3D VISUALIZATION AND QUERY TOOL FOR 3D CITY MODELS

R. Dogan[a], S. Dogan[b], M. O. Altan [c]

[a]Directorate of 12nd Region of Village Services, Samsun, Turkey - dogan_rukiye@hotmail.com
[b]Ondokuz Mayis University, Engineering Faculty, Dept. of Geodesy and Photogrammetry, 55 139, Kurupelit, Samsun, Turkey - (sedatdo@omu.edu.tr)
[c]ITU, Civil Engineering Faculty, 80626 Maslak Istanbul, Turkey - oaltan@itu.edu.tr

Commission III, WG III/7

KEY WORDS: 3D, city, modelling, GIS, databases, photorealism, query, analysis

ABSTRACT

For 3D city management visualization, query and the analysis of 3D data related to cities are very important tasks. In this paper, we present a tool (3D-City VAQS/ 3D-City Visualization Analyse and Query System) that can effectively visualize and query 3D city models. We use relational database model to represent geometry and topology of 3D city data. We organize 3D data in layers and themes and each theme has their own attribute tables. Our tool provides to query both geometry and attribute information of 3D data. We use V3D data structure and implement it in a relational database.

1. INTRODUCTION

Use of 3D spatial information for modern city management plays important role, especially for the tasks of spatial planning, communication, emergency management etc. These tasks can be realized easily and effectively by the 3D representations of the spatial and thematic attribute information together with in a geographic database. The generic idea of GISs is to incorporate geometric and semantic information in one system and to support analysis in both domains, (Zlatanova, 1999). A 3D GIS should satisfy various spatial operations. These operations can be summarized as follows: (Goodchild, 1987; Aronoff, 1995; Zlatonova, 1999).

- Access to semantic properties of one type object,
- Access to both semantic and location information,
- Operations which create object-pairs. (e.g. buildings in a given parcel that have one owner.)
- Operations analyse semantics of object pairs from one or more types.
- Operations which create a new type of object from existing objects.
- Retrieval operations. (e.g. what is the current information about a particular building.)
- Query operations retrieve data which satisfies some given conditions.
- Retrieval and query of semantic data.
- Integrated analysis of spatial and semantic data, (classification, measurement, overlay operations.)
- Neighbourhood operations, (search, topographic operations, contour generation etc.)
- Connectivity operations, (contiguity measurements, buffering, networking etc.)
- Output formatting, (map annotation, text labels, etc.)

For the above operations, firstly the geometric and the thematic characteristics of objects and their spatial relationships should be integrated in a database.

Geometry defines the location and shape of the 3D objects in space. On the other hand, topology allows to take into account various relationships between the objects in the space. Examples for these relationships can be given as adjacency, inclusion, overlapping etc. Here, topology can be considered as complement of the geometry. Topological relations of objects are deduced from their geometry. For example, in order to find adjacency of objects, one should look at the geometry and find the adjacent points, edges, facet surfaces etc. This operation requires expensive searches, computations and comparisons in the geometric domain. To overcome this costly expensive operation processes, it is a good solution approach to store topological relations in an explicit form. Thus, for one time expensive operations are performed and at the result of these operations, the constructed or found topological relations are stored in to a database for further query and analyse operations without repeating the complex processes. Here, the requirement of a database to be used for storing of the topological relations, can be seen easily. But on the other hand, to visualize 3D data on the computer's graphics hardware (on the screen), geometry of the spatial objects should be presented to the graphic displays in the proper rendering primitives, which are supported by OpenGL standards. For the rendering of the 3D data, geometry information should also be represented effectively in the database.

Research in the GIS community is trying to work out a conceptual model capable of integrating geometric and thematic characteristics of objects and mutual spatial relationships. These models can be considered as an explicit description of cells (or objects), (Zlatanova, 1999). In the past, several methods for 3D object description have been investigated. These models can be grossly subdivided into wire-frame, B-Rep, VBR, cell-decomposition, FBR, CSG, (Gruen and Wang, 1999). Another efficient model is Molenaar's 3D topological vector structure, the formal data structure (FDS), (Gruen

and Wang, 1999; Molenaar, 1992). In the topological model for geometric elements, are used as the elementary data types of geometrical part, and the relationships between geometric elements and object types are determined by five rules. FDS is a complete vector structure and shows powerful specifications for the presentation of topology, position and shape. The topology representation is benefical for topology queries among the geometric elements of an object, but to generate a FDS description of 3D objects is difficult, because FDS requires topological definitions between point and arc, arc and edge, edge and surface, edge and edge, etc., (Gruen and Wang, 1999). To overcome this situation, a modified facet model based on B-Rep, V3D has been proposed in (Gruen and Wang, 1999). This data model can be generated easily and is designed to adapt images as well.

Our visualization and query tool for 3D city models, presented in this paper depends on the V3D model. This model can easily be represented by relational database approaches by using relational tables. In the next chapter, we give description of geometric and topological modelling structure of objects in our tool which is still under development.

2. IMPLEMENTATION OF DATA STRUCTURE IN OUR 3D VISUALIZATION TOOL

In our tool, as being primitive geometric objects we use point, line, surface and body objects (houseObject / solid object) as defined in V3D model. A body object is decomposed into surface objects (facets), a surface object is decomposed into line (edge) objects and line object is decomposed into point (vertex) objects hierarchically. And these are all defined as object classes. We define a house object as follows:

```
struct houseObject{
        int houseId;
         int wallRed,wallGreen,wallBlue;
         int roofRed,roofGreen,roofBlue;
         float labelX;
         float labelY;
         float labelZ;
         Points* housePoints;
         CellArray *wallCells;
         CellArray *roofCells;
         PolyData *houseWalls;
         PolyData *houseRoofs;
         PolyDataMapper *houseWallMapper;
         PolyDataMapper *houseRoofMapper;
         Actor *houseWallActor;
         Actor *houseRoofActor;
         struct houseObject* nextHouse;
         struct houseObject* prevHouse;
         };
```

As seen in the structure definition, one house object contains all of the geometrical and topological information implicitly as a whole. In the structure, "Points" variable represents all of the vertex points of a house with point ids and point coordinates. "CellArray" variable represent all of the cells (faces) of the house with cell ids. PolyData uses points and cells and forms the houseObject in a polygonal model which is obtained with the integration of the points and cells. As being optional it is provided to construct the cell links with the member function of the class PolyData. Thus, PolyData object will represent the links of the cells (i.e. linking topology of the cells) too. PolyDataMapper is used to prepare the PolyData for rendering purposes. Actor is used to manage rendering operations of the mapper such as rotation, scaling, coloring etc. Finally, actors are rendered in the rendering window on the screen. At this point, geometry and topolgy of the house object with its geometric primitives haven't been stored explicitly in a database yet. But however, at this point, it is possible to render house object and query some geometric properties of the houseObject's geometric primitives. For example, coordinates and id number of a picked/selected point with mouse from the render window, can be shown on the screen. As the same way, a picked cell's id and its point ids can be listed too, and so etc. For the explicit storage of the topological relations between edges, faces, etc., user should ask the software to "build topology". This will be explained in detail in the following sections.

"houseId" member variable is the id code of a house object. This id number is used to link an object attribute table with thematic information related to house object, (e.g. surface area, volume of house, etc.). "labelX, labelY, labelZ" variables define the center of gravity of the bounding box of the houseObject. This label coordinate values are used for picking (selection) operations on the screen and finding related record(s) in the related attribute table(s). When one picks one or more house objects on the screen, bounding box(es) of the picked object(s) is/are returned with min and max coordinate values of the corner points. And label coordinates of the selected objects are computed and corresponding houseObject is searched in the memory and when found, houseId of the found object is handled. With this information, link between object geometry and attribute tables is constructed. Label coordinates serve as opposite work of houseId. In the house object structure, wall and roof geometry and topology are defined seperately and together. Thus, it is possible to make analysis and queries related to roofs and in a flexible way. Color components are used to represent the selection color of the house object. When a house object is selected it changes color with selection color defined by the user, and when unselected, it takes its original color properties.

With the above structure, it is possible to selecet indivudial parts of house, such as edges, faces, vertex points etc. Thus this property facilitates the indivual queries of the decomposed cells. Deletion of an existing part or adding new components are also easy with this description. CellArray class instance variable represents the definition of each facet cells and their topological links.

In the memory, we organize all house objects in a double sided binary tree. In Figure 1, we give the organisation of the house objects in a binary tree model.

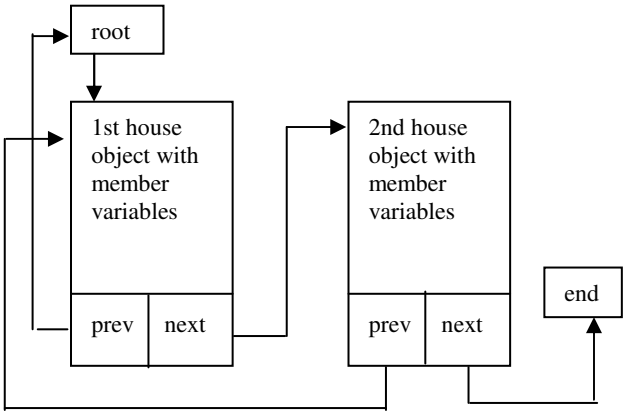

Figure 1. Organisation of house object in a binary tree

In Figure 1, each node of binary tree corresponds a house object and each node shows where the previous and next houses are in the memory, by using prevHouse and nextHouse pointers. With these pointers, each nodes are linked each other for the fast search tasks. In the above tree, a new house object can easily be added at the end of the tree by allocating necessary memory and linking this new memory area with the previous house object.

Now, we can give the detailed description of one individual house object by defining both its geometry and topology. As mentioned before, points, cell arrays, poly data, mappers and actors are used. Mapper and actor is used for 3D visualization purposes, in order to present the data to graphical environment for rendering purposes and well known from computer graphics and 3D visualization libraries such as OpenGL, and visualization toolkit, (Scroeder, 1998). Relations between points and cells are shown in Figure 2.

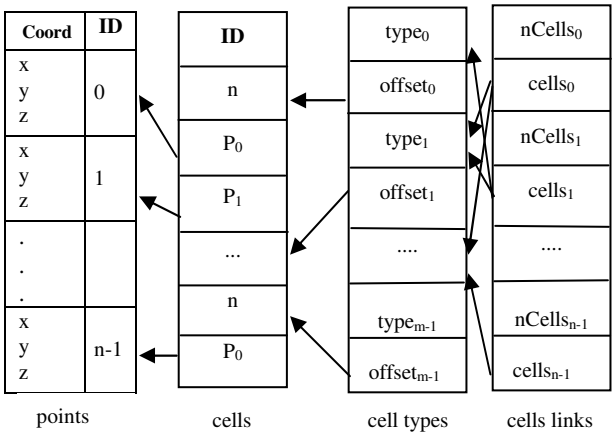

Figure 2. Implicit representation of geometry and topology

By using cell types, random access to each cell is provided. Above structure can easily be converted to relational model in our tool. For this purpose, from the main menu, "build topology" command should be selected. In this case, the topological relations of objects are written to relational tables. 3D City-VAQS, supports dBase and Paradox relational database management systems. In Figure 3, firstly we give the general representation of geometry and topology of 3D objects used in 3D-City VAQS and then in Figure 4, we give the relational database model representation.

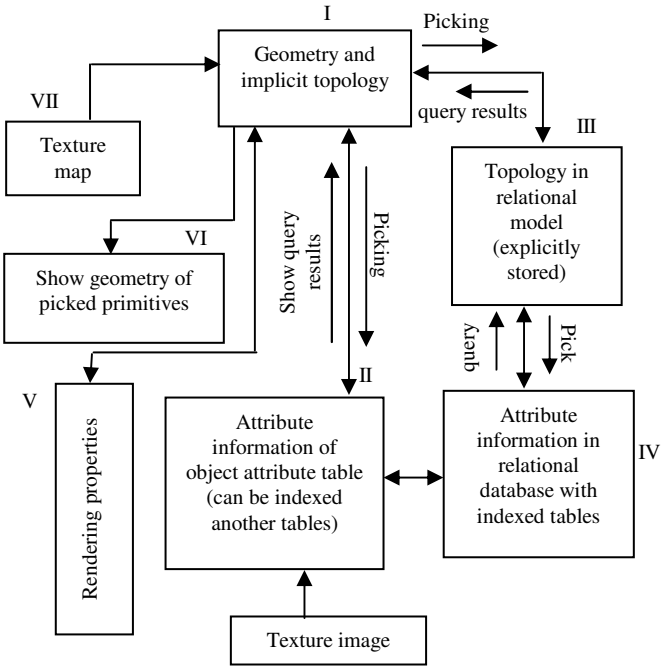

Figure 3. General representation of geometry and topology

With the representation shown in Figure 3, it is possible to manage the visualization tasks and database tasks in a flexible manner (independing and depending to each other). The box with number i in the figure represents the houseObject entities in the binary tree of visualization memory. This box wholly contains all of the member variables of houseObject entity. At this stage, even if the relational tables aren't created, the operations which are shown in the boxes with the numbers v, vi and vii are ready and effectively can be performed in the 3D visualization pipeline. I.e. this means, one can query the geometry of houseObject's geometric features such as point coordinates, id of any cell (face, edge, surface etc.) by picking on it with mouse on the render window. Here at this stage, only the volume and area of the houseObject can not be listed, since it requires additional volume and area computations and these computations are performed during the "building topology" process. Here, rendering properties of the objects can also be changed. For example; color, transparency, illumination conditions such as light, shading, reflection etc. Again at this point, only artificial textures can be mapped on to houseObject. Photorealistic texture mapping requires computation of texture coordinates, so photorealistic texture mapping can not be performed at this stage. It can be performed after building topology and with photogrammetric computations for obtaining texture coordinates.

During topology building, object attribute table (OAT) is created. Each column of the OAT represents one attribute feature and each row represents one houseObject. In OAT table, "InternalId", "UserId", "FacetArea" and "Volume" items are created by the software automatically.

"InternalId" item has values of the houseId of each houseObject kept in the binary tree. With this item, OAT is linked to geometric data. "UserId" is an empty item and used by the user to link OAT to any other data table which contains further attribute information about houseObject such as owner of the house, type of house etc. To manage relational database, 3D-City VAQS provides a database management system module, which is based on dBase and Paradox. Database management of the tool is explained in the chapter 3, in detail. In spite of OAT, optionally, attribute tables of the geometric primitives of the houseObject can also be created. These tables are also in relational database model. With relational tables, geometric queries can be performed and attribute information about each geometric primitive can be associated. For example, one can want to associate attribute information to individual faces of the house rather than the whole houseObject. For instance, one can want to know area of each faces (walls), chimney, roof etc. In this case, optionally the procedures iii and iv in Figure 3 can be performed. Thus a very effective and flexible data association is provided for 3D GIS. The operations of the relational model are based on V3D data structure as explained in (Gruen and Wang, 1999). In Figure 4, our relational data model representation is seen.

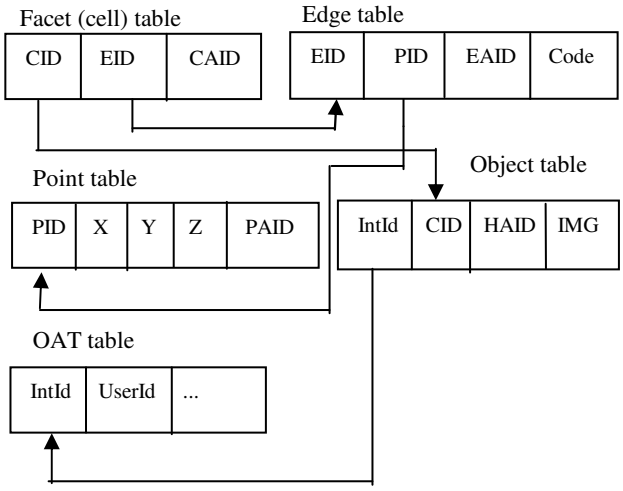

Figure 4. Relational data model representation

All tables in Figure 4 can be indexed and linked to each other as shown in figure. These tables can be indexed and linked to another new attribute table with attribute ids. Meaning of the items of tables in Figure 4 are as follows:

CID: cell id, EID: Edge id, CAID: cell attribute id, PID: point id, Code: point code (start point, end point), EAID: edge attribute table, x,y,z: coordinates of a points with point id, PAID: point attribute id, IntId: houseId, IMG: texture image. Hotlink image item can also added to object table as being optional by the user.

In the next chapter, we explain the database management model of the 3D-City VAQS and in the chapter 4 we briefly mention the data set which our tool uses.

3. DATABASE MANAGEMENT IN 3D-CITY VAQS

Our visualization tool provides a relational database management system which uses dBase and Paradox files. For topology generation, 3D data used in the project should be classified in different layers. During topology building, program asks user to define a theme and dataset where the topology information will be stored in the harddisk. Each theme bears their own topological relations in their own dataset directory independent from other themes. Not only the topology is provided for houseObjects, but also provided for DTM and other vector data too. But in this paper, since the most complex geometry belongs to 3D solid objects, topology management has been explained in detail for houses. When topology is created, related attribute table is automatically created and linked to geometry. Details have been explained in the previous section. In Figure 5, buildings on the DTM model and their attribute table is shown. In 3D-City VAQS, as explained in the previous section further attribute tables can be added to project and by using common item these tables can be linked to each other for the SQL queries. In Figure 5, the Owners table added to project to link the owner information of the buildings with object attribute table.

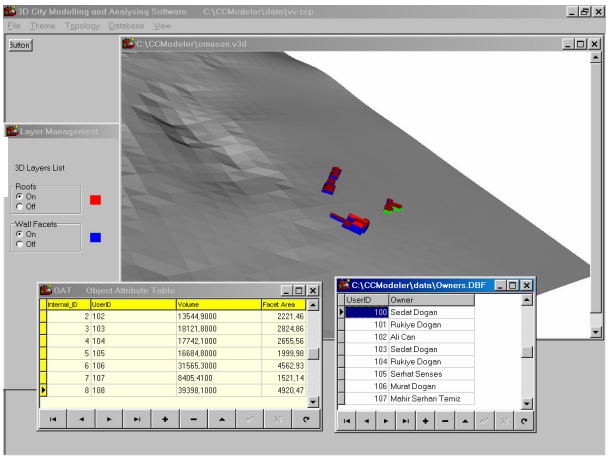

Figure 5. Object attribute table and owners table.

In the figure, these tables aren't linked. To link each table, on the database menu, link option is selected. Then, software asks user to select master and slave tables and to select index item to be used for linking. Here "UserID" items are used for linking and in Figure 6, link is shown.

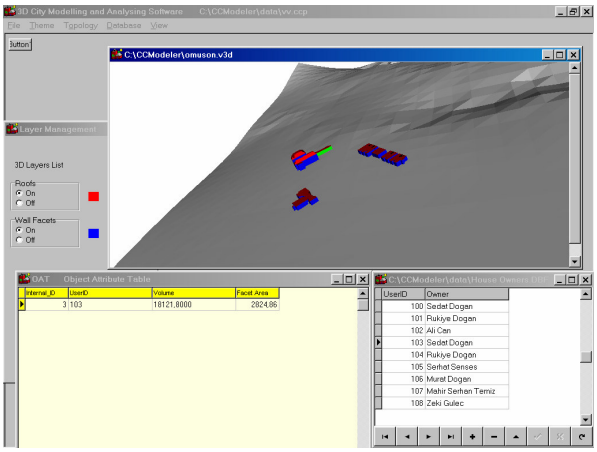

Figure 6. Linking attribute tables.

In Figure 6, owners table is master and the object attribute table is slave. As seen in the figure, when the user selects a record from master table, only the related record(s) are selected and shown in the slave table. In the figure, since the slave table is OAT table, and since it is linked to geometry, corresponding building in the render window is also shown with a different selection color. Here selection color is green as seen in the figure. With this link, owner table (owner information) is linked to geometry via OAT table. The link can be constructed double sided too. This means that, when the user pick an object on the render window, its corresponding records in the related tables are also been selected automatically. We provide SQL queries on every table added to project. For example, one can want to see a particular owner's building on the screen. In this case user should write the required SQL query script on the query window. Then, according to the query result, intended records are selected with their corresponding geometric objects.

One can easily create a new data table or modify existing one(s). Tables can be created with another software which support dBase and Paradox too. Externally prepared data tables can easily be added to project.

One can query any object by picking it with mouse on the render window. In Figure 7, a picked object's information taken from attribute table is shown in an id window.

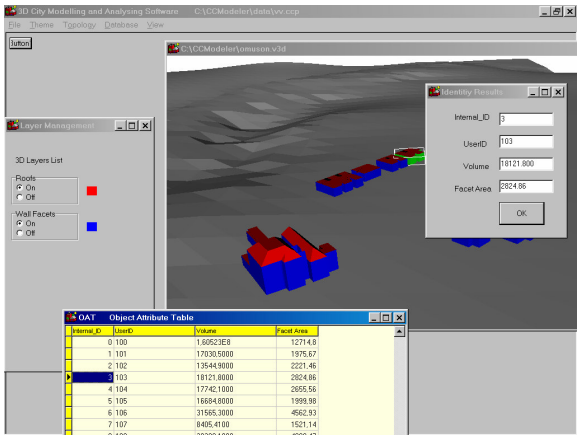

Figure 7. Query with picking object

On the id window, items of the OAT table are shown. When new items are added, they are automatically shown too.

4. DATA SET USED IN 3D-CITY VAQS

For the terrain representation, our tool uses DTM model. Regular grids or TIN can be used. Our tool can import and export ArcView ascii DTM file and USGS DTM files. It can also open x,y,z point dat files and creates DTM with regular grids or TIN. For TIN generation we use Delaunay triangulation. Photorealistic texture images (orthophoto, satellite images etc.) can be mapped on to DTM or buildings. Our tool can project grids or triangles of 2.5 D DTM to a 2D plane. Thus 2D GIS analysis can be performed by using 2D projections. We are still working on spatial 3D GIS analysis.

Our tool 3D-City VAQS can represent every kind of 3D object. This tool can open and visualize V3D files too. In our tool for realistic texture mapping, images can also be used.

5. CONCLUSION

With our tool, visualization and query of 3D city models can be performed effectively. But still it is under development to provide 3D GIS analysis. We can perform some 3D analysis such as buffer, clip, etc. at this point. But overlay analysis are too time consuming without using set theory together with geometry. For this purpose we are motivated on the description of 3D GIS analysis in a formalist manner. At this point, we can perform overlay analysis with complex geometric computations but still we can not use 3D topological relations effectively and so these computations are too time consuming. In order to overcome this problem, we are studying to combine topology and geometry by the means of set theory.

6. REFERENCES

Aronoff, S., 1995. *Geographic information systems:a management perspective*, WDL publications, Ottava, Canada.

Goodchild, M., 1987. A spatial analytical perspective on geographical information systems. *International Journal of GIS*, 1(4), pp. 327-334.

Gruen, A., Wang, X., 1999. Cyber city spatial information system (CC-SIS): A new concept for the management of 3D city models in a hybrid GIS. *The 20th Asian Conference on Remote Sensing,* HongKong, China. http://www.geod.ethz.ch/p02/general /persons/AG_pub/cc-sis_acrs.pdf (accessed, September, 2003).

Molenaar, M., 1992. A topology for 3D vector maps. *ITC Journal 1*, pp. 25-33

Schroeder ,W., et al., 1998. *The VisualizationToolkit An* Object *Oriented Approach to 3D Graphics*. Prentice Hall PTR, New Jersey.

Zlatanova, S., 1999. 3D GIS for urban development. PhD. Thesis.http://www.gdmc.nl/zlatanova/PhDThesis/html/ content. html (accessed January 2004).