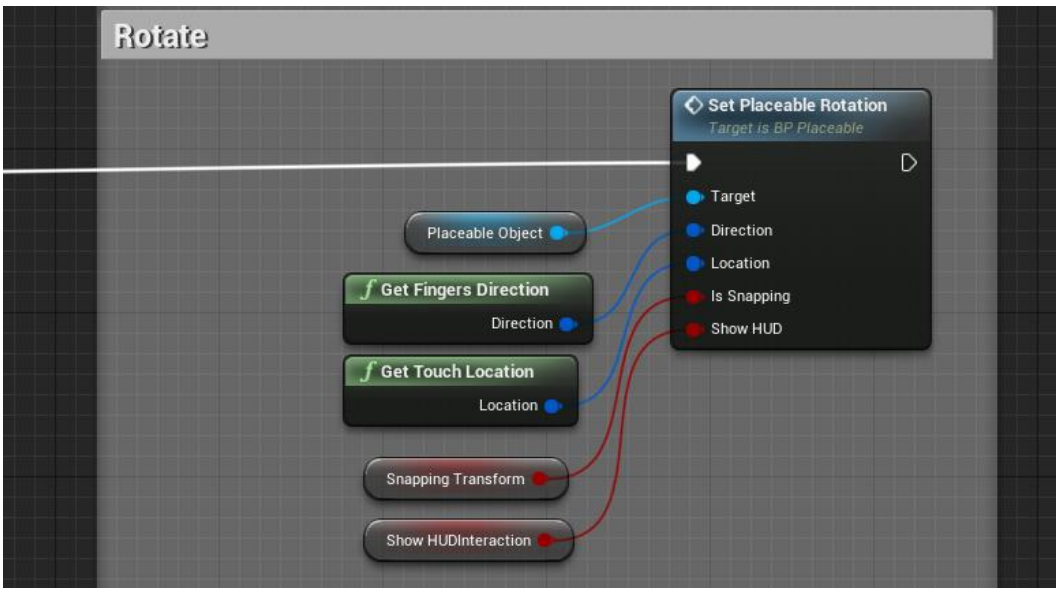


Рисунок 3.16 - модуль обертання

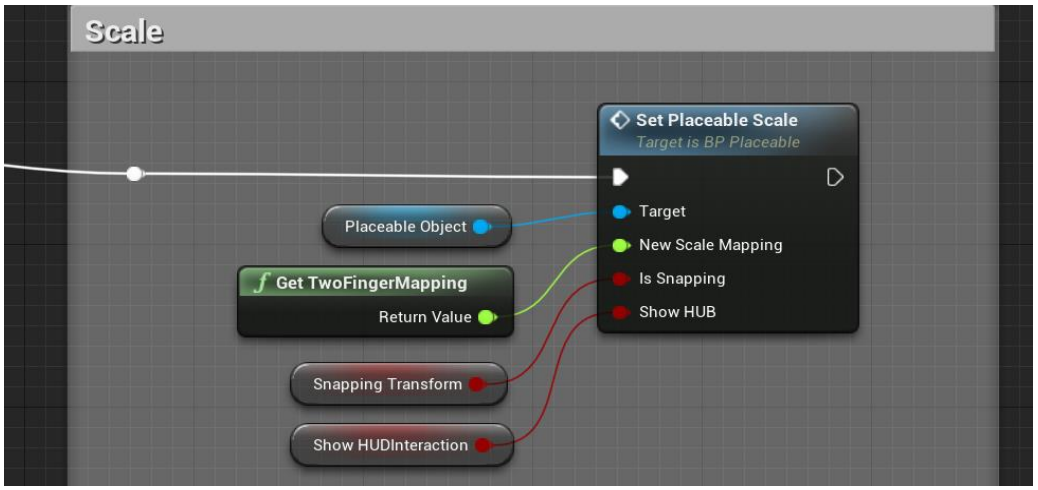


Рисунок 3.17 - модуль зміни розміру

Розглянемо конструкції, які відповідать за зчитування жестів, а саме one finger action та two finger action.

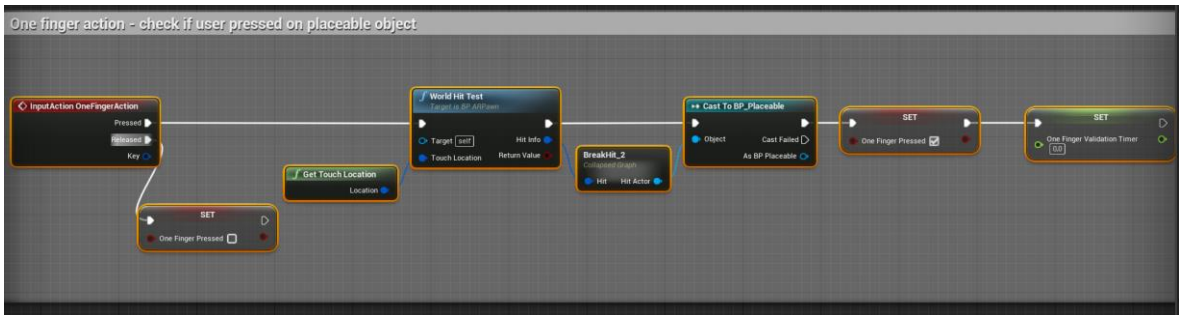


Рисунок 3.18 - one finger action

Спочатку програма фіксує натиск – одиничний жест, після чого викликає ноду World hit test та передає координати жесту до ноди Breaking_Hit2. У разі завершення дотику координати пов’язуються з об’єктом візуалізації та передаються до ноди Cast to BP_Placeable. Результатом є побудова об’єкту та перехід до первинного стану.

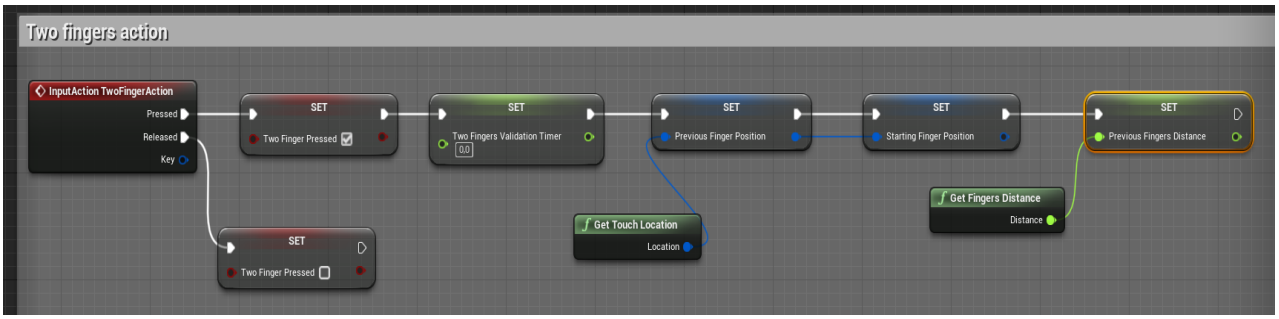


Рисунок 3.19 - two fingers action

У даному випадку програма фіксує подвійний натиск. У разі якщо це дійсність то перевіряється минула та початкова координата touch. У разі рівності визначається поле Touch Location, необхідне для обертання. У іншому ж випадку встановлюється значення Fingers Distance, яке визначає ступінь збільшення візуалізованого об’єкту.

Розглянемо клас Place_object, який відповідальний як за первинну побудову об’єкту, так і за її оновлення у разі сценаріїв із жестами.

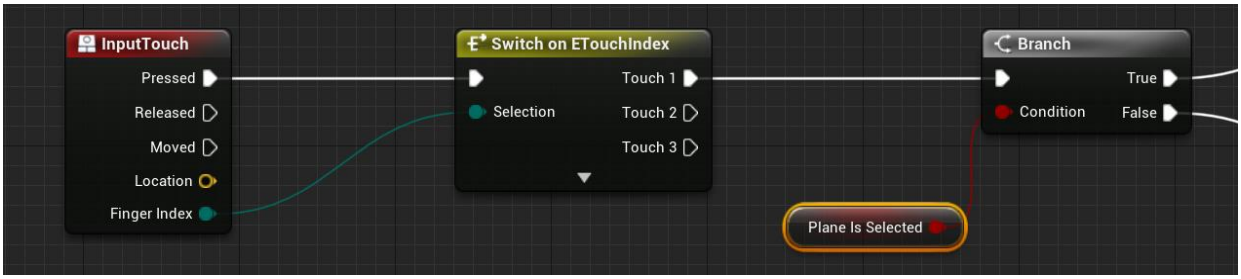


Рисунок 3.20 - place_object

Першим кроком відбувається перевірка натискання: якщо це дотик та їх кількість рівна 1 то відбувається наступна перевірка : чи визначено площину побудови. Якщо значення рівне false то переходимо до модулю BP Plane Selected.

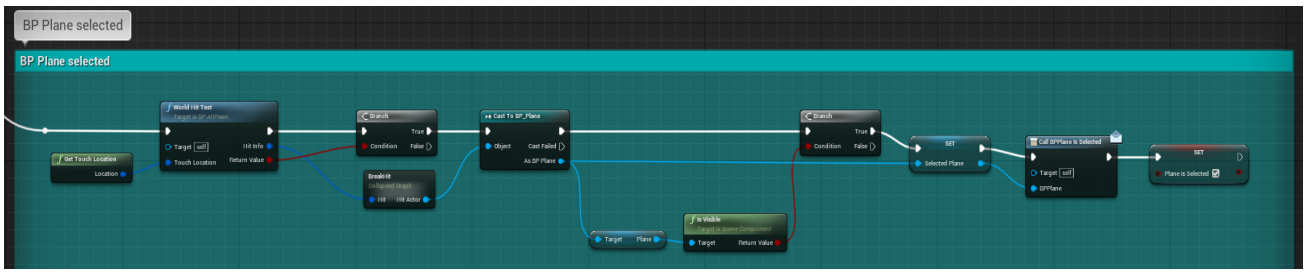


Рисунок 3.21 - BP Plane Selected

Тут сценарій дуже схожий на той, що описується в ноді розпізнавання жестів: визначення початкової координати дотику та перехід до ноди Cast to BP_Plane. У разі якщо площина видима то вона зберігається у полі Selected Plane, а маркер Plane is selected отримує значення true. Якщо ж значення після ноди Switch to ETouchIndex рівна true то відбувається перехід до модулю Plane Object.

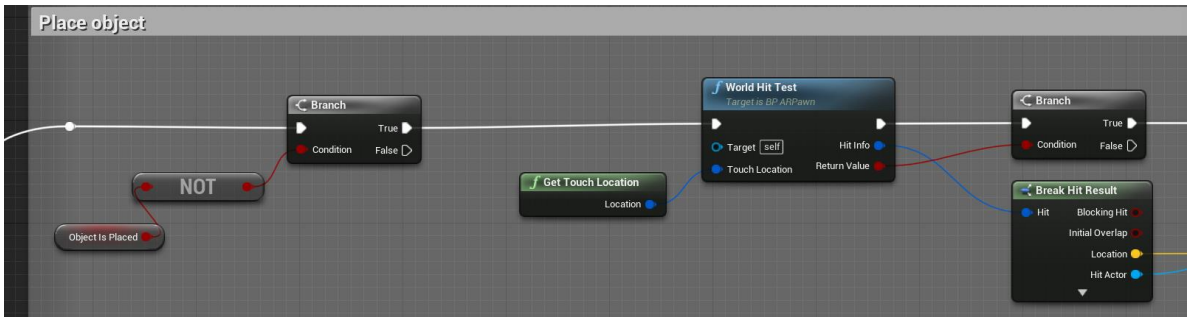


Рисунок 3.22 - Plane Object

У разі, якщо об’єкт ще не розташований то відбувається координатна перевірка та зберігання поточних координат площини. У разі якщо збережена площина та поточна рівні то програма переходить до побудови об’єкту за допомогою модуля SpawnActor BP Placeable .

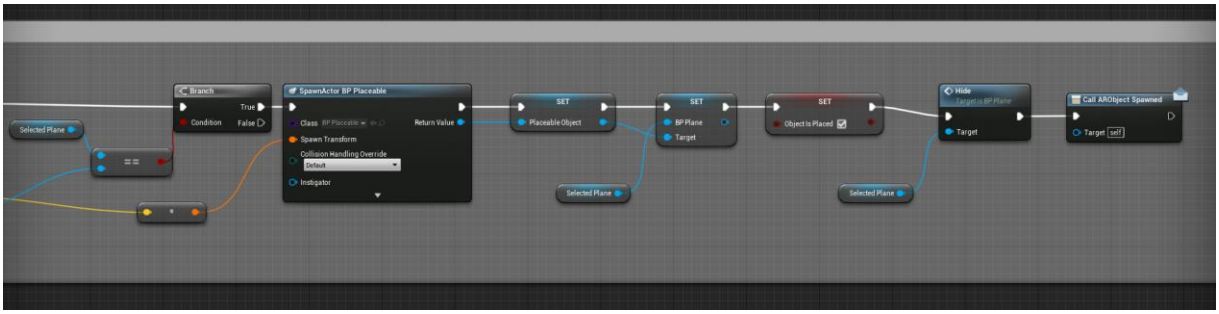


Рисунок 3.23 - Plane Object(2)

Після побудови оновлюються значення Selected Plane a Object is placed отримує значення true.

Процес освітлення та затінення моделі відбувається за результатами Depth map та заготовленої HDRI карти. HDRI карта - (High Dynamic Range Image) текстура з високим динамічним діапазоном, націлена для імітації освітлення навколишнього середовища.

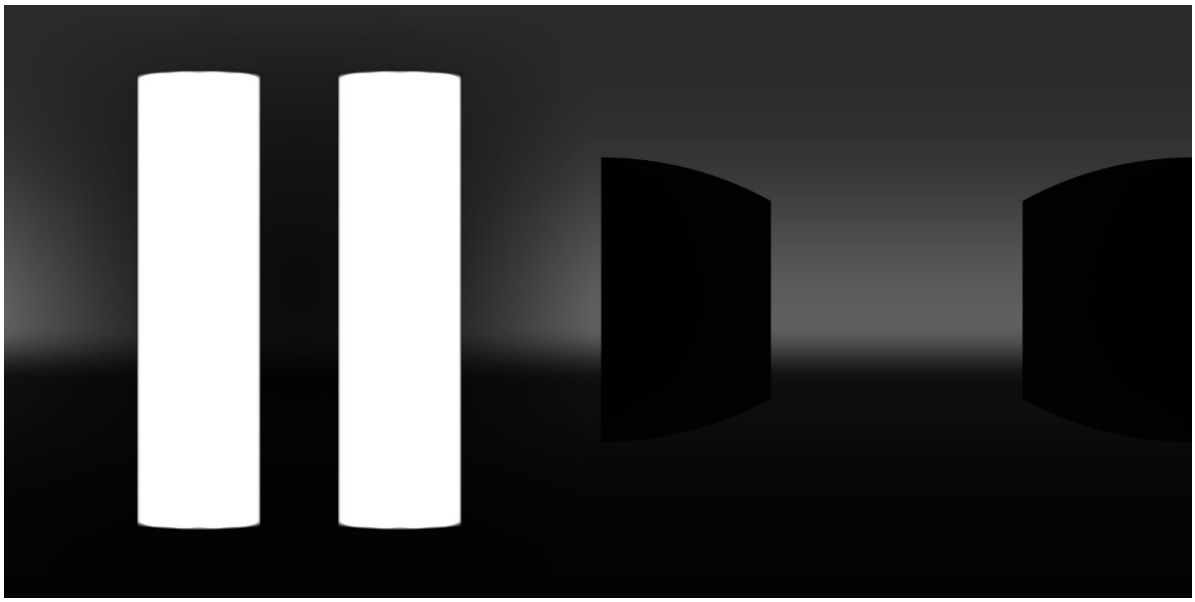
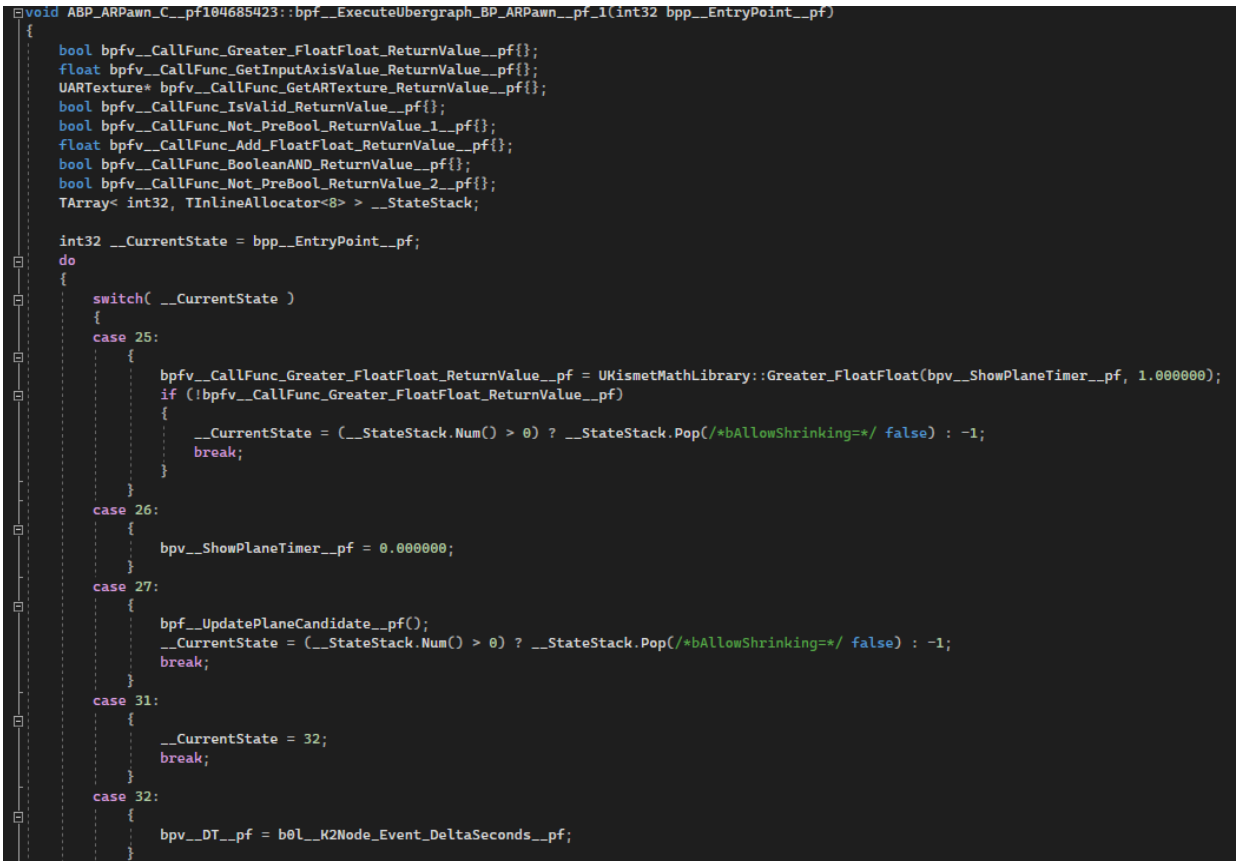


Рисунок 3.24 - приклад HDRI карти

За допомогою AR Core та можливості аналізу освітленості сканованого простору додаток може змінювати яскравість цієї текстури.

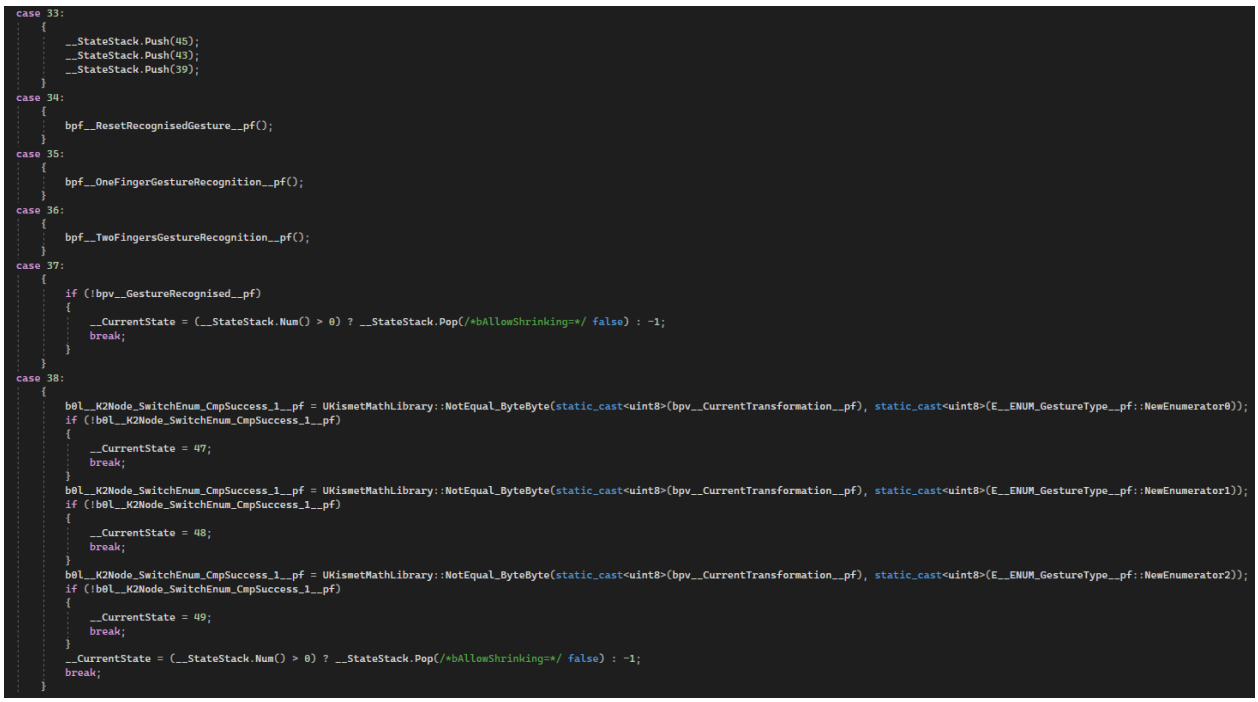
Для порівняння приведемо деякі створені блюпрінти до C++ коду.



```
void ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_1(int32 bpp__EntryPoint__pf)
{
    bool bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf{};
    float bpfv__CallFunc_GetInputAxisValue_ReturnValue__pf{};
    UARTexture* bpfv__CallFunc_GetARTexture_ReturnValue__pf{};
    bool bpfv__CallFunc_IsValid_ReturnValue__pf{};
    bool bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf{};
    float bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf{};
    bool bpfv__CallFunc_BooleanAND_ReturnValue__pf{};
    bool bpfv__CallFunc_Not_PreBool_ReturnValue_2__pf{};
    TArray< int32, TInlineAllocator<8> > __StateStack;

    int32 __CurrentState = bpp__EntryPoint__pf;
    do
    {
        switch( __CurrentState )
        {
        case 25:
            {
                bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf = UKismetMathLibrary::Greater_FloatFloat(bpv__ShowPlaneTimer__pf, 1.000000);
                if (!bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf)
                {
                    __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                    break;
                }
            }
        case 26:
            {
                bpv__ShowPlaneTimer__pf = 0.000000;
            }
        case 27:
            {
                bpf__UpdatePlaneCandidate__pf();
                __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                break;
            }
        case 31:
            {
                __CurrentState = 32;
                break;
            }
        case 32:
            {
                bpv__DT__pf = b0l__K2Node_Event_DeltaSeconds__pf;
            }
```

Рисунок 3.25 - код класу AR_Pawn(1)



```
case 33:
    {
        __StateStack.Push(45);
        __StateStack.Push(43);
        __StateStack.Push(39);
    }
case 34:
    {
        bpf__ResetRecognisedGesture__pf();
    }
case 35:
    {
        bpf__OneFingerGestureRecognition__pf();
    }
case 36:
    {
        bpf__TwoFingersGestureRecognition__pf();
    }
case 37:
    {
        if (!bpv__GestureRecognised__pf)
        {
            __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
            break;
        }
    }
case 38:
    {
        b0l__K2Node_SwitchEnum_CmpSuccess_1__pf = UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentTransformation__pf), static_cast<uint8>(E__ENUM_GestureType__pf::NewEnumerator0));
        if (!b0l__K2Node_SwitchEnum_CmpSuccess_1__pf)
        {
            __CurrentState = 47;
            break;
        }
        b0l__K2Node_SwitchEnum_CmpSuccess_1__pf = UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentTransformation__pf), static_cast<uint8>(E__ENUM_GestureType__pf::NewEnumerator1));
        if (!b0l__K2Node_SwitchEnum_CmpSuccess_1__pf)
        {
            __CurrentState = 48;
            break;
        }
        b0l__K2Node_SwitchEnum_CmpSuccess_1__pf = UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentTransformation__pf), static_cast<uint8>(E__ENUM_GestureType__pf::NewEnumerator2));
        if (!b0l__K2Node_SwitchEnum_CmpSuccess_1__pf)
        {
            __CurrentState = 49;
            break;
        }
        __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
        break;
    }
```

Рисунок 3.26 - код класу AR_Pawn(2)

```
case 39:
    {
        bpfv__CallFunc_Not_PreBool_ReturnValue_2__pf = UKismetMathLibrary::Not_PreBool(bpv__ScanningIsDone__pf);
        if (!bpfv__CallFunc_Not_PreBool_ReturnValue_2__pf)
        {
            __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
            break;
        }
    }
case 40:
    {
        bpfv__CallFunc_IsValid_ReturnValue__pf = UKismetSystemLibrary::IsValid(bpv__CameraDepthTexture__pf);
        if (!bpfv__CallFunc_IsValid_ReturnValue__pf)
        {
            __CurrentState = 41;
            break;
        }
        __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
        break;
    }
case 41:
    {
        bpfv__CallFunc_GetARTexture_ReturnValue__pf = UARBlueprintLibrary::GetARTexture(EARTextureType::SceneDepthMap);
    }
case 42:
    {
        bpv__CameraDepthTexture__pf = bpfv__CallFunc_GetARTexture_ReturnValue__pf;
        __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
        break;
    }
case 43:
    {
        bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf = UKismetMathLibrary::Not_PreBool(bpv__PlaneIsSelected__pf);
        bpfv__CallFunc_BooleanAND_ReturnValue__pf = UKismetMathLibrary::BooleanAND(bpv__ScanningIsDone__pf, bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf);
        if (!bpfv__CallFunc_BooleanAND_ReturnValue__pf)
        {
            __CurrentState = (__StateStack.Num() > 0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
            break;
        }
    }
case 44:
    {
        bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf = UKismetMathLibrary::Add_FloatFloat(bpv__DT__pf, bpv__ShowPlaneTimer__pf);
        bpv__ShowPlaneTimer__pf = bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf;
        __CurrentState = 25;
        break;
    }
case 45:
    {
        bpf__FeedLightEstimate__pf();
    }
```

Рисунок 3.27 - код класу AR_Pawn(3)

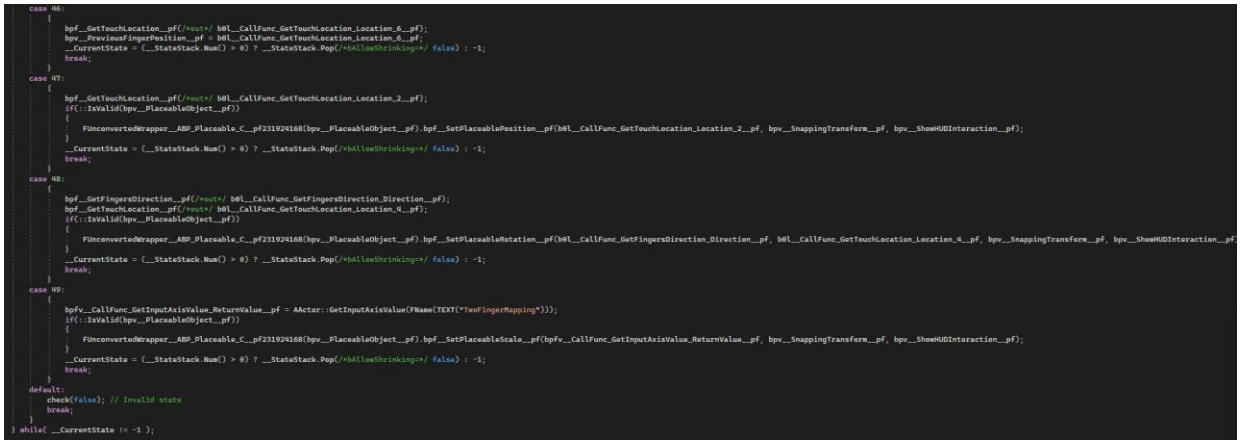


Рисунок 3.28 - код класу AR_Pawn(4)

```
void ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_2(int32 bpp__EntryPoint__pf)
{
    APlayerController* bpfv__CallFunc_GetPlayerController_ReturnValue__pf{};
    UUserWidget* bpfv__CallFunc_Create_ReturnValue__pf{};
    check(bpp__EntryPoint__pf == 64);
    // optimized KCST_UnconditionalGoto
    bpfv__CallFunc_GetPlayerController_ReturnValue__pf = UGameplayStatics::GetPlayerController(this, 0);
    bpfv__CallFunc_Create_ReturnValue__pf = UWidgetBlueprintLibrary::Create(
        this, CastChecked<UClass>(CastChecked<UDynamicClass>(ABP_ARPawn_C__pf104685423::StaticClass())->UsedAssets[1],
        ECastCheckedType::NullAllowed), bpfv__CallFunc_GetPlayerController_ReturnValue__pf);
    bpv__MainMenu__pf = bpfv__CallFunc_Create_ReturnValue__pf;
    if(::IsValid(bpv__MainMenu__pf))
    {
        bpv__MainMenu__pf->UUserWidget::AddToViewport(0);
    }
    return; //KCST_EndOfThread
}
```

Рисунок 3.29 - код класу AR_Pawn(5)

Як можна побачити зі скріншотів, модуль рушію Nativization Script відпрацьовує коректно, проте повертає шаблонний код. Із переваг : це надає розширений функціонал для створення проекту, тобто можна створювати власні незалежні ноди. Мінусом подібної технології є досить важкий для редагування код. У подібних досить легких додатках краще використовувати блюпринти, на швидкість роботи це ніяк не вплине.

3.3 Демонстрація роботи

Спробуємо за допомогою додатку помістити раніше створений диван на відсканований поверхню.

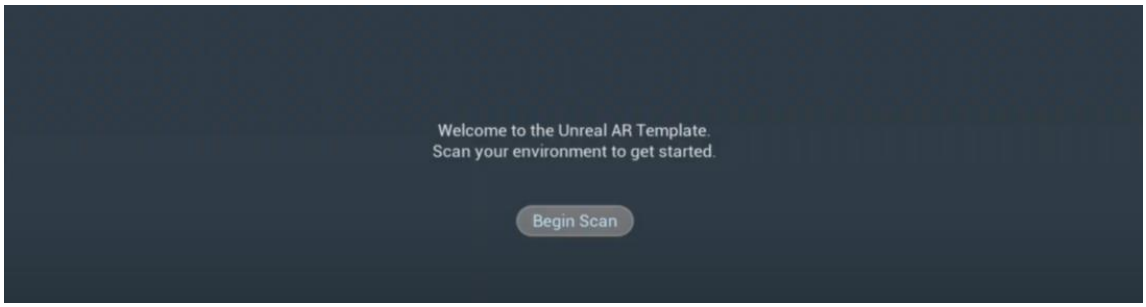


Рисунок 3.30 - початок роботи

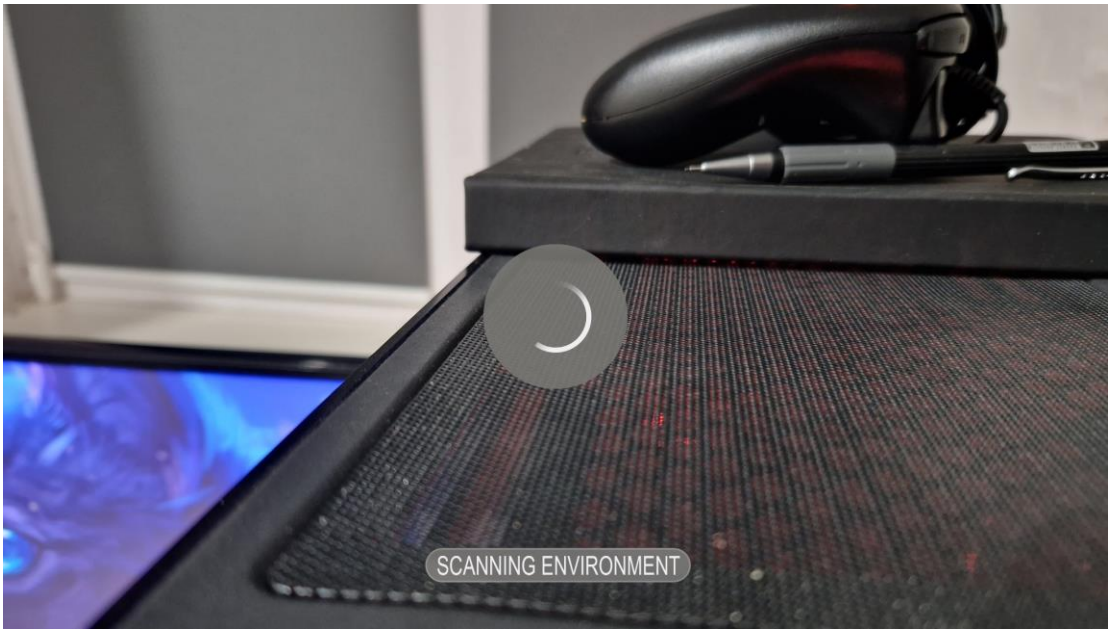


Рисунок 3.31 - сканування

Рисунок 3.32 - візуалізація

Рисунок 3.33 - обертання

Рисунок 3.34 - скейлінг

У даній демонстрації був не задіяний модуль аналізу рівня освітленості. Модель отримала затінення за допомогою вище приведеної HDR карти. Як результат тіні під візуалізованим об’єктом є нейтральні, та немаються якогось конкретного направлення. Проте навіть з базовими просити шейдерами візуалізація відбувається правильно. Усі налаштовані об’єкти виглядають аналогічно тим що були у редакторі.

ВИСНОВОК

Результатом даної роботи стало створення демонстраційного інформаційного додатку для візуалізації 3D об’єкту в інтер’єрі з використанням технологій доповненої реальності.

Під час виконання даної роботи було проаналізовано історію створення як віртуальної так і доповненої реальності, та їх ключові відмінності, а саме те, що віртуальна реальність повністю замінює вже існуючу, а доповнена націлена збагатити та розширити її.

Було проаналізовано різні сфери використання подібної технології. Найбільш поширеною вона є у сфері маркетингу, освіти а також використовується під час розробки ігор.

Також було порівняно основні і найбільші ядра для розробки додатків доповненої реальності, а саме ARCore та ARKit. Вони досить схожі по своєму функціоналу, проте мають як свої переваги так і недоліки. Головна відмінність у таргетній платформі: ARCore розроблявся для платформи андроїд, хоча здатен працювати на IOS, але не навпаки.

Перед розробкою демонстраційного додатку було порівняно два основні рушії, які можуть вільно взаємодіяти з вище описаними API. Мова йде про Unreal Engine та Unity 3d. Ці рушії мають деякі спільні риси, як кросплатформеність, та чудову інтеграцію бібліотек для роботи з доповненою реальністю, проте сам процес розробки суттєво відрізняється.

Unity може працювати лише зі скриптами, що досить відчутно може сповільнювати роботу у недосвіченого фахівця. Також відрізняється налаштування шейдерів об’єкту. Крім того відсутній модуль інтеграції вже раніше створених gltf моделей.

Unreal у цьому випадку має трохи інший пайплайн розробки додатків. Має безмежні можливості у покращенні графічних шейдерів. Та основна його перевага це технологія blueprint. Крім того її без суттєвих труднощів можна конвертувати у класичний C++ код. Це необхідно у разі написання власних

вузько направлених функцій, яких рушій немає. Крім того було порівняно блюпринти зі звичайним кодом. Основним мінусом є те, що код має шаблонну структуру, та погано підходить для редагування. До того ж використання технології blueprint не сказується на швидкості роботи додатку, лише на збірці та компіляції.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Epic games, “Unreal engine 4 documentation”, July 2011.

2. J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, M. Ivkovic, “Augmented reality technologies, systems and applications”, 2010

3. Wendy E. Mackay, “Augmented Reality: Linking real and virtual worlds A new paradigm for interacting with computers ”, Université de ParisSud

4. “Wikipedia. Unreal engine 4.” - [resource]: https://en.wikipedia.org/wiki/Unreal_Engine

5. Sean Morey, John Tinnell: Augmented Reality Innovative Perspectives Across Art, Industry, and Academia

6. Mitch McCaffrey: Unreal® Engine VR Cookbook: Developing Virtual Reality with UE4

7. Трусов, Б.О. Комп’ютерна 3D-гра з використанням Unity Engine и C# [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 122 - комп’ютерні науки (інформатика) / Б.О. Трусов; наук. кер. О.В. Шутилєва. - Суми: СумДУ, 2021. - 45 с.

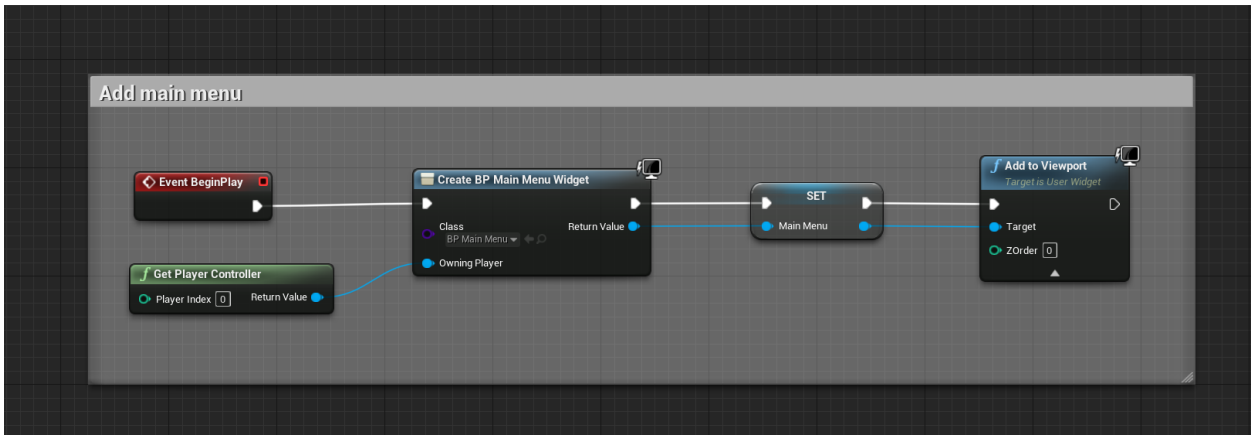
8. Mihael Lanham: Learn ARCore - Fundamentals of Google ARCore (2018)

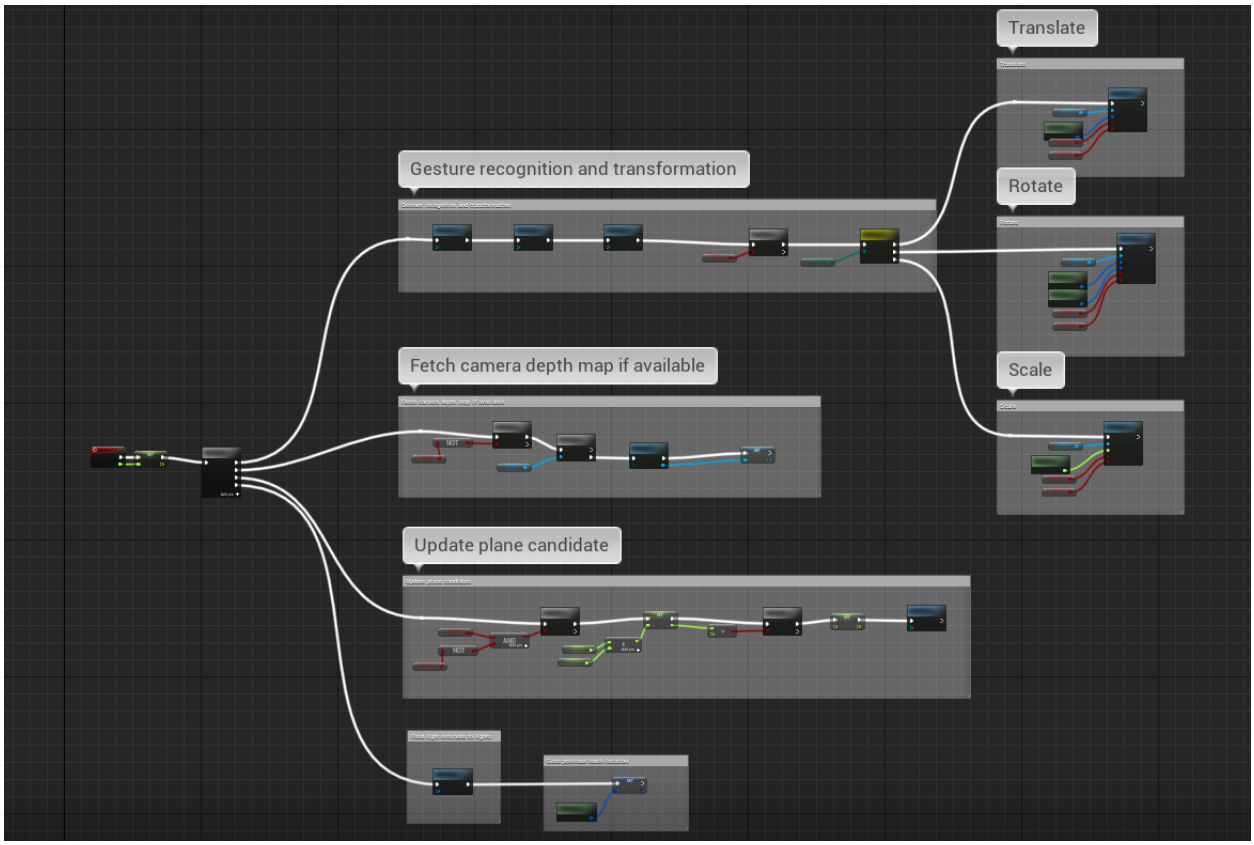
9. Tom Shannon: Unreal Engine 4 for design visualization

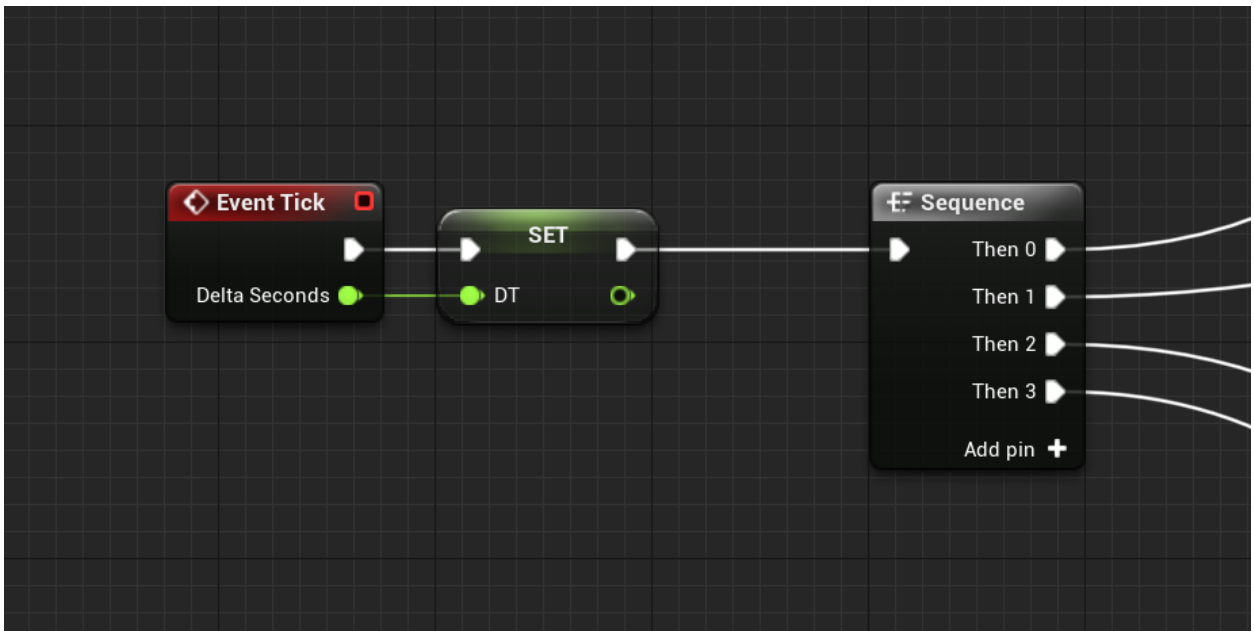
10. Brenden Sewell: Blueprints Visual Scripting for Unreal Engine: Build professional 3D games with Unreal Engine 4's Visual Scripting system (2015)

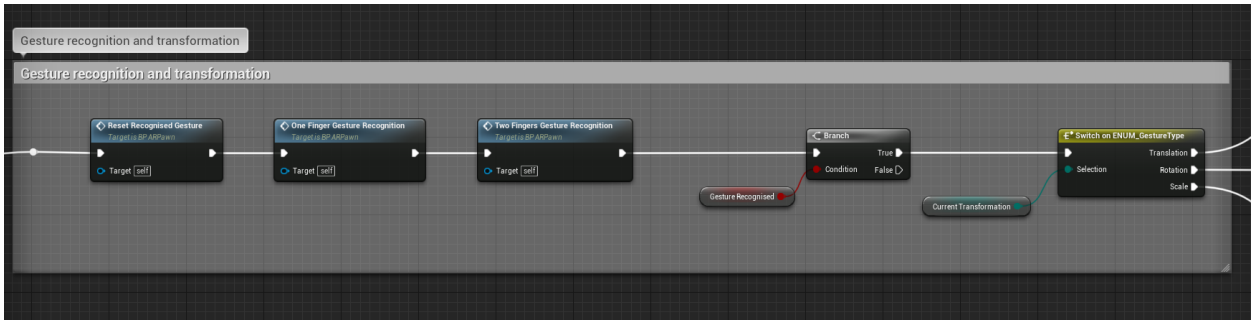
11. Anthony Davis : Unity 3D Game Development: Designed for passionate game developers—Engineered to build professional games (2022)

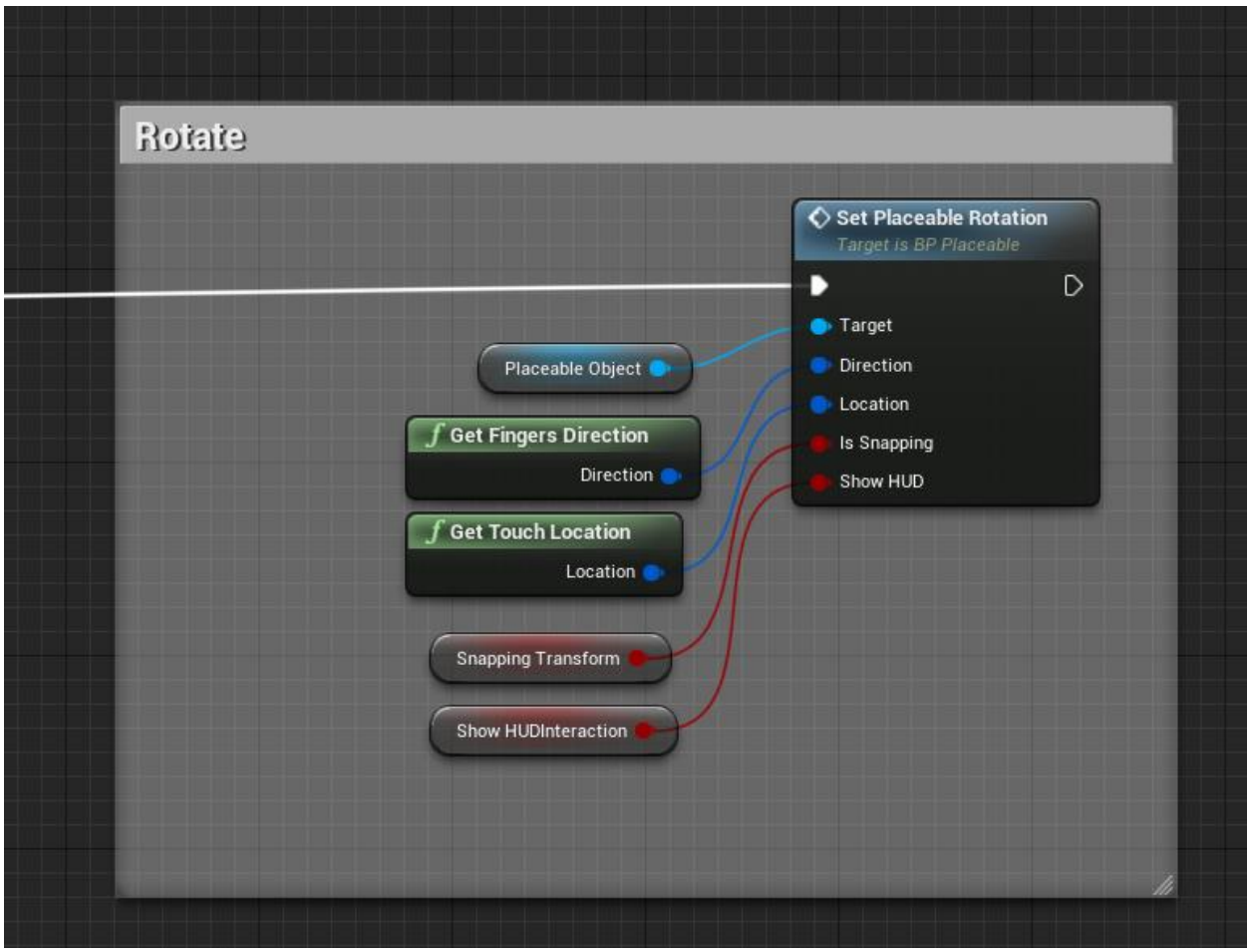
ДОДАТОК А

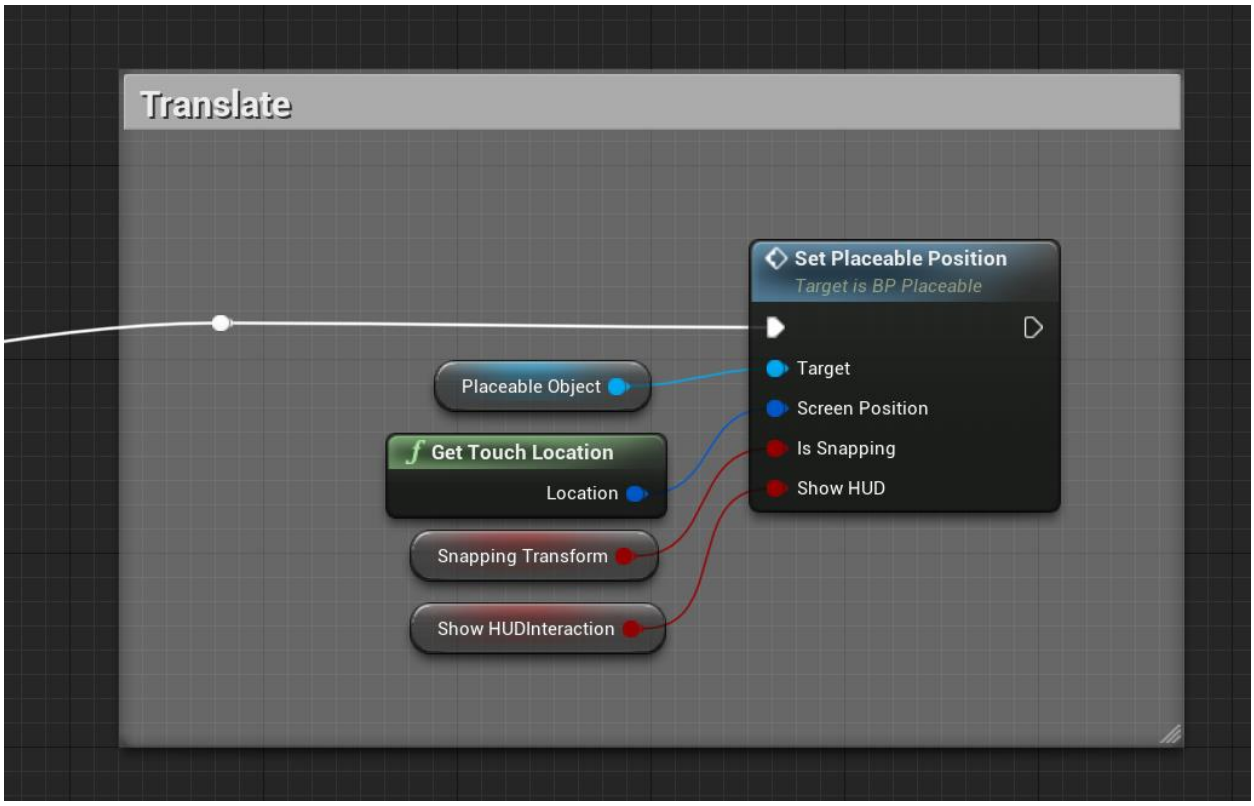

Add main menu
Event BeginPlay
Create BP Main Menu Widget
Class
BP Main Menu
Return Value
Owning Player
SET
Main Menu
Add to Viewport
Target is User Widget
Target
ZOrder
0
Get Player Controller
Player Index
0
Return Value

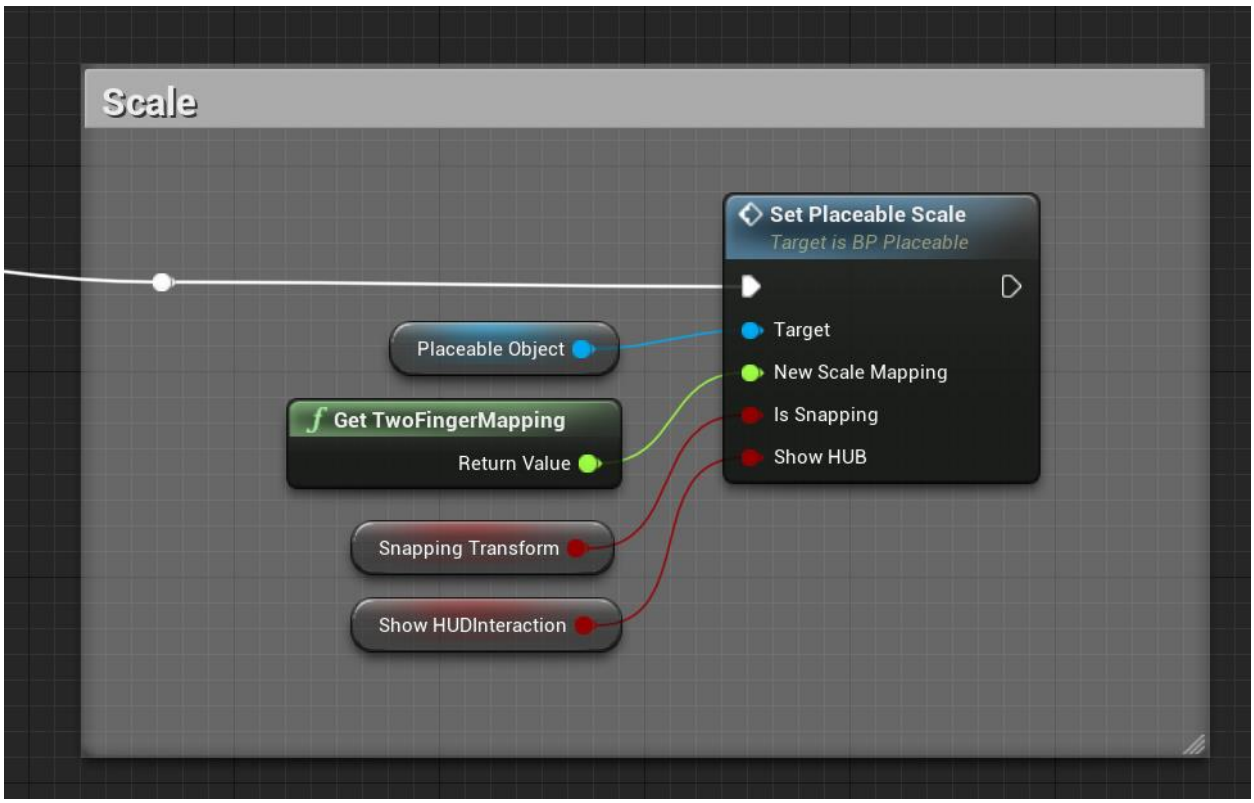

Translate
Gesture recognition and transformation
Rotate
Fetch camera depth map if available
Scale
Update plane candidate

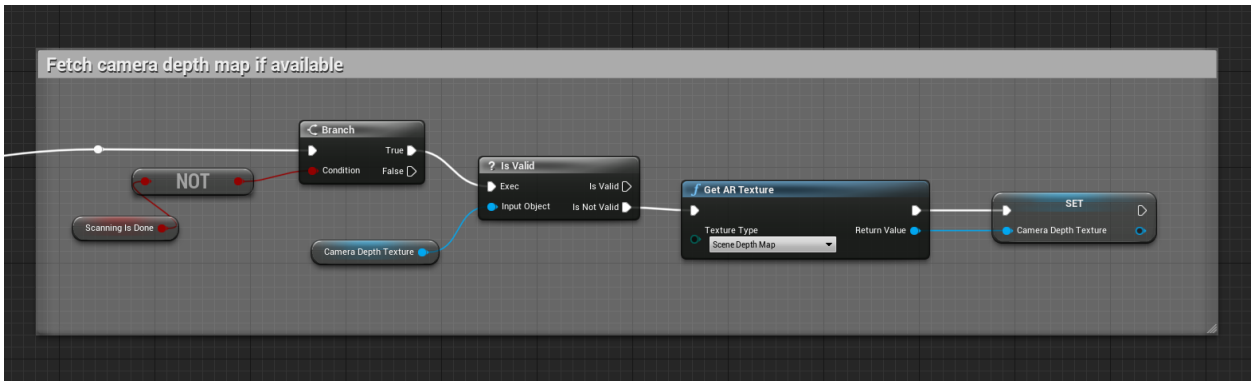

Event Tick
Delta Seconds
SET
DT
Sequence
Then 0
Then 1
Then 2
Then 3
Add pin

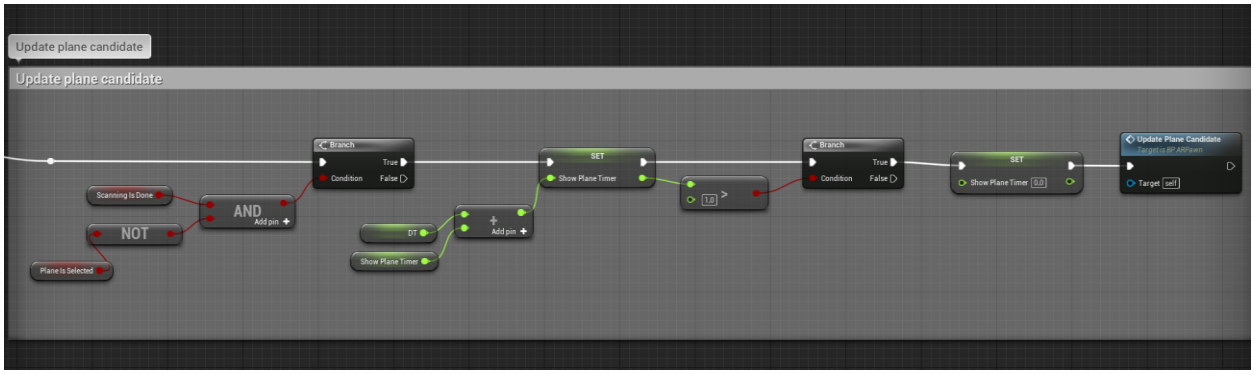

Gesture recognition and transformation
Gesture recognition and transformation
Reset Recognised Gesture
Target is BP_ARPawn
Target self
One Finger Gesture Recognition
Target is BP_ARPawn
Target self
Two Fingers Gesture Recognition
Target is BP_ARPawn
Target self
Branch
Condition
True
False
Gesture Recognised
Current Transformation
Switch on ENUM_GestureType
Selection
Translation
Rotation
Scale

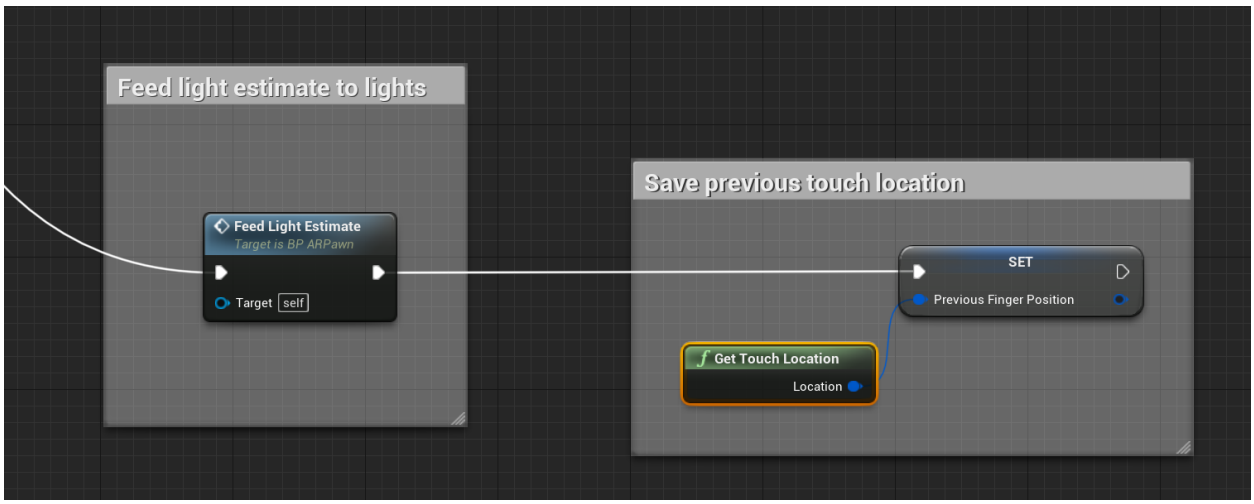

Rotate
Set Placeable Rotation
Target is BP Placeable
Target
Direction
Location
Is Snapping
Show HUD
Placeable Object
Get Fingers Direction
Direction
Get Touch Location
Location
Snapping Transform
Show HUDInteraction

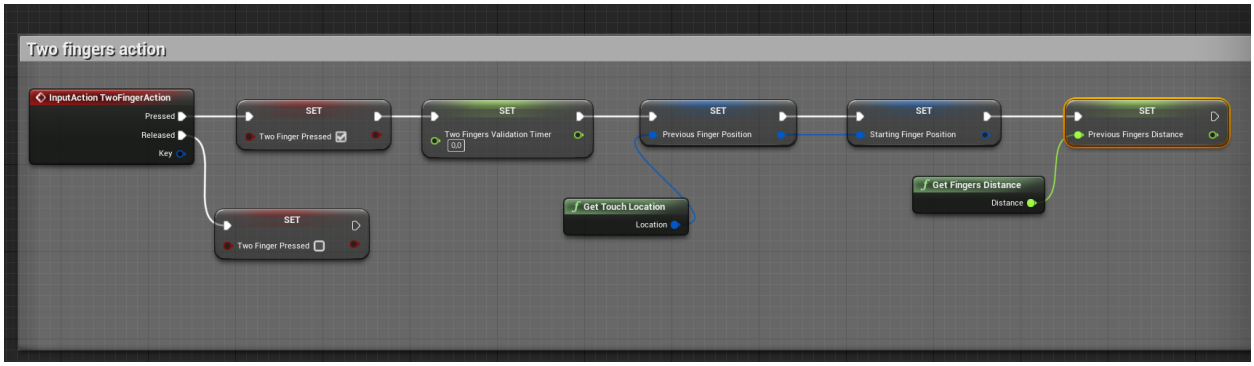

Translate
Set Placeable Position
Target is BP Placeable
Target
Screen Position
Is Snapping
Show HUD
Placeable Object
Get Touch Location
Location
Snapping Transform
Show HUDInteraction

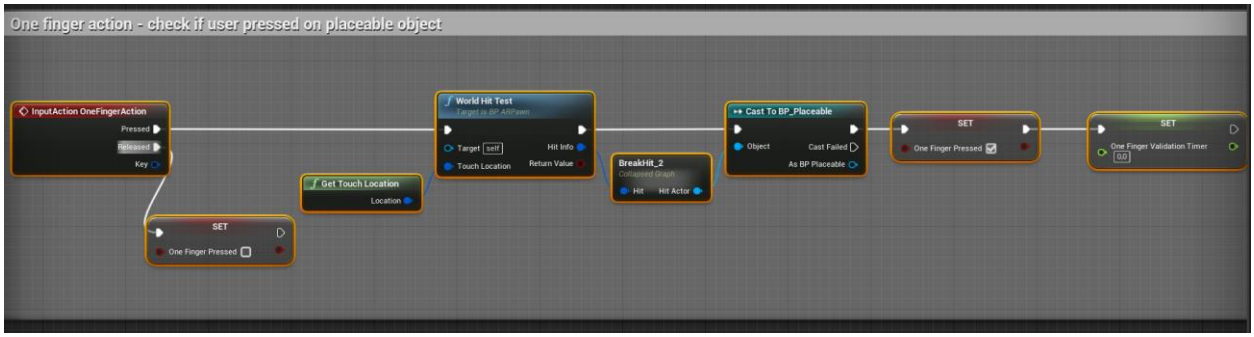

Scale
Set Placeable Scale
Target is BP Placeable
Placeable Object
Target
New Scale Mapping
Get TwoFingerMapping
Return Value
Is Snapping
Show HUB
Snapping Transform
Show HUDInteraction

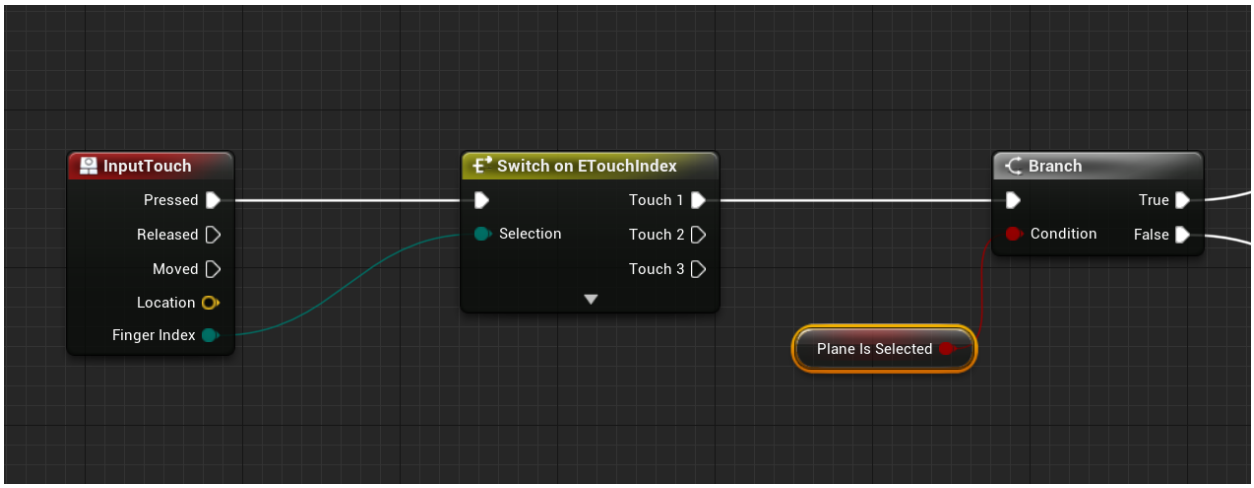

Fetch camera depth map if available
Branch
True
Condition
False
NOT
Scanning Is Done
Is Valid
Exec
Is Valid
Input Object
Is Not Valid
Camera Depth Texture
Get AR Texture
Texture Type
Scene Depth Map
Return Value
SET
Camera Depth Texture

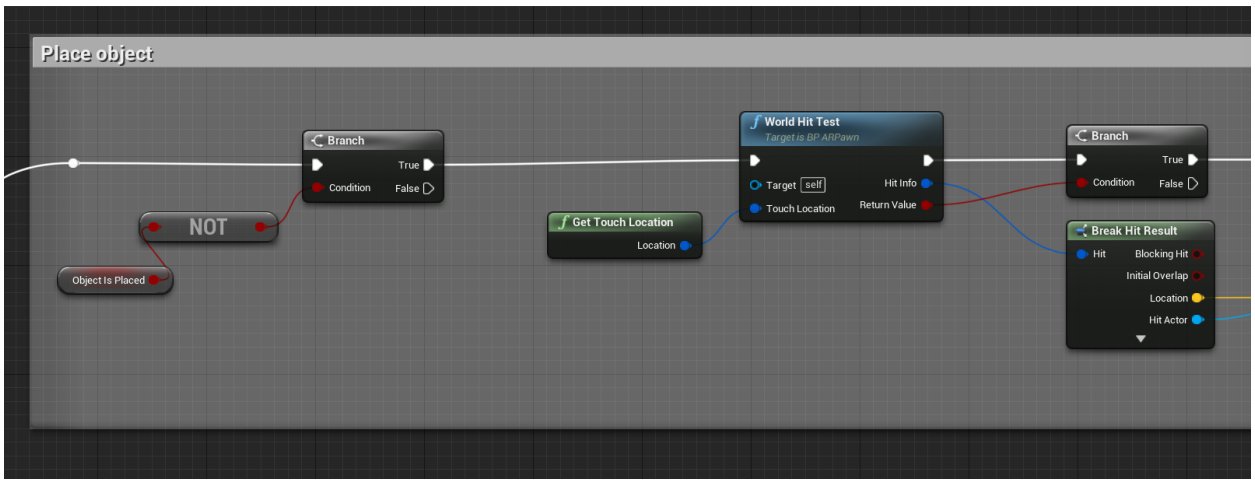

Update plane candidate
Update plane candidate
Branch
Scanning Is Done
NOT
Plane Is Selected
AND
Add pin
DT
Show Plane Timer
SET
Show Plane Timer
Update Plane Candidate
Target

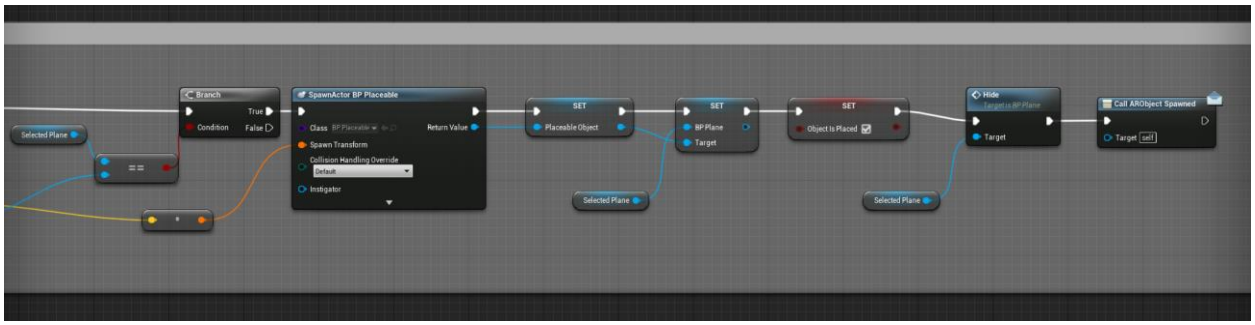

Feed light estimate to lights
Feed Light Estimate
Target is BP ARPawn
Target self
Save previous touch location
SET
Previous Finger Position
Get Touch Location
Location

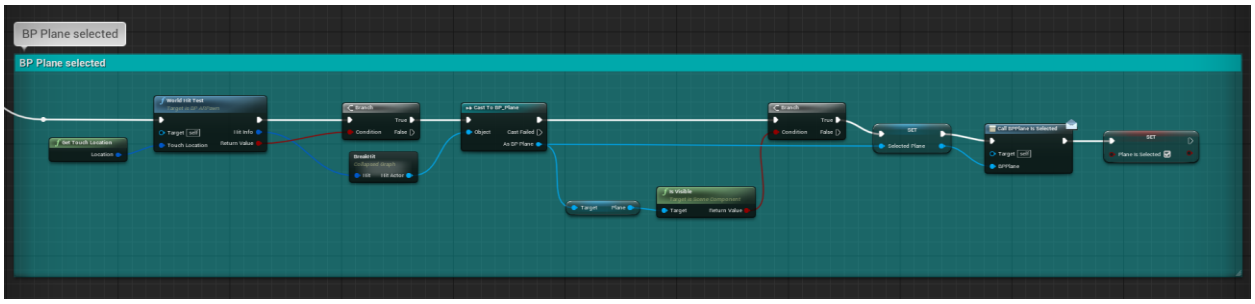

Two fingers action
InputAction TwoFingerAction
Pressed
Released
Key
SET
Two Finger Pressed
Two Fingers Validation Timer
Previous Finger Position
Starting Finger Position
Previous Fingers Distance
Get Touch Location
Location
Get Fingers Distance
Distance

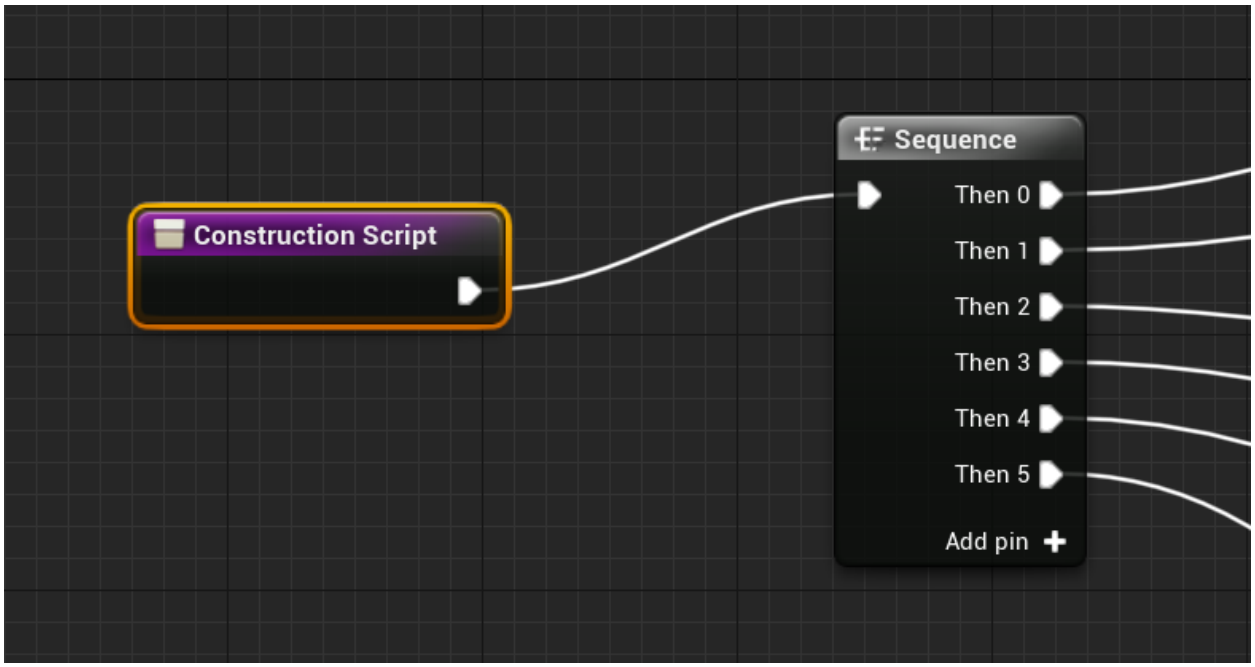

One finger action - check if user pressed on placeable object
InputAction OneFingerAction
Pressed
Released
Key
World Hit Test
Target self
Touch Location
Hit Info
Return Value
Get Touch Location
Location
BreakHit_2
Hit
Hit Actor
Cast To BP_Placeable
Object
Cast Failed
As BP Placeable
SET
One Finger Pressed
One Finger Validation Timer

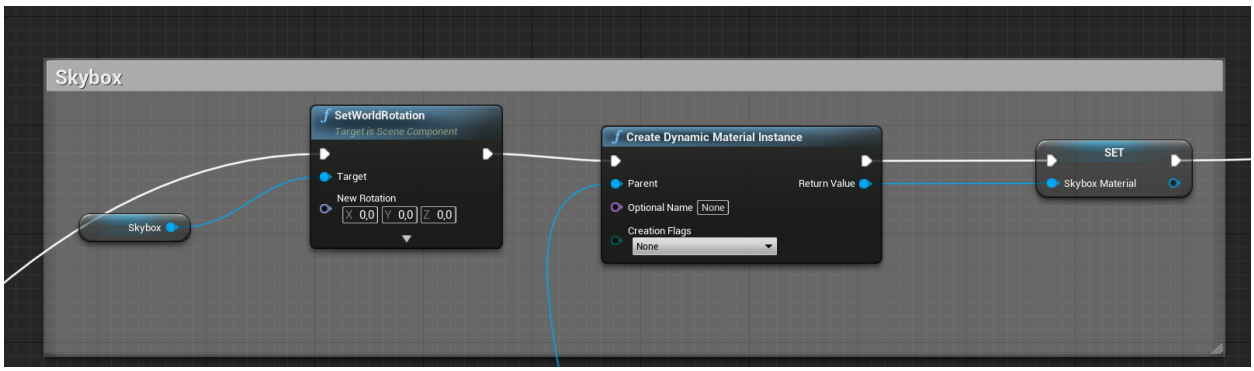

InputTouch
Pressed
Released
Moved
Location
Finger Index
Switch on ETouchIndex
Selection
Touch 1
Touch 2
Touch 3
Branch
Condition
True
False
Plane Is Selected

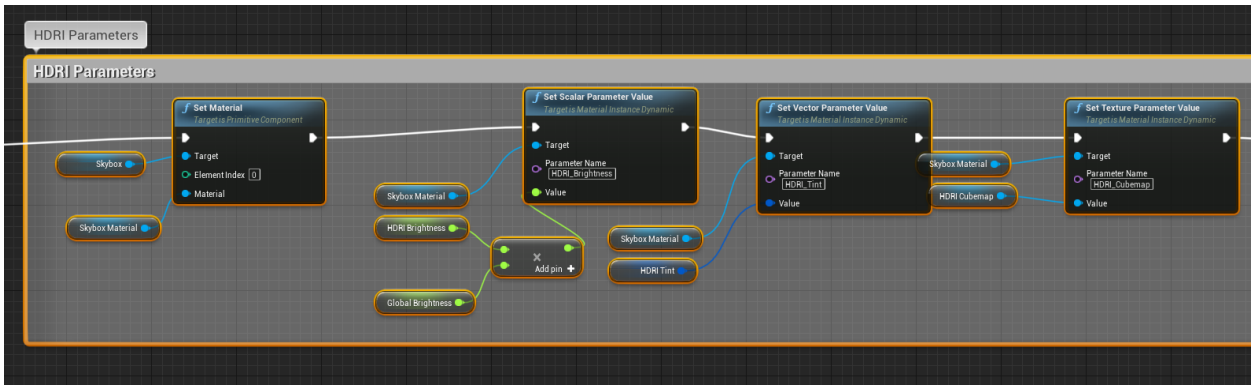

Place object
Branch
True
Condition
False
NOT
Object Is Placed
World Hit Test
Target is BP ARPawn
Target
self
Touch Location
Hit Info
Return Value
Get Touch Location
Location
Branch
True
Condition
False
Break Hit Result
Hit
Blocking Hit
Initial Overlap
Location
Hit Actor

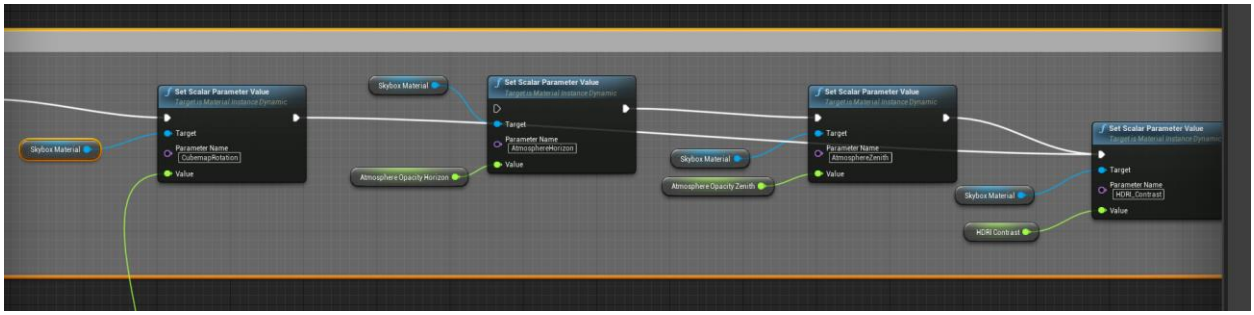


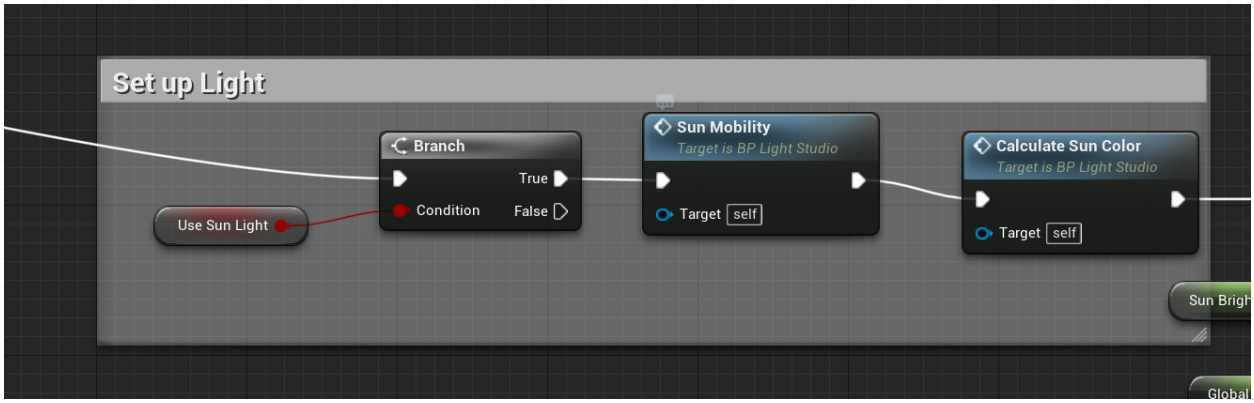

BP Plane selected
BP Plane selected

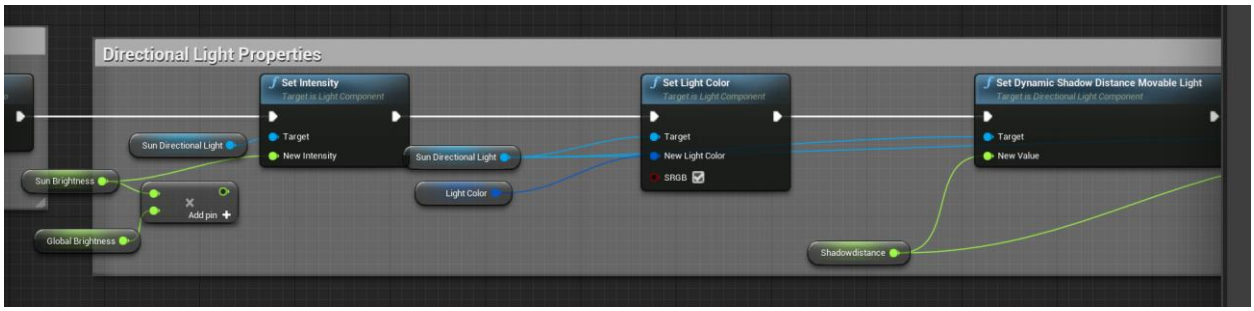

Construction Script
Sequence
Then 0
Then 1
Then 2
Then 3
Then 4
Then 5
Add pin

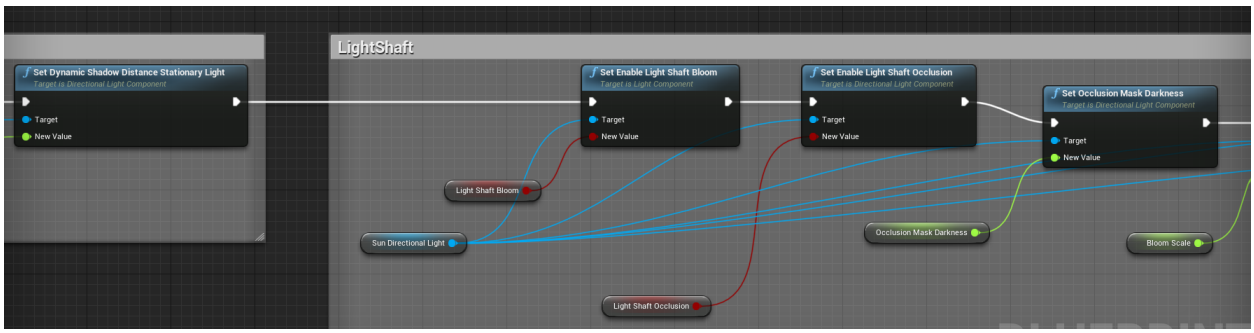

Skybox
SetWorldRotation
Target is Scene Component
Target
New Rotation
X 0,0 Y 0,0 Z 0,0
Skybox
Create Dynamic Material Instance
Parent
Return Value
Optional Name None
Creation Flags
None
SET
Skybox Material

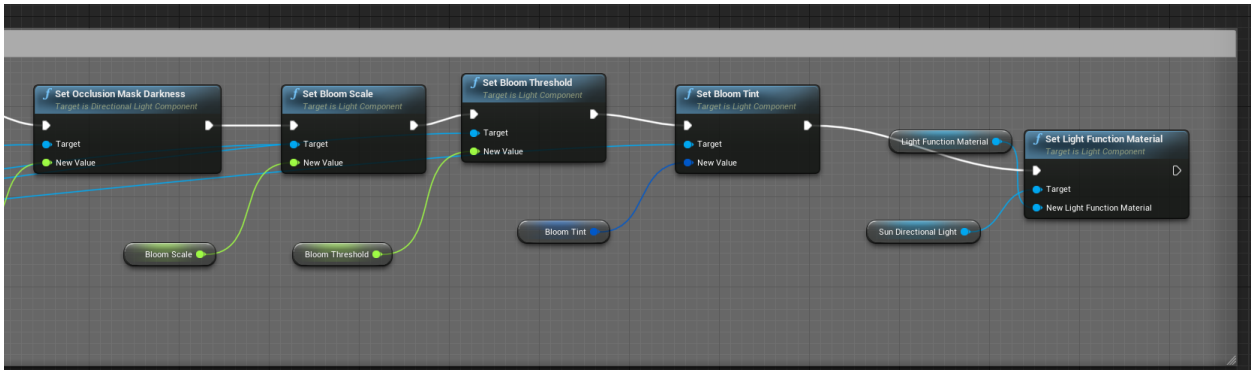

HDRI Parameters
HDRI Parameters
Set Material
Target is Primitive Component
Target
Element Index 0
Material
Skybox
Skybox Material
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name HDRI_Brightness
Value
Skybox Material
HDRI Brightness
Add pin
Global Brightness
Set Vector Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name HDRI_Tint
Value
Skybox Material
HDRI Tint
Set Texture Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name HDRI_Cubemap
Value
Skybox Material
HDRI Cubemap

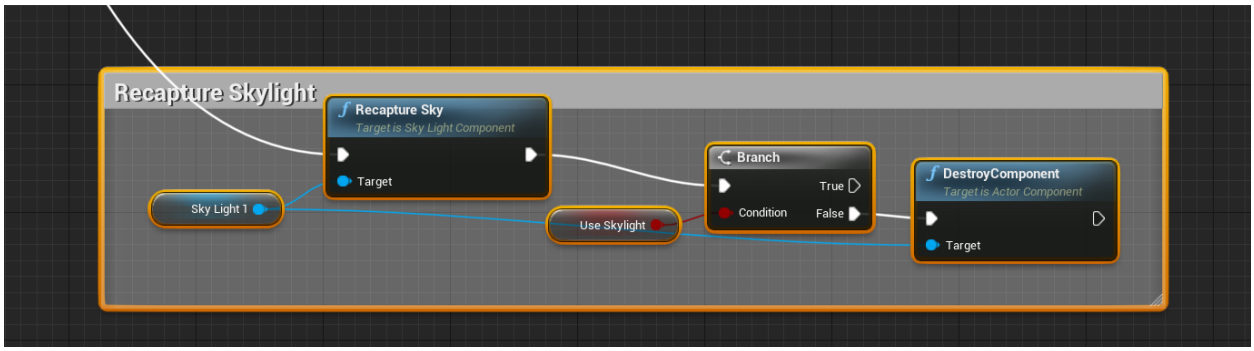

Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name CubemapRotation
Value
Skybox Material
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name AtmosphereHorizon
Value
Skybox Material
Atmosphere Opacity Horizon
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name AtmosphereZenith
Value
Skybox Material
Atmosphere Opacity Zenith
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name HDRI_Contrast
Value
Skybox Material
HDRI Contrast


Set up Light
Branch
True
Condition
False
Use Sun Light
Sun Mobility
Target is BP Light Studio
Target self
Calculate Sun Color
Target is BP Light Studio
Target self
Sun Brigh
Global


Directional Light Properties
Set Intensity
Target is Light Component
Target
New Intensity
Sun Directional Light
Sun Brightness
Add pin
Global Brightness
Sun Directional Light
Light Color
Set Light Color
Target is Light Component
Target
New Light Color
SRGB
Set Dynamic Shadow Distance Movable Light
Target is Directional Light Component
Target
New Value
Shadowdistance


LightShaft
Set Dynamic Shadow Distance Stationary Light
Target is Directional Light Component
Target
New Value
Set Enable Light Shaft Bloom
Target is Light Component
Target
New Value
Set Enable Light Shaft Occlusion
Target is Directional Light Component
Target
New Value
Set Occlusion Mask Darkness
Target is Directional Light Component
Target
New Value
Light Shaft Bloom
Sun Directional Light
Occlusion Mask Darkness
Bloom Scale
Light Shaft Occlusion


Set Occlusion Mask Darkness
Target is Directional Light Component
Target
New Value
Set Bloom Scale
Target is Light Component
Target
New Value
Set Bloom Threshold
Target is Light Component
Target
New Value
Set Bloom Tint
Target is Light Component
Target
New Value
Light Function Material
Set Light Function Material
Target is Light Component
Target
New Light Function Material
Bloom Scale
Bloom Threshold
Bloom Tint
Sun Directional Light


Recapture Skylight
Recapture Sky
Target is Sky Light Component
Target
Sky Light 1
Use Skylight
Branch
True
Condition
False
DestroyComponent
Target is Actor Component
Target

ДОДАТОК Б

```
PRAGMA_DISABLE_DEPRECATION_WARNINGS
PRAGMA_DISABLE_OPTIMIZATION
ABP_ARGameMode_C__pf104685423::ABP_ARGameMode_C__pf104685423(const
FObjectInitializer& ObjectInitializer) : Super()
{

      bpv__DefaultSceneRoot__pf =
CreateDefaultSubobject<USceneComponent>(TEXT("DefaultSceneRoot"));
      RootComponent = bpv__DefaultSceneRoot__pf;
      bpv__DefaultSceneRoot__pf->CreationMethod =
EComponentCreationMethod::Native;
      static TWeakFieldPtr<FProperty> __Local__1{};
      const FProperty* __Local__0 = __Local__1.Get();
      if (nullptr == __Local__0)
      {
            __Local__0 = (UActorComponent::StaticClass())->
FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
            check(__Local__0);
            __Local__1 = __Local__0;
      }
      (((FBoolProperty*)__Local__0)->
SetPropertyValue_InContainer((bpv__DefaultSceneRoot__pf), false,
0));

       DefaultPawnClass = ABP_ARPawn_C__pf104685423::StaticClass();
       ReplaySpectatorPlayerControllerClass =
 ABP_ARPlayerController_C__pf104685423::StaticClass();
      auto& __Local__2 =
 (*(AccessPrivateProperty<EActorUpdateOverlapsMethod >((this),
 AActor::
 __PPO__DefaultUpdateOverlapsMethodDuringLevelStreaming() ));
      __Local__2 = EActorUpdateOverlapsMethod::OnlyUpdateMovable;
 }
 PRAGMA_ENABLE_OPTIMIZATION
 void
 ABP_ARGameMode_C__pf104685423::PostLoadSubobjects(FObjectInstancin
 gGraph* OuterInstanceGraph)
 {
      Super::PostLoadSubobjects(OuterInstanceGraph);
      if(bpv__DefaultSceneRoot__pf)
      {
            bpv__DefaultSceneRoot__pf->CreationMethod =
 EComponentCreationMethod::Native;
      }
 }
```

```
PRAGMA_DISABLE_DEPRECATION_WARNINGS
PRAGMA_DISABLE_OPTIMIZATION
ABP_ARPawn_C__pf104685423::ABP_ARPawn_C__pf104685423(const
FObjectInitializer& ObjectInitializer) : Super(ObjectInitializer)
{

      bpv__DefaultSceneRoot__pf =
CreateDefaultSubobject<USceneComponent>(TEXT("DefaultSceneRoot"));
      bpv__Camera__pf = CreateDefaultSubobject<UCameraComponent>
(TEXT("Camera"));
      RootComponent = bpv__DefaultSceneRoot__pf;
      bpv__DefaultSceneRoot__pf->CreationMethod =
EComponentCreationMethod::Native;
      static TWeakFieldPtr<FProperty> __Local__1{};
      const FProperty* __Local__0 = __Local__1.Get();
      if (nullptr == __Local__0)
      {
            __Local__0 = (UActorComponent::StaticClass())->
FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
            check(__Local__0);
            __Local__1 = __Local__0;
      }
      (((FBoolProperty*)__Local__0)->
SetPropertyValue_InContainer((bpv__DefaultSceneRoot__pf), false,
0));
PRAGMA_ENABLE_OPTIMIZATION
void
ABP_ARPawn_C__pf104685423::PostLoadSubobjects(FObjectInstancingGra
ph* OuterInstanceGraph)
{
      Super::PostLoadSubobjects(OuterInstanceGraph);
      if(bpv__DefaultSceneRoot__pf)
      {
            bpv__DefaultSceneRoot__pf->CreationMethod =
EComponentCreationMethod::Native;
      }
      if(bpv__Camera__pf)
      {
            bpv__Camera__pf->CreationMethod =
EComponentCreationMethod::Native;
      }
}
```

```
PRAGMA_DISABLE_OPTIMIZATION
void ABP_ARPawn_C__pf104685423::
__CustomDynamicClassInitialization(UDynamicClass* InDynamicClass)
{
      ensure(0 == InDynamicClass->
ReferencedConvertedFields.Num());
      ensure(0 == InDynamicClass->MiscConvertedSubobjects.Num());
      ensure(0 == InDynamicClass->DynamicBindingObjects.Num());
      ensure(0 == InDynamicClass->ComponentTemplates.Num());
      ensure(0 == InDynamicClass->Timelines.Num());
      ensure(0 == InDynamicClass->ComponentClassOverrides.Num());
      ensure(nullptr == InDynamicClass->AnimClassImplementation);
      InDynamicClass->AssembleReferenceTokenStream();
      // List of all referenced converted enums
      InDynamicClass->
ReferencedConvertedFields.Add(LoadObject<UEnum>(nullptr,
TEXT("/Game/HandheldAR/Blueprints/GameFramework/ENUM_GestureType.E
NUM_GestureType")));
      FConvertedBlueprintsDependencies::FillUsedAssetsInDynamicCla
ss(InDynamicClass, &__StaticDependencies_DirectlyUsedAssets);
      auto __Local__3 = NewObject<UInputActionDelegateBinding>
(InDynamicClass, UInputActionDelegateBinding::StaticClass(),
TEXT("InputActionDelegateBinding_1"), (EObjectFlags)0x00000000);
      InDynamicClass->DynamicBindingObjects.Add(__Local__3);
      auto __Local__4 = NewObject<UInputAxisDelegateBinding>
(InDynamicClass, UInputAxisDelegateBinding::StaticClass(),
TEXT("InputAxisDelegateBinding_1"), (EObjectFlags)0x00000000);
      InDynamicClass->DynamicBindingObjects.Add(__Local__4);
      auto __Local__5 = NewObject<UInputTouchDelegateBinding>
(InDynamicClass, UInputTouchDelegateBinding::StaticClass(),
TEXT("InputTouchDelegateBinding_1"), (EObjectFlags)0x00000000);
      InDynamicClass->DynamicBindingObjects.Add(__Local__5);
      __Local__3->InputActionDelegateBindings =
TArray<FBlueprintInputActionDelegateBinding> ();
      __Local__3->InputActionDelegateBindings.AddUninitialized(4);
      FBlueprintInputActionDelegateBinding::StaticStruct()->
InitializeStruct(__Local__3->
InputActionDelegateBindings.GetData(), 4);
      auto& __Local__6 = __Local__3->
InputActionDelegateBindings[0];
```

```
      __Local__6.InputActionName = FName(TEXT("TwoFingerAction"));
      __Local__6.InputKeyEvent = EInputEvent::IE_Released;
      __Local__6.FunctionNameToBind =
FName(TEXT("InpActEvt_TwoFingerAction_K2Node_InputActionEvent_
3"));
      auto& __Local__7 = __Local__3->
InputActionDelegateBindings[1];
      __Local__7.InputActionName = FName(TEXT("TwoFingerAction"));
      __Local__7.FunctionNameToBind =
FName(TEXT("InpActEvt_TwoFingerAction_K2Node_InputActionEvent_
2"));
      auto& __Local__8 = __Local__3->
InputActionDelegateBindings[2];
      __Local__8.InputActionName = FName(TEXT("OneFingerAction"));
      __Local__8.FunctionNameToBind =
FName(TEXT("InpActEvt_OneFingerAction_K2Node_InputActionEvent_
0"));
      auto& __Local__9 = __Local__3->
InputActionDelegateBindings[3];
      __Local__9.InputActionName = FName(TEXT("OneFingerAction"));
      __Local__9.InputKeyEvent = EInputEvent::IE_Released;
      __Local__9.FunctionNameToBind =
FName(TEXT("InpActEvt_OneFingerAction_K2Node_InputActionEvent_
1"));
      __Local__4->InputAxisDelegateBindings =
TArray<FBlueprintInputAxisDelegateBinding> ();
      __Local__4->InputAxisDelegateBindings.AddUninitialized(1);
      FBlueprintInputAxisDelegateBinding::StaticStruct()->
InitializeStruct(__Local__4->InputAxisDelegateBindings.GetData(),
1);
      auto& __Local__10 = __Local__4->
InputAxisDelegateBindings[0];
      __Local__10.InputAxisName = FName(TEXT("TwoFingerMapping"));
      __Local__10.bOverrideParentBinding = false;
      __Local__5->InputTouchDelegateBindings =
TArray<FBlueprintInputTouchDelegateBinding> ();
      __Local__5->InputTouchDelegateBindings.AddUninitialized(1);
      FBlueprintInputTouchDelegateBinding::StaticStruct()->
InitializeStruct(__Local__5->InputTouchDelegateBindings.GetData(),
```

```
PRAGMA_ENABLE_OPTIMIZATION
void
ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_
0(int32 bpp__EntryPoint__pf)
{
      check(bpp__EntryPoint__pf == 65);
      // optimized KCST_UnconditionalGoto
      b0l__Temp_struct_Variable__pf =
b0l__K2Node_InputActionEvent_Key_2__pf;
      // optimized KCST_UnconditionalGoto
      bpv__TwoFingerPressed__pf = true;
      bpv__TwoFingersValidationTimer__pf = 0.000000;
      bpf__GetTouchLocation__pf(/*out*/
b0l__CallFunc_GetTouchLocation_Location_1__pf);
      bpv__PreviousFingerPosition__pf =
b0l__CallFunc_GetTouchLocation_Location_1__pf;
      bpv__StartingFingerPosition__pf =
bpv__PreviousFingerPosition__pf;
      bpf__GetFingersDistance__pf(/*out*/
b0l__CallFunc_GetFingersDistance_Distance__pf);
      bpv__PreviousFingersDistance__pf =
b0l__CallFunc_GetFingersDistance_Distance__pf;
      return; //KCST_EndOfThread
}
void
ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_
1(int32 bpp__EntryPoint__pf)
{
      bool bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf{};
      float bpfv__CallFunc_GetInputAxisValue_ReturnValue__pf{};
      UARTexture* bpfv__CallFunc_GetARTexture_ReturnValue__pf{};
      bool bpfv__CallFunc_IsValid_ReturnValue__pf{};
      bool bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf{};
      float bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf{};
      bool bpfv__CallFunc_BooleanAND_ReturnValue__pf{};
      bool bpfv__CallFunc_Not_PreBool_ReturnValue_2__pf{};
      TArray< int32, TInlineAllocator<8> > __StateStack;
```

```
	int32 __CurrentState = bpp__EntryPoint__pf;
	do
	{
		switch( __CurrentState )
		{
		case 25:
			{

	bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf =
UKismetMathLibrary::Greater_FloatFloat(bpv__ShowPlaneTimer__pf,
1.000000);
				if (!
bpfv__CallFunc_Greater_FloatFloat_ReturnValue__pf)
				{
					__CurrentState = (__StateStack.Num() >
0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
					break;
				}
			}
		case 26:
			{
				bpv__ShowPlaneTimer__pf = 0.000000;
			}
		case 27:
			{
				bpf__UpdatePlaneCandidate__pf();
				__CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
				break;
			}
		case 31:
			{
				__CurrentState = 32;
				break;
			}
		case 32:
			{
				bpv__DT__pf =
b01__K2Node_Event_DeltaSeconds__pf;
			}
```

```
            case 33:
                  {
                        __StateStack.Push(45);
                        __StateStack.Push(43);
                        __StateStack.Push(39);
                  }
            case 34:
                  {
                        bpf__ResetRecognisedGesture__pf();
                  }
            case 35:
                  {
                        bpf__OneFingerGestureRecognition__pf();
                  }
            case 36:
                  {
                        bpf__TwoFingersGestureRecognition__pf();
                  }
            case 37:
                  {
                        if (!bpv__GestureRecognised__pf)
                        {
                              __CurrentState = (__StateStack.Num() >
0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                              break;
                        }
                  }
            case 38:
                  {
                        b0l__K2Node_SwitchEnum_CmpSuccess_1__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>
(bpv__CurrentTransformation__pf), static_cast<uint8>
(E__ENUM_GestureType__pf::NewEnumerator0));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_1
__pf)
                        {
                              __CurrentState = 47;
                              break;
                        }
```

```
                }
            case 39:
                {
                    bpfv__CallFunc_Not_PreBool_ReturnValue_2__pf
= UKismetMathLibrary::Not_PreBool(bpv__ScanningIsDone__pf);
                    if (!bpfv__CallFunc_Not_PreBool_ReturnValue_
2__pf)
                    {
                        __CurrentState = (__StateStack.Num() >
0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
                    }
                }
            case 40:
                {
                    bpfv__CallFunc_IsValid_ReturnValue__pf =
UKismetSystemLibrary::IsValid(bpv__CameraDepthTexture__pf);
                    if (!bpfv__CallFunc_IsValid_ReturnValue__pf)
                    {
                        __CurrentState = 41;
                        break;
                    }
                    __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                    break;
                }
            case 41:
                {
                    bpfv__CallFunc_GetARTexture_ReturnValue__pf
=
UARBlueprintLibrary::GetARTexture(EARTextureType::SceneDepthMap);
                }
            case 42:
                {
                    bpv__CameraDepthTexture__pf =
bpfv__CallFunc_GetARTexture_ReturnValue__pf;
                    __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                    break;
                }
```

```
            case 43:
                  {
                        bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf
= UKismetMathLibrary::Not_PreBool(bpv__PlaneIsSelected__pf);
                        bpfv__CallFunc_BooleanAND_ReturnValue__pf =
UKismetMathLibrary::BooleanAND(bpv__ScanningIsDone__pf,
bpfv__CallFunc_Not_PreBool_ReturnValue_1__pf);
                        if (!
bpfv__CallFunc_BooleanAND_ReturnValue__pf)
                        {
                              __CurrentState = (__StateStack.Num() >
0) ? __StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                              break;
                        }
                  }
            case 44:
                  {

      bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf =
UKismetMathLibrary::Add_FloatFloat(bpv__DT__pf,
bpv__ShowPlaneTimer__pf);
                        bpv__ShowPlaneTimer__pf =
bpfv__CallFunc_Add_FloatFloat_ReturnValue__pf;
                        __CurrentState = 25;
                        break;
                  }
            case 45:
                  {
                        bpf__FeedLightEstimate__pf();
                  }
            case 46:
                  {
                        bpf__GetTouchLocation__pf(/*out*/
b01__CallFunc_GetTouchLocation_Location_6__pf);
                        bpv__PreviousFingerPosition__pf =
b01__CallFunc_GetTouchLocation_Location_6__pf;
                        __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
```

```
            case 47:
                  {
                        bpf__GetTouchLocation__pf(/*out*/
b0l__CallFunc_GetTouchLocation_Location_2__pf);
                        if(::IsValid(bpv__PlaceableObject__pf))
                        {

      FUnconvertedWrapper__ABP_Placeable_C__pf231924168(bpv__Place
ableObject__pf).bpf__SetPlaceablePosition__pf(b0l__CallFunc_GetTou
chLocation_Location_2__pf, bpv__SnappingTransform__pf,
bpv__ShowHUDInteraction__pf);
                        }
                        __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
                  }
            case 48:
                  {
                        bpf__GetFingersDirection__pf(/*out*/
b0l__CallFunc_GetFingersDirection_Direction__pf);
                        bpf__GetTouchLocation__pf(/*out*/
b0l__CallFunc_GetTouchLocation_Location_4__pf);
                        if(::IsValid(bpv__PlaceableObject__pf))
                        {

      FUnconvertedWrapper__ABP_Placeable_C__pf231924168(bpv__Place
ableObject__pf).bpf__SetPlaceableRotation__pf(b0l__CallFunc_GetFin
gersDirection_Direction__pf,
b0l__CallFunc_GetTouchLocation_Location_4__pf,
bpv__SnappingTransform__pf, bpv__ShowHUDInteraction__pf);
                        }
                        __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
                  }
```

```
            case 49:
                  {

      bpfv__CallFunc_GetInputAxisValue_ReturnValue__pf =
AActor::GetInputAxisValue(FName(TEXT("TwoFingerMapping")));
                        if(::IsValid(bpv__PlaceableObject__pf))
                        {

      FUnconvertedWrapper__ABP_Placeable_C__pf231924168(bpv__Place
ableObject__pf).bpf__SetPlaceableScale__pf(bpfv__CallFunc_GetInput
AxisValue_ReturnValue__pf, bpv__SnappingTransform__pf,
bpv__ShowHUDInteraction__pf);
                        }
                        __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
                  }
            default:
                  check(false); // Invalid state
                  break;
            }
      } while( __CurrentState != -1 );
}
void
ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_
2(int32 bpp__EntryPoint__pf)
{
      APlayerController*
bpfv__CallFunc_GetPlayerController_ReturnValue__pf{};
      UUserWidget* bpfv__CallFunc_Create_ReturnValue__pf{};
      check(bpp__EntryPoint__pf == 64);
      // optimized KCST_UnconditionalGoto
      bpfv__CallFunc_GetPlayerController_ReturnValue__pf =
UGameplayStatics::GetPlayerController(this, 0);
      bpfv__CallFunc_Create_ReturnValue__pf =
UWidgetBlueprintLibrary::Create(this, CastChecked<UClass>
(CastChecked<UDynamicClass>
(ABP_ARPawn_C__pf104685423::StaticClass())->UsedAssets[1],
ECastCheckedType::NullAllowed),
```

```
bpfv__CallFunc_GetPlayerController_ReturnValue__pf);
      bpv__MainMenu__pf = bpfv__CallFunc_Create_ReturnValue__pf;
      if(::IsValid(bpv__MainMenu__pf))
      {
            bpv__MainMenu__pf->UUserWidget::AddToViewport(0);
      }
      return; //KCST_EndOfThread
}
void
ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_
3(int32 bpp__EntryPoint__pf)
{
      bool bpfv__CallFunc_WorldHitTest_ReturnValue__pf{};
      check(bpp__EntryPoint__pf == 62);
      // optimized KCST_UnconditionalGoto
      b0l__Temp_struct_Variable_1__pf =
b0l__K2Node_InputActionEvent_Key__pf;
      // optimized KCST_UnconditionalGoto
      bpf__GetTouchLocation__pf(/*out*/
b0l__CallFunc_GetTouchLocation_Location__pf);
      bpfv__CallFunc_WorldHitTest_ReturnValue__pf =
bpf__WorldHitTest__pf(b0l__CallFunc_GetTouchLocation_Location__pf,
/*out*/ b0l__CallFunc_WorldHitTest_HitInfo__pf);
      UGameplayStatics::BreakHitResult(b0l__CallFunc_WorldHitTest_
HitInfo__pf, /*out*/

void
ABP_ARPawn_C__pf104685423::bpf__ExecuteUbergraph_BP_ARPawn__pf_
4(int32 bpp__EntryPoint__pf)
{
      check(bpp__EntryPoint__pf == 55);
      // optimized KCST_UnconditionalGoto
      b0l__Temp_struct_Variable_1__pf =
b0l__K2Node_InputActionEvent_Key_1__pf;
      // optimized KCST_UnconditionalGoto
      bpv__OneFingerPressed__pf = false;
      return; //KCST_EndOfThread
}
```

```
Super(ObjectInitializer)
{

      auto __Local__0 = CastChecked<USceneComponent>(this->
GetDefaultSubobjectByName(TEXT("TransformComponent0")),
ECastCheckedType::NullAllowed);
      if(__Local__0)
      {
            // --- Default subobject 'TransformComponent0' //
            static TWeakFieldPtr<FProperty> __Local__2{};
            const FProperty* __Local__1 = __Local__2.Get();
            if (nullptr == __Local__1)
            {
                  __Local__1 = (UActorComponent::StaticClass())->
FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__1);
                  __Local__2 = __Local__1;
            }
            (((FBoolProperty*)__Local__1)->
SetPropertyValue_InContainer((__Local__0), false, 0));
            // --- END default subobject 'TransformComponent0' //
      }
      InputYawScale = 2.500000f;
      InputPitchScale = -2.500000f;
      InputRollScale = 1.000000f;
      ForceFeedbackScale = 1.000000f;
      auto& __Local__3 =
(*(AccessPrivateProperty<USceneComponent*>((this), AController::
__PPO__TransformComponent() )));
      __Local__3 = __Local__0;
      auto& __Local__4 =
(*(AccessPrivateProperty<EActorUpdateOverlapsMethod >((this),
AActor::
```

```
__PPO__DefaultUpdateOverlapsMethodDuringLevelStreaming() ));
      __Local__4 = EActorUpdateOverlapsMethod::OnlyUpdateMovable;
}
PRAGMA_ENABLE_OPTIMIZATION
void
ABP_ARPlayerController_C__pf104685423::PostLoadSubobjects(FObjectI
nstancingGraph* OuterInstanceGraph)
{
      Super::PostLoadSubobjects(OuterInstanceGraph);
}
PRAGMA_DISABLE_OPTIMIZATION
void ABP_ARPlayerController_C__pf104685423::
__CustomDynamicClassInitialization(UDynamicClass* InDynamicClass)
{
      ensure(0 == InDynamicClass->
ReferencedConvertedFields.Num());
      ensure(0 == InDynamicClass->MiscConvertedSubobjects.Num());
      ensure(0 == InDynamicClass->DynamicBindingObjects.Num());
      ensure(0 == InDynamicClass->ComponentTemplates.Num());
      ensure(0 == InDynamicClass->Timelines.Num());
      ensure(0 == InDynamicClass->ComponentClassOverrides.Num());
      ensure(nullptr == InDynamicClass->AnimClassImplementation);
      InDynamicClass->AssembleReferenceTokenStream();
      FConvertedBlueprintsDependencies::FillUsedAssetsInDynamicCla
ss(InDynamicClass, &__StaticDependencies_DirectlyUsedAssets);
      auto __Local__5 = NewObject<USceneComponent>(InDynamicClass,
USceneComponent::StaticClass(),
TEXT("DefaultSceneRoot_GEN_VARIABLE"), (EObjectFlags)0x00280029);
      InDynamicClass->ComponentTemplates.Add(__Local__5);
      static TWeakFieldPtr<FProperty> __Local__7{};
      const FProperty* __Local__6 = __Local__7.Get();
      if (nullptr == __Local__6)
      {
            __Local__6 = (UActorComponent::StaticClass())->
FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
            check(__Local__6);
            __Local__7 = __Local__6;
      }
```

```
	(((FBoolProperty*)__Local__6)->
SetPropertyValue_InContainer((__Local__5), false, 0));
}
PRAGMA_ENABLE_OPTIMIZATION
PRAGMA_DISABLE_OPTIMIZATION
void ABP_ARPlayerController_C__pf104685423::
__StaticDependencies_DirectlyUsedAssets(TArray<FBlueprintDependenc
yData>& AssetsToLoad)
{
}
PRAGMA_ENABLE_OPTIMIZATION
PRAGMA_DISABLE_OPTIMIZATION
void ABP_ARPlayerController_C__pf104685423::
__StaticDependenciesAssets(TArray<FBlueprintDependencyData>&
AssetsToLoad)
{
	__StaticDependencies_DirectlyUsedAssets(AssetsToLoad);
	const FCompactBlueprintDependencyData
LocCompactBlueprintDependencyData[] =
	{
		{7, FBlueprintDependencyType(true, false, false,
false), FBlueprintDependencyType(false, false, false, false)},  //
Class /Script/Engine.SceneComponent
		{8, FBlueprintDependencyType(true, false, false,
false), FBlueprintDependencyType(false, false, false, false)},  //
Class /Script/Engine.PlayerController
		{9, FBlueprintDependencyType(false, true, false,
false), FBlueprintDependencyType(false, false, false, false)},  //
Class /Script/Engine.CheatManager
	};
	for(const FCompactBlueprintDependencyData& CompactData :
LocCompactBlueprintDependencyData)
	{

	AssetsToLoad.Add(FBlueprintDependencyData(F__NativeDependenc
ies::Get(CompactData.ObjectRefIndex), CompactData));
	}
}
```