МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Інсталятор та доповнення для інтеграції CAD Revit та Unreal Engune 4»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шутилєва О.В**

Студента групи ІН – 62 **Мальованик К.М**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-62 спеціальності «Інформатика» денної форми навчання Мальованика Костянтина Михайловича.

Тема: "**Інсталятор та доповнення для інтеграції CAD Revit та Unreal Engune 4**"

Затверджена наказом по СумДУ
№_____ от _____2020 р.

Зміст пояснювальної записки: 1) інформаційний огляд створення додатку для Revit та постановка завдання; 2) аналіз та вибір методів вирішення поставленого завдання; 3) опис та аналіз результатів роботи.

Дата видачі завдання "_____"_____2020 р
.
Керівник випускної роботи _____ Шутилєва О.В.

Завдання прийняв до виконання _____ Мальованик К.М.

РЕФЕРАТ

Записка: 39 стор., 15 рис., 3 додаток, 7 джерел.

Об’єкт дослідження – додаток для імпорту моделей з Revit до ігрового двигуна Unreal Engine 4.

Мета роботи – дослідження інтеграції ігрового рушія Unreal Engine 4 та Revit і розробка додатку для моделей з Revit (2018,2019,2020).

Предметна область – Імпорт/експорт 3D-моделей.

Результати – Розроблено інсталятор, який встановлює додаток для експорту 3D-моделей з Revit із використанням .NET Framework, WIX Toolset, C++, Blueprint C++.

ІНСТАЛЯТОР, CAD REVIT, UNREAL ENGUNE 4

ЗМІСТ

ВСТУП

У сучасному світі широко росповсюджене використання об’ємної графіки, важко уявити сферу виробництва, де на етапі конструювання вона не використовується. На кожному етапі створення продукту, будь це нескладний механізм або складна контструкція ракетного двигуна, орієнтуються на багатогранний макет. Він являє собою багатовекторне креслення, що має не тільки номінальну висоту, довжину і ширину, але і візуальне втілення.

3D- графіка – поняття яке широко використовується в сучасному житті. Її використовують для створення максимально реалістичної моделі, наприклад, при створенні міської архітектури або ланшафтів з мінімальними витратами. С цим поняттям часто вживають такі слова: звук, зображення, шутер, шоу, принтер і так далі – варіантів маса. Основний зміст залишається, при вживанні цього методу відбувається перехід з схематичного, однолінійного простору в реалістичне. Ця здатність «одухотворяти» неживе ставиться в основу багатьох починань.

Сучасні технології 3D змушують глядача зануриться в «реальність»: відчути себе на полі під час футбольного матчу, побачити вибухи, перестрілки в бойовиках, зануритись на дно світового океану в фільмах BBC про живу природу і таке інше – все це стало можливим в наші дні.

Воно широко застосовується в таких галузях, як реклама та маркетинг, промисловысть, Коп.ігри, кінематограф, архітектура та дизайн інтер'єрів, анімація. Галузі в яких використовуеться 3D, користуються високим попитом в сучасному житті, отже обрана тема є актуальною

Практичне завдання проекту полягає у наданні можливості досконально досліджувати 3D-модель, користувач може зануритися у власноруч створену модель та перевірити, правильно вона була створена чи є деякі нюанси, які необхідно виправити.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сутність 3D-графіки та 3D- моделювання

3D-графіка чи тривимірна графіка (від 3 dimensions – 3 виміри) – це розділ комп’ютерної графіки, набір інструментів (програмних або аппаратних), призначених для зображення об’їемних фігур. 3D зображення на площині навідміну від двовимірнго, включає побудову геометрічної проекції тривимірної моделі (сцени) на будь яку площину (екран комп’ютера,зображення від проєктора і так далі). Створена модель може відповідати об’єктам реального світу (будівлі, потяги, урагани і т.д), так і бути повністю абстрактною (проекція чотиривимірного фрактала) [1].

Моделювання 3D фігур у комп'ютерній графіці – це процес розробки математичного представлення, будь-якої поверхні об'єкта з використанням спеціального програмного забеспечення. Продукт моделювання – 3D-модель. Зазвичай вона відображається у вигляді коду (програмного) або відображається у в'юпорті чи вівері у вигляді 3D-модель, а також може бути у вигляді двовимірного зображення, що створюється за допомогою процесу рендерингу. 3D-модель може створюватись за допомогою 3D-сканнера, вручну чи автоматично.

У зв'язку зі створенням 3D-принтерів та 3D-дисплеїв, тривимірна графіка не завжди включає в себе, проектування на площині.

Категорії об'єктів на сцені:

- джерела світла для задання напрямку, інтенсивності, спектра освітлення, додавання об’ємності сцени за рахунок тіней, тощо;
- матеріали (інформація щодо візуальних властивостей моделі, наприклад колір стін та відбиваюча / заломлююча здатність вікон);
- віртуальні камери;
- дії і сили (налаштування динамічних взаємодій об'єктів, застосовується для створення анімації);

- додаткові ефекти (об'єкти, що імітують атмосферні явища: туман, хмари, полум'я і ін.).

Мета тривимірного моделювання – це описати об'єкти, розмістити їх на сцені із допомогою геометричних перетворень відповідно до вимог технічного завдання.

1.1.1 Методи промислового проектування

Головними користувачами є інженери, електрики, будівельники, працівники дорожніх служб – фахівці технічнго напряму. Їх інструмент – це твердотільні або порожнисті конструкції, що володіють математично точними параметрами, розрахунковими даними і реальної спрямованістю на роботу. Тому, особливо важливим для цієї категорії користувачів є не зовнішній вигляд моделі, а можливість застосування формул, роботи з ними, зрізові креслення, графіка, а також перевірка всього механізму на будь-якому етапі розробки. Таким чином, мета проектування - це не тільки візуалізація об'єкта, а більшою мірою,точна математична інформація про нього, яка в подальшому допоможе уникнути помилок в математичнх розрахунках і проектуванні.

Робота в CAD (україномовна абревіатура – САПР) передбачає профільну освіту. Вона ефективна, коли фахівець не тільки бачить образ, але знає матеріал, з якого буде створений макет, особливості використання матеріалу та багато інших нюансів. Тому програми розряду ZWCAD з широким спектром дій і великою кількістю інструментів, компанії замовляють комплектами, щоб забезпечити ПО весь відділ. Їх же встановлюють на комп'ютери студентам технічних і архітектурних ВНЗ, щоб майбутні фахівці відразу конструювали в зручному і багатофункціональному середовищі. Орієнтуючись не тільки на індивідуального покупця, але і на масові поставки, ZWSOFT розробив гнучку політику ліцензування і істотно знизив ціни на серійні закупівлі.

При роботі в таких системах автоматизованого проектування інженер отримує електронно-геометричну модель. Список дій, що можна робити з нею, в об'ємному 3D моделюванні:

- Виконати креслення будь-якого зрізу, в будь-якому зображенні під обраним кутом,таким чином необхідний один макет замість маси розрізнених графіків. Тому з одним файлом, використовуючи різні шари, можуть одночасно працювати різні фахівці, і навіть різні відділи.
- Підігнати параметри всього вироби, змінивши введення однієї цієї величини.
- Проводити розрахунки будь-якого показника або коефіцієнта. Як в статичному положенні, так і в прогнозованому русі.
- Написати пакет для комп'ютерного управління верстатом або іншим технічним обладнанням.
- Використовувати 3D-принтер і відтворити об'ємну модель для презентації або показового конструювання.
- Зробити рендеринг, тобто провести візуалізацію макета – накласти кілька шарів текстури, щоб представити фінальний зовнішній вигляд [2].

1.1.2 Поняття «Plug-In»

Плагін (англ. plug-in – підключати) – додаток, незалежно скомпільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення або використання її можливостей. Належить до загального програмного класу додатків. Плагіни, зазвичай, виконуються у вигляді динамічних бібліотек.

Плагіном до графічного редактора може бути фільтр, що якимсь чином змінює зображення, палітру, дозволяє роботу з додатковими форматами та ін.

Часто у вигляді плагінів виконується підтримка форматів файлів, наприклад, для звукових і відеопрогравачів, наборів офісних застосунків, програм обробки звуку і графіки. У програмах обробки звуку плагіни виконують обробку і створення звукових ефектів. Деякі плагіни змінюють технічні характеристики звуку: глибину квантизації, частоту дискретизації та ін.

Основна програма надає сервіси, які плагін може використовувати. До них відноситься надана плагіну можливість зареєструвати себе в основному додатку,

а також протокол обміну даними з іншими плагінами. Плагіни є залежними від сервісів, що надаються основним додатком, і переважно окремо не використовуються. На противагу їм, основний додаток незалежно оперує плагінами, надаючи кінцевим користувачам можливість динамічно додавати й оновлювати плагіни без необхідності внесення змін в основний додаток [3].

1.2 Поняття та сутність WalkThrough-3D додатка

WalkThrough-3D – це додаток, який надає змогу користувачеві досліджувати 3D-модель. Загалом такий додаток вбудовують в іншу програму, яка працює з розробкою моделей в 3D форматі. Тому розробнику потрібно реалізувати експорт моделі із основної програми та імпорт в іншу, частіше всього якою виступає ігровий рушій. Повноцінну програму можна вважати плагіном, який генерує окремий виконавчий пакет для перегляду експортованої моделі в тривимірному поданні, для роботи якого не потрібно ні програмного забезпечення CAD, ні установок рушія.

Основною програмою, в яку інсталюється плагін є Revit Autodesk, а взаємодія з експортом реалізується на ігровому рушії Unreal Engine. Цей движок був вибраний виходячи з таких пунктів:

- Такий плагін вже був реалізований на ігровому движку Unity.
- Unreal Engine безкоштовний.
- Unreal Engine повністю пристосований до роботи з 3D-об’єктами.

Autodesk Revit розробляє програмне забезпечення для інформаційного моделювання для архітекторів, ландшафтних архітекторів, інженерів-будівельників, інженерів-проектувальників та підрядників. Revit – це 4D BIM, здатний планувати та відстежувати різні етапи життєвого циклу будівлі, від концепції до будівництва та пізнього обслуговування та / або знесення.

Unreal Engine – це ігровий рушій, розроблений Epic Games, який успішно використовується в багатьох ігрових жанрах.

Оскільки кінцевий продукт розрахований на широке використання, то потрібно також реалізувати підтримку старих версій Autodesk Revit, запакувати за допомогою WixInstaller та створити CustomActions.

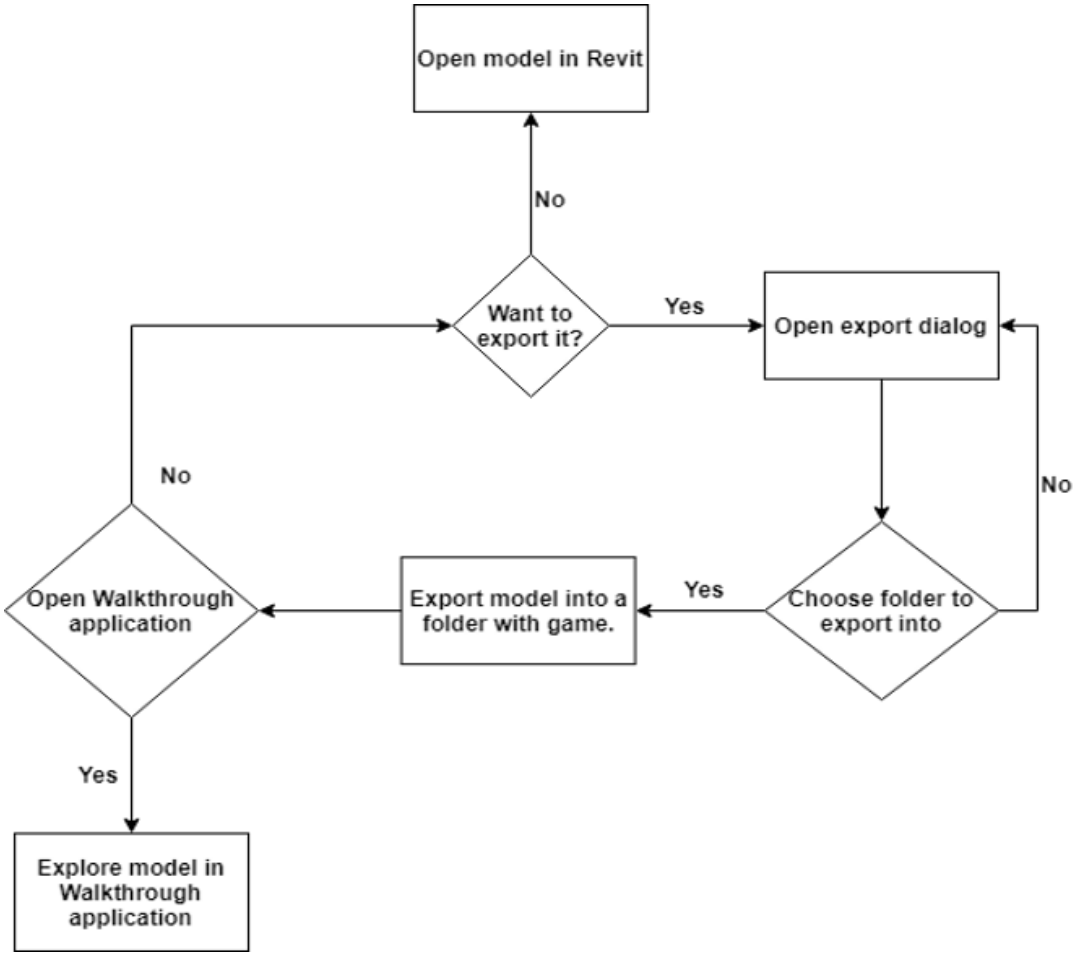

Рисунок 1.1 – Схема роботи

1.2.1 Експорт існуючих форматів

Revit надає можливість як експорту, так і імпорту існуючих 3D-форматів, наприклад: DWG, FBX, OBJ, IFC і т.д. Є два способи, щоб отримати такий файл:

1. Використання панелі програми.

2. Використання Revit Platform API.

Revit API виокремлює таких простір імен Autodesk.Revit.DB, в якому зберігається інформація про активний 3D-вид проекту і надає до нього доступ за допомогою класу Document. Цей клас має методи Export(String, String, ExportOptions), де ExportOptions залежить від типу бажаного 3D-формату.

Вже готовий файл буде містити в собі інформацію про 3D-mesh(3D-сітка),

assets (матеріали) та трансформацію об’єктів.

Переваги:

- Не потрібно в ручну створювати 3D-mesh.
- Не потрібно піклуватись про афінний простір.
- Підтримка багатьма програмами.

Недоліки:

- Немає змоги змінити алгоритм експорту.
- Можлива недостача певних матеріалів.
- Полігони можуть дещо відрізнятись.
- Відсутність змоги внесення змін до 3D-mesh.
- Імовірність відсутності підтримки ігровим движком.

1.2.2 Користувацький експортер

Revit API має ICustomExporter інтерфейс, який має ще двох спадкоємців та містить методи, які є загальними для обох інтерфейсів: IPhotoRenderContext та IModelExportContext.

Екземпляр класу, який реалізує головний інтерфейс, передається як параметр конструктора CustomExporter. А метод експорту цього класу запускає стандартний процес візуалізації або експортування в Revit, але замість того, щоб відображати результат на екрані або принтері, висновок направляється через заданий контекст, який обробляє геометричну, а також не геометричну інформацію.

Методи ICustomExporter інтерфейсу:

- Finish() – викликається в самому кінці процесу експортування, після обробки всіх об'єктів (або після скасування процесу).
- IsCanceled() – запитується на початку кожного елемента.
- OnElementBegin() – ей метод позначає початок експорту елемента.
- OnElementEnd() – позначає кінець експортованого елемента.
- OnFaceBegin() – позначає початок експорту особи.

- OnFaceEnd() – позначає кінець поточного експортованого обличчя.
- OnInstanceBegin() – позначає початок експорту сімейства.
- OnInstanceEnd() – позначає кінець експорту сімейства.
- OnLight() – позначає початок експорту світла, яке включено для візуалізації.
- OnLinkBegin() – позначає початок експорту посилання.
- OnLinkEnd() – позначає кінець експортованого екземпляра посилання.
- OnMaterial() – позначає зміну матеріалу.
- OnPolymesh() – викликається, коли виводиться мозаїчна полімеша 3D-обличчя.
- OnRPC() – позначає початок експорту об'єкта RPC.
- OnViewBegin() – позначає початок експорту 3D-зображення.
- OnViewEnd() – позначає кінець експорту 3D-зображення.
- Start() – викликається на самому початку процесу експорту, ще до того, як відправляється перший об'єкт моделі.

1.2.3 Огляд способів імпорту 3D-моделі в Unreal Engine

Ігровий рушій має декілька видів мешів: Skeletal, Static, Procedural.

Скелетні сітки складаються з двох частин: набір полігонів, що складаються з поверхні скелетної сітки, і ієрархічний набір взаємопов'язаних кісток, які можна використовувати для анімації вершин полігонів. Часто використовуються в Unreal Engine для представлення символів або інших анімаційних об'єктів. 3D-моделі, оснащення та анімації створюються у зовнішньому додатку моделювання та анімації (3DSMax, Maya, Softimage і т.д.), а потім імпортуються в Unreal Engine і зберігаються в пакунках за допомогою переглядача вмісту Unreal Editor.

Статична сітка – це частина геометрії, яка складається з набору полігонів, які можна кешувати у відеопам'яті та виводити відеокартою. Це дозволяє їм бути ефективними, тобто вони можуть бути набагато складнішими, ніж інші типи

геометрії, такі як кисті. Оскільки вони кешуються у відеопам'яті, статичні сітки можуть бути перекладені, повернені та масштабовані, але вони не можуть анімувати свої вершини.

Статичні сітки – це основна одиниця, що використовується для створення світової геометрії для рівнів, створених у Unreal Engine. Це 3D-моделі, створені у зовнішніх додатках для моделювання (наприклад, 3dsMax, Maya, Softimage і т. д.), які імпортуються в редактор Unreal через переглядач вмісту, зберігаються в пакунках, а потім використовуються різними способами для створення елементів, що підлягають відтворенню. Переважна більшість будь-якої карти в грі, створеній за допомогою Unreal, буде складатися зі статичних сіток, як правило, у формі акторів Static Mesh. Інші види використання статичних сіток призначені для створення рушіїв, таких як двері або підйомники, об'єкти фізики твердого тіла, прикраси листя і місцевості, процедурно створені будівлі, цілі гри та багато інших візуальних елементів.

Процедурні сітки – це сітки, які підлягають ручному налаштуванні. Розробник може задати власну геометрію сітки трикутника. А також змінювати певні субоб’єкти одного об’єкту, коли в статичних сітках об’єкт виступає єдиною цілою формою. Процедурне моделювання дозволяє редагувати сітки в недеструктурний спосіб. Можна виконувати топологічні зміни до сітки, яка може бути змінена, підігнана і навіть анімована. Можна легко повернути або змінити операції, зберігаючи решту операцій незмінними.

Процедурна система моделювання працює подібно до системи деформацій. Сітчастий шар містить базову сітку, яка процедурно модифікована. Це означає, що, коли редагується сітку, операції шаруються поверх базового шару. Якщо змінюється базовий шар, операції, накладені на вершину, повторно оцінюються для створення результату. При процедурному моделюванні можна змінювати будь-які операції сітки, не торкаючись решти списку.

Матеріал – це актив, який може бути застосований до сітки для керування візуальним виглядом сцени. На високому рівні, мабуть, найлегше думати про матеріал як про "фарбу", що застосовується до об'єкта. Але навіть це може трохи

ввести в оману, оскільки матеріал буквально визначає тип поверхні, з якої видається об'єкт. Надається змога визначити його колір, як він блискучий, чи можна побачити через об'єкт, і багато іншого.

У більш технічних термінах, коли світло від сцени потрапляє на поверхню, матеріал використовується для обчислення того, як це світло взаємодіє з цією поверхнею. Ці розрахунки виконуються з використанням вхідних даних, які надходять до матеріалу з різних зображень (текстур) і математичних виразів, а також з різних параметрів властивостей, властивих самому матеріалу.

Текстури – це зображення, які використовуються в матеріалах. Вони відображаються на поверхні, до яких застосовується матеріал. Або текстури застосовуються безпосередньо – наприклад, для текстур базового кольору - або значення пікселів текстури (або текселів) використовуються в матеріалі як маски або для інших обчислень. У деяких випадках текстури також можуть використовуватися безпосередньо, поза матеріалами, наприклад, для малювання до HUD. Здебільшого, текстури створюються зовні в межах програми для редагування зображень, такі як Photoshop, а потім імпортуються в редактор Unreal через вміст браузера. Однак деякі текстури генеруються в Unreal, такі як Render Textures. Вони зазвичай беруть деяку інформацію зі сцени і роблять її текстурою, яка буде використовуватися в іншому місці.

Один матеріал може використовувати декілька текстур, які всі відбираються і застосовуються для різних цілей. Наприклад, простий матеріал може мати базову кольорову текстуру, дзеркальну текстуру і нормальну текстуру карти. Крім того, може бути текстура для випромінювання і шорсткості, збережена в альфа-каналах однієї або більше з цих текстур.

1.2.4 Імпорт існуючих форматів

Для того, щоб переконатися, що перехід від програмного забезпечення для 3D-моделювання, будь то Maya, 3ds Max або інша програма моделювання, є кілька речей, які потрібно перевірити. По-перше, дуже корисно запам'ятати при моделюванні і перед експортом те, що вимірювання, що використовується в UE,

є Unreal Units і що 1 Unreal Unit дорівнює 1 сантиметру.

Крім того, тільки певні типи файлів, які можна імпортувати до UE, FBX є форматом файлу, рекомендованим для 3D-об'єктів. Також треба переконатись, що будь-які текстури та матеріали, застосовані до статичної сітки, підтримуються типи файлів.

Можна імпортувати сітку одним з двох простих способів. Перший спосіб – використати кнопку на панелі Unreal Editor. Варто відзначити, що при імпорту сітки потрібно вибрати її тип: скелетна чи статична. Процедурна сітка доступна лише при використанні C++ коду.

Другий спосіб за допомогою Visual Studio використовуючи, наприклад, FFbxImporter. Ця бібліотека допомагає створити актив через C++ код.

Переваги:

- Автоматичний імпорт 3D-моделі.
- Легкість застосування.

Недоліки:

- Імпортує лише як статичну/скелетну сітку.
- Неможливість ручного налаштування чи зміни параметрів, таких як матеріал актива.

1.2.5 Побудова 3D-моделі за допомогою допоміжних точок

Побудову 3D-моделі за допомогою допоміжних точок надає лише процедурна сітка. Для роботи з нею використовують Unreal Engine C++ API. Щоб створити такий меш потрібно виконати два пункти:

- Підключити ProceduralMeshComponent модуль.
- Використати метод:

```
CreateMeshSection
(    int32 SectionIndex,
   const TArray < FVector >& Vertices,
   const TArray < int32 >& Triangles,
   const TArray < FVector >& Normals,
```

const TArray < FVector2D >& UV0,

const TArray < FColor >& VertexColor,

const TArray < FProcMeshTangent >& Tangents,

bool bCreateCollision), де SectionIndex індекс секції меша, bCreateCollision булева змінна, яка відповідає за колізію секції.

Єдині труднощі можуть полягати в наданні векторів вершин, нормалей, UV-координат та трикутників.

Цей модуль дозволяє вносити такі налаштування до мешу:

- Контроль фізичного кукінгу з ігрового потоку.
- Контроль розгляду комплексної (Per poly) геометрії як «просте» зіткнення.
- Очистку вибірково/всіх секцій мешу.
- Додавання простої колізії до певного компоненту.
- Заміна на нову геометрію секції.
- Оновлення вибіркової секції мешу.
- Можливість витягти матеріал, застосований до конкретної поверхні сітки.

1.3 Постановка задачі

Головним завданням є розробка плагіну для Autodesk Revit, який експортує 3D-моделі та імпортує її до ігрового рушія Unreal Engine. Під час розробки додатку потрібно виконати наступне:

1. Реалізувати користувацький інтерфейс експорту з підтримкою різних версій Revit, запис бінарного файлу з 3D-моделлю та створити DLL бібліотеки плагіну.

2. Створити файл типу .addin та його маніфест.

3. Розробити форму для користувача з кнопками вибору місця збереження додатку, запуску, інформації.

4. Написати власний синтаксичний аналізатор для бінарного файлу з моделлю для її подальшого використання в ігровому рушії.

5. Реалізувати зчитування даних моделі та відбудова її на сцені рушія.

6. Реалізувати інтерфейс для взаємодії користувача із грою: довідка та основні настройки діб, погоди, світла, горизонту.

7. Запакування додатку для подальшого його використання без наявності ігрового рушія.

8. Створення інсталятора плагіна.

2. ВИБІР МЕТОДУ РІШЕННЯ

2.1 Аналіз Revit Platform API

Revit – це програмний комлекс для автоматизованого проектування, який реалізує принцип інформаційного моделювання будівель (BIM). Тобто, підхід до зведення, оснащення, забезпечення експлуатації та ремонту будівлі (до управління життєвим циклом об'єкта), який передбачає збір і комплексну обробку в процесі проектування всієї архітектурно-конструкторської, технологічної, економічної та іншої інформації про будівлю [5].

Призначений для архітекторів, проектувальників несучих конструкцій та інженерних систем. Надає можливості тривимірного моделювання елементів будівлі і плоского креслення елементів оформлення, створення призначених для користувача об'єктів, організації спільної роботи над проектом, починаючи від концепції і закінчуючи випуском робочих креслень і специфікацій [6].

Основні можливості API від Revit:

• Створення надбудов (add-ins) і макросів для автоматизації повторюваних завдань в інтерфейсі.

• Впровадження стандартів проектування проектів шляхом автоматичної перевірки помилок.

• Витяг даних проекту для аналізу і створення звітів.

• Імпорт зовнішніх даних для створення нових елементів або значень параметрів.

• Інтеграція інших додатків, включаючи аналітичні програми.

• Автоматичне створення проектної документації.

2.1.1. Простори імен Revit API

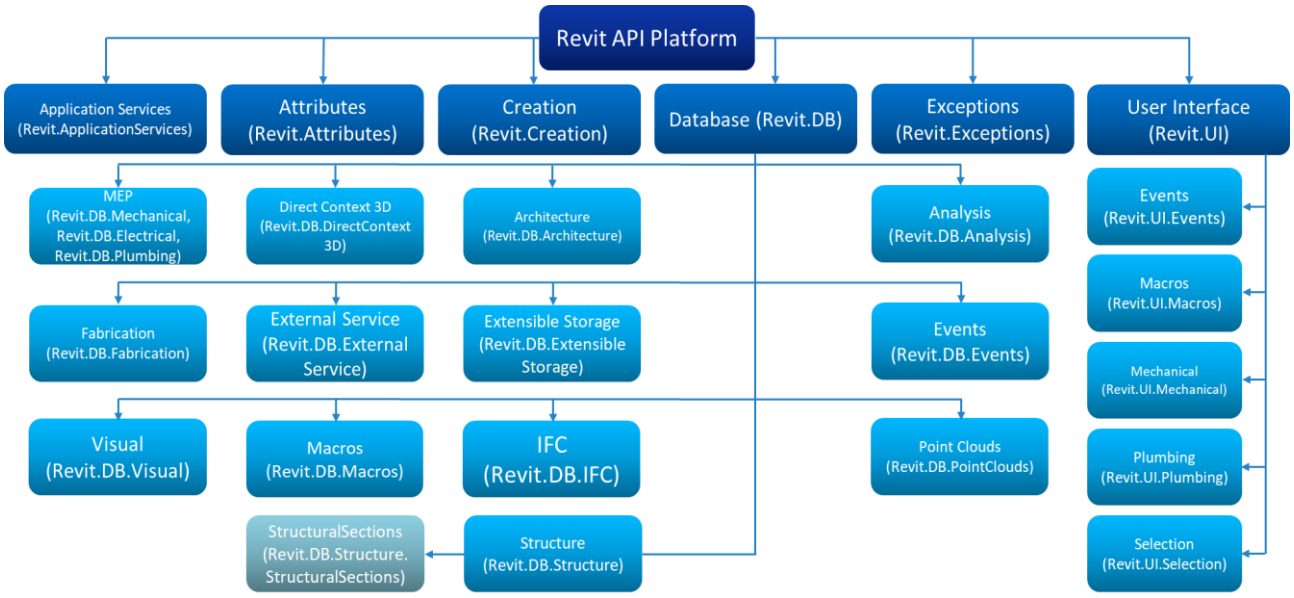

Рисунок.2.1. Структурна схема просторів імен Revit API

1. Application Services – простір імен для роботи з додатком Autodesk Revit, який забезпечує доступ до документів, параметрів і інших даних та налаштування для програми.

2. Attributes – простір імен, який надає доступ до атрибутів для управління зовнішніми командами і налаштуванням різних додатків.

3. Creation – простір імен, який призначений, в основному, для створення нових об'єктів, наприклад, Document або Application і дає гарантію на те, що створені елементи будуть правильно додані в ієрархію.

4. Database – простір імен, який призначений для роботи з різними елементами Revit, наприклад з підлогою, стінами, вікнами, лампочками з електрикою і т.д. І містить в собі безліч інших просторів імен, кілька з яких будуть розглянуті нижче.

5. Exceptions – простір імен, призначений для обробки різного роду винятків за типом проблем з доступом до файлів або, коли проблема в коді може привести до непередбаченої помилки.

6. User Interface – простір імен, який надає доступ до елементів призначених для користувача інтерфейсу і їх обробці.

2.1.2. Простір імен Autodesk.Revit.Attributes

API Revit надає кілька атрибутів для налаштування поведінки ExternalCommand (команди, які мають доступ до бази даних Revit, а також до обраних в даний момент елементів) і ExternalApplication (викликаються під час запуску і завершення Autodesk Revit. Вони можуть створювати нові панелі на вкладці надбудови і також можуть реєструвати обробники, які можуть реагувати на події, що відбуваються в інтерфейсі). Даний простір імен містить два основні класи: регенерації і транзакції аттрибутів.

Клас регенерації аттрибута використовують для управління регенераційні поведінкою зовнішньої команди або зовнішнього застосування. Має тільки одну опцію, режим Automatic все ще в розробці. У першому випадку, структура API не відновлюється після кожної зміни рівня моделі, тому в ручну примусово оновлюється документ після групи змін. У другому випадку, режим призупиняє всі оновлення для документа до тих пір, поки документ не буде відновлений, або він ризикує отримати доступ до застарілих даних.

Клас транзакції аттрибута повинен застосовуватися до класу реалізації інтерфейсу IExternalCommand для управління поведінкою транзакції для зовнішньої команди. Для цієї опції немає значення за замовчуванням. Цей режим управляє тим, як API-інфраструктура очікує, що транзакції будуть використовуватися при виклику команди.

Підтримувані значення:

• TransactionMode.Automatic – Revit створить транзакцію в активному документі до того, як буде виконана зовнішня команда, і транзакція буде виконана або відкат після завершення команди (на основі значення, що повертається на зворотний дзвінок ExternalCommand). Команда не може створювати і запускати свої власні транзакції, але вона може створювати SubTransactions. Команда повинна повідомляти про свій успіх або стані відмови з повертається значенням Result.

• TransactionMode.Manual – Revit не створюватиме транзакцію (але вона створить зовнішню групу транзакцій, щоб відкотити всі зміни, якщо зовнішня

команда повертає збій). Замість цього ви можете використовувати комбінації транзакцій, субтранзакцій і транзакційних груп, як вам завгодно. Revit перевірить, що всі транзакції (також групи і суб-транзакції) належним чином закриті після повернення з зовнішньої команди. Якщо немає, Revit скасує всі зміни, внесені в модель.

• TransactionMode.ReadOnly – транзакція (або група) не буде створена, і ніяка транзакція не може бути створена для часу життя команди. Зовнішня команда може використовувати тільки методи, які читаються з моделі. Винятки будуть викидатися, якщо команда або намагається запустити транзакцію (або групу), або намагається записати в модель [5].

2.1.3. Простір імен Autodesk.Revit.DB

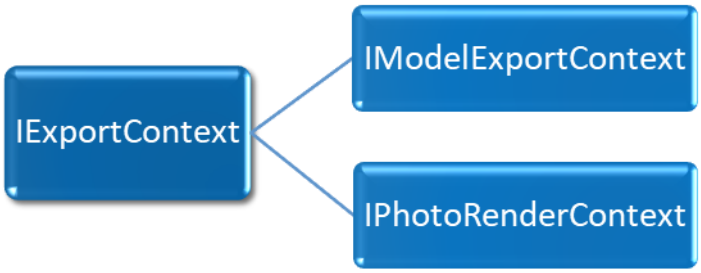

Рисунок 2.2 – Взаємозв’язок головних інтерфейсів простору імен Autodesk.Revit.DB

Докладний опис:

• CustomExporter – клас, який дозволяє експортувати 3D-види через контекст експорту. Метод експорту цього класу запускає стандартний процес рендеринга або експорту в Revit, але замість відображення результату на екрані або принтері висновок направляється через заданий для користувача контекст, який обробляє як геометричну, так і негеометричну інформацію.

• Document – являє собою проект Autodesk Revit і забезпечує доступ до всіх інших властивостей рівня документа. Revit може мати кілька проектів, відкритих і кілька видів для цих проектів. Активний або самий верхній вид буде активним проектом і, отже, активним документом, який доступний з об'єкта Application.

• Element – базовий клас для більшості постійних даних в документі Revit. Елемент зазвичай відповідає одному компоненту будівлі або малюнка, наприклад стіні, двері або розміру, але також може бути чимось більш абстрактним, як тип стіни або вид. Кожен елемент документа має унікальний ідентифікатор, представлений класом ElementId.

• Transform – клас, за допомогою якого можна перетворювати афінний простір. Афінний простір можна сприймати як векторний простір, в якому втратили початкову точку.

• View – базовий клас для всіх видів переглядів в Autodesk Revit. В уявленні можна побачити зображення, створене за допомогою моделі Revit. Уявлення можуть бути графічними (наприклад, плани, висоти або 3D-види) або текстові (наприклад, графіки). Уявлення відстежують елементи, які можна в них побачити.

• IExportContext – інтерфейс, який використовується в призначеному для користувача експорті для обробки моделі Revit. Це базовий клас для двох інших інтерфейсів, отриманих з нього: IPhotoRenderContext і IModelExportContext. Цей базовий клас містить методи, загальні для обох інтерфейсів листа.

• IPhotoRenderContext – інтерфейс, який використовується в призначеному для користувача експорті для рендеринга 3D-уявлень моделі Revit. З цим типом експортного контексту, використовуваним для виконання призначеного для користувача експорту, Revit буде перетинати модель і виводити геометрію моделі, як якщо б вона обробляла команду Render, викликану через призначений для користувача інтерфейс. Це означає, що будуть оброблятися і виводитися тільки такі елементи, які мають фактичну геометрію і придатні для відображення в візуалізованому поданні.

• IModelExportContext – інтерфейс, який використовується в призначеному для користувача експорті для експорту тривимірних уявлень моделі Revit. З цим типом експортного контексту, використовуваним для виконання призначеного для користувача експорту, Revit буде перетинати модель і виводити геометрію моделі, як якщо б вона перебувала в процесі регулярного відображення або

експорту 3D-виду. Це означає, що будь-яка геометрія, яку видно у відкритому вигляді (з урахуванням будь-якого поточного параметра видимості, що застосовується до уявлення), буде оброблятися і виводитися

Застосовуються діаграми діяльності, щоб проілюструвати потік управління в системі та спиратися на кроки, з'єднані з виконанням варіантів використання. Змодельовано почергові й паралельні дії, використовуючи діаграми діяльності. Діаграма діяльності концентрується на стані потоку і послідовності, в якій це відбувається [5].

2.1.4. Простір імен Autodesk.Revit.DB.IFC

Revit підтримує експорт і імпорт такого файлового розширення як IFC. IFC є глобальним стандартом обміну даними в будівельних галузях і загальної моделлю даних з відкритим файловим форматом.

Система IFC (Industry Foundation Classes) являє собою стандарт представлення даних і формат файлу, який використовується для визначення архітектурних і пов'язаних з конструкцією графічних даних САПР (система автоматизованого проектування – автоматизована система, що реалізує інформаційну технологію виконання функцій проектування, являє собою організаційно-технічну систему, призначену для автоматизації процесу проектування, що складається з персоналу і комплексу технічних, програмних та інших засобів автоматизації його діяльності) як тривимірних об’єктів реального віту.

Інтерфейс	**Класи**	
IExporterIFC	ExporterIFC	ImporterIFC
Інтерфейс, який використовується для реалізації власного експортера IFC.	Головний клас, наданий Revit для реалізації експорту IFC.	Головний клас, наданий Revit для реалізації імпорту IFC.

Таблиця 2.1 – Найбільш базовий інтерфейс і класи простору імен Autodesk.Revit.DB.IFC

Детальний опис:

• ExporterIFC – екземпляр цього класу наданий клієнту, щоб забезпечити реалізацію для експорта IFC. Він містить інформацію про параметри, вибрані користувачем для операцій експорта, а також членів, що використовуються для отримання доступу до даних, необхідних для правильної реалізації експорту.

• ІмпортерIFC – екземпляр цього класу наданий клієнту, щоб забезпечити реалізацію імпорту IFC. Він містить інформацію про параметри, вибрані користувачем для операцій імпорту, а також членів, що використовують для отримання доступу до даних, необхідних для правильного імпорту [5].

2.1.5. Простір імен Autodesk.Revit.DB.Visual

У даному просторі імен містять всі важливі класи для роботи з матеріалами. Матеріал може бути як текстурою (картинкою простіше кажучи), однотонним кольором або взагалі не бути. Властивості, які визначають матеріал, об'єднуються в, так звані, активи («Ассет»). Ассет - це групи властивостей, які контролюють певні характеристики або поведінку об'єкта.

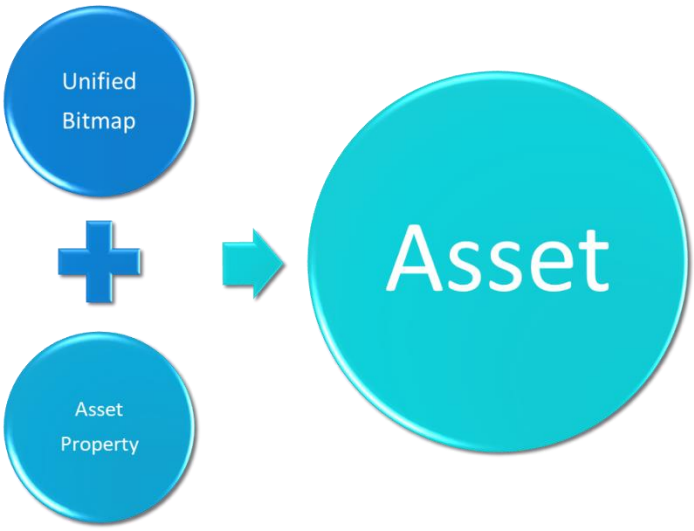

Рисунок. 2.3 – Схематичне зображення складової матеріалу

Виходячи з даних класів:

• Unified Bitmap – представляє зображення і його параметри і значення

відображення, тому він потрібен для того, щоб відшукати текстуру яка нанесена на елемент.

• AssetProperty – пропонує методи для надання можливості змінювати, додавати або видаляти Ассет, який приєднаний до властивості, тобто допомагає визначити потрібний ассет і, наприклад, забрати назви текстури або ж шлях до неї.

Як підсумок, ассети потрібні для того, щоб можна було зрозуміти який матеріал завдано на елемент і можливість з ним взаємодіяти: міняти колір, текстуру, прозорість і т.д. [5].

2.1.6. Простір імен Autodesk.Revit.Exception

Даний простір імен надає можливість обробки стандартних помилок. Найбільш використовувані класи Autodesk.Revit.Exception:

• Application Exception – цей клас є базовим класом всіх виключених Revit винятків, які виникає при виникненні помилки, не пов'язаної з фатальним результатом.

• Argument Exception – це базовий клас для винятків, які видаються при перевірці аргументів функції, якщо один з аргументів, наданих методу, недійсний.

• CentralModelException – базовий клас для винятків, які є спільними як для заснованих на файлах, так і для серверних центральних моделей або специфічні тільки для стандартних моделей на основі файлів.

• InvalidOperationException – виняток, яке викликається, коли виклик методу недійсний для поточного стану об'єкта.

• IOException – виняток, що виникає при виникненні помилки введення-виведення. [5]

2.1.7. Простір імен Autodesk.Revit.UI

Найбільш широко вікористовується простір імен для роботи з графічним інтерфейсом. Почнемо з того, що об'єктами верхнього рівня в API-інтерфейсі Revit є додаток і документ. Вони представлені класами Application,

UIApplication, Document і UIDocument.

• Об'єкт додатка відноситься до окремої сесії Revit, забезпечуючи доступ до документів, параметрам і іншими даними і налаштуванням всієї прикладної програми:

o Autodesk.Revit.UI.UIApplication – забезпечує доступ до інтерфейсів інтерфейсу UI для додатка, включаючи можливість додавання RibbonPanels в призначений для користувача інтерфейс і можливість отримання активного документа в інтерфейсі.

o Autodesk.Revit.ApplicationServices.Application – забезпечує доступ до всіх інших властивостей рівня додатки. Якщо є екземпляр об'єкта застосування рівня бази даних, то можна створити з нього UIApplication і отримати доступ до документів з рівня додатки

• Об'єкт документа являє собою один проект проекту Revit, що представляє собою модель будівлі. Revit може мати кілька відкритих проектів і кілька видів для одного проекту:

o Autodesk.Revit.UI.UIDocument – забезпечує доступ до інтерфейсів інтерфейсу UI для документа, таким як вміст вибору і можливість спонукати користувача робити вибір і вибирати точки.

Можна отримати документ рівня бази даних (який містить інтерфейси, не пов'язані з призначеним для користувача інтерфейсом) за допомогою властивості Document.

o Autodesk.Revit.DB.Document – забезпечує доступ до всіх інших властивостей рівня документа.

Якщо є документ рівня бази даних і йому потрібно отримати доступ до нього з призначеного для користувача інтерфейсу, можна створити новий UIDocument з цього об'єкта (документ повинен бути відкритим і видимим в інтерфейсі, щоб дозволити методам працювати успішно).

Також, якщо кілька документів відкриті, активним документом є той, чий вигляд активний сесії Revit. Чимало важливі інтерфейси:

• IExternalApplication – інтерфейс, який підтримує додавання зовнішніх

додатків в Revit. Зовнішнім програми мають дозвіл на налаштовувати користувальницький інтерфейс Revit і додавати події та оновлення до сесії.

Містить два методи: OnShutdown і OnStartup. Реалізують виконання деяких завдань при відключенні і запуску Revit відповідно.

• IExternalCommand – інтерфейс, який повинен бути реалізований для забезпечення реалізації зовнішньої команди надбудови Revit.

Містить один метод Execute, який перевантажують для реалізації і зовнішньої команди, а результат вказує, чи виконується виконання, чи завершується виконання або скасовується користувачем. Якщо це не вдасться, Revit скасує будь-які зміни, зроблені зовнішньою командою [5].

2.2. Ігровий рушій Unreal Engine

Unreal Engine є ігровим рушієм, який написаний на мові C ++, дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube і ін., а також на різних портативних пристроях, наприклад, пристроях Apple (iPad, iPhone), керованих системою iOS і інших.

Для гри по мережі підтримуються технології Windows Live, Xbox Live, GameSpy і інші, включаючи до 64 гравців (клієнтів) одночасно. Таким чином, рушій адаптували і для застосування в іграх жанру MMORPG (Масова багатокористувацька онлайнова рольова гра – жанр онлайнових рольових відеоігор, у якій велика кількість гравців взаємодіє один з одним у віртуальному світі (головним чином у жанрі фентезі). Як і в більшості RPG, гравцеві пропонується роль вигаданого героя і можливість управляти його діями. MMORPG відрізняються від однокористувацьких і невеликих мережевих рольових ігор безліччю гравців, а також віртуальним світом, який продовжує існувати за відсутності гравця. Віртуальний світ підтримується видавцем гри.) [7].

2.2.1. Unreal Editor

Unreal Editor – це редактор рівнів і інших ресурсів для ігор на ігровому движку Unreal Engine, який практично завжди йде в поставці з самою грою. Являє собою єдиний додаток для редагування рівнів гри і всього з ними зв'язаного, наприклад, створення скриптових сцен, імпорт ресурсів з сторонніх додатків і так далі. Все, що потрібно для створення повноцінного рівня, є в редакторі, ніякі додаткові утиліти не потрібні. Протягом часу розробки рушія функціональність редактора допрацьовувалася, але кардинальних змін не відбувалося. Основні можливості:

• Створення ігрових рівнів в WYSIWYG-режимі (Візуальний редактор, також вживається назва WYSIWYG («що бачиш, то і отримаєш») – властивість прикладних програм або веб-інтерфейсів, в яких вміст відображається в процесі редагування і виглядає максимально близько схожим на кінцеву продукцію, яка може бути друкованим документом, веб-сторінкою або презентацією. В даний час для подібних програм також широко використовується поняття «візуальний редактор».) [8], є також створення ландшафту. Ігрові об'єкти також додаються в WYSIWYG-режимі.

• Огляд об'єктів (класів, текстур, звуків, анімацій і т. д.) І вбудовані функції для їх базового редагування. Наприклад, до текстур можна застосовувати шейдери, звуки можна змішувати, прискорювати, змінювати тембр і так далі. Такі функції стали особливо сильні і різноманітні в Unreal Engine 3.

• Можливість запуску гри з поточним рівнем прямо з редактора. Починаючи з UE3, це більше не вимагає запуску нового процесу: гра запускається прямо в редакторі, що, очевидно, сильно економить ОЗУ і час.

2.2.2. Blueprint Visual Scripting/C++

Unreal Engine надає два інструментарії для програмістів, які також можуть використовуватися в тандемі, щоб прискорити технологічні процеси розробки.

Система візуальних сценаріїв Blueprints – це надійний інструмент, який дозволяє створювати класи в редакторі за допомогою об'єднання разом

функціональних блоків і посилань на властивості. Ця система надзвичайно гнучка і потужна, оскільки вона дозволяє дизайнерам використовувати практично весь спектр концепцій і інструментів, які зазвичай доступні тільки програмістам. [9]

Крім того, специфічна для Blueprint розмітка, доступна в реалізації Unreal Engine C ++, дозволяє програмістам створювати базові системи, і таким чином програмісти можуть створювати основні класи геймплея, які потім підкласифіціруються і повторюються розробниками рівнів.

2.2.3. Об'єкти: актори та компоненти

Всі елементи ігрового рушія представлені у вигляді об'єктів, що мають набір характеристик, і класу, який визначає доступні характеристики. У свою чергу, будь-який клас є «дочірнім» класом object.

Актори – базовий клас всіх об'єктів ігрового процесу, які можуть бути поміщені в світ. Отже, насправді всі екземпляри Unreal Engine є об'єктами; проте термін «Актори» зазвичай використовується для позначення примірників класів, які походять від AActor в їх ієрархії, а термін «Об'єкти» використовується для позначення примірників класів, які не успадковуються від класу AActor. Більшість класів, які ви створюєте, успадковують від AActor в якийсь момент своєї ієрархії.

Загалом, актори можна розглядати як цілі предмети або сутність, а об'єкти – більш спеціалізовані частини. Актори часто використовують Компоненти, які є спеціалізованими об'єктами, щоб визначити певні аспекти їх функціональності або зберегти значення для колекції властивостей. Візьміть автомобіль в якості прикладу. Автомобіль в цілому є актором, тоді як частини автомобіля, такі як колеса і двері, будуть компонентами цього актора.

2.2.4. Класи: пішаки та персонажі

Клас визначає поведінку і властивості конкретного Актора або Об'єкту, що використовуються при створенні гри Unreal Engine. Класи ієрархічні, тобто клас

успадковує інформацію від своїх батьківських класів (класи, які він був отриманий або «подкласифіцірован»), і передає цю інформацію своїм дочірнім елементам. Класи можуть бути створені в коді C ++ або в «Blueprint».

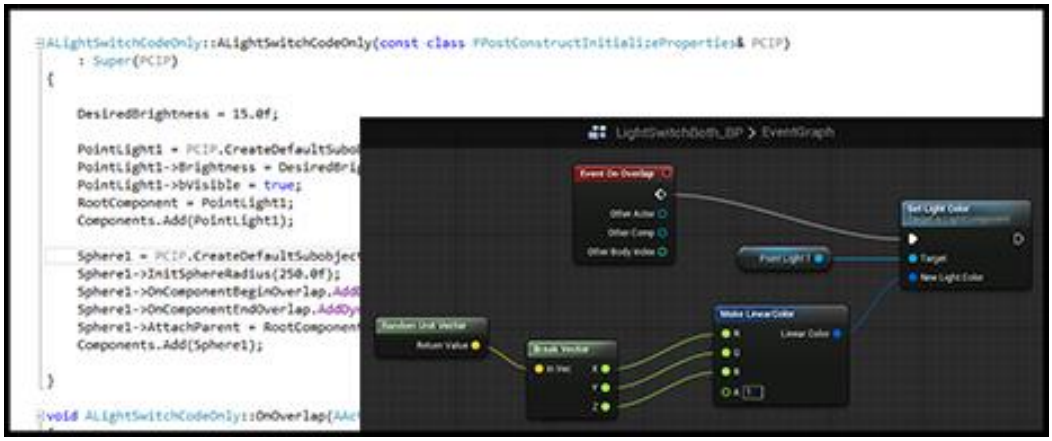

Рисунок.2.4 – Приклад простих класів C++ та «Blueprint»

Клас Pawn є базовим класом всіх Акторів, які можуть контролюватися гравцями або штучним інтелектом. Пішак - це фізичне уявлення гравця або об'єкта ШІ в світі. Це не тільки означає, що Пішак визначає, як виглядає гравець або об'єкт ШІ візуально, але також і те, як він взаємодіє зі світом з точки зору колізій та інших фізичних взаємодій. Це може збивати з пантелику в певних обставинах, так як деякі типи ігор можуть не мати видимої сітки гравця або аватара в грі. Незважаючи на це, Pawn як і раніше являє фізичне місце розташування, поворот і т. д. гравця або об'єкта в грі.

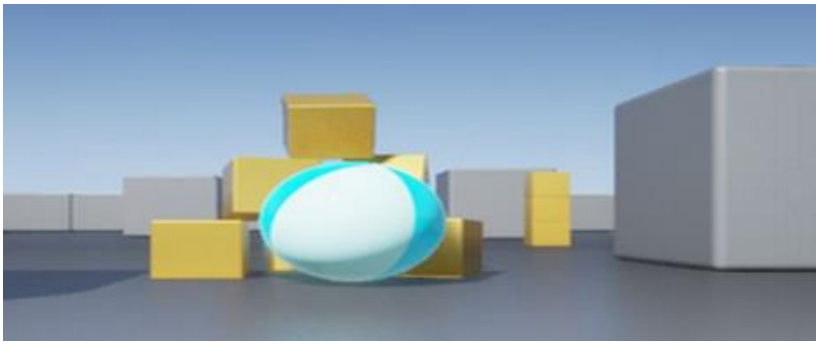

Рисунок 2.5 – Зображення пішака

Персонаж – це особливий тип пішаки, який може ходити і має людську подобу. Він може виконувати базове людиноподібна рух, плавно переміщатися і має деякі функції, пов'язані з анімацією.

Рисунок 2.6 – Зображення персонажу

2.2.5. Контролер: PlayerController і AIController

Контролер – це Актор, який відповідає за керівництво Пішаком. Зазвичай вони випускаються в двох варіантах: AIController і PlayerController. Контролер може «мати» пішаком, щоб взяти її під контроль.

PlayerController – це інтерфейс між Pawn і гравцем, який контролює його. PlayerController по суті являє бажання гравця.

PlayerController також є основною точкою мережевої взаємодії для багатокористувацьких ігор. Під час багатокористувацької гри сервер має один екземпляр PlayerController для кожного гравця в грі, так як він повинен мати можливість робити виклики мережевих функцій кожному гравцеві. У кожного клієнта тільки PlayerController, який відповідає їх гравцеві, і може використовувати свій PlayerController тільки для зв'язку з сервером.

У той час як PlayerController покладається на людського гравця, щоб приймати рішення про те, що робити, AIController більше сконцентрований на відповіді на введення з середовища і ігрового світу. Робота AIController полягає в тому, щоб спостерігати за навколишнім світом і приймати рішення реагуючи, відповідно, без явного введення від людини.

2.2.6. Відносини класу фреймворку

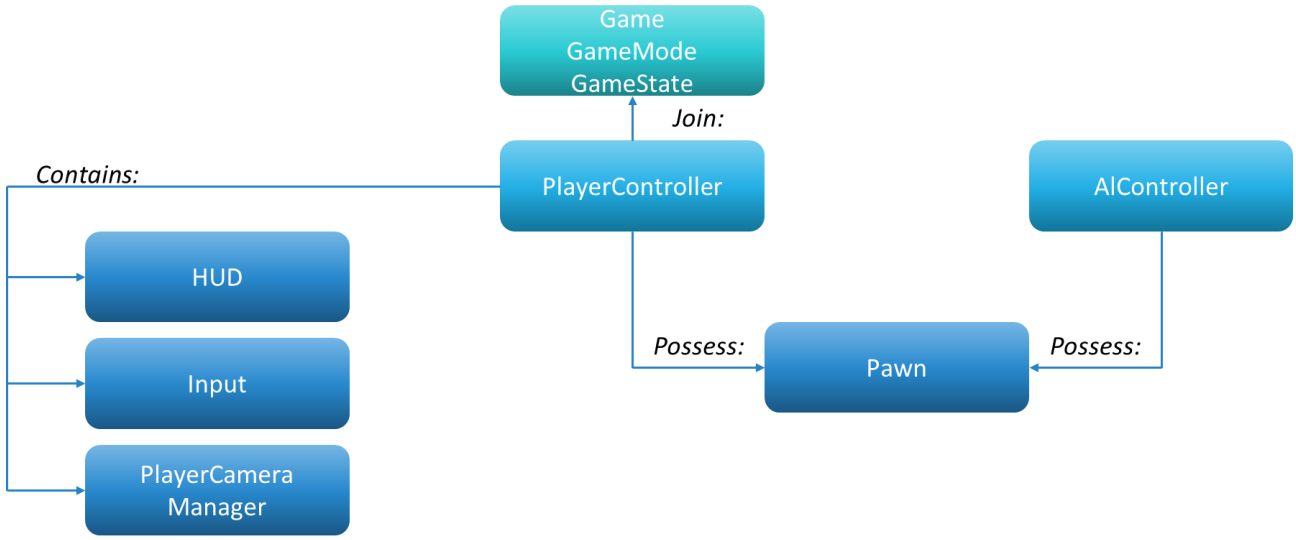

Рисунок 2.7 – Блок-схема відносин класів

Ця блок-схема ілюструє, як ці основні класи геймплея ставляться один до одного. Гра складається з GameMode і GameState. Людські гравці, що приєднуються до гри, пов'язані з PlayerControllers. Ці PlayerControllers дозволяють гравцям володіти пішаками в грі, щоб вони могли мати фізичні уявлення на рівні. PlayerControllers також надають гравцям елементи управління введенням, HUD (HUD - це «хедз-ап дисплей» або 2D-екран, який часто зустрічається в багатьох іграх. За типом здоров'я гравця, боєприпаси, приціли з пістолетами і т. д. Кожен PlayerController зазвичай має один з них), а PlayerCameraManager (PlayerCameraManager є «очним яблуком» для гравця і керує тим, як він себе веде. У кожного PlayerController зазвичай є і один з них) - для перегляду камер.

3. ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Організація рішення експорту

3.1.1. Створення plug-in додатку

Для того, щоб створити plug-in для Revit потрібно виконати такі дії:

1. Запуск середовища розробки Visual Studio. Для API Revit 2018/2019/2020 необхідно використовувати версії Visual Studio, яка підтримує .NET Framework 4.6. Revit 2019 буде .NET 4.7.

2. Створити проект бібліотеки класів.

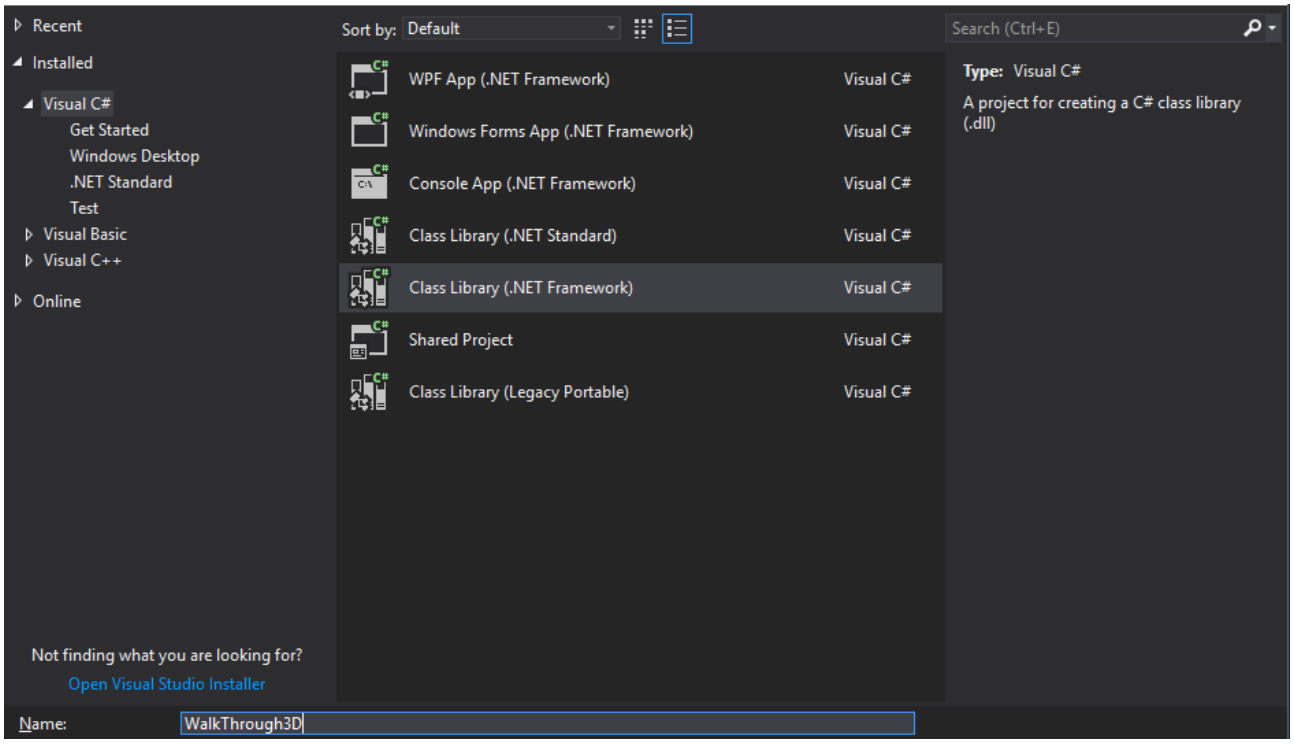

Рисунок 3.1 Створення DLL-бібліотеки.

3. Додати посилання на бібліотеки Revit (RevitAPI.dll, RevitAPIUI.dll), які знаходяться за шляхом C:\Program Files\Autodesk\Revit 20(1\2)x*. Потрібно додати два інтерфейсні DLL-файли: RevitAPI.dll, RevtAPIUI.dll.

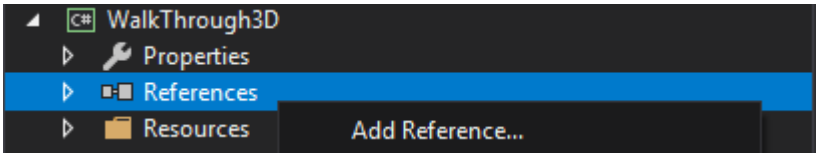

Рисунок 3.2 – Додавання посилань

Тепер у проекті є посилання на два інтерфейсні DLL-файли. Ці інтерфейсні файли відображаються у всіх API-інтерфейсах Revit, і проект може використовувати всі доступні API

4. Встановити файли посилання "Копіювати значення властивості" до значення False.

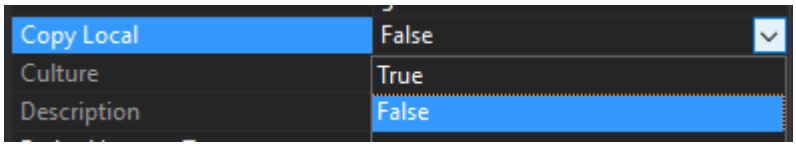

Рисунок 3.3 – Встановлення флагу

Весь плагін можна розбити на 3 важливі частини: класи, які реалізують IExternalCommand, IExternalApplication та сам маніфест.

Клас, що реалізує IExternalCommand має назву WalkThrough3DCommands. Основна його задача в тому, щоб передати для подальшої обробки UIApplication та Document поточного проекту. Клас, що реалізує IExternalApplication має назву WalkThroughAddin. Головна мета цього класу, обробити події завантаження та завершення роботи додатку. При завантаженні потрібно обов’язково ініціалізувати логування застосунку. Маніфест – це файл, розташований у певному місці, перевіреному Revit під час запуску програми. Маніфест містить інформацію, яку використовує Revit для завантаження та запуску плагіна. Лістинги класів WalkThrough3DCommands, WalkThroughAddin та маніфест WalkThrough3D.addin наведені в Додатку А.

Код з бібліотеки DLL компілюється до коду Common Intermediate Language (CIL – також відомий як MSIL), використовуючи спеціальний компілятор. CIL є незалежним від CPU набором інструкцій. Код CIL, згенерований з вихідного коду C# упаковується в збірку .NET. Така збірка являє собою бібліотеку CIL-коду, що зберігається у форматі Portable Executable (PE) (який містить як CIL, так і пов'язані з ним метадані). Збірки можуть бути як збірки процесу (EXE), так і бібліотечні збірки (DLL) [11].

Плагіни Revit компілюються в бібліотечні файли збірки (DLL), які потім завантажуються та виконуються з пам'яті Revit.

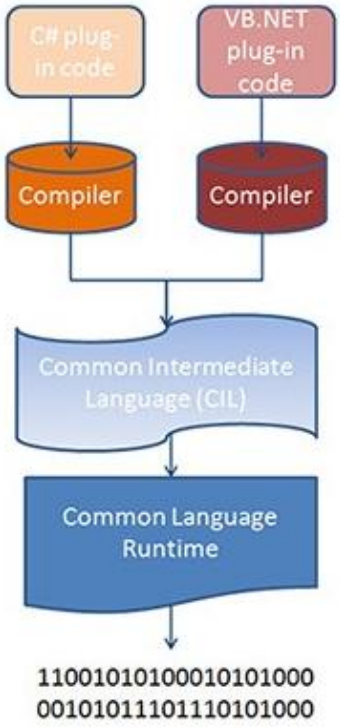

Рисунок 3.4 – Схема запуску виконуваних файлів

Під час виконання збірки .NET, CIL (що знаходиться в зборі) передається через компілятор JRT (CLR) для створення машинного коду. JIT-компіляція CIL до нативного коду відбувається при виконанні програми. Оскільки під час виконання не потрібний весь код, компілятор JIT перетворює CIL тільки тоді, коли це необхідно, таким чином, заощаджуючи час і пам'ять. Він також зберігає будь-який згенерований код в пам'яті, роблячи його доступним для подальшого використання без необхідності перекомпілювати. На останньому етапі цього процесу машинний код виконується процесором комп'ютера [12].

3.1.2. Створення структури даних

Для того, щоб створити структуру даних потрібно мати всі потрібні дані для побудови 3D-моделі та записати інформацію цієї моделі до файла. Для запису даних до файлу було створено інтерфейс ISerializable з методом:

```
void WriteData(System.IO.BinaryWriter file);
```

Для читання і запису двійкових значенні вбудованих в C# типів даних служать класи потоків BinaryReader і BinaryWriter. Використовуючи ці потоки, слід мати на увазі, що дані зчитуються і записуються у внутрішньому двоїчному форматі, а не у зрозумілій текстовій формі. Клас BinaryWriter служить оболонкою, у яку направляэться байтовий потік, який керує виводом двійкових даних. Таким чином, кожен клас, що наслідує цей інтерфейс, повинен реалізувати свій запис до файлу. Для чіткого розмежовування даних було створено такі класи:

- AFacet – клас, в якому зберігаються вершини трикутників полімеша.
- APoint – клас, в якому зберігаються точки вершин трикутників полімеша.
- AUV – клас, в якому зберігаються UV-координати (UV mapping (UV-розгортка) – відповідність між координатами на поверхні тривимірного об'єкту (X, Y, Z) і координатами на текстурі (U, V). Значення U і V зазвичай змінюються від 0 до 1). [10]
- AMaterial – клас, в якому зберігається колір або шлях до текстур полімеша.
- APolymesh – клас, який зберігає в собі всі вище вказані класи, а також ім’я мешу, матеріалу та значення колізій.

Лістинги всіх вище вказаних класів та інтерфейсу наведені в Додатку Б.

3.1.3. Реалізація експортера

Для реалізації кастомного експортеру було створено новий проект з назвою Exporter. Щоб мати змогу експортувати важливу інформацію з Revit, потрібно реалізувати IPhotoRenderContext інтерфейс.

У першу чергу, у експортері класа потрібно отримати поточний документ, оскільки подальша робота буде залежати від нього. При наслідуванні інтерфейсу потрібно реалізувати такі методи:

- Start() – у цьому методі потрібно указати в поверненні значення true;
- IsCanceled() – цей метод повинен повертати false;

- OnElementBegin() – в цьому методі потрібно дізнатись категорію елементу та призначити значення булевої змінної, яка відповідає за колізію.
- OnFaceBegin() – в цьому методі достатньо повернути RenderNodeAction.Proceed.
- OnInstanceBegin() – в цьому методі потрібно отримати трансформацію координат та занести їх до стеку.
- OnInstanceEnd() – в цьому методі достатньо видалити останній елемент стеку.
- OnLinkBegin() – цей метод не обов'язково віддавати на обробку (RenderNodeAction.Skip).
- OnLight() – в цьому методі потрібно отримати координати розташування світла.
- OnRPC() – в цьому методі також отримуємо координати розташування RPC – моделей (Rich Photo-realistic Content - плагін від Archvision).
- OnViewBegin() – в цьому методі просто проскаємо експорт.
- OnPolymesh() – головний метод, в якому отримуємо такі дані: вершини трикутників, координати трикутників, самі трикутники, UV-координати та нормалі.
- OnMaterial() – метод, який надає змогу отримати матеріал. Для нього було написано ще декалька методів, які будуть описані далі.
- OnElementEnd() – в цьому методі можна визивати метод з'єднання мешів.
- Finish() – це завершальний метод, в якому ведеться рахунок експортованих мешів.

При експорті полімешу постало таке питання правильного розподілу на меші та матеріал. Тому було реалізовано метод MergeMesh():

```
public List<APolymesh> MergeMesh(List<APolymesh> mergMeshes)
        {
            int first = 0;
            while (first < mergMeshes.Count)
            {
                int second = first + 1;
                while (second < mergMeshes.Count)
```

```
                    {
                        if (mergMeshes[first].NameOfMaterial ==
mergMeshes[second].NameOfMaterial && mergMeshes[first].GetCollision ==
mergMeshes[second].GetCollision)
                        {
                            Count = mergMeshes[first].PolymeshPoints.Count;

mergMeshes[first].PolymeshPoints.AddRange(mergMeshes[second].PolymeshPoints);

mergMeshes[first].PolymeshUvModels.AddRange(mergMeshes[second].PolymeshUvModels);

mergMeshes[first].PolymeshNormals.AddRange(mergMeshes[second].PolymeshNormals);
                            foreach (var fac in mergMeshes[second].PolymeshVertices)
                            {
                                mergMeshes[first].PolymeshVertices.Add(new AFacet(fac.V1
+ Count, fac.V2 + Count, fac.V3 + Count));
                            }
                            mergMeshes.RemoveAt(second);
                        }
                        else
                        {
                            second++;
                        }
                    }
                    first++;
                    Count = 0;
                }
                return mergMeshes;
            }
```

Суть цього методу полягає в тому, щоб об’єднати меші за спільними ознаками в одну секцію по-елементно. Це знадобиться для подальшого використання мешів при відбудові. При отриманні матеріалів виник ряд проблем, які були вирішенні за допомогою декількох методів. Для початку треба було отримати схеми для кожного елемента, для цього було розроблено метод CheckColor.

За допомогою цього методу визначається тип схеми у поточному ассеті та знаходяться можливі текстури та колір.

Матеріл має стандартні схеми:

- Ceramic
- Concrete
- Generic
- Glazing
- Hardwood
- MasonryCMU
- Metal

- MetallicPaint
- Mirror
- PlasticVinyl
- SolidGlass
- Stone
- WallPaint
- Water

Тому з кожної схемі потрібно отримати шлях до текстури, колір чи їхню відсутність. Для цього було розроблено два способи. Першим способом реалізовано доступ до текстур через індивідуальну схему, другим спосіб – доступ через основну схему UnifiedBitmap. Revit API надає методи пошуку за ім'ям схем та параметрів через властивості ассетів. Всі допоміжні методи (CheckColor(), CheckMaps(), CheckTint(), GetFull(), GetTextures(), TypesOfSchemas()) можна знайти в лістингах наведених в Додатку Б [13].

3.1.4. Впровадження WixInstaller з реалізацією CustomActions

Щоб зробити інсталлятор потрібно виконати такі пункти:

1. Завантажити Wix (http://wixtoolset.org/releases/) та встановити.

2. Для керування файлами програми було використано Visual Studio Extension, WaX. Управління файлами, як правило, найскладніша частина WiX Toolset, але WaX спрощує роботу [16].

3. Створення Wix setup проекту.

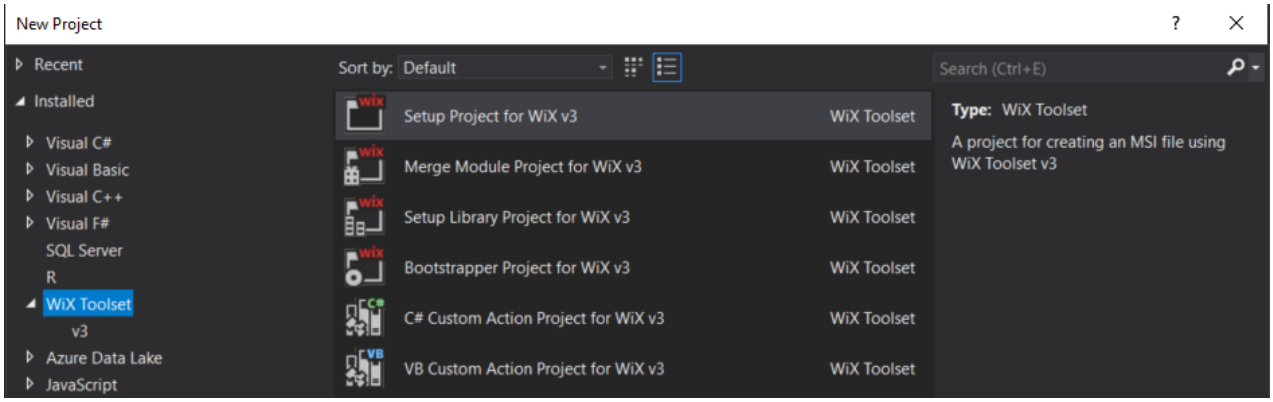

Рисунок 3.5 – Створення SetupPlugin

4. Додати необхідні файли до проекту налаштування. З урахуванням того, що в інсталляторі окрім файлів плагіну повинні бути файли гри, то решта буде дописана в кодовому файлі проекту.

5. Треба знайти атрибут Виробник у файлі Product.wxs і заповнити його.

Після виконання вище вказаних пунктів, проект можна зібрати. Залишилось дописати деталі по типу іконка інсталлятора, назву, додаткові файли і т.д. та роботу з CustomActions до файлу Product.wxs.

Щоб створити CustomAction достатньо додати новий C# Custom Action project до рішення. Цей проект створить спеціальний проект дій, який містить файл CustomAction.config і CustomAction.cs. Основна мета цього проекту – це перевірка наявності продукту Revit та отримання його версії. Після того, як отримано потрібні дані, реалізовано два маніфести з адресами: 1) розміщення файлів плагіну до потрібної папки; 2) видалення всіх файлів плагіну при деінсталяції. [17]

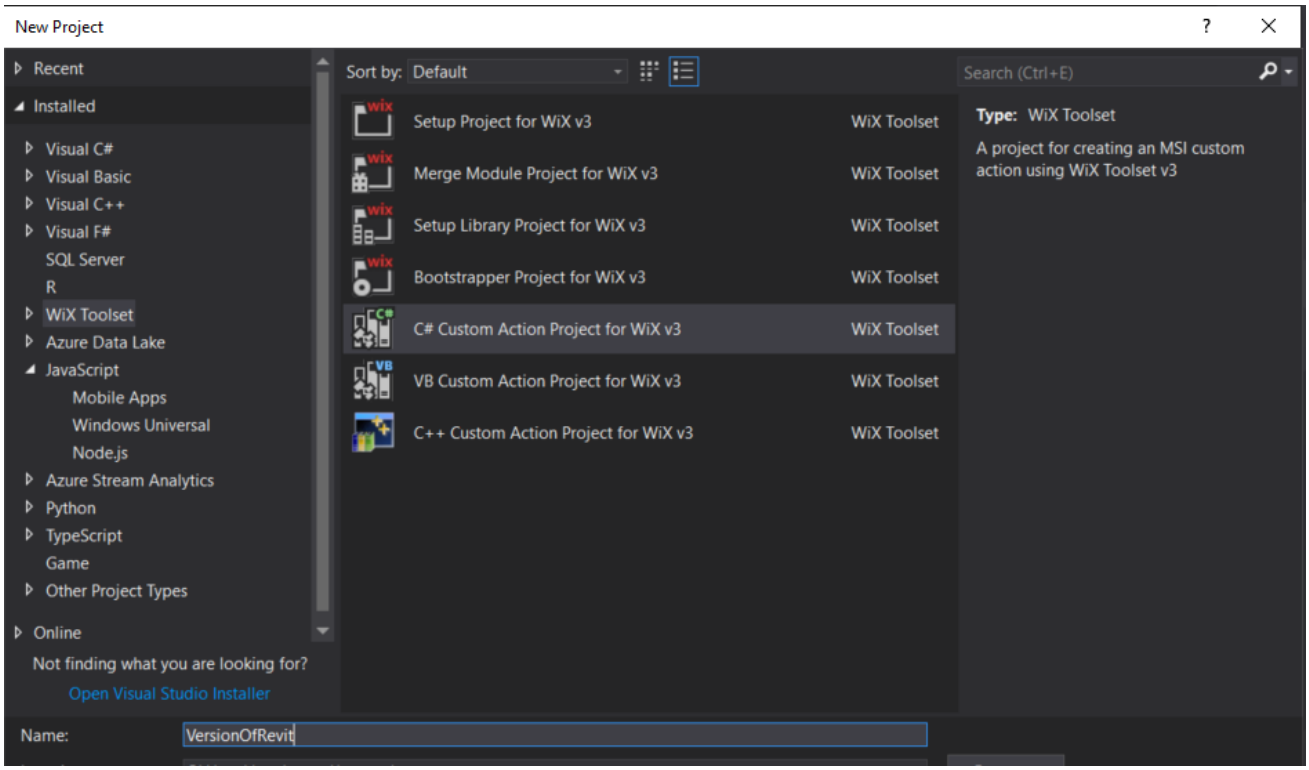

Рисунок3.6 – Створення VersionOfRevit

Лістинги до файлу Product.wxs та CustomActions наведені в Додатку Б.

3.1.5. Створення графічного інтерфейсу користувача

Однією із важливих частин є оформлення графічного інтерфейсу користувача. Інтерфейс повинен бути в першу чергу зрозумілим, а також містити інформацію про продукт. Для створення інтерфейсу було використано Windows Forms. Створено дві форми: Settings та About.

Форма Settings містить:

- Рядок для введення розміщення експортованої папки.
- Кнопку для пошуку папки.
- Кнопки “About”, “Run”, “Cancel”, відповідно інформація про продукт, запуск експорту, відміна.
- Індикатор виконання.

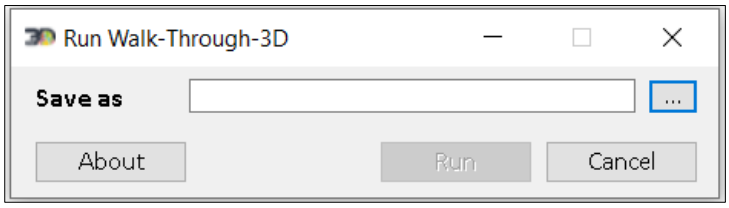

Рисунок 3.7 – Зовнішній вигляд форми Settings

При натисканні кнопки “…” відкривається Folder Browser, який був дещо відозмінений за допомогою WinAPI.

Спочатку було підключено .dll бібліотеку Windows з певною точкою входу, в залежності від призначення. Суть полягає в тому, щоб спочатку знайти потрібну форму та змінити її надпис, далі знайти кнопки на цій формі, надпис однієї з яких потрібно також змінити. За допогою кнопки “Run” викликається клас Exporter та починається експорт. За допомогою кнопки “About” викликається нова форма About, що містить в собі інформацію про версію продукту, веб-адресу компанії та електронну пошту. Веб-адреса та E-mail повинні бути діючими посиланнями. Лістинги класів Settings та About надані в додатку Д.

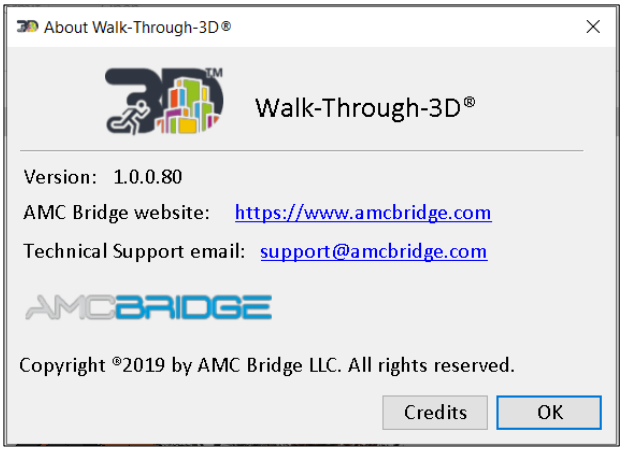

Рисунок.3.8 – Зображення форми About

3.2. Організація рішення парсеру

Після того як готовий плагін на виході дає бінарний файл із потрібною інформацією, її потрібно підготувати до імпорту. А саме: 1) отримати записані дані; 2) обробити вихідні дані.

У першу чергу, потрібно створити таку ж структуру даних, яка була при експорті полімешу. Оскільки движок підтримує лише мову програмування C++, то і структура даних повинна бути реалізована за допомогою C++ та для раціональності, проект зі структурою даних повинен збирати дві бібліотеки: .dll та .lib. По-друге, парсер повинен містити в собі методи, які нададуть доступ до проаналізованих даних [18].

3.2.1. Створення структури даних

Структура даних вже є розробленою, тому залишається тільки реалізувати певні класи:

- AFacet – клас, який призначений для зберігання вершин трикутнів (три змінних V1,V2, V3 типу float).
- APoint – клас, що створений для зберігання точок трикутника (три змінних X,Y,Z типу double).
- AUV – клас, призначений для UV-координат (дві змінні U, V типу double).

- AMaterial – клас, який містить в собі інформацію про матеріал (назва матеріалу, колір, прозорість, шляхи до текстур).
- APolymesh – клас, який уособлює всю 3D-модель (містить в собі всі вище вказані класи, нормалі та колізії).
- Parser – клас, за допомогою якого буде зчитуватись інформація з файлів та фасуватись.

Класи, що уособлють структуру даних мають назву з префіксом “A”, тому що таким чином виділяється їх призначення до класу акторів в рушії. Щоб всі написані класи входили до бібліотек було окремо визначено [19]:

```
#define DLL_EXPORT __declspec(dllexport),
```

та прописано біля кожного класу: `class DLL_EXPORT назва_класу {}`.

3.2.2. Створення аналізу даних

Головним завданням парсеру є правильне отримання інформації з файлу. Для зчитування даних використовувалась функція: reinterpret_cast < new_type > (expression) , sizeof(type), де new_type завжди char; expression містить зміну, в яку записує значення; type – тип даного, яке буде зчитуватись. Ця функція вікористовується для перетворення одного покажчика іншого покажчика будь-якого типу, незалежно від того, чи клас пов'язаний один з одним, чи ні. Вона не перевіряє, чи є тип вказівника і дані, вказані вказівником, однаковими чи ні.

На виході експортер дає папку “Meshes” з такими файлами:

- RPCs – зберігає координати, де повинна бути розміщена RPC-модель;
- Levels – зберігає назву та висоту рівня, потрібно для подальших налаштувань;
- Lights – зберігає координати, де повинна бути розміщена точка світла;
- Materials – зберігає назву матеріалів, колір, прозорість, текстури;
- Names – зберігає назву мешів;
- Файли з мешами, що мають назви згенеровані за допомогою GUID (GUID (англ. Globally Unique Identifier) – статистично унікальний 128-бітний ідентифікатор. Загальна кількість унікальних ключів настільки велика

(2^{128}=3,4028*10^{38}), що ймовірність того, що у світі будуть незалежно згенеровані два однакових ключі, вкрай мала) – зберігають вершини та точки трикутників, нормалі, UV-координати, колізії. Логіка парсера заключається в тому, що в кожному файлі є певна послідовність, яка не повинна змінюватись.

Таким чином, файл “Names” зберігає послідовно записані назви мешів типу string; файл “RPCs” та “Lights” зберігає записані точки типу double; файл “Levels” зберігає спочатку всі висоти типу double, а потім всі назви рівнів типу string; файл “Materials” зберігає спочатку назву матеріала типу string, потім два шляхи до різних текстур типу string, далі колір типу int та прозорість типу int.

Кожен файл, що зберігає інформацію про меш має таку послідовність:

1) Вершини;
2) Точки вершин;
3) UV-координати;
4) Нормалі;
5) Колізії;
6) Назву матеріала, який призначений для цього мешу.

Лістинги класу Parser.cpp наведені в додатку Е.

3.3. Організація рішення імпорту

Після побудови парсеру на виході є дві біблиотеки .dll та .lib. Ігровий рушій надає можливість підключати власні бібліотеки. Для цього потрібно створити папку “ThirdParty” та розмістити в ній бібліотеки з файлами типу header. Хедери являються точкою входу до бібліотек, які потрібно «підключити» в файлі *.Build.cs:

```
        PublicAdditionalLibraries.Add(Path.Combine(ThirdPartyPath,
"Parser/Win64/Parser.lib"));
                PublicIncludePaths.Add(Path.Combine(ThirdPartyPath, "Parser/includes"));
                PublicDelayLoadDLLs.Add("Parser.dll");
```

Після, потрібно обновити проект типу .uproject та обновити рішення Visual Studio. Далі достатньо підключити парсер через звичайний #include.

Головна мета полягає в побудові 3D-моделі з отриманих даних.

Найкращим рішення для відбудови мешу є використання UProceduralMeshComponent (компонент, який дозволяє вказати власну геометрію сітки трикутника [22].

Одним із важливих аспектів є імпорт матеріалів. Рушій створює свої файли типу uasset, тому потрібно розробити за допомогою Unreal Editor матеріал, а за допомогою коду отримати доступ до створеного матеріалу та змінити потрібні нюанси: колір, текстуру і т.д. Лістинги актора імпорту наведені в додатку Б.

3.4. Реалізація налаштувань для гри

Дана програма розрахована на широку аудиторію, тому окрім звичайного експорту повинна мати невеликі налаштування. Одним із головних вікон є вікно з довідкою

Рисунок 3.10 – Вікно довідки

Довідка містить в собі таку інформацію:

- Рухатись можна за допомогою клавіш WASD, стрибати – SpaceBar, прискорюватись – Shift.
- Ліва кнопка миші змінює режим руху персонажу: режим польоту чи режим ходьби.
- Кнопка “F” змінює уявлення: від першої чи третьої особи.
- Кнопка “O” відкриває налаштування гри.
- Кнопка “Escape” відповідає за відображення довідки.
- Кнопка “M” відповідає за відображення мапи
- Кнопка “Exit” закриває додаток.

Оскільки налаштування є тривіальним, то його можно реалізувати за допомогою Blueprints.

1. Перша вкладка має нзву “General” і містить в собі налаштування дня/ночі, а саме: тривалість доби, положення сонця відносно горизонталі/вертикалі та паузу.

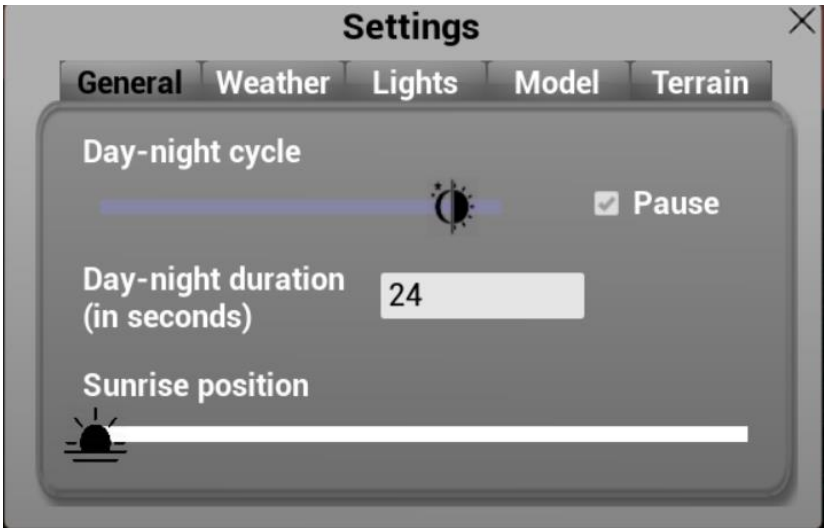

Рисунок 3.11 – Вигляд вкладки “General”

2. Друга вкладка має назву “Weather” і містить в собі налаштування погодних умов, а саме: інтенсивність дощу, інтенсивність снігу, інтенсивність туману та дальність бачення, інтенсивність хмар.

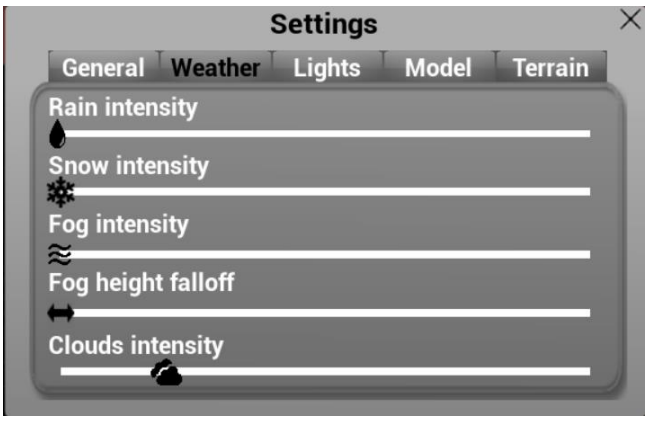

Рисунок 3.12 – Вигляд вкладки “Weather”

3. Третя вкладка має назву “Terrain” і містить в собі налаштування підлоги, а саме: висоту рівня(залежить від експортованих рівнів) та зсув підлоги.

Рисунок 3.13 – Вигляд вкладки “Terrain”

4. Четверта вкладка має назву “Lights” і містить в собі налаштування світла, а саме: тип глобального джерела світла та регулятори та яскравість світла/екватора/землі.

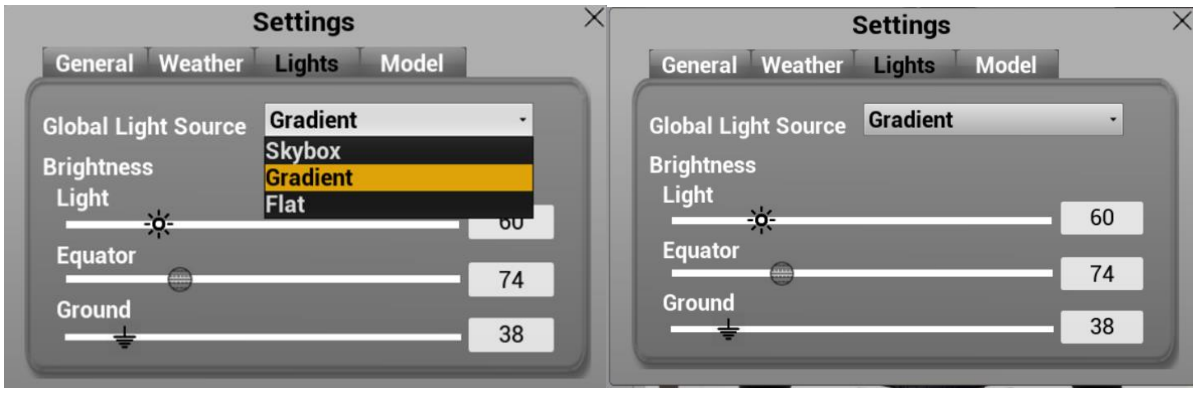

Рисунок 3.14 – Вигляд вкладки “Lights”

5. П'ята вкладка має назву “Model” і містить в собі слайдер за допомогою якого можнліво змінювати розміри(висоту) персонажа, значення висоти відображається в футах чи метрах.

Рисунок 3.15 – Вигляд вкладки “ Model ”

3.5. Пакування гри

Перед тим, як проект Unreal може бути розповсюджений серед користувачів, він повинен бути належним чином упакований. Упаковка гарантує, що весь код і вміст є актуальними і в належному форматі для запуску на бажаній цільовій платформі.

Кілька етапів виконуються під час процесу упаковки. Спочатку буде скомпільовано вихідний код проекту. Після того, як код буде скомпільований, весь необхідний вміст буде перетворений або "приготований" у формат, який може використовуватися цільовою платформою. Після цього скомпільований код і приготовлений вміст будуть вбудовані в набір розповсюджуваних файлів, таких як інсталятор [21].

Перш ніж упакувати гру, спочатку потрібно встановити мапу за замовчуванням гри, яка буде завантажуватися, коли починається запакована гра. Якщо не встановити карту і використовувати порожній проект, можна побачите лише чорний екран, коли починається запакована гра. Якщо використовувати один з шаблонів, як і шаблон першої особи, початкова карта буде завантажена.

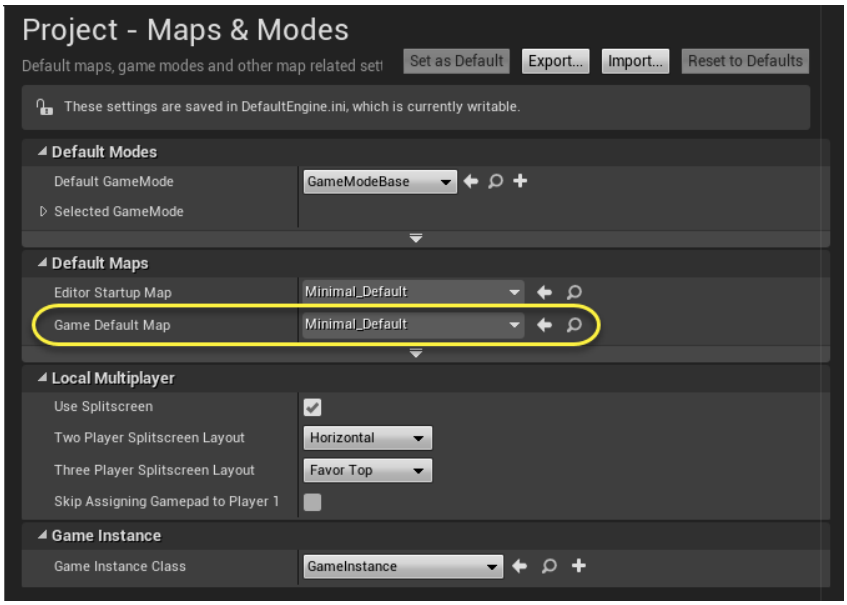

Рисунок 3.16 – Встановлення налаштувань карти

У головному меню Файл є підменю під назвою Пакетний проект. Це підменю надає список всіх підтримуваних платформ, для яких можна упакувати проект.

Буде представлено діалогове вікно для вибору цільового каталогу. Якщо пакунок завершиться успішно, цей каталог буде містити пакетний проект.

Після підтвердження цільового каталогу буде ініційовано фактичний процес, який упаковує проект для вибраної платформи. Оскільки пакування може займати багато часу, цей процес виконується у фоновому режимі, тому можна продовжувати використовувати редактор

Інструмент автоматизації дозволяє готувати і упаковувати гру за допомогою командного рядка, і оскільки всі операції збирання виконуються за допомогою UAT, їх можна запустити безпосередньо в командному рядку за допомогою пакетного файлу RunUAT, якщо надано відповідні аргументи.

Файли RunUAT можна знайти в Engine/Build/BatchFiles. Для Windows використовується файл RunUAT.bat, а для Mac/Linux використовується RunUAT.sh.

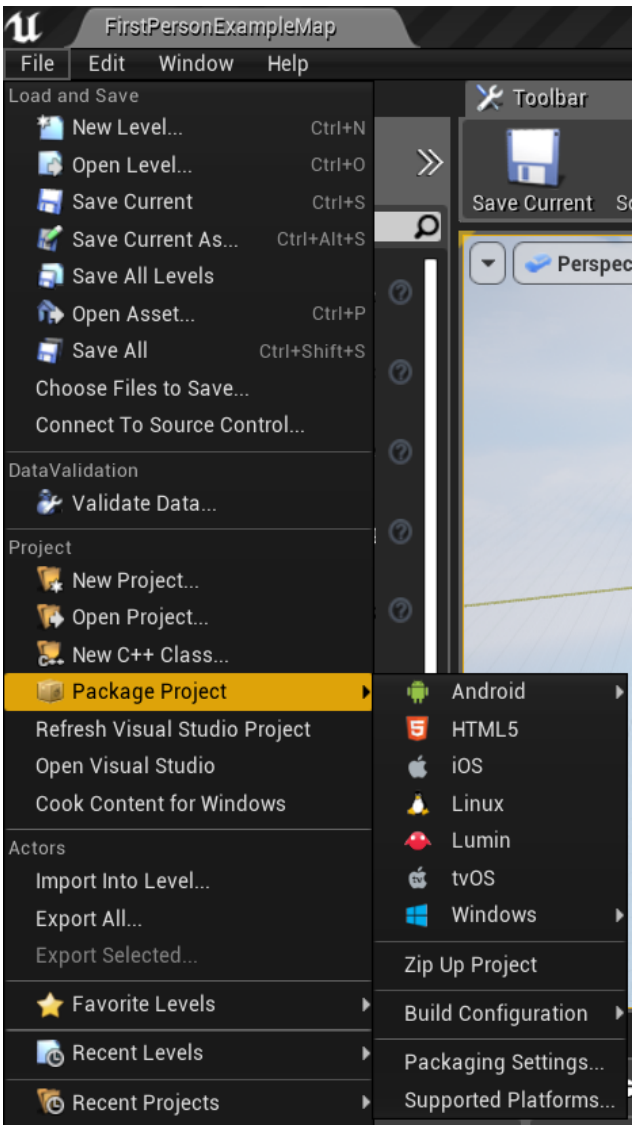

Рисунок 3.17 – Пакування гри за допомогою UnrealEditor

Основні моменти:

- Якщо використовуються незв'язані об'єкти, треба додати їх a) за допомогою командного рядка або b), створивши рівень з усіма об'єктами, які потрібно опублікувати в грі (не рекомендується).
- Кукінг – це процес видалення небажаних файлів з остаточної збірки.
- Якщо будується виділений сервер,потрібно мати компільовану версію Unreal або не працюватиме.
- Команда -build додається, якщо використовується Source Version з Github.

- Якщо використовується прапорець -allmaps, [AllMaps] з дійсним +Map=\Game\Maps\Map.umap слід додати до DefaultEditor.ini
- Конкретні карти можуть бути побудовані (необхідно видалити -allmaps прапор) за допомогою -maps = Map1+Map2+Map3.

Отже, було використано таку команду:

```
RunUAT BuildCookRun -
project="full_project_path_and_project_name.uproject" -noP4 -platform=Win64
-clientconfig=Development -serverconfig=Development -cook -allmaps -build -
stage -pak -archive -archivedirectory="Output Directory"
```

3.6. Використання готового продукту

У результаті отримано інсталятор:

1. Запускаємо інсталлятор:

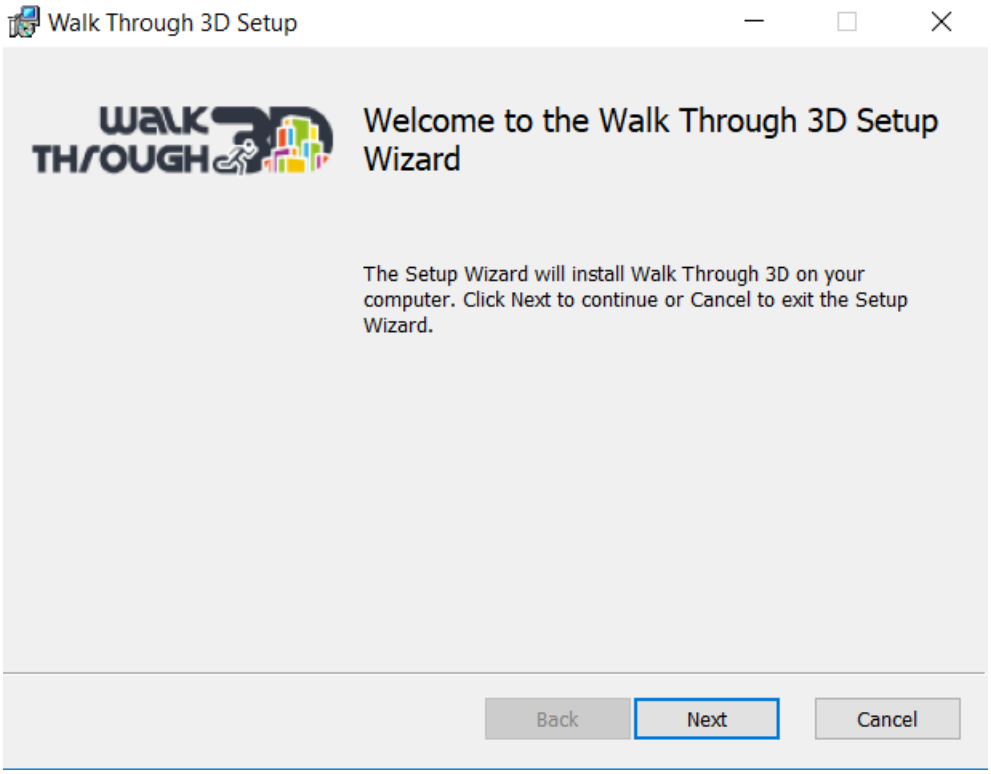

Рисунок 3.18 – Стартове вікно інсталятора

2. Приймаємо умови договору:

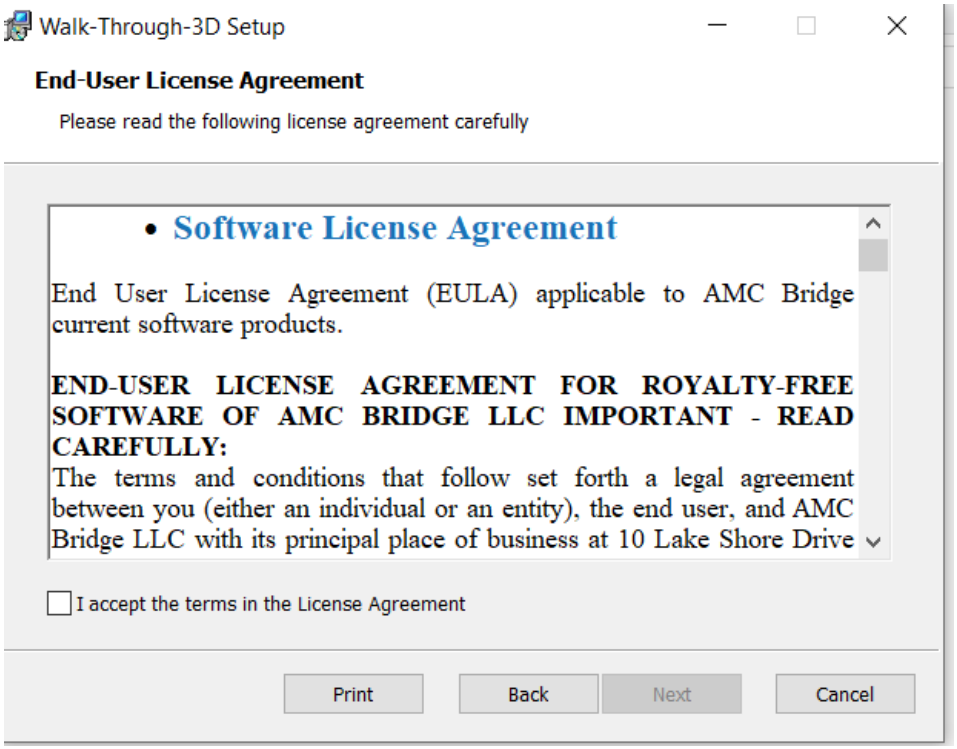

Рисунок 3.19. Вікно ліцензійної угоди кінцевого користувача

3. Обираємо місце розташування програми:

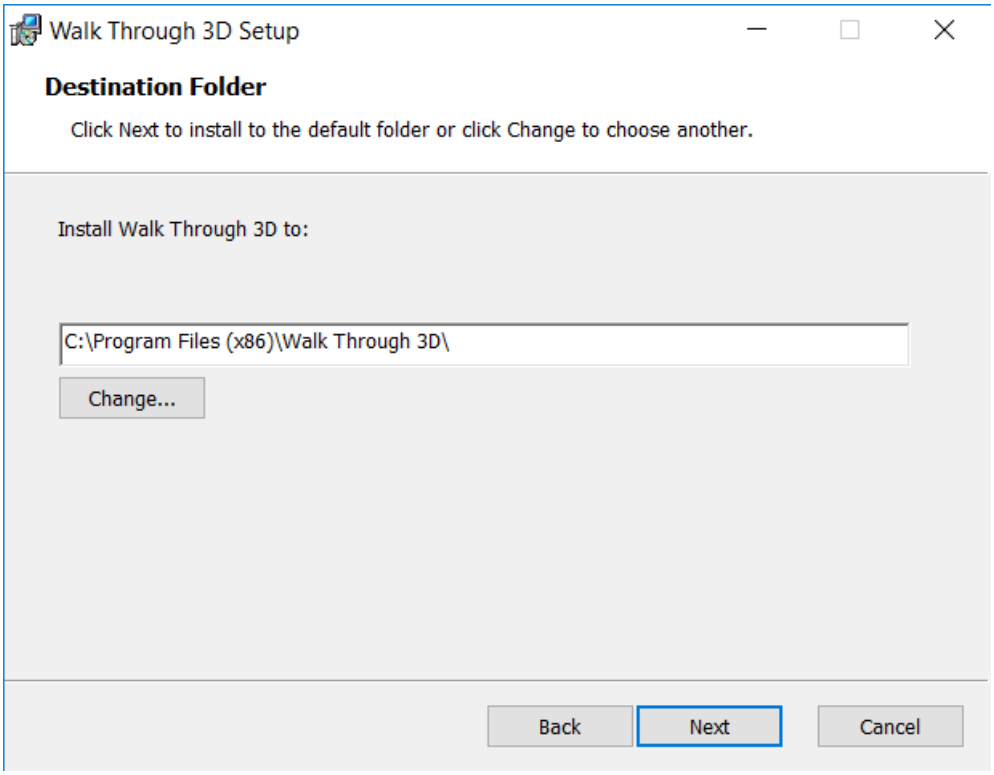

Рисунок 3.20. Вікно теки призначення

4. Очікуємо завантаження програми:

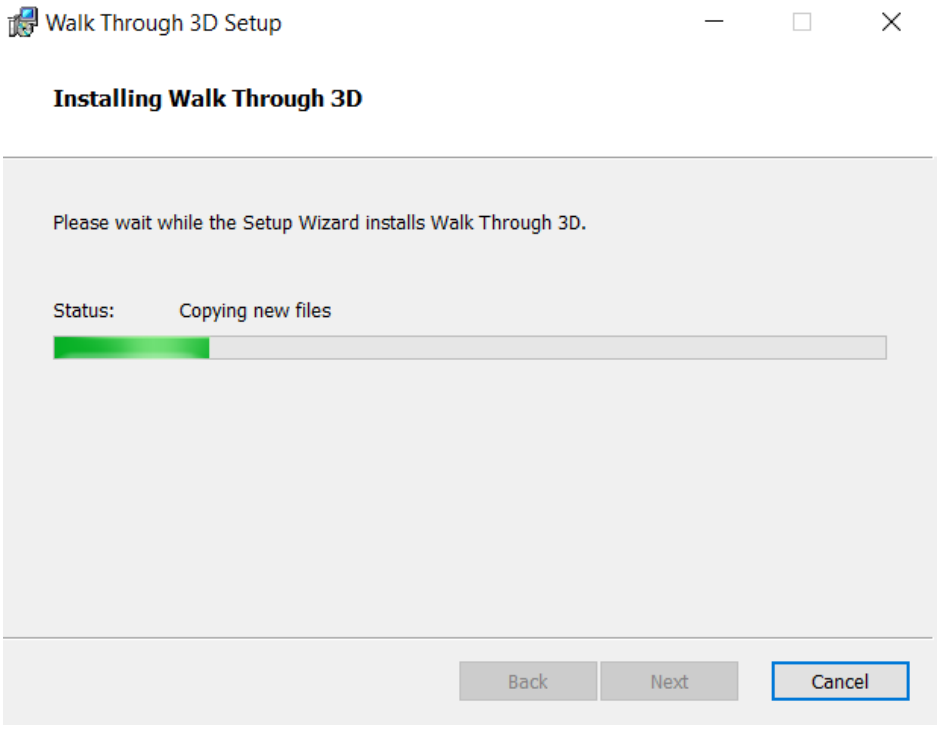

Рисунок 3.21. Вікно інсталяції програми

Якщо на ком'ютері немає основної програми Revit, то інсталяція прерветься:

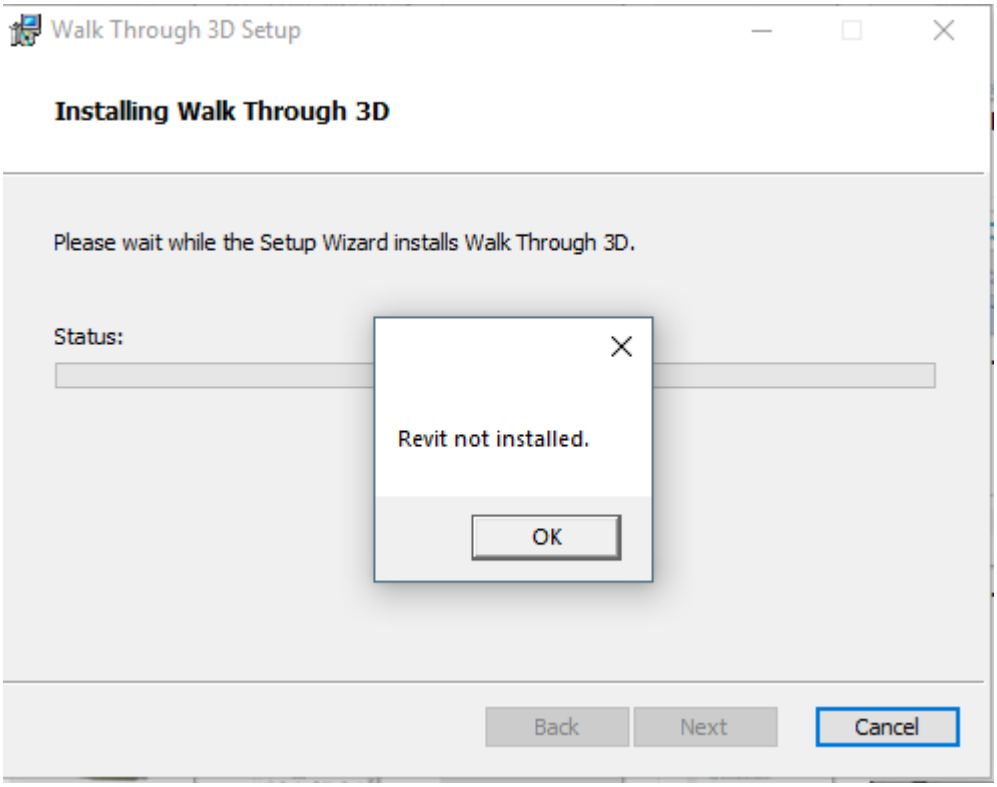

Рисунок 3.22 – Вікно прерваної інсталяції

5. При вдалому завантаженні отримаємо:

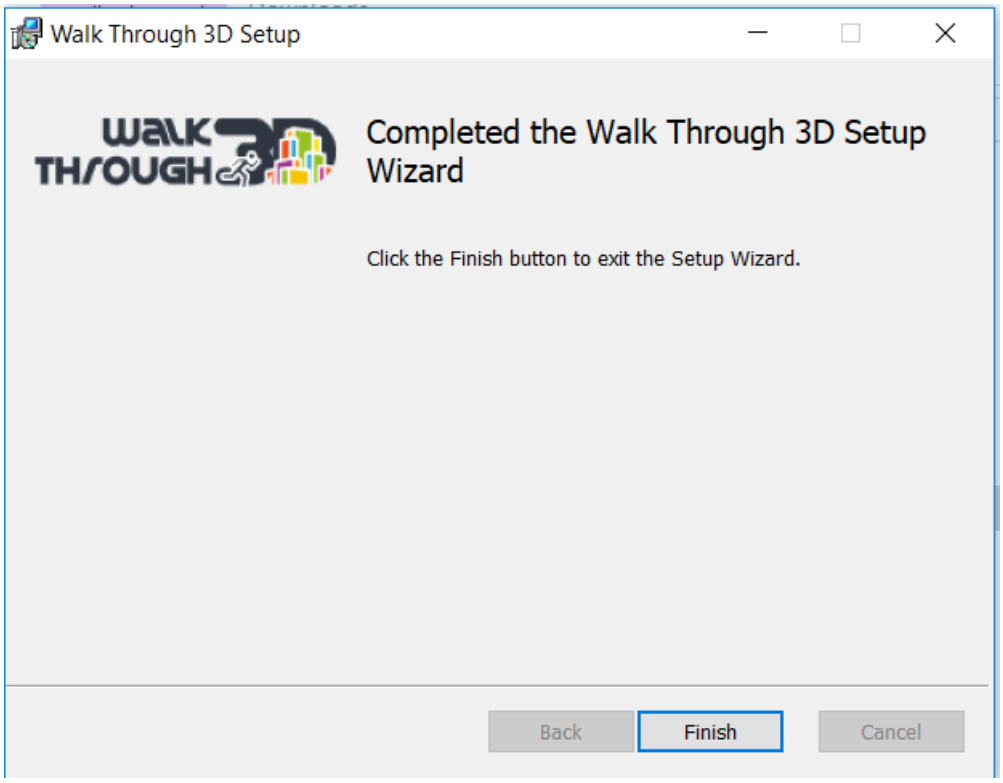

Рисунок 3.23 – Вікно вдалої інсталяції

При невдалому завантаженні:

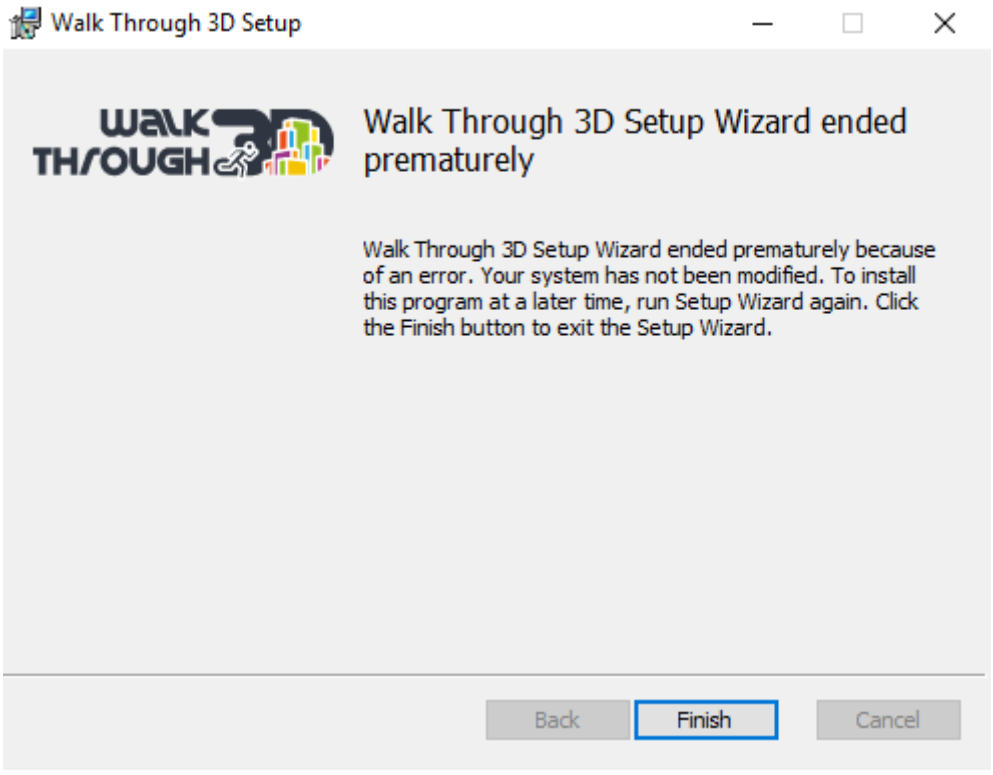

Рисунок 3.24 – Вікно невдалої інсталяції

Експлуатація програми після установки:

- Встановлена програма буде відображатись в Apps&features комп'ютера:

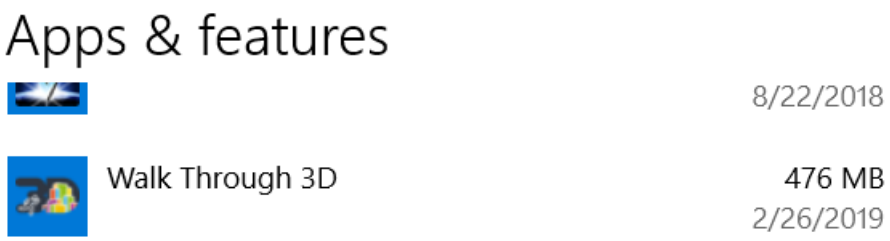

Рисунок 3.25. Зображення наявності продукту на комп’ютері

- Плагін буде відображатись на графічній панелі будь-якої версії Revit:

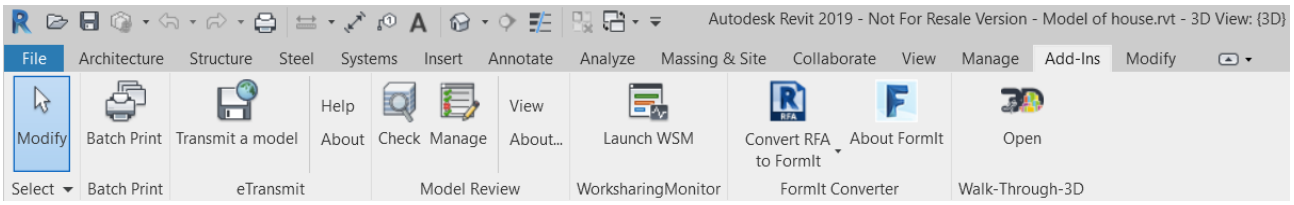

Рисунок.3.26. Розміщення плагіну в програмі Revit

- Для експорту моделі потрібно натиснути кнопку “Open”:

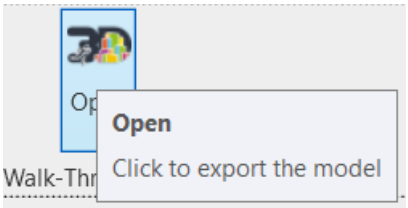

Рисунок 3.27. Кнопка запуску плагіну

- З'явиться вікно експорту:

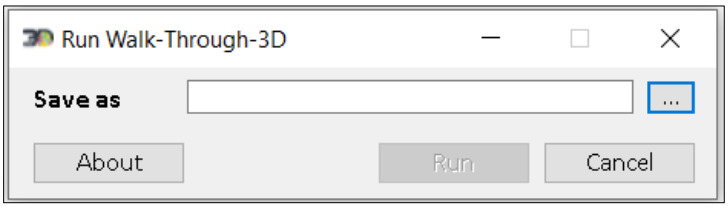

Рисунок 3.28. Вікно експорту

- Далі потрібно натиснути кнопку “…” чи самостійно ввести шлях до теки:

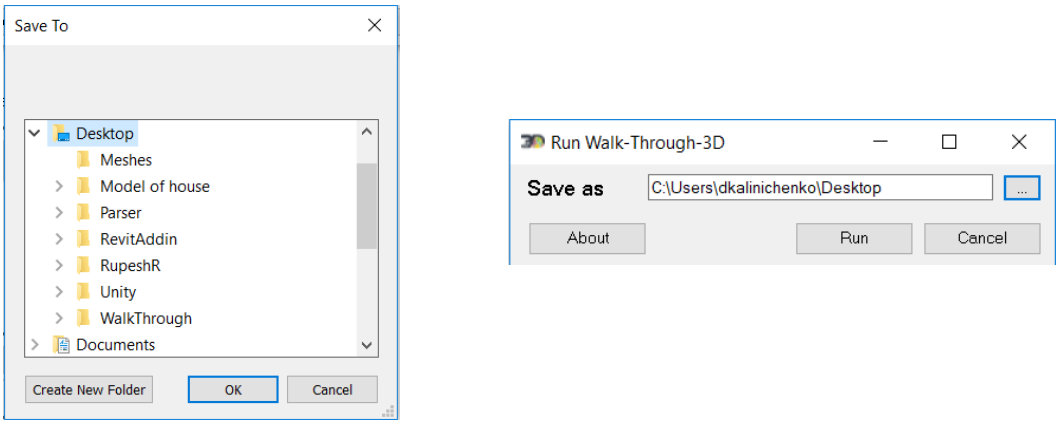

Рисунок 3.29. Вибір шляху експорту

- Відображення експорту почнеться з відображення на нижньому індикаторі Revit`a, а потім з’явиться індикатор запису плагіну:

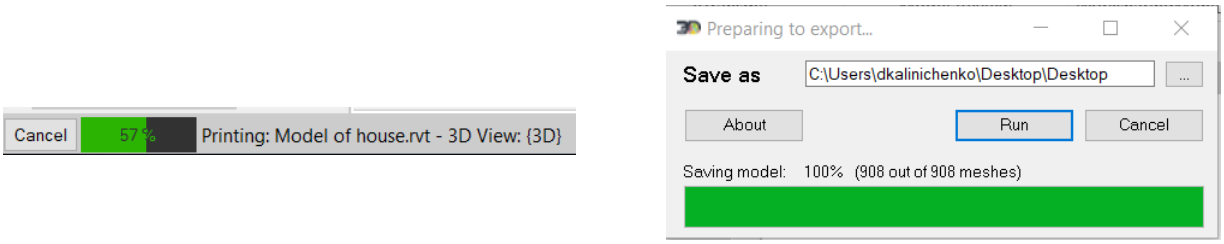

Рисунок 3.30. Індикатори експорту

При вдалому експорті з’явиться повідомлення:

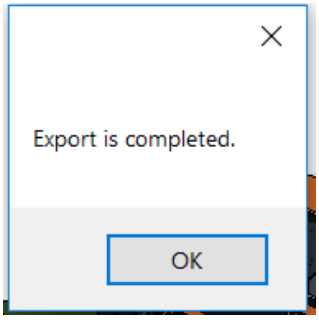

Рисунок.3.31. Повідомлення про завершення експорту

- В обраному шляху з’явиться папка з файлами експорту та їхнього запуску:

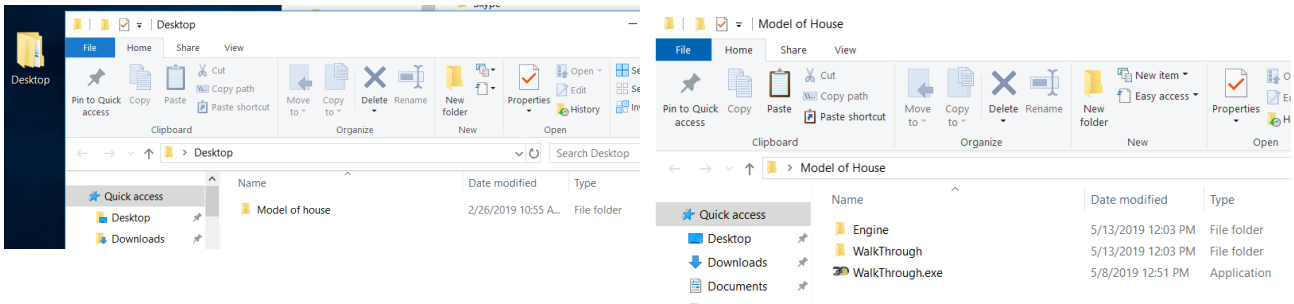

Рисунок 3.32. Папка з експортованими файлами

- Після відкриття .exe файлу з’явиться вікно завантаження моделі:

Рисунок 3.33. Вікно завантаження

- Після вдалого завантаження з’явиться 3D-модель:

Рисунок 3.34. Вікно завантаженої моделі

За декілька не складних кроків, ми можемо експортувати модель з Revit. На зображенні (рисунок 3.34) ми бачимо готову експортовану модель,яка зупущена в ігровому рушії Unreal Engine 4.

ВИСНОВКИ

У результаті виконання випускної роботи було проведено огляд останніх досліджень та публікацій, аналіз створення додатків типу Walk-Through 3D.

Було сформульовано мету та задачі дослідження, сформульовані вимоги до створеного додатку. В якості засобів реалізації додатку обрані мови .NET Framework та C++ також було прийнято рішенння використовувати середовище для швидкого програмування Visual Studio 2017, для розробки інсталятора для встановлення додатку обрано Wix Toolset, в якості програми для створення 3D - моделі було обрано Autodesk Revit, Unreal Engine 4 було обрано для дослідження експортованих 3D-моделей.

Було проведено структурно-функціональне моделювання роботи додатку, планування робіт з реалізації IT-проекту, організаційну структуру та можливі ризики під час виконання проєкту.

Визначено послідовність дій створення додатку для якісної та надійної роботи.

Розроблено інсталятор,який встановлює додаток для Autodesk Revit, що надає можливість користувачам експортувати, створену 3D-модель, зберігаючи всі матеріали і текстури, та запустити її в Unreal Engine 4 для детального дослідження від 1-ї та 3-ї особи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 3D-моделювання [Електронний ресурс] // wikipedia.org – Режим доступу: https://uk.wikipedia.org/wiki/3D-моделювання.

2. Методи промислового користування [Електронний ресурс] // wikipedia.org – Режим доступу http://rabota-ua.com.ua/qa/2019/12/28/uk/gde-3d-so-ce-take-de-vikoristovuut-ob39emne-trivimirne-modeluvanna-programi-dla-3d-konstruuvanna-na-zwsoftru.html.

3. Plug-In [Електронний ресурс] // wikipedia.org – Режим доступу: https://uk.wikipedia.org/wiki/Плагін.

4. Autodesk Knowledge Network [Електронний ресурс] // knowledge.autodesk.com –Режим доступу: https://forums.autodesk.com/.

5. Revit API BIM [Електронний ресурс] // shen.ua/tendentsii-rynka / – Режим доступу: https://shen.ua/tendentsii-rynka/tehnologiya-bim-yak-vazhlyvyj-element-fm/

6. Revit API тривимірне моделювання [Електронний ресурс] lemberg-academy.com / – Режим доступу: https://lemberg-academy.com/course/revit

7. Unreal Engine API [Електронний ресурс] //api.unrealengine.com – Режим доступу: http://api.unrealengine.com/INT/API/.

8. Unreal Engine [Електронний ресурс] // www.unrealengine.com – Режим доступу: https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

9. Візуальний редактор [Електронний ресурс] // wikipedia.org – Режим доступу: https://uk.wikipedia.org/wiki/Візуальний_редактор .

10. UV mapping [Електронний ресурс] // wikipedia.org – Режим доступу: https://uk.wikipedia.org/wiki/UV_mapping .

11. The Basic Plug-in [Електронний ресурс] // knowledge.autodesk.com – Режим доступу: https://knowledge.autodesk.com/support/revit-products/learn-explore/caas/simplecontent/content/lesson-1-the-basic-plug.html .

12. BinaryWriter Class [Електронний ресурс] // docs.microsoft.com – Режим доступу: https://docs.microsoft.com/en-

us/dotnet/api/system.io.binarywriter?view=netframework-4.7.2 .

13. Material [Електронний ресурс] // thebuildingcoder.typepad.com – Режим доступу: https://thebuildingcoder.typepad.com/blog/2017/11/modifying-material-visual-appearance.html.

16. CustomActions [Електронний ресурс] // www.add-in-express.com – Режим доступу: https://www.add-in-express.com/creating-addins-blog/2014/01/29/create-wix-custom-actions/.

17. WixInstaller [Електронний ресурс] // www.mesta-automation.com – Режим доступу: https://www.mesta-automation.com/create-an-installer-for-c-sharp-with-wix/.

18. Reinterpret_cast conversion [Електронний ресурс] // en.cppreference.com – Режим доступу: https://en.cppreference.com/w/cpp/language/reinterpret_cast.

20. Packaging Projects [Електронний ресурс] // docs.unrealengine.com – Режим доступу:

https://docs.unrealengine.com/en-us/Engine/Basics/Projects/Packaging.

21. Build Operations: Cook, Package, Deploy, and Run [Електронний ресурс] // docs.unrealengine.com – Режим доступу: https://docs.unrealengine.com/en-US/Engine/Deployment/BuildOperations.

22. UProceduralMeshComponent [Електронний ресурс] // api.unrealengine.com – Режим доступу: https://api.unrealengine.com/INT/API/Plugins/ProceduralMeshComponent.

Додаток А. Лістинги класів IExternalCommand, IExternalApplication та маніфесту WalkThrough3D.addin

Клас IExternalCommand.

```
Using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;

namespace WalkThrough3D
{


    [Transaction(TransactionMode.Manual)]
    public class AddinCommands : IexternalCommand
    {

        public Autodesk.Revit.UI.Result Execute(ExternalCommandData
commandData, ref string message,
            ElementSet elements)
        {

            UIApplication uiApp = commandData.Application;
            Document doc = uiApp.ActiveUIDocument.Document;

            System.Windows.Forms.Form form = new Settings(doc, uiApp);


            form.ShowDialog();
            return Autodesk.Revit.UI.Result.Succeeded;
        }

    }
}
```

Клас IExternalApplication.

```
using System;
using System.Reflection;
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using System.Windows.Media.Imaging;
using NLog.Config;
using NLog.Targets;
using NLog;
using System.IO;

namespace WalkThrough3D
{

    [Transaction(TransactionMode.Manual)]
    [Regeneration(RegenerationOption.Manual)]
    public class WalkThroughAddin : IExternalApplication
    {
        const string CcmdExporter = "cmdExporter";
        const string CClassName = "WalkThrough3D.AddinCommands";

        public Result OnShutdown(UIControlledApplication application)
        {
            return Result.Succeeded;
```

```
                    }

                    public Result OnStartup(UIControlledApplication application)
                    {
                        string assemblyPath = Assembly.GetExecutingAssembly().Location;
                        LoggingConfiguration config = new LoggingConfiguration();
                        FileTarget fileTarget = new FileTarget();
                        fileTarget.FileName =
Path.Combine(Path.GetDirectoryName(assemblyPath),"Info.log");
                        fileTarget.Layout = "${message}";
                        LoggingRule rule = new LoggingRule("*", LogLevel.Info, fileTarget);
                        config.LoggingRules.Add(rule);
                        config.AddTarget("Info", fileTarget);
                        LogManager.Configuration = config;
                        RibbonPanel ribbonPanel =
application.CreateRibbonPanel(Resources.Resources.DescriptionPanel);


                        PushButtonData buttonData= new PushButtonData(CcmdExporter,
                            Resources.Resources.Convert,
                            assemblyName: assemblyPath,
                            className: CClassName);


                        PushButton pushButton = ribbonPanel.AddItem(buttonData) as
PushButton;
                        pushButton.ToolTip = Resources.Resources.Tip;
                        BitmapImage butImage = new BitmapImage(new
Uri("pack://application:,,,/WalkThrough3D;component/Resources/favicon.png"));
                        pushButton.LargeImage = butImage;
                        return Result.Succeeded;
                    }

                }
            }
```

Маніфест WalkThrough3D.addin.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Application">
    <Name>WalkThrough3D</Name>
    <FullClassName>WalkThrough3D.WalkThroughAddin</FullClassName>
    <Text>Walk Through 3D</Text>
      <Assembly>WalkThrough3D.dll</Assembly>
    <AddInId>9EEB655C-1721-421A-B066-B2A0f038CD8C</AddInId>
    <VendorId>AMC Bridge</VendorId>
  </AddIn>
</RevitAddIns>
```

Додаток Б. Лістинги допоміжних класів проекту DataStructure

Клас AFacet.

```
using System.IO;

namespace DataStructure
{
   public class AFacet : ISerializable
    {
       public float V1 { get; set; }
       public float V2 { get; set; }
       public float V3 { get; set; }
       readonly private float[] vertice = new float[3];
        public AFacet(float v1, float v2, float v3)
        {
            vertice[0] = V1 = v1;
            vertice[1] = V2 = v2;
            vertice[2] = V3 = v3;
        }
        public void WriteData(BinaryWriter file)
        {
            foreach (float ver in vertice)
            {
                file.Write(ver);
            }

        }
    }
}
```

Клас APoint.

```
using System.IO;

namespace DataStructure
{
    public class APoint : ISerializable
    {
        //Increases the points from the revit to work in the UE4 measurement system.
        private const int multiplier = 40;
        public double X { get; set; }
        public double Y { get; set; }
        public double Z { get; set; }


        public APoint(double x, double y, double z)
        {
            X = x;
            Y = y;
            Z = z;
        }
        public void WriteData(BinaryWriter file)
        {
            file.Write(X*multiplier);
            file.Write(Y*multiplier);
            file.Write(Z*multiplier);
        }
    }

}
```

Клас AUV.

```
using System.IO;

namespace DataStructure
{
    public class AUV : ISerializable
    {
        public double U { get; set; }
        public double V { get; set; }
        public AUV(double u, double v)
        {
            U = u;
            V = v;
        }

        public void WriteData(BinaryWriter file)
        {
            file.Write(U);
            file.Write(V);
        }
    }
}
```

Клас AMaterial.

```
using System;
using System.IO;

namespace DataStructure
{
    public sealed class AMaterial : ISerializable, IEquatable<AMaterial>
    {
        public string Name { get; set; }
        public int R { get; set; }
        public int G { get; set; }
        public int B { get; set; }

        public string Textures { get; set; }
        public string Bumps { get; set; }
        public int Transparency { get; set; }
        readonly private int[] colors = new int[3];
        public AMaterial(string name, int r, int g, int b, int transpar, string
textures, string bump)
        {
            Name = name;
            colors[0] = R = r;
            colors[1] = G = g;
            colors[2] = B = b;
            Transparency = transpar;

            Textures = (textures != null && textures != "") ? textures :
"null";
            Bumps = (bump != null && bump != "") ? bump : "null";
        }
        public void WriteData(BinaryWriter file)
        {

            file.Write(Name + "\n");
            file.Write(Textures + "\n");
            file.Write(Bumps + "\n");
            foreach (var col in colors)
            {
```

```
                    file.Write(col);
                }
                file.Write(Transparency);
            }

            public bool Equals(AMaterial other)
            {
                if (other == null) return false;
                return Name.Equals(other.Name) && R.Equals(other.R) &&
G.Equals(other.G) && B.Equals(other.B);
            }
        }
    }
```

Клас APolymesh.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;

namespace DataStructure
{
    public class APolymesh : ISerializable
    {
        public List<AFacet> PolymeshVertices { get; set; }
        public List<APoint> PolymeshPoints { get; set; }
        public List<AUV> PolymeshUvModels { get; set; }
        public List<APoint> PolymeshNormals { get; set; }
        public string NameOfMaterial { get; set; }
        public string NameOfMesh { get; set; }
        public bool GetCollision { get; set; }

        public APolymesh()
        {
            PolymeshVertices = new List<AFacet>();
            PolymeshPoints = new List<APoint>();
            PolymeshUvModels = new List<AUV>();
            PolymeshNormals = new List<APoint>();
        }


        public void WriteData(BinaryWriter file)
        {
            try
            {
                file.Write(PolymeshVertices.Count);
                foreach (var ver in PolymeshVertices)
                {
                    ver.WriteData(file);
                }
                file.Write(PolymeshPoints.Count);
                foreach (var point in PolymeshPoints)
                {
                    point.WriteData(file);
                }
                file.Write(PolymeshUvModels.Count);
                foreach (var mod in PolymeshUvModels)
                {
                    mod.WriteData(file);
                }
                file.Write(PolymeshNormals.Count);
                foreach (var nor in PolymeshNormals)
                {
```

```
                        nor.WriteData(file);
                    }
                    file.Write(GetCollision);
                    file.Write(NameOfMaterial + "\n");
                }
                catch (Exception err)
                {
                    Debug.Print(err.ToString());
                }

            }
        }
    }
```

Інтерфейс ISerializable.

```
namespace DataStructure
{
    interface ISerializable
    {
        void WriteData(System.IO.BinaryWriter file);
    }
}
```

Додаток В. Лістинг класу Exporter

Current project is the property of company AMC Bridge

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.RegularExpressions;
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.DB;
using Autodesk.Revit.DB.Visual;
using DataStructure;

namespace Exporter
{
    public class Exporter : IPhotoRenderContext
    {
        readonly Document doc;
        int counter;
        string nameOfRPC;
        bool collision;
        APolymesh mesh;
        string fullPath = "C:\\Program Files (x86)\\Common Files\\Autodesk
Shared\\Materials\\Textures\\3\\Mats\\";

        public List<APoint> Lights { get; set; }
        public Dictionary<APoint, string> RPCs { get; set; }
        public List<APolymesh> Meshes { get; set; }
        public List<APolymesh> MergeMeshes { get; set; }
        public List<AMaterial> Materials { get; set; }
        public List<string> NameOfMeshes { get; set; }
        public int Count { get; set; }
        public string NameOfMaterials { get; set; }
        public string NameOfMesh { get; set; }
        public bool Floor { get; set; }
        public Dictionary<double, string> Levels { get; set; }

        private readonly Stack<Transform> m_TransformationStack = new
Stack<Transform>();
        public Exporter(Document _doc, Application app)
        {
            doc = _doc;
            Meshes = new List<APolymesh>();
            MergeMeshes = new List<APolymesh>();
            Materials = new List<AMaterial>();
            Lights = new List<APoint>();
            RPCs = new Dictionary<APoint, string>();
            NameOfMeshes = new List<string>();
            Levels = new Dictionary<double, string>();
        }
        public void Finish()
        {
            GetLevels(doc);
            Count = counter + Materials.Count + NameOfMeshes.Count +
Lights.Count + RPCs.Count + Levels.Count*2;
        }
        private void GetLevels(Document document)
        {
            int levelNumber = 0;
            FilteredElementCollector collector = new
FilteredElementCollector(document);
            ICollection<Element> collection =
collector.OfClass(typeof(Level)).ToElements();
            foreach (Element e in collection)
            {
```

```
                            Level level = e as Level;

                            if (null != level)
                            {
                                levelNumber++;
                                Levels.Add(level.Elevation, level.Name);
                            }
                        }
                    }
            public List<APolymesh> MergeMesh(List<APolymesh> mergMeshes)
                    {
                        int first = 0;
                        while (first < mergMeshes.Count)
                        {
                            int second = first + 1;
                            while (second < mergMeshes.Count)
                            {
                                if (mergMeshes[first].NameOfMaterial ==
mergMeshes[second].NameOfMaterial && mergMeshes[first].GetCollision ==
mergMeshes[second].GetCollision)
                                {
                                    Count = mergMeshes[first].PolymeshPoints.Count;

mergMeshes[first].PolymeshPoints.AddRange(mergMeshes[second].PolymeshPoints);

mergMeshes[first].PolymeshUvModels.AddRange(mergMeshes[second].PolymeshUvModels);

mergMeshes[first].PolymeshNormals.AddRange(mergMeshes[second].PolymeshNormals);
                                    foreach (var fac in
mergMeshes[second].PolymeshVertices)
                                    {
                                        mergMeshes[first].PolymeshVertices.Add(new
AFacet(fac.V1 + Count, fac.V2 + Count, fac.V3 + Count));
                                    }
                                    mergMeshes.RemoveAt(second);
                                }
                                else
                                {
                                    second++;
                                }
                            }
                            first++;
                            Count = 0;
                        }
                        return mergMeshes;
                    }
                    public bool IsCanceled()
                    {
                        return false;
                    }

                    public RenderNodeAction OnElementBegin(ElementId elementId)
                    {
                        Element element = doc.GetElement(elementId);
                        NameOfMesh = Guid.NewGuid().ToString();
                        FamilyInstance elFamInst = element as FamilyInstance;
                        if (elFamInst != null &&
elFamInst.Symbol.Family.Name.Contains("RPC"))
                        {
                            if (elFamInst.Symbol.Family.Name.Contains("Tree"))
                            {
                                nameOfRPC = "Tree";
                            }
                            else if (elFamInst.Symbol.Family.Name.Contains("Car") ||
```

```
elFamInst.Symbol.Family.Name.Contains("Beetle"))
                            {
                                nameOfRPC = "Car";
                            }
                            else
                            {
                                return RenderNodeAction.Skip;
                            }
                        }
                        if (element.Category != null)
                        {
                            BuiltInCategory builtInCategory =
(BuiltInCategory)element.Category.Id.IntegerValue;
                            collision = (builtInCategory != BuiltInCategory.OST_Doors);
                            if(builtInCategory == BuiltInCategory.OST_Topography ||
builtInCategory==BuiltInCategory.OST_TopographySurface)
                            {
                                Floor = true;
                            }
                        }
                        else
                        {
                            collision = true;
                        }
                        return RenderNodeAction.Proceed;
                    }

                    public void OnElementEnd(ElementId elementId)
                    {
                        if (MergeMeshes.Count != 0)
                        {
                            counter++;
                        }
                        Meshes.AddRange(MergeMesh(MergeMeshes));
                        MergeMeshes.Clear();

                    }
                public RenderNodeAction OnFaceBegin(FaceNode node)
                    {
                        return RenderNodeAction.Proceed;
                    }

                    public void OnFaceEnd(FaceNode node)
                    {
                        // Left empty for the future.
                    }

                    public RenderNodeAction OnInstanceBegin(InstanceNode node)
                    {
                        Transform transform = node.GetTransform();

                        if (m_TransformationStack.Count > 0)
                        {
                            transform = m_TransformationStack.Peek().Multiply(transform);
                        }

                        m_TransformationStack.Push(transform);
                        return RenderNodeAction.Proceed;
                    }

                    public void OnInstanceEnd(InstanceNode node)
                    {
                        m_TransformationStack.Pop();
                    }
```

```
                    public void OnLight(LightNode node)
                    {
                        Transform lightTrf = node.GetTransform();
                        Transform lightWorldTrf =
m_TransformationStack.Peek().Multiply(lightTrf);
                        Lights.Add(new APoint(lightWorldTrf.Origin.X,
lightWorldTrf.Origin.Y, lightWorldTrf.Origin.Z));
                    }

                    public RenderNodeAction OnLinkBegin(LinkNode node)
                    {
                        return RenderNodeAction.Skip;
                    }

                    public void OnLinkEnd(LinkNode node)
                    {
                        // Left empty for the future.
                    }
                    private void CheckTint(Asset asset, ref Color c)
                    {
                        AssetPropertyBoolean tintToggleProp = asset["common_Tint_toggle"]
as AssetPropertyBoolean;
                        AssetPropertyDoubleArray4d tintColorProp =
asset["common_Tint_color"] as AssetPropertyDoubleArray4d;
                        CheckColor(tintToggleProp != null, tintColorProp, ref c,
tintToggleProp.Value);
                    }
                    private void CheckColor(bool toggle,AssetPropertyDoubleArray4d check,
ref Color color, bool condition)
                    {
                        if (toggle && check != null && check.IsValidObject && condition)
                        {
                            Color c = new Color((byte)(check.GetValueAsDoubles()[0] * 255),
(byte)(check.GetValueAsDoubles()[1] * 255), (byte)(check.GetValueAsDoubles()[2] * 255));
                            color = c;
                        }
                    }
            private void CheckMaps (Asset asset,ref string genericMap, ref string bumpMap)
                    {
                        string intermediate = null;
                        if (genericMap == null && bumpMap == null)
                        {
                            genericMap = genericMap ?? GetFull(asset, "generic_diffuse") ??
GetTextures(asset, "unifiedbitmap_Bitmap", "UnifiedBitmapSchema");
                            bumpMap = bumpMap ?? GetFull(asset, "generic_bump_map") ??
GetTextures(asset, "bumpmap_Bitmap", "BumpMapSchema");
                        }
                        if (genericMap != null && (genericMap.Contains("refl") ||
genericMap.Contains("bump")))
                        {
                            intermediate = genericMap;
                            genericMap = (bumpMap != null && !(bumpMap.Contains("refl") ||
bumpMap.Contains("bump"))) ? bumpMap:null;
                            if (bumpMap==null)
                            {
                                bumpMap = intermediate;
                            }
                        }
                    }
                    private Color TypesOfSchemas(Asset asset, ref string genericMap, ref
string bumpMap, string image, string bump, string color)
                    {
                        Color c = Color.InvalidColorValue;
                        genericMap =genericMap ?? GetFull(asset, image) ??
```

```
GetTextures(asset, "unifiedbitmap_Bitmap", "UnifiedBitmapSchema");
                        bumpMap = bumpMap ?? GetFull(asset, bump) ?? GetTextures(asset,
"bumpmap_Bitmap", "BumpMapSchema");
                        AssetPropertyDoubleArray4d property = asset[color] as
AssetPropertyDoubleArray4d;
                        CheckColor(true,property,ref c,true);
                        CheckTint(asset, ref c);
                        CheckMaps(asset, ref genericMap, ref bumpMap);
                        return c;
                    }
                   private Color CheckColor(Asset asset, out string genericMap, out string
bumpMap)
                    {
                        Color color = Color.InvalidColorValue;
                        genericMap = bumpMap = null;
                        switch (asset.Name)
                        {
                            case "CeramicSchema":
                                bumpMap = GetFull(asset, "ceramic_pattern_map");
                                color = TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"ceramic_color", "ceramic_bump_map", "ceramic_color");
                                break;
                            case "ConcreteSchema":
                                bumpMap = bumpMap ?? GetFull(asset, "concrete_bm_map");
                                color = TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"concrete_color", "concrete_bump_map", "concrete_color");
                                break;
                            case "GenericSchema":
                                color = TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"generic_diffuse", "generic_bump_map", "generic_diffuse");
                                break;
                            case "GlazingSchema":
                                AssetPropertyDoubleArray4d glazing =
asset["glazing_transmittance_map"] as AssetPropertyDoubleArray4d ??
asset["glazing_transmittance_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(true, glazing, ref color, true);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "HardwoodSchema":
                                genericMap = GetFull(asset, "hardwood_color") ??
GetFull(asset, "hardwood_imperfections_shader") ?? GetTextures(asset,
"unifiedbitmap_Bitmap", "UnifiedBitmapSchema");
                                AssetPropertyInteger hardWoodtoggle =
asset["hardwood_tint_enabled"] as AssetPropertyInteger;
                                AssetPropertyDoubleArray4d hardWood =
asset["hardwood_tint_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(hardWoodtoggle != null, hardWood, ref color,
hardWoodtoggle.Value == 1);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "MasonryCMUSchema":
                                color = TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"masonrycmu_color", "masonrycmu_pattern_map", "masonrycmu_color");
                                break;
                            case "MetalSchema":
                                bumpMap = GetFull(asset, "generic_diffuse") ??
GetTextures(asset, "unifiedbitmap_Bitmap", "UnifiedBitmapSchema");
                                AssetPropertyDoubleArray4d metal = asset["metal_color"] as
AssetPropertyDoubleArray4d;
                                CheckColor(true, metal, ref color, true);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
```

```
                            case "MetallicPaintSchema":
                                AssetPropertyInteger metallicPainttoggle =
asset["metallicpaint_flecks"] as AssetPropertyInteger;
                                AssetPropertyDoubleArray4d metallicPaint =
asset["metallicpaint_flecks_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(metallicPainttoggle != null, metallicPaint, ref
color, metallicPainttoggle.Value == 1);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "MirrorSchema":
                                AssetPropertyBoolean mirrortoggle =
asset["mirror_color_by_object"] as AssetPropertyBoolean;
                                AssetPropertyDoubleArray4d mirror =
asset["mirror_tintcolor"] as AssetPropertyDoubleArray4d;
                                CheckColor(mirrortoggle != null, mirror, ref color,
mirrortoggle.Value);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "PlasticVinylSchema":
                                TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"plasticvinyl_color", "plasticvinyl_bump_map", "plasticvinyl_color");
                                break;
                            case "SolidGlassSchema":
                                AssetPropertyDoubleArray4d solidGlass =
asset["solidglass_transmittance_custom_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(true, solidGlass, ref color, true);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "StoneSchema":
                                TypesOfSchemas(asset, ref genericMap, ref bumpMap,
"stone_color", "stone_pattern_map", "stone_color");
                                break;
                            case "WallPaintSchema":
                                AssetPropertyDoubleArray4d wallPaint =
asset["wallpaint_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(true, wallPaint, ref color, true);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            case "WaterSchema":
                                AssetPropertyInteger watertoggle =
asset["water_tint_enable"] as AssetPropertyInteger;
                                AssetPropertyDoubleArray4d water =
asset["water_tint_color"] as AssetPropertyDoubleArray4d;
                                CheckColor(watertoggle != null, water, ref color,
watertoggle.Value == 7);
                                CheckTint(asset, ref color);
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                            default:
                                CheckMaps(asset, ref genericMap, ref bumpMap);
                                break;
                        }

                        return color;
                    }
            private string GetTextures(Asset asset, string property,string schema)
                    {
                        int size = asset.Size;
                        for (int assetIdx = 0; assetIdx < size; assetIdx++)
                        {
```

```
                            AssetProperty aProperty = asset[assetIdx];

                            if (aProperty.NumberOfConnectedProperties < 1)
                            {
                                continue;
                            }
                            Asset connectedAsset = aProperty.GetConnectedProperty(0) as
Asset;

                            if (connectedAsset.Name == schema)
                            {

                                fullPath = (Directory.Exists(fullPath)) ? fullPath :
fullPath.Replace("3", "2");
                                fullPath = (Directory.Exists(fullPath)) ? fullPath :
fullPath.Replace("2", "1");
                                AssetPropertyString path = connectedAsset[property] as
AssetPropertyString;
                                string imagesName = "";
                                string relativePath = path.Value.ToString();
                                if (relativePath == "")
                                {
                                    continue;
                                }
                                Regex condition = new Regex(@"([^\\\/]+)",
RegexOptions.RightToLeft);
                                MatchCollection matches = condition.Matches(relativePath);
                                if (matches.Count > 0)
                                {
                                    imagesName = matches[0].Value;
                                    fullPath = (!File.Exists(fullPath + imagesName)) ?
fullPath.Replace("3", "2") : fullPath;
                                    fullPath = (!File.Exists(fullPath + imagesName)) ?
fullPath.Replace("2", "1") : fullPath;
                                    return (File.Exists(fullPath + imagesName) ? fullPath +
imagesName : null);
                                }
                            }
                        }
                        return null;
                    }

                    private string GetFull(Asset asset1, string propertyName)
                    {
                        const string BitmapPropertyName = "unifiedbitmap_Bitmap";
                        string theValue = "";
                        if (asset1 != null)
                        {
                            AssetProperty property;
                            AssetPropertyString stringProperty;
                            AssetProperty property2;
                            for (int i = 0; i < asset1.Size; i++)
                            {
                                property = asset1[i];
                                if (property != null && property.Name == propertyName)
                                {
                                    IList<AssetProperty> propertiesConnected =
                                          property.GetAllConnectedProperties();

                                    if (propertiesConnected != null &&
propertiesConnected.Count > 0)
                                    {
                                        property2 = propertiesConnected[0];
                                        if (property2.Type == AssetPropertyType.Asset)
                                        {
```

```
                                            Asset asset2 = property2 as Asset;
                                            if (asset2 != null)
                                            {
                                                for (int i2 = 0; i2 < asset2.Size; i2++)
                                                {
                                                    property = asset2[i2];
                                                    if (property != null && property.Name
                                                        == BitmapPropertyName)
                                                    {
                                                        stringProperty = property
                                                          as AssetPropertyString;
                                                        if (stringProperty != null)
                                                        {
                                                            fullPath =
(Directory.Exists(fullPath)) ? fullPath : fullPath.Replace("3", "2");
                                                            fullPath =
(Directory.Exists(fullPath)) ? fullPath : fullPath.Replace("2", "1");
                                                            theValue =
stringProperty.Value;
                                                            Regex condition = new
Regex(@"([^\\\/]+)", RegexOptions.RightToLeft);
                                                            MatchCollection matches =
condition.Matches(theValue);
                                                            if (matches.Count > 0)
                                                            {
                                                                theValue =
matches[0].Value;
                                                                fullPath =
(!File.Exists(fullPath + theValue)) ? fullPath.Replace("3", "2") : fullPath;
                                                                fullPath =
(!File.Exists(fullPath + theValue)) ? fullPath.Replace("2", "1") : fullPath;
                                                                return
(File.Exists(fullPath + theValue) ? fullPath + theValue : null);
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                    break;
                                }
                            }
                        }
                        return null;
                    }
            public void OnMaterial(MaterialNode node)
                    {
                        Color color=new
Color(node.Color.Red,node.Color.Green,node.Color.Blue);
                        Color renderColor = Color.InvalidColorValue;
                        int transp=(int)node.Transparency;
                        string generic, bumps;
                        Material mat = doc.GetElement(node.MaterialId) as Material;
                        AMaterial material;
                        Asset asset = node.GetAppearanceOverride()?? node.GetAppearance();
                        if(asset!=null)
                        {
                            AssetPropertyDoubleArray4d prop = asset["generic_diffuse"] as
AssetPropertyDoubleArray4d;
                            CheckColor(true, prop, ref renderColor, true);
                        }
                        if (mat != null)
                        {
```

```
                            ElementId appearanceAssetId = mat.AppearanceAssetId;
                            AppearanceAssetElement appearanceAssetElem =
doc.GetElement(appearanceAssetId) as AppearanceAssetElement;
                            asset = appearanceAssetElem.GetRenderingAsset();
                            NameOfMaterials = Regex.Replace(mat.Name, @"[\\\///:*?<>|""]",
"");
                            Color c = CheckColor(asset, out generic, out bumps);
                            transp = mat.Transparency;
                            if (!mat.UseRenderAppearanceForShading && c.IsValid)
                            {
                                color = c;
                            }
                            else if (!mat.UseRenderAppearanceForShading &&
renderColor.IsValid)
                            {
                                color = renderColor;
                            }

                            material = new AMaterial(NameOfMaterials, (int)color.Red << 16,
(int)color.Green << 8, (int)color.Blue, transp, generic, bumps);
                            if (!Materials.Contains(material))
                            {
                                Materials.Add(material);
                            }
                        }
                        else
                        {
                            color = (renderColor.IsValid) ? renderColor : color;
                            NameOfMaterials = "null";
                            material = new AMaterial(NameOfMaterials, (int)color.Red << 16,
(int)color.Green << 8, (int)color.Blue, transp, "null", "null");
                            if (!Materials.Contains(material))
                            {
                                Materials.Add(material);
                            }

                        }

                    }

                    public void OnPolymesh(PolymeshTopology node)
                    {
                        mesh = new APolymesh
                        {
                            GetCollision = collision,
                            NameOfMaterial = NameOfMaterials,
                            NameOfMesh = NameOfMesh
                        };
                        if (!NameOfMeshes.Contains(NameOfMesh))
                        {
                            NameOfMeshes.Add(NameOfMesh);
                        }

                        Transform currentTransform = null;
                        IList<XYZ> normal = node.GetNormals();
                        foreach (var ver in node.GetFacets())
                        {
                            mesh.PolymeshVertices.Add(new AFacet(ver.V1, ver.V2, ver.V3));
                            mesh.PolymeshNormals.Add(new APoint(normal[0].X, normal[0].Y,
normal[0].Z));
                        }
                        foreach (var p in node.GetPoints())
                        {
                            if (m_TransformationStack.Count > 0)
                            {
```

```
                                currentTransform = m_TransformationStack.Peek();
                                XYZ point = currentTransform.OfPoint(p);
                                mesh.PolymeshPoints.Add(new APoint(point.X, point.Y,
point.Z));
                            }
                            else
                            {
                                mesh.PolymeshPoints.Add(new APoint(p.X, p.Y, p.Z));
                            }

                        }
                        foreach (var mod in node.GetUVs())
                        {
                            mesh.PolymeshUvModels.Add(new AUV(mod.U, mod.V));
                        }
                        MergeMeshes.Add(mesh);
                    }
                public void OnRPC(RPCNode node)
                    {

                        Transform rpcTrf = node.GetTransform();
                        Transform rpcWorldTrf =
m_TransformationStack.Peek().Multiply(rpcTrf);
                        RPCs.Add(new APoint(rpcWorldTrf.Origin.X, rpcWorldTrf.Origin.Y,
rpcWorldTrf.Origin.Z), nameOfRPC);
                    }

                    public RenderNodeAction OnViewBegin(ViewNode node)
                    {
                        return RenderNodeAction.Proceed;
                    }

                    public void OnViewEnd(ElementId elementId)
                    {
                        // Left empty for the future.
                    }

                    public bool Start()
                    {
                        return true;
                    }

                    public RenderNodeAction OnCurve(CurveNode node)
                    {
                        return RenderNodeAction.Skip;
                    }

                    public RenderNodeAction OnPolyline(PolylineNode node)
                    {
                        return RenderNodeAction.Skip;
                    }

                    public RenderNodeAction OnPoint(PointNode node)
                    {
                        return RenderNodeAction.Skip;
                    }

                    public void OnLineSegment(LineSegment segment)
                    {
                        // No need to export.
                    }

                    public void OnPolylineSegments(PolylineSegments segments)
                    {
```

```
                // No need to export.
            }

            public void OnText(TextNode node)
            {
                // No need to export.
            }
        }
    }
```

Додаток Г. Лістинги Product.wxs та CustomActions

Product.wxs.

```
<?xml version="1.0" encoding="UTF-8"?>

<?define ProductVersion = "1.0.0.80"?>
<?define ProductName = "Walk Through 3D"?>
<?define ProductUpgradeCode = "{28874DD0-E8C4-4A0E-93ED-
E3083C1C9174}"?>
<?define ManufacturerName = "AMC Bridge"?>

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">

      <?define Addin_TargetDir=$(var.WalkThrough3D.TargetDir)?>
      <?define
VersionOfRevit_TargetDir=$(var.VersionOfRevit.TargetDir)?>

      <Product Id="*" Name="$(var.ProductName)" Language="1033"
Version="$(var.ProductVersion)" Manufacturer="$(var.ManufacturerName)"
UpgradeCode="$(var.ProductUpgradeCode)">
            <Package InstallerVersion="200" Compressed="yes"
InstallScope="perMachine" />

            <MajorUpgrade DowngradeErrorMessage="A newer version of
[ProductName] is already installed." />
            <Media Id="1" Cabinet="WalkThrough3D.cab" EmbedCab="yes"
/>


            <Feature Id="ProductFeature" Title="Setup" Level="1">
                  <ComponentGroupRef Id="ProductComponents" />
                  <ComponentGroupRef Id="Game_files" />
                  <ComponentGroupRef Id="EnUs_files" />
                  <ComponentGroupRef Id="NVIDIA_files" />
                  <ComponentGroupRef Id="Oculus_files" />
                  <ComponentGroupRef Id="Ogg_files" />
                  <ComponentGroupRef Id="OpenVR1_files" />
                  <ComponentGroupRef Id="OpenVR2_files" />
                  <ComponentGroupRef Id="PhysX3_files" />
                  <ComponentGroupRef Id="Vorbis_files" />
                  <ComponentGroupRef Id="Windows_files" />
                  <ComponentGroupRef Id="Content_files" />
                  <ComponentGroupRef Id="Win64_files" />
            </Feature>
            <Property Id="WIXUI_INSTALLDIR">INSTALLFOLDER</Property>
            <UIRef Id="WixUI_InstallDir" />
            <UIRef Id="WixUI_ErrorProgressText" />
            <WixVariable Id="WixUILicenseRtf" Value="License.rtf" />
            <WixVariable Id="WixUIDialogBmp"
Value="Images\DialogBanner.png" />
            <WixVariable Id="WixUIBannerBmp"
Value="Images\Dialog.jpg" />
            <Icon Id="walk_through_3d_256x256.ico"
SourceFile="Images\walk_through_3d_256x256.ico"/>
            <Property Id="ARPPRODUCTICON"
Value="walk_through_3d_256x256.ico" />
```

```
                <Binary Id="CustomActions"
SourceFile="$(var.VersionOfRevit.TargetDir)$(var.VersionOfRevit.TargetNam
e).CA.dll" />
                <InstallExecuteSequence>
                     <Custom Action="CheckRevit" Sequence="1"></Custom>
                     <Custom Action="CreateAddinManifest"
Before="InstallFinalize">NOT REMOVE</Custom>
                     <Custom Action="RemoveAddinManifest"
After="InstallFinalize">(Installed) AND (NOT UPGRADINGPRODUCTCODE) AND
(NOT REINSTALL)</Custom>
                </InstallExecuteSequence>
           </Product>
           <Fragment>
                <CustomAction Id="CheckRevit" BinaryKey="CustomActions"
DllEntry="CheckRevit" Return="check" />

                <CustomAction Id="CreateAddinManifest"
BinaryKey="CustomActions" DllEntry="CreateAddinManifest" Return="ignore"
/>

                <CustomAction Id="RemoveAddinManifest"
BinaryKey="CustomActions" DllEntry="RemoveAddinManifest" Return="ignore"
/>
           </Fragment>


           <Fragment>
                <Directory Id="TARGETDIR" Name="SourceDir">
                     <Directory Id="ProgramFilesFolder">
                          <Directory Id="INSTALLFOLDER"
Name="$(var.ProductName)">
                               <Directory Id="Game" Name="Game">
                                    <Directory Id="WalkThrough"
Name="WalkThrough">
                                         <Directory
Id="WalkThroughBinaries" Name="Binaries">
                                              <Directory
Id="WalkThroughWin64" Name="Win64" />
                                         </Directory>
                                         <Directory Id="Content"
Name="Content">
                                              <Directory Id="Paks"
Name="Paks" />
                                         </Directory>
                                    </Directory>
                                    <Directory Id="Engine"
Name="Engine">
                                         <Directory Id="Extras"
Name="Extras">
                                              <Directory Id="Redist"
Name="Redist">
                                                   <Directory
Id="en_us" Name="en-us" />
                                              </Directory>
                                         </Directory>
                                         <Directory Id="EngineBinaries"
Name="Binaries">
```

```
                                                    <Directory
Id="ThirdParty" Name="ThirdParty">
                                                         <Directory
Id="NVIDIA" Name="NVIDIA">
                                                              <Directory
Id="NVaftermath" Name="NVaftermath">

     <Directory Id="NVidiaWin64" Name="Win64" />
                                                              </Directory>
                                                         </Directory>
                                                         <Directory
Id="Oculus" Name="Oculus">
                                                              <Directory
Id="OVRPlugin" Name="OVRPlugin">

     <Directory Id="OVRPlugin1" Name="OVRPlugin">

     <Directory Id="OculusWin64" Name="Win64" />

     </Directory>
                                                              </Directory>
                                                         </Directory>
                                                         <Directory Id="Ogg"
Name="Ogg">
                                                              <Directory
Id="OggWin64" Name="Win64">

     <Directory Id="OggVS2015" Name="VS2015" />
                                                              </Directory>
                                                         </Directory>
                                                         <Directory
Id="OpenVR" Name="OpenVR">
                                                              <Directory
Id="OpenVRv1_0_11" Name="OpenVRv1_0_11">

     <Directory Id="OpenVR1Win64" Name="Win64" />
                                                              </Directory>
                                                              <Directory
Id="OpenVRv1_0_16" Name="OpenVRv1_0_16">

     <Directory Id="OpenVR2Win64" Name="Win64" />
                                                              </Directory>
                                                         </Directory>
                                                         <Directory
Id="PhysX3" Name="PhysX3">
                                                              <Directory
Id="PhysX3Win64" Name="Win64">

     <Directory Id="PhysX3VS2015" Name="VS2015" />
                                                              </Directory>
                                                         </Directory>
                                                         <Directory
Id="Vorbis" Name="Vorbis">
                                                              <Directory
Id="VorbisWin64" Name="Win64">

     <Directory Id="VorbisVS2015" Name="VS2015" />
                                                              </Directory>
```

```
                                                    </Directory>
                                                    <Directory
Id="Windows" Name="Windows">
                                                          <Directory
Id="DirectX" Name="DirectX">

      <Directory Id="WindowsX64" Name="x64" />
                                                          </Directory>
                                                    </Directory>
                                              </Directory>

                                        </Directory>
                                  </Directory>
                            </Directory>
                      </Directory>
                </Directory>

                <ComponentGroup Id="ProductComponents"
Directory="INSTALLFOLDER">

                      <Component Id="NLog.config" Guid="7bc7116c-d909-
429d-a431-6487a21b14a6">
                            <File Id="NLog.config" Name="NLog.config"
Source="$(var.Addin_TargetDir)NLog.config" />
                      </Component>

                      <Component Id="DataStructure.dll" Guid="aadcbee2-
1a08-428f-9450-22d4bbdf19a2">
                            <File Id="DataStructure.dll"
Name="DataStructure.dll"
Source="$(var.WalkThrough3D.TargetDir)DataStructure.dll" />
                      </Component>
                      <Component Id="Exporter.dll" Guid="069b7b75-68ae-
4dad-9e28-67463287c35d">
                            <File Id="Exporter.dll" Name="Exporter.dll"
Source="$(var.WalkThrough3D.TargetDir)Exporter.dll" />
                      </Component>
                      <Component Id="NLog.dll" Guid="102cb1c7-b29a-4bb2-
b7cb-37e7086ddc4a">
                            <File Id="NLog.dll" Name="NLog.dll"
Source="$(var.Addin_TargetDir)NLog.dll" />
                      </Component>
                      <Component Id="WalkThrough3D.dll" Guid="02014fae-
9a59-4cf6-9f4d-d7def70ff8fb">
                            <File Id="WalkThrough3D.dll"
Name="WalkThrough3D.dll" Source="$(var.Addin_TargetDir)WalkThrough3D.dll"
/>
                      </Component>
                      <Component Id="VersionOfRevit.dll" Guid="795db39e-
9c4a-4c43-b715-203a2cdee146">
                            <File Id="VersionOfRevit.dll"
Name="VersionOfRevit.dll"
Source="$(var.VersionOfRevit_TargetDir)VersionOfRevit.dll" />
                      </Component>

                </ComponentGroup>
                <ComponentGroup Id="Game_files" Directory="Game">
```

```
                    <Component Id="Game_1" Guid="6b6fb941-c87d-41ac-
80de-3a9c6f6cb3c0">
                         <File Id="Game_WalkThrough.exe"
Name="WalkThrough.exe" Source="Game\WalkThrough.exe" />
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="EnUs_files" Directory="en_us">
                    <Component Id="en_us_1" Guid="b3d4a002-530d-413d-
9fa9-09f517ff8c55">
                         <File Id="UE4PrereqSetup_x64"
Name="UE4PrereqSetup_x64.exe" Source="Game\Engine\Extras\Redist\en-
us\UE4PrereqSetup_x64.exe" />
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="NVIDIA_files"
Directory="NVidiaWin64">
                    <Component Id="Nvidia_1" Guid="ea387d04-a996-4975-
821f-987ac9fab4e4">
                         <File Id="NVIDIA"
Name="GFSDK_Aftermath_Lib.x64.dll"
Source="Game\Engine\Binaries\ThirdParty\NVIDIA\NVaftermath\Win64\GFSDK_Af
termath_Lib.x64.dll" />
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="Oculus_files"
Directory="OculusWin64">
                    <Component Id="Oculus_1" Guid="28c90af1-5874-4016-
8229-d0443fb1350a">
                         <File Id="Oculus" Name="OVRPlugin.dll"
Source="Game\Engine\Binaries\ThirdParty\Oculus\OVRPlugin\OVRPlugin\Win64\
OVRPlugin.dll" />
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="Ogg_files" Directory="OggVS2015">
                    <Component Id="Ogg_1" Guid="ea404fe0-3c12-4d89-
bbe4-dfc0f5d59639">
                         <File Id="Ogg" Name="libogg_64.dll"
Source="Game\Engine\Binaries\ThirdParty\Ogg\Win64\VS2015\libogg_64.dll"
/>
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="OpenVR1_files"
Directory="OpenVR1Win64">
                    <Component Id="OpenVR_1" Guid="192a3a05-d768-4ec3-
b28d-c294bfc99f1d">
                         <File Id="OpenVR1" Name="openvr_api.dll"
Source="Game\Engine\Binaries\ThirdParty\OpenVR\OpenVRv1_0_11\Win64\openvr
_api.dll" />
                    </Component>
               </ComponentGroup>

               <ComponentGroup Id="OpenVR2_files"
Directory="OpenVR2Win64">
```

```
                        <Component Id="OpenVR_2" Guid="172a3a05-d768-4ec3-
b28d-c294bfc99f1d">
                            <File Id="OpenVR2" Name="openvr_api.dll"
Source="Game\Engine\Binaries\ThirdParty\OpenVR\OpenVRv1_0_16\Win64\openvr
_api.dll" />
                        </Component>
                    </ComponentGroup>

                    <ComponentGroup Id="PhysX3_files"
Directory="PhysX3VS2015">
                        <Component Id="PhysX3_1" Guid="2f383be9-653f-447e-
bb01-28d4f9c97bb9">
                            <File Id="APEX_ClothingPROFILE_x64.dll"
Name="APEX_ClothingPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\APEX_Clothing
PROFILE_x64.dll" />
                        </Component>
                        <Component Id="PhysX3_2" Guid="0dce5221-06d0-4253-
a576-51f91f56750f">
                            <File Id="APEX_ClothingPROFILE_x64.pdb"
Name="APEX_ClothingPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\APEX_Clothing
PROFILE_x64.pdb" />
                        </Component>
                        <Component Id="PhysX3_3" Guid="ff3cc6b8-ec87-4924-
acba-4d5c9fc0f1e5">
                            <File Id="APEX_LegacyPROFILE_x64.dll"
Name="APEX_LegacyPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\APEX_LegacyPR
OFILE_x64.dll" />
                        </Component>
                        <Component Id="PhysX3_4" Guid="d4e6c5f4-edf2-4106-
ae65-ac0257378212">
                            <File Id="APEX_LegacyPROFILE_x64.pdb"
Name="APEX_LegacyPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\APEX_LegacyPR
OFILE_x64.pdb" />
                        </Component>
                        <Component Id="PhysX3_5" Guid="e3d97fa6-7312-411f-
b8a9-1429d120e9ed">
                            <File Id="ApexFrameworkPROFILE_x64.dll"
Name="ApexFrameworkPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\ApexFramework
PROFILE_x64.dll" />
                        </Component>
                        <Component Id="PhysX3_6" Guid="4c2b13b2-f758-475d-
bfef-bb9dfd54de80">
                            <File Id="ApexFrameworkPROFILE_x64.pdb"
Name="ApexFrameworkPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\ApexFramework
PROFILE_x64.pdb" />
                        </Component>
                        <Component Id="PhysX3_7" Guid="5fc47b37-5f01-4866-
a15b-9c41c108ad32">
                            <File Id="NvClothPROFILE_x64.dll"
Name="NvClothPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\NvClothPROFIL
E_x64.dll" />
```

```
                    </Component>
                    <Component Id="PhysX3_8" Guid="5e66cf0c-501b-46fd-
9645-edf26dc8b9f0">
                        <File Id="NvClothPROFILE_x64.pdb"
Name="NvClothPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\NvClothPROFIL
E_x64.pdb" />
                    </Component>
                    <Component Id="PhysX3_9" Guid="1edbd57e-e3e3-46f3-
b292-0df9d91f2852">
                        <File Id="PhysX3CommonPROFILE_x64.dll"
Name="PhysX3CommonPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3CommonP
ROFILE_x64.dll" />
                    </Component>
                    <Component Id="PhysX3_10" Guid="bd68df73-23ff-4065-
9269-81577688550f">
                        <File Id="PhysX3CommonPROFILE_x64.pdb"
Name="PhysX3CommonPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3CommonP
ROFILE_x64.pdb" />
                    </Component>
                    <Component Id="PhysX3_11" Guid="b0293a7d-9c93-413d-
80a5-42b68d5d8d15">
                        <File Id="PhysX3CookingPROFILE_x64.dll"
Name="PhysX3CookingPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3Cooking
PROFILE_x64.dll" />
                    </Component>
                    <Component Id="PhysX3_12" Guid="7ffa3032-307d-4516-
9ed6-4c62e9465c4a">
                        <File Id="PhysX3CookingPROFILE_x64.pdb"
Name="PhysX3CookingPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3Cooking
PROFILE_x64.pdb" />
                    </Component>
                    <Component Id="PhysX3_13" Guid="e3c5c965-5587-4d45-
9f14-caa2b02879d4">
                        <File Id="PhysX3PROFILE_x64.dll"
Name="PhysX3PROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3PROFILE
_x64.dll" />
                    </Component>
                    <Component Id="PhysX3_14" Guid="51f84a40-ffb0-4a7b-
b6c3-d2b9af8846db">
                        <File Id="PhysX3PROFILE_x64.pdb"
Name="PhysX3PROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PhysX3PROFILE
_x64.pdb" />
                    </Component>
                    <Component Id="PhysX3_15" Guid="aed34815-8fd5-4efa-
8b9c-1c8e3f0f1967">
                        <File Id="PxFoundationPROFILE_x64.dll"
Name="PxFoundationPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PxFoundationP
ROFILE_x64.dll" />
                    </Component>
                    <Component Id="PhysX3_16" Guid="5d100975-edca-4283-
```

```
a6c7-2241cfc3e40e">
                              <File Id="PxFoundationPROFILE_x64.pdb"
Name="PxFoundationPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PxFoundationP
ROFILE_x64.pdb" />
                        </Component>
                        <Component Id="PhysX3_17" Guid="e96e1068-a324-4d32-
9271-7f2ede963d15">
                              <File Id="PxPvdSDKPROFILE_x64.dll"
Name="PxPvdSDKPROFILE_x64.dll"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PxPvdSDKPROFI
LE_x64.dll" />
                        </Component>
                        <Component Id="PhysX3_18" Guid="b8e8814e-d2ae-473a-
91bf-3af1abdc0628">
                              <File Id="PxPvdSDKPROFILE_x64.pdb"
Name="PxPvdSDKPROFILE_x64.pdb"
Source="Game\Engine\Binaries\ThirdParty\PhysX3\Win64\VS2015\PxPvdSDKPROFI
LE_x64.pdb" />
                        </Component>
                  </ComponentGroup>

                  <ComponentGroup Id="Vorbis_files"
Directory="VorbisVS2015">
                        <Component Id="Vorbis_1" Guid="f1a5b04f-96df-4a58-
8c81-f7e52acf2023">
                              <File Id="libvorbis_64"
Name="libvorbis_64.dll"
Source="Game\Engine\Binaries\ThirdParty\Vorbis\Win64\VS2015\libvorbis_64.
dll" />
                        </Component>
                        <Component Id="Vorbis_2" Guid="121758a7-c594-4b8d-
aa4e-dc5efff43165">
                              <File Id="libvorbisfile_64"
Name="libvorbisfile_64.dll"
Source="Game\Engine\Binaries\ThirdParty\Vorbis\Win64\VS2015\libvorbisfile
_64.dll" />
                        </Component>
                  </ComponentGroup>

                  <ComponentGroup Id="Windows_files"
Directory="WindowsX64">
                        <Component Id="Windows_1" Guid="771e18e6-cc43-4e6c-
b97d-6755d15584be">
                              <File Id="WinPixEventRuntime"
Name="WinPixEventRuntime.dll"
Source="Game\Engine\Binaries\ThirdParty\Windows\DirectX\x64\WinPixEventRu
ntime.dll" />
                        </Component>
                  </ComponentGroup>

                  <ComponentGroup Id="Content_files" Directory="Paks">
                        <Component Id="Paks_1" Guid="d6aaca9f-57b4-40df-
b2a0-082f615f1d9d">
                              <File Id="WalkThroughWindowsNoEditor"
Name="WalkThrough-WindowsNoEditor.pak"
Source="Game\WalkThrough\Content\Paks\WalkThrough-WindowsNoEditor.pak" />
                        </Component>
```

```
                </ComponentGroup>

                <ComponentGroup Id="Win64_files"
Directory="WalkThroughWin64">
                    <Component Id="WalkThroughWin64_1" Guid="a8eca6ee-
1863-4d46-a801-28751b7d87ee">
                        <File Id="Parser.dll" Name="Parser.dll"
Source="Game\WalkThrough\Binaries\Win64\Parser.dll" />
                    </Component>
                    <Component Id="WalkThroughWin64_2" Guid="80a4b70f-
1fe8-4553-95bb-1334ff59465e">
                        <File Id="Parser.lib" Name="Parser.lib"
Source="Game\WalkThrough\Binaries\Win64\Parser.lib" />
                    </Component>
                    <Component Id="WalkThroughWin64_3" Guid="0d9b01e3-
0153-49c9-8ce1-0eab1fc9d780">
                        <File Id="WalkThrough.exe"
Name="WalkThrough.exe"
Source="Game\WalkThrough\Binaries\Win64\WalkThrough.exe" />
                    </Component>
                </ComponentGroup>
            </Fragment>
        </Wix>
```

CustomActions.

```
using System;
using System.Collections.Generic;
using System.IO;
using Autodesk.RevitAddIns;
using Microsoft.Deployment.WindowsInstaller;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace VersionOfRevit
{

    public static class CustomActions
    {

        private static string ADDIN_MANIFEST_FILENAME = "WalkThrough3D.addin";

        private static string ADDIN_ASSEMBLY_NAME = "WalkThrough3D.dll";

        private static string INSTALLFOLDER = "INSTALLFOLDER";

        private const string MANIFEST_NAME_CAPTION = "WalkThrough3D";
        private const string MANIFEST_CLIENT_ID = "9EEB655C-1721-421A-B066-
B2A0f038CD8C";
        private const string MANIFEST_VENDOR = "AMC Bridge";

        private const string MANIFEST_ENTRY_POINT =
"WalkThrough3D.WalkThroughAddin";



        [CustomAction]
        public static ActionResult CreateAddinManifest(Session session)
        {
            session.Log("CustomAction: CustomAction: Begin CustomAction1");

            string installFolder = session.GetTargetPath(INSTALLFOLDER);
```

```
            if (string.IsNullOrEmpty(installFolder))
            {
                return ActionResult.Failure;
            }

            string manifestFolderPath = null;

            List<RevitProduct> revitProduct = getRevitProduct();
            foreach (var item in revitProduct)
            {
                if (item != null)
                {
                    manifestFolderPath = (item.Version == RevitVersion.Unknown)
? item.CurrentUserAddInFolder.Replace("Unknown", Regex.Replace(item.Name, @"[^\d]+", "")) :
item.CurrentUserAddInFolder;
                }
                if (string.IsNullOrEmpty(manifestFolderPath))
                {
                    return ActionResult.Failure;
                }

                string addinAssemblyPath = Path.Combine(installFolder,
ADDIN_ASSEMBLY_NAME);

                session.Log("CustomAction: Addin Revit version: " +
item?.Version.ToString() + ". Addin installFolder = " + addinAssemblyPath);
                session.Log("CustomAction: revitAddinManifestsPath = " +
manifestFolderPath);
                RevitAddInApplication kenestoApplication = new
RevitAddInApplication(
                MANIFEST_NAME_CAPTION, addinAssemblyPath,
Guid.Parse(MANIFEST_CLIENT_ID), MANIFEST_ENTRY_POINT, MANIFEST_VENDOR);

                RevitAddInManifest manifest = new RevitAddInManifest();
                manifest.AddInApplications.Add(kenestoApplication);

                FileInfo fileInfo = new
FileInfo(Path.Combine(manifestFolderPath, ADDIN_MANIFEST_FILENAME));

                if (!Directory.Exists(manifestFolderPath))
                {
                    Directory.CreateDirectory(manifestFolderPath);
                }
                manifest.SaveAs(fileInfo.FullName);

                if (!File.Exists(fileInfo.FullName))
                {
                    return ActionResult.Failure;
                }

            }

            return ActionResult.Success;
        }

        [CustomAction]
        public static ActionResult RemoveAddinManifest(Session session)
        {
            string manifestFolderPath = string.Empty;

            List<RevitProduct> revitProduct = getRevitProduct();
            foreach (var item in revitProduct)
            {
                if (item != null)
```

```
                            {
                                manifestFolderPath = (item.Version == RevitVersion.Unknown)
? item.CurrentUserAddInFolder.Replace("Unknown", Regex.Replace(item.Name, @"[^\d]+", "")) :
item.CurrentUserAddInFolder;
                            }

                            string revitAddinManifestsPath =
Path.Combine(manifestFolderPath, ADDIN_MANIFEST_FILENAME);
                            if (File.Exists(revitAddinManifestsPath))
                            {
                                try
                                {
                                    File.Delete(revitAddinManifestsPath);
                                }
                                catch (Exception)
                                {
                                    return ActionResult.Failure;
                                }

                            }
                            else
                            {
                                return ActionResult.Failure;
                            }
                        }


                        return ActionResult.Success;
                    }

                    [CustomAction]
                    public static ActionResult CheckRevit(Session session)
                    {
                        List<RevitProduct> products = getRevitProduct();
                        if (products.Count==0)
                        {
                            MessageBox.Show("Revit not installed.");
                            return ActionResult.Failure;
                        }
                        return ActionResult.Success;
                    }

                    private static List<RevitProduct> getRevitProduct()
                    {
                        List<RevitProduct> revitProducts =
RevitProductUtility.GetAllInstalledRevitProducts();
                        return revitProducts;
                    }
                }
            }
```

Додаток Д. Лістинги класів Settings та About

Клас Settings.

```
using System;
using System.Windows.Forms;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using System.IO;
using System.Linq;
using NLog;
using NLog.Fluent;
using System.Diagnostics;
using DataStructure;
using System.Collections.Generic;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Threading;

namespace WalkThrough3D
{
    public partial class Settings : System.Windows.Forms.Form
    {
        Document doc;
        UIApplication app;
        DirectoryInfo di;
        Stopwatch stopWatch;
        double sum, per;
        int left, width, height, upload;
        BinaryWriter file;
        string directoryPath, nameOfForm;
        List<AMaterial> Materials;
        List<APolymesh> Meshes;
        Dictionary<APoint,string> RPCs;
        Dictionary<double, string> Levels;
        List<APoint> Lights;
        List<string> NameOfMeshes;
        public delegate void ChangeLabelsDelegate();
        bool floor;

        [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        private static extern IntPtr FindWindowEx(IntPtr hwndParent, IntPtr
hwndChildAfter, string lpszClass, string lpszWindow);

        [DllImport("user32.dll", EntryPoint = "SetWindowText", CharSet =
CharSet.Ansi)]
        private static extern bool SetWindowText(IntPtr hWnd, String
strNewWindowName);

        [DllImport("user32.dll", EntryPoint = "FindWindow", CharSet =
CharSet.Ansi)]
        private static extern IntPtr FindWindow(string className, string
windowName);
        public Settings(Document document, UIApplication application)
        {
            doc = document;
            app = application;
            nameOfForm = this.Text;
            InitializeComponent();
        }
    private void bBrowserDialog_Click(object sender, EventArgs e)
        {
            this.BeginInvoke(new ChangeLabelsDelegate(ChangeLabels));
```

```
                        FolderBrowserDialog fbdPathSave = new FolderBrowserDialog();
                        fbdPathSave.RootFolder =
System.Environment.SpecialFolder.MyComputer;
                        if (fbdPathSave.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                        {
                            tbPath.Text = fbdPathSave.SelectedPath;

                        }
                    }
                    private static void ChangeLabels()
                    {
                        bool titleChanged = false;

                        TimerCallback timerCallback = delegate
                        {
                            try
                            {
                                IntPtr ptr = FindWindow(null, "Browse For Folder");
                                if (ptr != IntPtr.Zero)
                                {
                                    SetWindowText(ptr, "Save To");
                                    titleChanged = true;
                                }
                                IntPtr ButtonHandle = FindWindowEx(ptr, IntPtr.Zero,
"Button", null);
                                if (ButtonHandle != IntPtr.Zero)
                                {
                                    SetWindowText(ButtonHandle, "Create New Folder");
                                    titleChanged = true;
                                }

                            }
                            catch (Exception exception)
                            {
                                System.Diagnostics.Debug.WriteLine(exception.Message);
                            }
                        };

                        System.Threading.Timer timer = null;
                        try
                        {
                            timer = new System.Threading.Timer(timerCallback, null, 500,
500);
                            while (!titleChanged)
                            {
                                System.Windows.Forms.Application.DoEvents();
                                System.Threading.Thread.Sleep(0);
                            }
                        }
                        catch (Exception exception)
                        {
                            System.Diagnostics.Debug.WriteLine(exception.Message);
                        }
                        finally
                        {
                            if (timer != null) timer.Dispose();
                        }
                    }

                private void bCancel_Click(object sender, EventArgs e)
                    {
                        this.Close();
                    }
            private void bAbout_Click(object sender, EventArgs e)
```

```
                    {
                        About fAbout = new About();
                        fAbout.ShowDialog();
                    }

                    private void bRun_Click(object sender, EventArgs e)
                    {

                        View3D view = doc.ActiveView as View3D;

                        if (tbPath.Text == String.Empty)
                        {
                            MessageBox.Show(Resources.Resources.ErrorNoPath);
                            return;
                        }
                        string assemblyPath = Assembly.GetExecutingAssembly().Location;
                        string meshesPath = tbPath.Text + "\\" + doc.Title +
"\\WalkThrough\\Binaries\\Win64";
                        DirectoryCopy(Path.Combine(Path.GetDirectoryName(assemblyPath),
"Game"), tbPath.Text + "\\" + doc.Title, meshesPath + "\\Meshes", true);

                        try
                        {
                            di = Directory.CreateDirectory(meshesPath + "\\Meshes");
                        }
                        catch (Exception ex)
                        {
                            LogManager.GetCurrentClassLogger().Error(ex, "The process
failed.");
                        }

                        Exporter.Exporter context = new Exporter.Exporter(doc,
app.Application);

                        CustomExporter exporter = new CustomExporter(doc, context);
                        try
                        {
                            stopWatch = new Stopwatch();
                            stopWatch.Start();
                            this.Text = "Preparing to export...";
                            exporter.Export(view);
                            directoryPath = di.FullName;
                            Materials = context.Materials;
                            Meshes = context.Meshes;
                            Lights = context.Lights;
                            RPCs = context.RPCs;
                            Levels = context.Levels;
                            floor = context.Floor;
                            NameOfMeshes = context.NameOfMeshes;
                            progressBar.Maximum = context.Count;
                            height = progressBar.Location.Y + progressBar.Size.Height +
lsaveas.Location.Y;
                            this.ClientSize = new System.Drawing.Size(width, height);
                            backgroundWorker1.WorkerReportsProgress = true;
                            backgroundWorker1.RunWorkerAsync();
                            stopWatch.Stop();
                            CallLogger(context);
                        }
                        catch (Autodesk.Revit.Exceptions.ExternalApplicationException ex)
                        {
                            LogManager.GetCurrentClassLogger().Error(ex, "The process
failed.");
                        }
                    }
                    private void DirectoryCopy(string sourceDirName, string destDirName,
```

```
string meshPath, bool copySubDirs)
                    {
                        DirectoryInfo dir = new DirectoryInfo(sourceDirName);

                        if (!dir.Exists)
                        {
                            throw new DirectoryNotFoundException(
                                "Source directory does not exist or could not be found: "
                                + sourceDirName);
                        }

                        DirectoryInfo[] dirs = dir.GetDirectories();
                        if (!Directory.Exists(destDirName))
                        {
                            Directory.CreateDirectory(destDirName);
                            FileInfo[] files = dir.GetFiles();
                            foreach (FileInfo f in files)
                            {
                                string temppath = Path.Combine(destDirName, f.Name);
                                f.CopyTo(temppath, false);
                            }

                            if (copySubDirs)
                            {
                                foreach (DirectoryInfo subdir in dirs)
                                {
                                    string temppath = Path.Combine(destDirName,
subdir.Name);

                                    DirectoryCopy(subdir.FullName, temppath, meshPath,
copySubDirs);

                                }
                            }
                        }
                        else
                        {
                            DirectoryInfo dirInfo = new DirectoryInfo(meshPath);

                            foreach (FileInfo f in dirInfo.GetFiles())
                            {
                                f.Delete();
                            }
                        }
                    }
            private void CreateFiles(string path)
                    {
                        FileAttributes fileAttribute = FileAttributes.Normal;
                        FileStream fileWriteStream = new FileStream(directoryPath + path,
FileMode.Create);
                        fileAttribute = File.GetAttributes(directoryPath + path) |
fileAttribute;
                        File.SetAttributes(directoryPath + path, fileAttribute);
                        file = new BinaryWriter(fileWriteStream);
                    }


                    private void CallLogger(Exporter.Exporter context)
                    {
                        TimeSpan ts = stopWatch.Elapsed;
                        string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
                                                           ts.Hours, ts.Minutes,
ts.Seconds,
                                                           ts.Milliseconds / 10);
                        Logger logger = LogManager.GetLogger("Info");
                        logger.Info("Exported by meshes:" + context.Count);
                        logger.Info("Time of meshes creation:" +
```

```
Directory.GetCreationTime(di.FullName));
                        logger.Info("Time of collection of geometry:" + elapsedTime +
"\n");
                        NLog.LogManager.GetCurrentClassLogger().Trace();
                    }
                    private void Settings_Load(object sender, EventArgs e)
                    {
                        width = bBrowserDialog.Location.X + bBrowserDialog.Size.Width +
lsaveas.Location.X;
                        height = bAbout.Location.Y + bAbout.Size.Height +
lsaveas.Location.Y;
                        this.ClientSize = new System.Drawing.Size(width, height);
                    }

                    private void backgroundWorker1_DoWork(object sender,
System.ComponentModel.DoWorkEventArgs e)
                    {
                        left = progressBar.Maximum;
                        int progress = 0;
                        CreateFiles("\\Lights");
                        foreach (var light in Lights)
                        {
                            light.WriteData(file);
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);
                        }
                        CreateFiles("\\RPCs");
                        file.Write(RPCs.Count(c => c.Value=="Car"));
                        file.Write(RPCs.Count(t => t.Value == "Tree"));
                        foreach (var rpc in RPCs.OrderBy(rpc=>rpc.Value))
                        {
                            rpc.Key.WriteData(file);
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);
                        }
                        CreateFiles("\\Levels");
                        file.Write(floor);
                        file.Write(Levels.Count);
                        foreach (var lev in Levels)
                        {
                            file.Write(lev.Key);
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);
                        }
                        foreach (var lev in Levels)
                        {
                            file.Write(lev.Value + "\n");
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);
                        }
                        CreateFiles("\\Names");
                        foreach (var nam in NameOfMeshes)
                        {
                            file.Write(nam + "\n");
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);

                        }
                        CreateFiles("\\Materials");
                        foreach (var mat in Materials)
```

```
                        {
                            mat.WriteData(file);
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);

                        }
                        int j = 0;
                        while (j < Meshes.Count)
                        {
                            CreateFiles($"\\{Meshes[j].NameOfMesh}");
                            Meshes[j].WriteData(file);
                            int i = j + 1;
                            while (i < Meshes.Count && Meshes[j].NameOfMesh ==
Meshes[i].NameOfMesh)
                            {
                                Meshes[i].WriteData(file);
                                Meshes.RemoveAt(i);
                            }
                            Meshes.RemoveAt(j);
                            upload++;
                            left--;
                            backgroundWorker1.ReportProgress(progress++);
                        }
                        backgroundWorker1.ReportProgress(progress);
                    }
                private void backgroundWorker1_ProgressChanged(object sender,
System.ComponentModel.ProgressChangedEventArgs e)
                    {
                        progressBar.Value = e.ProgressPercentage;
                        sum += per;
                        percentages.Text = Math.Round(sum, 2) + "% ";
                        meshesCount.Text = "(" + upload.ToString() + " out of " +
progressBar.Maximum + " meshes)";
                        per = 100 / Convert.ToDouble(progressBar.Maximum);

                    }

                    private void backgroundWorker1_RunWorkerCompleted(object sender,
System.ComponentModel.RunWorkerCompletedEventArgs e)
                    {
                        if (MessageBox.Show("Export is completed.") == DialogResult.OK)
                        {
                            this.Text = nameOfForm;
                            height = bAbout.Location.Y + bAbout.Size.Height +
lsaveas.Location.Y;
                            this.ClientSize = new System.Drawing.Size(width, height);
                            per = 0;
                            upload = 0;
                            sum = 0;
                            this.Close();
                        }

                    }
                }
            }
```

Клас About.

```
using System;
using System.Windows.Forms;

namespace WalkThrough3D
{
      public partial class About : Form
```

```
        {
            Credits credits;
            const string webSite = "https://www.amcbridge.com";
            public About()
            {
                InitializeComponent();

            }

            private void button1_Click(object sender, EventArgs e)
            {
                this.Close();
            }

            private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
            {
                linkLabel1.LinkVisited = true;
                System.Diagnostics.Process.Start(webSite);
            }

            private void linkLabel2_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
            {
                System.Diagnostics.Process.Start("mailto:support@amcbridge.com");
            }

            private void button2_Click(object sender, EventArgs e)
            {
                credits = new Credits();
                credits.Show();
            }

            private void About_FormClosed(object sender, FormClosedEventArgs e)
            {
               credits?.Close();
            }
        }
    }
```

Додаток Е. Лістинги проекту Parser

Current project is the property of company AMC Bridge

Заголовок AFacet.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
using namespace std;
class DLL_EXPORT AFacet {
public:
	float getV1() const { return _v1; }
	float getV2() const { return _v2; }
	float getV3() const { return _v3; }
	AFacet(float v1, float v2, float v3);

private:
	float _v1, _v2, _v3;
};
```

Клас AFacet.cpp

```
#include "pch.h"
#include "AFacet.h"

using namespace std;
AFacet::AFacet(float v1, float v2, float v3)
{
	_v1 = v1;
	_v2 = v2;
	_v3 = v3;
}
```

Заголовок AMaterial.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
#include <string>
using namespace std;
class DLL_EXPORT AMaterial {
public:
	string getName()      const { return _name; }
	string getTexture()   const { return _texture; }
	string getBump()      const { return _bump; }

	int    getR()            const { return _r; }
	int    getG()            const { return _g; }
	int    getB()            const { return _b; }
	int    getTransparency() const { return _transparency;}

	AMaterial(string name, int r, int g, int b, int transpar, string textures,
string bumps);
private:
	int _r, _g, _b, _transparency;
	string _name, _texture,_bump;
};
```

Клас AMaterial.cpp

```
#include "pch.h"
#include "AMaterial.h"
```

```
using namespace std;
AMaterial::AMaterial(string name, int r, int g, int b, int transpar, string textures,
string bumps)
{
	_name = name;
	_r = r;
	_g = g;
	_b = b;
	_transparency = transpar;
	_texture = textures;
	_bump = bumps;

}
```

Заголовок APoint.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
using namespace std;
class DLL_EXPORT APoint {
public:
	double getX() const { return _x; }
	double getY() const { return _y; }
	double getZ() const { return _z; }
	APoint(double x, double y, double z);
private:
	double _x, _y, _z;
};
```

Клас APoint.cpp

```
#include "pch.h"
#include "APoint.h"

using namespace std;
APoint::APoint(double x, double y, double z)
{
	_x = x;
	_y = y;
	_z = z;
}
```

Заголовок APolymesh.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
#include "AFacet.h"
#include "APoint.h"
#include "AUV.h"
#include <fstream>
#include <vector>
#include <stdio.h>
using namespace std;
class DLL_EXPORT APolymesh {
public:
 APolymesh(vector<AFacet> facets, vector<APoint> points, vector<AUV> uvs,
```

```
vector<APoint> normals, string nameOfMaterial, bool collision);
              vector<AFacet>    getVertices()  const { return _polymeshVertices; }
              vector<APoint>    getPoints()    const { return _polymeshPoints; }
              vector<AUV>       getModels()    const { return _polymeshUV; }
              vector<APoint>    getNormals()   const { return _polymeshNormals; }
              string           getMaterial()  const { return _nameOfMaterial; }
              bool             getCollision() const { return _collision; }
             private:
              vector<AFacet>    _polymeshVertices;
              vector<APoint>    _polymeshPoints;
              vector<AUV>       _polymeshUV;
              vector<APoint>    _polymeshNormals;
              string           _nameOfMaterial;
              bool             _collision;
             };
```

Клас APolymesh.cpp

```
#include "pch.h"
#include "APolymesh.h"

using namespace std;
APolymesh::APolymesh(vector<AFacet> facets, vector<APoint> points, vector<AUV> uvs,
vector<APoint> normals, string nameOfMaterial,bool collision)
{
	_polymeshVertices = facets;
	_polymeshPoints = points;
	_polymeshUV = uvs;
	_polymeshNormals = normals;
	_nameOfMaterial = nameOfMaterial;
	_collision = collision;
}
```

Заголовок AUV.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
using namespace std;
class DLL_EXPORT AUV {
public:
	double getU() const { return _u; }
	double getV() const { return _v; }

	AUV(double u, double v);
private:
	double _u, _v;
};
```

Клас AUV.cpp

```
#include "pch.h"
#include "AUV.h"

using namespace std;
AUV::AUV(double u, double v)
{
	_u= u;
```

```
	_v= v;
}
```

Заголовок Parser.h

```
#pragma once
#define DLL_EXPORT __declspec(dllexport)
#include "AFacet.h"
#include "APoint.h"
#include "AUV.h"
#include "APolymesh.h"
#include "AMaterial.h"
#include <fstream>
#include <vector>
#include <string>
using namespace std;
class DLL_EXPORT Parser {
public:
	vector<APolymesh> ParseMeshes(ifstream &file);
	vector<AMaterial> ParseMaterials(ifstream &file);
	vector<string>    ParseNames(ifstream &file);
	vector<APoint>    ParseLights(ifstream &file);
	vector<int>       GetCountOfRCP(ifstream &file);
	vector<APoint>    ParseRPCs(int count,ifstream &file);
	vector<double>    GetLevelsElevation(ifstream &file);
	vector<string>    GetLevelsNames(ifstream &file);
	double            GetFloor(ifstream &file);

	int             R() const { return _r; }
	int             G() const { return _g; }
	int             B() const { return _b; }
	int             Transparency() const { return _transparency; }


private:
	vector<AFacet>    _polymeshVertices;
	vector<APoint>    _polymeshPoints;
	vector<AUV>       _polymeshUV;
	vector<APoint>    _polymeshNormals;
	vector<AMaterial> _materials;
	vector<string>    _names;
	vector<APolymesh> _meshes;
	vector<APoint>    _lights;
	vector<APoint>    _RPCs;
	vector<string>    _levelsNames;
	vector<int>       _numbers;
	vector<double>    _elevations;

	bool            _collisions,_floor;
	string          _nameOfMaterial;
	string          _nameOfMesh;
	string          _texturesPath;
	string          _bumpPath;
	int             _r;
	int             _g;
	int             _b;
	int             _transparency;

	int             _ReadInt(ifstream &file);

	double          _ReadDouble(ifstream &file);
	bool            _ReadBool(ifstream &file);
	string          _ReadString(ifstream &file);
```

```
        AFacet          _ReadFacet(ifstream &file);
        APoint          _ReadPoint(ifstream &file);
        AUV             _ReadModel(ifstream &file);
        APoint          _ReadNormal(ifstream &file);
        void            _ParseLevels(ifstream &file);

        void            _ClearData();

};
```

Клас Parser.cpp

```
#include "pch.h"
#include "Parser.h"

using namespace std;

vector<APolymesh> Parser::ParseMeshes(ifstream & file)
{
        _meshes.clear();
        while (file.peek() != EOF)
        {
                _ClearData();
                int facetsCount = _ReadInt(file);
                for (int i = 0; i < facetsCount;i++)
                {
                        _polymeshVertices.push_back(_ReadFacet(file));
                }
                int pointsCount = _ReadInt(file);
                for (int i = 0; i < pointsCount; i++)
                {
                        _polymeshPoints.push_back(_ReadPoint(file));
                }
                int uvModelsCount = _ReadInt(file);
                for (int i = 0; i < uvModelsCount; i++)
                {
                        _polymeshUV.push_back(_ReadModel(file));
                }
                int normalsCount = _ReadInt(file);
                for (int i = 0; i < normalsCount; i++)
                {
                        _polymeshNormals.push_back(_ReadNormal(file));
                }
                _collisions = _ReadBool(file);
                char space;
                file.read(reinterpret_cast<char*>(&space), sizeof(char));
                _nameOfMaterial = _ReadString(file);
                _meshes.push_back(APolymesh(_polymeshVertices, _polymeshPoints,
_polymeshUV, _polymeshNormals, _nameOfMaterial, _collisions));
        }
        return _meshes;
}
vector<AMaterial> Parser::ParseMaterials(ifstream & file)
{
        while (file.peek() != EOF)
        {
                char space;
                file.read(reinterpret_cast<char*>(&space), sizeof(char));
                _nameOfMaterial = _ReadString(file);
                file.read(reinterpret_cast<char*>(&space), sizeof(char));
                _texturesPath = _ReadString(file);
                if (_texturesPath[0] == '\x1')
                {
```

```
                        _texturesPath.erase(_texturesPath.find(_texturesPath[0]), 1);
                }
                file.read(reinterpret_cast<char*>(&space), sizeof(char));
                _bumpPath = _ReadString(file);
                if (_bumpPath[0] == '\x1')
                {
                        _bumpPath.erase(_bumpPath.find(_bumpPath[0]), 1);
                }
                _r = _ReadInt(file);
                _g = _ReadInt(file);
                _b = _ReadInt(file);
                _transparency = _ReadInt(file);
                _materials.push_back(AMaterial(_nameOfMaterial, _r, _g,
_b,_transparency, _texturesPath,_bumpPath));
        }
        return _materials;
}
vector<string> Parser::ParseNames(ifstream & file)
{
        _names.clear();
        while (file.peek() != EOF)
        {
                char space;
                file.read(reinterpret_cast<char*>(&space), sizeof(char));
                _nameOfMesh = _ReadString(file);
                _names.push_back(_nameOfMesh);
        }
        return _names;
}
vector<APoint> Parser::ParseLights(ifstream & file)
{
        while (file.peek() != EOF)
        {
                _lights.push_back(_ReadPoint(file));
        }
        return _lights;
}

vector<APoint> Parser::ParseRPCs(int count, ifstream & file)
{
        _RPCs.clear();
        for (int i = 0; i < count; i++)
        {
                _RPCs.push_back(_ReadPoint(file));
        }
        return _RPCs;
}
vector<double> Parser::GetLevelsElevation(ifstream & file)
{
        if (_floor == true && _elevations.size() == 0 && _levelsNames.size()==0)
        {
                _ParseLevels(file);
        }
        return _elevations;
}
vector<string> Parser::GetLevelsNames(ifstream & file)
{
        if (_floor == true && _elevations.size() == 0 && _levelsNames.size() == 0)
        {
                _ParseLevels(file);
        }
        return _levelsNames;
}
double Parser::GetFloor(ifstream & file)
{
```

```
	if (_floor == NULL)
	{
		_ParseLevels(file);
	}
	return _floor;
}
vector<int> Parser::GetCountOfRCP(ifstream & file)
{
	for (int i = 0; i < 2; i++)
	{
		_numbers.push_back(_ReadInt(file));
	}
	return _numbers;
}
int Parser::_ReadInt(ifstream & file)
{
	int count;
	file.read(reinterpret_cast<char*>(&count), sizeof(int));
	return count;
}

AFacet Parser::_ReadFacet(ifstream & file)
{
	float vertice;
	float vertices[3];
	int j = 0;
	for (int i = 0; i < 3; i++)
	{
		file.read(reinterpret_cast<char*>(&vertice), sizeof(float));
		vertices[j] = vertice;
		j++;
	}
	j = 0;

	AFacet facet(vertices[2], vertices[1], vertices[0]);
	return AFacet(facet);
}

APoint Parser::_ReadPoint(ifstream & file)
{
	double point;
	double points[3];
	int j = 0;
	for (int i = 0; i < 3; i++)
	{
		file.read(reinterpret_cast<char*>(&point), sizeof(double));
		points[j] = point;
		j++;
	}
	j = 0;
	APoint pointer(points[1], points[0], points[2]);
	return APoint(pointer);
}

AUV Parser::_ReadModel(ifstream & file)
{
	double model;
	double models[2];
	int j = 0;
	for (int i = 0; i < 2; i++)
	{
		file.read(reinterpret_cast<char*>(&model), sizeof(double));
		models[j] = model;
		j++;
	}
```

```
	j = 0;
	AUV uVModel(models[0], models[1]);
	return AUV(uVModel);
}
APoint Parser::_ReadNormal(ifstream & file)
{
	double normal;
	double normals[3];
	int j = 0;
	for (int i = 0; i < 3; i++)
	{
		file.read(reinterpret_cast<char*>(&normal), sizeof(double));
		normals[j] = normal;
		j++;
	}
	j = 0;
	APoint Normal(normals[0], normals[1], normals[2]);
	return APoint(Normal);
}

void Parser::_ParseLevels(ifstream & file)
{
	_levelsNames.clear();
	_elevations.clear();
	_floor = _ReadBool(file);
	int count = _ReadInt(file);
	for (int i = 0; i < count; i++)
	{
		_elevations.push_back(_ReadDouble(file));
	}
	_levelsNames = ParseNames(file);
}

bool Parser::_ReadBool(ifstream & file)
{
	bool collision;
	file.read(reinterpret_cast<char*>(&collision), sizeof(bool));
	return collision;
}

double Parser::_ReadDouble(ifstream & file)
{
	double count;
	file.read(reinterpret_cast<char*>(&count), sizeof(double));
	return count;
}

void Parser::_ClearData()
{
	_polymeshVertices.clear();
	_polymeshPoints.clear();
	_polymeshUV.clear();
	_polymeshNormals.clear();
}

string Parser::_ReadString(ifstream & file)
{
	string name;
	std::getline(file, name);
	return name;
}
```

Додаток Є. Лістинги класу MainActor.cpp

Current project is the property of company AMC Bridge

Заголовок MainActor.h

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "Parser.h"
#include "APolymesh.h"
#include "AFacet.h"
#include "AMaterial.h"
#include "APoint.h"
#include "AUV.h"
#include <fstream>
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <string>
#include <filesystem>
#include <stdlib.h>
#include "Engine/PointLight.h"
#include "Components/StaticMeshComponent.h"
#include "Components/PointLightComponent.h"
#include "IImageWrapper.h"
#include "PlatformFilemanager.h"
#include "FileHelper.h"
#include "Engine/SpotLight.h"
#include "ModuleManager.h"
#include "Engine/Texture2D.h"
#include "Engine/StaticMesh.h"
#include "IImageWrapperModule.h"
#include "Components/SceneComponent.h"
#include "ConstructorHelpers.h"
#include "Materials/MaterialInstanceDynamic.h"
#include "ProceduralMeshComponent.h"
#include "MainActor.generated.h"

UCLASS()
class WALKTHROUGH_API AMainActor : public AActor
{
GENERATED_BODY()

public:
// Sets default values for this actor's properties
AMainActor();

protected:
virtual void BeginPlay() override;


public:
// Called every frame
virtual void Tick(float DeltaTime) override;
UTexture2D * LoadTexture2D(const FString& FilePath);
UPROPERTY(VisibleAnywhere, Category = "MyProceduralMesh")
	UProceduralMeshComponent* mesh;
UPROPERTY()
	UTexture2D* mytex;
UPROPERTY(VisibleAnywhere)
	UPointLightComponent* MyLight;
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Levels")
```

```
        TArray<float> Elevations;
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Levels")
        TArray<FString> LevelsNames;
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Levels")
        bool Terrain;
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Levels")
        float MaxZ;
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Levels")
        float MinZ;

private:
void AddTriangle(int32 v1, int32 v2, int32 v3);
void findMinMax(APoint point);
void createBox();
void createModel();
void createLights();
void createRPCs();
void getLevels();
bool fexists(const string& filename);

const float dist = 2000;
void ClearMeshData();
void SearchForIndex();
double minX, minY, minZ;
double maxX, maxY, maxZ;
Parser *p;
map<string, int32>  indexes;
vector<APolymesh> *Meshes;
vector<AMaterial> *Materials;
vector<string>    *Names;
TArray<UMaterialInstanceDynamic*> _materials;
UPROPERTY()
        TArray<FVector> _vertices;
UPROPERTY()
        TArray<int32> _triangles;
UPROPERTY()
        TArray<int32> _trianglesRPC;
UPROPERTY()
        TArray<FVector2D> _uV0;
UPROPERTY()
        TArray<FVector> _normals;
UPROPERTY()
        UMaterialInterface* GetTextures;
UPROPERTY()
        UMaterialInterface* GetNoBumps;
UPROPERTY()
        UMaterialInterface* GetWater;
UPROPERTY()
        UMaterialInterface* GetColors;
UPROPERTY()
        UMaterialInterface* GetGlass;
UPROPERTY()
        UMaterialInterface* GetMirror;
UPROPERTY()
        UMaterialInterface* GetRelief;
UPROPERTY()
        UMaterialInterface* GetGrass;
UPROPERTY()
        UMaterialInstanceDynamic* DynamicMaterialToUse;
UPROPERTY()
        UStaticMesh* GetMesh;
UPROPERTY()
        UStaticMesh* GetTree;
UPROPERTY()
        UStaticMesh* GetCar;
```

```
};
```

Клас MainActor.cpp

```
#include "MainActor.h"


AMainActor::AMainActor()
{
// Set this actor to call Tick() every frame.  You can turn this off to improve
performance if you don't need it.
	PrimaryActorTick.bCanEverTick = true;
	mesh = CreateDefaultSubobject<UProceduralMeshComponent>(TEXT("Mesh"));
	SetRootComponent(mesh);
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
DynamicColors(TEXT("/Game/Materials/MyColor.MyColor"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
Water(TEXT("/Game/Materials/Water.Water"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
DynamicTextures(TEXT("/Game/Materials/WithBump.WithBump"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
DynamicNoBump(TEXT("/Game/Materials/NoBump.NoBump"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
Glass(TEXT("/Game/Materials/MyGlass.MyGlass"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
Mirror(TEXT("/Game/Materials/Mirror.Mirror"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
Relief(TEXT("/Game/Materials/WithRelief.WithRelief"));
	static ConstructorHelpers::FObjectFinder<UMaterialInterface>
Grass(TEXT("/Game/Materials/Grass.Grass"));
	if (DynamicTextures.Succeeded() && DynamicColors.Succeeded() && Glass.Succeeded() &&
DynamicNoBump.Succeeded())
	{
		GetTextures = DynamicTextures.Object;
		GetColors = DynamicColors.Object;
		GetGlass = Glass.Object;
		GetMirror = Mirror.Object;
		GetWater = Water.Object;
		GetNoBumps = DynamicNoBump.Object;
		GetRelief = Relief.Object;
		GetGrass = Grass.Object;
	}
	static ConstructorHelpers::FObjectFinder<UStaticMesh>
Cube(TEXT("/Game/Geometry/Meshes/1M_Cube.1M_Cube"));
	static ConstructorHelpers::FObjectFinder<UStaticMesh>
Tree(TEXT("/Game/Geometry/RPC/Trees/Tree1.Tree1"));
	static ConstructorHelpers::FObjectFinder<UStaticMesh>
Car(TEXT("/Game/Geometry/RPC/Cars/Car1/Car.Car"));
	if (Cube.Succeeded() && Tree.Succeeded() && Car.Succeeded())
	{
		GetMesh = Cube.Object;
		GetTree = Tree.Object;
		GetCar = Car.Object;
	}
	p = new Parser();


	}



	void AMainActor::BeginPlay()
	{
```

```
	Super::BeginPlay();
	SearchForIndex();
	createModel();
	getLevels();
	createBox();
	createRPCs();
	createLights();
	MaxZ = maxZ;
	MinZ = minZ;
	}
	void AMainActor::AddTriangle(int32 v1, int32 v2, int32 v3)
	{
	_triangles.Add(v3);
	_triangles.Add(v2);
	_triangles.Add(v1);
	_trianglesRPC.Add(v1);
	_trianglesRPC.Add(v2);
	_trianglesRPC.Add(v3);

	}
	void AMainActor::Tick(float DeltaTime)
	{
	Super::Tick(DeltaTime);

	}

	static TSharedPtr<IImageWrapper> GetImageWrapperByExtention(const FString
InImagePath)
	{
	IImageWrapperModule& ImageWrapperModule =
FModuleManager::LoadModuleChecked<IImageWrapperModule>(FName("ImageWrapper"));
	if (InImagePath.EndsWith(".png"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::PNG);
	}
	else if (InImagePath.EndsWith(".jpg") || InImagePath.EndsWith(".jpeg"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::JPEG);
	}
	else if (InImagePath.EndsWith(".bmp"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::BMP);
	}
	else if (InImagePath.EndsWith(".ico"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::ICO);
	}
	else if (InImagePath.EndsWith(".exr"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::EXR);
	}
	else if (InImagePath.EndsWith(".icns"))
	{
		return ImageWrapperModule.CreateImageWrapper(EImageFormat::ICNS);
	}
	return nullptr;
	}
	UTexture2D* AMainActor::LoadTexture2D(const FString& FilePath)
	{
	TArray<uint8> RawFileData;
	UTexture2D* MyTexture = NULL;
	if (FFileHelper::LoadFileToArray(RawFileData, *FilePath, 0))
	{
		IImageWrapperModule& ImageWrapperModule =
```

```
FModuleManager::LoadModuleChecked<IImageWrapperModule>(FName("ImageWrapper"));
		// Note: PNG format.  Other formats are supported
		TSharedPtr<IImageWrapper> ImageWrapper = GetImageWrapperByExtention(FilePath);
		if (ImageWrapper.IsValid() && ImageWrapper-
>SetCompressed(RawFileData.GetData(), RawFileData.Num()))
		{
			const TArray<uint8>* UncompressedBGRA = NULL;
			if (ImageWrapper->GetRaw(ERGBFormat::BGRA, 8, UncompressedBGRA))
			{
				// Create the UTexture for rendering
				MyTexture = UTexture2D::CreateTransient(ImageWrapper->GetWidth(),
ImageWrapper->GetHeight());
				// Fill in the source data from the file

				void* TextureData = MyTexture->PlatformData-
>Mips[0].BulkData.Lock(LOCK_READ_WRITE);
				FMemory::Memcpy(TextureData, UncompressedBGRA->GetData(),
UncompressedBGRA->Num());
				MyTexture->PlatformData->Mips[0].BulkData.Unlock();

				// Update the rendering resource from data.
				MyTexture->UpdateResource();
			}
		}
	}
	return MyTexture;
	}

	void AMainActor::SearchForIndex()
	{
	int32 i = 0;
	ifstream names("Meshes\\Names", ios::in | ios::binary);
	Names= new vector<string>(p->ParseNames(names));
	ifstream materials("Meshes\\Materials", ios::in | ios::binary);
	Materials = new vector<AMaterial>(p->ParseMaterials(materials));
	for (auto& mat:*Materials)
	{
		FColor Color;
		Color.R = Materials->at(i).getR() >> 16;
		Color.B = Materials->at(i).getB();
		Color.G = Materials->at(i).getG() >> 8;
		Color.A = Materials->at(i).getTransparency();
		if (Materials->at(i).getName().find("Glass") != std::string::npos)
		{

			DynamicMaterialToUse = UMaterialInstanceDynamic::Create(GetGlass,
this);
			DynamicMaterialToUse->SetVectorParameterValue("Glass", Color);

		}
		else if (Materials->at(i).getName() == "slate2" || Materials-
>at(i).getName()=="potable water" || Materials->at(i).getName()=="Default Form")
		{
			DynamicMaterialToUse = UMaterialInstanceDynamic::Create(GetWater,
this);
			DynamicMaterialToUse->SetVectorParameterValue("ColorOfWater", Color);
		}
		else if (Materials->at(i).getName() == "Mirror")
		{
			DynamicMaterialToUse = UMaterialInstanceDynamic::Create(GetMirror,
this);
			DynamicMaterialToUse->SetVectorParameterValue("Mirror", Color);
		}
		else if (fexists(Materials->at(i).getTexture()) && Materials-
>at(i).getTexture() != "null")
```

```
		{
			string path = Materials->at(i).getTexture();
			FString image(path.c_str());
			mytex = LoadTexture2D(image);
			if (Materials->at(i).getBump() != "null")
			{
				DynamicMaterialToUse =
UMaterialInstanceDynamic::Create(GetTextures, this);
				DynamicMaterialToUse->SetTextureParameterValue("Texture", mytex);
				path = Materials->at(i).getBump();
				FString bump(path.c_str());
				mytex = LoadTexture2D(bump);
				DynamicMaterialToUse->SetTextureParameterValue("Bump", mytex);
			}
			else
			{
				DynamicMaterialToUse =
UMaterialInstanceDynamic::Create(GetNoBumps, this);
				DynamicMaterialToUse->SetTextureParameterValue("Texture", mytex);
			}
			DynamicMaterialToUse->SetVectorParameterValue("ColorOfTexture", Color);

		}
		else if (Materials->at(i).getBump() != "null")
		{
			string path = Materials->at(i).getBump();
			FString relief(path.c_str());
			mytex = LoadTexture2D(relief);
			DynamicMaterialToUse = UMaterialInstanceDynamic::Create(GetRelief,
this);
			DynamicMaterialToUse->SetTextureParameterValue("Relief", mytex);
			DynamicMaterialToUse->SetVectorParameterValue("ColorOfRelief", Color);
		}
		else
		{
			DynamicMaterialToUse = UMaterialInstanceDynamic::Create(GetColors,
this);
			DynamicMaterialToUse->SetVectorParameterValue("Color", Color);
		}
		_materials.Add(DynamicMaterialToUse);
		indexes[mat.getName()] = i++;
	}
	}
	void AMainActor::createModel()
	{
	int32 i = 0, j =0;
	for (int32 nam = 0 ; nam < Names->size(); nam++)
	{
		j++;
		FString name = TEXT("Mesh");
		FString iString = FString::FromInt(j);
		name.Append(iString);
		mesh = NewObject<UProceduralMeshComponent>(this,
UProceduralMeshComponent::StaticClass(), FName(*name));
		mesh->RegisterComponent();

		ifstream meshes("Meshes\\" + Names->at(nam), ios::in | ios::binary);
		Meshes = new vector<APolymesh>(p->ParseMeshes(meshes));
		for (APolymesh& m:*Meshes)
		{
			ClearMeshData();
			for (APoint& point : m.getPoints())
			{
				_vertices.Add(FVector(point.getX(), point.getY(), point.getZ()));
```

```
                        findMinMax(point);
                }

                for (AFacet& fac : m.getVertices())
                {
                        AddTriangle(fac.getV1(), fac.getV2(), fac.getV3());
                }
                for (AUV& mod : m.getModels())
                {
                        if (m.getMaterial()== "slate2" || m.getMaterial() == "potable
water" || m.getMaterial() == "Default Form")
                        {
                                _uV0.Add(FVector2D(mod.getU()/20, mod.getV()/20));
                        }
                        else
                        {
                                _uV0.Add(FVector2D(mod.getU(), mod.getV()));
                        }
                }
                for (APoint& normal : m.getNormals())
                {
                        _normals.Add(FVector(normal.getX(), normal.getY(),
normal.getZ()*40));
                }
                if (m.getMaterial() == "RPC")
                {
                        _triangles += _trianglesRPC;
                }
                if (GetTextures && GetColors && GetGlass && GetMirror)
                {
                        int32 index = indexes[m.getMaterial()];
                        mesh->SetMaterial(i, _materials[index]);
                        mesh->CreateMeshSection(i++, _vertices, _triangles, _normals,
_uV0, TArray<FColor>(), TArray<FProcMeshTangent>(), m.getCollision());

                }

        }

}

}

void AMainActor::createLights()
{
int32 j = 0;
ifstream lights("Meshes\\Lights", ios::in | ios::binary);
vector<APoint> *Lights = new vector<APoint>(p->ParseLights(lights));
for (auto& l : *Lights)
{
        j++;
        FString name = TEXT("Light");
        FString iString = FString::FromInt(j);
        name.Append(iString);
        MyLight = NewObject<UPointLightComponent>(this,
UPointLightComponent::StaticClass(), FName(*name));
        MyLight->SetRelativeLocation(FVector(l.getX(), l.getY(), l.getZ()));
        MyLight->RegisterComponent();


}
}
void AMainActor::createRPCs()
{
```

```
	UStaticMesh *Get= GetCar;

	int32 j = 0;
	ifstream rpcs("Meshes\\RPCs", ios::in | ios::binary);
	vector<int32> *numbers = new vector<int>(p->GetCountOfRCP(rpcs));
	for (int32 i = 0; i < numbers->size(); i++)
	{
		vector<APoint> *RPCs = new vector<APoint>(p->ParseRPCs(numbers->at(i),rpcs));
		for (auto& rpc : *RPCs)
		{
				j++;
				FString name = TEXT("RPC");
				FString iString = FString::FromInt(j);
				name.Append(iString);
				UStaticMeshComponent* newRPC =
NewObject<UStaticMeshComponent>(this, UStaticMeshComponent::StaticClass(), FName(*name));
				newRPC->RegisterComponent();
				newRPC->SetStaticMesh(Get);
				newRPC->SetWorldScale3D(FVector(3.0f, 3.0f, 3.0f));
				newRPC->SetRelativeLocation(FVector(rpc.getX(), rpc.getY(),
rpc.getZ()));

		}

		Get = GetTree;
	}
	}

	void AMainActor::getLevels()
	{
	ifstream levels("Meshes\\Levels", ios::in | ios::binary);
	Terrain = p->GetFloor(levels);
	vector<double> *lvlElev = new vector<double>(p->GetLevelsElevation(levels));
	vector<string> *lvlNames = new vector<string>(p->GetLevelsNames(levels));
	for (int32 i=0; i<lvlElev->size(); i++)
	{
		Elevations.Add(lvlElev->at(i));
		LevelsNames.Add(FString(lvlNames->at(i).c_str()));

	}

	}

	bool AMainActor::fexists(const string & filename)
	{
	     ifstream ifile(filename.c_str());
	 return (bool)ifile;
	}



	void AMainActor::findMinMax(APoint point)
	{
		double x, y, z;
		x = point.getX();
		y = point.getY();
		z = point.getZ();
		minX = (minX > x) ? x: minX ;
		maxX = (maxX < x) ? x : maxX;
		minY = (minY > y) ? y : minY;
		maxY = (maxY < y) ? y : maxY;
		minZ = (minZ > z) ? z : minZ;
		maxZ = (maxZ < z) ? z : maxZ;
	}
```

```
	void AMainActor::createBox()
	{
	UStaticMeshComponent* floor = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("Floor"));
	floor->RegisterComponent();
	floor->SetStaticMesh(GetMesh);
	floor->SetMobility(EComponentMobility::Movable);

	UStaticMeshComponent* wall1 = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("W1"));
	wall1->RegisterComponent();
	wall1->SetStaticMesh(GetMesh);
	wall1->SetMobility(EComponentMobility::Movable);

	UStaticMeshComponent* wall2 = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("W2"));
	wall2->RegisterComponent();
	wall2->SetStaticMesh(GetMesh);
	wall2->SetMobility(EComponentMobility::Movable);

	UStaticMeshComponent* wall3 = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("W3"));
	wall3->RegisterComponent();
	wall3->SetStaticMesh(GetMesh);
	wall3->SetMobility(EComponentMobility::Movable);

	UStaticMeshComponent* wall4 = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("W4"));
	wall4->RegisterComponent();
	wall4->SetStaticMesh(GetMesh);
	wall4->SetMobility(EComponentMobility::Movable);

	UStaticMeshComponent* roof = NewObject<UStaticMeshComponent>(this,
UStaticMeshComponent::StaticClass(), FName("Roof"));
	roof->RegisterComponent();
	roof->SetStaticMesh(GetMesh);
	roof->SetMobility(EComponentMobility::Movable);

	float middle;
	float length;
	floor->SetRelativeLocation(FVector(maxX - ((maxX - minX) / 2), maxY - ((maxY - minY)
/ 2), minZ));
	floor->SetWorldScale3D(FVector((maxX - minX + dist * 2) / 100, (maxY - minY + dist *
2) / 100, 0.0f));
	floor->SetMaterial(0, GetGrass);

	middle = maxY - ((maxY - minY) / 2);
	length = (maxY - minY + dist * 2) / 100;
	if (Terrain)
	{
		floor->SetHiddenInGame(true);
		wall1->SetRelativeLocation(FVector(minX, middle, (maxZ + dist - minZ) / 2));
		wall2->SetRelativeLocation(FVector(maxX , middle, (maxZ + dist - minZ) / 2));
	}
	else
	{
		wall1->SetRelativeLocation(FVector(minX - dist, middle, (maxZ + dist - minZ) /
2));
		wall2->SetRelativeLocation(FVector(maxX + dist, middle, (maxZ + dist - minZ) /
2));
	}

	wall1->SetWorldScale3D(FVector(0.0f, length, (maxZ - minZ + dist * 2) / 100));
	wall1->SetHiddenInGame(true);
```

```
	wall2->SetWorldScale3D(FVector(0.0f, length, (maxZ - minZ + dist * 2) / 100));
	wall2->SetHiddenInGame(true);

	middle = maxX - ((maxX - minX) / 2);
	length = (maxX - minX + dist * 2) / 100;

	wall3->SetWorldScale3D(FVector(length, 0.0f, (maxZ - minZ + dist * 2) / 100));
	wall3->SetHiddenInGame(true);

	wall4->SetWorldScale3D(FVector(length, 0.0f, (maxZ - minZ + dist * 2) / 100));
	wall4->SetHiddenInGame(true);

	roof->SetWorldScale3D(FVector((maxX - minX + dist * 2) / 100, (maxY - minY + dist *
2) / 100, 0.0f));
	roof->SetHiddenInGame(true);
	if (Terrain)
	{
		wall3->SetRelativeLocation(FVector(middle, maxY, (maxZ + dist - minZ) / 2));
		wall4->SetRelativeLocation(FVector(middle, minY, (maxZ + dist - minZ) / 2));
	}
	else
	{
		wall3->SetRelativeLocation(FVector(middle, maxY + dist, (maxZ + dist - minZ) /
2));
		wall4->SetRelativeLocation(FVector(middle, minY - dist, (maxZ + dist - minZ) /
2));
		roof->SetRelativeLocation(FVector(maxX - ((maxX - minX) / 2), maxY - ((maxY -
minY) / 2), maxZ + dist));
	}
	}

	void AMainActor::ClearMeshData()
	{
	_vertices.Empty();
	_triangles.Empty();
	_trianglesRPC.Empty();
	_uV0.Empty();
	_normals.Empty();
	}
```