COMP.SGN.320.2023-2024 DSLF and Virtual Reality

Densely Sampled Light Field and Virtual Reality

1 General Instructions

This assignment is divided into two parts. In the first part of the assignment, the main goal is to cover some of the principles related to processing Densely Sampled Light Field (DSLF) data for a target light-field display. In the second part, the goal is to explore a few tools and a framework to capture, process and display virtual content in Head Monted Displays (HMDs).

Tasks

This assignment consists of several tasks that can be performed at home. However, you must attend the lab session in CIVIT (SA201) to capture your own dataset for the virtual reality tasks. Completing all mandatory tasks will grant you grade 1 and completing any additional task will increase the grade by 1 or 2 values, up to grade 5.

Deliverables

Each group should return:
1 – A (brief) **report** with all the equations and estimated values for task 3.1.
2 – **Matlab script** that does the required work. Your script should execute without any errors. Please, **list clearly which steps you have completed** in the comments, at the beginning of the main Matlab script.
3 – **Unity project folder** named **LW3Scene** which must contain all the assets and logic applied to your Unity project and the following files: **LW3Scene.meta, LW3Scene.unity** and **LW3Scene.unity.meta**.

The submission must contain all deliveries mentioned above, everything within a single **ZIP** file. See Moodle for the exact deadline. Possible extensions to the deadline should be negotiated **before** the deadline and may be awarded at the discretion of the assistant if valid arguments are presented.

2 Densely Sampled Light Field (DSLF)

The concept of light field was introduced as a plenoptic function which describes the light intensity at a particular position, direction, and time instance with a specific wavelength. The plenoptic function has 7 dimensions and it can be reduced to 5 dimensions by assuming a stationary scene with monochromatic illumination. Moreover, light's radiance stays constant while traveling from one point to another until it hits an opaque object. Therefore, the plenoptic function's dimensionality can be further reduced to 4 dimensions. There are multiple ways to represent a 4D light field, but two-plane parametrization is used in this task. The two-plane parametrization is denoted as $L(s,t,u,v)$, in which (s,t) and (u,v) are coordinates on each plane that the corresponding light ray travels through. In the two-plane parametrization, plane (s,t) corresponds to various viewpoints and plane (u,v) contains images of each viewpoint. A densely sampled light field (DSLF) is a light field in which the absolute value of the minimum and the maximum disparities between two adjacent camera views are less than or equal to one. The required sampling density of DSLF depends on several factors including the minimum distance of the scene to the camera plane and the camera resolution. Therefore, DSLF of a scene with objects closer to the view plane must have a smaller distance between adjacent views.

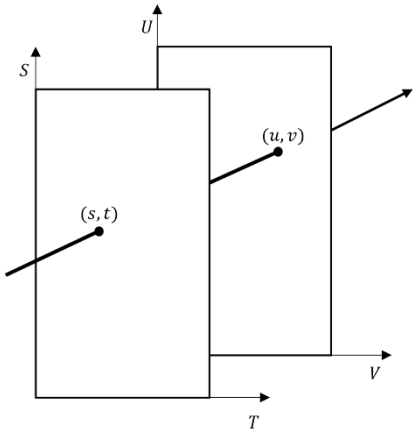


Figure 1 – Two-plane parameterization $L(s, t, u, v)$

The projection-based light field displays can reconstruct an approximation of the continuous plenoptic function out of a discrete set of rays. Projection-based light field displays are made of two main parts. The first part is a set of projectors that count as ray generators, and the second part is the screen plane which is an optical element that acts as a discrete to continuous converter.

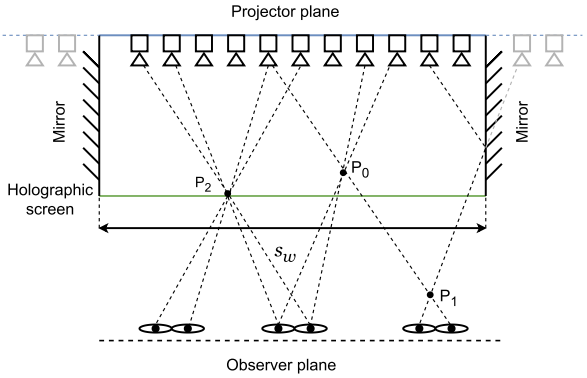


Figure 2 –Projection-based light field display principle (top view).

3 Homework tasks - DSLF

3.1. DSLF parameterization and values (mandatory)

To visualize 3D content on the projection-based light field display we must prepare a densely sampled light field. Thus, we must parameterize our capturing setup in function of the display working principle and its properties. The first step is to compute the missing values in the table below. To help you with this task, all geometric constraints related to the display and the capturing setup are depicted in Figure 3 and Figure 4. In addition, a set of equations are given below to help and support your calculations. Your task is to calculate the values for B, n, and FoV.

Parameter Name	Parameter Symbol	Value
minimum distance from camera to scene	Z_{min}	2.75 [meters]
maximum distance from camera to scene	Z_{max}	3.25 [meters]
width of the scene	X_{scene}	1.56 [meters]
baseline	B_{1_n}	2 [meters]
adjacent view distance	B	**?** [meters]
number of views	n	**?**
field of view	FoV	**?** [degrees]
horizontal resolution	S_x	1920 [pixels]
vertical resolution	S_y	1080 [pixels]

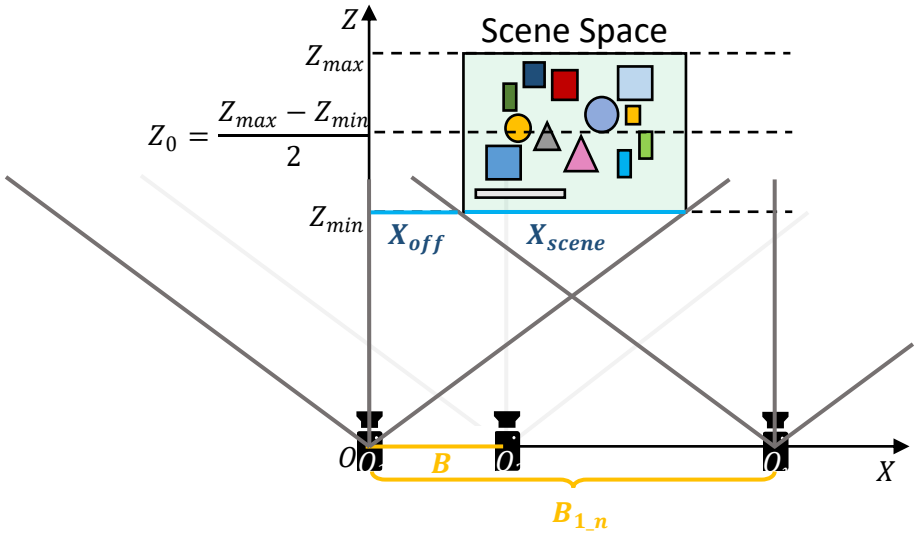


Figure 3 – Parameterization of capturing setup with respect to the scene space.

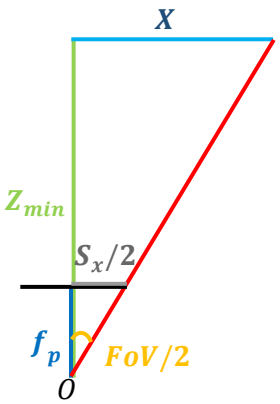


Figure 4 – Parameterization of a camera (cross-section).

Disparity constraint for densely sampled light field:

$$D_{dif} = D_{max} - D_{min} \leq 1 \qquad D_{min} = \frac{f_p B}{Z_{max}} \qquad D_{max} = \frac{f_p B}{Z_{min}}$$

Relation between focal length and field-of-view + relations between field-of-view, scene space and baseline:

$$f_p = \frac{S_x}{2}\cot\left(\frac{FoV}{2}\right) \qquad FoV = 2\text{atan}\left(\frac{X}{Z_{min}}\right) = 2\,atan\left(\frac{B_{1\_n} + X_{scene}}{2Z_{min}}\right)$$

Relation between baseline, adjacent view distance, number of images and scene space:

$$B = \frac{B_{1\_n}}{n-1} = \frac{X_{scene} + 2X_{off}}{n-1} = \frac{2X - X_{scene}}{n-1}$$

3.2. Create virtual content (+1)

Create a **custom** 3D model (a cube) consisting of N=8 vertices and M=12 polygons (or triangles) by manually defining the points, their connectivity, and other attributes. Please, use the auxiliar library "RenderingPipeline" for all your tasks.

- Each vertex is a 3D point (X_N, Y_N, Z_N) in 3D space and the set of vertices is stored in a 3xN matrix of coordinates.
- Each polygon (or face) is a connection between three vertices, and it is stored as triplets of indices into the array of vertices, i.e. 3xN matrix where each column represents a connection between the indexed vertices.
- The coordinates are given in meters and relative to the *world origin* (0, 0, 0).
- Assign different colors in a 3xN matrix to each polygon so that each face of the cube has a different color. You may also assign the color for each vertex.

In addition to the custom 3D model, you are encouraged to play with the other scene components, including materials and lights. Once you are done, you should add more 3D models (e.g., load .ply file, create more custom 3D models) and set your own scene within the scene space limits ($Z_{min}, Z_{max}, X_{scene}$).

To preview your model, you can use *patch* function.

3.3. Render virtual content (+1)

Once the scene is within the scene space, you must implement the main rendering loop to render n views, and set the field-of-view (FoV) of the main camera plus the distance/step between each adjacent view (B).

In the end you should save each rendered view in your local disk. These set of images will be the input data for the projection-based light field display in CIVIT.

4 Virtual Reality (VR)

Virtual Reality (VR) is a technology that has revolutionized the way we experience and interact with digital environments. It immerses users in a computer-generated 3D space, providing a sense of presence and interaction that goes beyond traditional screen-based experiences. In VR, users wear specialized headsets known as Head-Mounted Displays (HMDs) that transport them to entirely synthetic worlds. VR is nowadays present in many different types of content and applications ranging from gaming, digital marketing, education to healthcare, safety guidance and engineering.

At its core, the working principle of VR involves creating a sensory-rich environment that may mimic the real world or transports users to fantastical realms. The key components are specialized headsets (HMD) with audio, motion tracking devices, and often, haptic feedback systems. The HMD serves as the user's window into the virtual space, providing a stereoscopic display that adjusts as the user moves their head. Motion trackers, such as accelerometers and gyroscopes, capture the user's movements, ensuring that the virtual environment responds in real-time to their actions. Additionally, haptic feedback systems that simulate touch sensations, and audio signals provide a more immersive experience.

The creation and delivery of VR content involve a sophisticated pipeline to ensure a seamless and immersive experience. The process typically begins with the capture of real-world data or the creation of 3D models. This data is then processed to optimize it for real-time rendering in a virtual environment. Various technologies, such as photogrammetry or 3D modeling software, are employed in this stage.

Once the content is prepared, it undergoes a rendering process where it is transformed into a format suitable for the VR headset. This involves creating stereoscopic views to simulate depth and adjusting for the field of view and lens distortion of the headset. The processed content is then displayed in real-time as users interact with the virtual environment.

The pipeline also involves considerations for user interaction, integrating controllers, and designing interfaces that enable users to navigate and manipulate the virtual space effectively. An example of the full VR framework, from content creation to display, is depicted in Figure 5. It requires a seamless flow of data and synchronization to maintain the illusion of presence and responsiveness.

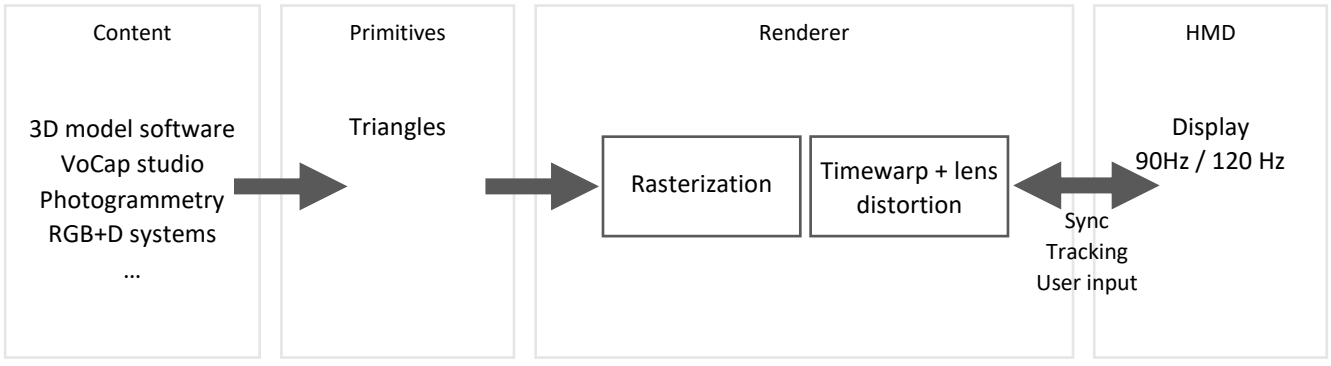


Figure 5 – Basic framework to capture, process and display VR content.

5 Labwork tasks – VR

5.1. Content capturing with VoCap (mandatory)

The virtual content is going to be generated using a volumetric capturing studio (VoCap). The VoCap is able to capture any 3D object from all directions. As a result, you get a dense point cloud and the respective mesh. VoCap can capture dynamic scenes but for the sake of this exercise we will capturing a static scene (e.g., 3D object, a person standing still) and in the end, it will be your homework task to animate or interact with the captured scene.

6 Homework tasks –VR

6.1. Interact with the virtual environment (+3)

As you can imagine, the renderer stage in Figure 5 is rather complex and therefore we will make use of existing tools and software. One of the most popular tools to handle the rendering stage and interact with the HMD is Unity: https://unity.com/. This is a free software and you are going to need it throughout this task. To make things more clear, this task is divided into smaller subtasks. You are encouraged to search things on internet or consult the TA whenever you face some problem or whenever you have a question.

NOTE: You will get **+1 point** for setting up Unity environment and importing the captured model. You will get **+2 points** if you create some interaction with the virtual environment (either with the captured model or with other virtual object or 3D model that you may add), or if you animate the captured model (in case you have captured some person), or if you add some other effects to the virtual environment (SFX, particle systems, etc). The goal of this exercise is not to learn how to use Unity but rather experience some of the things you can create and program in VR. There is plenty of material across the internet about VR development in Unity. You are free to use any resource available (google, ChatGPT, YouTube videos). You will have a chance to see your work in a VR headset after LW3 deadline.

Installing Unity and other tools:

For this task, you need to install **Unity Editor 2022.3.xxf1 LTS**, **Unity Hub 3.6.0** or any later version, and an IDE to edit the scripts. For the IDE, I recommend to use **Microsoft Visual Studio 2019 community** version or any later version.

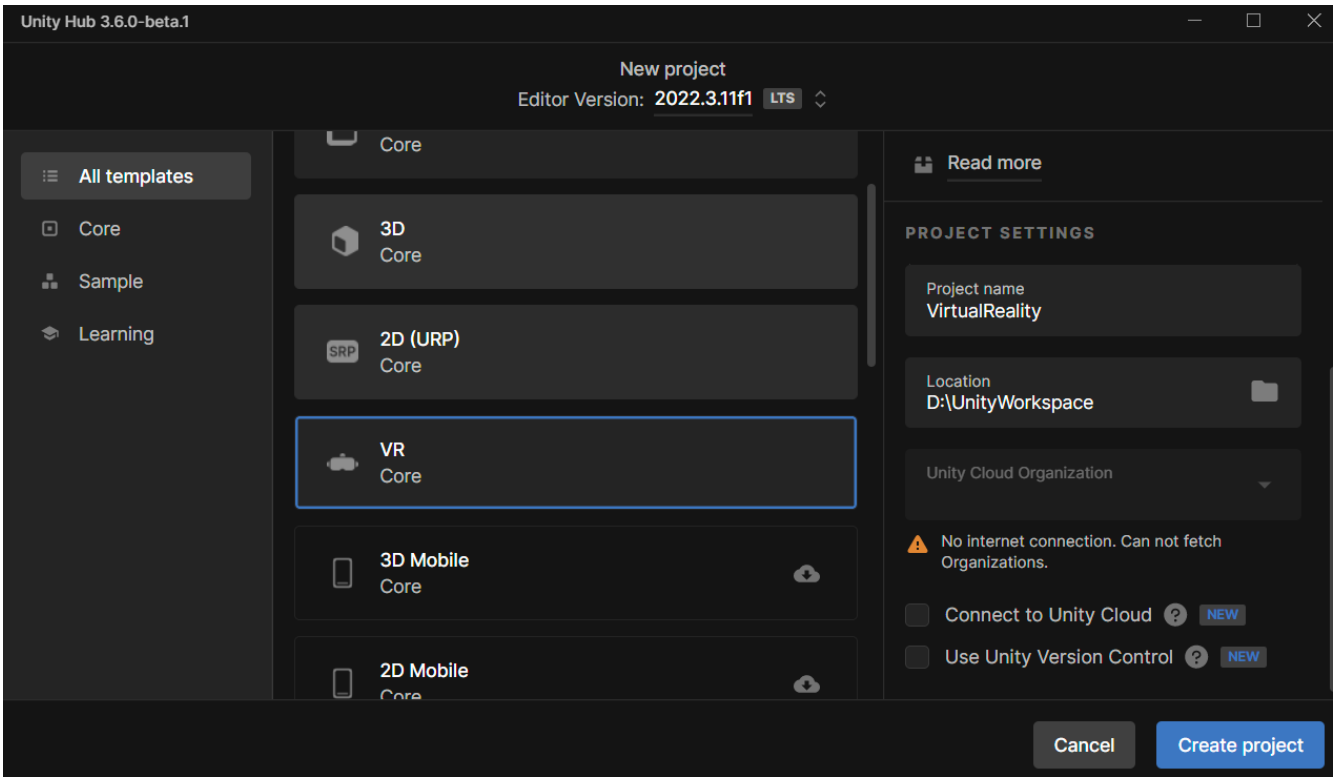
Start by installing the **Microsoft Visual Studio 2022**: https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&passive=false&cid=2030 . During this process, there will be a window with the modules that you would like to install. You should install at least **Desktop development with C++** and **Game Development with Unity**.

Then, once the IDE is installed, you can proceed and install Unity Hub: https://public-cdn.cloud.unity3d.com/hub/prod/UnityHubSetup.exe

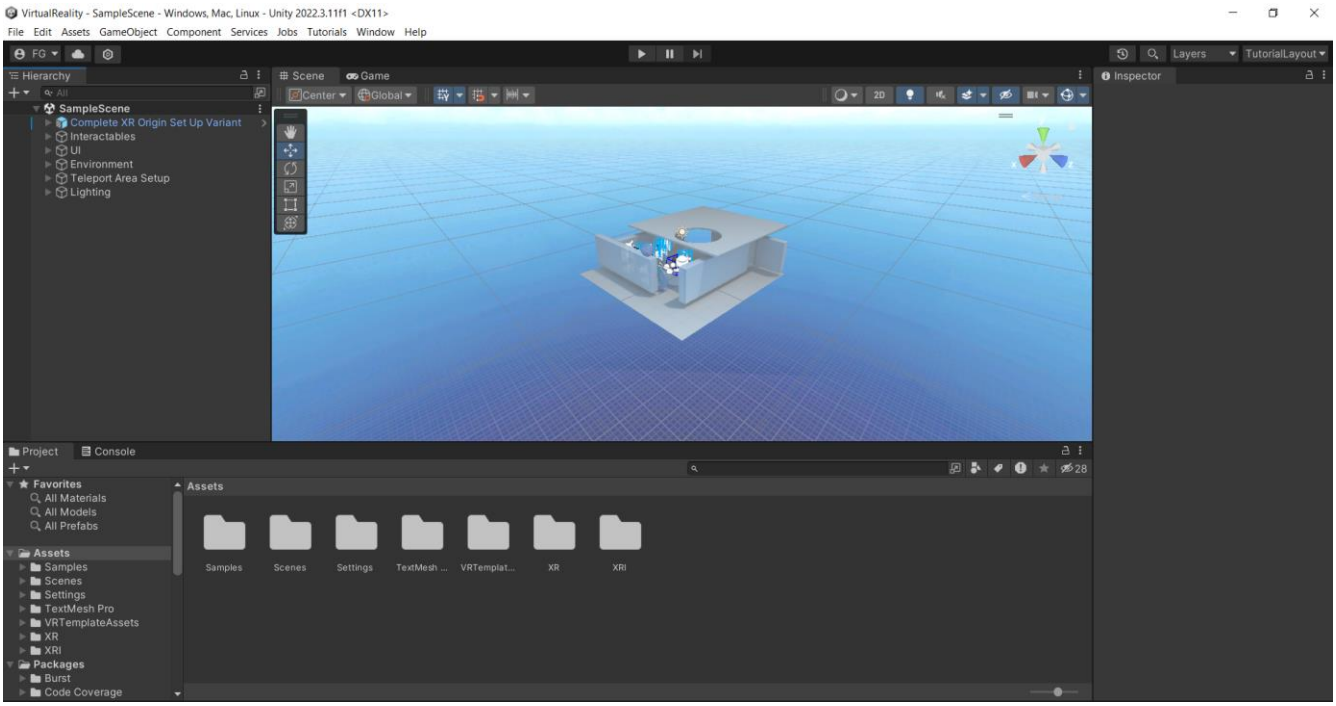
Finally, inside Unity Hub, you can install Unity Editor 2022.3.xx LTS version (recommended version).

Setting up unity environment:

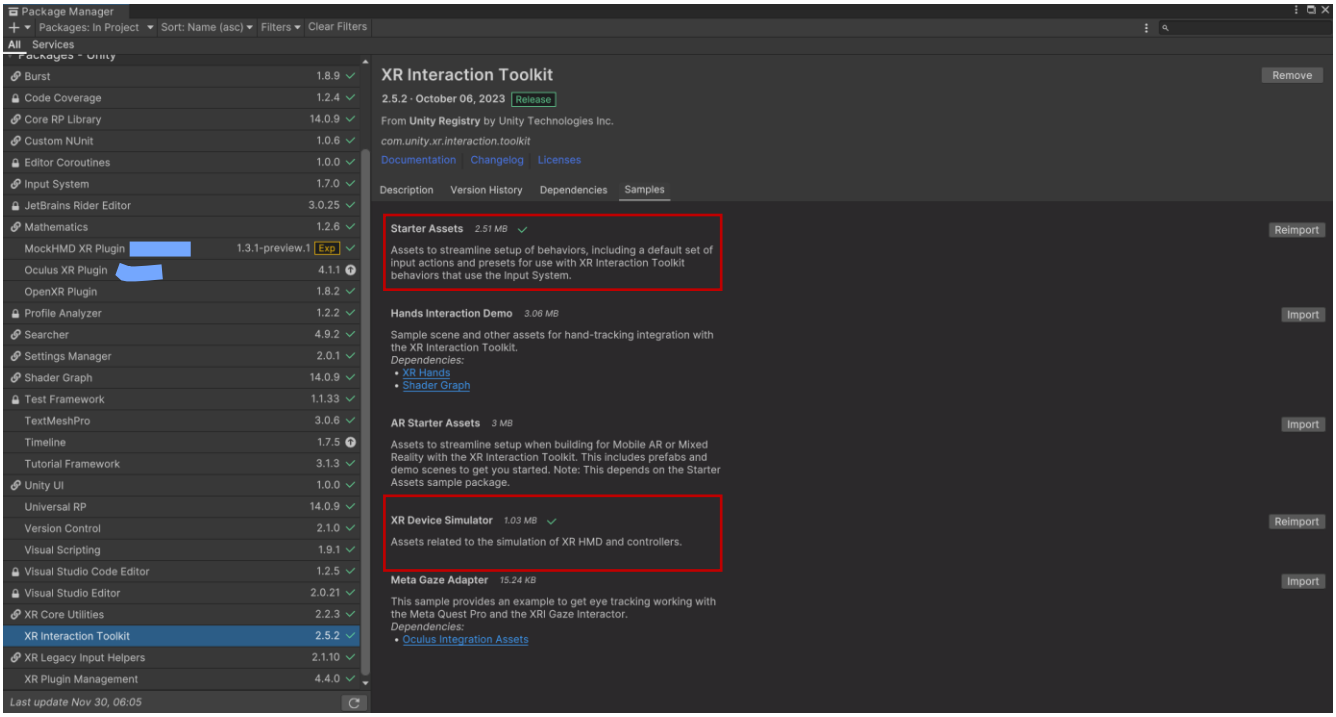
Once everything is installed, open Unity Hub. Then, inside Unity Hub, press **New project** and then press **VR-Core** template (you may need to download the template first). Once it is installed, give your project a name and pick the location for your project workspace. Uncheck all ticking boxes and press **Create Project**.



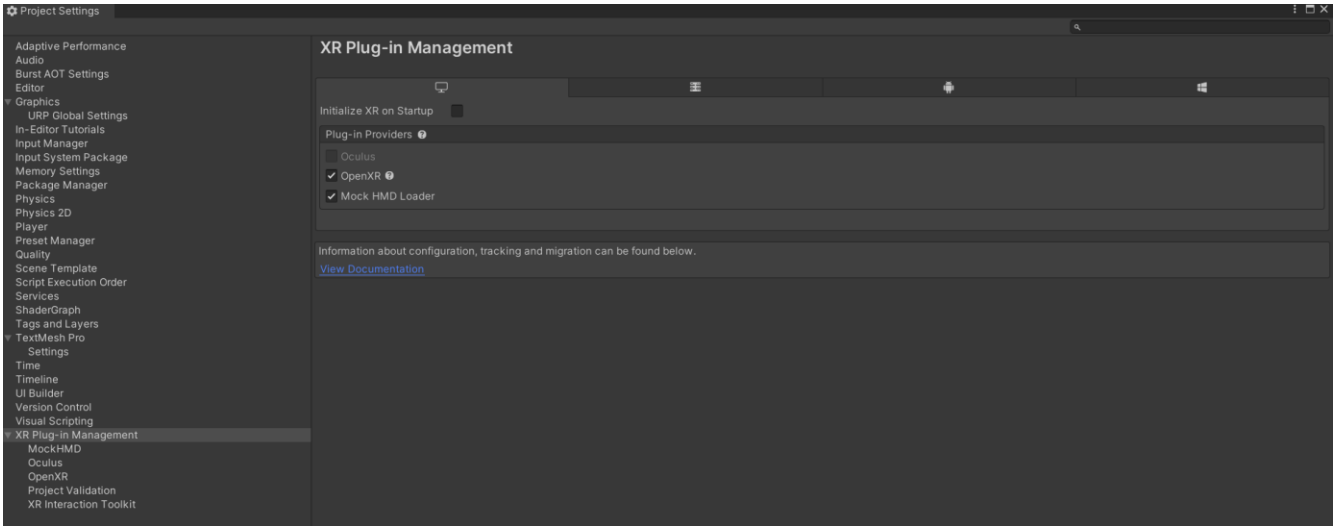
After creating the project, Unity will download some VR packages and then, you should see something like this:



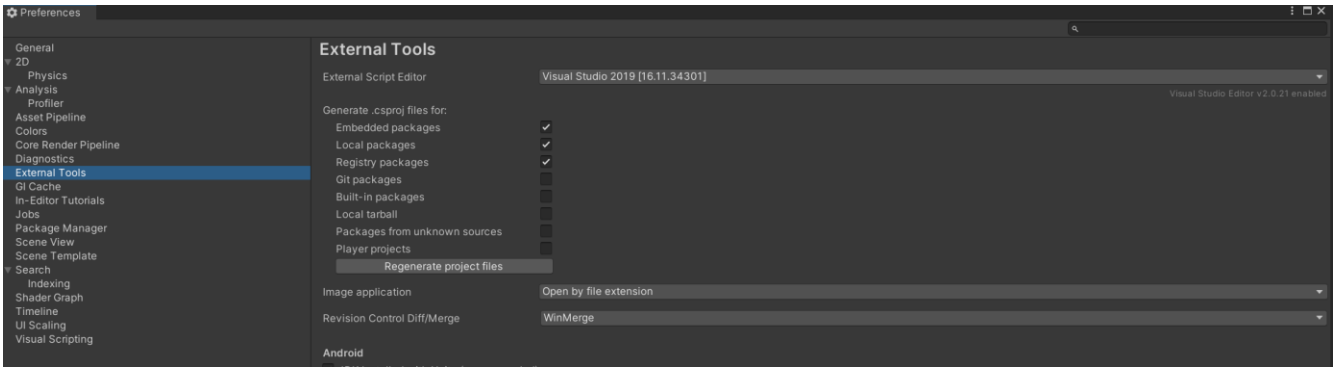
Before you start working on your project, there are a few VR packages and configurations that are missing. First, open **Window → Package Manager** located on the top menu. Make sure you have the packages listed below and that you have imported the Samples (marked in red) inside **XR Interaction Toolkit**:



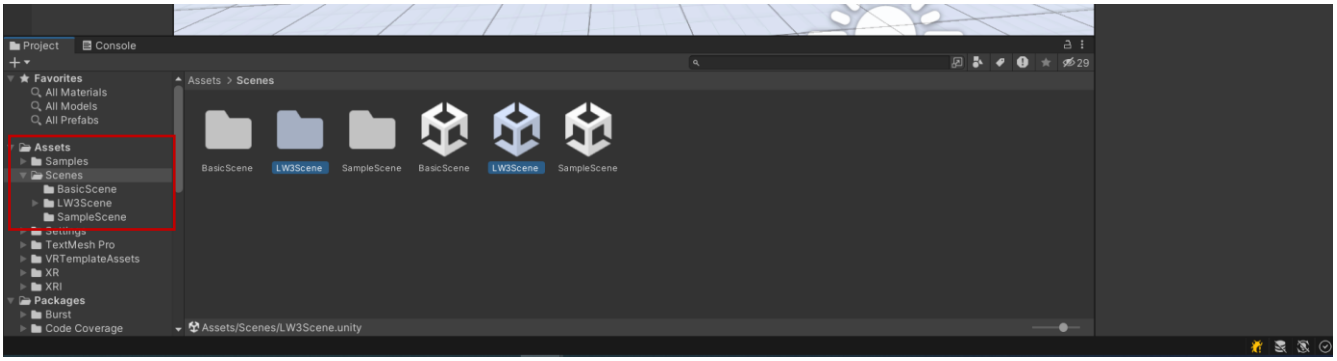
Then, we need to configure our project for offline VR development by going to **Edit → Project Settings…** , and pressing **XR Plug-in Management**. Then, select **Mock HMD Loader** and make sure **Initialize XR on Startup** is unticked:



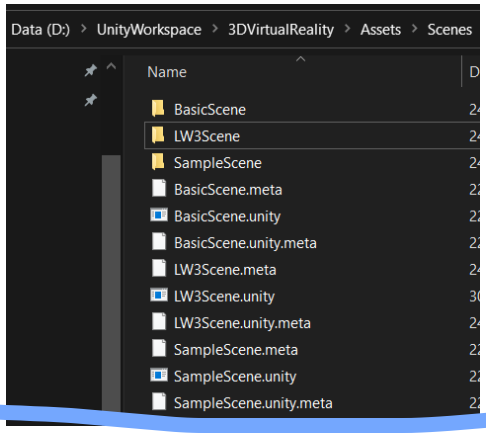
Finally, go to **Edit → Preferences**, **External Tools** and make sure that the **External Script Editor** is set to MSVS 2019 or MSVS 2022:



Once everything above is configured correctly, you can import the templates from Moodle to you project. **LW3Scene** folder and **LW3Scene.unity** should be pasted inside **Assets→Scenes** folder:



Make sure you also copy ***.meta** files (**LW3Scene.meta** and **LW3Scene.unity.meta)** to your working directory which should be located at **<PATH_TO_UNITY_WORKSPACE>\<PROJECT_NAME>\Assets\Scenes**:

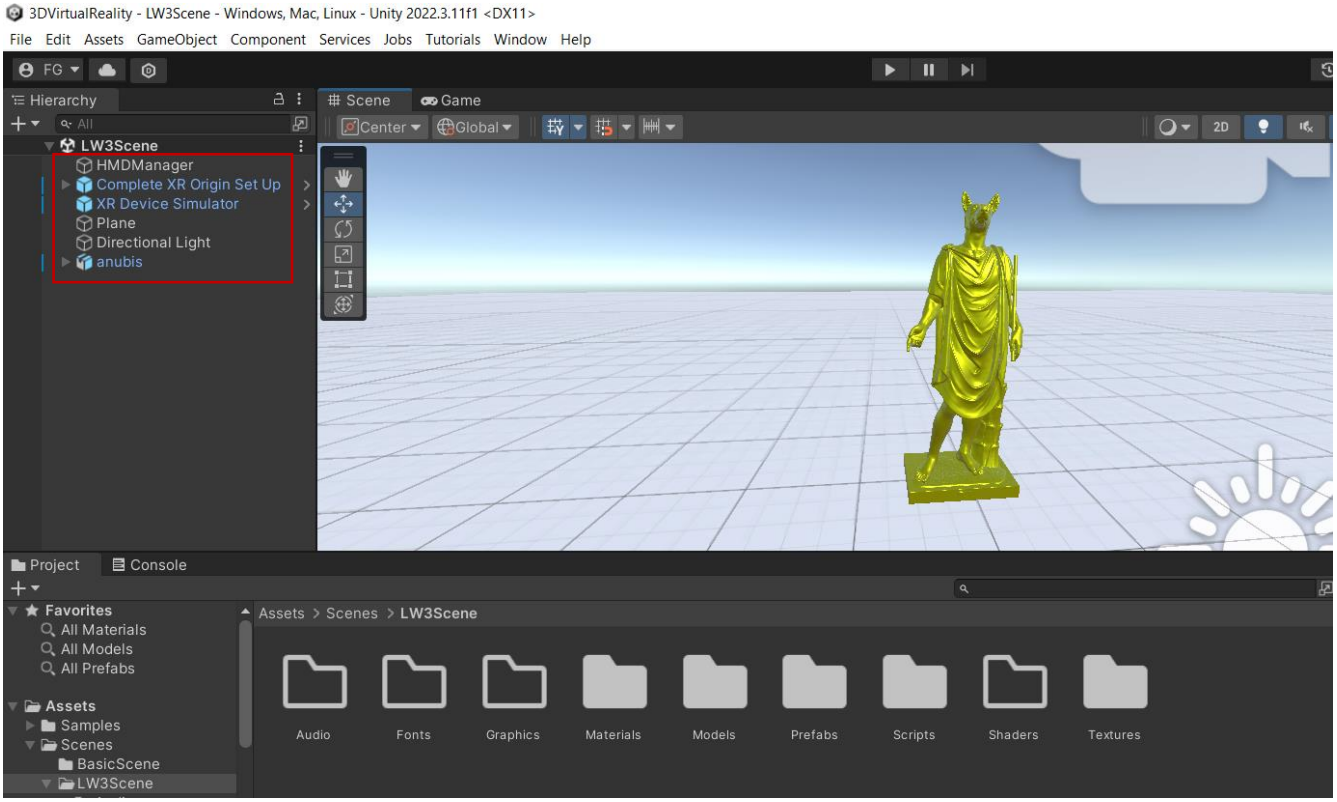


After the steps above, double click in **LW3Scene.unity** inside Unity Editor. This will open a new scene which is going to serve as a base for your task. If you need to add any resources or logic to your scene, use the folder **LW3Scene** and its subfolders.

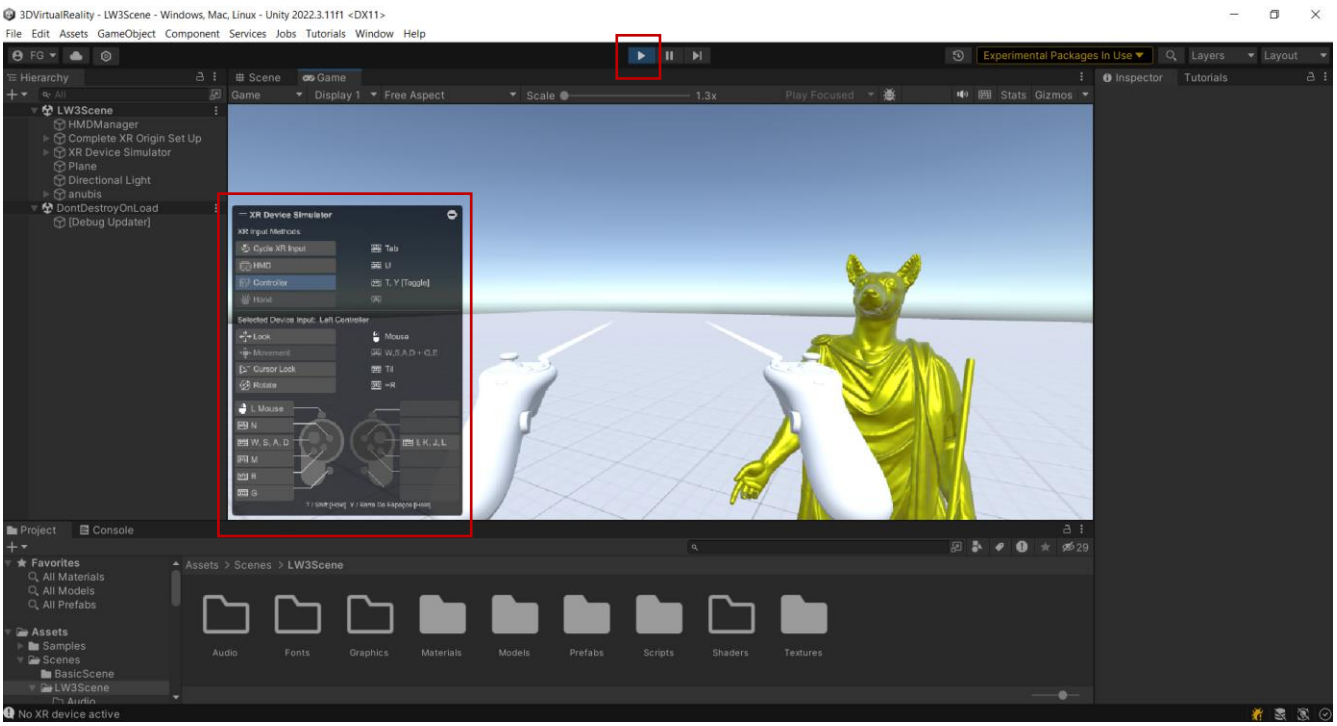
NOTE: **LW3Scene** folder, **LW3Scene.meta, LW3Scene.unity** and **LW3Scene.unity.meta** should be part of your submission. If for any reason you need to use other resources or packages that are not inside LW3Scene, please mention it to the TA.

Developing your scene:

As you can see in **Hierarchy** panel, you have some components by default. You are free to delete and edit all component except **HMDManager**, **Complete XR Origin Set Up**, and **XR Device Simulator**, as they play an important role in offline VR development. Thus, be very careful to not delete them or edit them.



If you press the play button, you should be able to navigate and use the VR controllers as if you would have the actual VR device. Please check the keyboard keys on the bottom-left panel on how to use the VR controllers and navigate in the scene:



After you are familiar with Unity environment and you know how to navigate and use the VR controllers in your scene, you can start composing your scene.

Some online resources that you can use in your development:

Animate 3D character: https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack and https://actorcore.reallusion.com/auto-rig/accurig

Unity store (requires Unity account though). Easy to import (free) assets: https://assetstore.unity.com/

VR dev. (tips and ideas): https://www.youtube.com/playlist?list=PLrk7hDwk64-a_gf7mBBduQb3PEBYnG4fU

Basic tutorial about Unity: https://www.youtube.com/watch?v=pwZpJzpE2lQ