Unity:	5.2.2f1
Date:	10.03.17
Author:	NoiseCrime

Portal Room Cube via Stencils Guide

Unity 4.2 introduced access to the stencil buffer, enabling various effects to be accomplished such as portals through masking areas of the screen.

This project is an exploration using stencils to provide a portal system (that's the technique, nothing to do with the game of the same name), allowing for multiple views into the same space but showing different results depending upon which portal is in view.

Previously the only way to accomplish this would be through render to texture, though that has some drawbacks such as high texture memory usage, dependant upon screen size and number of portals. Though stencil portals don't suffer from increased texture memory they have their own set of problems such as potential inefficient rendering as polygons will still go through the pipeline until culled by the stencil mask. So ultimately the best method to use is very dependant upon your requirements.

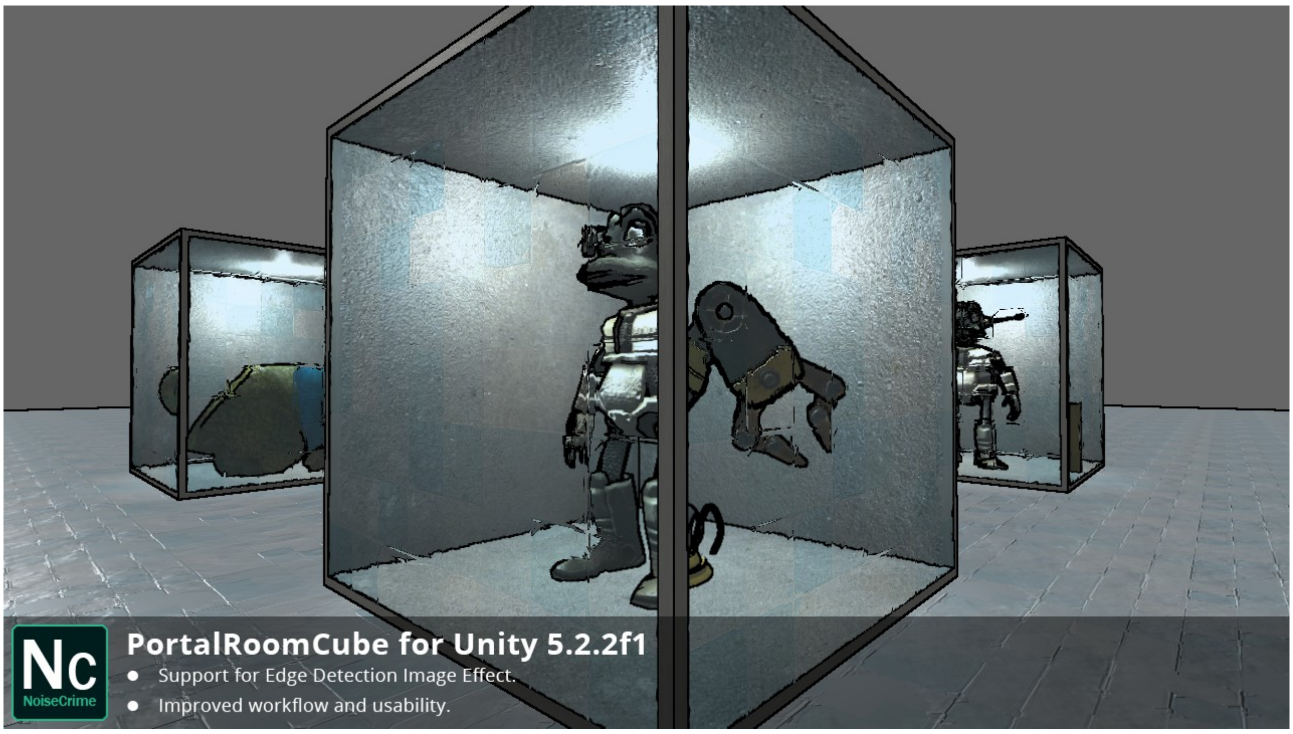

Nc
NoiseCrime
PortalRoomCube for Unity 5.2.2f1
Support for Edge Detection Image Effect.
Improved workflow and usability.

Table of Contents

1.0 Requirements & Known Issues

- Unity 5.2.2f1
- Only works with Forward Rendering
- Edge Detection only works in dx9 mode.
- MaterialPropertyBlock demo broken in 5.2.2f1 and upwards.

2.0 History

10.03.17 - Unity 5.2.2f1

This version has been updated to Unity 5.2.2f1, introducing several new features, streamlined design, simplified shader code, improved shader interface, support for use of DepthNormalsTexture in Image Effects and more.

Features

- Stencil shader simplification, no longer using hard-coded stencil properties.
- MaterialPropertyDrawers for stencil shader property assignment in inspector.
- Support for DepthNormalsTexture generation – DX9 only (e.g. edge detection).
-
- Updated to relevant packages – e.g. 5.2.2f1 Image Effects for Glass Reflection

Demo Features

- UI for enabling/disabling features and changing settings.
- Enable/Disable render scene with Edge Detection image effect.
- Enable/Disable stencil debug viewer.
- Enable/Disable Depth Mask Cubes – illustrate bugs.
- Enable/Disable dynamic portal content culling.
- Enable/Disable rendering of Glass walls and control distortion strength.
- Enable/Disable Help and frame rate.

22.07.13 - Unity 4.2 Pro (Original version)

Stencil shaders were not very streamlined as values had to be hard-coded in the shader, often requiring a shader per stencil ID.

3.0 The Stencil Buffer

Please refer to the Unity documentation in the reference section for ShaderLab to learn more about stencils, and how to use them in shaders.

See https://docs.unity3d.com/Manual/SL-Stencil.html

3.1 Stencil Tests

The Stencil Test is performed after the fragment shader, where the fragments stencil value is tested against the current value in the stencil buffer. If the stencil test fails the fragment is discarded, while if it passes other tests such as Depth Test may still discard the fragment. Even with early fragment tests (e.g. early Depth test) if the fragment is discarded the stencil buffer can still be modified.

Sadly as the general case for the stencil test happens after the fragment shader it suggests that in many cases the test will not necessarily improve performance.

However there is apparently a shader layout qualifier/attribute that can force the execution of early stencil tests, though there are restrictions and consequences to be aware of. See the following links for more information.

https://www.khronos.org/opengl/wiki/Early_Fragment_Test
https://msdn.microsoft.com/en-us/library/windows/desktop/ff471439(v=vs.85).aspx
https://stackoverflow.com/questions/17898738/early-z-test-depth-test-in-directx-11

3.2 Clearing Stencil Buffer

It should be noted that currently it is not possible to expressly clear the stencil buffer and that clearing the depth buffer will clear the stencil buffer too. However it should be possible to effectively reset/clear these independently via writing custom graphics.Blit code and shaders, though this is currently outside the scope of this document.

3.3 Stencils & Deferred Rendering

If using the deferred renderer it is important to understand that Unity makes extensive use of the stencil buffer and as such it is probably impractical to use it in many cases.

4.0 The Basic Stencil Portal Concept

The basic concept of using stencils for creating portals is simple and straightforward. You first render a polygon or mesh that sets a specific value in the stencil buffer, effectively creating a mask. Subsequent meshes are then checked against this stencil mask at the pixel/fragment level, if the pixel to be drawn has the same stencil reference value it continues down the pipeline to possibly being rendered, if not it is culled.

4.1 A Single 'Portal Room Cube'

The single portal cube room requires four stencil masks, one for each side of the cube (ignoring top and bottom) using a single shader, that renders a unique reference value (stencil ID) into the stencil buffer. The stencil mask shader must be rendered prior to any other meshes and therefore must have depth buffer writes disabled, so that subsequent content can be rendered inside the cube and not culled by the depth of the stencil mask itself.

Content for each portal room within the cube requires custom shaders with stencil support, and correct assignment of the stencil ID that matches one of the four stencil masks.

Complications arise in that the contents of each Portal room must have its own set of unique materials, as each must assign the relevant stencil ID for the portal being viewed.

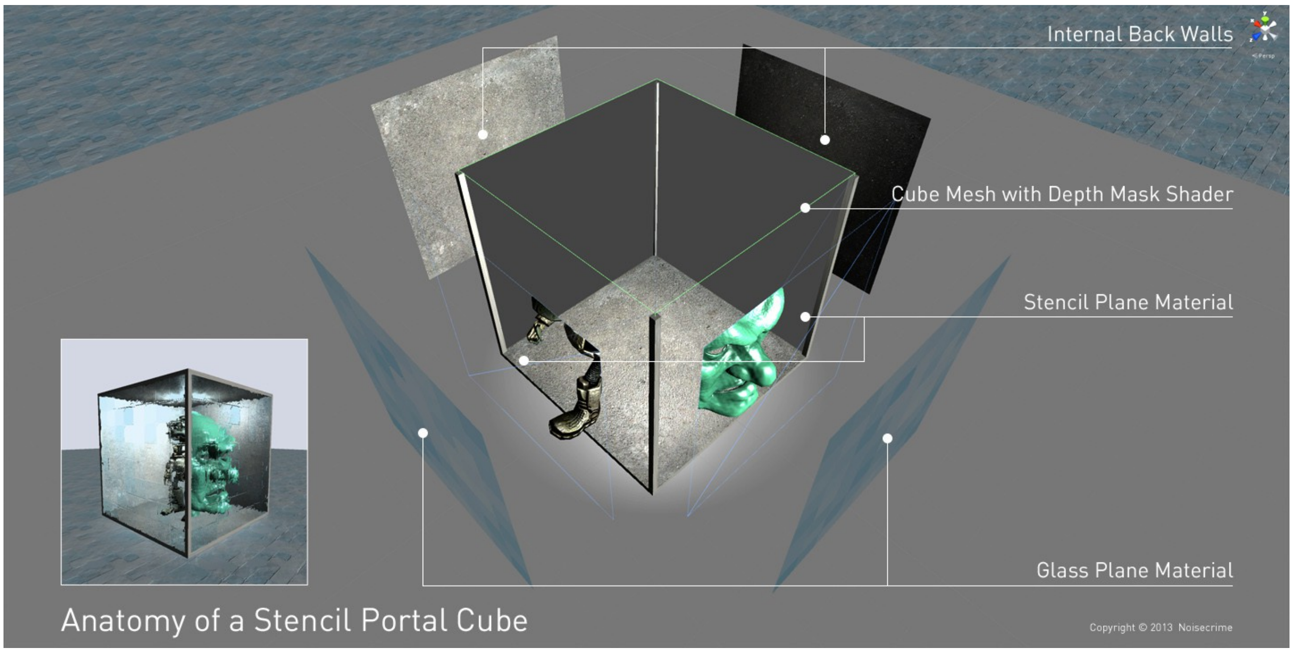
This could be greatly simplified and materials shared between portals if stencil state values were supported by MaterialPropertyBlocks, but sadly they are not (as of Unity 5.6.0b10) or due to a bug (bug report 889652).

Other aspects that make this more complex are the contents within a portal, for example lighting or physics colliders as these may need to be defined on unique layers to avoid affecting other portal instances.

4.2 A Room of Portal Room Cubes

Again the current limitations of stencils in Unity make this increasingly more complex than it needs to be. The biggest issue here is that we can't clear the depth buffer after rendering of the stencil mask portal planes. This forces us to resort to using stencil Mask shaders that do not write to depth buffer and creative use of a secondary cube mesh .

4.3 Anatomy of a Portal Room Cube


Internal Back Walls
Cube Mesh with Depth Mask Shader
Stencil Plane Material
Glass Plane Material
Anatomy of a Stencil Portal Cube
Copyright © 2013 Noisecrime

Each 'Room Cube' is comprised of the same elements, as shown in the exploded view above.

The Room its floor, ceiling, struts, internal walls and glass plane façades exist in the main world layer and are always rendered. Elements such as the internal walls and glass planes are single sided planes, so they are not rendered when facing away from the camera.

Just behind each glass plane façade is a plane with a unique stencil mask shader applied to it, see the StencilMask_X shaders in the project. There are four of these stencil mask shaders, one for each side of the cube and each having a unique stencil reference value (1 to 4 for simplicity).

These are drawn before the contents of the Room cube so as to populate the stencil buffer with their respective reference values, though use of the 'Queue' Tag in the shader and setting the value to Geometry-100.

Between the glass façade and the stencil plane is a cube mesh with a 'depth mask' shader applied and which only renders back facing polygons. This is used to fill the depth buffer to prevent incorrect drawing order of the stencil planes due to the fact that they do not write to the depth buffer. It also uses a modified 'queue' value (Geometry-120) to ensure that it is the first thing rendered every frame.

Each frame the gameObjects are rendered in a specific order, determined via the queue tag in the custom shaders.

• DepthMaskCube	- Queue Tag 'Geometry-120'
• StencilQuads	- Queue Tag 'Geometry-100'
• All other Content	
• Glass Façades	- Queue Tag 'Transparent+100'

4.4 Controlling the Render Order of the Scene

The first objects rendered to the screen each frame are the '_DepthMaskCube' gameObjects for each 'Room', accomplished through the use of the 'Queue' tag (set to 'Geometry-120') defined in the 'DepthMask' shader itself. These cubes are slightly smaller than the actual room, its faces are therefore behind the glass planes, but in front of the Stencil Mask planes.

As the DepthMask shader uses 'ColorMask 0' nothing will actually be rendered to the screen, but the depth of the pixels will be rendered into the depth buffer. Most importantly though we are not rendering the front faces of the cubes, but the back faces (via 'Cull Front' in the shader) for reasons which will become clear.

The problem with stencil masks in this project is that we have to render them with depth writes disabled (I.e. disable writing the depth of each pixel into the depth buffer). The reason for this is because rendering the contents of each room cube needs to happen after the stencil mask planes and as the content is behind the stencil mask plane if we write the planes depth to the buffer the content would be culled (as its depth is further away) and never make it to the screen.

With the Stencil Mask planes not writing to the depth buffer there is no way to control their draw order with respect to distance from the camera (without resorting to some scripting solutions to sort them) and as such the Stencil Mask planes from other cubes can randomly overlap those of the cubes in front.

You can observe this issue if you disable the 'DepthMaskCubes' option in the scene and it will be more evident if you enable the 'Stencil Viewer' too, then move the camera around.

The 'DepthMaskCube' is the simple solution to work around this problem (see following images), though it is very much dependant on the fact that this is a special case. By which I mean it takes advantage that the portals themselves are constructed around cubes which are convex and that the portal rooms inside each cube never extend beyond the cube. Without those special cases a more generalised solution would be needed, unless it becomes possible to clear the depth buffer part way through a render or between cameras.

The reason for the DepthMaskCube rendering only the back faces is that it fulfils all the requirements. It prevents portals from other cubes showing through, but importantly it wont prevent the content from within a cube being rendered as the depth buffer is being filed with the depths of the back walls of the cube, not the front walls (with respect to the camera viewpoint).

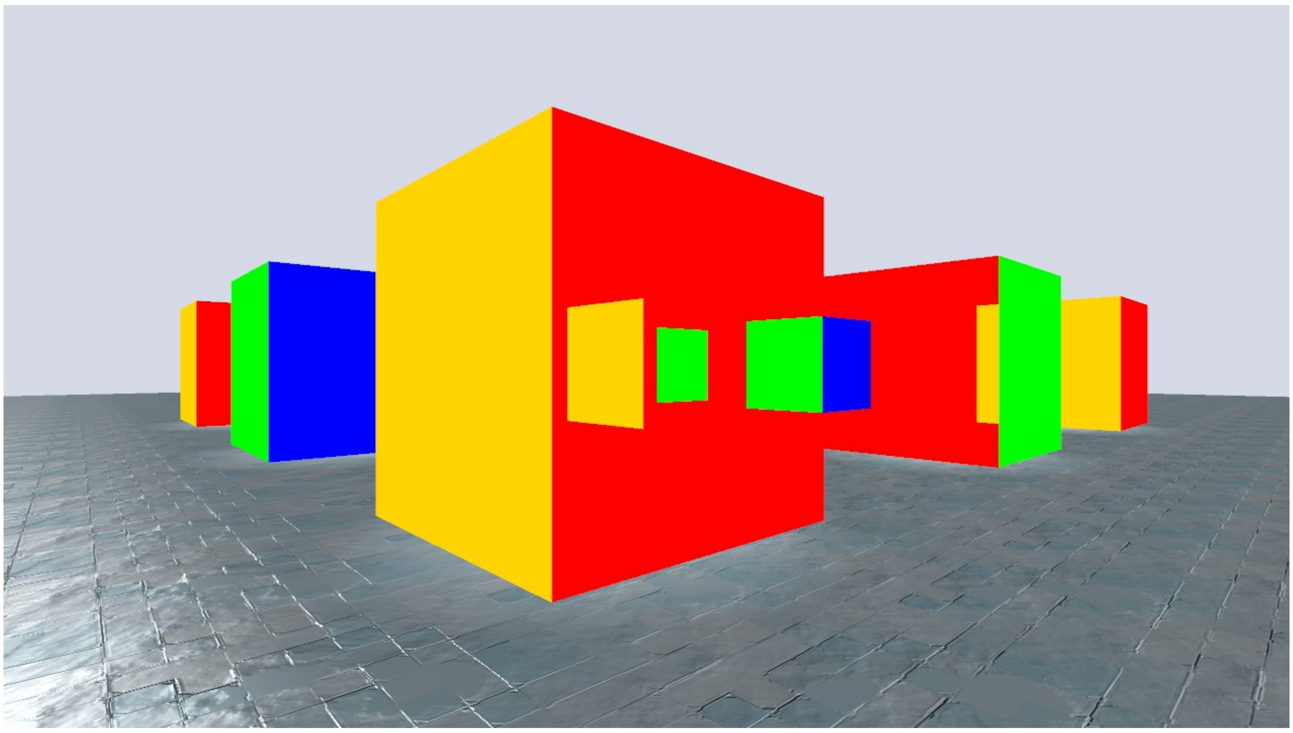


Illustration 1: Without the DepthmaskCube along with the stencil mask planes not writing the to depth buffer results in incorrect draw order, and random overlapping of stencil masks.

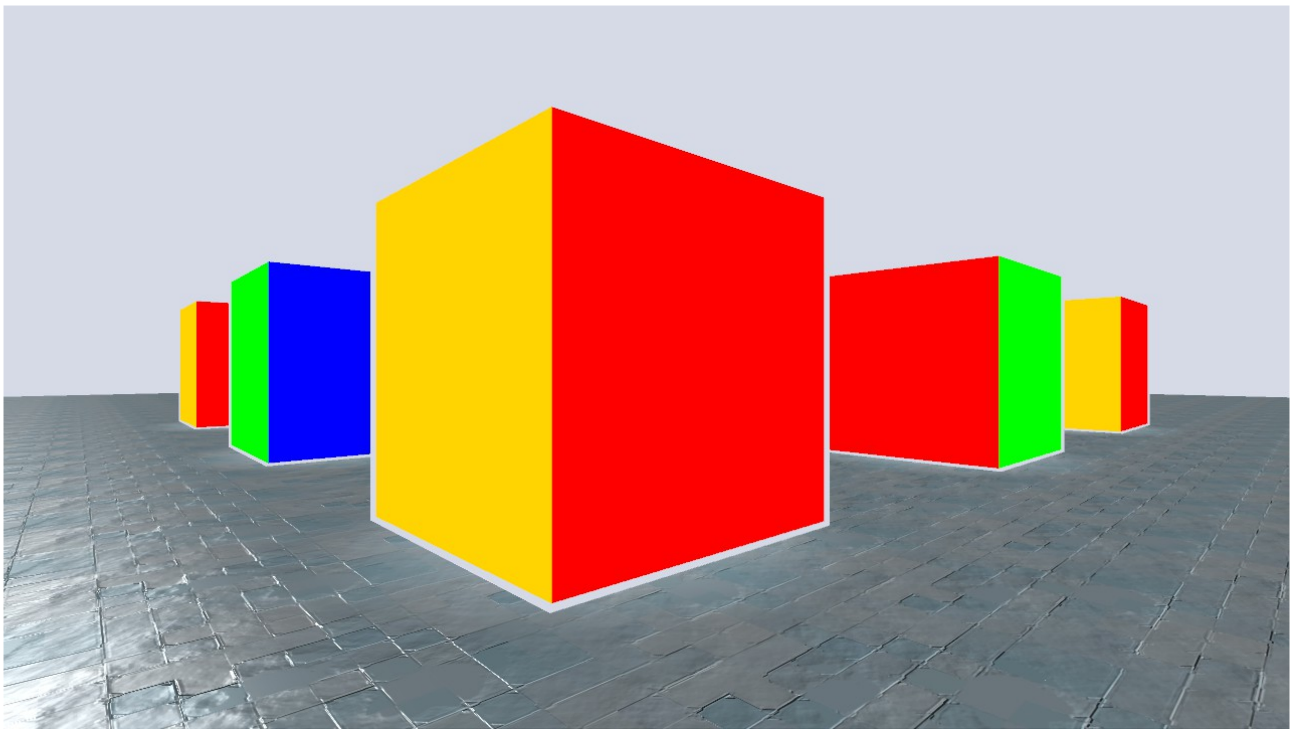


Illustration 2: With the DepthmaskCube and due to the fact that cubes are convex we get correct rendering of the stencil masks.

5.0 The Stencil Shaders

5.1 Stencil Masks

While each of the 4 sides of a portal room require a unique material, they all share the same shader through use of shader parameters that allow per material settings to be applied. The per material settings can be applied via code or through the inspector.

```
Stencil
{
        Ref[_StencilReferenceID]
        Comp[_StencilComp]   // always
        Pass[_StencilOp]     // replace
        ReadMask[_StencilReadMask]
        WriteMask[_StencilWriteMask]
}
```

So if _StencilReferenceID is set to a value of 1, then for every pixel that is rendered (and passes the depth test if active) it will put a value of 1 into the stencil buffer. This beahviour is controlled by the stencil compare function and stencil pass function. For the Stencil Mask shader these are 'always' and 'replace' respectively, meaning that the compare function 'always' passes and so it will replace whatever the value is in the stencil buffer with our stencilReferenceID.

More information about the specific keywords used and their meaning can be found in the Unity ShaderLab documentation.

5.2 Legacy Shaders

In the project all these shaders are simply taken directly from the Unity Built-in shader source which can be found at http://unity3d.com/unity/download/archive. The only difference is the addition of the stencil block which is the same as the above stencil mask shader.

However in these cases StencilReferenceID will be set to a value between 1 and 4 (representing any of the four portals), the compare function is set to 'Equal' and the pass function set to keep.

This means if the pixel being rendered to has the same value (equal) as the StencilReferenceID then the pixel is drawn to the screen. The pass function 'keep' indicates that we will not modify the stencil buffer.

6.0 The Examples

The project provides a number of different scenes that illustrate the process of building the PortalRoomCube example, from the basics to the most advanced.

- **PortalRoomBasic**
  This is a simple scene that illustrates the use of the stencils to create two rooms In the same volume that can be viewed from two sides.

- **PortalRoomComplex**
  Building on the previous scene, this one adds additional assets such as glass facades, four rooms sharing the same volume space and use of the depthMaskCube to avoid depth buffer issues.

- **PortalRoomComplexGrid**
  This scene expands the portal room concept further, duplicating the room 8 times to create a 3x3 grid of them.

- **PortalRoomComplexGridMPB_Broken**
  This scene was created to make use of MaterialPropertyBlocks (MPB) to make material usage more efficient (I.e less materials). Alas as of Unity 5.6.0b10 setting stencil states via MPB appears to be broken. Hopefully it is a bug that Unity can fix in the future.

When run, each scene provides some basic UI on screen, consisting of some or all of the following options;

- Enable/disable help display.
- Enable/disable framerate counter display.
- Toggle Edge Detection (dx9 only in Unity 5.2.2f1, but fixed by Unity 5.4.2f2)
- Toggle DepthMaskCube – ensures correct draw order of stencils in complex scenes.
- Toggle Portal Culling – performance improvements.
- Toggle Glass Walls.
- Glass Wall Strength – the strength of the distortion effect on the glass.

7.0 Features

7.1 Stencil Viewer

The stencil viewer is a simple class and shader that will provide visual feedback as to where stencils are being rendered in the game view.

It should be noted there does not appear to be an efficient approach to this, so you'll notice in the script it will render a full-screen quad for each stencil reference value in turn from (0 to 255 or the number of colours specified).

7.2 Edge Detection Image Effect

It became apparent that Unity's Edge Detection image effect failed to work correctly with the stencils. After some digging around I was able to discover the problem and solution.

In forward rendering Unity will render a custom DepthNormals texture map that the edge detection effect will use. It does this by re-rendering the view from the camera using shader replacement with 'Camera-DepthNormalTexture' (later versions renamed to 'Internal-DepthNormalsTexture'). However as the subshaders in this file are written for Unity's own shaders they do not recognise the stencil state changes made to the projects shaders.

The fix is relatively straight forward, add new subshaders to the file, with custom renderType tags that our stencil shaders will use. Now when Unity renders the depthNormals texture the stencil values are included and used correctly so it matches the scene.

Alas in Unity 5.2.2f1 this affect is only supported in dx9, by Unity 5.4.2f2 it looks as though issues with other api's were fixed and it works in dx9, dx11 and openglcore on Windows, Mac untested at this time.

7.3 Portal Culling

It should be obvious that a maximum of two sides of each portalRoomCube can be view at one time. It is therefore wasteful to render the contents of the other two sides and the room contents they relate to.

The portalCulling script performs a simple dot product check between the camera forward direction and the direction from the camera to the portal mask center. If this is more than 90 degrees we know that the portal cannot be seen and we can disable all the gameobject content for that respective portal room.