



UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco



Actividad 2

Michel Emanuel López Franco

Horario: Martes – Jueves

Hora: 11:00 – 13:00

Said Omar Hernández Grande

Código: #218515598

Índice de Phaser Trabajo No.-2 Javascript

Reporte sobre el código de Phaser	3
“Making Your First Phaser”	3
Index.html	3
Styles.css	4
Main.js	5
Conclusión:	11

Reporte sobre el código de Phaser

“Making Your First Phaser”

Index.html

Este proyecto consiste en crear un videojuego usando Phaser, un framework de JavaScript para juegos 2D. El archivo HTML tiene la siguiente estructura:

1. Metadatos:

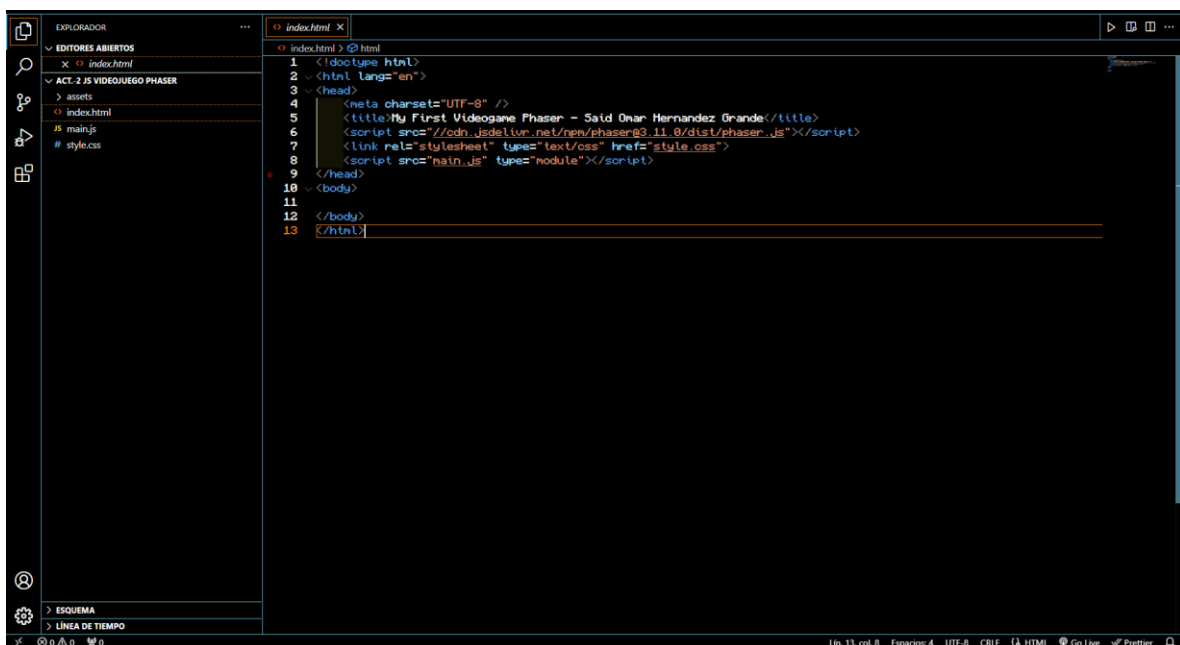
- Se define el documento como HTML5.
- El lenguaje es inglés, y el charset es UTF-8.
- El título es "My First Videogame Phaser - Said Omar Hernandez Grande".

2. Recursos:

- Se incluye Phaser desde un CDN.
- Se enlaza un archivo CSS (style.css) para estilos.
- Se carga el archivo JavaScript principal (main.js) que contiene la lógica del juego.

3. Cuerpo del Documento:

- El <body> está vacío, lo que indica que el contenido del juego se generará dinámicamente con JavaScript.



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <title>My First Videogame Phaser - Said Omar Hernandez Grande</title>
6   <script src="//cdn.jsdelivr.net/npm/phaser@3.11.0/dist/phaser.js"></script>
7   <link rel="stylesheet" type="text/css" href="style.css">
8   <script src="main.js" type="module"></script>
9 </head>
10 <body>
11
12 </body>
13 </html>
```

Styles.css

El código CSS proporcionado aplica estilos básicos al cuerpo del documento (body):

1. Márgenes:

- Elimina los márgenes (margin: 0;), lo que hace que el contenido se extienda de borde a borde en la página.

2. Alineación del Texto:

- Centra el texto horizontalmente en la página (text-align: center;).

3. Relleno:

- Añade un relleno de 150 píxeles en todos los lados (padding: 150px;), creando espacio alrededor del contenido dentro del cuerpo.



Main.js

Este proyecto define un juego simple utilizando Phaser, un framework de JavaScript para el desarrollo de juegos 2D. A continuación, se desglosan los principales componentes y funcionalidades del código:

1. Configuración del Juego

El juego se configura mediante un objeto config que define varias propiedades clave:

- **Tipo:** Phaser.AUTO, que permite a Phaser elegir automáticamente entre Canvas o WebGL, dependiendo del soporte del navegador.
- **Dimensiones:** El juego tiene un tamaño de 800 píxeles de ancho y 600 píxeles de alto.
- **Física:** Se utiliza el sistema de física "arcade" con una gravedad en el eje y de 300. La depuración está desactivada (debug: false).
- **Escenas:** Se definen tres funciones principales: preload, create, y update, que manejan la carga de recursos, la creación de objetos, y la lógica de actualización del juego, respectivamente.

Código:

```
1  var config = {
2      type: Phaser.AUTO,
3      width: 800,
4      height: 600,
5      physics: {
6          default: 'arcade',
7          arcade: {
8              gravity: { y: 300 },
9              debug: false
10         }
11     },
12     scene: {
13         preload: preload,
14         create: create,
15         update: update
16     }
17 };
18
```

Resultado:



2. Objetos Principales del Juego

- **Player (Jugador):** Un sprite que representa al personaje principal del juego. Puede moverse hacia la izquierda, derecha y saltar. Sus propiedades físicas incluyen un pequeño rebote (`setBounce(0.2)`) y la capacidad de colisionar con los límites del mundo (`setCollideWorldBounds(true)`).

Código:

```
// El jugador y sus configuraciones
player = this.physics.add.sprite(100, 450, 'dude');

// Propiedades físicas del jugador. Dale al pequeño un ligero rebote.
player.setBounce(0.2);
player.setCollideWorldBounds(true);
```

Resultado:



- **Stars (Estrellas):** Un grupo de objetos que el jugador puede recolectar. Las estrellas rebotan ligeramente al caer, con un rebote aleatorio entre 0.4 y 0.8.

Código:

```
// Algunas estrellas para recolectar, 12 en total, espaciadas uniformemente a 70 píxeles a lo largo del eje x
stars = this.physics.add.group({
  key: 'star',
  repeat: 11,
  setXY: { x: 12, y: 0, stepX: 70 }
});

stars.children.iterate(function (child) {
  // Dale a cada estrella un rebote ligeramente diferente
  child.setBounceY(Phaser.Math.FloatBetween(0.4, 0.8));
});
```

Resultado:

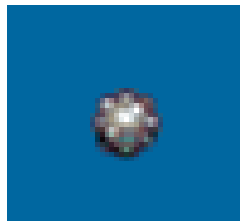


- **Bombs (Bombas):** Bombas que aparecen cuando el jugador recolecta todas las estrellas en pantalla. Las bombas rebotan y se mueven dentro del área del juego. Si una bomba golpea al jugador, el juego termina.

Código:

```
1 bombs = this.physics.add.group();
```

Resultado:



- **Platforms (Plataformas):** El suelo y las plataformas están hechos de un grupo estático que no se mueve, permitiendo que el jugador y otros objetos interactúen con ellos.

Código:

```
function create ()
{
    // Un fondo simple para nuestro juego.
    this.add.image(400, 300, 'sky');

    // El grupo de plataformas contiene el suelo y las 2 repisas sobre las que podemos saltar.
    platforms = this.physics.add.staticGroup();

    // Aquí creamos el terreno.
    // Escale para que se ajuste al ancho del juego (el sprite original tiene un tamaño de 400x32)
    platforms.create(400, 568, 'ground').setScale(2).refreshBody();

    // Ahora creamos algunas repisas.
    platforms.create(600, 400, 'ground');
    platforms.create(50, 250, 'ground');
    platforms.create(750, 220, 'ground');
}
```

Resultado:



3. Lógica del Juego

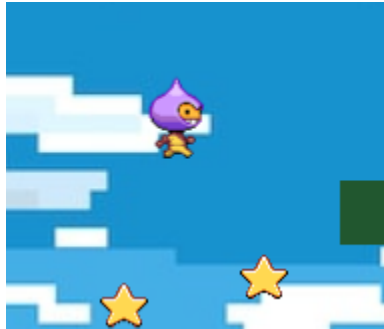
El juego sigue la lógica básica de movimiento y colisiones:

- **Movimiento del Jugador:** El jugador se mueve a la izquierda y derecha utilizando las teclas de flecha. Puede saltar si está tocando una plataforma.

Código:

```
117 function update ()
118 {
119     if (gameOver)
120     {
121         return;
122     }
123
124     if (cursors.left.isDown)
125     {
126         player.setVelocityX(-160);
127         player.anims.play('left', true);
128     }
129     else if (cursors.right.isDown)
130     {
131         player.setVelocityX(160);
132         player.anims.play('right', true);
133     }
134     else
135     {
136         player.setVelocityX(0);
137         player.anims.play('turn', true);
138     }
139
140     if (cursors.up.isDown && player.body.touching.down)
141     {
142         player.setVelocityY(-330);
143     }
144 }
145
146
147
```


Resultado:

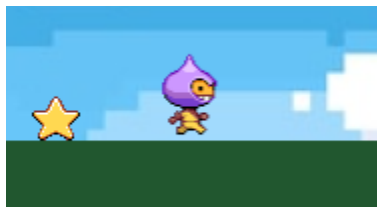


- **Colisiones:** El jugador puede colisionar con plataformas, estrellas, y bombas. Las estrellas desaparecen cuando son tocadas, y las bombas causan el final del juego.

Código:

```
106 // Choca al jugador y las estrellas con las plataformas.
107 this.physics.add.collider(player, platforms);
108 this.physics.add.collider(stars, platforms);
109 this.physics.add.collider(bombs, platforms);
110
111 // Comprueba si el jugador se superpone con alguna de las estrellas, si llama a la función CollectStar
112 this.physics.add.overlap(player, stars, collectStar, null, this);
113
114 this.physics.add.collider(player, bombs, hitBomb, null, this);
115
```

Resultado:

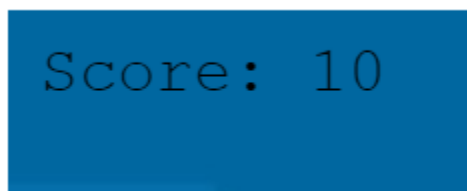


- **Puntuación:** El jugador obtiene puntos al recolectar estrellas, y la puntuación se muestra en la parte superior izquierda de la pantalla (scoreText).

Código:

```
103 // El puntaje
104 scoreText = this.add.text(16, 16, 'score: 0', { fontSize: '32px', fill: '#000' });
105
```

Resultado:



4. Eventos Especiales

- **Recolección de Estrellas:** Cuando el jugador recolecta una estrella, esta desaparece y la puntuación aumenta en 10 puntos. Cuando todas las estrellas han sido recolectadas, se generan nuevas estrellas y una bomba aparece aleatoriamente en la pantalla.

Código:

```
function collectStar (player, star)
{
    star.disableBody(true, true);

    // Y actualizamos el puntaje
    score += 10;
    scoreText.setText('Score: ' + score);

    if (stars.countActive(true) === 0)
    {
        // Un nuevo lote de estrellas para coleccionar
        stars.children.iterate(function (child) {
            child.enableBody(true, child.x, 0, true, true);
        });

        var x = (player.x < 400) ? Phaser.Math.Between(400, 800) : Phaser.Math.Between(0, 400);

        var bomb = bombs.create(x, 16, 'bomb');
        bomb.setBounce(1);
        bomb.setCollideWorldBounds(true);
        bomb.setVelocity(Phaser.Math.Between(-200, 200), 20);
        bomb.allowGravity = false;
    }
}
```

Resultado:



- **Golpe con Bomba:** Si una bomba golpea al jugador, el juego se pausa, el jugador se tiñe de rojo, y el estado de "game over" se activa, terminando el juego.

Código:

```
177 function hitBomb (player, bomb)
178 {
179     this.physics.pause();
180
181     player.setTint(0xff0000);
182
183     player.anims.play('turn');
184
185     gameOver = true;
186 }
187
```

Resultado:



Conclusión:

Bueno, esta actividad fue bastante entretenida y diferente a lo que estoy acostumbrado. Nunca había tenido una forma de aprender mientras hacía código, normalmente es teoría primero y luego un pequeño ejercicio. Con esto, fue como armar un rompecabezas donde al final terminas creando algo sorprendente. Me divertí mucho haciéndolo, y Phaser me pareció algo por donde empezar aun si no tienes bastante conocimiento del tema. El tutorial "Making your first Phaser" me gustó bastante, aunque debo admitir que la fecha de entrega es lo que me puso las pilas para hacerlo. Soy de esas personas que si no tienen una fecha límite, lo dejan casi para el último. A pesar de eso, me quedo con lo entretenido que fue hacerlo y, sobre todo, lo que aprendí en el proceso.