Animate the runner only when on the platform



**runner.left**

**runner.left + runner.width**

**platform.offset + platform.left**

**platform.offset + platform.left + platform.width**

Just some thoughts on objects available to solve the problem of the runner animates only when on the platform. As the runner is set to just the level 1 platform, let’s generalize the problem to the runner is in the ‘shadow’ of a platform … the runner is either under or above a platform.

This implies that we would need to cycle through the list of platforms to test this condition. Is like the code for testing whether a sprite is in view:

Now we need to design the code to loop through the platform data and switch the runner animation accordingly. Let’s look at this next class after the break.

OK. Here we are after the break. If you look at the javascript file that is attached, I had added a class asset called ***this.RUNNER\_ON\_TRACK***. And if you follow this variable you will find it in the functions ‘setSpriteOffsets()’ and ‘drawSprites()’. Initially, I checked for any platform which overlaps the runner’s position … which turned out to be extremely stupid on my part.

In the drawSprites function, we check the RUNNER\_ON\_TRACK boolean and set the *this.runner.runAnimationRate* accordingly.

Of course, the most difficult job is to figure out which functions are being called in each clock cycle. And, how to calculate the displacement of the platforms. Well, this didn’t work as I wished it would. So, I decided to put everything in the drawSprites function. Since the platforms appear before the runner in the Sprites array, the overlap calculation would precede the need for the check on RUNNER\_ON\_TRACK. See if this works.

drawSprites: function() {

var sprite;

this.RUNNER\_ON\_TRACK = false;

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

sprite = this.sprites[i];

if (sprite.visible && this.isSpriteInView(sprite)) {

this.context.translate(-sprite.hOffset, 0);

if (sprite.type === ‘platform’) {

// Cycle through all the platforms and see if any platform ‘straddles’ the

// left edge of the runner … the value 50

// If it does, set RUNNER\_ON\_TRACK to ‘true’.

}

if (sprite.type === 'runner') {

// Now, for the runner, when false set the runner.runAnimationRate to 0,

// otherwise set it to the RUN\_ANIMATION\_RATE class attribute of snailBailt.RUN\_ANIMATION\_RATE

}

sprite.draw(this.context);

this.context.translate(sprite.hOffset, 0);

}

}

},

I did learn that a few well placed console.log() output commands can assist greatly when trying to understand the flow of the code. It helped me determine when functions and variables weren’t being reset. And I had to truly understand the attributes assigned to each sprite … in particular the platforms and runner.

Challenge:

Edit the code to animate the runner only on the platforms which are track 1.