Topic	RESET BUTTON AND PLAYER CONTROLS	
Class Description	**Students will learn how to implement the reset button to reset the data in the database. Students will also create a leaderboard to show the score on the screen and add player controls.**	
Class	**C39**	
Class time	**45 mins**	
Goal	• Create a reset button to reset the values in the database. • Move the car left and right using arrow keys. • Create a leaderboard.	

Resources Required	• Teacher Resources
	  ○ VS Code Editor
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	• Student Resources
	  ○ VS Code Editor
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**WARM-UP** **Teacher-Led Activity** **Student-Led Activity** **WRAP-UP**	**5 Mins** **15 Mins** **20 Mins** **5 Mins**
WARM-UP SESSION - 5mins		
Teacher starts slideshow  **from slides 1 to 11** Refer to speaker notes and follow the instructions on each slide.		

Activity details	Solution/Guidelines
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Run the presentation from slide 1 to slide 4** **Following are the WARM-UP session deliverables:** • **Greet the student.** • **Revision of previous class activities.** • **Quizzes**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	**Answer**
Select the correct block of code to give the camera to the kangaroo on the x-axis.  A. `kangaroo.x=camera.positionX-270;` B. `kangaroo.x=camera.position.x-270;` C. `kangaroo.x=Camera.position.x-270;` D. `kangaroo.x=Camera.Position.x-270;`	**B**

Select the correct block of code to create a shrub sprite at camera position. A. ``` var shrub = createSprite (camera.positionX+500,330,40,10); ``` B. ``` var shrub = createSprite (camera.positionx+500,330,40,10); ``` C. ``` var shrub = createSprite (camera.position.y+500,330,40,10); ``` D. ``` var shrub = createSprite (camera.position.x+500,330,40,10); ```	**D**
Continue the WARM-UP session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 5 to slide 11 to set the problem statement. **Following are the WARM-UP session deliverables:** • Appreciate the student. • Make a reset button. • Create a Leaderboard.	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 15 mins	
Teacher Initiates Screen Share	
CHALLENGE • **Add Reset button and the text.**	

- **Create leaderboards.**

Teacher Action	Student Action
We have been resetting the database again and again in each class, any suggestions on what should we do to avoid that?	**ESR:** Varied
Today we will add a button that will reset the game and database.	
Which code can we use to create a button? Yes, similar to the play button we will use **p5.DOM.elements** to create a reset button.	**ESR:** Similar to the "play" button we created.
So let's get started. *The teacher downloads the code from Teacher Activity 1*	
We will start with a **game.js** file. Here we will create a **constructor()** function and in the function, we will create 2 elements one is the **resetButton** and next is the text for the reset button which will say "Reset Button". We can name it **resetTitle.**	
Can you help me with commands? *If a student is not able to answer, open form.js and revise with him again.*	ESR: Reset Title using **createElement()** function. Button can be created using **the createButton()** function.
These are an inbuilt function in **p5.DOM.js** library that can create an HTML element. We will pass the size of the text as an argument in this function here we will write "**h2**" which is the heading text size from HTML.	
We can keep its argument empty because we want to	

have an image attached to the button.	

Create a constructor() in game.js to declare properties for resetTitle and resetButton.

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

}
```

We have created properties for the reset button and the text but they won't be visible on the canvas yet. How do we display them? *The teacher can show form.js of the student unable to recollect.* We need to display them by setting their position and making them **HTML** elements. Everything we see on the screen has **HTML** in the backend. But these are handled by the libraries we are using, but in this case, we want to create the button and text in a more traditional way. In the **handleElement()** method of **game.js** we will write code to display the button and text. For the title, we will define the position of the text and the Text it will display. We are also adding this to the class so that we can apply styling to this button using CSS.	**ESR:** We can give .position() to each of them.

Create handleElements() to give position and text / image to resetTitle and reset Button in game.js.

```
handleElements() {
    form.hide();
    form.titleImg.position(40, 50);
    form.titleImg.class("gameTitleAfterEffect");


    this.resetTitle.html("Reset Game");
    this.resetTitle.class("resetText");
    this.resetTitle.position(width / 2 + 200, 40);

    this.resetButton.class("resetButton");
    this.resetButton.position(width / 2 + 230, 100);
```

For the button, we will just add this to the class and define its position.	
One part is done; now we need to do the same thing for the leader board text. That will show the player's score while the game is running. Can you help me with the code? First, we define the leader board text in the **constructor()** function of the game.js file. We will define text for **leaderboardTitle, leader1(player 1), and leader2(player2)**.	**ESR:** *Student can help with createElement()*
Create elements for leaderboardTitle,leader1 & leader2.	

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

   this.leadeboardTitle = createElement("h2");
   this.leader1 = createElement("h2");
   this.leader2 = createElement("h2");

 }
```

We created the text but now we need to display this in the **handleElements()** method.
We follow the same process we did for the reset button and text.

Assign position and text to leaderboardTitle, leader1 and leader2.

```
handleElements() {
  form.hide();
  form.titleImg.position(40, 50);
  form.titleImg.class("gameTitleAfterEffect");

  this.resetTitle.html("Reset Game");
  this.resetTitle.class("resetText");
  this.resetTitle.position(width / 2 + 200, 40);

  this.resetButton.class("resetButton");
  this.resetButton.position(width / 2 + 230, 100);

  this.leadeboardTitle.html("Leaderboard");
  this.leadeboardTitle.class("resetText");
  this.leadeboardTitle.position(width / 3 - 60, 40);

  this.leader1.class("leadersText");
  this.leader1.position(width / 3 - 50, 80);

  this.leader2.class("leadersText");
  this.leader2.position(width / 3 - 50, 130);
}
```

We have created the leader board text and also displayed it on the screen. But we have to add the logic to check which player is the leader.

The teacher will explain this code.

We have the player's rank updated in the database; here we use the player rank to display the leaderboards on the screen.

In the **showleaderBoard()** method we have 2 variables **leader1** and **leader2**, one more variable players which has information of the player taken from the database.

For that reason, we need to create a few more properties for **Player** class. We need properties to store rank and score.

If you notice in **Player.js,** inside **constructor(),** two more properties are added and assigned them to 0.

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

At the same time, we have to make sure we update these properties in the database using **addPlayer()** function and **update()** function.

UpdaddPlayer() is modified:

```
addPlayer() {
  var playerIndex = "players/player" + this.index;

  if (this.index === 1) {
    this.positionX = width / 2 - 100;
  } else {
    this.positionX = width / 2 + 100;
  }

  database.ref(playerIndex).set({
    name: this.name,
    positionX: this.positionX,
    positionY: this.positionY,
    rank: this.rank,
    score: this.score
  });
}
```

update() is modified.

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

  });
}
```

We can use these properties to show **leaderboard().**

The first condition is to check if our first player has rank 1. We will set the rank, name, and score of player 1 to the leader 1 variable.

And leader 2 will have the details of player 2. Here ** ** is an HTML tag that is used to add 4 spaces before the text.

Instead of using a for-in loop to traverse through **allPlayers**, we will learn another method, the **Object.values().** This method returns an array of a given object's own enumerable property values, in the same order as that provided by a for-in loop.

Create showLeaderboard() to display Player's rank, name & score.

```
showLeaderboard() {
   var leader1, leader2;
   var players = Object.values(allPlayers);
   if (
     (players[0].rank === 0 && players[1].rank === 0) ||
     players[0].rank === 1
   ) {
     //      This tag is used for displaying four spaces.
     leader1 =
       players[0].rank +
       " " +
       players[0].name +
       " " +
       players[0].score;

     leader2 =
       players[1].rank +
       " " +
       players[1].name +
       " " +
       players[1].score;
   }
```

But, if player 2 is having rank one then we need to set

player 2 as leader 1.

At last, we will pass the **leader1** and **leader2** variables in the HTML so that they can be displayed on the canvas.

Pass leader1 and leader2 as HTML elements.

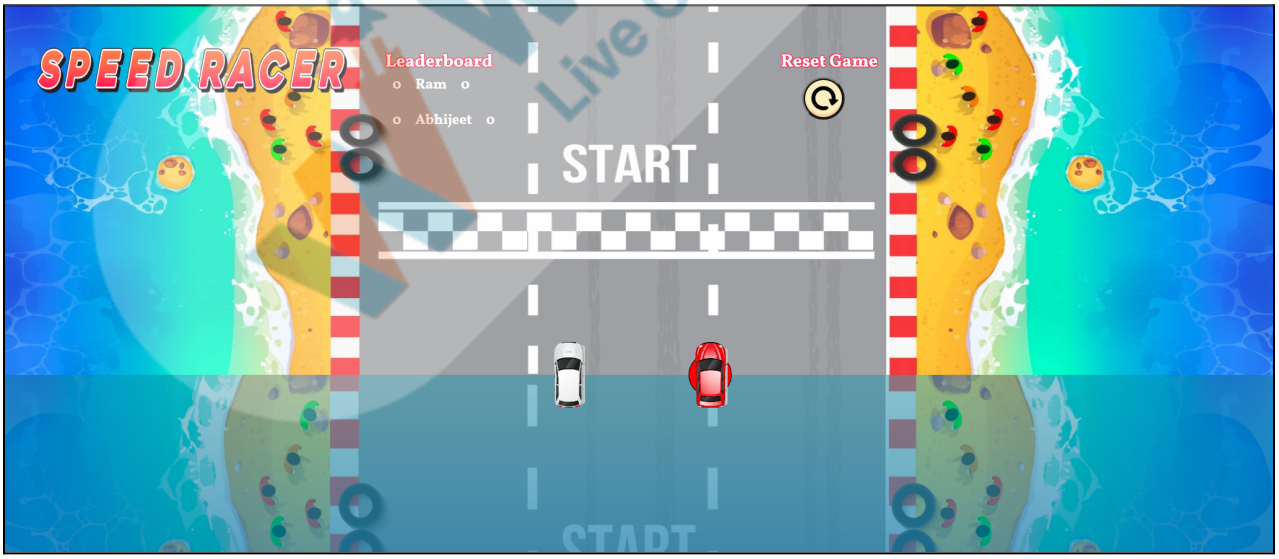
```
  if (players[1].rank === 1) {
    leader1 =
      players[1].rank +
      " " +
      players[1].name +
      " " +
      players[1].score;

    leader2 =
      players[0].rank +
      " " +
      players[0].name +
      " " +
      players[0].score;
  }
  this.leader1.html(leader1);
  this.leader2.html(leader2);
}
```

Call this.showLeaderboard() inside the play() method.

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

  Player.getPlayersInfo();

  if (allPlayers !== undefined) {
    image(track, 0, -height * 5, width, height * 6);

    this.showLeaderboard();

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

Now we can run the game and see the output.

Note: We have cropped the output to show the reset button and leader board text.



Isn't it looking great?

ESR: Varied

What should we do next? We will add functionality to the reset button to rest the database. And you will also code to move the car sideways using arrow keys.	

Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Fullscreen.**

ACTIVITY

- **Define the function to reset the game.**
- **Add player controls.**

Teacher starts slideshow  **:Slide 12 to 13**
Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
The teacher guides the student to open Student Activity 1. Download the zip folder, unzip it and save it as C39. *Make sure to add the database information in **index.html.*** *During the class, if a student is changing any position of any element or object in the code/ ask them to write it somewhere to use in codes of the next classes.* *In case any x or y positions were changed by the student in the code in the previous class, make those changes to fit into his screen size.*	*Alternatively, the student can clone Student Activity 1.*

We have created the button for the reset, but we have not defined the functionality of the button. What should happen when the user presses the reset button? Great! The reset button will work with the mouse press. For that, we will use the **mousePressed()** function of the **p5.js** and then we will bind the functionality of this button using the arrow function. *Ask the student to visit form.js, and revise how mousePressed() was used with the Play button.* What are all things we change in the database before running the code every time? Yes, the button will set the player count, game state as 0, and the player's fields to be removed. We will use .**set()** to overwrite values of database fields with 0 for **carsatEnd, playerCount and gameState**.We will also empty the **player's** information. Lastly, reload the window using **window.location.reload()** function We will do all these inside a separate method **handleResetButton()** in **game.js**.	**ESR:** Reset the player position and score in the database. ESR: We change playerCount to 0; gameState to 0 and remove all players.
Create handleResetButton() to reset all fields in the database and reload the game.	

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

Call the function handleResetButton() inside play()

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

  Player.getPlayersInfo();
```

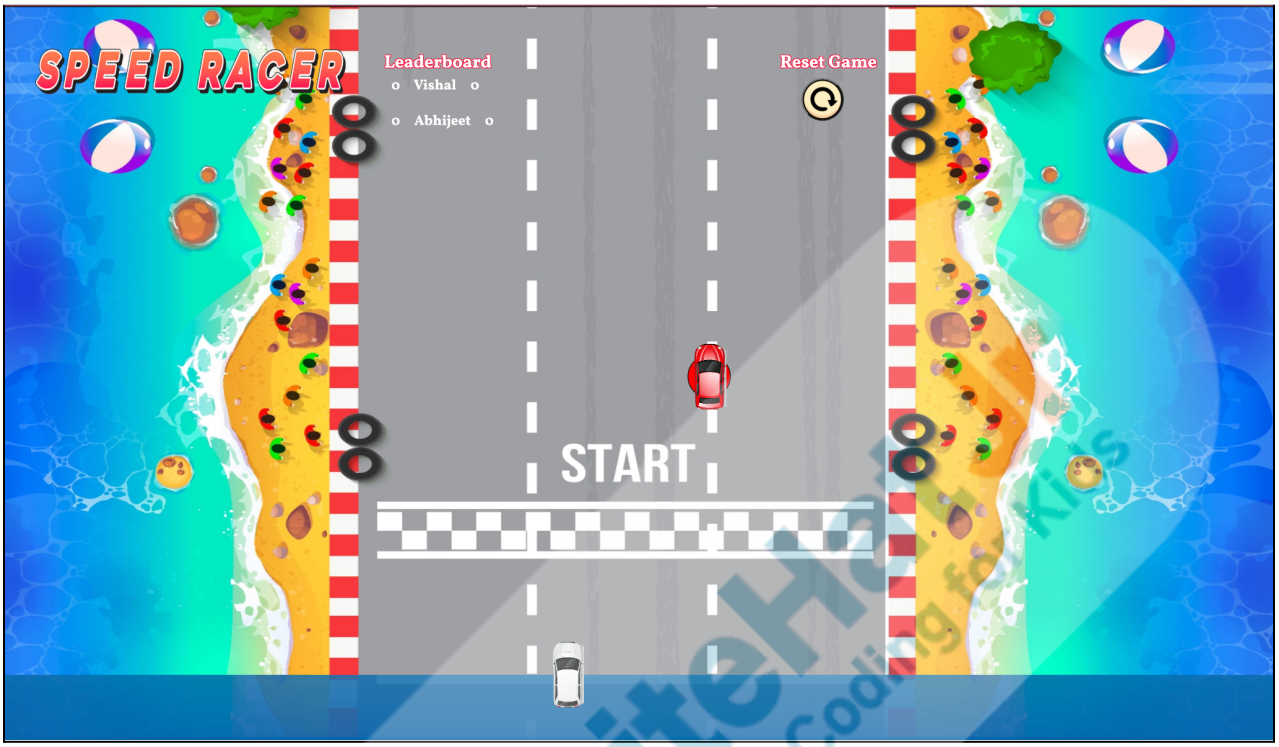
We have made the reset button work. If you remember our car only moves only in the upwards direction and we have no control to move it left and right. So now we will do that. Can you tell me how can we add control to the game? Very good! The car moves forward in the y-direction when the user presses the up arrow button. To move the car in the left and right direction, we will	*The student runs the code to test the reset button.* **ESR:** Changing x position with arrow keys.

change the x position value with the left and right arrow keys. We also don't want the player to go off track; to prevent that we will write the condition. In method **handeplayerControls()** in game.js add these controls. To move left we will use the **keyIsDown()** function and it will look for the **LEFT_ARROW** pressed by the user. We will add the condition using the and operator that player x position can be given greater than **width/3 -50**. ***(Guide the student to check using console.log and change accordingly.)*** This means if the user goes to the end of the track in the left direction, it will not move any further. Similar things we will implement for the right side also. To move the player on the left side, we need to subtract from its position, and to move towards the right we need to increment the x-position.	
Modify handlePlayerControls() to add conditions to move the car with Left & Right Arrow keys.	

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

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

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

We need to do one more important thing, which is to update the player's position in the database using the **player.update()** method.

Well done! We have implemented the reset button, leaderboards, and player controls.
Run the program and see if everything is working properly.



You can try to reset the game as well. Here our car can now move in left and right as well; we are collecting the fuel and coins. But our fuel bar is still not decreasing, which we will implement in the next class.	

Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 Mins	
Teacher starts slideshow  **from slide 14 to slide 25**	
Activity details	**Solution/Guidelines**
Run the presentation from slide 14 to slide 25	

Following are the WRAP-UP session deliverables: • **Appreciate the student.** • **Revise the current class activities.** • **Discuss the quizzes.**	Discuss with the student the current class activities and Student will ask doubts related to the activities.
Quiz time - Click on in-class quiz	
Question	**Answer**
Select the correct code snippet to create a reset button. A. this.reset = createReset('Reset'); B. this.reset = createElement('Reset'); C. this.reset = createInput('Reset'); D. this.reset = createButton('Reset');	**D**

What function is this piece of code performing?
 	
this.reset.mousePressed(() =>{
database.ref('/').set({
playerCount: 0
gameState:0
})
})
A. to update the playerCount and gameState in the database when the Reset button is pressed
B. to give each player the freedom to set the gameState and playerCount to whatever they want
C. to change the gameState and playerCount by pressing the mouse anywhere on the screen
D. to convert the Play button in the login form into a Reset button
	A

Which of the following methods is created to show the leaderboard using HTML elements?	**D**

A. handleElements() B. constructor() C. handleplayercontrol() D. showLeaderBoard()	
End the quiz panel	
FEEDBACK • **Appreciate the student for their efforts in the class.** • **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**	
Teacher Action	**Student Action**
You get Hats off for your excellent work! In the next class, we will make the game more fun and challenging by adding a few rewards and obstacles to the game.	*Make sure you have given at least 2 Hats Off during the class for:* 
*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.*** **Project Overview** **KANGAROO IN THE JUNGLE 2** **Goal of the Project:**	**Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.** *Students engage with the teacher over the project.*

In class 39, you learned how to end the game and reset the game. In this project, you have to practice and apply what you have learned in the class. In this project, the kangaroo in the jungle game includes code to reset the game when the reset button is pressed. Additionally, try to add a win condition to the game. ** This is a continuation of Project 38. Make sure you complete that project before attempting this one. ** **Story:** You have created the basic game for the kangaroo who is hungry and is eating all the tasty shrubs and plants in the jungle. To make the game interesting, add a WIN condition in the game when the score becomes 5 or more. Plus, display a victory message when the player score becomes 5. Also, create a functionality to reset the game when the reset button is pressed to play the game again. Bye Bye!	
Teacher ends slideshow 	
Teacher Clicks ✖ End Class	
ADDITIONAL ACTIVITIES	
Additional Activities *Encourage the student to write reflection notes in their reflection journal using Markdown.*	*The student uses the Markdown editor to write their reflections in a reflection journal.*

Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C39RV_SpeedRacer_TeacherActivity
Teacher Activity 2	Teacher Reference Code	https://github.com/pro-whitehatjr/C39RV_SpeedRacer_ReferenceCode
Student Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C39RV_SpeedRacer_StudentActivity
Teacher Reference	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C39_V3_lit_WithCues.html
Teacher Reference	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/376847cd-25dd-44e8-8136-3bb033fa56d3.pdf
Project Solution	Kangaroo in Jungle-2	https://github.com/pro-whitehatjr/Project-C39-V3