南京工程学院

**软件工程项目训练**

**实训报告**

课 程 名 称 软件工程项目训练

院（系、部、中心） 国际教育学院

专 业 软件工程

班 级 软工中外202

学 号 202201334

姓 名 吴之豪

实 习 企 业 优课达（南京）网络有限公司

|  |  |
| --- | --- |
| **一 实习概况** | |
| 实习单位 | 优课达（南京）网络有限公司 |
| 实习岗位 | 软件开发工程师 |
| **二 实习内容** | |
| 2048网页游戏  **1 简介**   * 1. **系统介绍** * 根据Javascript技术编写的2048小游戏   1. **采用技术和工具** * HTML,CSS,JavaScript,JQuery   **2 系统设计**  系统在设计时，通过HTML和CSS完成游戏界面的设计，通过JavaScript和jQuery按照游戏的规则完成相应功能的实现，之后玩家在网页中按照游戏设定的规则操作2048游戏。  **3 系统实现**  // JavaScript Document  //初始化游戏界面  (function(window, document, $){      function Game2048(opt){          var score = 0;                  //通过变量保存分数          var winNum = 2048;              //胜利条件          var isGameOver = true;          //游戏是否已经结束          //从opt对象参数中取出成员，并保存为对应名称的变量          var prefix = opt.prefix, len = opt.len, size = opt.size, margin = opt.margin;          //创建了view对象，该对象用于处理游戏的页面效果          var view = new View(prefix, len, size, margin);          view.init();          var board = new Board(len);          /\*测试          board.init();          console.log(board.arr);          board.arr = [              [0, 0, 0, 2], [0, 2, 0, 2], [2, 2, 2, 2], [0, 2, 4, 0],          ];          board.moveLeft();          console.log(board.arr);\*/            //测试view.addNum()          board.onGenerate = function(e) {              view.addNum(e.x, e.y, e.num);  //替换原来的“console.log(e);”          };          //以动画效果移动单元格          board.onMove = function(e) {               if (e.to.num >= winNum) {                   isGameOver = true;                   setTimeout(function() { view.win(); }, 300);               }              if(e.to.num > e.from.num) {    //e.to.num（合并后数字），e.from.num（被移动数字）                  score += e.to.num;         //累加分数                  view.updateScore(score);   //更新页面中显示的分数              }              //每当border.arr中的单元格移动时，调用此方法控制页面中的单元格移动              view.move(e.from, e.to);          };          board.onMoveComplete = function(e) {              //判断是否失败              if (!board.canMove()) {                  isGameOver = true;                  setTimeout(function() { view.over(score); }, 300);              }              if (e.moved) {                  //一次移动操作全部结束后，如果移动成功，则在棋盘中增加一个单元格                  setTimeout(function(){ board.generate(); }, 200);              }          };          $(document).keydown(function(e) {              //如果游戏结束，则不再执行移动操作              if (isGameOver) {                  if(score<=1000){                      alert("你太笨了");                  }                  return false;                }              switch (e.which) {                  case 37: board.moveLeft();  break;  //左移                  case 38: board.moveUp();    break;  //上移                  case 39: board.moveRight(); break;  //右移                  case 40: board.moveDown();  break;  //下移              }          });          //将初始化游戏相关的代码整理到star()函数          function start() {           //开始游戏              score = 0;               //将保存的分数重置为0              view.updateScore(0);     //将页面中的分数重置为0              view.cleanNum();         //清空页面中多余的数字单元格              board.init();            //初始化单元格数组              board.generate();        //生成第一个数字              board.generate();        //生成第二个数字              isGameOver = false;      //将游戏状态设为开始          }          $('#' + prefix + '\_restart').click(start); //为“重新开始”按钮添加单击事件          start();                                   //开始游戏（页面打开后自动开始游戏）      }      window['Game2048'] = Game2048;  })(window, document, jQuery);     //自调用函数，调用函数时传入三个参数，表示该函数依赖这些全局变量  //编写View函数，在构造函数中设置棋盘背景的宽度和高度  function View(prefix, len, size, margin){      this.prefix = prefix;    //id或class前缀      this.len = len;          //棋盘单边单元格数量（总数量为len × len）      this.size = size;        //单元格边长，单元格大小为size×size      this.margin = margin;    //单元格间距      this.container = $('#' + prefix + '\_container');     //用于获取页面中id为game\_container的<div>元素      var containerSize = len \* size + margin \* (len + 1);  //margin \* (len + 1)为间距的总边长      this.container.css({width: containerSize, height: containerSize});      this.nums = {};                                      //保存所有数字单元格对象      this.score = $('#' + prefix + '\_score');             //获取显示分数的元素对象，并保存到this.score中  }  //自动生成空棋盘格  View.prototype = {      getPos: function(n) {          return this.margin + n \* (this.size + this.margin);      },      //根据len（棋盘单边单元格数量）自动生成空单元格，生成后将会添加到game\_container容器      init: function(){          for(var x = 0, len = this.len; x < len; ++x) {              for(var y = 0; y < len; ++y) {                  var $cell = $('<div class="' + this.prefix + '-cell"></div>');                  //指定空单元格的样式，其top与left通过getPos()方法进行计算                  $cell.css({                      width: this.size + 'px',                      height: this.size + 'px',                      top: this.getPos(x),                      left: this.getPos(y)                  }).appendTo(this.container);              }          }      },      //在页面中显示数字单元格      addNum: function(x, y, num){          var $num = $('<div class="' + this.prefix + '-num ' + this.prefix + '-num-' + num + ' ">');          $num.text(num).css({              //用于从中心位置展开              top: this.getPos(x) + parseInt(this.size / 2),              left: this.getPos(y) + parseInt(this.size / 2)          }).appendTo(this.container).animate({      //animate(参数对象/数字,时间)过渡动画              width: this.size + 'px',              height: this.size + 'px',              lineHeight: this.size + 'px',              top: this.getPos(x),              left: this.getPos(y)          },100);          this.nums[x + '-' + y] = $num;      },      move: function(from, to) {          //根据参数from和to对象中保存的x、y值，拼接成“x-y”形式的字符串          //用于从this.nums中获取fromIndex（被移动对象下标）和toIndex（目标对象下标）元素          var fromIndex = from.x + '-' + from.y, toIndex = to.x + '-' + to.y;          //将this.nums中的toIndex对象替换成fromIndex对象          var clean = this.nums[toIndex];          this.nums[toIndex] = this.nums[fromIndex];          //删除this.nums中的FormIndex属性，此时只有toIndex是被移动对象          delete this.nums[fromIndex];          //为toIndex对象设置动画，以200ms过渡时间移动到目标对象位置          var prefix = this.prefix + '-num-';          var pos = {top: this.getPos(to.x), left: this.getPos(to.y)};          this.nums[toIndex].finish().animate(pos, 200, function() {          if (to.num > from.num) {             //判断数字是否合并（合并后to.num大于from.num）            clean.remove();            $(this).text(to.num).removeClass(prefix + from.num).addClass(prefix + to.num);          }        });      },      updateScore: function(score){          this.score.text(score);      },      //显示获胜和失败的信息      win: function() {          //用于在id为game\_over\_info元素中添加提示信息          $('#' + this.prefix + '\_over\_info').html('<p>您获胜了</p>');          //用于移除id为game\_over元素的class样式game\_hide，移除后提示信息就会显示出来          $('#' + this.prefix + '\_over').removeClass(this.prefix + '-hide');      },      over: function(score) {          $('#' + this.prefix + '\_over\_info').html('<p>本次得分</p><p>' + score + '</p>');          $('#' + this.prefix + '\_over').removeClass(this.prefix + '-hide');      },      //用于清空页面中所有的数字单元格      cleanNum: function() {          //用于清空this.nums中保存的所有数字单元格对象          this.nums = {};          //用于隐藏游戏结束时的提示信息          $('#' + this.prefix + '\_over').addClass(this.prefix + '-hide');          //用于移除页面中所有的数字单元格          $('.' + this.prefix + '-num').remove();      }  };  //控制游戏数值  function Board(len) {      this.len = len;      this.arr = [];  }  Board.prototype = {      onGenerate: function() {},     //每当generate()方法被调用时，执行此方法      onMove: function() {},      onMoveComplete: function() {},      //1.创建棋盘数组      init: function(){              //init()方法用于根据指定len创建二维数组          //外层数组表示行，内层数组表示列，如果单元格为空，则数组元素对应的值为0          for (var arr = this.arr, x = 0, len = this.len; x < len; ++x) {              arr[x] = [];              for(var y = 0; y < len; ++y) {                  arr[x][y] = 0;     //在初始情况下所有单元格都是空的，为每个单元格赋值为0              }          }          this.arr = arr;      },      //2.为棋盘生成随机数字单元格，随机生成数字2或4，保存到数组的随机位置      generate: function() {          var empty = [];          //查找数组中所有值为0的元素索引          for (var x = 0, arr = this.arr, len = arr.length; x < len; ++x) {          //用于获取this.arr数组中所有空单元格的下标并保存到empty数组中              for (var y = 0; y < len; ++y) {                  if (arr[x][y] === 0) {                      empty.push({x: x, y: y});                  }              }          }          if (empty.length < 1) {              return false;          }          var pos = empty[Math.floor((Math.random() \* empty.length))]; //随机选取empty中的一个空单元格          //Math.floor()向下取最大整数，Math.random()取0-1之间的随机数，empty[索引0-16]          this.arr[pos.x][pos.y] = Math.random() < 0.5 ? 2 : 4;        //随机生成一个2或4，0.5表示随机几率相同          this.onGenerate({x: pos.x, y: pos.y, num: this.arr[pos.x][pos.y]});//将数字填入到单元格中          //通过调用this.onGenerate()触发事件，将新创建的单元格在二维数组中的位置和数字内容传递过去      },      //实现单元格移动      //1.单元格左移      moveLeft: function() {        var moved = false;    //是否有单元格被移动        // 外层循环从上到下遍历“行”，内层循环从左到右遍历“列”        for (var x = 0, len = this.arr.length; x < len; ++x) {          for (var y = 0, arr = this.arr[x]; y < len; ++y) {            // 从 y + 1 位置开始，向右查找            for (var next = y + 1; next < len; ++next) {              // 如果 next 单元格是 0，找下一个不是 0 的单元格              if (arr[next] === 0) {                continue;              }              // 如果 y 数字是 0，则将 next 移动到 y 位置，然后将 y 减 1 重新查找              if (arr[y] === 0) {                arr[y] = arr[next];                //调用onMove()方法，该方法表示每次单元格移动时触发的事件，其参数是一个对象                //from保存被移动的单元格的x、y位置和数字，to保存目标单元格的x、y位置和数字                this.onMove({from: {x: x, y: next, num: arr[next]}, to: {x: x, y: y, num: arr[y]}});                arr[next] = 0;                canMove = true;                --y;              // 如果 y 与 next 单元格数字相等，则将 next 移动并合并给 y              } else if (arr[y] === arr[next]) {                arr[y] += arr[next];                this.onMove({from: {x: x, y: next, num: arr[next]}, to: {x: x, y: y, num: arr[y]}});                arr[next] = 0;                moved = true;              }              break;            }          }        }        //调用该方法，表示在整个左移操作完成后触发事件，其参数是一个对象，moved表示本次操作是否发生过单元格移动        //如果发生过单元格移动则会在棋盘中自动增加一个新的随机数字单元格        this.onMoveComplete({moved: moved});      },      //2.单元格右移      moveRight: function() {        var moved = false;         for (var x = 0, len = this.arr.length; x < len; ++x) {          for (var y = len - 1, arr = this.arr[x]; y >= 0; --y) {            for (var prev = y - 1; prev >= 0; --prev) {              if (arr[prev] === 0) {                continue;              }              if (arr[y] === 0) {                arr[y] = arr[prev];                this.onMove({from: {x: x, y: prev, num: arr[prev]}, to: {x: x, y: y, num: arr[y]}});                arr[prev] = 0;                moved = true;                ++y;              } else if (arr[y] === arr[prev]) {                arr[y] += arr[prev];                this.onMove({from: {x: x, y: prev, num: arr[prev]}, to: {x: x, y: y, num: arr[y]}});                arr[prev] = 0;                moved = true;              }              break;            }          }        }        this.onMoveComplete({moved: moved});      },      //3.单元格上移      moveUp: function() {        var moved = false;        for (var arr = this.arr, len = arr.length, y = 0; y < len; ++y) {          for (var x = 0; x < len; ++x) {            for (var next = x + 1; next < len; ++next) {              if (arr[next][y] === 0) {                continue;              }              if (arr[x][y] === 0) {                arr[x][y] = arr[next][y];                this.onMove({from: {x: next, y: y, num: arr[next][y]}, to: {x: x, y: y, num: arr[x][y]}});                arr[next][y] = 0;                moved = true;                --x;              } else if (arr[x][y] === arr[next][y]) {                arr[x][y] += arr[next][y];                this.onMove({from: {x: next, y: y, num: arr[next][y]}, to: {x: x, y: y, num: arr[x][y]}});                arr[next][y] = 0;                moved = true;              }              break;            }          }        }        this.onMoveComplete({moved: moved});      },      //4.单元格下移      moveDown: function() {        var moved = false;        for (var arr = this.arr, len = arr.length, y = 0; y < len; ++y) {          for (var x = len - 1; x >= 0; --x) {            for (var prev = x - 1; prev >= 0; --prev) {              if (arr[prev][y] === 0) {                continue;              }              if (arr[x][y] === 0) {                arr[x][y] = arr[prev][y];                this.onMove({from: {x: prev, y: y, num: arr[prev][y]}, to: {x: x, y: y, num: arr[x][y]}});                arr[prev][y] = 0;                moved = true;                ++x;              } else if (arr[x][y] === arr[prev][y]) {                arr[x][y] += arr[prev][y];                this.onMove({from: {x: prev, y: y, num: arr[prev][y]}, to: {x: x, y: y, num: arr[x][y]}});                arr[prev][y] = 0;                moved = true;              }              break;            }          }        }        this.onMoveComplete({moved: moved});      },      //判断游戏是否失败：数字填满所有单元格并且相邻单元格无法合并      canMove: function() {        for (var x = 0, arr = this.arr, len = arr.length; x < len; ++x) {          for (var y = 0; y < len; ++y) {            if (arr[x][y] === 0) {           //当前单元格为0，表示可以移动              return true;            }            var curr = arr[x][y], right = arr[x][y + 1];            var down = arr[x + 1] ? arr[x + 1][y] : null;            if (right === curr || down === curr) {  //相邻的右、下单元格数字相等，表示可以合并              return true;            }          }        }        return false;          //遍历结束没有符合条件的单元格则说明当前无法移动      }  }; | |
| **三 实习总结** | |
| 经过这次的实训,让我明白了许多，事情需要循序渐进，一步一步脚踏实地的完成，"一口不能吃成个胖子”，有些事情看起来很简单，但真的动手操作就有了一定的难度，只有经过不断的学习和细细挖掘，才把这个游戏的功能基本实现了，最后只实现了记录当前成绩的功能，本来还希望能够实现排行榜这个功能，不过以我目前的能力还不足以完成，不过我以后有机会定会去实现的它，完善它!  常言道，不积跬步无以至千里，不积小流无以成江海，在书中和网上找了又找，试了又试，一遍又一遍程序中的很多方法，最终才把需要的方法，能够使用的方法融入到程序当中，实现了必要的功能模块。通过本次的实训，我发现，程序的编写还是在于平时的积累，这样的话就不会遇到一个问题就需要去查阅资料，也就是说我所做的还是很不足的。针对我薄弱的方面，我会在今后的学习生活中加以改善，让我设计的这款游戏更加丰富、多彩。 | |
| **四 指导教师评语** | |
| 报告成绩： | |