Aston University
Birmingham

Tutorial 3

Simple Collisions

In this tutorial we shall be looking at adding platforms to your game that allow your character to jump around the level. This will be achieved by using the ReusableObject. We are going to be looking at:

- adding gravity
- adding platforms
- adding collision with platforms

Step 1

You will want to open up *tutorial2.js* in your text editor and *tutorial2.html* in Firefox. You will see something similar to the output of tutorial1, all we can currently do is walk left and right along the screen, he can also jump, but it looks funny right now as he has no gravity! Lets go ahead and fix that.

If you look at the documentation for the player class (Link) you should be able to figure out which function has to be called to add gravity to the player. A hint is that the function contains the word Gravity!

Once you have found it you will want to invoke it on the *player* variable, you will notice that it takes one parameter of type int, this value allows you to set how strong the gravity is, I would recommend about 100.

HINT:

It should look something like:

```
player.setGravity(100);
```

Step 2

The next step is to add some platforms for Mario to walk and jump on! The ReusableObject allows you to create groups of platforms that all act in the same way. The beauty of this is that you can set the properties of multiple platforms in one go, for example you can set hit detection in one line for multiple platforms. You will need to create a variable to store your object and then you will need to create it in the *preload* function with a line similar to below:

platforms = new ReusableObject(game, 'platforms', "img/platform.png");

As you can see the Platform object takes three parameters:

1. The game object
2. The unique image key used internally
3. The image to represent them

If you reload the page you will see no differences, the reason for this is that you need to go ahead and create a platform with the *createWidthHeight* function. This function is invoked on your *platform* variable and takes four parameters:
1. The x position of the platform
2. The y position of the platform
3. The width of the platform
4. The height of the platform

Alternatively you can call the *create* function which takes an *x* and *y* value, the width and height is generated from the images width and height.

Each of theses parameters can be passed as an integer which is a representation in pixels or they can be passed through as a string which acts as a percentage. You will want to put this in the *create* function and should look similar to:

platforms.createWidthHeight("0", "100", "50", "5");

Feel free to add in a few more platforms in various locations.

Step 3

Now you will notice that Mario just goes through the platforms instead of sitting on top of them, the reason for this is that you need to tell your game to check for collisions between the player and the platform. Like before you only need one line of code to get this set up, the ReusableObject has a function called *checkCollision* and one called *checkSimpleCollision.*

The first function allows you to specify a function to be run on collision, this is useful, say if you are shooting bullets, but we shall discuss this in another tutorial**.** The *checkSimpleCollision* function allows us to have two objects hit each other and sit on top of each other. We shall use this function now to determine when Mario should stand on top of the platform.

It takes one parameter which is the object you want to check for collisions against, so in our case it will be our *player* variable.

You will need to put the code in the *update* function as the game will check the collisions on every update. So you should have a line similar to the below one at the top of your *update* function:

```
platforms.checkSimpleCollision(player);
```

Refresh your browser...he moves the platform! Now this may come in useful in some cases, (maybe a Doodle Jump type game?) To fix this all we need to do is tell our platforms not to move, the function *setAllImmovable* function can be used within the *create* function on the platforms variable.

Now if you refresh the browser Mario now stays on the platform!

Step 4

This step is a bonus extra step in which we shall quickly have Mario warp from one side to the next. Can you think how we would do this? A few hints for you:

- **You will need to set collisions with the world boundaries to false in the player class**
- **When the players x position is just over the width of the world put them to x coordinate 0**
- **When the players x position is less than 0 you need to put the player to x coordinate of the worlds width**