CS-PRO-C47

COLISIÓN DE AUTOS

¿Cuál fue nuestro objetivo para este módulo?

Aprendimos a realizar colisiones entre autos y obstáculos.

¿Qué logramos en clase el día de hoy?

- Escribimos una función para realizar colisiones entre un grupo de obstáculos y autos.
- Actualizamos **player.life** - ***vida.jugador*** después de cada colisión.

¿Qué conceptos y bloques de código vimos hoy?

- Colisiones.
- Evitamos las colisiones contiguas.

¿Cómo hicimos las actividades?

1. Creamos **handleCollisions(index) - *manejodeColisión (índice)***:
    - Detectamos colisiones entre el arreglo de **autos** y los grupos de **obstáculos** utilizando una condición.
    - En colisión, redujimos la **player.life - *vida.jugador*** entre 4

      Nota: player.life puede dividirse entre cualquier valor.
    - Actualizamos la **player.life** en la base de datos utilizando **update() - *actualizar()***

```
handleObstacleCollision(index) {
  if (cars[index - 1].collide(obstacles)) {
    if (player.life > 0) {
      player.life -= 185 / 4;
    }

    player.update();
  }
}
```

- Llamamos **handleCollisions(index) - *manejo de colisión (índice)*** dentro del método **play() - *juego()***

```
if (index === player.index) {
  stroke(10);
  fill("red");
  ellipse(x, y, 60, 60);

  this.handleFuel(index);
  this.handlePowerCoins(index);
  this.handleObstacleCollision(index);
```

2. Actualizamos el valor de **vida** en la base de datos.
    - Modificamos la **update()** en **player.js**

```
update() {
  var playerIndex = "players/player" + this.index;
  database.ref(playerIndex).update({
    positionX: this.positionX,
    positionY: this.positionY,
    rank: this.rank,
    score: this.score,
    life: this.life
  });
}
```

3. Alejamos el auto del obstáculo para evitar que la vida pasara a cero en la primera colisión:
    - Creamos una propiedad en **this.leftKeyActive= false -** ***esta.TeclaIzquierdaActiva=falso*** en **constructor()** de **Game.js**.

```
class Game {
  constructor() {
    this.resetTitle = createElement("h2");
    this.resetButton = createButton("");

    this.leadeboardTitle = createElement("h2");

    this.leader1 = createElement("h2");
    this.leader2 = createElement("h2");
    this.playerMoving = false;
    this.leftKeyActive = false;

  }
```

- Revisamos cual tecla es presionada por el jugador utilizando esta propiedad en **handlePlayerControls()- *manejodeControlesdeJugador.***
- Cambiamos el valor de **this.leftKeyActive** a **verdadero** o **falso** en función de la tecla que presione el jugador.

```
handlePlayerControls() {

  if (!this.blast) {
    if (keyIsDown(UP_ARROW)) {
      this.playerMoving = true;
      player.positionY += 10;
      player.update();
    }

    if (keyIsDown(LEFT_ARROW) && player.positionX > width / 3 - 50) {
      this.leftKeyActive = true;
      player.positionX -= 5;
      player.update();
    }

    if (keyIsDown(RIGHT_ARROW) && player.positionX < width / 2 + 300) {
      this.leftKeyActive = false;
      player.positionX += 5;
      player.update();
    }
  }
}
```

4. Movimos el auto a la izquierda o derecha en función a la tecla de flecha presionada por el jugador.

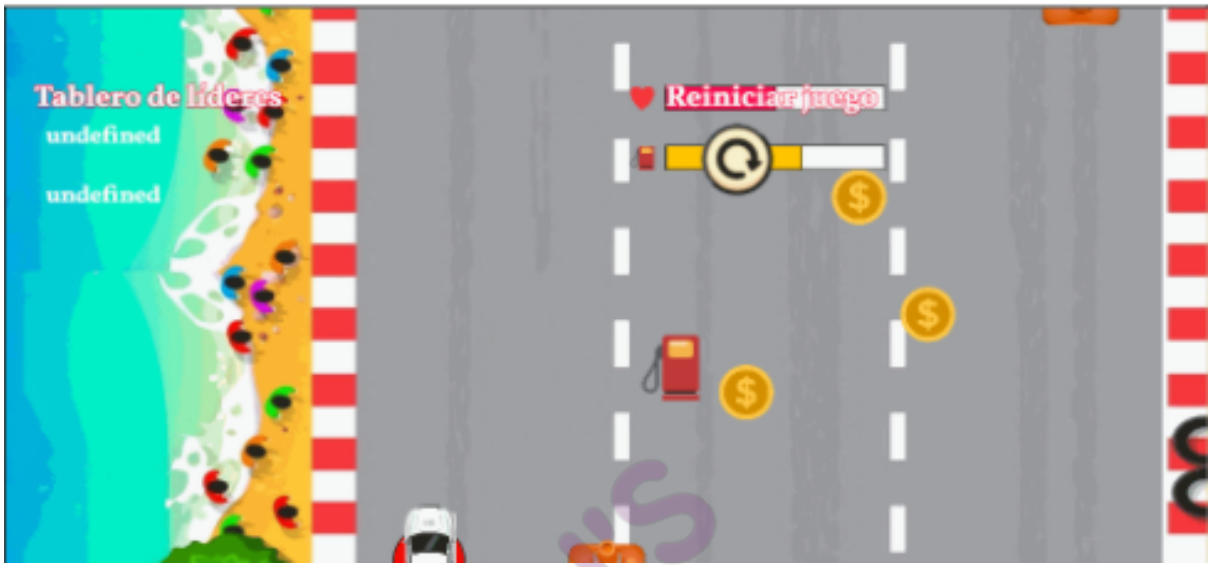
```
handleObstacleCollision(index) {
  if (cars[index - 1].collide(obstacles)) {

    if (this.leftKeyActive) {
      player.positionX += 100;
    } else {
      player.positionX -= 100;
    }

    if (player.life > 0) {
      player.life -= 185 / 4;
    }

    player.update();
  }
}
```

OUTPUT:



¿Qué sigue?

En la siguiente clase, revisaremos la colisión entre autos y cambiaremos la animación cuando el valor de **player.life** sea cero.

AMPLÍA TU CONOCIMIENTO:

1. Las funciones de colisión siempre hacen uso de los colisionadores incorporados creados para cada sprite. Utiliza el siguiente enlace para saber más sobre el método de **collide() - *colisionar*** en **JavaScript**: https://studio.code.org/docs/gamelab/collide/