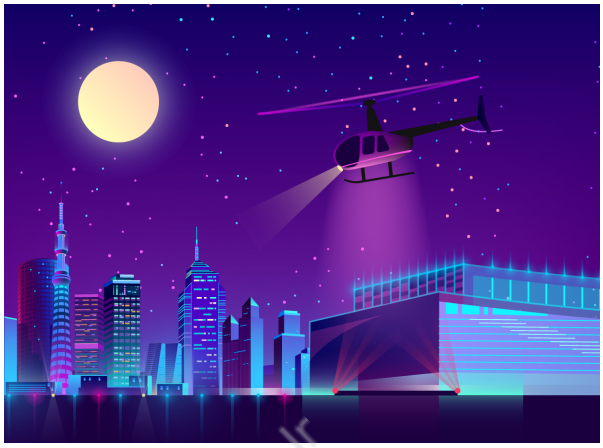
# A-Frame PHYSICS SYSTEM



## What is our GOAL for this MODULE?

The goal of this module is to learn to use physics in A-Frame VR and also learn about the static and dynamic body of the physics system.

## What did we ACHIEVE in the class TODAY?

- We learned how to add physics in A-Frame VR.
- We learned to make bodies static and dynamic.
- We learned to control the physics system bodies in A-Frame VR.

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

- Used https://cdn.jsdelivr.net/gh/n5ro/aframe - physics-system@v4.0.1/dist/aframe-p hysics-system.min.js library.
- Used addEventListener(), querySelector(elementId), registerComponent().
- Used the setAttribute() method.

## How did we DO the activities?

1. We learned how to add the physics system library

```
<head>
  <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>

</head>
```

2. We learned how to make entities as static-body and dynamic-body.

```
//set the static body attribute of physics system
ringEl.setAttribute("static-body", {
  shape: "sphere",
  sphereRadius: 2
});
```

```
//set the static body of the physic system
birdEl.setAttribute("static-body", {
  shape: "sphere",
  sphereRadius: 5
})
```

```
<!-- Plane -->
<a-entity
  id="plane_model"
  gltf-model="#plane"
  position="0 0 15"
  scale="1 1 1"
  rotation="0 90 0"
  plane-rotation-reader
  dynamic-body
>
</a-entity>
```

3. We learned how to keep the flight from falling because of gravity.

```
<!-- Plane -->
<a-entity
  id="plane_model"
  gltf-model="#plane"
  position="0 0 15"
  scale="1 1 1"
  rotation="0 90 0"
  plane-rotation-reader
  dynamic-body="mass: 0"
>
</a-entity>
```

4. We learned how to register a component 'game-play' and add it to the plane entity.

```
AFRAME.registerComponent("game-play", {
    schema: {
      elementId: { type: "string", default: "#ring1" }
    },


    isCollided: function(elementId) {

    }

  });
```

```
<!-- Plane -->
<a-entity
  id="plane_model"
  gltf-model="#plane"
  position="0 0 15"
  scale="1 1 1"
  rotation="0 90 0"
  plane-rotation-reader
  dynamic-body="mass: 0"
  game-play
>
</a-entity>
```

5. We learned how to detect the collision.

```
isCollided: function(elementId) {

  const element = document.querySelector(elementId);

  element.addEventListener("collide", e => {

    if (elementId.includes("#ring")) {

      console.log(elementId+" collision");

    }
    else if(elementId.includes("#hurdle")){

      console.log(elementId+" collision");
    }

  });
}
```

```
update: function() {
  this.isCollided(this.data.elementId);
},
```

```
ringEl.setAttribute("game-play", {
  elementId: `#${id}`
});
```

```
birdEl.setAttribute("game-play", {
  elementId: `#${id}`
});
```


We have successfully learned about collision event-handling in A-Frame.

### What's NEXT?

In the next class, we will learn about JavaScript timing events to show the timer clock in the virtual simulation scene.

### EXTEND YOUR KNOWLEDGE:

You can refer to the below link to explore more about AFrame
https://aframe.io/docs/1.1.0/introduction