CONTENIDOS

1. Tiles

Tiles es una técnica utilizada en el desarrollo de videojuegos para realizar escenarios a partir de la repetición de patrones, es decir, imágenes.

Phaser permite trabajar con Tiles, para ello utiliza un mapa de extensión Json, que contiene la información de los Tiles, ubicación, patrones utilizados, etc. Como ya hemos visto, además del mapa, debemos cargar, también, las imágenes.

Utilizar Tiles es muy sencillo, la parte más compleja es crearlos, para ello vamos a utilizar el programa Tiled. En este apunte sólo veremos cómo utilizar los Tiles, pero es posible ver cómo crearlos en el documento:

Crear_Tiles_con_Tiled.pdf

Por último, antes de empezar, tendremos a mano la documentación de Tiles:

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

1.1 Tilesmap

Vamos a utilizar dos tipos de Tiles, uno de ellos es el Tilemap. El Tilemap es lo que armamos para crear el escenario.

Vamos a utilizar la siguiente imagen de patrones, en la misma se hallan 25 patrones de 32x32 píxeles:

Figura 1*: TileMap ground_1x1.png*

Primero cargamos el mapa JSON, que como mencionamos, es el archivo que nos da la información de cómo ubicar los Tiles. A continuación cargaremos la imagen que vamos a utilizar,

```
function preload() {

        //Cargamos el mapa

   game.load.tilemap('map', '../../assets/tilemaps/maps/mapa_P3_1.json', null,
 Phaser.Tilemap.TILED_JSON);

//Cargamos la imagen correspondiente al mapa

        game.load.image('ground_1x1', '../../assets/tilemaps/tiles/ground_1x1.png');
```

Como se puede ver, no es muy distinto a lo que habíamos visto. La sintaxis del método load.tilemap es:

tilemap(**key**, **mapDataURL**, **mapData**, **format**)

Donde:

- key: Es el nombre por el cual lo identificaremos
- mapDataURL: Es la dirección del archivo

- mapData: Si no especificamos una ruta, podemos usar un objeto como referencia para crear el TileMap
- format: Para un archivo JSON **Phaser.Tilemap.TILED_JSON** y para un archivo CVS **Phaser.Tilemap.CSV**. Ambos son soportados por Phaser.

Ahora tendremos que cargar e identificar 3 elementos:

El Mapa, que es el elemento lógico que contiene a todo el escenario, el Tile que es la imagen y la Capa (layer) que es donde se alojan los distintos elementos lógicos (ver el documento de cómo crear Tiles con Tiled).

En el estado Create:

```
        //Creo el mapa
        map = game.add.tilemap('map');

        //agrego la imagen del tile
        tile = map.addTilesetImage('ground_1x1');

        //asigno las capas piso y pared
        layer = map.createLayer('piso');
        layer1 = map.createLayer('pared');


        //cuales son los tiles que van a colisionar, el tile elegido tiene 12
elementos, van a colisionar los elementos del 1 al 3
        map.setCollisionBetween(1,3);
```

El mapa es que contenedor y gestor de todo el escenario, el Tiled sólo gestiona la imagen y puede haber sólo una por mapa. Las capas son aquellos grupos lógicos de Tiles, por ejemplo, la capa “piso” va a contener todo aquello usado para pisar, mientras que la capa “pared” contiene algunos elementos a parte que va a adornar el mapa. En el documento de cómo crear Tiles se muestra mejor esto.

Con esto ya tenemos nuestro mapa cargado, ahora sólo debemos darle colisión a los elementos. Suponiendo que tenemos el actor Patricio que veníamos usando para los demás ejemplos, entonces en el estado Update, definimos el colide para Patricio contra la capa del piso:

```
    function update() {

        //defino colision entre Patricio y la capa del piso

        game.physics.arcade.collide(Patricio, layer);
```

1.2 Tilesprite

Los Tilesprite son los sprite creados mediante Tiles, es decir, que son cargados mediante una imagen de patrones y un mapa.

La forma de cargarlos es similar a la que usamos antes, necesitamos cargar el mapa y la imagen a la cual hace referencia. En este caso vamos a utilizar la siguiente imagen para esparcir un par de monedas por el mapa.

Figura 2: *TileSprite coins.png*

Vamos a utilizar sólo uno de los Tiles, aunque podríamos definir la animación completa.

La forma de utilizar Tilesprite es mediante un grupo, agregándolo al mapa. La función que agrega un TileSprite al mapa es la función *createFromObjects* perteneciente al objeto TileMap y su sintaxis es la siguiente:

createFromObjects (**name, gid, key, frame, exists, autoCull, group,**)

Donde:

- name: Nombre con el cual identificaremos el TileSprite
- gid: Es el ID de grupo, este valor está dado por el mapa (Ver documento adjuno)
- key: El Key del spritesheet
- frame: **Opcional**. El frame que queramos utilizar en el spritesheet del coin.png. Por ej: La moneda de canto tiene el frame "3"
- exists: **Opcional**. Si existe desde el principio o se va a agregar luego
- autoCull: **Opcional**. Cortar si está fuera del rango de la cámara.
- group: **Opcional**. Grupo al cual se agregará el objeto.

De esta manera, vamos a implementar las monedas definidas en el mapa. Primero cargamos el mapa y la imagen (si no lo hicimos antes):

```
function preload() {


        //Cargamos el mapa

   game.load.tilemap('map', '../../assets/tilemaps/maps/mapa_P3_1.json', null,
 Phaser.Tilemap.TILED_JSON);

//Cargamos la imagen correspondiente al mapa, en este caso las monedas

        game.load.spritesheet ('moneda', '../../assets/sprites/coin.png',32,32);
```

Ahora crearemos el grupo y agregaremos cada una de las monedas al grupo.

```
function create() {

    //creamos el grupo
   monedas = game.add.group();


    //GID: coin, 26
    map.createFromObjects('coin', 26, 'moneda', 0, true, false, monedas);

        }
```

Por último, podríamos definir una acción para la colisión entre el grupo “monedas” y nuestro personaje:

```
    function update() {

        game.physics.arcade.collide(Patricio, monedas, juntaMoneda, null, this);
}
```

En el ejemplo **00_Ejemplo_integrado_tiles** se puede ver la implementación de estos contenidos.

Recuerde ver también el documento *Crear_Tiles_con_Tiled.pdf* donde muestra cómo utilizar Tiled y entender el funcionamiento de los mapas.

2. Audio en Phaser

En esta sección veremos cómo incorporar audio a los videojuegos que desarrollemos con Phaser.

Para ello vamos a utilizar las clases *Sound* y *SoundManager*, cuya documentación oficial se encuentra disponible en:

docs.phaser.io/ Phaser.Sound.html

docs.phaser.io/ Phaser.SoundManager.html

Para cargar un audio, primero debemos cargar el archivo persistente, ello lo haremos en el estado de preload.

Para manejar archivos de audio en Phaser, primero debemos cargarlos en la caché. La función *load* de Phaser (Dependiente de Game) se encarga de realizar la precarga de los assets (archivos de audio, imágenes y demás cosas que utilizaremos en el juego). Su sintaxis muy sencilla y similar a la usada para imágenes. Debemos indicar que audios vamos a cargar y que ID le colocaremos. El ID nos permitirá identificar los audios cuando pretendamos recuperarlos

```
function preload() {
  game.load.audio('efecto1',
'../../assets/audio/SoundEffects/meow1.mp3');
}
```

Es importante mencionar que Firefox solo soporta el formato de archivo *ogg*, por lo tanto, puede ser de utilidad pasarle el archivo en varios formatos, para que el loader elija el segundo formato en caso de no soportar el primero.

```
function preload() {
  game.load.audio('efecto0',
['../../assets/audio/oedipus_wizball_highscore.mp3',
'../../assets/audio/oedipus_wizball_highscore.ogg']);

}
```

Ya estamos en condiciones de utilizar el audio en nuestro videojuego. Para ello debemos usar la función *add* del objeto Game del siguiente modo:

```
// agregamos el audio al game
var music0 = game.add.audio('efecto0');
```

Para reproducir el sonido recién agregado es necesario usar el método ***play****(marker, position, volume, loop)* de la siguiente manera:

```
// reproducimos el audio en el game
music0.play('', 0, 0.75, true);
```

Donde nos interesa configurar principalmente los argumentos *volume* que se setea entre el mínimo 0 y máximo 1 (default 1) y *loop* que especifica si permite reproducir de forma continua el archivo de sonido (default false).

Para pausar y reanudar la reproducción de un audio existen los métodos: *pause()* y *resume()*. El siguiente ejemplo permite pausar y reanudar la reproducción al picar sobre un sprite:

```
imagen0.events.onInputDown.add(clicks,this);

function clicks(item){
  if(music0.isPlaying){ // isPlaying es true cuando el audio se
encuentra en modo play
      music0.pause(); // pausa el audio
  }else if(music0.paused){ // paused es true cuando el audio esta
en modo pause
      music0.resume(); // reanuda el audio
  }
}
```

Como se observa, también hace uso de las propiedades *isPlaying* y *paused*, que devuelven valores booleanos indicando si el audio se encuentra modo pausado o reproduciendose.

Para modificar el volumen seteado inicialmente con el método *play()* se puede utilizar la propiedad *volume*:

```
// setea el volumen para este audio entre 0 y 1
music0.volume += 0.1;
```

En el ejemplo **00_audio** se puede observar en detalle la incorporación de sonidos.