10/11/2021

Asignación de funciones en js:

Puedo guardar funciones en variables, devolver una función, pasar funciones como parámetro:

Si dada una función suma(a,b), la quiero referenciar (por ejemplo, para un callback), escribo solamente suma, sin paréntesis. Ej: *let f = suma*; o *let f = this.suma*;

Si pusiera los paréntesis, me devolvería undefined.

En caso de una función arrow (función que genera otras funciones), entonces sí debería poner los paréntesis. Ej: *suma (a,b) return (a,b) -> a+b*; (No es muy común).

ANIMACIONES: **TWEENS**

Hasta ahora las habíamos hecho por frames, dibujando todos los frames cada x tiempo.

Ahora usaremos animaciones en las que solo tenemos que definir la animación inicial y la final (tweens). Por ejemplo, mover algo de manera lineal desde un punto hasta otro en x tiempo (Esto se llama **Interpolación**). Más ejemplos: Algo que se agranda y se contrae, una moneda que rota, cambio de color…

En Phaser…

Creamos un tween sobre un target (o varios targets), y le establecemos distintos parámetros. Los más importantes son duración (entre uno y otro). ->Propiedades de para configurar los tweens.

```
// this es Scene
let image = this.add.image(100, 80, 'fish', 0);

// Animación para mover de lado a lado una imagen
let tween = this.tweens.add({
    targets: [ image ],
    x: 700,
    duration: 4000,
    ease: 'Sine.easeInOut',
    flipX: true,
    yoyo: true,
    repeat: -1,
    delay: 10
});
```

Hay varias funciones de interpolación en esta página web.

También podemos **suscribirnos a eventos** que generan los tweens:

```
let tween = this.tweens.add({
    targets: image,
    x: 500,
    ease: 'Power1',
    duration: 3000
});
tween.on('stop', listener);

function listener() {
    // hacemos algo cuando termina el tween
```

```
}
```

Se pueden **encadenar** varios tweens para crear un timeline.

```
// this es Scene
var timeline = this.tweens.createTimeline();
// Primer tween
timeline.add({
    targets: image,
    x: 600,
    ease: 'Power1',
    duration: 3000
});
// Segundo tween; se ejecuta al terminar el primero
timeline.add({
    targets: image,
    y: 500,
    ease: 'Power1',
    duration: 3000
});
...

timeline.play();
```

- Nota: no tienen por qué ser imágenes, pueden ser cualquier tipo de game object.

https://rexrainbow.github.io/phaser3-rex-notes/docs/site/tween/

Mucha info sobre tweens.

AUDIO:

Para videojuegos: en general, la música debe estar en formatos comprimidos, y los efectos sonoros, en no comprimidos (p.ej. WAV).

Usando Web Audio, podemos crear nuestra propia onda de sonido. Ejemplo de Guille.

Uso de audio en Phaser:

Se trabaja con los sonidos en forma de onda.

Para gestionar audio, sólo tenemos que preocuparnos de:

1- Tener los archivos

2- Cargarlos:
```
scene.load.audio("explosion",assets/audio/SoundEffects/explosion.mp3");
```

2.1- Lista de prioridad de carga: en el preload (Sólo carga uno)

```
      scene.load.audio("boden", [
"assets/audio/bodenstaendig_2000_in_rock_4bit.mp3",
"assets/audio/bodenstaendig_2000_in_rock_4bit.ogg",]);
```

Como añadir al juego con una configuración inicial:

```
const config = {
  mute: false,
  volume: 1,
  rate: 1,
  detune: 0,
  seek: 0,
  loop: false,
```

```
  delay: 0,
}; // config es opcional
this.explosion = scene.sound.add("explosion", config);
```

3- (asegurarnos de que están bien cargados)

4- Reproducirlos – de fondo, como eventos, en loop…

```
explosion.play();
```

5- Enterarnos de cuándo han acabado de reproducirse (si queremos)

```
music.stop();
```

Para que se **repita**:

```
explosion.setLoop(true);
```

Si queremos **destruirlo**:

```
explosion.destroy();
scene.sound.remove("explosion");
```

Eventos de Sonido:

Evento cuando el sonido termina:

```
function create() {
  explosion.once("stop", (music) => {
    //…
  }); }
```

Otros conceptos: Sprites de audio, Recursos de audio.

17/11/2021

Grupos en Phaser: Array que puede guardar objetos y hacer cosas sobre ellos. No hay que suponer que los objetos sean del mismo tipo. Por ej. Sirven para comprobar colisiones, buscar algo en los objetos. No tienen física, pero se puede añadir físicas a los objetos que son parte del grupo. Hay distintos tipos de grupos. (entidades físicas/dinámicas).

Mapas en phaser con Tiled

Un **tile** (baldosa) es una imagen, generalmente cuadrada y de tamaño fijo, usada para dibujar un elemento en un videojuego. Simplifican el juego ya que usandolos se considera el mundo una matriz discreta.

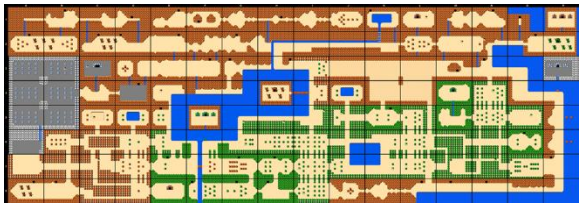
Pueden ser usados para dibujar pero también para la lógica de juego (roguelikes).

Cada tile es una imagen independiente, particular, o no.

Un **tile map** una simplificación de un mundo en 2D en la que dicho mundo es dividido en una cuadrícula de tamaño fijo (una matriz 2D). en cada celda se pinta un tile.

Partes de un mapa de tiles:

- Un fichero de datos con la definición del tile map.
  - su tamaño (ancho y alto)
  - el tamaño de las casillas (tiles)
  - qué tile o imagen va en cada casilla
  - Una hoja de sprites con todos los tiles disponibles juntos.



Tile Sets: Un tile set se suele implementar como un archivo de imagen en el que cada celda representa un tile. Ejemplo:

Cada celda del tile set representa una imagen independiente. Esto funciona bien para cosas aisladas: monedas, árboles, cajas…

Para evitar los "cortes", los tiles normalmente están creados para repetirse en la escena sin que se vean costuras, de forma que se pueden construir escenas muy complejas.

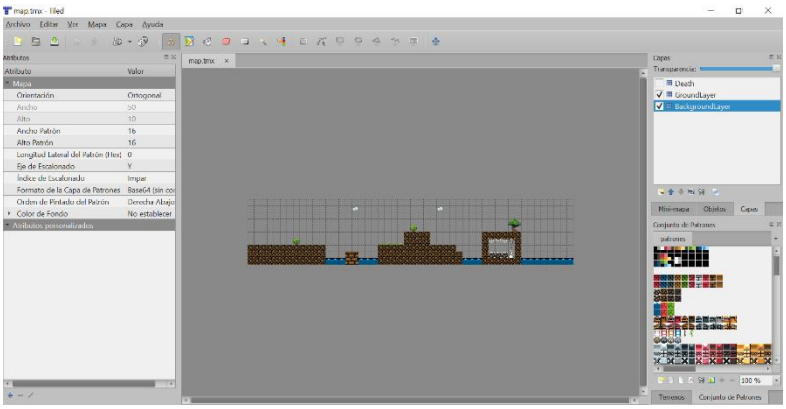
Se ocupa mucha menos memoria que dibujando la escena en una imagen (se carga una, se usa muchas veces). Es más eficiente porque ocupa menos.

Ventajas: Al tener todos los tiles en la misma hoja de sprites sólo necesitamos una llamada a pintado para pintar todo el escenario; cuando cargamos desde Internet, es fundamental reducir los tiempos de carga; dibujar la escena es más sencillo; sobre todo con un editor de tiles; podemos reutilizar tiles en diferentes escenarios.

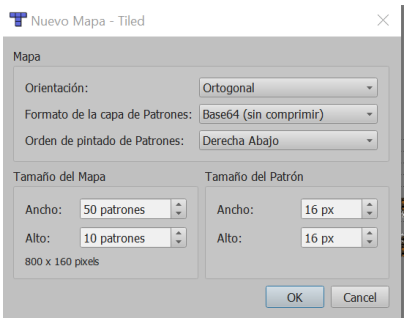
La desventaja es que los gráficos del escenario están repetidos.

Editores de tiles: Simplifican la edición del tilemap.

TILED editor de niveles 2D, puede exportar diferentes formatos como JSON. Soporta capas (layers), una capa especial de **objetos**, con la ubicación de los objetos colocados en el mapa.



PARTES DE LA INTERFAZ:
Menu contextual (izquierda)
Tilesets (derecha abajo)
Layers (derecha arriba)
Herramientas (parte superior)
Zona de edición (centro)

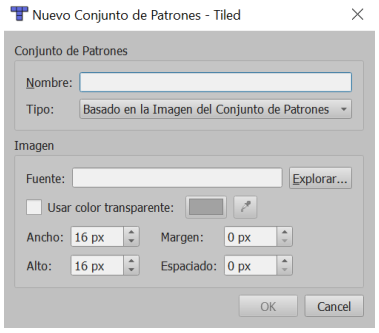


Al crear un nuevo proyecto establecemos:

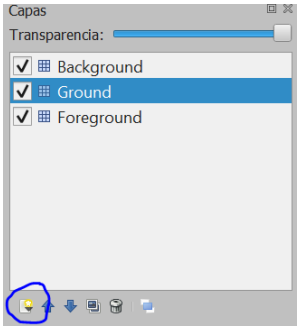
- El tamaño del patrón (tile) y el número de patrones que tendrá el nivel.
- Si queremos vista ortogonal o isométrica.
- El formato de la capa de patrones para Phaser ha de ser Base64 (uncompressed) sin comprimir.
- **No** creamos un mapa infinito.

Proyección otrogonal o isométrica. Tiled soporta ambas.
Puedo usar varios tilesets, no sólo uno.

Usar tilesets:



Añadir un tileset <-
¡! El nombre que le demos al tile set se usará posteriormente para cargar el spritesheet con los tiles en Phaser.
Añadir una **capa** -> . Son importantes para diferenciar los objetos del fondo con los del suelo. Pasa lo mismo con los nombres.



Exportar:
Los mapas se guardan con .tmx.
Para usar el mapa hay que exportar a .json.
Añadir siempre estas opciones a la configuración -> para que la exportación no de problemas con Phaser.