МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

СИМУЛЯТОР ПОЛИГОНА ДЛЯ МЕТАНИЯ СНАРЯДОВ НА ОСНОВЕ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Студент _____ 13.06.22 (подпись, дата) Бирюзов Д.А. (ФИО полностью)

Руководитель _____ 13.06.22 (подпись, дата) Бикташев Р.А. (фамилия, инициалы)

Нормоконтролёр _____ 14.06.22 (подпись, дата) Бычков А.С. (фамилия, инициалы)

Рецензент (должность, место работы) _____ 14.06.22 (подпись, дата) Пегорский О.В. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 2.06.22 г. № 13)

Заведующий кафедрой _____ (подпись) М.А. Митрохин (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 17 июня 2022 № 1)

Секретарь ГЭК _____ (подпись) Е.И. Гурин

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«26» 04 2022 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Бирюзов Д.А. гр. 17ВО2 факультета ВТ

Бирюзов Дмитрий Александрович

2.Тема работы: "Симулятор полигона для метания снарядов на основе виртуальной реальности"

Тема утверждена приказом ПГУ № 357 от " 26 " апреля 2022г.

3. Руководитель работы к.т.н., профессор каф. ВТ ПГУ Биташев Р.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.) Разработать приложение на основе устройств виртуальной реальности для симуляции полигона для метания снарядов. Разработать и отладить взаимодействие модулей приложения. Система должна быть представлена в виде единого приложения с целью обеспечения его применения в качестве обучающего материала полигона для метания снарядов.
Приложение должно иметь следующие технические возможности:

- Взаимодействие с объектами;
- Передвижение в пространстве;
- Метание и подрыв снарядов;
- Наличие нескольких типов снарядов;
- Наличие специальных мишеней;
- Наличие обучающих стендов с полезной информацией.

5. Перечень подлежащих разработке вопросов Требуется произвести анализ предметной области: изучить типы устройств виртуальной реальности, виды и различия пакетов для работы с устройствами виртуальной реальности, рассмотреть программное обеспечение для разработки приложения, изучить метод импортирования необходимых для окружения 3D моделей

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	20	26.04.2022-29.04.2022	
2	Анализ требований	15	30.04.2022-01.05.2022	
3	Выбор технологий, применяемых при проектировании и разработке	15	02.05.2022-03.05.2022	
4	Разработка симулятора полигона для метания снарядов на основе виртуальной реальности	150	04.05.2022-17.05.2022	
5	Оптимизация и исправление ошибок	40	18.05.2022-21.05.2022	
6	Оформление пояснительной записки	100	22.05.2022-14.06.2022	
	Общий объем часов	340		

Дата выдачи задания "26" апреля 2022г.

Руководитель работы _____ 24.04.22
(подпись, дата)

Задание к исполнению принял студент _____ 24.04.22
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 16.06.2022
(подпись, дата)

Реферат

Пояснительная записка содержит 118 листов, 50 рисунков, 5 таблиц, 7 источников, 6 приложений.

Объектом исследования является симулятор полигона для метания снарядов на основе виртуальной реальности.

Цель работы – разработать симулятор полигона для метания снарядов на основе виртуальной реальности.

Разработка велась на движке *Unreal Engine* 4 с применением системы Blueprint.

В процессе разработки был проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов.

В результате проделанной работы был разработан Симулятор полигона для метания снарядов на основе виртуальной реальности с использованием устройства виртуальной реальности *Oculus Rift S*.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ, а также рассмотрено воздействие работы с ПК и шлемом виртуальной реальности на зрение человека.

					ПГУ 1.09.05.01.01.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Бирюзов Д.А.		13.06.22	Симулятор полигона для метания снарядов на основе виртуальной реальности Пояснительная записка	Лит.	Лист 2 / Листов 118
Провер.		Бикташев Р.А.		13.06.22			
Н. Контр.		Бычков А.С.		13.06.22		ФВТ, каф. ВТ гр. 17ВО2	
Реценз.		Пегерский		14.06.			
Утверд.		Митрохин М.А.		20.06.22			

Содержание

Перечень сокращений и обозначений 5
Введение 6
1 Симулятор полигона для метания снарядов на основе виртуальной реальности……..……7
1.1 Обзор предметной области 7
1.2 Обзор VR технологий 8
1.2.1 Определение виртуальной реальности 8
1.2.2 Классификация 9
1.2.3 Анализ популярности 10
1.3 Постановка задачи 12
1.4 Проектирование системы 12
1.4.1 Выбор аппаратного обеспечения 12
1.4.2 Выбор программного продукта для разработки 16
1.5 Разработка приложения для устройства виртуальной реальности 19
1.6 Тестирование и отладка 42
2 Технико-экономическое обоснование разработки системы 54
2.1 Введение 54
2.2 Расчёт затрат на разработку программного обеспечения 55
2.3 Расчёт цены разработанной программы 56
2.4 Расчёт капитальных вложений 57
2.5 Расчёт эксплуатационных расходов 58
2.6 Определение расходов на интернет трафик 60
2.7 Технико-экономические показатели разработки 61
3 Безопасность жизнедеятельности. Эргономика рабочего места 62
3.1 Характеристика условий труда программиста 62
3.2 Требования к параметрам воздушной среды. 63
3.3 Требования к уровню шума и вибрации. 64

3.4 Требования к монтажу производственного оборудования..... 64
3.5 Требования к пользованию VR шлемов. 65
3.6 Требования к освещению рабочих мест. 66
3.7 Расчет освещения рабочего места программиста. 67
Заключение. 70
Список использованных источников 71
Приложение А – Диаграмма классов 72
Приложение Б – Схема программы (Триггер снаряда) 74
Приложение В – Схема программы (Инициализация сессии) 76
Приложение Г – Схема программы (Триггер мишени) 78
Приложение Д – Листинг программы 80
Приложение Е – Презентация ……..112

Перечень сокращений и обозначений

VR (Virtual reality) – виртуальная реальность

HMD (Head-Mounted Display) – шлем виртуальной реальности (наголовный дисплей)

API (Application programming interface) – интерфейс приложения для программирования

3D (Three-dimensional) – компьютерная трехмерная графика

SDK (Software development kit) – набор средств разработки

ПО – программное обеспечение

Введение

В наши дни качественная подготовка и усвоение необходимых навыков и умений личным составом является одной из первостепенных целей Вооруженных Сил. С ростом экономических и технических параметров страны появляется необходимость совершенствования не только технологической базы войск, но и методов обучения военнослужащих.

Современная подготовка солдат представляет собой дорогой и небезопасный процесс, и чем неопытнее боец – тем более больше ресурсов необходимо потратить. Помимо этого, процесс подготовки спортсменов также бывает опасным и очень затратным процессом, особенно на начальных этапах обучения. Для решения данных проблем можно привлечь современные методы, а именно технологию *VR*. Данная работа предлагает использование виртуальной реальности для внедрения новой ступени обучения военнослужащих и спортсменов безопасного обращения с различного рода гранатам и снарядам.

Разработка данного программного обеспечения представляет собой совокупность виртуальной модели реального мира для наибольшего уровня восприятия окружения пользователем и алгоритмов работы объектов этого мира для настройки характера взаимодействия объектов в нем. Поэтому необходимо проанализировать существующее ПО данного типа.

1 Симулятор полигона для метания снарядов на основе виртуальной реальности

1.1 Обзор предметной области

Одним из важнейших разделов по огневой подготовке являются тренировки солдат метанию ручных осколочных и специальных гранат из различных положений по целям различной сложности. Участие ВС РФ в боевых действиях первой и второй Чеченских кампаниях выявило несколько существенных недостатков в подготовке военнослужащих в данном вопросе.

Обучение метанию ручных гранат подразделяется на три основных последовательных этапа, к которым относятся:

- изучение материальной базы гранаты и ее боевых качеств;
- изучение требований безопасности и правил обращения с гранатами;
- изучение и освоение приемов метания гранат из различных положений и по целям различной сложности [1].

Начальные сведения о материальной части гранат и их боевых качествах солдаты получают на занятиях по тактической подготовке в части ознакомления с образцами вооружения и военной техники, состоящими на вооружении. Также военнослужащие знакомятся и с макетами реальных прототипов различных осколочных и специальных гранат, приемами и правилами метания, их свойствами.

Реальный физический полигон для обучения метанию гранат представляет собой совокупность сил и средств для демонстрации, обучения обращению с различными типами гранат и поддержанию на высоком уровне бросковых навыков всех военнослужащих.

Гранаты обладают огромным поражающим эффектом, даже учебные версии при неумелом обращении могут представлять опасность. Различные типы гранат требуют различного обращения. При метании боевых гранат на

занятиях и учениях необходимо соблюдать меры безопасности, исключающие поражение метающего и его соседей [2].

Таким образом, выстраивается проблема дороговизны создания обучающих площадок, полигонов, необходимость оборудования аудиторий для первоначального ознакомления, проблема невозможности полного обеспечения их безопасности, а также ограниченность использования в следствии большой поражающей способности гранат. Решением в данной работе предлагается использование *VR* технологий, которые не только будут более экономичными и безопасными, но и могут стать подготовительным этапом в тренировках людей перед применением реальных снарядов, а также в поддержании навыков броска на высоком уровне. Но сначала следует рассмотреть сами *VR* технологии, их особенности и эффективность применения.

1.2 Обзор *VR* технологий

1.2.1 Определение виртуальной реальности

Виртуальная реальность – это совокупность технического и программного обеспечения, создающая виртуальный мир, передающийся человеку через его зрение, осязание и слух. Объединение и комплексное воздействие этих факторов на чувства человека и определяет собой интерактивность виртуального мира. Пространство в этом мире представляет собой созданный с помощью компьютерной графики вселенную, в которую пользователь и погружается с помощью специальных контроллеров. Особенности виртуальной реальности разнообразны, но выделяются основные критерии, с помощью которых и создается виртуальный мир:

- правдоподобность – пользователь не сомневается в естественности вокруг происходящего;

- интерактивность – существование возможности взаимодействия с окружающим пользователя миром;

- возможность изучения – созданный мир должен быть проработанным и детализированным;

- эффект присутствия – пользователь должен чувствовать свою вовлеченность в события, происходящие вокруг него;

- бесперебойность – гарантирование работы без перебоев и подзагрузок.

Виртуальная реальность может с высокой точностью воспроизводить воздействие окружающей среды на пользователя, но это требует создания детальной и правдоподобной компьютерной программы, в которой и будет проходить просчет реакций и свойств интерактивного мира [3].

1.2.2 Классификация

Виртуальная реальность подразделяется на несколько типов, различных по техническим особенностям и типам восприятия пользователем:

- виртуальная реальность с полным погружением. Данный тип включает в себя три фактора: точная симуляция окружающего мира с высокой степенью проработки и детализации, высокопроизводительная ЭВМ, поддерживающая окружающую созданную реальность без прерываний и в реальном времени, дополнительное оборудование с подключением к ЭВМ, обеспечивающее сам эффект погружения и отслеживание действий пользователя контроллерами;

- виртуальная реальность без погружения. Не для каждой задачи необходима реальность с полным погружением. Для данной симуляции требуются качественное изображение, звук и отслеживание контроллеров, обычно транслируемые на широкоформатные экраны. Также под этот тип попадают многие исследовательски и выставочные проекты, археологические реконструкции, демонстрационные работы клиентам. Такие реальности позволяют чувствовать моделируемый мир намного глубже, чем это позволяет традиционная мультимедиа;

- виртуальная реальность с совместной инфраструктурой. Важнейшее

свойство данного типа является создание эффекта полного присутствия. В таких виртуальностях упор сделан на прописывание широкого взаимодействия между множеством пользователей, чего нету в реальностях других типов. Данному типу можно отнести виртуальные вселенные *SecondLife* и *VRchat*. Также программы данного типа позволяют организовывать рабочие и учебные виртуальные пространства, дающие эффект совместной работы. Совмещение полного погружения и социального взаимодействия сейчас является одним из важнейших направлений совершенствования виртуальных реальностей;

- виртуальная реальность на основе интернет технологий. Технология *Virtual Reality Markup Language*, созданная по прототипу *HTML*, позволяет разработчикам использовать новый метод создания виртуальных реальностей. Несмотря на ее несовершенность, возрастающий интерес к социальным сетям на основе виртуальных миров, будущие проекты могут быть основаны не только на социальных взаимодействиях, но и на интернет технологиях;

- дополненная реальность. Дополненная реальность (*AR*), в отличии от виртуальной реальности, является наложением виртуального мира на настоящий, являясь их совокупностью. Так мы получаем воспринимаем оба мира одновременно. Технически дополненная реальность не является виртуальной реальностью, но она имеет те же вопросы, возникающие и при создании обычных виртуальных миров, поэтому их обычно рассматривают в связи друг с другом. К примерам дополненной реальности можно отнести недавно шумевшую на весь мир *PokemonGo* [4].

1.2.3 Анализ популярности

Виртуальная реальность используется во многих сферах жизни человека, таких как:

- развлечения: виртуальная реальность невероятно популярна как игровая среда. Большим спросом пользуются как игры, так и различные

культурные мероприятия, туризм;

- образование: современные интерактивные реальности позволяют создавать среды для тренировок абсолютно любых сфер деятельности. Причем особенно развиты системы, которые в реальности могли бы представлять опасность для обучаемых, такие как операции, вождение техники, вооружение;

- наука: виртуальность позволяет не только облегчить исследования атомного и молекулярного миров, но и проводить различные симуляции и опыты внутри безопасной зоны. Также виртуальная реальность позволяет прикоснуться к науке многим людям, далеких от нее;

- медицина: включает в себя те особенности, которые были описаны выше – обучение новых специалистов, безопасное проведение операций, улучшение навыков медперсонала.

- архитектура: воплощение схем будущих строений и макетов в виртуальной реальности, в полном размере, без трат любых материалов и с возможностью изменить любую часть в любое время, как для демонстрации проекта заказчику, так и для общественного показа;

- армия: виртуальная реальность давно нашла свое место для использования в вооруженных силах различных стран. Программисты создаю различные тренажеры, дающие возможность молодым бойцам тренировать свои умения, не подвергая опасности свое здоровье и не тратя ресурсы на настоящую технику [5].

На основе этих выводов можно считать более чем возможным использование устройств виртуальной реальности для создания симулятора бросков гранат. Так как подобный тип обучения обращаться с вооружением в разы безопаснее и дешевле, чем применение реальных экземпляров. Наиболее подходящим для разработки такого рода приложения является ПО для разработки игр, так как современные игры, в особенности в *VR*, максимально приближены к реальным условиям.

1.3 Постановка задачи

Необходимо разработать программу «Симулятор полигона для метания снарядов на основе виртуальной реальности».

В данной программе будет реализована часть, отвечающая за подготовку и обучение военнослужащих метанию гранат. Возможности программы должны предоставлять взаимодействие с объектами в *VR*, передвижение в пространстве, броски гранат, наличие нескольких типов снарядов с разными особенностями, наличие мишеней, разрушаемых при близлежащем подрыве гранат. Для решения этих задач необходимо разработать и реализовать простой, удобный и понятный пользовательский интерфейс.

Программа должна быть многомодульной: необходимо поделить её на логические модули. Это способствует упрощению добавления в программу новых функций при дальнейшей разработке, а также упростит обнаружение и исправление ошибок при её отладке и тестировании.

Программа должна быть разработана для работы в *ОС Microsoft Windows* 10.

1.4 Проектирование системы

1.4.1 Выбор аппаратного обеспечения

В наши дни самыми доступными и распространенными гаджетами виртуальной реальности являются шлемы и очки виртуальной реальности. Они различаются между собой структурой устройства от компаний – производителей. Громоздкие устройства на всю голову называют шлемами, а компактные – очками. Однако все они имеют единое обозначение – *Head mounted display (HMD)* – наголовные дисплеи. Самыми распространенными являются:

- *Valve Index*: на сегодняшний день является самым передовым шлемом виртуальной реальности, но и самым дорогим. Это устройство объединяет в себе высочайшее качество картинки, производительность и комфорт.

Преимущества: в отличии от многих аналогичных устройств, *Valve Index* имеет в себе функцию отслеживания движения пальцев, широкий угол обзора, хорошая аудиосистема, широкий простор настройки девайса под каждого индивидуального пользователя. *Valve Index* позволяет настроить калибровку линз, что обеспечивает лучшую картинку и угол обзора персонально для каждого. Также присутствует регулировка межзрачкового расстояния в диапазоне от 58 до 70 мм.

Недостатки: в первую очередь потребителей отталкивает цена комплекта, превышающая стоимость аналогичных моделей других производителей в полтора – два раза. Также система отслеживания движений пользователя не встроена в шлем и требует установки двух отслеживающих баз в комнате. Они довольно трудны в установке и требуют источники электропитания. Также стоит отметить о функции сквозного обзора шлема, которая дает возможность видеть настоящую реальность не снимая шлема. Данная функция работает не столь быстро, как аналогичные функции других шлемов;

- *HP Reverb G2*: в то время как *Index* обеспечивает пользователю универсальную производительность, новейший *HP Reverb G2* — гарнитура с самым большим разрешением. Она отлично подойдёт пользователям, которые предпочитают путешествовать по виртуальным мирам сидя, так как система отслеживания движений у этого устройства не самая лучшая.

Преимущества: когда дело доходит до качества картинки, *HP Reverb G2* с уверенностью забирает корону первенства. Устройство способно обеспечить максимально красивую визуализацию в своём классе благодаря разрешению 2160 × 2160. Также шлем имеет те же наушники, что и *Valve Index*. А благодаря встроенной системе трекинга устройство способно

отслеживать движения пользователя без необходимости использовать внешние базы отслеживания.

Недостатки: хоть комплект устройства и имеет удобные контроллеры движений, их задержка работы достаточно велика и неточна по сравнению с аналогичными моделями. Они отлично работают с большинством приложений, но с более активными и соревновательными тренажерами они уже не справятся. По размеру поля зрения шлем похож на свои аналоги, хоть и проигрывает *Valve Index*. Сквозной обзор шлема выполнен в виде фонаря, что выполнен хуже многих аналогичных моделей;

- *Oculus Quest* 2: характеризует себя как автономный шлем виртуальной реальности, однако он имеет возможность подключить специальный кабель Oculus Link, который превращает его в привычный стационарный ПК шлем. Таким образом для него доступны сразу две библиотеки приложений - фирменный *Oculus Store* и популярный *SteamVR*.

Преимущества: шлем имеет встроенные камеры трекинга головы и контроллеров, что облегчает настройку к использованию. Автономность *Oculus Quest* дает возможность пользоваться им в абсолютно любом месте и без ПК. Обладает хорошей картинкой, регулировкой межзрачкового расстояния, контроллерами в комплекте. Производитель постоянно обновляет прошивку шлема, что делает работу еще стабильнее и удобнее. Цена на устройство является одной из самых низких среди шлемов высокого качества.

Недостатки: встроенные наушники неплохи, но по сравнению с другими шлемами отстает. Для пользования шлемом обязательна регистрация в фирменном магазине Oculus. Центр тяжести шлема немного неверно расположен, исправляется покупкой отдельного крепления [6];

- Гарнитуры *Windows Mixed Reality*: Помимо вышеперечисленных шлемов, которые являются лучшими моделями для массового потребителя, существует множество систем виртуальной реальности, созданные чисто под

производственные и рабочие цели. Они производятся *Microsoft* на базе *Windows Mixed Reality*, и представляют собой очки смешанной реальности, совмещая в себе технологии виртуальной и дополненной реальностей. Одними из линейки этих устройств являются очки *Microsoft HoloLens* 2. Они позиционируются как коммерческий вариант, поэтому заказать их можно только по заявкам разработчиков из некоторых стран. Сейчас их используют ученые из США, Канады, вооруженные силы этих стран. Среди партнеров корпорации *Microsoft*, которые создают свои очки смешанной реальности на основе платформы *Windows Mixed Reality, - Acer, Dell, HP, Lenovo* и *Samsung*. Устройства оснащены встроенными камерами отслеживания движений, контроллерами движений. Присутствует совместимость приложений из фирменного *Microsoft Store* и популярного *SteamVR*. Очки являются невероятно компактными, в качестве контроллеров движений могут выступать руки пользователя, не требуют подключения к ПК [7].

В таблице 1 сведены параметры различных *VR*-шлемов

Таблица 1 – Характеристики популярных *VR*-шлемов

	Oculus Quest 2	*Valve Index*	*HP Reverb G2*	*Microsoft HoloLens 2*
Общее разрешение	3664 x 1920	2880 x 1600	4320 x 2160	5120 x 1440
Технология экрана	LCD	LCD	LCD	OLCD
Частота обновления	90 Гц	144 Гц	90 Гц	144 Гц
Угол обзора	100 Градусов	130 Градусов	114 Градусов	140 Градусов
Вес	503 г	800 г	550 г	579 г
Цена производителя	299 долларов	999 долларов	859 долларов	3599 долларов

На основании всех вышеперечисленных плюсов и минусов для использования в данной работе было выбрано устройство *Oculus*, так как рассматривались простота обращения, высокие технические показатели и совместимость с большинством инструментов разработки приложений.

1.4.2 Выбор программного продукта для разработки

Наиболее популярное ПО для разработки *VR*-приложений:

- *Unity*;

- *CryEngine*;

- *Unreal Engine*.

Unity – игровой движок, получивший наибольшую популярность среди начинающих разработчиков. Его достаточно просто освоить, ведь этому помогают множество руководств и видеоуроков. Основным языком программирования является *C#*, но также присутствует система блокового программирования, позволяющая создавать приложения без написания кода. Имеет встроенную библиотеку готовых материалов, моделей и других ресурсов, что ускоряет процесс разработки.

Универсальность инструментария и является основной проблемой *Unity*. Если необходимо создать что-то нетривиальное – возникает необходимость модификации движка под индивидуальные нужды программы.

Достоинства:

- быстрое и удобное прототипирование;

- бесплатная версия для начинающих разработчиков и студентов;

- совместимость с любой платформой;

- разработка как небольших, так и ААА-проектов;

- во встроенной библиотеке множество готовых материалов.

Недостатки:

- медленная работа: например, алгоритм сжатия *LZW* занимает 180 секунд в многопоточном режиме;

- ограниченный инструментарий;

- закрытый исходный код;

- нестабильность редактора и отладчика.

CryEngine является лучшим движком при выборе наиболее фотореалистичной графики. Несмотря на свой давний релиз, он может создавать картинку, не уступающей современным инструментам. *CryEngine* дает возможность создавать миры с попиксельным освещением в реальном времени, картами отражений и собственной системой трассировки лучей.

Еще одна особенность игрового движка *CryEngine* 3 — упрощенное создание продвинутого *AI* внутри игр за счет комбинации *XML* и *LUA* скриптов.

Достоинства:

- множество инструментов для работы с графикой;

- гибкая ценовая политика: базовая версия бесплатна;

- поддержка *DirectX 12, Vulkan API, VR*;

- относительно простая настройка искусственного интеллекта по сравнению с остальными движками;

- в комплекте с базовой версией идет *GameSDK* — полноценный шутер, который адаптируется; под ваши нужды. Современный аналог *Source Engine*.

Недостатки:

- высокий порог вхождения;

- медленная работа техподдержки;

- отсутствие поддержки разработки приложений для мобильных устройств и портативных консолей;

- ограничения при разработке многопользовательских проектов.

Unreal Engine — передовой инструмент разработки приложений. Имеет довольно высокий порог вхождения, паже при наличии системы нодового проектирования приложений *Blueprint*. Все элементы *UE*4 распределяются на категории, имеющие настраиваемые классы и определяемые ими

характеристики. Основными категориями являются актеры (действующие объекты), пешки (объекты, управляемые пользователем и искусственным интеллектом) и мир (декоративные объекты пространства). Движок базируется на языке программирования *C++*: быстрый, но сложный в освоении.

Для обучения пользования движком существует множество уроков как от самих разработчиков, так и от пользователей сообщества. Несмотря на всю сложность, создать собственное приложение можно даже без особых умений в программировании. Лучше всего возможности игрового движка *Unreal Engine* 4 раскрываются при разработке AAA-проектов.

Достоинства:

- широкий набор инструментов;
- удобный интерфейс;
- доступность обширного количества обучающих роликов;
- бесплатна;
- возможность настройки интерфейса на любой вкус.

Недостатки:

- высокий порог вхождения;
- для комфортного использования требуется мощная конфигурация ПК;
- дорогие материалы во встроенном магазине.

На основании вышеописанных преимуществ и недостатков, а также выбранных устройства *Oculus* с интерфейсом *OpenVR,* в качестве платформы для разработки был выбран движок *Unreal Engine* 4, так как по опыту разработок, данная программная платформа лучше всего взаимодействует с интерфейсом *OpenVR* и стандартно использует встроенную в *OpenVR* платформу *SteamVR*, позволяющую использовать устройство на всех ОС с установленным *Steam*. А наличие библиотеки открытого и бесплатного контента облегчает наполнение визуальной составляющей приложения. *Unreal Engine* 4 в совокупности с *Oculus* выделяется на фоне конкурентов по

качеству, удобству, скорости разработки и оптимизации, и бесплатен в использовании.

1.5 Разработка приложения для устройств виртуальной реальности

Приложение симулятора полигона для метания снарядов на основе виртуальной реальности состоит из совокупности двух частей – визуальной и программной:

- визуальная часть:
    1) 3D модель окружающего мира и объектов в нем;
    2) Текстуры созданного мира;
    3) Области триггеров;
    4) Звуки и эффекты;
- программная часть:
    1) Управление;
    2) Скрипты;
    3) Интерфейс.

Для создания нового проекта в *Unreal Engine* 4 необходимо выполнить следующие действия:

- на стартовом окне приложения выбрать вкладку *New project categories* и нажать кнопку *Games* (рисунок 1);

- в открывшемся окне необходимо выбрать проект *Virtual Reality* (рисунок 2);

- далее нужно выбрать первоначальные настройки проекта, а именно: активация системы *Blueprints* – системы визуального программирования движка *Unreal Engine*, уровень графики *Maximum quality* для лучшего восприятия картинки, устройство воспроизведения *Desktop/Console*, наличие стартового контента *With starter content*, путь расположения проекта (рисунок 3).

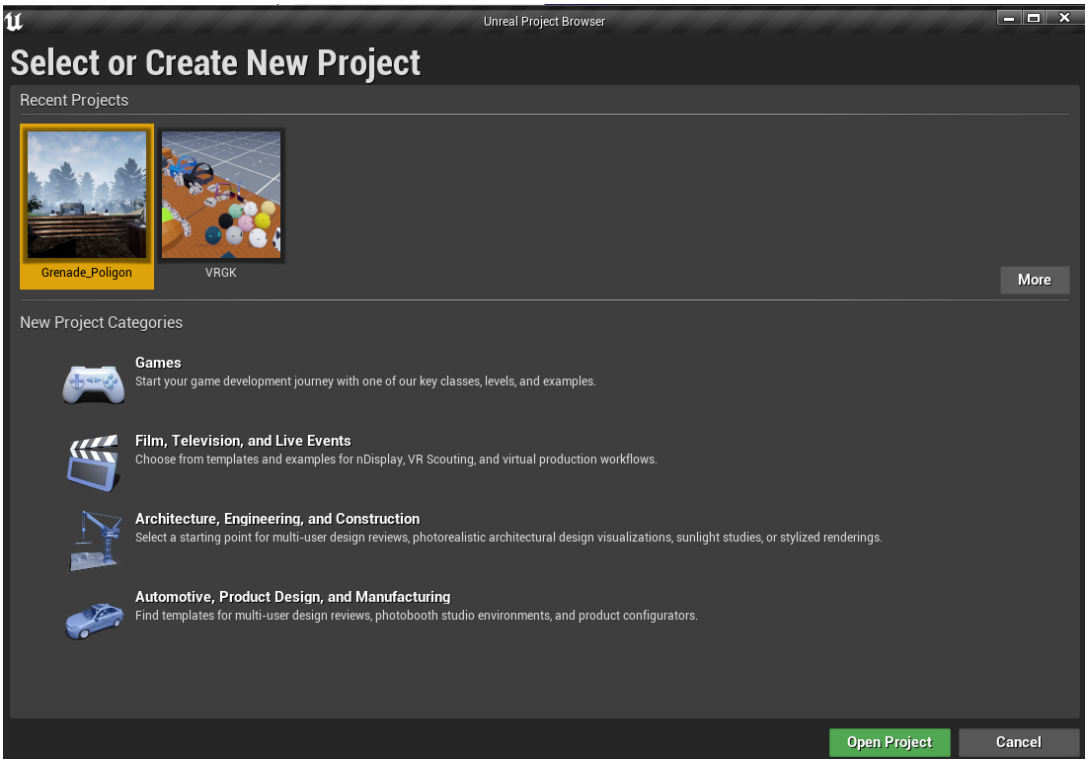


Рисунок 1 – Выбор создания нового приложения

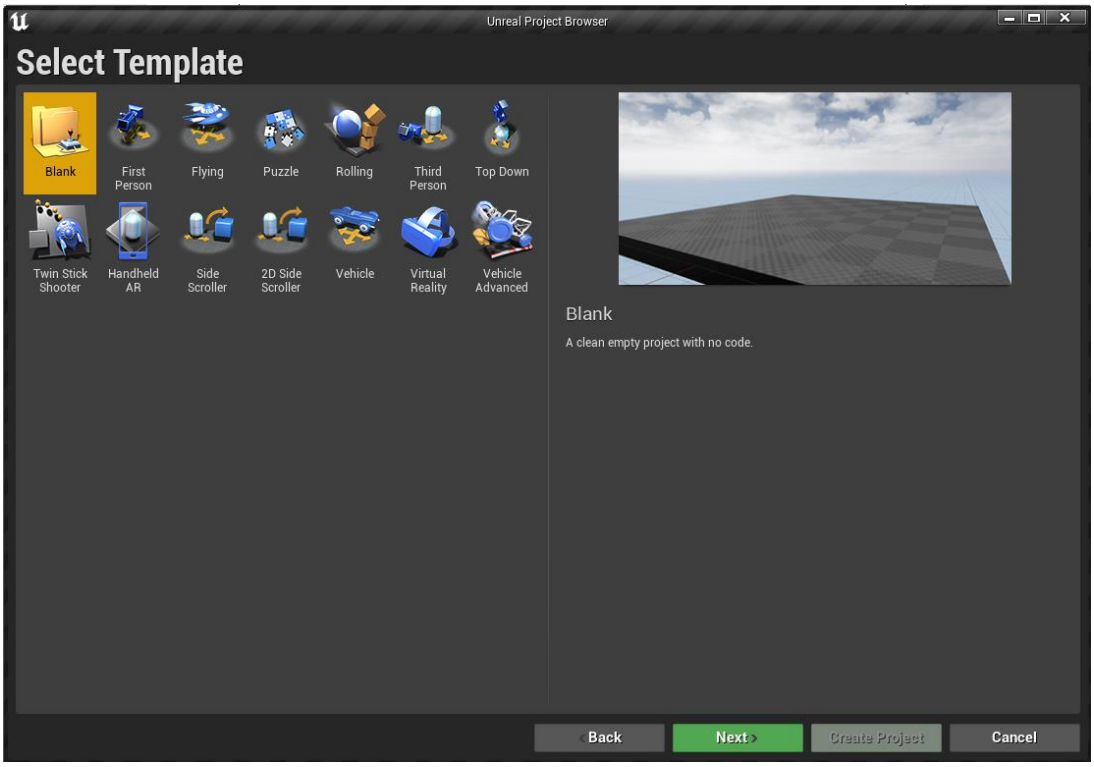


Рисунок 2 – Выбор проекта на основе *VR*

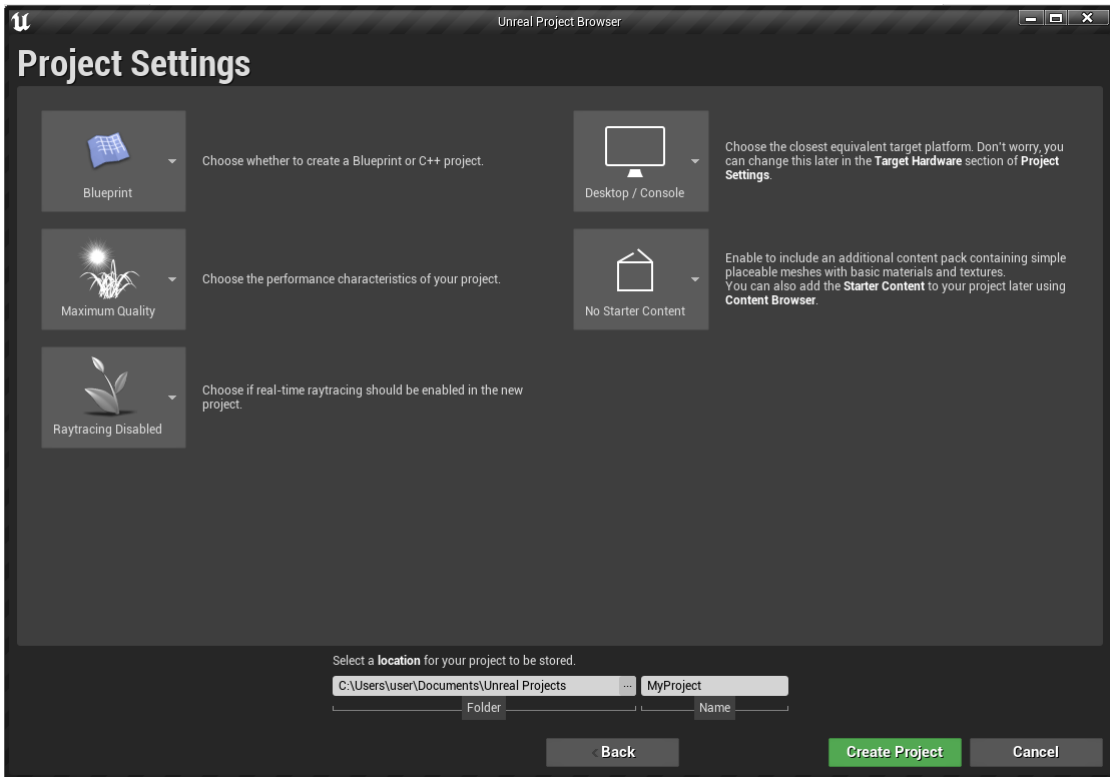


Рисунок 3 – Стартовые настройки проекта

Модели окружения, текстуры, эффекты и звуки можно импортировать из встроенного магазина *Epic Games Store*, из которого и загружается *Unreal engine* 4. Он содержит в себе множество сделанных профессионалами высококачественных материалов для создания любого типа контента. Множество этого материала является бесплатным.

Для импорта материалов необходимо:

- в окне проекта в верхней панели выбрать вкладку магазина *Marketplace* (рисунок 4);

- открывшееся окно магазина имеет широкий инструментарий поиска по нему, а также вкладку *Free* (бесплатные материалы) (рисунки 5, 6);

- выбрав необходимый материал, необходимо добавить его к проекту, нажав на соответствующую кнопку (рисунок 7).

Добавление целых наборов контента позволяет быстро и качественно получить необходимый уровень графики создаваемого проекта.

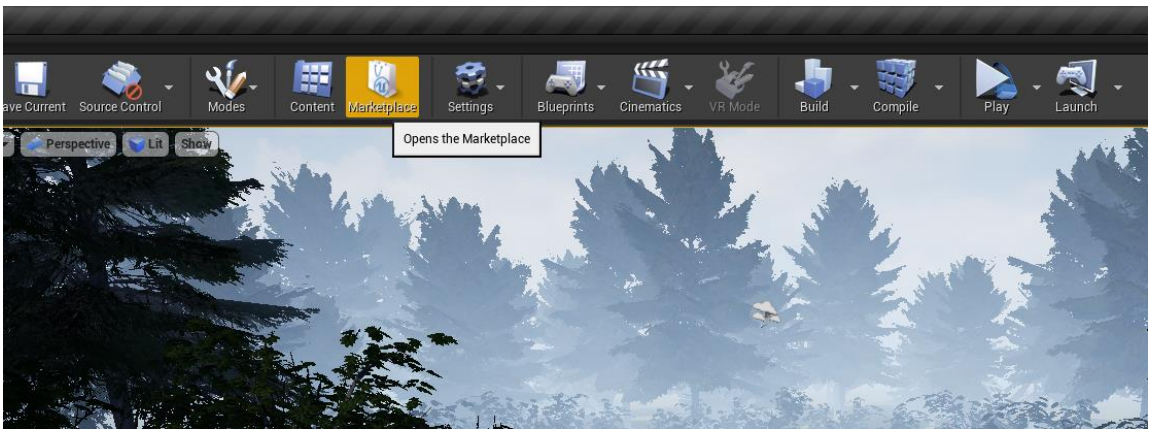


Рисунок 4 – Вкладка *Marketplace*

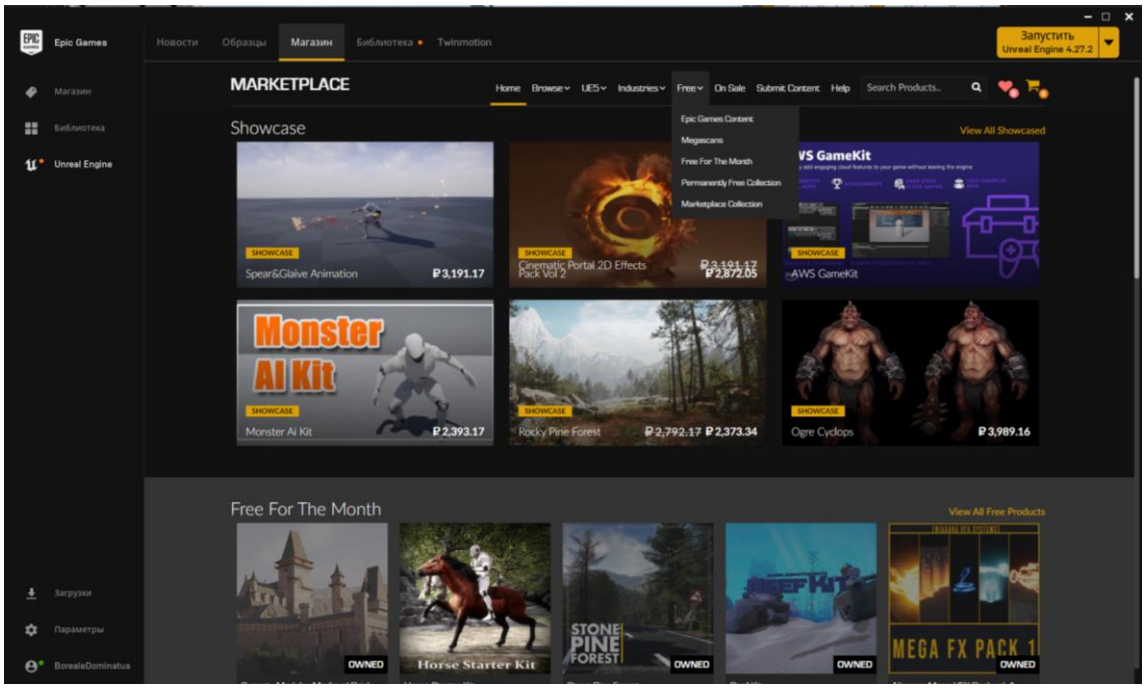


Рисунок 5 – Окно магазина

После добавления всех дополнений окно приложения разработки будет выглядеть следующим образом: сверху – кнопки управления состоянием проекта, его компиляции, запуска и сохранения; справа – окно с находящимися на сцене объектами и списком свойств этих объектов; снизу – иерархия всех материалов и блюпринтов, добавленных и созданных для проекта; слева – окно создания новых объектов сцены (рисунок 8).

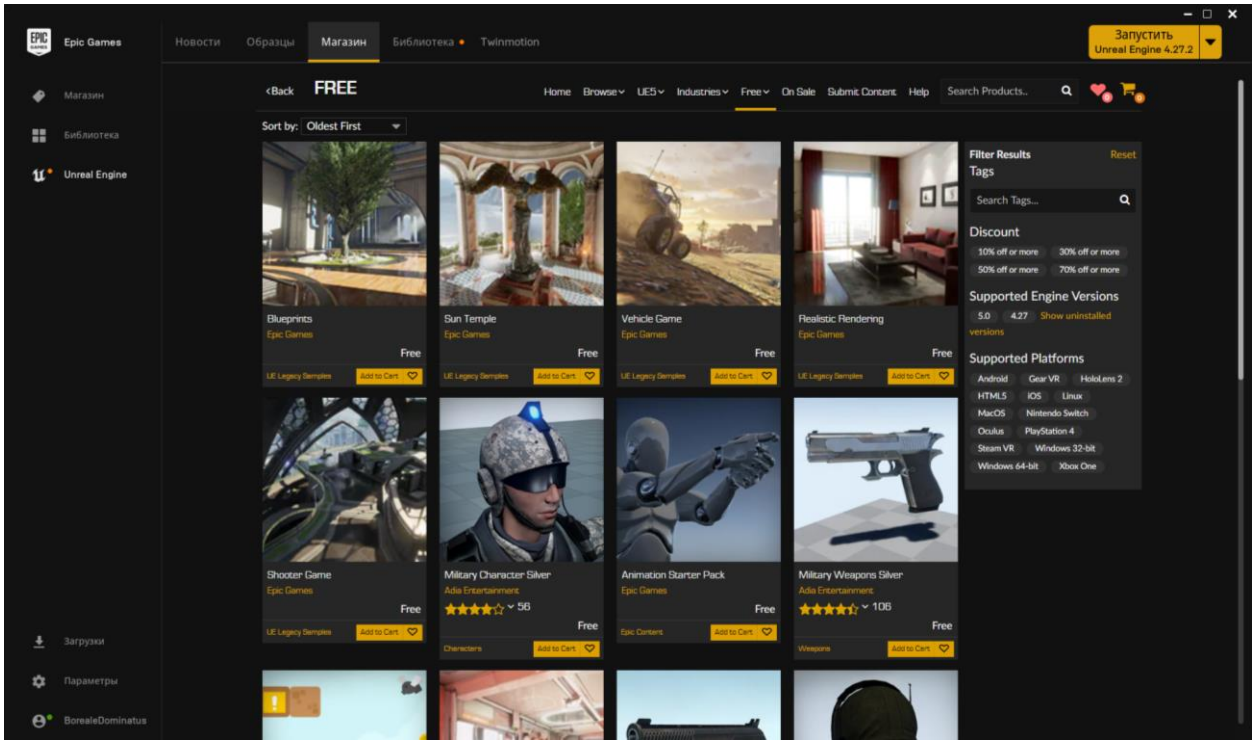


Рисунок 6 – Окно списка бесплатных материалов

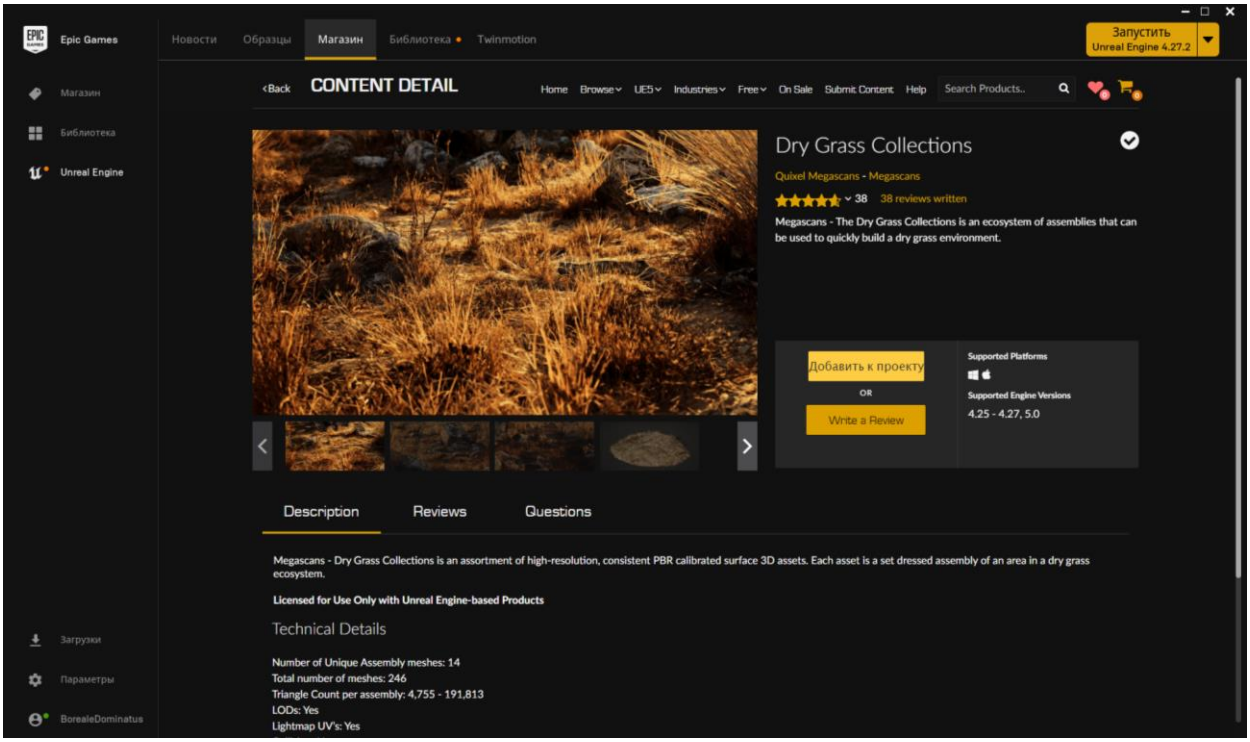


Рисунок 7 – Добавление материала к проекту

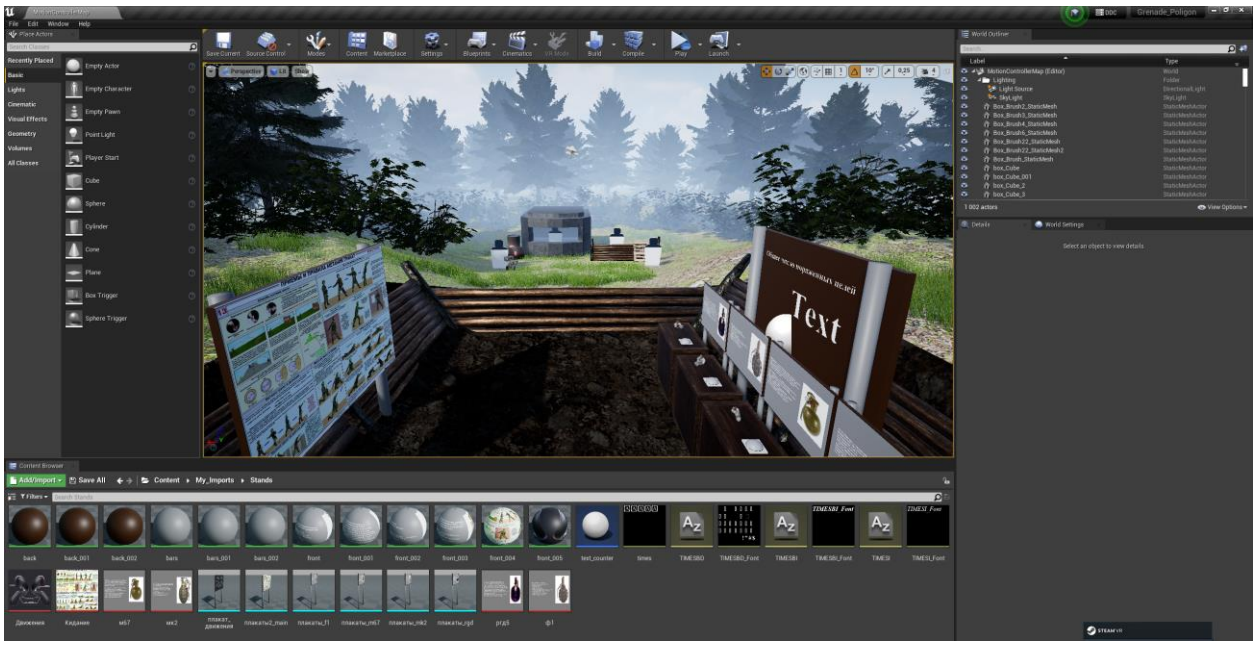


Рисунок 8 – Главное окно разработки

Каждый объект сцены является целом комплексом связанных между собой моделей, текстур, звуков и скриптов (блюпринтов). Редактирование этих объектов происходит при открывании окна *BlueprintEditor* (рисунок 9).

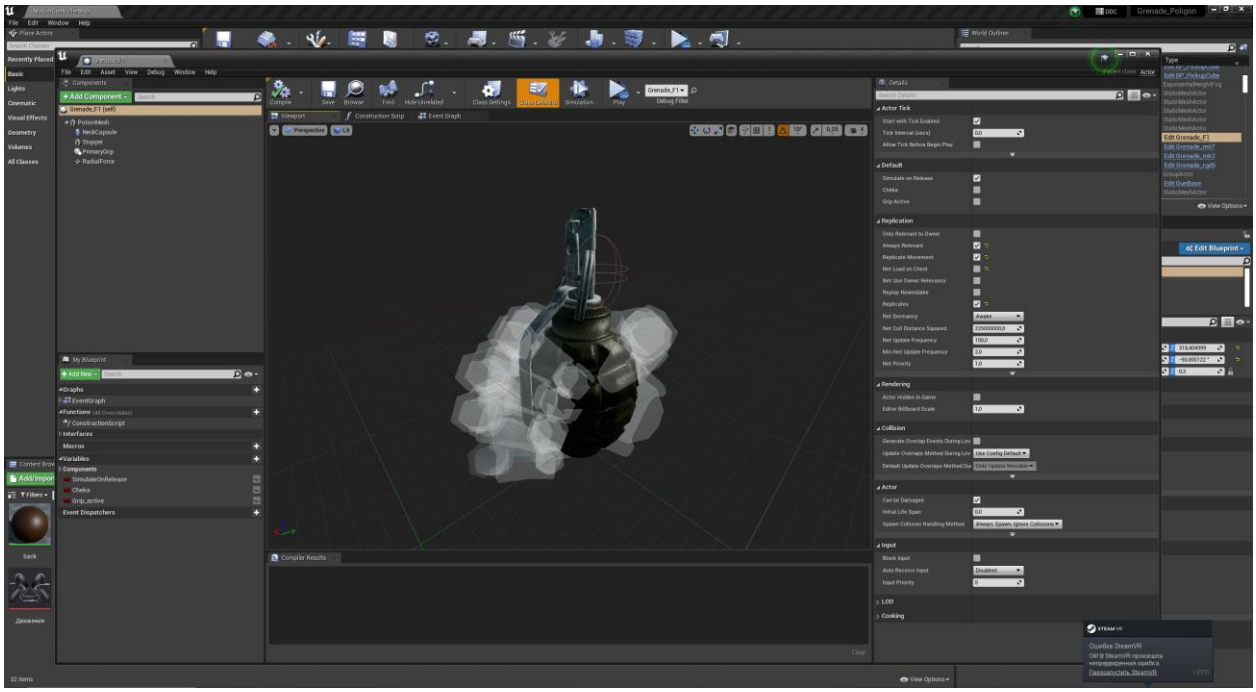


Рисунок 9 – Окно редактирования объектов

Данное окно позволяет редакцию и добавление визуальной части объекта. На левой стороне находится список уже находящихся на сцене объектов, так и меню добавления новых; сверху находится меню управления блюпринтом – его компиляция, сохранение, отладка; справа находится окно редактирования свойств каждого элемента объекта; центральное окно является визуальным представлением редактируемого объекта.

Выбор вкладки *EventGraph* позволяет открыть окно редактирования блюпринта выбранного объекта (Рисунок 10). Эта система позволяет использовать весь потенциал традиционного программирования, но вместо кода разработчик создает сети стандартных и создаваемых функций (нодов). Эти сети нодов движок при компиляции преобразует в код C++. В любое время можно переключиться в меню стандартного программирования, однако использование нодов дают множество преимуществ:

- стандартные ноды имеют исчерпывающее описание и просты в использовании (рисунок 11);

- поиск нодов вынесен в отдельное меню со множеством вкладок и поиском (рисунок 12);

- все созданные пользователем функции и переменные вынесены в отдельное окно, позволяя быстро редактировать их (рисунок 13);

- наглядное представление всего скрипта дает высокий уровень простоты понимания сути всего кода (рисунок 14). Есть возможность комментирования как отдельных нодов, так и целых групп (рисунок 15). При наличии ошибок в коде компилятор подсвечивает их (рисунок 16);

- режим симуляции позволяет просматривать работу скрипта в реальном времени, выявляя ошибки и места для редактирования (рисунок 17).

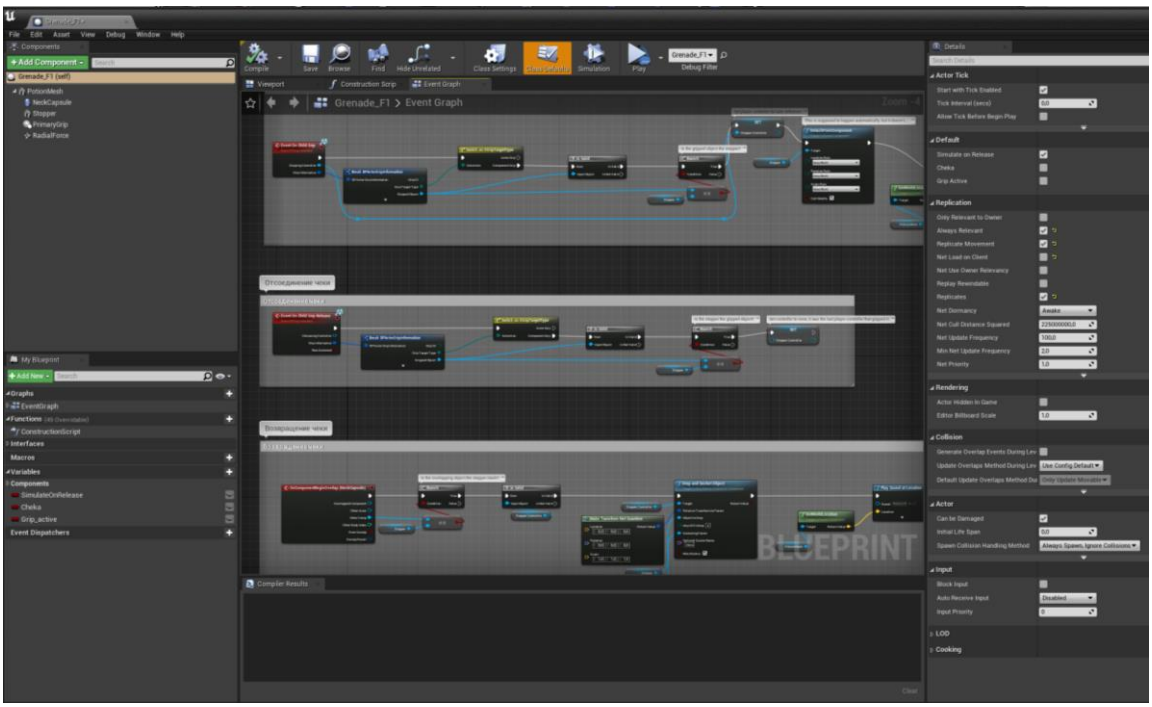


Рисунок 10 – Окно редактирования блюпринта

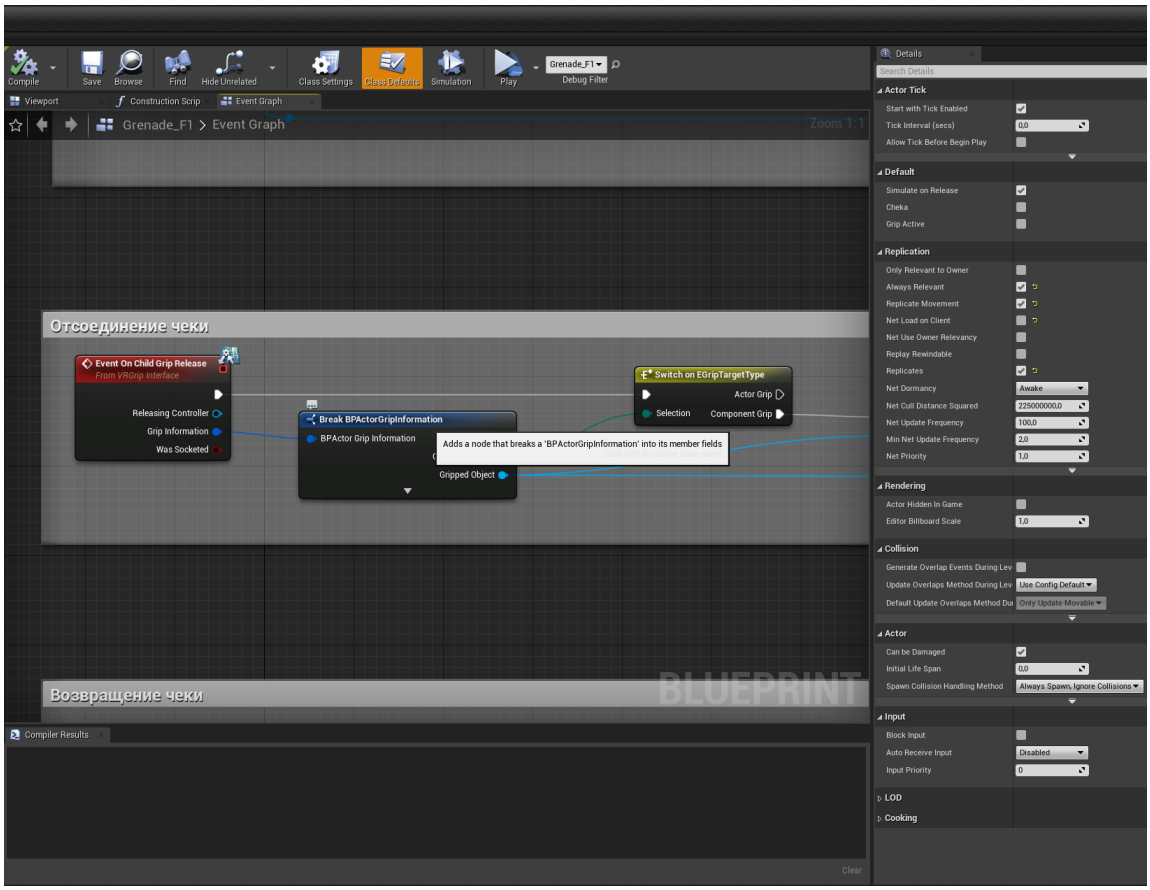


Рисунок 11 – Окно описания нода

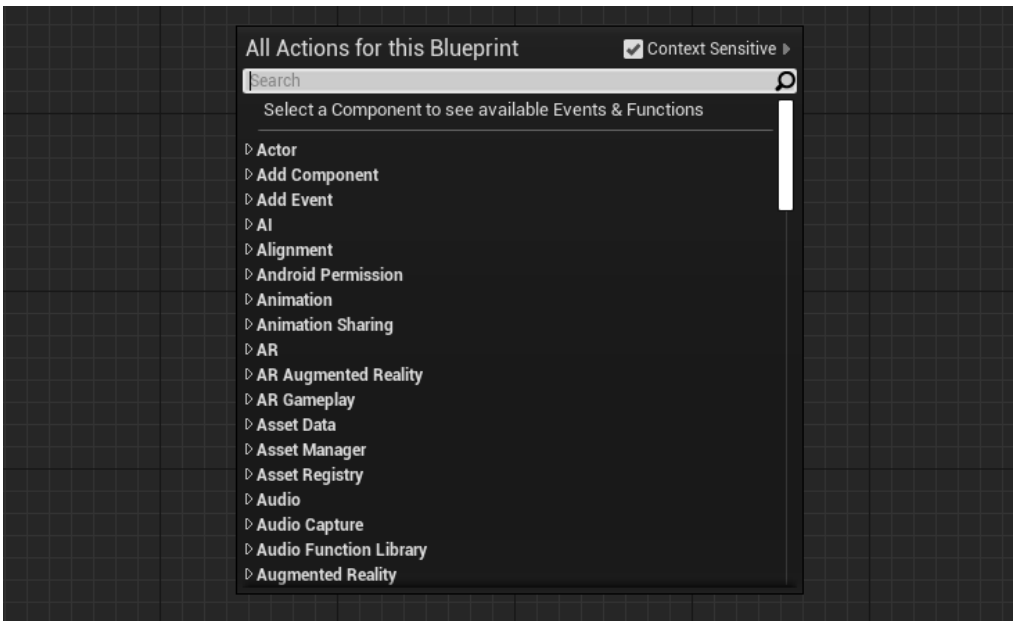


Рисунок 12 – Окно поиска нодов

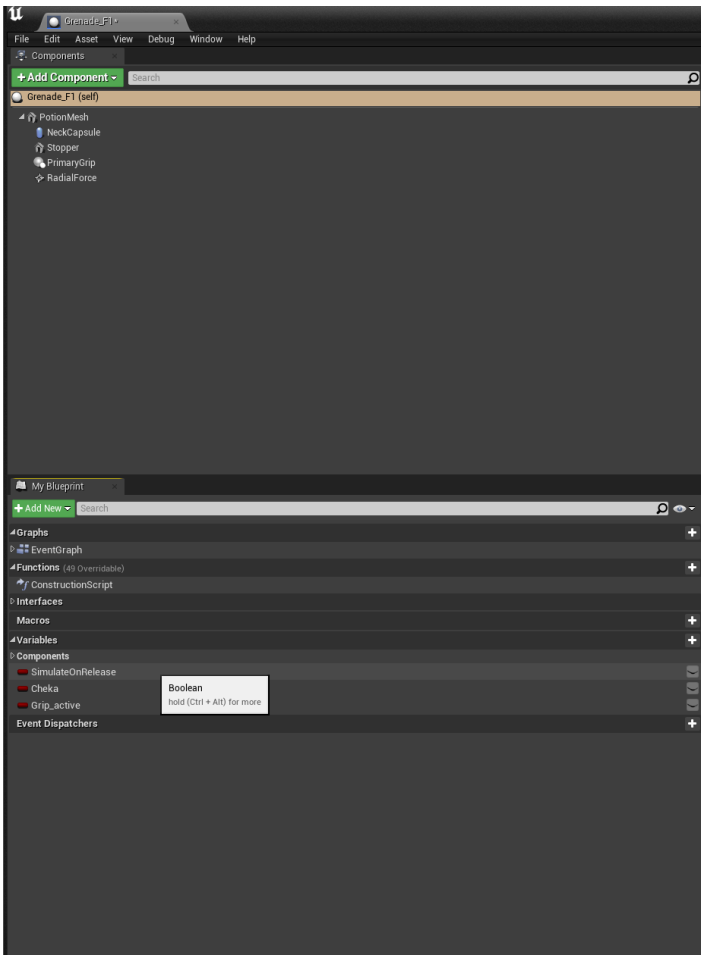


Рисунок 13 – Меню созданных функций и переменных

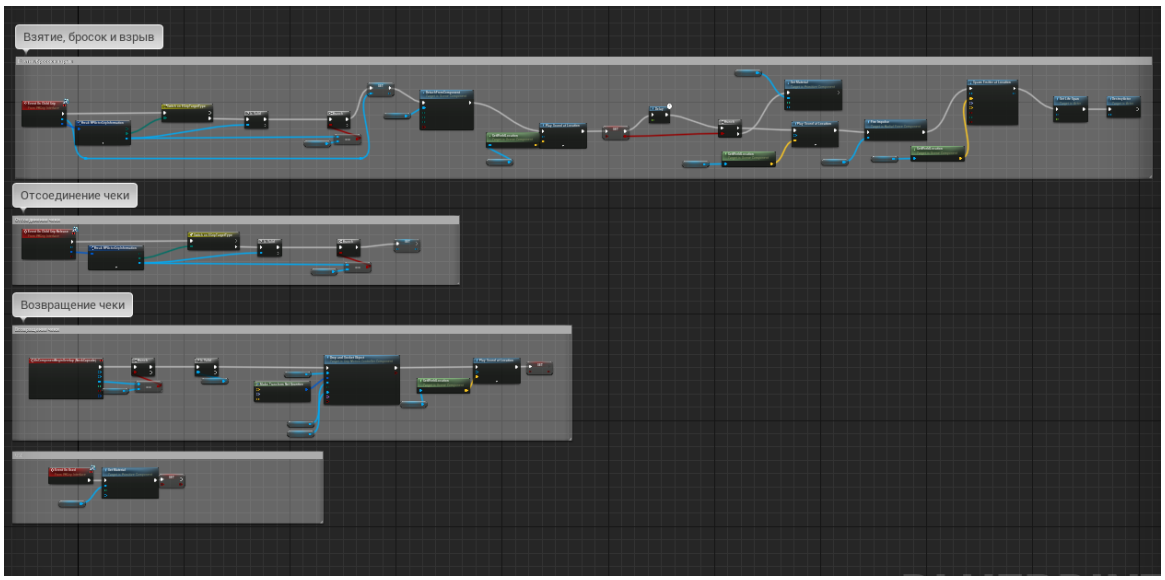


Рисунок 14 – Общий вид созданного скрипта

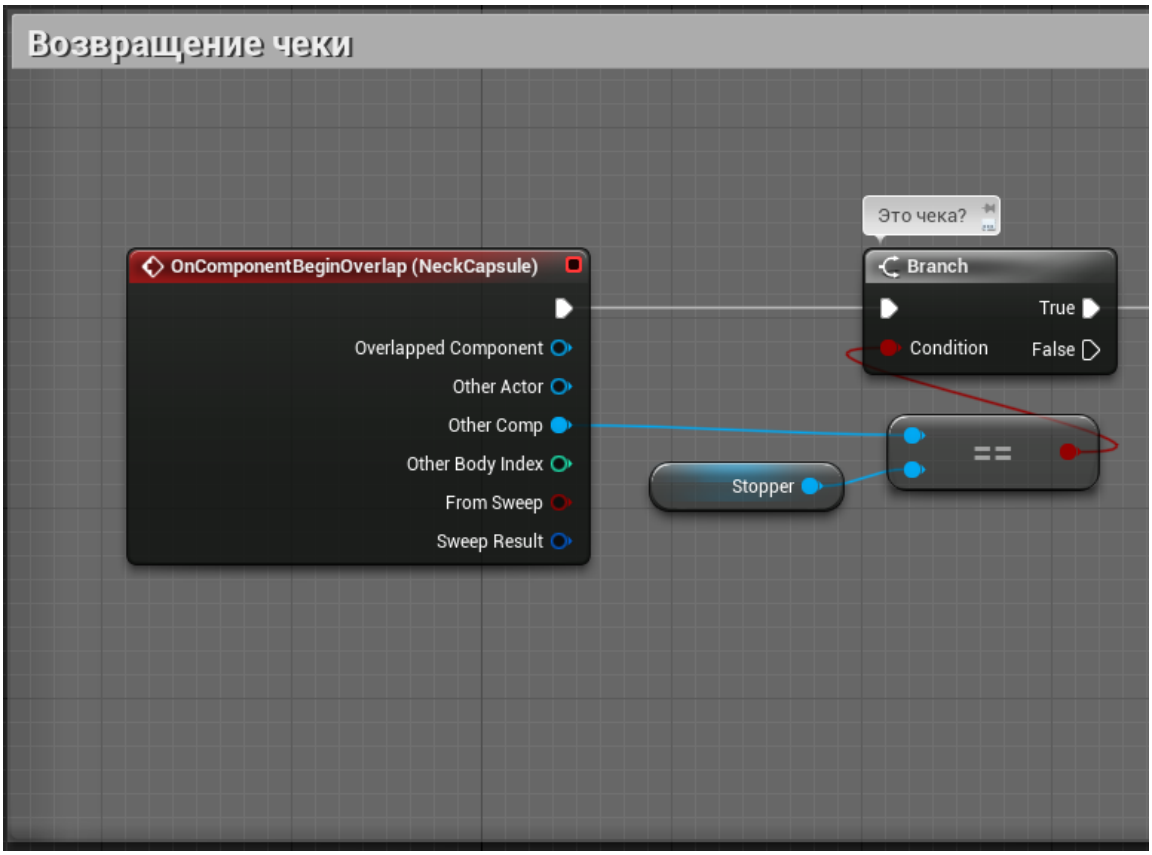


Рисунок 15 – Использование комментариев

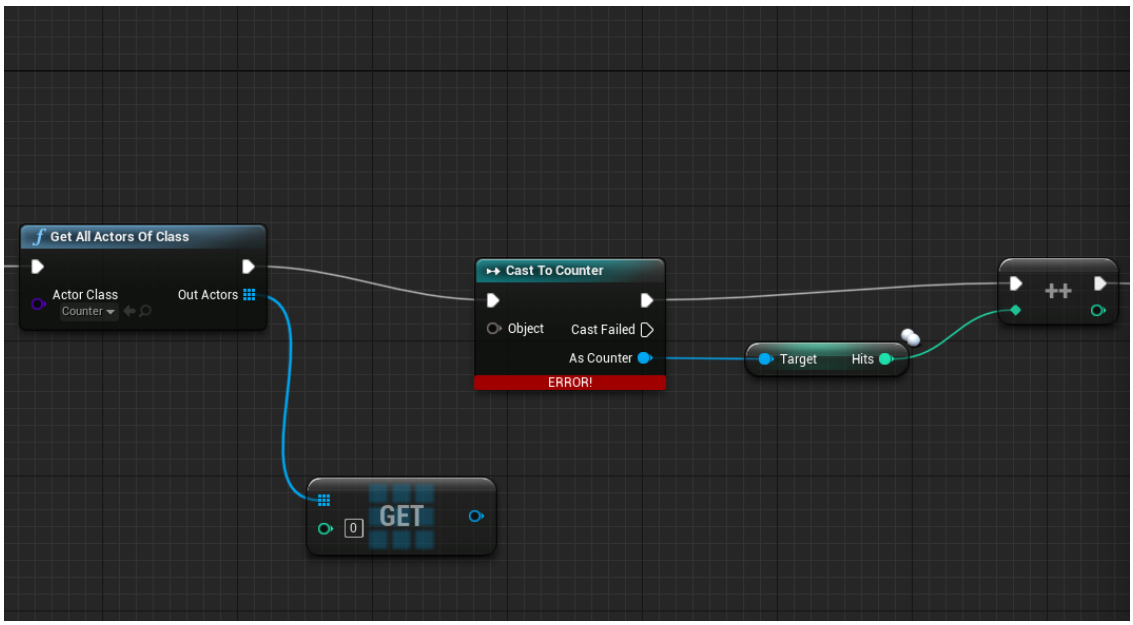


Рисунок 16 – Подсвечивание компилятором ошибок

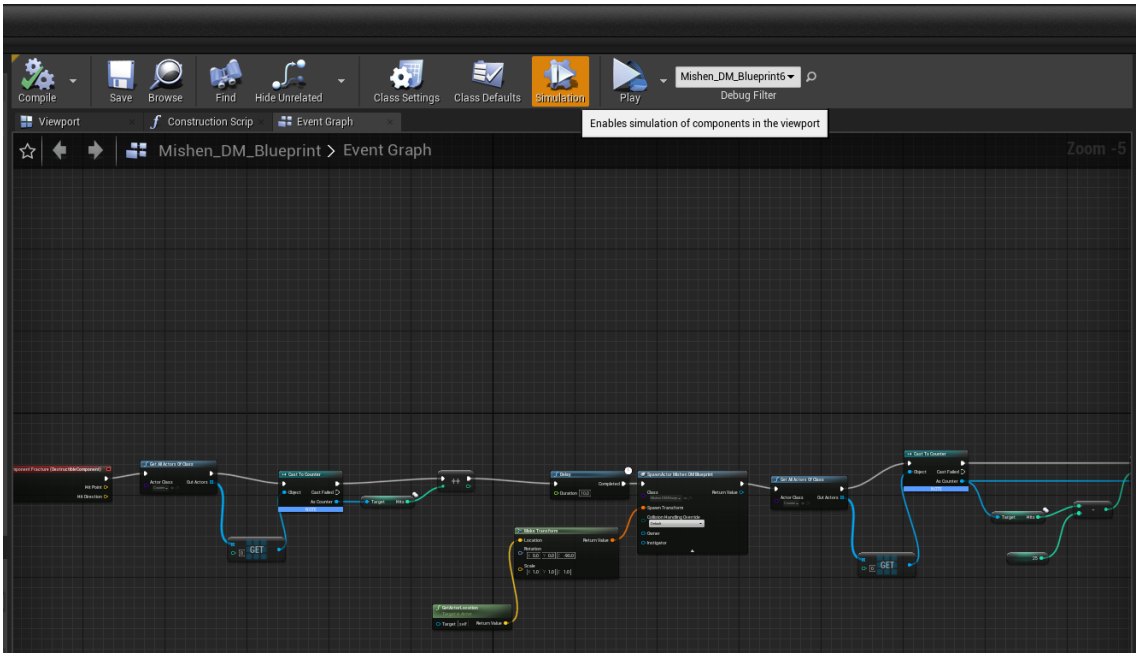


Рисунок 17 – Использование режима симуляции

На рисунке 18 показана диаграмма классов. Она представляет собой упрощенную модель взаимодействия и иерархии элементов системы и их атрибутов.

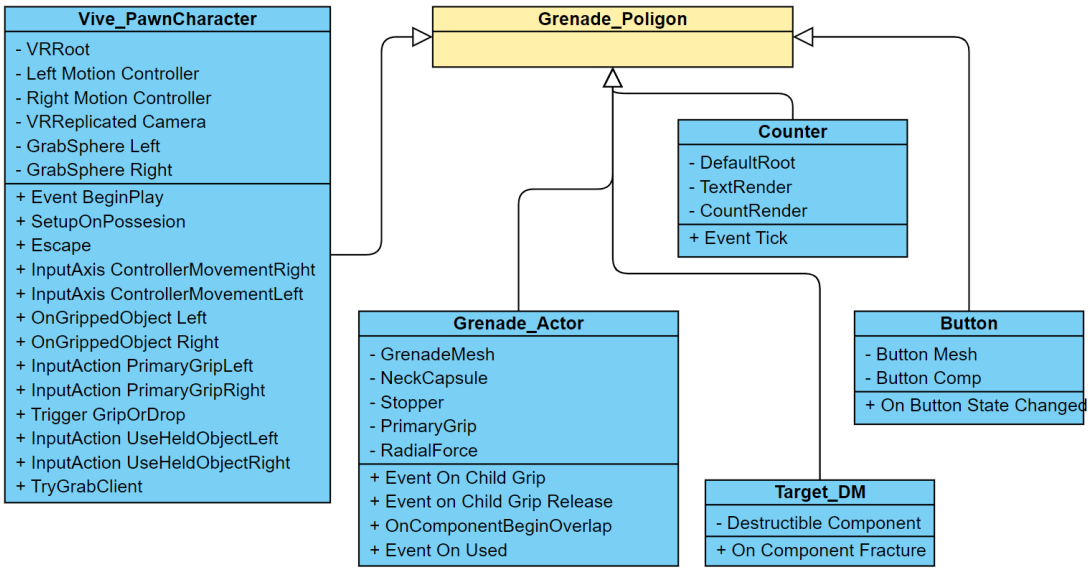


Рисунок 18 – Диаграмма классов

На диаграмме представлены следующие классы:

- *Grenade_Poligon* – основное звено проекта, объединяющий все объекты и блюпринты в единое целое;

- *Vive_PawnCharacter* – класс пользователя (игрока), содержащая функции взаимодействия пользователя и объектов мира:

1) *Event BeginPlay* – функция начала сессии, появления пользователя в мире;

2) *SetupOnPossesion* – функция настройки первоначальных параметров пользователя;

3) *Escape* – функция выхода из приложения;

4) *InputAxis ControllerMovementRight* – функция, отвечающая за передвижение пользователя в пространстве;

5) *OnGrippedObject Left* - функция, отвечающая за “поддержания” состояния поднятых предметов;

6) *InputAction PrimaryGripLeft* – функция, отвечающая за поднятие предметов;

7) *Trigger GripOrDrop* - функция, отвечающая за определение, поднимает или отпускает объект пользователь (так как оба эти

действия зависят от 1 кнопки);

8) *InputAction UseHeldObjectLeft* - функция, отвечающая за использование объекта в руке;

9) *TryGrabClient* - функция, отвечающая за проверку возможности поднятия/отпускания предмета;

- *Grenade_Actor* – класс, определяющий поведение гранат:

1) *Event On Child Grip* - функция, отвечающая за механику взведения гранаты, ее таймера и взрыва;

2*) Event On Child Release* - функция, отвечающая за снятие чеки;

3) *OnComponentBeginOverlap* - функция, отвечающая за возвращение чеки на гранате;

- *Counter* – класс, определяющий поведение счетчика: *Event Tick* - функция, отвечающая за подсчет счетчиком пораженных целей;

- *Button* – класс, определяющий поведение кнопок, ответственных за появление новых гранат: *On Button State Changed* – функция, отвечающая за появление новых гранат;

- *Target DM* – класс, определяющий поведение мишеней: *On Component Fracture* - функция, отвечающая за восстановление уничтоженных мишеней и отправки сигнала на счетчик об очередной пораженной цели.

Далее идет список всех функций с подробным описанием их действия на основе блюпринтов.

Блюпринт *Vive_PawnCharacter* представлен на рисунках 19…25.

Активация блюпринта происходит с помощью функции *Event BeginPlay* (рисунок 19). Данная функция определяет начало сессии. Затем функция *Set Grip Components* связывает наши контроллеры с соответствующими объектами сессии, используя как входные параметры созданные нами объекты “рук” *Grab Sphere Left* и *Grab Sphere Right*. Затем происходит инициализация системы трекинга наших движений с помощью функции *Set Tracking Origin*. Так как современные шлемы имеют встроенную

систему трекинга, используем в качестве параметра *Stage*, то есть пространство вокруг пользователя. Далее для процесса отладки вызываем функцию *Set Spectator Screen Mode*. Она позволяет вывести на экран монитора картинку, видимую пользователем в шлеме. Далее с помощью функции *Set Movement Hands* происходит настройка ответственности контроллеров за перемещение и повороты пользователя в пространстве (какой контроллер ответственен за повороты, а какой за движение). Далее происходит поочередный спавн контроллеров. Сначала происходит спавн самого контроллера в мире с помощью функции *SpawnActor BP Teleport Controller*, а далее происходит его привязка к телу пользователя с помощью функции *AttachActorToComponent*.

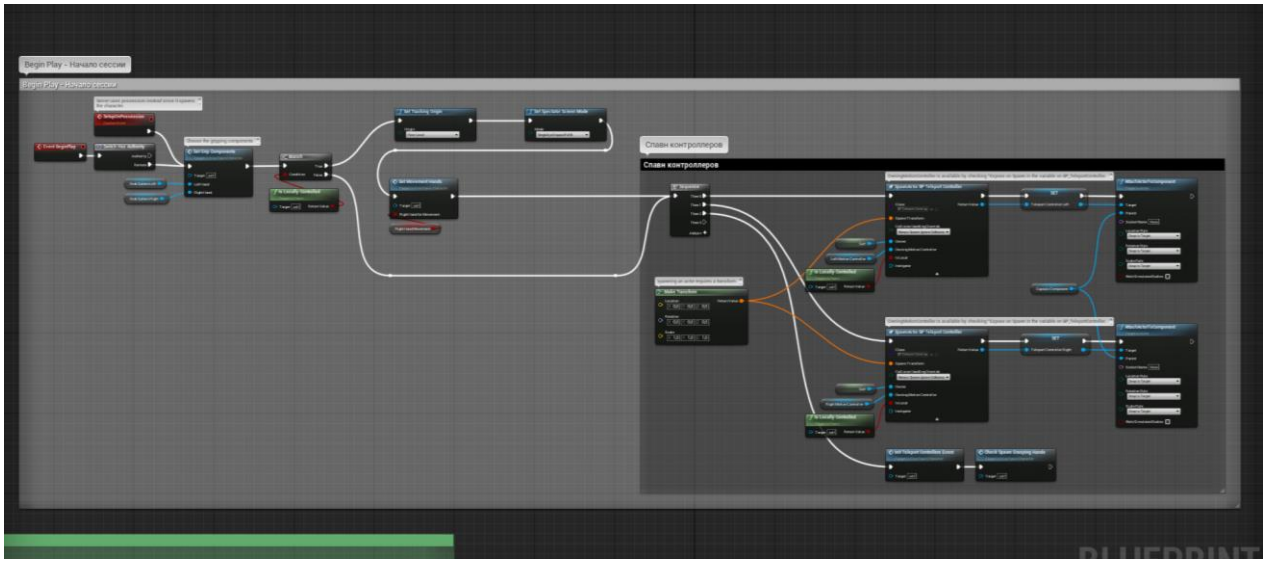


Рисунок 19 – Часть блюпринта *Vive_PawnCharacter*: функция начала сессии

Движение пользователя вызывается функцией *InputAxis ControllerMovementRight* (рисунок 20). Сначала с помощью условия *Branch* проверяется привязка контроллеров к видам перемещения (какой контроллер ответственен за повороты, а какой за движение). Далее вызываются две функции – *Handle Current Movement* и *Input Handle Turn Input*. Первая функция отвечает за передвижение пользователя в плоскости, а вторая

отвечает за его повороты. В качестве входных параметров они использую ранее инициализированные объекты контроллеров.

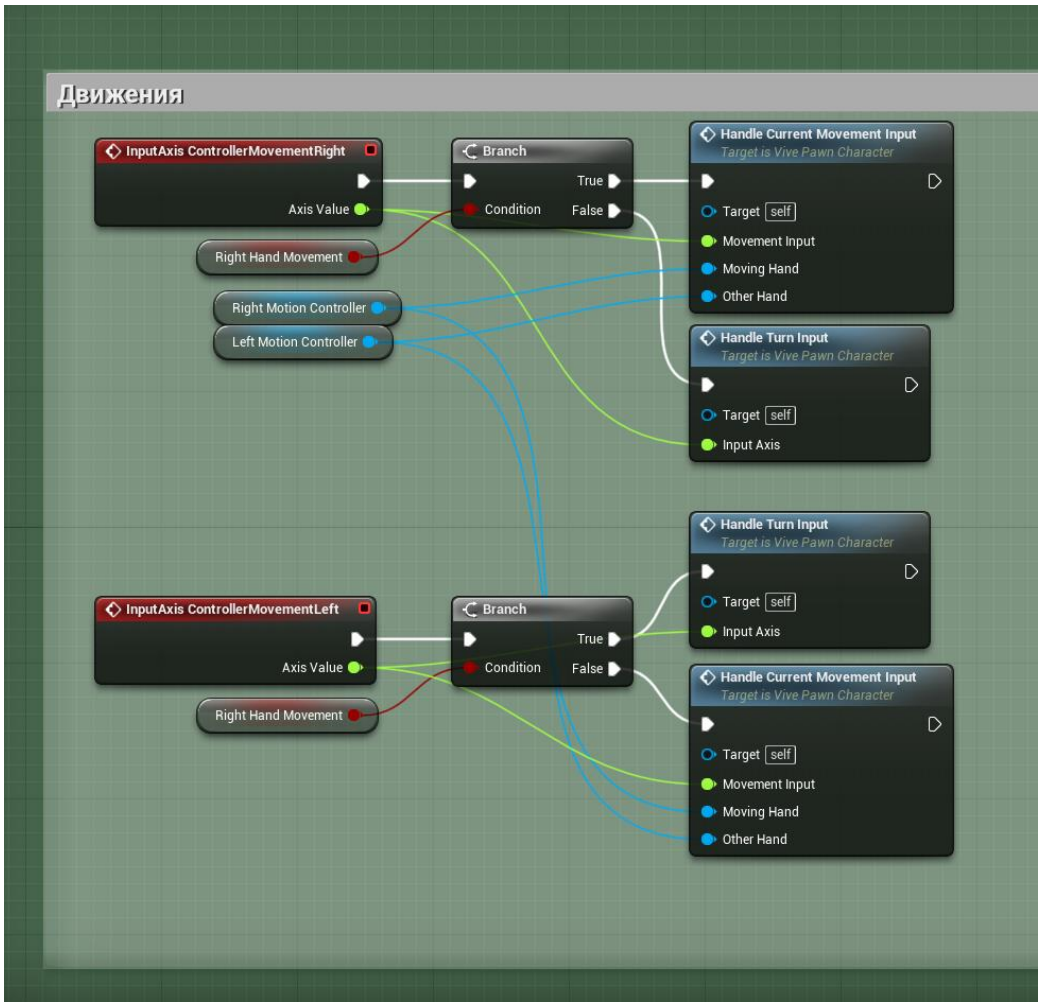


Рисунок 20 – Часть блюпринта *Vive_PawnCharacter*: функция привязки управления

Завершение сессии вызывается функцией *Escape* (рисунок 21). Сначала управление контроллерами передается от приложения к ОС шлема с помощью функции *Cast To PlayerController*, используя в качестве входных параметров объекты контроллеров, созданных нами ранее. Далее с помощью функции *Destroy Session* завершается текущая сессия, из памяти высвобождается используемая приложением память и ресурсы компьютера. В конце используется функция *Quit Game* для окончательного закрытия приложения.

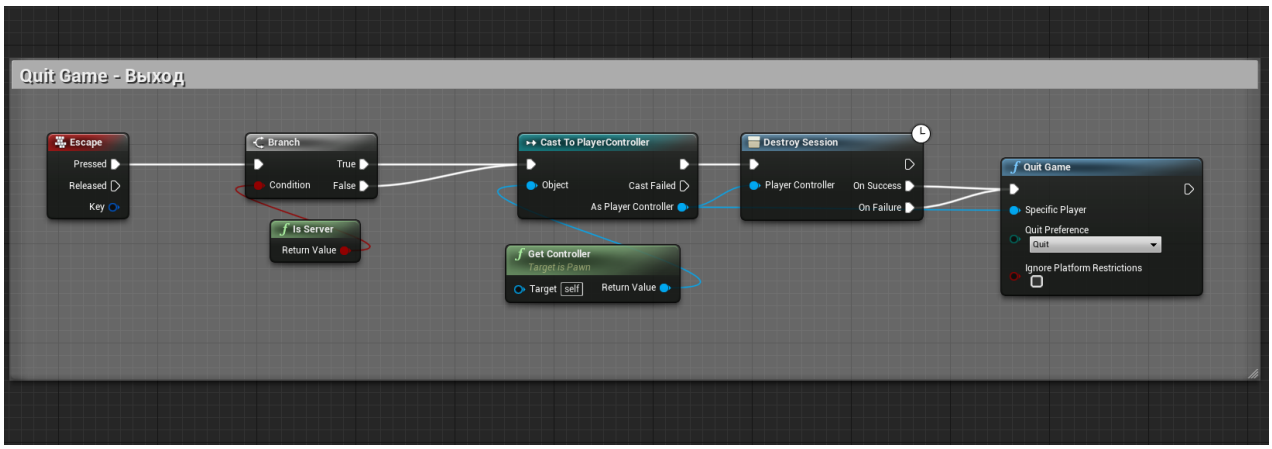


Рисунок 21 - Часть блюпринта *Vive_PawnCharacter*: функция закрытия приложения

Поднятие и отпускание предметов в приложении реализовано с помощью функций *InputAction PrimaryGripLeft* и *InputAction PrimaryGripRight* для левого и правого контроллера соответственно (рисунок 22). Данная функция имеет два состояния – *Pressed* и *Released*, ответственные за нажатие и отпускание соответственно. Каждое состояние вызывает функцию *Trigger Grip or Drop*. Она и воспроизводит действие в зависимости от подаваемого состояния. Также входными параметрами для нее являются используемый контроллер (левый или правый) и компонент области действия каждого контроллера *Hand Grip Component*.

Некоторые объекты сцены можно “использовать”, в нашем случае – гранату. Функции, определяющие данное использование – *InputAction UseHeldObjectLeft* и *InputAction UseHeldObjectRight* для левого и правого контроллера соответственно (рисунок 23). Эта функция имеет два состояния – нажатие и отжатие. Первым делом вызывается функция *If Over Widget Use* – она определяет наличие в руке какого-либо объекта. Далее вызывается функция *Check Use Held Items Check*, определяющая, имеет ли сам предмет какое-либо связанное действие. Далее с помощью функции *Use Secondary Attachment* определяется, имеют ли какие-либо вторичные приспособления объекта привязанные к ним действия.

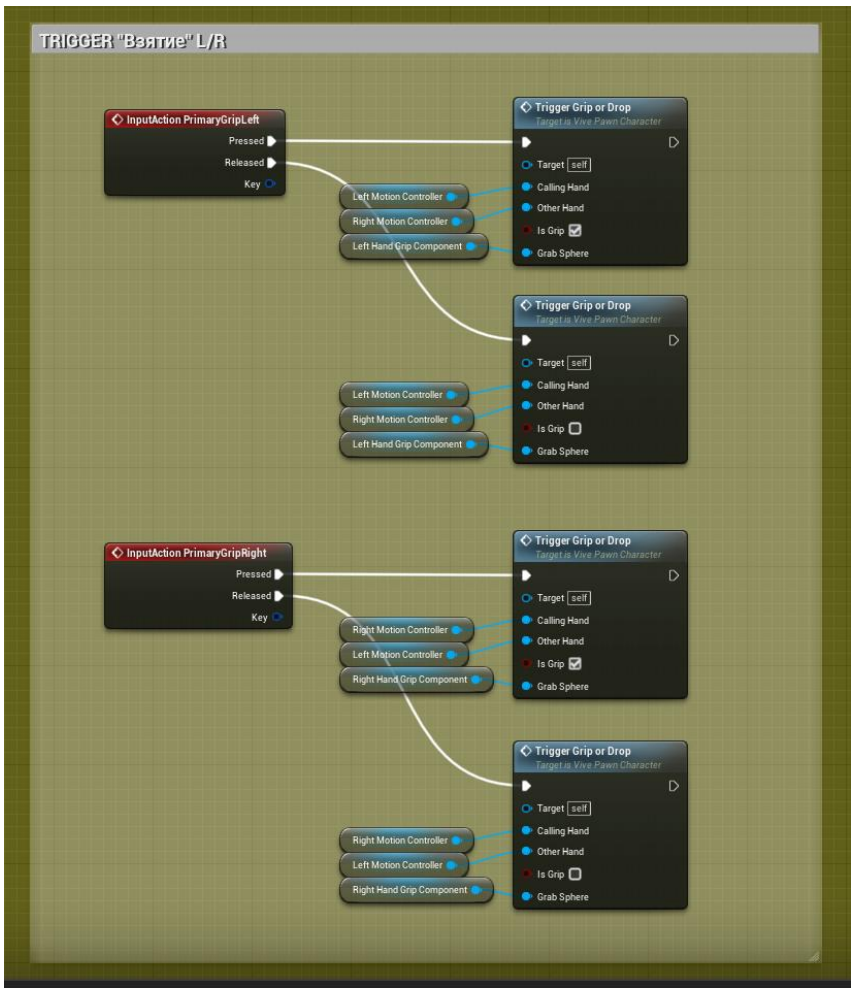


Рисунок 22 - Часть блюпринта *Vive_PawnCharacter*: функция взятия и отпускания объектов

При взятии/отпускании предмета важно, чтобы это действие было возможным и не прерывало какие-либо важные проходимые функции. Для этого создана функция *Trigger Grip or Drop* (рисунок 24). Сначала вызывается условие *Branch*, которое проверяет происходит поднятие или отпускание предмета. Если происходит взятие, то происходит вызов функции *Grip or Drop Object Clean*, которая проверяет, не мешает что-либо поднятию объекта. Если происходит выбрасывание объекта, то помимо функции *Grip or Drop Object Clean* вызывается функция *Clear Climbing*, которая возвращает контроллеру возможность поднятия нового объекта.

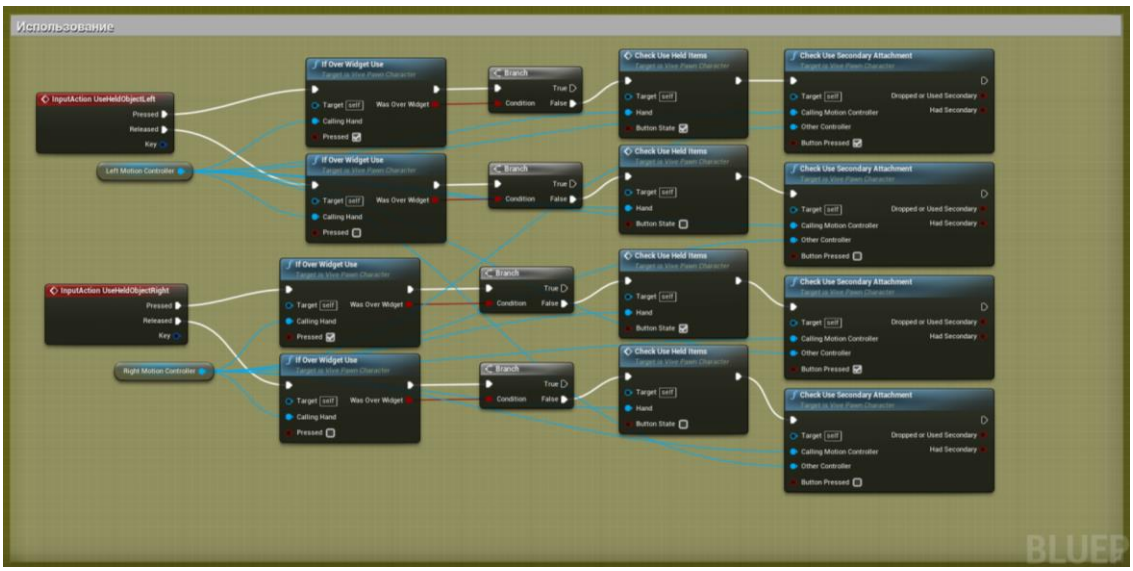


Рисунок 23 - Часть блюпринта *Vive_PawnCharacter*: функция использования объектов

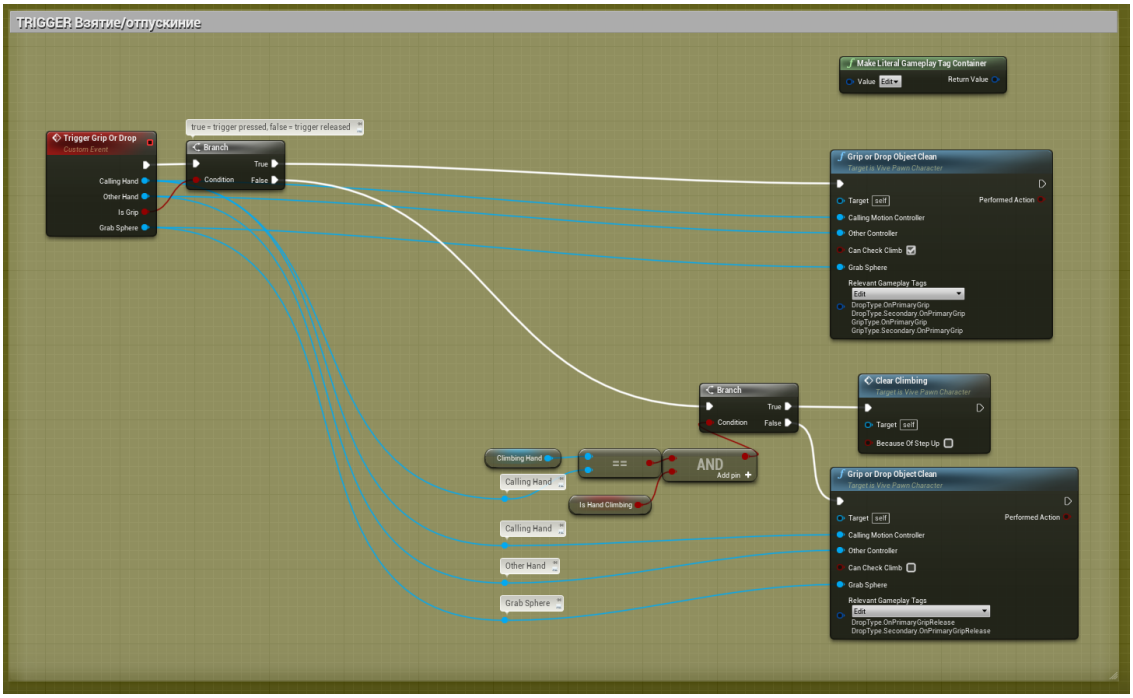


Рисунок 24 - Часть блюпринта *Vive_PawnCharacter*: функция возможности взятия/отпускания

Если требуется поменять объект между двумя руками, то для этого вызывается функция *TryGrabClient* (рисунок 25). Сначала вызывается функция *Switch on EControllerHand*, которая определяет текущую

контрольную руку. Далее вызывается функция *Try to Grab Object*, которая и вызывает смену контрольной руки. Входными параметрами являются текущая контрольная рука, объект, который будет перекидываться, ссылки на оба контроллера, кость привязки к контроллеру и переменная проверки доступности второй руки.

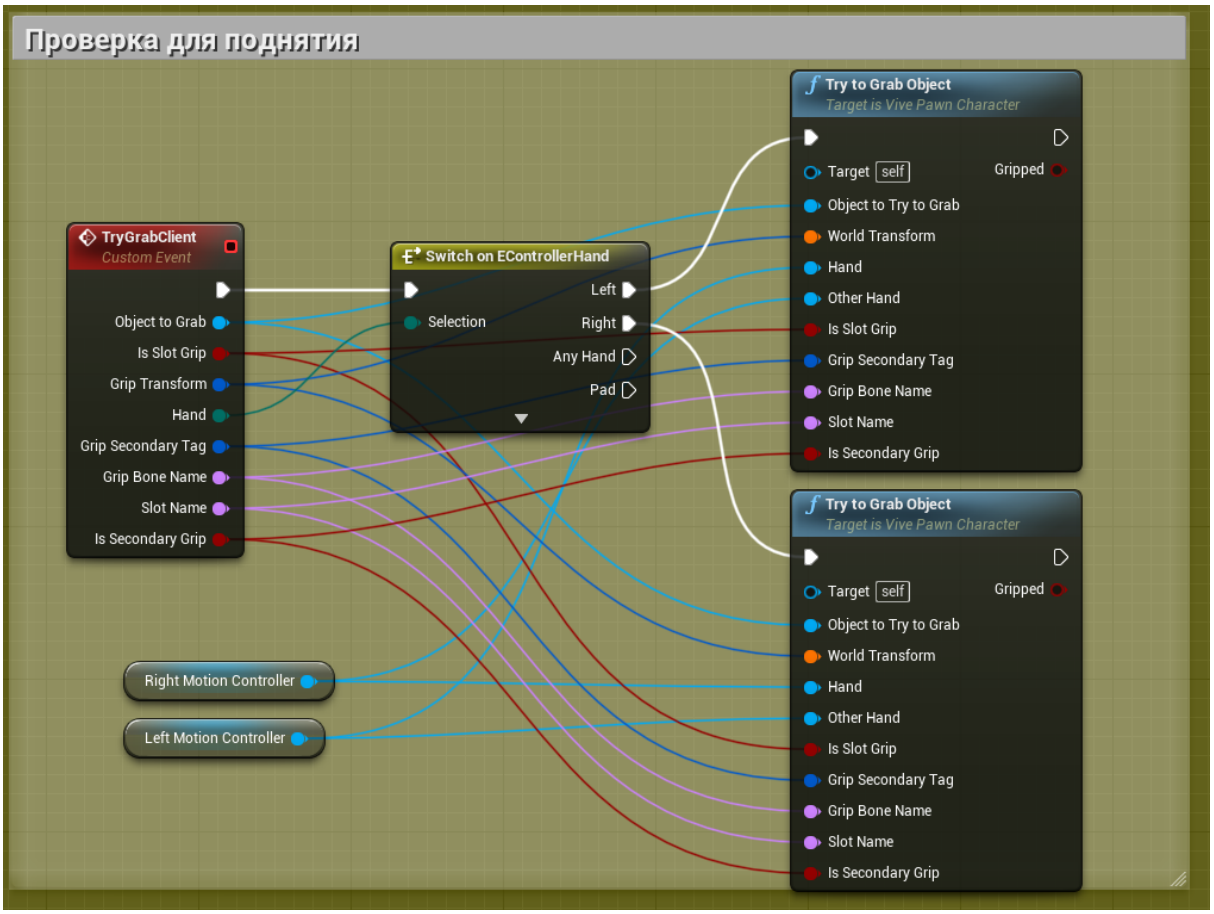


Рисунок 25 - Часть блюпринта *Vive_PawnCharacter*: функция смены рук

Каждый тик того времени, когда в руке находится какой-либо объект, этот объект должен оставаться в привязке к контроллеру. Данное действие осуществляют функции *On Gripped Object* (*LeftMotionController*) и *On Gripped Object* (*RightMotionController*) для левого и правого контроллеров соответственно (рисунок 26). Вначале вызывается условие *Branch*, проверяющее, держит ли все еще объект пользователь. Если условие верно, то вызывается функция *Reset Peak Low Pass Filter*. Она придает движение

поддерживаемому объекту согласно перемещению контроллера, подаваемого как входной параметр.

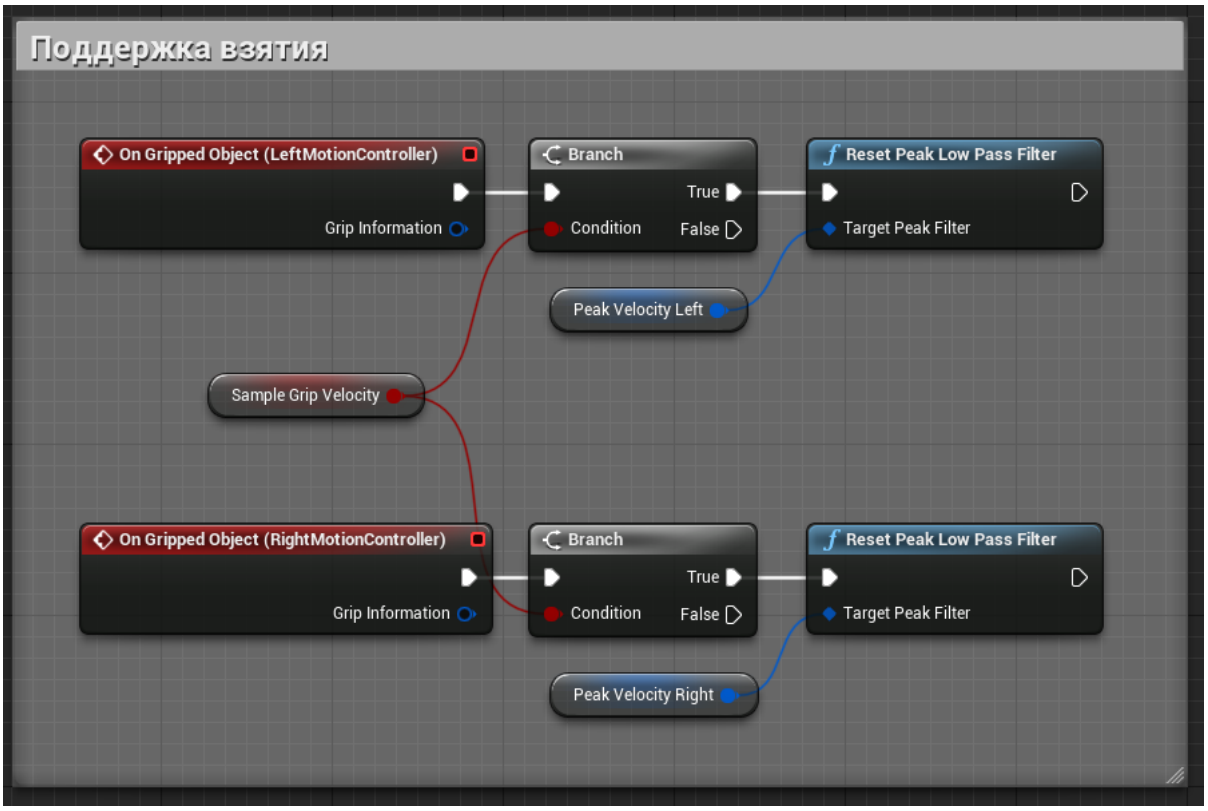


Рисунок 26 - Часть блюпринта *Vive_PawnCharacter*: функция поддержки держания

Блюпринт *Grenade_Actor* представлен на рисунках 27…29.

Функция *Event On Chid Grip* отвечает за взведение гранаты и ее подрыв (рисунок 27). Сначала вызывается функция *Break BPActorGripInformation*, которая дает информацию о текущем объекте, находящимся в руке. Далее с помощью функции *Switch on EGripTargetType* позволяет схватиться второй рукой за элемент чеки. Следующая функция *Is Valid* позволяет определить, действительно ли второй рукой схвачена чека. Если условие верно, то вызывается функция *DetachFromComponent*, которая и отсоединяет элемент чеки от гранаты. Далее воспроизводится звуковая функция *Play Sound at Location*, которая проигрывает звук снятия чеки. После снятия чеки функция *Set* устанавливает значение логической переменной *Cheka* в значение *True*.

Далее активируется функция-таймер *Delay* с входным значением, соответствующем ТТХ гранаты. Затем снова воспроизводится звуковая функция *Play Sound at Location*, которая проигрывает звук подрыва. Далее вызывается функция *Fire Impulse*, которая активирует параметр физического воздействия гранаты на окружающие мишени. Затем вызывается функция *Spawn Emitter at Location*, которая вызывает срабатывание визуального эффекта подрыва гранаты. Следующей вызывается функция *Set Life Span*, которая указывает время до уничтожения объекта (гранаты) и очищения памяти.

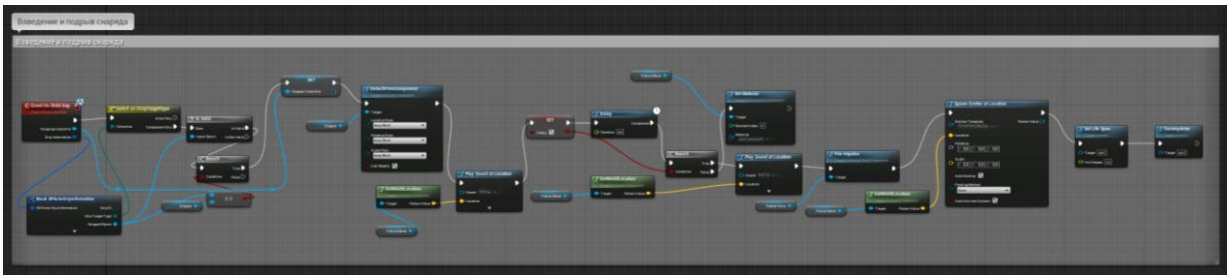


Рисунок 27 - Часть блюпринта *Grenade_Actor*: взведение и подрыв гранаты

Так как снятая чека уже представляет собой новый объект на сцене, то ей нужно дать параметры, которые бы определяли ее поведение как отдельного объекта. Это выполняет функция *Event on Child Grip Release* (рисунок 28). Сначала вызывается функция *Break BPActorGripInformation*, которая дает информацию о текущем объекте, находящимся в руке. Следующая функция *Is Valid* позволяет определить, действительно ли второй рукой схвачена чека. Если условие верно, то вызывается функция *Set*, которая устанавливает права чеки как самостоятельного объекта.

Возможность возвращения чеки на место определяет функция *On Begin Overlap* (*Neck Capsule*) (рисунок 29). *Neck Capsule* – объект, содержащий невидимый меш, при соприкосновении которого и будет происходить возвращение чеки.

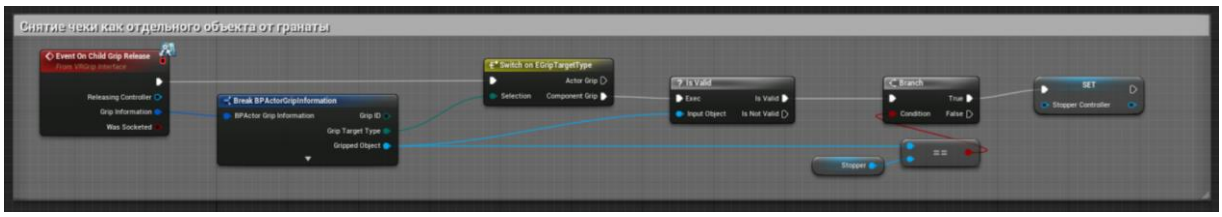


Рисунок 28 - Часть блюпринта *Grenade_Actor*: определение поведения чеки

Первым делом вызывается функция-условие *Branch*, определяющая, чекой ли является присоединяемый объект. Если ответ верный, то вызывается функция *Drop and Socket Object*, которая и присоединяет чеку к гранате. Далее вызывается звуковая функция *Play Sound at Location*, для воспроизведения звука возвращения чеки.

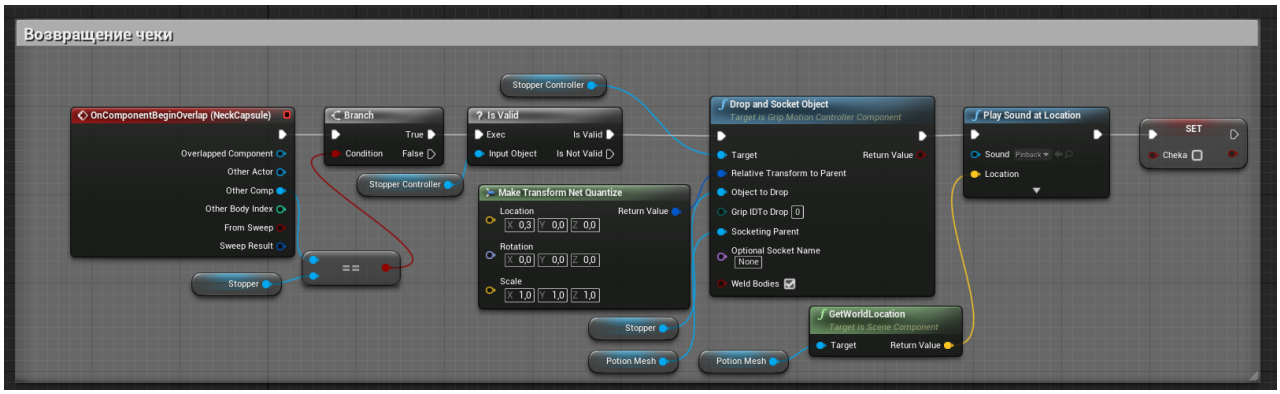


Рисунок 29 - Часть блюпринта *Grenade_Actor*: возвращение чеки

Блюпринт *Counter* представлен на рисунке 30.

Функция *Event Tick* каждый тик игрового времени обновляет в прямом времени счетчик показа числа пораженных целей, вызывая функцию *SetText* с параметром-переменной *Hits*, отвечающей за количество пораженных целей (рисунок 30).

Блюпринт *Target_DM* представлен на рисунке 31.

Функция *On Component Fracture (DestructibleComponent)* вызывается при каждом поражении одной из целей и отвечает за замену уже пораженных целей и посылку сигнала счетчику (рисунок 31).

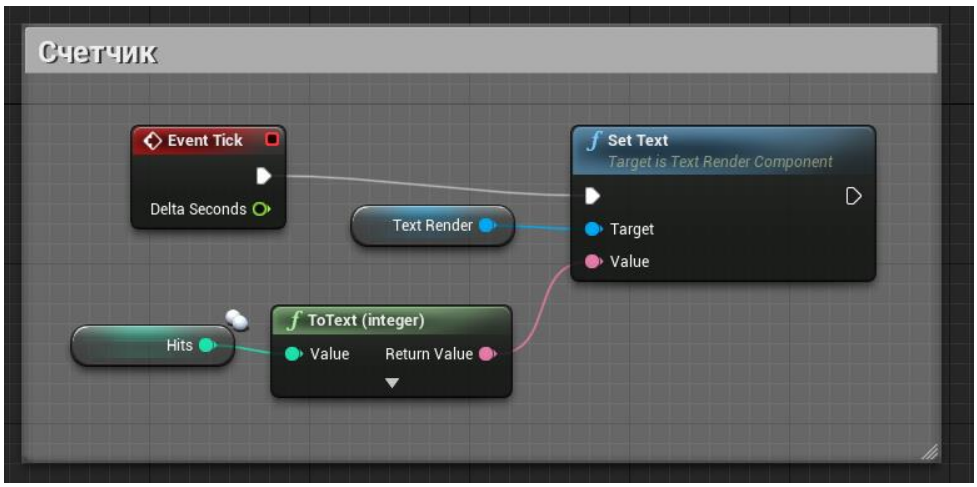


Рисунок 30 - Часть блюпринта *Counter*: обновление счетчика

Сначала вызывается функция *Get All Actors Of Class*, позволяющая любому объекту мишени отсылать сигнал на счетчик. Далее вызывается функция *Cast To Counter*, позволяющая из этого блюпринта обращаться к переменным, находящимся в блюпринте счетчика. Затем переменную счетчика *Hits* увеличиваем на единицу. Затем вызывается функция-таймер *Delay* с временем 10 секунд. Затем вызывается функция *SpawnActor Mishen DM Blueprint* восстанавливает уничтоженную мишень.

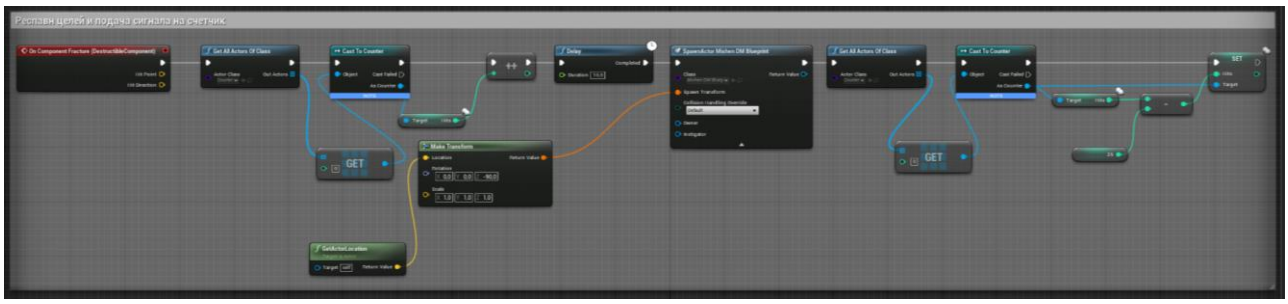


Рисунок 31 - Часть блюпринта *Target_DM*: замена мишеней

Блюпринт *Button* представлен на рисунке 32.

Функция *On Button State Changed (ButtonComp)* отвечает за спавн новых гранат (рисунок 32). Она вызывает функцию *SpawnActor* с параметрами места спавна и типа появляемой гранаты.

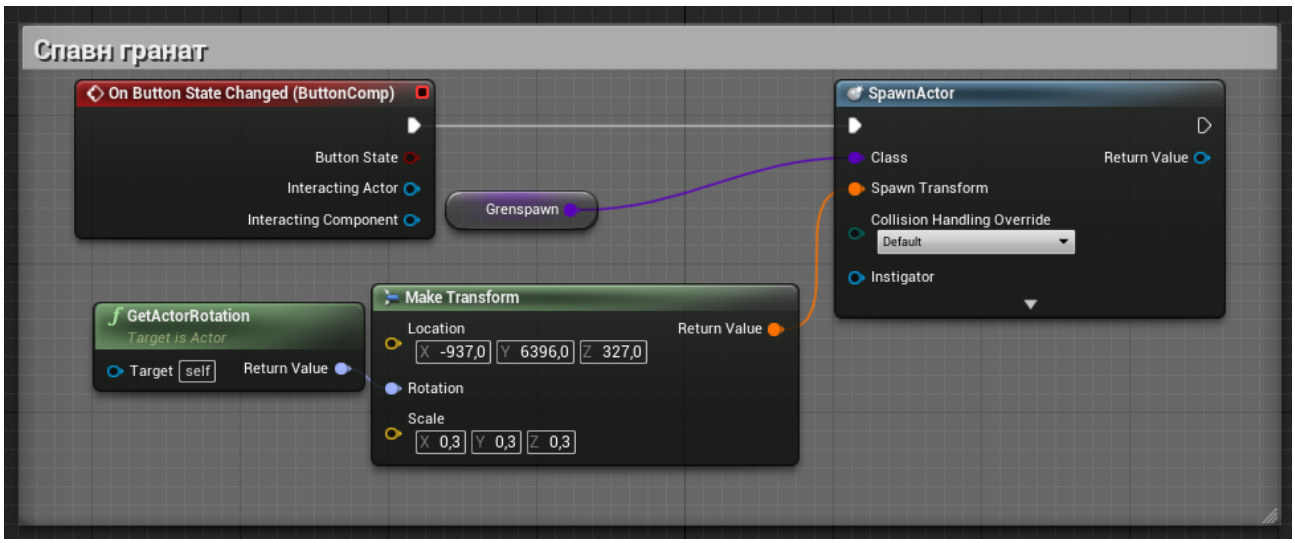


Рисунок 31 - Часть блюпринта *Button*: появление гранат

1.6 Тестирование и отладка

Тестирование и отладка проводились с использованием шлема виртуальной реальности *Oculus Rifs S* последнего поколения, имеющий следующие характеристики:

- общее разрешение 2560x1440;
- разрешение для каждого глаза 1280x1440;
- технология экрана IPS;
- частота обновления 80 Гц;
- угол обзора 110 градусов;
- регулировка межзрачкового расстояния - да;
- вес – 2000 г.;
- контроллер движений в комплекте.

Данный шлем является одним из лучших на данный момент. Имея высокое разрешение и частоту обновления картинки, он дает лучшее изображение.

Для тестирования программы был использован ПК со следующими характеристиками:

- видеокарта: *NVIDIA GeForce RTX 2060*;

- процессор: *AMD Ryzen 7 2700 3.20 GHz*;

- ОЗУ: 16 *GB*;

- ОС: *Windows 10 X64*;

- память: не менее 10 *GB*.

Локация, где изначально появляется пользователь – полигон, где и будет происходить его дальнейшее обучение. Первое что увидит обучаемый – плакат, объясняющий основы передвижения и взаимодействия с предметами в виртуальном мире (рисунок 32).

Рисунок 32 – Плакат с азами управления

Так как приложение создавалось с использованием *OpenVR* в составе *SteamVR*, при использовании других шлемов, управление будет оставаться идентичным для всех видов контроллеров.

Освоившись с управлением, обучаемый следует на основной полигон, представляющий главное место обучения бойца. Полигон имеет следующие места: стенд обучения пользования снарядами, стенды с полезной информацией и характеристиками гранат, ящики с неограниченным запасом гранат, полигон с мишенями (рисунок 33)

Рисунок 33 – Основной полигон

Слева от пользователя находится стенд обучения пользования гранатами. На нем детально описаны способы метания гранат, снаряжение гранаты, их хранение, методы броска гранаты лежа, сидя, стоя, из окопа, в движении, механика действия гранат (рисунок 34)

Справа от пользователя находятся ящики с гранатами, стенды с полезной информацией о них и счетчик пораженных целей (рисунок 35).

Стенды содержат визуальное представление и техническую характеристику различных типов гранат как отечественного, так и зарубежного производства (рисунок 36).

Каждый вид гранат имеет различные характеристики, выражающиеся во времени взведения гранаты, радиуса эффективного поражения, количестве осколков, весе и формы (рисунки 37…40).

Напротив стендов находятся ящики с расположенными на них кнопками. При нажатии данных кнопок будут появляться новые гранаты того типа, который изображен на стенде рядом с ящиком (рисунок 41)

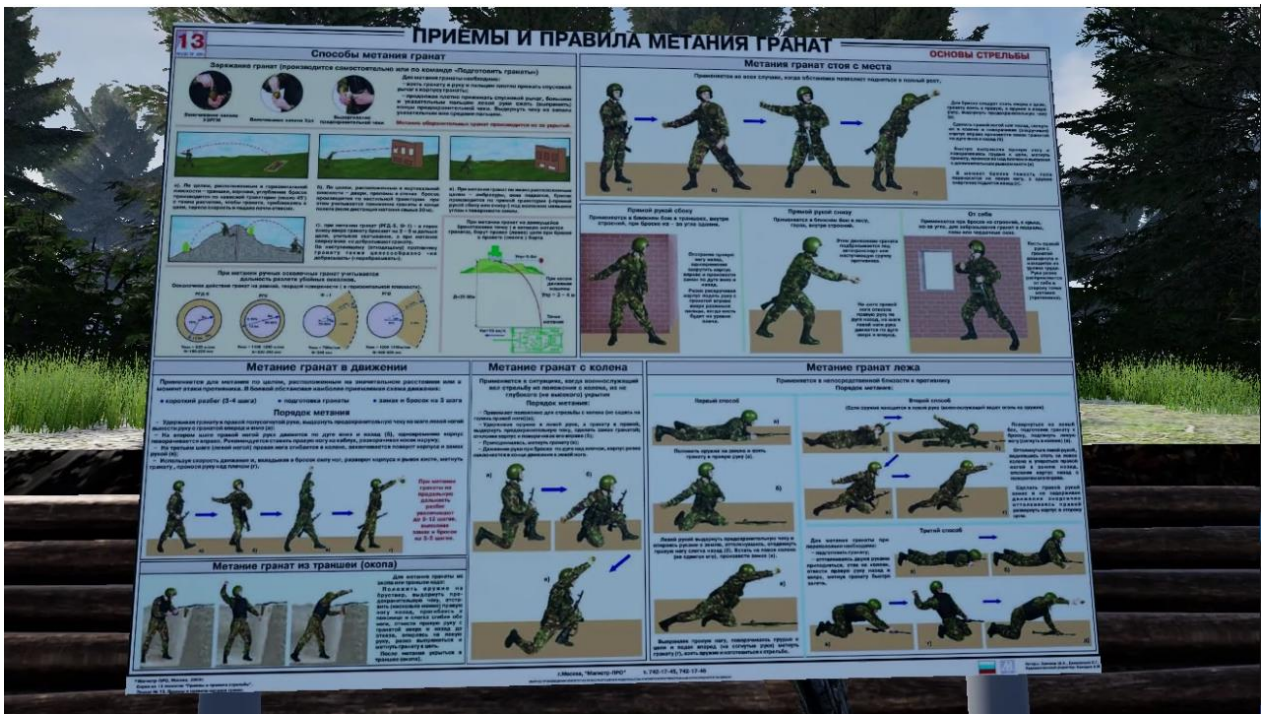


Рисунок 34 – Стенд обучения пользования гранатами

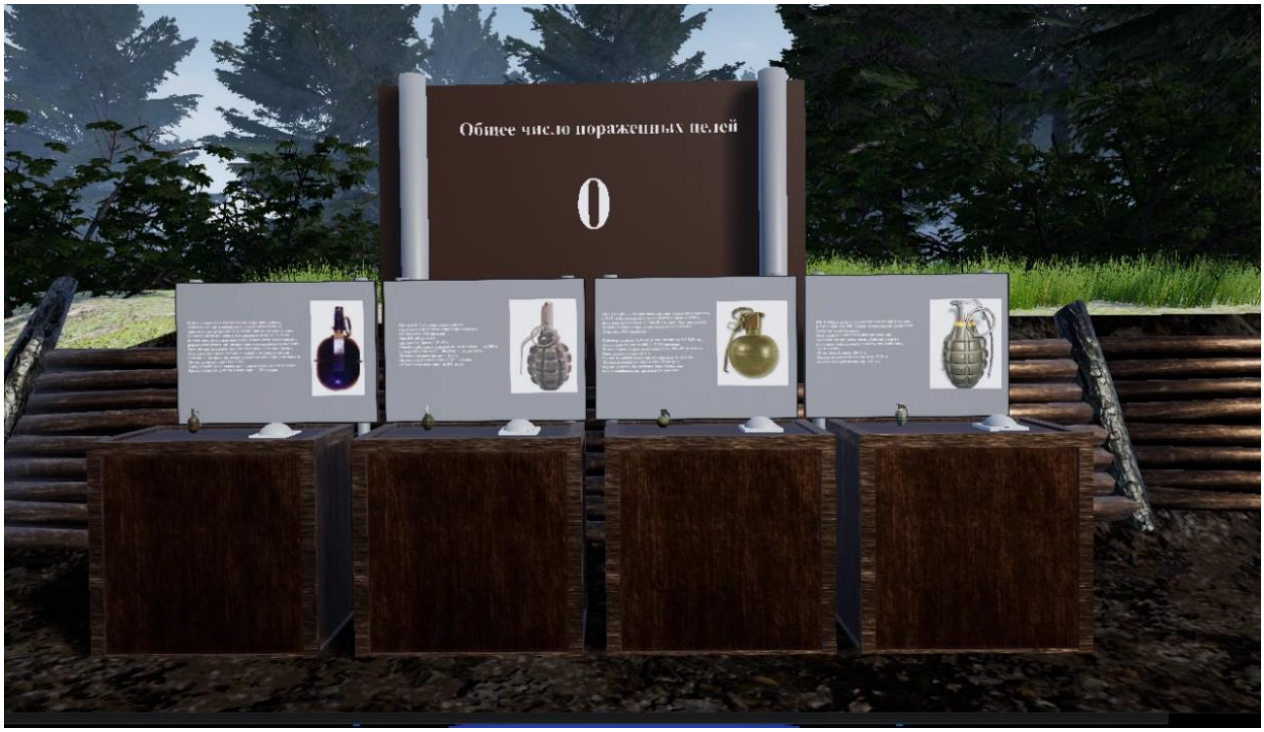


Рисунок 35 – Правая сторона полигона

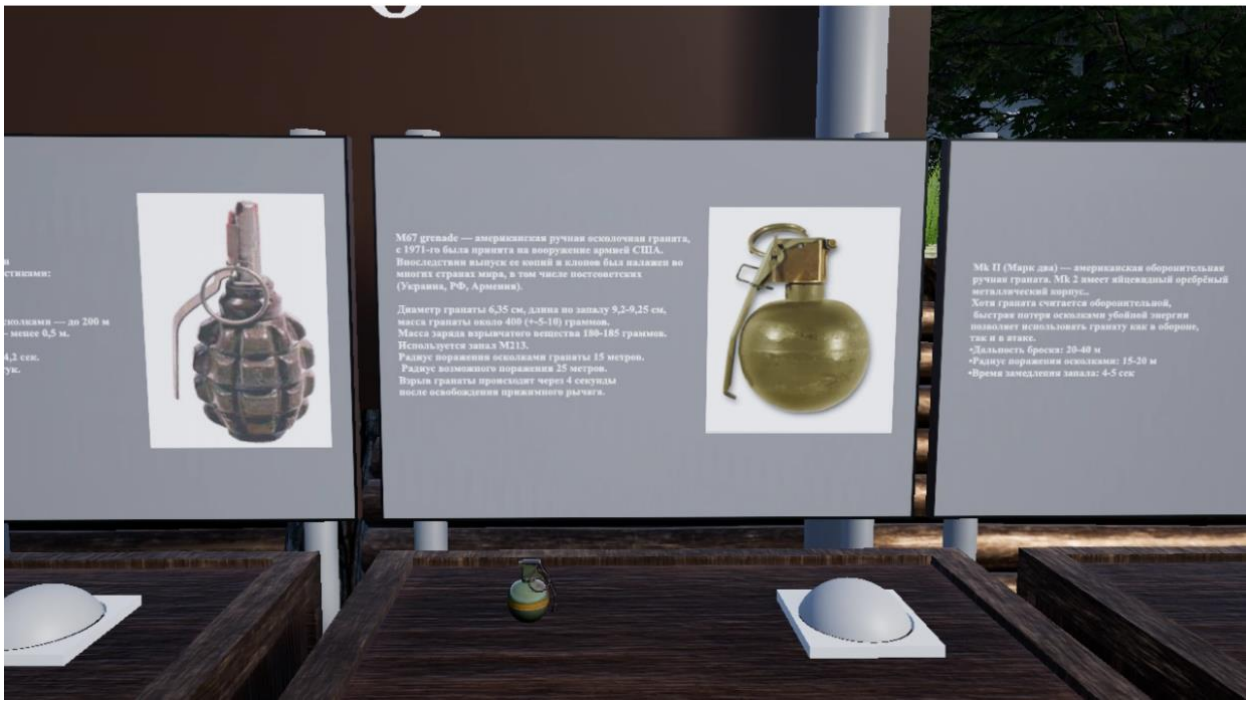


Рисунок 36 – Один из стендов с характеристиками гранат

Рисунок 37 – Граната РГД-5

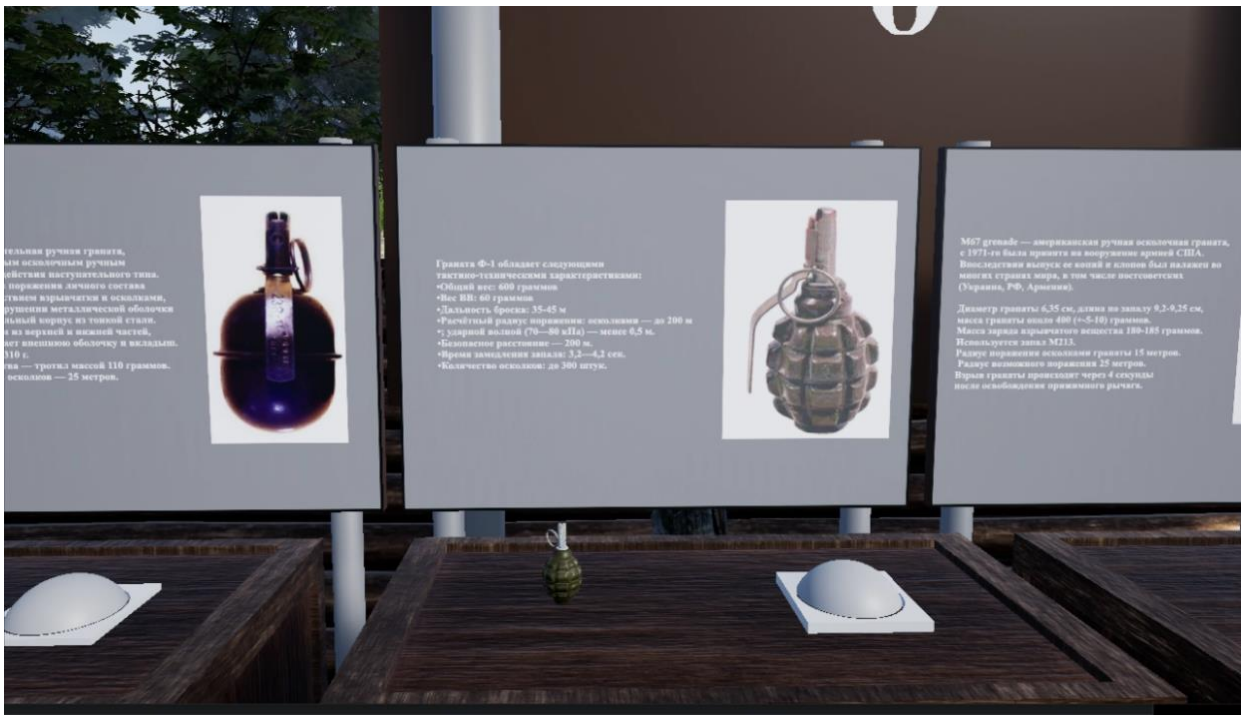


Рисунок 38 – Граната Ф-1

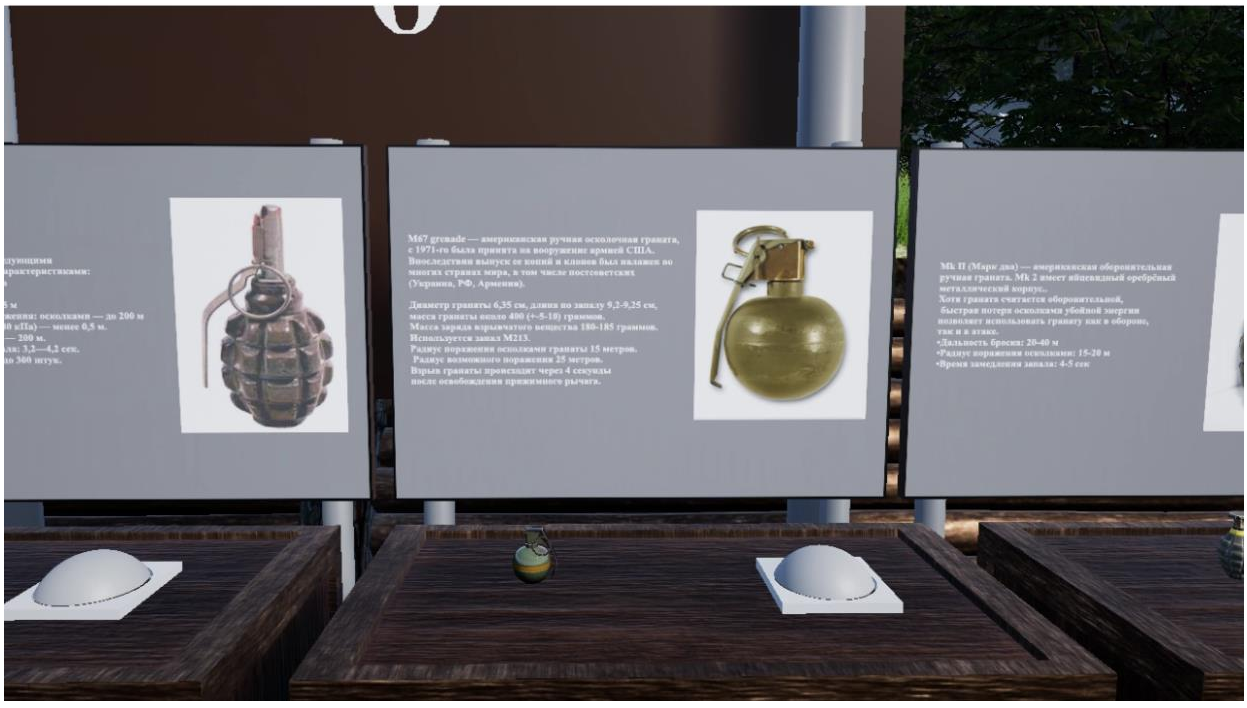


Рисунок 39 – Граната *M67*

За стендами с информацией находится табло, показывающее текущее количество пораженных на полигоне мишеней (рисунок 42).

Рисунок 40 – Граната *MK2*

Рисунок 41 – Выдача новых гранат при нажатии кнопки

Спереди от пользователя находится полигон с мишенями. Некоторые мишени находятся в укрытиях для усложнения их поражения. Каждая пораженная мишень записывается в табло пораженных мишеней и

появляется заново спустя некоторое время (рисунок 43)

Рисунок 42 – Табло количества пораженных мишеней

Рисунок 43 – Полигон с мишенями

После ознакомления со стендом с информацией по обращению с гранатами, пользователь может приступать к тренировкам. Для начала ему необходимо взять одну из гранат (рисунок 44)

Рисунок 44 – Взятая граната

Чтобы взвести гранату, необходимо выдернуть чеку. Для этого необходимо другой рукой взяться за чеку и дернуть ее в сторону. Она отсоединится от гранаты (рисунок 45)

Рисунок 45 – Отсоединение чеки

Выдернутую чеку можно вернуть на место, тогда взрыва гранаты не произойдет. Для этого чеку необходимо поднести к месту, где она была первоначально (рисунок 46)

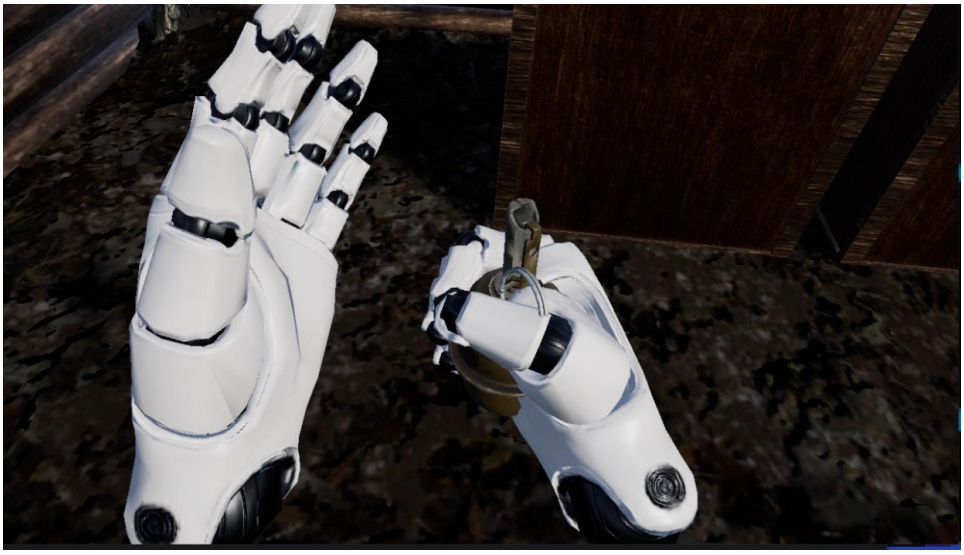


Рисунок 46 – Возвращение чеки

Для поражения целей на полигоне гранату нужно кинуть прямо к целям. Вынув чеку, это можно делать любым способом из представленных на стенде с обучающей информацией (рисунок 47).

Рисунок 48 – Бросок гранаты

После взведения гранаты, она взорвется спустя время, указанное в ее характеристиках. Если в этот момент рядом будут мишени, они будут поражены (Рисунок 49)

Рисунок 49 – Взрыв гранаты и ее поражение мишени

Каждая пораженная мишень засчитывается в табло (рисунок 50)

Рисунок 50 – Табло после поражения двух мишеней

Таким образом, программа выполняет все функции, описанные ранее, в полном объеме, картинка в шлеме плавная и с максимальным количеством кадров в секунду, что дает наибольший эффект погружения в процесс обучения и удовлетворяет требованиям оптимизации.

2 Технико-экономическое обоснование разработки программы

2.1 Введение

Целью технико-экономического обоснования разработки является количественное и качественное доказательство экономической целесообразности разработки и усовершенствования программы, а также определение организационно-экономических условий ее эффективного функционирования. Эффективность программного модуля определяется его качеством и эффективностью процесса разработки и сопровождения. Качество программного изделия определяется тремя составляющими:

- с точки зрения пользователя данного программного продукта;
- с позиции использования ресурсов и их оценки;
- по выполнению требований на программное изделие.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ПК, машинного времени, пропускной способности каналов передачи данных и др. – на стадии функционирования; времени разработки и денежных ресурсов – на стадии использования программного изделия). Содержание технико- экономического обоснования разработки программы заключается в следующем:

- расчет технико-экономических показателей и выбор базы сравнения;
- определение трудоемкости и стоимости программного обеспечения;
- расчет цены ПО;
- расчет капитальных и эксплуатационных затрат на разработку;
- определение показателей финансово-экономической эффективности.

Приведенные в данном разделе дипломного проекта результирующие таблицы, перечни данных, позволяют сопоставить результаты разработки и

затраты на нее, чтобы сделать вывод об эффективности проекта. Исходные данные для расчета экономических показателей приведены в таблице 2.

Таблица 2 - Исходные данные для расчета экономических показателей

Обозначение	Наименование показателя	Единицы измерения	Значение показателя		
СЭВМ	Стоимость ЭВМ и *VR* периферии	тыс. руб	120		
ДМ	Среднее количество рабочих дней в месяце	дни	25		
$\rho_н$	Норматив рентабельности	-	0,24		
$\omega_д$	Коэффициент, учитывающий дополнительную заработнуюплату разработчика программы	-	0,2		
$\omega_с$	Коэффициент, учитывающий начисления органамсоциального страхования	-	0,27		
$\omega_н$	Коэффициент, учитывающий накладные расходыорганизации	-	1,35		
qI	Количество I-задач, решаемых потребителем	Зад. год	60		
tM.B.I	Время решения I-ой задачи разработанной программой	Маш. час	1		
nn	Количество организаций, которые приобретут даннуюпрограмму	шт.	1г	2г	3г
			4	5	8
ZЭЛ	Тариф за 1 кВт/час	Руб.	3,44		
TC	Срок службы разработанной программы	год	5		
НДС	Налог на добавленную стоимость	%	20		
TP	Количество рабочих дней в году	дн	264		
NCM	Количество смен работы ЭВМ	-	1		
tCM	Продолжительность смены	ч	8		
α	Простои ЭВМ	%	10		
P	Мощность, потребляемая ЭВМ	кВт	0,4		
NCP	Среднее количество ремонтов в год	-	2		
SД	Стоимость деталей, заменяемых при ремонте	Руб.	2500		

2.2 Расчёт затрат на разработку программного обеспечения

Суммарные затраты на разработку проекта рассчитываются по следующей формуле:

$$S_{РП} = S_{ЗП} + S_{НАК}, \quad (1)$$

где $S_{ЗП}$ – затраты по заработной плате инженера-программиста;

$S_{НАК}$ – накладные расходы.

Затраты по заработной плате инженера-программиста рассчитываются поформуле:

$$S_{ЗП} = O_{ЗП} * (1+\omega_c) * (1+ \omega_д) * tp_i, \quad (2)$$

где ОЗП – основная заработная плата инженера-программиста за месяц (20000 руб.);

tp_i – время, необходимое для разработки программы программистом i-го разряда (чел.-мес.);

$\omega д$ – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

$\omega с$ – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика.

Система разрабатывалась 30 дней, если учесть, что в одном месяце 25 рабочих дней, то:

$$tp_{12} = 30/25 = 1{,}2 \text{ (чел./Мес)} \quad (3)$$

Таким образом, затраты по заработной плате инженера-программиста составят:

$$S_{ЗП} = 10000 * (1+0{,}268) * (1+0{,}20) * 1{,}2 = 18258{,}52 \text{ (руб.)}$$

Накладные затраты рассчитываются с учетом ωн – коэффициента, определяющего уровень накладных расходов организации по формуле:

$$S_{НАК} = O_{ЗП} * \omega_н * tp_{12}, \quad (4)$$

$$S_{НАК} = 20000 * 1{,}35 * 1{,}2 = 32400 \text{ (руб.)}$$

Таким образом, суммарные затраты на разработку системы составляют:

$$S_{РП} = 18258{,}52 + 32400 = 50658{,}52 \text{ (руб.)}$$

2.3 Расчет цены разработанной программы

Оптовая цена разработанной программы определяется по следующей

формуле:

$$З_П = S_{РП} + П, \quad (5)$$

где $З_П$ – оптовая цена (цена разработчика) (руб.);

$S_{РП}$ – суммарные затраты на разработку программы (руб.);

$П$ – прибыль, рассчитанная по формуле:

$$П = \rho_н * S_{РП}, \quad (6)$$

где $\rho_н$ – норматив рентабельности, учитывающий прибыль организации, разрабатывающей данную программу в долях ко всем затратам данной организации на разработку программы.

Итак,

$$З_П = S_{РП} * (1+ \rho_н), \quad (7)$$

$$З_П = 50658,52 * (1+ 0,24) = 62815,92 \text{ (руб.)}$$

Розничная цена программы рассчитывается с учетом налога на добавленную стоимость (НДС = 20%) по формуле:

$$З_{Пр} = З_П + Н_{ДС} = З_П + (З_N*20)/100= З_П * (1+0,20) \quad (8)$$

$$З_{Пр}=62815,92*(1+0,20) = 75379,1 \text{ (руб.)}$$

Выручка от продаж при условии nn – количество организаций, желающихприобрести программу, составит:

$$В = З_{Пр} * n_n \quad (9)$$

$$В1 = 75379,1 * 4 = 301516,41 \text{ (руб.)}$$

$$В2 = 75379,1* 5 = 376895,52 \text{ (руб.)}$$

$$В3 = 75379,1 * 8 = 603032,83 \text{ (руб.)}$$

2.4 Расчет капитальных вложений

Капиталовложения, связанные с работой ЭВМ рассчитываются по формуле:

$$К_{ЭВМ} = С_{ЭВМ} + S_З, \quad (10)$$

где $С_{ЭВМ}$ – стоимость ЭВМ (руб.);

$S_З$ – стоимость запасных частей (руб.).

Итак, произведем расчет коэффициентов, входящих в формулу расчетавеличины капиталовложений:

$$S_З = 0{,}15 * C_{ЭВМ}; \quad (11)$$

$$S_З = 0{,}15 * 120000 = 18000 \text{ (руб.)};$$

Капиталовложения в ЭВМ и *VR* периферию составляют:

$$K_{ЭВМ} = 120000 + 18000 = 138000 \text{ (руб.)}$$

2.5 Расчет эксплуатационных расходов

Эксплуатационные расходы на ЭВМ рассчитываются по формуле:

$$E = (T_{МВ} * E_ч) + Zn/Tc, \quad (12)$$

где $T_{МВ}$ – машинное время для решения задач с помощью разработанной программы, (маш.час/год);

$E_ч$ – эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ;

Zn – цена, по которой продается программа (руб.);

$T_С$ – срок службы программы (г).

Полезный фонд времени работы ЭВМ рассчитывается по формуле:

$$T_{ПОЛ} = T_{ОБЩ} * t_{СМ} * N_{СМ} * (1 - \alpha/100), \quad (13)$$

где $T_{ОБЩ}$ – общий фонд времени работы ЭВМ (дни); $T_{ОБЩ} = T_Р$; $N_{СМ}$ – количество смен работы ЭВМ;

$t_{СМ}$ –время одного рабочего дня (час);

α – простои ЭВМ (в % от общего фонда времени работы ЭВМ). Полезный фонд времени работы ЭВМ получим:

$$T_{ПОЛ} = 264 * 8 * 1 * (1 - 10/100) = 1900{,}8 \text{ (Маш. час /год)}.$$

Машинное время для решения задач с помощью данной программы рассчитывается по формуле:

$$T_{МВ} = qI * t_{МВI}, \quad (14)$$

где qI – количество I-задач, решаемых потребителем в год(шт.);

$t_{МВI}$ – время решения I-ой задачи, разработанной программой (Маш. час).

$$T_{МВ} = 60 * 1 = 60 \text{ (Маш. час / год)}.$$

Эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ, оцениваются по формуле:

$$E_ч = (A_о + S_{зп} + S_{эл} + R_{рт})/T_{пол}, \quad (15)$$

где $A_О$ – амортизационные отчисления (руб.);

$S_{ЗП}$ – затраты по заработной плате инженера в год (руб./год);

$S_{ЭЛ}$ – стоимость потребляемой энергии (руб.);

$R_{РМ}$ – затраты на ремонт ЭВМ (руб.);

$T_{ПОЛ}$ – полезный годовой фонд работы ЭВМ, (маш.час/год).

Амортизационные отчисления рассчитываются с учетом нормы амортизации ($Ан$ =12,5 %);

$$A_О = Ан * К_{ЭВМ} = 0{,}125 * 138000 = 17250 \text{ (руб.)}. \quad (16)$$

Затраты по заработной плате инженера за год рассчитывается по формуле:

$$S_{ЗП} = (1+ωс) * (1+ωд) * О_{ЗП} * 12, \quad (17)$$

где $ωс$ – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика;

$ωд$ – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

$О_{ЗП}$ – основная заработная плата инженера за месяц 8-го разряда (20000 руб.);

Рассчитываем годовые затраты по заработной плате и социальным отчислениям для инженера:

$$S_{ЗП} = (1+ 0{,}268) * (1+ 0{,}20) * 20000 * 12 = 365184 \text{ (руб./ год)}$$

Стоимость потребляемой энергии оценивается по формуле:

$$S_{ЭЛ} = P * T_{ПОЛ} * Z_{ЭЛ}, \quad (18)$$

где P – мощность, потребляемая ЭВМ (кВт);

$T_{ПОЛ}$ – полезный годовой фонд работы ЭВМ (маш.час/год);

$Z_{ЭЛ}$ – тариф за 1 кВт/час (руб. /кВт).

Итак, произведем расчет стоимости потребляемой энергии:

$$S_{ЭЛ} = 0,4 * 1900,8 * 3,44 = 2202,7 \text{ (руб.)}.$$

Затраты на ремонт ЭВМ вычисляются по формуле:

$$R_{PM} = N_{CP} * S_{Д}, \quad (19)$$

где N_{CP} – среднее количество ремонтов в год;

$S_{Д}$ – стоимость деталей, заменяемых при одном ремонте, в среднем.

$$R_{PM} = 2 * 2500 = 5000 \text{ (руб.)}.$$

Произведем вычисление эксплуатационных расходов, приходящихся на 1 час работы ЭВМ:

$$E_{ч} = (17250+365184+2202,7+5000)/1900,8 = 205 \text{ (руб./час)}.$$

Далее вычислим эксплуатационные расходы на ЭВМ:

$$E = (60 * 205) + 62815,92/5 = 24863,18 \text{ (руб.)}.$$

2.6 Определение расходов на интернет трафик

Для облегчения работы программиста и повышения его производительности в проекте предусмотрено использование глобальной сети интернет.

В настоящее время на пензенском рынке интернет очень много провайдеров предлагающие свои услуги. Цены, предлагаемые основными провайдерами, сведем в таблицу 3, для анализа и экономически целесообразного выбора поставщика услуги по интернет трафику.

Таблица 3 - Таблица интернет трафиков и провайдеров

Название провайдера	Предлагаемый тариф	Стоимость тарифа(руб.)	Скорость доступа(Мбит/с)
Ростелеком	Пакет-Для Впечатлений	699	100
МТС	Безлимитище	450	100
Дом.РУ	Скорость света	550	100

Т.к. вышеперечисленные провайдеры предлагают безлимитные интернет трафики с одинаковой скоростью доступа в сеть интернет, то

экономически целесообразно выбрать трафик с самой низкой стоимостью. Выбираем ПАО "МТС", тариф “Безлимитище”, с абонентской платой в месяц 450 рублей.

2.7 Технико-экономические показатели разработки

Обобщенные технико-экономические показатели разработки программы сведены в таблицу 4

Таблица 4 - Технико-экономические показатели разработки

Показатель	Значение
Капитальные вложения (руб.)	138000
Эксплуатационные расходы (руб.)	24863,18
Затраты на разработку ПО (руб.)	50658,52
Затраты на интернет трафик (руб.)	450
Затраты на заработную плату программисту (руб.)	30432
Суммарно (руб.)	244403,7

3 Безопасность жизнедеятельности. Эргономика рабочего места программиста

3.1 Характеристика условий труда программиста

Использование последних достижений технологического прогресса привел к изменению необходимых условий и требований производственного труда сотрудников. Работа программистов стала интенсивнее и требующей увеличенных трат физической и умственной энергии. Данная проблема требует комплексного решения вопроса в сферах оборудования рабочих мест, контроля уровня шума, света, воздушной среды.

В наши дни работа с ПК и *VR* шлемами интенсивно применяется во всех сферах работы человека. Такая деятельность подвергает человека влиянию ряда опасных электромагнитных излучений (СВЧ, УВЧ, ВЧ), шума, статики и других.

Взаимодействие с ПК и *VR* шлемом сопровождается увеличенной умственной нагрузкой на пользователя, постоянной зрительной работой и нагрузкой на мышцы шеи и рук при использовании шлема с контроллерами либо клавиатуры. Несомненная важность имеет правильное оборудование и размещение рабочих мест пользователей и разработчиков.

Также в процессе свой деятельности работникам следует помнить о правильном соблюдении режимов труда и отдыха. При пренебрежении данного режима у рабочих могут возникать различные симптомы, выражающиеся в головных болях, нарушениях сна, раздражительности, усталости, болях в пояснице, шее и руках.

Одними из критериев, по которым определяется пригодность рабочих пространств работников, являются требования к параметрам воздушной среды, требования к уровню шума и вибрации, требования к освещенности рабочих мест, требования к производственному оборудованию и требования к пользователям *VR*-шлемов.

3.2 Требования к параметрам воздушной среды

Климатические параметры на рабочих местах программистов в комнатах с компьютерами и *VR*-шлемами должны удовлетворять требованиям из таблицы 5

Таблица 5 - Климат рабочих помещений

Период года	Температура воздуха, С	Скорость движения воздуха, м/с	Относительная влажность воздуха, %
Холодный	22-24	до 0,1	40-60
Теплый	23-25	0,1-0,2	40-60

Воздух, находящийся в рабочих помещениях работников ЭВМ, должен быть профильтрован от пыли, загрязнений и микроорганизмов. Должно достигаться отсутствие патогенной микрофлоры.

Воздушное кондиционирование должно поддерживать параметры создаваемого микроклимата в установленных пределах во время всех сезонов года, очищать воздух от пыли и загрязнений, создавать необходимое давление в помещении для недопущения поступления неочищенного воздуха. Температура также должна быть не менее 19 градусов.

Температура воздуха в помещении должна регулироваться с учетом теплового потока от оборудования. Предпочтительным выбором оборудования должна быть техника с малой электрической мощностью. Оборудование техники должно быть установлено ток, чтобы тепловой поток не был направлен на рабочих. Следует ограничивать количество техники, находящейся в помещении.

Пол в помещениях должен быть ровный, из нескользкого материала, удобным для чистки и влажной уборки. Покрытия рабочих пространств должны обладать антистатическими свойствами и сохранять свои параметры

в процессе эксплуатации. В помещениях должна проводиться влажная уборка.

3.3 Требования к уровню шума и вибрации

Главными источниками шума в рабочих комнатах с установленными в них ЭВМ являются принтеры, плоттеры, вентиляторы, кондиционеры, вентиляционные системы, трансформаторы, множительная техника.

Для понижения уровня шума и вибрации в комнатах, находящееся в ней оборудование необходимо монтировать на специальные фундаменты и амортизирующие подкладки, предусмотренные нормативными документами.

Величина шума в рабочих комнатах не должен превышать уровень в 50 дБа. Нормирование шума производится путем применения малошумного оборудования, использования звукопоглощающих материалов (перфорированных панелей). Существует возможность использования акустических подвесных потолков.

Оборудование, уровень издаваемого шума которого превышает допустимые нормы, необходимо монтировать вне рабочих помещений.

3.4 Требования к монтажу производственного оборудования

Для обеспечения электропитанием ЭВМ и компьютеров в комнатах должна быть смонтирована отельная сеть электроснабжения.

Периферийные устройства компьютеров Должны подключаться исключительно к линии электроснабжения компьютерной сети.

Запрещается включение в линии электроснабжения компьютерной сети энергопотребителей, не относящихся к компьютерам (вентиляторы, лампы и т.п.).

Запрещается использовать оборудование, не подключенное к защитному заземлению, если оно предусмотрено конструкцией оборудования.

Запрещается использовать технику с открытыми корпусами, если это не предусмотрено правилами работы этого оборудования.

Проведение обслуживания рабочих мест программистов должно проводиться специально подготовленным персоналом, имеющим необходимую квалификацию техника (инженера).

Электрические розетки системы электропитания необходимо располагать так, чтобы кабели электрооборудования на рабочих местах не пересекали его, мешая работе.

3.5 Требования к пользованию *VR* шлемов

Важно не забывать о том, что *VR*-шлемы – хрупкое и дорогостоящее оборудование, требующего выполнения особых правил безопасности.

Любой *VR*-шлем (неважно какого производителя) представляет собой оборудование, изменяющее восприятие мира пользователем. Важно помнить, что чувства дезориентации и головокружения при использовании шлема обусловлены реакцией мозга пользователя на изображение. При снятии шлема данные симптомы проходят.

Не все *VR*-шлема подходят для использования в совокупности с обыкновенными очками. Им может как не найтись место в корпусе, либо будут мешать дужки у места креплений шлема. В таких случаях необходимо делать частые перерывы, чтобы избежать появления болей. Лучшим решением будет отказ от очков в пользу контактных линз во время пользования *VR*-шлема.

Перед началом работы убедитесь, что шлем надежно и удобно закреплен на голове, а кабели (при их наличии) не мешают вашим движениям. Крепления должны быть настроены так, чтобы исключать нежелательные качания шлема, ведь это может неблагоприятно воздействовать на восприятие картинки.

Многие *VR*-шлема имеют в себе функции регулирования фокусного и межзрачкового расстояния. Их необходимо настраивать персонально для каждого пользования. Перед началом активной работы в шлеме, посидите в нем несколько минут, привыкая к картинке и убеждаясь в надежности крепления.

При симптомах острого беспокойства, тошноты, головокружения, ряби в глазах или болях в голове необходимо снять шлем. При первых сессиях работы со шлемом это может произойти уже в первых минутах после запуска. Человеческий мозг быстро привыкает к новой реальности при регулярном пользовании *VR*-шлема, неприятные ощущения будут возникать намного реже. Однако даже опытным пользователям необходимо делать небольшие перерывы (по 10-15 минут) после каждого часа работы в нем.

3.6 Требования к освещению рабочих мест

Одним из важнейших параметров в мероприятиях создания оптимальных условий труда является уровень световой среды помещения. Это рационализация организации совокупности естественного и искусственного освещения на рабочих местах.

Комнаты, используемые для размещения рабочих мест пользователей ПК и *VR*-шлемов, должны иметь как естественное, так и искусственное освещение.

Осуществление искусственного освещения комнат с рабочими местами должно осуществляться с помощью световых проемов, ориентированных на север, либо северо-восток.

В том случае, если ориентация производится в другие стороны, необходимо предусмотреть наличие средств регулирования интенсивности поступаемого естественного освещения.

Окна в комнатах с расположенными в них компьютерах и средствами ЭВМ необходимо оборудовать средствами регулирования интенсивности естественного освещения типа занавесок, жалюзи.

3.7 Расчет освещения рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения. Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественномусвету;

- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);

- обладают повышенной светоотдачей (в 3-4 раза выше, чем у лампнакаливания);

- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15м^2, ширина которой 5м, высота - 3 м. Воспользуемся методом светового потока. Для определения количества светильников определим световой поток, падающийна поверхность по формуле:

$$F = E*K*S*Z/n, \quad (20)$$

где F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность

будет E = 300Лк;

S - площадь освещаемого помещения (в нашем случае S = 20м^2);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1…1,2, пусть Z = 1,2); K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае K = 1,5);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; характеризуется коэффициентами отражения от стен (*РС*) и потолка (*РП*), значение коэффициентов *РС* и *РП* были указаны выше: *РС* =40%, *РП* = 60%. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = S / h*(A+B), \quad (21)$$

где S - площадь помещения, S = 20 м2;

h - расчетная высота подвеса, h = 3 м;

A- ширина помещения, A = 4 м;

B - длина помещения, B = 5 м.

Подставив значения получаем:

$$I = 20/3*(4+5) = 60$$

Зная индекс помещения I, по таблице находим n = 0,22

Подставим все значения в формулу для определения светового потока F:

$$F = 300*1,5*20*1,2/0,22 = 49090 \text{ Лм}$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световойпоток которых F = 4320 Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = F/Fл, \quad (22)$$

где N - определяемое число ламп;

F - световой поток, F = 49090 Лм;

$Fл$- световой поток лампы, $Fл$ = 4320 Лм.

$$N = 49090/4320 = 11 \text{ штук.}$$

Заключение

В процессе выполнения данной выпускной квалификационной работы проведены анализ требований, проектирование и разработка, тестирование симулятора полигона для метания снарядов на основе виртуальной реальности.

В процессе анализа требований, были сформированы требования к системе. Проведён анализ вариантов использования. Разработана модель предметной области, представленная диаграммой классов. В результате выполнения выпускной квалификационной работы реализован симулятор полигона для метания снарядов на основе виртуальной реальности.

Проект реализован на движке *Unreal Engine* 4 с применением блюпринтов, в качестве технологии программирования объектов системы. На основе анализа, полученных результатов, можно сделать вывод, что все функции приложения отрабатываются корректно в штатном режиме.

Таким образом, требования технического задания были выполнены в полном объёме и в установленные сроки.

Список использованных источников

1. Обучение метанию ручных осколочных гранат [Электронный ресурс]: Режим доступа: *http://sarapylov-a-e.ucoz.ru/index/obuchenie_metaniju_granat/0-82*

2. Метание гранат: техника и правила. [Электронный ресурс]: Режим доступа: *https://rf-gk.ru/metanie-granat-tehnika-i-pravila-ruchnye-granaty-sohranenie-ravnovesiya-posle/*

3. Виртуальная и дополненная реальность: принцип работы и применение технологий VR и AR [Электронный ресурс]: Режим доступа: *https://cryptos.tv/virtualnaya-i-dopolnennaya-realnost-princip-raboty-i-primenenie-texnologij-vr-i-ar/*

4. Что такое виртуальная и дополненная реальность? Принцип работы VR и AR технологий [Электронный ресурс]: Режим доступа: *https://mining-cryptocurrency.ru/vr-ar-virtualnaya-dopolnennaya-realnost/*

5. Виртуальная реальность как способ изучения реального мира [Электронный ресурс] URL: *https://molotokrus.ru/virtualnaya-realnost-kak-sposob-izucheniya-realnogo-mira/*

6. Ready, Play, Go: лучшие VR-шлемы 2021 года [Электронный ресурс]: Режим доступа: *https://skillbox.ru/media/gamedev/luchshie-vrshlemy-2021-goda/*

7. Шлем виртуальной реальности для компьютера топ 2021 [Электронный ресурс]: Режим доступа: *https://everychild.ru/reyting/shlem-virtualnoy-realnosti-dlya-kompyutera-top-2021/*

8. Лучшие современные движки для разработок приложений [Электронный ресурс]: Режим доступа: *https://habr.com/ru/company/timeweb/blog/659891/*

Приложение А

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Диаграмма классов

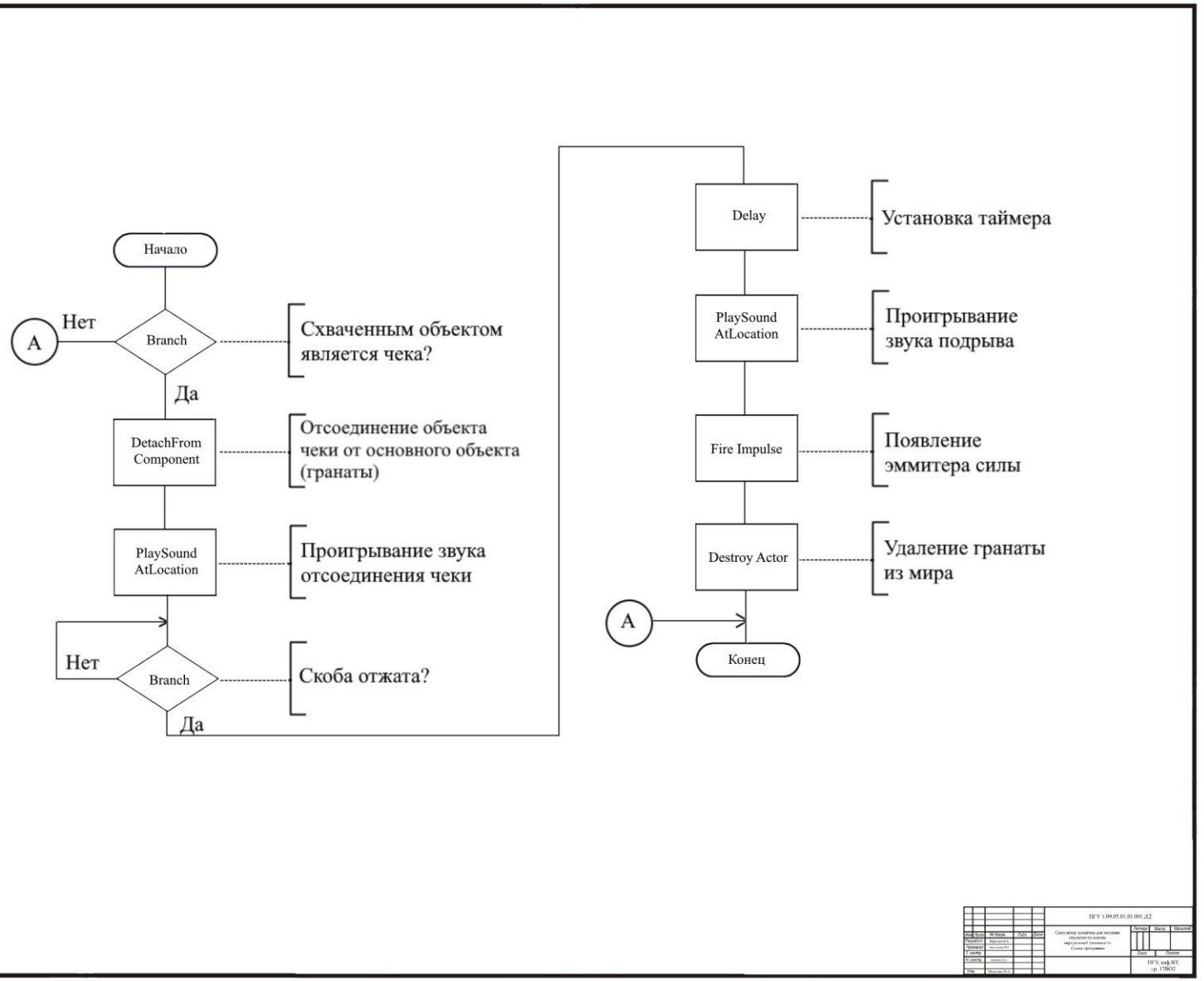
Button
- Button Mesh
- Button Comp
+ On Button State Changed
Counter
- DefaultRoot
- TextRender
- CountRender
+ Event Tick
Target_DM
- Destructible Component
+ On Component Fracture
Grenade_Poligon
Grenade_Actor
- GrenadeMesh
- NeckCapsule
- Stopper
- PrimaryGrip
- RadialForce
+ Event On Child Grip
+ Event on Child Grip Release
+ OnComponentBeginOverlap
+ Event On Used
Vive_PawnCharacter
- VRRoot
- Left Motion Controller
- Right Motion Controller
- VRReplicated Camera
- GrabSphere Left
- GrabSphere Right
+ Event BeginPlay
+ SetupOnPossesion
+ Escape
+ InputAxis ControllerMovementRight
+ InputAxis ControllerMovementLeft
+ OnGrippedObject Left
+ OnGrippedObject Right
+ InputAction PrimaryGripLeft
+ InputAction PrimaryGripRight
+ Trigger GripOrDrop
+ InputAction UseHeldObjectLeft
+ InputAction UseHeldObjectRight
+ TryGrabClient

Приложение Б

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Схема программы (Триггер снаряда)


Начало
A
Нет
Branch
Схваченным объектом является чека?
Да
DetachFrom Component
Отсоединение объекта чеки от основного объекта (гранаты)
PlaySound AtLocation
Проигрывание звука отсоединения чеки
Нет
Branch
Скоба отжата?
Да
Delay
Установка таймера
PlaySound AtLocation
Проигрывание звука подрыва
Fire Impulse
Появление эммитера силы
Destroy Actor
Удаление гранаты из мира
A
Конец

Приложение В

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Схема программы (Инициализация сессии)

Начало
EventBeginPlay
Запуск сессии
SetGripComponents
Определение, какой контроллер является правым, а какой - левым
SetTrackingOrigin
Установка центра отслеживания действий
SetSpectratorScreenMode
Подключение экрана наблюдателя
SetMovementHands
Распределение ролей контроллеров
SpawnActor
Появление контроллеров
AttachActor toComponent
Присоединение контроллеров к телу пользователя
Конец

Приложение Г

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Схема программы (Триггер мишени)

Начало
GetAllActor ofClass
Получение списка всех объектов класса "Счетчик"
CastToCounter
Обращение к переменным объекта "Счетчик"
Hits++
Увеличение переменной Hits счетчика на единицу
Delay
Установка таймера
SpawnActor
Восстановление разрушенной мишени
Конец

Приложение Д

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Листинг программы

```
#include "Grenade_Poligon.h"
#include "Vive_PawnCharacter__pf522378901.h"
#include "GeneratedCodeHelpers.h"
#include "Runtime/Engine/Classes/Engine/InputActionDelegateBinding.h"
#include "Runtime/Engine/Classes/Engine/ComponentDelegateBinding.h"
#include "Runtime/Engine/Classes/Engine/InputKeyDelegateBinding.h"
#include "Runtime/Engine/Classes/Engine/InputAxisDelegateBinding.h"
#include "Runtime/Engine/Classes/Engine/SimpleConstructionScript.h"
#include "Runtime/Engine/Classes/Engine/SCS_Node.h"
#include "Runtime/Engine/Classes/Components/StaticMeshComponent.h"
#include "Runtime/Engine/Classes/Components/MeshComponent.h"
#include "Runtime/Engine/Classes/Components/PrimitiveComponent.h"
#include "Runtime/Engine/Classes/Components/SceneComponent.h"
#include "Runtime/Engine/Classes/Components/ActorComponent.h"
#include "Runtime/Engine/Classes/Engine/EngineBaseTypes.h"
#include "Runtime/Engine/Classes/GameFramework/Actor.h"
#include "Runtime/Engine/Classes/GameFramework/DamageType.h"
#include "Runtime/PhysicsCore/Public/PhysicalMaterials/PhysicalMaterial.h"
#include "Runtime/PhysicsCore/Public/PhysicsSettingsEnums.h"
#include
"Runtime/PhysicsCore/Public/PhysicalMaterials/PhysicalMaterialPropertyBase.h"
#include "Runtime/PhysicsCore/Public/Chaos/ChaosEngineInterface.h"
#include "Runtime/Engine/Classes/GameFramework/Controller.h"
#include "Runtime/Engine/Classes/GameFramework/Pawn.h"
#include "Runtime/Engine/Classes/GameFramework/PawnMovementComponent.h"
#include "Runtime/Engine/Classes/GameFramework/NavMovementComponent.h"
#include "Runtime/Engine/Classes/GameFramework/MovementComponent.h"
#include "Runtime/Engine/Classes/GameFramework/PhysicsVolume.h"
#include "Runtime/Engine/Classes/GameFramework/Volume.h"
#include "Runtime/Engine/Classes/Engine/Brush.h"
#include "Runtime/Engine/Classes/Components/BrushComponent.h"
#include "Runtime/Engine/Classes/PhysicsEngine/BodySetup.h"
#include "Runtime/PhysicsCore/Public/BodySetupCore.h"
#include "Runtime/PhysicsCore/Public/BodySetupEnums.h"
#include "Runtime/Engine/Classes/PhysicsEngine/AggregateGeom.h"
#include "Runtime/Engine/Classes/PhysicsEngine/SphereElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/ShapeElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/BoxElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/SphylElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/ConvexElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/TaperedCapsuleElem.h"
#include "Runtime/Engine/Classes/PhysicsEngine/BodyInstance.h"
```

```
#include "Runtime/PhysicsCore/Public/BodyInstanceCore.h"
#include "Runtime/Engine/Classes/Engine/BrushBuilder.h"
#include "Runtime/Engine/Classes/AI/Navigation/NavigationTypes.h"
#include "Runtime/Engine/Classes/GameFramework/PlayerState.h"
#include "Runtime/Engine/Classes/GameFramework/Info.h"
#ifdef _MSC_VER
#pragma warning (push)
#pragma warning (disable : 4883)
#endif
PRAGMA_DISABLE_DEPRECATION_WARNINGS
PRAGMA_DISABLE_OPTIMIZATION
AVive_PawnCharacter_C__pf522378901::AVive_PawnCharacter_C__pf522378901(const
FObjectInitializer& ObjectInitializer) : Super(ObjectInitializer)
{

      auto __Local__0 = CastChecked<UVRRootComponent>(this-
>GetDefaultSubobjectByName(TEXT("CollisionCylinder")),
ECastCheckedType::NullAllowed);
      auto __Local__1 = CastChecked<UVRCharacterMovementComponent>(this-
>GetDefaultSubobjectByName(TEXT("CharMoveComp")),
ECastCheckedType::NullAllowed);
      auto __Local__2 = CastChecked<USkeletalMeshComponent>(this-
>GetDefaultSubobjectByName(TEXT("CharacterMesh0")),
ECastCheckedType::NullAllowed);
      auto __Local__3 = CastChecked<USceneComponent>(this-
>GetDefaultSubobjectByName(TEXT("NetSmoother")),
ECastCheckedType::NullAllowed);
      auto __Local__4 = CastChecked<USceneComponent>(this-
>GetDefaultSubobjectByName(TEXT("VRProxy")), ECastCheckedType::NullAllowed);
      auto __Local__5 = CastChecked<UReplicatedVRCameraComponent>(this-
>GetDefaultSubobjectByName(TEXT("VR Replicated Camera")),
ECastCheckedType::NullAllowed);
      auto __Local__6 = CastChecked<UParentRelativeAttachmentComponent>(this-
>GetDefaultSubobjectByName(TEXT("Parent Relative Attachment")),
ECastCheckedType::NullAllowed);
      auto __Local__7 = CastChecked<UGripMotionControllerComponent>(this-
>GetDefaultSubobjectByName(TEXT("Left Grip Motion Controller")),
ECastCheckedType::NullAllowed);
      auto __Local__8 = CastChecked<UGripMotionControllerComponent>(this-
>GetDefaultSubobjectByName(TEXT("Right Grip Motion Controller")),
ECastCheckedType::NullAllowed);
      if(__Local__0)
```

```
	{
		// --- Default subobject 'CollisionCylinder' //
		auto __Local__9 = CastChecked<UBodySetup>(__Local__0-
>GetDefaultSubobjectByName(TEXT("BodySetup_0")),
ECastCheckedType::NullAllowed);
		if(__Local__9)
		{
			// --- Default subobject 'BodySetup_0' //
			// --- END default subobject 'BodySetup_0' //
		}
		__Local__0->WalkingCollisionOverride =
ECollisionChannel::ECC_GameTraceChannel3;
		__Local__0->BodyInstance.LoadProfileData(false);
		static TWeakFieldPtr<FProperty> __Local__11{};
		const FProperty* __Local__10 = __Local__11.Get();
		if (nullptr == __Local__10)
		{
			__Local__10 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
			check(__Local__10);
			__Local__11 = __Local__10;
		}
		(((FBoolProperty*)__Local__10)-
>SetPropertyValue_InContainer((__Local__0), false, 0));
		// --- END default subobject 'CollisionCylinder' //
	}
	if(__Local__1)
	{
		// --- Default subobject 'CharMoveComp' //
		__Local__1->VRWallSlideScaler = 0.500000f;
		__Local__1->VRClimbingStepHeight = 60.000000f;
		__Local__1->VRClimbingEdgeRejectDistance = 10.000000f;
		__Local__1->VRClimbingStepUpMultiplier = 2.000000f;
		__Local__1->MaxAcceleration = 20048.000000f;
		__Local__1->BrakingDecelerationWalking = 20048.000000f;
		static TWeakFieldPtr<FProperty> __Local__13{};
		const FProperty* __Local__12 = __Local__13.Get();
		if (nullptr == __Local__12)
		{
			__Local__12 = (UNavMovementComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bUseAccelerationForPaths")));
			check(__Local__12);
```

```
                __Local__13 = __Local__12;
            }
            (((FBoolProperty*)__Local__12)-
>SetPropertyValue_InContainer((__Local__1), false, 0));
            static TWeakFieldPtr<FProperty> __Local__15{};
            const FProperty* __Local__14 = __Local__15.Get();
            if (nullptr == __Local__14)
            {
                __Local__14 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                check(__Local__14);
                __Local__15 = __Local__14;
            }
            (((FBoolProperty*)__Local__14)-
>SetPropertyValue_InContainer((__Local__1), false, 0));
            // --- END default subobject 'CharMoveComp' //
      }
      if(__Local__2)
      {
            // --- Default subobject 'CharacterMesh0' //
            __Local__2->VisibilityBasedAnimTickOption =
EVisibilityBasedAnimTickOption::AlwaysTickPose;
            auto& __Local__16 =
(*(AccessPrivateProperty<UParentRelativeAttachmentComponent*>((__Local__2),
USceneComponent::__PPO__AttachParent() )));
            __Local__16 = __Local__6;
            static TWeakFieldPtr<FProperty> __Local__18{};
            const FProperty* __Local__17 = __Local__18.Get();
            if (nullptr == __Local__17)
            {
                __Local__17 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                check(__Local__17);
                __Local__18 = __Local__17;
            }
            (((FBoolProperty*)__Local__17)-
>SetPropertyValue_InContainer((__Local__2), false, 0));
            // --- END default subobject 'CharacterMesh0' //
      }
      if(__Local__3)
      {
            // --- Default subobject 'NetSmoother' //
```

```
            auto& __Local__19 =
(*(AccessPrivateProperty<UVRRootComponent*>((__Local__3),
USceneComponent::__PPO__AttachParent() )));
            __Local__19 = __Local__0;
            static TWeakFieldPtr<FProperty> __Local__21{};
            const FProperty* __Local__20 = __Local__21.Get();
            if (nullptr == __Local__20)
            {
                  __Local__20 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__20);
                  __Local__21 = __Local__20;
            }
            (((FBoolProperty*)__Local__20)-
>SetPropertyValue_InContainer((__Local__3), false, 0));
            // --- END default subobject 'NetSmoother' //
      }
      if(__Local__4)
      {
            // --- Default subobject 'VRProxy' //
            auto& __Local__22 =
(*(AccessPrivateProperty<USceneComponent*>((__Local__4),
USceneComponent::__PPO__AttachParent() )));
            __Local__22 = __Local__3;
            static TWeakFieldPtr<FProperty> __Local__24{};
            const FProperty* __Local__23 = __Local__24.Get();
            if (nullptr == __Local__23)
            {
                  __Local__23 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__23);
                  __Local__24 = __Local__23;
            }
            (((FBoolProperty*)__Local__23)-
>SetPropertyValue_InContainer((__Local__4), false, 0));
            // --- END default subobject 'VRProxy' //
      }
      if(__Local__5)
      {
            // --- Default subobject 'VR Replicated Camera' //
            __Local__5->bSampleVelocityInWorldSpace = true;
            __Local__5->bSmoothReplicatedMotion = true;
```

```
            __Local__5->NetUpdateRate = 200.000000f;
            __Local__5->FieldOfView = 42.857143f;
            __Local__5->bLockToHmd = false;
            __Local__5->PostProcessBlendWeight = 0.000000f;
            __Local__5-
>PostProcessSettings.bOverride_DepthOfFieldFocalDistance = true;
            __Local__5->PostProcessSettings.AutoExposureMethod =
EAutoExposureMethod::AEM_Manual;
            __Local__5->PostProcessSettings.DepthOfFieldFocalDistance =
1238.971558f;
            auto& __Local__25 =
(*(AccessPrivateProperty<USceneComponent*>((__Local__5),
USceneComponent::__PPO__AttachParent() )));
            __Local__25 = __Local__4;
            static TWeakFieldPtr<FProperty> __Local__27{};
            const FProperty* __Local__26 = __Local__27.Get();
            if (nullptr == __Local__26)
            {
                  __Local__26 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__26);
                  __Local__27 = __Local__26;
            }
            (((FBoolProperty*)__Local__26)-
>SetPropertyValue_InContainer((__Local__5), false, 0));
            // --- END default subobject 'VR Replicated Camera' //
      }
      if(__Local__6)
      {
            // --- Default subobject 'Parent Relative Attachment' //
            auto& __Local__28 =
(*(AccessPrivateProperty<USceneComponent*>((__Local__6),
USceneComponent::__PPO__AttachParent() )));
            __Local__28 = __Local__4;
            static TWeakFieldPtr<FProperty> __Local__30{};
            const FProperty* __Local__29 = __Local__30.Get();
            if (nullptr == __Local__29)
            {
                  __Local__29 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__29);
                  __Local__30 = __Local__29;
```

```
            }
            (((FBoolProperty*)__Local__29)-
>SetPropertyValue_InContainer((__Local__6), false, 0));
            // --- END default subobject 'Parent Relative Attachment' //
      }
      if(__Local__7)
      {
            // --- Default subobject 'Left Grip Motion Controller' //
            __Local__7->ControllerNetUpdateRate = 60.000000f;
            __Local__7->bSmoothReplicatedMotion = true;
            static TWeakFieldPtr<FProperty> __Local__32{};
            const FProperty* __Local__31 = __Local__32.Get();
            if (nullptr == __Local__31)
            {
                  __Local__31 = (UPrimitiveComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bGenerateOverlapEvents")));
                  check(__Local__31);
                  __Local__32 = __Local__31;
            }
            (((FBoolProperty*)__Local__31)-
>SetPropertyValue_InContainer((__Local__7), false, 0));
            auto& __Local__33 =
(*(AccessPrivateProperty<USceneComponent*>((__Local__7),
USceneComponent::__PPO__AttachParent() )));
            __Local__33 = __Local__4;
            static TWeakFieldPtr<FProperty> __Local__35{};
            const FProperty* __Local__34 = __Local__35.Get();
            if (nullptr == __Local__34)
            {
                  __Local__34 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__34);
                  __Local__35 = __Local__34;
            }
            (((FBoolProperty*)__Local__34)-
>SetPropertyValue_InContainer((__Local__7), false, 0));
            // --- END default subobject 'Left Grip Motion Controller' //
      }
      if(__Local__8)
      {
            // --- Default subobject 'Right Grip Motion Controller' //
            __Local__8->ControllerNetUpdateRate = 60.000000f;
```

```
            __Local__8->bSmoothReplicatedMotion = true;
            static TWeakFieldPtr<FProperty> __Local__37{};
            const FProperty* __Local__36 = __Local__37.Get();
            if (nullptr == __Local__36)
            {
                  __Local__36 = (UPrimitiveComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bGenerateOverlapEvents")));
                  check(__Local__36);
                  __Local__37 = __Local__36;
            }
            (((FBoolProperty*)__Local__36)-
>SetPropertyValue_InContainer((__Local__8), false, 0));
            auto& __Local__38 =
(*(AccessPrivateProperty<USceneComponent*>((__Local__8),
USceneComponent::__PPO__AttachParent() )));
            __Local__38 = __Local__4;
            auto& __Local__39 = (*(AccessPrivateProperty<FVector
>((__Local__8), USceneComponent::__PPO__RelativeLocation() )));
            __Local__39 = FVector(0.000000, 0.000008, 0.000000);
            static TWeakFieldPtr<FProperty> __Local__41{};
            const FProperty* __Local__40 = __Local__41.Get();
            if (nullptr == __Local__40)
            {
                  __Local__40 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
                  check(__Local__40);
                  __Local__41 = __Local__40;
            }
            (((FBoolProperty*)__Local__40)-
>SetPropertyValue_InContainer((__Local__8), false, 0));
            // --- END default subobject 'Right Grip Motion Controller' //
      }
      bpv__Body__pf =
CreateDefaultSubobject<UStaticMeshComponent>(TEXT("Body"));
      bpv__HandMeshxLeft__pfG =
CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("HandMesh-Left"));
      bpv__GrabSphereLeft__pf =
CreateDefaultSubobject<USphereComponent>(TEXT("GrabSphereLeft"));
      bpv__TextR__pf =
CreateDefaultSubobject<UTextRenderComponent>(TEXT("TextR"));
      bpv__VOIPTalker__pf =
CreateDefaultSubobject<UVOIPTalker>(TEXT("VOIPTalker"));
```

```
	bpv__HandMeshxRight__pfG =
CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("HandMesh-Right"));
	bpv__TextL__pf =
CreateDefaultSubobject<UTextRenderComponent>(TEXT("TextL"));
	bpv__GrabSphereRight__pf =
CreateDefaultSubobject<USphereComponent>(TEXT("GrabSphereRight"));
	bpv__AimLeft__pf =
CreateDefaultSubobject<USceneComponent>(TEXT("AimLeft"));
	bpv__AimRight__pf =
CreateDefaultSubobject<USceneComponent>(TEXT("AimRight"));
	bpv__Body__pf->CreationMethod = EComponentCreationMethod::Native;
	bpv__Body__pf->AttachToComponent(__Local__6,
FAttachmentTransformRules::KeepRelativeTransform );
	auto& __Local__42 = (*(AccessPrivateProperty<UStaticMesh*
>((bpv__Body__pf), UStaticMeshComponent::__PPO__StaticMesh() )));
	__Local__42 =
CastChecked<UStaticMesh>(CastChecked<UDynamicClass>(AVive_PawnCharacter_C__pf
522378901::StaticClass())->UsedAssets[0], ECastCheckedType::NullAllowed);
	bpv__Body__pf->OverrideMaterials = TArray<UMaterialInterface*> ();
	bpv__Body__pf->OverrideMaterials.Reserve(1);
	bpv__Body__pf-
>OverrideMaterials.Add(CastChecked<UMaterialInterface>(CastChecked<UDynamicCl
ass>(AVive_PawnCharacter_C__pf522378901::StaticClass())->UsedAssets[1],
ECastCheckedType::NullAllowed));
	auto& __Local__43 =
(*(AccessPrivateProperty<TEnumAsByte<ECollisionChannel> >(&(bpv__Body__pf-
>BodyInstance), FBodyInstance::__PPO__ObjectType() )));
	__Local__43 = ECollisionChannel::ECC_Pawn;
	auto& __Local__44 =
(*(AccessPrivateProperty<TEnumAsByte<ECollisionEnabled::Type>
>(&(bpv__Body__pf->BodyInstance), FBodyInstance::__PPO__CollisionEnabled()
)));
	__Local__44 = ECollisionEnabled::Type::QueryOnly;
	auto& __Local__45 = (*(AccessPrivateProperty<FCollisionResponse
>(&(bpv__Body__pf->BodyInstance), FBodyInstance::__PPO__CollisionResponses()
)));
	auto& __Local__46 = (*(AccessPrivateProperty<TArray<FResponseChannel>
>(&(__Local__45), FCollisionResponse::__PPO__ResponseArray() )));
	__Local__46 = TArray<FResponseChannel> ();
	__Local__46.AddUninitialized(9);
	FResponseChannel::StaticStruct()-
>InitializeStruct(__Local__46.GetData(), 9);
```

```
	auto& __Local__47 = __Local__46[0];
	__Local__47.Channel = FName(TEXT("WorldStatic"));
	__Local__47.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__48 = __Local__46[1];
	__Local__48.Channel = FName(TEXT("WorldDynamic"));
	__Local__48.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__49 = __Local__46[2];
	__Local__49.Channel = FName(TEXT("Pawn"));
	__Local__49.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__50 = __Local__46[3];
	__Local__50.Channel = FName(TEXT("Visibility"));
	__Local__50.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__51 = __Local__46[4];
	__Local__51.Channel = FName(TEXT("Camera"));
	__Local__51.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__52 = __Local__46[5];
	__Local__52.Channel = FName(TEXT("PhysicsBody"));
	__Local__52.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__53 = __Local__46[6];
	__Local__53.Channel = FName(TEXT("Vehicle"));
	__Local__53.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__54 = __Local__46[7];
	__Local__54.Channel = FName(TEXT("Destructible"));
	__Local__54.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__55 = __Local__46[8];
	__Local__55.Channel = FName(TEXT("VRTraceChannel"));
	__Local__55.Response = ECollisionResponse::ECR_Ignore;
	bpv__Body__pf->SetCollisionProfileName(FName(TEXT("Custom")));
	auto& __Local__56 = (*(AccessPrivateProperty<FVector >((bpv__Body__pf),
USceneComponent::__PPO__RelativeLocation() ));
	__Local__56 = FVector(-15.000000, 0.000000, -61.598076);
	auto& __Local__57 = (*(AccessPrivateProperty<FVector >((bpv__Body__pf),
USceneComponent::__PPO__RelativeScale3D() ));
	__Local__57 = FVector(0.500000, 0.500000, 0.100000);
	static TWeakFieldPtr<FProperty> __Local__59{};
	const FProperty* __Local__58 = __Local__59.Get();
	if (nullptr == __Local__58)
	{
		__Local__58 = (UActorComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bCanEverAffectNavigation")));
		check(__Local__58);
		__Local__59 = __Local__58;
```

```
	}
	(((FBoolProperty*)__Local__58)-
>SetPropertyValue_InContainer((bpv__Body__pf), true, 0));
	bpv__HandMeshxLeft__pfG->CreationMethod =
EComponentCreationMethod::Native;
	bpv__HandMeshxLeft__pfG->AttachToComponent(__Local__7,
FAttachmentTransformRules::KeepRelativeTransform );
	bpv__HandMeshxLeft__pfG->AnimClass =
URightHand_AnimBP_C__pf558596347::StaticClass();
	bpv__HandMeshxLeft__pfG->ClothingSimulationFactory =
UClothingSimulationFactoryNv::StaticClass();
	bpv__HandMeshxLeft__pfG->SkeletalMesh =
CastChecked<USkeletalMesh>(CastChecked<UDynamicClass>(AVive_PawnCharacter_C__
pf522378901::StaticClass())->UsedAssets[2], ECastCheckedType::NullAllowed);
	bpv__HandMeshxLeft__pfG->VisibilityBasedAnimTickOption =
EVisibilityBasedAnimTickOption::AlwaysTickPoseAndRefreshBones;
	bpv__HandMeshxLeft__pfG->CastShadow = false;
	bpv__HandMeshxLeft__pfG->SetCollisionProfileName(FName(TEXT("Custom")));
	auto& __Local__60 = (*(AccessPrivateProperty<FVector
>((bpv__HandMeshxLeft__pfG), USceneComponent::__PPO__RelativeLocation() )));
	__Local__60 = FVector(-5.478077, 0.000000, 5.859251);
	auto& __Local__61 = (*(AccessPrivateProperty<FRotator
>((bpv__HandMeshxLeft__pfG), USceneComponent::__PPO__RelativeRotation() )));
	__Local__61 = FRotator(-60.000000, 0.000018, 89.999947);
	auto& __Local__62 = (*(AccessPrivateProperty<FVector
>((bpv__HandMeshxLeft__pfG), USceneComponent::__PPO__RelativeScale3D() )));
	__Local__62 = FVector(1.000000, 1.000000, -1.000000);
	(((FBoolProperty*)__Local__58)-
>SetPropertyValue_InContainer((bpv__HandMeshxLeft__pfG), false, 0));
	bpv__GrabSphereLeft__pf->CreationMethod =
EComponentCreationMethod::Native;
	bpv__GrabSphereLeft__pf->AttachToComponent(__Local__7,
FAttachmentTransformRules::KeepRelativeTransform );
	auto& __Local__63 = (*(AccessPrivateProperty<float
>((bpv__GrabSphereLeft__pf), USphereComponent::__PPO__SphereRadius() )));
	__Local__63 = 4.000000f;
	bpv__GrabSphereLeft__pf->AreaClass = UNavArea_Obstacle::StaticClass();
	auto& __Local__64 =
(*(AccessPrivateProperty<TEnumAsByte<ECollisionChannel>
>(&(bpv__GrabSphereLeft__pf->BodyInstance),
FBodyInstance::__PPO__ObjectType() )));
	__Local__64 = ECollisionChannel::ECC_Pawn;
```

```
	auto& __Local__65 =
(*(AccessPrivateProperty<TEnumAsByte<ECollisionEnabled::Type>
>(&(bpv__GrabSphereLeft__pf->BodyInstance),
FBodyInstance::__PPO__CollisionEnabled() )));
	__Local__65 = ECollisionEnabled::Type::QueryAndPhysics;
	auto& __Local__66 = (*(AccessPrivateProperty<FCollisionResponse
>(&(bpv__GrabSphereLeft__pf->BodyInstance),
FBodyInstance::__PPO__CollisionResponses() )));
	auto& __Local__67 = (*(AccessPrivateProperty<TArray<FResponseChannel>
>(&(__Local__66), FCollisionResponse::__PPO__ResponseArray() )));
	__Local__67 = TArray<FResponseChannel> ();
	__Local__67.AddUninitialized(9);
	FResponseChannel::StaticStruct()-
>InitializeStruct(__Local__67.GetData(), 9);
	auto& __Local__68 = __Local__67[0];
	__Local__68.Channel = FName(TEXT("WorldStatic"));
	__Local__68.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__69 = __Local__67[1];
	__Local__69.Channel = FName(TEXT("WorldDynamic"));
	__Local__69.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__70 = __Local__67[2];
	__Local__70.Channel = FName(TEXT("Pawn"));
	__Local__70.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__71 = __Local__67[3];
	__Local__71.Channel = FName(TEXT("Visibility"));
	__Local__71.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__72 = __Local__67[4];
	__Local__72.Channel = FName(TEXT("Camera"));
	__Local__72.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__73 = __Local__67[5];
	__Local__73.Channel = FName(TEXT("PhysicsBody"));
	__Local__73.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__74 = __Local__67[6];
	__Local__74.Channel = FName(TEXT("Vehicle"));
	__Local__74.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__75 = __Local__67[7];
	__Local__75.Channel = FName(TEXT("Destructible"));
	__Local__75.Response = ECollisionResponse::ECR_Overlap;
	auto& __Local__76 = __Local__67[8];
	__Local__76.Channel = FName(TEXT("VRTraceChannel"));
	__Local__76.Response = ECollisionResponse::ECR_Overlap;
	bpv__GrabSphereLeft__pf->SetCollisionProfileName(FName(TEXT("Custom")));
```

```
	bpv__GrabSphereLeft__pf->bHiddenInGame = false;
	(((FBoolProperty*)__Local__58)-
>SetPropertyValue_InContainer((bpv__GrabSphereLeft__pf), true, 0));
	bpv__TextR__pf->CreationMethod = EComponentCreationMethod::Native;
	bpv__TextR__pf->AttachToComponent(__Local__8,
FAttachmentTransformRules::KeepRelativeTransform );
	bpv__TextR__pf->Text = FTextStringHelper::CreateFromBuffer(
TEXT("NSLOCTEXT(\"\", \"570D2CDF4030A61594B16A8CEAE28B29\", \"R\")")   );
	bpv__TextR__pf->HorizontalAlignment = EHorizTextAligment::EHTA_Center;
	bpv__TextR__pf->VerticalAlignment =
EVerticalTextAligment::EVRTA_TextCenter;
	bpv__TextR__pf->TextRenderColor = FColor(2, 2, 7, 255);
	bpv__TextR__pf->WorldSize = 3.000000f;
	auto& __Local__77 = (*(AccessPrivateProperty<FVector >((bpv__TextR__pf),
USceneComponent::__PPO__RelativeLocation() )));
	__Local__77 = FVector(10.000000, -0.001106, 2.106884);
	auto& __Local__78 = (*(AccessPrivateProperty<FRotator
>((bpv__TextR__pf), USceneComponent::__PPO__RelativeRotation() )));
	__Local__78 = FRotator(82.999741, -179.999390, -0.000000);
	bpv__TextR__pf->bHiddenInGame = true;
	(((FBoolProperty*)__Local__58)-
>SetPropertyValue_InContainer((bpv__TextR__pf), false, 0));
	bpv__VOIPTalker__pf->CreationMethod = EComponentCreationMethod::Native;
	bpv__VOIPTalker__pf->Settings.AttenuationSettings =
CastChecked<USoundAttenuation>(CastChecked<UDynamicClass>(AVive_PawnCharacter
_C__pf522378901::StaticClass())->UsedAssets[3],
ECastCheckedType::NullAllowed);
	(((FBoolProperty*)__Local__58)-
>SetPropertyValue_InContainer((bpv__VOIPTalker__pf), false, 0));
	bpv__HandMeshxRight__pfG->CreationMethod =
EComponentCreationMethod::Native;
	bpv__HandMeshxRight__pfG->AttachToComponent(__Local__8,
FAttachmentTransformRules::KeepRelativeTransform );
	bpv__HandMeshxRight__pfG->AnimClass =
URightHand_AnimBP_C__pf558596347::StaticClass();
	bpv__HandMeshxRight__pfG->ClothingSimulationFactory =
UClothingSimulationFactoryNv::StaticClass();
	bpv__HandMeshxRight__pfG->SkeletalMesh =
CastChecked<USkeletalMesh>(CastChecked<UDynamicClass>(AVive_PawnCharacter_C__
pf522378901::StaticClass())->UsedAssets[2], ECastCheckedType::NullAllowed);
	bpv__HandMeshxRight__pfG->VisibilityBasedAnimTickOption =
EVisibilityBasedAnimTickOption::AlwaysTickPoseAndRefreshBones;
```

```
	static TWeakFieldPtr<FProperty> __Local__80{};
	const FProperty* __Local__79 = __Local__80.Get();
	if (nullptr == __Local__79)
	{
		__Local__79 = (UPrimitiveComponent::StaticClass())-
>FindPropertyByName(FName(TEXT("bGenerateOverlapEvents")));
		check(__Local__79);
		__Local__80 = __Local__79;
	}
	(((FBoolProperty*)__Local__79)-
>SetPropertyValue_InContainer((bpv__HandMeshxRight__pfG), true, 0));
	bpv__HandMeshxRight__pfG->CastShadow = false;
	auto& __Local__81 = (*(AccessPrivateProperty<FVector
>((bpv__HandMeshxRight__pfG), USceneComponent::__PPO__RelativeLocation() )));
	__Local__81 = FVector(-5.478077, 0.000000, 5.859251);
	auto& __Local__82 = (*(AccessPrivateProperty<FRotator
>((bpv__HandMeshxRight__pfG), USceneComponent::__PPO__RelativeRotation() )));
	__Local__82 = FRotator(-59.999893, 0.000013, 89.999969);

}
PRAGMA_ENABLE_OPTIMIZATION
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_0(int32 bpp__EntryPoint__pf)
{
	bool bpfv__CallFunc_DropGrip_ReturnValue__pf{};
	bool bpfv__CallFunc_IsValid_ReturnValue_1__pf{};
	bool bpfv__CallFunc_DropAndSocketGrip_ReturnValue__pf{};
	bool bpfv__CallFunc_DoesImplementInterface_ReturnValue__pf{};
	bool bpfv__CallFunc_DropObjectByInterface_ReturnValue__pf{};
	int32 __CurrentState = bpp__EntryPoint__pf;
	do
	{
		switch( __CurrentState )
		{
		case 216:
			{
			}
		case 217:
			{
				bpf__ShouldSocketGrip__pf(/*out*/
b01__K2Node_CustomEvent_GripToDrop__pf, /*out*/
```

```
b0l__CallFunc_ShouldSocketGrip_ShouldSocket__pf, /*out*/
b0l__CallFunc_ShouldSocketGrip_SocketParent__pf, /*out*/
b0l__CallFunc_ShouldSocketGrip_RelativeTransform__pf, /*out*/
b0l__CallFunc_ShouldSocketGrip_OptionalSocketName__pf);
                  }
            case 218:
                  {
                        if (!b0l__CallFunc_ShouldSocketGrip_ShouldSocket__pf)
                        {
                              __CurrentState = 220;
                              break;
                        }
                  }
            case 219:
                  {
                        if(::IsValid(b0l__K2Node_CustomEvent_Controller__pf))
                        {
                              bpfv__CallFunc_DropAndSocketGrip_ReturnValue__pf
= b0l__K2Node_CustomEvent_Controller__pf-
>UGripMotionControllerComponent::DropAndSocketGrip(b0l__K2Node_CustomEvent_Gr
ipToDrop__pf, b0l__CallFunc_ShouldSocketGrip_SocketParent__pf,
b0l__CallFunc_ShouldSocketGrip_OptionalSocketName__pf,
b0l__CallFunc_ShouldSocketGrip_RelativeTransform__pf, true);
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 220:
                  {
                  }
            case 221:
                  {
                  }
            case 222:
                  {
                        bpfv__CallFunc_IsValid_ReturnValue_1__pf =
UKismetSystemLibrary::IsValid(b0l__K2Node_CustomEvent_GripToDrop__pf.GrippedO
bject);
                        if (!bpfv__CallFunc_IsValid_ReturnValue_1__pf)
                        {
                              __CurrentState = 227;
                              break;
```

```
                        }
                  }
            case 223:
                  {
                  }
            case 224:
                  {
                        bpfv__CallFunc_DoesImplementInterface_ReturnValue__pf
=
UKismetSystemLibrary::DoesImplementInterface(b0l__K2Node_CustomEvent_GripToDr
op__pf.GrippedObject, UVRGripInterface::StaticClass());
                        if
(!bpfv__CallFunc_DoesImplementInterface_ReturnValue__pf)
                        {
                              __CurrentState = 226;
                              break;
                        }
                  }
            case 225:
                  {
                        if(::IsValid(b0l__K2Node_CustomEvent_Controller__pf))
                        {

      bpfv__CallFunc_DropObjectByInterface_ReturnValue__pf =
b0l__K2Node_CustomEvent_Controller__pf-
>UGripMotionControllerComponent::DropObjectByInterface(((UObject*)nullptr),
b0l__K2Node_CustomEvent_GripToDrop__pf.GripID,
b0l__K2Node_CustomEvent_Angle_Vel__pf,
b0l__K2Node_CustomEvent_Linear_Vel__pf);
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 226:
                  {
                        if(::IsValid(b0l__K2Node_CustomEvent_Controller__pf))
                        {
                              bpfv__CallFunc_DropGrip_ReturnValue__pf =
b0l__K2Node_CustomEvent_Controller__pf-
>UGripMotionControllerComponent::DropGrip(b0l__K2Node_CustomEvent_GripToDrop_
_pf, true, b0l__K2Node_CustomEvent_Angle_Vel__pf,
b0l__K2Node_CustomEvent_Linear_Vel__pf);
```

```
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 227:
                  {
                        __CurrentState = 226;
                        break;
                  }
            default:
                  break;
            }
      } while( __CurrentState != -1 );
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_1(int32 bpp__EntryPoint__pf)
{
      bool bpfv__CallFunc_IsInVREditorPreviewOrGame_ReturnValue__pf{};
      bool bpfv__CallFunc_LoadControllerProfileByName_ReturnValue__pf{};
      bool bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_1__pf{};
      bool bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_2__pf{};
      bool bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_3__pf{};
      int32 __CurrentState = bpp__EntryPoint__pf;
      do
      {
            switch( __CurrentState )
            {
            case 1:
                  {
                        UKismetSystemLibrary::Delay(this, 0.200000,
FLatentActionInfo(2, -792976935,
TEXT("ExecuteUbergraph_Vive_PawnCharacter_1"), this));
                        __CurrentState = -1;
                        break;
                  }
            case 2:
                  {
                  }
            case 3:
                  {
```

```
      UOpenXRExpansionFunctionLibrary::GetXRMotionControllerType(/*out*/
b0l__CallFunc_GetXRMotionControllerType_TrackingSystemName__pf, /*out*/
b0l__CallFunc_GetXRMotionControllerType_DeviceType__pf, /*out*/
b0l__CallFunc_GetXRMotionControllerType_Result__pf);
                  }
            case 4:
                  {
                        b0l__K2Node_SwitchEnum_CmpSuccess_12__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(b0l__CallFunc_GetXRM
otionControllerType_Result__pf),
static_cast<uint8>(EBPXRResultSwitch::OnSucceeded));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_12__pf)
                        {
                              __CurrentState = 5;
                              break;
                        }
                        b0l__K2Node_SwitchEnum_CmpSuccess_12__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(b0l__CallFunc_GetXRM
otionControllerType_Result__pf),
static_cast<uint8>(EBPXRResultSwitch::OnFailed));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_12__pf)
                        {
                              __CurrentState = 1;
                              break;
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 5:
                  {
                        bpv__CurrentControllerTypeXR__pf =
b0l__CallFunc_GetXRMotionControllerType_DeviceType__pf;
                  }
            case 6:
                  {
                        b0l__K2Node_SwitchEnum_CmpSuccess__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentControll
erTypeXR__pf),
static_cast<uint8>(EBPOpenXRControllerDeviceType::DT_ValveIndexController));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess__pf)
                        {
```

```
                        __CurrentState = 7;
                        break;
                  }
                  b0l__K2Node_SwitchEnum_CmpSuccess__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentControll
erTypeXR__pf),
static_cast<uint8>(EBPOpenXRControllerDeviceType::DT_ViveController));
                  if (!b0l__K2Node_SwitchEnum_CmpSuccess__pf)
                  {
                        __CurrentState = 8;
                        break;
                  }
                  b0l__K2Node_SwitchEnum_CmpSuccess__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentControll
erTypeXR__pf),
static_cast<uint8>(EBPOpenXRControllerDeviceType::DT_OculusTouchController));
                  if (!b0l__K2Node_SwitchEnum_CmpSuccess__pf)
                  {
                        __CurrentState = 9;
                        break;
                  }
                  b0l__K2Node_SwitchEnum_CmpSuccess__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentControll
erTypeXR__pf),
static_cast<uint8>(EBPOpenXRControllerDeviceType::DT_MicrosoftMotionControlle
r));
                  if (!b0l__K2Node_SwitchEnum_CmpSuccess__pf)
                  {
                        __CurrentState = 10;
                        break;
                  }
                  __CurrentState = -1;
                  break;
            }
      case 7:
            {

      bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_3__pf =
UVRGlobalSettings::LoadControllerProfileByName(FName(TEXT("KnucklesEV2")),
true);
                  __CurrentState = -1;
                  break;
```

```
                }
            case 8:
                {

      bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_1__pf =
UVRGlobalSettings::LoadControllerProfileByName(FName(TEXT("Vive_Wands")),
true);
                    __CurrentState = -1;
                    break;
                }
            case 9:
                {

      bpfv__CallFunc_LoadControllerProfileByName_ReturnValue__pf =
UVRGlobalSettings::LoadControllerProfileByName(FName(TEXT("Oculus_Touch_Home"
)), true);
                    __CurrentState = -1;
                    break;
                }
            case 10:
                {

      bpfv__CallFunc_LoadControllerProfileByName_ReturnValue_2__pf =
UVRGlobalSettings::LoadControllerProfileByName(FName(TEXT("Windows_MR")),
true);
                    __CurrentState = -1;
                    break;
                }
            case 249:
                {
                }
            case 250:
                {

      bpfv__CallFunc_IsInVREditorPreviewOrGame_ReturnValue__pf =
UVRExpansionFunctionLibrary::IsInVREditorPreviewOrGame();
                    if
(!bpfv__CallFunc_IsInVREditorPreviewOrGame_ReturnValue__pf)
                    {
                        __CurrentState = -1;
                        break;
                    }
```

```
                        __CurrentState = 3;
                        break;
                  }
            default:
                  break;
            }
      } while( __CurrentState != -1 );
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_2(int32 bpp__EntryPoint__pf)
{
      bool bpfv__CallFunc_Not_PreBool_ReturnValue__pf{};
      bool bpfv__CallFunc_IsLocallyControlled_ReturnValue_6__pf{};
      check(bpp__EntryPoint__pf == 256);
      bpfv__CallFunc_Not_PreBool_ReturnValue__pf =
UKismetMathLibrary::Not_PreBool(b0l__K2Node_Event_bNewSeatedMode__pf);
      if(::IsValid(VRMovementReference))
      {
            VRMovementReference->NavAgentProps.bCanSwim =
bpfv__CallFunc_Not_PreBool_ReturnValue__pf;
      }
      bpfv__CallFunc_IsLocallyControlled_ReturnValue_6__pf =
IsLocallyControlled();
      if (!bpfv__CallFunc_IsLocallyControlled_ReturnValue_6__pf)
      {
            return; //KCST_EndOfThreadIfNot
      }
      bpf__SetVehicleMode__pf(b0l__K2Node_Event_bNewSeatedMode__pf, /*out*/
b0l__CallFunc_SetVehicleMode_IsVR__pf);
      return; //KCST_EndOfThread
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_3(int32 bpp__EntryPoint__pf)
{
      int32 __CurrentState = bpp__EntryPoint__pf;
      do
      {
            switch( __CurrentState )
            {
            case 129:
```

```
                  {
                  }
            case 130:
                  {
                        if (!bpv__bRightHandMovement__pf)
                        {
                              __CurrentState = 132;
                              break;
                        }
                  }
            case 131:
                  {

      bpf__HandleCurrentMovementInput__pf(b0l__K2Node_InputAxisEvent_AxisValue
__pf, RightMotionController, LeftMotionController);
                        __CurrentState = -1;
                        break;
                  }
            case 132:
                  {

      bpf__HandleTurnInput__pf(b0l__K2Node_InputAxisEvent_AxisValue__pf);
                        __CurrentState = -1;
                        break;
                  }
            default:
                  break;
            }
      } while( __CurrentState != -1 );
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_4(int32 bpp__EntryPoint__pf)
{
      FTransform bpfv__CallFunc_MakeTransform_ReturnValue__pf{};
      bool bpfv__CallFunc_IsLocallyControlled_ReturnValue__pf{};
      bool bpfv__CallFunc_IsLocallyControlled_ReturnValue_2__pf{};
      bool bpfv__CallFunc_HasAuthority_ReturnValue__pf{};
      bool bpfv__CallFunc_IsLocallyControlled_ReturnValue_7__pf{};
      TArray< int32, TInlineAllocator<8> > __StateStack;

      int32 __CurrentState = bpp__EntryPoint__pf;
```

```
	do
	{
		switch( __CurrentState )
		{
		case 133:
			{
				__StateStack.Push(134);
				__CurrentState = 144;
				break;
			}
		case 134:
			{
				__StateStack.Push(135);
				__CurrentState = 138;
				break;
			}
		case 135:
			{
			}
		case 136:
			{
				bpf__InitTeleportControllers_Event__pf();
			}
		case 137:
			{
				bpf__CheckSpawnGraspingHands__pf();
				__CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
				break;
			}
		case 138:
			{
				bpfv__CallFunc_MakeTransform_ReturnValue__pf =
UKismetMathLibrary::MakeTransform(FVector(0.000000,0.000000,0.000000),
FRotator(0.000000,0.000000,0.000000), FVector(1.000000,1.000000,1.000000));

	b0l__CallFunc_BeginDeferredActorSpawnFromClass_ReturnValue_1__pf =
UGameplayStatics::BeginDeferredActorSpawnFromClass(this,
ABP_Teleport_Controller_C__pf522378901::StaticClass(),
bpfv__CallFunc_MakeTransform_ReturnValue__pf,
ESpawnActorCollisionHandlingMethod::AlwaysSpawn, this);
			}
```

```
            case 139:
                  {

      UKismetSystemLibrary::SetObjectPropertyByName(b0l__CallFunc_BeginDeferre
dActorSpawnFromClass_ReturnValue_1__pf,
FName(TEXT("OwningMotionController")), RightMotionController);
                  }
            case 140:
                  {
                        bpfv__CallFunc_IsLocallyControlled_ReturnValue__pf =
IsLocallyControlled();

      UKismetSystemLibrary::SetBoolPropertyByName(b0l__CallFunc_BeginDeferredA
ctorSpawnFromClass_ReturnValue_1__pf, FName(TEXT("bIsLocal")),
bpfv__CallFunc_IsLocallyControlled_ReturnValue__pf);
                  }
            case 141:
                  {
                        bpfv__CallFunc_MakeTransform_ReturnValue__pf =
UKismetMathLibrary::MakeTransform(FVector(0.000000,0.000000,0.000000),
FRotator(0.000000,0.000000,0.000000), FVector(1.000000,1.000000,1.000000));
                        b0l__CallFunc_FinishSpawningActor_ReturnValue_1__pf =
CastChecked<ABP_Teleport_Controller_C__pf522378901>(UGameplayStatics::FinishS
pawningActor(b0l__CallFunc_BeginDeferredActorSpawnFromClass_ReturnValue_1__pf
, bpfv__CallFunc_MakeTransform_ReturnValue__pf),
ECastCheckedType::NullAllowed);
                  }
            case 142:
                  {
                        bpv__TeleportControllerRight__pf =
b0l__CallFunc_FinishSpawningActor_ReturnValue_1__pf;
                  }
            case 143:
                  {
                        if(::IsValid(bpv__TeleportControllerRight__pf))
                        {
                              bpv__TeleportControllerRight__pf-
>AActor::K2_AttachToComponent((*(AccessPrivateProperty<UCapsuleComponent*
>((this), ACharacter::__PPO__CapsuleComponent() ))), FName(TEXT("None")),
EAttachmentRule::SnapToTarget, EAttachmentRule::SnapToTarget,
EAttachmentRule::SnapToTarget, false);
                        }
```

```
                        __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                        break;
                  }
            case 144:
                  {
                        bpfv__CallFunc_MakeTransform_ReturnValue__pf =
UKismetMathLibrary::MakeTransform(FVector(0.000000,0.000000,0.000000),
FRotator(0.000000,0.000000,0.000000), FVector(1.000000,1.000000,1.000000));

      b0l__CallFunc_BeginDeferredActorSpawnFromClass_ReturnValue__pf =
UGameplayStatics::BeginDeferredActorSpawnFromClass(this,
ABP_Teleport_Controller_C__pf522378901::StaticClass(),
bpfv__CallFunc_MakeTransform_ReturnValue__pf,
ESpawnActorCollisionHandlingMethod::AlwaysSpawn, this);
                  }
            case 145:
                  {

      UKismetSystemLibrary::SetObjectPropertyByName(b0l__CallFunc_BeginDeferre
dActorSpawnFromClass_ReturnValue__pf, FName(TEXT("OwningMotionController")),
LeftMotionController);
                  }
            case 146:
                  {
                        bpfv__CallFunc_IsLocallyControlled_ReturnValue_7__pf =
IsLocallyControlled();

      UKismetSystemLibrary::SetBoolPropertyByName(b0l__CallFunc_BeginDeferredA
ctorSpawnFromClass_ReturnValue__pf, FName(TEXT("bIsLocal")),
bpfv__CallFunc_IsLocallyControlled_ReturnValue_7__pf);
                  }
            case 147:
                  {
                        bpfv__CallFunc_MakeTransform_ReturnValue__pf =
UKismetMathLibrary::MakeTransform(FVector(0.000000,0.000000,0.000000),
FRotator(0.000000,0.000000,0.000000), FVector(1.000000,1.000000,1.000000));
                        b0l__CallFunc_FinishSpawningActor_ReturnValue__pf =
CastChecked<ABP_Teleport_Controller_C__pf522378901>(UGameplayStatics::FinishS
pawningActor(b0l__CallFunc_BeginDeferredActorSpawnFromClass_ReturnValue__pf,
bpfv__CallFunc_MakeTransform_ReturnValue__pf),
ECastCheckedType::NullAllowed);
```

```
                }
            case 148:
                {
                    bpv__TeleportControllerLeft__pf =
b0l__CallFunc_FinishSpawningActor_ReturnValue__pf;
                }
            case 149:
                {
                    if(::IsValid(bpv__TeleportControllerLeft__pf))
                    {
                        bpv__TeleportControllerLeft__pf-
>AActor::K2_AttachToComponent((*(AccessPrivateProperty<UCapsuleComponent*
>((this), ACharacter::__PPO__CapsuleComponent() ))), FName(TEXT("None")),
EAttachmentRule::SnapToTarget, EAttachmentRule::SnapToTarget,
EAttachmentRule::SnapToTarget, false);
                    }
                    __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                    break;
                }
            case 150:
                {
                    __CurrentState = (__StateStack.Num() > 0) ?
__StateStack.Pop(/*bAllowShrinking=*/ false) : -1;
                    break;
                }
            case 200:
                {
                }
            case 201:
                {
                    bpf__SetGripComponents__pf(bpv__GrabSphereLeft__pf,
bpv__GrabSphereRight__pf);
                }
            case 202:
                {
                    bpfv__CallFunc_IsLocallyControlled_ReturnValue_2__pf =
IsLocallyControlled();
                    if
(!bpfv__CallFunc_IsLocallyControlled_ReturnValue_2__pf)
                    {
                        __CurrentState = 133;
```

```
                        break;
                    }
                }
            case 203:
                {

    UHeadMountedDisplayFunctionLibrary::SetTrackingOrigin(EHMDTrackingOrigin
::Stage);
                }
            case 204:
                {

    UHeadMountedDisplayFunctionLibrary::SetSpectatorScreenMode(ESpectatorScr
eenMode::SingleEyeCroppedToFill);
                }
            case 205:
                {

    bpf__SetMovementHands__pf(bpv__bRightHandMovement__pf);
                    __CurrentState = 133;
                    break;
                }
            case 206:
                {
                }
            case 207:
                {
                }
            case 208:
                {
                    bpfv__CallFunc_HasAuthority_ReturnValue__pf =
AActor::HasAuthority();
                    if (!bpfv__CallFunc_HasAuthority_ReturnValue__pf)
                    {
                        __CurrentState = 200;
                        break;
                    }
                    __CurrentState = 150;
                    break;
                }
            case 270:
                {
```

```
                        __CurrentState = 201;
                        break;
                  }
            case 271:
                  {
                        __CurrentState = 206;
                        break;
                  }
            default:
                  check(false); // Invalid state
                  break;
            }
      } while( __CurrentState != -1 );
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_5(int32 bpp__EntryPoint__pf)
{
      int32 __CurrentState = bpp__EntryPoint__pf;
      do
      {
            switch( __CurrentState )
            {
            case 125:
                  {
                  }
            case 126:
                  {
                        if (!bpv__bRightHandMovement__pf)
                        {
                              __CurrentState = 128;
                              break;
                        }
                  }
            case 127:
                  {

      bpf__HandleTurnInput__pf(b0l__K2Node_InputAxisEvent_AxisValue_1__pf);
                        __CurrentState = -1;
                        break;
                  }
            case 128:
```

```
                  {

      bpf__HandleCurrentMovementInput__pf(b0l__K2Node_InputAxisEvent_AxisValue
_1__pf, LeftMotionController, RightMotionController);
                        __CurrentState = -1;
                        break;
                  }
            default:
                  break;
            }
      } while( __CurrentState != -1 );
}
void
AVive_PawnCharacter_C__pf522378901::bpf__ExecuteUbergraph_Vive_PawnCharacter_
_pf_6(int32 bpp__EntryPoint__pf)
{
      bool bpfv__CallFunc_BooleanOR_ReturnValue__pf{};
      bool bpfv__CallFunc_BooleanOR_ReturnValue_2__pf{};
      int32 __CurrentState = bpp__EntryPoint__pf;
      do
      {
            switch( __CurrentState )
            {
            case 121:
                  {
                        b0l__K2Node_SwitchEnum_CmpSuccess_3__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentMovement
Mode__pf), static_cast<uint8>(E__MovementMode__pf::NewEnumerator0));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_3__pf)
                        {
                              __CurrentState = 122;
                              break;
                        }
                        b0l__K2Node_SwitchEnum_CmpSuccess_3__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentMovement
Mode__pf), static_cast<uint8>(E__MovementMode__pf::NewEnumerator1));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_3__pf)
                        {
                              __CurrentState = 122;
                              break;
                        }
```

```
                        b0l__K2Node_SwitchEnum_CmpSuccess_3__pf =
UKismetMathLibrary::NotEqual_ByteByte(static_cast<uint8>(bpv__CurrentMovement
Mode__pf), static_cast<uint8>(E__MovementMode__pf::NewEnumerator11));
                        if (!b0l__K2Node_SwitchEnum_CmpSuccess_3__pf)
                        {
                              __CurrentState = 122;
                              break;
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 122:
                  {
                        bpfv__CallFunc_BooleanOR_ReturnValue__pf =
UKismetMathLibrary::BooleanOR(bpv__IsHandClimbing__pf,
bpv__bIsOutOfBody__pf);
                        bpfv__CallFunc_BooleanOR_ReturnValue_2__pf =
UKismetMathLibrary::BooleanOR(bpfv__CallFunc_BooleanOR_ReturnValue__pf,
bpv__DisableMovement__pf);
                        if (!bpfv__CallFunc_BooleanOR_ReturnValue_2__pf)
                        {
                              __CurrentState = 123;
                              break;
                        }
                        __CurrentState = -1;
                        break;
                  }
            case 123:
                  {
                        bpf__SetTeleporterActive__pf(EControllerHand::Left,
true);
                  }
            case 124:
                  {
                        bpf__SetTeleporterActive__pf(EControllerHand::Right,
false);
                        __CurrentState = -1;
                        break;
                  }
            case 292:
                  {
                  }
```

```
            case 293:
                  {
                        b01__Temp_struct_Variable_7__pf =
b01__K2Node_InputActionEvent_Key_14__pf;
                  }
            case 294:
                  {
                        if (!bpv__bRightHandMovement__pf)
                        {
                              __CurrentState = 121;
                              break;
                        }
                        __CurrentState = -1;
                        break;
                  }
            default:
                  break;
            }
      } while( __CurrentState != -1 );
}
PRAGMA_ENABLE_DEPRECATION_WARNINGS
#ifdef _MSC_VER
#pragma warning (pop)
#endif
```

Приложение Е

(обязательное)

Симулятор полигона для метания снарядов на основе виртуальной реальности

Презентация

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ДИПЛОМНЫЙ ПРОЕКТ НА ТЕМУ

СИМУЛЯТОР ПОЛИГОНА ДЛЯ МЕТАНИЯ СНАРЯДОВ НА ОСНОВЕ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Студент – Бирюзов Д.А.

Руководитель – Бикташев Р.А.

- **В наши дни качественная подготовка и усвоение необходимых навыков и умений личным составом является одной из первостепенных целей Вооруженных Сил. С ростом экономических и технических параметров страны появляется необходимость совершенствования не только технологической базы войск, но и методов обучения военнослужащих.**

- Виртуальная реальность – это совокупность технического и программного обеспечения, создающая виртуальный мир, передающийся человеку через его зрение, осязание и слух. Объединение и комплексное воздействие этих факторов на чувства человека и определяет собой интерактивность виртуального мира.

- Критерии:
  - - правдоподобность;
  - - интерактивность;
  - - возможность изучения;
  - - эффект присутствия;
  - - бесперебойность.

- Виды:
  - - с полным погружением;
  - - без погружения;
  - - с совместной инфраструктурой;
  - - на основе интернет технологий;
  - - дополненная реальность.

ЛУЧШИЕ МОДЕЛИ НА СЕГОДНЯШНИЙ ДЕНЬ

- - Valve Index
- - Oculus Quest
- - HP Reverb G2
- - Microsoft Hololens

ЦЕЛИ И ВОЗМОЖНОСТИ ПРОЕКТА

Цель - разработать приложение на основе устройств виртуальной реальности для симуляции полигона для метания снарядов

- Возможности проекта:
  - - взаимодействие с объектами;
  - - передвижение в пространстве;
  - - метание и подрыв снарядов;
  - - наличие нескольких типов снарядов;
  - - наличие специальных мишеней;
  - - наличие обучающих стендов с полезной информацией.

СРЕДА РАЗРАБОТКИ – UNREAL ENGINE 4

Интерфейс игрового движка UE4:

СИСТЕМА BLUEPRINT

- Блупринты — это скриптовая система в Unreal Engine 4, которая представляет собой визуальный интерфейс для создания элементов геймплея. Система очень гибкая и очень мощная, и позволяет дизайнерам использовать концепцию и почти полный потанциал программирования.

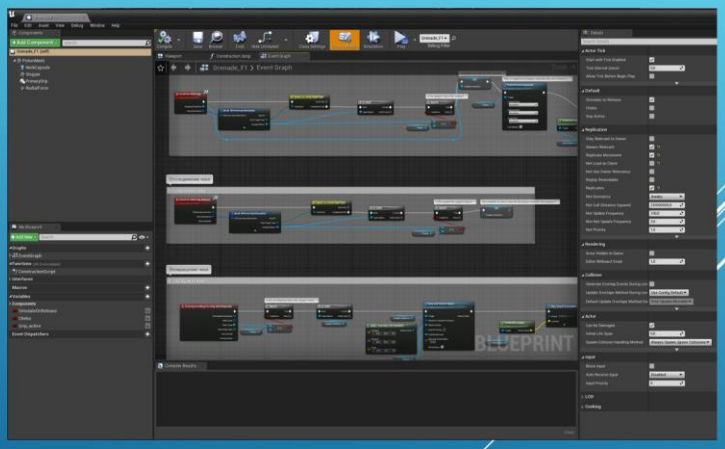
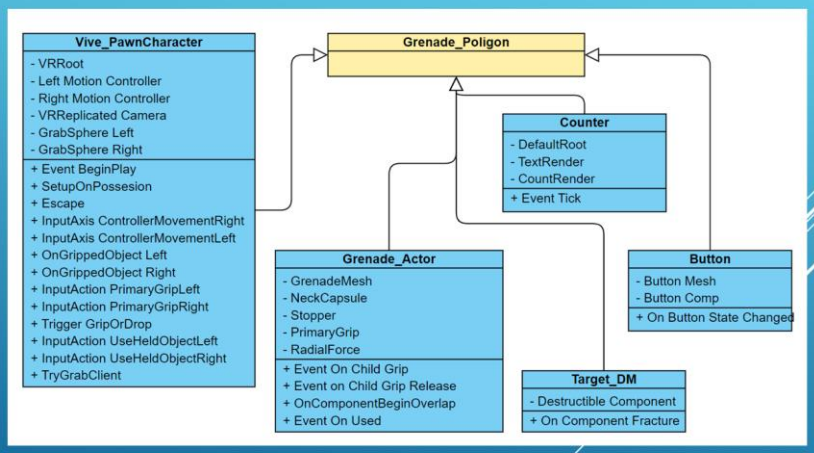


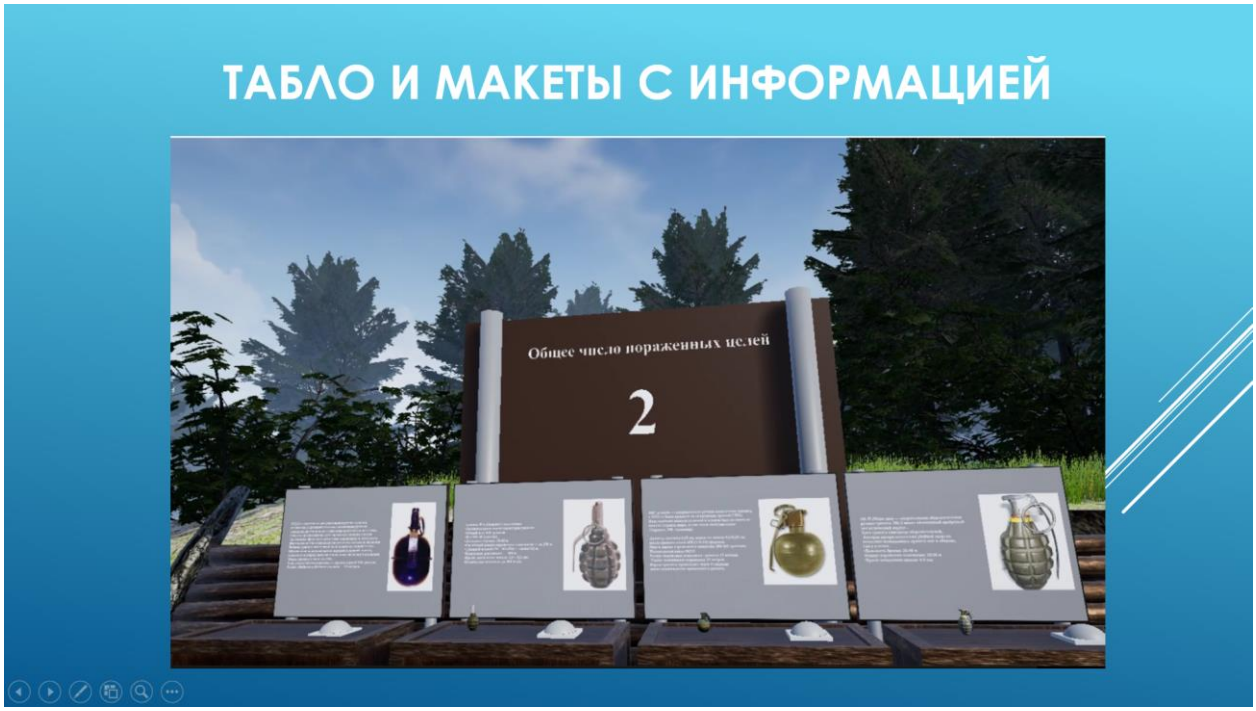
ДИАГРАММА КЛАССОВ

- Проект включает в себя целую сеть огромного числа различных элементов и объектов, исключая возможность полного представления из-за масштабов, однако имеется возможность выделить основные классы и их атрибуты/функции



ОСНОВНАЯ ЛОКАЦИЯ

РАБОТА С ГРАНАТАМИ


ТАБЛО И МАКЕТЫ С ИНФОРМАЦИЕЙ
Общее число пораженных целей
2