Modern Texture Mapping in Computer Graphics

Stephan Pajer*

Abstract

Textures have a variety of applications in computer graphics. While the original idea of texture mapping was just to cover the surface of an object with an image to avoid having to model every detail geometrically, in time a lot of other uses have been proposed. In this paper, the most important applications of texture mapping (except the original approach) will be explained.

Keywords: bump mapping, normal mapping, horizon mapping, displacement mapping, parallax mapping, relief mapping, shadow mapping, percentage closer filtering, perspective shadow mapping

1 Introduction

Since its introduction, texture mapping [Catmull 1974] has evolved from 'just' painting an image onto a geometric model. While most applications of textures still focus on the simulation of geometric detail, other applications like shadowing the scene using a texture have also been proposed. This paper will give an explanation of the most important algorithms for simulating surface details with textures.

This includes algorithms for altering the surface normals to modify the lighting and give the illusion of protrusions as well as ways to offset the rendered texels using height fields and combinations of these variants. Furthermore, the shadow mapping algorithm [Williams 1978] as well as its most important refinements will be described.

2 Exposition

2.1 Bump Mapping

Bump mapping [Blinn 1978] is an approach to simulate complex surfaces by perturbing the normal vector of a surface on a per-pixel basis. This yields good results due to the fact that the main effect of surface irregularities on the perceived intensities is due to their effect on the surface normal.

In Blinn's approach, the normal vector of the wrinkle function is calculated by taking the cross product of the functions partial derivates. The pertibation of the normal is then accomplished by adding the normal vectors of the wrinkle function to the normals of the smooth surface and illuminating the object as usual.

While Blinn's algorithm does improve the quality of the rendered image, there are a number of imprecisions compared to actually using a higher polygon model. Since his method does not actually increase the geometric complexity of the model, this algorithm fails to affect the silhouette of the object. Furthermore, it neither manages to produce self shadowing objects (unless the unmodified geometric object already shadowed itself before), nor does it modify the (color-) texture coordinates used to texture the model, thus

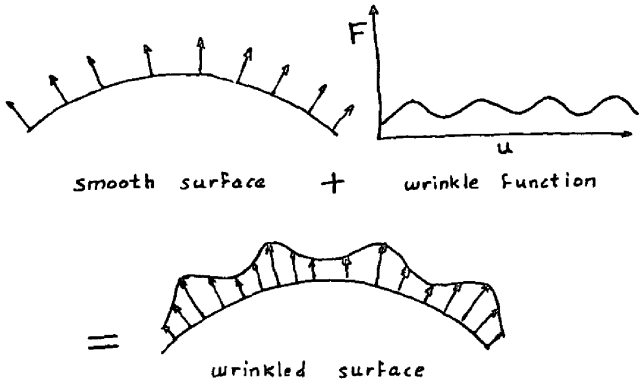

Figure 1: Basic principle of bump mapping, [Blinn 1978].

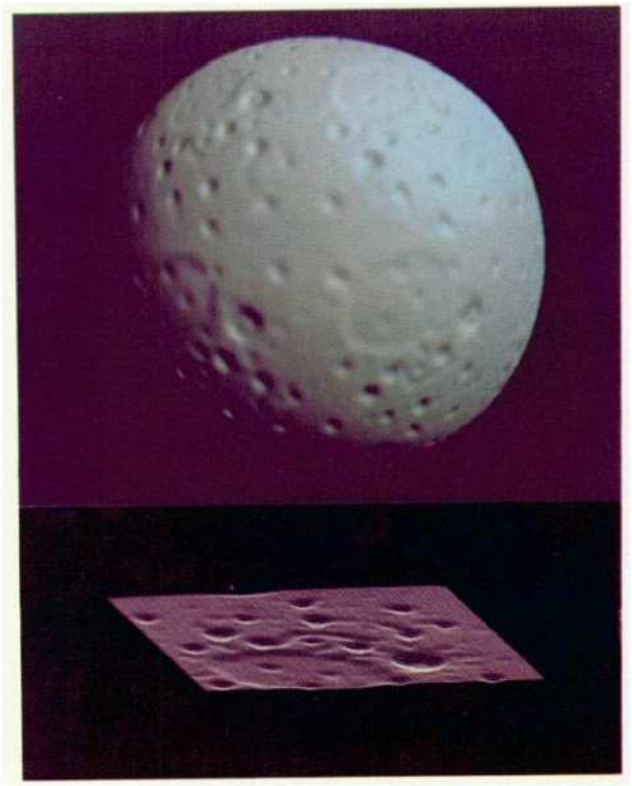

Figure 2: A sphere with bump mapping. (top) resulting image (bottom) image of the used wrinkle function. [Blinn 1978]

*e-mail: hadesbringer@netscape.net

failing to show self occlusion.

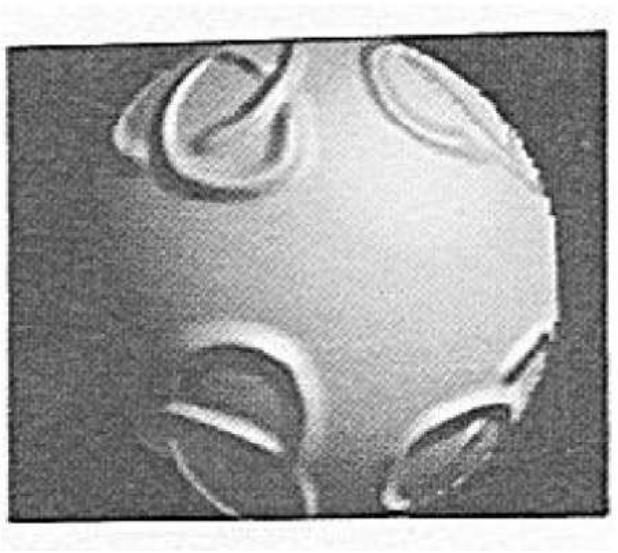

Figure 3: A sphere with bump mapping. Note that the silhouette does not show the presence of the bumps. [Blinn 1978]

Normal mapping, a variation of Blinn's original bump mapping algorithm, differs from the original by not simply perturbing the normal vectors of the surfaces, but completely replacing them with precalculated normal vectors usually stored in the red, green and blue components of a normal map. This results in a significant increase in rendering speed, but since the complete normal vector has to be stored in the texture, a texture used by this approach requires significantly more memory.

Another interesting variation of bump mapping is *horizon mapping* [Max 1986]. This approach adds the ability for bumps to cast shadows onto the object. This is accomplished by storing an additional horizon map for the surface. The horizon map contains the angle to the horizon (represents at what height the light can pass over the succeeding bumps in this direction) for a discrete number of directions (usually 8) for each texel. When the surface is being rendered, the angle to the light sources for the texel is compared to the angle required to pass over the horizon. If the angle is greater than the associated angle in the horizon map the respective pixel is illuminated, otherwise the light source is obscured by a bump and therefore cannot illuminate this position.

2.2 Displacement Mapping

Unlike bump mapping, *displacement mapping* [Cook 1984] actually changes the geometry of the object. Additional vertices are set onto the surfaces of the model and shifted to create the heights contained in the texture. This approach naturally allows self shadowing and self occlusion and also modifies the silhouette of the object. However, due to the fact that the object that is rendered is actually a lot more complex, this approach does suffer from poor performance. Thus, displacement mapping is not so much a way to simulate complex objects, but an approach to reduce the memory requirements to store a complex model.

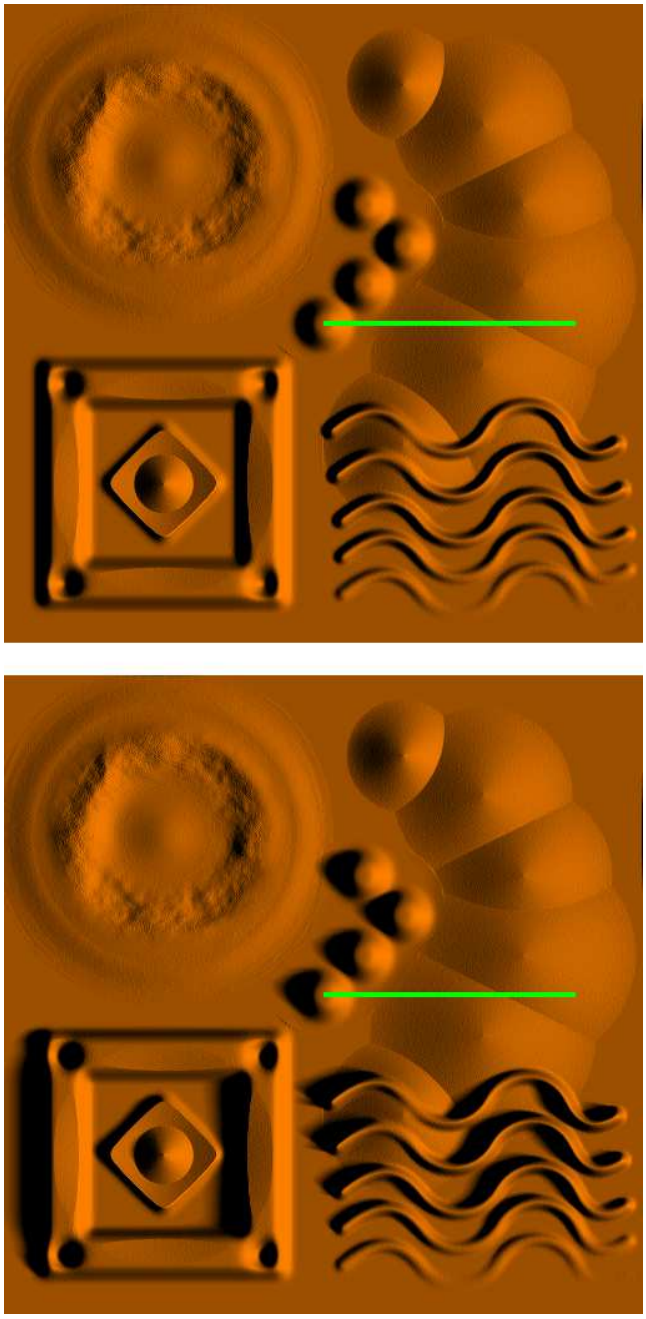

Figure 4: Rendered results, the green line shows the direction of the light (top) bump mapping (bottom) horizon mapping. Images taken from a demo program by Tom Nuydens.

2.3 Parallax Mapping

Parallax mapping [Kaneko et al. 2001] does not work by modifying the surface normals, but rather affects the way a texture is mapped to the surface. Thus, it is possible to use this approach in conjunction with bump mapping so the lighting is affected as well. From the regular texture position on the surface, the height displacement is calculated and transformed back onto the surface by multiplying it with the tangent of the angle between the texture axis and the visual axis. The texel at this new position is then used for texturing the pixel.

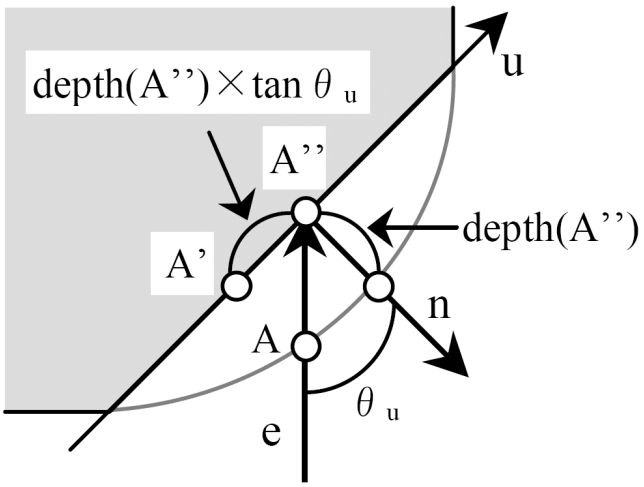

Figure 5: The basic principle of parallax mapping, [Kaneko et al. 2001]

This approach works well for smooth height functions on steep angles. While the approximation error can be considered marginally for such occasions, they can easily become significant if the angle becomes shallower or the height function gains higher frequencies.

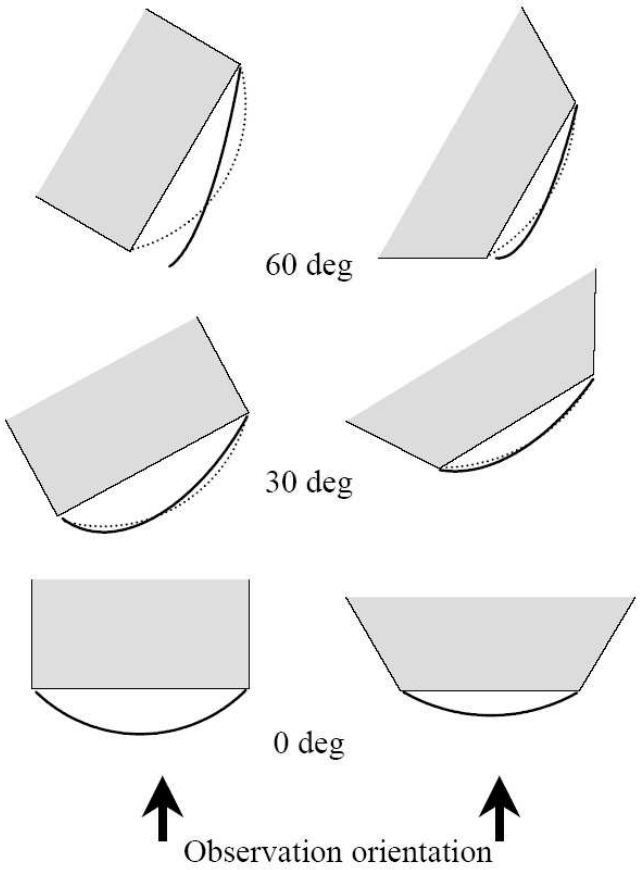

Figure 6: Accuracy of parallax mapping (dotted line) correct result (bold line) parallax texture mapping (surface of polygon) regular texture mapping, [Kaneko et al. 2001]

This technique does allow self occlusion but still lacks self shadows. While the perceived silhouette may be altered when using this approach (due to the fact that some new texture positions may be outside of the actual texture, and consequentially assumed to be completely transparent), the underlying geometric silhouette remains unaffected.

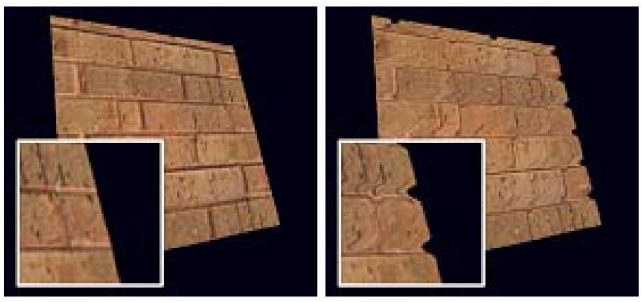

Figure 7: Rendered results (left) normal texture mapping (right) parallax texture mapping, [Kaneko et al. 2001]

2.4 Relief Mapping

Relief mapping [de Oliveira Neto 2000a], refined in [Policarpo et al. 2005] once again (see parallax mapping) warps portions of the texture map, but this time using a raycaster. The raycaster calculates the intersection point with the bumpy surface using a binary search between the original intersection point of the unmodified surface and the ray's intersection with the surface offset by the maximum difference in the heightfield preceded by a linear search in order to avoid finding the first intersection (might happen if a search point is outside the height field surface but has already intersected the surface before).

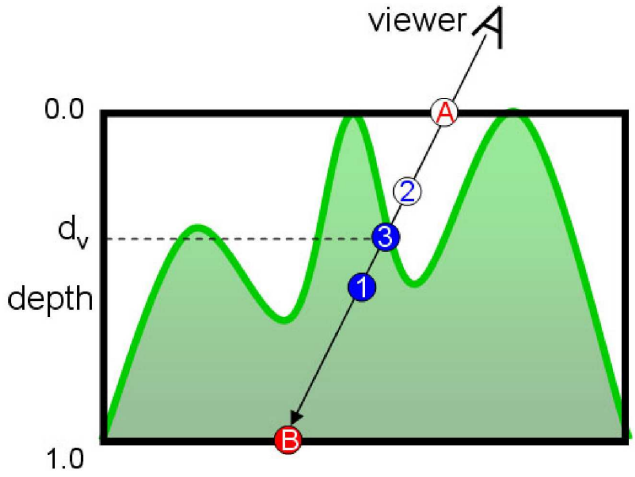

Figure 8: Binary search within a height field, [Policarpo et al. 2005]

This intersection point contains the coordinate of the texel to use as well as the depth information (from the observer). To add self shadowing to relief textures, another ray has to be cast towards each light source to check for occlusion with other bumps.

By properly refreshing the z-buffer with the modified z values, relief surfaces can even be correctly interpenetrated.

Figure 9: Rendered results, (top) regular textures (bottom) relief textures. [de Oliveira Neto 2000a]

Furthermore, it was shown in [Oliveira and Policarpo 2005] that by deforming the height field to match the object's geometry, it is possible to render silhouettes using this technique by deciding that a point does not belong to the model when the ray does not intersect the deformed height field.

Since the original shadow algorithm is no longer usable when using silhouettes, *shadow mapping* is the new shadow algorithm of choice.

Recently, relief mapping has been enhanced [Policarpo and Oliveira 2006] to support non-height-field structures of opaque, closed surfaces. Therefore, instead of having one height value in a height map, 2*n* height values (since closed surfaces always pass twice at any texel) are stored in one (or more) texture(s). This time, the height of the ray is not simply compared to the height in the height map, but it is checked if the ray is in a surface by checking if the height is between the two heights of any surface.

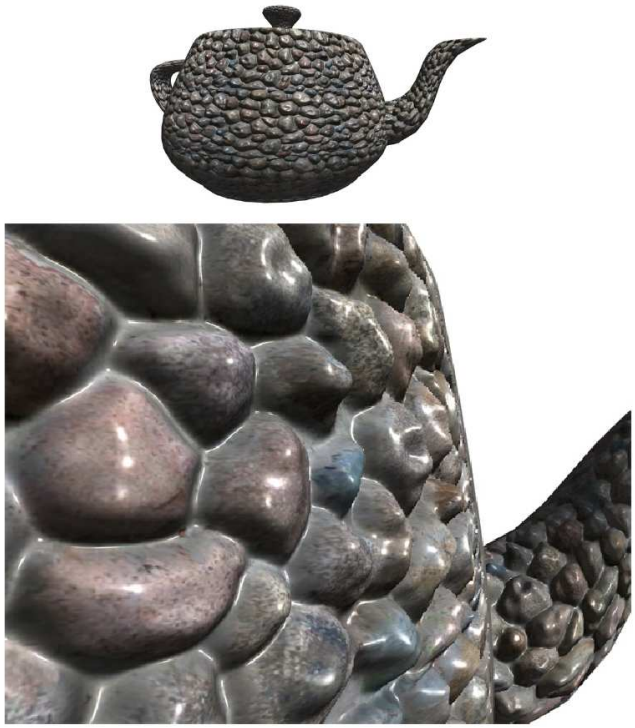

Figure 10: Rendered results, note the shadows. [Policarpo et al. 2005]

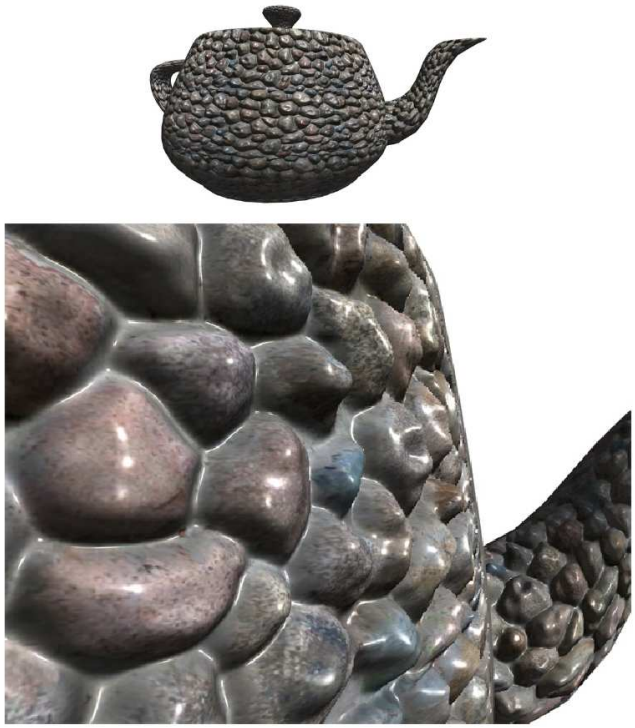

Figure 11: Rendered results, (top) relief mapped teapot (bottom) close-up view. [Policarpo et al. 2005]

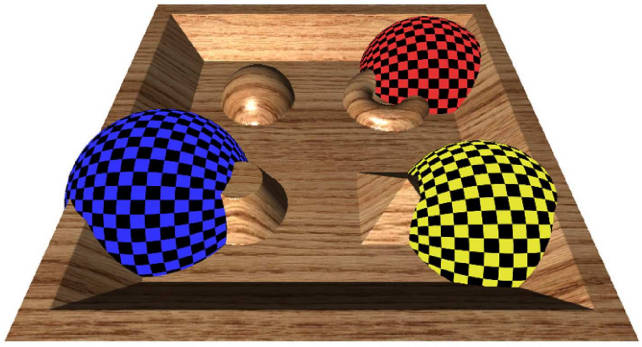

Figure 12: Interpenetration between a relief mapped surface and textured spheres, [Policarpo et al. 2005]

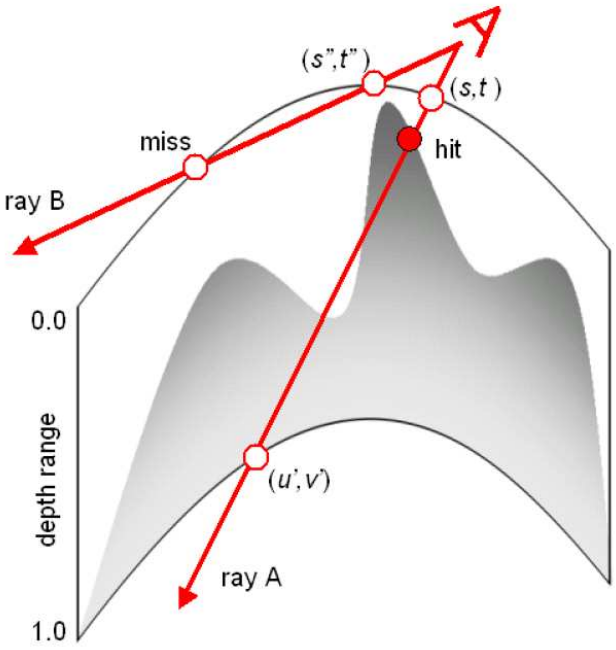

Figure 13: Rays traversing a deformed height field, [Oliveira and Policarpo 2005]

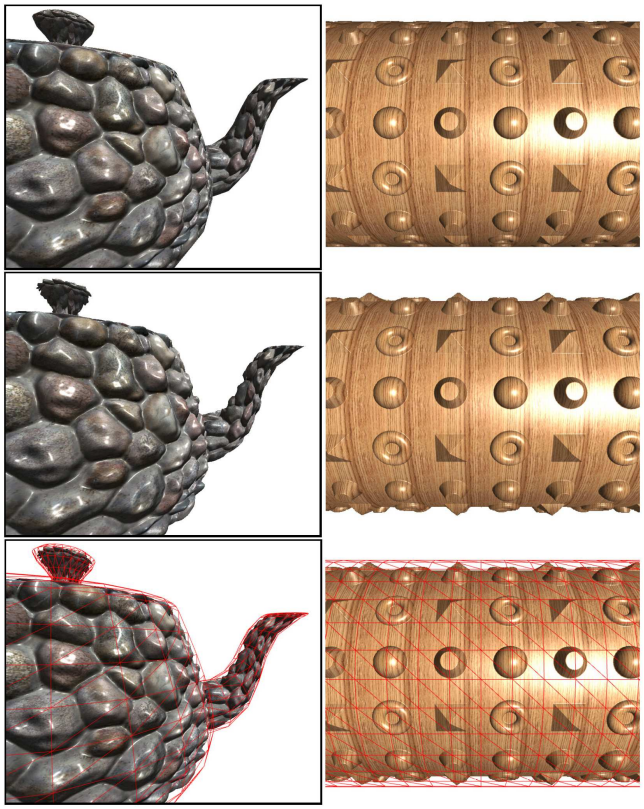

Figure 14: Rendered results, (top) original relief mapping (middle) relief mapping with silhouette (bottom) same as middle, but also showing a wireframe of the geometry. [Oliveira and Policarpo 2005]

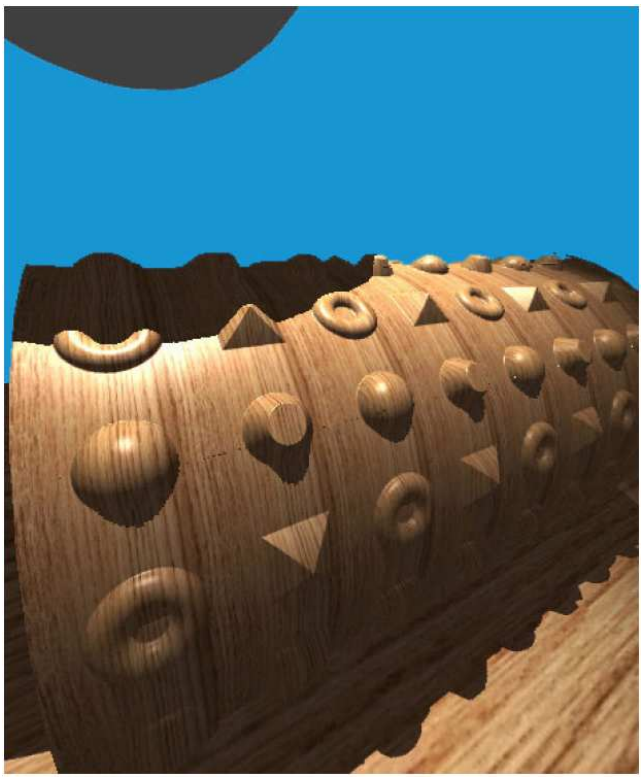

Figure 15: Rendered results, relief mapping showing silhouettes (shadowed using shadow mapping). [Oliveira and Policarpo 2005]

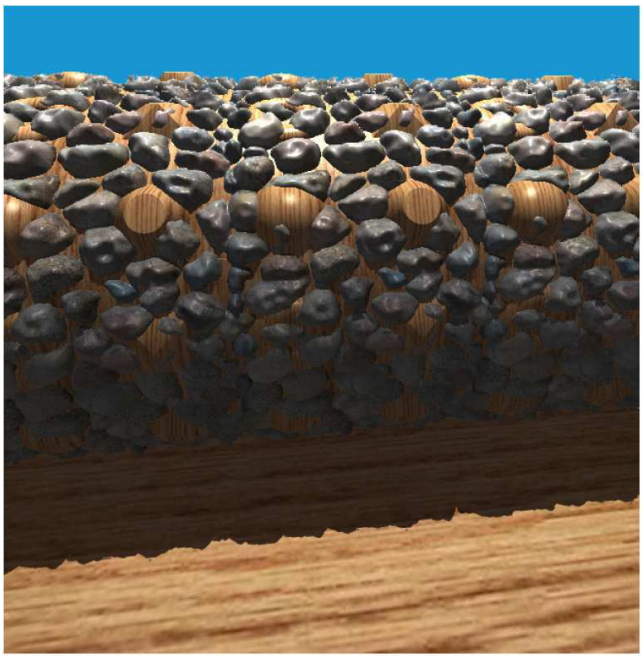

Figure 16: Rendered results, two coincident cylinders with different relief and texture maps. [Oliveira and Policarpo 2005]

Figure 17: Rendered results, (top) teapot with non-height-field weave pattern (bottom) close-up of a part of the teapot. [Policarpo and Oliveira 2006]

2.5 Shadow Mapping

Shadow mapping [Williams 1978], as the name suggests, is a way to shadow a scene. Unlike other shadowing algorithms, this algorithm does its work completely in the image space. This leads to the fact that it requires no knowledge of the actual geometry that casts the shadows, but, since it uses discrete sampling points to compute the shadows, it has to deal with aliasing artifacts.

With shadow mapping, the scene is rendered once normally without lighting as well as once for every light source that can cast shadows onto a surface of the visible part of the scene. However, since this algorithm only requires the depth information of the light sources, no color information of the scene has to be calculated for this pass, and only the z-buffer has to be filled. The shadowing itself is done by transforming the (x, y, z) positions of the computed scene (from the observer's point of view) to the coordinate system of the light views and subsequently checked for visibility from the light source.

If the rendered position is further away from the light source, this light does not affect the intensity of the point, and it has to be properly shaded. Otherwise, it equals the corresponding value of the light sources depth map.

However, in computer graphics, due to the discrete nature of the depth buffer (as well as numeric inaccuracies of machine arithmetic), the projected z-value will, even if it refers to the exact same (x, y, z) position, fall either above or below the z-value of the corresponding value of the light sources depth map.

Therefore, Williams introduced the idea of subtracting a (usually constant) bias from the z-value of the transformed point to avoid self-shadow artifacts. While this may alter the silhouette of the shadow slightly, it is preferable to an implementation without bias, since the odds that a fragment may shadow itself (if no bias is used) is quite high. On the other hand, the bias should not be too large since it effectively increases the minimum distance an object may have to a surface to be able to cast a shadow onto the surface.

This still leaves the aliasing problem, which is caused by one shadow map texel mapping to more than one pixel of the final scene. The sources of this undersampling with shadow maps have been well formalized [Stamminger and Drettakis 2002].

$$d = d_s \frac{r_s}{r_i} \frac{cos\beta}{cos\alpha}$$

The aliasing problem can be further split up in perspective aliasing $d_s r_s / r_i$, which typically happens when the user zooms the view to the edge of a shadow, and projection aliasing $cos\beta / cos\alpha$, which occurs when the light rays are close to parallel to the surface.

A number of ways to reduce the effect of aliasing using shadow maps have been proposed. *Percentage closer filtering* [Reeves et al. 1987] does so by comparing the result of the transformation to the surrounding depth values as well as the central one and calculating the percentage of shadowed values. The point is then considered to be in shadow by this percentage and consequently shaded accordingly.

One particularly nice feature is that it not only reduces the artifacts, but also allows partially shaded fragments. Thus this algorithm can not only reduce aliasing artifacts, but also supports a smooth transition from shadowed to illuminated areas.

A more sophisticated method of reducing aliasing are *perspective shadow maps* [Stamminger and Drettakis 2002]. Unlike uniform shadow maps, perspective shadow maps are computed in device coordinates rather than world coordinates like their uniform counterpart. This approach aims at reducing the influence of the fraction

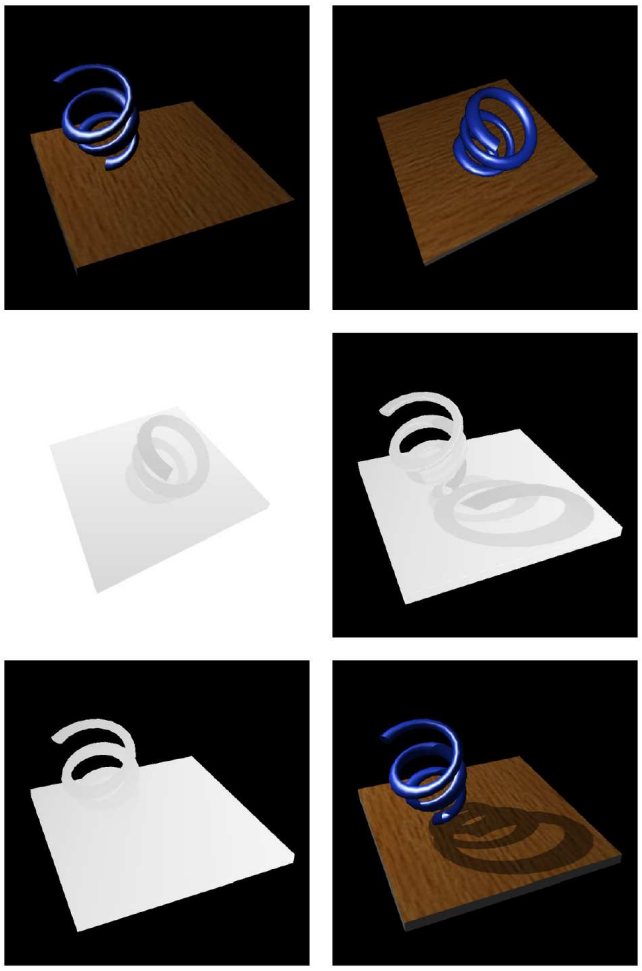

Figure 18: The shadow mapping algorithm: (top left) unshadowed scene (top right) unshadowed scene from the position of the light source (middle left) depth map of light source scene (middle right) depth map of light source transformed to observer's view (bottom left) light's planar distance transformed to observer's view (bottom right) shadowed scene. [Nealen 2002], taken from an NVIDIA demo.

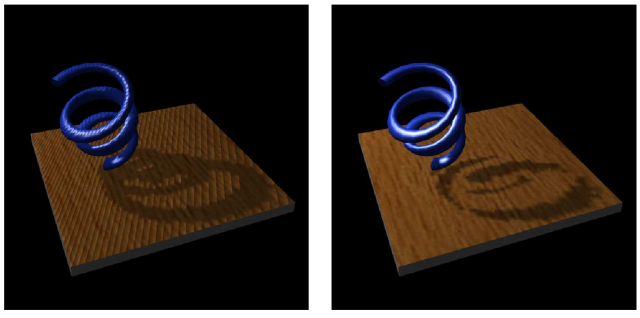

Figure 19: Effect of bias (left) bias too low, resulting in self-shadowing (right) bias too high, resulting in incomplete shadowing (only the highest part casts a shadow in this case). [Nealen 2002], taken from an NVIDIA demo.

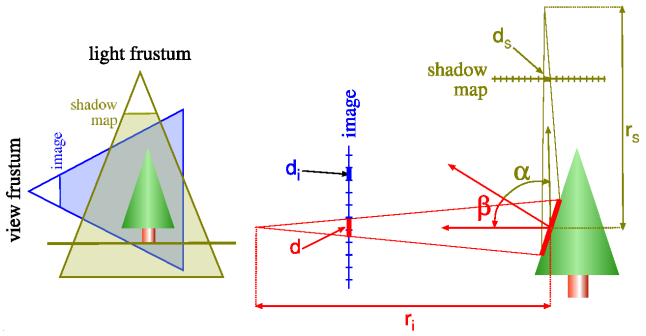

Figure 20: Shadow map projection qualities, [Stamminger and Drettakis 2002].

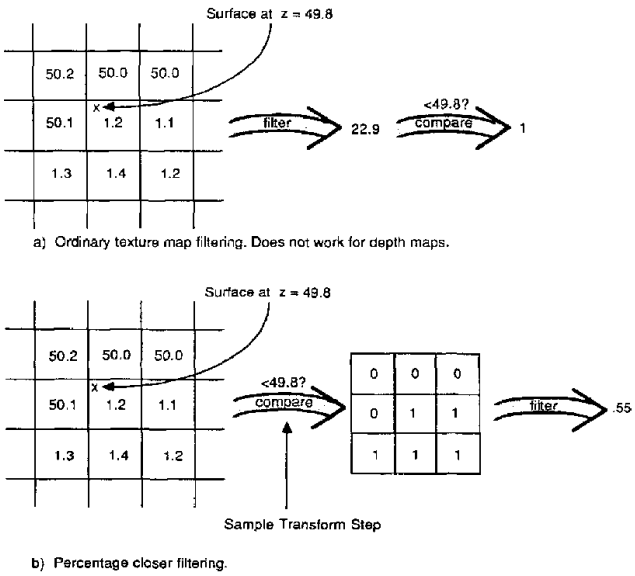

Figure 21: Principle of percentage closer filtering, [Reeves et al. 1987]

r_s/r_i since the projections empowering this fraction are removed after said transformation. However, special attention has to be taken with objects behind the camera plane.

3 Conclusion

While texture mapping can noticably increase the perceived quality of an image, more elaborate algorithms can still add greatly to the perceived realism of a scene without even modifying the geometry. While state-of-the-art algorithms can provide great visual results, they do not provide interactive framerates in large environments with current hardware.

References

BLINN, J. F. 1978. Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, 286 – 292.

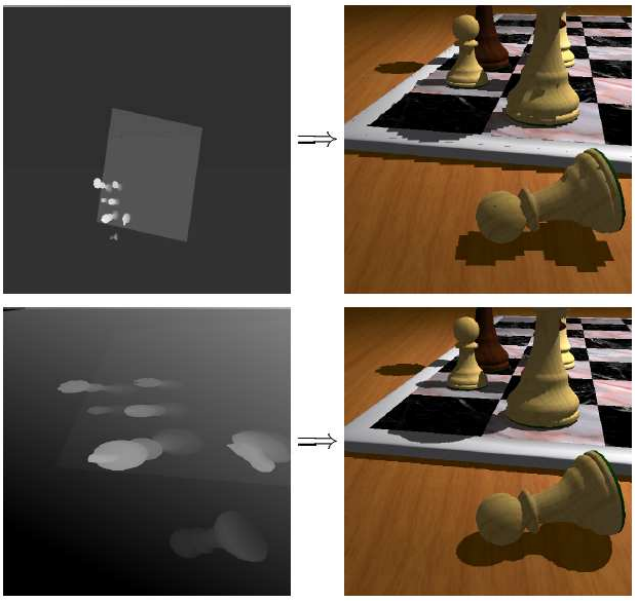

Figure 22: Shadow quality improvement from using perspective shadow maps instead of uniform shadow maps. (upper) uniform 512 x 512 shadow map (lower) perspective 512 x 512 shadow map. [Stamminger and Drettakis 2002]

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. 2002. Shadow mapping for hemispherical and omnidirectional light sources. In *Computer Graphics International (CGI) 2002 proceedings*, 397 – 408.

CATMULL, E. E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah.

CIGNONI, P., MONTANIY, C., ROCCHINIZ, C., AND SCOPIGNOX, R. 1998. A general method for preserving attribute values on simplified meshes. In *Proceedings of the conference on Visualization '98*, 59 – 66.

COHEN, J., OLANO, M., AND MANOCHA, D. 1998. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 115 – 122.

COOK, R. L. 1984. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 223 – 231.

DE OLIVEIRA NETO, M. M. 2000. Relief texture mapping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 359 – 368.

DE OLIVEIRA NETO, M. M. 2000. *Relief Texture Mapping*. PhD thesis, University of North Carolina at Chapel Hill.

KANEKO, T., TAKAHEI, T., INAMI, M., KAWAKAMI, N., YANAGIDA, Y., MAEDA, T., AND TACHI, S. 2001. Detailed shape representation with parallax mapping. In *Proceedings of the ICAT 2001*, 205 – 208.

MASCHEK, M. 2006. A state of the art report for displacement mapping. Tech. rep., Vienna University of Technology.

MAX, N. L. 1986. Shadows for bump-mapped surfaces. In *Proceedings of Computer Graphics Tokyo '86 on Advanced Computer Graphics*, 145 – 156.

MCMILLAN, L. 1997. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina.

NEALEN, A. V. 2002. Shadow mapping and shadow volumes: Recent developments in real-time shadow rendering. Tech. rep., University of British Columbia.

OLIVEIRA, M. M., AND POLICARPO, F. 2005. An ecient representation for surface details. Tech. rep., UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL.

PEERCY, M., AIREY, J., AND CABRAL, B. 1997. Efficient bump mapping hardware. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 303 – 306.

POLICARPO, F., AND OLIVEIRA, M. M. 2006. Relief mapping of non-height-field surface details. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, 55 – 62.

POLICARPO, F., OLIVEIRA, M. M., AND COMBA, J. L. D. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 155 – 162.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 283 – 291.

SLOAN, P.-P. J., AND COHEN, M. F. 2000. Interactive horizon mapping. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, 281 – 286.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 557 – 562.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, 270 – 274.