

Topic	SPRITE MOVEMENTS	
Class Description	The student learns to move the sprite using the velocity properties of the sprite. Student creates edge sprites in the game and bounces off the main sprite from the edges of the canvas. Also, the student learns to create abstractions in their code by declaring functions.	
Class	PRO-C2	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> Write custom functions to animate and move the ball in different directions. Write a function to create edges for the canvas such that the ball bounces back from the edges. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Code.org login Laptop with internet connectivity Earphones with mic Notebook and pen Smartphone Student Resources: <ul style="list-style-type: none"> Code.org login Laptop with internet connectivity Earphones with mic Notebook and pen 	
Class structure	Warm-Up Slides Teacher-Led Activity 1 Student-Led Activity 1 Teacher-Led Activity 2 Student-Led Activity 2 Wrap-Up Slides	10 mins 10 mins 5 mins 10 mins 10 mins 5 mins

WARM-UP SESSION - 10 mins



Teacher starts slideshow from slides 1 to 9
 Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
<p>How have you been? Are you excited to learn something new?</p> <p><i>Run the presentation from slide 1 to slide 3.</i></p> <p>Following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> Connect students to the previous class. Warm-Up Quiz Session. 	<p>ESR: Varied Response.</p> <p>Click on the slide show tab and present the slides.</p>

QnA Session - Click on in-class quiz

Question	Answer
<p>How do we differentiate between properties and methods/functions in the list below for sprite objects?</p> <p>A. Functions/Methods always have rounded brackets at the end and properties don't.</p> <p>B. Properties always have rounded brackets at the end and functions don't.</p> <p>C. Both are the same.</p> <p>D. None of the above.</p>	A
<p>Which instruction/function is used to draw the sprite on the screen?</p> <p>A. createSprite() B. shapeColor C. drawSprites() D. draw()</p>	C

Continue the warm-up session

Activity details	Solution/Guidelines
------------------	---------------------



The teacher opens the activity link and explains the boilerplate code which has two sprites: a ghost sprite and a helicopter sprite - [Teacher Activity 1](#).

Code:

```
Workspace
1  var sprite = createSprite(100, 100, 20, 20);
2  sprite.setAnimation("ghost_standing_1");
3
4  var sprite2 = createSprite(200, 200, 20, 20);
5  sprite2.setAnimation("helicopter_1");
6
7
8  function draw() {
9    drawSprites();
10 }
11
12
13
14
```

Output:



The first step to use a function is to **'Define a Function'**.

- 1) Go to **Functions** and pick up the **'Define a Function'** block in the **Workspace** area.

```
Workspace
1  var sprite = createSprite(100, 100, 20, 20);
2  sprite.setAnimation("ghost_standing_1");
3
4  var sprite2 = createSprite(200, 200, 20, 20);
5  sprite2.setAnimation("helicopter_1");
6
7
8  function draw() {
9      drawSprites();
10 }
11
12 function myFunction() {
13
14 }
15
```

Change the function name: The function name should be something that describes the functionality of the function.

Since we are writing a function to rotate the sprite, we should name our function similar to the functionality i.e. **rotateSprite()**.

Next, we should think if the function requires any input. For e.g. the **setAnimation()** function needs the name of the animation you want to set for your sprite i.e. Ghost or Volleyball or Helicopter.

For the **rotateSprite()** function, we will have to tell the computer which sprite is to be rotated, so that should go as input to the function.

We always pass the input as a parameter in function definition.

The teacher writes the code inside the function to change the rotation of the sprite by 10.

We have successfully defined the **rotateSprite()** function but to use it we will have to call it in our program.

We can call the function by its name and pass the input inside the parenthesis.

Syntax: **function_name(parameter);**

The teacher makes changes in the code to call the rotate function twice, once for each sprite present in the program.

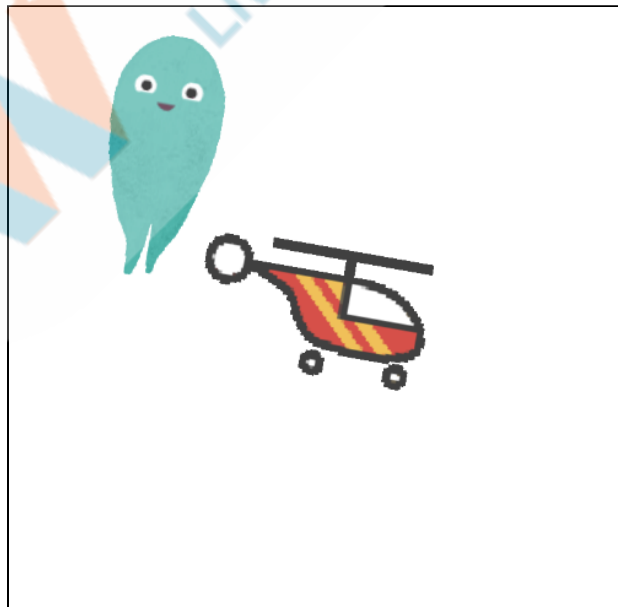
The teacher runs the program and shows the output to the student.

The student looks at the browser console and verifies the output of the code written by the teacher.

Code:

```
Workspace
1  var sprite = createSprite(100, 100, 20, 20);
2  sprite.setAnimation("ghost_standing_1");
3
4  var sprite2 = createSprite(200, 200, 20, 20);
5  sprite2.setAnimation("helicopter_1");
6
7  rotateSprite(sprite);
8  rotateSprite(sprite2);
9
10 function draw() {
11     drawSprites();
12 }
13
14 function rotateSprite(sprite) {
15     sprite.rotation = sprite.rotation + 10;
16     //sprite.rotation += 10;
17 }
18
```

Output:



Did you notice: We just instructed the computer once about how to rotate the sprite, and we can use it multiple times for different sprites without writing the same lines of code again and again? Isn't it powerful?

ESR: Yes.

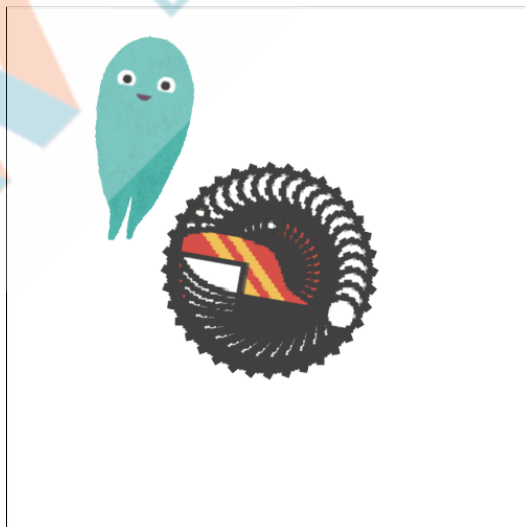
Functions can be called repeatedly to do the same set of instructions on different arguments. It helps us avoid repetition of code and hence shortens our code.


Let's try to call our function inside the **draw()** function which runs repeatedly.

Code:

```
3
4 var sprite2 = createSprite(200, 200, 20, 20);
5 sprite2.setAnimation("helicopter_1");
6
7 rotateSprite(sprite);
8 rotateSprite(sprite2);
9
10 function draw() {
11   rotateSprite(sprite2);
12   drawSprites();
13 }
14
```

Output:



<p>Did you see that the Helicopter sprite keeps on rotating repeatedly now?</p> <p>Because now we are calling the rotateSprite() function inside the draw().</p> <p>We also call these customized functions as 'User-defined' functions because we have created and customized these functions from scratch.</p> <p>I have a challenge for you now. Can you write a function to scale the sprite to half the original size?</p>	<p>ESR: Yes.</p>
<p style="text-align: center;">Teacher Stops Screen Share</p>	
<p style="text-align: center;">STUDENT-LED ACTIVITY 1 - 5 mins</p>	
<p>Now it's your turn. Please share your screen with me.</p>	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to the panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 	
<p style="text-align: center;"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • The student writes the function with parameters to resize sprites. • The student learns how to call the function in the program. 	
<div style="text-align: center;">  <p>Teacher starts slideshow for slide 10</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> </div>	
<p style="text-align: center;">Teacher Action</p>	<p style="text-align: center;">Student Action</p>
<p><i>Guide the student to open the Student activity 1 link and login into code.org if he/she hasn't still.</i></p> <ol style="list-style-type: none"> 1) Create a new function and give an appropriate name to it, in this case, resizeSprite. 2) Add input parameters to the new function i.e. sprite. 3) Call the new function in the code by its function name and inputs. resizeSprite(sprite). 	<p><i>The student clicks on Student Activity Link 1 and starts coding for his/her first customized function to resize the sprite.</i></p>

- 4) Add code for calling this function multiple times in the program with different inputs.

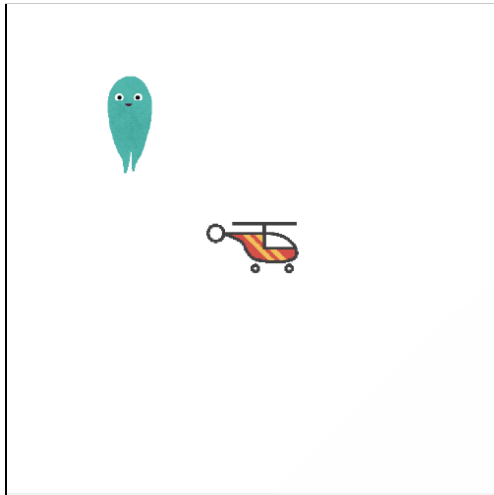
Guide the child and check for typos. Encourage the child to do it on his own. Ask questions for the next step if the child gets stuck at any place.

Appreciation and positive reinforcement will help the child gain confidence and make him/her confident to code.

Code:

```
Workspace
1  var sprite = createSprite(100, 100, 20, 20);
2  sprite.setAnimation("ghost_standing_1");
3
4  var sprite2 = createSprite(200, 200, 20, 20);
5  sprite2.setAnimation("helicopter_1");
6
7  resizeSprite(sprite);
8  resizeSprite(sprite2);
9
10 function draw() {
11   drawSprites();
12 }
13
14 function resizeSprite(sprite) {
15   sprite.scale = 0.5;
16 }
17
18
```

Output:



You did fabulous work. I am very confident about you. It will be a great journey together.

Teacher Guides Student to Stop Screen Share

TEACHER LED ACTIVITY 2 - 10 mins

CHALLENGE

- Learn to add velocity to the ghost object.
- Learn to create edges and introduce the mouse pressed event.



Teacher starts slideshow from slides 11 to 13
 Refer to speaker notes and follow the instructions on each slide.

Teacher Action

Student Action

In the last class, we learned how to create a sprite, set animation to the sprite and play around with some properties of the sprite.

Today, we will learn how to give movement to a sprite in the game.

Teacher clicks on [Teacher Activity Link 2](#) and opens the code from the previous class.

The student observes.

Currently, the sprite is static.

Shall we move it around by giving it speed?

ESR: Yes.

In science and programming, speed is also referred to as **velocity**.

ESR: Yes

Can you find a property from the list which can help us give some velocity/speed to the sprite?

ESR: Yes

*The teacher helps the student find **velocityX** & **velocityY**.
 The teacher assigns a positive value to **velocityX** first and runs the code.*

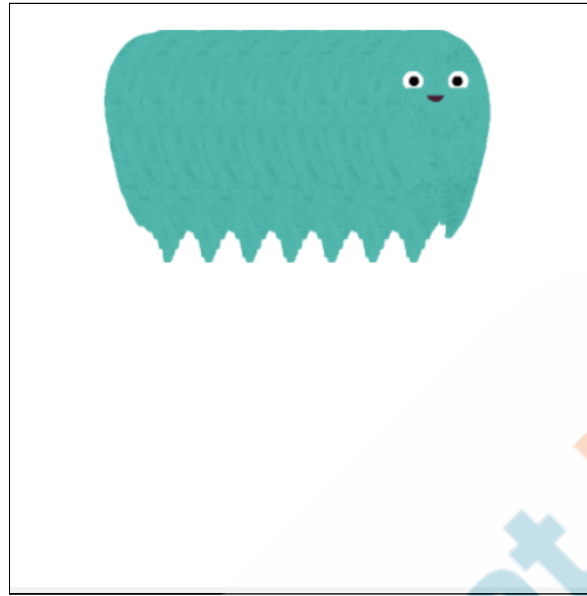
The student observes the code.

Code:

```

Workspace
1  var sprite = createSprite(100,100,20,20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityX = 2;
5
6  function draw() {
7    drawSprites();
8
9  }
10
  
```

Output:



What do you think happened?

Actually, the sprite is moving but drawing itself continuously at different positions. Do you remember I told you that the **draw()** function keeps running till the game ends?

What should we do to see only the last drawn sprite on the canvas/screen?

Yes we should clear the screen before drawing the sprite at a new position.

Let's try that. We go to the '**Drawing**' tab in the **Toolbox** and check for the **background()** function.

The teacher writes the code to clear the screen after every pass by making the background white in each pass.

ESR: The sprite is growing in size.

ESR: Yes.

ESR: Clear the screen before drawing the new sprite.

Code:

```
Workspace
1  var sprite = createSprite(100,100,20,20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityX = 2;
5
6  function draw() {
7    background("white");
8    drawSprites();
9  }
10 }
11
```

Output:



Great! Now the sprite is moving horizontally toward the right.

Let's try assigning **velocityY** instead of **velocityX**. What do you think will happen?

The teacher edits and runs the code.

ESR: The sprite will move vertically instead of horizontally.

Code:

```
Workspace
1  var sprite = createSprite(100,100,20,20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityY = 2;
5
6  function draw() {
7    background("white");
8    drawSprites();
9  }
10
11
```

Output:



The sprite is now moving vertically downwards.

What do you think will happen if we give both **velocityX** and **velocityY**?

Yes! The sprite will move both in a horizontal and vertical direction. It will look as if it is moving diagonally.

The teacher writes and runs the code.

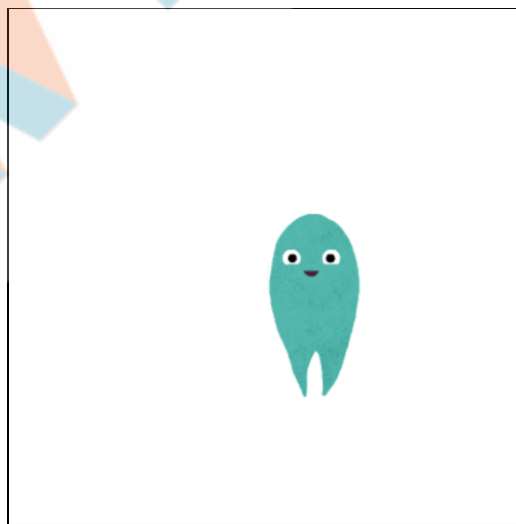
ESR:

The sprite starts moving diagonally.

Code:

```
Workspace
1  var sprite = createSprite(100,100,20,20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityX = 2;
5  sprite.velocityY = 2;
6
7
8  function draw() {
9    background("white");
10   drawSprites();
11
12 }
13
```

Output:



We got our sprite moving in the bottom right corner. How do we move our sprite in the horizontally opposite direction?

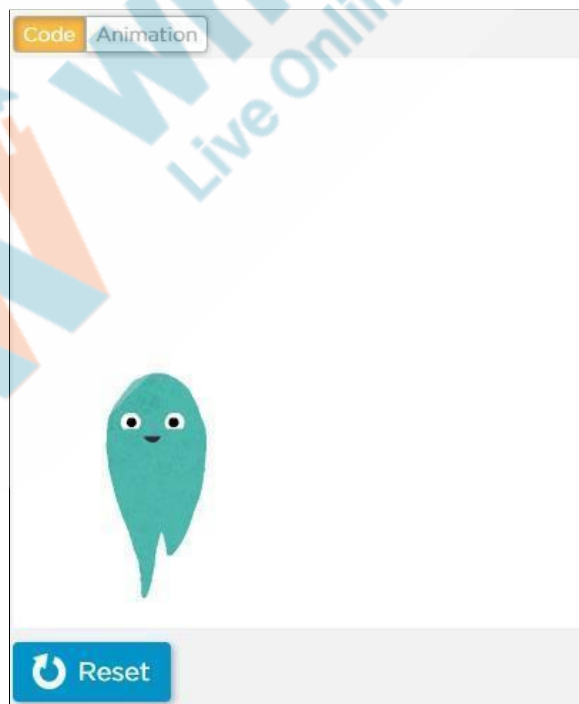
ESR: By giving negative value to **velocityX**.

*The teacher changes the **velocityX** of the sprite from 2 to -2 and runs the code.*

Code:

```
Workspace
1  var sprite = createSprite(200, 200, 20, 20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityX = -2;
5  sprite.velocityY = 2;
6
7  function draw() {
8    background("white");
9    drawSprites();
10 }
11
```

Output:



Similarly, we can move our sprite vertically upwards instead of downwards by assigning negative value to **velocityY**.

The teacher can ask the child to choose the values for both the velocity, predict the movement and test it.

We can also increase/decrease velocity in one of the directions which makes the sprite move faster in one direction.
For e.g. If the velocity in x-direction is more, the sprite will move horizontally faster.

Similarly, if we increase the value of **velocityY** and decrease the **velocityX**, the sprite will move vertically faster.

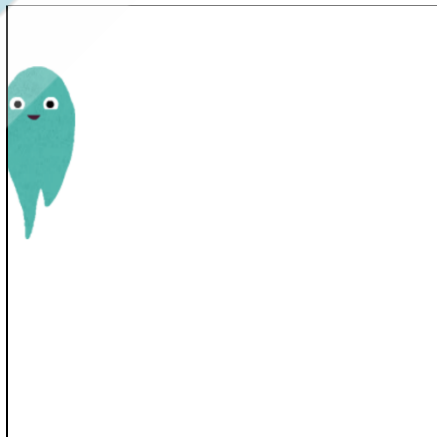
Let's try?

ESR: Yes

CODE:

```
Workspace
1 var sprite = createSprite(100,100,20,20);
2 sprite.setAnimation("ghost_standing_1");
3
4 sprite.velocityX = -2;
5 sprite.velocityY = 1;
6
7
8 function draw() {
9   background("white");
10  drawSprites();
11
12 }
13
```

OUTPUT:



Great! It seems we can move the sprite wherever we want now.

Now that our sprite has started moving. Do you notice any issues in the game?

To fix this, let's create edges on the screen. Can you find something in the sprite tab, to create edges on the canvas?

`createEdgeSprites()`

The teacher adds "createEdgeSprites()" to the code and asks the student why we should add the code outside draw()?

The teacher to run the code and show the output.

ESR: Yes, the sprite goes outside the screen.

ESR: `createEdgeSprites()`

ESR: Because we need to create the boundaries once. Whatever we write inside the `draw()` function repeats again and again.

The student observes the screen and learns the code.

CODE:

```

Workspace
1  var sprite = createSprite(100,100,20,20);
2  sprite.setAnimation("ghost_standing_1");
3
4  sprite.velocityX = -2;
5  sprite.velocityY = 1;
6
7  createEdgeSprites();
8
9  function draw(){
10     background("white");
11     drawSprites();
12 }
13
  
```

Oh! The ghost is still not bouncing off the boundaries/edges. Why?

We did make edge sprites, but we didn't ask the computer to have the sprite bounce off the boundary edges.

Remember, a computer only does whatever we tell it to.

ESR: Varied

Can you look for a **bounceOff(target)** in the **Sprite** tab?

*The teacher adds the **bounceOff** instruction in the code and changes the target to edges.*

Can you tell me why we add the **bounceOff()** instruction in the **draw()** function?

The teacher should help the child with the explanation if the child is not able to answer it.

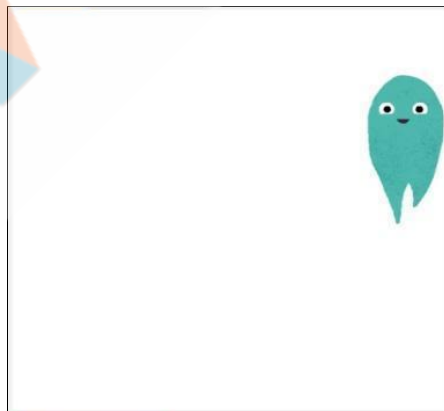
ESR: Yes.

ESR: Because we want the sprite to **bounceOff** the walls/edges throughout the game.

CODE:

```
Workspace
1 var sprite = createSprite(100,100,20,20);
2 sprite.setAnimation("ghost_standing_1");
3
4 sprite.velocityX = -2;
5 sprite.velocityY = 1;
6
7 createEdgeSprites();
8
9 function draw(){
10   background("white");
11
12   sprite.bounceOff(edges);
13
14   drawSprites();
15 }
16
```

OUTPUT:



Teacher Stops Screen Share

STUDENT-LED ACTIVITY 2 - 10 mins

Now it's your turn. Please share your screen with me.

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

CHALLENGE

- Make the ball bounce off the edges.
- Add a mouse press event. The ball should move or gain velocity **only** when the mouse button is clicked.



Teacher can show slideshow for slide 14-15

Refer to speaker notes and follow the instructions on each slide.

Teacher Action

Student Action

*Ask the student to click on the [Student Activity 2](#) and **Remix** the file.*

In the last class, you had created the ball sprite for the game.
Let's add movement to our ball sprite.

Guide the student to assign velocity to the ball.

The student writes code on top of the current stage of the game to assign the velocity to the sprite.

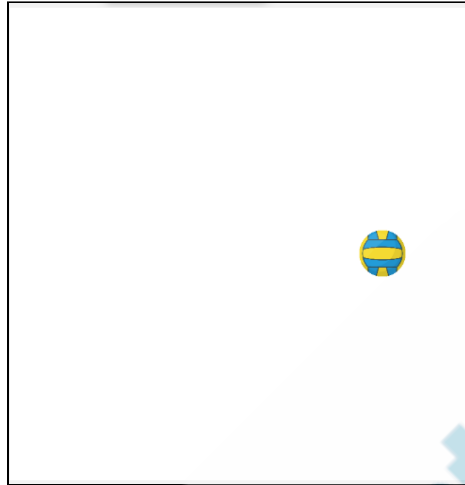
He/She runs the code to observe the output.

CODE:

```

Workspace
1  var ball;
2
3  ball = createSprite(100,100,20,20);
4  ball.setAnimation("volleyball2_1");
5  ball.scale = 0.1;
6
7  ball.velocityX = 4;
8  ball.velocityY = 2;
9
10 function draw() {
11   background("white");
12   drawSprites();
13
14 }
15
16
  
```

OUTPUT:



Guide the student to make the ball bounce off the edges as shown in the following code snippet:

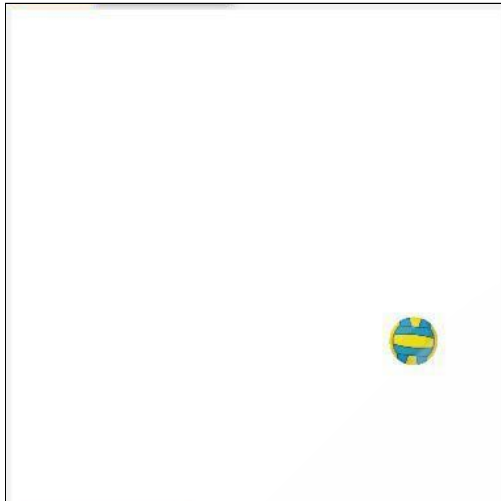
Observe the student, check for any typos she/he makes. Guide the student to write the code.

The student writes code to make the ball bounce from the edges.

CODE:

```
Workspace
1  var ball;
2
3  ball = createSprite(100, 100, 20, 20);
4  ball.setAnimation("volleyball2_1");
5  ball.scale = 0.1;
6
7  ball.velocityX = 4;
8  ball.velocityY = 2;
9
10 createEdgeSprites();
11
12 function draw(){
13   background("white");
14   drawSprites();
15
16   ball.bounceOff(edges);
17 }
18
```

OUTPUT:



Great! Now our ball is bouncing off the edges. Do you see the ball start moving as soon as we run the program?

ESR: Yes.

Amazing! You have successfully completed today's challenge. I am really impressed.

In the next class, we will create a paddle sprite for our game and move it using our mouse.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

FEEDBACK

- **Appreciate and compliment** the student for trying to learn a difficult concept.
- **Get to know how they are feeling** after the session.
- **Review and check their understanding.**






Teacher can show slideshow for slides 16 to 25
 Refer to speaker notes and follow the instructions on each slide.

For now, can we quickly summarize what we have learned today?

ESR:

- We learned to write functions in our code.
- We learned how to give movement to sprites.

	- We created edges and added the bounceOff() function for the sprite.
QnA Session - Click on in-class quiz	
Question	Answer
<p>Why do we write functions?</p> <p>A. To make the code more readable. B. To make the code reusable. C. To avoid writing repeated code. D. All of the above.</p>	D
<p>Choose the correct code syntax to make the ball bounce off the walls.</p> <p>A. <code>var ball= createSprite(200, 200,10,10);</code> <code>ball.velocityX=2;</code> <code>ball.velocityY =3;</code></p> <p><code>function draw() {</code> <code>background("white");</code> <code>ball.bounce(edges);</code> <code>drawSprites();</code> <code>}</code></p> <p>B. <code>var ball= createSprite(200, 200,10,10);</code> <code>ball.velocityX=2;</code> <code>ball.velocityY =3;</code> <code>createEdgeSprites();</code></p> <p><code>function draw() {</code> <code>background("white");</code> <code>drawSprites();</code> <code>}</code></p> <p>C. <code>var ball= createSprite(200, 200,10,10);</code> <code>ball.velocityX=2;</code> <code>ball.velocityY =3;</code> <code>createEdgeSprites();</code></p> <p><code>function draw() {</code> <code>background("white");</code> <code>ball.bounceOff(edges);</code></p>	C

<pre> drawSprites(); } D. var ball= createSprite(200, 200,10,10); ball.velocityX=2; ball.velocityY =3; createEdgeSprites(); function draw() { background("white"); ball.bounceOff(); drawSprites(); }. </pre>	
<p>Which function is used to stop the sprite from going outside the screen by creating the boundary around the canvas?</p> <p>A. drawSprites() B. createSprites() C. bounceOff() D. createEdgeSprites()</p>	<p>D</p>
<p>You get Hats off for your amazing performance today.</p> <p>Alright, we seemed to have a lot of learning in the class today.</p> <p>In the next class, we will learn to control the movement of the ball.</p> <p>Isn't that interesting!</p>	<p>Make sure you have given at least 2 Hats Off during the class for:</p> <div data-bbox="1019 1102 1312 1192">  +10 Creatively Solved Activities </div> <div data-bbox="1019 1213 1312 1297">  +10 Great Question </div> <div data-bbox="1019 1318 1312 1402">  +10 Strong Concentration </div>
<p>Project Overview</p> <p>HIT & RUN</p> <p>Goal of the Project: In Class 2, you learned how to define customized functions. Also, you learned how to create edge sprites and bounce off the ball from the edges. You will now create a small maze game to bounce the moving balls from edges and box Sprites.</p>	<p><i>The student engages with the teacher over the project.</i></p>

In this project, you will have to practice what you have learned in the class and apply it to make the ball move and bounce inside the maze and restrict it from going out of the canvas.

Story:

Dodo loves to play with the ball and have it bounce from the walls and different objects. But there is always a fear of breaking things inside the home. Dodo's mother helps him create an arena full of objects in the backyard, but since Dodo has begun his coding journey now, he is eager to try his hand at creating a virtual arena where he can bounce his ball around and play without fear of breaking things.

Can you help Dodo build such an arena?

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

Bye Bye!

Teacher ends slideshow



ADDITIONAL ACTIVITY

Student to share the screen

Today we learned how to create a customized function. Can you create a function to add two numbers?

Guide the student to click on Create-> Game Lab to start a new project from scratch.

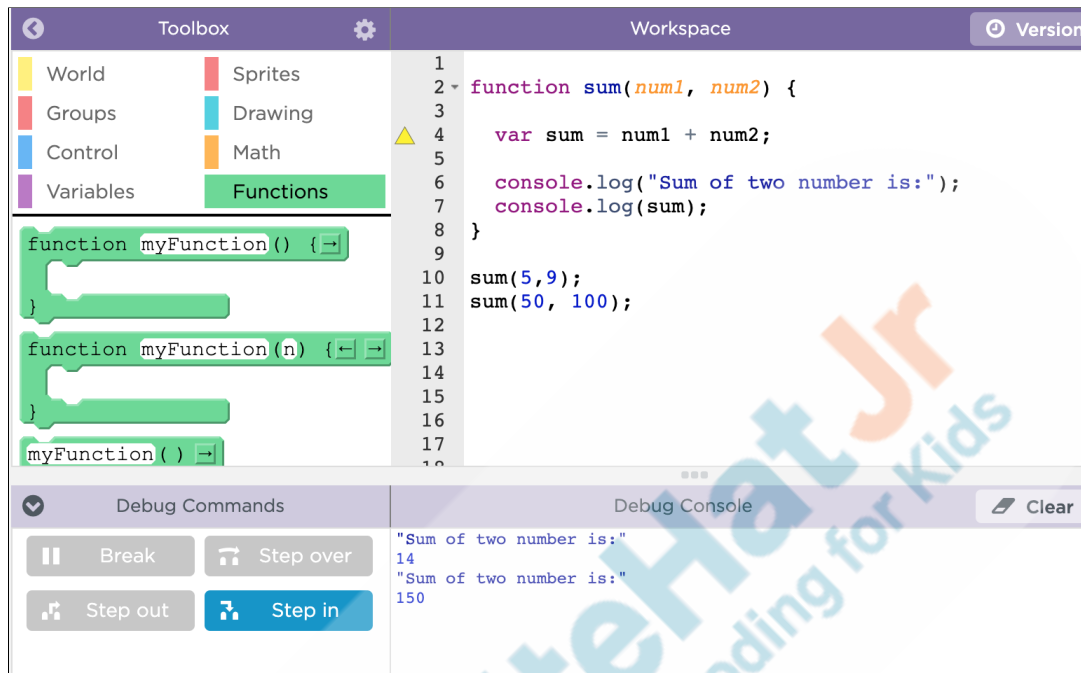
- What should be the name of the function?
- How many inputs should be passed to the function?

*Teacher guides the child in writing the **sum()** function with two input parameters num1 and num2. Print the result on the console.*

ESR: sum

ESR: 2

The teacher helps the child in calling the function with different inputs in the same code and running the code.



The screenshot shows the WhiteHat Jr IDE interface. On the left is the 'Toolbox' with categories like World, Groups, Control, Variables, Sprites, Drawing, Math, and Functions. The 'Functions' category is selected, showing two custom functions: 'myFunction()' and 'myFunction(n)'. The main 'Workspace' area contains the following code:

```

1
2 function sum(num1, num2) {
3
4   var sum = num1 + num2;
5
6   console.log("Sum of two number is:");
7   console.log(sum);
8 }
9
10 sum(5,9);
11 sum(50, 100);
12
13
14
15
16
17
18

```

At the bottom, the 'Debug Console' shows the output of the code execution:

```

"Sum of two number is:"
14
"Sum of two number is:"
150

```

You were fantastic today. You quickly grasped all the concepts.

Activity	Activity Name	Links
Teacher Activity Link 1	User Defined Function: Rotate Sprite	https://studio.code.org/projects/gamelab/pBWKXIECXXtiho66TCylOOzWkBP92_TRj95_o2w
Teacher Activity Link 1 (Ref Code)	User Defined Function: Rotate Sprite	https://studio.code.org/projects/gamelab/nLfHfPEgKMllxyEMX_-KvsQtAmcSXUj7RrwHB4FP0yY
Student Activity Link 1	Resize Sprite	https://studio.code.org/projects/gamelab/ZskSKWsW-595e6KS698sYrPZBspn8Oix4AQO5ajGjiM
Teacher Reference Code (Resize)	Resize Sprite	https://studio.code.org/projects/gamelab/ZskSKWsW-595e6KS698sYi51-JNifOydAXpd4pGbi3g
Teacher Activity Link 2	Ghost Movement	https://studio.code.org/projects/gamelab/WzAF0u77Tm1W2uWEzInmdlzTqDu5Ktb_mx4wN4ho6Cc

Teacher Activity Link 2 (Ref Code)	Ghost Movement	https://studio.code.org/projects/gamelab/WzAF0u77Tm1W2uWEzInmdunoay_OpT5CnwiTSAXBCuQ
Student Activity Link 2	Moving Ball	https://studio.code.org/projects/gamelab/fNykX4SJyIrTRciOW5am_m_88eBFxToR0BSFgRQ-1tA
Teacher Reference Code (Moving Ball)	Moving Ball	https://studio.code.org/projects/gamelab/fNykX4SJyIrTRciOW5am_nzTUgmSnc1rwpJq2Pf3B3U
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+As set/PRO_VD/BJFC-PRO-V3-C2L-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/cc9f9a01-46a1-4ba0-840a-df871f94db31.pdf
Additional Activity (Ref Link)	Sum Function	https://studio.code.org/projects/gamelab/nhetrZ_jg_0bCmHRx4d59N8MxB0qiK_vlPvOA7sXQDE