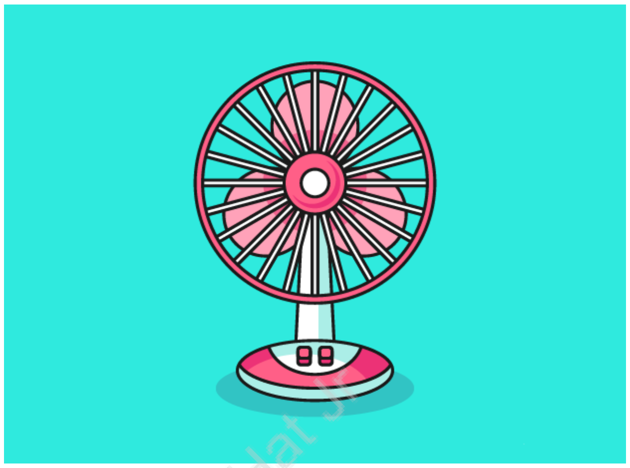
ADDING SOUNDS AND CHALLENGES



What is our GOAL for this MODULE?

In this class, we learned to, add sound and air blower features to the game in such a way that the air blower pushes the hanging fruit towards the direction of the bunny.

What did we ACHIEVE in the class TODAY?

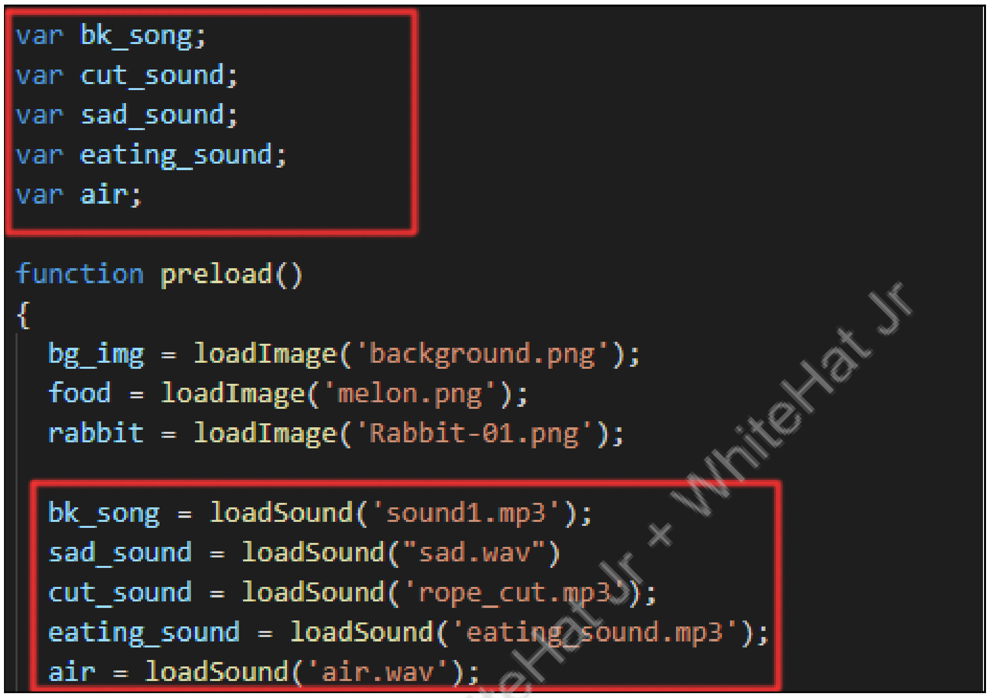
- Added background music to the game.
- Added sound effects for cutting, eating, and the air blower.
- Created the air blow effect on the fruit.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- About the **loadSound()** function.
- To apply necessary force on the fruit sprite.
- To add background music to the game.

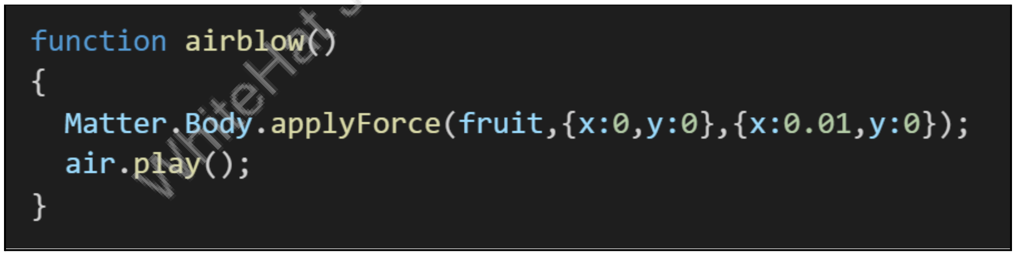
How did we DO the activities?

1. Load all the sound files in the code as shown in the below code snippet:



```
var bk_song;
var cut_sound;
var sad_sound;
var eating_sound;
var air;

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

  bk_song = loadSound('sound1.mp3');
  sad_sound = loadSound("sad.wav")
  cut_sound = loadSound('rope_cut.mp3');
  eating_sound = loadSound('eating_sound.mp3');
  air = loadSound('air.wav');
```

2. Write the **airblow()** function to apply force on the fruit and play the air sound effect when this function is executed.

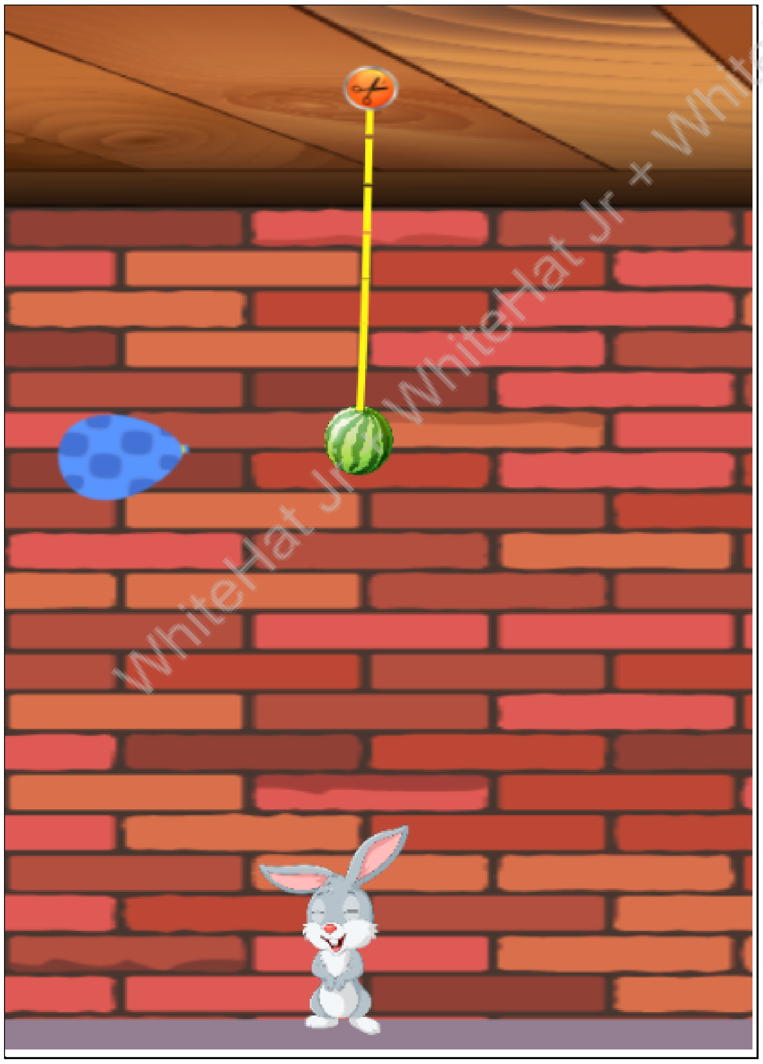


```
function airblow()
{
  Matter.Body.applyForce(fruit,{x:0,y:0},{x:0.01,y:0});
  air.play();
}
```

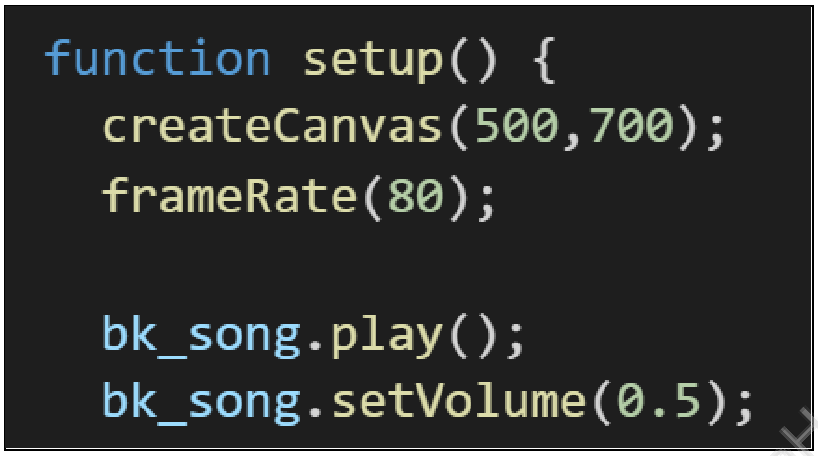
3. Now create the image button of the air-blower and also set its position while attaching the **airblow()** function with this **blower** button.

```
blower = createImg('assets/balloon.png');
blower.position(10,250);
blower.size(150,100);
blower.mouseClicked(airblow);
```

4. Run the code to see the air-blower button output.



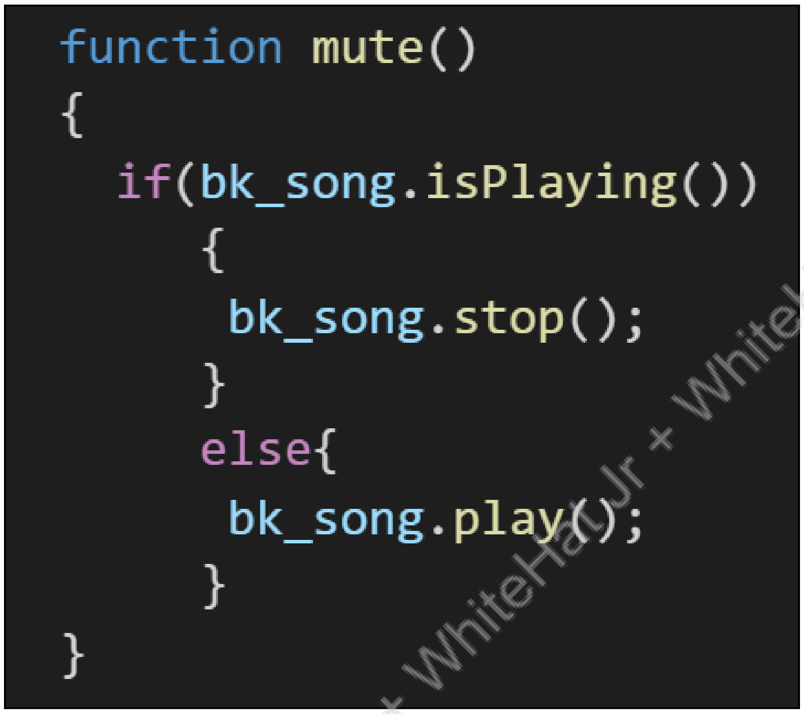
5. Play the background music using the **play()** function and set the volume using the **setVolume()** function. Set the volume to 50%, which is **0.5**.



```
function setup() {
  createCanvas(500,700);
  frameRate(80);

  bk_song.play();
  bk_song.setVolume(0.5);
```

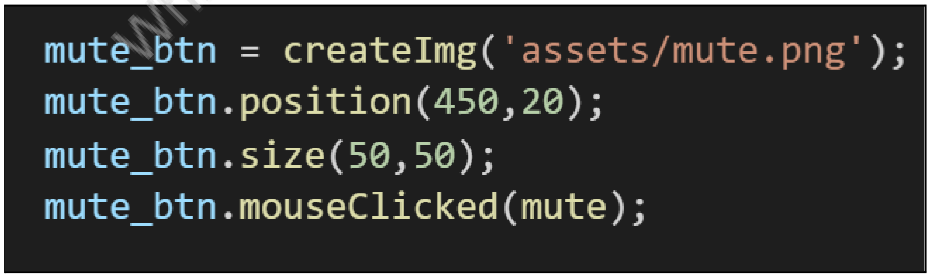
Now, playing the background music continuously can be irritating for some users.

6. For this, create a **mute()** function which gives the user a choice to stop or play the music with the press of a button. This function will first check if the music is playing or not. If the music is playing then it will stop the music else it will let the user play the music.



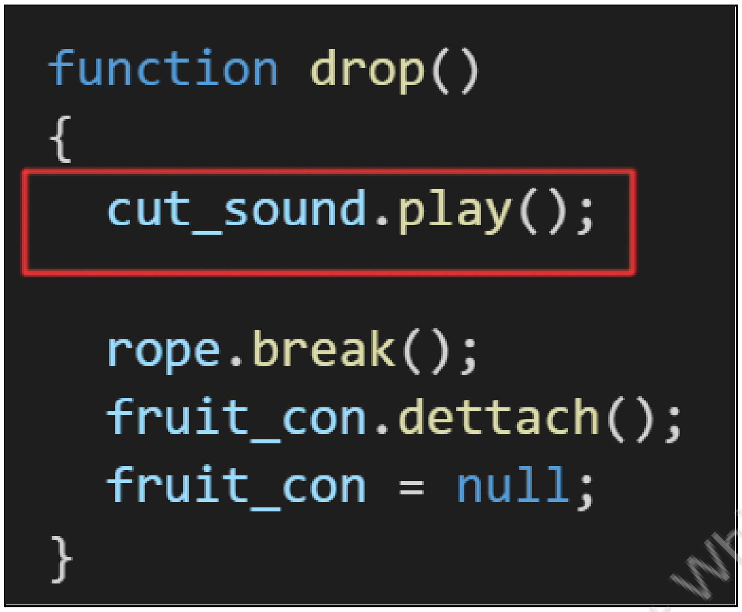
```
function mute()
{
  if(bk_song.isPlaying())
     {
      bk_song.stop();
     }
     else{
      bk_song.play();
     }
}
```

7. Now create a **mute** button using the **createImg()** function, and set the position and size of the **mute** button. When this button is pressed, the **mute()** function should be called.



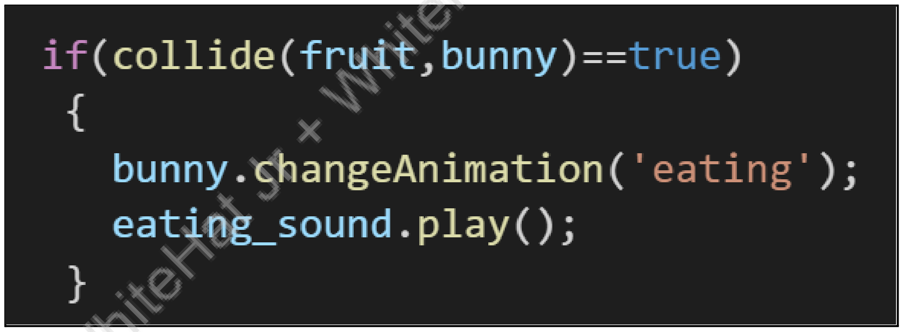
```
mute_btn = createImg('assets/mute.png');
mute_btn.position(450,20);
mute_btn.size(50,50);
mute_btn.mouseClicked(mute);
```

8. Next, add sound effects when the user cuts the rope.



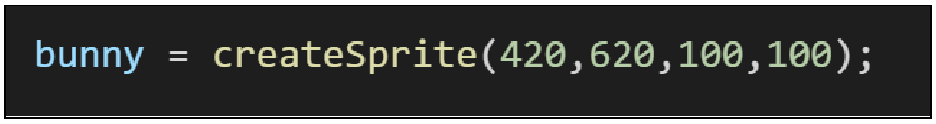
```
function drop()
{
   cut_sound.play();

   rope.break();
   fruit_con.dettach();
   fruit_con = null;
}
```

9. Detect the bunny and fruit collision using the **collide()** function, if the fruit collides with the bunny then play the eating sound and change the animation to eating.



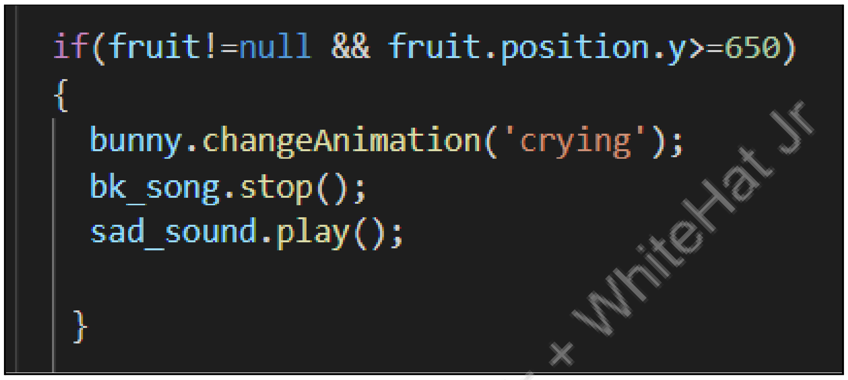
```
if(collide(fruit,bunny)==true)
 {
   bunny.changeAnimation('eating');
   eating_sound.play();
 }
```

10. We will also have the bunny moved to the right side of the canvas so that fruit doesn't fall directly on the bunny and the user has to use the air-blower to direct the fruit towards the bunny.



```
bunny = createSprite(420,620,100,100);
```

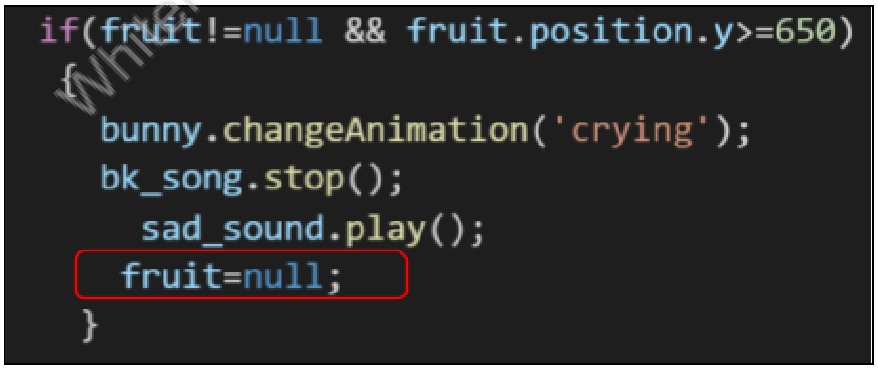
11. Now check the position of the fruit and also whether it is null or not. Use an **if** condition along with the **AND (&&)** operator to check the following conditions:
    a. Here, if the fruit **is not null**, and it reaches the ground, then update the bunny animation to crying.
    b. Stop the background music and play the sad sound.
    c. But since the above conditions are true, the sad sound will play continuously.
    d. This is a bug in our code.



```
if(fruit!=null && fruit.position.y>=650)
{
  bunny.changeAnimation('crying');
  bk_song.stop();
  sad_sound.play();

 }
```

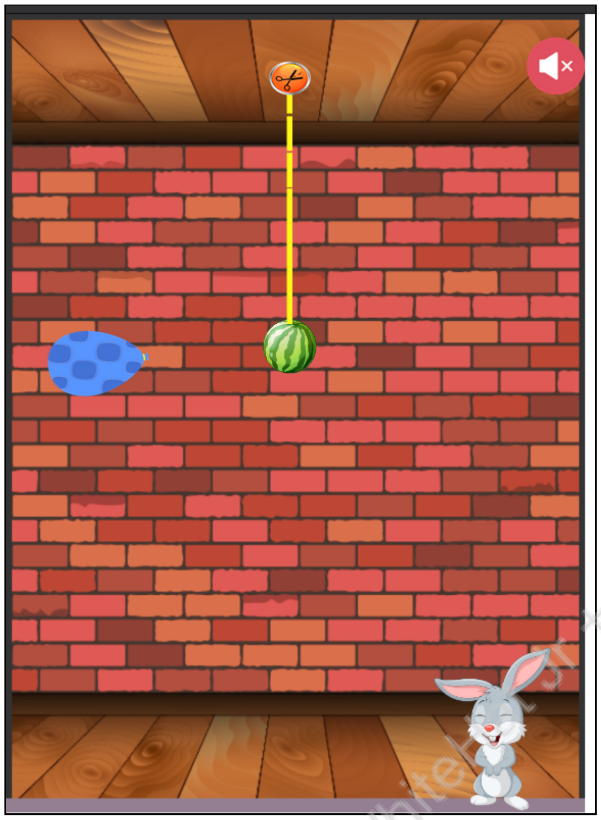
Playing the sound continuously in a game can make the user uncomfortable, hence we have to resolve this bug.

12. To fix this bug, we need to make one of the above conditions, **false**. For that, at the end of this conditional block, we will set fruit as **null**.
    a. Now once this code is executed the fruit will be **null** and the above condition would be **false**, so the following block of code will not be executed, thus fixing our bug.



```
if(fruit!=null && fruit.position.y>=650)
 {
   bunny.changeAnimation('crying');
   bk_song.stop();
     sad_sound.play();
    fruit=null;
  }
```

13. Run the code to see the final output:



What's next?

In the next class, we will level up the game by adding a few more ropes to the fruit and the user will have to cut the ropes in the right order to feed the bunny.

EXTEND YOUR KNOWLEDGE:

To know more about the Sound function is p5.js:
https://p5js.org/reference/#/libraries/p5.sound