HYPER**CUBE**
VOLUME PLUGIN

Unity Reference Manual

TABLE OF CONTENTS

How to use
HYPERCUBE

Drag the HYPERCUBE prefab into your scene.
THAT'S 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, so read on!
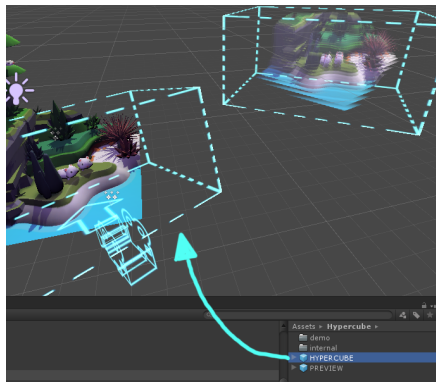
Previewing Content for Volume in the Unity Editor

Use the PREVIEW prefab.

Preview how your visualization will look inside Volume by dragging the PREVIEW.prefab into your scene. It will connect automatically to the HYPERCUBE Camera.

● The PREVIEW also adds a feature to your player: if a Volume is not detected at runtime, the preview will display itself instead of slices, hence your application will display a coherent image on a monitor when a Volume isn't accessible.

● This is only a preview. To skip to viewing your content in Volume itself,
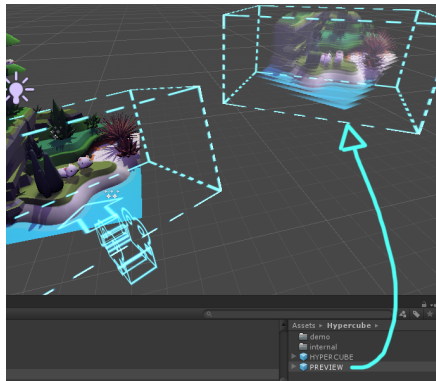
Previewing Unity Content Inside Volume

Use the HYPERCUBE Caster Window.

● For best results, set **Edit > Preferences > Colors > Playmode Tint** to **White** so that the window will not be greyed during play (occurs only on some versions of Unity)
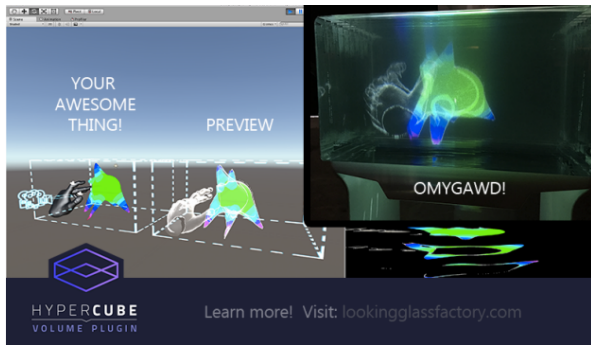
Editor content being shown directly in Volume in real-time using the Caster Window.

How to use the HYPERCUBE Caster Window

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: **1920 x 1080**, 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 computer

3. Right click on the **Desktop > Display Settings**

4. Configure Volume so that its display (2) is to the **left of your main display (1)**. Align the tops of both displays. 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: -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

Getting Touch Input

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. We're ready to grab input now :D
  Open: Assets/Hypercube/demo/DEMO_touchInput.scene to see
  working example code using the hypercube.input class.
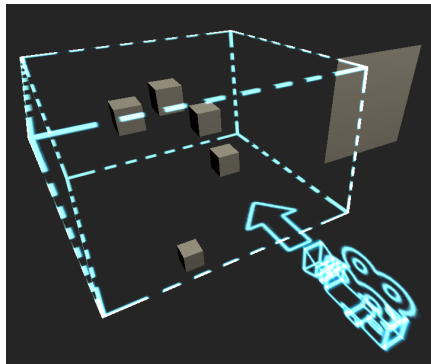
FEEEEL the power !

USING VOLUME INPUT:

There are two ways to get touch events from Volume:

1. hypercube.input.touchPanel.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.
NOTE: touchPanel may be NULL until Hypercube can connect to the hardware.

2. Inherit from hypercube.touchScreenTarget
and then, override it's methods to get events as they occur:
- public override void onTouchDown(hypercube.touch touch)
- public override void onTouchMoved(hypercube.touch touch)
- public override void onTouchUp(hypercube.touch touch)

Below are some snippets that use hypercube.input
If placed inside Update(), these lines of code will cause **Transform t** to be manipulated by touch input:

Translates the transform based on touch input
```
t.Translate(
    hypercube.input.touchPanel.averageDiff.x,
    hypercube.input.touchPanel.averageDiff.y,
    0f,
    Space.World);
```

Rotates the transform
```
t.Rotate(0f, hypercube.input.touchPanel.twist, 0f);
```

Scales the transform
```
t.localScale *= hypercube.input.touchPanel.pinch;
```

Getting the Best Results

CREATING 3D ART FOR VOLUME

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 be clearly visible in Volume

- Eyeballs, if they are modeled as spheres, would be visible inside of a character's head

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.

*Hypercube: Volume Plugin includes code by Daniel Wilches: http://www.cs.uwyo.edu/~dwilches/
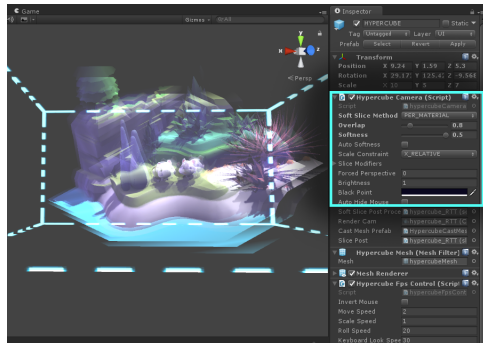
Play with the Hypercube visual settings to optimize slice blending for your project.

Learning to Kick Ass in Volume

SOFT SLICE METHOD

Hypercube comes with different rendering methods that you can use to draw a scene inside Hypercube.

HARD	PER_MATERIAL	POST_PROCESS	OCCLUDING
If you want it to look old-school	**If you want it to show fancy effects**	**If you can't use Hypercube shaders**	**If you mostly have 1 only large object to display**
No slice blending. Blending will be OFF.	All objects will draw, but only meshes using Hypercube/ shaders will be soft sliced. Use this method and use the provided Hypercube shaders if you want to have effects show well in Volume.	Uses the depth buffer in a post process to calculate soft slicing. This means shaders that do not ZWrite will be treated as empty space and draw black (effects, or transparent things). However, ANY opaque shader will be soft sliced. Use this if you want soft slicing, but don't want to use Hypercube shaders.	Draws the scene one time, and then uses a post process to determine what slices a pixel draws to. The result is that pixels drawn to 'front' slices occlude pixels drawn behind them. Effects and transparent shaders will most likely draw to the wrong slices with this method (because they typically use ZWrite OFF). Framing whole models (like a human head or whole opaque object) tend to show well with this method. NOTE: OCCLUDING mode does not allow for the use of slice modifiers (covered below).

SCALE CONSTRAINT

Which should you use? Choose the axis that would have the most effect on your application.

EXAMPLE 1
I am making a game that shows a character in the screen, it may not matter if I can see more behind him, or to his sides, hence I should choose **Y_RELATIVE** to make sure his head and feed don't get cut off. With **Y_RELATIVE**, my Y will be deemed holy and untouched by Hypercube. X and Z scales will be adjusted to suit the Y I set.

EXAMPLE 2
I am making a game where I am flying into a space. It is very dependent on being able to see into the space and judge distance. I should choose **Z_RELATIVE** because the depth in this game is critical to its play. Hypercube will then adjust X and Y scales to match the display.

- Just like monitors, some Volume models are wider, deeper, or taller than others. To make things in Volume display at 1:1 to match your 3D scene, Hypercube changes its aspect ratio to match the physical projection of the connected Volume (not doing so would cause things to appear squashed or stretched in the display).

- To still allow you to control its size, Hypercube allows you to choose a scale axis to leave free, and it will constrain the other 2 to keep the physical aspect ratio. EXAMPLE: If I have Hypercube Scale Constraint set to **X_RELATIVE**, I can set the X scale of my Hypercube to whatever I want, but the Z and Y scale will be controlled by Hypercube to keep a consistent aspect ratio based on the X I set.

- If I set my Scale Constraint to NONE, Hypercube will not try to control its aspect ratio and will simply allow things to appear squashed or stretched in Volume.

Cheat Sheet

AUTO HIDE MOUSE
Hides the mouse for you in the player.

AUTO SOFTNESS
Mathematically calculates the softness based on the overlap. This gives a correct value to have a true total opacity of 1 across slices, preserving the integrity of the geometry. In practice, however, other values often give better visual results.

BRIGHTNESS
This is a brightness modifier on the final output of the castMesh.

BLACK POINT
This color is added to the output color of all geometry in Hypercube. Because Volume is an additive display, there is no difference between 'void" in Volume, and 'black". Hence, this feature can be used to differentiate between a black surface and a space with nothing in it.

FORCED PERSPECTIVE
If the Hypercube's camera (hypercube_rtt) is set to perspective mode instead of orthographic, this modifies each successive slice's field of vision. Use it to make forced perspective effects in Hypercube.

OVERLAP
Controls how far into sister slices each slice should draw. This helps reduce the 'banding' between slices.

SLICE MODIFIERS
Slice modifiers are a way to add things like GUIs or backgrounds to slices. They work with a 'depth slider' to choose the slice. Because different Volume types have different slice counts, the 'depth' value you set on a particular slice mod is how the tools will decide which slice to best apply it to.

A slice modifier consists of:
- A depth setting (used to calculate which slice it should be applied to)
- A blend method (OVER, UNDER, ADD, MULTIPLY)
- A texture (it can be a render texture or other)

NOTE: If the depth is required to change in runtime, use the setDepth(d) method on the slice modifier you want to change in order for it to update correctly. Slice modifiers do not interpolate between slices. They round to the most appropriate slice based on the depth value you set. Hence, if you want interpolation, place your GUI in the scene itself and don't use the slice modifier feature.

NOTE: If you want to match the slice modifier texture resolution to the slice's output resolution, use hypercube.castMesh.rttResX/Y to get it, note that it may change up to several seconds after initial startup.

SOFTNESS
Controls the blending amount of the content inside slices. Valid values are 0 to .5. A 0 value means no blending (hard slicing), and a .5 value means that geometry will be completely blended from totally transparent when it is near the clipping planes of the slice, to completely opaque only when the geometry is perfectly centered in the slice. EXAMPLE: A value of .2 will make any geometry placed between .2-.8 of the slice be fully opaque while geometry in 0-.2 would be blended and .8 to 1 would also be blended.