How to Make an HTML5 Game, Part 3: Animating a Spritesheet

Before we get into this tutorial, make sure you've done your homework in the form of Part 1: The Basic Engine and Part 2: Jumping, Falling, and Stopping. And, of course, I hope you've played a few rounds of *Super Markup Man*, the game these HTML5 tutorials are based on.

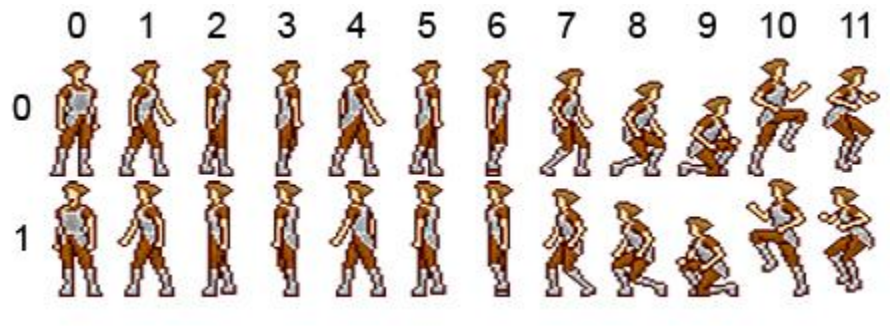
As always, I've already bundled up the new game engine in a downloadable zip file so you can follow along in the code. This is a pretty easy update, too. All we're doing today is animating our character so he can walk, jump, and fall in style. We accomplish this by using a spritesheet, an image that contains every frame we could possibly want for our character's movement:

While we *could* just tell our program to replace every image file with a new image file as we need them, that's not a very good way to go about animating something. It's easier to keep everything inside one image, then tell the program to only draw a small section of that image at a time. Thus, in the *images* folder, the *markup-man.png* file has been replaced with *spritesheet.png*. We'll load this new image into our game in the *graphics.js* file on line 52:

var spriteSheet = loadImg('images/spritesheet.png');

That's the only thing we need to do with *graphics.js*. We won't even touch the main *game.js* file in this update. Everything else is taken care of in *player.js*. So let's open that file and look at the new variables on line 8: **frameX**, **frameY**, and **delay**. **frameX** and **frameY** help us control what part of the spritesheet we want to draw. **delay** is a variable that adds a little extra time in between updates. Remember, our main game loop calls **player.draw()** every 17 milliseconds, which is a little too fast for animating. So we'll use **delay** to slow that part of it down.

If you scroll down to line 64, where the **draw()** function starts, you'll see that the **delay** value increases on every update. Once **delay** is greater than 4, *then* we can start changing the animation frames. That's where the **frameX** and **frameY** values come into play. If we number each frame on the spritesheet, we can see what these variables need to equal in order to get the frame we want:



So when the player is jumping, we can set **this.frameX = 10**. When the player is falling (determined by checking the **velocity**), we'll set **this.frameX = 11**. We need to take into account left and right movement, though. That's why there are two rows on the spritesheet. Go back up to lines 21 and 30, where we're checking if the player is moving left or right. We'll update the **frameY** value at the same time. Now when the player presses left, **this.frameY = 1** (the bottom row).

The code that ties all of this together is located on line 99. The **drawImage()** function has been replaced with a new one:

ctx.drawImage(this.sprite, this.frameX*this.width, this.frameY*this.height, this.width, this.height, this.x, this.y, this.width, this.height);

Yikes, that's a lot of stuff! Before, we were just telling the program to draw the graphic at the player's **x** and **y** location on the canvas. Since we've switched to using a spritesheet, we first have to tell the program what part of the spritesheet we want. Basically, we're "cutting out" a piece of it. Let's rewrite that function in normal English:

ctx.drawImage(our spritesheet, where to start "cutting" left and right, where to start "cutting" up and down, how wide the "cut out" will be, how tall the "cut out" will be, where to start drawing this on the canvas's x axis, where to start drawing on the y axis, how wide to make the graphic, how tall to make it);

And that's how it's done! You've probably noticed that we aren't using the crouching frames (7-9) for anything yet, though. In *Super Markup Man*, that's the animation that plays when the player picks up or drops an HTML block. While I could tell you in a fourth tutorial how to do that, wouldn't it be more fun to figure it out on your own? Yeah… I think it would. But if you get stuck, don't hesitate to ask questions. That's how we learn!