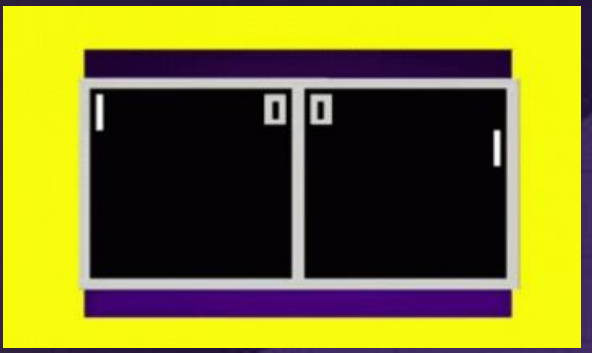
Update: 1/5/2024

Phaser 3
> Pong <



Mst. Erick Brenes.

Que haremos

- Uso de objetos.
- Uso de funciones.
- Creación de un juego.

1. Preparando el proyecto

Inicie el proyecto e instale phaser.

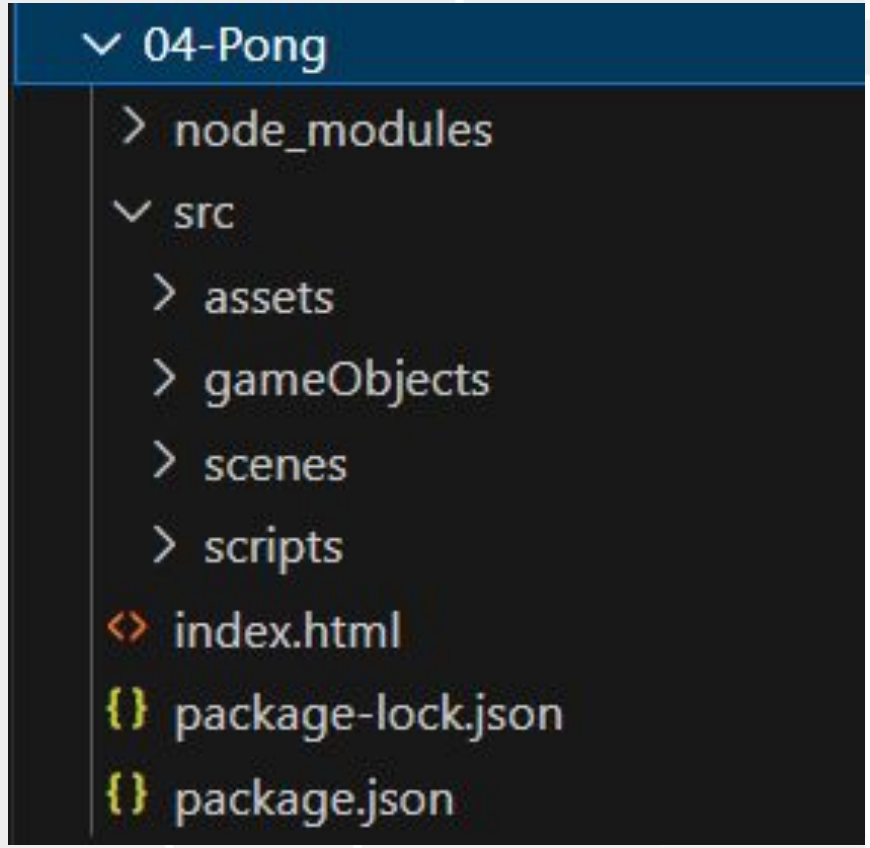
- npm init -y
- npm install --save phaser

Crear en la raíz:

- index.html
- carpeta [src]

Dentro de src crear:

- carpeta [assets]
- carpeta [gameObjects]
- carpeta [scenes]
- carpeta [scripts]



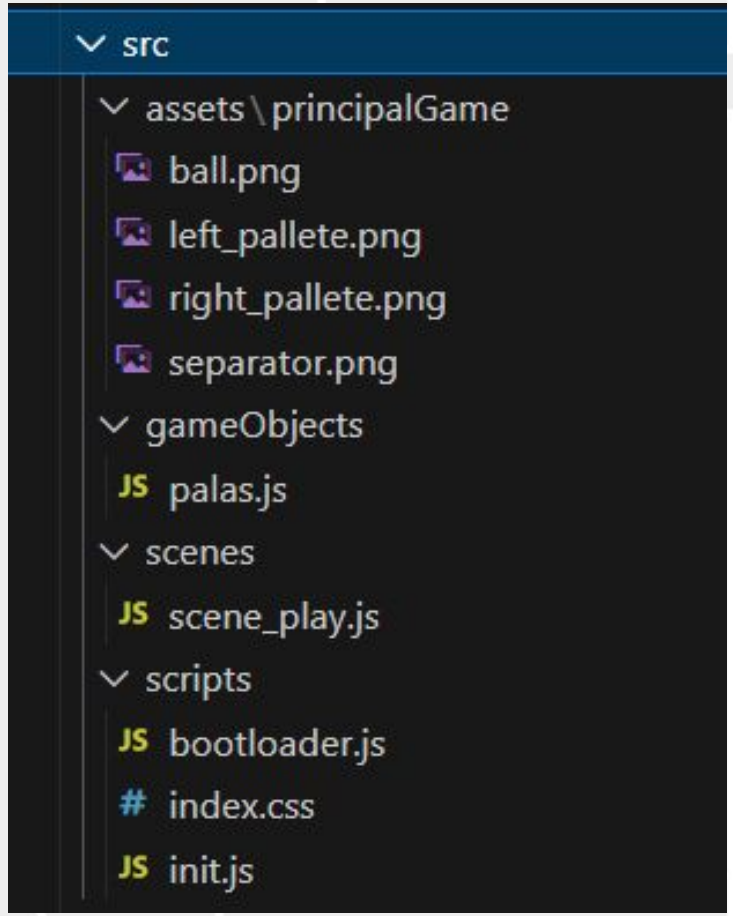
1. Preparando el proyecto

Assets

- Descargue los archivos que están en el moodle.

Archivos:

- palas.js [en la carpeta: gameObjects]
- Scene_Play.js [en la carpeta: scenes]
- bootloader.js [en la carpeta: scenes]
- init.js [en la carpeta: scenes]
- index.css [en la carpeta: scenes]



2. El Index

04-Pong > index.html > ...

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <link rel="stylesheet" href="./src/scripts/index.css">
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Cenfo-Pong</title>
    </head>
    <body>
        <main>
            <div id="contenedor"></div>
        </main>

        <div id="Scripts">
            <script src="./node_modules/phaser/dist/phaser.js"></script>
            <script type="module" src="./src/scripts/init.js"></script>
        </div>

    </body>
</html>
```

Cargue los scripts *Phaser* y el *init.js*

Cree un div llamado *contenedor*

3. Archivo init.js

04-Pong > src > scripts > JS init.js > ...

```
1   import Bootloader from './bootloader.js'
2   import Scene_Play from '../scenes/scene_play.js';
3
4   const config = {
5       width:  640,
6       height: 400,
7       parent: "contenedor",
8       type:   Phaser.AUTO,
9       backgroundColor:    '#392542',
10      scene: [Bootloader, Scene_Play],
11      physics:{
12          default: "arcade"
13      }
14  }
15
16  new Phaser.Game(config);
```

4. index.css

04-Pong > src > scripts > # index.css > ...

```css
main {
    width: 100%;
    height: 10vh;
    display: flex;
    justify-content: center;
}

#contenedor{
    margin-top: 50px;
}
```

4. Bootloader

04-Pong > src > scripts > JS bootloader.js > Bootloader

```
/*****************************************************************************
 * @class
 * La clase Bootloader representa la escena de carga del juego.
 * @extends Phaser.Scene
 * @since 5/1/2024
 * @creator Mst. Erick Brenes
 * @Updated 5/1/2024 by eBrenes
 * @version 1.0
 *****************************************************************************/

export default class Bootloader extends Phaser.Scene {
    /*************************************************************************
     * Constructor de la clase Bootloader.
     *************************************************************************/
    constructor() {
        super({ key: "Bootloader" });
    }

    /*************************************************************************
     * Método del ciclo de vida de Phaser. Se llama antes de que la escena se cargue.
     * Se utiliza para precargar los recursos necesarios para el juego.
     *************************************************************************/
    preload() {
        // Se establece un evento que se activará cuando la carga de recursos esté completa.
        this.load.on("complete", () => {
            // Inicia la escena de juego después de que se hayan cargado todos los recursos.
            this.scene.start("Scene_Play");
        });

        // Precarga de imágenes necesarias para el juego.
        this.load.image("bola", "./src/assets/principalGame/ball.png");
        this.load.image("izquierda", "./src/assets/principalGame/left_pallete.png");
        this.load.image("derecha", "./src/assets/principalGame/right_pallete.png");
        this.load.image("separador", "./src/assets/principalGame/separator.png");
    }
}
```

Universidad CENFOTEC
SOMOS LO QUE SABEMOS

5. Palas

04-Pong > src > gameObjects > JS palas.js > ...

```
/*****************************************************************************
 * @class
 * La clase Palas representa las palas del juego.
 * @extends Phaser.GameObjects.Sprite
 * @since 5/1/2024
 * @creator Mst. Erick Brenes
 * @Updated 5/1/2024 by eBrenes
 * @version 1.0
 *****************************************************************************/

export default class Palas extends Phaser.GameObjects.Sprite {
    /*************************************************************************
     * Constructor de la clase Palas.
     * @param {Phaser.Scene} scena - La escena a la que pertenece la pala.
     * @param {number} x - La posición X inicial de la pala.
     * @param {number} y - La posición Y inicial de la pala.
     * @param {string} type - El tipo de pala (izquierda/derecha).
     *************************************************************************/
    constructor(scena, x, y, type) {
        super(scena, x, y, type);
        scena.add.existing(this);
        scena.physics.world.enable(this);
        this.body.immovable = true;
        this.body.setCollideWorldBounds(true);
    }

    /*************************************************************************
     * Establece la velocidad vertical de la pala.
     * @param {number} pVel - La velocidad vertical deseada.
     *************************************************************************/
    setVelocity(pVel) {
        this.body.setVelocityY(pVel);
    }
}
```

6. scene_play

04-Pong > src > scenes > scene_play.js > Scene_Play > constructor

```
1   import Palas from './../gameObjects/palas.js';
2
3   /*****************************************************************************
4    * @class
5    * La clase Scene_Play representa la escena principal del juego.
6    * @extends Phaser.Scene
7    * @since 5/1/2024
8    * @creator Mst. Erick Brenes
9    * @Updated 5/1/2024 by eBrenes
10   * @version 1.0
11   *****************************************************************************/
12  export default class Scene_Play extends Phaser.Scene {
13      /**
14       * Constructor de la clase Scene_Play.
15       */
16      constructor() {
17          super({ key: "Scene_Play" });
18      }
19
```

6. scene_play

```
20  /*****************************************************************************
21   * Método del ciclo de vida de Phaser. Se llama una vez al inicio de la escena.
22   *****************************************************************************/
23  create() {
24      // Configuración de cuales bordes son permitidos salirse.
25      this.physics.world.setBoundsCollision(false, false, true, true);
26
27      // Controles para el jugador derecho
28      this.cursors = this.input.keyboard.createCursorKeys();
29
30      // Controles para el jugador izquierdo
31      this.key_W = this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.W);
32      this.key_S = this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.S);
33
34      // Valores para pintar en pantalla
35      let gW = this.getOptions(1);
36      let gH = this.getOptions(2);
37      let mP = this.getOptions(3);
38
39      // Creación de objetos palas
40      this.izq = new Palas(this, mP, gH / 2, "izquierda");
41      this.der = new Palas(this, gW - mP, gH / 2, "derecha");
42
43      // Creación de la línea separadora
44      this.add.image(gW / 2, gH / 2, "separador");
45
46      // Creación de la bola
47      this.bola = this.physics.add.image(gW / 2, gH / 2, "bola");
48
49      this.gamePhysicsConfig();
50
51      //puntajes
52      this.puntajeDer = 0;
53      this.puntajeIzq = 0;
54  }
```

6. scene_play

```
59      update() {
60          let gW = this.getOptions(1);
61          let gH = this.getOptions(2);
62          let bReset = (this.bola.x < 0) ? 1 : ((this.bola.x > gW) ? 2 : 0);
63
64          if (bReset !== 0) {
65              this.bola.setPosition(gW / 2, gH / 2);
66              this.bola.setVelocityX(this.velXBall(bReset));
67          }
68
69          // Validación del jugador derecho.
70          if (this.cursors.down.isDown)
71              this.der.setVelocity(300);
72          else if (this.cursors.up.isDown)
73              this.der.setVelocity(-300);
74          else
75              this.der.setVelocity(0);
76
77          // Validación del jugador izquierdo.
78          if (this.key_S.isDown)
79              this.izq.setVelocity(300);
80          else if (this.key_W.isDown)
81              this.izq.setVelocity(-300);
82          else
83              this.izq.setVelocity(0);
84      }
85
86      /**************************************************************************
87       * Determina la velocidad X de la bola según el lado al que se debe resetear.
88       * @param {number} pLado - Lado para resetear la bola (1 para izquierda, 2 para derecha).
89       * @returns {number} - Valor de velocidad X.
90       *************************************************************************/
91      velXBall(pLado) {
92          return (pLado === 1) ? 180 : -180;
93      }
```

6. scene_play

```
101     /**********************************************************************************
102      * Configura las propiedades de física del juego.
103      **********************************************************************************/
104     gamePhysicsConfig() {
105         this.bola.setCollideWorldBounds(true);
106         this.physics.add.collider(this.bola, this.izq, this.collitionPala, null, this);
107         this.physics.add.collider(this.bola, this.der, this.collitionPala, null, this);
108         this.bola.setBounce(1);
109         this.bola.setVelocityX(this.velXBall(1));
110     }
111
112     /**********************************************************************************
113      * Maneja la colisión entre la bola y las palas.
114      **********************************************************************************/
115     collitionPala() {
116         let vel = Phaser.Math.Between(-120, 120);
117         this.bola.setVelocityY(vel);
118     }
119
```

6. scene_play

```
120     /**************************************************************************
121      * Obtiene varias opciones basadas en el tipo proporcionado.
122      * @param {number} pType - Tipo de opción a recuperar.
123      * @returns {number} - El valor de la opción solicitada.
124      *************************************************************************/
125     getOptions(pType) {
126         let valor;
127         switch (pType) {
128             case 1: // Ancho del lienzo del juego
129                 valor = this.sys.game.config.width;
130                 break;
131             case 2: // Alto del lienzo del juego
132                 valor = this.sys.game.config.height;
133                 break;
134             case 3: // Margen de la barra en el borde lateral
135                 valor = 50;
136                 break;
137         }
138         return valor;
139     }
140 }
```

Tarea:

- Hacer la pelota un objeto como las paletas.
- Hacer que el juego se detenga cuando uno de los dos jugador anota 3 puntos puntos.
- Cargar una nueva escena con el nombre del ganador Derecha o Izquierda.
- Siga el video sobre animación. Modifiquelo para que la misma se ejecute solo si se presiona la tecla H