

Topic	Game States and Groups	
Class Description	Students learn to assign different game behavior for different states in the game. Students also learn to add similar objects in a group and assign the same behavior to all the objects in that group.	
Class	C14	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> • Create two game states - PLAY and END. Assign different game behavior for the different states. • Group similar game objects together in a group and assign the same behavior to all the objects in the group. • Create colliders for the trex and each obstacle. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources: <ul style="list-style-type: none"> ○ 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

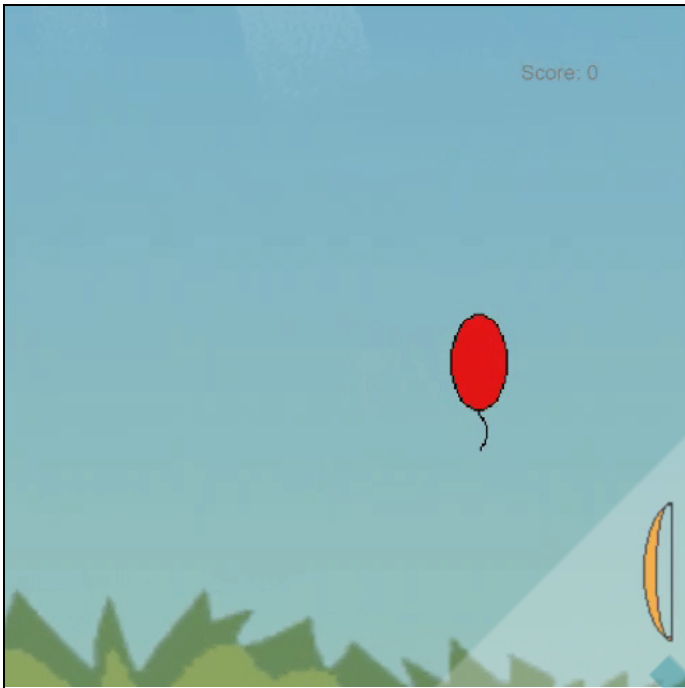
WARM-UP SESSION - 10 mins



Teacher starts slideshow from slides 1 to 5

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

Teacher Action	Student Action
<p><i>Hi, so good to see you again! How have you been? So this is going to be the first class of the first module. Are you excited to learn something new?</i></p> <p>Run the presentation from slide 1 to slide 3</p> <p>Following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> Recall the progress in the game from the last class. Recall the bugs identified in the last class. 	<p>ESR: Thanks, Yes I am excited about it.</p> <p>Click on the slide show tab and present the slides</p> <p>Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</p>
QnA Session	
Question	Answer
<p>Select the line of code to generate a random number from 1 to 4.</p> <p>A. <code>Math.round(random(1,4));</code> B. <code>Math.round(randomNumber(1,4));</code> C. <code>random(1,4);</code> D. <code>Math.round(random(0,3));</code></p>	A
<p>Select the block of code that would call the function <code>redBalloon()</code>, if the var <code>select_ballon</code> is equal to 1.</p>	D



- A.

```
if (select_balloon == 10) {  
    redBalloon();  
}
```
- B.

```
if (select_balloon == 1) {  
    greenBalloon();  
}
```
- C.

```
if (select_balloon != 1) {  
    redBalloon();  
}
```
- D.

```
if (select_balloon == 1) {  
    redBalloon();  
}
```

Continue the warm-up session

Activity details	Solution/Guidelines
Run the presentation from slide 4 & slide 5 to set the problem statement.	



Teacher ends slideshow

TEACHER-LED ACTIVITY 1- 10 mins

Teacher Initiates Screen Share

CHALLENGE

- Store data in a JavaScript object.
- Access data from the JavaScript object.

Teacher Action	Student Action
<p>Step 1: Teacher-led Activity</p> <p><i>Teacher downloads and runs the Template Code code in VS Code Editor.</i></p> <p>In programming there are multiple ways to store data in the code, we have been using variables for a while. In the last few classes we learned to use the array.</p> <p>But in this class we are going to learn about a very important way to store data in the code.</p> <p>This is so important that without this method your internet will crash in seconds, most of the data on the internet comes in the form of this data structure.</p> <p>This is called JavaScript objects. In the future lessons when we are going to learn about APIs we are going to use this extensively.</p> <p>What is a JavaScript object?</p> <p>This allows us to store the data at one place, which belongs to a certain object.</p>	<p>ESR:</p> <p>Student answers based on his/her understanding.</p>

For example: You are a student, you have certain information associated with you such as your name, in which school you study, your favorite subject, your marks [30,45,35] etc.

It is not a good practice to keep this data in an array because it involves different data types, such as strings, characters, integers, even an array itself.

Also if you want to store the data of multiple students then it will be very tedious to access the data of the correct student, because we have to remember the index of each student's data points.

Here, JavaScript objects come to our help.

This will allow us to store your data at one place and then we can access and change it very easily, which we are going to learn with this coding activity.

The teacher writes the code.

To define an object, first we declare a variable name and set it equal to empty curly brackets.

```
var Student = { };
```

At the end of the parentheses there will be a semicolon. Once we have our empty object ready, we can start adding the data in this.

Since this object is named as student, we are going to add data which is relevant to the student.

```
1  var Student = {  
2  
3  
4  };
```

Like we discussed earlier, for students we have data such as name, class, roll_no, favorite_subject, and marks of each subject.

Now we will add this data in the object.

To add the data, first we need to understand how data is stored inside the object.

It is stored in a key value pair; key is the name defining the value.

In our case roll_no is the key and the actual value of the roll no as 10 is the value.

Value is assigned to a key by using a colon (:) unlike an equal symbol.

Now we will add the data of the student in the object.

The teacher writes the code.

```
1  var Student = {  
2    name: "Sammy",  
3    class: 7,  
4    roll_no:21,  
5    favorite_subject: "coding",  
6    marks : [30,35,40,50]  
7  
8  };
```

Once we got the object defined, now we can use it in our code.

We can display any of these values on the console.

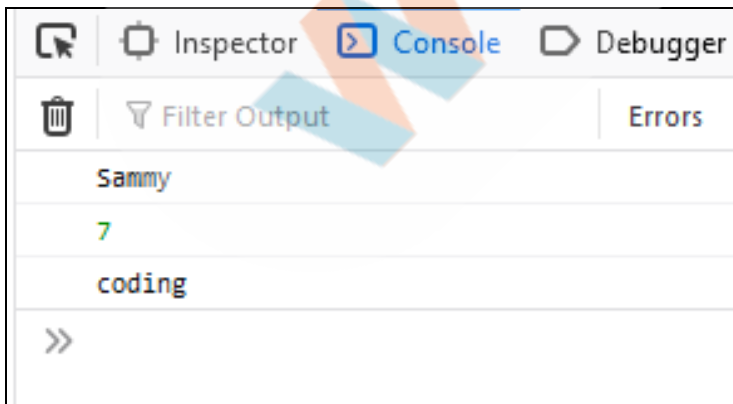
We can also change the values.

Let's see a few examples.

```
1  var Student = {
2    name: "Sammy",
3    class: 7,
4    roll_no: 21,
5    favorite_subject: "coding",
6    marks : [30, 35, 40, 50]
7
8  };
9
10 function setup() {
11
12   createCanvas(400, 400);
13   console.log(Student.name)
14   console.log(Student.class)
15   console.log(Student.favorite_subject)
16
17 }
```

To access the data from an object, we use the name of the object .(dot) and the name of the key you want the value of.

If we want to see the name of the student we can say **Student.name**.



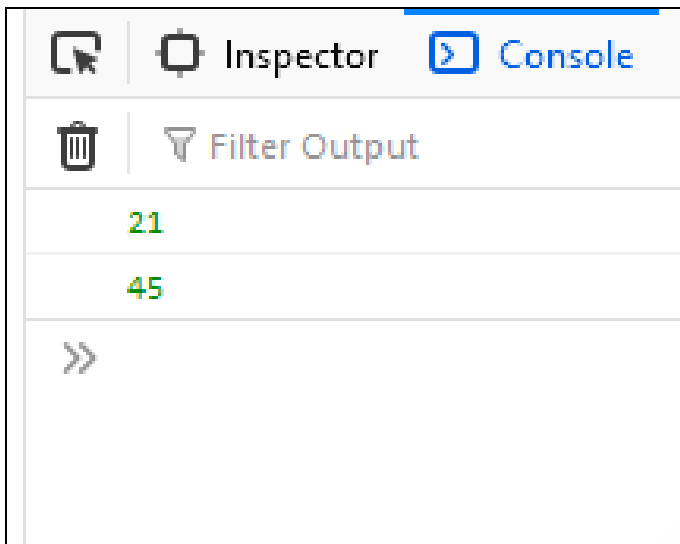
We can change the values as well.

To change the value we can write **Student.roll_no = 40**.

Here we can see the old values and the new values on the console.

```
1  function setup() {  
2  
3    createCanvas(400, 400);  
4    //changing the roll no  
5    console.log(Student.roll_no);  
6    Student.roll_no = 45;  
7    console.log(Student.roll_no);  
8  }
```

OUTPUT:



Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 5 mins

- 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

- Create an object for a ball by defining properties such as color, speed, etc.
- Access the properties in the code.
- Change the properties and make the ball move.



Teacher starts slideshow for slide 6 & 7

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

Teacher Action	Student Action
<p>Step-3: Student-led Activity</p> <p>In the last activity, we have created an object to represent a student, and we were able to access and change the properties.</p>	<p>Student downloads the code for Student activity template code and runs in the VS Code Editor.</p>

<p>In this activity, you are going to create an object for a ball. Before we start, we need to think about the properties of the ball.</p> <p>What do you think, what can be the properties of the ball?</p> <p>To create the ball in the code we first need the x and y coordinates of the ball, assuming the ball is of circular shape we need the radius of the ball as well.</p>	<p>ESR: Color, shape, position etc.</p>
<p>Before moving forward, let's create the object and these properties.</p> <p><i>The teacher guides the student to create an object.</i></p>	
	
<p>Apart from these we can have properties like speed in x direction and y direction and we can give colors as well.</p> <p>But instead of keeping 1 color, we can make an array of colors, so that we can change between multiple colors later in the code.</p>	

```
var ball = {  
  x:20,  
  y:30,  
  r:30,  
  xspeed:0,  
  yspeed:0,  
  color:["blue","red","green","purple"],  
};
```

We got our object ready, now we are going to create a ball using these properties. Since balls are circular shaped, we can use the **circle()** function to create the ball.

For the coordinates and the radius, we will take from the object.

How do we access the properties from the object?

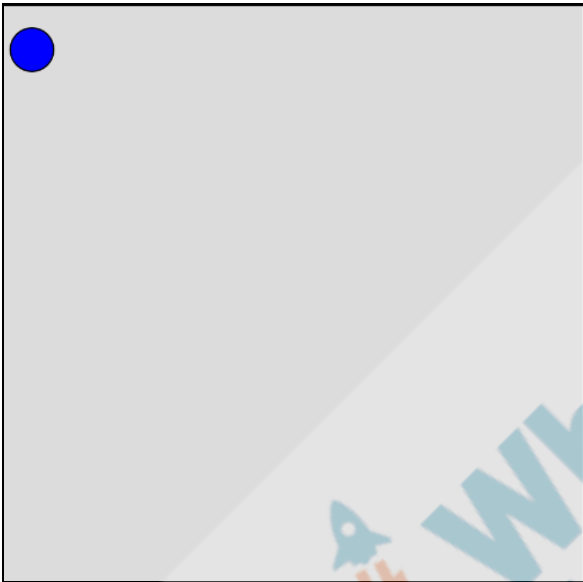
Great! So let's get the x, y position and the radius of the ball. Then we will get the color as well.

To fill color in the ball, we will use **fill()** function. In the fill function we will pass the first element of the color array. You can try to change the colors of the ball.

ESR:

Using the name of the object, .dot() and the name of the property.

```
function draw()
{
  background(220);
  circle(ball.x,ball.y,ball.r);
  fill(ball.color[0]);
}
```



We got the object on the canvas, now we will make it move.

But if you notice, while creating the object, we kept the x & y speed as 0, so if we want the ball to move, we need to change the speed.

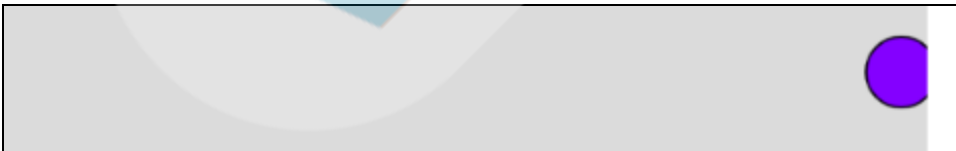
So we will set the **ball.xspeed = 1**. In the p5.js code, the draw function is running continuously and it is drawing the ball at the same position, but what if we change the ball's position as the draw function is running, we would be able to see the ball moving. Let's do this.

To the move the ball we can say **ball.x = ball.x+ball.xspeed**

Which means that they change the position of the ball by xspeed. Now if you run the code, you will be able to see the ball moving in the x direction (horizontal direction). But once it reaches the wall, it will go away.


```
function draw()
{
  background(220);
  circle(ball.x,ball.y,ball.r);
  fill(ball.color[0]);
  ball.xspeed = 1;
  ball.x= ball.x + ball.xspeed;
}
```

Output:



You can try to change the y speed as well and move it in the y (vertical direction).

Teacher Guides Student to Stop Screen Share

TEACHER LED ACTIVITY 2 - 10 mins	
Now we will work on our T rex Game.	
Teacher Initiates Screen Share	
<p style="text-align: center;"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Create two groups where all the cloud objects and the obstacle objects can be added. • Create two conditional statements to assign different behavior to the game objects in the two game states. 	
<p style="text-align: center;">  Teacher can show slideshow from slides 8 to 23 Refer to speaker notes and follow the instructions on each slide. </p>	
Teacher Action	Student Action
<p>Step 3: Teacher-led Activity <i>The teacher opens Teacher Activity 2.</i></p> <p>This is the code from our previous class. Let's quickly go through it so that we recall what we were doing. Comments which we have added in our code will make it easy to do so.</p> <p><i>The teacher takes the student through each block of code and asks the student to explain what those blocks of code are doing and what is their role in the game.</i></p>	<p><i>The student goes over each block of code with the teacher and explains what the blocks of code do in the program and why they are there.</i></p>

Before we go into game states, we will do one cool thing. We are spawning several cloud and obstacle (cactus) objects every few frames. We can group all of these objects into a single group.

Using group properties we can program the behavior of all the objects in a single stroke.

Let's see how.

Let's create two groups - called **obstaclesGroup** and **cloudsGroup**.

Group is a predefined class in p5 which can store different sprites and apply some rules on all the sprites.

The student observes and learns.

```
invisibleGround = createSprite(200,390,400,10);
invisibleGround.visible = false;

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

console.log("Hello" + 5);

score = 0;
}

function draw() {
  background(180);
  //displaying score
  text("Score: "+ score, 500,50);
```


We will add every cloud object or obstacle object we are creating to these two respective groups.

After this we can program for the group properties on all the objects together.

The student observes and learns.

```
case 2: obstacle.addImage(obstacle2);
        break;
case 3: obstacle.addImage(obstacle3);
        break;
case 4: obstacle.addImage(obstacle4);
        break;
case 5: obstacle.addImage(obstacle5);
        break;
case 6: obstacle.addImage(obstacle6);
        break;
default: break;
}

//assign scale and lifetime to the obstacle
obstacle.scale = 0.5;
obstacle.lifetime = 300;

//add each obstacle to the group
obstaclesGroup.add(obstacle);
}
}
```

```
function spawnClouds() {  
  //write code here to spawn the clouds  
  if (frameCount % 60 === 0) {  
    cloud = createSprite(600,300,40,10);  
    cloud.addImage(cloudImage);  
    cloud.y = Math.round(random(280,320));  
    cloud.scale = 0.4;  
    cloud.velocityX = -3;  
  
    //assign lifetime to the variable  
    cloud.lifetime = 134;  
  
    //adjust the depth  
    cloud.depth = trex.depth;  
    trex.depth = trex.depth + 1;  
  
    //adding cloud to the group  
    cloudsGroup.add(cloud);  
  }  
}
```

Let us introduce a variable which will hold the value of the game state.

Game State could be PLAY or END.

Let us set the initial state to be PLAY when we run the code.

We use CAPITAL LETTERS in the name of those variables which hold constant values that do not change inside the program.

The student observes and learns.

```
var PLAY = 1;
var END = 0;
var gameState = PLAY;

var trex, trex_running, trex_collided;
var ground, invisibleGround, groundImage;

var cloudsGroup, cloudImage;
var obstaclesGroup, obstacle1, obstacle2, obstacle3, obstacle4, obstacle5, obstacle6;

var score;
```

We will put an ‘if and **else if**’ condition inside the function **draw()**.

What do you think “if and **else if**” block would do?

The teacher writes the two conditions as shown below.

The student observes and learns.

ESR: It will do one thing if one condition is satisfied and it will do another thing if the other condition is satisfied.

```
function draw() {
  background(180);
  //displaying score
  text("Score: "+ score, 500,50);
  score = score + Math.round(frameCount/60);

  if(gameState === PLAY){

  }
  else if (gameState === END) {

  }
}
```

There are some behaviors in the game which we want when the **gameState** is **PLAY**. We will put **if (gameState === PLAY)** code behaviors related to these statements inside .

There are other behaviors in the game we want when the **gameState** is **END**. We will put code behaviors related to those statements inside the **else if (gameState === END)**.


There are some behaviors in the game we want irrespective of the **gameState**. We will put them outside the **if else** condition.

Let us do one for an example and then you can do the remaining.

Do we want the **background(180)** behavior at all times?

The student observes, learns and asks questions.

ESR: Yes.

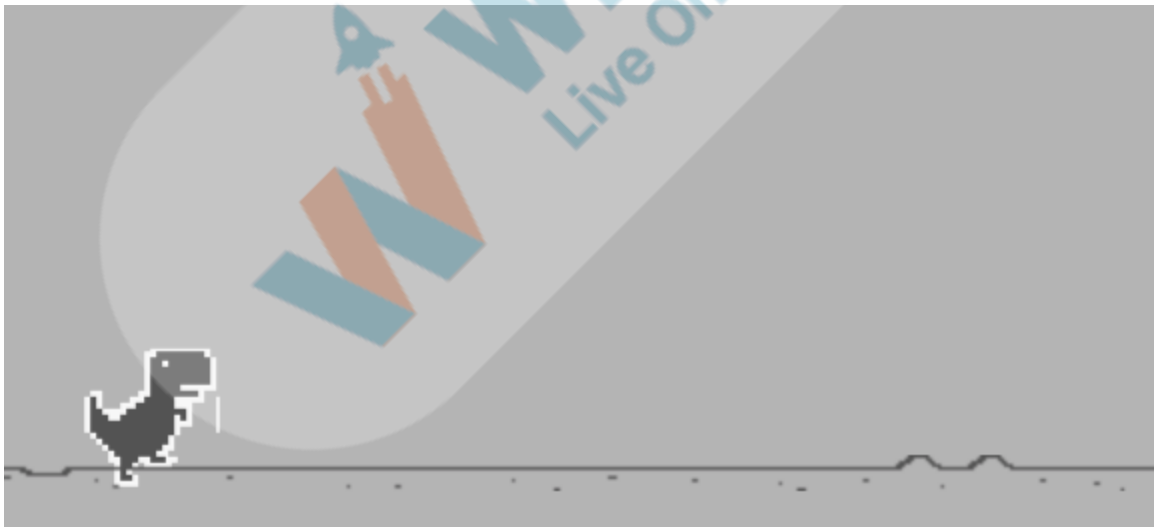
<p>Let's leave it outside the conditional statements then:</p>	
 <pre>function draw() { background(180); //displaying score text("Score: "+ score, 500,50); score = score + Math.round(frameCount/60); if(gameState === PLAY){ } else if (gameState === END) { } }</pre>	
<p>When do we want to move the ground, in PLAY state or in END state?</p> <p>Let's move the line inside the gameState === PLAY condition.</p>	<p>ESR: In PLAY state.</p>
<p>What do we want the ground speed to be when gameState === END?</p> <p>Let's add this to the else if condition.</p>	<p>ESR: 0</p>

```
//displaying score  
text("Score: "+ score, 500,50);  
score = score + Math.round(frameCount/60);
```

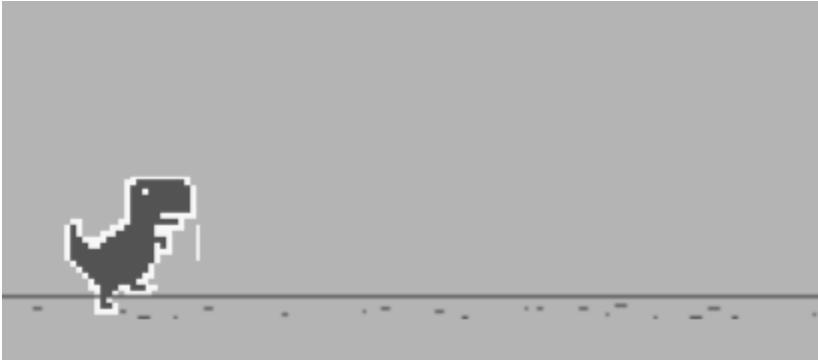

```
if(gameState === PLAY){  
  //move the ground  
  ground.velocityX = -4;  
}  
else if (gameState === END) {  
  //move the ground  
  ground.velocityX = 0;  
}
```

Output:

Moving ground when the game state is PLAY:



Static ground when game state is END:

	
<p>Do you think you can do the remaining on your own?</p> <p>Let's try. I will guide you through it.</p>	<p>ESR: Varied.</p>
<p>Teacher Stops Screen Share</p>	
<p>Now it's your turn. Please share your screen with me.</p>	
<p>STUDENT-LED ACTIVITY 2 - 10 mins</p>	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 	
<p><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • The student assigns different game behavior to the different objects in the game according to the state. • The student uses the group functions to manipulate the behavior of all the objects in the group. • The student writes the condition to change the game state from play to end. 	
<div>  <p>Teacher can show slideshow from slides 24 & 25</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> </div>	
<p>Teacher Action</p>	<p>Student Action</p>

<p>Step 4: Student-Led Activity</p> <p><i>Teacher guides the student to download the code from the GitHub link.</i></p> <p>Let's think about displaying the score.</p> <p>Do we want to display the score at all times?</p> <p>Let's keep this outside the 'if and else if' conditions.</p> <p>Do we want to update the score at all times?</p> <p>Let's put this into the PLAY conditional statement.</p>	<p><i>Student downloads the <u>Student Activity 2</u> code from GitHub and opens in the VS Code Editor.</i></p> <p>ESR: Yes!</p> <p>ESR: No! Only during the PLAY state.</p> <p><i>The student makes adjustments in the code - indenting the code wherever needed.</i></p>
---	--


```
function draw() {
  background(180);

  //displaying score
  text("Score: "+ score, 500,50);

  if(gameState === PLAY){
    //move the ground
    ground.velocityX = -4;
    //scoring
    score = score + Math.round(frameCount/60);

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

In which state do we want the ground to reset its position?

Our ground will be moving continuously, but at the Play state, we want to keep the ground at the center of the canvas.

Let's move that block of code into the **PLAY** state conditional statement.

What about the jump and gravity? Do we want the Trex to jump when the game state is **END**.

Let's move that block of code into the **PLAY** state condition then.

ESR:

Only during PLAY state.

ESR:

No! We want it to jump only during play state.

The student makes adjustments in the code - indenting the code wherever needed.

```
if(gameState === PLAY){  
  //move the ground  
  ground.velocityX = -4;  
  //scoring  
  score = score + Math.round(frameCount/60);  
  
  if (ground.x < 0){  
    ground.x = ground.width/2;  
  }  
  
  //jump when the space key is pressed  
  if(keyDown("space")&& trex.y >= 100) {  
    trex.velocityY = -13;  
  }  
}
```

```
//move the ground
ground.velocityX = -4;
//scoring
score = score + Math.round(frameCount/60);

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

//jump when the space key is pressed
if(keyDown("space")&& trex.y >= 100) {
  trex.velocityY = -13;
}

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

else if (gameState === END) {
  ground.velocityX = 0;
```

Do we want the invisible ground to support the trex at all times?

Let us keep it outside both the conditions then.

ESR:
Yes!

```

else if (gameState === END) {
    ground.velocityX = 0;

    obstaclesGroup.setVelocityXEach(0);
    cloudsGroup.setVelocityXEach(0);
}

//stop trex from falling down
trex.collide(invisibleGround);

drawSprites();
}

```

When do we want to spawn the cloud and the obstacles?

Let's move these into **PLAY** state.

ESR:

In PLAY state.

The student makes adjustments in the code - indenting the code wherever needed.

<Move **spawnClouds()** and **spawnObstacles()** inside the PLAY condition.>

```
//jump when the space key is pressed
if(keyDown("space")&& trex.y >= 100) {
    |   trex.velocityY = -13;
}

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

//spawn the clouds
spawnClouds();

//spawn obstacles on the ground
spawnObstacles();

if(obstaclesGroup.isTouching(trex)){
    |   gameState = END;
}
}

else if (gameState === END) {
    ground.velocityX = 0;
```

Good! We have done well so far!

Which state our game is in when it starts?

When do we want it to change into **END** state?

ESR:

PLAY state.

ESR:

When the trex collides with the obstacles/cactus. We use conditional programming (if block).

<p>What do we do when we instruct the computer to do something IF something happens?</p> <p>Let us write the code for that. We know all our obstacles are stored in obstaclesGroup.</p> <p>Do we see any “isTouching” function in obstaclesGroup?</p> <p><i>The teacher guides the student to write the code.</i></p>	<p>ESR: Varied</p> <p>ESR: Yes.</p> <p><i>The student writes an if condition inside another if condition to change the gameState to end when the trex collides with an obstacle.</i></p>
--	--


```
//spawn obstacles on the ground
spawnObstacles();

if(obstaclesGroup.isTouching(trex)){
    gameState = END;
}

else if (gameState === END) {
    ground.velocityX = 0;

    obstaclesGroup.setVelocityXEach(0);
    cloudsGroup.setVelocityXEach(0);
}

//stop trex from falling down
trex.collide(invisibleGround);

drawSprites();
}
```

Let us run the code now and see closely what happens.

What happened?

The student runs the code and checks the output.

ESR:

When the trex collides with the obstacle, the ground stops scrolling, the obstacles and the clouds stop being spawned. But they move at the same velocity; jump etc. is not working in the END state.

Good observation! We need to give 0 velocity to all the obstacles and the clouds in the game when the trex collides. We can use functions/instructions inside the group toolbox to do so.

The teacher guides the student to give 0 velocity to all the obstacles and the clouds using the `setVelocityXEach()`.

The student writes code to give 0 velocity to the obstacles and clouds in the END state.




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

if(obstaclesGroup.isTouching(trex)){
    gameState = END;
}
}

else if (gameState === END) {
    ground.velocityX = 0;

    obstaclesGroup.setVelocityXEach(0);
    cloudsGroup.setVelocityXEach(0);
}

//stop trex from falling down
trex.collide(invisibleGround);

//spawn the clouds
spawnClouds();

//spawn obstacles on the ground
spawnObstacles();
}
```


Let us run the code and see what happens.

What do you see?




The student runs the code to check the output.



ESR:

The trex stops just before it hits the obstacle. Then after

	sometime the obstacle and the clouds all vanish.
<p>So, we have bugs in the game. Let us solve these in the next class.</p> <p>Also, since we are looking and fixing bugs, it would be great if you can invite one of your friends for the next class.</p> <p>Sometimes, when we ourselves are working on a game, we can miss some obvious bugs in the game.</p> <p>It will be great to showcase the game to a third person and look at what are the bugs that he/she can find.</p> <p>Looking forward to seeing your friend in the next class then!</p>	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 mins	
<p><u>FEEDBACK</u></p> <ul style="list-style-type: none"> • 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. 	
<p>Teacher can show slideshow  from slides 26 to 36</p> <p>Refer to speaker notes and follow the instructions on each slide.</p>	
Teacher Action	Student Action
<p>Step 4:</p> <p>Wrap-Up</p> <p>(5 min)</p> <p>Let us quickly review what we did in today's class.</p>	<p>ESR:</p> <ul style="list-style-type: none"> • We learned how to group similar kinds of sprite objects into a single group and

What did we do today?	<p>change all their behavior at once.</p> <ul style="list-style-type: none"> • We also learned how to use gameState to separate the behavior of objects in the game for different states
And we have some bugs in the program which we are going to solve for the next class.	
<p>We are so close to making the final finished game of Trex.</p> <p>And you have done amazingly well so far.</p> <p>Great job!</p>	
Quiz time - Click on in-class quiz	
Question	Answer
<p>Which option contains the correct instruction to create a group of sprites?</p> <p>A. spritesGroup = newGroup(); B. spritesGroup = new Group(); C. spritesGroup.newGroup(); D. newGroup.add(spritesGroup);</p>	B
<p>Choose the correct piece of code to add a sprite to a group.</p> <p>A. sprite.add(spritesGroup); B. spritesGroup=sprite.add(); C. sprite=spritesGroup.add(); D. spritesGroup.add(sprite);</p>	D

<p>Select the correct option to give 0 velocity to a group of sprites.</p> <p>A. spritesGroup.setVelocityXEach(0); B. spritesGroup.setVelocityXEach=0; C. spritesGroup.velocityX=0; D. spritesGroup.velocityX(0);</p>	<p>A</p>
<p>End the quiz panel</p>	
<p style="text-align: center;"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> • Appreciate the student for their efforts in the class. • Ask the student to make notes for the reflection journal along with the code they wrote in today's class. 	
<p>You get hats-off for your excellent work!</p> <p>Meanwhile you can also think of solving some of these bugs which we will cover in the next class.</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div style="display: flex; flex-direction: column; align-items: center;"> <div style="background-color: #00728f; color: white; padding: 5px; margin-bottom: 5px; display: flex; align-items: center;"> Creatively Solved Activities  +10 </div> <div style="background-color: #00728f; color: white; padding: 5px; margin-bottom: 5px; display: flex; align-items: center;"> Great Question  +10 </div> <div style="background-color: #00728f; color: white; padding: 5px; display: flex; align-items: center;"> Strong Concentration  +10 </div> </div>
<p>* This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.</p> <p>Project Overview</p> <p>BALLOON BUSTER - 2</p>	<p>Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.</p>

<p>Goal of the Project:</p> <p>Today, you have learned about game states PLAY and END. You also learned how to create groups, add objects in groups, and use properties and functions for groups.</p> <p>In this project, you will have to practice and apply what you have learned in the class and Create an arrow function. And also release the arrow from the bow after pressing the space key. Give some lifetime to each sprite and write a condition to destroy the arrow and red balloon when the arrow touches any redBalloon sprite.</p> <p>** This is a continuation of Project 13, so make sure to complete that before doing this project. **</p> <p>Story:</p> <p>You have already helped Meera in creating a complete design of the game with balloons, bow and arrow. Now she wants to improve the game by adding a scoring system by incrementing scores with different numbers. And at last add a feature to the game where when an arrow hits a balloon, it gets destroyed.</p> <p>I am very excited to see your project solution and I know you will do really well.</p> <p>Bye Bye!</p>	<p><i>Students engage with the teacher over the project.</i></p>
<div>  Teacher ends slideshow </div>	
<div> Teacher Clicks  </div>	
<p>Additional Activities</p>	<p><i>The student uses the markdown editor to write</i></p>

<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> ○ 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? 	<p><i>her/his reflections in the reflection journal.</i></p>
--	--

Activity	Activity Name	Links
Teacher Activity 1	Template Code	https://github.com/pro-whitehatjr/p5js_template
Teacher Activity 1 Solution	Solution Code	https://github.com/pro-whitehatjr/C14_TA-1
Teacher Activity 2	Trex Stage 4	https://github.com/pro-whitehatjr/Pro_c14_trex4
Teacher Activity 3	Trex Stage 5 (For Teacher Reference)	https://github.com/pro-whitehatjr/PRO-C14-stage5
Student Activity 1	Template Code	https://github.com/pro-whitehatjr/p5js_template
Student Activity 1 Solution	Solution Code	https://github.com/pro-whitehatjr/C1-14-SA-1
Student Activity 2	Trex Stage 4.5	https://github.com/pro-whitehatjr/Pro_c14_trex4.5
Teacher Reference visual aid link	Visual aid link	https://s3-whjr-curriculum-uploads.whjr.online/1c74d670-dbd9-4273-b398-437

		52ef53754.html
Teacher Reference	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/8427656e-504f-4d32-998c-a60e846e640b.pdf
Project Solution	Balloon Buster-2	https://github.com/pro-whitehatjr/Project_C14_Balloon_Buster_2

