A Genetic Programming Application in Virtual Reality

Sumit Das, Terry Franguiadakis, Michael Papka,
Thomas A. DeFanti, Daniel J. Sandin
Electronic Visualization Laboratory
University of Illinois at Chicago
851 S. Morgan St., Rm. 1120
Chicago, IL, 60607-7053
mpapka@eecs.uic.edu

Abstract

Genetic programming techniques have been applied to a variety of different problems. In this paper, the authors discuss the use of these techniques in a virtual environment. The use of genetic programming allows the authors a quick method of searching shape and sound spaces. The basic design of the system, problems encountered, and future plans are all discussed.

1 Introduction

1.1 Background

Genetic algorithms have been used to solve a wide variety of problems, such as ones discussed in [8, 9, 10]. Used as an optimization technique, genetic algorithms have proven to be an effective way to search extremely large or complex solution spaces. Two such spaces are the vast domains of shape and sound. There exists an infinite variety of shapes, and sounds that can be associated with them. The search for pleasing combinations of the two cannot be carried out exhaustively, and aesthetic values cannot easily be parameterized. Since genetic algorithms do not rely on problem-specific knowledge, they can be used to discover solutions that would be difficult to find by other methods.

A *genotype* is a "blueprint" for a potential solution to a problem. The solution that results from following the directions in the genotype is called the *phenotype*. Genotypes can be altered or combined to create a set of variations. The resulting set of genotypes is used to create a set of phenotypes. A *fitness function* is used to evaluate the relative quality of each phenotype, and the genotypes corresponding to the phenotypes judged "best" are used as the basis for the next generation.

1.2 Graphics and Genetic Algorithms

William Latham and Stephen Todd explore the possibilities of shape in their use of *Mutator* [15]. Latham and Todd use basic geometric shapes and a simple set of rules for combining them to construct complex three-dimensional images and animations. The rules are empirically based on organic shapes, such as horns and shells. The use of genetic algorithms simplifies the search for pleasing forms [16]. Providing a selection of shapes, Mutator allows the artist to choose the most pleasing ones, which then survive to produce the next generation. This allows a large variety of shapes to be viewed very quickly. The artist has control over the mutation rate; by raising the rate at the outset and lowering it as the results become more desirable, the search can be optimized, steered, and fine-tuned.

Karl Sims has also used genetic programming to aid in the discovery of new visual images. Sims makes use of symbolic LISP expressions to generate images, shapes, and solid textures [14]. The symbolic expressions are then mutated to generate new images based on aesthetic choice. Both Sims' and Latham/Todd's work make use of the user's aesthetic judgement as the fitness function for the image.

1.3 Sound and Genetic Algorithms

The generation of musical sound is usually separated into timbre generation [8, 17] and composition [9, 17]. *Timbre* generation is analogous to selecting or creating instruments to play, while *composition* is the process of deciding what notes the selected instruments will play. There are well known algorithms for timbre generation, such as FM [3] and granular synthesis [17]. Finding a set of parameter values that will satisfy a listener or composer's requirements is a formidable task, and is usually accomplished by trial and error. Horner describes a technique for deriving FM parameters to approximate a desired sound using genetic algorithms [8]. The higher-level requirements of composition are also difficult to quantify and specify. Horner and Goldberg describe the application of genetic algorithms to thematic bridging, a compositional technique in which a repeating initial musical pattern is transformed into a target pattern over a specified duration [9].

1.4 Virtual Reality

Virtual reality (VR) is a method of interacting with a computer-simulated environment. One of the major attributes of VR is immersion; that is, to give a user the experience of actually "being there." Sandin defines a VR system as containing a substantial portion of the following: surround vision, stereo cues, viewer-centered perspective, real-time interaction, tactile feedback, and directional sound [13].

The CAVE (CAVE Automatic Virtual Environment) is a virtual reality project being developed at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago (UIC). It makes use of four projectors displaying computer images on three walls and the floor of a ten-foot cube. Images are projected in stereo, so that

a user wearing stereo glasses can see the images in true three-dimensional space. The user is tracked using an electromagnetic tracking system, so his/her instantaneous position and orientation are known. This allows the environment to be rendered in correct viewer-centered perspective. The user is able to manipulate objects within the CAVE using a wand, a three-dimensional analog of the mouse of current computer workstations. The user also gets audio feedback through the CAVE's sound system [1]. This includes facilities for spatial localization of sound sources, as well as the ability to play back sampled sounds through a MIDI interface. Facilities for software sound synthesis are under development. A more detailed discussion of the CAVE and its applications can be found in [2, 4, 5, 6, 7].

The ability to view and manipulate three-dimensional objects on a two-dimensional computer screen presents some inherent problems. It can be difficult to comprehend a complex three-dimensional shape on a computer screen. Also, viewing the object from different angles and distances can be cumbersome with mouse and keyboard based interfaces. The three-dimensional nature of the CAVE immediately solves these problems. This paper describes an application of genetic algorithms in virtual reality in which we populate a virtual world with both images and sound. This application, which was demonstrated as part of the CAVE installation at Supercomputing '93, presents one solution for developing a genetically evolving environment of shape and sound.

2 Genetic Art in Virtual Reality

2.1 System Overview

This application attempts to remedy some of the above problems, inherent in workstation interface technology. Objects are displayed in stereo, so shapes are easier to perceive. Viewers can examine objects in the same ways as in the real world: by walking around and moving relative to the objects, and also by manipulating objects to get a better look at them. Objects can be selected and then acted upon. Depending on the current mode, the selected object can be:

- Manipulated (e.g. rotated and moved)
- Mutated
- Mated to another object (also known as crossover)
- Saved to a file for future reference
- Refined (re-rendered much larger and with higher grid resolution, the significance of the grid resolution is discussed in section 2.3)

In the CAVE, a user is presented with four shapes and a status window which displays information on the current mode of the application and the action available to the user. The wand has three buttons, which are assigned functionality based on common conventions in systems using a three-button mouse. At any time, holding the middle button down allows the selected object to be manipulated. The right button steps through the possible modes. These changes are reflected in the status window, which shows an icon and brief text describing the available action. When the status window displays the desired action, the left button performs the action on the selected shape. Any shape can be selected simply by pointing to it with the wand. While a shape is selected, its associated music plays. Sound is also used in a more traditional way; for example, to provide feedback on choices the user makes and on the internal state of the application program.

2.2 System Implementation

The genetic structure is largely based on Sims' work [14], where the genotype is a symbolic mathematical expression. This has two advantages: it allows procedural information as well as parameter data to be encoded in the genotype, and the genotype is not limited to any particular size or structure. The expressions are composed of a small function set containing mathematical and geometric functions:

plus, minus, times, divide, cylinder, sphere, torus, union, intersection, complement, difference.

Each function takes a specified number of arguments and returns a real value. The entire expression

$$E(x, y, z)$$

where x, y, and z are interpreted as coordinates in a Cartesian space, also returns a real value. Both mutation and crossover are used to derive the next generation of objects. Mutation is a reproduction technique where a chosen parent's expression is randomly altered, deriving four children which replace the current population (Figure 1). Crossover is the combination of two expressions, with no mutation. Each expression in the new generation contains a part of both parents' expressions (Figure 2). Once again the new population replaces the old population. Both means of reproduction are extensions of techniques discussed in [10, 14], here implemented in C++.

2.3 Shape Generation

The shape of the object represents its graphical phenotype. Each shape is created as a polygonal isosurface within a cubical volume embedded in Cartesian space. A threshold value T is chosen (arbitrarily fixed at $T = 1.0$). The expression E is evaluated at discrete points within the volume. These points are arranged in a regular three-dimensional grid. For a point $P = (X, Y, Z)$, if $E(X,Y,Z)$ is within a tolerance ε of T, then P lies on the surface of the shape. The actual polygonization of the surface from the values is accomplished using the marching cubes algorithm [11], which given a three-dimensional grid of values and a threshold, generates a set of triangles representing the isosurface at that threshold.

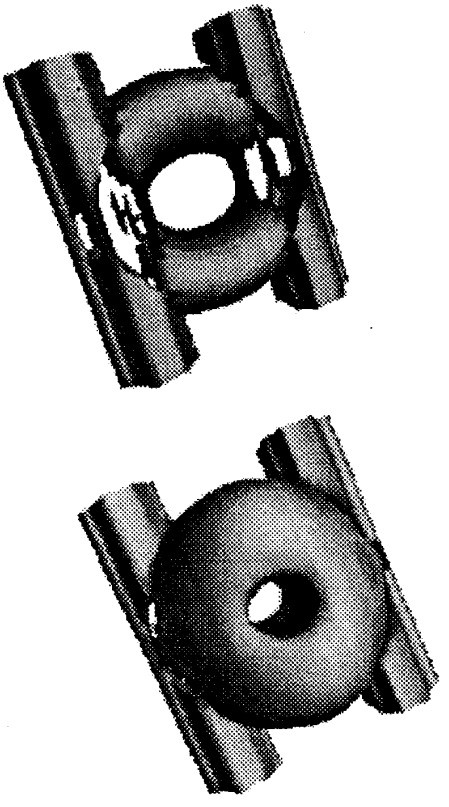

Figure 1: Mutation. The parent is displayed on the top and one of the four children is displayed on the bottom.

By adjusting the resolution of the grid, the smoothness and accuracy of the surface can be improved at the cost of increased computation and rendering time.

Example: The expression

$E(x, y, z)$: $X^2 + Y^2 + Z^2$

with

$T = 4.0$

creates a sphere centered at (X, Y, Z) with radius 2. The expression is evaluated at each (X, Y, Z) location corresponding to a grid point in the volume. The resulting surface exists everywhere that

$-\varepsilon < E(x, y, z) - 4.0 < \varepsilon$

so that a polygonal approximation to a sphere is produced.

The accuracy of the approximation depends on the resolution of the grid used. Normally, all four shapes are generated using a 15x15x15 grid. When refined, only the

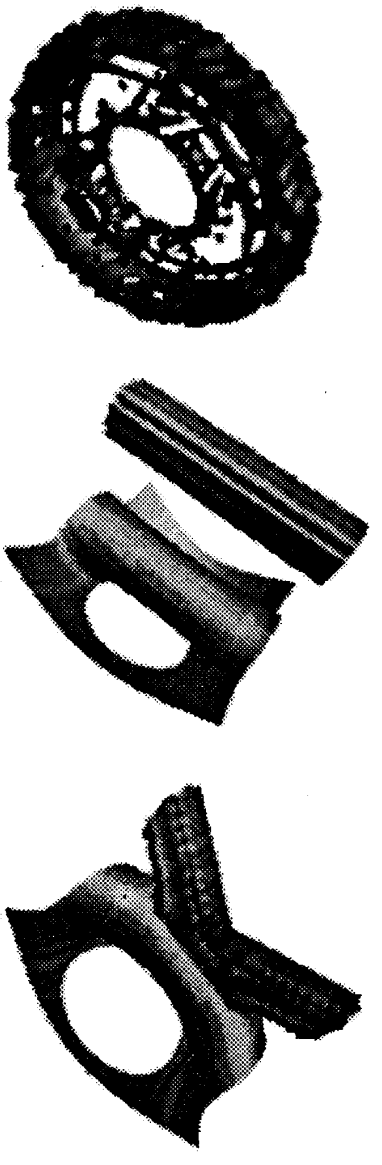

Figure 2: Mating. The parents are displayed on the top and one of the four children is displayed on the bottom

selected shape is displayed, generated using a 60x60x60 grid. When a mutation or mating is performed, all four current objects are replaced by four children resulting from the operation. The initial expressions are generated randomly or by an equation defined by the user, perhaps saved in the course of a previous exploration.

2.4 Shape Representation

To make the surfaces more visually interesting, and also to make the shapes more comprehensible, the two sides of each surface are colored differently. One side is left a solid medium gray. Solid noise [12] is used to create the surface texture for the other side. The expression that generates the shape can also be used to generate coloring schemes for one side of the surface.

The function set described above is a subset of the original set. In the authors' opinion, this subset yielded the most interesting shapes. The mutation rate is also important; if it is too high, the children may have little resemblance to the parent(s). If the rate is too low, there may be no perceptible change from one generation to the next.

2.5 Sound

The goal of the audio portion of this application is to vary stylistic elements across shape. In effect, each shape has a musical style associated with it. This style is the sonic phenotype. The music is played by a band composed of four MIDI instruments: percussion, bass, marimba, and guitar. The music is generated by manipulating precomposed fragments of music, each four beats long. Each fragment is a musical *phrase* to be played by one of the four instruments. The *phrases* are derived in the following manner: A number of 8-bar (32-beat) *compositions* are created for the band in common musical styles, e.g. reggae, waltz, minimalist. Each composition is separated into four parts, one for each instrument. Each of these parts is then divided into eight four-beat phrases. For any given shape, each instrument is given eight phrases to choose from. To provide some overall stylistic cohesiveness, these eight phrases can come from at most two compositions. Each instrument chooses the next phrase to play based on a Markov chain, a probabilistic finite state automaton [17]. The probability tables for the Markov chains are generated by repeated evaluations of the expression E, using empirically derived values for the arguments. The state variables that affect the behavior of each instrument and their mappings to the genotype and visual phenotype are as follows:

- Choice of fragments: number of triangles in the isosurface, modulus number of fragments available.
- Markov chain probability tables: derived from expression **E** (see above).
- Musical key to play in: surface area of shape.
- Tempo: number of triangles in isosurface, scaled to available tempo range.
- Probability of rhythmic perturbation of a note: square root of (number of triangles / surface area).
- Probability of pitch perturbation of a note: surface area of shape.

3 Conclusion and Future Work

This application is a work in progress; it is constantly being revised and enhanced, and there are still many improvements that can be made. The fact that there are only four shapes visible at any one time is a compromise to allow real-time rendering. Although interesting results are obtained, this limitation makes it difficult to follow a desired evolutionary path, as the genetic pool is so small. This problem can be solved by lowering the grid resolution of the objects or by using more powerful computers. Unfortunately, lower grid resolution results in shapes with less detail.

For purposes of clean implementation and predictability, the function set is severely restricted. A larger population coupled with a richer function set might improve on the visual results.

Many of the restrictions imposed on the sonic phenotype were decided on empirically. The goal throughout is to make the sounds enjoyable and comprehensible to the casual listener, while at the same time making the styles distinct from each other. The music is based on well-known styles of music. Nevertheless, as the phrases combine in various ways, styles emerge that seem distinct from any of the underlying compositions.

Current hardware limitations preclude generating the instrument sounds functionally; instead, sampled sounds are used. Since timbre is one of the major factors of stylistic musical classifications, this is a major limitation. The next version of this application will have facilities to generate at least some of its sounds through direct software synthesis, allowing a much greater range of timbre and interactivity.

The instruments' modifications to the musical fragments are limited to statistically driven perturbations of individual notes. A better solution might be to use a more context-dependent (neighboring notes and states of other instruments), musically-based modification scheme. The musical fragments themselves could also be generated functionally, although encompassing a wider range of styles while remaining "musical" to most listeners would be difficult.

The authors are currently exploring the possibility of extending this work to include behavioral characteristics for the objects, as well as non-user-driven fitness functions. This would necessitate a larger population so that statistical variation would not dominate the actual fitness of genes.

The use of supercomputers for this application is being explored, both for isosurface generation and expression evaluation. The CAVE hardware configuration currently supports a high-speed FDDI/HIPPI connection between the SGI Onyx and the Thinking Machines CM-5, and connections to other supercomputers are being developed. An FDDI/HIPPI prototype connection between the CAVE/SGI Crimson and a 64-processor CM-5, implemented by NCSA, was demonstrated at Supercomputing '93 in Portland, OR.

An interesting area we plan to investigate is *meta-evolution*, such as the evolution of fitness functions, or the evolution of the genotype-to-phenotype mapping. Also, instead of using the entire genotype expression to determine both shape and sound characteristics, we want to explore the possibility of using different parts of the expression to determine these and future characteristics. These may or may not be mutually exclusive parts of the expression.

4 Acknowledgments

This material is based upon the work supported by National Science Foundation under Grant No. CDA-9303433, which includes support from the Advanced Research Projects Agency. The authors thank Maxine Brown, Gary Lindahl, Dana Plepys, and Maggie Rawlings for all their help, and all the other members of the Electronic Visualization Laboratory.

5 References

[1] R. Bargar and S. Das, "Sound for Virtual Immersive Environments," notes from course #23, *Applied Virtual Reality*, SIGGRAPH 1993, pp 4.1 - 4.18.

[2] R. Bargar and S. Das, "Virtual Sound Composition for the CAVE," (Tokyo, Japan, September 1993), *Proceedings of the 1993 International Computer Music Conference*, International Computer Music Association, September, 1993 (addendum).

[3] J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", *Journal of the Audio Engineering Society*, 1973, Vol. 21, pp. 526-534.

[4] C. Cruz-Neira, J. Leigh, M. Papka, C. Barnes, S. Cohen, S. Das, R. Engelman, R. Hudson, T. Roy, L. Siegel, C. Vasilakis, T.A. DeFanti, and D.J. Sandin, "Scientists in Wonderland: A Report on Visualization Application in the CAVE Virtual Reality Environment," *IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, October 1993, pp. 59-66.

[5] C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti, "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," *Computer Graphics (Proceedings of SIGGRAPH 93)*, ACM SIGGRAPH, August 1993, pp. 135-142.

[6] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart, "The Cave: Audio Visual Experience Automatic Virtual Environment," *Communications of the ACM*, Vol. 35 No. 6, June 1992, pp. 65-72.

[7] T.A. DeFanti, D.J. Sandin, and C. Cruz-Neira, "A 'Room' with a 'View'," *IEEE Spectrum*, October 1993, pp. 30-33.

[8] A. Horner, J. Beauchamp, and L. Haken, "FM Matching Synthesis with Genetic Algorithms," *Proceedings of the 1992 International Computer Music Conference*, CMA, San Fransisco, 1992.

[9] A. Horner, and D. E. Goldberg, "Genetic Algorithms and Computer-Assisted Music Composition," *Technical report CCSR-91-20*, Center for Complex Systems Research, The Beckman Institute, University of Illinois at Urbana-Champaign, December 1991.

[10] J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[11] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics (Proceedings of SIGGRAPH 87)*, ACM SIGGRAPH, August 1987, pp. 83-91.

[12] K. Perlin, "A Hypertexture Tutorial", notes from course #23, *Procedural Modeling and Rendering Techniques*, SIGGRAPH 1992, pp. 3.1 - 3.28.

[13] D. J. Sandin, "Virtual Reality Technologies: Head Mounted Displays, Booms, Monitor and Projection Based Systems," notes from course #23, *Applied Virtual Reality*, SIGGRAPH 1993, pp 2-1 - 2-5.

[14] K. Sims, "Artificial Evolution for Computer Graphics," *Computer Graphics (Proceedings of SIGGRAPH 91)*, ACM SIGGRAPH, August 1991, pp. 319-328.

[15] S. Todd, and W. Latham, "Artificial Life or Surreal Art?, Towards a Practice of Autonomous Systems," *Proceedings of the First European Conference on Artificial Life*, 1991, MIT Press, Cambridge, Massachusetts, 1991, pp. 504-513.

[16] S. Todd, and W. Latham, *Evolutionary Art and Computers*, Academic Press, New York, 1992.

[17] I. Xenakis, *Formalized Music*, Bloomington: Indiana University Press, 1971.