Evolution of Artificial Terrains for Video Games Based on Accessibility

Miguel Frade[1], F. Fernandez de Vega[2], and Carlos Cotta[3]

[1] Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Portugal
mfrade@estg.ipleiria.pt
[2] Centro Universitario de Mérida, Universidad de Extremadura, Spain
fcofdez@unex.es
[3] ETSI Informática, Campus de Teatinos, Universidad de Málaga, Spain
ccottap@lcc.uma.es

Abstract. Diverse methods have been developed to generate terrains under constraints to control terrain features, but most of them use strict restrictions. However, there are situations were more flexible restrictions are sufficient, such as ensuring that terrains have enough accessible area, which is an important trait for video games. The Genetic Terrain Program technique, based on genetic programming, was used to automatically evolve Terrain Programs (TPs - which are able to generate terrains procedurally) for the desired accessibility parameters. Results showed that the accessibility parameters have negligible influence on the evolutionary system and that the terminal set has a major role on the terrain look. TPs produced this way are already being used on *Chapas* video game.

Key words: genetic terrain programming, artificial terrains, video games

1 Introduction

Generating artificial terrains is an important topic in computer graphics, specially in video games where its application is probably more prominent. Some of the most popular techniques are fractal algorithms. These algorithms are the favorite ones by game designers, mainly due to their speed and simplicity of implementation. However, procedural techniques, like fractals, have also their own limitations [5]. The main disadvantage is the difficulty of modelling with them. It is very difficult or impossible to know how to modify them to achieve a certain local effect.

Diverse methods have been developed to produce terrains under constraints in attempt to control terrain features. The technique in [12] consists in generating a terrain model by computing the interpolation of point clouds or contour lines. The authors of [10] managed to obtain good approximations of downsampled elevation datasets using radial basis functions. Other methods use patches [13], small-scale [3] or microscopic [4] features of existing terrains to synthesize new ones that satisfy the user global constraints. The method presented in [11] deforms an initial fractal terrain by applying local modifications to satisfy a set

of various fixed constraints. A constrained fractal model based on midpoint displacement algorithm is presented in [1]. However, most of these methods are focused on the reconstruction of Data Elevation Models and use only height constraints, besides many of them suffer from either time and/or manipulation complexity.

To use height constraints the designer must specify both the height value and its location. However, there might be situations were less strict restrictions are preferable. For instance, a designer might want a nearly flat, or mountainous terrain, without caring about the specific location or size of terrain features. A two step approach, designated Genetic Terrain Programming (GTP), was proposed in [6], [7] that allows the generation of terrains with the desired features or aesthetic appeal without formal restrictions. The first step consists of interactive evolution, with genetic programming, of mathematical expressions designated TPs (Terrain Programs). On the second step TPs are computed to generate height maps. Due to the interactive nature of this approach, from now on it will be abbreviated as GTP_i. Nevertheless, there are terrain features that are better evaluated by computers due to their numerical nature. Besides, the search for a terrain with very specific features might be a tiresome endeavor on interactive evolutionary applications [2].

Accessibility is a crucial structural feature that highly influences the attainable movements of characters/avatars around the video game terrain, as well as the potential placement of buildings and other structures. The author of [9] employs accessibility restrictions to generate terrains, but the used technique may not fully meet the constraints. On this paper it is presented a new version of GTP that performs automatic evaluation of terrains based on accessibility constraints, which will be designated from now on as GTP_a.

Section 2 introduces some background about the generation of height maps from TPs. The fitness function and the reasoning behind it are also detailed on section 3. A set of tests were conducted and the used parameters, as well as its results are showed in Section 4. Finally, the conclusions and future work are presented on Section 5.

2 Background

An important characteristic of procedural techniques is their ability to generate a scene with the required resolution and zoom level. This is probably the main advantage of the procedural techniques over other techniques. In spite of the procedural nature of TPs, the implementation in GTP_i only allowed control over terrain resolution, not zoom level. This is a limitation that runs against the procedural advantages. Therefore, in GTP_a both the terminal and function sets were modified in order to generate TPs that permit also the control over zoom. The function set used in GTP_i was simplified to remove all functions that prevented the zoom feature, Table 1 shows the GTP_afunction set.

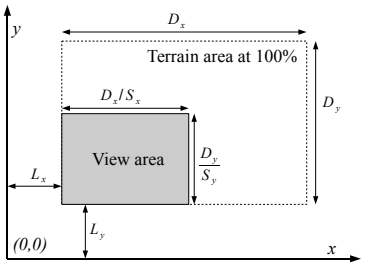
Three different terminal sets are used on this work to evaluate its influence on the resulting terrains: $T_1 = \{noise(x, y), rec\}$, $T_2 = \{x, y, rec\}$ and

Table 1. GP function set

Name	Description
$plus(a,b)$, $minus(a,b)$, $multiply(a,b)$	arithmetical functions
$sin(a)$, $cos(a)$, $tan(a)$, $atan(a)$	trigonometric functions
$exp(a)$	returns e^a
$myLog(a)$	returns 0 if $a = 0$ and $log(\|a\|)$ otherwise
$myPower(a,b)$	returns 1 if $b = 0$, 0 if $a = 0$ and $\|a\|^b$ otherwise
$myDivide(a,b)$	returns a if $b = 0$ and $a \div b$ otherwise
$mySqrt(a)$	returns $\sqrt{\|a\|}$
$negative(a)$	returns $-a$

$T_3 = \{noise(x,y), x, y, rec\}$, where rec stands for *random ephemeral constant*. The $noise(x,y)$ terminal is a stochastic function that is commonly used in fractals. Its ideal properties are [5]: have a repeatable pseudorandom output based on its inputs variables x, y; the output range is known, namely from -1 to 1; when analyzed in the frequency domain, the output is band-limited, with a maximum frequency of about 1; the noise function should not exhibit obvious periodicities or regular patterns; the noise function is stationary, that is, its statistical character should be translationally invariant; and the noise function is isotropic, that is, its statistical character should be rotationally invariant.

TPs are able to generate an unlimited continuous surface. In order to create an height map from a TP, the continuous surface must be sampled with fixed increments of x and y to obtain the corresponding altitude z, where $z = f(x,y)$, $x, y, z \in \mathbb{R}$. The altitudes values are stored in matrix H, whose size $n_r \times n_c$ depends on the amount of samples and therefore define the height map resolution. Equation (1) shows the relation between the height map matrix H and the TPs continuous function obtained with GTP_a. $h_{r,c}$ represents the elevation value for row r and column c, and D_x, D_y are the terrain dimensions. S_x, S_y allow the control of the zoom level and L_x, L_y allow us to localize the origin of the terrain view area (see Fig. 1).



$$h_{r,c} = f\left(\frac{c \times \frac{D_x}{n_c - 1}}{S_x} + L_x, \frac{r \times \frac{D_y}{n_r - 1}}{S_y} + L_y\right) .$$
$$r \in \{1, \cdots, n_r\}, c \in \{1, \cdots, n_c\}, \quad (1)$$
$$D_x, D_y, S_x, S_y \in \mathbb{R}^+ \text{ and } L_x, L_y \in \mathbb{R}$$

Fig. 1. Terrain view area

3 Accessibility Score Fitness Function

Movement and structure placing is often restricted to low slopes, therefore, it is necessary to analyze the changes in declination of the terrain. For that purpose the slope map S is defined to store the declination for each cell r, c of the height map H. The slope values are calculated as the magnitude of the gradient vector (tangent vector of the surface pointing in the direction of steepest slope) [8]. With this approach, the slope is computed at a grid point, which depends on the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ of the height map function $z = f(x, y)$. The most common approximation for partial derivatives is a weighted average of the elevation differences between the given point and all points within its 3×3 neighborhood [8]. The estimate of the partial derivatives for cell z_5 (see Fig. 2) are given by (2) and (3), were Δx and Δy are the height map distances between each cell.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Fig. 2. Neighbor positions

$$\frac{\partial f}{\partial x} \approx \frac{(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)}{8\Delta x} \ . \quad (2)$$

$$\frac{\partial f}{\partial y} \approx \frac{(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)}{8\Delta y} \ . \quad (3)$$

Player units must be able to move around the terrain and a sufficient number of nearly flat areas must exist for building placement. Therefore, cells with a declination below a certain threshold allowing unit movement should be connected in a large area. To analyze the slope map according to this criteria an accessibility map A is created with the same size of the height map. $A = \{a_{r,c}\}_{c \leq n_c}^{r \leq n_r}$ is a binary map whose cell values depend on the threshold S_t, with $a_{r,c} = 0$ if $s_{r,c} < S_t$ or $a_{r,c} = 1$ if $s_{r,c} \geq S_t$.

The nearly flat areas, whose $s_{r,c} < S_t$, that are not connected to the main area will be inaccessible, so the next step is to search for the biggest connected area in A with a component labeling algorithm. This algorithm returns the amount of connected areas and the number of cells contained within each one. Then, the smaller connected areas are removed from the accessibility map. Next we classify the terrains accordingly to the accessibility value υ given by (4), where $n_r n_c$ is total amount of cells in the height map and C_a is the number of cells contained on the biggest connected area of A. The formula in (4) was built to be minimized, so the smaller value of υ the larger the accessibility area is.

$$\upsilon = \frac{n_r n_c}{C_a}, \quad C_a \neq 0 \ . \quad (4)$$

The accessibility criteria alone would make a completely flat terrain the best fit. However, a completely flat terrain does not add realism or interest to the

terrain and does not provide obstacles to units movement, which is undesirable. To prevent this, we defined the accessibility score υ_s, in (5). The biggest connected area is limited by the threshold υ_t, where $p \in [0, 1]$ is the desired area for the biggest connected area. The *ceil* function is used to ensure that the amount of desired cells for the biggest connected area is round up to the nearest integer value. This way it will be possible for υ_s to achieve the exact value of zero and stop the evolutionary process. Otherwise we would have to stipulate a tolerance value within which υ_s would be considered close enough to zero to stop the evolutionary process. However, the tolerance value would be dependent from the chosen resolution for the height map, which is undesirable.

$$\upsilon_s = |\upsilon - \upsilon_t| , \quad \text{where} \quad \upsilon_t = \frac{n_r n_c}{\lceil p n_r n_c \rceil}, \quad p \neq 0 \ . \tag{5}$$

4 Tests and Results

A battery of tests was conducted to access the features and realism of terrains produced by terminal sets T_1, T_2 and T_3 evaluated by the accessibility score. We want also to assess the impact of different percentages of accessibility area, thus for each terminal set three different percentages were used: $p = 70\%$, $p = 80\%$ and $p = 90\%$. There are a total of 9 different tests that were performed on a Pentium Core 2 Duo at 2.0 GHz with 1 GB of RAM running GNU/Linux. All tests were conducted with the accessibility threshold set to $S_t = 10\%$ and the parameters shown on Table 2. On the left side of Table 2 are the GP system parameters and on the right side are the parameters used to compute height maps from TPs.

Table 2. GP and height map parameters

GP	Value	Height map	Value
maximum generations	50	n_r and n_c	128
population size	500	L_x and L_y	0
initialization method	half and half	S_x and S_y	1
	ramped from 2 to 6	D_x and D_y	10
max. depth	17		
selection operator	tournament, size 7		
crossover operator	rate 70%		
mutation operator	rate 30%		

Our main goal are TPs and the terrains they generate, Fig. 3 shows three examples of height maps as gray-scale images (one for each terminal set) and their respective accessibility maps. The second row shows the same terrains, but computed in 3 dimensions with Blender 3D (`www.blender.org`) and rendered

with YafaRay (`www.yafaray.org`). These terrains were chosen because they were the ones that presented the shortest TPs, which are displayed in Fig. 4.

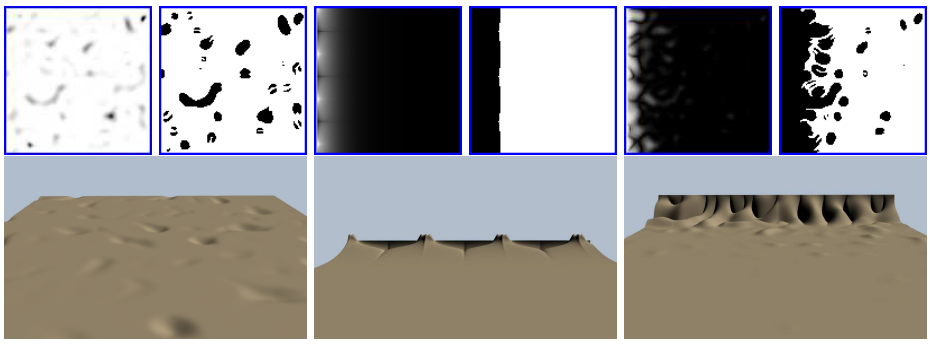


Fig. 3. Examples of height maps (as gray-scale images) and their respective accessibility maps on the first row. The first pair of images were obtained with T_1 (see TP_1 on Fig. 4), the second pair with T_2 (see TP_2) and the third pair with T_3 (see TP_3).The second row has the terrains rendered in 3D.

TP_1 = *sin(sin(cos(myPower(multiply(myNoise(x,y),myNoise(x,y)),minus(exp(atan(multiply(myNoise(x,y), myNoise(x,y))))),sin(myNoise(x,y)))))))*

TP_2 = *cos(atan(minus(exp(x),myDivide(multiply(0.07358,0.93756),myDivide(x,myLog(myLog(sin(y)))))))))*

TP_3 = *multiply(myPower(myNoise(x,y),x),cos(atan(x)))*

Fig. 4. Terrain programs that generated the terrains on Fig. 3.

The next step was to visually compare and analyze all the terrains produced by the resulting TPs of each terminal set. Our analysis focused on three terrain properties: suitability (to video games), realism and features diversity. The terrains obtained from terminal set T_1 were the ones that showed more usefulness with several obstacles distributed around the terrain area. They also present smooth transitions between the different terrain features and without visible pattern repetition, which give them a natural look. However, apart from a few exceptions, terrains tend to be very similar, like if the locations of the hills and valleys were simply shifted or resized. On the opposite side are terrains produced by terminal set T_2. Terrains tend to exhibit geometric patterns and sudden height changes, which gives them an unnatural look. The origin of the terrain view area seems to have a big impact, because $L_x = L_y = 0$, a value were the discontinuities of our protected functions happens. This suggests the further study of T_2 with a different view area origin values. Terrains obtained with T_2

showed more diversity across the 20 runs than T_1 terrains, but T_2 terrains tend to present a single large obstacle, which makes them less suitable. Our richest terminal set, T_3, was the one that brought out more diverse terrains. On these terrains geometric patterns are occasionally visible, but with much less visual impact than with T_2. Regarding realism T_3, some terrains look more natural than others and a few have also an odd look. To sum up, we classify our terminal sets (from better to worst) regarding realism: T_3, T_1 and T_2; regarding diversity: T_3, T_2 and T_1; finally, regarding suitability: T_1, T_3 and T_2.

The GP system returned also the number of generations, TP size (amount of nodes), TP depth (tree depth) and run time of each run. This data is summarized in Table 3 where the values shown are the average for 20 runs (with different seeds) and the correspondent standard deviation of the mean inside round brackets. The fitness value is not present on this table, because all runs were able to reach the goal of $v_s = 0$.

Table 3. Average values (and standard deviation of the mean) for 20 runs of GTP_a

Terminal set	T_1			T_2			T_3		
p	70%	80%	90%	70%	80%	90%	70%	80%	90%
Generations	8.25	8.80	8.00	7.95	8.00	9.10	8.40	7.30	7.65
	(0.49)	(0.39)	(0.43)	(0.53)	(0.45)	(0.64)	(0.48)	(0.61)	(0.44)
TP size	44.75	44.35	44.20	48.20	50.25	61.70	39.10	30.80	40.90
	(2.33)	(4.51)	(4.57)	(3.97)	(4.22)	(5.55)	(4.27)	(3.48)	(4.56)
TP depth	13.40	13.15	12.70	12.65	12.30	14.35	11.85	9.30	11.60
	(0.60)	(0.63)	(0.84)	(0.70)	(0.59)	(0.56)	(0.93)	(0.80)	(0.95)
Run time (s)	99.52	112.26	102.82	93.17	88.78	102.87	97.57	77.86	77.78
	(8.50)	(6.05)	(8.68)	(12.56)	(7.84)	(13.47)	(8.93)	(9.12)	(6.46)

Regarding the TPs sizes and depths what stands out is that for terminal set T_1 these values are almost constant for all 3 values of p, for terminal set T_2 they tend to increase with p and that terminal set T_3 is the one that has smaller values. These results show that the richer terminal set T_3 is able to produce shorter and therefore simpler TPs than the other terminal sets. Run times are also smaller for terminal set T_3, except for $p = 70\%$ which is similar to the other terminals. Run times for terminal set T_1 are slightly longer, which was expected because $noise(x, y)$ is far more complex than the other terminals.

The average number of generations across all tests are very similar with values ranging from 7.30 to 9.10. This means that the system is able to reach a fitness value of zero easily and that there are many solutions to reach that goal. The diversity of TPs to have a perfect fitness value is very interesting, because we want to be able to generate a width range of terrains that fit our goal. This raised the question: how different are the resulting terrains from each other when only

the seed is changed? To help answer this question we compared each accessibility map with the other 19 from the 20 runs of each test. The comparison consists in measuring how much inaccessible (black) area overlaps, as shown in Fig. 5. An overlap value of 100% would mean that the maps were equal.

Fig. 5. Comparison of two accessibility maps obtained with T_1 (first and center images) and the resulting overlap (right image) - the amount of overlap is 50.07%.

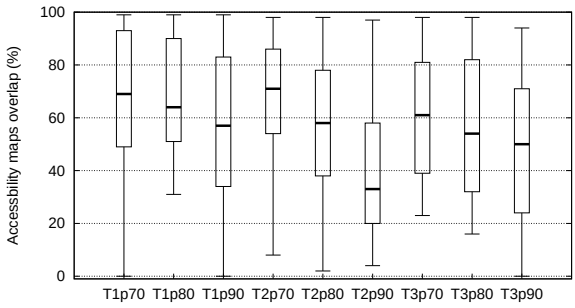


Fig. 6. Boxplot of accessibility maps overlap for each test

For each test the amount of comparisons is the result of combinations without repetitions of $\binom{20}{2} = 190$. The higher quantaties of comparisons returning lower overlap values the bigger are the differences between accessibility maps by changing only the seed. The boxplot on Fig. 6 show that the higher the value of p is, the less the accessibility maps tend to overlap, independently of the used terminal set. This is an expected result, because the inaccessible areas are smaller the larger p is, so there are less probability of overlap. The overlap values from maps obtained with terminal set T_1 tend to be higher than the other ones, which means that terrains will be more alike with each other, which corroborates our visual inspection. For terminal set T_2 the amount of overlap is highly influenced by the value of p, presenting the higher values for $p = 70\%$ and the lowest for $p = 90\%$. Finally, for terminal set T_3 there are lower chances to get similar terrains, which is an advantage in our case. Nevertheless, none of the comparisons, among all tests, returned overlap values of 0% or 100%.

TPs produced by GTP$_a$ are already being used on the video games Chapas, which is still in development (see Fig. 7 and `http://tr.im/chapas`). To generate a terrain with 1024×1024 cells, the fastest TP (from T_3) took 0.1 seconds and the longest (from T_1) took 1.1 seconds. Although these times are in the same magnitude of the ones obtained by [1], they can be further improved because TPs are easily parallelizable without requiring any interprocess communication.

Fig. 7. Screenshot of the *Chapas* video game with a terrain generated by a TP.

5 Conclusions

A new version of GTP, GTP$_a$, as been presented that performs automated evaluation of TPs based on accessibility parameters. These parameters allow us to control the slope threshold, that differentiate the accessible from the inaccessible terrain areas, and how much area should be made accessible. Through a series of experiments we have shown that our fitness function, the accessibility score, is able to produce many solutions that fit our goal. Furthermore, the required number of generations is not influenced by changing the terminal set or the amount of desired accessible area. The terminal set can have a great influence on the terrain look, with T_3 producing more realistic terrains, but also some odd looking terrains. T_3 showed us the importance of rich terminal sets to achieve shorter and simpler TPs. T_1 produced the better suitable terrains for a video game. However, our fitness function does not account for terrain realism, requiring a visual inspection before using them. With regard to the previous version of GTP, the new approach allowed to remove the human factor from the evolutionary process and this way avoid designer fatigue. The usefulness of this technique is shown by its integration on the Chapas video game.

Terrain view area is defined by several parameters and some of them, like L_x,L_y, have influence on the result. Further tests must be conducted to access the level and importance of their impact. Slope is just one metric from many that geomorphology uses to classify height maps of real terrains. Other metrics, such

as convexity, can be incorporated on the fitness function to allow finner control over terrain features. Multi-objective optimization techniques can be used in this context. Finally, in spite of our interesting results, we have not found yet the perfect terminal set to enable GTP_a to achieve a widther range of real looking terrains types. Further research on this topic is needed.

Acknowledgments. The second author is supported by National Nohnes project TIN2007-68083-C02-01 of Spanish MS. The third author is supported by project TIN2008-05941 of Spanish MICINN. Thanks are due to the reviewers for their detailed and constructive comments.

References

1. Belhadj, F.: Terrain modeling: a constrained fractal model. In: 5th International conference on CG, virtual reality, visualisation and interaction in Africa. pp. 197–204. ACM, Grahamstown, South Africa (2007)
2. Bentley, P.: Evolutionary Design by Computers. Morgan Kaufmann Publishers, Inc., CA, USA (1999)
3. Brosz, J., Samavati, F.F., Sousa, M.C.: Terrain synthesis by-example. In: First International Conference on Computer Graphics Theory and Applications (2006)
4. Chiang, M., Huang, J., Tai, W., Liu, C., Chang, C.: Terrain synthesis: An interactive approach. In: International Workshop on Advanced Image Tech (2005)
5. Ebert, D., Musgrave, K., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach. Morgan Kaufmann, 3rd edn. (2003)
6. Frade, M., de Vega, F.F., Cotta, C.: Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience. In: et al., M.G. (ed.) Applications of Evolutionary Computing. LNCS, vol. 4974, pp. 485–490. Springer, Napoli, Italy (2008)
7. Frade, M., de Vega, F.F., Cotta, C.: Breeding terrains with genetic terrain programming - the evolution of terrain generators. International Journal for Computer Games Technology 2009(Article ID 125714), 13 (2009)
8. Horn, B.: Hill shading and the reflectance map. Proceedings of the IEEE 69(1), 14–47 (1981)
9. Olsen, J.: 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)
10. Pouderoux, J., Gonzato, J.C., Tobor, I., Guitton, P.: Adaptive hierarchical rbf interpolation for creating smooth digital elevation models. In: GIS '04 - 12th annual ACM international workshop on Geographic information systems. pp. 232–240. ACM, New York, NY, USA (2004)
11. Stachniak, S., Stuerzlinger, W.: An algorithm for automated fractal terrain deformation. Computer Graphics and Artificial Intelligence 1, 64–76 (2005)
12. Vemuri, B., Mandal, C., Lai, S.H.: A fast gibbs sampler for synthesizing constrained fractals. Visualization and Computer Graphics, IEEE Transactions on 3(4), 337–351 (Oct-Dec 1997)
13. Zhou, H., Sun, J.: Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics 13(4), 834–848 (2007), member-Turk, Greg and Member-Rehg, James M.