Aston University
Birmingham

Tutorial 2

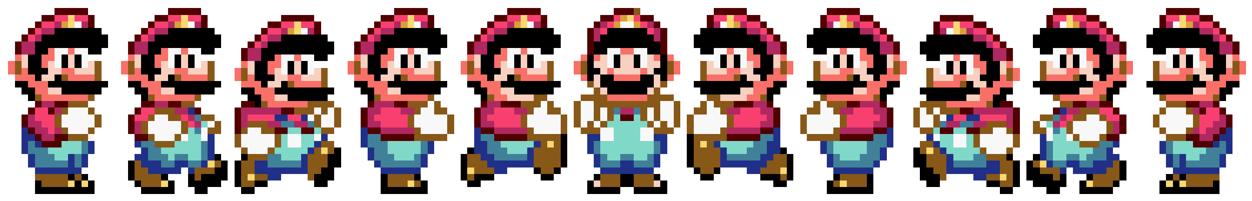
Animating Mario

In Tutorial 1 you managed to get mario walking around the screen, this tutorial will teach you how to animate mario which will make him more realistic!

Open up tutorial2.html in firefox, you will see the output of the last tutorial, mario with the ability to move him left and right.

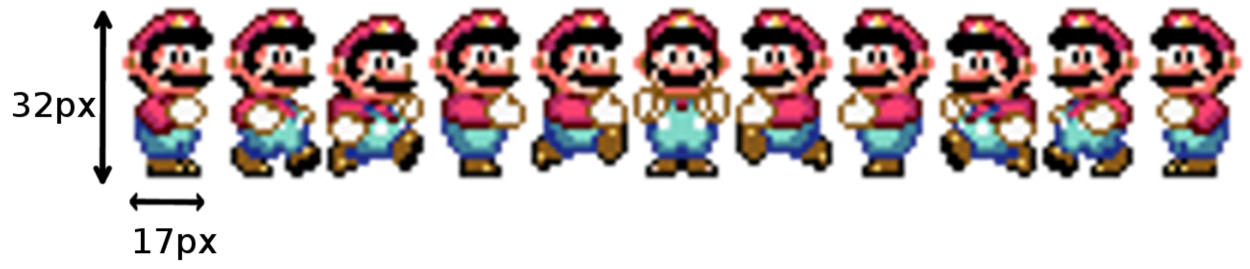
Step 1

So you now are able to move Mario around the screen but it looks rather silly, he is not actually walking and you can make him move backwards. To overcome this we can add animations to him, very simply. If you open up *mario-sprite.png* in the *img* folder you will something like this:



Now if you compare this to what you see in your browser you will see that the first mario is used throughout, this is because our Player class defaults to using the first image.

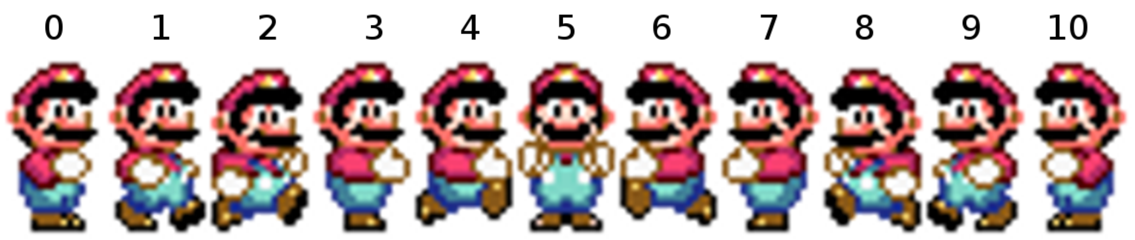
You may also recall that we passed through a *spriteX* and a *spriteY* parameter when we created the Player object before. These are used to tell our code how big each separate image is. If you look at the image below you can see that each mario is 32 pixels high and 17 pixels wide.



By telling the Game this information, the images can be split up correctly, this then allows us to loop through specific 'frames' allowing us to make mario more life-like! To be able to do this we need to create some animations with the *addAnimation* function that is invoked on the *player* variable. The *addAnimation* function takes 3 parameters:

1. *name* : The name of the animation
2. *frames* : An array of the frames to use in the animation, so will be in the form [0,1,2,3], the order given will be how they are played out.
3. *fps* : This is how many times you want to go through your frames in one second, 10 tends to work well but play with this number to see what works best

You are going to want to create two different animations, one for left and one for right, if you look at the images below you will see the frames needed for each.



In programming you always start at 0 and not 1 so for our right animation you will need the frames 0,1,2,and 3. The 4th frame is mario jumping. For our left animation what frames do you think should be used? (The 6th frame is mario jumping to the left)

You will need to place your code in the *create* function as you are creating animations. The function is called on the player object so will look something like:

```
player.addAnimation('right', [0,1,2,3], 10);
```

Make sure you have this in your *create* function and go ahead and create the left animation on the line below.

So you have now created your animations but they are not yet played, to do this we call the *playAnimation* function on the player object. The *playAnimation* function takes one parameter, this is the name of the animation to play. So to play the right animation you will place the code inside your previously created if statement that checks if the right directional key is pressed so your whole if block should now look something like:

```
if(left.isDown()) {

        player.moveX(2);

        player.playAnimation(‘left’);
}
```

Go and add in a similar line of code in your if block for the left directional key and refresh your *tutorial1.html* in Firefox. Move mario left and right and he will be walking! Theres just one problem, if you stop pressing one of the directional keys his animation keeps on playing causing him to still want to walk!

To overcome this you will want to use the *stop* function on your Player object to stop any animations. You will want to add this in your update function under your if block with an else statement. You currently have some code in the form of:

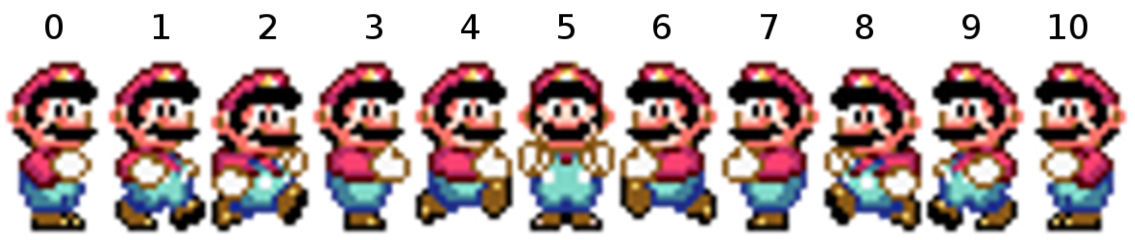
```
if(left.isDown()) {

        player.moveX(-2);

        player.playAnimation('left');

}
else if(right.isDown()) {

        player.moveX(2);

        player.playAnimation('right');
}
```

You will need to add in your else block in a similar way that you added in the else if block so your code will now look like:

```
if(left.isDown()) {

    player.moveX(-2);

    player.playAnimation('left');

}
else if(right.isDown()) {

    player.moveX(2);

    player.playAnimation('right');

} else {


}
```

Inside your *else* block you will want to invoke the *stop* function on your player object. If you load up *tutorial1.html* again you should see that mario stops walking when you stop pressing down on the directional keys. The only problem now is that mario always faces left, now why is this?

If you look back to this image:



You may notice that it is always image 0, this is that by default the Player class has image 0 as the default stopping class. Ideally we would like it to be image 5 as it looks like he is just standing there. To do this we can use the *setStopFrame* function on the player object to set what frame we want to be shown when nothing is happening. This should be placed in the create method. Mario should now be facing us whenever he is not moving!

Add extra for jumping