

IDST 258: Term Paper

Kunal Mittal

Fly by model of Beloit

Group Members: Rajeewa Basnayake

Kunal Mittal

Introduction:

"Little has been written about the transformation that occurs when we look at a map, a process that begins with physiology and perception but ends up as knowledge and worldview. Yet maps represent one of the fundamental tools by which we make sense of the world around us." --Stephen Hall, Mapping the Next Millennium: The Discovery of New Geographies.

A Geographic Information System (GIS) can be defined as 'an organized collection of hardware, software, geographic data, and personnel designed to efficiently capture, store, update, manipulate, analyze, and display all forms of geographically referenced information. (Source: "Understanding GIS: The ARC/INFO Method", Environmental Systems Research Institute).

Our objective is to build a 3D-flyby model of Beloit City using the 3D Analyst extension of Arc View. It is our belief that if a 3D model of Beloit City can be created successfully,

it would be a powerful visualization tool that can be of immense use to both individuals and groups involved with the day-to-day activities of the city.

For example,

- City planners, architects and the people involved with the Beloit 2000 project could use our model to run feasibility tests on proposed construction projects. With the powerful features found in Arc View and 3D Analyst, they could use the model as a base and manipulate its environment to simulate their scenarios.
- Marketing firms can use the flyby model to determine suitable locations for their billboards – effective dimensions, face direction where traffic is most etc.

Personally, I had some other interests in doing a project of this nature. I am a Computer Science major, and along with this class, I am doing a class in Algorithms for which my term project was a visual demonstration of two Convex Hull algorithms. (You can view this online - <http://stu.beloit.edu/~mittalk/ConvexHull.html>). In that class, we also studied Voronoi polygons. My research showed me that Voronoi polygons and Delaunay triangulation plays an important role in the creation of TINs (triangulated irregular networks). For our project, we are working with TINs and using 3D Analyst to do the fly by. Thus, I decided to include some of the theory behind this from the prospective of a Computer Scientist. I will discuss what Voronoi polygons and Delaunay Triangulation and I will describe few algorithms used to create TINs using these concepts. I go into reasonable detail on these issues as they were of key interest to me as I worked on this project along with Rajeewa.

I will also present the methods I used in the course of the project and describe in detail the problems I faced and provide potential solutions to these problems. I will also demonstrate the procedure used to convert the raw DEM into the final fly-by movie file.

Method and Problems:

At first we had problems obtaining a decent DEM or PNT file to work with. We did try working with a large PNT file and once we figured out how to import it into Arc View, we realized that it would be of no use. Thus we settled to work with a small seven and a half-minute quadrangle DEM of Beloit and imported that into a new project created in Arc View. This by itself was not an easy task. Without any previous knowledge of Arc View, we had to struggle to make sure that the correct extensions were loaded and we carried out the steps of selecting the extensions and then creating a new project in the same order. Another annoyance is that Arc View saves temporary theme files into the systems temp directory. Thus we need to make sure that we set “the working directory” appropriately otherwise all our work will be lost.

Once we had this done, we were then in a position to install the fly-by scripts (downloaded from the ESRI web site) and construct TINs, which could, then appropriately be used for the fly-by.

Working with various configurations of the TINs and Grids, we noticed that we had to convert back and forth from these in order for us to get enough elevation information.

The DEM we started of with was not very detailed and extracting information from it was going to a challenge. This is mainly because Beloit is flat and thus if the DEM is not digitized specially to a very small degree of accuracy, the elevation data reads almost the same. We solved this problem by converting the original imported DEM into a TIN and then back to a Grid. By doing this, we were able to increase the number of rows in the output Grid to get finer data. After different conversions, we settled on using the number of rows for the grid to be 750. Then we converted this Grid back to a TIN, with an aspect factor of 75% of the default. This gave us a TIN with reasonable elevation data. To confirm that this was a reasonable model, we constructed a contour map of the same with the contours spaced at 75. This showed us that the data had not lost its accuracy and the contours for the Beloit City area yet represented a flat terrain.

At this point, we were ready to apply the fly-by script and demonstrate a fly-by. The fly-by script has many options. Playing with these we came up with a collection of sample movie files, one of which we have shown in our presentation. It is possible to actually plot the path our plane should take through the image, the height it should fly at and the speed (frames per second). We plotted a path through the map and selected appropriately the 3D-display scene that we wanted. Once we had all this in place, we could take a minute to relax, as we knew that we were done.

We tried many of the other options like computing the hill shade and creation of contours from a base point file. These however were a total failure. The concepts of hill shades and aspect ratios do not apply to flat city maps. We did not realize this initially and kept

wondering what were we doing wrong. Furthermore, we tried creating different charts to represent our data, but never found anything that we thought we could use to construct the fly-by. This will be an exercise for anyone who uses our fly-by to get more detailed information. Our aim was not to get all the data that one can from a map of this sort, but just to create a process that one can duplicate to create a fly-by for specific needs.

Basic Concepts of the Geometric Algorithms behind 3D Analyst:

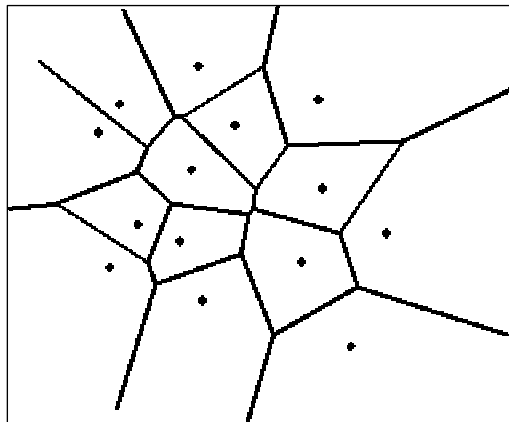
1. Voronoi Polygons

The Voronoi diagram is named after the mathematician M. G. Voronoi who explored this geometric construction in 1908. The Voronoi diagram is one of the most fundamental and useful constructs defined by irregular lattices. It is a widely used geometrical construction and is utilized in many distinct fields. The specific Voronoi diagrams discussed are often named after the person who first used it in that particular field. A. H. Thiessen was the first to use Voronoi diagram in a way that brought many people to develop a keen interest in this idea. He used the Thiessen polygons to define regions that surround unevenly distributed weather stations. Data from each weather station could then be presented in the enclosing polygon of the station. The polygons were constructed so that “regions be enclosed by a line midway between the station under consideration and surrounding stations”.

Although there is a broad spectrum of scientific disciplines that include interest in Voronoi diagrams, states that three aspects have been emphasized:

1. Their use in modeling natural phenomena.
2. The investigation of their mathematical, in particular, geometrical, combinatorial, and stochastic properties.
3. Their computer construction and representation.

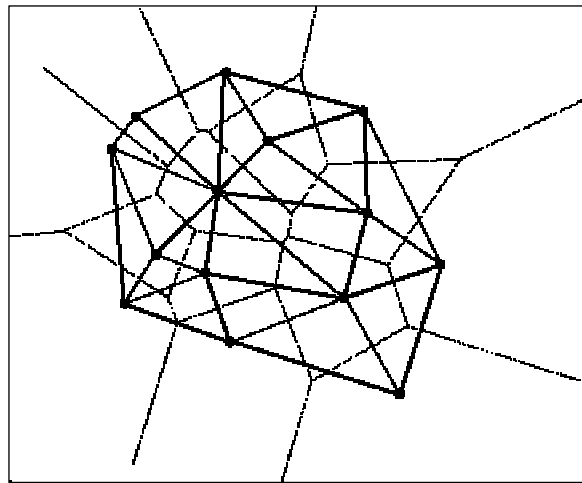
The Voronoi diagram for a set of random distributed points is shown below. Notice that the polygons on the boundary of the area are “open” because they have no neighboring points in that direction.



2. Delaunay triangulation

Delaunay triangulation is closely related to the Voronoi diagram. The triangulation is named after B. Delaunay, who first made use of this dual relationship. If we use the Voronoi diagram as a basis, we can construct the Delaunay triangulation by drawing the

lines between the points in adjacent polygons. When the construction is finished we have got a triangular network that covers the whole area. For many purposes we do not need the Voronoi diagram, but we want to make the Delaunay triangulation directly. Actually the generation of a Delaunay network in a computer is simpler than the construction of a Voronoi diagram. This forms the basis of many algorithms used to construct TINs. As a computer scientist, we obviously try all our standard algorithmic paradigms to try and develop an optimal algorithm. Our concepts of divide and conquer and dynamic programming have allowed us to develop reasonable algorithms that are fast and quite memory efficient.



3. Construction of a TIN

When a set of arbitrarily distributed data points is connected in a triangular network, this lattice consequently has to be irregular. If we want to make a regular network we usually have to interpolate new points, ordered in a regular pattern. Now I will describe some

algorithms for the generation of a TIN as a structure for surfaces, with emphasis on terrain surfaces. Delaunay triangulation is the most common technique for construction of a TIN. When the measured data points are well adapted to the terrain, the mathematical rules for Delaunay triangulation result in a triangular network well proportionated for the terrain surface. I will describe three algorithms that I particularly find interesting.

1. Recursive split algorithm.
2. Divide-and-conquer algorithm.
3. Incremental delete-and-build algorithm.

Since this paper is not directed towards computer scientists, I will skip the details on the data structures used to store the various data used by the algorithms and also the factors affecting the actual running time of these algorithms.

Recursive split algorithm

The main purpose of the algorithm is to split the area recursively into smaller and simpler regions. In the end we will have regions that only consist of three points forming a triangle. The algorithm goes:

- A. The region is split into two sub-regions. It is important to keep the sub-regions as “circular” as possible. The “circularity” is determined by measuring the product of the signed distances between the boundary points and the split line. When the region is split, the boundary between the two sub-regions must be chosen.

- B. The new sub-regions are recursively divided into new sub-sub-regions until each region consists of only three points. The root region now consists of non-overlapping triangles. Due to the definition of the boundary between two regions, the algorithm will produce quite a number of poorly shaped triangles (long and thin). Each quadrilateral in the triangulation is examined and angles are calculated. If the new triangles are chosen and the diagonal is swapped. The examination of the quadrilaterals goes on until no more changes appear.

Divide and Conquer Algorithm

This algorithm is comparable to the recursive-split algorithm. It divides the original data set into disjoint subsets. After obtaining a solution for each of these subsets, it combines the solutions to yield the final result.

- A. As in the recursive-split algorithm, the data set is first divided into two subsets. There is no calculation of the circularity of the sub-regions, but a division into two data sets that contain an equal number of points. The area is recursively split until each data set consists of a minimum of four points.
- B. Triangles are made at the lower level, and the merging of these data sets starts. In the recursive split algorithm a boundary for the sub-region was set during the split operation. In this algorithm the convex hull that circumscribes each data set defines the region. In the recursive process each region is merged together until the whole area is completed.

Incremental delete-and-build algorithm

The most important difference between this algorithm, and the two others discussed earlier, is the possibilities of keeping the triangular network as a Delaunay network during the triangulation process. The great advantage of this algorithm, compared to the previous ones, is the possibility of examination and calculation to determine whether any points make a significant contribution to the network. A plane is defined by three points only. An inclusion of a fourth point in the plane will not result in any improvements to the surface representation. The initial value for the triangulation is a valid Delaunay network (at least one triangle). For each point included in the network, the network will be rearranged until the circle criterion is met for all the triangles in the network. This is a dynamic algorithm and very easily parallelized.

- A. As in the previous algorithm an initial triangular network has to be created. The triangular network of the convex hull is used for this purpose. The network has to meet the maximum angle-sum criterion. The triangulation programs use the circumscribing rectangle, dividing them into two triangles.
- B. The first point of the interior area is included in the network. The point is connected to its enclosing triangle by three new triangle edges between the point and the vertices of the triangle.
- C. The quadrilaterals, which have got the old edges of the enclosing triangle as a diagonal, have to be tested by the maximum angle-sum rule. If they do not meet the criterion, their diagonals are swapped and the new, opposite edges to the inserted point will be examined as diagonals in their quadrilaterals.

D. The Delaunay network now contains one more point. All the remaining points will be included in the network in exactly the same way.

Conclusion:

Though this entire procedure sounds simple enough and trivial to implement, it was a time consuming procedure for us as we had very little experience with doing this and had to try multiple different things to finally figure out that this is the simplest way to achieve what we wanted. A major part of our time was lost in finding sources of information, reading help files and going through tutorials. We spent copious amounts of time learning how to use Arc View. Figuring out how to use 3D Analyst and the various options in the fly-by scripts was an interesting experience. Personally, I picked up another algorithmic idea while reading through the documentation provided by the fly-by script. This was the description of how Bessiers Algorithm is used to construct the fly-by.

Both Rajeeva and me have many ideas on how we could have come up with a more useful fly-by model of the Beloit area. The process of importing a DEM and applying the themes as described above would have yielded much better results if we had a more detailed and larger DEM to work with. Due to time constraints we were not able to digitize a map ourselves or obtain one from somewhere else.

Working with new equipment and new software is always a challenge. We successfully navigated our way through these to construct a successful fly-by using the limited maps that we had access to.

Appendices:

ample images of DEMs of the Beloit City after conversion to TINs and enhanced elevation data. Note that the images have been rotated using 3D Analyst to provide different views.

