Interactive Procedural Building Generation Using Kaleidoscopic Iterated Function Systems

Tim McGraw

Purdue University

Abstract. We present an approach to designing and generating buildings at interactive rates. The system can run entirely on the GPU in a fragment shader and results can be viewed in real time. High quality raycast or raytraced results can be efficiently visualized because the buildings are represented as distance fields. By exploiting the visual complexity of a class of fractals known as kaleidoscopic iterated function systems (KIFS) we can generate detailed buildings reminiscent of ornate architectural styles, such as Gothic and Baroque, with simpler rules than grammar based methods.

1 Introduction

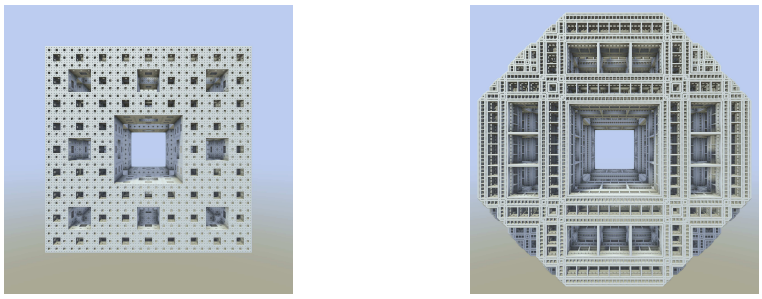
Many fractal images and surfaces are characterized by complex patterns that repeat over multiple scales. Techniques such as escape-time formulas and iterated function systems (IFS) can be used to create and explore these patterns. Familiar fractals, such as the Sierpinski gasket and Menger sponge, have a structure that can be easily deduced from the geometric rules for their construction. The Menger sponge is constructed by starting with a single cube which is subdivided into 27 equal sized smaller cubes. The cube from the center of the original cube and the cubes from the middle of each face of the original cube are removed. This process is repeated indefinitely for each remaining cube. At all scales the sponge has the characteristic perforated box-like appearance shown in Figure (1). An IFS construction of the sponge requires recursively applying affine mappings to a set of points. It can also be generated using distance estimation methods which rely on folding and scaling operations. But, as we shall see, a variety of complex shapes can be obtained by subtly changing the generation rules. It is less intuitive how the recursive construction of these modified Menger sponges lead to the resulting array of patterns.

For many 3D fractals it is possible to estimate the distance to the surface from a given point. This permits accelerated ray tracing algorithms and extraction of isosurfaces of the distance estimate. Modeling arbitrary structures with fractals requires the solution of a difficult inverse problem: finding the iterative process which results in a given shape. The Collage Theorem [1] states general conditions for the existence of a solution to this inverse problem and describes a general approach to its construction. In 2D this has led to fractal image compression algorithms, but general 3D solutions have been elusive.

Our approach to utilizing fractals in modeling 3D buildings is to warp a selected volume of an IFS fractal into another volume bounded by a rough building shape. In this work a periodic mapping with local symmetry is specified in terms of mod functions. The resulting shape can be visualized using ray tracing, or by rasterizing a triangle mesh generated by isosurface extraction. Our approach permits the user to select building parameters and instantly see the resulting structure.

2 Related Work

Procedural modeling methods permit the generation of graphical content (e.g. meshes or textures) by means of automatic or interactive algorithms. The reader is referred to the survey by Smelik et al. [2] for an overview of the various approaches to generating natural phenomena (such as terrain, plants, bodies of water) and man made objects (buildings, roads, cities). The scale and scope of building generation methods ranges from entire cities [3], to individual building facades [4], and building interior layouts.



(a) (b)

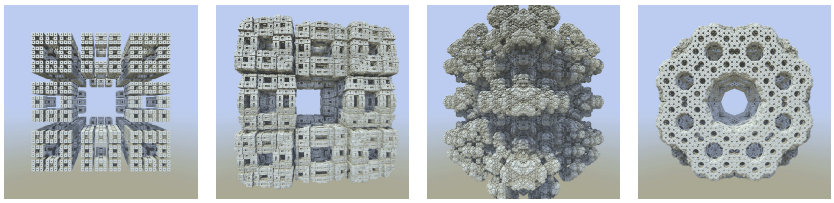
Fig. 1: Menger sponge (a) and KIFS (b) with $c = (0.93, 0.93, 0.33)$

Much previous work on building generation has focused on grammar-based approaches [5, 6]. A common framework is to extrude a footprint shape into volume mass-model that defines the overall shape of the building. Then a sequence of substitution rules governs how that model is divided into floors and how each floor is divided to produce the building facade. The substitution process terminates at simple primitives such as bricks, windows and doors. By contrast, our method uses a simple arithmetic equation to divide the facade and the visual complexity is achieved by using regions of fractals as our terminal primitives.

Our building modeling method is similar to the shape modeling process described by McGraw and Herring [7], with several important distinctions. We use a specific class of fractal IFS that is well-suited to creating building detail, rather than the Mandelbox and Mandelbulb fractals. The periodic and symmetric structure of most buildings, along with the simple bounding volumes that can

be expressed as unions of geometric primitives simplifies the process of mapping the fractal onto the surface. By contrast, McGraw and Herring require the user to interactively define a spline-based warp from the fractal domain onto a mesh by positioning individual vertices.

Iterated function systems [8] are a method of generating points in a fractal set by taking an input point set and repeatedly applying affine transformations to it. Early computer graphics applications of IFSs [9] showed that relatively small sets of transformations could approximate natural objects, such as leaves and ferns as well as reproducing classical fractals such as the Cantor set and Sierpinski gasket.



(a) (b) (c) (d)

Fig. 2: KIFS with $s = 3.5$ (a), $R_1 = R_y(\pi/25)$ (b), $R_1 = R_y(\pi/4)$ (c), and $R_2 = R_y(\pi/8)$

The Kaleidoscopic IFS (KIFS) fractal described in algorithm (1) was developed by Knighty [10] while developing distance estimates for the 3D fractal Sierpinksi tetrahedron and Menger sponge. The operations can be be seen as an iterated sequence of folding operations (lines 5-10), rotations (lines 3, 11) and uniform scaling about a center point given by x_c, y_c, z_c (lines 12-14). This algorithm generates the Menger sponge for $x_c = y_c = z_c = 1$, $s = 3$ and $R_1 = R_2 = I$. For other parameter values a rich set of features emerges, both organic and synthetic looking, depending on parameter values. As can be seen in Figure (2) the surface becomes sparse and disconnected for values of $s > 3$. For R_1 with small rotation angles the surface becomes less regular and resembles an ancient crumbling structure. Matrix R_2 can change the rectilinear structure into one with polygonal and star-like features. As with most fractal systems, a good way to get a sense of the range of shapes is to experiment and explore the parameter space. This is facilitated by a fast GPU implementation and a UI which permits parameter specification.

Hart et al. [11] introduced the idea of determining bounds on the distance to a fractal surface to accelerate ray tracing. Knowing that the distance to a surface is *at least* d we can safely step along a ray by distance d when iteratively searching for the ray-surface intersection. The search is terminated when d falls below some threshold. The process of real time rendering using such a raycasting process is described by Quilez [12] in the context of modern graphics hardware.

Algorithm 1 Algorithm for computing the distance estimate to a KIFS fractal from the point (x, y, z). The scale center parameters, x_c, y_c, z_c, and scale factor, s, are scalar values, and R_1, R_2 are rotation matrices. In our experiments we use maximum iterations $M = 6$ and bailout threshold $b = 1.5$.

function $d_{KIFS}(x, y, z, x_c, y_c, z_c, s)$
  for $i = 0$ to M **do**
    $[x\, y\, z]^T = R_1[x\, y\, z]^T$
    $x = |x|$, $y = |y|$, $z = |z|$
    if $x - y < 0$ **then** swap(x,y)
    end if
    if $x - z < 0$ **then** swap(x,z)
    end if
    if $y - z < 0$ **then** swap(y,z)
    end if
    $[x\, y\, z]^T = R_2[x\, y\, z]^T$
    $x = s(x - x_c) + x_c$, $y = s(y - y_c) + y_c$, $z = sz$
    if $z < z_c(s - 1)/2$ **then** $z = z - z_c(s - 1)$
    end if
    $r = x^2 + y^2 + z^2$
    if $r > b$ **then** break
    end if
  end for
return $(r^{1/2} - 2)s^{-i}$
end function

The distance function representation also allows us to easily perform constructive solid geometry (CSG) operations on shapes.

3 Methods

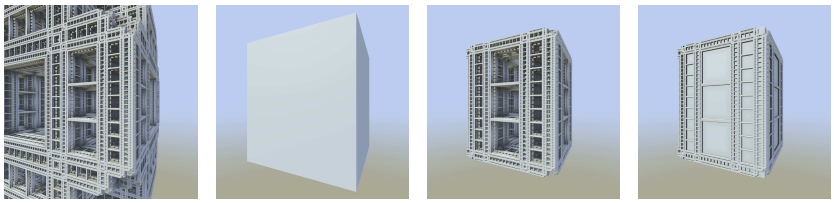
Our building creation system is integrated into a realtime raycasting renderer, and the buildings are volumetrically represented as distance functions. Rays are traced in a glsl fragment shader until they intersect the building, and then lighting is computed. Building mass models are built from simple primitives, such as 3D boxes. A box primitive has distance function

$$\max(|x| - h_x, |y| - h_y, |z| - h_z) \tag{1}$$

where x, y, z are point coordinates and h_x, h_y, h_z are the half-widths of the box along the x,y,z axes.

We give our users the selection of several building mass models created from CSG operations on boxes. The union ($\cup$) and intersection ($\cap$) operations are given by

$$\cup(A, B) = \min(d_A, d_B) \tag{2}$$
$$\cap(A, B) = \max(d_A, d_B), \tag{3}$$



(a) (b) (c) (d)

Fig. 3: KIFS detail (a), mass model outer shell (b), intersection of KIFS and outer shell $\cap(KIFS, Outer)$ (c), union of previous result with inner shell $\cup(Inner, \cap(KIFS, Outer))$

where A, B are shapes and d_A, d_B are distances to A and B.

The mass models used in our system consist of an inner and outer shell. The outer shell is the outermost extent of the building. Since we will later define an infinite tiling of the KIFS fractal, computing the intersection with this outer shell restricts the building to a finite domain. The inner shell represents windows and external walls. Computing the union of the fractal and the inner shell prevents the user from seeing into the interior of the fractal, as demonstrated in Figure (3).

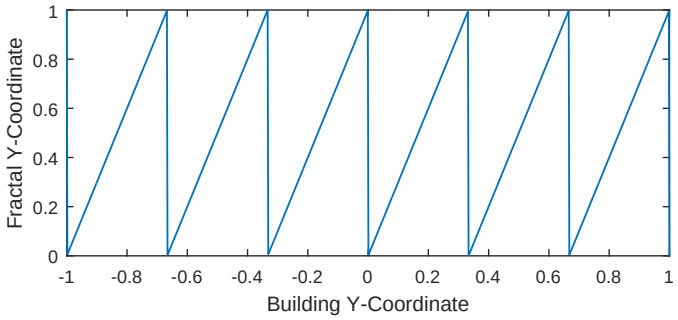


Fig. 4: Y-coordinate mapping function $f_2(y)$

Our building generation technique is based on distance and coordinate transformation of the KIFS system described in algorithm (1). If the distance to the fractal surface is given by $d_{KIFS}(x, y, z)$ then the distance to the modeled building is $g(d_{KIFS}(f_1(x, z), f_2(y), f_3(x, z)))$ where y is the vertical axis of the structure. The distance transformation $g()$ is a series of CSG operations which define the architectural mass model. Functions $f_1(x, z)$, $f_2(y)$ and $f_3(x, z)$ are coordinate transformations which map regions of the KIFS to the surface of the building.

The vertical coordinate transformation is given by

$$f_2(y) = s_y \bmod (y - y_0, h)/h \tag{4}$$

where the mod operation results in a periodic repetition of a part of the fractal which creates the stories or levels of the building. The parameters y_0, h define a vertical slab of the KIFS, and s_y defines how that slab is mapped onto the building. In Figure (4) a graph of $f_2(y)$ is shown for a 6 story building with $s_y = 1, y_0 = 0, h = 1/3$.

The building facade within each level is also periodic, but should permit symmetry and repetition of regions. These features are realized by a more complicated function using mod functions which is given in Algorithm (2). The symmetry is evident in the appearance of triangle waves in the plot of fractal coordinates in Figure (5), as opposed to the nonsymmetric repetition represented by the sawtooth waves in Figure (4).

Algorithm 2 Algorithm for computing the building transformed coordinates, $x_f = f_1(x, z), z_f = f_3(x, y)$, and facade ids, id_x, id_z. The coordinates being transformed are x, z; w_x, w_z is the repetition period of the pattern on the x- and z-faces of the building, and $\mathbf{r_x} = (r_{x,1}, r_{x,2}, r_{x,3})$ control the widths and ids within the pattern on the building x-faces. $\mathbf{r_z}$ operates similarly.

function $f_{xz}(x, z, w_x, w_z, \mathbf{r_x}, \mathbf{r_z})$
  $x_f = |2 \bmod (x, w_x)/w_x - 1/2|$
  $z_f = |2 \bmod (z, w_z)/w_z - 1/2|$
  $id_x = 0, id_z = 0$
  for $i = 1$ to 3 **do**
    $x_f = |x_f - r_{x,i}| - r_{x,i}$
    $z_f = |z_f - r_{z,i}| - r_{z,i}$
    if $x_f > 0$ **then** $id_x = id_x + 1$
    end if
    if $z_f > 0$ **then** $id_z = id_z + 1$
    end if
    $x_f = |x_f|$, $z_f = |z_f|$
  end for
return x_f, z_f, id_x, id_z
end function

An optional transformation of x, z coordinates supported by our system is conversion to polar coordinates to create curvilinear building shapes. This transformation, which generates cylindrical and curved buildings, is given by

$$x' = s_\theta(\arctan(z, x) + c_\theta) \tag{5}$$

$$z' = s_r(\sqrt{x^2 + (z - z_0)^2} + c_r), \tag{6}$$

where c_θ is a rotation angle, s_θ determines how much of a circular arc the building subtends, c_r, s_r control the inner and outer radii of the building. Examples of the curvilinear building mass model and facade ids are shown in Figure (6).

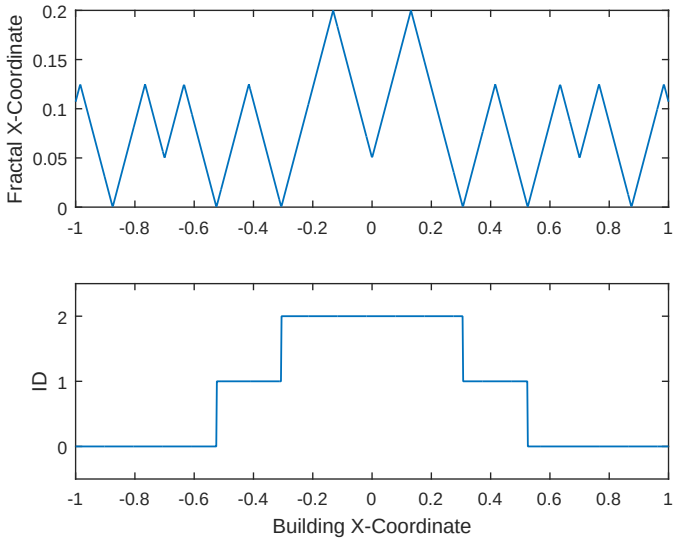
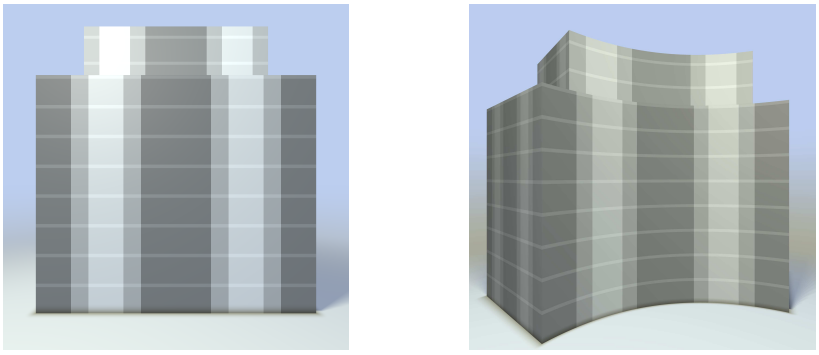


Fig. 5: Fractal coordinates (top) and facade ids (bottom) computed from building coordinates.



(a) (b)

Fig. 6: Building facade ids and floors colored on rectilinear mass model (a) and curvilinear mass model (b) for $\mathbf{r}_x = \mathbf{r}_z = (0.25, 0.125, 0.4)$.

The x_f, z_f coordinates are subject to further scaling and translation based on the building facade id, and then the distance function to the KIFS is evaluated. These id dependent transformations permit each facade region to have a different appearance by selecting from a different region of the KIFS fractal.

Finally, the distance to the fractal is processed using CSG operations. Let the building mass model be represented by inner and outer shells, and let the distance to those shells be given by d_i, d_o respectively. Then $g(d_{KIFS}, d_i, d_o) = \min(d_i, \max(d_o, d_{KIFS}))$. An outline of the entire modeling process is summarized below.

1. Compute inner and outer shell distances d_i, d_o to building mass model.
2. Optionally warp ray coordinates (x, y, z) to polar coordinates.
3. Compute $x_f = f_1(x, z), y_f = f_2(y), z_f = f_3(x, z)$ and facade ids.
4. Perform facade dependent translation and scaling $x_f = s_x x_f + t_x, z_f = s_z z_f + t_z$.
5. Compute KIFS distance, d_{KIFS}, to (x_f, y_f, z_f).
6. Perform CSG operations with building mass model $g(d_{KIFS}, d_i, d_o)$.

4 Results

Our KIFS building generation system was implemented in C++ and OpenGL on a Dell Optiplex workstation with 3.4 GHz Intel Core i7-3770 CPU and 8GB RAM, Nvidia GeForce GTX 750 Ti with 640 shader cores and 2 GB GDDR5 dedicated video RAM.

The set of parameters which determine the building appearance, and the values used by our system are given below:

- Select mass model from list (3 choices)
- Enable/disable curvilinear coordinates
  - If enabled pick $s_\theta, c_\theta, s_r, c_r$
- Pick $w_x, w_z, \mathbf{r}_x, \mathbf{r}_z$ for facade division
  - We simplify by letting $w_x = w_z$ and $\mathbf{r}_x = \mathbf{r}_z$ which make the x- and z-faces the same.
- Pick (s_x, t_x, s_z, t_z) for each id for facade transformation
  - We simplify by letting $s_x = s_z = a_1 id + a_2, t_x = t_z = b_1 id + b_2$
- Pick KIFS parameters $(x_c, y_c, z_c, s, R_1, R_2)$.
  - We fix $s = 3, R_1 = I, R_2 = R_y(\theta)$

This results in 13 parameters to specify a rectilinear building, and 4 more for a curvilinear building.

KIFS Buildings generated in rectilinear and curved coordinates are shown in Figures (7,8,9). Results were generated and rendered at between 24ms and 133 ms per frame (about 8 to 42 fps) in a 640×640 viewport.

The technique we presented has several limitations. Roofs and other features are not automatically handled. Physically impossible structures with floating blocks can be created. Our system can, however, be extended to handle more general building styles, such as those with a ground floor which doesn't match the appearance of the other floors, at the expense of requiring more user input.

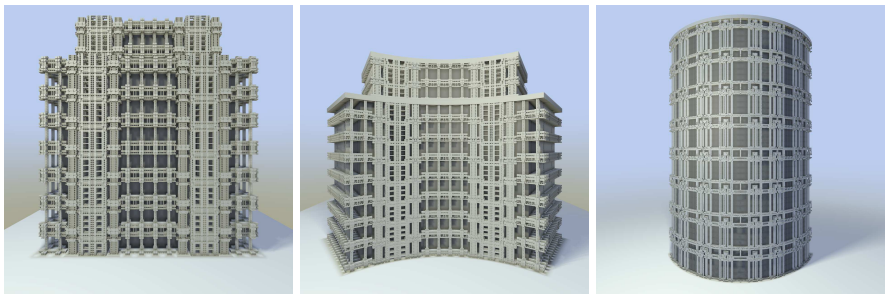


Fig. 7: Rectilinear (left) and curved (middle, right) KIFS buildings

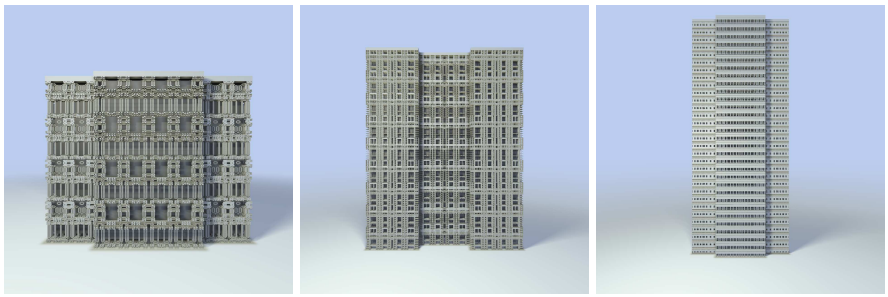


Fig. 8: KIFS buildings with t-shaped (left, right) and h-shaped mass model footprints.

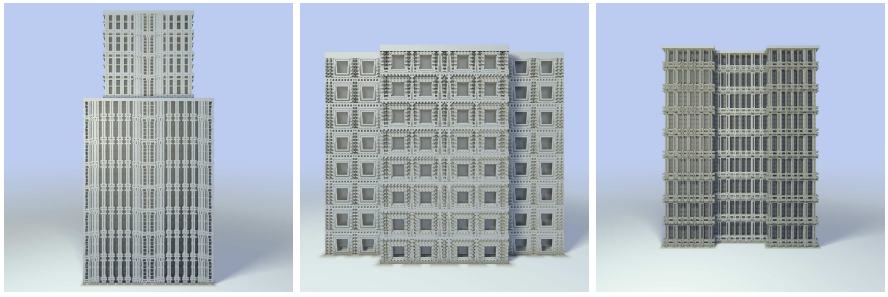


Fig. 9: Additional KIFS building results

5 Conclusion and Future Work

In this paper we have described a method for procedurally generating complex buildings by harnessing the ornate architectural patterns generated by kaleidoscopic iterated function systems (KIFS). We described a domain transformation method which permits periodic patterns and symmetry to be specified and controlled by the designer. Our system can interactively generate and raycast the resulting structures entirely on the GPU. The methods can be implemented in a few hundred lines of fragment shader code. Preliminary results show that this technique can efficiently produce plausible buildings. Future work will involve developing additional tools and interfaces to support building generation; improving support for irregular features, such as entrances and roofs; and user studies to assess the usability of the system.

References

1. Barnsley, M.F., Ervin, V., Hardin, D., Lancaster, J.: Solution of an inverse problem for fractals and other sets. Proceedings of the National Academy of Sciences of the United States of America **83** (1986) 1975–1977
2. Smelik, R.M., Tutenel, T., Bidarra, R., Benes, B.: A survey on procedural modelling for virtual worlds. Computer Graphics Forum **33** (2014) 31–50
3. Demir, I., Aliaga, D.G., Benes, B.: Proceduralization of buildings at city scale. In: 2014 2nd International Conference on 3D Vision (3DV). Volume 1., IEEE (2014) 456–463
4. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. In: ACM Transactions on Graphics. Volume 26., ACM (2007) 85
5. Aliaga, D.G., Rosen, P., Bekins, D.R.: Style grammars for interactive visualization of architecture. IEEE Transactions on Visualization and Computer Graphics **13** (2007) 786–797
6. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. ACM Transactions On Graphics **25** (2006) 614–623
7. McGraw, T., Herring, D.: Shape modeling with fractals. In: Advances in Visual Computing. Springer (2014) 540–549
8. Barnsley, M.F., Demko, S.: Iterated function systems and the global construction of fractals. In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. Volume 399., The Royal Society (1985) 243–275
9. Demko, S., Hodges, L., Naylor, B.: Construction of fractal objects with iterated function systems. In: ACM SIGGRAPH Computer Graphics. Volume 19., ACM (1985) 271–278
10. Knighty: Kaleidoscopic (escape time) IFS. http://www.fractalforums.com/ifs-iterated-function-systems/kaleidoscopic-(escape-time-ifs) (2010)
11. Hart, J.C., Sandin, D.J., Kauffman, L.H.: Ray tracing deterministic 3-d fractals. ACM SIGGRAPH Computer Graphics **23** (1989) 289–296
12. Quilez, I.: Modeling with distance functions. http://www.iquilezles.org (2008)
13. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games **3** (2011) 172–186