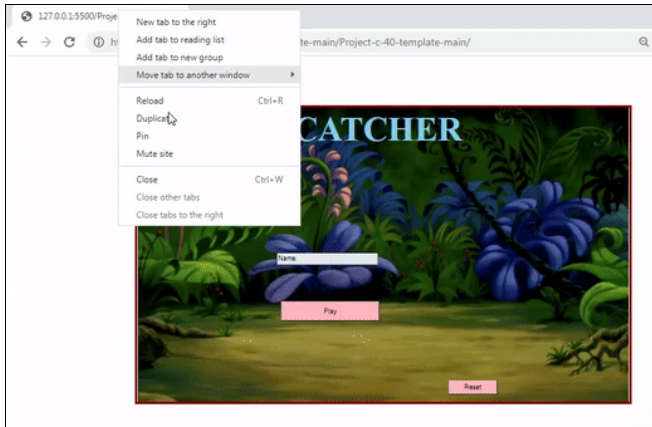


Topic	PLAYER RANKING & GAME OVER	
Class Description	Students build a ranking mechanism that ranks the player according to their performance in the car racing game. Students will also build a Progress Bar for fuel and the player's life property.	
Class	C41	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Store rank of individual players in the game as player property and update the database. • Display the rank of the player when the player crosses the finish line. • Create a progress bar for the player's life & fuel. • Display game over message when fuel is over before reaching the finish line. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources <ul style="list-style-type: none"> ○ 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 min 20 min 5 mins
WARM-UP SESSION - 5 mins		



Teacher starts slideshow from slides 1 to slide 11
Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines
<p><i>How are you doing? Are you excited to learn something new?</i></p> <p>Run the presentation slide 1 to slide 3.</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> Connecting students to the previous class. WARM-UP Quiz Session 	<p>ESR: Thanks. Yes I am excited about it.</p> <p>Click on the slide show tab and present the slides.</p>
QnA Session	
Question	Answer
<p>Select the correct option to include game.js in index.html?</p> <p>A. <code><script src="js/game.js"></script></code> B. <code><script href="js/game.js"></script></code> C. <code><src="js/game.js"></code> D. <code><script src="js/game.js"></code></p>	C
<p>Which of the following codes will display the player's name on the basket?</p>	D



- A. `text("Player 1 :" +allPlayers.player1,50,50);`
- B. `text("Player 1 :" +allPlayers.score,50,50);`
- C. `text("Player 1 :" +allPlayers,50,50);`
- D. `text("Player 1 :" +allPlayers.player1.score,50,50);`

Continue the WARM-UP session

Activity details	Solution/Guidelines
Run the presentation from slide 4 to slide 11 to set the problem statement.	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

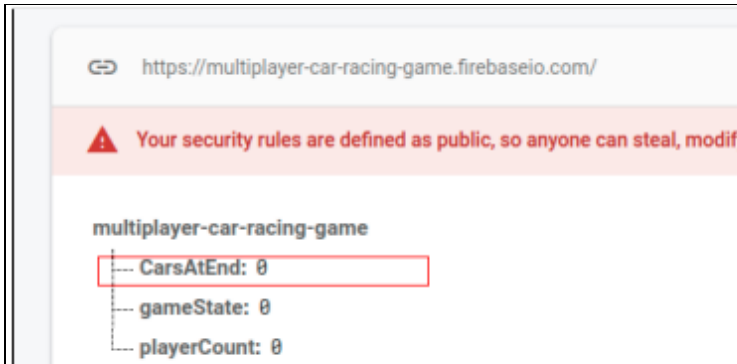
CHALLENGE

- Display the player rank.
- Display the life bar.

Teacher Initiates Screen Share

Teacher Action	Student Action
<p><i>Teacher to download Teacher Activity1 and replace the database.</i></p> <p>Our game is coming close to an end.</p> <p>We will start with giving ranks to players when they cross the finish line.</p> <p>Any suggestions on how we should start?</p> <p>To begin, as we can see one more property for each player called - 'rank' has been added. We will also add a field in our database to store ranks.</p>	<p>ESR: Varied</p> <p><i>Student Listens and Observes.</i></p>
<p>Following properties in the constructor() of player.js</p> <p>this.rank = 0 to store the rank of players.</p> <p>this.fuel = 185 to assign the initial fuel to the player; we will reduce it when the car is moving.</p> <p>this.life = 185 to assign initial life to the player, we will reduce it in later classes.</p> <p>this.score=0 to store the score.</p> <pre> 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; } </pre>	

Create a field “CarsAtEnd” in the Firebase database to record the player’s rank.



We have assigned both with 0; as at the start of the game cars are at the start point. The rank will be increased when the car crosses the finish line.

What should we do next?

Yes, we will write function **getCarsAtEnd()** to update **this.rank** with value of **CarsAtEnd** from database and vice versa in function **updateCarsAtEnd()**

Inside which file should create these functions?

Ask the student to guide you to write the function in player.js -

- *getCarsAtEnd() to read the value of the CarsAtEnd field from the database.*
- *updateCarsAtEnd() a static to update the CarsAtEnd field with the number of cars that have finished the race*

Do you remember what “static functions” are? We used it while getting **allPlayers**.

ESR: Read from the database!

ESR: player.js

ESR: Static functions are functions that are common to all objects created using the class.

Create `getCarsAtEnd()` and static `updateCarsAtEnd(rank)` to read and update value of `CarsAtEnd` field.

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

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

Call `getCarsAtEnd` in `play()` of `game.js`

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

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

Now that we have created a new field in the database, we have to make sure to reset it's value back to 0 when the reset button is pressed.

The teacher can show the added field in `handleResetButton()` of `Game.js`

```

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

```

Awesome! Where should we call this function?

ESR: Varied.

Inside the **play()** method of **game.js**, let's first declare the finish line.

The height of the track image is height*6. To make sure cars stay on the track we will define the finish line 100 px before the track ends.

Let's declare a **const finishLine = height * 6 -100**.

We will write an if condition when the positionY of a car goes beyond **finishLine** to change **gameState** to end by changing its value to 2. We will also increase **player.rank**. Call **update()** and **updateCarsAtEnd()** to update the database with position of cars and rank respectively.

In order to display the rank of the player, we will use the method **showRank()** and call within this condition. As we have this new method in the same class we can call it as **this.showRank()**.

ShowRank() is already given in boilerplate; if a student is very fast, you can stop here and let the student write showRank() as a part of Student Activity.

Create a condition for finishLine inside play() of game.js

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

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

Now let's understand the **showRank()** method. Here we will use the **swal()** function.

Open the [Student Activity 2](#), **swal()** is from SweetAlert library. <https://sweetalert.js.org/guides/>

ESR Student opens [Student Activity 2](#).

It is similar to the **alert()** function. However, alert gives a simple pop-up box. whereas in **swal()** we can customize image text and font etc.

swal() function accepts properties like:

- title:
- text:
- imageURL:
- imageSize:
- confirmationButtonText:

Explain a showRank() method to display player rank using swal() in game.js

```
showRank() {
  swal({
    title: `Awesome!${"\n"}Rank${"\n"}${player.rank}`,
```



```
text: "You reached the finish line successfully",
imageUrl:
```

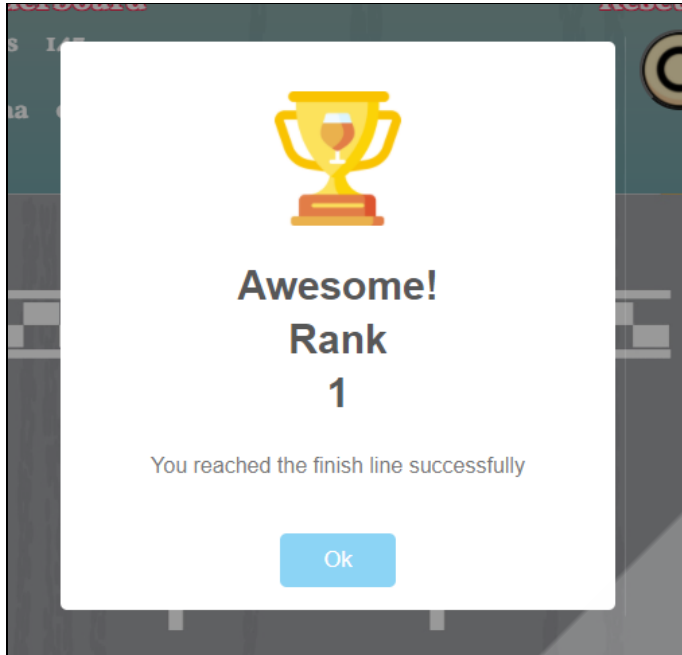
```
"https://raw.githubusercontent.com/vishalgaddam873/p5-multiplayer-car-race-game/master/assets/cup.png",
  imageSize: "100x100",
  confirmButtonText: "Ok"
});
}
```

For using swal() we need to add support libraries in index.html

```
<!-- Sweet Alert c40-->
<script
  src="https://code.jquery.com/jquery-3.5.1.min.js"
  integrity="sha256-9/aliU8dGd2tb60SsuzixeV4y/faTqgFtohetphbbj0="
  crossorigin="anonymous"
></script>
<script src="./lib/sweetalert.min.js"></script>
<link rel="stylesheet" type="text/css" href="./lib/sweetalert.css" />
```

The teacher can run the code to show the result.

OUTPUT:



Now let's write the code to **showLife()** in **game.js** to create a progress bar.

We are using this method to display two rectangle objects one on the other.

Do you remember which function we use to create a **rectangle()**?

We will create two rectangles one on another. The top one will be given a darker shade and the bottom one is white.

To create a progress bar effect, we will reduce the width of the upper rectangle with the value of **player.life**. When the life of the player reduces, the darker rectangle will become smaller showing a white rectangle.

We need to call this function as well in our **play()** method.

```
this.showLife();
```

The image displays a little heart image next to it.

ESR: rect()

*This image is preloaded in sketch.js
Make a note of the numbers if they are changing.*

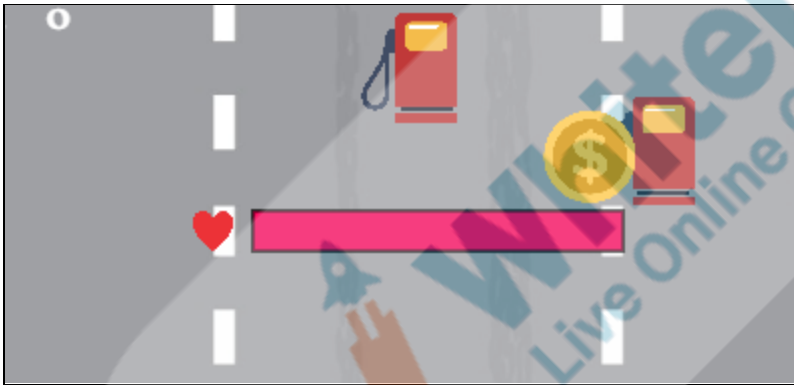
Create showLife() to display a progress bar for fuel in game.js

```

    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();
    }
  
```

Change this value in y position to display bar in your browser

OUTPUT:



Run the code to show the output.

With this guide in mind, why don't you create a fuel bar, and once the fuel is over we will use **swal()** to display game over?

Teacher Stops Screen Share

Share your screen; I will be guiding you in the exercise.

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

- Create a Fuel bar.
- Create gameOver().
- Give AI to cars in play().



Teacher starts slideshow from slide 12 to slide 14.
Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
<p>Guide the student to open the Student Activity 1 and download the zip folder, unzip it and save it as C41.</p> <p>Make sure to add the database information in index.html.</p>	<p>Alternatively, the student can clone Student Activity 1.</p>
<p>Similar to showLife() we have another function to showFuelBar().</p> <p>Code is given in boilerplate; the teacher should request the student explain the code.</p> <p>Ask the student to check the output; they may need to change y-position in showFuelBar() showLife() as per their browser size.</p>	<p>ESR:</p> <p>Student reads function showBarFuel();</p> <p>It draws two rectangles one on each other, lower one with white color and upper one with different shade.</p> <p>There is an image of a fuel tank on the left side.</p> <p>The function this.showFuelbar() is</p>

called inside the **play()** method.

Explain showFuelBar() in game.js

```
showFuelBar() {
  push();
  image(fuellImage, 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



Great! Now, what should we do with the Fuel bar?

It should reduce when the car is moving and it should increase when the car overlaps with the fuel tank.

For that, we need to keep checking when the car is moving or it is steady,

For this purpose, we will create a property in the **constructor()** of **game.js** to keep checking the car movement.

Initially, when the car is being displayed on the screen, we will keep this value false; and once the car starts moving we will change it to true.

ESR: Varied

ESR: Student creates property
this.playermoving = false
in constructor() of game.js

*Guide Student to assign a value true to this property in **handlePlayerControl()** function.*

Create a property `this.playerMoving= 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;
  }
}
```

Modify `handlePlayerControl()` to update `this.playerMoving` to true.

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

Now, based on **`this.playerMoving`** and **`player.fuel`** property that we created in the last class, we will modify the function we created to manage the fuel bar named **`handleFuel()`**.

The teacher guides the student to write the code and explains it.

Add conditions to check if **`this.playerMoving` is true** and if **`player.fuel` is greater than 0**.

*The student writes conditions in method **`handleFuel()`** in `Game.js`.*

If both conditions are **TRUE** then reduce the **player.fuel** by **0.3** (this number will decide the speed of fuel reduction it can be higher to reduce fuel faster).

In addition, we will also add a condition to check if fuel is less than or equal to 0 then gameState will be changed to 2 (end) and call **gameOver()** method.

```

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();
  }
}

```

If the value of **player.fuel** is greater than 0 and the player's car is moving, we will reduce player.fuel gradually.

Once **player.fuel** is 0, game over will be displayed for that player and **gameState** is changed to 2.

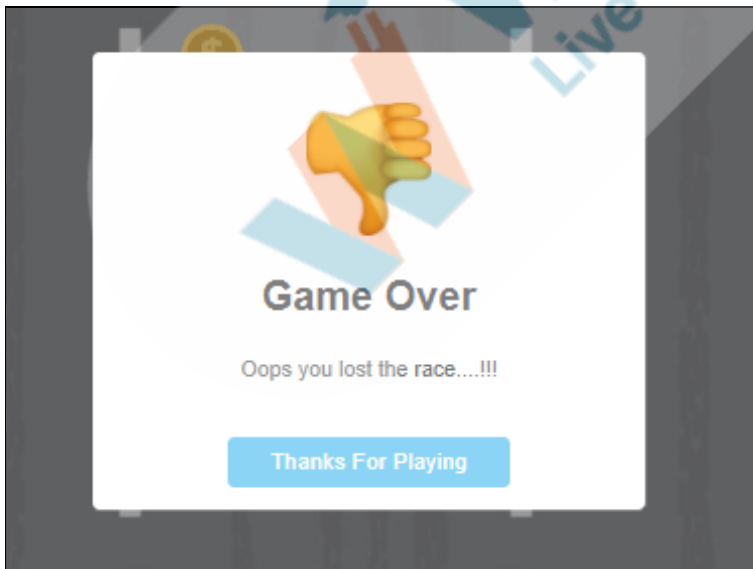
gameOver() method uses **swal()** function to display the pop-up.

```
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"  
  });  
}
```

Ask students to check the output. Ask the student to notice reducing fuel bar and not to collect fuel tanks so see the game over message.

ESR: Student runs the code to see the output.

OUTPUT:



Superb! However, there is a small bug in our game currently. Can you identify what is it?

Correct! Any idea how we can fix that issue?

Currently, we are moving the car 10 px when the up arrow key is pressed.

Guide the student to check `handlePlayerControl()` to see how cars are moving when the up arrow is key pressed; we add 10 to y-position of car.

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

We can make use of property **playerMoving**, to keep the car moving automatically even when the up arrow key is not pressed. Remember how we gave AI to T-rex in the game, similarly, we will add AI to cars.

So basically our up arrow key will work as an accelerator for increasing speed of the car.

Inside the **play()** method in **Game.js**, we can use a condition to move the car in the y direction when the value of **playerMoving** is true. And update the same in the database.

ESR: The car stops moving when we are going right or left or when we do not press the up arrow key.

ESR: Varied.

*ESR: Student writes following code in **play()** of **game.js***

Apply AI to car to move forward with if condition in play() of game.js

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

Superb! Let us run the code and check if the progress bar is working and if we get **swal()** at the end to display the rank.

If the time permits, upload the code on GitHub, where teacher and student can play the race together

We are adding so many new methods in each class; because we structured code correctly in the first class, it is easy to keep adding new methods easily.

Let us stop here for today. We have created our game in a rewarding way; we will make it more challenging in the next class.

Did you enjoy the class?

Student Runs the code

ESR: Varied.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 Mins



Teacher starts slideshow from slides 15 to slide 25.
Refer to speaker notes and follow the instructions on each slide.

Teacher Action

Student Action




Run the presentation from slide 15 to slide 25.

Following are the WRAP-UP session deliverables:

<ul style="list-style-type: none"> ● Revise the concepts ● WRAP-UP Quiz ● Explain the facts and trivia ● Project for the day ● Next class challenge 	Guide the student to develop the project and share with us
Quiz time - Click on in-class quiz	
Question	Answer
<p>What can be passed in swal()?</p> <p>A. images B. text C. title D. all of the above</p>	D
<p>What is the approach used for progress bars?</p> <p>A. set different images of the bars according to different conditions B. keep a darker rectangle on top of a lighter rectangle, and expose the lighter rectangle C. put a red image on top of the white image D. keep a lighter rectangle on top of a darker rectangle, and expose the lighter rectangle</p>	B
<p>How are we giving AI to car movement?</p> <p>A. by decreasing positionX when the left arrow key is pressed, and updating the same in the db B. by increasing positionX when the right arrow key is pressed, and updating the same in the db C. by increasing positionY when the up arrow key is pressed and updating the same in the db D. by increasing positionY when this.playerMoving is true and the game is in play() method</p>	D
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
<p>Awesome! You get hats off.</p> <p>In the next class, We will end this game by detecting collisions between cars and between cars and obstacles and reduce life.</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div>  </div> <div>  </div> <div>  </div>
<p>* This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.*</p> <p><u>Project Name:</u></p> <p>FRUIT CATCHER - 2</p> <p>In Class 41, you revised all the concepts learned in the Speed Racer game so far. You also learned how to create obstacles to make the game more difficult. In this project, you have to practice what you learned in the class to make some obstacles for the Fruit Catcher game.</p> <p>Story:</p> <p>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 these</p>	<p><i>Students engage with the teacher over the project.</i></p>

harvested fruits in a basket. Honey went with her cousin to the farm to collect the fruits.

Let's see who collects more, Honey or her cousin. Here, you will make the game more challenging in this project by adding obstacles to the game.

I am very excited to see your project solution and I know you will do really well.

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.

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?

The student uses the Markdown editor to write their reflections in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C41RV_SpeedRacer_TeacherActivity
Teacher Activity 2	Teacher Reference Code	https://github.com/pro-whitehatjr/C41RV_SpeedRacer_ReferenceCode
Student Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C41RV_SpeedRacer_StudentActivity
Student Activity 2	swal()	https://sweetalert.js.org/guides/
Teacher Reference	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Assessment/PRO_VD/PRO_C41_V3_lit_withcues.html
Teacher Reference	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/8443f29b-8a2e-48a0-b00e-30b833822269.pdf
Project Solution	Fruit Catcher-2	https://github.com/pro-whitehatjr/Project-C41-V3