Санкт-Петербургский государственный политехнический
университет
Петра Великого
Институт прикладной математики и механики
Кафедра «Теоретическая Механика»

КУРСОВОЙ ПРОЕКТ

Web-приложение – игра «2048»

по дисциплине «Математическое моделирование»

Выполнили
студенты гр. 13632/2

Зуев В.
Великов М.
Комаров Г.

Руководитель
ассистент

Д. В. Цветков

Санкт-Петербург
2018

Содержание

1 Постановка задачи

Требовалось создать компьютерную игру для исполнения в браузере с пользовательским интерфейсом (таблица чисел 4×4) и правилами:

1. В каждом раунде появляется плитка номинала «2»(с вероятностью 75%) или «4»(с вероятностью 25%)

2. Нажатием стрелки игрок может сдвинуть все блоки игрового поля в одну из 4 сторон. Если при сбрасывании два блока одного номинала сталкиваются, то они образуют один блок, номинал которого равен сумме соединившихся. После каждого хода на свободной секции поля появляется новый блок номиналом «2»или «4».

3. Если в одной строчке или в одном столбце находится более двух блоков одного номинала, то при сбрасывании они начинают соединяться с той стороны, в которую были направлены. Например, находящиеся в одной строке плитки $(4, 4, 4)$ после хода влево превратятся в $(8, 4)$, или же, после хода вправо, — в $(4, 8)$.

4. За каждое соединение игровые очки увеличиваются на номинал получившегося блока.

5. Игра заканчивается поражением при отсутствии возможных ходов.

Притом необходимо выводить на экран текущий счёт игры. В ситуации отсутствия возможности хода оповещать пользователя - выводить сообщение "Game over".

2 Реализация

2.1 Процесс игры

Пользователь видит таблицу с числами и счёт. Управлять (делать ходы) можно экранными кнопками «вверх», «вниз», «вправо», «влево». По кнопке «New game»страница перезагружается и начинается новая игра.

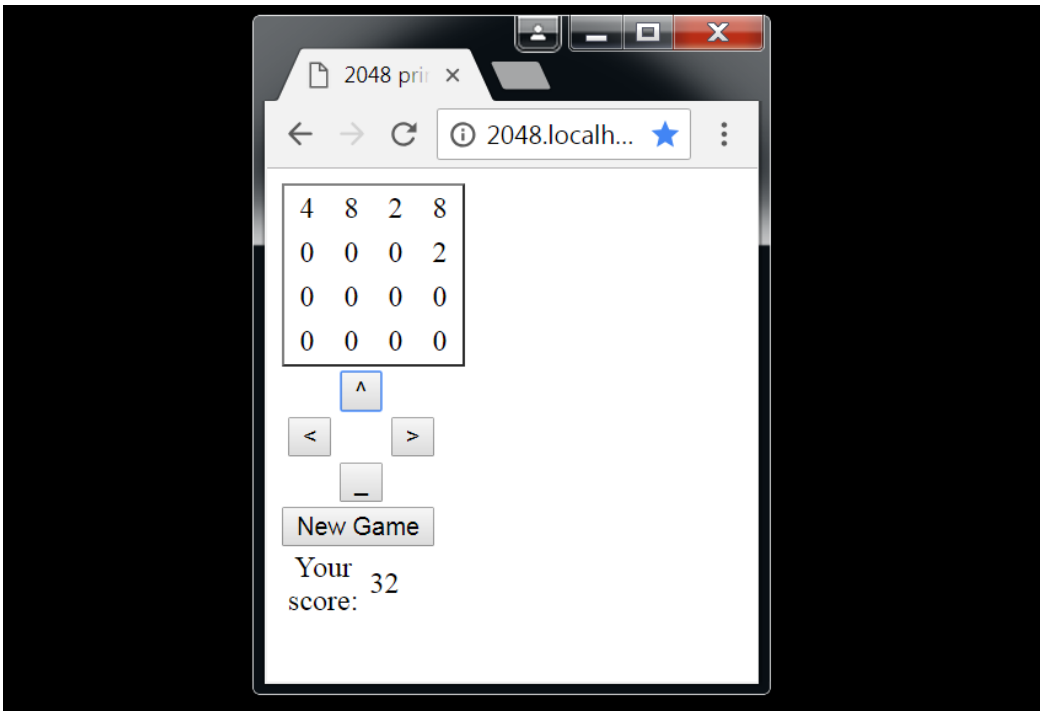

Рис. 1: Пользовательский интерфейс игры

В ситуации, когда следующий ход сделать невозможно, пользователю выводится сообщение «Game over!».

2.2 Архитектура приложения

Приложение состоит из четырёх файлов - сама страница (index.html), файл стиля (stylesheet.css), функции, вызываемые страницей (application.js), логика, вызываемая из application (calc.js), функции вывода результата в html и запросов содержимого html (graphics.js).
Все скрипты имеют доступ к глобальным переменным Sum (текущий счёт) и GameoverFlag (булева переменная; в случае false игра завершена)

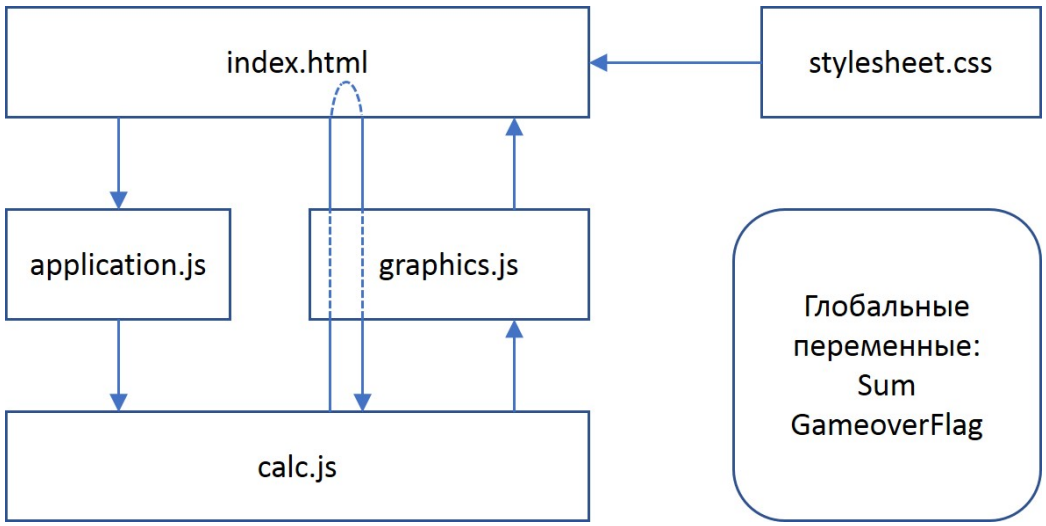

Рис. 2: Схема взаимодействия файлов приложения

Файлы скриптов содержат следующие функции (везде A - двумерный массив 4×4, положение на игровом поле):

1. application.js
   (a) init(): запускается, когда страница открывается или перезагружается
   (b) onrightpress(): запускается по нажатию экранной кнопки «вправо». Аналогично onleftpress, ondownpress, onuppress.
   Если GameoverFlag равна true, совершает ход (вызывает move)
2. calc.js
   (a) move(dir,A): выводит на экран таблицу - результат хода из исходной позиции, заданной двумерным массивом A, в направлении dir (0-вправо, 1 - вниз, 2 - влево, 3 - вверх)

(b) rotate(dir,A): возвращает матрицу A, повёрнутую по часовой стрелке dir раз. Обёртка функций line_rotate и angle_rotate

(c) rotateBack(dir,A): возвращает матрицу A, повёрнутую против часовой стрелки dir раз

(d) check(A): возвращает булеву переменную. Если false, из положения A невозможно совершить ход ни в какую сторону

(e) calc_new_pos(A): возвращает двумерный массив - результат хода из положения A вправо

3. graphics.js

(a) draw(A): заполняет таблицу - игровое поле в html значениями элементов A

(b) gameover(): выводит на экран сообщение «Game over!», устанавливает глобальную переменную GameoverFlag равной false

(c) addPoints(points): выводит на экран счёт (points)

(d) getMatrix(): возвращает текущее состояние таблицы из html

2.3 Как осуществляется ход игрока

2.3.1 Общая схема работы программы

При инициализации (init()) создаётся двумерный из нулей, в случайные места вставляются две двойки и результат выводится на экран. Пользователь может совершать ходы.
По нажатию экранной кнопки текуще положение считывается с экрана (getMatrix()) запускается move от этого положения и направления, соответствующего кнопке. Далее исполняются следующие команды:

```
function move (dir, A) {
        console.log("func_move");
        var A1 = rotate(dir,A);
        var A2=calc_new_pos(A1);
        var A3=rotateBack(dir,A2);
        addPoints(Sum);
        draw(A3);
        var flag = check(A3);
        if (flag == false) {
                gameover();
        }
```

}

Пояснение к коду:

1. Матрица поворачивается так, чтобы ход совершался направо;
2. Совершается ход направо, а также определяетсяо текущий счёт (изменяется глобальная переменная Sum);
3. Матрица поворачивается обратно;
4. Счёт выводится на экран;
5. Матрица выводится на экран (в игровое поле);
6. Проверяется, возможен ли следующий ход;
7. Если следующий ход невозможен, запускается функция gameover.

Функция Gameover, как было сказано выше, изменяет глобальную переменную GameoverFlag и оповещает пользователя о том, что игра закончена:

```
function gameover(){
        afterScoreText.innerHTML='Game over!';
        GameoverFlag=true;
}
```

2.3.2 Функции сдвига и заполнения случайных клеток таблицы

Все функции обёрнуты в функцию calc_new_pos(A), внутри которой для прерывания ссылки на A матрице A1 поэлементно присваиваются значения A, после чего вызывается shiftAndClap(A1).
Функция ShiftAndClap перебирает справа налево элементы массива и соединяет элементы одного номинала, присваивая удвоенную стоимость элементу справа.После этого осуществляется функция сдвига shift.
Функция сдвига shift(B) осуществляет перебор с четвертого элемента первой строки и двигается справа налево сверху вниз, находит элемент, справа от которого стоит ноль и меняет их местами. Если таковой элемент найден и сдвинут, возвращается в конец строчки и переходит на следующую только тогда, когда все элементы сдвинуты вправо
Функция добавления в случайное место 2 или 4 - addRandNum. Сначала идет генерация 2 или 4 с вероятностью 75/25 соответственно. После этого идет перебор всех элементов массива и запоминание мест нулей в

массивы placeOfNulli и placeOfNullj.Далее случайным образом выбирается место и присваивается сгенерированное число.

2.3.3 Функции поворота матрицы

В ходе работы над проектом наша группа приняла решение написать только одну функцию сдвига по направлению влево, вместо того чтобы писать функции сдвига для каждого из направлений. Данное решение привело к необходимости вращать матрицу до и после осуществления сдвига. Ниже приведена схема сдвигов, которая наглядно демонстрирует алгоритм поворота матрицы в зависимости от направления сдвига:

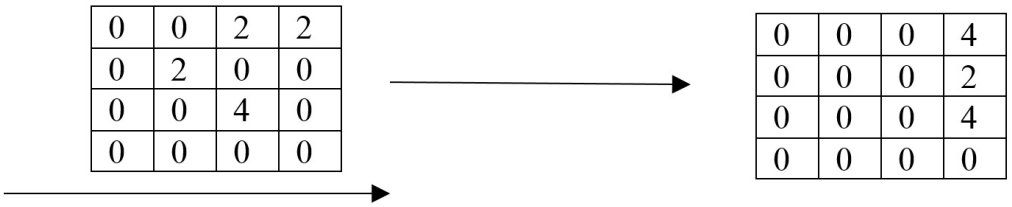

Рис. 3: Сдвиг вправо

Сдвиг вправо - базовый случай, поворот не требуется. К нему сводятся остальные путём поворота.

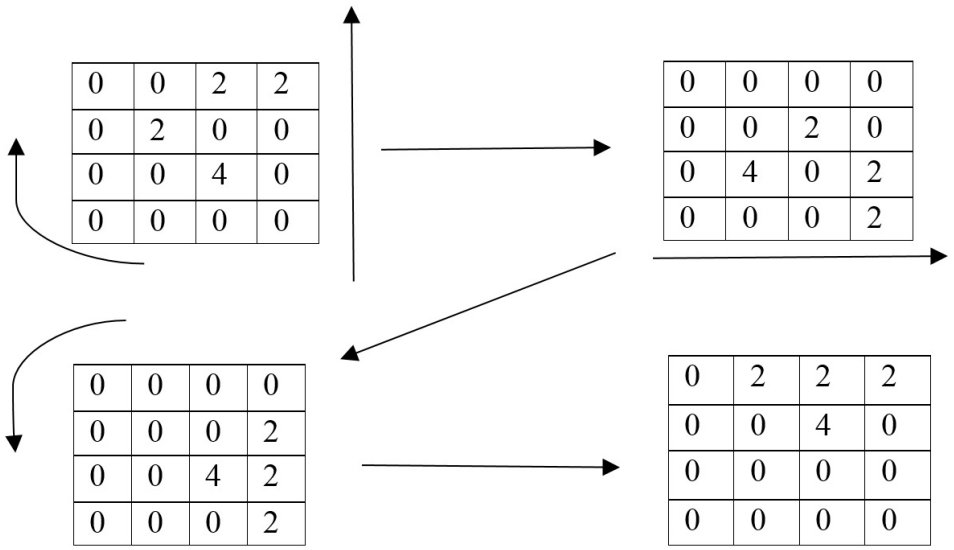

Рис. 4: Сдвиг вверх путём поворота и сдвига вправо

Приведём код этих поворотов:

1. разворот на 180 градусов для сдвига влево:

```
function line_rotate(A){
    //required for turn on 180 degrees
    var A_rotate=[];
    for(var i = 0; i < 4; i++){
        A_rotate.push([]);
        for(var p = 0; p < 4; p++){
            A_rotate[i].push(A[i][3-p]);
        }
    }
    return A_rotate;
}
```

   Исполнение данной операции осуществляется за счёт вложенного цикла, что позволяет переписать каждый из вложенных массивов в обратном порядке.

2. Разворот по или против часовой стрелки:

```
function angle_rotate(A, dir){
    var A_rotate=[]
    if (dir == 1){
        /*first turn for "up", second for "down"*/
      for(var i = 0; i < 4; i++){
          A_rotate.push([])
          for(var p = 0; p < 4; p++){
            A_rotate.push(A[3-p][i])
          }
        }
    }
    if (dir == 3){
        /*first turn for "up", second for "down"*/
        for(var i = 0; i < 4; i++){
            A_rotate.push([])
            for(var p = 0; p < 4; p++){
                A_rotate.push(A[p][3-i])
            }
        }
```

```
    }
    return A_rotate
}
```

Данная функция проверяет в каком направлении следует вращать матрицу и в зависимости от этого переписывает новую матрицу построчно с помощью вложенного цикла.

2.3.4 Функция проверки возможности кода check

Матрица A сравнивается с другой, полученной из A сдвигом направо. Если есть хоть один элемент, отличный от эл-та в матрице A - значит, ход возможен. Далее A поворачивается по часовой стрелке и процесс повторяется. Если даже после треьего поворота нет ни одной позиции, в которой матрица после сдвига была бы отличной от исходной, возвращается false (нет возможных ходов).

```
function check(A){
        var T=null;
        var flag=false;
        for (var i=0; i<4; i++){
                T=calc_new_pos(A);
                for (var j=0; j<4; j++){
                        for (var k=0; k<4; k++){
                                if(T[j][k]!=A[j][k]){
                                        flag=true;
                                        return flag;
                                }
                        }
                }
                A=rotate(1,A);
        }
        return flag; //false=>gameover
}
```

3 Распределение работ

Работа над проектом была распределена следующим образом:

- Зуев В.: Графический интерфейс и архитектура приложения
  Файлы - index.html, stylesheet.css, graphics.js, application.js

- Комаров Г.: Алгоритм вычисления следующей позиции после хода игрока и алгоритм вычисления текущего счёта игры
  Файл calc.js (функция calc_new_pos)

- Великов М.: Вспомогательные алгоритмы для задания позиции и проверка возможности совершить ход.
  Файл calc.js (функции rotate, rotateBack, check)

Все авторы принимали участие в написании отчёта.

4 Результаты

Создано Web-приложение - игра. Поставленные задачи (игра по правилам, вывод счёта, завершение игры) выполнены.
Исходный код проекта можно найти по адресу github.com/zuevval/2048.
Работу приложения можно посмотреть на сайте 2048.timefwd.ru.

5 Заключение

Объектно-ориентированный язык Javascript позволяет в связке с языком разметки HTML быстро создавать графический интерфейс браузерных приложений. Встроенный в браузер (например, в Google Chrome) отладчик предоставляет широкие возможности для оптимизации кода и устранения ошибок. Впрочем, необходимо учитывать особенности задания переменных через значения других - часто вместо создания независимой переменной новое имя содержит лишь ссылку на существующий объект. Возможностей Javascript хватило для написания полноценного web-приложения - игры «2048». Проект доведён до завершённого состояния, но может быть улучшен: планируется изменить дизайн (например, добавить анимацию сдвига цифр на поле методами Javascript) и добавить серверную часть для хранения рекордов.