**CMP510 (AR and VR Technologies)**

**Assignment**

****

**Navrachana University**

**Objective:**

To create a basic virtual environment in Unity that includes a ground plane, a skybox, environmental objects, lighting, and simple VR interaction. The player should be able to grab and move the grabbable objects in the environment.

**Task 1:** Set Up Your Unity Project & Configure the VR Environment **[5 marks]**

**Step 1:** **Create a New Unity Project:**

* Open Unity Hub and click on "New Project."
* Choose the **3D template** for VR.
* Give your project a name and set the project location.
* Click **Create Project**.

**Step 2:** **Install XR Interaction Toolkit:**

* Go to **Window > Package Manager**.
* In the Package Manager, click **+** (Add) and select **Add package by name**.
* Enter **com.unity.xr.interaction.toolkit** to install the XR Interaction Toolkit package.
* After it’s installed, ensure you import the **Starter Assets** from XR Interaction Toolkit.

**Step 3:** **Configure XR Plug-in Management:**

* Go to **Edit > Project Settings**.
* Under **XR Plug-in Management**, install the **XR Plug-in Management** system.
* Select **PC, Mac & Linux Standalone** and enable **XR Plugin Management**. Then, select **OpenXR** as the plugin provider.
* If errors appear, resolve them before proceeding.

**Step 4:** **Enable XR Interaction Toolkit:**

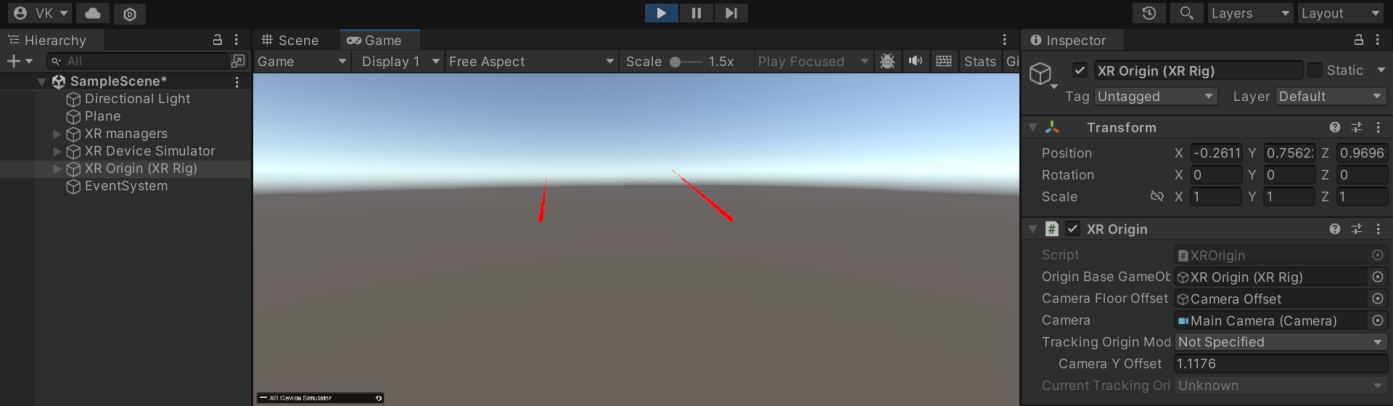
* Once installed, go to the **XR Interaction Toolkit** settings in the project settings and ensure everything is properly set up for your VR environment.
* Enable the **XR Device Simulator** for keyboard and mouse input, which will allow you to interact without physical controllers.

**Step 5:** **Add the XR Device Simulator:**

* In the **Hierarchy**, right-click and go to **XR > XR Device Simulator**.
* This allows you to simulate VR controller movements with keyboard input (like WASD keys and mouse).

**Step 6:** **Create a Simple Environment:**

* Add your scene to see player movement.



**Task 2:** Create the Ground Plane **[5 marks]**

● Create a larger ground area so player can move around

● You can use Terrain object for that

**Step 1:** **Create a Terrain Object:**

* In the **Hierarchy**, right-click and go to **3D Object > Terrain** to create a terrain for your ground.
* This terrain will serve as a large ground area, allowing the player to move around freely.

**Step 2:** **Resize the Terrain:**

* Select the **Terrain** in the **Hierarchy**.
* In the **Inspector**, under the **Terrain Component**, adjust the **Width** and **Length** to create a larger area for player movement (e.g., 500x500 units).

**Step 3:** **Sculpt the Terrain (Optional):**

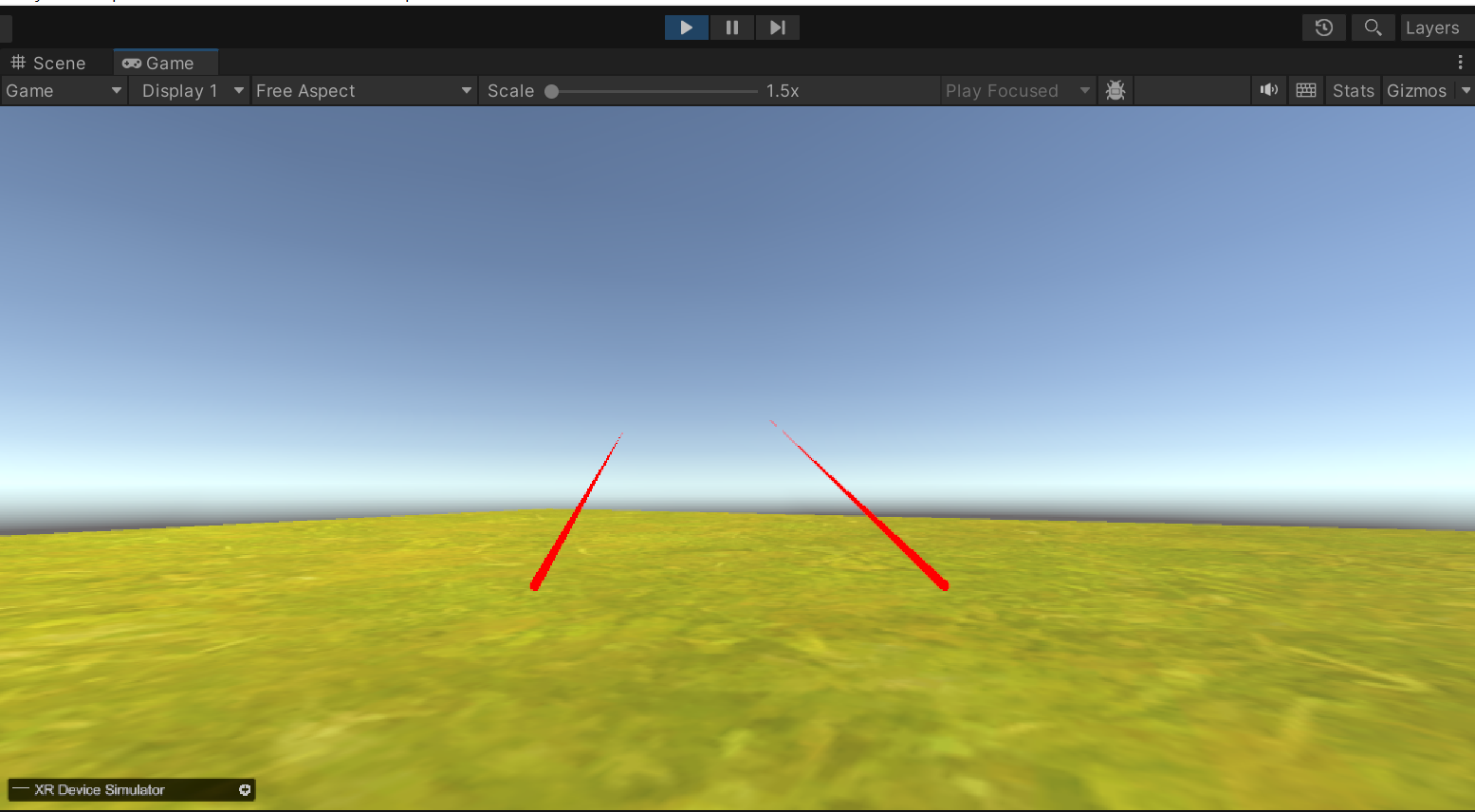
* If you want a more interesting landscape, you can use the **Terrain Tools** (like Raise/Lower Terrain or Paint Height) to sculpt mountains or valleys.

**Step 4: Add Ground Texture:**

* Right-click in the **Project** window and create a **Material** (e.g., for grass or dirt).
* Assign the material to the terrain by dragging it onto the **Terrain** object.
* You can also use a texture from the Unity **Asset Store** for more detailed ground surfaces. For example, search for "Grass Textures" or "Dirt Textures."

**Step 5: Ensure Player Movement:**

* Since you’re using the **XR Device Simulator** with keyboard input (WASD keys), make sure the terrain is large enough for smooth navigation within the scene.



**Task 3:** Add a Skybox **[5 marks]**

● You can use sky presets

● Try to make it more detailed and more interesting

**Step: 1** **Download or Select a Skybox:**

* Open the Unity **Asset Store** and search for “Skybox” or “Skybox Presets.”
* Download any skybox asset that fits your scene. For a more detailed and interesting effect, you can choose something like a **Day/Night Sky**, **Space Sky**, or **Fantasy Sky**.

**Step: 2** **Assign the Skybox:**

* Once the skybox is downloaded, go to **Window > Rendering > Lighting**.
* In the **Environment** section of the **Lighting Settings**, look for the **Skybox Material** field.
* Drag and drop the new skybox material into this field to apply it to your scene.

**Step: 3 Customize the Skybox (Optional):**

* If you want to customize the look, you can modify the skybox’s properties by selecting the material in the **Project** window and changing the settings in the **Inspector** (e.g., tint, exposure, rotation).

**Step: 4** **Preview Your Scene:**

* Run the scene to check how the skybox integrates with the overall environment. Ensure it enhances the atmosphere and provides a more immersive experience.



**Task 4:** Add Environment Objects **[15 marks]**

● You can use Assets to create an engaging Environment

● Create a grabbable objects spawning at random locations

**Step 1: Use Assets to Create an Engaging Environment**

* Go to the **Unity Asset Store** and search for environment-related assets like “Low Poly Environment” or “Dungeon Assets.”
* Once you find suitable assets, click **Download** and **Import** them into your Unity project.
* In the **Hierarchy** window, right-click and choose **3D Object > Create Empty** to organize the objects in a container or add individual objects.
* Drag objects from the **Project** window (under Assets) to the **Scene** view to place them in your environment. Position and rotate them to your preference to create an engaging setup.

**Step 2: Make Objects Grabbable**

* Choose which objects in the environment will be interactable (e.g., gems, pots).
* Add a **Collider** and a **Rigidbody** component to make the object interactable.
* Use the **XR Grab Interactable** component from **XR Interaction Toolkit** to enable grabbing functionality.
* Test the objects by running the scene and using your keyboard controls (WASD and Space/Shift) to move and interact with the grabbable objects.

**Step 3: Randomize Object Spawning**

* In Unity, create multiple spawn points where the objects will randomly appear each time the scene is loaded.
* Configure the spawn points and objects to be placed in various locations to ensure the player has to search for them in different spots each time.

**Environment Setup:**



**Grabbable Object:**



**Task 6:** Configure Lighting and Shadows **[5 marks]**

**Step 1: Adjust Global Lighting**

* Go to **Window > Rendering > Lighting Settings** to open the lighting panel.
* In the **Environment Lighting** section, choose an appropriate **Skybox Material** or set a specific color for ambient light to match the mood of your scene (e.g., a darker tone for a dungeon or brighter for an outdoor setting).
* For more realistic lighting, enable **Baked Global Illumination** and **Realtime Global Illumination** to allow the light to reflect off surfaces, adding depth to your scene.

**Step 2: Configure Object Lighting**

* Use **Point Lights** or **Spotlights** to highlight important objects or areas in your environment.
* Place lights near objects like the fire torches or near the exit to draw attention.
* Modify the **Range** and **Intensity** of each light to ensure they don't over-illuminate or under-illuminate the scene.

**Step 3: Enable Shadows**

* In each light's settings, make sure **Shadows** are set to **Hard** or **Soft** depending on your preference.
* Soft shadows can create a smoother, more natural look, while hard shadows are more defined.
* In **Lighting Settings**, you can adjust the **Shadow Distance** and **Shadow Resolution** to ensure that shadows look crisp and appropriate, without overtaxing performance.
* Run the scene and walk through it using your controls. Check how the lighting and shadows interact with the objects and player movements, and adjust as needed.



**Task 7:** Add Audio **[5 marks]**

**Step 1:** **Add an Audio Source to Objects**:

* Select the object that you want to play sound (e.g., gem, background, or other interactable objects).
* In the **Inspector** window, click **Add Component**.
* Search for and add the **Audio Source** component.

**Step 2: Assign Audio Clip:**

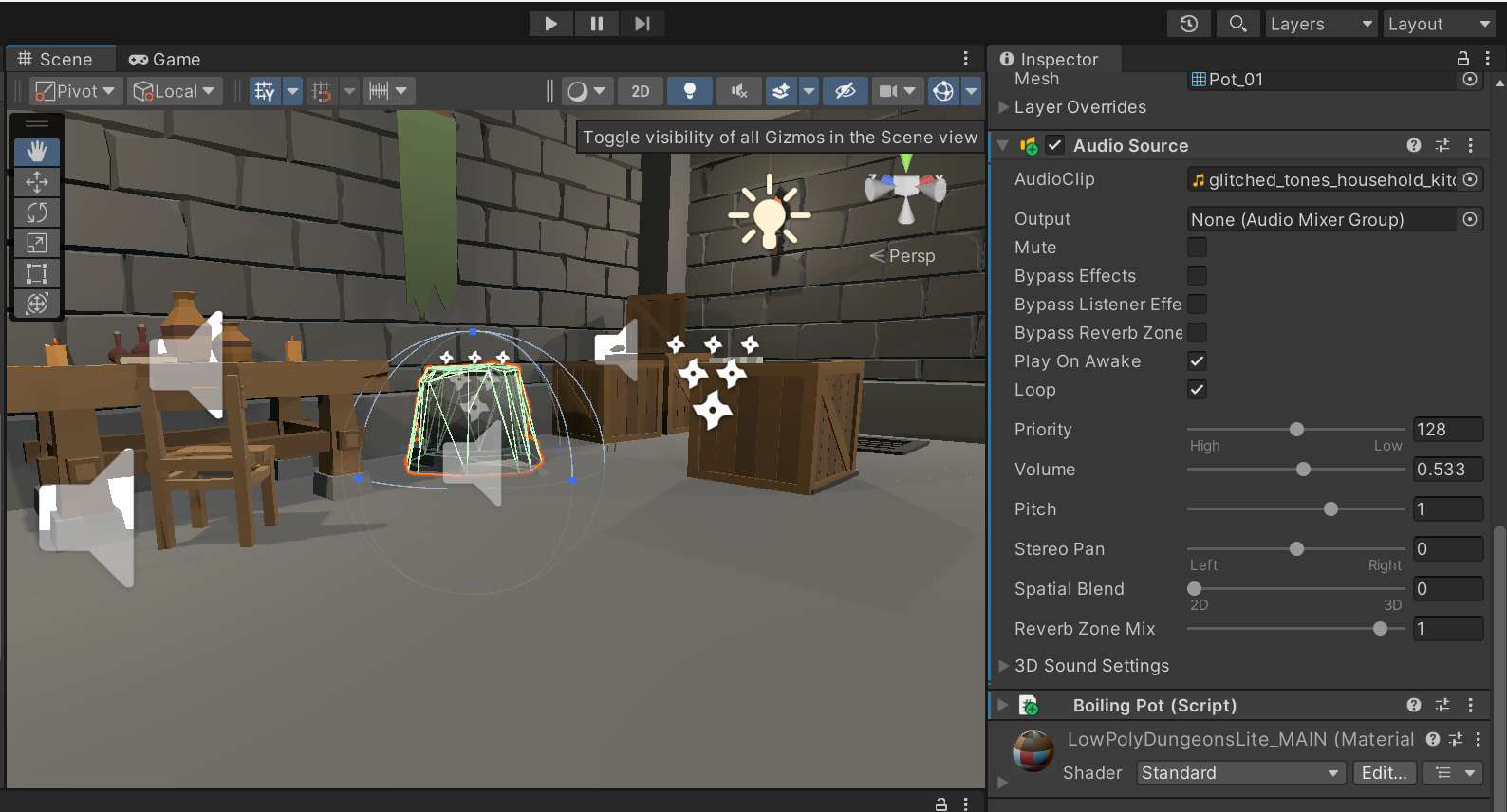
* In the **Audio Source** component, you’ll see a field named **Audio Clip**.
* Drag your desired audio file (e.g., background music) into this field from the **Project** window.

**Step 3: Adjust Audio Settings**:

* **Play on Awake**: If you want the sound to play automatically when the game starts, check this option.
* **Loop**: Check this if you want the audio to loop continuously (useful for background sound).
* **Volume**: Adjust the volume slider to control how loud the sound should be.

**Step 4: Trigger Audio During Events**:

* For interactable objects, such as picking up a gem, write or use a script (or modify the script you’ve already implemented). Ensure that when the object is interacted with, the **AudioSource.Play()** function is called to trigger the sound.
* For a 3D effect, adjust the **Spatial Blend** slider under the Audio Source settings. Move it from **2D** (flat sound) towards **3D** (spatialized sound), so the audio changes based on the player’s proximity.



**Task 8:** Implement Basic VR Interaction **[25 marks]**

● Create a Grabbable Object

● Add Grabbable and Grabber Components

**Step: 1** **Create a Grabbable Object**:

* Select any object you want to make grabbable (e.g., gem).
* In the **Inspector** window, click **Add Component**.
* Search for and add the **XR Grab Interactable** component. This component will allow the object to be interacted with.

**Step: 2** **Add Rigidbody**:

* A **Rigidbody** component is needed to allow physical interactions (like picking up and dropping the object).
* Go to **Add Component** and search for **Rigidbody**.
* Make sure to adjust the Rigidbody settings (e.g., Mass, Drag) for smooth interaction.

**Step 3:** **Add Grabber Components (for hands)**:

* Select your hand controllers (e.g., left or right hand models).
* Add the **XR Direct Interactor** or **XR Ray Interactor** component, depending on how you want to grab objects (direct touch or ray-based interaction).
* Ensure that the **XR Grab Interactable** script on the object and the **XR Interactor** script on the hand interact properly.

**Step 4: Testing the Grab**:

* Press play in Unity’s Game Mode and use your assigned keyboard buttons (e.g., **LSHIFT** for right hand, **SPACEBAR** for left hand) to test if your hands can grab the object.



**Task 9:** Write the VR Interaction Script **[25 marks]**

**Step: 1 Add Grabbing Interactions**

* Ensure your gem objects have an **XR Grab Interactable** component from the **XR Interaction Toolkit**. This will allow them to be grabbed.

**Step: 2 Modify NewBehaviourScript to Work with VR Interactions**

To integrate gem grabbing into this script, you need to trigger the AddGem method when the player successfully places a gem into the correct location (like a socket or box).

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

using UnityEngine.XR.Interaction.Toolkit;

public class GemManager : MonoBehaviour

{

[SerializeField] private int numberOfGems; // Total number of gems required

private int numberOfGemsInSocket = 0; // Current number of gems in the socket

public UnityEvent unlock; // Event to unlock when all gems are placed

// This function is called when a gem is successfully placed

public void AddGem()

{

numberOfGemsInSocket += 1;

CheckForAllGems();

}

// Check if all gems are placed to trigger unlock event

private void CheckForAllGems()

{

if (numberOfGemsInSocket == numberOfGems)

{

unlock.Invoke(); // Unlock when all gems are in place

}

}

// Optional: Function to remove a gem

public void RemoveGem()

{

numberOfGemsInSocket -= 1;

}

// Optional: Handle grabbing interaction

public void OnGrab(XRBaseInteractor interactor)

{

// You could add audio or visual feedback when the gem is grabbed

Debug.Log("Gem grabbed by: " + interactor.name);

}

// Optional: Handle release interaction

public void OnRelease(XRBaseInteractor interactor)

{

Debug.Log("Gem released by: " + interactor.name);

}

}

**Step: 3 Attach VR Interaction Components**

* For each **Gem**, add the **XR Grab Interactable** component from **XR Interaction Toolkit**.
* Also, add a **Collider** (probably a Box or Mesh Collider) and a **Rigidbody** (set it to **Kinematic**) so that the object can be grabbed.

**Step: 4 Detect Placement**

* You can create an **XR Socket Interactor** where the gems need to be placed.
* When a gem is placed in the socket, it will trigger the AddGem() function.

**Step: 5 Test in Play Mode**

* Use your keyboard keys for grabbing (LSHIFT and SPACEBAR) to pick up the gems and place them in the sockets.
* Ensure the unlock event is linked to opening the door in the scene.

**Detect Placement :**



**Unlock Event :**



**Task 10:** Demo application **[5 marks]**

For the demo of my VR application, I have attached a screen-share video to my GitHub repository. The video showcases the full functionality of the game, including the following key features:

1. **VR Environment Setup:** The player moves through the virtual room using keyboard controls (WASD keys) to interact with hidden gems.
2. **Object Interaction:** The player picks up three gems placed around the environment and places them onto a designated box in the scene.
3. **Door Mechanism:** Once all three gems are placed, the door unlocks, allowing the player to escape the room.
4. **Audio Integration:** Sounds are played during interactions, such as when the player grabs a gem.
5. **Lighting and Shadows:** Torches and spotlights are strategically placed in the room, providing immersive lighting effects.