

Министерство образования Российской Федерации
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Факультет: «Информатика и системы управления»

Кафедра: «Информационная безопасность»



Дисциплина: «Алгоритмические языки»

Пояснительная записка по курсовому проекту
на тему:
«PvP Змейка»

Группа: ИУ8-31

Студент: Комкова О.Е.

Преподаватель: Бородин А.А.

Москва, 2017 г.

СОДЕРЖАНИЕ

ЦЕЛЬ ПРОЕКТА.....	3
ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ.....	3
ВВЕДЕНИЕ	4
ТРЕБОВАНИЯ К ПРОЕКТУ	4
1 ПРОЕКТИРОВАНИЕ СИСТЕМЫ	5
2 ВЫБОР ТЕХНОЛОГИЙ.....	11
2.1 Выбор языка программирования.....	11
2.2 Выбор используемых библиотек.....	12
3 ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ.....	12
ЗАКЛЮЧЕНИЕ	13
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	14

ЦЕЛЬ ПРОЕКТА

Отработка навыков программирования на примере разработки браузерной игры «Змейка» для двух игроков на языке программирования JavaScript.

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

JavaScript – язык программирования, позволяющий создавать приложения, выполняемые на стороне клиента, т.е. эти приложения выполняются браузером на компьютере пользователя. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

HTML – язык разметки документов, отвечает за расположение информации. Прочитывается и анализируется данный язык браузерами, которые “знакомы” со стандартными командами, на выходе получаются документы в том виде, в котором были задуманы верстальщиком.

CSS – язык стилей, позволяющий придать тегам html уникальность. Обеспечивает более точный контроль над внешним видом страницы, возможно управление отображением множества документов с помощью одной таблицы стилей.

FPS (Frames Per Second) – косвенный показатель, определяющий количество кадров в секунду на экране монитора.

Unit-тесты – процесс в программировании, позволяющий проверить исходный код на корректность, также позволяет разработчику удостовериться в том, что внесенные изменения не привели к появлению ошибок в уже оттестированных местах программы.

ВВЕДЕНИЕ

Продукт данного проекта представляет собой кроссплатформенную и легковесную игру для двух игроков. Зачастую возникает потребность в играх, однако большинство людей не любит долго скачивать и устанавливать их. Именно поэтому браузерные игры пользуются популярностью в современном мире.

Игра «Змейка» является наиболее популярной игрой всех времен. В нее могут играть люди любых возрастов. Она очень проста в изучении, сильно затягивает и тренирует реакцию. Поэтому она подходит для детей в качестве развивающего занятия, для взрослых – отдых.

Основное отличие от большинства аналогичных решений – кроссплатформенность и легковесность.

ТРЕБОВАНИЯ К ПРОЕКТУ

Игра «PvP Змейка» должна работать на операционных системах:

- MacOS;
- Windows;
- Linux.

Поддерживаться следующими браузерами:

- Chrome;
- Mozilla Firefox;
- Internet Explorer 10+;
- Opera;
- Safari.

Вес приложения не должен превышать 2 Мб, чтобы приложение оставалось легко доступным при небольшой скорости интернета.

Анимация должна работать на скорости 60 кадров в секунду, чтобы обеспечить плавные перемещения объектов, отсутствие задержек.

1 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Прежде чем приступить к написанию кода приложения, необходимо его спроектировать.

Данное приложение состоит из html5 страницы, js файла со скриптом и css файла стилей. В теле страницы находится заголовок и контейнер для игрового поля, в который скрипт будет вставлять змейки и яблоки. В скрипте происходит обработка нажатий клавиатуры, выполняется отрисовка змеек и яблок, работает игровая логика: меняются координаты змеек в зависимости от направления движения, проверяются условия поражения, наличие яблок на месте головы каждой змейки, увеличение в случае совпадения координат головы с координатами яблока.

Два игрока должны играть с одного компьютера, они взаимодействуют между собой с помощью клавиатуры. Правила игры немного отличаются от классической Змейки. PvP Змейка представляет собой игру, в которой каждый из игроков управляет своей змейкой, которая может двигаться в одном из 4 направлений и растет на одну единицу длины, съедая объект на поле.

Правила:

- выигрывает игрок, который попадает в змейку оппонента сбоку головной частью;
- при столкновении лоб в лоб — ничья;
- при выходе за границы поля, игрок проигрывает.

Движение:

- изменить направление движения за один ход игрок может в 3 из 4 сторон, то есть кроме противоположному текущему;
- по умолчанию змейка перемещается в заданном направлении на одну клетку за один такт;
- скорость регулируется частотой обновления картинки.

Реализован класс Game, в конструкторе которого задается размер поля, скорость игры, змейки. Каждая змейка - экземпляр класса Snake, который реализует методы перемещения змейки и смены направления движения, хранит длину змейки, массив координат ее частей (каждая n-ная часть змейки содержит n пар координат x,y), направление движения. Класс Game реализует методы начала, паузы игры, окончания игры, запуска движения змеек, проверки на поражения и на наличие яблок каждый такт с частотой, зависящей от скорости игры, также реализует методы генерации яблок, их удаления, методы привязывающие обработчики на события нажатий клавиатуры, хранит координаты яблок. При начале игры навешиваются обработчики события нажатия клавиатуры, чтобы пользователи могли изменять направления движения змеек. Начало игры продемонстрировано на рисунке 1:

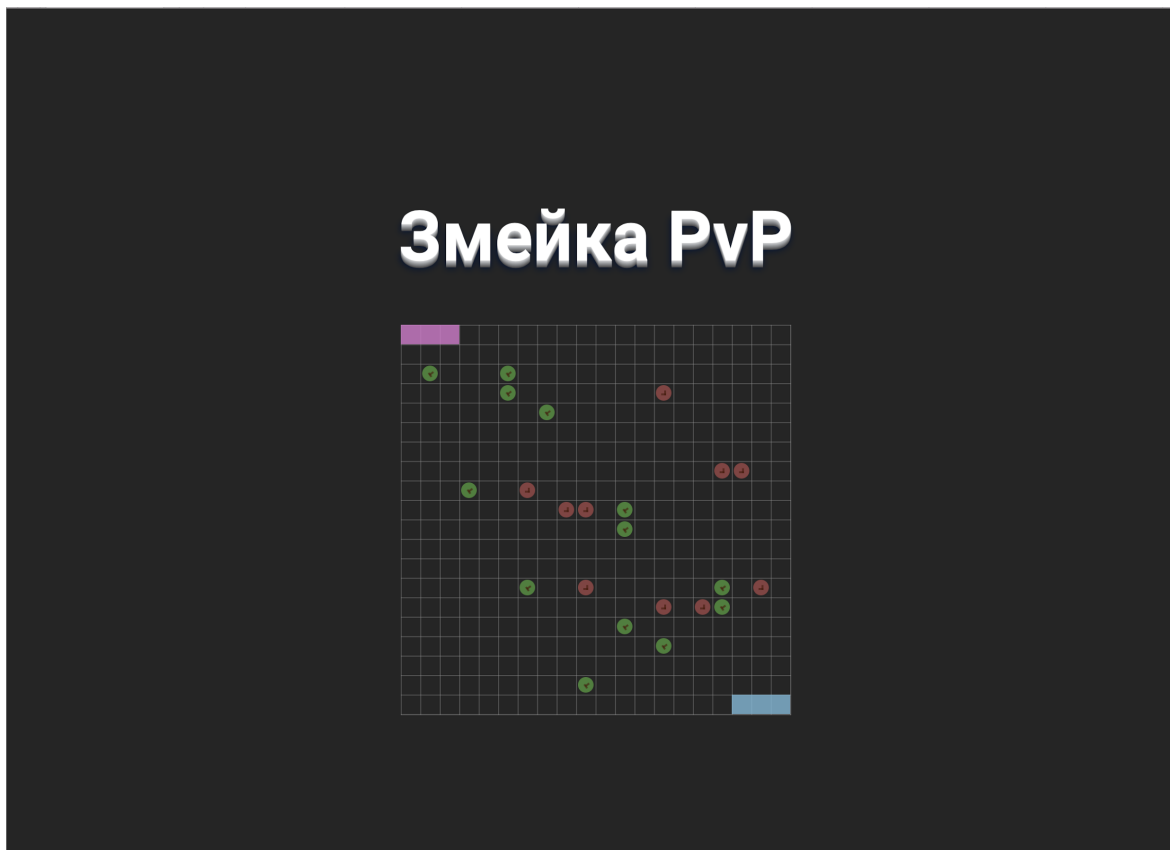


Рисунок 1 – Начало игры

Проверка победы/поражения каждого игрока осуществляется в каждом такте игры, запускаемом специальным методом класса Game путем попарного сравнения координат всех частей змеек и сравнения их головных частей с границами поля (не превышают ли размер поля и не меньше ли 0). При завершении игры появляется модальное окно с результатами “боя” (рисунок 2), кнопка “Restart” позволяет продолжить игру снова. Также реализован класс Apple, в котором есть методы постановки яблока на поле и удаления, при поедании змейкой. Класс SnakeRenderer отвечает за отрисовку змейки в соответствии с ее координатами, реализованы методы вставки змейки на поле - генерация ее html, удаления змейки и увеличения на 1 единицу длины.

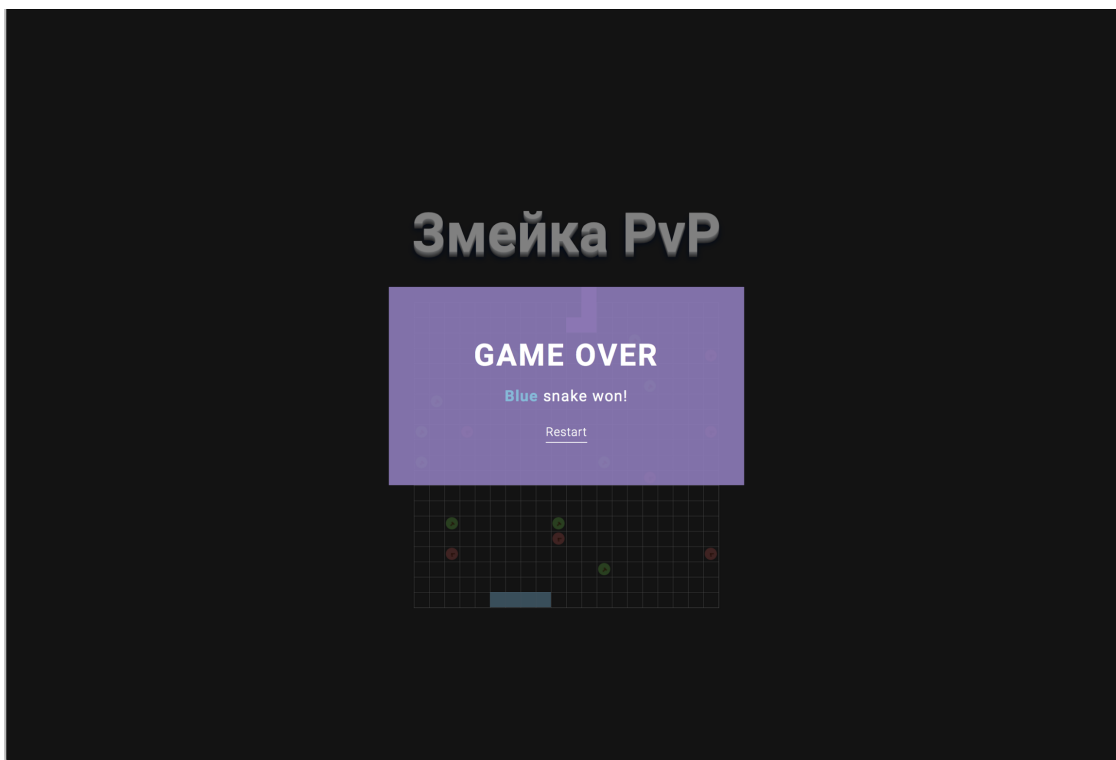


Рисунок 2 – Модальное окно с результатом игры - победа Синей змейки

В случае столкновения головами в модальном окне будет надпись о ничье как на рисунке 3:

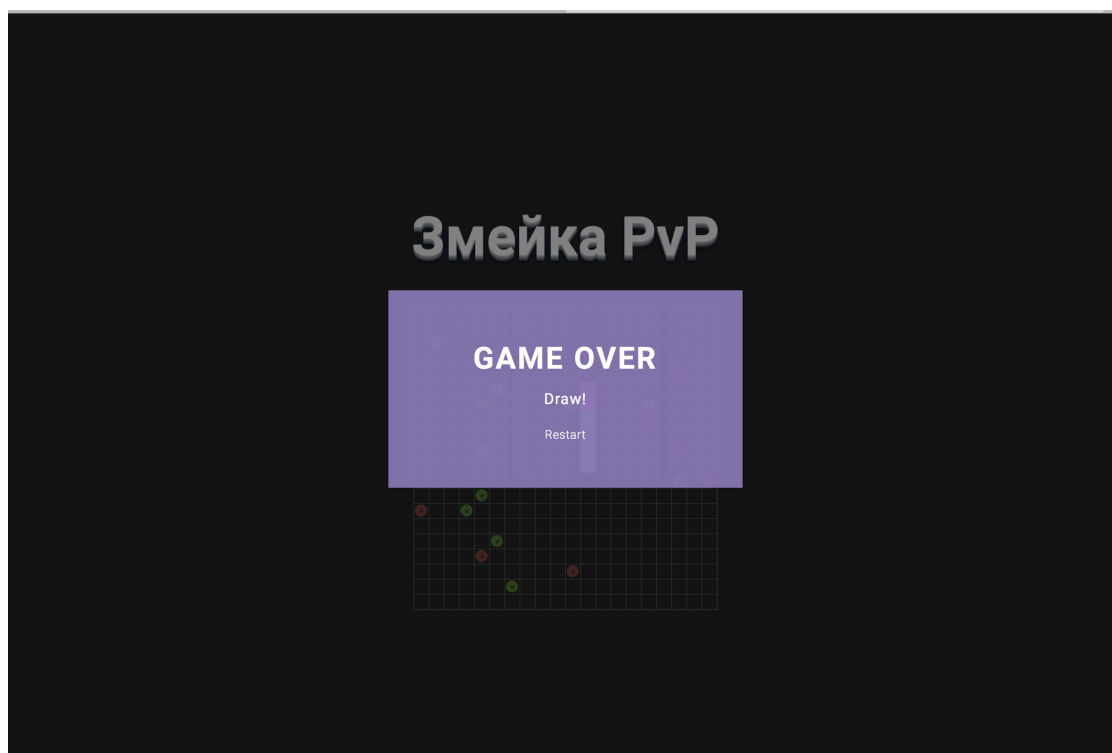


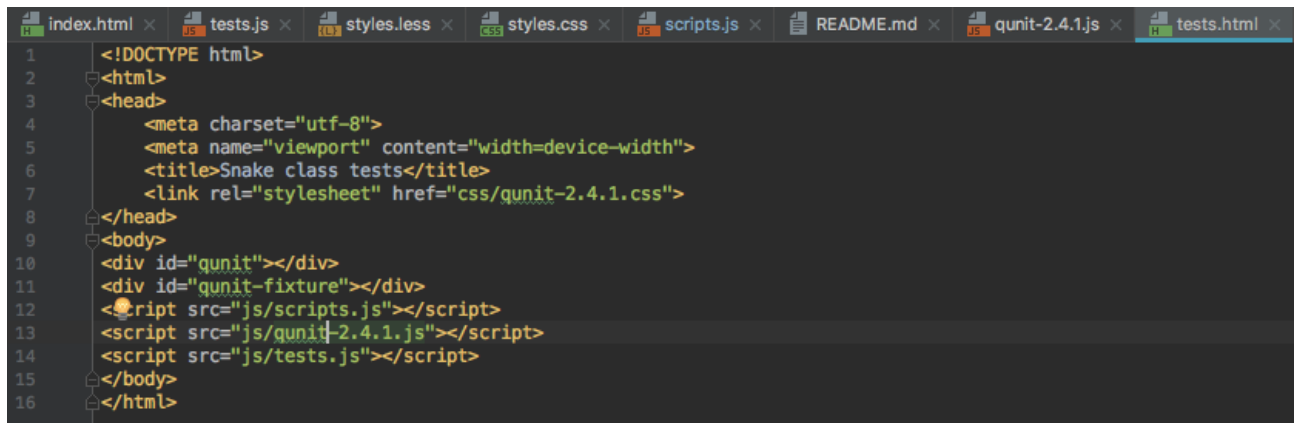
Рисунок 3 – Модальное окно с результатом игры – ничья

В процессе игры змейки вынужденно поедают яблоки и становятся длиннее, чем подвергают себя большей опасности быть съеденными оппонентом, пример отъевшихся змеек на рисунке 4:



Рисунок 4 – Змейки, съевшие несколько яблок

Написаны unit-тесты для класса Snake, позволяющие проверить правильную работу методов класса, а именно: Snake.init(), Snake.move(), Snake.changeDirection(), Snake.eatApple(), Snake.destroy(). Для написания тестов использовалась библиотека QUnit, которая является оптимальной рабочей средой для тестирования кода JavaScript. Данная библиотека подключается путем создания HTML документа tests.html (рисунок 5) с вставленными файлами QUnit.js и QUnit.css.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>Snake class tests</title>
7   <link rel="stylesheet" href="css/qunit-2.4.1.css">
8 </head>
9 <body>
10  <div id="qunit"></div>
11  <div id="qunit-fixture"></div>
12  <script src="js/scripts.js"></script>
13  <script src="js/qunit-2.4.1.js"></script>
14  <script src="js/tests.js"></script>
15 </body>
16 </html>
```

Рисунок 5 – Документ tests.html

В тестах проверка осуществляется с помощью утверждения сравнения `equals()`, оно предполагает, что первый параметр (который является действительным значением) эквивалентен второму параметру (который является ожидаемым значением), также используется и `deepEquals()` для сравнения объектов и массивов. Для запуска тестов, необходимо открыть HTML документ `tests.html` в браузере. Результат выполнения тестов представлен на рисунке 6:

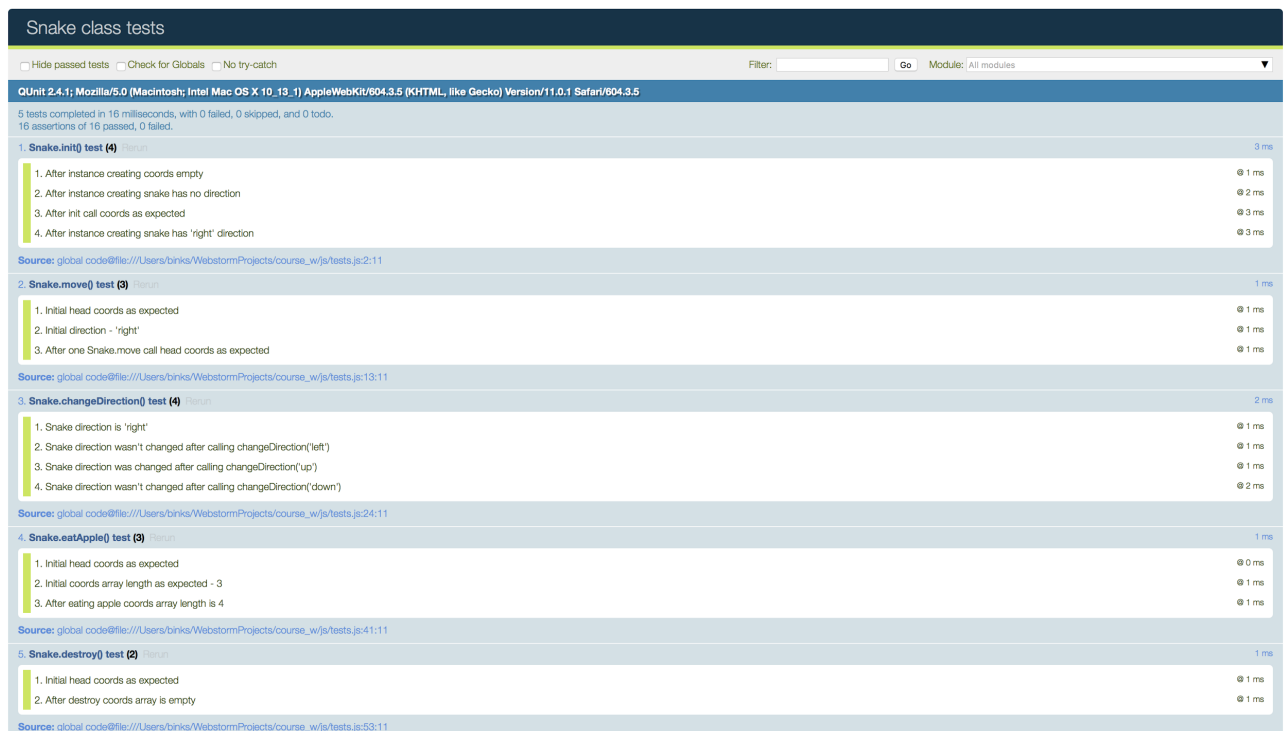


Рисунок 6 – Результаты тестов

2 ВЫБОР ТЕХНОЛОГИЙ

2.1 Выбор языка программирования

С целью написания кроссплатформенного легковесного приложения, не требующего установки, оптимальным языком программирования будет являться JavaScript. Для написания приложения с использованием данного языка программирования, необходимы также язык разметки веб-страниц - HTML и формальный язык описания внешнего вида страницы - CSS.

JavaScript – язык программирования, использующийся для придания интерактивности веб-страницам. Поддерживает объектно-ориентированный, функциональный и императивный стили. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. Структурно JavaScript можно представить в виде объединения трех четко различимых друг от друга частей: ядро (ECMAScript), объектная модель браузера (BOM), объектная модель документа (DOM). Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться, поэтому иногда их рассматривают как отдельные от JavaScript сущности. Данный язык позволяет писать кроссплатформенные приложения, не требующие установки, они могут быть залиты на веб-сервер и доступны с любого браузера на любом устройстве.

HTML – стандартизованный язык разметки документов в интернете, основанный на структурных и семантических элементах – дескрипторах, именуемых «тегами», у каждого из которых есть определенное назначение и атрибуты.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки, с помощью него интерфейс модуля приобретет пригодный для использования вид. В данном формальном языке существуют так называемые «селекторы», которые позволяют выбрать

элементы разметки страницы, которым будут заданы определенные в фигурных скобках свойства оформления, такие как позиция, размер, цвет и так далее.

2.2 Выбор используемых библиотек

Проект написан на чистом языке JavaScript, без каких-либо библиотек в целях обеспечения требуемой легковесности приложения.

Для unit-тестов использовалась библиотека QUnit. Это рабочая среда для модульного тестирования JavaScript, которая существенно помогает отлаживать код. Она разработана командой jQuery и отлично подходит для тестирования любого кода JavaScript.

3 ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ

В процессе разработки и тестирования приложения выявлялись различные баги.

Один из багов был связан с открытием модального окна с результатами игры при слишком скором проигрыше после рестарта. Проблема крылась в том, что первое модальное окно о результатах не успевало исчезнуть из DOM дерева (так как оно ждет завершения анимации прежде чем удалиться), как уже пыталось внедриться новое, в результате новое не получало класс для визуального отображения и зависало в скрытом виде. Решением было добавление проверки на наличие модального окна при открытии нового, и в случае обнаружения открытого - его немедленное удаление, а после вставка нового.

Также в процессе написания unit-тестов был выявлен баг зацикливающейся отрисовки из-за асинхронности выполнения метода draw класса Game. Данный баг был исправлен путем внедрения проверки флага stopped экземпляра класса Game внутри функции, которая вызывается с задержкой в n миллисекунд, которые зависят от установленной скорости игры и обеспечивают задержку между итерациями игры.

Последней, и самой серьезной, проблемой стала непригодность кода класса Snake к unit-тестированию из-за наличия методов работы с DOM деревом. Для решения данной проблемы был реализован класс SnakeRenderer, который взял на себя все функции изменения DOM дерева для визуализации перемещений и приращений змеек.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была реализована браузерная игра Змейка PvP, отвечающая заранее определенным требованиям совместимости с различными операционными системами и браузерами, легковесности и скорости анимации.

Дальнейшие пути развития проекта:

- добавление препятствий;
- случайное размещение змеек при рестарте;
- добавление элементов управления настройками игры, такими как скорость, размер поля, сложность (максимальное число яблок на поле);
- реализация возможности управления мульти-сенсорными устройствами (отлавливание событий движения пальцев по верхней половине поля для левой верхней змейки и нижней для правой нижней змейки).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы JavaScript [Электронный ресурс] / Илья Кантор, 2015, Режим доступа: <https://learn.javascript.ru/>, свободный (дата обращения 06.10.2017).
2. Документация QUnit [Электронный ресурс] / JQuery team, 2015, Режим доступа: <http://qunitjs.com>, свободный (дата обращения 14.11.2017).
3. JavaScript. Подробное руководство. [Электронный ресурс] / Дэвид Флэнаган, 2008, Режим доступа: <http://kharchuk.ru/JavaScript.pdf>, свободный (дата обращения 11.10.2017).
4. CSS. Каскадные таблицы стилей. Подробное руководство. [Электронный ресурс] / Эрик Мейер, 2008, Режим доступа: <https://www.books.ru/books/csskaskadnye-tablitsy-stilei-podrobnoe-rukovodstvo-3-e-izdanie-fail-pdf-614054/download/?type=demo&usg=AOvVaw1hgUddutvMsUdtlrctsT3Q>, свободный (дата обращения 06.10.2017).