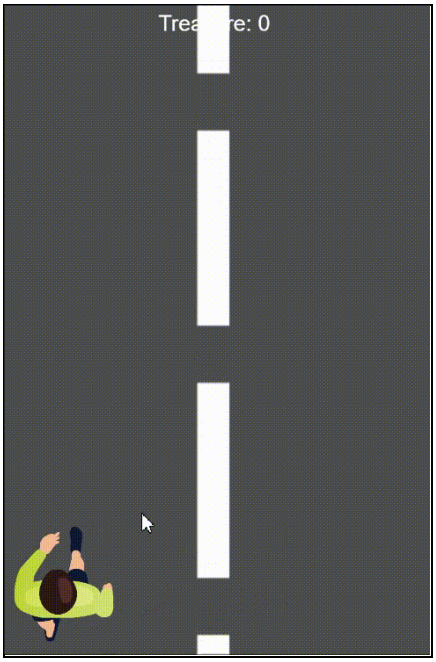
Topic	GAME ADAPTIVITY	
Class Description	**The student adds sound to the game. The student learns to make the game increasingly challenging for the player as the game progresses and also learn to add intelligence to the Trex so that it jumps automatically on seeing an obstacle.**	
Class	**C16**	
Class time	**50 mins**	
Goal	● Add sounds to the game. ● Increase the speed of the game for the player as the game progresses and the player's score increases. ● Add AI to the Trex.	
Resources Required	● Teacher Resources: ○ VS Code editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ● Student Resources: ○ VS Code Editor ○ 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 - 10mins		

Teacher starts slideshow from slides 1 to 13 Refer to speaker notes and follow the instructions on each slide.	
Teacher Action	**Student Action**
Hi, so good to see you again! How have you been? Are you excited to learn something new? ***Run the presentation from slide 1 to slide 3.*** **The following are the warm-up session deliverables:** ● Connecting students to the previous class.	**ESR**: I'm good, thanks. Yes, I am excited about it. Click on the slide show tab and present the slides.
QnA Session - Click on the in-class quiz	
Question	**Answer**
What will the following code block do? `treasureCollection= treasureCollection + 150;` A. It will decrease the value inside the **treasureCollection** variable by 150. **B.** It will increase the value inside the **treasureCollection** variable by 150. C. It will multiply the value inside the **treasureCollection** variable by 150. D. It will do nothing to the **treasureCollection** variable.	**B**
Select the correct code block to destroy cashGroup, diamondsGroup, jewelleryGroup and swordGroup.	**C**



A.
```
cashG.destroyEach;
diamondsG.destroyEach;
jwelleryG.destroyEach;
swordGroup.destroyEach;
```

B.
```
cashG.destroy();
diamondsG.destroy();
jwelleryG.destroy();
swordGroup.destroy();
```

C.
```
cashG.destroyEach();
diamondsG.destroyEach();
jwelleryG.destroyEach();
swordGroup.destroyEach();
```

D.
```
cashGdestroyEach();
diamondsGdestroyEach();
jwelleryGdestroyEach();
swordGroupdestroyEach();
```

Continue the warm-up session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 4 to slide 13 to set the problem statement. **Following are the warm-up session deliverables:** ● Explain the importance of making games challenging. ● Illustrate the importance of sound in games and how it relates to real-life situations.	Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow	
TEACHER-LED ACTIVITY 1 - 10mins	
Teacher Initiates Screen Share	
ACTIVITY ● **Create class and methods.** ● **Add a new JavaScript file in the index.html**	
Teacher Action	**Student Action**
Step 1: Teacher-led Activity *The teacher downloads the teacher Activity 1.* Now we are going to learn a different way of writing programs that is used by professional programmers worldwide, and it can make our code cleaner, and we can avoid writing the same type of code multiple times. This is called object-oriented programming. In OOP we have different terms which are very important to understand. These are classes and objects. First, what is a class?	**ESR:**

A class contains information about a category. A class can be understood as a blueprint. For example, before creating a car engineers create a blueprint of the car, and then create multiple cars which can be modified based on different users from that blueprint. Another example, consider a **class of dogs**, but as we know all dogs are not the same. We have multiple breeds such as labrador, pug, German shepherd, etc. But if you observe carefully, all the dogs have some things which are **common** in all the breeds, such as all dogs have 4 legs, 2 ears, 2 eyes, and they are furry, they wave their tail, they bark, etc. But depending upon their breed they have very specific features which make them different from other breeds, but they also have a lot of common features which make them look like dogs. So in this example, **Dog** is our class, which has all the information about how to create a new dog and what features to add to that dog. This information is called **attribute** and **functionality**. **Attributes** of a dog are height, length, fur color, and the **functionalities** are run, bark, howl. When we define a class in code, we can choose what attributes and functionalities we want to add to the class. Once we have the class ready then we can create multiple objects with the class. *The teacher writes the code.* *box.js file is already created and added in the index.html file to save time.*	The student may give some answers based on his/her understanding.

Teacher can explain the process of doing the same.
Now let's create a class for a box and let's see what are the important things we need to consider to create a class.

The benefit of OOP is that we can keep our code in multiple files which makes our code more manageable.

We create a **box.js** file and added that to the body section of our HTML file using:

<script src="box.js"></script> tag.

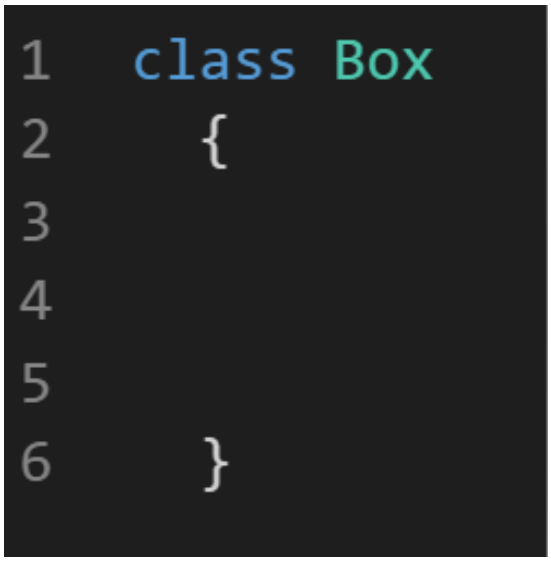
```html
<!DOCTYPE html><html lang="en"><head>
    <script src="p5.js"></script>
    <script src="p5.sound.min.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">

  </head>
  <body>
    <script src="sketch.js"></script>
    <script src="box.js"></script>


</body></html>
```

In this **box.js** file, we will **define our class**.

To define a class, first, we write the keyword **class** in lowercase then we write the name of the class which has the first letter in uppercase in our case, **Box**.

Then we begin with the parentheses, all the code related to the class will be kept inside these parentheses.

class Box
{
...

}	
	
In the class, first, we will define the properties or attributes of the class. What can be the attributes of a box? Great! But since we are creating the box on a canvas, we also need to specify the **x** and **y positions**. To add attributes we create a **constructor()** function. It's just like our normal function, but we don't need to write the keyword **function**. When our code will run, the constructor is the function to be executed. In this constructor, we will **define the properties**. This is interesting since when we define properties inside a class, we will use **this** keyword. So to set the position of the box, we will write **this.x = 100;** **this** keyword means that we are referring to this particular constructor. And similar for y, **this.y = 200,** **this.w = 50**—width of the box,	**ESR:** **Height**, **weight**, etc.

this.h = 50—height of the box.

```
class Box
  {
    constructor()
    {
      this.x =100;
      this.y = 200
      this.w = 50;
      this.h = 50;
    }


  }
```

Once we define the attributes through the **constructor()** function, we need to now define the functionality of the box.

First is the box that has to be displayed on the canvas, and this box will have a shape.

We will write a function to show the box on the canvas, we can call this function the show itself.

***Note**: It is not necessary to name the function as shown. We can give any name to our function, but for clarity, we always name the function based on what is the working of the function.*

Just like we defined the **constructor()** function, we will define the **show()** function, and we don't need to write the

function keyword.

Within the **show()** function we will add the **rect()** function because we want our box to be rectangular in shape.

rect(this.x,this.y,this.w,this.h)

Note: In class, whenever we need to use any property such as x, y position, we have to use the **this** keyword, otherwise the program will have an error that it does not understand the property.

```
class Box
  {
    constructor()
    {
      this.x =100;
      this.y = 200
      this.w = 50;
      this.h = 50;
    }

    show()
    {
      rect(this.x,this.y,this.w,this.h)
    }

  }
```

The last function we need for our class is to set the **speed of the box**.

To move the box we need to change the position; the bigger change we make in the position, the faster it will move.

We define a function named **set_speed(v)**.

Inside the function brackets, we will put an argument, so when the user calls this function the user can give any speed.

Inside the function, we can simply add the instruction to change **this.x** by adding the argument passed by the user, which will look like,

this.x = this.x + v. Here **v** is given by the user.

```
class Box
  {
    constructor()
    {
      this.x =100;
      this.y = 200
      this.w = 50;
      this.h = 50;
    }

    show()
    {
      rect(this.x,this.y,this.w,this.h)
    }

    set_speed(v)
    {
      this.x = this.x + v;
    }

  }
```

Code for the class is completed, now comes the fun

part—how would you use this class?

In the **sketch.js** file first, we will create a variable for the box, let's call it **var box1**.

This variable will become the object of the class.

To create the object in the **setup()** function we write **box1 = new Box()**.

The **new** keyword is used to create the object and after the **new** keyword, we add the name of the class.

Now, pay attention to the name of the class, its first letter is capital just like when we created the class.

```
var box1;

function setup() {
  createCanvas(600, 400);
  box1 = new Box();
}
```

We have the object of the class, now we can use this object to call the function of the class. Remember we made functions **show()** and **set_speed().**
We need to call these functions so that we can observe the effects on the object.

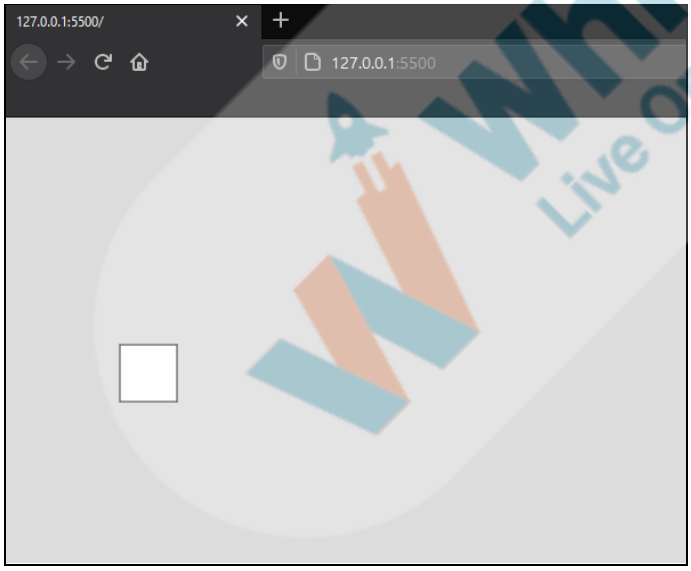
This we will do in the **draw()** function of the code.

To call the function from the class, write the name of the object followed by a dot then the function name. So it will be **box1.show()**.

Now we can run this code, and when we do that we can see a rectangle on the canvas.

```
function setup() {
  createCanvas(600, 400);
  box1 = new Box();
}

function draw() {
  background(220);
  box1.show();

}
```

Output:



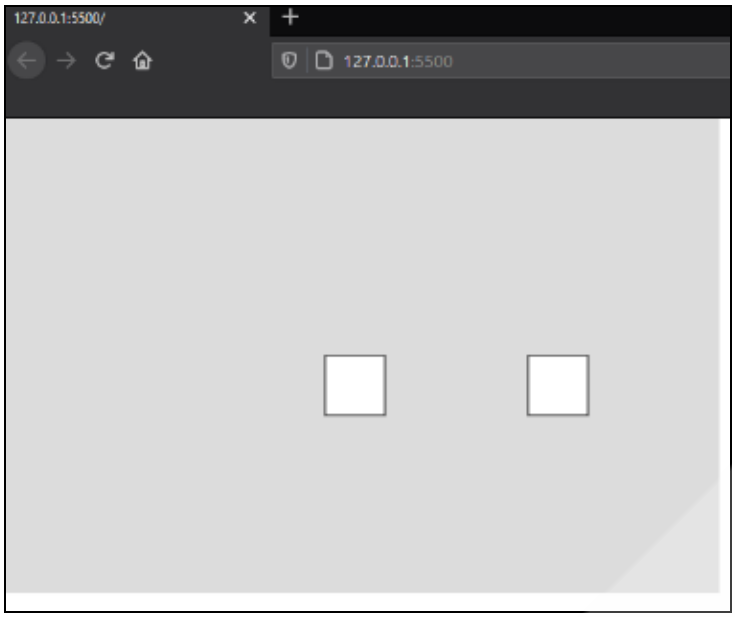
The next step is to call the function to set the speed of the box,
box1.set_speed(2); in the brackets of the function we can write any speed we want.

```
function draw() {
  background(220);
  box1.show();
  box1.set_speed(2);

}
```

Now, when we run the code, we should see the rectangle moving. We can experiment by changing the speed as well.	
The core benefit of OOP is we can create multiple objects and give them different speeds as shown in the code snippet below:	

```
var box1;
var box2;

function setup() {
  createCanvas(600, 400);
  box1 = new Box();
  box2 = new Box();
}

function draw() {
  background(220);
  box1.show();
  box1.set_speed(2);
  box2.show();
  box2.set_speed(1);

}
```

When we run this code we can see two rectangles moving at different speeds.	



Teacher Stops Screen share		
	So now it's your turn. Please share your screen with me.	
Teacher starts slideshow from slides 14 to 15 Refer to speaker notes and follow the instructions on each slide.		
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		**ESR:** Yes!

Teacher ends slideshow
Teacher Stops Screen Share
STUDENT-LED ACTIVITY 1 - 5 mins
• **Ask Student to press ESC key to come back to the panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**
ACTIVITY • **Create a Box class.** • **Write the function to change the width of the box.** • **Create an object for the box.** • **Call the function using the box object.**

Teacher Action	Student Action
Step:2 Student-led Activity In the last activity, we learned how to create a class and then create objects for the class. *box class code is provided as boilerplate code,students only need to add the set_width() function in the box.js file.*	*The student downloads the* Student Activity 1 *for the activity and runs in the VS Code.*

```
class Box
  {
    constructor()
    {
      this.x =100;
      this.y = 200
      this.w = 50;
      this.h = 50;
    }

    show()
    {
      rect(this.x,this.y,this.w,this.h)
    }

    set_width(l)
    {
      this.w = l;
    }
  }
```

We have the Code from the last activity, where we defined a **Box** class and created a **constructor()** and **show()**function,

The change we are performing is that we are creating a new function to change the width of the Box, like earlier we set the velocity. Here we are setting the width of the Box by creating a function **set_width()**. This function will also accept the argument from the user, it will pass the desired value of the width, and it will change the width of the Box.

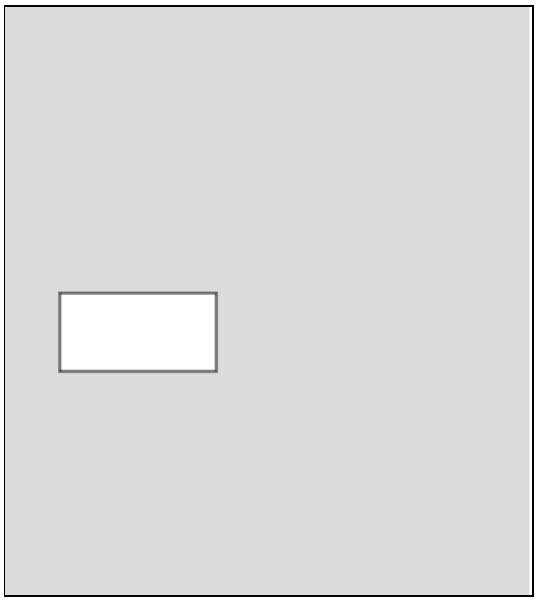
Once we are done with the code for the class, go to the

sketch.js file, and create the object of the **Box** class, call the **show()** function, and run the code to see if the box has the width which we specified in the **constructor()** function.

Then we write the **set_width()** function to see how we changed the width from original to the value passed in the **set_width()** function.

```
var box1;

function setup() {
  createCanvas(400, 400);
  box1 = new Box();

}

function draw() {
  background(220);
  box1.show();
  box1.set_width(100);

}
```

Output:



Here we can see that the width of the box object is changed. This is the power of the Object-Oriented Programming

With these two activities, we learned about the very basics of the OOP. Such as how to create a class and define methods for the class. To create the object of the class and call the methods using this object.

In the next few classes, we are going to learn more interesting concepts which will help you in writing more complex codes.

Teacher Guides Student to Stop Screen Share

Teacher starts slideshow : Slides 16 to 25

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

The teacher will set the context for the Trex Game activity. Please refer to the Visuals for the notes and explanation.

Teacher ends slideshow	
TEACHER-LED ACTIVITY 2 - 10mins	
Teacher Initiates Screen Share	
CHALLENGE • **Review the code from the last class.** • **Increase the speed of the game as the game progresses or as the player scores higher.** • **Adding sound to the game.**	
Teacher Action	**Student Action**
Step 2: Teacher-led Activity *Teacher opens **Teacher Activity 2**.* *Let us quickly review what we did in the last class.*	*The student goes through each block of code and explains what they are doing.* *The student also recalls and reviews the parts of code written in lesson C15.*
Awesome! Time to add the sounds. Do you remember what are the different sounds in the Trex Runner game, and when do we hear them?	**ESR:** There are three different sounds: • When the Trex jumps. • Every time when the Trex crosses a 100 milestone. • When the Trex dies.
Great. All these sounds are already uploaded in your Student Activity Link as: • jump.mp3 - Trex jump sound • die.mp3 - Trex dying sound	*Student listens.*

• checkPoint.mp3 - Trex crossing 100 milestone sound.	
Do you remember the function we used to play the sound?	**ESR:** play()
Amazing! Why don't you open your *student activity 2* and start adding these sounds in the game? It is going to be fun.	**ESR:** Varied.
Teacher starts slideshow : Slide 26-27	
Run the presentation for slide 26 to 27 to set the student activity context.	
Here's a challenge for you. You need to: • Add suitable sound to the game based on certain conditions. • Make the game challenging for the player, it should get tougher as the player advances in the game. • Make the Trex jump when it approaches an obstacle automatically. It would be interesting to make our Trex smart with some AI.	
Teacher ends slideshow	
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY 2 - 10mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Fullscreen.**	

ACTIVITY • **Add sound to the game.** • **Increase the speed of the game as the player's score increases.** • **Add AI to the Trex to jump automatically on seeing an obstacle.**	
Teacher Action	**Student Action**
Step 3: **Student-led Activity** If you notice, sounds are already added in **preload().** Let us add the jump sound first. How do you add sound to your game? *Guide the student to play the 'jump sound' whenever the Trex jumps in the game.*	*The student opens* ***Student Activity 2****.* **ESR:** varied *The student adds the code to play the jump sound when the Trex jumps in the game. The student runs the code to test it.*
Once the sound is loaded in the code, we need to write the code to play the sound based on the events. When the user presses the space key, the Trex will jump and the jump sound should be played. We now will write the code for that.	

```
if(gameState === PLAY){
  //move the
  gameOver.visible = false;
  restart.visible = false;

  ground.velocityX = -(4 + 3* score/100)
  //scoring
  score = score + Math.round(frameCount/60);

  if(score>0 && score%100 === 0){
      checkPointSound.play()
  }

  if (ground.x < 0){
    ground.x = ground.width/2;
  }

  //jump when the space key is pressed
  if(keyDown("space")&& trex.y >= 100) {
      trex.velocityY = -12;
      jumpSound.play();
  }

  //add gravity
  trex.velocityY = trex.velocityY + 0.8

  //spawn the clouds
  spawnClouds();
```

Now, add the 'die sound' when the Trex collides with the obstacle. *Guide the student to play the 'die sound' when the Trex collides with the obstacle.*	*The student writes the code to play the dying sound when the Trex collides with the obstacle.* The *student runs the code and plays the game to test it.*

```
JS sketch.js ×
JS sketch.js > draw
107        trex.velocityY = trex.velocityY + 0.8
108
109        //spawn the clouds
110        spawnClouds();
111
112        //spawn obstacles on the ground
113        spawnObstacles();
114
115        if(obstaclesGroup.isTouching(trex)){
116
117                gameState = END;
118                dieSound.play()
119
120        }
121    }
122    else if (gameState === END) {
123        gameOver.visible = true;
124        restart.visible = true;
125
126        ground.velocityX = 0;
127        trex.velocityY = 0
128        //change the trex animation
129        trex.changeAnimation("collided",trex_collided);
130
```

Amazing. Isn't it fun?	**ESR:** Yes!
Now we have to add a sound every time the score crosses the 100 milestone - that is we play sound at 100, 200, 300, and so on... How do you think we can do that?	**ESR:** Varied.

What would be the remainder if we divide these numbers by 100? What is the mathematical operator in the coding language which gives us the remainder?	**ESR:** 0 **ESR:** The modulo **(%)** sign.
Good. Let us use **%** on the **score** and check if it is **0**. If it is **0**, we will play the checkpoint sound. We also need to make sure that **count > 0**. We can join these two conditions using **&&** (**AND** operator).	*The student writes the code to play the checkpoint sound whenever the Trex crosses the 100th milestone.* *The student runs the code to verify the output.*

sketch.js

sketch.js > draw

```
82
83      if(gameState === PLAY){
84        //move the
85        gameOver.visible = false;
86        restart.visible = false;
87
88        ground.velocityX = -(4 + 3* score/100)
89        //scoring
90        score = score + Math.round(frameCount/60);
91
92        if(score>0 && score%100 === 0){
93            checkPointSound.play()
94        }
95
96        if (ground.x < 0){
97          ground.x = ground.width/2;
98        }
99
100       //jump when the space key is pressed
101       if(keyDown("space")&& trex.y >= 100) {
102           trex.velocityY = -12;
103           jumpSound.play();
104       }
105
106       //add gravity
107       trex.velocityY = trex.velocityY + 0.8
108
109       //spawn the clouds
110       spawnClouds();
111
112       //spawn obstacles on the ground
113       spawnObstacles();
```

Wow! The game is definitely fun to play now. Sounds make such a huge difference. There is one thing though - the Trex runs at the same speed all the time! Ideally, the game should become more challenging as it progresses and the player's score increases. What can we do about that?	**ESR:** Change the speed depending on the score.
Yes! Let's say we will increase **by 1** every time the Trex score increases by **+100**. This means the speed will keep increasing as the score increases, making it more challenging for the player. For example, if the score value is 205 then we will divide the score by 100, the result will be 2.05. This will be added to the speed of the obstacle.The negative sign is assigned to the speed because we want obstacles to move from right to left on the canvas. In JavaScript, we can simply divide the score by 100 and add it to the speed. Whose speed are we changing by the way? Perfect!	**ESR:** The speed of the ground and the obstacles. *The student writes the code to increase the speed for the ground and the obstacles with the score.*

We are changing the speed of the obstacles and keeping it as negative, because we want the obstacles to move from right to left on the canvas. Let's do it!	*The student runs the code to check the output.*

JS sketch.js

JS sketch.js > draw

```
147
148   function spawnObstacles(){
149    if (frameCount % 60 === 0){
150      var obstacle = createSprite(400,165,10,40);
151      obstacle.velocityX = -(6 + score/100);
152
153       //generate random obstacles
154       var rand = Math.round(random(1,6));
155       switch(rand) {
156         case 1: obstacle.addImage(obstacle1);
157                 break;
158         case 2: obstacle.addImage(obstacle2);
159                 break;
160         case 3: obstacle.addImage(obstacle3);
161                 break;
162         case 4: obstacle.addImage(obstacle4);
163                 break;
164         case 5: obstacle.addImage(obstacle5);
165                 break;
166         case 6: obstacle.addImage(obstacle6);
167                 break;
168         default: break;
169       }
170
171       //assign scale and lifetime to the obstacle
172       obstacle.scale = 0.5;
173       obstacle.lifetime = 300;
174
175      //add each obstacle to the group
176       obstaclesGroup.add(obstacle);
177    }
178   }
179
```

Is there any significant change in the speed? Let's change the speed by **3** times as compared to what we did before. We can do that by multiplying the count by **3**.	**ESR:** No! *The student adds the code to increase the speed by 3 times the previous number.* *The student runs the code to test.*

sketch.js

sketch.js > draw

```
82
83      if(gameState === PLAY){
84        //move the
85        gameOver.visible = false;
86        restart.visible = false;
87
88        ground.velocityX = -(4 + 3* score/100)
89        //scoring
90        score = score + Math.round(frameCount/60);
91
92        if(score>0 && score%100 === 0){
93            checkPointSound.play()
94        }
95
96        if (ground.x < 0){
97          ground.x = ground.width/2;
98        }
99
100       //jump when the space key is pressed
101       if(keyDown("space")&& trex.y >= 100) {
102           trex.velocityY = -12;
103           jumpSound.play();
104       }
105
106       //add gravity
107       trex.velocityY = trex.velocityY + 0.8
108
109       //spawn the clouds
110       spawnClouds();
111
112       //spawn obstacles on the ground
113       spawnObstacles();
```

	Does the speed increase appreciably as the game progresses?	**ESR:** Yes!

Let us do one more fun thing.	*Student experiment with the code and test it by running it.*
All programmers are always innovative. Why do we have to press the space key every time, right? Let us make the Trex artificially intelligent so that it jumps on its own when it sees the obstacle.	**ESR:** Varied.
Can you tell me how can we do that? We will increase the size of the Trex collider so that the Trex can see the obstacle before actually touching it. Whenever the Trex collider touches the obstacle, we will make the Trex jump. *Guide the student to add AI to the Trex.*	**ESR:** Varied

```
etch.js > draw
    gameOver = createSprite(300,100);
    gameOver.addImage(gameOverImg);

    restart = createSprite(300,140);
    restart.addImage(restartImg);

    gameOver.scale = 0.5;
    restart.scale = 0.5;

    invisibleGround = createSprite(200,190,400,10);
    invisibleGround.visible = false;

    //create Obstacle and Cloud Groups
    obstaclesGroup = createGroup();
    cloudsGroup = createGroup();

    console.log("Hello" + 5);

    trex.setCollider("rectangle",0,0,400,trex.height);
    trex.debug = true

    score = 0;

  }

  function draw() {

    background(180);
```

Once the collider radius is set we will add the condition that when an obstacle is in contact with the Trex, the Trex will jump and the jump sound will be played.

sketch.js

sketch.js > draw

```
112     //spawn obstacles on the ground
113     spawnObstacles();
114
115     if(obstaclesGroup.isTouching(trex)){
116
117         trex.velocityY=-12;
118         jumpSound.play();
119         // gameState = END;
120         // dieSound.play()
121
122     }
123   }
124   else if (gameState === END) {
125       gameOver.visible = true;
126       restart.visible = true;
127
128       ground.velocityX = 0;
129       trex.velocityY = 0
130       //change the trex animation
```

Our Trex's code has become intelligent now. It can never die, and you can score as many points as you want. Isn't watching the jumping Trex fun?	**ESR:** Yes!

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 Mins

Teacher starts slideshow from slide 28 to slide 38

Activity details	Solution/Guidelines
Run the presentation from slides 28 to slide 38. **Following are the wrap-up session deliverables:**	Guide the student to

• **Explain the facts and trivias** • **Next class challenge** • **Project for the day** • **Additional Activity**	develop the project and share it with us.
Quiz time - Click on the in-class quiz	
Question	**Answer**
Which of the following is true about Object-Oriented Programming? A. It can make our code cleaner. B. We can avoid writing the same type of code multiple times. C. It uses classes and object structure. **D. All of the above.**	**D**

Which of the following snippets of codes would create a class by the name Box with properties as x, y, w, and h?
 	
A.
`class Box{`
`  constructor(){`
`  this.x = 100`
`  this.y = 200;`
`  this.w = 50;`
`  this.h = 50;`
`  }`
`}`
 	
B.
`class Box(){`
`  constructor{`
`  this.x = 100`
`  this.y = 200;`
`  this.w = 50;`
`  this.h = 50;`
`  }`
`}`
	A

C.
```
class Box{
    constructor{
        this.x = 100
        this.y = 200;
        this.w = 50;
        this.h = 50;
    }
}
```

D.
```
class Box(){
    constructor(){
        this.x = 100
        this.y = 200;
        this.w = 50;
        this.h = 50;
    }
}
```

How to create an object of a class? A. Class_name = new object_name() **B. Object_name = new Class_name()** C. Object_name = Class_name D. Object_name = new Class_name	**B**

End the quiz panel

FEEDBACK

- **Encourage the student to make reflection notes in Markdown format.**
- **Complement the student for her/his effort in the class.**
- **Review the content of the lesson.**

Step 4: **Wrap-Up** Let us quickly wrap-up today's class. What did we do today?	**ESR:** - We added sounds to make the game fun. - We added some game adaptivity - the game now keeps becoming more challenging as the game progresses. - We made our Trex intelligent - it can jump all

	the obstacles on its own now.
Awesome. In the next class, we are going to learn something important called "scope" in programming. We are also going to start working on a new Game - "Pirate Invasion!!" Isn't it fun!!	**ESR:** Yes!
You get Hats Off for your excellent work! Looking forward to the next class!	*Make sure you have given at least 2 Hats Off during the class for:* 
*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.** **Project Overview** **CUT YOUR FRUITS** **Goal of the Project:** Today, you have learned how to increase the speed of the game by increasing the velocity of the ground and obstacles after a certain score. You also learned to add sound when the Trex jumps and also when the Trex dies. In this project, you have to apply what you have learned in the class and create a Fruit Cutting game by adding sound effects to the game to make it more interesting and	*Students engage with the teacher over the project.*

increase the velocities of fruits and monsters when the score reaches a certain point. **Story:** Sheena is learning new culinary skills every day. While searching on the internet for more ways of decorating and cutting fruits, she came across an interesting game to cut fruits. She really liked it. She has already completed half the design of the game. But now she has asked for your help to complete the game. I am very excited to see your project solution and I know you will do really well. Bye Bye!	
Teacher ends slideshow	
Teacher Clicks ✖ End Class	

Additional Activities
Encourage the student to write reflection notes in their reflection journal using Markdown.
 	
Use these as guiding questions:
• What happened today?
  ○ Describe what happened.
  ○ The code I wrote.
• How did I feel after the class?
• What have I learned about programming and developing games?
• What aspects of the class helped me? What did I find difficult?
	The student uses the Markdown editor to write her/his reflections in the reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Template Code	https://github.com/pro-whitehatjr/pro-c16-ta1-template
Teacher Reference	Reference code for Teacher Activity 1	https://github.com/pro-whitehatjr/pro-c16-ta-1-reference-code
Teacher Activity 2	Trex Stage 6	https://github.com/pro-whitehatjr/PRO_C16_LP_TA_2
Teacher Reference	Teacher reference code for student Activity 1	https://github.com/pro-whitehatjr/pro-c16-sa1-reference
Student Activity 1	Template Code	https://github.com/pro-whitehatjr/Pro-c16-sa1-template
Student Activity 2	Trex Stage 6.5 (template code)	https://github.com/pro-whitehatjr/PRO_C16_LP_SA2
Teacher Reference Final code	Trex Stage 7 (solution code)	https://github.com/pro-whitehatjr/PRO_C16_LP_SA_3
Visual Aid	Visual Aid	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C16-withcues.html

In-class Quiz	In-class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/e706d1f1-2ad6-44c3-b84a-6924879c4a2c.pdf
Project Solution	Cut Your Fruits	https://github.com/pro-whitehatjr/Project_C16_Fruit_Ninja