**Step 2 定义原型**

接下来定义一下英雄的原型。

var hero = {

speed: **256**,

x: canvas.width/**2**,

y: canvas.height/**2**

}

这个原型也很好理解，每秒钟英雄可以移动256个像素，英雄初始的位置为画布中央(x,y分别为坐标)。

接下来轮到怪物了。: )

function monster() {

this.x = Math.random() \* canvas.width;*//初始为止随机*

this.y = Math.random() \* canvas.height;

this.speed = **100**;

this.xDirection = **1**;*//默认移动方向为x轴正方向(以左上角为零点，下方和右方为正)*

this.yDirection = **1**;*//同样也为y轴正方向*

this.move = function (modifier) {*//移动函数*

this.x += this.xDirection \* this.speed \* modifier;

this.y += this.yDirection \* this.speed \* modifier;

if (this.x >= canvas.width - **32**) {*//碰撞返回部分*

this.x = canvas.width - **32**;

this.xDirection = -**1**;

} else if (this.x <= **0**) {

this.x = **0**;

this.xDirection = **1**;

} else if (this.y >= canvas.height - **32**) {

this.y = canvas.height - **32**;

this.yDirection = -**1**;

} else if (this.y <= **0**) {

this.y = **0**;

this.yDirection = **1**;

}

};

}

var monsterSum = **0**;

var monsterList = new Array();

monsterList[monsterSum] = new monster();

前面用var已经创建了英雄的实例，但是monster我们只建立了类而已，接下来要实例化。monsterSum表示怪物的数量，为方便，按照c的习惯，从0开始技术，即0表示有一个怪物。monsterList用来表示一个存怪物对象的数组，然后顺带新建一个怪物。

**Step 3 游戏动起来！**

先添加一个事件来接收键盘的动作，上下左右用对应的ascii码。因为游戏并不是摁一下方向键，就移动一段距离。而是，判断一个时间间隔内的动作(最后的动作，中间有可能会变化，故用数组保存最后结果)。

var keysDown = {};

addEventListener("keydown", function (e) {

keysDown[e.keyCode] = **true**;*//如果有"keydown"这个动作，即摁下某键，就会存进keysDown数组*

}, **false**);

addEventListener("keyup", function (e) {

delete keysDown[e.keyCode];

});

下面上主函数

var main = function () {

var now = Date.now();

var delta = now - then;

Move(delta / **1000**);*//每次间隔时间根本不是1ms，比1ms要大得多*

Draw();

Check();

then = now;

}

main函数就是主函数，就是一个刷新所执行的函数。now、then两个变量记录两次刷新的时间间隔，这个时间间隔并不是固定的，一般为几百毫秒。delta 是两者之差，单位为毫秒。下面依次解释各个函数:Move()用来计算英雄和怪物的新位置。Draw()用来画背景、人物、文字。Check()用来检查，怪物和英雄是否相撞。

*Move():*

var Move = function (modifier) {

if (**38** in keysDown) {

hero.y -= hero.speed \* modifier;

}

if (**40** in keysDown) {

hero.y += hero.speed \* modifier;

}

if (**37** in keysDown) {

hero.x -= hero.speed \* modifier;

}

if (**39** in keysDown) {

hero.x += hero.speed \* modifier;

}

if (hero.x >= canvas.width - **32**) {

hero.x = **0**;

}else if (hero.x <= **0**) {

hero.x = canvas.width - **32**;

}

if (hero.y >= canvas.height - **32**) {

hero.y = **0**;

}else if (hero.y <= **0**) {

hero.y = canvas.height - **32**;

}

for (var i = **0**; i <= monsterSum; i++) {

monsterList[i].move(modifier);

}

}

这里很好理解，判断这段时间间隔英雄的动作。算出新位置后，判断英雄是否跑出了画布，跑出了就从另一头出来(感谢 @昭曈 的创意)。接下来依次调用各个怪物的move函数，计算他们的新位置。

*Draw():*

var Draw = function () {

if (bgReady) {

ctx.drawImage(bgImage, **0** ,**0**);

}

if (heroReady) {

ctx.drawImage(heroImage, hero.x, hero.y);

}

if (monsterReady) {

for (var i = **0**; i <= monsterSum; i++)

ctx.drawImage(monsterImage, monsterList[i].x, monsterList[i].y);

}

ctx.fillStyle = "rgb(250, 250, 250)";

ctx.font = "24px Helvetica";

ctx.textAlign = "left";

ctx.textBaseline = "top";

last = Date.now() - start;

ctx.fillText(last/**1000**, **32**, canvas.height - **64**);

}

前三个函数类似，就是如果准备好了图片就画东西上去(前面的三个ready函数派上了用场)。下面先定义了文字的style，然后计算出时间间隔last，然后画上去。

*Check():*

var Check = function () {

if (monsterSum != Math.floor(last / **5000**)){*//如果时间经过5秒就增加一个怪兽实例*

monsterSum ++;

monsterList[monsterSum] = new monster();

}

for (var i = **0**; i <= monsterSum; i++) {

if ((monsterList[i].x - **32**) <= hero.x

&& hero.x <= (monsterList[i].x + **32**)

&& (monsterList[i].y - **32**) <= hero.y

&& hero.y <= (monsterList[i].y + **32**)) {

end = Date.now();

alert("你坚持了" + (end - start)/**1000** + "秒");

End();

}

}

}

第一步是 如果经过5秒就增加一个怪兽，然后下面一个个判断怪兽与英雄是否接触。(这里用的是矩形，@小樟 说可以用圆心，感兴趣的可以试试: ) )

大家注意到了下面用到了一个end函数。下面补充，如果不停止一直刷新就浏览器就一直动了，故要有个结束函数。

*End():*

var End = function () {

if (bgReady) {

ctx.drawImage(bgImage, **0** ,**0**); *//留住背景*

}

window.clearInterval(timer);

return;

}

用到的clearInterval()稍后会说到。

**Step 4 程序入口**

var then = Date.now();

var start = then;

timer = setInterval(main, **1**);

定义了初始的then和start，接下里用了setInterval(),意为每隔1ms就执行一次main函数。1ms是虚指，就是立即执行的意思。要想停止就用前面的clearInterval()函数。

到这里为止一个完整的游戏就写好了~ : )

后来 @志谦 实践出了一个bug，就是窗口如果最小化后，会一直计时，但不会被撞。问题在于高级的浏览器(当然没在说IE : ) )，默认如果最小化后，会终止interval、timeout这类函数的执行，以节约资源。但我们计时的方法是结束时间减去开始时间，所以就造成这个刷分的bug。感谢 @小樟 提供了main函数递归，全局变量表示是否停止的办法。

感谢 @润卿 的前端帮助， @SysLab 的测试~

哦，第一次就那么没了~

[Html5](http://billyellow.com/tags/Html5), [Javascript](http://billyellow.com/tags/Javascript)



***[billyellow](http://billyellow.com/)***

billyellow, 黄震昕, pythoner, 产品经理;

* [Twitter](http://twitter.com/billyellow)
* [GitHub](http://github.com/billyellow)
* [微博](http://weibo.com/billyellow)

Powered by [Letterpress](https://github.com/an0/Letterpress).