Real-Time Illumination of Foliage Using Depth Maps

Jesus Gumbau, Miguel Chover, Cristina Rebollo, and Inmaculada Remolar

Dept. Lenguajes y Sistemas Informaticos, Universitat Jaume I, Castellon, Spain
{jgumbau,chover,rebollo,remolar}@uji.es

Abstract. This article presents a new method for foliage illumination which takes into account direct, indirect illumination and self-shadowing. Both indirect illumination and self-shadowing are approximated by means of a novel technique using depth maps. In addition, a new shadow casting algorithm is developed to render shadows produced by the foliage onto regular surfaces which enhances the appeareance of this kind of shadows compared to traditional shadow mapping techniques.

Keywords: Foliage rendering, illumination, shadows, ambient occlusion, shaders, real-time, tree rendering.

1 Introduction

The representation of natural scenes has always been a problem due to the high amount of detail involved in the rendering process. Even a single tree or plant is composed of too much detail in order to be displayed efficiently on the current graphics hardware. To be able to render this kind of scenes a number of different modelling techniques have been developed that represent approximations of the tree with other rendering primitives that are more efficient for the real time [1][2][3].

However, improvements on the graphics hardware over the last years have made possible the development of multiresolution tree models based on geometry [4][5]. These continuous multiresolution LOD models enables us to reduce the geometric complexity of a tree, so that the time needed to display it is also reduced in a significant way.

Due to the scattered nature of the foliage, the standard Phong model is unable to illuminate realistically this kind of tree models, because when simply using a standard local illumination scheme the density of the leaves is lost as well as the real appeareance of tree. On the other hand, traditional radiosity and global illumination solutions, which are suitable for foliage rendering, are very expensive to calculate in real time.

Therefore, special methods for foliage lighting and shading are needed (such as [6]). This work presents a new method for foliage lighting, shading and shadowing that provides good quality illumination in real time while keeping acceptable frame rates. More over, it also provides a shadow casting algorithm of the foliage onto regular surfaces which captures the sense of depth of the foliage.

This document is divided as follows. Sect. 2 presents a state of the art in illumination and shadowing focusing on leaves. Sect. 3 introduces the contribution presented in this paper. After that, Sect. 4 shows the results achieved with our method. Finally, Sect. 5 discusses some aspects of the new method.

2 Related Work

2.1 Illumination

There are some methods to simulate global illumination in real time, such as Precomputed Radiance Transfer [7]. This method uses spherical harmonics to capture low frequency illumination scenarios (including soft shadows and interreflections of objects). On the other hand [8] uses a geometry instantiation system and precise phase functions for hierarchical radiosity in botanical environments.

Mendez et al. [9] introduce Obscurances as a method to simulate diffuse illumination by considering neighbour light contributions instead of the global ones. Ambient Occlusion [10] enhances the illumination of an object by determining the light visibility of each part of the object in a way that the most occluded is an object point the lesser light it will receive from the exterior. Other authors [11] adaptat [10] to the GPU so that the ambient occlusion is computed directly in the fragment shader. Another approach for real-time illumination of trees is [12]. In this work, ellipsoidal occluders that describe the shape of the tree are evaluated at run time.

Reeves and Blau [13] present a tree rendering algorithm which also takes into account lighting and shadowing. The method is based on particle systems. The relative position of each particle inside the tree is used to approximate the illumination and shadowing at a given point.

Jensen et al. [14] present a new model for subsurface light transport which is useful on translucent objects such as leaves. Franzke et al. [15] introduce an accurate plant rendering algorithm using [14] as a leaf illumination method and improving it for leaf rendering.

Finally, [6] presents an expressive illumination technique for foliage. It calculates implicit surfaces that approximate the general shape of the foliage. The implicit surfaces are used both for estimating the global illumination coefficient at a given point and for realigning leaves normals to calculate the diffuse reflection.

2.2 Shadows

Williams introduced shadow mapping for general meshes in 1978 [16]. Although this method is highly suitable for the graphics hardware, its main drawback is aliasing and its memory consumption. Thus lots of authors have suggested their own approaches to solve this problem.

Adaptive Shadow Maps (ASM) [17] reduces aliasing by storing the shadow map as a hierarchical grid. This allows us for huge memory savings, but it is not graphics hardware friendly, because of its hierarchical structure. Arvo [18] proposes to use a tiled grid data structure to tessellate the light's viewport, as a simplified version of ASM. Each cell in this grid contains a sampling density depending on a heuristical analysis.

There are some perspective parametrizations to maximize the area occupied by shadow casters if they are near the observer. This allows for rendering high quality shadows near the camera at the cost of loosing detail, but not quality, on points that are far away from the observer. The most representative shadowing methods that use this scheme are [19][20][21]. Parallel Split Shadow Maps [22] use a similar approach, but it treats the continuous depth range as multiple depth layers. This allows us to utilize better the shadow map resolution.

As seen, there are no few methods around this topic, but there are no specialized shadow casting methods for foliage that takes into account the leaves structure and its spread nature.

3 Contribution

This paper presents a method to handle the illumination of foliage, taking into account both direct and indirect illumination contributions as well as the auto-occlusion information of the foliage itself. Moreover the method also handles the shadow projection of the foliage onto other surfaces.

Our approach is based on the rendering equation introduced by Kajiya [23] described as follows:

$$L_o(x, \boldsymbol{w}) = L_e(x, \boldsymbol{w}) + \int_{\Omega} f_r(x, \boldsymbol{w'}, \boldsymbol{w}) L_i(x, \boldsymbol{w'})(\boldsymbol{w'} \cdot \boldsymbol{n}) d\boldsymbol{w'} \tag{1}$$

where the amount of light irradiating from an object $L_o(x, \boldsymbol{w})$ at a given point x and direction $\boldsymbol{w}$ depends on the light the object emanates $L_e(x, \boldsymbol{w})$, which we can ignore because leaves do not emit light, and on the incoming light $L_i(x, \boldsymbol{w'})$. Light incoming from all directions is modulated with the angle of incidence of the light onto the surface $\boldsymbol{w'} \cdot \boldsymbol{n}$ and the BRDF $f_r(x, \boldsymbol{w'}, \boldsymbol{w})$ that describes the reflectance function, which depends on the material properties.

This paper discusses how to implement each part of the Eq. 1 to provide a realistic illumination for the foliage in real-time.

Finally, a new shadow casting algorithm is introduced which takes into account leaf density information at a given light direction to render realistic foliage shadows over other surfaces, such as the ground or the trunk.

3.1 Indirect Illumination Contribution

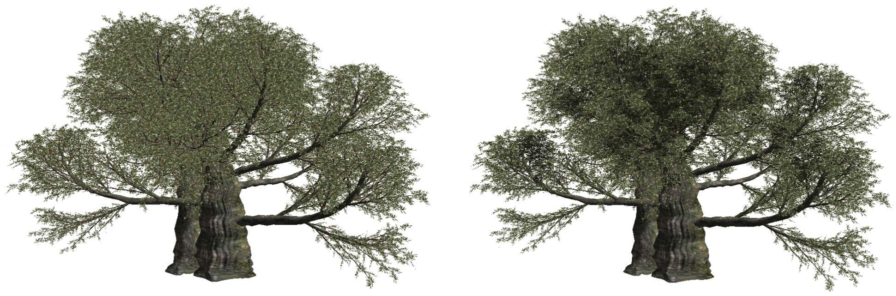
Although Eq. 1 takes into account the light incoming from all directions, evaluating all light directions would be very expensive. Thus, we apply our BRDF calculations only over the light direction which comes directly from the light source, separating the direct from the indirect light contributions. Therefore, we obtain the following formula for light irradiance at a given point x and direction

$\boldsymbol{w}$, where the direct light contribution is sepparated from the indirect lighting ($A_\Omega(x)$):

$$L_o(x, \boldsymbol{w}) = A_\Omega(x) + f_r(x, \boldsymbol{w}', \boldsymbol{w}) L_i(x, \boldsymbol{w}')(\boldsymbol{w}' \cdot \boldsymbol{n}) \qquad (2)$$

The method proposed replaces the A_Ω term in Eq. 2 by a new indirect lighting algorithm specifically designed for the foliage.

The indirect illumination of our model is calculated as a preprocess step. The amount of light a leaf receives from the scene is calculated as the visibility of the leaf from the exterior of the foliage. Fig. 1 shows the results of applying our indirect light algorithm.



(a) Tree foliage without illumination. (b) Indirect lighting affecting the foliage.

Fig. 1. Results of our indirect lighting approach

The ambient light received from a leaf depends on the visibility of each leaf from the exterior of the foliage. As all leaves on the tree are of the same size, the visibility value for each leaf is compared with the visibility value of a single leaf, which is given by the orthogonal projection of the leaf over a known virtual viewport.

This has been implemented by rendering each leaf with a unique colour with a depth buffer activated. Thus, the visibility of each leaf from outside the tree is given by the number of pixels of the same colour on the six faces of a cubemap surrounding the foliage. To represent the colour of the ambient light affecting each leaf, a texture read operation is performed over a downsampled cubemap that contains the environment of the tree. The normal of each leaf face is used as texture coordinates to fetch this data from the cubemap. Fig. 2 shows a tree illuminated using only indirect illumination with different scene light ambient absorption.

This visibility calculations are performed per face, because each face of a single leaf can receive different amounts of light with a different colour, depending on the scene and the depending on the direction the leaf is facing. However, light tend to spread across and through the leaf, depending on its translucency. Thus, the transparency level of the leaf is used to add light reception values from both

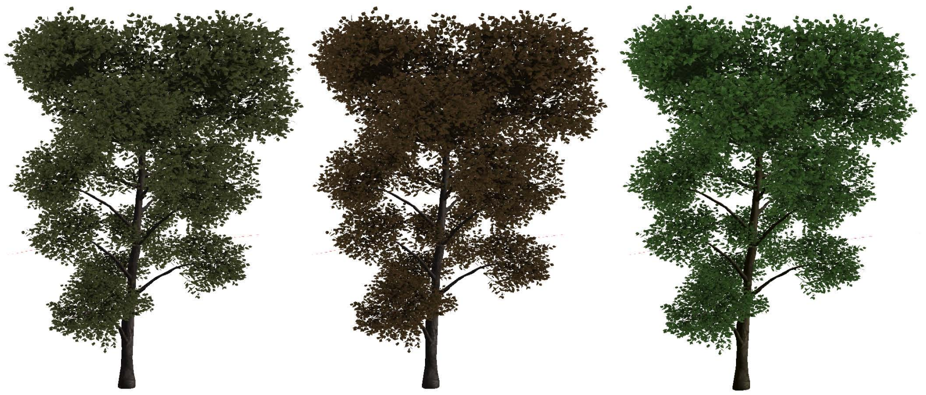


Fig. 2. Different views of a tree with different ambient light contributions. From left to right: white, reddish and yellow light.

sides of the leaf. Thus, the light scattering property of the leaves is taken into account to calculate the ambient occlusion term.

Finally, the ambient occlusion colour A_Ω for a given face of each leaf i is calculated as shown in Eq. 3.

$$A_\Omega = [C_i I_i (V_i/V)^n + \alpha[C'_i I'_i (V'_i/V)^n] \tag{3}$$

where α is the transparency of the leaf, I_i is the colour the current face of the leaf i absorves from the scene, I'_i is the colour the opposite face of the leaf absorves from the scene, C_i and C'_i represent the colour of the front and opposite faces of the leaf i, V_i and V'_i are the number of pixels generated by the current and opposite faces of the triangle i respectively and V is the number of pixels generated the projection of a reference leaf without occlusion. The parameter n is always positive and controls how rapidly the darkening for the ambient term occurs in the foliage. Values of $n > 1$ will result in a more rapid darkening and values of $n < 1$ will cause the darkening to slow down from outer to inner parts of the foliage.

The results of this process are stored per vertex so that it can be applied per leaf at run time.

3.2 Direct Illumination and Self-shadowing

Our illumination method has been developed having in mind the nature of the leaves in order to simulate the complex interaction of the light inside the foliage. A visual analysis of the light interaction with the foliage provides a simple conclusions about this issue: both inner leaves as well as those leaves in the opposite side from the light source appear darker. This is caused due to the leaves auto-occlusion which impedes the light to reach those leaves and makes them receive less light and appear darker.

Our approach is based on capturing the general shape of the foliage from the light source and illuminate each leaf depending on its position inside the foliage volume to approximate the self-shadowing of the leaves. We use two depth maps capturing the nearer and further parts of the foliage from the light source which we will call D_n and D_f. The main idea is that the nearer a leaf is to D_f respective to D_n the darker it should appear. In addition, another texture C is used to determine the amount of leaf intersections per pixel, from the light source. This value will be used to determine the leaf density in a given direction.

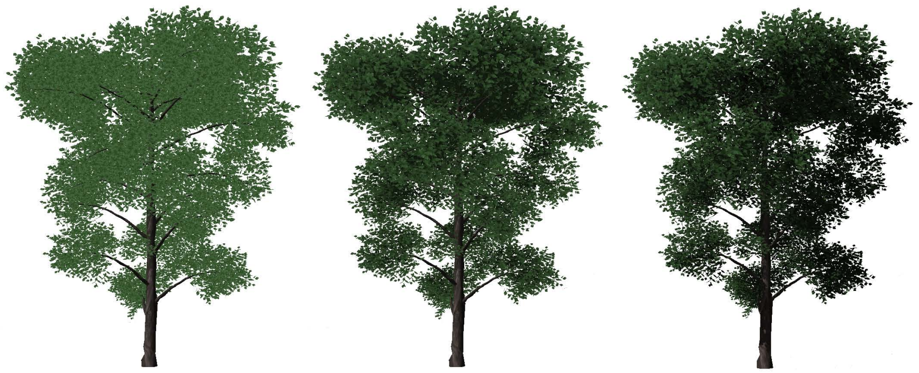


Fig. 3. Left: foliage without illumination. Middle: ambient lighting only. Right: the complete illuminaton system, including direct and indirect lighting, self-shadowing and shadows casted over the trunk and branches.

When rendering a leaf at run time, the pixel shader calculates the position of the leaf relative to the light source and calculates the adequate texture coordinates to access the depth maps, in the same way like in traditional shadow mapping. The shader compares the depth of the leaf in light space with the minimum and maximum depths at that point to determine its proximity to those values. This value is weighted with the value contained in the texture C which determines the amount of leaves at that light direction. Eq. 4 shows the formula used to calculate the self-shadowing factor S of the leaf depending on the light source i.

This shadowing factor replaces the $L_i(x, \boldsymbol{w}')$ term in Eq. 2:

$$L_i(x, \boldsymbol{w}') = \alpha N_c \left(1 - \frac{Z_x - Z_n}{Z_f - Z_n}\right) \tag{4}$$

where α is the transparency level of the leaf, N_c is the number of leaf collisions at a certain light direction given by the texture C, Z_x is the depth of the current leaf fragment in light space, and Z_n and Z_f are the minimum and maximum depths in that light direction given by textures D_n and D_f respectively.

Therefore Eq. 4 provides the darkening factor for each pixel depending on the light source direction and the general shape of the foliage volume. The results of this equation matches to the light intensity function of Eq. 2. Fig. 3 shows an example of our illumination approach for the foliage.

The direct lighting contribution of the foliage is calculated in the following way. Due to the translucent nature of leaves, a subsurface scattering based BRDF is needed to correctly simulate the illumination on the leaves. Jensen et al. [14] propose an efficient method for subsurface scattering which separates the scattering process in a single scattering term $L^{(1)}$ and a diffusion approximation term L_d, as shown in Eq. 5.

$$L(x, \boldsymbol{w}) = L^{(1)}(x, \boldsymbol{w}) + L_d(x, \boldsymbol{w}) \tag{5}$$

Frankze et al. [15] show how Eq. 5 can be approximated as shown in Eq. 6 due to the minimum thickness of the leaves providing a method easier to evaluate in real-time:

$$f_r(x, \boldsymbol{w}', \boldsymbol{w}) = L^{(1)} + L_d = (1 + e^{-s_i} e^{-s_o}) L_i(x_i, \boldsymbol{w}') \cdot (\boldsymbol{N} \cdot \boldsymbol{w}') \tag{6}$$

Where s_i is the leaf thickness and s_o is a random outgoing distance inside the material from the actual sample position. This approximation matches the $f_r(x, \boldsymbol{w}', \boldsymbol{w})$ component in Eq. 2 and describes the BRDF associated to the direct illumination.

Direct illumination is evaluated in the pixel shader fetching some parameters from textures such as leaf thickness and normal information.

3.3 Shadow Casting Over Other Surfaces

While we have covered the lighting interaction of the leaves, how the light penetrates across the foliage and reaches another surfaces is also important. The foliage can be seen as a set of multiple layers of translucent leaves. Therefore, the amount of shadowing other objects receive from the foliage depends on how many leaves intersect a light direction and their amount of transparency. To simulate this, we use the texture C (see Sect. 3.2) which stores the amount of leaf intersections at a given direction weighted with the leaf transparency at each point.

This texture can be calculated in a single pass and updated along with the others shadow maps. We use this information to render more realistic shadows over surfaces, where the depth of the foliage is taken into account to visualize more convincing foliage shadows.

Fig. 4 shows the final appeareance of our shadow casting algorithm. Notice how the shadow map shows the depth of the foliage, being more opaque where there is more leaf density throught the light direction, and being more transparent where there is lower leaf density.

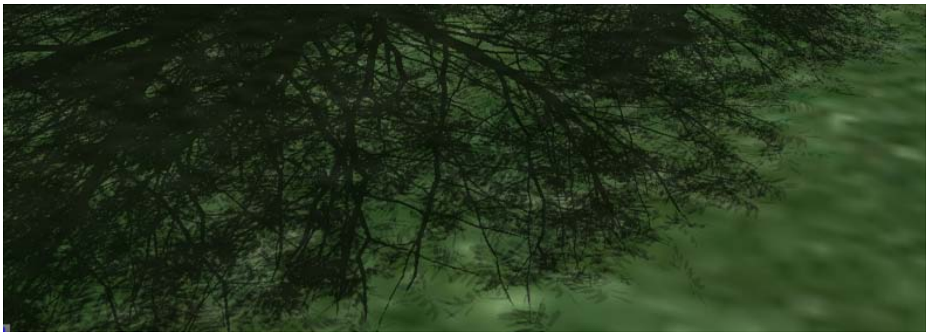


Fig. 4. A detailed view of our shadow mapping aproach for the foliage. Notice how the depth of the foliage is captured in the shadows.

4 Results

In our tests, we have used a geometry-based continuous level of detail algorithm for the foliage. This allows for performance optimizations when rendering the forest scene with such amount of geometric trees.

Fig. 5 shows the results of our illumination solution. Notice how the illumination captures general shape of the foliage, darkening those parts that are difficult to reach for the light.

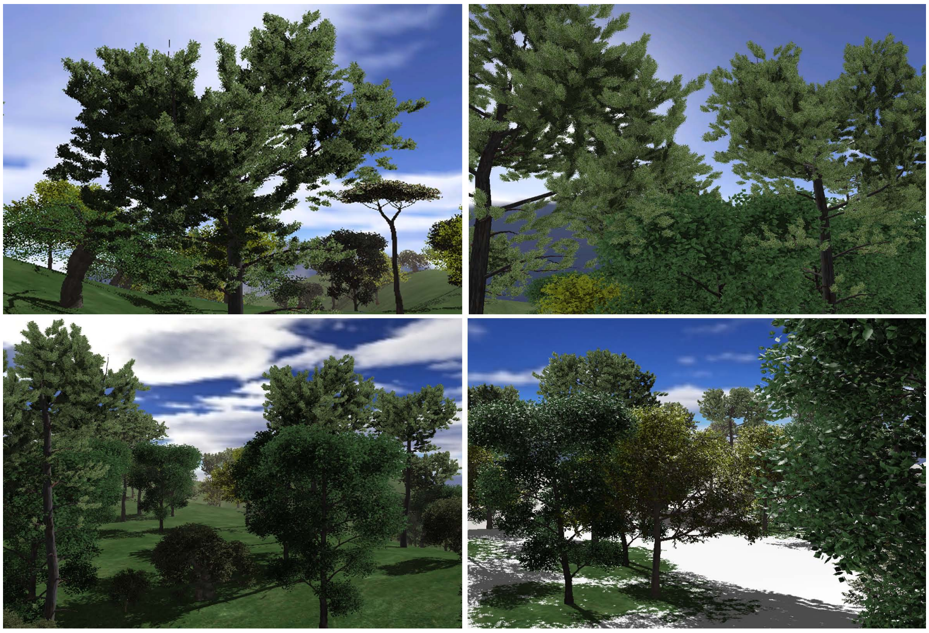


Fig. 5. Forest scenes with our illumination and shadowing approach

This method for foliage illumination requieres to access to three different depth maps per pixel to evaluate the illumination equation. However, this is optimized to require just one texture read by packing all textures in a single three channel floating point texture. Thus the overhead of applying this method is just one texture read and a few arithmetical operations in the pixel shader. To obtain a proper auto-occlusion shadowing of the foliage without this method a shadow map would be needed and also a texture read and some arithmetical operations would be needed as well. Therefore, applying our method adds a little overhead in these cases, being the main drawback to store three values per texel instead of just one. However, the visual quality of this method for foliage illumination justifies this storage overhead.

The cost of casting foliage shadows over the ground or any other surface (as the trunk) is negligible compared to a standard shadow mapping algorithm, as the only difference is what the shadow map contains and a couple of arithmetical operations in the pixel shader.

5 Conclusions

This paper presents an approach for foliage illumination and expressive leaves shadow casting algorithm which is applicable in real time. The algorithm is based on depth maps. This means that in scenarios where shadow mapping is being used to simulate the shadows of the foliage as well as the auto-occlusion of the leaves, this method will improve the visual quality of the scene at the expense of little computational cost.

We decided to calculate the ambient occlusion factor as a preprocessing step because the static nature of trees. Trees are always located in the same place in the space. Therefore we take this into account to accelerate the ambient occlusion calculation by preprocessing it and storing as per-vertex attributes. Thus, the cost of applying the ambient occlusion is negligible.

As said before, this method uses depth maps as the base tool to infer the illumination and to render the shadows, such as trapezoidal, perspective, light-space perspective shadow maps. This method is built on top on existing shadow mapping algorithms, dealing with the meaning of the information contained at each texel in the shadow map, so it does not compete with other shadow mapping methods, but extends them.

Altough we have used geometry-based trees in this article, the algorithm is also applicable to image-based or point-based trees because the information needed to calculate the illumination is stored in a separate map and is not attached to geometry [6] which is a restriction in real-time rendering.

Acknowledgments. This work has been supported by grant P1 1B2007-56 (Bancaixa), the Spanish Ministry of Science and Technology (Contiene Project: TIN2007-68066-C04-02) and FEDER funds.

References

1. Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., Prusinkiewicz, P.: Realistic modeling and rendering of plant ecosystems. In: SIGGRAPH 1998, New York, NY, USA, pp. 275–286 (1998)
2. Dietrich, A., Colditz, C., Deussen, O., Slusallek, P.: Realistic and Interactive Visualization of High-Density Plant Ecosystems. In: Natural Phenomena 2005, pp. 73–81 (August 2005)
3. Colditz, C., Coconu, L., Deussen, O., Hege, H.: Real-Time Rendering of Complex Photorealistic Landscapes Using Hybrid Level-of-Detail Approaches. In: Conference for Information Technologies in Landscape Architecture (2005)
4. Rebollo, C., Remolar, I., Chover, M., Gumbau, J., Ripollés, O.: A clustering framework for real-time rendering of tree foliage. Journal of Computers (2007)
5. Rebollo, C., Gumbau, J., Ripolles, O., Chover, M., Remolar, I.: Fast rendering of leaves. In: Computer Graphics and Imaging (February 2007)
6. Luft, T., Balzer, M.: Deussen O. Expressive illumination of foliage based on implicit surfaces. In: Natural Phenomena 2007 (September 2007)
7. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: SIGGRAPH 2002, New York, USA, pp. 527–536 (2002)
8. Soler, C., Sillion, F., Blaise, F., Dereffye, P.: An efficient instantiation algorithm for simulating radiant energy transfer in plant models. ACM Trans. Graph, 204–233 (2003)
9. Méndez, A., Sbert, M., Catá, J.: Real-time obscurances with color bleeding. In: SCCG 2003, New York, USA, pp. 171–176 (2003)
10. Pharr, M., Green, S.: Ambient Occlusion (2004)
11. Bunnell, M.: Dynamic Ambient Occlusion And Indirect Lighting (2005)
12. Hegeman, K., Premoze, S., Ashikhmin, M., Drettakis, G.: Approximate ambient occlusion for trees. In: Sequin, C., Olano, M. (eds.) SIGGRAPH 2006, ACM SIGGRAPH, New York (March 2006)
13. Reeves, W., Blau, R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. In: SIGGRAPH 1985, New York, USA, pp. 313–322 (1985)
14. Jensen, H.W., Marschner, S.R., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: SIGGRAPH 2001 (2001)
15. Franzke, O.: Accurate graphical representation of plant leaves (2003)
16. Williams, L.: Casting curved shadows on curved surfaces. In: SIGGRAPH 1978, New York, USA, pp. 270–274 (1978)
17. Fernando, R., Fernandez, S., Bala, K., Greenberg, D.P.: Adaptive shadow maps. In: SIGGRAPH 2001, New York, USA, pp. 387–390 (2001)
18. Arvo, J.: Tiled shadow maps. In: Proceedings of Computer Graphics International 2004, pp. 240–247 (2004)
19. Stamminger, M., Drettakis, G., Dachsbacher, C.: Perspective shadow maps. In: Game Programming Gems IV (2003)
20. Wimmer, M., Scherzer, D., Purgathofer, W.: Light space perspective shadow maps (June 2004)
21. Martin, T., Tan, T.S.: Anti-aliasing and continuity with trapezoidal shadow maps. In: Rendering Techniques, pp. 153–160 (2004)
22. Zhang, F., Sun, H., Xu, L., Lun, L.K.: Parallel-split shadow maps for large-scale virtual environments. In: VRCIA 2006, pp. 311–318 (2006)
23. Kajiya, J.T.: The rendering equation. In: SIGGRAPH 1986, pp. 143–150 (1986)