Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» грудня 2023 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «КОНФІГУРАТОР РОЗМІЩЕННЯ ЕЛЕМЕНТІВ ІНТЕР'ЄРУ БУДІВЛІ НА ОСНОВІ ТЕХНОЛОГІЇ UNREAL ENGINE»

Захищено на засіданні
Атестаційної комісії № 40
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:

Студент 2 курсу, групи КІ– 61
за спеціальністю 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології
Титаренко Олег Вячеславович

Керівник: д.т.н., с.н.с., професор кафедри теоретичної та прикладної системотехніки
ТОЛСТОЛУЗЬКА Олена Геннадіївна

Рецензент: к.ф.м.н., доцент, в.о. завідувача кафедри електроніки і управляючих систем _____
ХРУСЛОВ Максим Михайлович

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і трьох додатків. Загальний обсяг роботи складає 86 сторінки, із яких 69 сторінок основної частини з 16 рисунками, 30 найменуваннями списку використаних джерел та трьома додатками.

Метою кваліфікаційної роботи є розробка інноваційного комп'ютерного конфігуратора, який дозволить користувачам ефективно планувати та візуалізувати розміщення елементів інтер'єру в будівлях. Використання технології Unreal Engine у цій роботі має на меті забезпечити високу якість візуалізації та реалістичність зображень, що є ключовим для точного та ефективного проектування інтер'єрів.

Об'єкт дослідження – Процеси розміщення та взаємодія з обєктами елементів інтер'єру в просторових моделях будівель за допомогою комп'ютерних технологій.

Предмет дослідження – Комп'ютерний конфігуратор на базі технології Unreal Engine для інтерактивного розміщення та елементів інтер'єру в будівлях та взаємодії з ними.

Проблема, яка вирішується в кваліфікаційній роботі, є необхідність створення ефективного, інтерактивного та візуально реалістичного інструменту для планування та візуалізації інтер'єру будівель. Цей інструмент, який розробляється на основі технології Unreal Engine, має на меті вирішити проблему складності візуалізації та концептуалізації дизайну інтер'єру в реальному часі, забезпечуючи динамічну взаємодію з елементами дизайну. Вона допомагає користувачам краще розуміти та оцінювати різні конфігурації простору, матеріали, кольори, меблі та інші елементи інтер'єру, дозволяючи швидко вносити зміни та побачити їх вплив на загальний вигляд простору. Такий підхід є важливим для підвищення ефективності проектування інтер'єрів, скорочення

часу та витрат, пов'язаних з традиційними методами дизайну, та покращення взаємодії між дизайнерами та їх клієнтами

Ключові слова: Конфігуратор інтер'єру, Unreal Engine, C++, 3D візуалізація, Моделювання простору, blueprints, реалізація проекту, програмне забезпечення, підхід, Архітектура, апаратне забезпечення.

ABSTRACT

The explanatory note for the master's thesis consists of an introduction, three chapters, conclusions, a list of used sources, and two appendices. The total volume of the work is 86 pages, of which 69 pages belong to the main part with 16 figures, the 30 names of used sources, and three appendices.

The aim of the qualification work is to develop an innovative computer configurator that will allow users to efficiently plan and visualize the placement of interior elements in buildings. The use of Unreal Engine technology in this work aims to provide high-quality visualization and realism of images, which is crucial for accurate and effective interior design.

The research object is the processes of placement and interaction with objects of interior elements in spatial building models using computer technologies.

The research subject is a computer configurator based on Unreal Engine technology for interactive placement of interior elements in buildings and interaction with them.

The problem addressed in the qualification work is the need for an efficient, interactive, and visually realistic tool for planning and visualizing interior spaces in buildings. This tool, developed based on Unreal Engine technology, aims to address the complexity of real-time visualization and conceptualization of interior design, providing dynamic interaction with design elements. It helps users better understand and evaluate different space configurations, materials, colors, furniture, and other interior elements, allowing for quick changes and seeing their impact on the overall appearance of the space. Such an approach is essential for enhancing the efficiency of interior design, reducing the time and costs associated with traditional design methods, and improving interaction between designers and their clients.

Keywords: Interior configurator, Unreal Engine, C++, 3D visualization, Space modeling, blueprints, project implementation, software, approach, Architecture, hardware.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UE — Unreal Engine

VR — Virtual Reality

DI — Digital Interior

ERI — Ergonomic Interface

DP — Design Process

CDT — Contemporary Digital Technology

IVM — Interactive Visualization Method

UBT— Unreal Build Tool

CFG — Configurator

TEX — Textures

LGT — Lighting

OBJ — Objects

CPT — Computing Power

UI — User Interface

FNC — Functionality

PT— Path Tracer

UHT — Unreal Header Tool

ВСТУП

Актуальність цифрових технологій у сучасному дизайні інтер'єру є незаперечною, з огляду на стрімке впровадження інноваційних рішень в цій галузі. Розвиток технологій віртуальної реальності та їх застосування в архітектурному дизайні стали ключовими факторами, які формують нові підходи до візуалізації та реалізації дизайнерських проектів. У цьому контексті, кваліфікаційна робота, представлена у даному документі, відіграє важливу роль, демонструючи практичне застосування цих технологій через створення конфігуратора інтер'єру на базі Unreal Engine.

Ця робота є підсумком глибокого дослідження в області цифрового дизайну та програмування, з акцентом на реалістичному відтворенні кожного аспекту інтер'єру, від текстур до освітлення. Проект спрямований на створення інструменту, який не тільки спростить процес розробки дизайну, але й забезпечить вищу гнучкість та якість кінцевого продукту, дозволяючи користувачам здійснювати інтерактивні зміни в реальному часі.

Детальний огляд існуючих методів візуалізації інтер'єру та аналіз ринкових інструментів став базою для розробки цього конфігуратора. Основну увагу було зосереджено на ергономічності інтерфейсу, оптимізації робочого процесу та інтеграції передових технологій, які відповідають сучасним стандартам цифрового дизайну.

Процес створення конфігуратора включав розробку від ідеї до технічного втілення, при цьому були враховані потреби кінцевих користувачів та специфіка роботи дизайнерів. Програмний продукт, який було розроблено, містить комплекс функцій для вибору та розміщення об'єктів, налаштування освітлення, кольору, текстур і інших параметрів. Важливою особливістю є його здатність адаптуватися до різних рівнів потужності обладнання, що робить його доступним для широкого кола користувачів.

У підсумку, кваліфікаційна робота демонструє не лише актуальність та необхідність використання цифрових технологій у сфері дизайну інтер'єру, а й

реалізує практичний інструмент, здатний суттєво підвищити ефективність та якість дизайнерської праці. Робота відкриває нові перспективи для подальших досліджень та розробок в області цифрового дизайну, надаючи цінний внесок у розвиток даного напрямку.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ РОЗРОБКИ КОМП'ЮТЕРНОЇ МОДЕЛІ КОНФІГУРАТОРА РОЗМІЩЕННЯ ЕЛЕМЕНТІВ ІНТЕРЄРУ БУДІВЛІ НА ОСНОВІ ТЕХНОЛОГІЇ UNREAL ENGINE

1.1. Дефініція та призначення конфігуратора

1.1.1. Визначення конфігуратора

Конфігуратор – це програмне забезпечення або інтерактивний інструмент, розроблений для допомоги користувачам у виборі, налаштуванні, компонуванні продуктів або послуг згідно з їхніми специфічними потребами чи вимогами. Ключовою особливістю конфігуратора є його здатність динамічно відображати результати на основі вибору користувача, забезпечуючи таким чином візуальний зворотний зв'язок.

Конфігуратор застосовується у різних сферах, наприклад, у дизайні продуктів, таких як автомобілі, комп'ютери, меблі або одяг, дозволяючи вибирати колір, форму, розмір, стиль та аксесуари. Він також використовується для вибору послуг, як-от туристичних пакетів, страхових планів, підписок або курсів, дозволяючи налаштовувати місце призначення, тривалість, бюджет, покриття та зміст послуги. Крім того, конфігуратори застосовуються для налаштування систем, як-то мереж, програмного забезпечення, баз даних чи вебсайтів, шляхом встановлення параметрів, опцій, функцій та функціональності [1].

Переваги конфігураторів включають збільшення задоволеності клієнтів, підвищення диференціації та інноваційності продуктів чи послуг, зниження витрат та покращення ефективності процесів. Вони також допомагають оптимізувати конфігураційний процес, забезпечуючи сумісність та реалізованість рішення.

У контексті інтер'єру будівлі та 3D моделювання, конфігуратор дозволяє дизайнерам, архітекторам або кінцевим користувачам взаємодіяти з елементами

простору в реальному часі, перевіряти різні комбінації матеріалів, кольорів, меблів та інших елементів інтер'єру. Це може бути вкрай корисним для візуалізації фінального вигляду приміщення або для демонстрації потенційним покупцям чи клієнтам [2].

1.1.2. Роль конфігуратора в дизайні інтер'єру

Конфігуратори грають ключову роль у процесі дизайну інтер'єру, оскільки вони забезпечують динамічне і гнучке середовище для моделювання та візуалізації просторових рішень. Основними аспектами впливу конфігураторів на дизайн інтер'єру є [3]:

Інтерактивність і Кастомізація: Конфігуратори дозволяють дизайнерам і клієнтам взаємодіяти з елементами дизайну в реальному часі. Це означає, що можна швидко змінювати компонування, кольори, матеріали та інші аспекти дизайну, щоб побачити, як це вплине на загальний вигляд та відчуття простору.

Візуалізація та Передбачення: Сучасні конфігуратори, особливо ті, що використовують Unreal Engine, забезпечують високоякісну візуалізацію, яка допомагає краще уявити кінцевий результат. Це важливо для прийняття обґрунтованих рішень щодо дизайну інтер'єру, дозволяючи клієнтам та дизайнерам ефективно співпрацювати.

Ефективність та Зниження Витрат: Використання конфігураторів може значно скоротити час та витрати, пов'язані з проектуванням інтер'єру. Замість фізичного створення макетів або декількох дизайнів, конфігуратор дозволяє швидко моделювати та оцінювати різні варіанти, знижуючи ймовірність помилок та неефективних рішень.

Відповідність Клієнтським Очікуванням: Конфігуратори забезпечують можливість створення дизайнів, які точно відповідають вимогам та побажанням клієнтів. Завдяки гнучкості конфігурацій, клієнти можуть відчувати себе більш включеними у процес дизайну, що підвищує їхнє задоволення кінцевим результатом.

Інтеграція з Іншими Системами: Сучасні конфігуратори часто інтегруються з іншими інструментами та системами, наприклад, з системами управління проектами або CRM-системами. Це дозволяє створювати більш злагоджені та ефективні робочі процеси.

1.2. Типи конфігураторів

Продуктові конфігуратори призначені для налаштування конкретних продуктів. Вони дозволяють користувачам вибирати різні опції, такі як колір, матеріал, розмір, щоб створити продукт, який відповідає їхнім уподобанням. Це особливо поширено в роздрібній торгівлі та електронній комерції, де клієнти можуть налаштовувати все від меблів до одягу та ювелірних виробів. Використання таких конфігураторів забезпечує більшу задоволеність клієнтів, оскільки вони отримують продукт, що відповідає їхнім індивідуальним побажанням.

Просторові конфігуратори зосереджені на взаємодії з просторовими елементами, такими як меблі чи декорації у віртуальному просторі. Вони дають можливість користувачам маніпулювати об'єктами в тривимірному просторі, щоб перевірити різні варіанти розміщення та дизайну. Цей тип конфігуратора є важливим інструментом для архітекторів та дизайнерів інтер'єру, дозволяючи їм та їх клієнтам візуально оцінити і планувати дизайн простору перш ніж перейти до реалізації.

Візуальні конфігуратори фокусуються на естетичних аспектах, таких як кольори, текстури, та візуальні ефекти. Цей тип конфігуратора широко використовується у сферах, де візуальний аспект є критичним - наприклад, у дизайні одягу чи текстилю. Візуальні конфігуратори надають можливість користувачам легко експериментувати з різними візуальними варіантами, значно спрощуючи процес вибору дизайну.

Функціональні конфігуратори зосереджені на технічних аспектах продуктів, таких як їхні властивості та функціональність. Ці конфігуратори зазвичай використовуються у промисловому секторі, де важливо точно

налаштувати технічні параметри продукту. Вони дозволяють користувачам вибирати специфікації та компоненти, щоб створити продукт, який повністю відповідає їхнім технічним вимогам та очікуванням [4].

1.3. Програмне забезпечення для створення конфігураторів

1.3.1. Unity

Unity, розроблений компанією Unity Technologies, був випущений у 2005 році і спочатку націлений на зроблення розробки 3D-ігор доступнішою для незалежних розробників. З того часу двигун пройшов значну еволюцію, розширивши свої можливості і виріс у один з найпопулярніших ігрових двигунів на світовому ринку. Особливістю Unity є його мультиплатформність. Двигун дозволяє експортувати проекти на понад 20 різних платформ, включаючи ПК, мобільні пристрої, ігрові консолі, а також системи віртуальної та доповненої реальності. Це робить Unity ідеальним інструментом для розробників, які прагнуть досягти широкої аудиторії. Unity включає інтуїтивно зрозумілий візуальний редактор, який полегшує розміщення об'єктів, редагування сцен та інтеграцію активів. Вбудований фізичний двигун дозволяє реалістично моделювати взаємодії в ігровому світі, а візуальний шейдерний редактор дозволяє розробникам створювати та налаштовувати матеріали без необхідності писати код [5]. Програмування в Unity засноване на мові C#, яка є сучасною, об'єктно-орієнтованою мовою і відомою своєю легкістю у вивченні та широкою підтримкою. Скриптова архітектура Unity дає можливість швидко змінювати поведінку та взаємодії в грі, що робить розробку гнучкою та інтерактивною. Unity відрізняється гнучкістю у ліцензуванні. Для стартапів та незалежних розробників доступна безкоштовна версія, тоді як для більших комерційних проектів та студій пропонуються спеціалізовані платні опції. Двигун підтримується великою глобальною спільнотою, що постійно розвивається та допомагає одне одному через форуми, групи та навчальні матеріали. Unity застосовується не тільки у сфері розробки ігор. Його гнучкість і потужність

знайшли використання у таких галузях, як архітектура, медицина, освіта, де він використовується для створення проектів віртуальної та доповненої реальності. Unity здобув світове визнання завдяки своїм відмінним можливостям, сильній підтримці спільноти та здатності працювати на різних платформах [6].

Гнучке Управління Активами: Unity дозволяє ефективно управляти активами, включаючи 3D-моделі, текстури, анімації та звукові файли. Це забезпечує зручність організації та інтеграції різноманітного контенту в проект.

Розширені Можливості Анімації: Unity містить потужні інструменти для створення та редагування анімацій, що дозволяє розробникам створювати плавні та динамічні анімації для персонажів та об'єктів.

Кросплатформне Розгортання: Unity забезпечує кросплатформність, дозволяючи розробникам легко розгортати свої проекти на різних платформах, включаючи Windows, MacOS, iOS, Android, різні VR/AR системи та багато інших.

Система Скриптів на C#: Unity використовує C# як мову програмування, надаючи розробникам потужні та гнучкі можливості для створення ігрової логіки та інтерактивності.

Інтеграція Фізичного Двигуна: Unity включає в себе фізичний двигун, який дозволяє реалістично симулювати фізичні взаємодії в цифровому світі, надаючи реалістичність рухам та динаміці сцен.

Інтуїтивний Візуальний Редактор: Unity надає зручний і інтуїтивно зрозумілий інтерфейс, який дозволяє розробникам легко розміщувати та маніпулювати об'єктами в сцені, а також налаштовувати освітлення та камери.

Підтримка Спільноти та Ресурси: Unity має велику та активну спільноту розробників. Це забезпечує доступ до численних навчальних ресурсів, готових активів, плагінів та скриптів, що допомагає новачкам швидко освоїти платформу та допомагає досвідченим розробникам підвищити продуктивність своєї роботи.

Адаптивність до Різних Проектів: Unity не обмежується лише розробкою ігор. Його можна застосовувати у різних сферах, таких як архітектура, освіта,

медицина та інші, для створення інтерактивних візуалізацій та проектів віртуальної реальності [7].

1.3.2. WebGL

WebGL (Web Graphics Library) є JavaScript API для рендеринга 2D та 3D графіки безпосередньо в браузері, що уникає необхідності використання будь-яких плагінів. Базуючись на OpenGL ES 2.0, WebGL може використовувати апаратне прискорення графіки, що доступне на багатьох сучасних системах.

Однією з головних переваг WebGL є його кросплатформність. Завдяки тому, що контент інтегрований в браузер, він може працювати на будь-якій платформі або пристрої, який підтримує сучасний веб-браузер. Ця універсальність робить WebGL ідеальним для створення контенту, доступного на широкому спектрі пристроїв.

Крім того, він не вимагає установки додаткового програмного забезпечення або плагінів, роблячи його легко доступним та простим у використанні. Завдяки здатності використовувати графічний процесор (GPU) для рендеринга, WebGL забезпечує високу продуктивність та деталізацію графіки [8].

Для роботи з WebGL, багато розробників використовують вищерівневі JavaScript-бібліотеки, такі як Three.js та Babylon.js. Ці бібліотеки полегшують створення 3D-сцен, матеріалів, світла та інших елементів графіки, роблячи процес більш доступним та менш складним, особливо для тих, хто не має глибоких знань у графіці.

Однак, при роботі з WebGL, розробники стикаються з певними викликами. Оскільки WebGL працює на низькому рівні, існують потенційні проблеми безпеки. Крім того, хоча більшість сучасних браузерів підтримує WebGL, деякі старіші версії або специфічні налаштування можуть не підтримувати цю технологію. Також продуктивність може відрізнятися в залежності від пристрою, браузера та драйверів графічної карти [9].

Враховуючи ці аспекти, WebGL вимагає від розробників глибокого розуміння графіки та здатності працювати з низькорівневим API. Розробка зазвичай включає ініціалізацію WebGL, створення шейдерів, визначення геометрії об'єктів, роботу з уніформами та текстурами, рендеринг сцени та оптимізацію. Для дебагінгу та аналізу графічної піплайні використовуються спеціалізовані інструменти, такі як WebGL Inspector. Також WebGL часто інтегрується з іншими веб-API, розширюючи можливості інтерактивних застосунків [10].

1.4. Загальний огляд технології Unreal Engine

1.4.1. Історія та розвиток Unreal Engine

Unreal Engine, один з найвідоміших ігрових двигунів, був створений компанією Epic Games. Його перша версія, Unreal Engine 1, була випущена у 1998 році разом з грою "Unreal". Первісно двигун був розроблений для потреб шутерів від першої особи, але з часом його можливості значно розширились, перетворивши його на універсальний інструмент для створення ігор різних жанрів.

Найновіша версія, Unreal Engine 5, була анонсована в 2020 році та вражає своїми інноваціями у графіці та реалізмі, пропонуючи ще більш потужні інструменти для створення фотореалістичних сцен та динамічного освітлення. Завдяки таким технологіям, як Nanite та Lumen, Unreal Engine 5 відкриває нові горизонти для розробників у різних галузях, розширюючи межі можливого в цифровій візуалізації [11].

1.4.2. Основні функції та можливості Unreal Engine у розробці конфігураторів

Unreal Engine, розроблений Epic Games, є одним з найпотужніших інструментів для створення ігор та інтерактивних досвідів. Він відомий своїми

передовими можливостями, які відкривають широкий спектр можливостей для розробників у різних галузях.

Unreal Editor: Основний редактор движка, який працює за принципом "те, що ви бачите, те й отримаєте". Іншими словами, фінальний результат буде таким самим, як те, що розробники бачать в тривимірному вікні. Редактор дуже зручний у використанні і заощаджує багато часу команди - всі ресурси, такі як моделі, візуальні ефекти, джерела світла і т. д., можна розмістити на сцені, просто перетягуючи їх з папок. Фактично, Unreal Editor є комплексною системою, розробленою для надання розробникам всіх необхідних інструментів в одному місці [12].

Blueprint: Unreal Engine дозволяє розробникам використовувати власну візуальну систему програмування, Blueprint, що робить движок доступним навіть для користувачів без навичок програмування. Blueprint ідеально підходить для розробки та швидкого тестування нових ідей.

Sequencer: Інструмент для створення кінематографічних сцен. Його функціональність схожа на звичайний відеоредактор: кожен елемент - звук, анімація і т. д. - має свій власний трек, де можна додавати ключові кадри або редагувати необхідний сегмент окремо. Іншими словами, Sequencer надає можливості для повного редагування відео без використання сторонніх сервісів, що досить зручно [13].

World Composition: Unreal Engine полегшує розробку великих ігор з відкритим світом, надаючи систему потокової передачі рівнів. З його допомогою розробники можуть створювати плавний дизайн рівнів з ілюзією безперервної наративності та оптимізацією продуктивності. Практично це дає гравцям можливість вільно переміщатися між рівнями без екрану завантаження - всі дані про розташування завантажуються в пам'ять і просто відображаються або приховуються при переміщенні користувача.

Control Rig: Інструмент для маніпулювання кістками персонажа в редакторі, що значно спрощує процес створення ріггінгу.

Плагіни: Unreal Engine також обладнаний вбудованими плагінами від Epic Games. З їх допомогою розробники можуть швидко додавати створені моделі персонажів з MetaHuman Creator, високоякісні ресурси з Quixel Megascans або ресурси з Unreal Marketplace [14].

1.4.3. Переваги та недоліки роботи з Unreal Engine

Переваги:

Приголомшлива графіка: Unreal Engine відомий своєю високоякісною графікою та реалістичними ефектами освітлення. Він підтримує такі передові функції, як трасування променів, глобальне освітлення, динамічні тіні та пост-обробні ефекти. Unreal Engine може створювати візуально вражаючі ігри, які можуть конкурувати з AAA-титулами.

Гнучке програмування: Unreal Engine підтримує як мову програмування C++, так і візуальні скриптові мови Blueprint. C++ - потужна та широко використовувана мова програмування, яка дає розробникам повний контроль над логікою гри та продуктивністю. Blueprint - це графічний інтерфейс, який дозволяє розробникам створювати логіку гри без написання коду. Blueprint ідеально підходить для прототипування, тестування та розробки ігрових функцій. Unreal Engine також має багатий набір інструментів та редакторів, які полегшують та прискорюють розробку.

Підтримка багатьох платформ: Unreal Engine може розгортати ігри на різних платформах, включаючи Windows, Mac, Linux, iOS, Android, PlayStation, Xbox, Nintendo Switch та VR/AR пристрої. Unreal Engine автоматично обробляє специфічні вимоги та оптимізації для кожної платформи, тому розробники можуть зосередитися на дизайні та функціональності гри.

Недоліки:

Крута крива навчання: Unreal Engine - це складний та витончений ігровий движок, який вимагає багато часу та зусиль для освоєння. Він має багато функцій та опцій, які можуть бути важкими для початківців. Unreal Engine також має інший робочий процес та термінологію, ніж інші ігрові движки, тому

розробникам потрібно вивчити спосіб роботи Unreal Engine. Unreal Engine може не бути найкращим вибором для простих або казуальних ігор, які не потребують його повного потенціалу.

Випадкові збої та помилки: Unreal Engine - не ідеальний ігровий движок, і він може мати проблеми зі стабільністю та сумісністю. Деякі користувачі повідомляють, що Unreal Engine випадково збивається або зависає під час розробки. Unreal Engine також може мати деякі помилки або збої, які впливають на продуктивність або функціональність гри. Unreal Engine може потребувати частих оновлень та патчів для виправлення цих проблем [15].

1.4.4. Програмування на C++ у Unreal Engine

Програмування на C++ в Unreal Engine - це спосіб створення логіки гри, користувацького інтерфейсу, функціональності редактора та багато іншог. Програмування в Unreal Engine схоже на стандартний C++, але має деякі відмінності та розширення, які полегшують роботу з можливостями та системами двигуна.

Unreal Engine надає два методи, C++ та візуальне сценаріювання Blueprint, для створення нових геймплейних елементів. C++ є основною мовою програмування в Unreal Engine і використовується для створення основних класів двигуна, геймплейного каркасу, інструментів редактора та багато іншого. Blueprint - це візуальна система сценаріювання, яка дозволяє створювати логіку та геймплей без написання коду, використовуючи вузли та зв'язки для представлення функцій та змінних. Blueprint базується на C++ і може отримувати доступ до класів та функцій, які викладені з C++. Ви також можна створювати власні вузли Blueprint за допомогою C++, що може розширити функціональність та можливості Blueprint [16].

Однією з ключових функцій Unreal Engine для програмування на є Unreal Header Tool (UHT), який є спеціальним попереднім процесором, який аналізує заголовкові файли C++ і генерує додатковий код для двигуна. UHT дозволяє використовувати макроси Unreal, які є спеціальними ключовими словами, що

починаються з UPROPERTY, UFUNCTION, UCLASS тощо. Ці макроси дозволяють нам відкривати наші класи, функції та змінні для Unreal Engine Reflection System, яка є системою, що дозволяє динамічні функції, такі як серіалізація, реплікація, збір сміття, інтеграція з редактором та доступ Blueprint. Наприклад, використовуючи макрос UPROPERTY, ми можемо зробити наші змінні редагованими в редакторі, видимими в дебагері та доступними в Blueprint. Використовуючи макрос UFUNCTION, ми можемо зробити наші функції викликаними в Blueprint, перекритими в підкласах та під'єднаними до подій.

Ще одною ключовою функцією є Unreal Build Tool (UBT), який є спеціальною системою збірки, яка відповідає за компіляцію та зв'язування вашого коду на C++ та коду двигуна. UBT використовує модулі для організації коду в логічні одиниці, і кожен модуль має файл .build.cs, який визначає його залежності, налаштування та правила. UBT також використовує файли цілей для визначення конфігурації та виводу проекту, такого як гра, редактор, сервер і т. д. UBT може автоматично виявляти зміни в коді і перезбирати тільки ті модулі, які були змінені, що може прискорити процес розробки.

Unreal Engine також надає багатий набір інструментів та функцій для полегшення програмування, таких як:

Unreal Editor: Основний інструмент для створення та редагування проектів, ресурсів, рівнів та блупрінтів. Unreal Editor також має режим VR, який дозволяє створювати та будувати світи в віртуальному середовищі реальності, використовуючи всі можливості набору інструментів редактора.

Unreal Debugger: Інструмент, який дозволяє відлагоджувати ваш та логіку блупрінтів, встановлювати точки зупинки, перевіряти змінні, обчислювати вирази та інше.

Unreal Automation Tool (UAT): Інструмент, який дозволяє автоматизувати різні завдання та робочі процеси, такі як упаковка, тестування, розгортання та сценаріювання.

Unreal Testing Framework (UTF): Фреймворк, який дозволяє створювати та запускати автоматизовані тести для коду та логіки блупрінтів, використовуючи перевірки, тестові кейси, тестові набори та тестові виконавці [17].

У програмуванні на C++ у Unreal Engine використовуються численні класи, кожен з яких має свої особливості та призначення. Ці класи дозволяють створювати і розширювати функціонал ігор і інтерактивних додатків. Ось детальний опис деяких з цих класів із прикладами коду.

AActor є базовим класом для всіх об'єктів у грі, які можуть взаємодіяти з ігровим світом. Наприклад, якщо ви хочете створити простий об'єкт, який може бути розміщений у світі, ви можете створити клас, який успадковується від AActor:

```
#include "GameFramework/Actor.h"
#include "MySimpleObject.generated.h"
UCLASS()
class MYGAME_API AMySimpleObject : public AActor
{
  GENERATED_BODY()
  // Конструктор класу
  AMySimpleObject();
};
```

APawn є підкласом AActor і представляє об'єкти, якими може управляти гравець або штучний інтелект (AI). Це основа для створення ігрових персонажів:

```
#include "GameFramework/Pawn.h"
#include "MyGamePawn.generated.h"
UCLASS()
class MYGAME_API AMyGamePawn : public APawn
{
  GENERATED_BODY()
  // Функції та змінні для управління Pawn
```

```
};
```

ACharacter є спеціалізованим підкласом APawn для персонажів, що пересуваються пішки. Він має вбудовану підтримку для ходьби, бігу, стрибків та інших рухових дій:

```
#include "GameFramework/Character.h"
#include "MyGameCharacter.generated.h"
UCLASS()
class MYGAME_API AMyGameCharacter : public ACharacter
{
  GENERATED_BODY()
  // Конструктор та функції для персонажа
};
```

Ці класи служать основою для створення основних елементів ігор у Unreal Engine, дозволяючи розробникам створювати багатий і різноманітний ігровий світ. Використання цих класів дозволяє точно налаштовувати поведінку і властивості ігрових об'єктів, надаючи необхідну гнучкість для реалізації унікальних ігрових механік.

У програмуванні на C++ у Unreal Engine, ряд ключових методів використовуються для реалізації ігрової логіки та управління об'єктами. Ось детальний опис цих методів з прикладами коду, що демонструють їх використання.

BeginPlay():

Метод **BeginPlay()** викликається на початку гри або коли об'єкт створюється і часто використовується для ініціалізації об'єктів. Наприклад, якщо вам потрібно налаштувати початкові параметри об'єкта, це можна зробити тут:

```
void AMyGamePawn::BeginPlay()
{
  Super::BeginPlay();
  // Ініціалізація та налаштування
```

```
}
```

Tick(float DeltaTime):

Tick(float DeltaTime) є методом, який викликається на кожному кадрі гри. Це ідеальне місце для оновлення логіки, такої як пересування об'єктів або відстеження стану гравця:

```
void AMyGamePawn::Tick(float DeltaTime)
{
   Super::Tick(DeltaTime);
   // Оновлення логіки об'єкта
}
```

OnOverlapBegin() та OnOverlapEnd():

Ці методи використовуються для виявлення та обробки колізій між об'єктами. Наприклад, якщо гравець входить у визначену зону або зіштовхується з предметом [18]:

```
void AMyInteractiveObject::OnOverlapBegin(class UPrimitiveComponent* OverlappedComp,
class AActor* OtherActor,
class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex,
bool bFromSweep, const FHitResult& SweepResult)
{
   // Логіка, що виконується при вході в зону
}
void AMyInteractiveObject::OnOverlapEnd(class UPrimitiveComponent* OverlappedComp,
 class AActor* OtherActor,
 class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex)
{
   // Логіка, що виконується при виході з зони
}
```

1.4.5. Використання Blueprints у Unreal Engine

Система Blueprints у Unreal Engine, є візуальною системою скриптів, яка дозволяє розробникам та дизайнерам ігор створювати комплексну логіку без необхідності глибоких знань у програмуванні. Blueprints дозволяють візуалізувати код як схему з вузлами, які представляють різні функції та операції. Вони інтегровані безпосередньо в Unreal Editor, що надає інтуїтивно зрозумілий інтерфейс для створення ігрової логіки. Blueprints відрізняються від традиційного програмування на C++ тим, що вони мінімізують необхідність написання коду, пропонуючи більш графічний та інтерактивний підхід.

Архітектура Blueprints у Unreal Engine включає декілька ключових компонентів, які дозволяють розробникам і дизайнерам створювати комплексну ігрову логіку та взаємодію в іграх. Основні компоненти Blueprints включають Event Graph, Function Graph, Variables та інші елементи, які разом формують гнучку та потужну систему для візуального програмування. Event Graph у Blueprints є місцем, де визначається логіка реагування на різні ігрові події, такі як натискання кнопок, зміна станів об'єктів чи зіткнення. Це дозволяє реалізувати різні сценарії взаємодії у грі, від простих до складних. Function Graph забезпечує можливість створювати власні функції всередині Blueprint. Ці функції можуть включати серію вузлів, що виконують певні дії або обчислення, тим самим розширюючи стандартні можливості Blueprints. Variables в Blueprints використовуються для зберігання даних, які можуть змінюватися протягом ігрового процесу. Змінні можуть бути різних типів, таких як числа, рядки, об'єкти або навіть складніші структури даних. Інтеграція Blueprints з Unreal Editor дозволяє візуально керувати всіма аспектами створення ігрової логіки. Розробники можуть безпосередньо взаємодіяти з елементами інтерфейсу, налаштовувати властивості та зв'язувати різні частини ігрового процесу через графічний інтерфейс, що робить процес розробки більш інтуїтивним та доступним. Процес створення Blueprint починається з вибору відповідного базового класу. Наприклад, для створення ігрового персонажа, може бути вибраний клас Character. Це робиться через меню "Add New" у Content Browser,

де розробники можуть вибрати потрібний тип Blueprint. Після створення Blueprint, розробники мають можливість візуально програмувати логіку ігрового процесу через інтерфейс Blueprint Editor. Це включає розміщення вузлів для подій, таких як натискання кнопки або входження в тригерну зону, та додавання логічних послідовностей, які визначають реакцію гри на ці події [19].

Blueprints можуть взаємодіяти з класами і функціями, написаними на C++, що розширює можливості обох підходів. Розробники можуть створювати кастомні класи на C++, які потім можуть бути використані в Blueprints. Наприклад, якщо розробник створює клас персонажа на C++, цей клас може мати функції та змінні, які будуть доступні для використання в Blueprint. Це дає можливість використовувати складні алгоритми та операції, написані на C++, безпосередньо в візуальному скрипті Blueprint. З іншого боку, Blueprints можуть бути використані для прототипування або реалізації швидких змін в логіці гри, після чого логіку можна перенести або переписати на C++ для оптимізації та подальшого розвитку. Це дає можливість швидко експериментувати з ідеями в Unreal Engine, а потім використовувати переваги C++ для ефективності та гнучкості.

Одним із ключових сценаріїв використання Blueprints є прототипування ігрових елементів та механік. Завдяки візуальному інтерфейсу Blueprints, дизайнери ігор можуть швидко створювати та тестувати нові ідеї, експериментуючи з різними аспектами геймплея. Це особливо корисно на ранніх етапах розробки, коли потрібно швидко оцінити відчуття та функціональність ігрових елементів. У більш складних проектах Blueprints використовуються для створення інтерактивності в ігровому світі. Це може включати створення складних взаємодій між гравцем та ігровим середовищем, таких як відкриття дверей, взаємодія з об'єктами або активація пасток. Blueprints дозволяють легко візуалізувати і налаштувати ці взаємодії, роблячи процес розробки більш інтуїтивним та доступним. Крім того, Blueprints часто використовуються для створення інтерфейсу користувача, анімації персонажів та керування камерою.

Ці інструменти дозволяють розробникам контролювати візуальні та інтерактивні аспекти гри, забезпечуючи плавну і залучену ігрову взаємодію.

Використання Blueprints у Unreal Engine пропонує значні переваги, але також має певні обмеження, які важливо враховувати під час розробки ігор. Однією з основних переваг Blueprints є їхнє візуальне середовище, яке робить процес програмування більш доступним і зрозумілим, особливо для тих, хто не має глибоких знань у кодуванні. Це ідеально підходить для прототипування, дозволяючи швидко експериментувати з ігровими механіками та взаємодією. Інша перевага полягає в тому, що Blueprints дозволяють швидко впроваджувати зміни у грі без необхідності компіляції коду, що пришвидшує процес розробки. Також вони спрощують співпрацю між різними членами команди, такими як дизайнери та програмісти, оскільки Blueprints надають чітке візуальне представлення логіки гри. Тим не менш, існують обмеження, які слід враховувати [20].

1.4.6. Технологія Path Tracer

Path Tracer - це режим візуалізації в Unreal Engine, який використовує трасування променів для створення реалістичних і фізично точних зображень. Він доступний в Unreal Engine 5.0 і 5.3 і може бути використаний для рендерингу сцен з об'ємними ефектами, такими як дим, вогонь і туман. Для активації Path Tracer у проекті потрібна операційна система Windows 10, графічна карта NVIDIA RTX або GTX, а також активація апаратного трасування променів у налаштуваннях проекту. Також можете використовувати Path Tracer разом із Sequencer і Movie Render Queue для створення високоякісних кінематографічних відео.

Path Tracer ґрунтується на тій ж архітектурі трасування променів, що й інші функції трасування променів в Unreal Engine, такі як трасування променів у реальному часі та GPU Lightmass. Проте, Path Tracer не використовує той самий код трасування променів, який був розроблений для ефективної роботи в режимі реального часу. Замість цього, Path Tracer покладається лише на геометрію та

матеріали сцени для створення безпередбачувого, фотореалістичного та фізично точного результату. Це відбувається без необхідності в попередньо обчислених даних або наближень, забезпечуючи послідовну якість незалежно від складності освітлення та матеріалів сцени. Path Tracer працює шляхом трасування променів світла від камери до сцени, а потім відбиває їх від поверхонь до тих пір, поки вони не потрапляють до джерела світла або не досягають максимальної кількості відбиттів. Кожен промінь несе інформацію про кольори і інтенсивність світла, а також властивості матеріалів поверхонь, з якими він зіштовхується. Кінцевий колір кожного пікселя обчислюється шляхом усереднення внесків усіх променів, які його вдаряють. Цей процес повторюється для кожного кадру, і результат поступово удосконалюється з часом, доки не стає стабільним зображенням.

Nanite: Path Tracer може рендерити сцени з мешами, які підтримують Nanite, це високодеталізовані та оптимізовані меші, які можна імпортувати безпосередньо з вихідних високополігонових активів, таких як ZBrush або фотограметричні скани. Path Tracer може обробляти мільйони трикутників на кожному меші, не пожертвуючи продуктивністю або якістю.

Path Tracer повністю інтегрований з Sequencer і Movie Render Queue, які є інструментами, що дозволяють створювати кінематографічні відео в Unreal Engine. Ви можете використовувати Path Tracer для рендерингу високоякісних кадрів з рухомим блюром, глибиною різкості та згладжуванням, а потім експортувати їх у вигляді послідовностей зображень або відеофайлів. Ви також можете використовувати Path Tracer разом із системою компонування Composure, яка дозволяє комбінувати зйомки на реальному місці з комп'ютерно-графічними елементами в Unreal Engine.

Path Tracer не призначений для рендерингу в реальному часі, оскільки для створення зображень без шуму він вимагає великої обчислювальної потужності та часу. Path Tracer найкраще підходить для візуалізації, кінематографії та остаточного виробництва. Path Tracer не підтримує деякі функції, які доступні в інших режимах рендерингу, такі як декалі, функції освітлення, профілі світла, світлові валіки та відбитки об'єктивів. Також Path Tracer не підтримує деякі

ефекти післяопрацювання, такі як відображення в просторі екрану, оклюзія оточуючого середовища та картографування тонів. Path Tracer може створювати артефакти "світлячки", які є яскравими пікселями, які з'являються випадково на зображенні через події з низькою ймовірністю, такі як удар об'єктивом дуже малої джерела світла або дуже відбиваючої поверхні. [21].

1.4.7. Віртуальна реальність (VR) у Unreal Engine

Віртуальна реальність (VR) - це технологія, яка дозволяє користувачам переживати іммерсивні та інтерактивні симуляції різних середовищ. VR може використовуватися для розваг, освіти, навчання та багатьох інших цілей. Unreal дозволяє розробникам створювати захоплюючі VR-враження з високоякісною графікою, реалістичною фізикою та динамічним звуком.

Unreal Engine підтримує різноманітні VR-пристрої, такі як Oculus, SteamVR, Windows Mixed Reality та OpenXR. Ми можемо використовувати шаблон VR як вихідну точку для своїх проектів VR, який включає логіку телепортації. Шаблон VR використовує фреймворк OpenXR, який є стандартом для розробки VR та AR і розроблений кількома компаніями. З допомогою плагіна OpenXR в Unreal Engine ми можемо зробити свої проекти VR сумісними з різними платформами і пристроями без будь-яких перевірок або викликів.

Unreal Engine також надає багатий набір інструментів та функцій для покращення проектів VR, таких як:

Sequencer: кінематографічний інструмент, який дозволяє створювати кінематографічні послідовності з анімаціями, камерами, освітленням та звуком у VR4.

Niagara: система візуальних ефектів, яка дозволяє створювати вражаючі частинкові ефекти та симуляції у VR4.

Audio Mixer: система звуку, яка дозволяє створювати іммерсивний та просторовий звук у VR4. Blueprints: візуальна система скриптів, яка дозволяє створювати логіку та геймплей без написання коду у VR4.

UMG: система користувацького інтерфейсу, яка дозволяє створювати інтерфейси користувача та меню у VR4.

Мультиплеер: мережева система, яка дозволяє створювати багатокористувацькі та соціальні враження у VR4[22].

1.5. Створення 3D-моделей для Unreal Engine

1.5.1. Методологія та інструменти 3D-моделювання

Моделювання в тривимірному форматі - це процес створення тривимірних моделей різних типів або форм об'єктів, які математично представлені в трьох вимірах. Тривимірне моделювання може використовуватися для різних цілей, таких як розваги, освіта, інженерія, архітектура та багато інших. Для тривимірного моделювання використовуються програми та інструменти, які можуть варіюватися в залежності від типу та складності моделі

Методологія тривимірного моделювання - це підхід або техніка, яка використовується для створення тривимірної моделі. Існує багато типів методів тривимірного моделювання, але деякі з найпоширеніших включають[23]:

Box modeling: Цей метод включає в себе початок з простого об'єкта, такого як куб або сфера, і використання класичних моделювальних інструментів для його зміни та вдосконалення у бажану форму. Box modeling підходить для об'єктів з твердою поверхнею, таких як будівлі, автомобілі та меблі. Box modeling легко вивчити та контролювати, але він може бути нудним і часомірним для складних форм. Box modeling підтримується більшістю програм для тривимірного моделювання, таких як Blender, 3ds Max і Maya [24].

Polygon modeling: Цей метод включає в себе створення тривимірної моделі шляхом визначення та з'єднання окремих полігонів, які є плоскими поверхнями з трьома або більше сторонами. Метод полігонного моделювання схожий на метод box modeling, але він надає більше гнучкості і точності у формуванні моделі. Полігонне моделювання підходить для органічних і деталізованих об'єктів, таких як персонажі, тварини та рослини. Полігонне моделювання є

більш високорозвинутим та вимагає більше навичок та знань, але воно може створювати реалістичні та високоякісні моделі. Полігонне моделювання також підтримується більшістю програм для тривимірного моделювання [25].

Nurbs та curve modeling: Цей метод включає в себе створення тривимірної моделі за допомогою кривих та поверхонь, які визначаються математичними формулами, такими як сплайни, криві Без'є та Nurbs. Nurbs та curve modeling може створювати гладкі та безперервні форми, які ідеально підходять для дизайну промислових виробів, автомобільного дизайну та дизайну ювелірних виробів. Nurbs та curve modeling також можуть використовуватися для створення складних форм, які важко досягти за допомогою полігонів, таких як труби, труби та пляшки. Nurbs та curve modeling легко редагувати та маніпулювати, але вони можуть бути викликані в перешкоди під час конвертації у полігони для рендерингу та анімації. Nurbs та curve modeling підтримуються деякими програмами для тривимірного моделювання [26].

Digital 3D sculpting: Цей метод включає в себе створення тривимірної моделі за допомогою інструментів, які симулюють поведінку інструментів для скульптури у реальному світі, таких як пензлики, глина та інструменти для вирубування. Цифровий 3D скульптинг може створювати органічні та деталізовані моделі, схожі на традиційну скульптуру, такі як людські обличчя, тіла та одяг. Цифровий 3D скульптинг також може використовуватися для додавання деталей та текстур до існуючих моделей, таких як зморшки, пори та шрами. Цифровий 3D скульптинг є інтуїтивним та виразним, але він також може вимагати великої обчислювальної потужності та високороздільних сіток. Цифровий 3D скульптинг підтримується спеціалізованим програмним забезпеченням для тривимірного моделювання [27].

Фотограметрія: Цей метод включає в себе створення тривимірної моделі за допомогою фотографій реального об'єкта чи сцени. Фотограметрія може зафіксувати форму, колір та текстуру об'єкта чи сцени з високою точністю та реалізмом. Фотограметрія може використовуватися для створення моделей історичних об'єктів, природних пейзажів та культурних артефактів. Для

фотограметрії потрібно багато фотографій з різних кутів та умов освітлення, а також програма, яка може обробити та об'єднати їх в одну тривимірну модель. Фотограметрія підтримується деякими програмами для тривимірного моделювання [28].

1.5.2. Процес створення моделей та їх інтеграція в Unreal Engine

Процес створення 3D моделей та їх інтеграція в Unreal Engine охоплює декілька комплексних етапів, кожен з яких вимагає певних знань та навичок. На початку розробки моделі, ретельно аналізуються вимоги проекту, з урахуванням таких факторів, як рівень деталізації, текстуризація, та функціональність моделі в середовищі Unreal Engine.

Перший крок – це детальне проектування моделі, що включає створення концепт-малюнків та 3D скетчів. Цей етап має велике значення, оскільки він визначає візуальні та функціональні характеристики моделі. Після цього відбувається безпосереднє моделювання. З використанням спеціалізованих програм, таких як Blender, Maya чи 3ds Max, створюється 3D модель. На цьому етапі важливо звернути увагу на топологію моделі, оскільки вона впливає на ефективність та якість анімації.

Далі йде процес текстуризації та налаштування матеріалів, який включає накладення текстур, кольорів, та інших візуальних атрибутів на модель. Цей етап вимагає глибокого розуміння світлових ефектів та взаємодії матеріалів зі світлом у віртуальному середовищі.

Після завершення моделі вона імпортується в Unreal Engine, де проходить подальшу настройку. В Unreal Engine виконується налаштування освітлення, фізики, та інших інтерактивних елементів, що забезпечують інтеграцію моделі в загальний геймплей або сцену. На цьому етапі також можуть бути додані анімації та інші динамічні елементи.

Особливу увагу потрібно приділити оптимізації моделі для забезпечення її ефективної роботи в рамках двигуна Unreal Engine, особливо з урахуванням обмежень, пов'язаних з обчислювальною потужністю та пам'яттю.

Завершальний етап – тестування та налагодження, в ході якого виявляються та виправляються помилки, вдосконалюється взаємодія моделі з іншими елементами віртуального середовища, та здійснюється кінцева настройка параметрів [29].

1.6. Сучасні підходи до 3D-моделювання в архітектурі та дизайні інтер'єру

Одним із сучасних підходів до 3D моделювання в архітектурі та інтер'єрному дизайні є параметричний дизайн, який є методом створення 3D моделей на основі параметрів та правил, що визначають відносини та поведінку елементів моделі. Параметричний дизайн дозволяє дизайнерам створювати складні та динамічні форми, які можуть адаптуватися до різних контекстів та сценаріїв, а також оптимізувати продуктивність та ефективність моделі. Параметричний дизайн також може сприяти співпраці та ітерації, оскільки модель може бути легко змінена та оновлена шляхом зміни параметрів. Параметричний дизайн підтримується програмним забезпеченням, таким як Grasshopper, Dynamo та Revit, яке дозволяє дизайнерам створювати та маніпулювати 3D моделями за допомогою графічних інтерфейсів та візуального програмування.

Третій сучасний підхід до 3D моделювання в архітектурі та інтер'єрному дизайні - це фотограмметрія, яка є методом створення 3D моделей за допомогою фотографій реального об'єкту або сцени. Фотограмметрія може точно та реалістично відтворювати форму, колір та текстуру об'єкту або сцени. Фотограмметрія може використовуватися для створення моделей історичних місць, природних ландшафтів та культурних артефактів. Фотограмметрія вимагає багато фотографій з різних кутів та в різних умовах освітлення, а також програмного забезпечення, яке може обробляти та складати їх разом у 3D модель. Фотограмметрія підтримується деякими програмами для 3D моделювання, такими як Agisoft Metashape, RealityCapture та Meshroom [30].

1.7. Інтерактивність та користувацький досвід у конфігураторах

Інтерактивність та користувацький досвід - це два важливих аспекти конфігураторів, які є програмними інструментами, що дозволяють користувачам налаштовувати та візуалізувати продукти згідно з їхніми перевагами та потребами. Інтерактивність означає ступінь та якість взаємодії між користувачем та конфігуратором, тоді як користувацький досвід відноситься до загального задоволення та емоцій, які користувач відчуває під час використання конфігуратора.

Інтерактивність у конфігураторах може бути виміряна за декількома вимірами, такими як:

Зворотній зв'язок: Здатність конфігуратора надавати негайні та чіткі відповіді на дії та введення користувача, такі як оновлення зображення продукту, показ ціни та відображення повідомлень про помилки.

Контроль: Здатність користувача маніпулювати та досліджувати варіанти та характеристики продукту, такі як зміна кольору, розміру, форми та матеріалу, обертання та збільшення зображення продукту.

Навігація: Здатність користувача пересуватися та отримувати доступ до різних частин та етапів процесу конфігурації, таких як вибір категорії продукту, моделі та варіанту, перегляд та підтвердження замовлення.

Керівництво: Здатність конфігуратора допомагати та підтримувати користувача у прийнятті рішень та завершенні процесу конфігурації, таких як надання рекомендацій, пропозицій та пояснень, та перевірка сумісності та доступності варіантів продукту.

Залучення: Здатність конфігуратора привертати та утримувати увагу та інтерес користувача, таких як використання анімацій, звуків та елементів ігрового процесу, та створення відчуття занурення та присутності. Користувацький досвід у конфігураторах може бути вплинутий декількома факторами, такими як:

Зручність використання: Легкість та ефективність використання конфігуратора, такі як інтуїтивність, простота та швидкість інтерфейсу та

функціональності, а також наскільки добре вони відповідають очікуванням та цілям користувача.

Естетика: Візуальна привабливість та привабливість конфігуратора, такі як приємність, гармонійність та послідовність дизайну та розташування, а також наскільки добре вони відображають ідентичність бренду та якість продукту.

Емоції: Емоційний та емоційний вплив конфігуратора, такі як насолода, веселість та задоволеність від процесу конфігурації, а також наскільки добре він викликає позитивні почуття та емоції у користувача.

Цінність: Сприйнята користь та цінність конфігуратора, такі як корисність, допомога та інформативність процесу конфігурації, а також наскільки добре він задовольняє потреби та бажання користувача.

1.8. Виклики та перспективи у розробці інтер'єрних конфігураторів

Конфігуратори інтер'єру можуть надавати багато переваг як для клієнтів, так і для бізнесу, наприклад, підвищуючи задоволеність клієнтів, збільшуючи продажі, знижуючи витрати та поліпшуючи імідж бренду. Однак розробка конфігураторів інтер'єру також ставить перед собою багато викликів та вимагає ретельного планування та виконання. Деякі з викликів та перспектив у розробці конфігураторів інтер'єру:

Інтерфейс користувача та досвід користувача: Дизайн та функціональність інтерфейсу користувача та досвіду користувача є ключовими для успіху конфігураторів інтер'єру, оскільки вони визначають, наскільки легко, інтуїтивно та приємно для користувачів процес налаштування. Інтерфейс користувача та досвід користувача повинні балансувати між простотою та складністю, реалізмом та абстракцією, функціональністю та естетикою, зворотнім зв'язком та контролем, зацікавленістю та керівництвом. Інтерфейс користувача та досвід користувача також повинні відповідати цільовій аудиторії, категорії продукту та ідентичності бренду бізнесу. Для створення зручного та привабливого інтерфейсу користувача та досвіду користувача потрібно багато досліджень, тестувань та ітерацій.

3D моделювання та рендеринг: Якість та точність 3D моделей та рендерингів інтер'єрних продуктів та середовищ є важливими для реалізму та занурення конфігураторів інтер'єру, оскільки вони впливають на те, як користувачі сприймають та оцінюють продукти та їх варіанти. 3D моделі та рендеринги повинні відтворювати форму, колір, текстуру та матеріал продуктів, а також освітлення, тіні та відображення середовищ. Створення реалістичних та деталізованих 3D моделей та рендерингів вимагає багато навичок, інструментів та ресурсів.

Знання про продукт та логіка: Знання та логіка інтер'єрних продуктів та їх опцій та функцій є основою конфігураторів інтер'єру, оскільки вони визначають, як продукти можуть налаштовуватися та конфігуруватися користувачами. Знання про продукт та логіка повинні включати специфікації, ціни, доступність, сумісність та обмеження продуктів та їх опцій, а також правила та алгоритми, що керують процесом конфігурації. Отримання, організація та оновлення знань про продукт та логіки вимагає багато співпраці, комунікації та документації.

Продуктивність та оптимізація: Продуктивність та оптимізація конфігураторів інтер'єру важливі для ефективності та надійності процесу налаштування, оскільки вони впливають на швидкість та плавність роботи конфігураторів та відповідь на дії та вводи користувача. Продуктивність та оптимізація конфігураторів інтер'єру залежать від розміру, складності та роздільної здатності 3D моделей та рендерингів, швидкості та стабільності Інтернет-з'єднання та сумісності та функціональності веб-браузерів та пристроїв. Покращення продуктивності та оптимізації конфігураторів інтер'єру вимагає багато аналізу, вимірювання та регулювання.

1.9. Висновок

У даному розділі ми розглянули процес створення конфігураторів інтер'єру та їх важливість у сучасному дизайні та архітектурі, з акцентом на використанні Unreal Engine як інструменту для реалізації високоякісних візуалізацій. Особлива

увага приділялася інтерактивності та користувацькому досвіду, як ключовим елементам успішного конфігуратора, що забезпечує гнучкість і привабливість для кінцевих користувачів. Важливість Unreal Engine в цьому контексті полягає у його здатності створювати деталізовані, реалістичні візуалізації інтер'єрів, що є критичним для точного відображення продуктів. Разом з тим, процес інтеграції та використання цього потужного інструменту вимагає глибоких технічних знань та розуміння специфіки роботи двигуна. З цього випливає, що успіх розробки конфігураторів тісно пов'язаний з вмінням ефективно використовувати можливості Unreal Engine, а також з забезпеченням високої інтерактивності та зручності користувацького інтерфейсу. Це, у свою чергу, відкриває широкі перспективи для подальшого розвитку технологій у цій галузі.

РОЗДІЛ 2. РОЗРОБКА КОНФІГУРАТОРА РОЗМІЩЕННЯ ЕЛЕМЕНТІВ ІНТЕРЄРУ БУДІВЛІ НА ОСНОВІ ТЕХНОЛОГІЇ UNREAL ENGINE

2.1 Розробка архітектури та структури компонентів конфігуратора

Структура та компоненти конфігуратора, розробленого на базі Unreal Engine для оптимізації розміщення елементів інтер'єру, визначаються взаємодією між дизайнерами та клієнтами, а також інструментами, які вони використовують. Конфігуратор включає в себе інтерактивні інструменти редагування, що дозволяють дизайнерам управляти доступом до конфігуратора, створювати та редагувати бібліотеки матеріалів і активів проекту, а також редагувати функції, використані в проекті. За допомогою цих інструментів дизайнер може вносити зміни до запланованих елементів та матеріалів у сцені проекту, а також керувати освітленням, часом доби і напрямком камери.

Клієнти мають змогу інтерактивно переглядати проект, вносити зміни до запланованих елементів сцени, взаємодіяти з предметами в сцені та керувати освітленням і камерою в реальному часі. Це дозволяє клієнтам отримати краще розуміння проекту та забезпечує більш ефективний процес затвердження дизайну.

Загальна діаграма комп’ютерної моделі зображена на рис. 2.1.

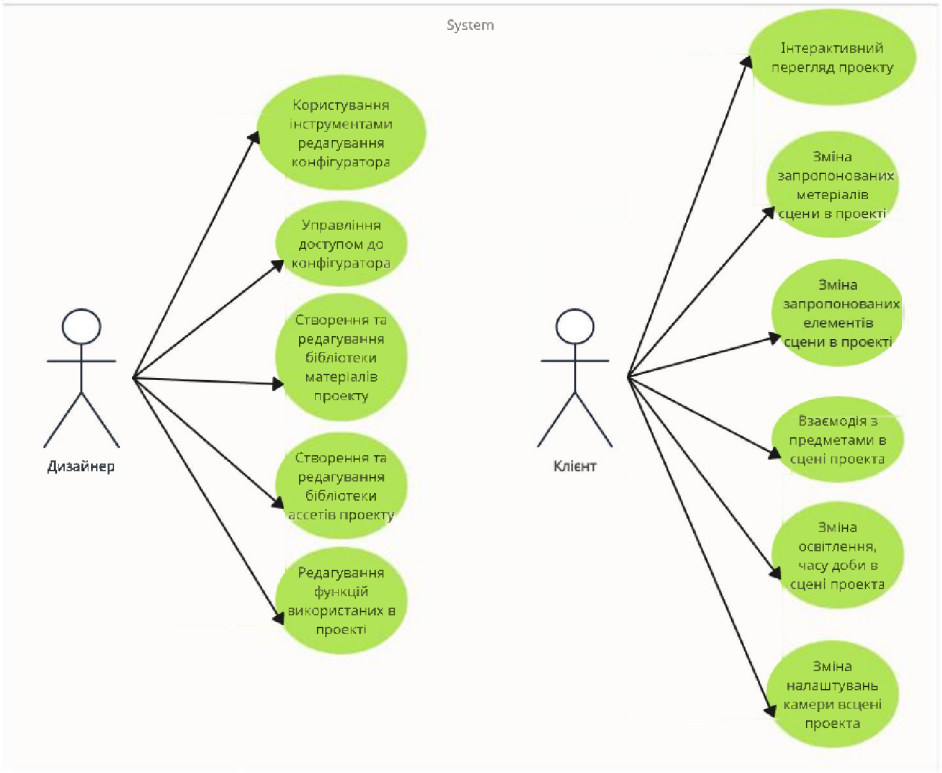

Рисунок 2.1 — Діаграма варіантів використання (UseCase diagram)

Компонентна діаграма, що представлена у цьому розділі, ілюструє внутрішню структуру системи конфігуратора рис. 2.2. На діаграмі чітко визначені основні модулі програми та їх взаємодія, що підкреслює логічну архітектуру та інтеграцію компонентів. Кожен з модулів виконує специфічний набір функцій, необхідних для забезпечення гнучкості та ефективності роботи конфігуратора. Детальний опис компонентів пояснює їх функціональне призначення та важливість для загальної системи, спрощуючи розуміння процесу розробки та взаємозв'язків усередині програмного забезпечення.

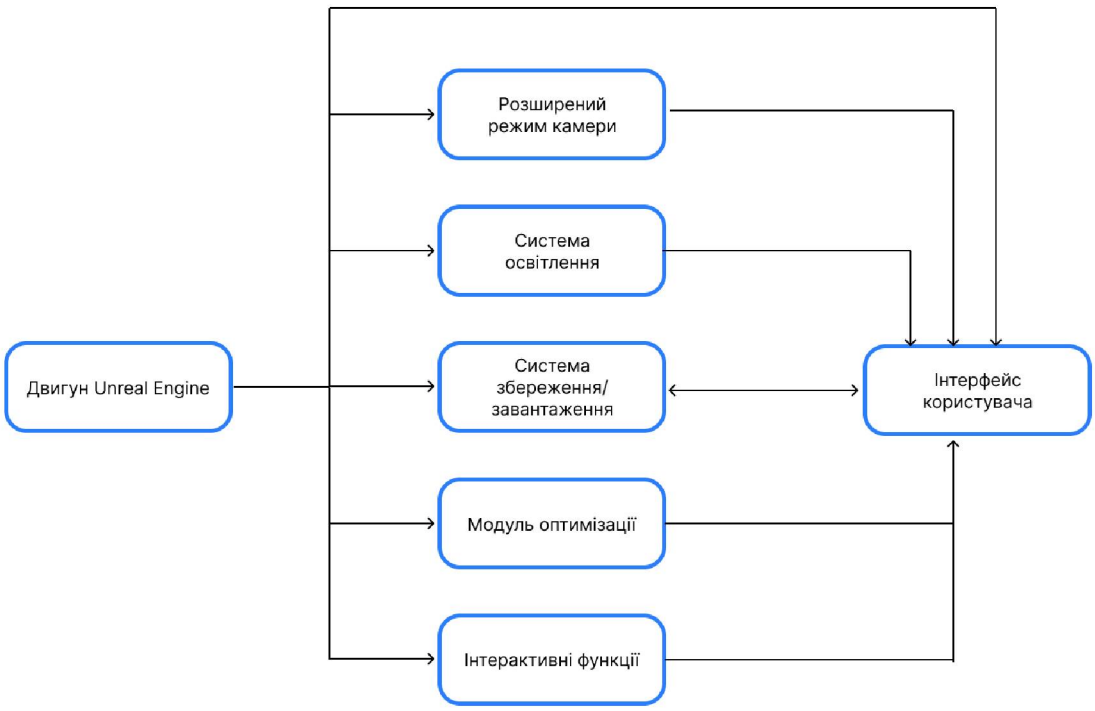

Рисунок 2.2 — Діаграма компонентів

Двигун Unreal Engine: Ядро системи, що забезпечує основні можливості для роботи конфігуратора, включаючи рендеринг сцени, обробку даних та взаємодію з користувацьким інтерфейсом.

Інтерфейс користувача: Компонент, що включає всі елементи інтерфейсу, через які користувач взаємодіє з конфігуратором. Він є основним засобом для отримання вхідних даних від користувача та відображення відповідної інформації.

Інтерактивні функції: Цей компонент управляє всіма інтерактивними аспектами конфігуратора, такими як зміна меблів, вибір елементів декору та інші інтерактивні дії в інтер'єрі.

Система освітлення: Відповідає за управління освітленням всередині віртуальної сцени, дозволяючи користувачу змінювати інтенсивність та колір світла, а також інші параметри, що впливають на зовнішній вигляд сцени.

Система збереження/завантаження: Компонент, який забезпечує можливість зберігання поточного стану конфігурації інтер'єру та завантаження раніше збережених конфігурацій.

Розширений режим камери: Модуль, що надає додаткові можливості для перегляду інтер'єру, такі як зміна точки зору камери, зум та інші функції, що дозволяють користувачу краще оцінити дизайн інтер'єру.

Модуль оптимізації: Відповідає за підвищення продуктивності та ефективності конфігуратора, оптимізуючи використання ресурсів та забезпечуючи плавність роботи програми.

Діаграма станів є важливою частиною проектування програмного забезпечення, яка дозволяє візуалізувати різні стани, в яких може перебувати система, та переходи між ними. Вона відіграє ключову роль у розумінні поведінки системи та взаємодії користувача з нею. Нижче представлена діаграма станів для конфігуратора інтер'єру, розробленого на основі Unreal Engine, яка описує можливі стани програми та їх зміни відповідно до дій користувача рис. 2.3.

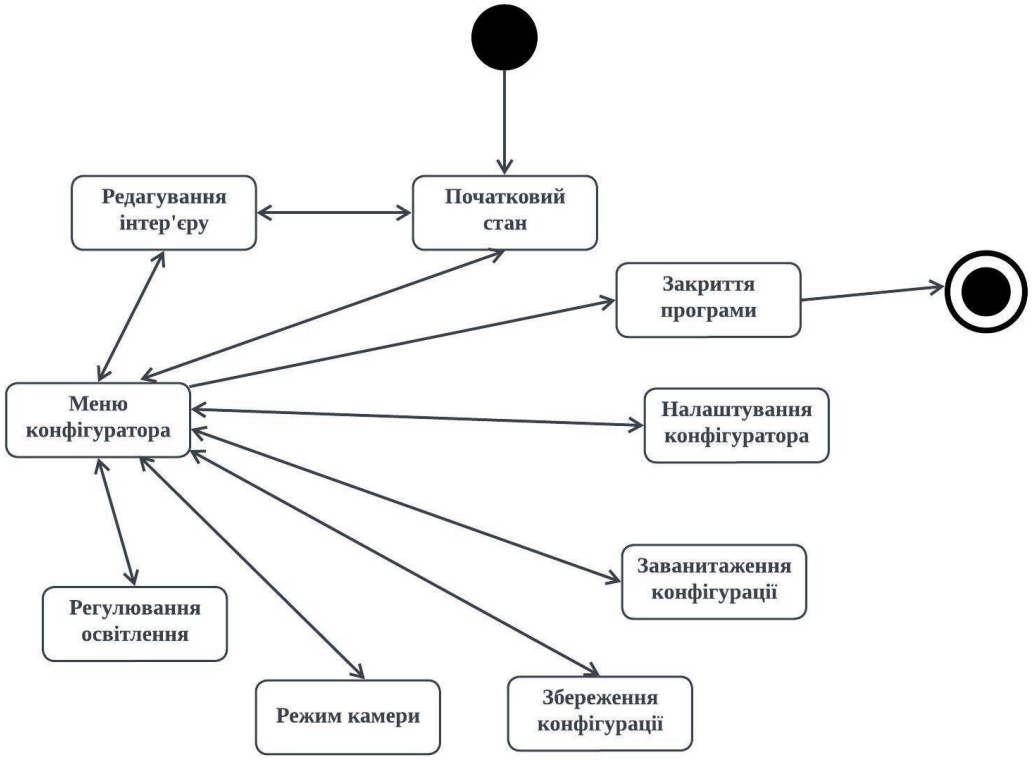

Рисунок 2.3 — Діаграма станів

Детальний опис: На діаграмі ідентифіковано такі основні стани системи:

Початковий стан : вихідна точка в інтеракції користувача з конфігуратором, з якої він може розпочати будь-яку діяльність.

Редагування інтер'єру: стан, в якому користувач може змінювати компоненти інтер'єру.

Регулювання освітлення: стан, у якому користувач налаштовує параметри освітлення в сцені.

Режим камери: стан, який надає можливість огляду виробленого дизайну з різних точок зору.

Збереження конфігурації: стан, у якому відбувається збереження поточних налаштувань конфігурації.

Завантаження конфігурації: стан, де користувач може завантажити збережену раніше конфігурацію.

Закриття програми: фінальний стан, який сигналізує про завершення роботи з конфігуратором.

Налаштування конфігуратора: стан, де користувач може змінити налаштування конфігуратора

Меню конфігуратора: стан, де користувач може перейти в усі інші стани.

2.1.2 Реалізація компоненту “BP_MasterSettings Actor”

У контексті кваліфікаційної роботи було розроблено ключовий компонент — BP_MasterSettings Actor, який є центральним вузлом для налаштування та керування різноманітними параметрами конфігуратора. Цей актор використовує систему Blueprint для візуального програмування, дозволяючи визначити логіку взаємодії з користувачем та управління внутрішніми налаштуваннями програми. Завдяки гнучкості Blueprints, було можливо імплементувати різні аспекти геймплею, такі як управління фоновою музикою, налаштування взаємодії та збереження станів об'єктів при переході між рівнями, забезпечуючи таким чином більш залучений та персоналізований досвід користувача.

Використання BP_MasterSettings Actor у проекті виявилося ефективним способом для уніфікації налаштувань і параметрів, що впливають на роботу

конфігуратора. Це не тільки спростило процес розробки, але й забезпечило можливість швидкої та інтуїтивно зрозумілої модифікації властивостей інтер'єру без потреби в глибоких знаннях програмування. Реалізовані механізми були детально продемонстровані на прикладах скріншотів з Unreal Engine, де візуальні блоки Blueprint описують конкретні дії та умови, на основі яких відбувається управління станами та параметрами гри.

Set Interaction on or off at start: Це логіка, що визначає, чи має бути активована взаємодія з об'єктами конфігуратора на початку гри. Вона отримує інформацію з глобального контексту гри (Game Instance) і використовує її для умовного вузла, який встановлює, чи буде взаємодія включена рис. 2.4.

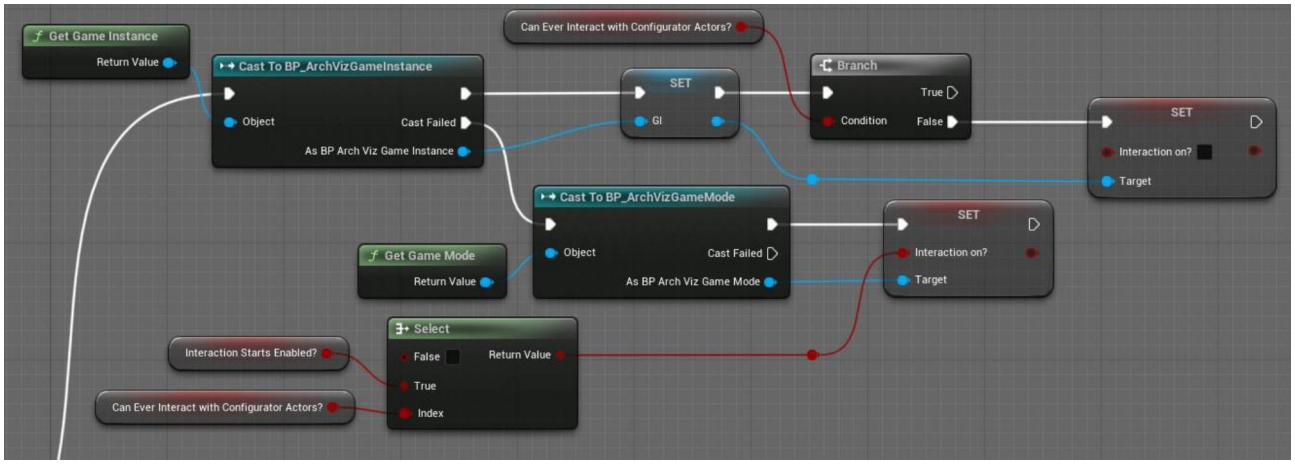

Рисунок 2.4—логіка Set Interaction on or off at start

Background music: Тут встановлюється логіка для фонової музики. Вона перевіряє, чи існує фонова музика і, якщо так, то відтворює її. Якщо параметр "Background Music Play at Start?" установлений у "True", музика буде відтворюватися при запуску. Інакше, якщо фонова музика не повинна відтворюватися, використовується вузол "Stop" для зупинки аудіо-компонента рис. 2.5.

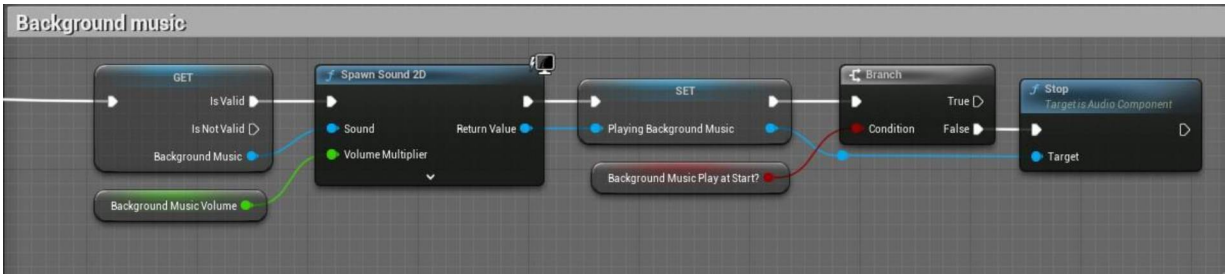

Рисунок 2.5—логіка Background music

Load configurator object state on level transition: Цей Blueprint контролює збереження стану об'єктів конфігуратора під час переходу між рівнями. Здійснюється перевірка валідності глобального контексту (Game Instance) і, за наявності валідного контексту, запускається таймер, який після затримки застосовує зміни до рівня. Потік виконання проходить через цикл "For Each Loop", який ітерує через масив об'єктів та застосовує необхідні зміни до них рис. 2.6.

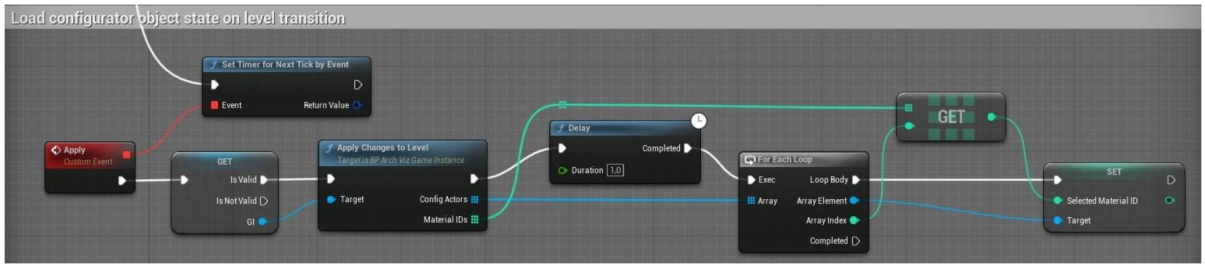

Рисунок 2.6—логіка Load configurator object state on level transition

2.2 Реалізація інтерактивних функцій

2.2.1 Зміна меблів та елементів інтер'єру.

Для реалізації зміни меблів та елементів інтер’єру було використано наступні кроки:

Був створений клас C++ **AReplaceableObject**, який є нащадком **AActor**, для представлення об’єктів, що підлягають заміні. У цьому класі були додані властивості для збереження списку можливих моделей (**StaticMesh**) і компонент для поточної моделі (**StaticMeshComponent**).

У Blueprint редакторі було створено Blueprint клас на основі **AReplaceableObject**. До нього було додано компоненти для візуалізації об'єкта, такі як **BoxCollision**, що використовується для визначення області наведення курсору.

Також було створено Widget Blueprint з назвою **UReplaceMenu** для відображення випадаючого списку. Він містить такі компоненти, як **ScrollBox** для прокручування та **Button** для вибору моделі.

Для контролю гри було створено Blueprint клас **AReplaceGameMode**, де були додані властивості для збереження посилань на об'єкт, який можна замінити (**AReplaceableObject**), і віджет, що відображає випадаючий список (**UReplaceMenu**).

Логіка гри була налаштована у Blueprint редакторі таким чином:

При наведенні курсору на об’єкт, що можна замінити, викликалася функція **OnBeginCursorOver**, яка встановлювала поточний об'єкт у **AReplaceGameMode** і створювала віджет **UReplaceMenu**, заповнюючи його списком моделей з поточного об'єкта. Ця функція також викликала **AddToViewport**, щоб додати віджет до ігрового вікна біля курсору.

При відведенні курсору з об’єкта активувалася функція **OnEndCursorOver**, яка скидала поточний об'єкт у **AReplaceGameMode** і видаляла віджет **UReplaceMenu** з ігрового екрану, використовуючи **RemoveFromParent**.

При натисканні кнопки на віджеті **UReplaceMenu** запускалася функція **OnClicked**, яка змінювала модель поточного об'єкта на вибрану зі списку, застосовуючи **SetStaticMesh** на компоненті **StaticMeshComponent**.

2.2.2 Модифікація матеріалів та текстур.

Було створено клас C++ **AReplaceableMaterial** як нащадок **UObject** для представлення матеріалів, що підлягають заміні. У цьому класі були додані властивості для збереження списку можливих текстур (**Texture2D**) та поточної текстури (**MaterialInstanceDynamic**).

У Blueprint редакторі було створено Blueprint клас на основі **AReplaceableMaterial**. Додано компоненти, необхідні для візуалізації матеріалу, такі як **MaterialBillboard** для відображення текстури у 3D просторі.

Widget Blueprint **UReplaceMenu** було модифіковано так, щоб він міг відображати не тільки список моделей, а й список текстур для обраного об'єкта. Для цього було додано компоненти, такі як **Image** для показу зразка текстури та **Text** для відображення назви текстури.

Логіка гри була змінена у Blueprint редакторі:

При наведенні курсору на замінний об'єкт була активована функція **OnBeginCursorOver**, яка встановлювала поточний об'єкт у **AReplaceGameMode** і створювала віджет **UReplaceMenu**, заповнюючи його списками моделей та текстур. Віджет додавався до ігрового екрану за допомогою **AddToViewport** і розміщувався біля курсору.

При відведенні курсору з об'єкта була активована функція **OnEndCursorOver**, яка скидала обраний об'єкт у **AReplaceGameMode** і видаляла **UReplaceMenu** з ігрового екрану, викликаючи **RemoveFromParent**.

При виборі опції у віджеті **UReplaceMenu** запускалася функція **OnClicked**, яка змінювала модель або текстуру на вибрану зі списку, використовуючи **SetStaticMesh** або **SetTexture** на компоненті **StaticMeshComponent** або **MaterialInstanceDynamic**.

2.3 Система управління освітленням

Для реалізації зміни глобального освітлення було зроблені наступні дії:

Було створено динамічне світлове джерело типу Directional Light для імітації сонця, та були налаштовані його параметри, зокрема кут, інтенсивність та колір, щоб додати більше реалізму сцені.

Для відображення неба, хмар, зірок тощо була створена небесна сфера (Sky Sphere), з налаштованими параметрами освітлення, кольору та анімації.

Розроблено матеріал для небесної сфери, який динамічно змінюється в залежності від положення світлового джерела. У матеріалі були використані функції для контролю висоти сонця, яскравості сонця та градієнту горизонту.

Створено Blueprint, що контролює зміну часу доби та напрямку світла сонця, з використанням змінних для часу доби, ротації сонця та швидкості обертання сонця.

Реалізовано меню, яке дозволяє користувачеві вибирати бажаний час доби та напрямок сонячного світла. Меню містить елементи управління, такі як слайдери, кнопки та текст.

Меню було інтегровано з Blueprint, що керує освітленням, за допомогою функцій для встановлення часу доби, ротації сонця та швидкості зміни часу доби.

Реалізація вимкнення та увімкнення світла:

Було створено Spot Light, який відповідає за світло, яке випромінює об'єкт. Його параметри, такі як інтенсивність, колір та радіус, були налаштовані для досягнення бажаного ефекту.

Розроблено Blueprint Class з Parent Class Actor Component для керування логікою увімкнення та вимкнення світла.

У Blueprint Editor до Blueprint Class було додано Point Light або Spot Light як компонент і зв'язано його зі змінною, що називається Light Component.

В Event Graph була додана подія Begin Play, яка ініціюється при старті гри. За допомогою функції Set Visibility світло було встановлено невидимим за замовчуванням.

Також в Event Graph було додано події On Mouse Enter та On Mouse Leave. При наведенні мишки на об'єкт функція Set Visibility активує світло, а коли курсор покидає область об'єкта — вимикає його.

Після збереження змін у Blueprint Editor було перейдено до Level Editor, де об'єкт, який випромінює світло, було додано на сцену.

В Details Panel до об'єкта було додано створений Blueprint Class як компонент.

2.4 Інтерфейс користувача та система навігації

2.4.1 Розробка меню користувача та налаштувань.

У рамках створення користувацького інтерфейсу для проекту на тему конфігуратора інтер'єру в Unreal Engine 5, було ініційовано створення нового User Interface Blueprint (UI Blueprint). Це було зроблено у Content Browser, де для нового UI Blueprint було обрано Widget Blueprint в якості Parent Class. Цей вибір забезпечив стартовий набір властивостей і функцій, необхідних для подальшої розробки інтерфейсу.

У Designer Mode UI Blueprint було додано кілька ключових елементів інтерфейсу, таких як Text для надання інформації користувачу, Button для виконання команд, Slider для регулювання параметрів (наприклад, розмір об'єктів чи інтенсивність освітлення), та Checkbox для включення чи вимкнення певних елементів інтер'єру, як-от вогні чи прилади. Кожен елемент був ретельно налаштований для забезпечення зручності та естетичності: від розмірів та положення на екрані до стилів шрифтів і кольорової палітри, яка відповідає загальному дизайну проекту.

Застосування Anchor Tool дозволило забезпечити, що елементи інтерфейсу будуть правильно масштабуватися та позиціонуватися незалежно від розміру екрану користувача. Це забезпечило, що інтерфейс залишиться функціональним і естетично привабливим на різних пристроях, від мобільних телефонів до великих моніторів.

Переходячи в Graph Mode, було розроблено логіку, яка визначає поведінку кожного з інтерфейсних елементів. Наприклад, подія On Clicked для кнопок ініціює зміни в інтер'єрі, On Value Changed для слайдерів регулює параметри об'єктів, а On Checked відповідає за включення чи вимкнення певних елементів сцени. Ці події взаємодіють з основними блюпринтами проекту, виконуючи такі дії, як пауза збереження та завантаження стану гри, зміна режиму камери, регулювання глобального освітлення, а також управління звуковим супроводом.

Після завершення всіх налаштувань та програмування в Blueprint Editor, було перейдено до Level Editor для інтеграції розробленого інтерфейсу в ігровий світ. В Details Panel було призначено створений UI Blueprint як компонент до об'єкта в сцені, що дозволило забезпечити його взаємодію з іншими елементами гри.

2.4.2 Система збереження/завантаження конфігурацій.

У рамках розробки конфігуратора елементів інтер'єру в Unreal Engine 5, було створено Save Game Object Blueprint, призначений для зберігання даних про конфігурацію. Цей об'єкт включає в себе інформацію, таку як назва конфігурації, дату створення, список обраних елементів, їх параметри та положення в просторі.

У Graph Mode цього Blueprint було додано змінні, такі як Configuration Name, Configuration Date, Configuration Items, які репрезентують відповідні атрибути конфігурації. Кожна змінна була типізована відповідно до своєї ролі, наприклад, як String для назви, DateTime для дати та Array of Structs для переліку об'єктів.

Після цього Save Game Object Blueprint було збережено та закрито. Далі було створено Game Instance Blueprint, відповідальний за управління збереженнями гри. У ньому були реалізовані функції для створення, видалення та завантаження збережених станів.

У Graph Mode Game Instance Blueprint було розроблено змінні, які відображають поточне збереження, доступні збереження та ім'я слоту збереження. Кожній змінній було призначено відповідний тип, такий як Reference, Array чи String.

До Blueprint були додані функції для реалізації процесу збереження, такі як Create Save Game, Delete Save Game, Load Save Game, Save Configuration та Load Configuration, які забезпечують інтеракцію з файловою системою за

допомогою вбудованих функцій, таких як Does Save Game Exist, Save Game to Slot та інших.

UI Blueprint, який відповідає за меню користувача, було оновлено, додавши елементи для відображення процесів збереження та завантаження. Кожен елемент, такий як Text для відображення інформації, Button для ініціації дій, List View для переліку збережень та Scroll Box для навігації, був налаштований з урахуванням зручності користувача та візуального стилю проекту.

У Graph Mode UI Blueprint були додані події, які активуються при взаємодії користувача з елементами меню, включно з On Clicked для кнопок та On Item Selected для обраних елементів. Ці події взаємодіють з Game Instance Blueprint, викликаючи відповідні функції для збереження або завантаження стану гри.

2.5 Реалізація продвинутого режиму камери

Для розширення функціоналу конфігуратора інтер'єру був доданий продвинутий режим камери, який дозволяє користувачам створювати рендери високої якості з різними візуальними ефектами та налаштуваннями. Ось як було реалізовано цей режим:

Створення камери:

- Було розроблено Blueprint клас камери, який інтегрує вбудовану камеру Unreal Engine з додатковими параметрами для контролю візуальних налаштувань рис. 2.7.

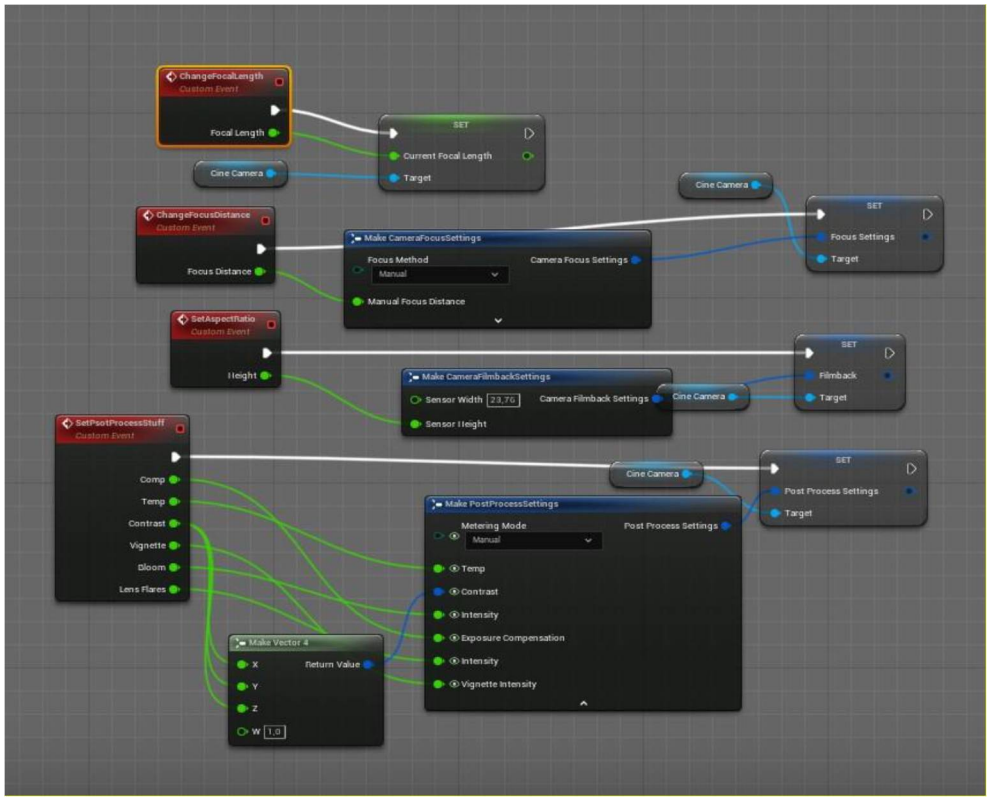

Рисунок 2.7—Реалізація продвинутого режиму камери

Налаштування параметрів камери:

- **Глибина різкості (Depth of Field):** Параметр було налаштовано так, щоб користувачі могли контролювати рівень розмитості фону відносно об'єкта фокусування, додаючи більше глибини та реалістичності зображенню.
- **Фокусна відстань (Focal Length):** Дозволяє користувачам імітувати фізичну фокусну відстань об'єктива камери, змінюючи видовий кут та перспективу зображення.
- **Баланс білого (White Balance):** Коригує колірну температуру зображення, щоб кольори виглядали більш природно в залежності від світлових умов.
- **Експозиція (Exposure):** Контролює загальну яскравість зображення, дозволяючи користувачам налаштувати, як світло або темно має бути рендер.

- **Яркість (Brightness):** Ад'юстування загальної яркості рендера для підкреслення деталей або створення певної атмосфери.
- **Віньєтка (Vignette):** Функція для додавання темних країв навколо зображення, що може допомогти зосередити увагу на центрі кадру.
- **Вибір розширення фото (Resolution Selection):** Було додано опції для вибору роздільної здатності рендера, включаючи 1080p, 2K, 4K та 8K, забезпечуючи гнучкість для різних потреб візуалізації, від веб-презентацій до високоякісних друкованих матеріалів.

Інтеграція з UI:

- У користувацькому інтерфейсі було створено окремі контроли для кожного з параметрів камери, дозволяючи користувачам легко налаштовувати та попередньо переглядати зміни в реальному часі.

Збереження та застосування налаштувань:

- Застосування змін до камери відбувається в реальному часі, з можливістю збереження певних конфігурацій для швидкого доступу в майбутньому.

2.6 Оптимізація

Оптимізація конфігуратора інтер'єру для різних систем та роздільних здатностей є критично важливою для забезпечення його ефективної роботи на різноманітних пристроях. У цьому процесі було досягнуто адаптивності інтерфейсу користувача, що дозволяє елементам інтерфейсу динамічно змінюватися відповідно до розміру та пропорцій екрану різних пристроїв. Також була проведена оптимізація графічних ресурсів, зокрема, автоматичне налаштування якості текстур, освітлення та тіней, в залежності від потужності системи користувача. Важливим аспектом є масштабування роздільної здатності, яке дозволяє користувачам вибирати оптимальну якість зображення, варіюючи від 1080p до 8K, що забезпечує гнучкість та високу якість візуалізації.

Крім того, реалізовано легку зміну налаштувань для користувачів, дозволяючи їм швидко адаптувати графічні параметри, такі як рівні деталізації,

налаштування освітлення та інші візуальні ефекти. Це забезпечує відмінне балансування між візуальною якістю та продуктивністю. Особлива увага була приділена оптимізації продуктивності, використовуючи такі методи як лодинг (LOD) для моделей та ефективні алгоритми рендерингу, що забезпечують стабільну продуктивність навіть на менш потужних системах.

2.7 Тестування конфігуратора

У рамках розробки конфігуратора інтер'єру основну роль у тестуванні та отриманні зворотного зв'язку відігравав я, як розробник. Це дало можливість детально перевірити всі аспекти програми та забезпечити її високу якість та стабільність.

Процес тестування починався з внутрішніх перевірок, під час яких я аналізував базову функціональність конфігуратора, виявляючи та виправляючи помилки. Це включало тестування інтерфейсу користувача, функціональності різних інструментів та загальної продуктивності програми.

Далі було проведено бета-тестування, де обраним користувачам було запропоновано випробувати конфігуратор і надати свої відгуки. Це дозволило отримати важливу інформацію про користувацький досвід, зручність використання та можливі недоліки програми. Зворотний зв'язок був уважно проаналізований, що допомогло внести необхідні покращення та вдосконалення.

Тестування конфігуратора інтер'єру, розробленого на базі Unreal Engine, охоплювало такі аспекти:

1. **Функціональне тестування**: Перевірка відповідності всіх функцій технічним вимогам.
2. **Тестування продуктивності**: Оцінка швидкості реакції програми та оптимізація ресурсів.
3. **Тестування користувацького інтерфейсу (UI)**: Перевірка зручності інтерфейсу та доступності функцій.
4. **Тестування сумісності**: Забезпечення сумісності програми з різними платформами та пристроями.

5. **Тестування безпеки**: Захист даних користувача та стійкість системи.
6. **Регресивне тестування**: Перевірка збереження функціональності після оновлень.
7. **Юзабіліті-тестування**: Оцінка зручності використання та збір відгуків реальних користувачів.

Такий підхід до тестування забезпечив глибоке розуміння потреб користувачів та дозволив вдосконалити конфігуратор, зробивши його більш ефективним та корисним інструментом у сфері дизайну інтер'єру.

2.8 Майбутні оновлення та розвиток

Розвиток цифрових технологій вносить значні зміни у методи візуалізації та розробки дизайну інтер'єру. Конфігуратор інтер'єру, створений на основі Unreal Engine у рамках цієї кваліфікаційної роботи, наразі фокусується на базових аспектах інтерактивності та візуалізації. Попри це, в майбутніх планах розробки продукту передбачається інтеграція можливостей віртуальної (VR) та доповненої реальності (AR), які відкриють нові перспективи для користувачів та дизайнерів.

Хоча VR та AR поки що не підтримуються, розглядається можливість їх впровадження для забезпечення більш іммерсивного взаємодії з цифровими інтер'єрами. Віртуальна реальність дозволить користувачам зануритися в дизайн, отримуючи повноцінне уявлення про простір, тоді як доповнена реальність відкриє можливість перегляду віртуальних об'єктів у реальному фізичному середовищі. Ці технології значно покращать взаємодію з конфігуратором, дозволяючи користувачам не лише бачити, але й відчувати пропоновані зміни до втілення проекту.

Плани розвитку конфігуратора передбачають дослідження та впровадження цих інноваційних технологій, що дозволить залишатися на передовій індустрії дизайну та забезпечить користувачам передовий інструмент для реалізації їхніх творчих візій."

1. Введення Розширених налаштувань Користувача
    - Додавання опцій для персоналізації інтерфейсу та управління налаштуваннями.
    - Реалізація збереження індивідуальних налаштувань користувача.
2. Інтеграція Віртуальної Реальності (VR)
    - Розробка VR-інтерфейсу для іммерсивного перегляду дизайнів.
    - Впровадження інтерактивних VR-елементів для редагування інтер'єру.
3. Функції Доповненої Реальності (AR)
    - Розробка AR-інструментів для перегляду об'єктів дизайну в реальному середовищі.
    - Інтеграція AR-модуля для візуалізації змін у живому режимі.
4. Оптимізація для Різних Платформ
    - Адаптація конфігуратора для різних пристроїв та операційних систем.
    - Вдосконалення продуктивності та сумісності програми.
5. Вдосконалення Інтерактивних Функцій
    - Розширення можливостей редагування та кастомізації об'єктів інтер'єру.
    - Додавання інтелектуальних рекомендацій на основі вибору користувача.
6. Реалізація Штучного Інтелекту (AI)
    - Використання AI для аналізу тенденцій дизайну та автоматичного підбору стилів.
    - Інтеграція AI для оптимізації розміщення об'єктів у просторі.
7. Модуль Зворотного Зв'язку
    - Введення функції збору відгуків від користувачів.
    - Аналіз відгуків для подальшого вдосконалення конфігуратора.

2.9 Висновки

2.9.1 Оцінка досягнутих результатів.

Оцінка досягнутих результатів розробки конфігуратора інтер'єру показала, що були успішно реалізовані ключові цілі проекту. Конфігуратор ефективно інтегрує інноваційні технології Unreal Engine для створення реалістичних візуалізацій інтер'єру, пропонуючи користувачам інтуїтивно зрозумілі інструменти для налаштування елементів дизайну. Розширений режим камери та можливість кастомізації різних аспектів інтер'єру, включаючи освітлення, меблі та декор, значно підвищили гнучкість та ефективність процесу дизайну. Також було досягнуто високої рівня оптимізації для різних систем, що забезпечує стабільну роботу конфігуратора на широкому спектрі пристроїв.

2.8.2 Можливості для подальшого розвитку та удосконалення.

Подальший розвиток та удосконалення конфігуратора можуть включати кілька напрямків. Одним з ключових аспектів є інтеграція штучного інтелекту для автоматизації певних процесів, наприклад, для рекомендацій щодо дизайну на основі переваг користувача або актуальних трендів. Також важливим є розвиток можливостей віртуальної та доповненої реальності, які можуть забезпечити ще більш занурювальний досвід візуалізації інтер'єру. Вдосконалення інструментів колаборації може дозволити кільком користувачам одночасно працювати над проектом, що особливо важливо для професійних дизайнерів. Крім того, розширення бібліотеки активів та матеріалів забезпечить більш широкі можливості для кастомізації дизайну. Нарешті, підвищення зручності та інтуїтивності користувацького інтерфейсу залишається важливим аспектом, що сприяє кращому залученню користувачів та їх задоволенню від процесу дизайну.

РОЗДІЛ 3. РЕКОМЕНДАЦІЇ ПО ВИКОРИСТАННЮ КОНФІГУРАТОРА РОЗМІЩЕННЯ ЕЛЕМЕНТІВ ІНТЕРЄРУ БУДІВЛІ НА ОСНОВІ ТЕХНОЛОГІЇ UNREAL ENGINE

3.1 Введення в інтерфейс та основні функції конфігуратора

3.1.1. Огляд основного меню та інтерфейсу

При інтеграції конфігуратора до проекту, користувачі отримують доступ до демонстраційного рівня, який був заздалегідь створений для тестування ключових функцій інструментарію. Цей демо-рівень дозволяє користувачам оцінити реальні можливості конфігуратора в контексті вже розробленого дизайну інтер'єру. Для доступу до меню конфігуратора необхідно використати клавішу "I" на клавіатурі, після чого перед користувачем відкривається інтерактивне меню рис. 3.1. з рядом наступних опцій:

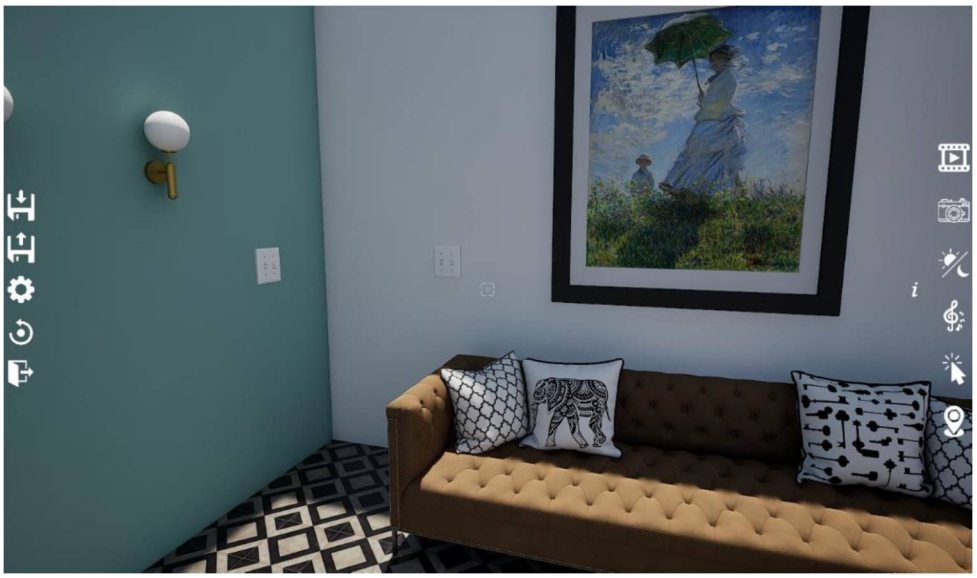

Рисунок 3.1 — Вигляд інтерфейсу конфігуратора

"Зберегти та завантажити рис. 3.2. конфігурацію для декількох дизайнів" надає змогу зберігати різні варіанти дизайну та швидко перемикатися між ними,

що особливо зручно при порівнянні альтернативних рішень або відновленні попередніх налаштувань.

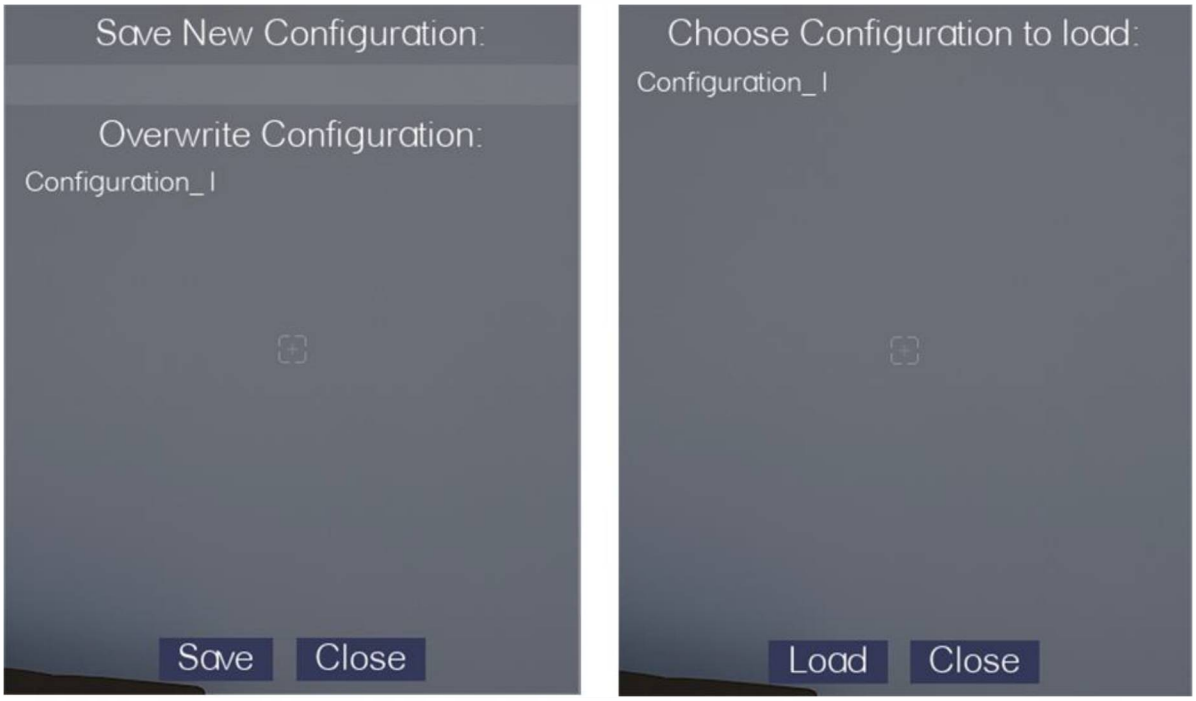

Рисунок 3.2 — Вікна збереження та завантаження конфігурації.

Меню налаштувань рис. 3.3. пропонує різноманітні опції, які дозволяють адаптувати якість графіки та роздільну здатність до технічних характеристик слабших комп'ютерів, забезпечуючи оптимальну продуктивність.

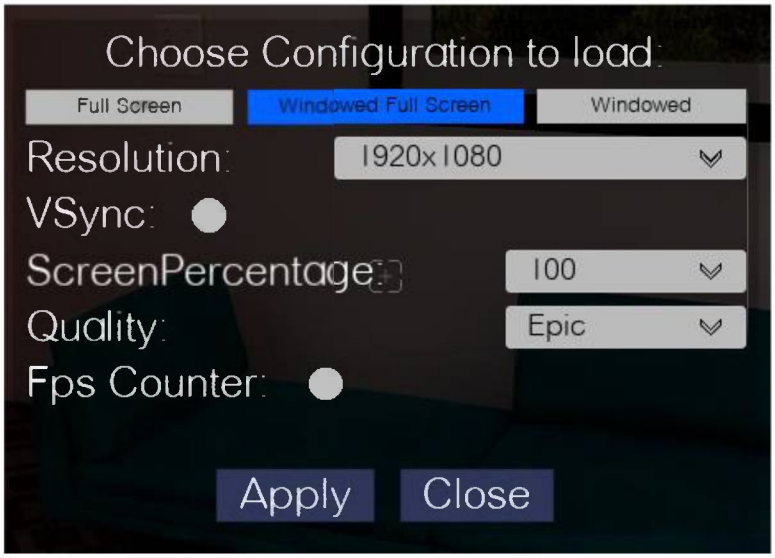

Рисунок 3.3 — Вікно меню налаштуфань конфігуратора.

Функція "Скинути рівень до базової конфігурації" є корисною для ситуацій, коли необхідно відкинути всі зміни та повернутися до початкового стану дизайну.

"Кнопка виходу" забезпечує швидке та зручне закриття програми.

Опція "Почати показ фільму в режимі секвенцера" дає можливість переглянути автоматизовану презентацію проекту, що використовується для демонстрації дизайну у динаміці.

Перемкнутись до режиму камери для збереження зображень рис. 3.4. дозволяє користувачам фіксувати та зберігати скріншоти робочого простору.

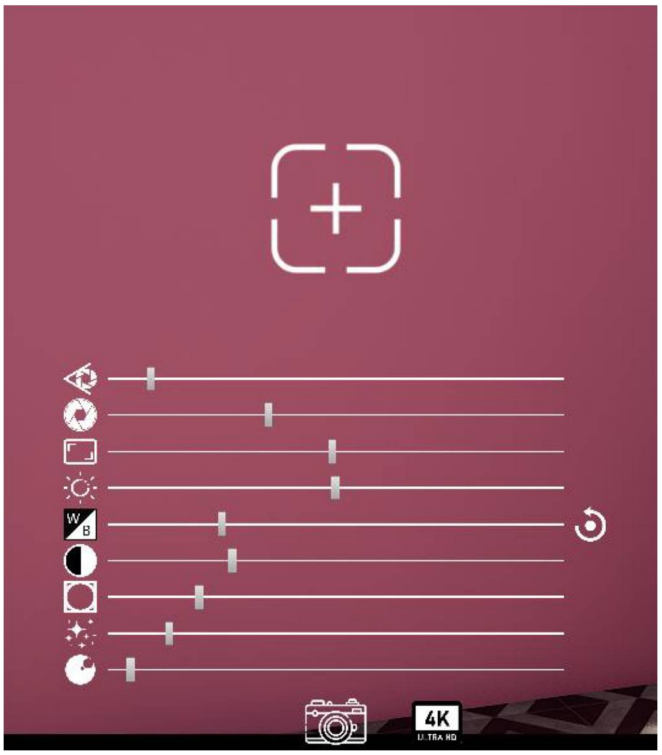

Рисунок 3.4 — Інтерфейс налаштувань режиму камера.

Функція "Змінити час доби та положення сонця" рис. 3.5. надає змогу користувачам візуально оцінити, як освітлення впливає на зовнішній вигляд інтер'єру в різні періоди дня.

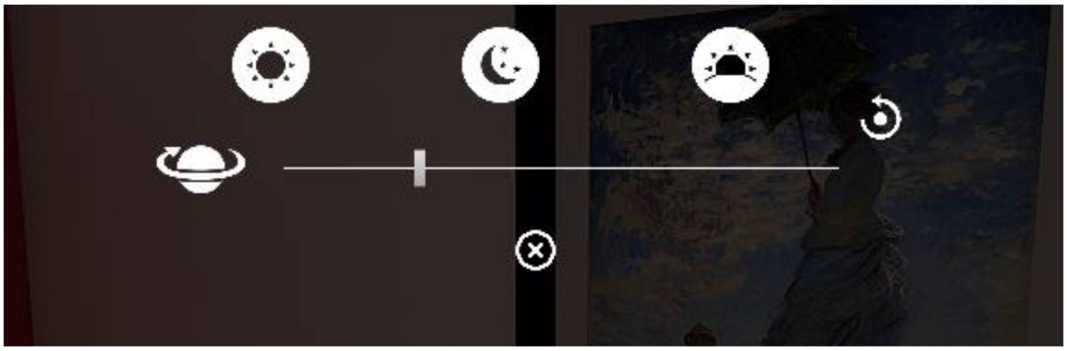

Рисунок 3.5 — Інтерфейс налаштувань часу доби та положення сонця.

За допомогою опцій "Увімкнути та вимкнути фонову музику" та "Увімкнути та вимкнути взаємодію з об'єктами" користувачі можуть налаштувати звуковий супровід та інтерактивність сцени згідно з власними перевагами.

3.1.2. Управління та взаємодія з об'єктами в конфігураторі

Для навігації по сцені використовуються клавіші W, S, A, D, що забезпечують плавний рух уперед, назад, вліво та вправо відповідно. Це дозволяє користувачам вільно переміщатися по сцені для детального огляду дизайну.

Взаємодія з об'єктами реалізована дуже просто і зрозуміло: користувачі можуть підходити до обраного елемента, навести на нього курсор і натиснути ліву кнопку миші (ЛКМ) для активації відповідного меню рис. 3.6. Наприклад, для зміни дивану користувач наводить курсор на диван і натискає ЛКМ, після чого з'являється меню, в якому можна вибрати інший тип дивану або змінити його матеріал.

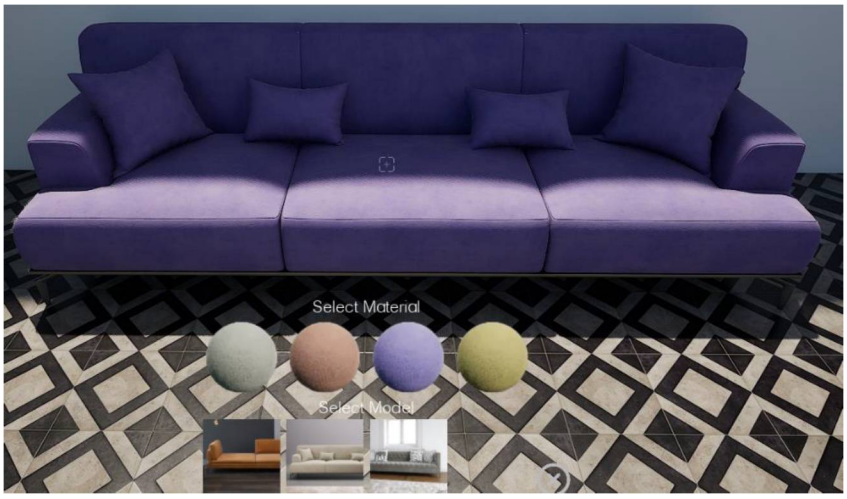

Рисунок 3.6 — Меню зміни об’єкта чи його матеріалу.

Аналогічний процес застосовується для зміни кольору стін: користувач вибирає стіну, натискає на неї ЛКМ і отримує доступ до палітри кольорів рис. 3.7.

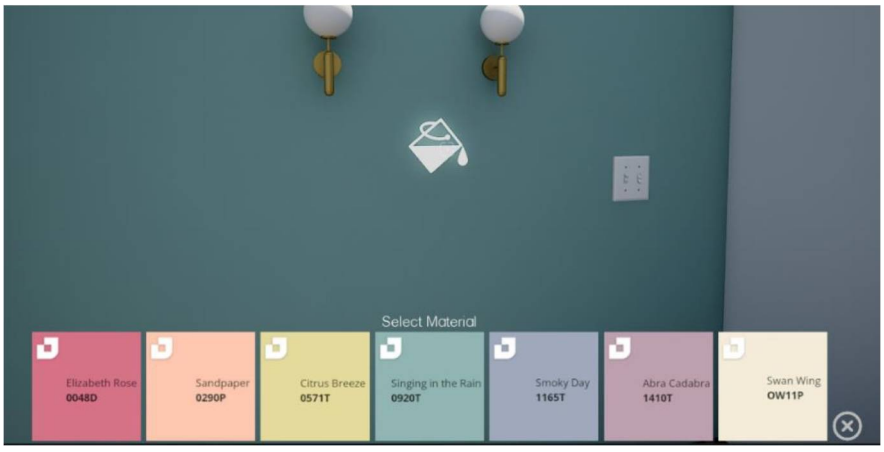

Рисунок 3.7 — Меню зміни кольору стіни.

Що стосується мультимедійних пристроїв, управління ними є простим і зрозумілим. Користувачі можуть активувати чи деактивувати мультимедійні пристрої, такі як телевізори чи аудіосистеми, простим натисканням на них у віртуальному середовищі. Це створює враження реального управління домашньою електронікою і додає рівень реалістичності до досвіду користувача.

Такий підхід до управління в конфігураторі забезпечує зручність і легкість взаємодії з елементами дизайну, дозволяючи користувачам ефективно модифікувати інтер'єр за своїми перевагами та потребами.

3.2 Налаштування конфігуратора

3.2.1 Налаштування актора BP_MasterSttings

У панелі деталей актора **MasterSettings Blueprint** можна встановити кілька значень, які контролюють рух гравця, фонову музику, перемикання рівнів, управління часом доби, послідовність рівнів та багато іншого рис. 3.8. Базові налаштування:

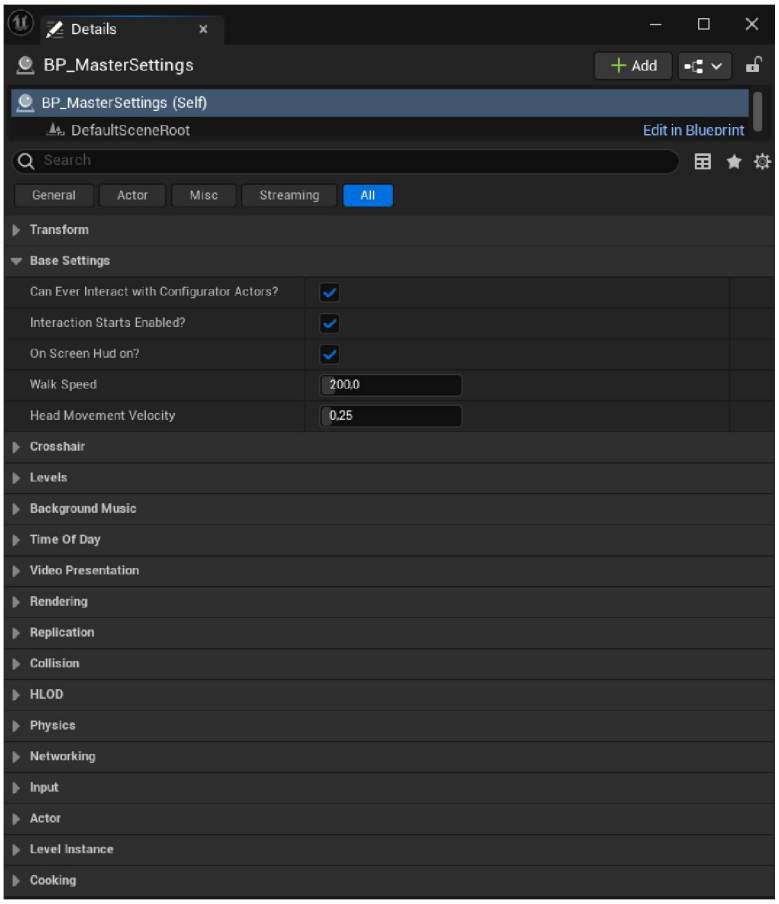

Рисунок 3.8 — Панель налаштувань актора MasterSettings Blueprint.

Can Ever Interact With Configurator Actors - Якщо ця опція увімкнена, клієнт зможе взаємодіяти з акторами конфігуратора у вашій сцені, такими як вимикачі світла, перемикання матеріалів меблів тощо. Якщо вона вимкнена, актори рівня не будуть інтерактивними. Рекомендується залишити цю опцію увімкненою.

Interaction Starts Enabled - Якщо ця опція увімкнена, клієнт почне гру з вимкненою взаємодією, і він зможе увімкнути її у меню користувача в грі.

On screen Hud on - Якщо ця опція увімкнена, в грі буде доступне меню користувача. Якщо вона вимкнена, меню користувача буде недоступним, але вам потрібно меню користувача для багатьох функцій ArchViz Configurator, тому рекомендується залишити цю опцію увімкненою.

Walk Speed - Швидкість руху головного персонажа гравця в грі.

Head Movement Velocity - Швидкість руху голови головного персонажа, коли ви рухаєте мишу.

Crosshair:

Crosshair on - Якщо ця опція увімкнена, в грі буде відображатися перевізник + значок під час гри для кращого прицілювання до об'єктів, з якими можна взаємодіяти.

Crosshair Size - Розмір перевізника, якщо він увімкнений.

Crosshair Opacity - Прозорість перевізника, якщо він увімкнений. Розділ "Рівні" контролює перемикання між різними рівнями в вашому проекті з меню користувача:

Level Names - Клацнувши на знак "+", ви можете додати рівні до списку для перемикання між ними з меню користувача під час гри. В полі "Added Index" введіть назву рівня точно так, як вона написана в вмістовому браузері.

Level Images - Клацнувши на знак "+", ви можете додати мініатюри, що представляють рівні в меню користувача. Зображення в індексі [0,1,2 і так далі] буде представляти назву рівня в індексі [0,1,2 і так далі] у вкладці "Назви рівнів". Рівні без мініатюр не відображатимуться в меню користувача, і ви не зможете перемкнутися на них. Вам потрібно мати точно таку ж кількість зображень, як кількість назв рівнів.

Image Size - Розмір мініатюр у меню користувача. Розділ "Фонова музика" контролює фонову музику рівня:

Background Music - Якщо ви бажаєте мати фонову музику для свого рівня, призначте тут ассет Music Wav.

Background Music Play at Start - Якщо ця опція увімкнена, фонова музика розпочне відтворюватися, як тільки гра розпочнеться. В іншому випадку гравець зможе розпочати відтворення музики з меню користувача під час гри.

Background Music Volume - Гучність фонової музики.

У розділі "Time of Day" вам потрібно призначити актор Directional Light, який представляє сонце у вашому рівні, це необхідно для керування часом доби та позицією сонця в меню користувача.

Sun - Тут вам потрібно призначити Directional Light, який використовується як сонце у вашій сцені. Якщо сонце не призначено, функціональність часу доби в меню користувача буде вимкнена.

У розділі "Video Presentation" вам потрібно призначити Level Sequencer або Master Sequencer, який ви хочете відтворити, коли користувач натисне на кнопку відео в меню користувача.

3.2.2 Налаштування актора BP_Universal

Актор **BP_Universal** є основним актором **ArchViz Configurator**. Всі інтерактивні об'єкти будуть акторами **BP_Universal** (окрім об'єктів, які можна підняти). Щоб зробити об'єкт інтерактивним у вашому рівні, просто перенесіть актора **BP_Universal** з браузера контенту > **BP_Universal** у вашу сцену та призначте налаштування в панелі деталей рис. 3.9.

Панель деталей актора **BP_Universal** - **Main** - У цьому розділі ви встановлюєте основні налаштування інтерактивного актора, такі як іконка взаємодії, колір іконки, тип об'єкта, як користувач буде взаємодіяти з актором та більше.

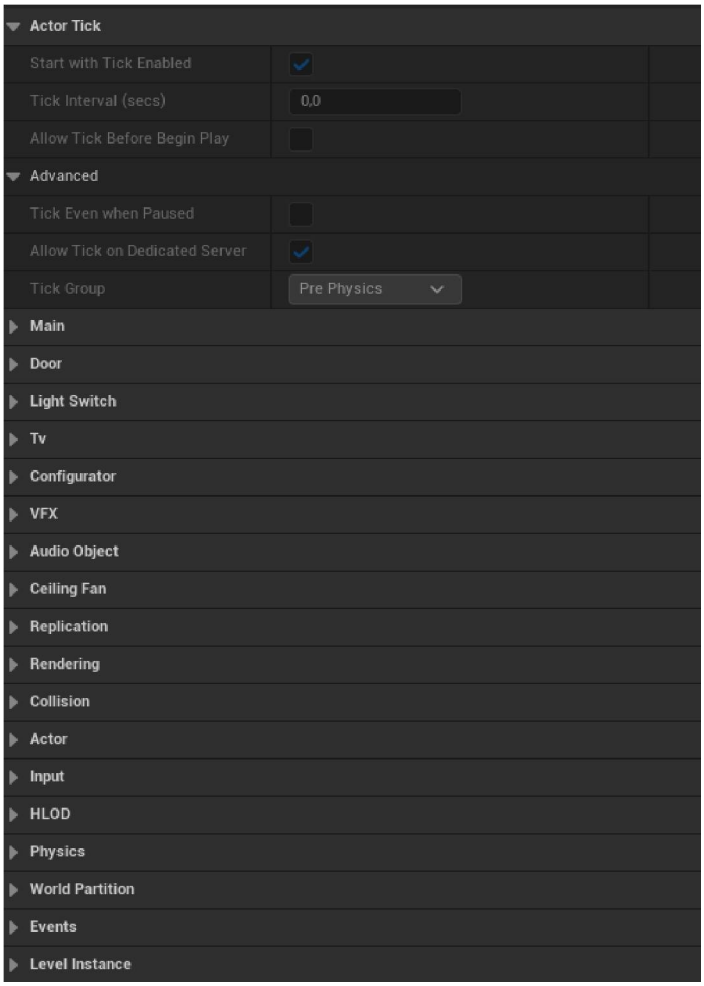

Рисунок 3.9 — Панель налаштувань актора BP_Universal.

Object Type - Тут ви обираєте тип об'єкта з наступних варіантів:

Door - Двері, шухляда, шафа або будь-що, що може рухатися вбік, назад\вперед або обертатися.

LightSwitch - Лампа, вимикач світла або будь-який світильник, на який ви хочете натиснути, щоб вмикати\вимикати світло.

CeilingFan - Вентилятор або будь-який об'єкт, який буде постійно обертатися на власній осі, наприклад обертові двері.

TV - Телевізор, відеостіна, проектор або будь-що, на що ви хочете відтворити медіафайл.

Configurator - Це будь-який об'єкт, який ви хочете, щоб користувач міг змінити, наприклад тип меблів, колір стін, матеріал підлоги, тип та матеріал килима або будь-що інше, про що ви можете подумати.

VFX - Будь-який тип візуального ефекту, який ви хочете, щоб користувач міг вмикати або вимикати, наприклад вогонь у каміні, вода з крана, вода з душової головки тощо (підтримує як стару систему частинок Cascade Unreal 4, так і нову систему частинок Niagara Unreal 5).

AudioObject - Музичний об'єкт, який ви хочете, щоб користувач міг увімкнути або вимкнути, наприклад грамофон, джукбокс, домашню медіасистему тощо.

LevelSequanceAnimated - Для більш складних типів об'єктів, якщо перші 7 варіантів недостатньо. Використовуючи цей варіант, ви можете підключити будь-яку рівневу послідовність з будь-якою анімацією в ній (наприклад, 3 розсувні двері, які відкриваються одночасно або ліфт, який рухається вгору та вниз, та багато іншого) і перемикати анімацію включено\виключено.

Interaction Type - Як користувач буде взаємодіяти з цим об'єктом, тут ми маємо 2 варіанти:

LookAt - Коли користувач дивиться на цей об'єкт, з'являється іконка, що повідомляє, що користувач може натиснути на об'єкт та розпочати взаємодію.

WalkTo - Коли користувач підходить до об'єкта, взаємодія починається, коли він відходить від об'єкта, взаємодія припиняється.

Outline Object - Ця опція керує, чи хочете ви, щоб навколо об'єкта взаємодії був контур, коли користувач на нього дивиться (буде працювати лише, якщо тип взаємодії встановлено як “LookAt”).

Icon Follow Model - Чи хочете ви, щоб іконка взаємодії слідувала за об'єктом взаємодії, коли об'єкт рухається, наприклад, якщо двері відкриваються, чи хочете ви, щоб іконка взаємодії дверей “прилипала” до дверей, коли вони рухаються, і рухалася разом з ними, чи хочете ви, щоб іконка залишалася на своєму початковому місці і не слідувала за дверима.

Icon Always Faces Player - Чи хочете ви, щоб іконка взаємодії завжди оберталася та дивилася на гравця або залишалася на місці та не оберталася (добре для таких речей, як об'єкти конфігуратора меблів, погано для таких речей, як

двері, коли ви хочете, щоб іконка залишалася на дверях та не оберталася всередині них).

Icon Color - Колір іконки взаємодії.

Icon Mask A - Вигляд іконки до взаємодії, наприклад, зображення закритих дверей. Зображення повинно бути чорно-білим без альфа-каналу та у квадратному форматі, білі частини представляють саму іконку, а чорні частини представляють те, що буде маскуватися та залишатися невидимим.

Icon Mask B - Вигляд іконки після взаємодії, наприклад, зображення відкритих дверей. Для об'єктів, які не рухаються або не змінюються, таких як конфігуратор меблів, встановіть однакову іконку для Icon A та Icon B.

Sound Fx 1 - Звуковий ефект, який буде відтворюватися при початку взаємодії, наприклад, скрип відкриваються дверей.

Sound Fx 2 - Звуковий ефект, який буде відтворюватися, коли взаємодія буде натиснута вдруге, наприклад, скрип закриваються дверей.

Якщо ви оберете тип об'єкта Configurator, вам потрібно налаштувати цей розділ.

3d Model - тут ми додамо 3D моделі, між якими ви хочете перемикатися під час взаємодії з цим актором під час гри.

Якщо ви хочете змінити тільки матеріал об'єкта, а не його сітку (Mesh), додайте лише одну 3D модель з такою ж сіткою, як у статичної моделі самого актора.

Щоб додати 3D модель, натисніть на іконку "+", потім з'являться 4 поля для заповнення:

3dModel - Тут ви вказуєте статичну модель, на яку ви хочете перейти.

MaterialElement - Тут ви вказуєте номер елемента матеріалу на статичній моделі, який представляє матеріал, який ви хочете змінювати під час гри.

3d Model Ui Icon - Іконка, яка буде відображатися в меню користувача при виборі цієї 3D моделі.

Materials - Щоб додати вибір перемикання матеріалів до цієї 3D моделі, натисніть іконку "+" і з'являться 2 поля для заповнення:

Material - Матеріал, який буде встановлено при виборі

Material Icon - Іконка, яка представлятиме матеріал в меню користувача.

Button Size - Розмір кнопок вибору в меню користувача.

Компоненти **BP_Universal** - **BP_Universal** має кілька компонентів, які нам потрібно буде використовувати та налаштовувати для різних типів об'єктів та типів взаємодії.

NameOfActor(Self) - це основний компонент актора, коли він вибраний, усі раніше обговорені налаштування з'являються у панелі деталей, тому якщо ви не бачите налаштувань, ймовірно, ви вибрали інший компонент. Важливо!!! Переконайтеся, що ви вибрали цей основний компонент при переміщенні цілого актора, інакше ви перемістите лише той компонент, який ви вибрали в той час.

CollisionArea - Коли ви використовуєте тип взаємодії "WalkTo", вам потрібно змінити розмір і перемістити цей компонент так, щоб він охоплював усю область, яка спричинить взаємодію, коли користувач увійде в неї. Цей компонент представлений як контур куба, який ви можете переміщати та змінювати розмір у сцені.

MediaSound - Коли ви обираєте тип об'єкта "TB", ви можете встановити область, де звук телевізора найгучніший і наскільки далеко від телевізора звук зменшується. Цей компонент представлений як 2 контури сфер (менша представляє внутрішній радіус, де гучність максимальна, і більша, після якої гучність буде 0). Щоб змінити внутрішній і зовнішній радіуси звуку TB, вам потрібно змінити ці значення в панелі деталей (коли вибрано компонент MediaSound).

InteractionIcon - Коли тип взаємодії встановлено як "Look at", ця іконка компонента з'явиться, коли ви дивитесь на актора, нам потрібно перемістити, обертати та змінювати розмір цієї іконки до місця, де ми хочемо, щоб вона з'являлася при погляді на актора.

StaticMesh - Це сама 3D модель, вам потрібно вказати статичну сітку актора тут у ВСІХ ВИПАДКАХ. Якщо це тип "Door", вам також потрібно вирівняти цей компонент так, щоб петля обертання дверей була вирівняна з

компонентом Gizmo. Якщо це тип "VFX" або "LevelSequencer", вам потрібно встановити цю статичну сітку як "Cube 1x1" (включено до Configurator) і змінити її розмір до бажаного, щоб захопити погляд гравця, і встановити її як невидиму, оскільки вона використовується лише для виявлення зіткнення погляду користувача.

Audio - Коли ви обираєте тип об'єкта "AudioObject", ви можете встановити область, де аудіо найгучніше, і наскільки далеко від актора звук зменшується. Цей компонент представлений як 2 контури сфер (менша представляє внутрішній радіус, де гучність максимальна, і більша, після якої гучність буде 0). Щоб змінити внутрішній і зовнішній радіуси аудіо, вам потрібно змінити ці значення в панелі деталей (коли вибрано компонент Audio).

Усі статичні сітки в **BP_Universal** повинні мати налаштоване зіткнення, інакше функція "Look At" не працюватиме.

Щоб взаємодіяти з об'єктами "Look At", дивіться на них з розумної відстані і коли з'являється їх іконка, натисніть ліву кнопку миші, щоб взаємодіяти.

3.3 Практичне використання конфігуратора для дизайну інтер'єру

Створення інтер'єру в конфігураторі є послідовним процесом, який включає наступні етапи:

1. **Вибір Сцени:** Початок роботи з конфігуратором передбачає вибір базової сцени або темплейту, який найкраще відповідає задуму користувача. Це може бути пуста кімната або вже частково оформлений простір.
2. **Навігація та Орієнтація:** За допомогою клавіш W, S, A, D користувачі можуть навігувати по сцені, вивчаючи простір та визначаючи потрібні зони для дизайну.
3. **Вибір та Розміщення Об'єктів:** Користувачі взаємодіють з об'єктами, натискаючи ЛКМ для активації меню вибору, де можна обрати тип меблів, аксесуарів, освітлення тощо.

4. **Кастомізація Об'єктів:** Після вибору об'єкта відкривається інтерфейс кастомізації, де можна змінити кольори, матеріали, текстури, а також додаткові параметри, як-от розмір чи форма об'єкта.
5. **Налаштування Освітлення:** Користувачі можуть змінити час доби та положення сонця для перегляду того, як природне світло впливає на інтер'єр, а також додати або змінити штучні джерела світла.
6. **Застосування Декору та Фінальних Акцентів:** Додавання декоративних елементів, таких як картини, рослини, килими, допомагає завершити образ інтер'єру та надати йому особистісного характеру.
7. **Перегляд та Редагування:** Після основного оформлення, користувачі можуть переглянути інтер'єр в режимі секвенцера або в режимі камери, що дозволяє оцінити загальний вигляд та зробити фінальні корективи.
8. **Збереження та Експорт Проекту:** Готовий проект може бути збережений для подальшої роботи або експортований у форматі зображень чи відео для презентації клієнтам або в портфоліо.

3.4 Оцінка та рекомендації для користувачів

3.4.1 Оцінка загальної ефективності конфігуратора.

Після детального аналізу можливостей та функціональності конфігуратора інтер'єру можна зробити висновок про його високу ефективність. Інструмент успішно справляється з завданнями візуалізації та кастомізації дизайну, надаючи користувачам гнучкі інструменти для реалізації творчих концепцій. Зручність навігації, інтуїтивно зрозуміле меню налаштувань, легкість взаємодії з об'єктами та широкий спектр опцій редагування роблять конфігуратор незамінним інструментом для дизайнерів інтер'єру та архітекторів.

3.4.2 Рекомендації для майбутнього використання.

На основі отриманого досвіду в роботі з конфігуратором, можна рекомендувати наступне:

- **Експериментуйте з налаштуваннями:** Не бійтеся пробувати різні комбінації матеріалів та елементів дизайну, щоб досягти оптимального результату.
- **Використовуйте функції збереження конфігурацій:** Регулярно зберігайте ваші робочі проекти, щоб мати можливість повертатися до попередніх версій.
- **Адаптуйте інтерфейс під свої потреби:** Налаштуйте інтерфейс користувача та графічні параметри відповідно до вимог вашого проекту та потужності обладнання.
- **Використовуйте режим камери для презентацій:** Створюйте високоякісні зображення та відео для демонстрації вашого проекту клієнтам або в портфоліо.
- **Отримуйте зворотний зв'язок:** Презентуйте ваші проекти колегам та клієнтам, використовуючи конфігуратор для збору відгуків та подальшого вдосконалення дизайну.
- **Будьте в курсі оновлень:** Слідкуйте за оновленнями програмного забезпечення, щоб використовувати всі нові можливості та вдосконалення конфігуратора.

3.4.3 Висновки

У третьому розділі було представлено рекомендації щодо практичного використання розробленого конфігуратора інтер'єру на основі технології Unreal Engine. Цей розділ включає введення в інтерфейс та основні функції конфігуратора, надаючи детальний огляд основного меню, інтерфейсу та управління об'єктами в конфігураторі. Окрім того, були описані кроки для створення інтер'єру з використанням конфігуратора, що демонструє практичне застосування інструменту в дизайні інтер'єру.

Висновки та рекомендації для користувачів підсумовують ефективність конфігуратора та його користувацьку цінність. Особливий акцент зроблено на можливості подальшого розвитку та удосконалення конфігуратора, зокрема

шляхом інтеграції штучного інтелекту, використання технологій віртуальної та доповненої реальності, впровадження колаборативних інструментів для роботи над проектами групами дизайнерів та розширення бібліотеки активів та матеріалів. Завершується розділ оцінкою загальної ефективності конфігуратора та рекомендаціями для його майбутнього використання, відкриваючи нові перспективи для індустрії інтер'єрного дизайну.

ВИСНОВКИ

Оцінюючи досягнуті результати в розробці конфігуратора інтер'єру на основі технології Unreal Engine, можна констатувати, що ключові цілі проекту були успішно втілені. Використання Unreal Engine дозволило створити високоякісні візуалізації, які разом з інтуїтивно зрозумілими інструментами налаштування надають користувачам велику гнучкість в моделюванні дизайну. Продвинутий режим камери, різноманітність параметрів налаштування інтер'єру, зокрема освітлення, меблів, декору, забезпечують ефективність і творчий простір для користувача. Оптимізація під різні системи забезпечує стабільну роботу програми на різних пристроях, що робить її доступною для широкого спектру користувачів.

Водночас, існує потенціал для подальшого розвитку та удосконалення конфігуратора. Інтеграція штучного інтелекту може автоматизувати вибір дизайну, а використання технологій віртуальної та доповненої реальності забезпечити більш занурювальний досвід. Колаборативні інструменти дозволять виконувати роботу над проектами групами дизайнерів, що є особливо важливим для професійних студій. Розширення бібліотеки активів та матеріалів відкриє додаткові можливості для кастомізації дизайнів. Нарешті, подальше удосконалення інтерфейсу користувача сприятиме збільшенню залученості та задоволення користувачів від процесу дизайну.

Таким чином, розроблений конфігуратор є потужним інструментом для створення та візуалізації дизайну інтер'єру, який вже на даному етапі демонструє високу ефективність та користувацьку цінність. У майбутньому, з урахуванням технологічних оновлень та зворотного зв'язку від користувачів, його можливості тільки розширюватимуться, відкриваючи нові перспективи для індустрії інтер'єрного дизайну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цзяньпен Цао, Девід Ф. Бухер, Даніель М. Холл, Єркер Лессінг. "Перехресний конфігуратор продукту для модульних будівель із використанням набору частин". URL: https://www.sciencedirect.com/science/article/pii/S0926580520310177. (дата звернення: 2 січня 2023 р).

2. Генрі Девід Лаут, Сезар Фрагачан, Вішу Бхушан, Шаджай Бхушан. "Архітектура та дизайн для промисловості 4.0". URL: https://link.springer.com/chapter/10.1007/978-3-031-36922-3_40. (Дата звернення: 2 січня 2023 р).

3. Сяомен Сун. "Зелений та екологічний інтер'єрний дизайн на основі мережевого процесора та вбудованої системи". URL: https://www.sciencedirect.com/science/article/abs/pii/S0141933121000909. (Дата звернення: 3 січня 2023 р).

4. Драган Перакович; Аннамарія Бегунова; Люція Кнапчікова. "Аналіз конфігураторів продуктів, що використовуються у масовому виробництві на замовлення". URL: https://actalogistica.eu/issues/2020/III_2020_06_Perakovic_Behunova_Knapcikova.pdf с.43-60.

5. "Офіційний сайт документації Unity". URL: https://docs.unity3d.com/Manual/ Дата звернення: 5 січня 2023 р.

6. Алекс Окіта. "Вивчення програмування на C# з Unity 3D, друге видання". URL: https://www.taylorfrancis.com/books/mono/10.1201/9780429810251/learning-programming-unity-3d-second-edition-alex-okita. Дата звернення: 6 січня 2023 р.

7. "Офіційний сайт навчання Unity". URL: https://learn.unity.com/. (Дата звернення: 6 січня 2023 р).

8. "Офіційний сайт документації WebGl". URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API. Дата звернення: 7 січня 2023 р.

9. "Основи WebGl". URL: https://webglfundamentals.org/. (Дата звернення: 8 січня 2023 р).

10. "Api 3D-графіки низького рівня на основі OpenGL ES". URL: https://www.khronos.org/webgl/. Дата звернення: 8 січня 2023 р.

11. "Unreal Engine: що варто знати". URL: https://vokigames.com/ua/unreal-engine-scho-varto-znatu-novachku-pto-zanadto-skladne-pz-na-yakomu-stvoruutsya-skhedevru/

12. Ясін А М Ель-Ваджех, Пол В Хаттон, Ніколас Дж Лі. "Unreal Engine 5 та занурювальне хірургічне навчання: перенесення досягнень у технологіях геймінгу до програм навчання хірургічної симуляції з розширеною реальністю". URL: https://academic.oup.com/bjs/article/109/5/470/6523981?login=false. (Дата звернення: 2 вересня 2023 р).

13. "Офіційний сайт документації Unreal Engine". URL: https://docs.unrealengine.com/5.3/en-US/. (Дата звернення: 3 вересня 2023 р).

14. Бо Карн. "Unreal Engine 5 Crash Course with Blueprint". URL: https://www.freecodecamp.org/news/unreal-engine-5-crash-course-with-blueprint/. (Дата звернення: 5 вересня 2023 р).

15. "Розкриття потенціалу Unreal Engine 5. Реальне застосування (туторіал для дизайнерів)". URL: https://ux.pub/sergeykoznov/rozkrittia-potientsialu-unreal-engine-5-riealnie-zastosuvannia-tutorial-dlia-dizainieriv-2g57 (Дата звернення: 6 вересня 2023 р).

16. Кевін Мак, Роберт Рууд. "Unreal Engine 4 Virtual Reality". с.55-64.

17. Джон П. Доран, Вільям Шеріф, Стівен Віттл. "Unreal Engine 4.x Scripting with C++". с.80-135.

18. "Сайт Unreal Engine. Programming with C++". URL: https://docs.unrealengine.com/5.3/en-US/programming-with-cplusplus-in-unreal-engine/. (Дата звернення: 7 вересня 2023 р).

19. Маркос Ромеро, Бренден Сьюелл. "Blueprints Visual Scripting for Unreal Engine 5". с.15-67.

20. "Unreal Engine 5 Blueprints Tutorial". URL: https://www.kodeco.com/36212581-unreal-engine-5-blueprints-tutorial. (Дата звернення: 8 вересня 2023 р).

21. Люк Герстхофер. "Просторово-часове фільтрування для реального трасування шляхів у віртуальній реальності". URL: https://publik.tuwien.ac.at/files/publik_292090.pdf. Дата звернення: 8 вересня 2023 р.

22. Роберт Рууд, Кевін Мак. "Unreal Engine 4". с.25-49.

23. "Методи 3D моделювання обладнання для планування розміщення". URL: https://www.vistable.com/blog/tools/5-methods-for-3d-modeling-of-equipment-for-layout-planning/. (Дата звернення: 9 вересня 2023 р).

24. "Box modeling". URL: https://it-s.com/what-is-box-modeling/. (Дата звернення: 9 вересня 2023 р).

25. "Полігональне моделювання". URL: https://devopedia.org/polygonal-modelling. (Дата звернення: 9 вересня 2023 р).

26. "Що таке NURBS криві у 3D моделюванні". URL: https://www.makeuseof.com/what-are-nurbs-curves/. (Дата звернення: 9 вересня 2023 р).

27. "Що таке 3D цифрове скульптурування?" URL: https://conceptartempire.com/what-is-3d-sculpting/. (Дата звернення: 10 вересня 2023 р).

28. "Що таке фотограмметрія?" URL: https://blogs.nvidia.com/blog/what-is-photogrammetry/. (Дата звернення: 10 вересня 2023 р).

29. "Етапи створення 3D моделі". URL: https://cghero.com/articles/stages-of-creating-3d-model. (Дата звернення: 10 вересня 2023 р).

30. "Еволюція 3D моделювання в архітектурі та дизайні: 30-річний ретроспективний огляд". URL: https://archicgi.com/cgi-news/evolution-of-3d-modeling-retrospective/. (Дата звернення: 11 вересня 2023 р).

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 «Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Титаренко Олег Вячеславович

1. Тема роботи «Конфігуратор розміщення елементів інтер'єру будівлі на основі технології Unreal Engine».

керівник роботи Толстолузька Олена Геннадіївна, д.т.н., с.н.с.

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

2. Перелік питань, які потрібно розробити)
1) Обгрунтування вибору середовища розробки моделі.
2) Огляд та аналіз принципів створення конфігуратора на основі технології Unrela Engine.
3) Розробка вимог до 3D-моделей.
4) Розробка алгоритмів функціонування конфігуратора.
5) Розробка програмної реалізації моделі та інтерфейсу користувача на основі технології Unreal Engine.
6) Перевірка на працездатність та оцінка результатів.
7) Підготовка та оформлення пояснювальної записки.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1)	Аналіз можливостей Unreal Engine.	08.12.2022р
2)	Огляд існуючих конфігураторів та їх реалізації.	01.01.2023р
3)	Визначення технічних вимог до конфігуратора.	10.02.2023р
4)	Проектування архітектури системи.	15.03.2023р
5)	Розробка необхідних 3D моделей.	09.04.2023р
6)	Реалізація логіки розміщення об'єктів.	10.05.2023р
7)	Інтеграція інтерфейсу користувача.	11.06.2023р
8)	Тестування базової функціональності.	20.07.2023р
9)	Оптимізація та вдосконалення.	20.08.2023р
10)	Підготовка та розробка програмної документації.	21.09.2023р
11)	Розробка та оформлення пояснювальної записки	28.11.2023р

5. Дата видачі завдання 08.12.2022

Студент О.В Титаренко _____
ініціали, прізвище підпис

Керівник роботи О.Г Толстолузька _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання

на розробку програмного виробу «Конфігуратор розміщення елементів інтерєру будівлі на основі технології UNREAL ENGINE»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу конфігуратор розміщення елементів інтер'єру будівлі на основі технології Unreal engine 1.2. Галузь застосування комп'ютерні технології
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія 2.2. Завдання на кваліфікаційну роботу магістра № 4101-5/3197 від «11» 2023 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета розробки програмного виробу створення інтуїтивно зрозумілого конфігуратора інтер'єру на базі Unreal Engine для оптимізації процесу дизайну, надання широких можливостей візуалізації та ефективного моделювання інтер'єрних просторів. 3.2. Призначення програмного виробу є надання дизайнерам інтер'єру інструменту для детальної візуалізації, кастомізації та ефективного планування простору, з використанням передових можливостей Unreal Engine для створення реалістичних та динамічних 3D-моделей інтер'єру 3.3. Вихідні дані для розробки конфігуратора інтер'єру включають технічне завдання, аналіз ринку, вимоги цільової

	аудиторії, ресурси Unreal Engine, дизайнерські стандарти, графічні матеріали та технічні специфікації обладнання.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик немає 4.2. Вимоги до надійності передбачають стабільну та безперебійну роботу, мінімальний ризик збоїв чи помилок під час експлуатації, а також забезпечення точності та актуальності обробки даних. 4.3. Вимоги до умов експлуатації немає 4.4. Вимоги до складу і параметрів технічних засобів для даного програмного виробу включають наявність сучасного комп'ютерного обладнання з достатньою обчислювальною потужністю, включаючи високопродуктивний центральний процесор, графічний процесор, що підтримує рендеринг високої якості, достатній обсяг оперативної пам'яті та місця на жорсткому диску, а також сумісність з останніми версіями операційних систем. 4.5. Вимоги до інформаційної та програмної сумісності немає 4.6. Вимоги до маркування та упаковки немає 4.7. Вимоги до транспортування і зберігання немає 4.8. Спеціальні вимоги немає
5. Вимоги до Програмної документації.	Програмною документацією до виробу «конфігуратор розміщення елементів інтер'єру будівлі на основі технології Unreal engine» вважати: 1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленого

<table>
<tr><td></td><td colspan="2">програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).
3) Опис програмного виробу (представити в розділі 2 пояснювальної записки до кваліфікаційної роботи).
4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">1) орієнтовна оцінка ефективності виконуваної роботи вказує на значне підвищення продуктивності та якості процесу дизайну інтер'єру.
2) Терміни розробки програмного виробу та витрати грошових коштів залежать від обсягу та складності проекту, але загалом вимагають значних інвестицій часу та ресурсів.
3) Порівняно з вітчизняними та зарубіжними аналогами, розроблений програмний виріб демонструє вищу адаптивність та функціональність, що може забезпечити конкурентні переваги на ринку.</td></tr>
<tr><td rowspan="4">7. Стадії і етапи розробки</td><td>Назви етапів роботи</td><td>Термін виконання роботи</td></tr>
<tr><td>1. Аналіз можливостей Unreal Engine.</td><td>08.12.2022р</td></tr>
<tr><td>2. Огляд існуючих конфігураторів та їх реалізації.</td><td>01.01.2023р</td></tr>
<tr><td>3. Визначення технічних вимог до конфігуратора.</td><td>10.02.2023р</td></tr>
</table>

	4. Проектування архітектури системи.	15.03.2023р
	5. Розробка необхідних 3D моделей.	09.04.2023р
	6. Реалізація логіки розміщення об'єктів.	10.05.2023р
	7. Інтеграція інтерфейсу користувача.	11.06.2023р
	8. Тестування базової функціональності.	20.07.2023р
	9. Оптимізація та вдосконалення.	20.08.2023р
	10. Підготовка та розробка програмної документації.	21.09.2023р
	11. Розробка та оформлення пояснювальної записки	28.11.2023р
8. Порядок контролю і приймання	В даному розділі повинні бути вказані загальні вимоги до приймання розробленого програмного виробу наприклад: 1) Перевірка ходу розробки програмного виробу. Керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії.	

	4) Пояснювальну записку надати на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.

Виконавець
студент групи КІ- 61
Титаренко О.В.

Замовник
д. т. н.,
Толстолузька О. Г.

Додаток В

Програма і методика випробувань

програмного виробу

«КОНФІГУРАТОР РОЗМІЩЕННЯ ЕЛЕМЕНТІВ ІНТЕР'ЄРУ БУДІВЛІ НА ОСНОВІ ТЕХНОЛОГІЇ UNREAL ENGINE»

1 Об'єкт випробувань

1.1 Найменування випробуваного програмного виробу: конфігуратор розміщення елементів інтер'єру будівлі на основі технології Unreal Engine.

1.2 Область його застосування: комп'ютерна графіка, дизайн, архітектура.

2. Мета випробувань

Підтвердження функціональних та інших характеристик розробленого програмного виробу вимогам, які сформульовані в ТЗ (Додаток Б).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2 Місце і тривалість випробувань

Приймальні випробування відбувалися в реальній дизайн-студії, де дизайнери використовували цей інструмент у своїй повсякденній роботі.

3.3 Обсяг випробувань

Приймальні (приймально-здавальні) випробування програмного виробу проводяться в обсязі відповідному цієї Програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Випробування програмного продукту здійснювались спеціалістами з комп'ютерної графіки в студії комп'ютерної візуалізації, які оцінювали його здатність до ефективного планування та візуалізації інтер'єрних елементів, забезпечуючи відповідність до вимог реалізму та інтерактивності.

4. Вимоги до програми або програмного виробу

Вимоги до програми вашої дипломної роботи включають здатність до ефективної навігації і керівництва користувача через процес конфігурації, залучення уваги з використанням інтерактивних елементів, інтуїтивну та зручну ергономіку інтерфейсу, естетичний дизайн, який відображає ідентичність бренду, позитивний емоційний вплив, високу сприйняту цінність і корисність, стабільність роботи.

5. Вимоги до програмної документації

Інформація запозичуються з відповідного розділу технічного завдання. «Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку програмного виробу (представлено в Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Ця Програма і методика випробувань розробленого програмного виробу (представлена в Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представлено в розділі 3 пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для випробувань програмного продукту з конфігурації інтер'єру потрібен комп'ютер з високою обчислювальною потужністю та стабільним інтернет-з'єднанням для забезпечення плавності роботи і візуалізації.

6.2 Порядок проведення випробувань

На першому етапі випробувань програмного продукту:

1. **Перевірка комплектності програмної документації:** Було здійснено ретельну оцінку наявності всіх документів, що вимагаються згідно з технічним завданням, для забезпечення повної комплектності та відповідності документації.

2. **Оцінка комплектності технічних і програмних засобів:** Перевірено, що всі необхідні технічні та програмні компоненти є на місці і правильно налаштовані, готові до подальших етапів випробувань.

3. **Розробка методики проведення перевірок:** Встановлено чіткі процедури та критерії для проведення перевірок на цьому етапі, щоб забезпечити ефективну та систематичну оцінку програмного продукту перед його більш детальним тестуванням на другому етапі.

На другому етапі випробувань програмного продукту:

1. **Успішне проходження всіх перевірок**: Кожен аспект і компонент програми був детально перевірений, забезпечуючи повну відповідність усім заданим критеріям та стандартам.

2. **Відповідність технічним характеристикам**: Програма була тестована для відповідності всім технічним вимогам, описаним у технічному завданні. Це включало перевірку продуктивності, сумісності, інтерфейсу користувача та інших ключових функцій.

3. **Стабільна та ефективна робота функціоналу**: Всі основні та додаткові функції програми були перевірені на предмет стабільності. Тестування показало, що програма ефективно виконує свої функції, забезпечуючи надійність і стабільність під час використання.

Виконавець  Титаренко О.В.