**在了解我们需要操作的项目和内容之后，在脑海中大概绘制出一个思路，但是具体对不对，要付出实践才知道你的思路是否符合代码逻辑，是否达到你所想要的效果。**

**1、首先：创建2d画布，因为我们要在画布上绘制图形。**

**2、其二：我将在后期操作过程中用到的参数 进行初始化，或者设置默认；**

var stop\_status=true; //停止键是否有效，用来判断游戏结束后不能按停止键；

var stop; //停止动画

var canvas\_status=true; //初始未按停止键

//初始化

//球

var ballRadius = 10;//小球的尺寸

var ballColor = "#f44280";//小球颜色

var x = canvas.width/2;

var y = canvas.height-25;

//球移动速度（初始往左移动）

var dx = -2; //往左走

var dy = -2; //往上走

//接盘

var paddleHeight = 15;//接盘的高度

var paddleWidth = 75;//接盘的宽度

var paddleX = (canvas.width-paddleWidth)/2;//接盘位置

var rightPressed = false;//开始之后 键盘控制接盘右移动

var leftPressed = false;//开始之后 键盘控制接盘左移动

//bricks 顶部砖块

var brickRowCount = 5;//砖块行数

var brickColumnCount = 6;//砖块列数

var brickWidth = 70;//每一个砖块的宽度

var brickHeight = 20;//每一个装快的高度

var brickPadding = 6;//每一个砖块之间的距离

var brickOffsetTop = 35;//砖块整体距离顶部的距离

var brickOffsetLeft = 15;//砖块整体距离左侧的距离

var score = 0; //分数初始化

var lives = 3; //生命

**2-1、 在设置默认之后，var一个空的数组，用来盛放砖块（也就是初始化一个二维数组来存储砖块信息）**

//初始化二维数组 ，存储砖块信息 status:1砖块显示，0砖块消失

var bricks = [];

for(c=0; c<brickColumnCount; c++) {

bricks[c] = [];//存储列

for(r=0; r<brickRowCount; r++) {

bricks[c][r] = { x: 0, y: 0, status: 1 };

}

}

**2-2、接下来，将我们的小球绘制出来并且放到我们的画布当中下方正中间位置**

//画球

function drawBall() {

ctx.beginPath();//beginPath() 方法在一个画布中开始子路径的一个新的集合。

ctx.arc(x, y, ballRadius, 0, Math.PI\*2);//创建一个圆形 底部剧中

ctx.fillStyle = ballColor;//规定小球颜色

ctx.fill();//填充当前图像

ctx.closePath();//创建当前点到开始点的路径（绘制一条路径，形式是字母 L，然后绘制线条以返回开始点：）

}

**将小球放置在画布中**

$(function(){

drawBall();//球

})

**2-3、继续，将我们的接球滑盘绘制出来，并且放置在我们画布当中小球下方位置**

//滑盘

function drawPaddle() {

ctx.beginPath();//beginPath() 方法在一个画布中开始子路径的一个新的集合。

ctx.rect(paddleX, canvas.height-paddleHeight, paddleWidth, paddleHeight); //绘制矩形 居中底部

ctx.fillStyle = "#0000ff";//规定颜色

ctx.fill();//填充颜色

ctx.closePath();//规定画盘路径

}

**将滑盘放置在画布中**

$(function(){

drawBall();//球

drawPaddle();//滑盘

})

**2-4、接着将砖块逐个循环排列放置在画布中**

//砖块放置排列 画砖块

function drawBricks() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

if(bricks[c][r].status == 1) {

var brickX = (c\*(brickWidth+brickPadding))+brickOffsetLeft;//创建一个砖块所占的宽度

var brickY = (r\*(brickHeight+brickPadding))+brickOffsetTop;//创建一个砖块所占的高度

bricks[c][r].x = brickX;//砖块x轴的位置

bricks[c][r].y = brickY;//砖块的y轴位置

ctx.beginPath();//在一个画布中开始子路径的一个新的集合。

ctx.rect(brickX, brickY, brickWidth, brickHeight); //绘制矩形

ctx.fillStyle = "#8f63cc";//规定颜色

ctx.fill();//填充颜色

ctx.closePath();//路径

}

}

}

}

**将绘制好的砖块放置在画布中**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

})

**2-5、在画布顶端左侧添加分数值**

//分数

function drawScore() {

ctx.font = "16px Arial";//文字大小

ctx.fillStyle = "#f44280";//文字颜色

ctx.fillText("Score: "+score, 15, 20);//文字内容和 x轴 y轴 位置

}

**将分数添加到画布**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

})

**2-6、在画布顶端右侧添加生命值**

//生命

function drawLives() {

ctx.font = "16px Arial";//文字大小

ctx.fillStyle = "#f44280";//文字颜色

ctx.fillText("Lives: "+lives, canvas.width-70, 20);//文字内容和 x轴 y轴 位置

}

**将生命-值添加到画布中**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

})

**3、运动**

**3-1-、点击开始游戏让小球运动起来**

function draw() {

ctx.clearRect(0, 0, canvas.width, canvas.height);//清空canvas

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

//碰左右墙壁换方向

if(x + dx > canvas.width-ballRadius || x + dx < ballRadius) {

dx = -dx;

}

//到达顶部,

if(y + dy < ballRadius) {

dy = -dy;

}

x += dx;

y += dy;

stop=requestAnimationFrame(draw);//动画继续

}

**点击事件绑定**

$(function(){

// ctx.clearRect(0, 0, canvas.width, canvas.height);//在canvas中清除全部

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

$(".start").click(function(){//点击开始时

draw();

})

**3-2、小球到达底部 如果 滑盘接到小球则改变方向，如果未接到则生命值-1，并且滑盘和小球一起回到初始位置重新开始生值归0游戏重置**

function draw() {

ctx.clearRect(0, 0, canvas.width, canvas.height);//清空canvas

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

//碰左右墙壁换方向

if(x + dx > canvas.width-ballRadius || x + dx < ballRadius) {

dx = -dx;

}

//到达顶部,

if(y + dy < ballRadius) {

dy = -dy;

}

else if(y + dy > canvas.height-(ballRadius+paddleHeight)) {//底部

if(x > paddleX && x < paddleX + paddleWidth) {//盘子接住了球，改变方向

dy = -dy;

}

//如果盘子接不住小球

else {

lives--;//生命减少一个

if(lives==0) {

//游戏结束，重置参数

// drawGO(); //结束游戏

$(".start").attr("flage","flage");//设置或返回被选元素的属性值。

lives=4;//生命回归四条

score=0;//分数变为0

stop\_status=false;

return;

}

else {

x = canvas.width/2;

y = canvas.height-30;

dx = 3;

dy = -3;

paddleX = (canvas.width-paddleWidth)/2;//接盘位置回归中间

}}}

x += dx;

y += dy;

stop=requestAnimationFrame(draw);//动画继续

}

**3-3、当生命之为0 游戏重置之前在当前画布上添加GAME OVER 文字**

//Game over

function drawGO() {//游戏结束

ctx.font = "16px Arial";//字号

ctx.fillStyle = "#f44280";//颜色

ctx.fillText("GAME OVER",200,200);//文字内容 和坐标

}

**将上一步游戏结束注释解开**

**3-4、对接盘进行鼠标跟随移动操作**

**//移动鼠标控制接盘**

function mouseMoveHandler(e) {

var relativeX = e.clientX - canvas.offsetLeft;// e.clientX 事件属性返回当事件被触发时鼠标指针向对于浏览器页面（或客户区）的水平坐标。

if(relativeX > 0 && relativeX < canvas.width) {

paddleX = relativeX - paddleWidth/2;//接盘滑动的位置

}

}

**在点击开始中添加事件监听**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

$(".start").click(function(){//点击开始时

draw();

document.addEventListener("mousemove", mouseMoveHandler, false);//事件监听 滑盘鼠标跟随事件

})

})

1. **对生命之为0游戏重新开始进行重置恢复初始状态**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

$(".start").click(function(){//点击开始时

if($(".start").attr("flage")=="flage"){//设置或返回被选元素的属性值。//判断是否游戏结束之后可以继续点击开始游戏

///重新开始之后重置砖块

for(c=0; c<brickColumnCount; c++) {//循环列

for(r=0; r<brickRowCount; r++) {//循环行

bricks[c][r].status =1;//判断砖块是否显示

}

}

$(".start").attr("flage","");

draw();

document.addEventListener("mousemove", mouseMoveHandler, false);//事件监听 滑盘鼠标跟随事件

}

})

})

1. **对滑块进行键盘左右方向键绑定操作，空格键暂停绑定操作**

//键盘右键，左键控制接盘移动开关

function keyDownHandler(e) {

if(e.keyCode == 39) {//键盘右箭头数字

rightPressed = true;

}

else if(e.keyCode == 37) {//键盘左箭头数字

leftPressed = true;

}

else if(e.keyCode == 32){//键盘空客数字

if(stop\_status){//停止键是否有效，用来判断游戏结束后不能按停止键；

if(canvas\_status){

window.cancelAnimationFrame(stop);//可以取消该次动画。

canvas\_status=false;

}

else{

stop=requestAnimationFrame(draw);//动画继续

canvas\_status=true;

}

}

}

}

function keyUpHandler(e) {

if(e.keyCode == 39) {

rightPressed = false;

}

else if(e.keyCode == 37) {

leftPressed = false;

}

}

**对键盘操作进行事件监听，开始之后生效**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

$(".start").click(function(){//点击开始时

if($(".start").attr("flage")=="flage"){//设置或返回被选元素的属性值。//判断是否游戏结束之后可以继续点击开始游戏

///重新开始之后重置砖块

for(c=0; c<brickColumnCount; c++) {//循环列

for(r=0; r<brickRowCount; r++) {//循环行

bricks[c][r].status =1;//判断砖块是否显示

}

}

$(".start").attr("flage","");

draw();

//实时监听操作键盘和鼠标事件

document.addEventListener("keydown", keyDownHandler, false);

document.addEventListener("keyup", keyUpHandler, false);

document.addEventListener("mousemove", mouseMoveHandler, false);//事件监听 滑盘鼠标跟随事件

}

})

})

**控制键盘移动**

function draw() {

ctx.clearRect(0, 0, canvas.width, canvas.height);//清空canvas

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

collisionDetection();//当球与砖块碰撞时候的条件

//碰左右墙壁换方向

if(x + dx > canvas.width-ballRadius || x + dx < ballRadius) {

dx = -dx;

}

//到达顶部,

if(y + dy < ballRadius) {

dy = -dy;

}

else if(y + dy > canvas.height-(ballRadius+paddleHeight)) {//底部

if(x > paddleX && x < paddleX + paddleWidth) {//盘子接住了球，改变方向

dy = -dy;

}

//如果盘子接不住小球

else {

lives--;//生命减少一个

if(lives==0) {

//游戏结束，重置参数

drawGO(); //结束游戏

$(".start").attr("flage","flage");//设置或返回被选元素的属性值。

lives=4;//生命回归四条

score=0;//分数变为0

stop\_status=false;

return;

}

else {

x = canvas.width/2;

y = canvas.height-30;

dx = 3;

dy = -3;

paddleX = (canvas.width-paddleWidth)/2;//接盘位置回归中间

}

}

}

//键盘控制接盘移动

if(rightPressed && paddleX < canvas.width-paddleWidth) {//如果右移接盘的位置小于canvas宽度-接盘宽度

paddleX += 5;

}

else if(leftPressed && paddleX > 0) {//否则如果左移的位置大于0

paddleX -= 5;

}

x += dx;

y += dy;

stop=requestAnimationFrame(draw);//动画继续

}

1. **最后一步对当前点击事件添加暂停效果**

$(function(){

drawBall();//球

drawPaddle();//滑盘

drawBricks();//砖块

drawScore();//分数

drawLives();//生命

$(".start").click(function(){//点击开始时

stop\_status=true;//停止键是否有效，用来判断游戏结束后不能按停止键；

if($(".start").attr("flage")=="flage"){//设置或返回被选元素的属性值。//判断是否游戏结束之后可以继续点击开始游戏

///重新开始之后重置砖块

for(c=0; c<brickColumnCount; c++) {//循环列

for(r=0; r<brickRowCount; r++) {//循环行

bricks[c][r].status =1;//判断砖块是否显示

}

}

$(".start").attr("flage","");

draw();

//实时监听操作键盘和鼠标事件

document.addEventListener("keydown", keyDownHandler, false);

document.addEventListener("keyup", keyUpHandler, false);

document.addEventListener("mousemove", mouseMoveHandler, false);//事件监听 滑盘鼠标跟随事件

}

})

})