ELSEVIER

Landscape and Urban Planning 64 (2003) 105–113

LANDSCAPE AND URBAN PLANNING

This article is also available online at:
www.elsevier.com/locate/landurbplan

A framework for geometry generation and rendering of plants with applications in landscape architecture

Oliver Deussen[*]

Faculty of Computer Science, Dresden University of Technology, Mommsenstr. 13, D-1062 Dresden, Germany

Abstract

Methods for creating virtual vegetation using techniques of computer graphics are presented. First, a new generation method for plant geometries is shown. The resulting models are combined to form a realistically looking vegetation. In the final step, the models are converted into a special representation that is used to create synthetic illustrations of plants. The illustrations can be drawn in various drawing styles, animation of the illustrations is possible. The method can be used to generate interactive walkthroughs in a sketched virtual world.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Computer graphics; Visualization; Plant models; Non-realistic computer graphics; Illustration

1. Introduction

After three decades of research, algorithms for synthetic image generation have reached a limit where the visual difference between real and synthetic images is vanishing. Applications such as visualization, film, advertising and many others take advantage of the results. The market for computer graphics products grows steadily.

As a surprise for non-experts, since a few years researchers return to the simple line drawings that have been produced in the early years of computer graphics. A variety of techniques have been proposed to sketch and non-photorealistically render objects. Research in this area was motivated by the realization that drawings are able to convey visual information in a different way than photo-realistic images do (Strothotte and Strothotte, 1997). This is one of the reasons why a large percentage of images in many books are drawings (cf. Strothotte et al., 1994). In contrast to the early line drawings of computer graphics, today's algorithms try to generate images similar to artistic drawings, in some cases even new drawing styles have been developed.

While the proposed methods allow creating line drawings of many objects and in many different styles, the illustration of plants has so far been neglected. This is surprising because drawings of these objects are needed in areas such as landscape architecture and planning. In both cases early designs are preferentially visualized as abstract line drawings (Schumann et al., 1996).

The present article consists of three parts, each describing a technique that is important for our goal of creating synthetic plant illustrations. The first part describes a new method for generating plant geometries which has been developed by Bernd Lintermann (ZKM Center for Arts and Media Technology Karlsruhe, Germany) in collaboration with the author. Various aspects of the method are described by Lintermann and Deussen (1999) as well as Deussen and Lintermann (2001).

[*] Tel.: +49-351-463-38212; fax: +49-351-463-38396.
E-mail address: deussen@inf.tu-dresden.de (O. Deussen).
URL: http://www.inf.tu-dresden.de/cgm

In the second part, it is demonstrated how to create synthetic vegetation using the plant models. This was done in collaboration with Bernd Lintermann, Matt Pharr and Pat Hanrahan (Stanford University) as well as Radomir Mech and Premyslaw Prusinkiewicz (University of Calgary). Computer graphics details of this method can be found in Deussen et al., 1998. In the third part, the plant models are used to generate synthetic illustrations. This has also been published to the graphics community, please refer to Deussen and Strothotte (2000) for a complete description.

In the present article, computational details are omitted, instead the applicability of the methods to landscape architecture is discussed. Therefore, the first two parts are roughly sketched and some related approaches are given, whereas the illustration method is described more in detail. We also try to estimate how much computational effort is needed to visualize realistic and non-realistic vegetation and in which places our techniques might be useful at all.

2. Realistic plant models

In the beginning, work on plant generation was biologically motivated. Pioneering work was done by Aristid Lindenmayer and later by Premyslaw Prusinkiewicz who described the structure of plants by so called Lindenmayer-Systems operating on a set of rules (Prusinkiewicz and Lindenmayer, 1990). In L-Systems, a plant is specified in terms of local growth rules. This is intuitive for biologists, but from the modeling point of view, one is also very interested in dealing directly with the various aspects of a plant.

To provide more intuitive parameters and to ease the control over the models, procedural plant models have been evolved that can be customized. Oppenheimer (1986) presented a fractal tree model where in each branch, parameters like branching angle, size ratio between main stem and branch, or the number of branches per stem can be specified. De Reffye et al. (1988) developed a procedural model based on birth and death of growing buds which allows to control the generation of plants by some parameters. AMAP, a commercial library of plants—particularly trees—and generation procedures works on the basis of this idea. The user edits the parameters of a plant type and runs the simulation to produce the desired geometry.

Another procedural model of trees was proposed by Holton (1994) who assigned a strand to any path from the root to the leaves of a tree. The number of strands in a fork determines the fork angle, length and taper of branches. After specifying additional parameters for representing different tree types, the models are rendered which consumes some time and makes interaction difficult.

While the authors listed above concentrated on finding efficient descriptions which fit into the botanical principles, Weber and Penn (1995) focused on the goal of generating a visually favorable geometry without adhering strictly to botanical laws. Special emphasis was put on modeling the overall shape of a tree. This was done by specifying the shape geometrically and restricting the model to grow within the bounds of the shape. A set of textually edited parameters described the geometry for each branching level of the tree.

A highly interactive system for generating trees is Onyxtree by Onyx Computing (Onyx Computing Inc. Onyx tree professional. http://www.onyxtree.com). A tree is composed of several branching levels which can be customized graphically. Operations like cutting a stem are possible, the user gets immediate feedback about the created geometry. This system makes it easy to generate and modify the models, but is restricted to trees.

2.1. Rule-based object generation

In our approach, we combined the rule-based methods with procedural elements. A plant is described by a structure graph that consists of different components each describing a geometrical part of the plant or a generation algorithm. A set of eleven different components allows us to create all plant species.

While the 'Simple' component (cf. Fig. 1), which is one of the five entities that form the group of geometry producing components, offers only a basic parameter set, the others are used for more complex tasks: 'Revo' for creating surfaces of revolution, 'Horn' for twigs and stems, 'Leaf' for leaves and 'Tree' for modeling trees.

The 'Tree' component offers also multiplying functionality. Every child component in the structure tree is multiplied according to the multiplying procedure implemented in the component. This is used to arrange twigs along a stem or other objects. Other multipliers

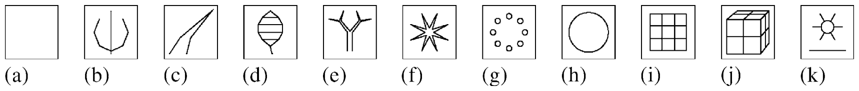

Fig. 1. Components of the proposed modelling method: (a) Simple; (b) Revo; (c) Horn; (d) Leaf; (e) Tree; (f) Hydra; (g) Wreath; (h) Phiball; (i) FFD; (j) Hyperpatch; (k) World.

are 'Wreath' which multiplies child components like candles on a Christmas tree, 'Hydra' which multiplies them in a star shaped arrangement and 'Phiball' that uses a distribution according to the golden section (Vogels formula, see Vogel, 1979) which is found in many phyllotactic patterns.

Three components are used for global modeling and form the last group of components: 'FFD' and 'Hyperpatch' enable to apply free form deformations to the models and 'World' allows to redefine growing tendencies of the plants like the gravitropism, where a plant directs its branches to the ground (by default in negative z-direction) or phototropism, where the plant organs are directed according to the light (by default in positive z-direction).

Each component offers a set of parameters which allow to individualize the generated data. Parameters can be geometrical properties like size and orientation or structural properties such as the number of multiplied child components. In our implementation, the parameters of each component can be altered by a specialized graphical user interface.

In our method, the structure graph represents the rule system. The geometric generation is done by performing generation rules. If the father component of a link is forced to produce its geometry it forces all children to do the same. In the case the father component is a multiplying component, several copies of subsequent components are created, each with an individual orientation and parameter set. This mechanism—for a complete description see Lintermann and Deussen (1999)—allows us to arrange leaves, petals, seeds and other elements of plants.

As a first example we show how to model a small sunflower. A natural leaf of a sunflower is scanned and applied as a texture to the surface of a 'Leaf'

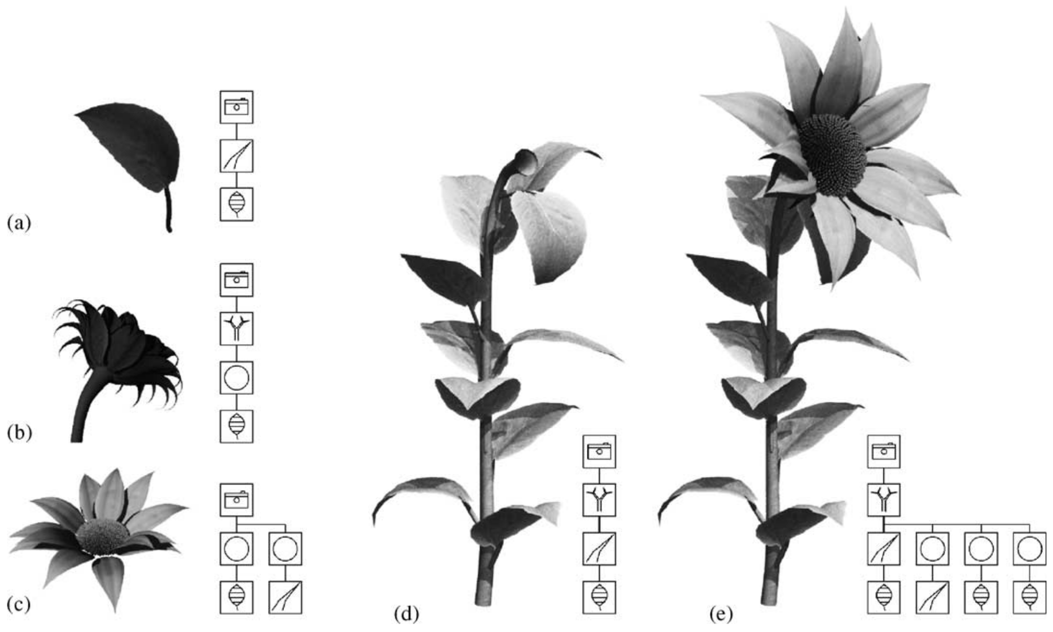

Fig. 2. Modelling a sunflower: (a) representation of a leaf; (b–d) petals and seeds are distributed by golden section placement; (e) the leaves are combined with the stick; (f) the complete flower is described using only 10 components.

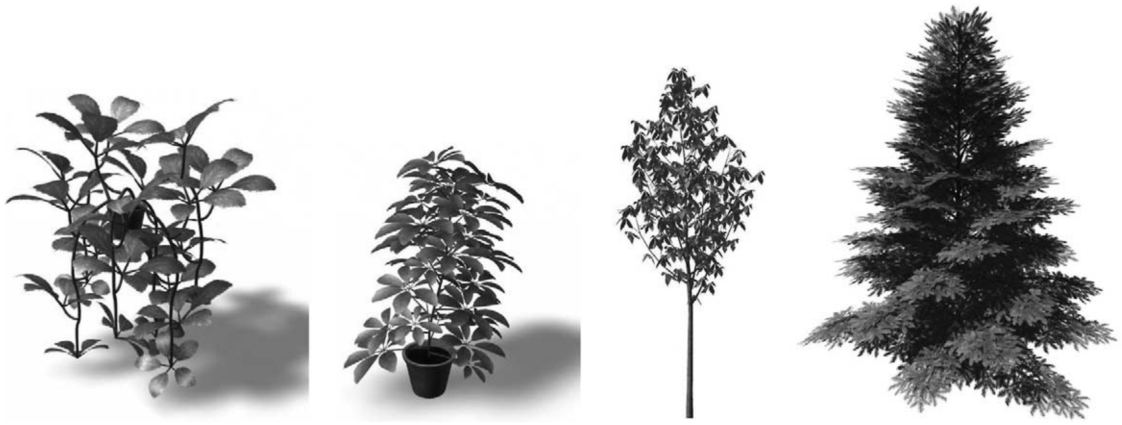
Fig. 3. Several plants modeled using the proposed method.

component (cf. Fig. 2(a)). A tiny stalk is connected to the leaf. So far the structure graph consists of three components, the ‘Camera’ component (the general root component with a set of viewing parameters), a ‘Horn’ component for the stalk, and a ‘Leaf’ component. By editing splines the curvature and scale of ‘Horn’ and ‘Leaf’ components is chosen appropriately to achieve the typical outline of a small leaf.

In the next step the leaves are iterated as branches of a ‘Tree’ component describing the stalk of the plant (Fig. 2(d)). The top of the stalk is opened to form the head for the flower. This is shown in Fig. 2(b), where some leaves are iterated by a ‘Phiball’ component and placed on the top of the stalk.

Similarly, the blossom of the sunflower is constructed by two ‘Phiball’ components, one for arranging the leaves and the other for distributing the seeds (cf. Fig. 2(c)). Finally, everything is connected to the full structure graph which is shown in Fig. 2(e).

Other objects can be generated in a similar way; in Fig. 3 some of them are shown. Whereas small bushes and flowers need individual structure graphs, trees are usually modeled using a sequence of ‘Tree’ components. In this case the main work is to find the right modeling parameters for each component, which is still a time consuming task.

To avoid this effort for non-expert users, a commercial version of the modeling program called xfrog (see http://www.greenworks.de) offers several plant libraries with hundreds of plants that cover a selection of the main species in Europe and North America. So far these plants are modeled from the computer graphics point of view, in the future we will generate special libraries for purposes of landscape architecture and botany with all important plants of Northern Europe and North America.

3. Synthetic vegetation

In its general form, creating complex synthetic vegetation is simple: a number of plants is arranged to form the plant coverage on the top of a given height field (Fig. 4). But a closer look to the problem offers a number of tricky problems which arise in this context, some of them were solved in Deussen et al. (1998), others are still open. The most important problems are

- Geometric complexity:
  For a realistic representation of a single tree geometry, several tens of thousands of triangles are needed. Triangles are the most common geometric element to represent botanic surfaces in computer graphics because of their simplicity. A square meter of a meadow needs tens of thousands of triangles. If a whole landscape is to be generated, billions of triangles have to be manipulated.
  The problem can be reduced by a number of methods. A set of similar plants, i.e. a population of a single species, can be approximated by one master geometry and a set of instances.

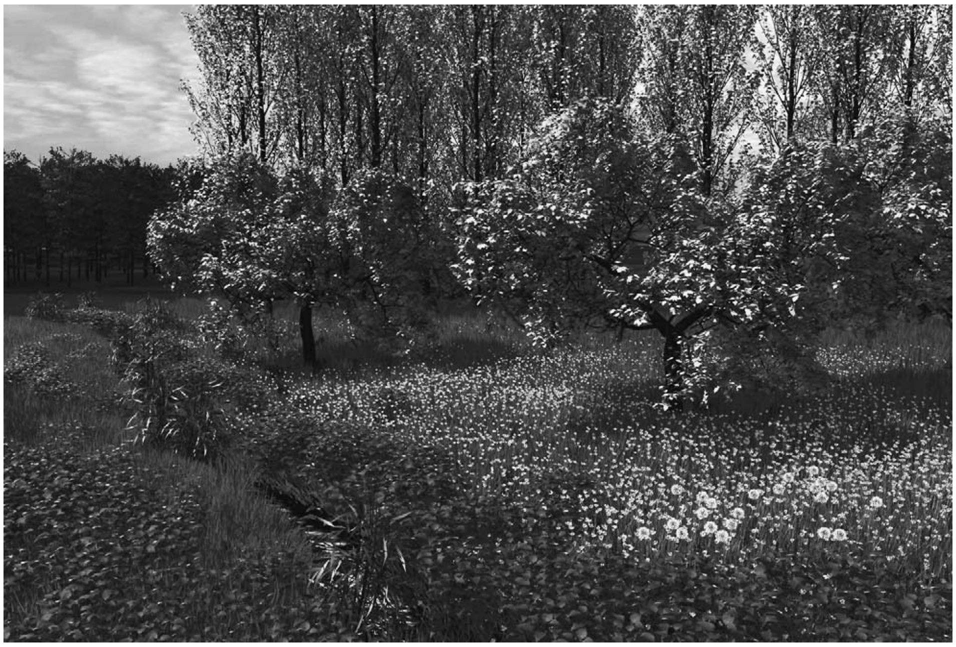

Fig. 4. A synthetic landscape, generated from about 15 species and 20 Mio triangles, courtesy of Bernd Lintermann.

These instances are represented by their location and a pointer to the master geometry. This helps to reduce the geometric data. Since in nature each plant differs from all others, this kind of instancing is only a visual approximation. Nevertheless, our empirical results show that a set of 10–20 master geometries is sufficient to represent a whole population without obvious visual artefacts.

Another reduction method is to change the geometric resolution of the plant in correspondence to its distance from the viewer. A plant at the back is represented by a few triangles, a plant close to the viewer by its full complexity (level-of-detail concept).

- Distribution properties:

  Each plant and each plant community has a distinct distribution pattern for its entities which has to be specified for a computer image. Such patterns can be simulated using mathematical methods. Another method is to specify them directly by using predefined patterns.

- Interaction of plants:

  The shape of a tree near a wall or a tree in a forest differs from the shape of a solitary tree. This has to taken into account while generating a plant population. Therefore, a collection of models representing a single species has to be defined. In addition, several ages and also the shapes for the different seasons have to be created for each plant.

The geometrical complexity has an effect on the maximum frame rate in interactive applications. Today's graphics processing units (GPUs) are able to generate tens of Millions textured polygons per second. To interactively show a complex botanical scenery much more is needed. The problem can be reduced by transferring only the part of the geometric data to the GPU which is potentially visible. This excludes data behind the viewer or data far away. The corresponding algorithms are called visibility culling methods, specialized versions for plant scenes have to be developed here.

In the last months, dramatic improvements have been made that allow to visualize complex vegetation with interactive frame rates on modern PCs (cf. Deussen et al. (2002)). The user is now able to walk through a complex virtual landscape in real time.

4. Non-realistic images of plants

As mentioned in the introduction, the realistic plant models can be used to create non-realistic plant images. Each surface-oriented plant description can be used for that purpose. However, in the context of this article, the xfrog modeling system is used.

Among the various styles used by artists to render trees—for a large set of examples see Evans (1996)—one can distinguish between flat styles that solely represent the shape of a tree and others that also approximate natural light interaction (cf. Lohan, 1993). The tree skeleton is usually drawn up to the second branching level, primarily by silhouette lines and crosshatching on the stem surface. The shape of the foliage is either represented by an abstract outline or by a collection of many small objects which do not necessarily resemble natural leaves but instead represent the characteristics of the foliage. In addition, the outline is sometimes drawn by many small line segments or just a few strokes.

The visual appearance of the foliage can be divided into three areas. The top of the tree is usually in the direct light and is therefore visualized by only some details and its outline. In the half shadow, more details are drawn to achieve an appropriate grey level. In this area the outline of the leaves is often drawn in detail. The third area is the shaded part. The three areas are generally not found in a single illustration, often only the half shadow and the full shadow region is drawn. Sometimes the complete foliage is represented uniformly.

Artists use different methods to generate shadows on the foliage: in many styles more details are drawn and thick lines are used, sometimes with whole areas being drawn in black. Other styles add crosshatching to the foliage.

The first step to create a tree illustration is to generate a tree with a conventional tree modeling system. The final model—some of them are shown in Fig. 5—is pre-processed and two files are created. In the first file, the geometry of the tree skeleton is stored. Like artists, we only draw the trunk and branches up to the second-order in most of our illustrations with higher order branches being removed.

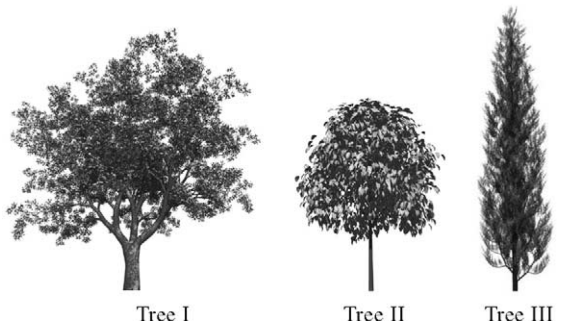

Fig. 5. Photorealistically rendered images of the synthetic sample trees: Tree I: complex tree; Tree II: young lime tree; Tree III: conifer.

The second file stores the leaves as a set of point objects, each with a position and a normal vector. The normal vectors are obtained by using the normal vector of the original leaves. If too much data is generated for all the leaves—Tree I in Fig. 5 has about 183,000 leaves—we reduce them in the modeling system by reducing the number of leaves at each twig. If this is still too much we position the particles at the branching positions of the highest-order twigs. In the case of Tree I we end up with 8800 particles.

The illustrations are generated as follows: the trunk and branches are drawn by applying techniques already known from research in non-photorealistic rendering (Fig. 6). The foliage is rendered by representing each leaf by a drawing primitive—a disc or

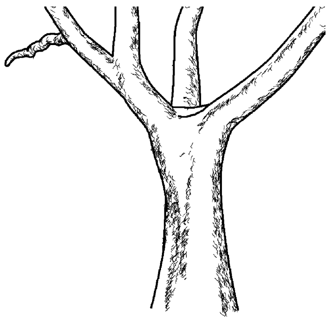
Fig. 6. The trunk and main branches of Tree I are extracted and rendered by silhouette lines and cross hatching.

arbitrary polygon facing the viewer—and by applying the depth difference algorithm to determine which part of the primitive outlines are to be drawn. Shadows can be applied at this stage, vegetation on the ground can also be added and is processed the same way. The resulting drawings are then composed to constitute the final image.

4.1. *Drawing the tree skeleton*

The tree skeleton is an assembly of generalized cylinders each representing a branch. The surface is more or less smooth, which allows us to apply an analytical silhouette algorithm. In addition, the skeleton is shaded with cross hatches. The algorithm places short strokes at positions where the grey scale value is above a given threshold. The area of the stroke is determined and the corresponding error value is subtracted from the neighboring pixel values. The direction of the strokes is either at random or affected by the normal vector of the stem geometry. A similar technique for directing strokes was used in Markosian et al. (1997).

4.2. *Drawing the foliage*

The foliage of a tree differs by its very nature from all smooth surfaces and therefore must be handled separately. Several thousand individual surfaces must be combined visually to a shape or a set of strokes. In our first experiments, we placed special textures on the leaves of our realistic tree models that looked like strokes. This is a fast and simple method, but the generated images never appeared like drawings.

The observation that artists do not draw leaves correctly but try to represent their visual appearance led us to the use of abstract drawing primitives. Each leaf is represented by the outline of such a primitive, whereas its position is determined by the leaf's position in the original tree. The size is controlled by the user. A very simple drawing primitive is a view-facing disc. While other abstract drawing primitives are given below, we first describe the second ingredient of our approach, the depth difference algorithm.

4.3. *Depth differences*

Depth differences are used to determine what part of each drawing primitive is to be drawn to constitute the foliage. Saito and Takahashi (1990), two of the early researchers in non-photorealistic rendering, used the depth-buffer to determine the outline of objects which then were used to enhance photo-realistic images. This buffer is provided by many graphics systems, it is computed automatically during image generation. For each pixel of the screen it contains the distance of the displayed geometry to the viewer.

While first and second order depth derivatives helped Saito and Takahashi to find important lines on smooth surfaces, zero-order derivatives are helpful for determine important lines in collections of isolated surfaces like assemblies of drawing primitives: The outline of a primitive is drawn if the maximal depth difference of the surface to the neighboring surfaces is above a given threshold.

Instead of computing the differences analytically—which is computationally expensive in the case of complex tree models—we use the depth buffer for this purpose. For each pixel the depth difference is computed by comparing its depth value with all neighbor values. The maximal positive difference for each pixel is taken. This value indicates how far the pixel is in front of its neighboring pixels. It is stored in a separate buffer.

Fig. 7 shows two sketches of Tree I. In Fig. 7(a) small discs are used and the threshold is low. This results in high detail and a good approximation of the real model. A more abstract rendering is achieved if disc size and threshold are enlarged (Fig. 7(b)).

4.4. *Abstract drawing primitives*

Apart from discs, a number of drawing primitives can be used to represent the leaves. In Fig. 8(a) and (b),

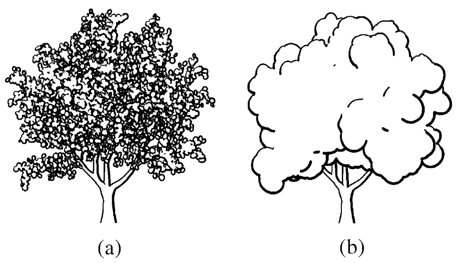

Fig. 7. Tree I rendered with varying primitive size—in this case discs are used—and depth difference threshold: (a) size = 0.15, threshold = 1000; (b) size = 0.7, threshold = 2000.

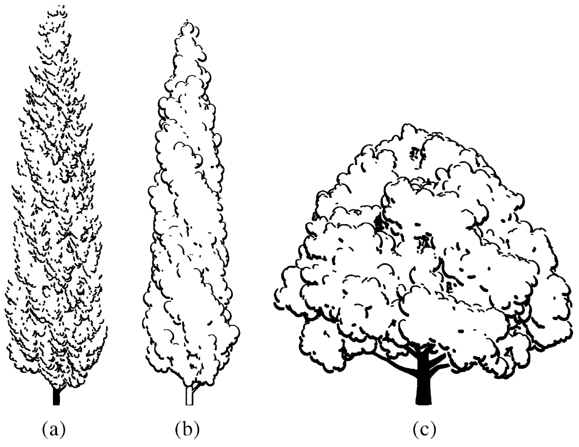

Fig. 8. (a, b) Thuja, rendered with different drawing styles; (c) Maple tree, constructed from 16,200 drawing primitives.

Tree III is drawn using view-facing elliptic primitives of random orientation. After determining which part of each primitive has to be drawn, a small deformation was applied to each outline. This helps us to achieve a more sketched drawing style.

In Fig. 8(b), all visible outlines are drawn, and a small threshold is used. The drawing of Fig. 8(a) was created using a slight modification of the algorithm: only the lower part of each ellipse is drawn when visible, the threshold having a very small value. Rendering is performed in 10 s on our PC with Nvidia Geforce2 GPU, the conifer consists of 13,200 particles.

The maple tree of Fig. 8(c) consists of 16,200 particles which is far below the original number of 200,000 leaves. The parameterization of Fig. 8(a) and a larger threshold was used. The tree in Fig. 9 consists of 90,000 particles, very small ellipses were used, shadow is added as black regions. The ground is represented by 23,000 elliptic primitives of larger size. Only the shadow is drawn, no primitive outlines are used. In this case rendering is performed in about 1 min.

In the interactive version of the proposed algorithm it is possible to render three trees consisting of 20,000 primitives each and 25,000 ground particles with three frames per second on a SGI Graphics Workstation at lower image quality. We hope to improve this in the future.

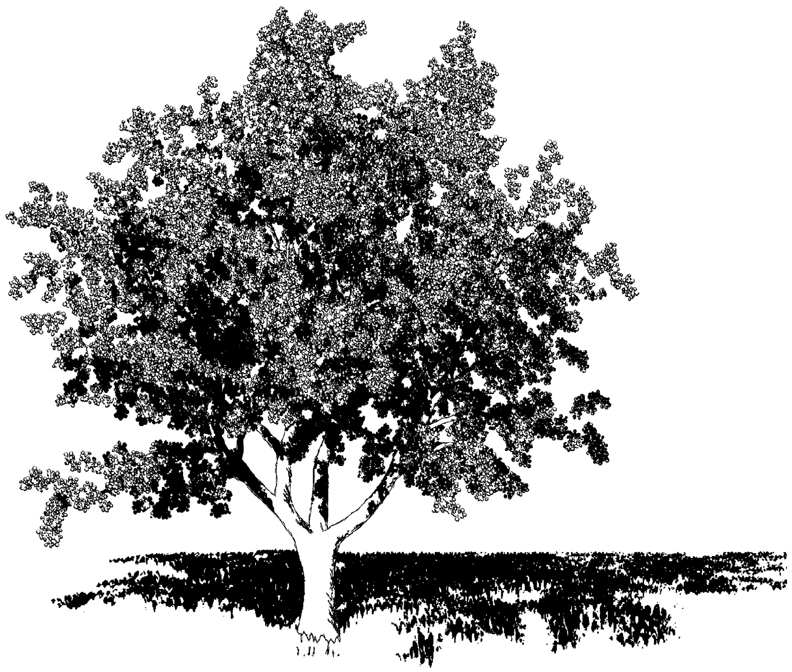

Fig. 9. A sketched virtual scene.

5. Conclusion

In this paper, a framework for geometry generation and rendering of plants with applications in computer graphics and landscape architecture was given. Three parts showed the generation of single plants, their combination to virtual vegetation and a new rendering technique to visualize vegetation using non-realistic computer graphics.

While realistic rendering methods have applications in areas such as film industry, advertisement, biological simulation and visualization as well as education, the non-realistic algorithms may be used in fields such as architecture, landscaping or cartooning.

In these areas one is more interested in visualizing a model by abstract representations than to simulate a specific landscape by realistic images. Apart from this cognitive reason, a non-realistic description offers also a method to reduce geometric detail tremendously. A tree can be sketched by some strokes instead of using tens of thousands of triangles for a realistic representation.

Unfortunately, the generation of the strokes so far is quite time consuming. Here, more efficient methods have to be found. The new capabilities of modern graphics cards, i.e. the register combiners in Nvidia GPUs, may allow to compute silhouettes and depth differences in real time.

On the modeling side, efficient generation algorithms for plant distributions have to be found. Currently, the modeling effort that is needed to generate a virtual landscape is far to high for a commercial use. In this form, a visualization is possible only for very large and expensive projects. In the future we plan to couple our system with a GIS in order to use already existing data bases for image generation. Such an integrated system might be a powerful tool for landscape planners, at least for companies specialized in landscape visualization.

References

De Reffye, P., Edelin, C., Francon, J., Jaeger, M., Puech, C., 1988. Plant models faithful to botanical structure and development. In: SIGGRAPH'88 Proceedings, ACM SIGGRAPH. pp. 151–158.

Deussen, O., Lintermann, B., 2001. Computerpflanzen. Scientific American, German issue (Spektrum der Wissenschaft), January 2001.

Deussen, O., Strothotte, T., 2000. Computer-generated pen-and-ink illustration of trees. In: SIGGRAPH'2000 Conference Proceedings, ACM SIGGRAPH. pp. 13–18.

Deussen, O., Hanrahan, P., Pharr, M., Lintermann, B., Mech, R., Prusinkiewicz, P., 1998. Realistic modelling and rendering of plant ecosystems. In: SIGGRAPH'98 Conference Proceedings, ACM SIGGRAPH. pp. 275–286.

Deussen, O., Colditz, C., Stamminger, M., 2002. Efficient rendering of complex ecosystems using points and lines. In: Proceedings of IEEE Visualization 2002.

Evans, L., 1996. The New Complete Illustration Guide: The Ultimate Trace File for Architects, Designers, Artists, and Students. Van Nostrand Reinhold, New York.

Holton, M., 1994. Strands, gravity and botanical tree imagery. Computer Graphics Forum 13 (1), 57–67 (Eurographics Association).

Lintermann, B., Deussen, O., 1999. Interactive modelling of plants. IEEE Computer Graphics and Applications 19 (1), 56–65.

Lohan, F., 1993. The Drawing Handbook. Contemporary Books, Chicago.

Markosian, L., Kowalski, M.A., Trychin, S.J., Bourdev, L.D., Goldstein, D., Hughes, J.F., 1997. Real-time non-photorealistic rendering. In: SIGGRAPH'97 Conference Proceedings, ACM SIGGRAPH. pp. 415–420.

Oppenheimer, P.E., 1986. Real-time design and animation of fractal plants and trees. In: SIGGRAPH'86 Conference Proceedings, ACM SIGGRAPH. pp. 55–64.

Prusinkiewicz, P., Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer, New York.

Saito, T., Takahashi, T., 1990. Comprehensive rendering of 3D shapes. In: SIGGRAPH'90 Conference Proceedings, ACM SIGGRAPH. pp. 197–206.

Schumann, J., Strothotte, T., Raab, A., Laser, S., 1996. Assessing the effect of non-photorealistic images in computer-aided design. In: Proceedings SIGCHI'96 Conference ACM Human Factors in Computing Systems. ACM Press, pp. 35–41, 13–15.

Strothotte, C., Strothotte, T., 1997. Seeing Between the Pixels: Pictures in Interactive Systems. Springer, Berlin, Heidelberg, New York.

Strothotte, T., Preim, B., Raab, A., Schumann, J., Forsey, D.R., 1994. How to render frames and influence people. Computer Graphics Forum 13 (3), 455–466 (Eurographics Association).

Vogel, H., 1979. A better way to construct the sunflower head. Mathematical Biosciences 44, 179–189.

Weber, J., Penn, J., 1995. Creation and rendering of realistic trees. In: SIGGRAPH'95 Conference Proceedings, ACM SIGGRAPH. pp. 119–128.

Oliver Deussen is professor of Computer Graphics at Dresden University of Technology, Germany. He graduated at the University of Karlsruhe and was assistant professor at the University of Magdeburg. His research interests include the simulation, modelling and visualizing of complex biological objects, non-photorealistic rendering, human–computer interaction and visualization.