# 《Web 程序设计》

# 开放实验报告

| 实验日期 | 2020 年 4 月 7 日 | | |
|---|---|---|---|
| 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 |

# 目录

# 一、 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，并将游戏改编、加工。

# 二、 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、 实验环境

Windows10 64 位、Chrome 浏览器

# 四、 实验内容

## (一) 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>
```

以下是游戏主题部分：

```
<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]，初始化这条蛇，索引 0 为蛇头。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
```

最后的 setTimeout，循环执行当前函数，设置执行周期来调蛇的移动速度。

```
document.onkeydown = function(evt) {
     //当键盘上下左右键摁下的时候改变direction
 direction = snake[1] - snake[0] == (n = [-1, -20, 1, 20][(evt || event).keyCode - 37] || direction) ? direction : n;
};
```

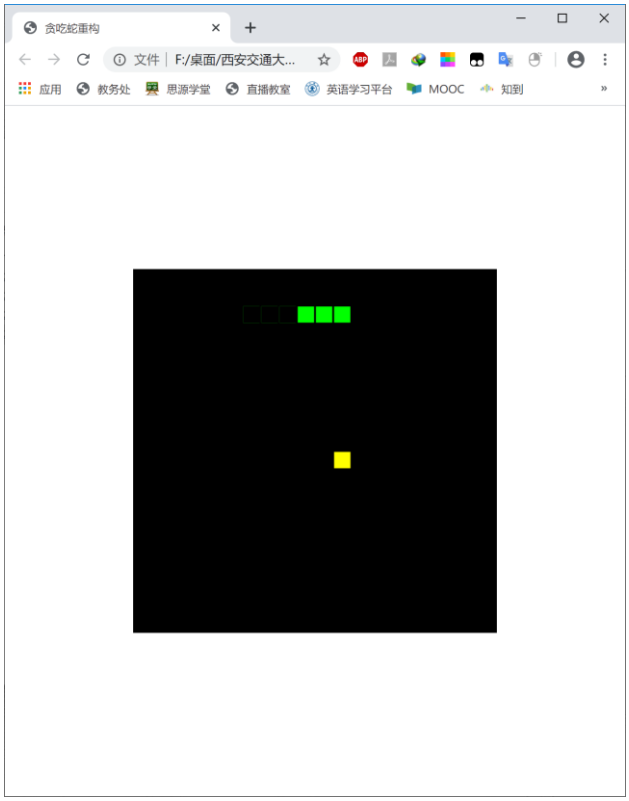
前边的判断语句又可分为两部分：

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。

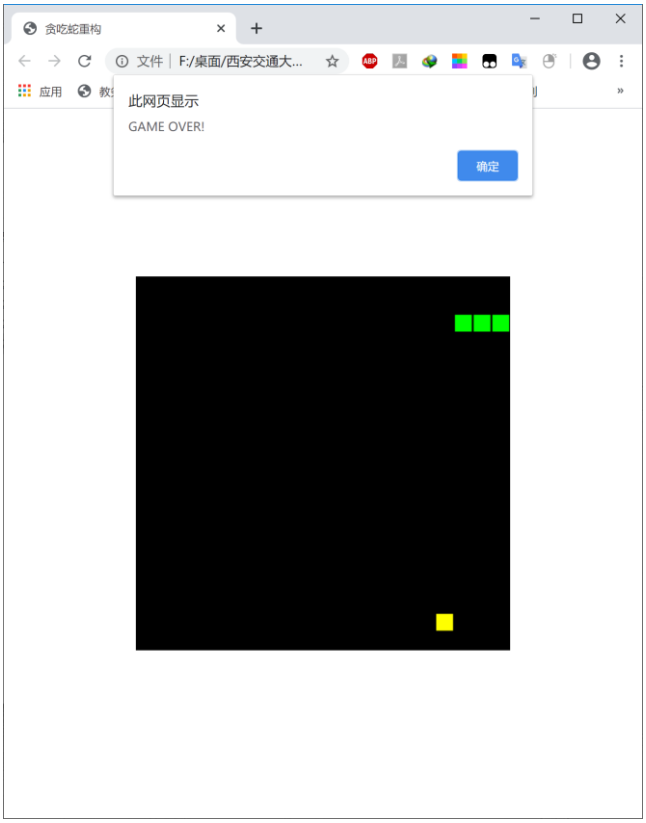
在==后边，[-1, -20, 1, 20][(evt || event).keyCode - 37]中前边的[]是一个数组，后边的[]是取索引，左上右下四个键的 keyCode 分别为 37, 38, 39, 40，计算后的索引为 0, 1, 2, 3，使方向键与 direction 的取值对应起来。于如果按下的按键不是方向键，在数组中将得不到对应的值，返回 undefine。此时，由于之后的||运算符，n 会取到 direction 原来的值。

**2.实际运行效果**

游戏开始




游戏结束



# 五、 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

# 六、 附录

### (一) 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
    direction = snake[1] - snake[0] == (n = [-1, -20, 1, 20][(evt || event).keyCode - 37] || direction) ?
direction : n;
  };
  !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>
```