Práctica 2 – Ciclo de vida de una Escena

Objetivo: En *Phaser* las escenas son la base constructora del juego ya que aquí se van colocando todos los objetos de un juego ó *game objects*. Por lo tanto, el propósito es dejar preparada la estructura para una escena que además tiene un ciclo de vida básico.

1. **FORMA 1:** Comprobar el ciclo de vida básico de una única escena manipulada por funciones de JavaScript.
    - Desde el Explorador de Windows, crea una copia de la carpeta `p01/`, y nómbrala `p02_sinclases/`.
    - Carga el proyecto en `p02_sinclases/` con *VSCode*.
    - Editar el archivo `main.js` y modificar el objeto `scene` de la siguiente manera:

```
   scene: {          // Configuración de la escena
        init,        // Para inicializar y cargar datos
        preload,     // Para pre-cargar archivos
        create,      // Para cargar los objetos
        update,      // Bucle del juego
    },
```

    - Posteriormente, al final del archivo (fuera del JSON), crear las todas las funciones DUMMY:

```
function init() {
    console.log("Soy init");
}
function preload() {
    console.log("Soy preload");
}
function create() {
    console.log("Soy create");
}
function update(time, delta) {
    // ESTA FUNCION CREA UN CICLO INFINITO
    //      time:   ¿?
    //      delta:  ¿?
}
```

    - A continuación lanza el Live Server y visualiza los mensajes de Consola.
    - En seguida, usa `console.log` para mostrar `time`, y después `delta`. ¿A qué crees que corresponde cada parámetro? Anota y compara tu respuesta.

2. **FORMA 2:** Dado que el objetivo de Phaser3 es "modularizar" todo, se tiene que crear un archivo .js que contendrá la escena a la que se hará referencia:
    - Desde el Explorador de Windows, crea una copia de la carpeta `p01/`, y nómbrala `p02_conclases/`.
    - Carga el proyecto en `p02_conclases/` con *VSCode*.
    - Desde el *Explorador* de *VSCode*, crea la carpeta `scenes/` dentro de carpeta `p02_conclases/src/`.
    - A continuación crea el archivo `Bootloader.js` dentro de `scenes/`
    - Modifica el objeto `scene` y vuélvelo un arreglo, es decir:

```
    scene: [Bootloader],    //Aquí irá la lista de scenas del juego
```

    - Para poder trabajar con módulos es necesario poder exportar e importar archivos, por lo tanto edita el archivo `index.html` y agrega el atributo `type="module"` a la referencia del archivo `main.js`

```
    <script src="./src/main.js" type="module"></script>
```

    - Ahora es necesario crear la clase *Bootloader,* y futura escena, dentro del archivo `Bootloader.js`. Su propósito es operar las diferentes partes de Phaser, así como para cargar archivos.

```
class Bootloader extends Phaser.Scene{
    constructor(){
        super({
            key: "Bootloader" //Nombre interno o clave de referencia
        });
    }
}

export default Bootloader;
```

    - Importa la clase *Bootloader* al inicio del archivo `main.js`: y con esto debe desaparecer el error previo:

```
import Bootloader from "./scenes/Bootloader.js"
```

    - Finalmente crea el ciclo de vida de la escena, definiendo los métodos `init`, `preload`, `create` y `update`, dentro de la clase:

```
    init() {
        console.log("Soy init");
    }
    preload() {
```

```
        console.log("Soy preload");
    }
    create() {
        console.log("Soy create");
    }
    update(time, delta) {
        // ESTA FUNCION CREA UN CICLO INFINITO
    }
```

3. Después de las escenas, los objetos más importantes son las imágenes, así que para comenzar a manipularlas realiza lo siguiente:
    - Vacía los métodos `preload` y `create`, sólo `init` debe quedarse con su mensaje.
    - Crea una carpeta llamada `assets/` dentro de la `p02_conclases/`
    - Descarga las imágenes adjuntas (**yoshi.png** y **yoshi_fondo.png**). Estas imágenes deben poderse visualizar desde *VSCode*.
    - Utiliza el método `preload` para cargar las imágenes, y el método `create` para crear los objetos necesarios para poder manipular las imágenes:

```
    preload() {
        this.load.path = "./assets/";          //Ruta de todas las imagenes
        this.load.image("yoshi", "yoshi.png"); //alias y archivo
        this.load.image("yoshi_fondo"); //Sin nombre de archivo, se toma por
                                        //defecto el nombre del alias
    }
    create() {
        this.yoshi = this.add.image(130, 130, "yoshi_fondo"); //atributo
        this.yoshif = this.add.image(100, 100, "yoshi");      //atributo
    }
```

Nótese que una forma simplificada para cargar imágenes es la siguiente:

```
    preload() {
        this.load.path = "./assets/";
        this.load.image(["yoshi_fondo", "yoshi"]);    //Arreglo de imágenes
    }
```

- Revisa la guía adjunta sobre "*Origen de una imagen*" y a continuación configura los orígenes de las imágenes de la siguiente manera y visualiza qué pasa:
    - **yoshi**: origen horizontalmente al *centro* y verticalmente *arriba*
    - **yoshif**: origen en la esquina *inferior izquierda*
- Alguna de las propiedades más importantes que pueden aplicarse a los objetos de imagen previamente creados son los siguientes.

```
flipX = <booleano>;              //Voltear imagen en horizontalmente.
flopY = <booleano>;              //Voltear imagen en verticalmente.
setVisible(<booleano>);          //Mostrar u ocultar la imagen.
setScale(<prop. x>, <prop. y>);      //Escalar la imagen.
setAlpha(<proporción>);          //Transparencia y opacidad [0,1]
setTint(<0xhexadecimal>);        //Entintar de un color la imagen.

x = <numero>;                    //Posición en X en el canvas
y = <numero>;                    //Posición en Y en el canvas
angle = <grados>;                //Giro en el eje Z
rotation = <radianes>;           //Giro en el eje Z
setDepth(<número de capa>);      //Número de capa (la primera es 0)
```

- Prueba cada una de las propiedades anteriores sobre **yoshi**. Considera que el orden de creación de cada imagen afecta directamente el número de capa a la que pertenece, por lo tanto cuando pruebes el método `setDepth` tal vez necesites superponer ambas imágenes y aplicarlo sobre **`yoshi`** y **`yoshif`**.

- IMPORTANTE: con la intensión de que más adelante puedas obtener las propiedades de cualquier objeto de *Phaser* que crees, a continuación muestra en Consola el contenido del objeto de **yoshi** con `console.log(this.yoshi)` para visualizar la lista completa de propiedades.

- Como prueba final de esta sección, utiliza el método `update` para crear un procedimiento que mueva al personaje de *derecha-a-izquiera* y de *izquierda-a-derecha* de forma continua mientras el programa se esté ejecutando en el navegador. El personaje debe voltear al regresar de un lado a otro.

4. ¡Reto¡

Ahora sí, estás listo para crear tu primer escenario utilizando los conceptos previos. Por lo tanto, tu escenario debe tener lo siguiente:

- Un fondo.
- Al menos dos personajes.
- Objetos complementarios (rocas, arboles, etc.).
- Transformación del personaje o de algún otro elemento (posición, giro, escalamiento).
- Cambio visible de tinte.
- Opcionalmente transparencias.

[+Creatividad = 1 Comodín]