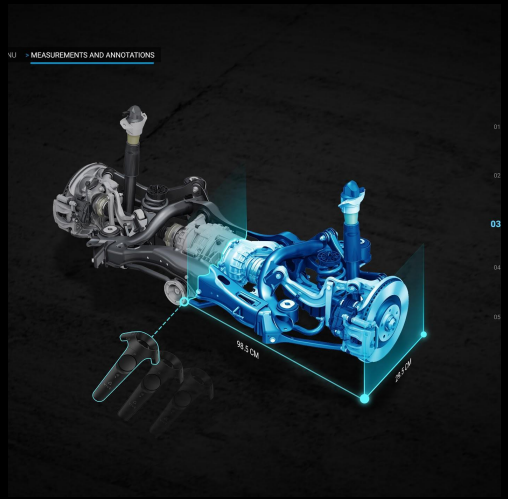
Session 5

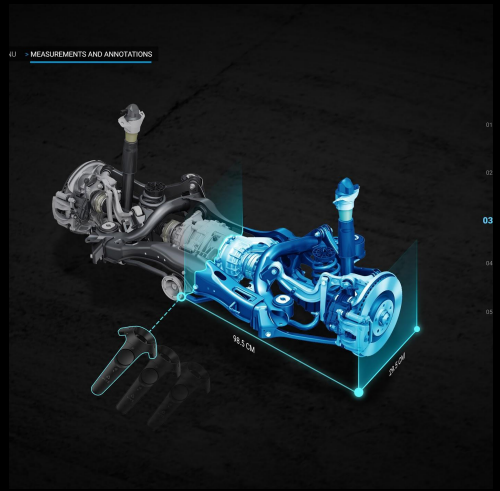
Advanced Interactions in VR

What are some interactions that might require the use of two hands in VR?

Instructor(s)

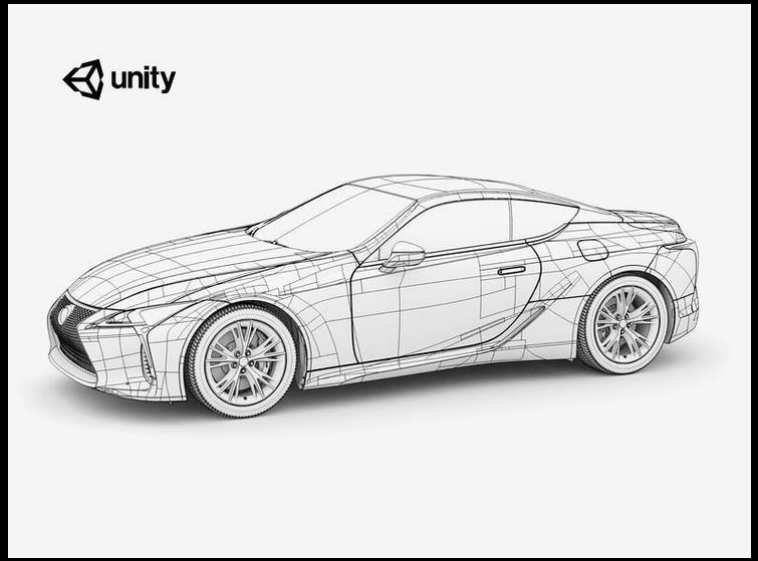


Instructor Name
Instructor Title
Instructor Company



TA Name
TA Title
TA Company

Session Goals



In this session, you'll:

- Use Interactable States and Event Callbacks to perform actions at specific times in the interaction process
- Interpret input from hardware and states within the scene using the XR Interaction Toolkit Debugger Window.
- Extend the XR Interaction Toolkit with scripts to create a variety of objects with advanced functionality.

Session outline

1. Interactable States and Event Callbacks
    - Challenge: Reading Input fr

1. XR Interaction Toolkit Debugger
    - Challenge: Add feedback via primary/secondary buttons

1. Extending XR Interaction Toolkit/XRBaseInteractable
    - Challenge: Two-Handed Scaling

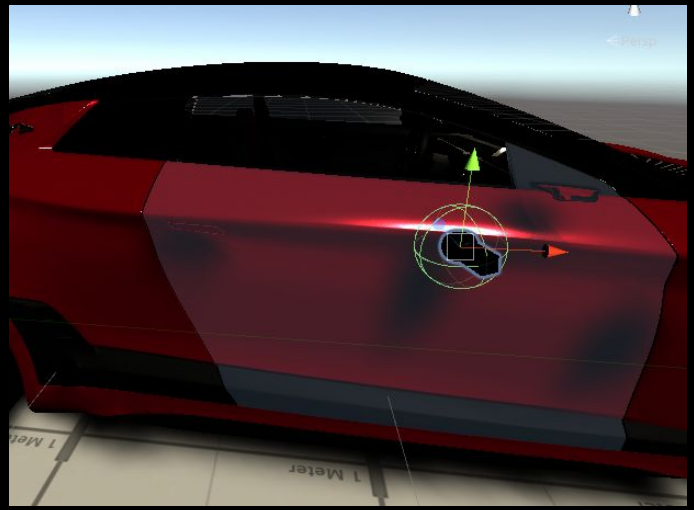
Activity 5: Adding and configuring Two-handed Interactions

Interactable States

Hover

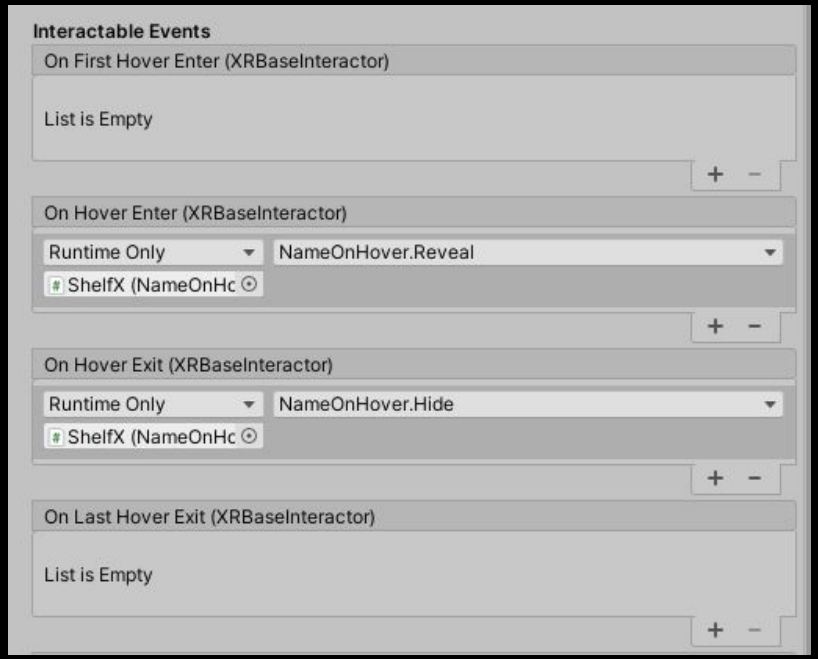
Select

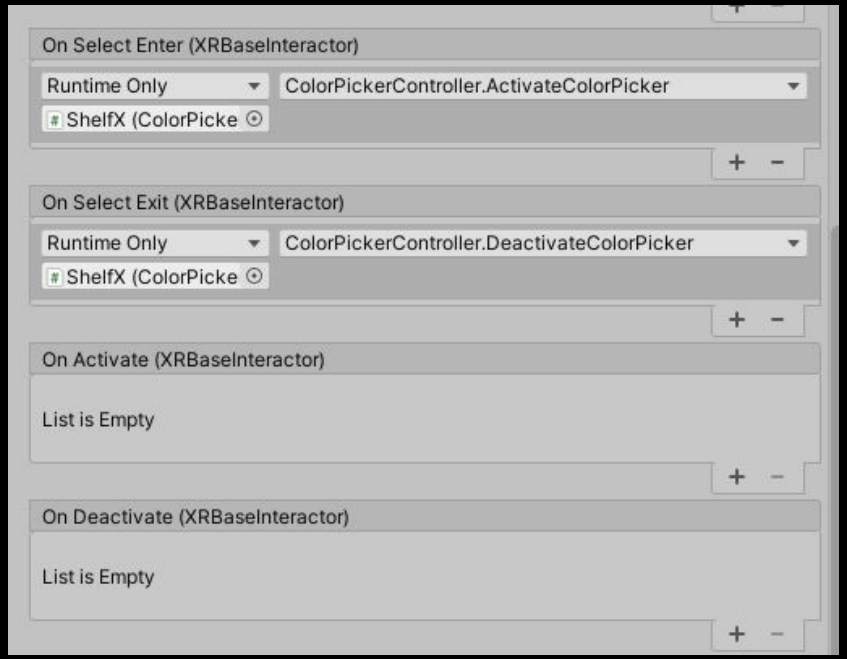
Activate



unity

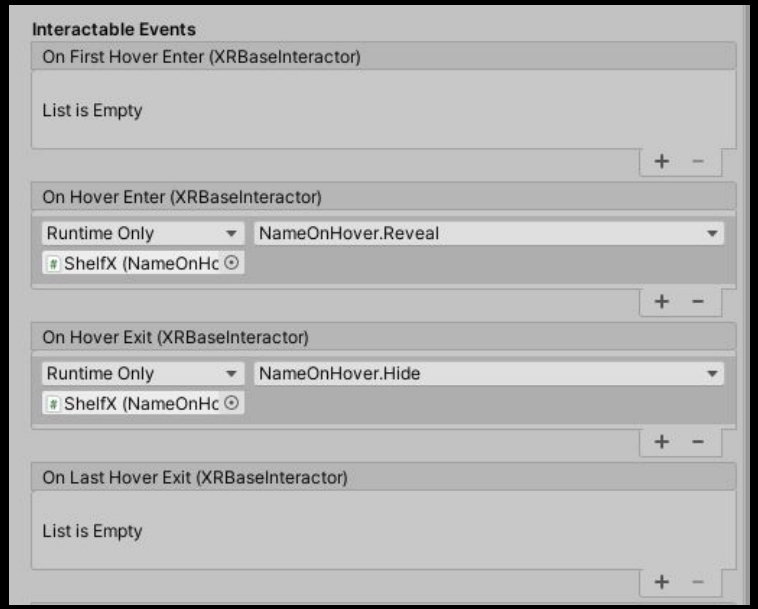
Interactor and Interactable Event Callbacks





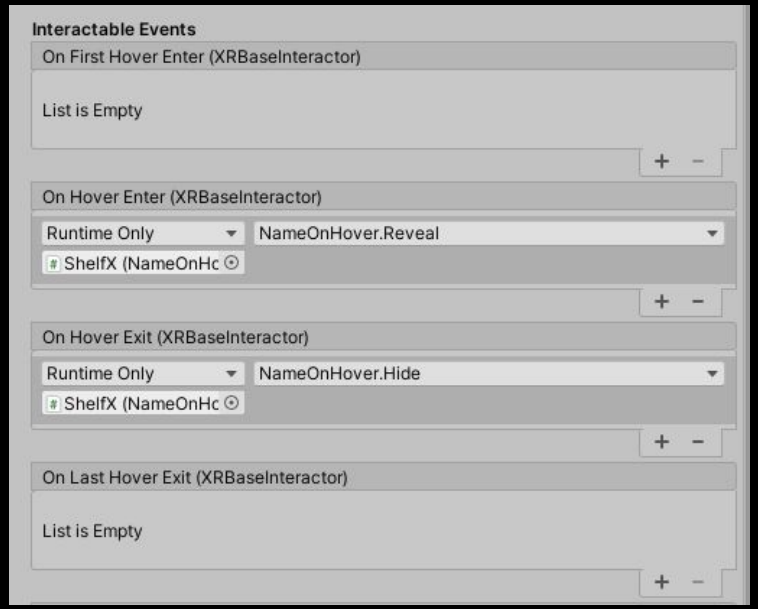
Implement Event Callbacks

1. Choose 1-2 Interactable objects in your scene.
2. Using the Event Callback properties in the Interactable component's Inspector, implement feedback depending on each state.
3. Test the Interactions in your scene



Implement Event Callbacks

1. Choose 1-2 Interactable objects in your scene.
2. Using the Event Callback properties in the Interactable component's Inspector, implement feedback depending on each state.
3. Test the Interactions in your scene



unity

Reading Input from Controllers

```
InputDevice m_RightController;
InputDevice m_LeftController;

[SerializeField]
[Tooltip("The buttons on the controller that will trigger a transition to the Teleport Controller.")]
List<InputHelpers.Button> m_ActivationButtons = new List<InputHelpers.Button>();
/// <summary>
/// The buttons on the controller that will trigger a transition to the Teleport Controller.
/// </summary>
public List<InputHelpers.Button> activationButtons { get { return m_ActivationButtons; } set { m_ActivationButtons = value; } }
```

```
if (m_LeftController.isValid)
{
    bool activated = false;
    for(int i = 0; i < m_ActivationButtons.Count; i++)
    {
        m_LeftController.IsPressed(m_ActivationButtons[i], out bool value);
        activated |= value;
    }
```

XR Interaction Toolkit Debugger

XR Interaction Debugger

Input Devices | Interactables | Interactors

Devices

Name	Role	Type	Value
Devices			
Oculus Rift	Generic		
TrackingState		System.UInt32	63
IsTracked		System.Boolean	True
DevicePosition		UnityEngine.Vector3	(0.1, -0.4, 0.0)
DeviceRotation		UnityEngine.Quaternion	(0.0, 0.1, 0.0, -1.0)
DeviceVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAngularVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAcceleration		UnityEngine.Vector3	(0.0, 0.1, 0.0)
DeviceAngularAcceleration		UnityEngine.Vector3	(2.9, 1.2, 3.8)
LeftEyePosition		UnityEngine.Vector3	(0.0, -0.4, 0.0)
LeftEyeRotation		UnityEngine.Quaternion	(0.0, 0.1, 0.0, -1.0)
LeftEyeVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
LeftEyeAngularVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
LeftEyeAcceleration		UnityEngine.Vector3	(0.0, 0.1, 0.0)

Interactors

Name	Type	Hover Active	Select Active	Hover Interactable	Select Interactable
InteractionManager					
Left Ray Interactor	XRRayInteractor	True	True		Sphere_5_2
Right Direct Interactor	XRDirectInteractor	True	False	Sphere_5_2	
Socket	XRSocketInteractor	True	True	Sword	Sword

Interactables

Name	Type	Layer Mask	Colliders	Hover	Select
InteractionManager					
Sphere_1_3	XRGrabInteractable	-1	Sphere_1_3	False	False
Sphere_2_4	XRGrabInteractable	-1	Sphere_2_4	False	False
Sphere_3_2	XRGrabInteractable	-1	Sphere_3_2	False	False
Sphere_5	XRGrabInteractable	-1	Sphere_5	False	False
Sphere_5_2	XRGrabInteractable	-1	Sphere_5_2	True	True
Sphere_6	XRGrabInteractable	-1	Sphere_6	False	False
Sphere_7	XRGrabInteractable	-1	Sphere_7	False	False
Sword	XRGrabInteractable	-1	Sword,Sword	True	True

unity

Add Feedback via Primary/Secondary Buttons

1. Using a primary/secondary button aside from each Usage trigger in the XRController Class, implement feedback in the scene when that button is pressed
2. Use ControllerManager.cs as a reference for creating a simple script to read input

```
if (m_LeftController.isValid)
{
    bool activated = false;
    for(int i = 0; i < m_ActivationButtons.Count; i++)
    {
        m_LeftController.IsPressed(m_ActivationButtons[i], out bool value);
        activated |= value;
    }
```

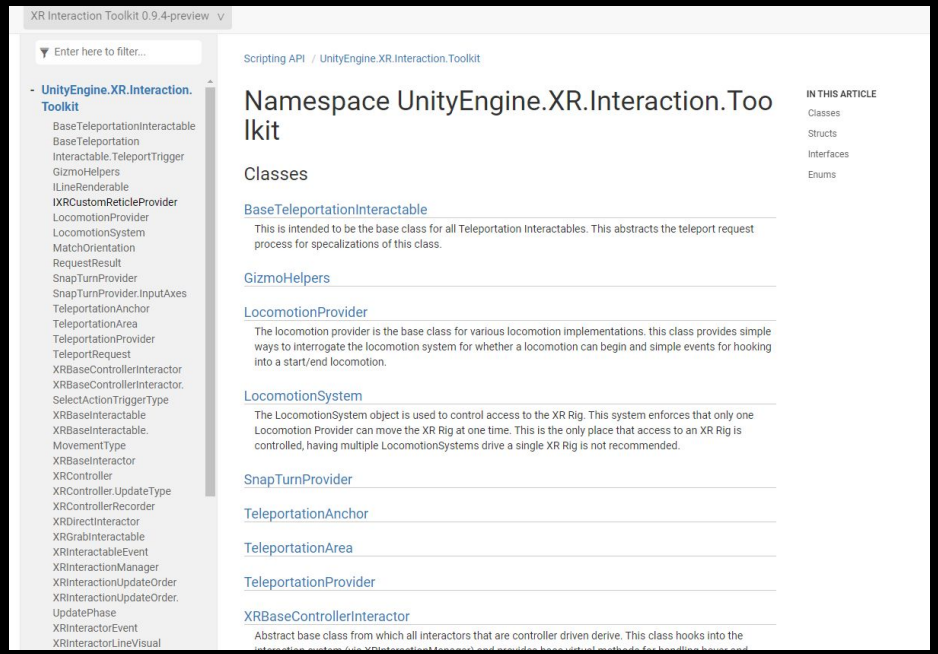
Add Feedback via Primary/Secondary Buttons

1. Using a primary/secondary button aside from each Usage trigger in the XRController Class, implement feedback in the scene when that button is pressed
2. Use ControllerManager.cs as a reference for creating a simple script to read input

```
if (m_LeftController.isValid)
{
    bool activated = false;
    for(int i = 0; i < m_ActivationButtons.Count; i++)
    {
        m_LeftController.IsPressed(m_ActivationButtons[i], out bool value);
        activated |= value;
    }
```

Extending XR Interaction Toolkit

https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/api/index.html



unity

Extending XRBaseInteractable Class

SimpleHandle.cs

Assembly-CSharp | SimpleHandle | OnDestroy()

```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

0 references
public class SimpleHandle : XRGrabInteractable
{
    public float breakDistance = 1.0f;
    public XRGrabInteractable interactable = null;

    0 references
    private void Start()
    {
        interactable.onSelectEnter.AddListener(Show);
        interactable.onSelectExit.AddListener(Hide);
        Hide(null);
    }

    0 references
    private void OnDestroy()
    {
        interactable.onSelectEnter.RemoveListener(Show);
        interactable.onSelectExit.RemoveListener(Hide);
    }

    2 references
    private void Show(XRBaseInteractor interactor)
    {
        gameObject.SetActive(true);
    }

    3 references
    private void Hide(XRBaseInteractor interactor)
```

100 % | No issues found | Ln: 18 | Ch: 57 | SPC | CRLF

unity

Two-Handed Grab Interactable

Inspector

SimpleHandle Import Settings

Open... | Execution Order...

Interaction Manager	None (XR Interaction Manager)
Custom Reticle	None (Game Object)
Attach Transform	None (Transform)
Interactable	None (XR Grab Interactable)

Default references will only be applied in edit mode.

Imported Object

SimpleHandle

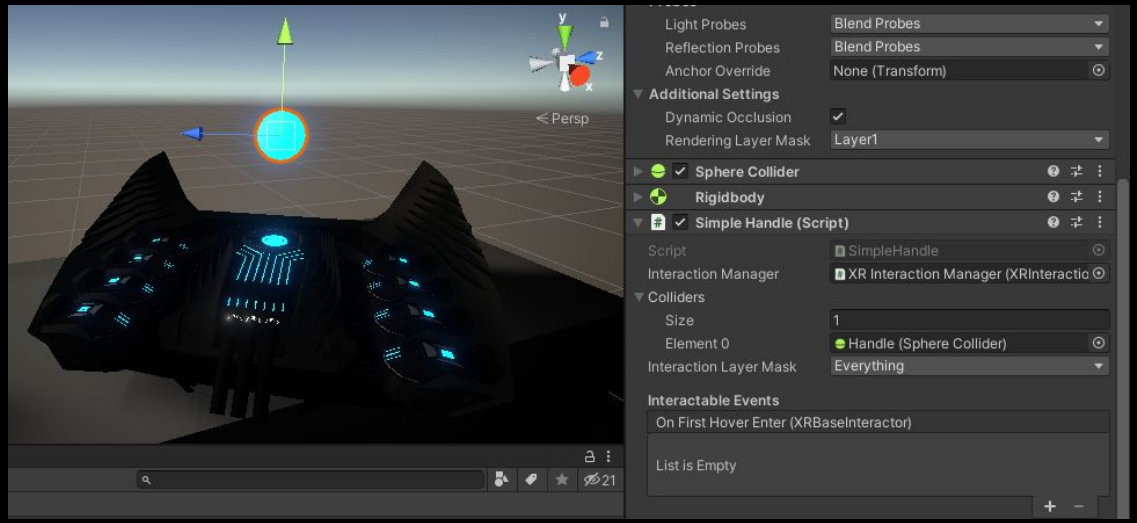
Assembly Information

Filename	Assembly-CSharp.dll

```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class SimpleHandle : XRGrabInteractable
{
    public float breakDistance = 1.0f;
    public XRGrabInteractable interactable = null;

    private void Start()
    {
        interactable.onSelectEnter.AddListener(Show);
        interactable.onSelectExit.AddListener(Hide);
        Hide(null);
    }

    private void OnDestroy()
    {
        interactable.onSelectEnter.RemoveListener(Show);
        interactable.onSelectExit.RemoveListener(Hide);
    }

    private void Show(XRBaseInteractor interactor)
    {
        gameObject.SetActive(true);
    }

    private void Hide(XRBaseInteractor interactor)
    {
        gameObject.SetActive(false);
    }

    protected override void OnSelectEnter(XRBaseInteractor interactor)
    {
        base.OnSelectEnter(interactor);
        SetTracking(false);
    }
```

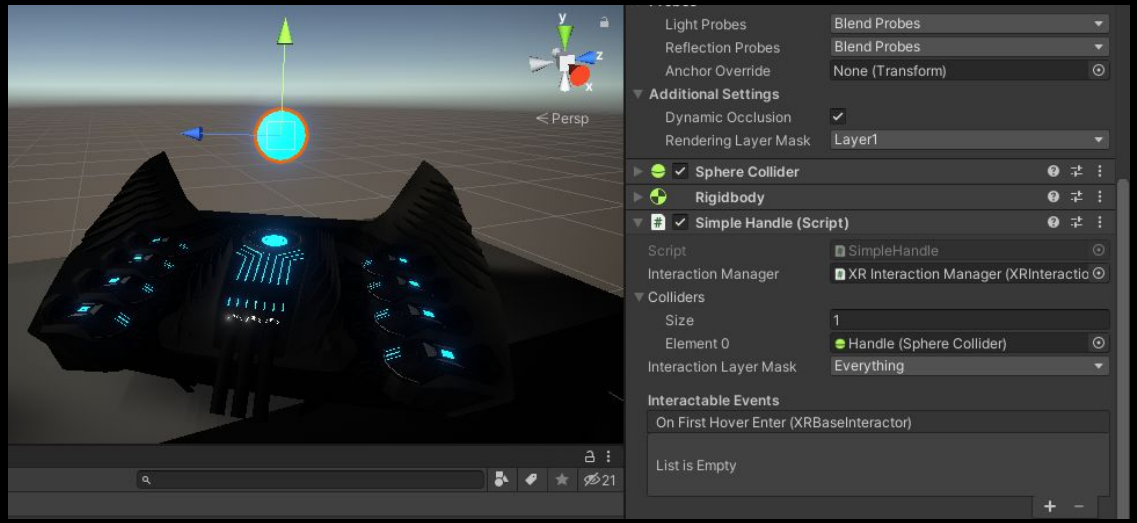
Add Two-handed Interaction

1. Add and configure a Two-handed Grab interactable in your project



Add Two-handed Interaction

1. Add and configure a Two-handed Grab interactable in your project



Activity Session 5

Creating Two-Handed Objects

Generative Art – Made with Unity

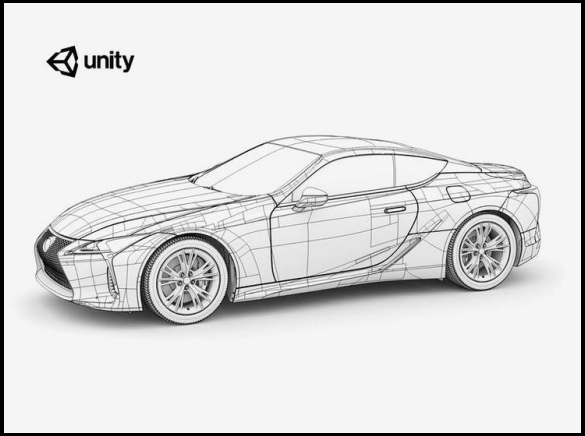
unity

Activity 5 Goals

1. Attend Office hours to Review UI, Locomotion, and Interactions in your project

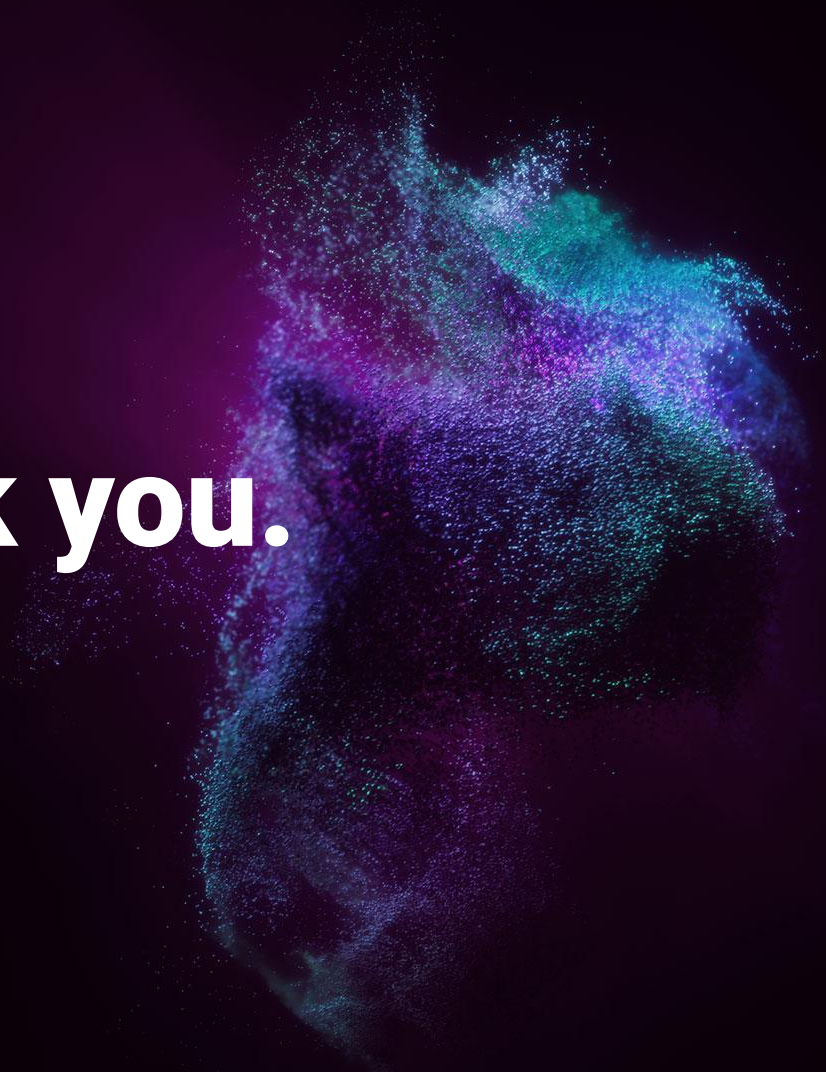
1. Add and configure a few two handed Manipulatable objects to your project

   Bonus: Implement your own scripts to create additional interactions in your project



Feel free to ask questions!

Thank you.



unity