# GenTree: An Interactive Genetic Algorithms System for Designing 3D Polygonal Tree Models

Clare Bates Congdon[1] and Raymond H. Mazza[2]

[1] Department of Computer Science, Colby College
5846 Mayflower Hill Drive, Waterville, ME 04901 USA
ccongdon@colby.edu
[2] Entertainment Technology Center, Carnegie Mellon University
5000 Forbes Avenue, Doherty Hall 4301, Pittsburgh, PA 15213 USA
rmazza@andrew.cmu.edu

**Abstract.** The creation of individual 3D models to include within a virtual world can be a time-consuming process. The standard approach to streamline this is to use procedural modeling tools, where the user adjusts a set of parameters that defines the tree. We have designed GenTree, an interactive system that uses a genetic algorithms (GA) approach to evolve procedural 3D tree models. GenTree is a hybrid system, combining the standard parameter adjustment and an interactive GA. The parameters may be changed by the user directly or via the GA process, which combines parameters from pairs of parent trees into those that describe novel trees. The GA component enables the system to be used by someone who is ignorant of the parameters that define the trees. Furthermore, combining the standard interactive design process with GA design decreases the time and patience required to design realistic 3D polygonal trees by either method alone.

## 1 Introduction

This paper describes GenTree, an interactive system that models 3D polygonal trees as a set of parameters for a procedural tree description and uses genetic algorithms (GA's) to evolve these parameters.

Realistic-looking organic objects for 3D virtual worlds are time consuming to model. Typically, designers must become skilled in the use of a commercial modeling program such as 3D Studio Max or Maya in order to be able to construct realistic tree models to be used in simulations. Creation of a single tree by even a skilled modeler would typically take several hours using this approach.

Procedural modeling systems, such as described in [11] (or commercial systems SpeedTree and Tree Storm) are intended to streamline the tree model creation process. They allow users to adjust a number of parameters that control the appearance of a tree without having to focus on low-level components of the model.

In procedural systems, many of the possible parameters interact with each other; for example, the length of branches, the base width of branches relative to

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2034–2045, 2003.

the parent branch, and the rate at which they taper will combine to determine the width of the tip of each branch. Thus, it can be time consuming and difficult to design a tree with a desired look, and it is difficult for the user to gain an understanding of the effects and interactions of the parameters. Since the trees are constructed one at a time, exploring variations (and creating multiple models) is further time consuming. Furthermore, more realism and variation in the space of possible trees is achieved in procedural tree programs by increasing the parameter space. Increasing the size of this search space increases the burden of using and learning to use such a system.

GenTree seeks to extend the capabilities of the procedural tree-building approach by adding a GA component to the search process. GenTree enables users, even those with no experience in using 3D modeling tools, to quickly design trees for use in virtual worlds. The system is interactive, allowing (but not requiring) the user to adjust the parameters that define each tree. The system also allows (but does not require) the user to assign a numeric rating to each tree; this information is then used by the GA to favor the more appealing trees as parents in the evolutionary process.

This work does not focus on rendering the trees smoothly, and is instead focused on specifying the parameters that define the general shapes of the trees. Consequently, there is much room for improvement in the component of the system that renders the trees, depending on the intended use for these trees.

## 2 Background

There are several examples of evolutionary approaches to design in recent literature, including [1][2]. These collections in particular include several examples of aesthetic evolutionary projects, but differ in focus from the project presented here.

Sims evolves 3D creatures [1] (and [10][9]), but his focus was on the mobility of the creatures (for example, joints) and not a natural appearance of the creatures. Soddu [2] evolves cityscapes, composed of individual buildings composed in turn of architectural elements. These do not have the added constraint of intended realism in modeling natural objects. Hancock and Frowd's work in the generation of faces is also relevant[2]; however, this project is aimed at altering photographs, rather than constructing 3D models of the faces.

Bentley's system [1] evolves solutions to a wide variety of 3D modeling problems, but focuses on problems that can be evaluated independently of aesthetic considerations. From these collections, the work here is most similar to an extension to Bentley, reported in [2], which allowed the GADES system to interact with a user. The system is able to evolve a wide variety of shapes in response to the user-provided fitness feedback.

L-systems an an established approach to modeling the development of plants, for example [5][7], and has been used directly or inspired previous work on the evolution of plant forms, for example [6]. Finding a set of rules to achieve a desired appearance is a search problem, not unlike the search for parameters

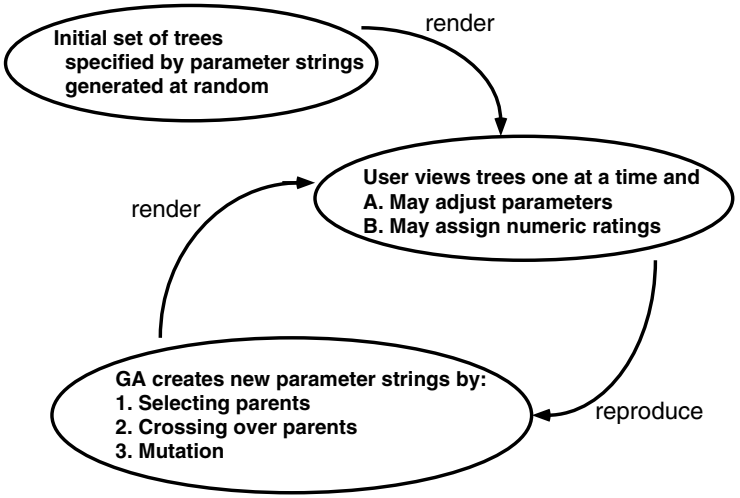


**Fig. 1.** GenTree iterates between user input and the genetic algorithm in the creation of new trees.

in a procedural tree-modeling program. However, since these systems are evolving rule sets or hierarchical structures, the concerns are different from those in this paper. Also, since the underlying representations are more complex, these approaches are much harder for users to attempt without a genetic algorithms approach (or another form of help with the search process).

Sims [8] evolved plant shapes using parameter strings, but the system is strictly evolutionary and does not allow for interactive parameter tuning. In [4], we describe a GA approach to designing 3D polygonal trees that led to the work reported here. However, this work was an initial exploration of the concept and uses a limited parameter set, resulting in fanciful, rather than realistic, trees. Also, like [8], the system is strictly evolutionary, so the trade-offs between a strictly procedural approach and an evolutionary approach are not explored, nor is the potential for a hybrid approach explored.

## 3 System Details

As demonstrated in [8] and [4], a GA approach is able to help navigate the search space of a procedural tree-building program. The parameters that define the trees are easily represented as strings, and the user-provided rating of the quality of the solutions serves as a fitness function. Finally, new (offspring) parameter sets can be derived from existing (parent) parameter sets using typical GA processes of crossover and mutation.

This paper extends these ideas to combine the interactive approach of a standard procedural program (where the user adjusts parameters) with an interactive GA.

```
main
  create (random) initial population
  getFitness
  while (maxTrials not reached)
    generateNew
    getFitness

generateNew
  select parents
  mutate
  crossover
  save best (elitism)
```

**Fig. 2.** High-level pseudocode for the GA system in GenTree.

```
getFitness
  (all trees start with default fitness of 0.0)
  while (not DoneWithGeneration)
    renderTree -- display each tree for user
      (each tree rotates to show all angles)
    user may increment or decrement fitness for each tree
    user may advance to next tree or backup to previous tree
    user may change any of the parameters that define the tree
    user may signal DoneWithGeneration
```

**Fig. 3.** High-level pseudocode for the interactive component.

### 3.1 System Overview

GenTree starts with an initial population of trees whose parameters are generated at random. The user may look at each tree in turn, may change any of the parameters that define any of the trees, and may assign each tree a numeric rating that indicates its quality. When signaled by the user, a new generation of trees is produced from the old, using the GA process. Most notably, the parameter sets from two different trees "cross over", producing two new parameter sets. The new set of trees is viewed by the user, and may again be altered and/or rated, and a new set of trees produced from the old via the GA process. A high-level view of the system is illustrated in Figure 1; pseudocode is shown in Figures 2 and 3.

### 3.2 GA Component

To hasten the development of our system, we used Genesis [3] for the genetic algorithms component of GenTree; the Genesis code has been combined with a modest OpenGL system (developed by the authors) to display the trees for evaluation by the user. We named the system GenTree to reflect a genetic algorithms approach to creating tree models.

**Table 1.** A description of the values represented in the strings evolved by GenTree and rendered as 3D polygonal trees.

| Gene # | Name | Description | Range of Values |
|---|---|---|---|
| 0 | Number of Branches | The number of sub-branches that protrude from any one branch (modified by noise) | [1..8] |
| 1 | Length Ratio | The ratio of the length of a branch to that of the parent branch | [0.2 .. 0.9] |
| 2 | Width Ratio | The ratio of the width of the base of a sub-branch to that of the parent branch | [0.2 .. 0.9] |
| 3 | Branch Taper | The ratio of the end of a branch to the base of a branch | [0.2 .. 0.9] |
| 4 | Branch Proximity | From the tip of the parent branch, the percent of the parent to be used for distributing the sub-branches along the parent. | [0.1 .. 0.8] |
| 5 | Tree Depth | The number of branches that can be traversed from the trunk to the tips (excluding the trunk) | [2 .. 9] |
| 6 | Branch Angle Delta | The change in x and z direction in the vector that determines the angle of a sub-branch, relative to the parent branch | [0.0 .. 0.1] |
| 7 | Vertical change | Added to direction inherited from parent branch. | [0.0 .. 1.0] |
| 8 | Parent influence | Strength of influence of parent vector on offspring branches. | [0.0 .. 1.0] |
| 9 | Branch Direction Noise | Noise in the x, y, and z direction of the growth of a branch section (off the defined normal vector for the branch section) | [0.0 .. 1.0] |
| 10 | Branch Number Noise | Noise in determining the number of branches at each level of growth (higher noise tends to add branches) | [0.0 .. 1.0] |
| 11 | Subtree Depth Noise | Noise in determining how deep a subtree is (higher noise tends to truncate subtrees) | [0.0 .. 1.0] |
| 12 | Random Seed | Stored with tree parameters so that tree will render consistently | [0.0 .. 1.0] |

The mechanics of the genetic algorithm (GA) component of the system are largely unchanged from the original Genesis code, although the interactive nature of the system changes the flow of control, since mouse clicks and keystrokes from the user affect the GA. This will be described later in this section.

### 3.3 Procedural Tree Component

In designing the parameters used to define the trees, we looked to existing procedural tree programs, plus a bit of trial and error, to determine what would make good looking trees. The parameters evolved by the system and their ranges are provided in Table 3.3.

```
renderTree
  drawSpan(trunk)

drawSpan
  setDirection
  draw conic section
  flip biased coin to possibly add a branch (bias determined by gene 10)
  if (subBranches)
    for each branch
      flip biased coin to possibly make a smaller subtree
        (bias determined by gene 11)
      drawSpan(branch)

setDirection
  if trunk, straight up
  else, combine normal vectors for:
    parent direction, weighted by gene 8
    random compass direction, weighted by gene 6
    random y direction, weighted by gene 7
```

**Fig. 4.** High-level pseudocode for the rendering component.

### 3.4 Rendering a GA String into OpenGL Trees

The trees are rendered recursively from the trunk through the branches. High-level pseudocode for the rendering process is provided in Figure 4. Each GA string uniquely translates into a specific rendering, including a seed used for random numbers (gene 12), so the process is deterministic for a given string.

The trunk and each branch are rendered as primitive conic sections[1]. The trees are generated without leaves, but green triangles can be generated at the tips of the branches to assess what a tree would look like if it had leaves on it.

## 4 Results

One is first tempted to evaluate the GenTree system by judging how good the resulting trees look, and based on this criteria, the system is able to generate good looking trees in a short period of runtime.

[1] In OpenGL, conic sections are drawn using a specified number of faces. Because the rendering component is not the focus of the work reported here, the number of faces used here is four to keep the polygon counts low and the rendering fast. Thus, the sections are rectangular.

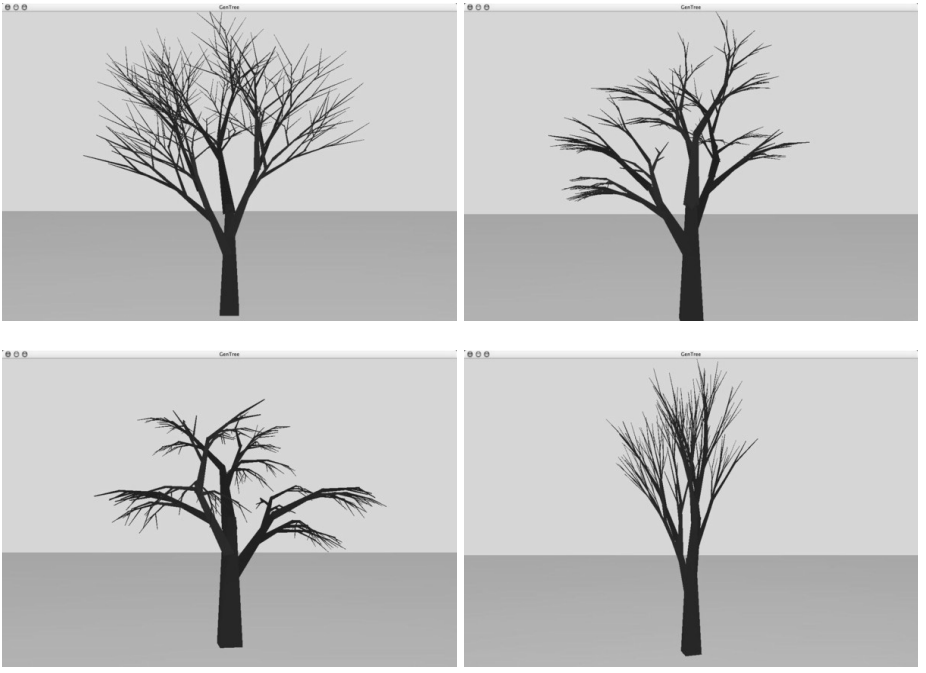

**Fig. 5.** A variety of trees from the final population.

### 4.1 Resulting Trees

Subjective results show that with an initial population of 20 trees[2], several interesting and realistic looking trees can be produced within 5-10 generations of the GA and with less than an hour of the user's time. Because the GA is working with several models simultaneously, the end result of a run is a population of trees, several of which are likely to be attractive and useful. Furthermore, similar looking but distinct trees can be produced from a favorite tree by altering its parameters. Therefore, it is possible to easily construct a variety of trees as the result of one run of the system.

Figure 5 shows a variety of trees from the final population from a run of 7 generations of 20 trees, with both interactive fitness and interactive adjustment used.

Figure 6 shows another example tree produced at the end of the same run, and then shows this same tree with leaves added.

Recall that since the trees (and leaves) are rendered using simple cylinders (and triangles), it is the structure of the trees that should be considered, and not the surfaces of the trees.

[2] A population size of more than 20 starts to become tedious from the usability perspective, as the user will typically view and interact with each tree before advancing to the next generation.

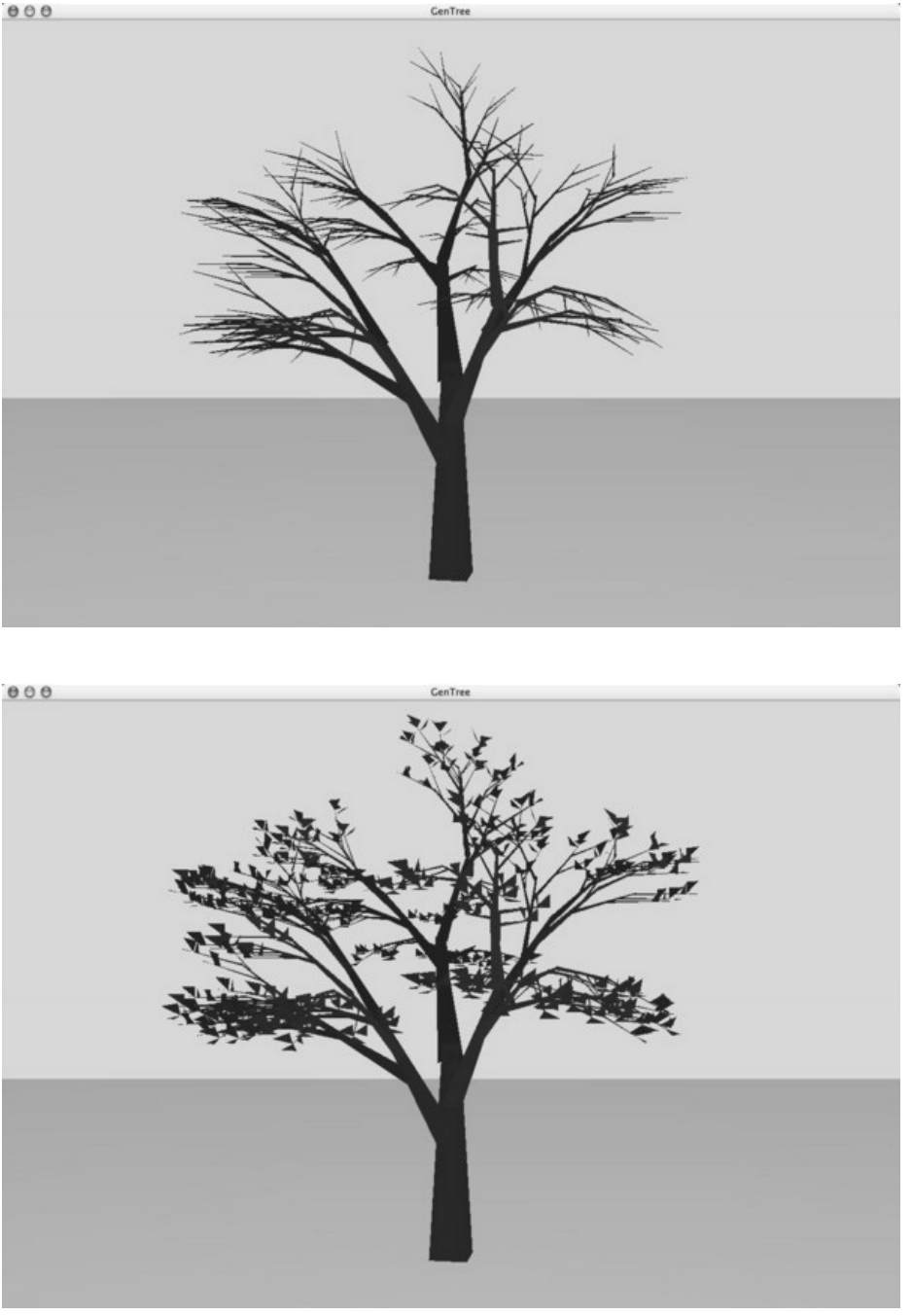


**Fig. 6.** One of the best trees from the final population, shown with and without leaves.

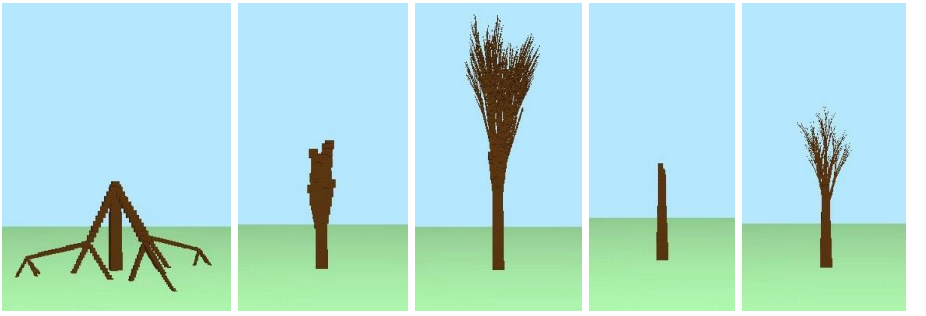

**Fig. 7.** Five example trees from the initial population.

### 4.2 Initial Population

As a reference point, a sampling of trees from the initial population are shown in Figure 7. This sample (every fourth tree) includes the two from the initial population that most resemble trees (the third and the fifth shown), as well as one with branches growing into the ground and two with branches growing inside each other. The three clearly errant trees shown in this figure illustrate typical features of the randomly generated trees.

### 4.3 Comparison of Interaction Styles

The primary question to be asked in the context of the work reported here is whether the GA component assists with the development of the 3D tree models. To assess this, the system was run four different ways:

1. Without providing fitnesses for trees and without altering them in any way.
2. Altering tree parameters for trees that can be improved upon each generation, but not rating the trees.
3. Rating the trees, but not altering them beyond the GA process.
4. Altering tree parameters for trees that can be improved upon each generation and also rating these trees before they are used in the GA process.

In the first instance, the system starts with randomly generated trees and recombines them, so it is not surprising that it does not produce anything interesting. It has been mentioned here merely for completeness.

The second instance is not typical of the GA process in that no information is provided to the GA about solutions that are "better" or "worse" than others, and this information is typically used for preferential parent selection. However, in this case, the alternative mechanism of "eugenics" (helping there be more good solutions to be used as parents) achieves a similar role of recombining primarily good solutions.

The third instance corresponds to the standard use of Genetic Algorithms, and the fourth instance corresponds to the full capacity of the GenTree system to take input from the user in two forms.

The system was run by the authors in all four variants with the general goal of finding good-looking trees with minimal effort. Although this remains a subjective test and evaluation, the authors explicitly did not fine tune any trees in the adjustment process. A population size of 20 trees was used, with a crossover rate of 60 percent and a mutation rate of 0.1 percent (each bit has a .1 percent chance of being set to a random 0 or 1 value). With each approach save the first, for each variation of altering trees and supplying ratings, GA generations were run until approximately an hour had elapsed. This resulted in 5 generations for variation 2, 20 generations for variation 3, and 7 generations for variation 4.

An explanation for the varying numbers of generations is that with the second variation, the most time was spent adjusting the parameters to each tree, since all trees are equally likely to be "parents" to the next generation. In variation 4, when the ratings were used, bad trees (those that could not easily be adjusted) were simply given low ratings. This happens sometimes, for example, when a very angular tree pops up. (Such a tree can be useful "grist for the mill" or might be useful when constructing deliberately fanciful trees, but can sometimes be difficult to smooth out without considering how parameters interact and perhaps changing the parameter set severely. Thus, it can be expedient to simply give a difficult tree a low rating.) With the third variation, no alterations were allowed, so each tree was viewed and assigned a rating, allowing more generations in the time frame.

All three approaches are capable of producing comparable trees, and the assessment of "best" may have to do with the user's background. As users familiar with the available parameters, it is frustrating to do the third run and restrain oneself from "touching up" a promising tree. However, the ease with which a naive user can design trees using only ratings and without knowing the parameters has been witnessed several times in curious colleagues trying the software. For more knowledgeable users, there seems to be little reason to run the system with the second variation because the ability to weed out bad trees or favor interesting ones can be expedient, even if used only occasionally.

The differences observed between the second and fourth variations are that when both ratings and alterations are used, the trees in the final population tend to show more variation than when just alterations (and no ratings) are used. This is a somewhat counterintuitive result in that in many GA applications, the population tends to converge, and it is impossible to discern at this point whether this is a quirk of the author's use of the system or a more meaningful effect. However a possible explanation can be constructed in that when adjusting most of the trees each generation (and not being allowed to simply indicate a tree is undesirable and leave it behind), it appears likely that one gets into the habit of adjusting the same parameters into similar values. Whereas when the ratings are used, less parameter adjusting happens and therefore less bias in adjusting particular values is introduced. (This is just one possible explanation of observed differences in the final populations.)

The question must also be asked about the results of altering parameters only (without advancing any generations in the GA process) for an hour, during which

time a knowledgeable user would surely be able to produce one or more attractive trees. Using the system this way, more skill would be required in assessing the relative contributions of the different parameters and the interactions of the various parameter settings with each other. The addition of the GA process (with or without ratings) considerably lessens the need for understanding the effects of individual parameters, or even knowing what the parameters are.

### Further Observations

Overall, we are quite pleased with the wide variety of shapes that are possible in the system and its ability to yield some trees that are approaching realism. Not surprisingly, the noise parameters appear to be instrumental in making the trees look more realistic and providing more variations of "similar" trees.

## 5 Conclusions and Future Work

The GenTree system is able to produce a variety of trees with realistic shapes. Although it is difficult to effectively evaluate a system whose output is judged on aesthetic criteria, the addition of the GA mechanisms to a procedural tree modeling system simplifies the development of 3D tree models in that users do not need to understand the effects of individual parameters or even know what the parameters are in order to develop useful models. Rather than having to identify why a tree is good or bad (or the ways in which a tree is good or bad), the user can assign a subjective rating.

### Future Work

The introduction of more parameters into the procedural definition of the tree would be useful in that it would allow for a greater variety of trees, including pine trees and palm trees. However, this would also make the space of possible trees more difficult for users to navigate. Thus, as the parameter space increases, the GA mechanisms are expected to be even more useful for helping the user discover appealing trees.

The trees would be significantly more realistic by improving the rendering engine, even if the GA and interactive facets of the system were left unchanged. This would include adding textures to the surfaces and smoother transitions from branches to sub-branches. Furthermore, it may be desirable to parameterize the rendering component, for example, to allow favoring low-polygon-count trees (for some applications) or favoring realism.

Additional benefits could be seen from parameterizing leaf characteristics and improving the rendering of the leaves. Root structure would also make trees more realistic. In addition, different colors and textures could be specified for bark and for leaves, and the colors and textures could be subject to evolution. The ability to add flowers and fruit, and variations on these, could also be added as parameters.

In a few years, we will no longer be constrained by polygonal limits in virtual worlds; some applications will call for trees of high intricacy, which will be too time consuming even to attempt to model through regular methods. There is an obvious advantage to using GenTree in this situation.

## References


1. P. J. Bentley. *Evolutionary Design By Computers.* Morgan Kaufmann, San Francisco, CA, 1999.
2. P. J. Bentley and D. W. Corne. *Creative Evolutionary Systems.* Morgan Kaufmann, San Francisco, CA, 2002.
3. J. J. Grefenstette. A user's guide to GENESIS. Technical report, Navy Center for Applied Research in AI, Washington, DC, 1987. Source code updated 1990; available at `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/genesis/`.
4. R. H. Mazza III and C. B. Congdon. Towards a genetic algorithms approach to designing 3D polygonal tree models. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002).* IEEE Computer Society Press, 2002.
5. C. Jacob. Genetic L-system programming. In *Parallel Problem Solving from Nature Proceedings (PPSN III), Lecture Notes in Computer Science*, volume 866. Springer, 1994.
6. G. Ochoa. On genetic algorithms and Lindenmayer systems. In *Parallel Problem Solving from Nature Proceedings (PPSN V), Lecture Notes in Computer Science*, volume 1498. Springer, 1998.
7. P. Prusinkiewicz, M. Hammel, R. Mech, and J. Hanan. The artificial life of plants. In *SIGGRAPH course notes, Artificial Life for Graphics, Animation, and Virtual Reality.* ACM Press, 1995.
8. K. Sims. Artificial evolution for computer graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques (SIGGRAPH91)*, pages 319–328. ACM Press, 1991.
9. K. Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV Proceedings*, pages 28–39. MIT Press, 1994.
10. K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH94)*, pages 15–22. ACM Press, 1994.
11. J. Weber and J. Penn. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH95)*, pages 119–128. ACM Press, 1995.