Chapter 1

Efficient Self-Shadowed Radiosity Normal Mapping

Chris Green[1]

VALVE

Normal Mapped

Normal mapped with ambient occlusion

Self-shadowed

Figure 1. *Comparison of surface illumination techniques.*

1.1 Abstract

In Valve's *Source* graphics engine, bump mapping is combined with precomputed radiosity lighting to provide realistic surface illumination. When bump map data is derived from geometric descriptions of surface detail (such as height maps), only the lighting effects caused by the surface orientation are preserved. The significant lighting cues due to lighting occlusion by surface details are lost. While it is common to use another texture channel to hold an "ambient occlusion" field, this only provides a darkening effect which is independent of the direction from which the surface is being lit and requires an auxiliary channel of data.

In this chapter, we present a modification to the *Radiosity Normal Mapping* system that we have described in this course in the past. This modification provides a directional occlusion function to the bump maps, which requires no additional texture memory and is faster than our previous non-shadowing solution.

[1] email: cgreen@valvesoftware.com

1.2 Introduction

In order to increase surface detail and perceived realism, bump mapping is used heavily in modern real-time 3D games [Blinn78] [PAC97]. Bump maps are often used as an approximation of higher detailed geometry which would either be to slow to render in real time or too memory intensive to store reasonably. However, one weakness of bump maps is that they only modify the surface normal which is used for lighting computations. While this provides a realistic directional lighting cue, effects such as self-shadowing of surface details and ambient occlusion are not rendered.

Traditional bump mapping also cannot be combined with conventional precomputed light maps, a technique in which a global illumination solution is generated as a precomputation and then stored in a low resolution texture which is used to modulate the brightness of the surface texture [ID97].

With *Radiosity Normal Mapping*, the precomputed light map information was extended to encompass lighting from multiple directions and allowed arbitrary bump mapped data to be combined with precomputed lighting textures [McTaggart04] [MMG06]. Using Radiosity Normal Mapping, the distribution of incoming distributed lighting can be stored in many possible bases [Sloan06], with the tangent-space surface normal evaluated per pixel and then convolved with the incoming light distribution for each output pixel.

In this presentation, we extend Radiosity Normal Mapping by modifying the bump map representation to be pre-convolved against the tangent-space lighting basis in order to gain efficiency and reduce rendering artifacts. We then extend this pre-convolution to take into account directional self-occlusion information, allowing for both occlusion of isotropic ambient lighting and dynamic directional lighting.

1.3 Related Work

Many different techniques have been used to add shadowing information to bump mapped surfaces for real-time rendering. Horizon mapping augments bump map data by precomputing and storing the angle to the “horizon” for a small set of fixed tangent-space directions and uses this representation to produce hard shadows [Max98] [SC00]. In [KHD00], an oriented ellipse is fitted to the distribution of non-shadowed lights over a bump map texel. In [OS07], a spherical cap is used to model the visible light directions over a bump map texel, and this data is used to render hard and soft shadows from point and area lights in real time.

Recently, techniques have been developed for direct rendering of height fields in real time using graphics hardware [POC05] [MM05] [Tatarchuk06]. Since these techniques are able to compute visibility of height field texels from any viewpoint, they are also able to implement shadowing of height fields by computing visibility to light sources.

1.4 Representation and Generation

We wished to implement self-shadowing of bump maps that would

- Mesh well with our existing radiosity-lit bump map approach
- Work on older generations of graphics hardware as well as current systems.
- Run as fast as, or faster than our current non-shadowed solution.
- Improve bump map anti-aliasing
- Work with dynamics lights as well as our pre-calculated radiosity lighting
- Provide soft shadows and ambient occlusion
- Allow shadowing information to be generated either from height data or from arbitrary geometry
- Not use any increased texture storage space compared to ordinary bump mapping. In particular, we wanted to be able to preserve existing uses of the alpha channel of bump maps as a mask for various effects.

We successfully implemented a method to generate diffuse soft shadows from static and dynamic light sources, with no increase in bump map storage space, while providing an actual performance increase over our existing non-shadowed radiosity bump map implementation. The source engine calculates lighting at each surface pixel using the following operations: `normal = 2.0 * normalTexel - 1.0`.

$$\vec{B}_{0..2} = \left\{ -\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}} \right\}, \left\{ -\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}} \right\}, \left\{ -\frac{\sqrt{2}}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}} \right\} \quad (1)$$

$$\vec{N} = 2 \times \vec{T} - 1 \quad (2)$$

$$\vec{D}_i = \frac{saturate(\vec{N} \cdot \vec{B}_i)^2}{\sum_{i-0}^{2} \vec{D}_i} \quad (3)$$

$$pixelcolor = albedo \times \sum_{i=0}^{2} \vec{L}_i \vec{D}_i \quad (4)$$

where $\vec{T}$ is the tangent-space bump map texel (which must be scaled and biased before use, because texels are unsigned), $\vec{B}$ is the set of tangent-space directions for which incoming light has been precomputed, $\vec{L}$ are the 3 precomputed lighting values, and $\vec{albedo}$ is the color of the surface texel being lit. `Saturate(x)` is the HLSL function which clamps its input to be in the range between 0 and 1.

```
float3 normal = 2.0 * normalTexel - 1.0;
float3 dp;
dp.x = saturate( dot( normal, bumpBasis[0] ) );
dp.y = saturate( dot( normal, bumpBasis[1] ) );
dp.z = saturate( dot( normal, bumpBasis[2] ) );
dp *= dp;

float sum = dot( dp, float3( 1.0f, 1.0f, 1.0f ) );
float3 diffuseLighting = dp.x * lightmapColor1 +
                         dp.y * lightmapColor2 +
                         dp.z * lightmapColor3;
diffuseLighting /= sum;
```

Listing 1. *HLSL source code for original* Source *radiosity bumpmapping.*

We observed that some execution time could be saved if, instead of storing the surface normal, $\vec{N}$ in our texture maps, we instead stored the value of $\vec{D}$ in the 3 color components of the bump map. This is a trivial modification to any bump map generation program. This reduces the lighting equation to equation (4) above.

```
float3 diffuseLighting = normalTexel.x * lightmapColor1 +
                         normalTexel.y * lightmapColor2 +
                         normalTexel.z * lightmapColor3;
```

Listing 2. *HLSL source code for new directional-ambient-occlusion bump mapping*

Just doing this saves a substantial number of pixel shader instructions. However, we no longer have a tangent-space surface normal for use to calculate the lighting from dynamic lights and reflection vectors. Nonetheless, when needed, we can use our original basis directions $\vec{B}$ to reconstruct a suitable tangent-space normal for reflections. For dynamic lights, we can project the lighting direction onto our basis directions and use that directly in the lighting equation, which gives us a form of shadowing from dynamic light sources.

Once bump maps are stored in this format, some advantages are seen besides the increased pixel shader performance:

- Because the bump maps now just represent positive light map texture blending weights, no special processing is required when filtering or mip-mapping them.
- Numeric precision is increased for the radiosity bump mapped case, since we are now storing the bump maps in the exact numeric representation that their data is needed in, and since we do not have to represent negative numbers.
- Surface textures stored in this form can be processed by existing art tools with no special interpretation. For instance, filters such as sharpen and blur can be used in Photoshop™.
- Texture blending for operations such as detail texturing, texture cross fading, etc. are much more straightforward.
- Fewer aliasing artifacts will be seen when textures are minimized.

- Textures in this format can be directly generated from geometry in 3D rendering packages, by placing red/green/blue unshadowed point lights in the scene, offset along the 3 predefined basis vectors.

Ordinary bump maps can only change the apparent lighting orientation of the surface. However, when rendering with this representation, if we uniformly scale the RGB values of the normal map texels, we can provide a darkening effect which is independent of the lighting direction. This allows us to have normal maps also act to modulate surface albedo, without having to store a separate brightness channel, or change the RGB values of the base albedo texture. Since it is often possible to produce good imagery by combining a fairly low frequency albedo texture with a high frequency bump map texture, this can save us texture memory.

A common bump map production method involves taking elevation maps which are either painted, created in a modeling package, or acquired from real-word sources, and then using these elevation maps to extract a surface normal. Then, the same elevation data is used to calculate an ambient occlusion channel, which is typically generated by firing from each texel, a large set of rays. The results of these ray intersections are used to determine the cosine-weighted proportion of the hemisphere above the surface which can be seen without the surface obscuring it. The result of this ambient occlusion calculation is then either stored in its own texture channel, or multiplied into the base texture. We can do the exact same thing in our representation, except that *we can encode this channel directly into the 3 channel normal map*.

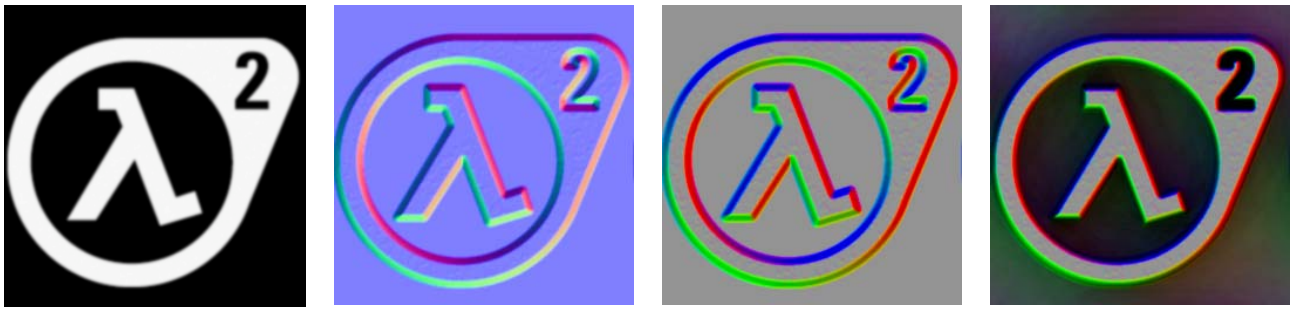

Figure 2. *(a) Height map (b) Bump map (c) Bump map stored in our basis (d) With directional ambient occlusion*

Moreover, since each of the channels of our modified bump map store the amount of light coming from 3 fixed tangent-space directions, we are able to do better than just encoding ambient occlusion. If we weight the contribution of each ray intersection test based upon the dot product between it and the fixed tangent-space direction associated with each channel, we can calculate a separate occlusion value for each channel, with that occlusion value representing a smaller angular distribution centered upon the corresponding basis vector. This gives us a “directional ambient occlusion” effect which causes ambient and direct lighting arriving from different directions to be darkened when that light would have been blocked by the self-shadowing effects of the surface. These 3 ambient occlusion directions are simply multiplied into the 3 non-shadowed bump maps we are already using in our representation. This gives us form of diffuse self-shadowing

essentially “for free”, providing self-shadowing of direct and indirect light. When these directional ambient occlusion textures are converted back into normal vectors for reflection calculations, something akin to the use of “bent normals” is achieved.

We implemented an efficient multi-threaded SIMD ray tracing system [WBW+01] in order to perform the hundreds of ray intersection tests per texel necessary to generate accurate direction ambient occlusion. Our off line generation utility takes as input an elevation map and an elevation scale factor. A user-configurable bilateral filter [TM98] is applied to the input image to reduce stair-stepping, and then 300 rays per output texel are traced in order to generate bump maps with directional ambient occlusion in our new format.

It is also possible to generate textures in this format through standard 3D rendering packages, by the careful placement of area lights in the scenes. Analogously, such maps could be captured from real world materials via photography with appropriately placed lights and reflectors.

The standard techniques for generating bump maps for a coarsely tessellated model by tracing rays against a more finely tessellated one can be easily extended to support this bump map representation, which will allow for animated articulated models with self-shadowing surface detail.

We can easily extend this technique to support more channels in order to produce more accurate lighting and shadows, at the expense of higher texture storage. Differing combinations of sample directions and lighting precomputation directions can be used, for instance to provide more accurate shadows from dynamic lights without increasing the storage needed for light maps.

Figure 3. *Cave walls exhibiting self-shadowing from the flashlight in Half-Life®2: Episode 2.*

1.5 Conclusion

A form of self-shadowing can be easily added to radiosity bump mapping by “baking” the light sampling basis into the actual bump map data, with no decrease in performance or increase in texture memory cost. We are able to make heavy use of this technique in the games Half-Life® 2: Episode 2 and Team Fortress 2.

1.6 References

[BLINN78] BLINN, J. F. 1978. Simulation of wrinkled surfaces. In SIGGRAPH ’78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 286–292.

[ID97] ID SOFTWARE, 1997. Quake 2.

[KHD00] KAUTZ, J., HEIDRICH, W., AND DAUBERT, K. 2000. Bump map shadows for OpenGL rendering. Tech. Rep. MPI-I-2000-4-001, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

[MAX98] MAX, N. L. 1998. Horizon mapping: shadows for bump-mapped surfaces. In The Visual Computer, 109–117.

[MM05] MCGUIRE, M., AND MCGUIRE, M. 2005. Steep parallax mapping. I3D 2005 Poster.

[MCTAGGART04] MCTAGGART, G., 2004. Half-life 2 shading. GDC Direct3D Tutorial.

[MMG06] MITCHELL, J., MCTAGGART, G., AND GREEN, C. 2006. Shading in Valve’s source engine. In SIGGRAPH ’06: ACM SIGGRAPH 2006 Courses, ACM Press, New York, NY, USA, 129–142.

[OS07] OAT, C., AND SANDER, P. V. 2007. Ambient aperture lighting. In I3D ’07: Proceedings of the 2007 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, 61–64.

[PAC97] PEERCY, M., AIREY, J., AND CABRAL, B. 1997. Efficient bump mapping hardware. Computer Graphics 31, Annual Conference Series, 303–306.

[POC05] POLICARPO, F., OLIVEIRA, M. M., AND COMBA, J. L. D. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In I3D ’05: Proceedings of the 2005 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, 155–162.

[SC00] SLOAN, P.-P. J., AND COHEN, M. F. 2000. Interactive horizon mapping. In Proceedings of the Eurographics Workshop on Rendering Techniques 2000, Springer-Verlag, London, UK, pp. 281–286.

[SLOAN00] SLOAN, P.-P. 2006. Normal mapping for precomputed radiance transfer. In I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, pp. 23–26.

[TATARCHUK06] TATARCHUK, N. 2006. Dynamic parallax occlusion mapping with approximate soft shadows. In proceedings of AMD SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 63-69, Redwood City, CA.

[TM98] TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In ICCV, 839–846.

[WBW+01] WALD, I., BENTHIN, C., WAGNER, M., AND SLUSALLEK, P. 2001. Interactive rendering with coherent ray tracing. Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001), v. 20, i. 3, pp. 153-164.