



PLEASE NOTE

END YOUR CLASS WITH WOW FACTOR.


Amaze your student with a FUN WITH TECH

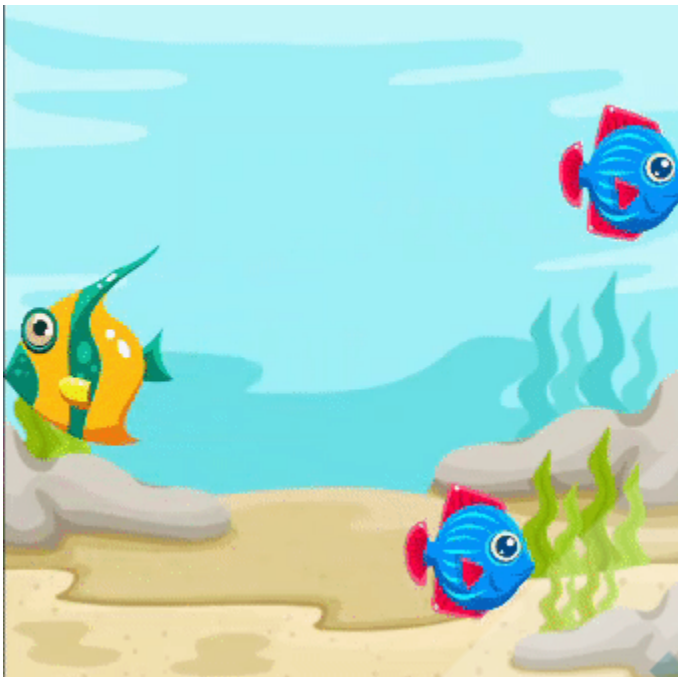
Find the Details in VA

The 5 min activity can increase your chance of Student Renewal

AT THE END OF THE CLASS REMIND STUDENTS ABOUT PTM IN NEXT CLASS

| Topic | LOOPS |
|--------------------|--|
| Class Description | The student will learn about loops and start creating a Breakout Game which has a paddle at the bottom of the screen and multi-colored bricks on the top. The player has to break all the bricks to win the game. |
| Class | PRO-C4 |
| Class time | 55 mins |
| Goal | <ul style="list-style-type: none"> • Understand the use of for loop. • Use 'for loop' and custom function to create multiple rows of bricks. • Create a sprite group for bricks. |
| 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 Teacher-Led Activity 1 Student-Led Activity 1 Teacher-Led Activity 2 Student-Led Activity 2 Wrap-Up | 10 mins 10 mins 5 mins 10 mins 10 mins 10 mins |
| WARM-UP SESSION - 10 mins | | |
|  Teacher starts slideshow from slides 1 to 12 Refer to speaker notes and follow the instructions on each slide. | | |
| Teacher Action | | Student Action |
| How have you been? Are you excited to learn something new? <i>Run the presentation from slide 1 to slide 3.</i> Following are the warm-up session deliverables: <ul style="list-style-type: none"> • Connect students to the previous class • Warm-Up Quiz Session | | ESR: Varied Response. Click on the slide show tab and present the slides. |
| QnA Session | | |
| Question | | Answer |
| Select the correct answer that checks whether the fish is going out of the canvas. | | C |



```
for(fish1.x > 430) {  
    fish1.x = 0;  
}
```

A.

```
else(fish1.x > 430) {  
    fish1.x = 0;  
}
```

B.

```
if(fish1.x > 430) {  
    fish1.x = 0;  
}
```

C.

```
whether(fish1.x > 430) {  
    fish1.x = 0;  
}
```


D.

Which block of code can be used to execute some code when the mouse is clicked on the canvas?

A

```
function mousePressed() {  
    // code to execute  
}
```

A.

| <p>B. <pre>function pressed() { // code to execute }</pre></p> <p>C. <pre>function keyPressed() { // code to execute }</pre></p> <p>D. <pre>function mouse() { // code to execute }</pre></p> | |
|--|--|
| Continue the warm-up session | |
| Activity details | Solution/Guidelines |
| <p>Run the presentation from slide 4 to slide 10 to set the problem statement.</p> <p>Following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> • Use of for loop • About Breakout Game | <p>Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in students.</p> |
| <div>Teacher ends slideshow </div> | |
| TEACHER-LED ACTIVITY 1 - 10 mins | |
| Teacher Initiates Screen Share | |
| <p><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Learn to avoid repetition in the code using the 'for loop'. • Understand the syntax and flow of 'for loop'. | |
| Teacher-led Activity (10 mins) | |
| Teacher Action | Student Action |

Today we will learn a very important concept of Loops. Be attentive in the class and stop me wherever you have a question.

Do you remember whatever we write **outside** the **draw()** function runs only **once** and the code **inside** the **draw()** function runs in a loop for an infinite number of times?

*The teacher opens the [Teacher activity 1 link](#) and writes code to print a statement, one inside the **draw()** function and the other one outside the **draw()** function as shown below.*

The teacher runs the code and shows the output to the child.

ESR: Yes

The student observes the code and the output.

CODE:

```

1
2 console.log("print once");
3
4 function draw()
5 {
6   console.log("print repeatedly");
7 }
8
  
```

OUTPUT:

```

"print once"
"print repeatedly"
"print repeatedly"
"print repeatedly"
"print repeatedly"
"print repeatedly"
"print repeatedly"
"print repeatedly"
"print repeatedly"
  
```

We can conclude that **"print once"** is displayed only one time and **"print repeatedly"** is displayed infinite times.

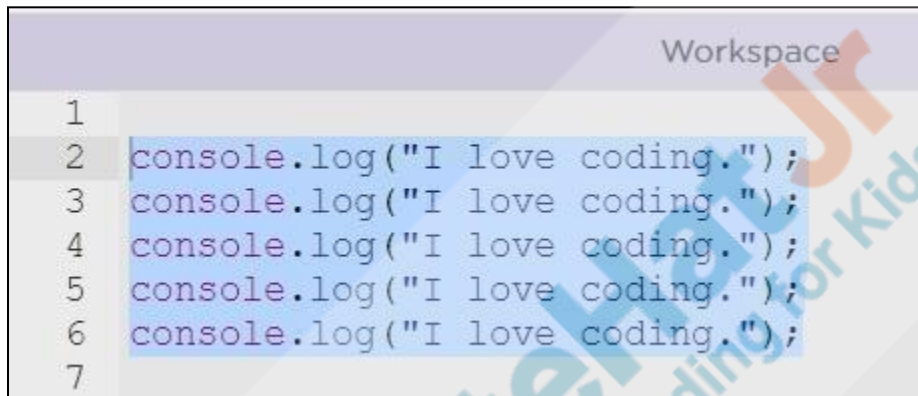
Can you tell me how can we print 'I love coding' on the console **five** times?

ESR: Write it five times outside **draw()**.

The teacher adds the changes to the code and the student observes the changes.

The student looks at the Browser console.

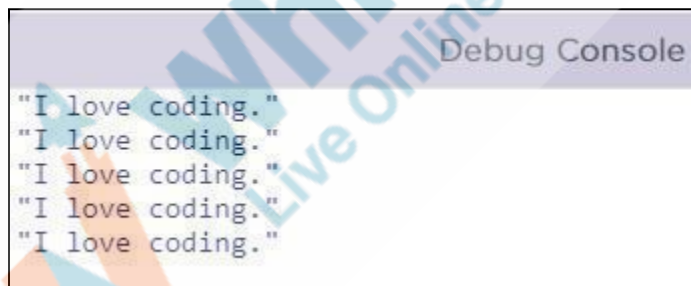
CODE:



```

1
2 console.log("I love coding.");
3 console.log("I love coding.");
4 console.log("I love coding.");
5 console.log("I love coding.");
6 console.log("I love coding.");
7
  
```

OUTPUT:



```

"I love coding."
"I love coding."
"I love coding."
"I love coding."
"I love coding."
  
```

But what if I ask you to print it ten times or maybe even more say a hundred times. Will writing the same command a hundred times work?

ESR: Yes.

Wouldn't your code become too long and repetitive? This is not how a good programmer works.

ESR: Yes.

To avoid such repetition of code, we use **loops**.

Can you see the '**for**' command in the **Control** tab of the **Toolbox**?

ESR: Yes.

World

Groups

Control

Variables

Sprites

Drawing

Math

Functions

for (var i = 0 ; i < 4 ; i++) {

}

while () {

Workspace

Version History

```

1
2 function draw() {
3   background("white");
4   drawSprites();
5 }
6
7

```

for loop

Creates a loop consisting of an initialization expression, a conditional expression, an incrementing expression, and a block of statements executed for each iteration of the loop.

[See examples](#)

Now we will write a loop to print 'I love coding' ten times on the console.

*The teacher will explain the **counter** variable and its **initialization**, **exit condition**, and the **flow** of the for loop and **increment expression**.*

The teacher should also explain the "<=" less than equal to the comparison operator. (if needed)

Run the program and count the number of times the command inside the loop ran for.

The teacher makes changes in the code and runs it.

CODE:

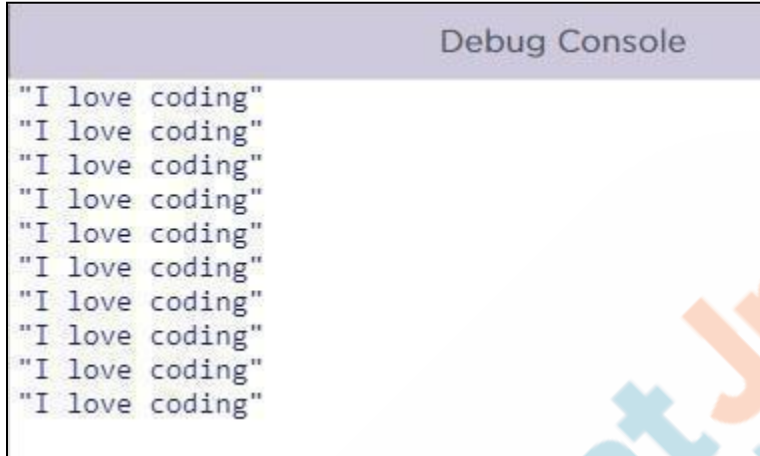
The student looks at the browser console and observes the output.

Workspace

```

1
2 for(var i=1; i<=10; i=i+1)
3 {
4   console.log("I love coding");
5 }
6
7

```


OUTPUT:

```
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
"I love coding"
```

Aren't loops powerful? We can actually print '**I love coding**' a thousand or more times with just three lines of code using loops.

We can also have multiple commands inside the **for** loop block.

Let's print the value of the counter variable as well inside the loop. It should print the value of **i** in every pass.

Run the code and check.

You will see two statements printing in each pass. i.e. '**counter number**' and '**I love coding**'.

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

CODE:


```

Workspace
1
2 for(var i=1; i<=10; i=i+1){
3   console.log(i);
4   console.log("I love coding");
5 }
6
  
```

OUTPUT:

```

1
"I love coding"
2
"I love coding"
3
"I love coding"
4
"I love coding"
5
"I love coding"
6
"I love coding"
7
"I love coding"
8
"I love coding"
9
"I love coding"
10
"I love coding"
>
  
```


Let's try doing a few more interesting things using the **for()** loop.
 Can you tell me all odd numbers less than 10?

Can you now tell me how to program a computer to print all the odd numbers less than **10**?

*The teacher discusses with the student what should be the **initial value(1)**, **exit condition(<10)**, and the **increment value(2)**. The teacher explains the logic behind incrementing the value by 2 in*

ESR: 1,3,5,7,9

Let the student think and answer.

| | |
|---|---|
| <i>each pass for odd numbers because integers are alternatively odd and even.</i> | |
| <div> <div> Workspace <pre> 1 2 3 for(var i=1; i<=10; i=i+2) 4 { 5 console.log(i); 6 } 7 </pre> </div> <div> Debug <pre> 1 3 5 7 9 </pre> </div> </div> <div> <div>CODE</div> <div>OUTPUT</div> </div> | |
| <p>Awesome! You are doing good. Now that you know how loops work here is a challenge for you:</p> <p>Can you write a program to print all the even numbers between 4 and 20?</p> | |
| Teacher Stops Screen Share | |
| STUDENT-LED ACTIVITY 1 - 10 mins | |
| Now it's your turn. Please share your screen with me. | |
| <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>CHALLENGE</p> <ul style="list-style-type: none"> • The student writes the function to print even numbers between 4 and 20. | |
| <div>  </div> <p>Teacher starts slideshow for slides 13 to 14</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> | |
| Teacher Action | Student Action |
| <i>Help the student open the Student activity 1 link and login to code.org if he/she hasn't still.</i> | The student clicks on Student activity link 1 and starts coding |

Help the student drag the **for()** loop block from the toolbox.

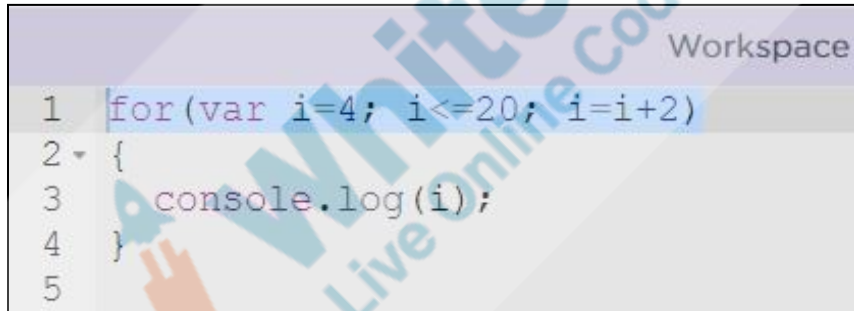
Guide the student to make changes in the code to write a **for** loop which prints even numbers between **4** and **20**.

Explain to the student:

- 1) The initialize condition should be **i=4** because we need to find even numbers starting from **4** and not before.
- 2) The **exit condition** should be **less than or equal to 20**. Once it exceeds **20**, the loop should **end**.
- 3) As briefed earlier, since even and odd numbers occur alternatively, **we need to add 2 to the preceding even number to get the next even number**.

to print all the even numbers between 4 and 20.

CODE:



```
1 for (var i=4; i<=20; i=i+2)
2 {
3   console.log(i);
4 }
5
```

OUTPUT:

Debug Console

```

4
6
8
10
12
14
16
18
20

```

Great, we have got all the even numbers between 4 and 20.
 Awesome! You did it comfortably. You are becoming a PRO at it.

Teacher Guides Student to Stop Screen Share

TEACHER LED ACTIVITY 2 - 10 mins

CHALLENGE

- Learn to create bricks in the game.
- Learn to bring abstraction into the code.


Teacher starts slideshow from slides 15 to 18
 Refer to speaker notes and follow the instructions on each slide.

| Activity details | Solution/Guidelines |
|---|-----------------------|
| <i>The teacher clicks on Teacher Activity Link 2 and opens the code from the previous class.</i> | The student observes. |
| <p>Great, we are ready with the ball and paddle in the game.</p> <p>What do you think is missing in the game now?</p> <p>Today we will create bricks in the game.</p> <p>Let's look at the original game to see how many rows and columns of bricks do we need in the game.</p> | ESR: Bricks |

The teacher opens the '[Reference Game code](#)' link and runs it for the student.

We have to build four rows and in each row we will have six bricks.

Should we write **createSprite()** for 6 times to create a row of bricks or rather use a **for** loop for the same?

The teacher explains to the student that we will need to write a loop for creating a row which should run 6 times and in each pass we will create one brick sprite.

We will keep the size of the brick 50 x 25 but what about the position of the brick?

In case Visual Aid (Slides) is not shown to the child, then use the [image](#) below to explain the following:

i) The x-position of the brick will change but the y-position will remain the same for a row.

ii) Calculation of the offset position for 1st brick.

Canvas width: 400

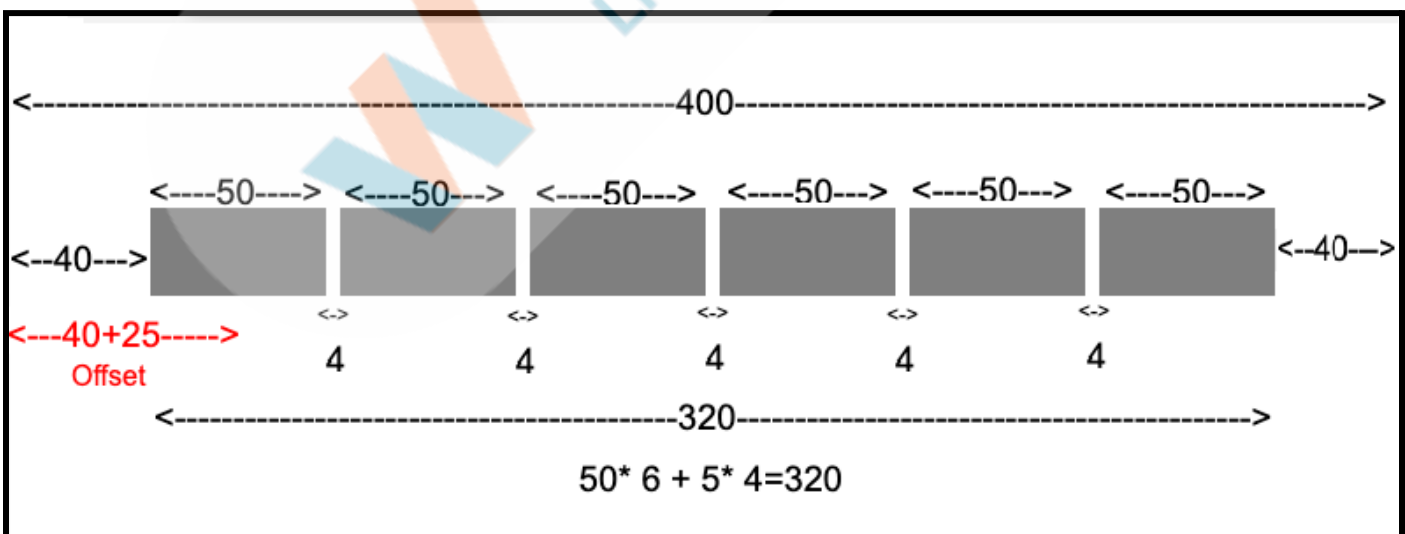
Brick width: 50

Gap between 2 bricks: 4

Margin left: $400 - 300$ (6 bricks of 50) - 20 (5 gaps of 4) = 80
so we will have an offset of 40 on each side. (i.e. $80/2$)

The student observes the screen.

ESR: Loops



The 1st brick will start from 40 (offset) + 25 (width/2 for center) = 65 and subsequent bricks at a distance of 54 from the last brick created.

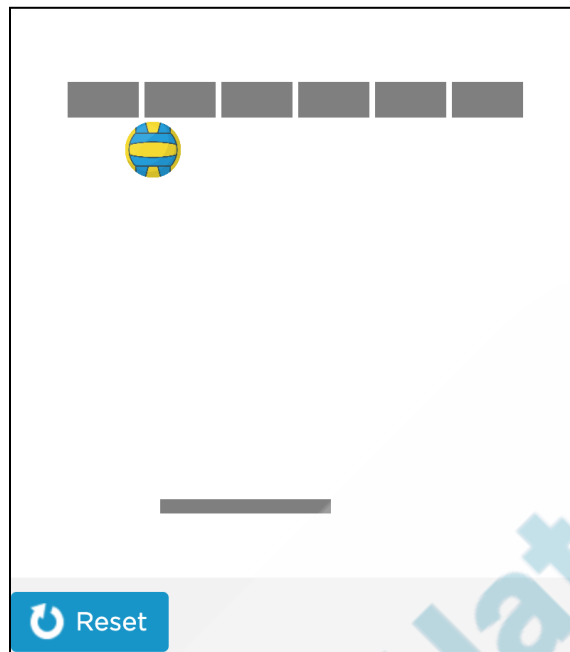
c is the column number.

We are starting from **0** that's the reason we have used the **less-than operator** in the exit condition.

Code:

```
9
10 var paddle = createSprite(200, 350, 120, 10);
11
12 createEdgeSprites();
13
14 for(c=0; c<6; c++)
15 {
16     var brick = createSprite(65+54*c, 65, 50, 25);
17 }
18
19 function draw(){
20     background("white");
21
22     paddle.x = World.mouseX;
23
24     if(paddle.x < 60){
25         paddle.x = 60;
```

Output:



Great. Now our first row is ready. But there is something missing. Can you tell me what is missing?

ESR: Color.

Yes. We have to set the color of the bricks. For the first row, we need the brick to be of red color.

Do you remember how we set the color of the sprite?

ESR: shapeColor property

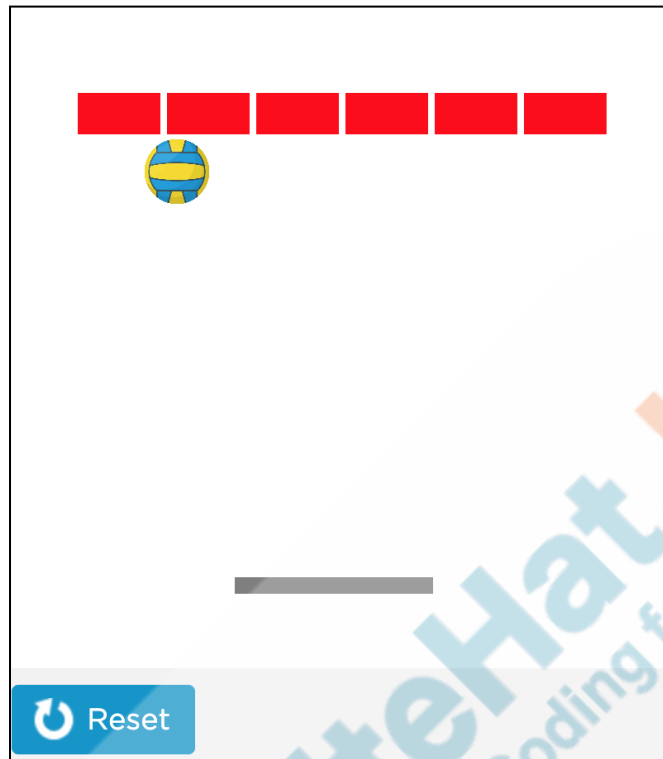
The teacher writes the code to set the color of the sprite to a 'red' color.

CODE:

```

10 var paddle = createSprite(200, 350, 120, 10);
11
12 createEdgeSprites();
13
14 for(c=0; c<6; c++)
15 {
16   var brick = createSprite(65+54*c, 65, 50, 25);
17   brick.shapeColor = "red";
18 }
19
20 function draw(){
21   background("white");
22
  
```


OUTPUT:



Awesome. First row is ready. Can you create the rest of the rows?

What do you think will change in the other rows?

Correct! We will just have to change the y-position of the bricks in the other rows. x-position, width, and height will remain exactly the same.

Note for Teachers: Students can go ahead and repeat this loop with different y-values and color values for creating the next 3 rows
OR

Teachers can define a function to create a row with the 'y' and 'color' parameters. Students can then call this function 3 more times in the program to create the next 3 rows.

Function Definition: createRowBrick() [optional]

Remember we created a customized function **'rotateSprite()'** to rotate any sprite by 10 degrees by passing the sprite name in the function call instead of writing the code again.

Similarly, let's define the function for creating a row and pass


ESR: Yes.

ESR: y-position and color of bricks.

y-position and color as arguments in the function call instead of writing the same code again and again. This is called **abstraction**.

The teacher defines the 'createBrickRow(y, color)' function, copies the loop inside the function, and calls it for the first row of bricks.

CODE:



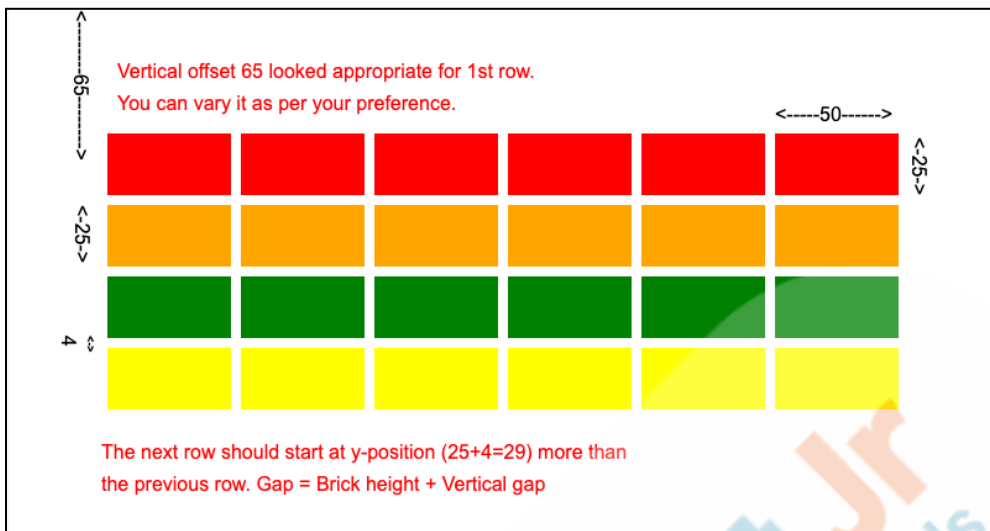
```

6
7 ball.velocityX = 0;
8 ball.velocityY = 0;
9
10 var paddle = createSprite(200, 350, 120, 10);
11
12 createEdgeSprites();
13 createBrickRow(65, "red");
14
15 function createBrickRow(y, color) {
16   for(c=0; c<6; c++)
17   {
18     var brick = createSprite(65+54*c,y,50, 25);
19     brick.shapeColor = color;
20   }
21 }
22
23 function draw(){
24   background("white");
25
26   paddle.x = World.mouseX;
27
  
```

Calculation of y-position for next 3 Brick Rows:

Let's figure out the y-position of bricks for the other rows:
 We started the first row at a vertical offset of 65. The next row should start at a gap of brick height (i.e. 25) + vertical gap between bricks (i.e. 4) = 29.

The teacher shows the [image](#) below to explain the total vertical gap between two bricks:



This was one of the most difficult concepts we have learned today. If you have understood this, the rest of it will be easy.

Great, now we know the values of y-position and color for the other rows. It's your time to call the **createBrickRow()** function and create multiple rows of bricks in the game.

Teacher Stops Screen Share

STUDENT-LED ACTIVITY 2 - 10 mins

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

CHALLENGE

- Create the next rows of bricks in the game.
- Enhance the game graphics.
- Create a sprite group 'bricks' and bounce the ball from the bricks group.



Teacher starts slideshow from slides 19 to 20
Refer to speaker notes and follow the instructions on each slide.

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

Teacher Action

Student Action

Guide the student to click on the [Student Activity 2](#) link and click on

The student opens the [Student](#)

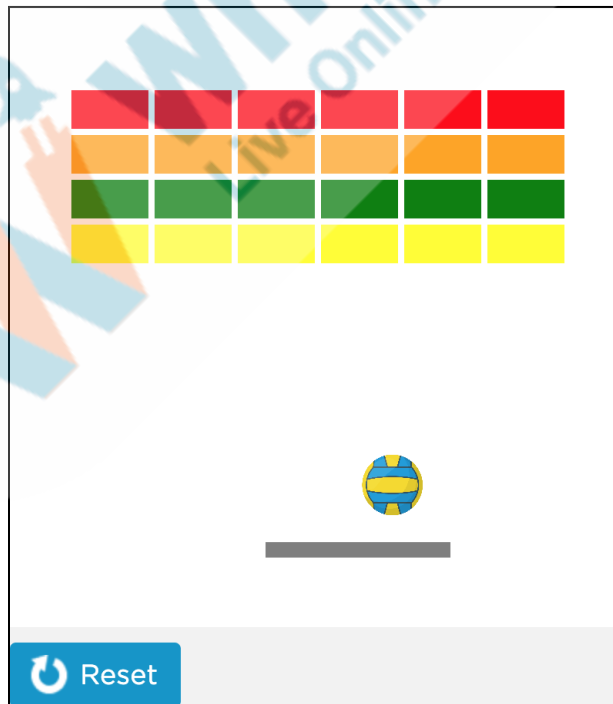
Remix.

Guide the student on calling the **createBrickRow(y, color)** function 3 times more to create 4 bricks in the game.

[Activity 2](#) link and remixes the code.

CODE:

```
7 ball.velocityX = 0;
8 ball.velocityY = 0;
9
10 var paddle = createSprite(200, 350, 120, 10);
11
12 createEdgeSprites();
13 createBrickRow(65, "red");
14 createBrickRow(65+29, "orange");
15 createBrickRow(65+29+29, "green");
16 createBrickRow(65+29+29+29, "yellow");
17
18
19 function createBrickRow(y, color) {
20   for(c=0; c<6; c++)
21   {
```

OUTPUT:

The teacher also encourages the student to enhance the graphics of the game by:

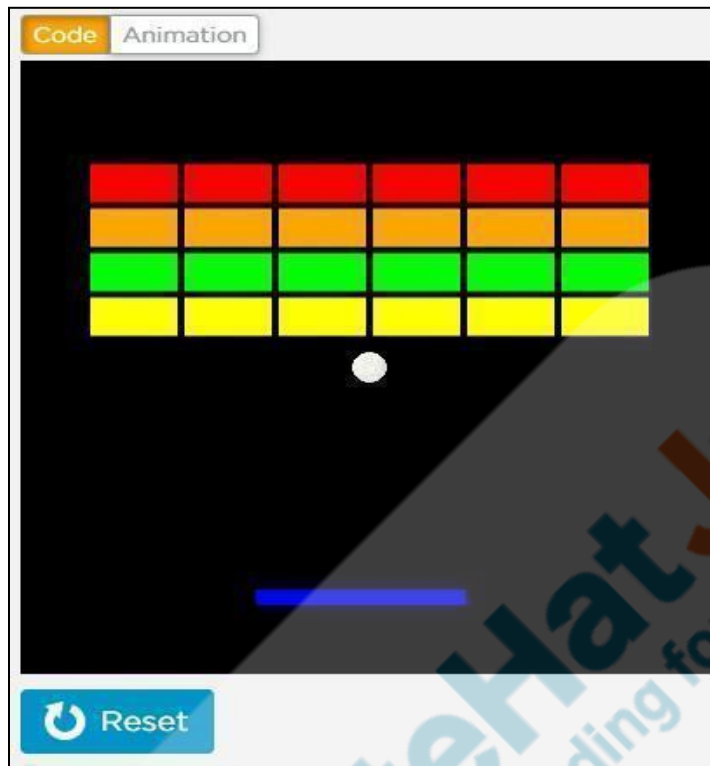
- 1) Set the background to '**black**' color.
- 2) Set the animation of the ball Sprite to a 'golf ball'. Set scale of the ball sprite to(**0.05** works well).
- 3) Set the paddle sprite's shape color property to **blue**.

CODE:

```
2
3 ball = createSprite(100, 100, 20, 20);
4 ball.setAnimation("golfball_1");
5 ball.scale = 0.05;
6
7 ball.velocityX = 0;
8 ball.velocityY = 0;
9
0 var paddle = createSprite(200, 350, 120, 10);
1 paddle.shapeColor = "blue";
2
3 createEdgeSprites();
4 createBrickRow(65, "red");
```

```
28 function draw(){
29   background("black");
30
31   paddle.x = World.mouseX;
32
33   if(paddle.x < 60){
34     paddle.x = 60;
35   }
36 }
```

OUTPUT:



Awesome! The graphics of the game look complete now.

Ask the child to run the game and see what is missing in the game?

Do you know how to make the ball **bounceOff()** from the bricks?

What should happen to the brick once the ball bounces off it?

Listen carefully since there are so many brick sprites with the same functionality. In such a case instead of writing 'bounce off' and 'destroy' commands for each one of them, we can group them together and write instructions for the group.

Ask the child to click on the '**Groups**' tab in **Toolbox** and find out a command to create a new group from the list below.

Help the child rename the variable name to **bricks**.

We have created the group, but we have not added sprites to the group yet. We will have to add the sprites to the group as soon as they are created inside the **for()** loop.

ESR: Ball is passing through the brick and not bouncing off.

ESR: Yes.

ESR: It should be destroyed.

| | |
|---|--|
| <p>Can you find a command to add a sprite to the group in the Toolbox?</p> <p>Drag and drop the group.add(sprite) command below the brick.shapeColor command inside the for() loop.</p> <p><i>Encourage the child to figure out and rename the group name and sprite name in the code to bricks and brick respectively.</i></p> | <p>ESR: group.add(sprite)</p> |
| <p>CODE:</p> <pre> 10 var paddle = createSprite(200, 350, 120, 10); 11 paddle.shapeColor = blue; 12 13 var bricks = createGroup(); 14 15 createEdgeSprites(); 16 17 createBrickRow(65, "red"); 18 createBrickRow(65+29, "orange"); 19 createBrickRow(65+29+29, "green"); 20 createBrickRow(65+29+29+29, "yellow"); 21 22 function createBrickRow(y, color) { 23 for(c=0; c<6; c++) 24 { 25 var brick = createSprite(65+54*c,y,50, 25); 26 brick.shapeColor = color; 27 bricks.add(brick); 28 } 29 } 30 </pre> | |
| <p>Great, now our sprite group 'bricks' is ready. We can now add instructions to have the ball bounce off of the bricks.</p> <p>Do you know the command to bounce off a sprite from the other sprite?</p> <p>The teacher writes a command to bounce the ball off the bricks.</p> | <p>ESR: sprite.bounceOff();</p> |
| <p>CODE:</p> | |


```

39  if(paddle.x > 340){
40      paddle.x = 340;
41  }
42  drawSprites();
43  ball.bounceOff(topEdge);
44  ball.bounceOff(leftEdge);
45  ball.bounceOff(rightEdge);
46  ball.bounceOff(paddle);
47  ball.bounceOff(bricks);
48  }
49

```

OUTPUT:



Great job! Our ball is bouncing off the bricks now.

But it is still not able to destroy the bricks. We will learn how to destroy the bricks in the next class. Also, we will add sound and score in our next class.

Isn't our game turning out to be exciting?

ESR: Yes.

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 10 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 starts slideshow from slides 21 to 35
 Refer to speaker notes and follow the instructions on each slide.

| Activity details | Solution/Guidelines |
|--|---------------------|
| <p>Run the presentation from slide 21 to slide 34.</p> <p>Following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> • Explain the facts and trivia. • Next class challenge. • Project for the day. • Additional Activity. | |
| QnA Session - Click on the in-class quiz | |
| Question | Answer |
| <p>Why do we use loops in coding?</p> <p>A. Executes a block of statements a certain number of times. B. Instead of typing a block of statements, again and again, we include it in a loop. C. Multiple instructions can be added in a loop. D. All of the above.</p> | D |
| <p>In the following piece of code, how many times will the text "I love coding" will be printed?</p> <div data-bbox="90 1491 997 1862" data-label="Code-Block"> <pre> Workspace 1 2 for(var i=1; i<=10; i=i+1){ 3 console.log(i); 4 console.log("I love coding"); 5 } 6 </pre> </div> | B |

| | |
|--|---|
| <p>A. 11 B. 10 C. 9 D. Infinitely</p> | |
| <p>Why do we use Groups in the program?</p> <p>A. To group all sprites together. B. To create collections of sprites with similar behavior. C. To assign the properties and functions to the group instead of the individual sprites. D. All of the above</p> | <p>D</p> |
| <p style="text-align: center;">End the quiz panel</p> | |
| <p style="text-align: center;">FUN WITH TECH FOR STUDENT TO PERFORM (MUST)</p> | |
| <ul style="list-style-type: none"> • Ask the student to press ESC key to come back to the panel • Guide the student to start Screen Share • The teacher gets into full screen | |
| <p>You did a fabulous job today. It was one of the trickier classes, but you did phenomenally well.</p> <p>It is now time to open the FUN WITH TECH.</p> | |
| <p><i>The teacher can guide the student to open a link from Student Activity 3.</i></p> <p>Today's fun activity is a sentiment analysis app that you too will be developing in your AI classes. This app uses AI (Artificial Intelligence) to read your sentences, recognize the keywords and tag it with the appropriate emojis.</p> <p>Consider this as your digital journal where you can write about the activities that you did in the day or about your feelings.</p> <p><i>The teacher can share the following instructions about how to play this.</i></p> | <p><i>Student opens the Student Activity 3.</i></p> |

1. To use the app just write about the activities you did in the given Text box.
2. Press Enter after you are done writing the sentence to analyze the sentiment behind the sentence.
3. You can save your entry by clicking on the save entry button.

While student is playing the game Teacher can mention:

The sentiment detection is made using Python language. To create this we have trained our application to recognize the emotion of the sentences. Big businesses and e-commerce use it to monitor brands and product sentiment by analyzing customer feedback. Some websites use this concept for rating newly released movies.

Did you enjoy finding emojis for your inputs?

Great!

You will also build such an app in your advanced/future classes.

For now, you can stop sharing the screen and Let's move ahead.

For teacher reference: this app will be created in class 114.

ESR: Yes.



PTM ANNOUNCEMENT : [Please bring your parents to the next class.](#)

<Student_Name>, the next class is going to be an interesting one! Do You know why? Because I plan on bringing joy to your parents by showcasing the activities built by you!

They will also get to know about all the fun parts you enjoyed throughout our classes, the concepts learned by you, and the concepts you will learn over the next few classes!

Isn't that exciting? So make sure your parents attend the next class!

Project Overview

ALIEN ATTACK

Goal of the Project:

In Class 4, you have learned the concept of loops to create rows of bricks and add them to the sprite group. You will use it in this project to create a group of **UFOs(Unidentified Flying Object)** in the sky.

In this project, you will have to practice and apply what you have learned in the class and create a sprite group with multiple UFOs flying in the sky and ready to attack.

Story:

Dodo loves to play games that have aliens and monsters. Daisy on the other hand likes to read and research about UFOs. Can you help Dodo and Daisy create an animation of UFOs flying over the sky and getting ready to attack the Dodo's world?

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

Bye Bye!

The student engages with the teacher over the project.

Guide the student to develop the project and share it with us.

Teacher ends slideshow



| Activity | Activity Name | Links |
|------------------------------------|--------------------------|---|
| Teacher Activity Link 1 | Introduction to for loop | https://studio.code.org/projects/gamelab/W28PwFP2CbtIBDNMiweGhd47jT3qqNe8HHfHcwh eNDc |
| Teacher Activity Link 1 (Ref Code) | Introduction to for loop | https://studio.code.org/projects/gamelab/W28PwFP2CbtIBDNMiweGhUqnFvwcFnXRTTv_0D Exw6w |
| Teacher Activity | Creating Bricks | https://studio.code.org/projects/gamelab/jlQoL |

| | | |
|---------------------------------------|--|---|
| Link 2 | | KwVh1_SRMbHe2V_-vhA2s54aCj9MkwuxWA Maeg |
| Teacher Activity Link 2 (Ref Code) | Creating Bricks | https://studio.code.org/projects/gamelab/speUDHhCnDvwv2-UnLortBxqHVgjk0rD2QOpi3J6HdU |
| Student Activity 1 | Print even numbers between 4 and 20 | https://studio.code.org/projects/gamelab/BnWLq5ls5RxRNPEjaB6pUmj201uN3e3djA9_sgeIVPM |
| Teacher Ref Code (even no.) | Print even numbers between 4 and 20 | https://studio.code.org/projects/gamelab/qlzKsd3-mOvwKKeb584ptaH_cZN52pRQ1bbaE2wEix0 |
| Student Activity Link 2 | Color the Bricks | https://studio.code.org/projects/gamelab/speUDHhCnDvwv2-UnLortHbNCWjDTB3czDCstfA7AYw |
| Teacher Ref Code (Breakout 1.4) | Breakout Game Stage 1.4 | https://studio.code.org/projects/gamelab/t_fdxXOC4uwccAlEicDqCWv0llwWGs1Ncp4DSPg27XM |
| Reference Game Code final | Ref Breakout Game Code | https://studio.code.org/projects/gamelab/ttl4IAup7OVMuoN_rxwTNUA7sXwCfuL5hyIFVRYUqFQ |
| Teacher Ref. Visual Aid Link | Visual aid link | https://s3-whjr-curriculum-uploads.whjr.online/066c9f82-ef3a-48f9-966e-1835185129c7.html |
| Teacher Ref. In-class Quiz | In-class quiz | https://s3-whjr-curriculum-uploads.whjr.online/6a8ced5f-0a56-405d-a576-1a98da0416a6.pdf |
| Row width Image | Offset Calculation | https://s3-whjr-curriculum-uploads.whjr.online/51b5735d-22d0-465f-9780-41586cd0943f.png |
| Row height Image | Vertical Gap Calculation | https://s3-whjr-curriculum-uploads.whjr.online/f53aaa1f-21d7-423e-9656-5046278bc99e.png |
| Teacher Activity 4 | FUN WITH TECH | Emotion Sentiment Detection |
| Student Activity 3 | FUN WITH TECH | Emotion Sentiment Detection |