ORIGINAL SCIENTIFIC PAPER
RECEIVED: 16-04-2013
ACCEPTED: 30-08-2013

ACTA GRAPHICA 218

Shape Metamorphosis – Automatic 3D Mesh Generation, Topology Verification and Analysis

Authors

Tomasz Zawadzki*, Dominik Kujawa

University of Zielona Góra
Institute of Control & Computation Engineering
Poland
E.-mail: t.zawadzki@weit.uz.zgora.pl

Abstract:

The objective of this paper is a 3D shape construction that benefits from discrete and continuous modelling approaches. The proposed solution addresses the problem of automated modelling of virtual structures such as caves, buildings and clouds and presents an alternative solution in the form of a hybrid system. Parallel realizations of these solutions are tested on various processors of graphic cards with the use of NVIDIA 'CUDA' technology. This paper describes the implementation of algorithms (approaches) and their parallel speedup, efficiency, throughput. Modelled structures are geometrically complex, with an inner graph structure more optimized than in the classical CSG approach. Moreover, they can be rendered up to very high levels of visual realism. In this paper we mainly focus on the description of the algorithm. We also propose very useful measures that can be used to verify the model geometry.

Keywords:

Procedural Modelling, Shape Grammar, Morphing, Hybrid Measures·

1. Introduction

Complex structures of buildings (*Wonka et al., 2003*), whole urban structures (*Parish et al., 2001, Greuter et al., 2003*), terrains (*Peytavie et al, 2009, Warszawski et al., 2009*), clouds (*Bouthors et al., 2004, Schpok et al., 2003, Dobashi et al., 2000, Ebert et al., 1997, Elinas et al., 2000, Nishita et al., 1996*), plants (*Prusinkiewicz et al., 1991*) or caves (*Am Ende et al., 2001, Boggus et al., 2009, Schuchardt et al., 2007, Johnson et al., 2010*) can be modelled with systems based on automated shape formation. The algorithms that enable full automation of this process also help to achieve large savings of the digital media production time. We can observe a constant development of new methods i.e. merging technology and dynamical systems (*Clempner et* *al., 2011, Di Trapani et al., 2010*). As we can see clearly, the problem of automated shape modelling constitutes an important area of computer graphics activity and has drawn attention of digital media industry for several years. Digital movies have created constant demand for pleasing visual effects in three-dimensional graphics. Apart from the pure entertainment purposes, shape modelling has the practical use ranging from CAD engineering applications through scientific visualization to advanced game programming and Virtual Environments.

2. Shape grammar and morphing - introduction

Stiny and Gips are the precursors of shape grammars. Their research was aimed at supporting the design process using a "linguistic model of the generational system" (*Stiny et al., 1972, 1975*). The definition of shape grammars is analogous to the formal grammars and is graphically expressed in the language of words composed of symbols with different grammatical rules (*Stiny, 1980*).

Generally, morphing is usually defined as the continuous and smooth process of transformation of one shape into another. The so-called key shapes (by analogy to key-framed animation) may have different topologies and the smoothness of transformation does not have to be the case of homeomorphism (Martyn, 2004).

There is a number of metamorphosis methods for 2D shapes represented mostly by polygons (*Alexa et al., 2000*), but also by images (*Wolberg, 1998*). The problem has also been investigated in the 3D domain and according to Lazarus (*Lazarus et al., 1998*), it spans the methods based on polygonal mesh representation (*Kent et al., 1992, Lee et al., 1999*) and the methods utilizing voxel representations (*Turk et al., 1999*).

3. Parallelization level estimating indicators

System architecture can be adapted to a particular computational task in itself is not a determinant of performance or efficiency.

$$y_k(p) = F^l(u_k(p) \mid F^{l-1}(u_k(p) \mid \ldots \mid F^1(u_k(p) \quad (1)$$

where:
y – number of cubes inside,
k – quantity of sequential cubes inside,
u – cube edge
p – the point with coordinates (x, y, z) in Euclidean R3
| – executing operations simultaneously
F^δ – operator, $\delta \in \{1,\ldots,l\}$.

Many operations can be performed simultaneously during the position points calculation for different productions. The dedicated computing parallelism is also used, although there are no streams of instructions. The parallelization is significantly facilitated by eliminating the communication overhead for the acquisition of stream commands.

$$S_m = \frac{T_1}{T_m} \quad (2)$$

where:
S_m – the speedup ,
S_1 – the execution time of the sequential algorithm,
T_m – the execution time of the parallel algorithm with *m* processors

For the evaluation of computational tasks performance in multi-core systems speedup formula was used (4).

In multiprocessor systems, the actual value of the speedup is smaller than the theoretical (based on the number of processors) due to communication overheads and the need to share resources such as memory and buses. This property describes a model of efficiency, which is a measure of the concurrent use of resources

$$E_m = \frac{S_m}{m} \quad (3)$$

where:
E_m – the speedup efficiency with m processors $E_m \in (0,1\rangle$.

In the ideal case, the efficiency is 1.0 which means that the speedup S_m is proportional to the number of processor m or computing elements.

In the assessment of computing, activity is often used as a criterion of time needed to process a specific task or quantum computing. The concept of throughput was introduced, which corresponds to the number of data processed per time unit.

$$P = \frac{D}{T} \qquad (4)$$

where:
P – throughput,
D – the number of computed data
T – measurement time.

Relative to the computing system has the ability to compute the limit, usually in the long term.

Throughput relative to the computing system has the ability to compute the limit, usually in the long term.

The tables 1, 2 and 3 show the comparison of the indicators of speedup, throughput and the delay of parallel performance algorithm for the sequential algorithm

A – cave generation

B – cloud generation

C – building generation

Table 1. Evaluation of the parallel algorithm - speedup for m=255

Parallel algorithm			
Stage	A	B	C
	0.096[s]	0.125[s]	0.069[s]

Sequential algorithm			
Stage	A	B	C
	0.81[s]	0.62[s]	0.54[s]

Table 2. Evaluation of the parallel algorithm – efficiency for m=255

Efficiency (m = 255)			
Stage	A	B	C
	0.028	0.015	0.027

Table 3. Evaluation of the parallel algorithm – throughput for m=255

Throughput (m = 255)	
Sequential algorithm	Parallel algorithm
0.8[GFlop/s]	20.61[GFlop/s]

The algorithms were tested on a PC computer with Genuine Intel ® CPU T2300 @ 1.66 GHz, and 2512 MB RAM NVIDIA GTX 460 graphic card.

Figure 1 shows the speedup of a parallelized shape morphing system and in Figure 2 shape hybrid system speedup as a function of the number of measurements of α (α is set of the position points).

The points were changed from 5 to 700 and the run time for the sequential shape morphing system and the shape hybrid system and their parallel versions were measured.

The speed up was evaluated on the base of these measurements.

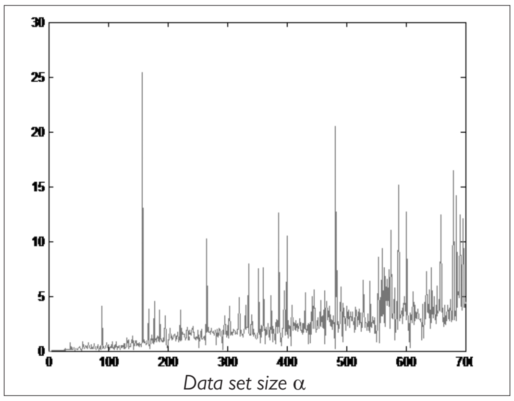

Figure 1. Speedup of the parallel shape morphing system

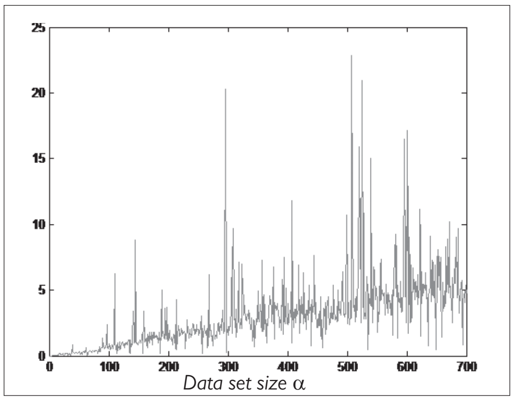

Figure 2. Speedup of the parallel shape hybrid system

Speedups, shown in Figure 1 and 2, are equal to formally calculated speedups of parallel algorithms. Parallel implementations run about 5-6 times faster than their sequential versions.

4. The idea of morphic rules

Each of the rules of a shape is composed of expressions found on the left side (*LHS, Left-Hand Side shape*) and those on the right side (*RHS, Right-Hand Side shape*) (Fig.3).

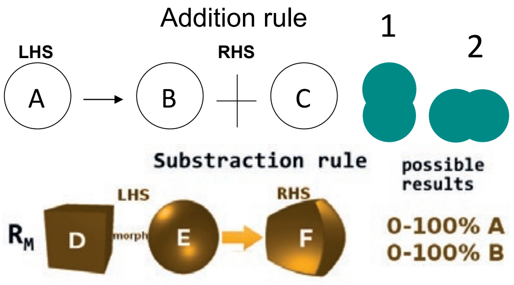

Figure 3. An example of addition and substraction rules.

Computation (derivation) of shape grammars is based on processing the rules (from initial shape to last rule):

$$I_0 \Rightarrow R_1 \Rightarrow R_2 \Rightarrow \ldots \Rightarrow R_m \qquad (5)$$

In our case the initial shape (I_o) is a point which is summed with the shape randomly selected from a database of non-terminal shapes (N) – sphere, torus, cuboid, cone, cylinder etc.

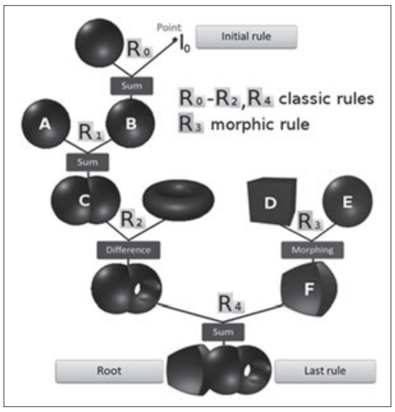

Figure 4. Example of realization classic and morphic rules (one variant).

The base of terminal shapes (T) is larger than that of non-terminal shapes, because it is dynamically created during the modelling process. In subsequent rules (R), basic shapes evolve into new structures created using Bwoolean operations or morphing into another shapes.

The classic shape grammar is based on the following operations: sum, difference, and intersection and a new innovative approach involves the addition of morphing based on linear interpolation (Fig. 4.).

We can describe the above example as follows:

$$I_0 \Rightarrow R_1 \Rightarrow R_4,\ R_2 + R_3 \Rightarrow R_4$$

Shape deformation is calculated based on morphic rules, which are estimated on percentage contributions of the current (base shape), and added shape which is called morphing parameter (M_P range from 0-100%) (Fig. 5).

Classic productions are based on addition and subtraction rules, while morphic productions are based only on subtraction rules. This work is focused on procedural modelling of shapes by extending scene graph by new grammatical rules.

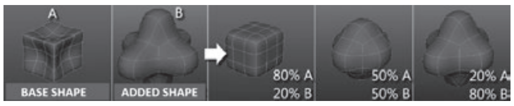

Figure 5. Presentation of the percentage contribution of shape A and B on the result shape obtained with morphic production.

Definition 4. The total sum of all rules is given by:

$$R_T = R_C + R_M \qquad (6)$$

where:
RT – all rules,
RC – classic rules,
RM – morphic rules.

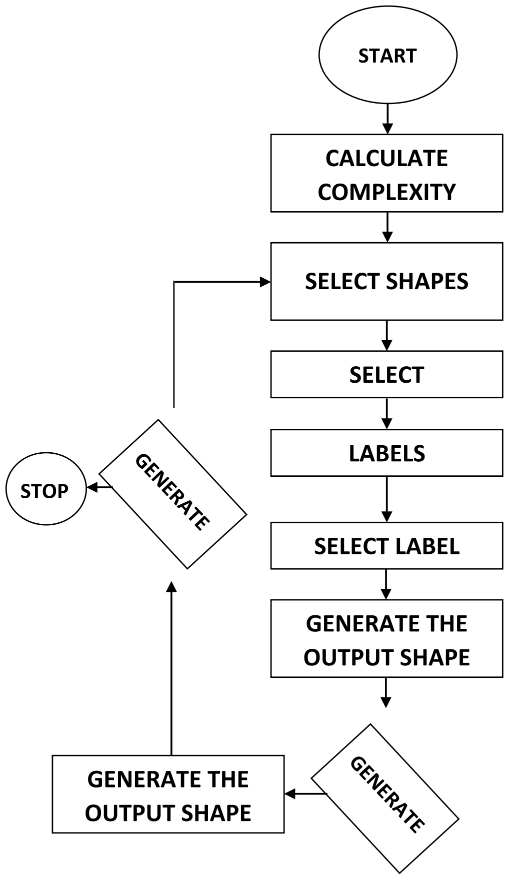

Figure 6. Main SG+M block diagram.

5. Shape grammar and morphing hybrid - introduction

The system SG+M is based on two independent algorithms. The first one, which is used to perform operations on shapes is very fast (Fig. 10.), while the second one which is used only to display the results (Marching cubes), is quite complex. Computational complexity of the first algorithm based on equation creation only (which will be useful in next step to show results) and we can't measure that time.

6. Visualization

The hybrid algorithm also facilitates the modelling of clouds, which is an implementation based on the summing-up of many spheres of different radius. The resulting cloud structure resembles a blob. In the final stage of rendering we can apply a volumetric shader (Fig. 7).

Morphic productions and morphing parameter customization open path to modelling a wider range of cave structures. It is possible to obtain results from cubical to spherical topology tunnels structures based on the morphing parameter. (Fig. 8).

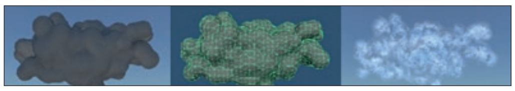

Figure 7. The obtained cloud structure: left - image render with lambert material, middle - 3D mesh of the cloud shape, right – image render with volumetric shader.

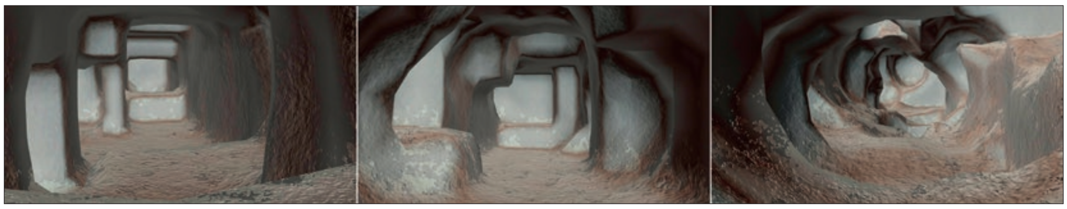

Figure 8. The obtained cave structures. Left – cubic, middle – various, right spheric.

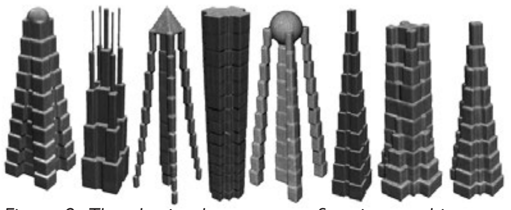
Figure 9. The obtained structures of various architecture objects types for LD = 10 and various settings for width, height and convexity for floors.

The algorithm offers three-dimensional modelling of various architectural structures illustrated in Fig. 9.

7. Geometry verification methods

In our research, we also considered development of valuable verification methods. We propose several metrics that can be used to verify the geometry of the rendered model.

7.1 Surface area of the model

After generating the grid, we obtain triangles that make up the model. For each triangle, we have three vertices (V_1, V_2, V_3) in the three-dimensional space described by *x*, *y*, and *z* coordinates. There is a formula for triangle surface that can be described on the basis of the vector product:

$$S = \tfrac{1}{2} * (V_2 - V_1) \times (V_3 - V_1) \qquad (7)$$

The relationship between the parameters of sampling density and the surface of the resulting structures possesses logarithmic characteristics (Fig 22). The obtained surface structure of these objects is in the range from 0 to some value, and is expressed in m² (Fig. 10).

7.2 Volume of the model

The volume is calculated by (Fig. 11):

$$V_M = N_C * (C_E)^3 \qquad (8)$$

where:
V_M – the volume of the model,
C_E – the cube edge *(C_E = 1/grid density)*,
N_C – numbers of cubes inside.

7.3 Independent model objects

For the generated mesh, all triangles are viewed and each creates a new group. If any of the vertices of one group overlaps with tips of another group, these groups are combined into one unit. The number of groups after reviewing all of the triangles is equal to the number of elements that make up the grid. Our research has shown that objects of cave, architecture and cloud topology mainly of one type of elements, but they may also consist of many different elements. It is most likely that a single type of object exists mostly in the case of architecture modelling, secondary for the caves and the least likely in the case of clouds modelling (Fig 12).

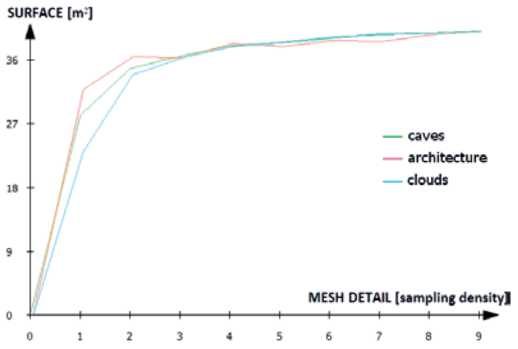

Figure 10. Comparison of surface vs. mesh density examples: top - cave, middle – cloud, bottom - building (LD, represented by sampling density)

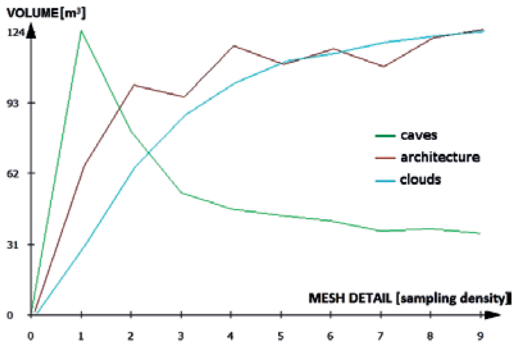

Figure 11. Comparison of volume vs mesh density examples: top - cave, middle – cloud, bottom - building (LD, represented by the sampling density).

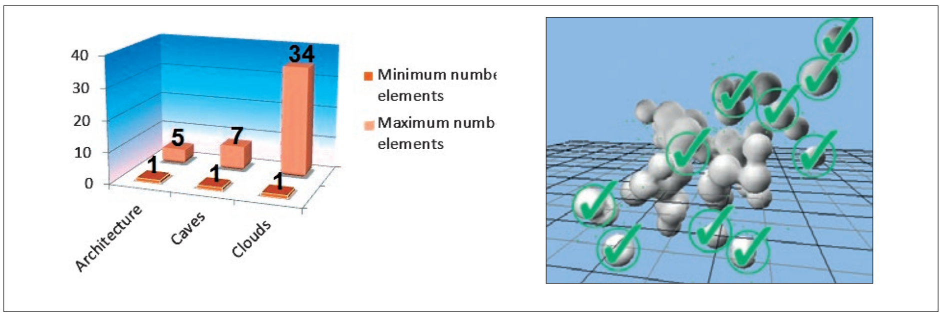

Figure 12. Left - independent model objects for: architecture, caves and clouds. Right - example of the distributed cloud with 10 separate elements.

7.4 HISTOGRAMS

The purpose of this stage is to generate a histogram based on the grid model. From our point of view, the model consists of edges. Each of them consists of two points (vertices). In addition, the vertices are assigned normal vectors. The first step in the preparation is a set of edges of all triangles that make up the model (if the edge belongs to two triangles is added twice, but perhaps with different normal vectors). For each edge, the value scaled to the interval [0, 1], the difference of two angles between normal vectors and the line containing the vertices is calculated (Fig. 13).

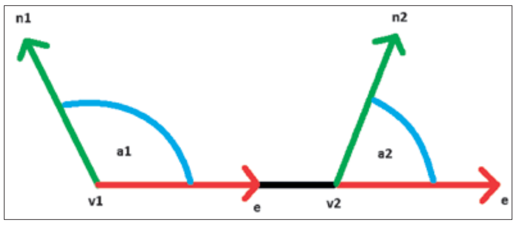

Figure 13. Calculation of the angles between the vertices.

$$e=\frac{(V_2-V_1)}{|V_2-V_1|} \tag{9}$$

$$a_1=\arccos(e\bullet n_1) \tag{10}$$

$$a_2=\arccos(e\bullet n_2) \tag{11}$$

$$d=a_2-a_1 \tag{12}$$

$$v=\frac{d}{2(P+1)} \tag{13}$$

where:
n1,n2 – normal vectors,
PI~3.14.

The obtained values belong to the interval [0,0.5), when the surfaces are concave or to the interval (0.5, 1], where the surfaces are convex. In the next step, a discrete histogram is created indicating the frequency of numbers from given interval. For each value from the set of interval, the histogram is calculated using the formula:

$$i=floor(v*R)+1 \tag{14}$$

where:
R – the number of histogram intervals.

The multiplicity of the i-th interval is incremented by one. After taking into consideration all values of the set, the histogram is normalized and all compartments are divided by the number of edges. Thus, the sum of the ranges of the histogram is equal to 1. For classification, it is necessary to make the histograms corresponding to master object classes (Fig. 28).

The verification of the correctness of the model depends on the density of the grid. The best results are obtained for L_D=*10* (Fig. 14).

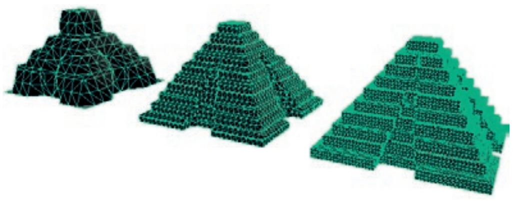

Figure 14. Level of detail (LD), from left LD=2, LD=5, LD=10.

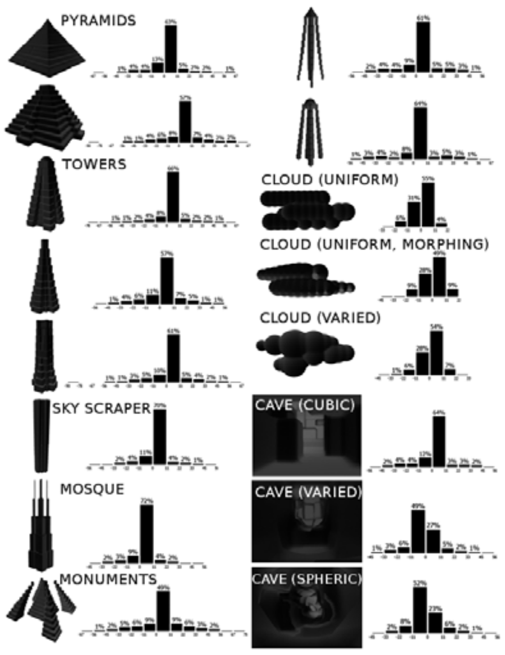

Figure 15. Histograms for the main classes of results.

When density is large, the histogram is reduced to a single bar, the most common value of angle.

The classification is based on the generation of a class histogram for new objects and its comparison to the master-histogram from the base, for which the mean square error is the smallest.

8. Time complexity

Computational complexity is the dependence of the algorithm execution time T (or the number of operations necessary to execute the algorithm) on the data set of position points (P), is the sphere radius (r), is a top radius (a), is a bottom radius (b)

$$T=f(P,r,a,b) \qquad (15)$$

It is difficult to determine the exact form of this function, which is why we use its asymptotic form.

The total number of basic operations - the number of assignments, sums, differences, intersections, comparisons that make up the complexity of the individual elements of the algorithm. The total computational complexity is composed of computational complexities:

- classical production
- morphing production

$$T_{ABC}(P,r,a,b) = P(r+r_0)(4a+0.5r(b+a))$$

Computational complexity of the shapes morphing system:

$$O(\alpha) = 41P(r+r_0+ab)[a+r(b+a)]^2$$

Computational complexity of the hybrid system shapes:

$$O(\alpha) = 14P(r+r_0+ab)[a+r(b+a)]^2$$

One of the most important aspects in the analysis of the results, shown in Fig. 18, is the time complexity or performance of the proposed method (Fig. 16):

All simulations were performed on nVidia GeForce GTX 460M GPU, i7-2630QM CPU and 12 GB RAM.

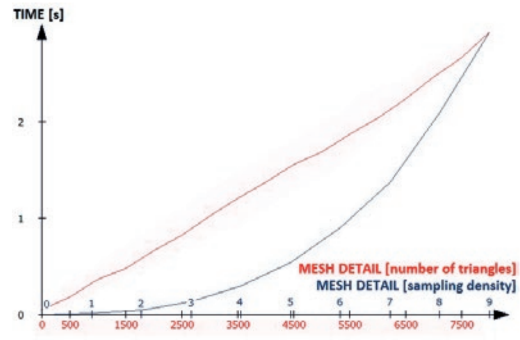

Figure 16. Time vs. mesh detail (LD, represented by sampling density) and number of triangles for the cave example shown in Figure 18.

Table 4. Comparison of methods

Methods	Geometry types			Steering parameters				Correctness mesh		
	Clouds	Architecture	Caves	LOD: mesh (M), noise (SZ), Simple shapes (P)	Modeling proces controls	Shape morphing	Texture mapping	Removing high surfaces	Removing overlapping verticals	No holes
Greuter's method	NO	YES	NO	YES (P)	YES (Extrusion)	YES	YES	NO	NO DATA	YES
Voxel method	NO	NO	YES	YES (SZ)	YES (Voxels)	NO	NO	YES	NO	YES
Cui's method	NO	NO	YES	YES (S)	YES (Voxels)	NO	NO	NO	NO	NO
Birger's method	YES	NO	NO	YES (SZ)	YES (Noise)	NO	YES	Does not apply	Does not apply	Does not apply
Marson's method	NO	YES	NO	YES (P)	YES (Tree map)	NO	NO	NO	YES	YES
Lipp's method	NO	YES	NO	YES(P)	YES (Shape grammar)	NO	YES	YES	NO DATA	YES
Cabral's method	NO	YES	NO	YES(P)	YES (Edge and texture transformation)	NO	YES	NO DATA	YES	YES
Bouthor's method	YES	NO	NO	YES(SZ)	YES (Anisotropic deffusion)	NO	YES	N/O	Does not apply	Does not apply
Mullera 's method	NO	YES	NO	YES(P)	YES (Shape grammar)	NO	NO	YES	YES	YES
System GK-M	YES	YES	YES	YES (S)	YES (Shape grammar, morphing, CSG)	YES	NO	YES	YES	YES

9. Conclusion

Nowadays, we observe a noticeable trend of applying new methods of modelling 3D objects in real-time computer graphics systems. This is forced mainly by the market demand created in areas of digital entertainment and simulation for 3D gaming. This paper presents an innovative method for real-time procedural modelling of three-dimensional geometry of caves, clouds, and buildings. By adding to the formalism of shape grammars an additional feature – morphing – we obtain a greater variety and geometric complexity of synthesized objects, highly influencing visual realism. This method is advantageous comparing it to other procedural methods.

References:

ALEXA M, COHEN-OR D, LEVIN D. As rigid as possible polygon morphing. Computers Graphics (SIGGRAPH '2000) 2000;34:157-64.

AM ENDE BA. 3D Mapping of Underwater Caves, IEEE Computer Graphics Applications 2001; 21(2):14-20.

BOGGUS M, CRAWFIS R. Procedural Creation of 3D Solution Cave Models, Proceedings of the 20th IASTED International Conference on Modelling and Simulation, 2009: 180-186.

BOGGUS M, CRAWFIS R. Explicit Generation of 3D Models of Solution Caves for Virtual Environments, Proceedings of the 2009 International Conference on Computer Graphics and Virtual Reality, 2009: 85-90.

BOUTHORS A, NEYRET F. Modelling Clouds Shape, Proceedings EUROGRAPHICS, 2004.

CLEMPNER JB, POZNYAK AS. Convergence method, properties and computational complexity for Lyapunov games, The international journal of Applied Mathematics and Computer Science, 2011; 21(2): 349-361.

DI TRAPANI LJ, INANC T. NTGsim: A graphical user interface and a 3D simulator for nonlinear trajectory generation methodology, The international journal of Applied Mathematics and Computer, 2010; 20(2): 305-316.

DOBASHI Y, KANEDA K, YAMASHITA H, OKITA T, NISHITA T. A simple, efficient method for realistic animation of clouds. Proceedings of ACM SIGGRAPH 2000, 2000; 19-28.

EBERT DS. Volumetric procedural implicit functions: A cloud is born. SIGGRAPH 97 Technical Sketches Program, Whitted T., (Ed.), ACM SIGGRAPH, Addison Wesley, 1997, ISBN 0-89791-896-7.

ELINAS P, STURZLINGER W. Real-time rendering of 3D clouds. Journal of Graphics Tools. 2000; 5(4):33-45.

GREUTER S, PARKER J, STEWART N, LEACH G. Real-time procedural generation of pseudo infinite cities. Proceedings (GRAPHITE'03), ACM Press, 2003; 87-95.

JOHNSON L, YANNAKAKIS GN, TOGELIUS J. Cellular Automata for Real-time Generation of Infinite Cave Levels, Proceedings of the 2010 Workshop on Procedural Content Generation in Games (PC Games, 10), 2010: 1-4.

KENT JR, CARLSON WE, PARENT RE. Shape transformation for polyhedral objects. Computer Graphics (SIGGRAPH '92) 1992;26:47-54.

LAZARUS F, VERROUS A. Three-dimensional metamorphosis: a survey. The Visual Computer 1998;14(8-9):373-89.

LEE AWF, DOBKIN D, SWELDENS W, SHROEDER P. Multiresolution mesh morphing. Computer Graphics (SIGGRAPH '99) 1999;26:43-6.

MARTYN T. A new approach to morphing 2D affine IFS fractals. Computers & Graphics 2004; 28:249-72.

NISHITA T, NAKAMAE E, DOBASHI Y. Display of clouds taking into account multiple anisotropic scattering and sky light. SIGGRAPH 96 Conference Proceedings, Rushmeier H., (Ed.), ACM SIGGRAPH, Addison Wesley, 1996; 379-386.

PARISH YIH, MULLER P. Procedural modeling of cities. Proceedings (SIGGRAPH'01), ACM Press, E. Fiume, 2001; 301–308.

PEYTAVIE A, GALIN E, GROSJEAN J, MERRILOU S. Arches: a Framework for Modelling Complex Terrains, Computer Graphics Forum, Proceedings EUROGRAPHICS, 2009; 28(2):457-467.

PRUSINKIEWICZ P, LINDENMAYER A. The Algorithmic Beauty of Plants. Springer-Verlag, 1991: 101–107. ISBN 978-0387972978.

SCHUCHARDT P, BOWMAN DA. The Benefits of Immersion for Spatial Understanding of Complex Underground Cave Systems, Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology (VRST '07), 2007: 121-124.

SCHPOK J, SIMONS J, EBERT DS, HANSEN C. A real-time cloud modeling, rendering, and animation system. Symposium on Computer Animation'03, 2003; 160-166.

STINY G, GIPS J. Shape grammars and the generative specification of painting and sculpture. Information Processing 71, North-Holland Publishing Company, 1972: 1460-1465.

STINY G. Pictorial and Formal Aspects of Shape and Shape Grammars. Birkhauser Verlag, Basel, 1975.

STINY G. Introduction to shape and shape grammars. In Environment Planning B. 1980; 7(3): 343–361.

TURK G, O'BRIEN JF. Shape Transformation using variational implicit functions. Computer Graphics (SIGGRAPH '99) 1999;33:335-42.

WARSZAWSKI, K., NIKIEL, S. A proposition of particle system-based technique for automated terrain surface modeling. In Proceedings of the 5th International North American Conference on Intelligent Games and Simulation (Game-On-NA '09), 2009; 17-19, ISBN 978-9077381-49-6.

VELHO L, GOMES J, FIGUEIREDO LH. Implicit Objects in Computer Graphics. Springer, 2002, ISBN: 978-0387984247.

WOLBERG G. Image morphing: a survey. The Visual Computer 1998;14(8-9):360-72.

WONKA P, WIMMER M, SILLION F, RIBARSKY W. Instant architecture. ACM Transactions on Graphics 2003; 22(3):669–77.