CONTENIDOS

1. Empezando con Phaser

Primeramente, resulta importante recordar que la librería se encuentra dentro de la carpeta *build*, la misma se llama *phaser.js*. Este archivo debe ser linkeado en el archivo *html*. Aquí un ejemplo:

```
<html>
    <head>

        <title>Ejemplo Phaser!</title>
        <script type="text/javascript" src="js/phaser.js"
></script>
        <script type="text/javascript" src="juego.js" ></script>

    </head>
    <body>


    </body>
</html>
```

Donde *juego.js* es el script de nuestro juego. Es importante incluirlo luego de la librería ya que dependemos de ella.

Phaser se maneja a partir de 4 instancias: *preload*, *create*, *update* y *render*. Estos estados son gestionados por su objeto principal *Game*. Este objeto es la clase principal del juego, es aquel que crea todos los actores y gestiona las escenas. De él, además, podemos obtener valores fundamentales, como el tiempo del juego, la dimensión del lienzo u operar herramientas como la clase *rnd* de manipulación aleatoria.

Aquí un ejemplo de la estructura de estados:

```
var game = new Phaser.Game(800, 600, Phaser.AUTO, '', {
 preload: preload,
 create: create,
 update : update,
 render: render });


    function preload () {

    }

    function create () {

    }

    function update () {

    }

    function render () {

    }
```

Donde parámetros del Game son:

- Ancho del lienzo en pixeles
- Alto del lienzo en pixeles
- Motor de rendirizado (WegGL o Context2d. Si no se especifica nada lo toma automáticamente)
- Padre de del DOM HTML (no importa saber esto, sólo mencionaremos que esta propiedad permite “colgar” el lienzo en una parte específica de la página, en caso de que hubiera más elementos en ella. Si se lo deja vacío, lo cuelga como primer hijo del BODY)

- Objeto que define las 4 funciones que se ejecutarán en los 4 estados.

Para más información de esta función y del objeto Game, se puede consultar la documentación:

http://docs.phaser.io/Phaser.Game.html

Allí se podrá ver que el objeto puede recibir más parámetros que permiten especificar mejor las características de nuestro juego.

1. Imágenes y sprite en Phaser

Para cargar una imagen, primero debemos cargar el archivo persistente, ello lo haremos en el estado de *preload*.

Para manejar archivos de imágenes en Phaser, primero debemos cargarlos en la caché. La función *load* de Phaser (Dependiente de Game) se encarga de hacer la precarga de los assets (archivos de imágenes, audio y demás cosas que utilizaremos en el juego). Su sintaxis muy sencilla, allí indicamos que imágenes vamos a cargar y que ID le colocaremos. El ID nos permitirá identificar las imágenes cuando pretendamos recuperarlas.

```
function preload() {
  // Imagenes a cargar
  game.load.image('nombre_id', 'URL_asset');
}
```

En el siguiente ejemplo cargaremos la imagen **cara_img.png** a la cual le asignaremos el ID **cara** (no hace falta que tenga el mismo nombre que la imagen).

```
function preload() {
  game.load.image('cara', '../../assets/sprites/cara_img.png');
}
```

Ya estamos en condiciones de utilizar la imagen en nuestro juego. Para colocarla en nuestra escena lo que debemos hacer es agregarla a la escena utilizando la función *add* del objeto Game del siguiente modo:

```
game.add.sprite(posX, posY, 'cara');
```

Donde se deben especificar las posiciones x e y de la imagen dentro del Canvas de renderizado, medidas en pixeles tomadas desde la esquina superior izquierda.

En el siguiente ejemplo colocaremos una imagen centrada en el canvas.

Para ello, dentro de la función *create* definimos la imagen:

```
  function create () {
    // parametros x,y,imagen
    var image = game.add.sprite(game.world.centerX, game.world.centerY,
'cara');
    // la anclamos al centro
    image.anchor.setTo(0.5, 0.5);
```

```
}
```

Los parámetros *game.world.centerX* y *game.world.centerY* ubican el centro exacto del lienzo, despreocupándonos de saber nosotros su dimension. Aprenderemos más del objeto Game a medida que lo usemos.

Se puede observar que además estamos modificando el punto de anclaje, que por defecto viene definido en la esquina superior izquierda de la imagen, llevándolo hacia el centro de la misma.

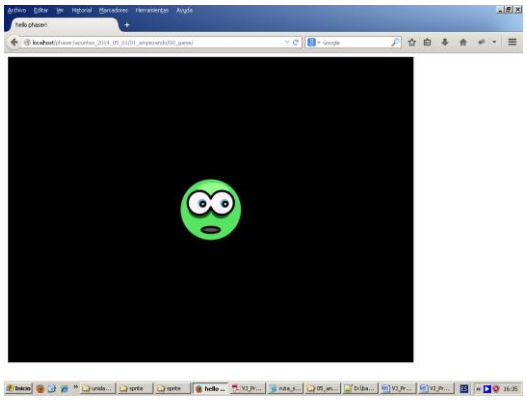

Figura 1: *Espacio de coordenadas de Canvas.*

Más adelante podremos realizar algunas acciones sobre la imagen como por ejemplo desplazarla, escalarla, rotarla o reemplazarla.

Si ahora deseamos brindarle movimiento a la imagen, tendremos que incorporar la función *update* debido a que allí se ejecutan instrucciones que necesitan ser leidas en cada ciclo.

```
var game = new Phaser.Game(800, 600, Phaser.AUTO, '', { preload: preload,
create: create , update:update });
```

Para hacer rebotar el personaje contra los bordes laterales simplemente le modificaremos la posición horizontal mediante *image.x*.

```
function update() {
  // hacemos rebotar el personaje contra los bordes laterales
  posX+=velX;
  image.x=posX;
  if (((posX+(anchoSprite/2)) > 800) ||
      ((posX-(anchoSprite/2)) < 0)){
    velX*= -1;
  }
}
```

2. Animar sprite

Para animar un sprite contenido en una spritesheet podemos hacerlo de dos maneras. Para una bastará con solo con la imagen que contiene todos los frames del sprite. Por ejemplo la siguiente imagen:

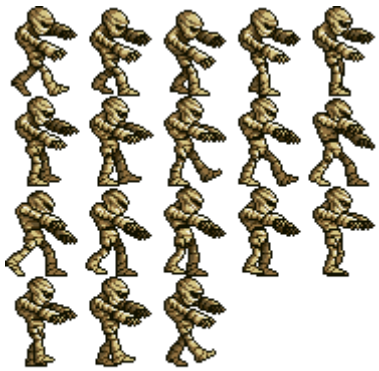

Figura 2: *Spritesheet de única animación*

Para tal caso bastará con decirle a Phaser que ejecute la única animación possible de la imagen.

Primero cargamos el spritesheet en el estado *preload*, notesé que ahora estamos agregando un spritesheet. Los parámetros que usaremos son los siguientes:

_ Id del sprite que usaremos

_ URL del spritesheet

_ Ancho del Frame

_ Alto del Frame

_ Cantidad de Frames que contiene el spritesheet

Si el spritesheet contiene todos los frames de una misma animación, este ultimo parámetro puede omitirse.

Para conocer más sobre la carga de sprite, se puede recurrir a la documentación:

http://docs.phaser.io/Phaser.Loader.html#spritesheet

```
game.load.spritesheet('mummy',
'../../assets/sprites/metalslug_mummy37x45.png', 37, 45, 18);
```

Ahora en el estado *create* vamos a crear el sprite a partir de la imagen que cargamos antes:

```
        //Creamos el sprite
        var mummy = game.add.sprite(200, 200, 'mummy');
        //Le cargamos la animacion
        mummy.animations.add('walk');
```

```
        //La hacemos correr

        mummy.animations.play('walk',20,true);
```

Primero creamos el sprite a partir de la imagen que cargamos antes y el id. Se ubicará en la posición 200 de *x* y 200 de *y*. Hasta aquí no hay diferencia a lo que habíamos visto. Ahora le agregaremos una animación, como el spritesheet solo tiene una no hay que definirle nada más que el nombre. A partir de la función *animation* de la clase *sprite*:

```
mummy.animations.add('walk');
```

La llamaremos “walk” pero podría haberse llamado de cualquier manera.
Ahora siemplemente tenemos que ejecutarla con la función *play* también de *animation:*

```
mummy.animations.play('walk',20,true);
```

El primer parámetro dice que animación queremos ejecutar, el segundo son los frame por segundo y el tercero es una bandera que dice si es de ejecución cíclica.

Para saber más sobre la carga de sprites pueden consultar la documentación:

http://docs.phaser.io/Phaser.AnimationManager.html

Ahora bien, es posible que nuestro spritesheet tenga más de una animación, cómo la siguiente imagen:

Figura 3: *Spritesheet de varias animaciones*

Veremos que la forma de definir un sprite con estas características no difiere mucho de lo que vimos.

Partiremos de cargar la imagen en el estado *preload*:

```
game.load.spritesheet('guybrush', '../../assets/sprites/guyb1.png',
169, 246);
```

Como mencionamos antes, si vamos a utilizar todos los frames que hay en la imagen, no hace falta especificar la cantidad que queremos usar, aquí solo definimos el ID, la URL, el ancho y alto del frame.

Ahora crearemos el sprite de la misma manera que lo hicimos antes en el estado *create*.

```
var GuyWalking = game.add.sprite(100, 100, 'guybrush');
```

Como siempre, especificamos una posición en *X*, una posición en *Y* y el ID del *sprite* que definimos antes.
Ahora lo que vamos a hacer es crearle las 3 animaciones, aunque que hace falta crear todas si no las vamos a usar:

```
GuyWalking.animations.add('frente',[0,1,2,3,4,5],10,true);

GuyWalking.animations.add('costado',[6,7,8,9,10,11],10,true);

GuyTalking.animations.add('hablar',[12,13,14,15,16,17],10,true);
```

El primer parámetro especifica el nombre de la animación: "*frente*", "*costado*" y "*hablar*", el Segundo parámetro es un arreglo que dice los indices de los frames que tiene esa animación en la imagen, luego los fps y la bandera que especifica si vamos a dejar corriendo la animación cíclicamente.

Ahora lo único que queda por hacer es decirle que animación ejecutar:

```
GuyTalking.animations.play('hablar');
```

Cómo ya especificamos los fps y la bandera, solo debemos decir que la ejecute. La definición de los fps puede hacerse en el momento de la creación o a la hora de correrlo, es indistinto. Todo dependerá de si queremos definir los fps inertes a la animación o lo cambiaremos cada vez que la ejecutemos, lo mismo con la bandera de corrida cíclica.