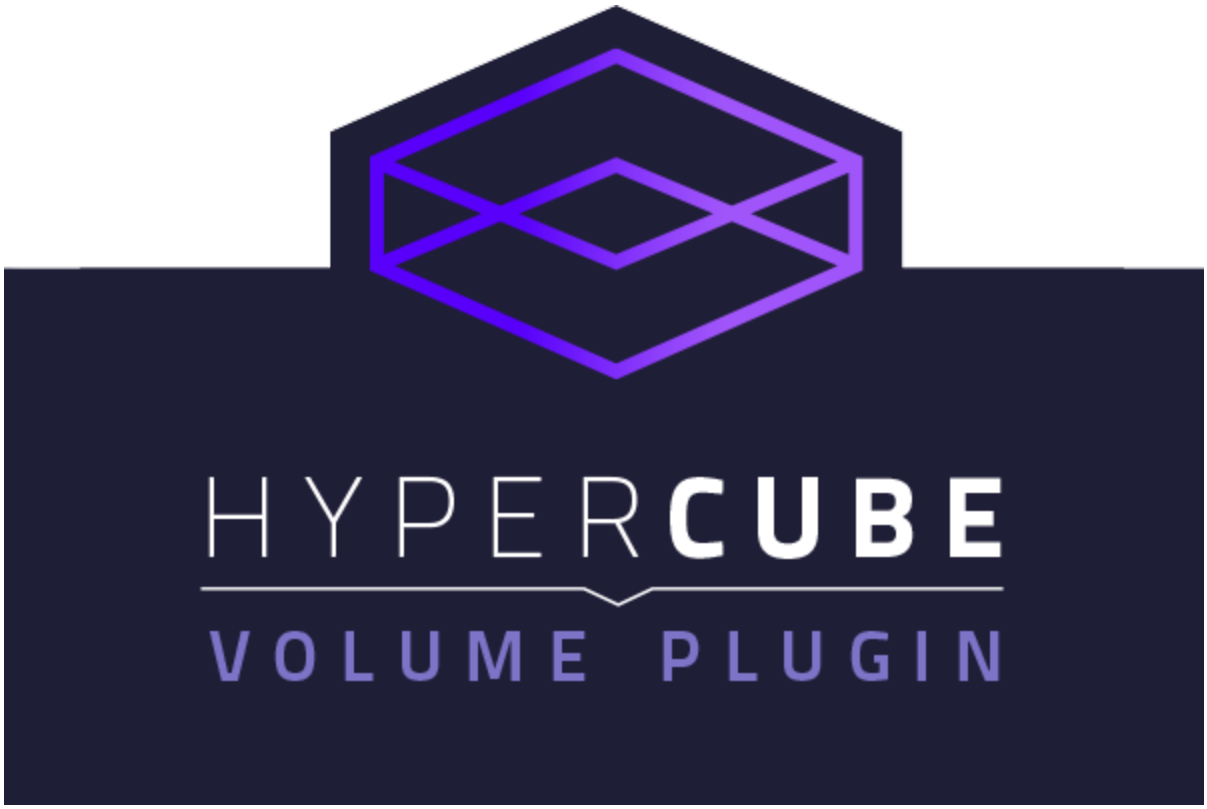
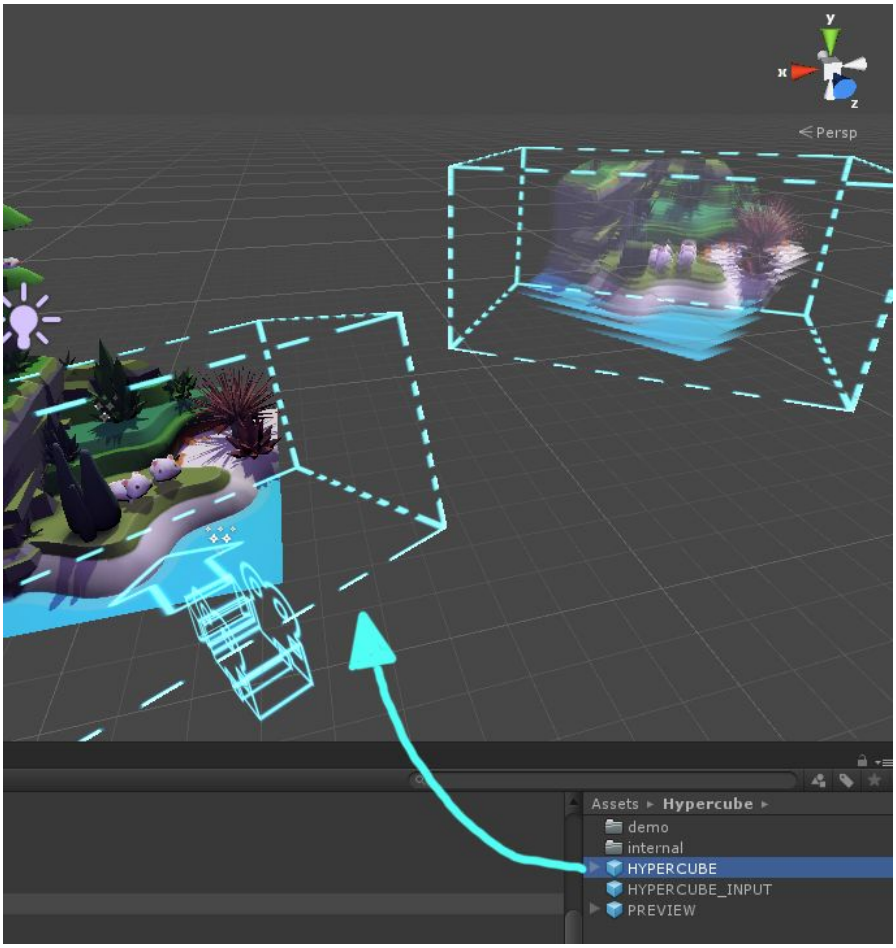

HYPERCUBE
VOLUME PLUGIN

How to use Hypercube:



Drag the Hypercube prefab into your scene.
THATS it!

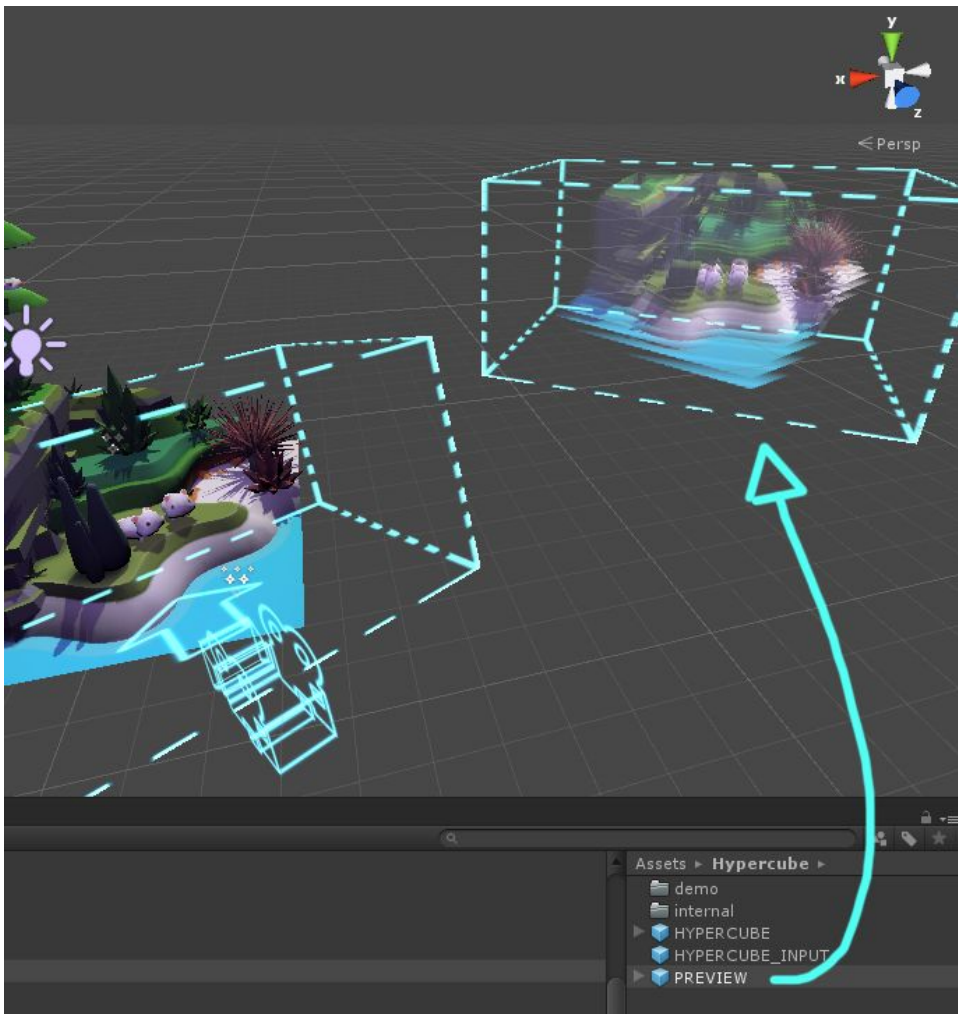
Any geometry inside the Hypercube will be rendered accordingly in Volume!
Is it really that simple? Yes. Just BUILD! :)

For best results, however, you will probably want to use a few more tools to make your life easier yet, so read on!

Previewing content for Volume in the Unity Editor:

Use the PREVIEW prefab.

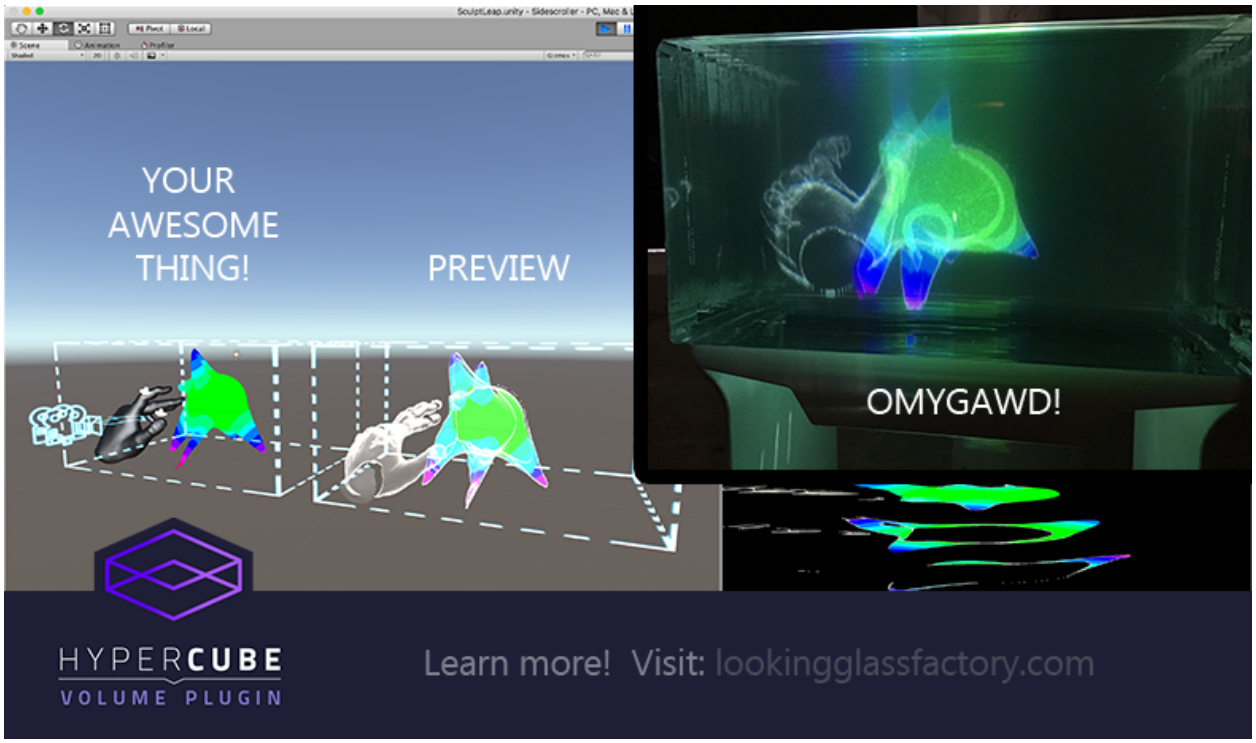
This is not a way to view your content in the Volume itself, but you can preview an idealized visualization of how it will look inside Volume by dragging PREVIEW.prefab into your scene. It will connect automatically to the Hypercube Camera.



The PREVIEW itself can not be rendered by the Hypercube (it is purposefully set on a different layer).

Previewing Unity content inside Volume:

Use the Hypercube Caster Window.



Editor content being shown directly in volume, in real-time, using the Caster Window.

NOTE: For best results, set "Edit > preferences > Colors > playmode tint" to white so that the window will not be greyed during play (which occurs only on some versions of Unity)

OSX:

1. Go to *System Preferences > Mission Control > Displays have separate spaces* set to **OFF** (this will remove the OSX menu bar from the Volume's display). This will require a logout if it was previously ON.
2. Use *Volume > Caster Window Prefs* to set the **resolution to: 1920x1080, other values should be 0**.
3. **Move your mouse over to Volume's screen** and press ⌘E to toggle the window.

4. If you want to close the caster window, also press ⌘E

Windows:

1. If open, close Unity.
2. Connect Volume's HDMI to your CPU.
3. *RMB on the Desktop > Display Settings*
4. Configure Volume so that it's display is to the **left of your main display**. Align the top of it to the top of your main monitor. If that is not possible, align the bottoms.
5. Start Unity.
6. Open *Hypercube > Caster Window Prefs*
7. Set the resolution of your Volume to **1920 x 1080**, and the "**X Position**": to **-1920**.
8. If the Volume display is aligned with the top of your main display set **"Y Position" to 0**, otherwise if the bottoms are aligned set it to (**Main Display Height - 1080**).
9. Press Ctrl + E to toggle the window.

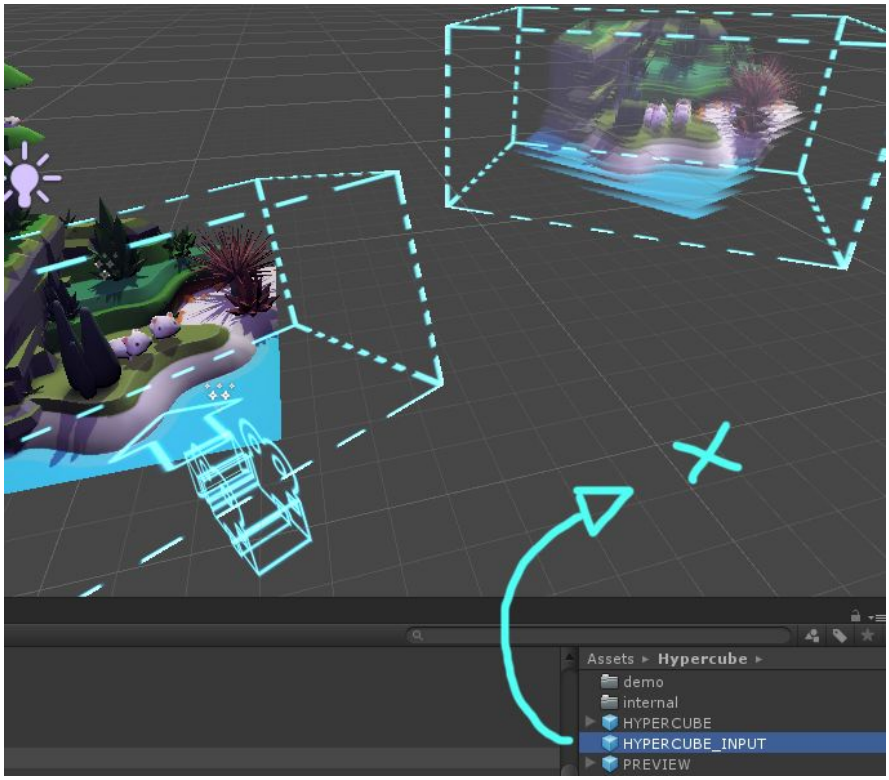
Note: *If the caster window is misconfigured, it is possible for it to block important Unity GUI elements. Close it with Ctrl+E.*

Note: These settings will be saved, and from then on be able to open/close the window with the Toggle menu or hotkey.

Getting input from Volume:

Volume is equipped with touchscreen input. How can we read these events? Super easy:

1. **Hypercube > Load Volume friendly Unity prefs.**
   Among other things, what this does is add a preprocessor define needed for Volume input called HYPERCUBE_INPUT, and also sets our build to use .Net 2.0, as opposed to .Net 2.0 subset. This is needed because .Net 2.0 subset doesn't include the IO.Ports library which we need to talk to the touch screen via serial port.
2. **Drag in the HYPERCUBE_INPUT.prefab into your scene**



We're ready to grab input now :D

VOLUME INPUT CODE:

Once the above are done, there are 2 ways to get touch events from Volume:

1) **hypercube.input.frontScreen.touches**
   Is an array that will contain any touches during that frame. From there you can get positions, movement difference from the last frame, etc.

2) Inherit from **hypercube.touchScreenTarget**, and then override the methods
   public override void onTouchDown(hypercube.touch touch)
   public override void onTouchMoved(hypercube.touch touch)
   public override void onTouchUp(hypercube.touch touch)
   To get events each frame.

Here are some more things that you can do with hypercube.input:
If placed inside Update(), these lines of code will cause **Transform t** to be manipulated by touch input:

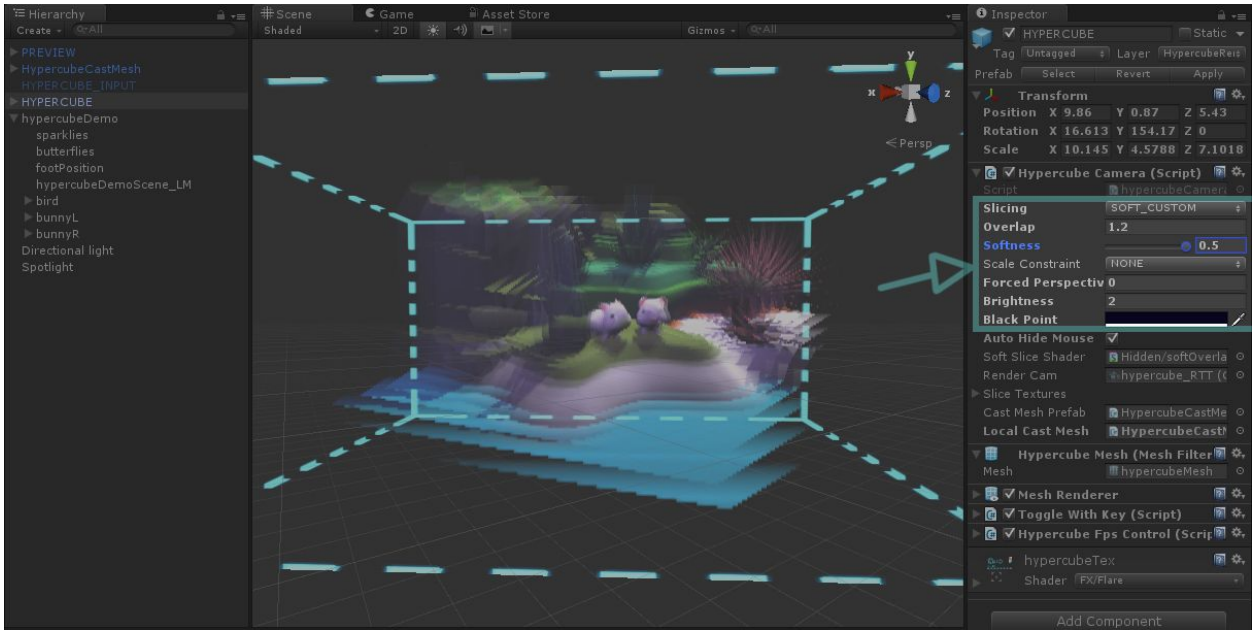
```
t.Translate(hypercube.input.frontScreen.averageDiff.x, hypercube.input.frontScreen.averageDiff.y, 0f, Space.World);

t.Rotate(0f, hypercube.input.frontScreen.twist, 0f);  //ROTATE t

t.localScale *= hypercube.input.frontScreen.pinch;  //SCALE t
```

Getting best results:

To get the best output you can into Volume, you may find that you have to rethink some common assumptions about 3D art. For example, geometries can't simply be clipped into each other because those faces would now still be clearly visible in Volume. Another example is eyeballs, which would be visible inside of a character's head if they are modelled as spheres. Further, because Volume uses separate slices to generate the 3D effect, how the slices blend is an important part of generating convincing art in Volume. The slice blending should be tweaked to best meet your project's needs. Full control of this is available on the HYPERCUBE asset.



Play with the Hypercube visual settings to optimize slice blending for your project.

Another thing to note about rendering via HYPERCUBE is that **many materials will not draw properly with soft slicing** enabled. Hypercube brings its own set of shaders designed to be used with it. These have all been tweaked to work well with soft slicing, and for drawing back faces, if desired, and should meet 99% of your shader needs in Volume. Therefore it is recommended to use these shaders for all geometry rendered by the HYPERCUBE.

Finally, it should be noted that if soft slicing is used, it is not possible for any blended effect shaders to work properly in HYPERCUBE (that is: non-opaque shaders like alpha-blend, add, multiply) . This is because these effects require them to NOT write to the z-buffer (zwrite off). The problem with this is that the z-buffer is how soft slicing determines how to blend between the slices, and therefore anything with 'zwrite off' will be treated as empty space and not drawn at all by the soft-slicing postprocess. Hence, for any alpha transparency needs, only alpha-test shaders will work well along with soft slicing. We provide the *Hypercube > Unlit Cutout* shader for this purpose. Alternatively, HARD can be chosen in the 'Slicing' option on HYPERCUBE, but this *generally* doesn't produce as pleasing a result in Volume.

*__Hypercube: Volume Plugin__ includes code by Daniel Wilches: http://www.cs.uwyo.edu/~dwilches/