AR NAVIGATION

What is our GOAL for this MODULE?

We learned about location-based augmented reality and also learned to use location coordinates to create augmented reality navigation paths from the source to the destination using Mapbox GL JS library.

What did we ACHIEVE in the class TODAY?

- We learned about location-based augmented reality.
- We learned how to render augmented navigation paths using the Mapbox GL JS library.

Which CONCEPTS/CODING BLOCKS did we cover today?

- append()
- jQuery ajax()
- gps-entity-place
- gps-camera
- ngrok

How did we DO the activities?

1. Add the libraries and initialize the arjs in the scene.

```
<script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>

<script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
```

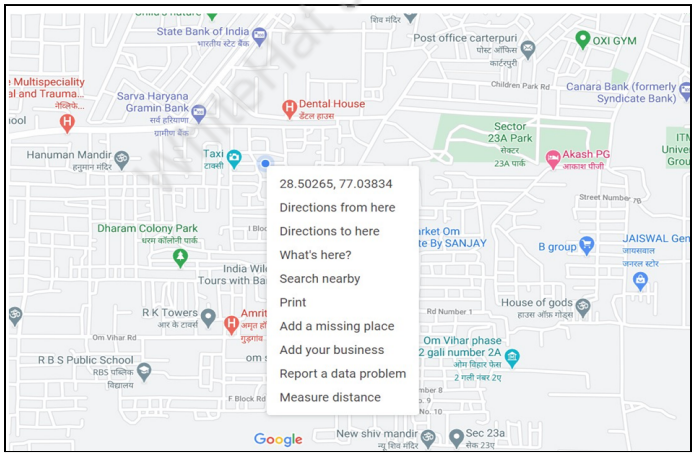
```
<a-scene vr-mode-ui="enabled: false" embedded arjs="sourceType: webcam; debugUIEnabled: false;">


</a-scene>
```

2. Use the <a-entity> tag to add the 3D model to the scene and set rotation, position and scale attributes to set the orientation and size of the model.

```
<a-assets>
  <a-asset-item id="ballModel" src="./ball/scene.gltf">
  </a-asset-item>
</a-assets>
```

```
<a-entity gltf-model="#ballModel"  scale="1 1 1" position="0 0 0" rotation="0 0 -10" ></a-entity>
```

3. Take the values of latitude and longitude from Google Maps.



4. After we have the location coordinates (latitude and longitude), place the entity in A-Frame using gps-entity-place component.

```
<a-entity gltf-model="#ballModel"  scale="1 1 1" position="0 0 0" rotation="0 0 -10"
gps-entity-place="latitude: 22.7868542 ; longitude: 88.3643296;"></a-entity>
```

5. To enable location AR, we need to set the <a-camera> as a GPS camera using the gps-camera and give the value of the look-at component as gps-camera so that it always faces the camera.

```
<a-entity gltf-model="#ballModel" look-at="[gps-camera]" scale="1 1 1" position="0 0 0" rotation="0 0 -10"
gps-entity-place="latitude: 22.7868542 ; longitude: 88.3643296;"></a-entity>
```

6. We can test the output using ngrok.

7. Write a function render_elements(), to show the arrow images on the path to show the direction to take turns and call the function in the dollar function.

```
$(document).ready(function () {
    get_coordinates();
    render_elements();
})


function render_elements() {

}
```

8. jQuery ajax() method to call the Mapbox Directions API using the access token.

```
$.ajax({
    url: `https://api.mapbox.com/directions/v5/mapbox/driving/${coordinates.source_lon}%2C${coordinates.source_lat}%3B${coordinates.destination_lo
    type: "get",
    success: function(){

    }
})
```

9. Get data from the response in the success function.

```
success: function (response) {

    let steps = response.routes[0].legs[0].steps

    for (let i = 0; i < steps.length; i++) {
        let image;
        let distance = steps[i].distance
        let instruction = steps[i].maneuver.instruction

        if (instruction.includes("Turn right")) {
            image = "turn_right"
        } else if (instruction.includes("Turn left")) {
            image = "turn_left"
        }


    }
}
```

10. Define images variables in JSON format.

```
success: function (response) {
    let images = {
        "turn_right": "ar_right.png",
        "turn_left": "ar_left.png",
        "slight_right": "ar_slight_right.png",
        "slight_left": "ar_slight_left.png",
        "straight": "ar_straight.png"
    }
    let steps = response.routes[0].legs[0].steps
```

11. Write the append() method:
    - Select the <a-scene> element using jQuery selector.
    - Use append() method to set the <a-entity> and <a-image>.

```
if (i > 0) {
    $("#scene_container").append(
        `
            <a-entity gps-entity-place="latitude: ${steps[i].maneuver.location[1]}; longitude: ${steps[i].maneuver.location[0]};">
                <a-image
                    name="${instruction}"
                    src="./assets/${images[image]}"
                    look-at="#step_${i - 1}"
                    scale="5 5 5"
                    id="step_${i}"
                    position="0 0 0"
                >
                </a-image>
                <a-entity>
                    <a-text height="50" value="${instruction} (${distance}m)"></a-text>
                </a-entity>
            </a-entity>
        `
    )
} else {
    $("#scene_container").append(
        `
            <a-entity gps-entity-place="latitude: ${steps[i].maneuver.location[1]}; longitude: ${steps[i].maneuver.location[0]};">
                <a-image
                    name="${instruction}"
                    src="./assets/ar_start.png"
                    look-at="#step_${i + 1}"
                    scale="5 5 5"
                    id="step_${i}"
                    position="0 0 0"
                >
                </a-image>
                <a-entity>
                    <a-text height="50" value="${instruction} (${distance}m)"></a-text>
                </a-entity>
            </a-entity>
        `
    )
}
```

12. Host the application on GitHub to test the output later on after the class.

We have successfully learned to add GeolocateControl and MapDirections control to the maps.

What's NEXT?

In the next class, we will learn about face detection.

EXTEND YOUR KNOWLEDGE:

- You can refer to the link below to explore more about AFrame
  A-Frame
- You can refer to the link below to explore more about jQuery
  jQuery
- You can refer to the link below to explore more about Mapbox API
  Mapbox API
- You can refer to the link below to explore more about Mapbox Directions API