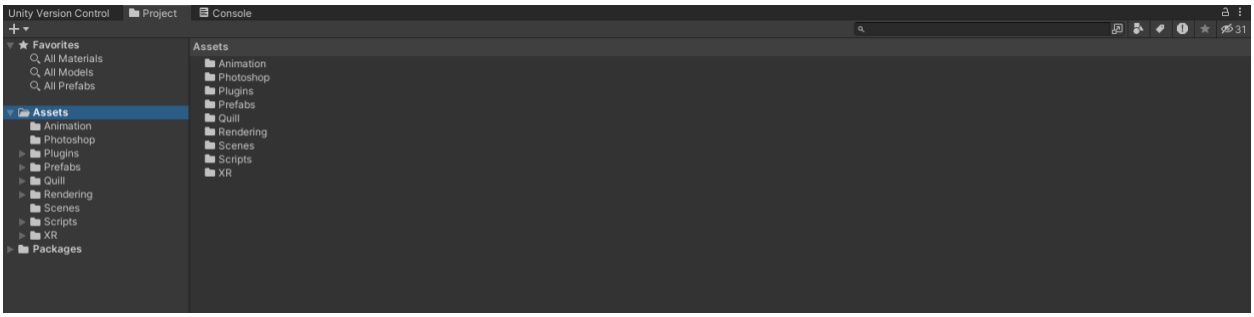
Starting with Unity

1. Start with a Unity provided AR template project.
2. Add XR Plug-in via project settings
3. Add TouchManager game object
4. Add Touch Manager script to TouchManager game object
    a. Dev writes this script
5. Install new input system
6. Add Player Input component to TouchManager game object
7. Add AR Tracked Image Manager component to AR Session Origin game object
8. Add Reference Image Library in Serialized Library portion of AR Tracked Image Manager component
    a. Dev creates a reference image library in the Unity project folder
9. Add Multiple Images Tracking Manager script to AR Session Origin
    a. Dev writes this script
10. Only other game object in the hierarchy would be debugging tools that are turned off
11. Set build settings to projected build platform.

Unity assets folder structured as so:

- Animation
    - Animation Clips
    - Animator Controllers
- Photoshop

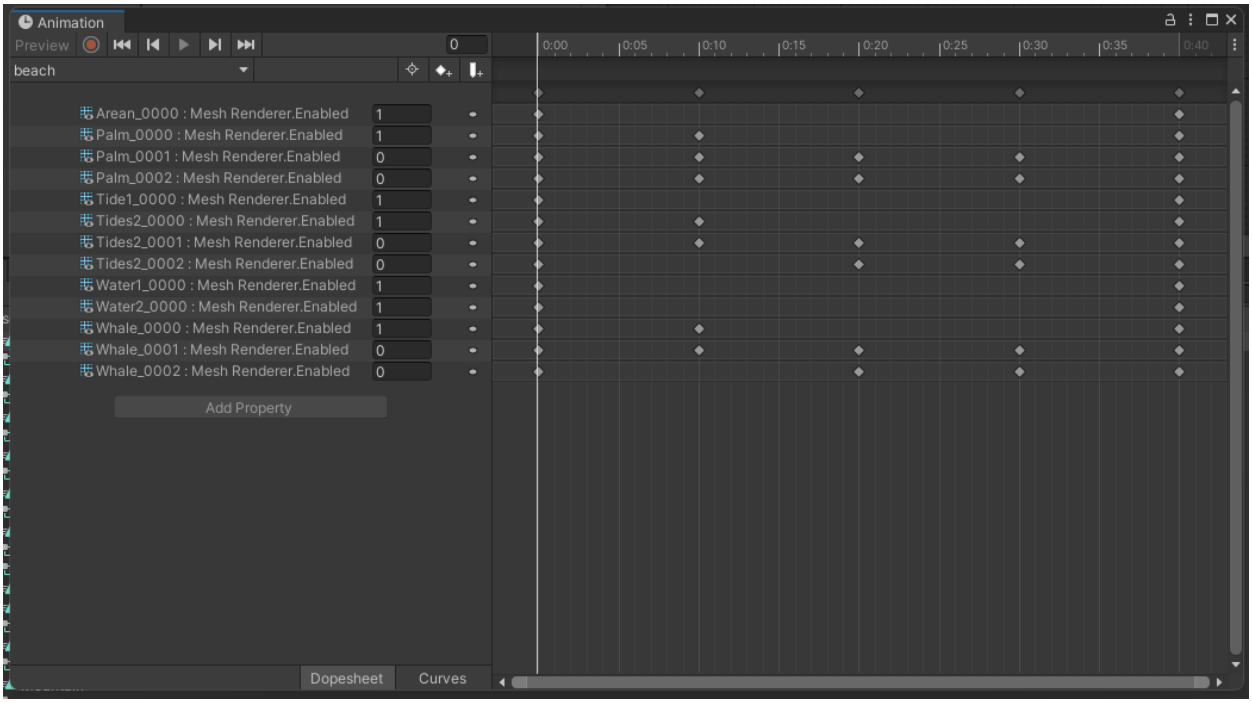
    - Texture tiles (psd & png)
- Plugins
    - TextMesh Pro
    - Unity AR example assets
- Prefabs
    - Tile prefab that each have two or more options
    - Prefabs of each individual option (Quill asset modified with Animator component and Animator Controller, and positioned/scaled at origin)
- Quill
    - Imported FBX files from Quill with QuillVertex shader applied.
- Rendering
    - QuillVertex shader & material placed here (downloaded from internet, created material and placed shader on it).
- Scenes
    - Automatically created with the template, our scene is placed here.
- Scripts (we've written)
    - MultipleImagesTrackingManager.cs
    - TogglePrefabs.cs
    - TrackObject.cs
    - TouchManager.cs
    - Inputs.cs (generated via Inputs (created by right clicking and creating new Input Actions)
        - In Inpsector, generate C# class (this creates Inputs.cs)
- XR
    - Reference Images
    - Reference Image Library
        - Add image, name it, specify size
        - Uncheck do not keep texture at runtime.
    - Other folders that were automatically added



<u>Importing Quill asset and preparing for Unity AR use</u>

1. Drag FBX models into Quill folder.
2. In Inspector, choose not to convert units (under model tab)
3. Place the model into the scene.
4. Select the game object, add Animator component

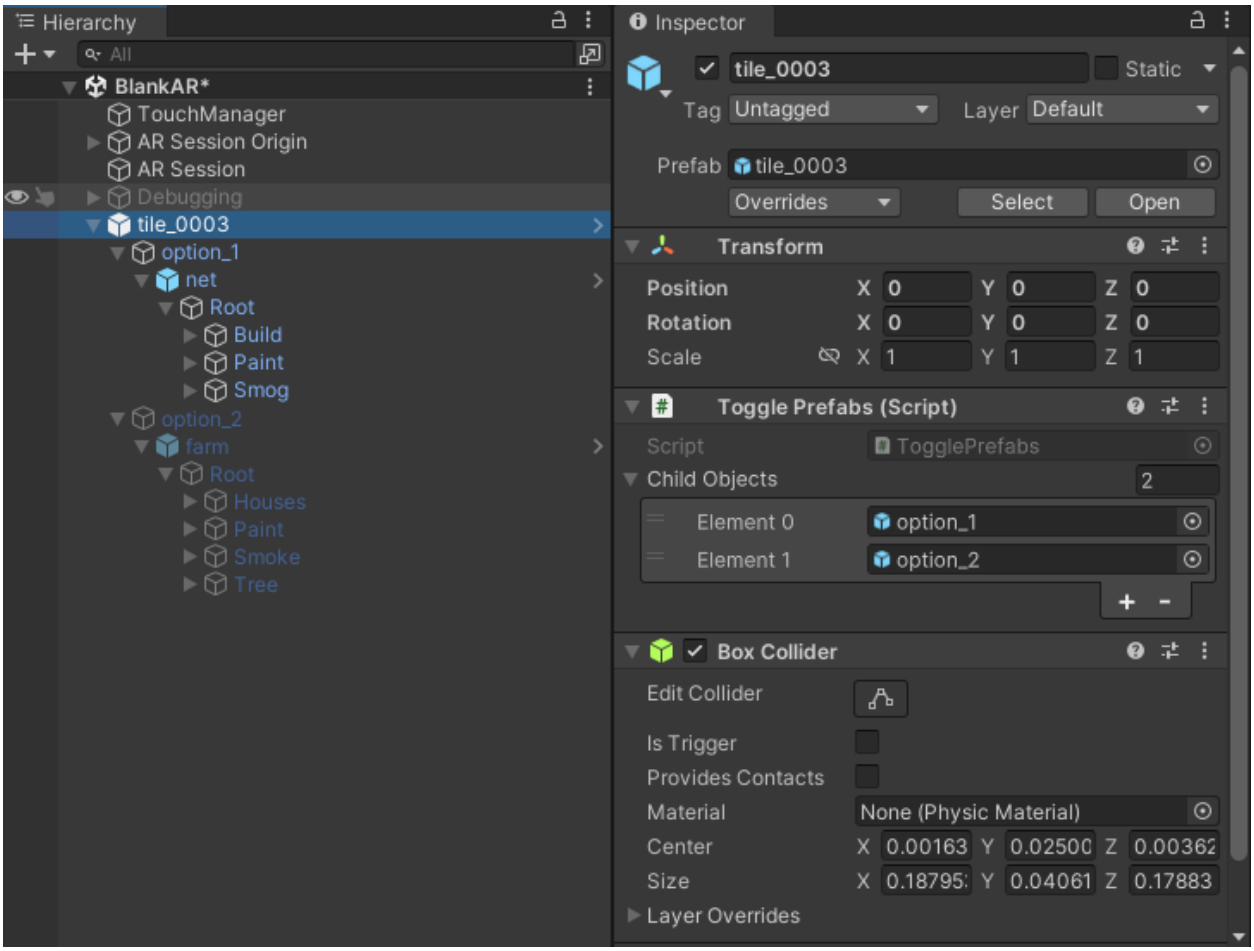
5. In Animaton folder, right click, create new Animator Controller and Animation (clip)
    a. Name them in alignment of the imported FBX name
        i. Make sure to have neat naming structures
6. In the game object's inspector panel, add the Animator Controller asset where it is asking for it in the Animator component.
7. Open up Animator controller by double clicking on it, drag the animation clip into the space. It should have an arrow connecting it from 'Entry' to the clip you just placed in.
8. Go back to the Scene view. At this point, the game object which you created from an FBX model should have an Animator component, which has an Animator Controller with an animation clip inside.
9. Scale the object to .1 (might need to fudge this a bit later) and move the object to the origin.
10. Double click the Animation Clip associated to this game object (located in the Animation folder).
    a. An animation window should open.
    b. Add 'Mesh Renderer.Enabled' properties for each part of the Quill FBX (now game object) that you'd like to appear.
    c. In the dopesheet, select which is enabled (1), or disabled (0). This is creating the animation.
    d. Press play to test (watch in the scene view).
    e. While the Animation Clip is selected, in the inspector, enable 'Loop Time'.
11. Right click the game object in the hierarchy and Set as Default Parent
12. Drag the game object from the Hierarchy into the prefabs folder.



Creating Tiles

1. In the scene, create an empty game object.

    a. Within that game object, place two empty children game objects.
2. The parent game object should be called ‘tile_0001’ (or whichever number tile it is) and the children should be named ‘option_1’ and ‘option_2’
3. Place a prefab (made from the Quill FBX in the previous step) into option_1
4. Repeat the last step but with a different prefab for option_2
5. In the Inspector, uncheck (or turn off) option_2 and all of the children within it.
6. Add a box collider to the tile_0001 (parent object). Make sure to cover roughly the size of the prefabs, this will be the box that detects touch via a raycast to initiate a swap.
7. Add Toggle Prefabs (script) component to tile_0001 (parent object).
    a. In the Inspector, place the child object options (option_1 & option_2). This will let the script know which option should be toggled per amount of touches on the screen.
8. Once completed, pull this game object into the prefabs folder to make this tile game object a prefab.



Preparing (new) Input System for AR

1. Install or check to see if package is installed.
2. Create Input folder within the Scripts folder
3. Create Input Actions (right click, create)

    a. Double click the created Input Actions and set actions (I followed samyam tutorial listed below)
4. Create TouchManager.cs
    a. write as needed: I followed this samyam tutorial and asked ChatGPT for some help)
        i. https://www.youtube.com/watch?v=4MOOitENQVg
5. Window > Analysis > Input Debugger
    a. Click Options dropdown
        i. Click Simulate Touch Input from Mouse or Pen
6. Test and debug within Unity using the Console for messages.