|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_Робототехника и комплексная автоматизация\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_Системы автоматизированного проектирования\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Разработка сетевых компонентов и их интеграция в шаблон многопользовательской игры на Unreal Engine 4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_РК6-83Б\_\_\_\_\_\_ \_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Боженко Д.В. \_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Витюков Ф.А.\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2023 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой *\_\_РК6*\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ *А.П. Карпенко* \_

(И.О.Фамилия)

« 15 » февраля *2023* г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

Студент группы \_\_ РК6-83Б\_\_\_\_\_

\_\_ Боженко Дмитрий Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Тема квалификационной работы

Разработка сетевых компонентов и их интеграция в шаблон многопользовательской игры на Unreal Engine 4.

Источник тематики (НИР кафедры, заказ организаций и т.п.):

НИР кафедры

Тема квалификационной работы утверждена распоряжением по факультету \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_№\_\_\_\_\_\_\_\_\_\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

***Часть 1.*** Аналитическая часть.

*Необходимо провести изучение, анализ и подбор материала для выполнения проекта: исследовать программные интерфейсы EOS для внедрения в проект возможностей современных многопользовательских игр и их ограничения, изучить концепции сетевого программирования в Unreal Engine 4. Изучить с учетом ограничений, какие онлайн сервисы предоставляют возможность развертывания выделенных игровых серверов для запуска отдельных игровых сессий.*

***Часть 2.*** Практическая часть 1.

*Реализовать автоматическую систему записи очков в глобальную таблицу лидеров. При этом необходимо провести исследование проблемы хранения чисел с большой точностью для аддитивных систем и проблемы читаемости данных человеком. Отобразить глобальную таблицу лидеров на уровне приложения с учетом особенностей изученных проблем.*

***Часть 3.*** Практическая часть 2.

*Необходимо реализовать систему подбора игроков по одному из существующих алгоритмов на основе данных в созданной глобальной таблице лидеров. Внедрить в проект возможность создавать отдельные комнаты для ожидания остальных игроков перед началом игрового процесса.*

***Оформление квалификационной работы:***

Расчетно-пояснительная записка на \_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Дата выдачи задания «10» февраля 2023 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «11» июня 2023 г.

**Руководитель квалификационной работы** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ Витюков Ф.А. \_

(Подпись, дата) (И.О.Фамилия)

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ Боженко Д.В. \_\_

(Подпись, дата) (И.О.Фамилия)

Примечание:

1. Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**ФАКУЛЬТЕТ \_\_\_\_РК\_\_ \_\_**  УТВЕРЖДАЮ

**КАФЕДРА\_\_\_\_\_РК6\_\_\_\_\_\_** Заведующий кафедрой \_\_\_РК6\_\_\_\_

(Индекс)

**ГРУППА \_\_РК6-83Б\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Карпенко

« 15 » февраля 2023г.

**КАЛЕНДАРНЫЙ ПЛАН**

**выполнения выпускной квалификационной работы**

студента: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Боженко Дмитрия Владимировича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Тема квалификационной работы: «Разработка сетевых компонентов и их интеграция в шаблон многопользовательской игры на Unreal Engine 4»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | 1 часть. Теоретическая часть. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Заведующий кафедрой | Карпенко А.П. |
|  | 2 часть. Практическая часть 1. | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | 3 часть. Практическая часть 2. | *\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | 1-я редакция работы | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | Подготовка доклада и презентации | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Студент | Боженко Д.В. |
|  | Заключение руководителя | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Руководитель ВКР | Витюков Ф.А. |
|  | Допуск работы к защите на ГЭК (нормоконтроль) | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* | Нормоконтролер | Грошев С.В. |
|  | Внешняя рецензия | *\_\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* |  |  |
|  | Защита работы на ГЭК | *\_\_*  *Планируемая дата* | *\_\_\_*  *Фактическая дата* |  |  |

*Студент \_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись, дата) (подпись, дата)

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**НАПРАВЛЕНИЕ НА ЗАЩИТУ**

**Выпускной квалификационной работы**

**Председателю**

**Государственной Экзаменационной Комиссии № \_\_\_\_\_**

факультета «Робототехники и комплексной автоматизации» МГТУ им. Н.Э. Баумана

Направляется студент *Боженко Дмитрий Владимирович* группы *РК6-83Б* на защиту выпускной квалификационной работы «Разработка сетевых компонентов и их интеграция в шаблон многопользовательской игры на Unreal Engine 4»

Декан факультета *\_\_\_\_Шашурин Г.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Справка об успеваемости**

Студент *Боженко Дмитрий Владимирович* за время пребывания в МГТУ имени Н.Э. Баумана

с *2019* г. по *2023* г. полностью выполнил учебный план со следующими оценками:   
отлично – %, хорошо –  %, удовлетворительно –  %.

Инспектор деканата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отзыв руководителя выпускной квалификационной работы**

Место для ввода текста.

Руководитель **Витюков Ф.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

«18» июня 2023 г.

**Заключение кафедры о выпускной квалификационной работе**

Выпускная квалификационная работа просмотрена, и студент может быть допущен к защите этой работы в Государственной Экзаменационной Комиссии.

Зав. кафедрой \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

# АННОТАЦИЯ

Тип работы: выпускная квалификационная работа.

Тема работы: Разработка сетевых компонентов и их интеграция в шаблон многопользовательской игры на Unreal Engine 4.

Объект исследований: Процесс разработки и внедрения сетевых компонентов.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 6](#_Toc129821599)

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 8](#_Toc129821600)

[ВВЕДЕНИЕ 10](#_Toc129821601)

[1. Возможности современных многопользовательских игр 11](#_Toc129821602)

[2. СЕТЕВое программирование в unreal engine 13](#_Toc129821603)

[2.1. Структура многопользовательской игры 13](#_Toc129821604)

[2.2. Репликация 14](#_Toc129821605)

[2.3. Важнейшие классы в UE 4 для реализации многопользовательской игры 15](#_Toc129821606)

[2.4. RPC 17](#_Toc129821607)

[2.5. Сетевые роли 18](#_Toc129821608)

[2.6. Игровые сессии 19](#_Toc129821609)

[3. Таблицы лидеров 21](#_Toc129821610)

[3.1. Алгоритм начисления внутриигровых очков 21](#_Toc129821611)

[3.2. Проблема переполнения разрядной сетки при хранении чисел с плавающей запятой с большой точностью 21](#_Toc129821612)

[3.3. Проблема читаемости данных 21](#_Toc129821613)

[3.4. Отображение заработанных очков в интерфейсе игры 21](#_Toc129821614)

[4. Матчмейкинг 22](#_Toc129821615)

[5. Лобби 23](#_Toc129821616)

[6. внутриигровой голосовой чат 24](#_Toc129821617)

[ЗАКЛЮЧЕНИЕ 25](#_Toc129821618)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc129821619)

[ПРИЛОЖЕНИЕ А 27](#_Toc129821620)

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**ЯП —** язык программирования.

**Многопользовательская игра** — режим компьютерной игры, в котором играет более одного пользователя по сети Интернет.

**Движок** — это программный фреймворк, предназначенный в первую очередь для разработки видеоигр и обычно включающий соответствующие библиотеки и программы поддержки.

**UE 4** — движок Unreal Engine 4.

**Клиент** — машина, которая получает информацию об игровом мире через сервер и на которой происходит отрисовка игрового процесса.

**Сервер** — мощная вычислительная машина, через которую происходит обмен информацией об игровом мире без отрисовки графики и воспроизведения звуков.

**Удаленный игрок** — игрок, который находится на другой машине в пределах одной игровой сети.

**Локальный игрок** — игрок, который находится на локальной машине.

**LAN** — локальная вычислительная сеть, где все участники находятся, как правило, в пределах одной ограниченной территории.

**AActor** — один из основных классов в UE 4, являющийся базовым для всех остальных классов, представленных в игровом мире.

**Online Subsystem (далее OSS)** — кроссплатформенная система, позволяющая использовать современные возможности многопользовательских игр.

**Epic Online Services (далее EOS)** — OSS, предоставляемая компанией Epic Games.

**Epic Account Services (далее EAS)** — плагин, являющийся частью EOS, предоставляющий доступ к интерфейсу авторизации и многим другим интерфейсам.

**Epic Game Services** — плагин, являющийся частью EOS, предоставляющий доступ к интерфейсам, связанным с игровым процессом.

**Dev Portal** — Интернет-ресурс Epic Games, предназначенный для создания и редактирования настроек приложения, использующего EOS.

**УЗ** — Учетная запись.

# ВВЕДЕНИЕ

Рынок видеоигр стремительно развивается с каждым годом. На сегодняшний день рынок игр во всем мире является одним из самых больших сегментом мирового рынка цифрового контента, ежегодно генерируя многомиллиардные доходы и привлекая огромную аудиторию. Наибольшая доля в структуре российского рынка приходится на сегмент онлайн-игр. По данным *Mail.ru Group*, в 2019 году его объем увеличился на 9% и составил 56,7 млрд рублей (около $1 млрд).

Среди всех жанров игр на данный момент самыми популярными являются *MMO* (массовые многопользовательские онлайн игры), которые использует *Real-Time Multiplayer*. *Real-Time Multiplayer* — это тот режим игры, в котором каждый пользователь получает и отправляет данные об игровом мире с выделенного игрового сервера несколько десятков раз за отведенный промежуток времени. Данное понятие называется тикрейт, т. е. какое количество запросов сервер может обрабатывать за установленные промежуток времени.

Целью данной практической работой является изучение одной из доступных подсистем для движка Unreal Engine 4, а именно предоставляемых ей программных интерфейсов, таких как матчмейкинг, голосовой чат, таблицы лидеров и т. п., которые широко применяются в современных многопользовательских играх. На основе полученных знаний разработать вышеперечисленные сетевые компоненты и внедрить их в шаблон многопользовательской игры.

Данная цель является актуальной, так как изучение концепции создания многопользовательских игр является необходимым условием освоения рынка видеоигр, которые в свою очередь стремительно развиваются и набирают большую популярность в сфере информационных технологий.

# Возможности современных многопользовательских игр

Современные многопользовательские игры включают в себя множество возможностей, которые они могут предоставить пользователям. Самыми распространенными из них являются авторизация, система достижений, матчмейкинг (система подбора игроков), создание лобби, таблицы лидеров, система голосового чата и античит-система.

В UE 4 существует несколько OSS, которые предоставляют доступ к возможностям современных многопользовательских игр, а именно Online Subsystem EOS, Online Subsystem Steam, Online Subsystem Oculus, Online Subsystem Google Play, а также Online Subsystem Null. Каждая из перечисленных подсистем предоставляет возможность использовать возможности современных многопользовательских игр, добавляя собственные интеграции.

Online Subsystem EOS является хорошим выбором, так как предоставляет широкие возможности выбора интерфейсов для реализации возможностей многопользовательских игр, которые могут расширить функционал любого шаблона многопользовательской игры. Также EOS имеет подробную документацию, которую необходимо использовать при интеграции предоставленных интерфейсов в проект.

EOS подразделяется на два вида сервисов: EAS и EOS. Оба сервиса предоставляют большое количество интерфейсов, которые добавляют в игру возможности современных многопользовательских игр. Оба плагина могут использоваться как одновременно, так каждый по отдельности в независимости друг от друга.

EAS в основном предоставляет интерфейсы для авторизации пользователей с помощью УЗ Epic Games и управления списком друзей. Epic Games Services предоставляют интерфейсы, которые связаны с управлением многопользовательского игрового процесса пользователей.

Для того, чтобы в приложении можно было использовать данные сервисы, необходимо провести предварительную настройку приложения на Dev Portal Epic Games, где нужно получить необходимые данные для инициализации приложения и разрешить использование двух вышеперечисленных сервисов.

# СЕТЕВое программирование в unreal engine

Соединение в многопользовательских играх происходит по клиент-серверной модели, когда несколько клиентов подключаются к выделенному серверу и через него передают друг другу информацию об игровом мире.

На этапе разработки многопользовательской игры, важно понимать, как и с помощью каких инструментов игрового движка реализовано сетевое взаимодействие между игроками, какие при этом создаются объекты классов, сколько копий каждого класса создается и на какой машине они находятся.

## Структура многопользовательской игры

В UE 4 существует четыре основных мода многопользовательской игры:

Standalone — автономная игра, где сам экземпляр игры является сервером. Игра, запущенная в таком режиме, не принимает никаких подключений от удаленных игроков.

Client — режим игры, в котором она имеет роль клиента, и работает только при подключении к игровому серверу.

Listen-Server — режим игры, в котором сервером становится один из клиентов и на котором размещена сетевая многопользовательская сессия. В таком режиме игра как принимает запросы от удаленных игроков, так и содержит своих локальных игроков. Такой режим многопользовательской игры хорошо подходит для развертывания кооперативных игр, где все игроки находятся в пределах LAN и сетевые взаимодействия осуществляются по P2P-сети.

Dedicated-Server — режим игры, в котором сервер расположен на отдельно выделенной машине. Экземпляр игры, запущенный на выделенном сервере, принимает запросы от удаленных игроков, но сам не содержит никаких локальных игроков. Следовательно, на таком экземпляре игры отсутствуют такие функции, ориентированные на игроков, как отрисовка графики, вывод звуков и пользовательский ввод. Данное решение является основным для большинства многопользовательских игр, где есть много игроков, так как выделенные сервера обладают большой вычислительной мощностью и обеспечивают безопасность от обмана.

## Репликация

Репликация — это синхронизация информации об игровом мире между сторонами. Другими словами, репликация — это механизм, который создает множество копий одного и того же объекта. Объект и его копии хранятся на разных машинах (на клиенте и на сервере), за счет чего между игроками, находящимися на разных игровых машинах, происходит синхронизация информации об одном и том же объекте. Реплицировать можно переменные, события и объекты. Передача измеренной информации об объекте может осуществляться от сервера к одному клиенту, от любого клиента к серверу, и от сервера ко всем клиентам.

В UE 4 для реализации репликации объектов используется возможность класса AActor, так как все объекты, представленные в игровом мире, являются сущностями этого класса.

Важно понимать, что игрок видит перемещение другого игрока у себя на локальной машине именно за счет репликации. Это происходит за счет того, что при репликации объекта, его копии хранятся на локальных машинах других игроков, а также на сервере. При перемещении игрок посылает серверу свои координаты, тот, в свою очередь, анализируя эти координаты, передает их копиям этого игрока, которые расположены на остальных клиентах. Следовательно, когда игрок видит перемещение другого игрока у себя на локальной машине, он видит перемещение копии этого игрока, которая полностью управляется сервером.

Реплицированние переменных в UE 4 также осуществляется особым образом. Переменную, которую надо реплицировать помечается с помощью макроса UPROPERTY(Replicated). Важно понимать сам механизм реплицирования. Когда переменная, находящаяся на сервере, изменяет свое значение, то ее значение также будет изменено на всех клиентах. При этом также возможно указать функцию, которая будет выполняться при изменении реплицированной переменной через флаг ReplicatedUsing. Важно заметить, что данная функция будет выполняться только при изменении реплицированной переменной и только на клиентах, и никогда не будет выполняться на сервере.

## Важнейшие классы в UE 4 для реализации многопользовательской игры

Для базовой реализации простой многопользовательской игры в UE 4 существует несколько базовых классов:

АGameMode — самый важный класс, который отвечает за правила игры. Важно знать, что экземпляр такого класса находится только на сервере. Чтобы избежать нечестной игры, все действия, связанные с игровой логикой, запрашиваются через сущность этого класса. Важно понимать, что попытка получить доступ к сущности класса AGameMode с клиента будет безуспешной.

AGameState — класс, который содержит в себе информацию о текущем состоянии игры, например, о количестве подключенных к сессии игроков. Сущность данного класса располагается на сервере, а также его копия располагается на каждом из клиентов. Таким образом, сущность данного класса самая важная, которая необходима для передачи общей информации между сервером и клиентами.

APlayerState — класс, который содержит в себе всю текущую информацию об игроке, подключенном к игровой сессии. Сущность данного класса находится на каждом клиенте, а также копия сущности класса APlayerState каждого клиента находится на сервере. Следовательно, сервер знает о сущности APlayerState каждого клиента, а клиент знает о существовании только собственного класса APlayerState.

АPlayerController — класс, который остается за игроком на протяжении всей игровой сессии. С помощью APlayerController клиента можно легко управлять интерфейсом игрока, когда необходимо освежить новую информацию о состоянии объектов игрового мира, например, изменение очков, изменение здоровья и другое. Распределение по клиентам и серверу такое же, как у APlayerState.

AHUD — класс, который существуют только на клиенте, который является владельцем данного класса. AHUD используется для управления виджетами клиентами, управлением данным классом можно осуществлять с клиента, либо же с сервера с помощью RPC.

APawn — класс, который представляет из себя объект на сцене, которым управляет игрок либо же сервер. Данный класс является производным от класса AActor, поэтому возможно его реплицирование. Каждый клиент и сервер знает о существовании о каждом объекте класса APawn.

Ниже на рисунке 1 представлена схема репликации классов.

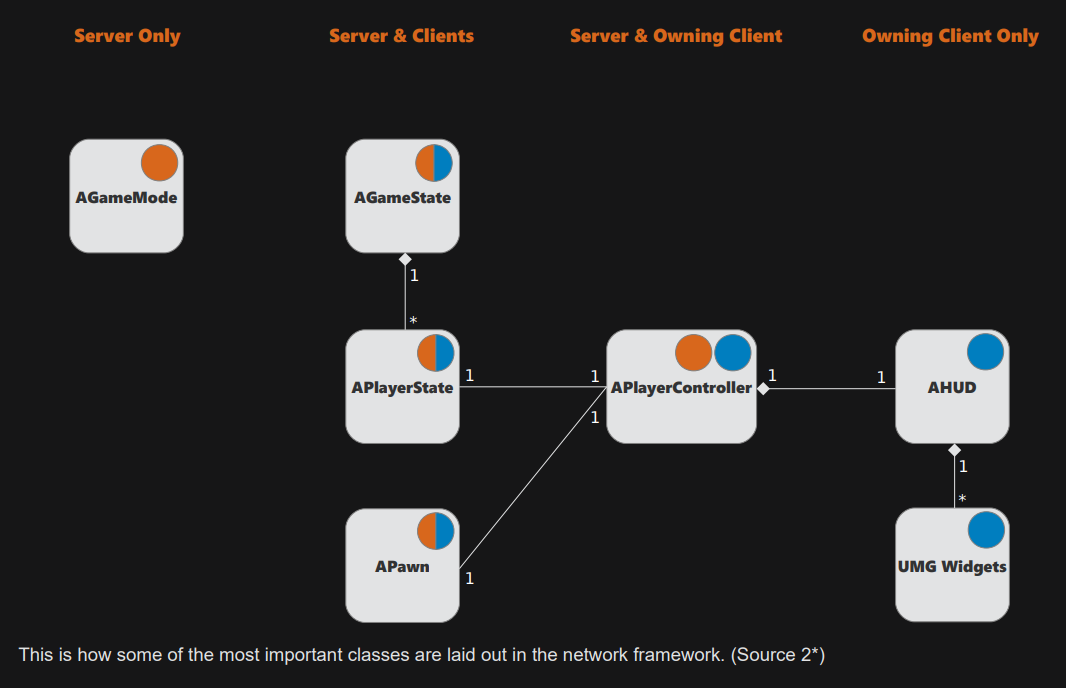


Рисунок — Схема репликации основных классов UE

## RPC

RPC — это особые функции в UE 4, которые вызываются на одной машине, а выполняются на другой. RPC помечаются через макрос UFUNCTION(). Всего существует три вида RPC:

Client — данный тип RPC, как правило, вызывается с сервера. Модификатор Client говорит о том, что данная функция будет выполнена только на том клиенте, который владеет данной RPC-функцией. Данный вид RPC хорошо подходит для обновления виджета в HUD клиента, когда на сервере произошло изменение реплицированной переменной.

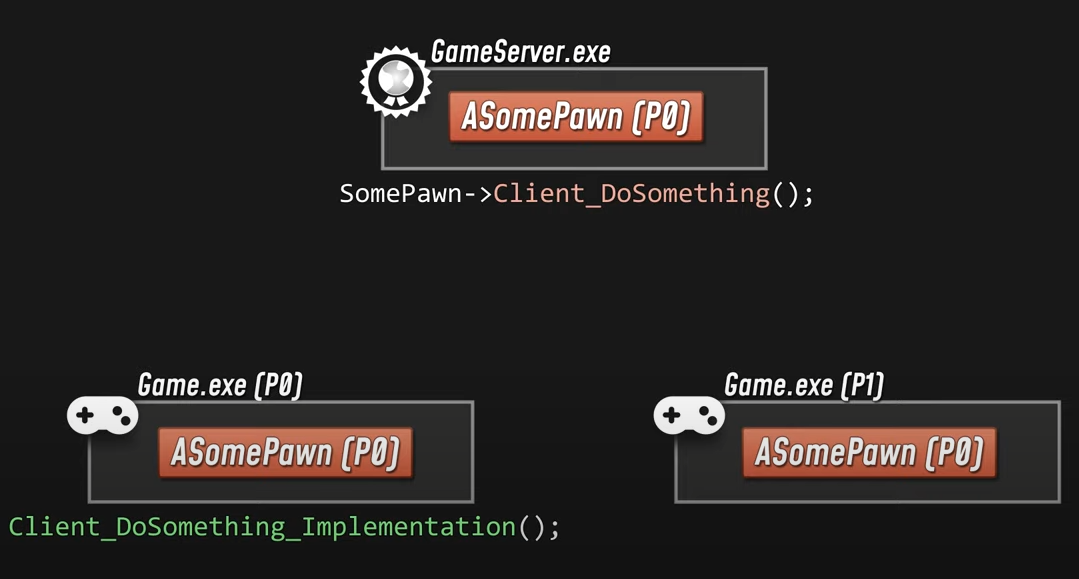


Рисунок — Схема вызова и выполнения Client RPC

Server — данный тип RPC вызывается с клиента, а выполнение функции производится только на сервере.

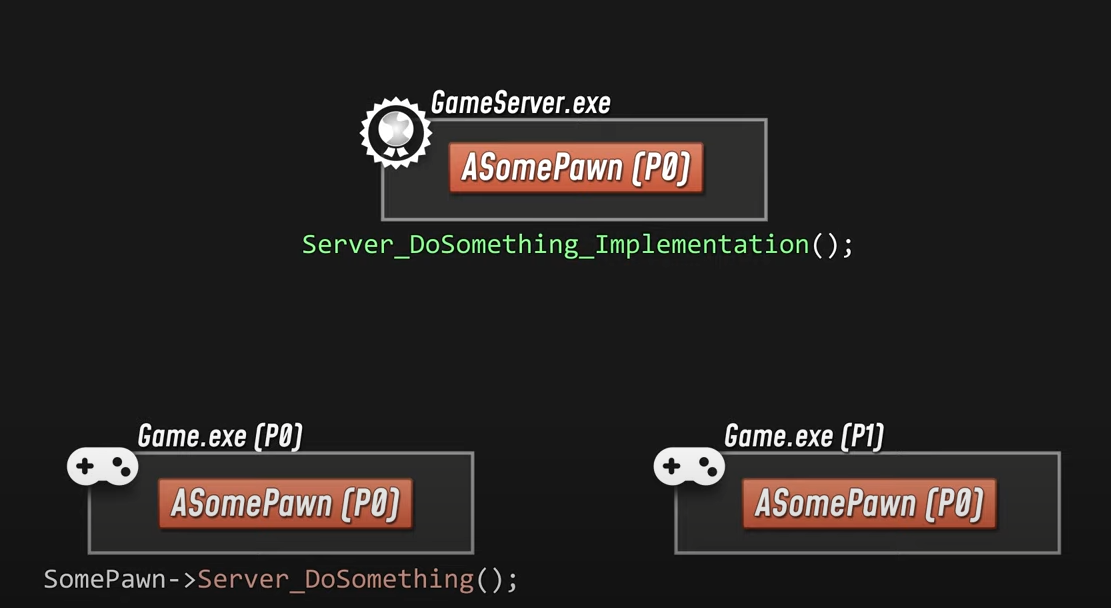


Рисунок — Схема вызова и выполнения Server RPC

NetMulticast — данный тип RPC всегда вызывается с сервера и будет выполнен как на всех клиентах, так и на сервере. Такой тип RPC хорошо подходит, например, для проигрывания анимации после смерти игрока.

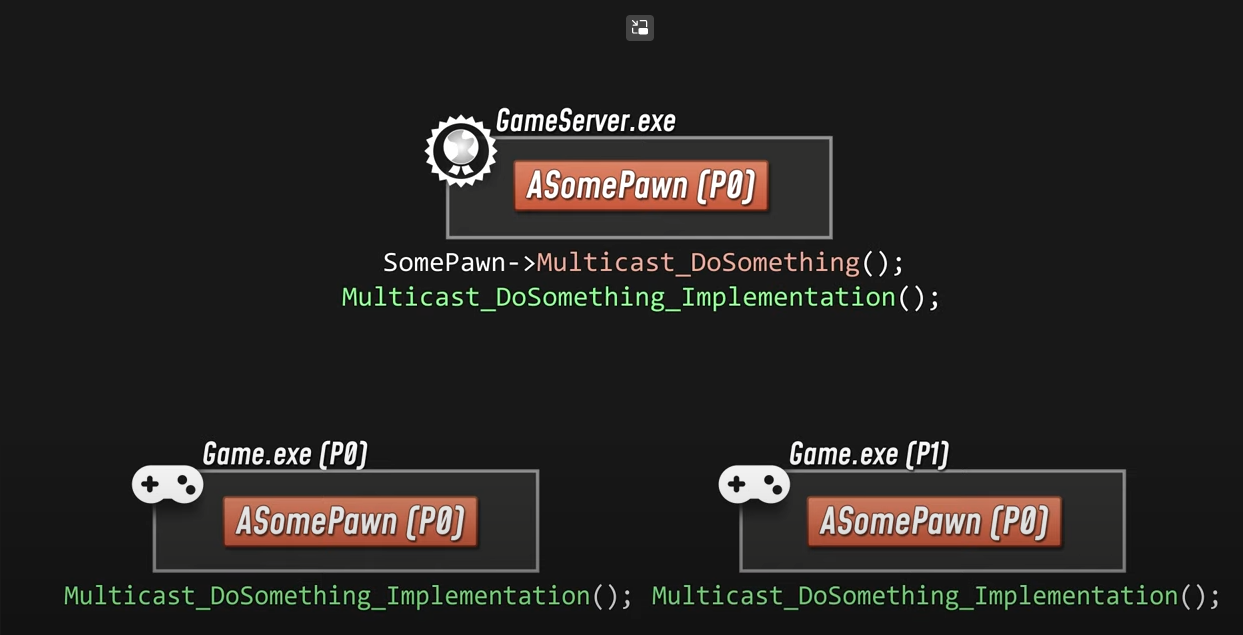


Рисунок — Схема вызова и выполнения NetMulicast RPC

## Сетевые роли

В UE 4 существует 4 основных сетевых роли, которые может иметь AActor:

ROLE\_AutonomousProxy — сетевая роль, которая показывает, что данный AActor находится на клиенте и управляется живым игроком.

ROLE\_SimulatedProxy — сетевая роль, которая показывает, что данный AActor находится на клиенте и управляется сервером.

ROLE\_Athority — сетевая роль, которая определяет, что данный AActor или его копия находится на сервере.

ROLE\_None — сетевая роль, которая дается AActor в случае, если он не обладает ни одной ролью из вышеперечисленных.

На рисунке 5 представлена таблица, с помощью которой можно понять, какую роль получает AActor в зависимости от этого, на какой машине находится он или его копия.

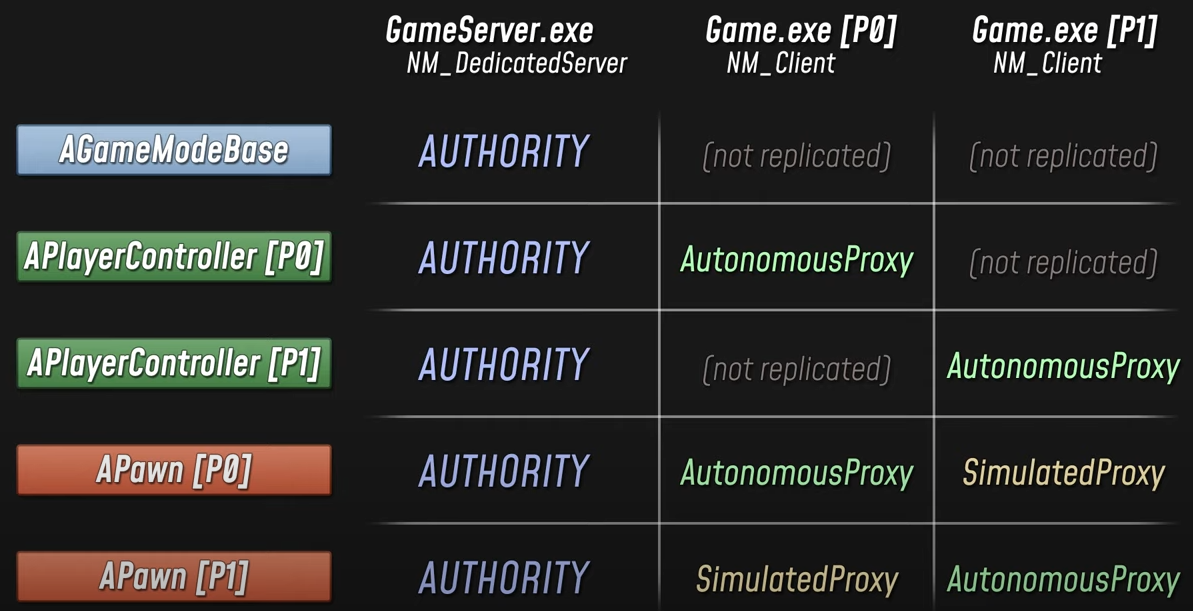


Рисунок — Схема распределения сетевых ролей на примере двух игроков

## Игровые сессии

Сессии, также, как и сетевое программирование, является базовым аспектом, с помощью которого строятся многопользовательские игры. С помощью сетевого программирования прорабатывается процесс взаимодействия персонажей друг с другом и с игровым миром в целом. Сессии служат для того, чтобы контролировать подключение игроков к выделенному серверу или хосту.

Сессия — это большая структура данных, которая содержит в себе множество полей, настроек и методов, с помощью которых можно задавать правила подключения игроков к серверу. Базовыми настройками игровой сессии является:

Максимальное количество игроков, которые могут одновременно подключиться к игре.

1. Доступность игры из сети Интернет.
2. Возможность использовать лобби.
3. Возможность подключаться к сессии после начала игрового процесса.

Сущность сессии всегда находится на сервере. Клиенты, которые хотят подключиться к северу, сперва должны подключиться к игровой сессии. Если подключение клиента успешно прошло проверку, ему разрешается присоединиться к игровому серверу. Таким образом роль сессии заключается в том, что она является промежуточным узлом подключения между клиентом и сервером, который проверяет подключение каждого клиента и управляет им.

# Таблицы лидеров

## Алгоритм начисления внутриигровых очков

## Проблема переполнения разрядной сетки при хранении чисел с плавающей запятой с большой точностью

## Проблема читаемости данных

## Отображение заработанных очков в интерфейсе игры

# Матчмейкинг

# Лобби

# внутриигровой голосовой чат

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было создано развертывание многопользовательского приложения, построенного на игровом движке Unreal Engine 4. Полученный с помощью Kubernetes и Docker кластер был проанализирован. В результате тестирования были сделаны выводы о нагрузке, которую выдерживает развертывание. На каждые 256 игроков необходимо 1 vCPU и 4 Гб оперативной памяти.

Данная работа демонстрирует полный цикл развертывания многопользовательского приложения. Процесс развертывания может меняться и дополнятся, в зависимости от требований к целевому результату, программному обеспечению или иных обстоятельств.

Создание инфраструктур для многопользовательских приложений – актуальная задача, так как позволяет увеличить качество предоставляемых услуг, снизить расходы предприятия (например, при использовании инфраструктуры в облаке).

Решения в области развертываний не ограничиваются игровой индустрией, они активно используются повсеместно – в развлекательных сервисах (трансляции, видео), соцсетях, различных инженерных приложениях и прочих, где необходимо обеспечить масштабируемость и отказоустойчивость.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Services Documentation. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs> (дата обращения 25.11.2022).
2. Auth Interface. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/epic-account-services/auth-interface> (дата обращения 26.11.2022).
3. Platform Interface. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/game-services/eos-platform-interface> (дата обращения 26.11.2022).
4. EOS Game Services. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/game-services> **(**дата обращения 26.11.2022).
5. EOS Account Services. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/epic-account-services>. (дата обращения 27.11.2022);
6. EOS SDK Errors Code. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/epic-online-services/sdk-error-codes>. (дата обращения 01.12.2022);
7. Уильям Шериф. Unreal Engine 4.x Scripting with C++ Cookbook / Уильям Шериф, Стивен Уиттл, Джон Доран. — Packt Publishing, 2019 г. — 708 c.
8. Арам Куксон. Unreal Engine 4 Game Development in 24 Hours, Sams Teach Yourself / Арам Куксон, Райан Даулингсока, Клинтон Крамплер. — Москва: Бомбора, 2019 г. — 528 с.
9. Маркус Ромеру. Blueprints Visual Scripting for Unreal Engine 2nd Edition / Маркус Ромеру, Бренден Сьюэлл. — Packt Publishing, 2019 г. — 380 с.

# ПРИЛОЖЕНИЕ А

Список графических материалов: