COLLIDING CARS

What is our GOAL for this MODULE?

In this class, we used the concept of collision to detect collisions between cars and with obstacles. Changed car image to a "BOOM" image and disabled car controls when the player's life is over.

What did we ACHIEVE in the class TODAY?

- Reduced the life of a player, when cars collided with each other or with obstacles.
- Changed the animation of the car, when the life of the player reduces to 0.
- Disabled control of cars, once the animation is changed.

Which CONCEPTS / CODING BLOCKS did we cover today?

- Collision
- Bouncing away after each collision.
- Disable the control without changing the **gameState.**

How did we DO the activities?

To first detect the collision between cars and obstacles and reduce the life of the player.

1. Create **handleCollisions(index)**:
    - Detect collisions between **cars** array and **obstacles** group using a condition.
    - On collision, reduce **player.life** by **4**.
    - Update **player.life** in the database using **update().**

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

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

    player.update();
  }
}
```

- Call **handleCollisions(index)** inside the **play()** method.

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

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

2. To update the value of **life** in the database.
    - Modify **update()** in **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
  });
}
```

Bounce the car away from obstacle post-collision.

3. Move the car away from the obstacle to stop life becoming zero immediately on the very first collision:
    - Create a property in **this.leftKeyActive= false** in **constructor()** of **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;
  }
```

    - Check which key is pressed by the player we are using this property in **handlePlayerControls().**
    - Change the value of **this.leftKeyActive** to **true** or **false** based on which key is pressed by the player.

```
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. Shift the car to the left or right of its current position based on the arrow key pressed by the player.

```
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();
  }
}
```

Now to update the image of the car to the “Boom” image if the car crashes and the player runs out of lives.

5. Preload image for the blast in **sketch.js**.

```
sketch.js > ...
1   var canvas;
2   var backgroundImage, car1_img, car2_img, track;
3   var fuelImage, powerCoinImage, lifeImage, obstacl
4   var blastImage;                    //C41// TA
5   var database, gameState;
6   var form, player, playerCount;
7   var allPlayers, car1, car2, fuels, powerCoins, ob
8   var cars = [];
9
0   function preload() {
1     backgroundImage = loadImage("./assets/backgrour
2     car1_img = loadImage("../assets/car1.png");
3     car2_img = loadImage("../assets/car2.png");
4     track = loadImage("../assets/track.jpg");
5     fuelImage = loadImage("./assets/fuel.png");
6     powerCoinImage = loadImage("./assets/goldCoin.p
7     lifeImage = loadImage("./assets/life.png");
8     obstacle1Image = loadImage("./assets/obstacle1.
9     obstacle2Image = loadImage("./assets/obstacle2.
0     blastImage = loadImage("./assets/blast.png");
1   }
2
```

6. Add this image to car sprites in the **Start()** method of **Game.js**.

```
start() {
  player = new Player();
  playerCount = player.getCount();

  form = new Form();
  form.display();

  car1 = createSprite(width / 2 - 50, height - 100);
  car1.addImage("car1", car1_img);
  car1.scale = 0.07;

  car1.addImage("blast", blastImage);

  car2 = createSprite(width / 2 + 100, height - 100);
  car2.addImage("car2", car2_img);
  car2.scale = 0.07;

  car2.addImage("blast", blastImage);

  cars = [car1, car2];
```

7. Change the car's animation based on the value of **player.life**.
    - **Play()** method is running in each **frameCount** of the **draw()** function.
    - Get an updated **life** count for the player in each **frameCount**.
    - Use the **player.life** property to change the animation of the car.

```
//index of the array
var index = 0;
for (var plr in allPlayers) {
  //add 1 to the index for every loop
  index = index + 1;

  //use data form the database to display the cars in x and y direction
  var x = allPlayers[plr].positionX;
  var y = height - allPlayers[plr].positionY;

  //save the value of player.life in temp variable currentlife
  var currentlife = allPlayers[plr].life;

  if (currentlife <= 0) {
    cars[index - 1].changeImage("blast");
    cars[index - 1].scale = 0.3;
  }

  cars[index - 1].position.x = x;
  cars[index - 1].position.y = y;
```

Now, to detect the collision between cars.

8. Create a new **handleCarACollisionwithCarB(index)** function.

```
handleCarACollisionWithCarB(index) {
  if (index === 1) {
    if (cars[index - 1].collide(cars[1])) {
      if (this.leftKeyActive) {
        player.positionX += 100;
      } else {
        player.positionX -= 100;
      }

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

```
      player.update();
    }
  }
  if (index === 2) {
    if (cars[index - 1].collide(cars[0])) {
      if (this.leftKeyActive) {
        player.positionX += 100;
      } else {
        player.positionX -= 100;
      }

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

      player.update();
    }
  }
}
```


- Call it in **play()** method.

```
      cars[index - 1].position.x = x;
      cars[index - 1].position.y = y;

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

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

9. Disable the control after **player.life = 0**:
    - Create a property in the **constructor()** of **Game.js** to check the status of the blast.
    - Keep **this.blast = false** initially.

```
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;
    this.blast = false;
  }
```

    - Change **this.blast** to **true**, when **player.life <= 0**.
    - Change the **playerMoving** to **false** (created in the last class).

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

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


  if (player.life <= 0) {
    this.blast = true;
    this.playerMoving = false;
  }
```

10. Add the **if** condition in **handlePlayerControl()** to read controls by arrow key when **this.blast** is **false.**

```
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();
    }
  }
}
```

Output:

What's next?

In the next class, you will be learning important concepts of Game Elements.

EXTEND YOUR KNOWLEDGE:

Bookmark the following link to know more about the Firebase Database:
https://firebase.google.com/use-cases