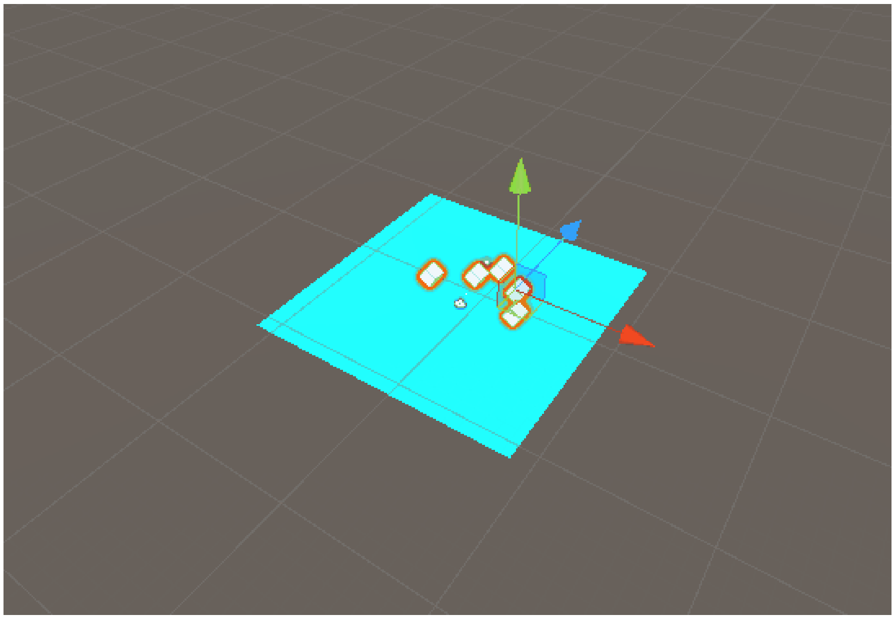
HW 2: Beatsaber Lite!

Created: Apr 4, 2022
Updated: Mar 1, 2023

1. Move some Cubes!

Over the next week and a half, we'll be working on making our first VR Game - a primitive version of Beat Saber! Add your work replicating something like this to a scene in your VR project from HW 0. Get creative and make your scene do something unique! For this first part, we'll be focusing on a simple script that moves blocks towards you, as seen below:



We'll be adding to this over the week, but here are some tips to help you out:
We went through the scripts in class pretty fast, and you might've missed some of the lines. No worries! Here are some of the highlights

Setting Position

- transform.position = new Vector3(0, 0, 1);
    - Sets the transform of a GameObject to a certain position
- transform.position += new Vector3(0, 0, 1);
    - Adds to the current position of the GameObject

Waiting for x seconds:

- public float timeLeft = 2.0f;
    - Put this above the start function
- timeLeft -= Time.deltaTime;
- if (timeLeft < 0) { }
    - Put these in the update loop

Now, any code put between the curly braces will only execute after 2 seconds!

Translate (remember, this acts in local coordinates!)

- transform.Translate(0, 0, 2);

Normalize for Framerate

- transform.Translate(0, 0, - 2 * Time.deltaTime);

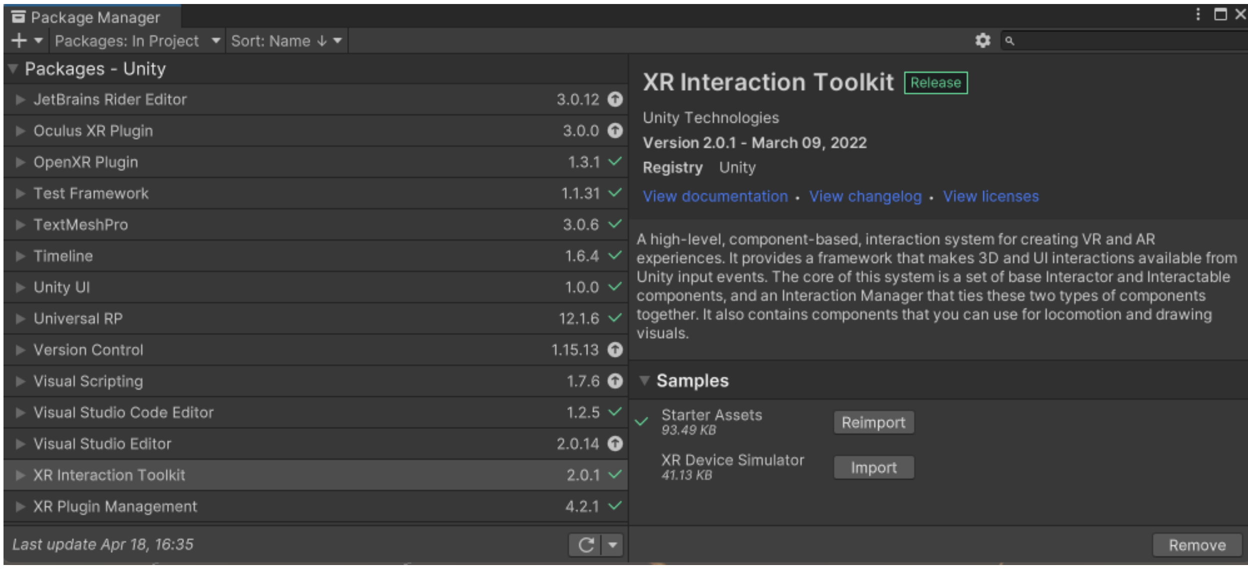
Testing in Editor

- Press the Play button at the top of the Unity window. The Scene view window will be switched to the Game view (from the perspective of the camera in the scene), but you can click on the Scene tab to see everything again.

2. Build to your Quest 2 (Wait until Wednesday, 01/25)

For part 2 of this assignment, you'll build this (very simple) app to your Oculus Quest 2 and test it in your headset! Here's how to do it.

1. Set up your project for VR development.
    a. If you've followed HW 0: Unity Installation, you're all good to go.
2. Enable Developer Mode on your Quest
    a. Follow these instructions to enable Developer Mode. This is necessary to sideload your app to your Quest directly through a USB cable instead of having to upload it to the Oculus Store.
3. Connect your Quest
    a. Connect your Quest 2 to your computer via a USB cable (you can use the charger cable if your computer has a USB-C port).
    b. Turn on your headset, put it on, and use a controller to grant debug access for this computer.
4. Set up controller tracking
    a. In **Window->Package Manager**, find the **XR Interaction Toolkit** (you might need to select "Packages: In Project" in the top left) and click **Import** next to **Starter Assets** to import some default controller actions into the project (Note: you may need to expand the **Samples** category on the right).



Packages - Unity

JetBrains Rider Editor	3.0.12	⬆
Oculus XR Plugin	3.0.0	⬆
OpenXR Plugin	1.3.1	✓
Test Framework	1.1.31	✓
TextMeshPro	3.0.6	✓
Timeline	1.6.4	✓
Unity UI	1.0.0	✓
Universal RP	12.1.6	✓
Version Control	1.15.13	⬆
Visual Scripting	1.7.6	⬆
Visual Studio Code Editor	1.2.5	✓
Visual Studio Editor	2.0.14	⬆
XR Interaction Toolkit	2.0.1	✓
XR Plugin Management	4.2.1	✓

Last update Apr 18, 16:35

XR Interaction Toolkit Release

Unity Technologies

Version 2.0.1 - March 09, 2022

Registry Unity

View documentation • View changelog • View licenses

A high-level, component-based, interaction system for creating VR and AR experiences. It provides a framework that makes 3D and UI interactions available from Unity input events. The core of this system is a set of base Interactor and Interactable components, and an Interaction Manager that ties these two types of components together. It also contains components that you can use for locomotion and drawing visuals.

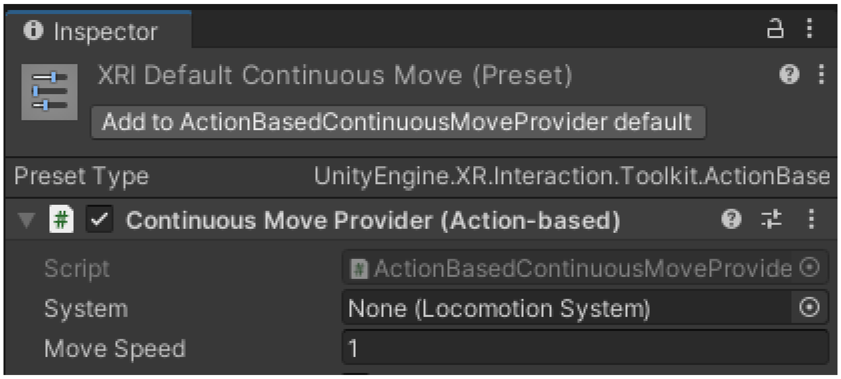
Samples

✓ Starter Assets 93.49 KB

XR Device Simulator 41.13 KB

    b. In your project browser, go to Assets->Samples->XR Interaction Toolkit->YOUR_VERSION->Starter Assets. Click on each preset object (icon has a bunch of sliders) and in the top of the Inspector click **Add to ActionBasedContinuousMoveProvider**

default (do this for each of these sample action presets)



XRI Default Continuous Move (Preset)

Add to ActionBasedContinuousMoveProvider default

Preset Type UnityEngine.XR.Interaction.Toolkit.ActionBase

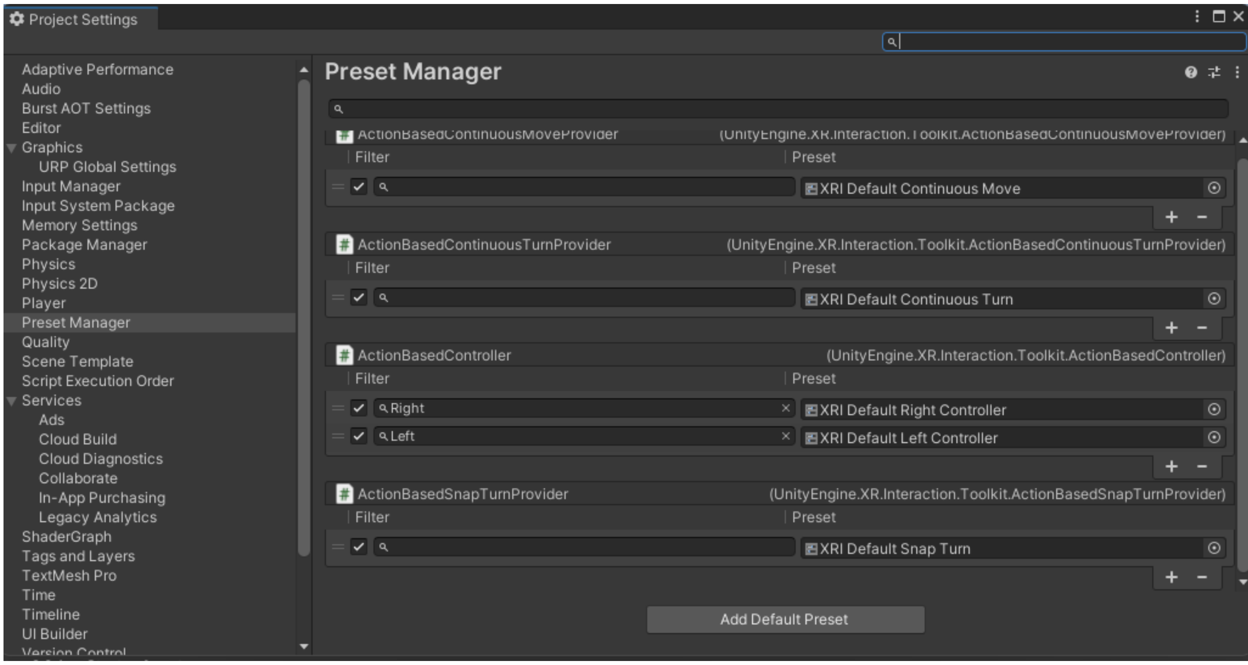
Continuous Move Provider (Action-based)

Script ActionBasedContinuousMoveProvide

System None (Locomotion System)

Move Speed 1

c. In Edit->Project Settings->Preset Manager, find the **ActionBasedController** presets and add the text **Right** and **Left** in the text boxes corresponding to the actions.



Preset Manager

ActionBasedContinuousMoveProvider (UnityEngine.XR.Interaction.Toolkit.ActionBasedContinuousMoveProvider)

Filter	Preset
	XRI Default Continuous Move

ActionBasedContinuousTurnProvider (UnityEngine.XR.Interaction.Toolkit.ActionBasedContinuousTurnProvider)

Filter	Preset
	XRI Default Continuous Turn

ActionBasedController (UnityEngine.XR.Interaction.Toolkit.ActionBasedController)

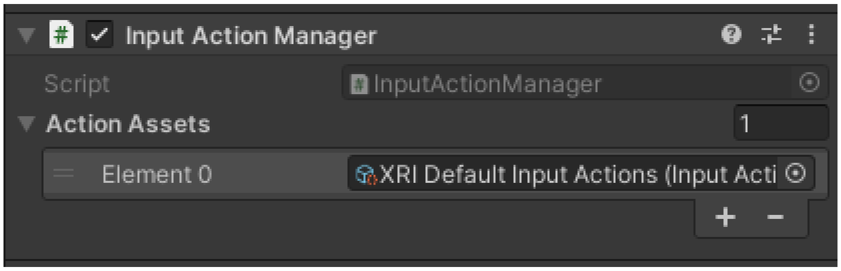
Filter	Preset
Right	XRI Default Right Controller
Left	XRI Default Left Controller

ActionBasedSnapTurnProvider (UnityEngine.XR.Interaction.Toolkit.ActionBasedSnapTurnProvider)

Filter	Preset
	XRI Default Snap Turn

5. Add an XR Origin to your scene
    a. In Unity, right click in your hierarchy and click “XR->XR Origin (Action-based)” (**important**: don’t click the non-action-based “XR Origin”). This will add several gameobjects to the scene which enable tracking your headset and controllers, among other things.
    b. Move your new XR Origin game object to be in a spot in front of where the blocks go.

    c. Select your XR Origin object in the hierarchy. Add an **Input Action Manager** component in the Inspector, expand its Action Assets list, add an element with the + icon, and drag in or use the target button to select your XRI Default Input Actions from the folder with all the action presets from step 4 above.



Input Action Manager

Script: InputActionManager

Action Assets: 1

Element 0: XRI Default Input Actions (Input Acti

6. Set up your build settings
    a. In Unity, open File->Build Settings.
    b. Add your scene with the XR Origin to the build by dragging it in to the **Scenes In Build** box from the Project Browser or clicking **Add Open Scenes** and drag it to the top of the list if there are more than 1 scene there so that it's the scene which opens first.
    c. If your platform isn't set to **Android** (there should be a little Unity cube icon next to **Android** on the left), select **Android** in **Platforms** and click **Switch Platform.**
    d. Under **Run Device** in Project Settings, select your Quest 2. If it doesn't show up, press **Refresh**, ensure you've followed steps 2. and 3. above, then ask for help on Ed.
7. Build and Run
    a. Click **Build and Run**, then select a name and a location (maybe "[PATH_TO_YOUR_PROJECT]/Builds") for the .apk file (Android app package) to build to your Quest and start the app.
    b. When it finishes (it will take slightly longer the first time, probably 2-15 minutes depending on your computer and the assets in your project), you'll be able to unplug your Quest 2, put on the headset, and test your first VR app! Try looking around and watching the cubes past through you.
    c. You can also find your app later on the Quest by opening the Library, clicking the dropdown in the top-right, and selecting

Unknown Sources. Do your cubes remember where they were if you close the app and reopen it?

8. When something goes wrong…
    a. Post on Ed or comment on this doc to get help and help your classmates!

If you'd prefer to see a video going over most of these steps, check out this video by Justin P Barnett, a Unity VR YouTuber. Note that Justin uses some different Project Settings that us. Feel free to test out different settings, but defer to the settings we specified in HW 0 as a ground truth.

For testing your project on Quest in PCVR while plugged into a Windows computer with a solid graphics card, switch your platform to Windows, Mac, Linux in Build Settings then see the end of this video. If you have a SteamVR headset like a Valve Index/HTC Vive or a Windows Mixed Reality like an HP Reverb, simply turn on your desktop runtime as if you were going to play a VR game, ensure you've enabled OpenXR support, and press the Unity play button.

Submission Instructions

Submit a screen recording of **your moving cubes** to the Canvas assignment by class on Monday (April 11th)! Try to record from your Quest if you can build to it (here's a video about Quest 2 screen recording), otherwise, record from your Unity editor.