Direct Illumination of Dense Foliage Using Z-Buffer

Bedřich Beneš
DCSE, Czech Technical University,
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
e-mail: benes@sgi.felk.cvut.cz

Abstract

Estimation of the light flux reaching light sensitive organs of a visual model of a plant is an interesting problem in computer graphics as well as an important problem from the viewpoint of biology. One kind of algorithm solving this problem uses Z-buffer sampling of the plant either in the direction from the leaf to the light or vice versa. The problem of aliasing and consequent error of the estimation is known but has not been discussed in this context up to now. We will try to show a way to predict an error of estimation and how to find sampling resolution according to the plant character.

1 Introduction

Simulation of plant development presents to date a vibrant area which bridges the gap between biology and computer graphics. A large number of papers has been published regarding this topic. An important part of the current research in computer graphics presents developing of algorithms dealing with interaction between a plant and its environment. One of the most interesting parts from the computer graphics viewpoint as well as one of the most important parts from the viewpoint of biology includes evaluation of the light incidenting onto surfaces of the plant organs. In the paper, we will focus on methods dealing with this problem from the computer graphics viewpoint. The main aim of the paper is to show that Z-buffer sampling is a useful and efficient technique in the simulation of the plant development for the estimation of the amount of light reaching plant organs.

This paper is structured as follows. After the previous work section we shortly describe our sampling technique and its pros and cons. Section 4 discusses the problem of a completely full raster which arises when very dense foliage is viewed from the light position. The next section describes an average visible area of the polygons in the raster and the problem of a sampling resolution is discussed in Section 6. The paper concludes with some results in Section 7.

2 Previous Work

The algorithms for calculation of the amount of light in a visual model of a plant can be classified into two, partially overlapping, classes. The first class of algorithms deals with vegetation from the point of *rendering* and this class includes algorithms for multiple scattering, *e.g.* [2, 12, 14], as well as techniques for the estimation of light inside dense foliage [18]. On the other hand, the algorithms belonging to the second class estimate light flux reaching only those parts of the plant which are sensitive to the light, *i.e.* leaves, buds, and blossoms, in order to simulate the *plant development*.

We have found several algorithms belonging to the second class of the published methods. Greene [10] uses sampling of the sky with sun trajectory for very crude light estimation. A similar method was also published by Arvo and Kirk in [1]. In both cases the amount of light is used just for managing the growth of the plant in the direction of incoming light and, in the case of Arvo and Kirk, for lengthening the blade of grass.

According to biology, every plant grows in such a way that the light reaching a leaf produces photosynthates which are then transported to the adjacent bud which gives a proper response; *i.e.* it grows or it produces a blossom. This fact was captured by Chiba *et al.* [6, 7] who introduced an algorithm for the precise estimation of the incoming light. The algorithm is *de facto* a kind of a clustering technique which uses either ray–casting or Z-buffer sampling from every cluster to the light. The ray–casting version of this approach was also used by Měch and Prusinkiewicz [17].

Another method for the estimation of the amount of light reaching the plant is based on Z-buffer sampling in the opposite direction than in the work of Chiba *et al.* that is from the light sources. This algorithm was introduced by Beneš [3, 4, 5]. This paper presents a discussion of this technique, but the results can be applied in the Z-buffer sampling version of the algorithm of Chiba as well. The algorithm principle is described in Section 3. We have a classic aliasing problem here. Although the problem of Z-buffer aliasing was discussed in the context of hemi cube sampling in radiosity, *e.g.,* [9, 16], and Z-buffer sampling was even used for reconstruction of the model of a plant [13, 15], we have not found the aliasing problem discussed in the context of simulation of plant development where it arises in a specific way.

3 Method Description

Here we shortly recall our technique for the estimation of the amount of light reaching plant organs using Z-buffer sampling from the light sources (*c.f.* [3, 5] for more details).

Without loss of generality we assume the plant to be developing in outdoor conditions, and the lights in the scene to be substituted by a finite set of parallel light sources. This is achieved by approximation of the celestial hemisphere by polyhedron (see Figure 1), with faces denoted by F_k, $k = 1, \dots, m,\ m \in N$

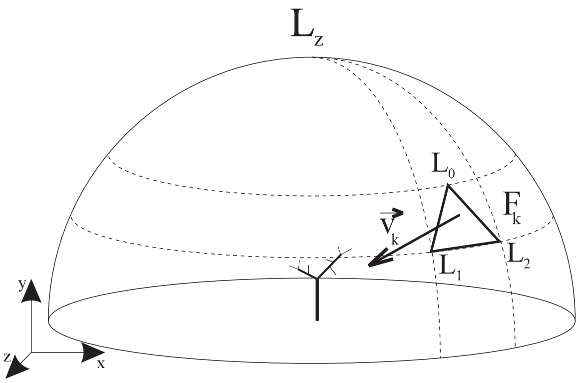

Figure 1: Light source S_k is assigned according to corner radiances of the face F_k.

The flux density of every face is denoted by B_k and it is computed as an average of radiances L_0, L_1, and L_2, of the points in its corners. We use the CIE equation for completely clear sky according to [8, page:296]. In the next step, every face F_k is substituted by a parallel light source denoted by S_k which emanates perpendicularly to the F_k towards to the center of the hemisphere. Once these values are assigned, we are able to evaluate the area of a leaf which is affected by the light incoming from S_k and therefore the incoming radiant flux, *i.e.,* the amount of incoming energy.

The leaves used to be modeled as curved surfaces, and, although the Z-buffer can render arbitrary surfaces, it is rather complex and impractical to calculate their illumination. That is why such objects are usually converted into a set of triangles. It is important that, according to the properties of the tessellation algorithms, we can assume the triangles to be "well behaved", *i.e.* equilateral. We will discuss only light reaching the i-th triangle and we denote its area by A and its effective projected area (area affected by light), by A_i^e.

Without loss of generality, we assume frame buffer to cover the unit area, and thus $A < 1$. Resolution of the frame buffer is denoted by r, and therefore one pixel covers an area of $1/r$.

The amount of light is evaluated simply by rendering the plant using the Z-buffer algorithm. We position our viewing camera perpendicularly to the face F_k and we set the clipping planes in such a way that the plant is completely visible. Thus, what we see in our camera is affected by the light S_k. We use orthographic projection, so our sampling does not depend on the distance from the plant. Every triangle of each light–sensitive organ has a different color assigned, and the rest is colored as background (*item buffer*). Constant shading and no lighting is used. When the scene is successfully rendered, we capture the resulting frame buffer image and compute its histogram. This histogram is denoted by h and the $i-$th element h_i carries information about the visible area of the $i-$th triangle, which is affected by the light coming from the light source S_k. The visible (or effective) projected area of the triangle previously denoted by A_i^e is easily calculated as

$$A_i^e = \frac{h_i}{r}, \tag{1}$$

and the total light flux $\Phi_{i,k}$ reaching the i-th triangle from S_k is

$$\Phi_{i,k} = A_i^e B_k.$$

The algorithm has several advantages over the ray–casting method. First, the number of samples is constant and it is equal to the number of the lights, *i.e.* to the precision of the sky light approximation. Some hierarchization of the scene such as in [11] would improve efficiency of the algorithm significantly. Of course, we can use the hardware of our workstations for the Z-buffer visibility problem. The later makes the algorithm very attractive.

The algorithm has some drawbacks as well. We have not found a way to incorporate transparency of leaves up to now, so this method is useful only for direct light and no transparency, and even scattering cannot be implemented. The second and more important problem is that the algorithm suffers from aliasing caused by finite resolution of the raster. We will focus on the latter problem here.

4 How Many Triangles Are Really Visible?

The basic assumption is that some leaves in dense foliage are completely hidden, and therefore we may ask how many triangles from the total amount are really visible. We will make some simplifying assumptions allowing us to find the order of the function.

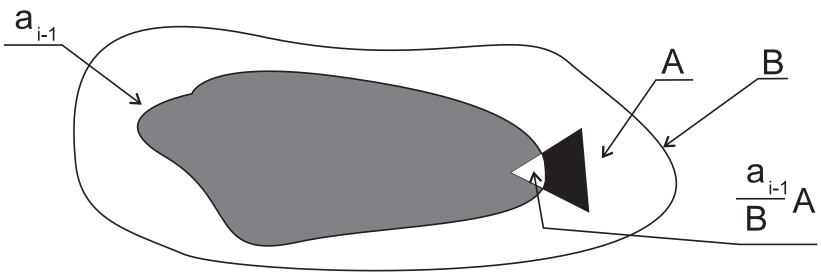

Figure 2: Notions for the equation 2.

We have an area denoted by B (this is typically the unit area) and we fill it by equilateral triangles of area A, which are laid out randomly with equal probability distribution. The triangles might, of course, overlap. Next, we denote by a_i the area covered by $i = 1, \ldots, n$ triangles. Obviously $a_0 = 0$, there is no triangle, and $a_1 = A$. If we have an area a_{i-1} (see Figure 2) and we add one new triangle then the probability that one single point would fall within the area covered by a_{i-1} and the newly added triangle is a_{i-1}/B. Thus probability that area A of the newly added triangle would cover an area already covered by a_{i-1} is $A\ a_{i-1}/B$. Then, with every new triangle we contribute by a new area of $A - A\ a_{i-1}/B$ and this gives

$$a_i = A - \frac{a_{i-1}}{B}A + a_{i-1} = a_{i-1}\left(1 - \frac{A}{B}\right)A \tag{2}$$

Next we let $c = 1 - A/B$ and then we have:

$$\begin{aligned} a_1 &= A \\ a_2 &= Ac + A \\ \dots \\ a_i &= Ac^i + Ac^{i-1} + \dots + A \end{aligned}$$

and clearly

$$a_i = B - B\left(1 - \frac{A}{B}\right)^i. \tag{3}$$

We measured and estimated this task for triangles covering 0.1%, 1% and 10% of the unit area. In order to keep the error of aliasing as low as possible we used resolution 1M pixel and every scene was rendered one hundred times using jittered sampling. The final value was then computed as average of the samples. We found that 99.7% of the area is filled after a certain number of rendered triangles, and this value does not change then. Results are shown in Figure 3 a).

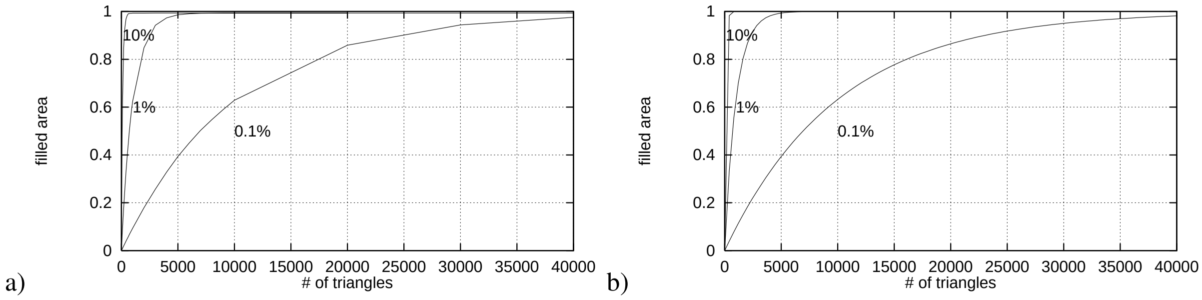

Figure 3: Measured a) and estimated b) coverage of the unit area.

It is easy to derive from (3) the total number of triangles of area A which will cover the unit area from at least 99.7% which is a stable value from our real tests. We denote this number of triangles by n_{estim} and from (3)

$$n_{estim} = \frac{\ln 0.003}{\ln(1 - A)} \tag{4}$$

Table 1 compares the estimated number n_{estim} and the measured one n_{max}.

A	n_{max}	n_{estim}
10%	800	687
1%	5000–6000	6904
0.1%	60000–70000	69074

Table 1: Total amount n_{max} compared to n_{estim}.

We can see in Figure 3 that the estimated results match quite well with the measured ones. The important conclusion is that the area is filled after some number n_{max} of triangles and any new triangle will *not* contribute to the error due to aliasing. In other words, in very dense foliage only a small part of the leaves is really affected by direct light. We conclude this section by saying that the total amount of *visible* triangles, which are randomly put in the scene will be according to (4) equal or less than n_{estim}.

5 Average Visible Area

The triangles in the scene are subject of mutual occlusions. It manifests itself in such a way that the effective area of the triangle A_i^e, which is obtained from (1), is usually smaller than A_i. Thus we may ask what the *average visible area* of a triangle in a completely full scene will be. We will denote this value by $\overline{A}$.

We have tried to find this function analytically several times but it is a surprisingly hard task and unfortunately we have not yet succeeded. An approximate value $1/n_{estim}$ can be used from (4). But this is really a crude approximation taking no regard of the geometry of the objects.

The easiest way how to get the value is its measuring. We rendered a random plant having an area of one triangle 1% of the raster. The total number of triangles in the foliage was ten times higher than n_{esitm}. We repeated this measurement procedure one hundred times and the average result is presented in the first row of Table 2. Then we measured the second row of the table for $A = 0.5\%$, *etc.*.

A	$\overline{A}$	%
0.01	0.002418	24.18
0.005	0.001287094	25.74
0.001	0.0002815023	28.15
0.0005	0.0001507789	30.15
0.0001	0.00003938336	39.38

Table 2: Average visible area of triangles having area A from an outside view of dense foliage.

Table 2 represents typical sizes of triangles in the virtual plants. In the next text we will suppose that the triangle has an area equal to the middle value of the table, *i.e.,* 0.1% of the raster. Thus the average visible area is approximately 0.028% of the raster.

We will see in Section 6 that the error of our estimation depends on the ratio of the area of an object and its border. Thus small objects contribute to the error of our estimation more significantly.

According to the previous idea, an important question is how many leaves will be smaller than our average area. The advantage is that there are still many leaves which are completely exposed to the light, thus a very large area of our raster image is covered by completely unoccluded triangles. Figure 4 presents this fact. We have measured one hundred random trees with very dense foliage. The triangles of the trees had an average area of a triangle equal to the typical case from Table 2, *i.e.* 0.1%. The x axis carries information about the size of a triangle in percentage of the area A. The y axis indicates how important the part of a leaf is. This value was computed as the total area covered by part of triangles of equal size, *i.e.* as the number of part of triangles of equal size multiplied by its area. The average value $\overline{A}$ is also indicated in the figure. The area below $\overline{A}$ represents about 15% of the whole image.

6 Sampling Resolution

Finally, we will try to say how large the resolution of the frame buffer image would be if the average area of leaves is known and we know that the raster was fully occupied. For this reason we need to find how precisely the triangle is represented in a raster.

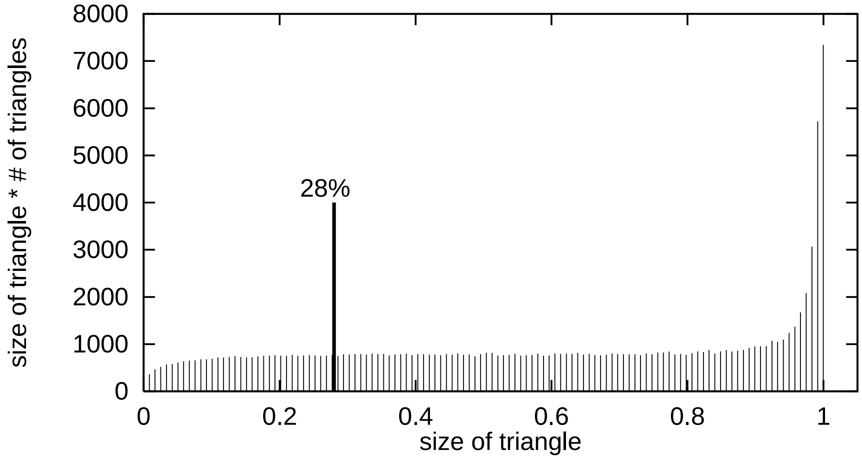

Figure 4: The importance of the part of triangles. The x-axis indicates the percentage of the area of the triangle whereas the y-axis says how large the total area covered by those parts is.

The error of the rasterization algorithms comes from assuming the polygon visible at the sampling point, located in the center of the pixel, is visible inside the *whole* pixel. This error does not manifest itself in the inner pixels of the raster representation of the triangle but merely on the border because constant color is assigned to the triangle. This situation is shown in Figure 5.

The rasterization will cause some protrusion and intrusion of the border of the triangle. The area of a triangle is denoted by A and we denote the estimate by $\tilde{A}$. The estimate will differ at most by ε *i.e.*

$$\tilde{A} = A \pm \varepsilon. \tag{5}$$

In other words, if we measure one triangle randomly put many times in a raster, the expected value of average of A_i^e will converge to its area A whereas the variance will be nonzero.

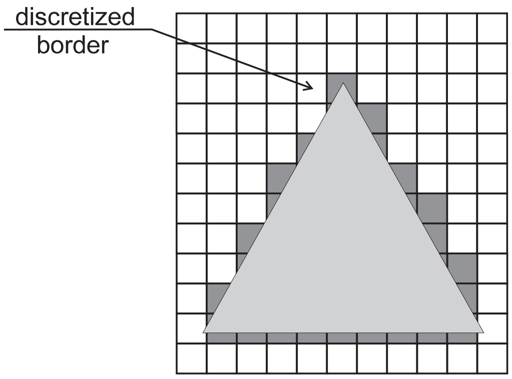

Figure 5: The border of a triangle contributes to the error of its raster representation.

We can assume the area A to be bigger compared to the area of one pixel $1/r$ otherwise the triangle would not appear on display at all. We use this assumption without fear, as presented in the previous sections.

The error ε depends mainly on the border of the triangle. Assume the worst case, *i.e.,* all pixels passed by the border will be either added or subtracted from the A in equation (5). We denote the length of the border by l.

$$l = q\sqrt{A}, \tag{6}$$

where q is a term depending on the geometry of the area A. It is easy to show that in the case of the equilateral triangle, this term is equal to $2/\sqrt[4]{3}$.

The discretized border occupies some area which can be thought of as a rectangle having one side of length l and the second side of half pixel length *i.e.* $1/\sqrt{r}$. From this fact and from (6) we have

$$\varepsilon = q\sqrt{\frac{A}{r}}. \tag{7}$$

For the case $q = 2/\sqrt[4]{3}$ and $A = 0.001$ the error is equal to

$$\varepsilon = \frac{0.04806}{\sqrt{r}}.$$

We have tested the error of one triangle having an area equal to 0.1% of the total area displayed. We rasterized this triangle one hundred times in random positions and we have measured the worst cases. The estimates $A \pm \varepsilon$ as well as the measured values are displayed in Figure 6.

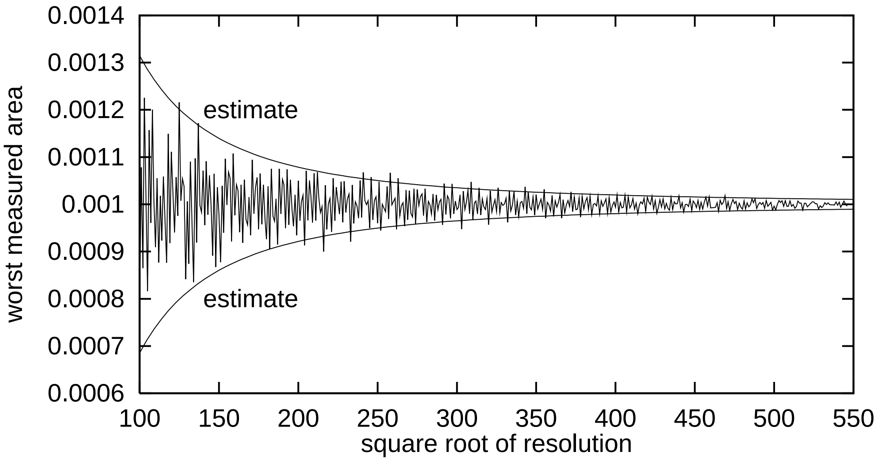

Figure 6: Variance of the area of a triangle as a function of the resolution.

In the end, we may finally ask the following question. Let us have a plant with leaves polygonized into equilateral triangles of area A. We have completely full frame buffer image by triangles having an average area $\overline{A}$ in a typical case equal to 28% of A. We want the error of representation of one triangle to be smaller than a certain value denoted by η. This value is expressed *e.g.* as a percentage of the size of the triangle

$$\varepsilon = \eta\overline{A} \tag{8}$$

It is an easy task now to express the desired resolution from equations (7) and (8) as

$$r > \frac{q}{2\eta\sqrt{\overline{A}}} \tag{9}$$

7 Results

A typical triangle, produced by a polygonizer of some curved surfaces, covers approximately 0.1% of the projected area, *i.e.* $A = 0.001$. According to Table 2, the average projected area will be $\overline{A} = 0.28A$. Assuming the triangles to be equilateral, we may use the geometrical term $q = 2\sqrt[4]{3}$. We wish the error of one triangle to be at most

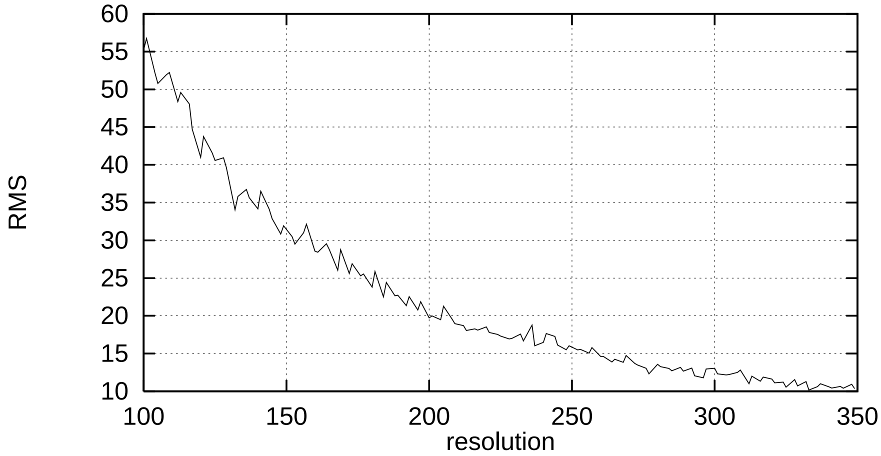

Figure 7: Error caused by aliasing of the full scene rendered.

0.1% of its area A, which gives $\eta = 0.001$. According to these values, equation (9) gives a rather small resolution $r > 213 \times 213$ pixels.

Figure 7 presents normalized RMS error between the sampled image in square root of resolution drawn at the x-axis and the control scene. The control scene was sampled in resolution 1M pixel and jittered one hundred times. We can see that the error is higher than the estimated one, which was 0.1%. The reason is probably the accuracy of the geometrical term q which captures the geometry of a triangle roughly and, more important, it does not reflect changes in geometry caused by occlusions and rotations. So we may have very long and thin triangles in scene and these triangles certainly have the q term greater.

We also performed one very simple visual test. We approximated the sky hemisphere by one thousand light sources and we estimated light reaching leaves of a very dense maple as shown in Figure 8. Because the CIE equation for the sky radiance also captures the position of the sun, it was easy to animate the plant illuminated by the sun. The plant foliage is very noisy information, and according to our estimation, the suitable sampling resolution would be greater than 200×200 pixels. Using this resolution, the plant was slightly flickering in some parts. When the resolution was increased up to 300×300 pixels, the final animation was very smooth and without any visual artifacts.

8 Conclusions

We have presented that the Z-buffer sampling is useful for computing the outside view of a plant when affected by light. We have found that for an "average" visual model of a plant, consisting of leaves approximated by a set of triangles having an area of about 0.1% of a raster, sampling resolution 300×300 pixels is useful. If we want more leaves in scene to be sampled by Z-buffer we can easily derive resolution from equation (9). For doing this, we need to know the average area of a visible triangle which is typically 28% of the area of one triangle and the geometrical term q which captures how much the border of one triangle contributes to the error of the estimation.

Future work includes precise derivation of the q term which is probably a function of A as well as of the number of visible triangles in scene.

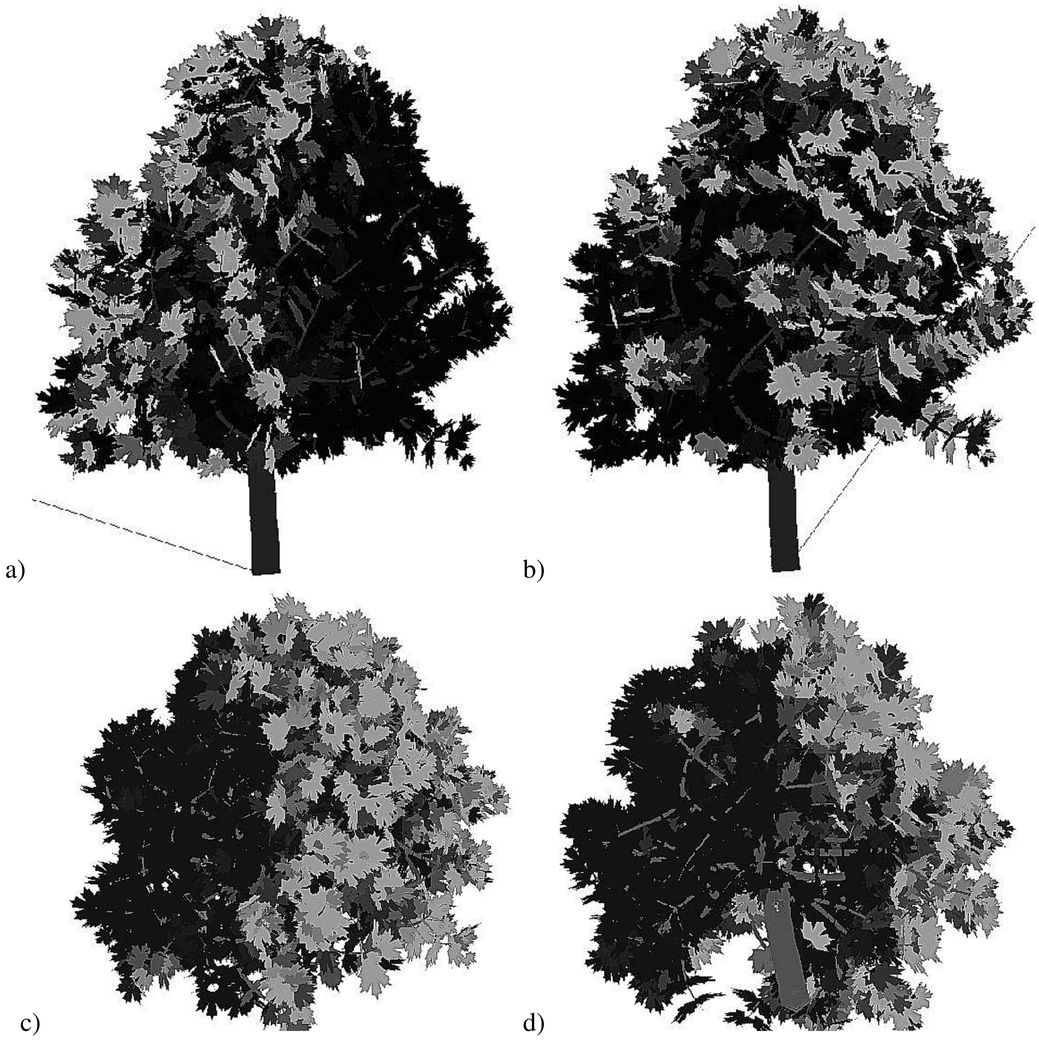

Figure 8: Very dense maple when the sun is a) just above the horizon and b) near the zenith (the line points to the sun). The sky was approximated by 1000 light sources, and the sampling resolution was 300×300 pixels. Only the irradiance incidenting onto surfaces of the leaves is displayed. Top c) and bottom view d) of the same maple when the sun is just above the horizon at the opposite side.

9 Acknowledgments

I would like to thank Jiří Bittner and Vlastimil Havran for fruitful discussions and their help with equation (3), Lia Hejdová for proofreading of the text, and the anonymous referee for many comments which helped clarify the paper.

This work was supported by the CTU Grant 30/98097/336.

References

[1] J. Arvo and D. Kirk. Modeling Plants with Environment-Sensitive Automata. In *Proceedings of Ausgraph'88*, pages 27–33, 1988.

[2] G. Baranoski and J. Rokne. An Algorithmic Reflectance and Transmittance Model for Plant Tisue. In *Computer Graphics Forum*, volume 16(3) of *Eurographics Proceedings Series*, pages C–141–C–150. Eurographics, 1997.

[3] B. Beneš. An Efficient Estimation of Light in Simulation of Plant Development. In *Computer Animation and Simulation'96*, Springer Computer Science, pages 153–165. Springer–Verlag Wien New York, 1996.

[4] B. Beneš. Fast Estimation of Light in Simulation of Plant Development. In *Proceedings of WSCG'97*, volume I, pages 1–10. University of West Bohemia Press, Feb. 1997.

[5] B. Beneš. Visual Simulation of Plant Development with Respect to Influence of Light. In *Computer Animation and Simulation'97*, Springer Computer Science, pages 125–136. Springer–Verlag Wien New York, 1997.

[6] N. Chiba, K. Ohshida, K. Muroaka, and S. Nobuji. Visual Simulation of Leaf Arrangement and Autumn Colors. *The Journal of Visualization and Computer Animation*, 7:79–93, 1996.

[7] N. Chiba, S. Okawa, K. Muroaka, and M. Muira. Visual Simulation of Botanical Trees Based on Virtual Heliotropism and Dormancy Break. *The Journal of Visualization and Computer Animation*, 5:3–15, 1994.

[8] M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, 1993.

[9] A. Glassner. *Principles of Digital Image Synthesis volume I and II.* Morgan Kaufman Publishers, Inc, 1995.

[10] N. Greene. Voxel Space Automata: Modeling with Stochastic Growth Processes in Voxel Space. In *Proceedings of SIGGRAPH '89*, volume 23(4), pages 175–184, 1989.

[11] N. Greene, M. Kass, and G. Miller. Hierarchical Z-Buffer Visibility. In *Proceedings of SIGGRAPH '93*, volume I, pages 231–238, 1993.

[12] P. Hanrahan and W. Krueger. Reflection from Layered Surfaces due to Subsurface Scattering. In *Proceedings of SIGGRAPH '93*, volume 27(4), pages 165–174, 1993.

[13] N. Max. Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. In *EG WS on Rendering'96*, Springer Computer Science, pages 165–174. Springer–Verlag Wien New York, 1996.

[14] N. Max, C. Mobley, B. Keating, and E. Wu. Plane–Parallel Radiance Transport for Global Illumination in Vegetation. In *EG WS on Rendering*, Springer Computer Science, pages 239–250. Springer–Verlag Wien New York, 1997.

[15] N. Max and K. Ohsaki. Rendering Trees from Precomputed Z-Buffer Views. In *EG WS on Rendering'95*, Springer Computer Science, pages 74–81. Springer–Verlag Wien New York, 1995.

[16] N. Max and R. Troutman. Optimal Hemicube Sampling. In *EG WS on Rendering'93*, Springer Computer Science, pages 185–200. Springer–Verlag Wien New York, 1993.

[17] R. Měch and P. Prusinkiewicz. Visual Models of Plants Interacting With Their Environment. In *Proceedings of SIGGRAPH '96*, volume 30(4), pages 397–410, 1996.

[18] C. Patmore. Illumination of Dense Foliage Models. In *EG WS on Rendering'93*, Springer Computer Science, pages 63–70. Springer–Verlag Wien New York, 1993.