Real-time Rendering of Translucent Volumetric Surfaces

Ville Timonen*
Åbo Akademi University

Figure 1: *Dyed glass on a height mapped surface*

Abstract

We present a real-time technique for rendering translucent features on opaque volumetric surfaces. It relies on modern reprogrammable graphics hardware to compute the visual phenomena seen in translucent materials. The proposed technique is based on relief mapping techniques, and is therefore capable of interacting with the underlying opaque surface accurately. A hard self-shadowing technique used for shadowing opaque relief mapped surfaces is extended to suit the simulation of translucent features. Also, light absorption and planar local reflection methods are suggested as improvements to the base technique. The proposed rendering technique proves feasible for real-time applications, and is capable of simulating small-scale translucent features, such as drops of liquid, on opaque surfaces in a very realistic manner.

Keywords: Real-time graphics, volumetric surfaces, relief mapping, translucency

1 Introduction

Real-life objects and surfaces often consist of translucent features of some sort. For instance, wrist watches have glass covers, opaque surfaces can be coated with drops of liquid, various ornaments are composed of transparent materials such as diamond, and layered car paints have transparent coatings. Therefore methods for simulating translucency are useful for the production of realistic graphics.

Translucency simulation is challenging due to the complicated visual phenomena seen in translucent materials. These phenomena are mainly the result of light's unique behaviour within, and especially on the boundary of, translucent mediums that have indices of refraction other than that of the environment's. Translucent features also intensively interact with the environment, making real-time implementations particularly tricky. Because of these properties, physically accurate simulation models for translucent mediums have traditionally been seen exclusively in pre-rendered graphics.

In real-time graphics rendering, the trend on increasing detail has lately been towards more complicated fragment-level computation, made possible by the improved processing power and reprogrammability of modern GPUs. Texture and bump mapping [Weinhaus and Devarajan 1997] have been succeeded by normal and offset mapping techniques [Heidrich and Seidel 1999] [Welsh 2004], which simulate small-scale details on opaque surfaces with an increased accuracy.

Using reprogrammable hardware, it is possible to incorporate iterative algorithms, usually seen in ray tracing techniques, in fragment-level real-time rendering of volumetric surface. Relief mapping techniques utilize this possibility, and have been under research [Policarpo et al. 2005] [Tatarchuk 2006] [Donnelly 2005] recently. One significant advantage of these techniques in simulating surface geometry is that they have little overhead; they can simulate per-pixel geometry very efficiently. Considering the complexity of translucency rendering, the above characteristics make relief mapping based techniques research-worthy for simulating local translucent features on volumetric surfaces.

The contribution of this paper is to demonstrate the feasibility of relief-mapped surfaces in high-detail local translucency rendering. This is done by presenting a real-time algorithm to fit the said purpose, which utilizes and combines known relief-mapping techniques and the well-studied relating optics.

The hard self-shadowing technique for relief-mapped surfaces [Policarpo et al. 2005] is extended to take translucent features into account. A method for simulating simple absorption of light within the translucent material is proposed. Also, two practical enhancements to the base technique are suggested: planar local reflections and Phong-specular intensification.

2 Related work

Relief mapping
Opaque volumetric surfaces can be efficiently rendered in real-time using *relief mapping* techniques. These techniques utilize a *height map* representing local height of the surface, effectively defining its geometry. The viewing ray is then traced to the surface, until an intersection point with the height map is found with an adequate accuracy. This intersection point is used to sample color and normal maps in order to compose the appearance of the rendered frag-

*e-mail: vtimonen@abo.fi

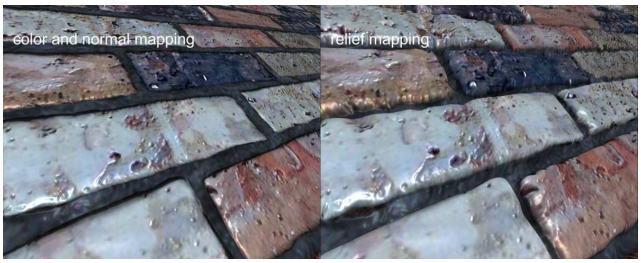

Figure 2: *A color and normal mapped surface without (left) and with (right) relief mapping*

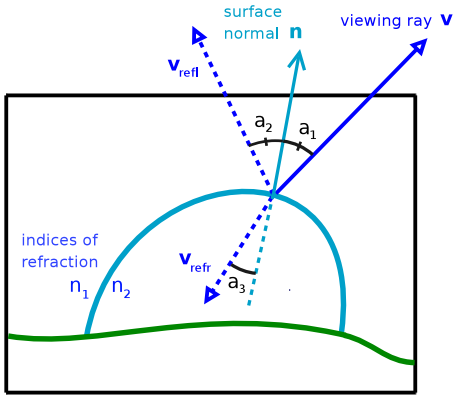

Figure 3: *Light behaviour during refraction and reflection*

ment. As a result, surface geometry is simulated with great detail, as demonstrated in Figure 2.

A viewing ray can be traced using several methods that suit different surface types and different stages of the intersection search. In the context of this paper, the interesting methods are *linear search* [Policarpo et al. 2005], *binary search* [Policarpo et al. 2005], and *linear height field approximation* [Tatarchuk 2006]. Self-shadowing of opaque volumetric surfaces can be implemented as proposed in [Policarpo et al. 2005], which produces hard[1] self-shadows, or as proposed in [Tatarchuk 2006], which produces soft[2] self-shadows.

Optics

The appearance of most translucent features is determined by the way light behaves on the boundary of the translucent material. When light hits the boundary of translucent material with an *index of refraction* other than that of the environment's, portion of the light intensity is reflected and the remaining refracted, as demonstrated in Figure 3.

Intensities, in which the light splits, are given by the *Fresnel terms* [Blinn 1977]:

$$I_{refl} = \frac{\left(\frac{n_1 cos(a_1) - n_2 cos(a_3)}{n_1 cos(a_1) + n_2 cos(a_3)}\right)^2 + \left(\frac{n_1 cos(a_3) - n_2 cos(a_1)}{n_1 cos(a_3) + n_2 cos(a_1)}\right)^2}{2}$$

$$I_{refr} = 1 - I_{refl}$$

[1] Shadows as cast by a light source that is either infinitely small or infinitely far away.

[2] Smooth shadows that react properly to light source distance and dimensions.

The directions of reflected and refracted viewing rays in vector form are given by equations [Shirley 2002]:

$$\mathbf{v_{refl}} = \mathbf{v} + 2cos(a_1)\mathbf{n}$$

$$\mathbf{v_{refr}} = \left(\frac{n_1}{n_2}\right)\mathbf{v} + \left(cos(a_3) - \frac{n_1}{n_2}cos(a_1)\right)\mathbf{n}$$

Translucent materials can also absorb light. If that is the case, one option to simulate its effect on light intensity is to apply a coefficient as a function of distance d:

$$I = 1 - \left(\frac{1}{2}\right)^{d/d_0} \quad (1)$$

Hued materials absorb light of certain wavelengths, and the coefficient 1 needs to be computed as a function of wavelength. For some absorbing mediums such as water, this is just an approximation; the actual process of light absorption might be more complicated.

Environment mapping

The rendering technique presented in this paper relies on environment mapping. An environment can be simulated using *perspective dependent maps* [Vlachos et al. 2002], if it is being mapped in a relatively undistorted manner, for example as a reflection on water that is either still or only slightly rippled. A more general-purpose environment mapping technique is *cube mapping* [Kilgard 1999], which uses six textures to represent the surrounding environment as a cube.

Translucency rendering

There are several — mostly polygon-based[3] — rendering techniques for simulating specific translucent materials. They are not usually general-purpose, but rather specialize in approximating effects that dominate e.g. on large water surfaces [Premoze and Ashikhmin 2001] [Belyalev 2003] [Yung-Feng and Chun-Fa 2006] or in ice [Seipel and Nivfors 2006].

A polygon-based rendering technique proposed in [Chan and Wang 2005] is capable of simulating fully transparent and translucent objects of nearly arbitrary shape. It simulates reflection and refraction in a realistic fashion, and implements light scattering as well. As a polygon-based technique, it is only able to simulate fully translucent objects. Refractions and reflections are sampled from the environment alone, making the translucent features incapable of interacting with opaque features within the vicinity. Also, the technique relies on precomputed geometry data, and as such does not suit dynamic or animated objects.

During the development of the rendering technique presented in this paper, an implementation [Baboud and Décoret 2006] was published, which simulates water volumes partly the same way as the independently developed technique presented here. However, it incorporates techniques which are better suited for simulating deep water volumes, rather than small-scale translucent details on opaque surfaces — the intended application of the technique proposed in this paper.

3 Translucency rendering using height maps

The proposed simulation model uses layered relief maps to describe the surface geometry. The purpose is to simulate a layer of translucent medium on a solid surface. Two-layer relief maps are used to suit the purpose: one describing the solid layer and one defining

[3] No sub-polygon geometry is introduced via e.g. a height map

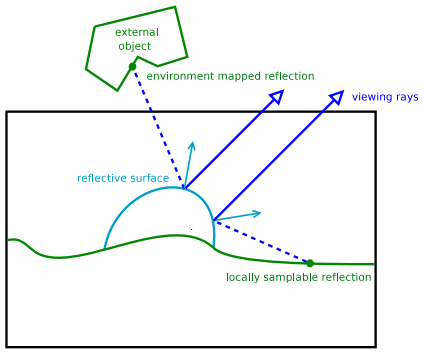

Figure 4: *Local and environment sampling of reflections*

the surface of the translucent material between the layers. Normal maps are used for both layers.

If the layers are close to each other and have height variations of similar magnitude, the same bounding box can be used for both layers. In this example implementation, separate bounding boxes are used. Floating point height maps can be utilized as well, if height variations are large enough to require additional resolution or depth. Algorithm 1 lists a pseudo-algorithm for the fragment program.

Various different relief mapping algorithms can be used to search for the viewing ray intersection points. In this example implementation, linear search is used to close in on the intersection range and binary search is used to refine the range. Linear height field approximation is used to compute the final intersection coordinates.

When the viewing ray hits the solid layer first, ordinary surface rendering procedure is used. When the translucent layer is encountered first, portion of light is reflected and the remainder is refracted. As an optimization only the first encounter with the translucent layer is taken into account to avoid the rapid increase of rays to be traced. This means that double reflections or reflections on multiple layers are not simulated.

The resulting rays can be further traced into the surface itself or into the environment, as can be seen in Figure 4. The environment can be simulated using a cube map [Kilgard 1999] if it is relatively distant to the simulated surface.

Algorithm 1 Composing a fragment

$intersection \leftarrow$ TRACE($viewing\_ray$)
if $intersection$ is on solid layer **then**
  return SAMPLE($intersection$)
else
  $I_{reflected} \leftarrow$ REFLECTIONINTENSITY($intersection$)
  $I_{refracted} \leftarrow 1 - I_{reflected}$
  $v_{reflected}, v_{refracted} \leftarrow$ VIEWINGRAYS($intersection$)
  $refraction \leftarrow I_{refracted} \cdot$ SAMPLE(TRACE($v_{refracted}$))
  $intersection \leftarrow$ TRACE($v_{reflected}$)
  if $intersection$ escapes the surface **then**
    $reflection \leftarrow I_{reflected} \cdot$ ENVIRONMENT($v_{reflected}$)
  else
    $reflection \leftarrow I_{reflected} \cdot$ SAMPLE($intersection$)
  end if
  return $refraction + reflection$
end if

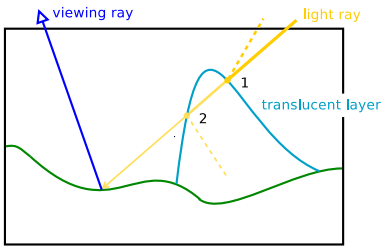

Figure 5: *Light intensity decreases as it passes through a translucent layer*

3.1 Light occlusion

In order to produce hard shadows cast by the translucent medium, it is possible to trace light rays through the upper layer, as demonstrated in Figure 5. Light intensity needs to be calculated according to the Fresnel terms every time a surface boundary is passed. The remaining refraction intensity can be used as an approximate[4] light intensity reaching the fragment that is being rendered.

This type of light occlusion simulation can be thought as an extended hard self-shadowing presented in [Policarpo et al. 2005]; therefore similar properties apply. Instead of finding a binary value for whether a fragment is lit or not, an occlusion coefficient is computed. Algorithm 2 lists a pseudo-algorithm for determining the coefficient for a fragment.

Algorithm 2 Computing a light occlusion coefficient

$light \leftarrow 1$
loop
  $intersection \leftarrow$ TRACENEXTBACKWARDS($light\_ray$)
  if no $intersection$ **then**
    return $light$
  else if $intersection$ is on solid layer **then**
    return 0
  else
    $light \leftarrow light \cdot$ REFRACTIONINTENSITY($intersection$)
  end if
end loop

3.2 Practical improvements

Light absorption
Real-life translucent mediums — even pure water — often absorb light of certain wavelengths making the medium and its shadows hued. When light rays are traced as proposed in this paper, distances traveled within the translucent medium are trivial to compute. That distance information can be utilized in constructing a color vector used for dimming the medium in a physically approximate fashion.

Planar local reflections
Typical relief mapping algorithms can be used to trace the refracted and reflected viewing rays into the surface. If the surface is reasonably flat, ray intersections can be computed against a flat plane representing the solid layer of the surface. This solution increases performance at the expense of image quality. Especially reflections are often indistinguishable enough to justify this choice. In addition, reflected viewing rays tend to intersect the surface in steep

[4] Incident light also undergoes refraction, which changes its direction. This change in direction, along with reflected light, is left unsimulated.

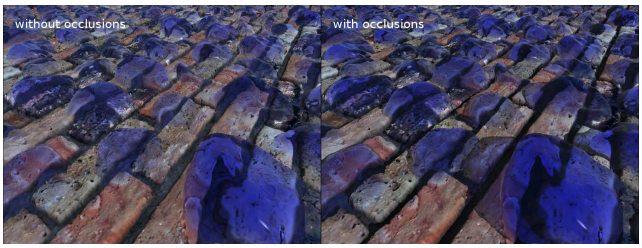

Figure 6: *Blue hued translucent medium with and without occlusion of light*

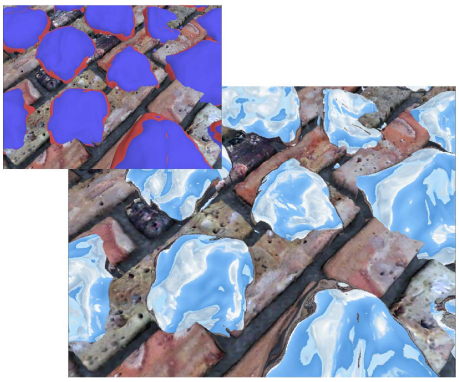

Figure 7: *Reflections that can be locally sampled are marked in red and environment reflections in blue. A cloudy sky [Partanen 2005] is used as the environment map. Surface textures are from [Cloward 2006].*

angles where relief mapping algorithms are unoptimal[5].

Phong-specular intensification
If the proposed rendering technique is applied in environments where the *Phong lighting model* [Blinn 1977] is applied for specular reflections, light source reflections on the translucent layer need to be intensified to make them appear visible after they have lost intensity during reflection calculations. If no such correction is applied, light source reflections lose a significant portion of their initial peak intensity of 1 (the maximum color value), ending up with unrealistically low intensities.

4 Results

The suggested rendering technique produces visually plausible results. This is to be expected, as the rendering technique does not resort to extensive faking or approximating the optical behaviour of the surface. The translucent medium simulated in Figure 6 absorbs wavelengths other than that of blue, and also features light occlusion. Planar local reflections can be successfully applied on reasonably flat surfaces. This type of reflections is used in Figure 7, where the simplification gives plausible results.

In Phong-lighted environments, specular light reflections on the translucent layer should be intensified. Figure 8 demonstrates this type of light source reflection intensification.

When the environment is light, realistic reflections require that the environment maps are *high dynamic range* (HDR)

[5] Steep viewing angles require larger numbers of intersection search iterations to guarantee that intersection skipping and inaccuracy are kept to a reasonable level.

Figure 8: *Intensified specular reflections*

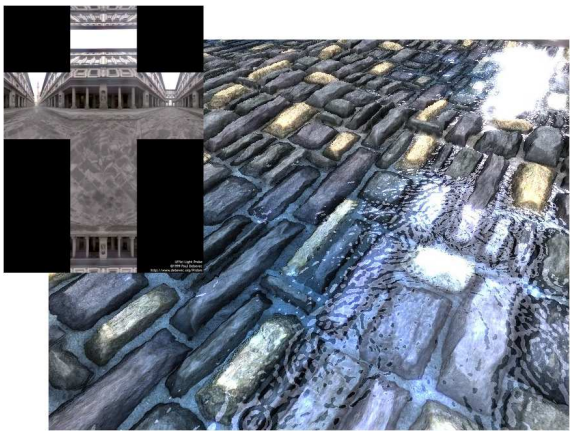

Figure 9: *A water layer reflecting a HDR environment map [pau 1998]*

[Green and Cebenoyan 2004]. By using a HDR environment, reflections are not clamped to unrealistically low values due to color value limitations of traditional low dynamic range (LDR) textures. Figure 9 illustrates a water layer reflecting a HDR environment map [pau 1998]. *Light blooming* is implemented by separating HDR values from the frame buffer object using a square root based value controlling, and interpolatively down sampling and Gaussian blurring the HDR map, before additively blending with the LDR values.

4.1 Applications

The technique proposed in this paper is designed to suit the simulation of small-scale translucent features on volumetric surfaces. The index of refraction of the translucent medium can be freely chosen to represent any material, such as water, glass, or diamond.

Simulable surface types include:

- Drops of water or colored liquids on surfaces
- Sweaty or bloody skin
- Surfaces with full or partial glass coating
- Objects with embedded translucent parts, e.g. diamonds
- Even deep water layers, as illustrated by Figure 10

The rendering technique can be utilized in computer games, virtual worlds, etc. — anywhere where realistic simulation is preferred.

The light occlusion method is applicable on surfaces that need self-shadowing. It is unable to simulate *caustics*, and is therefore only suitable for small-scale translucent features if physical reality is required. For example, water volumes of significant depth require e.g. *photon mapping* based techniques to appear completely realistic.

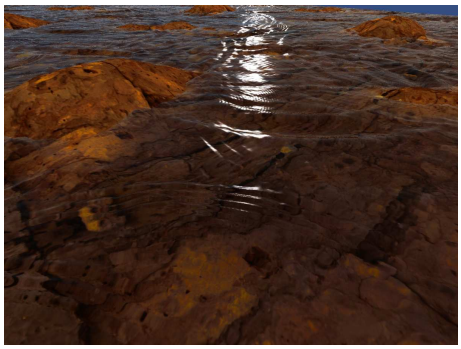

Figure 10: *A layer of clear water. The ground texture is from [Cloward 2006]*

Table 1: *Performance measurements*

Rendering technique	Performance [fps]
Normal and color mapping	229
Relief mapping	33.0
Two-layer translucency rendering	13.8
Occlusion simulation	4.43
Hued light absorption	3.79

4.2 Performance

Although performance depends on several implementation and hardware specific factors, it is possible to estimate the computational weight of the proposed technique by comparing rendering times to popular techniques, e.g. to single layer relief mapping. Rendering in this technique is done entirely in the fragment shader, making the performance linearly dependent on screen coverage and view port resolution. Table 1 lists some of the benchmark results. The rendering techniques are applied cumulatively. The benchmarked techniques are not optimized.

The shader program [6] is written in *OpenGL Shading Language* (GLSL), and NVIDIA's GeForce 7600GS (G73 architecture) graphics card is used for the execution [7] of the shader at a resolution of 800x600 using full screen coverage.

The base technique achieves interactive frame rates on commodity hardware. The light occlusion simulation approaches the performance of hard self-shadowing [Policarpo et al. 2005] when light rays seldom hit the translucent layer. The performance drops quickly, however, if the translucent layer shadows significant portions of the surface, as intersection calculations with the translucent layer are computationally intensive.

5 Summary

We have proposed a technique to simulate volumetric translucent features on solid surfaces using layered relief-maps. Reflections, refractions, and light absorption are simulated in a physically authentic manner. Interactive frame rates are achieved, and the technique is best suited to simulate small-scale translucent details, such as drops of liquid, on a detailed volumetric surface.

[6] The program can be obtained from
`http://wili.cc/research/translucency/`

[7] The application environment is a GNU/Linux operating system with the NVIDIA's OpenGL 2.0 implementation using Linux x86 driver version 1.0-8774.

The suggested method produces results that are not achieved through polygon-based techniques in real-time. The translucent layer is able to interact with the accompanied solid layer without having to resort to crude approximations, and the technique does not rely on precomputation, so it is suitable for dynamic and animated surfaces. Other advantages of fragment-level geometry apply: dynamic per-fragment level of detail, and little overhead in geometry-dependent operations.

The suggested light occlusion technique is an extension to the hard self-shadowing method used for solid relief-mapped surfaces. According to the benchmarks, it is computation-hungry on surfaces where translucent shadows dominate, but produces plausible results and can be applied on high-end hardware.

6 Future work

Light scattering is not simulated in our model. The method proposed in [Chan and Wang 2005] simulates light scattering in translucent mediums and produces plausible results. The method can be applied on the technique proposed in this paper, as distances traveled within the translucent medium can be trivially computed. The proper way of blurring and sampling the solid layer in our technique needs further research, as does the question, of whether it can be done in real-time to preserve our technique's ability to render animated surfaces.

Interaction with polygonal objects within the vicinity of the translucent layer also needs new techniques. For example, objects partially submerged in the translucent medium and object reflections cannot be simulated without approximations using the proposed technique as it is. Environment mapping techniques for this purpose need further research. It might also be practical to render external objects into the color, normal, and height maps of the surface itself.

It is possible to use high dynamic range environment maps for lighting the surface using photon mapping based algorithms. Photons could be shot from the environment map according to its color values. Real-time suitability of such techniques needs further research.

The technique proposed in this paper can be extended into simulating gases and air of varying temperature within the vicinity of the surface. An animable 3D texture consisting of samples of the medium's index of refraction can be used as a basis for tracing the viewing ray's distorted path to the surface, or simulating its bounce back into the environment. The latter effect can be seen on roads during a hot day (*mirage*). The way refraction bends the viewing ray would then have to be calculated for each sampling step of the simulated medium.

References

BABOUD, L., AND DÉCORET, X. 2006. Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena*, Eurographics.

BELYALEV, V. 2003. Real-time simulation of water surface. In *GraphiCon-2003, Conference Proceedings*, MAX Press, 131–138.

BLINN, J. 1977. Models of light reflection for computer synthesized pictures. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 192–198.

CHAN, B., AND WANG, W. 2005. Geocube – gpu accelerated real-time rendering of transparency and translucency. *Visual Computer 21*, 8-10, 579–590.

CLOWARD, B., 2006. Texture resources. Web-page, referenced 12.10.2006. http://www.bencloward.com/resources_textures.shtml.

DONNELLY, W. 2005. *Per-Pixel Displacement Mapping with Distance Functions*. Addison Wesley Professional, 123–136.

GREEN, S., AND CEBENOYAN, C. 2004. High dynamic range rendering on the geforce 6800. Tech. rep.

HEIDRICH, W., AND SEIDEL, H. 1999. Realistic, hardware-accelerated shading and lighting. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 171–178.

KILGARD, M. 1999. Perfect reflections and specular lighting effects with cube environment mapping. Tech. rep.

PARTANEN, S., 2005. A camera picture. Web-page, referenced 12.10.2006. http://www.partanen.net/tmp/taustoja/Norja_taustakuva.jpg.

1998. Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics With Global Illumination and High Dynamic Range Photography.

POLICARPO, F., OLIVEIRA, M., AND COMBA, J. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 155–162.

PREMOZE, S., AND ASHIKHMIN, M. 2001. Rendering natural waters. *Computer graphics forum 20*, 4, 189–200.

SEIPEL, S., AND NIVFORS, A. 2006. Efficient rendering of multiple refractions and reflections in natural objects. *SIGRAD 2006 19*, 1.

SHIRLEY, P. 2002. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., Natick, MA, USA.

TATARCHUK, N. 2006. Dynamic parallax occlusion mapping with approximate soft shadows. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 63–69.

VLACHOS, A., ISIDORO, J., AND OAT, C. 2002. *Rippling reflective and refractive water*. Addison-Wesley.

WEINHAUS, F., AND DEVARAJAN, V. 1997. Texture mapping 3d models of real-world scenes. *ACM Comput. Surv. 29*, 4, 325–365.

WELSH, T. 2004. Parallax mapping with offset limiting: A per-pixel approximation of uneven surfaces. Tech. rep.

YUNG-FENG, C., AND CHUN-FA, C. 2006. Gpu-based ocean rendering. Tech. rep.