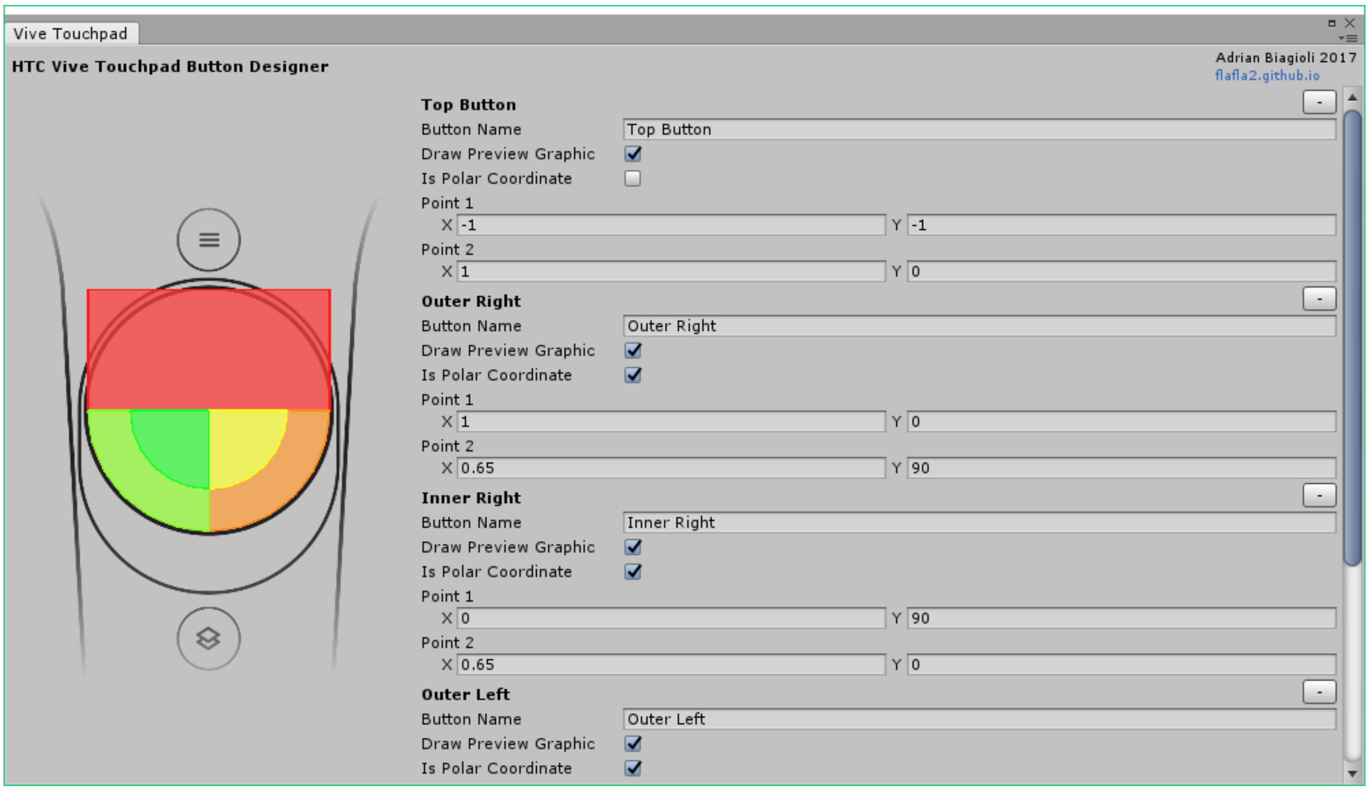
Virtual Button Designer for HTC Vive Controllers

This is a free and open-source tool to create *Virtual Buttons* on the touchpad of HTC Vive Virtual Reality controllers. It is built for the Unity game engine. The buttons themselves are configured via an interactive Editor window GUI, as shown below.


Vive Touchpad

HTC Vive Touchpad Button Designer

Adrian Biagioli 2017
flafla2.github.io

Top Button
Button Name: Top Button
Draw Preview Graphic: ✓
Is Polar Coordinate: ☐
Point 1
X -1 Y -1
Point 2
X 1 Y 0

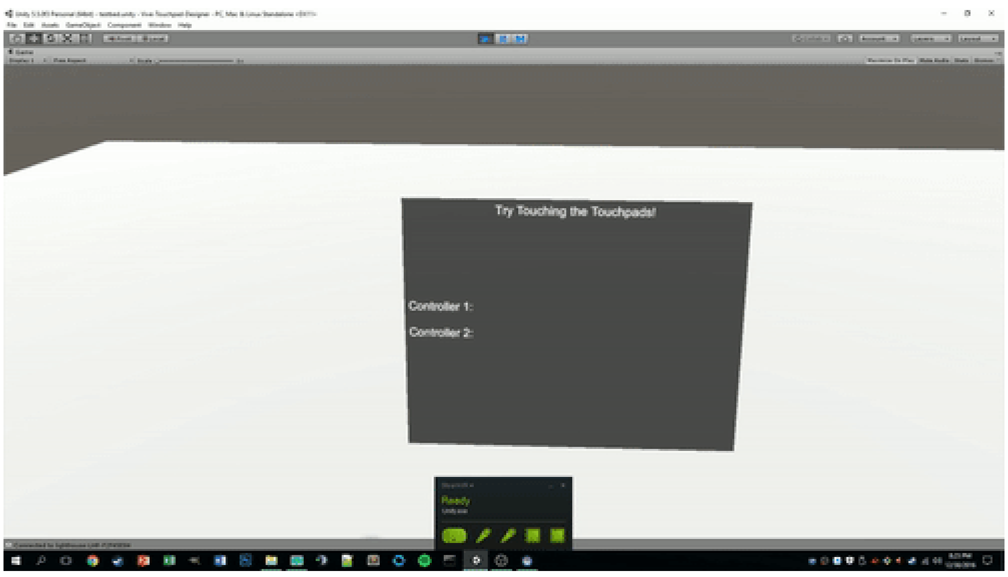
Outer Right
Button Name: Outer Right
Draw Preview Graphic: ✓
Is Polar Coordinate: ✓
Point 1
X 1 Y 0
Point 2
X 0.65 Y 90

Inner Right
Button Name: Inner Right
Draw Preview Graphic: ✓
Is Polar Coordinate: ✓
Point 1
X 0 Y 90
Point 2
X 0.65 Y 0

Outer Left
Button Name: Outer Left
Draw Preview Graphic: ✓
Is Polar Coordinate: ✓

Users can define virtual button regions in standard (Euclidean) coordinates, or in Polar Coordinates. Polar coordinates are useful for creating circular buttons, such as the ones in the above graphic.

I have also included a tool to visualize the virtual buttons themselves in VR. This allows the player to more easily see what they are selecting:

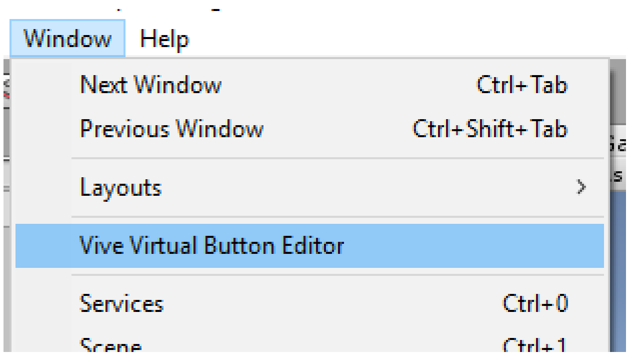

Try Touching the Touchpads!

Controller 1:

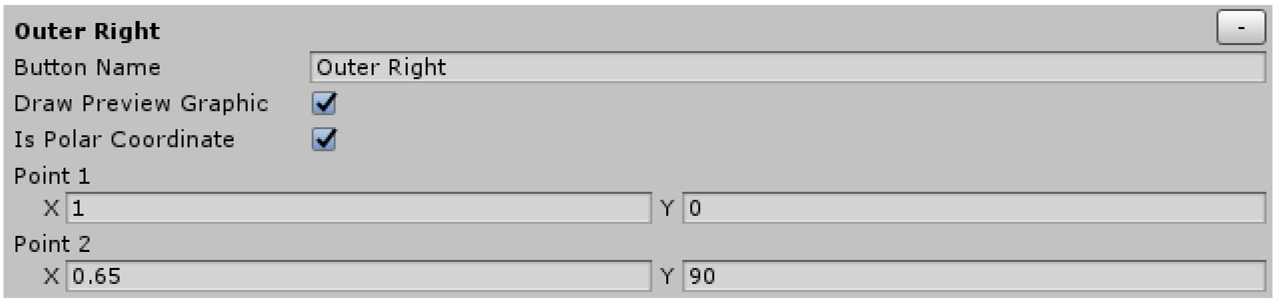
Controller 2:

Getting Started

First, you need to define the locations of all of your virtual buttons. To do this, open the *Virtual Button Editor* via `Window > Vive Virtual Button Editor`:



Changes made in the editor will be saved with your project (via `File > Save`). You can add buttons by clicking the "Add Button" button at the bottom of the editor. You can configure / position each button as desired, as described below:



Outer Right

Button Name: Outer Right
Draw Preview Graphic: ✓
Is Polar Coordinate: ✓
Point 1
X 1 Y 0
Point 2
X 0.65 Y 90

- **Button Name**: Arbitrary name of each button, similar to the "Axis Name" in Unity's input system. You will use this name in code when polling the system for input changes.
- **Draw Preview Graphic**: If true, a preview graphic will be shown on the left side of the Virtual Button Editor.
- **Is Polar Coordinate**: If this is true, the *X* and *Y* coordinates in *Point 1* and *Point 2* (below) correspond to the radius and angle of a polar coordinate (respectively). Otherwise, *Point 1* and *Point 2* will represent normal (Euclidean) coordinates. This is useful if you want your buttons to be circular.
- **Point 1 and Point 2**: These two points define the two corners of the region your button uses. If you are working in Euclidean coordinates, -1 corresponds to the left/top of the touchpad and 1 corresponds to the right/bottom of the pad. In polar coordinates, $r = 0$ corresponds to the center of the pad, and $r = 1$ is at the outer edge. In polar coordinates, the angle is represented in degrees.

Polling Button Presses

If you want to know if the player is touching one of your virtual buttons, simply call `ViveVirtualButtonManager::GetButtonHover` like so:

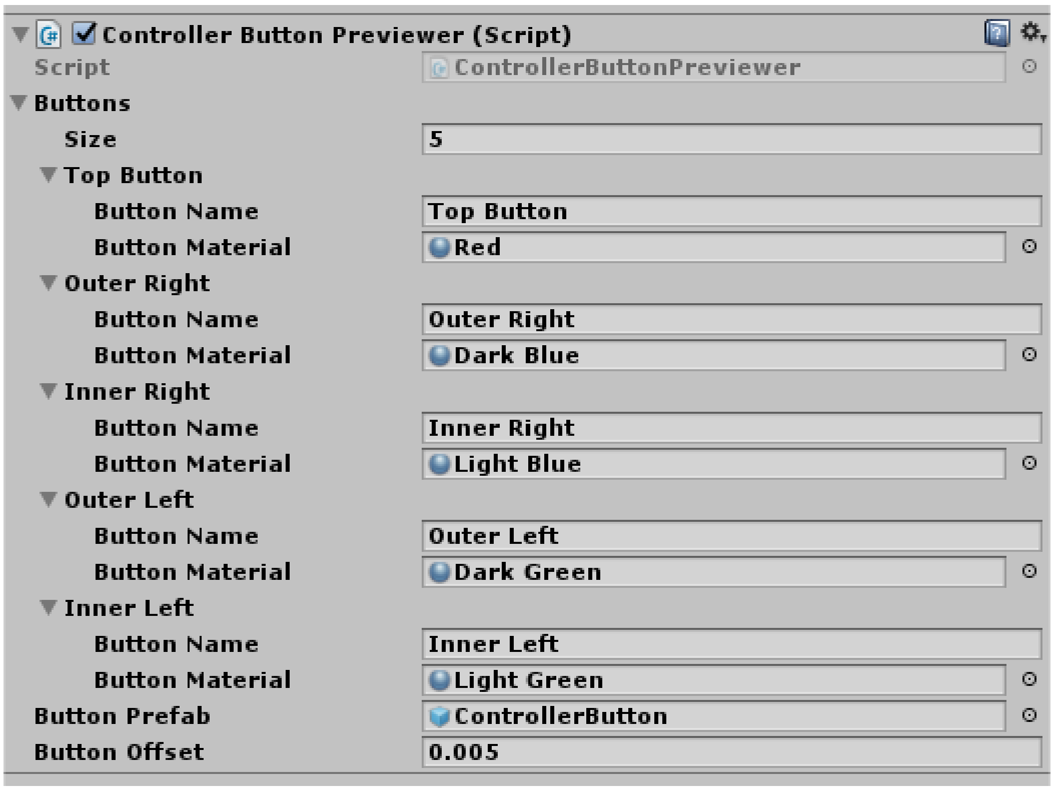
```
// Assume these variables have been populated
string ButtonName;
```

```
SteamVR_TrackedObject Controller;

bool istouching = ViveVirtualButtonManager.Instance.GetButtonHover(ButtonName,
Controller);
```

Visualizing Virtual Buttons

If you want, you can use the `Controller Button Previewer` script to display an interactive overlay on top of the SteamVR tracked controller model. Attach the script to each `SteamVR_TrackedObject` in the scene - if you are using Valve's `[CameraRig]` prefab, then you want to attach this script to the `Controller (left)` and `Controller (right)` objects.



Controller Button Previewer (Script)

Script	ControllerButtonPreviewer
Buttons	
Size	5
Top Button	
Button Name	Top Button
Button Material	Red
Outer Right	
Button Name	Outer Right
Button Material	Dark Blue
Inner Right	
Button Name	Inner Right
Button Material	Light Blue
Outer Left	
Button Name	Outer Left
Button Material	Dark Green
Inner Left	
Button Name	Inner Left
Button Material	Light Green
Button Prefab	ControllerButton
Button Offset	0.005

- **Buttons**: An array of Buttons that you want to display. For each entry you must supply the Button Name (created in the editor as described above) and a material to render the preview mesh with. UVs in the generated mesh correspond to (0, 0) on the top left of the touchpad area and (1, 1) on the bottom right of the touchpad area. Note that you do not need to provide *all* of the buttons that you create in the editor, so you can use different combinations of buttons for different gameplay scenarios.
- **Button Prefab**: This is the prefab that will be instantiated for each virtual button. It *must* have an `Animator` component attached. The script will send `Hover` and `Press` boolean parameters to the Animator for button Hover and button Press events.
- **Button Offest**: Describes the distance that the button meshes should protrude out of the touchpad origin point.