# playGame网页小游戏开发周报

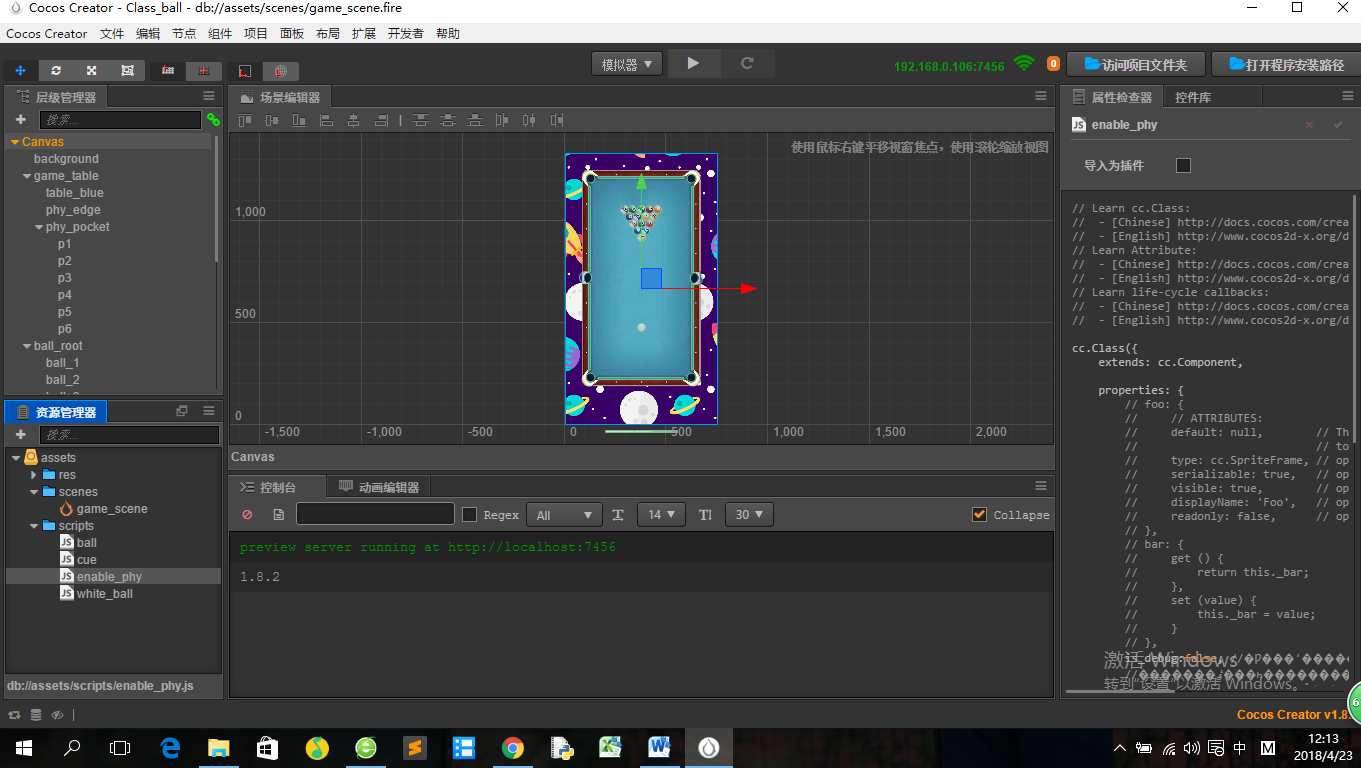
谭嘉 2013110105 2018.04.15

**一、设计历程**：

清明节前夕，软件工程团队项目给出立项时间和发布时间。我一开始也很茫然，自己的编程能力不好，两人项目也最终没有完成。最初的设想很多，也尝试了很多开发工具，学习了基础语言，但是寻找团队给我最初的困扰。由于插班和班级同学不熟悉，并没有找到组织。也是个人交技能力有限吧。最主要的还是能力有限。

接下来，我开始做爬虫项目，数据可视化过程中，bug比较多，数据库搭建不成功等，我不得不放弃。后来，我想做一个像iphone中的siry这样的智能语音助手，接触了tensorFlow框架，自然语言和循环神经网络，着实感觉脑子不够用了。

尝试第三个设想，做单机小游戏。使用cocos creator 平台搭建桌球小游戏，用了两天的时间从界面设计，素材制作到代码编辑，最终完成了设想中的一小步，最终还是因为物理引擎没有搭建出来，放弃了。



但是这个项目的过程中我学到了JS对象函数制作小游戏的基本思路。加上自己一个人寻找出路的过程我萌生了接下来的想法。

**二、一个人的小游戏**

（1）项目设想：在5月10日发布之前的三周时间实用化ＪＳ语言模仿3款经典单机小游戏；

（2）设想来源：第一因为自己编程能力不高，第二因为其实有些时候游戏也可以自己一个人玩对不对，就像没有团队，也要一个人走下去，不然放弃挣扎，才是对自己最大否定。

（3）项目计划：　ａ．俄罗斯方块（４月２２日）

　　　ｂ．扫雷（４月２９日）

ｃ．汉塔诺　（５月６日）

**三、第一周周报**

ＪＳ实现俄罗斯方块　tetris

**ｃｓｓ**

|  |
| --- |
| div.tetrisGame {  position: relative;  margin: 50px auto;  padding: 0;  border: none;  width: 600px;  height: 600px;  }  div.tetrisGame div.tetrisLeftPane {  position: absolute;  top: 0;  left: 0;  width: 300px;  }  div.tetrisLeftPane div.tetrisScore {  position: absolute;  top: 150px;  left: 0;  font-size: 20pt;  }  div.tetrisLeftPane div.tetrisPreview {  position: absolute;  left: 150px;  width: 120px; /\* 4 cols \* 30px \*/  height: 120px; /\* 4 rows \* 30px \*/  }  div.tetrisLeftPane div.tetrisUsage {  position: absolute;  left: 0;  top: 300px;  }  div.tetrisLeftPane div.tetrisUsage th,  div.tetrisLeftPane div.tetrisUsage td {  text-align: left;  }  div.tetrisLeftPane div.tetrisUsage td {  padding-left: 1em;  }  div.tetrisRightPane {  position: absolute;  left: 300px;  top: 0px;  width: 300px; /\* 10 cols \* 30px \*/  height: 600px; /\* 20 rows \* 30px \*/  }  div.tetrisRightPane div.tetrisBoard {  position: absolute;  left: 0;  top: 0;  width: 300px; /\* 10 cols \* 30px \*/  height: 600px; /\* 20 rows \* 30px \*/  }  div.tetrisBlock {  position: absolute;  width: 29px;  height: 29px;  border: 1px solid black;  background-color: black;  }  div.tetrisBlock.habitated {  border: 1px solid black;  background-color: gray;  } |

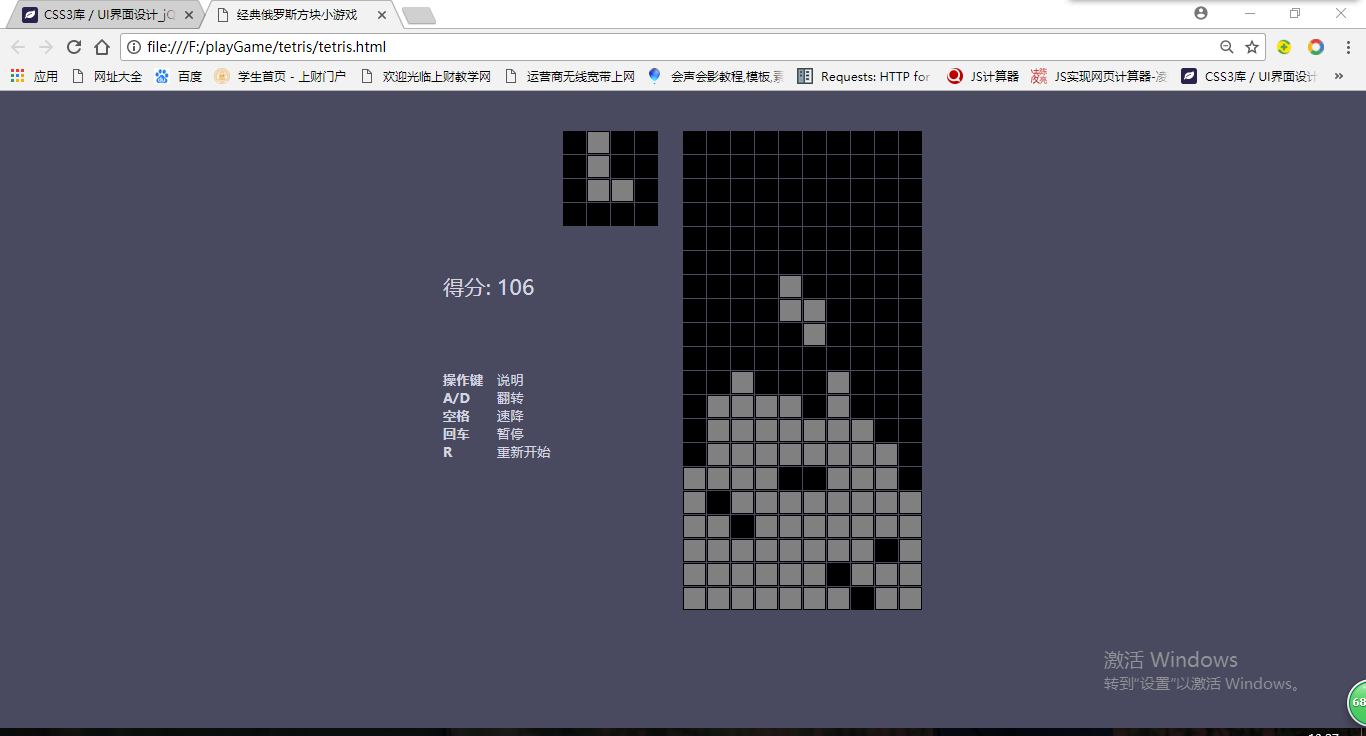
　ＪＳ

|  |
| --- |
| var NUM\_ROWS = 20;  var NUM\_COLS = 10;  var BLOCK\_WIDTH = 30;  var BLOCK\_HEIGHT = 30;  var TICK\_MS = 400;  var CURSOR\_LEFT = 37;  var CURSOR\_RIGHT = 39;  var CURSOR\_DOWN = 40;  var KEY\_A = 65;  var KEY\_D = 68;  var KEY\_R = 82;  var KEY\_ENTER = 13;  var KEY\_SPACE = 32;  var blockPiece = [  [0, 0, 0, 0],  [0, 1, 1, 0],  [0, 1, 1, 0],  [0, 0, 0, 0]  ];  var longPiece = [  [0, 0, 1, 0],  [0, 0, 1, 0],  [0, 0, 1, 0],  [0, 0, 1, 0]  ];  var tPiece = [  [0, 0, 1, 0],  [0, 1, 1, 0],  [0, 0, 1, 0],  [0, 0, 0, 0]  ];  var zlPiece = [  [0, 0, 0, 0],  [0, 0, 1, 1],  [0, 1, 1, 0],  [0, 0, 0, 0]  ];  var zrPiece = [  [0, 0, 0, 0],  [0, 1, 1, 0],  [0, 0, 1, 1],  [0, 0, 0, 0]  ];  var llPiece = [  [0, 0, 1, 0],  [0, 0, 1, 0],  [0, 1, 1, 0],  [0, 0, 0, 0]  ];  var lrPiece = [  [0, 1, 0, 0],  [0, 1, 0, 0],  [0, 1, 1, 0],  [0, 0, 0, 0]  ];  function rotateLeft(piece) {  return [  [piece[0][3], piece[1][3], piece[2][3], piece[3][3]],  [piece[0][2], piece[1][2], piece[2][2], piece[3][2]],  [piece[0][1], piece[1][1], piece[2][1], piece[3][1]],  [piece[0][0], piece[1][0], piece[2][0], piece[3][0]]  ];  }  function rotateRight(piece) {  return [  [piece[3][0], piece[2][0], piece[1][0], piece[0][0]],  [piece[3][1], piece[2][1], piece[1][1], piece[0][1]],  [piece[3][2], piece[2][2], piece[1][2], piece[0][2]],  [piece[3][3], piece[2][3], piece[1][3], piece[0][3]]  ];  }  function intersects(rows, piece, y, x) {  for (var i = 0; i < 4; i++)  for (var j = 0; j < 4; j++)  if (piece[i][j])  if (y+i >= NUM\_ROWS || x+j < 0 || x+j >= NUM\_COLS || rows[y+i][x+j])  return true;  return false;  }  function apply\_piece(rows, piece, y, x) {  var newRows = [];  for (var i = 0; i < NUM\_ROWS; i++)  newRows[i] = rows[i].slice();  for (var i = 0; i < 4; i++)  for (var j = 0; j < 4; j++)  if (piece[i][j])  newRows[y+i][x+j] = 1;  return newRows;  }  function kill\_rows(rows) {  var newRows = [];  var k = NUM\_ROWS;  for (var i = NUM\_ROWS; i --> 0;) {  for (var j = 0; j < NUM\_COLS; j++) {  if (!rows[i][j]) {  newRows[--k] = rows[i].slice();  break;  }  }  }  for (var i = 0; i < k; i++) {  newRows[i] = [];  for (var j = 0; j < NUM\_COLS; j++)  newRows[i][j] = 0;  }  return {  'rows': newRows,  'numRowsKilled': k,  };  }  function randomPiece() {  var pieces = [blockPiece, longPiece, tPiece, zlPiece, zrPiece, llPiece, lrPiece];  var i = Math.floor(Math.random() \* pieces.length);  return pieces[i];  }  function TetrisGame() {  this.paused = false;  this.gameOver = false;  this.score = 0;  this.currentPiece = randomPiece();  this.nextPiece = randomPiece();  this.pieceY = 0;  this.pieceX = 3;  this.rows = [];  for (var i = 0; i < NUM\_ROWS; i++) {  this.rows[i] = []  for (var j = 0; j < NUM\_COLS; j++) {  this.rows[i][j] = 0;  }  }  }  TetrisGame.prototype.tick = function() {  if (this.paused || this.gameOver)  return false;  if (intersects(this.rows, this.currentPiece, this.pieceY + 1, this.pieceX)) {  /\* burn current piece into board \*/  this.rows = apply\_piece(this.rows, this.currentPiece, this.pieceY, this.pieceX);  var r = kill\_rows(this.rows);  this.rows = r.rows;  this.score += 1 + r.numRowsKilled \* r.numRowsKilled \* NUM\_COLS;  /\* fetch next piece \*/  if (intersects(this.rows, this.nextPiece, 0, NUM\_COLS / 2 - 2)) {  this.gameOver = true;  } else {  this.currentPiece = this.nextPiece;  this.pieceY = 0;  this.pieceX = NUM\_COLS / 2 - 2;  this.nextPiece = randomPiece();  }  } else {  this.pieceY += 1;  }  return true;  }  TetrisGame.prototype.togglePaused = function() {  this.paused = !this.paused;  }  TetrisGame.prototype.steerLeft = function() {  if (!intersects(this.rows, this.currentPiece, this.pieceY, this.pieceX - 1))  this.pieceX -= 1;  }  TetrisGame.prototype.steerRight = function() {  if (!intersects(this.rows, this.currentPiece, this.pieceY, this.pieceX + 1))  this.pieceX += 1;  }  TetrisGame.prototype.steerDown = function() {  if (!intersects(this.rows, this.currentPiece, this.pieceY + 1, this.pieceX))  this.pieceY += 1;  }  TetrisGame.prototype.rotateLeft = function() {  var newPiece = rotateLeft(this.currentPiece);  if (!intersects(this.rows, newPiece, this.pieceY, this.pieceX))  this.currentPiece = newPiece;  }  TetrisGame.prototype.rotateRight = function() {  var newPiece = rotateRight(this.currentPiece);  if (!intersects(this.rows, newPiece, this.pieceY, this.pieceX))  this.currentPiece = newPiece;  }  TetrisGame.prototype.letFall = function() {  while (!intersects(this.rows, this.currentPiece, this.pieceY+1, this.pieceX))  this.pieceY += 1;  this.tick();  }  TetrisGame.prototype.get\_rows = function() {  return apply\_piece(this.rows, this.currentPiece, this.pieceY, this.pieceX);  }  TetrisGame.prototype.get\_next\_piece = function() {  return this.nextPiece;  }  TetrisGame.prototype.get\_score = function() {  return this.score;  }  TetrisGame.prototype.get\_game\_over = function() {  return this.gameOver;  }  function draw\_blocks(rows, num\_rows, num\_cols) {  var boardElem = document.createElement('div');  for (var i = 0; i < num\_rows; i++) {  for (var j = 0; j < num\_cols; j++) {  var blockElem = document.createElement('div');  blockElem.classList.add('tetrisBlock');  if (rows[i][j])  blockElem.classList.add('habitated');  blockElem.style.top = (i \* BLOCK\_HEIGHT) + 'px';  blockElem.style.left = (j \* BLOCK\_WIDTH) + 'px';  boardElem.appendChild(blockElem);  }  }  return boardElem;  }  function draw\_tetrisGame(game) {  var leftPaneElem = draw\_tetrisLeftPane(game);  var rightPaneElem = draw\_tetrisRightPane(game);  var gameElem = document.createElement('div');  gameElem.classList.add('tetrisGame');  gameElem.appendChild(leftPaneElem);  gameElem.appendChild(rightPaneElem);  return gameElem;  }  function draw\_tetrisLeftPane(game) {  var scoreElem = draw\_tetrisScore(game);  var previewElem = draw\_tetrisPreview(game);  var usageElem = draw\_tetrisUsage(game);  var leftPaneElem = document.createElement('div');  leftPaneElem.classList.add('tetrisLeftPane');  leftPaneElem.appendChild(previewElem);  leftPaneElem.appendChild(scoreElem);  leftPaneElem.appendChild(usageElem);  return leftPaneElem;  }  function draw\_tetrisRightPane(game) {  var boardElem = draw\_tetrisBoard(game);  var rightPaneElem = document.createElement('div');  rightPaneElem.classList.add('tetrisRightPane');  rightPaneElem.appendChild(boardElem);  return rightPaneElem;  }  function draw\_tetrisBoard(game) {  var rows = game.get\_rows();  var boardElem = draw\_blocks(rows, NUM\_ROWS, NUM\_COLS);  boardElem.classList.add('tetrisBoard');  return boardElem;  }  function draw\_tetrisScore(game) {  var score = game.get\_score();  var scoreElem = document.createElement('div');  scoreElem.classList.add('tetrisScore');  scoreElem.innerHTML = '<p>SCORE: ' + score + '</p>';  if (game.get\_game\_over())  scoreElem.innerHTML += '<p>GAME OVER</p>'  return scoreElem;  }  function draw\_tetrisPreview(game) {  var piece = game.get\_next\_piece();  var pieceElem = draw\_blocks(piece, 4, 4);  var previewElem = document.createElement('div');  previewElem.classList.add('tetrisPreview');  previewElem.appendChild(pieceElem);  return previewElem;  }  function draw\_tetrisUsage(game) {  var usageElem = document.createElement('div');  usageElem.classList.add('tetrisUsage');  usageElem.innerHTML =  "<table>" +  "<tr><th>Cursor Keys</th><td>Steer</td></tr>" +  "<tr><th>a/d</th><td>Rotate</td></tr>" +  "<tr><th>Space bar</th><td>Let fall</td></tr>" +  "<tr><th>Enter</th><td>Toggle pause</td></tr>" +  "<tr><th>r</th><td>Restart game</td></tr>" +  "</table>";  return usageElem;  }  function redraw(game, containerElem) {  var gameElem = draw\_tetrisGame(game);  containerElem.innerHTML = '';  containerElem.appendChild(gameElem);  }  function tetris\_run(containerElem) {  var game = null;  var intervalHandler = null;  var keyHandler = null;  function setIntervalHandler() {  intervalHandler = setInterval(  function() {  if (game.tick())  redraw(game, containerElem);  },  TICK\_MS  );  }  function clearIntervalHandler() {  clearInterval(IntervalHandler);  intervalHandler = null;  }  function setKeyHandler() {  keyHandler = containerElem.addEventListener('keydown', function(kev) {  if (kev.shiftKey || kev.altKey || kev.metaKey)  return;  var consumed = true;  if (kev.keyCode === CURSOR\_LEFT) {  game.steerLeft();  } else if (kev.keyCode === CURSOR\_RIGHT) {  game.steerRight();  } else if (kev.keyCode === CURSOR\_DOWN) {  game.steerDown();  } else if (kev.keyCode === KEY\_A) {  game.rotateLeft();  } else if (kev.keyCode === KEY\_D) {  game.rotateRight();  } else if (kev.keyCode === KEY\_SPACE) {  game.letFall();  } else if (kev.keyCode === KEY\_ENTER) {  game.togglePaused();  } else if (kev.keyCode === KEY\_R) {  game = new TetrisGame();  } else {  consumed = false;  }  if (consumed) {  kev.preventDefault();  redraw(game, containerElem);  }  });  }  function clearKeyHandler() {  containerElem.removeEventListener('keydown', keyHandler);  keyHandler = null;  }  game = new TetrisGame();  redraw(game, containerElem);  setIntervalHandler();  setKeyHandler();  } |

ＨＴＭＬ

|  |
| --- |
| <!DOCTYPE html>  <html lang="zh">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>经典俄罗斯方块小游戏</title>  <link rel="stylesheet" type="text/css" href="css/normalize.css" />  <link rel="stylesheet" type="text/css" href="css/htmleaf-demo.css">  <link rel="stylesheet" type="text/css" href="css/tetris.css">  <script src="js/tetris.js"></script>  </head>  <body onload="tetris\_run(document.body)">  </body>  </html> |

页面布局：

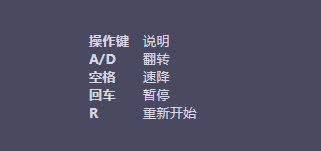


预览区，显示下一个方块的形状：

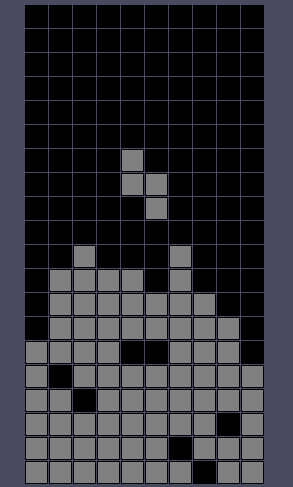
显示当前得分情况：



操作键说明具体玩法：



游戏视图区:



四、项目总结：

因为比较简单的设计，也比较容易实现，中途也曾经试图改代码实现两个人对战，比赛分数，但是能力有限，没有完成。总之，虽然没有团队，中间缺少了团队开发的乐趣和经历，但是希望老师能够酌情给分，第一周结束啦。谢谢老师啦。