Genetic Terrain Programming
An Aesthetic Approach to Terrain Generation

Miguel Frade
ESTG – Instituto Politécnico de Leiria
Morro do Lena, Alto do Vieiro, Apartado 4163, 2411-901 Leiria, Portugal
+351 244 820 300
mfrade@estg.ipleiria.pt

F. Fernandez de Vega
Universidad de Extremadura
C/Sta. Teresa de Jornet, 38, 06800 Mérida – Badajoz, Spain.
+34 924 387 068
fcofdez@unex.es

Carlos Cotta
Universidad de Málaga
ETSI Informática (3.2.49), Campus de Teatinos, 29071-Málaga, Spain
+34 952 137 158
ccottap@lcc.uma.es

Abstract

Nowadays there are a wide range of techniques for terrain generation, but are focused on providing realistic terrains often neglecting the aesthetic appeal. The Genetic Terrain Programming technique, based on evolutionary design with Genetic Programming, allows designers to evolve terrains according to their aesthetic feelings or desired features. This technique evolves TPs (Terrain Programmes) that are capable of generating different terrains, but consistently with the same features. This paper presents a study about the perseverance of terrain features of the TPs across different LODs (Levels Of Detail). Results showed it is possible to use low LODs during the evolutionary phase without compromising results and the terrain features generated by a TPs are scale invariant.

Keywords

Genetic terrain programming, evolutionary systems, terrain generator, level of detail.

1. Introduction

Artificial terrain generation techniques are used across a broad range of applications, including computer animation, architecture, virtual reality and video games. This last area is, probably, the one where its use is more prominent. A detailed terrain model involves a huge amount of polygons to be represented, even when considering only the portion of the scene that is visible. Clark suggested [1] using simpler versions of the geometry for objects that had lesser visual importance, such as those far away from the viewer. These simplifications are called Levels of Detail (LODs) and allow adapting structures, such as terrains, to the processing power requirements.

Nowadays there are many techniques for terrain generation (see Section 2), but procedural techniques are one of the most popular among game's designers, mostly due to their speed, ease of implementation and to their ability to create irregular shapes across an entire range of LODs. However, these techniques allow only a confined variety of terrain types and it only allows the generation of real looking terrains. Although this is important, in some areas, such as video games, it might be more relevant designers' creativity. A designer could evolve a terrain accordingly to their aesthetic feelings rather than realism. This can lead to the creation of terrains with an exotic look, but might also increase users' interest on a video game. The GTP (Genetic Terrain Programming) technique [2] allows the evolution of TPs (Terrain Programmes) based on aesthetic evolutionary design with GP (Genetic Programming). For a specific LOD it is known the ability of those TPs to generate different terrains, but with coherent terrain features. However, this property has not been studied across different LODs. This paper analyses the perseverance of terrain features generated by TPs over a range of LODs. This is a desired characteristic by video games' designers and can help to improve performance during the TPs' evolutionary phase.

Section 2 introduces some background about the traditional terrain generation techniques and their main constrains. It is also presented an overview of evolutionary systems applied to terrain generation. Section 3 succinctly describes the GTP technique and Section 4 shows the achieved results. Finally, the conclusions and future work are presented.

2. Background

Although other data structures exist, height maps are frequently used to represent terrains. Formally, a height map is a scalar function of two variables, such that for every coordinate pair *(x,y)* corresponds an elevation value *h*. A well-known limitation of height maps is the inability to represent structures where multiple heights exist for the same pair of coordinates (e.g., caves). Nevertheless, height maps can be used in numerous scenarios, and on top of that, they can be highly optimised in operations such as rendering and collision detection [3].

2.1 Traditional Generation Techniques

Traditional techniques for terrain generation can be categorised into three main groups: (1) measuring, (2) modeling and (3) procedural. Next, we briefly review each of these techniques.

(1) Measuring techniques gather elevation data through real-world measurements, producing so-called Digital Elevation Models[1]. These models are commonly built using remote sensing techniques such as satellite imagery and land surveys. One key advantage of measuring techniques lies in the fact that they produce highly realistic terrains with minimal human effort, although this comes at the expenses of the designer control. In fact, if the designer wants to express specific goals for the terrain's design and features, this approach may be very time-consuming since the designer may have to search extensively for real-world data that meet her targeted criteria.

1 http://rockyweb.cr.usgs.gov/nmpstds/demstds.html

(2) A key advantage of the modeling technique for terrain generation, that departs itself from the other two techniques, lies in its adaptability. In the modeling approach, an human artist manually models or sculpts the terrain morphology resorting to a 3D modeling program (e.g. Maya[2], Blender[3]). The way the terrain is built is different depending on the features provided by the chosen editor, but the general principle is the same. Contrary to the measuring technique, under the modeling approach the designer retains the full control, a characteristic that has its drawbacks: it might force the designer to consume significant time, effort and the resulting terrain is fully dependent on the designers' skills.

Finally, (3) procedural techniques are those in which the terrains are generated programmatically. This category can further be divided into physical, spectral synthesis and fractal techniques. The physical approach aims to simulate real phenomena such as erosion [4], or plate tectonics movements. Physically-based techniques generate highly realistic terrains, but require an in-depth knowledge of physical laws to be properly implemented and used. Another procedural approach is the spectral synthesis. Random frequency data is generated in the frequency domain and then converted into altitudes, in the space domain, by applying the inverse Fast Fourier Transform (FFT). The problem of using this technique for simulating real world terrain is that it is statistically homogeneous and isotropic, two properties that real terrain does not share [5]. Furthermore, it does not allow much control on the outcome of terrains' features. Fractal techniques are based on the self-similarity concept. An object is said to be self-similar when magnified subsets of the object look like the whole and to each other [6]. This allows the use of fractals to generate terrain which still looks like terrain, regardless of the LOD in which it is displayed [7]. This is one of the reasons why fractal techniques are popular among game's designers, besides their speed and ease of implementation. Several tools exist that are predominantly based on fractal algorithms (e.g. Terragen[4] and GenSurf[5]). However, not all terrain types present the self-similarity characteristic. Furthermore, generated terrains by this technique are easily recognised because of the self-similarity pattern and the designer has little control on the resulting terrain features.

2.2 Evolutionary Generation Techniques

Evolutionary algorithms (EA) are a kind of bio-inspired algorithms that apply the Darwin's theory [8] of natural evolution of the species, were living organisms are rewarded through its continued survival and the propagation of its own genes to its successors. There are four main classes of

2 http://www.autodesk.com/fo-products
3 http://www.blender.org
4 http://www.planetside.co.uk/terragen
5 http://tarot.telefragged.com/gensurf

EAs: genetic algorithms (GA) [9], evolutionary strategies [10], genetic programming (GP) [11] and evolutionary programming [12].

To the best of our knowledge, Teong Ong et al. [13] were the first authors to propose an evolutionary approach to generate terrains. Their approach, based on GA, breaks down the terrain generation process into two stages: the terrain silhouette generation phase, and the terrain height map generation phase. A database of height map samples, representative of the different terrain types, is used to search an optimal arrangement of elevation data that approximates the map generated in the first phase.

M. Frade et al. proposed a new evolutionary approach designated GTP (Genetic Terrain Programming) [2]. Their approach consists on the combination of evolutionary art systems with GP to evolve mathematical expressions, designated TPs (Terrain Programmes), to generate artificial terrains as height maps. GTP relies on GP as evolutionary algorithm, which creates computer programs, or mathematical expressions as the solution (represented in a tree form). GP algorithms uses four steps to solve problems: (1) generate an initial population of random compositions of the functions and terminals of the problem; (2) execute each program in the population and assign it a fitness value according to how well it solves the problem, on interactive systems this task is performed by a human; (3) create a new population by copying the best existing solution and creating new individuals by mutation and crossover (sexual reproduction); (4) The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of GP [11].

3. Genetic Terrain Programming

The GTP technique [2] consists of a guided evolution, by means of Interactive Evolution, accordingly to a specific desired terrain feature or aesthetic appeal. This technique can yield both aesthetic and real terrains and is capable of generating different terrains, but consistently with the same features. Furthermore, by way of resorting to several TPs to compose the full landscape, it is possible to control some localised terrain features, thus eliminating the main drawback of traditional procedural techniques. The combination of GP with evolutionary art systems also diminish the effort and time required to create complex terrains, relatively to modeling techniques and the results that are not dependent on the designer's skills.

In GTP the first population is created randomly, with initial trees depth size limited to 20 and a fixed population size of 12. The number of generations is decided by the designer, who can stop the application at any time. The designer can select one or two individuals to create the next population. Like in others IEC systems, the fitness function relies exclusively on designers' decision, either based on his aesthetic appeal or on desired features. The individuals of a population are represented as trees composed by nodes and terminals. The tree nodes can contain any of the functions in Table 1 and the possible terminals are presented in Table 2. Contrarily to other GP implementations, where the terminals are scalar values, in GTP the terminals are two-dimensional matrices which represent height maps.

Table 1 – GP functions

Name	**Description**
plus(a,b)	plus: *a + b*
minus(a,b)	minus: *a - b*
multiply(a,b)	multiply: *a x b*
sin(a), cos(a), tan(a), atan(a)	trigonometric functions
myLog(a)	returns *0*, if *a=0* and *log(|a|)* otherwise
myPower(a,b)	returns *0* if *a^b* is *NaN*, *Inf*, or or has imaginary part, otherwise returns *a^b*
myDivide(a,b)	returns *a* if *b=0*, otherwise returns *a/b*
myMod(a,b)	is *0* if *b=0*, otherwise returns *mod(a/b)*
mySqrt(a)	returns *sqrt(|a|)*
negative(a)	returns *-a*
FFT(a)	returns 2-D discrete Fourier Transform
smooth(a)	circular averaging filter with *r=5*

Table 2 – GP Terminals

Name	**Description**
rand(s)	map with random hights between 0 and 1
fftGen(s)	spectral synthesis based hight map, whose spectrum depends on a REC
gauss(s)	gaussian bell shape hight map, whose wideness depends on a REC
plane(s)	flat inclined plane hight map whose orientation depends on a REC
step(s)	step shape hight map whose orientation depends on a REC
sphere(s)	semi-shpere hight map whose radius depends on a REC

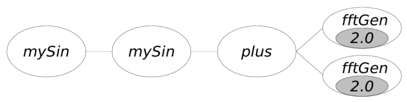


Figure 1 – Example of a TP in tree form.

Most terminals depend upon a Random Ephemeral Constant (REC) to define some characteristics. Figure 1 presents an example of a TP in tree form with two REC values represented in grey ellipses within the terminals.

In GTP the 12 individuals of the population must be executed during the interactive evolutionary phase to be evaluated by a designer, which will choose the TPs for the next generation. This means that using high LODs on this phase will consume more time and the application will be less responsive. The LOD is controlled through the terminals' variable *s* during the TP execution. The axis values in the terminals' functions are discrete with regular intervals and the variable *s* controls the spacing between axis values by specifying the height map grid size, which covers a predefined area. The greater the *s* value is, the lesser is the distance between each grid point and greater is the LOD.

4. Results

An experiment was conducted to test the perseverance of terrain features across several LODs and the consequent impact in generation times for our evolutionary tool *GenTP* (developed with GPLAB[6], an open source GP toolbox for Matlab[7]). A set of TPs was chosen to generate terrains with grid sizes from 50 to 450. Figure 2 presents the results of the execution of four different TPs at three LODs with grid sizes of *50x50*, *150x150* and *450x450*. The first row corresponds to TP1, the second to TP2 and so on. TP1 (with 8 nodes) and TP2 (with 17 nodes) were evolved by their aesthetic appeal and the TP3 and TP4 were evolved with a terrain feature in mind. A mountain in TP3 (with 13 nodes) and a volcano in TP4 (with 7 nodes). In this experience all TPs have preserved their main features independently of the chosen grid size. Due to terminals' randomness consecutive calls of the same terminal will always generate a slightly different height map. This is a desired characteristic, but it can be controlled for a specific LOD, by fixating the random number seed. However, this approach does not work for generating terrains at different LODs, because the amount of necessary random numbers will vary accordingly with the chosen LOD. This explains the differences from terrains at different LODs generated by the same TP. Figure 3 shows the average time of 10 execution of each TP at each grid size on a Pentium Core 2 Duo at 1,66 GHz with 2 GB of RAM. As expected the generation time increases at a quadratic pace with the increase of the number of grid points, e.g. for TP4 from *18,4 ms* at *50x50* to *1066,0 ms* at *450x450*. The generation time also increased, as anticipated, with the number of TP's nodes. These results show us that it is possible to evolve TPs with low LOD and consequently less computation time, thus improving the response time of our tool, without affect the designers' judgement about the terrains features of the selected TP. Anyhow, through the analyse function implemented in our tool, it is possible to select the desired grid size to inspect the coherence of terrain features of a TP across 8 consecutive executions.

TP1*=myLog(myLog(myMod(myLog(fftGen(s,3.75)),myLog(myLog(fftGen(s,4.25))))))*

6 http://gplab.sourceforge.net/
7 http://www.mathworks.com/

TP2*=myPower(cos(myDivide(myLog(smooth(fftGen(s,2.75))),myMod(sin(fftGen(s,0.50)),myDivide(myLog(smooth(fftGen(s,2.75))),myMod((sin(fftGen(s,0.50))),fftGen(s,2.25))))))*

TP3*=times(sin(fftGen(s,3.00)),smooth(times(sin(cos(sin(cos(times(fftGen(s,1.75),fftGen(s,0.75)))))),fftGen(s,0.50))))*

TP4*=plus(fftGen(s,3.00),smooth(myMod(gauss(s,0.75),cos(fftGen(s,1.00)))))*

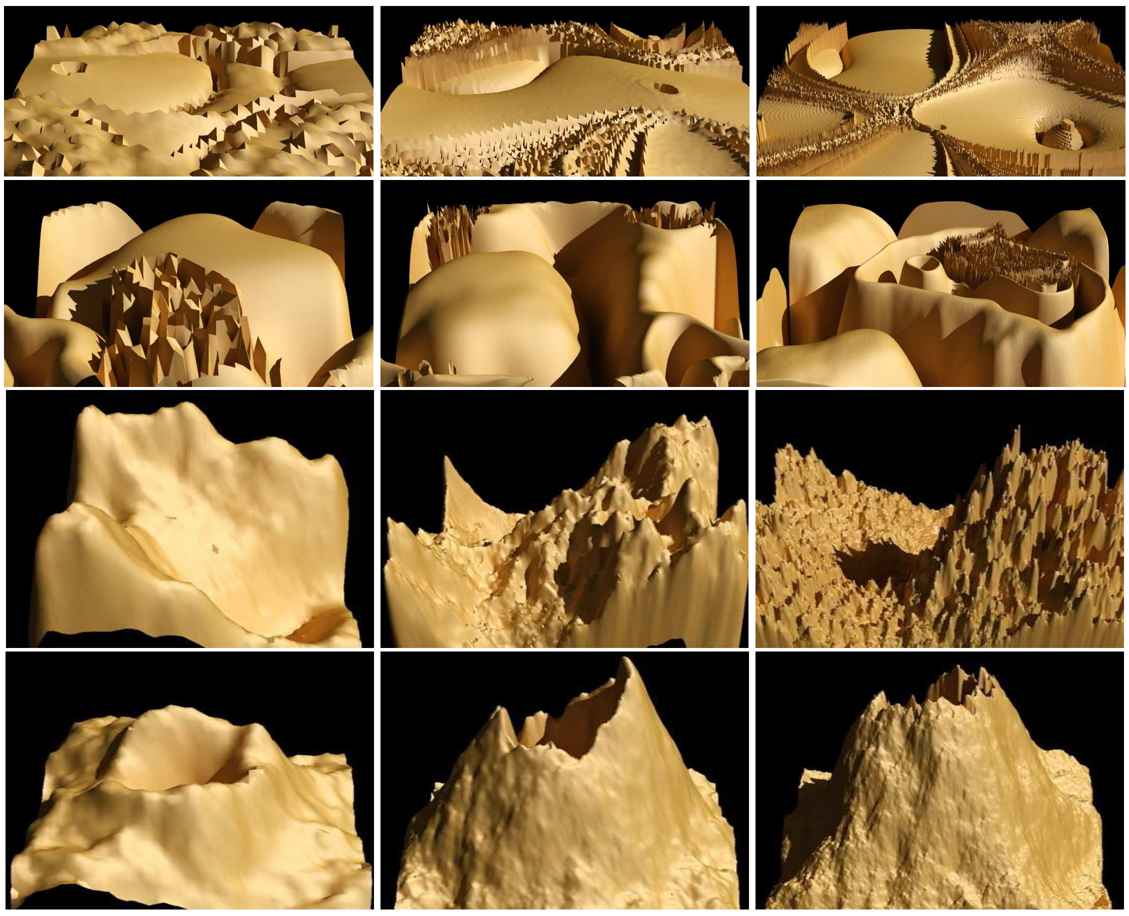
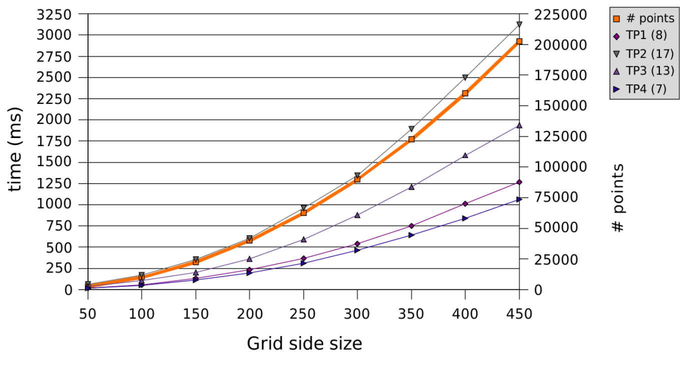


Figure 2 – Each row as an example of a TP executed with a grid size of 50x50, 150x150 and 450x450.



Grid	Time (ms)			
size	TP1 (8)	TP2 (17)	TP3 (13)	TP4 (7)
50	18,5	61,4	37,7	18,4
100	59,6	171,9	106,9	53,9
150	136,4	355,1	205,3	113,1
200	233,2	605,7	362,9	198,6
250	368,3	962,5	590,1	312,6
300	539,8	1342,3	878,7	466,8
350	753,5	1889,8	1210,3	643,4
400	1012,7	2501,1	1583,6	841,3
450	1269,8	3122,1	1939,1	1066,0

Figure 3 – Terrain generation time versus grid size

Conclusion

This paper presented the GTP technique which allows the evolution of TPs to produce terrains accordingly to designers' aesthetic feelings or desired features. Through a series of experiments we have shown that the feature perseverance is true independently of the chosen LOD. This means that during the evolutionary phase low LODs can be used without compromising the result. Consequently less time will be required for our evolutionary tool, enabling it to be more responsive, which is an important characteristic on interactive tools. Additionally, the resulting TPs can be incorporated in video games, like any other procedural technique, to generate terrains, with the same features, independently of the chosen LOD. Furthermore, this technique offers two levels of control regarding randomness: a specific TP will always generate terrains with the same features; and the seed for the random number generator can be kept the same across separate runs, allowing the same terrain to be regenerated as many times as desired.

The TPs' scale invariance showed in our results preludes the implementation of a zoom feature. Fixating the the random number generator seed is not enough to implement this feature due to the variation of the amount of necessary random numbers accordingly with the zoom. Besides some terminals, like *rand* and *fftGen*, are not based on continuous functions. Another future work will be the inclusion of more features in our technique in order to generate full landscapes including vegetation and buildings.

References

[1] J.H. Clark. Hierarchical geometric models for visible-surface algorithms. Proceedings of the 3rd annual conference on Computer graphics and interactive techniques, pages 267-267, 1976.

[2] Miguel Frade, F. Fernandez de Vega, and Carlos Cotta. Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience. In EvoWorkshops 2008, volume 4974, pages 485-490, Napoli, Italy, 2008. Springer.

[3] Mark Duchaineau, Murray Wolinsky, David Sigeti, Mark Millery, Charles Aldrich, and Mark Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In VIS '97: Proceedings of the 8th conference on Visualization '97, pages 81-88, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[4] Alex Kelley, Michael Malin, and Gregory Nielson. Terrain simulation using a model of stream erosion. In SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, pages 263-268, NY, USA, 1988. ACM.

[5] Jacob Olsen. Realtime procedural terrain generation - realtime synthesis of eroded fractal terrain for use in computer games. Department of Mathematics And Computer Science (IMADA), University of Southern Denmark, 2004.

[6] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. Chaos and Fractals - New Frontiers of Science. Springer, 2nd edition, 2004.

[7] Richard Voss. Fractals in nature: characterization, measurement, and simulation. SIGGRAPH, 1987.

[8] C. Darwin. On the Origin of Species by Means of Natural Selection. John Murray, 1859.

[9] J.H. Holland. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, 1975.

[10] T. Bäck, F. Hoffmeister, and H-P. Schwefel. A survey of evolution strategies. Proceedings of the Fourth International Conference on Genetic Algorithms, page 2-9, 1991.

[11] J. R. Koza. Genetic programming. on the programming of computers by means of natural selection. Cambridge MA: The MIT Press., 1992.

[12] L. Fogel, A. Owens, and M. Walsh. Artificial intelligence through simulated evolution. Wiley, 1966.

[13] Teong Joo Ong, Ryan Saunders, John Keyser, and John J. Leggett. Terrain generation using genetic algorithms. In GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, pages 1463- 1470, NY, USA, 2005. ACM.