Towards a Genetic Algorithms Approach
to Designing 3D Polygonal Tree Models

Raymond H. Mazza, III
Entertainment Technology Center
Carnegie Mellon University
5000 Forbes Avenue
Doherty Hall 4301
Pittsburgh, PA 15213
rmazza@andrew.cmu.edu

Clare Bates Congdon
Department of Computer Science
Colby College
5846 Mayflower Hill Drive
Waterville, ME 04901
ccongdon@colby.edu

Abstract - The OpenGL graphics libraries [4] provide a powerful environment for programming three-dimensional virtual worlds, and are used in a wide variety of applications. However, the creation of individual 3D models to include within a virtual world is an arduous and time-consuming process. We have designed GenTree, an interactive system that uses genetic algorithms to evolve 3D tree models to be used in virtual environments. The system evolves a set of parameters used to render the trees, with a fitness function provided by user input. The initial system works with a modest set of parameters and uses relatively crude rectangles for rendering, but is able to evolve a variety of plant shapes, including deciduous trees, evergreens, and shrubbery.

I. INTRODUCTION

The OpenGL graphics libraries [4] provide a stable and powerful environment for programming three-dimensional virtual worlds, and are used in a wide variety of applications, including interactive games and animated movies. However, the creation of individual objects to include within a virtual world is an arduous and time-consuming process, particularly when there is a desire for realistic-looking organic objects. Typically, designers must become skilled in the use of a modeling program such as 3D Studio Max or Maya in order to be able to construct the models to be used in their simulation. In addition to striving for realistic appearance of these models, care must also be taken to keep the count of component polygons low, to hasten rendering time. This is particularly true of interactive worlds that are rendered in real time. Creation of a single tree by even a skilled modeler would typically take several hours, and represents just one small part of a virtual world.

This paper described GenTree, a system that uses genetic algorithms to evolve parameters used to construct tree models to be used in OpenGL virtual worlds. The system is interactive, allowing the user to identify the trees that are most (and least) appealing. Although designed with deciduous trees in mind, GenTree is able to create a wide variety of plant shapes, including deciduous trees both with and without leaves, evergreens, cactus-like plants, and shrubbery.

Related Work

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

Sims evolved 3D creatures [1], [6], and [5], but his focus was on the mobility of the creatures (for example, joints) and not the aesthetic appearance of the creatures. Soddu [2] evolved 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 work was aimed at altering photographs, rather than constructing 3D models of the faces.

Bentley’s system [1] evolved solutions to a wide variety of 3D modeling problems, but focused on problems that can be evaluated independently of aesthetic considerations. 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. Probably because of the initial intentions of the system, the resulting figures are rather minimalistic in form.

II. OVERVIEW OF THE GENTREE SYSTEM

This work represents an exploration of the potential for using an evolutionary approach to developing models of trees for 3D virtual worlds. Therefore, we used components of existing systems to the extent possible. In particular, we used Genesis [3] for the genetic algorithms component. The Genesis code has been combined with an OpenGL 3D rendering engine (developed by the authors) to display the trees for evaluation by the user. We named the system GenTree to reflect a genetic algorithm approach to creating tree models.

The GenTree system evolves bitstrings that represent parameters defining a tree to be rendered using the 3D engine developed for this work. The fitness for each tree model in the population is provided interactively by the user. The overall system is illustrated in Figure 1. Pseudocode for the basic GA component of the system is provided in Figure 2, and the inter-
A. GA Mechanism

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

B. The Interactive Fitness Function

The initial population of trees is created at random. On each generation, for each tree in the population, the user is shown the tree to be evaluated. The tree is displayed rotating around the y axis in a 3D world, so that the user sees all perspectives of the tree. (The user may also change the viewing angle to see the tree from different angles.) The user may then increase or decrease the fitness of the tree according to his or her preferences. The user may also advance to the next tree, or back to a previous tree, or may signal that the evaluation of this generation is complete. The next generation of trees is then created, using the standard Genesis operations of parent selection, crossover, mutation, and elitism, which operate on the phenotype of the bitstring, rather than the genotype of the rendered tree. The evolution process continues until terminated by the user or until the specified number of generations has elapsed.

All trees are evaluated each generation so that users may change their evaluations of trees relative to others in the population. Each click from the user increases or decreases the fitness of the current tree by 0.5; the initial fitness for each tree is 0.0.

It is possible that the tree specified by a GA string will have too many polygons to effectively be rendered. When this happens, these trees are automatically given the lowest possible fitness (-10.0).

C. Representation

GenTree evolves strings that represent the tree to be rendered by the OpenGL engine. This section describes the elements of the GA string. There are 15 different genes in the bit string, which represent attributes such as how branchy the tree will be, how long the branches will be, how wide the branches will be, how quickly the branches will taper, the number of branches from trunk to tip in the tree, the angle of the branches (relative to the parent branch), the location of branches on the parent branch, and the colors to be used for trunk and leaves. The bit string is described more fully in Table I.

Some clarifications for a few of the genes:

1. The parameter that determines leaf size excludes the value zero, which would otherwise have the equivalent effect as 0 percent leafiness.

2. A branch proximity of 0 corresponds to sub-branches evenly distributed over the parent branch; a value of 1 corresponds to all sub-branches occurring at the tip of the parent. A value of 0.5 means that sub-branches will be distributed on the outer half of the branch.

3. The branch length and branch number genes are also used to seed a random number generator that uniquely determines for each tree which of the x, y, and z will be affected by the branch angle delta. This determines the angle of the sub-branch from the parent.

D. Rendering a GA string into OpenGL Trees

Figure 4 provides pseudocode for the process that translates GA strings into 3D OpenGL trees. Each string uniquely translates into a specific rendering.

The trunk and each branch is rendered as a series of cubes. The number of cubes is calculated based on the branch width, branch length, and branch taper parameters, so that branches that taper more will be comprised of more cubes. Then the dimensions of each individual cube is calculated.

The tree is rendered from trunk to leaves. After the trunk is drawn, the process is recursive and deterministic. A random number is used to determine the x, y, and z angle of the sub-branches relative to the parent branches, but the seed for this
getFitness
(all trees start with default fitness of 0.0)
while (not DoneWithGeneration)

display trees for user (each tree rotates to show all angles)
user may increment (left click) or decrement (right click) fitness for each tree
user may advance to next tree (right arrow)
user may backup to previous tree (left arrow)
if (user clicks return) DoneWithGeneration = TRUE

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

<table>
<thead>
<tr>
<th>Gene #</th>
<th>Name</th>
<th>Description</th>
<th>Number of Bits</th>
<th>Range of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Number of Branches</td>
<td>The number of sub-branches that protrude from any one branch</td>
<td>3</td>
<td>[1..8]</td>
</tr>
<tr>
<td>1</td>
<td>Width Ratio</td>
<td>The ratio of the width of the base of a sub-branch to that of the parent branch</td>
<td>3</td>
<td>[.25 .. 1.0]</td>
</tr>
<tr>
<td>2</td>
<td>Length Ratio</td>
<td>The ratio of the length of a branch to that of the parent branch</td>
<td>3</td>
<td>[.25 .. 1.0]</td>
</tr>
<tr>
<td>3</td>
<td>Tree Depth</td>
<td>The number of branches that can be traversed from the trunk to the tips (excluding the trunk)</td>
<td>3</td>
<td>[0 .. 7]</td>
</tr>
<tr>
<td>4</td>
<td>Branch Angle Delta</td>
<td>The change in x, y, and z in the vector that determines the angle of a sub-branch, relative to the parent branch</td>
<td>5</td>
<td>[-2 .. 2]</td>
</tr>
<tr>
<td>5</td>
<td>Wood Red</td>
<td>The amount of red color present in the trunk and branches</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>6</td>
<td>Wood Green</td>
<td>The amount of green color present in the trunk and branches</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>7</td>
<td>Wood Blue</td>
<td>The amount of blue color present in the trunk and branches</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>8</td>
<td>Leaf Red</td>
<td>The amount of red color present in the leaves</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>9</td>
<td>Leaf Green</td>
<td>The amount of green color present in the leaves</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>10</td>
<td>Leaf Blue</td>
<td>The amount of blue color present in the leaves</td>
<td>4</td>
<td>[0 .. 1.0]</td>
</tr>
<tr>
<td>11</td>
<td>Leafiness</td>
<td>The percentage of branches that are tipped with leaves</td>
<td>3</td>
<td>[0 .. 100]</td>
</tr>
<tr>
<td>12</td>
<td>Leaf Size</td>
<td>The size of each leaf, relative to the size of the trunk</td>
<td>3</td>
<td>[0.1 .. 1.0]</td>
</tr>
<tr>
<td>13</td>
<td>Branch Taper</td>
<td>The ratio of the end of a branch to the base of a branch</td>
<td>3</td>
<td>[0.1 .. 1.0]</td>
</tr>
<tr>
<td>14</td>
<td>Branch Proximity</td>
<td>From the tip of the parent branch, the percent of the parent to be used for distributing the sub-branches along the parent.</td>
<td>3</td>
<td>[0 .. 100]</td>
</tr>
</tbody>
</table>

TABLE I
A description of the values represented in the string evolved by GenTree.

random number comes from other parameters in the tree so that the tree is fully described by the bit string.

III. RESULTS

For the experiments reported here, the following parameters were used: Population size of 100, 1000 generations, 60% crossover (60% of parents participate in crossover), 0.001% mutation (the probability of each bit being set to a random 0/1 value). Elitism was used to ensure that the best tree survives into the next generation.

The systems yields a wider variety of plant shapes than had originally occurred to us, including evergreen shapes, shrub shapes, and cactus-like shapes, as well as many forms that have a more distant resemblance to natural plants. We are pleased with the wide variety of shapes that are possible in the system and its ability to yield some trees that are approaching realism.

Because the leaves are currently rendered as cubes, the leafless trees are the most realistic looking trees.

As a practical concern, 1000 generations of viewing 100 trees cannot be achieved in a timely manner if the user is assigning a fitness to every tree. However, the user need not complete 1000 generations to find interesting trees, and is likely to find some appealing trees in as little as 5-10 gen-
drawtree
  calculate convergence of trunk and branches from branch taper parameter
  draw trunk
  calculate points where branches will be connected, based on branch proximity
    parameter and number of branches parameter
  drawbranch

drawbranch
  if (depth = tree depth parameter)
    drawleaf, based on leafiness parameter
  return
  for each branch
    calculate length, based on length ratio parameter
    calculate width, based on width ratio parameter
    calculate angle, based on branch angle parameter

Fig. 4. High-level pseudocode for rendering trees from bitstrings.

Fig. 5. Three trees with sparse leaves.

Fig. 6. Another tree with sparse leaves, a shrub, and trees resembling an evergreen, a cactus, and a flowering plant.
eralations. Also, it is possible to advance generations without assigning fitnesses. The result of doing so is not quite the same as random tree generation, since the population contains a biased sample of the possible trees, corresponding to a space more likely to be favored by the user. Furthermore, high-polygon trees are likely to be excluded as parents in the GA process, so there is a slight effect of fitnesses on the parent-selection process.

Depending on user preferences, a wide variety of trees may be evolved from these GA parameters. Example trees are shown in Figures 5, 6, 7, and 8. Among the plant shapes evolved are the following:

1. Trees with dense foliage. In these trees, the individual branches are largely obscured by the leaves.
2. Trees with sparse foliage. In these trees, the branches are generally skinny and long.
3. Trees with no foliage. Again, the branches are generally slim and long.
4. Shrub-like trees, with dense leaves.
5. Evergreen-like trees, with a conical shape.
6. Cactus-like plants, with thick branches that grow almost vertically.
7. Flower-like plants. An example of this is the fifth picture in Figure 6, which has green stalks and red “leaves”, which somewhat resemble flower heads.

IV. CONCLUSIONS AND FUTURE WORK

It is difficult to effectively evaluate a system whose output is judged on aesthetic criteria. However, the work reported here is a fascinating illustration of the potential for the GenTree system to evolve trees that rival the realism of three-dimensional tree models constructed painstakingly with tools.
such as 3D Studio Max or Maya.

One of the intriguing facets of the GenTree system is the interplay between the interactive process and the evolutionary process. The user can use fitnesses to trim the search space for the first several generations and then step back and let crossover and mutation explore (with effectively random parent selection) the restricted search space. For example, this approach would be particularly helpful if trying to evolve a set of similar but distinct trees. Allowing users to interactively adjust parameters on a particular tree would extend this facet of the system even further, allowing an expedient way to “fine tune” a promising tree.

The current system is able to generate plants suitable for fanciful 3D virtual worlds; additional work should yield a system that is able to generate more realistic trees and plants. To work toward this end, the current system has a number of limitations that need to be addressed.

The current representation uses 53 bits, which yields $2^{53}$ different possible trees, and yet still does not achieve the level of realism that we would like to achieve. There are several reasons for this, including the use of cubes to render the trees, an overly restricted set of tree parameters, and an overemphasis placed on the color of the trees.

The current rendering of the GA parameters is very primitive, dividing branch lengths into a series of cubes. The cubes are relatively straightforward to compute, and were, therefore a convenient starting point for the system. However, more elegant rendering could be achieved by using conic sections instead. This would require more complex calculations to determine the triangular surfaces that compose each section. However, the visual appeal of the resulting trees would likely be much higher. Furthermore, such trees would have lower polygon counts. In other words, the trees would be significantly more realistic by improving the rendering engine, even if the GA facet of the system were left unchanged.

Additional parameters in the GA string would be beneficial. For example, the random number that determines the branching angle could instead be an independently evolved parameter. Furthermore, for greater realism of trees, different branches should have slight variations, rather than following the same rules. This might be realized, for example, by a variable-length GA string.

It is not clear that evolving the color of the trees along with the shapes is beneficial. It appears that the color and the shape of trees are separable concerns. Color could be handled either external to the GA process or by a second GA process. However, it must be noted that the inclusion of color yields some serendipitous events, such as pink-leaved trees that appear to be cherry trees in blossom.

Additional benefits could be seen from using other shapes for leaves, and subjecting this shape to evolution. In addition, different textures could be used for bark and for leaves, and the textures used could be subject to evolution. Flowers and fruit could also be added as parameters.

References