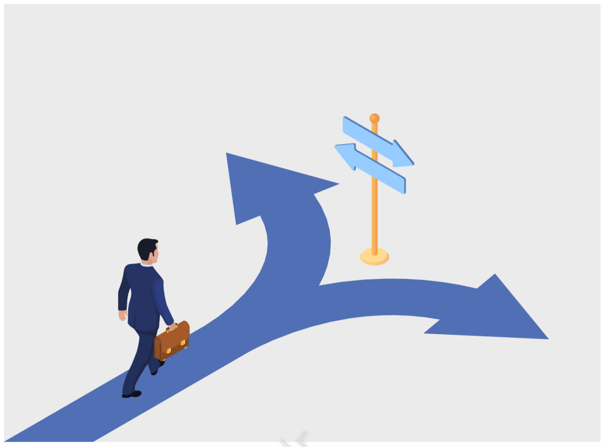
# VECTOR DIRECTION



## What is our GOAL for this MODULE?

We learned to find the direction between to position vectors of two elements in the A-Frame scene using Three.js methods.

## What did we ACHIEVE in the class TODAY?

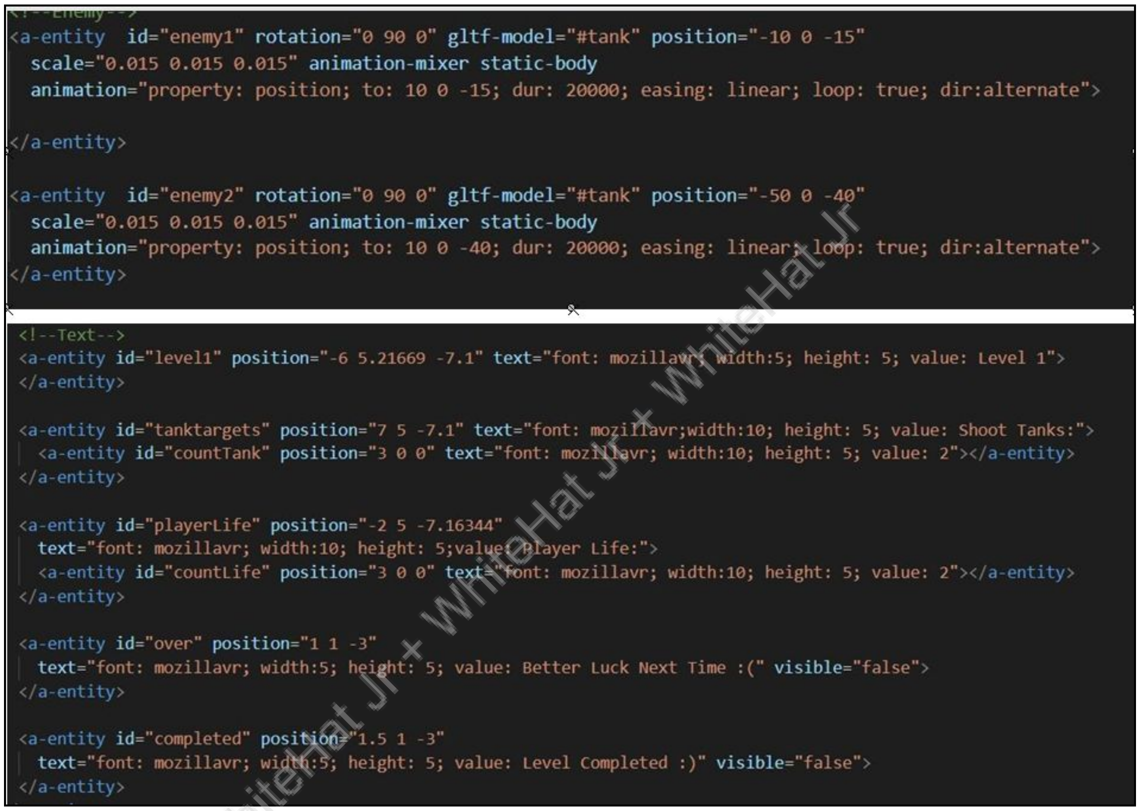
- Learned to find the direction vector between two position vectors.
- Learned to use Three.js method to find the direction vector.

## Which CONCEPTS/CODING BLOCKS did we cover today?

- .subVectors(),.normalize() methods
- THREE.Vector3(), .getWorldDirection(vectorVariable)
- document.queryselectorAll() , .addEventListener(), setAttribute(), getAttribute(), .registerComponent() methods

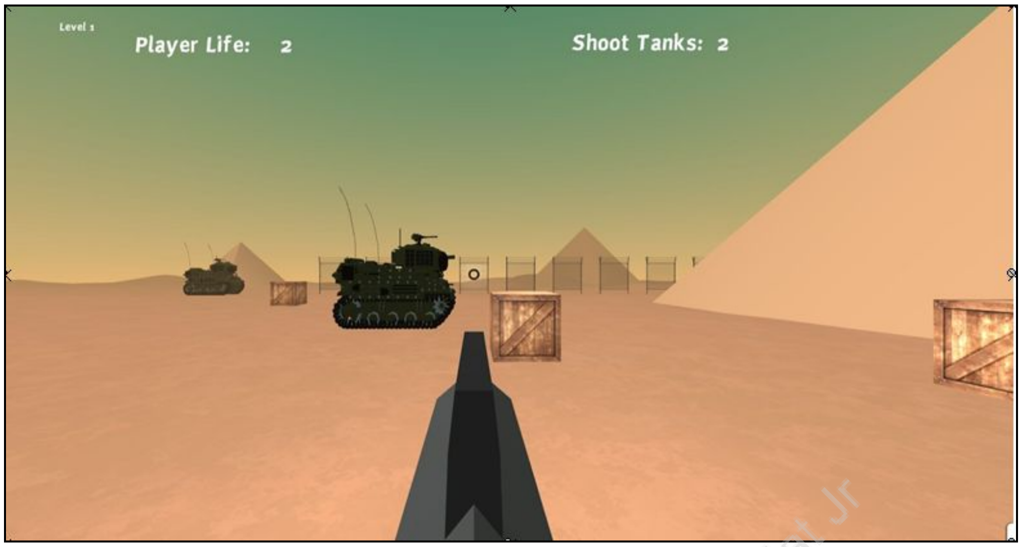
## How did we DO the activities?

1. Create the aframe scene which shows the player's life and the number of tanks left to shoot.

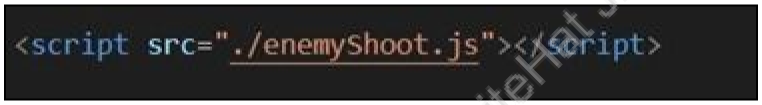


```
<!--Enemy-->
<a-entity  id="enemy1" rotation="0 90 0" gltf-model="#tank" position="-10 0 -15"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -15; dur: 20000; easing: linear; loop: true; dir:alternate">

</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>
```

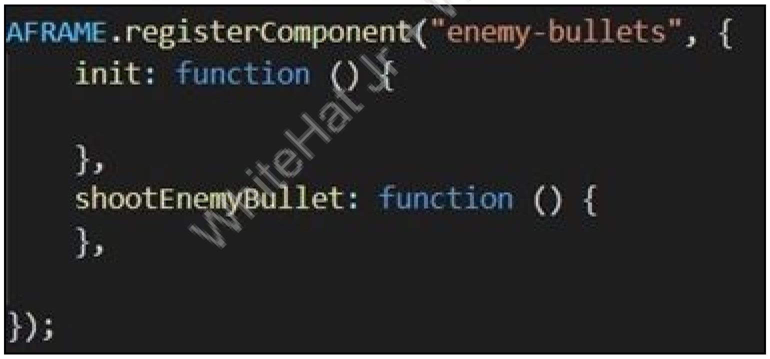
```
<!--Text-->
<a-entity id="level1" position="-6 5.21669 -7.1" text="font: mozillavr; width:5; height: 5; value: Level 1">
</a-entity>

<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: 2"></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>
```



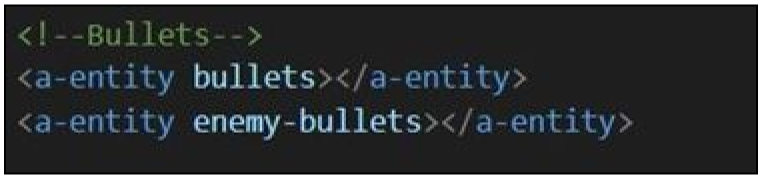
2. Register the “enemy-bullets” component having **init()** method and **shootEnemyBullet()** functions and attach it to the <a-entity> in the index.html file.



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

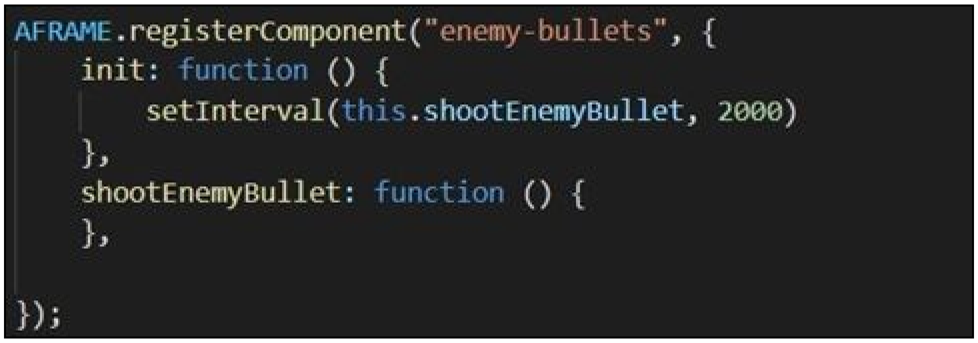


```
AFRAME.registerComponent("enemy-bullets", {
    init: function () {

    },
    shootEnemyBullet: function () {
    },

});
```



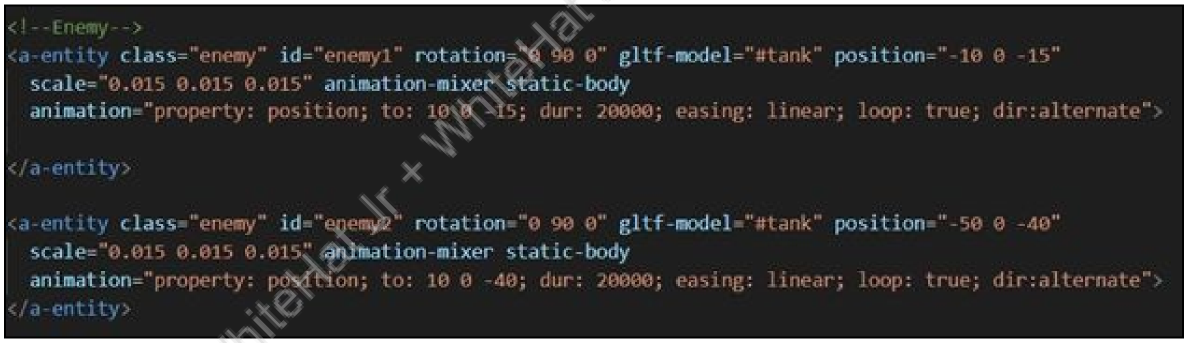
```
<!--Bullets-->
<a-entity bullets></a-entity>
<a-entity enemy-bullets></a-entity>
```

3. Use the **setInterval()** method to call the function to shoot enemy bullets continuously.



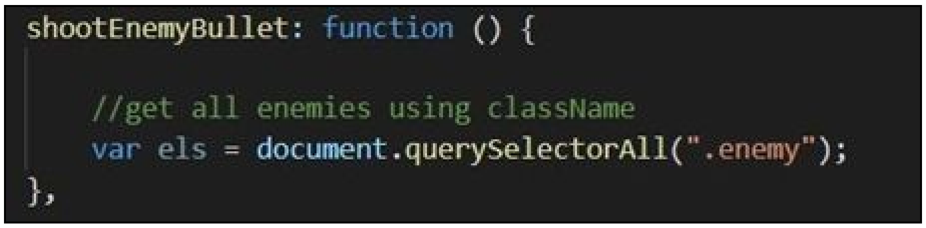
```
AFRAME.registerComponent("enemy-bullets", {
    init: function () {
        setInterval(this.shootEnemyBullet, 2000)
    },
    shootEnemyBullet: function () {
    },

});
```

4. Add the class name to enemy entities.



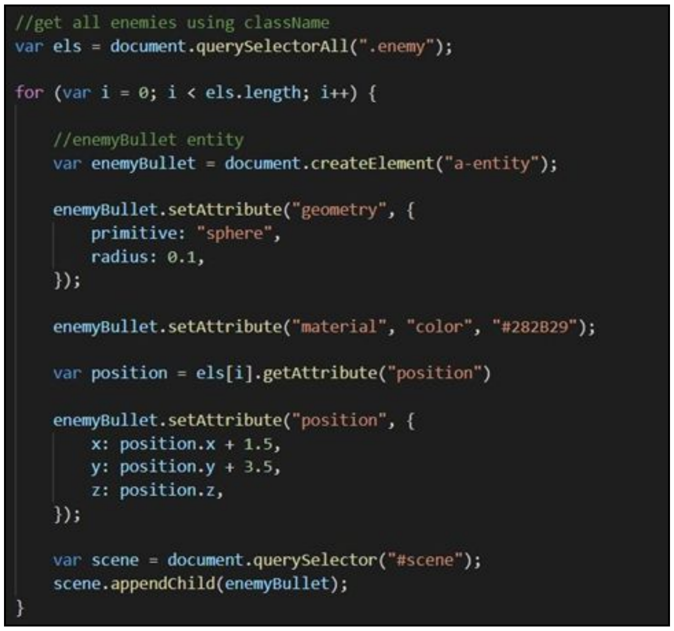
```
<!--Enemy-->
<a-entity class="enemy" id="enemy1" rotation="0 90 0" gltf-model="#tank" position="-10 0 -15"
  scale="0.015 0.015 0.015" animation-mixer static-body
  animation="property: position; to: 10 0 -15; dur: 20000; easing: linear; loop: true; dir:alternate">

</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>
```

5. Create a variable to select an entity using class name.



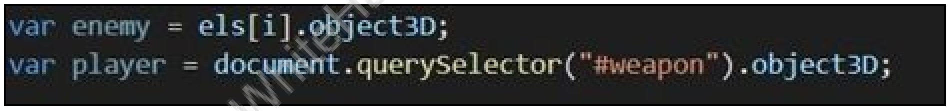
```
shootEnemyBullet: function () {

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

6. Create an entity element and set its geometry, position, material using **setAttribute()**.



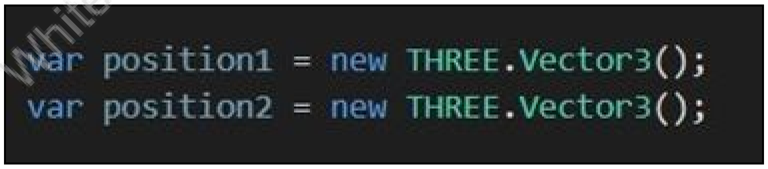
```
//get all enemies using className
var els = document.querySelectorAll(".enemy");

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);
}
```

7. To get the position vectors:
    - Add 2 variables and get the enemy and player position as a Three.js object.



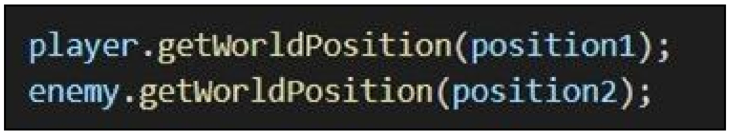
```
var enemy = els[i].object3D;
var player = document.querySelector("#weapon").object3D;
```

    - Create two **THREE.Vector3()** variables in which we can store the position of enemy object and the player object.



```
var position1 = new THREE.Vector3();
var position2 = new THREE.Vector3();
```

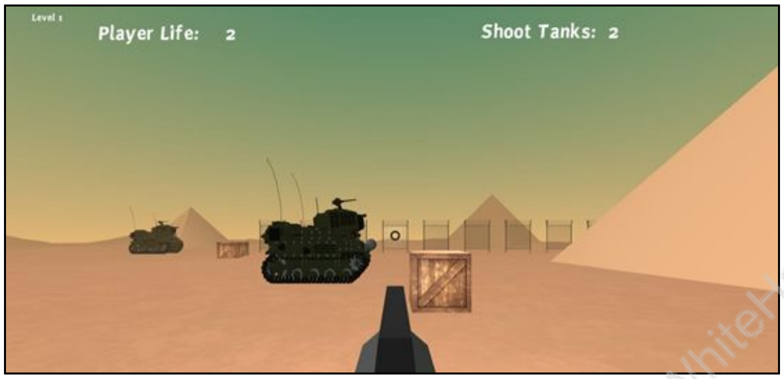
    - Use the **getWorldPosition()** method of the three.js library to store the value of player position and enemy position as vectors.



```
player.getWorldPosition(position1);
enemy.getWorldPosition(position2);
```

8. Set velocity using direction and **multiplyScalar()** method and show output.

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

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

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

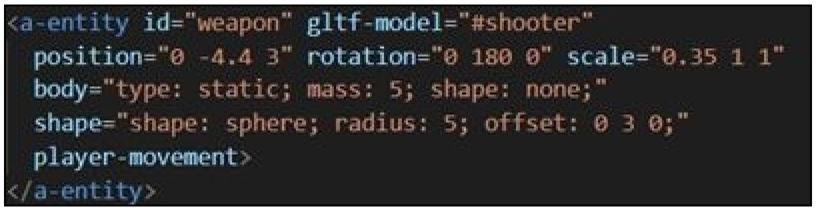


9. Detect the collision between the bullet and the player to update the player’s life.

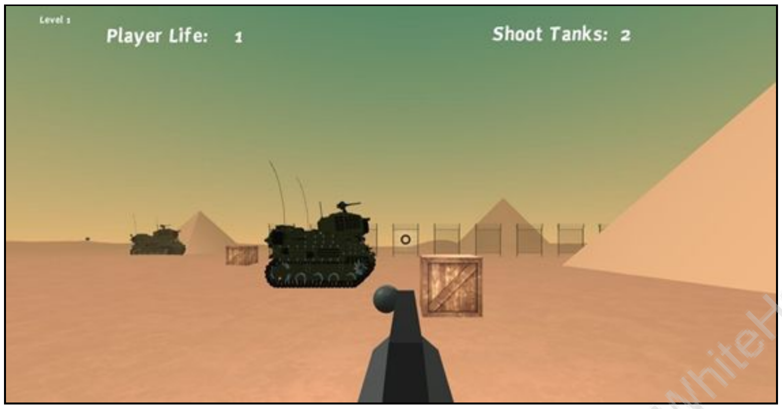
```
enemyBullet.setAttribute("dynamic-body", {
    shape: "sphere",
    mass: "0",
});
```

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

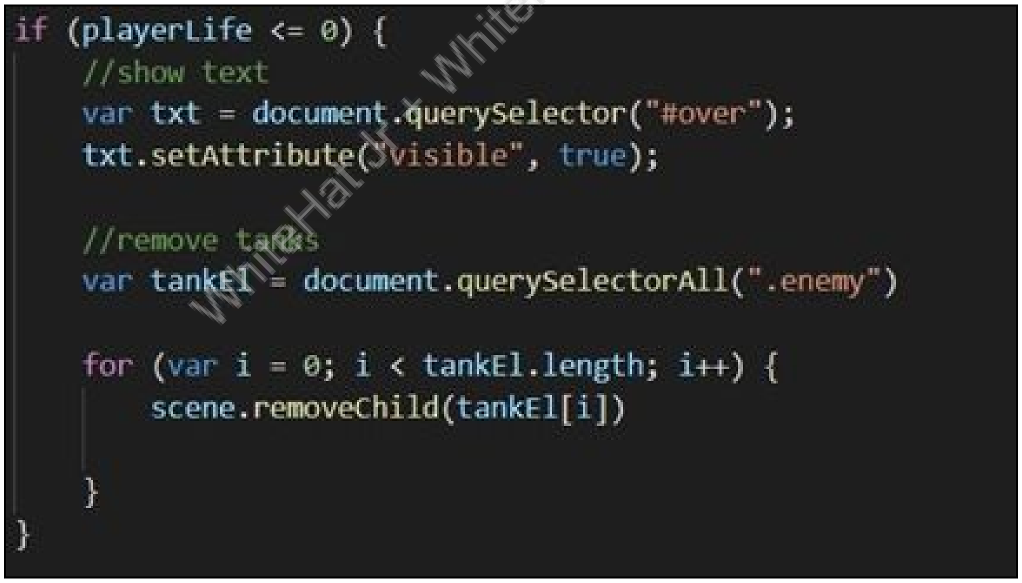
```
//collide event on enemy bullets
enemyBullet.addEventListener("collide", function (e) {
    if (e.detail.body.el.id === "weapon") {

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

    }
});
```



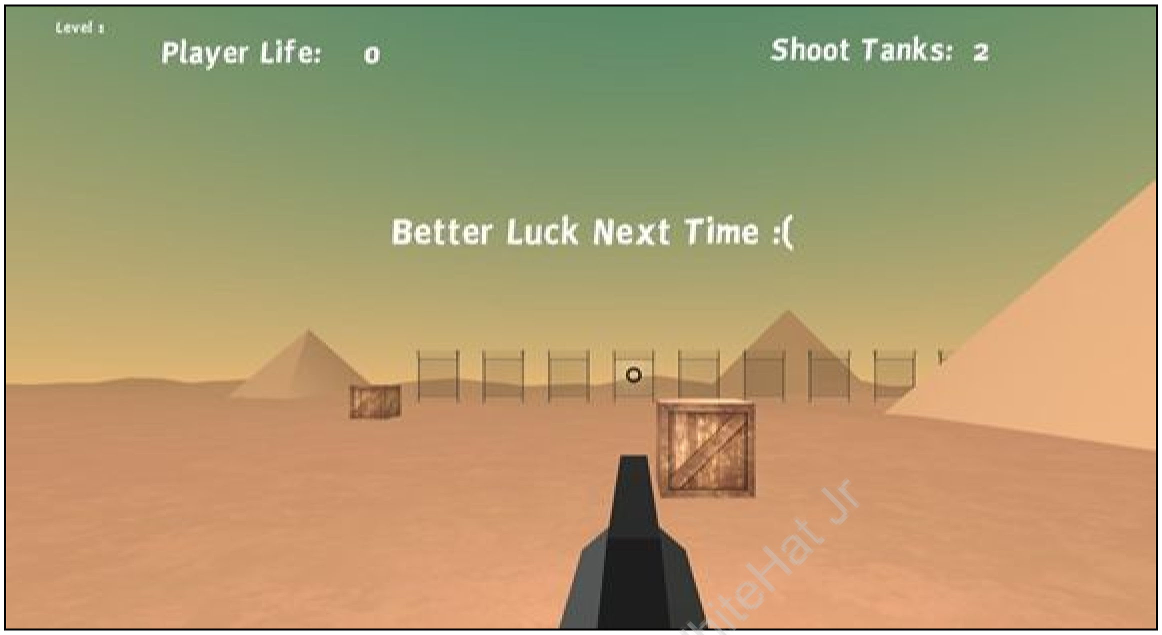
```
<a-entity id="weapon" gltf-model="#shooter"
  position="0 -4.4 3" rotation="0 180 0" scale="0.35 1 1"
  body="type: static; mass: 5; shape: none;"
  shape="shape: sphere; radius: 5; offset: 0 3 0;"
  player-movement>
</a-entity>
```



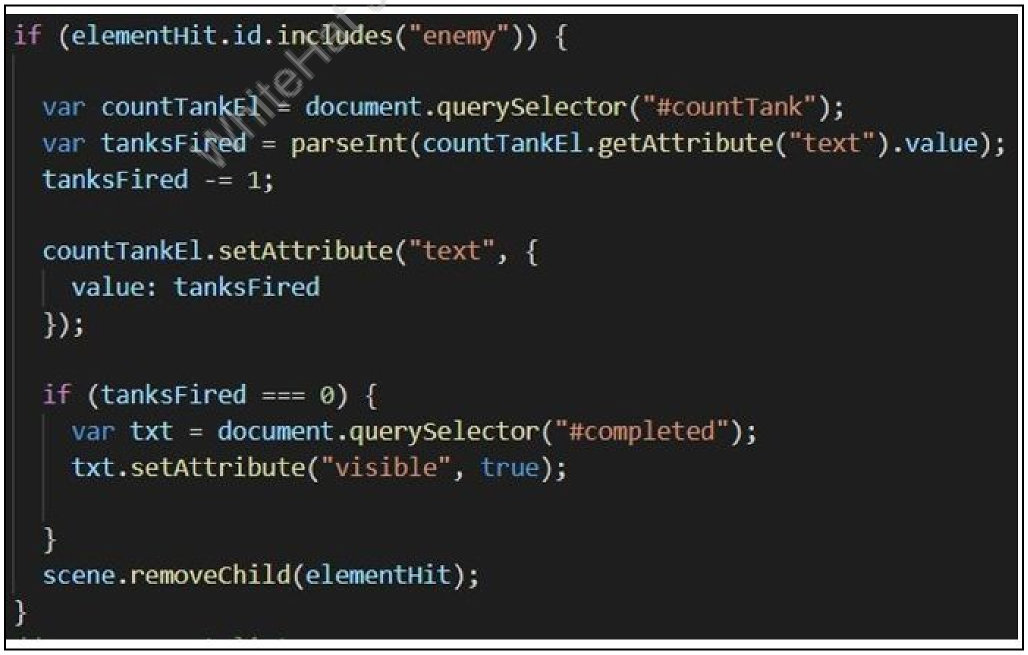
10. Once the player's life is zero, show the game over text and remove all the tank elements from the scene.



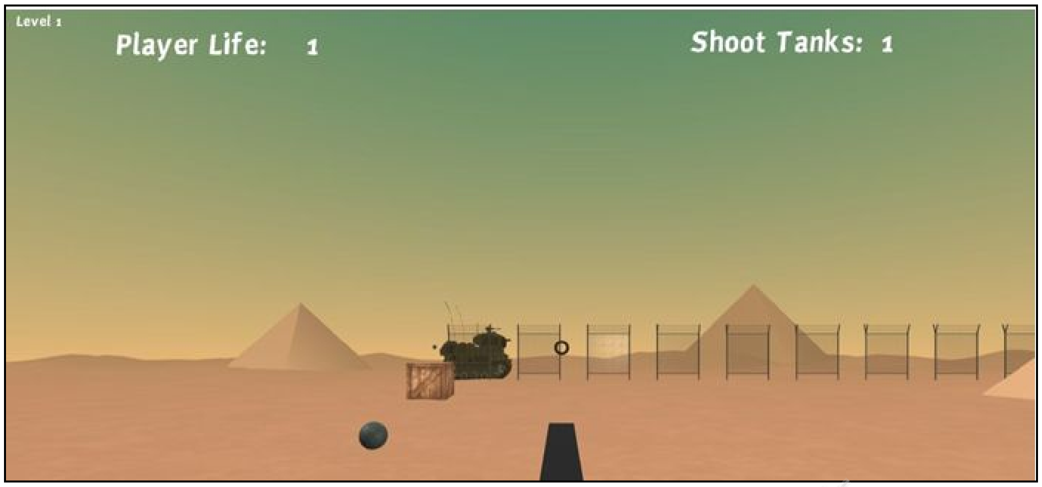
```
if (playerLife <= 0) {
    //show text
    var txt = document.querySelector("#over");
    txt.setAttribute("visible", true);

    //remove tanks
    var tankEl = document.querySelectorAll(".enemy")

    for (var i = 0; i < tankEl.length; i++) {
        scene.removeChild(tankEl[i])

    }
}
```

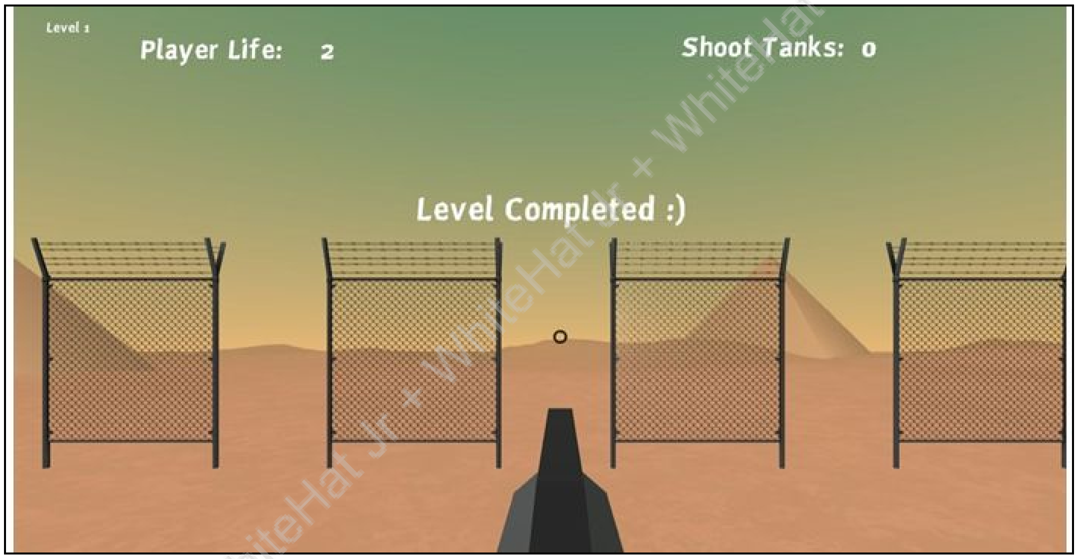


11. Update the tanks left to shoot to complete the level.



```
if (elementHit.id.includes("enemy")) {

  var countTankEl = document.querySelector("#countTank");
  var tanksFired = parseInt(countTankEl.getAttribute("text").value);
  tanksFired -= 1;

  countTankEl.setAttribute("text", {
    value: tanksFired
  });

  if (tanksFired === 0) {
    var txt = document.querySelector("#completed");
    txt.setAttribute("visible", true);

  }
  scene.removeChild(elementHit);
}
```





We have successfully learned to find direction vectors using Three.js method.

**What's NEXT?**

In the next class, we will learn the basics of Augmented Reality Web Apps.

**EXTEND YOUR KNOWLEDGE:**

- You can refer to the link below to explore more about A-Frame: A-Frame.
- You can refer to the link below to explore more about Three.js: Three.js Object3D.