WhiteHat Jr
Live online Coding for Kids

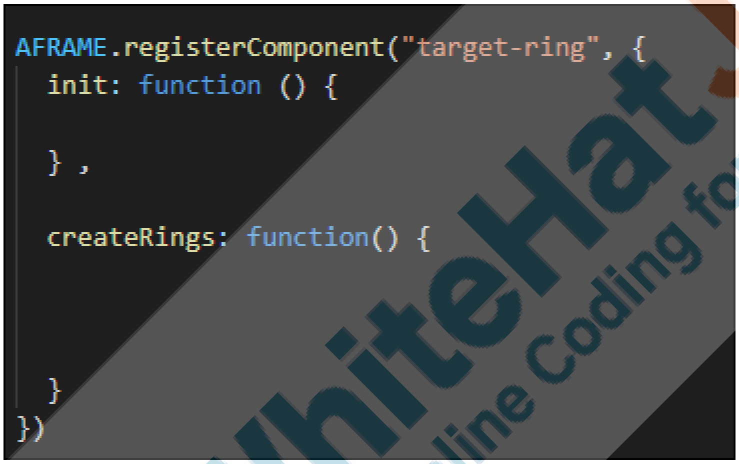
Topic	**A-Frame DYNAMIC ENTITY**	
Class Description	**Students learn to create entities dynamically in the A-Frame scene when the program is running. Students will learn how to add shapes and 3D models and set its attributes through A-Frame components. Students will learn how to add animated models in the A-Frame scene.**	
Class	**C154**	
Class time	**50 mins**	
Goal	● Learn how entities are created dynamically in the A-Frame scene. ● Learn to add basic shapes and models to the scene through components. ● Learn to add animated models in the scene. ● Learn how to set single and multiple value attributes in the component.	
Resources Required	● Teacher Resources ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen ● Student Resources ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen	
Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**	**05 mins** **20 mins** **20 mins** **05 mins**

WARM-UP SESSION - 05 mins	
Teacher starts slideshow from slides 1 to 10 Refer to speaker notes and follow the instructions on each slide.	
Activity details	**Solution/Guidelines**
Hi, 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:** ● Reconnect with previous class topics. ● Warm-Up quiz session.	**ESR**: Varied Response. Click on the slide show tab and present the slides.
Q&A Session	
Question	**Answer**
How did we move the flight forward? A. by giving a forward velocity to the flight B. by updating the forward position value of the flight C. by moving the terrain backward D. by giving a backward velocity to the flight	**C**
What should be updated to create a turning effect? A. position attribute B. rotation attribute C. revolution Attribute D. turn attribute	**B**
Continue the WARM-UP session	

<table>
<tr><th colspan="2">Activity details</th><th>Solution/Guidelines</th></tr>
<tr><td colspan="2"><b><i>Run the presentation from slide 4 to slide 10 to set the problem statement.</i></b>
<b>The following are the warm-up session deliverables:</b>
● <b>Review code from the last class.</b>
● <b>Teacher ends slideshow</b></td><td></td></tr>
<tr><td colspan="3"><b>Teacher ends slideshow</b></td></tr>
<tr><td colspan="3"><b>Teacher Initiates Screen Share</b></td></tr>
<tr><td colspan="3"><b>TEACHER-LED ACTIVITY - 20 mins</b></td></tr>
<tr><td colspan="3"><b><u>CHALLENGE</u></b>
● <b>Create a ring entity and display it inside the scene using A-Frame components.</b></td></tr>
<tr><td><b>Step 2:
Teacher-led Activity
(15 min)</b></td><td><i>&lt;The teacher opens the code from the previous class.&gt;</i>
So let's start by creating the ring target elements.
What can be done about this? Any idea?
Yes. That is right.
But today, we will be learning how to add entities through components.</td><td><b>ESR</b>: We can add the <a-entity> in the scene and add a torus shape.
<i>The student observes.</i></td></tr>
</table>

	For this, let's create a file and include that in index.html. *<The teacher creates a Target.js file in the Component folder and include that file in index.html.>* Now, to create a ring target element through the component, we first need to register it in A-Frame. Can you tell me how we can do that? Great! *<The teacher codes to register the component in A-Frame.>* As we are creating the rings, let's name this component as "**target-ring**". *<The teacher names the component as "target-ring".>* We will be creating multiple targets with the help of this component. We'll be writing our own function to create the rings and add its attributes. To write our own function in the A-Frame component, we need to follow the same syntax followed by the components' predefined methods, init(), tick() etc.	**ESR**: We can use **AFRAME.registerComponent()**.

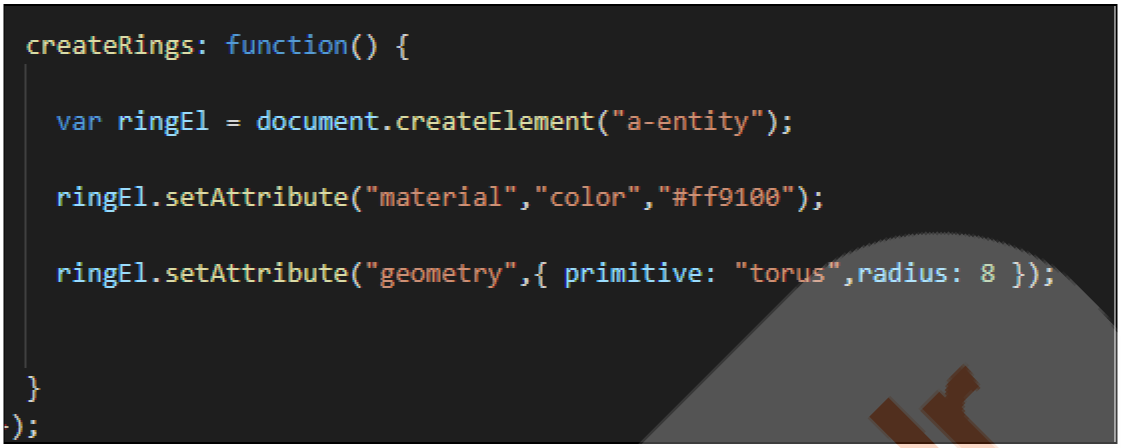
	You can pick your own meaningful name for the function.
	functionName: function(){
	}
	<The teacher codes to write the createRings function.>	



```
AFRAME.registerComponent("target-ring", {
  init: function () {

  },

  createRings: function() {



  }
})
```

	Now, to create a new element, JavaScript DOM provides us a predefined function called **createElement()** which will help us in creating the new entity. *<The teacher uses document.createElement() to create a new entity and store it in the ringEl variable.>* Now, using the **setAttribute()** method, we'll set attributes of the rings like material color and its geometry to the ring.	ESR: The thing will have a shape, position, color etc.

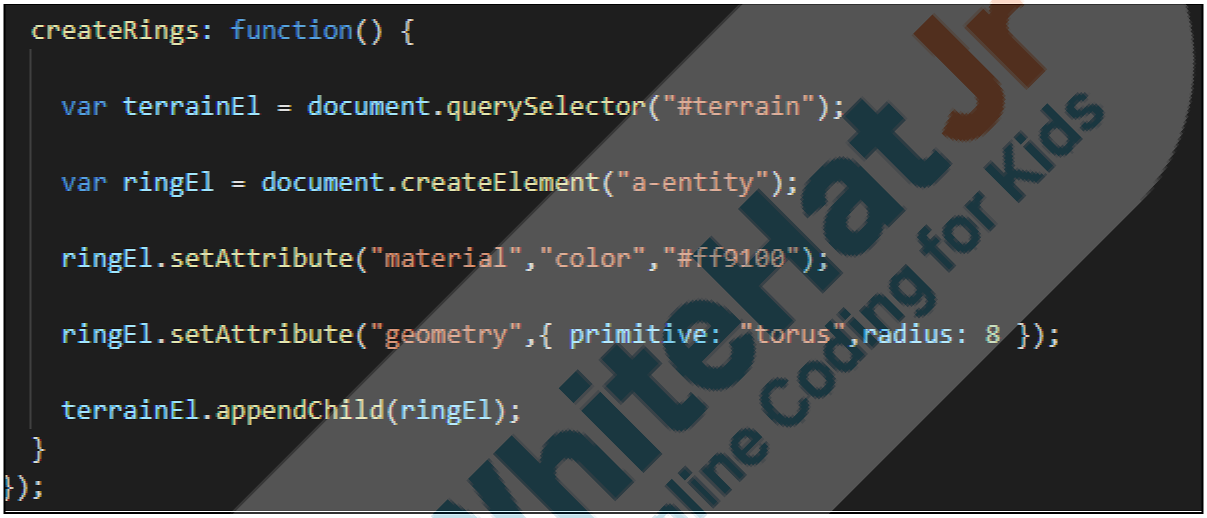
	But before that, do you remember the single and multiple property value components? Amazing! Both the material and geometry are multi-value property components. If we just want to set the single value, we can directly take the component name, property name and its value. For example, we just want to set the color property with the color value for the material component. But for the geometry component, we want to set the primitive shape as a torus and the radius of the shape, which will be written as a JSON object. *<The teacher codes to set the material color and it's geometry to the ring using the setAttribute() method.>*	**ESR**: Yes. A component with one property value is a single property component and with multiple property values has many values as properties of the component.



```
createRings: function() {

  var ringEl = document.createElement("a-entity");

  ringEl.setAttribute("material","color","#ff9100");

  ringEl.setAttribute("geometry",{ primitive: "torus",radius: 8 });


 }
);
```

	Now, we have written the code to create the ring but we haven't added it to the scene. We can attach this entity to any of the elements in the scene. Let’s make the rings a child of the terrain element. Inside the function, we'll first select the terrain using the JavaScript DOM function **document.querySelector()** using the id given to the terrain element. **querySelector()** function helps to select the particular entity element once the id is given to the entities. Now, to make the ring entities child of the terrain entity, add the ring entities as the child of the terrain using **terrainEl.appendChild(ringEl)**.	*The student observes and learns.*

	appendChild() is also the JavaScript DOM function that can be used to make one element as a child of another element. *<The teacher codes to add the **ringEl** to the terrain using the **appendChild()** method.>*	



```
 createRings: function() {

    var terrainEl = document.querySelector("#terrain");

    var ringEl = document.createElement("a-entity");

    ringEl.setAttribute("material","color","#ff9100");

    ringEl.setAttribute("geometry",{ primitive: "torus",radius: 8 });

    terrainEl.appendChild(ringEl);
  }
});
```

	Perfect! So, we have written the function to create rings. Now, we need to use this function to create multiple rings.	
	Can you guess how we can do that?	**ESR**: We can have a loop that will run multiple times and then we can call this **createRing** function inside it.
	Awesome!	

	We need to use this function inside a predefined method of the A-Frame component. Let's use the .init() method which is loaded as soon as the component is attached to the entity. We will use the for loop which will run 20 times to create 20 ring target entities. Each ring that we create needs to have a unique id; to assign this unique id, we can use the counter value of the loop.	
	Do you remember with the help of template literal, **${}** we can manipulate the string to generate a **unique id**?	**ESR:** Yes!
	Great! We also need to have different positions for each ring.	
	Can you tell me how we can assign random positions to the rings?	**ESR:** We can use the random function to generate a random number.
	Yes, using the random number, we will generate random positions for the rings.	

In JavaScript, the **Math.random()** function **gives the values between 0 and 1.**

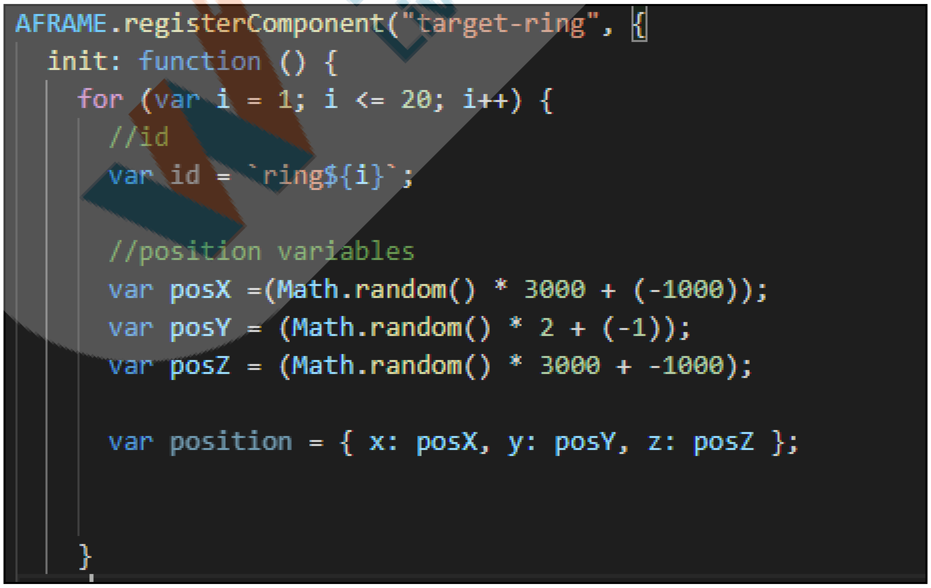
We can multiply the random number with a number to get the value between any other numbers.

For example, Math.random()*100 will give random values between 0 and 100.

We can also add and subtract to get the different range of numbers.

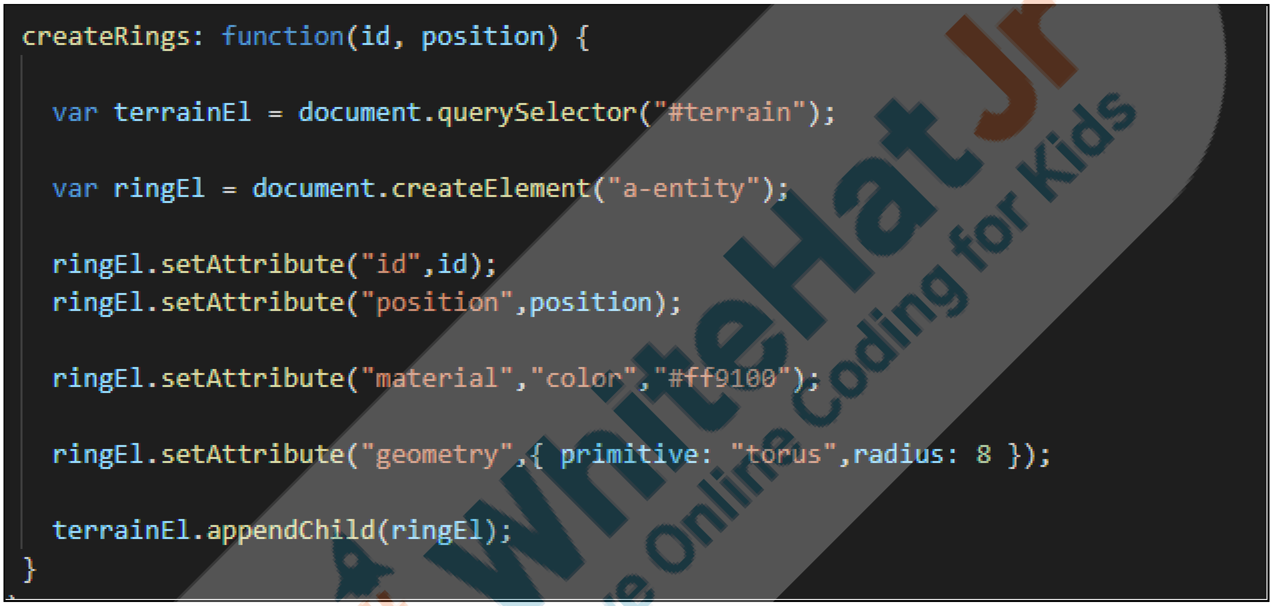
For example, Math.random()*100 + (-50) will give random values between -50 and 50.

Let’s create variables to get values of positions in x, y and z direction randomly.

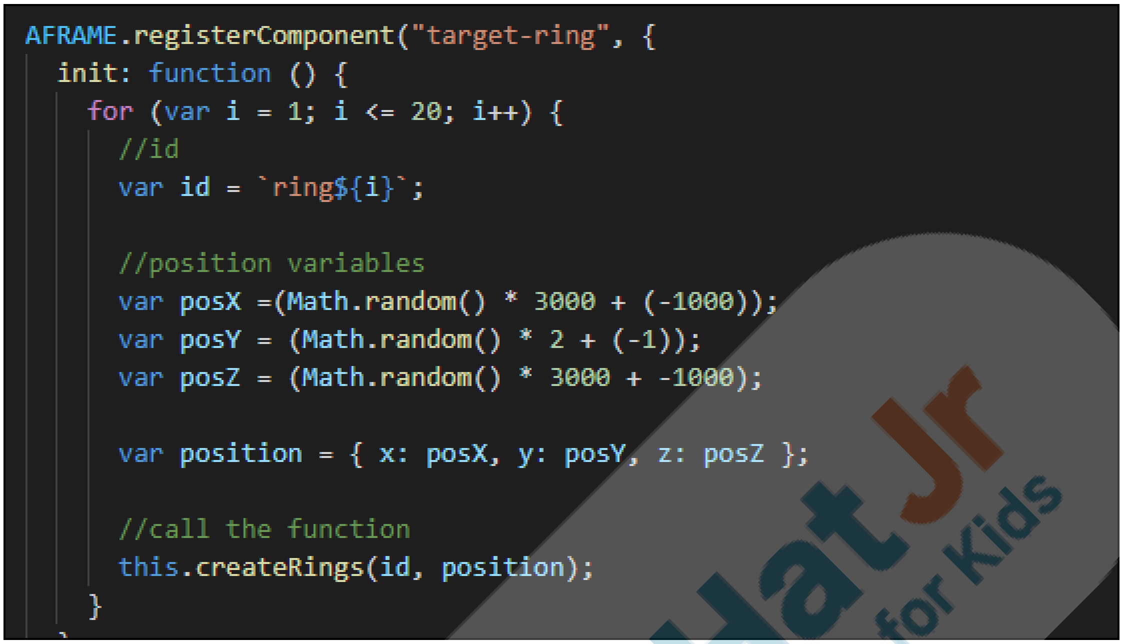


```
AFRAME.registerComponent("target-ring", {
  init: function () {
    for (var i = 1; i <= 20; i++) {
      //id
      var id = `ring${i}`;

      //position variables
      var posX =(Math.random() * 3000 + (-1000));
      var posY = (Math.random() * 2 + (-1));
      var posZ = (Math.random() * 3000 + -1000);

      var position = { x: posX, y: posY, z: posZ };


    }
```

	Now let's pass these values as arguments of the functions and update the values in the function with the help of **setAttribute()** function. Then the function can be called inside the loop with the help of the "this" keyword.	



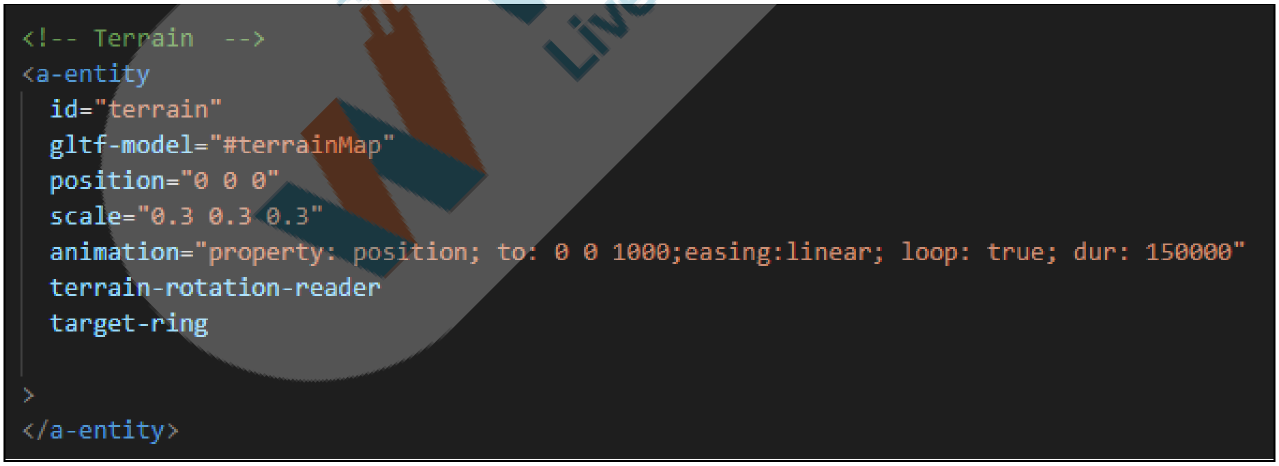
```
createRings: function(id, position) {

  var terrainEl = document.querySelector("#terrain");

  var ringEl = document.createElement("a-entity");

  ringEl.setAttribute("id",id);
  ringEl.setAttribute("position",position);

  ringEl.setAttribute("material","color","#ff9100");

  ringEl.setAttribute("geometry",{ primitive: "torus",radius: 8 });

  terrainEl.appendChild(ringEl);
}
```



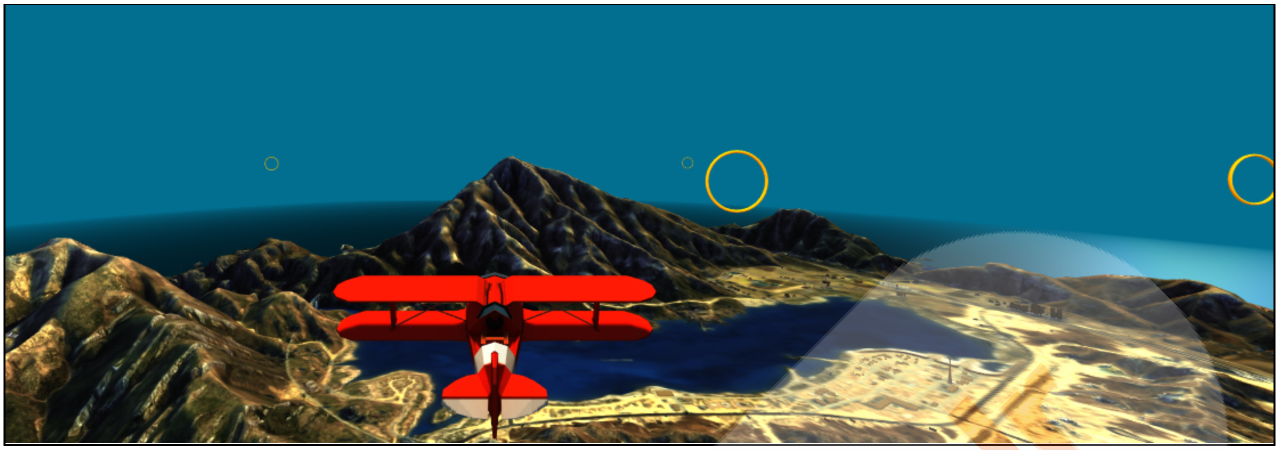
```
AFRAME.registerComponent("target-ring", {
  init: function () {
    for (var i = 1; i <= 20; i++) {
      //id
      var id = `ring${i}`;

      //position variables
      var posX =(Math.random() * 3000 + (-1000));
      var posY = (Math.random() * 2 + (-1));
      var posZ = (Math.random() * 3000 + -1000);

      var position = { x: posX, y: posY, z: posZ };

      //call the function
      this.createRings(id, position);
    }
```

Now, let's attach the component to the terrain, run and check the output.

<The teacher runs the code and checks for the output.>



```
<!-- Terrain  -->
<a-entity
  id="terrain"
  gltf-model="#terrainMap"
  position="0 0 0"
  scale="0.3 0.3 0.3"
  animation="property: position; to: 0 0 1000;easing:linear; loop: true; dur: 150000"
  terrain-rotation-reader
  target-ring

>
</a-entity>
```



	Now, can you try creating some bird entities using a component in A-Frame?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
STUDENT-LED ACTIVITY - 20 mins		

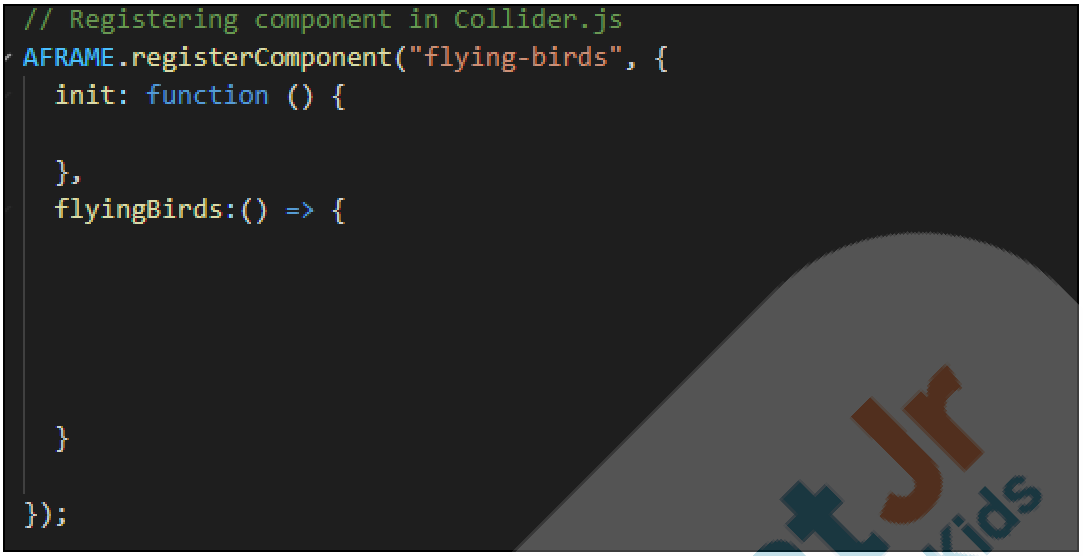
- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start screen share.**
- **Teacher gets into fullscreen.**

ACTIVITY

- **Write a component to create multiple dynamic gLTF model entities.**
- **Set the attributes of the gLTF models.**
- **Attach the model to the scene.**

Teacher starts slideshow from slide 11 to slide 14

Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to open the code from the previous class.* ***[Student Activity 1]***	*The student opens the code from the previous class.*
	So first we'll start by creating a file called Collider.js file in the components folder. *<The teacher guides the student to create the Collider.js file in the component folder and include that file in index.html.>*	*The student creates the Collider.js file in the component folder and adds the file in index.html.*
	Now, what do we have to do? *<The teacher guides the student to register the component using the **AFRAME.registerComponent()**.>* *<The teacher guides the student to name the component as flying-birds.>*	**ESR:** We need to register the component using the AFRAME.registerComponent(). *The student codes to register components.* *The student names the component as flying-birds.*



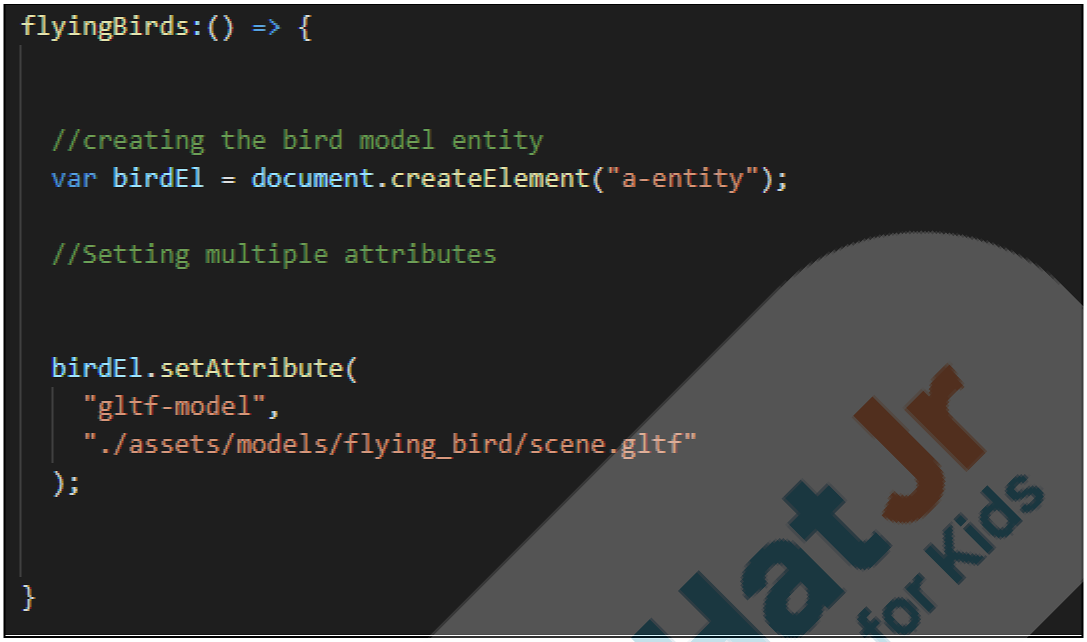
```
// Registering component in Collider.js
AFRAME.registerComponent("flying-birds", {
  init: function () {

  },
  flyingBirds:() => {





  }

});
```

Next, we'll code to create the bird element. Here, we'll be using the animated gLTF model of the bird.

The animation in the model is added while the model was created by 3D models for artists.

<The teacher guides the student to download the gLTF model of the bird>

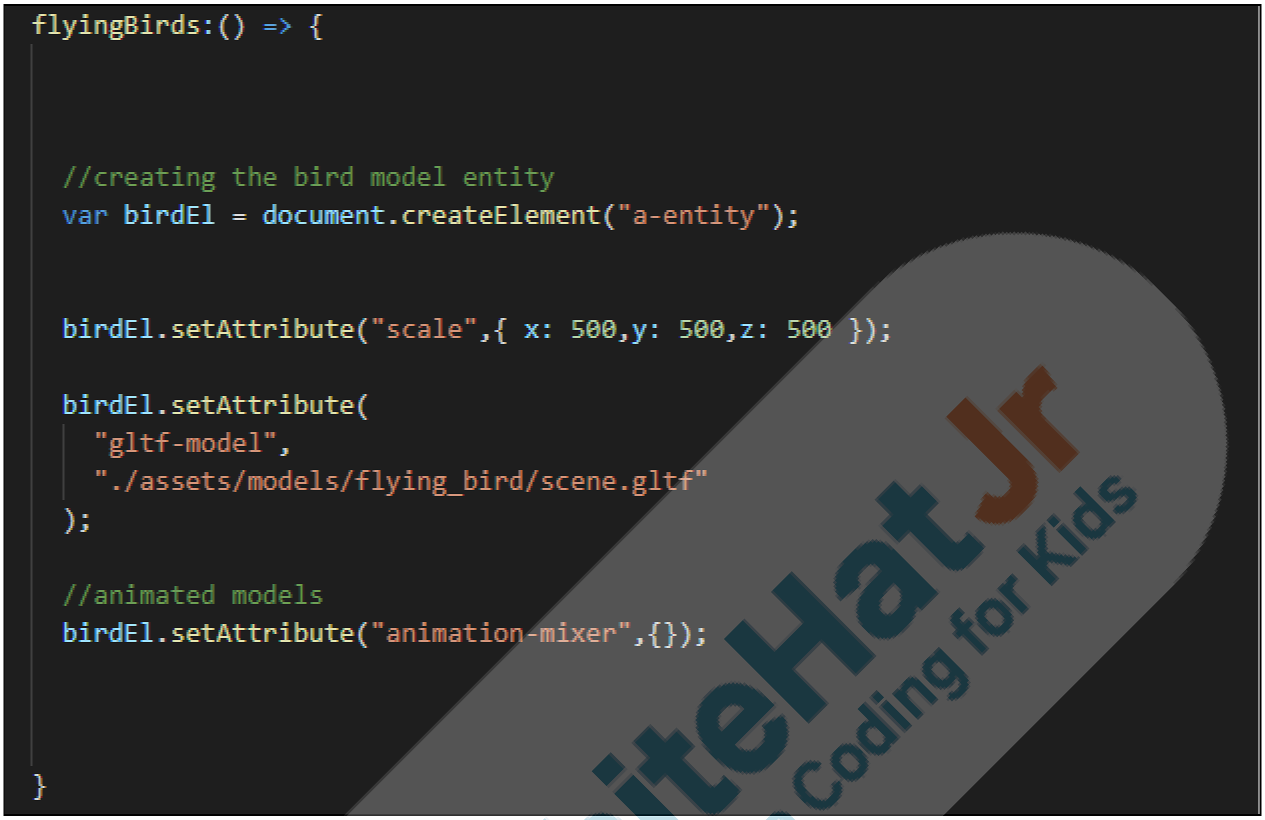
<The teacher guides the student to create bird entity element and the set the "gltf-model" attribute.>



```
flyingBirds:() => {


  //creating the bird model entity
  var birdEl = document.createElement("a-entity");

  //Setting multiple attributes


  birdEl.setAttribute(
    "gltf-model",
    "./assets/models/flying_bird/scene.gltf"
  );


}
```

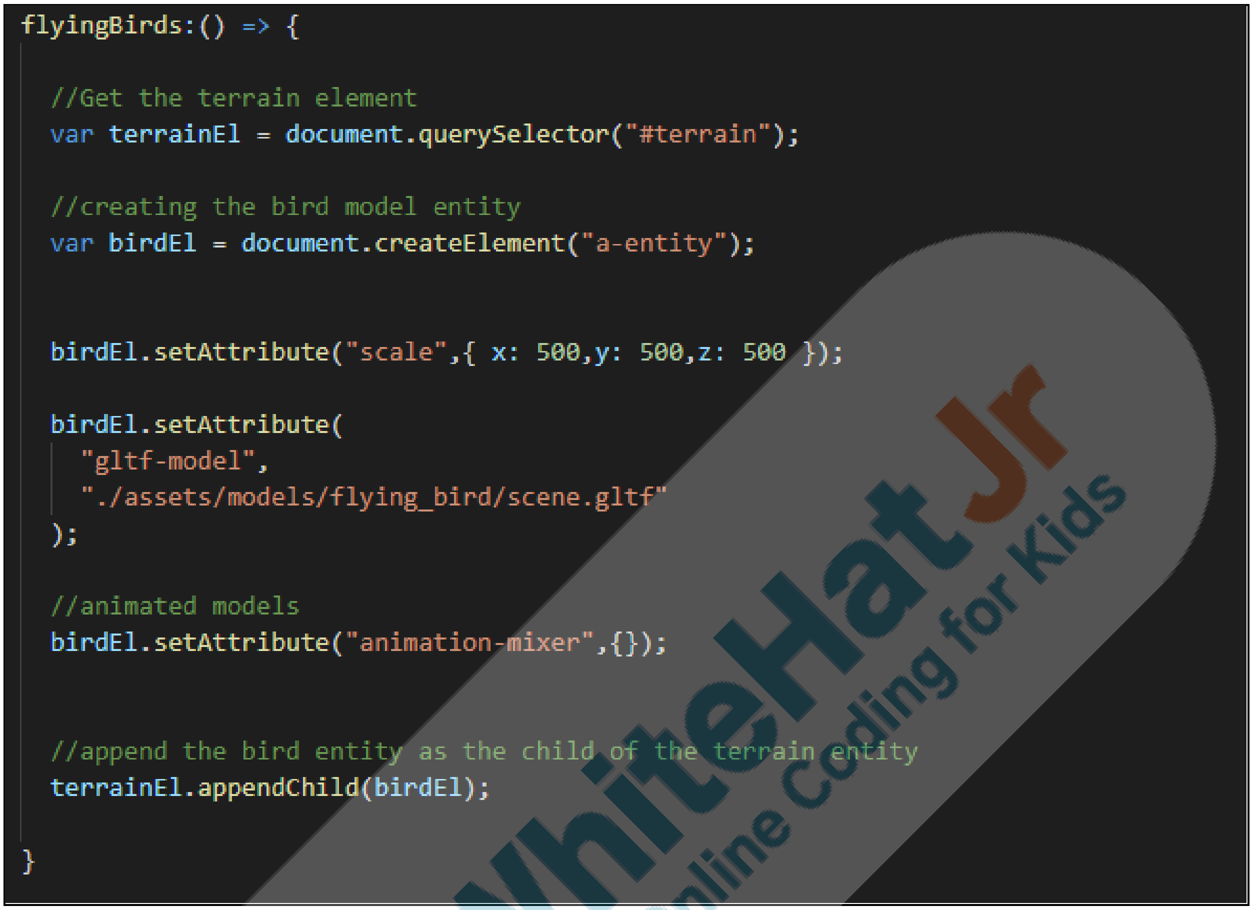
	But the birds that we would see in the simulation would be static. And we want the birds to be animated. Can you tell me how we can make the birds animated? In the setAttribute() method we'll use the **animation-mixer component**. With the help of this component we can activate the animation of the model. This component is a part of another library of A-Frame - “**aframe-extras**”. Link: https://cdn.jsdelivr.net/gh/donmccurdy	**ESR**: Varied.

	[/aframe-extras@v6.1.1/dist/aframe-extras.min.js]() For this, we will need to include this library of the A-Frame in the index.html file. <script src="https://cdn.jsdelivr.net/gh/donmccurdy/aframe-extras@v6.1.1/dist/aframe-extras.min.js"></script> The aframe-extras library provides many other components that can be used in the A-Frame scene. The animation-mixer component of this library has many properties which we will explore later. For this, we can just use the empty JSON object to keep the default values of any component.	



```
flyingBirds:() => {



  //creating the bird model entity
  var birdEl = document.createElement("a-entity");


  birdEl.setAttribute("scale",{ x: 500,y: 500,z: 500 });

  birdEl.setAttribute(
    "gltf-model",
    "./assets/models/flying_bird/scene.gltf"
  );

  //animated models
  birdEl.setAttribute("animation-mixer",{});



}
```

	Now, we have to append the bird entity as the child of the terrain entity.	*Student codes to append bird entities as the child of the terrain entity.*

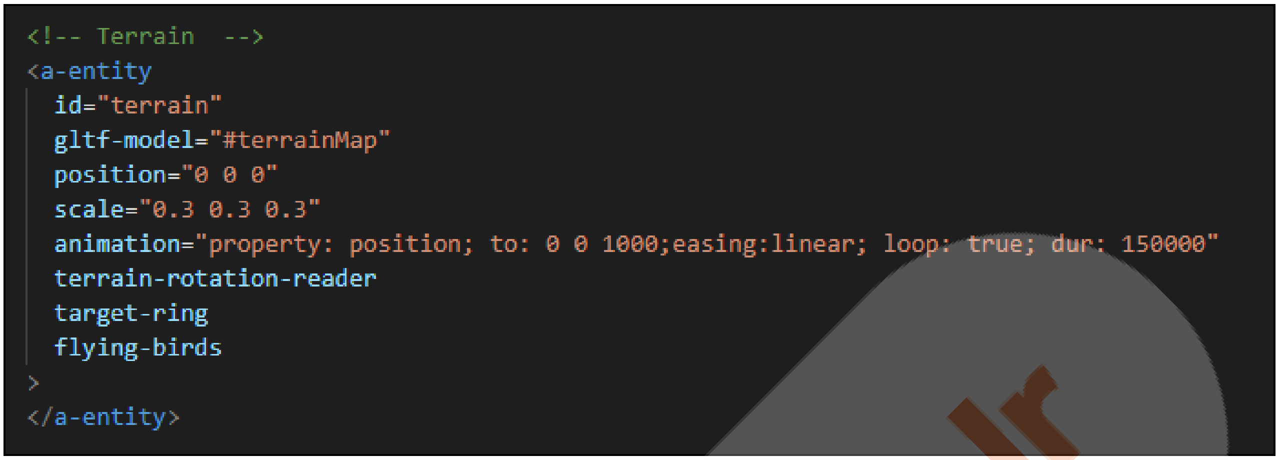


```
flyingBirds:() => {

  //Get the terrain element
  var terrainEl = document.querySelector("#terrain");

  //creating the bird model entity
  var birdEl = document.createElement("a-entity");


  birdEl.setAttribute("scale",{ x: 500,y: 500,z: 500 });

  birdEl.setAttribute(
    "gltf-model",
    "./assets/models/flying_bird/scene.gltf"
  );

  //animated models
  birdEl.setAttribute("animation-mixer",{});


  //append the bird entity as the child of the terrain entity
  terrainEl.appendChild(birdEl);

}
```

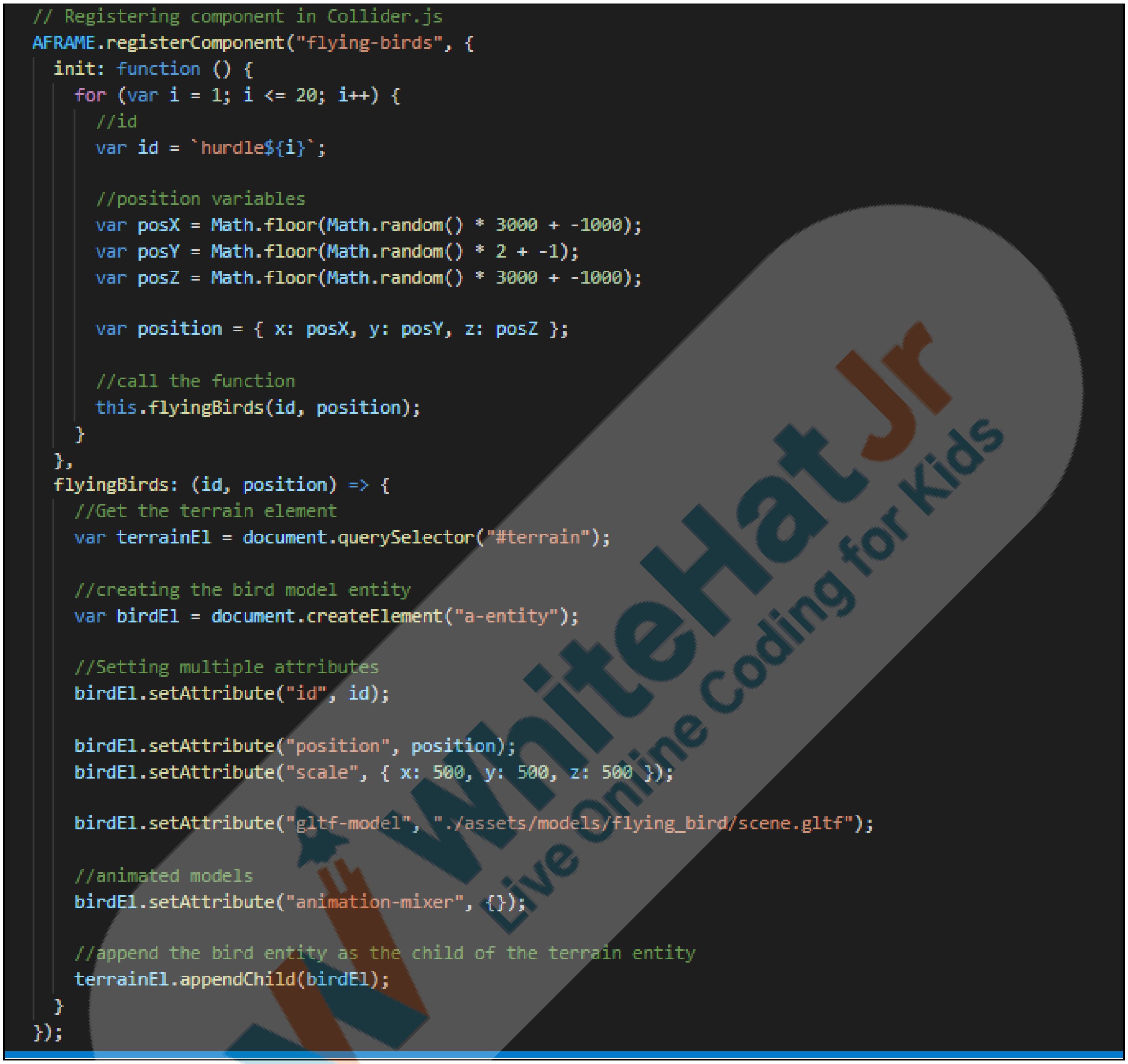
Now, to create the multiple birds we'll write a for loop and call the createBird function inside it.

Guide the student to :-

1. *Use the counter value as the id number.*
2. *Use a random function to generate random positions and store the x,y,z values in position objects.*
3. *Call the flyingBirds() and pass the id and position value to it.*
4. *Attach the component to the terrain entity.*

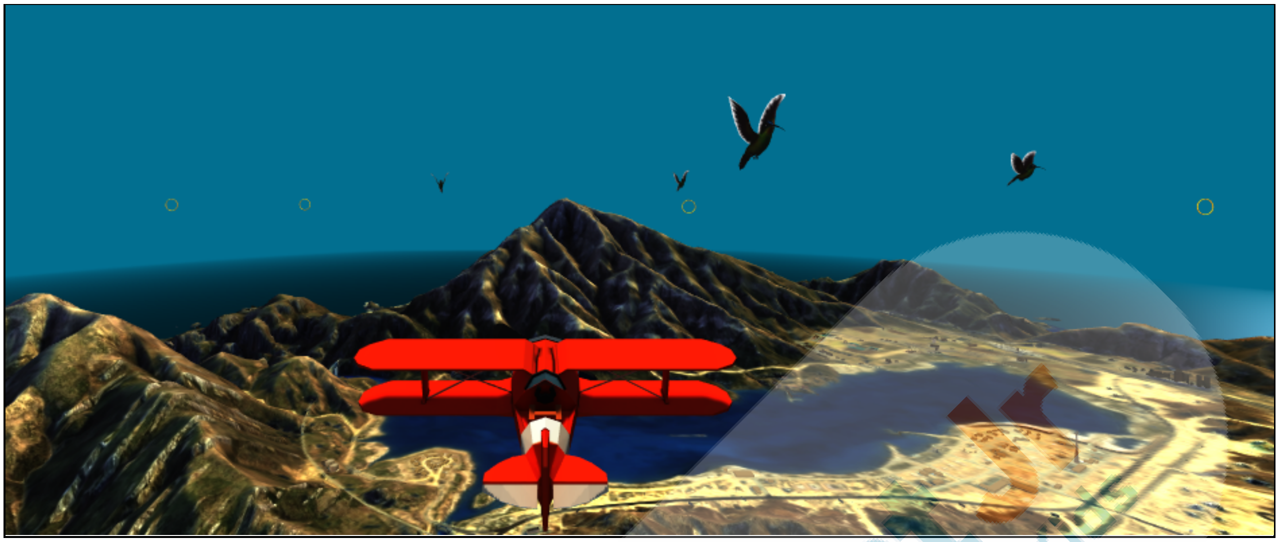


```
<!-- Terrain  -->
<a-entity
  id="terrain"
  gltf-model="#terrainMap"
  position="0 0 0"
  scale="0.3 0.3 0.3"
  animation="property: position; to: 0 0 1000;easing:linear; loop: true; dur: 150000"
  terrain-rotation-reader
  target-ring
  flying-birds
>
</a-entity>
```



```
// Registering component in Collider.js
AFRAME.registerComponent("flying-birds", {
  init: function () {
    for (var i = 1; i <= 20; i++) {
      //id
      var id = `hurdle${i}`;

      //position variables
      var posX = Math.floor(Math.random() * 3000 + -1000);
      var posY = Math.floor(Math.random() * 2 + -1);
      var posZ = Math.floor(Math.random() * 3000 + -1000);

      var position = { x: posX, y: posY, z: posZ };

      //call the function
      this.flyingBirds(id, position);
    }
  },
  flyingBirds: (id, position) => {
    //Get the terrain element
    var terrainEl = document.querySelector("#terrain");

    //creating the bird model entity
    var birdEl = document.createElement("a-entity");

    //Setting multiple attributes
    birdEl.setAttribute("id", id);

    birdEl.setAttribute("position", position);
    birdEl.setAttribute("scale", { x: 500, y: 500, z: 500 });

    birdEl.setAttribute("gltf-model", "./assets/models/flying_bird/scene.gltf");

    //animated models
    birdEl.setAttribute("animation-mixer", {});

    //append the bird entity as the child of the terrain entity
    terrainEl.appendChild(birdEl);
  }
});
```

	Now, let's run and test the code. We have got the animated birds flying in the scene. Good Job!	*Student runs the code to check the output.*



Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 Mins

Teacher starts slideshow from slide 15 to slide 24

Activity details	Solution/Guidelines
Run the presentation from slide 15 to slide 24 **Following are the wrap-up session deliverables:** ● Explain the facts and trivias ● Next class challenge ● Project for the day ● Additional Activity	Guide the student to develop the project and share with us.

Quiz time - Click on In-Class Quiz

Question	Answer
Which of the following components can be used to animate the birds? A. music component B. attribute component	C

<table>
<tr><td colspan="2">C. animation-mixer component
D. register component</td><td></td></tr>
<tr><td colspan="2">What is the use of the <b>querySelector()</b> function?

A. it helps to arrange the entities
B. it helps to select the particular entity element
C. it helps to add an entity element
D. it helps to remove the entity element</td><td><b>B</b></td></tr>
<tr><td colspan="2">Which of the following can be used to make one element as a child of another element ?

A. setAttribute()
B. Math.random()
C. appendChild()
D. querySelector()</td><td><b>C</b></td></tr>
<tr><td colspan="3"><b>● End the quiz panel</b></td></tr>
<tr><td></td><td>You get a “hats-off”.

Alright. See you in the next class.</td><td><i>Make sure you have given at least 2 Hats Off during the class for:</i>

 </td></tr>
<tr><td><b>Project Overview</b></td><td><b>UNDERSEA TREASURES</b>

<b>Goal of the Project:</b>

In this project, you will add more elements in the virtual underwater scene for the scuba diver to explore</td><td></td></tr>
</table>

	the world under the previous class's ocean. **Story:** Your friend always wanted to go scuba diving, but he is terrified of water and high waves in the ocean. He always wished to travel under the ocean without going underwater. Write an A-Frame program to add more elements in the scene like fish, treasure coins using components. I am very excited to see how to create a virtual ocean for your friend. Bye!	
	Teacher Clicks	
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?	*The student uses the markdown editor to write their reflections in a reflection journal.*

	● What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	Teacher Reference Code	https://github.com/whitehatjr/PRO-C154-Teacher-Ref-Code
Teacher Activity 2	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/410f86fe5a124a408a8bd26c56b8efff.mp4
Teacher Activity 3	A-Frame Extras Library Link	https://cdn.jsdelivr.net/gh/donmccurdy/aframe-extras@v6.1.1/dist/aframe-extras.min.js
Student Activity 1	Flight Simulation Stage 2	https://github.com/whitehatjr/PRO-C154-Student-Activity
Project Solution Link	Undersea Treasures	https://github.com/whitehatjr/PRO-C154-Project
Teacher Ref. Visual Aid Link	Visual Aid Link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C154_withcues.html
Teacher Ref. In-Class Quiz	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/1987fd17-2dce-49f9-a796-2fc2cef267d4.pdf