# 文档说明（目录）

该文档用于介绍编程思想以及过程中遇到的一些bug及解决方案等等。详情以目录形式列出以便查询。

目录

[文档说明（目录） 1](#_Toc479252499)

[原生js版 2](#_Toc479252500)

[Html部分 2](#_Toc479252501)

[Js部分 2](#_Toc479252502)

[Canvas基本操作 2](#_Toc479252503)

[图片预加载 2](#_Toc479252504)

[背景图片绘制 3](#_Toc479252505)

[文字绘制 3](#_Toc479252506)

[地鼠图片绘制 3](#_Toc479252507)

[地鼠定义 4](#_Toc479252508)

[地鼠生成 5](#_Toc479252509)

[地鼠上下动作 6](#_Toc479252510)

[点击响应 7](#_Toc479252511)

[开始游戏循环 7](#_Toc479252512)

[总结 8](#_Toc479252513)

[PIXI.js版 8](#_Toc479252514)

游戏有两个版本，一个是基于原生js开发的初版，另一个是基于PIXI.js进行开发的正式版。

# 原生js版

## Html部分

新建一个html文件，在body中插入canvas标签，并给canvas设置id值。原生js版完全基于canvas实现游戏功能。

<canvas id="myCanvas"></canvas>

在body中插入script标签，可在其中直接写代码或使用src引入外部js文件。（注：使用原生js编写的外部js文件只能在body中引用，不能在head中引用）

## Js部分

把js想象成java，function相当于java中的方法，其余内容相当于java的main函数中的内容，js被加载时立即执行，执行顺序从上到下（大概）。

### Canvas基本操作

首先获取到html中的canvas元素，随后创建context对象。（固定写法）

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

var context = c.getContext("2d");

之后的绘制全在context上进行。Canvas基本绘制教程参考：

<http://www.cnblogs.com/tim-li/archive/2012/08/06/2580252.html>

### 图片预加载

在实际绘制图片前加载图片，确保图片能绘制成功。

var bgReady = false;

var background\_img = new Image();

background\_img.src="imgs/background.jpg";

background\_img.onload = function(){

bgReady = true;

};

### 背景图片绘制

if(bgReady)

context.drawImage(background\_img,120,0,1000,300,0,0,1000,300);

drawImage有三种语法，参考

<http://www.w3school.com.cn/tags/canvas_drawimage.asp>

背景、洞、地鼠都使用drawImage进行绘制。洞和背景数量固定且无动作，可直接绘制。本例中有5个地鼠洞，1个背景。地鼠洞使用for循环均匀绘制在一行上。

for(var i = 0;i<5;i++){ ctx.drawImage(grass\_img,10+(i\*190),80); }

地鼠有动作且数量不定，绘制方法虽一样，但需要做点改动。

注：使用一个方法来放置所有的绘制内容。function render() 每次刷新时调用该方法即可重新绘制画面。

### 文字绘制

context.font = "30px Arial";

context.fillText(("内容"),10,40);

使用fillText绘制文字，定义x，y坐标。使用font设置文字字体及字号。需要写入的文字内容为字符串形式。

### 地鼠图片绘制

地鼠有正常和被打击两种形态，初始化一个二维数组来存放这两种形态的图片。

var zombie\_object = [zombie\_img,clicked\_img];

（zombie\_img及clicked\_img都已进行[图片预加载](#_图片预加载)）

初始化一个空数组存放未知个数的地鼠。var zombie = new Array(); 数组中存放的内容格式为JSON，JSON中定义一个参数为obj，可取0和1两个值；定义参数x，y表示地鼠在画布中的起始坐标。

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

context.drawImage(zombie\_object[zombie[i].obj],zombie[i].x,zombie[i].y);

}

遍历数组，绘制所有的地鼠，绘制哪种形态的地鼠由obj的值决定。随着地鼠上下移动，x、y值也会相应的变化，可完成动画绘制。

### 地鼠定义

上文提到需要初始化一个空数组来存放未知个数的地鼠，其中的内容格式为JSON。JSON参数均可自定义，下面以本例中使用到的的参数进行介绍：

|  |  |  |
| --- | --- | --- |
| 参数名称 | 取值 | 含义 |
| obj | 0或1 | 地鼠是否被击中，击中为1否则为0 |
| flag | false或true | 地鼠运动方向，true为向下false向上 |
| randomNum | randomNum | 0~4间取随机值，决定哪个洞出现地鼠 |
| x | zombie\_x[randomNum] | 将五个洞的地鼠x坐标存在数组中，根据随机值选取对应x值进行绘制。 |
| y | 280 | 固定的起始高度，位于洞以下 |
| speed | 6 | 地鼠上下的速度 |
| stay | 15 | 地鼠到达顶端时停止的次数 |

### 地鼠生成

计算当前时间与初始时间的间隔，大于某个时间则生成新地鼠，初始时间重新赋值为当前时间，进行下一次计时。

新建一个更新类来完成地鼠生成及动作实现。在方法外定义全局变量，deltaTime（间隔时间），then（初始时间）。

function update(){

var now = Date.now();

var delta = now-then;

if(delta>deltaTime){

}

}

当if返回true时，执行随机函数获取randomNum，使用该随机数生成一个地鼠JSON数据放入地鼠数组中。zombie.push(JSON.parse(“JSON内容参考定义”))；

#### Tip：地鼠不连续随机到一个洞

定义一个tempIndex用于存放上一次随机出的randomNum，如果tempIndex的值与新的随机值相同则再次取随机，直到不同。

while(tempIndx==randomNum)

randomNum = Math.floor(Math.random()\*5);

tempIndx = randomNum;

（注：Math.floor向下取整，Math.random() 0~1随机）

#### Tip：地鼠出现速度逐渐加快

减小时间间隔即可加快地鼠出现速度。

if(deltaTime>400) deltaTime -= 10;

设置一个最小时间间隔和依次递减量，该判断放在if(delta>deltaTime)内部，每次出现一次地鼠则加速一次。

### 地鼠上下动作

改变y坐标即可实现地鼠上下的位置变化，使用for循环依次改变地鼠数组中的每一个地鼠元素y坐标。该例中，当y=60时到达顶端，y=280到达低端。

if(zombie[i].y<=60){

zombie[i].flag = "true";

}

到达顶端时，flag初始值为false，此时地鼠向上运行，当false值设为true，提示地鼠应该向下运动。

if(zombie[i].flag == "true"){

zombie[i].y += parseInt(zombie[i].speed);

}else

zombie[i].y -= zombie[i].speed;

（注：JSON中的内容均为字符串，因此此处的true和false不可作为boolean格式使用；speed同理也为字符串，如果要实现相加需要使用parseInt将字符串转换为数字）

当地鼠经过一次完整的上下运动后，从地鼠数组中移除该地鼠。如果此时地鼠未被击中过，则计一次错过。

if(zombie[i].y >=280){

if(zombie[i].obj == 0)

miss++;

zombie.splice(i,1);

}

地鼠在顶端时停留一段时间，递减stay，当stay为零时才开始执行向下运动。

if(zombie[i].stay>0)

zombie[i].stay -=1;

### 点击响应

鼠标或键盘操作如下

window.addEventListener("mousedown",function(e){ …},false);

参考<http://www.runoob.com/jsref/dom-obj-event.html>

由参数e可获取到鼠标点击的坐标

var mouse\_x=e.pageX;var mouse\_y=e.pageY;

使用for循环依次获取地鼠数组中每个地鼠所在的最小矩形，判断鼠标点击地方是否位于矩形中。如果击中设置obj为1，得分增加并退出当前for循环（因为理论上鼠标一次只能点击一个地鼠）。当obj为1时，不再进行击中判断（防止重复计分）。

### 开始游戏循环

function main(){

update();

render();

requestAnimationFrame(main);

};

main();

main函数放在方法外时，意味着js被调用后会自动执行该函数。依次调用update（生成地鼠、地鼠动作更新）和render（绘制图像），使用requestAnimationFrame完成定时更新。参考<http://www.webhek.com/post/requestanimationframe.html>

## 总结

原生版可用于熟悉canvas基本操作，缺点就是看上去地鼠是动态的物体，其他都是静态的，实际上绘制时，是地鼠、洞、背景都在不断的重新绘制，因此每次执行点击操作时是针对整个画布进行操作，而不是体感的仅针对地鼠操作（大概）。

# PIXI.js版

PIXI.js官网：<http://www.pixijs.com/>

建议先把github中提供的examples运行一遍，就大概懂得基本操作了，官网中提供的demo可作为后期的参考。