

Topic	VECTOR DIRECTION	
Class Description	Students learn to find the direction between two position vectors of two elements in the A-Frame scene using Three.js methods.	
Class	C165	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>Learn how to find the direction vector between two position vectors.</li> <li>Learn to use Three.js method to find the direction vector.</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>Teacher Resources               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> <li>Student Resources               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> </ul>	
Class structure	<b>Warm-Up</b> <b>Teacher-led Activity</b> <b>Student-led Activity</b> <b>Wrap-Up</b>	<b>05 mins</b> <b>15 mins</b> <b>20 mins</b> <b>05 mins</b>
WARM-UP SESSION - 5 mins		
<b>CONTEXT</b> <ul style="list-style-type: none"> <li>Learn to find the direction vector using Three.js objects and methods in A-Frame.</li> </ul>		



### 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 15

#### 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
	<p>We have already learned how we can find camera direction using Three.js objects and methods in A-Frame. That's quite easy, right?</p> <p>It's really helpful for us to access methods that are present in the Three.js library.</p> <p>We'll be learning to access a few more methods from the three.js library</p>	<p><b>ESR:</b> Yes.</p>

	<p>and we will be finishing the simple shooting game by adding enemy tanks in the game.</p> <p>With this will be winding up creating Virtual Reality applications and we will be moving on to learn how to create Augmented Reality applications in upcoming classes.</p> <p>We will understand the difference between VR and AR and we will learn how we can make web based AR applications and mobile based AR applications.</p> <p>Are you excited?</p>	<p><b>ESR:</b> Yes.</p>
	Let's get started then.	
<div>  <p><b>Teacher Ends Slideshow</b></p> </div>		
<b>TEACHER-LED ACTIVITY - 15 mins</b>		
<b>Teacher Initiates Screen Share</b>		
<p style="text-align: center;"><b><u>CHALLENGE</u></b></p> <ul style="list-style-type: none"> <li>• Learn about the finding direction between two position vectors.</li> <li>• Use Three.js methods to find the direction vector.</li> </ul>		

<p><b>Step 2:</b> <b>Teacher-led Activity</b> <b>(15 mins)</b></p>	<p><i>&lt;The teacher clones the code from Teacher Activity 1 and the output scene.&gt;</i></p> <p><u><i>[Teacher Activity 1]</i></u></p> <p>Let me open and run the activity. We can see that in the scene we have enemy tanks models which are moving back and forth.</p> <p>Can you tell me how this is done?</p> <p>Yes, that's right!</p> <p>Also, we have some text entities added, such as the child entity of the camera, which shows the player's life and the number of tanks left to shoot.</p>	<p><b>ESR:</b> Using the animation component.</p>
--	--	---

```
<!--Enemy-->
<a-entity id="enemy1" rotation="0 90 0" gltf-model="#tank" position="-10 0 -15"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -15; dur: 20000; easing: linear; loop: true; dir:alternate">
</a-entity>

<a-entity id="enemy2" rotation="0 90 0" gltf-model="#tank" position="-50 0 -40"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -40; dur: 20000; easing: linear; loop: true; dir:alternate">
</a-entity>
```

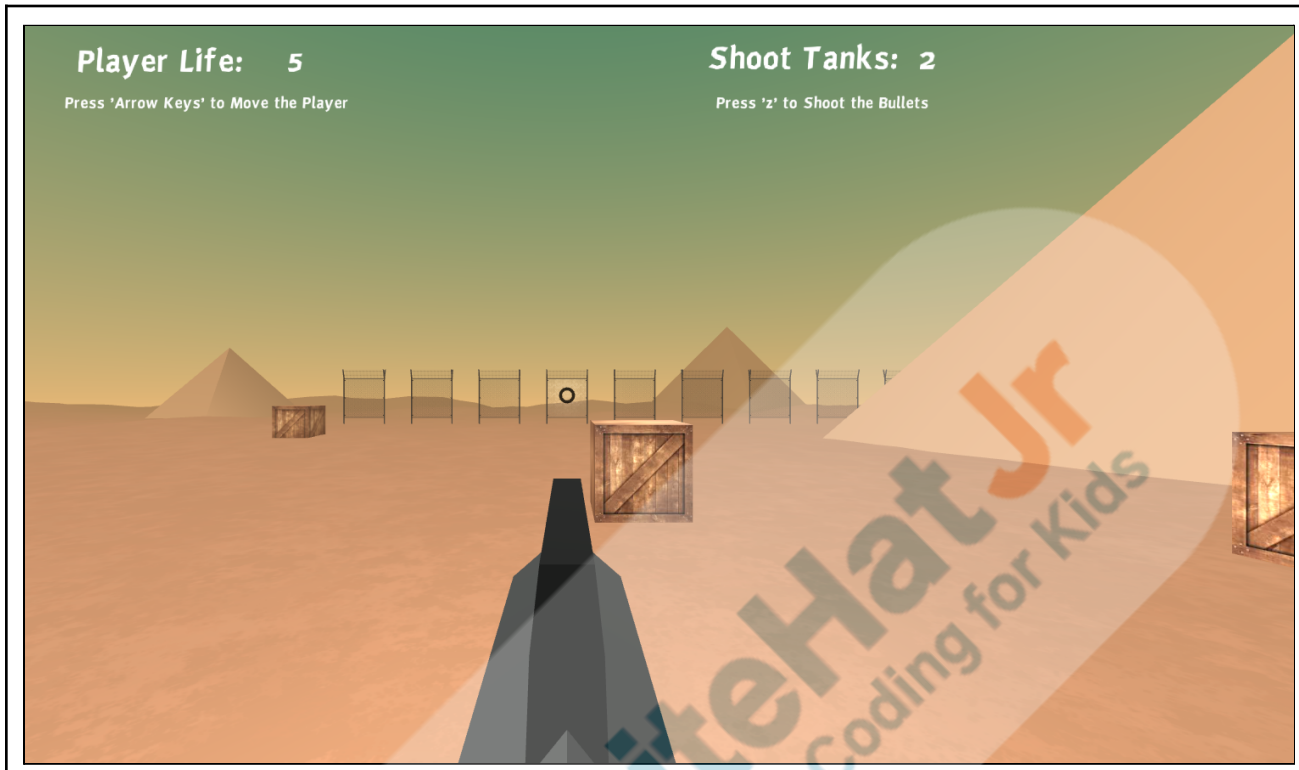
```
<!--Text-->
<a-entity id="ins1" position="-4.62 4.25669 -7.1"
  text="font: mozillavr; width:5; height: 5; value: Press 'Arrow Keys' to Move the Player">
</a-entity>
<a-entity id="in2" position="4.62 4.25669 -7.1"
  text="font: mozillavr; width:5; height: 5; value: Press 'z' to Shoot the Bullets">
</a-entity>

<a-entity id="tanktargets" position="7 5 -7.1" text="font: mozillavr;width:10; height: 5; value: Shoot Tanks:">
  <a-entity id="countTank" position="3 0 0" text="font: mozillavr; width:10; height: 5; value: 2"></a-entity>
</a-entity>

<a-entity id="playerLife" position="-2 5 -7.16344"
  text="font: mozillavr; width:10; height: 5;value: Player Life:">
  <a-entity id="countLife" position="3 0 0" text="font: mozillavr; width:10; height: 5; value: 5"></a-entity>
</a-entity>

<a-entity id="over" position="1 1 -3"
  text="font: mozillavr; width:5; height: 5; value: Better Luck Next Time :(" visible="false">
</a-entity>

<a-entity id="completed" position="1.5 1 -3"
  text="font: mozillavr; width:5; height: 5; value: Level Completed :)" visible="false">
</a-entity>
</a-entity>
```



Now we will be learning how to keep firing bullets from these two tanks. Also will be updating the player life and the number of tanks left to shoot in the game.

Before we do this, let's quickly summarise what we have done in the shooting game till now.

In a basic or simple shooting game, we have one player who shoots towards the enemy.

Can you tell me how the player shooting was achieved?

**ESR:** Yes. We wrote the component to create bullet entities. In that we:

	<p>Yes, that's amazing!</p> <p>We also learned how to navigate around the scene without colliding objects by creating a navigation mesh and using the movement-control component.</p>	<ul style="list-style-type: none"> <li>• Get the camera direction as a Three.js object.</li> <li>• Get the camera position as an A-Frame element.</li> <li>• Set the velocity of the bullet.</li> </ul> <p>We also added the collision event on the boxes and removed the bullet element from the scene after the collision.</p>
	<p>We have enemy tanks models in the scene.</p> <p>Now we should be able to fire bullets continuously from these tanks.</p> <p>How should we start?</p> <p>Great! The direction of shooting will be from the enemies towards the player.</p>	<p><b>ESR:</b> First, we need to find in which direction we want to shoot.</p>

	<p>For this, we need to find the direction vector between the enemy and the player.</p> <p>What do you think is the first requirement to know the direction between two objects?</p> <p>Yes! The first requirement is to know where the objects are, then we can find the direction from one position to the other.</p> <p>To find the direction vector, we will get the object's position, the enemy object, and the player object, as the Three.js positions.</p> <p>Can you tell me how we can do this?</p> <p>Great! Let's look at the <code>enemyShoot.js</code> file that registers the A-Frame component called <b>"enemy-bullets"</b> with a function named <code>shootEnemyBullet()</code></p> <p><i><b>Note 1:</b> The <code>enemyShoot.js</code> file is included in <code>&lt;head&gt;</code> tag</i></p> <p><i><b>Note 2:</b> The <b>"enemy-bullets"</b> component is having <code>.init()</code> method</i></p>	<p><b>ESR:</b> To know the position of the objects.</p> <p><b>ESR:</b> We will take two variables and initialize Three.js vector variables using new <b>THREE.Vector3()</b>.</p>
--	---	--



	<p>and <b>shootEnemyBullet()</b> function and is attached it to the <b>&lt;a-entity&gt;</b> in the <b>index.html</b> file.&gt;</p>	
<pre> &lt;script src="./enemyShoot.js"&gt;&lt;/script&gt;  AFRAME.registerComponent("enemy-bullets", {   init: function () {    },   shootEnemyBullet: function () {   }, });  &lt;!--Bullets--&gt; &lt;a-entity bullets&gt;&lt;/a-entity&gt; &lt;a-entity enemy-bullets&gt;&lt;/a-entity&gt; </pre>		
	<p>Since we want to shoot the bullets continuously, what should we use for that?</p> <p>Superb!</p> <p>Yes, we will be using the <b>setInterval()</b> method so that we can call the function to shoot enemy bullets continuously.</p>	<p><b>ESR:</b> We should use JavaScript timing events. We can use the <b>setInterval()</b> method.</p>

```
AFRAME.registerComponent("enemy-bullets", {
  init: function () {
    setInterval(this.shootEnemyBullet, 2000)
  },
  shootEnemyBullet: function () {
  },
});
```

What should we do next?

I want to shoot bullets from both the tanks, so shall I create the bullet entities for each tank one by one?

If we want to create bullet entities for each tank one by one, we can get each tank element's id and create an entity for each one of them.

Well, this will be a very long process.

Suppose we have 10 enemies in the scene, then creating bullet entities for each tank will be cumbersome.

To avoid that, we are going to give a class name to each of the enemy tank entities and then we can access all the entities using the class name.

*<The teacher adds the class name to enemy entities.>*

**ESR:** Create entities to shoot bullets.

**ESR :** Yes/No.

```
<!--Enemy-->
<a-entity class="enemy" id="enemy1" rotation="0 90 0" gltf-model="#tank" position="-10 0 -15"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -15; dur: 20000; easing: linear; loop: true; dir:alternate">
</a-entity>

<a-entity class="enemy" id="enemy2" rotation="0 90 0" gltf-model="#tank" position="-50 0 -40"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -40; dur: 20000; easing: linear; loop: true; dir:alternate">
</a-entity>
```

Now to access entities using class names we are going to use the **document.querySelectorAll()** method and the class name inside the brackets.

To select elements using class names we use dot instead of hash.

*<The teacher creates a variable to select an entity using class name.>*

```
shootEnemyBullet: function () {
  //get all enemies using className
  var els = document.querySelectorAll(".enemy");
},
```

After selecting the elements using class name, how many elements will be there in the **els** variable?

***Note:** The teacher can show the element selected using `console.log(els)`.*

We have the elements; now we should have to loop through all the

**ESR:** We will get 2 elements.

elements to create the bullet entity for each one of them.

*<The teacher creates an entity element and sets its geometry, position, material using **setAttribute()** and test the output.>*

```
//get all enemies using className
var els = document.querySelectorAll(".enemy");

for (var i = 0; i < els.length; i++) {

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

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



    enemyBullet.setAttribute("material", "color", "#282B29");

    var position = els[i].getAttribute("position")

    enemyBullet.setAttribute("position", {
        x: position.x + 1.5,
        y: position.y + 3.5,
        z: position.z,
    });

    var scene = document.querySelector("#scene");
    scene.appendChild(enemyBullet);
}
```



Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
<div>  <b>Teacher Starts Slideshow</b>  <b>Slide 16 to 17</b>            Refer to speaker notes and follow the instructions on each slide.         </div>		
We have one more class challenge for you. Can you solve it?  Let's try. I will guide you through it.		
<div>  <b>Teacher Ends Slideshow</b> </div>		
STUDENT-LED ACTIVITY - 20 mins		
<ul style="list-style-type: none"> <li>• Ask the student to press the ESC key to come back to the panel.</li> <li>• Guide the student to start screen share.</li> <li>• Teacher gets into fullscreen.</li> </ul>		
<p align="center"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> <li>• Create enemy bullets in the player direction.</li> <li>• Update the number of tanks left to shoot.</li> </ul>		
<b>Step 3:</b> <b>Student-Led Activity</b> <b>(20 mins)</b>	<p><i>The teacher guides the student to clone the code from Student Activity 1.</i></p> <p><a href="#">[Student Activity 1]</a></p> <p><i>The following code is a boilerplate template for the student to start the activity.</i></p>	

```
//Three.js Vector Variables

//Get enemy and player position using Three.js methods

//set the velocity and it's direction

//Set dynamic-body attribute

//Get text attribute

//collide event on enemy bullets
enemyBullet.addEventListener("collide", function (e) {
    if (e.detail.body.el.id === "weapon") {

        //Add the conditions here

    }
});
```

Next we need to set up the shooting direction.

We are going to use the **Three.js** method to find the direction vector.

For this first we will be getting the position as a Three.js object.

*Guide the student to create 2 variables **enemy** and **player** and get position as a Three.js object.*

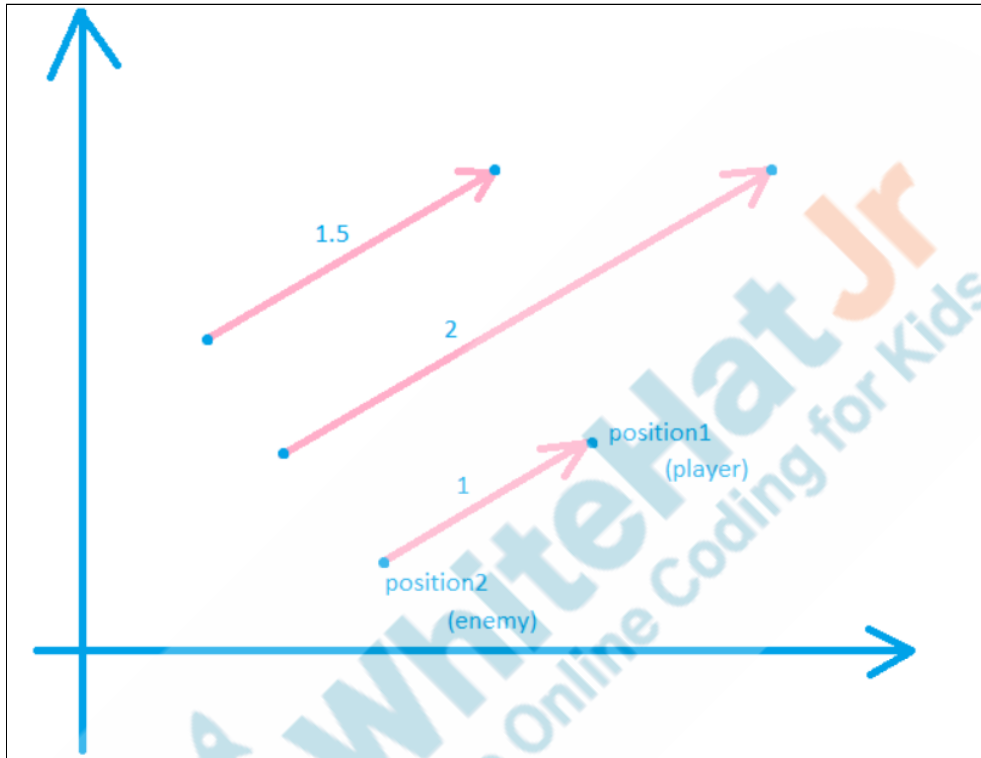
***Note:** Help the student to recollect the we use **.object3D** after*

	<i><b>.querySelector()</b> method to access the A-Frame entity as Three.js object</i>	
<pre>var enemy = els[i].object3D; var player = document.querySelector("#weapon").object3D;</pre>		
	<p>We need two <b>THREE.Vector3()</b> variables, <b>position1</b> and <b>position2</b> in which we can store the position of the enemy object and the player object.</p> <p>We can use the <b>getWorldPosition()</b> method of the <b>Three.js</b> library to store the value of player position and enemy position as vectors.</p>	
<pre>var position1 = new THREE.Vector3(); var position2 = new THREE.Vector3();  player.getWorldPosition(position1); enemy.getWorldPosition(position2);</pre>		
	<p>Now to get the direction vector from any two players, we need to use the <b>subVectors()</b> method which gives the result after <b>subtracting two vectors</b>.</p> <p>Before we can use the <b>subVectors()</b> method, let's first understand the concept of finding direction vectors.</p> <p><i>&lt;The teacher opens the <b>Visual Aids</b> to explain the concept of <b>vectors</b>, <b>standard(unit) vectors</b> and <b>different between two vectors</b>.&gt;</i></p>	



	<p><b>Vectors</b> are mathematical objects which have <b>magnitude</b> and <b>direction</b>.</p> <p>The <b>magnitude</b> value tells how long the length will be starting from one point to another.</p> <p>The <b>direction</b> tells in which direction the vector will be starting from the origin that is (0 0 0).</p> <p>Also, to find the <b>direction between two points</b>, say point 1 and point 2, we will have to <b>subtract the position vectors of the points in the reverse order</b>.</p> <p>In the image you can see that we have an enemy and the player at position 2 and 1.</p> <p>To get the direction from enemy to player we need to find the difference between position 1 and 2.</p> <p>You can also see that there are different lengths of the vectors in the same direction.</p> <p>The length of <b>1 unit is the standard length of the vectors</b> which is known as the <b>unit vector</b>.</p> <p>If you multiply 1.5 to the same vector, we will get the topmost vector in the</p>	
--	---	--

same direction. Similarly, if you multiply two unique vectors, we will get a second vector.



Okay, now let's use the **Three.js** method to find the direction vector.

*Guide the student to take a **direction** variable and use a **subVectors()** method to find the vector between **position1** and **position2**.*

We use **.normalize()** to get the unit vector that is the direction vector of length 1.

Once we get the standard vector we can multiply it with any number to increase the velocity in that direction.

*Guide the student to set velocity using direction and **multiplyScalar()** method and test output.*

```
//set the velocity and it's direction
var direction = new THREE.Vector3();

direction.subVectors(position1, position2).normalize();

enemyBullet.setAttribute("velocity", direction.multiplyScalar(10));
```



We have the bullet coming out of the enemy tanks.

Now we should be able to detect the collision between the bullet and the player to update the player's life.

For this:

- Set **dynamic-body** attribute of the enemy bullet entity.
- Select the **countLife** text to update the player life count.
- Add the **collide** event listener to check if the weapon element has been hit.
- Decrease the player's life and update the text attribute.

#### enemyShoot.js

```
enemyBullet.setAttribute("dynamic-body", {  
  shape: "sphere",  
  mass: "0",  
});
```

```
var element = document.querySelector("#countLife");  
var playerLife = parseInt(element.getAttribute("text").value);
```

```
//collide event on enemy bullets
enemyBullet.addEventListener("collide", function (e) {
  if (e.detail.body.el.id === "weapon") {

    if (playerLife > 0) {
      playerLife -= 1;
      element.setAttribute("text", {
        value: playerLife
      });
    }
  }
});
```

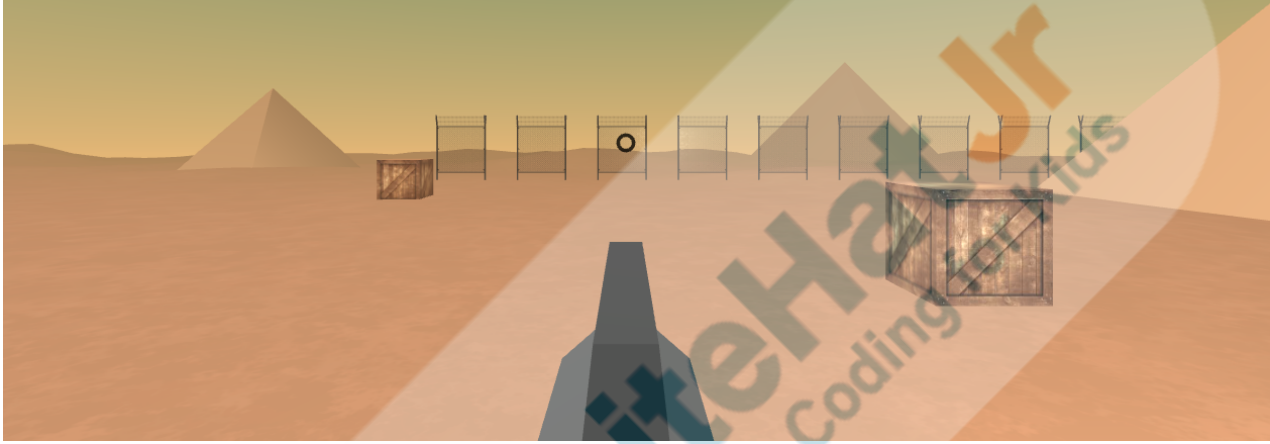


To make sure the collision is being detected by the weapon element:

- Update the **weapon** entity bounding shape to set the body and the shape component manually.

#### index.html

```
<a-entity id="weapon" gltf-model="#shooter"
  position="0 -4.4 3" rotation="0 180 0" scale="0.35 1 1"
  body="type: static; mass: 5; shape: none;"
  shape="shape: sphere; radius: 5; offset: 0 3 0;"
  player-movement>
</a-entity>
```

		
	Once the player life is zero, we can show the over text and remove all the tank elements from the scene.	
<pre>if (playerLife &lt;= 0) {   //show text   var txt = document.querySelector("#over");   txt.setAttribute("visible", true);    //remove tanks   var tankEl = document.querySelectorAll(".enemy")    for (var i = 0; i &lt; tankEl.length; i++) {     scene.removeChild(tankEl[i])   } }</pre>		

<div> <div> <b>Player Life: 0</b>  <small>Press 'Arrow Keys' to Move the Player</small> </div> <div> <b>Shoot Tanks: 2</b>  <small>Press 'z' to Shoot the Bullets</small> </div> </div> <div> <p><b>Better Luck Next Time :(</b></p>  </div>	
<b>Teacher Guides Student to Stop Screen Share</b>	
<b>WRAP UP SESSION - 5 mins</b>	
<b>Teacher Starts Slideshow</b> Slide 18 to 23	
<b>Activity details</b> <b>Following are the WRAP-UP session deliverables:</b> <ul style="list-style-type: none"> <li>• Appreciate the student.</li> <li>• Revise the current class activities.</li> <li>• Discuss the quizzes.</li> </ul>	
<b>WRAP-UP QUIZ</b> Click on In-Class Quiz	
<b>Continue WRAP-UP Session</b>	

## Slide 24 to 29

### Activity Details

#### Following are the session deliverables:

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

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:*



### 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.

*The student uses the markdown editor to write their reflections in a reflection journal.*



	<ul style="list-style-type: none"> <li>• How did I feel after the class?</li> <li>• What have I learned about programming and developing games?</li> <li>• What aspects of the class helped me? What did I find difficult?</li> </ul>	
--	---	--

Activity	Activity Name	Links
Teacher Activity 1	Teacher Activity 1	<a href="https://github.com/whitehatjr/PRO-C165-Teacher-Activity">https://github.com/whitehatjr/PRO-C165-Teacher-Activity</a>
Teacher Activity 2	Teacher Reference Code	<a href="https://github.com/whitehatjr/PRO-C165-Teacher-Ref">https://github.com/whitehatjr/PRO-C165-Teacher-Ref</a>
Teacher Activity 3	Output Reference	<a href="https://drive.google.com/file/d/12duPNgakjVO8r5BzorCON44g0jKaeOfq/view?usp=sharing">https://drive.google.com/file/d/12duPNgakjVO8r5BzorCON44g0jKaeOfq/view?usp=sharing</a>
Student Activity 1	Boilerplate Code	<a href="https://github.com/whitehatjr/PRO-C165-Student-Activity">https://github.com/whitehatjr/PRO-C165-Student-Activity</a>
Teacher Reference 1	Project Document	<a href="https://s3-whjr-curriculum-uploads.whjr.online/2bc1af9a-0cf9-4b5a-ba63-34a9141970f2.pdf">https://s3-whjr-curriculum-uploads.whjr.online/2bc1af9a-0cf9-4b5a-ba63-34a9141970f2.pdf</a>
Teacher Reference 2	Project Solution	<a href="https://github.com/whitehatjr/PRO-C165-Project-Solution">https://github.com/whitehatjr/PRO-C165-Project-Solution</a>
Teacher Reference 3	Visual-Aid	<a href="https://s3-whjr-curriculum-uploads.whjr.online/1c347ac3-86b8-4511-9af5-fb7c33d32a18.html">https://s3-whjr-curriculum-uploads.whjr.online/1c347ac3-86b8-4511-9af5-fb7c33d32a18.html</a>
Teacher Reference 4	In-Class Quiz	<a href="https://s3-whjr-curriculum-uploads.whjr.online/18be25e5-a0f2-4a7e-bda7-7f1281015091.pdf">https://s3-whjr-curriculum-uploads.whjr.online/18be25e5-a0f2-4a7e-bda7-7f1281015091.pdf</a>