Topic	A-Frame ENVIRONMENT AND AUDIO	
Class Description	**Students will learn about the A-Frame environment component. Students will also learn about the audio asset management in A-Frame and how to play sound using the sound component methods.**	
Class	**C163**	
Class time	**45 mins**	
Goal	● Learn about the A-Frame environment component. ● Learn about the A-Frame audio assets. ● Learn to use sound component methods.	
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** **15 mins** **20 mins** **05 mins**

WARM-UP SESSION - 5 mins

CONTEXT

- **Use an A-Frame environment component.**
- **Use the audio component.**

Teacher Starts Slideshow **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 4 to 11**	
Following are the session deliverables: ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	

Class Steps	Teacher Action	Student Action
	We should remove entities which are being created multiple times and are not required after some time in the scene. Can you tell me why this is important?	**ESR**: We should remove entities from the scene so that it does not overload the game or

	Yes, great! And we have already learned how we can find camera direction using Three.js objects and methods in A-Frame. That's quite easy, right? It's really helpful for us to access methods that are present in the Three.js library. We'll be learning to access a few more methods from the three.js library in upcoming classes. Before that, let's make our shooting game a little more interesting by creating a shooting environment and adding our shooter gun with a few more objects in the game like enemy tanks or a few boxes lying around. We will use environments already designed in A-Frame. Are you excited? Let's quickly solve the quiz based on the previous class.	program. It will help us prevent it from crashing down when we are using it for a longer period of time. **ESR:** Yes. **ESR**: Yes.

	Let's get started then.	
Teacher Ends Slideshow		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE ● **Learn about the A-Frame environment component.** ● **Learn about <audio> assets.**		
Step 2: **Teacher-led Activity** **(15 mins)**	*<The teacher clones the code from the Teacher Activity 1.>* [***Teacher Activity 1***] We have a shooter weapon shooting in the camera direction. Now let's move on to making the game scene. What kind of game environments have you come across in shooting games? Well, we can have different types of environments in shooting game scenes such as snowy terrain or a desert or maybe a forest, a palace or a fort. There could be a lot of options where we can play shooting games, right?	**ESR:** Varied. **ESR:** Yes!

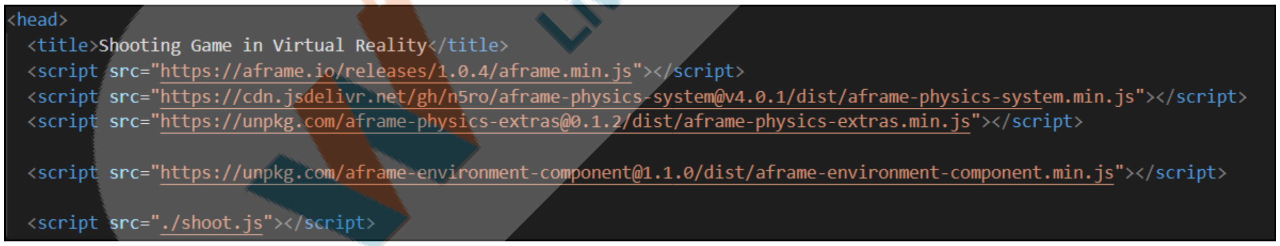
A-Frame environment has a few predefined environment settings available that we can use so that we don't have to create the whole environment ourselves.

For this we will use the A-Frame environment component. This component is available in the "**aframe-environment-component**" library.

Link:
https://unpkg.com/aframe-environment-component@1.1.0/dist/aframe-environment-component.min.js

Let's quickly include this library in the <head> tag in **index.html**.

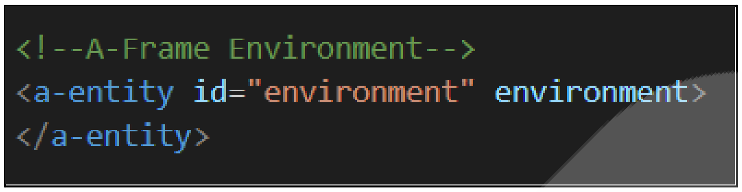
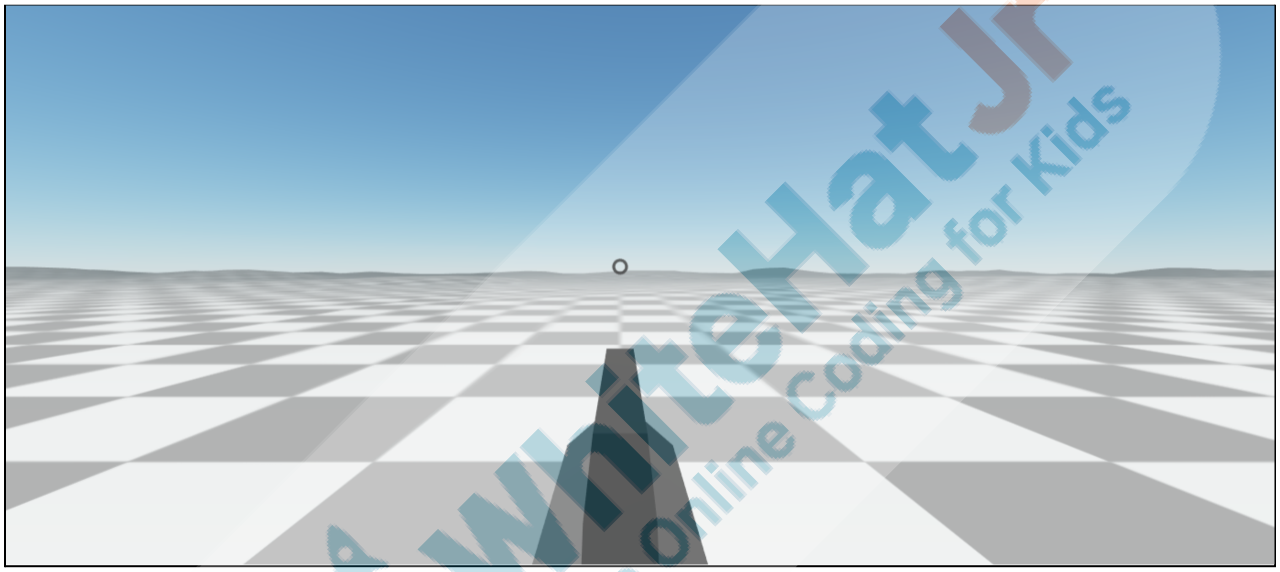
<The teacher includes library source link in the <head> tag.>



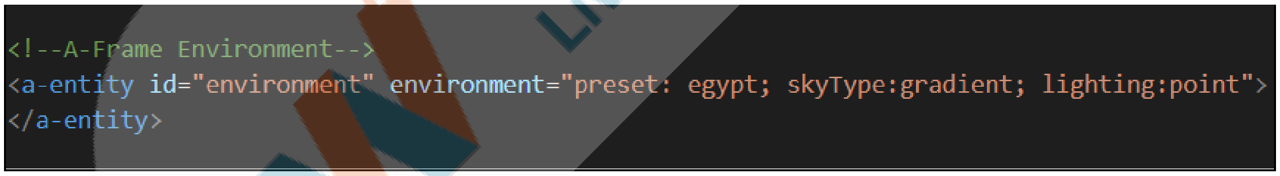
```
<head>
  <title>Shooting Game in Virtual Reality</title>
  <script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
  <script src="https://cdn.jsdelivr.net/gh/n5ro/aframe-physics-system@v4.0.1/dist/aframe-physics-system.min.js"></script>
  <script src="https://unpkg.com/aframe-physics-extras@0.1.2/dist/aframe-physics-extras.min.js"></script>

  <script src="https://unpkg.com/aframe-environment-component@1.1.0/dist/aframe-environment-component.min.js"></script>

  <script src="./shoot.js"></script>
```

Now, we can attach the **environment component** to the scene using <a-entity> tag.

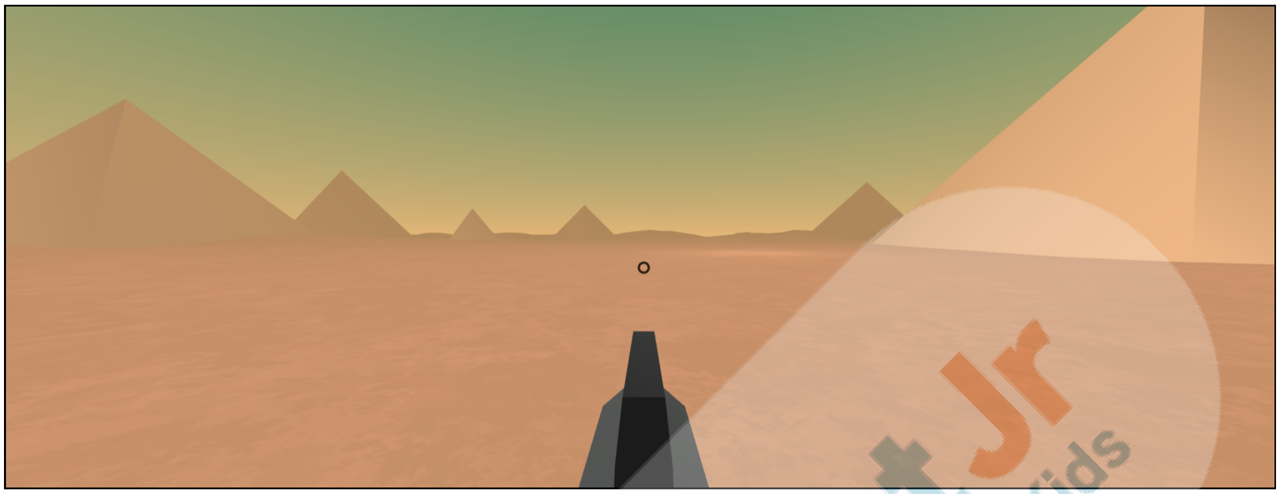
This will set up the default environment properties.

	<The teacher adds the environment component and shows the output.>	
 ```<!--A-Frame Environment--> <a-entity id="environment" environment> </a-entity>```		
		
	There are many properties which are available that we can set for the environment component. We will be learning to use a few of the properties associated with this component to override the default setting, like- **preset-** This is used to set the environment type and valid values are none, default, contact, egypt, checkerboard, forest, goaland, yavapai, goldmine, threetowers, poison, arches, tron, japan, dream, volcano, starry, osiris.	**ESR:** Yes!

	skyType- This is used to set the type of sky element and valid values are color, gradient, atmosphere. **skyColor-** This can be used to set the color of the sky element when the **skyType** is set color or gradient. **lighting-** This is used to set up the lighting in the environment and the valid values are none, distant, point. Now let's set the properties of the environment component and check the output. *<The teacher adds the preset, skyType and lighting properties of the environment component and shows the output.>*	



```
<!--A-Frame Environment-->
<a-entity id="environment" environment="preset: egypt; skyType:gradient; lighting:point">
</a-entity>
```



Well that was pretty quick to have a beautiful environment in our scene.

Let’s make this scene a very basic military shooting scene.

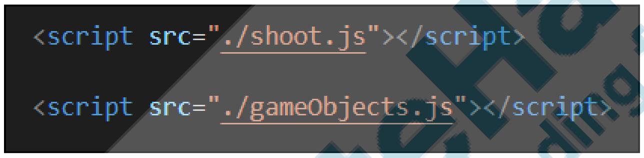
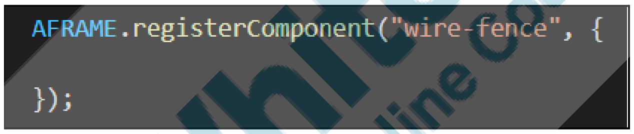
For this we can add a few more elements by ourselves by adding models in the scene.

Since we will make a military shooting scene, we can use some **enemy tanks**, a few **boxes** that might be lying around in the scene and some **wire-fences**.

Let's get started.

To start with, I will be adding some wire-fences. We will use wire-fence 3D models and write a component to add fences in all four directions.

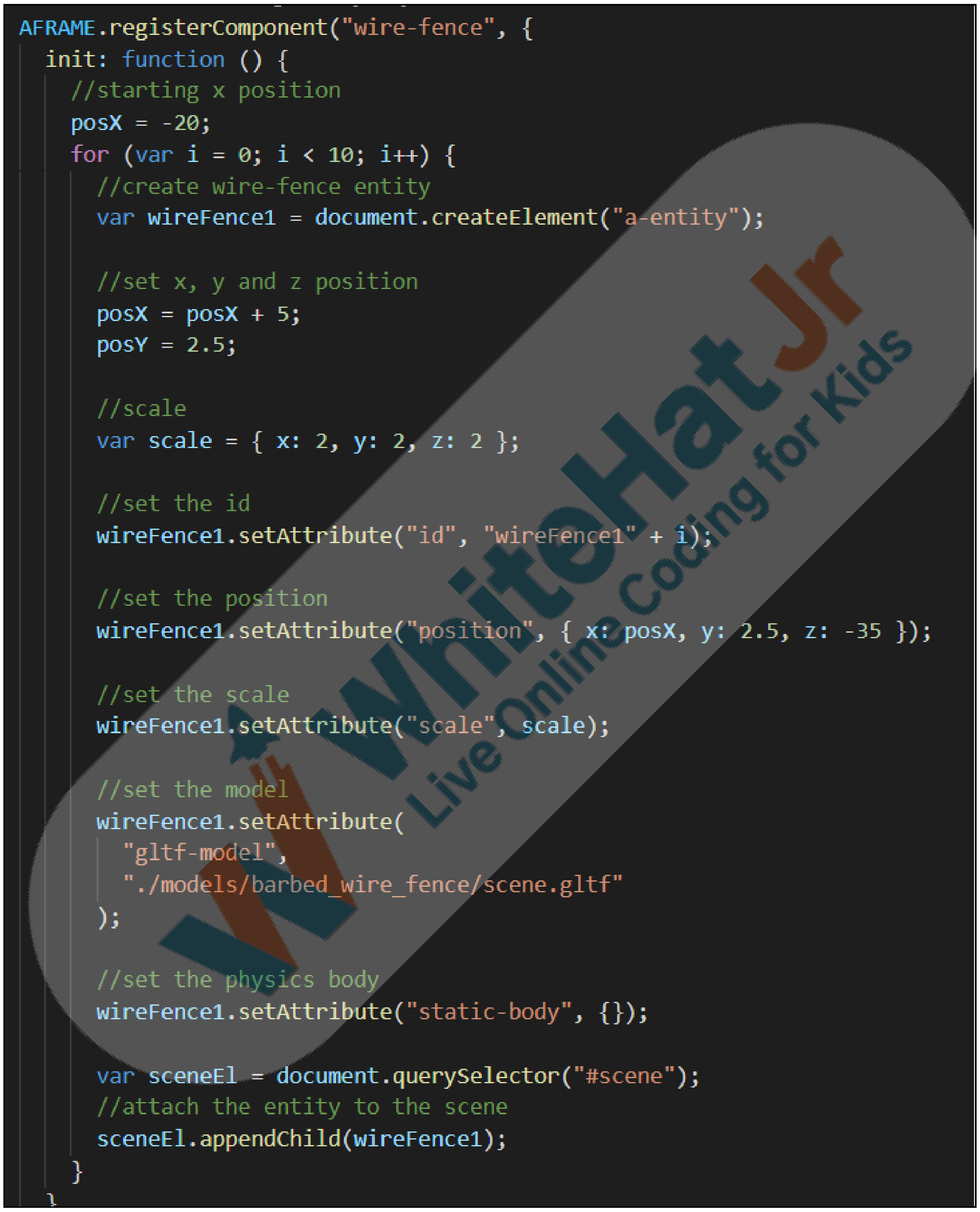
	*<The teacher opens the **gameObjects.js** file and explains it to the student>* 1. *The **gameObjects.js** is added in <head> of the **index.html** file.* 2. *The “**wire-fence**” component is created to add wire-fence as a boundary on all the side in the game environment*	

 `<script src="./shoot.js"></script>`
 `<script src="./gameObjects.js"></script>`
 	
  `AFRAME.registerComponent("wire-fence", {`
 `});`		

	We want to create a boundary fence from left to right at a far away distance from the camera into the screen. We can add “**for** loop” to create **10** entities from left to right in the **.init()** method. ***Note**: Adding so many gLTF models will make the scene very heavy to load based on the system compatibility. If the system is taking a lot of time to load the scene, reduce the number of models on each side to load the scene by reducing the loop counter to 5 or*	

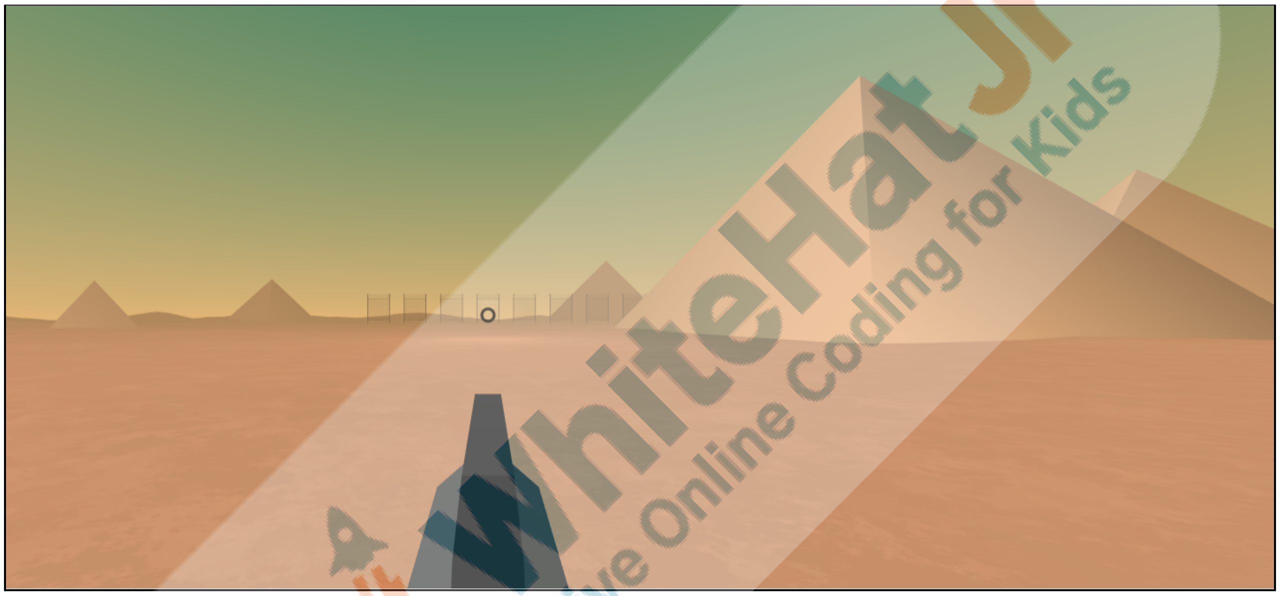
	any other number which will help to load the scene. We can start with x position -20 and keep on increasing it by 5 till the end of the loop, keeping y and z position fixed. Now, we can create the entity using **document.querySelector()** in the loop. And set the attribute- ● id ● position ● scale ● gltf-model ● static-body using **setAttribute()**. Then we can append the child to the scene element. *<The teacher explains the for loop to create an entity and set its properties using **setAttribute()**.>*	

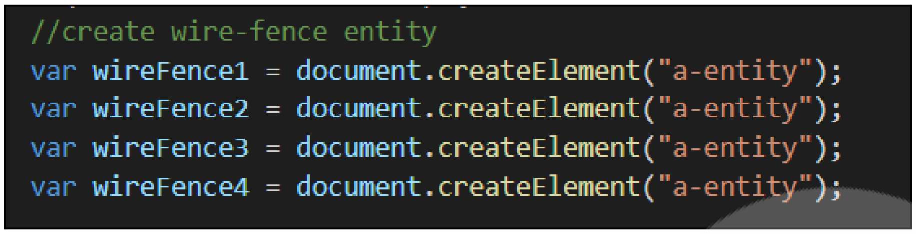
The following is a part of the Teacher ***Boilerplate Code****.*



```
AFRAME.registerComponent("wire-fence", {
  init: function () {
    //starting x position
    posX = -20;
    for (var i = 0; i < 10; i++) {
      //create wire-fence entity
      var wireFence1 = document.createElement("a-entity");

      //set x, y and z position
      posX = posX + 5;
      posY = 2.5;

      //scale
      var scale = { x: 2, y: 2, z: 2 };

      //set the id
      wireFence1.setAttribute("id", "wireFence1" + i);

      //set the position
      wireFence1.setAttribute("position", { x: posX, y: 2.5, z: -35 });

      //set the scale
      wireFence1.setAttribute("scale", scale);

      //set the model
      wireFence1.setAttribute(
        "gltf-model",
        "./models/barbed_wire_fence/scene.gltf"
      );

      //set the physics body
      wireFence1.setAttribute("static-body", {});

      var sceneEl = document.querySelector("#scene");
      //attach the entity to the scene
      sceneEl.appendChild(wireFence1);
    }
```

	Create an ***<a-entity>*** *tag and attach a "****wire-fence****" component in the* ***index.html*** *file and then test the output.*
	
	`<!--Game Objects-->`
	`<a-entity wire-fence></a-entity>`	

		
	Let's have three more variables for creating a fence boundary on the other three sides. *<The teacher write the code to create three more entities and set its properties using setAttribute().>* ***Note****: Take the help of boilerplate code to create wire fence copies of the other three sides.*	

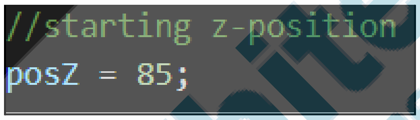


```
//create wire-fence entity
var wireFence1 = document.createElement("a-entity");
var wireFence2 = document.createElement("a-entity");
var wireFence3 = document.createElement("a-entity");
var wireFence4 = document.createElement("a-entity");
```

Now we can repeat steps to set the attributes of these wire fences also.

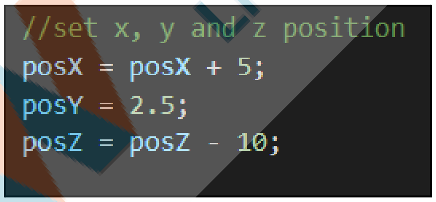
For the fences on the left and right side, we will need to increase the z position and keep x and y fixed going from front to back direction.

Take a variable at top for z position.

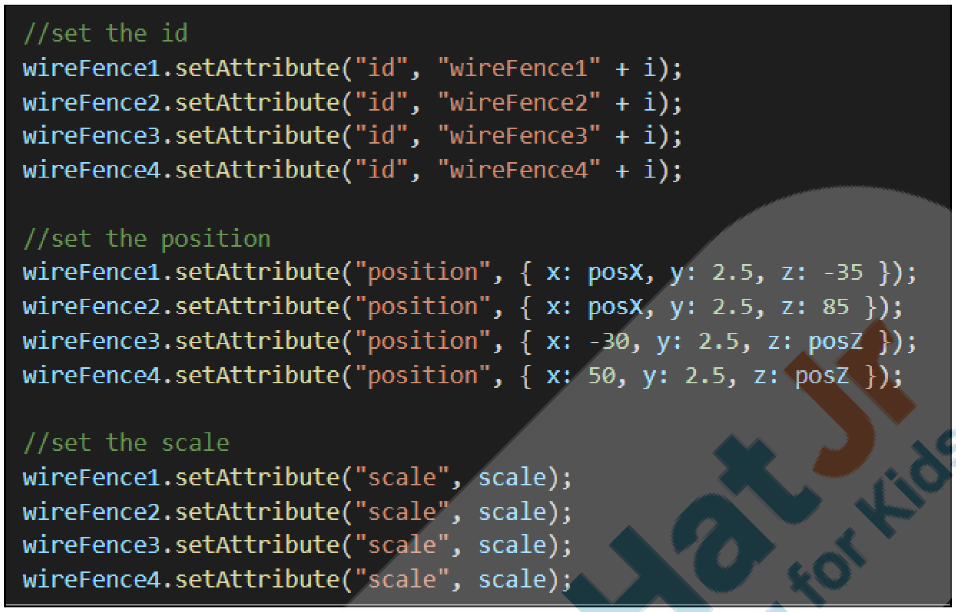


```
//starting z-position
posZ = 85;
```

Decrease the z position till the end of the loop.

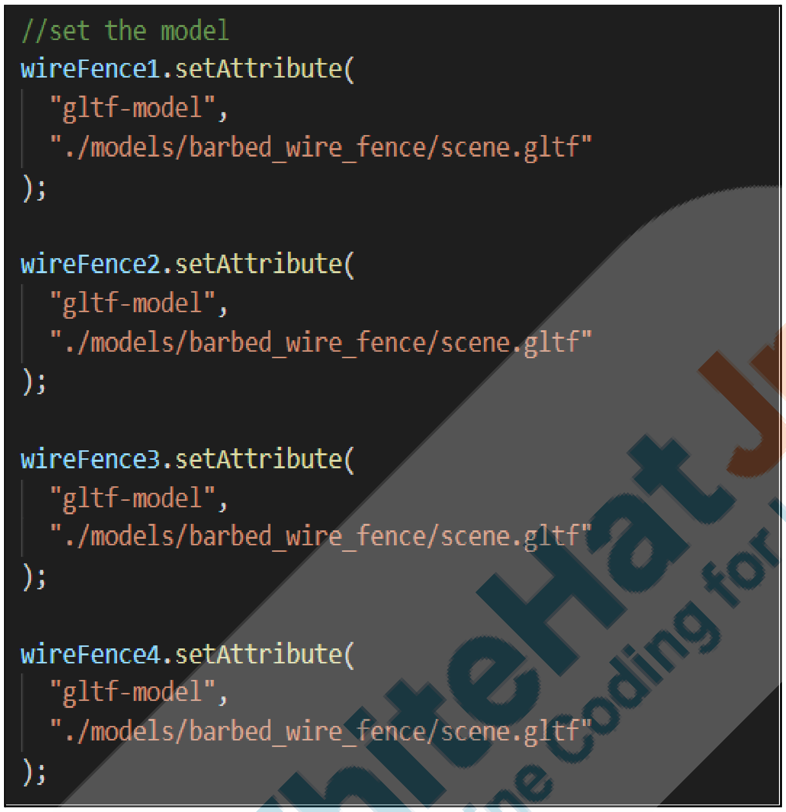


```
//set x, y and z position
posX = posX + 5;
posY = 2.5;
posZ = posZ - 10;
```



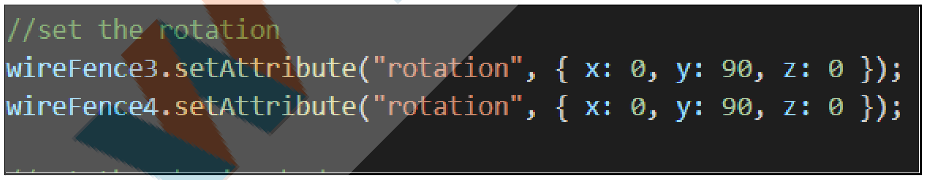
```
//set the id
wireFence1.setAttribute("id", "wireFence1" + i);
wireFence2.setAttribute("id", "wireFence2" + i);
wireFence3.setAttribute("id", "wireFence3" + i);
wireFence4.setAttribute("id", "wireFence4" + i);

//set the position
wireFence1.setAttribute("position", { x: posX, y: 2.5, z: -35 });
wireFence2.setAttribute("position", { x: posX, y: 2.5, z: 85 });
wireFence3.setAttribute("position", { x: -30, y: 2.5, z: posZ });
wireFence4.setAttribute("position", { x: 50, y: 2.5, z: posZ });

//set the scale
wireFence1.setAttribute("scale", scale);
wireFence2.setAttribute("scale", scale);
wireFence3.setAttribute("scale", scale);
wireFence4.setAttribute("scale", scale);
```

	Set the gLTF model attribute for each fence. ***Note**: Adding so many gLTF models will make the scene very heavy to load based on the system compatibility. If the system is taking a lot of time to load the scene, reduce the number of models on each side to load the scene by reducing the loop counter to 5 or any other number which will help to load the scene.*	



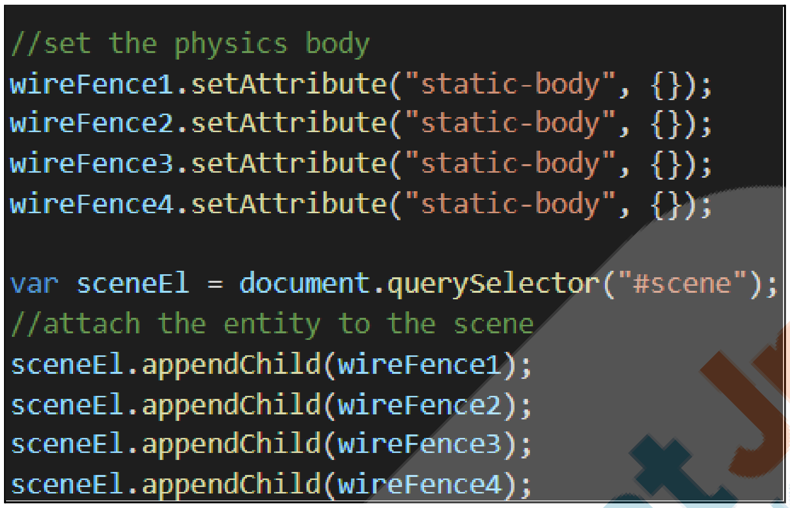
```
//set the model
wireFence1.setAttribute(
  "gltf-model",
  "./models/barbed_wire_fence/scene.gltf"
);

wireFence2.setAttribute(
  "gltf-model",
  "./models/barbed_wire_fence/scene.gltf"
);

wireFence3.setAttribute(
  "gltf-model",
  "./models/barbed_wire_fence/scene.gltf"
);

wireFence4.setAttribute(
  "gltf-model",
  "./models/barbed_wire_fence/scene.gltf"
);
```

Set the rotation of two wire-fences to 90 degrees on y axis to make them vertical.

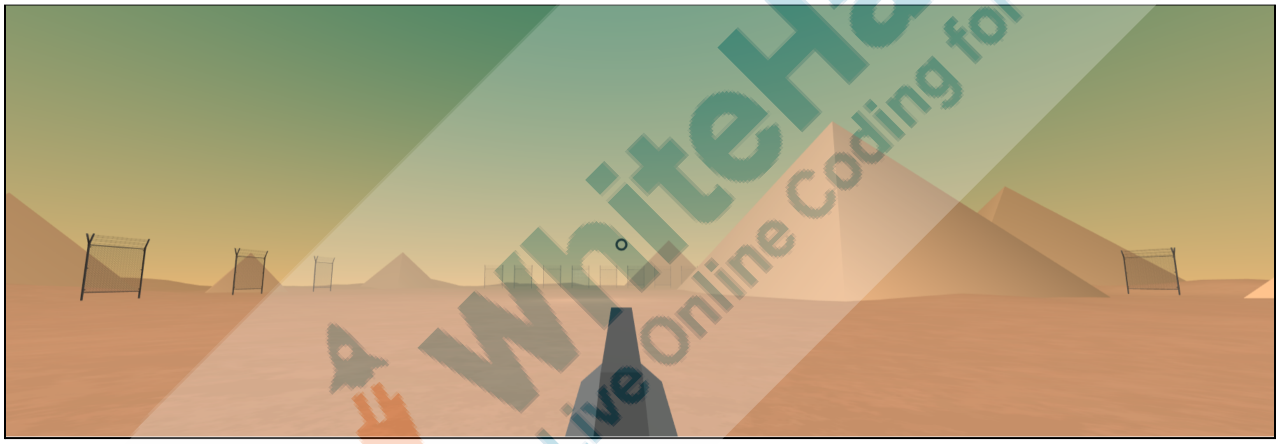


```
//set the rotation
wireFence3.setAttribute("rotation", { x: 0, y: 90, z: 0 });
wireFence4.setAttribute("rotation", { x: 0, y: 90, z: 0 });
```

Set the static-body attribute and append the entities to the scene element.

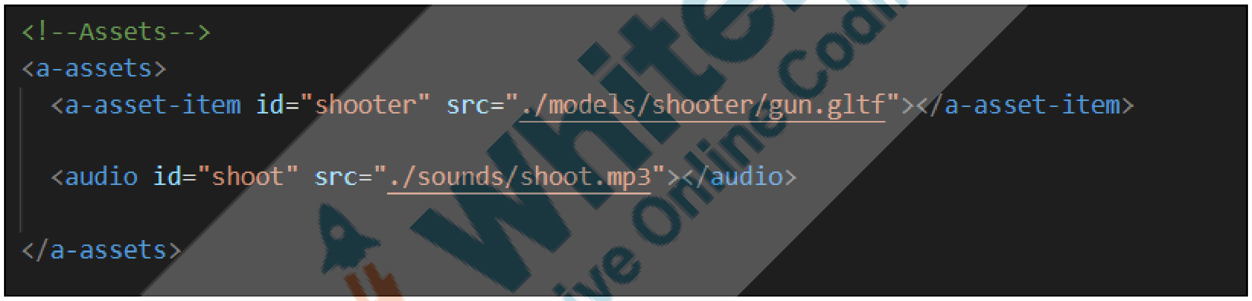


```
//set the physics body
wireFence1.setAttribute("static-body", {});
wireFence2.setAttribute("static-body", {});
wireFence3.setAttribute("static-body", {});
wireFence4.setAttribute("static-body", {});

var sceneEl = document.querySelector("#scene");
//attach the entity to the scene
sceneEl.appendChild(wireFence1);
sceneEl.appendChild(wireFence2);
sceneEl.appendChild(wireFence3);
sceneEl.appendChild(wireFence4);
```



	Now our shooting game is a bit better now. Now let's add shooting sounds. Every time the player shoots the bullet, the shot sound must be played. We have already used the sound in earlier classes, but today we will be using the A-Frame asset management system to add the sound.	

	Do you remember why we use an asset management system in A-Frame? For this we will use A-Frame **<audio>** in asset management, <a-assets>. In <audio>, we can give the id to the audio asset and set the src of the sound. *<The teacher adds id and src path of the shot sound.>*	**ESR:** This is done to preload all assets before rendering the scene.

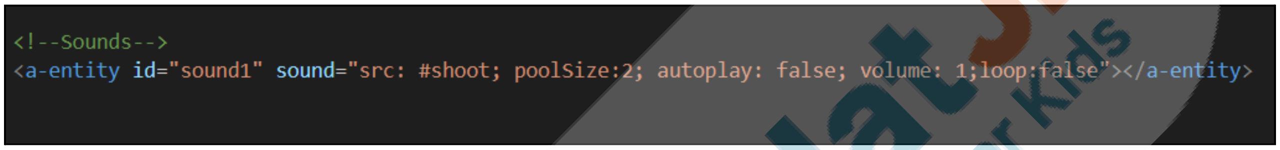


```
<!--Assets-->
<a-assets>
  <a-asset-item id="shooter" src="./models/shooter/gun.gltf"></a-asset-item>

  <audio id="shoot" src="./sounds/shoot.mp3"></audio>

</a-assets>
```

	Now let's create an entity to set the sound component properties. **src-** The id source of the sound set in the <audio> asset. **poolSize-** Number of times the sound can be played simultaneously. **autoplay-** Whether to play sound automatically or not. **volume-** volume of the sound.	

loop- Whether to repeat the sound infinitely.

<The teacher adds the <a-entity> and sets the src, poolSize, autoplay, volume and loop properties of the sound component.>

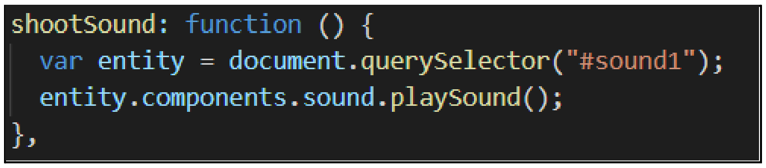


```
<!--Sounds-->
<a-entity id="sound1" sound="src: #shoot; poolSize:2; autoplay: false; volume: 1;loop:false"></a-entity>
```

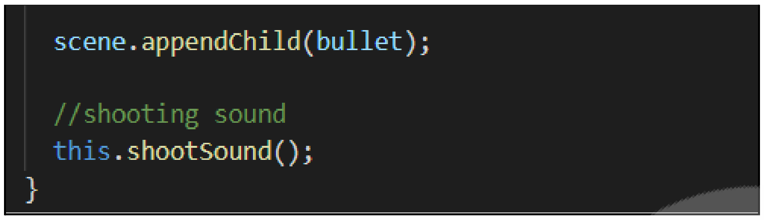
To play the sound, we can write a function, **shootSound()**.

<The teacher writes a function in the shoot.js file in the “bullets” component.>

- Select the entity using document.querySelector().
- Call playSound() method of the component, **entity.components.sound.playSound().**
- Call the shootSound() in the shoot() function.



```
shootSound: function () {
  var entity = document.querySelector("#sound1");
  entity.components.sound.playSound();
},
```



```
scene.appendChild(bullet);

//shooting sound
this.shootSound();
}
```

	Now that we have learned how to add the environment component and audio in the A-Frame scene, you will add the environment and a few more boxes in the scene. Also you will write a function to play the footstep sound whenever the player walks. Are you excited?	**ESR**: Yes!

Teacher Stops Screen Share

So now it's your turn. Please share your screen with me.	

Teacher Starts Slideshow
Slide 12 to 13
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.	

Teacher Ends Slideshow

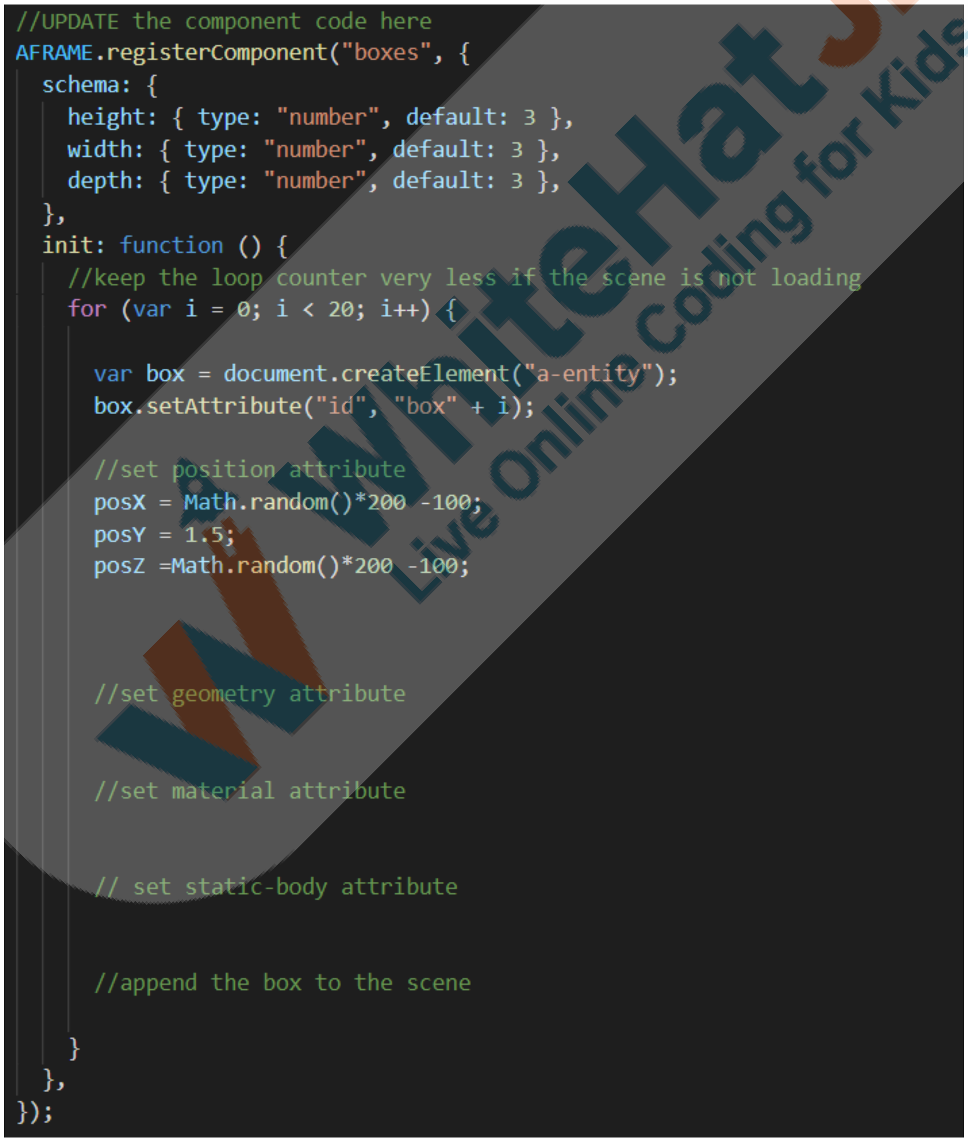
STUDENT-LED ACTIVITY - 20 mins

- **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 • **Add an A-Frame environment component.** • **Add the boxes in the environment.** • **Add the footstep sound when arrow keys are pressed.**		
Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to clone the code from the student activity 1.* *[Student Activity 1]*	
	What should we do now? Yes. *Guide the student to update “**boxes**” components to add the boxes at random positions.* The **boxes** component has a schema with **height**, **width** and **depth** of boxes each with default values of 3. The **box** variable is used to create the box entity. The boxes in the game's environment will be lying at random positions. For random numbers, there is a method called **random()** in the Javascript Math library which can be accessed using **Math.random()**.	**ESR**: Add the game objects.

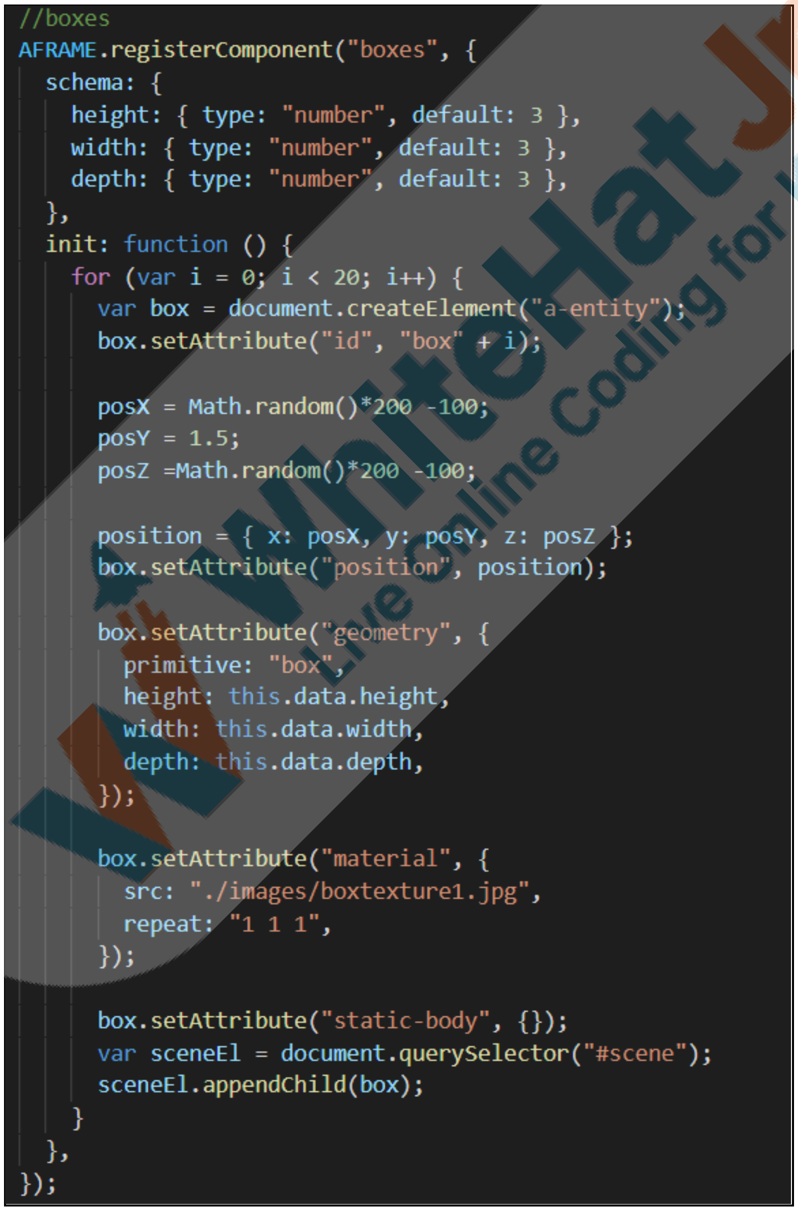
	The height of boxes will be fixed as all of the boxes will lie on ground, hence the y position attribute will remain constant. But the x and z position will be random to change the position of the horizontally from left to right and to change the depth.	

The below code is a part of the ***Student Boilerplate Code.***

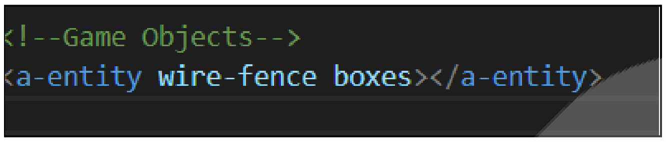


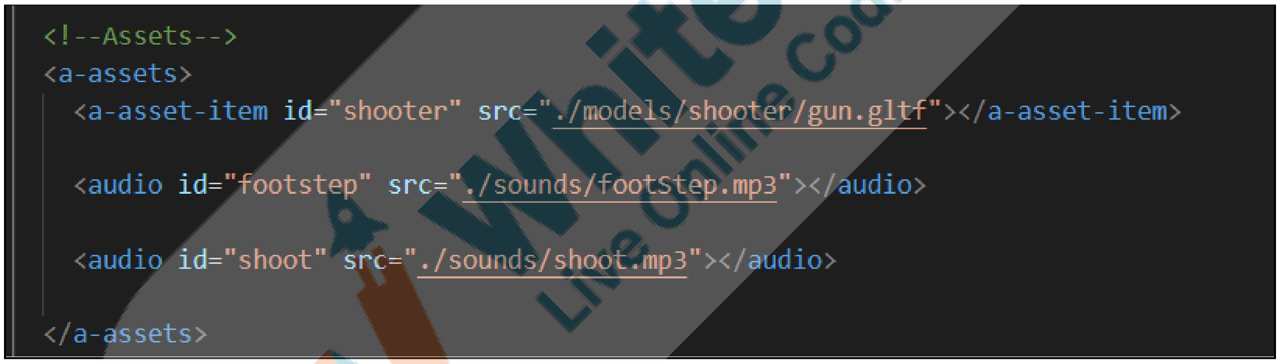
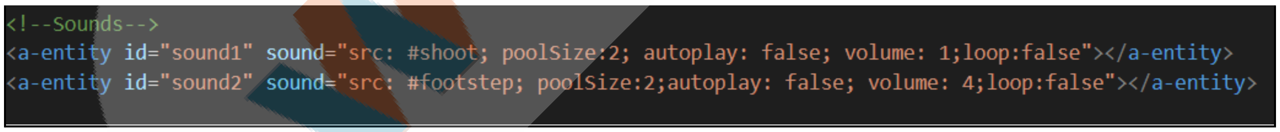
```
//UPDATE the component code here
AFRAME.registerComponent("boxes", {
  schema: {
    height: { type: "number", default: 3 },
    width: { type: "number", default: 3 },
    depth: { type: "number", default: 3 },
  },
  init: function () {
    //keep the loop counter very less if the scene is not loading
    for (var i = 0; i < 20; i++) {

      var box = document.createElement("a-entity");
      box.setAttribute("id", "box" + i);

      //set position attribute
      posX = Math.random()*200 -100;
      posY = 1.5;
      posZ =Math.random()*200 -100;


      //set geometry attribute


      //set material attribute


      // set static-body attribute


      //append the box to the scene

    }
  },
});
```

	Guide the student to set the: ● Take a **position** variable to set the position attribute ● Set the **geometry** attribute. ● Set the **static-body** attribute. ● Append the entity to the scene element.	



```
//boxes
AFRAME.registerComponent("boxes", {
  schema: {
    height: { type: "number", default: 3 },
    width: { type: "number", default: 3 },
    depth: { type: "number", default: 3 },
  },
  init: function () {
    for (var i = 0; i < 20; i++) {
      var box = document.createElement("a-entity");
      box.setAttribute("id", "box" + i);

      posX = Math.random()*200 -100;
      posY = 1.5;
      posZ =Math.random()*200 -100;

      position = { x: posX, y: posY, z: posZ };
      box.setAttribute("position", position);

      box.setAttribute("geometry", {
        primitive: "box",
        height: this.data.height,
        width: this.data.width,
        depth: this.data.depth,
      });

      box.setAttribute("material", {
        src: "./images/boxtexture1.jpg",
        repeat: "1 1 1",
      });

      box.setAttribute("static-body", {});
      var sceneEl = document.querySelector("#scene");
      sceneEl.appendChild(box);
    }
  },
});
```

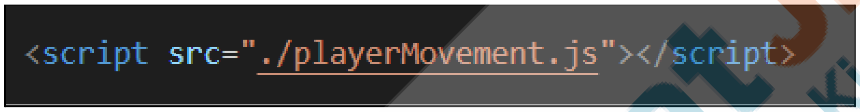
	Guide the student to attach the components in the ***index.html*** *file.*	
	 `<!--Game Objects-->` `<a-entity wire-fence boxes></a-entity>`	
	The **footStep.mp3** sound file path is added using **<audio>** in **<a-assets>** *Guide the student to add the* ***sound2*** *source with id="****footstep****" using* ***<a-entity>*** *and set its attribute.*	

 `<!--Assets-->`
`<a-assets>`
`  <a-asset-item id="shooter" src="./models/shooter/gun.gltf"></a-asset-item>`
 	
`  <audio id="footstep" src="./sounds/footStep.mp3"></audio>`
 	
`  <audio id="shoot" src="./sounds/shoot.mp3"></audio>`
 	
`</a-assets>`
 `<!--Sounds-->`
`<a-entity id="sound1" sound="src: #shoot; poolSize:2; autoplay: false; volume: 1;loop:false"></a-entity>`
`<a-entity id="sound2" sound="src: #footstep; poolSize:2;autoplay: false; volume: 4;loop:false"></a-entity>`		

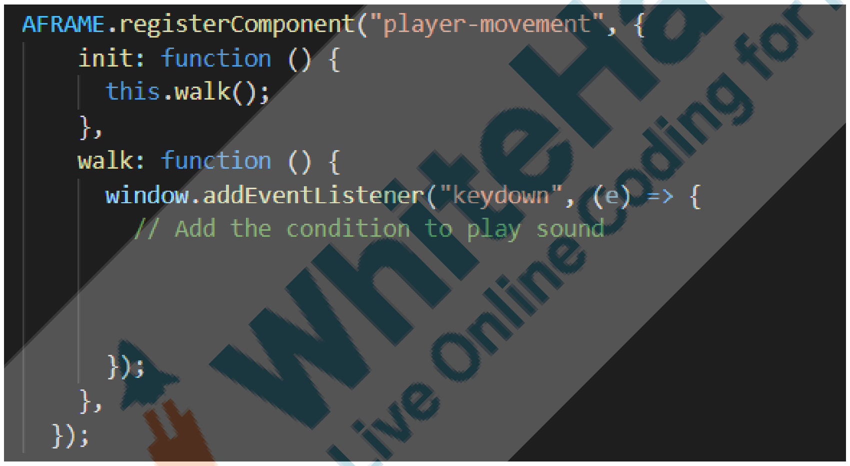
	The **playerMovement.js** file is added to the <head> tag in **index.html** file. The"**player-movement**" component is added to the **playerMovement.js** file	

	The “**player-movement**” component is attached to the weapon entity in index.html file The **walk()** function will be used to add the keydown event listener to play when the player is moving around the scene using arrow keys!	

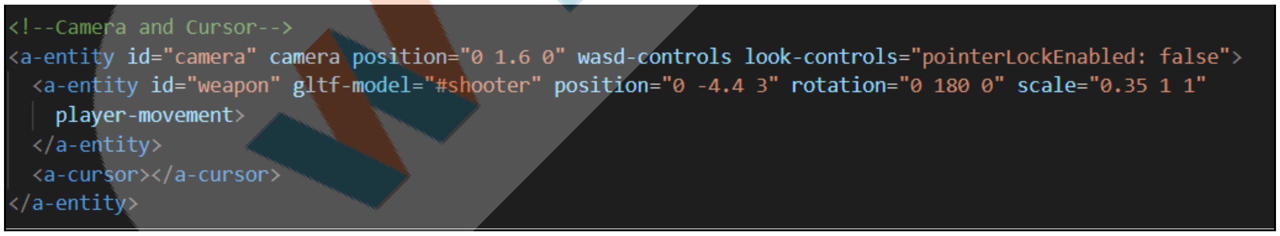
*The below code is a part of the **Student Boilerplate Code.***



```html
<script src="./playerMovement.js"></script>
```



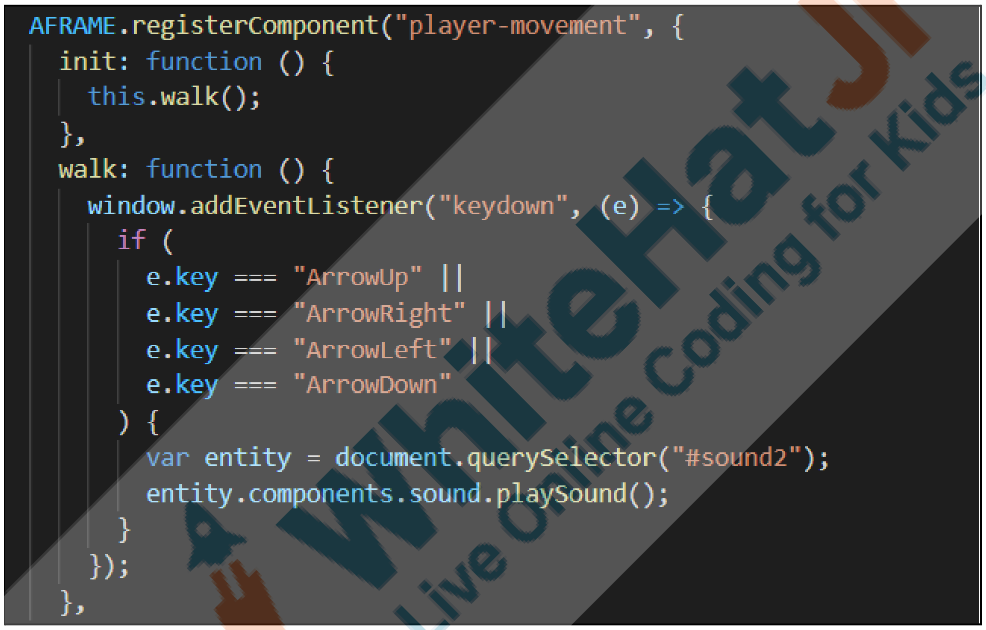
```javascript
AFRAME.registerComponent("player-movement", {
    init: function () {
      this.walk();
    },
    walk: function () {
      window.addEventListener("keydown", (e) => {
        // Add the condition to play sound




      });
    },
  });
```



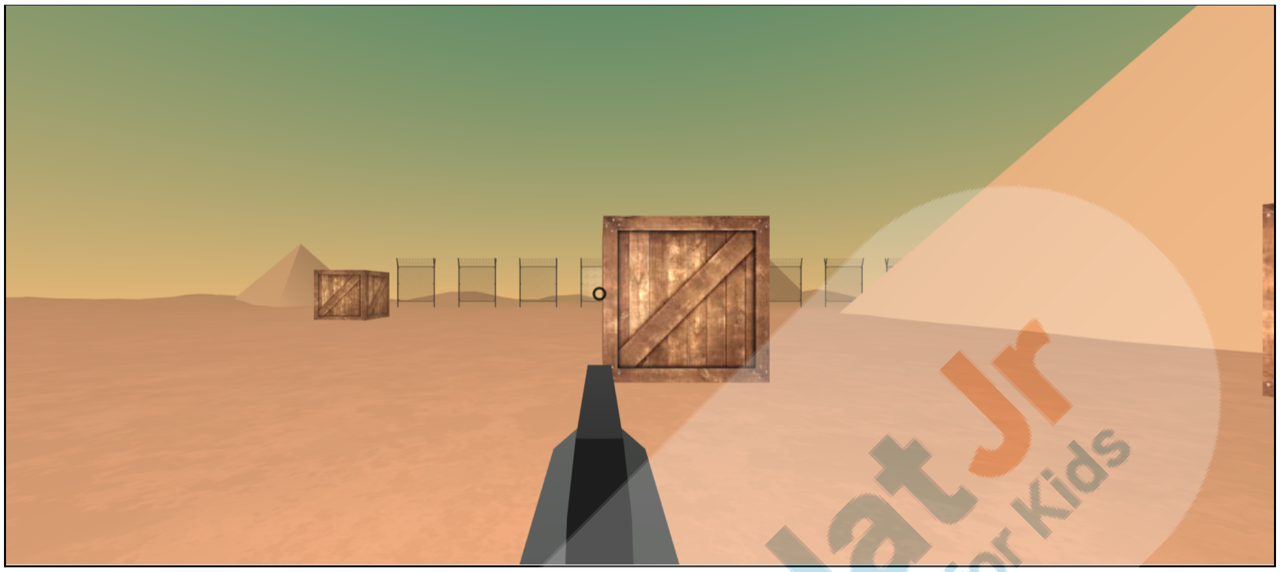
```html
<!--Camera and Cursor-->
<a-entity id="camera" camera position="0 1.6 0" wasd-controls look-controls="pointerLockEnabled: false">
  <a-entity id="weapon" gltf-model="#shooter" position="0 -4.4 3" rotation="0 180 0" scale="0.35 1 1"
    player-movement>
  </a-entity>
  <a-cursor></a-cursor>
</a-entity>
```

	Guide the student to write a play sound when arrow keys are pressed: ● Add the “**keydown**” event listener inside the the “**walk**” function	

	● Add the "**if**" condition to check if the arrow keys are pressed. ● Select the sound entity. ● Call the **playSound()** method of the sound component. ● Call the "**walk**" function in the .init() method.	



```
AFRAME.registerComponent("player-movement", {
  init: function () {
    this.walk();
  },
  walk: function () {
    window.addEventListener("keydown", (e) => {
      if (
        e.key === "ArrowUp" ||
        e.key === "ArrowRight" ||
        e.key === "ArrowLeft" ||
        e.key === "ArrowDown"
      ) {
        var entity = document.querySelector("#sound2");
        entity.components.sound.playSound();
      }
    });
  },
```



	Great job making the military shooting practice area! But there are two issues in the game scene right now. Can you tell me what are those? *<Let the student think and come up with some ideas.>* First one is, in the game when you move around the scene, you can pass through the boxes and other objects in the scene. Well this should not happen, right? Also every time you load the game, the position of the boxes change.	**ESR**: Varied. **ESR**: Yes.

WhiteHat Jr
Live online Coding for Kids

	This shouldn't happen in the game. The boxes must have fixed positions, right? We will fix these two issues in the next class, meanwhile I want you to come up with a few solutions and ideas of your own.	**ESR**: Yes.

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 14 to 19

Activity details
Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

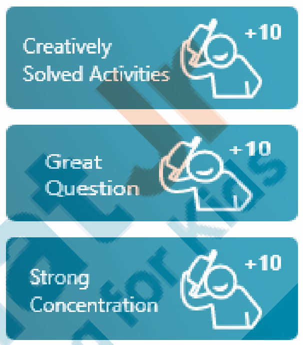
Continue WRAP-UP Session
Slide 20 to 25

Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

• **Compliment the student for her/his effort in the class.** • **Encourage the student to think and come up with their own solutions.**		
	You get a “hats-off”. Alright. See you in the next class.	*Make sure you have given at least 2 Hats Off during the class for:*  Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections		
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? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write their reflections in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Teacher Boilerplate Code	https://github.com/whitehatjr/PRO-C163-Teacher-Activity
Teacher Activity 2	A-Frame environment component Link.	https://unpkg.com/aframe-environment-component@1.1.0/dist/aframe-environment-component.min.js
Teacher Activity 3	Teacher Reference Code	https://github.com/whitehatjr/PRO-C163-Teacher-Ref
Teacher Activity 4	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/A-Frame+environment+%26+audio.mp4
Student Activity 1	Student Boilerplate Code	https://github.com/whitehatjr/PRO-C163-Student-Activity
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/11417c30-58aa-4b7f-a91c-d621

		ed7cd540.pdf
Teacher Reference 2	Project Solution	https://github.com/whitehatjr/PRO-C163-Project-Solution
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/06c9824c-4875-42f7-84be-2767b9011cdc.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/bf4d837d-315e-4dc0-8016-e894a75d3eeb.pdf