

Intro a Phaser 3



¿Que es phaser 3?

Phaser 3 es un marco (framework) de desarrollo de juegos en 2D para JavaScript.

Es especialmente popular entre los desarrolladores de juegos web y móviles.

Phaser 3 se utiliza para crear juegos interactivos y experiencias multimedia utilizando tecnologías web estándar como HTML, CSS y JavaScript.

Características

- Renderización WebGL y Canvas.
- Físicas y Colisiones.
- Manejo de Entradas.
- Animaciones y Sprites.
- Sonido y Música.
- Extensible y Personalizable.



Manos a la obra

Phaser desarrollo manual



1. Estructura del proyecto


Nombre	Fecha de modificación	Tipo	Tamaño
Assets	31/12/2023 13:08	Carpeta de archivos	
Scripts	31/12/2023 13:02	Carpeta de archivos	
index.html	2/1/2024 13:47	Chrome HTML Do...	1 KB




Datos (D:) > Desarrollo > HTML > Phaser > Juego1 > Scripts

Nombre	Fecha de modificación	Tipo	Tamaño
index.js	2/1/2024 13:52	JSFile	1 KB
juego.js	2/1/2024 17:31	JSFile	4 KB
phaser.min.js	31/12/2023 12:50	JSFile	1 140 KB

2. HTML básico

```
<> index.html >  html
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1">
6          <title>Primer Cenfo - Juego</title>
7
8      </head>
9      <body>
10         <h1> Bienvenidos </h1>
11
12         <script src="./Scripts/phaser.min.js"></script>
13         <script src="./Scripts/index.js" type="module"></script>
14     </body>
15 </html>
```

2. HTML básico

<> index.html >  html

1 <!DOCTYPE html>

2 <html lang="es">

3 <head>

4 <meta charset="UTF-8">

5 <meta name="viewport" content="width=device-width, initial-scale=1">

6 <title>Primer Juego</title>

7 </head>

8 <body>

9 <h1>Bienvenidos </h1>

10 <script src="Scripts/phaser.min.js"></script>

11 <script src="Scripts/index.js" type="module"></script>

12 </body>

13 </html>

Es un html común y corriente que contendrá el videojuego.

Este es un ejemplo muy básico pero se pueden usar más elementos para organizar mejor el código.

Importante asignar phaser al html.

Phaser 3 utiliza JS y typescript. Por lo tanto podemos usar objetos.

3. Archivo de configuración básica

```
1  import { Game } from './juego.js';
2
3  const config = {
4      type: Phaser.AUTO,
5      width: 800,
6      height: 800,
7      scene: [Game], //arreglo de pantallas del juego, menu, play, game over...
8      physics: { //tipos de fisicas que podriamos configurar
9          default: 'arcade',
10         arcade: {
11             gravity: {y:400},
12             debug: false
13         }
14     }
15 }
16
17 var game = new Phaser.Game(config); //instancia el nuevo juego
```


3. Archivo de configuración básica

```
1 import { Game } from './juego.js';
```

```
2
```

```
3 const config = {
```

```
4   type: Phaser.AUTO,
```

```
5   width: 800,
```

Type: funciona para escoger el tipo puede escoger entre CANVAS / WEBGL / AUTO

```
6   height: 800, //arreglo de pantallas del juego, menu, play, game over...
```

```
7   physics: { //tipos de fisicas que podriamos configurar
```

Scene: un arreglo con las posibles escenas del videojuego.

```
8     default: 'arcade',
```

```
9     arcade: {
```

Physics: permite seleccionar el tipo de físicas que vamos a usar en nuestro código, la más básica es *arcade*.

```
10       gravity: {y:400},
```

```
11       debug: false
```

```
12     }
```

```
13   }
```

```
14 }
```

```
15
```

```
16
```

```
17 var game = new Phaser.Game(config); //instancia el nuevo juego
```

4. Clase base de Escenas

```
export class Game extends Phaser.Scene {
  constructor(){
  }

  /**
   * Hace una carga inicial de assets para usar en el juego.
   */
  preload(){
  }

  /**
   * Es la funcion que se encarga de "colocar" los assets precargados
   * en escena. Usa Ajax
   */
  create(){
  }
}
```

5. Constructor

```
export class Game extends Phaser.Scene {
  constructor(){
    super({key: 'game'}); //nombre de la escena que corresponde al arreglo donde esta la configuración
  }

  /**
   * Hace una carga inicial de assets para usar en el juego.
   */
  preload(){

  }

  /**
   * Es la funcion que se encarga de "colocar" los assets precargados
   * en escena. Usa Ajax
   */
  create(){

  }
}
```

5. Constructor

```
export class Game extends Phaser.Scene {  
  constructor(){  
    super({key: 'game'}); //nombre de la escena que corresponde al arreglo donde esta la configuración  
  }  
  
  /**  
   * Vemos que es una clase que se puede exportar y que extiende de  
   * Phaser.Scenes  
   */  
  preload(){  
    En la creación de esta pantalla estamos colocando el nombre de la  
    escena que corresponde a la que tenemos en la variable de  
    configuración.  
    /**  
     * Se encarga de "colocar" los assets precargados  
     * en escena. Usa Ajax  
     */  
  }  
  create(){  
  }  
}
```

Agreguémono assets al juego

```
export class Game extends Phaser.Scene {
  constructor(){
    super({key: 'game'}); //nombre de la escena que corresponde al arreglo donde esta la configuración
  }

  /**
   * Hace una carga inicial de assets para usar en el juego.
   */
  preload(){
    var assetPath = 'Assets/Imagenes/'

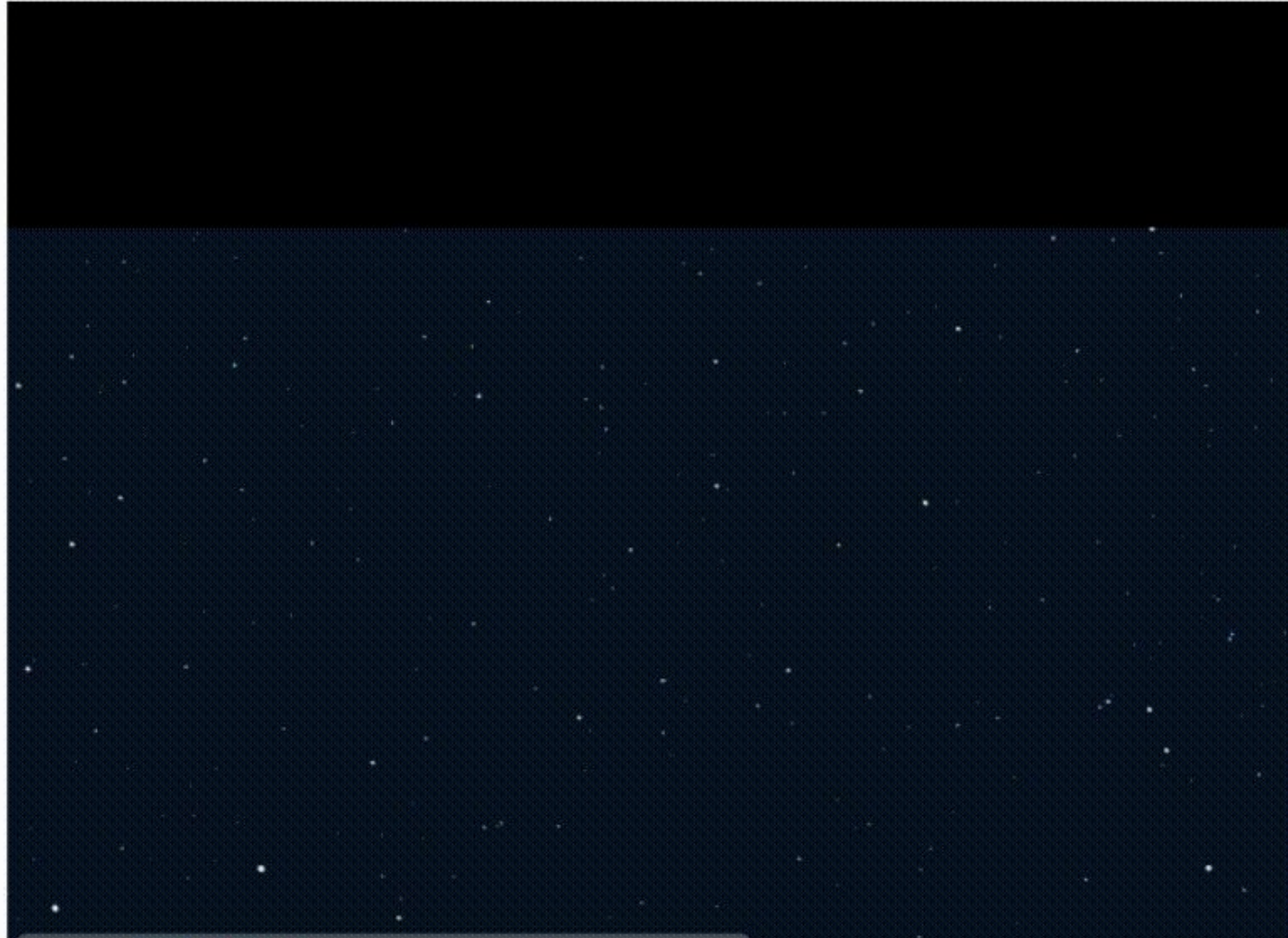
    this.load.image('background', assetPath + 'background.png');
    this.load.image('gameover', assetPath + 'gameover.png');
    this.load.image('plataforma', assetPath + 'CenfoPlataformaAzul.png');
    this.load.image('fireball', assetPath + 'BolaFuego.png');
  }
}
```


6. Agreguemos las imágenes a la pantalla

```
19
20  /*****
21  * Es la funcion que se encarga de "colocar" los assets precargados
22  * en escena. Usa Ajax
23  *****/
24  create(){
25      this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
26      this.gameoverImage = this.add.image(400,250,'gameover');
27      this.gameoverImage.visible = false;
28
29      //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
30      this.plataform = this.physics.add.image(400, 460, 'plataforma');
31
32
33
34
35
36      //agregamos la pelota
37      this.ball = this.physics.add.image(400, 30, 'fireball');
38  }
39
```

Primera ejecución del juego.

Bienvenidos



Pero ¿qué paso?

Al crear la imagen de plataforma usando *physics* le estamos agregando físicas que le aplicarán al elemento. En nuestra configuración existe la gravedad por lo tanto la plataforma tiene ese efecto aplicado a él.

La imagen game over, si le asignan true al visible podrán ver que a esta no le afecta la gravedad.

Para solucionarlo hay que decirle a Phaser que la plataforma no le afecta la física.



7. Solución a la gravedad

```
19
20 /*****
21  * Es la funcion que se encarga de "colocar" los assets precargados
22  * en escena. Usa Ajax
23  *****/
24 create(){
25     this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
26     this.gameoverImage = this.add.image(400,250,'gameover');
27     this.gameoverImage.visible = false;
28
29     //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
30     this.plataform = this.physics.add.image(400, 460, 'plataforma');
31     this.plataform.body.allowGravity = false;
32
33
34
35
36
37
38 }
39
```


8. Agregamos movimiento de la plataforma con teclado

```
19
20  /*****
21  * Es la funcion que se encarga de "colocar" los assets precargados
22  * en escena. Usa Ajax
23  *****/
24  create(){
25      this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
26      this.gameoverImage = this.add.image(400,250,'gameover');
27      this.gameoverImage.visible = false;
28
29      //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
30      this.plataform = this.physics.add.image(400, 460, 'plataforma');
31      this.plataform.body.allowGravity = false;
32
33      //trabajamos con las pulsaciones del keyboard
34      this.cursors = this.input.keyboard.createCursorKeys();
35
36
37
38  }
39
```


8. Agregamos movimiento de la plataforma con teclado

```
/* ****  
 * Función que se ejecuta cada X tiempo desde que se inicia el código  
 * del juego.  
 **** */  
update(){  
    var iVel = 0;  
  
    if(this.cursors.left.isDown)  
        iVel = -500;  
  
    if(this.cursors.right.isDown)  
        iVel = 500;  
  
    this.plataform.setVelocityX(iVel);  
}
```

8. Agregamos movimiento de la plataforma con teclado

```
/* **** */
```

```
* Función que se ejecuta cada X tiempo desde que se inicia el código  
* del juego.
```

```
**** */
```

```
update(){
```

Esta función se ejecuta cada segundo. Va a revisar lo que pusimos dentro. En este caso si presionamos a la derecha se da un valor 500 positivo si presionamos izquierdo pondrá un valor 500 negativo que mueve la plataforma hacia la izquierda de la pantalla.

```
var iVel = 0;
```

```
if(this.cursors.left.isDown)
```

```
iVel = -500;
```

```
if(this.cursors.right.isDown)
```

```
iVel = 500;
```

```
this.plataform.setVelocityX(iVel);
```

```
}
```

9. Agregamos la bola a la pantalla

```
19  /*****
20  * Es la funcion que se encarga de "colocar" los assets precargados
21  * en escena. Usa Ajax
22  *****/
23  create(){
24      this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
25      this.gameoverImage = this.add.image(400,250,'gameover');
26      this.gameoverImage.visible = false;
27
28      //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
29      this.plataform = this.physics.add.image(400, 460, 'plataforma');
30      this.plataform.body.allowGravity = false;
31
32      //trabajamos con las pulsaciones del keyboard
33      this.cursors = this.input.keyboard.createCursorKeys();
34
35      //agregamos la pelota
36      this.ball = this.physics.add.image(400, 30, 'fireball');
37  }
38
39
```


Segunda ejecución

Bienvenidos

CENFO PLATAFORMA

Pero ¿qué paso?

Bueno agregamos la bola y esta tiene fisica, pero no ve la plataforma como parte de su mundo. En este momento es un objeto extra que no tiene ninguna propiedad que diga que hacer en el momento de que ambos objetos colisionen.

Lo solucionamos con agregar collision en los objetos



10. Agregamos una Colisión en el código

```
20 /*****
21  * Es la funcion que se encarga de "colocar" los assets precargados
22  * en escena. Usa Ajax
23  *****/
24 create(){
25     this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
26     this.gameoverImage = this.add.image(400,250,'gameover');
27     this.gameoverImage.visible = false;
28
29     //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
30     this.plataform = this.physics.add.image(400, 460, 'plataforma');
31     this.plataform.body.allowGravity = false;
32
33     //trabajamos con las pulsaciones del keyboard
34     this.cursors = this.input.keyboard.createCursorKeys();
35
36     //agregamos la pelota
37     this.ball = this.physics.add.image(400, 30, 'fireball');
38     this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision entre pelota y plataforma
39 }
```

Tercera ejecución

Bienvenidos

Pero ¿qué paso?

Ok, lo que pasa es “FISICA”

Cuando un objeto está en reposo y otro colisiona, se produce una transferencia de energía. Puede resultar en movimiento, deformación o cambio de velocidad, dependiendo de las masas y velocidades involucradas.

En este caso, está transmitiendo la fuerza de la pelota a la plataforma.



11. Solución de transferencia de fuerza

```
/* *****  
* Es la funcion que se encarga de "colocar" los assets precargados  
* en escena. Usa Ajax  
***** */  
create(){  
  this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos  
  this.gameoverImage = this.add.image(400,250,'gameover');  
  this.gameoverImage.visible = false;  
  
  //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.  
  this.plataform = this.physics.add.image(400, 460, 'plataforma').setImmovable();// No permite que le afecte la fisica de otro objeto  
  this.plataform.body.allowGravity = false;  
  
  //trabajamos con las pulsaciones del keyboard  
  this.cursors = this.input.keyboard.createCursorKeys();  
  
  //Agregamos la pelota  
  this.ball = this.physics.add.image(400, 30, 'fireball');  
  this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision entre pelota y plataforma  
}
```


12. Agregar rebote de la pelota

```
/* *****  
* Es la funcion que se encarga de "colocar" los assets precargados  
* en escena. Usa Ajax  
***** */  
create(){  
  this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos  
  this.gameoverImage = this.add.image(400,250,'gameover');  
  this.gameoverImage.visible = false;  
  
  //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.  
  this.plataform = this.physics.add.image(400, 460, 'plataforma').setImmovable();// No permite que le afecte la fisica de otro objeto  
  this.plataform.body.allowGravity = false;  
  
  //trabajamos con las pulsaciones del keyboard  
  this.cursors = this.input.keyboard.createCursorKeys();  
  
  //Agregamos la pelota  
  this.ball = this.physics.add.image(400, 30, 'fireball');  
  this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision entre pelota y plataforma  
  this.ball.setBounce(0.9); //Permitimos que rebote. Primer parametro cantidad de fuerza.  
}
```


12. Agregar rebote de la pelota

```
/* *****  
* Es la función que se encarga de "colocar" los assets precargados  
* en escena. Usa Ajax  
* ***** */  
create(){  
  this.add.image(100, 250, 'background'); //Creamos la imagen de la imagen de los fondos que vamos a usar  
  this.gameoverImage = this.add.image(400, 250, 'gameover');  
  this.g...  
  //Agregamos la plataforma a una variable y lo hacemos por medio de físicas, para poder hacer uso de ellas.  
  this.plataform = this.physics.add.image(400, 460, 'plataforma').setImmovable();// No permite que le afecte la fisica de otro objeto  
  this.plataform.body.allowGravity = false;  
  //trabajamos con las pulsaciones del keyboard  
  this.cursors = this.input.keyboard.createCursorKeys();  
  //Agregamos la pelota  
  this.ball = this.physics.add.image(400, 30, 'fireball');  
  this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision entre pelota y plataforma  
  this.ball.setBounce(0.9); //Permitimos que rebote. Primer parametro cantidad de fuerza.  
}
```

***Set Immovable* en la creación de la plataforma hace que la misma no se vea afectada con el impacto de la pelota, es decir no afecta la física al momento de la colisión.**

***el Set Bounce* estamos agregando un valor de rebote. En esta caso al ser menor a 1, pierde fuerza. Si se agrega valores mayores a 1 aumentaría fuerza.**

13. Bola hacia diferentes lados

```
/**
 * Es la funcion que se encarga de "colocar" los assets precargados
 * en escena. Usa Ajax
 */
create(){
    this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos
    this.gameoverImage = this.add.image(400,250,'gameover');
    this.gameoverImage.visible = false;

    //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.
    this.plataform = this.physics.add.image(400, 460, 'plataforma').setImmovable();// No permite que le afecte la fisica de otro objeto
    this.plataform.body.allowGravity = false;

    //trabajamos con las pulsaciones del keyboard
    this.cursors = this.input.keyboard.createCursorKeys();

    //Agregamos la pelota
    this.ball = this.physics.add.image(400, 30, 'fireball');
    this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision entre pelota y plataforma
    this.ball.setBounce(1); //Permitimos que rebote. Primer parametro cantidad de fuerza.
    var ballVel = 100 * Phaser.Math.Between(1.3, 2);

    if (Phaser.Math.Between(0, 10) > 5) ballVel = 0 - ballVel;

    this.ball.setVelocity(ballVel,10);
}
```

Cuarta Ejecución



Pero ¿qué paso?

Ya casi lo tenemos, ahora el problema es que la pelota se sale del marco de ejecución al igual que la plataforma. Si en el update pusieramos un *console.log* notaríamos que sigue la ejecución y esto no es bueno para la memoria de la computadora. Hay que agregar limites en el mundo y a quienes deben apegarse a ellos.



14. Colocamos los límites de la escena

```
/* *****  
 * Es la funcion que se encarga de "colocar" los assets precargados  
 * en escena. Usa Ajax  
 ***** */  
create(){  
    this.physics.world.setBoundsCollision(true,true,true,false); //Delimitamos el mundo para que solo se pueda ir hacia abajo  
  
    this.add.image(400, 250, 'background'); //Coordenadas. Usa el centro de la imagen en las coordenadas que ponemos  
    this.gameoverImage = this.add.image(400,250,'gameover');  
    this.gameoverImage.visible = false;  
  
    //Agregamos la plataforma a una variable y lo hacemos por medio de fisicas, para poder hacer uso de ellas.  
    this.plataform = this.physics.add.image(400, 460, 'plataforma').setImmovable();// No permite que le afecte la fisica de otro objeto  
    this.plataform.body.allowGravity = false;  
  
    //trabajamos con las pulsaciones del keyboard  
    this.cursors = this.input.keyboard.createCursorKeys();  
  
    //Agregamos la pelota  
    this.ball = this.physics.add.image(400, 30, 'fireball');  
    this.physics.add.collider(this.ball, this.plataform); //Agregamos una colision en etre pelota y plataforma  
    this.ball.setBounce(1); //Permitimos que rebote. Primer parametro cantidad de fuerza.  
    var ballVel = 100 * Phaser.Math.Between(1.3, 2);  
  
    if (Phaser.Math.Between(0, 10) > 5) ballVel = 0 - ballVel;  
  
    this.ball.setVelocity(ballVel,10);  
  
    this.ball.setCollideWorldBounds(true);  
    this.plataform.setCollideWorldBounds(true);  
}
```


15. ¿Y si perdemos? Game Over

```
/* ****  
 * Función que se ejecuta cada X tiempo desde que se inicia el código  
 * del juego.  
 **** */  
update(){  
    var iVel = 0;  
  
    if(this.cursors.left.isDown)  
        iVel = -500;  
  
    if(this.cursors.right.isDown)  
        iVel = 500;  
  
    this.plataform.setVelocityX(iVel);  
  
    if(this.ball.y > 500){  
        this.gameoverImage.visible = true;  
        this.scene.pause();  
    }  
}
```

15. ¿Y si perdemos? Game Over

```
/* ****  
 * Función que se ejecuta cada x tiempo desde que se inicia el código  
 * del juego.  
 **** */  
update(){  
    var iVel = 0;  
  
    if (this.ball.y < -500){  
        iVel = -500;  
    }  
    if (this.ball.y > 500){  
        iVel = 500;  
    }  
    this.plataform.setVelocityX(iVel);  
  
    if (this.ball.y > 500){  
        this.gameoverImage.visible = true;  
        this.scene.pause();  
    }  
}
```

Actualizamos la función *update* para que al momento de que la bola está por debajo del tamaño máximo de nuestro canvas de juego, aparezca la pantalla de game over y se ponga en pausa la escena.

Ejecución final

Bienvenidos



CENFO PLATAFORMA

