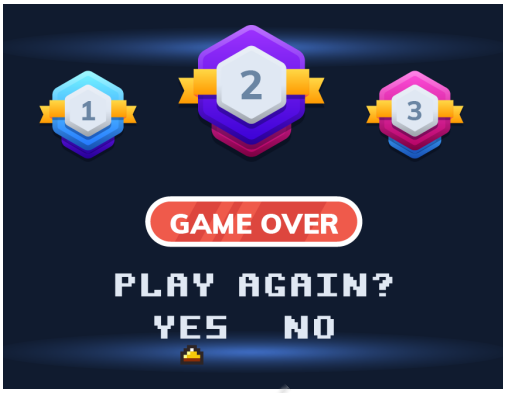
PLAYER RANKING & GAME OVER



What is our GOAL for this MODULE?

In this class, we built a ranking mechanism that ranks the player according to their performance in the car racing game. We also built a progress bar for fuel and the player's life property.

What did we ACHIEVE in the class TODAY?

- Created database query to update and read player rank from the database.
- Defined finish line for players to end the game.
- Changed **gameState** to 2.
- Created Progress bar for Fuel & Life.
- Updated **handleFuel()** function to show status of fuel.
- Used **swal()** to display a pop-up message for winner/game over.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Creating progress bar
- **Swal()** from SweetAlert
- Game over & game won conditions
- Moving car without pressing any key (using a flag)

How did we DO the activities?

1. Create a property in the **constructor()** of **player.js**.

```
class Player {
  constructor() {
    this.name = null;
    this.index = null;
    this.positionX = 0;
    this.positionY = 0;
    this.rank = 0;
    this.fuel = 185;
    this.life = 185;
    this.score = 0;
  }
```

2. Add a **"CarsAtEnd"** field in the database.

3. Write a method **getCarsAtEnd()** to read rank from the database.
    - Write a **static updateCarsAtEnd()** function to update the same.

```
getCarsAtEnd(){
  database.ref('carsAtEnd').on("value",(data)=>{
    this.rank = data.val()
  })
}

static updateCarsAtEnd(rank) {
  database.ref("/").update({
    carsAtEnd: rank
  });
}
```

- Call **getCarsAtEnd()** in the **play()** function of **game.js.**

```
play() {
  this.handleElements();
  this.handleResetButton();

  Player.getPlayersInfo();
  player.getCarsAtEnd();
```

- Add a command to reset the Cars_At_End field of the database in handleResetButton() of Game.js.

```
handleResetButton() {
  this.resetButton.mousePressed(() => {
    database.ref("/").set({
      playerCount: 0,
      gameState: 0,
      players: {},
      carsAtEnd: 0
    });
    window.location.reload();
  });
}
```

4. Create a **const finishLine** declaring the finish line to be 100 px less than the length of trackImage.

- Write an if condition to compare the y-position of the player's car with the **finishLine.**
- When the car crosses the finish line, **gameState** is changed to **2 (END)**.
- Call **showRank()** function.
- At the same time update data for the player in the database.

```
const finshLine = height * 6 - 100;

if (player.positionY > finshLine) {
  gameState = 2;
  player.rank += 1;
  Player.updateCarsAtEnd(player.rank);
  player.update();
  this.showRank();
}
```

5. Create a **showRank()** method to display player rank using **swal()** in **game.js.**

   - **swal()** function accepts properties like:
     - title:
     - text:
     - imageURL:
     - imageSize:
     - confirmationButtonText:

```
showRank() {
  swal({
    title: `Awesome!${"\n"}Rank${"\n"}${player.rank}`,
    text: "You reached the finish line successfully",
    imageUrl:

"https://raw.githubusercontent.com/vishalgaddam873/p5-multiplayer-car-race-game/master/assets/cup.png",
    imageSize: "100x100",
    confirmButtonText: "Ok"
  });
}
```

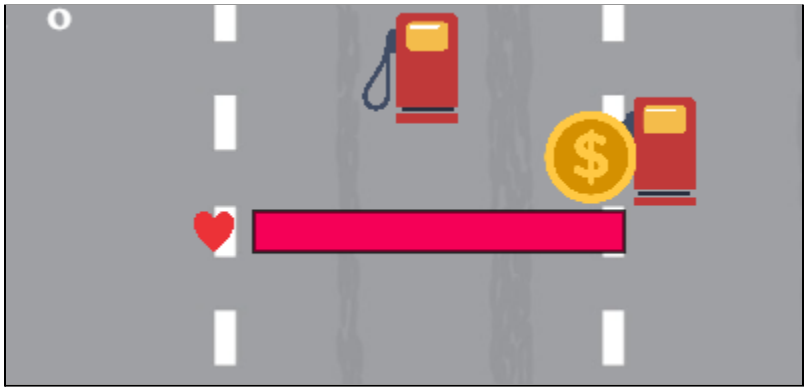
6. Add a library **sweetalert.css** in **index.html** to use **swal()** function.

```
<!-- Sweet Alert c40-->
<script
  src="https://code.jquery.com/jquery-3.5.1.min.js"
  integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
  crossorigin="anonymous"
></script>
<script src="./lib/sweetalert.min.js"></script>
<link rel="stylesheet" type="text/css" href="./lib/sweetalert.css" />
```

7. Create a progress bar for fuel and power coin.
    - Create **showLife()** for power coin.
    - Assign the position as per screen size.
    - Call this function in the **play()** method.

```
showLife() {
        push();
        image(lifeImage, width / 2 - 130, height - player.positionY - 400, 20, 20);
        fill("white");
        rect(width / 2 - 100, height - player.positionY - 400, 185, 20);
        fill("#f50057");
        rect(width / 2 - 100, height - player.positionY - 400, player.life, 20);
          noStroke();
          pop();
        }
```

OUTPUT:



- Create **showFuelBar()** for fuel.
- Call this function in the **play()** method.

```
showFuelBar() {
  push();
  image(fuelImage, width / 2 - 130, height - player.positionY - 100, 20, 20);
  fill("white");
  rect(width / 2 - 100, height - player.positionY - 100, 185, 20);
  fill("#ffc400");
  rect(width / 2 - 100, height - player.positionY - 100, player.fuel, 20);
  noStroke();
  pop();
}
```

OUTPUT:

8. Reduce the fuel:
   - Check if the player's car is moving or not.
   - For that, create a property **this.playerMoving** 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 will work as a flag to give a sign **true** or **false**.
- False indicates the car is not moving and true will indicate the car is moving.
- Update **this.playerMoving** to true in **handlePlayerControls()**.

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

9. Modify **handleFuel(index)** to reduce Fuel when the car is moving and when it is empty. The game should get over.

```
handleFuel(index) {
  // Adding fuel
  cars[index - 1].overlap(fuels, function(collector, collected) {
    player.fuel = 185;
    //collected is the sprite in the group collectibles that triggered
    //the event
    collected.remove();
  });

  // Reducing Player car fuel
  if (player.fuel > 0 && this.playerMoving) {
    player.fuel -= 0.3;
  }

  if (player.fuel <= 0) {
    gameState = 2;
    this.gameOver();
  }
}
```

10. Create **gameOver()** function uses **swal()** to show Game Over pop-up.

```
gameOver() {
        swal({
            title: `Game Over`,
            text: "Oops you lost the race....!!!",
            imageUrl:

"https://cdn.shopify.com/s/files/1/1061/1924/products/Thumbs_Down_Sign_Emoji_Icon_ios10_grande.png",
            imageSize: "100x100",
            confirmButtonText: "Thanks For Playing"
          });
        }
```

11. Write a condition to move the car slowly even when the up arrow key is not pressed.
    - Use the **this.playerMoving** property.

```
if (this.playerMoving) {
  player.positionY += 5;
  player.update();
}
```

What's next?

In the next class, We will end this game by detecting collisions between cars and between cars and obstacles and reduce life.

EXTEND YOUR KNOWLEDGE:

1. Learn more about SweetAlert at: https://sweetalert.js.org/guides/