Spritesheets Tasks

Animating a Spritesheet

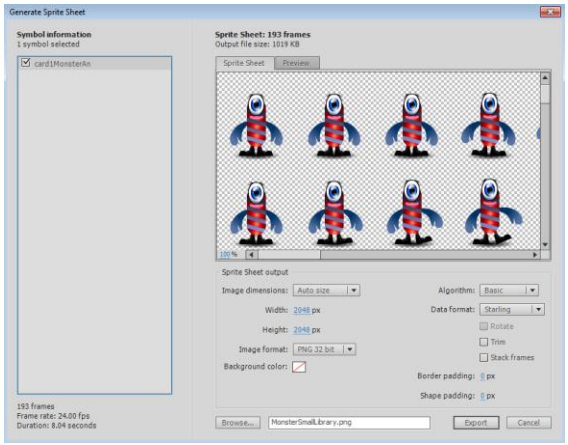
Definition

A sprite-sheet is a collection of frames from an animation and graphic assets that are bundled into one graphic to reduce memory consumption. This is used in casual and mobile games because of memory and graphic card constraints.

Creating a spritesheet with Flash CS 6

Flash CS6 has a the new method of outputting a spritesheet from selected assets in the library.

Select a graphic in the library and in right mouse button contextual menu press “Generate Spritesheet ”.



This allows you to generate a spritesheet from an animation inside a movieclip.

You can set the size, select a background colour or transparency and change a data format that facilitates the integration of the spritesheet into a games framework (such as EaselJS).

When generating an animation try to avoid animation where a large part of the image stays static, instead isolate the animation in a nested clip and create your spritesheet from this. Also avoid animation where the object travels over a great distance, for example in a motion tween, as this will signficantly increase the filesize of the generated spritesheet. The technique is ideal for single User Interface elemens, such as buttons and for animation of different states of characters.

Task

1. Create your own spritesheet from a Flash Movieclip

The spritesheet is loaded into a HTML document with the following Javascript code:

```
<script type="text/javascript" src="raf.js"></script>
<script type="text/javascript">
	var ctx;
	var count = 0;
	var x;
	var y;
	var img = new Image();
	img.src = "sprite_sheet.png";
	img.onload = draw;

	function draw() {
		requestAnimationFrame(draw);
		x = (count % 9)* 212;
		y = Math.floor(count /9) * 201;
		ctx.drawImage(img, x, y, 212, 201, 0,0, 212,201);
		if(count== 149) {
			count= 0;
		} else {
			count++
		}
	}

</script>
```

Explanation:

- The variables ctx holds the 2d context of the canvas as before. The variables count, x, y are necessary for selecting the proper frame of the img. The `img` variable stores a loaded spritesheet.
- The main work of the animation is done inside the draw method. This method gets executed once the image has completed loading.
- The call to the `requestAnimationFrame` repeatedly runs the draw function again.
- In order to animate we will need to select a portion of the entire image and display it. Next, we display the next image that is beside it for frame 2 and so forth. When we reach the end of the row, we will move down to row number 2. The calculations for the coordinates for this operation are down inside the draw method. They are relying on a counter (`var count`) and are stored in the variables x and y. The values 212 and 202 are the width and height of each frame on the spritesheet respectively. The value 9 is the number of columns in each row on the spritesheet. These values need to be modified for individual spritesheet animations.
- The `ctx.drawImage` cuts out the particular section of the image and places it on the canvas.

- The if… then statement increments the count variable up to the maximum number (149) of frames and then resets it to 0.

Task

2. Animate the Spritesheet to display the animation in a Javascript request animation loop

When running the example the animation loop will run in the top left corner but the pixels of theprevious frames are still visible. Next we will need to delete the pixels from the previous frame before we draw the new one. We achieve this by using the ctx.clearRect command.

```
function draw() {
	requestAnimationFrame(draw);
	ctx.clearRect(0,0,212,201);
	x = (count % 9)* 212;
	y = Math.floor(count /9) * 201;
	ctx.drawImage(img, x, y, 212, 201, 0,0, 212,201);
	if(count== 149) {
		count= 0;
	} else {
		count++
	}
}
```

Task

3. Modify the code so that pixels of previous frames get wiped before the next frame is drawn.

The position of the spritesheet is fixed in the top left corner. Next we will animate the position of the animated sprite. Recall that the 6th and 7th argument of the `ctx.drawImage()` method represent the x and y coordinates where the Image is drawn to the canvas. In order to animate it we just need to change these values here (currently they are set to 0) to variables such as dx and dy and update their values.

```
function draw() {
	requestAnimationFrame(draw);
	ctx.clearRect(0,0,212,201);
	x = (count % 9)* 212;
	y = Math.floor(count /9) * 201;
	ctx.drawImage(img, x, y, 212, 201, dx++ ,dy++, 212,201);
	if(count== 149) {
		count= 0;
	} else {
		count++
	}
}
```

Task

4. Make the Spritesheet move to the bottom left of the browser window.

Finally we can create a simple eventlistener that listens to keypresses. If they correspond to particular keys we can change movement of the sprite in aparticular direction by updating the values of dx and dy.

Task

5. Modify the code so that the spritesheet moves when you press the direction key: “a. “s”.”w” and “z”.

Summary:

You have learnt how to link create a spritesheet and animate it using the requestAnimationFrame in Javascript. You are done. Well done.