УДК 004.773

Разработка браузерной клиент-серверной игры

Черкашин Александр Михайлович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Научный руководитель:
Лучанинов Дмитрий Васильевич
Приамурский государственный университет имени Шолом-Алейхема
Старший преподаватель информационных систем, математики и правовой информатики

Аннотация
Целью исследования является разработка серверного игрового приложения и описание структуры работы сетевого взаимодействие между клиентом и сервером. В работе использовалась библиотека Socket.IO для сетевого клиент-серверного взаимодействия и библиотека PlayCanvas для создания компьютерной игры. В результате созданы игровой сервер и клиент.
Ключевые слова: PlayCanvas, Socket.IO, Игровой сервер, межпроцессное взаимодействие, База данных, Web-сервер, php, JavaScript.

Development of a browser client-server game

Cherkashin Alexander Mihailovich
Sholom-Aleichem Priamursky State University
student

Scientific adviser:
Luchaninov Dmitry Vasilyevich
Sholom-Aleichem Priamursky State University
Senior Lector of Information Systems, Mathematics and Legal informatics

Abstract
The purpose of the study is to develop a server gaming application and a description of the structure of the network interaction between the client and the server. The work used the Socket.IO library for network client-server interaction and PlayCanvas library to create a computer game. As a result, a game server and client are created.
Keywords: PlayCanvas, Socket.IO, Game server, interprocessor interaction, Database, Web server, PHP, JavaScript.

1 Введение

1.1 Актуальность

В настоящий момент достаточно актуально разрабатывать серверное игровое приложение.

1.2 Обзор исследований

Л. Бишоп, Д. Эберли, Т. Уиттед, М. Финч, М. Шанц. рассматривают описание архитектура игровое движка в виде структура хранение игровые данные в виде дерево, каждый лист дерево это объект в определенный тип, также рассматривают рекурсивный перебор графа для обработки столкновение [1]. Исследователи Э.Ф. Андерсон, Ш. Энгель, П. Комнинос, Л. Маклафлин рассматривают проблему разработки компьютерные игровые технология [2]. П. Венер, К. Пибергер, Д. Герингер описывают возможности формата JSON для обмена сообщение с устройствами интернет вещей [3]. К.С. Лаккунди, В Аграхари, С. Чималаконда описывают возможность сайт Github добывать данные по запросу проекты игровые движки, и описывают проблему разработчиков компьютерные игры по выбору игровое движок [4]. Автор диссертационный работы R. Ojala описывает создания масштабируемый серверную систему многопользовательскую игровую систему, с сборкой кластеры и создание игровой сервер. Рассматривает вариант выбор язык программирования, игровой движок, операционная система. [5]. И. Куо рассматривает проблемы качество и устойчивость сбора данных из результаты игры в многопользовательской системе [6]. . Описывают архитектура многопользовательскую игровая система клиент — сервер Э. Кронин, Б. Филструп, А. Курц [7]. Т. Хендерсон рассматривает проблемы задержки в многопользовательская сетевая игра от жанра шутер от первого лица [8]. Авторы статьи Й. Д. Пеллегрино, Ц. Довролис рассматривают возможности сетевую архитектуру сервер — клиент и пиринговую сеть в многопользовательской игровую систему для исследование пропускной способности [9]. А Рауф, С Али, Х. Явед показывают проблемы с безопасности аутентификация пользователя в web приложения, и анализирует и поиске обнаружение уязвимости системы аутентификация с использование метод SQL инъекция [10].

1.3 Цель исследования

Целью исследования является разработка клиент - серверного игрового приложения и описание структуры работы сетевого взаимодействие между клиентом и сервер.

2 Методы исследования

В игровом решении использовано:

1. PlayCanvas — это фреймворк, написанный на языке javascript, визуальная платформа разработки интерактивных игры в веб приложений. Всё что разрабатывается с помощью PlayCanvas основано на возможностях HTML5 [11].

2. Socket.IO — это библиотека, который обеспечивает двунаправленного обмен данными в реальном времени связи между клиентом и сервером, используется протокол API WebSocket [12].

3. WebSocket — это протокол связи, который обеспечивает полнодуплексную и низкозадержку между сервером и клиентом [12].

В решении использована библиотеки:

- на стороне клиента Socket.IO,
- на стороне сервера PHPSocketIO.

В создания игровые карты применена программа 3D редактор Blender и рендерный движок Eevee. Для представление игры на рисунке, использовался рендеринг для рабочего проекта «игровая карта». Для рисования интерфейса игры - программа векторный редактор Inkscape. Каждый рисунок в интерфейс игры имеет художественные представления.

Данная статья описывает информационную модель.

3 Результаты и обсуждения

Данная система содержит web-приложение, систему аутентификации, игровой сервер, базу данных, клиентскую часть с использованием библиотека PlayCanvas. Web сервер используется Apache. Web-приложение написано на языке php является самописнный сайт.

3.1 Процесс создание игры и web приложение

В данной статье была создана одна игровая карта. Игровая карта реализована с помощью 3D редактора Blender (рис. 1.1), а также применялись текстуры и выполнялись экспорт 3D модели в формате fbx и импортировали в конструктор игр PlayCanvas Editor (рис. 1.2), игровые скрипты написаны на языке JavaScript. Web-приложение написано на языке php, использовался редактор Eclipse для создание web приложение. Web-приложение содержит систему аутентификация, игровой сервер, при разработки web-приложения применялась библиотека Router[13].

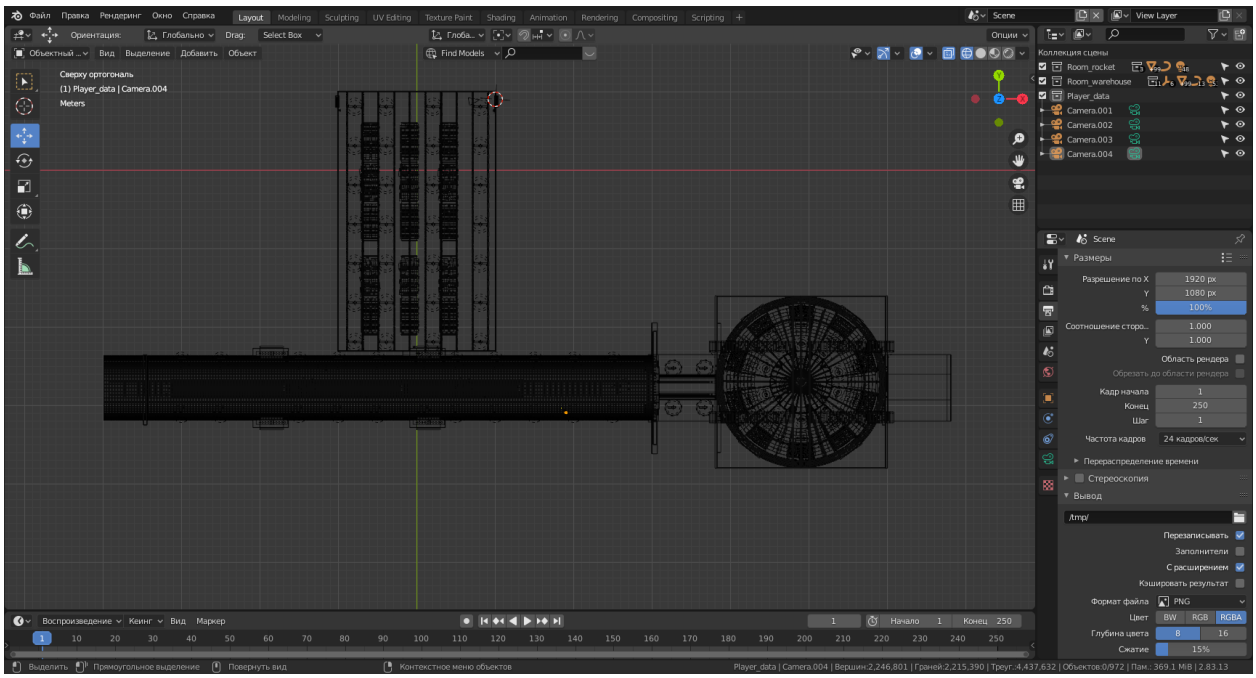

Рисунок 1.1. Игровая карта создана в программе Blender

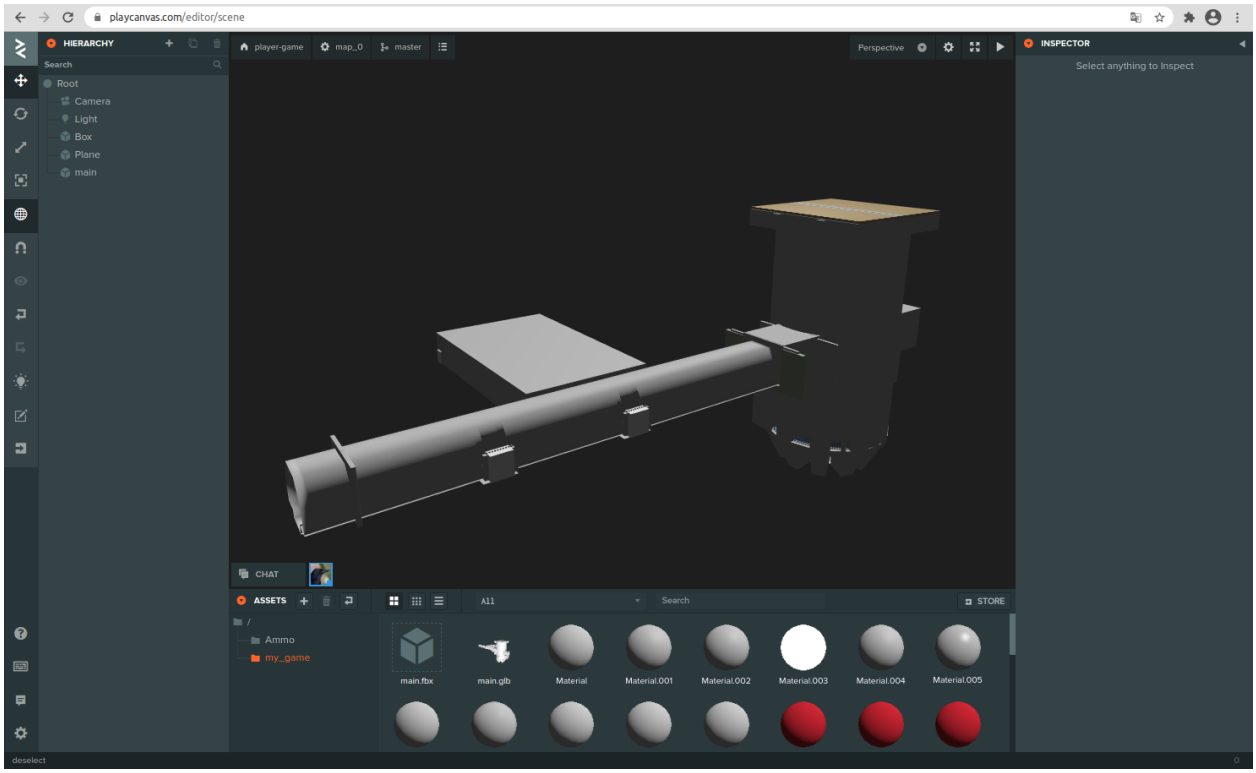

Рисунок 1.2. Игровая карта импортирована в конструктор игр в PlayCanvas Editor

3.2 Сервер база данных

Реализованное серверное программное обеспечение использует MariaDB. Сервер баз данных обеспечивает хранение данные (рис. 2.1).

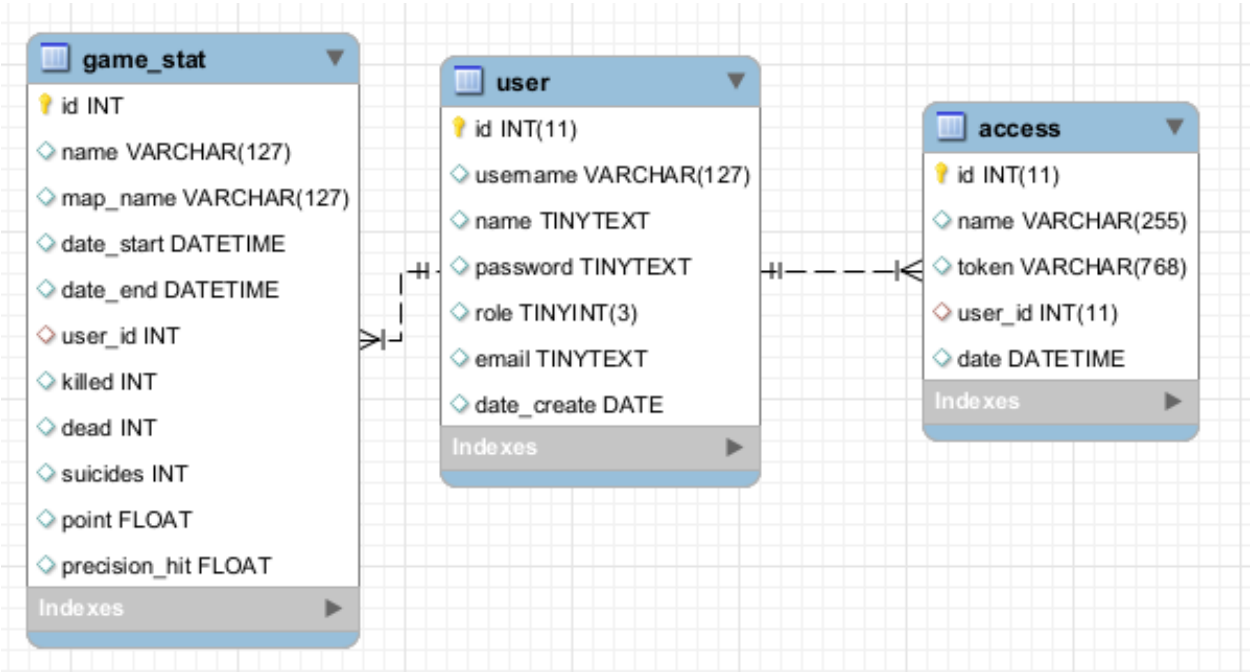

Рисунок 2.1. Логическая схема базы данных

Таблица «user» содержит данные о пользователях. Столбец «ID» — уникальный идентификатор, «username» — уникальное имя пользователя для входа систему, «name» — отображаемые имя, «password» — зашифрованный пароль, «role» — роль пользователя («0» — суперпользователь, «1» — обычный пользователь, «-1» — гость). «Email» — адрес электронной почты. «data_create» — дата регистрация пользователя (рис. 2.1).

Таблица «access» содержит данные о ключе для доступа пользователя (альтернативная аутентификация), используется игровым сервером для идентификация клиента. «ID» - уникальный идентификатор. «Name» — название токена. «token» — ключ (токен) для входа в систему. «user_id» — связь с таблицей «user». «Date» — дата создания ключа. Данные в таблице «access» хранятся в оперативной памяти, при остановке/перезапуске работы сервера, данные в таблице удаляется полностью (рис. 2.1).

Таблица «game_stat» содержит игровую статистику. «ID» - уникальный идентификатор. «name» - название игрового результата. «map_name» - название игровой карты. «date_start» - дата и время начала игры. «date_end» - дата и время конеца игры. «user_id» - связь с таблицей user». «killed» - количество побед. «dead» - количество побед противника. «suicides» - количество поражений самого игрока. «point» - количество очков. «precision_hit» - точность попадания противника, значение обычно от 0 до 1 (рис. 2.1).

3.3. Игровой сервер

Игровой сервер написан на языке PHP с использованием библиотеки SocketIO, протокол Socket.IO запускается через CLI. Сервер имеет доступ к серверу баз данных. Для обеспечения обмена объектами между клиентом и сервером использует формат JSON.

Игровой сервер имеет зависимость от клиента в роли «Master», клиент «Master» обеспечивает обработку столкновений и строит математическую модель физического симулятора, рассчитывает перемещение разных игроков. При отключении клиента «Master» игровой процесс полностью останавливается, все остальные клиенты «Slave» выводят на экран ожидание подключения (рис. 3.1).

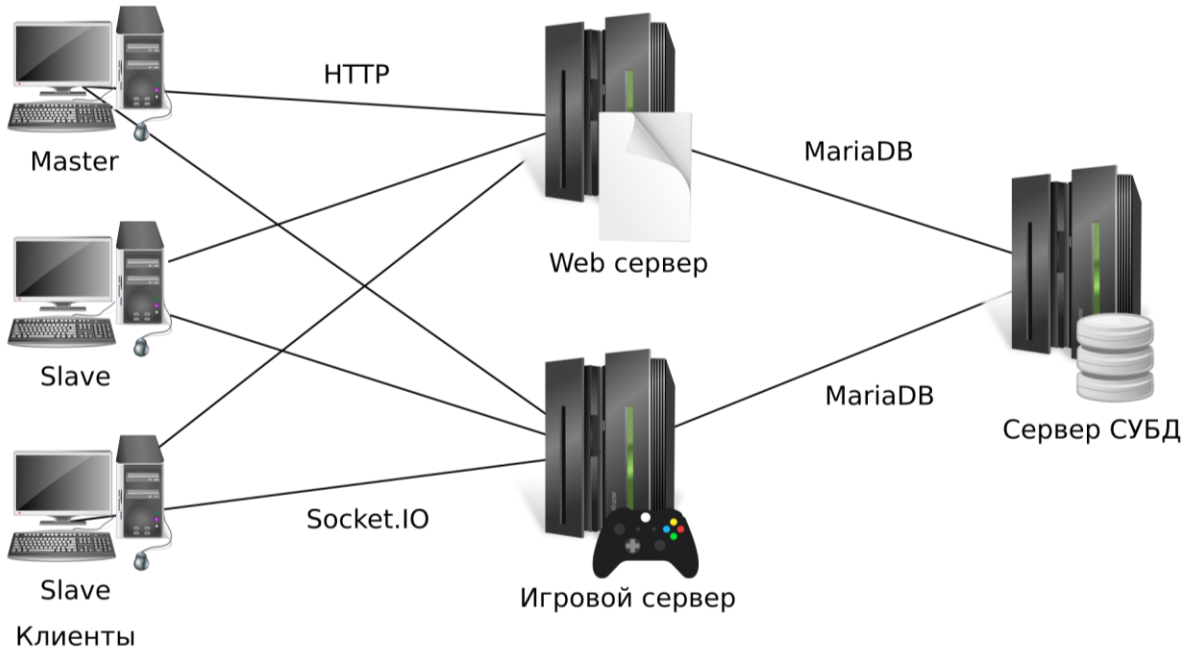

Рисунок 3.1. Сетевая архитектура

Переменная «user_data» тип данных массив, ключ — ID пользователя, значение объект «Player_data» (Таблица 2.1).

Переменная «client_id» тип данных массив, ключ — ID подключенного клиента, значение - ID пользователя.

Переменная «Session_player» содержит структуру «Session_player_str».

Переменная «Respawn_map» содержит структуру «Respawn_str».

Переменная «Master_player» тип данных число или массив, значение - ID подключенного клиента.

Таблица 1. Структура данных «Player_data»

Название	Тип данные	Описание
Token	String	Идентификация пользователя через таблицу «access» в базе данных
user_id	Int	Идентификация пользователя через таблицу «user» в базе данных
Role	Int	Роль пользователя. «0» - «Master», «1» - «Slave».
PlayerName	String	Имя игрока

Название	Тип данные	Описание
PlayerStatus	Int	Состояние игрока
Location	Vector3f	Вектор координат.
Rotation	Vector4f	Вектор поворота.
Health	Int	Атрибут «здоровье игрока»
Weapon	Array	Список «Weapon» содержимое.
sel_weapon	Int	Текущий выбранный список «Weapon» игрок в игре
Count_hit	Int	Количество попаданий в цель
Count_notHit	Int	Количество промахов
Precision_hit	Float	Точность попадания, значение от 0 до 1
Message	Array	Сообщение
Fps	Float	Частота кадров в секунду
Time_update	Float	Время последнего обновления (Секунда)
Time_create	Float	Время регистрации (Секунда)
Count_update	Int	Количество обновлений за 1 секунду
Stat_killed	Int	Количество побед игроков
Stat_suicides	Int	Количество поражений игрока
Stat_score	Int	Количество очков
Stat_dead	Int	Количество поражений игрока.

Структура данные «Player_data» представляет собой хранилище пользовательских данных. «Token» - ключ аутентификации пользователя альтернатива логину и паролю. «user_id» - обеспечивает связь с таблицей «user» базы данных. «Role» - роль пользователя, «Master» роль администратора, «Slave» - обычный пользователь, данные «Role» передаются из таблицы «user» баз данных. «PlayerStatus» – состояние игрока, 0 — активный игрок, 1 — мертвый игрок, 2 — игрок недоступен, 3 — зритель, игрок не участвующий в игре. «Location» – вектор третьего порядка, показыавет расположение игрока внутри игрового пространства. «Rotation» – вектор четвертого порядка, показывает поворот игрока внутри игрового пространства. «Health» – атрибут здоровье игрока, значение от 0 до 100. «Weapon» – список оружия игрока. «sel_weapon» – текущее оружие игрока игрока. «Precision_hit» – точность попадания. «Message» – сообщения игрока. «Fps» – частота обновления кадров в секунду. «Time_update» - время последнего обновления. «Time_create» – время регистрации игрока. «Count_update» – счетчик обновления. «Time_update», используется для нахождения «Fps».

Таблица 2. Структура данных «Weapon_str»

Название	Тип данные	Описание
Name	String	Название
Type	Int	Атрибут «Тип боя»
Phit	Float	Атрибут. Вероятность попадания от 0 до 1.
Damage	Int	Атрибут «Урон»
Temp	Float	Атрибут «Температура»
Temp_max	Float	Атрибут «Максимальная температура»
Cooling	Float	Атрибут «Охлаждение»
Asset_id	Int	Идентификатор игрового ресурса

Структура данных «Weapon_str» – содержит описание характеристик оружия для игрока. «Name» – название. «Type» – тип, 0 — ближний бой, 1 — дальний бой. «Phit» – вероятность попадания, значение от 0 до 1. «Damage» – урон, при атаке противника отнимает значение «Health». «Temp» – значение текущий температуре оружия. «Temp_max» – значение максимальной температуры. При достижении значения «Temp» оружие становится небоеспособным, игрок теряет способность к атаке. «Cooling» – скорость охлаждение оружия. «Asset_id» – идентификатор игрового ресурса.

Таблица 3. Структура данных «Message_str»

Название	Тип данные	Описание
Recipient_name	String	Название игрока получателя
Recipient_user_id	Int	Идентификация пользователя через таблицу «user» в базе данных (получателя)
Sender_name	String	Название игрока отправителя
Sender_user_id	Int	Идентификация пользователя через таблицу «user» в базе данных (отправителя)
Message	String	Текст сообщения
Datetime	Float	Временная отметка

Структура данных «Message_str» предназначена для хранения текстовых сообщений. «Recipient_name» и «Recipient_user_id» указатель получателя сообщений игрока. «Sender_name» и «Sender_user_id» указатель

отправителя сообщений игрока. «Sender_user_id» значение «-1» указатель уведомления. «Datetime» - временная отметка получения сообщения.

Таблица 4. Структура данных «Session_player_str»

Название	Тип данные	Описание
Name	String	Название игры
Map_name	String	Название игровой карты
Status	Int	Состояние игры
Date_start	Float	Время начала игры
Date_end	Float	Время конца игры
Stat_killed	Int	Общее количество победителей
Stat_suicides	Int	Общее количество проигравших игроков
Stat_score	Int	Общее количество очков
Limit_max_killed	Int	Максимальное значение критерия цель победы
Limit_max_respawn	Int	Ограничение возможностей игрока

Структура данных «Session_player_str» содержит информацию сеанса игры. «Name» – название сеанса. «Map_name» – название игровой карты. «Status» – режим игры, 0 — игра не запущена, 1 — режим «Deathmatch», 2 — режим песочницы, игра не имеет цели победить. «Date_start» – время начала игры. «Date_end» – время конца игры, при заданном числовом значении больше чем «Date_start» в начале игры, имеет ограничение по времени, при 0 игра не ограничена по времени, изменяется значение при завершении игры. «Stat_killed» – общее количество убитых игроков при поражение игроков. «Stat_suicides» – общее количество суицидов игроков. «Stat_score» – общее количество очков. «Limit_max_killed» – ограничение по которому при достижение значения «Stat_killed» определяется победитель, значение 0 — без ограничений, победитель не выявляется. «Limit_max_respawn» – ограничение по которому при достижении значения «Stat_dead» игрок теряет возможность восстанавливается (респаун), игрок проиграл, значение 0 — без ограничений.

Таблица 5. Структура данных «Respawn_str»

Название	Тип данные	Описание
Map_name	String	Название игровой карты
Location	Vector3f	Координаты вектора
Rotation	Vector4f	Координаты вектора поворота

Название	Тип данные	Описание
Probability	Float	Вероятность место респаун игрока
Asset_id	Int	Идентификатор игрового ресурса

Структура данных «Respawn_str» содержит данные о местоположении для респаун игрока.

Таблица 6. Коды и константы сообщение состояние «status» клиента и сервер.

Коды	Название	Описание
0	loggedIn	Пользователь вошел в систему
1	loggedOut	Пользователь вышел из системы
2	LoginFailed	Ошибка входа в систему
3	playerIsRegistered	Игрок вошел в игру
4	playerIsDeregistered	Игрок вышел из игры
5	playerIsError	Ошибка входа в игру
6	SessionStart	Начало игры
7	SessionEnd	Конец игры
8	PlayData	Клиент или сервер получил данные «Player_data»
9	PlayerInput	Пользователь вводит
10	PlayerOutput	Пользователь выводит
11	SendMesg	Игрок отправил сообщение
12	ReceivedMesg	Игрок получил сообщение
13	PlayerKilled	Игрок нанес поражение противнику
14	PlayerSuicides	Игрок совершил суицид
15	PlayerDead	Игрок потерпел поражение

Таблица 7. Структура данные «PlayerInput_str». Объект события.

Название	Тип данные	Описание
Type	Int	Тип события, 0 — клавиатура, 1 — мышь.
KeyName	String	Клавиатурный ввод символ
KeyCode	Int	Клавиатурный код
KeyUp	Bool	Клавиатурная кнопка отпущена
KeyDown	Bool	Клавиатурная кнопка нажата

Название	Тип данные	Описание
MouseCoord	Vector2f	Расположение курсора относительно элемента «Window» (DOM)
MouseMove	Vector2f	Измененные координаты курсора с последнего момента события мыши
MouseButton	Int	Кнопка мыши
MouseUp	Bool	Кнопка мыши отпущена
MouseDown	Bool	Кнопка мыши нажата

«MouseButton» - нажата кнопка мыши, 0 — левая кнопка мыши, 1 — средняя кнопка мыши, 2 — правая кнопка мыши.

Переменная «PlayerInput» определен в клиентской стороне, тип данные массив, значение объект «PlayerInput_str».

Таблица 8. Файловая структура проекта.

Название	Описание
/	Корень проекта
/index.php	Главный php скрипт
/asset	Файлы ресурсы игры
/asset/init.js	Скрипт для инициализация загрузки ресурсов.
/asset/script	Скрипты на JavaScript
/asset/models	3D модели
/asset/texture	Текстуры
/asset/material	Материалы
/asset/fonts	Шрифты
/map	Игровая карта
/lib	Библиотека JavaScript и php
/vendor	Библиотека php созданный composer
/page	Шаблоны для формирование web страницы
/page/layer	Шаблоны для формирование «шапка» и «подвал» web страницы
/page/blab_game.php	Скрипт php формирование страницы для игры
/page/login.php	Форма авторизация пользователя
/page/signup.php	Форма регистрация нового пользователя
/page/signin.php	Скрипт обработка входа и выхода пользователя и регистрация пользователя

Название	Описание
/*service*/game.php	Скрипт php для игровой сервер
/index.php	Главный скрипт php
/loader.php	Загрузчик скриптов php
/controller	Контроллер, делает адресную структура url
/stdata_auth.php	Реализация система аутентификация

3.4 Идентификация пользователя

В системе предусмотрена аутентификация и авторизация для расширения прав пользователя в игровом процессе. В гости могут только наблюдать за игровым процессом.

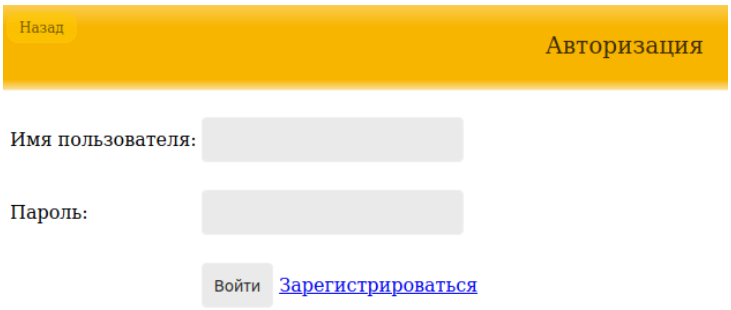

Рисунок 4. 1. Авторизация пользователя

Вход в систему осуществляется вводом имени пользователя и пароля, после входа в систему, система сохраняет сессию, отправляет клиенту Cookie, и перенаправляет страницу где требовалось авторизоваться.

В системе есть ссылка «зарегистрироваться».

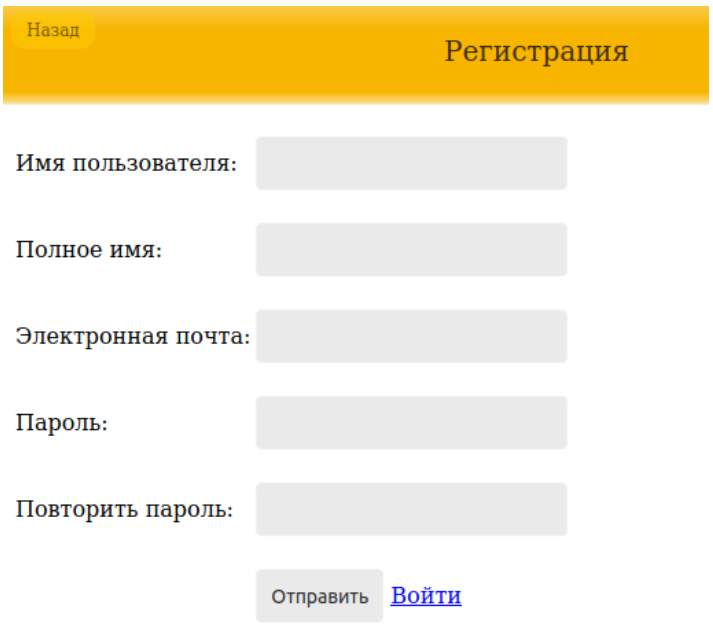

Рисунок 4.2. Регистрация пользователя

Гость в системе заполняет имя пользователя для авторизации, «Полное имя» для отображения имя игрока, «электронная почта» используется для восстановления входа в систему в случае утраты возможности получения доступа к системе. Роль по умолчанию для новых пользователя равна 1. Роль администратора 0, роль для гостя -1. После регистрация нового пользователя система вносит изменения в таблицу базы данных.

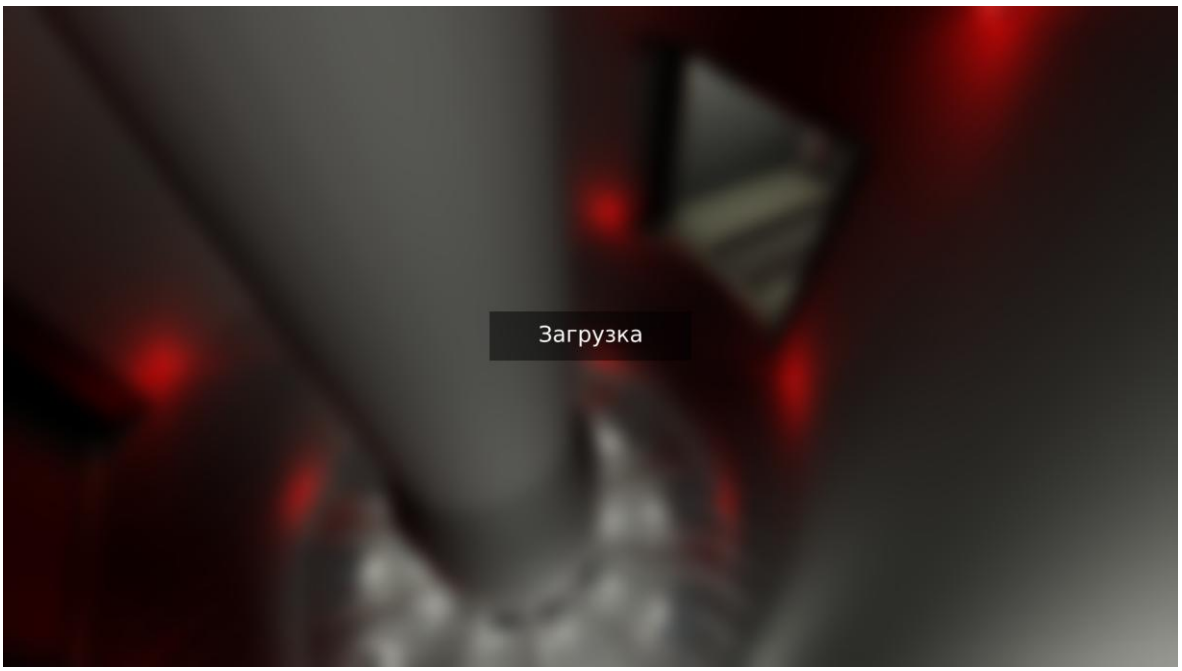

Рисунок 5.1. Загрузка

3.5. Загрузка игровых данных

Пользователь входит в страницу игры. На web серверной стороне генерируется токен который вносит запись в таблицу «access» базы, с названием «game_player», столбец «name» и привязывает идентификатор пользователя «user_id». В клиентской части при загрузке страницы, загружается JavaScript и Html, исполняет его, скрипт загружает библиотеку PlayCanvas и Socket.IO и выводит на страницу «загрузка» (рис 5.1). После успешной загрузки страницы, скрипт «/main.js» загружает и включает скрипт например «/map/missile_silos/main.js», инициализирует и загружает данные (карту, материалы, текстуры, и т.д), после успешной загрузки, скрипт Socket.IO подключает к игровому серверу, если подключение не удалось то Socket.IO автоматический повторяет подключение до тех пор пока не подключится к игровому серверу.

После успешного подключения на стороне сервера срабатывает функция «connection» и генерирует уникальный идентификатор, делая «рукопожатие», вносит в список переменную «client_id».

По полученному токену игровой сервер регистрирует игрока (проверяет базу данных «access» столбец «token») и определяет роль пользователя, после успешной регистрация сервер отправляет (функцию «status» аргумент «loggedIn») в случае ошибки сервер отправляет (функцию

«status» аргумент «LoginFailed»), в клиентской части выводит сообщение об ошибке пользователю. Клиент отправляет запрос «play» игровому серверу, сервер проверяет свободное место в списоке «user_data» и, в случае наличия свободного места (максимальное количество игроков 32), добавляет его в список, отправляет «status» аргумент «playerIsRegistered» и «PlayerData» аргумент «Player_data» клиенту, «statusMsg» аргумент «playerIsRegistered» всем остальным клиентам.

Клиент, получая обработчик «status» и «PlayerData», разграничивает возможности пользователя управлять игрой (рис 5.2). При ввода с клавиатуры и мыши пользователя клиент автоматический отправляет обработчик «PlayerInput», игровой сервер, получая обработчик «PlayerInput» отправляет «PlayerInput» аргумент «user_id» к клиенту «Master», клиент «Master» обрабатывает игровой процесс и автоматический отправляет обработчик «PlayerOutput» аргумент «Player_data» игровому серверу. Игровой сервер, получая «PlayerOutput», отправляет «PlayerOutput» всем клиентам.

Рисунок 5.2. Игровой интерфейс.

3.6. Игровой процесс

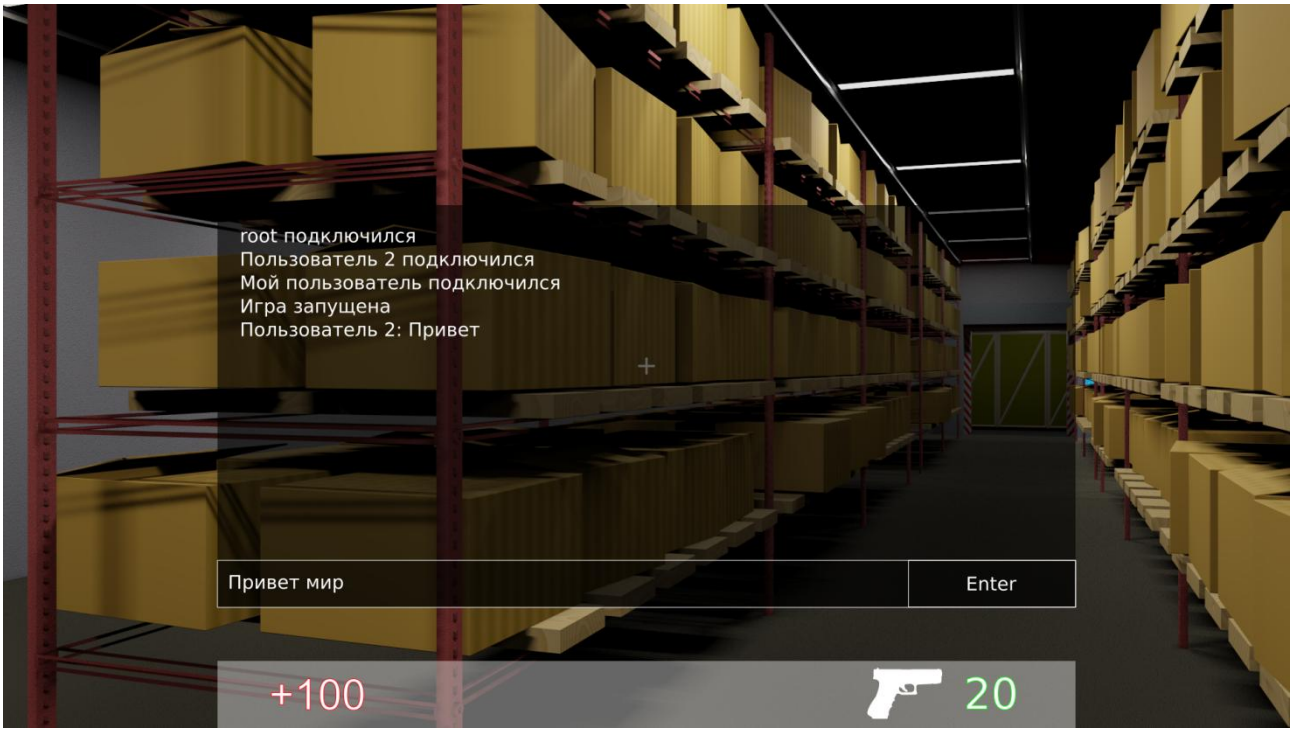

Рисунок 5.3. Окно сообщение

В данной игровой системе предусмотрен чат во время игры. Ввод осуществляется с клавиатуры. Клавиша «~» открывает/закрывает панель ввода чата, при вводе в сообщение символа «#» можно указать получателя сообщения, без «#» получатели сообщения — все.

Клиент отправляет «PlayerMessage» в игровой сервер. Игровой сервер добавляет в список «Message» из переменный «Player_data» в «user_data» как отправитель сообщения и добавляет в список как получатель сообщение и отправляет «PlayerMessage» указанному клиенту сообщение, у клиента отображается окно сообщения (рис 5.3).

Здоровье игрока, «Health», восстанавливается («регенерация») через 30 секунд (по умолчанию), через 5 секунд достигает 100%, после ранений (рис 5.3). Оружие по умолчанию у игрока бейсбольная бита и пистолет. Игрок не имеет ограничений на колическтво патронов, но есть ограничение по температуре, при открытие огня температура «Temp» увеличивается, при достижении максимального значения, игрок временно теряет возможность открывать огонь, пока «Weapon» не «остынет», не достигнет минимального значения заданного «Temp_min», скорость остывания определяется значением «Cooling» в секунду, по достижение минимальное значение «Temp» игрок вновь может открыть огонь. Точность попадание в цель определяет переменная «Phit» (таблица 2.2). Урон определяет переменная «Damage».

В игровой системе использует раунды только игровой режим «Deathmatch», перед началом раунда клиент «Master» отправляет «status» аргумент «SessionStart» игровому серверу. Игровой сервер принимает запрос

«status» и в определенный список переменных «Respawn_map» распределяет игроков (раздает координаты) внутри игрового пространства.

В игровой системе предусмотрено два игровых режима – «Deathmatch» (цель игры - победить всех) и «sandbox» (режим песочницы, цель игры отсутствует). Как только один игрок победит всех, игра завершается с результатом. При завершении игры клиент «Master» отправляет «status» аргумент «SessionEnd» серверу. Игровой сервер завершает процесс игры и вносит изменения в переменную «Session_player», отправляет результаты переменной «Session_player» всем клиентам. В графическом интерфейсе клиента отображается окно с результатами (рис 5.4).

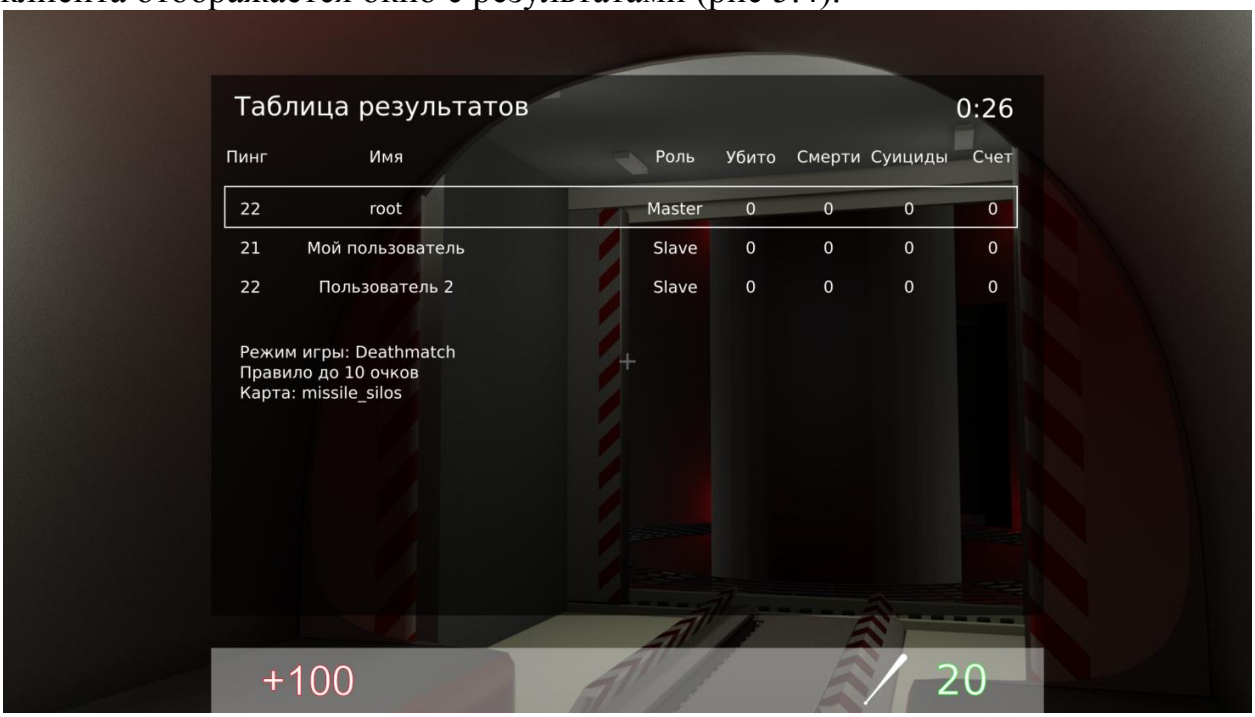

Рисунок 5.4. Таблица результатов. Нажата клавиша «Tab»

4 Выводы

В данной системе разработано web-приложение для генерация web-страниц, реализована аутентификация для пользователя, что позволяет идентифицировать пользователя и хранить данные игровых достижений, реализованы разные роли пользователей. Создана база данных для хранения данных о пользователях и игровой статистики. Разработан игровой сервер для взаимодействия клиентов, позволяющий работать с локальной сетью. Игровой сервер позволяет взаимодействовать между клиентами с использованием протокола Socket.IO. Данный Socket.IO реализует событийно-ориентированную модель, что упрощает разработку web-приложения, обеспечивая постоянное соединение до завершение сеанса.

В результате проекта было полностью разработано три серверных приложения.

Библиографический список

1. Bishop L. et al. Designing a PC game engine //IEEE Computer Graphics and Applications. 1998. Т. 18. №. 1. С. 46-53.
2. Anderson E. F. et al. The case for research in game engine architecture // Proceedings of the 2008 Conference on Future Play: Research, Play, Share. 2008. С. 228-231.
3. Wehner P., Piberger C., Göhringer D. Using JSON to manage communication between services in the Internet of Things //2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC). IEEE, 2014. С. 1-4.
4. Lakkundi C. S., Agrahari V., Chimalakonda S. GE852: A dataset of 852 game engines //arXiv preprint arXiv:1905.04482. 2019.
5. Ojala R. Scalable Multiplayer Components. 2020.
6. Kuo Y. et al. Community-based game design: experiments on social games for commonsense data collection //Proceedings of the acm sigkdd workshop on human computation. 2009. С. 15-22.
7. Cronin E., Filstrup B., Kurc A. A distributed multiplayer game server system //University of Michigan. 2001.
8. Henderson T. Latency and user behaviour on a multiplayer game server //International Workshop on Networked Group Communication. Springer, Berlin, Heidelberg, 2001. С. 1-13.
9. Pellegrino J. D., Dovrolis C. Bandwidth requirement and state consistency in three multiplayer game architectures //Proceedings of the 2nd workshop on Network and system support for games. 2003. С. 52-59.
10. Ali S., Shahzad S. K., Javed H. Sqlipa: An authentication mechanism against sql injection //European Journal of Scientific Research. 2009. Т. 38. №. 4. С. 604-611.
11. Обзор возможностей PlayCanvas для создания Web VR приложений // Хабр URL: https://habr.com/ru/post/445692/ (дата обращения: 2020-12-12).
12. Introduction // Socket.IO URL: https://socket.io/docs/v3 (дата обращения: 2020-12-21).
13. GitHub - izniburak/php-router: simple and flexible Router class for PHP. with Controllers and Middlewares support. // GitHub URL: https://github.com/izniburak/php-router (дата обращения: 2021-04-22).