



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 (РК-6)

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Боженко Дмитрий Владимирович

(фамилия, имя, отчество)

Группа РК6-83Б

Тип практики Преддипломная

Название предприятия Общество с ограниченной ответственностью
АВ Софт

Студент РК6-83Б
(Группа)

Д.В. Боженко
(подпись, дата)

Руководитель практики
от кафедры

Ф.А. Витюков
(подпись, дата)

Оценка _____

Москва, 2023 г

УТВЕРЖДАЮ

Заведующий кафедрой РК-6
(Индекс)

_____ А.П. Карпенко
(И.О.Фамилия)

« _____ » _____ 2023 г.

З А Д А Н И Е

на прохождение технологической практики (производственная)

Студент Боженко Дмитрий Владимирович, 4 курса, группы РК6-83Б
(фамилия, имя, отчество, № курса, индекс группы)

в период с «15» мая 2023 г. по «28» мая 2023 г.

Предприятие: Общество с ограниченной ответственностью АВ Софт

Подразделение: Место для ввода текста.

(отдел/сектор/цех)

Руководитель практики от предприятия (наставник):

Чухнов Антон Павлович

(фамилия имя отчество полностью, должность)

Руководитель практики от кафедры:

Витюков Федор Андреевич

(фамилия имя отчество полностью, должность)

Задание.

1. Изучить проблему читаемости пользователем чисел с плавающей запятой.
2. Учитывая изученную проблему реализовать отображение на экране каждого пользователя заработанных им в конце матча очков.
3. Реализовать глобальную таблицу лидеров на уровне приложения.
4. Реализовать в разработанной таблице лидеров разделение игроков на лиги.

Дата выдачи задания «15» мая 2023 г.

Руководитель практики от предприятия

А.П. Чухнов

(подпись, дата)

Руководитель практики от кафедры

Ф.А. Витюков

(подпись, дата)

Студент РК6-83Б

(группа)

Д.В. Боженко

(подпись, дата)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1. Алгоритм начисления очков для глобальной таблицы лидеров	5
2. Проблема читаемости данных.....	8
3. Отображение заработанных очков в интерфейсе игры	9
ЗАКЛЮЧЕНИЕ	18
СПИСОК ЛИТЕРАТУРЫ.....	19

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 *ММО* (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 предоставляемого ей программного интерфейса, такого как матчмейкинг и таблицы лидеров, 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.

1. АЛГОРИТМ НАЧИСЛЕНИЯ ОЧКОВ ДЛЯ ГЛОБАЛЬНОЙ ТАБЛИЦЫ ЛИДЕРОВ

Для преобразования внутриигровых очков каждого игрока в очки, которые будут представлены в глобальной таблице лидеров, необходимо применить определенный алгоритм. Для многопользовательских игр, где игра происходит в режиме “каждый сам за себя” и “команда на команду”, необходимо анализировать таблицу игроков с внутриигровыми очками и в соответствии с заработанными очками определить, какое значение отправить в глобальную таблицу лидеров. Для реализации вышеописанного алгоритма, можно использовать нормализацию внутриигровых очков и дальнейшую нормировку данных, которые будут отправлены в глобальную таблицу лидеров. Нормализация — это один из способов предобработки информации, при котором входные данные приводятся к заданному диапазону, например, к диапазону $[0;1]$ или $[-1;1]$.

Для получения нормализующей функции, которая работает с нормализованными внутриигровыми очками, нужно определить дискретный набор данных (Рисунок 1).

-1	-1
-0,9	-0,72
-0,8	-0,6
-0,7	-0,48
-0,6	-0,4
-0,5	-0,32
-0,4	-0,24
-0,3	-0,16
-0,2	-0,08
-0,1	-0,04
0	0
0,1	0,04
0,2	0,08
0,3	0,16
0,4	0,24
0,5	0,32
0,6	0,4
0,7	0,48
0,8	0,6
0,9	0,72
1	1

Рисунок 1 — Дискретный набор данных для кривой начисления очков

Далее по представленному набору данных с помощью интерполяции можно получить аналитическое выражение необходимой функции. Полученное аналитическое выражение функции имеет вид

$y = 0.4849x^3 - 1e^{-14}x^2 + 0.4674x + 3e^{-14}$. График данной функции представлен ниже (Рисунок 2).

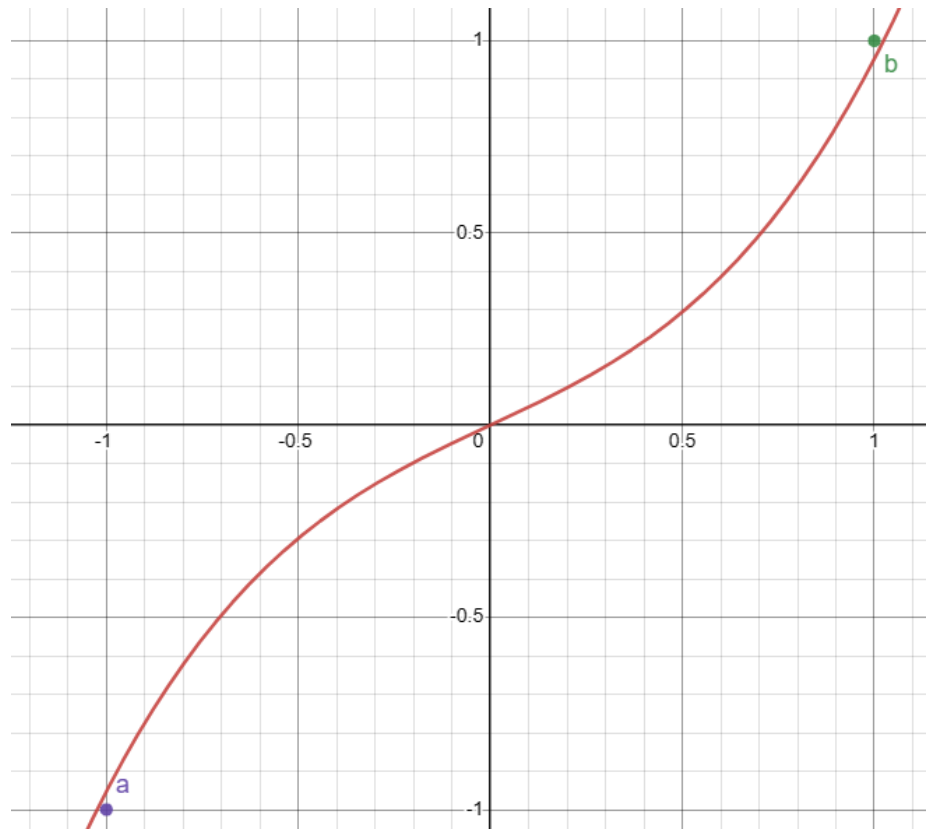


Рисунок 2 — График преобразующей линии тренда $[-1;1]$

Как видно на графике, при $x = 1$ и при $x = -1$ значение функции не равняется граничным значениям 1 и -1 соответственно. Это произошло вследствие того, что полученное аналитическое выражение полинома является аппроксимацией. Для этого необходимо найти значения аргумента, при которых функция принимает необходимые граничные значения и скорректировать интервал принимаемых значений. Скорректированный интервал будет иметь вид $[-1.03;1.03]$.

Так как функция должна принимать на вход параметр, принадлежащий интервалу $[-1.03;1.03]$, необходимо привести внутриигровые очки каждого игрока к данному интервалу. Для этого необходимо воспользоваться формулой

общего вида, которая приводит данные к произвольному диапазону $[a;b]$ и выражается следующим образом:

$$\hat{x} = a + \frac{(x - x_{\min})(b - a)}{x_{\max} - x_{\min}}$$

где a — левая граница требуемого диапазона, b — правая граница требуемого диапазона, x_{\min} — минимальное число внутриигровых очков, которое заработали все игроки, x_{\max} — максимальное число внутриигровых очков, которое заработали все игроки, x — число внутриигровых очков, которое заработал текущий игрок.

Тогда алгоритм начисления очков заключается в том, что игрок, набравший наибольшее количество внутриигровых очков, получает максимальное нормализованное количество очков 1.00, игрок, набравший минимальное количество внутриигровых очков, получает минимальное нормализованное количество очков -1.00. Игрок, занявший среднюю позицию во внутриигровой таблице, получает 0 внутриигровых очков за заверченный раунд.

2. ПРОБЛЕМА ЧИТАЕМОСТИ ДАННЫХ

Использование нормализованных данных, где точность составляет до 7 знаков после запятой достаточно удобно для хранения, так как маловероятно, что число переполнит разрядную сетку. С другой стороны, возникает проблема, которая заключается в том, что у пользователя будут трудности с чтением и анализом отображаемых глобальных очков. Для решения данной проблемы необходимо выполнить нормировку нормализованных значений и получить результаты в определенных понятных единицах.

Нормировка — это корректировка нормализованных значений в соответствии с выбранным алгоритмом с целью сделать их более читаемыми.

Чтобы нормировать подсчитанное количество внутриигровых очков каждого игрока, было принято решение умножать высчитанные нормализованные очки на коэффициент = 25 и округлять полученные результат до ближайшего целого числа. Округление результата необходимо, так как в EOS очки в глобальных таблицах лидеров хранятся только в целочисленных значениях. Округление до ближайшего целого числа осуществляется с помощью метод `FloorToInt` класса `FMath`: `FMath::FloodToInt(float)`.

В итоге после нормировки пользователь получает данные в более понятных для него единицах. Игрок, получивший максимальное количество очков, получает +25 очков за матч, минимальное количество очков — -25. Данные числа кратны 5, их удобно считать и анализировать, однако теперь они будут занимать больше места при хранении в EOS.

3. ОТОБРАЖЕНИЕ ЗАРАБОТАННЫХ ОЧКОВ В ИНТЕРФЕЙСЕ ИГРЫ

Для того чтобы начать использовать таблицы лидеров в своем приложении, необходимо зарегистрировать в системе EOS на Developer Portal новую таблицу лидеров и статистику, которую эта новая таблица будет отслеживать.

В качестве статистики разработчик имеет возможность отслеживать различные игровые данные игроков: количество собранных предметов, время прохождения определенного уровня, общее количество поражений и побед или просто общее количество раз совершения игроком какого-либо определенного действия. Ниже представлен способ создания новой отслеживаемой статистики на Developer Portal (Рисунок 3).

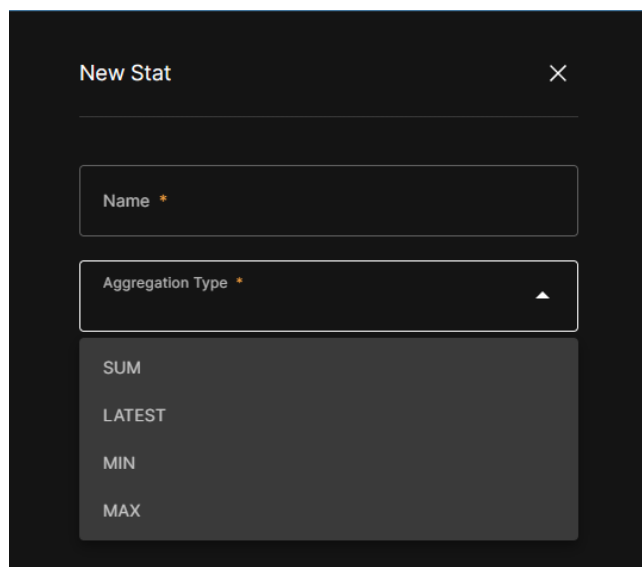


Рисунок 3 — Создание отслеживаемой игровой статистики на Developer Portal

Как видно, для создания статистики необходимо задать ее имя и тип. Всего в системе EOS существует 4 типа статистики:

1. Тип SUM — позволяет суммировать каждый результат, который отправляет игрок. Игрок, набравший наибольшее количество очков, занимает первое место.

2. Тип MIN — позволяет зарегистрировать наименьший результат игрока. Игрок, заработавший наименьшее количество очков, занимает первое место.
3. Тип MAX — позволяет зарегистрировать наибольший результат игрока. Игрок, набравший наибольшее количество очков, занимает первое место.
4. Тип LATEST — позволяет зарегистрировать самый последний по дате результат игрока. Игрок, набравший наибольшее количество очков, занимает первую позицию.

Для создания глобальной таблицы лидеров отлично подходит тип SUM, так как необходимо для игрока суммировать его заработанные очки в конце каждого матча и сразу отсортировать глобальную таблицу лидеров по данному значению.

Для создания глобальной таблицы лидеров также необходимо задать ее имя, статистику, которая она будет отслеживать, и время жизни таблицы (для удобства можно поставить бессрочный период жизни). Создание глобальной таблицы лидеров представлено ниже (Рисунок 4).

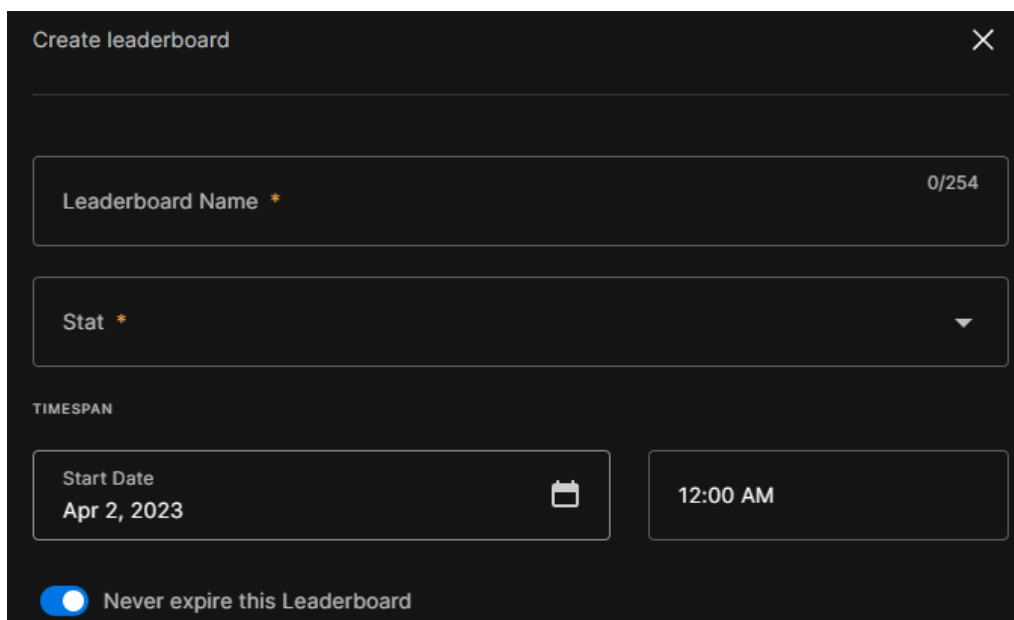
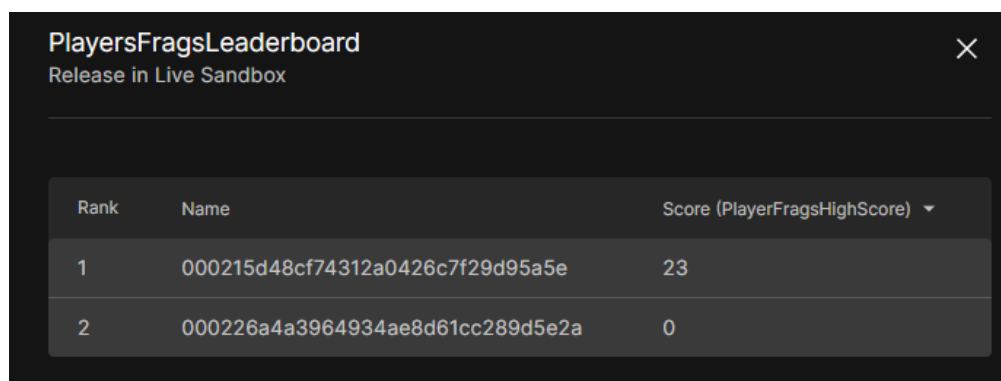


Рисунок 4 — Создание глобальной таблицы лидеров на Developer Portal

На Developer Portal можно получить доступ к созданной таблице лидеров и увидеть все записи игроков, данные которых записывались в созданную

глобальную таблицу. Данная таблица доступна только владельцу приложения и является необходимой лишь во время разработки (Рисунок 5).



PlayersFragLeaderboard		
Release in Live Sandbox		
Rank	Name	Score (PlayerFragHighScore) ▾
1	000215d48cf74312a0426c7f29d95a5e	23
2	000226a4a3964934ae8d61cc289d5e2a	0

Рисунок 5 — Глобальная таблица лидеров на Developer Portal

Необходимо разработать отдельную таблицу с делениями на лиги на уровне приложения, чтобы каждый пользователь имел доступ к данной таблице и имел возможность просматривать ее содержимое.

Для реализации данной задачи были созданы два виджета средствами UMG: виджет глобальной таблицы игроков и виджет записи игрока, которая добавляется в таблицу друг за другом.

Таблица представляет собой виджет, состоящий из кнопок для управления лигами и их контейнера, в который можно добавлять бесконечно много записей и прокручивать с помощью колеса мыши (Рисунок 6).

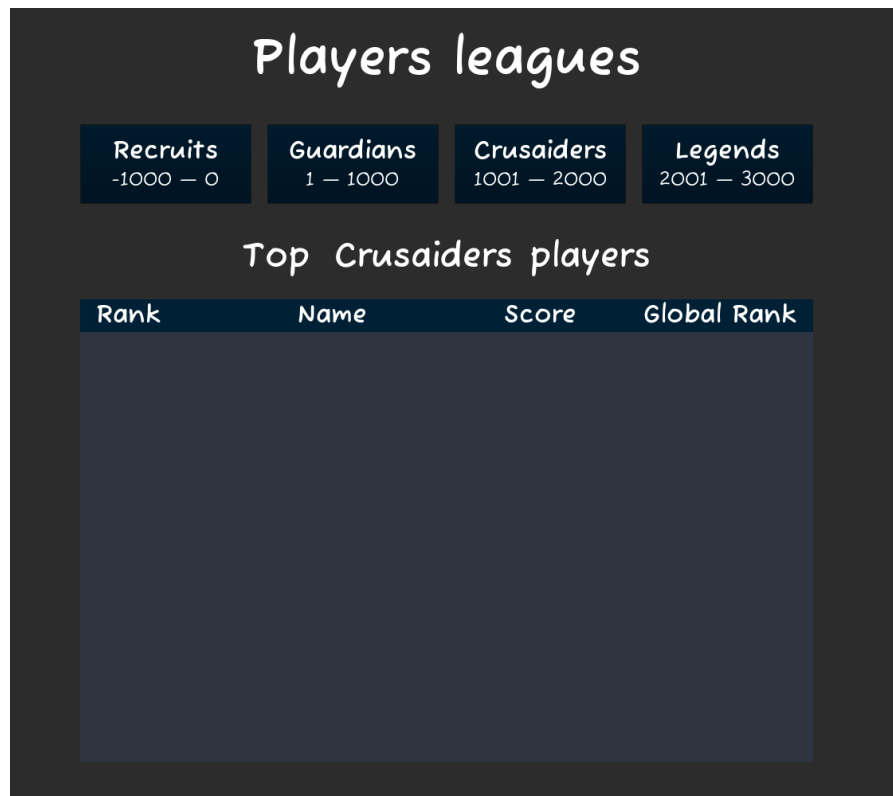


Рисунок 6 — Виджет глобальной таблицы лидеров



Рисунок 7 — Глобальная таблица лидеров на уровне приложения

Чтобы заполнять таблицу лидеров, необходимо с помощью языка C++ и интерфейса, предоставляющего возможность управлять глобальными таблицами лидеров в EOS, выполнить запрос в EOS, обработать полученные данные об игроках из таблицы и добавить записи в глобальную таблицу в виде виджетов, где указан ранг игрока в лиге, его имя в системе, его глобальные очки и глобальный ранг.

Запрос данных в EOS и последующая обработка выполненного результата представлена ниже (Листинг 1).

Листинг 1 — Получение информации об игроках из глобальной таблицы лидеров

```
void ULab4GameInstance::QueryGlobalRanks(const int32 LeftBoundry,
const int32 RightBoundry)
{
    LeftScoreBoundary = LeftBoundry;
    RightScoreBoundary = RightBoundry;

    LeaderboardsPtr = LeaderboardsPtr == nullptr ? OnlineSubsystem-
>GetLeaderboardsInterface() : LeaderboardsPtr;

    if (!LeaderboardsPtr.IsValid())
    {
        return;
    }

    FOnlineLeaderboardReadRef ReadRef =
    MakeShared<FOnlineLeaderboardRead, ESPMode::ThreadSafe>();
    ReadRef->LeaderboardName = RankedLeaderboardName;
    ReadRef-
>ColumnMetadata.Add(FColumnMetadata(FName(TEXT("PlayerFragHighSco
re")), EOnlineKeyValuePairDataType::Int32));
    ReadRef->SortedColumn = FName(TEXT("Score"));
    QueryGlobalRanksDelegateHandle = LeaderboardsPtr-
>AddOnLeaderboardReadCompleteDelegate_Handle(
        FOnLeaderboardReadComplete::FDelegate::CreateUObject(
            this,
            &ULab4GameInstance::HandleQueryGlobalRanksResult,
            ReadRef
        )
    );

    if (!LeaderboardsPtr->ReadLeaderboardsAroundRank(
        0,
        100,
        ReadRef
    ))
    {
        LeaderboardsPtr-
>ClearOnLeaderboardReadCompleteDelegate_Handle(QueryGlobalRanksDel
egateHandle);
        QueryGlobalRanksDelegateHandle.Reset();
    }
}
```

Проанализировав Листинг 1, можно увидеть, что запрос в глобальную таблицу лидеров выполняется с помощью программного интерфейса `LeaderboardsInterface`. Далее в методе создается переменная `FOnlineLeaderboardReadRef ReadRef` ссылочного типа, которая будет содержать в себе запрашиваемые данные о таблице лидеров. Чтобы сделать запрос с

помощью метода `OnlineLeaderboardInterface::ReadLeaderboardsAroundRank` необходимо указать имя глобальной таблицы, имя статистики, которую отслеживает данная таблица, а также имя поля, в которое будет помещен результат запроса и по которому будет производиться сортировка.

Для обработки запроса данных глобальной таблицы лидеров создается коллбэк `ULab4GameInstance::HandleQueryGlobalRanksResult(const bool bWasSuccessful, FOnlineLeaderboardReadRef LeaderboardReadRef)`. В нем происходит фильтрация пользователей на лиги, в соответствии с их очками в глобальной таблице. Также для каждого пользователя создается виджет, который далее помещается в виджет глобальной таблицы на уровне приложения.

В результате каждый пользователь имеет возможность на уровне приложения просматривать глобальную таблицу лидеров с разделением игроков на лиги (Рисунок 7).

Для того, чтобы таблица лидеров заполнялась автоматически после каждого завершеного матча, необходимо в соответствии с вышеописанным алгоритмом начисления глобальных очков вычислить, какое количество очков заработал каждый игрок, выполнить их нормировку для лучшей читаемости и затем после нормировки отправить результат в EOS и отразить результат подсчетов в HUD игрока.

Для отображения результатов конца матча, где указан выигравший игрок и заработанное количество очков, также необходимо создать виджет, который будет автоматически появляться по завершении игры на экране игрока. Пример такого виджета представлен ниже (Рисунок 8).

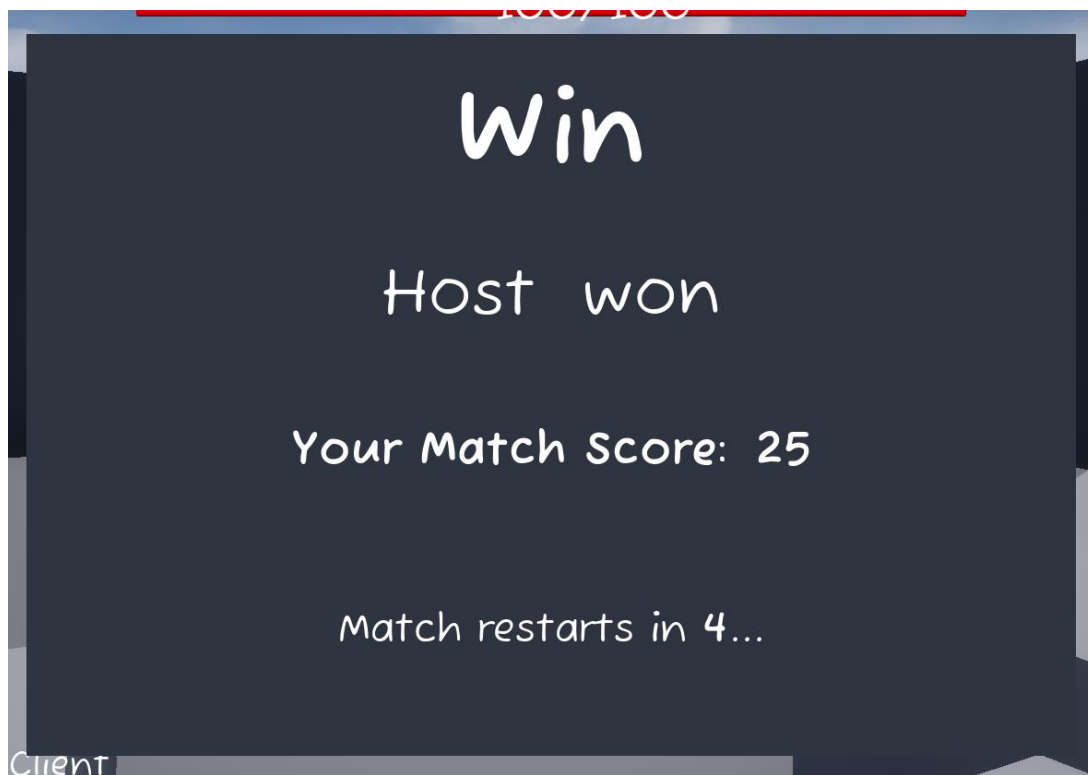


Рисунок 8 — Виджет выигравшего игрока с информацией о завершившемся матче

Управление таким виджетом также осуществляется с помощью C++. Сперва на сервере необходимо подсчитать, какое количество очков заработал каждый игрок, затем с помощью RPC, которое будет вызываться с сервера на клиент, необходимо показать данный виджет. В данном случае использование RPC типа Client оправдано, так как управление всеми виджетами во время игрового процесса осуществляется с помощью классов ANUD и APlayerController, которые доступны только клиенту, который владеет экземплярами данных классов. Ниже представлены два метода, с помощью которых выполняется управление виджетом (Листинг 2).

Листинг 2 — Методы управления виджетом с результатами матча

```
void
ALab4PlayerController::ClientGameOverToggle_Implementation(ALab4Pl
ayerState* WinnerPlayerState, float NormalizedPlayerScore)
{
    Lab4HUD = Lab4HUD == nullptr ? Cast<ALab4HUD>(GetHUD()) :
    Lab4HUD;

    if (Lab4HUD && WinnerPlayerState)
    {
        UE_LOG(LogTemp, Error, TEXT("Show UI winner"))
        Lab4HUD->ShowGameOverWidget(
            WinnerPlayerState, NormalizedPlayerScore);
        SetRestartCountdownTimer();
    }
}

void ALab4HUD::ShowGameOverWidget(const ALab4PlayerState*
WinnerPlayerState, float NormalizedPlayerScore)
{
    APlayerController* PlayerController =
    GetOwningPlayerController();

    if (PlayerController && GameOverWidgetClass)
    {
        GameOverWidget = CreateWidget<UMyUserWidget>
        (PlayerController, GameOverWidgetClass);
    }

    if (GameOverWidget && WinnerPlayerState)
    {
        GameOverWidget->SetWinnerText(
            WinnerPlayerState, NormalizedPlayerScore);
        GameOverWidget->AddToViewport();
    }
}
```

ЗАКЛЮЧЕНИЕ

В результате выполнения практической работы была разработана система глобальной таблицы пользователей с использованием программного интерфейса Leaderboards Interface, расположенного в Epic Online Subsystem с автоматическим начислением очков после каждого завершеного матча.

Для отображения результатов матча было проведено изучение проблемы читаемости пользователем чисел с плавающей запятой. С учетом изученной проблемы был создан виджет, который отображает удобочитаемые данные, показывающие результаты конца матча.

Также был разработан виджет глобальной таблицы лидеров на уровне приложения с разделением пользователей на лиги.

Задание выполнено в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. EOS Game Services. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/game-services> (дата обращения 20.05.2023).
2. EOS Account Services. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/epic-account-services>. (дата обращения 16.05.2023);
3. Leaderboards Interface. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/leaderboards> (дата обращения 22.05.2023).
4. Leaderboards Sample. Unreal Engine Documentation [Электронный ресурс] // Режим доступа: <https://dev.epicgames.com/docs/leaderboards-sample> (дата обращения 22.05.2023).