1. Постановка задачи

Задача проекта - создать браузерную игру для двоих игроков с использованием HTML, JS, а также технологией serverless-webrtc

2. Используемые алгоритмы и технологии

Для связи пользователей между собой используется технология serverless-webrtc: https://github.com/cjb/serverless-webrtc Это позволяет избежать создания сервера для работы, однако требует от пользователей больше действий для соединения. Я считаю этот вариант оптимальным, так как из-за наличия режима игры на одном устройстве, все вычисления и данные проходят на устройствах пользователей, и единственное что игрокам нужно знать друг о друге - это их ходы. Логика игры заключена в двух js файлах для двух версий игры, и их единственное отличие заключается в том, что второй файл дополнительно связывает игроков и обрабатывает получаемые данные Внутри самой игры используется структура данных для поддержания "системы непересекающихся множеств", позволяющая объединить два множества, создать новое и узнать, находятся ли два элемента в одном множестве. Для нахождения различных цепочек из фишек используется алгоритм поиска в ширину. Также для оптимальной работы этого алгоритма используется стркутура данных "очередь", умеющая добавлять элемент в конец контейнера и удалять элемент из начала. Для отрисовки нужных элементов на экране используется HTML-модуль **canvas** (далее - холст). Вся отрисовка происходит в 2d.

3.Руководство для программиста

function resize(arr, newSize, neutral) - функция принимает на вход массив, его новую длину и нейтральный элемент, возвращает массив arr с рамзером, измененным на newSize, при надобности дополненный элементами neutral

class Queue - структура данных "очередь". Имеет поля `elements, begin, end` - контейнер с элементами, индекс первого элемента, индекс последнего элемента + 1 и методы:

push(element) - добавляет element в конец очереди

length() - возвращает длину очереди

isEmpty() - проверяет очередь на пустоту - true/false

pop() - удаляет первый элемент из очереди, возвращает его

front() - возвращает первый элемент из очереди

back() - возвращает последний элемент из очереди

dsu - массив в котором явно хранятся массивы с множествами. В каждом массиве хранятся пары x, y - координанты элементов на поле

dsuEdges - массив, в котором хранятся массивы с ребрами множеств. В каждом из массиве хранятся **u, v** - индексы элементов в массиве **dsu**

function getVtxByXY(x, y, id) - принимает на вход координаты объекта и номер его множества, возвращает его индекс в этом множестве

uniteDsu(u, v) - объединяет множества с номерами **u, v**

var canvas = document.getElementById('aboba') - холст на котором идет отрисовка.

var ctx = canvas.getContext('2d'); - содержимое холста.

class Cell - структура "клетка" поля, имеет поля **x,y** - его координаты, **type** - тип клетки *(dead, alive)* и ее цвет. Имеет метод **draw()**, который отрисовывает клетку на экран

class Point() - структура "точка", появляется на поле при нажатии, конструктор принимает координанты точки, ее цвет и номер ее множества. Ее поля аналогичны + поле **type** - тип клетки *(dead, alive, border)*. Имеет метод **draw()**, который отрисовывает клетку

function startGame() - обнуляет поле, систему непересекающих множеств и номер хода и очки игроков

function restartGame() - существует только в онлайн версии, перезапускает игру, переопределяет цвет игроков

function getNeighbours(x, y) - принимает на вход координаты точки, возвращает список всех ее живых соседей точек такого же цвета, с которыми можно провести ребро и оно не будет пересекаться с уже существующими.

function drawGuys(x1, y1, x2, y2) - принимает на вход две пары координат точек, рисует между ними ребро.

function makeGraphList(edged) - принимает на вход список ребер, возвращает преобразованный из него список смежности

function findDead(vertexes) - принимает список ограничивающих точек, возвращает список мертвых клеток, "убитых" этими точками

function makeNeighbours(x, y) - принимает на вход координаты точки, проделывает различные операции с ее соседями - отрисовка ребер, нахождение цепочек точек, в

которых участвует новая точка, преобразует систему непересекающихся множеств, отрисовывает мертвые клетки.

function getCoorsByXY(x, y) - принимает координаты точки в таблице, возвращает ее координаты на холсте

function getXYByCoors(cx, cy) - принимает координаты точки на холсте, возвращает ее координаты в таблице

function getMousePos(event) - получает событие нажатия клавиши мыши на экран, возвращает ее координаты на холсте

function calculareScore() - возвращает список, где первый элемет - очки синего игрока, второй элемент - очки красного игрока, третий - флажок, который показывает закончилась ли игра(есть ли еще ходы или нет)

function tableClick(event) - принимает событие нажатия на экран, обрабатывает ход

function getStep(x, y) - существует только в онлайн версии, принимает координаты точки в таблице, которой сделал ход соперник, обрабатывает этот ход

function drawField() - отрисовывает поле

_**chatChannel.onopen** - функция реагирующая на открытие канала

_**chatChannel.onmessage** - функция реагирующая на получение сообщения, обрабатывает его

_**chatChannel.onclose** - функция реагирующая на закрытие канала

function sendMessage(text) - отправляет второму пользователю текст

4.Руководство для пользователя

Пользователя встречает игровое поле посередине экрана и информация о счёте и ходе сверху. Каждый ход игроки ставят точки на поле и получают очки в зависимости от суммарного количества принадлежащих им объектов на поле. Каждая новая точка соединяется со всеми соседними живыми клетками того же цвета, не пересекая уже существующие соединения. Клетки располагаются в узлах нарисованной сетки. Первыми начинают синие. Клетка считается мертвой, если она находится внутри замкнутой одноцветной ломаной. Мертвые клетки принадлежат окружившему их игроку. Связи мертвых клеток также принадлежат "убившему" их игроков. Мертвые клетки не могут образовывать в новых связях, создавать новые ломаные. Очки игрока - это сумма принадлежащих ему клеток и соединений. Игра заканчивается, когда на поле не остается

свободных клеток. С помощью кнопки restart сверху игра перезапускается и начинается заново.

Действия для установки соединения между игроками:

1. Нажмите кнопку **createOffer** рядом с левым полем для текста (оно находится под игровым полем)
2. Скопируйте сгенерированный текст и отправьте его второму игроку
3. Второй игрок должен вставить этот текст в правое поле для текста (находится справа от левого) и нажать кнопку **Answer**
4. Слева появляется сгенерированный текст, второй игрок должен отправить его Вам.
5. Вставьте полученный текст в правое поле для текста, нажмите **Answer**
6. Вы подключились друг к другу! Также сверху будет написан ваш цвет. В остальном интерфейс игры не отключается от оффлайн версии.
7. В случае возникновения ошибок, проверьте интернет-соединение, очистите поля для ввода и повторите написанные выше действия заново.

5.Список литературы

Правила игры-прообраза: https://ufgo.org/Rules9x9/Go%20Rules%209x9.htm

Руководство для работы с serverless-webrtc: GitHub - cjb/serverless-webrtc: A demo of using WebRTC with no signaling server.

Документация и руководство по использованию canvas: Базовое использование Canvas - Интерфейсы веб API | MDN