Efficient modelling and rendering of synthetic landscapes

Oliver Deussen[1], Carsten Colditz[1], Liviu Cocunu[2], Hans-Christian Hege[2]

Abstract

Generating synthetic images of rich landscapes is still a challenging task in various respects. Firstly, efficient modelling methods for single plants have to be developed, as so far used approaches do not allow designing plants efficiently. Our hybrid method enables the user to generate even complex plants in short time; this is described in the first paragraph. Secondly, plant models must be combined in order to form vegetation. Spatial-temporal simulation models based on individual plants help here. The vast amount of geometry involved here must be reduced to allow interactive rendering. A specially designed level-of-detail algorithm represents the plant geometry in dependency to its size on screen and allows frame rates of several frames per second even for complex landscapes.

1. Synthetic plant models and landscapes

Besides the fields of landscaping and landscape architecture synthetic plants play an important role in many applications. In botany, such models can be used for modelling plant growth and genetic expression, for evaluating mathematical models applied to backscattering measurements and for visualizing spatio-temporal processes. In the movie industry they are used for modelling special effects or special plants that cannot be found in nature. In computer games they are used for synthetic backgrounds. The different applications need different models, but generally, the need for very detailed visual plant descriptions grows as fast as computers are able to handle those complex models.

2. Modelling of plant models

In recent years several important advances have been made to model such realistic plants models efficiently. Two major mechanisms are described in literature, on the one hand procedural methods are used, parameterized algorithms that generate plant geometry for one or a small set of plants. Various algorithms were proposed [1,2,3,4], but only the AMAP system, now developed by Bionatics is still developed.

One the other hand, rule-based systems was introduced that describe the plant geometry by a set of rules that are applied to create a complex model from a simple initial state. The most prominent approach is known as L-Systems, developed by Lindenmayer and Prusinkiewicz [5]. Here, a textual rule basis is used for describing the plant. The rules do string rewriting in a given text. Starting from an initial word, the sequence grows until a given number of rewritings was performed. In a second step, the final string is interpreted graphically to produce the geometry. Over the years these systems were extended by mechanisms for interaction of plants with their environment [6] and by introducing positional information [7]. The latter

[1] University of Konstanz, Germany
[2] Zuse Institute Berlin, Germany

extension allows the user to edit his models quite efficiently. Spline functions can be used to vary important parameters of plants along their growth axes. Instead of programming L-System rules manually, the user is able to change the shape of plants by parameters based on these functions.

A similar editing technique was proposed by Bernd Lintermann and one of us in combination with the plant modelling tool xfrog [8]. Here, a mixture of procedural and rule-based methods is used to model a plant: procedures compute the geometry of plant parts and are combined by a simple rule based mechanism. The procedures are represented by components. A set of components is connected by the user to describe the structure of the plant. The algorithms are controlled by graphical user interfaces on the basis of spline functions. For a complete description see [9], a sample plant is shown in Fig. 1. For the xfrog modelling system a plant library of several hundreds of different plants exists that can be maintained by the user [10]. Also for L-Systems many models have been created.

Figure 1: Steps in creating a sunflower: real leaf textures are scanned and projected on the surface of a leaf element represented by a leaf component. Arrangement of plant parts is performed by multiplying components as shown in the blossom. Putting all together a set of ten components is able to represent the whole plant.

3. Modelling synthetic landscapes

A landscape is formed by human artefacts, a biotic items and a set of plants. For modelling plant societies, growth patterns and statistic values of the plant populations are obtained from nature and are represented by the computer. The main problem is simply the number of plants and the huge amount of geometry which is necessary to model vegetation. A single square meter of a meadow contains thousands of plants and needs millions of triangles to represent it realistically. Millions of plants must be combined to represent a square mile. In [11] an open system is described that is able to model and render complex plant scenes with hundreds of millions of triangles. The raytracing algorithms used for that purpose render an image in about one hour on a sixteen processor SGI computer – far away from interactive rates.

Figure 2: A synthetic landscape modelled by Bernd Lintermann using xfrog and the open system described in [11].

In [12] the rendering performance for the same scenes was improved to several frames per second – now using graphics hardware and a specially designed level-of-detail (LOD) algorithm. The plant is represented by geometry, if the virtual viewer is close and by a set of points and lines if far away. Points are used for compact objects such as leaves, lines for long and thin objects such as branches. Both sets are obtained by randomly sampling the plants surface. For a given distance to the plant or a size of the projected plant model, respectively, a specific number of points and lines are displayed. This number decreases such that the number of points and lines always represents the plant model faithfully.

The problem with most of these LOD schemes is the computing time necessary for each frame. In our case we solved the problem by randomly reordering the point and line sets for each plant and storing them in graphics memory. This reordering is done in a way that each head of a set always results in a valid approximation of the plant geometry. The amount of points and lines shown for each plant is now computed by a simple formula and results in a number which tells the graphics card, how much of the points and lines for a plant model have to be displayed. Doing so, no transfer of data to the graphics card is necessary during rendering, all the approximation data is stored on the card in a pre-processing step.

Figure 3 shows some examples of plant approximations: In the upper line, a plant model is represented by geometry and by several sets of points. In the middle, a set of plants is shown; the plants in the background are represented by points. In the last line, a line approximation of a plant with thin leaves is given.

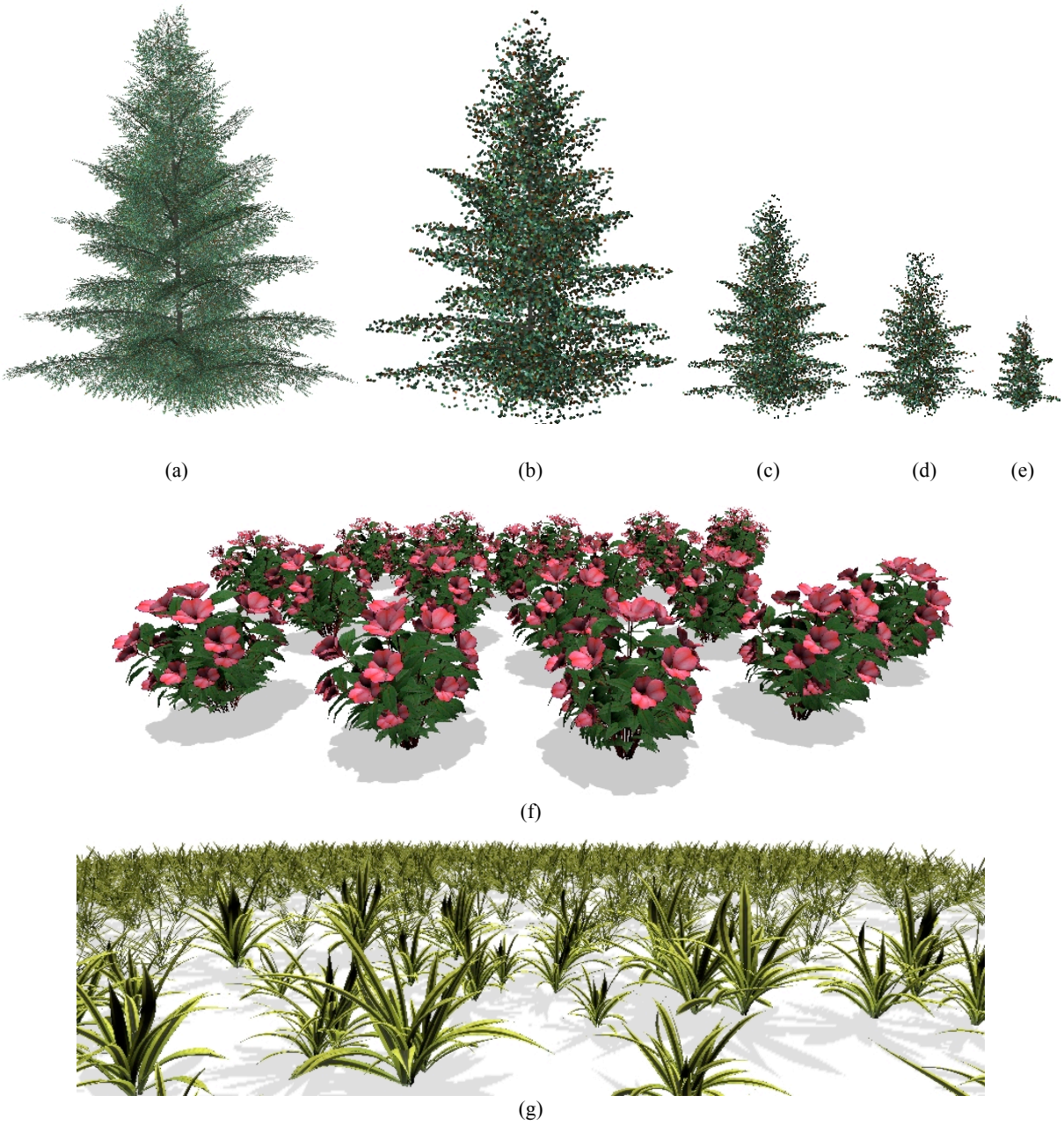

Figure 2: Level-of-detail description of a plants: (a) geometric description of a pine tree; (b) representation by 13.000 points; (c) 6.500 points; (d) 3.250 points; (e) 1.600 points; (f) several plants, the models in the background are displayed by point sets; (g) line approximation of a plant with thin leaves.

Showing larger scenes requires some additional effort. In a plant population of several million plants it is very time consuming to represent every plant even if only one point is shown. This arises from the fact that each model has to be visited and the model representation has to be read. Much cheaper is it to represent a number of plants –a square meter or larger– by a new virtual plant description. This is done by building a spatial data structure for larger terrains that stores squares of the terrain in the form of virtual plants. Doing so, even larger scenes can be displayed interactively.

On the other hand sometimes a very large plant has to be subdivided into several virtual plants. This is due to the fact that if the camera is close to such a plant, the hear leaves have to

be represented as geometry while the far leaves still can be approximated. As our LOD method works for the plant as a whole, it has to be split into parts to improve performance.
Currently we are working on hierarchical spatial data structures to enable interactive rendering speed for arbitrary landscapes and –in the farer future– for a whole synthetic planet.

4. Rendering terrain

Another time-consuming task for the computer is to represent terrain. Usually terrain data is represented by irregular sets of triangles (TIN) or regular grids for the surface and large textures for its visual appearance. Geometry and texture can be of enormous size, often it is not possible to store the whole data in main memory and therefore efficient hierarchical caching algorithms have to be used. More than that, the complexity of the data requires in many cases to model important parts of the scene with a higher complexity than others. This can be represented easily by TINs but requires some effort using regular grids. In this case, a hierarchical representation is used: a basic grid represents the scene; important parts do have their own local grids that replace the basic grid.

For display, another kind of hierarchy is required: the scene is divided into parts using a so-called quadtree. In the first step the complete scene is divided into four parts of equal size. If the geometric complexity of the parts is above a given threshold, they are further divided into four parts. All parts are stored as knots in the quadtree. Doing so, it is necessary to cut the geometry of each part at the border. Again this is easy for TINs but a harder job if a grid hierarchy is used. In this case, all the local refinements must be clipped also.

Now geometry is represented hierarchically for each part of the tree. This is done by representing the highest order knots by a base mesh that is refined by the data associated with the child knots. The algorithm ensures that the error in the altitude is always below a threshold for each approximation inside the quadtree. If the virtual viewer moves towards a point in the scene, closer and closer approximations are computed by moving downwards the tree hierarchy. Moving horizontally requires obtaining new parts of the base meshes and refining them. Both operations can be realized using solely local refinements of the geometry and by ensuring a minimal data stream. Our approach is based on a work presented by U. Thatcher [13], which allows handling such complex terrain data interactively.

5. Conclusion

It is still a difficult task to represent complex virtual landscapes interactively. The developed algorithms for modelling and rendering of single plants seem to work well for the quality needed in outdoor scenes. Indeed, much more work has to be done to handle all the data involved in this process.

In this context, the rendering capacity of modern graphics cards is not so much the problem, much more important is the bandwidth between hard disk, main memory and graphics memory to transport the enormous amount of data needed to represent outdoor scenes. Theoretically, buses like AGP 8X enable to transfer up to 2 Gigabyte per second for uploading onto the graphics board but in practice values are much lower. Therefor efficient data representation and compression techniques have to be developed to reduce the amount of data necessary to transfer. More and more operations can be performed in the graphics card, which will help to solve these problems.

Open problems are animation and interactive editing. So far our data structures are quite static and do not work well with dynamically changing data. We are able to cheat in order to simulate wind, but animations like growth of plants is not possible yet. Also, the user has limited possibilities to change landscapes interactively. This is due to the fact that the geometric description of many plants is combined to an overall representation of the scene to enable interactive rendering. Single plants can be added here, but after changing substantial parts of the scene, pre-processing must be performed again before display.

In the future we will work in this direction, our goal is to develop a frontend for GIS that allows the user to enhance GIS data by adding visual features necessary for display and then enabling him/her to visualize complex landscapes with rich vegetation and all other artefacts needed for landscaping.

References

[1] M. Aono und T. L. Kunii. Botanical Tree Image Generation. IEEE Computer Graphics and Applications, 4(5):10–34, 1984.

[2] P. Oppenheimer. Real Time Design and Animation of Fractal Plants and Trees. In: Computer Graphics (SIGGRAPH ’86 Proceedings), pp. 55—64, 1986.

[3] J. Bloomenthal. Modeling the Mighty Maple. In: B. A. Barsky, Hrsg., Computer Graphics (SIGGRAPH ’85 Proceedings), pp. 305—311, 1985.

[4] M. Holton. Strands, Gravity and Botanical Tree Imagery. Computer Graphics Forum, 13(1):57—67, 1994.

[5] P. Prusinkiewicz und A. Lindenmayer. The Algorithmic Beauty of Plants. Springer-Verlag, New York, 1990.

[6] R. Mech and P. Prusinkiewicz. Visual models of plants interacting with their environment. In Proceedings of SIGGRAPH 1996, pp. 397—410.

[7] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modelling of plants. In Proceedings of SIGGRAPH 2001, pp. 289-300.

[8] B. Lintermann and O. Deussen. Interactive modelling of plants. IEEE Computer Graphics and Applications, 19(1):56—65, 1999.

[9] O. Deussen and B. Lintermann: Designing Nature, Springer-Verlag New York, 2004 (to be published)

[10] Greenworks. Home page of the xfrog modelling software. http://www.greenworks.de.

[11] O. Deussen, P. Hanrahan, M. Pharr, B. Lintermann, R. Mech, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In SIGGRAPH 98 Conference, Proceedings, pages 275—286. ACM Siggraph

[12] [O. Deussen, C. Colditz, M. Stamminger und G. Drettakis. Interactive visualization of complex plant ecosystems. In: IEEE Visualization 2002, S. 219—226. IEEE, 2002.

[13] U. Thatcher. Rendering massive terrains using chunked level of detail, SIGGRAPH 2002 course material.