A Sierpinski Space-filling Curve based Modeling of Height Fields

Yusnier Valle Martínez.
Universidad de las Ciencias Informáticas, Cuba.
yvm@uci.cu.

***Abstract**. A continuous curve in 2, 3, or higher dimensions, can be thought as a path of a continuously moving point. This class of curves has been used in the design and implementation of spatial data structures. This paper is essentially focused on the efficient representation of terrain models. We propose to implement a binary triangles tree by means of the Sierpinski space-filling curve. The hierarchical nature of this curve makes it very useful to represent this kind of spatial structure. Traversal algorithms, neighbor finding techniques, among others, are presented on the paper demonstrating the efficiency of the proposed structure.*

***Keywords**. Sierpinski; bitrees; triangulation; multiresolution; visualization; space-filling curve.*

Introduction

Digital elevation models (DEMs) are an important component in a wide range of application domains, such as scientific visualization, Geographic Information Systems, mapping applications, interactive 3D games, urban planning, or flight simulators. In general, the terrain model is not the most important component in a simulation, hence the importance of ensuring its representation at high frame rates. On the other hand, due to the increasing size and complexity of DEMs, real-time display imposes significant efficiency constraints on the visualization engine, which is forced to dynamically trade rendering quality with usage of limited system resources.

The problem of mesh simplification and multiresolution surface triangulation has been widely studied over the last two decades. This paper addresses the problem of real time visualization of terrain datasets that entirely fit into the main memory. The proposed technique, called SBT (Sierpinski-Bitree Triangulation), is focused on the efficient representation of the spatial model by means of a binary triangles tree (Evans et al., 2001). The spatial data structure is implemented as a hierarchical Sierpinski space-filling curve, which is the base of the algorithm proposed for fast triangle strip generation in order to speed out the visualization process.

Previous work

A Digital Terrain Model (DTM) is a digital representation of the properties of the topography of a surface. Digital Elevation Models are among the best known, which are usually stored as a regular grids, triangulated irregular networks or contour lines.

A large number of researchers have developed algorithms for fast and efficient terrain rendering by means of polygonal meshes. A number of approaches are based on the principle of Delaunay triangulation (Van Kreveld, 1997) to create Triangulated Irregular Networks (TINs) over irregularly spaced sets of points. Another important contribution belongs to (Hoppe, 1996) in which the progressive mesh representation is introduced, a new scheme for storing and transmitting arbitrary triangle meshes.

The use of spatial data structures in order to create digital terrain models from regular meshes has proven highly effective, largely due to the high efficiency and flexibility of these kinds of structures. In (Pajarola et al., 2007) the authors analyze multiresolution approaches that takes into account a certain semi-regularity of the data, such as models based on restricted quadtrees and binary triangles trees (Pajarola, 1998, Lindstrom et al., 1996, Lindstrom et al., 2002, Röttger et al., 1998).

Among the most used techniques for terrain rendering in a wide variety of applications are those that represent surfaces by means of Hierarchies of Right Triangles (HRT), where each resulting triangle is right-angled and isosceles. Similar to (Duchaineau et al., 1997, De Floriani et al., 2002), the main contribution of (Evans et al., 2001) with their Right-triangulated Irregular Network (RTIN) is the proposal of a very efficient data structure for representing terrain models in main memory. In the RTIN, each triangle is recursively labeled appending 0 or 1 to the codification of its ancestor, depending on its position as left or right child. The representation of the model takes the form of a binary triangles tree, for which a very efficient neighbor finding technique is presented in the paper.

Spatial data structure

The process of building a binary triangles tree is performed from an arbitrary set of points in the plane, which form a regular grid represented by a two-dimensional array M of size (2”+1) x (2”+1), $n \geq 2$. The process starts by dividing the main quadrant of the mesh with a diagonal, obtaining two triangles as a result. The division continues recursively by adding the midpoint (in the mesh) of the hypotenuse of each triangle until the desired resolution is reached.

Instead of representing the structure as a RTIN, the SBT is represented as a one-dimensional array that, at each position, holds a two-dimensional array of triangles, Figure 1.]

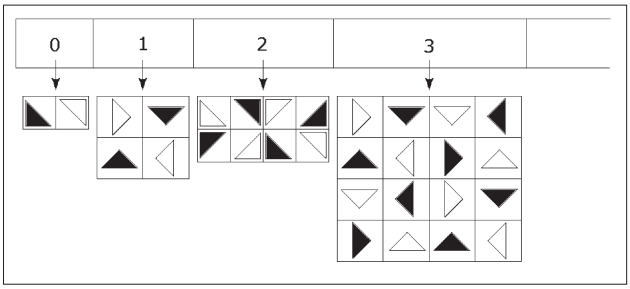

Figure 1. First 4 levels in a SBT.

Each index in the one-dimensional array corresponds to a level in the structure, and each two-dimensional array holds the information corresponding to the nodes in the corresponding level. The first 4 levels are fixed patterns of triangles, and from level 1, 1 > 3, the information from 1 – 2 is replicated 4 times in the level 1. Given a two-dimensional array M from size (2"+1) x (2"+1), n ≥ 2, Algorithm 1 illustrates the way the process is performed.

Algorithm 1: Creation of the binary triangles tree.

```
1: procedure SIERPINSKI_TREE()
2: levels – 2* n + 1
3: stree – new array[levels][,]
   .....
   //create patterns from levels 0 to 3…
   .....
3: for i – 4 to levels-1 do
4: REPLICATE_PATTERN (i-2,i)
5: endfor
6: endprocedure
```

In lines 2 and 3 the structure is initialized and subsequently the fixed patterns are constructed for levels 0 through 3. The loop from line 3 completes the process of creation by multiple callings to the Algorithm 2, which receives as parameters the source level to replicate to a given destiny level.

Algoritmo 2: Replica of the information stored in a source level to a destiny level.

```
1: procedure REPLICATE_PATTERN (src: integer, dest: integer)
2: stree[dest] – new array[ROWS(stree(src))*2, COLUMNS(stree(src))*2]
3: srcRows – ROWS(stree(src))
4: srcCols – COLUMNS(stree(src))
5: for i=0 to srcRows-1 do
6: for j=0 to srcCols-1 do
7: stree[dest][i,j] = stree[src][i,j]
8: stree[dest][i+srcRows,j] = stree[src][i,j]
9: stree[dest][i,j+srcColumns] = stree[src][i,j]
10: stree[dest][i+srcRows,j+srcColumns] = stree[src][i,j]
    .....
    //actualize children's information…
    .....
11: endfor
12: endfor
    .....
    //actualize parent's information…
    .....
13: endprocedure
```

In line 2, the two-dimensional array is created for the destination level. The section from line 5 to 10 performs a replica of the information, and later the positions of its triangles children and parents are updated. Each triangle requires 4 bytes for the coordinates of its left child (black triangles, Figure 1), 4 bytes for the corresponding coordinates of its right child (white triangles, Figure 1), and 4 bytes to the position of its parent triangle. Additionally, 3 bits are needed to identify the type of each triangle (there are 8 different types according to its orientation).

The neighbor finding strategy proposed in (Evans et al., 2001) is based on the code associated with each triangle. The way we represent the SBT is the basis for the neighbors finding technique for a given triangle T, since each neighbor is in the same two-dimensional array that T in the structure. Just as in (Evans et al., 2001), if we number the vertices of T from 1 to 3 in counter-clockwise, the i-neighbor N of T is defined as the neighbor that does not share the vertex i of T. The relative position to T of each N can be calculated by adding or subtracting 1 to the T coordinates, for which 2 bits are stored in T for each N_i: one to indicate the coordinate of T that is affected by the operation, and the other to indicate the type of operation (addition or subtraction). Algorithm 3 returns the same-size i-neighbor of a triangle at a valid position [level][r,c].

Algoritmo 3: Find the same-size i-neighbor of a triangle at a valid position [level][r,c].

```
1: procedure I-NEIGHBOR (i, level, r, c: integer)
   .....
   //find op and n depending on i…
   .....
2: return stree[level][r+(opˆ1)*(-1)^n,c+(op &1)*(-1)^n]
3: endprocedure
```

Before line 2, the values of op y n (0 or 1) are calculated by using logical operations depending on the parameter i. The operations (opˆ1) and (op &1) determine the parameter (r or c) affected by $(-1)^n$.

Visualization

The speed at which triangulated surfaces can be displayed is decisive in almost all scientific visualization techniques (Arkin et al., 1994). Triangle strip generation based on space-filling curves has been widely used to generate efficient triangulations over hierarchical data structures (Lindstrom et al., 2002, Pajarola, 1998, Velho et al., 1999). The technique proposed in this section is based on the Sierpinski space-filling curve, where the type of each triangle corresponds to a segment of the curve, Figure 2 (left). The segments of the curve are connected to each other making a simple post-order traversal of the structure, Figure 2.

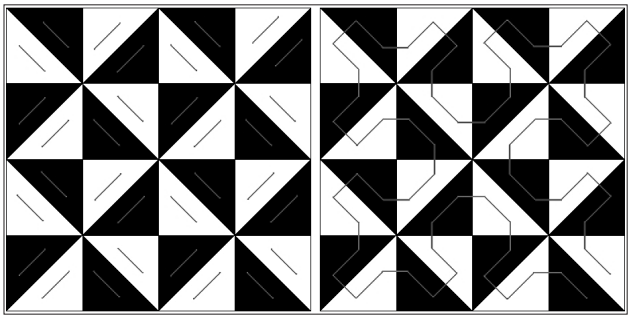

Figura 2. Sierpinski sections (left) and curve (right).

Algorithm 4 illustrates the way each triangle is drawn in a given level of the structure.

Algoritmo 4: Draws the Sierpinski space-filling curve corresponding to the level drawLevel.

```
1: procedure SIERPINSKI (level, r, c, drawLevel: integer)
2: if level < drawLevel
3: lc – LCHILD(stree[level][r,c])
4: rc – RCHILD(stree[level][r,c])
5: SIERPINSKI (level+1, lc.X , lc.Y, drawLevel)
6: SIERPINSKI (level+1, rc.X , rc.Y, drawLevel)
7: else
8: size – ROWS(M) div 2^(level/2)
   .....
   //draw the triangle at [level][r,c]…
   .....
9: endif
10: endprocedure
```

The lines from 2 to 6 performs a post-order traversal of the structure. Each time a triangle T corresponding to the level drawLevel is reached, the coordinates of T in the two-dimensional array M are calculated given its location, its size in the SBT, and the size of the cells in M. Each calculated vertex takes part of a generalized triangle strip that is sent to the graphics hardware in order to be visualized. Thus, the Algorithm 4 only uses the 66% of 3N vertices needed to represent N triangles.

Conclusions and future work

In this paper has been presented a new strategy for representing a binary triangles tree, in order to model and display terrain surfaces interactively and in real time. In order to show the effectiveness of the proposed model for fast an efficient terrain rendering, some screen shoots were taken from a simple application running in a Toshiba Satellite L20-273 with a 1.40 GHz Intel Celeron M360 processor and a 64 MB ATI RADEON XPRESS 200M Series (0x5A62) graphics card. For visualization purposes, we use sample data describing the Great Canyon and Puget Sound areas, USA, with the elevation data artificially scaled in order to exaggerate the elevation changes. Figure 3 shows a view from the Great Canyon data modeled by a SBT.

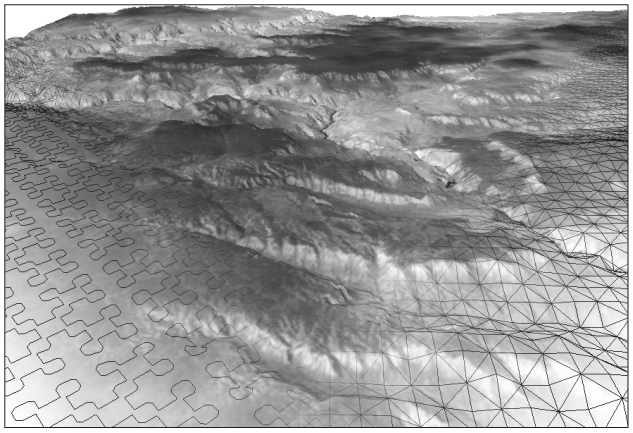

Figure 3. Landscape and the corresponding triangulation and curve.

Both the RTIN of (Evans et al., 2001) as the SBT addresses the problem of the efficient representation in memory of the surface data. The running time complexity of the main algorithms is the same in both structures, in addition to the size of each structure in main memory. (Evans et al., 2001) does not directly address the issue of the efficiency in information visualization, a key element in this type of applications. In a SBT each triangle T is considered as a segment of the Sierpinski space-filling curve, and the way each T is located in the structure allows the efficient generation of triangle strips to speed out the visualization process.

Even when the performance shown by the visualization technique is quite efficient, further research is needed in order to apply an effective scheme to extract on-the-fly multiresolution triangulations from de model. Beside this, due to GPUs are become increasingly powerful, we are working on a CPU/GPU communication model that is not processor intensive and takes advantage of current graphics hardware.

References

Arkin, E. M., Held M., Mitchel, J. S. B., Skiena S.: 1994, Hamiltonian Triangulations for Fast Rendering, in ESA ´94: Proceedings of the Second Annual European Symposium on Algorithms, Springer-Verlag, pp. 36-47.

De Floriani, L., Magillo, P.: 2002, Triangle-based Multi-Resolution Models for Height Fields, Curve and Surface Fitting: Saint-Malo, A. Cohen, J.-L. Merrien, L.L. Schumaker (eds.), Nashboro Press, Brentwood, TN, USA, 2003, pp. 97-106.

Duchaineau, M. A., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., Mineev-Weinstein, M. B.: 1997, ROAMing terrain: real-time optimally adapting meshes, IEEE Visualization, pp. 81-88.

Evans, W. S., Kirkpatrick, D. G., Townsend, G.: 2001, Right-triangulated irregular networks. Algorithmica 30, 2, pp. 264-286.

Hoppe, H.: 1996, Progressive meshes, in SIGGRAPH'96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM, pp. 99–108.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L., Faust, N., Turner, G.: 1996, Real-time continuous level of detail rendering of height fields, Proceedings of SIGGRAPH '96, pp. 109-118.

Lindstrom, P., Pascucci, V.: 2002, Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization, IEEE Transactions on Visualization and Computer Graphics, Volume 8, Issue 3, pp. 239-254.

Pajarola, R. B., Gobbetti, E.: 2007, Survey of semi-regular multiresolution models for interactive terrain rendering, The Visual Computer, 8, Springer-Verlag New York, Inc., pp. 583-605.

Pajarola, R. B.: 1998, Large scale terrain visualization using the restricted quadtree triangulation, IEEE Visualization '98, D. Ebert, H. Hagen, and H. Ily Rushmeier, Eds., pp. 19-26.

Röttger, S., Heidrich, W., Slussallek, P., Seidel, H-P.: 1998, Real-Time Generation of Continuous Levels of Detail for Height Fields, Proc. 6th Int. Conf. in Central Europe on Computer Graphics and Visualization, pp. 315—322.

Van Kreveld, M. J.: 1997, Algorithms for Triangulated Terrains, in Conference on Current Trends in Theory and Practice of Informatics, pp. 19-36.

Velho, L., de Figueiredo, L. H., Gomes, J.: 1999, Hierarchical Gereneralized Triangle Strips, The Visual Computer 15, 1, pp. 21-35.