EXERCISE

（JavaScript Canvas DAY 03）

版本编号 1.0

JavaScript Canvas DAY03

1. 使用Canvas技术制作小球碰撞反弹的效果

注：讲师可以提前给学员演示一下动态的效果

运行结果如图1

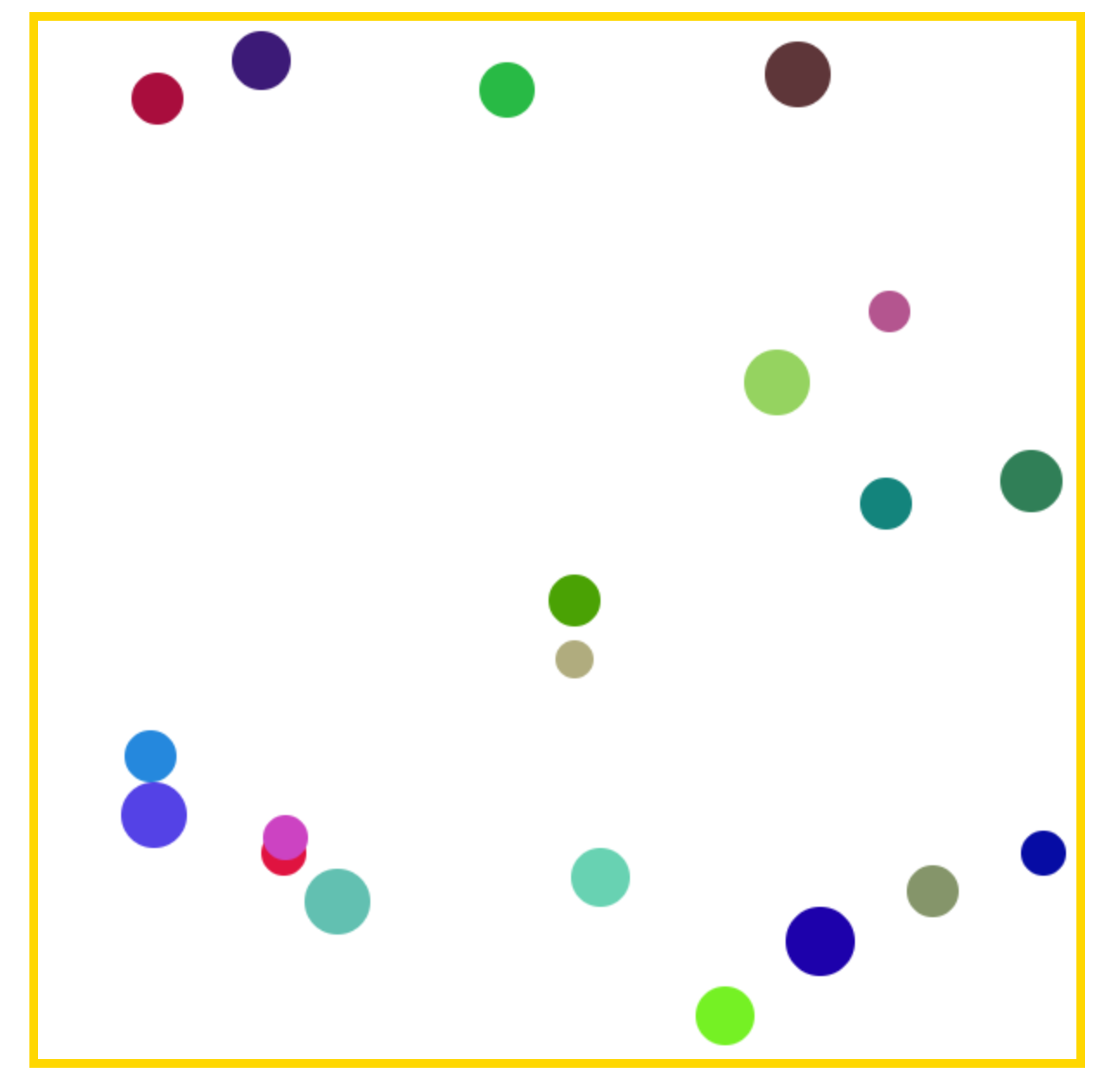


图- 1

* 参考答案

此案例需要用户创建html文件，进行网页布局，创建Canvas标签，使用js进行canvas的操作，封装小球类，实现小球碰壁反弹，并创建多个小球，实现网页的效果。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件

首先新建一个名为index的html文件，进行网页布局，代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#my-canvas {

display: block;

margin: 0 auto;

border: 5px solid gold;

}

</style>

</head>

<body>

<canvas id="my-canvas" width="600" height="600">

您的浏览器不支持canvas

</canvas>

</body>

</html>

步骤二：获取canvas的画笔

创建script标签，获取canvas元素，并获取画笔，代码如下所示：

//获取canvas

var myCanvas = document.getElementById("my-canvas");

var ctx = myCanvas.getContext("2d");

步骤三：封装小球类

封装随机函数，实现随机生成某个数。代码如下所示：

//随机函数

function random (x, y) {

return Math.floor(Math.random() \* (y - x + 1) + x);

}

创建构造函数，实现随机生成小球的半径，颜色，圆心，速度。代码如下所示：

//小球类

function Ball () {

//随机半径

this.r = random(10, 20);

//随机颜色

this.color = "rgb(" + random(0, 255) + "," +  random(0, 255) + "," + random(0, 255) + ")";

//随机圆心位置

this.x = random(this.r, myCanvas.width - this.r);

this.y = random(this.r, myCanvas.height - this.r);

//随机速度

this.speedX = random(2, 5) \* (random(0, 1) ? 1 : -1);

this.speedY = random(2, 5) \* (random(0, 1) ? 1 : -1);

}

创建小球的移动方法，进行碰壁的检测。代码如下所示：

//小球移动

Ball.prototype.move = function () {

this.x += this.speedX;

this.y += this.speedY;

//判断是否碰到左边界

if (this.x <= this.r) {

this.x = this.r;

this.speedX \*= -1;

}

//右边界

if (this.x >= myCanvas.width - this.r) {

this.x = myCanvas.width - this.r;

this.speedX \*= -1;

}

//上边界

if (this.y <= this.r) {

this.y = this.r;

this.speedY \*= -1;

}

//下边界

if (this.y >= myCanvas.height - this.r) {

this.y = myCanvas.height - this.r;

this.speedY \*= -1;

}

}

创建小球的绘制方法，小球的绘制。代码如下所示：

//画小球

Ball.prototype.draw = function () {

ctx.beginPath();

ctx.arc(this.x, this.y, this.r, 0, Math.PI \* 2, true);

ctx.fillStyle = this.color;

ctx.fill();

}

步骤四 ：创建多个小球

通过小球类生成多个小球，并存到数组中，方便后期使用。代码如下所示：

//创建小球对象

var balls = [];//存小球对象

for (var i = 0; i < 20; i++) {

var ball = new Ball();

balls.push(ball);

}

步骤五：开启动画

使用系统新增的方法requestAnimationFrame，执行小球的移动并绘制。代码如下所示：

//动画

function animation () {

ctx.clearRect(0, 0, myCanvas.width, myCanvas.height);

for (var i = 0; i < balls.length; i++) {

balls[i].move();

balls[i].draw();

}

requestAnimationFrame(animation);

}

animation();

然后运行HTML文件，按下鼠标，并进行滑动即可进行刮刮卡。

本案例的完整代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#my-canvas {

display: block;

margin: 0 auto;

border: 5px solid gold;

}

</style>

</head>

<body>

<canvas id="my-canvas" width="600" height="600">

您的浏览器不支持canvas

</canvas>

<script type="text/javascript">

//随机函数

function random(x, y) {

return Math.floor(Math.random() \* (y - x + 1) + x);

//return parseInt(Math.random() \* (y - x + 1) + x);

}

//获取canvas

var myCanvas = document.getElementById("my-canvas");

var ctx = myCanvas.getContext("2d");

//小球类

function Ball() {

//随机半径

this.r = random(10, 20);

//随机颜色

this.color = "rgb(" + random(0, 255) + "," + random(0, 255) + "," + random(0, 255) + ")";

//随机圆心位置

this.x = random(this.r, myCanvas.width - this.r);

this.y = random(this.r, myCanvas.height - this.r);

//随机速度

this.speedX = random(2, 5) \* (random(0, 1) ? 1 : -1);

this.speedY = random(2, 5) \* (random(0, 1) ? 1 : -1);

}

//小球移动

Ball.prototype.move = function() {

this.x += this.speedX;

this.y += this.speedY;

//判断是否碰到左边界

if(this.x <= this.r) {

this.x = this.r;

this.speedX \*= -1;

}

//右边界

if(this.x >= myCanvas.width - this.r) {

this.x = myCanvas.width - this.r;

this.speedX \*= -1;

}

//上边界

if(this.y <= this.r) {

this.y = this.r;

this.speedY \*= -1;

}

//下边界

if(this.y >= myCanvas.height - this.r) {

this.y = myCanvas.height - this.r;

this.speedY \*= -1;

}

}

//画小球

Ball.prototype.draw = function() {

ctx.beginPath();

ctx.arc(this.x, this.y, this.r, 0, Math.PI \* 2, true);

ctx.fillStyle = this.color;

ctx.fill();

}

//创建小球对象

var balls = []; //存小球对象

for(var i = 0; i < 20; i++) {

var ball = new Ball();

balls.push(ball);

}

//动画

function animation() {

ctx.clearRect(0, 0, myCanvas.width, myCanvas.height);

for(var i = 0; i < balls.length; i++) {

balls[i].move();

balls[i].draw();

}

requestAnimationFrame(animation);

}

animation();

</script>

</body>

</html>

1. 使用canvas技术，模拟两个小球在盒子内的碰撞效果

运行结果如图2

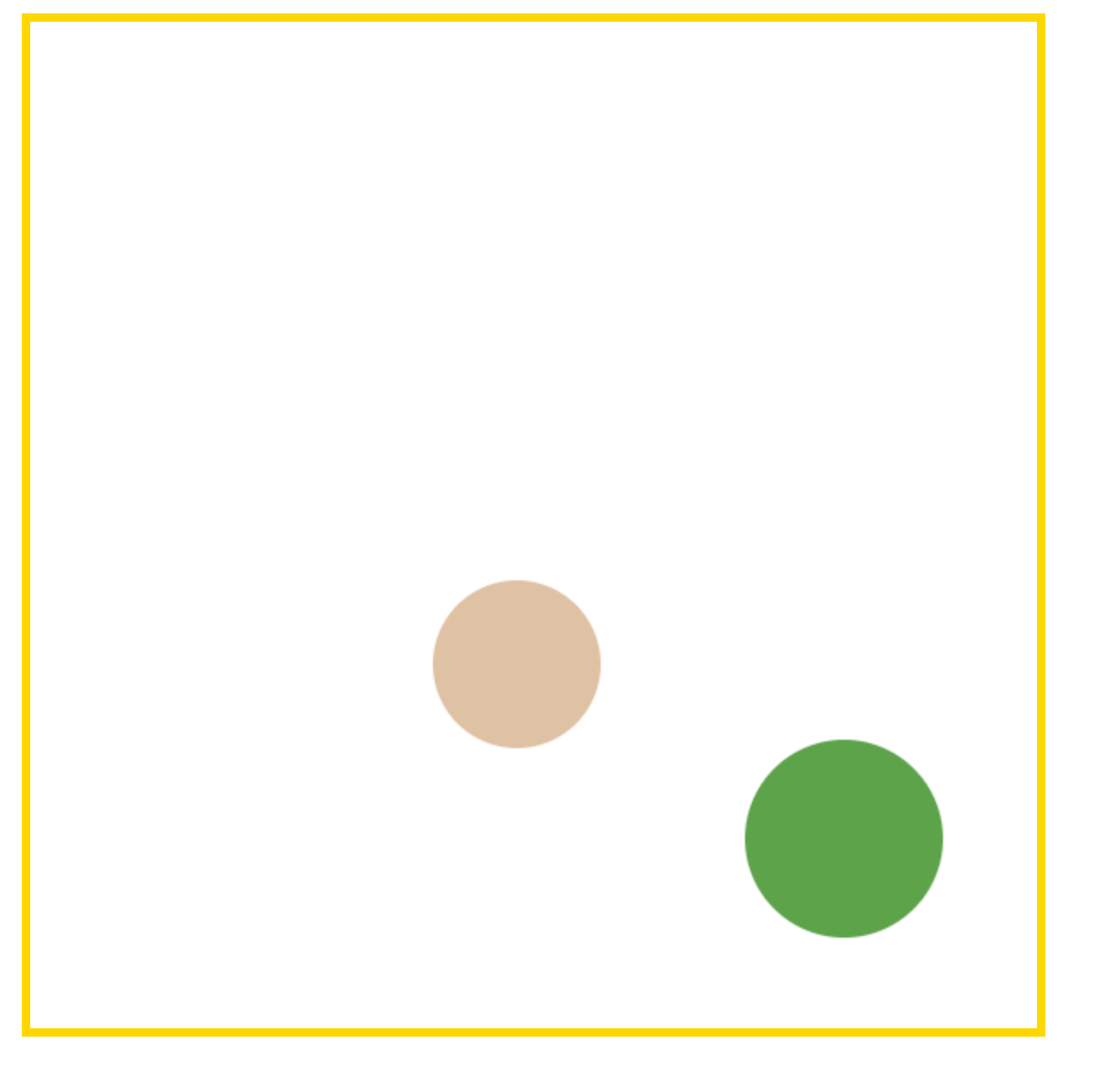


图- 2

* 参考答案

此案例需要用户创建html文件, 创建canvas标签，封装小球类，并生成2个小球，并进行小球之间的碰撞检测。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件，进行网页布局

首先新建一个名为index的html文件，创建canvas标签，并进行简单的修饰，代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#my-canvas {

display: block;

margin: 0 auto;

border: 5px solid gold;

}

</style>

</head>

<body>

<canvas id="my-canvas" width="600" height="600">

您的浏览器不支持canvas

</canvas>

</body>

</html>

步骤二：获取画笔

创建script标签，获取canvas元素，并获取绘制画笔。代码如下所示：

//获取canvas

var myCanvas = document.getElementById("myCanvas");

var ctx = myCanvas.getContext('2d');

步骤三：封装小球类

封装随机函数，实现随机生成某个数。代码如下所示：

//随机函数

function random (x, y) {

return Math.floor(Math.random() \* (y - x + 1) + x);

}

创建构造函数，实现随机生成小球的半径，颜色，圆心，速度。代码如下所示：

//小球类

function Ball () {

//随机半径

this.r = random(10, 20);

//随机颜色

this.color = "rgb(" + random(0, 255) + "," +  random(0, 255) + "," + random(0, 255) + ")";

//随机圆心位置

this.x = random(this.r, myCanvas.width - this.r);

this.y = random(this.r, myCanvas.height - this.r);

//随机速度

this.speedX = random(2, 5) \* (random(0, 1) ? 1 : -1);

this.speedY = random(2, 5) \* (random(0, 1) ? 1 : -1);

}

创建小球的移动方法，进行碰壁的检测。代码如下所示：

//小球移动

Ball.prototype.move = function () {

this.x += this.speedX;

this.y += this.speedY;

//判断是否碰到左边界

if (this.x <= this.r) {

this.x = this.r;

this.speedX \*= -1;

}

//右边界

if (this.x >= myCanvas.width - this.r) {

this.x = myCanvas.width - this.r;

this.speedX \*= -1;

}

//上边界

if (this.y <= this.r) {

this.y = this.r;

this.speedY \*= -1;

}

//下边界

if (this.y >= myCanvas.height - this.r) {

this.y = myCanvas.height - this.r;

this.speedY \*= -1;

}

}

创建小球的绘制方法，小球的绘制。代码如下所示：

//画小球

Ball.prototype.draw = function () {

ctx.beginPath();

ctx.arc(this.x, this.y, this.r, 0, Math.PI \* 2, true);

ctx.fillStyle = this.color;

ctx.fill();

}

步骤四 ：创建2个小球

通过小球类生成2个小球，并存到数组中，方便后期使用。代码如下所示：

//创建小球对象

var balls = [];//存小球对象

for (var i = 0; i < 2; i++) {

var ball = new Ball();

balls.push(ball);

}

步骤五：小球的碰撞检测

先封装两个球类的碰撞检测函数，方便后期的使用。代码如下所示：

//小球的碰撞检测

function ballCrash (ball1, ball2) {

//两个小球的距离

var distance = Math.sqrt(Math.pow(ball1.x - ball2.x, 2) + Math.pow(ball1.y - ball2.y, 2));

if (distance <= ball1.r + ball2.r) {

return true;

} else {

return false;

}

}

使用系统的requestAnimationFrame方法，并移动实时修改小的位置，并实现动画效果。代码如下所示：

//动画

function animation () {

ctx.clearRect(0, 0, myCanvas.width, myCanvas.height);

for (var i = 0; i < balls.length; i++) {

balls[i].move();

balls[i].draw();

//移动后做碰撞检测

for (var j = 0; j < balls.length; j++) {

//判断是不是同一个小球

if (balls[i] == balls[j]) {

continue;

}

//检测

if (ballCrash(balls[i], balls[j])) {

//互换速度

var tempX = balls[i].speedX;

balls[i].speedX = balls[j].speedX;

balls[j].speedX = tempX;

var tempY = balls[i].speedY;

balls[i].speedY = balls[j].speedY;

balls[j].speedY = tempY;

}

}

}

requestAnimationFrame(animation);

}

animation();

然后运行HTML文件，在输入框里输入名字，并点击确认按钮，即可看到效果。

本案例的完整代码如下所示：

代码量较大，请查看02\_练习源码/2小球碰撞

1. 使用canvas技术，实现烟花的效果，效果如下图

注：讲师可以先带学生看一下效果

运行结果如图3

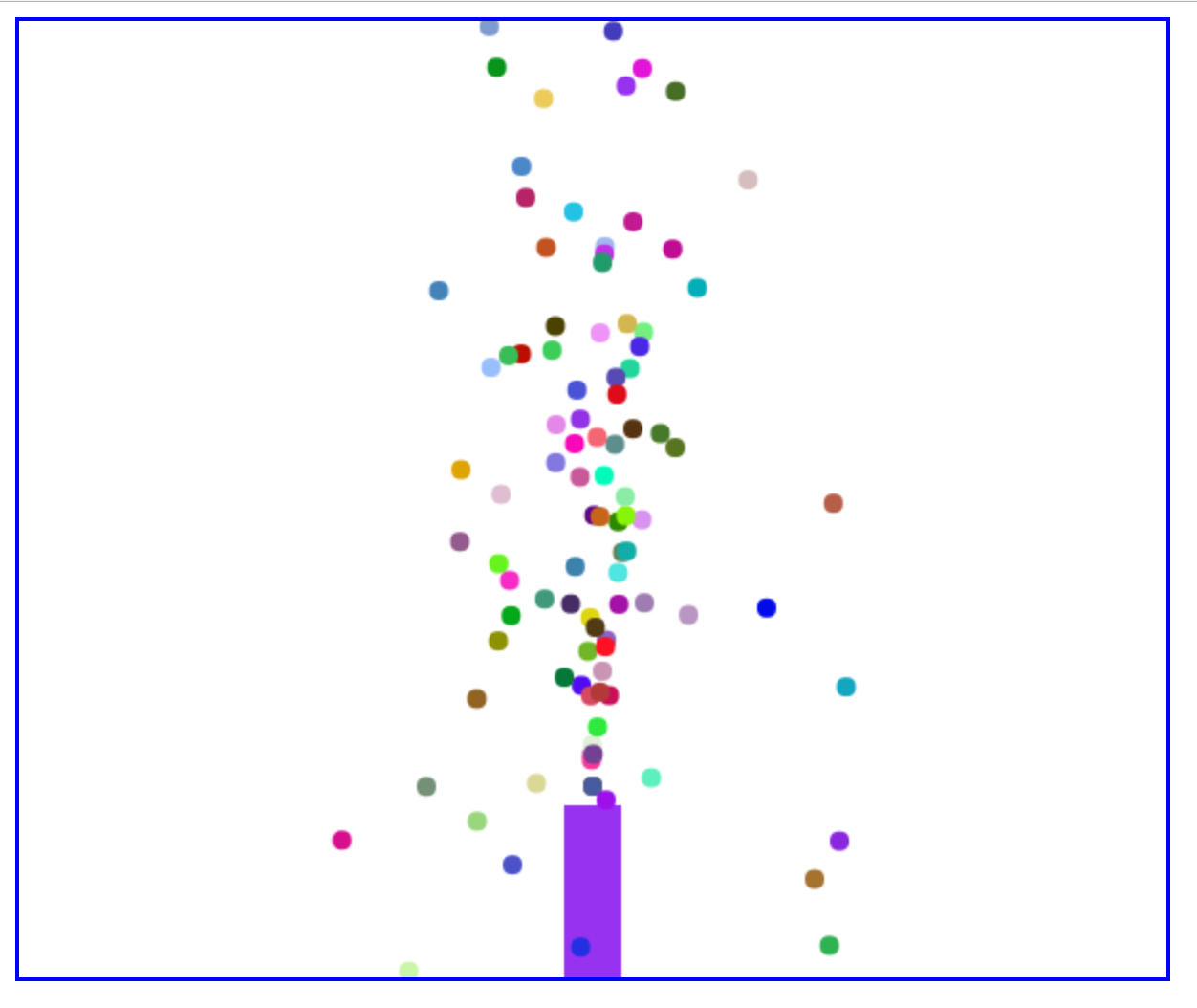


图- 3

* 参考答案

此案例需要用户创建html文件, 创建canvas标签，绘制喷桶，并封装颗粒类，随机数小球，模型重力感应效果。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件，进行网页布局

首先新建一个名为index的html文件，创建canvas标签，代码如下所示：

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<canvas id="MymyCanvasvasArea" width="600" height="500" style="border:2px solid blue;">你的浏览器不支持myCanvasvas</canvas>

</body>

</html>

步骤二：定义变量

定义变量记录画板，画笔，颗粒集合，颗粒数量，重力系数。代码如下所示：

//定义变量

var myCanvas; //canvas

var ctx; //画笔

var particles = []; //小球集合

var numParticles = 100; //小球数量

var gravity = 0.4; //重力系数

步骤三：创建小球

使用for循环，快速生成多个小球，并存到数组中，方便使用。代码如下所示：

for(var i = 0; i < numParticles; i++) {

r = parseInt(Math.random() \* 255);

g = parseInt(Math.random() \* 255);

b = parseInt(Math.random() \* 255);

var particle = {

x: myCanvas.width / 2,

y: myCanvas.height - 100,

vx: Math.random() \* 4 - 2,

vy: Math.random() \* -14 - 7,

pcolor: 'rgb(' + r + ',' + g + ',' + b + ')'

};

particles.push(particle);

}

步骤四：小球移动

封装小球移动的方法，实时更新小球的位置，并考虑重力效果。代码如下所示：

//更新小球的位置

function update() {

for(var i = 0; i < numParticles; i++) {

var part = particles[i];

part.x = part.x + part.vx; //adding the speed

part.y = part.y + part.vy; //adding the speed

part.vy += gravity;

if(part.x > myCanvas.width || part.x < 0 || part.y > myCanvas.height || part.y < 0) {

part.x = myCanvas.width / 2;

part.y = myCanvas.height - 100;

part.vx = Math.random() \* 4 - 2;

part.vy = Math.random() \* -14 - 7;

}

}

}

步骤五：小球绘制

封装小球绘制的方法，方便后期使用。代码如下所示：

function draw() {

var r, g, b;

for(var i = 0; i < numParticles; i++) {

var part = particles[i];

ctx.beginPath();

ctx.fillStyle = part.pcolor;

ctx.arc(part.x, part.y, 5, 0, Math.PI \* 2, false);

ctx.fill();

}

}

步骤五：执行动画

使用系统的requestAnimationFrame方法，实时调用方法。代码如下所示：

//动画

function animate() {

reqAnimFrame = window.mozRequestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.msRequestAnimationFrame ||

window.oRequestAnimationFrame;

clear();

drawBarrel();

draw();

update();

reqAnimFrame(animate);

}

然后运行HTML文件，在输入框里输入名字，并点击确认按钮，即可看到效果。

本案例的完整代码如下所示：

代码量较大，请查看02\_练习源码/3小球动画