Documentation:
IEEE Metaverse Initiative WebXR Application

1. Introduction

The application is a cross-platform 3D environment demonstrating information about the IEEE Metaverse initiative. The materials that are demonstrated in the environment are the same as the materials presented in the IEEE Metaverse initiative website (https://metaversereality.ieee.org/). The application can run on three different platforms: HMD VR, computer desktops and mobile VR.

The application comprised four different glitch projects developed using the A-Frame framework (WebXR), each representing a different virtual space which presents different information, categorised by a glitch project:

a. **Welcome**: https://ieeemv-welcome.glitch.me/
b. **About:** https://ieeemv-about.glitch.me/
c. **Events and Publications:** https://ieeemv-events.glitch.me/
d. **Tribute to Roberto Saracco:** https://ieeemv-robertosaracco.glitch.me/

In the Welcome virtual space, the user can find information on how to navigate the environments and interact with an object depending on its selected platform (i.e., VR, computer-desktop, mobile VR). Additionally, a Quiz game is provided in that scene which enables users to engage in an activity where they can test their knowledge of concepts related to the metaverse.

The About virtual space presents relevant information about the initiative which is presented on the initiative's website. The presented information includes the mission statement of the initiative, some "About Us" related information about the initiative, the Initiative Co-chair and the steering committee members. Furthermore, some clickable objects (buttons) are implemented in this virtual space which demonstrate the supporting OUs of the initiative, the Standards of the initiative and the social media of the initiative. These objects when pressed, redirect the user to the corresponding website, like the website.

The Events and Publications virtual space includes clickable objects which redirect the user to 1) papers that were published in events related to the IEEE Metaverse Initiative, and 2) video presentations made under the initiative.

The Tribute to Roberto Saracco virtual space presents a short bio of Roberto Saracco, who recently passed away in early December 2023.

In all the virtual spaces a navigation panel exists where it enables the users to move from one virtual space to the other. (e.g., from the Welcome space to the About space).

2. Technical Architecture

2.1. Framework and Technologies Used

The application consists of four interconnected virtual spaces with each space presenting specific content. The application was developed using Glitch[1], and AFrame[2] framework. AFrame is an open-source web framework for building 3D and virtual reality (VR) experiences in the browser. It's built on top of HTML and WebGL (using the Three.js library) and typically it is used for creating immersive cross-platform, 3D or VR environments directly on the web, without needing to rely on heavy desktop-based software. Glitch, on the other hand, is a web-based platform for creating, remixing, and hosting web applications, which provides an easy-to-use editor for building and deploying apps in real time. Glitch is typically used for quickly prototyping web applications, experimenting with ideas, or sharing code with others. A-Frame projects are often hosted on Glitch to create and share VR experiences easily with collaborators or an audience. Considering the materials of the application, it incorporates interactive clickable elements/objects (buttons) to redirect the user to relevant information, and a quiz in the welcome virtual space. When it comes to the structure of the environment, each space was hosted on a separate glitch project and for that reason, each environment can be accessed using a different link. The reason for implementing the spaces in different projects instead of one is because of several reasons such as 1) modularity (easier to manage features and content specific to each environment), 2) simplicity in maintenance (If a bug occurs in one environment, isolating the environments ensures the issue is confined to a single project, making it easier to identify and fix), and 3) scalability and Performance (Hosting environments separately reduces the amount of data loaded for each interaction, ensuring faster load times and a smoother user experience).

2.2. Hardware and Software Requirements

The application was developed with cross-platform compatibility and currently supports three platforms: 1) HMD VR (i.e., Oculus Quest 2), Desktop (browser-based) and Mobile VR. Hence considering the hardware and software requirements, for HMD VR, an HMD is required, for Desktop a browser is required (running on the majority of popular browsers) and for Mobile VR, the application runs on both Android and Apple devices via a browser. However, for Apple devices, the rotation should be made manually and not using the device rotation.

3. User Guide

3.1. Getting Started

From the user side, to use the application, it is recommended to start from the Welcome space where they will get informed on how to use the application. From the welcome virtual space, they can navigate to the other spaces as they prefer.

3.2. Navigation and Controls

Depending on the platform/technology that the user utilises there are differences in navigation. What is in common with all platforms is the way that the user interacts with the clickable objects. To press

[1] Glitch: https://glitch.com/
[2] AFrame: https://aframe.io/

on an object, in the middle of the screen a circular crosshair is implemented (see Fig 1). This crosshair when hovers over a clickable object starts to minimize. When the circle closes completely then it presses the button and redirects the user to the corresponding information. Considering the navigation, due to the differences in the input and output systems of the platforms/technologies, the navigation (movement and rotation) is handled differently.

- **Virtual Reality:** Rotation is performed as the user rotates their body, while movement is performed using the Touch Controllers Thumb-sticks of both left and right controllers.
- **Desktop:** Rotation is performed using the mouse, while movement is performed using the WASD keys of the keyboard.
- **Mobile VR:** Rotation is performed as the user rotates its body or by swiping left and right on the screen, while movement is performed by pressing the screen.

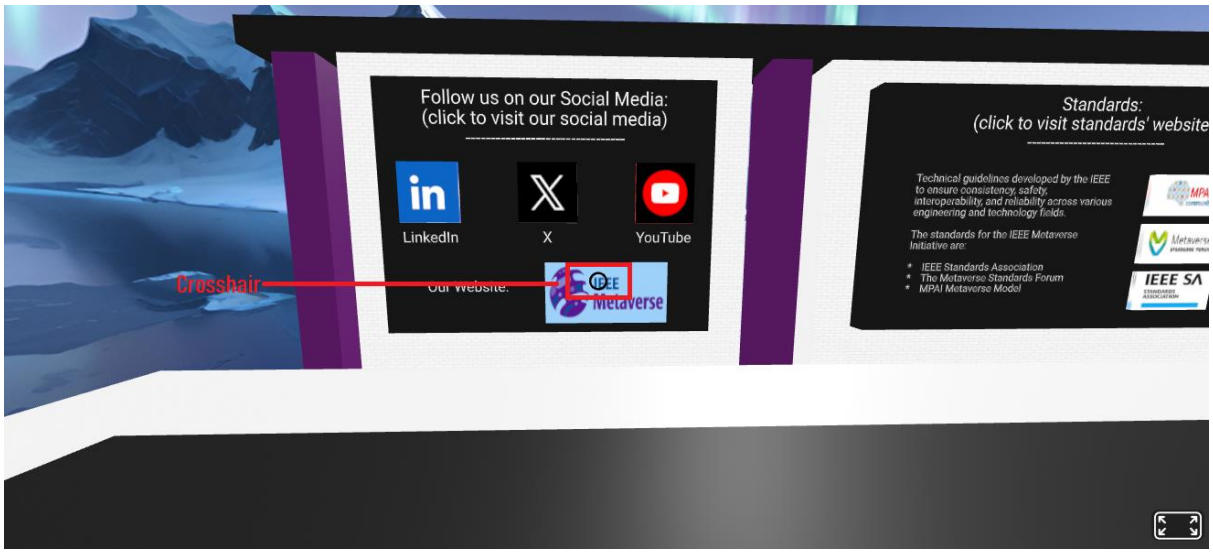

Figure 1: Demonstration of Crosshair.

3.3. Considerations and Troubleshooting

There might be some cases where elements of the virtual spaces are not working as expected. This might be a result of the 1) internet connection, and 2) incompatibility with the device. Some common errors are 1) when the materials are not loaded properly, 2) clickable objects are unable to be pressed and 3) the rotation on mobile VR is not working as described. To fix these issues reload each page, check the compatibility of the device with glitch recommendations and in the extreme case that there is an issue with the rotation capability in HMD VR and mobile VR, clear the cookies of the browser.

4. Developer Guide

All the virtual spaces are implemented using the AFrame framework which utilises HTML and WebGL. For that reason, the basic structure of each space is encapsulated between HTML tags and marked-down syntax as demonstrated in Listing 1. Note that to be able to use the AFrame's components we need to include the AFrame library as indicated in Listing 1 in between the <head> and </head> tags. To use this code we need to remix a WebVR project from the AFrame website.

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.6.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-cylinder>
      <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4" >
          </a-plane>
      <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```

Listing 1: Basic AFrame code structure.

Considering the implemented key components the below listings demonstrate the code structure which is used for their implementation.

A. Camera control: To enable controlling the camera on all three platforms the following code was utilised.

```
<a-scene id="main-scene" physics="gravity: 0">
    <a-entity
     id="cameraRig"
     movement-controls="constrainToNavMesh: true"
     position="0 0 0" rotation="0 90 0"
     dynamic-body>
     <!-- Camera -->
     <a-entity id="head" camera position="0 1.6 0" look-controls="pointerLockEnabled: true">
      <!-- Cursor component -->
      <a-entity id="crosshair" cursor="fuse: true; fuseTimeout: 2000"
       raycaster="objects: .grabbable, .ui-button; showLine:true;"
       position="0 0 -1"
       geometry="primitive: ring; radiusInner: 0.03; radiusOuter: 0.04"
       material="color: black; shader: flat"
       animation__fusing="property: scale; startEvents: fusing; dur: 2000; from: 1 1 1; to: 0.1 0.1 0.1"
      ></a-entity>
     </a-entity>
    </a-entity>
```

Listing 2: Camera controls.

B. Clickable objects: In the environments, all clickable objects have been implemented in a box which enables the user to press it. The boxes have their own materials (skins) to demonstrate to the users where they will navigate them when pressed, while to indicate to the users that the object is about to be pressed, the material changes colour when the user hovers over it. An example of the clickable object structure is provided below. The code provided demonstrates a clickable object structure where when the user hovers over it changes colour and redirects the user to a YouTube video.

```
<a-box
   id="AdoptionButton" class="ui-button clickable"
   data-url="https://www.youtube.com/watch?v=jyVe6634HJg&ab_channel=IEEEFutureDirections"
   animation__mouseenter="property: material.color; to: #9ed1fa; startEvents: mouseenter; dur: 0"
   animation__mouseleave="property: material.color; to: #FFFFFF; startEvents: mouseleave; dur: 0"
   depth="0.1" width="2" height="1" position="6.2 1.5 11.45" rotation="0 0 0"
   material="src: url(https://cdn.glitch.global/1a1a6f80-4207-4e00-ba61-
426807a3e786/AdoptionToMV.PNG?v=1727040478382); color: #FFFFFF; shader: flat;"
></a-box>
```

Listing 3: Basic structure of clickable objects.

To enable all the platforms to react in the same way when the user presses on the clickable object (to redirect the user), the following functions are implemented on a different javascript file. The code to enable this functionality is provided below.

```
<script>
 // Function to load new pages (e.g., About, Welcome, Tribute pages)
 function loadAboutPage() {
   const newProjectURL = "https://ieeemv-about.glitch.me/";
   const sceneEl = document.querySelector('a-scene');

   if (sceneEl.is('vr-mode')) {
     sceneEl.exitVR();  // Exit VR mode
     setTimeout(() => {
       window.location.href = newProjectURL;  // Navigate in the same tab after exiting VR
     }, 100);  // Small delay to ensure the exitVR() process completes
   } else {
     window.location.href = newProjectURL;  // Open in the same tab in non-VR mode
   }
 }

 function loadWelcomePage() {
   const newProjectURL = "https://ieeemv-welcome.glitch.me/";
   const sceneEl = document.querySelector('a-scene');

   if (sceneEl.is('vr-mode')) {
     sceneEl.exitVR();
     setTimeout(() => {
       window.location.href = newProjectURL;
     }, 100);
   } else {
     window.location.href = newProjectURL;
   }
 }

 function loadTributeRSPage() {
   const newProjectURL = "https://ieeemv-robertosaracco.glitch.me/";
   const sceneEl = document.querySelector('a-scene');

   if (sceneEl.is('vr-mode')) {
     sceneEl.exitVR();
```

```
    setTimeout(() => {
      window.location.href = newProjectURL;
    }, 100);
   } else {
    window.location.href = newProjectURL;
   }
  }

  // Function to handle button clicks for opening links or performing actions
  function handleButtonClick(event) {
   const action = event.target.getAttribute('data-action'); // Get action from button

   // Check if action is defined and call corresponding function
   if (action === "loadWelcomePage") {
    loadWelcomePage();
   } else if (action === "loadTributeRSPage") {
    loadTributeRSPage();
   } else if (action === "loadAboutPage") {
    loadAboutPage();
   } else {
    // If no action is defined, check if there's a data-url to open
    const url = event.target.getAttribute('data-url');
    if (url) {
     const sceneEl = document.querySelector('a-scene');
     if (sceneEl.is('vr-mode')) {
      sceneEl.exitVR();
      setTimeout(() => {
       window.location.href = url; // Open in the same tab after exiting VR
      }, 100);
     } else {
      window.location.href = url; // Open in the same tab in non-VR mode
     }
    }
   }
  }

  // Wait for the scene to load and add event listeners to buttons
  document.addEventListener('DOMContentLoaded', function () {
   // Get all elements with the 'ui-button' class
   const buttons = document.querySelectorAll('.ui-button');

   // Add click event listeners to each button
   buttons.forEach(button => {
    // Add click listener to buttons with data-action
    button.addEventListener('click', handleButtonClick);
   });
  });
</script>
```

Listing 4: Script which ensures the same functionality between the three targeted platforms.

To extend the code provided, potentially to include more virtual spaces, modify the existing ones or add additional functionality, the user must create an account on Glitch, target the projects using the link provided and then remix the project. To do that:

1. **Open the original project:** Visit the project you want to remix on Glitch. This is typically shared as a public link, like https://example-project.glitch.me or https://glitch.com/~example-project.
2. **Click on the Remix button:** On the project page, look for the "Remix Your Own" or "Remix" button, usually located in the top-right corner of the page.
3. **Wait for the Remix to load:** Glitch will create a copy of the project under your account. This new version is private by default (unless you choose to make it public).
4. **Edit the Remix:** Once the remix loads, you'll be taken to the Glitch editor, where you can modify the project's files, code, and settings as you wish.
5. **Preview and Publish the new Remix:** As you make changes, the app automatically updates in real time. You can view your changes using the preview pane or by visiting your new project's URL.

5. Environments

The following figures demonstrate the different features of each virtual space.

5.1. Welcome Virtual Space

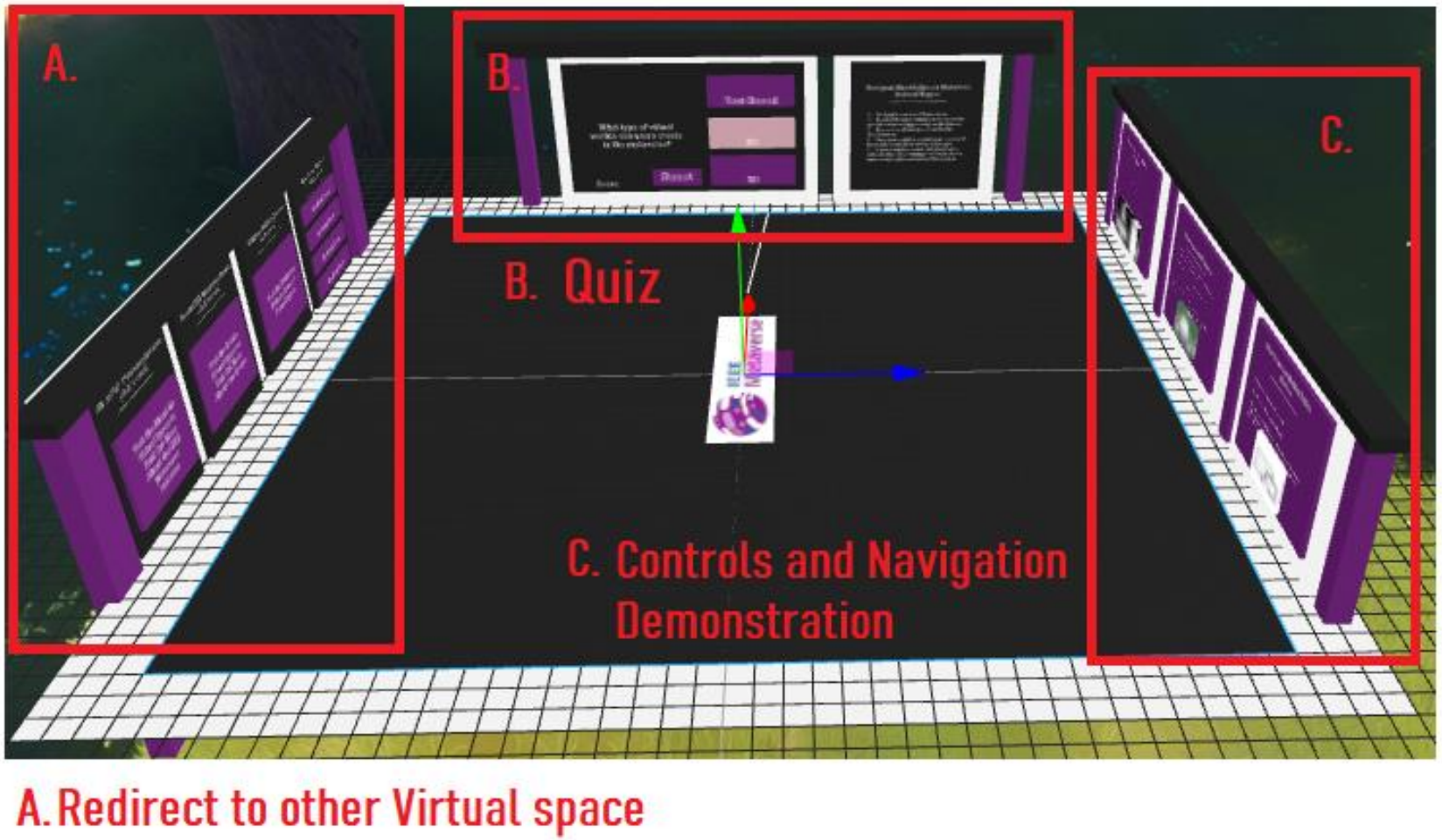

Figure 2: The Welcome virtual space.

5.2. About Virtual Space

Figure 3: About Virtual Space.

5.3. Events and Publications Virtual Space

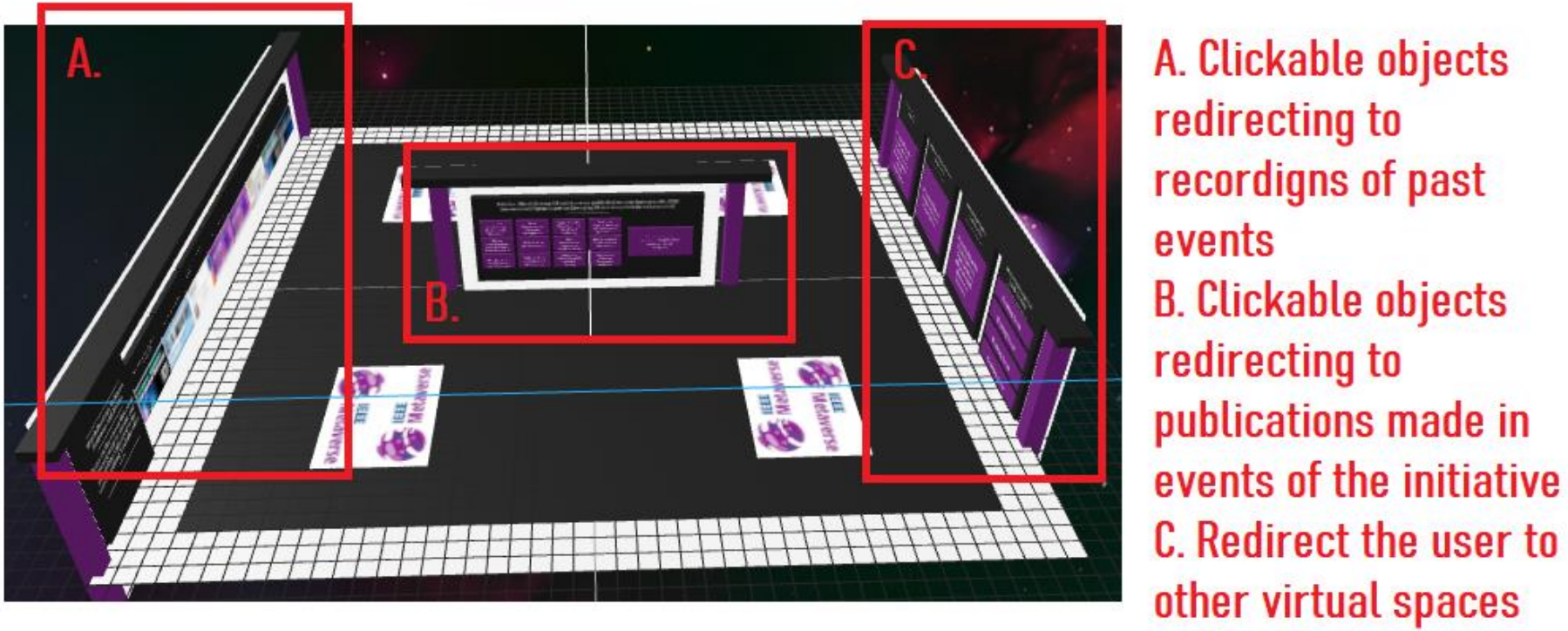

Figure 4: Events and Publications Virtual Space.

5.4. Tribute to Roberto Saracco Virtual Space

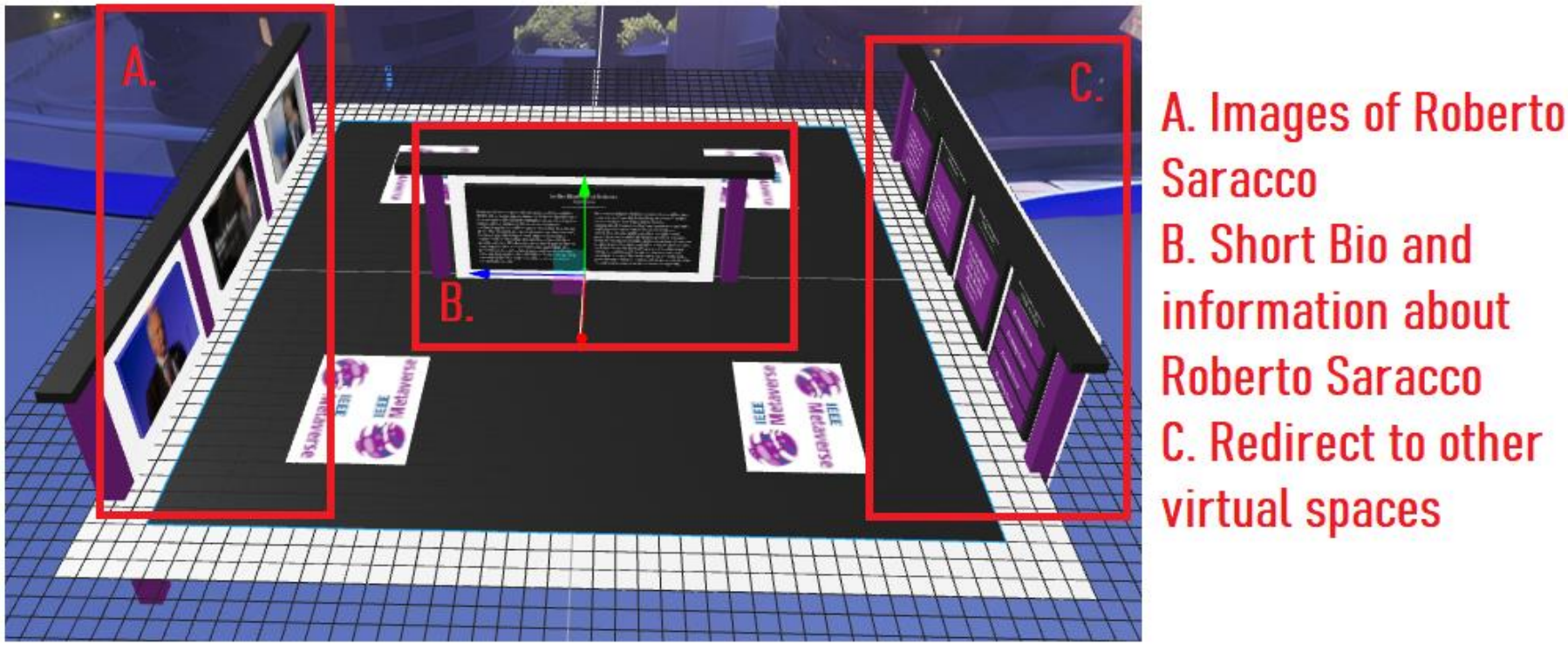

Figure 5: Tribute to Roberto Saracco Virtual Space.