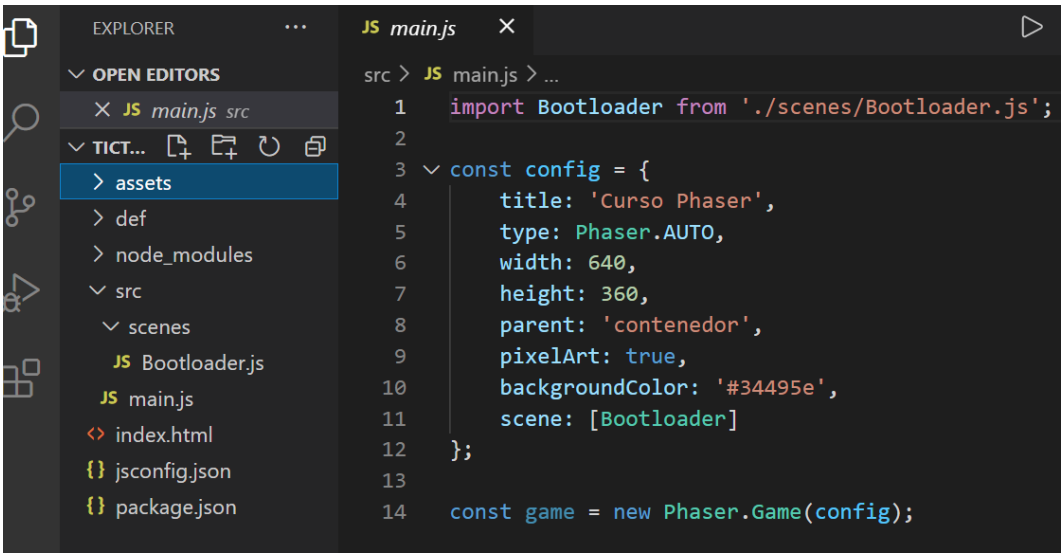
Juego de mesa

Para elaborar nuestro destinado a jugar con el móvil, vamos a introducir en la configuración un escalador de pantalla. Es un recurso creado por Phaser no hace mucho tiempo. Abrimos el archivo “Main.js”:



Introducimos el parámetro “scale” con un json (no olvidar tras cerrar las llaves, la coma):

```
const config = {
    title: 'Curso Phaser',
    scale: {

    },
    type: Phaser.AUTO,
    width: 640,
```

Copiamos el “parent”, “width” y “height” y añadimos:

```
scale: {
    mode: Phaser.Scale.FIT,
    autoCenter: Phaser.Scale.CENTER_BOTH,
    parent: 'contenedor',
    width: 640,
    height: 360,
},
```

Con esto la pantalla del juego se escalará y se centrará tanto horizontal como verticalmente.

Pero esta configuración lleva aparejada en el canvas que se creen unos márgenes que no nos interesan. Para evitar esto, abro el archivo “index.html” y configuro los estilos:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Juego de Mesa</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        body{
            background-color: black;
        }
    </style>
</head>
```

```
scale: {
    mode: Phaser.Scale.FIT,
    autoCenter: Phaser.Scale.CENTER_BOTH,
    parent: 'contenedor',
    width: 256,
    height: 320,
},
```

Volvemos a la configuración del juego, en el archivo "Main.js" y cambiamos el tamaño del canvas para nuestro juego:

Este tamaño es el del ancho de la imagen del tablero (256x256px) y a la altura le he sumado 64 px. Cambio el color de fondo del canvas:

```
const config = {
    title: 'Curso Phaser',
    scale: {
        mode: Phaser.Scale.FIT,
        autoCenter: Phaser.Scale.CENTER_BOTH,
        parent: 'contenedor',
        width: 180,
        height: 320,
    },
    type: Phaser.AUTO,
    pixelArt: true,
    backgroundColor: '#372538',
    scene: [Bootloader]
};
```

Puedes cambiar este color de fondo modificando el valor del parámetro "backgroundColor" de Phaser.

Continuamos con el archivo "Bootloader.js" y "seteamos" el "Path" para precargar los "assets" (imágenes):

Agregamos las imágenes a través de un array:

Bootloader.js

src > scenes > JS Bootloader.js > Bootloader > preload

```
class Bootloader extends Phaser.Scene {
    constructor() {
        super('Bootloader');
    }
    init() {
        console.log('Scene Bootloader');
    }
    preload() {
        this.load.path = './assets/';

    }
    create() {

    }
}

export default Bootloader;
```

```
preload() {
    this.load.path = './assets/';
    this.load.image([
        'tablero',
        'tablero_win',
        'reload',
        'position',
        'player1',
        'player2',
        'score'
    ]);
```

Cargamos la imagen del dado:

```
//Carga de dado
this.load.image('dado', 'dado/dadosinjugar.png');
this.load.image('dado1', 'dado/dado1.png');
this.load.image('dado2', 'dado/dado2.png');
this.load.image('dado3', 'dado/dado3.png');
this.load.image('dado4', 'dado/dado4.png');
this.load.image('dado5', 'dado/dado5.png');
this.load.image('dado6', 'dado/dado6.png');
```

Se agregó también la imagen del tipo de letra que se va a usar 'font'. Por último, agrego el "json" de la fuente.

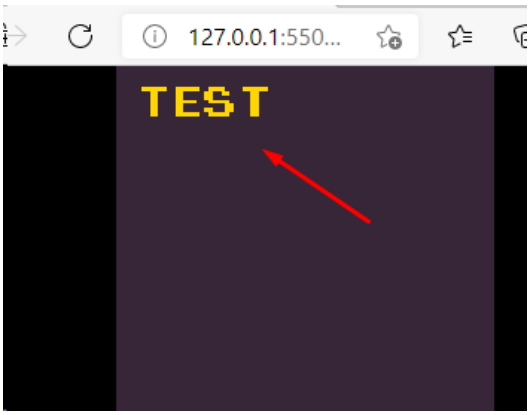
Borramos el método "create" de este archivo porque no lo vamos a necesitar y ejecutamos el evento "complete" al completar la carga del archivo para cargar la fuente:

```
this.load.on('complete', () => {
    const fontConfig = this.cache.json.get('fontConfig');
    this.cache.bitmapFont.add('pixelFont', Phaser.GameObjects.RetroFont.Parse(this, fontConfig));

});
```

Podemos hacer una prueba agregando un texto en la propia "array function":

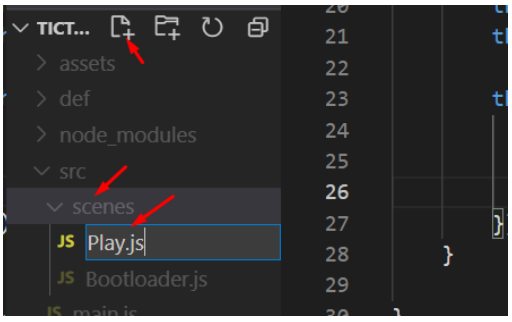
```
this.add.bitmapText(10, 10, 'pixelFont', 'TEST');
```



Una vez se haya comprobado que se carga la fuente con el texto de prueba, lo borramos o comentamos para que no se dibuje.

Cargamos la escena "Play" que crearemos a continuación:

```
this.load.on('complete', () => {
    const fontConfig = this.cache.j
    this.cache.bitmapFont.add('pixe
    this.scene.start('Play');
});
```



←Ahora crearemos un archivo llamado "Play.js" dentro de la carpeta "scenes":

Y copiaremos el método "constructor" del archivo "Bootloader.js". Cambiamos los datos por los de la clase que llamaremos "Play":

```
scenes > JS Play.js > Play
class Play extends Phaser.Scene {
    constructor() {
        super('Play');
    }
}

export default Play;
```

Exportamos la escena con *export default Play;*

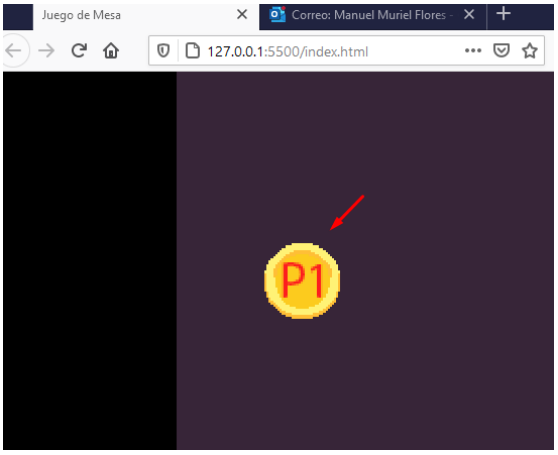
Añadimos el método "create" y la primera imagen del jugador1:

```
class Play extends Phaser.Scene {
    constructor() {
        super('Play');
    }
    create(){
        this.add.image(60, 100, 'player1');
    }
}
```

No se muestra la imagen porque hay que importar la escena en el archivo “Main.js”

```
main.js > config > scene
import Bootloader from './scenes/Bootloader.js';
import Play from './scenes/Play.js';
```

```
backgroundColor: '#372538',
scene: [
    Bootloader,
    Play
]
```



Se carga la imagen del “jugador1”. Podemos probar con el resto de imágenes para ver si se ven al cargarlas y asegurarnos que funciona correctamente.

Cargamos la imagen del tablero y creamos un contenedor con los marcadores de los jugadores 1 y 2. Creamos la imagen de fondo (marcador y marcador2), el texto para el contador (textoMarcador y textoMarcador2) y la imagen de los “player”:

```
create(){
        this.add.image(0, 64, 'tablero')
            .setOrigin(0);
    //Añadimos texto con el turno
    this.turnoText = this.add.bitmapText((this.sys.game.config.width /2) - 20, 50, 'pixelFont', 'TURNO DE:' + db.jugadorActual)
        .setOrigin(.5)
        .setScale(.5);
    //Contenedor
    this.marcador = this.add.image(0, 0, 'score')
        .setScale(.3)
        .setOrigin(0);
    this.textoMarcador = this.add.bitmapText(130, 15, 'pixelFont', '0');
    this.player = this.add.image(25, 23, 'player1')
        .setScale(.8);
    this.marcador2 = this.add.image(190, 0, 'score')
    .setScale(.3)
    .setOrigin(0);
    this.textoMarcador2 = this.add.bitmapText(320, 15, 'pixelFont', '0');
    this.player2 = this.add.image(215, 23, 'player2')
        .setScale(.8);
    const container = this.add.container(10, -200)
        .setScale(.5);
    container.add([
        this.marcador,
        this.textoMarcador,
        this.player,
        this.marcador2,
        this.textoMarcador2,
        this.player2,
    ]);
    this.add.tween({
        targets: [container],
        y: 15,
        ease: 'Bounce'
    });
```

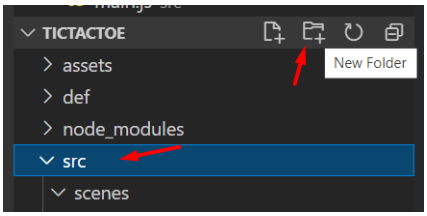
Se añade el dado en el “create” de la escena “Play”:

```
//Añadimos el dado
this.dado = this.add.image(225, 30, 'dado').setScale(.15);
```

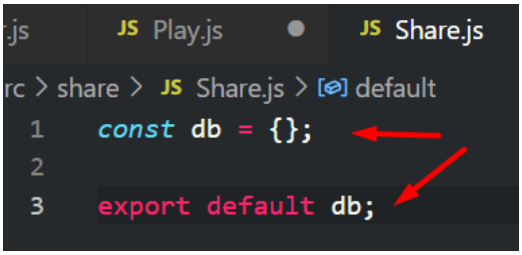
Y los jugadores 1 y 2:

```
//Añadimos los jugadores
this.jugador1 = this.add.image(-8, 310, 'player1');
this.jugador2 = this.add.image(-8, 310, 'player2');
```

Voy a crear dentro de la carpeta “src” una carpeta llamada “share” para compartir y dentro de ella un archivo “Share.js”:



Y dentro del archivo crearemos una base de datos para compartir llamada “db” y la exportamos:



Creamos una variable para establecer los turnos del jugador 1 y jugador 2.

Y por último, estableceremos los puntos:

Tanto para el jugador 1 como para el jugador 2 comenzarán con 0 puntos.

En la escena “Play” introduciremos la variable “turno” en el método “init” que crearemos antes del “create” y que tendrá un valor de “true” para el jugador 1 y de “”false” para el jugador 2, otra para saber si comenzó el juego y dos para ver si se desplazan las fichas hacia la derecha o a la izquierda:

```
init(){
    this.turno = true //Para el jugador 1, false para el jugador 2
    this.comienzo = false //Para saber si comenzó el juego
    this.direccionDcha1 = true; //Cuando el jugador1 se desplaza a la derecha
    this.direccionDcha2 = true; //Cuando el jugador2 se desplaza a la derecha
    this.XJugador1 = -8;
    this.YJugador1 = 312;
    this.XJugador2 = -8;
    this.YJugador2 = 312;

}
```

Vamos a agregar un texto con el turno del jugador que le toca jugar:

```
//Añadimos texto con el turno
this.turnoText = this.add.bitmapText((this.sys.game.config.width /2) -20, 50, 'pixelFont', 'TURNO DE:' + ((this.turno) ? 'JUGADOR1' : 'JUGADOR2'));
    .setOrigin(.5)
    .setScale(.5);
```

Con el operador ternario “?” si “this.jugador” es “true” escribirá “JUGADOR1” y si es “false” escribirá “JUGADOR2”.

Jugada:

En el “create” de la clase “Play”, creamos el código para la tecla “P” usando los “keycodes” de Phaser:

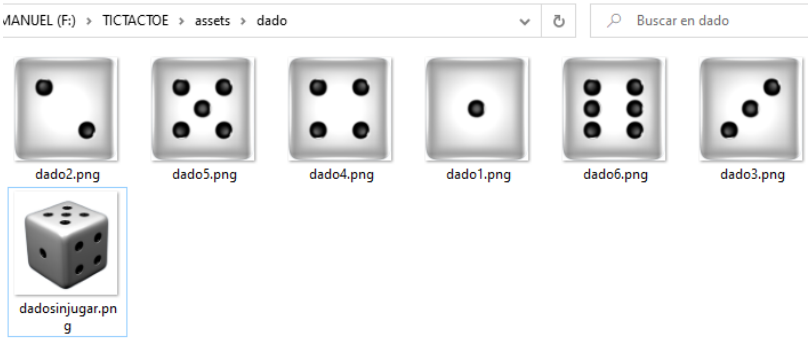
```
//Creamos la tecla de jugada "P"
const keyCodes = Phaser.Input.Keyboard.KeyCodes;
this.teclaP = this.input.keyboard.addKey(keyCodes.P);

this.teclaP.on('down', ()=>{
    console.log('Se ha pulsado la tecla P');
});
```

Y añadimos en esta clase el método “on”para este tecla, que ejecutará una función anónima que en principio la probamos con un “console.log(‘Se ha pulsado la tecla P’);” para probarla. Y ahora la sustituimos por la función que hace que gire el dado rápidamente durante un tiempo y se elija un número aleatorio que marcará la tirada del dado.

Se le aplica un “tween” para que se reproduzca el efecto de que cae rebotando desde la parte superior de la pantalla (inicialmente con y= -100) Hemos reducido la escala del “container” a 0.7 para dejar espacio para el dado.

Para realizar la jugada, vamos a realizar un “tween” en el que usaremos las siguientes imágenes que vamos a almacenar en la carpeta “dado”:



Observa que para la jugada los nombres de las imágenes son “dado1.png” para la tirada que valdría 1, y así con el resto de las seis caras.

```
this.teclaP.on('down', ()=>{
    this.dado = this.add.image(225, 30, 'dado').setScale(.15);
    this.interpolacion = this.add.tween({
        targets: [this.dado],
        rotation: 270,
        repeat: 2,
        duration: 340,
        onStart: () =>{
            this.sound.play('tirada');
        },
```

Con la función “onStart” iniciamos el sonido que tendremos que cargar en el archivo “Bootloader”:

```
this.load.audio('tirada', 'lanzarDado.mp3');
```

Avance del jugador:

En la función “onComplete” del “tween” añadimos:

```
onComplete: () =>{
    this.numero = Math.round(Math.random() * (6 - 1) + 1);
    this.dado.setVisible(0);
    this.jugada = this.add.image(225, 30, 'dado' + this.numero)
        .setScale(.075);

        if(this.turno){
            //Si se mueve a la derecha el jugador1 y no llega al borde derecho
            if (this.XJugador1 + this.numero * 16 < 256 && this.direccionDcha1 ){
                this.XJugador1 = this.XJugador1 + this.numero * 16;
            //Si el jugador1 sobrepasa el borde derecho
            }else  if (this.XJugador1 + this.numero * 16 >256 && this.direccionDcha1){
                this.XJugador1 = 256 - ((this.XJugador1 + this.numero * 16)-256);
                this.YJugador1 = this.YJugador1 - 16;
                //Ahora se moverá hacia la izquierda
                this.direccionDcha1 = false;
            }
            //Si el jugador1 se mueve a la izquierda y no llega al borde izquierdo
            else if(this.XJugador1 - this.numero * 16 > 0 && !this.direccionDcha1){
                this.XJugador1 = this.XJugador1 - this.numero * 16;
            }
            //Si el Jugador1 se mueve a la izquierda y sobrepasa el borde izquierdo
            else  if (this.XJugador1 - this.numero * 16 < 0 && !this.direccionDcha1) {
                this.XJugador1 = - (this.XJugador1 - this.numero * 16);
                this.YJugador1 = this.YJugador1 - 16;
                //Ahora se moverá hacia la izquierda
                this.direccionDcha1 = true;
            }

            this.jugador1.x = this.XJugador1;
            this.jugador1.y = this.YJugador1;

        }else if (!this.turno){

        }
```


Esto hará que el jugador 1 se desplace hacia izquierda mientras esté en la fila inferior y no sobrepase la casilla 16 (240 px). En el momento que la sobrepasa se le resta la diferencia y se sube la coordenada “y” una fila, cambiando el valor “direccionDcha” a false para que a partir de ahora se mueva hacia la izquierda. De forma parecida se programa cuando la coordenada x es menor que 0, se le suma la diferencia y se sube a la siguiente fila. Ahora, programaremos algo similar con el jugador 2 y tendremos que establecer el cambio de turno. También, debemos aplicar los desplazamientos cuando se caiga en una casilla de escaleras o de cabezas de serpiente. Por último, se programará la suma de puntos y la victoria en el caso de que cualquiera de los jugadores alcance la casilla 257.

```
}else if (!this.turno){
    //Si se mueve a la derecha el jugador1 y no llega al borde derecho
    if (this.XJugador2 + this.numero * 16 < 256 && this.direccionDcha2 ){
        this.XJugador2 = this.XJugador2 + this.numero * 16;
    //Si el jugador1 sobrepasa el borde derecho
    }else  if (this.XJugador2 + this.numero * 16 >256 && this.direccionDcha2){
        this.XJugador2 = 256 - ((this.XJugador2 + this.numero * 16)-256);
        this.YJugador2 = this.YJugador2 - 16;
        //Ahora se moverá hacia la izquierda
        this.direccionDcha2 = false;
    }
    //Si el jugador1 se mueve a la izquierda y no llega al borde izquierdo
    else if(this.XJugador2 - this.numero * 16 > 0 && !this.direccionDcha2){
        this.XJugador2 = this.XJugador2 - this.numero * 16;
    }
    //Si el Jugador1 se mueve a la izquierda y sobrepasa el borde izquierdo
    else  if (this.XJugador2 - this.numero * 16 < 0 && !this.direccionDcha2) {
        this.XJugador2 = - (this.XJugador2 - this.numero * 16);
        this.YJugador2 = this.YJugador2 - 16;
        //Ahora se moverá hacia la izquierda
        this.direccionDcha2 = true;
    }

    this.jugador2.x = this.XJugador2;
    this.jugador2.y = this.YJugador2;

    this.turno = true;

}
```

En este caso empleamos las coordenadas XJugador2 e YJugador2 y direccionDcha2 como parámetros para controlar el movimiento de la ficha 2. Pero las fichas aparecen directamente en la casilla y no es un efecto muy realista. Podemos hacer que se deslicen aplicando un "**timeline**" o línea de tiempo. Sustituimos las líneas:

this.jugador1.x = this.XJugador1;

this.jugador1.y = this.YJugador1;

por la línea de tiempo. Pero, para que se desplace de forma correcta al llegar a los bordes derecho e izquierdo debemos cambiar el movimiento de la línea de tiempo:

```
if(this.turno){ ← Cuando el turno es del Jugador 1
    //Si se mueve a la derecha el jugador1 y no llega al borde derecho
    if (this.XJugador1 + this.numero * 16 < 256 && this.direccionDcha1 ){
        this.XJugador1 = this.XJugador1 + this.numero * 16;
        this.lineDeTiempo = this.tweens.timeline({
            targets: [this.jugador1],
            tweens: [
                {
                    x: this.XJugador1,
                }
            ]
        });
```

```
//Si el jugador1 sobrepasa el borde derecho
}else  if (this.XJugador1 + this.numero * 16 >256 && this.direccionDcha1){
    this.XJugador1 = 256 - ((this.XJugador1 + this.numero * 16)-256);
    this.YJugador1 = this.YJugador1 - 16;
    //Ahora se moverá hacia la izquierda
    this.direccionDcha1 = false;
    this.lineDeTiempo = this.tweens.timeline({
        targets: [this.jugador1],
        tweens: [
            {
                x: 248,
            },
            {
                y: this.YJugador1,
            },
            {
                x: this.XJugador1,
            }
        ]
    });
}
```

```
//Si el jugador1 se mueve a la izquierda y no llega al borde izquierdo
else if(this.XJugador1 - this.numero * 16 > 0 && !this.direccionDcha1){
    this.XJugador1 = this.XJugador1 - this.numero * 16;
    this.lineDeTiempo = this.tweens.timeline({
        targets: [this.jugador1],
        tweens: [
            {
                x: this.XJugador1,
            }
        ]
    });
}
```

```
//Si el Jugador1 se mueve a la izquierda y sobrepasa el borde izquierdo
else  if (this.XJugador1 - this.numero * 16 < 0 && !this.direccionDcha1) {
    this.XJugador1 = - (this.XJugador1 - this.numero * 16);
    this.YJugador1 = this.YJugador1 - 16;
    //Ahora se moverá hacia la izquierda
    this.direccionDcha1 = true;
    this.lineDeTiempo = this.tweens.timeline({
        targets: [this.jugador1],
        tweens: [
            {
                x: 8,
            },
            {
                y: this.YJugador1,
            },
            {
                x: this.XJugador1,
            }
        ]
    });
}

this.cambiarTurno();
this.turnoText.setText('TURNO DE: '+ db.jugadorActual);
```

Al finalizar cualquiera de estas posibilidades se cambia el turno llamando a la función "cambiarTurno()" y se cambia el texto del turno.

Para que se pueda seguir la casilla en puntos introducimos en cada jugada:

```
this.cambiarTurno();
this.turnoText.setText('TURNO DE: ' + db.jugadorActual);
//Aumento los puntos del jugador 1
db.puntos.jugador1 += this.numero;
this.textoMarcador.setText(db.puntos.jugador1);
```

Y lo mismo para el Jugador 2.

Para que al comenzar el juego elija a un jugador de forma aleatoria, sustituimos el valor de "this.turno" por (recuerda está en el método "init" de la clase Play):

```
this.turno = Phaser.Math.Between(0, 1);
```

Y debemos cambiar el texto que aparece debajo de los marcadores. Usamos la función "cambiarTurno" que creamos nosotros:

```
this.cambiarTurno();
```

```
cambiarTurno(){
    this.turno = !this.turno;
    db.jugadorActual = (this.turno) ? 'Jugador 1' : 'Jugador 2';
}
```

Estamos empleando la variable "jugadorActual" que se encuentra en el archivo "Share.js" de la carpeta del mismo nombre que tengo que importar.

```
import db from "../share/Share.js";

class Play extends Phaser.Scene {
```

Ahora, vamos a detectar si cae en las casillas de las serpientes y escaleras. Una de las maneras de la lógica sería detectando las coordenadas de estas casillas. No es la más adecuada, pero si es quizás la más sencilla con los conocimientos que tenemos hasta ahora. Las coordenadas de las casillas serían:

Escaleras:

Casilla 7: x= 104 y= 256 ➔ x= 104 y= 296

Casilla 53: x= 192 y=192 ➔ x=176 y=144

Casilla 95: x=48 y=176 ➔ x=64 y=128

Casilla 115: x=240 y=144 ➔ x=256 y=96

Casilla 123: x=112 y=144 ➔ x=80 y=32

Casilla 172: x=176 y=96 ➔ x=176 y=48

Serpientes:

Casilla 92: x=256 y=176 ➔ x=208 y=224

Casilla 107: x=160 y=160 ➔ x=144 y=192

Casilla 99: x=32 y=160 ➔ x=32 y=240

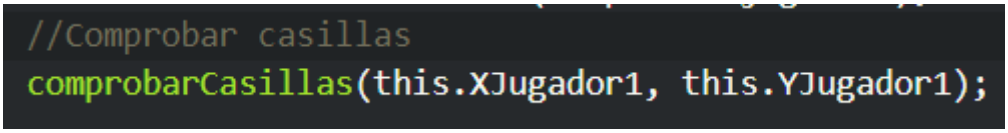
Casilla 176: x=240 y=96 ➔ x=160 y=112

Casilla 236: x=176 y=32 ➔ x=256 y=48

Casilla 253: x=80 y=16 ➔ x=32 y=64

Casilla ganadora 257: x=16 y=16 ➔ **GANÓ**

Vamos a crear una función llamada “comprobarCasillas” con dos parámetros (las coordenadas x e y del jugador y que verifique si se ha caído en alguna de estas casillas y que mueva y disminuya los puntos del jugador y la llamo en cada jugada.



De forma análoga se llamaría cuando juega el Jugador 2.

La función es la siguiente:

```
comprobarCasillas(casilla){
    console.log(casilla);
    if(casilla === 7){
        if(this.turno){
            this.XJugador1 = 104;
            this.YJugador1 = 296;
            db.puntos.jugador1 = 26;
            this.lineDeTiempo = this.tweens.timeline({
                targets: [this.jugador1],
                tweens: [
                    {
                        x: this.XJugador1,
                    },
                    {
                        y: this.YJugador1,
                    }
                ]
            });
            //Ahora se moverá hacia la izquierda
            this.direccionDcha1 = false;
        }else{
            this.XJugador2 = 104;
            this.YJugador2 = 296;
            db.puntos.jugador1 = 26;
            this.lineDeTiempo = this.tweens.timeline({
                targets: [this.jugador2],
```