Topic	CAMERA DIRECTION	
Class Description	**Students learn how to follow the camera direction in A-Frame. Students will learn about the relationship between A-Frame and Three.js. Students will learn how to access Three.js objects using A-Frame entities.**	
Class	**C161**	
Class time	**45 mins**	
Goal	• Learn about the camera direction in A-Frame. • Learn how to access Three.js objects using A-Frame entities	

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

- **Introduce relation between A-Frame and Three.js.**

Teacher Starts Slideshow 

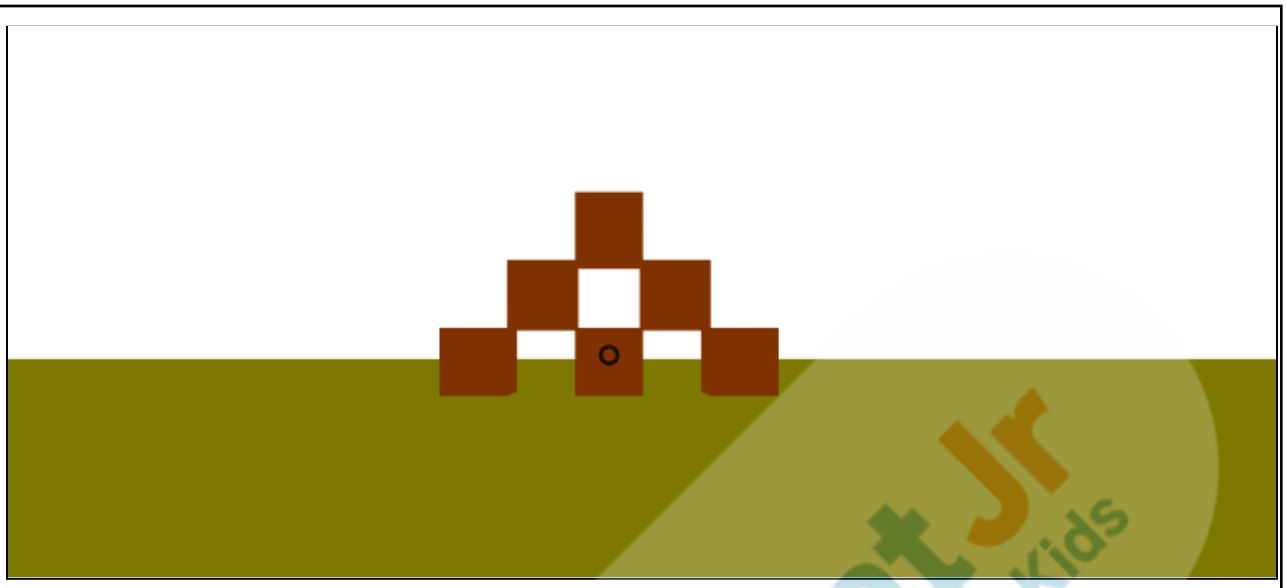
Slide 1 to 3 Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**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 12** 	
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 have made quite a few small VR applications that can be useful in real life scenarios, like a virtual reality room design used by various real estate agents for showcasing their houses or rooms before selling them to the customer. We also designed a few simulations which are used In real life to give virtual reality experience to learn something.	

	In the upcoming classes, we will be learning how to design a straightforward virtual reality shooting game using the concept we have covered and learn a few more concepts that will be used while making the game. Are you excited?	**ESR:** Yes.
Teacher Ends Slideshow 		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Understand the relation between A-Frame and Three.js.** • **Use Three.js method to get the camera direction.**		
Step 2: Teacher-led Activity (15 mins)	Let's start by adding some basic elements to the scene. We can add a light and camera element in the scene and attach the cursor element to the camera. Then we can add a plane for the ground and a few boxes to shoot at. *<Teacher codes to add all the elements in the scene.>*	

```
<!--Light-->
<a-entity light="type: ambient; color: yellow; intensity:0.5"></a-entity>

<!--Camera and Cursor-->
<a-entity
  id="camera"
  camera
  position="0 1.6 0"
  wasd-controls
  look-controls="pointerLockEnabled: true"
>
  <a-cursor></a-cursor>

</a-entity>

<!--Ground-->
<a-plane
  id="ground"
  repeat="5 5"
  position="0 1 0"
  rotation="-90 0 0"
  height="200"
  width="200"
  color="#FBF2D4"
  static-body
  visible="true"
>
</a-plane>
```

```
<!--Boxes-->
<a-box position="-2 1.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>

<a-box position="0 1.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>

<a-box position="2 1.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>

<a-box position="-1 2.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>

<a-box position="1 2.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>

<a-box position="0 3.5 -10" color="tomato" depth="1" height="1" width="1"></a-box>
</a-scene>
```



	Now, what would you do to shoot the bullets?	**ESR:** We can shoot the bullet every time the screen is clicked, or some key is pressed.
	So what would be the first step to shoot the bullets?	**ESR:** We can write a component to create bullet entities.
	Yes, correct! Let's get cracking!	
	Let's write a component, called “bullets”, which will create a bullet every time the “z” key is pressed. Then we can create an entity on the “keydown” event.	

	We can set different attributes with the help of the **setAttribute()** method.	

```
AFRAME.registerComponent("bullets", {

  init: function () {
    this.shootBullet();
  },
  shootBullet: function () {
    window.addEventListener("keydown", (e) => {
      if (e.key === "z") {

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

        bullet.setAttribute("geometry", {
          primitive: "sphere",
          radius: 0.1,
        });

        bullet.setAttribute("material", "color", "black");


      }
    });
  },
});
```

	What do you think will be the position of the bullet entity that means from where we should start shooting the bullet? A-Frame camera represents the first-person view that means if you are standing at a place, the way you will see the world objects or entities around you, that's how the camera is positioned.	**ESR**: The position of the bullets should be the same as the camera.

Keeping that in mind, if we want to shoot the bullet, we should start shooting it from the camera's position.

Let's select the camera attribute and get the position attribute of the camera.

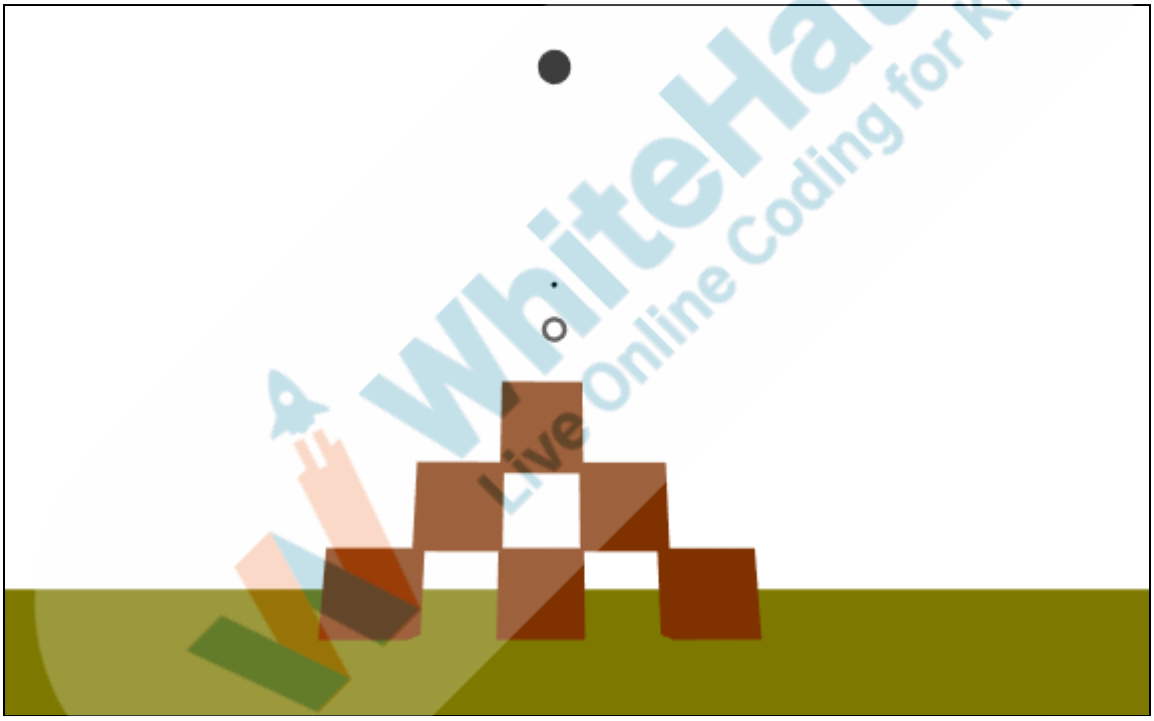
This can be used to set the position of the bullet.

```
shootBullet: function () {
  window.addEventListener("keydown", (e) => {
    if (e.key === "z") {
      var bullet = document.createElement("a-entity");

      bullet.setAttribute("geometry", {
        primitive: "sphere",
        radius: 0.1,
      });

      bullet.setAttribute("material", "color", "black");

      var cam = document.querySelector("#camera");

      pos = cam.getAttribute("position");

      bullet.setAttribute("position", {
        x: pos.x,
        y: pos.y,
        z: pos.z,
      });

      cam.appendChild(bullet);
    }
  });
}
```

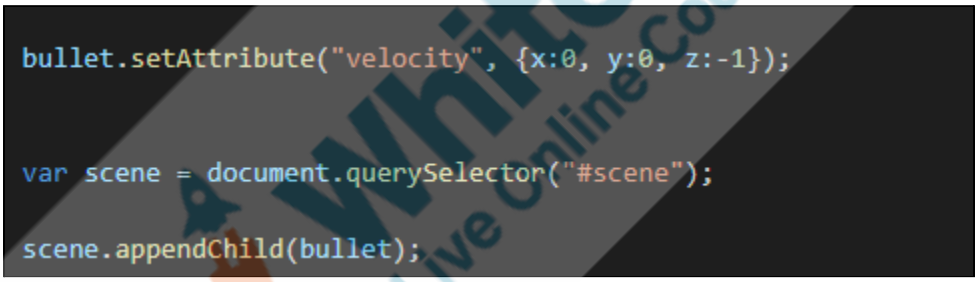
	Whenever we press the “z” key, we are not able to see the bullet entity because it has been created exactly at the camera position. What do you think can be done to see the bullet? To see the bullet, let’s start moving it away from the camera. To move any element, “velocity” is attached every time you create an entity in the physics world. Let’s add the A-Frame physics library and set the velocity attribute of the bullet entity. To set the velocity, we will need to decide in which direction we want to shoot the bullet. The direction of velocity can be set using a 3D point value in x, y and z direction. Do you remember which axis points to the camera direction? Yes, the negative z-axis is pointing towards the screen and the positive z-axis is pointing towards us.	**ESR**: Varied. **ESR**:The z-axis.

	To move the bullet into the screen, we should set the velocity direction in the negative z-axis.	

```
bullet.setAttribute("position", {
  x: pos.x,
  y: pos.y,
  z: pos.z,
});

bullet.setAttribute("velocity", { x: 0, y: 0, z: -1 });

cam.appendChild(bullet);
```



	We can see that the bullet starts coming from the camera position but there's some issue with this. What happens if I start moving the camera with the help of arrow keys?	**ESR**: The bullet is also moving in the same direction.

	Can you tell me why this is happening? The bullet starts moving with the camera position because the bullet is created as the child of the camera, hence if the camera moves, every child entity of that is also affected. What do you think can be done to fix this? Instead of making a bullet as the child of the camera, we should make it a child of the scene.	**ESR**: Varied. **ESR**: Varied.
	 	
	But there is one more issue, if I click and drag the mouse on the screen in A-Frame scene, the angle of rotation for the camera changes. This is similar to the way you move your head up and down or right and left along the y and x axis. But the velocity of the bullet is fixed in one direction; it does not follow the direction of rotation of the camera,	

	hence it moves in only a single direction. Let's understand what is happening with the A-Frame Visual Inspector tool. We can use **<Ctrl+Alt+i>** to open the A-Frame Visual Inspector. ***[Camera Direction 1 Illustration]*** If you click on the particular entity in the Visual Inspector that entity is highlighted. Let's select the camera on the left navigation panel. In A-Frame, the camera is represented by a frustum of the pyramid shape and its open end point is towards the negative z-axis. Here the white line represents the direction in which the camera is pointing. By default, camera keeps pointing towards the negative z-axis If I rotate the camera, it will move the whole frustum of the camera. As the frustum moves, you can see that the rotation of the camera changes.	

If you shoot the bullet, you will see that the bullet is coming from the camera's position, but it is not going through the white light.

If the camera is pointing towards the negative z-axis, same as the current velocity direction, then the bullet will go in the same direction.

Once the camera is rotated, it will not go in the other direction as the velocity direction is fixed.

To fix this, we need to find a way to get the camera's direction and use that as the direction of velocity for the bullet.

This can be done with the help of a predefined method in Three.js library.

We know that A-Frame is built on Three.js. We need to see how we can access any object or function present in Three.js using A-Frame.

A-Frame's <a-scene> is mapped with a Three.js scene.

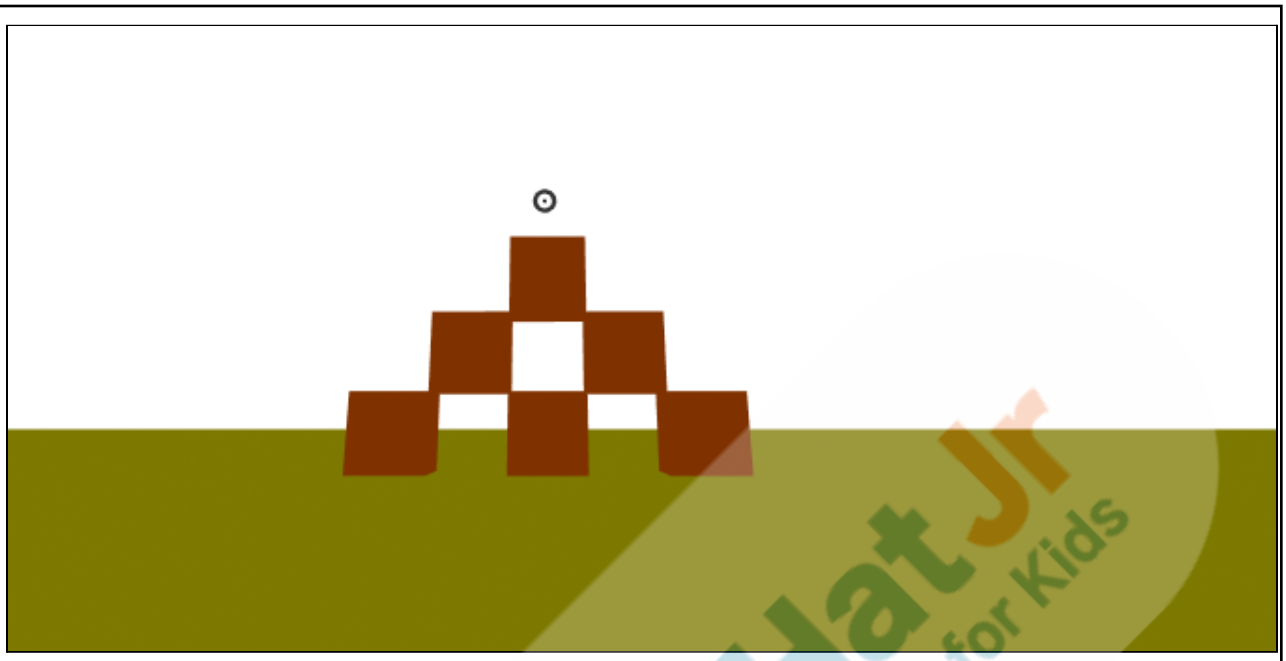
A-Frame's <a-entity> is mapped to one or more Three.js objects.

The Three.js scene is accessible from the <a-scene> element as **.object3D**.

	To access the Three.js scene we can use: **document.querySelector('a-scene').object3D** Every A-Frame entity (<a-entity>) has its own THREE.Object3D. To access entity as the Three.js object we can use: **document.querySelector('a-entity').object3D** Let's get the camera entity as the Three.js object using: document.querySelector('#camera').object3D Now, to define the Three.js vector, we need to create Three.js vector objects using "**new THREE.Vector3()**". We can create a variable direction to create a new Three.js vector object. Now, to get the direction of the camera element we can use the .getWorldDirection(vectorVariable) method defined in Three.js. camera.getWorldDirection(direction), will set the value of the vector given by the method in the direction variable. Let's check the value of direction using console.log(direction).	

	Click and drag the mouse to change the direction of the camera and check the result in the console. We can see that as we move the camera, the value of direction changes. To increase the bullet speed, we can multiply the direction vector with a number using **multiplyScalar()**. We will be using the negative number in the direction going into the screen. Now let's test the output. We can see that the bullet always goes in the camera direction. ***[Camera Direction 2 Illustration]***	

```
var camera = document.querySelector("#camera").object3D;

//get the camera direction as Three.js Vector
var direction = new THREE.Vector3();
camera.getWorldDirection(direction);

//set the velocity and it's direction
bullet.setAttribute("velocity", direction.multiplyScalar(-10));

var scene = document.querySelector("#scene");

scene.appendChild(bullet);
```



	Now that we have learned how to create custom components and use them in the A-Frame scene, it's your turn.	
	Teacher Stops Screen Share	
	We have learned how we can use Three.js to get the camera direction to set the bullet's shooting direction. Now you will add the shooting gun model and write a component to shoot the bullet out of the model. Please share your screen with me.	

Teacher Starts Slideshow

Slide 13 to 14

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 • **Learn how to set the velocity direction.** • **Learn how to get the THREE.js camera direction using A-Frame.**		
Step 3: Student-Led Activity (20 mins)	*Guide the student to download the code from the student activity 1.* ***[Student Activity 1]*** *C1: Guide the student to open the Visual Studio Code Editor and create the shoot.js file.* *Include the file in index.html before <a-scene>.* *C2: Guide the student to write a component to create bullet entities and add in the index.html file.*	*The student writes the code to create a JavaScript file and include that in index.html.*

```
AFRAME.registerComponent("bullets", {
  init: function () {
    this.shootBullet();
  },
  shootBullet: function () {

  },
});
```

```
<!--Bullets-->
<a-entity bullets></a-entity>

<!--Ground-->
<a-plane
```

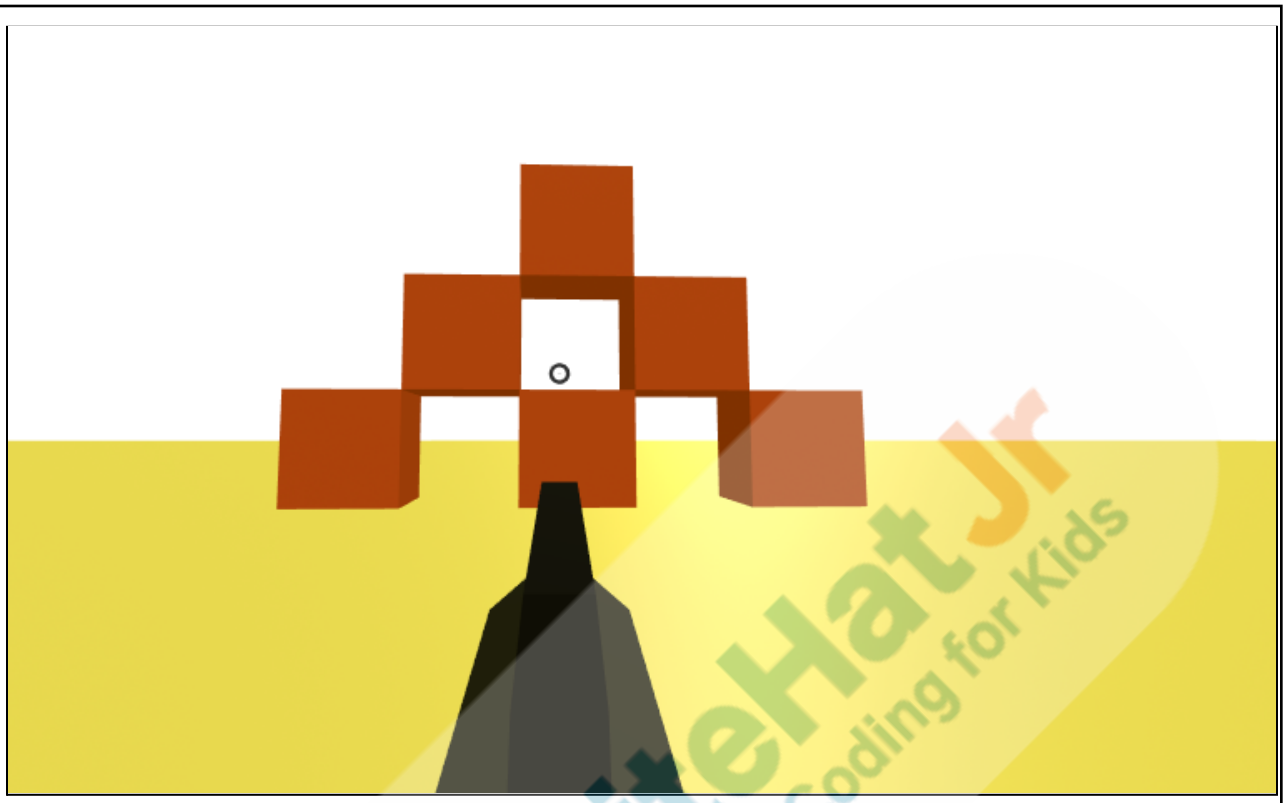
	C3: Guide the student to write the code in shootBullet function to create a bullet entity and set its position and direction of the shoot.	

```
shootBullet: function () {
  window.addEventListener("keydown", (e) => {
    if (e.key === "z") {
      var bullet = document.createElement("a-entity");

      bullet.setAttribute("geometry", {
        primitive: "sphere",
        radius: 0.1,
      });

      bullet.setAttribute("material", "color", "black");

      var cam = document.querySelector("#camera");

      pos = cam.getAttribute("position");

      bullet.setAttribute("position", {
        x: pos.x,
        y: pos.y,
        z: pos.z,
      });

      var camera = document.querySelector("#camera").object3D;

      //get the camera direction as Three.js Vector
      var direction = new THREE.Vector3();
      camera.getWorldDirection(direction);

      //set the velocity and it's direction
      bullet.setAttribute("velocity", direction.multiplyScalar(-10));

      var scene = document.querySelector("#scene");

      scene.appendChild(bullet);
    }
  });
},
```

	C4: Guide the student to add the gLTF model of the gun shooter in the scene as the child entity of the camera and test the output.	

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

</a-assets>
```

```
<!--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"
    >
  </a-entity>
  <a-cursor></a-cursor>
</a-entity>
```



Amazing!

You have done a great job of writing your own component to shoot bullets.

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 15 to 20

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 21 to 26

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. I will look forward to seeing you create your own component.	*Make sure you have given at least 2 Hats Off during the class for:*   

PROJECT OVERVIEW DISCUSSION
Refer the document below in Activity Links Sections

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 their reflections in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Camera Direction 1	https://curriculum.whitehatjr.com/PRO+Asset/camera+directon1.mp4
Teacher Activity 2	Camera Direction 2	https://curriculum.whitehatjr.com/PRO+Asset/camera+direction2.mp4
Teacher Activity 3	Teacher Reference code	https://github.com/whitehatjr/PRO-C161-Teacher-Ref
Teacher Activity 4	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/shooter.mp4
Student Activity 1	Blank Activity	https://github.com/whitehatjr/PRO-C161-Student-Activity
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/b04e0eca-3da9-4339-a4bc-9a9b2cef7ce9.pdf
Teacher Reference 2	Project Solution	https://github.com/whitehatjr/PRO-C161-Project-Solution

Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/2e21a7e3-267c-42cd-806a-a63357c2be4d.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/ea805bd5-42d6-4855-b27a-45d70cf330f5.pdf