# Design of an Integrated Feature and Terrain Triangulation Algorithm for Driving Simulation Terrain Database Generation

**Project ID# 4**

**<u>Final Report to the Center for advanced Transportation Systems Simulation</u>**

**April 27, 2000**

**Guy A. Schiavone, Art Cortes, Jian-ping Gu, Ying Dai, Grace Yu,
Walter Wimberly, Christian Buhl, Ryan Leitch**

**Visual Systems Laboratory
Institute for Simulation and Training
University of Central Florida**

# I. Two surface simplification algorithms for polygonal terrain with integrated road features

## ABSTRACT

Terrain database generation is one of the most expensive tasks in the development of human-in-the-loop visual simulations. There are many factors associated with the efficiency of generating the terrain database. Automating the process of extracting from remote sensing imagery the required database primitives, and constructing detailed 3D feature models offers many challenging problems. Another problem is to simplify the terrain model by using fewer polygons without significant loss in the visual characteristics of the rendered imagery, thereby reducing the complexity of the terrain database and improving real-time rendering performance. In this paper we present two surface simplification algorithms designed for the purpose of constructing a terrain database that is optimized for driving simulation; one using a bottom-up, polygonal refinement approach, the other using a top-down, polygonal removal approach. These two algorithms are applied to terrain surfaces that include integrated, "stitched-in" road features, and are used to generate terrain surfaces of various levels of detail. We provide a discussion on the design of these two algorithms, some experimental results of applying the algorithms to real terrain surfaces, as well as the comparison of the two approaches on the factors of height error and the distance from the road.

Keywords: terrain, polygonal surface simplification, Delauney triangulation, level of detail

## 1. INTRODUCTION

Although it is not generally recognized, terrain database generation is often one of the most expensive and time-consuming tasks in the development of human-in-the-loop visual simulations. Much of the expense incurred in terrain database generation is due to the lack of automation throughout the data pipeline, creating the need for extensive "hand-modeling" and artistic license. Some examples of points in the pipeline that require extensive human intervention can be seen in Figure 1, which depicts a typical instance of a terrain database generation pipeline.

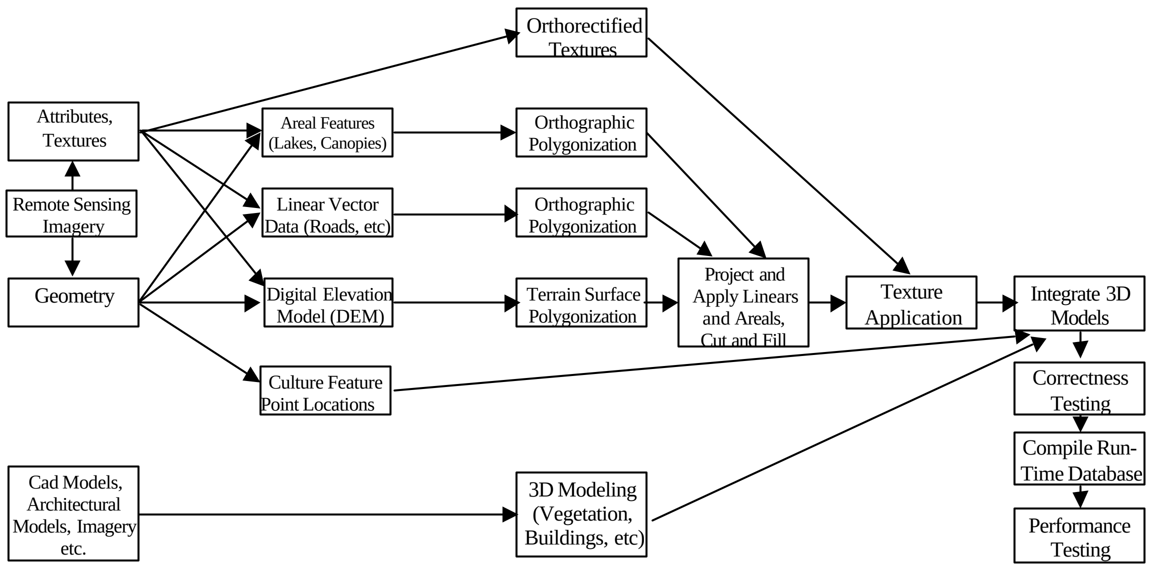

Orthorectified Textures
Attributes, Textures
Areal Features (Lakes, Canopies)
Orthographic Polygonization
Remote Sensing Imagery
Linear Vector Data (Roads, etc)
Orthographic Polygonization
Geometry
Digital Elevation Model (DEM)
Terrain Surface Polygonization
Project and Apply Linears and Areals, Cut and Fill
Texture Application
Integrate 3D Models
Culture Feature Point Locations
Correctness Testing
Compile Run-Time Database
Cad Models, Architectural Models, Imagery etc.
3D Modeling (Vegetation, Buildings, etc)
Performance Testing

**FIGURE 1. EXAMPLE OF THE TERRAIN DATABASE GENERATION PROCESS.**

The process of extracting from remote sensing imagery the required database primitives, such as geospecific textures, geometry, and attributes (e.g. surface material types), may be semi-automated using modern image processing techniques but almost always requires a large degree of intervention in the current state of the art. Although automated extraction of buildings and other 3D features from imagery has been an active area of open research for several years, this task also still requires extensive interactivity at present. Construction of detailed 3D models is a process that is well known as being labor intensive, requiring a great deal of domain-specific knowledge and artistry. Another set of hand-modeling (or at best, semi-automated) operations generally referred to as "cut and fill" operations are applied in the process of integrating into the terrain areal and linear features such as roads, rivers and lakes. These operations refer to local alterations and repolygonizations to the terrain surface necessary to prevent erroneous conditions such as rivers flowing uphill, lake surfaces higher than the surrounding terrain, and roads with incorrect grades, improper banks, or overly steep shoulders. Cut and fill operations are usually necessary in the process of integrating 3D models into the terrain, as well.

In various applications of computer graphics, polygon meshes are used to represent the geometric model of objects, including terrain model. A common form for representing terrain surfaces is that of the triangulated irregular network (TIN), which allows for inhomogeneous sampling of terrain in the plane, and a polygonal representation via linear triangles. With the increasing complexity of the terrain model, tasks such as spatial searching and real-time rendering become more time-consuming, leading to unacceptable performance in real-time applications such as a driving simulator. One peculiar requirement of a terrain database designed for driving simulators is that of non-homogeneous fidelity. High scene complexity and finely resolved attributes are required only in the vicinity of roads and other transportation routes. As a result, the application of procedures and algorithms designed for the generation of terrain databases suited for other types of simulation, when applied to terrain database generation for driving simulators, may lead to frequent redesign and recompilation of the terrain database. One point in the terrain database generation process where there is an apparent lack of algorithms suited to driving simulation is the terrain polygonization step. As an example, one terrain polygonization algorithm widely used in military simulation applications is the Iterative Triangulated Irregular Network (ITIN) algorithm developed by Polis and McKeown [1]. This algorithm is one of a large class of algorithms that attempts to minimize elevation error in a triangulated terrain surface based on some norm. The basic ITIN algorithm is as follows:

- Start the triangulation with 4 points at the corners of the terrain bounding box.
- Calculate absolute elevation error for each planimetric point of the underlying digital elevation model.
- Take the point that has the greatest error and accept it as a new vertex in the triangulation.
- Re-triangulate the surface using a Delauney triangulation.
- Update the absolute elevation error value for the remaining non-vertex points in the digital elevation model.
- Repeat the steps 3 and 4 until a stopping criterion is satisfied.

Using the ITIN, it's possible to generate several levels of detail by storing the successive triangulations, but still there are several problems with the ITIN algorithm when applied to terrain database generation for driving simulation:

- In its basic form, the ITIN does not allow for direct incorporation of road polygons into the terrain surface.
- Use of the ITIN results in a spatial distribution of triangles that is not well suited for driving simulations. The triangle density is too high in regions far from transportation routes, and too low in regions near the routes. The result is essentially a degradation in rendering performance.
- The Delauney triangulation employed in the ITIN results in unnecessary error. A data-driven approach, with constraints on sliver polygons, is worthy of investigation as an alternative.
- The ITIN does not allow for edge constraints, which is necessary for preserving critical linear features associated with roads.
- The basic ITIN does not address the topological and logical attributions that are necessary for driving simulation.

There are many simplification algorithms for polygon mesh models that have been proposed in the last few years. We can classify them into four categories according to the underlying mechanism they use to reduce the number of polygons in the scene model. Nearly every simplification technique uses some variation or combination of the four basic polygonalization mechanisms:

- *Sampling* schemes begin by sampling the geometry of the initial model. The algorithm proceeds to create a polygonal simplification that closely matches the sampled data. For an example, see [2].

- *Adaptive* subdivision approaches create a very simple polygonal approximation of the scene, called the base model. The base model consists of elementary shapes that lend themselves to recursive subdivision until the result surface lies with in some user-specified threshold of the original surface. For an example, see [3].

- *Decimation* techniques iteratively remove vertices or edges or faces from the mesh, retriangulating the resulting "hole" after each step. For an example, see [4,5,6].
- *Vertex merging* schemes operate by merging two or more vertices of a triangulated model together into a single vertex, which can in turn be merged with other vertices. These algorithms use a limited vertex merging called an edge collapse, in which only the two vertices sharing an edge are collapsed in each operation. For an example, see [7,8,9].

Using this classification of the simplification mechanisms, we can see there are two general simplification methodologies that can be used for the terrain surface: polygonal refinement, a bottom-up approach, and polygonal removal, a top-down approach. Both the refinement and the removal approaches share a very important characteristic: they seek to derive an approximation through a transformation of some initial surface. A refinement algorithm is an iterative algorithm that begins with an initial coarse approximation and adds elements at each step. As opposed to refinement methods, a removal algorithm begins with the most detailed surface and iteratively removes elements at each step.

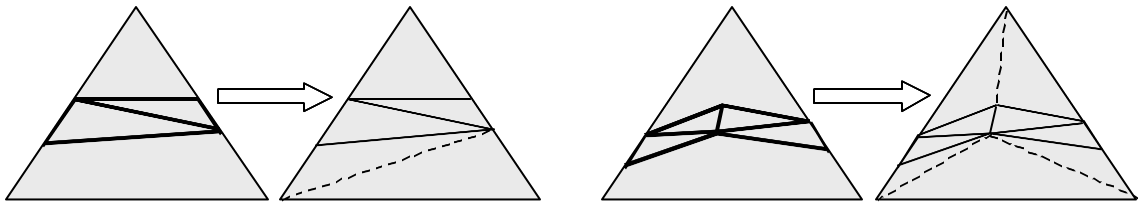
By investigating the problems appearing in the generation of terrain databases for driving simulation, in this paper we explain two surface simplification algorithms using the above two different approaches: refinement and removal, respectively based on a modified ITIN and the edge collapse. The overall goal is produce different level of detail approximations of a terrain model for driving simulation. In section 2 we describe the algorithms and related preprocesses. In section 3 the experimental data are provided to illustrate the algorithm results, and to compare the two algorithms by the factors of height error and the distance away from the road.

## 2. ALGORITHMS

Before introducing the specific algorithm, we must first give a description of the data format for the terrain model and a necessary preprocess on it. In practice, we constructed the initial terrain model using the Multigen Modeler and output the result in the Openflight format. For the convenient handling of the terrain data in the ZCAP[*] software developed by IST, the terrain model is represented by four related data: vertex data, triangle data, header data and feature data. Vertex data gives the coordinate of the sample points on the terrain model, and triangle data contains the connective relationship of the sample points contained in the vertex data, while the header and feature data provide the extra summary information about the terrain model.

We use a format conversion tool in ZCAP to convert the terrain model data in Openflight format into the ZCAP native format that uses the above described four data files to represent the terrain model. At this point in the database generation process, the road polygons are distinguished from the terrain polygons by their feature identification code, with road triangles overlapping over the terrain triangles rather than being built into the terrain surface. It is at this point we tessellate the road triangles into the terrain model surface by using following method, illustrated by Figure 2:

With reference to the road integration algorithm presented here, the general case includes more complicated ways that road triangles may overlap on the terrain triangles. In our present algorithm we consider the two of the simplest cases to clarify the explanation of the algorithms workings. By using this preprocessing step, we obtain the four data files on the terrain model with the road triangles "stitched-in", and an additional road data file that explicitly lists the road vertices.



[*] ZCAP is a suite of software tools developed to address terrain database interoperability. It provides capabilities for terrain and culture correlation testing, terrain and image registration, database analysis and terrain database visualization. The newest ZCAP release ZCAP4.1 available at http://www.usl.ist.ucf.edu/groups/vtb/

( a )
two road triangles overlapping on one terrain triangle.

( b )
four road triangles overlapping on one terrain triangle with one corner.

**Figure 2. tessellating operation**
triangle with bold line is road triangle; triangle with non-bold line is terrain triangle

## 2.1. Edge-Collapse-Based Terrain Simplification Algorithm

For the polygon removal methodology, the edge-collapse based simplification scheme starts with the highest-detailed model and then iteratively contracts the edges making up the terrain model on each pass. The steps are as follows:

- Read the five data files (.header, .vertex, .triangles, .road, .features), prepared by above preprocess program.
- Initialize the elements ( vertex, triangle, edge) of the terrain data.
- Choose the edges that are to be contracted.
- Repeat step 3 until the desired number of triangles is reached.

### 2.1.1. Initialization

For convenience in implementing the simplification step that follows, in this initial step we need to obtain some required parameters describing the data elements:

- Since this algorithm is based on edge collapse, by processing the given vertex and triangle data we form the edge information and use both the triangle and edge data to indicate the connection between different points.
- We calculate the normal of all the triangles, the dihedral angles of all the edges, and the minimal 2D distance from the road for every point, all of which is information used for the selection of candidate edge to be contracted (removed).
- W flag the vertex, edge and triangle that are on the road and on the border. The flag provides additional required information used to retain important terrain characteristics during the terrain simplification process.

### 2.1.2. Edge Contract

Generally the purpose of forming levels of detail (LODs) is to reduce the complexity of the representation, while retaining the maximum accuracy as measured by some relevant metric. Some distinctive features of polygonal terrain that can be used for simplification include [3]:

- *Near-Planar areas* that can be identified by inspecting the normal of adjacent polygons. The adjacent polygon can then be merged to form bigger ones.
- *Sharp edges* that can be found by using the angle between the normal of adjacent faces. They can be simplified by merging connected edges that are nearly collinear.
- *Pointed edges*, such as the tip of a pyramid, that have high visibility (i.e. have a relatively large probability of appearing on the silhouette of the terrain from a random viewpoint). They can be detected by measuring the local curvature around the vertex. These can be simplified by pushing neighboring vertices towards the tip of spike.
-

Because of the particular nature of our terrain that includes integrated road features, and by primarily dealing with the characteristics of planar area and sharp edges, we decide to use not only the dihedral angle of the edge, but also the distance of the edge as a criteria for the candidate edge selection. Searching all the edges by their weight
:

$$w_i = ( Cos?_i + 1 ) / d_i \; ( i = 1, 2, \ldots, e),$$

where $w_i$ is the weight of edge $i$, $Cos?_i$ is the Cosine of the dihedral angle of the edge $i$, $d_i$ is the distance that the edge is away from the road and $e$ is the number of edges.

Choosing the edge that has the least weight and at the same time is not located on the road or border, we perform the edge contract operation illustrated by the Figure 3:

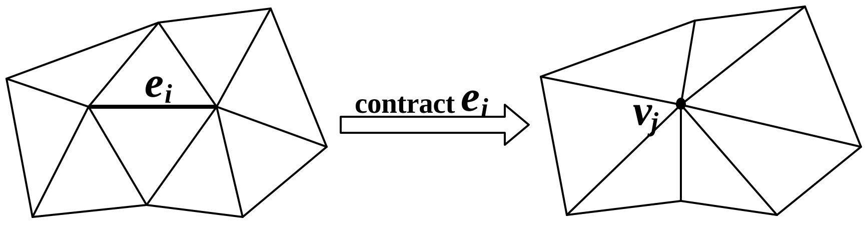


**Figure 3. edge contract operation**

In our polygon removal method we contract the edge $e_i$ to vertex $v_j$, which is the midpoint of edge $e_i$. As the simplification proceeds and the simplification depth increases, we may find some edges adjacent to the road or the outside border that needs to be contracted. In these cases we collapse the edges to the vertex on the road or the vertex on the border to retain the road environment and the border shape characteristics of the terrain database. Following the edge collapse procedure the changed parameters of the elements in the terrain model around the collapse area are updated, which completes the overall collapse operation. This process continues until the desired number of triangles in the terrain is reached, at which point we end the simplification procedure and get the level of detail terrain model that we need.

## 2.2. Delauney-based Integrated Feature and Terrain Triangulation Algorithm

Our refinement method is a multi-pass algorithm that begins with a minimal initial approximation; on each pass one or more points are inserted into the terrain model. The procedure repeats either until the error used for controlling becomes less than desired error, or until the desired number of vertices is reached. Our Delauney-based Integrated Feature and Terrain Triangulation (D-IFATT) algorithm is a refinement methodology. Before introducing this algorithm, we need to further describe the input terrain data. Since refinement methods begin with a simple approximation, as opposed to the polygon removal approach that starts with the highest-detail terrain model, we will require two terrain models: the highest-detail terrain model in the form of a regular triangulation created from the source digital elevation model (DEM), and a low-detail terrain model, both with integrated roads. In our work we created these two files using the Multigen Modeler to create the polygonal terrain surfaces, which were then refined to integrate the roads by the preprocess program discussed above. At this point we apply the D-IFATT algorithm to the data as follows:

- Delete the road vertices in the highest-detailed terrain.
- Check which of the remaining vertices in the highest detailed terrain are within the boundary of the lowest detailed terrain, excluding vertices on the boundary. Insert these vertices into the vertex list of the lowest-detailed terrain.
- Build the expanded vertex list of candidate vertices for Delauney Triangulation, which involves the vertices in lowest-detailed terrain, including boundary vertices and non-boundary vertices, and the inserted vertices from highest detailed terrain, extracted in step2.
- Compute the distance to the road for each of the vertices in the candidate vertices list, and store it as another item in the vertex structure in addition to coordinate x, y, z.
- Execute Delauney Triangulation to insert (Figure 4 ) the desired number of vertices to form required level of detail terrain model, according to order of the error of each vertex, which is computed by absolute height error divided by its distance to the road.

In our algorithm, we use the circle criterion to construct the Delauney Triangulation, which works by choosing the appropriate diagonal of each quadrilateral formed by pairs of triangles which share an edge. The principle of the circle criterion is: choose an arbitrary diagonal, and construct the circumcircle of one of the two triangles in the quadrilateral; let the circumscribed triangle be $t_i$ and the opposite triangle be $t_j$. Consider the exclusive vertex $v_k$ of $t_j$, defined as the vertex not associated with the shared edge. If $v_k$ lies within the circumcircle, switch the diagonals. Repeat for every quadrilateral in the triangulation, until an entire pass yields no diagonal switch ( Figure 5 ).

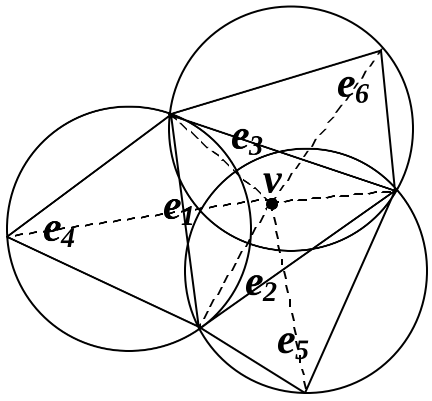


**Figure 4. One Delauney Triangulation Operation for a point into one terrain triangle**
Triangle with bold line is terrain triangle; $v$ is the point to be inserted into. In this situation: $e_2$ switch to $e_5$, $e_3$ switch to $e_5$, while $e_1$ needn't switch to $e_4$.

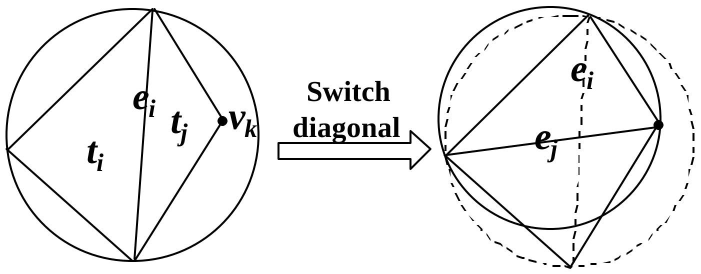


**Figure 5. One diagonal switching operation for Delauney Triangulation**
In this situation: $v_k$ locates into $t_i$'s circumcircle, $e_i$ switch to $e_j$, if not locates into, no switch needed.

## 3. EXPERIMENT

We implemented the Edge-Collapse-Based Terrain Simplification algorithm in the C language on Pentium 400Hz, with 128M RAM, using the Linux Redhat 6.0 operating system, while the D-IFATT algorithm was implemented in C language on SGI IRIX workstation with 128M RAM. The terrain model we used for these two simplification algorithms, a piece of Wyoming area terrain, is created using the Multigen Modeler. Its size: for the highest-detail terrain model consist of 1829 vertices and 955 triangles; for the lowest-detail terrain model consist of 205 vertices and 136 triangles. Figure 6(b-e) gives the simplification results based on the Edge-Collapse-Based Terrain Simplification algorithm with the initial highest-detail terrain model, while Figure 7(b-e) provides the results based on the D-IFATT algorithm with the initial lowest-detail terrain model Figure 6a.

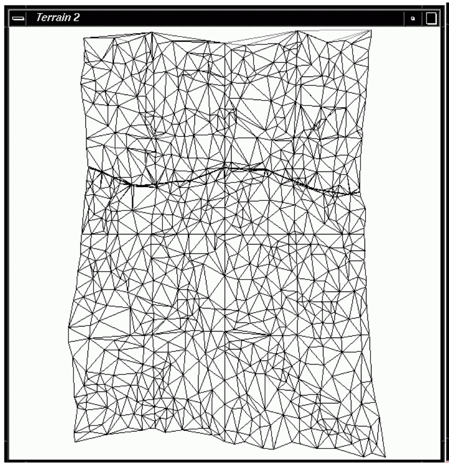


**Figure 6a. Original highest-detail terrain model with built-in road**
The total number of vertices = 955
The total number of triangle = 1829

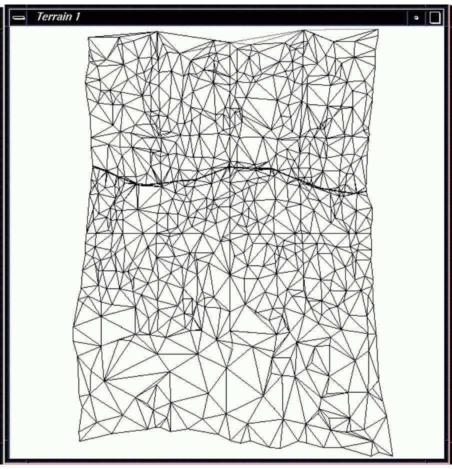


**Figure 6b. First LOD terrain model created by Edge-Collapse-Based Terrain Simplification Algorithm**
The total number of vertices = 790
The total number of triangle = 1499
Maximal error = 0.012704
Average error = 0.017874

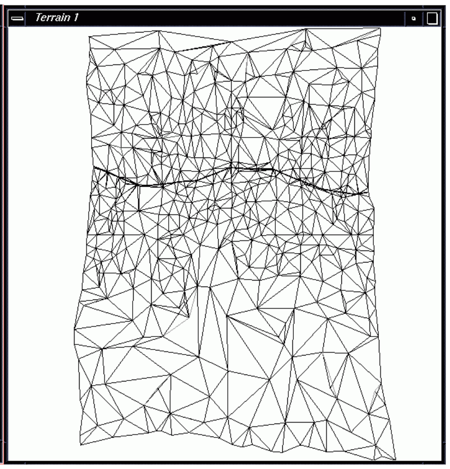


**Figure 6c. Second LOD terrain model created by Edge-Collapse-Based Terrain Simplification Algorithm**
The total number of vertices = 640
The total number of triangle = 1199
Maximal error = 0.023878
Average error = 0.018333

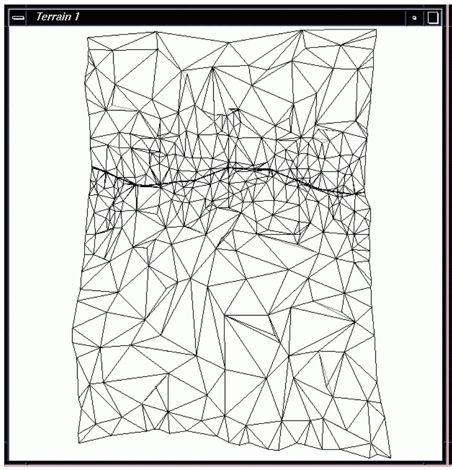


**Figure 6d. Third LOD terrain model created by Edge-Collapse-Based Terrain Simplification Algorithm**
The total number of vertices = 490
The total number of triangle = 890
Maximal error = 0.042482
Average error = 0.020689

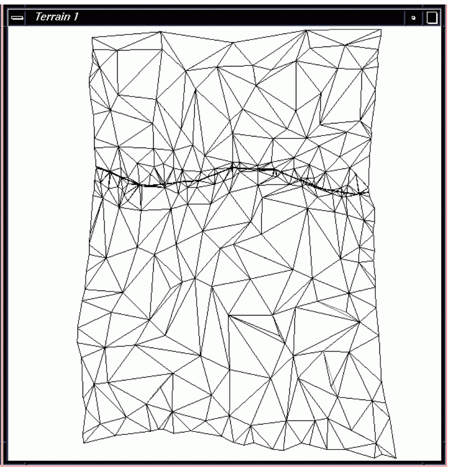


**Figure 6e. Fourth LOD terrain model created by Edge-Collapse-Based Terrain Simplification Algorithm**
The total number of vertices = 365
The total number of triangle = 649
Maximal error = 0.099428
Average error = 0.024463

Since our role is creating the static different LOD terrain model to construct the terrain data base offline, there is no compelling requirement on execution time of the algorithm. For Edge-Collapse-Based Terrain Simplification algorithm, a large portion of the time cost comes from forming the edge list and selecting the candidate edges to be collapsed, with time complexities of *O(te)* and *O(e(v+t))*[*] respectively. In the removal algorithm, the basic data structures are three linked lists: the vertex list, triangle list and edge list. With every edge contract operation, we delete one vertex, two edges and one triangle, but we also add a new vertex node and two new edges nodes into the vertex list and the edge list. The resulting space complexity is *O(v+e+t+[e/2]*3)*. For the D-IFATT algorithm, time is largely spent on checking whether points in high-detailed terrain model lie in the boundary of the lowest-detailed terrain model, whose time complexity is *O(vt)*. This time obviously can be improved by choosing a more efficient spatial organization, such as a simple quadtree. In the refinement algorithm, the process of selecting the candidate inserted point and applying Delauney Triangulation on current terrain model is also time consuming, with time complexity is *O(vt)*. In the refinement algorithm, we use an expanded vertex list and a triangle list, whose space complexity is *O(v+t)*. According to the time comp lexity and the space complexity, the refinement approach has a slight advantage in the present implementation.

The goal of both algorithms is *Min-e* based, which means when given an expected size for the approximated terrain model, the objective is to minimize the error or difference between the original and the resulting terrain model. In both algorithms, we only minimize the error for each simplification primitive operation, not for entire simplification process. In this kind of process, it is important to note that minimizing the error at each step does not guarantee the least error for the final LOD terrain model. Based on the specific of our terrain model, we agree on a simple error evaluation function:

$$\textit{for each vertex } v_i : error_i = ?h_i = | z_{i\text{-}new} - z_{i\text{-}old} | \textit{, then the average error} = S(?h_i / d_i)/v \quad (i = 1,2,\ldots,v)$$

The vertices set *V* sampled to compute the error are those original vertices in the highest detail terrain model. $z_{i\text{-}old}$ is the z coordinate value of vertex $v_i$ when it's in original highest-detailed terrain model, while $z_{i\text{-}new}$ is the z coordinate value of vertex $v_i$ when it's in resulting LOD terrain model; $d_i$ is the distance that the vertex $v_i$ is away from the road and *v* is the number of vertices set *V*. The average error and maximal error of the all vertex error for each LOD terrain model is also presented in Figure5 and Figure 6.

Unlike the Delauney Triangulation that maximizes the sum of the minimum angles of the triangle to prevent slim triangles in terrain model, the Edge-Collapse-Based Simplification algorithm is a data-driven approach that potentially has the problem

[*] *v,t,e* is the number of vertices, edges and triangles in the terrain model.

of distorting the terrain through the production of many sliver triangles. We prevent the overproduction of slivers by controlling the number of triangles sharing the same vertex. This controlling measure limits the simplification degree, with the result that the lowest LOD we can generate is simplified by approximately 60%; we are prevented from obtaining larger simplification ratios, say in the vicinity of 90%. In contrast, since the refinement approach starts from the lowest-detailed terrain model, the simplification degree is not a problem, and its interpolation procedures perform best on triangulation which are nearly equiangular. However, the refinement approach suffers from at least one additional disadvantage not shared by the edge-collapse approach: since it employs the Delauney triangulation instead of a data-driven approach, this method may not be able to capture the exact geometry of the original model, because the edge-swapping destroys the original topology of the triangulation. Thus, sharp features such as valleys and ridges may be lost.

Bearing in mind the above defects, the comprehensive height error by distance for the D-IFATT algorithm performs better than for the Edge-Collapse-Based Simplification algorithm, with the average and maximal error shown in the final experiments.

We reckon the reason is that the D-IFATT procedure itself is using the height error as its vertex insertion criterion. If we optimize the criterion for choosing the candidate contraction edge in Edge-Collapse-Based Simplification algorithm and control the vertex location of contracted edge, we estimate a lowest-detailed terrain model with better quality will be achieved, which can be used for the D-IFATT algorithm to get the terrain LOD of more uniform triangles.

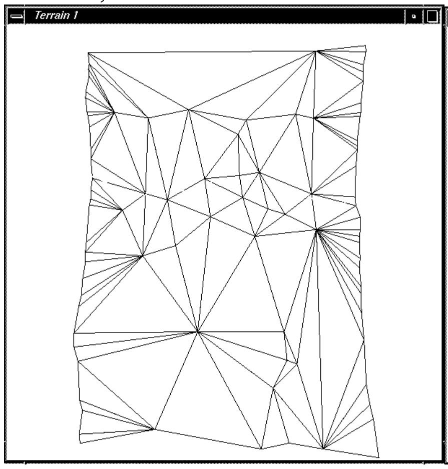


**Figure 7a. Original lowest-detail terrain model with built-in road**
The total number of vertices = 136
The total number of triangle = 205

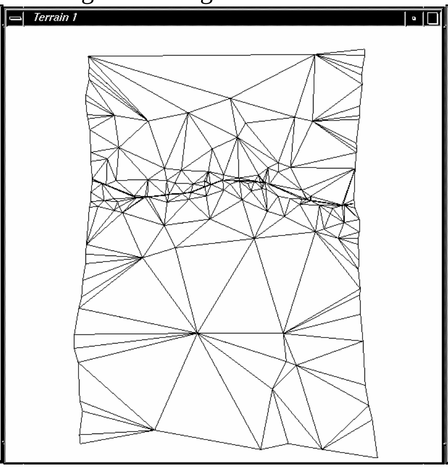


**Figure 7b. First LOD terrain model created by D-IFATT Simplification Algorithm**
The total number of vertices = 184
The total number of triangle = 301
Maximal error = 0.114426
Average error = 0.026095

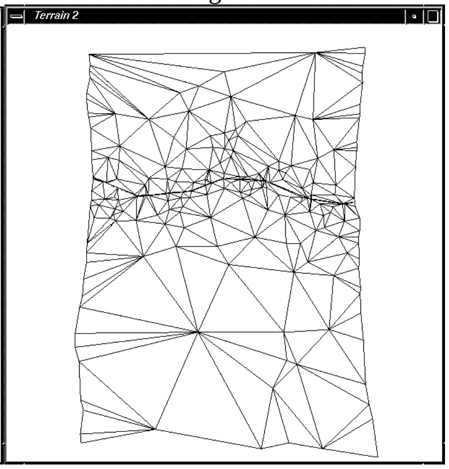
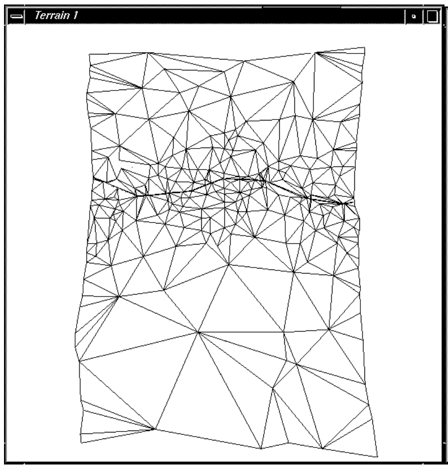


**Figure 7c. Second LOD terrain model created by D-IFATT Simplification Algorithm**
The total number of vertices = 234
The total number of triangle = 401
Maximal error = 0.067909
Average error = 0.019152



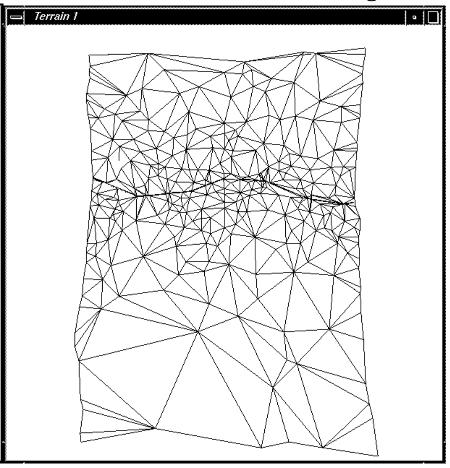


**Figure 7d. Third LOD terrain model created by D-IFATT Simplification Algorithm**
The total number of vertices = 334
The total number of triangle = 601
Maximal error = 0.037218
Average error = 0.011698

**Figure 7e. Fourth LOD terrain model created by D-IFATT Simplification Algorithm**
The total number of vertices = 384
The total number of triangle = 701
Maximal error = 0.026923
Average error = 0.010154

## 4. CONCLUSION

In this paper we propose two algorithms to simplify terrain models containing integrated features: one is Edge-Collapse-Based Terrain Simplification algorithm and the other is the D-IFATT algorithm. The two algorithms can, to a good degree, achieve the goal of trading higher detail in the vicinity of roads for lower detail away from the roads. Experimental results show that the D-IFATT does better on producing uniform terrain, while Edge-Collapse-Based simplification has the advantage of better capturing the geometry of the terrain model. For the future work, we plan further investigations in quality control on the model simplification procedure.

# II. Software design for extracting linear feature data from aerial/satellite images and generating terrain databases with integrated features

For driving simulation, it is critical to have accurate geospecific source data for transportation networks. To generate terrain databases with integrated features such as roads and trails, the precondition is to obtain reliable feature data. Unfortunately, for many areas, it is often the case that the feature data are either not available or not accurate enough. For example, there are no accurate road feature data available for UCF area. In Fig.8, it is shown a database for UCF area that was built by using the feature data (DLG format) from United States Geological Survey (USGS). It can be seen that there are quite a few incorrect places in the database. The roads at the top-left part of the database do not exist in reality. The circle at the center of the UCF campus and most of Gemini Blvd are missed. It can be seen that many other errors exist as well. It is worthwhile to point out that some of these errors still appear in some newest Orlando street maps published in 1999.

For our project, it is urgent to obtain the related feature data for UCF and Research Park area. There are aerial and satellite images available for this area. But one needs to extract the road information from the aerial and satellite images. To solve this problem, we have programmed our own software to extract the roads and trails from the aerial and satellite images that is designed to form a file that can be used to generate terrain with integrated linear features. Our own software can extract any linear feature such as roads, trails, rivers and rows of trees etc. from satellite and/or aerial images for any area of interest. Our software has following properties:

1) The original input images can be aerial or satellite images.

2) Our own tool also accepts maps printed on the papers as input image files supposing the paper maps can be scanned beforehand.

3) The roads and trails are extracted out by clicking the mouse at the corresponding region in the image.

4) There is no limitation for the number of roads being extracted.

5) The output feature data generated by our own tool can be used together with terrain elevation data to generate 3 dimensional terrain databases with integrated features. In other words, the roads are built in an elevation field. It is also possible to build two dimensional databases, which is a special case, by using our tool.

6) Our generated feature files for roads and trails can be used together with terrain elevation databases (for example, DED files) to build terrain databases with integrated features such as roads and rivers at various levels of details (LODs).

7) By specifying the parameters in the output file, the widths and surface materials such as concrete, asphalt etc. for different roads can be specified individually.

Fig.9 shows the database for a part of the UCF campus built by extracting roads from an aerial image using our own software. For simplicity's sake, only some of the roads are included in the figure.

Our tool to extract linear features from aerial/satellite images is easy to use. The procedure to extract the linear features is as follows:

1) Convert the aerial/satellite images into "xpm" format by using some graphics format converter.

2) Type the name of the executable file "FLTKpick_SatelliteImage" and a window will open.

3) Click "load" and pressing "s" key simultaneously.

4) Choose the input "xpm" aerial/satellite image file by clicking the name and pressing "s" key simultaneously.

5) Click "OK" while pressing "s" key.

6) Click a reference point at the southwest (bottom-left) corner of the image.

7) Click the second reference point at the northeast (top-right) corner of the image.

8) Click the first point of the road, then click the second point of the road and so on, click the last point and press "space bar" simultaneously to end the input of that road. One may record the coordinates of the last point since later this information might be needed.

9) Use the same way to pick out other roads. For each clicking, the coordinates of the point are recorded in a file called "FLTKsatelliteOUT". The coordinates are also displayed on the screen at the same time.

10) After having picked out all the roads needed, quit the program.

Another program "satellite2dlg" converts the obtained coordinates on the image to UTM coordinates and forms a data file that can be used to form a file which format is similar to USGS's digital line graphs (DLG) file. To use the software "satellite2dlg", simply type:

```
satellite2dlg FLTKsatelliteOUT File1 File2
```

The coordinates for the points of each road in the image are converted to UTM coordinates and the UTM coordinates are stored in File1. The coordinates for the points of each road are also recorded in File2. However, it is optional to include File2.

It is also required to decide the longitudes and latitudes for the reference points. This information can be obtained from available geospecific data for the specific area. A file that has the similar format as USGS's DLG format can be formed by using File1 after several geological parameters for the specific area have been decided. The generated feature data file is further converted to a file in DFD format that can be accepted by 3D modeling tool "MultiGen". With this data, we are able to build 3D terrain databases containing both linear features such as roads and elevation information.

We have solved quite a few problems during the implementation of the software. These problems include how to get coordinates from aerial/satellite image by clicking the mouse, how to convert the coordinates on images to UTM coordinates, how to convert the data to MultiGen acceptable image format, and how to decide quite a few geographic parameters involved in the transitional file.

We are going to use four- or six- parameter transformations to further improve the accuracy. At moment, we use linear interpolation method to convert the coordinates of the image into UTM coordinates. The accuracy of linear interpolation method is acceptable for a small area such as UCF area, but the error may increase for a larger area.

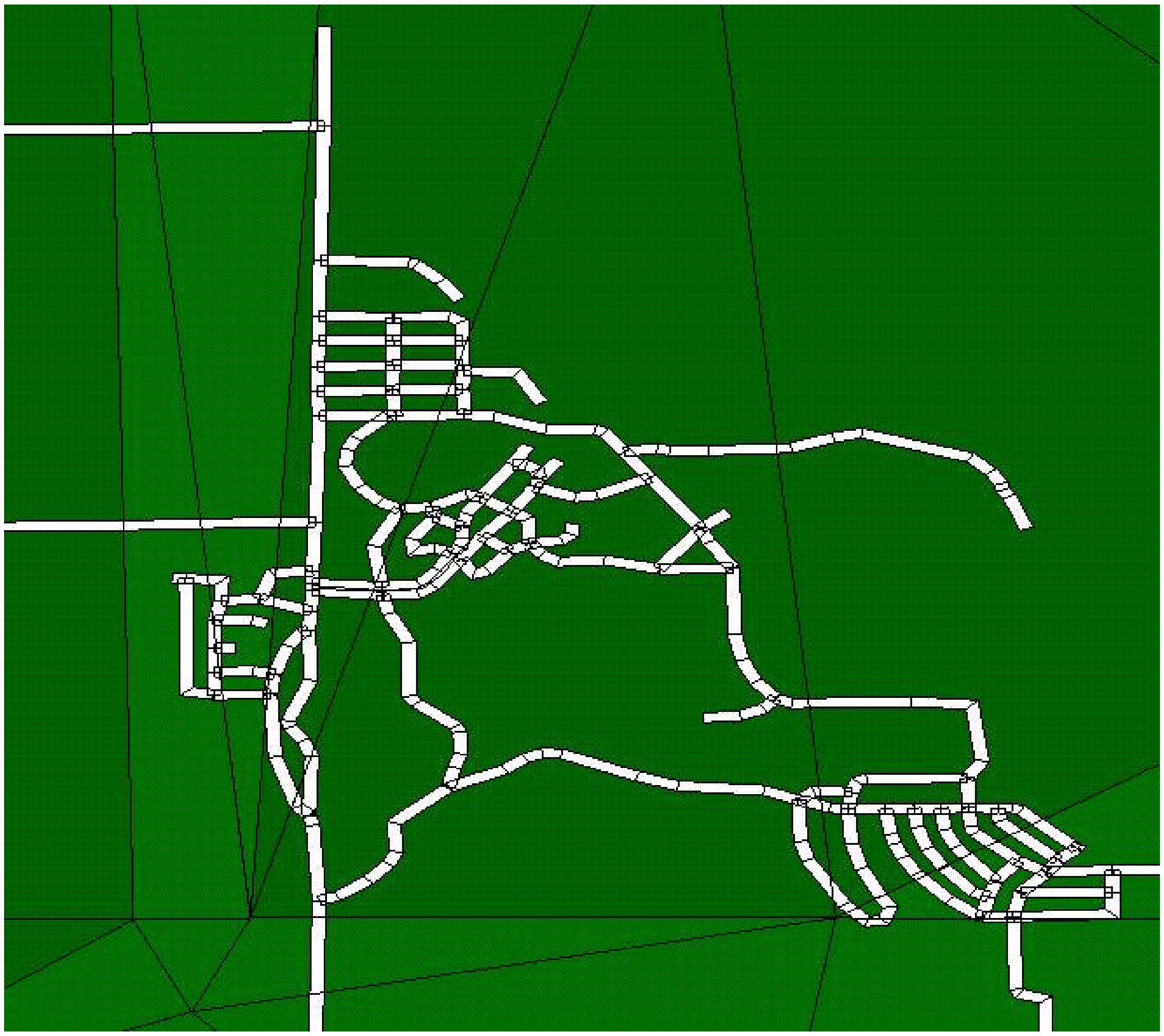


Fig.8 The database for the area of UCF campus built by using USGS's DLG feature data. Note that there exist quite a few errors. For example, the roads at the top-left part do not exist in reality, the circle at the center of the campus and part of Gemini Blvd. are missed and so on.

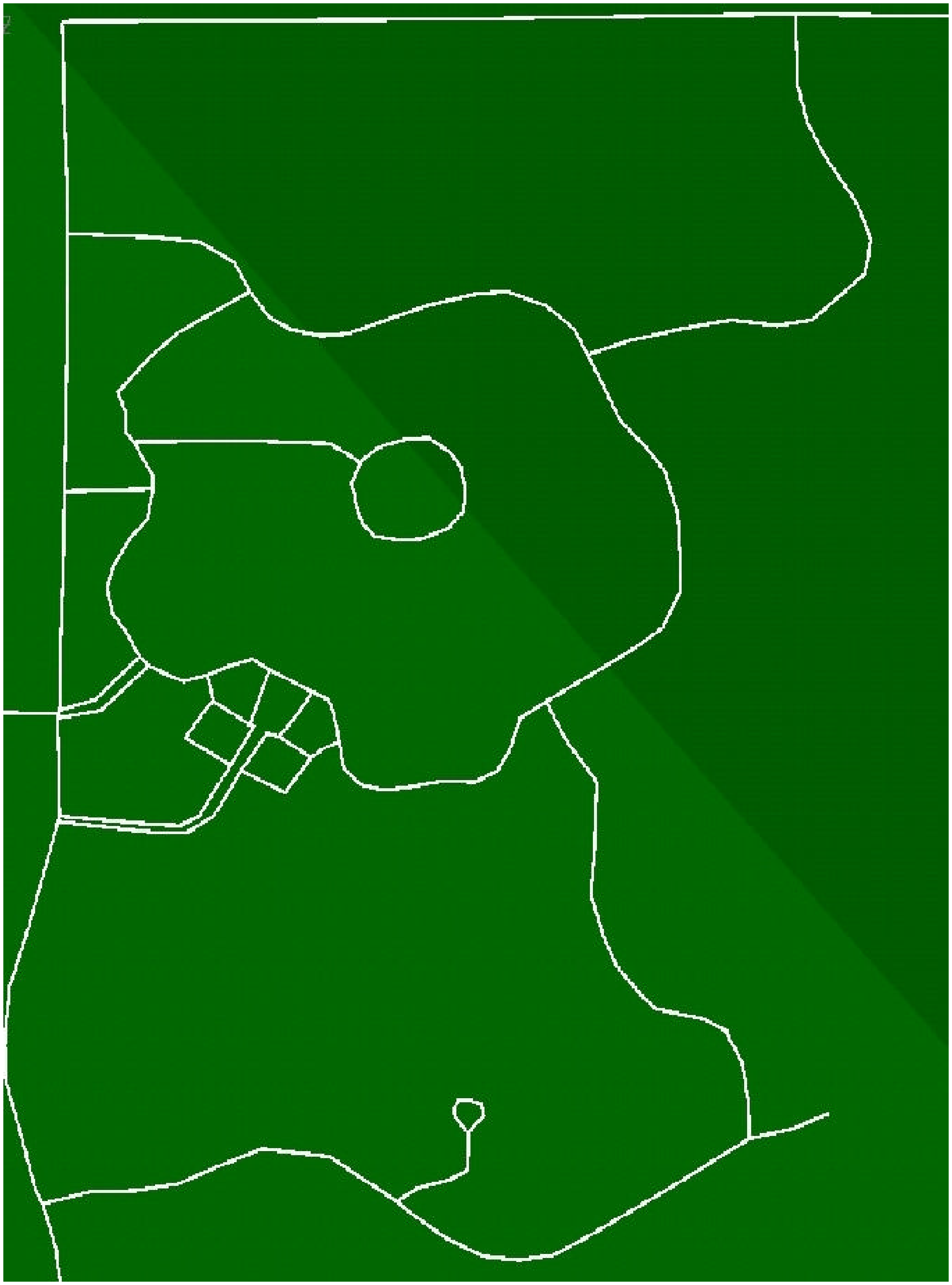


Fig.9 The database for the area of UCF campus. All the roads were obtained from aerial image by using our own software. Note that only some roads in the area are included and the present terrain is at the very low level of detail (two triangles).

## III. Building of the database for the Research Park of Orlando

We have built the database for Research Park area using the "MultiGen" modeling tool. The database can be used as input to driving simulator for visual simulation of driving on the roads of Research Park Way and Progress Drive.

It is quite time-consuming to build such database. For the usage of driving simulation, the database should be built as real as possible. First, the 3D models were built for the buildings in the area. A digital aerial image was used as a reference and the shapes and locations of the buildings were derived according to the aerial image. For getting the right sizes of the related objects, we have measured the widths of Research Park Way and Progress Drive. The data for the estimated heights of the buildings were obtained from "Central Florida Research Park Office". Although it is possible for us to build roads on the terrain using the way mentioned in Section II of this report, for meeting the requirements of UCF driving simulator, the present database does not contain elevation information for the terrain.

About 20 models of the buildings have been generated. The similar amount of 3D models of the plates for these buildings and other models for traffic signs and trees have also been produced and input as external references to the main database. For each face of the models, the corresponding texture has been applied. In the preparation of these texture files, many photographs were taken. A few have been chosen to generate texture files. It is often the case that there were trees and shadows in front of the buildings. For getting a good view effect, we spent a lot of time to remove these trees and shadows in the photographs by using the tool "Photoshop". The processed photographs were saved in proper format and size and were applied to the corresponding faces of the buildings. Alignment of the textures was performed to fit contiguous textures on adjacent faces.

For efficient rendering of the driving simulator, two models have been prepared for each object (building or traffic sign). One model is at higher level of detail (LOD) and the other is at lower level of detail. The model at lower level of detail normally has no texture applied. The switch distances between the eye point and the object can be specified for each model individually.

The traffic signs and signals are important for driving simulation. We constructed models for various traffic signs such as stop signs and speed limit signs. In most cases, two models with different levels of detail for each sign have been constructed. There are four traffic lights in the area. In addition to building the models, we also used an animation technique to change traffic signals within a period of time. The color of the traffic lights on Research Park Way was chosen to be different from that on Technology Park Way to simulate the real circumstance.

To make the models of the trees, it is needed to keep the alpha component in the texture files to make the non-tree part of the texture image transparent. We have put a few trees in the database. However, the number of the trees planted in the database is limited to reduce the burden to the real-time performance of the driving simulator. The actual number of trees we used was determined empirically based on several trials using the driving simulator visual system. All traffic signs, traffic lights and trees have been imported into the database as external references.

The databases have been tested at UCF Driving Simulation Lab (UCFDSL) several times, and the databases were thereafter modified to fit the performing requirements of the UCF driving simulator. The database at the time of the first trial included only a few models of the buildings and two roads. Only the IST building and the roads were textured. The textures could not be displayed during that test. In the subsequent test, the database could be displayed by using "MultiGen" at UCFDSL. However, the textures could not be seen when the database was loaded to driving simulator. Later, we changed the sizes of all the texture files to make the dimension of every texture pattern (in x and y) to be a power of two.

For further reducing texture memory, we re-textured all the models over the course of several weeks. To do this, we first de-textured all the faces of the models. We reduced the sizes of the texture files to get rid of the extraneous portions of the digital imagery, and saved all the texture files again. Then, each face of the models was re-textured.

In a test at the end of February, we learned that the present driving simulator did not properly render a terrain that includes elevation differences. Our understanding is that the database should contain no elevation information to perform in an acceptable manner. Therefore, we removed part of the database that was built using terrain elevation data, and reconstructed the database correspondingly.

For solving the Z-buffer conflict problem, we first tried to put the beads of the road polygons beneath the ground polygons in the database hierarchy. The problem was apparently solved when the database was displayed using “Perfly” on IST’s SGI platforms. However, the problem remained when the database was tested at UCFDSL. To fit the requirements of the UCF Driving Simulator, we were required to carefully modify the database to avoid any overlaps of the polygons.

The database was tested again at the beginning of March, it was found that the database still did not provide acceptable performance when rendered by the driving simulator, due to the size of the individual textures that were being used. In addition to the buildings and roads, the database included a large number of external references, such as the signs of the companies, and traffic signs. Animating the traffic lights also increased the computational burden, affecting performance of the driving simulator, although this effect was found to be within acceptable limits.

To improve the performance characteristics of the database for UCF driving simulator, we prepared another simplified database. For that database, we shortened the LOD switch distances and deleted some models (most of the trees, for example). The design of the traffic lights also was changed to reduce the polygon budget. The two databases were tested at UCFDSL on March 24. The simplified database was processed successfully by UCF driving simulator, but test involving a new version of the higher detail database resulted in poor performance.

We performed further modifications to the high-detail database to improve performance. Specifically, we decreased the sizes of the texture files and re-textured some models (buildings, plates of the companies, traffic signs etc.). We also rebuilt models for some buildings and deleted some polygons representing the ground surface where no buildings were present. Also, the aerial image texture that had been used as reference was deleted. After these adjustments, we believe that the total texture burden was been nearly optimized to fit the processing capacity of the UCF driving simulator image generation system. This version of the database has been tested successfully on the UCF driving simulator, and is our final database to be delivered.

Some portions of our database for the Research Parkway area are shown in Figs. 10-12.

Based on our experiences, now we can:

- build similar databases for transportation networks for other areas.
- build databases for other purposes, such as rail or river transportation.
- build 3D terrain databases with integrated linear features such as roads, trails and rivers for specified region. Each road may have its own attribution (width and surface material).
- build 3D terrain databases with integrated linear features such as roads, trails and rivers for the same region at different levels of detail (LODs).

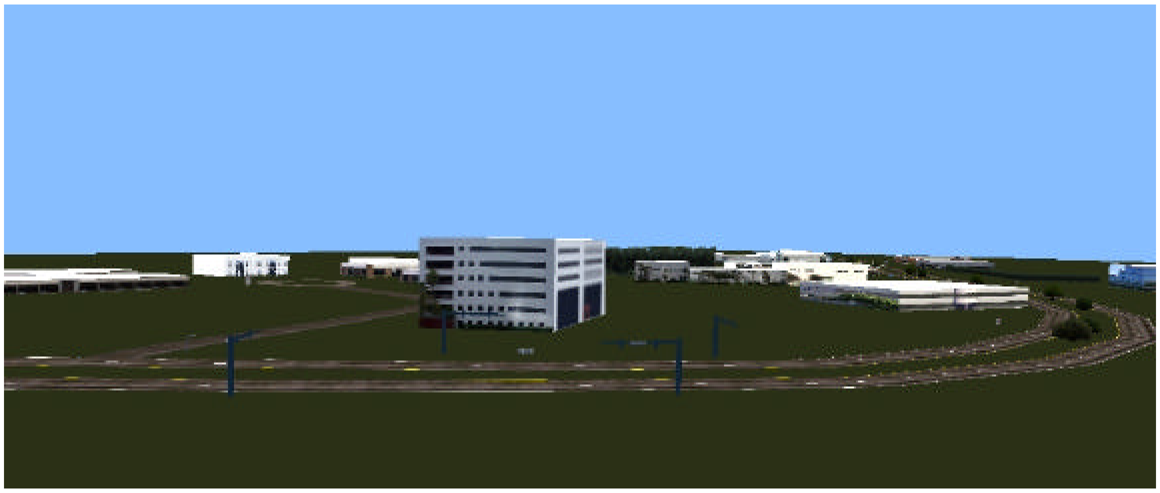


Fig.10 Part of the database of the Research Park. IST is on the left.

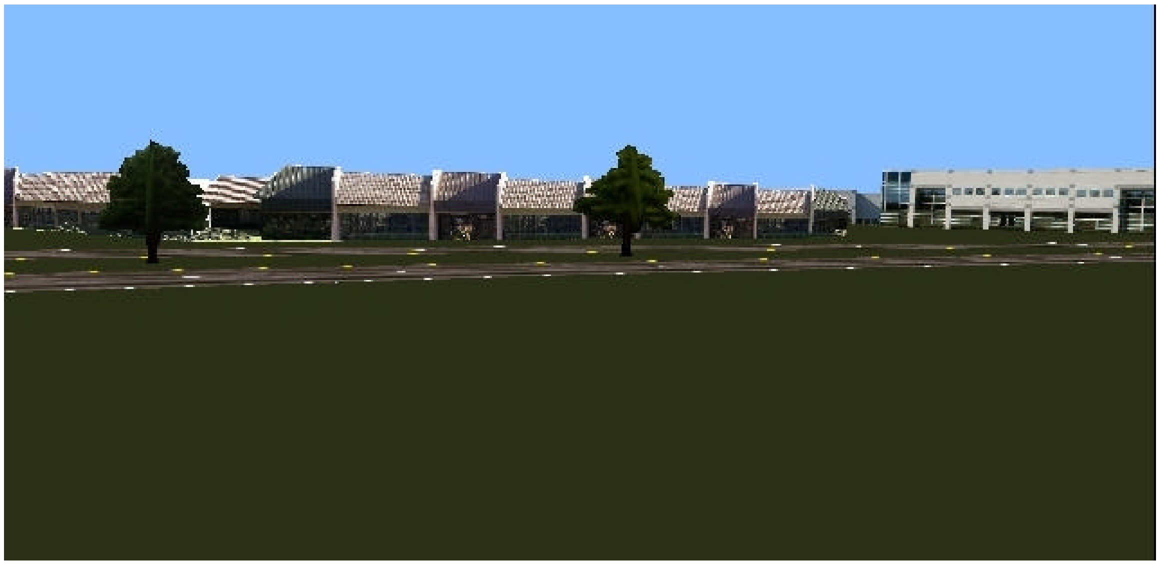


Fig. 11  Orlando Technology Center and University Technology Center.

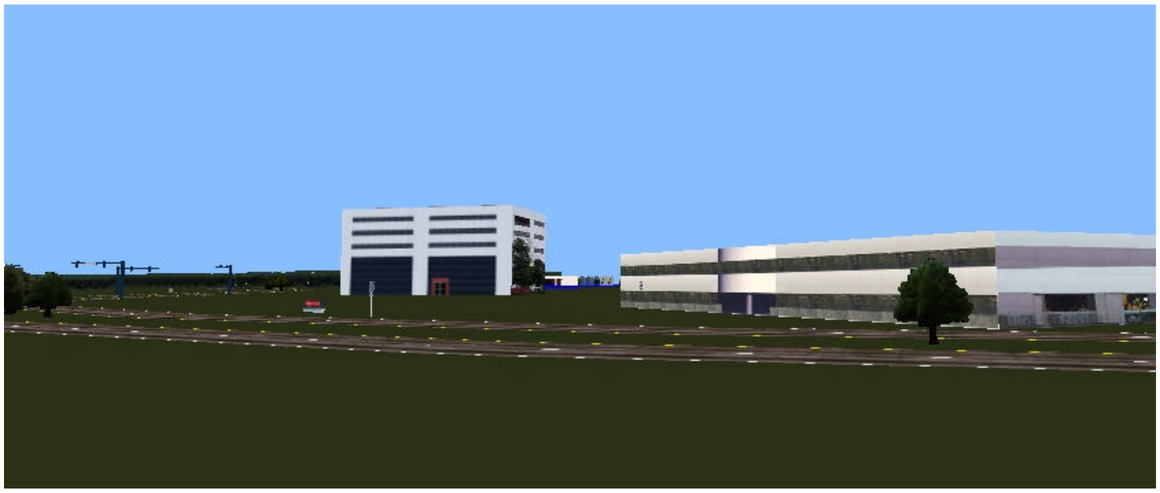


Fig. 12 The buildings of Siemens-Westinghouse and Raytheon Companies.

## OTHER MAJOR ACCOMPLISHMENT

1) The paper “Terrain modeling for driving simulation with the integrated feature and terrain triangulation (IFATT) algorithm“ based on the present work has been accepted by “SPIE’s 14th Annual International Symposium on Aerospace/Defense Sensing, simulation, and Controls”. This Symposium will be held in Orlando, USA on April 24-28, 2000. The paper has been presented in Session 6, Model Abstraction Applications on April 26, 2000.

2) During the present work, Guy A. Schiavone and Art Cortes supervised and guided the whole work. Guy A. Schiavone also designed the terrain surface simplification algorithms with integrated road feature. Following people also worked and had chance to be trained during the work with the financial support to this project. Jian-ping Gu designed the software to extract road feature from aerial images and was involved in the work of Research Park database (RPD) building. Ying Dai and Grace Yu worked on the terrain surface simplification algorithms with integrated road feature and programmed the codes to implement the terrain surface simplification using two algorithms. Walter Wimberly was involved in the work of RPD building and gave many good suggestions. Christian Buhl was involved in the work of RPD building and generated many fine textures and some models for the buildings. Ryan Leitch and Priya Krishnamachary were involved in the work of RPD building and design of terrain simplification algorithms, respectively.

3) With the support of the project, Ying Dai will finish her Master thesis based on the present work.

## ACKNOWLEDGEMENTS

The authors wish to thank the Florida Department of Transportation for their support in this work. We also would like to acknowledge the support and guidance provided by our colleagues in the Center for Advanced Transportation Systems Simulation (CATSS) at the University of Central Florida. We thank Xian-hong Zhang at UCFDSL for testing the databases.

## REFERENCE

1. M.F. Polis and D. M. McKeown, "Issues in Iterative TIN Generation to Support Large Scale Simulations", *Proc. AntoCarto 11*, pp. 267-277, 1993.
2. J. Rossignac and P. Borrel, "Multi-Resolution 3D Approximations for Rendering Complex Scenes", *IBM Research Division: T. J. Watson Research Center, IBM Technical Report: RC17697*, 1993.
3. M. DeHaemer Jr. and M. J. Zyda "Simplification of Objects Rendered by Polygonal Approximations", *Computer & Graphics 15(2)*, pp. 175-184, 1991.
4. W. Schroeder, J. Zarge and W.Lotenson "Decimation of Triangle Meshes", *Computer Graphics 26(2)*, pp. 65-70, 1992
5. F. Schroder and P. RoBbach "Managing the Complexity of Digital Terrain Models", *Computer & Graphics 18(6)*, pp. 775-783, 1994.
6. P. Hinker and C. Hansen "Geometric Optimization", *Proceedings of Visualization, Los Alamitos*, pp. 189-195, 1993.
7. B. Hamann, "A Data Reduction Scheme for Triangulated Surfaces", *Computer Aided Geometric Design 11(3)*, pp. 197-214, 1994.
8. H. Hoppe, "Mesh Optimization", *ACM SIGGRAPH'93 Proceeding 27*, pp. 19-26, 1993.
9. M. Garland and P. Heckbert "Surface Simplification Using Quadric Error Metrics", *ACM SIGGRAPH'97 Proceeding 31*, pp. 209-216, 1997.
10. P. Cignoni, C. Montani and R. Scopigno "A Comparison of Mesh Simplification Algorithm", *Computer & Graphics 22(1)*, pp. 37-54, 1998.
11. D. T. Lee and B. J. Shacter, "Two Algorithms for Constructing a Delauney Triangulation", *Int. J. Comp. Inf. Sci. 9(3)*, 1980.