**СОДЕРЖАНИЕ**

[Cписок использованных обозначений, сокращений и терминов 7](#_Toc516235709)

[Введение 8](#_Toc516235710)

[1 Техническое задание на создание системы 10](#_Toc516235711)

[1.1 Назначение и цели создания системы 10](#_Toc516235712)

[1.2 Характеристика объекта автоматизации 10](#_Toc516235713)

[1.3 Общие требования к системе 10](#_Toc516235714)

[1.4 Требования к функциям, выполняемым системой 11](#_Toc516235715)

[1.5 Требования к видам обеспечения 13](#_Toc516235716)

[1.6 Анализ аналогичных разработок 14](#_Toc516235717)

[2 Информационное обеспечение системы 17](#_Toc516235718)

[2.1 Выбор средств управления данными 17](#_Toc516235719)

[2.2 Проектирование файлов данных 18](#_Toc516235720)

[2.3 Организация сбора, передачи, обработки и выдачи информации 18](#_Toc516235721)

[3 Алгоритмическое обеспечение системы 20](#_Toc516235722)

[3.1 Алгоритм отрисовки игровых объектов 20](#_Toc516235723)

[3.2 Алгоритм отрисовки интерфейса 23](#_Toc516235724)

[3.3 Алгоритм игрового цикла (Сервер) 26](#_Toc516235725)

[4 Программное обеспечение системы 29](#_Toc516235726)

[4.1 Структура программного обеспечения и функции его компонентов 29](#_Toc516235727)

[4.2 Выбор компонентов программного обеспечения 29](#_Toc516235728)

[4.3 Разработка прикладного программного обеспечения 31](#_Toc516235729)

[4.4 Особенности реализации, эксплуатации и сопровождения системы 41](#_Toc516235730)

[4.5 Руководство пользователя 41](#_Toc516235731)

[5 Тестирование системы 45](#_Toc516235732)

[5.1 Условия и порядок тестирования 45](#_Toc516235733)

[5.2 Исходные данные для контрольных примеров 45](#_Toc516235734)

[5.3 Результаты тестирования 49](#_Toc516235735)

[6 Экономический раздел 50](#_Toc516235736)

[6.1 Расчёт трудозатрат на разработку программного продукта 50](#_Toc516235737)

[6.2 Амортизационные отчисления 54](#_Toc516235738)

[6.3 Экономическая эффективность 56](#_Toc516235739)

[6.4 Расчет плановой прибыли 56](#_Toc516235740)

[Заключение 58](#_Toc516235741)

[Список использованых источников 59](#_Toc516235742)

[Приложение А 61](#_Toc516235743)

# Cписок использованных обозначений, сокращений и терминов

ИС – информационная система;

Игрок – пользователь информационной системы;

ПО – программное обеспечение;

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

# Введение

Танк – бронированная боевая машина, чаще всего на гусеничном ходу, как правило, с пушечным вооружением в полноповоротной башне в качестве основного. Своим появлением танки обязаны Первой мировой войне.

Танки всегда вызывали интерес у создателей всевозможных action-игр, как и любая военная техника.

Первая игра по танкам была создана в 1985 году. Это была японская игра «Battle City» (с англ. – «Город битв») – компьютерная игра для игровых приставок Famicom и Game Boy. В России и странах СНГ выпускалась на пиратских картриджах как в оригинальном виде, так и в модификации Tank 1990, и известна под неофициальным названием «Танчики».

Сейчас существует множество вариаций данной игры, но в целом процесс игры всегда остается практически одинаковым. Чаще всего они предлагают пользователю выбор среди различных видов танков, позволяет играть в компании друзей.

Проект является очень актуальным в данный момент времени, сейчас интернет предлагает огромное разнообразие всевозможных игр и пользователям будет особенно интересно поиграть в танчики, так как любая вариация этой игры вызывала неизменно высокий интерес у игроков. Также данный проект является неплохой возможностью продемонстрировать работу с JavaScript.

PixelTank – игра между множеством противников на квадратном поле 5500 x 5500 (не включая границы).

Игрок может управлять одним из танков на поле. При этом он противостоит всем остальным игрокам, управляющим танками-противниками. Основная задача игрока – уничтожить все танки противников и набрать максимальное количество очков.

У каждого игрока есть ограниченное количество здоровья (брони танка). Стреляя в своих противников, игрок сокращает количество их здоровья, как только здоровье полностью кончается, танк игрока считается уничтоженным.

В случае, если танк игрока будет уничтожен, он может начать игру заново, при этом, его количество очков будет прежним.

Игра ведет статистику по количеству уничтоженных игроком танков и по количеству полной потери здоровья.

В самом начале игры, игроки могут выбирать себе любой танк из трех возможных вариантов:

* легкий танк (самый быстрый танк с самой слабой прочностью, скорострельность);
* средний танк (универсальный танк, средняя скорость, прочность ниже, чем у мощного танка, но выше, чем у легкого танка);
* тяжелый танк (самый медленный танк с самой мощной прочностью, большой урон).

# Техническое задание на создание системы

## Назначение и цели создания системы

Назначение системы заключается в обеспечении функций носящая не столько соревновательный, сколько развлекательный характер.

Основными целями создания системы являются:

* развлечение пользователей игры.

## Характеристика объекта автоматизации

### Общее описание

Объектов автоматизации является игровой процесс. В который входит разработка теории и правил игры.

Пользователями системы являются игроки.

### Состав и порядок функционирования

Требования к оборудованию:

* доступ к сети интернет;
* бесперебойная подача питания;
* защита от несанкционированного воздействия.

Требования к процессам взаимодействия с ИС:

* в соответствии с стандартами и законам.

## Общие требования к системе

### Требования к структуре и функционированию системы

Система должна представлять собой онлайн игру. В которой пользователи системы быстро получают возможность играть без необходимости в регистрации. В случае отсутствия игроков на сервере создавать искусственных пользователей для сохранения игровой составляющей системы.

Выделяемые подсистемы:

* подсистема сервера игры;
* подсистема клиента игры;
* подсистема искусственных пользователей.

### Дополнительные требования

Требуется обслуживание системы как минимум одним системным администратором. Обучение пользователя игровому управления не должен занимать много времени.

## Требования к функциям, выполняемым системой

### Подсистема сервера игры

Сервер представляет собой авторитарный сервер. Использование авторитарного сервера требует от сервера просчёта и выполнения всех действий, затрагивающих мир игры, применение правил игры и обработки ввода от игроков-клиентов. Каждый клиент высылает введенную информацию (в форме нажатия клавиш или запрашиваемых действий) на сервер и постоянно получает от сервера текущее состояние игры. Клиент не может самостоятельно вносить какие-либо изменения в состояние игры. Вместо этого, он сообщает серверу, что конкретно он хочет сделать, сервер обрабатывает этот запрос и отвечает клиенту, объясняя что получилось в результате запроса.

Принципиально важно, что существует разница между тем, что игрок хочет сделать и тем, что в действительности происходит. Это позволяет серверу принимать запросы от каждого клиента, после чего решать, как изменить состояние игры.

Преимущество такого подхода в том, что он значительно усложняет клиентам возможность использования нечестных приёмов. Например, у клиента нет возможности обмануть сервер (и, соответственно, остальных клиентов), что враг был убит, поскольку клиент не принимает таких решений. Он может только лишь сказать серверу, что оружие выстрелило и, начиная с этого момента, только сервер определяет, было ли совершено убийство или нет.

Сроки и периодичность определяется активностью пользователя. Формализуемость: высокая. Источники информации: конфигурация параметров игровых объектов, пользователь. Показатели: число одновременно работающих пользователей (зависит от производительности сервера, не рассчитывается). В качестве выходных данных: информация о пользователях и их состоянии, а также игровые события.

### Подсистема клиента игры

Возможным недостатком системы с авторитарным сервером можно назвать время, которое требуется для пересылки сигнала через сеть. Если игрок нажал кнопку, чтобы двинуться вперёд, а ответ от сервера о возможности такого движения приходит через десятую долю секунды, то такая задержка будет ощутима для игрока. Одним из возможных решений этой проблемы будет использование так называемых прогнозов движения на стороне клиента client-side prediction. Суть этой технологии в том, чтобы позволить клиенту обновлять локальную версию состояния игры, но при этом иметь возможность получать уточняющие данные со стороны сервера по мере необходимости. Как правило, это применяется только для простых действий в игре и не должно оказывать влияние на значительные изменения логики игры. Например, было бы неразумно сообщать игроку, что враг был убит только лишь затем, чтобы сервер в дальнейшем переписал это решение.

Из-за чего пользователь имеет минимум влияющих действия на игру и должен быть доступен интерфейс выбора начальных игровых данных и функция отправки действий пользователя.

Сроки и периодичность определяется обновлением данных сервера. Формализуемость: высокая. Входными данными, в этом случае будут выступать: выбор начальных параметров, а также данные о действиях. Выходными данными будут выступать генерируемые данные о стоянии пользователя в момент времени.

### Подсистема искусственных пользователей

Подсистема должна имитировать действия пользователя. Осуществляющее свое поведение на основе информации об окружающих пользователях. Входными данными, в этом случае будут выступать: выбор начальных параметров случайным образом, а также данные о действиях и состоянии окружения. Выходными данными будут выступать генерируемые данные о стоянии пользователя в момент времени.

## Требования к видам обеспечения

### Требования к информационному обеспечению

В системе должны быть обеспечено хранение следующей информации:

* информация о всех пользователях в игре;
* информация о стоянии игровых объектов.

Состав, структура и организация данных в АС обеспечивает выполнение ее функций.

### Требования к алгоритмическому обеспечению

Пользовательский интерфейс системы должен быть выполнен на русском языке.

Вся документация, разрабатываемая в рамках создания системы, должна быть представлена на русском языке.

При разработке программного обеспечения системы должны использоваться эффективные языки высокого уровня. Выбор языков программирования для создания АС производится на стадии разработки технического проекта.

### Требования к программному обеспечению

Требования к используемому программному обеспечению системы определяются на стадии разработки технического проекта. При выборе программного обеспечения АС необходимо учитывать требования к патентной чистоте.

В состав программных средств должны входить:

Для серверной части:

* Node.js;
* менеджер пакетов npm.

Для рабочих мест пользователей:

* любая ОС с поддержкой современного браузера.

Пользовательский интерфейс, реализованный в виде «тонкого клиента» должен быть полностью совместим с наиболее распространенными Интернет-браузерами: Safari 11.0 и выше, Chrome 66.0 и выше.

## Анализ аналогичных разработок

*Agar.io*

Agar.io – двухмерная массово-многопользовательская браузерная игра в жанре .io, созданная пользователем Reddit под ником Zeach. В Agar.io игрок контролирует круг, используя мышь и клавиатуру. Цель игры заключается в увеличении круга посредством поглощения гранул, которые случайным образом появляются на карте, и кругов других игроков. На рисунке 1.1 представлен вид главной страницы игры.

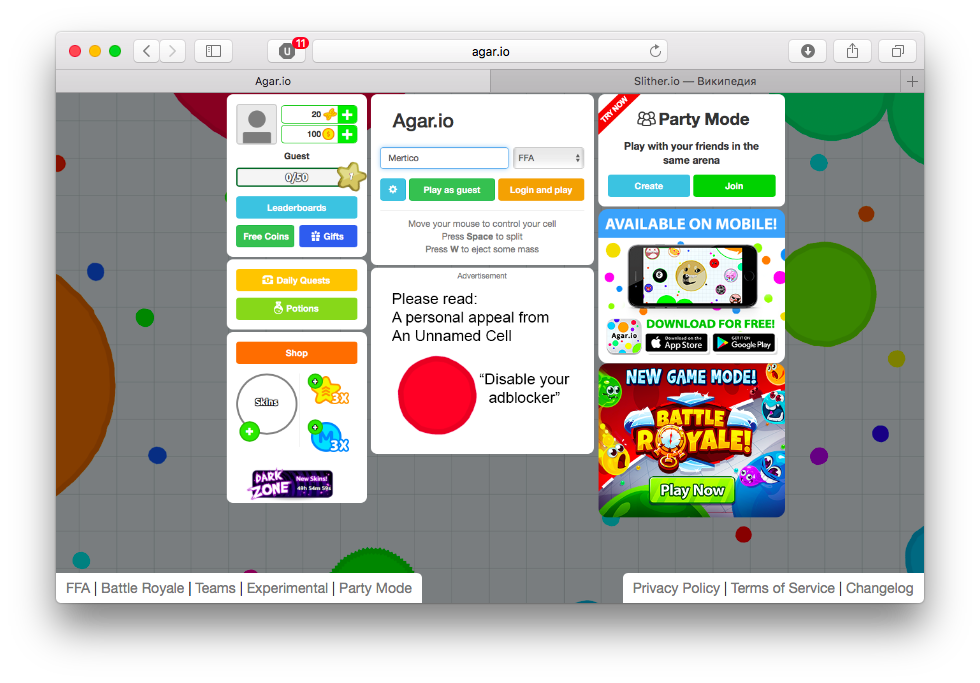


Рисунок 1.1 – Главная страница игры Agar.io

В последнее время игра набрала очень большую популярность. В игре есть распределение игроков по континентам (или крупным странам): Северная Америка, Южная Америка, Европа, Россия, Турция, Восточная Азия, Китай и Океания. Континент определяется автоматически, но его можно изменить в настройках. В правом верхнем углу экрана находится список лидеров (Leaderboard) – 10 самых массивных игроков в данной комнате.

Резюмируя вышесказанное, к достоинствам системы можно отнести приятный для глаза дизайн.

*Slither.io*

Slither.io – браузерная игра, в которой необходимо управлять змеёй и поглощать шарики, расти, чтобы в итоге сделать свою змею самой большой на сервере. На рисунке 1.2 представлен вид главной страницы игры.

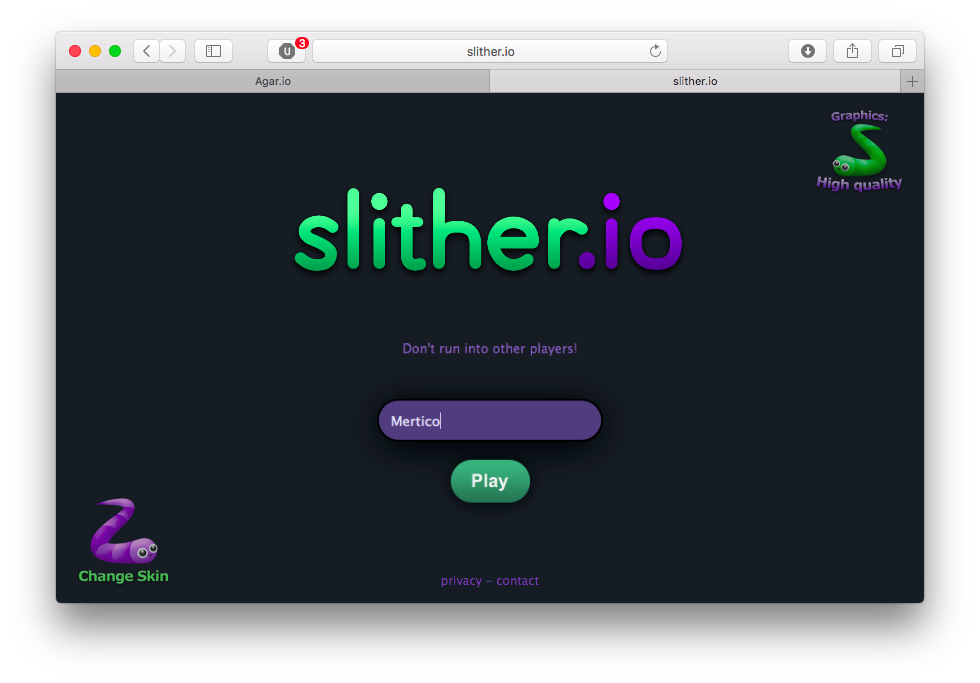


Рисунок 1.2 – Главная страница игры Slither.io

Игровой процесс подобен Agar.io. Целью является управление змеёй, перемещение в пространстве, поглощение шариков, уничтожение противников с последующим их поглощением. Чем больше шариков поглощено, тем больше становится змея, до тех пор, пока она не станет наибольшей в этот момент игры. Если голова змеи касается другого игрока, то змея уничтожается и надо начинать игру заново. Побеждённая змея превращается в яркие светящиеся точки, которые может съесть другой игрок.

Достоинства: приятный дизайн, минимальная выводимая информация.

# Информационное обеспечение системы

## Выбор средств управления данными

В разрабатываемой системе предполагается активное использование достаточно большого количества изменяемых данных представляющие сведение об игроках и игровых объектах. В один игровой цикл осуществляется множество перемещений игровых объектов, а также создание и удаление объектов, а в секунду осуществляется 60 игровых циклов, поэтому для организации, хранения и обработки данных нерационально использовать существующие СУБД, а необходимо использовать внутренние объекты.

Так для работы информационной системы используется конфигурационный файл, в котором описаны ключевые характеристики игровых объектов. Конфигурационный файл читается при запуске сервера и определят характеристики работы сервера, влияющие на игровой баланс, а также нагрузку на систему в целом. Так например увеличение числа отправок данных клиенту увеличит плавность игры, но увеличит нагрузку на канал связи. В свою очередь уменьшая число обновление игрового мира приведут к падению точности расчета физики, а также возможность перелета снарядов сквозь противников в некоторых ситуациях.

Спецификация конфигурационного файла представлена в таблице 2.1., дополнительная спецификация характеристики танков в таблице 2.2.

Таблица 2.1 – Спецификация конфигурационного файла

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Описание** |
| FPS | Число | Количество итераций просчета игрового мира в секунду |
| FPS\_send | Число | Частота отправки обновлений (в секунду) |
| tank | Объект | Объект содержит в себе характеристики танков («light», «medium», «heavy») |

Таблица 2.2 – Спецификация характеристики танков

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Описание** |
| hp | Число | Показать здоровья |
| max\_speed | Число | Максимальная развиваемая скорость |
| acceleration | Число | Ускорение |
| reload | Число | Скорость перезарядки (в секундах) |
| damage | Число | Наносимый урон |
| chassis.angle | Число | Угол поворота корпуса (в градусах) |
| turret.angle | Число | Угол поворота башни (в градусах) |

## Проектирование файлов данных

В создаваемой информационной системе хранение файлов не предусматривается.

## Организация сбора, передачи, обработки и выдачи информации

### Сбор информации

Для разработки информационной системы не устанавливалось дополнительных требований к сбору информации с помощью специальных технических средств, в виду этого, всю требуемую для функционирования системы информацию, пользователь может ввести самостоятельно в предназначенную для этого форму.

### Передача информации

Передача информации в ИС производится при помощи всемирной системы объединённых компьютерных сетей интернет при помощи протокола HTTP. Основой которой является технология клиент-сервер, и предполагает существование клиентов, которые инициируют соединение и посылают запрос, и серверов, которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом клиенту, клиент принимает его ответ.

Для осуществления связи клиента и сервера используется WebSocket – протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

В настоящее время в W3C осуществляется стандартизация API WebSockets.

WebSocket разработан для воплощения в web-браузерах и web-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket – это независимый протокол, основанный на протоколе TCP. Он делает возможным более тесное взаимодействие между браузером и web сайтом, способствуя распространению интерактивного содержимого и созданию приложений реального времени.

### Технология выдачи информации

Выдача информации осуществляется на экран монитора, при помощи специального прикладного программного обеспечения для просмотра web-страниц (браузера), которая генерирует экранную форму на основании данных полученных по каналам связи.

# Алгоритмическое обеспечение системы

## Алгоритм отрисовки игровых объектов

На рисунке 3.1 приведена структура отрисовки игровых объектов содержит в себе функции отрисовки снарядов на рисунке 3.2 и функции отрисовки танков на рисунке 3.3.



Рисунок 3.1 – Блок-схема структуры отрисовки игровых объектов

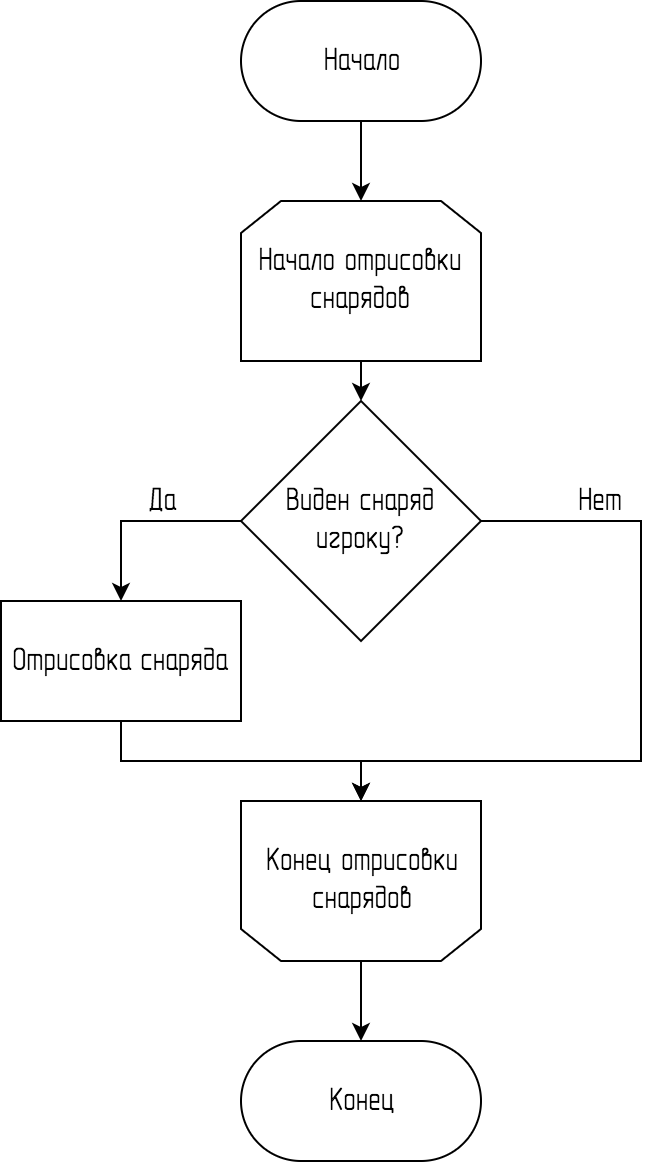


Рисунок 3.2 – Блок-схема структуры функции отрисовки снарядов

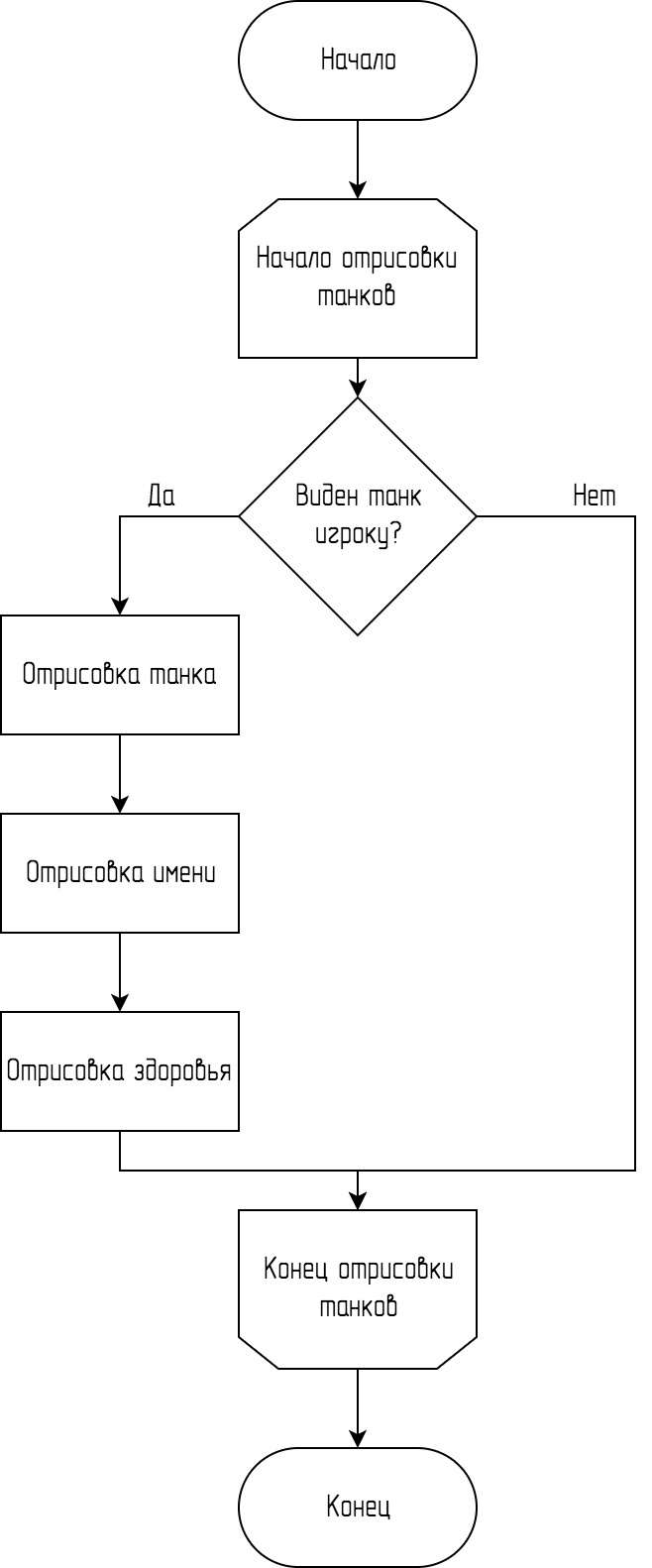


Рисунок 3.3 – Блок-схема структуры функции отрисовки танков

## Алгоритм отрисовки интерфейса

На рисунке 3.4 приведена структура отрисовки игрового интерфейса содержит в себе функции отрисовки радара на рисунке 3.5, функции отрисовки событий на рисунке 3.6 и отрисовка статистики на рисунке 3.7.



Рисунок 3.4 – Блок-схема структуры алгоритма отрисовки интерфейса

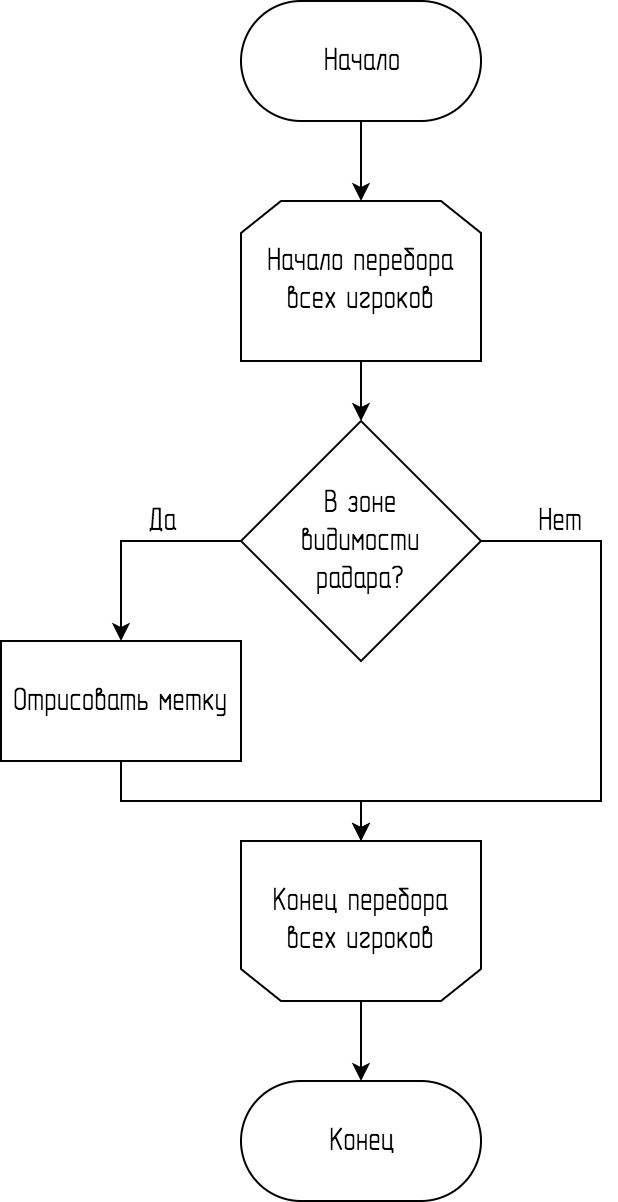


Рисунок 3.5 – Блок-схема структуры функции отрисовки радара



Рисунок 3.6 – Блок-схема структуры функции отрисовки событий

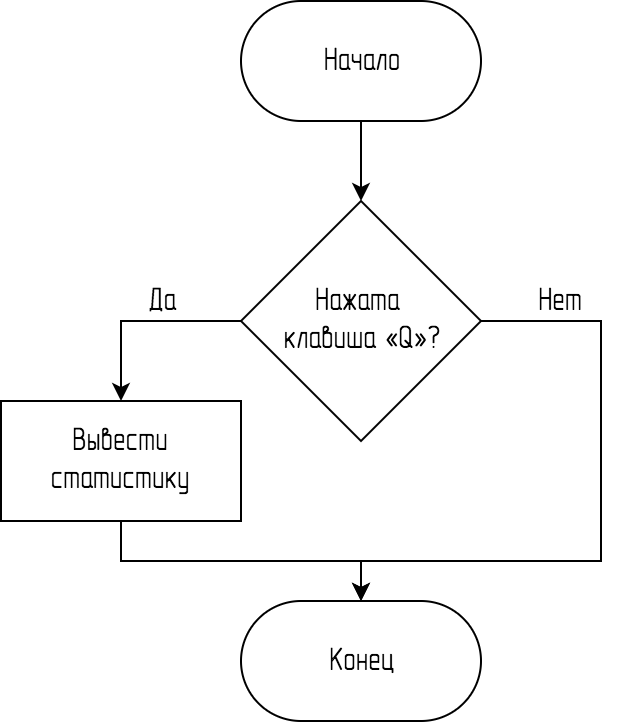


Рисунок 3.7 – Блок-схема структуры функции отрисовки статистики

## Алгоритм игрового цикла (Сервер)

На рисунке 3.8 приведена структура алгоритма игрового цикла. Представляет собой последовательность функций выполняемые в каждый игровой цикл. Содержит в себе функции перемещения снарядов на рисунке 3.9, просчет попаданий снарядов, перемещение игроков и добавление снарядов.



Рисунок 3.8 – Блок-схема структуры алгоритма игрового цикла

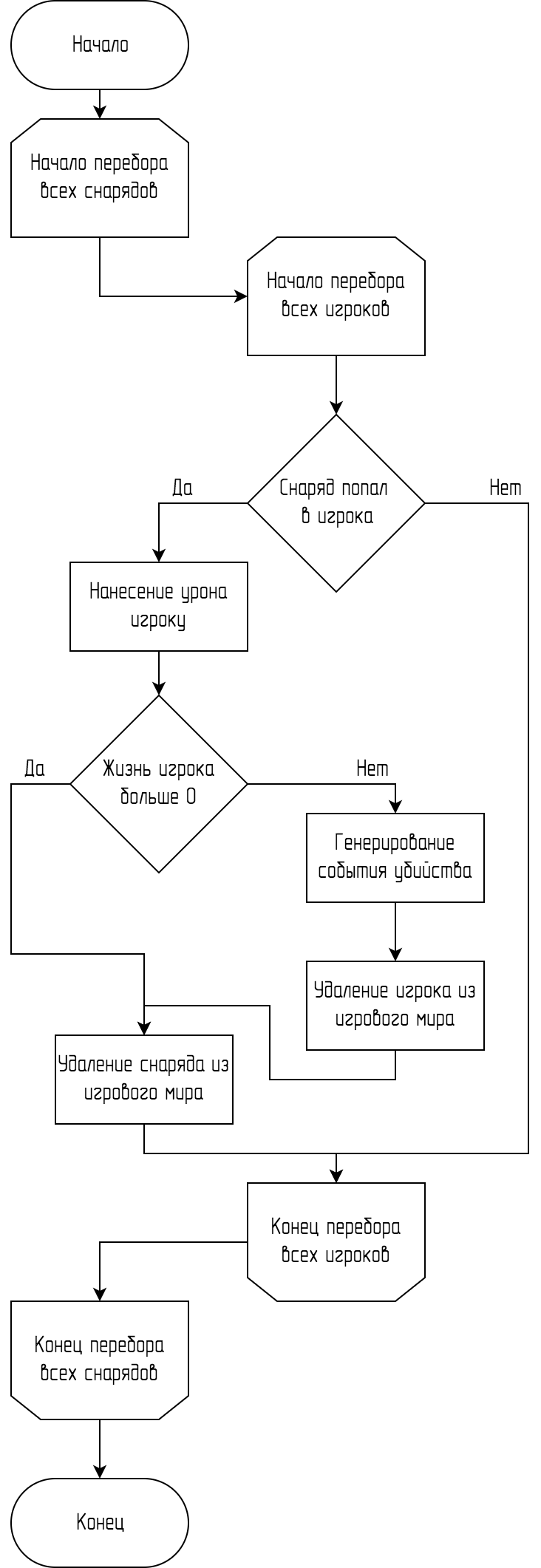


Рисунок 3.9 – Блок-схема структуры функции отрисовки статистики

# Программное обеспечение системы

## Структура программного обеспечения и функции его компонентов

При создании информационной системы в качестве операционной системы была использована операционная система MacOS

В качестве инструментального средства разработки использовалась Microsoft Visual Studio Code.

Так как, существует необходимость в том, чтобы система функционировала в основных современных браузерах, то при разработке, а также на этапах тестирования, были использованы браузеры Safari, Google Chrome.

В качестве текстового редактора для написания документации к ИС выступал Microsoft Office.

## Выбор компонентов программного обеспечения

### Операционная система

#### Рабочая станция

Ввиду ориентирования системы на Web, главным критерием выбора операционной системы является поддержка современного браузера. В качестве такого браузера может служить Google Chrome или ему подобные. А в качестве операционной системы могут быть семейства Microsoft Windows или Unix.

#### Сервер

Согласно пункту 1.5.3 технического задания установлены требования необходимые для корректной работы. Система разработана так, что выбор любой современной операционной системы для сервера не влияет на функционирование и не накладывает никак ограничений. Примером таких операционных систем могут быть macOS, GNU/Linux, Solaris, FreeBSD, OpenBSD, webOS и Microsoft Windows.

### Инструментальное средство разработки и язык программирования

Исходя из пункта 1.5.3 технического задания, система не имеет требований для разработки информационной системы. И может осуществляться любым текстовым редактором.

Язык программирование был выбран JavaScript, так как это основной язык программирования, который могут воспринимать браузеры.

Для разработки сервера использовал Node.js и библиотеки Socket.IO, P2 Node или Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера.

Socket.IO – JavaScript-библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для node.js. Оба компонента имеют похожее API. Подобно node.js, Socket.IO событийно-ориентированная.

Socket.IO главным образом использует протокол WebSocket, но если нужно, использует другие методы, например Adobe Flash сокеты, JSONP запросы или AJAX запросы, предоставляя тот же самый интерфейс. Помимо того, что Socket.IO может быть использована, как оболочка для WebSocket, она содержит много других функций, включая вещание на несколько сокетов, хранение данных, связанных с каждым клиентом, и асинхронный ввод/вывод.

P2 – это 2D движок физики твердого тела, написанный на JavaScript. Включает обнаружение столкновения, контакты, трение, реституцию, двигатели, пружины, расширенные ограничения и различные типы фигур.

### Вспомогательное программное обеспечение

В силу некоторых ограничений согласно пункту 1.5.3 технического задания в качестве интернет браузера для разработки следуют использовать три браузера, а именно: Google Chrome, Safari, Mozilla Firefox.

В качестве пакета офисных программ, которые должны включать в себя программное обеспечение для работы с текстовыми документами, электронными таблицами, а также позволяющим создавать схемы, отображающие алгоритмы действий, рассматривались такие пакеты программ как: Microsoft Office и Open Office. В виду больших возможностей Microsoft Office был выбран именно этот вариант.

Для ускорения цикла разработки рассматривались инструменты для сборки веб-приложений, призванными автоматизировать повторяющиеся процессы вроде конкатенации (склеивания) файлов, сжатия картинок, таблиц стилей и файлов JavaScript. Под эти критерия подходили два инструмента Gulp и Grunt. Был выбран Gulp, так как отличается тем от Grunt, что код задач записывается JavaScript кодом, а не в стиле конфигурационного файла. Что позволяло более гибко настроить под данный проект.

## Разработка прикладного программного обеспечения

Информационная система «Онлайн игра PixelTank». Состоит из трёх подсистем: подсистема «Сервер», подсистема «Клиент», подсистема «Искусственные пользователи».

### Структура прикладного программного обеспечения

На рисунке 4.1 приведена структура разработанного программного продукта.

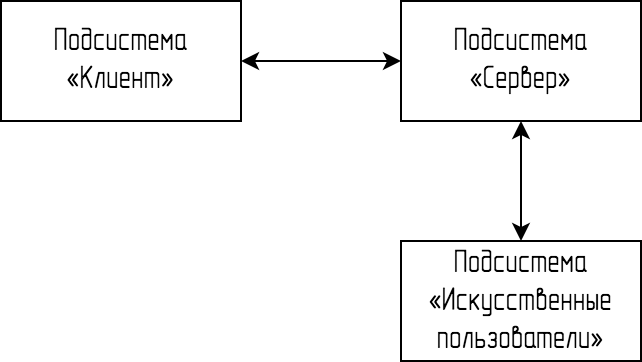


Рисунок 4.1 – Блок-схема структуры разработанного программного продукта

### Подсистема «Сервер»

На рисунке 4.2 приведена структура подсистемы «Сервер».

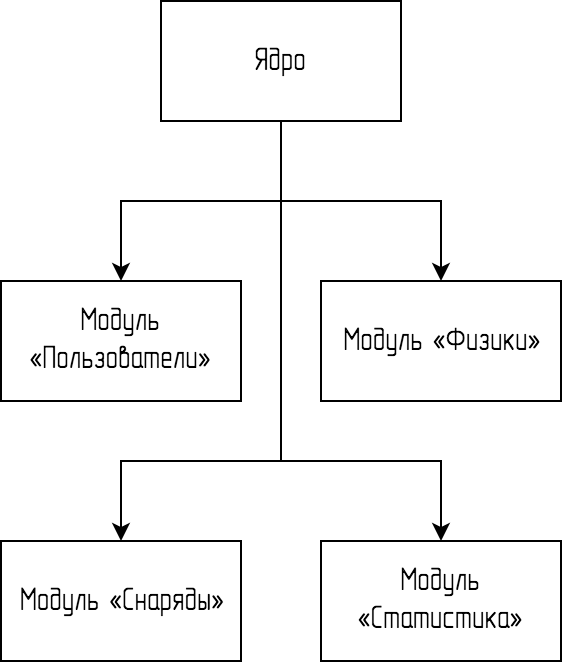


Рисунок 4.2 – Блок-схема структуры подсистемы «Сервер»

В подсистеме «Сервер» ядро отвечает за соединение клиента и сервера, а также поддержку соединения в течении всей игровой сессии с помощью библиотеки Socket.IO. Содержит в себе данные обо всех игровых объектах. Осуществляет обновление игрового цикла с периодичность (по умолчанию) 60 раз в секунду и рассылку обновлённых данных игрокам с периодичность (по умолчанию) 21 раз в секунду. В одном игровом цикле происходит полный просчет всех возможных игровых ситуаций (смерть, убийство, столкновение, попадание в игрока, выход и вход пользователя). Объём в строках 283 строки.

#### Прототип Bullet

Прототип Bullet предназначен для обработки игрового объекта снаряда.

Обрабатывает перемещение снарядов по игровому пространству, а также добавление и удаление снарядов из игрового пространства. Объём в строках 35 строк. Спецификация прототипа представлена в таблице 4.1.

Таблица 4.1 – Спецификация прототипа Bullet

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | add | Добавление снаряда в игровое пространство |
| 2 | move | Перемещение снарядов по игровому пространству |
| 3 | remove | Удаление снарядов из игрового пространства |
| 4 | getList | Вывод списка снарядов |

#### Прототип User

Прототип User предназначен для обработки пользователя как игрового объект, а также его свойства и атрибуты.

Обрабатывает перемещение игрока по игровому пространству, а также добавление и удаление игроков из игрового пространства. Объём в строках 206 строк. Спецификация прототипа представлена в таблице 4.2.

Таблица 4.2 – Спецификация прототипа User

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | add | Добавление снаряда в игровое пространство |
| 2 | move | Перемещение снарядов по игровому пространству |
| 3 | fire | Осуществляет работу со стрельбой конкретного игрока. |
| 4 | physics | Представляет физическое представление игрока |
| 5 | remove | Удаление снарядов из игрового пространства |
| 6 | getList | Вывод списка снарядов |

#### Прототип Stats

Прототип Stats предназначен для ведения статистики игрового процесса игроков.

Обрабатывает сведения об убитых и умерших игроках. Объём в строках 35 строк. Спецификация прототипа представлена в таблице 4.3.

Таблица 4.3 – Спецификация прототипа Stats

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | add | Добавление статистики для нового игрока |
| 2 | update | Обновление статистики игрока |
| 3 | remove | Удаление статистики выбранного игрока |
| 4 | get | Вывод статистики |

#### Прототип Physics

Прототип Physics предназначен для обработки и работы с физическим движком P2.

Обрабатывает сведения об игровых объектах в игровом пространстве. Объём в строках 35 строк. Спецификация прототипа представлена в таблице 4.4.

Таблица 4.4 – Спецификация прототипа Physics

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | addTank | Добавляет физическую модель с танком игрока |
| 2 | addBorder | Добавляет игровые границы пространства |
| 3 | updateTankSpeed | Удаление статистики выбранного игрока |
| 4 | updateTankAngle | Вывод статистики |
| 5 | getTank | Возвращает сведения о физической модели танка игрока |
| 6 | removeTank | Удаляет физическую модель игрока |
| 7 | worldStep | Запускает расчет физики |

### Подсистема «Клиент»

На рисунке 4.3 приведена структура подсистемы «Клиент».

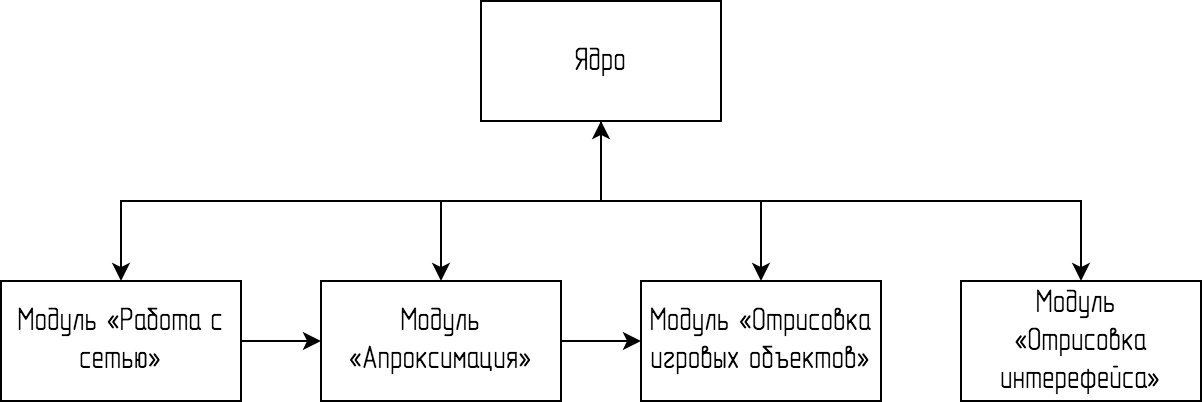


Рисунок 4.3 – Блок-схема структуры подсистемы «Клиент»

Ядро представляет собой набор слушателей событий, глобальный объект об игровых данных, а также игровой цикл.

#### Прототип Loader

Прототип Loader представляет собой загрузчик необходимых компонентов ИС для успешной работы клиента. Является самовызывающиеся функцией JavaScript.

Объём в строках 102 строк.

#### Прототип Worker

Прототип Worker представляет собой модуль «Работа с сетью» работает по технологии Web Workers. Web Workers это механизм, который позволяет скрипту выполняться в фоновом потоке, который отделен от основного потока веб-приложения. Преимущество заключается в том, что кропотливая обработка может выполняться в отдельном потоке, позволяя запустить основной (обычно пользовательский) поток без блокировки и замедления. Осуществляет соединение с сервером и отвечает за маршрутизацию данных между клиентом и сервером.

Объём в строках 116 строк.

#### Прототип Approximation

Прототип Approximation представляет собой модуль «Аппроксимация».

Обрабатывает значение на будущие кадры отрисовки, так как сервер вещает 21 раз в секунду необходимо произвести экстраполирование для 60 кадров в секунду. То есть примерно 3 кадра для обеспечения плавности игры. Для уменьшения нагрузки на клиент было принято решения не рассчитывать физику на клиенте. Из-за чего обеспечивается небольшая точность в аппроксимации данных.

Объём в строках 83 строки.

Спецификация прототипа представлена в таблице 4.5.

Таблица 4.5 – Спецификация прототипа Approximation

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | Approximation | Выполняет аппроксимацию всех объектов игрового мира |
| 2 | RotateTurret | Выполняет расчет поворота орудия с заданной скоростью поворота |

#### Прототип Game

Прототип Game осуществляет отрисовку всех игровых объектов (танки, снаряды). Так как данный прототип производит большую нагрузку систему пользователя были приняты меры для снижения нагрузки. В виде ограничения на количество видимых элементов.

Объём в строках 143 строк.

Спецификация прототипа представлена в таблице 4.6.

Таблица 4.6 – Спецификация прототипа Game

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | Camera | Задает видимую игроку область |
| 2 | IsVisible | Выполняет расчет поворота орудия с заданной скоростью поворота |
| 3 | DrawTanks | Выполняет отрисовку танков |
| 4 | DrawBullets | Выполняет отрисовку снарядов |

#### Прототип UI

Прототип UI осуществляет отрисовку интерфейса (радар, статистика, лог событий).

Объём в строках 136 строк.

Спецификация прототипа представлена в таблице 4.7.

Таблица 4.7 – Спецификация прототипа UI

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | Radar | Выполняет отрисовку радара, а также выборку игроков для отображения меток. |
| 2 | DrawLog | Выполняет отрисовку игровых событий |
| 3 | DrawStat | Выполняет отрисовку статистики, а также выборку лучших игроков. |
| 4 | sDecrease | Подготовливает данные об игроках в порядке уменьшения результативности |

#### Прототип Background

Прототип Background осуществляет отрисовку фона.

Объём в строках 18 строк.

Спецификация прототипа представлена в таблице 4.8.

Таблица 4.8 – Спецификация прототипа Game

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | Background | Выполняет отрисовку игрового фона |

#### Обработчики событий

Используя обработчики событий было реализовано захват событий о действиях игрока.

Объём в строках 98 строк.

Спецификация обработчиков событий представлена в таблице 4.9.

Таблица 4.9 – Спецификация обработчиков событий

|  |  |  |
| --- | --- | --- |
| **№** | **Название** | **Описание** |
| Методы | | |
| 1 | onmousemove | Определяет положение мыши |
| 2 | onmousedown | Опеределяет событие начала стрельбы |
| 3 | onmouseup | Определяет событие окончание стрельбы |
| 4 | onresize | Определяет размер видимой области игры при изменение размеров окна |
| 5 | onkeydown | Определяет событие начала нажатия клавишь |
| 6 | onkeyup | Определяет событие окончания нажатия клавишь |
| 7 | onclick | Определяет событие старта игры |
| 8 | onmessage | Определяет событие получение данных с сервера для актуализации данных |

### Подсистема «Искусственные пользователи»

На рисунке 4.4 приведена структура подсистемы «Искусственные пользователи».



Рисунок 4.4 – Блок-схема структуры подсистемы «Искусственные пользователи»

Модуль «Центр управления» осуществляет запуск заданного количества модулей «Искусственных пользователей» в отдельных процессах. В случае ошибки или иной причины завершения процесса потомков, осуществляет перезапуск нужной единицы. Каждый «Искусственный пользователь» осуществляет работу каждый игровой цикл по одному алгоритму, которая приведена на рисунке 4.5. Объём в строках 165 строк.

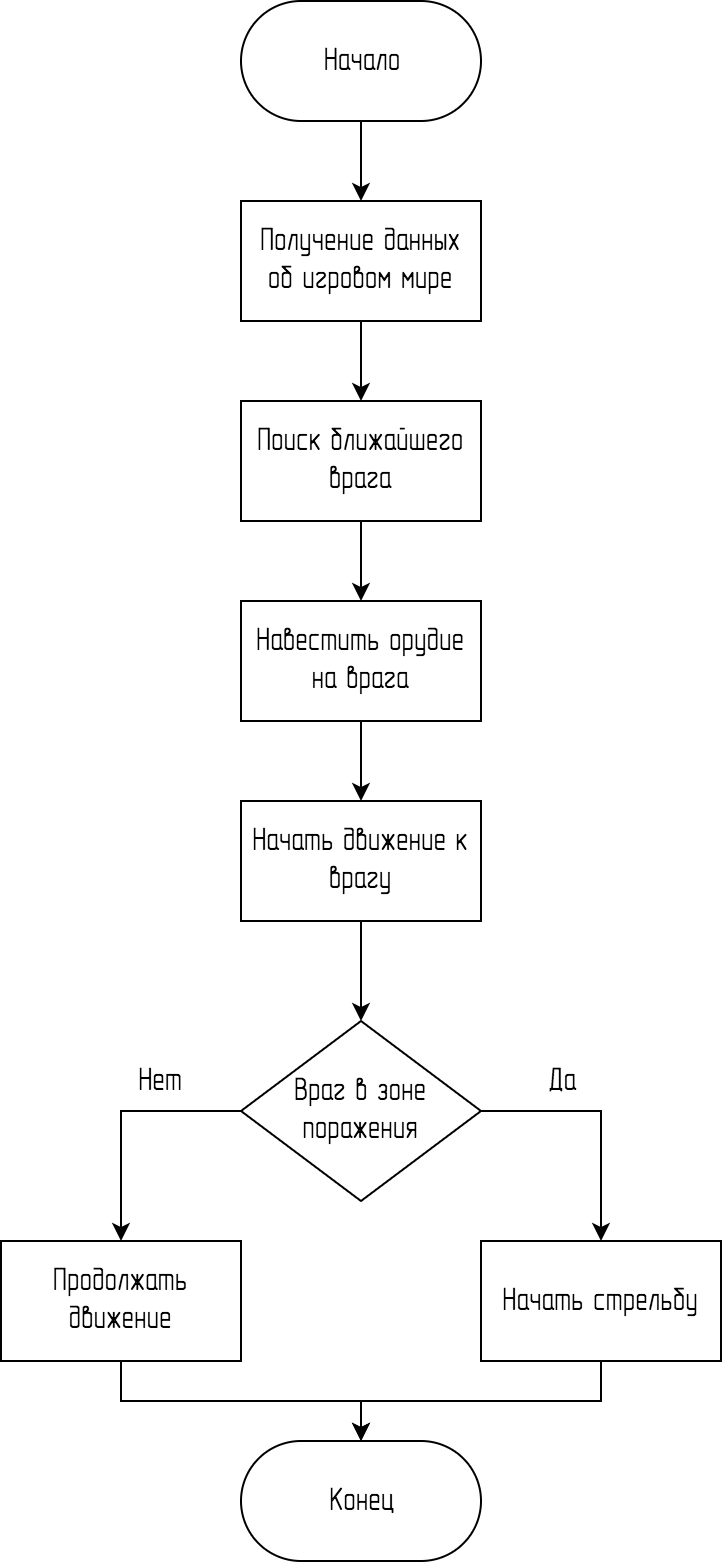


Рисунок 4.5 – Алгоритм «Искусственного пользователя»

## Особенности реализации, эксплуатации и сопровождения системы

## Руководство пользователя

### Требования к условиям эксплуатации

Для корректной работы системы необходим компьютер, характеристики которого не ниже представленной конфигурации:

* процессор с минимальной тактовой частотой 1.6 Ghz;
* оперативная память объёмом 4 Гб;
* жёсткий диск свободным объёмом 10 Gb;
* видеоадаптер.

### Инсталляция и настройка

Работа с информационной системой со стороны пользователя полностью обеспечивается средствами одного из браузеров Google Chrome, Safari.

### Порядок и особенности работы

Общий функционал информационной системы представление развлекающей функционал с игровой составляющей.

Описание игрового интерфейса для игрока.

Заходя в игру, пользователь видит стартовую страницу, представленное на рисунке 4.6. На странице также есть пояснение о управлении в ИС. Где клавиши WASD – используются для перемещения, мышь – для наведения и стрельбы, а Q – для отображения статистики.

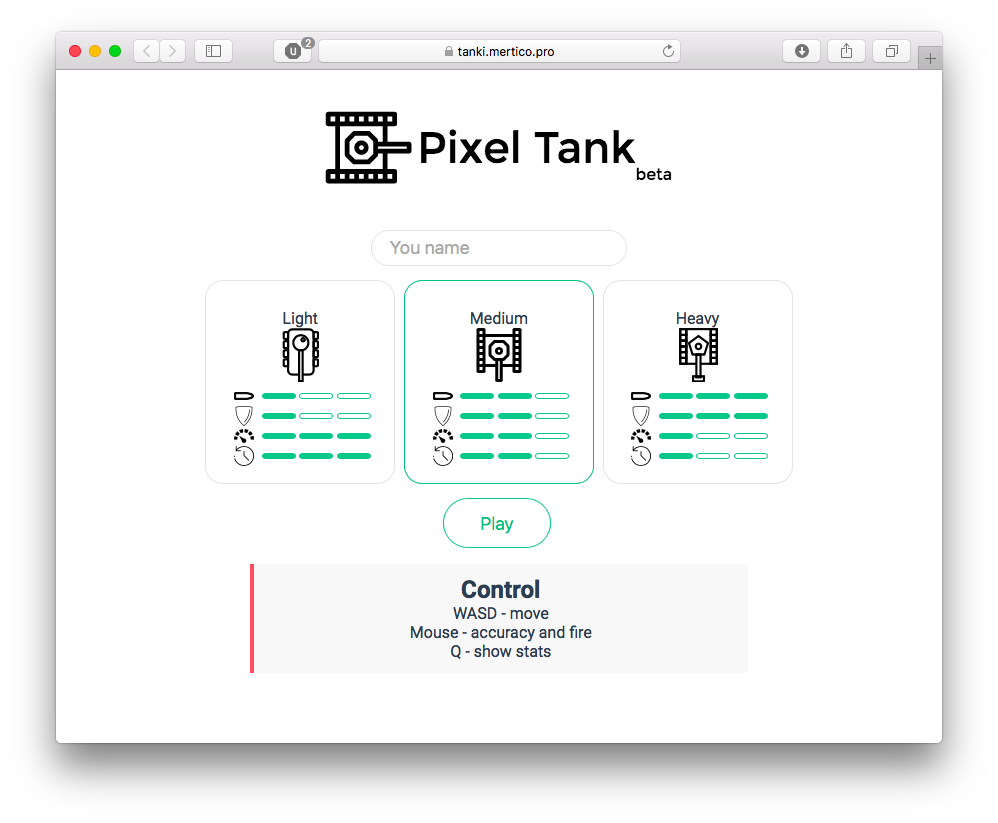


Рисунок 4.6 – Интерфейс стартового меню

В поле вода «You name» можно ввести имя, под которым будет играть пользователь. Нажимая на иконки можно выбрать любой из трех танков. И нажимая на кнопку «Play» пользователь начинает игру.

После начала игры пользователь видит интерфейс игры рисунок 4.7.

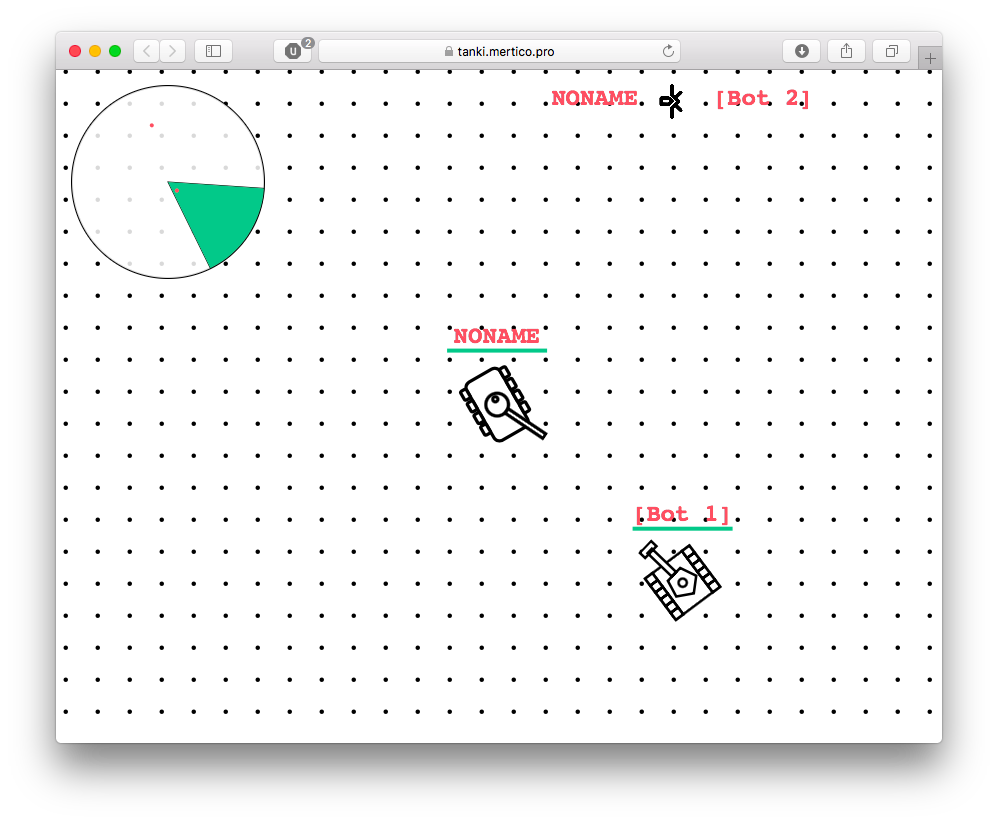


Рисунок 4.7 – Интерфейс игры

Над танком указано имя пользователя и количество здоровья, как только здоровье кончится, игрок возрождается в случайном месте на случайном танке. Радар, на котором отображаются другие пользователи. История пяти последних событий c информацией об уничтоженных танках.

Нажимая клавишу «Q» в игре можно увидеть текущую статистику рисунок 4.8.

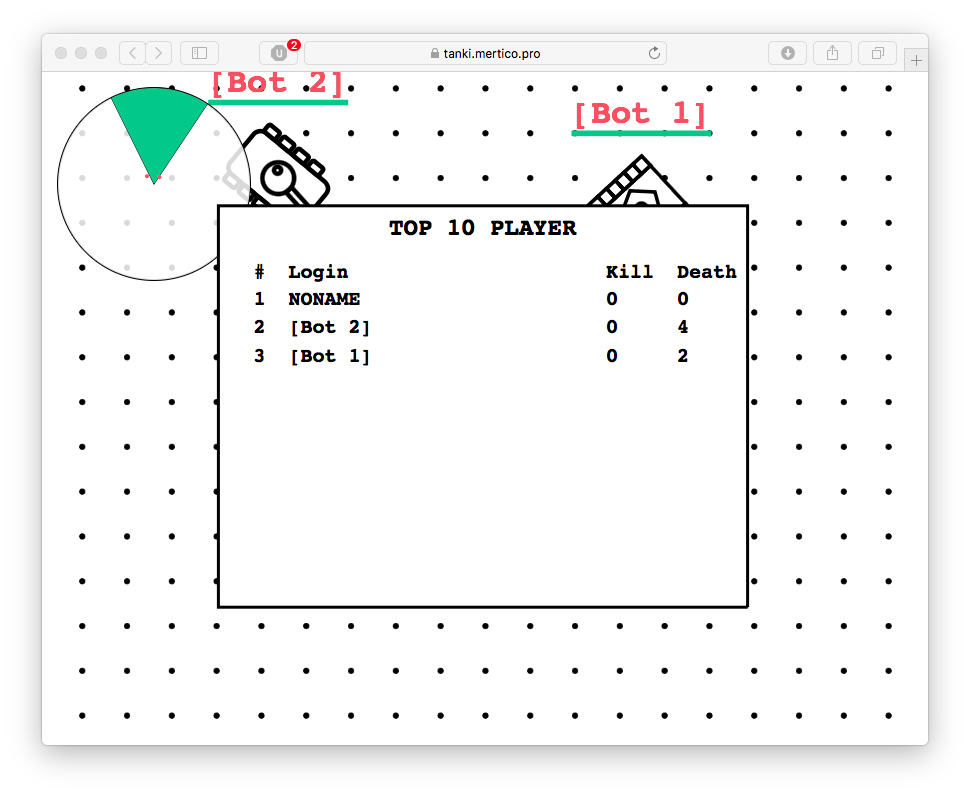


Рисунок 4.8 – Статистика

# Тестирование системы

Тестирование программного продукта – это процесс исследования, испытания системы которое позволяет выявить ошибки, недоработки

## Условия и порядок тестирования

Объектом тестирования будет являться информационная система. Тестирование будет производится по методу «Чёрный ящик» что подразумевает под собой тестирование объекта с точки зрения внешнего мира (пользователя), при котором тестировщик имеет доступ к информационной системе только через те интерфейсы, которые будут доступны пользователям системы, не используя знания о её внутреннем устройстве.

Тестирование будет производится на локальной машине с использованием веб сервера. Для непосредственного просмотра клиентской части программы будут использованы поочерёдно web браузеры Safari, Google Chrome.

Входе тестирования будут проверены такие параметры как корректность отображения внешнего вида системы, корректность работы основных функций приложения, совместимость с браузерами, указанными в пункте 1.5.3 технического задания, а также правильной работы системы в случае ввода некорректных данных. Одним из немаловажных этапов тестирования является проверка совместимости со всеми браузерами, указанными в требованиях пункта 1.5.3 технического задания.

## Исходные данные для контрольных примеров

В ходе проведения тестирования была рассмотрена корректность работы функции выбора танка и имени игрока, форма которой представлена на рисунке 5.1 при тестировании которой моделировались ситуации: выбора пустой строки имени, выбор произвольного имени.

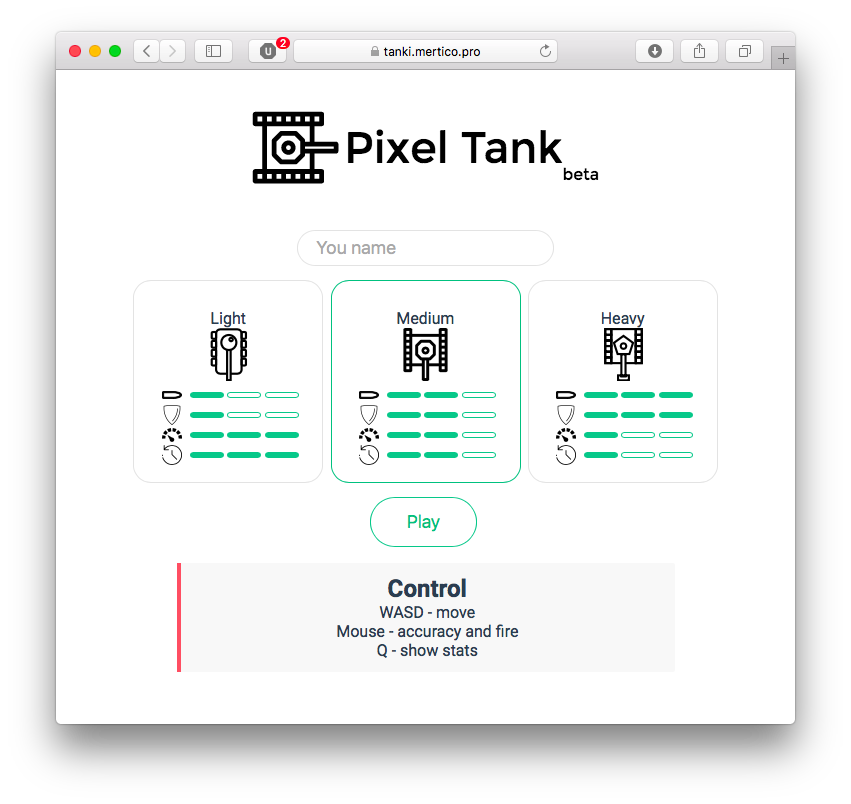


Рисунок 5.1 – Форма входа в игру

Также было произведено тестирования управления. На рисунке 5.2 и рисунке 5.3 показан поворот башни.

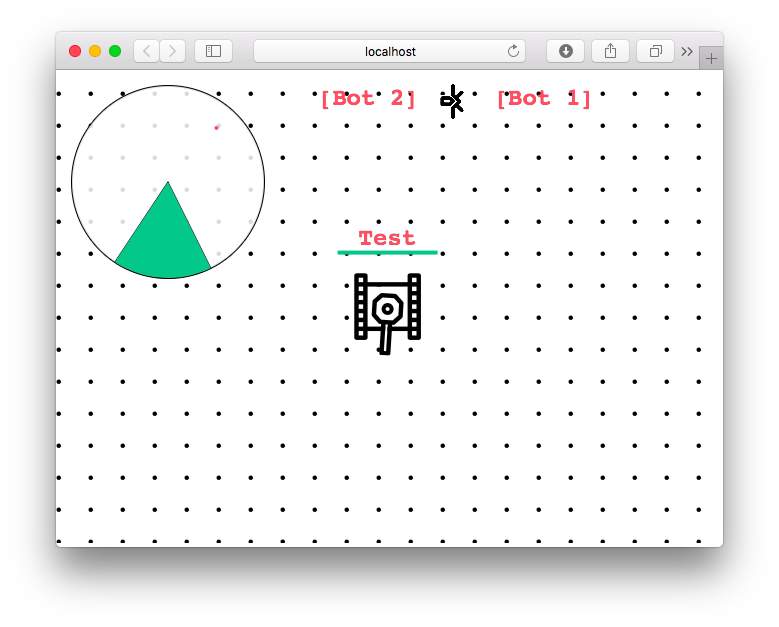


Рисунок 5.2 – Поворот башни вниз

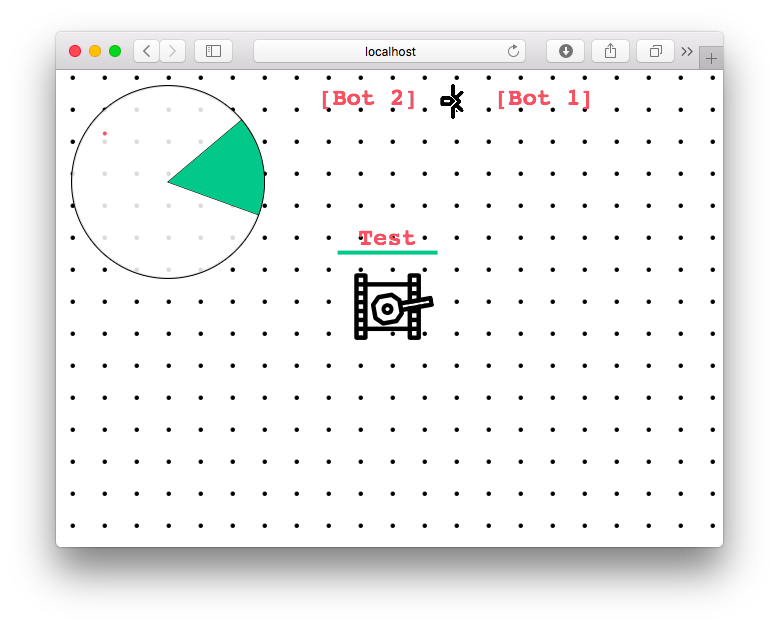


Рисунок 5.3 – Поворот башни вправо

На рисунке 5.4 отображена стрельба

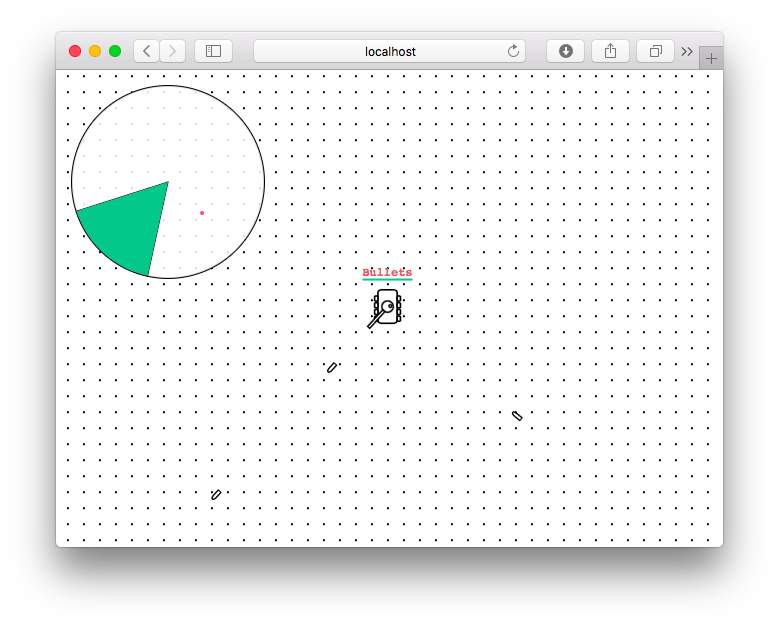


Рисунок 5.4 – Стрельба

На рисунке 5.5 отображен поворот танка

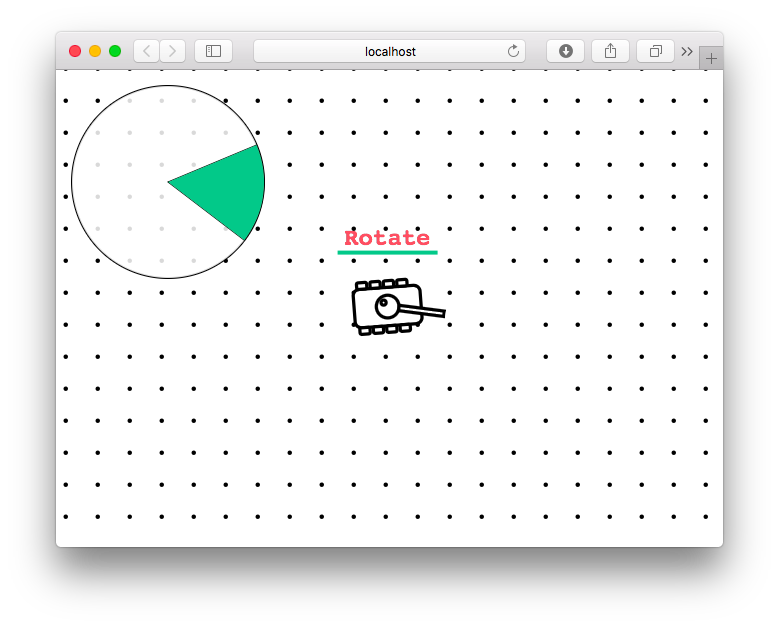


Рисунок 5.5 – Поворот танка

## Результаты тестирования

В ходе проверки на корректность работы основных функций были проведены тесты, имитирующие игру. Тестировалась корректность игры по сети интернет.

Тестирование показало, что основной функционал системы реализован и работает исправно.

# Экономический раздел

Основными критериями оценки являются значения себестоимости прибыли и цены изделия.

Для расчёта полной себестоимости необходимо подсчитать такие статьи как:

* расчёт заработной платы;
* расчёт электроэнергии, затраченной при производстве;
* затраты на канцелярские товары;
* услуги связи.

## Расчёт трудозатрат на разработку программного продукта

В настоящее время для определения трудоёмкости разработки информационных приложений применяется способ оценки работ в человеко-часах. Величина параметра трудоёмкости для разрабатываемого программного решения состоит из суммы значений трудоёмкости для каждого этапа разработки Данные расчётов ожидаемой длительности работ приведены в таблице 6.1.

Таблица 6.1 – Длительность работ на этапе проектирования

|  |  |  |
| --- | --- | --- |
| **Этап** | **Содержание работы** | **Трудозатраты чел/ч.** |
| **1** | **2** | **3** |
| Разработка технических требований | Постановка задачи | 25 |
| Уточнение требований к системе | 20 |
| Разработка технических требований | Разработка технического задания | 40 |

Продолжение таблицы 6.1

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | |
| Разработка программных модулей | Разработка серверной части | 80 | |
| Разработка клиентской части | 60 | |
| Разработка интерфейса программы и форм для ввода информации | 10 | |
| Отладка и тестирование | 20 | |
| Описание программы | Написание документации | 30 | |
| Итого трудоёмкость выполнения системы | | 285 |

В таблице 6.2 представлены значения основной заработной платы работников, участвующих в разработке продукта.

Таблица 6.2 – Расчёт основной заработной платы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Категория работника** | **Квалификация** | **Трудоемкость разработки,**  **чел.-ч** | **Часовая ставка, руб.-ч.** | **Сумма, руб.** |
| Разработчик системы | Программист | 285 | 350 | 99 750 |
| ИТОГО затраты на оплату труда | | | | 99 750 |

Используя показатель затраты на оплату труда представленные в таблице 6.2, вычислим размер заработной платы инженера, учитывая страховые взносы.

Страховые взносы состоят из отчислений в:

* Пенсионный фонд – 22%;
* Фонд социального страхования – 2,9%;
* Федеральный фонд обязательного медицинского страхования – 5,1%;
* Обязательное социальное страхование от несчастных случаев – 0,2%.

Представлен расчёт коэффициента страховых взносов

Кстр = 22% + 2,9% + 5,1% + 0,2% = 30,2%

Представлен расчёт значения страховых отчислений относительно значений основной заработной платы.

ЗПстр = ЗП × Кстр

ЗПстр = (99 750 × 30,2) / 100 = 30 124,50 (руб.)

Рассчитаем заработную плату с учетом начислений:

ЗПобщ = ЗП + ЗПстр

ЗПобщ = 99 750 + 30 124,50 = 129 874,50 (руб.)

В таблице 6.3 представлены значения мощности и количества использованного при разработке программного продукта оборудования.

Таблица 6.3 – Расчёт затрат на электроэнергию

|  |  |  |
| --- | --- | --- |
| **Наименование оборудования** | **Количество** | **Мощность единицы, кВт** |
| Ноутбук | 1 | 0.045 |
| Принтер | 1 | 0.1 |
| Общая потребляемая мощность | | 0.145 |

Рассчитаем затраты на электроэнергию:

Сэл = Р ×Т ×Z,

где Р – общая мощность;

Т – время;

Z – стоимость 1 кВт/ч.

Сэл = 0,145 ×225 ×2,49 = 102,89 (руб.)

Одноставочный тариф на электроэнергию на свет для населения, проживающего в Ульяновске и городских населенных пунктах Ульяновской области в домах, оборудованных в установленном порядке электрическими плитами и (или) электроотопительными приборами составляет (на 30.06.2018): 2.49 рублей за 1 кВт.ч.

В таблице 6.4 представлены значения материальных затрат, которые состоят из канцелярских товаров.

Таблица 6.4 − Материальные затраты

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Количество** | **Цена за шт. руб.** | **Общая стоимость, руб.** |
| Бумага | 1 | 350 | 350 |
| Ручка | 6 | 3 | 18 |
| ИТОГО | | | 368 |

В таблице 6.5 представлены данные о прочих затратах, сопряжённых с разработкой проекта.

В числе прочих затрат включено значение затрат за использование услуг связи (интернета), получим значение в расчёте на месяц за услуги связи. Количества рабочих часов, затраченных на проект 285, примем значение рабочего дня равное 8 часам, исходя из числа рабочих дней в месяце 21 получим:

285 / 8 / 21 = 1,69

Приравняем полученное значение к 2 месяцам.

Таблица 6.5 − Прочие затраты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование** | **Единица измерения** | **Количество** | **Цена за единицу руб.** | **Стоимость, руб.** |
| Услуги связи (Интернет) | месяц | 2 | 350 | 700 |
| Заправка картриджа принтера | шт. | 1 | 600 | 600 |
| Аренда сервера | месяц | 2 | 500 | 1000 |
| ИТОГО | | | | 2300 |

## Амортизационные отчисления

Амортизационные отчисления – отчисления части стоимости основных фондов для возмещения их износа. Амортизационные отчисления включаются в издержки производства.

В процессе разработки программного продукта использовалось оборудование: ноутбук, принтер их цена указана в таблице 6.6.

Таблица 6.6 – Цена оборудования

|  |  |  |
| --- | --- | --- |
| **Наименование оборудования** | **Количество** | **Цена** |
| Ноутбук | 1 | 72 000 |
| Принтер | 1 | 5 000 |
| ИТОГО | | 77 000 |

Общая сумма амортизационных отчислений определяется:

ЗАМ = (Ф × НА × Т) / 100 × ТЭФ,

где ЗАМ – общая сумма амортизационных отчислений;

Ф – стоимость оборудования;

НА – годовая норма амортизационного оборудования;

Т – время работы оборудования за период разработки;

ТЭФ – эффективный фонд времени работы оборудования.

ЗАМ = (77 000 × 10 × 332) / (100 × 2040) = 1 253,13 (руб.)

Общая сумма дополнительных затрат будет складываться из материальных, прочих затрат, затрат на амортизационные отчисления и будет равна значению 3 921,13 руб. Исходя из данных, полученных на предыдущих этапах, рассчитаем базовую себестоимость:

Сп.о. = ЗПобщ + Сэл + Робщ,

где ЗПобщ – расходы на заработную плату;

Сэл – расходы на электричество;

Робщ – прочие затраты.

Сп.о. = 129 874,50 + 102,89 + 3 921,13 = 133 898,52 (руб.)

В таблице 6.7 приведены суммарные затраты на проект

Таблица 6.7 – Суммарные затраты на проект

|  |  |
| --- | --- |
| **Наименование показателя** | **Базовый вариант** |
| Основная заработная плата, руб. | 99 750,00 |
| Отчисления, руб. | 30 124,50 |
| Затраты на электроэнергию, руб. | 102,89 |
| Материальные затраты, руб. | 368,00 |
| Прочие затраты, руб. | 3 921,13 |
| ИТОГО | 133 898,52 |

## Экономическая эффективность

Экономический эффект – это величина, характеризующая достигнутые благодаря созданию или совершенствованию ПО дополнительные экономические результаты. Экономическая эффективность – результативность экономической деятельности, экономических программ и мероприятий, характеризуемая отношением полученного экономического эффекта к затратам факторов, обусловившим получение этого результата.

Экономический эффект определяется как разность между годовой экономией (или годовым приростом) и нормативной прибылью.

Так как ИС разрабатывалась с целью принесения прибыли с помощью монетизации. Это добавление рекламы после проигрыша и возможность отключать рекламу.

## Расчет плановой прибыли

От того, насколько достоверна определена плановая прибыль, будет зависеть успешная финансово-хозяйственная деятельность предприятия. Расчет плановой прибыли должен быть экономически обоснованным. Это позволит осуществлять своевременное и полное финансирование инвестиций, прироста собственных оборотных средств и соответствующих выплат сотрудникам.

Плановая прибыль реализации программного решения:

П = Сп.о.× ( 1+ Рн / 100),

где Сп.о. – полная себестоимость, руб.;

Рн– норматив рентабельности.

При нормативе рентабельности, равном 30%, прибыль будет составлять 174 068,08 руб.:

П = 133 898,52 × (1 + 30/100) = 174 068,08 (руб.)

С учетом налога на прибыль, составляющим 20%, доход составит:

П = 174 068,08 – 0,2 × 174 068,08 = 139 254,46 (руб.)

Посчитать среднюю стоимость поддержки web-приложения не сложно – 30 000 руб./год.

Допустим, что доход с показов рекламы от Google Adwords за тысячу показов в видимой области экрана всегда будет равняться 35 руб.

Так как реклама будет показываться после смерти и будет считаться отдельным показом. То при одновременной игре в среднем 80 человек будет примерно 30 смертей в минуту, что будет равнять 1 512 рублей в день

139254,46 / (0,035×30×1440) = 92,09 (дней)

Учитывая стоимость разработки, программное решение покажет свою

экономическую выгоду уже после 3 месяцев эксплуатации системы.

# Заключение

Выпускная квалификационная работа была выполнена в необходимом и достаточном объёме.

В рамках выпускной квалификационной работы были выполнены все требования, изложенные в техническом задании

Были использованы средства информационного, функционального моделирования, интегрированная среда разработки и иное программное обеспечение.

Дальнейшее развитие ИС предусматривает возможность реализации горизонтального масштабирования использую адаптер для Socket.IO. Адаптер redis расширяет базовый адаптер, позволяя обрабатывать большее количество клиентов.

# Список использованых источников

1. Альтернативные браузерные технологии [Электронный ресурс] / Javascript.ru – 2017. – Режим доступа: https://learn.javascript.ru/intro#альтернативные-браузерные-технологии
2. p2 – npm [Электронный ресурс] / Режим доступа: https://www.npmjs.com/package/p2
3. gulp – npm [Электронный ресурс] / Режим доступа: https://www.npmjs.com/package/gulp
4. uws – npm [Электронный ресурс] / Режим доступа: https://www.npmjs.com/package/uws
5. socket.io – npm [Электронный ресурс] / Режим доступа: https://www.npmjs.com/package/socket.io
6. socket.io-client – npm [Электронный ресурс] / Режим доступа: https://www.npmjs.com/package/socket.io-client
7. p2 docs [Электронный ресурс] / Режим доступа: http://schteppe.github.io/p2.js/docs/
8. Optimizing WebSockets Bandwidth [Электронный ресурс] / Режим доступа: http://buildnewgames.com/optimizing-websockets-bandwidth
9. Node.js v7.7.2 Documentation [Электронный ресурс] / Режим доступа: https://nodejs.org/dist/latest-v7.x/docs/api/
10. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. –Питер. – 2016. – 336 с.
11. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.– М. : Стандартинформ, 2010. – 8 c.
12. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М. : Стандартинформ, 1996. – 9 c.
13. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания. – М. : Изд-во стандартов, 1997. – 5 c.
14. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. М. : Стандартинформ, 2010. – 166 с.
15. Родионов, В. В. Выполнение выпускной квалификационной работы бакалавра: учебно-методическое пособие для студентов направления 09.03.02 «Информационные системы и технологии» / В. В. Родионов. – Ульяновск: УлГТУ, 2017. – 74 с.

# Приложение А

(обязательное)

**Текст программы**

Программный код подсистемы «Сервер»

Файл config.js

'use strict';

var config = {

FPS: 60, // Частота обновлений в секунду

FPS\_send: 21, // Частота отправки обновлений в секунду

BULLET\_SPEED: 900,

tank: {

light: {

hp: 180, // текущее хп

max\_speed: 400, // Макс. скорость

acceleration: 185, // Скорость разгона (px/sec)

reload: 0.35, // Период между выстрелами (в секундах)

damage: 55, // Урон наносимый снарядом

chassis: {

angle: 96 // Скорость поворота шасси (в градусах)

},

turret: {

angle: 180 // Скорость башни шасси (в градусах)

}

},

medium: {

hp: 275,

max\_speed: 310,

acceleration: 160,

reload: 0.68,

damage: 85,

chassis: {

angle: 68

},

turret: {

angle: 150

}

},

heavy: {

hp: 640,

max\_speed: 235,

acceleration: 110,

reload: 1.12,

damage: 125,

chassis: {

angle: 48

},

turret: {

angle: 120

}

}

}

};

module.exports = config;

module.exports.getTank = function(tank) {

return config.tank[tank];

}

Файл core.js

'use strict';

var Config = require('./config');

var Cooldown = require('./modules/cooldown');

var Death = require('./modules/death');

var Stats = require('./modules/stats');

var Physics = require('./modules/physics');

var Control = require('./modules/control');

var Bullet = require('./modules/bullet');

var User = require('./modules/user');

var Core = require('./core');

// Границы карты

Physics.addBorder( 0, 2530,Math.PI); // bottom

Physics.addBorder(-2560, 0,-Math.PI/2); // left

Physics.addBorder( 2530, 0,Math.PI/2); // right

Physics.addBorder( 0,-2560,0); // top

var LastRefTime = process.hrtime();

setInterval(function Refresh() {

// Расчет разницы во времени между кадрами (В секундах)

let NowRefTime = process.hrtime();

let delta = ((NowRefTime[0]-LastRefTime[0]) + (NowRefTime[1]-LastRefTime[1])/1000000000);

LastRefTime=NowRefTime;

// Полет снарядов

Bullet.move(delta);

// Полчуние информации об игроках и снарядов

var bullets = Bullet.getList();

var users = User.getList();

// Просчет столкновения со снарядом

for(var bi = 0; bi < bullets.length; bi++) {

for(var ui = 0, ul = users.length; ui < ul; ui++) {

if (users[ui].id != bullets[bi].user)

if(Math.pow(Math.abs(bullets[bi].x-users[ui].x),2)+Math.pow(Math.abs(bullets[bi].y-users[ui].y),2) <= 56\*56) {

if(CollisionBullet(users[ui].x,users[ui].y,users[ui].chassis.angle,bullets[bi].x,bullets[bi].y)) {

//console.log(bullets[bi].user + ' повредил ' + users[ui].id + ' на ' + bullets[bi].damage + ' ('+ users[ui].hp + '/150)');

if (users[ui].hp-bullets[bi].damage >= 0 && users[ui].hp > 0) {

users[ui].hp-=bullets[bi].damage;

} else {

//console.log(bullets[bi].user + ' уничтожил ' + users[ui].id);

// Отправка данных о смерти

Stats.update(bullets[bi].user, users[ui].id);

Death.add(bullets[bi].user,users[ui].id);

let plname = users[ui].name;

let usersid = users[ui].id;

User.remove(ui);

// Возрождение, но нужно выделить в отдельный месседж

// в виде replay, а саму смерть оптравлять в виде death

////////////////

User.add(usersid,plname);

///////////////////

}

Bullet.remove(bi);

break;

}

}

}

}

User.move(delta);

Physics.worldStep(1/delta);

// Считывание данных с физ. мира

User.physics();

User.fire();

//counter++;

}, 1000/Config.FPS);

/\*

counter++;

var counter=0;

setInterval(function() {

console.log(counter);

counter=0;

}, 1000);

/\*\*/

function CollisionBullet(x,y,a,bx,by) {

/\*

x,y - координаты танка

a - поворот танка

bx,by - координаты снаряда

x1,y1 --- x2,y2

| |

| x,y |

| |

x3,y3 --- x4,y4

\*/

var R = Math.sqrt(Math.pow(bx-x,2)+Math.pow(by-y,2));

bx = Math.cos(a) \* R + x;

by = Math.sin(a) \* R + y;

var x1=x-40;

var x4=x+40;

var y1=y-40;

var y4=y+40;

if(bx > x1 && by > y1

&& bx < x4 && by < y4)

return true;

return false;

}

function getRandomInt(min, max) {

return Math.floor(Math.random() \* (max - min + 1)) + min;

}

Файл cooldown.js

'use strict';

var cd = [];

// Уменьшение перезарядки

setInterval(function () {

for(var i = 0; i < cd.length; i++) {

cd[i].time-=1/30;

if(cd[i].time <= 0)

cd.splice(i, 1);

}

}, 1000/30);

// Добавление перезарядка игрока по ид пользователя

exports.add = function(id, time) {

cd.unshift({

id,

time

});

}

// Вывод перезарядки

exports.get = function(id) {

for(var i = 0, l = cd.length; i < l; i++) {

if(cd[i].id == id) {

return false;

break;

}

}

return true;

}

Файл control.js

'use strict';

var control = [];

// Изменение управления игрока по ид пользователя

exports.update = function(id, data) {

control[id]={

move: data[0],

rotate: data[1],

angle: data[2],

fire: data[3]

};

}

// Добавление управления игрока по ид пользователя

exports.add = function(id) {

control[id]={

move: 0,

rotate: 0,

angle: 0,

fire: false

};

}

// Удаление управления игрока по ид пользователя

exports.remove = function(id) {

delete control[id];

}

// Вывод управления

exports.get = function(id) {

return control[id];

}

Файл user.js

'use strict';

var Config = require('../config');

var Physics = require('./physics');

var Control = require('./control');

var Bullet = require('./bullet');

var Cooldown = require('./cooldown');

var users = [];

exports.add = function(id,name,tank) {

switch(tank) {

case 'heavy':

break;

case 'medium':

break;

case 'light':

break;

default:

tank=(Math.random() > .66) ? 'heavy' : (Math.random() > .5) ? 'light' : 'medium';

break;

}

if (name == '') {

name="NONAME";

}

let x = getRandomInt(-2000, 2000);

let y = getRandomInt(-2000, 2000);

users.unshift({

x : x,

y : y,

speed : 0,

turret : {

angle: -Math.PI/2

},

chassis : {

angle: -Math.PI/2

},

hp : Config.tank[tank].hp,

tank : tank,

name : name,// заготовка для ника

id : id

});

Physics.addTank(id,tank,x,y,-Math.PI/2);

}

exports.move = function(delta) {

for(var ui = 0; ui < users.length; ui++) {

// Обновление данных о направление орудия

users[ui].turret.angle = RotateTurret(users[ui].turret.angle,Control.get(users[ui].id).angle,(delta\*Config.getTank(users[ui].tank).turret.angle\*Math.PI/180));

// Вычисление изменений скорости

if(Control.get(users[ui].id).move == 1) {

users[ui].speed += (delta\*Config.getTank(users[ui].tank).acceleration\*ac(users[ui].speed,Config.getTank(users[ui].tank).max\_speed));

if (users[ui].speed > Config.getTank(users[ui].tank).max\_speed) users[ui].speed = Config.getTank(users[ui].tank).max\_speed;

}

if(Control.get(users[ui].id).move == -1) {

users[ui].speed -= (delta\*Config.getTank(users[ui].tank).acceleration\*0.6\*ac(users[ui].speed,Config.getTank(users[ui].tank).max\_speed));

if (users[ui].speed < -Config.getTank(users[ui].tank).max\_speed\*0.3) users[ui].speed = -Config.getTank(users[ui].tank).max\_speed\*0.3;

}

if(Control.get(users[ui].id).move == 0) {

if (users[ui].speed < 10 && users[ui].speed > -10) {

users[ui].speed = 0;

}

if (users[ui].speed <= -10) {

users[ui].speed += (delta\*Config.getTank(users[ui].tank).acceleration\*3\*ac(users[ui].speed,users[ui].max\_speed));

}

if (users[ui].speed >= 10) {

users[ui].speed -= (delta\*Config.getTank(users[ui].tank).acceleration\*3\*ac(users[ui].speed,users[ui].max\_speed));

}

}

// Поворот танка

let angleTank = 0;

if(Control.get(users[ui].id).rotate == -1) angleTank -=Config.getTank(users[ui].tank).chassis.angle\*Math.PI/180\*ac(users[ui].speed,Config.getTank(users[ui].tank).max\_speed);

if(Control.get(users[ui].id).rotate == 1) angleTank +=Config.getTank(users[ui].tank).chassis.angle\*Math.PI/180\*ac(users[ui].speed,Config.getTank(users[ui].tank).max\_speed);

Physics.updateTankAngle(users[ui].id, angleTank);

// Передвижение танка на величину скорости

Physics.updateTankSpeed(users[ui].id,users[ui].speed);

}

}

exports.physics = function() {

// Считывание данных с физ. мира

for(var ui = 0, ul = users.length; ui < ul; ui++) {

let tmp = Physics.getTank(users[ui].id);

users[ui].x = tmp.position[0];

users[ui].y = tmp.position[1];

users[ui].chassis.angle = tmp.angle;

users[ui].speed = tmp.velocity[0]\*Math.cos(tmp.angle)+tmp.velocity[1]\*Math.sin(tmp.angle);

}

}

exports.fire = function() {

// Стрельба

for(var ui = 0, ul = users.length; ui < ul; ui++) {

if(Control.get(users[ui].id).fire) {

// Проверка возможности стрельбы

if(Cooldown.get(users[ui].id)) {

// Включение блокировки на стрельбу (перезарядка)

Cooldown.add(users[ui].id, Config.getTank(users[ui].tank).reload);

// Добавление снаряда

Bullet.add(users[ui].id,users[ui].x,users[ui].y,users[ui].turret.angle,Config.getTank(users[ui].tank).damage);

}

}

}

}

exports.remove = function(id) {

Physics.removeTank(users[id].id);

users.splice(id, 1);

}

exports.getList = function() {

return users;

}

// Функция плавного поворота на угол angle со скоростью speed

function RotateTurret(ta,ma,s) {

if (ta < 0)

ta = ta + 2\*Math.PI;

let a = ta - ma;

if (a > Math.PI)

a = a - 2\*Math.PI;

else if (a < -Math.PI)

a = a + 2\*Math.PI;

if (a >= -s && a <= s)

ta = ma;

else if (a > s)

ta = ta - s;

else if (a < -s)

ta = ta + s;

return ta;

}

function ac(speed, max\_speed=120) {

//(2+x^0.1)/(2+x^0.3) (x from 0 to 100)

// получаем коэффицент скорости разворота

// перемножаем на скорость поворота танка

return (1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.1))/(1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.3));

}

function getRandomInt(min, max) {

return Math.floor(Math.random() \* (max - min + 1)) + min;

}

Файл death.js

'use strict';

var death = [];

// Добавление смерти

exports.add = function(id\_kill, id\_death) {

death[death.length]=[id\_kill, id\_death];

}

// Вывод статистики

exports.get = function() {

let tmp = death;

death = [];

return tmp;

}

Файл bullet.js

'use strict';

var Config = require('../config');

var bullet = []; // Снаряды

exports.add = function(id,x,y,angle,damage) {

bullet.unshift({

x: x+36\*Math.cos(angle),

y: y+36\*Math.sin(angle),

life: 1.5,

angle: angle,

damage: damage,

user: id

});

}

exports.move = function(delta) {

for(var bi = 0; bi < bullet.length; bi++) {

// Передвижение снаряда на величину скорости

bullet[bi].x +=delta\*Config.BULLET\_SPEED\*Math.cos(bullet[bi].angle);

bullet[bi].y +=delta\*Config.BULLET\_SPEED\*Math.sin(bullet[bi].angle);

// Уменьшение времени жизни снаряда

bullet[bi].life -=delta;

// Удаление снарядов которые пролетели больше необходимого

if(bullet[bi].life < 0)

bullet.splice(bi, 1);

}

}

exports.remove = function(id) {

bullet.splice(id, 1);

}

exports.getList = function() {

return bullet;

}

Файл physics.js

'use strict';

var p2 = require('p2');

var world = new p2.World({

gravity:[0, 0]

});

var PhysicsObject=[];

// Добавление танка в мир

exports.addTank = function(id,tank,x,y,angle) {

let mass;

switch (tank) {

case 'light':

mass=15.5;

break;

case 'medium':

mass=36;

break;

case 'heavy':

mass=188.9;

break;

default:

mass=15.5;

}

let body = new p2.Body({ mass: mass, position: [x,y],angle: angle, damping: .1,angularDamping: .95 });

PhysicsObject[id]=body.id;

switch (tank) {

case 'light':

//light

body.fromPolygon([[22,26],[22,19.6],[34,19.6], [34,-19.6],[22,-19.6],[22,-26],

[-30,-26],[-30,-19.6],[-34,-19.6], [-34,19.6],[-30,19.6],[-30,26] ]);

break;

case 'medium':

//medium

body.fromPolygon([[32.5,32.5],[32.5,20],[25,20], [25,-20],[32.5,-20],[32.5,-32.5],

[-32.5,-32.5],[-32.5,-20],[-25,-20], [-25,20],[-32.5,20],[-32.5,32.5] ]);

break;

case 'heavy':

//heavy

body.addShape(new p2.Box({ width: 54, height: 57.6 }));

break;

default:

//light

body.fromPolygon([[22,26],[22,19.6],[34,19.6], [34,-19.6],[22,-19.6],[22,-26],

[-30,-26],[-30,-19.6],[-34,-19.6], [-34,19.6],[-30,19.6],[-30,26] ]);

}

world.addBody(body);

}

// Добавление границ в мир

exports.addBorder = function(x,y,angle) {

let body = new p2.Body({position:[x,y],angle: angle});

body.addShape(new p2.Plane());

world.addBody(body);

}

// Изменение танка в мире по ид пользователя

exports.updateTankSpeed = function(id,speed=0) {

let changeTankId = world.getBodyById(PhysicsObject[id]);

changeTankId.velocity[0] = speed\*Math.cos(changeTankId.angle);

changeTankId.velocity[1] = speed\*Math.sin(changeTankId.angle);

}

exports.updateTankAngle = function(id,angle=0) {

world.getBodyById(PhysicsObject[id]).angularVelocity=angle;

}

// Получение информации о танке из мира по ид пользователя

exports.getTank = function(id) {

return world.getBodyById(PhysicsObject[id]);

}

// Удаление танка из мира по ид пользователя

exports.removeTank = function(id) {

world.removeBody(world.getBodyById(PhysicsObject[id]));

PhysicsObject.splice(id, 1);

}

exports.worldStep = function(fps) {

world.step(1/fps);

}

Файл stats.js

'use strict';

var stats = [];

// Изменение статистики игрока по ид пользователя

exports.update = function(id\_kill, id\_death) {

for(var i = 0, l = stats.length; i < l; i++) {

if (stats[i].id == id\_kill)

stats[i].kill++;

if (stats[i].id == id\_death)

stats[i].death++;

}

}

// Добавление статистики игрока по ид пользователя

exports.add = function(id) {

stats.unshift({

id: id,

kill: 0,

death: 0

});

}

// Удаление статистики игрока по ид пользователя

exports.remove = function(id) {

for(var i = 0; i < stats.length; i++) {

if (id == stats[i].id)

stats.splice(i, 1);

}

}

// Вывод статистики

exports.get = function() {

return stats;

}

Файл server.js

'use strict';

var io = require('socket.io')(3600);

io.set('transports', ['websocket']);

io.noServer = false;

io.clientTracking = false;

io.perMessageDeflate = false;

var Config = require('./config');

var Death = require('./modules/death');

var Stats = require('./modules/stats');

var Physics = require('./modules/physics');

var Control = require('./modules/control');

var Bullet = require('./modules/bullet');

var User = require('./modules/user');

var Core = require('./core');

console.log('Starting server');

//node -nouse-idle-notification -expose-gc -max-old-space-size=4096 server.js

// DEBUG=socket.io\* node server.js

// node --prof server

// node --prof-process isolate-0x\*-v8.log > processed.txt

// Отправка данных игроков //sockets

setInterval(function SendStats() {

io.emit('stats', {data: {stats: Stats.get(),death: Death.get()}});

}, 750);

// Отправка данных игроков //sockets

setInterval(function SendRefresh() {

process.nextTick(function () {

var t = Date.now();

io.emit('refresh', [SendArray(User.getList(),Bullet.getList()), t]);

});

}, 1000/Config.FPS\_send);

/\*

//counter++;

var counter=0;

setInterval(function() {

console.log(counter);

counter=0;

}, 1000);

\*/

io.on('connection', function (socket) {

// Обработка полученых от пользователя данных

socket.on('refresh', (m) => Control.update(socket.id, m));

// Вход нового игрока и создание данных о нем

socket.on('join', function (m) {

// Вывод в лог информации о новом игроке

console.log('Join new client: ' + socket.id);

// Отправка данных соединения

socket.emit('init', {tank: Config.tank, stats: Stats.get, bs: Config.BULLET\_SPEED});

// Добавление данных нового пользователя

User.add(socket.id,m.name,m.tank);

Stats.add(socket.id);

Control.add(socket.id);

});

// Обработка дисконекта

socket.on('disconnect', function (m) {

//io.emit('user disconnected');

// Удаление данных удаленного игрока

io.clients(function(error, clients) {

if (error) throw error;

let users = User.getList();

users.forEach(function (value, index) {

let del=true;

clients.forEach(function (val) {

if (val == value.id) {

del=false;

}

});

if(del) {

Stats.remove(value.id);

User.remove(index);

Control.remove(value.id);

}

});

});

// Вывод в лог информации о дисконекте

console.log("Disconnect: " + m);

});

});

// Оптимизация отправляемых данных

function SendArray(users, bullets) {

var array=[[],[]];

for(var ui = 0, ul = users.length; ui < ul; ui++) {

array[0][ui]=[

((users[ui].x\*100) | 0)/100,

((users[ui].y\*100) | 0)/100,

((users[ui].speed\*100) | 0)/100,

((users[ui].turret.angle\*100000) | 0)/100000,

((users[ui].chassis.angle\*100000) | 0)/100000,

users[ui].hp,

users[ui].tank,

users[ui].name,

users[ui].id

];

}

for(var bi = 0, bl = bullets.length; bi < bl; bi++) {

array[1][bi]=[

((bullets[bi].x\*100) | 0)/100,

((bullets[bi].y\*100) | 0)/100,

((bullets[bi].angle\*100000) | 0)/100000

];

}

return array;

}

Программный код подсистемы «Клиент»

Файл background.js

// Отрисовка игрового фона (точек)

function Background() {

ctxbg.save();

//Camera();

if(Data.scale != 1) ctxbg.scale(Data.scale, Data.scale);

let roundedX = (0.5 + -Data.vision.center.x+bg.width/2/Data.scale) | 0;

let roundedY = (0.5 + -Data.vision.center.y+bg.height/2/Data.scale) | 0;

ctxbg.translate(roundedX, roundedY);

for (var x = -2560; x < 2560; x += 256) {

for (var y = -2560; y < 2560; y += 256) {

if (IsVisible(x, y,256,256)) {

ctxbg.drawImage(images.background, x, y,256,256);

}

}

}

ctxbg.restore();

}

Файл control.js

'use strict';

window.onmousemove = function(event) {

Data.control.mouse.x = event.pageX;

Data.control.mouse.y = event.pageY;

Data.control.mouse.angle = Math.atan2(event.pageY-game.height/2,event.pageX-game.width/2);// / Math.PI \* 180;

}.throttle(50);

window.onmousedown = () => {

Data.control.mouse.down = true;

};

window.onmouseup = () => {

Data.control.mouse.down = false;

};

window.onwheel = event => { // если выключить синхр пропадет?

if(event.wheelDelta > 0) {

Data.scale += 0.1;

} else {

Data.scale -= 0.1;

}

if (Data.scale < 0.5)

Data.scale = 0.5;

if (Data.scale > 1.4)

Data.scale = 1.4;

};

window.onresize = () => {

game.width = window.innerWidth-4;

game.height = window.innerHeight-4;

ui.width = window.innerWidth-4;

ui.height = window.innerHeight-4;

bg.width = window.innerWidth-4;

bg.height = window.innerHeight-4;

};

window.onkeydown = event => {

if(event.keyCode == 87) Data.control.key.w=true;

if(event.keyCode == 83) Data.control.key.s=true;

if(event.keyCode == 65) Data.control.key.a=true;

if(event.keyCode == 68) Data.control.key.d=true;

if(event.keyCode == 81) Data.control.key.q=true;

};

window.onkeyup = event => {

if(event.keyCode == 87) Data.control.key.w=false;

if(event.keyCode == 83) Data.control.key.s=false;

if(event.keyCode == 65) Data.control.key.a=false;

if(event.keyCode == 68) Data.control.key.d=false;

if(event.keyCode == 81) Data.control.key.q=false;

};

button.onclick = () => {

Data.playerName = document.forms["select"].elements["select-name"].value.trim();

Data.playerTank = document.forms["select"].elements["select-tank"].value.trim();

if(Data.playerTank == "light" || Data.playerTank == "medium" || Data.playerTank == "heavy") {

// console.log(Data.playerName,Data.playerTank);

document.forms["select"].classList.remove('select-active');

Connect();

UpdateLoop();

GameLoop();

BGLoop();

UILoop();

}

};

worker.onmessage = event => {

switch (event.data.type) {

case 'refresh':

Data.server=event.data.data.server;

//addTanks();

break;

case 'init':

Data.tank=event.data.data.tank;

Data.BULLET\_SPEED=event.data.data.bs;

break;

}

};

setInterval(() => {

worker.postMessage({

type: 'control',

data: {

control: Data.control

}

});

},1000/30);

function Connect() {

worker.postMessage({

type: 'connect',

data: {

name: Data.playerName,

tank: Data.playerTank

}

});

}

Файл env.js

'use strict';

var Data = {

playerName: null,// Логин игрока

playerTank: null,// Выбраный танк игрока

tank: null, // Параметры танков

scale: 1, // Коэффицент масштабирования (0.5 - 1.4)

fps: 60,

server: null, // connect

BULLET\_SPEED: null, // Скорость полета снаряда

/\*

Область видимости выглядит так

@ ====

| с |

==== #

\*/

vision: {

start: { // @ - Верхний левый угол

x: 0,

y: 0

},

center: { // с - Центр

x: 0,

y: 0

},

end: { // # - правый нижний

x: 0,

y: 0

}

},

// передаем нажатие мыши и клавиатуры

control: {

mouse: {

x: 0, // положение мыши по x

y: 0, // положение мыши по y

down: false, // Нажата ли кнопка

angle: 0 // На какой угол от центра находиться указатель

},

key: {

w: false, // ↑

s: false, // ↓

a: false, // ←

d: false, // →

q: false // Stats vision

}

}

}

Файл game.js

// Центрирование холста на игрока

function Camera() {

if(Data.scale != 1) ctxgame.scale(Data.scale, Data.scale);

let roundedX = ((0.5 + (-Data.vision.center.x+game.width/2/Data.scale)\*10) | 0)/10;

let roundedY = ((0.5 + (-Data.vision.center.y+game.height/2/Data.scale)\*10) | 0)/10;

ctxgame.translate(roundedX, roundedY);

}

// Проверка видимости объекта игроком на холсте

function IsVisible(x,y,w=0,h=0) {

let IsVisible=false;

if (Data.vision.start.x<x+w)

if (Data.vision.start.y<y+h)

if (Data.vision.end.x>x)

if (Data.vision.end.y>y)

IsVisible=true;

return IsVisible;

}

// Возврат среднего значения массива

function ArrayAverage(array) {

return (array.reduce(function(previousValue, currentValue) {

return previousValue + currentValue;

})/array.length);

}

// Отрисовка танков на игровом поле

function DrawTanks() {

ctxgame.save();

Camera(); // Центриование на игрока

for(var ui = 0, ul = Data.server.users.length; ui < ul; ui++) {

if (IsVisible(Data.server.users[ui].x-40, Data.server.users[ui].y-40,80,80)) {// Видит ли танки

var roundedX = ((0.5 + Data.server.users[ui].x\*10) | 0)/10;

var roundedY = ((0.5 + Data.server.users[ui].y\*10) | 0)/10;

switch (Data.server.users[ui].tank) {

case 'light':

// Отрисовка шасси легкого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(Data.server.users[ui].chassis.angle);

// SVG.light.chassis.draw(ctxgame);

ctxgame.drawImage(images.lchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни легкого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(Data.server.users[ui].turret.angle);

// SVG.light.turret.draw(ctxgame);

ctxgame.drawImage(images.lturret, -21.5, -40, 80,80);

ctxgame.restore();

break;

case 'medium':

// Отрисовка шасси среднего танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(Data.server.users[ui].chassis.angle);

// SVG.medium.chassis.draw(ctxgame);

ctxgame.drawImage(images.mchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни среднего танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY+2.5);

ctxgame.rotate(Data.server.users[ui].turret.angle);

// SVG.medium.turret.draw(ctxgame);

ctxgame.drawImage(images.mturret, -33.5, -40, 80,80);

ctxgame.restore();

break;

case 'heavy':

// Отрисовка шасси тяжелого танка

ctxgame.save();

ctxgame.translate(roundedX, roundedY);

ctxgame.rotate(Data.server.users[ui].chassis.angle);

// SVG.heavy.chassis.draw(ctxgame);

ctxgame.drawImage(images.hchassis, -40,-40, 80,80);

ctxgame.restore();

// Отрисовка башни тяжелого танка

ctxgame.save();

ctxgame.translate(roundedX, Math.round(Data.server.users[ui].y\*10)/10);

ctxgame.rotate(Data.server.users[ui].turret.angle);

// SVG.heavy.turret.draw(ctxgame);

ctxgame.drawImage(images.hturret, -27, -40, 80,80);

ctxgame.restore();

break;

}

ctxgame.save();

// Отображение ника игрока

ctxgame.textAlign = "center";

ctxgame.textBaseline = "bottom";

ctxgame.fillStyle = "#f66";

ctxgame.font = 'bold 24px monospace';

ctxgame.fillText(Data.server.users[ui].name,roundedX, roundedY-56);

// Отображение хп

ctxgame.beginPath();

ctxgame.lineWidth=4;

ctxgame.strokeStyle = "#4fc18d";

ctxgame.moveTo(roundedX-50,roundedY-54);

ctxgame.lineTo(roundedX-50+(100\*(Data.server.users[ui].hp/Data.tank[Data.server.users[ui].tank].hp)),roundedY-54);

ctxgame.stroke();

ctxgame.beginPath();

ctxgame.strokeStyle = "#000000";

ctxgame.moveTo(roundedX-50+(100\*(Data.server.users[ui].hp/Data.tank[Data.server.users[ui].tank].hp)),roundedY-54);

ctxgame.lineTo(roundedX+50,roundedY-54);

ctxgame.stroke();

ctxgame.restore();

}

}

ctxgame.restore();

}

function DrawBullets() {

ctxgame.save();

Camera();

for(var bi = 0, bl = Data.server.bullets.length; bi < bl; bi++) {

if (IsVisible(Data.server.bullets[bi].x, Data.server.bullets[bi].y)) {

ctxgame.save();

ctxgame.translate(Data.server.bullets[bi].x, Data.server.bullets[bi].y);

ctxgame.rotate(Data.server.bullets[bi].angle);

ctxgame.drawImage(images.bullet, -12.5,-12.5,25,25);

ctxgame.restore();

}

}

ctxgame.restore();

}

Файл loader.js

(function( w ){

var loadJS = function( src, cb ){

"use strict";

var ref = w.document.getElementsByTagName( "script" )[ 0 ];

var script = w.document.createElement( "script" );

script.src = src;

script.async = true;

ref.parentNode.insertBefore( script, ref );

if (cb && typeof(cb) === "function") {

script.onload = cb;

}

return script;

};

// commonjs

if( typeof module !== "undefined" ){

module.exports = loadJS;

}

else {

w.loadJS = loadJS;

}

}( typeof global !== "undefined" ? global : this ));

// Данные о изображениях

var images = {};

var loadedImages = 0;

var numImages = 0;

var sourceImages = {

//background: "images/background.svg",

background: "images/background.png",

lchassis: "images/tank/light/chassis.svg",

lturret: "images/tank/light/turret.svg",

mchassis: "images/tank/medium/chassis.svg",

mturret: "images/tank/medium/turret.svg",

hchassis: "images/tank/heavy/chassis.svg",

hturret: "images/tank/heavy/turret.svg",

bullet: "images/bullet.svg",

death: "images/death.png",

};

// Данные о JS

var loadedJS = 0;

var numJS = 0;

var sourceJS = {

limit: "js/vendor/limit.js", // первый загружается??

// heavy\_chassis: "images/tank/heavy/chassis.js",

// heavy\_turret: "images/tank/heavy/turret.js",

//

// medium\_chassis: "images/tank/medium/chassis.js",

// medium\_turret: "images/tank/medium/turret.js",

//

// light\_chassis: "images/tank/light/chassis.js",

// light\_turret: "images/tank/light/turret.js",

update: "js/update.js",

game: "js/game.js",

ui: "js/ui.js",

background: "js/background.js",

render: "js/render.js",

control: "js/control.js",

//p2: "js/vendor/p2.js",

//physics: "js/physics.js",

env: "js/env.js"

};

// Подсчет загружаемых файлов

for(var src in sourceImages) {

numImages++;

}

for(var src in sourceJS) {

console.log(src);

numJS++;

}

for(var src in sourceImages) {

images[src] = new Image();

images[src].src = sourceImages[src];

images[src].onload = function () {

loadedImages++;

Loading();

};

}

for(var src in sourceJS) {

loadJS(sourceJS[src],function () {

loadedJS++;

Loading();

});

}

function Loading() {

if(loadedJS+loadedImages == numJS+numImages) {

document.body.classList.add('active');

}

document.getElementById('progress--after').style.width

=Math.round((loadedJS+loadedImages)/(numJS+numImages)\*100) + '%';

console.log(Math.round((loadedJS+loadedImages)/(numJS+numImages)\*100) + "% resource loaded");

}

var worker = new Worker('js/worker.js');

Файл render.js

'use strict';

var last = performance.now();

var dt;

function UpdateLoop() {

let now = performance.now();

dt = (now - last) / 1000;

last=now;

Approximation();

requestAnimationFrame(UpdateLoop);

}

var game = document.getElementById("game-layer"),

ctxgame = game.getContext('2d');

game.width = window.innerWidth-4;

game.height = window.innerHeight-4;

function GameLoop() {

ctxgame.clearRect(0, 0, game.width, game.height);

DrawBullets();

DrawTanks();

requestAnimationFrame(GameLoop);

}

var ui = document.getElementById("ui-layer"),

ctxui = ui.getContext('2d');

ui.width = window.innerWidth-4;

ui.height = window.innerHeight-4;

function UILoop() {

ctxui.clearRect(0, 0, ui.width, ui.height);

DrawLog();

Radar();

DrawStat();

requestAnimationFrame(UILoop);

}

var bg = document.getElementById("background-layer"),

ctxbg = bg.getContext('2d');

bg.width = window.innerWidth-4;

bg.height = window.innerHeight-4;

function BGLoop() {

ctxbg.clearRect(0, 0, bg.width, bg.height);

Background();

requestAnimationFrame(BGLoop);

}

Файл ui.js

// Отрисоквка радара

function Radar() {

// Отрисовка фона радара

ctxui.save();

ctxui.beginPath();

ctxui.lineWidth = 2;

ctxui.fillStyle = "rgba(255, 255, 255, 0.85)";

ctxui.arc(112, 112, 96, 0, Math.PI\*2, false);

ctxui.closePath();

ctxui.stroke();

ctxui.fill();

//Данные об игроке

let angle;

let x,y;

Data.server.users.forEach(function (value, index) {

if (value.id == Data.server.id) {

angle = value.turret.angle;

x = value.x;

y = value.y;

}

});

// Отрисовка сектора направления орудия

ctxui.beginPath();

ctxui.lineWidth = 1;

ctxui.fillStyle = "#4fc18d";

ctxui.arc(112,112,96,angle-(Math.PI/6),angle+(Math.PI/6));

ctxui.lineTo(112, 112);

ctxui.closePath();

ctxui.stroke();

ctxui.fill();

// Данные об окружающих игроках

Data.server.users.forEach(function (value, index) {

if (value.id != Data.server.id) {

let r = Math.sqrt(Math.pow(Math.abs(value.y-y),2)+Math.pow(Math.abs(value.x-x),2));

if(r <= 2000) {

let degree = Math.atan2(value.y-y,value.x-x);

ctxui.beginPath();

ctxui.fillStyle = "#f66";

ctxui.arc(112+96\*(r/2000)\*Math.cos(degree), 112+96\*(r/2000)\*Math.sin(degree), 2, 0, Math.PI\*2, false);

ctxui.closePath();

ctxui.fill();

}

}

});

ctxui.restore();

}

// Отрисовка логов о смертях и убийствах

function DrawLog() {

Data.server.death.forEach(function (value, index) {

ctxui.save();

let death\_name="",killer\_name="";

Data.server.users.forEach(function (player) {

if (player.id == value.killer)

killer\_name = player.name;

if (player.id == value.death)

death\_name = player.name;

});

ctxui.textBaseline = "top";

ctxui.fillStyle = "#f66";

ctxui.font = 'bold 24px monospace';

ctxui.textAlign = "start";

ctxui.fillText(death\_name, game.width-225, 12.5+36\*index);

ctxui.textAlign = "end";

ctxui.fillText(killer\_name, game.width-300, 12.5+36\*index);

ctxui.drawImage(images.death, game.width-280, 12.5+36\*index, 26, 38);

ctxui.restore();

});

}

// Отрисовка логов о смертях и убийствах

function DrawStat() {

if (Data.control.key.q) {

ctxui.save();

ctxui.beginPath();

ctxui.lineWidth=3;

ctxui.strokeStyle = "#000000";

ctxui.moveTo(game.width\*.8,game.height\*.8);

ctxui.lineTo(game.width\*.8,game.height\*.2);

ctxui.lineTo(game.width\*.2,game.height\*.2);

ctxui.lineTo(game.width\*.2,game.height\*.8);

ctxui.lineTo(game.width\*.8,game.height\*.8);

ctxui.fillStyle = "white";

ctxui.fill();

ctxui.stroke();

ctxui.textBaseline = "top";

ctxui.fillStyle = "#000000";

ctxui.font = 'bold 24px monospace';

ctxui.textAlign = "center";

ctxui.fillText("TOP 10 PLAYER", game.width\*.5, game.height\*.21);

ctxui.textBaseline = "top";

ctxui.fillStyle = "#000000";

ctxui.font = 'bold 20px monospace';

ctxui.textAlign = "start";

ctxui.fillText('#', game.width\*.24, game.height\*(.28));

ctxui.fillText('Login', game.width\*.28, game.height\*(.28));

ctxui.fillText('Kill', game.width\*.64, game.height\*(.28));

ctxui.fillText('Death', game.width\*.72, game.height\*(.28));

let unsort = Data.server.stats;

unsort.sort(sDecrease).forEach(function (value, index) {

if (index < 10) {

let name;

Data.server.users.forEach(function (val) {

if(value.id == val.id) name = val.name;

});

ctxui.fillText(index+1, game.width\*.24, game.height\*(.32+0.043\*index));

ctxui.fillText(name, game.width\*.28, game.height\*(.32+0.043\*index));

ctxui.fillText(value.kill, game.width\*.64, game.height\*(.32+0.043\*index));

ctxui.fillText(value.death, game.width\*.72, game.height\*(.32+0.043\*index));

}

});

ctxui.restore();

}

}

function sDecrease(i, ii) { // По убыванию убийств

if (i.kill > ii.kill)

return -1;

else if (i.kill < ii.kill)

return 1;

else

return 0;

}

Файл update.js

//Модуль апроксимации

function Approximation() {

// Полет снарядов

for(var bi = 0, bl = Data.server.bullets.length; bi < bl; bi++) {

if (IsVisible(Data.server.bullets[bi].x-12.5, Data.server.bullets[bi].y-12.,25,25)) {

// Передвижение снаряда на величину скорости

Data.server.bullets[bi].x +=(dt\*Data.BULLET\_SPEED)\*Math.cos(Data.server.bullets[bi].angle);

Data.server.bullets[bi].y +=(dt\*Data.BULLET\_SPEED)\*Math.sin(Data.server.bullets[bi].angle);

Data.server.bullets[bi].x = ((0.5 + Data.server.bullets[bi].x\*10) | 0)/10;

Data.server.bullets[bi].y = ((0.5 + Data.server.bullets[bi].y\*10) | 0)/10;

}

}

// Передвижение танка

for(var ui = 0, ul = Data.server.users.length; ui < ul; ui++) {

if (Data.server.users[ui].id == Data.server.id) {

// Обновление данных о направление орудия

//Data.server.users[ui].turret.angle = mouse.angle;

Data.server.users[ui].turret.angle = RotateTurret(Data.server.users[ui].turret.angle,Data.control.mouse.angle,(dt\*Data.tank[Data.server.users[ui].tank].turret.angle\*Math.PI/180));

// Вычисление изменений скорости

if(Data.control.key.w) {

Data.server.users[ui].speed += (dt\*Data.tank[Data.server.users[ui].tank].acceleration\*ac(Data.server.users[ui].speed,Data.tank[Data.server.users[ui].tank].max\_speed));

if (Data.server.users[ui].speed > Data.tank[Data.server.users[ui].tank].max\_speed) Data.server.users[ui].speed = Data.tank[Data.server.users[ui].tank].max\_speed;

}

if(Data.control.key.s) {

Data.server.users[ui].speed -= (dt\*Data.tank[Data.server.users[ui].tank].acceleration\*0.6\*ac(Data.server.users[ui].speed,Data.tank[Data.server.users[ui].tank].max\_speed));

if (Data.server.users[ui].speed < -Data.tank[Data.server.users[ui].tank].max\_speed\*0.3) Data.server.users[ui].speed = -Data.tank[Data.server.users[ui].tank].max\_speed\*0.3;

}

if((!Data.control.key.s || Data.server.users[ui].speed > 0) && !Data.control.key.w) {

if (Data.server.users[ui].speed < 10 && Data.server.users[ui].speed > -10) {

Data.server.users[ui].speed = 0;

}

if (Data.server.users[ui].speed <= -10) {

Data.server.users[ui].speed += (dt\*Data.tank[Data.server.users[ui].tank].acceleration\*3\*ac(Data.server.users[ui].speed,Data.server.users[ui].max\_speed));

}

if (Data.server.users[ui].speed >= 10) {

Data.server.users[ui].speed -= (dt\*Data.tank[Data.server.users[ui].tank].acceleration\*3\*ac(Data.server.users[ui].speed,Data.server.users[ui].max\_speed));

}

}

// Поворот танка

if(Data.control.key.a) Data.server.users[ui].chassis.angle -= (dt\*Data.tank[Data.server.users[ui].tank].chassis.angle\*Math.PI/180\*ac(Data.server.users[ui].speed,Data.tank[Data.server.users[ui].tank].max\_speed));

if(Data.control.key.d) Data.server.users[ui].chassis.angle += (dt\*Data.tank[Data.server.users[ui].tank].chassis.angle\*Math.PI/180\*ac(Data.server.users[ui].speed,Data.tank[Data.server.users[ui].tank].max\_speed));

Data.vision.start.x = Data.server.users[ui].x-game.width/2/Data.scale;

Data.vision.start.y = Data.server.users[ui].y-game.width/2/Data.scale;

Data.vision.center.x = Data.server.users[ui].x;

Data.vision.center.y = Data.server.users[ui].y;

Data.vision.end.x = Data.server.users[ui].x+game.width/2/Data.scale;

Data.vision.end.y = Data.server.users[ui].y+game.height/2/Data.scale;

}

if (IsVisible(Data.server.users[ui].x-40, Data.server.users[ui].y-40,80,80)) {

// Передвижение танка на величину скорости

Data.server.users[ui].x +=(dt\*Data.server.users[ui].speed)\*Math.cos(Data.server.users[ui].chassis.angle);

Data.server.users[ui].y +=(dt\*Data.server.users[ui].speed)\*Math.sin(Data.server.users[ui].chassis.angle);

// Граница карты

if(Data.server.users[ui].x > 2560) Data.server.users[ui].x = 2560;

if(Data.server.users[ui].y > 2560) Data.server.users[ui].y = 2560;

if(Data.server.users[ui].x < -2560) Data.server.users[ui].x = -2560;

if(Data.server.users[ui].y < -2560) Data.server.users[ui].y = -2560;

}

}

}

// Функция плавного поворота башни

function RotateTurret(ta,ma,s) {

if (ta < 0)

ta = ta + 2\*Math.PI;

let a = ta - ma;

if (a > Math.PI)

a = a - 2\*Math.PI;

else if (a < -Math.PI)

a = a + 2\*Math.PI;

if (a >= -s && a <= s)

ta = ma;

else if (a > s)

ta = ta - s;

else if (a < -s)

ta = ta + s;

return ta;

}

function ac(speed, max\_speed=120) {

//(2+x^0.1)/(2+x^0.3) (x from 0 to 100)

// получаем коэффицент скорости разворота

// от 1 до почти 0.616

// перемножаем на скорость поворота танка

return (1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.1))/(1+Math.pow((Math.abs(speed)/max\_speed)\*10,0.3));

}

Файл worker.js

self.importScripts('vendor/socket.io-1.4.5.js');

//self.importScripts('vendor/p2.js');

var socket = io('http://localhost:3600/',{transports: ['websocket']});

//var socket = io('https://ws.mertico.ru/',{transports: ['websocket'], secure: true});

var server={

death:[],

bullets:[],

users:[],

stats:[],

id:0

};

var timeStamp=0, startTime=0;

socket.on('connect', () => { server.id=server.id; });

onmessage = event => {

var message = event.data;

switch (message.type) {

case 'connect':

//socket.on('connect', function () {

socket.emit('join', {name: message.data.name, tank: message.data.tank});

//socket.emit('join', {name: 123, tank: 13});

//});

break;

case 'control':

//control=message.data.control;

socket.emit('refresh', sendControl(message.data.control));

break;

}

};

socket.on('init', function (m) {

server.stats=m.stats;

postMessage({

type: 'init',

data: {

tank: m.tank,

bs: m.bs

}

});

});

socket.on('stats', function (m) {

m.data.death.forEach(function(element, index) {

server.death.unshift({killer: element[0], death: element[1], life : 3});

});

server.death.splice(5, 1);

server.stats=m.data.stats;

});

socket.on('refresh', function (m) {

// Проверка временой метки (насколько актуален пакет данных)

if(m[1] >= timeStamp) {

timeStamp = m[1];

[server.users,server.bullets] = InSend(m[0]);

postMessage({

type: 'refresh',

data: {

server

}

});

}

});

setInterval(() => {

server.death.forEach(function (value, index) {

value.life-=1;

if(value.life<=0)

server.death.splice(index, 1);

});

},1000);

function sendControl(control) {

return [

control.key.w && 1 || // move player

control.key.s && -1 ||

0,

control.key.d && 1 || // rotate player

control.key.a && -1 ||

0,

((0.5 + control.mouse.angle\*100000) | 0)/100000, // mouse angle

control.mouse.down // fire

];

}

function InSend(data) {

let users=[], bullets=[];

for(var index = 0, length = data[0].length; index < length; index++) {

users[index] = {

x: data[0][index][0],

y: data[0][index][1],

speed: data[0][index][2],

turret: {

angle: data[0][index][3]

},

chassis: {

angle: data[0][index][4]

},

hp: data[0][index][5],

tank: data[0][index][6],

name: data[0][index][7],

id: data[0][index][8]

};

}

for(var index = 0, length = data[1].length; index < length; index++) {

bullets[index] = {

x: data[1][index][0],

y: data[1][index][1],

angle: data[1][index][2]

};

}

return [users, bullets];

}

Программный код подсистемы «Искусственные пользователи»

Файл server.js

'use strict';

const cluster = require('cluster');

var socketio = require('socket.io-client');

//const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {

// Запуск воркеров по кол-ву потоков минус поток мастера

for (var i = 0; i < 2; i++) {

cluster.fork();

}

cluster.on('exit', (worker, code, signal) => {

console.log(`Worker #${worker.id} pid:${worker.process.pid} died`);

if( signal ) {

console.log("worker was killed by signal: "+signal);

} else if( code !== 0 ) {

console.log("worker exited with error code: "+code);

} else {

console.log("worker success!");

}

cluster.fork();

});

} else {

var socket = socketio('http://localhost:3600',{transports: ['websocket']});

socket.on('connect', function () {

socket.emit('join', {name: "[Bot "+cluster.worker.id+"]", tank: (Math.random() > .66) ? 'heavy' : (Math.random() > .5) ? 'light' : 'medium'});

//socket.on('join', function (m) {

console.log('Bot №'+cluster.worker.id+' connect');

//});

});

socket.on('disconnect', function () {

console.log('Bot №'+cluster.worker.id+' disconnect');

});

socket.on('refresh', function (m) {

var [users,bullets] = InSend(m[0]);

var bot = {

x:0,

y:0,

turret: 0,

chassis: 0

};

// передаем нажатие мыши и клавиатуры

var mouse = {

x : 0,

y : 0,

down : false,

angle : 0

};

var key = {

w : false, // ↑

s : false, // ↓

a : false, // ←

d : false // →

};

var target = {

x: 0,

y: 0,

angle: -Math.PI / 2,

r: 9999999999,

track: false

};

for(var ui = 0, ul = users.length; ui < ul; ui++) {

if (users[ui].id == socket.id) {

bot.x=users[ui].x;

bot.y=users[ui].y;

bot.turret=users[ui].turret.angle;

bot.chassis=users[ui].chassis.angle;

}

}

for(var ui = 0, ul = users.length; ui < ul; ui++) {

if (users[ui].id != socket.id) {

let r = Math.sqrt(Math.pow(Math.abs(users[ui].y-bot.y),2)+Math.pow(Math.abs(users[ui].x-bot.x),2));

if (target.r > r) {

// Передвижение танка на величину скорости для стрельбы на опережнение

//users[ui].x +=users[ui].speed\*Math.cos(users[ui].chassis.angle);

//users[ui].y +=users[ui].speed\*Math.sin(users[ui].chassis.angle);

target.x=users[ui].x;

target.y=users[ui].y;

target.angle=Math.atan2(users[ui].y-bot.y,users[ui].x-bot.x);

target.angle+=(users[ui].speed\*Math.sin(users[ui].chassis.angle-bot.turret)/900);

target.r=r;

target.track=true;

}

}

}

if (target.track) {

key.w=true;

if (bot.chassis.toFixed(1) != target.angle.toFixed(1)) {

if (bot.chassis < 0)

bot.chassis = bot.chassis + 2\*Math.PI;

let a = bot.chassis - target.angle;

let s = Math.PI/18

if (a > Math.PI)

a = a - 2\*Math.PI;

else if (a < -Math.PI)

a = a + 2\*Math.PI;

if (a >= -s && a <= s)

key.w=true;

else if (a > s)

key.a=true;

else if (a < -s)

key.d=true;

}

if(target.r<300)

key.w=false;

}

if (target.r<2000) {

mouse.angle=target.angle;

}

if (target.r<1500) {

if (bot.turret.toFixed(1) == target.angle.toFixed(1)) {

mouse.down=true;

}

}

socket.emit('refresh', sendControl({mouse : mouse,key :key}));

});

function sendControl(control) {

return [

control.key.w && 1 || // move player

control.key.s && -1 ||

0,

control.key.d && 1 || // rotate player

control.key.a && -1 ||

0,

((0.5 + control.mouse.angle\*100000) | 0)/100000, // mouse angle

control.mouse.down // fire

];

}

function InSend(data) {

let users=[], bullets=[];

for(var index = 0, length = data[0].length; index < length; index++) {

users[index] = {

x: data[0][index][0],

y: data[0][index][1],

speed: data[0][index][2],

turret: {

angle: data[0][index][3]

},

chassis: {

angle: data[0][index][4]

},

hp: data[0][index][5],

tank: data[0][index][6],

name: data[0][index][7],

id: data[0][index][8]

};

}

for(var index = 0, length = data[1].length; index < length; index++) {

bullets[index] = {

x: data[1][index][0],

y: data[1][index][1],

angle: data[1][index][2]

};

}

return [users, bullets];

}

}