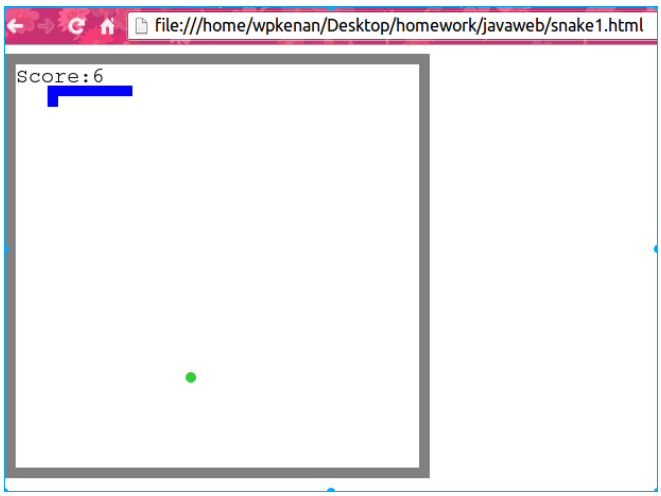
贪吃蛇

1. 设计目标和内容

构建经典的街机游戏贪吃蛇，在贪吃蛇中，玩家控制一条蛇向上、向下、向左或向右移动。随着蛇在游戏区域内移动，苹果会出现。当蛇碰到一个苹果，它会吃掉苹果并且会变长。但是，如果蛇碰到了墙壁或者自己身体的某个部分，游戏就结束了。游戏的过程中需要记录4项内容并将其绘制到屏幕上：边框（灰色）、分数（黑色）、贪吃蛇（蓝色）和苹果（浅绿色）。整体效果图：



2. 主要功能代码

2.1. 创建游戏对象

我们要创建两个对象，即“贪吃蛇”和“苹果”，我们将为这两个对象创建构造方法（名为Snake和Apple），然后我们将方法（例如move和draw）添加到这些构造方法的原型中。我们可以把游戏面板当成一个个方块大小的网格组成的，我们将创建Block对象的构造方法来表示每一个小网格。我们使用一个单个的Block对象来保存苹果当前的位置。贪吃蛇身体也是由多个Block组成。

2.2. 设置键盘控制

我们使用onkeydown来响应键盘按下，响应地设置贪吃蛇前进的方向

2.3. 编写结束游戏gameOver函数

当贪吃蛇碰到墙壁或者碰到自己的身体的时候，我们将调用gameOver 函 数来结束游戏。gameOver 函数使用clearInterval 来停止游戏，并且在画布上显 示文本“Game Over”。

2.4. 构建Block构造方法

我们将定义一个Block 构造方法，它会创建对象来表示不可见的游戏网格中的单个的块。每个块都有 col（column 的缩写）和 row 属性， 它们将存储特定的块在网格上的位置。下图展示了这 个带有数目固定的一 些列和行的网格。尽管这个网格并不会真的出现在屏幕上，游戏设计成让苹果和贪吃蛇段总是能够和网格中的块对齐。

```
var sampleBlock = new Block(5,5);
```

上面只是创建了实例，但是没有将实例绘制出来，要在游戏面板上绘制出该实例还必须使用drawSquare或drawCircle方法。

2.5. 添加equal方法

在游戏中，需要知道两个块是否位于同一位置。例如，如果苹果和贪吃蛇的头部位于同一位置，这意味着，贪吃蛇会吃掉苹果。另一方面，如果贪吃蛇的头部和尾部位于同一位置，那么，贪吃蛇碰到了自己。 为了使得比较块的位置更为容易，我们给Block 构造方法原型添加了 equal 方法。当在一个块对象上调用equal并传递另一个对象作为一个参数， 如果两个对象位于相同的位置，它将返回true（否则的话，返回false）。代码如下：

```
Block.prototype.equal = function (otherBlock){
          return this.col === otherBlock.col && this.row === otherBlock.row;
}
```

2.6. 创建贪吃蛇

我们把贪吃蛇的位置存储为一个名为segments 的数组，其中包含了一系列的块对象。为了移动贪吃蛇，我们在segments 数组的 开头添加一个新的块，并且从数组的尾部删除该块。Segments 数组的第一个元素将表示贪吃蛇的头部。

- Snake构造方法，贪吃蛇一开始长度为3个网格 direction 属性存储了贪吃蛇的当前位置 nextDirection 属性，它存储了贪吃蛇在下一个动画步骤将要移动的方向，除非按下方向键修改该属性，不然贪吃蛇的前进方向不会改变。构造方法将这两个属性都设置为 “right”，因此游戏一开始的时候，贪吃蛇向右移动。

```
var Snake = function (){
  this.segments = [
      new Block(7,5),
      new Block(6,5),
      new Block(5,5)
  ];
      this.direction = "right";
  this.nextDirection = "right";
};
```

2.7. 绘制贪吃蛇

为了绘制贪吃蛇，我们直接遍历其segments 数组中的每一个块，在每个 块上调用在前面所创建的drawSquare 方法。这将会为贪吃蛇的每一段都绘制 一个方块。

```
Snake.prototype.draw = function(){
    for(var i=0;i < this.segments.length; i++){
        this.segments[i].drawSquare("Blue"); //等价于Block实例.drawSquare("Blue");
    }
};
```

2.8. 移动贪吃蛇

贪吃蛇移动看起来好像很复杂，身体的每个块都得移动，有的块可能方向可能会变。其实很简单，我们不用去移动贪吃蛇每个网格组成部分，只要往头部添加一个网格，把最尾部网格删除，其他网格不动就可以了。 我们将创建一个 move 方法，沿着贪吃蛇的当前方向将其移动一个块。为 了移动贪吃蛇，我们添加了一个新的头部段（在segments 数组的开头添加了一个新的block 对象），然后，从segments 数组 删除尾部段。 move方法还将调用一个checkCollision方法， 来查看新的头部是否与贪吃蛇其他的部分或者墙 发生碰撞，以及新的头部是否吃到了苹果。如果新的头部与身体或墙发生碰撞，调用gameOver 函数来结束游戏。 如果贪吃蛇吃到了苹果，我们增加分数，并且将苹果移动到 新的位置。 将this.direction 设置为和this. nextDirection 相等，这会将贪吃蛇的移动方向更 新为与近按下的箭头键一致

使用equal 方法来比较newHead 和 apple.position。如果这两 个块位于相同的位置，equal 方法将会返回true，这意味着，贪吃蛇吃掉了苹果。

将要完成的效果如下：

```
//创建一个新的网格加到当前贪吃蛇前进方向的头部，删除尾部网格
Snake.prototype.move = function(){
    var head = this.segments[0];
    var newHead;
    this.direction = this.nextDirection;
    if(this.direction === "right"){
        newHead = new Block(head.col+1,head.row);
    }else if(this.direction === "down"){
        newHead = new Block(head.col,head.row+1);
    }else if(this.direction === "left"){
        newHead = new Block(head.col-1,head.row);
    }else if(this.direction === "up"){
        newHead = new Block(head.col,head.row-1);
    }

    if (this.checkCollision(newHead)){
        gameOver();
        return;
    }
```

```
    this.segments.unshift(newHead);

    if(newHead.equal(apple.position)){
        score++;
        apple.move();
    }else{
        this.segments.pop();
    }
}
```

2.9. 添加checkCollision 方法

每次为贪吃蛇的头部设置一个新的位置的时候，都必须检查碰撞。碰撞检测在游戏机制中是一个很常见的步骤，往往也是游戏编程中较为复杂 的一个方面。好在，在贪吃蛇游戏中，碰撞检测相对简单。 注意：该游戏中(0,0)这样的位置不是活动位置边界，已经是由灰色边框网格填充了。游戏中四个边界都有10像素宽的边界，所以贪吃蛇头部网格的新位置head.col === 0就“Game Over”了。 ##添加 keydown 事件处理程序

```
var directions = {
    37: "left",
    38: "up",
    39: "right",
    40: "down"
};

$("body").keydown(function(event){
    var newDirection = directions[event.keyCode];
    if(newDirection !== undefined){
        snake.setDirection(newDirection);
    }
});
```

2.10. 游戏入口程序

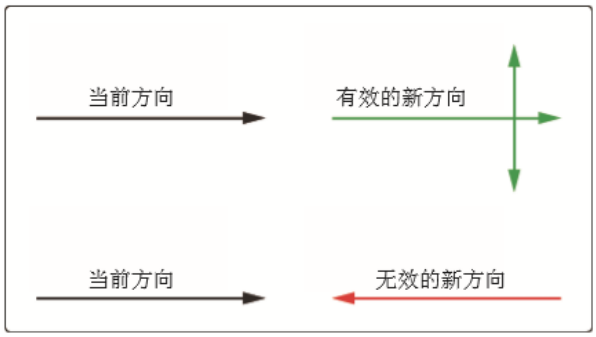
```
var intervalId = setInterval(function(){
    ctx.clearRect(0,0,width,height);
    drawScore();
    snake.move();
    snake.draw();
    apple.draw();
    drawBorder();
},100);
```

2.11. 移动苹果

由于边框的存在，苹果出现的范文只能（1~38）10，0和39都是边框。 调用 Math.floor（Math.random() 38），它给出了从0到37的一个随机数，然后，给结果加1以得到1到38之间的一个数字。

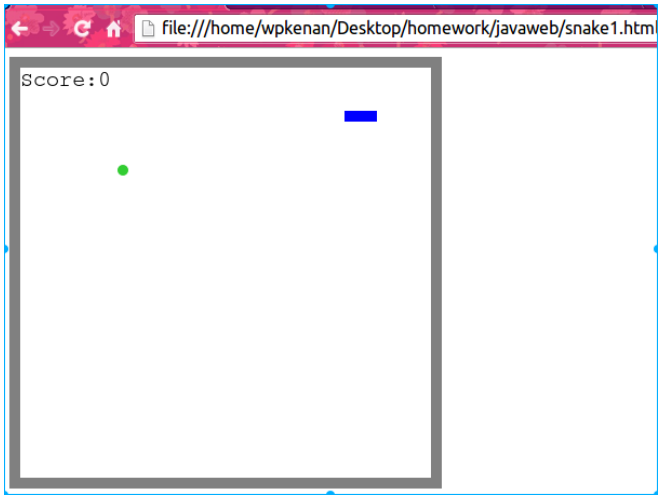
2.12. 添加 setDirection 方法

setDirection 还防止玩家调头以导致贪吃蛇立即碰到自己。例如，如果贪吃蛇向右移动，然后它突然向左转而不向上或向下移 以改变路径，那么它会和自己碰撞。这种现象叫作非法调头，因为我们不想让玩家这么做。也就是如果当前方向是向右，那么按键是向左直接return，不更新当前方向。

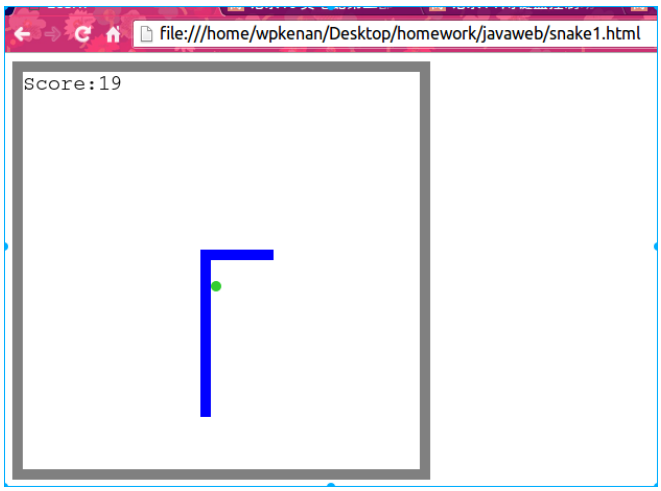


3. 运行结果

3.1. 开始



3.2. 中间过程



3.3. 结束

4. 完整代码

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Learn</title>
</head>
<body>
    <canvas id="canvas" width="400" height="400"></canvas>
    <script type="text/javascript" src="jquery-3.1.1.js"></script>
    <script type="text/javascript">
        var canvas = document.getElementById("canvas");
        var ctx = canvas.getContext("2d");
        var width = canvas.width;
        var height = canvas.height;

        //游戏面板看成是40个10*10的网格组成的
        var blockSize = 10;//每个网格的大小
        var widthInBlocks = width / blockSize;
        var heightInBlocks = height / blockSize;

        //每次贪吃蛇吃掉一个苹果的时候，分数加1
        var score = 0;

        //创建一个drawBorder函数来绘制围绕画布的灰色边框，1个块（10 像素）那么宽。
        var drawBorder = function(){
            ctx.fillStyle = "Gray";
            //绘制4条边框
            ctx.fillRect(0,0,width,blockSize);
            ctx.fillRect(0,height-blockSize,width,blockSize);
            ctx.fillRect(0,0,blockSize,height);
            ctx.fillRect(width-blockSize,0,blockSize,height);
        }

        //编写记分函数
        var drawScore = function(){
            ctx.font = "20px Courier";
            ctx.fillStyle = "Black";
            ctx.textAlign = "left";
            ctx.textBaseline = "top"; //文本左对齐
            ctx.fillText("Score:"+score,blockSize,blockSize);
        }

        //结束游戏
        var gameOver = function(){
            clearInterval(intervalId);
            ctx.font = "60px Courier";
            ctx.fillStyle = "Black";
            ctx.textAlign = "center";
            ctx.textBaseline = "middle";
            ctx.fillText("Game Over",width/2,height/2);
        }

        //绘制圆的方法
        var circle = function(x,y,radius,fillCircle){
            ctx.beginPath();
            ctx.arc(x,y,radius,0,Math.PI*2,false);
            if(fillCircle){
                ctx.fill();
            }else{
                ctx.stroke();
            }
        };

        //Block构造方法
        var Block = function(col,row){
            this.col = col;
            this.row = row;
        };

        //绘制方块
        Block.prototype.drawSquare = function(color){
            var x = this.col * blockSize;
            var y = this.row * blockSize;
            ctx.fillStyle = color;
            ctx.fillRect(x,y,blockSize,blockSize);
        }

        //绘制圆封装成Block的原型方法
        Block.prototype.drawCircle = function(color){
            var centerX = this.col * blockSize + blockSize/2;
            var centerY = this.row * blockSize + blockSize/2;
            ctx.fillStyle = color;
            circle(centerX,centerY,blockSize/2,true);
        }

        //检查Block是否再同一个位置已经是其他的Block了（如贪吃蛇前进，后面的块顶替前面块的位置，都是在同一个位置却
        Block.prototype.equal = function (otherBlock){
            return this.col === otherBlock.col && this.row === otherBlock.row;
        }

        //贪吃蛇构造方法
        var Snake = function (){
            this.segments = [
                new Block(7,5),
                new Block(6,5),
                new Block(5,5)
            ];

            this.direction = "right";
            this.nextDirection = "right";
        };

        //绘制贪吃蛇身体的每一个网格部分
        Snake.prototype.draw = function(){
            for(var i=0;i < this.segments.length; i++){
                this.segments[i].drawSquare("Blue");
            }
        };

        //创建一个新的网格加到当前贪吃蛇前进方向的头部
        Snake.prototype.move = function(){
            var head = this.segments[0];
            var newHead;
```

```
        this.direction = this.nextDirection;
        if(this.direction === "right"){
            newHead = new Block(head.col+1,head.row);
        }else if(this.direction === "down"){
            newHead = new Block(head.col,head.row+1);
        }else if(this.direction === "left"){
            newHead = new Block(head.col-1,head.row);
        }else if(this.direction === "up"){
            newHead = new Block(head.col,head.row-1);
        }

        if (this.checkCollision(newHead)){
            gameOver();
            return;
        }

        this.segments.unshift(newHead);

        if(newHead.equal(apple.position)){
            score++;
            apple.move();
        }else{
            this.segments.pop();
        }
    }

    //检查蛇的头部是否碰到墙体或者吃到自己的尾巴
    Snake.prototype.checkCollision = function(head){
        var leftCollision = (head.col === 0);
        var topCollision = (head.row === 0);
        var rightCollision = (head.col === widthInBlocks-1);
        var bottomCollision = (head.row === heightInBlocks-1);

        var wallCollision = leftCollision || topCollision ||
            rightCollision || bottomCollision;

        var selfCollision = false;

        for(var i=0;i<this.segments.length;i++){
            if(head.equal(this.segments[i])){
                selfCollision = true;
            }
        }
        return wallCollision || selfCollision;
    }

    //根据按下的键盘决定贪吃蛇下次的移动方向
    Snake.prototype.setDirection = function(newDirection){
        if(this.direction === "up" && newDirection === "down" ){
            return;
        }else if(this.direction === "right" && newDirection === "left"){
            return;
        }else if(this.direction === "down" && newDirection === "up"){
            return;
        }else if(this.direction === "left" && newDirection === "right"){
            return;
        }
        this.nextDirection = newDirection;
    }

    //苹果构造器
    var Apple = function(){
        this.position = new Block(10,10);
    };

    //添加原型方法，在苹果的坐标属性的位置上绘制一个苹果
    Apple.prototype.draw = function(){
        this.position.drawCircle("LimeGreen");
    };

    //移动苹果到一个新的随机位置
    Apple.prototype.move = function(){
        var randomCol = Math.floor(Math.random()*(widthInBlocks-2))+1;
        var randomRow = Math.floor(Math.random()*(heightInBlocks-2))+1;
        this.position = new Block(randomCol,randomRow);
    };

    //游戏开始时，创建贪吃蛇和苹果
    var snake = new Snake();
    var apple = new Apple();

    //游戏动画,加载script便签就得到间歇调用的执行。其他代码比如函数定义也是执行了，只不过都是函数定义
    var intervalId = setInterval(function(){
        ctx.clearRect(0,0,width,height);
        drawScore();
        snake.move();
        snake.draw();
        apple.draw();
        drawBorder();
    },100);

    //keyCode用更形象的字符串来表示方向
    var directions = {
        37: "left",
        38: "up",
        39: "right",
        40: "down"
    };

    //获取键盘事件
    document.onkeydown=function(event){
          var e = event || window.event || arguments.callee.caller.arguments[0];
          var newDirection = directions[e.keyCode];
            if(newDirection !== undefined){
                snake.setDirection(newDirection);
            }
    };

//        $("body").keydown(function(event){
//            var newDirection = directions[event.keyCode];
//            if(newDirection !== undefined){
```

```
 //               snake.setDirection(newDirection);
   //         }
     //   });

    </script>
</body>
</html>
```