Automatic evolution of programs for procedural generation of terrains for video games: Accessibility and edge length constraints

Article *in* Soft Computing · November 2012

DOI: 10.1007/s00500-012-0863-z

CITATIONS
4

READS
57

3 authors:

Miguel Frade
Instituto Politécnico de Leiria
13 PUBLICATIONS **109** CITATIONS
SEE PROFILE

Francisco Fernández de Vega
Universidad de Extremadura
191 PUBLICATIONS **1,128** CITATIONS
SEE PROFILE

Carlos Cotta
University of Malaga
322 PUBLICATIONS **2,794** CITATIONS
SEE PROFILE

Soft Computing
DOI 10.1007/s00500-012-0863-z

Automatic Evolution of Programs for Procedural Generation of Terrains for Video Games

Accessibility and Edge Length Constraints

Miguel Frade · Francisco Fernandez de Vega · Carlos Cotta

Abstract Nowadays the video game industry is facing a big challenge: keep costs under control as games become bigger and more complex. Creation of game content, such as character models, maps, levels, textures, sound effects and so on, represent a big slice of total game production cost. Hence, the video game industry is increasingly turning to procedural content generation to amplify the cost-effectiveness of the efforts of video game designers. However, procedural methods for automated content generation are difficult to create and parametrize. In this work we study a Genetic Programming based procedural content technique to generate procedural terrains that do not require parametrization, thus, allowing to save time and help reducing production costs. Generated procedural terrains present aesthetic appeal; however, unlike most techniques involving aesthetic, our approach does not require a human to perform the evaluation. Instead, the search is guided by the weighted sum of two morphological metrics: terrain accessibility and obstacle edge length. The combination of the two metrics allowed us to find a wide range of fit terrains that present more scattered obstacles in different locations, than our previous approach with a single metric. Procedural terrains produced by this technique are already in use in a real video game.

Keywords Terrains · procedural content generation · video games · aesthetic appeal

Miguel Frade
School of Technology and Management,
Polytechnic Institute of Leiria
Campus 2 - Morro do Lena, Alto do Vieiro; Apartado 4162,
2411-901 Leiria; Portugal
Tel.: +351-244 843 428, Fax: +351-244 820 310
E-mail: miguel.frade@ipleiria.pt

Francisco Fernandez de Vega
Centro Universitario de Mérida, University of Extremadura
C/Sta. Teresa de Jornet, 38; 06800 Mérida - Badajoz; Spain
Tel.: +34-924 387 068, Fax: +34-924 303 782
E-mail: fcofdez@unex.es

Carlos Cotta
ETSI Informática (3.2.49), University of Málaga
Campus de Teatinos, 29071-Málaga; Spain
Tel.: +34-952 137 158, Fax: +34-952 131 397
E-mail: ccottap@lcc.uma.es

1 Introduction

Traditionally, the main techniques used in content development for video games have been artistry. From 3D models to textures, bitmap graphics and sound effects, all game content has been, generally, handcrafted by artists and designers working specifically to that end. This approach assures game developers full control over their creations. Nevertheless, by delegating most or all of the details up to the designer, manual content production impose high requirements on designer in terms of time and effort. This has a huge impact in production costs as games became bigger and more complex (Edwards, 2006). Therefore, game industry is increasingly turning to procedural content generation methods to keep costs under control. Procedural content generation allows parametric control, which provides amplification of designers productivity: a few parameters yield large amounts of details (Ebert et al, 2003).

Procedural generation allows the automation of content creation (Nelson and Mateas, 2007). For instance, it is possible to generate a forest were each plant specie is represented by a set of parameters and each tree is slightly different just by changing the seed for the pseudo-random numbers generator (Lane and Prusinkiewicz, 2002; Prusinkiewicz and Lindenmayer, 2004).

The representation of procedural content is also extremely compact and can be measured in *Kilobytes*, while others require *Megabytes* of storage. One good example of compactness is the classic game *Elite* [1], which succeed to keep 8 galaxies of 256 planets each in a few tens of kilobytes by representing each planet with just a few numbers. Another advantage that some procedural content has is the ability to be computed at any desired resolution. Fractals are a good example of this characteristic (Mandelbrot, 1983). Finally, procedural techniques allow more dynamic processes during the game development cycle. Designers can change the location of some level elements without having to redraw everything else. The procedural content can have rules built in to automatically adapt to those changes.

However, procedural content generation has also its own drawbacks. One disadvantage is its evaluation. This operation requires intense computations and can be very expensive. Procedural methods typically use a set of parametric controls that enable a procedure to generate many different outputs. To make a procedure more useful, additional controls can be added. While the power of a procedure may be enhanced in this way, the resulting interface can become overly complex. In the case of a human using the interface, coming up with good results from a powerful procedure often degenerates into an authoring processing of trial and error. Besides, procedural algorithms present a certain degree of unpredictability: a small change in one parameter can result in big changes in the outcome, or big changes might not result in any significant modification. Whatever is the case, designers end up performing a lot of tests and simulations until they learn how the procedural system behaves to tune it. This search, for the right input parameters and algorithm tune, is time consuming. For example, the development of "Far Cry 2" [2] video game took as much as 15 times more time to refine and tune procedural tools than the amount of time developing the underlying game engine (Remo, 2008). However, if the desired content has characteristics that can be measured then the search process can be automated. That automation can be achieved by Search-Based Procedural Content Generation (SBPCG) techniques (Togelius et al, 2011).

SBPCG techniques apply a *generate and test* approach where procedural content is generated and then tested, according to some criteria. A test function (also called fitness function) grades the procedural content instead of simply accepting or rejecting it. Then, new content is produced, that is dependent on the score of previous content, and this way tries to find better scoring content. The process is repeated until the content is considered good enough. Evolutionary algorithms are a perfect match for this approach, although not the only search mechanism of SBPCG.

In this work we use one SBPCG technique, which we coined as Genetic Terrain Programing (GTP) (Frade et al, 2009b), to generate procedural terrains for video games. GTP utilizes Genetic Programing (GP) as an automated evolutionary search tool for procedural terrains, designated Terrain Programs (TPs). We believe this approach will allow the generation of new terrain types with aesthetic appeal. However, unlike other evolutionary techniques were aesthetic evaluation is performed by humans, our technique relies only on geomorphological metrics. Those metrics are accessibility and obstacles edge length. The evolutionary search of terrains with this characteristics will produce TPs that do not require any parameter input. Therefore, TPs can be integrated in video games without a human performing parameter tuning, thus allowing to save time and money. Another purpose of our technique could be its integration in authoring tools to inspire designer imagination and serve as base of their work. The presented work is the continuation of our previous research (Frade et al, 2010a,b).

Section 2 focus on the importance of terrains in video games, previous research about procedural terrain generation and their different approaches. The GTP technique and the used geomorphological metrics are detailed in Section 3. Section 4 presents the applied test methodology, results and its discussion. Finally, conclusions and future work are laid out in Section 5.

2 Background

The desire for providing the player with novel and engaging content without a large investment in designers resources drives the goal of automatic content generation. Previous work has shown the viability of this approach, some examples can be found in Hastings et al (2009); Nelson and Mateas (2007); Togelius and Schmidhuber (2008); Togelius et al (2011). Terrains are one of the many assets whose generation can be automated. They play a fundamental role in some type of games (Forbus et al, 2002; Smelik et al, 2010; Wells, 2005) and increases game replayability value (Sampath, 2004).

Fractals are the most common procedural method to generate artificial terrains. They are the favorite algorithms of game designers, mainly due to their speed and simplicity of implementation. They offer unlimited extent landscapes and can cover an arbitrarily large area without seams or unwanted pattern repetition (Ebert et al, 2003).

[1] video game published by Acornsoft in 1984
[2] video game published by Ubisoft in 2008

Self-similarity is the key concept behind any fractal technique (Peitgen et al, 2004). This characteristic allows the generation of surfaces regardless of the scale in which they are displayed. However, real terrains present this characteristic only on a limited scale (Goodchild, 1980). Mandelbrot (1983) was the first to realize the similarity between the trace of one dimensional fractional Brownian motion and the contours of mountains peaks. Over the years other fractal algorithms were invented and nowadays there are five different approaches: Poisson faulting (Mandelbrot, 1983; Voss, 1987); Fourier filtering (Mandelbrot, 1983; Mastin et al, 1987; Sakas, 1993; Voss, 1987); midpoint displacement (Miller, 1986); successive random additions (Voss, 1987); and finally summing band-limited noises (also known as noise synthesis) (Miller, 1986; Musgrave et al, 1989; Perlin, 1985).

The statistical behavior of fractals results in maps that present homogeneous features that are noticeable on large scales, which makes them easily recognizable. To address this issue Musgrave et al (1989) introduced a noise synthesis variant that enables some control over fractal dimension to create eroded fractal terrain, referred as multifractal. To increase realism they also resort to physical simulation of erosion. However, erosion simulation is slow and introduces more parameters for the user to control. To alleviate this problem Olsen (2004) proposed several optimizations that sacrifice physical correctness over performance with little visual impact. Other fractal based terrain generation approaches have been proposed by Pabst and Jense (1995) and Pi et al (2006).

All terrain synthesis based purely on fractals, control the output by means of parameters, such as the Holder exponent, fractal dimension, octaves and lacunarity, just to name a few. These parameters impact the generated terrain as a whole and do not allow the specification of features location or their dimensions. Besides, to grasp the effect of each parameter requires a deep understanding of fractal mathematics and/or trial and error experiments until the desired effect is found. This process is time consuming and there is no guarantee the desired features are discovered. To overcome this problem a new set of methodologies have been devised over the years that can be categorized into: (1) synthesis by example of real world data; (2) constrained generation; (3) interactive modification of a base terrain; (4) use of software agents; and finally (5) search based algorithms.

(1) *Synthesis of terrain by example of real world data* - Techniques in this category consist on: extracting features from Digital Elevation Models (DEM); classify them; compose a new terrain with the desired characteristics; and finally smooth the transitions between the different terrain features. This concept is applied by Chiang et al (2005) where an interactive environment was created to synthesize terrains based on microscopic terrain features. They use geometric primitives to build the terrain profile and then a matching procedure is applied to replace them by real world data. Later Tu et al (2008) proposed several improvements to this method. A similar approach is presented by Brosz et al (2006), were small scale characteristics from one real terrain are extracted and applied to a base terrain to increase its detail and resolution. A similar method is described by Zhou et al (2007) where a terrain is generated based on extracted features of a input height-map and a user line drawing that defines the occurrence of large-scale features. Yet another approach is presented by Li et al (2006). Their proposal has four stages: terrain silhouette generation; terrain feature retrieval; region selection and filling; and texture generation. The main advantage of all techniques in this category is the realism of produced terrains. However, they require a suitable set of examples to be able to create all desired terrain features. Although nowadays there are many free sources of real world DEM, building the appropriate data set can be tedious and time consuming.

(2) *Constrained generation* - Control of terrain features can also be attained by imposing constraints, where the process takes into account some restrictions during or after the initial generation phase. There are several methods in this area that can be further sub-categorized into: surface approximations and deformation. Surface approximation methods are commonly used to reconstruct sparse DEM data, or to procedurally amplify DEM resolution (Vemuri et al, 1997). With the same goal Pouderoux et al (2004) managed to obtain good approximations using radial basis functions. A constrained fractal model based on midpoint displacement algorithm is presented by Belhadj (2007). His main goal is to reconstruct DEM's by specifying the exact locations and height of some DEM points. Belhadj and Audibert (2005) use ridge and river networks to constraint the fractal creation of the height map. However, these algorithms do not appear to be controllable as there is no guarantee as to the shape of the terrain between points. A different method is introduced by Szeliski and Terzopoulos (1989), where they apply a surface fitting algorithm using splines that is perturbed by adding fractal detail. However, due to the use of a coarse spline mesh, only large scale modifications are possible. The proposal of Kamal and Uddin (2007) resembles Poisson faulting fractals, it draws straight lines across the base map to create a series of randomly placed polygons in each iteration. A set of locations with desired terrain

feature can be specified and some additional control is provided by means of three additional parameters, but its impact on the resulting height map is not intuitive.

(3) *Interactive modification of a base terrain* - On the demand for easily and intuitively control of terrain features from the user perspective, several methods have emerged based on interactive modification of a base terrain. These methods have one interactive phase were the user can specify major features of maps, but rely on procedural techniques to add the small details. Schneider et al (2006) introduced a real time editor where the user edits the terrain by interactively modifying the base functions of the noise generator by replacing the Perlin noise grid with a set of user-drawn gray-scale images. This approach has the advantages to break the too homogeneous look of large scale fractal terrains. Carpentier and Bidarra (2009) created an application that allows users to paint height-maps directly in 3D view by applying procedural brushes. However, this approach shares some disadvantages of other manual editing methods, such as large amount of memory to store the resulting terrain and final result depends on user skills. Smelik et al (2010) proposes another sketch based approach were users compose a digital sketch of the rough terrain layout. Then, the framework generates a high-resolution map that complies to the specified features at large with high level of detail on a small scale. Although interactivity can be seen as the main strength of these techniques, it is also its main disadvantage because it prevents terrain generation from being fully automated.

(4) *Software agents* - A new approach, based on software agents, has been proposed by Doran and Parberry (2010). Their generator applies agents in three phases: coast line, landform and erosion. The authors claim their approach to be more intuitive and controllable than fractals. However, the quantity of parameters that need to be defined is huge (12 only for the mountains agent) and will require a certain amount of trial and error experiments until the desired result is achieved.

(5) *Search based algorithms* - The main challenge of parametric approaches is to find the right values of parameters that produce the desired terrain features. To address this problem several proposals have been made that relay on search based algorithms to find the right parameters, or generate new procedures, to achieve the desired terrain features. For instance, Stachniak and Stuerzlinger (2005) employ a stochastic local search algorithm that finds an acceptable set of deformation operations to apply to a base terrain in order to obtain a map that approximately adheres to the specified constraints. An evolutionary approach was proposed by Ong et al (2005), where genetic algorithms are used to transform height maps in order to conform them to the required features. Their approach has two stages: the two-dimension terrain silhouette phase, and the terrain height map generation phase. The 2D terrain silhouette and a database of representative height map samples are the only form of control. Ashlock et al (2008) propose co-evolution of L-systems parameters and grammar to fit a specific terrain shape, which has some resemblance to symbolic regression. A different perspective is proposed by Togelius et al (2010b). They apply multi-objective EAs to evolve height maps that fit some user predicted entertainment metrics to hopefully increase players interest on the game. This concept is further developed and applied to StarCraft video game (Togelius et al, 2010a). None of these approaches addresses aesthetic appeal or creativity of the generated terrains.

Our proposal fits the search-based category. Terrain Programs (TPs) are generated and tested according to the weighted sum of two morphological criteria: accessibility and obstacles edge length, were the evolutionary search mechanism in use is genetic programing. *GTP* aims to generate content with aesthetic appeal, a trait common to many research work in the field of Interactive Evolutionary Computation (IEC), like Sims (1991), Unemi (1998) or Stanley (2007), just to name a few. Our technique shares with these examples an indirect phenotype representation, like an expression encoded as a tree or as neural network. However, contrarily to IEC, our goal is to generate aesthetic content without human intervention during the evolutionary process. Once found, TPs can be easily integrated in video games and will not require any input parameter to control its look. To the best of our knowledge, this area has not been address in previous procedural content generation research. The following section details the proposed technique, the fitness function and the reasoning behind it.

3 Genetic Terrain Programming

Our SBPCG technique, Genetic Terrain Programing (GTP), consists on the application of GP (Koza, 1992; Poli et al, 2008) to evolve mathematical expressions, designated TPs, that will generate artificial terrains. Our first implementation of GTP was interactive (GTP_i) (Frade et al, 2008, 2009b). In spite of the good results obtained this way, this approach presented two limitations: user fatigue (a common trait of Interactive Evolutionary Systems) and the lack of ability to perform zoom (more details about the zoom limitation can be found in (Frade et al, 2009a)). To address GTP_i limitations a second version of GTP, designated GTP_a, was

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 $-a$

developed to evaluate and classify TPs automatically. GTP_a removes the human factor from the evolutionary process, that was present in GTP_i, and uses instead a direct fitness function. With GTP_a terrain generation from a given TP is deterministic and will allow designers to search TPs offline (during game development phase) and incorporate them as procedures into a game. In spite of the differences of GTP_a over GTP_i, the goal of generating aesthetic terrains remains. Frade et al (2010a) presented the first study of GTP_a with a fitness function based on terrain accessibility and Frade et al (2010b) showed the results for a fitness function based on obstacles edge length constraints. This paper continues the work of the two previous publications to study GTP_a behavior when both metrics are combined.

The function set used in GTP_a is described in Table 1 and three different terminal sets are used: $T_1 = \{ myNoise(x,y), ERC\}$, $T_2 = \{ X, Y, ERC\}$ and $T_3 = \{ myNoise(x,y), X, Y, ERC\}$. ERC stands for *Ephemeral Random Constant* (Koza, 1992) and $ERC \in [0, 10]$. The terminal $myNoise(x,y)$ is a lattice noise function based on Blender 3D lattice noise primitive $orgBlenderNoise$[3], see Eq. (1). Lattice noise functions, commonly used as fractals primitives, use one or more set of uniformly distributed pseudo random numbers at every integer coordinate point. The intermediate values are calculated using spline interpolation (Ebert et al, 2003). The ideal properties and mathematical details of a noise function are presented by Perlin (1985, 2002). In $myNoise(x,y)$ the input parameters are implicit, so to differentiate them from the explicit parameters X and Y small case is used. Given the way we implemented terminal $myNoise$, it is possible to produce a terrain with only this terminal, Fig. 1 shows a three-dimensional render of such terrain.

[3]Source code available under GNU General Public License at http://www.blender.org/

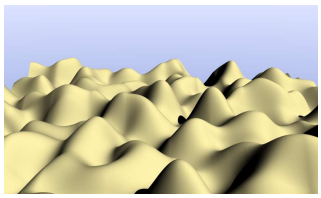

Fig. 1 Terrain generated by $myNoise(x,y)$ with height map parameters specified in Table 5

$$myNoise(x,y) = 2 \times orgBlenderNoise(x,y,0) - 1 \quad (1)$$

We use three different terminal sets to evaluate its influence in the resulting terrains and because TPs generated with them will have different properties. Terrain Programs generated with terminal set T_1 will have only implicit functions. Therefore, it will be possible for a single TP to generate many view areas (see Fig. 2) that share the same morphological look. This can be done by changing L_x and L_y with values big enough to avoid overlapping of viewing areas. This property can be used to simulate randomness, were L_x and L_y would work as seeds. This feature opens the possibility to game developers to offer players with novel, but similar, terrains each time they play and that way increase game replayability value. However, the small amount of terminals in T_1 will restrict terrain diversity. Still, we want to access if the combination of both metrics will have a positive impact in terrain diversity. Two more terminal sets were created: T_2 and T_3. Terminal set T_2 presents standard GP terminals with two variables X and Y and T_3 is the union of the previous terminal sets. Although TPs from both T_2 and T_3 will lack the possibility of generating different view areas with the same morphological look, we want to study their behavior regarding terrain appeal, diversity and if they are more fit for our evaluation function.

3.1 Terrain Programs Evaluation

With GTP_a, the evaluation of TPs will be performed by a fitness function instead of a human. That evaluation will be based on the map they produce, therefore it is required to convert them to a height map. To that end, the continuous surface that a TP can generate must be delimited and sampled to obtain the corresponding altitude h, where $h = f(x,y)$, $h, x, y \in \mathbb{R}$. The altitude values are stored in matrix $H = \{h_{r,c}\}_{c \leqslant n_c}^{r \leqslant n_r}$, whose size $n_r \times n_c$ depends on the amount of samples and therefore define the height map resolution. Equation (2) shows the relationship between the height map matrix H and

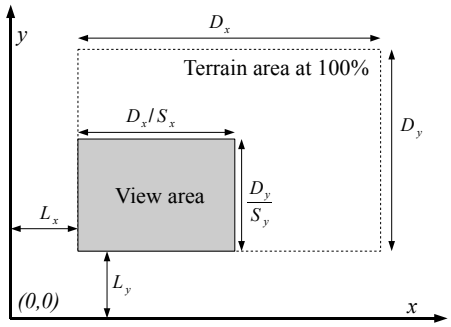

Fig. 2 Terrain view area

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Fig. 3 Neighbor positions

the TPs continuous functions. The value $h_{r,c}$ 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. 2).

$$h_{r,c} = f\left(\frac{c \times \frac{D_x}{n_c - 1}}{S_x} + L_x, \frac{r \times \frac{D_y}{n_r - 1}}{S_y} + L_y\right)$$
$$r \in \{1, \cdots, n_r\}, c \in \{1, \cdots, n_c\}, \quad (2)$$
$$D_x, D_y, S_x, S_y \in \mathbb{R}^+ \text{ and } L_x, L_y \in \mathbb{R}$$

The proposed method for evaluating TPs is based on two morphological metrics: accessibility score (Frade et al, 2010a) and obstacles edge length score (Frade et al, 2010b). The accessibility score aims to generate terrains were a certain percentage of the terrain area is accessible. A part of a terrain is accessible if its slope is under a defined threshold. So, we create the slope map $S = \{s_{r,c}\}_{c \leqslant n_c}^{r \leqslant n_r}$ to store the declination for each cell r, c of the height map H. The slope values are calculated with Eq. (3) and the partial derivatives for cell z_5 (see Fig. 3) are estimated by Eq. (4) and Eq. (5), where Δx and Δy are the height map distances between each cell (Horn, 1981).

$$Slope(\%) = 100 \times \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (3)$$

$$\frac{\partial f}{\partial x} \approx \frac{(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)}{8\Delta x} \quad (4)$$

$$\frac{\partial f}{\partial y} \approx \frac{(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)}{8\Delta y} \quad (5)$$

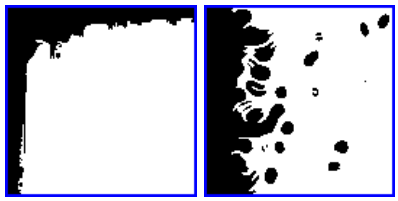

Fig. 4 Example of two accessibility maps using only accessibility score with terminal set T_2 (left) and T_3 (right). The black areas represent terrain obstacles.

Once the slope map S is calculated it is necessary to determine the cells that are accessible. To that end an accessibility map $A = \{a_{r,c}\}_{c \leqslant n_c}^{r \leqslant n_r}$ is created with the same size of the height map. A is a binary map, with either 0 or 1 value in each cell depending on the selected slope threshold. The accessible cells of a terrain should be connected in an large area to allow player units to move around and for building placement. Therefore, we search the biggest connected accessible area in A, recurring to a component labeling algorithm. Then the terrain is evaluated by Eq. (6), where A_+ is the amount of cells that belong to the biggest accessible area.

The accessibility criteria alone would make a completely flat terrain the best fit. However, such terrain does not add realism or interest to the terrain. To prevent this, the accessibility score υ_s, is defined in Eq. (7). The biggest accessible area is limited by the threshold υ_t, where $p_a \in [0, 1]$ is the percentage of desired accessible area. The *ceil* function will allow υ_s to achieve the exact value of zero and stop the evolutionary process. Otherwise we would have to stipulate a tolerance value whose value would be dependent from the chosen resolution for the height map, which is undesirable.

$$\upsilon = \frac{n_r n_c}{A_+}, \quad \text{where} \quad A_+ = \sum_{r=1}^{n_r} \sum_{c=1}^{n_c} a_{r,c}, \quad A_+ \neq 0 \quad (6)$$

$$\upsilon_s = |\upsilon - \upsilon_t|, \quad \text{where} \quad \upsilon_t = \frac{n_r n_c}{\lceil p_a n_r n_c \rceil}, \quad p_a \neq 0 \quad (7)$$

However, as shown by Frade et al (2010a), this metric alone tends to produce terrains with a single or very few obstacles, see Fig. 4. This situation is specially obvious with terminals T_2 and T_3. These terrains are less interesting for video game usage. To address this problem we decided to measure obstacles' edge length and use it also to calculate individuals' fitness. This metric will increase the amount of obstacles and its edge complexity (Frade et al, 2010b). The edge line can be determined through the Laplacian operator over the accessibility map A (Gonzalez and Woods, 2002). This operator can be estimated for cell z_5 (see Fig. 3) by Eq. (8) and whenever it returns a positive value means that z_5 belongs to the edge line and the correspondent cell

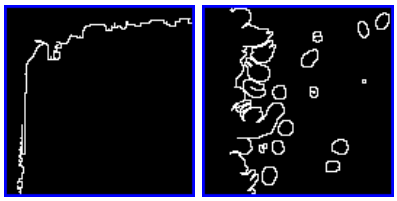

Fig. 5 Edge maps built from the accessibility maps in Fig. 4

$e_{r,c}$, of the binary edge map $E = \{e_{r,c}\}_{c \leqslant n_c}^{r \leqslant n_r}$, is filled with value 1. Fig. 5 shows two examples of edge maps.

$$\nabla^2 f \approx 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9) \quad (8)$$

Based on the amount of cells that belong to the edge, we classify the terrain by the edge value ε defined in Eq. (9), where E_+ is the sum of all cells with $e_{r,c} = 1$.

$$\varepsilon = \frac{n_r n_c}{E_+}, \quad \text{where} \quad E_+ = \sum_{r=1}^{n_r} \sum_{c=1}^{n_c} e_{r,c}, \quad E_+ \neq 0 \quad (9)$$

$$\varepsilon_s = |\varepsilon - \varepsilon_t|, \quad \text{where} \quad \varepsilon_t = \frac{n_r n_c}{\lceil p_e n_r n_c \rceil}, \quad p_e \neq 0 \quad (10)$$

The edge value ε used alone as fitness function, would prevent the formation of large accessible areas. To avoid this problem we defined the edge score ε_s in Eq. (10). This way the edge length is limited by the threshold ε_t, where $p_e \in [0, 1]$ is the desired percentage of edge length in relation to the total terrain area.

We want to access the ability of GTP_a to generate aesthetic terrains without human intervention during the evolutionary process and its applicability to video games. Many computer games using large-scale outdoor terrain, such as real time strategy genre, pose more requirements to terrain shape than visual appearance. Player characters must be able to move around the terrain and there must exist a decent number of flat areas to allow various structures to be placed upon (Olsen, 2004). These considerations can be formalized into two criteria:

- height map cells with an inclination below a certain threshold s allowing unit movement and structures placement should be in an area as large as possible;
- the height map should not be flat, the inaccessible areas should be scattered throughout the terrain to increase aesthetic appeal and hopefully player interest.

To fulfill these criteria we developed the accessibility metric (to ensure large accessible areas) and the edge length metric to guarantee that inaccessible areas have a complex clipping and are dispersed throughout the terrain. A similar reasoning is used by Olsen (2004). However, instead of obstacles edge length he uses slope map standard deviation. We have also made some tests with this metric to find out that it was not adequate to our purposes. Olsen (2004) uses a base terrain and then applies erosion algorithms to help the appearance of both flat areas and obstacles. Due to its nature, these transformations are limited by the base terrain. On the other hand, GTP creates terrains from scratch without any constraints regarding their initial form. Therefore, the GP system was able to easily generate them with the desired standard deviation values by producing stair forms. This was undesired, because it was limiting the appearance of more diverse terrain types.

So far both accessibility score (Frade et al, 2009b) and edge length score metrics have been studied separately (Frade et al, 2010b). In this work we study how the weighted sum of these two metrics, see Eq. (11), can improve terrain diversity, impact terrains aesthetic and influence the GP search performance. A multi-objective implementation is planned as future work, were more geomorphological metrics might be easily added.

$$fitness = w_a v_s + w_e \varepsilon_s \quad (11)$$

Table 2 Test parameters and their values

Par.	Value	Par.	Value
T_1	$\{ERC, myNoise\}$	s_1	18%
T_2	$\{ERC, X, Y\}$	s_2	27%
T_3	$\{ERC, X, Y, myNoise\}$	s_3	36%
pa_1	70%	pe_1	20%
pa_2	80%	pe_2	25%
pa_3	90%	pe_3	30%
w_a	$0.0, 0.1, ..., 1.0$	$seed$	$1, 2, ..., 20$

4 Tests and Results

As detailed in previous section, TPs evaluation depends on several parameters: slope threshold (from now on represented by s), percentage of accessibility area pa, percentage of the edge length pe and weights w_a and w_e. All parameters and terminal sets will impact both GP performance and resulting terrains. Therefore, to understand the behavior of GTP_a with weighted sum of accessibility and edge length scores, we devised a series of tests.

We grouped in Table 2 a set of parameters, which we designate as *Test Parameters*, whose influence we want to study. T_i where $i = 1, .., 3$ represent terminal sets whose propose was detailed in Section 3. Slope is another important parameter, it will affect the construction of the accessibility map A, and that way in-

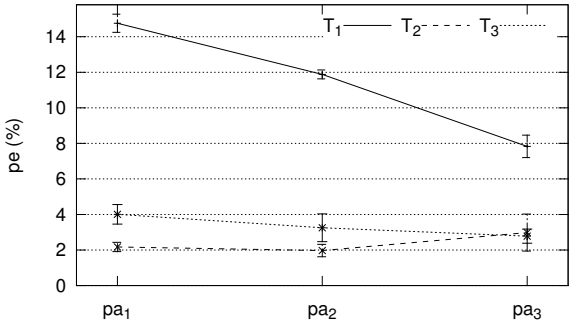

Fig. 6 Mean percentage of pe values calculated from the results obtained with Accessibility Score function (Frade et al, 2010a). Error bars show the standard error of the mean for 20 runs.

fluence the fitness of a given TP. Three different slopes s_j, $j = 1, .., 3$, were tested, whose values are in Table 2. These slope values were chosen because they are big enough to affect the movement of a common motorized vehicles and to see how flexible our system is and its impact in GTP_a performance. We also want to verify if this test parameter can indirectly influence terrain smoothness.

$$pe(\%) = 100 \times \frac{E_+}{n_r n_c} \tag{12}$$

The percentage of accessible terrain and edge length are controlled by pa and pe respectively. We performed tests with three different values for both parameters. We analyzed the accessibility maps produced in (Frade et al, 2010a) and measured the edge length from all maps, and calculated the correspondent pe values with Eq. (12), where $n_r, n_c = 128$. Figure 6 shows the results from that analysis, where it is noticeable that terminals T_2 and T_3 produce terrains with significantly smaller pe values than T_1. This observation was reinforced by a Mann-Whitney U-test of each parameter combination for T_2 and T_3 with respect to T_1. All tests returned *p-values* lower than 0.05 (see Table 3), which means that T_1 edge values are different, with statistical significance, from the ones obtained with T_2 and T_3. In face of these values and considering that the maximum pe obtained was 22.25% we decided to perform tests with the following values: $pe_1 = 20\%$, $pe_2 = 25\%$ and $pe_3 = 30\%$.

Finally, for these series of tests we established a linear relation between w_a and w_e as shown in Eq. (13). Due to this relation, from now on, we will refer only to w_a in results' discussion.

$$w_a + w_e = 1 \tag{13}$$

Table 3 Mann-Whitney U-test for edge values calculated when only accessibility was in use.

Test parameters		T_1		
		pa_1	pa_2	pa_3
T_2	pa_1	$6.302e^{-08}$	$6.302e^{-08}$	$6.302e^{-08}$
	pa_2	$6.302e^{-08}$	$6.302e^{-08}$	$4.871e^{-07}$
	pa_3	$1.122e^{-06}$	$1.122e^{-06}$	$1.122e^{-06}$
T_3	pa_1	$6.302e^{-08}$	$6.302e^{-08}$	$3.929e^{-05}$
	pa_2	$7.415e^{-07}$	$1.122e^{-06}$	$1.175e^{-05}$
	pa_3	$6.302e^{-08}$	$6.302e^{-08}$	$7.415e^{-07}$

Our tests included all the combinations between all the test parameters T_i, s_j, pa_k, pe_l and w_m. For each combination, 20 runs ($r = 1, 2, .., 20$) were performed with different seeds, which sums to 17 820 different executions. The experiments were performed in a cluster with 18 virtual machines in heterogeneous computers, all running GNU Linux.

Besides the Test Parameters, there are two more sets whose values where fixed for all runs. *GP Parameters* is one of them, whose maximum and initial values, as well as operators, are defined in Table 4. The search stops whenever the fitness reaches the value of zero or the amount of generations reaches the value of 50, whichever comes first. Both crossover and mutation operators are the same as the ones used by Koza (1992). The crossover operator uses tournament selection to chose two individuals and swap between them two randomly selected subtrees. Our tests were performed with a tournament size of 7, however preliminary tests were made with different sizes, but they all presented similar results. The mutation operator is subtree mutation and is applied to randomly chosen individuals, where a randomly selected subtree is replaced by another randomly created subtree. The mutation rate might be considered too high for most GP applications. However, our goal is not optimization, but to use the GP system as a tool to explore many different solutions. Therefore, a high mutation rate will help to avoid equal solutions for different runs.

The other parameter set is the *Height Map Parameters*, whose values are presented in Table 5. They are necessary because the evaluation of the GP individuals is made after converting them to high maps. These parameters were also fixed across all the runs we made.

Sub-section 4.1 presents the results of the test parameters over fitness, number of generations and tree size. Terminals and functions frequency analysis is presented in sub-section 4.2 followed by a terrain overlap

Table 4 GP Parameters

GP	Value
maximum generations	50
population size	500
initialization method	half and half
ramped	from 2 to 6
max. depth	17
selection operator	tournament, size 7
crossover operator	rate 70%
mutation operator	subtree, rate 30%

Table 5 Height map parameters

Height map	Value
n_r and n_c	128
L_x and L_y	0
S_x and S_y	1
D_x and D_y	10

study in section 4.3. Finally, the render of some TPs are shown in sub-section 4.4.

4.1 GP System

The amount of time the search phase will take is influenced by the complexity of the fitness function and test parameters values. So, in order to analyze how our GP system performed we plotted the average number of generations (Fig. 7), tree sizes (Fig. 8) and fitness values (Fig. 10).

Figure 7 shows the average number of generations that our system had to perform until a solution was found. The smaller the number of generations the better (less computations to find a solution). Five graphics are presented in Fig. 7. On top is plotted the mean number of generations regarding all performed experiments (global mean m_g) for each w_a. Bellow, four additional plots are presented regarding the difference between the mean number of generations for a given test parameter $m_{<parameter>}$ and the global mean. Those graphics, with difference values, are sorted by test parameters: *terminals* $(m_{T_i} - m_g)$, *slopes* $(m_{s_j} - m_g)$, accessibility $(m_{pa_k} - m_g)$ and edge length $(m_{pe_l} - m_g)$. This approach, of one global plot followed by four plots of differences, is also applied to Fig. 8, 9, 10, 11, 12 and 15.

The first thing to stand out from Fig. 7 (and also in Fig. 8) is that $w_a = 0$ and $w_a = 1$ are special cases. The amount of required generations on both situations is considerably lower than for $0.1 \leqslant w_a \leqslant 0.9$. For $0.1 \leqslant w_a \leqslant 0.9$ the number of generations present a small tendency to decrease as w_a increases. Regarding the influence of each terminal, it is clear that on average T_2 requires more generations than T_1 and T_3. The accessibility parameter pa_3 also requires more generations than pa_1 and pa_2 before achieving a solution. On the other hand, both slope and edge length parameters present a small influence on the number of generations.

For $0.1 \leqslant w_a \leqslant 0.9$ average tree sizes, see Fig 8, present a small trend to increase with the increase of w_a. Terminal T_2 generate trees whose size is consistently higher than T_1 and T_3. T_3 presents the smaller tree sizes, but with a very small difference to T_1. Parameter pe_1 displays smaller tree sizes than the others edge length parameters, but that advantage decreases as w_a increases and vanish after $w_a = 0.7$. The test parameters for slope and accessibility have a very small influence in tree sizes. Curiously, the amount of generations tends to decrease as w_a increases, in the range $0.1 \leqslant w_a \leqslant 0.9$, while tree sizes present the opposite behavior. This suggests that, in this range, more complex TPs are able to solve the problem better and that higher w_a promotes this kind of trees.

Since TPs can be used to generate terrains dynamically, their execution time is of most importance. Therefore, we measured the execution time of the best TPs generating a height map of size 1024×1024 with *double* precision. This task was performed in a single computer with a Core 2 Duo CPU running at 2.4GHz with 1GB of RAM. Execution time varied between 0.101 and 9.613 seconds, depending on tree size and functions at use. Figure 9 presents the average execution time of best TPs. Although it presents bigger tree sizes, terminal T_2 has lower execution times than T_1 and T_3. This is explained by the fact that terminal $myNoise$ is very complex and time consuming function, which penalizes execution times of T_1 and T_3.

Our results show that TPs can generate big maps with times in the same order of magnitude of the ones obtained by Belhadj (2007). However, each cell value of the height map is independent from others cells, so TPs present very good scalability and can take advantage of modern multi core CPUs or GPUs to speed up its generation. Therefore, TPs execution times can be greatly improved.

Our fitness function was built to be minimized, therefore the closer the fitness values are to zero, the better. Globally, the higher w_a is (except for $w_a = 0$) the better the fitness values are (closer to zero), see Fig. 10. It is clear that as pa increases the fitness values get worse, which was expected. However, we did not anticipated such a huge difference between pa_3 and the others pa values. The slope impact in fitness shows that s_1 has

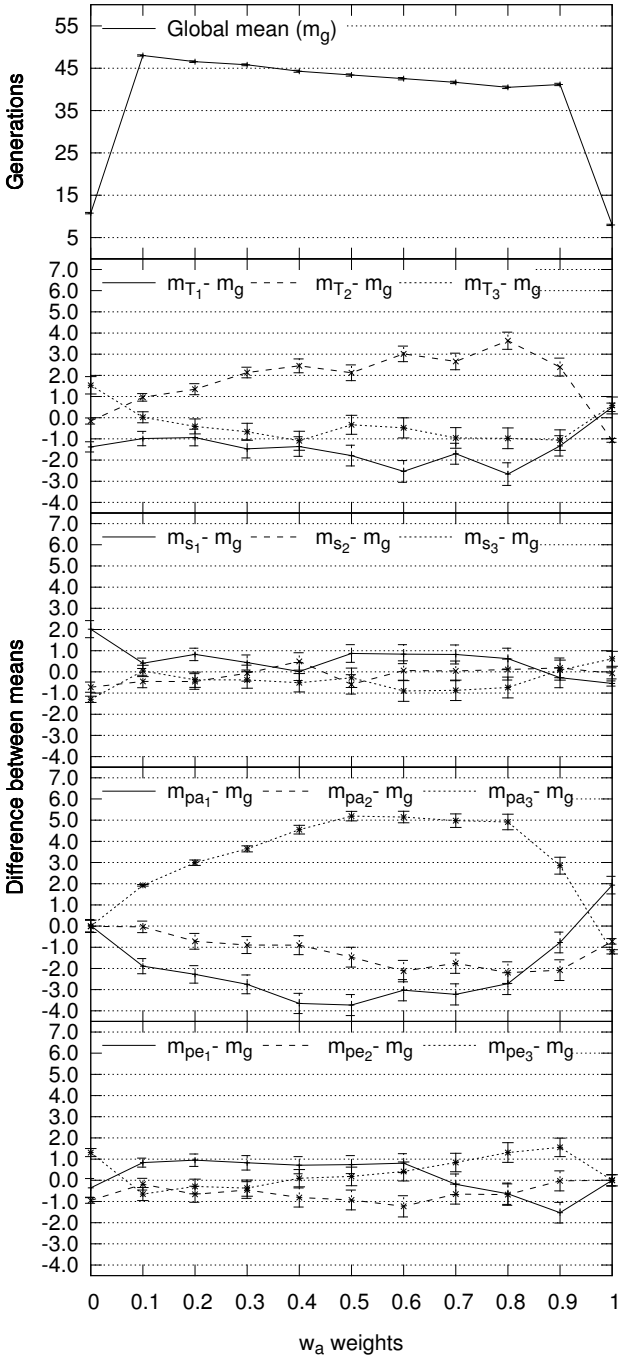

Fig. 7 Mean number of generations versus w_a. Error bars represent the standard error of the mean.

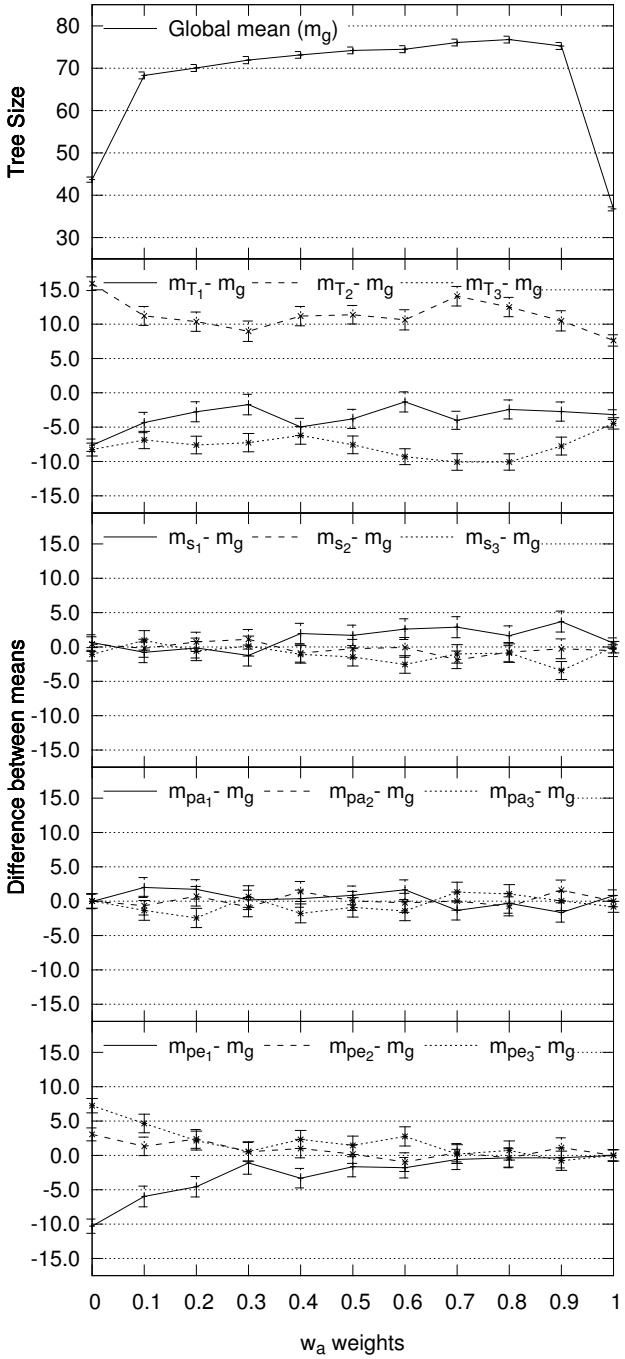

Fig. 8 Mean of GP tree sizes versus w_a. Error bars represent the standard error of the mean.

worse performance than s_2 and s_3. We were expecting that smaller slope values would have better fitness. A more detailed analysis was conducted and we noticed that runs with the combination of s_1 with pa_3 was the main cause for the globally bad performance of s_1. Slopes s_2 and s_3 have a similar behavior.

Considering the average edge length values in Fig. 6 we expected that the higher the pe value was, the worse the fitness would be. However, pe_1 presents worse values than the others, pe_2 has the better fitness values, followed closely by pe_3. This might mean that our system does not behave linearly with the edge length parameter, further tests with lower pe values are required to better understand this parameter.

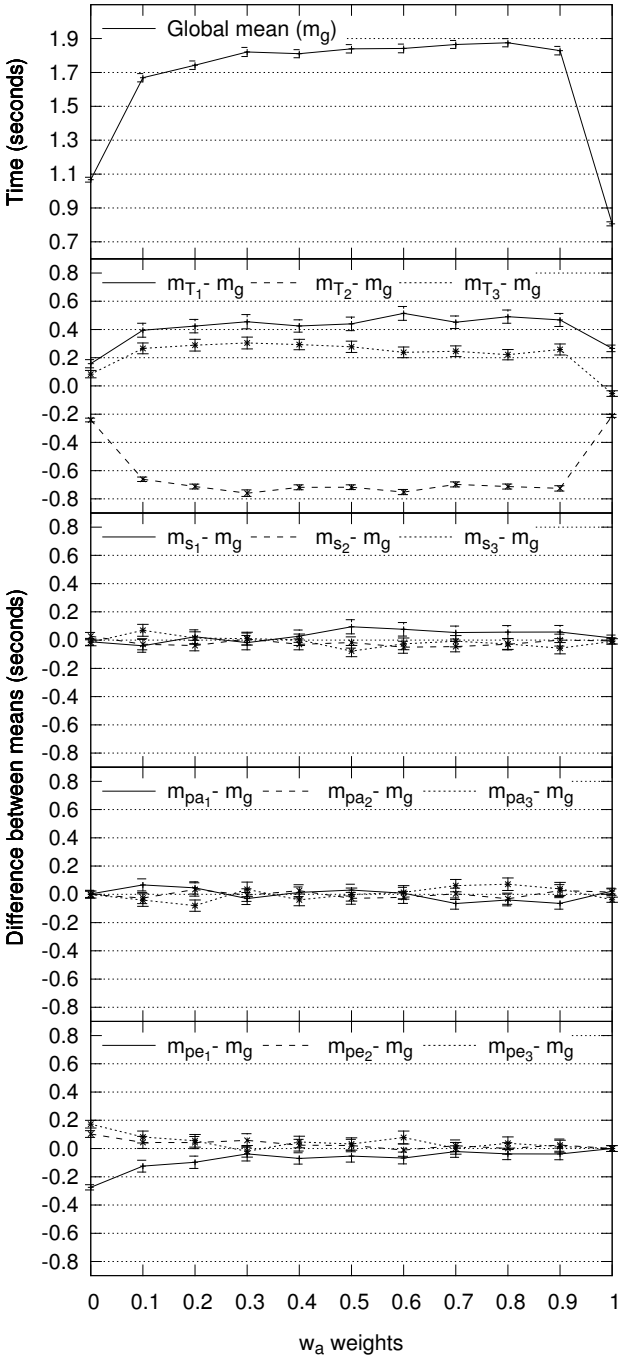

Fig. 9 Mean of TP execution times versus w_a. Error bars represent the standard error of the mean.

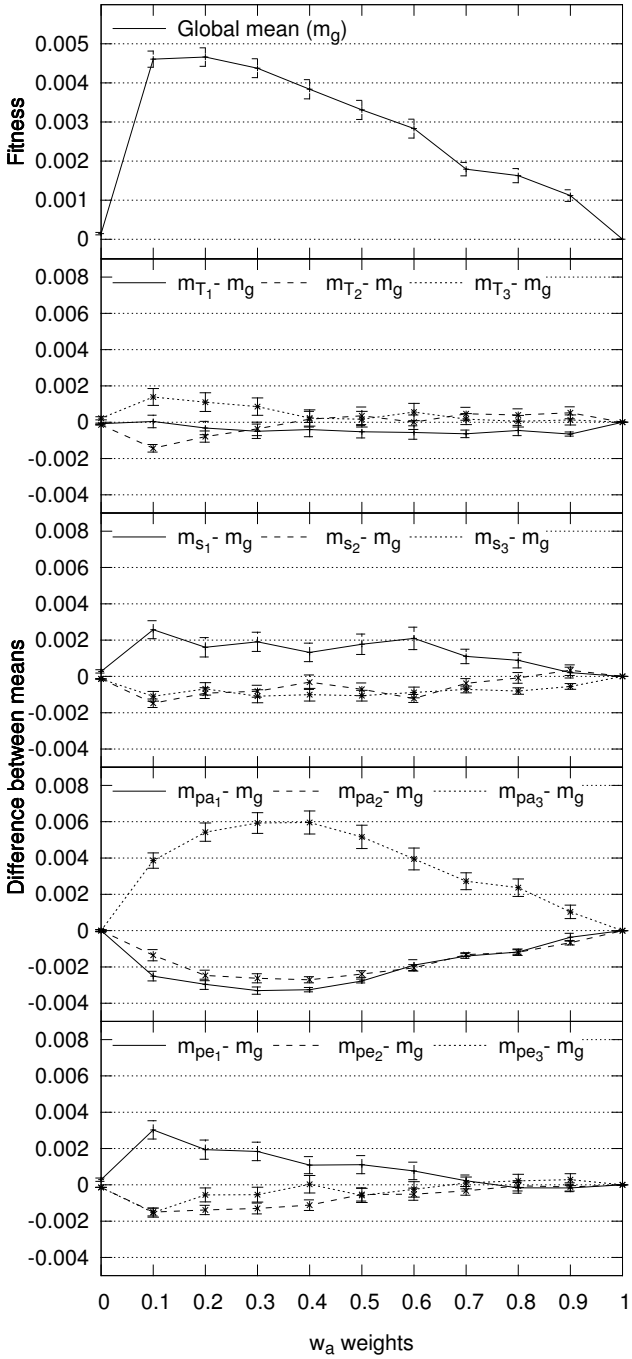

Fig. 10 Mean fitness values versus w_a. Error bars represent the standard error of the mean.

We expected the fitness of T_3 to be similar or better than the other two terminals sets, given that T_3 is the union of T_1 and T_2. To find out why this was happening we plotted the percentage of solutions (TPs) that reached fitness zero. Although T_3 presented the worse average fitness value for most of the w_a range, it clearly presented higher percentage of TPs with fitness zero than the ones with T_2, see Fig. 11. This suggests that T_3 might be able to achieve better fitness values than the others terminals if a higher limit of generations is used.

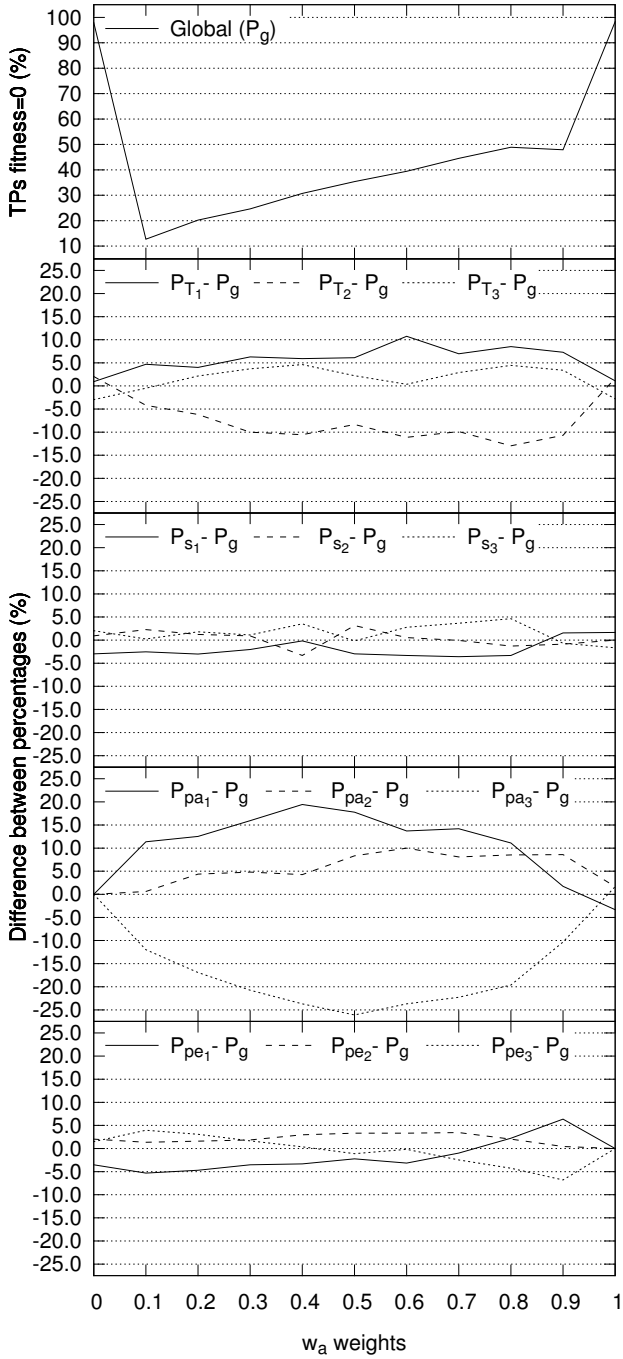

Fig. 11 Percentage of TPs that reached fitness 0 versus w_a.

4.2 Occurrence Analysis

To see which functions and terminals contributed the most to achieve the best solutions, we calculated how often each of them occurred according to Eq. (14). The terminal or function which we want to calculate is represented by fun_h, TP_r is the solution for the seed r and N is the sum of tree sizes for all seeds r and is calculated by Eq. (15).

$$Oc(fun_h) = \frac{1}{N}\sum_{r=1}^{r_n} count(fun_h, TP_r) \tag{14}$$

$$N = \sum_{h=1}^{fun_n}\sum_{r=1}^{r_n} count(fun_h, TP_r) \tag{15}$$

Figure 12 shows the occurrence of each terminal and function and their variation imposed by test parameters. As expected, terminals have a great impact. For T_1 and T_3 the terminal $myNoise$ is quite predominant when compared with the remaining functions. Terminal $myNoise$ seems to be main responsible for the good results of T_1 and T_3 depicted in Fig. 11, although X and Y seem to be better at finding solutions for the edge length score function for $0 < w_a < 0.3$. Terminal ERC occurrence is impacted by terminals and, with less significance by slope, accessibility and edge length parameters. The third most common function is *cos* which is affected by chosen terminal, slope or edge length parameters. It is also noticeable that *pa* has almost no influence in functions occurrence, *pe* only influences *cos* and $mySqrt$ significantly. Finally, *slope* influences mainly *cos*, $multiply$, $myDivide$ and $mySqrt$. Figure 13 shows the influence of w_a over the average occurrence of each function. However, only $w_a = 1$ has a considerable impact on functions occurrence.

Given that one of our goals is to explore different solutions, we decided also to see if there were any repeated solutions (TPs). We found a total of 106 different TPs that appeared more than once, relative to 248 runs (which represents 1.39% of the runs). As we expected, the higher concentration of repeated TPs was in the w_a values where the fitness values were worse, specially for $w_a = 0.1$. A larger limit for the amount of generations might reduce, or even eliminate, the amount of repeated TPs, but more tests are needed to confirm it.

4.3 Overlap

As shown in Fig. 11, there was a relative large number of TPs to reach the perfect fitness value of zero for different test parameters combinations. We want to investigate if this happens due to the existence of several different solutions, or due to convergence of the solutions. We already know that there are some repeated TPs, but these solutions have not reached a fitness value of zero, besides there might exist different TPs that are mathematically equivalent and render the same terrain. Therefore, we compared each accessibility map with the other 19 from the 20 runs of each test (changing only the seed). The comparison consists in counting how much

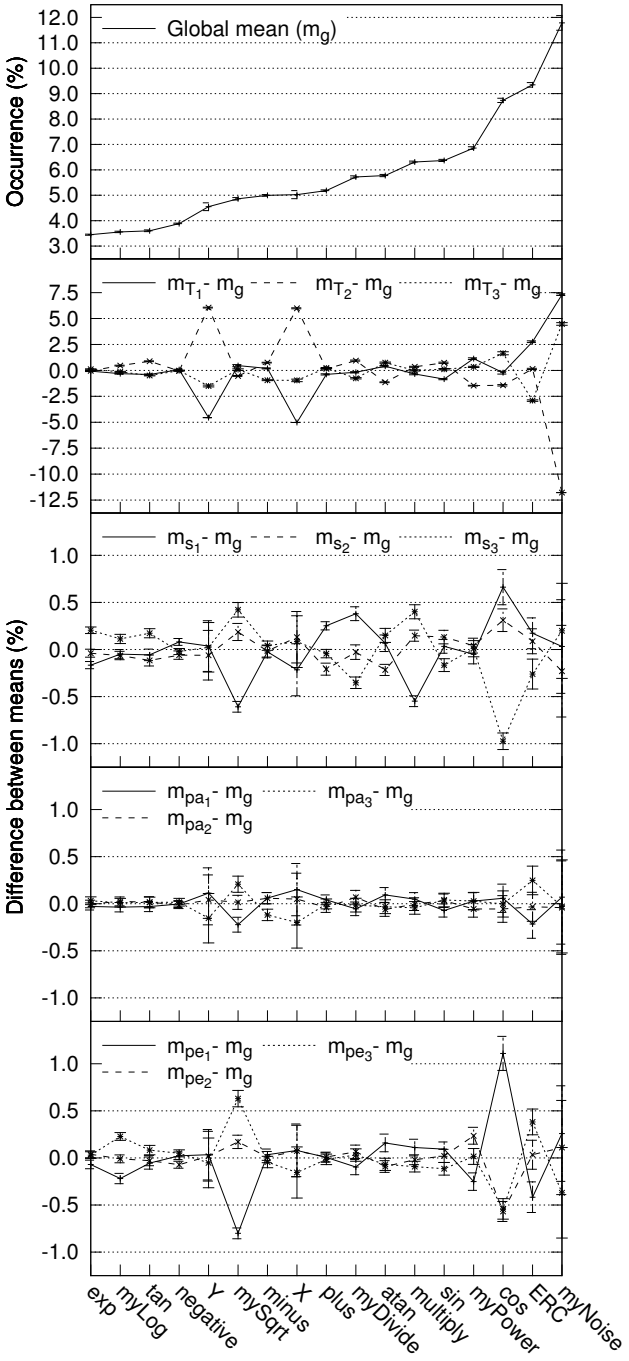

Fig. 12 Mean occurrence of functions and terminals versus w_a. Error bars represent the standard error of the mean.

inaccessible (black, $a_{r,c} = 0$) area overlaps between two accessibility maps, see example in Fig 14. To compute the overlap value $o_{p,q}$ between two maps A_p and A_q we used Eq. (16), were an overlap value of 100% means that maps A_p and A_q are equal. Then we defined the overlap value of each map o_p as the average of all $o_{p,q}$, as shown in Eq. (17).

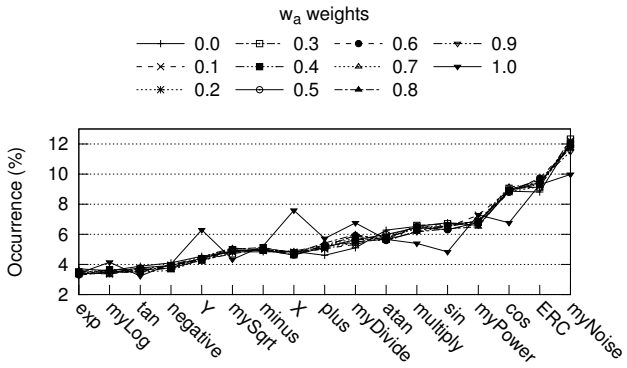

Fig. 13 Mean occurrence of functions and terminals for a given w_a weight.

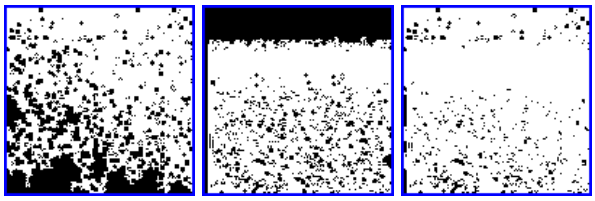

Fig. 14 Overlap of inaccessible areas between two maps. These maps are from T_2, s_2, pa_1, pe_3, r_8 on the left (1) and r_{16} at center (2). On the right is the resulting overlap with $o_{1,2} = 21.67\%$

$$o_{p,q}(\%) = 100 \times \frac{\sum_{r=1}^{n_r}\sum_{c=1}^{n_c}(\neg A_p \wedge \neg A_q)}{max\left(\sum_{r=1}^{n_r}\sum_{c=1}^{n_c}\neg A_p, \sum_{r=1}^{n_r}\sum_{c=1}^{n_c}\neg A_q\right)} \quad (16)$$

$$o_p = \frac{1}{r_n - 1}\sum_{q=1}^{r_n} o_{p,q}, \quad q \neq p \quad (17)$$

Figure 15 shows the average overlap values o_p and the correspondent influence of the test parameters. Overall, the overlap value for $w_a = 0$ and $w_a = 1$ are higher than for the remaining range of w_a. This is in part explained by the hight amount of terrains that presented an overlap of 100%, which was 1.30% and 1.11% respectively. For $w_a = 0.1$ and $w_a = 0.2$ the amount of maps with an overlap of 100% was 0.19% and 0.06%, while for the remaining w_a values was 0.00%. Overall, the weighted combination of the accessibility score and edge length score is beneficial to reduce the average overlap values. Which is good, because we want to be able to generate as much diverse terrains as possible.

Terminal T_2 is the one that provides lower overlap values, followed by T_3 and then T_1. Slope has no significant impact in overlap and pe only makes difference for pe_1 and $w_a = 0$, for the remaining values it also has no

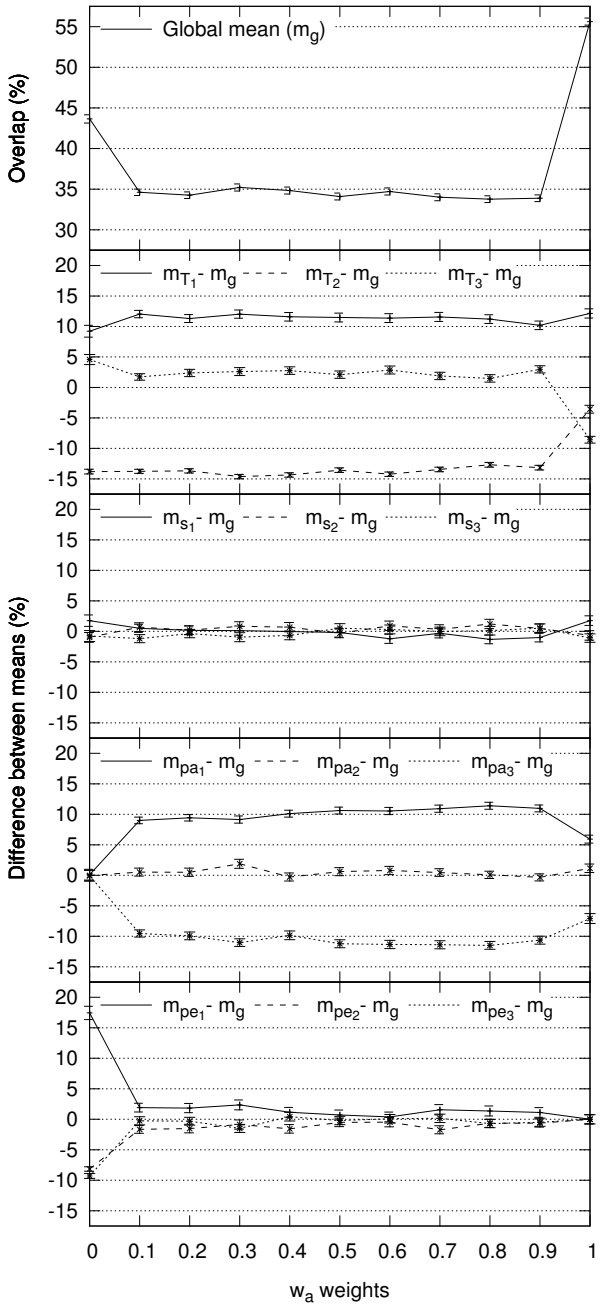

Fig. 15 Overlap of inaccessible areas versus w_a. Error bars represent the standard error of the mean.

influence. Finally, the accessibility parameter presents an expected behavior, the higher *pa* is the lower the overlap values are.

4.4 Sample Terrains

Given the huge amount of results, we only performed a visual inspection of 100 terrains for each terminal set. To illustrate them we present 8 different TPs for each terminal set, which are displayed in Fig 16 to 27. Both the visual inspected and presented terrains were randomly selected, although we deliberately avoid to present any terrain obtained with $w_a = 0$ and $w_a = 1$, because these situations were already address in our previous work (Frade et al, 2010a,b).

For all depicted terrains, we present on top the H map displayed as gray scale image and its correspondent accessibility map A. At the bottom, we show a rendered image of a three dimension view point from the terrain. Those renders were performed in Blender 3D without textures to emphasize terrains surface shape. Each figure has the identification of the TP that generated it with the following syntax: *terminal*, *slope*, *pa*, *pe*, w_a and *seed*. For abbreviation proposes we replaced w_a by w_m and *seed* by r_u, where m can take values in the range $m = 0, .., 10$ and $u = 1, .., 20$.

From our visual inspection, it is clear that terminal sets have a great impact in both terrains look and diversity. Terminal set T_1 is the one that has the lowest diversity. We found several terrains that were quite similar, one example is the right terrain from Fig. 16 and the left one from Fig 18. This similarity is due the small number of terminals in T_1 and the high frequency value of $myNoise$. T_3 presents more diversity than T_1, but the influence of terminal $myNoise$ is quite noticeable, which was expected given its high rate of occurrence shown in Fig. 12. The impact of terminals X and Y is also perceptible, but much more subtle. For instance, on left terrain of Fig. 24 it is possible to see the wave shape of the terrain (this feature is easier to perceive in the gray scale image), although with a very small amplitude. On the right terrain of the same figure the height values steadily increase along the Y axis (see also correspondent gray scale image). Another good example of X and Y terminals influence on T_3 are both terrains shown in Fig. 27, were the terrains change their look at a given point, abruptly in the left terrain and smoothly in the right one. Terminal set T_2 is the one that presents more diverse terrains. From the analyzed samples we have not found terrains with a high degree of similarity as the example mentioned previously. However, terrains from T_2 tend to present geometric patterns and symmetry, which give them a strange look.

Results regarding diversity were somehow expected, given previous experience. Still, we had hope that the combination of the accessibility and edge length metrics would have a positive impact in diversity. Our hopes were increased when the overlap values (presented in Fig. 15), showed smaller overlap values when both metrics were used. However, after performing our visual inspection we can not state that the diversity of terrains

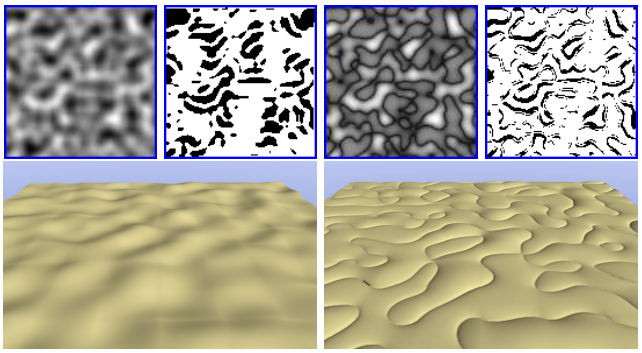

Fig. 16 Terrains generated by TP $T_1, s_1, pa_1, pe_1, w_2, r_5$ with $fitness = 0.000000$ on the left, and $T_1, s_1, pa_2, pe_3, w_4, r_{16}$ with $fitness = 0.000445$ on the right

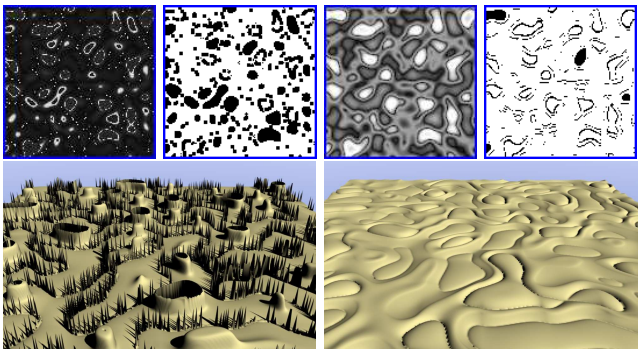

Fig. 19 Terrains generated by TP $T_1, s_3, pa_1, pe_2, w_4, r_{18}$ with $fitness = 0.000000$ on the left, and $T_1, s_3, pa_3, pe_2, w_5, r_{10}$ with $fitness = 0.000151$ on the right

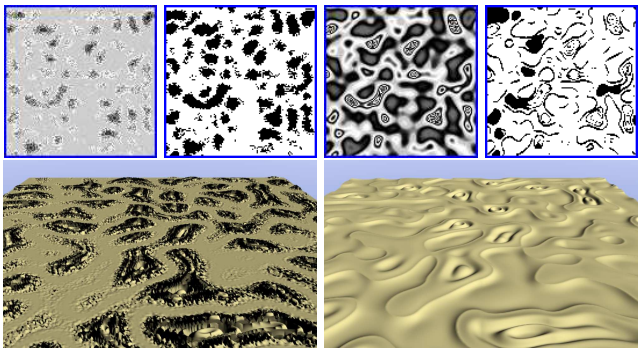

Fig. 17 Terrains generated by TP $T_1, s_2, pa_1, pe_2, w_9, r_9$ with $fitness = 0.000098$ on the left, and $T_1, s_2, pa_2, pe_3, w_8, r_1$ with $fitness = 0.000000$ on the right

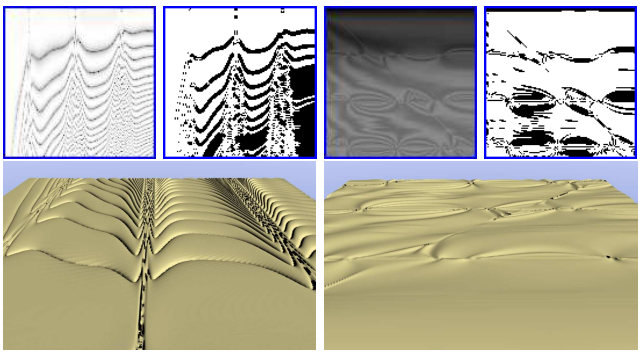

Fig. 20 Terrains generated by TP $T_2, s_1, pa_1, pe_2, w_7, r_2$ with $fitness = 0.000000$ on the left, and $T_2, s_1, pa_2, pe_1, w_6, r_4$ with $fitness = 0.000400$ on the right

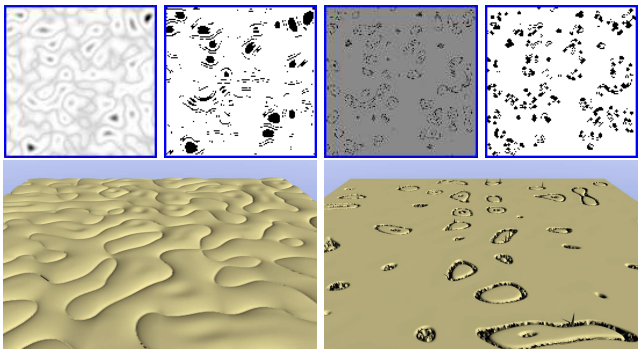

Fig. 18 Terrains generated by TP $T_1, s_2, pa_3, pe_1, w_1, r_{14}$ with $fitness = 0.000053$ on the left, and $T_1, s_2, pa_3, pe_1, w_5, r_2$ with $fitness = 0.000000$ on the right

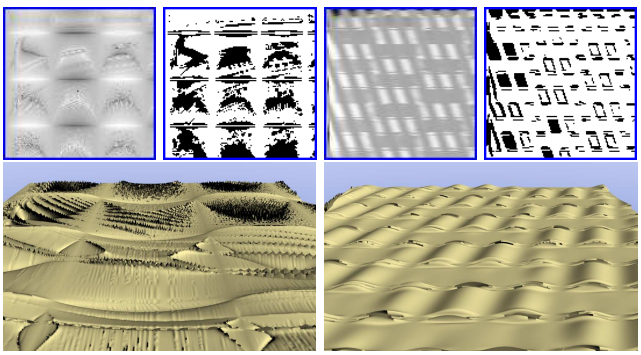

Fig. 21 Terrains generated by TP $T_2, s_2, pa_1, pe_2, w_9, r_9$ with $fitness = 0.000000$ on the left, and $T_2, s_2, pa_2, pe_2, w_1, r_{18}$ with $fitness = 0.001144$ on the right

has increased. We believe that the increase of diversity can be better addressed by fine-tuning the terminal set.

We also noticed an unexpected side effect of using both metrics to generate terrains. Generally, the amplitude of terrains (the difference between the lowest and highest height values) was very small. The left terrain from Fig. 19 is one of the few exceptions, but even that one does not present high amplitudes as some terrains obtained for $w_a = 1$. This was strange, because we do not impose any restriction to height values. Our function set (see Table 1) is composed by continuous functions, with only three exceptions: $myLog(a)$ when $a = 0$, $myDivide(a, b)$ when $b = 0$ and $myPower(a, b)$ when $a = 0$ and $b < 0$. We thought those exceptions were enough to create sudden changes in terrain and create hight obstacles this way. However, to accom-

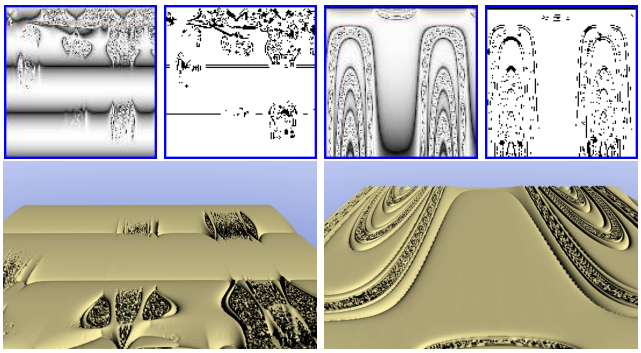

Fig. 22 Terrains generated by TP $T_2, s_2, pa_3, pe_1, w_8, r_3$ with $fitness = 0.000181$ on the left, and $T_2, s_2, pa_3, pe_2, w_9, r_8$ with $fitness = 0.000068$ on the right

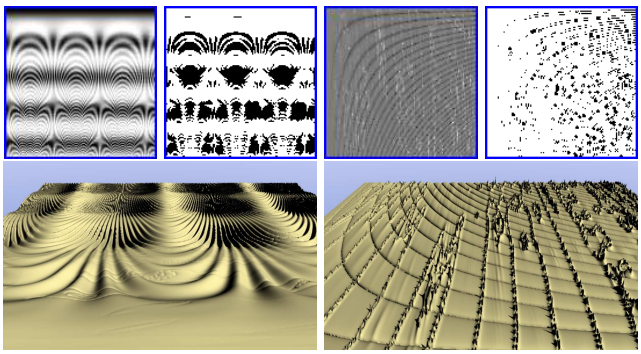

Fig. 23 Terrains generated by TP $T_2, s_3, pa_1, pe_2, w_2, r_{13}$ with $fitness = 0.000199$ on the left, and $T_2, s_3, pa_3, pe_3, w_1, r_2$ with $fitness = 0.000015$ on the right

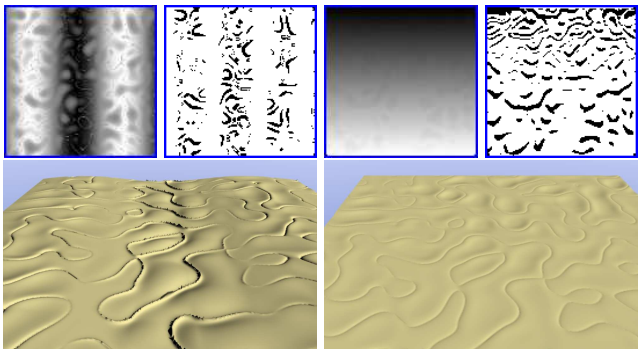

Fig. 24 Terrains generated by TP $T_3, s_1, pa_3, pe_1, w_2, r_6$ with $fitness = 0.008510$ on the left, and $T_3, s_1, pa_3, pe_2, w_4, r_{11}$ with $fitness = 0.042003$ on the right

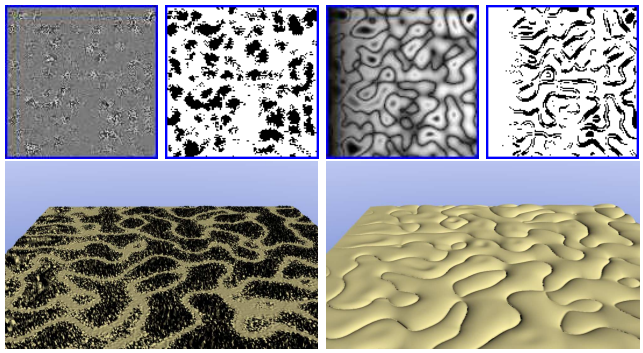

Fig. 25 Terrains generated by TP $T_3, s_2, pa_1, pe_3, w_8, r_{17}$ with $fitness = 0.000470$ on the left, and $T_3, s_2, pa_2, pe_3, w_4, r_{16}$ with $fitness = 0.000000$ on the right

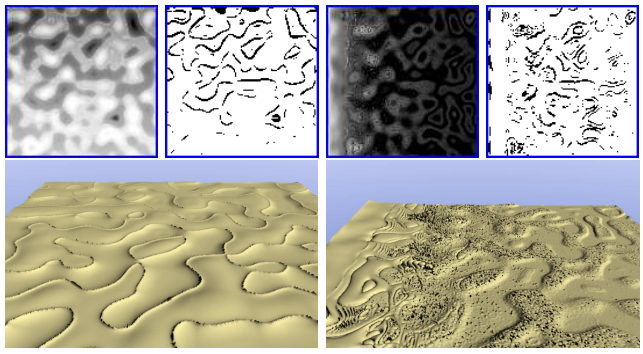

Fig. 26 Terrains generated by TP $T_3, s_2, pa_3, pe_1, w_8, r_{16}$ with $fitness = 0.000060$ on the left, and $T_3, s_2, pa_3, pe_2, w_7, r_8$ with $fitness = 0.002825$ on the right

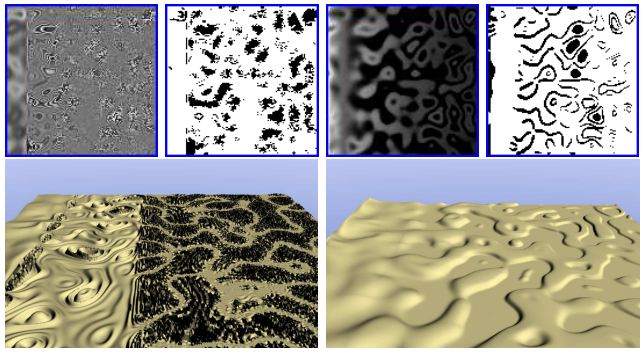

Fig. 27 Terrains generated by TP $T_3, s_3, pa_2, pe_2, w_8, r_{10}$ with $fitness = 0.000000$ on the left, and $T_3, s_3, pa_3, pe_2, w_2, r_8$ with $fitness = 0.020270$ on the right

plish the required edge length terrain height values must change often. Therefore, we believe the edge length metric is the main responsible for small amplitude terrains, specially with the chosen *pe* values. Frequency results in Fig. 13 corroborate this reasoning, because only for $w_a = 1$ the frequency values change significantly, besides the presence of periodic functions, like *cos* and *sin*, decrease. Still, we think further tests with smaller *pe* values should be performed to confirm whether they allow terrains with bigger amplitudes.

Another option to address the amplitude issue would be to include one or more functions with discontinuous behavior on the function set, for example *mod* (remainder for the modulo operation) or the *if* statement. However, in this case we think that some additional measures should be taken to prevent those dis-

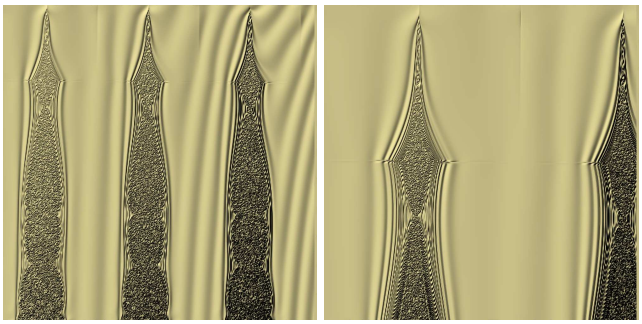

Fig. 28 Top view of TP $T_2, s_1, pa_3, pe_3, w_5, r_1$ with 2 different zoom levels: $S_x = S_y = 1$ and 2 (see Eq. 2)

Fig. 29 Screenshot of *Chapas* video game were the terrain was generated online by a TP.

continuous functions to dominate the solutions, which would prevent the appearance of smooth terrains. One of these measures could be different probability values for a given function to be chosen from the function set.

Although the picked slope values would have a severe impact in the mobility of vehicles, their differences were not big enough to impact terrains in a visible way. In fact, considering the results regarding the GP system, overall *slope* has a very limited influence, being only significant in fitness values. Therefore, we think further tests must be performed with *slope* values covering a bigger range to access if they can influence terrains smoothness.

As stated previously, TPs generate a continuous surface that needs to be sampled and limited to generate the height map. This is achieved by Eq. (2) which also allow us to control the zoom level (through S_x and S_y) and resolution (through n_r and n_c). The zoom level allows video games to compute only a small portion of the terrain that needs to be displayed. This can be used to simulate a player approaching or getting away from a particular point in the terrain, see Fig. 28. On the other hand, resolution will allow video game developers to control the amount of processing required to generate the terrain at the expense of terrain details.

To show the usability of TPs as a procedural technique to generate terrains dynamically in a real video game, a few selected TPs were embedded in *Chapas*[4] video game. Chapas (Rodrigues et al, 2010) is an open source turn-based bottlecap racing game, with 3D graphics, where the players strategically control the racers with cards and has support for multi-players. Figure 29 presents a screenshot of Chapas video game at the beginning of the running phase.

[4]Available for download at `http://sourceforge.net/projects/chapas/`

5 Conclusions and Future Work

The GTP_a technique performs automated search of procedures, by means of genetic programing, that are able to generate terrains according to the weighted sum of two metrics: accessibility score and edge length score. The parameters allow us to control the slope threshold (that differentiate the accessible from the inaccessible terrain areas), how much area should be made accessible, and the edge length of the inaccessible areas. Throughout a series of experiments we have shown that our system is able to find many different solution that fit our fitness function. Both accessibility score and edge length metrics perform the desired function, but our results show that GTP_a can achieve better fitness values for accessibility score than for the edge length metric (see Fig. 10). The combination of the two metrics also helps to decrease terrain overlap, which is desirable as it means more diverse solutions. However, it will not increase terrain diversity when compared with the use of a single metric. This combination also presents the side effect of generating terrains with small amplitudes (the difference between the lowest and highest height values), whose main responsible is the edge length score function. We believe this problem can be addressed using lower *pe* values and by introducing more discontinuous functions in the function set.

Repeated TPs represent 1.39% of the solutions and appear when the fitness values are worse, but overall 45.22% of the solutions reached the perfect score of zero. Both situations, increasing the amount of solutions reaching fitness value of zero and reducing the amount of repeated TPS, can be addressed by increasing the maximum allowed generations. This will also help terminal set T_3 to have better fitness values, because the more elements the terminal set has, the bigger the search space is, and more generations will be required to find a good solution. Chosen slope threshold values also influence fitness values, were $s_1 = 20\%$

presented the worse results, however its impact is negligible for the remaining GP system performance.

The search for the right TP can be long, depending mainly on how many generations are allowed, population size and used metrics, but this is a common characteristic of search-based techniques. However, once found TPs execution times are short and in the same order of magnitude of other procedural techniques. TPs have also the advantages of offering room for execution time improvements as they are easily parallelized, a feature that many procedural techniques do not present. To create a terrain from a TP only height map parameters (see Table 5) are need, but these only concern terrain resolution, zoom level and origin, not its look. Therefore, TPs do not require any input parameter to model terrain shape and there is no need for a time consuming and expensive phase of parameter tunning. To prove the viability of our technique some TPs are already in use in a real video game, were the terrain generation occurs online.

As expected, terminal sets have a big impact in terrain diversity, look and aesthetic appeal. Terminal set T_1 has few diversity, but showed us the potentiality of implicit functions as terminal by providing appealing surfaces without pattern repetition. On the other hand, T_2 has many diverse terrain types, but exhibits many geometric patterns. Finally, T_3, which is the union of the other two terminal sets, reinforces the importance of $myNoise$ terminal to achieve fit solutions. Therefore, most terrains produced with T_3 present a heavy influence of $myNoise$ terminal in its looks. In regard to slope parameter, it did not present a significant change in terrains looks, which we believe to be a direct consequence of a narrow range of the used values.

In spite of the interesting results, this work opens many challenges for future research. Logically the next step would be to test our system under a multi-objective approach, given that we used two different metrics and more could be added this way. Nevertheless, there are many topics that can also be addressed in future work. For instance, the prevalence of $myNoise$ in T_3 showed us that fractal based function are important to find fitter solutions with an interesting aesthetic appeal. However, the diversity of terrain types is not big enough, so it could be augmented by adding more discontinuous functions to the function set and by adding new fractal based functions. Given the current used metrics discontinuous functions can overtake the predominant role and that way avoiding the appearance of smooth terrains. So, a new line of work can be the study of different probabilities for the functions to be selected from their set. With new fractal based functions we could obtain changes in frequency and amplitude in terrain features, but this approach introduces new questions. An open question is whether the new fractal functions should be introduced as terminals with implicit parameters, or whether those parameters should evolve as well. Some fractal based functions present parameters like octaves and lacunarity whose values are valid or interesting only in a limited range. So, if we let those parameters to evolve the values must be normalized, which raises the question of what normalization function to use.

Another possible research line could be to try new metrics from the geomorphology field to see if this way it would be possible to obtain more realistic terrains. Some researchers claim to be possible to classify all real terrains with only 3 parameters designated as geometric signatures (Iwahashi and Pike, 2007), using them as a search criteria can be of interest. Other types of search criteria can also be studied, for instance level curves to define desired terrain shape, instead of parameters.

Acknowledgements We are deeply grateful to the Informatics and Communications Services of Computer Science department from University of Coimbra, Portugal, for giving us a time slot on 18 nodes of their MILIPEIA cluster. Also, a special word of appreciation to Patrício Domingues that made the required arrangements that made possible our use of the cluster. Without their generosity our tests would not have been possible.

The authors acknowledge the support of Spanish Ministry of Science and Innovation under project ANYSELF (TIN2011-28627-C04). The first an second authors are supported by University of Extremadura, project *Grupo GEA*. The second author is also supported by *Gobierno de Extremadura, Consejeria de Economia-Comercio e Innovacion* and FEDER, project GRU09105. The third author is supported by Spanish Ministry of Science and Innovation under projects TIN2008-05941, and by *Junta de Andalucia* under project TIC-6083.

References

Ashlock D, Gent S, Bryden K (2008) Embryogenesis of artificial landscapes. In: Hingston PF, Barone LC, Michalewicz Z (eds) Design by Evolution, Natural Computing Series, Springer Berlin Heidelberg, pp 203–221, URL `http://dx.doi.org/10.1007/978-3-540-74111-4\_12`

Belhadj F (2007) Terrain modeling: a constrained fractal model. In: 5th International conference on CG, virtual reality, visualisation and interaction in Africa, ACM, Grahamstown, South Africa, pp 197–204, DOI 10.1145/1294685.1294717

Belhadj F, Audibert P (2005) Modeling landscapes with ridges and rivers: bottom up approach. In: GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia,

ACM, New York, NY, USA, pp 447–450, DOI http://doi.acm.org/10.1145/1101389.1101479

Brosz J, Samavati FF, Sousa MC (2006) Terrain Synthesis By-Example. In: First International Conference on Computer Graphics Theory and Applications

Carpentier G, Bidarra R (2009) Interactive GPU-based procedural heightfield brushes. In: Proceedings of the 4th International Conference on Foundations of Digital Games, ACM New York, NY, USA, pp 55–62

Chiang M, Huang J, Tai W, Liu C, Chang C (2005) Terrain synthesis: An interactive approach. In: International Workshop on Advanced Image Tech

Doran J, Parberry I (2010) Controlled Procedural Terrain Generation Using Software Agents. IEEE Transactions on Computational Intelligence and AI in Games 2(2)

Ebert D, Musgrave K, Peachey D, Perlin K, Worley S (2003) Texturing and Modeling: A Procedural Approach, 3rd edn. Morgan Kaufmann

Edwards R (2006) The Economics of Game Publishing. Website (accessed on Sep. 2011), http://uk.games.ign.com/articles/708/708972p1.html

Forbus KD, Mahoney JV, Dill K (2002) How Qualitative Spatial Reasoning Can Improve Strategy Game AIs. IEEE Intelligent Systems 17(4):25–30, DOI http://dx.doi.org/10.1109/MIS.2002.1024748

Frade M, Fernández de Vega F, Cotta C (2008) Modelling Video Games' Landscapes by Means of Genetic Terrain Programming - A New Approach for Improving Users' Experience. In: Giacobini M, et al (eds) Applications of Evolutionary Computing, Springer, Napoli, Italy, Lecture Notes in Computer Science, vol 4974, pp 485–490

Frade M, Fernández de Vega F, Cotta C (2009a) Adding Zoom Feature to Terrain Programmes. In: VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09), Málaga, Spain, pp 293–300

Frade M, Fernández de Vega F, Cotta C (2009b) Breeding Terrains with Genetic Terrain Programming - The Evolution of Terrain Generators. International Journal for Computer Games Technology 2009(Article ID 125714):13, DOI 10.1155/2009/125714

Frade M, Fernández de Vega F, Cotta C (2010a) Evolution of Artificial Terrains for Video Games Based on Accessibility . In: Chio CD, et al (eds) Applications of Evolutionary Computation, Springer, Lecture Notes in Computer Science, vol 6024, pp 90–99

Frade M, Fernández de Vega F, Cotta C (2010b) Evolution of Artificial Terrains for Video Games Based on Obstacles Edge Length. In: IEEE Congress on Evolutionary Computation 2010, pp 1–8, DOI 10.1109/CEC.2010.5586032

Gonzalez RC, Woods RE (2002) Digital Image Processing, 2nd edn. Prentice Hall

Goodchild M (1980) Fractals and the accuracy of geographical measures. Mathematical Geology 12:85—-98

Hastings EJ, Guha RK, Stanley KO (2009) Evolving Content in the Galactic Arms Race Video Game. Computational Intelligence pp 241–248

Horn B (1981) Hill shading and the reflectance map. Proceedings of the IEEE 69(1):14–47

Iwahashi J, Pike RJ (2007) Automated classifications of topography from DEMs by an unsupervised nested-means algorithm and a three-part geometric signature. Geomorphology 86(3-4):409 – 440, DOI DOI:10.1016/j.geomorph.2006.09.012, URL http://www.sciencedirect.com/science/article/B6V93-4M6SB5Y-3/2/510ca957d542d84fba3e3af50968fdf8

Kamal KR, Uddin YS (2007) Parametrically controlled terrain generation. In: GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, ACM, New York, NY, USA, pp 17–23, DOI http://doi.acm.org/10.1145/1321261.1321264

Koza JR (1992) Genetic Programming. On the programming of computers by means of natural selection. Cambridge MA: The MIT Press

Lane B, Prusinkiewicz P (2002) Generating Spatial Distributions for Multilevel Models of Plant Communities. In: Proceedings of Graphics Interface 2002

Li Q, Wang G, Zhou F, Tang X, Yang K (2006) Example-Based Realistic Terrain Generation. In: Pan Z, Cheok A, Haller M, Lau R, Saito H, Liang R (eds) Advances in Artificial Reality and Tele-Existence, Lecture Notes in Computer Science, vol 4282, Springer Berlin / Heidelberg, pp 811–818, URL http://dx.doi.org/10.1007/11941354_84, 10.1007/11941354_84

Mandelbrot BB (1983) The Fractal Geometry of Nature. W. H. Freeman

Mastin G, Watterberg P, Mareda J (1987) Fourier synthesis of ocean scenes. IEEE Computer Graphics and Applications 7:16–23, DOI http://doi.ieeecomputersociety.org/10.1109/MCG.1987.276961

Miller GSP (1986) The definition and rendering of terrain maps. In: SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, pp 39–48, DOI \url{http://doi.acm.org/10.1145/15922.15890}

Musgrave FK, Kolb CE, Mace RS (1989) The synthesis and rendering of eroded fractal terrains. In: SIG-

GRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques, ACM, NY, USA, pp 41–50, DOI http://doi.acm.org/10.1145/74333.74337

Nelson MJ, Mateas M (2007) Towards automated game design. In: AI*IA '07: Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence on AI*IA 2007, Springer-Verlag, Berlin, Heidelberg, pp 626–637, DOI http://dx.doi.org/10.1007/978-3-540-74782-6_54

Olsen J (2004) 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

Ong TJ, Saunders R, Keyser J, Leggett JJ (2005) Terrain generation using genetic algorithms. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM, NY, USA, pp 1463–1470, DOI http://doi.acm.org/10.1145/1068009.1068241

Pabst J, Jense H (1995) Dynamic terrain generation based on multifractal techniques. In: Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualisation, Swansea

Peitgen HO, Jürgens H, Saupe D (2004) Chaos and Fractals - New Frontiers of Science, 2nd edn. Springer

Perlin K (1985) An image synthesizer. SIGGRAPH Comput Graph 19(3):287–296, DOI http://doi.acm.org/10.1145/325165.325247

Perlin K (2002) Improving noise. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, pp 681–682, DOI http://doi.acm.org/10.1145/566570.566636

Pi X, Song J, Zeng L, Li S (2006) Procedural terrain detail based on patch-LOD algorithm. In: Pan Z, Aylett R, Diener H, Jin X, Göbel S, Li L (eds) Technologies for E-Learning and Digital Entertainment, Lecture Notes in Computer Science, vol 3942, Springer Berlin / Heidelberg, pp 913–920, URL http://dx.doi.org/10.1007/11736639_111, 10.1007/11736639_111

Poli R, Langdon WB, McPhee NF (2008) A field guide to genetic programming. Lulu.com, URL http://www.gp-field-guide.org.uk, (With contributions by J. R. Koza)

Pouderoux J, Gonzato JC, Tobor I, Guitton P (2004) Adaptive hierarchical RBF interpolation for creating smooth digital elevation models. In: GIS '04 - 12th annual ACM international workshop on Geographic information systems, ACM, New York, NY, USA, pp 232–240, DOI http://doi.acm.org/10.1145/1032222.1032256

Prusinkiewicz P, Lindenmayer A (2004) The Algorithmic Beauty of Plants. Springer-Verlag

Remo C (2008) MIGS: Far Cry 2's Guay On The Importance Of Procedural Content. Website (accessed on Sep. 2011), http://www.gamasutra.com/php-bin/news_index.php?story=21165

Rodrigues N, Frade M, Fernández de Vega F (2010) Development of chapas an open source video game with genetic terrain programming. In: VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), Valencia, Spain,, pp 1–8

Sakas G (1993) Modeling and animating turbulent gaseous phenomena using spectral synthesis. The Visual Computer: International Journal of Computer Graphics 9(4):200–212, DOI \url{http://dx.doi.org/10.1007/BF01901724}

Sampath D (2004) ABRCon, Adaptive oBject ReCONfiguration: an approach to enhance, repeat playability of games and repeat watchability of movies. In: ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology, ACM, New York, NY, USA, pp 313–316, DOI http://doi.acm.org/10.1145/1067343.1067388

Schneider J, Boldte T, Westermann R (2006) Real-time editing, synthesis, and rendering of infinite landscapes on GPUs. In: Vision, modeling, and visualization 2006: proceedings, November 22-24, 2006, Aachen, Germany, IOS Press, p 145

Sims K (1991) Artificial evolution for computer graphics. In: SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques, ACM, NY, USA, pp 319–328, DOI http://doi.acm.org/10.1145/122718.122752

Smelik RM, Tutenel T, de Kraker KJ, Bidarra R (2010) Declarative terrain modeling for military training games. International Journal of Computer Games Technology 2010:11, DOI 10.1155/2010/360458, article ID 360458

Stachniak S, Stuerzlinger W (2005) An algorithm for automated fractal terrain deformation. Computer Graphics and Artificial Intelligence 1:64–76

Stanley K (2007) Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines 8:131–162, URL http://dx.doi.org/10.1007/s10710-007-9028-8

Szeliski R, Terzopoulos D (1989) From splines to fractals. SIGGRAPH Comput Graph 23(3):51–60, DOI http://doi.acm.org/10.1145/74334.74338

Togelius J, Schmidhuber J (2008) An experiment in automatic game design. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG), pp 111–118

Togelius J, Preuss M, Beume N, Wessing S, Hagelback J, Yannakakis G (2010a) Multiobjective exploration of the starcraft map space. In: Computational Intelligence and Games (CIG), 2010 IEEE Symposium on, IEEE, pp 265–272

Togelius J, Preuss M, Yannakakis GN (2010b) Towards multiobjective procedural map generation. In: PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, ACM, New York, NY, USA, pp 1–8, DOI http://doi.acm.org/10.1145/1814256.1814259

Togelius J, Yannakakis GN, Stanley KO, Browne C (2011) Search-based procedural content generation: A taxonomy and survey. Computational Intelligence and AI in Games, IEEE Transactions on 3(3):172 – 186, DOI 10.1109/TCIAIG.2011.2148116

Tu SC, Huang CY, Tai WK (2008) Terrain synthesis based on microscopic terrain feature. In: Pan Z, Zhang X, El Rhalibi A, Woo W, Li Y (eds) Technologies for E-Learning and Digital Entertainment, Lecture Notes in Computer Science, vol 5093, Springer Berlin / Heidelberg, pp 644–655, URL `http://dx.doi.org/10.1007/978-3-540-69736-7\_69`

Unemi T (1998) A design of multi-field user interface for simulated breeding. In: Proceedings of the Third Asian Fuzzy and Intelligent System Symposium, Masan, Korea, pp 489–494

Vemuri B, Mandal C, Lai SH (1997) A fast gibbs sampler for synthesizing constrained fractals. Visualization and Computer Graphics, IEEE Transactions on 3(4):337–351, DOI 10.1109/2945.646237

Voss R (1987) Fractals in nature: characterization, measurement, and simulation. SIGGRAPH

Wells WD (2005) Generating enhanced natural environments and terrain for interactive combat simulations (genetics). In: VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology, ACM, New York, NY, USA, pp 184–191, DOI http://doi.acm.org/10.1145/1101616.1101655

Zhou H, Sun J, Turk G, Rehg JM (2007) Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics 13(4):834–848, DOI http://dx.doi.org/10.1109/TVCG.2007.1027, member-Turk, Greg and Member-Rehg, James M.