Real-Time Terrain Surface Extraction at Variable Resolution

LUIZ CARLOS CASTRO GUEDES

ADDLabs - Laboratório de Documentação Ativa e Design Inteligente,
Departamento de Ciência da Computação, Universidade Federal Fluminense
Praça Valonguinho s/n, Ed. Instituto de Matemática 4o andar, Centro, 24210-130 Niterói, RJ, Brasil
guedes@dcc.uff.br

Abstract. Multiresolution terrain models allow the extraction of the surface at different levels of details. Real terrain data is large enough to prevent from real-time walk-through at a reasonable level of detail. Moreover, a coarse view of a terrain may not attend some application requirements. Therefore extracting the surface at variable resolution may offer impressive good-looking views if, for instance, regions closer to the observer are shown at a higher resolution then regions farther away. Besides, variable resolution extraction dramatically reduces the number of polygons to be rendered allowing real-time walk-through of a large terrain model. This paper proposes a multiresolution terrain model, based on Delaunay triangulation, that takes into account the whole domain during triangulation without replicating unchanged triangles through different levels of detail. The proposed model is the first that matches a whole domain Delaunay triangulation with accuracy of extraction and low storage cost algorithms.

Keywords: multiresolution model, terrain model, real-time extraction, variable resolution

1. Introduction

Interactive applications are becoming more common every day. Many high performance applications can be run on low-end personal computers. Particularly, flight simulators and landscape viewers are achieving an impressive level of realism and performance. However, their accuracy is not very high. Geographic Information Systems (GIS), on the other hand, deal with very large data bases and need much higher accuracy, what makes the problem of interactive three-dimensional visualization very hard.

When displaying a terrain model, it is easy to observe that regions far from the observer need not to be rendered with the same level of detail as the closer ones. Henceforth, many researchers [DeFPup95] [CiPuSc95] [deBDob95] [Bert+95] have proposed polygonal models that offer efficient storage of terrain data at different levels of detail and linear time extraction of the surface. Those models are called *multiresolution*.

A multiresolution terrain model may support extraction of the terrain surface satisfying a fixed resolution or based on a resolution function that specifies how the resolution should vary throughout the whole domain (variable resolution). Usually this resolution function is monotonically decreasing on the distance from the observer. This is the case of flight simulators and landscape viewers.

We may qualitatively evaluate a multiresolution terrain model based on many issues, such as supporting *explicit/implicit multiresolution*, being *hierarchical or pyramidal, being matching or non-matching* and working on *regular or irregular distributed data* [DeFlo+96a].

Implicit multiresolution terrain models ensure that the greatest error found at most refined approximation is less than a specified error value $\boldsymbol{\varepsilon}$. On the other hand, in explicit multiresolution models each refinement layer i accepts an error less than an error value $\boldsymbol{\varepsilon}_i$ from a sequence $\boldsymbol{\varepsilon} = [\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots \boldsymbol{\varepsilon}_n]$ of error values.

A multiresolution terrain model where the region covered by a polygon in a given refinement is entirely covered by only one polygon (called its parent) of the previous refinement layer is called an *hierarchical terrain model*. It means that the whole model may be represented by an hierarchical tree [DeFlo+96a]. In a hierarchical model, a polygon is refined independently of the other polygons of the same layer. This locality may affect the shape of the polygons creating very elongated triangles close to the borders that may bring numerical errors during rendering. This problem may be overcome [DeFlo+96a] but the result will never represent a global optimal solution like the one given by a single Delaunay triangulation of the whole domain [Rippa90].

Models where the refinement process takes into account the entire domain may not preserve the hierarchical relation among polygons belonging to distinct layers. In such models each layer covers the entire domain and it

requires much storage due to the replication of triangles that does not change from one layer to another. Those models are called *pyramidal terrain models* [DeFlo89]. On the other hand pyramidal models may achieve a global optimal approximation for each layer.

Terrain models where edges are preserved from one layer to another, that is, edges from one level of refinement are never refined further at the more refined levels, are said to be *matching*. The great advantage of matching terrain models is that continuity of the surface extracted at any of the given levels of resolution is guaranteed [DeFlo+96a]. Matching is an exclusive property of hierarchical terrain models that directly affects the extraction of a surface at variable resolution. Nonetheless, it is possible to extract a continuous surface at variable resolution from a pyramidal model solving discontinuities on the fly during the rendering process.

Even though real terrain data is usually sampled on a regular space grid, it is highly interesting to adapt the given data to the surface characteristics. Variable data distribution allows a better compression without loss of rendering quality. Triangle decimation techniques [GarHec95][SchRoβ94] should be employed to obtain from a regular distributed data the relevant information needed to extract a terrain surface at a given resolution.

This paper proposes a multiresolution terrain model, called *Pyramidal Delaunay Triangulation* (PDT), suitable for extracting continuous surfaces at variable resolution. *Pyramidal Delaunay Triangulation* takes into account the whole domain during triangulation without replicating unchanged triangles through different levels of details and ensuring the accuracy of the extracted surface. Although the model is non-matching, a simple solution is proposed to ensure the continuity of the extracted surfaces.

The rest of the paper is organized as follows. In section 2, related works are briefly reviewed. Section 3 presents the *Pyramidal Delaunay Triangulation*. In section 4 a data structure to encode PDT is described. In section 5 the algorithms to build a PDT and extract a surface at variable resolution are described and analyzed. And, finally, in section 6, some applications are described and some concluding remarks are given.

2. Related work

Several multiresolution models have been proposed on the literature, see [DeFlo+96a] for an interesting survey. Most of them only support extraction at a constant resolution. Here we will focus on the models that are suitable for variable resolution extraction.

In [DefPup95], the Hierarchical Delaunay Triangulation (HDT) is proposed. An Hierarchical Delaunay Triangulation is a matching tree structure that supports an explicit multiresolution representation of irregularly distributed terrain data. Each level, except the root, is built from the previous level by inserting points inside each triangle until the corresponding resolution of its level is achieved. Points are inserted following an algorithm called DELAUNAY_SELECTOR [DeFlo+96b], that performs a Delaunay triangulation only inside the corresponding triangle. The final triangulation is not optimal, since only local points are considered during the Delaunay test, and the result presents many elongated triangles near the borders. This slivering problem is solved by insertion of interpolated points on the edges performing a *conforming Delaunay triangulation* [DeFlo+96b], although the solution is still sub-optimal since adjacent triangles are refined independently. They present two algorithms to extract a continuous model at variable resolution from a hierarchical triangulated surface, although the computational complexity is suboptimal.

In [CiPuSc95], an alternative approach called *HyperTriangulation* (HT) is proposed. The idea is to store a sort of history of the incremental refinement process. Each vertex in the structure stores the number of the refinement step of when it was inserted (elevation) and each triangle stores the error of the approximation when it was created (birth error) and when it was refined (death error). Assuming that the terrain covers a region of the XY plane, at each refinement step *i* the new vertices inserted are raised along the Z axis at elevation *i*. The x and y coordinates of the vertices together with the elevation builds a virtual 3D structure of overlapping triangles where the upper surface corresponds to the most refined approximation of the terrain. Variable resolution extraction is achieved by traversing the structure and checking the birth and death errors of each triangle against a monotonically increasing error function that selects which triangles present the maximum error that satisfies the error function. This algorithm works with a breadth-first order traversal starting at the view point with an incremental construction of the surface. The computational complexity is suboptimal .

Opposing to the HDT and the HT, in [deBDob95] a bottom-up approach called *Implicit Delaunay Pyramid* is proposed. It starts with a Delaunay triangulation of the terrain at the maximum resolution and eliminates a percentage α of vertices from the approximation at each step. Because of this geometry based construction, the tolerance of each level is not explicitly controlled. Although variable resolution surfaces can be extracted at

linear time complexity, the accuracy of the model may not be guaranteed.

In [Puppo96] a theoretical framework is proposed in order to formalize hierarchical triangulations, and an abstract structure called *MultiTriangulation* (MT) is presented. MultiTtriangulations are axiomatically defined and do not exist by themselves. In his work Puppo shows some interesting properties of MTs and claims that if someone is able to identify how to match one model with the MultiTriangulation framework then the MT properties are automatically ensured. He illustrates how to identify the HyperTriangulation and the Hierarchical Delaunay Triangulation as MultiTriangulations.

In [Bert+95] a formalization of a broader class of multiresolution models in arbitrary dimensions that can incorporate both hierarchical and pyramidal models as special cases is proposed. The intention of this model is to compare hierarchical and pyramidal models in terms of storage cost and performance and to gain some highlight in understanding when each model should be more appropriate.

3. Pyramidal Delaunay Triangulation

A *Pyramidal Delaunay Triangulation* (PDT) is a surface model for terrain approximation that resembles a Delaunay Pyramid [DeFlo89] in the sense that it may be considered, at first, as a sequence of Delaunay triangulations that cover (each one) the entire domain of approximation. The basic difference from the Delaunay Pyramid is that triangles that do not change from one level to another are not replicated in the next levels.

A *Mathematical Terrain Model* (MTM) is a pair $M = <V, \Phi>$, where $V = \{ v_1, v_2, ..., v_n \}$ is a finite set of points in a domain D of the Euclidean plane R^2, and $\Phi: V \rightarrow R$ is a function that associates to each point in V its corresponding elevation in the original terrain. Φ is called an elevation function. A *Digital Terrain Model* (DTM) is triple $D = < V, E, \Phi >$, where V and Φ are defined as above and E is a set of edges connecting the vertices of V, such that <V, E> defines a planar connected graph. A DTM induces a set of faces F that covers the domain of approximation. Figure 1 illustrates a DTM.

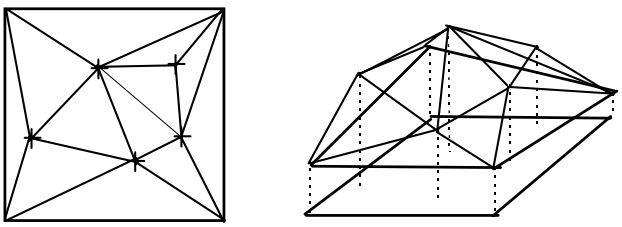

Figure 1 - A Digital Terrain Model

Given a DTM D, we call a coarse representation of D to any DTM $D' = < V', E', \Phi >$, where $V' \subset V$. When this happens we say that a $D' \subset D$. Given two DTMs D and D', such that $D' \subset D$, and a vertex $v \in V - V'$, we call the *approximation error* of v, $E(v) = |\Phi(v)-h|$, to the difference between the elevation of v and value h assigned to v by the facet induced by D' that covers v.

A PDT is partially a hierarchical structure, built from a DTM D, where each layer, except for the root, is built from the previous one by inserting into each facet the point p with the maximum approximation error that falls inside the facet and by inserting also the edges that connect p to the vertices of that facet. After the point insertion, a modified Delaunay triangulation is performed to minimize the roughness of the approximation [Rippa90]. The root is a degenerated node that points to the coarsest approximation of the terrain. Each layer built this way is more refined that the previous one and the most refined layer corresponds to the same approximation given by D. By construction, every facet induced by the model is a triangle.

A triangle **t** does not change from one level to the another if there are no points to be inserted into the area covered by **t**. In order to ensure that unchanged triangles are not replicated from one level to another, the Delaunay test is propagated only within triangles belonging to the same level. It means that the Voronoi diagram of each layer considers unchanged triangles as belonging to all of them, but they only propagate the Delaunay test at the first level that they appear. When a vertex is inserted inside a triangle **t**, we say that **t** has been refined.

Each triangle, roughly speaking, points to its three adjacent triangles (adjacency links) and to the triangles of the next level that have a non-empty intersection with it (interference links). Figure 2 shows the layers built from the DTM of figure 1 with and without the interference links. Triangles that are not the destination of any interference link do not exist in fact. Thick lines represent the border edges.

4. Data Structure to Encode the PDT

The Pyramidal Delaunay Triangulation should use an edge oriented representation in order to prevent any vertex from being considered more then once. Camera transformations over vertices are computed just once per frame. To do so, we should have an array of vertices and use the index of the position of the vertices at this array to build the triangles.

The vertex array must store for each vertex its world coordinates, camera coordinates and screen coordinates. Camera and screen coordinates are updated at every frame

exhibition. A flag to indicate whether these coordinates have already been updated is needed to avoid recalculating the projection of the same vertex twice.

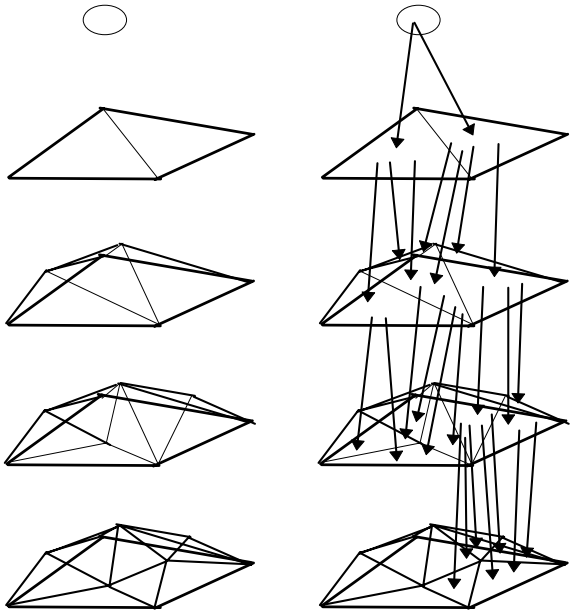

Figure 2 - A Pyramidal Delaunay Triangulation

```
typedef struct point
{
      num x,y,z,w;
}     point;

typedef struct vertex
{
      point world;
      point camera;
      point screen;
      boolean ready;
}     vertex;
vertex Vertices[MAX_VERTICES];
```

Considering the algorithms to extract a surface representation, it is desirable to store information about the edges of the triangles. Each edge that links two vertices is unique for the whole structure. Until an edge is split (point on an edge) or swapped (Delaunay test) it is kept from one level to another. This is the reason why triangle adjacency is not stored on the edges. Each edge must store its two vertices and the level which it is split or swapped (death level). A flag indicating whether one of its adjacent triangles has already been visited is needed in order to check for continuity on the frontier between two different resolutions.

```
typedef struct edge
{
     int         Vertices[2];
     int         DeathLevel;
     boolean     AlreadyPainted;
}    edge;
```

Every triangle must store the index of its three vertices and the index of its interior point with the maximum approximation error. It must also store information about its three edges and an array of pointers to the triangles of the next level (its children) that have a non-empty intersection with it (interference links). Storing the information whether the point with the maximum error lies on an edge, and the number of the edge, should help speed up the extraction algorithms. Pointers to its three neighbors (adjacency links) and to the next triangle in the influence ring are also stored in the triangle directly.

```
typedef struct triangle
{
     int      Vertices[3];
     int      MaxErrorPoint;
     int      MaxErrorPointOnAnEdge;
     edge     *Edges[3];
     triangle *Neighbors[3];
     num      BirthError;
     num      DeathError;
     int      BirthLevel;
     triangle *Children[MAX_CHILDREN];
     triangle *NextInTheRing;
}    triangle;
```

5. Building a PDT

We now describe the PDT construction algorithm. Although the final result is close to a Delaunay Pyramid, the algorithm to build a PDT is very similar to the algorithm to build a Ternary Triangulation [DeFlo+84]. A Ternary Triangulation (3T) is a multiresolution terrain model where at each step just one point is inserted inside each triangle. The main problem with this structure is that at each level, the points are inserted closer to the edges, resulting to very thin triangles. The main difference between the PDT and the 3T is that the PDT performs the Delaunay Selector algorithm after inserting the points at each level. The input to the algorithm is basically the domain of approximation D and the maximum error allowed ε.

Pyramidal triangulations suffer the problem of non-matching between the edges from one level to another. This is due to the Delaunay test that forces some edges to swap. Typically, a single edge swap is harmless to the continuity of the extracted surface. The problem arises when successive swaps of adjacent edges occur. When edge swapping leads to an edge at level i+1 that connects two interior points of two non adjacent triangles of level i, then we say that this edge is potentially harmful to the continuity of the extracted surface. This means that the resolution function may not change across such edges. Edge P_1P_2, in figure 3, is a harmful edge. Therefore, we have to link the triangles of the previous level that overlap with the harmful edge in a way that the extraction

procedure will draw them separately. These links build a ring called the *influence ring*.

We call *internal edges* the edges created to link an internal point P of a triangle **t** to the vertices of t. Anytime an internal edge is swapped, it turns into a *harmful edge*. In figure 3, edge PC is an internal edge that turns to P_1P_2 when swapped.

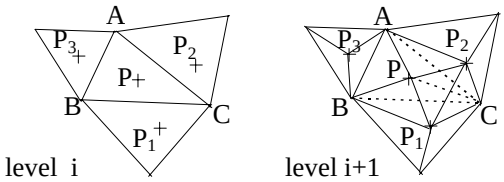

Figure 3 - Harmful edges

Influence rings are easily maintained during the tree building process. Every triangle has to keep one single pointer to the next triangle in the ring. In the beginning, this pointer is assigned to the triangle itself. Anytime an edge swap occurs, if the edge is an internal edge, then the parents of the two triangles that share the internal edge have their rings united, otherwise, the two triangles that share the swapped edge are inspected whether any of them have a non unitary influence ring. If it occurs, their rings are united. The insertion of points on the edges deserve a special attention due to the potential they have to produce discontinuities on the extracted surface. Anytime a point is inserted on an edge, the four triangles created at the new level are tied together in the same ring and the rings of their parents are also united.

An iterative algorithm for constructing a Pyramidal Delaunay Triangulation is described in algorithm 1. The building of the influence rings may also be performed during the Delaunay test procedure.

```
Build(triangle root)
{    queue Q; int Level = 1;
     for all children t of the root
     {
          t.BitrhLevel = 1;
          Q.put(t);
     }
     t =  Q.get();
     while( t != nil )
     {
          if( t.BirthLevel == Level )
          {
               P= Point with the maximum error;
               if( E(P) >= ε )
                    BuildChildren(t,P,Level+1);
               t =  Q.get();
          }
          else
          {
               DelaunaySelector(root,Level);
               BuildInfluenceRings(root,Level++);
          }
     }
}
```

Algorithm 1 - General surface building algorithm

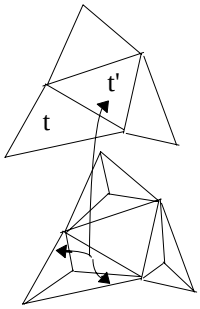

Figure 4 - Adjacency links without a direct neighbor

Sometimes, there may not be any point to be inserted in a triangle **t'** of level i. If this is the case, the triangles of level i+1 that share an edge with **t'** must have their adjacency link related to that edge pointing to **t'**. Figure 4 shows the adjacency links of a triangle with a neighbor belonging to the previous level. The Delaunay Selector performed at step 3 cannot propagate through an edge between triangles belonging to different levels

When the internal point **p** that presents the maximum error falls on an edge **e**, the triangle **t** and the corresponding neighbor will have their opposite vertices connected to **p** at level i+1, yielding only two children to each one. This situations is shown at figure 5.

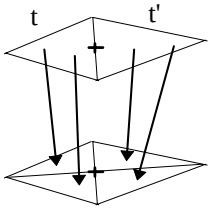

Figure 5 - Insertion of points on an edge

Considering that the number of points at level i is n_i, the height of the PDT is h, and that, from Euler's formula, the total number of triangles at level i is bounded by $2n_i$-5, the total space needed to store the triangles of the entire PDT is given by

$$\sum_{i=0}^{h}(2n_i\text{-}5) \le (h+1)(2n\text{ -}5)$$

Since the PDT allows each triangle from one level to point to every triangle of the next level, the theoretical space complexity needed to store interference links is given by

$$\sum_{i=0}^{h}(n_i\ .n_{i-1}) \le (h+1)(n^2)$$

Hence, the total space complexity to store the PDT is the same of the Delaunay Pyramid encoded with the *complete data structure* [DeFlo89] and bounded by

$$(h+1).(2n-5)+(h+1).(n^2) = (h+1).(n^2 + 2n -5),$$

where **h** is the height of the PDT and **n** is the total number of vertices on the PDT, therefore $O(hn^2)$.

However, since the PDT does not replicates an unchanged triangle **t** from one level **i** to any other level that **t** belongs, interference links are also reduced because trivial links are avoided and **t** is linked only to triangles that have a non-empty intersection with it at the first level greater than **i** that **t** does not occurs. Although the complexity of the worst case remains unchanged, the practical spatial performance of the PDT is bounded by a linear function of the number of points because the average degree of a vertex in a Delaunay triangulation is never greater than six ([PreSha85], p.211) and we insert just one point per triangle then the average number of children is never greater then twenty four. The total space required for interference links is then given by

$$\sum_{i=1}^{h}(24n_i) \leq (h)(24n)$$

and the total space complexity is bounded to

$$(h+1).(2n-5)+(h).(24n) \leq (h+1).(26n-5),$$

therefore ***O*(hn)**. In [Bert+95] an improved data structure was proposed to encode the Delaunay Pyramid called *sequence of lists of triangles* (SLT). The idea consists of storing each level as a list of triangles created at that level. As the SLT structure does not stores interference links, only extraction at a fixed resolution is allowed with low cost. The space complexity is (h+1)*(2n-5) and the number of stored triangles may be much smaller if the refinement process modifies few triangles at each level.

6. Surface Extraction from a PDT

A PDT allows extraction of surfaces at either fixed or variable resolution for resolutions with error values greater then the error tolerance ε given when the PDT was built.

Surface extraction is performed according to a resolution function **R** that checks whether a triangle is to be rendered or not. One such surface can be extracted from the PDT through a breadth-first traversal that starts at the root and selects for each level the triangles that satisfies the resolution function and have to be painted. Each triangle is checked twice because a triangle that satisfies the resolution function may be refined if some triangle of its influence ring does not satisfies.

For fixed resolution extraction, the resolution function selects the triangles that have the birth error less or equal to the desired extraction resolution, following the ideas of the HyperTriangulation [CiPuSc95]. For landscape viewing or flight simulation, the resolution function should select triangles based on their distance to the observer or the size of their projection on the view plane. Algorithm 2 illustrates the general surface extraction algorithm.

When extracting a surface at variable resolution we must be aware of discontinuities that may occur at the frontier between two different resolutions. Such discontinuities are related to the death of an edge, i.e., the edge exists in the coarser level and has been swapped or split at the more refined level. Therefore, *the only safe situation to render a triangle ensuring continuity is when all of its neighbors satisfies the resolution function.* Hence, if a triangle **t** satisfies the resolution function **R** and a neighbor **t'** of **t** does not satisfy **R**, then we must handle this situation properly.

```
Extract(triangle root)
{
     queue CurrentQueue, NextQueue;

     for all children t of the root
     {
          CurrentQueue.put(t);
     }

     do
     {
          First = t = CurrentQueue.get();
          do
          {
               if( !R( t ) )
               {
                    CurrentQueue.PutRing(t);
               }
               else
               {
                    CurrentQueue.put(t);
               }

               t = CurrentQueue.get();
          }
          while( t != First );

          do
          {
               t = CurrentQueue.get();
               Render(t, NextQueue);
          }
          while( not CurrentQueue.empty() );

          CurrentQueue = NextQueue;
          NextQueue.Reset();
     }
     while( not CurrentQueue.empty() )
}
```

Algorithm 2 - General surface extraction algorithm

When we insert a point **P** inside a triangle **t** to build the next level, two or three triangles are created and we call them the *virtual children of **t***. We say they are virtual because they may not exist as children of **t** at the next level due to a Delaunay edge swap. In figure 6, the virtual children of triangle ABC are the triangles PAB, PBC, PAC. Triangle PBC is also a *real child* of triangle ABC.

Since a virtual child may not exist in the final triangulation, it is not safe to maintain their interference links to the next level. They only need adjacency links to their real neighbors.

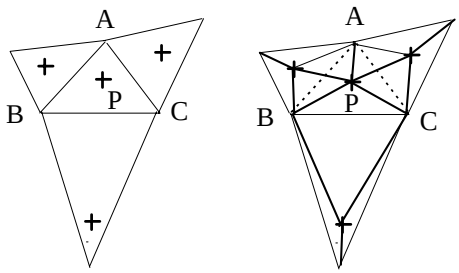

Figure 6 - Typical Triangle Refinement

Virtual children are used to handle non harmful edges. When a triangle **t** does not satisfy the resolution function, it means that it must be refined (its children must be checked) and so its entire ring of influence. When a triangle **t** satisfies the resolution function **R** and its influence ring contains only itself then, if a neighbor **t'** of **t** does not satisfy **R**, the virtual child of **t** corresponding to **t'** will be put in the rendering queue if **t'** belongs to a coarser level (did not change from one level to another) or t' satisfies the resolution function **R** or their common edge **e** is alive in the next level or **e** dies due to an splitting or **e** has already been painted.

The absence of interference links means that virtual triangles are ease and cheap to create during the tree creation process with a constant spatial cost per real triangle. In figure 7, the virtual children of triangle ABC are the triangles PAB, PBC. Triangle PBC is also a real child of triangle ABC.

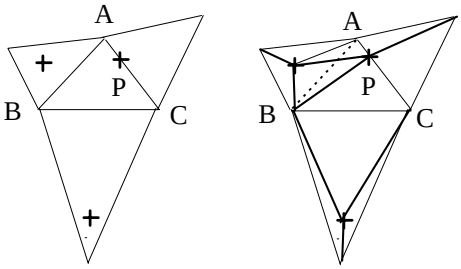

Figure 7 - Triangle with Maximum Error on an Edge

Function `Render`, shown in algorithm 3, directly paints the triangle if it is a virtual triangle or if it has no children or if all of its neighbors satisfy the resolution function. If the triangle does not satisfy the resolution function **R** then its real children have to be put in the queue of the next level, NextQueue. On the other hand, if some neighbor does not satisfy the resolution function then its virtual children are put on the queue of the next level according to the situations described above.

```
void Render( triangle t, queue &NextQueue )
{
     if( t has no children or t is virtual)
     {
          Draw(t);
     }
     else
     if( !R( t ) )
     {
          NextQueue.PutChildren(t);
     }
     else
     if(R holds for all neighbors of t)
     {
          Draw(t);
     }
     else
     {
          for all triangles t' neighbors of t
          {
               e = common edge of t and t';

               if(  t' is from a coarser level or
                    R( t' ) or
                    e does not die due to a swap or
                    e.AlreadyPainted )
               {
                    NextQueue.PutVirtualChild(t,e);
               }
          }
     }
}
```

Algorithm 3 - Render Algorithm

Considering that all operations performed by function `Render` on a triangle require constant time and that the total number of visited triangles is linear with the number of the extracted triangles, **n**, time complexity of the extracting operation is ***O*(n)**, where **n** is the number of the extracted triangles.

7. Conclusion

We have tested the PDT building TINs from regular square grid distribution and irregular distributed input set. In both cases we have used a decimation technique based on the error of the approximation to eliminate redundant information on the terrain. Figure 8 illustrates two images from the same view of a terrain data in a 120×120 grid and the image corresponding to the difference between them. The first view is rendered with full resolution accepting an error tolerance of 0.1 pixel. The second view is rendered with variable resolution checking the perimeter of the projected triangle as the resolution function. The full resolution mesh has 17759 triangles and the coarse one has 14485 (81.55%). The difference image corresponds to an error of 2.38%. Figure 9 shows the corresponding meshes of both rendered view.

We have considered our results very encouraging and appropriate to virtual environment modeling where interactive fly-through is necessary. Now we are

interested in adapting the model for very large data input and multiresolution models for arbitrary objects placed on the terrain.

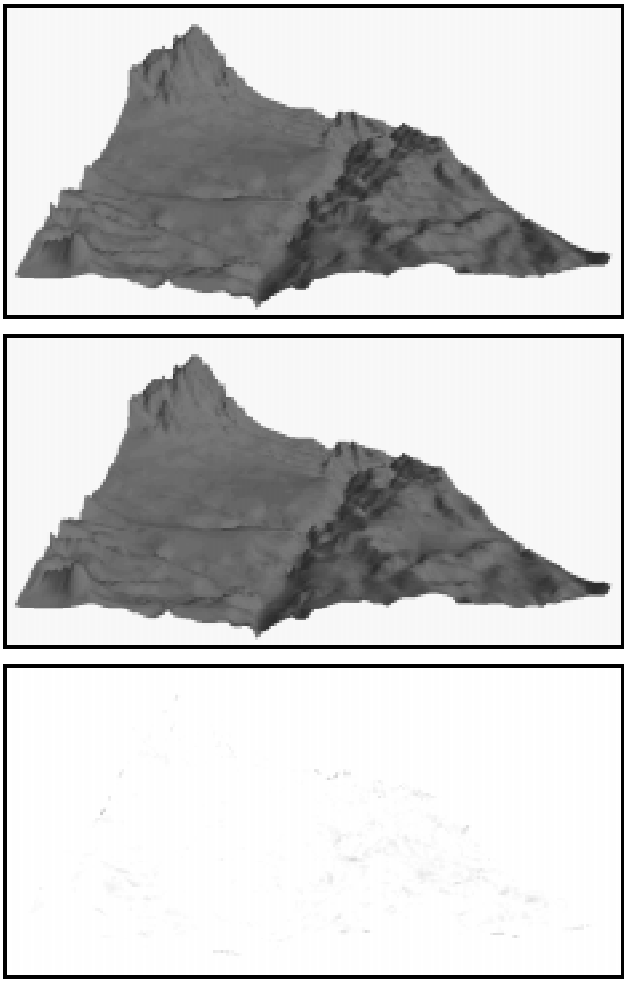

Figure 8 - Full and variable resolution views

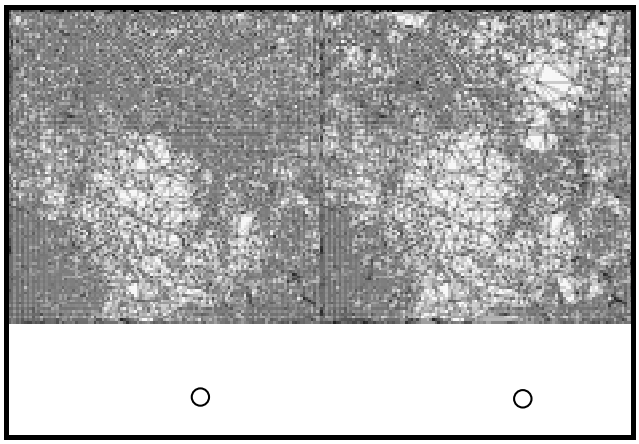

Figure 9 - Full and variable resolution meshes

Acknowledgments

I would like to thanks to Ana Cristina Garcia for her unconditional faith on this work and to Airam Marques for implementing and discussing the algorithms presented in this paper.

References

[Bert+95] - Bertolotto, M., De Floriani, L., Marzano, P., "Pyramidal Simplicial Complexes", ACM Symp Solid Modeling and Applic., Salt Lake City, Utah.

[CiPuSc95] - Cignoni,P., Puppo,E., Scopigno, R., "Representation and Visualization of Terrain Surfaces at Variable Resolution", Intl. Symp. Scientific Visualization, World Scientific, Cagliari, Italy.

[deBDob95] - de Berg, M., Dobrindt, K.T.G., "On Levels of Detail in Terrains", Utrecht University, tech.rep CS-1995-12.

[DeFlo+96a] - De Floriani, L., Marzano, P., Puppo,E., "Multiresolution Models for Topographic Surface Description", The Visual Computer 12, 7:317-345.

[DeFlo+96b] - De Floriani, L., Magillo, P., Bussy, S., Bailey, E., "Triangle-Based Surface Models", Genova University,tech.rep.DISI-TR-??-??

[DeFlo89] - De Floriani, L., "A Pyramidal Data Structure for Triangle-Based Surface Description", IEEE Computers Graphics & Applications 9,2:67-68

[DeFlo+84] - De Floriani, L. et al. "A Hierarchical Data Structure for Surface Approximation", Comp. and Graphics 8, 2:475-484.

[DeFPup95] - De Floriani, L., Puppo,E., "Hierarchical Triangulation for Multiresolution Surface Description", ACM Transactions on Graphics 14, 4:363-411.

[GarHec95] - Garland, M., Heckbert, P., "Fast Polygonal Approximation of Terrains and Height Fields", Carnegie Mellon University, tech. Rep. CMU-CS-95-181.

[PreSha85] - Preaparata, F., Shamos, M., "Computational Geometry: An Introduction", Springer Verlag, 1985.

[Puppo96] - Puppo, E., "Variable Resolution Terrain Surfaces", Genova University, tech.rep DISI-TR-96-6.

[Rippa90] - Rippa, S., "Minimal Roughness Property of the Delaunay Triangulation", Computer Aided Geometric Design 7:293-301.

[SchRoβ94] - Schöder, F., Roβbach, P., "Managing the Complexity of Digital Terrain Models", Computers and Graphics 18, 6:775-783.