# XR App 2

In this assignment, you will build on your XRA1 and turn it into a design app. You will implement various UIs and add some key functionality and novel interactions to your app, as described below. You should refer to the demo video for a more detailed description of the requirements.

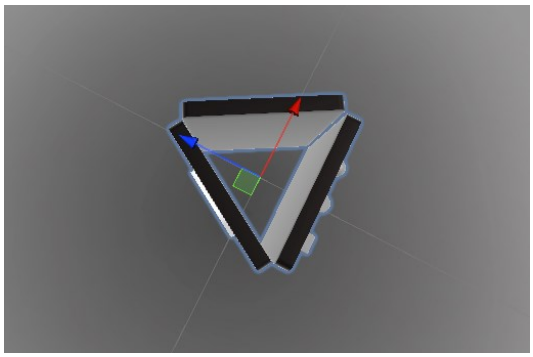
## Environment

Your environment should use a different skybox and floor than the ones used in the template project posted on Canvas. The more customizations you make the better. At the very least, your environment should resemble the one shown in the demo, with the platform included. All shapes should be placed on the wrist menu (you need to use at least six different shapes from the VR Interactable Objects package provided on Canvas). The idea is that users will select the shapes from the menu.

## Controllers (2pts)

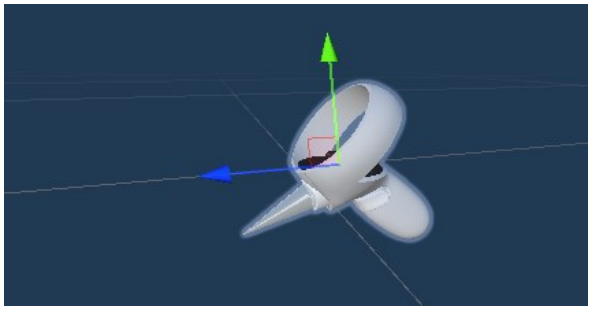
The left controller will be used for wrist menu and should have no interaction capabilities. The wrist menu is a 3D object that can be rotated based on the left joystick's horizontal movement. **Hint:** for this rotation to work, you want to carefully structure your menu. The catch is to ensure that the reference point is at the bottom center of the menu.


Here is the top-down view of the menu shown in the video:



The right controller will be used for direct interaction only and shouldn't have any teleportation capabilities. You will need to add a pointer to the right controller (I'd duplicate the prefab and make a custom one for this assignment. You don't modify to use the original right hand controller prefab.) All direct interactions should

be invoked when the tip of the pointer hovers/touches the objects. I used the cone shape available in the basic shapes packages that we've been using in class. Here is an example of what this could look like:



## Feedback (2pts)

Your app should provide carefully-designed feedback to increase users' sense of presence in the environment. For controllers, use appropriate Hover and Select feedback by playing appropriate sound clips and by applying vibrations to the controllers. For instance, vibrations for hovering should be less intense and shorter than those applied when users select an object.

All interactable objects should play a sound clip when they collider with anything, similar to what we did in class. They should all have a somewhat bouncy material, so they look more realistic when they are dropped on to a solid object.

## Wrist Menu (14pts)

Your app should implement the 3D wrist menu shown in the demo. It should have the three panels and support the functionality described in the video (rotation based on the left joystick's horizontal movement).

1. **Shapes Panel**
   - Must be the default face shown to the users.
   - Must include six different shapes that can be added to the scene.
   - When hovered, the shapes should be highlighted white.
   - When selected, the shape should be added to the scene as shown in the video (all effects should be mimicked). Make sure the newly added copies of the shapes are properly scaled (the ones in the menu will be tiny).
   - Note that the shapes in the menu aren't grabbable, but the newly added copies are.

2. **Colors Panel**
   - Must include at least four different colors that can be selected.
   - When hovered by the pointer, the colors should be automatically applied to the pointer attached to the right controller without any key input.
   - Color icons should provide some sort of visual feedback to indicate that they are being hovered. Do your best to mimic the effect shown in the video (the background color changes).

3. **Menu Panel**
   - Must include the Reset button.
   - When pressed, the Reset button loads the same level.
   - We shouldn't need to enter the scene name to load. Instead, your code should automatically identify the current scene and load it.

## Interactions (7pts)

You will add a number of interactions to your interactable objects, as explained below:

1. **Highlight Object**
   - When hovered by the right controller's pointer, objects should be highlighted by matching the color applied to their material to that of the pointer's current color.
   - When the object is no longer hovered by the pointer, the original color should be reapplied.

2. **Color Object**
   - If the user presses the trigger while an object is highlighted, the current color of the pointer should become the hovered object's current color.

3. **Clone Object**
   - Users should be able to clone a selected object by pressing the A button (primary button) on the right controller. This interaction should clone the current version of the object, not a pre-determined prefab.
   - After the cloning operation, the original object must remain intact.
   - The newly cloned object should respond to physics (except for gravity), as shown in the video.
   - When the cloned object is at a certain distance (say 10 units) away from the player, it should be destroyed. The corresponding visual and audio effects should be applied as shown in the demo.

4. **Destroy Object**
   - Users should be able to destroy a grabbed object by pressing the B button (secondary button) on the right controller.
   - When destroying an object, a visual effect should be played (you can use the particle effect provided and modify it however you see fit), along with an appropriate sound clip.

5. **Resize Object**
   - Users should be able to scale up and down a selected object by using the joystick on the right controller. Moving the joystick up (North) should continuously scale the object up (or enlarge) by 4x at most, whereas moving the joystick down (South) should continuously scale the object down (or shrink) by 0.5x at most.

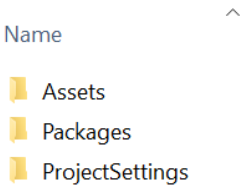
## App Polish (5pts)

Part of your grade will come from how polished your app is. The polish of your app refers to the look and feel of the app and reflects how much effort you've put into making the best app possible given the requirements. For the most part, this is a subjective quality of your app; you'll know it when you see it. Here is the scale that will be used to assess the polish of your app:

> **Excellent:** goes above and beyond the requirements to improve mechanics/interactions; provides aesthetically pleasing look; adds extra interactions/behaviors when applicable to make the app interesting/innovative
>
> **Satisfactory:** meets the requirements; graphics look good but can be more polished; no extra effort to make the most interesting/innovative/different app; no extras; does a good job implementing the minimum
>
> **Half-baked:** one or more requirements is missing; problematic mechanics/weird controls/behaviors; graphics/colors/assets don't look very good; no extras; doesn't convey an effort to produce the best app possible
>
> **Dull:** bare minimums in all aspects; multiple problems with mechanics/graphics/assets. No effort to make the app look and feel good at all.

## Submission Requirements

1. Submit an apk build of your prototype. We should be able to install and run it on our headsets. Your app name (which you can specify in Unity) should use the following naming convention: CS4097_XRA2_LastNameFirstName **or** CS5097_XRA2_LastNameFirstName (based on your section).
2. You should also submit a zipped folder of your Unity Project (please delete redundant assets as they will increase file size). When zipping your Unity Project, include the following folders **only**:




Simply, go to your project folder (you can open it while in Unity). Then select these three folders and zip them. This is different from zipping the entire Unity folder (Unity files aren't included in this method, which will mean smaller files). This folder isn't expected to be very large. Use the same naming convention for the folder.

3. **Also attach all your script files to your submission. The number of script files will depend on how you structure your game. All must be attached! Failing to do this will result in a grade of 0.**

**Failing to meet submission requirements will result in up to 25% penalty above and beyond other point deductions.**

**Missing functionality will lead to additional point deductions than specified for each component.**