Tutorial – Basic Sprite Animations

In this tutorial we'll be integrating the animated sprite code presented in this week's lesson with our own code from last week's tutorial.

Adding Animated Spites:

1. Download the sprite.js script from AIE Portal and add it to your project. Don't forget to update index.html
2. First we'll create some constant variables so that we can reference our animations a bit easier.
   Add the following code to the top of the player.js file:

```
var LEFT = 0;
var RIGHT = 1;

var ANIM_IDLE_LEFT = 0;
var ANIM_JUMP_LEFT = 1;
var ANIM_WALK_LEFT = 2;
var ANIM_IDLE_RIGHT = 3;
var ANIM_JUMP_RIGHT = 4;
var ANIM_WALK_RIGHT = 5;
var ANIM_MAX = 6;
```

   The first two variables, LEFT and RIGHT will be used when keeping track of the player's current direction. We need to keep track of this so that we can use the correct animation depending on which direction the player is facing.

   The following variables will be used when setting the current animation. This allows us to set an animation by name, and not just using 'magic' numbers.

3. Next we need to create the sprite animations in the constructor of our player. This code will replace the static image we've been using up till now.

   Update the player constructor as follows:

```
var Player = function() {
	this.sprite = new Sprite("ChuckNorris.png");
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[0, 1, 2, 3, 4, 5, 6, 7]);
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[8, 9, 10, 11, 12]);
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]);
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[52, 53, 54, 55, 56, 57, 58, 59]);
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[60, 61, 62, 63, 64]);
	this.sprite.buildAnimation(12, 8, 165, 126, 0.05,
		[65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78]);

	for(var i=0; i<ANIM_MAX; i++)
	{
		this.sprite.setAnimationOffset(i, -55, -87);
	}

	~~this.image = document.createElement("img");~~
	this.position = new Vector2();
	this.position.set( 9*TILE, 0*TILE );

	this.width = 159;
	this.height = 163;

	~~this.offset = new Vector2();~~
	~~this.offset.set(-55,-87);~~

	this.velocity = new Vector2();

	this.falling = true;
	this.jumping = false;

	this.direction = LEFT;

	~~this.image.src = "hero.png";~~
};
```

The for loop here will allow us to set the same offset for all animations. This is so that when we position Chuck Norris somewhere, his feet will be placed at the position we specify.

Notice we've also added a new variable called 'direction' and set it to the value RIGHT.

Oh, and if you haven't already, you can remove the variable for rotation. We won't need that anymore and not having it simplifies our drawing code.

4. Call this.sprite.update() at the top of your update function:

```
Player.prototype.update = function(deltaTime)
{
      this.sprite.update(deltaTime);

      . . .
}
```

5. In your draw() function, call the sprite's drawing function

```
Player.prototype.draw = function()
{
   this.sprite.draw(context, this.position.x, this.position.y);
}
```

Run your game now and you should see Chuck Norris gettin' his idle on.

Changing Animations:

So far Chuck Norris is looking pretty good. But when we move around the screen things don't look right. We can't use the idle animation for all our movements.

We need to add some code to our update function so that the sprite animation changes according to our keyboard input. When the player presses jump, Mr. Norris should switch to the jump animation – and use either the left or right jump animation according to the current direction.

The following changes all take place in the player's update() function/

1. Start by adding code to switch to the left walk animation whenever the left key is down:

```
if(keyboard.isKeyDown(keyboard.KEY_LEFT) == true) {
      left = true;
      this.direction = LEFT;
      if(this.sprite.currentAnimation != ANIM_WALK_LEFT)
            this.sprite.setAnimation(ANIM_WALK_LEFT);
}
```

2. Add similar code for when the right key is pressed:

```
if(keyboard.isKeyDown(keyboard.KEY_RIGHT) == true) {
      right = true;
      this.direction = RIGHT;
      if(this.sprite.currentAnimation != ANIM_WALK_RIGHT)
            this.sprite.setAnimation(ANIM_WALK_RIGHT);
}
```

If you run your game now, you'll see that the sprite changes to the left or right walk animation whenever you press a direction key.

The only problem is that the animation never returns to the idle animation. We'll address that now.

3. We'll change our two *if* statements to a series of *if-else* statements. Our logic will be 'if the left key isn't pressed then check the right key, if the right key isn't pressed then return to the idle animation.

   Update the code as follows:

```
if(keyboard.isKeyDown(keyboard.KEY_LEFT) == true) {
      . . .
}
else if(keyboard.isKeyDown(keyboard.KEY_RIGHT) == true) {
      . . .
}
else {
      if(this.jumping == false && this.falling == false)
      {
            if(this.direction == LEFT)
            {
               if(this.sprite.currentAnimation != ANIM_IDLE_LEFT)
               this.sprite.setAnimation(ANIM_IDLE_LEFT);
            }
            else
            {
               if(this.sprite.currentAnimation != ANIM_IDLE_RIGHT)
               this.sprite.setAnimation(ANIM_IDLE_RIGHT);
            }
      }
}
if(keyboard.isKeyDown(keyboard.KEY_SPACE) == true)
{
      jump = true;
      if(left == true) {
            this.sprite.setAnimation(ANIM_JUMP_LEFT);
      }
      if(right == true) {
            this.sprite.setAnimation(ANIM_JUMP_RIGHT);
      }
}
```

   Notice our use of the direction variable to ensure we return to the correct idle animation.

   Run your game now and see how things are looking.

4. Our final modification is to change to the jump animation at the appropriate time. This isn't difficult, we just need to put the code in the right place (and it's not when we press the jump key, it's when we set the this.jumping variable to true)

   Find the line in the update() function that sets this.jumping = true. We want to add the sprite switching code below this line.

```
if (jump && !this.jumping && !falling)
{
      // apply an instantaneous (large) vertical impulse
      ddy = ddy - JUMP;
      this.jumping = true;
      if(this.direction == LEFT)
            this.sprite.setAnimation(ANIM_JUMP_LEFT)
      else
            this.sprite.setAnimation(ANIM_JUMP_RIGHT)
}
```

   Run this now. It almost looks right.

   Notice how if you hold down the left or right key when jumping Chuck Norris will walk iin mid-air. We can fix that.

5. In the code that switches to the left or right walking sprite, add a check so that we don't change the sprite when jumping:

```
if(keyboard.isKeyDown(keyboard.KEY_LEFT) == true) {
      left = true;
      this.direction = LEFT;
      if(this.sprite.currentAnimation != ANIM_WALK_LEFT &&
            this.jumping == false)
            this.sprite.setAnimation(ANIM_WALK_LEFT);
}
else if(keyboard.isKeyDown(keyboard.KEY_RIGHT) == true) {
      right = true;
      this.direction = RIGHT;
      if(this.sprite.currentAnimation != ANIM_WALK_RIGHT &&
            this.jumping == false)
            this.sprite.setAnimation(ANIM_WALK_RIGHT);
}
```

Awesome work. If everything is running smoothly you should now have a player character that uses an animated sprite, and the sprite changes in response to the actions of the player.

Exercises:

Add the animations for climbing and shooting. If you think you can you can try to implement these actions, but they are covered in a later lecture. For now, prepare your code by adding these animations to the animated sprite.

If you have enemies or other objects in your game, add animated sprites for these objects. Even your splash screen could use an animated sprite!