Accurate Reflections and Refractions by Adjusting for Object Distance

Chris Brennan

Introduction

Environment mapping is a technique by which a texture map or maps contain the view of the environment from a particular point. The texture map is then addressed using a function of a reflection or refraction vector when rendering an object. There are several forms of environment mapping: spherical, dual paraboloid, and cubic, to name a few. They are all basically the same concept, differing only in what function is used to create and address the map. Cubic environment mapping is a method of sampling a texture consisting of six renderings of the world from the center of a cube out through each of its faces. The texture is addressed with a vector that indicates the direction from the center of the cube to the sample point on one of its faces. Therefore, a calculated reflection vector can be directly used as an address into the environment map. Because of these simple properties, cube maps lend themselves to dynamic real-time creation and addressing within pixel shaders. The following artifacts apply to all environment mapping, but this example is applied to cubic environment maps.

The Artifacts of Environment Mapping

One of the common problems seen when environment mapping is that flat surfaces do not render well. There is very little variation in the reflection vector direction itself, and because environment maps are indexed from the exact center, it does not take into account that a reflection vector can be originated off-center and therefore end up reflecting a different object. This deficiency would make a flat mirror appear to reflect all the same point. The result is that only a few texels are magnified and used in the final image. This artifact is worse the closer that reflected objects are relative to the viewer. The same artifact appears when using a refraction vector as opposed to a reflection vector, but it is more obvious where the transparent object meets the object behind. See Figure 1 for an example of the artifact as seen through a window using a refraction vector. Notice how the part of the chest seen through the window does not properly match the chest itself.

Team LRN

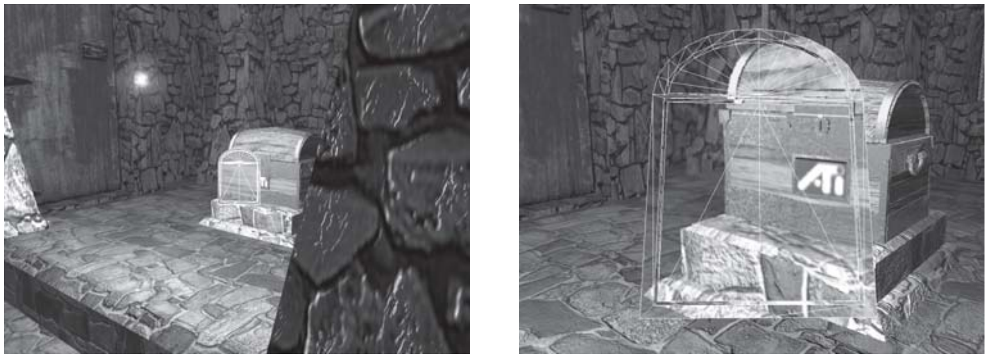

Figure 1: This example shows a windowpane with environment map artifacts that produce incorrect looking transparency.

Environment mapping based solely on the reflection vector direction acts as if the objects in the environment map are infinitely far away. *Advanced RenderMan* [Apodaca00] describes a way to alleviate this by adjusting the use of the environment map such that it has a finite radius that is close to the real proximity of the objects. The most precise way to do this is by intersecting the reflection or refraction ray with the environment map sphere and then using the vector from the center of the map to the intersection point (*CI*) to index the map. See Figure 2.

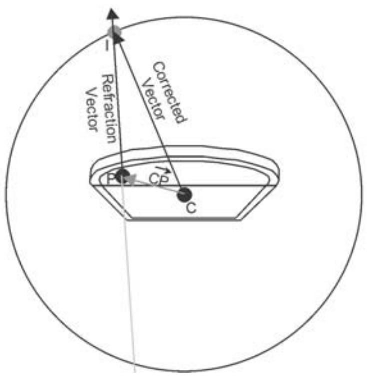

Figure 2: Corrected vector based on an intersection with a finite radius

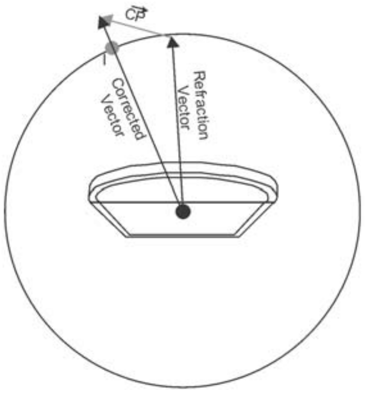

Figure 3: Corrected vector adjusted by a vector based on the point's distance from center

Calculating the intersection of a circle and a vector is difficult in a pixel shader, but another approximation of the same goal can be achieved by adding a scaled version of the vector from the center point of the environment map to the point on the object being drawn (*CP*). Figure 3shows how the approximate corrected version of the vector intersects the same point *I* on the environment map. The scaling factor is 1/(environment map radius) and is derived from:

$$\overrightarrow{R'} = \vec{R} \cdot Radius + \overrightarrow{CP}$$

Team LRN

which when divided by the environment map radius changes to:

$$\vec{R''} = \vec{R} + \frac{\vec{CP}}{Radius}$$

The total correction factor of *CP*/*Radius* can be calculated in the vertex shader and interpolated for adding in after the per-pixel reflection calculation. The following figure illustrates the amount of correction applied.

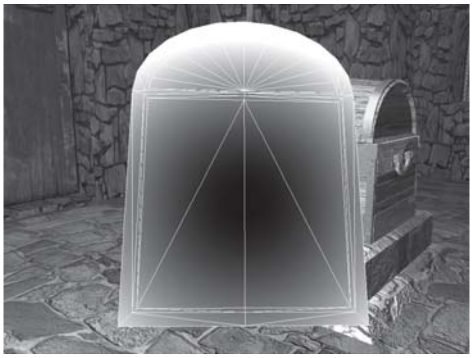

Figure 4: The magnitude of the correction vector displayed as grayscale intensity.

The corrected version of Figure 1 is shown in Figure 5:

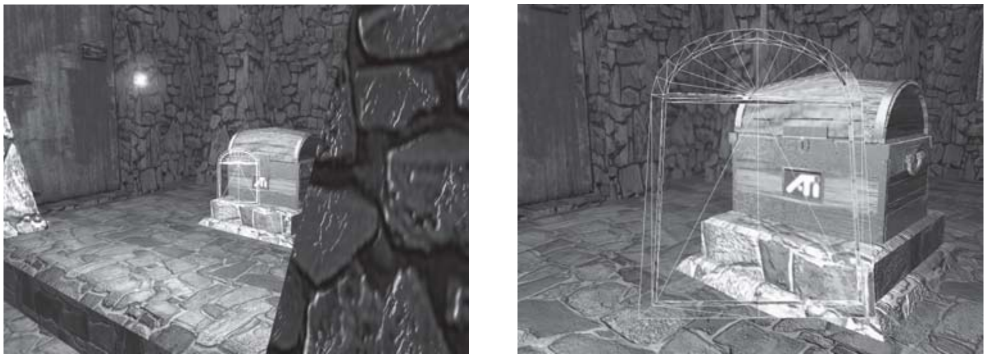

Figure 5: The adjusted version of the refraction vector shows the geometry behind at the correct scale at all viewing distances.

Notice how the refracted image now matches the geometry behind it. The correction can be applied per-pixel as seen in Figure 6 and Color Plate 4.

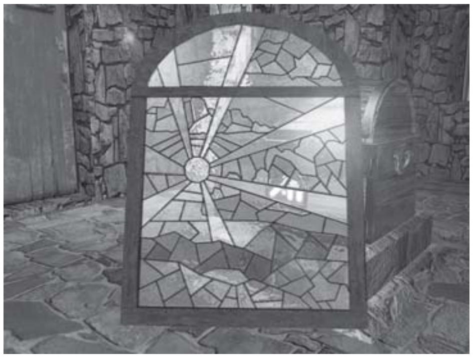

Figure 6: Stained glass color plate

The composite effect was rendered with the following shader:

```
; Sample vertex shader which outputs a 3x3 orthogonal transform which
; takes you from world space to tangent space.  The result is stored in
; output tex coords 1,2,and 3. Tex coord 4 stores a view vector which
; can be used in a reflect operation.
;
; Tex coordinate 5 stores a scaled version of the centroid to vertex
; vector that is used as a correction factor for environment mapping.
;
; This sample expects the incoming vertex data to be set up as follows:
;
; v0     - Vertex Position
; v3     - Vertex Normal
; v8     - Vertex Tangent U
; v9     - Vertex Tangent V
; This sample expects the constant store to be set up in the following manner:
;
; c1       - Color of material
; c12..c15 - World Matrix
; c16..c19 - Composite World-View-Projection Matrix
; c20..c23 - Env Transform Matrix  (object to world or view coord transform
;             (depends on type of env mapping you want to do) )
; c28      - ModelSpace Camera Position
; c95      - 0, .5, 1, 2
vs.1.0
def c94, 0.18f, 0.18f, 0.18f, 0.18f  ; Scalar for cube map adjustment
                                     ; = roughly 1/Radius of environment map.

; OutPos = ObjSpacePos * Composite World-View-Projection Matrix
m4x4   oPos, v0, c16
mov    oT0.xy, v7          ; passthrough texcoord 0 for bump map addressing

;derive per vertex eye vector using camera position
add     r8, c28, -v0       ; view direction
m3x3    r9, r8, c20        ; transform view vector from object space to env space
dp3     r11.x, r9, r9      ; magnitude^2
rsq     r11.y, r11.x       ; 1/mag
mul     oT4.xyz, r9, r11.y ; normalize

; upper 3x3 of world matrix by upper tangent space transposed
dp3     oT1.x, c20, v8
dp3     oT1.y, c20, v9
dp3     oT1.z, c20, v3

dp3     oT2.x, c21, v8
dp3     oT2.y, c21, v9
dp3     oT2.z, c21, v3

dp3     oT3.x, c22, v8
dp3     oT3.y, c22, v9
dp3     oT3.z, c22, v3

; Calculate an adjustment amount to account for cube map center not
; being at this object's center.
m3x3 r0, v0, c12    ; Object->World space position without translation.
mul  oT5, r0, c94.w ; Scale by 1/appoximate scene size

ps.1.4

def     c0, 0.9f, 0.0f, 0.0f, 0.0f
texld   r0, t0              ; Look up normal map.
texcrd  r1.rgb, t4          ; Eye vector
texcrd  r4.rgb, t1          ; t row of environment matrix
texcrd  r2.rgb, t2              ; 21sst row of environment matrix
texcrd  r3.rgb, t3              ; 3rd row of environment matrix
texcrd  r5.rgb, t5              ; Cube map position adjustment

dp3     r4.r, r4, r0_bx2        ; 1st row of matrix multiply
dp3     r4.g, r2, r0_bx2        ; 2nd row of matrix multiply
dp3     r4.b, r3, r0_bx2        ; 3rd row of matrix multiply
lrp     r2.rgb, c0.r, -r1, -r4  ; Refract by c0 = index of refraction fudge factor
add     r2.rgb, r5, r2          ; Adjust for cube map center
phase

texld   r2, r2                  ; Sample cubic reflection map
texld   r3, t0                  ; Sample base map
texld   r4, r4                  ; Sample cubic diffuse map
texld   r5, t0                  ; Sample gloss map

mul     r1.rgb, r5, r2          ; Specular = Gloss*Reflection
mad     r0.rgb, r3, r4_x2, r1   ; Base*Diffuse + Specular
+mov    r0.a, r3
```

The bold lines of the shader are the required lines for the environment map adjustment. This technique makes reflections look more accurate, but it also makes refraction possible because the eye is much more sensitive to the inaccuracies of refraction. This technique, including shaders and source code, is used in the Treasure Chest demo available on the companion CD and also at http://www. ati.com/na/pages/resource_centre/dev_rel/Treasurechest.html. There are menus to view the reflection cube map that can be accessed after going to windowed mode by pressing Alt+Enter.

References

[Apodaca00] Anthony A. Apodaca and Larry Gritz, *Advanced RenderMan* (Morgan Kaufmann Publishers, 1999), pp. 212-215.