Art-based Rendering with Continuous Levels of Detail

Lee Markosian[1] *Barbara J. Meier*[1] *Michael A. Kowalski*[2] *Loring S. Holden*[1]
J.D. Northrup[1] *John F. Hughes*[1]

[1]Dept. of Computer Science
Brown University
Providence, RI 02912
{lem,bjm,lsh,jdn,jfh}@cs.brown.edu

[2]ATR Media Integration &
Communications Research Laboratories
Kyoto, 619-0288, Japan
kowalski@mic.atr.co.jp

Abstract

In previous work [6], we presented an algorithm for rendering virtual scenes using art-based styles. We demonstrated the ability to render fur, grass, and trees in a stylized manner that evoked the complexity of these textures without representing all their components explicitly. We achieved this with stroke-based procedural textures that generated detail elements, or *graftals*, just as needed.

Our implementation had several drawbacks. First, each new graftal texture required a procedural implementation that included writing code. Also, graftals were regenerated in each frame in a way that led to excessive introduction and elimination of graftals even for small changes in camera parameters. Lastly, our system provided no way to continuously vary the properties of graftals, including color, size, or stroke width. Such an ability could be used to achieve better frame-to-frame coherence, or more generally to animate graftals.

In this paper, we present a new framework for graftal textures that addresses these issues. Our new framework allows all major decisions about graftal look and behavior to be specified in a text file that can be edited by a designer. We have achieved greater frame-to-frame coherence by using graftals that remain in fixed positions on the model surface. The look and behavior of graftals as they appear or disappear can now be animated to create smooth transitions. Finally, we introduce the concept of *tufts* which manage the multiresolution behavior of graftals according to the specifications of the scene designer.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Display algorithms.

Additional Key Words: Graftals, Strokes, Non-photorealistic rendering, Procedural textures.

1 Introduction

In a recent paper [6], we presented a technique for stylized depiction of virtual 3D scenes that include grassy landscapes, trees, or furry creatures. The key observation underlying that work is that hand-drawn representations of such scenes may be inherently simple, while realistic depictions require a vast amount of detail. Thus there are potentially three advantages of the "art-based" approach to rendering such scenes. First, the underlying model representation can be orders of magnitude less complex, reducing the burden of modeling the data as well as the overhead of storing or transmitting it. Second, the cost of rendering the final image can be drastically reduced, in line with the reduction of model complexity. But third and most important, art-based rendering algorithms can be flexibly adapted to suit the required purposes of communication. Judging by the prevalence of hand-drawn images in children's picture books, for example, it appears that simple, brightly colored images that omit subtle detail but clearly delineate important objects appeal better to children than do photographs. The point about communication is also made in a number of recent papers [3, 7, 8, 9, 10, 12, 13, 15, 17, 16].

In our original method, the scene geometry itself does not contain any detail elements such as leaves, blades of grass, etc., which we call *graftals*. Instead, these are generated each frame just as needed. One benefit of this approach is that no work is expended on processing graftals that are offscreen, behind mountains, clustered too densely in the distance, or otherwise won't contribute to the final image. The downside of this approach is that achieving temporal coherence is especially difficult – the number of graftals appearing and disappearing in each frame is visually distracting. Also, because graftals are placed only on visible surfaces, none are placed just behind silhouettes, even if they would protrude above the silhouette. As new portions of surface become visible, graftals suddenly pop into view. We'd prefer graftals to be drawn if they would be partially visible, even if their base position is occluded. A second source of visual distraction is that graftals choose from among a small number of discrete "levels of detail" at which to draw. The transition between levels is sudden and visually distracting.

Another limitation of our original method is that changing the appearance or behavior of graftals is done by deriving new graftal subclasses in C++, then recompiling. This complicates the task of building an expressive user interface to let an artist easily customize the look and behavior of graftals. Providing such control to artists is essential to make graftal textures a useful medium.

In this paper, we describe a new approach to generating graftals that addresses these difficulties. The key idea is to use a "static" placement scheme, where graftals are distributed onto surfaces during the modeling phase, when the designer creates the scene. Then, in each frame, graftals are drawn (or not) view-dependently. Certain graftals, called *tufts*, have a multi-resolution structure, so that a single graftal, seen from a distance, transforms into several graftals when viewed from nearby. Further, these transformations are carried out in a continuous manner by smoothly varying the shape, size, position, and orientation of graftals as they are introduced or removed. Smaller-scale changes in level of detail (e.g. number of

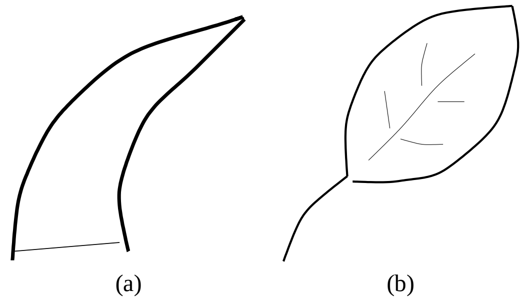

Figure 1 Simple graftals composed of several drawing primitives.

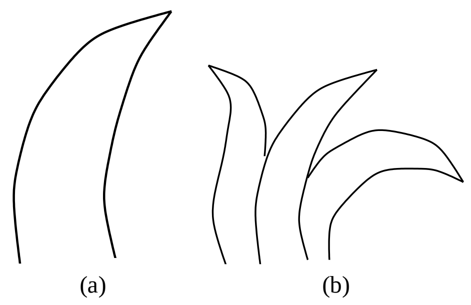

Figure 2 A tuft drawn at two levels of detail.

strokes, as well as stroke length, width, and color) are also varied view-dependently according to a continuously changing measure of desired level of detail.

In addition, we have redesigned the software framework to allow the constituent parts of graftals (strokes and filled surfaces) to be specified and edited without the need to modify and recompile code.

The benefits include dramatically improved frame-to-frame coherence and more control over the appearance and time-varying behavior of graftals, including the ability to animate them. Our multi-resolution implementation of tufts does allow some reduction of the cost of processing graftals that do not contribute to a given frame. That is, we can determine that an entire group of graftals need not be drawn in a single step, by visiting the tuft that contains them rather than visiting each graftal in the set individually. The notion of multi-resolution tufts potentially makes possible a whole new class of graftals such as those based on L-systems, as originally conceived by Alvy Ray Smith [14]. The simple design of tufts described here would have to be extended to make this possible.

2 A new framework

Our new framework is based on a small collection of *drawing primitives* that can be combined in various ways to create graftals as shown in Figure 1. The graftal in (a) consists of a triangle strip primitive (the filled inner part) and two strokes around its boundary that create an outline. As in our previous system, such a graftal could be used to depict fur or cartoonish leaves. In (b) more primitives are used to define a graftal: One or two triangle strips could be used for the filled inner part, plus strokes for the leaf outlines, stem, and veins. An advantage of separating the notion of drawing primitive from that of graftal is that new graftals can be easily created and modified by simply adding, removing, or redefining the drawing primitives that make them up.

Our new framework also contains the notion of a composite graftal, or *tuft.* These are graftals whose definition consists of other graftals, as shown in Figure 2. Both tufts and basic graftals can be assigned a position and local coordinate system, and both may decide how much of their constituent parts to draw based on view-dependent criteria. For example, in Figure 1 (b), the veins of the leaf may be drawn only when the leaf is sufficiently close to the viewer. Similarly, the tuft in Figure 2 may draw fewer leaves when viewed from a distance.

As graftals vary their level of detail under changing viewing conditions, it is important that transitions between levels be handled smoothly. For example, rather than introduce all the veins of the leaf in Figure 1 (b) at once, we might like to first introduce the central vein, followed eventually by the remaining ones. Each vein, drawn as a stroke, may itself be introduced smoothly: Its length, thickness, and color may all be manipulated independently to gradually introduce the stroke.

While the flexibility to allow these types of smooth transitions is clearly desirable, in general there is no "right way" to carry out these transitions. These questions are best addressed by an art director who seeks to achieve a particular effect; they should not be hard-coded into the implementation of distinct graftal types.

The framework described in this section is designed to provide this kind of flexibility. Both the shape and constituent parts of graftals, and the manner in which transitions between levels of detail are carried out, can be customized without any change to the code. In the current implementation, this is done by editing graftal description files; in the future we envision a graphical user interface that would allow a designer to draw graftals directly into the scene.

In the following sections we provide specific details on the implementation of drawing primitives, graftals, and tufts.

2.1 Drawing primitives

Drawing primitives come in two basic varieties: strokes and triangle-based primitives. A stroke is represented as a polyline describing its central path, plus information describing the stylization to use in drawing the stroke. This approach is much like that of "skeletal strokes" developed by Hsu *et al.* [4, 5].

Figure 3 illustrates the basic idea. In (a), the stroke path is given, together with crossbars that define the varying widths to be applied to the stroke – in this case, to taper the endpoints. This information is used to build a triangle strip (b), which is then rendered in a solid color in OpenGL [1] with an antialiased outline. The last step is achieved simply and efficiently by drawing a one-pixel wide OpenGL line strip around the outline of the triangle strip, with antialiasing enabled for lines. More details on how we achieve this and other stroke stylizations are given in an accompanying paper [11].

In addition to strokes we use triangle-based primitives. These are simply OpenGL triangle strips or fans. With these, graftals can draw filled regions such as the interior of a leaf. By combining triangle-based primitives and strokes, a wide range of graftal shapes can be produced.

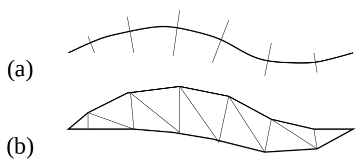

Figure 3 Defining a stroke. In (a), a stroke path is given with varying widths along it. The data is used to construct an OpenGL triangle strip (b). A stroke can be simulated by drawing the triangle strip in a solid color with an antialiased outline.

2.2 Basic graftals

Basic graftals are used to depict small detail elements such as individual leaves or blades of grass. Each graftal is composed of one or more drawing primitives, together with a set of 3D points or *vertices*. Note that some of the strokes and triangle strips in Figure 1 have vertices in common. Rather than duplicate these 3D points, drawing primitives just reference the set of vertices stored on the graftal. This makes the task of animating the graftal more straightforward: The graftal redefines its vertices, and each drawing primitive automatically "follows along."

The vertices of a graftal are defined from a set of *canonical vertices*, given in some canonical space, and an affine map M from that space to the local coordinate frame of the graftal. The local coordinate frame is defined by a position (typically on a surface) and two axes – typically the surface normal and a vector perpendicular to it. The affine map M may optionally scale and rotate the canonical vertices before applying the rigid transformation between the two coordinate frames.

2.2.1 The local frame

A graftal's local frame may be view-dependent – see for example the leaves of the bush in Figure 4. Each leaf defines its local frame from its surface position p and two axes: the surface normal $\vec{n}_p$ at p and the vector $\vec{x} = \vec{n}_p \times \vec{v}_p$, where $\vec{v}_p$ is the "view vector" pointing from p to the camera's position. Other graftals compute $\vec{x}$ differently – those on the pine trees in the animation of the night scene (Figures 7 - 8) take $\vec{x} = \vec{n}_p \times (\vec{n}_p \times \vec{y})$, where $\vec{y}$ is the (constant) vector pointing straight up. The choice of how a graftal should orient itself can be made in the graftal description file: we hard-code a few useful rules, then select the desired one for a particular graftal using a keyword.

2.2.2 Placement and duplication

Graftals are assigned positions on object surfaces. We have developed a simple editing tool for explicitly selecting points on a surface where graftals will be placed. An alternative is to place graftals procedurally over the surface. Simple procedures we have tried include placing one graftal per mesh vertex or face, or randomly choosing one face out of every k faces (chosen by the designer) at which to place a graftal. Of course, these simple procedural placement schemes require a suitable mesh triangulation.

Once surface positions have been determined for graftals, we generate graftals at those positions using a (typically) small number of example graftals whose properties are specified in the graftal description file, or *GDF*. The use of example graftals lets the scene designer change parameters of a large collection of graftals just by editing those of the example graftals in the GDF.

When a new graftal is copied from one of the example graftals, some of its parameters can be varied randomly, within a specified range of values. In this way we can vary characteristics such as size, orientation, color, stroke width, and parameters affecting the level of detail calculation described in the next section. In this way the graftals achieve a less uniform appearance.

2.2.3 Level of detail

Up to this point we have discussed how graftals can be defined from a basic canonical shape and a set of drawing primitives, distributed over a surface and provided with some rule for defining a local coordinate frame. In order to achieve effects like those in hand-drawn images, where much of the complexity of a natural scene is merely suggested with a few carefully chosen strokes, it is crucial that graftals be drawn just as needed, and with an appropriate level of detail, or *LOD*. In this section we describe the way that graftals determine an appropriate level of detail and draw accordingly (or perhaps not at all). In section 2.3 we describe an additional mechanism for controlling LOD in which graftals can be introduced or removed as needed as the camera moves through the scene.

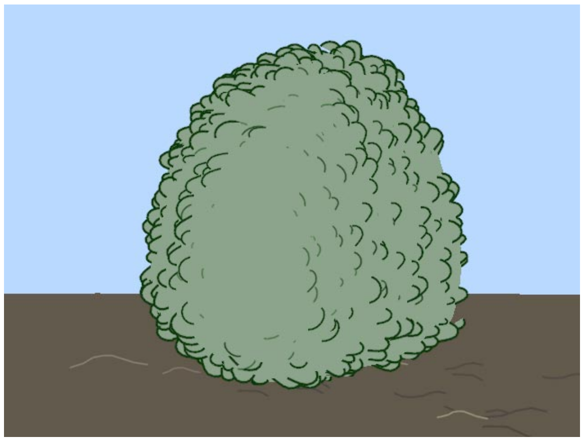
Figure 4 A bush whose leaves are duplicated (with variation) from three example leaves that are specified in a graftal description file.

The first step in drawing at an appropriate LOD is to compute a number $\lambda \in [0, \infty)$ that quantifies the desired amount of detail. That is, a small value for λ indicates little or no detail is needed, while a large value indicates much detail is needed. To compute λ, we first compute three other values: σ, which provides a measure of the graftal's size; ρ, which depends on the graftal's orientation; and τ, which measures the amount of time that has elapsed since the graftal was introduced into the scene. The elapsed time is only an issue in the context of tufts, described in the following section.

The way that a graftal's orientation affects the computation of ρ can be chosen by a designer to achieve certain effects. For graftals to be drawn primarily near silhouettes, we evaluate

$$\rho = 1 - |\vec{v}_p \cdot \vec{n}_p|$$

where, as before, $\vec{n}_p$ and $\vec{v}_p$ are the (unit length) surface normal and "view vector," respectively, at the graftal's surface position p. More generally ρ can depend on lighting calculations involving $\vec{n}_p$. In any case, ρ is computed so that its value lies in the range $[0, 1]$, with values near 0 indicating little desired detail, and values near 1 indicating full detail is allowable. The choice of which rule to use in computing ρ can be specified in the GDF in a way similar to that used for deciding the local frame.

To compute σ, which depends on the graftal's apparent size, we require that some measurement of its 3D extent be provided, as well as a value that measures its expected screen size under "normal" viewing conditions. For example, a leaf might be designed with the expectation that it will normally measure 25 pixels across. For a given frame, we determine the world space diameter D of a sphere whose screen-space diameter is 25 pixels, given that the sphere is centered along the camera's line of sight and separated from the camera position by the same distance as the graftal. Let the graftal's world-space length be W.[1] Then the ratio of these values, $\sigma = W/D$, measures the graftal's actual size in relation to its

[1] A graftal's length is actually defined by specifying two points in its canonical space that represent its extent. The vector between these points is transformed from canonical space to world coordinates; the length of the resulting vector is W.

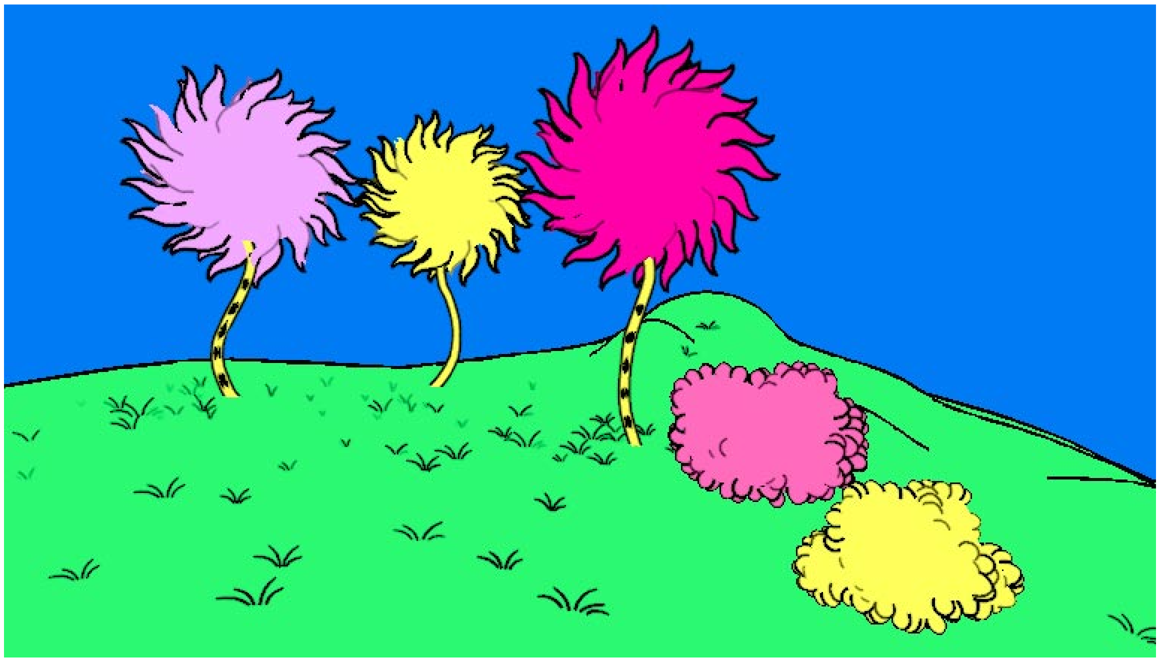

Figure 5 The truffula scene: the grass and the leaves of the bushes and truffula trees are implemented with tufts.

"preferred" size. The value σ lies in the interval $(0, \infty)$; a value of 1 indicates the graftal is the "expected" size, so it can draw with the normal amount of detail. For larger values, the graftal may draw with increased detail. For example, a leaf may draw strokes indicating veins only when σ exceeds 3, say.

The final measure λ of desired detail can be computed from σ and ρ in a variety of ways. For example: $\lambda = \rho\sigma$. In this case, ρ can be considered a modulation factor that suppresses λ under certain conditions, as when a graftal is far from a silhouette. Again, the specific rule for computing λ is specified with a keyword in the GDF. In section 3 we list some of the particular rules used by graftals in the scenes from the accompanying still images and animations.

Having computed λ, the graftal passes this value to each drawing primitive in turn as it is drawn. Drawing primitives vary how they draw according to this number. For example, strokes can vary their width, length, and color according to the passed-in value. Each attribute to be varied is given a start and end value, to be interpolated as λ varies over a specified sub-interval of $[0, \infty)$.

2.3 Tufts

We now introduce the notion of a compound graftal or *tuft*. Tufts provide a mechanism to control the screen-space density of graftals, introducing additional graftals as needed when the camera moves close, and eliminating them when the camera moves away. Tufts also allow a designer specific control over the appearance and behavior of graftals as they are introduced and removed. Specifically, the shape, position, orientation, size, and LOD of individual graftals can be varied smoothly during transitional periods as graftals are introduced and removed. The designer has direct control over how these attributes change during transitions, including the amount of time it takes to complete a transition.[2]

Because of their multi-resolution nature, tufts can provide some measure of efficiency by reducing the cost of processing graftals that are ultimately not drawn, e.g. due to their distance from the camera. That is, a single tuft might control all the blades of grass in a region of ground. When the camera is sufficiently far away, the tuft can determine with a single computation that no blades need be drawn without visiting individual blades.

[2]"Time" can be measured by the system clock for real-time interaction, or by the frame number for recorded animations.

In our implementation, each tuft contains a number of graftals, organized into distinct "levels." Associated with each level i is a value λ_i. To draw its graftals, the tuft first computes its desired LOD λ, exactly as previously described for basic graftals. It then chooses the highest level i for which $\lambda_i \leq \lambda$; graftals at this and lower levels can be drawn for the current frame.

To allow smooth transitions between levels, we don't use i (an integer value) directly. Instead we perform two filtering operations. First, we keep a LIFO queue of previous choices for i. The size of this list is chosen by the scene designer for each example graftal. Having chosen i, we temporally filter it by adding it to the queue of recent choices and calculating the average value u of the numbers in the queue. We now choose the "current level" i_c from u with hysteresis. For example, if in the previous frame the chosen level i_c was 2, and in the new frame u is 2.6, we may choose not to simply round u to find the new value for i_c. The designer can impose hysteresis by requiring u to pass a higher threshold (say 2.75) before i_c can be incremented to 3. Once i_c is incremented, u may then have to fall below 2.25 before i_c can drop back to 2. Once i_c is chosen, we draw all graftals at levels up through i_c.

As i_c varies over time, specific levels of the tuft become active or inactive. When a given level i switches from active to inactive, the graftals at that level are still drawn for a period of time T designated by the designer. We choose the number τ so that it varies from 1 to 0 during this time. Similarly, when level i first becomes active, τ varies from 0 to 1 over time T (then remains at 1). Graftals at level i can take τ into account to smoothly animate transitions when they are introduced and removed. For example, each graftal at level i might be scaled by a factor varying from 0.2 to 1 as τ ranges from 0 to 1, to make the graftal appear to grow into the scene. More generally, the canonical points of a graftal might undergo an animation (affecting both size and shape) controlled by the parameter τ. The computation of a graftal's LOD λ can also take into account τ. For

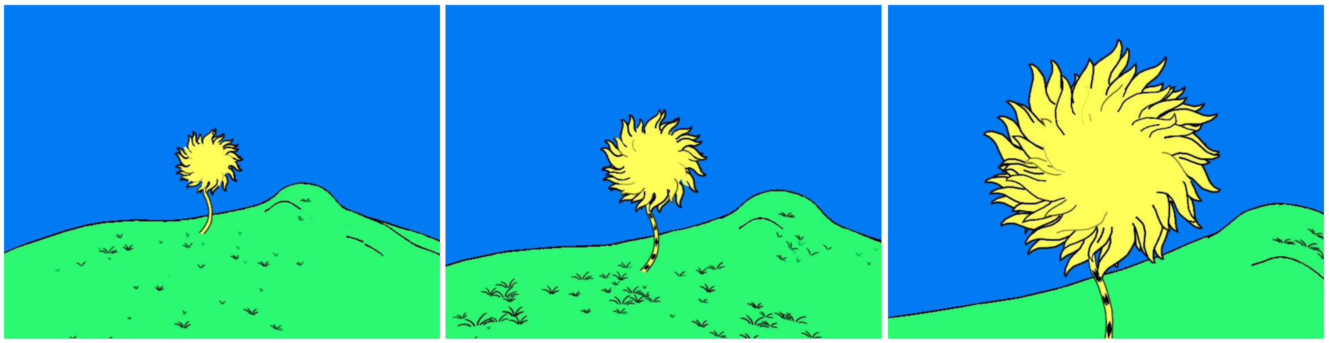

Figure 6 A single truffula shown at 3 different stages of tufts.

example, choosing $\lambda = \tau\rho\sigma$ allows the graftal's drawing primitives to smoothly increase in detail as the graftal is introduced, or decrease when it is removed.

3 Results and discussion

Figures 5 - 9 show still frames from interactive sessions and animations created with our system. In our previous work, we created imagery based on a scene in Dr. Seuss's *The Lorax* [2]. We have revisited this scene to demonstrate the improvements possible within the context of our new framework. (See Figure 5 and the accompanying animation.) As before, we have placed graftal textures on the truffula trees, bushes, and grassy ground. Figure 6 shows a single truffula tree top and demonstrates increasing levels of drawing detail as the viewer approaches the tree top. In Figure 7, we show a moonlit landscape created with our system.

The values of λ for the graftals in these scenes were computed as follows. The tufts on the ground in the night scene (Figure 7), and the tufts for the grass in the truffula scene, use $\rho = |\vec{n}_p \cdot \vec{v}_p|$, where again $\vec{n}_p$ and $\vec{v}_p$ are the unit surface normal and view vector, respectively, defined at the graftal's base position p. Then $\lambda = \rho\sigma$. This is intended to promote graftals just on surfaces that are sufficiently front-facing and near the viewer.

Tufts of the truffula trees and leaves on the bushes and pine trees use $\rho = 1 - |\vec{n}_p \cdot \vec{v}_p|$. The truffula tufts use $\lambda = \tau\rho\sigma$ so that their drawing primitives fade in gently. The tufts of the pine tree and bushes just use $\lambda = \rho\sigma$. That is, τ is used to control their size as they are introduced, but it does not modify the LOD value passed to their drawing primitives.

New graftals are introduced when their tufts decide to increase resolution. In the case of the truffula trees and the bushes, new graftals are introduced beside the original graftals. As can be seen in the accompanying animation, these graftals grow from their bases. They are rotated slightly away from the original graftal and even when full grown don't get quite as big as the original. The timing of the transitions has been perturbed so that new graftals are introduced gradually instead of simultaneously. All of these decisions were made by a designer to create an interesting but non-distracting smooth transition. As new graftals are introduced in this way, their LOD is independent of this growth phase; the drawing detail is still determined as described above as can be seen with the growing transition of the more interior truffula and bush leaves. In this example, the designer has created three levels of resolution for the trees and bushes. This was determined to be enough to show appropriate detail when the objects were far away or close to the viewer.

In designing the truffula tops, we considered several alternatives that would give different looks. One alternative was to bring in new graftals by having full-size graftals fan out from behind existing ones. Another possibility would be for a single graftal to divide into two by drawing a stroke down its middle and gradually splitting apart. Both of these alternatives are possible within the framework of our system.

Like the treetops and bushes, the grass in the truffula scene also uses tufts to manage levels of resolution. We wanted to achieve a look in which the grass strokes appeared evenly distributed, but would be drawn only when the viewer was sufficiently close. We used the editing tool mentioned in section 2.2.2 to explicitly define the positions of the tufts (and its original graftal) as well as all of the tufts' children. Our automatic distribution algorithms resulted in grass that was too evenly spaced, so we opted for the manual method. The three-blade sections of grass are single graftals and are drawn by varying the timing of their strokes.

In the night landscape, the graftals on the pine trees and shrubs were drawn with levels of detail in a similar way to the truffula tree graftals. While all the shrubs share the same graftal definitions, the ones that appear closer to the viewer show more detail than the ones further away as can be seen by comparing the close and distant ones in Figure 7. The animation of this scene and a comparison of Figures 7 and 8 show how the combined use of tufts and levels of drawing detail allow more strokes and thus more visual detail to be drawn as the camera approaches the background trees. The trees use a different orientation rule that allows the pine branch graftals to hang down. Because the underlying triangle mesh of the pine trees was uneven, the tuft positions were positioned manually using the graphical editor described above.

The designer chose to define the cloud graftals with only one level of graftal resolution. That is, no matter how close we get, no new "puffs" are created, under the assumption that in this scene we would not be flying into the clouds. If the clouds fill more of the image; however, they are drawn with increased detail, as seen in the accompanying animation and Figure 8. Like the grass in the truffula scene, the graftals on the bumpy ground in this landscape use tufts to manage the quantity of bumps, depending on how close or far away the viewer is from a particular section. Like the grass, the bumps were manually placed in groups, but the bumpy ground uses an additional level of tufts: the original graftal and its children for each manually defined group are themselves tufts that create multiple bumps. This allowed us to quickly generate enough detail without the overhead of dealing with thousands of individual elements. Once defined, the hierarchical tufts and graftals manage all the low-level decisions about what is allowed to draw.

Finally, in our last example shown in Figure 9, Ant-imation, we show a different style of resolution transition. In this scene, we show a group of ants that, from a distance, appear to be specks, but as we approach, acquire more detail in the form of antennae and legs. As we change resolutions, new ants are introduced but in a way quite different from the landscapes described above. The

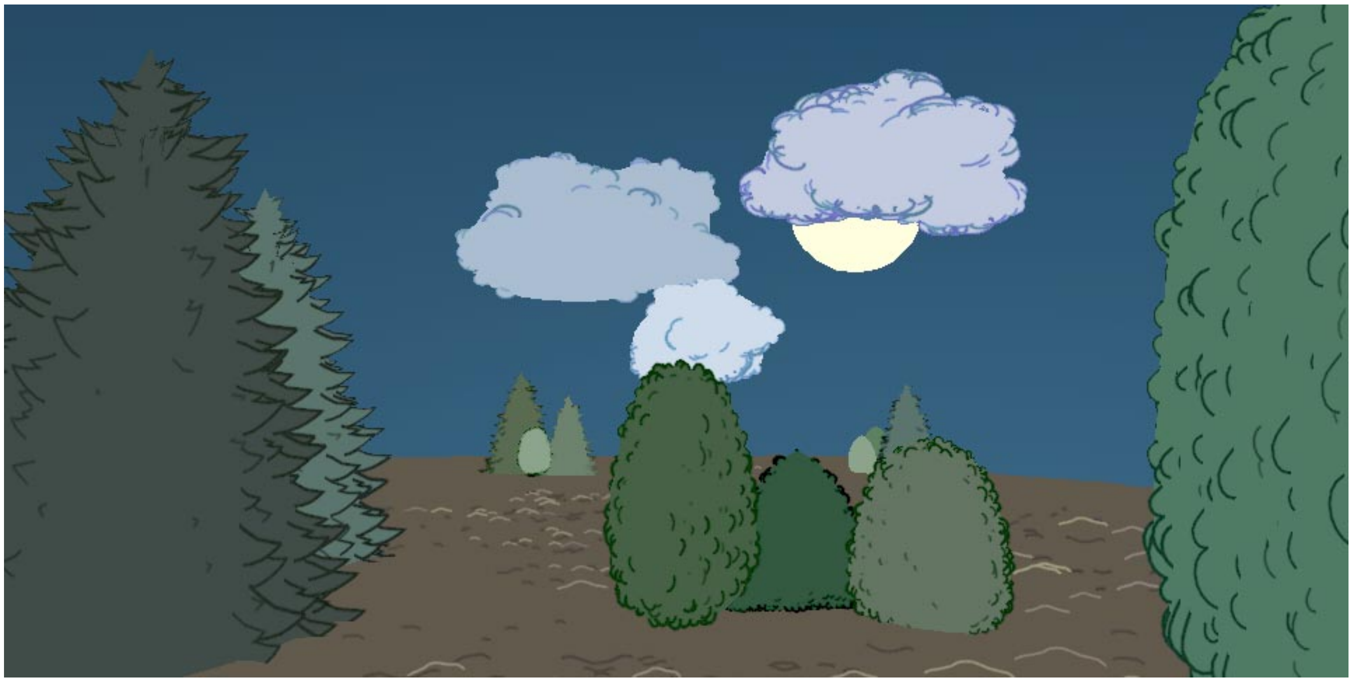

Figure 7 Night scene – beginning of camera move.

original ants actually move around to give the newly introduced ants their own space. While this whimsical example may not mimic nature, it demonstrates another style of transition choreography.

4 Conclusions

We have demonstrated a method for creating stylized illustrations and animations that evoke complex textures with a few simple shapes and strokes. We believe that stylized depictions such as those crafted by skilled users with our system can more effectively communicate many ideas that would be visually cluttered or too time-consuming to produce with traditional computer graphics.

The modeling and rendering time required is minimal compared with traditional photorealistic methods: 1 - 2 frames per second for the truffula and nighttime scenes, and 10 - 15 frames per second for simpler scenes (e.g. a single bush).[3] In this respect our new approach is comparable to the old one, though it's slower for large, complex scenes.

Our framework for creating and displaying graftal textures addresses many of the problems encountered with the previous method. Static graftals provide greater temporal coherence than before. Graftals stick to their surfaces and are not redistributed for each frame. Their "tips" remain visible when their base positions are occluded which reduces a previous source of flickering. The introduction or elimination of graftals as they become visible or disappear can be appropriately choreographed with gradual transitions as opposed to having them suddenly pop in or out of the scene.

Finally, we have created a textual format for defining a broad range of looks and behaviors for graftals that doesn't require writing new code.

[3]Our test machine is a 440 MHz CPU Sun Ultra 10 with Elite 3D graphics.

5 Future work

Graftal textures are a new medium ripe for artistic and technical exploration. Chief among our goals for future work are providing better tools for artists and designers to realize their visions. While the text file for defining graftals is a marked improvement over adding new procedural code, we would like to provide an interface that uses an artist's existing drawing and design skills. The underlying framework could be expanded to allow many additional illustration styles to achieve traditional or new looks. With each new element we created for this paper, we had many more ideas of styles to try.

On the technical side, we would like to provide better graftal distribution algorithms and better ways of choosing the level of drawing detail for graftals.

While there is still much room for exploration of graftal-based rendering, we see this work as providing one framework of many for creating varied art-based and illustrative styles. We believe that continued collaborations with designers and illustrators will evolve the way we make pictures with computers.

6 Acknowledgments

We thank Jun Ohya and Ryohei Nakatsu of ATR Labs for invaluable advice and support. This work is supported in part by the NSF STC for Computer Graphics and Scientific Visualization, Adobe, Advanced Network and Services, Alias/Wavefront, Department of Energy, IBM, Intel, Microsoft, National Tele-Immersion Initiative, Sun Microsystems, and TACO.

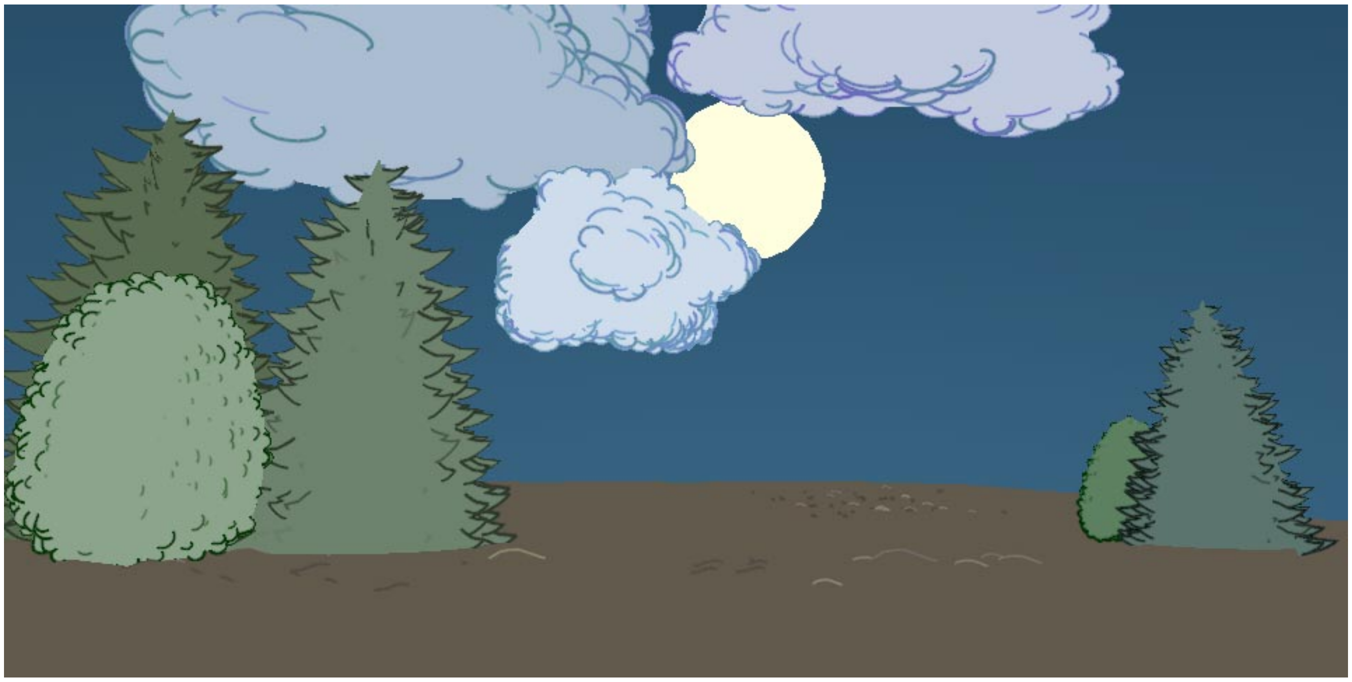

Figure 8 Night scene – end of camera move.

References

[1] OpenGL Architecture Review Board. *OpenGL Reference Manual, 2nd Edition*. Addison-Wesley Developers Press, 1996.

[2] Dr. Seuss (Theodor Geisel). *The Lorax*. Random House, New York, 1971.

[3] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-Photorealistic Lighting Model for Automatic Technical Illustration. *Proceedings of SIGGRAPH 98*, pages 447–452, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[4] S. C. Hsu, I. H. H. Lee, and H. E. Wiseman. Skeletal Strokes. In *Proceedings of UIST '93*, pages 197–206, November 1993.

[5] Siu Chi Hsu and Irene H. H. Lee. Drawing and Animation Using Skeletal Strokes. *Proceedings of SIGGRAPH 94*, pages 109–118, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[6] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, and John Hughes. Art-Based Rendering of Fur, Grass, and Trees. *Proceedings of SIGGRAPH 99*, pages 433–438, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[7] John Lansdown and Simon Schofield. Expressive Rendering: A Review of Nonphotorealistic Techniques. *IEEE Computer Graphics & Applications*, 15(3):29–37, May 1995.

[8] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-Time Nonphotorealistic Rendering. *Proceedings of SIGGRAPH 97*, pages 415–420, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[9] Maic Masuch, Lars Schuhmann, and Stefan Schlechtweg. Frame-To-Frame-Coherent Line Drawings for Illustrated Purposes. In *Proceedings of Simulation und Visualisierung '98*, pages 101–112. SCS Europe, 1998.

[10] Barbara J. Meier. Painterly Rendering for Animation. *Proceedings of SIGGRAPH 96*, pages 477–484, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[11] J. D. Northrup and Lee Markosian. Artistic Silhouettes: A Hybrid Approach. In *Proceedings of the First International Symposium on Non Photorealistic Animation and Rendering (NPAR) for Art and Entertainment*, June 2000. Held in Annecy, France.

[12] Victor Ostromoukhov. Digital Facial Engraving. *Proceedings of SIGGRAPH 99*, pages 417–424, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[13] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3D Shapes. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):197–206, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[14] Alvy Ray Smith. Plants, Fractals and Formal Languages. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):1–10, July 1984. Held in Minneapolis, Minnesota.

[15] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to Render Frames and Influence People. *Computer Graphics Forum*, 13(3):455–466, 1994.

[16] Georges Winkenbach and David H. Salesin. Rendering Parametric Surfaces in Pen and Ink. *Proceedings of SIGGRAPH 96*, pages 469–476, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[17] Georges Winkenbach and David H. Salesin. Computer-Generated Pen-And-Ink Illustration. *Proceedings of SIGGRAPH 94*, pages 91–100, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

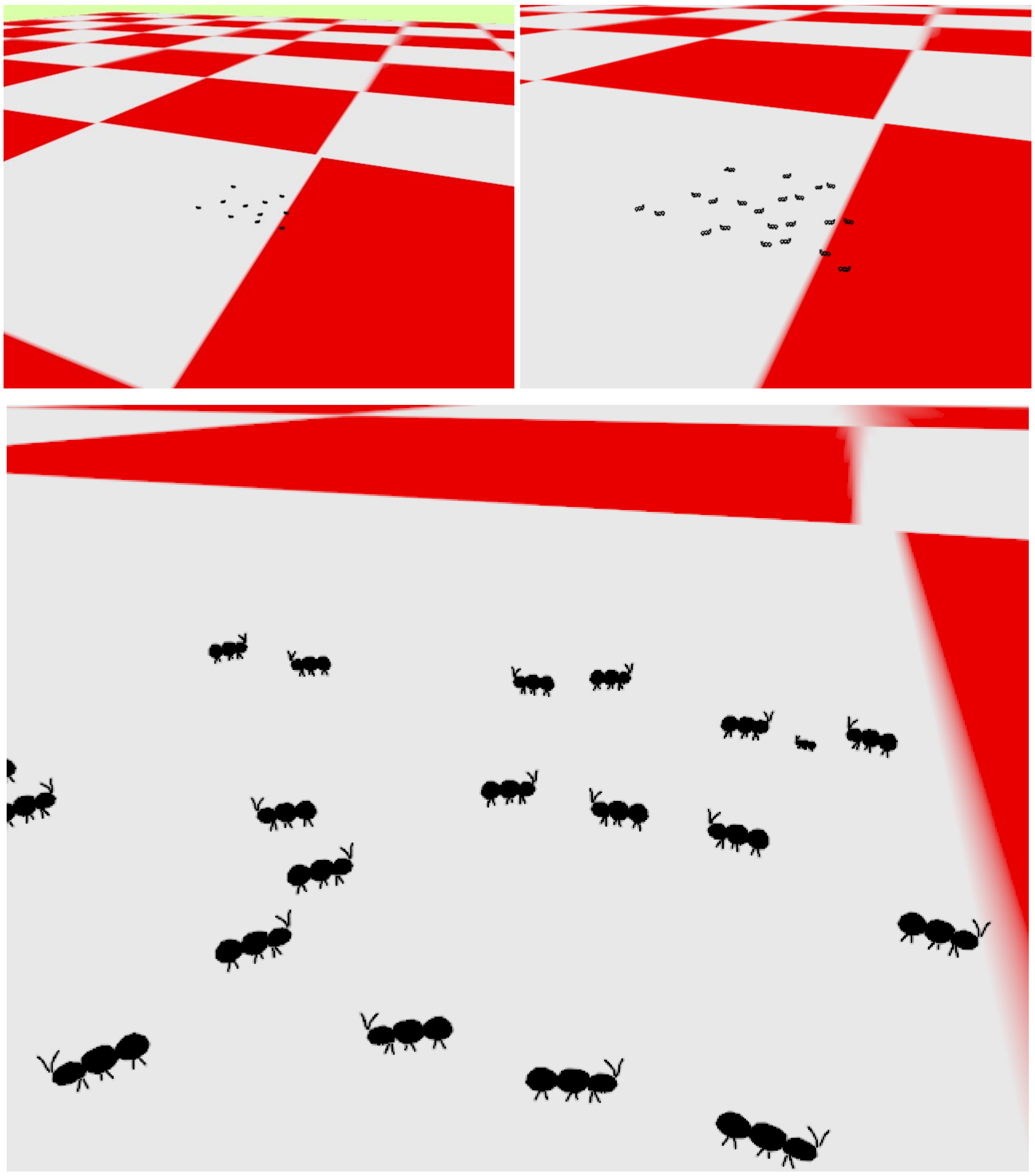

Figure 9 Ants. “I swear there was just one a second ago.”