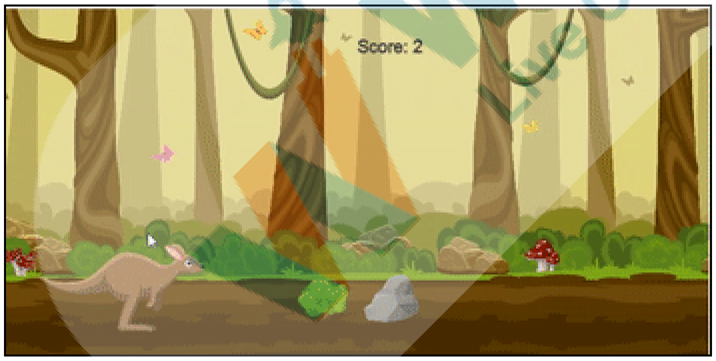
| Topic | REWARDS AND OBSTACLES | |
|---|---|---|
| Class Description | **The student learns to create sprites and add them to groups on a real-time basis. The student will also create random fuel & Powercoins as rewards for players to collect and add obstacles to make the game challenging.** | |
| Class | **C40** | |
| Class time | **45 mins** | |
| Goal | • Create fuels, coins, and obstacles to appear randomly in the game.<br>• Create a function to collect coins and fuels for players. | |
| Resources Required | • Teacher Resources<br>○ VS Code Editor<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br><br>• Student Resources<br>○ VS Code Editor<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen | |
| Class structure | **WARM-UP**<br>**Teacher-led Activity**<br>**Student-led Activity**<br>**WRAP-UP** | **5 mins**<br>**15 min**<br>**20 min**<br>**5 mins** |

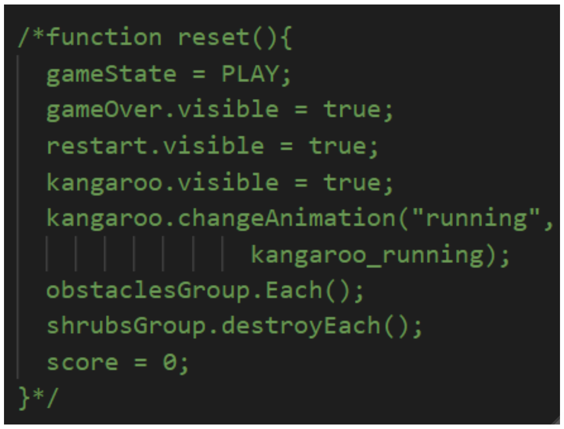
**WARM-UP SESSION - 5 mins**

**Teacher starts slideshow from slides 1 to slide 10**
Refer to speaker notes and follow the instructions on each slide.

| Teacher Action | Student Action |
|---|---|

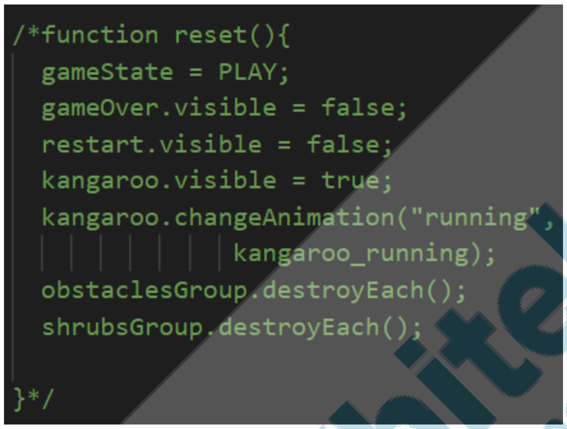
| | |
|---|---|
| *How are you doing? Are you excited to learn something new?*<br><br>***Run the presentation, slide 1 to slide 3.***<br><br>**Following are the WARM-UP session deliverables:**<br>● Connecting student to the previous class.<br>● WARM-UP Quiz Session | **ESR**: Thanks, yes, I am excited about it.<br><br>Click on the slide show tab and present the slides. |
| **QnA Session** | |
| **Question** | **Answer** |
| Select the correct option to the **gameOver** sprite invisible?<br>A. gameOver.visible = true;<br>B. gameOver.visibility = true;<br>C. gameOver.visible = false;<br>D. gameOver.visibility = false; | **C** |
| Which of the following code blocks will **reset** the game as shown in the following output?<br><br> | **D** |

A.

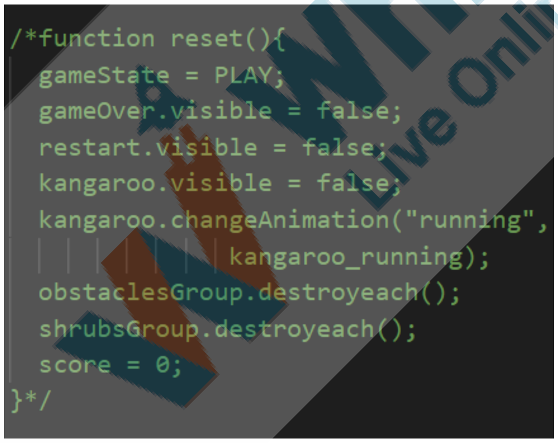


```
/*function reset(){
  gameState = PLAY;
  gameOver.visible = true;
  restart.visible = true;
  kangaroo.visible = true;
  kangaroo.changeAnimation("running",
                kangaroo_running);
  obstaclesGroup.Each();
  shrubsGroup.destroyEach();
  score = 0;
}*/
```

B.

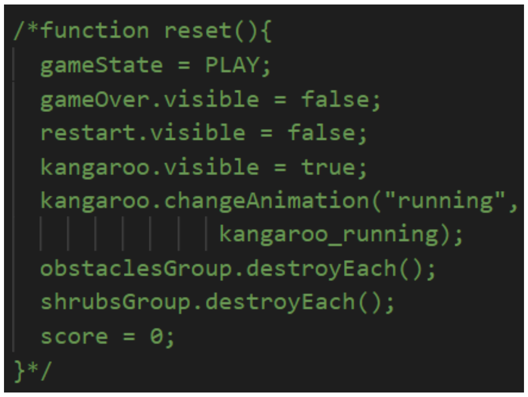


```
/*function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;
  kangaroo.visible = true;
  kangaroo.changeAnimation("running",
                kangaroo_running);
  obstaclesGroup.destroyEach();
  shrubsGroup.destroyEach();

}*/
```



```
/*function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;
  kangaroo.visible = false;
  kangaroo.changeAnimation("running",
                kangaroo_running);
  obstaclesGroup.destroyeach();
  shrubsGroup.destroyeach();
  score = 0;
}*/
```

C.

D. 

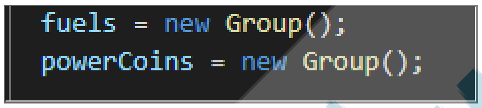
```
/*function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;
  kangaroo.visible = true;
  kangaroo.changeAnimation("running",
                           kangaroo_running);
  obstaclesGroup.destroyEach();
  shrubsGroup.destroyEach();
  score = 0;
}*/
```

| **Continue the WARM-UP session** | |
|---|---|
| **Teacher Action** | **Student Action** |
| ***Run the presentation from slide 4 to slide 10 and set the problem statement.*** | Narrate the slides by using hand gestures and voice modulation methods to bring in more student interest. |
| **Teacher ends slideshow** | |
| **TEACHER-LED ACTIVITY - 15 mins** | |
| **Teacher Initiates Screen Share** | |
| **<u>CHALLENGE</u>**<br>• **Create multiple sprites at random positions for fuel and Powercoins.** | |
| **Teacher Action** | **Student Action** |
| Today, we will make our game more fun and exciting by adding a few rewards for our racers.<br><br>You must have played many games so far. What kind of rewards do you think we can give to our racers? | **ESR**: Varied. |

| | |
|---|---|
| In our game, we will add gold coins for the player to make gain points and also fuel tanks for refilling their cars.<br><br>*The teacher downloads [Teacher Activity 1](#) and opens the code in VSC.*<br><br>Let's revise the code first. Can you explain to me the code in **sketch.js?**<br><br>*Make sure to replace the database code with an SDK from your own database.*<br><br>Correct, as you notice, the images for fuel and Powercoins are also loaded into respective global variables.<br>What is next?<br><br>*The teacher can help the student understand the code in* ***sketch.js*** *which was done in the previous class and clear questions if any.*<br><br>Fantastic!! | **ESR**:<br>In **sketch.js** we use the function **preload()** to add in the variables.<br><br>**ESR:**<br>In function **setup()** we are creating game object and calling **.getState()**, and the **.start()** function from **game.js.**<br><br>In function **draw(),** we are checking if **playerCount** is **2** we update **gameState** to **1;** and if **gamestate** is **1** we call **game.play().** |

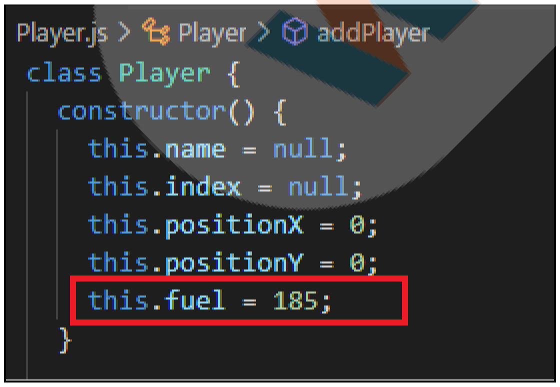
| | |
|---|---|
| Now we have 3 objects, **form, player & game**. Where do you think we should add fuel and the Powercoins sprites?<br><br>As you can see, images for Powercoins and fuel tanks are added in the **preload()** function of **sketch.js.**<br>Our game execution code is in **game.js**; we want to display fuel and Powercoins while the game is being played; hence, we will create these sprites in the **start()** function of **game.js.** We will achieve this using a **Group**.<br><br>We also used groups while making the Trex game. Can you help me with syntax?<br><br>*The teacher can remind the student if he/she is not able to recollect.*<br>Now, lets us create two groups named **fuels** and **powerCoins** in the **start()** method of **game.js.** | **ESR**: Varied<br><br>**ESR**:<br>**groupname = new Group()** |

Create a new group as **fuels** and **powerCoins** inside **start()** of **game.js**.



```
fuels = new Group();
powerCoins = new Group();
```

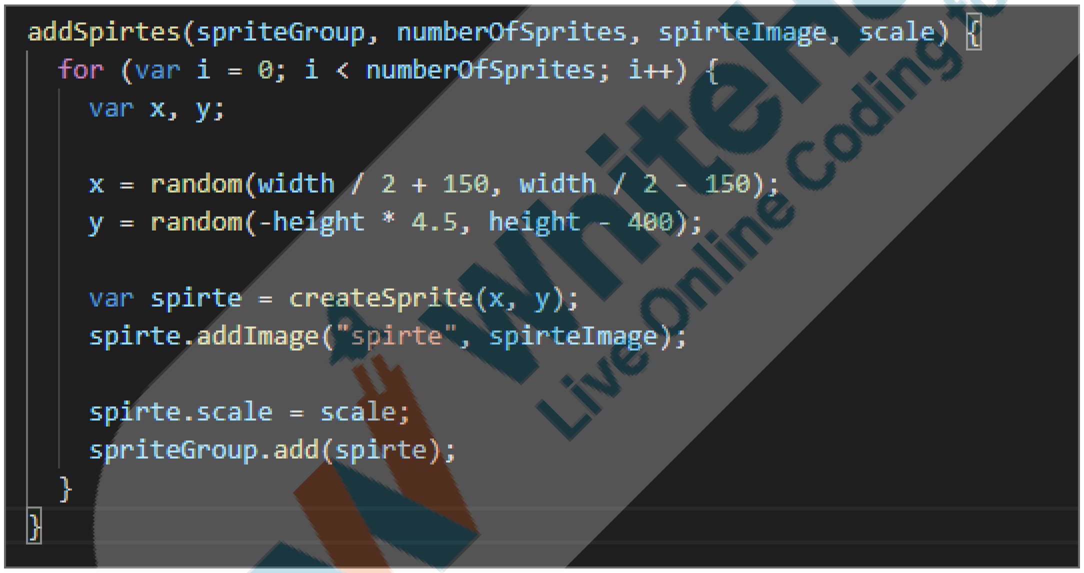
Also, add **this.fuel = 185** in **constructor()** of **player.js**. We are creating this property for players to assign initial fuel. In the next class, we will see how we will use this property to reduce the fuel of each player's car as they move in the game.



```
Player.js > Player > addPlayer
class Player {
  constructor() {
    this.name = null;
    this.index = null;
    this.positionX = 0;
    this.positionY = 0;
    this.fuel = 185;
  }
```

| | |
|---|---|
| We will also create a separate function to create sprites and to add them to the respective group.<br><br>What is common in both sprites?<br><br>They have images and should appear randomly.<br><br>We will create a common function for creating both kinds of sprites. We will pass parameters like **spriteGroupname**, **number of sprites required, image, size of image(scale)** to create Sprites and have them appear randomly while adding them to their respective group. | *The student listens and observes*<br><br>**ESR**: Varied. |

Create a method, **addSprites()** in **game.js**.

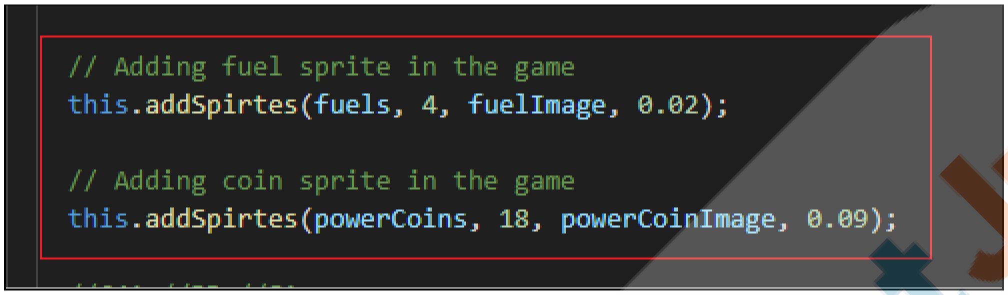


```
addSpirtes(spriteGroup, numberOfSprites, spirteImage, scale) {
  for (var i = 0; i < numberOfSprites; i++) {
    var x, y;

    x = random(width / 2 + 150, width / 2 - 150);
    y = random(-height * 4.5, height - 400);

    var spirte = createSprite(x, y);
    spirte.addImage("spirte", spirteImage);

    spirte.scale = scale;
    spriteGroup.add(spirte);
  }
}
```

| | |
|---|---|
| *The teacher writes code, and you can ask the student to explain; all the concepts are covered previously so that they feel more confident.*<br><br>*The teacher can guide the student if required to explain the code.* | **ESR:**<br>We are using a **'for'** loop to create multiple sprites; each sprite is getting a random x & y position. We then **addImage** to sprite and add them in respective groups. |

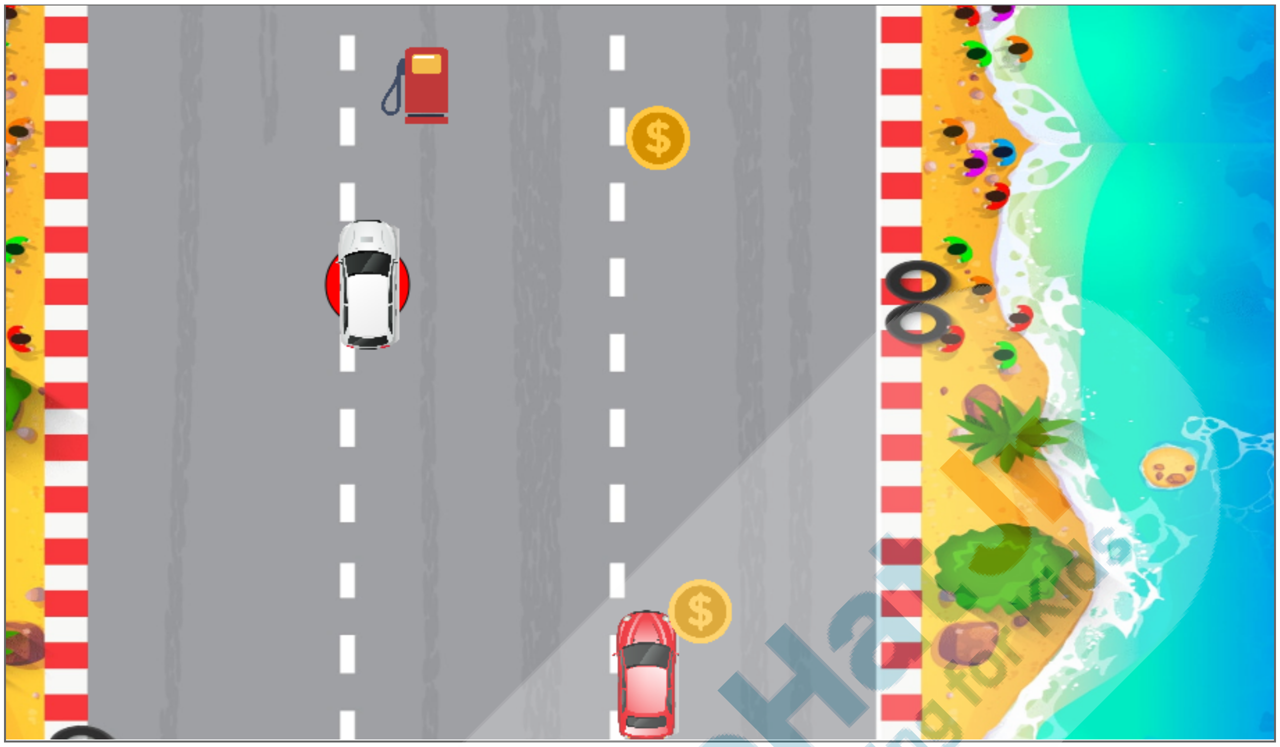
| | |
|---|---|
| Awesome! Where should we call this function?<br><br>In the **start()** method, right after we created **group().** | **ESR**: Varied.<br><br>*The student observes.* |

While calling **addSprites()** in **start()** we are passing **grouname, numberofSprites, Imagename, scaleofimage.**



```
// Adding fuel sprite in the game
this.addSpirtes(fuels, 4, fuelImage, 0.02);

// Adding coin sprite in the game
this.addSpirtes(powerCoins, 18, powerCoinImage, 0.09);
```

| | |
|---|---|
| Let's run the code and test it.<br><br>*You will see coins and fuel appearing randomly on the track for both players.* | |



| | |
|---|---|
| As you can see, powerCoins and fuel tanks are appearing randomly on the track.<br><br>What should we do next?<br><br>Yes, now let's write code for when a player's car touches the fuel tank or powerCoins we should make that particular sprite disappear.<br><br>We will also make the game more challenging by adding obstacles.<br><br>With this guide in mind, why don't you start with the coding exercise? I will be guiding you in the exercise. | **ESR**: Players should be able to collect these powerCoins and fuel tanks. |

**Teacher Stops Screen Share**

**STUDENT-LED ACTIVITY - 20 mins**

| | | |
|---|---|---|
| | Now it's your turn. Please share your screen with me. | |

- **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**

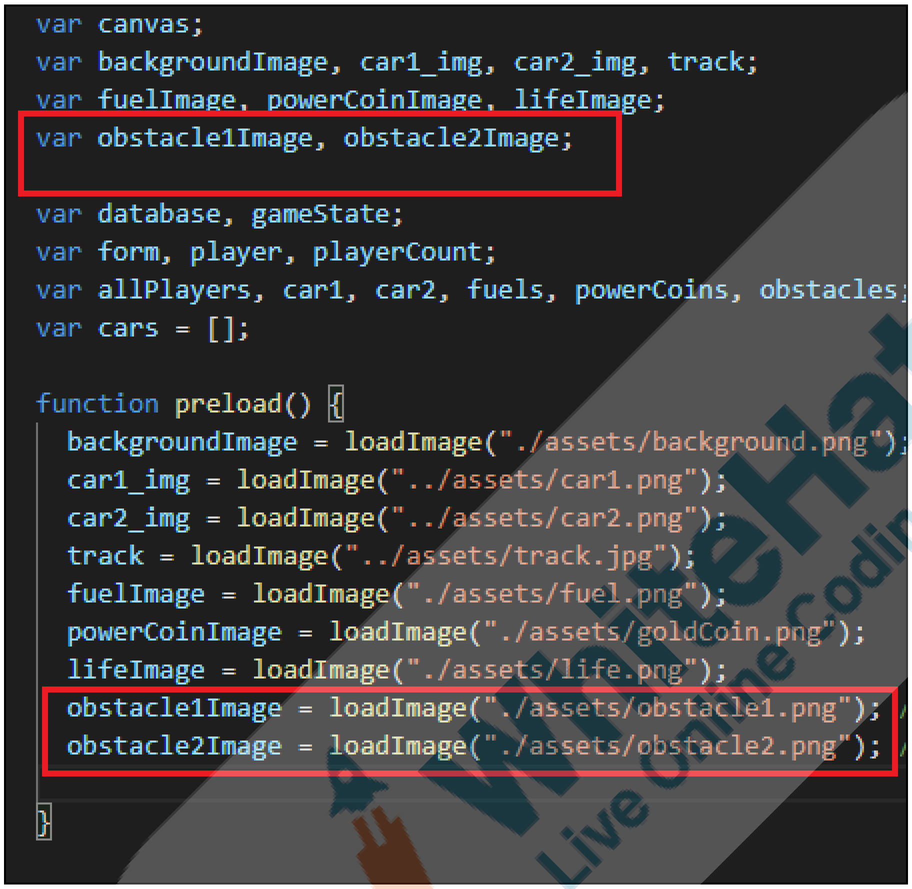
- **Create Obstacles using Group and addSprites()**
- **Use overlap() to collect powerCoins and fuel tank.**

**Teacher starts slideshow for slide 11 to slide 15**

Refer to speaker notes and follow the instructions on each slide.

| Teacher Action | Student Action |
|---|---|
| *Guide the student to open Student Activity 1 and download it as a zip folder, unzip it and save it as **C40**.*<br><br>*Make sure to add the database information in **index.html.***<br><br>*During the class, if the student is changing any position of any element or object in the code/ ask them to write it somewhere to use in codes of next classes.*<br>*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.* |
| Before we grab fuel and coins, let us make our game challenging by adding a few obstacles. On colliding with obstacles, we will reduce the life bar gradually.<br><br>To start, we will declare the global variables for preloading images. Where should we do that? | **ESR**: **sketch.js** |

Create variables and preload images in **sketch.js**
Create 3 global variables **obstacle1Image** and **obstacle2Image** for loading images and **obstacles** for creating a group.
Use **preload()** to load images for **obstacle1Image** and **obstacle2Image.**

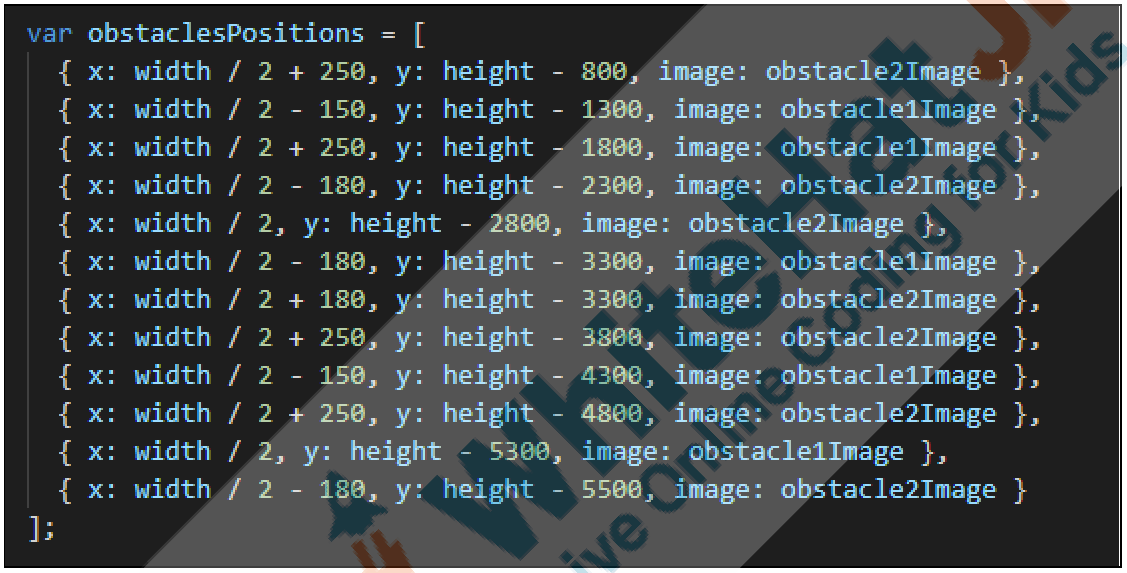


```
var canvas;
var backgroundImage, car1_img, car2_img, track;
var fuelImage, powerCoinImage, lifeImage;
var obstacle1Image, obstacle2Image;

var database, gameState;
var form, player, playerCount;
var allPlayers, car1, car2, fuels, powerCoins, obstacles;
var cars = [];

function preload() {
  backgroundImage = loadImage("./assets/background.png");
  car1_img = loadImage("../assets/car1.png");
  car2_img = loadImage("../assets/car2.png");
  track = loadImage("../assets/track.jpg");
  fuelImage = loadImage("./assets/fuel.png");
  powerCoinImage = loadImage("./assets/goldCoin.png");
  lifeImage = loadImage("./assets/life.png");
  obstacle1Image = loadImage("./assets/obstacle1.png");
  obstacle2Image = loadImage("./assets/obstacle2.png");

}
```

| | |
|---|---|
| Similar to **powerCoins** and **fuel**, we will create a group for obstacles and add various obstacles groups in that. | *The student writes code with the teacher's direction.* |
| Open **game.js** and create a new group in the **start()** method. | |

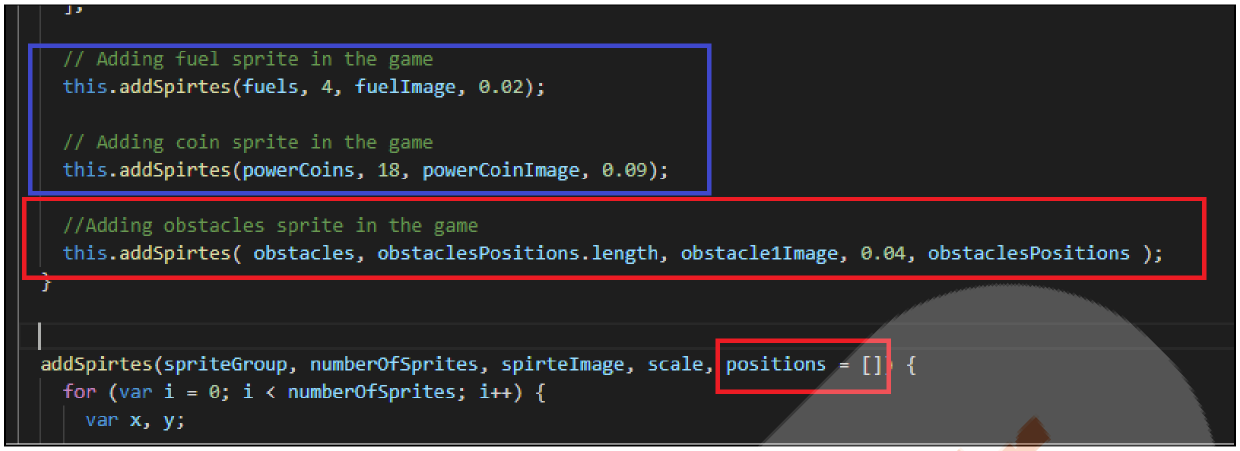
Create a group in **start() of Game.js**.

```
obstacles = new Group();
```

| | |
|---|---|
| We can use the same **addSprites()** functions to create multiple obstacles and add them to the group.<br><br>However, in doing so, one of the players might get more obstacles than others. In order to prevent this, we will pre-define the position of the obstacles in the game.<br><br>If you notice in the code, for **start(),** an array variable is created to hold positions of obstacles. | |
| ***Note: This code is given in Student's boilerplate.**<br><br> | |
| With lots of trials and errors, we have already placed each obstacle at fixed positions. After the class, you can change the value of x & y to experiment with their positions.<br><br>Now, we can use our **addSprite()** method to create sprites at specific positions for obstacles.<br><br>Can you tell me what parameters we need to pass while creating sprites?<br><br>Correct, but for obstacles, we also need to pass their positions as we have already defined their position in an | *The student listens and asks if there is any doubt.*<br><br>**ESR**:<br>**SpriteGroupName, numberofSprites, SpriteImage,** and **scaleofimage** |

```
var obstaclesPositions = [
  { x: width / 2 + 250, y: height - 800, image: obstacle2Image },
  { x: width / 2 - 150, y: height - 1300, image: obstacle1Image },
  { x: width / 2 + 250, y: height - 1800, image: obstacle1Image },
  { x: width / 2 - 180, y: height - 2300, image: obstacle2Image },
  { x: width / 2, y: height - 2800, image: obstacle2Image },
  { x: width / 2 - 180, y: height - 3300, image: obstacle1Image },
  { x: width / 2 + 180, y: height - 3300, image: obstacle2Image },
  { x: width / 2 + 250, y: height - 3800, image: obstacle2Image },
  { x: width / 2 - 150, y: height - 4300, image: obstacle1Image },
  { x: width / 2 + 250, y: height - 4800, image: obstacle2Image },
  { x: width / 2, y: height - 5300, image: obstacle1Image },
  { x: width / 2 - 180, y: height - 5500, image: obstacle2Image }
];
```

| | |
|---|---|
| array. For fuel and powerCoins we wanted a random position, hence we assigned it later.<br><br>Hence, we will have to modify **addSprites()** to accept the positions of obstacles. The number of Sprites we can get by checking the length of our array **obstaclePositions**.<br><br>Let's add one more parameter in **addSprites(), position = [ ].**<br>An empty array to accept **obstaclePositions.**<br><br>With this addition, do we need to make any change to create fuel and coins? ***(Highlighted within the blue rectangle in the image below.)***<br><br>No, we do not need to change, as the fifth parameter is assigned with an empty array in **addSprites()**. It will capture that empty value for fuel & coins.<br><br>When parameters are defined in the function, we can ignore passing values for those predefined parameters while calling the function. | *The student calls method* ***addSprite()*** *in* ***start()*** *and passes the parameters as obstacles,* ***obstaclePositions.length, obstacleImage, scale, obstacleposition.***<br><br>*The student adds* ***position = []*** *in* ***addSprites().***<br><br>**ESR**: Varied. |
| Use **addSprites()** for creating obstacles and add them into the group;<br><br>Modify **addSprites()** to add position parameter | |

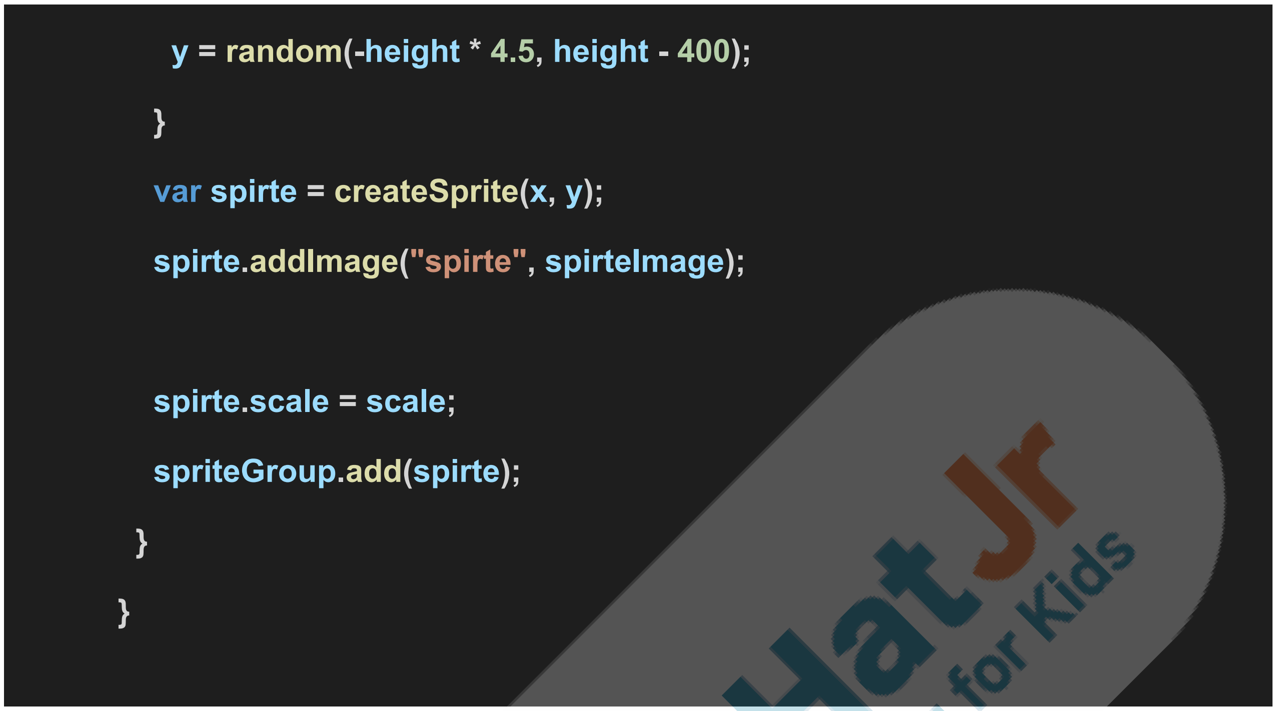


```
    });

    // Adding fuel sprite in the game
    this.addSpirtes(fuels, 4, fuelImage, 0.02);

    // Adding coin sprite in the game
    this.addSpirtes(powerCoins, 18, powerCoinImage, 0.09);

    //Adding obstacles sprite in the game
    this.addSpirtes( obstacles, obstaclesPositions.length, obstacle1Image, 0.04, obstaclesPositions );
  }

  addSpirtes(spriteGroup, numberOfSprites, spirteImage, scale, positions = []) {
    for (var i = 0; i < numberOfSprites; i++) {
      var x, y;
```

*Note: The code highlighted in the blue box is covered as a part of Teacher Activity. The student will add the code highlighted in the red box.*

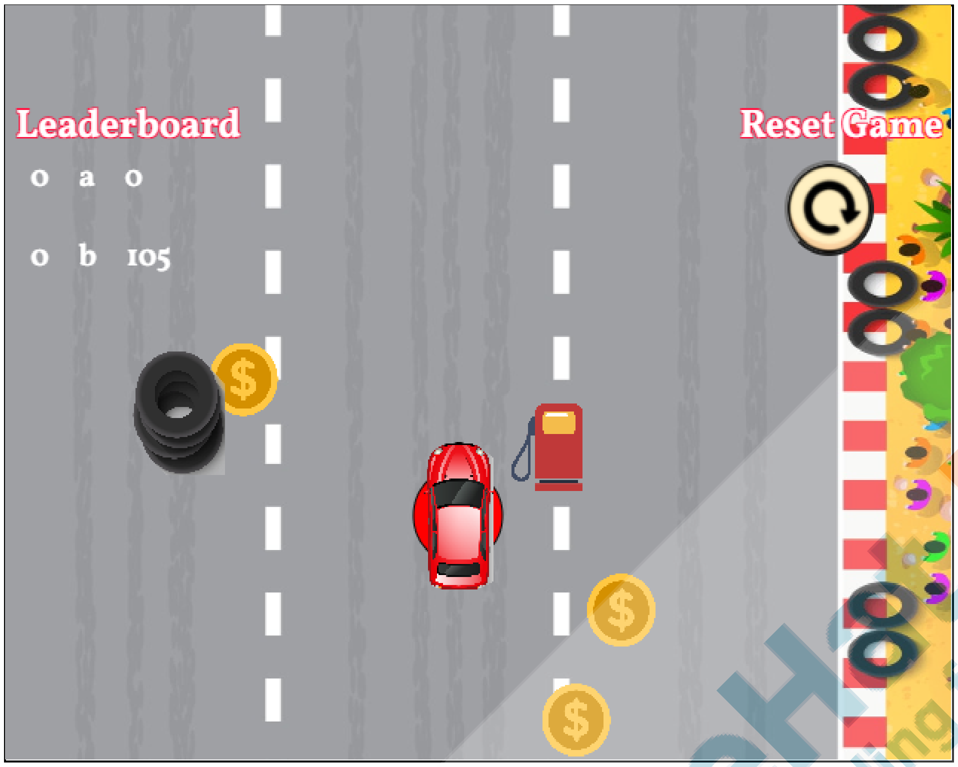
| | |
|---|---|
| Great!<br><br>Modify the code in **addSpites()** to show obstacles.<br><br>Check the length of the position and will assign images and add them to the group.<br><br>When there is no position, it will execute code for coin & fuel to give a random position. | *The student writes the code.* |

```
addSpirtes(spriteGroup, numberOfSprites, spirteImage, scale, positions = []) {
      for (var i = 0; i < numberOfSprites; i++) {
        var x, y;
        if (positions.length > 0) {
          x = positions[i].x;
          y = positions[i].y;
          spirteImage = positions[i].image;
        } else {
          x = random(width / 2 + 150, width / 2 - 150);
```



```
            y = random(-height * 4.5, height - 400);
          }
          var spirte = createSprite(x, y);
          spirte.addImage("spirte", spirteImage);


          spirte.scale = scale;
          spriteGroup.add(spirte);
        }
      }
```

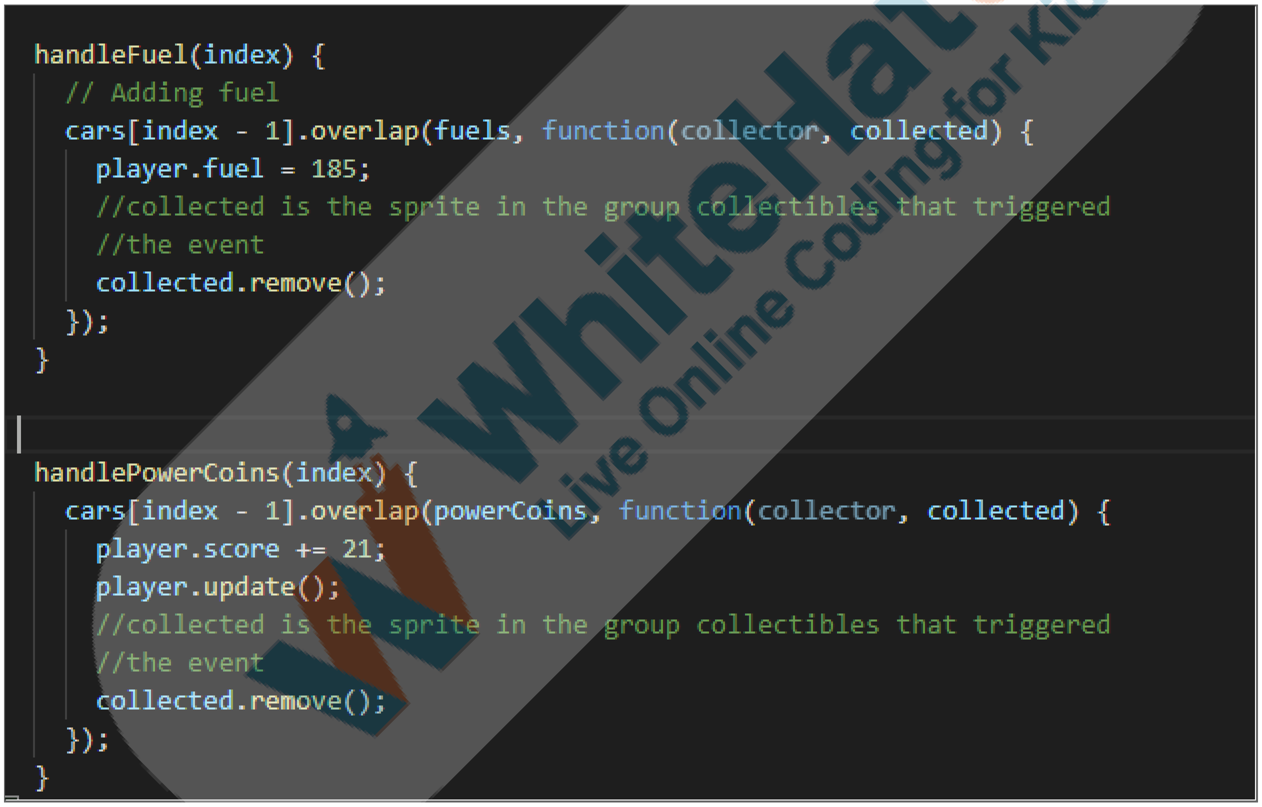
| | |
|---|---|
| Let us check the output. | *The student runs the code to see the obstacles on the track.* |



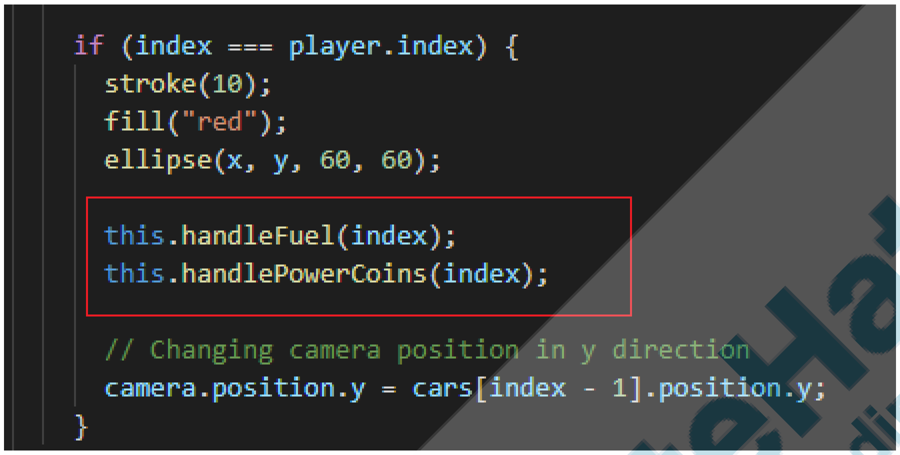
| | |
|---|---|
| We can now see the obstacles placed evenly on the track for both the players.<br><br>Now to the final part of today's class, we need to remove that sprite from the group once the car touches the coin or a fuel tank.<br><br>Any suggestions on how we can check if the car is touching any of the sprites?<br><br>Yes, we can use any of those, but we will use the **overlap()** function.<br><br>The advantage of using the **overlap()** function over **isTouching()** or **collide()** function is that if the target is a group of sprites, the function will traverse to check for each sprite for overlapping. | **ESR**: We can use the **isTouching()** or **collide()** function. |

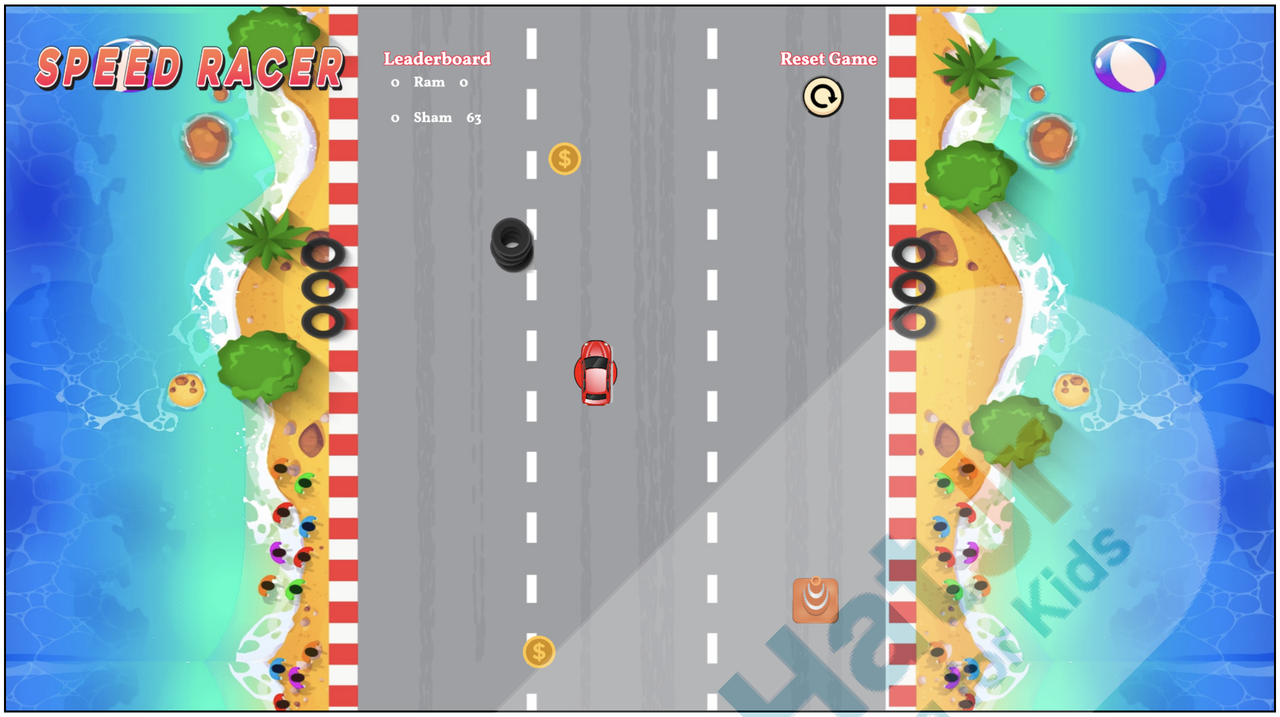
| | |
|---|---|
| Let us open the link to understand the **overlap()** function a bit more. | *The student opens Student Activity 2.* |
| *The teacher guides the student to use the code to check an overlap between **SpriteGroup** and **car**, and remove the sprite from the group.*<br><br>Create two functions, **handleFuel()** & **handlePowerCoin()** functions. | *The student writes the code.* |

Create **handleFuel()** and **handlePowerCoins** functions in **game.js.**



```
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();
  });
}


handlePowerCoins(index) {
  cars[index - 1].overlap(powerCoins, function(collector, collected) {
    player.score += 21;
    player.update();
    //collected is the sprite in the group collectibles that triggered
    //the event
    collected.remove();
  });
}
```

| | |
|---|---|
| Superb!<br><br>So where should we call these functions?<br><br>Correct, Let us do that. We need to know which player's car is touching the coin or fuel tank. So we need to pass index to these functions. | <br><br>**ESR:** In the **play()** function.<br><br><br>*The student writes the code.* |
| <br>`if (index === player.index) {`<br>`  stroke(10);`<br>`  fill("red");`<br>`  ellipse(x, y, 60, 60);`<br><br>`  this.handleFuel(index);`<br>`  this.handlePowerCoins(index);`<br><br>`  // Changing camera position in y direction`<br>`  camera.position.y = cars[index - 1].position.y;`<br>`}` | |
| Let us run the code and check if fuel and coin are being removed from the track once the player car touches it.<br><br>First, reset the values of **gameState** and **playerCount** in the database to **0** and delete players. | |

| | |
|---|---|
|  SPEED RACER<br>Leaderboard<br>Ram 0<br>Sham 63<br>Reset Game | |
| As you can see, the powerCoin or the fuel tank disappears from the track once the car touches it.<br><br>Isn't our game look fun with rewards and obstacles?<br><br>For now, we will stop here for today. | **ESR:** Varied |
| **Teacher Guides Student to Stop Screen Share** | |
| **WRAP-UP SESSION - 5 Mins** | |
| **Teacher starts slideshow from slides 16 to slide 25.**<br>Refer to speaker notes and follow the instructions on each slide. | |
| **Teacher Action** | **Student Action** |
| ***Run the presentation from slide 16 to slide 25.***<br><br>**Following are the WRAP-UP session deliverables:**<br>● **Revise the concepts** | Guide the student to |

| | |
|---|---|
| ● **Wrap Up Quiz**<br>● **Explain the facts and trivias**<br>● **Project for the day**<br>● **Next class challenge** | develop the project and share it with us. |
| **Quiz time - Click on in-class quiz** | |
| **Question** | **Answer** |
| What does the overlap() function of p5.play do?<br><br>A. Removes the overlapped sprites<br>B. Destroys the sprite on overlap with another sprite<br>C. Overlaps the sprite on touching another sprite<br>D. Calls a function if the sprite is overlapping another sprite or a group | **D** |
| Which of the following is incorrect for adding the obstacles?<br><br>A. We used the addSprites() to add obstacles.<br>B. We passed the position of the obstacles as the fifth parameter to addSprites().<br>C. The position parameter is an array containing the obstacle positions.<br>D. None of the above. | **D** |
| What is the correct format for creating a group?<br><br>A. groupname = new Group()<br>B. groupname = createGroup()<br>C. groupname.add(Group)<br>D. groupname = createSprite() | **A** |
| **End the quiz panel** | |
| **FEEDBACK**<br>● **Encourage the student to make reflection notes in Markdown format.**<br>● **Compliment the student for her/his effort in the class.**<br>● **Review the content of the lesson.** | |

| Teacher Action | Student Action |
|---|---|
| You get Hats off for your excellent work!<br><br>In the next class, you will build a ranking mechanism that ranks the player according to their performance in the car racing game. We will also build a Progress Bar for fuel and player's life property. | *Make sure you have given at least 2 Hats Off during the class for:*<br>Creatively Solved Activities +10<br>Great Question +10<br>Strong Concentration +10 |
| *** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.**<br><br>**Project Overview**<br>FRUIT CATCHER - 1<br><br>**Goal**<br>In Class 40, you learned how to store the rank of the players in the database and display it when the player crosses the finish line. You also learned to create the progress bar and display the game over message when the fuel finishes before reaching the finish line. In this project, you have to practice what you learned in the class in this game called The Fruit Catcher.<br><br>**Story:**<br>Honey visited her grandparents, where there was a farm. Farmers there were cutting fruits to harvest them. Her grandfather gave her the responsibility to collect | *The student engages with the teacher over the project.* |

| | |
|---|---|
| these harvested fruits in a basket. Honey went with her cousin to the farm to collect the fruits.<br><br>Let’s see who collects more, Honey or her cousin.<br><br>I am very excited to see your project solution and I know you will do really well.<br><br>Bye Bye! | |
| **Teacher ends slideshow** | |
| **Teacher Clicks** ✖ End Class | |
| **ADDITIONAL ACTIVITIES** | |
| **Additional Activities**<br><br>*Encourage the student to write reflection notes in their reflection journal using Markdown.*<br><br>Use these as guiding questions:<br>● What happened today?<br>○ Describe what happened<br>○ Code I wrote<br>● How did I feel after the class?<br>● What have I learned about programming and developing games?<br>● What aspects of the class helped me? What did I find difficult? | *The student uses the Markdown editor to write her/his reflection as a reflection journal.* |

| ACTIVITY LINKS | | |
|---|---|---|
| **Activity** | **Activity Name** | **Links** |
| Teacher Activity 1 | BoilerPlate for C40 | https://github.com/pro-whitehatjr/C40RV_SpeedRacer_TeacherActivity |
| Teacher Activity 2 | Teacher Reference Code | https://github.com/pro-whitehatjr/C40RV_SpeedRacer_ReferenceCode |
| Student Activity 1 | BoilerPlate for C40 | https://github.com/pro-whitehatjr/C40RV_SpeedRacer_StudentActivity |
| Student Activity 2 | overlap() | https://molleindustria.github.io/p5.play/docs/classes/Sprite.html#method-overlap |
| Teacher Reference | Visual aid link | https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C40_V3_lit_WithCues.html |
| Teacher Reference | In-class quiz | https://s3-whjr-curriculum-uploads.whjr.online/91cd3010-fe46-4786-b42b-537f03b06919.pdf |
| Project Solution | Fruit Catcher-1 | https://github.com/pro-whitehatjr/Project-C40-V3 |