Topic	COLLIDING CARS	
Class Description	The student will use the concept of collision to detect the collision between cars and with obstacles and add a blast (BOOM!) image upon a collision between cars.	
Class	C42	
Class time	45 mins	
Goal	• Detect when cars collide with each other and with obstacles to reduce the life of players in the game. • Change the animation of the car when the life of a player reduces to 0. • Disable the control of cars, once the animation is changed (when the life of the player reduces to 0).	
Resources Required	• Teacher Resources ○ VSC Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources ○ VSC 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 min 20 min 5 mins
WARM-UP SESSION – 5 mins		
Teacher starts slideshow  from slides 1 to 8 Refer to the speaker notes and follow the instructions on each slide.		
Activity Details		Solution/Guidelines

How are you doing? Are you excited to learn something new? ***Run the presentation from slide 1 to slide 3.*** **Following are the objectives of the WARM-UP session:** ● Discuss the recap of the previous class with the student. ● Conduct the WARM-UP Quiz Session.	**ESR**: Thanks, yes, I am excited about it. Click on the slide show tab and present the slides.
Q&A Session	
Question	**Answer**
Select the correct option to create the **obstacles** at **random** x position. A. X = 200 B. X = random(0, width-100) C. X = random(0) D. X = random()	B
Select the correct option to assign value to **gameState** to end the game when the **obstaclesGroup** touches the **players**.  A. gameState = 0	C

B. gameState = 1 C. gameState = 2 D. gameState = 3	
Continue the WARM-UP session	
Activity Details	**Solution/Guidelines**
Run the presentation from slide 4 to slide 8 to set the problem statement. **Following are the objectives of the WARM-UP session:** ● Appreciate the student. ● Revise the collision concept learned in the previous class. ● Change the car image to a blast (BOOM!) image upon collision. ● Explain the reduction of the life bar and how it will be implemented.	Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in the student.
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 15 mins	
ACTIVITY ● **Reduce player's life once collision detected between car and obstacle.** ● **Bounce the car away from an obstacle post-collision.**	
Teacher Action	**Student Action**
Feel free to ask any questions or doubts you may have from previous classes?	**ESR:** Varied

We have almost completed the game, in today's class, we will add code to make our cars collide with each other and also with the obstacles group. We will also change the image of the car when the life of the player is reduced to zero. So, let us begin.	**ESR:** We can use **car[0].collide(Car[1])**
The teacher downloads Teacher Activity 1 and opens the code in VSC. *Make sure to replace the database code with SDK from your own database.* What should happen when a car collides with an obstacle?	**ESR**: The life bar should reduce.
Correct! We will create a new method to **handleCollision()** between the car and the obstacle in **Game.js**. Do you remember the **collide()** method we used in Trex? *In case the student says no, remind him about the **collide()** function used to find when the Trex is colliding with obstacles.* We will use that method here to check the collision between **obstacleGroup** & the **car**. Remember, the **cars** is an array, so we will need to check collision using the **index** number. We will check if the **player.life** (declared in **player.js** in the previous class) is **greater than 0**; if yes, we are gradually reducing the number by dividing the value by **4**, similar to what we did with fuel in the previous class.	**ESR**: Yes/ No

We set **player.life =185;** dividing it with **4** at each collision will give each player **3** chances before the value becomes **< 0**.

The teacher can change this number based on how many chances students want to give the player before life ends.

*The teacher writes the following command inside the **if** condition to match **(index === player.index)** so that the index of a particular car can be passed on to function in **play()**.*

We also need to modify the **update()** function in **player.js** to store the updated **life** property.

Call **handleObstacleCollision()** in the **play()** method and pass the index of the car to it.

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

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

Create **handleObstcaleCollision()** in game.js with an **if** condition to use **collide()** between cars and obstacles.

Create another **if** condition to reduce life if it is greater than 0.

Call this function in **play()**.

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

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

    player.update();
  }
}
```

Modify **update()** in player.js to save the value of **player.life** in the database.

This will allow us to update the value of **player.life** in the database.

```
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
  });
}
```

Let us check the output once. Do you see any issues? If you see closely, The life bar quickly goes to **0** upon colliding with an obstacle. This is because the collision condition remains true as the car is touching an obstacle and **player.life** keeps dividing by **4** resulting in it going below **0**.	**ESR**: Yes/No.
To resolve this issue, we will move the car away from obstacles as soon as it collides.	

Any suggestions on how we achieve it? For that, we need to know which player is pressing which arrow key. Let's create another property. We can name it anything, let us say we call it **leftKeyActive** in the **constructor()** of **game.js.** We will keep that key **false**, but when it is true on collision, we will bounce our car to either left or right based on which the user presses a key.	**ESR**: Varied. *The student declares the variable.*

Create **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;

  }
```

We also need to modify **handlePlayerControl()** to find out the key which the user has pressed, either the left or right key. On pressing the left arrow Key **this.leftKeyActive** property will be **true** & on pressing the right arrow key, the property will be **false**.	

Based on the condition in which the key is pressed, change the value of **this.leftKeyActive**.

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

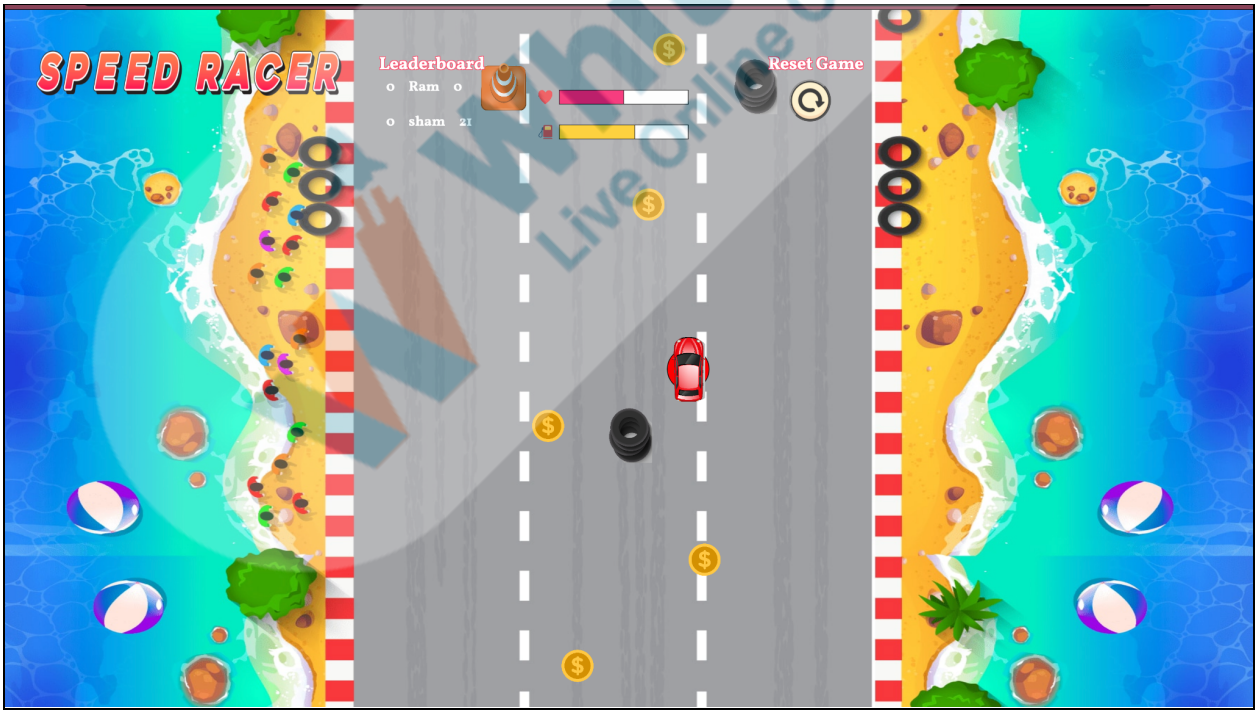
Finally, we will add a condition in **handleCollision()** to move the car away from obstacles after a collision based on which arrow key is pressed by the player.

Remember to call **this.handleObstacleCollision(index)** inside **play()** method.

The condition is added to check the status of **this.leftKeyActive** and x position of the player is move +/ - 100.

```
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();
  }
}
```

Let us run the code and check the result.



Superb! You can see how the life bar is reduced after each collision and the car is bouncing away. What else do we need to do in the game? Yes, we will also write code to collide both the cars, why don't you try to do it while I will guide you wherever needed.	**ESR**: We need to reduce the life bar and make the game over.
TEACHER STOPS SHARING SCREEN	
STUDENT-LED ACTIVITY – 20 mins	
STUDENT SHARES HIS SCREEN	
• **Ask the Student to press ESC key to come back to the panel** • **Guide the Student to Start Screen Share** • **The teacher gets into Fullscreen**	
ACTIVITY • **Detecting collisions between cars.** • **Replace car image with BOOM image.**	
Teacher starts slideshow  **from slides 9 to 13** Refer to the speaker notes and follow the instructions on each slide.	
Teacher Action	**Student Action**

Guide the student to open Student Activity 1, and download the zip folder, unzip it and save it as C42. *Make sure to add the database information in **index.html**.* *Note: In case any x or y positions were changed by the student in the code, make those changes in the position of objects as per his/her screen.* We will also add a blast (BOOM!) image when the player runs out of life (reduced to zero). The blast (BOOM!) image is given in the **assets** folder. Preload the blast image in **sketch.js** as shown below.	*Alternatively, the student can download **Student Activity 1.*** *The student declares the global variable and preloads the image to it.*

```
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
```

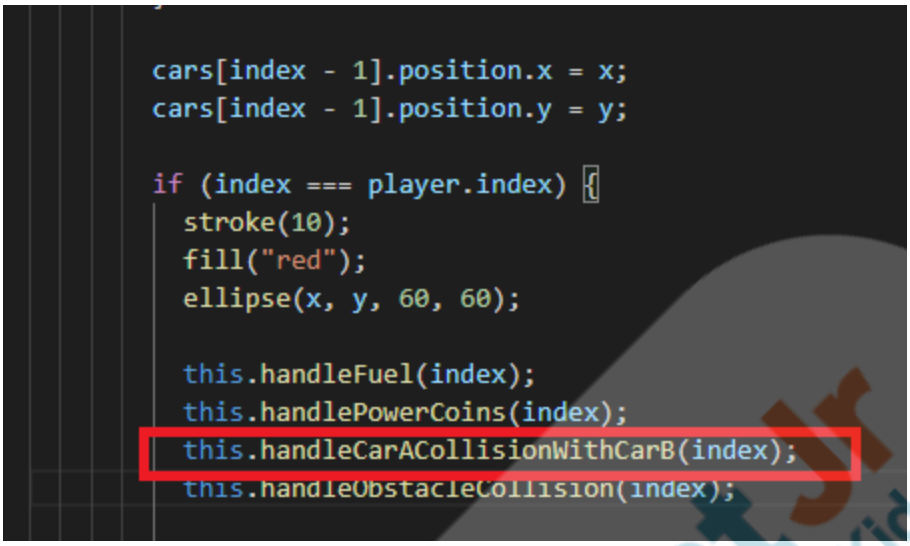
Add this image in the **Start()** method of **Game.js** on the Car Sprites.	*The student uses **addImage()** for both cars.*
	
When do we change the animation from car to blast? Inside the **for** loop that we created inside the **play()** method, we will create a variable to save the life of the current active player. Remember, we have added the life property to players in the previous class. Hence, our current **allPlayers** also contain a life property value. The **allplayers** is in **JSON** structure. Hence, we can access the value using **allPlayers[plr].life**.	**ESR:** When value of **player.life = 0;** *The student writes the condition within the **for** loop, to change the animation of the respective car,* *inside **play()** method.*

```
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];
```

We will save this value in a local variable **currentLife** and when this value becomes **0**, we will change the car animation with the blast (BOOM!) image. Remember the **changeImage** we used in a **T-Rex** game that we had created earlier?	

```
//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, let's check the collision between cars, we will call the **this.handleCarACollisionWithCarB()** function in **play()** method like all other functions.	*The student writes the function inside the **if** condition (**index == player.index**). So, that car's index can be passed on to the **this.handleCarACollision WithCarB()** function.*



```
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);
```

We will create this function similar to **handleObstacleCollision(index);** However, we need to check which car is colliding with the other, else, the life of both cars can be affected.

Hence, we will write a condition to check the index number of the car, and based on the index number, we will check its collision with the other car.

We will also make use of the **leftArrowKey** active property created in the last class to move cars away from the other cars after a collision.

Yes, apart from the above conditions, the rest of the code remains the same.

*The student looks at **handleObstacleCollision(index)** and writes a similar function for car collision and includes the **if-else** condition to check the car number.*

The condition checks if **index === 1,** and we will check collision between

```
(cars[index - 1].collide(cars[1])
```

and for **index === 2** the check will be as shown below:

```
(cars[index - 1].collide(cars[0])
```

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

Let us run the code and see the output.	*The student runs the code and checks the blast animation when the player's life is reduced to 0.*
	
Superb! What do you think is still left to do? We should disable all controls, once the car gets blown. How do you achieve that? We should create a property in the **constructor()** of **Game.js** to check the status of the blast. We will keep it **false** initially and when **player.life <= 0** we will change it to **true**. We also need to change **playerMoving** to **false** (created in the last class) so that the car does not stay in a forward movement in the **play()** method. Later, based on the value of the blast property, we will activate controls.	**ESR:** Varied *The student creates the **this.blast = false** property in **game.js**.*
Create a property **this.blast** 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;
    this.blast = false;
  }
```

Change **this.blast = true** in **play()** method.

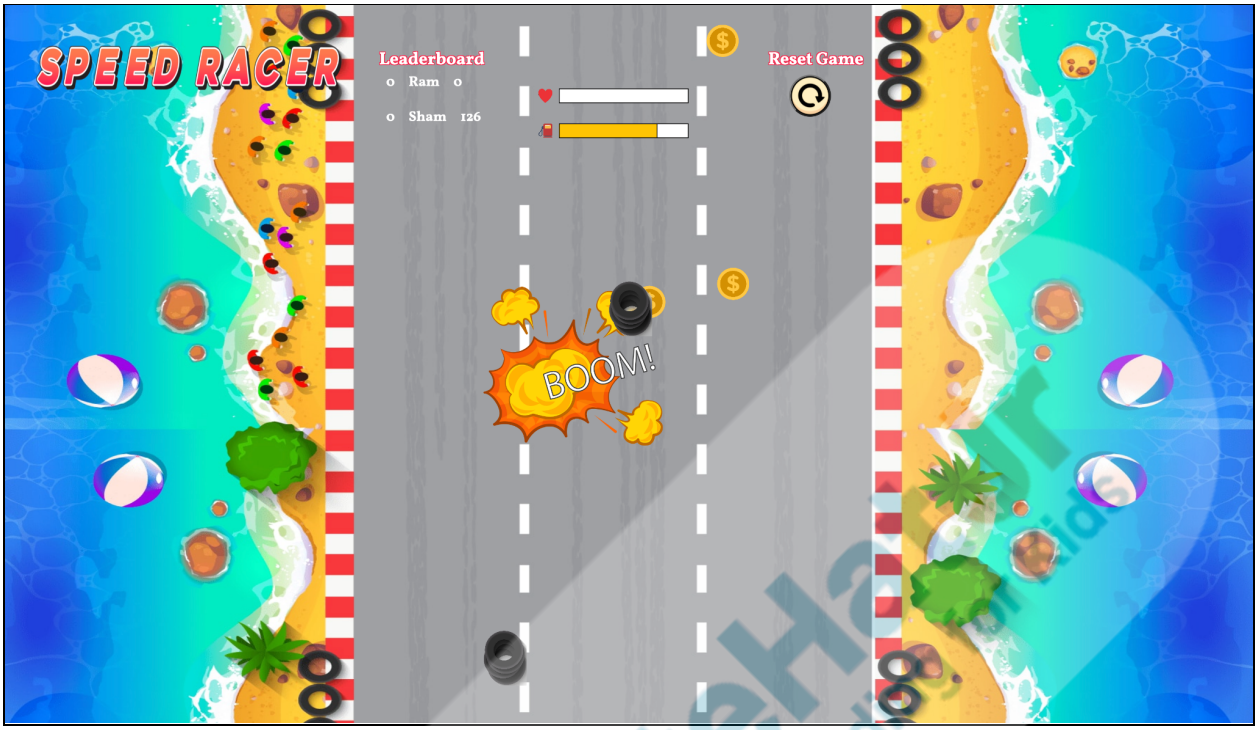
```
  if (index === player.index) {
    stroke(10);
    fill("red");
    ellipse(x, y, 60, 60);

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

    //C41 //TA
    if (player.life <= 0) {
      this.blast = true;
      this.playerMoving = false;
    }
```

One last thing before we check the output. Our entire **handlelayerControl()** should be activated only if **this.blast** is **true**.	*The student creates the **if** condition around the control given in the code.*

So, we will move the entire code in the function inside the **if** condition to make sure that the controls move the car only if **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();
    }
  }
}
```

Ask the student to run the code on a local drive. *The teacher can help the student fix any typos in the code if there is an error; else, check if all objects are placed well, you can change the x and y positions of the objects, if needed.*	*The student runs the code to check the output.*



Why don't you go through the entire code and check if we have any bugs to fix? *The Student may or may not notice we are calling **game.end()** in the **draw()** function, but we have not created it. You can ask him to open the console log after the game is over to check that error.* *Check if a student is able to spot the error, or else guide the student to spot that error.* Yes, that one last code we need to include in our **game.js**, we are not doing anything when **gameState** is 2. Hence, we can create an **end()** method with just a **console.log** message in **Game.js**.	**ESR:** Student reads the code **ESR:** *Student creates **end()** method in **Game.js**.*

```
}
end() {
  console.log("Game Over")
}
```

Output:

We will stop our game here, but if you like, you can keep making a game that is even more challenging and fun. You can add an option to refill the player's life. Any idea how would you do that? Great, what other things can you do in this game?	**ESR:** Create another additional sprite to increase the life of players. We can make the additional life sprite appear randomly on the track, when the car touches it, the life value of the player will increase. **ESR:** Varied.
*After everything runs fine, you can ask students to upload the game on **GitHub**. Publish the link on **GitHub** pages.*	*The student uploads the game on **GitHub**.*

Once a student shares a link with the teacher in open chat; the teacher and student race their cars to make the game more fun.	*The Student and teacher become players and play the game.*
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
FEEDBACK • **Encourage the student to make reflection notes in Markdown format.** • **Complement the student for her/his effort in the class.** • **Review the content of the lesson.**	
Teacher starts slideshow  **from slides 14 to 25.** Refer to speaker notes and follow the instructions on each slide.	
Activity Details	**Solution/Guidelines**
Run the presentation from slide 14 to slide 25. **Following are the objectives of the WRAP-UP session:** • **Revise the concepts** • **Wrap Up Quiz** • **Explain the facts and trivias** • **Project for the day** • **Next class challenge**	Guide the student to develop the project and share it with us.
Quiz time - Click on the in-class quiz	
Question	**Answer**
Why do we need the **this.blast** property? A. To change the car's animation B. To trigger a blast in the game C. To disable the car controls and movement after the player runs out of lives.	**C**

D. To assign the value of life for players as 0	
What should we do, if we want to show GameOver swal after this.blast is true? A. Create a new swal(); B. Call gameOver() method when the car changes animation. C. Call gameOver() method when this.blast is true. D. Cannot be done.	**C**
Which of the following is true for the collision between cars? A. Collision is being checked based on the player's index. B. The life of both the cars will increase on collision. C. The player's y position is changed on collision. D. We used .isTouching() to check the collision.	**A**
End the quiz panel	
Today's class was a lot of fun! Did you enjoy creating this game and playing with me? You get hats off.	**ESR: Varied** *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 Name:** **SHOOTING RANGE** **Goal of the Project:** In Class 42, you learned how to change animation and find collisions between Cars. In this project, you have to practice what you learned in the class to make a Shooting Range game. **Story:** Jake wants to participate in a national rifle shooting competition. Here, in this project, you will make a virtual shooting range game for Jake, so that he can practice and prepare himself before the competition. Bye Bye!	*Students engage with the teacher over the project.*
Teacher ends slideshow 	
Teacher Clicks 	
ADDITIONAL ACTIVITIES	
Encourage the student to write reflection notes in their reflection journal using Markdown.	*The student uses the markdown editor to write her/his reflections in the 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 LINKS		
Activity	**Activity Name**	**Links**
Teacher Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C42RV_SpeedRacer_TeacherActivity
Teacher Activity 2	Final Reference Code	https://github.com/pro-whitehatjr/C42RV_SpeedRacer_ReferenceCode
Student Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C42RV_SpeedRacer_StudentActivity
Teacher Reference	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C42_V3_lit_withcues.html
Teacher Reference	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/bfc9de76-e538-40c0-8724-d2ec2ab2b7ef.pdf