МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____ 2022 г.

ВИДЕОРЕДАКТОР С ИНТЕГРАЦИЕЙ 3D-МОДЕЛЕЙ В ВИДЕО

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.252 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ В. А. Парасич
«___»_____ 2022 г.

Автор работы,
студент группы КЭ-406
_____ С. П. Соловьев
«___»_____ 2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М. А. Алтухова
«___»_____ 2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___» _____2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Соловьеву Сергею Павловичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Видеоредактор с интеграцией 3D-моделей в видео» утверждена приказом по университету от 12 декабря 2021 г. №308/141

2. Срок сдачи студентом законченной работы: 1 июня 2022 г.

3. Исходные данные к работе:

Обеспечить следующий основной функционал приложения:

– возможность создания собственных 3D-моделей;

– создание паттерна поведения для каждой 3D-модели;

– реализация естественного взаимодействия моделей с объектами видеоряда;

– возможность модификации существующих 3D-моделей;

– создание инструментов для визуализации естественного освещения для добавленных объектов;

– возможность подбора качества текстур для 3D-моделей в зависимости от разрешения видео;

– возможность модификации видеоряда визуальными эффектами.

Среда и средства реализации – по выбору автора.

4. Перечень подлежащих разработке вопросов:

– анализ аналогов разрабатываемого приложения;

– детализация требований к приложению;

– выбор среды и средств реализации;

– проектирование архитектуры приложения;

– организация базы данных для базовых 3D-моделей;

– тестирование разработанного программного обеспечения.

5. Дата выдачи задания: 1 декабря 2021 г.

Руководитель работы _____/ *В. А. Парасич* /

Студент _____/ *С. П. Соловьев* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	07.02.2022	
Анализ аналогов разрабатываемого приложения	07.03.2022	
Детализация требований к приложению	09.03.2022	
Выбор среды и средств реализации	04.04.2022	
Проектирование архитектуры приложения	11.04.2022	
Организация базы данных для базовых 3D-моделей	18.04.2022	
Тестирование, отладка, эксперименты	10.05.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/ *В. А. Парасич* /

Студент _____/ *С. П. Соловьев* /

АННОТАЦИЯ

С. П. Соловьев. Видеоредактор с интеграцией 3D-моделей в видео. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 72 с., 38 ил., библиогр. список – 15 наим.

В рамках выпускной квалификационной работы разрабатывается десктопное приложение для добавления 3D-моделей с естественным взаимодействием с окружающей средой в видеоряд. Система учитывает недостатки аналогичных приложений и является их улучшенной версией с более обширным функционалом построения и интеграции 3D-моделей в видео.

Настоящий документ включает в себя обзор аналогов, технические и функциональные требования, реализацию и результаты тестирования десктопного приложения для добавления 3D-моделей с естественным взаимодействием с окружающей средой в видеоряд.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ПК – персональный компьютер;

ПО – программное обеспечение;

3D – трехмерное пространство;

ГБ – гигабайт;

ОЗУ – оперативное запоминающее устройство;

ОС – операционная система;

IDE – интегрированная среда разработки;

UE – Unreal Engine;

C++ – компилируемый типизированный язык программирования общего назначения;

Мбит/с – мегабит в секунду.

ВВЕДЕНИЕ

Технические характеристики персональных устройств пользователей не стоят на одном месте. Так, в 2013 году видеоигра Crysis 3 не имела возможности запуститься на максимальных настройках графики. Но для современных ПК подобное не вызывает никаких проблем. И с увеличением мощностей компьютеров растут и запросы пользователей.

На сегодняшний момент видеоигра или фильм, в представлении обывателя, должны обладать высоким качеством картинки. И если с разработкой новых игр на более современном игровом движке для удовлетворения запросов вопросов не возникает, то с видеоредакторами дело обстоит иначе.

Наложение визуальных эффектов уже недостаточно. В дело вступает Motion Capture – метод анимации персонажей и объектов, при котором анимация создаётся не вручную, а путём оцифровки движений реального объекта и последующего переноса их на трёхмерную модель, который позволяет встроить реалистичную 3D-модель в видеоряд. В таком случае картинка получается приятной глазу, не вызывает отторжения от встроенного объекта. Но, на мой взгляд, это лишь промежуточное решение.

При работе с системой Motion Capture 3D-модель создают в редакторе, прописывают детали в поведении отдельных участков тела, и в итоге человек, чьи движения улавливают в объектив для наложения, является лишь скелетом для будущего персонажа фильма или видео.

Есть и несколько более простое решение – добавить 3D-модель в видео при помощи встроенных во многие редакторы систем. Но в этом случае огромное внимание необходимо уделить каждой детали. Например, модель не должна «провалиться в картинку». Помимо прочего, изображение при тщательном просмотре вызывает некоторое отторжение неестественным

поведением, т.к. большинство видеоредакторов не обеспечивают пользователя полным функционалом для работы с моделями. Решение вышеуказанных проблем в одном приложении, посвященном интеграции 3D-моделей в видеоряд, – является целью моей выпускной квалификационной работы.

В одном видеоредакторе будут объединены функции создания 3D-моделей, задания скелетов будущих объектов и построение пространства взаимодействия с окружением видео. Так же должны быть реализованы база встроенных моделей и некоторых паттернов поведения, возможность изменения качества текстур, инструменты для добавления эффектов и теней.

Для достижения поставленной цели, необходимо решить следующие задачи:

– рассмотреть существующие видеоредакторы, имеющие функциональные решения для работы с 3D-моделями;

– провести детальный анализ современных программных технологий для разработки собственного видеоредактора;

– разработать архитектуру собственного программного комплекса на базе существующих аппаратных решений и представить конечную её реализацию с использованием ПО Unreal Engine;

– оценить работоспособность каждого элемента функционала разработанного программного комплекса (рабочее пространство для создания/редактирования 3D-моделей, база данных моделей и паттернов, инструментарий приложения, пространство работы с видеорядом);

– произвести выборку результатов и проанализировать работу системы на примере специальной задачи создания видео.

Основными критериями анализа современных программных технологий будут являться: поддержка работы с 3D-моделированием, возможность работы с текстурами высокого качества.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Один из самых распространенных пакетов программного обеспечения в области графических редакторов принадлежит компании Adobe System. В их зону компетенции так же входит создание и обработка видеоряда (Adobe Premier Pro [1]) и дополнение этого видеоряда различными эффектами, в том числе 3D-моделями (Adobe After Effects [2]).

Adobe Premier Pro – программа нелинейного монтажа. Данный продукт поддерживает высококачественное редактирование видео с 32-битовым цветом, разрешением 4Кх4К и выше, помимо этого имеется возможность работы со звуковыми дорожками.

Функционал Premier Pro в большей мере подходит для создания видеороликов, т.е. монтажа и сборки различных видеофрагментов в общий видеоряд. Но при этом также существует возможность добавить титры и произвести цветокоррекцию.

Adobe After Effects – программа для редактирования видео и динамических изображений. Как следует из названия, данный продукт применяется для наложения эффектов на уже существующий видеоряд. К функциональным возможностям так же можно отнести разработку композиций и анимации. Несмотря на вышесказанное, After Effects поддерживает и монтирование видео, но в таком случае трудозатраты на производство одного видеоролика сильно увеличиваются, нежели при использовании продуктов Adobe System в паре.

Разберем функционал Adobe After Effects подробнее. В продукте представлена обширная библиотека эффектов:

– эффекты моделирования;

– стилизованные эффекты;

– другие категории эффектов.

Эффекты моделирования – 18 имитационных эффектов. Они могут быть использованы для создания чего угодно – от дождя до волос. Все можно полностью настроить под себя. Когда вы соединяете их с другими эффектами, вы можете значительно расширить их функциональность.

Стилизованные эффекты**,** найденные в After Effects, полезны для создания видео слоев, которые иначе были бы невозможны. Этими эффектами drag-and-drop можно легко манипулировать, чтобы создать особенный внешний вид и стилистику. Одними из самых выдающихся эффектов являются – CC Glass и пастеризация.

Другие категории эффектов – десятки категорий, которые делают что-то другое. Например, существует целая категория эффектов, посвященная разным типам размытия. Есть и категория цветовой коррекции с эффектами, которые могут генерировать любой вид, который вы хотите передать. Также существует категория «Перспектива», которая включает в себя эффект трекера 3D-камеры, который позволяет легко отслеживать видеозапись.

Еще одним отличительным признаком After Effects являются шаблоны, которые позволяют при помощи заранее настроенных фильтров перенимать кадры из проектов других пользователей для своего собственного.

Помимо прочего в After Effects существует возможность интеграции в видео 3D-моделей. Для данной задачи существуют множество инструментов: 3D-слои, камера, функция 3D-композиции и маски. В возможности создания этих инструментов можно отнести фиксацию объекта к определенной позиции на видео, движение 3D-объектов по плоскости видеоряда, а так же создание самих 3D-моделей на основе технологии псевдо-3D.

К минусам программного продукта можно отнести следующее:

– зависимость от других продуктов Adobe System;

– сложность в освоении.

Из-за зависимости от других продуктов компании финансовые затраты на разработку видеоролика при помощи данного программного обеспечения выше.

Несмотря на внешнюю простоту в работе с After Effects, для большинства новичков в сфере графического дизайна на освоение потребуется много времени, т.к. не весь функционал интуитивно понятен.

Еще одним распространенным программным обеспечением для работы с видеорядом, а также для интеграции в него 3D-моделей, является Final Cut Pro компании Apple Inc [3].

Final Cut Pro – профессиональное программное обеспечение для нелинейного редактирования видео, базирующееся на базе macOS. Как 64-битное приложение использует 4 ГБ ОЗУ. Имеет поддержку OpenCL, что позволяет ускорить работу графического процессора при воспроизведении и рендеринге. Поддерживает размеры изображений от SD до 4K.

Помимо стандартного набора инструментов и функций видеоредактора Final Cut Pro имеет возможность добавления необходимого модуля от стороннего разработчика. Таким образом, любой пользователь может создать необходимый ему инструмент, а затем выложить на просторы сети интернет. После этого каждый сможет установить его себе, в случае надобности. Данный прием позволяет значительно расширить функционал программы.

Еще одним достоинством Final Cut является поддержка 360-градусной виртуальной реальности, которая включает в себя поддержку крупных производителей VR-гарнитур и 360-градусных камер. Такой инструмент позволяет добавить в видеоряд 3D-модели, а так же применить к видео эффекты размытия или свечения.

Рассмотрим минусы Final Cut Pro:

– базирование на базе macOS;

– отсутствие русификатора.

Как говорилось ранее, программа базируется на системе macOS. Это сокращает охват пользователей, поскольку данная система не является самой распространенной, а для ее использования требуется специализированная аппаратура.

Для большинства пользователей, проживающих в России, отсутствие русского языка является существенным минусом, который отталкивает их от использования данной программы.

Рассмотрим еще один пакет программного обеспечения для создания 3D-графики и анимации Cinema 4D от фирмы Maxon [4].

Cinema 4D — универсальная программа для 3D моделирования, редактирования объектов и создания эффектов. Также выполняет рендеринг объектов по методу Гуро. Распространяется на такие ОС как macOS, Windows, Linux.

К функционалу данного программного пакета можно отнести следующее:

– совмещение различных типов моделирования, что сильно упрощает создание 3D-моделей и анимации к ним;

– возможность добавления дополнительных модулей со специальными инструментами, что сильно расширяет функционал Cinema 4D;

– возможность загрузки заранее созданных сложных моделей и анимаций из шаблонов;

– функция глубокого освещения, позволяющая создавать правдоподобный свет на модели.

Программный пакет имеет понятный интерфейс, а активное сообщество пользователей, которое постоянно публикует инструкции и уроки для новичков, позволяет быстро разобраться в функционале и приступить к работе.

Но у Cinema 4D существует один серьезный минус. Данная программа является идеальной средой для 3D-моделирования, но не поддерживает полный функционал интеграции этих моделей в уже существующий видеоряд.

1.2 Анализ основных технологических решений

Для разработки в рамках выпускной квалификационной работы были выбраны игровой движок Unreal Engine 4 и расширяемая среда IDE Visual Studio Community.

Unreal Engine (UE) – это современный игровой движок на базе языка C++, разработкой и поддержкой которого занимается компания Epic Games. Возможности UE позволяют создавать и редактировать элементы 3D-анимации, спецэффекты в кинофильмах и играх, а также разрабатывать различные обучающие программы. Код приложения на этом движке работает на большинстве современных платформ и операционных систем (Android, iOS, Linux, MacOS, Microsoft Windows, PlayStation 4, PSP, Xbox One, PS Vita). Таким образом, одной из особенностей движка Unreal Engine является его универсальность.

Данный игровой движок обладает встроенным многофункциональным редактором UnrealEd [5], который обладает следующими функциями:

– пользователь с помощью редактора может осуществлять манипуляции со свойствами различных 3D-объектов, изменять их, программируя скриптовые сценарии;

– изменения, вносимые пользователем, такие как: расположение объектов, распределение эффектов, обновления текстур – видны через камеру редактора, что позволяет пользователю проверять возможные ошибки в построениях;

– редактор позволяет манипулировать объектами, созданными с помощью сторонних графических пакетов.

Анализируя функционал, можно понять, что работа с UnrealEd позволит нам создавать свои 3D-модели, на которые в дальнейшем будут наложены текстуры необходимого разрешения.

Так же в Unreal Engine встроен редактор BluePrint, который отвечает за создание логики поведения 3D-объекта. Данная функция дает возможность сделать модель подвижной, создать видимость естественности в видеоряде.

Достоинством UE на фоне других игровых движков является визуальное программирование, которое сильно упрощает разработку и отладку продукта.

Еще одним преимуществом разработки на Unreal Engine 4 является то, что с 02.03.2015 движок имеет бесплатную систему распространения, при условии того, что прибыль разработанного приложения не превышает $3000 за квартал [6].

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры, и приложения с графическим интерфейсом. IDE Visual Studio Community является адаптивной под любые нужды разработчика, из-за чего для разрабатываемого продукта базирование UE на ее основе – является оптимальным решением.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического

интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Visual Studio Community для написания программ на C++ тоже является полностью бесплатной программой, которую необходимо будет настроить на работу с Unreal Engine 4, используя инструкцию из официального руководства Epic Games [7].

1.3 Выводы

Разрабатываемое десктопное приложение для добавления 3D-моделей с естественным взаимодействием с окружающей средой в видеоряд должно избежать указанных ранее проблем уже существующих программных продуктов, а значит:

1) приложение должно быть автономным от других программных пакетов;

2) должно иметь интуитивно понятный интерфейс и необходимую сопровождающую документацию, дающую полную картину функционала программы;

3) интерфейс приложения должен быть выполнен на русском языке для большего удобства работы пользователей, на которых ориентирован продукт, а именно, жителей России;

4) приложение должно не только давать возможность работы с 3D-моделями, но и позволять интегрировать их в видеоряд.

Видеоредактор будет разработан на основе игрового движка Unreal Engine 4 с поддержкой Visual Studio Community IDE для реализации всех поставленных задач.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации данной системы необходим следующий набор подсистем:

– подсистема 3D-визуализации;

– база данных, обеспечивающая хранение данных о пользователях, 3D-моделях и паттернов их поведения;

– подсистемы работы с видеорядом (конвертор форматов, встроенный проигрыватель видеороликов, база инструментария);

– клиентское приложение под различные платформы;

– графический интерфейс клиентского приложения;

– графический интерфейс администратора.

Подсистема 3D-визуализации позволит использовать 3D-модели.

База данных обеспечит возможность хранения данных о пользователях, 3D-моделях и паттернов их поведения.

Подсистемы работы с видеорядом будут предоставлять функционал для работы непосредственно с видеофрагментами.

Клиентское приложение необходимо для организации выполнения вычислений за счет ресурсов пользователя.

Графический интерфейс клиентского приложения будет ответственен за настройку и контроль работы клиентского приложения, визуальное отображение регистрации в системе, получение обновлений системы видеоредактора, отправку обратной связи разработчику.

2.1 Функциональные требования

Система должна обеспечить возможность выполнения следующих функций:

1) позволить пользователю создавать свои 3D-модели/паттерны поведения;

2) предоставить возможность редактирования базы данных моделей (загрузка/удаление моделей/паттернов поведения);

3) дать пользователю функционал стандартного набора видеоредактора;

4) возможность сохранения видео в классическом формате (например, .mp4), а также в формате удобном для продолжения работы с установленными 3D-моделями в видеоряд (например, .psd для программы Adobe Photoshop);

5) предоставить инструмент создания плоскостей пространства для отображения границ объектов на видео, чтобы добавить естественности взаимодействия 3D-модели и окружения видео;

6) позволить пользователю создавать собственные инструменты для обработки видео;

7) дать пользователю возможность отправить отчет об ошибке разработчику;

8) предоставить пользователю возможность точной настройки движений 3D-модели;

9) предоставить инструмент для создания и настройки теней объектов;

10) предоставить пользователю возможность авторизации:

10.1) неавторизованный пользователь имеет возможность пользоваться полным функционалом приложения на протяжении 30 календарных дней;

10.2) авторизованный пользователь сможет использовать приложение неограниченно по времени и функционалу;

11) автоматическое определение количества источников света, их интенсивности и направленности, с возможностью ручной корректировки.

2.2 Нефункциональные требования

Рассмотрим минимальные требования к аппаратному обеспечению для корректной работы программного продукта, который будет написан на основе Unreal Engine 4:

– ОС: Windows 7/8/8.1/10 64 Bit;

– процессор: 4 ядра 2,5 GHz;

– оперативная память: 8 GB;

– видеокарта: с поддержкой DirectX 11.

Поскольку версии ОС Windows 7/8/8.1 уже не так популярны (а также теряют поддержку Microsoft), разрабатываемое приложение должно использоваться с ОС Windows 10.

Характеристики процессора, оперативной памяти и видеокарты примем из указанных выше минимальных требований.

Кроме функциональных требований, направленных на непосредственную реализацию предоставляемых видеоредактором функций, и аппаратных требований, необходимо учесть прочие требования, не относящиеся к предыдущим пунктам разработки:

1) итоговое видео не должно превышать емкость исходного видеоряда более чем на 60%, таким образом, при исходном значении в 8 ГБ (формат файла MP4 (1440x1080) длительностью 1:24:40) после обработки емкость будет составлять ≤ 12.8 ГБ;

2) предусмотреть поддержку 3D-моделей, созданных вне разрабатываемого приложения;

3) в разработке интерфейса придерживаться классической схемы, интуитивно понятной пользователю, то есть дизайн интерфейса не должен

кардинально отличаться от, например, Adobe Premiere Pro или Adobe After Effects;

4) предусмотреть ограничение функционала для неавторизованного пользователя по истечению 30 календарных дней с момента первого использования приложения.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

Разрабатываемое приложение будет построено на архитектуре Model, view, controller (MVC). На рисунке 1 представлена данная модель.

Рисунок 1 – Модель MVC

Пользователь использует контроллер, а именно загружает видео и 3D-модели, взаимодействует с иконками инструментов, управляет тем самым моделью (инструментами), изменяет представление (видео и 3D-модели), которое сам же может наблюдать.

Разрабатываемое в рамках выпускной квалификационной работы программное обеспечение состоит из следующих функциональных блоков:

1) авторизация пользователей;
2) работа с 3D-моделями;
3) работа с видео;
4) создание пользовательских инструментов;
5) рендер видео.

Функции, исполняемые каждым блоком, будут являться асинхронными, т.к. при обращении на один и тот же адрес 3D-модели блоками 2 и 3 пользователю невозможно внести изменения в параметры модели. Аналогичная ситуация и с блоком 4. Обращение к инструментам редактора возможно только с параметром только для чтения.

Для визуализации архитектуры предлагаемого решения использовалась программа Ramus Educational [8].

На рисунке 2 изображена контекстная диаграмма, отображающая все элементы конечного продукта. Входными данными служат исходное видео, т.е. видео до обработки, а также пользователи, которые будут работать с программой. Прибегая к инструментам редактора и руководству по эксплуатации, в итоге будет получено смонтированное видео.

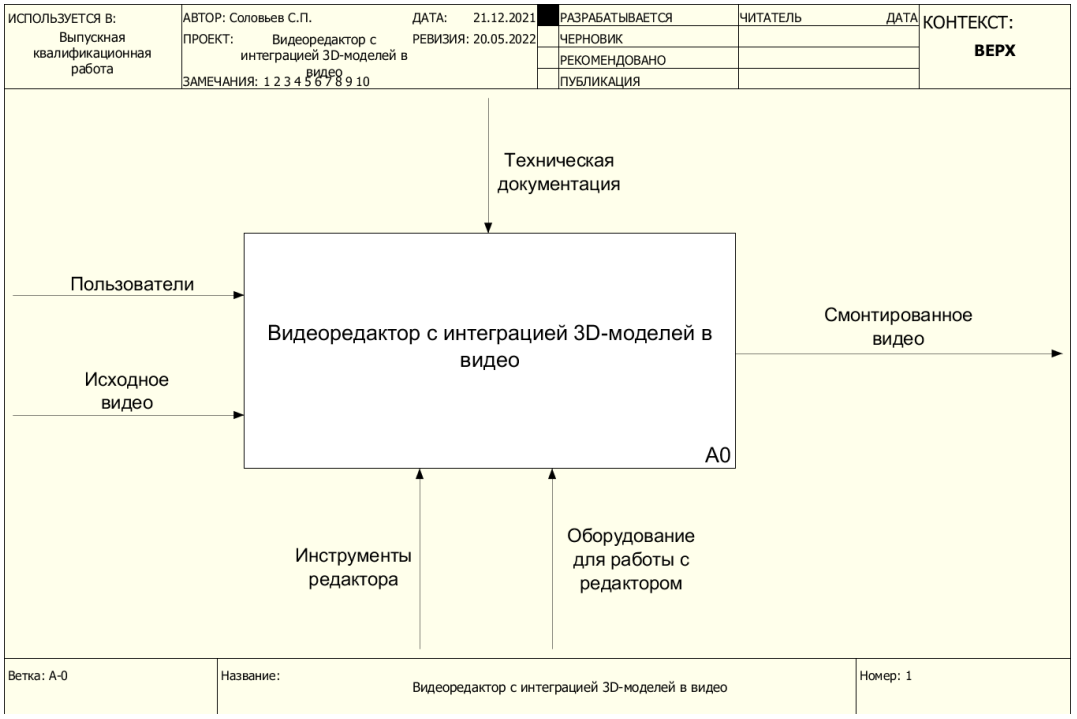

Рисунок 2 – Контекстная диаграмма

На рисунке 3 изображены основные блоки работы видеоредактора – блок авторизации пользователя, блок обработки видеоряда, и блок рендера. Рассмотрим каждый блок отдельно.

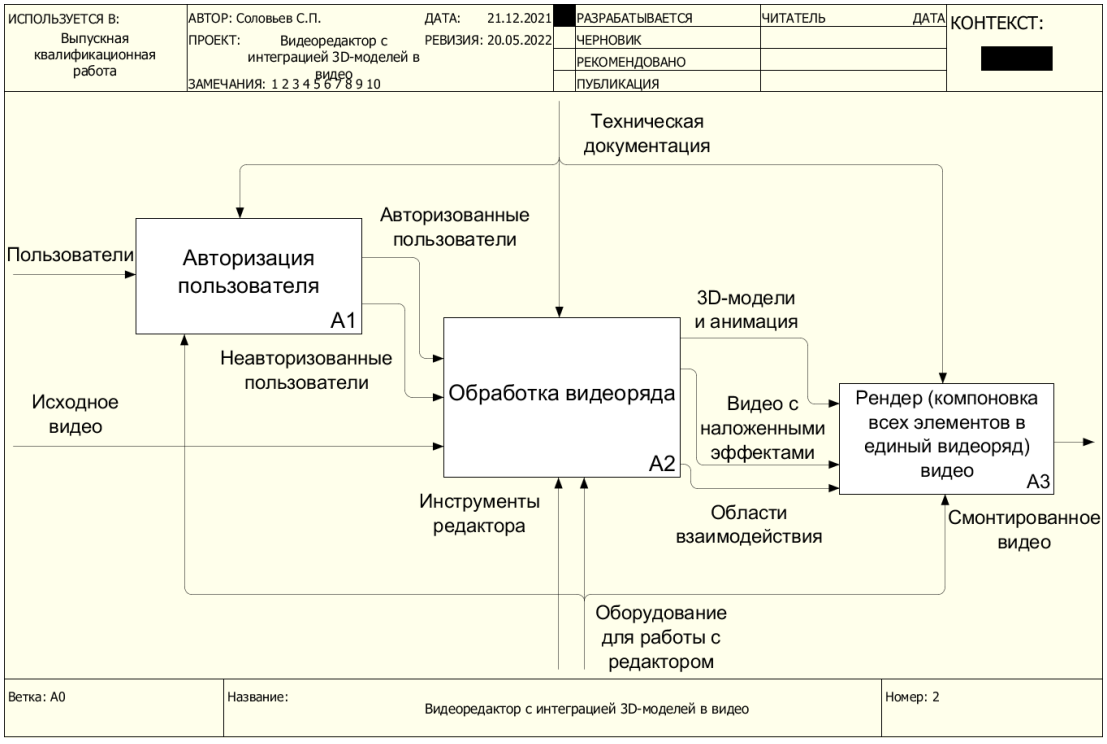

Рисунок 3 – Диаграмма декомпозиции

На рисунке 4 продемонстрирована декомпозиция блока авторизации. Пользователь вводит свой логин и пароль, после чего, полученные данные проверяются на правильность. При положительном исходе пользователю присваивается класс авторизованного, что позволяет использовать полный функционал редактора. Также имеется вариант работы пользователя в неавторизованном формате. В таком случае возможность беспрепятственной работы в видеоредакторе ограничивается 30 календарными днями.

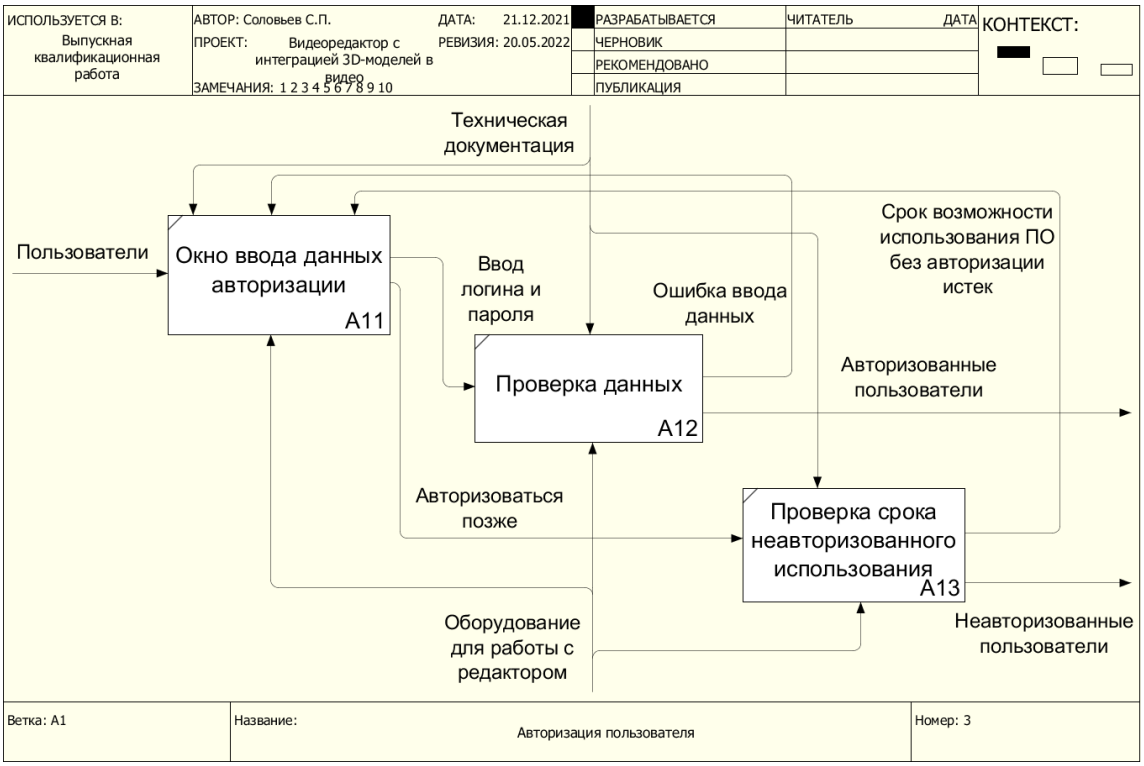

Рисунок 4 – Декомпозиция авторизации пользователя

На рисунке 5 изображена декомпозиция блока обработки видеоряда. Пользователь попадает в несколько взаимосвязанных окон приложения.

Первое окно – создание моделей, наложение на них текстур, добавление анимации, второе – работа с видеорядом (монтаж, эффекты, плоскости объема), третье окно – среда создания пользовательских инструментов, в котором можно настроить уже существующие и собрать собственные. На выходе из этого блока мы получаем – видеоряд с эффектами и структуру 3D-моделей и плоскостей на видео.

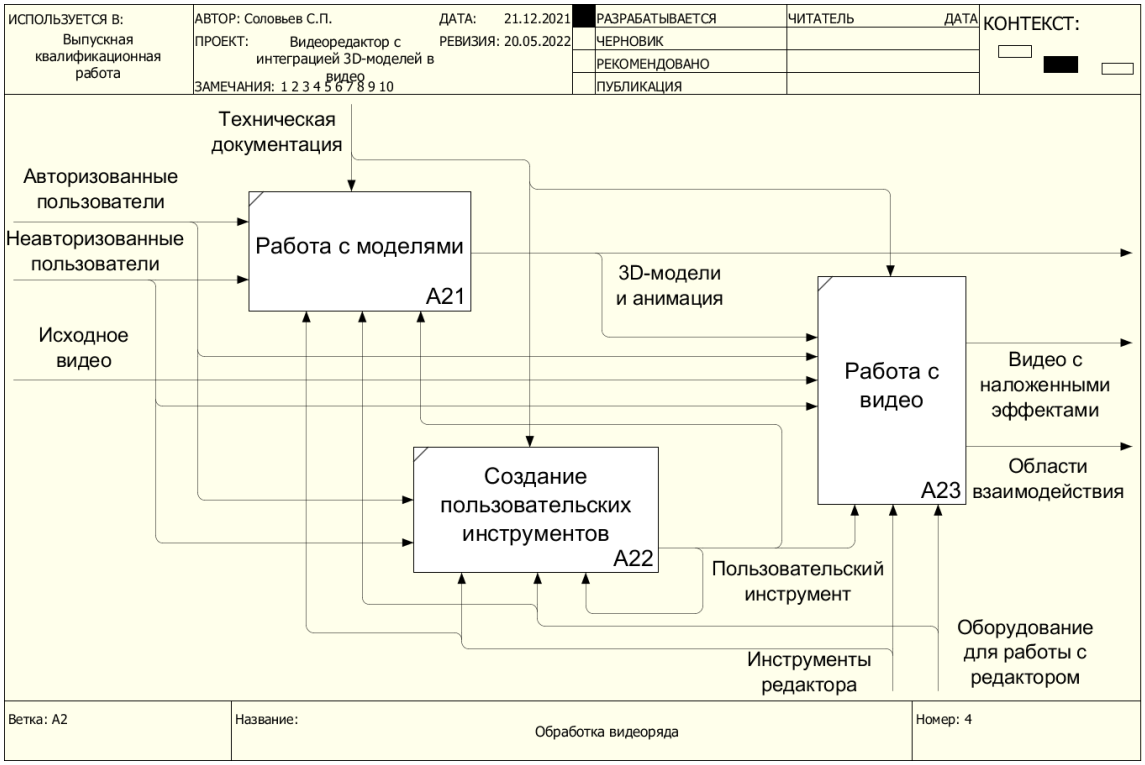

Рисунок 5 – Декомпозиция обработки видеоряда

На рисунке 6 изображена декомпозиция блока рендера видео. Здесь в два этапа видео собирается воедино и конвертируется в формат видео *.mp4. После всех вышеуказанных процедур пользователь получает смонтированное видео с интегрированными в видеоряд 3D-моделями.

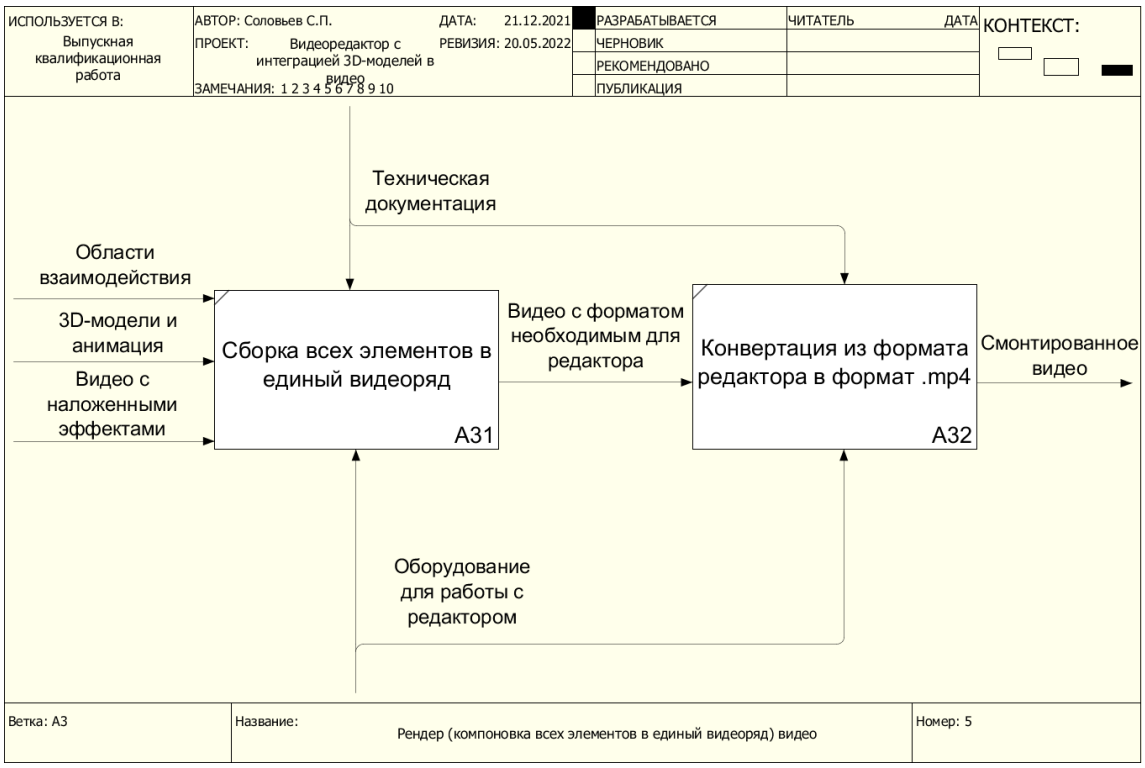

Рисунок 6 – Декомпозиция рендера видео

3.2 Описание данных

Функционал разрабатываемого приложения нацелен на работу с видеофайлами, 3D-моделями, текстурами для моделей и пользовательскими данными, такими как логин и пароль для авторизации в системе.

Исходя из вышеперечисленного, выделим следующие форматы данных:

– формат 3D-моделей;

– формат текстур 3D-моделей;

– формат видеофайлов;

– текстовый формат.

Рассмотрим каждый подробнее.

3.2.1 Формат 3D-моделей

Трехмерная модель состоит из множества точек, которые соединяются между собой гранями и образуют полигоны.

Вершина – это точка, которая имеет свои координаты в трехмерной системе, то есть X, Y, Z.

Грань, или ребро – отрезок, соединяющий две вершины. В трехмерной графике гранью называют ограничитель полигонов.

Основной составляющей в трехмерной графике считается полигон – плоский многоугольник, множество которых образует трехмерную фигуру. Любая фигура будет строиться из многочисленных простых фигур. Чем больше будет простых фигур в составе сложной, тем более гладкой будет казаться поверхность 3D-модели.

На рисунке 7 продемонстрирован принцип создания 3D-моделей на примере трехмерной фигуры шар. Каждая точка на шаре является вершиной, меридианы и параллели (линии между точками) – грани, или же ограничители полигонов, образуемые вершинами и гранями многоугольники – полигоны, форма которых градируется от треугольника (на полюсах шара) до квадрата (на экваторе).

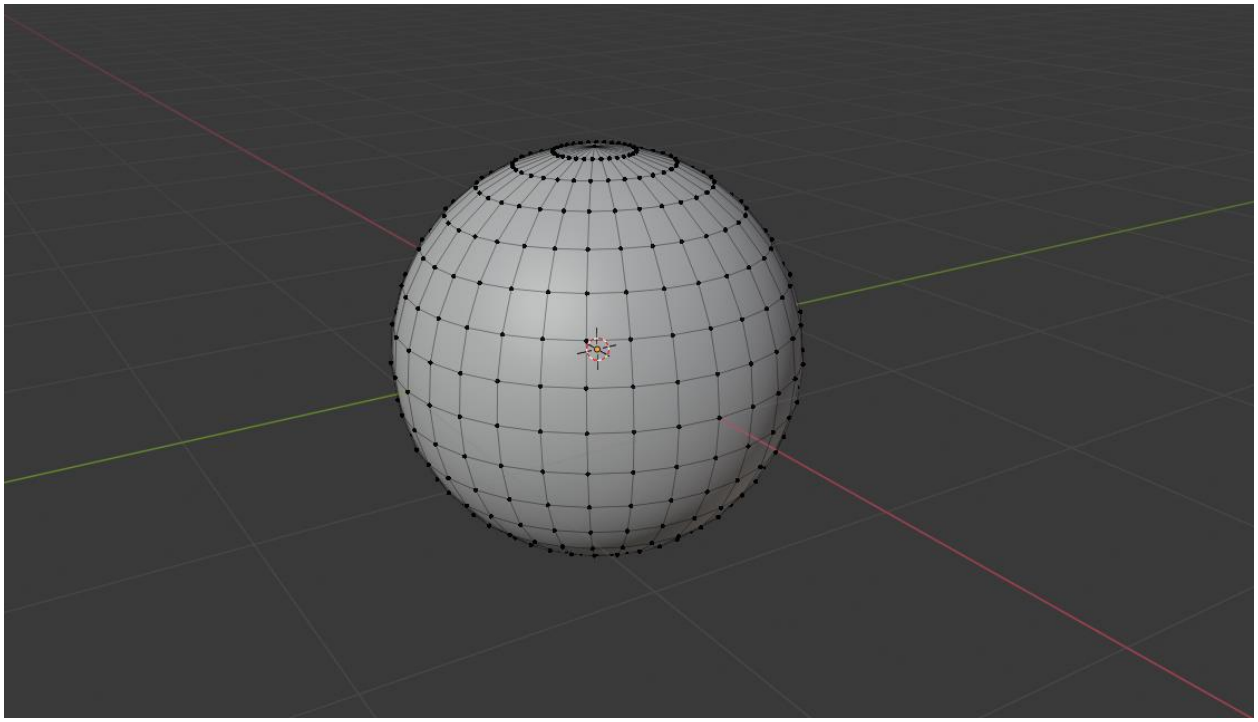

Рисунок 7 – Графическое отображение принципа создания 3D-моделей

В 3D-моделях для игр и видео предусмотрено внедрение коллизий. Коллизия – физическая модель объекта, определяющая его границы или придающая «осязаемости» 3D-модели.

Существует множество форматов для 3D-моделей [9]. Это связано с тем, что большинство разработчиков прибегает к созданию собственного формата, уникального и удобного в работе для разрабатываемого ими софта.

Для Unreal Engine оптимальным в работе форматом моделей является *.fbx [10].

Рассмотрим возможности данного формата и применяемые к нему требования для правильной работы с игровым движком.

FBX-файлы позволяют передавать 3D-объекты, источники света, текстуры моделей, 2D-объекты с параметром высоты, точки, сетки моделей и камеры между различными средами моделирования.

Для правильного переноса в среду Unreal Engine необходимо, чтобы 3D-модель соответствовала следующим требованиям:

– если модель имеет возможность анимации, то ей должен быть присвоен соответствующий каркас;

– опорная точка должна быть расположена в месте удобном для размещения модели;

– модели могут использовать перекрывающиеся текстурные координаты части для максимального использования текстур;

– модели должны быть приближены к масштабу 1 Unreal юнит = 1 сантиметр;

– все модели должны иметь материал применяемый к ним (текстуры);

– если визуальное качество требует использования групп сглаживания, необходимо включить их при экспорте из 3D пакета и при импорте в UE4;

– все модели должны быть ориентированы передом в сторону положительного направления оси X;

– для мобильных устройств модели должны быть ограничены числом вершин в 65535. На ПК ограничения накладываются в зависимости от аппаратной части;

– все модели должны иметь соответствующую им коллизию.

FBX является распространенным форматом, конвертация в который предусмотрена многими редакторами. В связи с этим, выбрав *.fbx для работы с 3D-моделями в разрабатываемом приложении, система не только будет работать с форматом оптимальным для Unreal Engine 4, но и удовлетворит условию 2 нефункционального требования.

3.2.2 Формат текстур 3D-моделей

Рассмотрев формат 3D-моделей, сделаем вывод относительно текстур моделей. Форматом FBX предусмотрены встроенные текстуры, т.е. при загрузке

3D-моделей из других продуктов моделирования нам нет необходимости накладывать свое изображение.

При этом при создании собственных моделей такая необходимость есть. Поэтому стоит рассмотреть, какие форматы статического изображения поддерживает среда разработки, для оптимизации работы приложения.

В рамках разработки на движке Unreal Engine 4 предусмотрены следующие форматы [11]:

– *.bmp – формат хранения растровых изображений, разработанный компанией Microsoft;

– *.pcx – стандарт представления графической информации, разработанный компанией ZSoft Corporation;

– *.png – растровый формат хранения графической информации, использующий сжатие без потерь по алгоритму Deflate [12];

– *.psd – растровый формат хранения изображений Adobe Photoshop, использующий сжатие без потерь, содержит слои с масками, цветовые пространства, ICC профили, прозрачность, текст, альфа-каналы и плашечные цвета, фигуры отсечения и двухцветные настройки

– *.tga – растровый графический формат, созданный компанией Truevision Inc для графических адаптеров;

– *.jpg – один из самых распространенных форматов хранения статической графической информации;

– *.exr – формат графических файлов для хранения изображений с широким динамическим диапазоном яркости;

– *.dds – формат хранения данных, разработанный корпорацией Майкрософт для использования в DirectX SDK [13];

– *.hdr – формат который содержит сверхвысокое разрешение и информацию о яркости и цвете для каждого пикселя.

Так же в Unreal Engine существуют следующие ограничения на используемые изображения:

– все текстуры должны иметь размеры, равные степени двойки (32, 64, 1024, 2048);

– для MIP-текстурирования текстура не должна превышать 4096 пикселей (4K);

– некоторые графические процессоры имеют аппаратное ограничение в 8192 пикселя (8K).

Такие форматы как *.jpeg и *.png являются одними из самых распространенных в программном обеспечении ориентированном как и мой продукт на ОС семейства Windows. Исходя из условия, что разрабатываемое ПО должно иметь возможность работы с другими редакторами 3D-моделей и видео, я остановился на данных форматах изображения для дальнейшей разработки.

3.2.3 Формат видеофайлов

Среда разработки Unreal Engine 4 позволяет создать приложения, способное работать с большинством существующих форматов видео [14].

Сравнивая применения и обработку тех или иных форматов видео, я остановился на *.avi и *.mp4.

Формат AVI обладает высоким качеством изображения на видео, но в то же время занимает очень много пространства памяти. В то время как MP4 имеет чуть более слабую прорисовку графики, но занимает в разы меньше места [15].

Моей задачей является интегрировать модель в видеоряд так, что бы она выглядела естественно на фоне окружения. И при этом существует условие, что итоговое видео не должно занимать большой объем памяти.

Сделав выводы из вышесказанного, я пришел к решению, что MP4 является оптимальным вариантом для работы с видеорядом в разрабатываемом приложении.

Рассмотрим параметры данного формата.

Хранилище MP4 поддерживает четыре формата данных:

1) видеопотоки;
2) аудиопотоки;
3) субтитры;
4) статичные изображения.

Видеопотоки кодируются в форматах MPEG-H Part 2 (H.265/HEVC), H.264/MPEG-4 Part 10 (AVC), MPEG-4 Part 2, H.262/MPEG-2 Part 2, H.261/MPEG-1 Part 2.

Аудиопотоки кодируются в форматах MPEG-4 Part 3, MPEG-2 Part 7, MPEG-1 Audio Layer III (MP3), MPEG-1 Audio Layer II, MPEG-1 Audio Layer I.

Субтитры – MPEG-4 Part 17.

Статичные изображения кодируются в форматах JPEG и PNG.

Кроме вышеперечисленных форматов данных в MP4 может храниться предусмотренная стандартом информация об авторе файла, дате создания, продолжительности потоков и языковой код потока. Иными словами, MP4 может содержать и метаданные.

MP4 осуществляет воспроизведение видеопотоков с размерами кадров следующих диапазонов:

– от 240p до 720p (HD);

– 1080p (Full HD);

– 4K (Ultra HD).

Для соответствия требованиям, из-за которых был выбран формат MP4, размер кадров ограничим диапазоном с верхней границей 1080p. Битрейт такого видео будет в пределах от 0.5 Мбит/с до 40 Мбит/с. Под битрейтом в свою очередь следует понимать степень сжатия потока и, следовательно, размер канала для передачи. Существует 2 основных подхода к распределению битов в потоке: постоянный (CBR) и переменный (VBR). Разница в поведении битрейта при двух разных подходах видна на графике, изображенном на рисунке 8. Во время трансляции видеопотока при CBR передается одинаковое количество бит за 1 секунду, а при VBR – количество бит может изменяться. Во время кодирования при постоянном битрейте наполнение и опустошение буфера происходит с одной и той же скоростью. При переменном – параметр скорости варьируется.

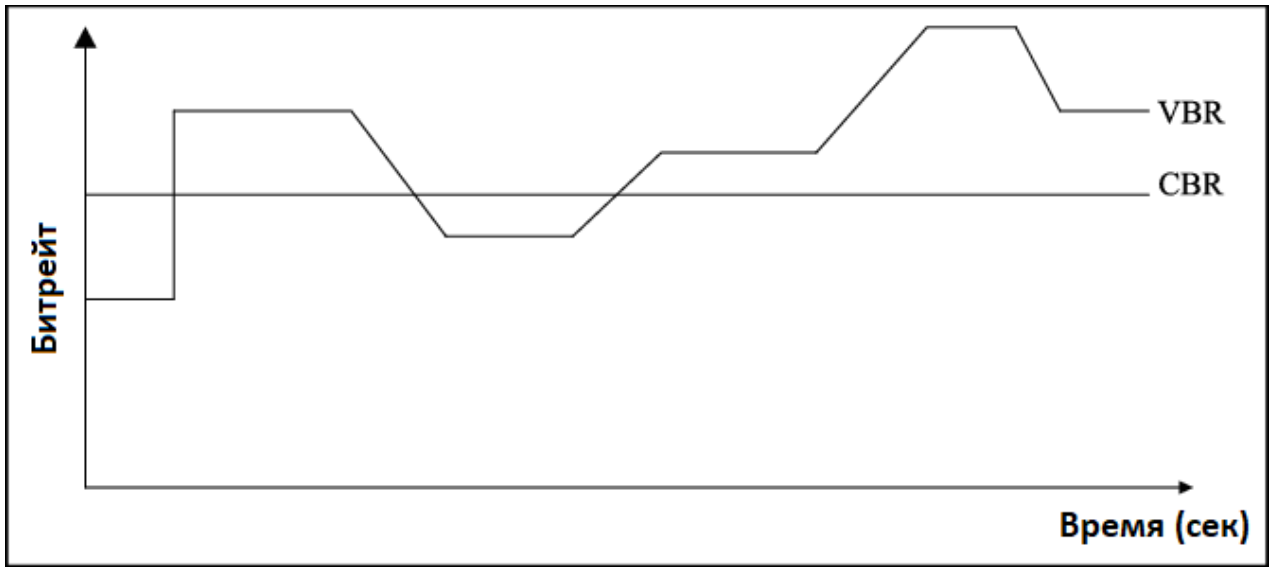

Рисунок 8 – График поведения битрейта в зависимости от времени при подходах VBR и CBR

Для реализации функций остановки, запуска, ускорения и замедления воспроизведения видеоряда будет применятся подход VBR. При кодировании частота записи должна быть постоянной, поэтому будет применяться подход распределения битов CBR.

4 РЕАЛИЗАЦИЯ

На основе структуры приложения, разработанной в главе 3.1, были реализованы следующие элементы:

– авторизация пользователей;

– главное меню, выполняющее функции открытия и перехода к блокам обработки видео, создания и редактирования 3D-моделей, создания и настройки пользовательских инструментов;

– окно работы с 3D-моделями;

– окно для обработки видео и применения инструментов 3D-моделирования;

– окно создания и настройки пользовательских инструментов.

Элементы выполнены в виде уровней. Каждый уровень является отдельным исполняемым файлом, в тело которого можно расположить как функции, ответственные за пользовательский интерфейс, так и объекты классов, отображаемые на сцене. Под сценой следует понимать рабочее пространство, где происходит любая визуализация. На рисунке 9 представлены имеющиеся в приложении уровни и их названия.

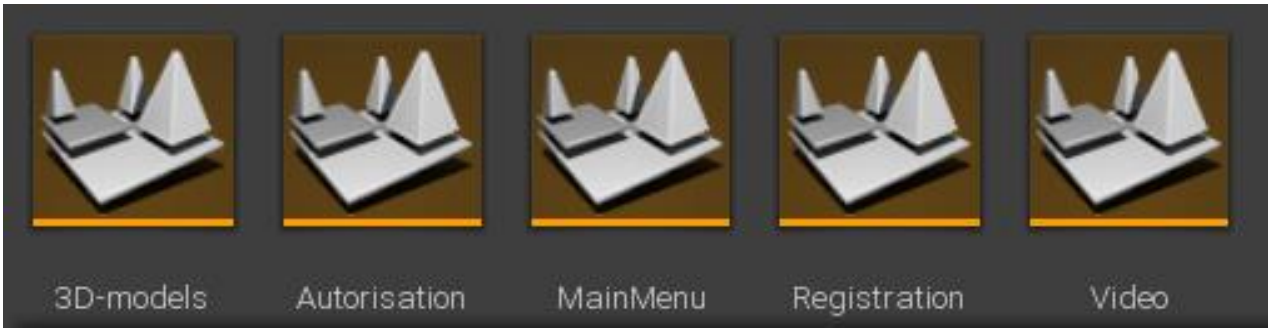

Рисунок 9 – Уровни приложения

Для создания интерфейса приложения применялись пользовательские Widget Blueprint’ы, т.е. изначально пустые функциональные схемы, в которых впоследствии создавались последовательности вызова функций. В данных

функциях были созданы элементы графического представления, кнопки для вызова функций, условия возникновения событий.

Также у каждого уровня есть Level Blueprint, в котором я создал функции, которые выполняются при возникновении события `BeginPlay`, иными словами, при инициализации исполняемого файла. Подробнее созданный алгоритм, выполняемый при инициализации, описан в пункте, посвященном главному меню.

Рассмотрим каждый элемент в отдельности.

4.1 Авторизация пользователей

Авторизация и регистрация пользователя реализованы на уровнях `Autorisation` и `Registration`.

Авторизация пользователя происходит путем введения логина и пароля. Если пользователь ввел верные логин и пароль, то система считает его авторизованным и пропускает в главное меню. В противном случае выводится сообщение о неправильном вводе и вновь происходит переход в окно авторизации. На рисунке 10 демонстрируется интерфейс авторизации пользователя.

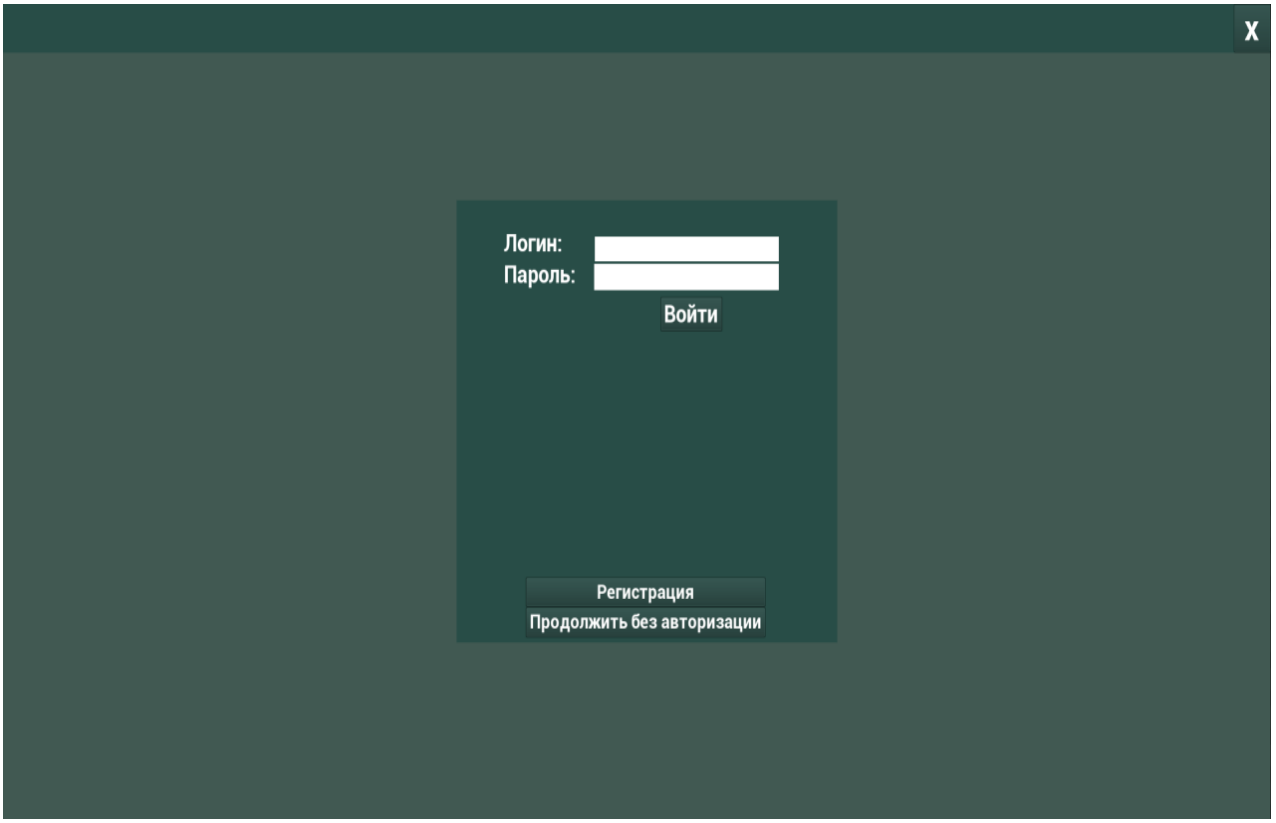

Рисунок 10 – Интерфейс авторизации пользователя

Кнопка «закрыть» или же «x», отвечающая за закрытие приложения. На рисунке 11 представлен граф событий данной кнопки. Она была реализована мной во всех пользовательских интерфейсах приложения.

Условием входа в тело функции является нажатие левой кнопки мыши на кнопку, обозначенную «x». Она расположена в верхней правой части экрана. При попадании в тело цикла выполняется проверка входного значения переменной `Player_Index` типа `byte`. Поскольку предусматривается использование данного приложения одномоментно лишь одним пользователем, то все дальнейшие требования к пользователю, будут закрываться вызовом пользователя с `PlayerIndex = 0`. При успешном прохождении проверки параметр `InputModeGameOnly` с типом данных `bool` будет переведен в состояние true, а параметры `InputModeUIOnly` и `InputModeGameAndUI` с типом данных bool обратятся в состояние false. За данное присвоение новых значений переменным отвечает функция `SetInputMode`. Необходимость создания данной

функции заключается в сокращении количества ошибок при генерации запросов. Для закрытия приложения необходимо прописать запрос, функцией которого служит `Exit`, после которой происходит выход из тела Название_уровня.h. Для обращения к библиотекам UE4 обращение к этой функции было переименовано в `Quit`.

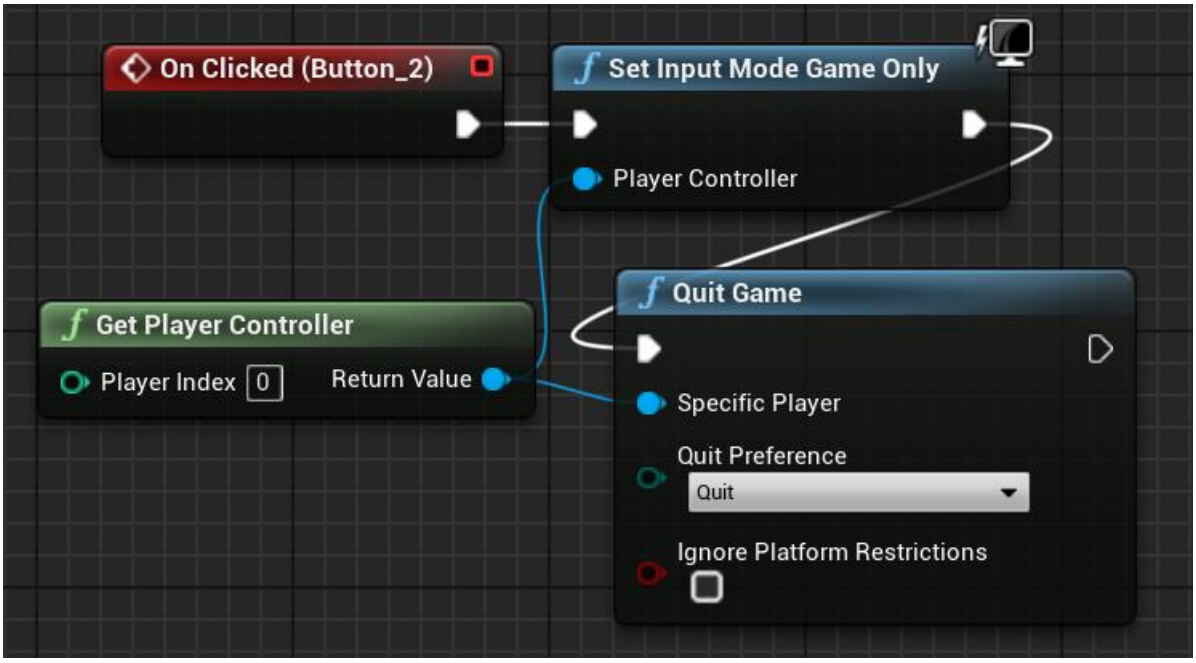

Рисунок 11 – Граф событий кнопки закрытия

Кнопки «Войти», «Регистрация» и «Продолжить без авторизации» открывают уровень, соответствующий их назначению. На рисунке 12 представлен граф событий перехода на другой уровень. Алгоритм тела функций схож с функцией выхода, но изменяется запрос с выхода из тела программы, на открытие другого исполняемого файла (уровня). Поиск необходимого исполняемого файла происходит по названию файла с критерием соответствия формату *.umap. При корректном осуществлении запроса происходит инициализация полученного при запросе файла, а нынешний исполняемый файл вызывает функцию закрытия. При этом заголовочный файл продолжает выполнение в фоновом режиме.

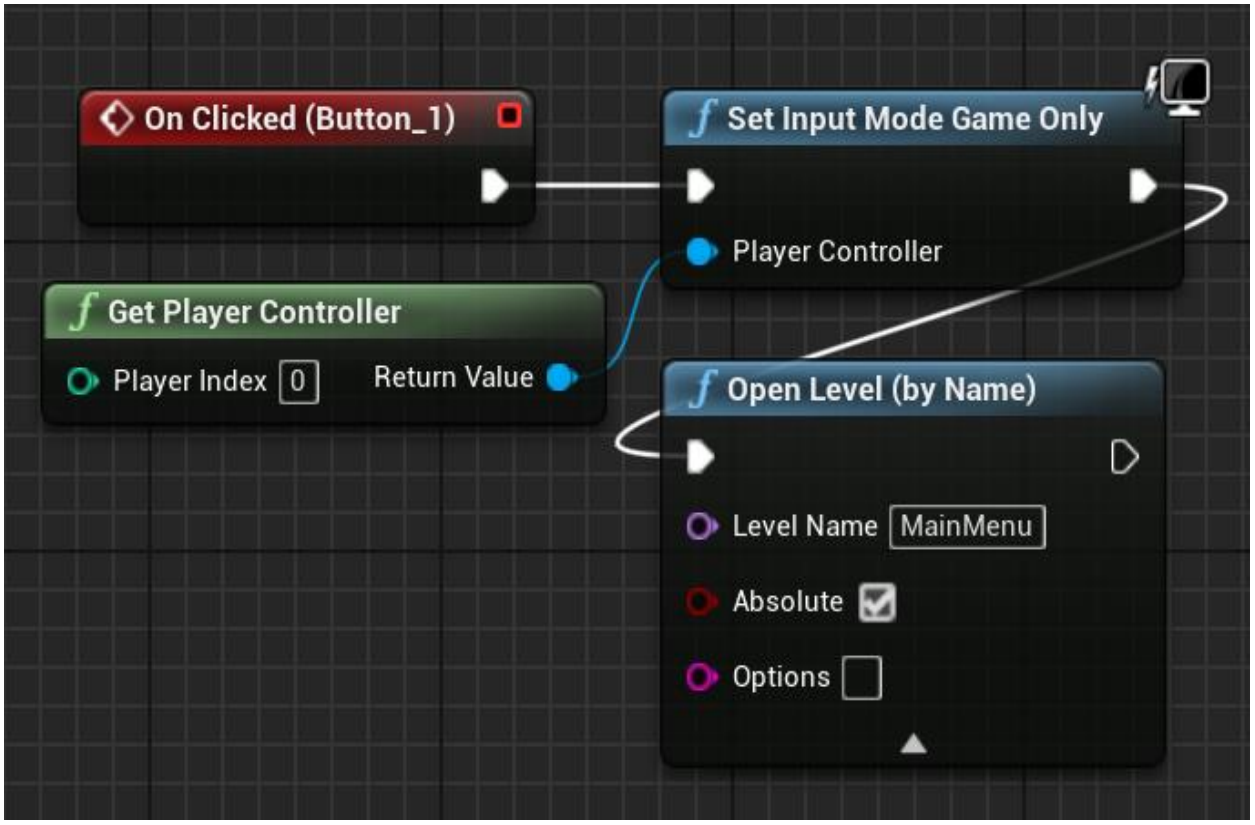

Рисунок 12 – Граф событий кнопки «Продолжить без авторизации»

Реализована функция регистрации пользователя в системе приложения. При переходе в данное окно пользователю предлагают придумать логин, ввести пароль и подтвердить его. При подтверждении регистрации система проверяет наличие введенного логина в системе. Если совпадений нет, то регистрация завершается успешно. В противном случае выводится сообщение о том, что данный логин занят, и просьба изменить логин. На рисунке 13 демонстрируется интерфейс регистрации пользователя.

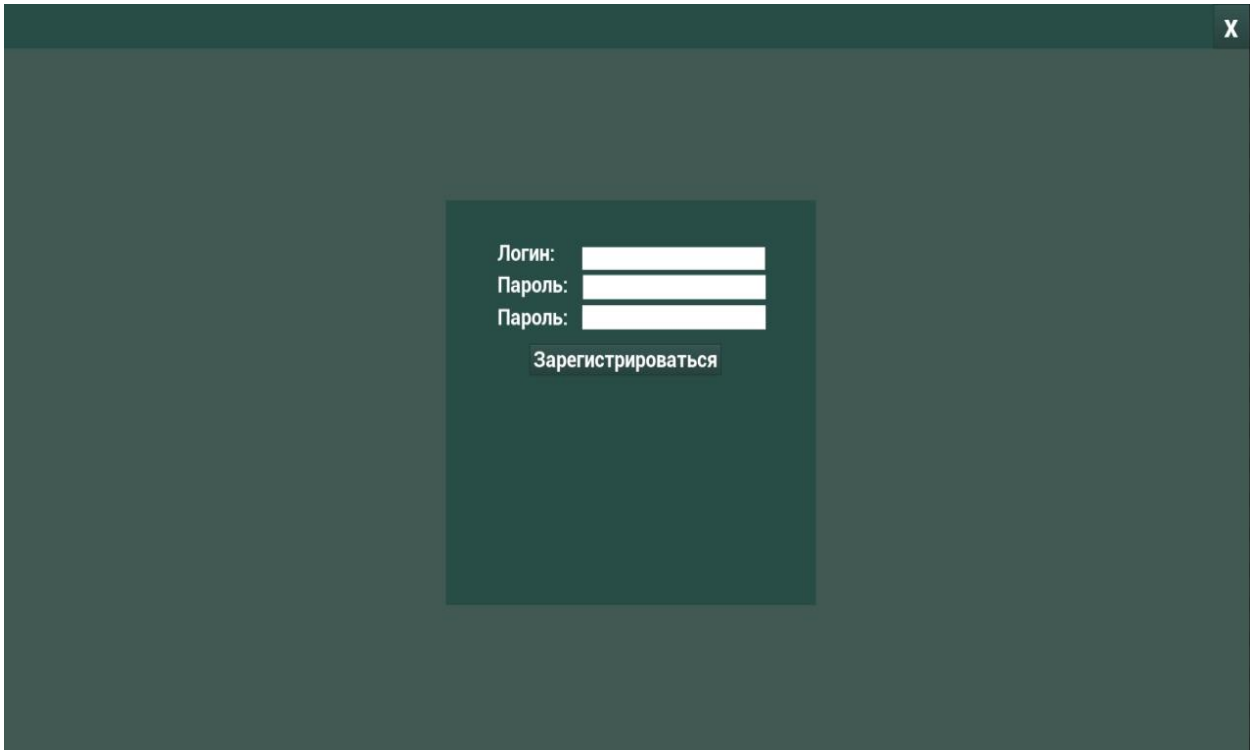

Рисунок 13 – Интерфейс регистрации пользователя

Пользователь может работать в неавторизованном режиме, для этого существует кнопка «Продолжить без регистрации». При нажатии на нее происходит проверка на время использования приложения. Если срок использования не истек, окно авторизации пропускается, и происходит переход в главное меню.

4.2 Главное меню

Главное меню реализовано на уровне `MainMenu`.

На рисунке 14 представлен интерфейс главного меню. Попадая в это окно, пользователь может перейти в 3 разных интерфейса.

Кнопка «3D-моделирование» отвечает за переход в раздел моделирования, в котором представлены инструменты работы с моделями, а также создание анимации.

Кнопка «Видеоредактор» открывает интерфейс работы с видеорядом.

Кнопка «Инструменты» заблокирована, поскольку на данный момент не реализована возможность прямой интеграции инструментов.

Каждое окно открывается отдельно, поскольку в их уровнях находятся объекты на сцене, которые нельзя объединить на одном уровне.

Для реализации перехода в запрос `OpenLevel` передается параметр названия уровня с типом данных String. Для открытия видеоредактора параметр `LevelName = "Video"`, для 3D-моделирования `LevelName = "3D-models"`.

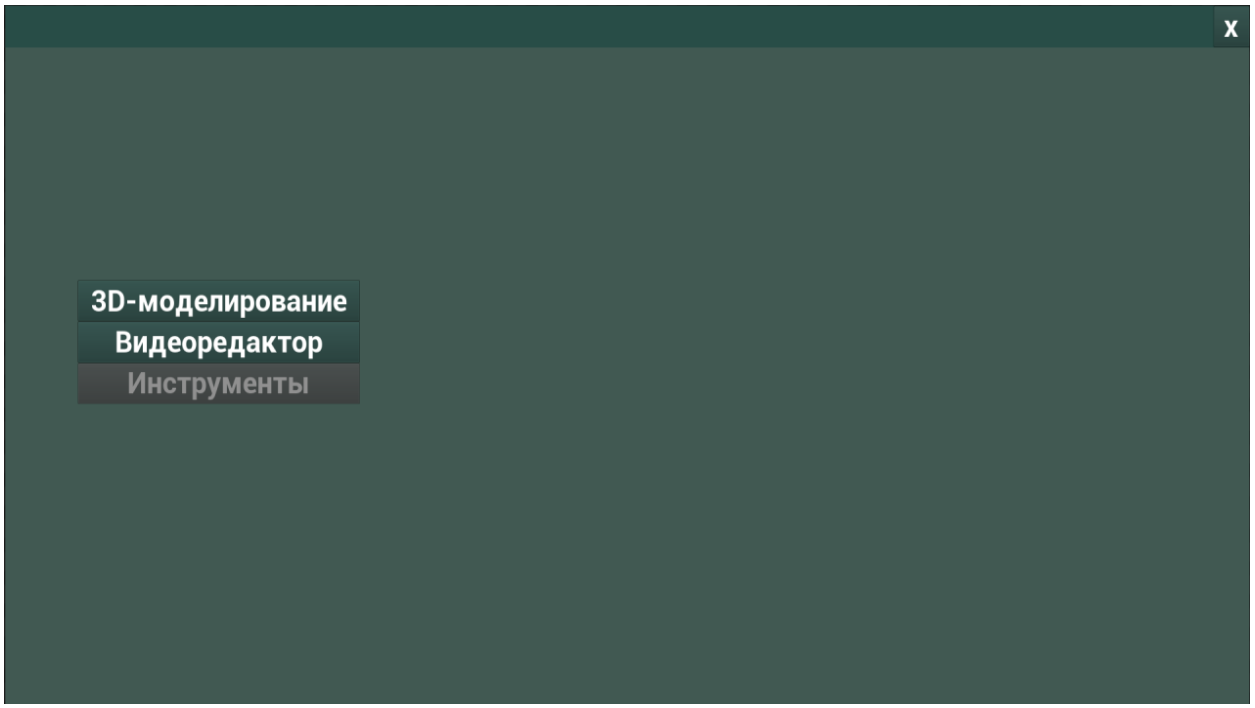

Рисунок 14 – Интерфейс главного меню

На рисунке 15 представлена реализация интерфейса главного меню.

При вызове исполняемого файла уровня `MainMenu` из исполняемого файла `Autorisation.cpp` пользователю на экран должен выводиться интерфейс этого окна. Для этого в исполняемом файле `MainMenu.cpp` переменная `ShowMouseCursor` с типом данных `bool` должна инвертироваться из изначального значения false в true. Инверсия значения `ShowMouseCursor` будет выполняться функцией класса `UPlayerController SetShowMouseCursor`. При вызове она инвертирует значение, которое хранится в `ShowMouseCursor`.

Теперь необходимо инициализировать выполнение функции, в которой мы разработали интерфейс. Для вызова напишем функцию `CreateNewWidget`, входными значениями которой будут `ClassName` типа данных `string` и `PlayerIndex` типа данных `byte`. При совпадении условий, что в функцию обращается пользователь с `PlayerIndex = 0`, и что данный `ClassName` существует на экран будет транслироваться интерфейс из вызванного `ClassName` и курсор мыши.

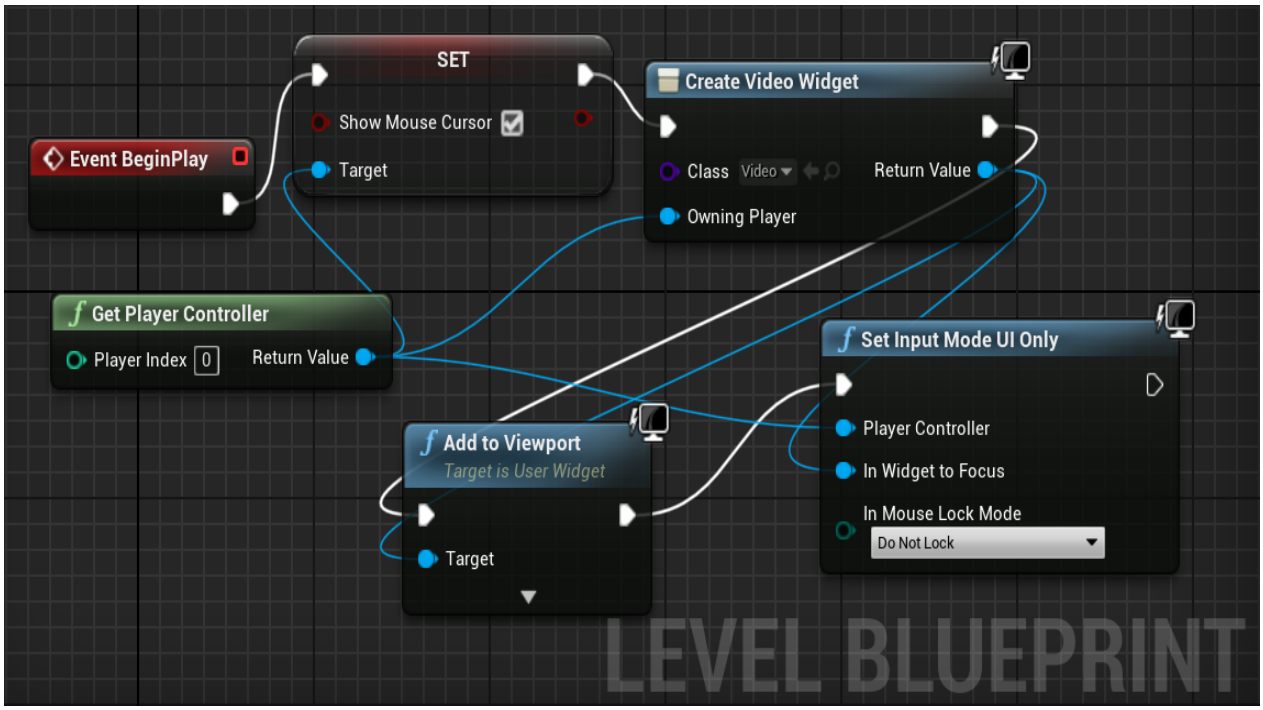

Рисунок 15 – Реализация интерфейса главного меню

Аналогичным алгоритмом реализованы и другие интерфейсы, с той разницей, что генерируются собственные классы каждого интерфейса.

4.3 Работа с 3d-моделями

Окно работы с 3D-моделями реализовано на уровне `3D-models`.

Перейдя в это окно, пользователь может рассмотреть со всех сторон заранее выбранную 3D-модель, а также посмотреть ее анимации. На рисунке 16 представлен интерфейс 3D-моделирования.

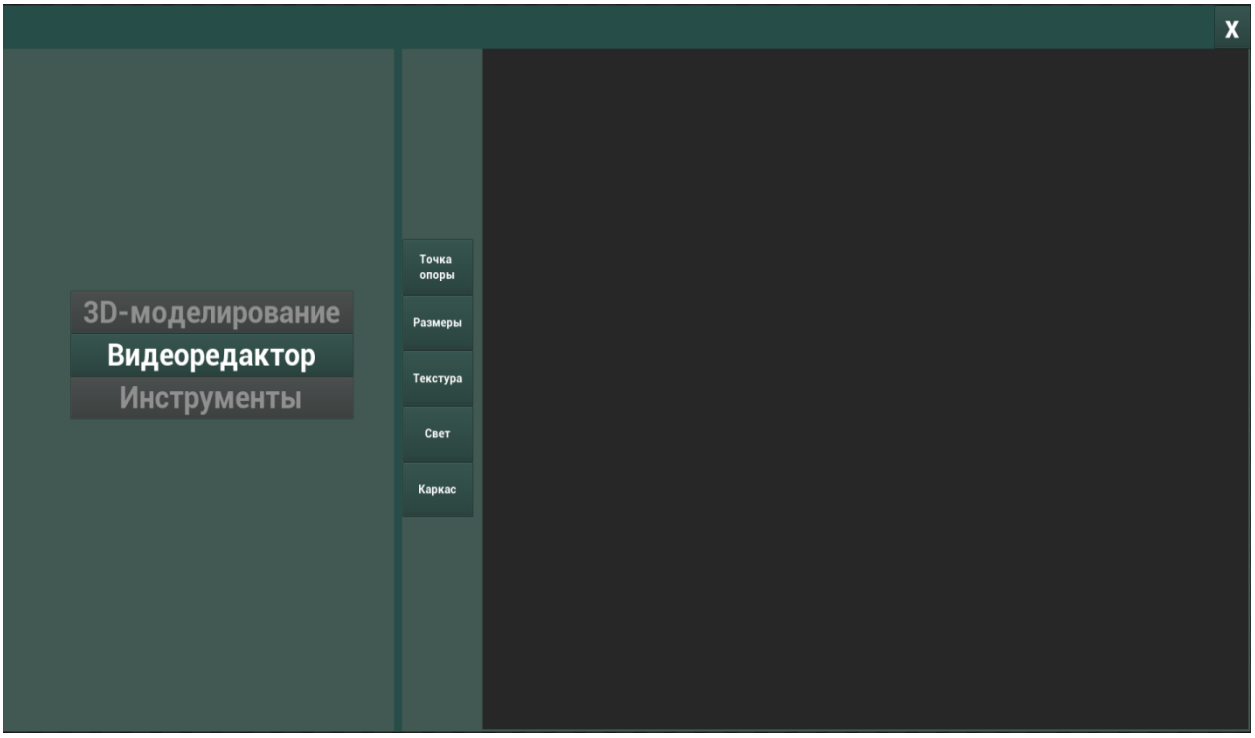

Рисунок 16 – Интерфейс 3D-моделирования

Справа расположено окно для просмотра 3D-модели. Здесь же можно посмотреть анимации модели, путем нажатия на клавиши WASD.

По центру расположены инструменты работы с моделями:

– установить точку опоры;

– настройка размеров модели;

– выбор текстур модели;

– настройка источников света;

– создание каркаса модели.

Рассмотрим устройство сцены уровня с 3D-моделями.

На рисунке 17 представлены созданные объекты на сцене, а на рисунке 18 их графическое отображение.

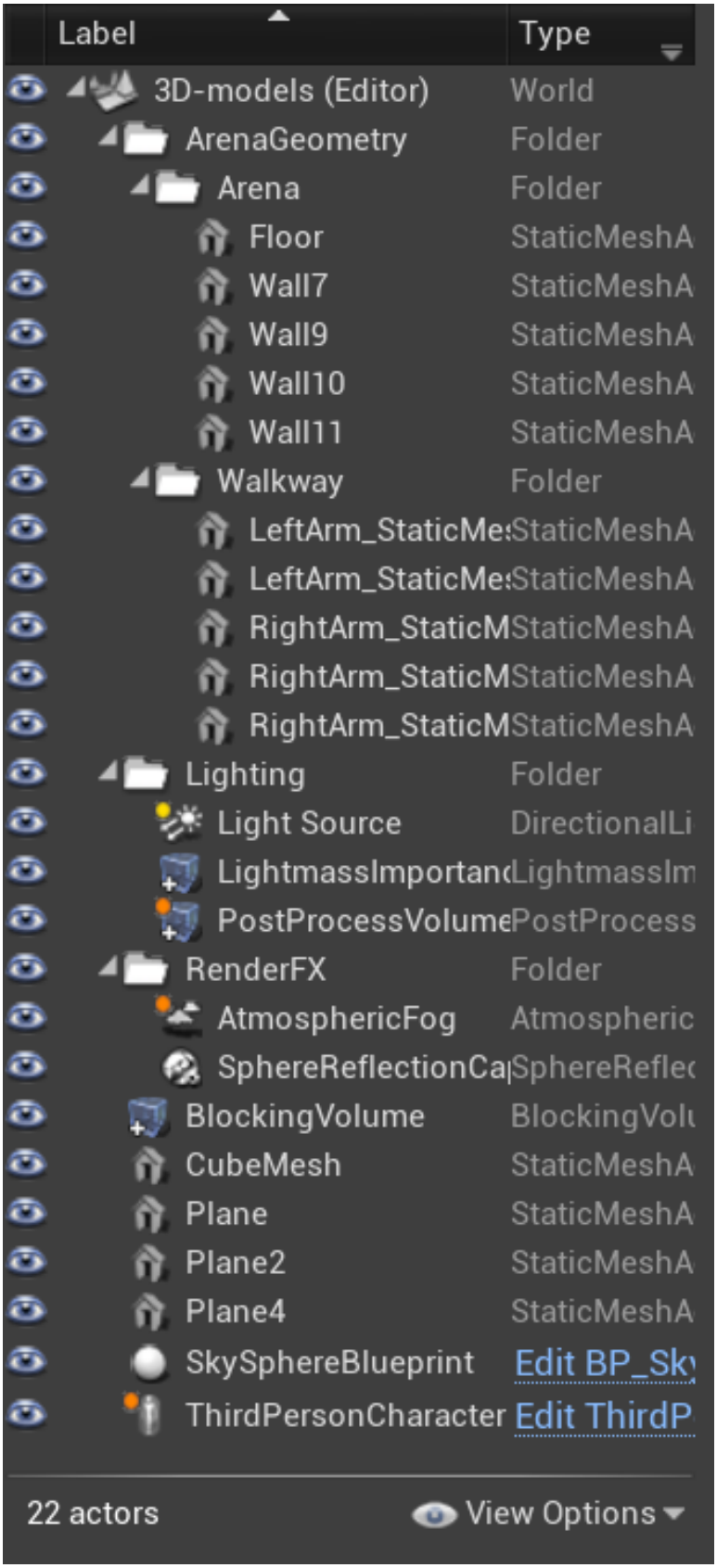

Рисунок 17 – Список объектов на уровне 3D-models

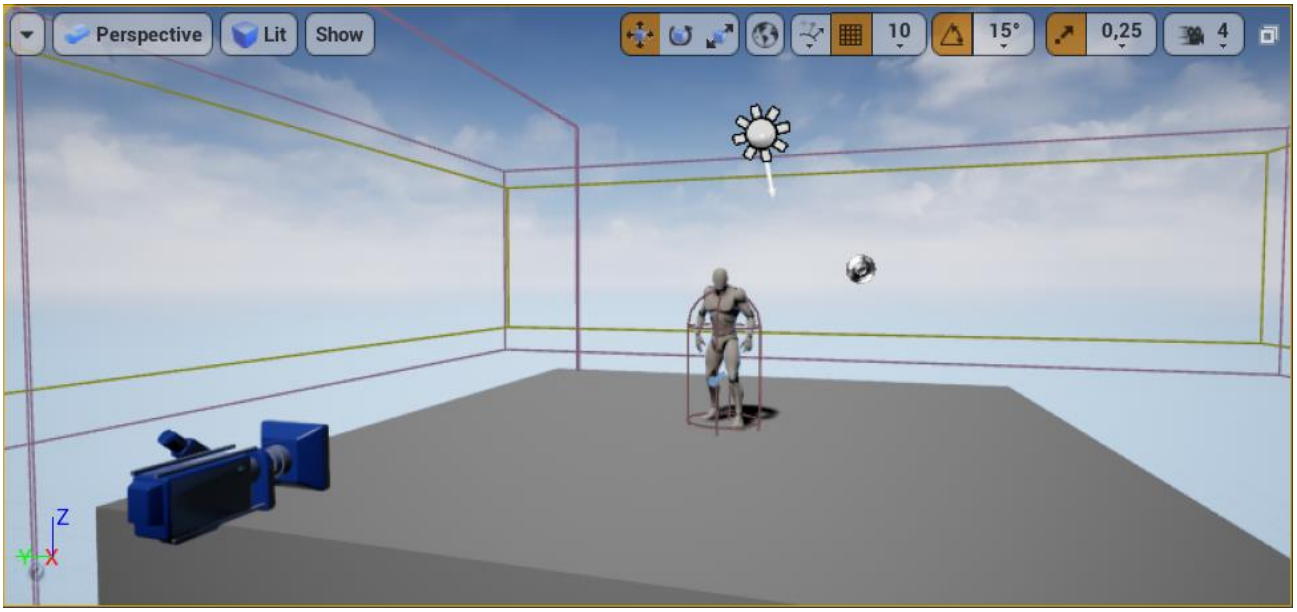

Рисунок 18 – Графическое отображение объектов на уровне 3D-models

В разделе `Arena` представлены объекты для присваивания заднего фона. `Walkway` – совокупность объектов, отвечающих за поверхность, на которой стоит 3D-модель. Для сцены уровня `3D-models` данные объекты видимы для пользователя. `Lighting` отвечает за освещение и тень на всей сцене. `RenderFX` – генерация подвижных текстур для фона.

Для того чтобы рассмотреть модель и проверить анимацию движения, предусмотрен функционал передвижения по платформе `Walkway`. Активация этого происходит путем нажатия на экран с моделью. После этого пользователь при помощи клавиш WASD может побегать за 3D-модель.

Для создания данной функции реализован `New C++ Class`, т.е. был написан и интегрирован в среду разработки UE4 код для выполнения анимации.

В заголовочном файле (листинг 1) необходимо объявить наследование от класса `Character` для созданного нами `My3DObject`. Данное наследование было выполнено для возможности использование данного класса средой разработки UE4 в роли объекта, управление над которым принадлежит пользователю. `GENERATED_BODY` создает сетку анимации. Объявляем методы `BeginPlay` как защищенный, `Tick` и `SetupPlayerInputComponent` как открытый. Данные

методы объявлены для корректной работы класса в приложении. Последний метод будет получать на вход введенную пользователем команду. Команды, связанные с самой анимацией (`MoveForward` и `MoveRight`), объявим как приватные методы класса, на вход которых будет приходить значение типа данных `float` для количественного обозначения смещения.

Листинг 1 – Код заголовочного файла анимации 3D-моделей

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Character.h"
#include "My3DObject.generated.h"

UCLASS
class My3DObject_API AMy3DObject : public ACharacter
{
	GENERATED_BODY()

public:
	AMy3DObject();
protected:
	virtual void BeginPlay() override;
public:
	virtual void Tick(float DeltaTime) override;
	virtual void SetupPlayerInputComponent(class
UInputComponent* PlayerInputComponent) override;
private:
	void MoveForward(float AxisValue);
	void MoveRight(float AxisValue);
};
```

В исполняемом файле (листинг 2) создадим обработчики методов, объявленных ранее. Стоит подробнее остановиться на обработчике метода `SetupPlayerInputComponent`. Здесь выполнена привязка клавиш к действиям, которые будет выполнять модель. При движении модели вперед и назад задействуется обработчик `MoveForward`. При движении влево и вправо – `MoveRight`. Текстовое входное значение – обозначение данных привязок к кнопкам в среде UE4. Также стоит подробнее разобрать сами обработки движения. В обработчике создается вектор движения на основе того, какая

команда введена. После создания вектора происходит перемещение на величину `AxisValue`.

Листинг 2 – Код исполняемого файла анимации 3D-моделей

```
#include "My3DObject.h"
#include "Kismet/KicmetMathLibrary.h"

AMy3DObject::AMy3DObject()
{
     PrimaryActorTick.bCanEverTick = true;
}

void AMy3DObject::BeginPLay()
{
     Super::BeginPlay();
}

void AMy3DObject::Tick(float DeltaTime)
{
     Super::Tick(DeltaTime);
}

void AMy3DObject::SetupPlayerInputComponent(UInputComponent*
PlayerInputComponent)
{
     Super::SetupPlayerInputComponent(PlayerInputComponent);
     PlayerInputComponent->BindAxis(TEXT("MoveForward"), this,
&AMy3DObject::MoveForward);
     PlayerInputComponent->BindAxis(TEXT("MoveRight"), this,
&AMy3DObject::MoveRight);
}
void AMy3DObject::MoveForward(float AxisValue)
{
     FVector Forward =
UKismetMathLibrary::GetForwardVector(FRotator(0,GetControlRota
tion().Yaw, 0));
     AddMovementInput(Forward(), AxisValue);
}
void AMy3DObject::MoveRight(float AxisValue)
{
     FVector Right =
UKismetMathLibrary::GetRightVector(FRotator(0,
GetControlRotation().Yaw, 0));
     AddMovementInput(Forward, AxisValue);
}
```

Для того чтобы модель не упала с платформы, были созданы невидимые стены. Их функционал выполняют объекты BlockingVolume, Plane, Plane2, Plane4.

ThirdPesonCharacter – сама 3D-модель, над которой будут происходить все преобразования. Поскольку в редакторе нас интересует графическая составляющая, рассмотрим, над какими параметрами мы можем производить манипуляции для изменения внешнего вида.

На рисунке 19 представлены основные параметры отображения. В настройках Transform нас интересует параметр Scale, который имеет 3 координаты. Каждая из них отвечает за размеры по оси X, Y или Z, соответственно. Animation отвечает за отображение движения модели. В Animation Mode выбираем пункт, что будем использовать анимацию, написанную на blueprint. В самом же blueprint выставим наследование нашей модели от созданного C++ класса My3DObject. Таким образом мы интегрируем разработанный нами класс анимации в среду UE4 без необходимости дополнительных конвертаций. В Anim Class устанавливаем алгоритм, просчитывающий передвижения модели, т.е. после установки собственной анимации My3DObject данный алгоритм возвращает нам AxisValue типа float. Граф данного алгоритма представлен на рисунке 20.

В настройках Mesh устанавливается сам объект, на который впоследствии накладывается текстура. Skeleton Mesh – непосредственно 3D-модель, которую загружает пользователь. На данный момент в базе Skeleton Mesh мной созданы две сетки – мужской и женский вариант антропоморфных моделей.

В настройках Materials устанавливаются текстуры 3D-моделей. Количество устанавливаемых текстур варьируется от модели к модели. В моем случае, таких текстур необходимо две. Element 0 отвечает за контуры и

текстуры всего тела модели. `Element 1` накладывается поверх нулевого в уровне торса для дополнительных графических элементов.

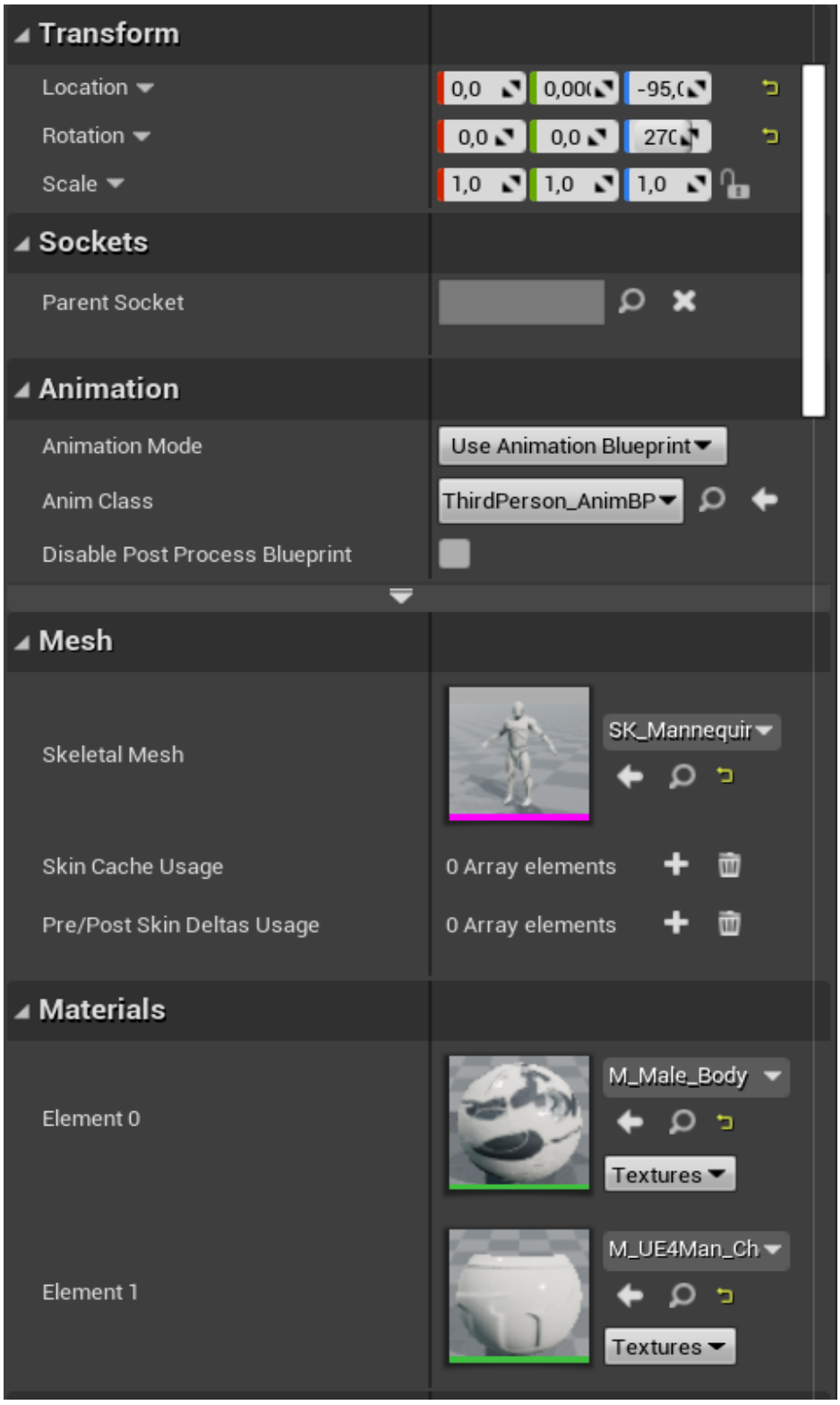

Рисунок 19 – Основные параметры отображения персонажа

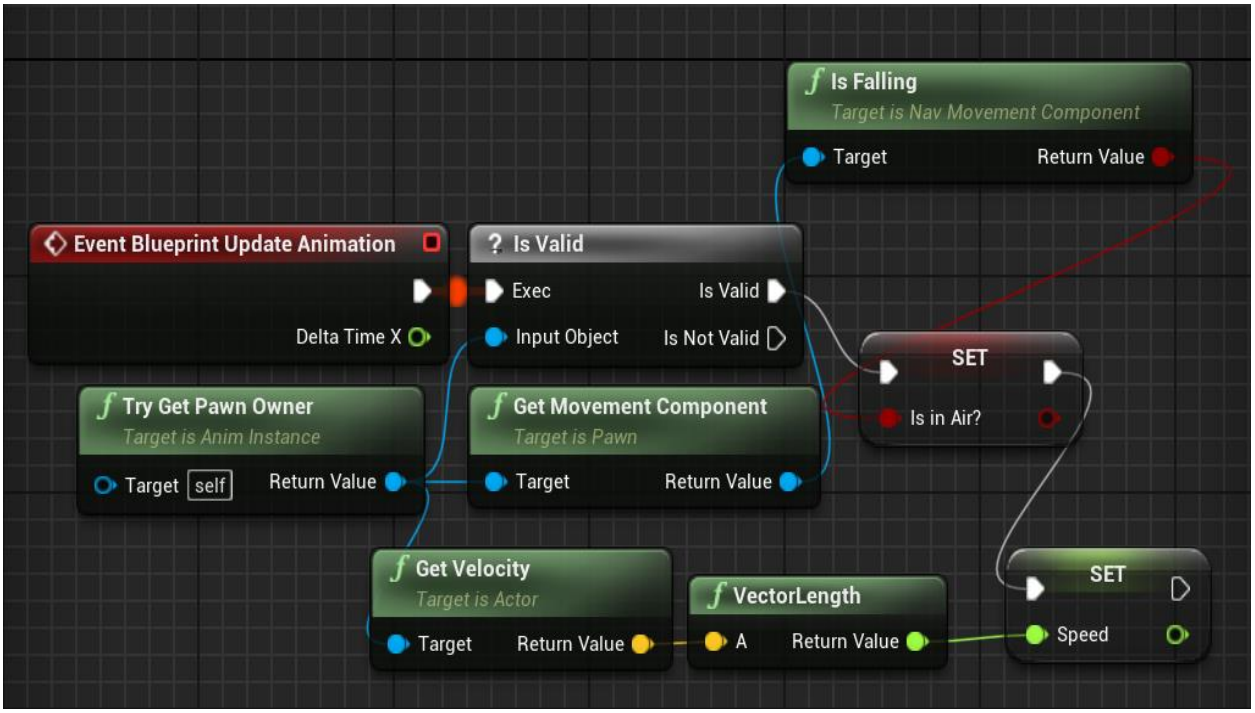

Рисунок 20 – Граф алгоритма расчета анимации персонажа

Для того, что бы вызвать созданный `Mesh`, его необходимо поместить по следующему пути: Приложение\Content\Mennequin\Character\Mesh. Данное хранилище будет использоваться движком Unreal Engine для поиска 3D-моделей по полю `SkeletalMeshName` с типом данных `string` со свойством расширения *.uasset. На рисунке 21 представлены созданные на данный момент `Skeleton Mesh`.

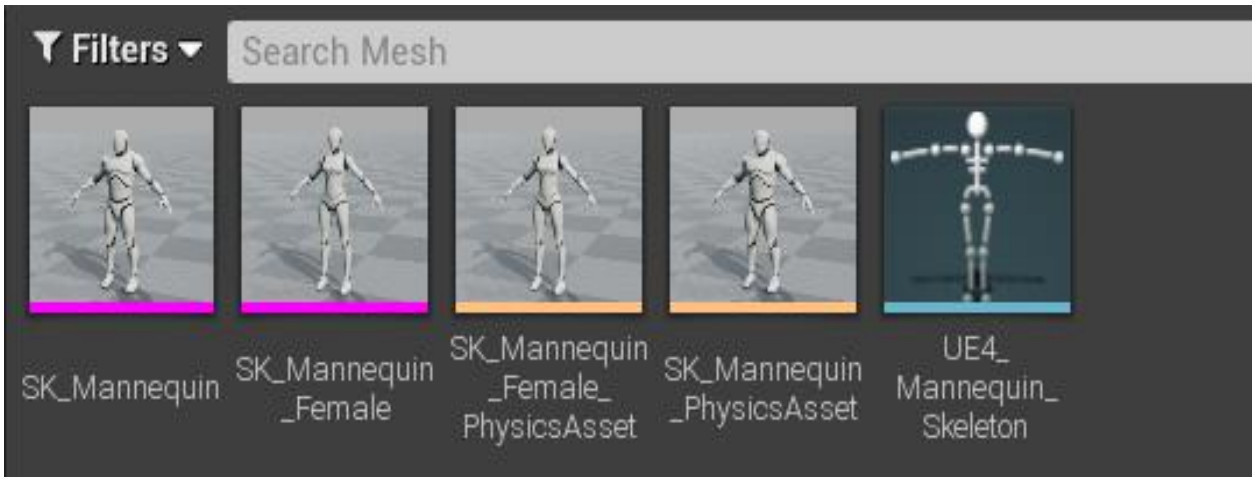

Рисунок 21 – База Mesh’ей

4.4 Обработка видео

Окно работы с видео реализовано на уровне `Video`.

На рисунке 22 представлен интерфейс обработки видео. Данное окно создано идентично интерфейсу 3D-моделирования в плане отображения информации для большей интуитивности процесса работы пользователя.

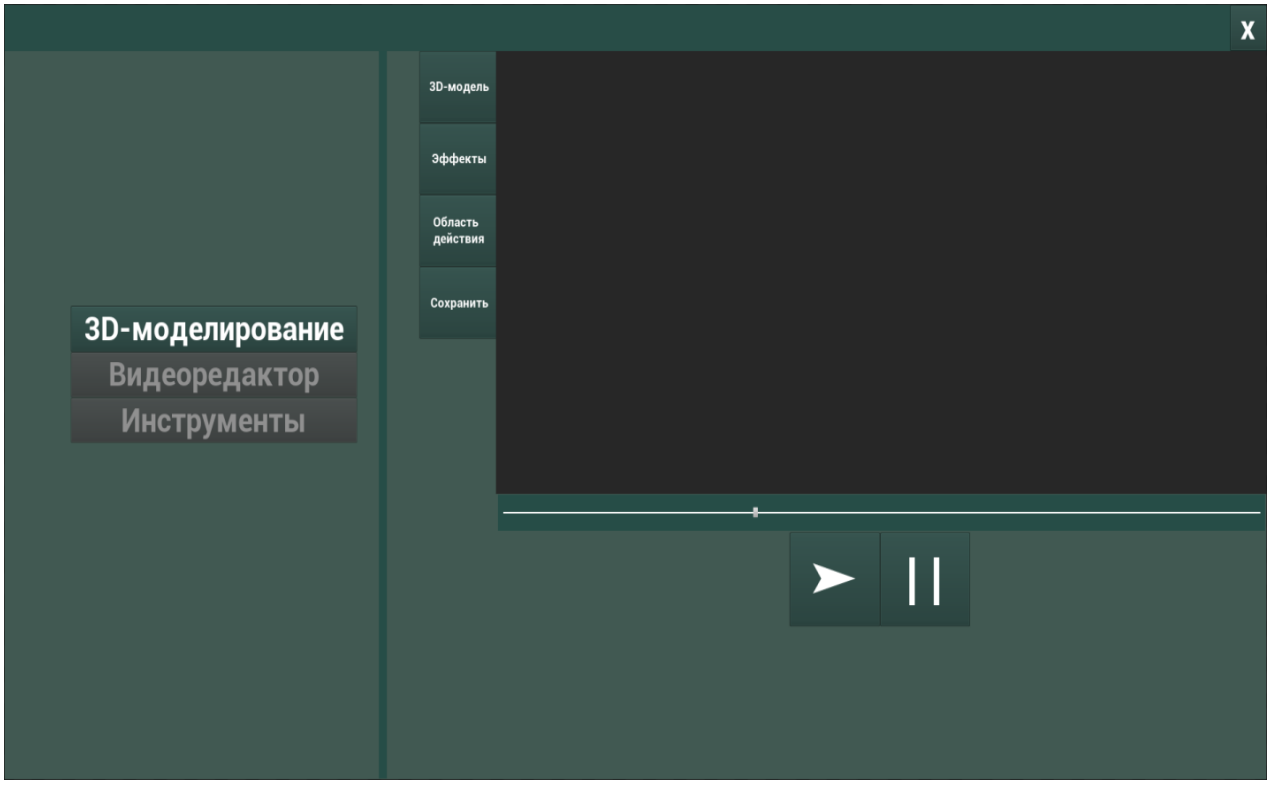

Рисунок 22 – Интерфейс обработки видео

Ниже окна отображения видео расположена шкала времени, которая применяется для прокрутки видеофрагмента вперед в ускоренном темпе, или же назад, в темпе стандартного воспроизведения видео. Граф алгоритма для этого элемента представлен на рисунке 23.

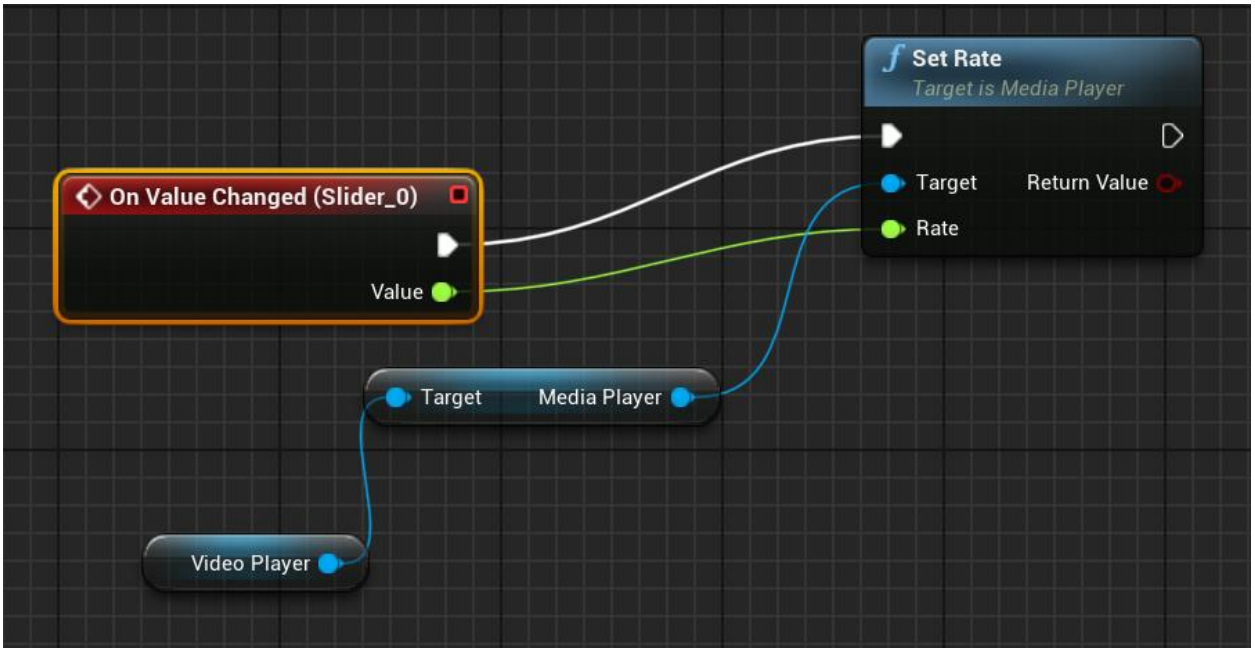

Рисунок 23 – Граф алгоритма шкалы прокрутки видео

`Slider_0` передает значение на функцию установки скорости воспроизведения. Диапазон передаваемых значений находится в пределах от -1 до 2, где -1 – проигрывание видео в обратном порядке со скоростью 1 к 1 в стандартном режиме воспроизведения, 2 – ускоренный в 2 раза режим воспроизведения видео. Целью, на которую применяется функция изменения скорости, является `Video Player`. Данный объект наследует методы базового класса `Media Player`.

В области инструментов представлены эффекты для видеоряда, добавление 3D-моделей.

На рисунке 24 представлена реализация окна проигрывания видео.

Проигрыватель видео начинает свою работу, как только был собран интерфейс, к которому он присоединен, а именно интерфейс обработки видео. После выполнения события происходит обращение к `Media Player`, который хранит в себе информацию о видеоряде.

Спустя некоторую задержку для смягчения картинки с нулевого кадра начинается отображение видео.

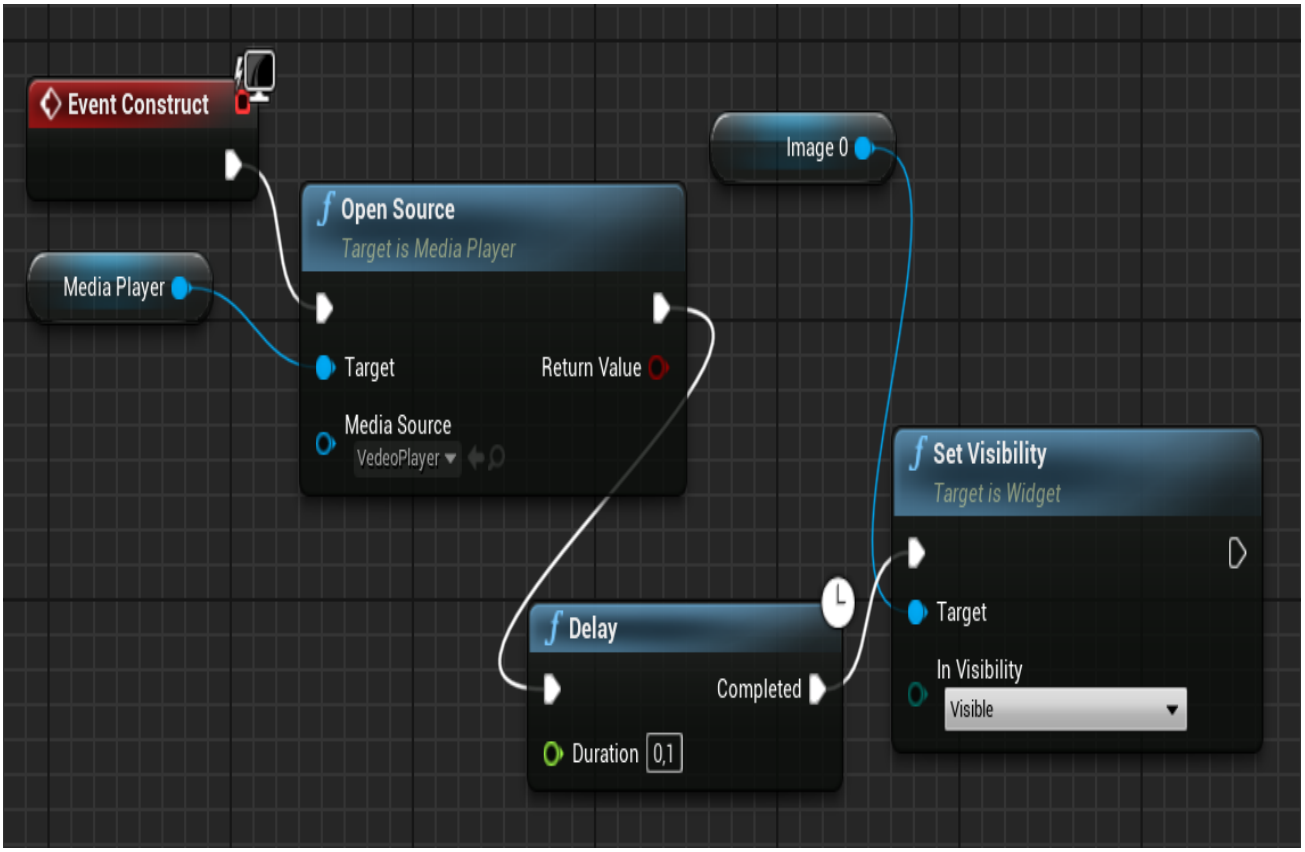

Рисунок 24 – Реализация окна проигрывания видео

Рассмотрим подробнее устройство уровня, предназначенного для работы с видео. На рисунке 25 представлен список объектов на сцене, а на рисунке 26 – их графическое отображение.

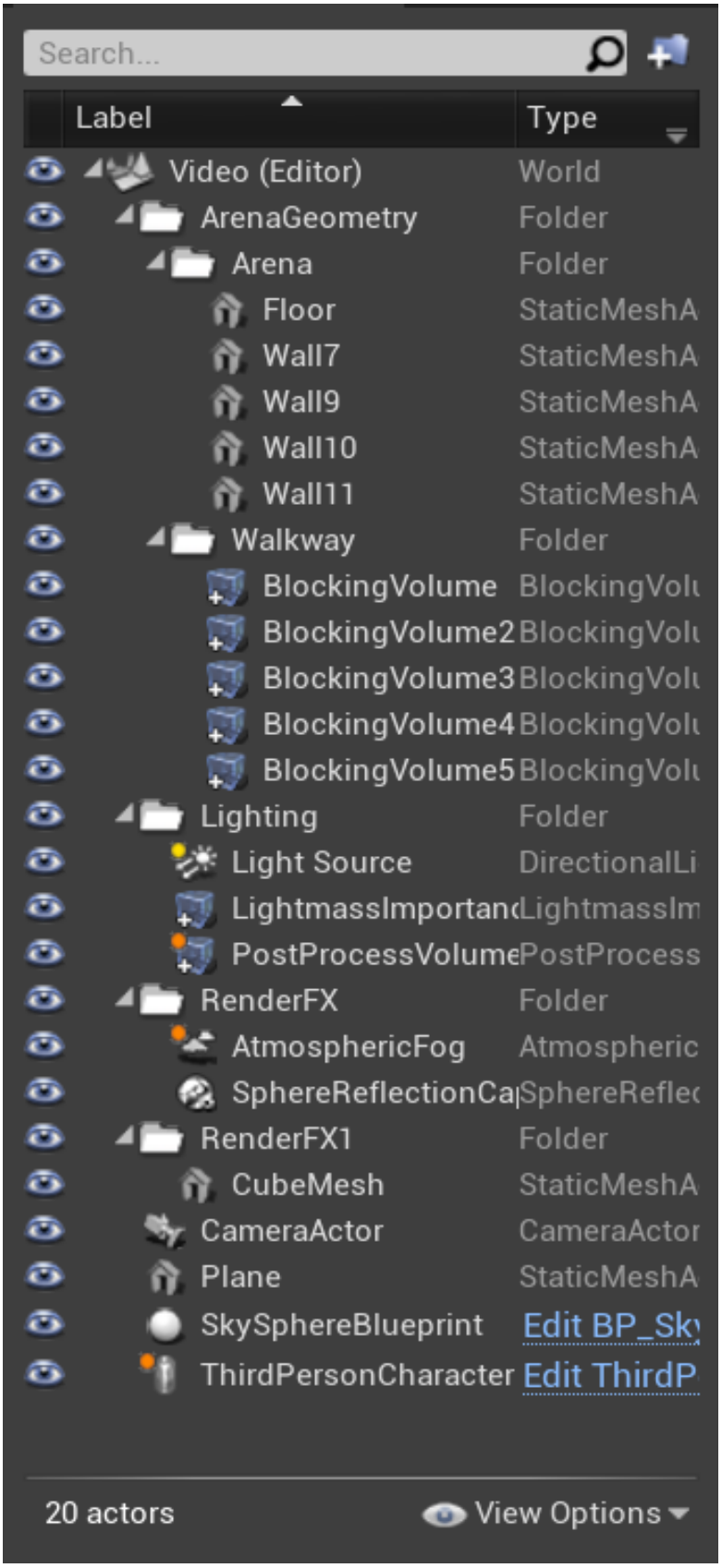

Рисунок 25 – Список объектов на уровне Video

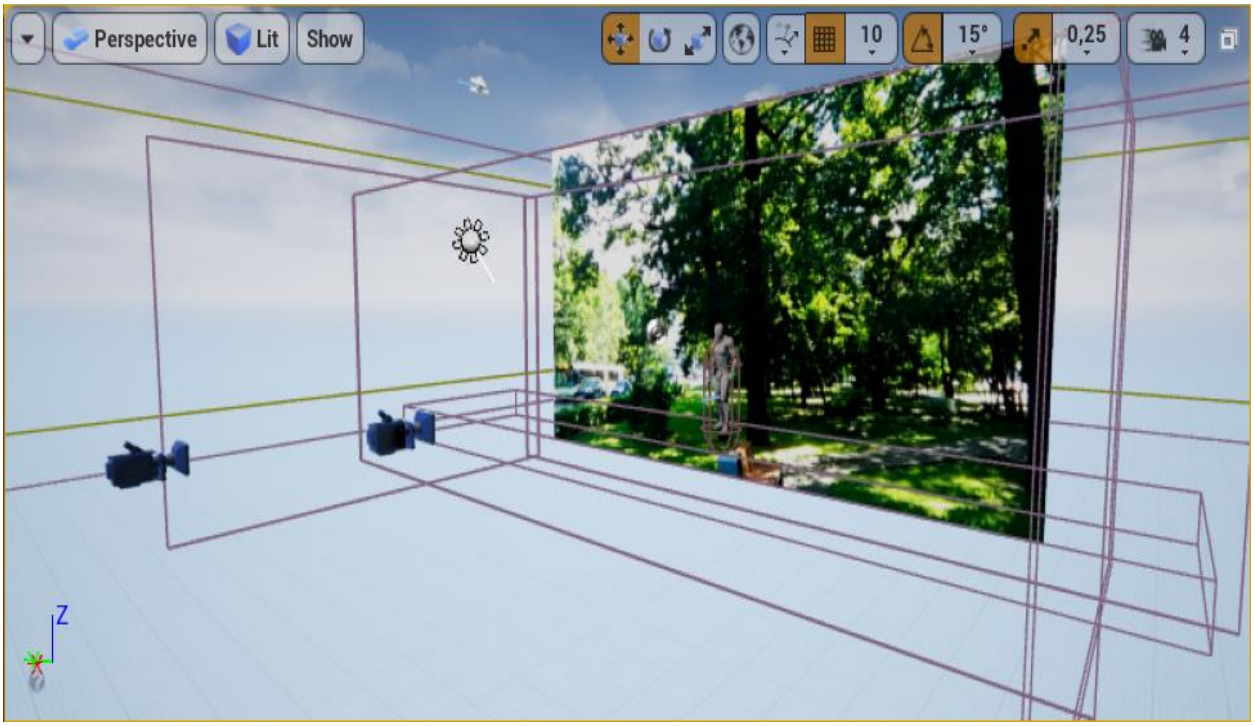

Рисунок 26 – Графическое отображение объектов на уровне Video

Если сравнить сцену уровней `3D-models` и `Video` можно заметить изменения в области `Walkway`, а также добавление таких объектов как `Camera Actor` (2 синих объекта на рисунке 26) и `Plane` (экран с видео за моделью на рисунке 26).

В разделе `Walkway` все видимые ранее объекты были заменены на абсолютно невидимые, но при этом осязаемые 3D-моделью объекты `BlockingVolume` (фиолетовые границы на рисунке 26). Данные изменение вызвано необходимостью расположить 3D-модель на видео, поэтому никаких отображений дополнительных текстур не должно быть.

Дополнительная камера `CameraActor` была создана для статичной картинки, сфокусированной на экране с видео. Таким образом, при движении 3D-модели, камера сохраняет свою позицию. На рисунке 27 отображены настройки для статичности камеры.

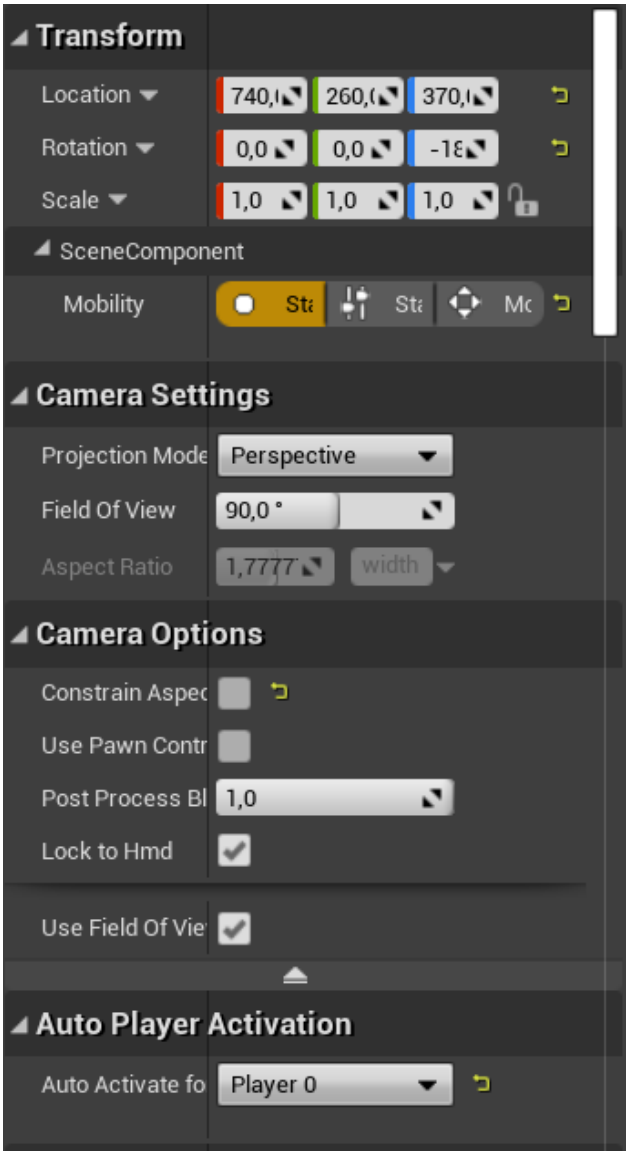

Рисунок 27 – Настройки камеры

Интересующие нас пункты находятся в `Transform` и `Auto Player Activation`.

В параметрах `Transform` переменной `Static` типа данных `bool` присваиваем значение истины. Это заставит камеру не передвигаться ни при каких обстоятельствах во время работы приложения. В параметрах `Auto Player Activation` в метод `AutoActivateFor` передаем переменную типа `byte PlayerIndex = 0`.

Теперь при запуске программы и переходе на уровень `Video` будет запускаться настроенная нами камера.

Рассмотрим функционал добавленного объекта `Plane`. `Plane` является экраном для воспроизведения видео. Чтобы понять, как все устроено, необходимо рассмотреть объекты и материалы, расположенные в созданной мной внутри игровой папке Movies. На рисунке 28 представлено все содержимое данной папки.

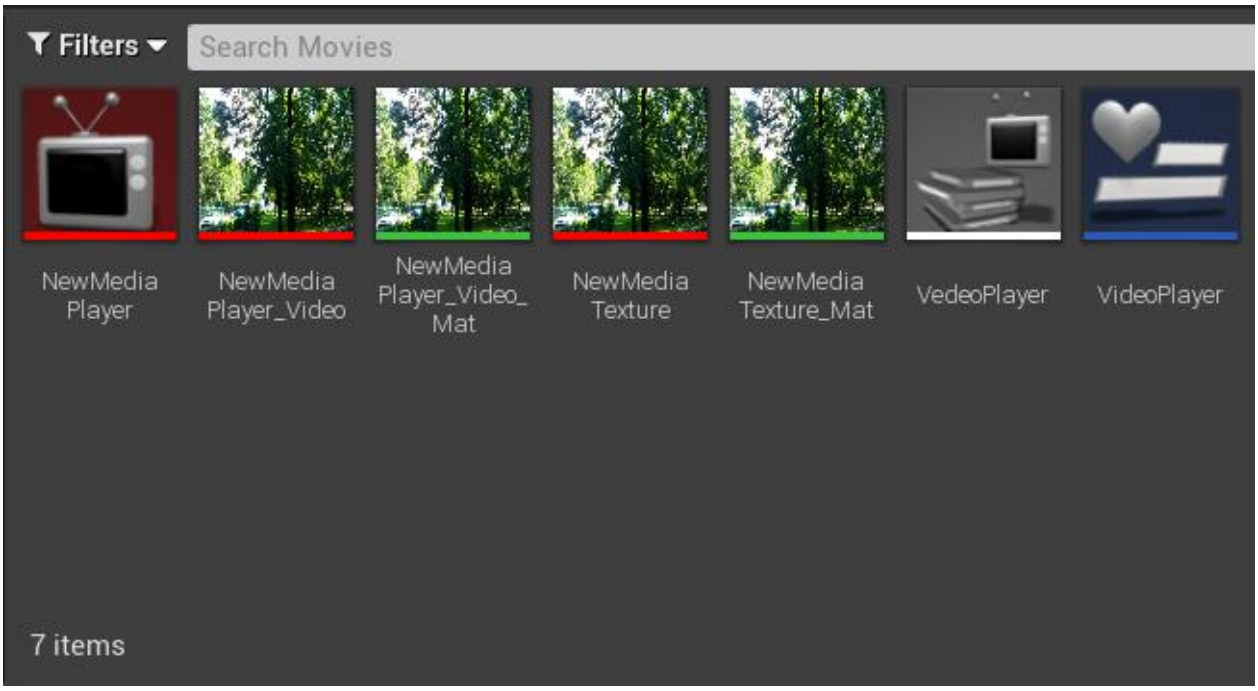

Рисунок 28 – Содержимое папки Movies

`NewMediaPlayer` – предусмотренный движком Unreal Engine элемент, позволяющий проигрывать видеоролики. `NewMediaPlayer_Video` и `NewMediaTexture` являются текстурами, на которые может выводится воспроизводимый в `Media Player` контент. `NewMediaPlayer_Video_Mat` и `NewMediaTexture_Mat` – генерируемые после первого воспроизведения материалы, которые проигрываются на изначально пустых текстурах `NewMediaPlayer_Video` и `NewMediaTexture`.

`VedeoPlayer` является хранилищем и конвертором медиа ресурса. На рисунке 29 представлен интерфейс данного элемента.

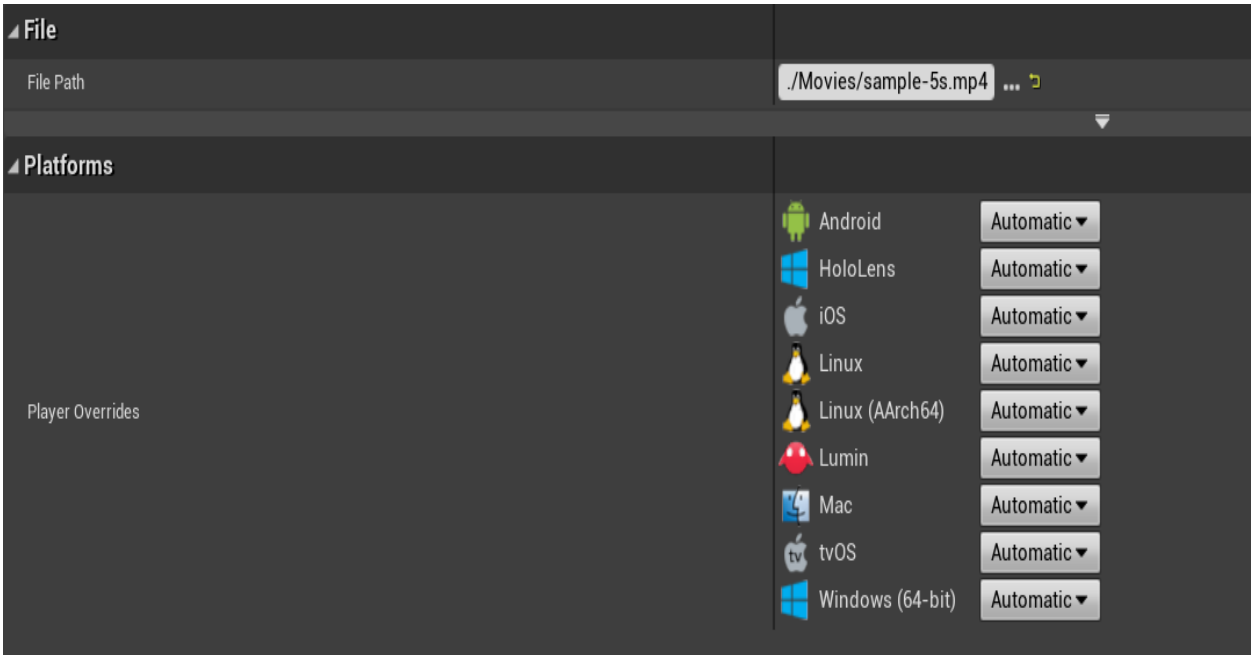

Рисунок 29 – Интерфейс VedeoPlayer’a

В опции `File Path` выбираем файл формата *.mp4. Необходимо, что бы данный файл был расположен в созданной мной внутри проекта папке Movies, в противном случае будет выдавать ошибку о некорректности расположения выбранного файла. В настройках `Platforms` выставляем автоматическую адаптацию под все платформы.

`VideoPlayer` является Widget Blueprint’ом, в котором реализована функция воспроизведения видео из окна пользовательского интерфейса. Граф реализации данного виджета был представлен ранее на рисунке 24.

Вернемся к объекту на сцене уровня `Video Plane`. На сетку данного объекта нанесена текстура `NewMediaTexture`, а поверх нее установлен материал `NewMediaTexture_Mat`. На рисунке 30 представлены настройки `Plane`, которые я описал.

Если бы мы сейчас запустили приложение, то видеоролик не стал бы проигрываться ни при каких условиях. Это связано с тем, что кнопки воспроизведения и остановки принадлежат пользовательскому интерфейсу, а материал с видеороликом – игровому пространству, или же сцене.

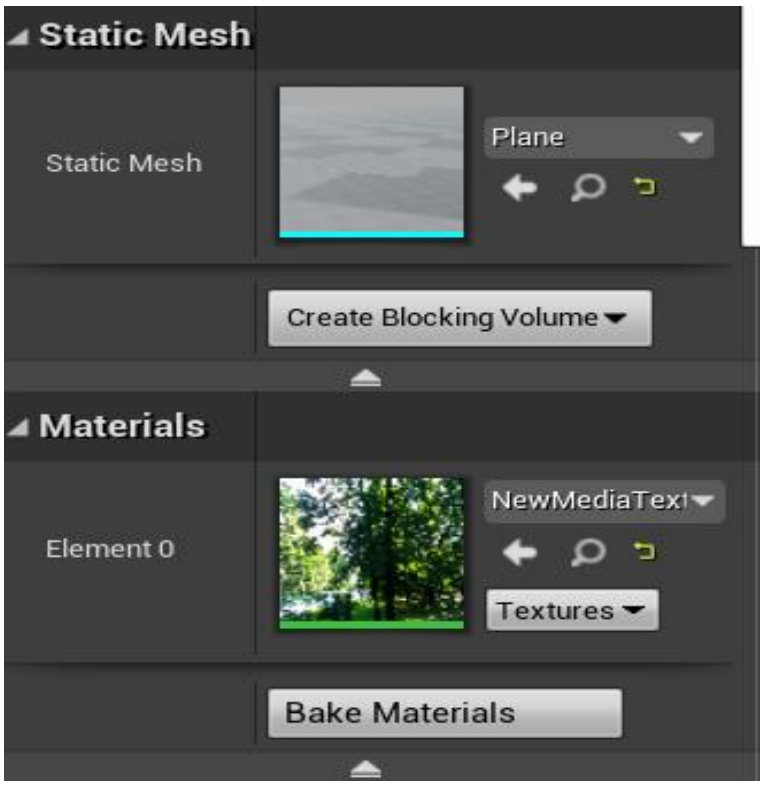

Рисунок 30 – Настройки отображаемой текстуры на объекте Plane

Добавим созданный ранее Widget Blueprint `VideoPlayer` в Widget Blueprint `Video`, который отвечает за пользовательский интерфейс уровня `Video`. Иерархия интерфейса уровня `Video` представлена на рисунке 31.

Поскольку нам нужно отображать как пользовательский интерфейс, так и сцену, то выставляем на данном Widget'е прозрачность `A = 0`, что делает его невидимым для пользователя. Данная настройка находится в параметрах `Appearance` и называется `Color and Opacity`. В этом же пространстве добавим кнопку передачи управления от пользовательского интерфейса к приложению. На рисунках 32 и 33 показана реализация передачи управления.

Для воспроизведения и остановки показа видеоролика, как говорилось ранее, существуют две кнопки. Их можно увидеть на рисунке 20. Они расположены ниже `Slider_0`. При нажатии на кнопку старт, обозначенной стрелкой, вызывается функция `Play`, которая направлена на объект `Video Player`. При нажатии на кнопку остановки, обозначенную двумя

вертикальными чертами, вызывается функция `Pause`, направленная на тот же объект, что и предыдущая функция. Первая запускает ход видео или воспроизводит его заново, если предыдущее проигрывание завершилось. Функция `Pause` останавливает воспроизведение видео, сохраняя позицию на временной шкале. Наглядная демонстрация реализации данных кнопок представлена на рисунке 34.

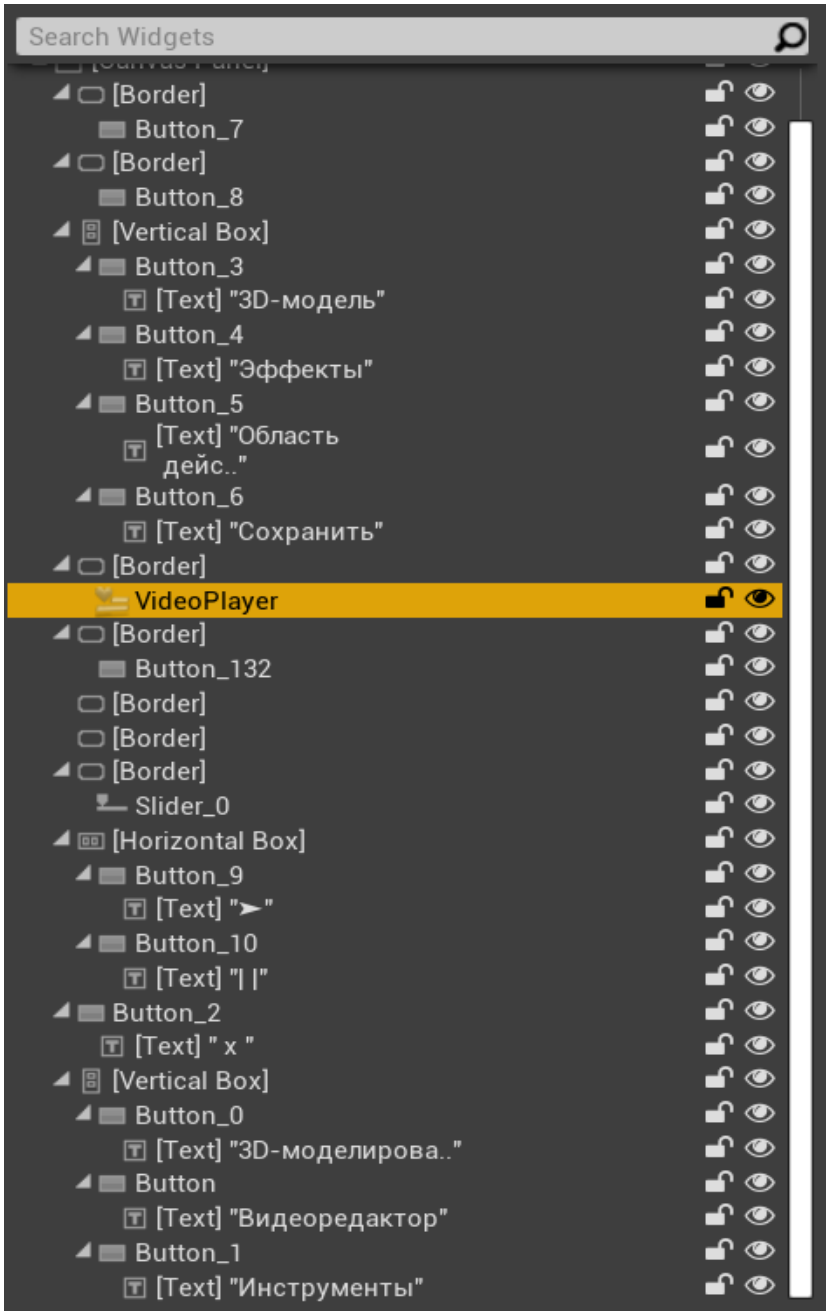

Рисунок 31 – Иерархия элементов графического интерфейса Video

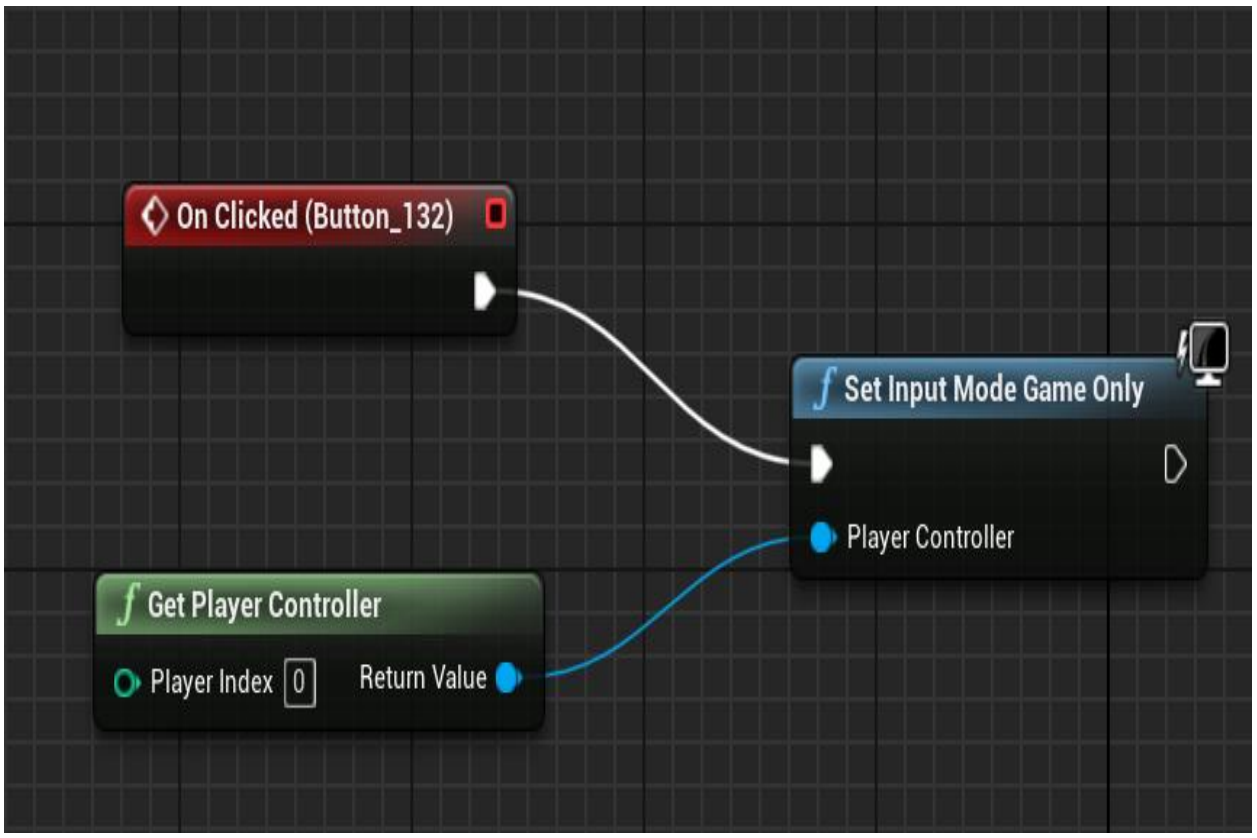

Рисунок 32 – Передача управления от интерфейса приложению

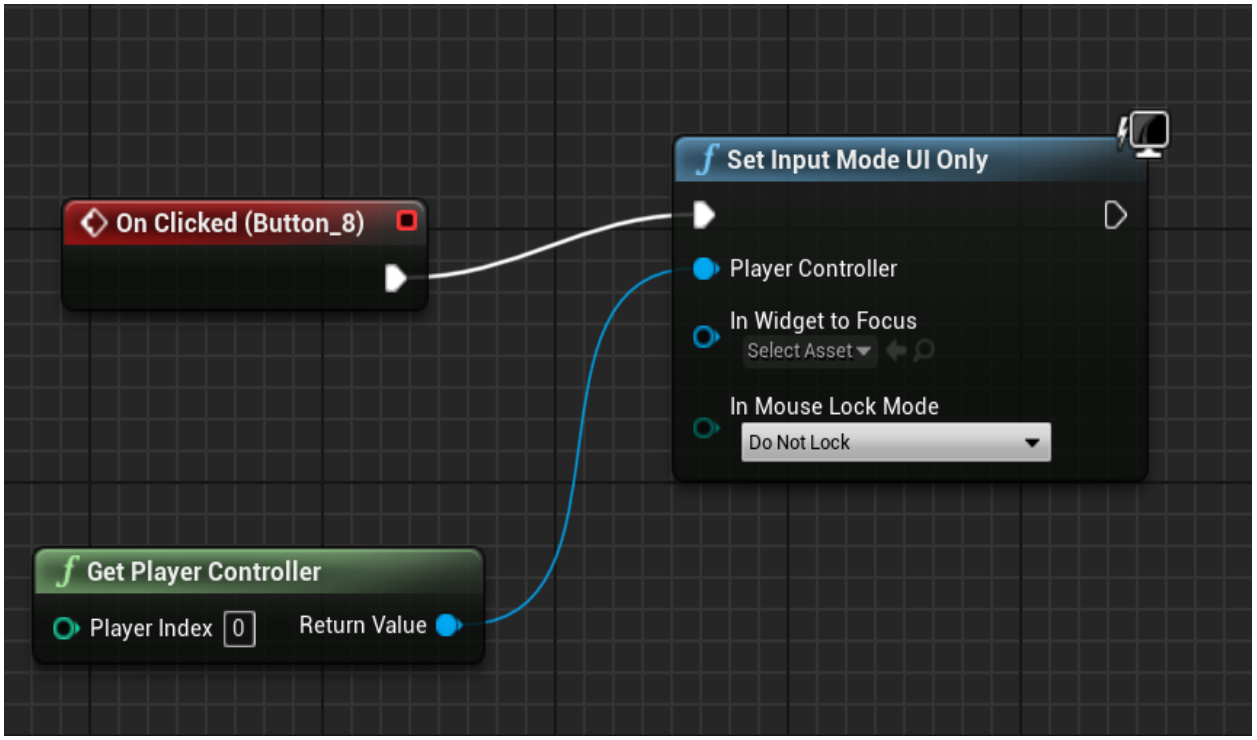

Рисунок 33 – Передача управления от приложения интерфейсу

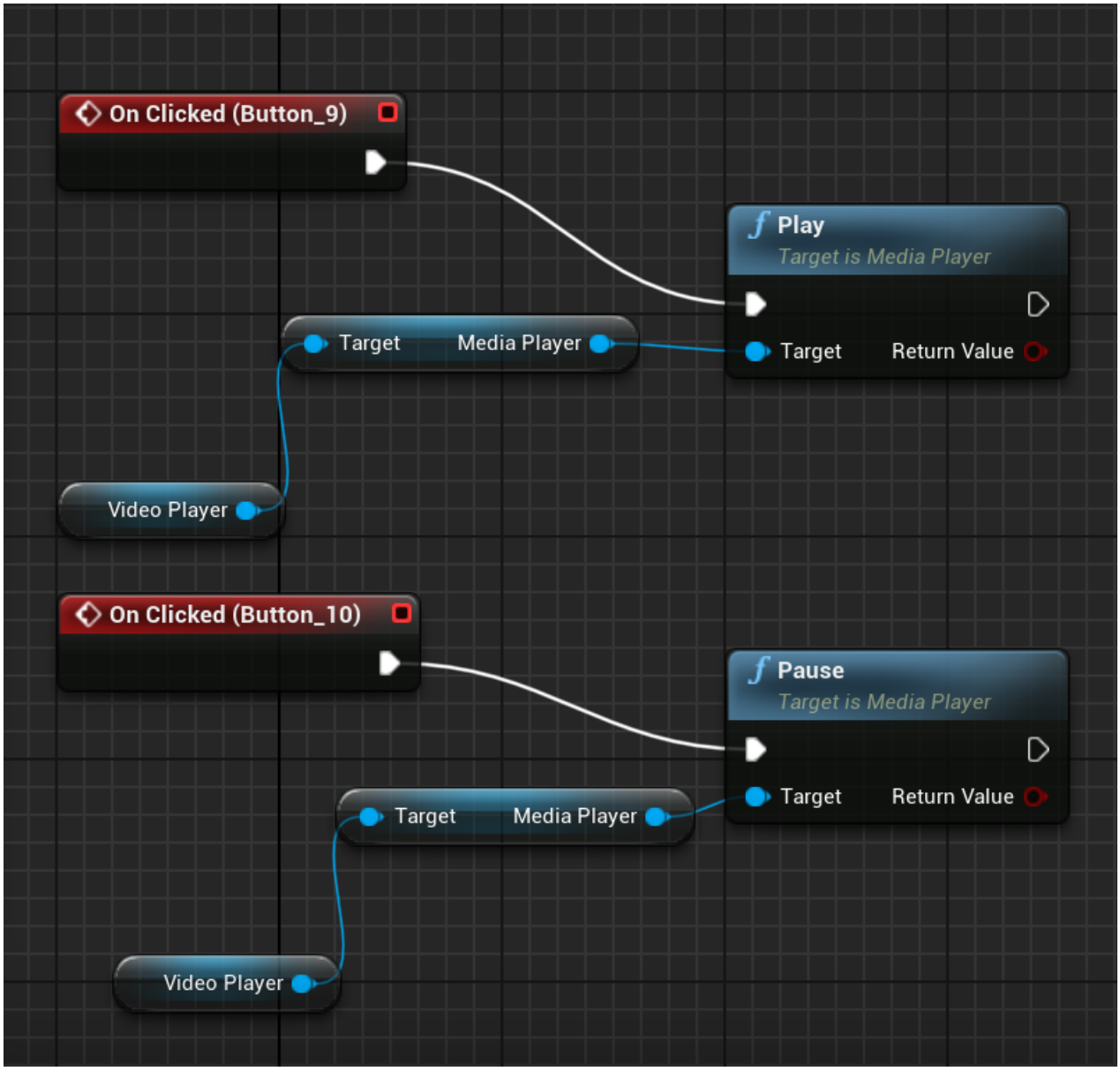

Рисунок 34 – Реализация кнопок воспроизведения и остановки

После выполнения всех вышеуказанных действий мы имеем в своем распоряжении связанные между собой игровую сцену, на которую можно расположить 3D-объект, а также на ней расположен экран с воспроизводимым видео, и пользовательский интерфейс, через который и происходит работа с видео и моделью на нем. Описанный метод воспроизведения видео из пользовательского интерфейса имеет претензию на уникальность.

Еще одной уникальной разработкой в данном видеоредакторе можно считать функционал записи видеоряда. Когда все элементы собраны воедино, а

именно открыто окно видеоредактора, загружено видео и 3D-модель, необходимо соединить это вместе в один медиафайл. За данную функцию отвечает разработанная мной функция записи `INeedRecord`. В заголовочном файле (листинг 3) объявлены методы начала и остановки записи и созданы функции вызова данных методов из Blueprint алгоритмов. В исполняемом файле (листинг 4) прописаны обработчики данных методов.

Листинг 3 – Код заголовочного файла функции записи

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include "Recorder/TakeRecorderPanel.h"
#include "Recorder/TakeRecorderBlueprintLibrary.h"
#include "INeedRecord.generated.h"

UCLASS()
class INEEDRECORDDEMO_API AINeedRecord : public AActor
{
     GENERATED_BODY()

public:
     AINeedRecord();
protected:
     virtual void BeginPlay() override;
public:
     UFUNCTION(BlueprintCalleble)
          void StartRecording();
     UFUNCTION(BlueprintCalleble)
          void StopRecording();
};
```

Листинг 4 – Код исполняемого файла функции записи

```
#include "INeedRecord.h"

using UnrealBuildTool;

public class INeedRecordDemo : ModuleRules
{
     public INeedRecordDemo(ReadOnlyTargetRules Target) :
base(Target)
     {
          PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
          Public DependencyModuleNames.AddRange(new string[]{
"Core", "CoreUObject", "Engine", "InputCore", "TakeRecorder" });
```

```
            PrivateDependencyModuleNames.AddRange(new string[]{ });
      }
};

AINeedRecord::AINeedRecord()
{
      PrimaryActorTick.bCanEverTick = true;
}
void AINeedRecord::BeginPlay()
{
      Super::BeginPlay();
}
void AINeedRecord::Tick(float DeltaTime)
{
      Super::Tick(DeltaTime);
}
void AINeedRecord::StartRecording()
{
      Panel = UTakeRecorderBlueprintLibrary::GetTakeRecorderPanel();
      if (Panel != nullptr)
      {
            Panel->StartRecording();
      }
}
void AINeedRecord::StopRecording()
{
      Panel = UTakeRecorderBlueprintLibrary::GetTakeRecorderPanel();
      if (Panel != nullptr)
      {
            Panel->StopRecording();
      }
}
```

4.5 Создание и настройка пользовательских инструментов

На данный момент функция реализована как загрузка дополнительных плагинов к основным, уже существующим инструментам программы. Поскольку допускать пользователя к коду напрямую – неразумное решение, то дополнение предусматривает создание внешних blueprint’ов.

5 ТЕСТИРОВАНИЕ

Поскольку приложение писалось по блокам, то при отладке необходим блочный метод тестирования.

Блочное тестирование – тестирование методов какого-то класса программы в изоляции от остальной программы. Такое тестирование позволяет определить корректность выполнения каждого блока кода. Одним из основных блоков, который необходимо протестировать – блок работы с видео.

На рисунке 35 представлен результат выполнения уровня `3D-models`. Тест проводился на сетке для персонажей мужского пола. Модель передвигается на клавиши WASD, как и было задумано.

Текстуры на модель применяются корректно.

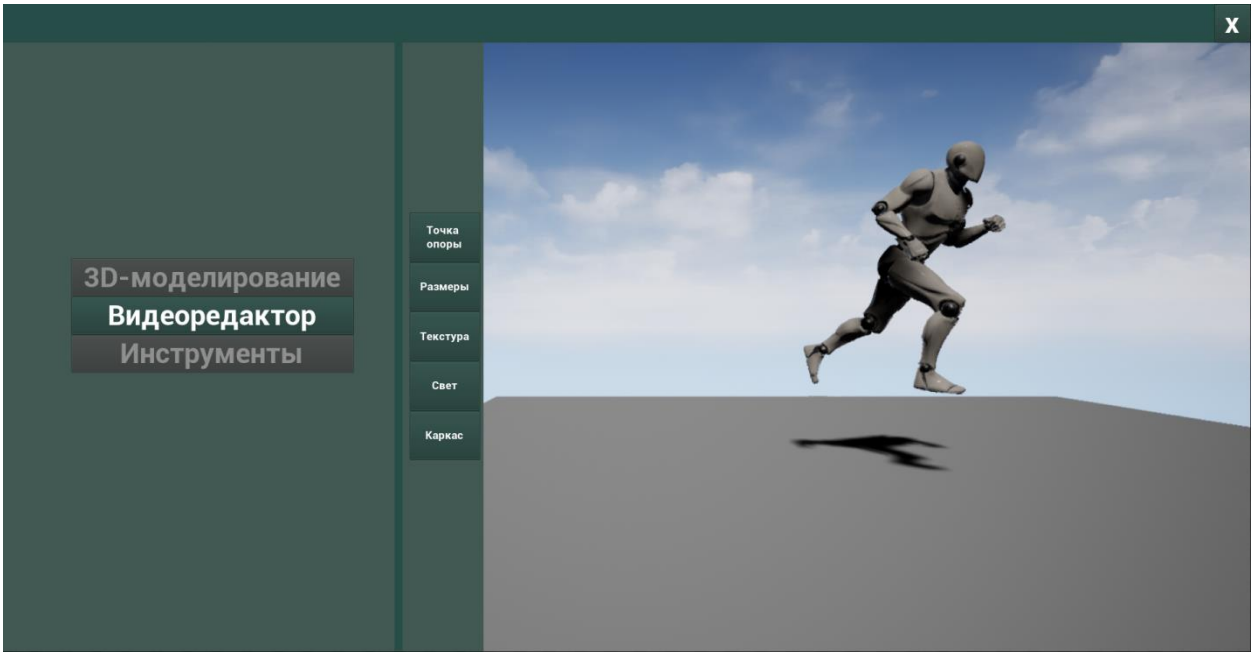

Рисунок 35 – Демонстрация результатов теста анимации движения на уровне 3D-models

На рисунке 36 представлен результат выполнения уровня `Video` на примере некоторого короткого видеоролика формата *.mp4.

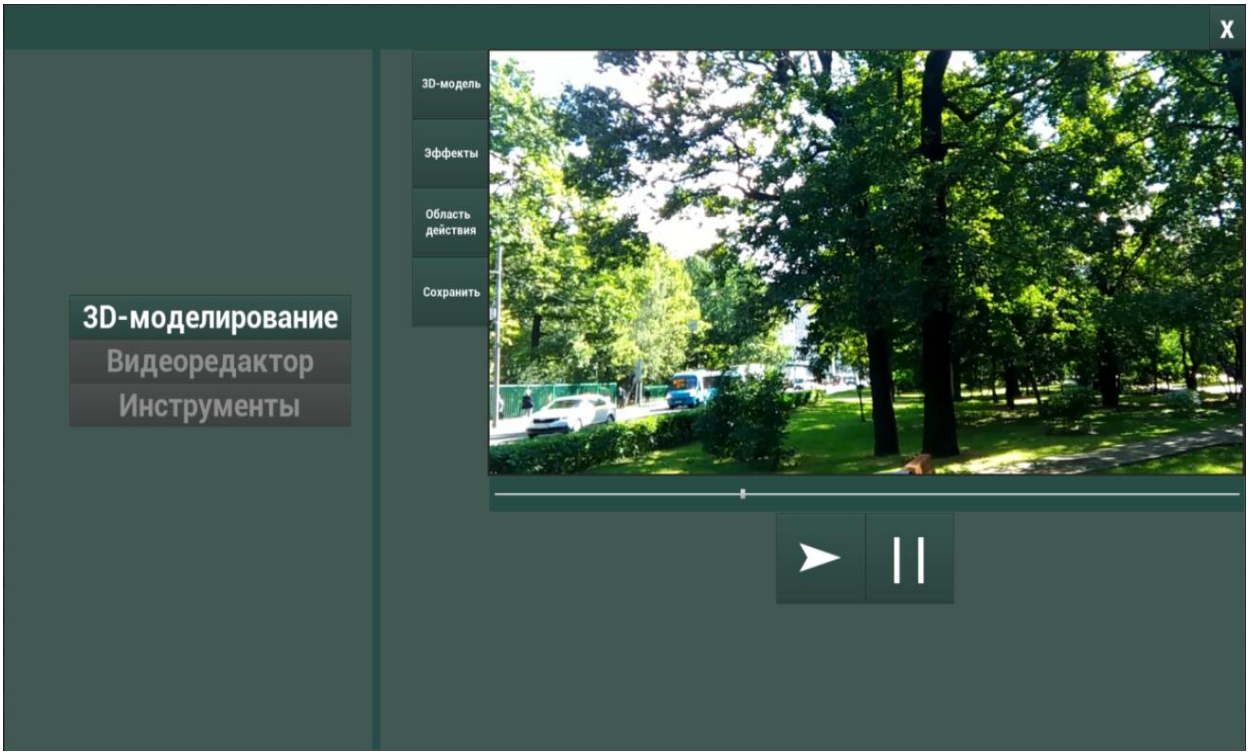

Рисунок 36 – Демонстрация выполнения работы с видео

Файл видеоряда выбирается заранее. Ниже экрана видео можно видеть шкалу управления скоростью прокрутки видео. При крайнем левом положении видео проигрывается в обратную сторону. При переводе шкалы до конца вправо, видео проигрывается в 2 раза быстрее изначальной скорости воспроизведения.

Также представлены кнопки запуска и остановки проигрывания видео, которые функционируют так, как и задумывались.

Перейдем к тестированию интеграции 3D-модели в видео. На рисунке 37 представлен результат добавления модели на видео. Модель находится справа на тротуаре. Для наглядности я подсветил ее контуры красным светом.

Во время тестов возникли неточности в расположении модели. Т.е. исходное видео снято без стабилизации и несколько подвижно, поэтому внешнее отображение нахождения модели в нужной точке не всегда корректно. Поэтому видеоредактор способен обрабатывать видео, положение картинки которого не поворачивается на протяжении всего видеоролика.

Еще одной проблемой стало то, что итоговая картинка на видео ниже по разрешению, чем изначальная. Если на вход подавалось видео с разрешением 1080p (1920x1080), то на выходе качество сократилось до ~480p. Это связано с недостаточной производительностью устройства, на котором проводились тесты.

Рисунок 37 – Демонстрация интеграции 3D-модели в видео

После сборки всех компонентов проведем тест записи видеофрагмента. На рисунке 38 представлены материалы, которые зафиксировал видеоредактор во время записи. Съемка велась на камеру, созданную для фиксации видеоряда, загружаемого пользователем. Также существует дорожка персонажа, т.е. запись его анимации. Данный материал остается конвертировать в видеоформат MP4 из формата данных *.uassert. На данный момент автоматическая конвертация не предусмотрена.

Еще один недостаток такой системы – сильная нагрузка на оперативную память. Данная проблема наблюдается у большинства современных редакторов, поэтому видеоредактор успешно проходит проверку на компиляцию видеоряда.

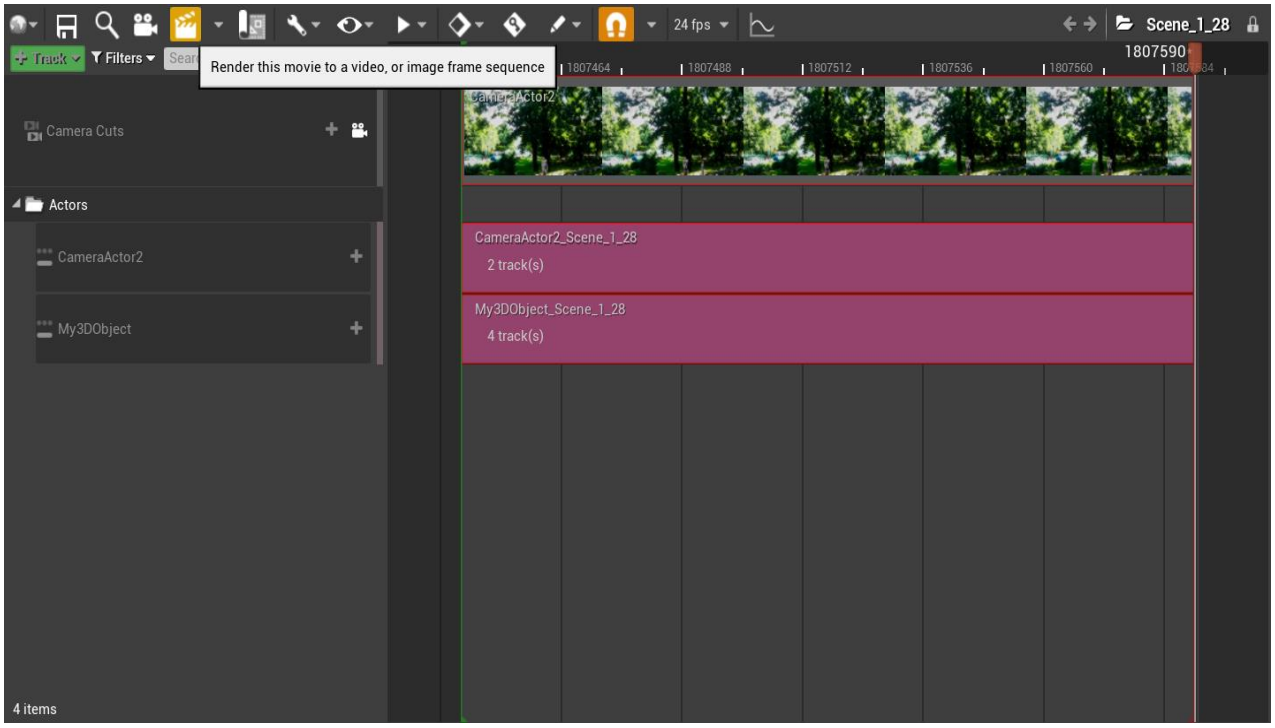

Рисунок 38 – Материалы видео после съемки

На данный момент приложение протестировано только мной, поэтому можно сказать, что оно прошло альфа-тестирование.

ЗАКЛЮЧЕНИЕ

По итогам выпускной квалификационной работы были выполнены следующие виды работ:

– проведен анализ аналогов разрабатываемого приложения, среди которых были рассмотрены как известные видеоредакторы, так и приложения для анимации и создания 3D-моделей;

– детализированы требования к приложению;

– выбраны среда и средства реализации видеоредактора, которыми стали игровой движок Unreal Engine 4 с поддержкой Visual Studio Community IDE;

– спроектирована архитектура приложения, состоящая из отдельных блоков, которые выполняют свои функции, слабо зависящие друг от друга, что позволило выполнить принцип разделения обязанностей в приложении;

– организовано хранилище для базовых 3D-моделей на основе подключаемой к движку архитектурной папки;

– проведен альфа-тест разработанного приложения.

Итоговым результатом стал разработанный и протестированный видеоредактор с функцией интеграции 3D-моделей в видео, в который включаются интерфейсы приложения, хранилище 3D-моделей, архитектура приложения и функциональные блоки, отвечающие за выполнение непосредственных задач видеоредактора и среды для 3D-моделирования.

Данный результат не окончателен. В дальнейшем планируется работа над детальным освещением, продвинутым редактором 3D-моделей, усложнение организации баз данных и вывод пользовательского хранилища в сетевой режим, для возможности дальнейшей монетизации приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Профессиональный видеоредактор Adobe Premier Pro. – Текст. Изображение : электронные // Adobe : [официальный сайт]. – URL: https://www.adobe.com/ru/products/premiere.html (дата обращения: 07.06.2022).

2. Программа для создания анимированной графики и визуальных эффектов. – Текст. Изображение : электронные // Adobe : [официальный сайт]. – URL: https://www.adobe.com/ru/products/aftereffects.html (дата обращения: 07.06.2022).

3. Final Cut Pro – Apple (RU). – Текст. Изображение : электронные // Apple : [официальный сайт]. – URL: https://www.apple.com/ru/final-cut-pro/ (дата обращения: 07.06.2022).

4. Maxon Программа для 3D-моделирования, текстурирования. – Текст. Изображение : электронные // Maxon : [официальный сайт]. – URL: https://www.maxon.net/ru/cinema-4d (дата обращения: 07.06.2022).

5. UnrealEd. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Unreal Engine : [сайт]. – URL: https://docs.unrealengine.com/4.27/en-US/API/Editor/UnrealEd/ (дата обращения: 07.06.2022).

6. Sweeney, T. If You Love Something, Set It Free / Tim Sweeney. – Текст. Изображение : электронные // Unreal Engine : [сайт]. – Март 2015. – URL: https://www.unrealengine.com/en-US/blog/ue4-is-free (дата обращения: 07.06.2022).

7. Setting Up Visual Studio for Unreal Engine. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Unreal Engine : [сайт]. – URL: https://docs.unrealengine.com/4.26/en-US/ProductionPipelines/DevelopmentSetup/VisualStudioSetup/ (дата обращения: 07.06.2022).

8. RAMUS. – Текст. Изображение (неподвижное ; двухмерное) : электронные // RAMUS : [сайт]. – URL: http://ramussoftware.com/ (дата обращения: 07.06.2022)

9. List of file formats. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Wikipedia : [сайт]. – Июнь 2022. – URL: https://en.wikipedia.org/wiki/List_of_file_formats#3D_graphics (дата обращения: 07.06.2022).

10. FBX Content Pipeline. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Unreal Engine : [сайт]. – URL: https://docs.unrealengine.com/5.0/en-US/fbx-content-pipeline/ (дата обращения: 07.06.2022).

11. Texture Import Guide. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Unreal Engine : [сайт]. – URL: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Textures/Importing/ (дата обращения: 07.06.2022).

12. Евтухов, А. Алгоритм Deflate на примере формата / Андрей Евтухов. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Habr : [сайт]. – Январь 2016. – URL: https://habr.com/ru/post/274825/ (дата обращения: 07.06.2022).

13. DirectX Software Development Kit. – Текст. Изображение (неподвижное ; двухмерное): электронные // Microsoft : [официальный сайт]. – URL: https://www.microsoft.com/en-us/download/details.aspx?id=6812 (дата обращения: 07.06.2022).

14. Media Framework Technical Reference. – Текст. Изображение (неподвижное ; двухмерное) : электронные // Unreal Engine : [сайт]. – URL: https://docs.unrealengine.com/4.27/en-

US/WorkingWithMedia/IntegratingMedia/MediaFramework/TechReference/ (дата обращения: 07.06.2022).

15. Миллер, А. AVI VS MP4 различия и сходства между форматами / Аллен Миллер. – Текст. Изображение (неподвижное ; двухмерное) : электронные // FVC : [сайт]. – Июль 2020. – URL: https://www.free-videoconverter.net/ru/tips/avi-vs-mp4/ (дата обращения: 07.06.2022).