## 通知

游戏目前还在更新，博客上核心功能实现，但代码规范、页面优化等都没有管。

游戏源码地址：” <https://github.com/programluo/snackeat>”,

发布地址：

初始版本：” https://programluo.github.io/snackeat/old-version/snakeat.html”

最新版本：“https://programluo.github.io/snackeat/”

## 实现思路

### 背景和蛇实现

使用canvas画布，蛇的身体是一串的正方形

用二维数组去存蛇的身体，二维数组中的每一个数组为蛇身体的每一块的坐标，并跟据坐标在canvas上画方块

var sn = [

[2, 4],

[1, 4]

]

function draw(t, c) {

ctx.fillStyle = c;

ctx.fillRect(t[0] \* 20 + 1, t[1] \* 20 + 1, 18, 18);

}

### 蛇的移动和转向

#### 移动

蛇向前移动就是在二维数组头部插入一个新的数组，删除二维尾部最后一个数组，画图时也是如此，头部画一个方块，尾部把方块的背景色改成画布背景色。

sn.unshift(n = [sn[0][0] + rlx[0], sn[0][1] + rlx[1]]); //sn数组头部添加一个元素

draw(n, "Lime");

draw(sn.pop(), "black"); //sn数组去掉最后一个元素

#### 转向

获得前后左右键的方法

document.onkeydown = function(e) {

fx = e.keyCode;

}

其中

keycode 37 = Left

keycode 38 = Up

keycode 39 = Right

keycode 40 = Down

这些操作对应的下个头结点坐标的运算是

Left: head.x-1, head.y+0;

Up: head.x+0, head.y-1;

Right: head.x+1, head.y+0;

Down: head.x+0, head.y+1;

action = [[-1,0], [0, -1], [1, 0], [0, 1]],

document.onkeydown = function(e) {

if ((fx - e.keyCode) % 2 != 0) {

fx = e.keyCode;

rlx = action[e.keyCode - 37] || rlx;

}

}

sn.unshift(n = [sn[0][0] + rlx[0], sn[0][1] + rlx[1]]);

### 死亡判断

撞墙：判断新增的头数组的左右坐标超过边界

撞自己：二维数组已存在新增头数组（排除头数组，即从二维数组的第二个数组开始比较）

if ((n[0] < 0 || n[0] > 15) || (n[1] < 0 || n[1] > 15) || sn.indexOfArray(n, 1) >= 0) {

return alert("GAME OVER");

}

Array.prototype.indexOfArray = function() {

var len = arguments.length;

if (1 == len) {

var arr = arguments[0];

for (var i = this.length - 1; i >= 0; i--) {

if (this[i].toString() == arr.toString()) {

return i;

}

}

} else {

var arr = arguments[0];

var index = ~~(arguments[1]);

for (var i = this.length - 1; i >= index; i--) {

if (this[i].toString() == arr.toString()) {

return i;

}

}

}

return -1;

}

### 奖励食物

#### 食物生成规则

吃掉食物后生成新的食物，生成食物的坐标要在地图内，生成食物的坐标不可和蛇身体重复

if (dz.toString() == n.toString()) {

while ((sn.indexOfArray(dz = [~~(Math.random() \* 15), ~~(Math.random() \* 15)])) >= 0);

draw(dz, "Yellow");

}

#### 获得食物

获得食物，蛇身增长一段(不执行删除尾借点操作)

if (dz.toString() == n.toString()) {

} else

draw(sn.pop(), "black"); //sn数组去掉最后一个元素

## 代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="http://cdn.static.runoob.com/libs/jquery/2.1.1/jquery.min.js"></script>

</head>

<body>

<canvas id="cas" style="border:solid 1px #CCC;background-color: black" height="300" width="300"></canvas>

</body>

<script>

var sn = [ /\*存蛇身体坐标的二维数组\*/

[2, 4],

[1, 4]

],

action = [

[-1, 0],

[0, -1],

[1, 0],

[0, 1]

],

dz = [3, 4],

/\*食物坐标\*/

rlx = [1, 0],

/\*蛇下一步爬行操作\*/

fx, /\*当前方向，即键盘上下左右值\*/

n, /\*蛇头坐标\*/

ctx = document.getElementById("cas").getContext("2d");

Array.prototype.indexOfArray = function() {

var len = arguments.length;

if (1 == len) {

var arr = arguments[0];

for (var i = this.length - 1; i >= 0; i--) {

if (this[i].toString() == arr.toString()) {

return i;

}

}

} else {

var arr = arguments[0];

var index = ~~(arguments[1]);

for (var i = this.length - 1; i >= index; i--) {

if (this[i].toString() == arr.toString()) {

return i;

}

}

}

return -1;

}

function draw(t, c) {

ctx.fillStyle = c;

ctx.fillRect(t[0] \* 20 + 1, t[1] \* 20 + 1, 18, 18);

}

document.onkeydown = function(e) {

if ((fx - e.keyCode) % 2 != 0) {

fx = e.keyCode;

rlx = action[e.keyCode - 37] || rlx;

}

}

! function() {

sn.unshift(n = [sn[0][0] + rlx[0], sn[0][1] + rlx[1]]); //sn数组头部添加一个元素

if ((n[0] < 0 || n[0] > 15) || (n[1] < 0 || n[1] > 15) || sn.indexOfArray(n, 1) >= 0) {

return alert("GAME OVER");

}

draw(n, "Lime");

if (dz.toString() == n.toString()) {

while ((sn.indexOfArray(dz = [~~(Math.random() \* 15), ~~(Math.random() \* 15)])) >= 0);

draw(dz, "Yellow");

} else

draw(sn.pop(), "black"); //sn数组去掉最后一个元素

setTimeout(arguments.callee, 130);

}();

</script>

</html>