Procedural Generation of 3D Cave Models with Stalactites and Stalagmites

Juncheng Cui[†], Yang-Wai Chow[†] and Minjie Zhang[††],

School of Computer Science and Software Engineering, University of Wollongong, Australia
Advanced Multimedia Research Lab[†]
Intelligent Systems Research Lab[††]

Summary
The increasing popularity of computer graphics applications in video games and movie production has resulted in a growing demand for the development of virtual environments with rich visual scene content. As such, the use of procedural content generation techniques is an attractive solution that can avoid the manual effort involved in the creation of highly complex scenes, by automating the generation of scene content. However, while there is much research on procedural content generation techniques, the procedural generation of 3D cave models is relatively unexplored. The focus of our research is on procedural cave generation, and this paper presents a method of producing a smooth triangle mesh of procedurally generated interior cave walls with visually plausible stalactites and stalagmites.

Key words:
Procedural Content Generation, Caves, Stalactite, Stalagmite.

1. Introduction

The creation of visual content for virtual environments can be a laborious and time consuming task. Nevertheless, the increasing popularity of computer graphics applications in areas such as movie production, video games, training simulations, and the like, has resulted in the growing expectation for virtual environments with rich scene content. This demand for visually realistic scenes increases the level of complexity and the amount of detail required in the creation of scene content, like 3D models, textures, weather effects, etc. It is an extremely arduous task for virtual environment developers to manually create such complex content. As such, procedural content generation is a research area that has received much attention in recent years.

Procedural content generation, or procedural modelling, is the automated generating of scene content using a computer. While reductions in the amount of work and costs associated with manual content creation is one of the main advantages of generating content procedurally, the development of procedural content generation techniques offers a variety of other benefits including automatically adding a degree of randomness to the generated content. Therefore, procedural techniques are great for creating naturally occurring phenomena, like fire and smoke, and many researchers have proposed a variety of different methods for such purposes [1, 2, 3, 4].

Caves are natural environments that are common in movies and video games. Furthermore, the creation of virtual 3D cave models is useful for the development of training simulations for cave explorers [5]. Yet research examining procedural techniques for the generation of underground and cavernous environments has remained largely unexplored [6, 7]. Our research examines the use of procedural techniques in the generation of synthetic 3D cave models. The focus of this research is not to develop a physically-based modelling technique, but rather to generate visually plausible 3D cave models.

In our previous work, an approach to procedurally generating 3D cave structures and storing this in voxel-based octrees, then obtaining the cave wall surfaces from the voxels was developed [8]. This paper presents a method of smoothing the triangle mesh representation of the interior cave walls in a crack-free manner, as well as an approach to procedurally add stalactites and stalagmites to the 3D cave model.

The remaining contents of the paper are organised as follows. An overview of related work is provided in section 2, which includes previous efforts on procedural content generation, as well as the creation of 3D cave models, stalactites and stalagmites. In section 3, a brief overview of our previous work is described. Section 4 details the approach developed in this research. This is followed by section 5 which presents and discusses experimental results. Finally, section 6 concludes the paper and describes our future work.

Manuscript received August 5, 2011
Manuscript revised August 20, 2011

2. Related Work

2.1 Procedural Content Generation

There are a variety of diverse procedural content generation techniques that have been developed by the research community for many different areas. Researchers have devised techniques for using algorithms to automate the generation of urban environment features, for example, the creation of buildings [9], roads [10], and even entire cities [11]. There is also a large body of work on procedural techniques for creating scene content for natural environments, such as plants and trees [12], skies [13], terrains and landscapes [14, 15]. In addition, the procedural simulation of other naturally occurring phenomena that have additional requirements, like animation properties, have also been studied, these include fire [1, 2], smoke [3, 4], and ocean waves [16].

2.2 Modelling 3D Caves

The development of procedural techniques for terrain generation is an especially successful area of procedural content generation, which is particularly relevant to cave generation. Over the years, much progress has been made toward developing efficient methods for generating synthetic terrains, using approaches like fractal modelling and physical erosion simulations [14]. However, traditional methods for representing terrains typically rely on 2D height maps for storing height field information. While height maps are simple and easy to use, this approach creates the limitation in that there can only be a single height value for each position on the horizontal plane. Thus, this restriction precludes the representation of overhands, arches or even caves [15].

To overcome this limitation, voxel grids have been used to represent concave terrain features [17]. In spite of this, the use of voxels is generally demanding in terms of data storage and computation. As such, methods to encode surfaces in sparse voxel octrees have been shown to significantly reduce memory and computational requirements [18]. Peytavie, et al. [15] developed a compact volumetric discrete data-structure for representing complex terrains, including caves. However, while they proposed a technique for the procedural generation of rock piles, their work mainly focused developing a framework for easing the burden of complex terrain creation through the use of high level terrain modelling and sculpting tools.

Boggus and Crawfis [5, 6] previously investigated the procedural generation of synthetic 3D cave models of solution caves. These are caves formed by rock being dissolved by acidic water. Their proposed method was a physically-based approach that involved approximating water transport to create coarse level of detail models for cave passages. Nonetheless, the generated cave models were limited to two planar surfaces; namely, the cave's floor and ceiling. In their latter work, they proposed a framework for modelling and editing 3D caves with any geometry [19]. However, while they described how speleothems could procedurally be added to their specialised cave data structure, this did not form part of their reported work.

2.3 Speleothems

Speleothems refer to mineral depositions formed in caves [7]. Common examples of these cave formations in limestone caves include stalactites, stalagmites and columns. The latter is typically formed when stalactites and stalagmites meet. The growth of stalactites and stalagmites is due to water dissolving some of the limestone. Studies have proposed models to approximate the physical formation of stalactites [20], and similar methods have also been developed to simulate 3D icicle formations [21].

Tortelli and Walter [7] presented an approach for modelling speleothem growth based on geological studies. In their work, they take advantage of the powerful computational capabilities of GPUs to model the genesis and growth of stalactites, stalagmites and columns, in real-time from a set of meaningful geological parameters. Their method, however, is not suitable for this research because the purpose of this research is not to grow speleothems, but rather to add visually plausible stalactites and stalagmites to a 3D cave model.

3. Previous Work

In our previous work, we developed an approach to procedurally generate 3D cave structures from 3D noise functions and storing the resulting interior cave walls in a voxel-based octree [8]. The octree data structure was employed to reduce memory storage requirements. In addition, previous work described a method of determining the polygonal surfaces of the cave walls from the voxel octree. An example of the resulting interior cave walls is shown in Fig. 1, where Fig. 1(a) portrays a 2D cross-section of a cave generated using a noise function, and Fig. 1(b) depicts a view of 3D voxels representing the cave walls.

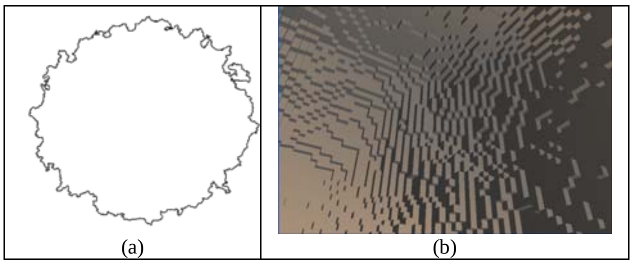


Fig. 1 Interior cave walls. (a) 2D cross-section; (b) 3D voxels.

It can be seen that the resulting cave model does not contain any notable cave features. Furthermore, rendering the polygonal cave wall surfaces obtained directly from the low resolution voxels produces unrealistic cube like walls, as can be seen in Fig. 1(b). Therefore, the research presented in this paper builds on top of existing work by demonstrating a method of smoothing the resulting polygonal mesh, as well as adding visually plausible stalactites and stalagmites to the procedurally generated cave model.

4. Our Approach

4.1 Polygonal Mesh Smoothing

Our aim is to convert the interior cave wall surface polygons obtained from data stored in a voxel-based octree, into a smoother looking polygonal mesh. This resulting mesh can then be rendered in real-time using standard triangle rasterisation techniques. While there are existing techniques to extract iso-surfaces from voxel data, for example, the popular marching cubes technique [22], we did not want to generate completely smooth surfaces as this would be unrealistic for caves. In other words, we want to smoothen the edges of the voxels so that the cave would not look like cube blocks, yet we did not want completely smooth iso-surfaces.

To achieve this, a Laplacian smoothing function [23] was employed to smooth the mesh:

$$v' = \lambda v + \frac{(1-\lambda)}{n}\sum_{i=1}^{n} v_i, (0 \le \lambda \le 1) \qquad (1)$$

Where v represents the original vertex location, and v_i stands for all the neighbouring vertices surrounding v. Thus, v' represents the adjusted vertex position. λ is the weighting factor and is typically set to 0.5. The implementation of this function effectively changes the original location of the triangle vertices and moves them closer toward their adjacent vertices, hence removing sharp corners and creating a smoother polygonal mesh.

However, simply using equation 1 gives rise to the appearance of cracks or `holes' in the cave model walls. This is because the original polygons were obtained from the voxel-based octree, hence their sizes are non-uniform as the octree voxel nodes differ in sizes. Therefore, shifting certain vertices away from their original locations will result in gaps due to certain polygons not aligning. Fig. 2(a) is a screenshot that shows the appearance of cracks in the cave walls (i.e. the black areas). Fig. 2(b) illustrates the reason why the cracks occur; the new vertex E' does not align with the A'C' edge, hence producing a gap between the edge and the vertex. This is a problem also seen in terrain Level of Detail (LOD) management, when trying to simplify polygonal meshes [24]. However, unlike triangle simplification algorithms used in terrain LOD management that deal with information stored in 2D height maps, in our case a 3D octree solution is required.

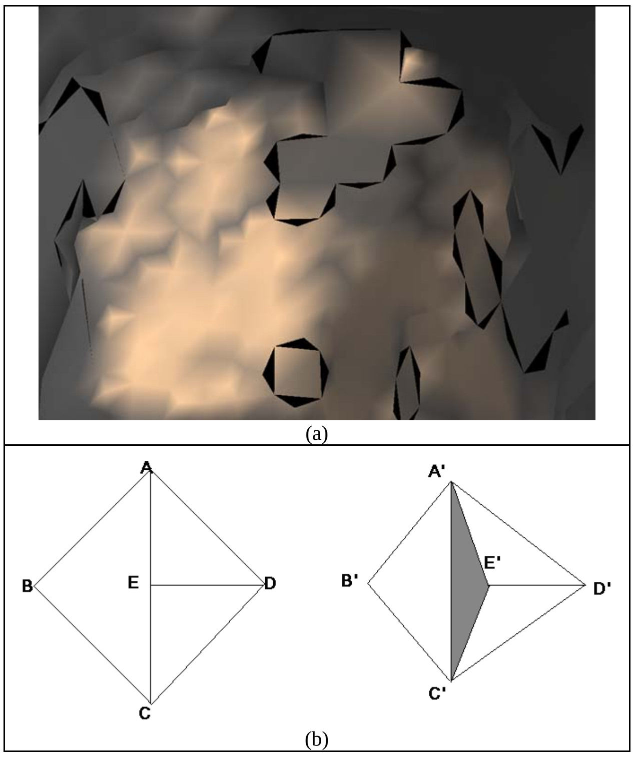


Fig. 2 'Holes' that appear during naïve polygon mesh smoothing. (a) Cracks that appear in the rendered 3D mesh; (b) Illustration of the reason for the cracks.

Our solution is to split the large triangle into smaller corresponding triangles that perfectly align. While this will increase the number of triangles in the final mesh, it will smoothly patch the cracks in the cave walls. We also need to maintain the original winding order of the vertices (i.e. clockwise or counter-clockwise order), as this has implications on the rendering of the triangles, e.g. back-face culling. Fig. 3 illustrates examples of how triangles would be split to produce a crack-free mesh from the smoothening processes, note that this actually takes place in 3D. In Fig. 3(a), changing the original locations of vertices A to H will result in the appearance of cracks. To overcome this, triangles 1, 2, and 3 will have to be split. This will result in the smaller triangles as shown in Fig. 3(b). Triangles were checked and split in a particular order. Fig. 4 illustrates the possible scenarios that result from the checking and splitting of triangles in the specific order of edge BC, followed by edge AB, then finally edge AC. In the figure, scenario 1 represents the case where no splitting is necessary; on the other hand, scenarios 2, 3 and 4 occur when only splitting only has to be performed along one of the edges; whereas the remaining scenarios show the cases where the triangle has to be split along multiple edges.

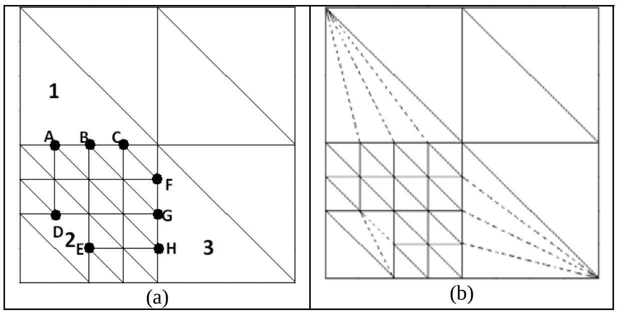


Fig. 3 Triangles to be split to avoid cracks.

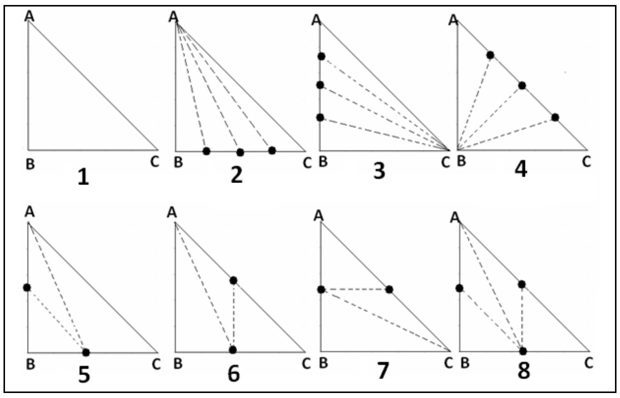


Fig. 4 Triangle splitting order.

4.2 Generating Stalactites and Stalagmites

Short et al. [20] proposed the following physically-based equation to approximate the shape of stalactites based on the relationship between factors like the radius and length:

$$\zeta(\rho) \cong \frac{3}{4}\rho^{\frac{3}{4}} - \rho^{\frac{2}{3}} - \frac{1}{3}\ln\rho + const. \qquad (2)$$

In equation 2, ζ is proportionate to the asymptotic shape of the stalactite. On the other hand, ρ is proportionate to the ratio of the radius and the stalactite's length. Therefore, we can generate stalactites with different shapes by assigning a different initial radius to each stalactite. Also, a scaling factor was applied to the results of ζ to make the stalactite appear more slender or stockier.

Stalagmites grow in conjunction with stalactites, as water from the cave ceiling flows to the tip of a stalactite before dripping to the floor of the cave, thus forming a corresponding stalagmite over time. A parameter was used to control the growth ratio between the stalactites and stalagmites. A value of 1.0 for this parameter means that the stalactites and stalagmites grow at equal rates. Stalagmites, however, typically grow at a faster rate as compared to stalactites. Therefore, the parameter was usually set to 1.5.

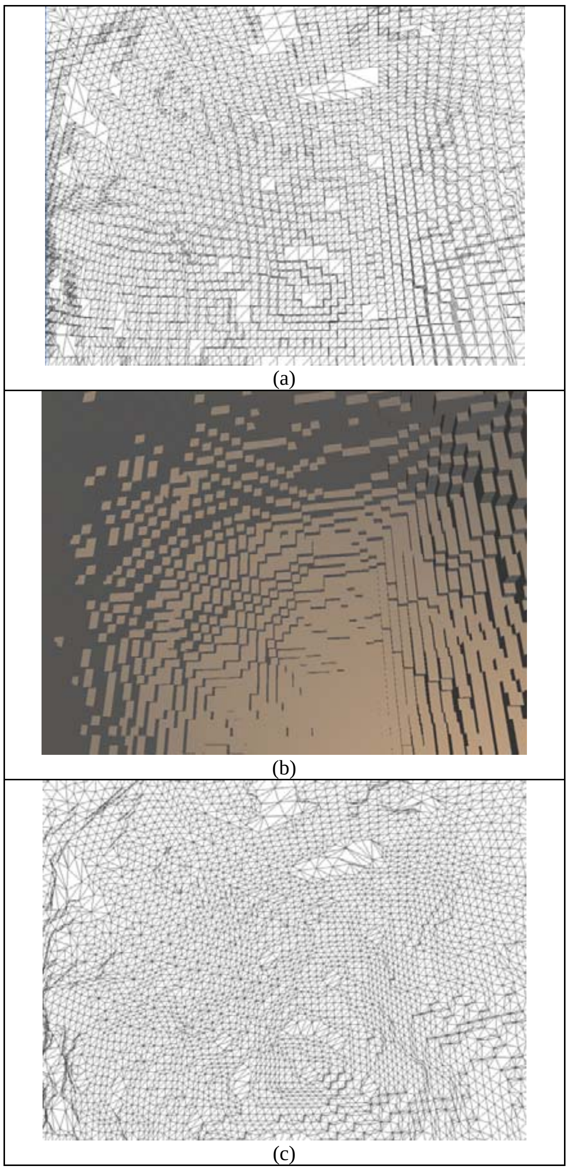
The locations of the stalactites and stalagmites can be randomly placed on the ceiling and floor of the cave structure. Additionally, the resulting shape of stalactites and stalagmites that are created using different parameters can be combined together. This gives rise to variations in the appearance of the stalactites and stalagmites.

5. Experiments

The techniques described in this paper were implemented in an application program. The program generates voxel-based 3D cave structures with stalactites and stalagmites procedurally, then produces a smooth polygonal mesh of the resulting cave model. The implementation was written in C# and the XNA framework was used to render the resulting polygonal mesh of the 3D cave in real-time. The 3D cave model was rendered with directional lighting without any textures. Experiments were conducted to examine the effectiveness of the crack-free smoothening algorithm, as well as the appearance of the procedurally generated stalactites and stalagmites. The experimental results are presented below, along with a discussion about the results.

5.1 Results and Discussion

Fig. 5 shows a comparison of the interior cave walls before and after the smoothening process. Screenshots showing the wireframe and solid polygonal mesh representations before smoothening are provided in Fig. 5(a) and Fig. 5(b), respectively. Fig. 5(c) and Fig. 5(d) in turn, show screenshots of the wireframe and solid polygonal mesh representations after the smoothening process. It can be seen that there are no cracks in the resulting smoothened cave walls. This shows the effectiveness of our triangle splitting approach in eliminating potential cracks due to vertex perturbations.



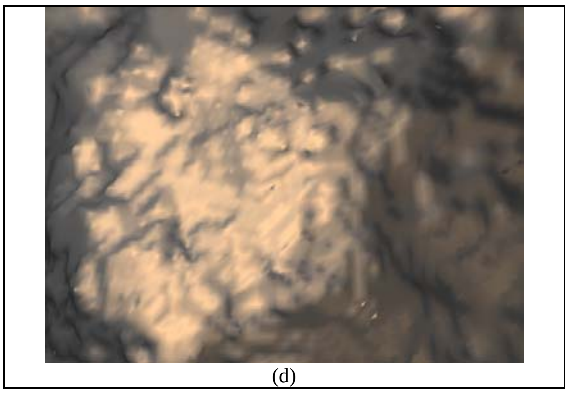


Fig. 5 Before and after smoothing. (a) Wireframe before the smoothing process; (b) Solid mode before smoothing; (c) Wireframe after the smoothing process; (d) Solid mode after smoothing.

Table 1, shows typical results of the triangle splitting approach used to eliminate cracks. The table depicts the total number of triangles before and after the smoothening process, for increasing levels of voxel resolution used in the generation of the 3D cave structure. Results show that in general, the number of additional triangles generated to produce a crack-free mesh from the smoothening process is less than 10%. This increase is a small price to pay when it comes to generating a smooth crack-free mesh.

Table 1: Total number of triangle before and after the smoothing process.

Max Voxel Octree Depth	***Number of Triangles***	
	Before Smoothing	***After Smoothing***
5	1296	1344
6	5196	5388
7	21376	22328
8	90760	95888
9	388084	418864

Fig. 6 shows a 2D cross-sectional depiction of the shape of stalactites and stalagmites that can be added to the ceiling and floor of a cave. Fig. 6(a), (b) and (c) depict stalactites and stalagmites of diverse shapes formed when different radii, *r*, and scaling factors, *s*, were used. The figures show that smaller scaling factors facilitates stockier shapes. The shape of stalactites and stalagmites resulting from a combination of those in Fig. 6(a), (b) and (c), is shown in Fig. 6(d). The combination of the stalactites and stalagmites shapes that were generated using different parameters, gives rise to variation and the appearance of randomness. This is realistic because variation and randomness is part of naturally occurring speleothems.

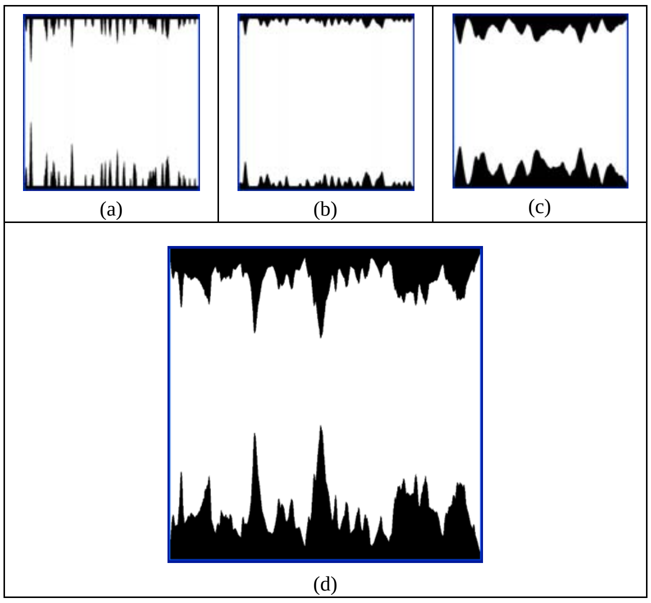


Fig. 6 2D cross-section of stalactites and stalagmites. (a) r = 5-10 *units*, s = 10; (b) r = 10-20 *units*, s = 1; (c) r = 20-50 *units*, s = 0.5; (d) Combining a, b and c.

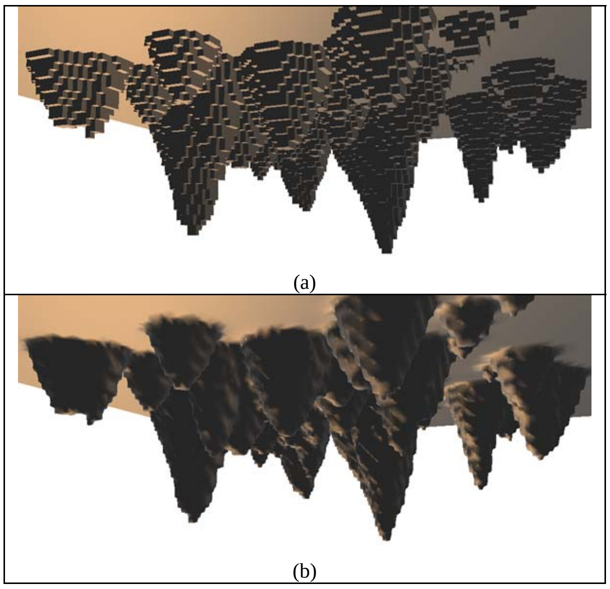


Fig. 7 Simulation of 3D stalactites. (a) Before smoothing; (b) After smoothing.

Fig. 7(a) shows a 3D voxel representation of stalactites, whereas Fig. 7(b) shows the same stalactites after the polygonal smoothening process. A voxel representation of a procedurally generated 3D cave passage with stalactites and stalagmites is shown in Fig. 8(a). The same cave passage is shown after the polygonal smoothing process in wireframe mode in Fig. 8(b) and in solid mode in Fig. 8(c). It can be seen that our approach is effective in producing visually plausible 3D cave models with stalactites and stalagmites.

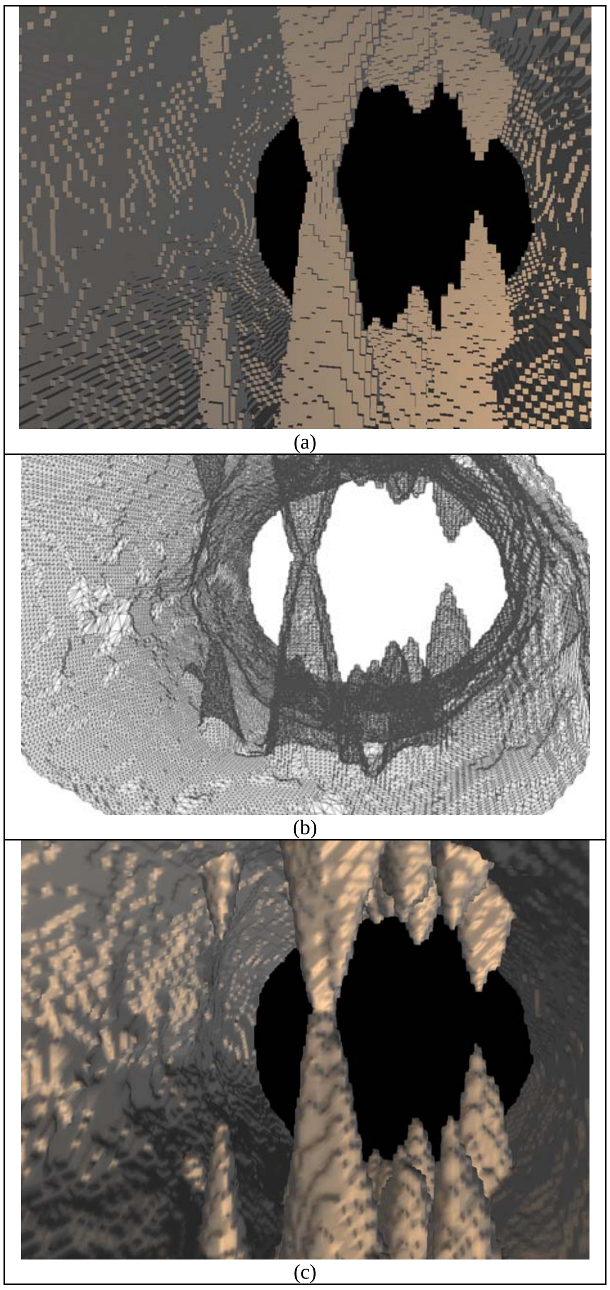


Fig. 8 3D cave model with stalactites and stalagmites. (a) Voxel representation of the cave before smoothing; (b) Wireframe after smoothing; (c) Solid mode after smoothing.

6. Conclusion and Future Work

This paper is concerned with the procedural generation of 3D cave models. Our main contributions include a method of producing a smooth triangle mesh of procedurally generated interior cave walls. In addition, an approach to splitting triangles to overcome the problem of cracks in the smoothened mesh was described.

This paper also presents a method of procedurally generating visually plausible stalactites and stalagmites and adding these to the cave structure. We have implemented these methods experimentally and presented our results to demonstrate the effectiveness of our approach. Future work will investigate techniques of generating other cave features like cave columns, flowstones, cave scallops, etc. Furthermore, the procedural generation and incorporation of realistic textures for caves will also be examined.

References

[1] Nguyen, D.Q., Fedkiw, R. and Jensen, H.W., "Physically based Modeling and Animation of Fire," ACM Transactions on Graphics (TOG), Vol. 25, No. 3, pp. 614-623, 2006.
[2] Fuller, A.R., Krishnan, H., Mahrous, K., Hamann, B. And Joy, K.I., "Real-time Procedural Volumetric Fire," in Proceedings of the 2007 Symposium on Interactive 3D Graphics (SI3D '07), pp. 175-180, 2007.
[3] Rasmussen, N., Nguyen, D.Q., Geiger, W. and Fedkiw, R., "Smoke Simulation for Large Scale Phenomena," ACM Transactions on Graphics (TOG), Vol. 22, No. 3, pp. 703-707, 2003.
[4] Shi, L. and Yu, Y., "Controllable Smoke Animation with Guiding Objects," ACM Transactions on Graphics (TOG), Vol. 24, No. 1, pp. 140-164, 2005.
[5] Boggus, M. and Crawfis, R., "Procedural Creation of 3D Solution Cave Models," in Proceedings of the 20th IASTED International Conference on Modelling and Simulation, pp. 180-186, 2009.
[6] Boggus, M. and Crawfis, R., "Explicit Generation of 3D Models of Solution Caves for Virtual Environments," in Proceedings of the 2009 International Conference on Computer Graphics and Virtual Reality, pp. 85-90, 2009.
[7] Tortelli, D.M. and Walter, M., "Modeling and Rendering the Growth of Speleothems in Real-time," in Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications (GRAPP '09), pp. 27-35, 2009.
[8] Cui, J., Chow, Y.W. and Zhang, M., "A Voxel-based Octree Construction approach for Procedural Cave Generation," International Journal of Computer Science and Network Security, Vol. 11, No. 6, pp.160-168, 2011.
[9] Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Van Gool, L., "Procedural Modeling of Buildings," ACM Transactions on Graphics (TOG), Vol. 25, No. 3, pp. 614-623, 2006.
[10] Galin, E., Peytavie, A., Marechal, N. and Guerin, E., "Procedural Generation of Roads," in Computer Graphics Forum, Vol. 29, No. 2, pp. 429-438, 2010.
[11] Parish, Y.I.H. and Müller, P., "Procedural Modeling of Cities," in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01), pp. 301-308, 2001.
[12] Lluch, J., Camahort, E. and Vivo, R., "Procedural Multiresolution for Plant and Tree Rendering," in Proceedings of the 2nd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH '03), pp. 31-38, 2003.
[13] Roden, T. and Parberry, I., "Clouds and Stars: Efficient Real-time Procedural Sky Rendering using 3D Hardware," Advances in Computer Entertainment Technology, pp. 434-437, 2005.
[14] Zhou, H., Sun, J., Turk, G. and Rehg, J.M., "Terrain Synthesis from Digital Elevation Models," IEEE Transactions on Visualization and Computer Graphics, Vol. 13, No. 4, pp. 834-848, 2007.
[15] Peytavie, A., Galin, E., Grosjean, J. and Merillou, S., "Arches: a Framework for Modeling Complex Terrains," Computer Graphics Forum, Proceedings of EUROGRAPHICS, Vol. 28, No. 2, pp. 457-467, 2009.
[16] Jeschke, S., Birkholz, H. and Schumann, H., "A Procedural Model for Interaction Animation of Breaking Ocean Waves," in Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '03), 2003.
[17] Jones, M.D., Farley, M., Butler, J. and Beardall, M., "Directable Weathering of Concave Rock using Curvature Estimation," IEEE Transactions on Visualization and Computer Graphics, Vol. 16, No. 1, pp. 81-94, 2010.
[18] Laine, S. and Karras, T., "Efficient Sparse Voxel Octrees," in Proceedings of the 2010 Symposium on Interactive 3D Graphics (SI3D '10), pp. 55-63, 2010.
[19] Boggus, M. and Crawfis, R., "Prismfields: A Framework for Interactive Modeling of Three Dimensional Caves," in Proceedings of the 6th International Conference on Advances in Visual Computing (ISVC '10), Volume Part II, pp. 213-221, 2010.
[20] Short, M.B., Baygents, J.C., Beck, J.W., Stone, D.A., Toomey III, R.S. and Goldstein, R.E., "Stalactite Growth as a Free-boundary Problem: A Geometric Law and its Platonic Ideal," Physical Review Letters, Vol. 94, No 1, Article 018501, 2005
[21] Kim, T., Adalsteinsson, D. and Lin, M.C., "Modeling Ice Dynamics as a Thin-film Stefan Problem," in Symposium on Computer Animation (SCA '06), pp. 167-176, 2006.
[22] Lorensen, W.E. and Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in Proceedings of the 14th Annual Conference on Comupter Graphics and Interactive Techniques (SIGGRAPH '87), pp. 163-169, 1987.
[23] Vollmer, J., Mencl, R. and Müller, H., "Improved Laplacian Smoothing of Noisy Surface Meshes," in Computer Graphics Forum, Vol. 18, No. 3, pp. 131-138, 1999.
[24] Wu, J., Yang, Y.F., Gong, S.R. and Cui, Z.M., "A New Quadtree-based Terrain LOD Algorithm," Journal of Software, Vol. 5, No. 7, 2010.

Juncheng Cui received the B.S. degrees in Software Engineering from Tongji University, China, in 2007. He is currently working toward to the M.S. degree under the supervision of Dr. Yang-Wai Chow and A/Prof. Minjie Zhang. His research interests include computer graphics, procedural content generation and artificial intelligence.

Yang-Wai Chow received his BSc., B.Eng. (Hons.) and Ph.D. from Monash University, Australia, in 2003 and 2007. He is currently a Lecturer in the School of Computer Science and Software Engineering, at the University of Wollongong, Australia. His research interests include computer graphics, virtual reality, interactive real-time interfaces, human visual perception and human computer interaction.

Minjie Zhang is an Associate Professor in the School of Computer Science and Software Engineering and the Director of Intelligent System Research Group in the Faculty of Informatics, at University of Wollongong, Australia. She received her BSc. degree from Fudan University, China in 1982 and the PhD degree in Computer Science from the University of New England, Australia in 1996. Her research interests include distributed artificial intelligence, multi-agent systems, agent-based simulation and modeling in complex domains, grid computing, and knowledge discovery and data mining.