Федеральное государственное автономное образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
В.В. Шайдуров
«14» 06 2016 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

МОДЕЛИРОВАНИЕ КОНЕЧНЫХ АВТОМАТОВ

Научный руководитель
кандидат физико-математических наук 14.06.16 С.Н. Баранов

Выпускник 14.06.16 А.С. Шарафутдинов

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы:

1. Изучить конечные автоматы,
2. Рассмотреть клеточные автоматы и игру «Жизнь» в качестве примера конечных автоматов,
3. Изучить методы работы с JavaScript,
4. Написать Web-приложение реализующий клеточный автомат, используя язык программирования JavaScript.

Применение конечных автоматов

Автоматное программирование широко применяется при построении лексических анализаторов (классические конечные автоматы) и синтаксических анализаторов (автоматы с магазинной памятью).

Кроме того, мышление в терминах конечных автоматов (то есть разбиение исполнения программы на *шаги автомата* и передача информации от шага к шагу через состояние) необходимо при построении событийно-ориентированных приложений. В этом случае программирование в стиле конечных автоматов оказывается единственной альтернативой порождению множества процессов или потоков управления.

Продуманное применение конечных автоматов облегчает организацию и сопровождение, как логики пользовательского интерфейса, так и логики приложения. Благодаря этому код будет более гибким и надежным

1 Конечные автоматы

1.1 Общие сведения

Конечный автомат — это некоторая абстрактная модель, содержащая конечное число состояний чего-либо. Используется для представления и управления потоком выполнения каких-либо команд. Конечный автомат идеально подходит для реализации искусственного интеллекта в играх, получая аккуратное решение без написания громоздкого и сложного кода.

Конечные автоматы подразделяются на детерминированные и недетерминированные.

- Детерминированным конечным автоматом (ДКА) называется такой автомат, в котором нет дуг с меткой *ε*(предложение, не содержащее ни одного символа), и из любого состояния по любому символу возможен переход в точности в одно состояние.
- Недетерминированный конечный автомат (НКА) является обобщением детерминированного. Недетерминированность автоматов может достигаться двумя способами: либо могут существовать переходы, помеченные пустой цепочкой ε, либо из одного состояния могут выходить несколько переходов, помеченных одним и тем же символом.

1.2 Виды конечных автоматов

Автомат Мили (*абстрактный автомат первого рода*) — конечный автомат, выходная последовательность которого зависит от состояния автомата и входных сигналов.

Автомат Мура (*абстрактный автомат второго рода*) — конечный автомат, выходное значение сигнала в котором зависит лишь от текущего состояния данного автомата, и не зависит напрямую, в отличие от автомата Мили, от входных значений.

2 Клеточные автоматы

2.1 Общие сведения

Для моделирования конечного автомата будет использоваться клеточный автомат.

Клеточный автомат — являются частным случаем конечных автоматов, используемых для моделирования динамического поведения однородных сред. При этом пространство и время считаются дискретными, а физические величины в каждой точке моделируемой среды могут принимать конечное множество дискретных значений. Включает регулярную решётку ячеек, каждая из которых может находиться в одном из конечного множества состояний, таких как 1 и 0. Решетка может быть любой размерности. Для каждой ячейки определено множество ячеек, называемых окрестностью. К примеру, окрестность может быть определена как все ячейки на расстоянии не более 2 от текущей (окрестность фон Неймана ранга 2). Для работы клеточного автомата требуется задание начального состояния всех ячеек, и правил перехода ячеек из одного состояния в другое. На каждой итерации, используя правила перехода и состояния соседних ячеек, определяется новое состояние каждой ячейки. Обычно правила перехода одинаковы для всех ячеек и применяются сразу ко всей решётке.

Основное направление исследования клеточных автоматов — алгоритмическая разрешимость тех или иных задач. Также рассматриваются вопросы построения начальных состояний, при которых клеточный автомат будет решать заданную задачу.

2.2 Классификация клеточных автоматов

Стивен Вольфрам предложил 4 класса, на которые все клеточные автоматы могут быть разделены в зависимости от типа их эволюции. Классификация Вольфрама была первой попыткой классифицировать сами

правила, а не типы поведения правил по отдельности. В порядке возрастания сложности классы выглядят следующим образом:

Класс 1: Результатом эволюции почти всех начальных условий является быстрая стабилизация состояния и его гомогенность. Любые случайные конструкции в таких правилах быстро исчезают.

Класс 2: Результатом эволюции почти всех начальных условий является быстрая стабилизация состояния, либо возникновение колебаний. Большинство случайных структур в начальных условиях быстро исчезает, но некоторые остаются. Локальные изменения в начальных условиях оказывают локальный характер на дальнейший ход эволюции системы.

Класс 3: Результатом эволюции почти всех начальных условий являются псевдослучайные, хаотические последовательности. Любые стабильные структуры, которые возникают и почти сразу же уничтожаются окружающим их шумом. Локальные изменения в начальных условиях оказывают широкое, неопределяемое влияние на ход всей эволюции системы.

Класс 4: Результатом эволюции почти всех правил являются структуры, которые взаимодействуют сложным и интересным образом с формированием локальных, устойчивых структур, которые способны выживать длительное время. В результате эволюции правил этого класса могут получаться некоторые последовательности Класса 2, описанного выше. Локальные изменения в начальных условиях оказывают широкое, неопределяемое влияние на ход всей эволюции системы. Некоторые клеточные автоматы этого класса обладают свойством универсальности по Тьюрингу. Последний факт был доказан для Правила 110 и игры «Жизнь».

2.3 Виды окрестностей клетки

Окрестность фон Неймана клетки (англ. *von Neumann neighborhood*) — совокупность четырёх клеток на квадратном паркете, имеющих общую сторону с данной клеткой.

Окрестность Мура клетки (англ. *Moore neighborhood*) — в двумерном случае — совокупность восьми клеток на квадратном паркете, имеющих общую вершину с данной клеткой

3 Игра «Жизнь»

3.1 Общие сведения

Игра «Жизнь» - клеточный автомат, придуманный английским математиком Джоном Конвеем в 1970 году.

Правила игры «Жизнь» (Джон Конвей):

- Место действия этой игры — «вселенная» — это размеченная на клетки поверхность или плоскость — безграничная, ограниченная, или замкнутая (в пределе — бесконечная плоскость).
- Каждая клетка на этой поверхности может находиться в двух состояниях: быть «живой» или быть «мёртвой» (пустой). Клетка имеет восемь соседей (окружающих клеток).
- Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:

1. в пустой (мёртвой) клетке, рядом с которой ровно три живые клетки, зарождается жизнь;
2. если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае (если соседей меньше двух или больше трёх) клетка умирает («от одиночества» или «от перенаселённости»)

- Игра прекращается, если на поле не останется ни одной «живой» клетки, если при очередном шаге ни одна из клеток не меняет своего состояния (складывается стабильная конфигурация) или если конфигурация на очередном шаге в точности (без сдвигов и поворотов) повторит себя же на одном из более ранних шагов (складывается периодическая конфигурация).

Эти простые правила приводят к огромному разнообразию форм, которые могут возникнуть в игре.

Игрок не принимает прямого участия в игре, а лишь расставляет или генерирует начальную конфигурацию «живых» клеток, которые затем взаимодействуют согласно правилам уже без его участия (он является наблюдателем).

Хотя игра состоит всего из двух простых правил, тем не менее она более сорока лет привлекает пристальное внимание учёных. Игра «Жизнь» и её модификации повлияли (в ряде случаев взаимно) на многие разделы таких точных наук, как математика, информатика, физика. Это, в частности:

- Теория автоматов,
- Теория алгоритмов,
- Теория игр и математическое программирование,
- Алгебра и теория чисел,
- Теория вероятностей и математическая статистика,
- Комбинаторика и теория графов,
- Фрактальная геометрия,
- Вычислительная математика,
- Теория принятия решений,
- Математическое моделирование.

Кроме того, многие закономерности, обнаруженные в игре, имеют свои аналогии в других, подчас совершенно «нематематических» дисциплинах.

Вот список наук, теории которых имеют интересные точки соприкосновения с феноменами «Жизни»:

- Кибернетика. Сама игра является удачной попыткой Конвея доказать существование простых самовоспроизводящихся систем.
- Биология. Внешнее сходство с развитием популяций примитивных организмов впечатляет.
- Физиология. Рождение и смерть клеток аналогичны процессу возникновения и исчезновения нейронных импульсов, которые и формируют процесс мышления. А также аналогичны созданию импульсов в нервной системе многоклеточных организмов.
- Астрономия. Эволюции некоторых сложных колоний удивительным образом схематично повторяют этапы развития спиралевидных галактик.
- Физика твёрдого тела. Теория автоматов вообще и игра «Жизнь» в частности используются для анализа «явлений переноса» — диффузии, вязкости и теплопроводности.
- Квантовая физика. Поведение «жизненных» ячеек (рождение новых и взаимное уничтожение) во многом напоминают процессы, происходящие при столкновении элементарных частиц.
- Наномеханика. Стационарные и пульсирующие колонии являются показательным примером простейших устройств, созданных на основе нанотехнологий.
- Электротехника. Правила игры используются для моделирования самовосстанавливающихся электрических цепей.
- Химия. Конфигурации, подобные строящимся в игре, возникают во время химических реакций на поверхности, в частности в опытах М. С. Шакаевой возникают движущиеся молекулярные конструкции аналогичные «жизненному» планеру. Также предпринимаются попытки

объяснить периодические химические реакции с помощью многомерных клеточных автоматов. Самоорганизацией элементарных частиц также занимается супрамолекулярная химия.

- Социология. Процессы доминации, вытеснения, поглощения, сосуществования, слияния и уничтожения популяций во многих аспектах схожи с явлениями, происходящими при взаимодействии больших, средних и малых социальных групп.
- Философия. Приведённый список примеров снова наводит на мысль, что всё во Вселенной развивается по одним и тем же нескольким фундаментальным законам, пока ещё не познанным человеком.

Возможно, эта игра связана и с другими научными явлениями, в том числе и с теми, о которых современной науке пока неизвестно. Также возможно, что не открытые на сегодня законы Природы и Общества станут более понятными благодаря «Жизни» и её модификациям.

3.2 Компьютерная реализация

В компьютерных реализациях игры поле ограничено и, как правило, верхняя граница поля «соединена» с нижней, а левая граница — с правой, что представляет собой эмуляцию поверхности тора, но на экране поле всегда отображается в виде равномерной сетки.

Простейший алгоритм «смены поколения» последовательно просматривает все ячейки решетки и для каждой ячейки подсчитывает соседей, определяя судьбу каждой клетки (не изменится, умрет, родится). Такой простейший алгоритм использует два двумерных массива — один для текущего поколения, второй — для следующего.

Более сложный, но и более быстрый алгоритм составляет списки клеток для просмотра в последующем поколении; клетки, которые не могут измениться, в списки не вносятся. Например, если какая-либо клетка и ни

одна из её соседей не поменялись на предыдущем ходу, то эта клетка не поменяется и на текущем ходу.

3.3 Фигуры

Вскоре после опубликования правил, было обнаружено несколько интересных шаблонов (вариантов расстановки живых клеток в первом поколении), в частности: r-пентамино и планер (glider).

Некоторые такие фигуры остаются неизменными во всех последующих поколениях, состояние других периодически повторяется, в некоторых случаях со смещением всей фигуры. Существует фигура (Diehard) всего из семи живых клеток, потомки которой существуют в течение 130 поколений, а затем исчезают.

Конвей первоначально предположил, что никакая начальная комбинация не может привести к неограниченному размножению и предложил премию в 50 долларов тому, кто докажет или опровергнет эту гипотезу. Приз был получен группой из Массачусетского технологического института, придумавшей неподвижную повторяющуюся фигуру, которая периодически создавала движущиеся «планеры». Таким образом, количество живых клеток могло расти неограниченно. Затем были найдены движущиеся фигуры, оставляющие за собой «мусор» из других фигур.

К настоящему времени более-менее сложилась следующая классификация фигур:

1. **Устойчивые фигуры:** фигуры, которые остаются неизменными.
2. **Мафусаилы:** фигуры, которые долго меняются, прежде чем стабилизироваться.
3. **Периодические фигуры:** фигуры, у которых состояние повторяется через некоторое число поколений.
4. **Двигающиеся фигуры:** фигуры, у которых состояние повторяется, но с некоторым смещением.

5. **Ружья:** фигуры, у которых состояние повторяется, но дополнительно появляется двигающаяся фигура.

6. **Паровозы:** двигающиеся фигуры, которые оставляют за собой следы в виде устойчивых или периодических фигур.

7. **Пожиратели:** устойчивые фигуры, которые могут пережить столкновения с некоторыми двигающимися фигурами.

8. **Сорняки:** фигуры, которые при столкновении с некоторыми фигурами дублируются.

4 JavaScript

4.1 Общие сведения

JavaScript – это интерпретируемый язык программирования с объектно-ориентированными возможностями. С точки зрения синтаксиса базовый язык JavaScript напоминает C, C++ и Java такими программными конструкциями, как инструкция if, цикл while и оператор &&. Однако это подобие ограничивается синтаксической схожестью. JavaScript – это не типизированный язык, т. е. в нем не требуется определять типы переменных. Объекты в JavaScript отображают имена свойств на произвольные значения. Этим они больше напоминают ассоциативные массивы Perl, чем структуры C или объекты C++ или Java. Механизм объектно-ориентированного наследования JavaScript скорее похож на механизм прототипов в таких малоизвестных языках, как Self, и сильно отличается от механизма наследования в C++ и Java. Как и Perl, JavaScript – это интерпретируемый язык, и некоторые его инструменты, например регулярные выражения и средства работы с массивами, реализованы по образу и подобию языка Perl. Ядро языка JavaScript поддерживает работу с такими простыми типами данных, как числа, строки и булевы значения. Помимо этого он обладает встроенной поддержкой массивов, дат и объектов регулярных выражений. Обычно JavaScript применяется в web-браузерах, а расширение его

возможностей за счет введения объектов позволяет организовать взаимодействие с пользователем, управлять web-браузером и изменять содержимое документа, отображаемое в пределах окна web-браузера. Эта встроенная версия JavaScript запускает сценарии, внедренные в HTML-код web-страниц. Как правило, эта версия называется *клиентским* языком JavaScript, чтобы подчеркнуть, что сценарий исполняется на клиентском компьютере, а не на web-сервере. В основе языка JavaScript и поддерживаемых им типов данных лежат международные стандарты, благодаря чему обеспечивается прекрасная совместимость между реализациями. Некоторые части клиентского JavaScript формально стандартизированы, другие части стали стандартом де-факто, но есть части, которые являются специфическими расширениями конкретной версии браузера. Совместимость реализаций JavaScript в разных браузерах зачастую приносит немало беспокойства программистам, использующим клиентский язык JavaScript.

На сегодняшний день web-ресурсы являются важным аспектом жизни каждого человека и JavaScript неотъемлемая их часть.

4.2 JavaScript – это не Java

Одно из наиболее распространенных заблуждений о JavaScript состоит в том, что этот язык представляет собой упрощенную версию Java, языка программирования, разработанного в компании Sun Microsystems. Кроме некоторой синтаксической схожести и способности предоставлять исполняемое содержимое для web-браузеров, эти два языка между собой ничто не связывает. Схожесть имен – не более чем уловка маркетологов (первоначальное название языка – LiveScript – было изменено на JavaScript в последнюю минуту). Однако JavaScript и Java могут взаимодействовать друг с другом.

4.3. JavaScript не простой язык

Поскольку JavaScript является интерпретируемым языком, очень часто он позиционируется как язык сценариев, а не как язык программирования, при этом подразумевается, что языки сценариев проще и в большей степени ориентированы не на программистов, а на обычных пользователей. В самом деле, при отсутствии контроля типов JavaScript прощает многие ошибки, которые допускают неопытные программисты. Благодаря этому многие web-дизайнеры могут использовать JavaScript для решения ограниченного круга задач, выполняемых по точным рецептам.

Однако за внешней простотой JavaScript скрывается полноценный язык программирования, столь же сложный, как любой другой, и даже более сложный, чем некоторые. Программисты, пытающиеся решать с помощью JavaScript не тривиальные задачи, часто разочаровываются в процессе разработки из-за того, что недостаточно понимают возможности этого языка.

4.4 Клиентский JavaScript

Когда интерпретатор JavaScript встраивается в web-браузер, результатом является клиентский JavaScript. Это, безусловно, наиболее распространенный вариант JavaScript, и большинство людей, упоминая JavaScript, обычно подразумевают именно клиентский JavaScript. В этой книге клиентский язык JavaScript описывается вместе с базовым JavaScript, который представляет собой подмножество клиентского JavaScript.

Клиентский JavaScript включает в себя интерпретатор JavaScript и объектную модель документа (Document Object Model, DOM), определяемую web-браузером. Документы могут содержать JavaScript-сценарии, которые в свою очередь могут использовать модель DOM для модификации документа или управления способом его отображения. Другими словами, можно сказать, что клиентский JavaScript позволяет определить поведение статического содержимого web-страниц. Клиентский JavaScript является

основой таких технологий разработки web-приложений, как DHTML, и таких архитектур, как Ajax.

Спецификация ECMA_262 определила стандартную версию базового языка JavaScript, а организация World Wide Web Consortium (W3C) опубликовала спецификацию DOM, стандартизующую возможности, которые браузер должен поддерживать в своей объектной модели. Основные положения стандарта W3C DOM достаточно полно поддерживаются наиболее распространенными браузерами за одним важным исключением – Microsoft Internet Explorer; в этом браузере отсутствует поддержка механизма обработки событий.

JavaScript обеспечивает возможность управления не только содержимым HTML-документов, но и их поведением. Другими словами, JavaScript-программа может реагировать на действия пользователя: ввод значения в текстовое поле или щелчок мышью в области изображения в документе. Это достигается путем определения *обработчиков событий* для документа – фрагментов JavaScript-кода, исполняемых при возникновении определенного события, например щелчка на кнопке.

5 HTML

HTML (от англ. *Hyper Text Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство web-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879.

Язык XHTML является более строгим вариантом HTML, он следует всем ограничениям XML и, фактически, XHTML можно воспринимать как приложение языка XML к области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

6 CSS

CSS (англ. *Cascading Style Sheets — каскадные таблицы стилей*) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида web-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

7 Реализация

7.1 Используемые теги

Для написания код были использованы следующие теги:

- элемент является универсальным блочным элементом и предназначен для группирования элементов документа с целью изменения вида содержимого через стили. Для этого добавляется атрибут class или id с именем класса или идентификатора.
- <input></input> является одним из разносторонних элементов формы и позволяет создавать разные части интерфейса и обеспечивать взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. Хотя элемент <input> не требуется помещать внутрь контейнера <form>, определяющего форму, но если введенные пользователем данные должны

быть отправлены на сервер, где их обрабатывает серверная программа, то указывать <form> обязательно. То же самое обстоит и в случае обработки данных с помощью клиентских приложений, например, скриптов на языке JavaScript.

Основной атрибут <input>, определяющий вид элемента — type. Он позволяет задавать следующие элементы формы: текстовое поле (text), поле с паролем (password), переключатель (radio), флажок (checkbox), скрытое поле (hidden), кнопка (button), кнопка для отправки формы (submit), кнопка для очистки формы (reset), поле для отправки файла (file), кнопка с изображением (image) и др. Для каждого элемента существует свой список атрибутов, которые определяют его вид и характеристики. Кроме того, в HTML5 добавлено еще более десятка новых элементов.

- <select></select> позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором. Конечный вид зависит от использования атрибута size, который устанавливает высоту списка. Ширина списка определяется самым широким текстом, указанным в элементе <option>, а также может изменяться с помощью стилей. Каждый пункт создаётся с помощью элемента <option>, который должен быть вложен в контейнер <select>. Если планируется отправлять данные списка на сервер, то требуется поместить<select> внутрь формы. Это также необходимо, когда к данным списка идёт обращение через скрипты.

- <canvas></canvas> создает область, в которой при помощи JavaScript можно рисовать разные объекты, выводить изображения, трансформировать их и менять свойства. При помощи элемента <canvas> можно создавать рисунки, анимацию, игры и др.

7.2 Построение

Для создания и отладки web-приложения нам понадобиться сервер. Чтобы запустить виртуальный сервер будем использовать приложение Denwer (www.denwer.ru). Написание кода программы не требует дополнительных приложений, достаточно текстового редактора (блокнот, Notepad++ или др.).

Файл index.php будет запускаться браузером автоматически. В файле подключаются дополнительные библиотеки и файлы, делается первоначальный шаблон страницы с использованием HTML-разметки и вызывается функция JavaScript (см. приложение 1).

Файлы с кодом JavaScript хранятся в каталоге javascript и имеют формат js.

Файлы с кодом стилей хранятся в каталоге css и имеют формат css (см.приложения 5 и 6).

Все рисунки хранятся в каталоге image (см. приложение 7).

Функция game_life(id) в качестве аргумента принимает id тега, в котором будет запускаться приложение. Эта функция создает стартовую HTML-разметку и запускает обработчик событий мыши (см. приложение 2).

Функция game_field_start(arr) в качестве аргумента принимает массив, хранящий значения поля. Эта функция формирует поле (см. приложение 2).

Функция lib_ex() устанавливает стартовые значения поля и панели инструментов (см. приложение 3).

Функция play_click() срабатывает после клика по кнопке play с определенным интервалом и обеспечивает смену поколений. Правила смены поколений зависят от установленных значений в панели инструментов (см. приложение 2).

Функция plot_create(id,step_x,step_y) в качестве аргументов принимает id тега canvas, и длину шага по осям X и Y. Эта функция рисует разметку графика (см. приложение 4).

Функция plot_draw(id,step,arr,color) в качестве аргументов принимает id тега canvas, номер поколения (шага), массив со значениями и цвет популяции (black или red). Эта функция рисует график от предыдущего поколения к текущему (см. приложение 4).

Функция plot_draw2(id,step,arr,color) в качестве аргументов принимает id тега canvas, номер поколения (шага), массив со значениями и цвет популяции (black или red). Эта функция рисует график от начала до текущего поколения (см. приложение 4).

Также в коде приложения присутствуют и другие функции несущий вспомогательный характер.

8 Интерфейс

8.1 Web-приложение

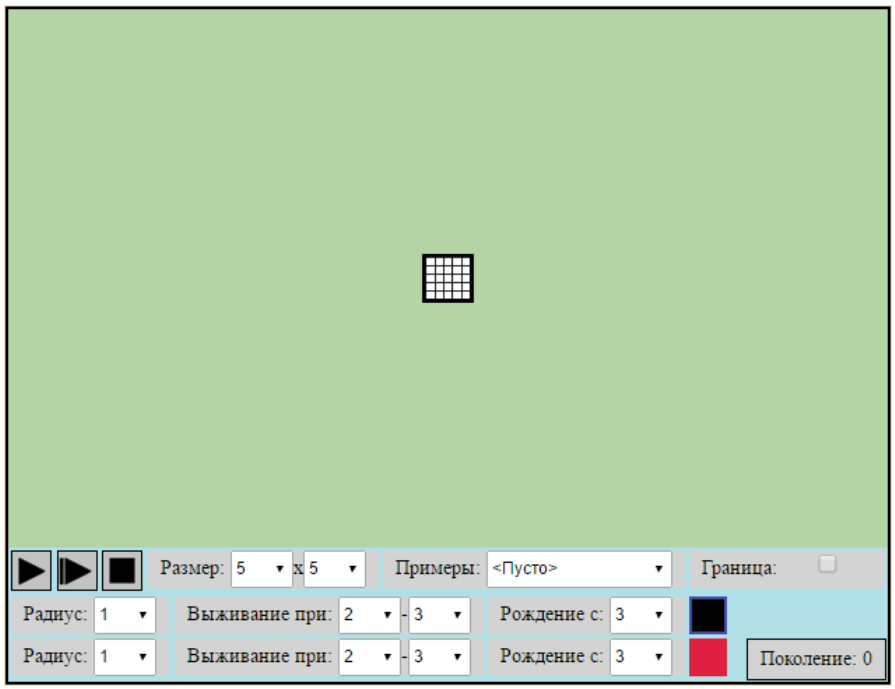

Рисунок 1- внешний вид web-приложения

Данное web-приложение выглядит как окно, имеющее рабочее поле и панель инструментов.

В центре рабочего поля располагается поле игры «Жизнь». Характеристики поля игры «Жизнь» могут меняться в зависимости от состояния панели инструментов. При открытии web-страницы поле игры «Жизнь» не заполнено, имеет размер 5 на 5 клеток, граница выключена, условия для смены поколений стандартные.

8.2 Панель инструментов

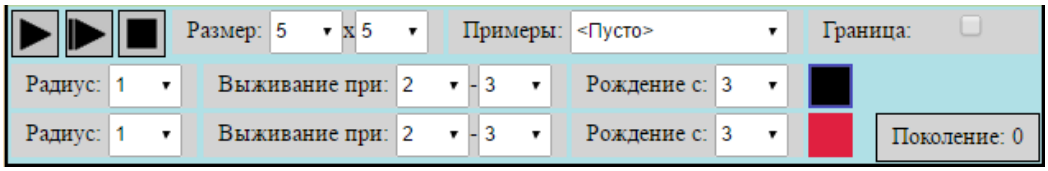

Рисунок 2 - Панель инструментов

В верхнем левом углу располагаются кнопки управления.

Рисунок 3 - кнопка play

Кнопка play запускает беспрерывную смену поколений, при этом она меняется на кнопку pause.

Рисунок 4 - кнопка pause

Кнопка pause при запуске страницы отсутствует, он появляется вместо кнопки play. Кнопка предназначена для временной приостановки смены поколений. При нажатии заменяется кнопкой play.

Рисунок 5 - кнопка step

Кнопка step делает однократную смену поколений. Если в момент клика смена поколений активна, то действие копки ее прервет.

Рисунок 6 - кнопка stop

Кнопка stop прерывает смену поколений, сбрасывает поле и счетчик поколений к начальным значениям.

Далее идут инструменты регулирующие параметры поля.

Рисунок 7 - инструмент "Размер"

Инструмент «Размер» может изменить размер поля. Минимальные значения поля – 5x5, максимальные – 60x100. Размер поля можно изменить, только если в инструменте «Примеры» стоит значение <Пусто>.

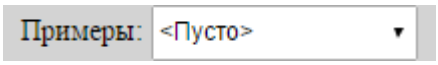

Рисунок 8 - инструмент "Примеры"

Инструмент «Примеры» изначально имеет значение <Пусто>. С помощью этого инструмента можно выставить на поле некоторые фигуры.

Рисунок 9 - инструмент "Граница"

Инструмент «Граница» отвечает за поведение популяции при достижении границы. По умолчанию граница выключена, и игра «Жизнь» проходит на неограниченном поле (поверхность тора).

В нашей игре можно установить сразу две популяции, которые существуют независимо, но ограничивают свободное пространство для размножения. Во 2 и 3 строке панели инструментов ставят условия смены поколений для каждой популяции.

Рисунок 10 - инструмент "Радиус"

Инструмент «Радиус» может принимать значение 1 или 2. Указывает, на каком расстоянии от текущей клетки искать представителей своей популяции.

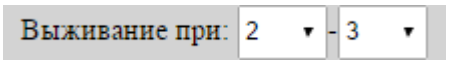

Рисунок 11 - инструмент "Выживание"

Инструмент «Выживание» указывает, при каком количестве соседей в занятой клетке сохраняется жизнь. Минимальное значение не может превышать максимальное, максимальное зависит от значения инструмента «Радиус». По умолчанию значения установлены на 2-3, что соответствует стандартным условиям игры «Жизнь».

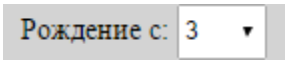

Рисунок 12 - инструмент "Рождение"

Инструмент «Рождение» указывает, начиная с какого количества соседей, в пустой клетке, будет рождаться новая жизнь. Значение этого инструмента не может быть меньше минимального и больше максимального значения инструмента «Выживание». По умолчанию установлено на 3.

Рисунок 13 - иструмент выбора цвета

Инструмент выбора активного цвета. Используется для установки на поле активных клеток популяции, в соответствии с выбранным цветом. При

клике на неактивный цвет выделение переходит на него. По умолчанию выбран «черный» цвет.

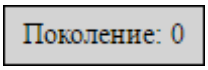

Рисунок 14 - счетчик

Счетчик поколений, показывает, сколько смен поколений произошло. Счетчик находится в нижнем правом углу.

8.3 Поле

Рисунок 15 - внешний вид поля (5x5)

Поле имеет по умолчанию размер 5x5, который можно изменить с помощью инструмента «Размер». При наведении на клетку срабатывает подсветка. При нажатии на клетку она меняет цвет, который зависит от выделенного цвета в панели инструментов.

8.4 График

График отображает численность популяций в зависимости от количества поколений. График автоматически масштабируется, если значения превышают текущий диапазон, как по горизонтали, так и по вертикали.

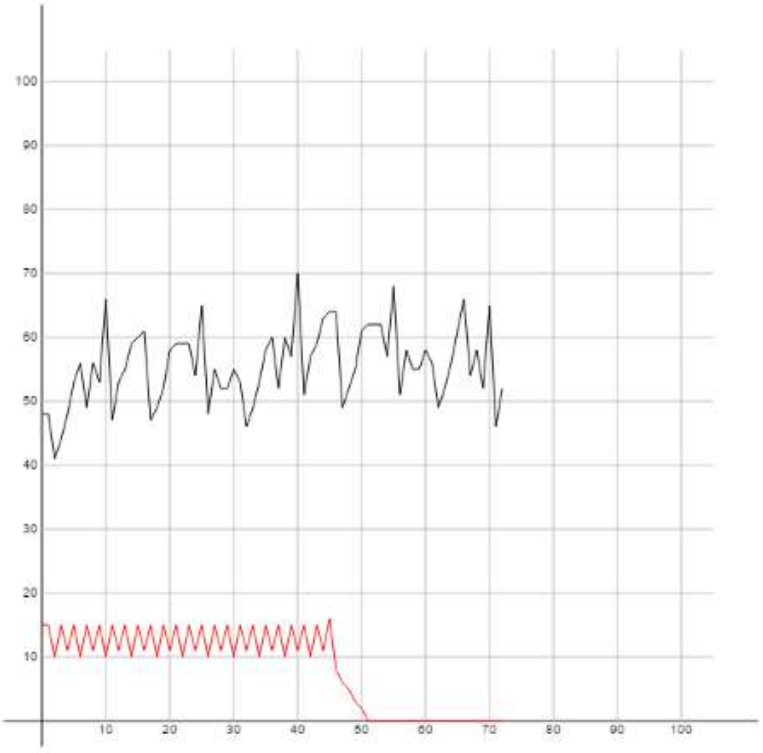

Рисунок 16 - график (с примером работы)

8.5 Структура web-приложения

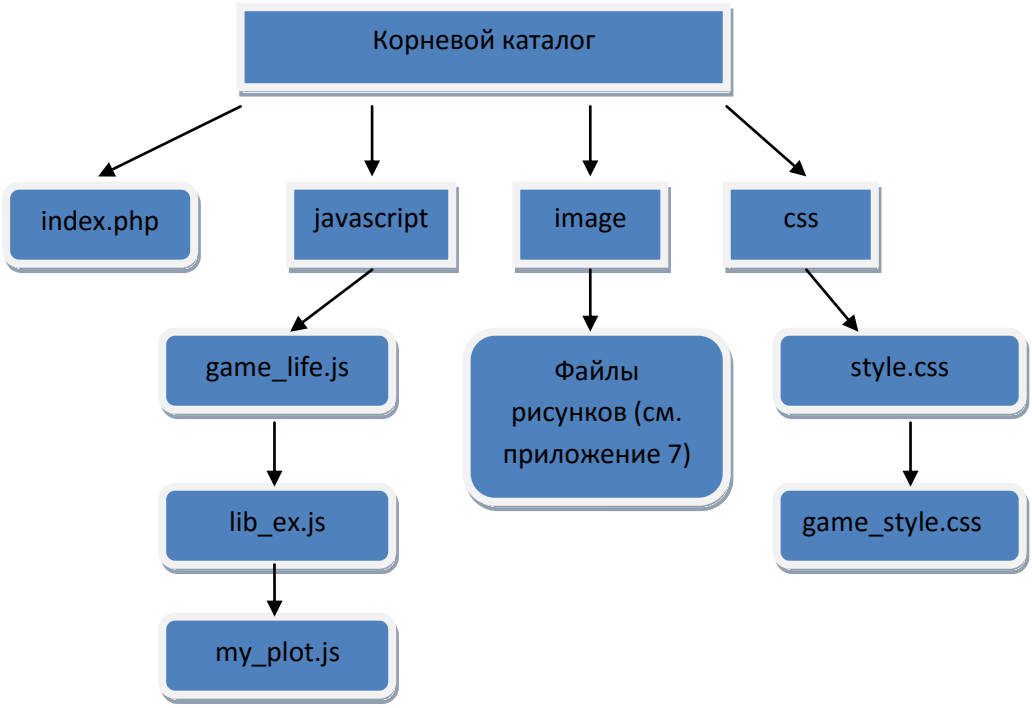

Рисунок 17 - блок схема программы

9. Примеры работы

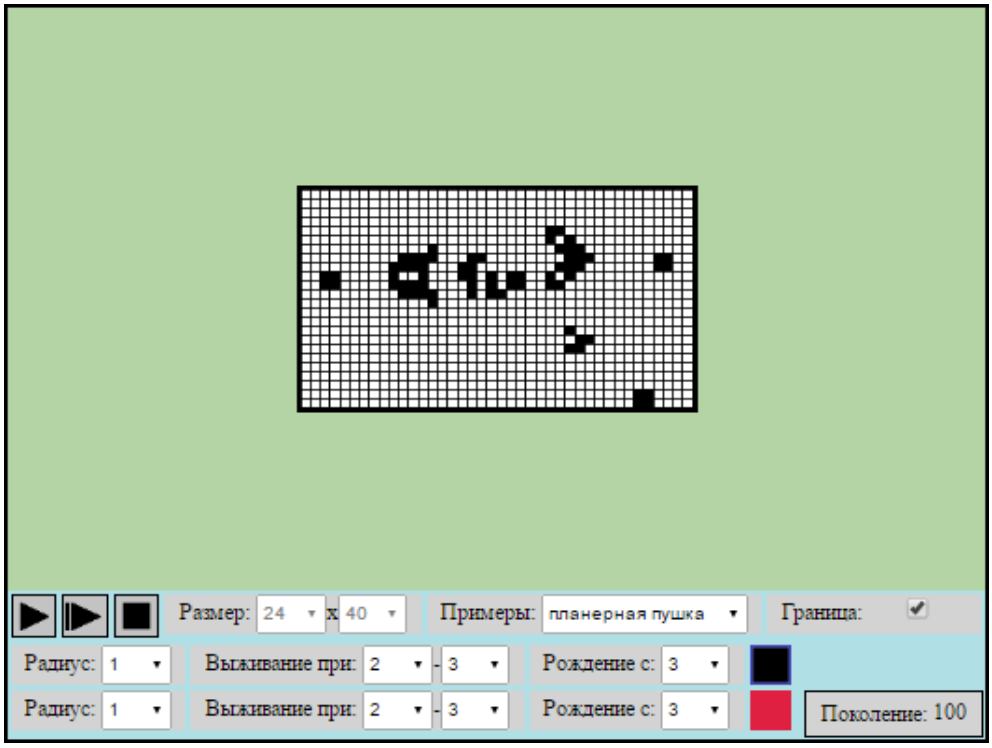

Рисунок 18 - планерная пушка (приложение)

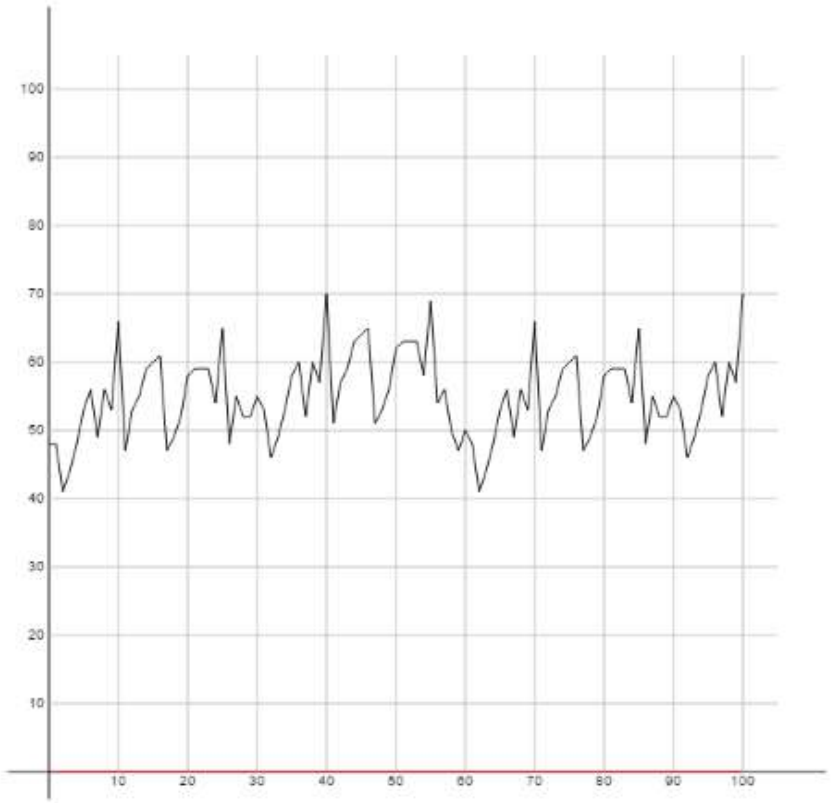

Рисунок 19 - планерная пушка (график)

На рисунках 18 и 19 изображены работа приложения и соответствующий ей график. В качестве примера были взяты настройки соответствующие игре «Жизнь» и фигура «планерная пушка». На графике видно что цикл повторяется каждые 60 поколений.

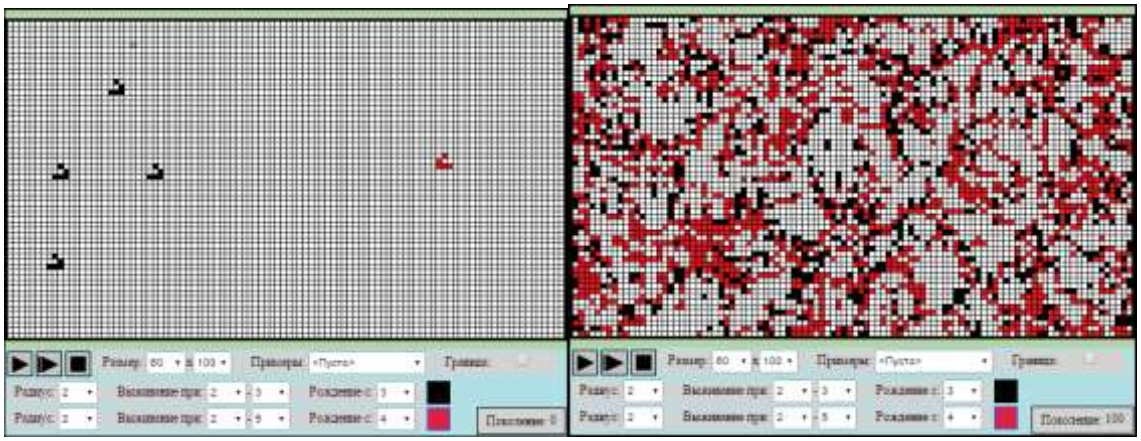

Рисунок 20 -две популяции (слева - поколение 0, справа - поколение 100)

На рисунке 20 изображен пример с двумя популяциями. Условия смены поколений разные. Обе популяции имеют быстрый рост, при заполнении всего поля сохраняют свою численность.

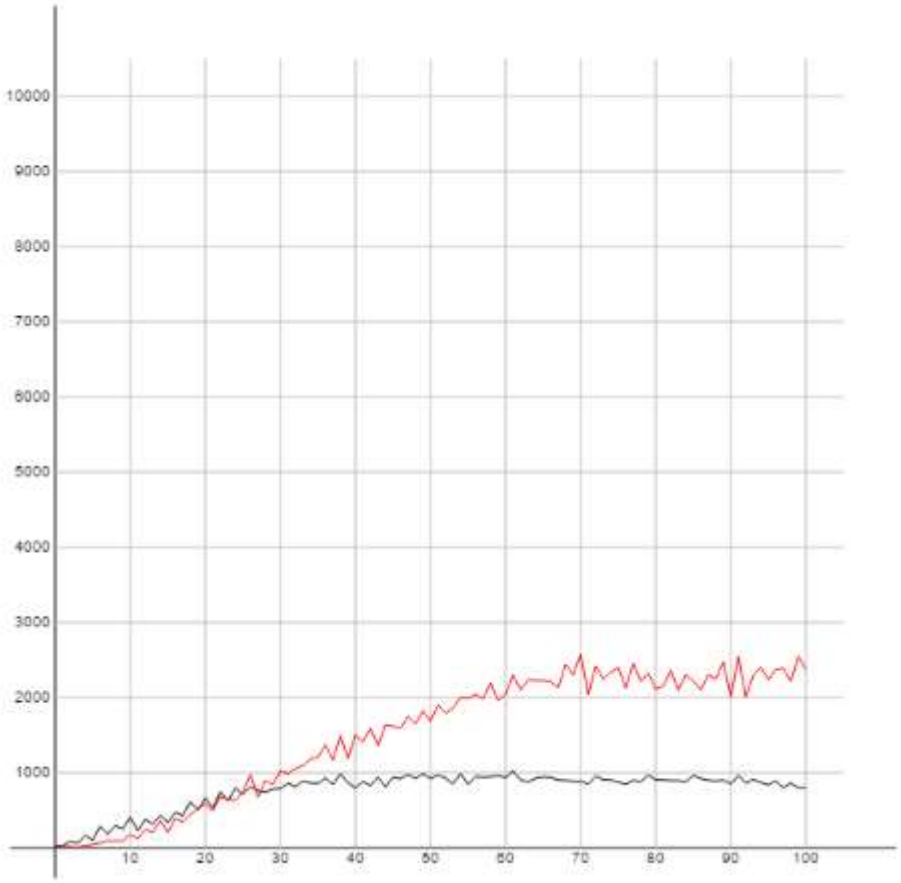

Рисунок 21 - две популяции (график)

На рисунке 21 изображен график соответствующий рисунку 20. По графику видно, что обе популяции стабильно поддерживают свою численность, при этом численность красной популяции более чем в два раза превосходит численность черной популяции.

ЗАКЛЮЧЕНИЕ

В процессе работы были получены следующие результаты:

1. Была рассмотрена и изучена стандартная модель Джона Конвея.
2. Придуманы способ настройки смены поколения двух популяций.
3. Были изучены методы работы с JavaScript.
4. При помощи средств JavaScript было разработано приложение, реализующее простой интерфейс управления клетками и полем.
5. При помощи средств JavaScript был написан скрипт, рисующий график.

Данная модель может применяться для ознакомления с разнообразием клеточных автоматов, а так же для развития теоретического аппарата клеточных автоматов с различными условиями смены поколений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Клумова, И.Н. Игра «Жизнь» / И.Н. Клумова // Квант. - 1974. - № 9. - С. 26–30.
2. Тоффоли, Т. Машины клеточных автоматов / Т. Тоффоли, Н. Марголус. - М.: Мир, 1991. – 283 с.
3. Флэнаган, Д. JavaScript. Подробное руководство/Д. Флэнаган. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 992 с., ил.
4. The Game of Life [Электронный ресурс] – Режим доступа: http://life.written.ru.
5. Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.or.
6. WebReference.ru [Электронный ресурс] – Режим доступа: https://webref.ru.
7. JavaScript.ru [Электронный ресурс] – Режим доступа: http://javascript.ru.
8. Хабрахабр [Электронный ресурс] – Режим доступа: https://habrahabr.ru/

ПРИЛОЖЕНИЕ А

Файл index.php являющийся шаблоном web-страницы который браузер запускает по умолчанию

```
<html>
	<head>
		<link type="text/css" href="css/game_style.css" rel="stylesheet">
		<link type="text/css" href="css/style.css" rel="stylesheet">
		<script type="text/javascript" src="javascript/lib_ex.js"></script>
		<script type="text/javascript" src="javascript/game_life.js"></script>
		<script type="text/javascript" src="javascript/my_plot.js"></script>
	</head>
	<body class="body" style="background: wheat">
		<h2 class="center_block">Клеточный автомат</h2>
		<div id="game_box" class="center_block"></div>

	</body>
	<footer>
		<script>
			game_life("game_box");
		</script>
	</footer>
</html>
```

ПРИЛОЖЕНИЕ Б

Файл game_life.js в котором хранятся все основные функции игры

```
function step_life(rad,i,j,arr,color){
	var v1=0;
	var sel=document.getElementById("height_game");
	var game_height=Number(sel.options[sel.selectedIndex].value);
	var sel=document.getElementById("width_game");
	var game_width=Number(sel.options[sel.selectedIndex].value);
	var border_on=document.getElementById("border_on").checked;
	for(var k=-rad;k<rad+1;k++){
		for(var l=-rad;l<rad+1;l++){
			var t=i+k;
			var s=j+l;
			if(!((t==i)&&(s==j))){
				if(border_on==false){
					if(t<0){
						t=game_height+t;
						if(s<0){
							s=game_width+s;
						}
						if(s>game_width-1){
							s=s-game_width;
						}
					}else if(t>game_height-1){
						t=t-game_height;
						if(s<0){
							s=game_width+s;
						}
						if(s>game_width-1){
							s=s-game_width;
						}
					}else{
						if(s<0){
							s=game_width+s;
						}
						if(s>game_width-1){
							s=s-game_width;
						}
					}
				}else{
					if(t<0){
						continue;
					}
					if(t>game_height-1){
						continue;
					}
					if(s<0){
						continue;
					}
					if(s>game_width-1){
						continue;
					}
				}
				if(arr[t][s]==color){
					v1=v1+1;
				}
			}
		}
	}
	return v1;
}
function play_click(){
	var arr=[];
	var countval={red:0,black:0};
	var cv=[];
	cv[0]=countval;
	var sel=document.getElementById("height_game");
	var game_height=Number(sel.options[sel.selectedIndex].value);
	var sel=document.getElementById("width_game");
```

```
var game_width=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("rad_sel1");
var rad1=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("rad_sel2");
var rad2=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("life_on_start1");
var start1=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("life_on_finish1");
var finish1=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("new_life_on1");
var life_on1=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("life_on_start2");
var start2=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("life_on_finish2");
var finish2=Number(sel.options[sel.selectedIndex].value);
var sel=document.getElementById("new_life_on2");
var life_on2=Number(sel.options[sel.selectedIndex].value)
for(var i=0;i<game_height;i++){
        arr[i]=[];
        for(var j=0;j<game_width;j++){
                if(document.getElementById("box_"+i+"_"+j).className=="box_black"){
                        arr[i][j]=1;
                }
                if(document.getElementById("box_"+i+"_"+j).className=="box_red"){
                        arr[i][j]=2;
                }
                if(document.getElementById("box_"+i+"_"+j).className=="box_white"){
                        arr[i][j]=0;
                }
        }
}
for(var i=0;i<game_height;i++){
        for(var j=0;j<game_width;j++){
                var v1=step_life(rad1,i,j,arr,1);
                var v2=step_life(rad2,i,j,arr,2);
                if(arr[i][j]==0){
                        a1=(finish1-life_on1+1)/((2*rad1+1)*(2*rad1+1));
                        a2=(finish2-life_on2+1)/((2*rad2+1)*(2*rad2+1));
                        if((v1>=life_on1)&&(v1<=finish1)&&(a1>a2)){
                                document.getElementById("box_"+i+"_"+j).className="box_black";
                                countval.black++;
                        }
                        if((v2>=life_on2)&&(v2<=finish2)&&(a1<a2)){
                                document.getElementById("box_"+i+"_"+j).className="box_red";
                                countval.red++;
                        }
                        if((a1=a2)){
                                if(((v1<life_on1)||(v1>finish1))&&(v2>=life_on2)&&(v2<=finish2)){

document.getElementById("box_"+i+"_"+j).className="box_red";
                                        countval.red++;
                                }
                                if((v1>=life_on1)&&(v1<=finish1)&&((v2<life_on2)||(v2>finish2))){

document.getElementById("box_"+i+"_"+j).className="box_black";
                                        countval.black++;
                                }
                        }
                }else{
                        if(((v1>finish1)||(v1<start1))&&((v2>finish2)||(v2<start2))){
                                document.getElementById("box_"+i+"_"+j).className="box_white";
                        }else{
                                if(arr[i][j]==1){
                                        countval.black++;
                                }
                                if(arr[i][j]==2){
                                        countval.red++;
                                }
                        }
                }
        }
}

var step_number=Number(document.getElementById("step_line_val").getAttribute("value"));
```

```
        step_number=step_number+1;
        document.getElementById("step_line_val").innerHTML=step_number;
        document.getElementById("step_line_val").setAttribute("value",step_number);
        if(step_number>1){
                cv=JSON.parse(localStorage.cv);
        }
        cv[step_number]=countval;
        localStorage.cv=JSON.stringify(cv);
        var swidth=Number(document.getElementById("my_plot").getAttribute("data-width"));
        var sheight=Number(document.getElementById("my_plot").getAttribute("data-height"));
        if(step_number>swidth){
                swidth=swidth*10;
                document.getElementById("my_plot").setAttribute("data-width",swidth);
                plot_create("my_plot",swidth/10,sheight/10);
                plot_draw2("my_plot",step_number,cv,"black");
                plot_draw2("my_plot",step_number,cv,"red");
        }
        if((countval.black>sheight)||(countval.red>sheight)){
                sheight=sheight*10;
                document.getElementById("my_plot").setAttribute("data-height",sheight);
                plot_create("my_plot",swidth/10,sheight/10);
                plot_draw2("my_plot",step_number,cv,"black");
                plot_draw2("my_plot",step_number,cv,"red");
        }
        plot_draw("my_plot",step_number,cv,"black");
        plot_draw("my_plot",step_number,cv,"red");
}
function createSel1(name,min,max){
        var sel=document.createElement('select');
        sel.id=name;
        sel.className="size_game";
        sel.setAttribute("onchange","lib_ex()");
        size_block.appendChild(sel);
        for(var i=min;i<max+1;i++){
                var opt=document.createElement('option');
                opt.setAttribute("value",i);
                opt.innerHTML=i;
                document.getElementById(name).appendChild(opt);
        }
}
function game_resize3(n,max){
        document.getElementById("life_on"+n).childNodes[2].nodeValue="";
        var sel = document.getElementById("life_on_start"+n);
        sel.parentNode.removeChild(sel);
        var sel = document.getElementById("life_on_finish"+n);
        sel.parentNode.removeChild(sel);
        createSel4(n,max);
        document.getElementById("life_on"+n).innerHTML+='-';
        createSel5(n);
}
function createSel2(n){
        var sel=document.createElement('select');
        sel.id="rad_sel"+n;
        sel.className="size_game";
        sel.setAttribute("onchange","game_resize2("+n+")");
        document.getElementById("radius"+n).appendChild(sel);
        for(var i=1;i<3;i++){
                var opt=document.createElement('option');
                opt.setAttribute("value",i);
                opt.innerHTML=i;
                document.getElementById("rad_sel"+n).appendChild(opt);
        }
}
function game_resize2(n){
        var sel = document.getElementById("life_on_start"+n);
        var min=Number(sel.options[sel.selectedIndex].value);
        var sel = document.getElementById("life_on_finish"+n);
        var max=Number(sel.options[sel.selectedIndex].value);
        var sel = document.getElementById("new_life_on"+n);
        sel.parentNode.removeChild(sel);
        game_resize3(n,max);
        if(min>max){
                document.getElementById("life_on_start"+n).value=max;
        }else{
```

```
                document.getElementById("life_on_start"+n).value=min;
        }
        document.getElementById("life_on_finish"+n).value=max;
        createSel3(n);
}
function createSel3(n){
        var sel1=document.getElementById("life_on_finish"+n);
        var max=Number(sel1.options[sel1.selectedIndex].value);
        var sel1=document.getElementById("life_on_start"+n);
        var min=Number(sel1.options[sel1.selectedIndex].value);
        var sel=document.createElement('select');
        sel.id="new_life_on"+n;
        sel.className="size_game";
        document.getElementById("new_life"+n).appendChild(sel);
        for(var i=min;i<max+1;i++){
                var opt=document.createElement('option');
                opt.setAttribute("value",i);
                opt.innerHTML=i;
                document.getElementById("new_life_on"+n).appendChild(opt);
        }
        document.getElementById("new_life_on"+n).value=max;

}
function game_resize4(n){
        var sel = document.getElementById("new_life_on"+n);
        sel.parentNode.removeChild(sel);
        createSel3(n);
}
function createSel4(n,max){
        var sel=document.createElement('select');
        sel.id="life_on_start"+n;
        sel.className="size_game";
        sel.setAttribute("onchange","game_resize4("+n+")");
        document.getElementById("life_on"+n).appendChild(sel);
        for(var i=2;i<max+1;i++){
                var opt=document.createElement('option');
                opt.setAttribute("value",i);
                opt.innerHTML=i;
                document.getElementById("life_on_start"+n).appendChild(opt);
        }
}
function createSel5(n){
        var sel1=document.getElementById("rad_sel"+n);
        var rad=Number(sel1.options[sel1.selectedIndex].value);
        var sel=document.createElement('select');
        sel.id="life_on_finish"+n;
        sel.className="size_game";
        sel.setAttribute("onchange","game_resize2("+n+")");
        document.getElementById("life_on"+n).appendChild(sel);
        for(var i=3;i<(rad*2+1)*(rad*2+1);i++){
                var opt=document.createElement('option');
                opt.setAttribute("value",i);
                opt.innerHTML=i;
                document.getElementById("life_on_finish"+n).appendChild(opt);
        }
}
function arr_opt(element,index,array){
        var opt=document.createElement('option');
        opt.setAttribute("value",element.value_ex);
        opt.innerHTML=element.text_ex;
        ex_sel.appendChild(opt);
}
function createSel6(){
        var sel=document.createElement('select');
        sel.id="ex_sel";
        sel.setAttribute("onchange","lib_ex()");
        example.appendChild(sel);
        var mas_ex=[
                {value_ex: 'default', text_ex:'<Пусто>'},
                {value_ex: 'ex1', text_ex:'глайдер'},
                {value_ex: 'ex2', text_ex:'косм. корабль'},
                {value_ex: 'ex3', text_ex:'планерная пушка'},
                {value_ex: 'ex4', text_ex:'осциллятор'},
        ];
```

```
        mas_ex.forEach(arr_opt);
}
function game_field_start(arr){
        var sel=document.getElementById("height_game");
        var game_height=sel.options[sel.selectedIndex].value;
        var sel=document.getElementById("width_game");
        var game_width=sel.options[sel.selectedIndex].value;
        var div=document.createElement('div');
        div.id="game";
        game_field.appendChild(div);
        for(var i=0;i<game_height;i++){
                for(var j=0;j<game_width;j++){
                        var div=document.createElement('div');
                        div.id="box_"+i+"_"+j;
                        switch(arr[i][j]){
                                case 0:
                                        div.className="box_white";
                                        break;
                                case 1:
                                        div.className="box_black";
                                        break;
                                case 2:
                                        div.className="box_red";
                                        break;
                        }
                        game.appendChild(div);
                }
                var div=document.createElement('br');
                game.appendChild(div);
        }
}
function game_life(game_id){
        var div=document.createElement('div');
        div.id="game_sq";
        document.getElementById(game_id).appendChild(div);
        var canvas=document.createElement('canvas');
        canvas.id="my_plot";
        canvas.style="background:#fff";
        canvas.setAttribute("data-width",10);
        canvas.setAttribute("data-height",10);
        document.getElementById(game_id).appendChild(canvas);
        var div=document.createElement('div');
        div.id="game_field";
        game_sq.appendChild(div);
        var div=document.createElement('div');
        div.id="toolbar_box";
        game_sq.appendChild(div);
        var div=document.createElement('div');
        div.id="play";
        toolbar_box.appendChild(div);
        var div=document.createElement('div');
        div.id="pause";
        toolbar_box.appendChild(div);
        var div=document.createElement('div');
        div.id="step";
        toolbar_box.appendChild(div);
        var div=document.createElement('div');
        div.id="stop";
        toolbar_box.appendChild(div);
        var div=document.createElement('div');
        div.id="size_block";
        div.className="sel_block";
        div.innerHTML="Размер: ";
        toolbar_box.appendChild(div);
        createSel1("height_game",5,60);
        document.getElementById("size_block").innerHTML+='x';
        createSel1("width_game",5,100);
        var sel1=document.getElementById("height_game");
        var game_height=sel1.options[sel1.selectedIndex].value;
        var sel2=document.getElementById("width_game");
        var game_width=sel2.options[sel2.selectedIndex].value;
        var div=document.createElement('div');
        div.id="example";
        div.className="sel_block";
```

```
div.innerHTML="Примеры: ";
toolbar_box.appendChild(div);
createSel6();
var div=document.createElement('div');
div.id="border_box";
div.className="sel_block";
div.innerHTML="Граница: ";
toolbar_box.appendChild(div);
var div=document.createElement('input');
div.id="border_on";
div.type="checkbox";
div.innerHTML="Граница: ";
border_box.appendChild(div);
document.getElementById("toolbar_box").innerHTML+='</br>';
var div=document.createElement('div');
div.id="radius"+1;
div.className="sel_block";
div.innerHTML="Радиус: ";
toolbar_box.appendChild(div);
createSel2(1);
var div=document.createElement('div');
div.id="life_on"+1;
div.className="sel_block";
div.innerHTML="Выживание при: ";
toolbar_box.appendChild(div);
createSel4(1,3);
document.getElementById("life_on"+1).innerHTML+='-';
createSel5(1);
var div=document.createElement('div');
div.id="new_life"+1;
div.innerHTML="Рождение с: ";
toolbar_box.appendChild(div);
createSel3(1);
var input=document.createElement('input');
input.id="color"+1;
input.name="Color";
input.value=1;
input.type="radio";
input.setAttribute("checked","checked");
toolbar_box.appendChild(input);
var label=document.createElement('label');
label.setAttribute("for","color"+1);
toolbar_box.appendChild(label);
document.getElementById("toolbar_box").innerHTML+='</br>';
var div=document.createElement('div');
div.id="radius"+2;
div.className="sel_block";
div.innerHTML="Радиус: ";
toolbar_box.appendChild(div);
createSel2(2);
var div=document.createElement('div');
div.id="life_on"+2;
div.className="sel_block";
div.innerHTML="Выживание при: ";
toolbar_box.appendChild(div);
createSel4(2,3);
document.getElementById("life_on"+2).innerHTML+='-';
createSel5(2);
var div=document.createElement('div');
div.id="new_life"+2;
div.innerHTML="Рождение с: ";
toolbar_box.appendChild(div);
createSel3(2);
var input=document.createElement('input');
input.id="color"+2;
input.name="Color";
input.value=2;
input.type="radio";
toolbar_box.appendChild(input);
var label=document.createElement('label');
label.setAttribute("for","color"+2);
toolbar_box.appendChild(label);
var div=document.createElement('div');
div.id="step_line";
```

```
        div.innerHTML="Поколение: ";
        toolbar_box.appendChild(div);
        var div=document.createElement('div');
        div.id="step_line_val";
        div.setAttribute("value",0);
        div.innerHTML=0;
        step_line.appendChild(div);
        lib_ex();
        plot_create("my_plot",1,1);
        var id_interval;
        var play_on=0;
        window.onclick = function() {
                var el=event.target;
                if((el.className=="box_white")||(el.className=="box_black")||(el.className=="box_red")){
                        if(document.all.Color[0].checked){
                                if(el.className=="box_white"||el.className=="box_red"){
                                        el.className="box_black";
                                }else{
                                        el.className="box_white";
                                }
                        }else{
                                if(el.className=="box_white"||el.className=="box_black"){
                                        el.className="box_red";
                                }else{
                                        el.className="box_white";
                                }
                        }

                }
                if(el.id=="play"){
                        if(play_on==0){
                                id_interval=setInterval("play_click()",300);
                                play_on=1;
                                document.getElementById("play").style.display="none";
                                document.getElementById("pause").style.display="inline-block";
                        }
                }
                if(el.id=="pause"){
                        if(play_on==1){
                                clearInterval(id_interval);
                                play_on=0;
                                document.getElementById("pause").style.display="none";
                                document.getElementById("play").style.display="inline-block";
                        }
                }
                if(el.id=="step"){
                        if(play_on==1){
                                clearInterval(id_interval);
                                play_on=0;
                                document.getElementById("pause").style.display="none";
                                document.getElementById("play").style.display="inline-block";
                        }else{
                                play_click();
                        }
                }
                if(el.id=="stop"){
                        if(play_on==1){
                                clearInterval(id_interval);
                                play_on=0;
                                document.getElementById("pause").style.display="none";
                                document.getElementById("play").style.display="inline-block";
                        }
                        lib_ex();
                }
        }
}
```

ПРИЛОЖЕНИЕ В

Файл lib_ex.js хранит функцию хранящую некоторые примеры фигур

```
function lib_ex(){
	document.getElementById("step_line_val").setAttribute("value",0);
	document.getElementById("step_line_val").innerHTML=0;
	document.getElementById("game_field").innerHTML="";
	var sel=document.getElementById("ex_sel");
	var ex=sel.options[sel.selectedIndex].value;
	document.getElementById("my_plot").setAttribute("data-width",10);
	document.getElementById("my_plot").setAttribute("data-height",10);
	plot_create("my_plot",1,1);
	switch(ex){
		case 'default':
			var sel=document.getElementById("height_game");
			var height=sel.options[sel.selectedIndex].value;
			var sel=document.getElementById("width_game");
			var width=sel.options[sel.selectedIndex].value;
			var arr=[];
			for(var i=0;i<height;i++){
				arr[i]=[];
				for(var j=0;j<width;j++){
					arr[i][j]=0;
				}
			}
			document.getElementById("border_on").checked=false;
			document.getElementById("height_game").value=arr.length;
			document.getElementById("width_game").value=arr[0].length;
			document.getElementById("height_game").disabled=false;
			document.getElementById("width_game").disabled=false;
			document.getElementById("rad_sel1").value=1;
			game_resize3(1,3);
			document.getElementById("life_on_start1").value=2;
			document.getElementById("life_on_finish1").value=3;
			game_resize2(1);
			document.getElementById("new_life_on1").value=3;
			game_field_start(arr);
			break;
		case 'ex1':
			var arr=[
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,1,0,0,0,0,0],
				[0,0,0,0,0,1,0,0,0,0],
				[0,0,0,1,1,1,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0]
			];
			document.getElementById("border_on").checked=false;
			document.getElementById("height_game").value=arr.length;
			document.getElementById("width_game").value=arr[0].length;
			document.getElementById("height_game").disabled=true;
			document.getElementById("width_game").disabled=true;
			document.getElementById("rad_sel1").value=1;
			game_resize3(1,3);
			document.getElementById("life_on_start1").value=2;
			document.getElementById("life_on_finish1").value=3;
			game_resize2(1);
			document.getElementById("new_life_on1").value=3;
			game_field_start(arr);
			break;
		case 'ex2':
			var arr=[
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,0,0,0,0,0],
				[0,0,0,0,0,1,0,0,0,0],
				[0,0,0,0,0,0,1,0,0,0],
```

```
                [0,0,1,0,0,0,1,0,0,0],
                [0,0,0,1,1,1,1,0,0,0],
                [0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0,0]
            ];
            document.getElementById("border_on").checked=false;
            document.getElementById("height_game").value=arr.length;
            document.getElementById("width_game").value=arr[0].length;
            document.getElementById("height_game").disabled=true;
            document.getElementById("width_game").disabled=true;
            document.getElementById("rad_sel1").value=1;
            game_resize3(1,3);
            document.getElementById("life_on_start1").value=2;
            document.getElementById("life_on_finish1").value=3;
            game_resize2(1);
            document.getElementById("new_life_on1").value=3;
            game_field_start(arr);
            break;
        case 'ex3':
            var arr=[

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,1,1,0,0],

[0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
            ];
            document.getElementById("border_on").checked=true;
            document.getElementById("height_game").value=arr.length;
            document.getElementById("width_game").value=arr[0].length;
```

```
                        document.getElementById("height_game").disabled=true;
                        document.getElementById("width_game").disabled=true;
                        document.getElementById("rad_sel1").value=1;
                        game_resize3(1,3);
                        document.getElementById("life_on_start1").value=2;
                        document.getElementById("life_on_finish1").value=3;
                        game_resize2(1);
                        document.getElementById("new_life_on1").value=3;
                        game_field_start(arr);
                        break;
                case 'ex4':
                        var arr=[
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0],
                                [0,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0],
                                [0,0,1,1,0,1,1,0,0,0,1,0,0,0,0,0],
                                [0,0,0,0,0,1,0,0,1,0,1,0,1,1,0,0],
                                [0,0,0,0,0,1,0,1,0,0,1,0,1,1,0,0],
                                [0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
                                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
                        ];
                        document.getElementById("border_on").checked=false;
                        document.getElementById("height_game").value=arr.length;
                        document.getElementById("width_game").value=arr[0].length;
                        document.getElementById("height_game").disabled=true;
                        document.getElementById("width_game").disabled=true;
                        document.getElementById("rad_sel1").value=1;
                        game_resize3(1,3);
                        document.getElementById("life_on_start1").value=2;
                        document.getElementById("life_on_finish1").value=3;
                        game_resize2(1);
                        document.getElementById("new_life_on1").value=3;
                        game_field_start(arr);
                        break;
        }
}
```

ПРИЛОЖЕНИЕ Г

Файл my_plot.js хранит функции для рисования графика

```
function plot_create(id,step_x,step_y){
	var plot = document.getElementById(id);
	plot.width=704;
	plot.height=600;
	var  pc = plot.getContext('2d');
	pc.beginPath();
	pc.moveTo(100,plot.height-10);
	pc.lineTo(100,10);
	pc.moveTo(70,plot.height-30);
	pc.lineTo(660,plot.height-30);
	pc.strokeStyle="#000";
	pc.stroke();
	pc.closePath();
	pc.beginPath();
	for(var i=1;i<11;i=i+1){
		pc.moveTo(100+i*50,plot.height-30);
		pc.lineTo(100+i*50,plot.height-555);
		pc.moveTo(100,plot.height-30-i*50);
		pc.lineTo(625,plot.height-30-i*50);

	}
	pc.strokeStyle="#bfbfbf";
	pc.stroke();
	pc.closePath();
	plot_step_x(id,step_x);
	plot_step_y(id,step_y);
}
function plot_step_x(id,step_x){
	var plot = document.getElementById(id);
	var  pc = plot.getContext('2d');
	pc.beginPath();
	pc.clearRect(101,plot.height-29,600,30);
	pc.font = "10px Arial";
	for(var i=1;i<11;i=i+1){
		if(i*step_x<10){
			pc.fillText(i*step_x,97+i*50,plot.height-20);
		}else if(i*step_x<100){
			pc.fillText(i*step_x,94+i*50,plot.height-20);
		}else if(i*step_x<1000){
			pc.fillText(i*step_x,91+i*50,plot.height-20);
		}else{
			pc.fillText(i*step_x,88+i*50,plot.height-20);
		}
	}
	pc.stroke();
	pc.closePath();
}
function plot_step_y(id,step_y){
	var plot = document.getElementById(id);
	var  pc = plot.getContext('2d');
	pc.clearRect(0,0,99,plot.height-31);
	pc.beginPath();
	pc.font = "10px Arial";
	for(var i=1;i<11;i=i+1){
		if(i*step_y<10){
			pc.fillText(i*step_y,90,plot.height-27-i*50);
		}else if(i*step_y<100){
			pc.fillText(i*step_y,85,plot.height-27-i*50);
		}else if(i*step_y<1000){
			pc.fillText(i*step_y,79,plot.height-27-i*50);
		}else if(i*step_y<10000){
			pc.fillText(i*step_y,73,plot.height-27-i*50);
		}else {
			pc.fillText(i*step_y,67,plot.height-27-i*50);
		}
	}
	pc.stroke();
	pc.closePath();
```

```
}
function plot_draw(id,step,arr,color){
        var plot = document.getElementById(id);
        var  pc = plot.getContext('2d');
        var swidth=Number(document.getElementById(id).getAttribute("data-width"));
        var sheight=Number(document.getElementById(id).getAttribute("data-height"));
        pc.beginPath();
        pc.moveTo(100+(step-1)*500/swidth,plot.height-30-arr[step-1][color]*500/sheight);
        pc.lineTo(100+step*500/swidth,plot.height-30-arr[step][color]*500/sheight);
        pc.strokeStyle=color;
        pc.stroke();
        pc.closePath();
}
function plot_draw2(id,step,arr,color){
        var plot = document.getElementById(id);
        var  pc = plot.getContext('2d');
        var swidth=Number(document.getElementById(id).getAttribute("data-width"));
        var sheight=Number(document.getElementById(id).getAttribute("data-height"));
        pc.beginPath();
        pc.moveTo(100,plot.height-30-arr[0][color]*500/sheight);
        for(var i=1;i<step;i++){
                pc.lineTo(100+i*500/swidth,plot.height-30-arr[i][color]*500/sheight);
        }
        pc.strokeStyle=color;
        pc.stroke();
        pc.closePath();
}
```

ПРИЛОЖЕНИЕ Д

Файл style.css хранит стили для шаблона

```
.center_block{
        display:table;
    margin:auto;
}
.body{
        width:1024px;
}
```

ПРИЛОЖЕНИЕ Е

Файл game_style.css хранит стили для игры

```
#game{
        display: table;
        border: #000;
        border-width: 3px 4px 4px 3px;
        border-style: groove;
        margin: auto;
}
#game_sq{
        display: table;
        border: #000;
        border-width: 3px 3px 3px 3px;
        border-style: groove;
        height:428;
        -webkit-user-select: none;
}
#game_field{
        background: #B5D4A5;
        height: inherit;
        display: table-cell;
    vertical-align: middle;
}
.box_white{
        height:7;
        width:7;
        display: inline-block;
        cursor:pointer;
        background:url(../image/box_white.BMP);
}
.box_white:hover{
        background:url(../image/box_white_hover.BMP);
}
.box_black{
        height:7;
        width:7;
        display: inline-block;
        cursor:pointer;
        background:url(../image/box_black.BMP);
}
.box_black:hover{
        background:url(../image/box_black_hover.BMP);
}
.box_red{
        height:7;
        width:7;
        display: inline-block;
        cursor:pointer;
        background:url(../image/box_red.BMP);
}
.box_red:hover{
        background:url(../image/box_red_hover.BMP);
}
#play{
        background-image:url(../image/play.BMP);
        height:30;
        width:30;
        display: inline-block;
        border: #000;
        border-width: 1px 1px 1px 1px;
        border-style: groove;
        margin: 2px;
        cursor:pointer;
}
#play:hover{
        background-image:url(../image/play_hover.BMP);
}
#pause{
        background-image:url(../image/pause.BMP);
        height:30;
```

```
        width:30;
        border: #000;
        border-width: 1px 1px 1px 1px;
        border-style: groove;
        margin: 2px;
        display: none;
        cursor:pointer;
}
#pause:hover{
        background-image:url(../image/pause_hover.BMP);
}
#step{
        background-image:url(../image/step.BMP);
        height:30;
        width:30;
        display: inline-block;
        border: #000;
        border-width: 1px 1px 1px 1px;
        border-style: groove;
        margin: 2px;
        cursor:pointer;
}
#step:hover{
        background-image:url(../image/step_hover.BMP);
}
#stop{
        background-image:url(../image/stop.BMP);
        height:30;
        width:30;
        display: inline-block;
        border: #000;
        border-width: 1px 1px 1px 1px;
        border-style: groove;
        margin: 2px;
        cursor:pointer;
}
#stop:hover{
        background-image:url(../image/stop_hover.BMP);
}
#step_line{
        float: right;
        padding-top: 6px;
    padding-bottom: 6px;
        display: inline-block;
        border: #000;
        border-width: 1px 1px 1px 1px;
        border-style: groove;
        margin: 2px;
        padding-left: 10px;
        padding-right: 10px;
        background: lightgray;
}
#step_line_val{
        display: inline-block;
}
.sel_block{
        display: inline-block;
        background: lightgray;
        vertical-align: top;
        height: 30px;
        margin: 2px;
        padding-left: 10px;
        padding-right: 10px;
}
#border_on{
        height: 16px;
        width: 64px;
        cursor:pointer;
}
#new_life1,#new_life2{
        display: inline-block;
        background: lightgray;
        vertical-align: top;
        height: 30px;
```

```
        margin: 2px;
        padding-left: 10px;
        padding-right: 10px;
}
#toolbar_box{
        background: powderblue;
        height:35px;
        border: #000;
        border-width: 0px 3px 3px 3px;
        border-style: groove;
        vertical-align: bottom;
    display: table-row;
        width:700px;

}
#ex_sel{
        display: inline-block;
    height: 30px;
        width:147px;
        cursor:pointer;
}
.size_game{

        display: inline-block;
    height: 30px;
        width:50px;
        cursor:pointer;
}
#color1,#color2{
        display:none;
}
#color1 + label{
        margin-top: 2px;
        margin-left: 2px;
        background-image:url(../image/BLACK.BMP);
        width:30;
        height:30;
        display: inline-block;
        cursor:pointer;
}
#color1:checked + label{
        background-image:url(../image/BLACK_click.BMP);
}

#color2 + label{
        margin-top: 2px;
        margin-left: 2px;
        background-image:url(../image/RED.BMP);
        width:30;
        height:30;
        display: inline-block;
        cursor:pointer;
}
#color2:checked + label{
        background-image:url(../image/RED_click.BMP);
}
```

ПРИЛОЖЕНИЕ Ж

Файлы рисунков

Файл 1 - BLACK.bmp

Файл 2 - BLACK_click.bmp

Файл 3 - RED.bmp

Файл 4 - RED_click.bmp

Файл - pause.bmp

Файл 5 - pause_hover.bmp

Файл 6 - play.bmp

Файл 7 - play_hover.bmp

Файл 8 - step.bmp

Файл 9 - step_hover.bmp

Файл 10 - stop.bmp

Файл 11 - stop_hover.bmp

Файл 12-18 - box_black.bmp, box_black_hover.bmp, box_red.bmp, box_red_hover.bmp, box_white.bmp, box_white_hover.bmp (слева на право)