西安交通大學
XI'AN JIAOTONG UNIVERSITY

Web程序设计-游戏设计

第一小组

2020年4月7日

Contents Title

一、小组人员与分工安排

二、实验目的与实验概述

三、实验内容

四、实验总结

一. 小组人员与参与情况

大家都积极参与，无划水现象存在，讨论气氛良好！

班级	姓名	学号	学院
电气810	聂永欣	2186113564	电气学院
测控（食品）81	冯效震	2182112878	机械学院
电类904	马秉	2196412440	本科生院

二. 实验目的与实验概述

实验目的:

运用 web 程序设计有关知识详细解读游戏“贪吃蛇”的实现过程，并将游戏进行改编、加工。

实验概述

小组先在网络上选择了一款“贪吃蛇”的小游戏，运用所学知识对小游戏的 实现过程进行了详细的解读，将其做了一定改编，并通过图像处理将游戏“定制”。



三. 实验内容

贪吃蛇

1.代码解释

游戏的背景界面设置：

```
<style>
 body {
  display: flex;
  height: 100vh;
  margin: 0;
  padding: 0;
  justify-content: center;
  align-items: center;
 }
</style>
```

游戏主题部分：

```
<canvas id="can" width="400" height="400" style="background-color: black">对不起，您的浏览器不支持canvas</canvas>
<script>

var snake = [41, 40], //snake队列表示蛇身，初始节点存在但不显示
 direction = 1,  //1表示向右，-1表示向左，20表示向下，-20表示向上
 food = 43,   //食物的位置
 n,     //与下次移动的位置有关
 box = document.getElementById('can').getContext('2d');
      //从0到399表示box里[0~19]*[0~19]的所有节点，每20px一个节点
```

以20px*20px为一个方格，组成20行20列的方阵，总共400格，然后绿色填充的格子表示蛇身，用黄色表示食物。这400个格子和数字0~399一一对应，对应的方式就是以20作为基数，n / 20再取整表示第几行，n % 20表示第几列。行数和列数都用0~19表示。

蛇用一个一维数组表示，每个值都是这400个数中的一个，用var snake = [41, 40]，。

food表示食物的位置，direction表示蛇头下一次运动的转向。

蛇的运动就用添加和删除数组元素来实现，每次执行绘制蛇头，去掉蛇尾，循环执行使蛇运动。

这是立即执行函数的一种写法。给蛇头添加一个节点n，其值为当前蛇头的值加direction的值。

```
!function() {
 snake.unshift(n = snake[0] + direction);
      //此时的n为下次蛇头出现的位置，n进入队列
 if(snake.indexOf(n, 1) > 0 || n < 0 || n > 399 || direction == 1 && n % 20 == 0 || direction == -1 && n % 20 == 19) {
      //if语句判断贪吃蛇是否撞到自己或者墙壁，碰到时返回，结束程序
  return alert("GAME OVER!");
 }
```

下一行是一个if语句，这个语句就是判断即将出现的蛇头是不是属于蛇身，或者跑到box外边去了。如果没有死亡，就把这个蛇头绘制出来，这是绘制的代码：

```
function draw(seat, color) {
 box.fillStyle = color;
 box.fillRect(seat % 20 *20 + 1, ~~(seat / 20) * 20 + 1, 18, 18);
      //用color填充一个矩形，以前两个参数为x，y坐标，后两个参数为宽和高。
}
```

填充时填充18*18的像素，留1px边框。box.fillRect()中第一个参数就是要绘制的矩形的x坐标seat % 20 *20 + 1，即先得到所要绘制的矩形块在方阵中的位置：第(seat / 20)行，第seat % 20列，再* 20 + 1具体到像素点。

```
 if(n == food) {    //如果吃到食物时，产生一个蛇身以外的随机的点，不会去掉蛇尾
  while (snake.indexOf(food = ~~(Math.random() * 400)) > 0);
  draw(food, "yellow");
 } else {     //没有吃到食物时正常移动，蛇尾出队列
  draw(snake.pop(),"black");
 }
```

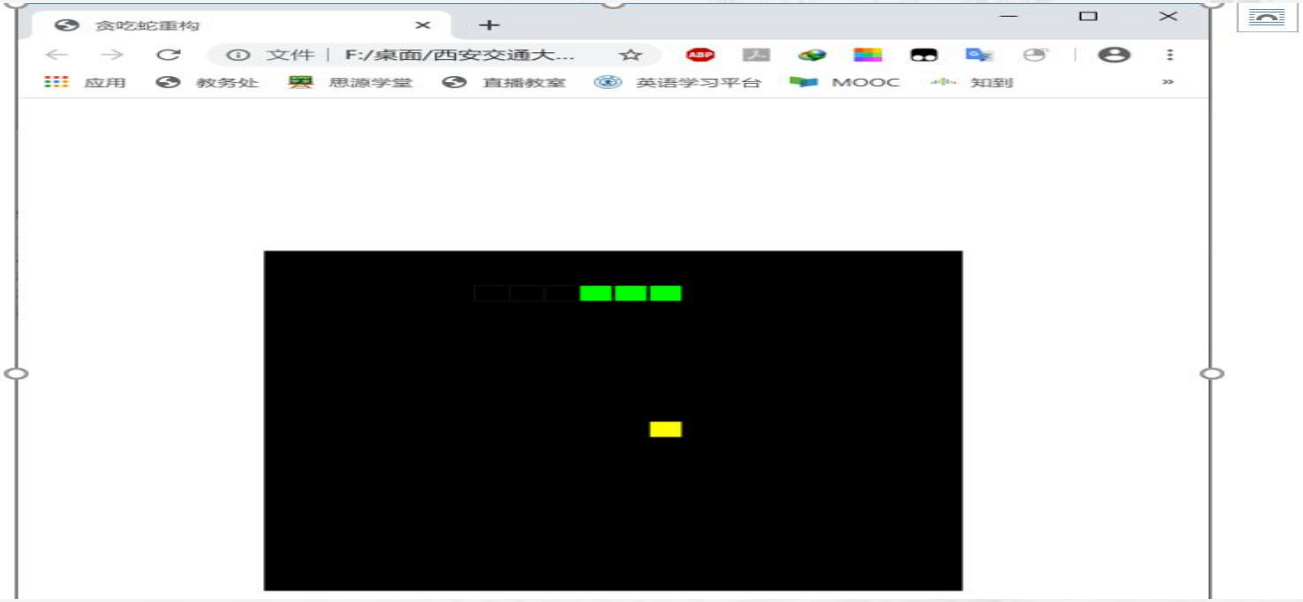
第47行是一个判断语句，判断下次蛇头出现的位置是不是和当前的食物的位置相同：如果相同，生成下一个食物，食物的位置为一个随机数，但是要判断这个点不是出现在当前的蛇身上，绘制食物。如果没有吃到食物，即蛇在正常运动时，每向前一次，将蛇尾弹出，并利用其返回值将这个点重新绘制为黑色。

```
setTimeout(arguments.callee, 150);
    //每隔0.15秒执行函数一次，可以调节蛇的速度
```

前边的判断语句又可分为两部分：

snake[1] - snake[0]的值应该就是-direction。由于玩家可能在一个函数周期中多次改变direction的值，最后使得direction和当前真正的运动方向不一致，导致游戏崩溃.在==后边，[-1, -20, 1, 20][(evt || event).keyCode - 37]中前边的[]是一个数组，后边的[]是取索引，左上右下四个键的keyCode分别为37, 38, 39, 40，计算后的索引为0, 1, 2, 3，使方向键与direction的取值对应起来。于如果按下的按键不是方向键，在数组中将得不到对应的值，返回undefine。此时，由于之后的||运算符，n会取到direction原来的值。

2.实验运行效果：



3.实验代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>贪吃蛇重构</title>
 <style>
  body {
   display: flex;
   height: 100vh;
   margin: 0;
   padding: 0;
   justify-content: center;
   align-items: center;
  }
 </style>
</head>
<body>
 <canvas id="can" width="400" height="400" style="background-color: black">对不起，您的浏览器不支持canvas</canvas>
 <script>
  var snake = [41, 40]，//snake队列表示蛇身，初始节点存在但不显示
    direction = 1， //1表示向右，-1表示向左，20表示向下，-20表示向上
    food = 43， //食物的位置
    n， //与下次移动的位置有关
    box = document.getElementById('can').getContext('2d');
        //从0到399表示box里[0~19]*[0~19]的所有节点，每20px一个节点
   function draw(seat, color) {
    box.fillStyle = color;
    box.fillRect(seat % 20 *20 + 1, ~~(seat / 20) * 20 + 1, 18, 18);
        //用color填充一个矩形，以前两个参数为x，y坐标，后两个参数为宽和高。
   }
    document.onkeydown = function(evt) {
        //当键盘上下左右键摁下的时候改变direction
};
```

```
!function() {
  snake.unshift(n = snake[0] + direction);
        //此时的n为下次蛇头出现的位置，n进入队列
  if(snake.indexOf(n, 1) > 0 || n < 0 || n > 399 || direction == 1 && n % 20 == 0 || direction == -1 && n % 20 == 19) {
        //if语句判断贪吃蛇是否撞到自己或者墙壁，碰到时返回，结束程序
    return alert("GAME OVER!");
  }
  draw(n, "lime");  //画出蛇头下次出现的位置
  if(n == food) {   //如果吃到食物时，产生一个蛇身以外的随机的点，不会去掉蛇尾
while (snake.indexOf(food = ~~(Math.random() * 400)) > 0);
    draw(food, "yellow");
  } else {    //没有吃到食物时正常移动，蛇尾出队列
    draw(snake.pop(),"black");
  }
  setTimeout(arguments.callee, 150);
        //每隔0.15秒执行函数一次，可以调节蛇的速度
 }();
 </script>
</body>
</html>
```

四. 实验总结

（一）完成目标：

- 完成了对游戏“贪吃蛇”的解读
- 实现了对游戏“贪吃蛇”的修改

（二）实验收获：

- 加深掌握了html各标签及各属性的作用及运用
- 加深掌握了JavaScript语言结构及设计
- 加深了对CSS的理解和使用
- 能初步完成简单的html页面、程序设计
- 增强了HTML、JavaScript文件的理解能力



西安交通大学
XI'AN JIAOTONG UNIVERSITY

谢谢大家