Rendering Trees from Precomputed Z-Buffer Views

Nelson Max

University of California, Davis, CA, 95616, USA

Keiichi Ohsaki

Hokkaido Industrial Research Institute, Sapporo, Japan

Abstract. Parallel projection z-buffer images are precomputed for a number of preset viewing directions on the unit sphere. Using the depth information, we can reconstruct a 3-D point from each image pixel. Then parallel or perspective views can be found from any other viewpoint by rotating the 3-D points from the nearest three or four precomputed views into the proper position and projecting them onto an output z-buffer. A temporary z-buffer image, constructed in the same way for a viewpoint at the light source, is used for a z-buffer shadow algorithm. The rendering algorithm is applied to broad leaf and needle leaf trees, and tested in animation. Our rendering algorithm is similar to that in Chen and Williams [1], with differences listed below.

1 Introduction

Rendering complex objects like trees from a polygonal model often takes a long time, because many polygons are required. This paper presents a method, similar to that of Chen and Williams [1], for reconstructing an image from precomputed z-buffer views. Our method is different from that in [1] in four respects. First, our precomputed images store multiple z levels at each pixel, allowing hidden objects to be reconstructed from even a single view. Second, we store encoded normal and color information, so that the shading can be done as a post process. Next, we use parallel rather than perspective projection for our precomputed views. Finally, since we are dealing with fractal-like objects with high depth complexity instead of smooth textured surfaces, the morphing techniques of [1] are not applicable, and we must reproject pixel by pixel. As in [1], we also reconstruct a z-buffer from the point of view of the light source, and then apply a z-buffer shadow algorithm during the shading step.

2 Previous work

Many people have tried reprojection of pixels, dots, or texture maps as a way of reconstructing complex objects, surfaces, or scenes. The Cloudvu system of Bishop and Austin [2] takes a collection of randomly ordered 3-D points, with associated normals and colors, and shades and projects as many as possible during the frame update time. This gives a scattered version of the object during viewpoint motion and a smoother version when the motion stops. Adelson and Hodges [3] show how reprojection of earlier frames in an animation can speed up ray-tracing. Pixels not guaranteed to be correct are ray traced from scratch, so this is not a real-time technique. On the other hand, Maciel and Shirley [4] have developed a system for guaranteed real-time performance, using “imposters” formed from one or a few texture mapped polygons, for objects, or even clusters of objects, which are too complex to render in 3-D detail. The imposter chosen may depend on the viewing direction as well as the distance, so the image may visibly jump when the choice switches.

An extreme example of an imposter is a “billboard tree”, a single polygon with color/transparency texture which is always turned to face the viewer (see Rohlf and Helman [5]). This technique has been applied to training simulators and video games, and is quite effective in still images. However, in animated motion, the 2-D nature of the texture is obvious, because the expected motion parallax between branches at different depths is absent. In addition, the textures are only appropriate for a single lighting direction.

Purely polygon-based level-of-detail transitions have also been used. (See [4] and Crow [6]). The long-term goal of this work is a level-of-detail system for complex organic hierarchical objects like trees, based on raster views instead of polygons. This paper discusses the preliminary work on the reprojection, without the hierarchies. Our reprojection scheme should work on more general scenes, but the hierarchical version will be specialized for trees, so we will discuss the current algorithm in that context.

3. Trees

Trees are particularly difficult to render because of the tremendous detail they contain. We wished to provide full 3-D animation, with changing illumination, at speeds greater than those possible with full polygon rendering. Therefore we decided to use the z-buffer reconstruction algorithm, based on a small number of precomputed images. The computational cost of this algorithm is proportional to the number of pixels in one of the precomputed images, and does not depend on the complexity of the polygonal model, making it practical to animate complex rigid objects like trees.

The trees were constructed from two growth models, one for *Abies sachalinensis* Masters, a needle leaf tree [7], [8], and one for *Magnolia obavata* Thunberg, a broad leaf tree [9]. The magnolia leaves have two different colors, one for the top surface, and one for the bottom. The correct color is chosen by testing whether the polygon normal is pointing towards or away from the viewer. The output of the tree growth model is an array of vertex coordinates, and an array of triangles, specified by their vertex and color table indices. The triangle normal is computed as a cross product of triangle edges, and the vertex indices are ordered to make this cross product point to the top surface of the magnolia leaves.

4. Reconstruction

The basic algorithm uses precomputed parellel projection z-buffer views as input. Each pixel in an input view is used to reconstruct a 3-D point, using the *x* and *y* pixel indices and the z-buffer depth in the inverse of the projection function. The 3-D point is then transformed into a new viewing position, and projected onto the output z-buffer for the new viewpoint.

Our eventual goal, the addition of standard prerendered trees to exterior architectural or landscape renderings, is different from the interior walkthroughs of Chen and Williams [1]. They use multiple image morphing to transform from each of the precomputed views to the output image. This is appropriate for the painted plane surfaces in their virtual museum application, but is less useful for finely divided objects like trees, where the depth coherence at adjacent pixels is much more limited. Therefore we transform each z-buffer pixel separately.

In addition, for their virtual museum, they use multiple perspective views from a collection of viewpoints near eye level, while we want to be able to move the viewpoint arbitrarily, and to composite multiple trees into a single image. Therefore in our implementation, we use parallel views as input. We can thus use precomputed images for a small collection of viewing directions on the unit sphere, instead of for a larger collection of viewpoints in 3-D space. We can also use the true *z* coordinate in the z-buffers for the precomputed views, while for planar polygon scan conversion in perspective projection, a non-linearly distorted $z' = -1./z$ is necessary to give the correct visible surfaces. (See [10].) In the z-buffer for the output view, the true *z* can also be used even in perspective projection, because we project single 3-D reconstructed points instead of polygons. Thus the following arithmetical steps are involved. (See [11] for the basic concepts.)

1) From the pixel coordinates (i, j) construct the window coordinates (x, y) using the inverse of the window-to-viewport transformation, and fetch z from z-buffer(i, j).

2) Multiply $(x, y, z, 1)$ by the 4 x 4 matrix which transforms the 3-D coordinates of the stored view into the coordinates $(x', y', z', 1)$ for the desired view.

3) If desired, revise the x' and y' coordinates to account for perspective. Determine the new pixel coordinates (i', j') from x' and y' using the window-to-viewport transformation. Use z' in the z-buffer comparison to determine whether to overwrite the z and shading data at pixel (i', j').

A problem with this reconstruction and reprojection algorithm is that pixels may be absent in the final output image. A pixel may be missed either 1) because it lies between transformed and reprojected pixels from a surface in the input image, or 2) because it lies on a surface which is hidden in the input image. The resampling problem 1) also occurs in Chen and Williams [1], where it is partially solved by color interpolation from adjacent pixels, but more sophisticated texture interpolation is also proposed. With the high depth complexity in our images, there is little coherence in the pixel displacements, so texture interpolation is not an alternative. We chose instead to leave undefined pixels transparent, so that the background shows through as between gaps in the leaves. Problem 1) can also be partially solved by using a higher resolution input image, but this will not help problem 2).

Problem 2) can be partially solved by precomputing more input images, so that one can be found closer to the desired viewing direction. As shown in [1], it also helps to use more than one input image at a time. In the computation for the output image, reconstructed points from several nearby input images can be combined into the same output z-buffer. With more input images, the likelihood that a visible surface will be hidden in all of them decreases. The larger number of 3-D samples also helps solve problem 1), so that lower resolution will suffice.

5. Multiple-Layered Z-Buffers

Another partial solution to problem 2) is to use multiple-layered z-buffers for the precomputed images. Each layer contains both a z value and the color and normal information necessary for the shading post-process. The layers at each pixel are sorted from front to back in z. When a polygon is scan converted for a precomputed view, the z value at each pixel is compared to the z values stored in the different layers. If the new z is closer than the farthest layer, the new data is inserted at the appropriate position, perhaps forcing the data at the farthest position to be discarded. A threshold specifies the minimum z separation between layers, so that nearby leaves or needles on the same branch can occupy the same layer, with the shading data chosen from the front-most one. Because we use these multiple z layers, our algorithm cannot take advantage of current z-buffer hardware, and is implemented in software.

6. Precomputed Directions

In our system, we use three or four precomputed input images for each output image. The preset directions are chosen along the latitude circles of a unit sphere, with the north-south axis along the up-down axis of the tree trunk. The user specifies the number n of latitude bands. The latitude positions are then at the north pole, the south pole, and on n - 1 equally spaced latitude circles separated by π / n radians. On each latitude circle, the smallest possible number of equally spaced points is chosen so that the spacing is less than π / n. This divides the sphere into triangles, as shown in figure 1 for $n = 4$. If a desired viewing direction lies inside one of the triangles meeting at the north or south poles, three precomputed images are used as input, otherwise four are used.

The vertical axis along the tree trunk is used as the “camera up vector” in the viewing transformation for the precomputed images. (See [11].) Thus the tree appears vertical in each view, allowing a rectangular image to efficiently contain it. The latitude and longitude of the desired output viewing direction are used to choose the input images, and to compute the rotation necessary to transform their reconstructed 3-D points into the final viewing coordinate system.

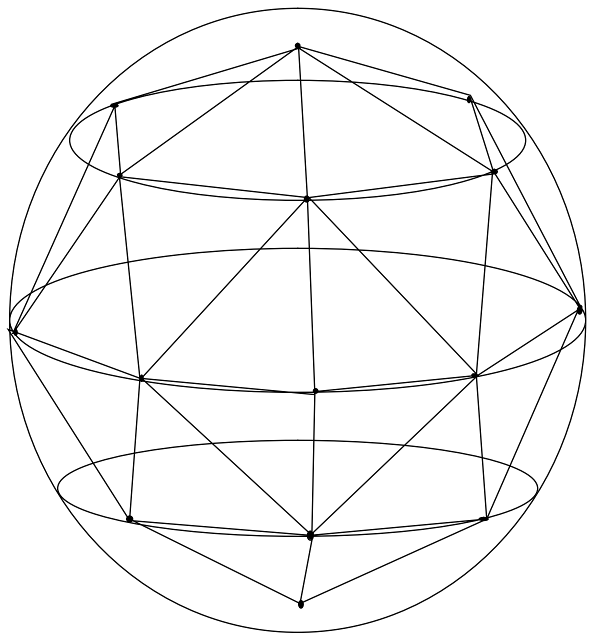

Fig. 1. Some viewing positions on latitude circles on the unit sphere.

7. Normals, Shading, and Shadows

The flat shading normals of the input polygons are coded into two bytes, one for the latitude, and one for the longitude. There is a third byte for a color table index, specifying whether the polygon belongs to the trunk, a branch, or a leaf. The normals are not transformed during the reprojection. Instead, all three bytes are copied into the final z-buffer image. During the shading post-process, tables of sines and cosines, indexed by the two normal bytes, are used to find the normal in the original coordinate system. The dot product of this normal with the untransformed lighting direction is then used to compute Lambert law diffuse shading. Phong shading could also be easily included.

Shadows can be found, as in Williams [12] and Reeves *et al*. [13], using a z-buffer from the point of view of the light source, reconstructed as described above from the same precomputed views. Currently we have only implemented an infinite light source for the sun, using parallel projection in the reconstruction, but for other applications, a finite light source could easily be implemented using perspective projection.

During the final shading step, the *x* and *y* pixel indices and the z-buffer depth value at each output pixel are again used to reconstruct a 3-D surface point. This point is then rotated into the light source viewing coordinates, and its *z* value is compared to the light source z-buffer depth value. If the 3-D point is farther from the light source by an amount larger than a preset error threshold, it is in shadow, and only ambient shading is used; otherwise diffuse shading is added.

Since the normal and color index information is available, the shadow and shading computations can be done as a post-process, only once per pixel, instead of each time pixel data is deposited into the final z-buffer, as was necessary for Williams [12].

8. Results

Figures 2, 3, and 4 show a 10 year old *Abies* tree, from three precomputed directions, at latitudes 90, 45 and 0 degrees, respectively, and at longitude 0. The value of *n* was set at 4, resulting in 23 precomputed images. Figure 5 is a perspective view, reconstructed from four of the precomputed images. Figure 6 shows the same tree at age 15 years.

Figure 7 shows a reconstructed 14 year old *Magnolia* tree in parallel projection, without shadows. Figure 8 shows the same tree in perspective with shadows. Figure 9 shows the same view, with the sun in a different position. For figures 7, 8, and 9, *n* was three, resulting in 14 precomputed images. For all these cases, the precomputed images were at 300 by 600 resolution, with two z-buffer layers, requiring 2,520,000 bytes each, and the output images were at 256 by 512 resolution. The output resolution must be somewhat less than that of the precomputed images, in order to make sure that there are no holes from pixels missed in the reconstruction. Figure 10 shows a grove of 7 rotated and translated copies of the same *Magnolia* tree, reconstructed from the same 14 two layer z-buffer views. This 500 by 350 image took 73 seconds to reconstruct.

The first segment in the videotape shows a 360 frame rotation cycle of the 15 year old *Abies* tree, rendered in parallel projection by software polygon scan conversion, at 6.5 seconds a frame at 400 by 400 resolution, on an SGI 4D/35 with a Mips 3000 processor. The scintillation comes from the non-antialiased rendering of many small needles. The second sequence shows a similar rotation cycle in perspective with shadows, reconstructed from 23 views, each with three z layers. It took 207 seconds to precompute the 23 views, and then 12.3 seconds a frame for the reconstruction at 400 by 400 resolution. Aside from the same scintillation, there are no glitches when the selection of images used for reconstruction changes at several points during the rotation.

The third segment of the videotape shows the *Magnolia* tree reconstructed from 14 views, using only one z layer. Here the leaves are broad, and the scintillation and glitches come from incomplete pixel coverage, rather than aliasing problems. The preprocessing took 68 seconds, and the reconstruction took 2.3 seconds a frame at 300 by 300 resolution. The fourth segment shows the same tree reconstructed from the same 14 views, but with 3 layers in the z buffers. The preprocessing took 82 seconds, and the reconstruction took 6.9 seconds a frame. The scintillation and glitches have disappeared.

9. Conclusions and future work

The technique of pixel by pixel reconstruction from prerendered multiple-layer z-buffers is useful for complex objects like trees, defined by many small polygons. It should similarly be useful for contour surfaces of complex objects constructed by the marching cubes algorithm [14].

For fractal objects, a hierarchy of z-buffer images of varying pixel sizes could be used, with higher level objects built by reprojecting multiple lower level objects at lower resolution. Then a final output image could be produced by combining objects chosen from different levels in the hierarchy, according to the proximity of the various objects to the viewpoint. Thus a tree could be built from standard branches, branches from subbranches, and a subbranch from twigs and leaves, all represented in multiple-layer z-buffers, built by reprojecting their children in the hierarchy. As the viewpoint moved closer, the tree could be replaced by its trunk and branches, and a branch could be replaced by its subbranches or twigs. We also hope to apply the level-of-detail transitions in Becker and Max [15] to the bump-mapped bark of Bloomenthal [16].

One of the Eurographics reviewers suggested precomputing a radiosity solution for the trees. This could possibly be done by the techniques of Patmore [17, 18]. However, we wanted to use the same tree in various different orientations to disguise the repetition when representing a forest, and to make renderings at different times of the day, so precomputing the effects of the sun's illumination would not be useful. On the other hand, both Patmore [18] and Greene and Kass [19] give efficient methods for finding the sky illumination within complex geometries like foliage, and a uniformly radiating sky hemisphere would be appropriate for any of the uses mentioned above.

10. Acknowledgments

This work was performed at the Hokkaido Industrial Research Institute where the first author was funded by the Hokkaido Overseas Guest Researchers Invitation Program. We wish to thank Isao Omura, Atsushi Okuda, and Shiichi Nagao for setting up an environment for efficient work during a short visit, and Michitaka Nami, Shinichi Maruyama, and Tsuyoshi Yamamoto for arranging the invitation. Jan Nunes helped record the videotape.

References

1. Chen, S. E., Williams, L.: View Interpolation for Image Synthesis. Siggraph '93 Conference Proceedings, 279-288 (1993).

2. Bishop, T., Austin J: Method and Apparatus for Fractional Double Buffering. U S Patent 4,910,683 (March 20, 1990). See also: TAAC-1 User's Manual, Sun Microsystems, Inc., Mountain View, CA

3. Adelson, S., Hodges, L.: Generating Exact Ray-Traced Animation Frames by Reprojection. IEEE Computer Graphics and Applications 15, 43-52 (May 1995).

4. Maciel P., Shirley, P.: Visual Navigation of Large Environments Using Textured Clusters. Proceedings of the ACM Siggraph Symposium on Interactive 3D Graphics, 95-102 (April 1995).

5. Rohlf, J., Helman, J.: IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. Siggraph ’94 Conference Proceedings, 381-394 (1994).

6. Crow, F.: A More Flexible Image Generation Environment. Siggraph ’82 Conference Proceedings, 9-18 (1982).

7. Ohsaki, K., Yamamoto, Y., Suzuki, T., Sato, H.: A Representation Method for Needle-Leaf-Trees Based on Growth Models (Japanese with English abstract). Graphics and CAD 52-4, 19-26 (August 16, 1991).

8. Suzuki, T., Ohsaki, K., Saito, H., Yamamoto,Y.: A Representation Method for Todo-Fir Shapes using Computer Graphics (Japanese). Journal of the Japanese Forestry Society 74, 504-508 (November 1992).

9. Ohsaki, K., Suzuki, T.: A Growth Model of Botanical Trees Having Abilities to Interact with the Light Environment (Japanese with English abstract). Graphics and CAD 65-6, 37-44 (October 22, 1993).

10. Newmann, W., Sproull, S.: Principles of Interactive Computer Graphics, first edition. New York: McGraw Hill (1979).

11. Foley, J., van Dam, A., Feiner, S., Hughes, J.: Computer Graphics, Principles and Practice, 2nd Edition. Reading Massachusetts: Addison Wesley (1992)

12. Williams, L.: Casting Curved Shadows on Curved Surfaces. Siggraph ’78 Conference Proceedings, 270-274 (1978).

13. Reeves, W., Salesin, D., Cook, R.: Rendering Antialiased Shadows with Depth Maps. Siggraph ’87 Conference Proceedings, 283-291 (1987).

14. Lorensen W., Cline, H.: Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. Siggraph ’87 Conference Proceedings, 163-169 (1987).

15. Becker B., Max, N.: Smooth Transitions between Bump Rendering Algorithms. Siggraph ’93 Conference Proceedings, 183-190 (1993).

16. Bloomenthal, J.: Modeling the Mighty Maple. Siggraph ’85 Conference Proceedings, 305 - 311 (1985).

17. Patmore, C.: Simulated Multiple Scattering for Cloud Rendering. In: Mudur S., Pattniak, S. (eds.): Graphics, Design, and Visualization: Proceedings of the International Conference in Computer Graphics, ICC93. Elsevier Science Publishers 1993 (pp. 29-40).

18. Patmore, C.: Illumination in Dense Foliage Models. Proceedings of the Fourth Eurographics Workshop on Rendering, 63-71 (1993).

19. Greene N., Kass, M.: Approximating Visibility with Environment Maps. Apple Computer Technical Report #41 (November 1994).

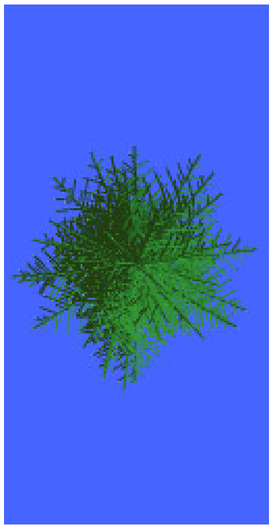

Fig. 2. Top view of *Abies*.

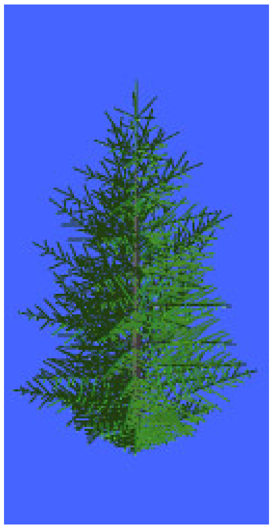

Fig. 3. Slant view of *Abies*.

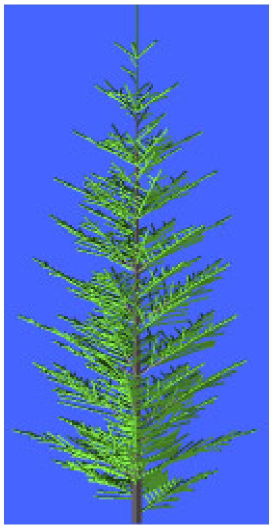

Fig. 4. Side view of *Abies*.

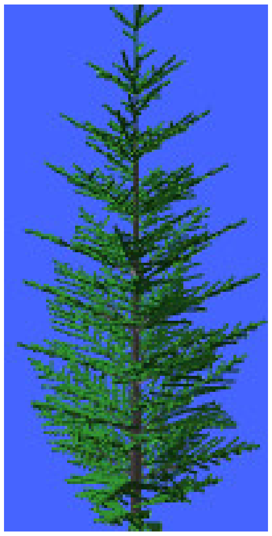

Fig. 5. Reconstructed *Abies*.

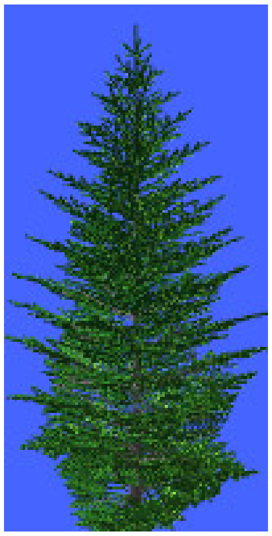

Fig. 6. Reconstructed *Abies*.

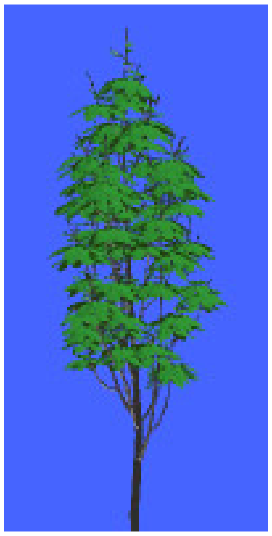

Fig. 7. Reconstr. *Magnolia*.

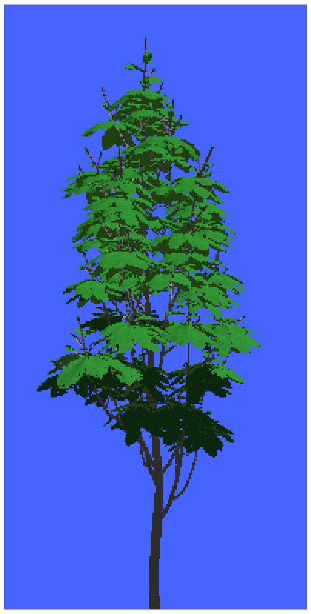
Fig. 8. *Magnolia* with shadows.

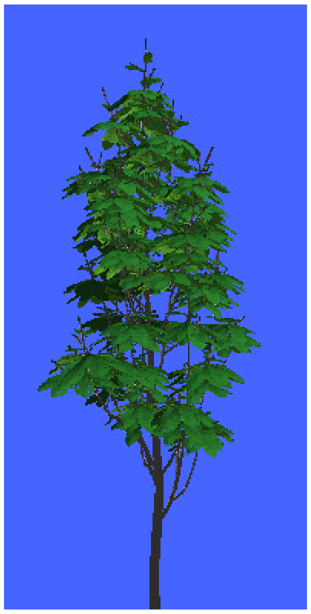
Fig. 9. *Magnolia* with shadows.

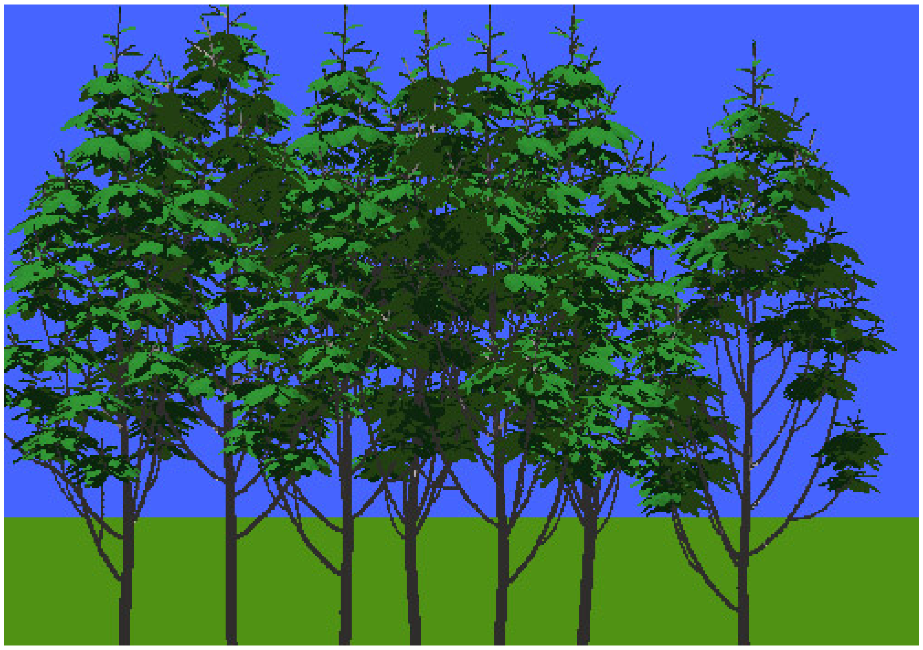
Fig. 10. A grove of seven identical rotated and translated *Magnolia* trees with shadows.