ICCS404: Week10 - MindAR

Part 1: Basic Image Tracking (45 minutes)

In this lab, you will learn how to create a basic augmented reality web application using MindAR and A-Frame. You'll implement image tracking and add 3D models with animation.

Step 1: Understanding Image-based AR

Visit the MindAR Quick Start guide: https://hiukim.github.io/mind-ar-js-doc/quick-start/overview

Follow each section to understand:

- How to compile target images
- Basic page structure
- Running an AR application
- Working with 3D assets

Step 2: Project Setup

Create your project folder structure:

```
Your-Project/
├── Assets/
│   ├── target-mindar.mind    # Your compiled target file
│   ├── card-mindar.png       # Your original image
│   └── model-mindar.glb      # 3D model (if using local model)
└── aframe_mindar_part1.html
```

Step 3: Target Image Compilation

1. Go to the MindAR Image Compiler: https://hiukim.github.io/mind-ar-js-doc/tools/compile
2. Choose a target image:
    - Use a high-contrast image
    - Avoid repetitive patterns
    - Use the sample image from quick start for testing
3. Compile steps:
    - Upload your image
    - Click "Start Compile"
    - Download the `.mind` file
    - Save it as `target-mindar.mind` in your Assets folder

Step 4: Basic Application Setup

Create `aframe_mindar_part1.html` with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <script src="https://aframe.io/releases/1.6.0/aframe.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/mind-ar@1.2.5/dist/mindar-image-aframe.prod.j
s"></script>
  </head>
  <body>
    <a-scene
      mindar-image="imageTargetSrc: ./Assets/target-mindar.mind;"
      color-space="sRGB"
```

```
      renderer="colorManagement: true, physicallyCorrectLights"
      vr-mode-ui="enabled: false"
      device-orientation-permission-ui="enabled: false">

      <a-assets>
        <img id="card" src="./Assets/card-mindar.png" />
        <a-asset-item id="avatarModel" src="./Assets/model-mindar.glb">
        </a-asset-item>
      </a-assets>

      <a-camera position="0 0 0" look-controls="enabled: false"></a-camera>

      <a-entity mindar-image-target="targetIndex: 0">
        <a-plane src="#card" position="0 0 0" height="0.552" width="1" rotation="0 0 0"></a
-plane>
        <a-gltf-model
          rotation="0 0 0"
          position="0 0 0.1"
          scale="0.005 0.005 0.005"
          src="#avatarModel"
          animation="property: position; to: 0 0.1 0.1; dur: 1000; easing: easeInOutQuad; l
oop: true; dir: alternate">
        </a-gltf-model>
      </a-entity>
    </a-scene>
  </body>
</html>
```

Step 5: Code Explanation

Key elements of the code:

1. Scene Configuration:

```
<a-scene mindar-image="imageTargetSrc: ./Assets/target-mindar.mind;">
```

- Sets up the AR environment
- Specifies the target image file
- Configures rendering options

2. Asset Management:

```
<a-assets>
  <img id="card" src="./Assets/card-mindar.png" />
  <a-asset-item id="avatarModel" src="./Assets/model-mindar.glb">
  </a-asset-item>
</a-assets>
```

- Preloads all media assets
- Assigns IDs for reference
- Improves performance

3. 3D Model Setup:

```
<a-gltf-model
  position="0 0 0.1"     <!-- 0.1 units above the marker -->
  scale="0.005 0.005 0.005"   <!-- Scaled down to size -->
  rotation="0 0 0">      <!-- No rotation applied -->
```

Step 6: Exercises

1. Model Position Testing:

```
<!-- Try each position and observe changes -->
position="0 0 0.1"     <!-- Default -->
position="1 0 0.1"     <!-- Move right -->
position="0 1 0.1"     <!-- Move up -->
position="0 0 1"       <!-- Move forward -->
```

2. Animation Variations:

```
<!-- Test different animation settings -->
animation="property: position;
          to: 0 0.2 0.1;     <!-- Higher bounce -->
          dur: 2000;         <!-- Slower movement -->
          easing: linear"    <!-- Different easing -->
```

Next Steps

After completing all exercises, ensure:

- Your target image compiles successfully
- The 3D model appears when target is detected
- The animation works as expected * You can modify position and animation parameters

Save your working code as `aframe_mindar_part1.html`. You'll build upon this foundation in Part 2.

Part 2: Multi-Target Tracking (45 minutes)

In this part, you'll extend your AR application to track multiple targets simultaneously and add interactive 3D models.

Step 1: Project Setup

Update your project folder with the new assets:

```
Your-Project/
├── Assets/
│   ├── Textures                # Textures used by the models
│   ├── human-orc-targets.mind  # Compiled targets file
│   ├── card-human.png          # Human marker image
│   ├── card-orc.png            # Orc marker image
│   ├── character-human.glb     # Human 3D model
│   ├── character-orc.glb       # Orc 3D model
│   └── weapon-spear.glb        # Spear 3D model
└── aframe_mindar_part2.html
```

Step 2: Compiling Multiple Targets

2. Go to the MindAR Image Compiler: https://hiukim.github.io/mind-ar-js-doc/tools/compile

3. This time, upload both target images:

    - Upload `card-human.png` as first target
    - Upload `card-orc.png` as second target
    - Click "Start Compile"
    - Save the compiled file as `human-orc-targets.mind`

Step 3: Multi-Target Application

Create `aframe_mindar_part2.html` with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <script src="https://aframe.io/releases/1.6.0/aframe.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/mind-ar@1.2.5/dist/mindar-image-aframe.prod.j
s"></script>
  </head>
  <body>
    <a-scene
      mindar-image="imageTargetSrc: Assets/human-orc-targets.mind; maxTrack: 2"
      color-space="sRGB"
      renderer="colorManagement: true, physicallyCorrectLights"
```

```
      vr-mode-ui="enabled: false"
      device-orientation-permission-ui="enabled: false">

      <a-assets>
        <img id="card-human" src="Assets/card-human.png" />
        <img id="card-orc" src="Assets/card-orc.png" />
        <a-asset-item id="model-human" src="Assets/character-human.glb"></a-asset-item>
        <a-asset-item id="model-orc" src="Assets/character-orc.glb"></a-asset-item>
        <a-asset-item id="model-spear" src="Assets/weapon-spear.glb"></a-asset-item>
      </a-assets>

      <a-camera position="0 0 0" look-controls="enabled: false"></a-camera>

      <a-entity mindar-image-target="targetIndex: 0">
        <a-plane src="#card-human" position="0 0 0" height="1" width="1" rotation="0 0 0"><
/a-plane>
        <a-gltf-model src="#model-human" scale="1 1 1" position="0 0 0" rotation="90 0 0"><
/a-gltf-model>
      </a-entity>

      <a-entity mindar-image-target="targetIndex: 1">
        <a-plane src="#card-orc" position="0 0 0" height="1" width="1" rotation="0 0 0"></a
-plane>
        <a-gltf-model src="#model-orc" scale="1 1 1" position="0 0 0" rotation="90 0 0"></a
-gltf-model>
        <a-gltf-model
          id="element-spear"
          src="#model-spear"
          scale="1 1 1"
          position="-0.33 -0.03 0"
          rotation="90 0 0"
          animation="property: position; to: -0.33 -0.03 0.03; dur: 1000; easing: easeInOut
Quad; loop: true; dir: alternate">
        </a-gltf-model>
      </a-entity>
    </a-scene>
  </body>
</html>
```

Step 4: Code Explanation

4. Multi-Target Configuration:

```
mindar-image="imageTargetSrc: Assets/human-orc-targets.mind; maxTrack: 2"
```

- `maxTrack: 2` enables tracking two targets simultaneously
- Each target gets a unique `targetIndex`

3. Asset Management:

```
<a-assets>
  <img id="card-human" src="Assets/card-human.png" />
  <img id="card-orc" src="Assets/card-orc.png" />
  <a-asset-item id="model-human" src="Assets/character-human.glb"></a-asset-item>
  <a-asset-item id="model-orc" src="Assets/character-orc.glb"></a-asset-item>
  <a-asset-item id="model-spear" src="Assets/weapon-spear.glb"></a-asset-item>
</a-assets>
```

- Multiple image targets
- Multiple 3D models
- Unique IDs for each asset

4. Target Entities:

```
<!-- First target (Human) -->
<a-entity mindar-image-target="targetIndex: 0">
  <a-plane src="#card-human"></a-plane>
  <a-gltf-model src="#model-human"></a-gltf-model>
```

```
</a-entity>

<!-- Second target (Orc) -->
<a-entity mindar-image-target="targetIndex: 1">
  <a-plane src="#card-orc"></a-plane>
  <a-gltf-model src="#model-orc"></a-gltf-model>
  <a-gltf-model id="element-spear" src="#model-spear"></a-gltf-model>
</a-entity>
```

Step 5: Exercises

5. Model Positioning:

```
<!-- Try different positions for the models -->
position="0 0 0"      <!-- Default -->
position="0 0.5 0"    <!-- Raised -->
position="-0.5 0 0"   <!-- Shifted left -->
```

4. Spear Animation:

```
<!-- Experiment with spear animation -->
animation="property: position;
          to: -0.33 -0.03 0.05;   <!-- Longer movement -->
          dur: 1500;              <!-- Slower -->
          easing: easeInOutCubic" <!-- Different easing -->
```

Next Steps

After completing this part: * Verify both targets are detected * Confirm models appear correctly * Test spear animation * Try different positions and animations

Save your working code as `aframe_mindar_part2.html`. You'll extend this further in Part 3 with custom components.

Part 3: Interactive Animation (60 minutes)

In this part, you'll create custom A-Frame components to handle marker tracking and animate a 3D model based on marker positions.

Step 1: Project Setup

Ensure your project folder contains all required assets:

```
Your-Project/
├── Assets/
│   ├── Textures
│   ├── human-orc-targets.mind
│   ├── card-human.png
│   ├── card-orc.png
│   ├── character-human.glb
│   ├── character-orc.glb
│   └── weapon-spear.glb
└── aframe_mindar_part3.html
```

Step 2: Component Architecture

Create `aframe_mindar_part3.html` and add the component scripts:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <script src="https://aframe.io/releases/1.6.0/aframe.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/mind-ar@1.2.5/dist/mindar-image-aframe.prod.j
s"></script>
    <script>
```

```
    // Global state for marker visibility
    const marker_visible = {
      'human': false,
      'orc': false
    };

    // First component: Marker event handling
    AFRAME.registerComponent("marker-events", {
      init: function() {
        let el = this.el;

        // Your task: Add event listeners for targetFound and targetLost
        // Each should update marker_visible state and log to console
        // Hint: el.addEventListener("targetFound", function() { ... });

      }
    });

    // Second component: Spear animation
    AFRAME.registerComponent("animate-spear", {
      init: function() {
        // Your task: Initialize necessary references and variables
        // 1. Get marker references using querySelector
        // 2. Get spear reference
        // 3. Create THREE.Vector3 objects for positions
        // 4. Add animation state tracking

      },

      tick: function() {
        // Your task: Implement the animation logic
        // 1. Check if both markers are visible
        // 2. If yes and animation not set:
        //    - Get world positions
        //    - Calculate relative position
        //    - Set up animation
        // 3. If either marker lost:
        //    - Hide spear
        //    - Reset animation state

      }
    });
    </script>
  </head>
  <body>
```

Step 3: Understanding the Components

6. Marker Events Component:

```
// Example structure of event listener
el.addEventListener("targetFound", function() {
  marker_visible[el.id] = true;  // Update state
  console.log(el.id + " found"); // Log event
});

// You need to:
// 1. Add targetFound listener
// 2. Add targetLost listener
// 3. Update the marker_visible object
```

5. Animation Component Initialization:

```
init: function() {
  // Example of getting element reference
  this.marker0 = document.querySelector('#human');
```

```
  // Example of creating position vector
  this.p0 = new THREE.Vector3();

  // Remember to:
  // 1. Get both markers
  // 2. Get spear element
  // 3. Create two position vectors
  // 4. Initialize animation state
}
```

5. Animation Tick Function:

```
tick: function() {
  // Check marker visibility example
  if (marker_visible['human'] && marker_visible['orc']) {
    // Your animation code here
  }

  // Position calculation example
  element.object3D.getWorldPosition(vector);

  // Animation setting example
  element.setAttribute('animation', {
    property: 'position',
    from: "0 0 0",
    to: "1 1 1",
    dur: 2000,
    dir: 'alternate',
    loop: true,
    easing: 'easeInOutQuad'
  });
}
```

Step 4: Scene Setup

Add this scene configuration to your HTML:

```
<a-scene
  mindar-image="imageTargetSrc: Assets/human-orc-targets.mind; maxTrack: 2"
  color-space="sRGB"
  renderer="colorManagement: true, physicallyCorrectLights"
  vr-mode-ui="enabled: false"
  device-orientation-permission-ui="enabled: false">

  <a-assets>
    <img id="card-human" src="Assets/card-human.png" />
    <img id="card-orc" src="Assets/card-orc.png" />
    <a-asset-item id="model-human" src="Assets/character-human.glb"></a-asset-item>
    <a-asset-item id="model-orc" src="Assets/character-orc.glb"></a-asset-item>
    <a-asset-item id="model-spear" src="Assets/weapon-spear.glb"></a-asset-item>
  </a-assets>

  <a-camera position="0 0 0" look-controls="enabled: false"></a-camera>

  <a-entity id="human" mindar-image-target="targetIndex: 0" marker-events>
    <a-plane src="#card-human" position="0 0 0" height="1" width="1" rotation="0 0 0"></a-p
lane>
    <a-gltf-model src="#model-human" scale="1 1 1" position="0 0 0" rotation="90 0 0"></a-g
ltf-model>
  </a-entity>

  <a-entity id="orc" mindar-image-target="targetIndex: 1" marker-events>
    <a-plane src="#card-orc" position="0 0 0" height="1" width="1" rotation="0 0 0"></a-pla
ne>
```

```
    <a-gltf-model src="#model-orc" scale="1 1 1" position="0 0 0" rotation="90 0 0"></a-gltf-model>
    <a-gltf-model id="spear" src="#model-spear" scale="1 1 1" position="-0.33 -0.03 0" rotation="90 0 0"></a-gltf-model>
  </a-entity>

  <a-entity animate-spear></a-entity>
</a-scene>
```

Lab Submission

- You can submit your lab individually or as a pair.
- Save your completed code of *Part 3* as a single HTML file.
- Name the file as *week10_firstname.html* or *week10_firstname1_firstname2.html*, all in lowercase (example: *week10_pisut_tanaboon.html*).
- Submit the file through Google Classroom.
- Do not submit other files.
- Do not forget to add your names as a comment at the beginning of your code.