Aston University
Birmingham

Tutorial 4

Adding Enemies

In this tutorial we shall be recreating the classic game Space Invaders!

To start you will want to have a look at the Enemy object here. You may notice that it is quite similar to the player class, you can give it an image/spritesheet, give it different animations and move it. The difference is that you can create multiple enemies in this one object, this means that you can create a group of them and tell them all what to do at once. This works very well in this tutorial as we have this big group of aliens we want to all do the same thing. It may be good in some example but bad in others, if you want to have separate enemies then you can just create separate enemy objects.

Step 1

You will want to open up tutorial3.html in Firefox, you will see a small ship at the bottom of the screen that can be moved left and right with the arrow keys!

The first thing that you will want to do is create an enemy variable and initialize it in the *preload* function. The enemy object takes 5 parameters:

1. *game* : This is our game object
2. *name* : This is the name of our enemy, used as a reference for the image
3. *image* : This is the string reference of the image to use
4. *spriteX* : This is the height of your image
5. *spriteY* : This is the width of your image

The image is "img/*space_invader_sprite.png",* the *spriteX* in this case is 24 and the *spriteY* is 16 so you should have something similar to this:

```
enemy = new Enemy(game, 'invader', "img/space_invader_sprite.png", 24, 16);
```

Okay so we now need to go ahead and create an enemy, this can be done with the *createEnemySpriteSheet* function that takes two parameters:

1. *x* : The x position to place this enemy
2. *y* : The y position of the enemy

So go ahead and place an enemy in the game in any location you want. If you refresh the browser you should see your enemy there, but how do we go about adding in more? We could add in multiple lines of this but this would get messy if we want 40 enemies.

Step 2

This is where loops come in, loops allow us to execute a certain amount of blocks of code as many times as you like. A very good example of where loops are used are in the *update* function, your game is running a constant loop where the code in your *update* function is executed.

You can read up on the different loops **here,** the most simple type is a while loop, this will continue executing code whilst something is true. The other type is a *for* loop which is slightly more advanced but tends to be cleaner to read. The for loop contains three statements:

1. A line of code that is executed before the start of the loop, this is usually creating a variable
2. A condition to check for, for example *i < 10*
3. A line of code executed after each loop, for example I++ which adds 1 to the variable i

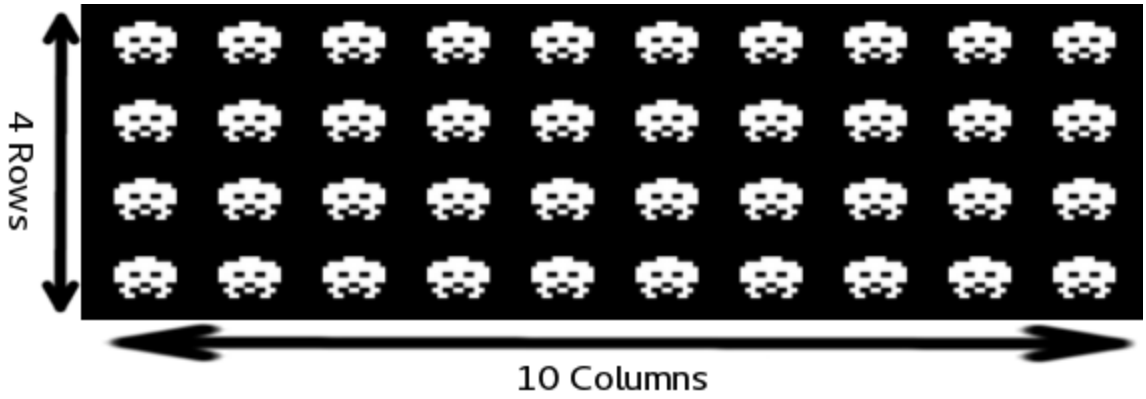
A good example is below:

```
for(var i = 0; i < 10; i++) {

}
```

So if we break this down we are creating our variable, we then check if that variable is less than 10; if this is the case we can execute the contents of the loop. Then after one loop we increase the value of our variable i by 1. So how many times do we go through this loop?

Okay so back to our game, we want to have 4 columns of 10 rows which would look something like this:



So how would we code this? The best way is to create the first row of 10, then create one directly underneath. The best way to do this is with two 'for loops' inside each other. The first loop is for the 4 columns and then the second loop is for the row. So we should have something like this:

```
for(var i = 0; i < 4; i++) {

    for(var j = 0; j < 10; j++) {

        //create enemy!

    }

}
```

Now all that is left to do is to create an enemy where it says to, so inside the second loop! To do this we want to create two variables, x and y, we will multiply values against the i and j values to position the enemy correctly.

The coordinate values represent the values of i and j for each invader in the form (j, i) this is due to the way we set up the loops. So the value of j and i is 0 in the top lefthand corner, in the top right hand corner j is 9 and i is 0.

The next thing we want to consider is the space between each invader, if you look at the image size of each, they have a width of 24 and a height of 16.

We want to ensure that the values for x and y are bigger than these otherwise they will overlap. We want to assign the x and y values of the invaders to the value of j and i multiplied by something respectively. This is the same as positioning the next invader a certain value away from the previous one. So using the width offset of 40 and the height offset of 30 each invader is 40 pixels away from the previous one in that column and each row is 30 pixels below the previous one.

So inside your loop (under the existing code) we will have something like:

```
var x = (j * 40);
var y = (i * 30);

enemy.createEnemySpriteSheet('invader', x, y);
```

Feel free to experiment with the values of the width and height offset but ensure that they are bigger than the width and height of the imager otherwise there will be collisions!