**VR ASSIGNMENT OF GRABBING AND MOVING GRABBABLE OBJECTS IN A VIRTUAL ENVIRONMENT**

**Name: Jainil Manish Parikh**

**Id: 22000235**

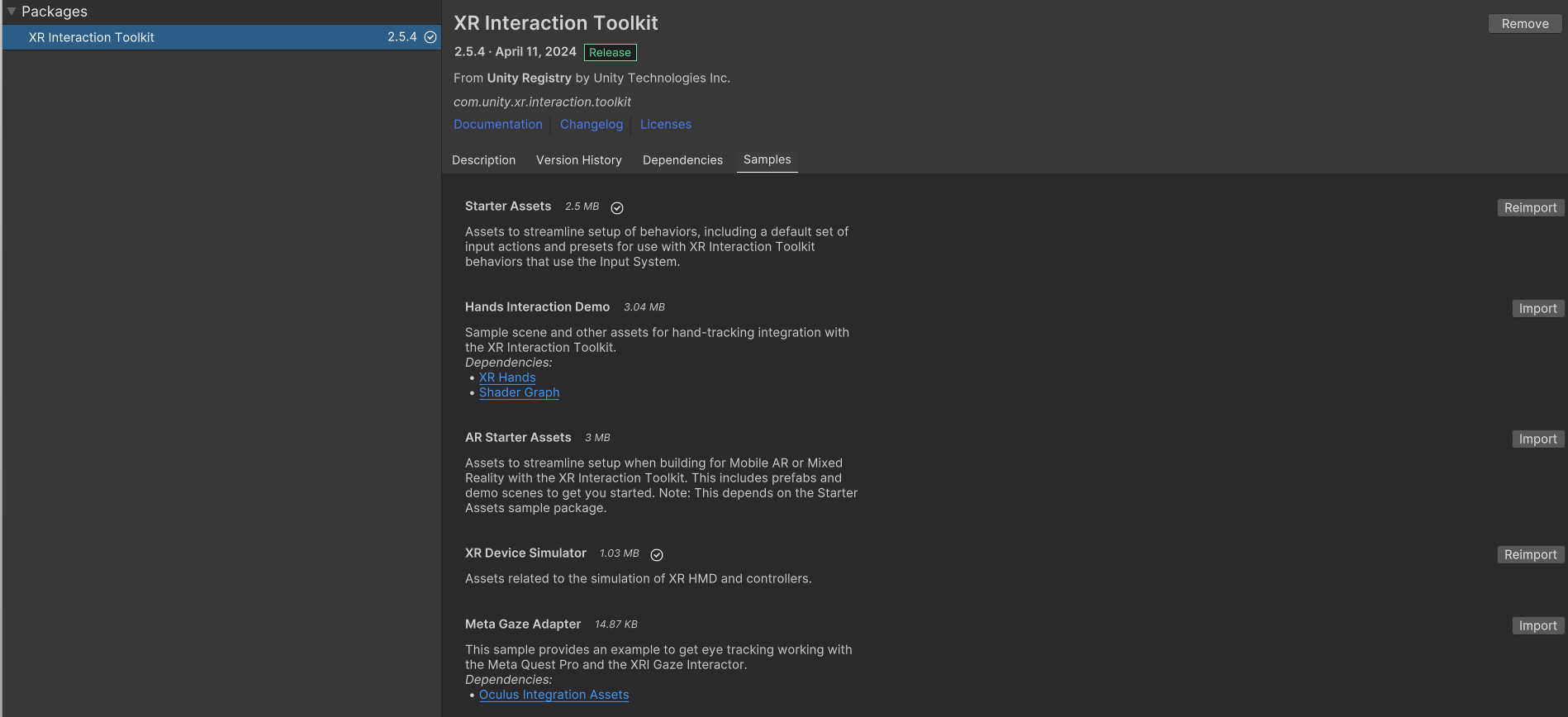
**TYBCA (Batch C)**

**Assignment description:** I created a basic virtual environment of horror room and in it placed three grabbable objects at three different locations. The user is able to grab and move those objects around the environment and place it wherever they want. The entire game can be experienced without vr headset.

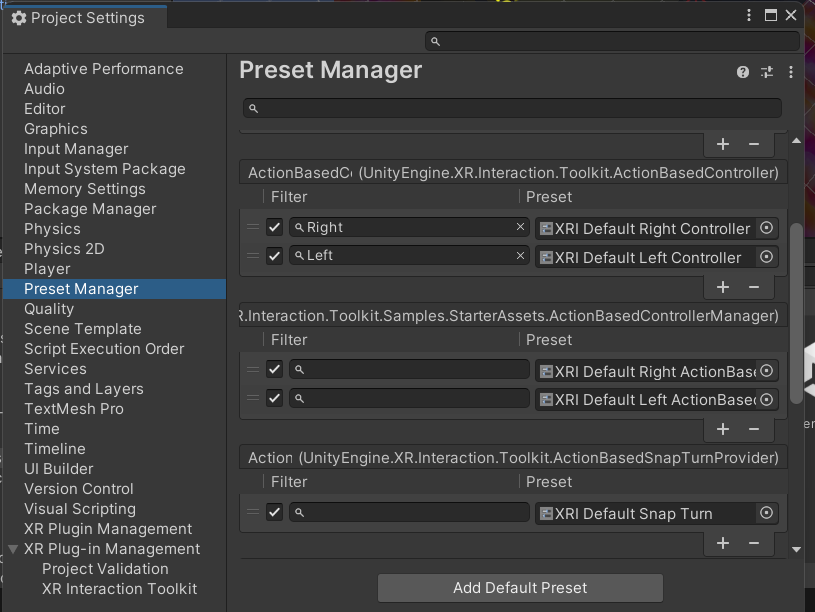
**Task 1: Set up your unity project and configure the VR environment**

1. Created a new project.
2. In windows package manager, in unity registry installed xr interaction toolkit.
3. After installation of XR interaction toolkit, below it, is samples. So, from it import starter assets and xr device simulator.
4. From unity store I imported low poly dungeon lite asset to set up the basic environment and also imported some rocks and ancient weapons like axe and knife. (These were the grabbable objects)
5. From windows package manager I imported these all assets in my project.
6. Right click in hierarchy and select xr-> xr origin (vr).
7. In xr origin vr there are left and right-hand controller so go in starter assets in project window and there enable add to action-based controller as default for all entities.
8. In player settings in preset manager give keywords left and right for both controllers.
9. From project window drag xr device simulator in hierarchy and also delete main camera.

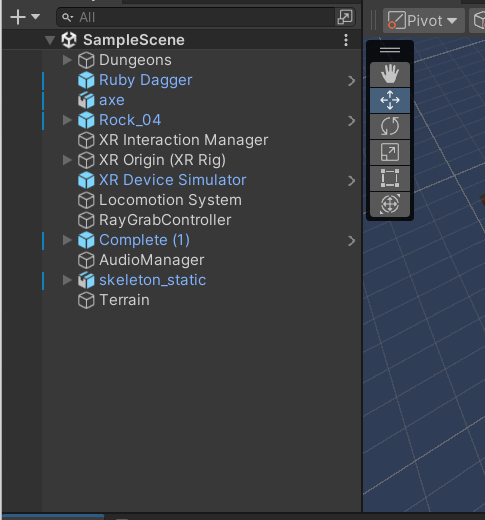
**XR Interaction Toolkit and starter assets and XR device simulator imported.**



**Left and right keywords for left and right controller**



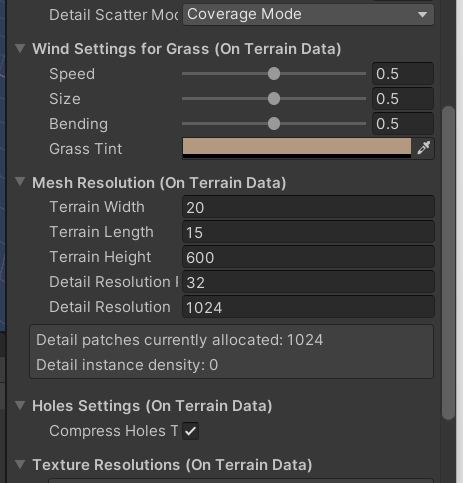
**Imported assets and basic hierarchy window**



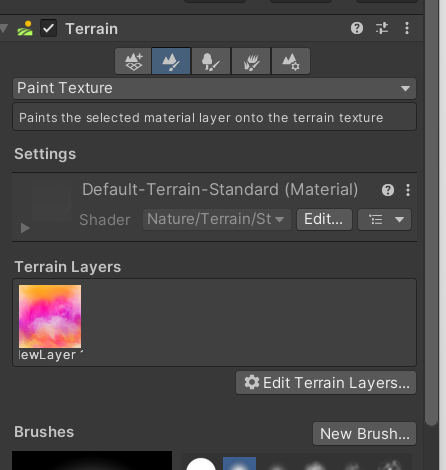
**Task 2: Create ground plane for player to move and also use terrain object for it.**

1. Right click in hierarchy and select 3D object->Terrain.
2. In terrain settings I adjusted the height, width and length of the terrain.
3. I imported one texture from google and used it in paint terrain option to add it to my terrain.

**Terrain settings**



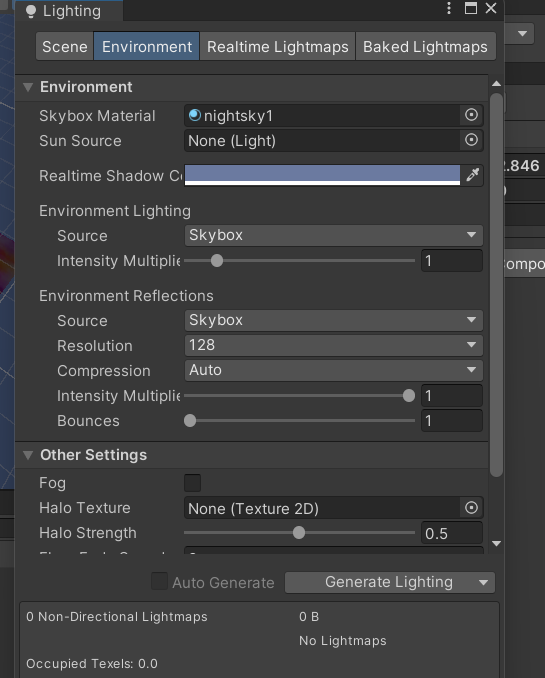
**Terrain Texture**



**Task 3: Add a skybox.**

1. Imported a night sky from unity store and used it in my project.
2. Go to windows->Rendering-> Lighting-> Environment.
3. There an option of skybox material is there. Drag and drop the imported night sky material in it.

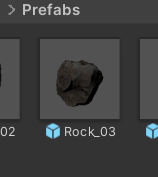
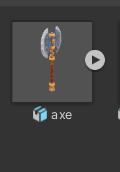
**Night sky added**



**Task 4: Add environmental objects.**

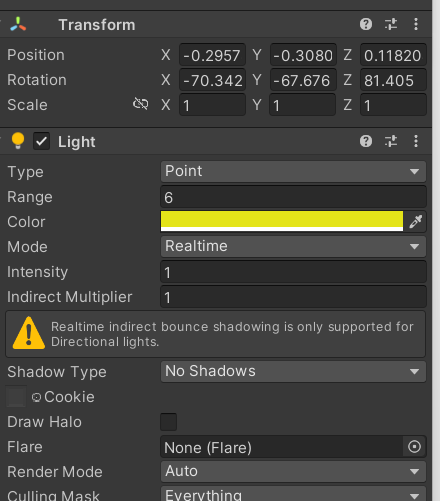
1. Imported rocks and ancient weapons like knife and axe from unity asset store.
2. Imported them in the project through windows->package manager->my assets
3. Adjusted the size of them by scaling and also placed it in the random positions of my dungeon room.
4. Knife was placed on box, axe on table and rock on the floor besides the big pot.
5. Also, my whole dungeon room had three walls so I duplicated one wall and covered the empty space.

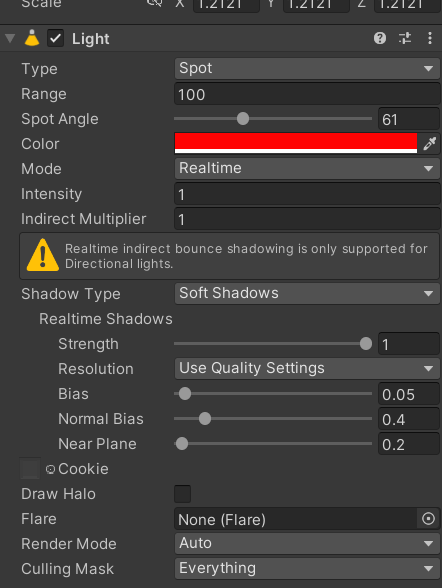
Rock Knife Axe

**  **

**Task 5: Configure lighting and shadows.**

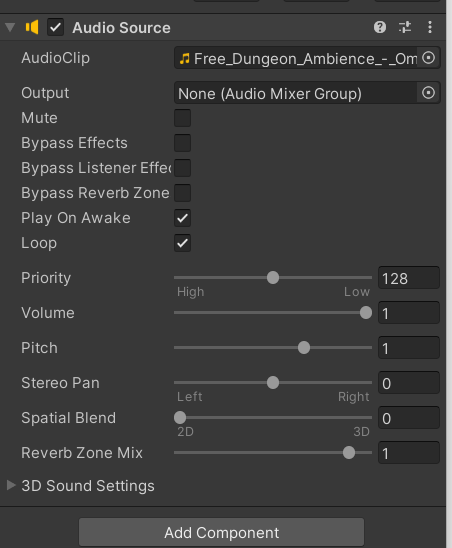
1. I imported lamp from unity assets and used it in my project.
2. But it was just a lamp structure and not original lamp so I added point source of lighting to it.
3. Right click in the hierarchy and go to light. There are various light sources to use.
4. For point light, I used colour as yellow and range of 6 and intensity of 1.
5. Also, I deleted the main directional light so that the vr environment looks like horror room.
6. For shadow effects, I imported a skeleton from unity assets store and used it in my project.
7. My aim was to use a spot light and create a shadow effect of skeleton on the wall behind.
8. I kept the colour of spot light as red and its range as 100 and shadow type as soft shadows.
9. I arranged the spot light in such a way that the light come from little bit downwards and whole skeleton shadow comes on the wall.





**Task 6: Add audio**

1. Created an empty game object named audio manager. I added audio source component to it and in audio clip added an mp3 audio taken from google. Also, the option of loop was checked so that the audio goes on playing repeatedly.



**Task 7 and Task 8: Implement VR interaction**

1. Some basic scripts were already there like xr device simulator.
2. But main scripts for grabbing objects, moving and ungrabbing them and also for ray grab were implemented by me.

**Script for grab object.**

using UnityEngine;

using UnityEngine.XR.Interaction.Toolkit;

public class GrabObjectWithKey : MonoBehaviour

{

public XRRayInteractor rayInteractor;

private XRGrabInteractable grabbedObject;

private XRBaseInteractor interactor;

void Start()

{

interactor = rayInteractor.GetComponent<XRBaseInteractor>();

}

void Update()

{

if (Input.GetKeyDown(KeyCode.G))

{

if (rayInteractor.TryGetCurrent3DRaycastHit(out RaycastHit hit))

{

grabbedObject = hit.collider.GetComponent<XRGrabInteractable>();

if (grabbedObject != null)

{

grabbedObject.interactionManager.SelectEnter(interactor, grabbedObject);

}

}

}

if (Input.GetKeyUp(KeyCode.G) && grabbedObject != null)

{

grabbedObject.interactionManager.SelectExit(interactor, grabbedObject);

grabbedObject = null;

}

}

}

**Ray Grab Script**

using UnityEngine;

public class DualControllerGrab : MonoBehaviour

{

public Transform leftControllerRayOrigin;

public Transform rightControllerRayOrigin;

public float rayLength = 5f;

public LayerMask grabbableLayer;

private GameObject leftGrabbedObject = null;

private GameObject rightGrabbedObject = null;

private Transform leftGrabbedObjectOriginalParent = null;

private Transform rightGrabbedObjectOriginalParent = null;

void Update()

{

HandleControllerInteraction(leftControllerRayOrigin, ref leftGrabbedObject, ref leftGrabbedObjectOriginalParent);

HandleControllerInteraction(rightControllerRayOrigin, ref rightGrabbedObject, ref rightGrabbedObjectOriginalParent);

}

void HandleControllerInteraction(Transform rayOrigin, ref GameObject grabbedObject, ref Transform grabbedObjectOriginalParent)

{

Ray ray = new Ray(rayOrigin.position, rayOrigin.forward);

RaycastHit hit;

if (Physics.Raycast(ray, out hit, rayLength, grabbableLayer))

{

if (Input.GetKeyDown(KeyCode.G))

{

if (grabbedObject == null)

{

grabbedObject = hit.collider.gameObject;

grabbedObjectOriginalParent = grabbedObject.transform.parent;

OnGrab(rayOrigin, grabbedObject);

}

else

{

OnRelease(grabbedObject, grabbedObjectOriginalParent);

grabbedObject = null;

}

}

}

}

void OnGrab(Transform rayOrigin, GameObject grabbedObject)

{

if (grabbedObject != null)

{

Rigidbody rb = grabbedObject.GetComponent<Rigidbody>();

if (rb != null)

{

grabbedObject.transform.SetParent(rayOrigin);

rb.isKinematic = true;

Debug.Log("Object Grabbed: " + grabbedObject.name);

}

else

{

Debug.LogWarning("No Rigidbody found on object: " + grabbedObject.name);

}

}

}

void OnRelease(GameObject grabbedObject, Transform originalParent)

{

if (grabbedObject != null)

{

Rigidbody rb = grabbedObject.GetComponent<Rigidbody>();

if (rb != null)

{

grabbedObject.transform.SetParent(originalParent);

rb.isKinematic = false;

Debug.Log("Object Released: " + grabbedObject.name);

}

}

}

}

**Controller Action Script**

using UnityEngine;

using UnityEngine.InputSystem;

using UnityEngine.XR.Interaction.Toolkit;

public class ControllerInteraction : MonoBehaviour

{

public XRDirectInteractor leftInteractor;

public InputActionReference grabObjectAction;

private XRGrabInteractable currentInteractable;

void OnEnable()

{

grabObjectAction.action.Enable();

}

void OnDisable()

{

grabObjectAction.action.Disable();

}

void Update()

{

if (grabObjectAction.action.ReadValue<float>() > 0)

{

TryGrabObject();

}

else if (leftInteractor.interactablesSelected.Count > 0)

{

leftInteractor.interactionManager.SelectExit(leftInteractor, leftInteractor.interactablesSelected[0]);

}

}

void TryGrabObject()

{

if (currentInteractable != null && leftInteractor.interactablesSelected.Count == 0)

{

leftInteractor.interactionManager.SelectEnter(leftInteractor, currentInteractable);

}

}

void OnTriggerEnter(Collider other)

{

var interactable = other.GetComponent<XRGrabInteractable>();

if (interactable != null)

{

currentInteractable = interactable;

}

}

void OnTriggerExit(Collider other)

{

var interactable = other.GetComponent<XRGrabInteractable>();

if (interactable != null && interactable == currentInteractable)

{

currentInteractable = null;

}

}

}

**Working of Game**

So, in my game the user can move with help of wasd keys and also, he can look around with mouse. Then by holding shift key the left controller is enabled and then we can move it and then using G key we can grab the object and release it. Similarly, by holding space bar right controller is enabled and then user can move it and grab and ungrab the object with G key.

**Whole scene**

