WhiteHat Jr
Live online Coding for Kids

# CHECKPOINT REVISION

**CHECK THE NUMBER OF PENDING PROJECTS & QUIZZES OF THE STUDENT IN THE PANEL.**

**IF THE STUDENT HAS MORE THAN 3 PENDING/EXPIRE D PROJECTS OR QUIZZES, THEN GO AHEAD WITH THE REVISION CLASS.**

**IF THE STUDENT HAS COMPLETED ALL THE PROJECTS OR HAVING NO DOUBTS, THEN CONTINUE WITH THE BELOW LESSON PLAN.**

**Note to teachers [Only Applicable for C34]:**

**This is a CHECKPOINT REVISION CLASS meant for revising concepts learned so far. Teachers should guide students to complete pending/expired projects. If the student has pending projects less than 3, then you can continue to do the class activity.**

**IF STUDENT HAS > 3 PENDING PROJECT**
**SKIP CLASS ACTIVITY (INCLUDING VA & QUIZ) & HELP THEM COMPLETE PROJECTS**

**ELSE**
**CONTINUE CLASS ACTIVITY**

**Next class will be C35**

| | |
|---|---|
| **Topic** | **CHECKPOINT REVISION CLASS: PHYSICS ENGINE GAME** |
| **Class Description** | Students will learn how to add stars and how to collect them with fruit to make the game more challenging to play. |
| **Class** | **C34** |
| **Class time** | **55 mins** |
| **Goal** | ● Add the star sprite.<br>● Add a blower button.<br>● Create the star counter.<br>● Detect collision with stars |
| **Resources Required** | ● Teacher Resources<br>○ VS Code Editor<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>● Student Resources<br>○ VS Code Editor<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen |

| Class structure | Warm Up - Slide show option<br>Teacher-Led Activity<br>Student-Led Activity<br>Wrap Up - Slide show option | 10 Mins<br>10 Mins<br>30 Mins<br>5 Mins |
|---|---|---|

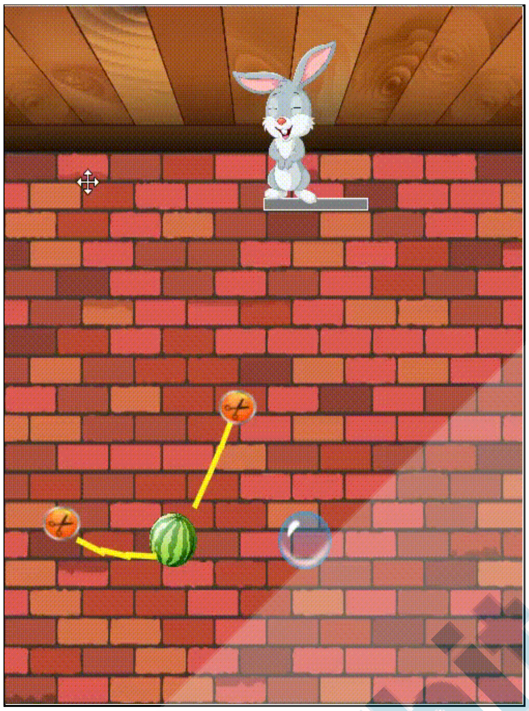
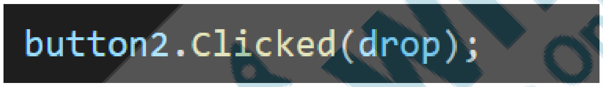
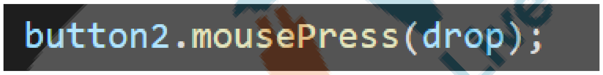
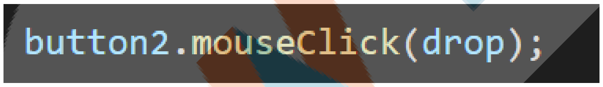
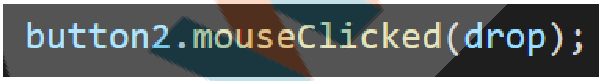
**WARM UP SESSION - 10 mins**

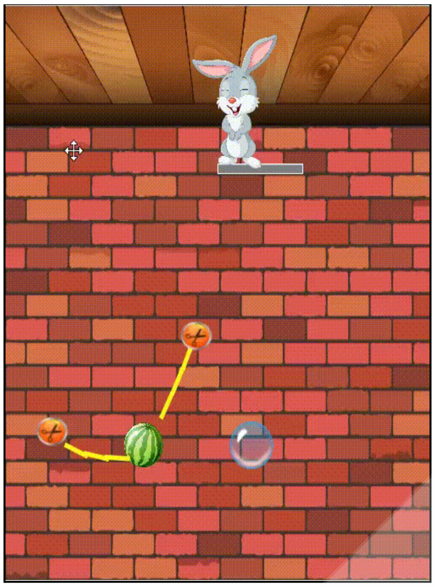
**Teacher starts slideshow from slides 1 to 13**
Refer to speaker notes and follow the instructions on each slide.

| 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?*<br>**Run the presentation from slide 1 to slide 3**<br>**Following are the warm up session deliverables:**<br>● **Recall the previous class concept** | **ESR**: Hi, thanks, Yes I am excited about it!<br>*Click on the slide show tab and present the slides.* |

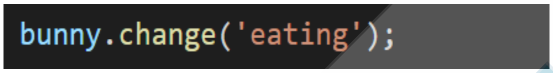
**QnA Session**

| Question | Answer |
|---|---|

| | |
|---|---|
| Which of the following codes will add the button to cut the rope on the left side?<br><br>A.  `button2.Clicked(drop);`<br>B.  `button2.mousePress(drop);`<br>C.  `button2.mouseClick(drop);`<br>D.  `button2.mouseClicked(drop);` | **D** |
| What is the instruction to play bunny's eat animation when the fruit collides with bunny? | **B** |

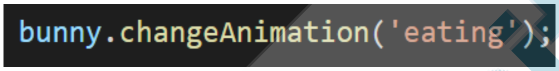


A.



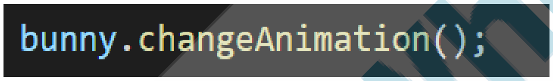
```
bunny.change('eating');
```

B.



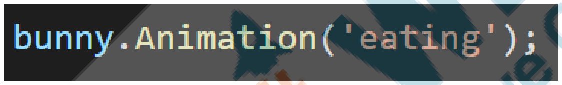
```
bunny.changeAnimation('eating');
```

C.



```
bunny.changeAnimation();
```

D.



```
bunny.Animation('eating');
```

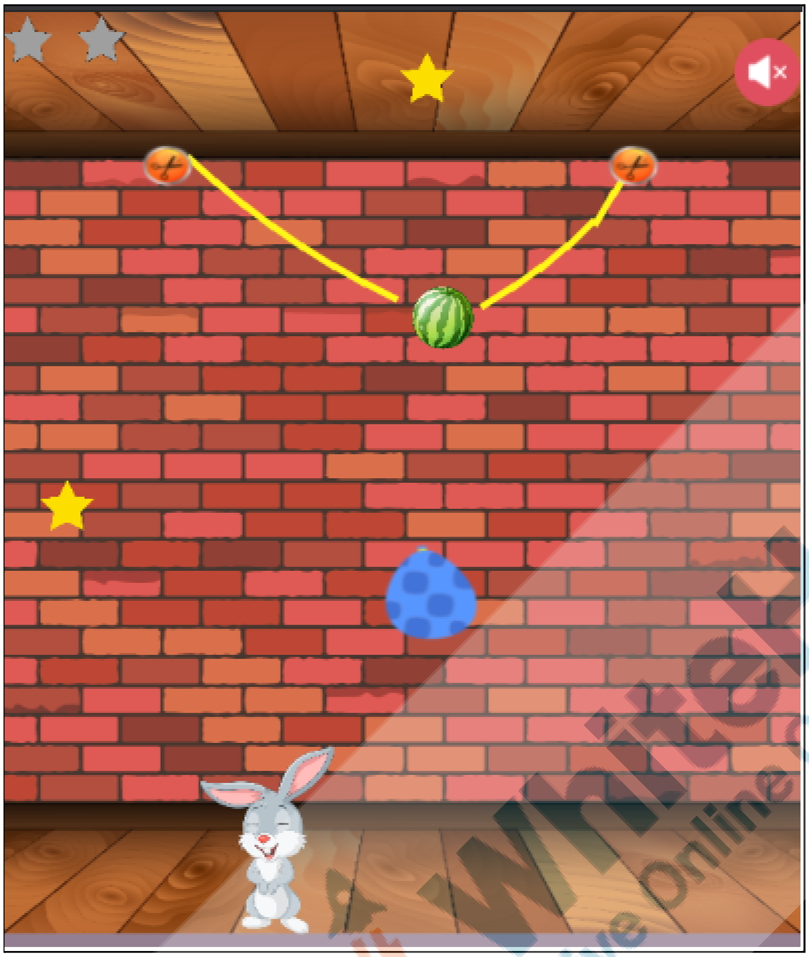
**Continue the warm up session**

| **Activity details** | **Solution/Guidelines** |
|---|---|
| **Run the presentation from slide 4 to slide 13 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 - 10 mins**

<table>
<tr><th colspan="2">Teacher Initiates Screen Share</th></tr>
<tr><td colspan="2"><u>CHALLENGE</u><br>● New level Overview.</td></tr>
<tr><th>Teacher Action</th><th>Student Action</th></tr>
<tr><td><i>Teacher opens the <u>hosted game</u> link.<br>Shows the complete game output to the students.</i><br><br>Till this point, we have added multiple ropes and a blower in the game. We can make the game even more challenging.<br>In this class, you are going to make this game more challenging for the player to play.<br><br>Do you have any ideas for making this game more challenging?<br><br>Great ideas, you can incorporate those too...<br><br>For today’s capstone class, we will revise most of the concepts we have learned in Physics Engine.<br><br>You will add stars in the game that the player has to collect. But these stars are not in the normal reach of the fruit. So to get the star we need to push the fruit.<br><br>I also want you to add a scoring system, which checks how many stars we have collected.<br><br>Are you excited to make this?</td><td><b>ESR:</b><br>Varied<br><br><b>ESR:</b><br>Yes!</td></tr>
</table>

| | |
|---|---|
|  | |
| Now please share your screen with me. | |

| Teacher Stops Screen Share |
|---|
| **STUDENT-LED ACTIVITY - 30mins** |
| ● **Ask Student to press ESC key to come back to panel**<br>● **Guide Student to start Screen Share**<br>● **Teacher gets into Fullscreen** |

<table>
<tr><th colspan="2">

**ACTIVITY**

- **Change the rope's position**
- **Create Star sprites**
- **Detect collision with stars**

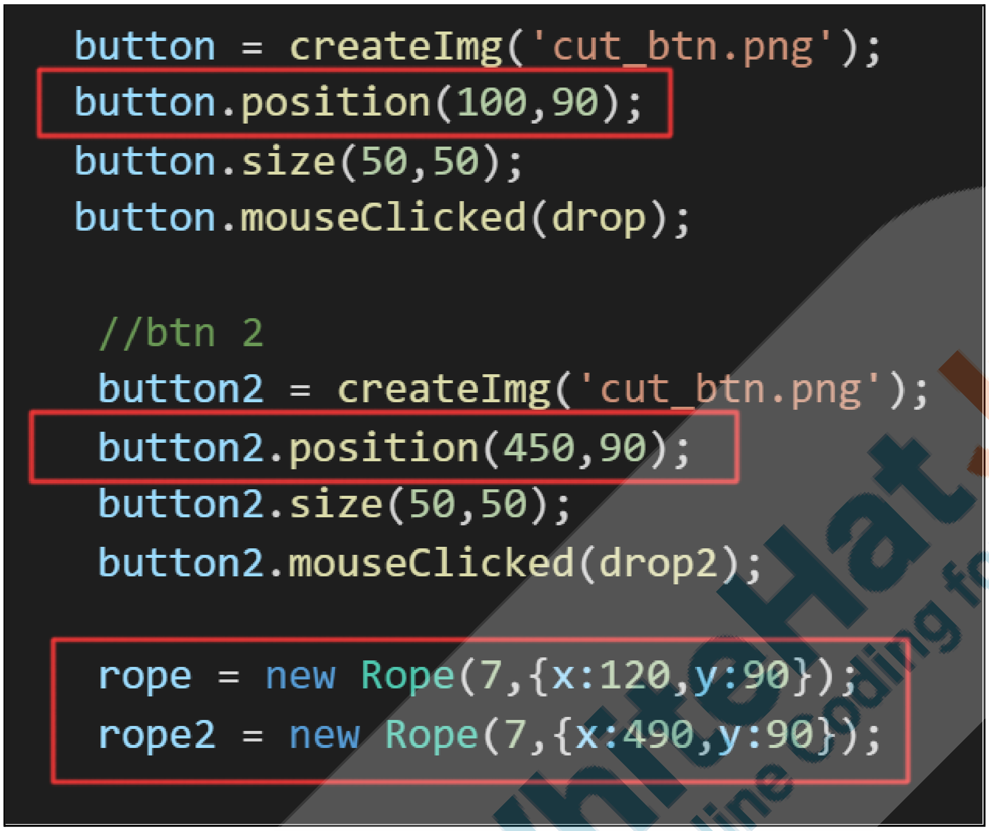
</th></tr>
<tr><td>

**Student - Led Activity**
*Student downloads the code from student Activity 1 and opens it in the VS code editor.*

As you have seen in the game, we have a different arrangement of the ropes now.
We only have 2 ropes, one is connected at the left side and the other is connected at the right side in such a way that fruit hangs in the middle of the canvas.

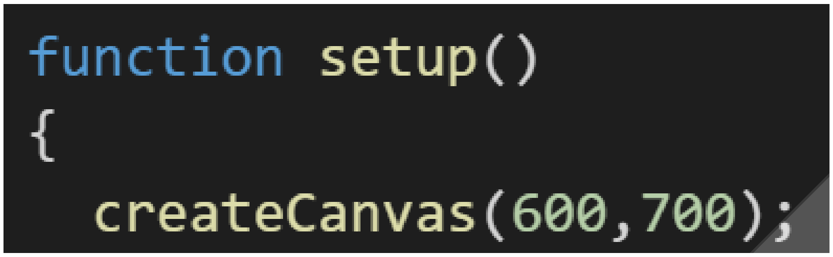
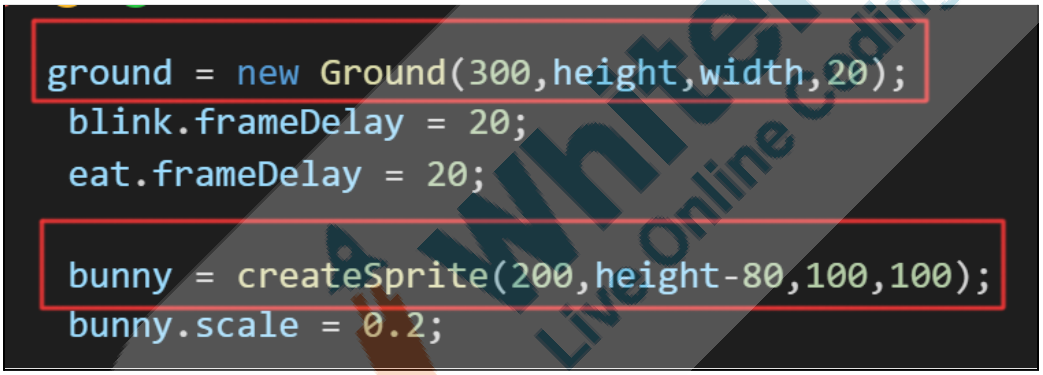
First, you will change the position of the ropes and along with the rope you need to change the positions of buttons as well.
For both the ropes we will keep the number of elements the same, that is 7.
Because we want to keep the length of the rope equal.
*Student writes the code.*

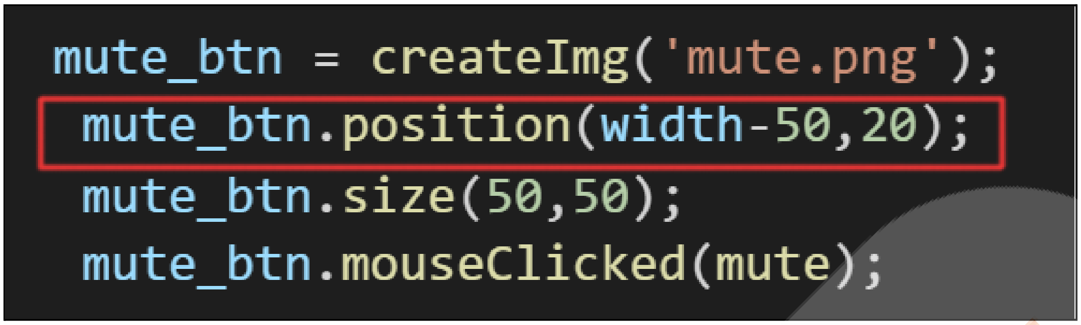
</td><td></td></tr>
</table>



```
  button = createImg('cut_btn.png');
  button.position(100,90);
  button.size(50,50);
  button.mouseClicked(drop);

   //btn 2
   button2 = createImg('cut_btn.png');
   button2.position(450,90);
   button2.size(50,50);
   button2.mouseClicked(drop2);

   rope = new Rope(7,{x:120,y:90});
   rope2 = new Rope(7,{x:490,y:90});
```

| | |
|---|---|
| Let's run the code.<br>We have our ropes in place but everything is looking a bit crowded.<br>You can increase the width of the canvas to fit everything nicely, earlier we had the width as 500 px but now you can increase it upto 600 px.<br>When we change the canvas size, do you think it will impact other objects?<br>Correct, we need to adjust the position of the mute button, ground body, and the background images. We will specify their position in such a way that they are related to the width of the canvas. | *Student starts the live server and observes the output.*<br>**ESR:**<br>Yes, the width of other objects needs to be adjusted with the new canvas size. |

| | |
|---|---|
| Let's modify the **x** position and **width** of the canvas. | |
| <br>`function setup()`<br>`{`<br>`  createCanvas(600,700);` | |
| Now let’s arrange the position of the mute button, ground body, bunny sprite, and the size of the image.<br>We will set their new position in relation to the width and height of the canvas. So that when we increase or decrease the canvas width or height we don't have to set the position manually. | *The student changes the canvas size and the x position of other objects.* |
| <br>`ground = new Ground(300,height,width,20);`<br>`  blink.frameDelay = 20;`<br>`  eat.frameDelay = 20;`<br><br>`  bunny = createSprite(200,height-80,100,100);`<br>`  bunny.scale = 0.2;` | |
| For the position of the mute button we can do a similar thing: we can define its x position as width-50.<br><br>*The teacher can ask the student to try different numbers instead of giving the position directly.* | *Student tries different values for the mute button* |

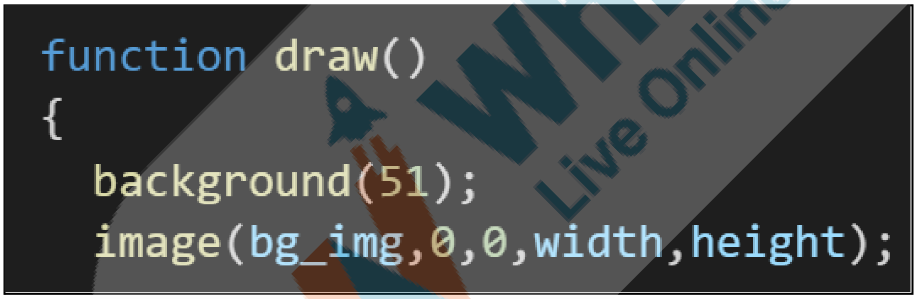


```
mute_btn = createImg('mute.png');
  mute_btn.position(width-50,20);
  mute_btn.size(50,50);
  mute_btn.mouseClicked(mute);
```

When we add the image on canvas, we can define its width and height by our choice also, if we don't then p5.js takes it as the size of the image added.
The original size of the background image is 500 x 700, but now the canvas is 600 in width so we have to change the width of the image displayed on the canvas to cover the whole canvas.
We can do this in the **image()** function.

We will define the width and height of the image the same as the width and height of the canvas.

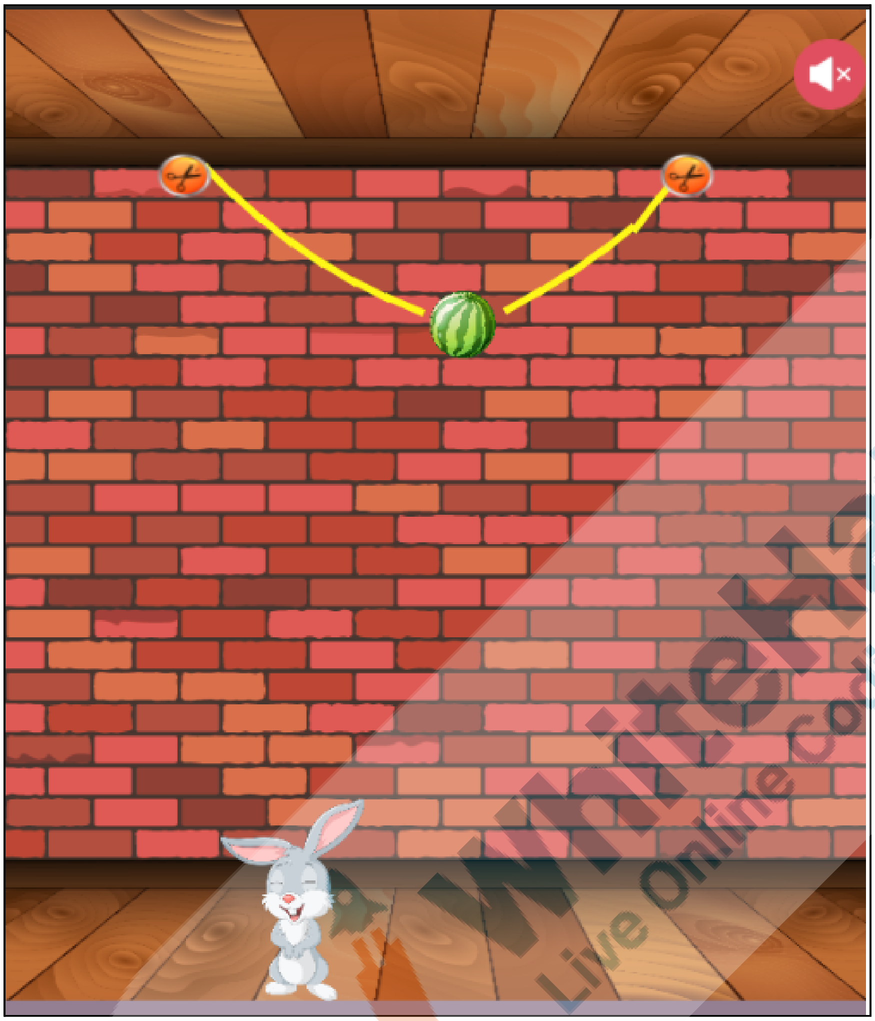
*The student tries different values for the background image.*



```
function draw()
{
  background(51);
  image(bg_img,0,0,width,height);
```

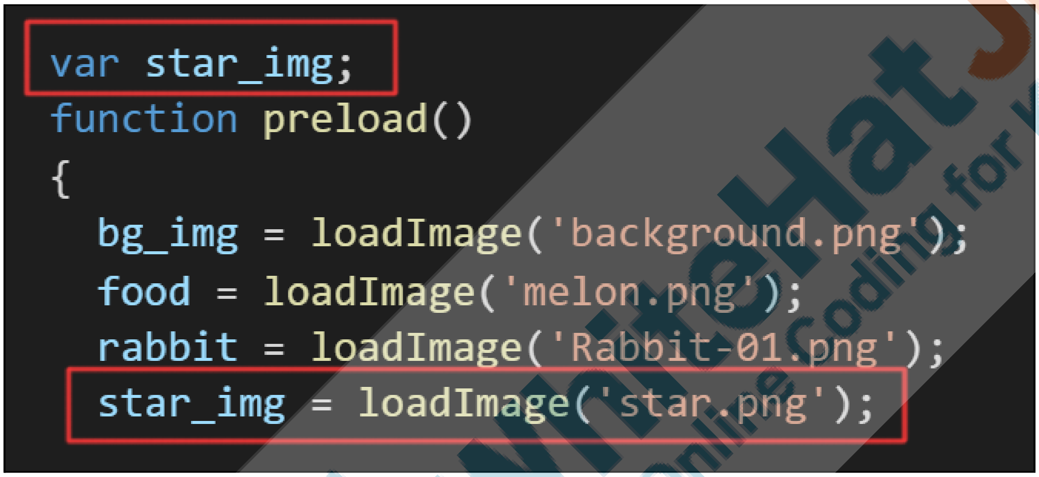
Let's run the code and see how it looks.

*The student runs the code and observes the output.*



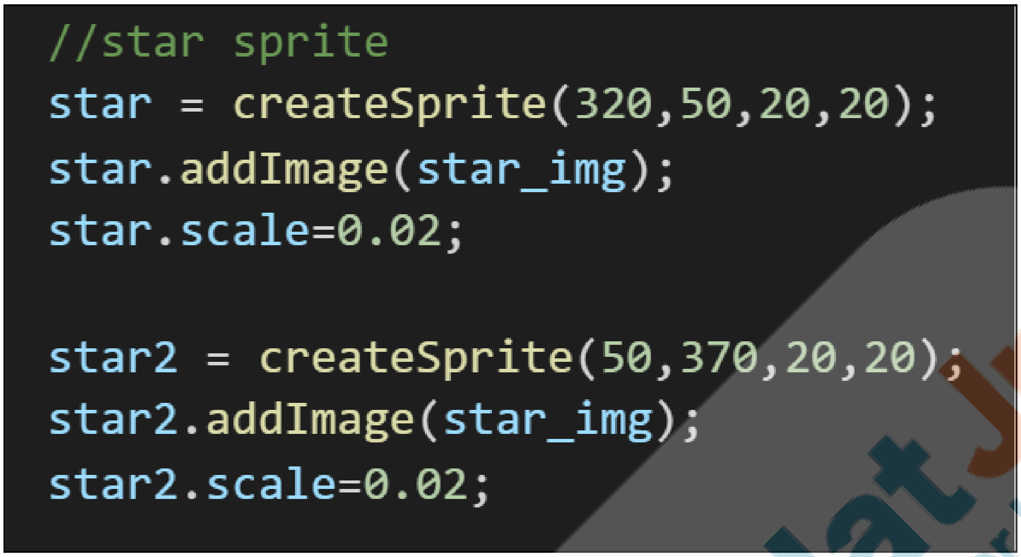
| | |
|---|---|
| Great!<br>Our game looks really good now.<br>Now you can add the stars.<br>There are few concepts with the stars which we need to understand.<br>1. You create 2 star sprites that have images of the star.<br>2. You need to detect the collision of fruit with the star.<br>3. You will create a star counter system that will show how many stars have been collected.<br><br>First, let's create 2 star sprites and load and add images | **ESR:**<br>First, we laid the images in |

| | |
|---|---|
| for them.<br>Can you tell me how we can do this?<br><br>Great!<br>Very well done.<br>Let’s start by loading the image in the **preload()** function.<br><br>First, define the variables for the image of the star, we only need to load 1 image because we can set the same image for both the stars for now. | the **preload()** function.<br>Then create sprites using the **createSprite()** function and then add the image to the sprites. |



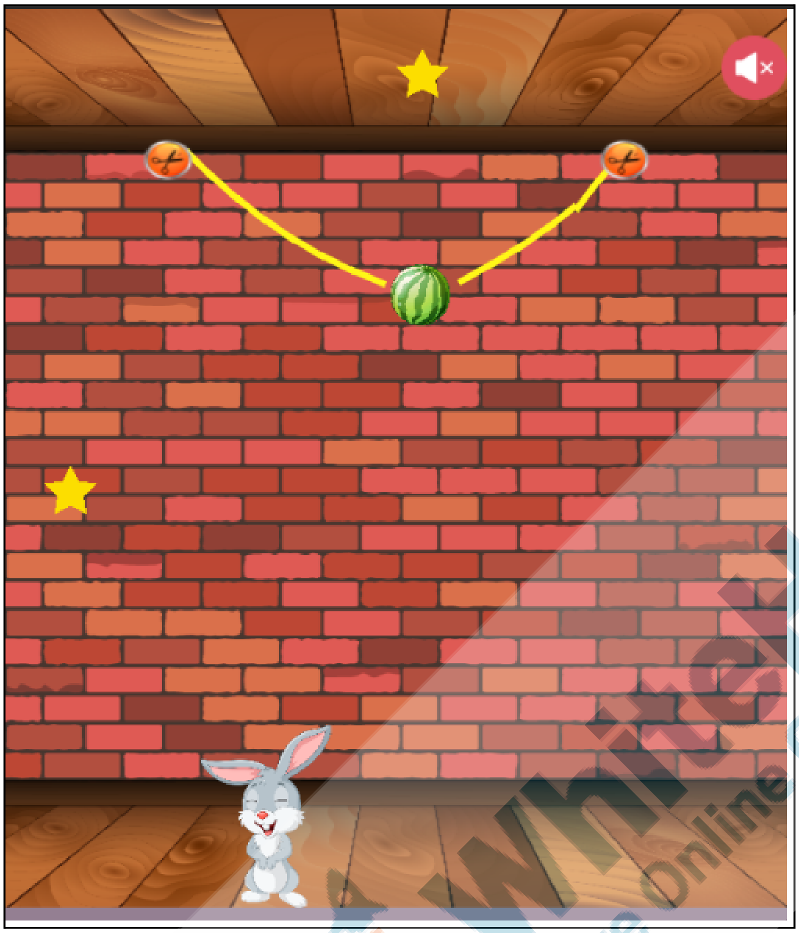
```
var star_img;
function preload()
{
  bg_img = loadImage('background.png');
  food = loadImage('melon.png');
  rabbit = loadImage('Rabbit-01.png');
  star_img = loadImage('star.png');
```

| | |
|---|---|
| Now you can create the sprite and add the images.<br><br>First declare 2 variables as **var star,star2;**<br><br>You also need to change the scale of the sprite because the image used is very large in size. | |

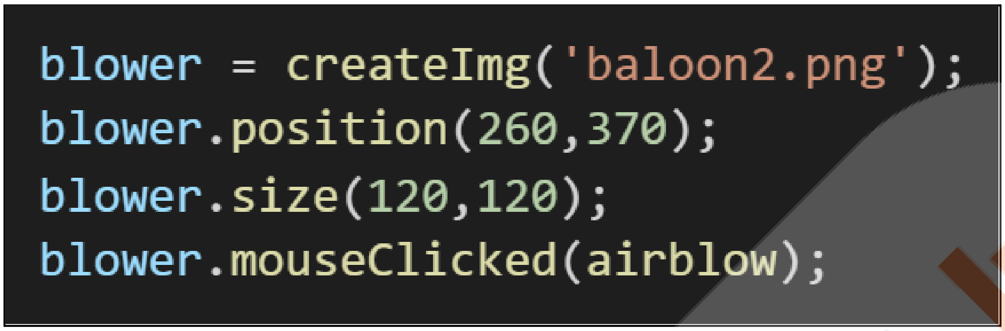


```
//star sprite
star = createSprite(320,50,20,20);
star.addImage(star_img);
star.scale=0.02;

star2 = createSprite(50,370,20,20);
star2.addImage(star_img);
star2.scale=0.02;
```

| | |
|---|---|
| Now we can run the code to see the stars on the canvas. | *The student runs the code and observes the output* |

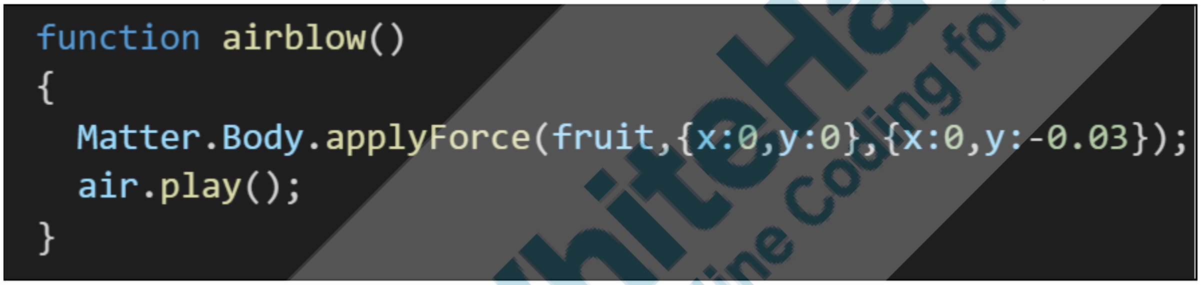
| | |
|---|---|
|  | |
| We have the star on the canvas. But one star is above the fruit.<br>Our fruit can not reach that star in a normal way.<br>Can you think what can be done to solve the issue?<br>Great! | **ESR:**<br>We can use an air blower to push the fruit upwards. |
| You will create the air blower image button and attach the function to apply the force on the fruit.<br>In which direction should we apply the force? | **ESR:**<br>Y direction. |

Good!
We will apply the force in the upward y direction.



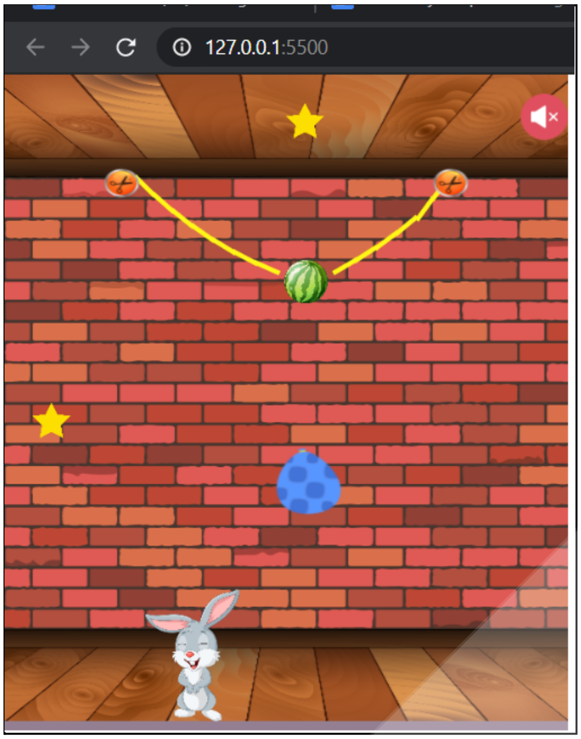
```
blower = createImg('baloon2.png');
blower.position(260,370);
blower.size(120,120);
blower.mouseClicked(airblow);
```

Now let’s define the **airblow()** function.



```
function airblow()
{
  Matter.Body.applyForce(fruit,{x:0,y:0},{x:0,y:-0.03});
  air.play();
}
```

Let’s run the code and click on the balloon button to see how it works.

| | |
|---|---|
|  | |
| What happens when we click on the balloon button? | **ESR:**<br>Fruit goes up. |
| Yes!<br><br>We got the balloon working.<br><br>Now let's focus on how to detect the collision with the star. We have done this in previous classes where we detect collision between the bunny and fruit.<br><br>Can you suggest how to do that?<br><br>There we created the **collide()** function which takes 2 arguments which are the bodies or the sprites between which we want to detect the collision.<br><br>But it has the distance value fixed.<br><br>What will we do here to incorporate distance? | **ESR:** We can check the distance between the two objects.<br><br>**ESR:** Varied |

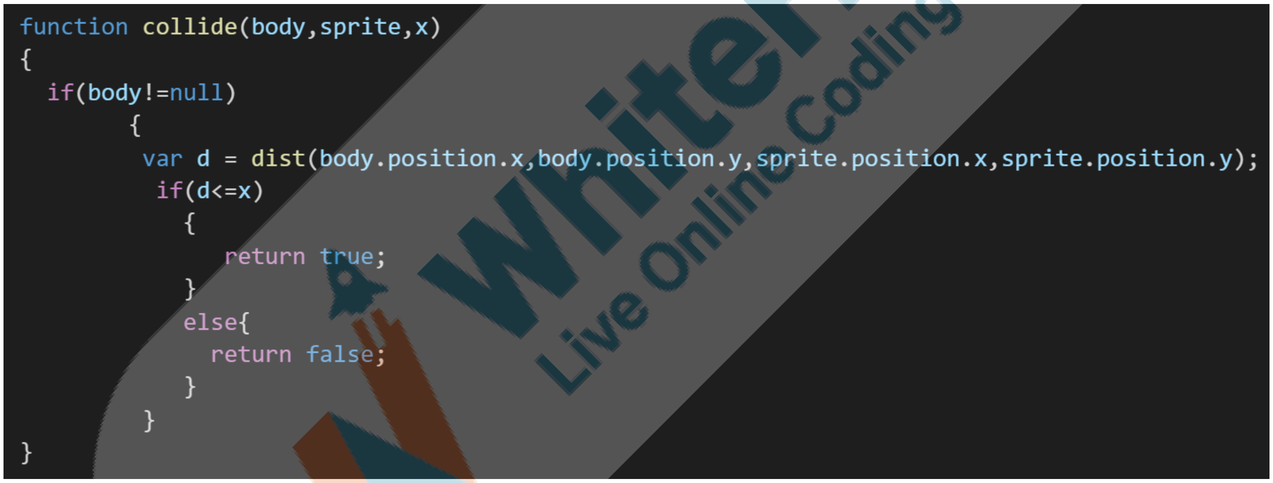
We can pass one more argument as a fixed distance. Modify this function to detect the collision between the objects when the distance between them is less than 80.

We can also do the same by making it user-defined by passing that value by the user.

Now this function will require 3 arguments when we call the function.

2 sprite or body objects and the distance threshold value from which we can say that it is a collision.

If the distance between these two objects is less than that value it will be a collision.



```
function collide(body,sprite,x)
{
  if(body!=null)
        {
         var d = dist(body.position.x,body.position.y,sprite.position.x,sprite.position.y);
          if(d<=x)
            {
               return true;
            }
            else{
              return false;
            }
         }
}
```

You can call this function to check the collision between the star and the fruit.

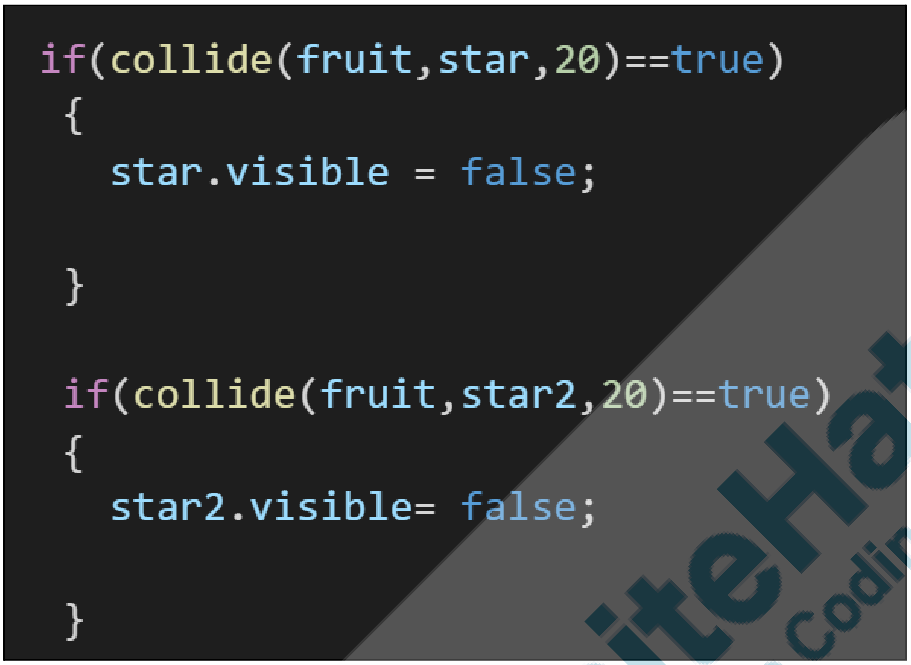
Next, we also need to think about what will happen after the collision.
What do you think should happen?

**ESR:**
Varied

Once the fruit hits the star we will make the star disappear, by changing its visibility.
Do you remember how we can change the visibility of a

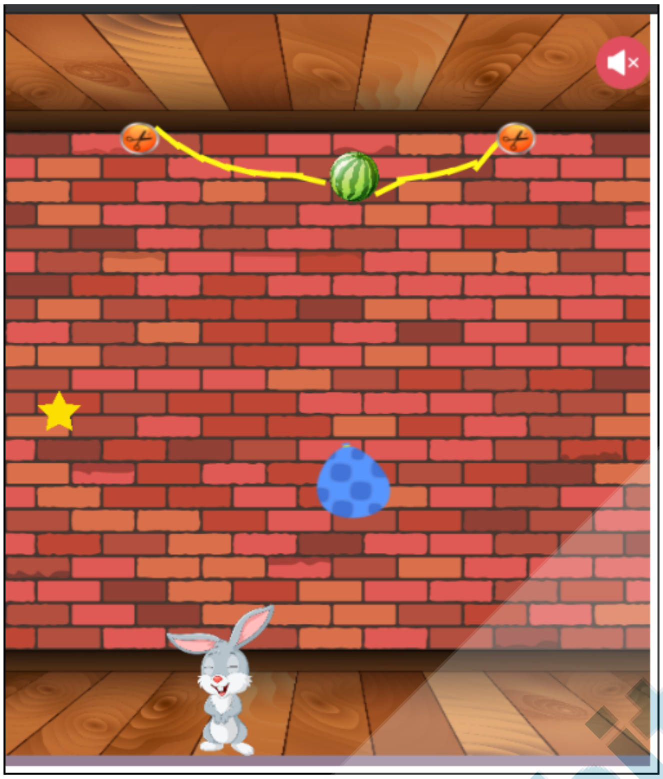
**ESR:** We can do that by

| | |
|---|---|
| sprite? | changing the visible property of sprite to false. |



```
if(collide(fruit,star,20)==true)
 {
   star.visible = false;

 }

 if(collide(fruit,star2,20)==true)
 {
   star2.visible= false;

 }
```

| | |
|---|---|
| Let’s run the code to see the output. | *Student runs the code and clicks on the balloon button to collect the upper star, then breaks the right rope to collect the left star.* |



| | |
|---|---|
| Well done!<br>You have done a great job.<br><br>So let's move to your next challenge, adding a score for the game.<br><br>How do we start? | **ESR:** We can declare a variable score and increase it whenever a star is collected. |
| Superb!<br>You must have seen in games that when you collect the star, you have a star counter at the top which shows how many stars you have collected. | **ESR:** Yes/ No |
| Let's add that to our game.<br>But before we start coding, let's understand the logic behind it.<br>We will create a sprite and it will have 3 animations.<br>● One with an empty star<br>● Second with one star<br>● Third with two stars | |

Then based on how many stars the users have collected we will switch the animation.

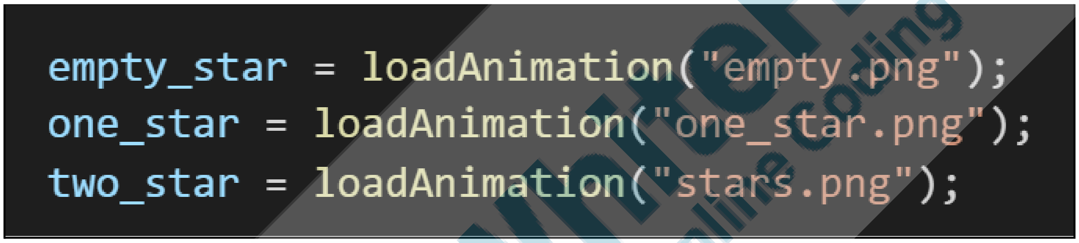
First, we load the animation in the **preload()** function and then we create the sprite and add the animation to that sprite.

We create a sprite as **star_display.**

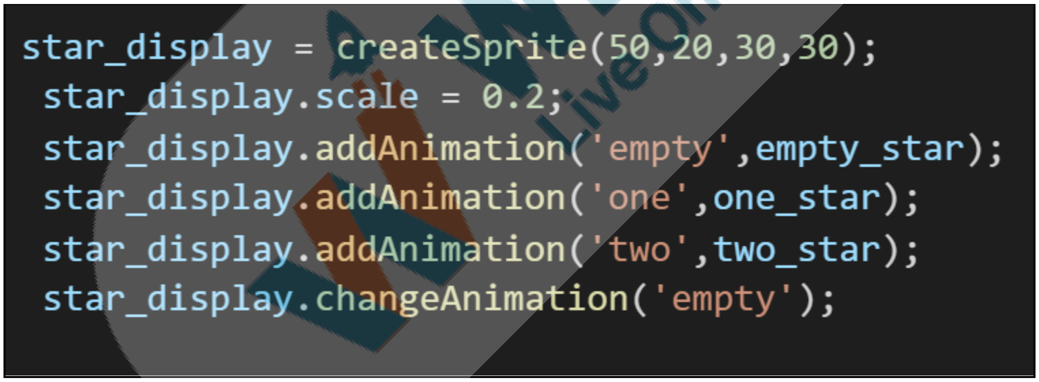
Set the **scale** and add the animation.

When the game starts it will have an empty animation.

We will change the animation when the fruit collides with the star.

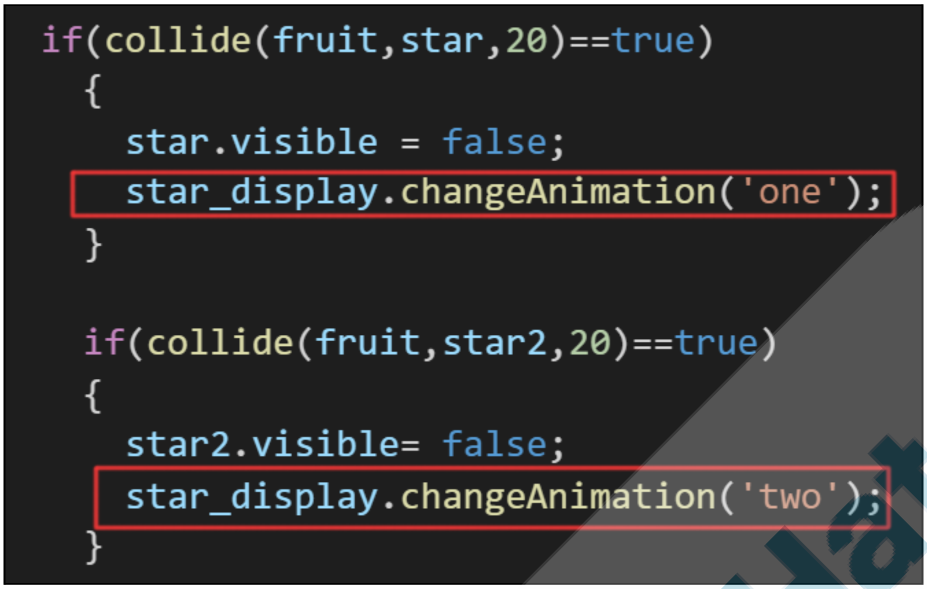


```
empty_star = loadAnimation("empty.png");
one_star = loadAnimation("one_star.png");
two_star = loadAnimation("stars.png");
```



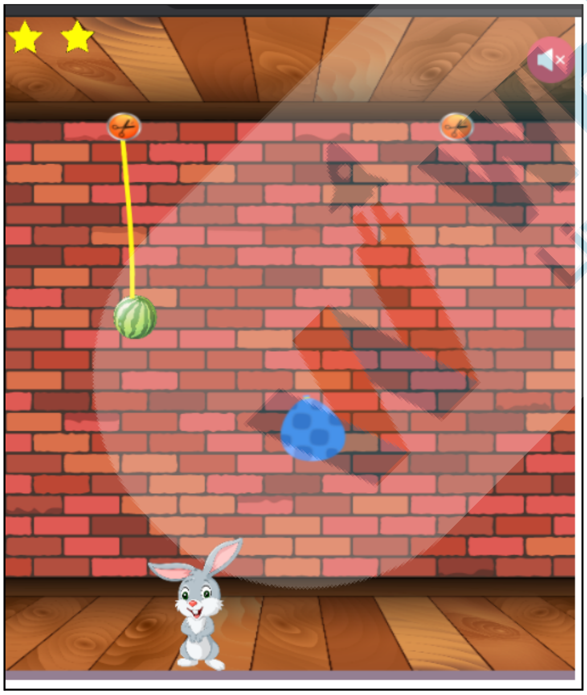
```
star_display = createSprite(50,20,30,30);
 star_display.scale = 0.2;
 star_display.addAnimation('empty',empty_star);
 star_display.addAnimation('one',one_star);
 star_display.addAnimation('two',two_star);
 star_display.changeAnimation('empty');
```

Now we will change the animation where we detect the collision.



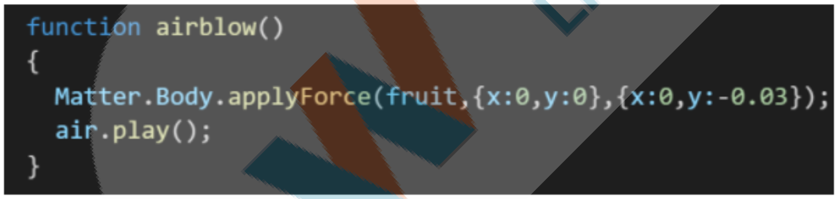
```
if(collide(fruit,star,20)==true)
  {
    star.visible = false;
    star_display.changeAnimation('one');
  }

  if(collide(fruit,star2,20)==true)
  {
    star2.visible= false;
    star_display.changeAnimation('two');
  }
```

| | |
|---|---|
| Now let’s run the game to see the output. | *Student runs the code.* |



| | |
|---|---|
| Very well done!<br>We have made a really fun game. | |

| | |
|---|---|
| In this class, we have changed the arrangement of the rope, added stars, changed the collision algorithm, and built a star counter.<br>You can also make this game mobile compatible and host it on the GitHub servers.<br>Then make a mobile downloadable app using the thunkable platform just like we did in the last class. | *The student uploads the game on GitHub, publishes the code, and uses thunkable to create an apk file.* |

| **Teacher Guides Student to Stop Screen Share** | |
|---|---|
| **Quiz time - Click on in-class quiz** | |
| **Question** | **Answer** |
| Which instruction do we use to scale down the image of the star?<br>A. star.scaleDown = 0.02<br>B. star.scale = 0.02<br>C. star.resize = 0.02<br>D. star.scale = 20 | **B** |
| What does the following code snippet do?<br><br>function airblow()<br>{<br>  Matter.Body.applyForce(fruit,{x:0,y:0},{x:0,y:-0.03});<br>  air.play();<br>}<br>A. Defines the function to apply force on the fruit.<br>B. Adds a balloon image.<br>C. Moves the bunny to a new position.<br>D. Moves the bunny to the right. | **A** |
| Which of the following instructions will make the star disappear after collision with the fruit?<br>A.if(collide(fruit, star, 20) == true)<br>{<br>star.invisible = false; | **D** |

```
        }
    B.if(collide(fruit, star, 20) = true)
        {
          star.visible = false;
        }
    C.if(collide == true)
        {
          star.visible = false;
        }
    D.if(collide(fruit, star, 20) == true)
        {
          star.visible = false;
        }
```

**End the quiz panel**

**WRAP UP SESSION - 5 Mins**

**Teacher starts slideshow from slide 14 to slide 23**

| Activity details | **Solution/Guidelines** |
|---|---|
| **Run the presentation from slide 14 to slide 23**<br>**Following are the wrap up session deliverables:**<br>● **Explain the facts and trivia**<br>● **Next class challenge**<br>● **Project for the day**<br>● **Additional Activity** | *Guide the student to develop the project and share it with us.* |

**Teacher ends slideshow**

**ADDITIONAL ACTIVITIES**

- **Ask Student to press ESC key to come back to the panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

| | | |
|---|---|---|
| | You get Hats Off for your excellent work! | *Make sure you have given at least 2 Hats Off during the class for:*<br> Creatively Solved Activities<br>+10<br>Great Question<br>+10<br>Strong Concentration<br>+10 |
| **Project Overview**<br>**Your Own Game with Physics Engine**<br>In the last few classes, you have learned how to add a physics engine in the javascript games to make it look more attractive.<br>In this project, you will apply what you have learned in the classes to create a game using a physics engine.<br>**Story:**<br>You have got a chance to participate in a Game Creating Coding Contest. You need to submit your creation which uses the OOP concept and physics engine.<br>I am very excited to see your project solution and I know you will do really well.<br>Bye Bye! | | **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.**<br>*Students engage with the teacher over the project.* |

| Activity | Description | Link |
|---|---|---|
| Teacher Activity 1 | Working Game | https://pro-whitehatjr.github.io/c34-game.github.io/ |
| Student Activity 1 | Template Code | https://github.com/pro-whitehatjr/C34_template_code |
| Teacher Activity 2 | Reference Code | https://github.com/pro-whitehatjr/C34-solution |
| Teacher Reference visual aid link | Visual aid link | https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C34-withcues.html |
| Teacher Reference In-class quiz | In-class quiz | https://s3-whjr-curriculum-uploads.whjr.online/488b6d4d-f537-4e47-92b6-18498b6df6c1.pdf |
| Teacher Reference | Project booster LP | https://s3-whjr-v2-prod-bucket.whjr.online/32dad903-b495-40ea-bfdd-a49ade21168d.pdf |
| Teacher Reference VA | Booster VA | https://curriculum.whitehatjr.com/Visual+Project+Asset/Boosterclass_withoutcues.html |
| Teacher Reference Video | Scaffolding Techniques | https://www.youtube.com/watch?v=0kB6pNmLoM8 |