WhiteHat Jr
Live online Coding for Kids

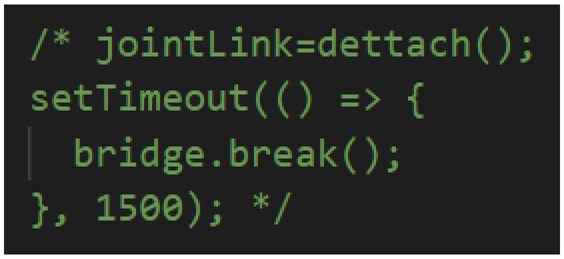
| | | |
|---|---|---|
| **Topic** | **COLLISION DETECTION AND ANIMATION** | |
| **Class Description** | **The student learns to detect the collision of the fruit with the bunny and adds different animations to the bunny. The student will also learn to change the animation based on different conditions.** | |
| **Class** | **C31** | |
| **Class time** | **55 mins** | |
| **Goal** | ● Detect collision of the fruit with the bunny.<br>● Add animation to the bunny sprite. | |
| **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 - Slide show option**<br>**Teacher-Led Activity**<br>**Student-Led Activity**<br>**Wrap-Up - Slide show option** | **5 Mins**<br>**15 Mins**<br>**30 Mins**<br>**5 Mins** |

**WARM-UP SESSION - 5 mins**

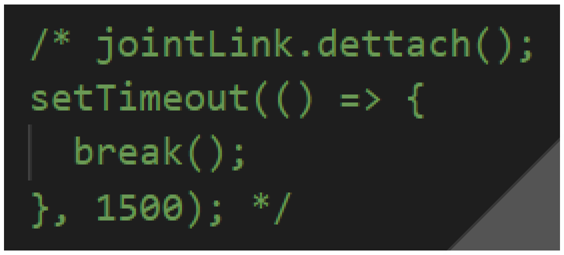
**Teacher starts slideshow from slides 1 to 10**
Refer to speaker notes and follow the instructions on each slide.

| **Activity details** | **Solution/Guidelines** |
|---|---|

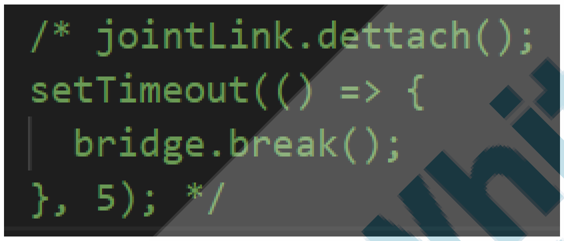
| | |
|---|---|
| *Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?* ***Run the presentation from slide 1 to slide 3.*** **The following are the warm-up session deliverables:** • **Greet the student.** • **Revision of previous class activity.** • **Quiz** | **ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides |
| **QnA Session** | |
| **Question** | **Answer** |
| Select the correct option to call the **handleButtonPress()** function on the **mousePressed** property of the button. A. `//breakButton.mouseClicked(handleButtonPress);` **B.** `//breakButton.mousePressed(handleButtonPress);` C. `//breakButton.mouse(handleButtonPress);` D. `//breakButton.mousePressed(ButtonPress);` | B |
| Select the correct option to **detach()** the **jointLink** and use **setTimeout()** function to call the **bridge.break()** after 5 seconds. | D. |

A. 

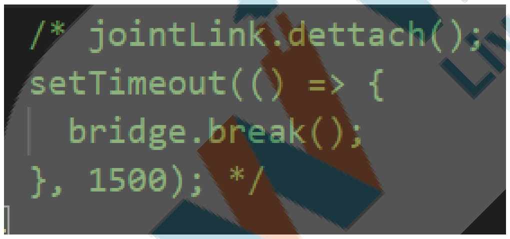
```
/* jointLink=dettach();
setTimeout(() => {
  bridge.break();
}, 1500); */
```

B. 

```
/* jointLink.dettach();
setTimeout(() => {
  break();
}, 1500); */
```

C. 

```
/* jointLink.dettach();
setTimeout(() => {
  bridge.break();
}, 5); */
```

D. 

```
/* jointLink.dettach();
setTimeout(() => {
  bridge.break();
}, 1500); */
```

**Continue the warm-up session**

| Activity details | Solution/Guidelines |
|---|---|
| ***Run the presentation from slide 4 to slide 10 to set the problem statement.***<br><br>**The following are the warm-up session deliverables:** | Narrate the slides by using hand gestures and voice modulation methods to bring in more |

| | |
|---|---|
| ● Appreciate the student on his performance in the quizzes<br>● Create an animation for the bunny. | interest in students. |
| **Teacher ends slideshow** | |
| **TEACHER-LED ACTIVITY - 15 mins** | |
| **Teacher Initiates Screen Share** | |
| **<u>CHALLENGE</u>**<br><br>● **Load animations in the preload() function.**<br>● **Add animation in the bunny Sprite.**<br>● **Update the Animation play speed.** | |
| **Teacher Action** | **Student Action** |
| *Teacher downloads [Teacher Activity 1]() from GitHub and runs it in the VS Code interface.*<br><br>In the previous class, we added a button to cut the rope and added the images in the **preload()** function.<br><br>In this class we are going to:<br><br>● Add animations to our bunny sprite.<br>● We will also detect the collision of the fruit with the bunny.<br>● We will also change the animation based on different conditions.<br><br>First, we will load the animation in the **preload()** function just like we loaded the images in the previous class.<br><br>Animation consists of multiple images, if we play these images continuously it looks like an animation.<br><br>In the **preload()** function we will load the animation with help of the **loadAnimation()** function, the argument for the | |

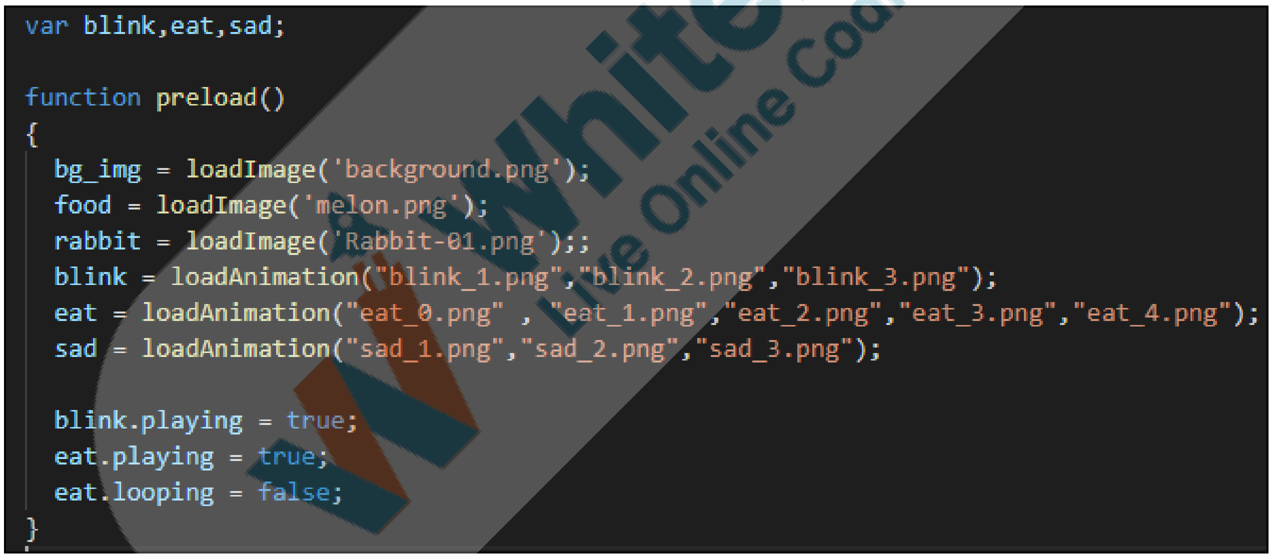
function is going to be the image.

Before we load the animation, let’s create variables in which the animations will be stored.
**var blink;** and,
**var eat;**

We will play only **two** animations, for now, **blink** and **eat**, the bunny blinks its eyes, and once the fruit falls and collides with the bunny it will start eating, after which we will play the eating animation.

In the **preload()** function, assign the **loadAnimation()** function to the **blink** and **eat** variables and pass the corresponding images.

*The teacher writes code as shown below.*



```
var blink,eat,sad;

function preload()
{
  bg_img = loadImage('background.png');
  food = loadImage('melon.png');
  rabbit = loadImage('Rabbit-01.png');;
  blink = loadAnimation("blink_1.png","blink_2.png","blink_3.png");
  eat = loadAnimation("eat_0.png" , "eat_1.png","eat_2.png","eat_3.png","eat_4.png");
  sad = loadAnimation("sad_1.png","sad_2.png","sad_3.png");

  blink.playing = true;
  eat.playing = true;
  eat.looping = false;
}
```

Once we loaded the animation, we needed to set some properties of the animation.
First, we need to set both animations as **playing** equal to **true**.
This will enable the play mode of the animation. This makes our animation when we run the code. If we set this to false, then our animation will not run until we specify that in the code.

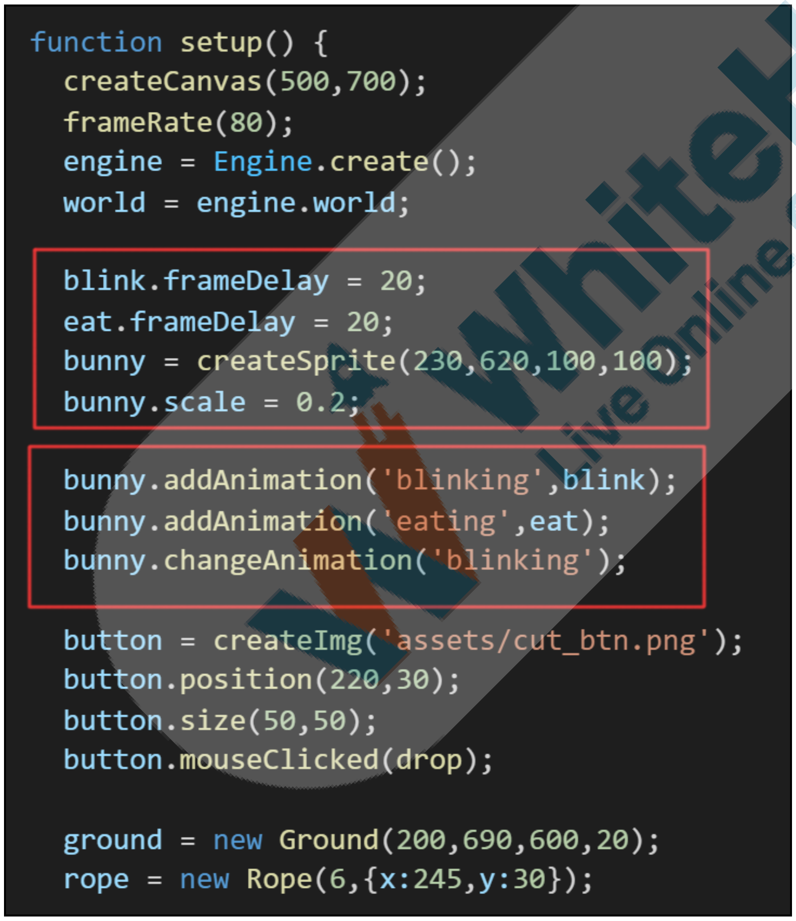
| | |
|---|---|
| Next for the **eat** animation we don't want the animation to loop or play again and again, so to prevent that we will keep **eat.looping = false**. | |
| We have loaded the animation, now we need to add this animation to our bunny sprite so that we can switch between animations.<br><br>But we need to do one more thing before that, which is to set the speed of the animation.<br><br>The computer tries to play the animation as fast as possible, but we want our animation to play a little slower so that we can see what is happening in the animation.<br><br>To slow the speed of the animation we need to set a frame delay. This is going to be a number, the higher the number the slower will be the speed.<br><br>We will set this for both the loaded animations, blink and eat.<br>**blink.frameDelay = 20;**<br>**eat.frameDelay = 20;**<br>We can keep any value between **15** and **20**, that will look good for our animation.<br><br>**Note:** The teacher can show the effect of the value on the speed of the animation by changing it to **5** and then keeping it at **20**.<br><br>Now we will add this animation to our bunny sprite using the **addAnimation()** function.<br><br>In the parameters of the function, we will pass a string value as the first argument that will identify the animation, you can think of it as the name of the animation,<br>then the second argument will be loading the animation.<br><br>So the syntax will look like this: | |

***bunny.addAnimation('blinking',blink);***

A string could be anything, but it is good practice to keep it relevant.
Similarly, we will add the animation for eating as well.

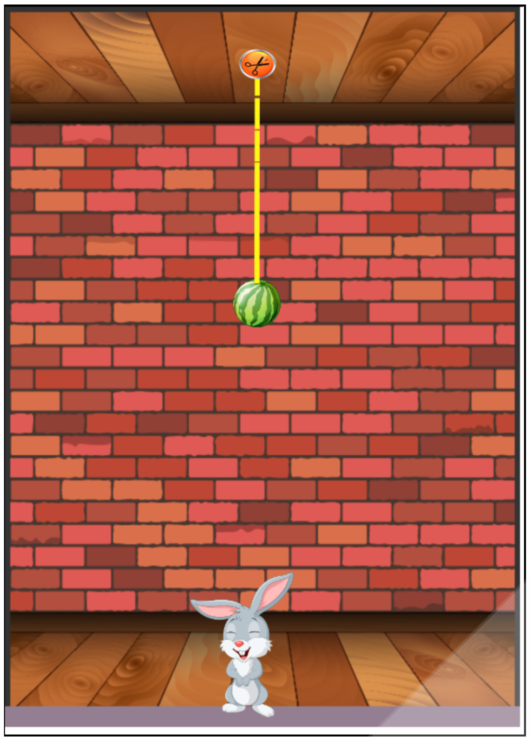
We will also change the animation to blinking by using the **bunny.changeAnimation()** function as shown ahead.

Because in the beginning, we want to play the blinking animation. Which means when our code runs, the bunny will be playing the blinking animation to show that the bunny is waiting for the fruit.



```
function setup() {
  createCanvas(500,700);
  frameRate(80);
  engine = Engine.create();
  world = engine.world;

  blink.frameDelay = 20;
  eat.frameDelay = 20;
  bunny = createSprite(230,620,100,100);
  bunny.scale = 0.2;

  bunny.addAnimation('blinking',blink);
  bunny.addAnimation('eating',eat);
  bunny.changeAnimation('blinking');

  button = createImg('assets/cut_btn.png');
  button.position(220,30);
  button.size(50,50);
  button.mouseClicked(drop);

  ground = new Ground(200,690,600,20);
  rope = new Rope(6,{x:245,y:30});
```

Output:

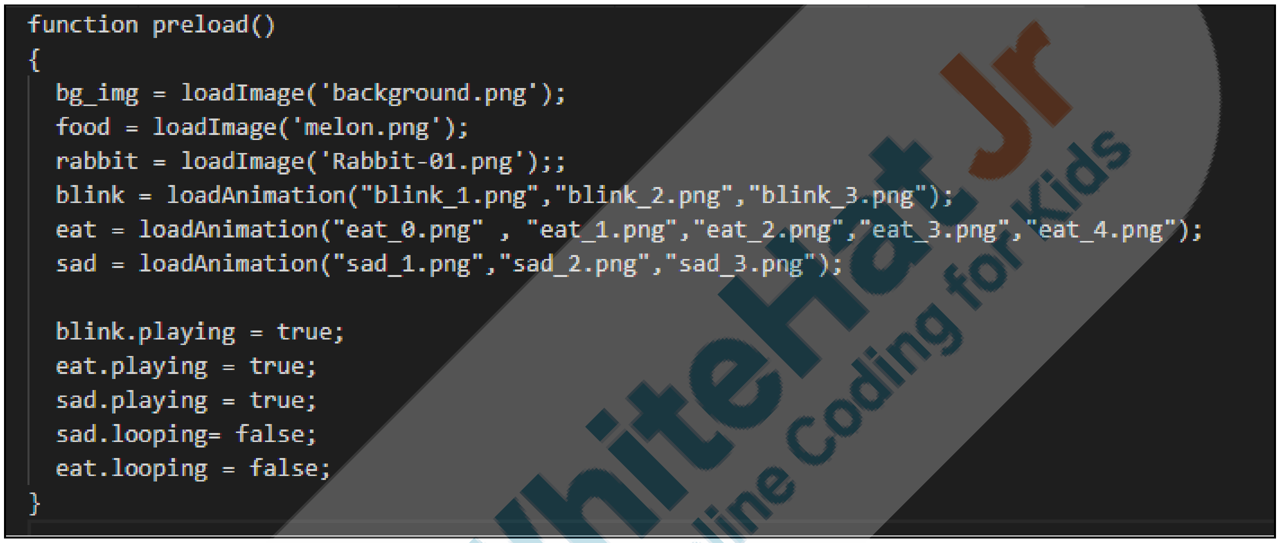
| | |
|---|---|
|  | |
| Run the code to see the output. The bunny is playing the blinking animation.<br><br>Apart from the blinking animation, we need animation for the **happy bunny** and the **bunny being sad**.<br><br>We will use the sad animation only when the bunny misses eating the fruit.<br><br>Just like, we added the animation of blinking and eating.<br><br>You need to add the animation for the happy and sad. | |
| Now it's your turn.<br><br>Are you excited to add life to our bunny?<br><br>Please share your screen with me. | **ESR:** Yes! |

| Teacher starts slideshow : Slide 11 to Slide 19 | |
|---|---|
| **Run the presentation slide to set the student activity context.** | |
| • Create sad bunny animation.<br>• Code to detect the collision between the bunny and the fruit. | |

| **Teacher ends slideshow** |
|---|
| **Teacher Stops Screen Share** |
| **STUDENT-LED ACTIVITY - 30 mins** |
| • **Ask Student to press ESC key to come back to panel**<br>• **Guide Student to start Screen Share**<br>• **Teacher gets into Fullscreen** |
| **ACTIVITY**<br>• **Add the animation of eating and the sad bunny.**<br>• **Detect the collision of fruit with the bunny.**<br>• **Remove the fruit body once it collides with the bunny.**<br>• **Change the bunny animation based on collision** |

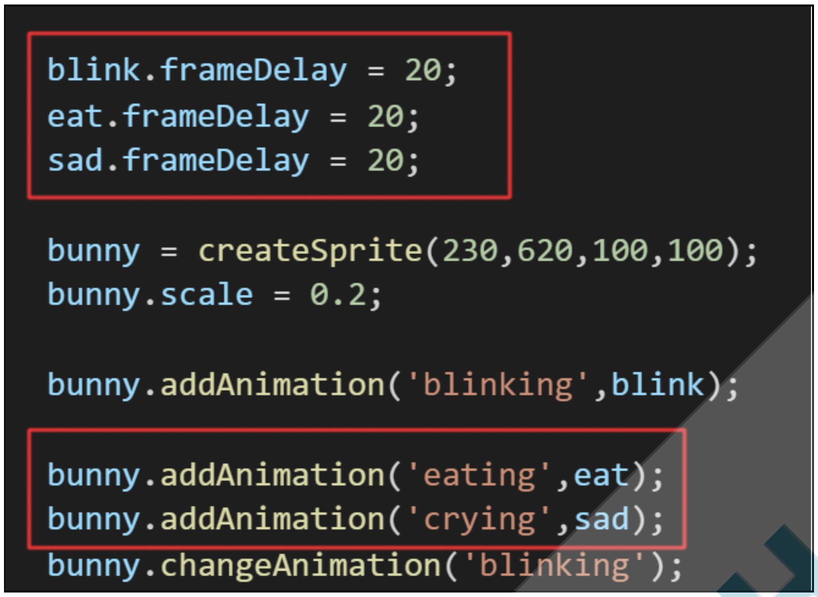
| **Teacher Action** | **Student Action** |
|---|---|
| *The teacher guides the student to download the code from GitHub and run it in the VS Code Editor.*<br><br>Just like we added the animation of blinking. We follow the same process for eating and the sad animation as well.<br><br>First, declare variables to store the animation and then | *The student downloads the [Student Activity 1](#) code from GitHub and opens it in the VS Code editor.* |

| | |
|---|---|
| load the animation in the **preload()** function.<br><br>Apart from loading the animation, we need to set them as **playing = true** for playing the animations for sad and the eating bunny however, we don't want the eating and sad animation to be played over and over again.<br>So we set their **looping** as **false** as shown below: | |



```
function preload()
{
  bg_img = loadImage('background.png');
  food = loadImage('melon.png');
  rabbit = loadImage('Rabbit-01.png');;
  blink = loadAnimation("blink_1.png","blink_2.png","blink_3.png");
  eat = loadAnimation("eat_0.png" , "eat_1.png","eat_2.png","eat_3.png","eat_4.png");
  sad = loadAnimation("sad_1.png","sad_2.png","sad_3.png");

  blink.playing = true;
  eat.playing = true;
  sad.playing = true;
  sad.looping= false;
  eat.looping = false;
}
```

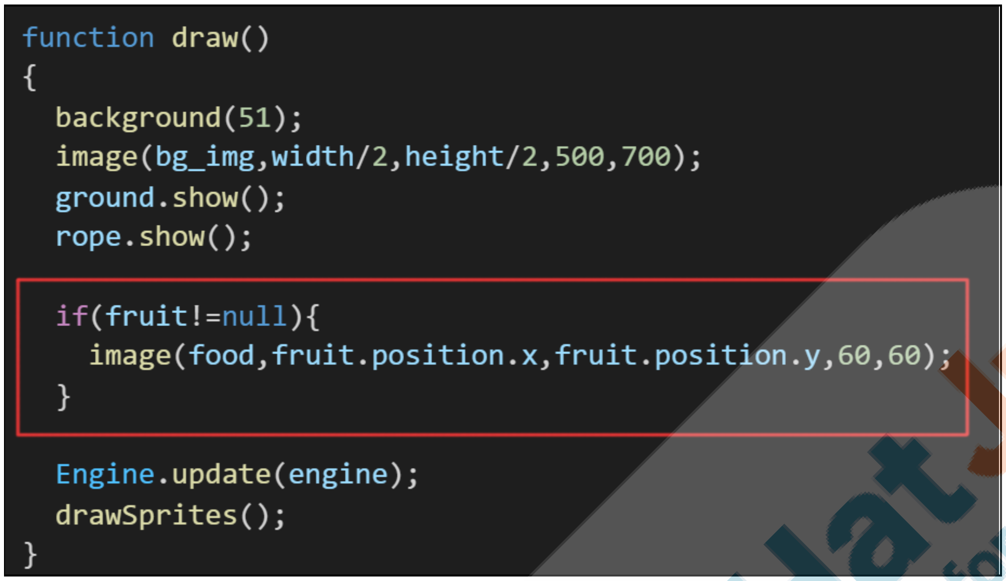
| | |
|---|---|
| Once we have loaded the animation, we will now add this animation to our bunny sprite.<br><br>Do you know how we can set the animation speed?<br><br>Yes, that was one way to achieve it, but today, we will see another concept to control the speed of the animation. It is called **frameDelay**.<br><br>As animations are created using frames; we can change the number for **frameDelay**. | **ESR**: Varied; student may recall **animate()** from Pirate Invasion game.<br><br>*Let the student try different values to see the effect of **frameDelay.*** |



```
blink.frameDelay = 20;
eat.frameDelay = 20;
sad.frameDelay = 20;

bunny = createSprite(230,620,100,100);
bunny.scale = 0.2;

bunny.addAnimation('blinking',blink);

bunny.addAnimation('eating',eat);
bunny.addAnimation('crying',sad);
bunny.changeAnimation('blinking');
```

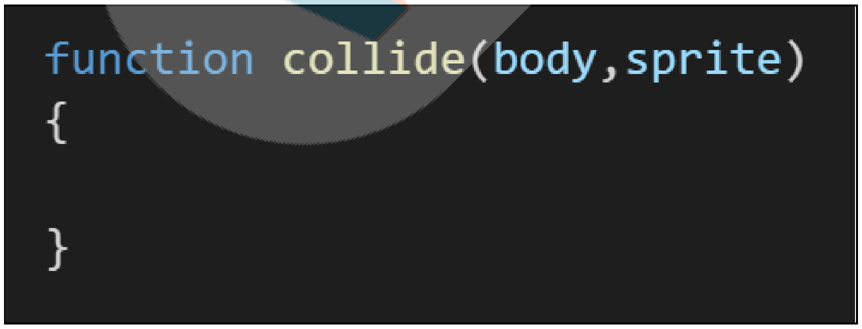
| | |
|---|---|
| Animation is done.<br><br>What should we do next?<br><br>*The teacher can then inform the student, to have the eating and sad animation playing we need to detect the fruit which is the next thing we will be doing.*<br><br>Yes, you will now detect the collision of the fruit with the bunny and then change the animations accordingly.<br><br>You will create a function that will detect the collision between the fruit body and the bunny.<br><br>To detect the collision, you are going to use a very simple algorithm. Which is to find the distance between fruit and bunny.<br><br>As the fruit moves toward the bunny, the distance decreases. We can write the condition in such a way that if the distance is less than a certain value we can say both the objects have collided. | *The student runs and checks the output for each value he/she enters.*<br><br>**ESR**: We need to see if the bunny is touching the fruit, or if the bunny misses it. |

| | |
|---|---|
| ***Note**: The teacher can explain this using two props/hands to show how distance reduces when palms/objects come closer and vice-versa.*<br><br>This is the overview of the function you are going to create.<br>But here are some important details which we have to pay attention to, otherwise our program will give us multiple errors. The details are as follows:<br><br>● Once the fruit falls down, we need to remove the fruit from the world. But if we remove the fruit then we can not calculate the distance between the fruit and the bunny, so we need to put the condition in such a way, it will only calculate the distance if the fruit body exists.<br><br>● The second important thing is if we remove the fruit from the world then we can't draw it. Because we have deleted the body from the world, and while drawing we are referencing the x and y position of the fruit body. If a fruit body does not exist, and we are still drawing it on the canvas, then it will give us an error.<br><br>Let's first add the code to resolve these issues.<br><br>We only want to show the fruit if its body exists.<br>So we write the code as:<br><br>**if(fruit!=null)**<br>**{**<br>**Show the fruit body.**<br>**}** | |



```
function draw()
{
  background(51);
  image(bg_img,width/2,height/2,500,700);
  ground.show();
  rope.show();

  if(fruit!=null){
    image(food,fruit.position.x,fruit.position.y,60,60);
  }

  Engine.update(engine);
  drawSprites();
}
```

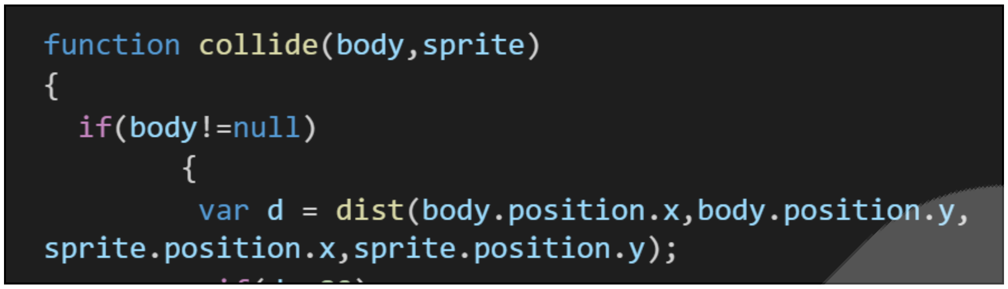
| | |
|---|---|
| This code will ensure that it will only show the fruit body, if it exists, if we delete the fruit body, it will not give us any error. | |
| Now comes the tricky part, where we will create a function to detect the collision.<br><br>Let's first define the function as **collide(body,sprite)**, this will take two arguments one is the body second is the sprite.<br><br>**Note**: We can pass the two bodies as well, the name is only kept for understanding.<br>In this case, it will be the fruit and bunny. | |



```
function collide(body,sprite)
{

}
```

| | |
|---|---|
| In the function first, we need to check whether the body | |

| | |
|---|---|
| exists or not.<br>The body here is the fruit body.<br><br>So we will test this by using the **not equal to (!=)** operator, in the condition, we will write **if(fruit!= null)**.<br><br>Do you know what is the **not equal to** operator?<br><br>If you recall in Class 7 where we learned operators, the **Not equal to** operator is used to **test whether a variable is not equal to a certain value**. Here we want to check if our fruit is not equal to null, then only we will proceed further to detect collision.<br>If the fruit is null then we can not detect collision.<br><br>Then in the parenthesis of this condition, we need to write the algorithm to detect the collision.<br><br>When can we say that 2 bodies have collided?<br><br>Yes, we can say that when the distance between 2 bodies is **0** we can say they have collided.<br><br>We will use the same logic here, we need to find the distance between the fruit and the bunny.<br>That we will do using the **dist()** function, which is an in-built function of the **p5.js** library.<br><br>This function needs four points:<br>● X position of first body or shape.<br>● Y position of first body or shape then,<br>● X position of the second body or shape.<br>● Y position of the second body or shape.<br><br>It will then calculate the distance between two points.<br><br>We will need a variable to store this value, so we can declare a variable here and assign the **dist()** function to it. | **ESR:**<br>Varied<br><br>**ESR:**<br>When they touch each other or come very close. |



```
function collide(body,sprite)
{
  if(body!=null)
        {
         var d = dist(body.position.x,body.position.y,
sprite.position.x,sprite.position.y);
```

Once we have the distance between the fruit and the bunny, it then becomes very easy.

We just need to write a condition that if the distance is less than or equal to some value we say, it would cause a collision.
In our case, we will set it as **80**.

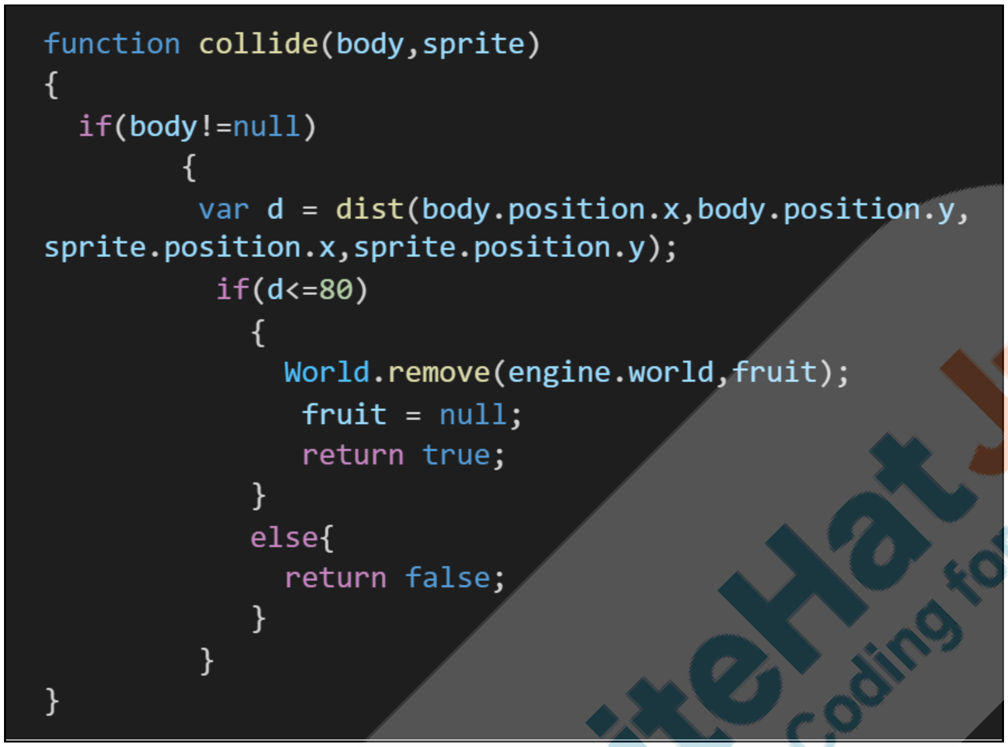
If this condition is **true** then we need to remove the fruit from the world and make it **null**. Because when the fruit collides with the bunny it will eat the fruit, so it is supposed to disappear from the scene.

For that, we will use the **World.remove()** function and then set the fruit **null**.

We also need to return **true** as a **callback**.

If this condition is not satisfied then we will simply return **false**.

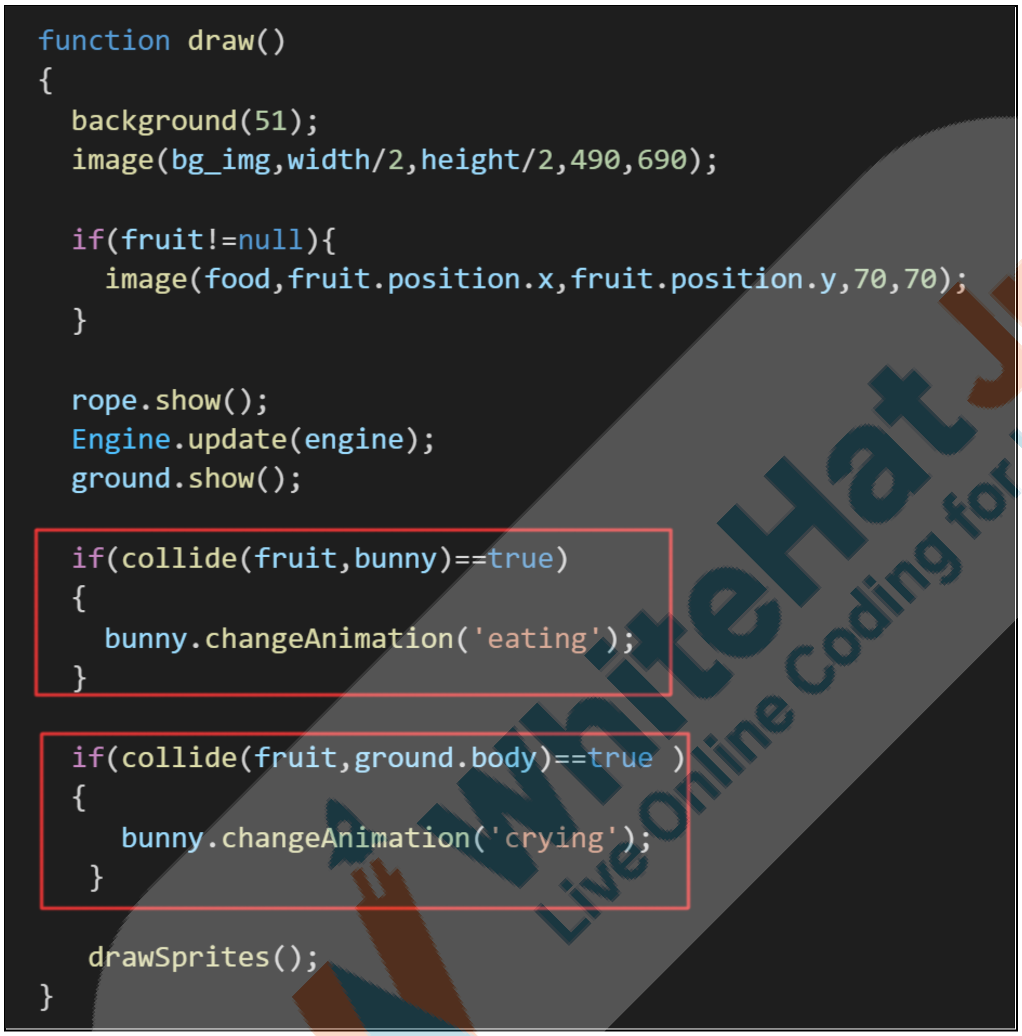
And the fruit will stay in the scene.



```
function collide(body,sprite)
{
  if(body!=null)
        {
         var d = dist(body.position.x,body.position.y,
sprite.position.x,sprite.position.y);
          if(d<=80)
            {
              World.remove(engine.world,fruit);
               fruit = null;
               return true;
            }
            else{
              return false;
            }
          }
}
```

| | |
|---|---|
| We will call this function in the **draw()** loop and if this function returns true we will change the animation to eating.<br><br>We will also detect the collision with the ground. If the fruit falls on the ground, not colliding with the bunny. Then we will play the sad animation.<br><br>**Note:** In the coming classes we will add an air balloon that pushes the fruit in a direction, and we will also add multiple ropes to hold the fruit, where the user has to cut the rope in such a way that the fruit falls directly on the bunny. | |
| We will call the **collide()** function in the **draw()** loop.<br><br>If the fruit collides with the bunny we will change the animation to eating and if the fruit collides with the ground we will change the animation to sad. | |
| Here our fruit may not collide with the ground, because we kept it directly on the bunny. But in future classes, we can | |

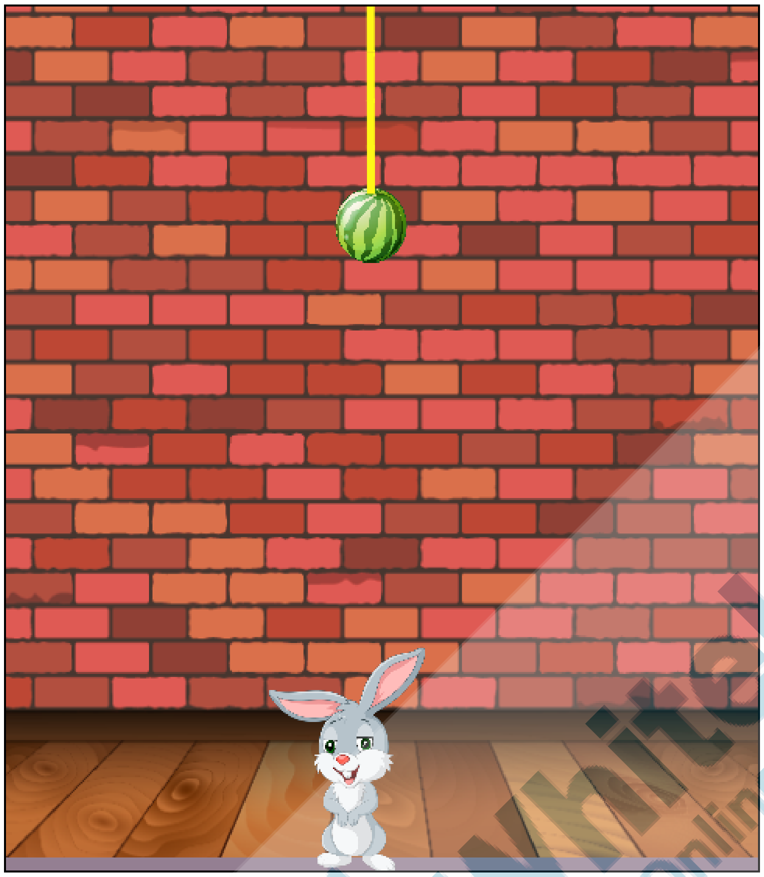
| | |
|---|---|
| have such circumstances. | |



```
function draw()
{
  background(51);
  image(bg_img,width/2,height/2,490,690);

  if(fruit!=null){
    image(food,fruit.position.x,fruit.position.y,70,70);
  }

  rope.show();
  Engine.update(engine);
  ground.show();

  if(collide(fruit,bunny)==true)
  {
    bunny.changeAnimation('eating');
  }

  if(collide(fruit,ground.body)==true )
  {
     bunny.changeAnimation('crying');
   }

   drawSprites();
}
```

| | |
|---|---|
| We are calling the collide function two times, one for **fruit** and **bunny**, the other for **fruit** and **ground.body**.<br><br>Based on the result of the collision of fruits with the bunny we will change the animation to happy and with the collision of fruit with the ground, it should be sad.<br><br>***Note****: It is very important to pass the* ***ground.body*** *otherwise the program will throw an error.* | |

WhiteHat Jr
Live online Coding for Kids

| | |
|---|---|
|  | |
| This brings us to the end of this class. How did you like today's class?<br><br>In the next class we are going to add sounds in the game. To increase the challenge, we will also add a balloon which will push the fruit when the user clicks on it. This will help us to drop the fruit on the bunny even when the fruit is not directly above the bunny. | **ESR**: Varied |
| **Teacher Guides Student to Stop Screen Share** | |
| **WRAP-UP SESSION - 5 Mins** | |
| **Teacher starts slideshow** **from slide 20 to slide 29** | |

| Activity details | **Solution/Guidelines** |
|---|---|
| ***Run the presentation from slide 20 to slide 29.***<br><br>**Following are the wrap-up session deliverables:**<br>● **Appreciate the student.**<br>● **Revise the current class activities.**<br>● **Discuss the quizzes.** | Guide the student to develop the project and share it with us. |
| **Quiz time - Click on in-class quiz** | |
| **Question** | **Answer** |
| Which of the following functions is used to load the animation in the game?<br><br>A. loadImage()<br>B. loadAnimation()<br>C. load.animation()<br>D. Loadanimation() | **B** |
| To slow the speed of the animation, which property do we use?<br><br>A. animation.speed<br>B. animation.slow<br>C. animation.frameDelay<br>D. animation.delayFrame | **C** |
| What are the parameters of the addAnimation() function?<br><br>A. sprite.addAnimation(animation,label);<br>B. animation.addAnimation(label,animation);<br>C. sprite.addAnimation(animation);<br>D. sprite.addAnimation(label,animation); | **D** |
| **End the quiz panel** | |
| **<u>FEEDBACK</u>**<br>● **Encourage the student to make reflection notes.**<br>● **Compliment the student for her/his effort in the class.** | |

| ● **Review the content of the lesson.** | | |
|---|---|---|
| | You get hats off.<br><br>See you in the next class then. | 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 |
| *** 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><br>CRUSH THE ZOMBIES - 3<br><br>**Goal of the Project:**<br><br>In class 31 we learned to detect the collision between the bunny and the fruit using the **dist()** function. In this project, we will detect the collision between the stones and zombies. Set the sad zombie image when the distance between the zombie and stone is less than 50.<br><br>**Story**:<br><br>A far away village is always troubled by zombies. The only way to kill the zombie is to drop a stone in its head. You have mostly seen that the zombie travels under the bridge to get to the village, so you plan to stack the bridge with the stones and drop it on the zombie when it | | *Students engage with the teacher over the project.* |

| | |
|---|---|
| comes under the bridge.<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** | |

**Links**:

| Activity | Description | Link |
|---|---|---|
| Teacher Activity 1 | Previous Class Code | https://github.com/pro-whitehatjr/C31_TA1 |
| Student activity 1 | Activity Link | https://github.com/pro-whitehatjr/C31_SA1 |
| Teacher Activity 2 | Reference link | https://github.com/pro-whitehatjr/C31_Complete_game |
| Project Solution | Crush the Zombies - 3 | https://github.com/whitehatjr/zombie-crush-3 |
| Teacher Reference visual aid link | Visual aid link | https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/slides-deck-31_withcues.html |
| Teacher Reference In-class quiz | In-class quiz | https://s3-whjr-curriculum-uploads.whjr.online/190f6215-35dd-4945-8fd9-2658bfab1270.pdf |
| Project Solution | Crush The Zombies-3 | https://github.com/whitehatjr/zombie-crush-3 |