Principles of Computer Graphics

Implementation of Hybrid Terrain Representation in Nasa WorldWind: Regular Grid and Triangulated Irregular Networks (TIN)

Raffaele De Amicis, Giuseppe Conti

Alessio Guerrieri
pufforrohk@gmail.com

Gavriel Smith
gavri.smith@gmail.com

Yohanes Baptista Dafferianto Trinugroho
thezultimate@gmail.com

Introduction

Terrain in WorldWind is modeled using a regular grid. This means that for small areas with large differences in heights the image rendered using a regular grid is not very accurate. This can not be solved by using a higher resolution regular grid since such an approach will use up a very large amount of memory. A solution to the problem is to refine the regular grid by adding a TIN (Triangulated Irregular Network) that represents more complex terrain areas. This TIN is comprised of a set of points which are able to describe irregular surfaces more accurately than a regular grid because the points do not have to be in constant intervals. Adding such a TIN to a regular grid creates a hybrid terrain. In such a terrain, areas with constant heights are described using the points from the regular grid and therefore are described by using a small number of points. Irregular areas are described by using the TIN and therefore described in higher detail. Figure 1 shows an example of a TIN.

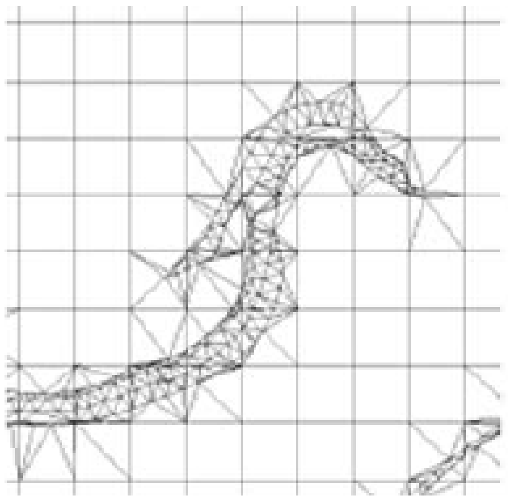


Figure 1: An example of a TIN, added to a regular grid

In this project we incorporated the use of a TIN inside WorldWind's regular grid. Direct rendering of both the regular grid and the TIN would generate geometric discontinuities. We used several methods to eliminate these discontinuities.

Keywords: Tessellation, triangulation, triangle blobs, tears, skirts, simplification, LOD (Level of Detail), convexification.

Terrain representation in WorldWind

WorldWind uses a regular grid. In the WorldWind implementation, the world is divided into sectors. The size of a sector depends on the LOD (Level of Detail), as does the distance between the points in the regular grid. Passing from one LOD to the next one the sector is divided into four smaller sectors. The LOD depends on the distance from which the user views the terrain.

Every sector is triangulated and rendered independently. In addition, in order to prevent tears on the boundary between sectors, a vertical “wall” is rendered, from the boundary of the sector, downwards. These are called “skirts”. This method means that the user cannot see the tears, because the skirts blend into the landscape. Example of skirts is depicted in Figure 2.

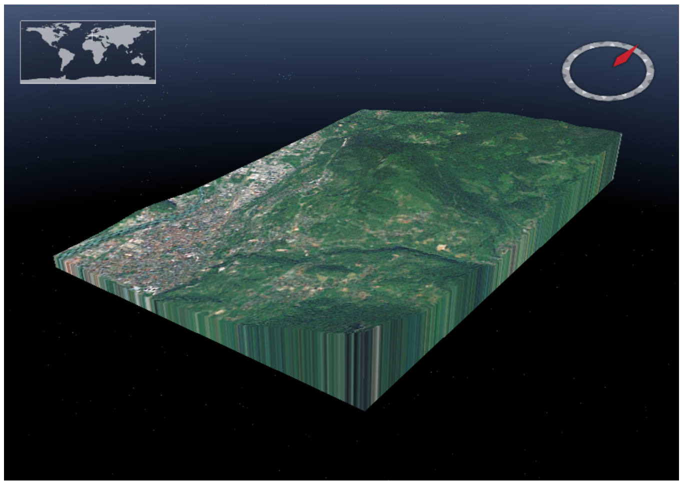


Figure 2: Skirts in WorldWind

In our implementation, sectors which do not intersect with the TIN are rendered as usual, by the WorldWind code. The sectors which intersect with the TIN, however, are rendered by our code. The TIN file we receive is just a list of points, without triangles. For each sector we compute the points from the regular grid which lie in the sector and we add them to the set of points from the TIN which lie in the sector. This set of points is then passed to a triangulation process, which treats all the points equally, without differentiating between the points from the TIN and the points from the grid. In other words, the triangulation is done sector by sector.

Triangulation

Given a set of points (some from the regular grid, some from the TIN) we want to compute a set of triangles that covers all the surface. The approach we decided to use is to divide the process into two steps: first we compute a simple triangulation using a simple iterative algorithm, and then we optimize it to obtain a Delauney Triangulation.

A Delauney Triangulation is a special type of triangulation which has several interesting properties. This kind of triangulation ensures that for every triangle there is no other point in the data set that lies inside the circle defined by the three point of the triangle. Such a triangulation is unique for any set of points (so the result is the same with any algorithm that computes the triangulation). This property ensures that the triangles are as close to being equilateral as possible.

The simple triangulation is computed starting with a triangle made of three random points. Then, for every point, we check if it's inside a triangle. If that is the case, we break that triangle into three smaller triangles. If it is outside the triangle we connect the point with all the visible edges of the convex hull of the triangulation already computed. This implementation is quite slow, since we do not use any complex data structure like quad-tree to find the triangle that encloses a point.

After the triangulation we apply a flipping algorithm, which is the simplest way to compute a Delauney Triangulation from a given triangulation. For every pair of neighboring triangles we check if the situation is improved by switching the diagonal. Every time we flip a certain pair of triangles, we need to check whether or not to flip the neighboring triangles. We use a queue of triangles that have to be checked, and we make sure no triangle appears twice in the queue. A list of possible algorithms for computing the Delauney Triangulation, including ours, are presented in [1].

Figure 3 shows how the flipping algorithm works.

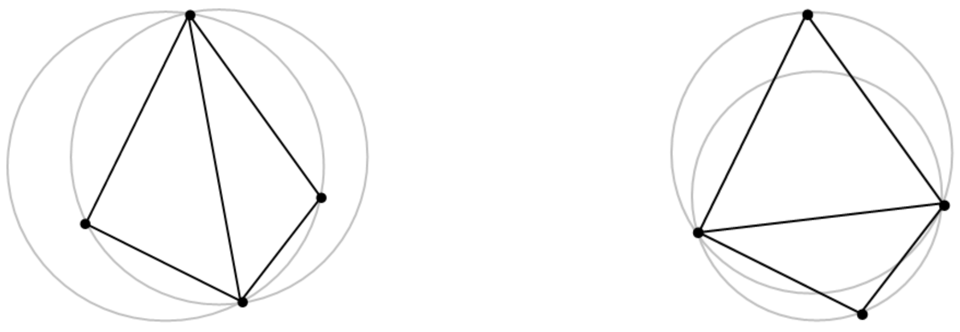
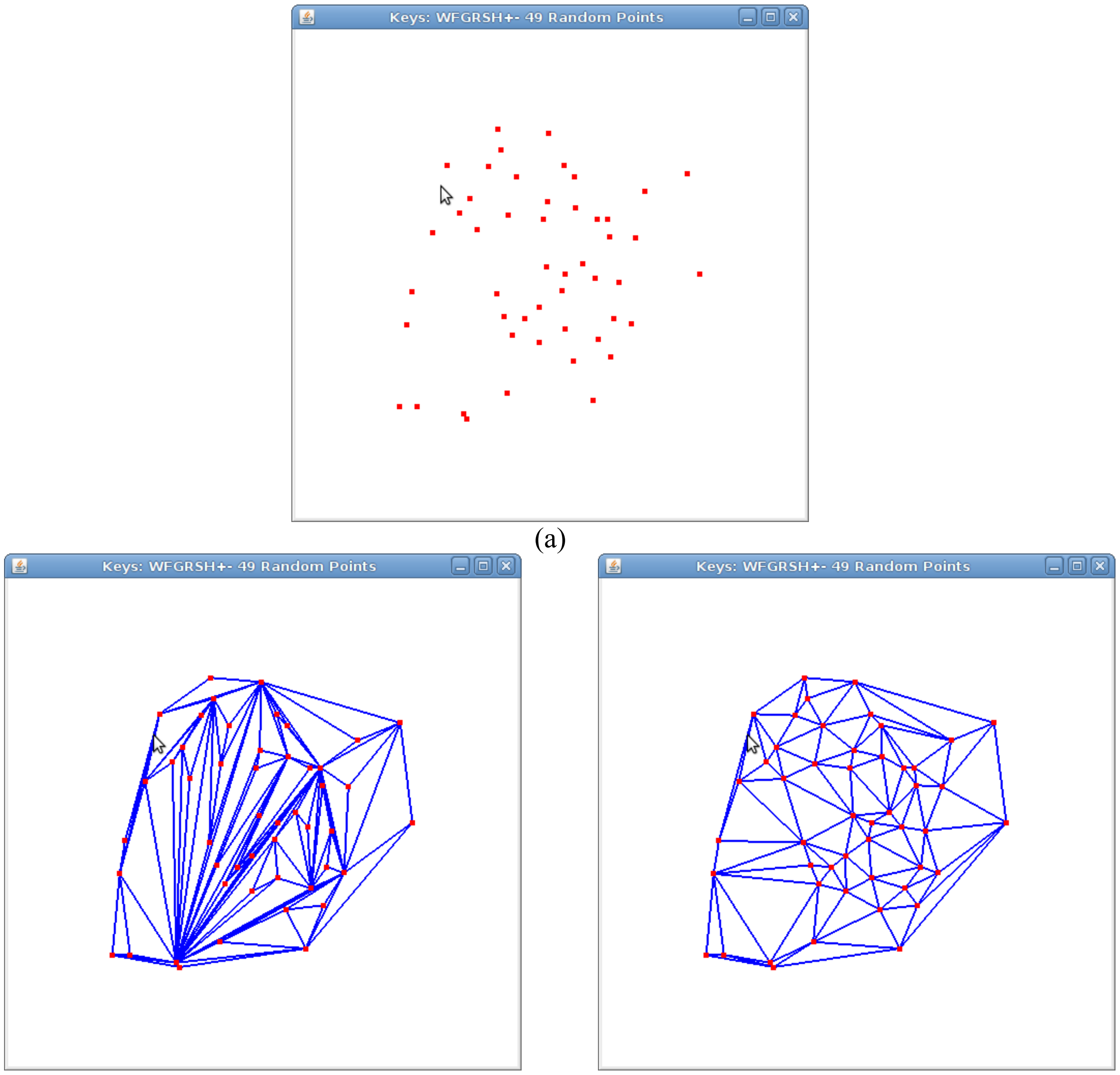

(a) (b)

Figure 3: Flipping algorithm: (a) before flipping; (b) after flipping

Figure 4 below depicts a simulation of our triangulation which incorporates the flipping algorithm.



Keys: WFGRSH+- 49 Random Points
(a)
Keys: WFGRSH+- 49 Random Points
Keys: WFGRSH+- 49 Random Points

(b) (c)

Figure 4: Simulation of flipping algorithm: (a) set of points before triangulation; (b) before applying the flipping algorithm; (c) after applying the flipping algorithm

An approach which we used at the beginning of our project was one of using triangle blobs. This idea assumes that the TIN does not use points from the grid. For each sector which is completely covered by the

TIN, it is replaced by the set of triangles from the TIN which creates a shape that resembles the sector best. Such a shape is called a triangle blob. Then, when rendering the TIN, instead of rendering it sector by sector, the rendering is done blob by blob. We decided against using this method, because adding points from the sector to the TIN does not harm the accuracy of the TIN, and this makes the implementation much simpler. For this reason, we use points from the regular grid in our TIN, and so the shape of the triangles that lie in a certain sector, is exactly the shape of the sector.

As mentioned above, WorldWind uses a LOD approach. In our implementation, every time a new LOD is needed, we calculate the points from the regular grid corresponding to this LOD, and add them to our TIN. Then we simplify the TIN, taking the LOD into account (if it is a high LOD, we need to render a large number of points from the TIN, and vice versa). This gives us a TIN which is in the form of a set of sectors. In addition, we can create the skirts the same way that world wind does (they are added to the TIN after the triangulation).

In addition, WorldWind renders the triangles using the "triangle strip" setting, for performance reasons. We do not create our triangles in a way that enables us to draw them as a strip, so we use the "triangle list" setting instead.

Tessellation

One approach to tessellation is to convexify the TIN. Then, to decide for each sector whether it is not covered, partially covered, or completely covered by the TIN. Then, triangles are added to the TIN, in the sectors which are partially covered, to make them completely covered. This process is called tessellation. In this way, the convex hull of the TIN consists of a list of sectors. Then, when the terrain is rendered, not covered sectors are rendered directly, and the TIN is rendered. This way, all the terrain is rendered, because completely covered sectors are rendered as part of the TIN. This algorithm is described in [2]. In addition another algorithm is described in [3].

We do not need to use this approach, because we do the triangulation ourselves, so we simply calculate the bounding sector of our TIN, and add the points from the regular grid which are inside the sector to the list of points in the TIN. Then, after the triangulation, the convex hull of the TIN is exactly this bounding sector, and no tessellation is needed.

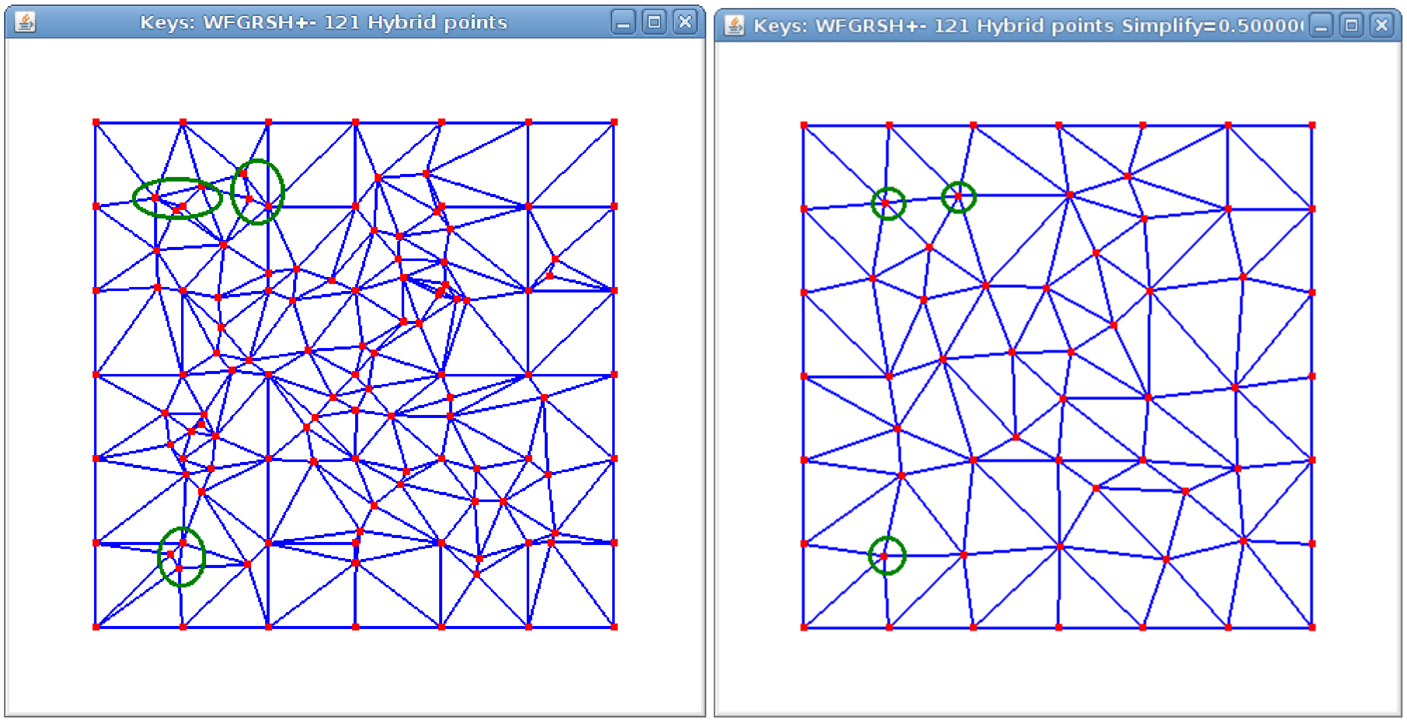
Simplification

As mentioned above, WorldWind works using an LOD approach. If the terrain is rendered from a large distance, rendering it using the highest LOD would be very slow and wasteful since the user cannot see the difference between this kind of rendering and rendering using a lower LOD. For this reason WorldWind renders the terrain according to the distance from which the user views the terrain. This must be considered while rendering the TIN for the same reason: rendering the entire TIN when viewing the terrain from a large distance would be wasteful and slow. We solve this problem by simplifying the TIN according to the LOD.

Many algorithms exist for TIN simplification. For instance, one algorithm starts from a set of points (for instance, the convex hull of the TIN) and at each iteration, the point from the TIN which has the highest error is added to the list of points [4].

The algorithm we use is an adaptation of an agglomerative hierarchical clustering algorithm which we learned in machine learning. On the initialization phase, every point is put in a separate cluster. At every iteration the algorithm chooses the two clusters which are nearest to each other and joins them to one cluster. The position of each cluster is simply the average of the points in the cluster. We do not change the boundary points during the algorithm, and so the boundary of the sector does not change. The algorithm terminates when the number of clusters is bellow a certain number, received as input (given a priori). In our case, this number is proportional to the LOD. Some similar clustering algorithms are presented in [5].

One decision we had to make was whether the simplification should be before the triangulation, after it, or to do both simultaneously. We decided first to simplify and then to triangulate, for efficiency reasons. Simplifying after or during the triangulation means updating the triangulation according to the simplified set of points, which can be more complex.


Keys: WFGRSH+- 121 Hybrid points
Keys: WFGRSH+- 121 Hybrid points Simplify=0.50000

(a) (b)

Figure 5: Simulation of 2D triangulation: (a) before simplification; (b) after simplification. In this case, the simplification returns half the number of initial points.

Creating a TIN

In order to be able to see the difference between our implementation of a hybrid terrain, and the normal WorldWind implementation, a TIN sample is needed. We create such a sample to be shown by our implementation. We use the functions from WorldWind to get the elevation in a certain point, given the latitude and longitude, and with this information we can create such a TIN file. This way, we can see that the hybrid terrain is being rendered. For testing, we created some TIN files which have mountains or valleys that do not exist in reality.

In order to create a high resolution TIN, we sample a list of points from the highest resolution possible in WorldWind. Then, in order to see the difference between the sectors which we render, and the sectors rendered by WorldWind, we restrict WorldWind to working only up to a lower resolution. Some results are shown in the following paragraph.

Results

Below, we present some screen-shots of our implementation of a hybrid terrain representation in WorldWind, which uses our TIN (i.e. a mountainous area). In order to be able to view our rendering of the hybrid terrain in better way, our program allows to move from normal view to wireframe view, and from the normal WorldWind implementation to our one.

Figure 6 depicts a low resolution terrain of a mountainous area where the entire terrain is rendered in the same resolution. Figure 7 shows screen-shots of our implementation of a hybrid terrain representation in WorldWind with our TIN. The TIN has a better resolution than its surroundings, making it look more detailed. Figure 8 shows a wireframe view of our hybrid terrain implementation. The presence of the skirts causes the view to be a little cluttered.

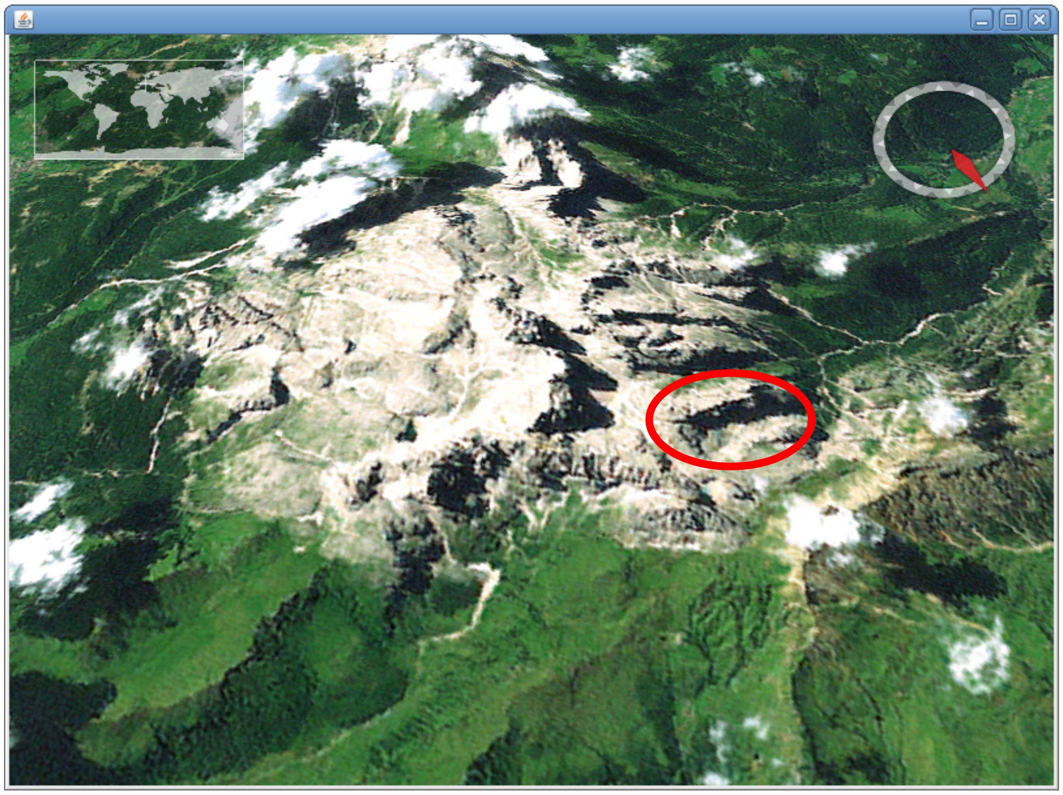


Figure 6: Low resolution terrain without our TIN

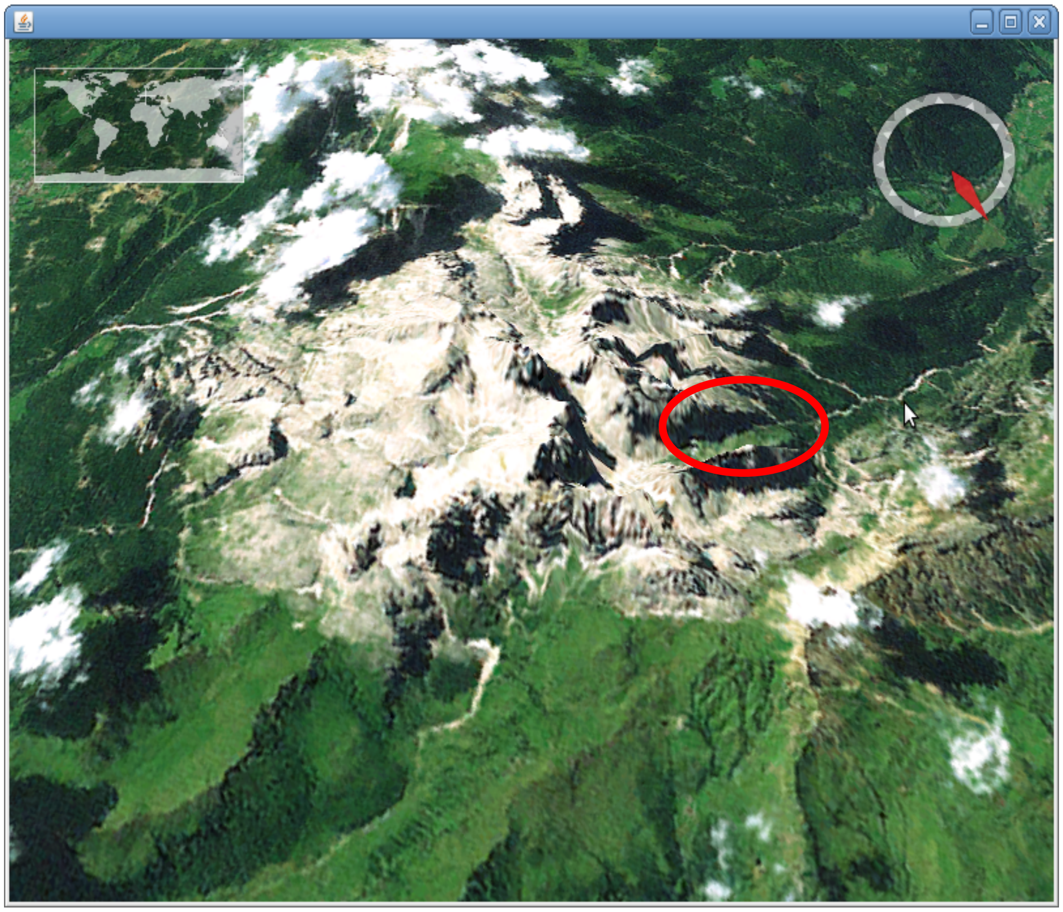


Figure 7: Hybrid terrain: low resolution terrain with our TIN

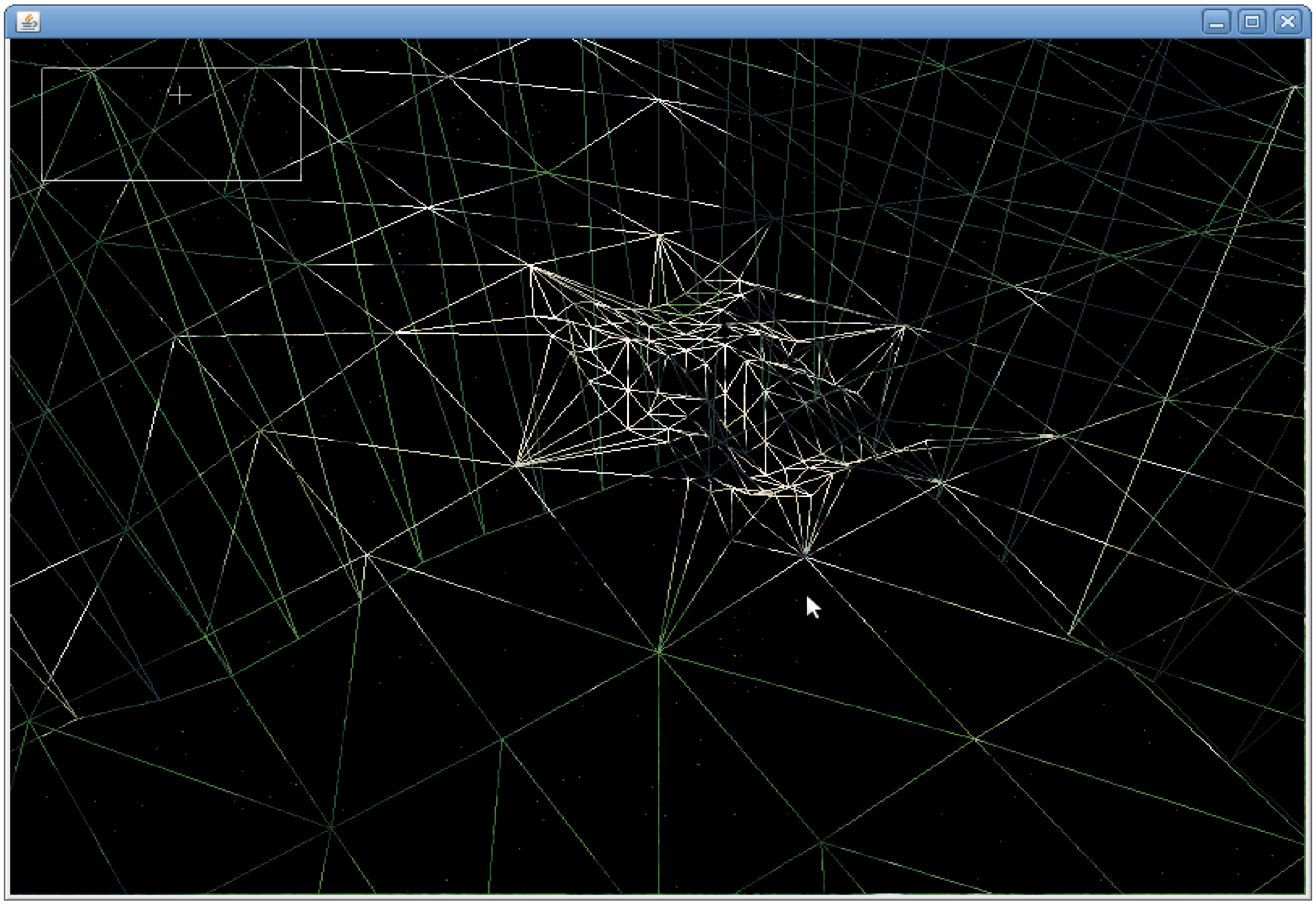


Figure 8: Wireframe of the hybrid terrain

Bibliography

1: M Bern, D Eppstein, Mesh Generation and Optimal Triangulation, 1992

2: M. Bóo; M. Amor; J. Döllner, Unified Hybrid Terrain Representation Based on Local Convexifications, 2007

3: Pajarola, Renato; Antonijuan, Marc; Lario, Roberto;, QuadTIN: Quadtree based Triangulated Irregular Networks, 2004

4: Michael Garland and Paul S. Heckbert, Fast Polygonal Approximation of Terrains and Height Fields, 1995

5: P Berkhin, Survey of clustering data mining techniques, 2002