# The Evolution of 3D Procedural Textures

**Adam Hewgill**
Department of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
ahewgill@hotmail.com

**Brian J. Ross**
Dept. of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
bross@cosc.brocku.ca

## Abstract

Genetic programming is used to automatically synthesize procedural textures for 3D surfaces. The GP system evolves textures with similar colour and surface characteristics as training textures sampled on 3D models. The texture language includes mathematical operators, colour and noise functions, as well as surface information for the point being rendered, such as coordinates, normal, and gradient. Experiments successfully generated procedural textures that displayed visual characteristics similar to target textures.

## 1 INTRODUCTION

This paper briefly summarizes work in (Hewgill and Ross 2003), in which genetic programming is used to evolve procedural textures for 3D surfaces[1]. A procedural texture is a colour generating function that computes the colour of some point in 2D or 3D space (Ebert *et al.* 1998). For example, a 2D procedural texture will take as input the X and Y coordinate of a pixel, and compute the RGB colour for the pixel. 3D procedural textures are similar defined for 3D space. Procedural textures define colours for all points in a scene, without the seams and tiling effects that arise with bitmapped textures. Furthermore, a variety of realistic phenomena are readily modeled by procedural formulae, such as stone, marble, and wood. Procedural textures can be complicated, however, and a great deal of technical expertise and trial-and-error experimentation is required to create a desired effect. Such expertise is beyond the ability of most casual users.

The implementation of computer support tools to aid in the synthesis of procedural textures is a worthwhile goal. In particular, evolutionary computation has been investigated as a tool for procedural texture creation (Sims 1993, Rooke 2002), normally using a human as an interactive fitness function. Work in (Ibrahim 1998, Ross and Zhu 2002, Wiens and Ross 2002) uses genetic programming to evolve 2D textures using automatic fitness scoring via feature analysis. Although 2D procedural textures are applicable to 3D surfaces, the results are often unsatisfactory because the 2D formulae do not account for 3D surface features. The rendered results will usually obscure surface detail, unless external local lighting effects are applied along with the procedural texture.

This research investigates the texture evolution problem for 3D surfaces. The main enhancement required for 3D procedural texture synthesis is the inclusion of surface information in the texture formulae, such as XYZ coordinates, normals, and surface gradients.

## 2 EXPERIMENT

| *Parameter* | *Value* |
|---|---|
| Population size | 1000 |
| Generations | 200+ |
| Runs/experiment | 10 |
| Initialization | ramped half&half |
| Initial ramped tree depth | 5 to 10 |
| Max. tree depth | 17 |
| Crossover rate | 0.9 |
| Mutation rate | 0.1 |
| Selection scheme | tournament (size 5) |

Table 1: Genetic Programming Parameters

The strongly-typed lilGP 1.1 system is used (Zongker and Punch 1995). The target language is designed for procedural texture generation. Terminals include surface coordinates, surface normals, surface gradients (localized normal deviation), and ephemeral constants.

[1] http://www.cosc.brocku.ca/Department/Research/TR/cs0306.pdf

Functions include standard mathematical operators, sine, cosine, min, max, if-then-else, average, and noise. Another function combines 3 floating point expressions into a triple, which is then interpretable as an RGB colour. Strong typing is used for differentiating floating-point expressions and 3-channel RGB expressions. Other GP parameters are in Figure 1.

Fitness scoring evaluates a candidate texture on a predefined training set. This training set is obtained manually from a target texure defined on some given 3D surface. The RGB distance is determined between the generated texture colour and the desired training colour for a surface point with a given coordinate, normal, and gradient. The overall fitness score is the cumulative sum of the RGB distances for all examples. A low score indicates a close correspondence to the training set.

A run's final solution texture is given to a ray-tracer, which uses it to render a set of test images, each using a different 3D model.

## 3 RESULTS

See (Hewgill and Ross 2003) for colour images of the results. A number of different textures were evolved, using the following training textures:

1. A cube with sides having primary colours.
2. A mountainous terrain with natural colouring: white peak, grey cliff, and green base.
3. A natural terrain as above, but swap the blue and gradient channel values.
4. Render a female figure with green shirt, pink face and arms, blonde hair, purple pants, black shoes.
5. Render a surface to have a red top, green sides, and white base.
6. A blue cube with coloured orthogonal stripes.

Experiments 1, 2, 4, and 5 evolved textures that closely corresponded to the given target texture. Experiment 5 was particularly interesting, for it proved that genetic programming was able to evolve textures that corresponded to particular surface orientations of interest. Although the results of experiment 3 were difficult to intuit or predict, the end results were textures with a glowing mineral-like quality. A good training match was challenging for experiment 6. Nevertheless, the resulting textures were glimmering and opalescent in appearance.

## 4 CONCLUSIONS

Automatic 3D procedural texture synthesis is well-suited to genetic programming. As is commonly found with artistic applications of evolutionary computation, we discovered that raw fitness scores are not necessarily the best indicator of the aesthetic suitability of a result. Rather, fitness gives evolution a direction to explore. A number of parallel runs should then be used, and the final explored results can be compared and judged. This points to the fact that user interaction may still play a useful role alongside automatic synthesis.

## Acknowledgments

This research is supported by NSERC Operating Grant 138467-1998.

## References

Ebert, D.S., F.K. Musgrave, D. Peachey, K. Perlin and S. Worley (1998). *Texturing and Modeling: a Procedural Approach*. 2 ed.. Academic Press.

Hewgill, A. and B.J. Ross (2003). Procedural 3D Texture Synthesis Using Genetic Programming. Technical Report CS-03-06. Brock University, Dept. of Computer Science.

Ibrahim, A.E.M. (1998). GenShade: an Evolutionary Approach to Automatic and Interactive Procedural Texture Generation. PhD thesis. Texas A&M University.

Rooke, S. (2002). Eons of Genetically Evolved Algorithmic Images. In: *Creative Evolutionary Systems* (P.J. Bentley and D.W. Corne, Eds.). pp. 330–365. Morgan Kaufmann.

Ross, B.J. and H. Zhu (2002). Procedural Texture Evolution Using Multiobjective Optimization. Technical Report CS-02-18. Brock University, Dept. of Computer Science.

Sims, K. (1993). Interactive evolution of equations for procedural models. *The Visual Computer* **9**, 466–476.

Wiens, A.L. and B.J. Ross (2002). Gentropy: Evolutionary 2D Texture Generation. *Computers and Graphics Journal* **26**(1), 75–88.

Zongker, D. and B. Punch (1995). *lil-gp 1.0 User's Manual*. Dept. of Computer Science, Michigan State University.