

# Portal Techniques

## PRELIMINARY ONLY – DO NOT PRINT

Subject: Various Techniques for creating Portals  
 Author: VideoLogic Ltd (PowerVR)  
 File Name: CGDCMB99.doc  
 Issue Number: 1.0.65  
 Issue Date: 02 March 99  
 Copyright: © VideoLogic Ltd 1999. All rights reserved.  
 This material may not be republished in any format without the express consent of its authors. Information contained herein is provided “as-is”, without representations or warranties, and is subject to change without notice.

### Introduction

This document describes both the types and various techniques of implementing portal effects. A set of terms is defined to help in the discussion. Each of the basic types of portals is described. These include the *room portal*, the *mirror or TV portal*, a *glass pane portal* for reflective and transmissive glass and a science-fiction-like *transport portal*.

These definitions and descriptions are used to describe many different ways to create the portals, including Z-buffer clearing, stencils, rendering to a texture, Z-partitioning and software clipping and 3D volumes.

There is additional discussion, where applicable, about implementation details for traditional 3D renderers and PowerVR Series1™ and Series2™.

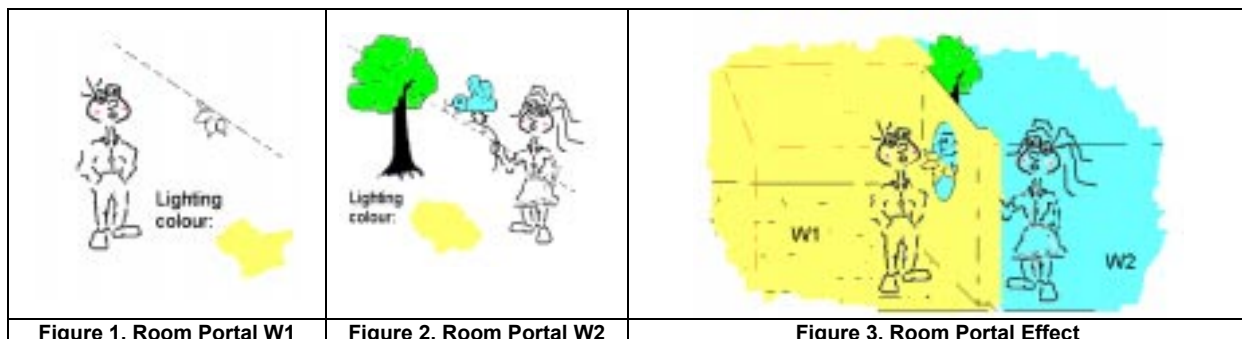
This paper was presented as a lecture at the Games Developers Conference in March 1999.

### Definitions

- W1 (World 1) The world we're in, or the normal render
- W2 (World 2) The world through portal, or the special render
- Crossing Objects Special care is needed when considering any objects that pass between the worlds across the portal boundary
- Room Portal Quake style portals
- Mirror or TV Portal Portals that need the second world clipped to the portal surface.
- Glass Pane Portal Portals that need the two worlds blended together before they are clipped to the portal surface
- 3D Portals Making the portal completely general. The portal is not just a surface, but is a volume that the W2 is present in.

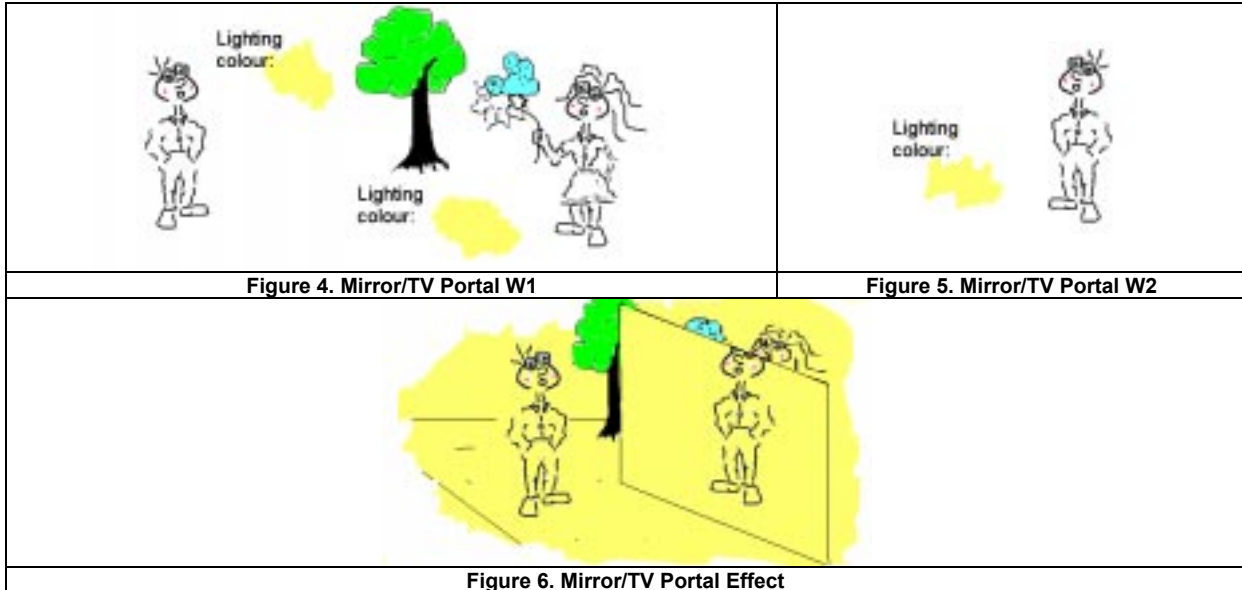
### Various Portal Types

#### Room Portal



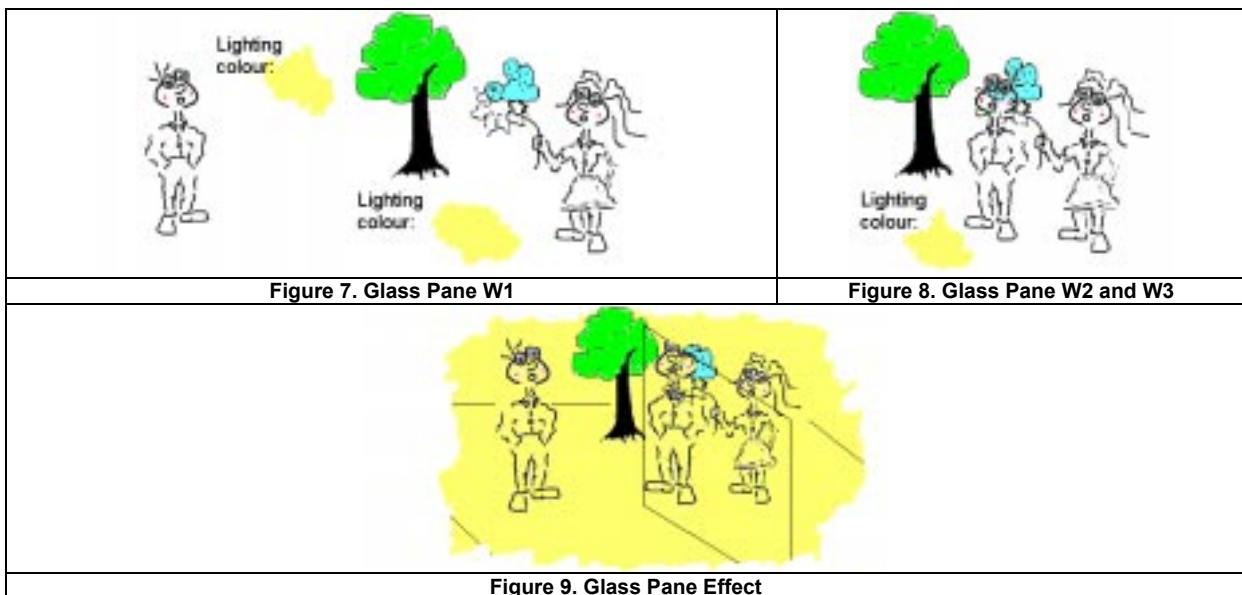
The key elements to this type of portal are that the worlds have opaque walls everywhere other than at the portal itself. Also the W1 and W2 models do not have any objects that exist in the other world co-ordinates. The models are explicitly clipped with respect to each other. This type of portal is easy to render due to the constraints mentioned. A render order will generate the correct image. This type is currently used in many games already and is well understood. Thus this type of portal will not be discussed in any more depth here.

**Mirror/TV Portal**



The key addition to this type of portal is that objects in W1 can be behind the portal surface. This presents two issues. The first is that all the objects in W2 must be rendered in their own Z comparison so that they do not interact with W1 objects. The second issue is to create a 3-dimensional clipping of W2 that will make it only visible from within the portal bounds. As the portal is a 100% reflective mirror or a TV surface, the image will be rendered with the expected lighting of the world and will not have to be affected by the portal.

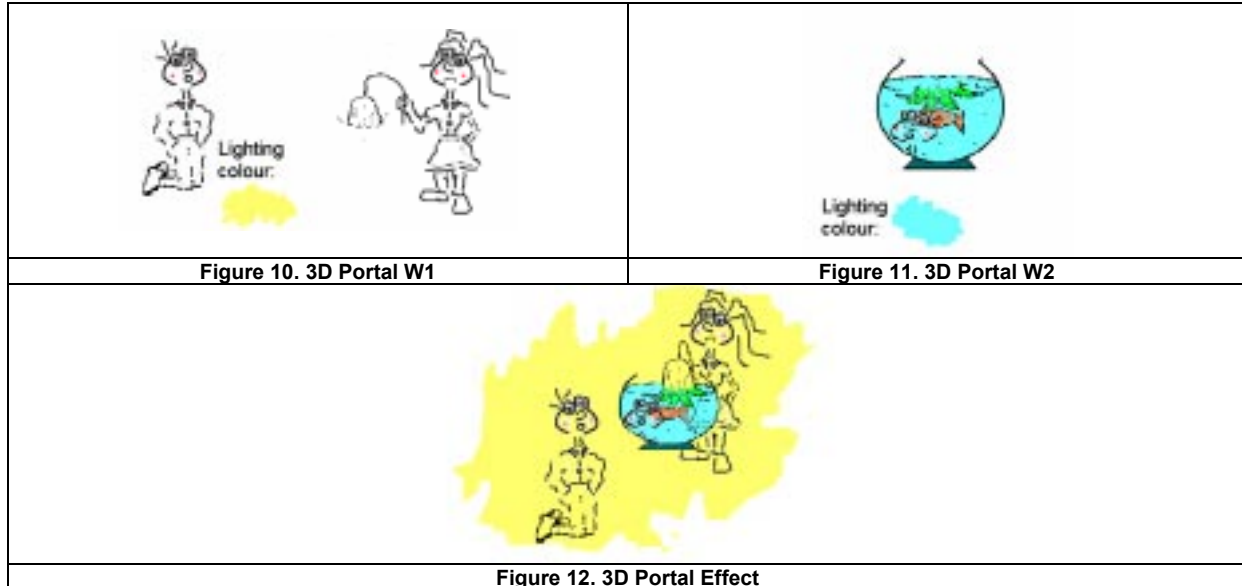
**Glass Pane Portal**



Allowing the portal to be partially reflective and transmissive creates additional issues. The items in W1 that are behind the portal must be treated differently. They need to be blended with the image that is created by the reflection. Note also that this reflection of W1 uses a blend as well. Assuming an approximation to normal glass, this can be represented by an alpha, 1- alpha blend. Note also that W1 behind the portal (referred to as W3)

needs to be clipped and rendered separately so that the results can be used in this blending process. The biggest difficulty comes when the reflection is rendered over W3. The W2 render needs to be done to a temporary location so that the final results of both W2 and W3 are blended together.

### 3D Portal



A portal can be considered an object that occupies space. A 3D portal uses the effect of W2 entering into W1 only in the volume of the portal itself. This raises the issue that the W1 objects must be seen behind the portal and potentially through it as well. If the portal is not a convex hull, then the worlds may be intertwined. Again, it may be simpler to refer to W1 behind the portal as W3.

### Notes on Culling

In all the discussions, it is assumed that gross culling of the objects in the worlds is performed based on the arbitrary viewing frustum created by the portal boundaries. This may be nontrivial, but it is not discussed here. The whole world can be submitted as a first approach to test for the desired results.

### Notes on Crossing Objects

In each type of portal, objects near the portal surface need to be dealt with carefully. Due to the discontinuity of the lighting between worlds and the possibility of objects in W2 partially obscuring a crossing object, any crossing object is added to both worlds unless there are constraints that allow for a simplification. These constraints will typically be under the control of the game and thus these issues will be left to individual requirement and implementations.

### Various Portal Techniques

For each of the types of portal above there are different implementations available. The following will be discussed here: Z-buffer clearing, stencils, rendering to a texture, Z-partitioning and software clipping, and 3D volumes. Not all possible combinations and issues are discussed.

### Z Buffer Clearing

The glass pane portal has extra work to be done above the mirror/TV portal implementation and will be discussed separately. The stages are outlined with complex stages described in more detail below.

#### Mirror/TV Portal

1. Submit W2 objects.
2. Perform Z-clearing to a depth of infinity.

3. Submit the portal surface.
4. Submit W1 objects.

***Glass Pane Portal***

1. Submit W1 objects.
2. Perform Z-clearing to a depth of infinity.
3. Submit W2 objects as depth only.
4. Submit W2 objects with a blend mode.
5. Perform Z Clearing to a depth of infinity.
6. Submit the portal surface.
7. Submit W1 objects.

***Perform Z-Clearing to a Depth of Infinity.***

This can be done two ways. The first is to use a full-screen fully transparent polygon, and the second is to use a Z-clear of the Z-buffer. The result is the current image left in the frame buffer and the Z-buffer in an initialised state. This allows all objects in the following pass (W1) to appear in front of the objects that have already been sent (W2).

If the objects rendered so far are entirely within the portal's boundaries or the objects have already been clipped by some means explicit or implicit, then this stage is not needed. (Aaron: I do not think using the Z compare function set ALWAYS is required here.)

The full screen polygons will be more likely to work on all hardware, but the Clear2() call with the D3DCLEAR\_ZBUFFER flag would be the cleaner implementation. If the portal is small relative to the screen, then this stage is the one that will be the largest cost on most hardware.

***Submit the Portal Surface.***

Submit the surface of the portal as a fully transparent object. This will bring forward the Z values to protect the image in the frame buffer that is under the portal. This is the clipping phase that is done on W2. Any objects that are in W1 and behind with portal will be clipped by these values.

This stage should only use fully transparent polygons. If the portal has other properties such as a textured surface, this is best treated independently.

***Submit W2 Objects as Depth Only.***

The glass pane portal requires the objects from the clipped W1 and W2 to be blended together. However, this is required as a blend after the two worlds have been rendered. This cannot be done using simple blends at the same time as the Z compares are being done for W2. Any polygon that does not contribute to the final Z, and thus what should be the final colour, will have already 'corrupted' the clipped W1 image with a blend. The simplest description of this is that the images need to be rendered into their own scratch frame buffers and then blended together afterward.

However, there is an approach using only Z operations that will achieve the same result. The W2 objects are submitted as depth-only so that the final result of the W2 objects is built up in the Z-buffer. Essentially the Z-buffer is being used as the scratch surface. Submitting the objects as depth-only can be done by disabling frame buffer write cycles (if available in the API) or as a render with the blend (0, 1).

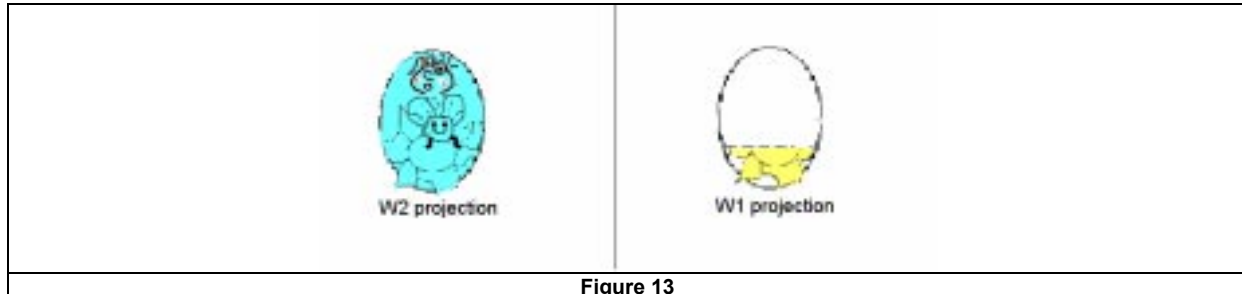
***Submit W2 Objects with a Blend Mode.***

Using the appropriate blend mode (alpha, 1-alpha) blends the final result of W2 with the final result of W1 already in the frame buffer.

Note that as the Z has already been set to the top-most polygons, a normal render with Z-compare set to LESSTHANOREQUAL will render the top-most polygons, as it will pass only the EQUAL values, the ones required.

### Crossing Objects

For the case of an object passing through the portal, i.e. the flower in Figure 13, the portion in front of the portal will be Z-clipped and will not obscure the fly. Note that if the crossing objects were not added to both worlds, the flower would retain the W2 lighting.



### Full Screen Coverage

If the W1 render, including the portal, does not cover the screen, it is possible that areas of the W2 objects that are not behind the portal polygon will also be seen. The Z-clearing simply allows these areas to be overwritten by W1. A full-screen opaque background polygon, at distance infinity, would make sure that there is always pixels in W1 that will be drawn.

### Implementation Issues

For PowerVR Series2 using PowerSGX Direct2™, the Z-clearing should be performed as a polygon sent by the user and a NEWPARTITION indicated.

For PowerVR Series1, Z Clearing is not really viable because translucent polygons are rendered after the opaque polygons. Draw order could be maintained by forcing all polygons in W1 to be translucent, but for a typical scene this would decrease the performance too much.

### Stencils

Stencils are naturally built to generate clipping information. It can be considered that this is a 2D clip, however it is the full 3D clip that is required when combined with the Z-compare. As the Stencil of only 1 bit is required and is typically read and written at the same time as the Z-buffer this method can save the time the Z-clearing technique would have to use on the full screen Z-clear.

Stencils are very powerful because the comparisons and setting can be controlled completely, which allows for a result identical to this example, except sending W1 first followed by W2. However, this requires more stencil reads and writes and is not a natural way to consider the total scene, so it is not described here.

### Mirror/TV Portal

1. Submit the portal surface with Z-writes off and set the stencil bit.
2. Submit W2 objects with a stencil test.
3. Submit the portal surface.
4. Submit W1 objects.

### Glass Pane Portal

1. Submit the portal surface with Z-writes off and set the stencil bit.
2. Submit W1 objects with a stencil test.
3. Submit the portal surface at a depth of infinity.
4. Submit W2 objects as depth only with a stencil test.
5. Submit W2 objects with a blend mode with a stencil test.
6. Submit the portal surface.
7. Submit W1 objects.

**3D Portals**

1. Submit any object lying behind W2 that could possibly be seen through W2 (the girl) with yellow lighting. (This is so we see the girl through the fish bowl.)
  2. Do a Z-blit.
  3. Submit W2 objects (water, fish, flower and boy's reflection) with blue lighting.
  4. Do a Z-blit.
  5. Submit the transparent polygons depicting the W2 clipping region (after calculating what this region should be.)
  6. Submit W1 (boy, fish and flower) once again with yellow lighting. (Won't overwrite the W2 region.)
- (Carlos: The above is Aaron's, what is the order you do for Scanner Stencils?)

**Submit the Portal Surface with Z Writes Off and Set the Stencil Bit**

Where the pixel is covered by the portal surface only, the stencil bit will be set. The frame buffer and Z-buffer will not be updated. This is the stage that is setting up the clipping.

(Carlos: What D3D operations are needed.)

**With a Stencil Test**

On stages that require this, continue with the pixel render only if the stencil is set.

(Carlos: What D3D operations are needed.)

**Submit the Portal Surface at a Depth of Infinity**

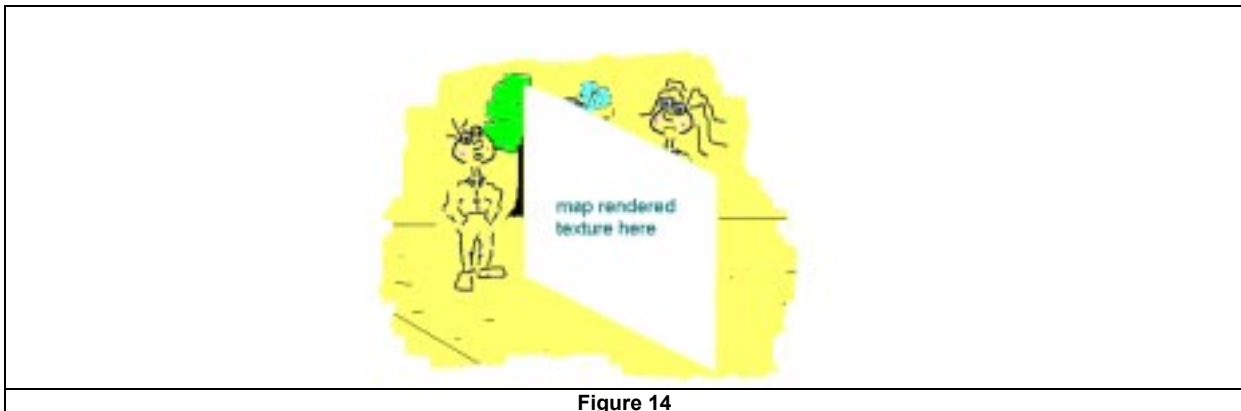
Submit the surface of the portal as a fully transparent object with Z values set at infinity. This will perform a local Z-clear and save the image results in the frame buffer for later blending.

**Implementation Issues**

For the limited set of stencil operations that are used for portals, PowerVR Series2 can use modifier volumes as an exact replacement.

**Render to a Texture**

Render to a texture is a technique that simplifies the understanding of the portal process if not the implementation. The W2 image is rendered to a texture as a scratch surface so that it can be used in the normal render as the texture for the portal surface. This allows for a lot of flexibility.



**Figure 14**

**Mirror/TV Portal**

1. Submit W2 objects rendering to a texture.
2. Submit the portal surface using the rendered texture.
3. Submit W1 objects.

**Glass Pane Portal**

1. Submit W1 objects rendering to a texture.
2. Submit W2 objects rendering to a texture.
3. Submit the portal surface with the two rendered textures blended.
4. Submit W1 objects.

**Rendering to a Texture**

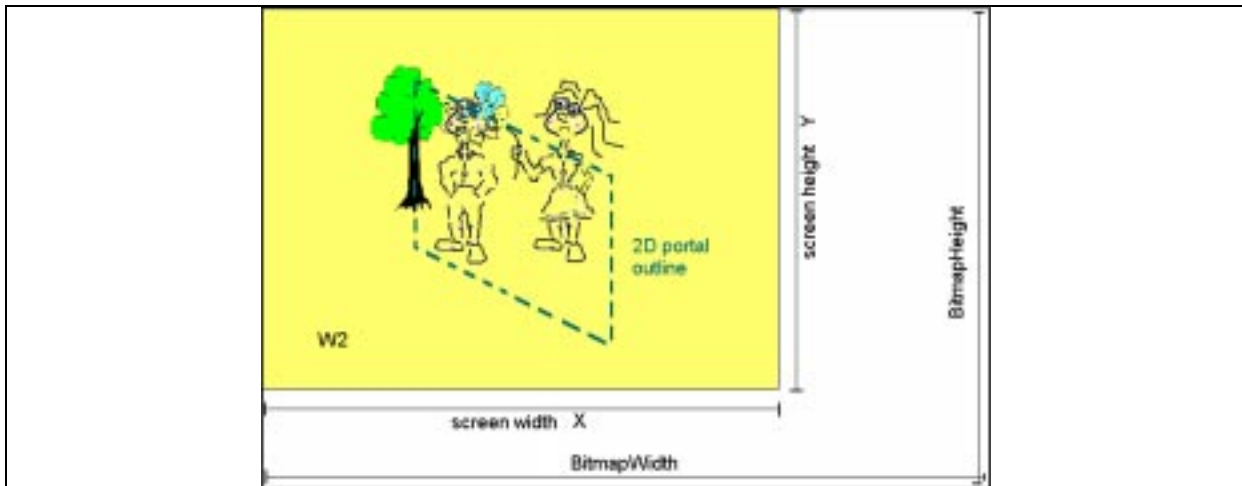
The target of the render needs to be a 3D surface and a texture at the same time.

(Aaron: what are the D3D flags/calls for this?)

**Using the Rendered Texture**

When the rendered texture is to be used, the UV co-ordinates are determined by the screen positions  $[X_{Si}, Y_{Si}]$  and texture positions  $[X_{Ti}, Y_{Ti}]$  of the portal vertices, as in Equation 1 and Equation 2.

(Aaron: This needs more explanation. How do I go from the portal XY screen vertices to the UV?)



**Figure 15**

$U_i = \frac{X_{Si}}{X_{res}} = \frac{X_{Ti}}{BitmapWidth}$	$V_i = \frac{Y_{Si}}{Y_{res}} = \frac{Y_{Ti}}{BitmapHeight}$
<b>Equation 1</b>	<b>Equation 2</b>

**Texture Size, Screen Size and FOV**

Most hardware renderers are optimised for rendering from square textures of width a power of two. In these cases, the perspective transform used when rendering to the texture includes the FOV for the square texture and not the screen resolution. Thus, it will be rendered with an aspect ratio of 1.333 due to a screen size of about 640x480.

**Implementation with Independent Z and RHW**

Most rendering systems use Z and RHW (A.K.A. InvW, 1/W) parameters; Z is used for the depth sort and RHW for perspective correction of interpolated values, such as texture co-ordinates. This independence allows the RHW values to be altered without changing the depth values of polygons.

The when the rendered texture is used as the texture for the portal polygon, the RHW values should all be set to the same value, e.g., 0.5, in order to prevent perspective correction.

This method can be used on systems that supply both Z and RHW values. Unless W-buffering is used, this includes all PC-based Direct3D™ systems.

**Implementation with Nonindependent Z and RHW**

For systems that only use the RHW parameter, for both depth sorting and perspective correction, the above technique cannot be used.

For this case, the portal surface has to be rendered twice, once for the texture and then for the depth. The example above becomes:

1. Submit W2 objects rendering to a texture.
2. Submit the portal surface using the rendered texture at a depth of infinity.
3. Submit the portal surface.
4. Submit W1 objects.

Note that the portal polygon is drawn first to the normal surface to make sure the draw order is correct.

This system can be used on systems that only supply the RHW values. This includes Direct3D in W-buffering mode, and Dreamcast™ systems. Note one of the main reasons for using a system with only RHW values is to decrease the parameter bandwidth to obtain the maximum performance.

**Z Partitioning and Software Clipping**

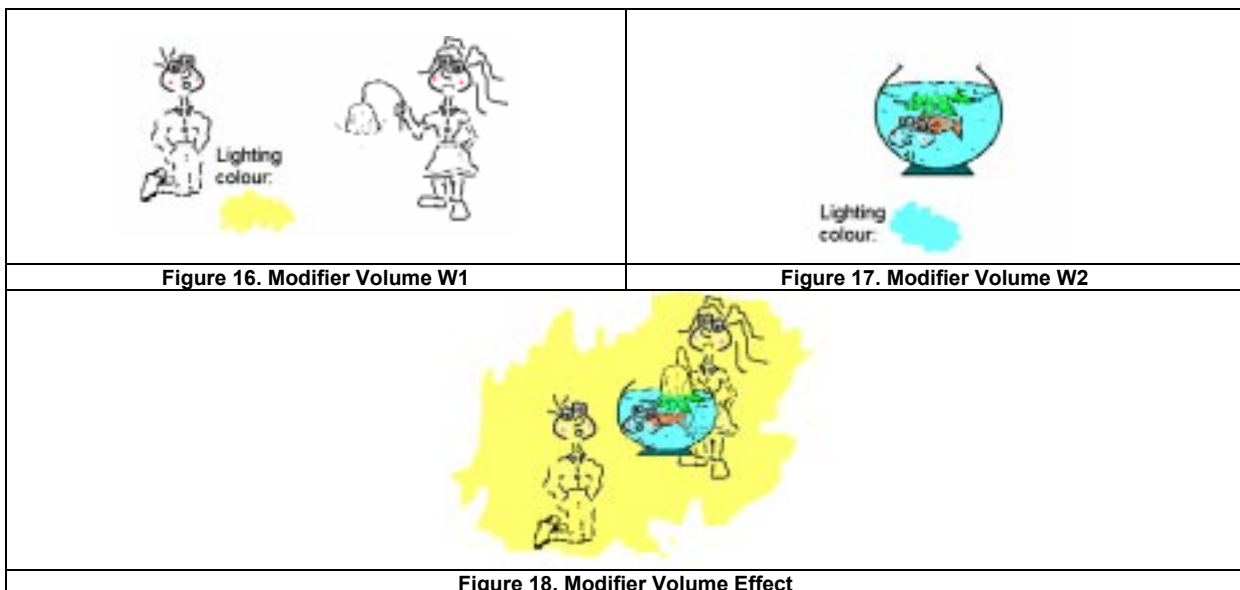
The W2 objects can be rendered normal by having software perform clipping to the portal boundaries. Once the W2 objects are clipped they need to be rendered together with the W1 objects. To make sure that they do not interact with one another, the Z values of each world should be partitioned. Z-partitioning is discussed in full in another document.

Note that during the clipping the full generation of new vertices is required, including interpolating for the new vertex colors, texture co-ordinates and fog values.

The software clipping can be slow, especially if the portal shape is complex. However, this method has the advantage of being guaranteed to work with any 3D hardware. This is already in common use, for example, as a rectangular rear view mirror and so will not be discussed any further here.

**3D Volumes**

PowerVR Series2 (including Dreamcast) has support in hardware for 3D Volumes or Modifier Volumes. These allow a model to be sent as part of the scene that represents a volume that other objects are inside or outside of. Normal objects can then be given two sets of parameters, each being used depending on the objects being inside or outside of the modifier volume.



If W2 is a 3D portal rather than a 2D portal like a mirror or glass pane, modifier volumes provide an easy and single-pass solution.

**3D Portals**

1. Submit the modifier volume.
2. Submit W2 with two sets of parameters.
3. Submit W1.

**Submit W2 with Two Sets of Parameters**

Objects are given two sets of base and specular colours, vertex alphas and texture co-ordinates.

For this example, it is necessary to make the boy's reflection completely transparent outside the volume, and maybe semitransparent when within. Setting the vertex alphas to translucent and opaque respectively, with all other parameters identical, will suffice. The flower is visible both inside and out; only the lighting effects change. Setting the base colours to blue and yellow will give the necessary effects.

The fish resides entirely within the volume, making it possible to use a simple object approach with only one set of colours and one texture.

**Demonstrations**

These demonstrations are taken from the Series2 PowerVR SDK. Source code is included.

**FilmTV – A Mirror/TV Portal Example**

This scene tries to show the elements that are required for this type of portal in the scene itself. This makes it rather recursive, which results in the 70s special effect. The scene that is to be W2 is just another camera position in the original W1. There is a modelled camera to show where this is. The TV and the projection screen are showing the results of that camera position.

Note that the scene is generated with the previous frames results used in the next frame. This is what makes the recursive image spiral. This example simulates what happens when this is done with a real camera and TV. The TV is showing an image that has already been recorded by the camera. The camera is now recording the next image with the old image on the TV screen.

This is shown using each of the techniques, in order of expected performance: Z-buffer clearing, stencils and rendering to a texture.

**Glass Pane – A Glass Pane Portal Example**

This example shows a simple set of objects with clear colouring so that the results can be easily understood. The glass pane has a variable reflectivity/transmissivity. We can also control colour of the glass to make it tinted. There is also a texture on the mirror as a patina.

The difference in visual quality can be seen between the techniques of Z-buffer clearing and render to a texture, due to the rendered texture being expanded upward. The Z-buffer clearing is expected to be a slower technique on most 3D hardware due to the additional memory write cycles required.

**Scanner – A 3D Portal Example**

The scanner beam is acting as 3D portal. Note that the 3D portal or beam itself is being made visible by the addition of a translucent model that is exactly the same shape as the portal. This is equivalent to having a patina on the mirror to show its presence in a form other than its effect on the world. The W2 is a rendered world, which coincides with the W1 models. The only difference is that the dinosaur skin in W2 is transparent and whitened in colour. Thus, you can see W2 wherever the 3D portal encloses in W1.

This demonstration uses PowerVR modifier volumes and stencils. As stencils require more than one render of the world compared to sending extra data during the one render for modifier volumes. The single-pass modifier volume technique is expected to be faster.

## **Summary**

Portals are great visual features that will be required more and more as the realism of games is increased. The type and technique of portal used should be selected with some care. This can be done by selecting the complexity of the type of portals and balancing this with the technique making the decision based on the visual quality and performance trade-off.

We personally prefer rendering to a texture, as this technique allows for the maximum flexibility and performance for an acceptable visual quality in most cases.

## **Questions?**

**NEC**

**VideoLogic**