Electrodynamics2d\_websocket

Contents

[Сборка и запуск 2](#_Toc117106003)

[Архитектура 3](#_Toc117106004)

[Текущее состояние 3](#_Toc117106005)

[План 3](#_Toc117106006)

# Сборка и запуск

Particles.sln включает два приложения – Server and Room. Они собираются только под Windows.

Приложения используют библиотеки websocketpp, boost\_1\_66 (достаточно бинарнх файлов), protobuf (версия 3.6.1), и pugixml (есть в репозитории). Помучившись, можно собрать.

Protobuf используется для формирования и чтения пакетов в сети в пакетах, используемых протоколом websocket, они читаются/пишутся в коде на c++ и js, поэтому нужно запускать соответствующие батники для генерации нужных заголовочных фалов (proto\_cpp.bat и proto\_js.bat). Версия 3.6.1 работает корректно, в некоторых других есть проблемы.

Сейчас, чтобы запустить систему, нужно запустить сервер с параметрами server.exe 127.0.0.1 27000 и room с параметрами room.exe 26001 52\_pm\_2p.xml – и будет запущен сервер, который будет ждать от веб-клиента (браузера) обращения к порту 27000, но при этом будет общаться с room’ом через порт 26001 (в сервере этот порт сейчас захардкорен). Веб-клиент после минимальных переговоров с сервером подкоючится к игровой комнате и будет общаться уже с ней. Как организована передача портов, я сейчас точно не помню, но это в любом случае нужно будет переделать (см. дальше).

Кроме того, нужно запустить game.html, предварительно в нем поменяв строки с

var roomMode = "{{ roomMode }}";

var roomIP = "{{ roomIP }}";

var roomPort = {{ roomPort }};

var webRTCGameId = {{ webRTCGameId }};

на

var roomMode = "play";//"{{ roomMode }}";

var roomIP = "127.0.0.1";//"{{ roomIP }}";

var roomPort = 26001;//{{ roomPort }};

var webRTCGameId = -1;//{{ webRTCGameId }};

В дальнейшем это все тоже нужно будет как-то поисправлять.

# Архитектура

## Текущее состояние

Сейчас сервер и игровую комнату нужно запускать вручную. Версия с main.html работала, там вручную запускать вроде ничего не нужно было (хотя не уверен, что я все правильно помню, но это влюбом случае нужно поменять). Но с main.html есть несколько проблем:

1. предполагалось, что веб-сервер работает в экосистеме облака гугл, а от этого нужно уйти
2. я реализовывал протокол WebRTC на стороне веб-клиента и веб-сервера для организации работы «браузер-браузер»; для двух игроков это работало, но как-то тяжело, для трех и больше не работало; скорее всего, этот протокол пока что нужно отложить. Если же в будущем я к нему вернусь, то нужно попытаться сделать все через уже готовые реализации, а то выходит слишком сложно для поддержки
3. реализация в 2) имела еще одну проблему – в процессе переговоров веб-клиент формировал соединение, которое должно было передаваться дальше – на веб-страницу игры; но оказалось, что так делать нельзя, т.к. в этом случае все ресурсы теряются; нужно было организовывать одно-документную реализацию с сокрытием части контролов; так, видимо, и нужно поступить при переделках

## План

1. Пофиксить несколько проблем в game.html:
   1. как-то криво отрисовываются силовые линии, а эквипотенциальные не отрисовываюся вообще; нужно дать возможность на стороне клиента управлять бОльшим количеством параметров отрисовки,- например, формулой, по которой считатся цвет потенциала в точке
   2. не подгружаются палитры; нужно найти способ их загрузки на клиенте в любом случае
   3. пофиксить прочие проблемы отрисовки шейдеров
   4. и вообще, нужно создать список багов
2. Сервер и игровая комната написаны на с++, это нужно поменять на javascript, и запускать их на NodeJS. При этом код на веб-клиенте, который ведет рассчеты, брать оттуда же. Таким способом избавляюсь и от web-assembly, оно здесь не нужно.
3. Нужно придумать какое-то более архитектурное решение для запуска игровых комнат и работы с ними – вроде запуска их в докере или как-то так.
4. Изменить main.html
   1. single document architecture
   2. пока что без webrtc, это потом
5. Веь-сервер переписать на фласке, соответственно, возможно, частично, поменять веб-клиент.

Итого, новая архитектура должна выглядеть следующим образом (нужно описать функции каждого узла)

database

Web-server (port 8080, http)

Room

(port 26001,

websocket)

Web-client 1

Room

(port 26004)

Room

(port 26005)

Room

(port 26003)

Room

(port 26002)

Web-client 2

Socket-server

(port 27000,

websocket)

Web-client 3

Краткое описание таково.

1. Предварительно сокет-сервер сообщает свой ip адрес веб-серверу. Вопрос с зеркалами пока не стоял.
2. Веб-клиент обращается к веб-серверу. Веб-сервер первоначально предоставляет веб-клиенту информацию об уже запущенных играх, - собственно только порты игровых комнат и из описание. Веб-клиент может обратится к игровым комнатам чтобы или смотреть уже запущенные игры, или подключиться к игре, если она в состоянии ожидания. Также веб-клиент может запросить веб-сервер создать новую игру. Тогда веб-сервер обращается к совет-серверу, а тот уже должен создать игру. Насколько я помню, пока что все комнаты нужно было создавать вручную. Веб-сервер общается с клиентом, исходя из прав, которые этот клиент имеет. Клиент может быть зарегистрированным и не зарегистрированным. Каждый клиент имеет право создать столькото там игр и т.д. Веб-сервер берет на себя всю работу и возвращает клиенту адрес комнаты.
3. Когда комната создается, она ждет подключения игроков, после этого стартует игру. Может выдавать данные только для просмотра. Никакие зеркала пока не рассматривались.