**第15章 边界检测**

这一章我们给大家介绍了边界检测的4种情况：边界限制、边界生成、边界环绕和边界反弹。

**15.6.1 边界限制**

边界限制，指的是通过边界检测的办法来限制物体的运动范围，使得其无法超出这个运动范围，而只限在范围内运动。

**语法：**

if (ball.x < ball.radius) {

//小球“碰到”左边界时

} else if (ball.x > cnv.width - ball.radius) {

//小球“碰到”右边界时

}

if (ball.y < ball.radius) {

//小球“碰到”上边界时

} else if (ball.y > cnv.height - ball.radius) {

//小球“碰到”下边界时

}

**说明：**

这里注意一个关键字眼：“碰到”。当小球碰到边界时，此时小球在画布里面，小球中心与画布边界的距离刚好是半径。

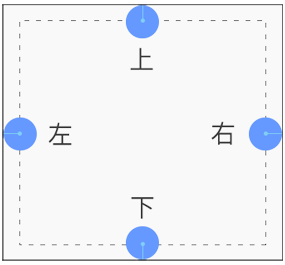


图15-13 碰到边界判断

**15.6.2 边界环绕**

边界环绕，指的是当物体从一个边界消失后，它就会从对立的边界重新出现，从而形成一种环绕效果。

**语法：**

if(ball.x < -ball.radius){

//小球“完全超出”左边界时

}

if(ball.x>cnv.width + ball.radius){

//小球“完全超出”右边界时

}

if(ball.y<-ball.radius){

//小球“完全超出”上边界时

}

if(ball.y>cnv.height + ball.radius){

//小球“完全超出”下边界时

}

**说明：**

这里注意一个关键字眼：“完全超出”。当小球完全超出边界时，此时小球在画布外面，小球中心与画布边界的距离刚好也是半径。

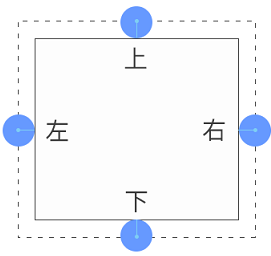


图15-14 超出边界判断

**15.6.3 边界环绕**

边界生成，指的是物体完全超出边界之后，会在最开始的位置重新生成。

**语法：**

if (ball.x < -ball.radius ||

ball.x > cnv.width + ball.radius ||

ball.y < -ball.radius ||

ball.y > cnv.height + ball.radius) {

……

}

**说明：**

当这4种情况任何一种为true时，都表示物体已经完全超出了边界。

**15.6.4 边界反弹**

边界反弹，指的是物体触碰到边界之后就会反弹回来，就像现实世界中小球碰到墙壁反弹一样。

**语法：**

//碰到左边界

if (ball.x < ball.radius) {

ball.x = ball.radius;

vx = -vx;

//碰到右边界

} else if (ball.x > canvas.width - ball.radius) {

ball.x = canvas.width - ball.radius;

vx = -vx;

}

//碰到上边界

if (ball.y < ball.radius) {

ball.y = ball.radius;

vy = -vy;

//碰到下边界

} else if (ball.y > canvas.height - ball.radius) {

ball.y = canvas.height - ball.radius;

vy = -vy;

}

**说明：**

边界反弹跟边界环绕的判断条件是不一样的。在边界环绕中，判断的是小球“完全移出”边界的情况。而在边界反弹中，判断的是小球“刚刚碰到”边界的情况。

边界检测代码不是死的，我们应该根据实际开发需求来改进，而不是直接套进去就完事了。在边界检测中，共通的是它的思想，而不是它的语法。

其实，对于这些边界检测方法，我们也可以封装在工具函数库里面的，这个小伙伴们可以自行实现一下。