VR Lab #3: Interaction

1.1. Introduction and goals

During the first two labs, you have learned how to reproduce content in virtual environments, and how to design simple, yet meaningful scenarios. This one will focus on *interaction*, a key aspect in VR that can noticeably increase immersion and realism, enhancing the overall experience. Actions like moving around or grabbing items can drastically improve the final experience, and make it feel more realistic. This last laboratory assignment will also serve as a first contact with the setup you will be using in your Course Project (Meta Quest 2).

In this assignment, we are going to apply basic knowledge about movement and interaction in virtual environments, and we will, for the first time, use a Head-mounted Display (HMD) and an external controller. Specifically, you will learn how to:

- Use open-source, pre-designed assets.
- Use the XR Interaction Toolkit.
- Design and run your application in the Meta Quest 2.

Equipment. We will be using the Meta Quest 2. The first step is to set up the hardware so you can work on your laptop. This will be done during the lab session.

Then, there will be different types of tasks for this assignment. Most of them are small, self-explanatory tasks to give you a robust basis for the next assignments. When a new concept is introduced, you may have some **mandatory tasks** to reinforce the learning. Mandatory tasks are preceded by [MT-X]. Finally, **optional tasks** are presented at some points, and preceded by [0T-X]. To be eligible for continuous evaluation, you must complete at least all the mandatory tasks. To opt for the maximum grade (10/10), you need to complete both mandatory and optional tasks.

1.2. Setting up the Meta Quest 2

First, we are going to link the device to your laptop so you can work properly with Unity. For this you will need:

- **Meta Quest Link**: Download and install the Meta Quest Link application from the official Meta website. Once installed, open the Meta Quest Link application, Log in with

your Meta account (talk to the instructors for this step!). Finally, connect your Meta Quest 2 headset to your PC using the Link (USB-C) cable, provided with the Meta Quest 2. Make sure to enable USB debugging when prompted inside the headset. To allow content from unknown sources: Open the Meta Quest app on your PC. Select Settings in the left menu then select the General tab. Next to Unknown Sources, adjust the toggle and then confirm to allow content from unknown sources.

- **Unity 2022 LTS** (this you should have already). If not, make sure to install Unity 2022.3.58f1 (LTS) version, ensuring that Android's package is included in the installation.

Now, we are going to create an empty project from Unity Hub:

1. Open Unity Hub and create a new project: **New Project** → **3D (Built-In Render Pipeline)**.
2. Ensure XR support:
   a) Open **Window** → **Package Manager**.
   b) Change **Packages in Project** to **Unity Registry**.
   c) Scroll down and install:
      - **XR Plugin Management**
      - **XR Interaction Toolkit**
   d) After installing the **XR Interaction Toolkit**, restart Unity when prompted.
   e) Reopen the **Package Manager**, navigate to **XR Interaction Toolkit** → **Samples** → **Starter Assets**, and click **Import**.

   This provides a set of presets for handling inputs and interactions.
3. Adjust project settings for build:
   a) Go to **Edit** → **Project Settings** → **XR Plugin Management**.
   b) Enable **Oculus** in the **PC** or **Android** tab, depending on whether you want to play directly from Unity or build an APK. PC mode allows you to preview your application directly in the Quest when clicking Unity's Play button. We recommend this for development, as it enables quick iteration and testing. Android mode is used for building the final APK, which can be installed and run independently on the Quest. Switch to this mode once your app is complete and ready for deployment.

1.3. First sample scene

Now, we will set up the scene and integrate VR functionality.

1. **Remove the default camera and add an XR camera:** In the **Hierarchy** window, right-click and select: `XR ->XR Origin (VR)`.

2. **Create a simple scene:**
   - In the **Hierarchy**, add a basic environment: `GameObject ->3D Object ->Plane` `GameObject ->3D Object ->Cube`
   - (Optional) You may also import a specific asset for a more structured environment but we recommend to do this later so you can make sure that everything is working with a simple scene first.
3. **Enable Quest Link to preview in the headset:**
   - Put on your Quest 2 headset.
   - Navigate to **Settings** (bottom left of the Quest menu).
   - Select **Quest Link** and enable it.
   - Now, when you press **Play** in Unity, you should see the scene inside your Quest 2.
4. **(Alternative) Build an APK and install it:** If you want to run the app independently on the Quest, you can build an APK and install it using the Meta Developer Hub.

Now, you should be able to see your scene if you put on your Quest 2. Next, we are going to add virtual controllers to the scene.

1. **Download and import controller models:**
   - Download the assets from: `https://developers.meta.com/horizon/downloads/package/oculus-controller-art/`
   - Locate the file `quest2_controllers_div0.fbx` and drag it into the **Project Assets** tab.
   - Drag the imported model into the **Hierarchy** window.
2. **Attach controllers to hand tracking:**
   - Right-click on `quest2_controllers_div0` in the **Hierarchy** and select: `Prefab ->Unpack`.
   - Expand the unpacked hierarchy, locate the objects labeled **_mesh**, and move them under the appropriate hand in the XR Origin structure:
     - Drag the left controller mesh under **XR Origin - Left Controller**.
     - Drag the right controller mesh under **XR Origin - Right Controller**.
     - You may need to reset the transforms (set them to (0,0,0)) of the **_mesh** objects so they don't appear shifted with respect to their parents.
   - Delete any unnecessary parts of the imported model.
3. **Add materials and Render the controllers:**
   - Locate the texture file `nextControllerBoth_color1k.png` in the package you downloaded before with the controllers fbx. Drag this image into the project to use it for the controller materials.

- Right-click in the **Assets** tab and create a new material. Assign the imported texture to the Albedo property of the material.
- Remove the "Skinned Mesh Rendererçomponent from both the left and right controller **_mesh** objects, as it is designed for animated, character-like meshes. Instead, add a **Mesh Renderer** component, which is better suited for static objects like controllers. To do this:
  - Select each **_mesh** object, click **Add Component**, and add a **Mesh Renderer**.
  - Also, add a **Mesh Filter** component by selecting **Add Component** and choosing **Mesh Filter**.
  - In the Mesh Filter component, assign the appropriate mesh: either `left_quest2_mesh` or `right_quest2_mesh`.
- After setting up the Mesh Renderer, drag the material you created earlier onto the Mesh Renderer's Material property. This will provide a basic visual representation of the controllers. Figure 1.1 shows how this should look in your project when you are done.

Fig. 1.1: This figure illustrates how the controllers should appear in your project once the material and mesh renderers are applied correctly.

4. **Add tracking capabilities to the controllers:**
   - Locate the "LeftHand Controller."and RightHand Controller."bjects in the **Hierarchy**. These objects will be responsible for handling the controller's interaction and tracking.

- You should see an existing component called "XR Controller (Action-based).ºn both objects. We will use the default settings for now.
- To assign the appropriate controller mappings, click the button next to the question mark in the "XR Controller (Action-based)çomponent. From the dropdown, select the appropriate option:
  - Select **XRI Default Left Controller** for the left hand controller.
  - Select **XRI Default Right Controller** for the right hand controller.

5. **Configure input action manager:**

   - In the **Hierarchy** window, find the "XR Interaction Manager.ºbject. This object manages interactions between XR devices and the scene.
   - Select the **XR Interaction Manager**, then click **Add Component** and choose **Input Action Manager**. This will allow the controllers to respond to input actions.
   - Once added, click the **+** button in the Input Action Manager component, and select **XRI Default Input Actions** from the dropdown menu. This is a default set of input actions that will handle interactions such as button presses and gestures.

Now you should be able to see the scene and the controllers when you press "Playïn Unity. If everything is working properly, we may move to a nicer scene.

1.4. Importing assets and building a more complex scene

Once your test scene is working, you can move to more complex scenes. In this case, we are going to work with an existing free asset from the Asset Store.

1. **Download the Low Poly Forest Asset** You can use this low poly forest asset to populate your scene: `https://assetstore.unity.com/packages/3d/environments/landscapes/free-low-poly-nature-forest-205742?srsltid=AfmBOoqg8ApirDxONpC8ilQfdups3c__bkWAIvR9wfjuJverxY3nSKiy` Download the asset and add it to your Unity project.

2. **Import the Asset into Your Project** After downloading, go to `Window ->Package Manager`, then from the top-left dropdown, select `My Assets`. Find the `Low Poly Nature Forest` asset, download, and import it into your project.

3. **Add the Demo Scene** Once the asset is imported, navigate to the `Project Console`: `Assets ->Pure Poly ->Free Low Poly Nature Pack ->Scenes`. Drag the `Demo_01` scene into the `Hierarchy` window. This will load the demo scene for the forest environment.

4. **Set Up the Main Scene** To make this new scene your main scene, follow these steps:

   - In your `test scene`, locate the `XR Origin` and `XR Interaction Manager` objects in the `Hierarchy`.
   - Drag and drop both of them into the `Hierarchy` of the `Demo_01` scene.

- Right-click on the new scene in the `Hierarchy` and select `Set Active Scene` to make it your main scene.
- Finally, delete the old test scene to clean up. Your new project should look similar to Figure 1.2.

5. **Adjust the Camera Position** If the camera in the new scene is too deep into the ground, select the entire `forest` object in the `Hierarchy` and adjust its `Y` position to `-5` (or as needed). This will raise the terrain and place the camera at an appropriate height.

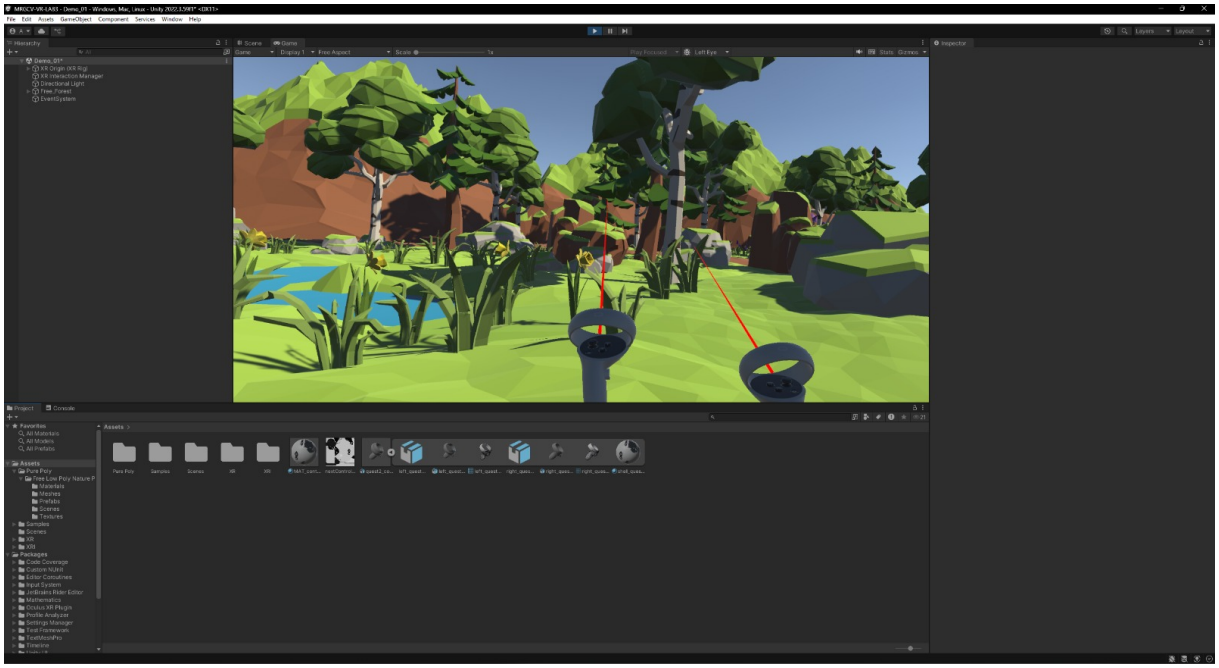

Fig. 1.2: This figure shows how your final scene should look while running, with the forest as the main scene and the controllers.

1.5. Tasks for this Lab Session

[MT-0] Implement a teleportation-based movement system that allows you to teleport a certain distance in a given direction. *Hint: Look into the* ***`XR Interaction Toolkit`*** *and the* ***`Teleportation Provider`****. Unity 2022 provides built-in support for teleportation mechanics, which can be configured using the* ***`XR Ray Interactor`*** *and* ***`XR Teleportation Area`****.*

[MT-1] Enable object interactions: Implement the ability to grab and move objects using the right controller. *Hint: The* ***`XR Grab Interactable`*** *component in the* ***`XR Interaction Toolkit`*** *provides an easy way to allow grabbing and moving objects. Ensure that the* ***`XR Interaction Manager`*** *is properly set up in your scene, as it handles all interaction events.*

[OT-0] Research the games *Half-Life: Alyx* and *Horizon Call of the Mountain* and analyze their interaction and movement paradigms. These games are widely regarded as some of the best VR experiences, featuring high-quality interactions and multiple locomotion options.

In your report, describe the different ways players can move and interact with the environment, and discuss how these design choices improve immersion and usability. Consider terms like `continuous movement`, `blink teleportation`, `gravity gloves`, and `realistic object interactions`.

1.6. Reporting your results

You should submit a **report** (in .PDF format) including your results for the tasks (results should be clearly labeled following the labels of the tasks they correspond to). There is no required style or fixed structure for your report, you will have to choose how to report the work you did. There are, however, some guidelines that you need to follow:

- The report should not be longer than **three pages** (*tres "carillas", no tres páginas*), and should at least address the mandatory tasks. The report file should be named `labXX-mainReport-YYYYYY.pdf`, with `XX` the number of the lab, and `YYYYYY` your NIP.
- Do not include whole snippets of code in the report: You may submit an additional .ZIP file with the code and shaders you wrote, and indicate in the report its function with a couple sentences at most (if necessary).
- Including in the report the **main difficulties, thoughts, or insights** you had through the whole laboratory, as well as its relation to the lectures of the course, will be positively evaluated.
- If there is any other part of your work that cannot fit in the report (e.g., short videos or sets of a large number of screenshots), you should submit them in a separated, supplementary .ZIP file, **but always adequately referencing it in the main report, so we are aware of the existence of that file**.
- We will not look at submitted items that have not been referenced in the main report.

In terms of evaluation, mandatory tasks can get you up to 8 points out of 10, and the 2 remaining points can be obtained through the optional tasks and the quality of the report.

You should submit the assignment via Moodle, uploading them in the corresponding Moodle task. **Deadline: April 1st, 2025 @ 23:59**.